FOUNDATIONS OF DISCRETE MATHEMATICS WITH ALGORITHMS AND PROGRAMMING

SRIRAMAN SRIDHARAN
R. BALAKRISHNAN

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Foundations of Discrete Mathematics with Algorithms and Programming

Foundations of Discrete Mathematics with Algorithms and Programming

Sriraman Sridharan
Laboratoire LAMPS
Département de Mathématiques et d'Informatique
Université de Perpignan Via Domitia
Perpignan
FRANCE

R. Balakrishnan
Bharathidasan University
Tiruchirappalli
Tamil Nadu
INDIA

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper
Version Date: 20180813

International Standard Book Number-13: 978-0-8153-7848-8 (Hardback)

Library of Congress Cataloging-in-Publication Data

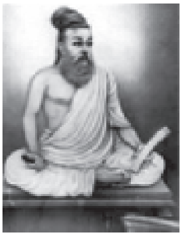
Names: Sridharan, Sriraman, author. | Balakrishnan, R. (Rangaswami), author.
Title: Foundations of discrete mathematics with algorithms and programming /
Sriraman Sridharan, R. Balakrishnan.
Description: Boca Raton : Taylor & Francis, a CRC title, part of the Taylor &
Francis imprint, a member of the Taylor & Francis Group, the academic
division of T&F Informa, plc, 2019. | Includes bibliographical references
and index.
Identifiers: LCCN 2018018635| ISBN 9780815378488 (hardback : acid-free paper)
| ISBN 9781351019149 (ebook)
Subjects: LCSH: Computer science--Mathematics.
Classification: LCC QA76.9.M35 S725 2019 | DDC 004.01/51--dc23
LC record available at https://lccn.loc.gov/2018018635

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

*தொட்டனைத் தூறும் மணற்கேணி மாந்தர்க்குக்
கற்றனைத் தூறும் அறிவு.*

திருவள்ளுவர்



Translation:

*In sandy soil, when deep you delve, you reach the springs below;
The more you learn, the freer streams of wisdom flow.*

Tamil Poet Tiruvalluvar (500 BC)

The first author S. S. affectionately dedicates this book in memory of his parents:

Geeyar Sriraman
Padmavathi Sundaravaradhan

The second author R. B. dedicates this book in memory of

the Late Professor Jacob K. Goldhaber,

formerly of the University of Maryland, USA for all the good things in life the author learned from him, besides, of course, mathematics.

Contents

Preface

Discrete mathematics and programming courses are all-pervasive in computer science, mathematics, and engineering curricula. This book is intended for undergraduate/postgraduate students in computer science, mathematics and engineering. No specific computer science and mathematics background is assumed. The student with high school level mathematics, can use this book for self-study and reference. A glance at the table of contents will reveal that the book treats fundamentals of discrete mathematics, computer science and programming languages. A number of examples have been given to enhance the understanding of concepts. The programming languages used are Pascal and C. In the chapters on combinatorics, graph theory, algorithms and data structures, examples are given and then the relevant concept/ algorithm is abstracted from the examples. The aim of the book is to bring together the fundamentals of discrete mathematics, algorithms and programming for the student community.

The scope of the book

In Chapter 1, titled “Sets, Relations, and Functions,” the reader is given a review of the basic concepts of sets, relations and functions. Partially ordered sets, lattices and Boolean algebras are treated with examples. Next, cardinal numbers are introduced. The fact that $2^\alpha > \alpha$ for any cardinal number α is proved. The celebrated Schroder-Berstein theorem is established. The chapter ends with a section on lattices which describes modular lattices, distributive lattices and Boolean algebras.

In Chapter 2, titled "Combinatorics," we start from the elementary rules of counting, then study permutations and combinations, binomial coefficients, binomial theorem, multinomial coefficients, multinomial theorem, Stirling numbers, and Bell numbers. Next, the Principle of Inclusion and Exclusion (simple and weighted versions) and its application to number theory and the theory of permanents, generating function techniques and recurrence relations, Bernoulli numbers, Catalan numbers, and algorithms for generating all the subsets of a given finite set are studied.

Chapter 3 is titled "Basics of Number Theory." After recalling some elementary concepts, congruences, the complete system of residues, and the Chinese remainder theorem, lattice points visible from the origin are presented.

In Chapter 4, titled "Introduction to Graph Theory," we begin from directed graphs as binary relations. Elementary theorems concerning in-degrees, out-degrees, and degrees are presented. An informal and intuitive introduction to the Classes P, NP and NP-complete is given. Multigraphs and simple graphs are then treated along with the idea of degree sequences. Some special graphs are then introduced. Graph isomorphism is covered with simple results on enumeration of graphs. Various types of sub-graphs and f-factors are defined and the fundamental result of Tutte on the existence of an f-factor in a multigraph is stated. The Erdös-Gallai theorem on the degree sequence of graphs is given. Walks, paths, distance concepts, and connectedness are studied. Examples of the use of graphs to solve puzzles (like the Königsberg bridge problem and the five Pandava princes problem) are covered along with Eulerian circuits. Ramsey numbers are introduced and fundamental results involving these numbers are proved. Graph algebra is presented together with the concept of graph minors.

The concept of the algorithm is at the very heart of computer science. In Chapter 5, titled "Introduction to Algorithms and Data Structures," we begin with the notion of an algorithm and the classic example of the greatest common divisor algorithm (the granddad of all algorithms, according to Donald Knuth). We then introduce big oh, big omega, and big theta notations to study the complexity of an algorithm. Frequently occurring complexity functions along with the corresponding algorithms are stated. Next, an overview of a classical computer is presented. Fundamentals of programming are studied. These include variables, types, assignment, the conditional statement, loop statement (while, repeat, for), function, procedure, parameter passing mechanisms, recursion and how these programming concepts are implemented in the languages Pascal/C are explained with examples. Next, fundamental data structures are presented. These include the data model list, its representation as an array, and the linked list. Access-restricted fundamental data models like stack, queue, and circular queue are covered along with their representation as an array and a linked list. Examples of programs in C whose complexities are frequently occurring functions are written. These include binary search, quick sort, selection sort, matrix multiplication, tower of Brahma-Hanoi and permutation generation.

Chapter 6 is titled “Introduction to Logic and Probability.” In the first section, we study the concepts of statements, truth assignments to statements, implication (conditional statement), logically equivalent statements, truth tables, tautology, contradiction, valid arguments, arguments with fallacies (invalid arguments), and proof techniques in mathematics.

In the next section on probability, we study elementary probability theory. We introduce sample space, events, probability, random variables (discrete and continuous), conditional probability, independence of events, expectation, variance, and binomial and Poisson distributions.

Answers to even-numbered exercises are given. The reader is requested to give a reasonable effort to exercises, before looking at the solutions.

Use of the Book

Chapters 1 and 3 have been taught by the second author in various universities in India.

Chapters 2, 4, 5, and 6 have been taught by the first author at the following French universities: Université de Bordeaux I, Université de Pierre et Marie Curie and Université de Perpignan Via Domitia, Cour du Soir au CNAM (Conservatoire National des Arts et Métiers), and at the Institute IMERIR (Institut Méditerranéen de Recherche En Informatique et En Robotique).

The instructor has a great deal of flexibility in choosing the material from the book. For example, Chapters 1, 2, 3, and 6 will serve as a solid introductory course in combinatorics. Chapter 5 can be used for courses on data structures and programming, and Chapter 4 can serve as an accelerated course on the theory of graphs. A course on graph algorithms can use Chapters 4 and 5. Many illustrative examples have been given to enforce the understanding of the concepts. Every algorithm treated is illustrated with sample inputs. Algorithms are expressed as informal pseudocodes and programs in Pascal/C. Of course, these can be easily translated into any language like C++ or JAVA. Some concepts and results are found in two different chapters in different contexts. Exercises at the end of each chapter or section test the understanding of the concepts developed in the text.

We feel that the presentation of these chapters would go a long way in providing undergraduate students of mathematics, computer science and engineering, a solid foundation in discrete mathematics.

Acknowledgment

The authors thank S. Baskaran for going through the chapters which deal with algebra and number theory. We also thank A. Anuradha and R. Dhanalakshmi for their help in typesetting some of the chapters. Thanks are also due to N. Sridharan and A. Srinivasan for going through some chapters of the book and offering constructive comments and criticisms.

We take the opportunity to thank our institutions–Bharathidasan University, Tamil Nadu, India and Université de Perpignan Via Domitia, France—for their academic support. Our thanks are also due to the faculties in our departments whose encouragement proved vital in attaining our goal. We also thank the four anonymous reviewers for suggesting some changes on the

initial version of the book. Last but not least, we thank Aastha Sharma and Shikha Garg of CRC Press for their kind understanding of our problems and for the patience exhibited by them till our completion of the manuscript.

The first author expresses his deep gratitude to Professor K. R. Parthasarathy of the Indian Institute of Technology, Chennai, India for introducing him to graph theory and guiding his PhD thesis. A thousand thanks go to Professor Claude Berge, one of the greatest pioneers in graph theory and combinatorics, who invited him to CAMS (Centre d'Analyse Mathématique Sociale) and guided his doctoral work in Paris. Claude has been a source of immense inspiration to him. Special thanks are also due to Professors R. Balasubramanian (A. M. Jain College, TN, India), Philippe Chrétienne of Université de Pierre et Mairie Curie, Robert Cori of Université de Bordeaux I, Alain Fougère (UPVD), Michel Las Vergnas (CNRS), UFR secretary Mme. Fabien Pontramont (UPVD), Mircea Sofonea (Directeur de Laboratoire LAMPS, UPVD), N. Srinivasan (A. M. Jain College, TN, India), Michel Ventou (UPVD), Annick Truffert (Dean of the faculty of Sciences, UPVD). Many thanks are also due to my students of UPVD for their feedback. We are responsible for all the remaining errors, but still, we feel that the initial readers of our manuscript could have smoked out some more bugs. Though it is not in the Hindu custom to explicitly thank the family members, he would like to break this tradition and thank his wife Dr. Usha Sridharan and his daughters Ramapriya and Sripriya for putting up with an unusually long prolonged absence, as well as Dheeraj.

The entire book was composed using the TEX and LATEX systems developed by D. E. Knuth and L. Lamport, respectively.

The authors welcome corrections, comments and criticisms from readers which would be gratefully acknowledged. They can be sent by email to rbsri2018@gmail.com.

S. S.
R. B.

Perpignan, France
Tiruchirappalli, Tamil Nadu, India
February 2018

Chapter 1

Sets, Relations and Functions

> No solitary tree can pass on for a grove.
>
> *Proverb in Tamil*

> Seuls comptent les ensembles.
>
> *F. Braudel*

Summary

The basic operations of sets, namely union, intersection and complementation, are defined and De Morgan's laws are derived. Injective, surjective and bijective functions are then defined. This is followed by consideration of finite and infinite sets. This leads to the study of cardinal numbers of sets, and, in particular, to countable and uncountable sets. This is followed by a proof of the Schroder-Bernstein theorem. As a consequence, the cardinality of the power set $\mathcal{P}(X)$ of a set X is strictly greater than that of X.

The next major part of the chapter deals with partially ordered sets and lattices. This is followed by a study of distributive and modular lattices. The theorem that any modular lattice must contain the pentagonal lattice as a sublattice is then proved.

The last two sections of this chapter consider atoms in a lattice and Boolean algebras. Basic properties of Boolean algebras

are established. The chapter ends up with the Representation theorem for finite Boolean algebras which states that any finite Boolean algebra $\mathcal{B}$ may be thought of as the Boolean algebra $\mathcal{P}(S)$ defined on the set S of its atoms.

In this chapter, we recall some of the basic facts about sets, functions, relations and lattices. We are sure that the reader is already familiar with most of these, which are usually taught in high school algebra with the exception of lattices. We also assume that the reader is familiar with the basics of real and complex numbers.

1.1 Introduction

Intuitively, a *set* is a collection of objects viewed as a single entity. Objects are also referred to as elements or members or points of the set. Elements of a set are enclosed in braces. For example, the set A consisting of the members $1, 2, 3$ is written as

$$A = \{\,1, 2, 3\,\}.$$

The element $1 \in A$ (read "1 belongs to or is in the set A") and the element $4 \notin A$ (read "4 does not belong to or is not in A").

A set can be visualized as a box containing objects which are its elements, and the empty set is an empty box, written as $\emptyset$. The set E consisting of all even positive integers is

$$E = \{\,2, 4, 6, \dots\,\}.$$

Note that A is a finite set since the number of its elements (also called the cardinality of the set A), namely, $|A| = 3$ while E is an infinite set.

We do not repeat elements in a set and the order of the elements in a set is irrelevant. For example, $\{\,1, 2, 3\,\} = \{\,2, 1, 3\,\}$. If some elements are repeated in a set, then we speak of a multiset and the number of times an element is repeated is called its multiplicity. Two sets A and B are *equal*, denoted by $A = B$, if they consist of identical elements. We now study some operations on sets. Usually, sets are denoted by capital letters like A, B, etc., and the elements of the sets by small letters a, b, etc.

There are generally two ways of specifying sets. One way is to list its members inside braces, with elements separated by commas like $\{3, 6, 9\}$. The infinite set of all prime numbers can be written as $\{2, 3, 5, 7, 11, \ldots, \}$. Sometimes it is not possible to give an exhaustive list all members of the set. In this case, we specify a set by a *characterizing property* of a typical element as in the following example.

The set of all odd positive integers is written as

$$\{\, n \mid n \text{ is an odd integer } , n > 0 \,\}.$$

The vertical line | is read as "such that" and (,) as "and."

Venn diagrams and operations on sets

Sets are usually described by their Venn diagrams (see Figure 1.1). A Venn diagram is a geometrical representation of sets. In this pictorial representation, sets are represented by enclosed areas in the plane. If A and B are sets, then their union, $A \cup B$, is defined to be the set of all elements x such that $x \in A$ or $x \in B$; here "or" is used in the sense of and/or; their intersection, $A \cap B$ is defined to be the set of all elements x such that $x \in A$ and $x \in B$, that is, common to both A and B. In Figure 1.1, the set $A = \{1, 2, 3, 4, 5\}$ and the set $B = \{4, 5, 6, 7\}$ have been marked by their Venn diagrams. Here $A \cup B$ is the set $\{1, 2, 3, 4, 5, 6, 7\}$ while $A \cap B = \{4, 5\}$. The sets A and B are disjoint if they have no elements in common. If A and B are sets then $A \subseteq B$ if every

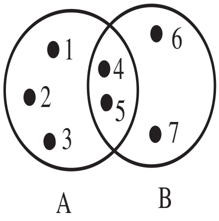


Figure 1.1: Venn diagram of sets

element of A is an element of B (that is, A is a subset of B, and A may be equal to B). In this case, the complement of A with respect to B is the set $B \setminus A$ consisting of all elements of B not belonging to A (the symbol $\setminus$ stands for set subtraction). The sets A and B are equal if $A \subseteq B$ and $B \subseteq A$. In other words, A and B have identical elements.

For example, the set of all prime numbers except 2, is a subset of the set of all odd positive integers. (Note that 2 is the only even prime number.) The symmetric difference of A and B is the set of those elements of A and B which are not in both A and B. It is denoted by $A \triangle B$. Hence

$$A \triangle B = (A \cup B) \setminus (A \cap B).$$

Again, for sets A and B of Figure 1.1, $A \triangle B = \{1, 2, 3, 6, 7\}$. Using Venn diagrams for the sets A and B, the set $A \triangle B$ can be pictorially given as in Figure 1.2 where the marked region denotes $A \triangle B$

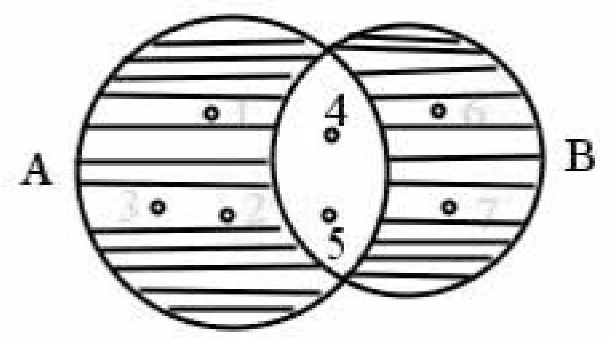


Figure 1.2: Set $A \triangle B$, the symmetric difference

We observe that

$$A \triangle B = (A \setminus B) \cup (B \setminus A),$$

and that

$$A \cup B = (A \triangle B) \cup (A \cap B).$$

In Figure 1.2, $A \setminus B = \{1, 2, 3\}$ and $B \setminus A = \{6, 7\}$.

The notion of union, intersection of two sets can be extended to any number of sets. To this end, we introduce:

Index Set: Given a non-empty set I, we say that I serves as an index set for the family of sets $\mathcal{F} = \{A_\alpha\}$, if to each $\alpha \in I$, there is a set A_α in the family of sets $\mathcal{F}$. The index set I can be any set, finite or infinite. Typically the index set is $I = \{\,1, 2, \ldots, n\,\}$.

For instance, if $A_1 = \{1, 2\}$, $A_2 = \{3, 4\}$ and $A_3 = \{5, 6, 7\}$, then $\{A_\alpha\}_{\alpha \in I}$ is a disjoint family of sets, where the index set $I = \{1, 2, 3\}$. Let $F = \{A_\alpha\}_{\alpha \in I}$ be a family of sets. Here for each $\alpha \in I$, there exists a set A_α of F. Assume that each A_α is a subset of a set X. Such a set X certainly exists since we can take $X = \underset{\alpha \in I}{\cup} A_\alpha$. For each $\alpha \in I$, denote by A'_α the complement X/A_α of A_α in X. We then have the celebrated laws of De Morgan.

Theorem 1.1.2 (De Morgan's laws)**:**
Let $\{A_\alpha\}_{\alpha \in I}$ be a family of subsets of a set X. Then

1. $(\cup_{\alpha \in I} A_\alpha)' = \cap_{\alpha \in I} A'_\alpha$, and
2. $(\cap_{\alpha \in I} A_\alpha)' = \cup_{\alpha \in I} A'_\alpha$.

Proof. We prove (i); the proof of (ii) is similar.

Let $x \in (\underset{\alpha \in I}{\cup} A_\alpha)'$. Then $x \notin \underset{\alpha \in I}{\cup} A_\alpha$, and therefore, $x \notin A_\alpha$, for each $\alpha \in I$. Hence $x \in A'_\alpha$ for each α, and consequently, $x \in \underset{\alpha \in I}{\cap} A'_\alpha$. Thus $(\underset{\alpha \in I}{\cup} A_\alpha)' \subseteq \underset{\alpha \in I}{\cap} A'_\alpha$. Conversely, assume that $x \in \underset{\alpha \in I}{\cap} A'_\alpha$. Then $x \in A'_\alpha$ for each $\alpha \in I$, and therefore, $x \notin A_\alpha$ for each $\alpha \in I$. Thus $x \notin \underset{\alpha \in I}{\cup} A_\alpha$, and hence $x \in (\underset{\alpha \in I}{\cup} A_\alpha)'$. Consequently, $\underset{\alpha \in I}{\cap} A'_\alpha \subseteq (\underset{\alpha \in I}{\cup} A_\alpha)'$. This proves (i). ■

Definition 1.1.1:
A family $\{A_\alpha\}_{\alpha \in I}$ is called a disjoint family of sets if whenever $\alpha \in I$, $\beta \in I$ and $\alpha \neq \beta$, we have $A_\alpha \cap A_\beta = \emptyset$.

1.2 Functions

A relation between two sets is defined by means of a function.

Definition 1.2.1:
A function (also called a map or mapping or a single-valued function) $f : A \to B$ from a set A to a set B is a rule by which to each $a \in A$, there is assigned a unique element $f(a) \in B$. $f(a)$ is called the image of a under f.

For example, if A is a set of students of a particular class, then for $a \in A$, if $f(a)$ denotes the height of a, then $f : A \to \mathbf{R}^+$ is the set of positive real numbers is a function.

Definition 1.2.2:
Two functions $f : A \to B$ and $g : A \to B$ are called equal if $f(a) = g(a)$ for each $a \in A$.

Definition 1.2.3:
If E is a subset of A, then the image of E under $f : A \to B$ is $\underset{a \in E}{\cup} \{f(a)\}$. It is denoted by $f(E)$.

Definition 1.2.4:
A function $f : A \to B$ is one-to-one (or 1–1 or injective) if for a_1 and a_2 in A, $f(a_1) = f(a_2)$ implies that $a_1 = a_2$.

Equivalently, the last definition means that $a_1 \neq a_2$ implies that $f(a_1) \neq f(a_2)$. Hence f is 1–1 iff distinct elements of A have distinct images in B under f.

As an example, let A denote the set of 1,000 students of a college A, and B the set of positive integers. For $a \in A$, let $f(a)$ denote the exam registration number of a. Then $f(a) \in B$. Clearly, f is 1–1. On the other hand if for the above sets A and B, $f(a)$ denotes the age of the student a, f is not 1–1, as the ages of 1,000 students in a restricted age group cannot all be distinct.

Definition 1.2.5:
A function $f : A \to B$ is called onto (or surjective) if for each $b \in B$, there exists at least one $a \in A$ with $f(a) = b$ (that is, the image $f(A) = B$).

For example, let A denote the set of integers $\mathbf{Z}$, and B the set of even integers. If $f : A \to B$ is defined by setting $f(a) = 2a$, then $f : A \to B$ is onto. Again, if $f : \mathbf{R} \to$ (set of non-negative reals) defined by $f(x) = x^2$, then f is onto but not 1–1.

Definition 1.2.6:
A function $f : A \to B$ is bijective (or is a bijection) if it is both 1–1 and onto.

The function $f : \mathbf{Z} \to 2\mathbf{Z}$ defined by $f(a) = 2a$ is bijective.

An injective (respectively surjective, bijective) mapping is referred to as an injection (respectively surjection, bijection).

Definition 1.2.7:
Let $f : A \to B$ and $g : B \to C$ be functions. The composition of g with f, denoted by $g \circ f$, is the function

$$g \circ f : A \to C$$

defined by $(g \circ f)(a) = g(f(a))$ for $a \in A$.

As an example, let $A = \mathbf{Z}$ denote the set of integers, $B = \mathbf{N} \cup \{0\}$, where $\mathbf{N}$ is the set of natural numbers $\{1, 2, \ldots\}$, and $C = \mathbf{N}$. If $f : A \to B$ is given by $f(a) = a^2$, $a \in A$, and $g : B \to C$ is defined by $g(b) = b + 1$, $b \in B$, then $h = g \circ f : A \to C$ is given by $h(a) = g(f(a)) = g(a^2) = a^2 + 1, a \in \mathbf{Z} = A$.

Definition 1.2.8:
Let $f : A \to B$ be a function. For $F \subseteq B$, the inverse image of F under f, denoted by $f^{-1}(F)$, is the set of all $a \in A$ with $f(a) \in F$. In symbols:

$$f^{-1}(F) = \{a \in A : f(a) \in F\}.$$

Let $f : S \longrightarrow \{1, 2, 3, 4, 5\}$ be the function from the set S of students of an elementary school to the set $\{1, 2, 3, 4, 5\}$ of grades, where for $s \in S$, $f(s)$ denotes the grade to which the student s belongs. If $F = \{1, 3\}$, then $f^{-1}(F)$ stands for the set of all students in S who are either in the 1^{st} grade or in the 3^{rd} grade.

Theorem 1.2.9:
Let $f : A \to B$, and $X_1, X_2 \subseteq A$ and $Y_1, Y_2 \subseteq B$. Then the following statements are true:

$$\begin{array}{rlcl} (i) & f(X_1 \cup X_2) & = & f(X_1) \cup f(X_2), \\ (ii) & f(X_1 \cap X_2) & \subseteq & f(X_1) \cap f(X_2), \\ (iii) & f^{-1}(Y_1 \cup Y_2) & = & f^{-1}(Y_1) \cup f^{-1}(Y_2), \quad \textit{and} \\ (iv) & f^{-1}(Y_1 \cap Y_2) & = & f^{-1}(Y_1) \cap f^{-1}(Y_2). \end{array}$$

Proof. We prove (iv). The proofs of the other statements are similar.

So assume that $a \in f^{-1}(Y_1 \cap Y_2)$, where $a \in A$. Then $f(a) \in Y_1 \cap Y_2$, and therefore, $f(a) \in Y_1$ and $f(a) \in Y_2$. Hence $a \in f^{-1}(Y_1)$ and $a \in f^{-1}(Y_2)$, and therefore, $a \in f^{-1}(Y_1) \cap f^{-1}(Y_2)$. The converse is proved just by retracing the steps. ■

Note that, in general, we may not have equality in (ii). Here is an example where equality does not hold well. Let $A = \{1, 2, 3, 4, 5\}$ and $B = \{6, 7, 8\}$. Let $f(1) = f(2) = 6$, $f(3) = f(4) = 7$, and $f(5) = 8$. Let $X_1 = \{1, 2, 4\}$ and $X_2 = \{2, 3, 5\}$. Then $X_1 \cap X_2 = \{2\}$, and so, $f(X_1 \cap X_2) = \{f(2)\} = \{6\}$. However, $f(X_1) = \{6, 7\}$, and $f(X_2) = \{6, 7, 8\}$. Therefore $f(X_1) \cap f(X_2) = \{6, 7\} \neq f(X_1 \cap X_2)$.

We next define a family of elements and a sequence of elements in a set X.

Definition 1.2.10:
A family $\{x_i\}_{i \in I}$ of elements x_i in a set X is a map $x : I \to X$, where for each $i \in I$, $x(i) = x_i \in X$. I is the indexing set of the family (in other words, for each $i \in I$, there is an element $x_i \in X$ of the family).

DEFINITION 1.2.11:
A sequence $\{x_n\}_{n\in\mathbf{N}}$ of elements of X is a map $f : \mathbf{N} \to X$. In other words, a sequence in X is a family in X where the indexing set is the set $\mathbf{N}$ of natural numbers. For example, $\{2, 4, 6, \ldots\}$ is the sequence of even positive integers given by the map (or function) $f : \mathbf{N} \to \mathbf{N}$ with $f(n) = 2n$.

1.3 Equivalence Relations

DEFINITION 1.3.1:
The Cartesian product $X \times Y$ of two (not necessarily distinct) sets X and Y is the set of all ordered pairs $(x,\ y)$, where $x \in X$ and $y \in Y$. In symbols:

$$X \times Y = \{(x, y) : x \in X, y \in Y\}.$$

In the ordered pair (x, y), the order of x and y is important whereas the unordered pairs (x, y) and (y, x) are equal. As ordered pairs they are equal if and only if $x = y$. For instance, the pairs (1, 2) and (2, 1) are not equal as ordered pairs, while they are equal as unordered pairs.

DEFINITION 1.3.2:
A relation R on a set X is a subset of the Cartesian product $X \times X$.

If $(a, b) \in R$, then we say that b is related to a under R and we also denote this fact by aRb. For example, if $X = \{1, 2, 3\}$, the set $R = \{(1, 1), (1, 2), (2, 2)\}$ is a relation on X.

One of the important concepts in the realm of relations is the equivalence relation.

DEFINITION 1.3.3:
A relation R on a set X is an equivalence relation on X if

1. R is reflexive, that is, $(a, a) \in R$ for each $a \in X$,

2. R is symmetric, that is, if $(a, b) \in R$ then $(b, a) \in R$, and

3. R is transitive, that is, if $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$.

We denote by $[a]$ the set of elements of X which are related to a under R. In other words, $[a] = \{x \in X : (x, a) \in R\}$. $[a]$ is called the *equivalence class* defined by a by the relation R.

Example 1.3.4:

1. On the set $\mathbf{N}$ of positive integers, let aRb mean that $a|b$ (a is a divisor of b). Then R is reflexive and transitive but not symmetric.
2. On the set $\mathbf{N}$ of positive integers, set xRy iff x is prime to y. Then R is symmetric but neither reflexive nor transitive. (For instance, if $x = 5$ is prime to $y = 7$, and y is prime to $z = 55$, 5 is not prime to 5 and it is not prime to 55).

   It is clear that similar examples can be constructed.

Example 1.3.5 (Example of an equivalence relation):
On the set $\mathbf{Z}$ of integers (positive integers, negative integers and zero), set aRb iff $a - b$ is divisible by 5. Clearly R is an equivalence relation on $\mathbf{Z}$.

Definition 1.3.6:
A partition $\mathcal{P}$ of a set X is a collection $\mathcal{P}$ of nonvoid subsets of X whose union is X such that the intersection of any two distinct members of $\mathcal{P}$ is empty.

Theorem 1.3.7:
Any equivalence relation R on a set X induces a partition on X in a natural way.

Proof. Let $[x]$ denote the equivalence class defined by x. We show that the classes $[x]$, $x \in X$, define a partition on X. First of all, each x of X belongs to class $[x]$ since $(x.x) \in R$. Hence

$$X = \bigcup_{x \in X} [x].$$

We now show that if $(x, y) \notin R$, then $[x] \cap [y] = \phi$. Suppose on the contrary, $[x] \cap [y] \neq \phi$. Let $z \in [x] \cap [y]$. This means that $z \in [x]$ and $z \in [y]$; hence $(z, x) \in R$ and $(z, y) \in R$. This of course means that $(x, z) \in R$ and $(z, y) \in R$ and hence by transitivity of R, $(x, y) \in R$, a contradiction. Thus $\{[x] : x \in X\}$ forms a partition of X. ■

Example 1.3.8:
Let $X = \mathbf{Z}$, the set of integers, and let $(a, b) \in \mathbf{R}$ iff $a - b$ is a multiple of 5. Then clearly, $\mathbf{R}$ is an equivalence relation on $\mathbf{Z}$. The equivalence classes are:

$$\begin{aligned}
[0] =&\{\ldots, -10,\ -5,\ 0,\ 5,\ 10,\ \ldots\},\\
[1] =&\{\ldots, -9,\ -4,\ 1,\ 6,\ 11,\ \ldots\},\\
[2] =&\{\ldots, -8,\ -3,\ 2,\ 7,\ 12,\ \ldots\},\\
[3] =&\{\ldots, -7,\ -2,\ 3,\ 8,\ 13,\ \ldots\},\\
[4] =&\{\ldots, -6,\ -1,\ 4,\ 9,\ 14,\ \ldots\}.
\end{aligned}$$

Note that $[5]{=}[0]$, $[6]{=}[1]$, $[7]{=}[2]$ etc. Then the collection $\{[0], [1], [2], [3], [4]\}$ of equivalence classes forms a partition of $\mathbf{Z}$.

1.4 Finite and Infinite Sets

Definition 1.4.1:
Two sets are called equipotent if there exists a bijection between them. Equivalently, if A and B are two sets, then A is equipotent to B if there exists a bijection $\phi : A \to B$ from A onto B.

If $\phi : A \to B$ is a bijection from A to B, then $\phi^{-1} : B \to A$ is also a bijection. Again, if $\phi : A \to B$ and $\psi : B \to C$ are bijections, then $\psi \circ \phi : A \to C$ is also a bijection. Trivially, the identity map $i : A \to A$ defined by $i(a) = a$ for each $a \in A$, is a bijection on A. Hence if X is a nonvoid set, and $\mathcal{P}(X)$ is the power set of X, that is, the collection of all subsets of X, then for $A, B \in \mathcal{P}(X)$, if we set ARB iff there exists a bijection ϕ from A onto B, then R is an equivalence relation on $\mathcal{P}(X)$. Equipotent sets have the same "cardinal number" or "cardinality".

Let $\mathbf{N}_n$ denote the set $\{1, 2, \ldots, n\}$. $\mathbf{N}_n$ is called the initial segment of $\mathbf{N}$ defined with respect to n.

DEFINITION 1.4.2:
A set S is finite if S is equipotent to $\mathbf{N}_n$ for some positive integer n; otherwise S is called an infinite set.

If S is equipotent to $\mathbf{N}_n$, then the number of elements in it is n. Hence if $n \neq m$, $\mathbf{N}_n$ is not equipotent to $\mathbf{N}_m$. Consequently, no finite set can be equipotent to a proper subset of itself.

For instance, the set of trees in a garden is finite whereas the set $\mathbf{Z}$ of integers is infinite. Any subset of a finite set is finite and therefore any superset of an infinite set is infinite. (If S is an infinite set and $S \subseteq T$, then T must be infinite; otherwise, S being a subset of a finite set T, must be finite).

Theorem 1.4.3:
Let S be a finite set and $f : S \to S$. Then f is 1–1 iff f is onto.

Proof. Suppose $f : S \to S$ is 1–1, and $T = f(S)$. If $T \neq S$, as f is a bijection from S to $T\Big(\underset{\neq}{\subset} S\Big)$, S and T have the same number of elements, a contradiction to the fact that T is a proper subset of the finite set S. Hence f must be onto.

Conversely, assume that f is onto. If f is not 1–1, there exists at least one $s \in S$ having at least two preimages. For each $s \in S$ choose a preimage $s' \in S$ under f. Let S' be the set of all such s'. Clearly, if s and t are distinct elements of S, then $s' \neq t'$. Hence S' is a proper subset of S. Moreover, the function $\phi : S \to S'$ defined by $\phi(s) = s'$ is a bijection. Thus S is bijective with the proper subset S' of S, a contradiction to the fact that S is a finite set. ∎

Example 1.4.4:
We show by means of examples that the conclusions in Theorem 1.4.3 may not be true if S is an infinite set.

First, take $S = \mathbf{Z}$, the set of integers and $f : \mathbf{Z} \to \mathbf{Z}$ defined by $f(a) = 2a$. Clearly f is 1–1 but not onto (the image of f being the set of even integers). Next, let $\mathbf{R}$ be the set of real numbers, and let $f : \mathbf{R} \to \mathbf{R}$ be defined by

$$f(x) = \begin{cases} x - 1 & \text{if} \quad x > 0 \\ 0 & \text{if} \quad x = 0 \\ x + 1 & \text{if} \quad x < 0 \end{cases}$$

Clearly f is onto; however, f is not 1–1 since

$$f(1) = f(0) = f(-1) = 0. \qquad \blacksquare$$

Theorem 1.4.5:
The union of any two finite sets is finite.

Proof. First we show that the union of any two disjoint finite sets is finite. Let S and T be any two disjoint finite sets of cardinalities n and m respectively. Then S is equipotent to $\mathbf{N}_n$ and T equipotent to $\mathbf{N}_m = \{1, 2, \ldots, m\}$. Clearly T is also equipotent to the set $\{n+1, n+2, \ldots, n+m\}$. Hence $S \cup T$ is equipotent to $\{1, \ldots, n\} \cup \{n + 1, \ldots, n + m\} = \mathbf{N}_{n+m}$. Hence $S \cup T$ is also a finite set.

By induction it follows that the union of a disjoint family of a finite number of finite sets is finite.

We now show that the union of any two finite sets is finite. Let S and T be any two finite sets. Then $S \cup T$ is the union of the three pair-wise disjoint sets $S \backslash T$, $S \cap T$ and $T \backslash S$ and hence is finite (here $S \setminus T$ stands for the set of points of S which are not in T). $\blacksquare$

Corollary 1.4.6:
The union of any finite number of finite sets is finite.

Proof. By induction on the number of finite sets. $\blacksquare$

1.5 Cardinal Numbers of Sets

In this section, we briefly discuss the cardinal numbers of sets. Recall that a set A is equipotent to a set B if there exists a bijection f from A onto B, and that equipotence between members of a collection of sets $\mathcal{S}$ is an equivalence relation on $\mathcal{S}$.

As mentioned before, the sets in the same equivalence class are said to have the same cardinality or the cardinal number. Intuitively it must be clear that equipotent sets have the same "number" of elements. The cardinal number of any finite set is a positive integer, while the cardinal numbers of infinite sets are denoted by certain symbols. The cardinal number of the infinite set $\mathbf{N}$ (the set of positive integers) is denoted by $\aleph_0$ (aleph not). $\aleph$ is the first character of the Hebrew language.

Definition 1.5.1:
A set is called denumerable if it is equipotent to $\mathbf{N}$ (equivalently, if it has cardinal number $\aleph_0$). A set is countable if it is finite or denumerable. It is uncountable if it is not countable (clearly, any uncountable set must be infinite).

Lemma 1.5.3:
Every infinite set contains a denumerable subset.

Proof. Let X be an infinite set and let $x_1 \in X$. Then $X_1 = X \setminus \{x_1\}$ is an infinite subset of X (if not, $X = X_1 \cup \{x_1\}$ is a union of two finite subsets of X and therefore finite by Corollary 1.4.6). As X_1 is infinite, X_1 has an element x_2, and $X_1 \setminus \{x_2\} = X \setminus \{x_1, x_2\}$ is infinite. Suppose we have found out distinct elements $x_1, x_2, \ldots, x_n$ in X with $X_n = X \setminus \{x_1, \ldots, x_n\}$ infinite. Then there exists x_{n+1} in X_n so that $X_n \setminus \{x_{n+1}\}$ is infinite. By induction, there exists a denumerable subset $\{x_1, x_2, \ldots, x_n, x_{n+1}, \ldots\}$ of X. ■

Theorem 1.5.4:
A set is infinite iff it is equipotent to a proper subset of itself.

Proof. That no finite set can be equipotent to a proper subset of it has already been observed (Section 1.4).

Assume that X is infinite. Then, by Lemma 1.5.3, X contains a denumerable subset $X_0 = \{x_1, x_2, \ldots, \}$. Let

$$Y = (X \setminus X_0) \cup \{x_2, x_3, \ldots\} = X \setminus \{x_1\}.$$

Then the mapping $\phi : X \to Y$ defined by

$$\phi(x) = \begin{cases} x & \text{if} \quad x \in X \setminus X_0 \\ x_{n+1} & \text{if} \quad x = x_n, n \geq 1, \end{cases}$$

(so that $\phi(x_1) = x_2$, $\phi(x_2) = x_3$ and so on) is a 1–1 map of X onto Y, and therefore an equipotence (that is, a bijection). Thus X is equipotent to the proper subset $Y = X \setminus \{x\}$ of X. ■

Notation

We denote the cardinality of a set X by $|X|$.

If X is a set of, say, 17 elements and y is a set of 20 elements, then $|X| < |Y|$ and there exists a 1–1 mapping of X to Y. Conversely, if X and Y are finite sets and $|X| < |Y|$, then there exists a 1–1 map from X to Y. These ideas can be generalized to any two arbitrary sets.

Definition 1.5.5:
Let X and Y be any two sets. Then $|X| \leq |Y|$ iff there exists a 1–1 mapping from X to Y.

Suppose we have $|X| \leq |Y|$, and $|Y| \leq |X|$. If X and Y are finite sets, it is clear that X and Y have the same number of elements, that is, $|X| = |Y|$. The same result holds well even if X and Y are infinite sets. This result is known as the Schroder–Bernstein theorem.

Theorem 1.5.6 (Schroder–Bernstein)**:**
If X and Y are sets such that $|X| \leq |Y|$ and $|Y| \leq |X|$, then $|X| = |Y|$.

For the proof of Theorem 1.5.6, we need a lemma.

Lemma 1.5.7:
Let A be a set and A_1 and A_2 be subsets of A such that $A \supseteq A_1 \supseteq A_2$. If $|A| = |A_2|$, then $|A| = |A_1|$.

Proof. If A is a finite set, then $|A| = |A_2|$ gives that $A = A_2$, as A_2 is a subset of A. Hence $A = A_1 = A_2$, and therefore $|A| = |A_1|$.

So assume that A is an infinite set. $|A| = |A_2|$ means that there exists a bijection $\phi : A \to A_2$. Let $\phi(A_1) = A_3 \subseteq A_2$. We then have

$$A_1 \supseteq A_2 \supseteq A_3, \quad \text{and} \quad |A_1| = |A_3| \tag{1.1}$$

So starting with $A \supseteq A_1 \supseteq A_2$ and $|A| = |A_2|$, we get (1.1). Starting with (1.1) and using the same argument, we get

$$A_2 \supseteq A_3 \supseteq A_4 \quad \text{and} \quad |A_2| = |A_4|. \tag{1.2}$$

Note that the bijection from A_2 to A_4 is given by the same map ϕ. In this way, we get a sequence of sets

$$A \supseteq A_1 \supseteq A_2 \supseteq A_3 \supseteq \dots \tag{1.3}$$

with $|A| = |A_3|$, and $|A_i| = |A_{i+2}|$, for each $i \geq 1$. Moreover,

$$\begin{aligned} \big|A \setminus A_1\big| &= \big|A_2 \setminus A_3\big|, \\ \big|A_1 \setminus A_2\big| &= \big|A_3 \setminus A_4\big|, \\ \big|A_2 \setminus A_3\big| &= \big|A_4 \setminus A_5\big|, \end{aligned}$$

and so on (see Figure 1.3). Once again, the bijections are under the same map ϕ. Let $P = A \cap A_1 \cap A_2 \cap \dots$

$$\text{Then } A = (A \setminus A_1) \cup (A_1 \setminus A_2) \cup \ldots \cup P, \tag{1.4}$$

$$\text{and } A_1 = (A_1 \setminus A_2) \cup (A_2 \setminus A_3) \cup \ldots \cup P, \tag{1.5}$$

where the sets on the right are pairwise disjoint. Note that since $A \supseteq A_1 \supseteq A_2 \supseteq A_3 \supseteq \cdots$, is a decreasing sequence of sets, $\cap_{i\geq 1} A_i = A \cap (\cap_{i \geq 1} A_i)$ and both of these are equal to P. Now in the Equations 1.4 and 1.5, in the first two terms on the R.H.S. we observe that $|A \setminus A_1| = |A_2 \setminus A_3|$ while the term $|A_1 \setminus A_2|$ is common. In the next two terms of the two expressions on the

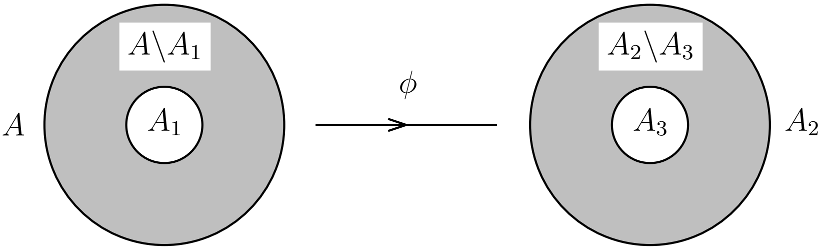


Figure 1.3: Illustration

R.H.S. $|A_2 \setminus A_3| = |A_4 \setminus A_5|$ while the term $|A_3 \setminus A_4|$ is common. In general, for each $n \geq 1, |A_{2n} \setminus A_{2n+1}| = |A_{2n+2} \setminus A_{2n+3}|$ while the term $|A_{2n+1} \setminus A_{2n+2}|$ is common to both the expressions on the R.H.S. As the terms are pairwise disjoint, this proves that $|A| = |A_1|$. ■

Proof of Schroder–Bernstein theorem. By hypothesis $|X| \leq |Y|$. Hence there exists a 1–1 map $\phi : X \to Y$. Let $\phi(X) = Y^* \subseteq Y$. Then

$$|X| = |Y^*| \tag{1.6}$$

Again by hypothesis, $|Y| \leq |X|$. Hence there exists a 1–1 map $\psi : Y \to X$. Let $\psi(Y) = X^* \subseteq X$. Then

$$|Y| = |X^*| \tag{1.7}$$

As $Y^* \subseteq Y$, $\psi(Y^*) = (\text{say})X^{**} \subseteq \psi(Y) = X^*$. Hence

$$|Y^*| = |X^{**}| \tag{1.8}$$

Thus $X \supseteq X^* \supseteq X^{**}$, and by Equations 1.6 and 1.8, $|X| = |X^{**}|$. This implies, by Lemma 1.5.7 that $|X| = |X^*|$. But by Equation 1.7, $|X^*| = |Y|$. Therefore $|X| = |Y|$. ■

1.6 Power Set of a Set

We recall the definition of the power set of a given set from Section 1.4.

Definition 1.6.1:
The power set $\mathcal{P}(X)$ of a set X is the set of all subsets of X.

For instance, if $X = \{1, 2, 3\}$, then

$$\mathcal{P}(X) = \big\{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{2, 3\}, \{3, 1\}, \{1, 2, 3\} = X\big\}$$

The empty set $\emptyset$ and the whole set X, being subsets of X, are elements of $\mathcal{P}(X)$. Now each subset S of X is uniquely defined by its characteristic function $\chi_S : X \to \{0, 1\}$ defined by

$$\chi_S = \begin{cases} 1 & \text{if} \quad s \in S \\ 0 & \text{if} \quad s \notin S. \end{cases}$$

Conversely, every function $f : X \to \{0, 1\}$ is the characteristic function of a unique subset S of X. Indeed, if

$$S = \{x \in X : \ f(x) = 1\},$$

then $f = \chi_S$.

Definition 1.6.2:
For sets X and Y, denote by Y^X, the set of all functions $f : X \to Y$.

Theorem 1.6.3:
$|X| < |\mathcal{P}(X)|$ for each nonvoid set X.

Theorem 1.6.3, implies that there exists a 1–1 function from X to $\mathcal{P}(X)$ but none from $\mathcal{P}(X)$ to X.

Proof. First of all, the mapping $f : X \to \mathcal{P}(X)$ defined by $f(x) = \{x\} \in \mathcal{P}(X)$ is clearly 1–1. Hence $|X| \leq |\mathcal{P}(X)|$. Next, suppose there exists a 1–1 map from $\mathcal{P}(X)$ to X. Then by the Schroder–Bernstein theorem, there exists a bijection $g : \mathcal{P}(X) \to X$. This means that for each element S of $\mathcal{P}(X)$, the mapping $g : S \to g(S) = x \in X$ is a bijection. Now the element x may or may not belong to S. Call $x \in X$ *ordinary* if $x \in S$, that is, x is a member of the subset S of X whose image under the map g is x. Otherwise, call x *extraordinary.* Let A be the subset of X consisting of all the extraordinary elements of X. Then $A \in \mathcal{P}(S)$. (Note: A may be the empty set; still, $A \in \mathcal{P}(S)$). Let $g(A) = a \in X$. Is a an ordinary element or an extraordinary element of X? Well, if we assume that a is ordinary, then $a \in A$; but then a is extraordinary as A is the set of extraordinary elements. Suppose we now assume that a is extraordinary; then $a \notin A$ and so a is an ordinary element of X, again a contradiction. These contradictions show that there exists no 1–1 mapping from $\mathcal{P}(X)$ to X $(X \neq \emptyset)$, and so $|\mathcal{P}(X)| > |X|$. ■

1.7 Exercises

1. State true or false and provide the reason for your answer:

   (a) Parallelism is an equivalence relation on the set of all lines in the plane.

   (b) Perpendicularity is an equivalence relation on the set of all lines in the plane.

   (c) A finite set can be equipotent to a proper subset of itself.

2. Prove the statements (i), (ii) and (iii) in Theorem 1.2.9.

3. Prove that the set $\mathbf{Z}$ of integers is denumerable.

4. (a) Prove that a denumerable union of denumerable sets is denumerable.

   (b) Prove that a countable union of denumerable sets is denumerable.

5. Prove that the set of all rational numbers is denumerable.

6. Let $n \in \mathbf{N}$, and $M_n = \{m \in \mathbf{N} : m$ is a positive multiple of n i.e., $m = an$, where $a \in \mathbf{N}\}$. Find

   (a) $\bigcup\limits_{n \in \mathbf{N}} M_n$.

(b) $M_{n_1} \cap M_{n_2}$.

(c) $\bigcap\limits_{n \in \mathbf{N}} M_n$.

(d) $\bigcup\limits_{p=\text{a prime}} M_p$.

7. Let $f : A \to B$, and $g : B \to C$ be functions.

   Prove:

   (a) If f and g are 1–1, then so is $g \circ f$.

   (b) If f and g are onto, then so is $g \circ f$.

   (c) If $g \circ f$ is 1–1, then f is 1–1.

   (d) If $g \circ f$ is onto, then g is onto.

8. Give an example of a relation which is
   (i) reflexive and symmetric but not transitive
   (ii) reflexive and transitive but not symmetric.

9. Does there exist a relation which is not reflexive but both symmetric and transitive?

10. Give a detailed proof of Corollary 1.4.6.

11. Let X be the set of all ordered pairs (a, b) of integers with $b \neq 0$. Set $(a, b) \sim (c, d)$ in X iff $ad = bc$. Prove that $\sim$ is an equivalence relation on X. What is the class to which (1,2) belongs?

1.8 Partially Ordered Sets

Definition 1.8.1:
A relation R on a set X is called antisymmetric if, for a, $b \in X$, $(a, b) \in R$ and $(b, a) \in R$ together imply that $a = b$.

For instance, in the relation R defined on $\mathbf{N}$, the set of natural numbers, setting that "$(a, b) \in R$ iff $a|b$ (a divides b)" is an antisymmetric relation. However, the same relation defined on $\mathbf{Z}^\star = \mathbf{Z}/\{0\}$, the set of nonzero integers, is not antisymmetric. For instance, $5|(-5)$ and $(-5)|5$ but $5 \neq -5$.

Definition 1.8.2:
A relation R on a set X is called a partial order on X if it is (i) Reflexive, (ii) Antisymmetric and (iii) Transitive.

A partially ordered set is a set with a partial order defined on it.

Examples

1. Let R be defined on the set $\mathbf{N}$ by setting aRb iff $a|b$. Then R is a partial order on $\mathbf{N}$.

2. Let X be a nonempty set. Define a relation R on $\mathcal{P}(X)$ by setting ARB in $\mathcal{P}(X)$ iff $A \subseteq B$. Then $\mathcal{P}(X)$ is a partially ordered set with respect to the above partial order.

It is customary to denote a general partial order on a set X by "$\leq$." We then say that $(X, \leq)$ is a *partially ordered set* or a *poset* in short.

Every poset $(X, \leq)$ can be represented pictorially by means of its *Hasse diagram*. This diagram is drawn in the plane by taking the elements of X as points of the plane and representing the fact that $a < b$ in X by placing b above a and joining a and b by a line segment. If $a < b$ and $b < c$, then certainly, $a < c$ but then we do not join a and c just to avoid cluttering of lines in the Hasse diagram.

As an example, take $S = \{1, 2, 3\}$ and $X = \mathcal{P}(S)$, the power set of S, and $\subseteq$ to stand for "$\leq$". Figure 1.4 gives the Hasse diagram of $(X, \leq)$. Note that $\emptyset \leq \{1, 2\}$ since $\emptyset$ is a subset of $\{1,2\}$. However, we have not drawn a line between $\emptyset$ and $\{1, 2\}$ but a (broken) line exists from $\emptyset$ to $\{1, 2\}$ via $\{1\}$ or $\{2\}$. When there is no relation between a and b of X, both a and b can appear at the same horizontal level.

Definition 1.8.3:
A partial order "$\leq$" on X is a total order (or linear order) if for any two elements a and b of X, either $a \leq b$ or $b \leq a$ holds.

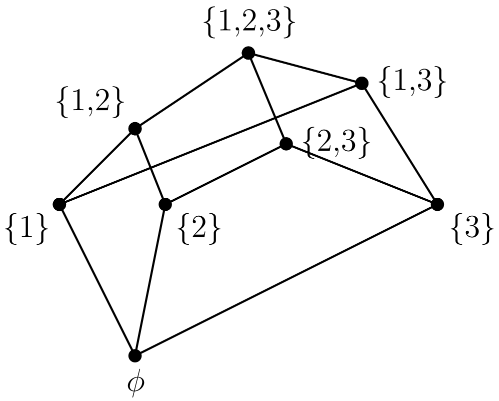


Figure 1.4: The Hasse diagram of $\mathcal{P}(\{1,2,3\})$

For instance, if $X = \{1, 2, 3, 4\}$ and "$\leq$" is the usual "less than or equal to" then $(X, \leq)$ is a totally ordered set since any two elements of X are comparable, that is, for any two elements a, b of X, either $a \leq b$ or $b \leq a$. The Hasse diagram of $(X, \leq)$ is given in Figure 1.5.

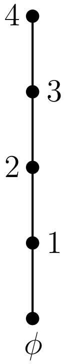


Figure 1.5: Hasse diagram of $(X, \leq)$, $X = \{1, 2, 3, 4\}$

The Hasse diagrams of all lattices with at most five elements are given in Figure 1.6.

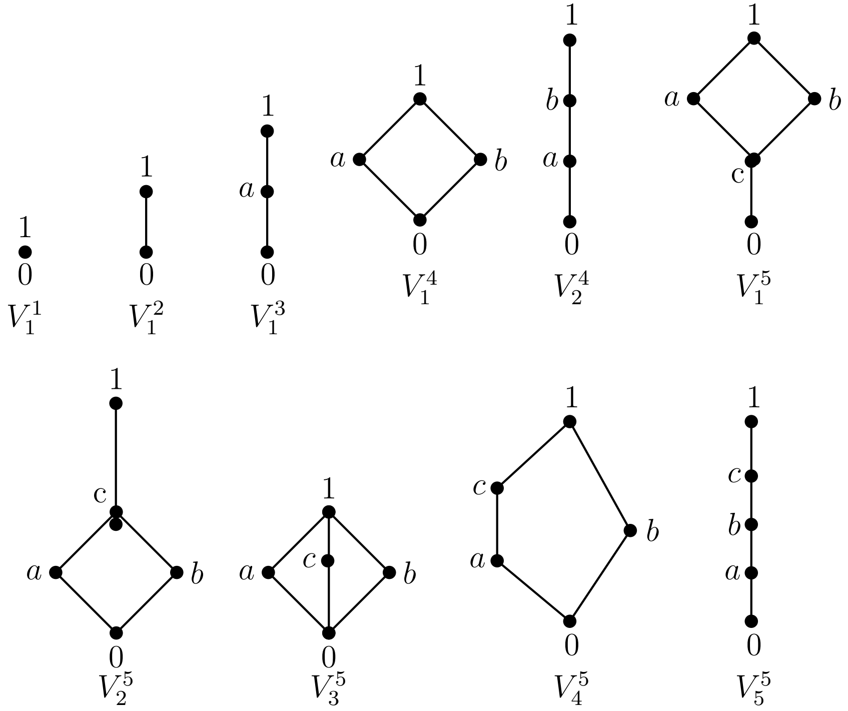


Figure 1.6: Hasse diagrams of all lattices with at most five elements. V_p^q stands for the q-th lattice in the set of all lattices with p elements (in the above order).

If S has at least two elements, then $(\mathcal{P}(S), \leq)$ is not a totally ordered set. Indeed, if $a, b \in S$, then {a} and {b} are incomparable (under $\subseteq$) elements of $\mathcal{P}(S)$.

Definition 1.8.4 (Converse relation):
If $f : A \longrightarrow B$ is a relation, the relation $f^{-1} : B \longrightarrow A$ is called the converse of f provided that $(b,\, a) \in f^{-1}$ iff $(a,\, b) \in f$.

We note that if $(X, \leq)$ is a poset, then $(X, \geq)$ is also a poset. Here $a \geq b$ in $(X, \geq)$ is defined by $b \leq a$ in $(X, \leq)$.

Definition 1.8.5:
Let $(X, \leq)$ be a poset.

1. a is called a greatest element of the poset if $x \leq a$ for each

$x \in X$. An element $b \in X$ is called a smallest element if $b \leq x$ for each $x \in X$.

If a and a' are greatest elements of X, by definition, $a \leq a'$ and $a' \leq a$, and so by antisymmetry $a = a'$. Thus a greatest (smallest) element, if it exists, is unique. The greatest and least elements of a poset, whenever they exist, are denoted by 1 and 0 respectively. They are called the universal elements of the poset.

2. An element $a \in X$ is called a *minimal element* of X if there exists no element $c \in X$ such that $c < a$ (that is, $c \leq a$ and $c \neq a$). $b \in X$ is a *maximal element* of X if there exists no element c of X such that $c > b$ (that is, $b \leq c, b \neq c$).

Clearly, the greatest element of a poset is a maximal element and the least element a minimal element.

Example 1.8.6:
Let $(X, \subseteq)$ be the poset where $X = \{\{1\}, \{2\}, \{1,2\}, \{2,3\}, \{1,2,3\}\}$. In X, $\{1\}$, $\{2\}$ are minimal elements, $\{1, 2, 3\}$ is the greatest element (and the only maximal element) but there is no smallest element.

Definition 1.8.7:
Let $(X, \leq)$ be a poset and $Y \subseteq X$.

1. $x \in X$ is an upper bound for Y if $y \leq x$ for all $y \in Y$.

2. $x \in X$ is a lower bound for Y if $x \leq y$ for all $y \in Y$.

3. The infimum of Y is the greatest lower bound of Y, if it exists. It is denoted by $\inf Y$.

4. The supremum of Y is the least upper bound of Y, if it exists. It is denoted by $\sup Y$.

Example 1.8.8:
If $X = [0, 1]$ and $\leq$ stands for the usual ordering in the reals, then

1 is the supremum of X and 0 is the infimum of X. Instead, if we take $X = (0, 1)$, X has neither an infimum nor a supremum in X. Here we have taken $Y = X$. However, if $X = \mathbf{R}$ and $Y = (0, 1)$, then 1 and 0 are the supremum and infimum of Y respectively. Note that the supremum and infimum of Y, namely, 1 and 0, do not belong to Y.

1.9 Lattices

We now define a lattice.

Definition 1.9.1:
A lattice $L = (L, \wedge, \vee)$ is a nonempty set L together with two binary operations $\wedge$ (called meet or intersection or product) and $\vee$ (called join or union or sum) that satisfy the following axioms:

For all a, b, $c \in L$,

(L_1) $a \wedge b = b \wedge a$; $a \vee b = b \vee a$, (Commutative law)

(L_2) $a \wedge (b \wedge c) = (a \wedge b) \wedge c$; $a \vee (b \vee c) = (a \vee b) \vee c$, (Associative law)

and (L_3) $a \wedge (a \vee b) = a$; $a \vee (a \wedge b) = a$, (Absorption law).

$$\text{Now, by } (L_3), \quad a \vee (a \wedge a) = a,$$
$$\text{and hence again by } (L_3), \quad (a \wedge a) = a \wedge (a \vee (a \wedge a)) = a.$$

Similarly, $a \vee a = a$ for each $a \in L$.

Theorem 1.9.2:
The relation "$a \leq b$ iff $a \wedge b = a$" in a lattice $(L, \wedge, \vee)$, defines a partial order on L.

Proof. 1. Trivially $a \leq a$ since $a \wedge a = a$ in L. Thus "$\leq$" is reflexive on L.

2. If $a \leq b$ and $b \leq a$, we have $a \wedge b = a$ and $b \wedge a = b$. Hence $a = b$ since by (L_1), $a \wedge b = b \wedge a$. This proves that "$\leq$" is antisymmetric.

3. Finally we prove that "$\leq$" is transitive. Let $a \leq b$ and $b \leq c$ so that $a \wedge b = a$ and $b \wedge c = b$.

   Now

$$\begin{aligned} a \wedge c = (a \wedge b) \wedge c &= a \wedge (b \wedge c) \quad \text{by } (L_2) \\ &= a \wedge b = a \quad \text{and hence} \quad a \leq c. \end{aligned}$$

Thus $(L, \ \leq)$ is a poset. ■

The converse of Theorem 1.9.2 is as follows:

Theorem 1.9.3:
Any partially ordered set $(L, \leq)$ in which any two elements have an infimum and a supremum in L is a lattice under the operations,

$$a \wedge b = \inf(a,\, b), \quad \textit{and} \quad a \vee b = \sup(a,\, b).$$

Proof. Follows from the definitions of supremum and infimum. ■

Examples of Lattices

For a nonvoid set S, $\big(\mathcal{P}(S),\, \cap,\, \cup\big)$ is a lattice. Again for a positive integer n, define D_n to be the set of divisors of n, and let $a \leq b$ in D_n mean that $a \mid b$, that is, a is a divisor of b. Then $a \wedge b = (a,\, b)$, the gcd of a and b, and $a \vee b = [a,\, b]$, the lcm of a and b, and $\big(D_n,\, \vee,\, \wedge\big)$ is a lattice (see Chapter 2 for the definitions of gcd and lcm). For example, if $n = 20$, Fig. 1.7 gives the Hasse diagram of the lattice $D_{20} = \{1,\, 2,\, 4,\, 5,\, 10,\, 20\}$. It has the least element 1 and the greatest element 20. We next give the Duality Principle valid in lattices.

Duality Principle

In any lattice $(L, \wedge, \vee)$, any formula or statement involving the operations $\wedge$ and $\vee$ remains valid if we replace $\wedge$ by $\vee$ and $\vee$ by $\wedge$.

The statement obtained by the replacement is called "the dual statement" of the original statement.

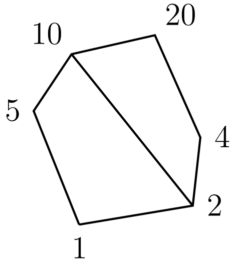


Figure 1.7: The lattice D_{20}

The validity of the duality principle lies in the fact that in the set of axioms for a lattice, any axiom obtained by such a replacement is also an axiom. Consequently, whenever we want to establish a statement and its dual, it is enough to establish one of them. Note that the dual of the dual statement is the original statement. For instance, the statement:

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$$

implies the statement

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c).$$

Definition 1.9.4:
A subset L' of a lattice $L = (L, \wedge, \vee)$ is a sublattice of L if $(L', \wedge, \vee)$ is a lattice.

A subset S of a lattice $(L, \wedge, \vee)$ need not be a sublattice even if it is a poset with respect to the operation " $\leq$ " defined by

$$a \leq b \quad \text{iff} \quad a \wedge b = a$$

For example, let $(L, \cap, \cup)$ be the lattice of all subsets of a vector space L and S be the collection of all subspaces of L. Then S is, in general, not a sublattice of L since the union of two subspaces of L need not be a subspace of L.

To understand it better at this stage, take the vector space L

to be the 3-dimensional Euclidean space $\mathbf{R}^3$. Then the set union of two distinct lines through the origin is not a line through the origin.

Lemma 1.9.5:
In any lattice $L = (L, \wedge, \vee)$, the operations $\wedge$ and $\vee$ are isotone, that is, for a, b, c in L,

$$\textit{if } b \leq c, \quad \textit{then} \quad a \wedge b \leq a \wedge c \quad \textit{and} \quad a \vee b \leq a \vee c.$$

Proof. We have (see Exercise 5 of Section 1.12)

$$\begin{aligned} a \wedge b = a \wedge (b \wedge c) &= (a \wedge b) \wedge c \quad \text{(by L2)} \\ &\leq a \wedge c \quad (\text{as } a \wedge b \leq a). \end{aligned}$$

Similarly (or by duality), $a \vee b \leq a \vee c$.

∎

Lemma 1.9.6:
Any lattice satisfies the two distributive inequalities:

1. *$x \wedge (y \vee z) \geq (x \wedge y) \vee (x \wedge z)$, and*

2. *$x \vee (y \wedge z) \leq (x \vee y) \wedge (x \vee z)$.*

Proof. We have $x \wedge y \leq x$, and $x \wedge y \leq y \leq y \vee z$. Hence, $x \wedge y \leq \inf(x, y \vee z) = x \wedge (y \vee z)$, Also $x \wedge z \leq x$, and $x \wedge z \leq z \leq y \vee z$. Thus $x \wedge z \leq x \wedge (y \vee z)$. Therefore, $x \wedge (y \vee z)$ is an upper bound for both $x \wedge y$ and $x \wedge z$ and hence greater than or equal to their least upper bound, namely, $(x \wedge y) \vee (x \wedge z)$. The second statement follows by duality. ∎

Lemma 1.9.7:
The elements of a lattice satisfy the modular inequality:

$$x \leq z \quad \textit{implies} \quad x \vee (y \wedge z) \leq (x \vee y) \wedge z.$$

Proof. We have $x \leq x \vee y$ and $x \leq z$. Hence $x \leq (x \vee y) \wedge z$. Also, $y \wedge z \leq y \leq x \vee y$ and $y \wedge z \leq z$, whence $y \wedge z \leq (x \vee y) \wedge z$. These together imply that

$$x \vee (y \wedge z) \leq (x \vee y) \wedge z. \quad \blacksquare$$

In other words:

$$\begin{aligned} \text{By Lemma 1.9.6} \quad x \vee (y \wedge z) &\leq (x \vee y) \wedge (x \vee z) \\ &= (x \vee y) \wedge z, \text{ as } x \leq z. \quad \blacksquare \end{aligned}$$

Distributive and Modular Lattices

Two important classes of lattices are the distributive lattices and modular lattices. We now define them.

Definition 1.9.8:
A lattice $(L, \wedge, \vee)$ is called *distributive* if the two distributive laws:

$$\begin{aligned} a \wedge (b \vee c) &= (a \wedge b) \vee (a \wedge c), \\ \text{and} \quad a \vee (b \wedge c) &= (a \vee b) \wedge (a \vee c) \end{aligned}$$

hold for all $a, b, c \in L$.

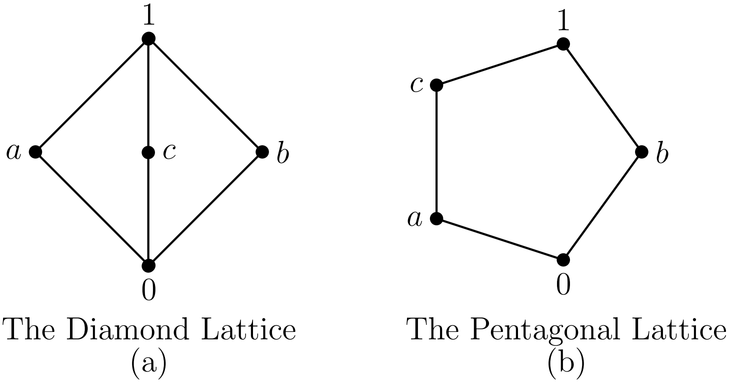
Note that in view of the duality that is valid for lattices, if one of the two distributive laws holds in L, then the other would automatically remain valid.

Example 1.9.9 (Examples of distributive lattices):

1. $\big(\mathcal{P}(S), \cap, \cup\big)$
2. $(\mathbf{N}, \text{gcd}, \text{lcm})$. (Here $a \wedge b = (a, b)$, the gcd of a and b, and $a \vee b = [a, b]$, the lcm of a and b.)

Example 1.9.10 (Examples of nondistributive lattices):

1. The “diamond lattice” of Figure 1.8 (a)
2. The “pentagonal lattice” of Figure 1.8 (b)



The Diamond Lattice
(a)

The Pentagonal Lattice
(b)

Figure 1.8: Hasse diagram of the diamond and pentagonal lattices

In the diamond lattice,

$$a \wedge (b \vee c) = a \wedge 1 = a, \quad \text{while} \quad (a \wedge b) \vee (a \wedge c) = 0 \vee 0 = 0\,(\neq a).$$

In the case of the pentagonal lattice,

$$a \vee (b \wedge c) = a \vee 0 = a, \quad \text{while}$$
$$(a \vee b) \wedge (a \vee c) = 1 \wedge c = c(\neq a).$$

Complemented Lattice

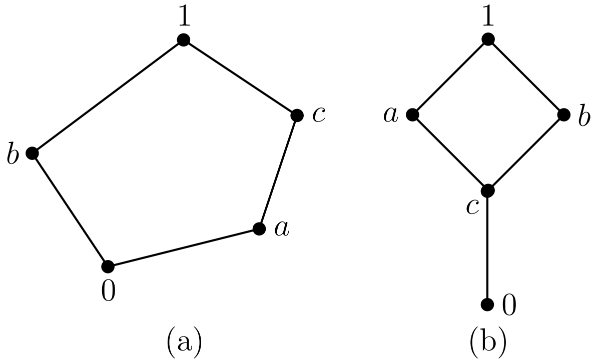
Definition 1.9.11:
A lattice L with 0 and 1 is complemented if for each element $a \in L$, there exists at least one element $b \in L$ such that

$$a \wedge b = 0 \quad \text{and} \quad a \vee b = 1.$$

Example 1.9.12:

1. Let $L = \mathcal{P}(S)$, the power set of a nonvoid set S. If $A \in L$, then A has a unique complement $B = L \setminus A$. Here $0 =$ the empty set and $1 =$ the whole set S.

2. The complement of an element need not be *unique*. For example, in the lattice of Figure 1.9 (a), both a and c are complements of b, since $b \wedge a = b \wedge c = 0$, and $b \vee a = b \vee c = 1$.



(a) (b)

Figure 1.9: (a) A complemented lattice, (b) a non-complemented lattice

3. Not every lattice with 0 and 1 is complemented. In the lattice of Figure 1.9 (b), a has no complement.

That the diamond lattice and the pentagonal lattice (of Figure 1.8) are crucial in the study of distributive lattices is the content of Theorem 1.9.13.

Theorem 1.9.13:
A lattice is distributive iff it does not contain a sublattice isomorphic to the diamond lattice or the pentagonal lattice.

The necessity of the condition in Theorem 1.9.13 is trivial but the proof of sufficiency is more involved.

However, a much simpler result is the following:

Theorem 1.9.14:
If a lattice L is distributive, then for a, b, $c \in L$, the equations $a \wedge b = a \wedge c$ and $a \vee b = a \vee c$ together imply that $b = c$.

Proof. Assume that L is distributive. Suppose that $a \wedge b = a \wedge c$ and $a \vee b = a \vee c$. We have to show that $b = c$.

$$\begin{aligned} \text{Now} \quad b &= b \wedge (a \vee b) && \text{(by absorption law)} \\ &= b \wedge (a \vee c) && \text{(by hypothesis)} \end{aligned}$$

$$\begin{aligned}
&= (b \wedge a) \vee (b \wedge c) && \text{(by distributivity)}\\
&= (a \wedge b) \vee (b \wedge c) = (a \wedge c) \vee (b \wedge c) &&\\
&= (a \vee (b \wedge c)) \wedge (c \vee (b \wedge c)) && \text{(by hypothesis)}\\
&= (a \vee (b \wedge c)) \wedge c && \text{(by absorption law)}\\
&= ((a \vee b) \wedge (a \vee c)) \wedge c = ((a \vee c) \wedge (a \vee c)) \wedge c &&\\
&= (a \vee c) \wedge c && \text{(by hypothesis)}\\
&= c. &&
\end{aligned}$$

■

We now consider another important class of lattices called modular lattices.

Modular Lattices

DEFINITION 1.9.15:
A lattice is modular if it satisfies the following modular identity:

$$x \leq z \Rightarrow x \vee (y \wedge z) = (x \vee y) \wedge z.$$

Hence the modular lattices are those lattices for which equality holds in Lemma 1.9.7. It is well known that the normal subgroups of any group form a modular lattice.

The pentagonal lattice of Figure 1.9 is nonmodular since in it $a \leq c, a \vee (b \wedge c) = a \vee 0 = a$ while $(a \vee b) \wedge c = 1 \wedge c = c (\neq a)$. In fact, the following result is true.

Theorem 1.9.16:
Any nonmodular lattice L contains the pentagonal lattice as a sublattice.

Hence, as a consequence, we can conclude that if a lattice L does not contain the pentagonal lattice as a sublattice, then L is modular.

Proof. As L is nonmodular, there exist elements a, b, c in L such that

$$a < c \quad \text{and} \quad a \vee (b \wedge c) \neq (a \vee b) \wedge c.$$

(Note: For $a = c$, equality holds on the right.) But the modular inequality (Lemma 1.9.7) holds for any lattice. Hence

$$a < c \quad \text{and} \quad a \vee (b \wedge c) < (a \vee b) \wedge c.$$

Set $x = a \vee (b \wedge c)$, and $y = (a \vee b) \wedge c$, so that $x < y$. Now

$$\begin{aligned}
x \vee b &= \big(a \vee (b \wedge c)\big) \vee b \\
&= a \vee \big((b \wedge c) \vee b\big) \\
&= a \vee b \\
&= (a \wedge c) \vee b && \text{(by absorption)} \\
&= (a \wedge c) \vee (b \wedge c) \vee b \\
&= ((a \vee b) \wedge c) \vee b = y \vee b
\end{aligned}$$

By duality, $x \wedge b = y \wedge b$. Now since $y \leq c$, $b \wedge y \leq b \wedge c \leq x$, the lattice of Figure 1.10 is the pentagonal lattice contained in L. ■

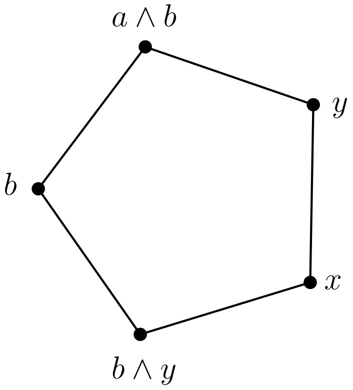


Figure 1.10: Illustration

A consequence of Theorems 1.9.13 and 1.9.16 is that every distributive lattice is modular (since L is distributive ⇒ L does not contain the pentagonal lattice as a sublattice ⇒ L is modular). The diamond lattice is an example of a modular lattice that is not distributive. Another example is the lattice of all vector subspaces of a vector space V. (See Exercise 1.12.)

Theorem 1.9.17:
In a distributive lattice L, an element can have at most one complement.

Proof. Suppose x has two complements y_1, y_2. Then

$$x \wedge y_1 = 0 = x \wedge y_2, \quad \text{and} \quad x \vee y_1 = 1 = x \vee y_2.$$

As L is distributive, by Theorem 1.9.14, $y_1 = y_2$. ■

1.10 Boolean Algebras

1.10.1 Introduction

A Boolean algebra is an abstract mathematical system (abstraction of the algebra of sets and propositions, see Chapter 6 for details) primarily used in computer science and in expressing the relationships between sets. This system was developed by the English mathematician George Boole in 1850 to permit an algebraic manipulation of logical statements. Such a manipulation can demonstrate whether or not a statement is true and show how a complicated statement can be rephrased in a similar, more convenient form without losing its meaning.

Definition 1.10.1:
A Boolean algebra is a complemented distributive lattice.

Hence a Boolean algebra B has the universal elements 0 and 1 and every element x of B has a complement x', and since B is a distributive lattice, by Theorem 1.9.17, x' is unique. The Boolean algebra B is symbolically represented as $(B, \wedge, \vee, 0, 1,')$.

1.10.2 Examples of Boolean Algebras

1. Let S be a nonvoid set. Then $(\mathcal{P}(S), \cap, \cup, \emptyset, S, ')$ is a Boolean algebra. Here, if $A \in \mathcal{P}(S), A' = S \setminus A$ is the complement of A in S.

2. Let $\mathcal{B}^n$ denote the set of all binary sequences of length n. For $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n) \in \mathcal{B}^n$, set

   $(a_1, \ldots, a_n) \wedge (b_1, \ldots, b_n) = (\min(a_1, b_1), \ldots, \min(a_n, b_n))$,
   $(a_1, \ldots, a_n) \vee (b_1, \ldots, b_n) = (\max(a_1, b_1), \ldots, \max(a_n, b_n))$,
   and $(a_1, \ldots, a_n)' = (a_1', \ldots, a_n')$, where $0' = 1$ and $1' = 0$.

   Note that the zero element is the n-vector $(0, 0, \ldots, 0)$, and the unit element is $(1, 1, \ldots, 1)$. For instance, if $n = 3$, $x = (1, 1, 0)$ and $y = (0, 1, 0)$, then $x \wedge y = (0, 1, 0)$, $x \vee y = (1, 1, 0)$, and $x' = (0, 0, 1)$.

Theorem 1.10.2 (De Morgan's laws)**:**
Any Boolean algebra B satisfies De Morgan's laws: For any two elements a, $b \in B$,

$$(a \wedge b)' = a' \vee b', \quad \textit{and} \quad (a \vee b)' = a' \wedge b'.$$

Proof. We have by distributivity,

$$\begin{aligned}
(a \wedge b) \wedge (a' \vee b') &= (a \wedge (a' \vee b')) \wedge (b \wedge (a' \vee b')) \\
&= ((a \wedge a') \vee (a \wedge b')) \wedge ((b \wedge a') \vee (b \wedge b')) \\
&= (0 \vee (a \wedge b')) \wedge ((b \wedge a') \vee 0) \\
&= (a \wedge b') \wedge (b \wedge a') \\
&= (a \wedge b \wedge a') \wedge (b' \wedge b \wedge a') \\
&\qquad\qquad\qquad \text{(see Exercise 1.12 \#4)} \\
&= (a \wedge a' \wedge b) \wedge (b' \wedge b \wedge a') \\
&= (0 \wedge b) \wedge (0 \wedge a') \\
&= 0 \wedge 0 \\
&= 0 \quad (\text{since } a \wedge a' = 0 = b \wedge b').
\end{aligned}$$

Similarly,

$$\begin{aligned}
(a \wedge b) \vee (a' \vee b') &= (a \vee (a' \vee b')) \wedge (b \vee (a' \vee b')) \\
&\qquad\qquad\qquad \text{(by distributivity)} \\
&= ((a \vee a') \vee b') \wedge ((b \vee b') \vee a') \\
&\qquad\qquad\qquad (\text{since } a' \vee b' = b' \vee a') \\
&= (1 \vee b') \wedge (1 \vee a') = 1 \wedge 1 = 1.
\end{aligned}$$

Hence the complement of $a \wedge b$ is $a' \vee b'$. In a similar manner, we can show that $(a \vee b)' = a' \wedge b'$. ∎

Corollary 1.10.3:
In a Boolean algebra B, for $a, b \in B$, $a \leq b$ iff $a' \geq b'$

Proof. $a \leq b \Leftrightarrow a \vee b = b \Leftrightarrow (a \vee b)' = a' \wedge b' = b' \Leftrightarrow a' \geq b'$. ∎

Theorem 1.10.4:
In a Boolean algebra B, we have for all $a, b \in B$,

$$a \leq b \quad \textit{iff} \quad a \wedge b' = 0 \quad \textit{iff} \quad a' \vee b = 1.$$

Proof. Since $(a \wedge b')' = a' \vee b$ and $0' = 1$, it is enough to prove the first part of the theorem. Now $a \leq b \Rightarrow$ (by isotone property) $a \wedge b' \leq b \wedge b' = 0$. $\Rightarrow a \wedge b' \leq 0 \Rightarrow a \wedge b' = 0$.

Conversely, let $a \wedge b' = 0$. Then

$$\begin{aligned} a &= a \wedge 1 = a \wedge (b \vee b') = (a \wedge b) \vee (a \wedge b') \\ &= a \wedge b \Rightarrow a \leq b. \end{aligned}$$ ∎

Next we briefly discuss Boolean subalgebras and Boolean isomorphisms. These are notions similar to subgroups and group-isomorphisms in the theory of groups.

Boolean Subalgebras

DEFINITION 1.10.5:
A Boolean subalgebra of a Boolean algebra $B = (B, \wedge, \vee, 0, 1, ')$ is a subset B_1 of B such that $(B_1, \wedge, \vee, 0, 1, ')$ is itself a Boolean algebra with the same elements 0 and 1 of B.

Boolean Isomorphisms

DEFINITION 1.10.6:
A Boolean homomorphism from a Boolean algebra B_1 to a Boolean algebra B_2 is a map $f : B_1 \to B_2$ such that for all a, b in B_1,

1. $f(a \wedge b) = f(a) \wedge f(b)$,

2. $f(a \vee b) = f(a) \vee f(b)$, and

3. $f(a') = (f(a))'$.

Conditions (1) and (2) imply that f is a lattice homomorphism from B_1 to B_2 while condition (3) tells us that f takes the complement of an element (which is unique in a Boolean algebra) in B_1 to the complement in B_2 of the image of that element.

Theorem 1.10.7:
Let $f : B_1 \to B_2$ be a Boolean homomorphism. Then

1. *$f(0) = 0$, and $f(1) = 1$.*

2. *f is isotone.*

3. *The image $f(B_1)$ is a Boolean subalgebra of B_2.*

Proof. Straightforward. ■

Example 1.10.8:
Let $S = \{1, 2, \ldots, n\}$, and let A be the Boolean algebra $(\mathcal{P}(S) = A, \cap, \cup, ')$, and let B be the Boolean algebra defined by the set of all functions from S to the 2-element set $\{0, 1\}$. Any such function is a sequence $(x_1, \ldots, x_n)$ where $x_i = 0$ or 1. Let $\wedge$, $\vee$ be as in Example 2 of Section 1.10.1. Now consider the map $f : A = \mathcal{P}(S) \to B = \{0, 1\}^S$, the set of all 2^n functions from $S \to \{0, 1\}$, defined as follows: For $X \subset S$ (that is, $X \in \mathcal{P}(S) = A$), let $f(X) = (x_1, x_2, \ldots, x_n)$, where $x_i = 1$ or 0 according to whether $i \in X$ or not. For $X, Y \in P(S)$, set $f(X) \wedge f(Y) = f(X \cap Y)$, the binary sequence having 1 only in the places common to X and Y as per the definitions in Example 2 of Section 1.10.1 Similarly, 1.10.1. Similarly, $f(X \cup Y) =$ the binary sequence having 1 in all the places corresponding to the 1's in the set $X \cup Y = f(X) \vee f(Y)$.

Further, $f(X') = f(S \setminus X) =$ the binary sequence having 1's in the places where X has zeros, and zeros in the places where X has 1's $= (f(X)')$. f is 1–1 since distinct binary sequences in B arise

out of distinct subsets of S. Finally, f is onto, since any binary sequence in B is the image of the corresponding subset (that is, the subset corresponding to the places of the sequence with 1) of X. Thus f is a Boolean isomorphism.
Remark: The binary vector $f(X)$ is usually called the characteristic vector of the set X.

Example 1.10.9:
Let A be a proper Boolean subalgebra of $B = \mathcal{P}(S)$. Then if $f : A \to B$ is the identity function, f is a lattice homomorphism since,

$$\begin{aligned} & f(A_1 \wedge A_2) = A_1 \wedge A_2 = f(A_1) \wedge f(A_2), \\ \text{and} \quad & f(A_1 \vee A_2) = A_1 \vee A_2 = f(A_1) \vee f(A_2). \end{aligned}$$

However $f(A') = f$ (complement of A in A)$= f(\emptyset) = \emptyset = 0_B$, while $(f(A))' = A' = B \setminus A \neq \emptyset$.
Hence $f(A') \neq f(A')$, and f is *not* a Boolean homomorphism.

1.11 Atoms in a Lattice

Definition 1.11.1:
An element a of a lattice L with zero is called an atom of L if $a \neq 0$ and for all $b \in L$, $0 < b \leq a$ implies that $b = a$. That is to say, a is an atom if there is no nonzero b strictly less than a.

Definition 1.11.2:
An element a of a lattice L is called *join-irreducible* if $a = b \vee c$, then $a = b$ or $a = c$; otherwise, a is join-reducible.

Lemma 1.11.3:
Every atom of a lattice (with zero) is join-irreducible.

Proof. Let a be an atom of a lattice L, and let $a = b \vee c$, $a \neq b$. Then $b \leq b \vee c = a$, but since $b \neq a$ and a is an atom of L, $b = 0$, and hence $a = c$. ■

Lemma 1.11.4:
Let L be a distributive lattice and $c \in L$ be join-irreducible. If $c \leq a \vee b$, then $c \leq a$ or $c \leq b$. In particular, the same result is true if c is an atom of L.

Proof. As $c \leq a \vee b$, and L is distributive, $c = c \wedge (a \vee b) = (c \wedge a) \vee (c \wedge b)$. As c is join-irreducible, this means that $c = c \wedge a$ or $c = c \wedge b$, that is, $c \leq a$ or $c \leq b$. The second statement follows immediately from Lemma 1.11.3. ■

Definition 1.11.5:

1. Let L be a lattice and a and b, $a \leq b$, be any two elements of L. Then the closed interval $[a, b]$ is defined as:
$$[a, b] = \{x \in L : a \leq x \leq b\}.$$
2. Let $x \in [a, b]$. x is said to be relatively complemented in $[a, b]$, if x has a complement y in $[a, b]$, that is, $x \wedge y = a$ and $x \vee y = b$. If all closed intervals $[a, b]$ of L are complemented, then the lattice L is said to be relatively complemented.
3. If L has a zero element and all elements in $[0, b]$ have complements in L for every nonzero b in L, then L is said to be sectionally complemented.

Our next theorem is crucial for the proof of the representation theorem for finite Boolean algebras.

Theorem 1.11.6:
The following statements are true:

1. *Every Boolean algebra is relatively complemented.*
2. *Every relatively complemented lattice is sectionally complemented.*
3. *In any finite sectionally complemented lattice, each nonzero element is a join of finitely many atoms.*

Proof. 1. Let $[a,b]$ be a closed interval in a Boolean algebra B, and $x \in [a,b]$.

We have to prove that $[a,b]$ is complemented. Now, as B is the Boolean algebra, it is a complemented lattice and hence there exists x' in B such that $x \wedge x' = 0$, and $x \vee x' = 1$. Set $y = b \wedge (a \vee x')$. Then $y \in [a,b]$. Also, y is a complement of x in $[a,b]$ since

$$\begin{aligned} x \wedge y &= x \wedge (b \wedge (a \vee x')) \\ &= (x \wedge b) \wedge (a \vee x') = x \wedge (a \vee x') \\ &\qquad\qquad\qquad\qquad (\text{as } x \in [a,b],\ x \leq b) \\ &= (x \wedge a) \vee (x \wedge x') \qquad (\text{as } B \text{ is distributive}) \\ &= (x \wedge a) \vee (0) = a \quad (\text{as } \ a \leq x) \\ \text{and,}\quad x \vee y &= x \vee (b \wedge (a \vee x')) = (x \vee b) \wedge (x \vee (a \vee x')) \\ &= b \wedge \big((x \vee x') \vee a\big) \quad (\text{again by distributivity}) \\ &= b \wedge 1 = b \quad (\text{since } x \vee x' = 1, \text{ and } 1 \vee a = 1). \end{aligned}$$

Hence B is complemented in the interval $[a,b]$.

2. If L is relatively complemented, L is complemented in $[0,b]$ for each $b \in L$ (take $a = 0$). Hence L is sectionally complemented.

3. Let a be a nonzero element of a finite sectionally complemented lattice L. As L is finite, there are only finitely many atoms $p_1, \ldots, p_n$ in L such that $p_i \leq a$, $1 \leq i \leq n$, and let $b = p_1 \vee \cdots \vee p_n$. Now, $b \leq a$, since b is the least upper bound of $p_1, \ldots, p_n$ while a is an upper bound of $p_1, \ldots p_n$. Suppose $b \neq a$, then b has a nonzero complement, say, c, in the section $[0,a]$ since we have assumed that L is sectionally complemented. Let p be an atom such that $p \leq c \ \ (\leq a)$. Then $p \in \{p_1, \ldots, p_n\}$, as by assumption $p_1, \ldots, p_n$ are the only atoms with $p_i \leq a$, and hence, $p = p \wedge b \leq c \wedge b = 0$ (as c is the complement of b), a contradiction. Hence $b = a = p_1 \vee \cdots \vee p_n$. ■

An immediate consequence of Theorem 1.11.6 is the following result.

Corollary 1.11.7:
In any finite Boolean algebra, every nonzero element is a join of atoms.

We end this section with the representation theorem for finite Boolean algebras which says that any finite Boolean algebra may be thought of as the Boolean algebra $\mathcal{P}(S)$ defined on a finite set S.

Theorem 1.11.8 (Representation theorem for finite Boolean algebras)**:**
Let B be a finite Boolean algebra and A, the set of its atoms. Then there exists a Boolean isomorphism $B \simeq \mathcal{P}(A)$.

Proof. For $b \in B$, define $A(b) = \{a \in A : a \leq b\}$ so that $A(b)$ is the set of the atoms of B that are less than or equal to b. Then $A(b) \in \mathcal{P}(A)$. Now define

$$\phi : B \to \mathcal{P}(A)$$

by setting $\phi(b) = A(b)$. We now prove that ϕ is a Boolean isomorphism. We first show that ϕ is a lattice homomorphism, that is, for $b_1, b_2 \in B$, $\phi(b_1 \wedge b_2) = \phi(b_1) \wedge \phi(b_2) = \phi(b_1) \cap \phi(b_2)$, and $\phi(b_1 \vee b_2) = \phi(b_1) \vee \phi(b_2) = \phi(b_1) \cup \phi(b_2)$.

Equivalently, we show that

$$\begin{aligned} A(b_1 \wedge b_2) &= A(b_1) \cap A(b_2), \\ \text{and} \quad A(b_1 \vee b_2) &= A(b_1) \cup A(b_2). \end{aligned}$$

Let a be an atom of B. Then $a \in A(b_1 \wedge b_2) \Leftrightarrow a \leq b_1 \wedge b_2 \Leftrightarrow a \leq b_1$ and $a \leq b_2 \Leftrightarrow a \in A(b_1) \cap A(b_2)$. Similarly, $a \in A(b_1 \vee b_2) \Leftrightarrow a \leq b_1 \vee b_2 \Leftrightarrow a \leq b_1$ or $a \leq b_2$. (As a is an atom, a is join-irreducible by Lemma 1.11.3 and B being a Boolean algebra, it is a distributive lattice. Now apply Lemma 1.11.4) $\Leftrightarrow a \in A(b_1)$ or $a \in A(b_2)$ $\Leftrightarrow a \in A(b_1) \cup A(b_2)$. Next, as regards complementation,

$$a \in \phi(b') \Leftrightarrow a \in A(b') \Leftrightarrow a \leq b' \Leftrightarrow a \wedge b = 0 \;\text{ (by Theorem 1.10.4)}$$

$$\Leftrightarrow a \nleq b \Leftrightarrow a \notin A(b) \Leftrightarrow a \in A \setminus A(b) = (A(b))'.$$

$$\text{Thus} A(b') = (A(b))'.$$

Finally, $\phi(0) =$ set of atoms in B that are ≤ 0
$= \emptyset$ (as there are none), the zero element of $\mathcal{P}(A)$,
and $\phi(1) =$ set of atoms in B that are ≤ 1
$=$ set of all atoms in $B = A$, the unit element of $\mathcal{P}(A)$.

All that remains to show is that ϕ is a bijection. By Corollary 1.11.7, any $b \in B$ is a join, say, $b = a_1 \vee \cdots \vee a_n$ (of a finite number n) of atoms $a_1, \ldots, a_n$ of B. Hence $a_i \leq b$, $1 \leq i \leq n$. Suppose $\phi(b) = \phi(c)$, that is, $A(b) = A(c)$. Then each $a_i \in A(b) = A(c)$ and so $a_i \leq c$ for each i, and hence $b \leq c$. In a similar manner, we can show that $c \leq b$, and hence $b = c$. In other words, ϕ is injective.

Finally we show that ϕ is surjective. Let $C = \{c_1, \ldots, c_k\} \in \mathcal{P}(A)$ so that C is a set of atoms in A. Set $b = c_1 \vee \cdots \vee c_k$. We show that $\phi(b) = C$ and this would prove that ϕ is onto. Now $c_i \leq b$ for each i, and so by the definition of ϕ, $\phi(b) = \{$ set of atoms $c \in A$ with $c \leq b\} \supseteq C$. Conversely, if $a \in \phi(b)$, then a is an atom with $a \leq b = c_1 \vee \ldots \vee c_k$. Therefore $a \leq c_i$ for some i by Lemma 1.11.4. As c_i is an atom and $a \neq 0$, this means that $a = c_i \in C$. Thus $\phi(b) = C$. ■

1.12 Exercises

1. Draw the Hasse diagram of all the 15 essentially distinct (that is, non-isomorphic) lattices with six elements.

2. Show that the closed interval $[a, b]$ is a sublattice of the lattice $\big(\mathbf{R}, \inf, \sup\big)$.

3. Give an example of a lattice with no zero element and with no unit element.

4. In a lattice, show that
(i) $(a \wedge b) \wedge c = (c \wedge b) \wedge a$,
(ii) $(a \wedge b) \wedge c = (a \wedge c) \wedge (b \wedge c)$.

5. Prove that in a lattice $a \leq b \Leftrightarrow a \wedge b = a \Leftrightarrow a \vee b = b$.

6. Show that every chain is a distributive lattice.

7. Show that the three lattices of Fig. 1.11 are *not* distributive.

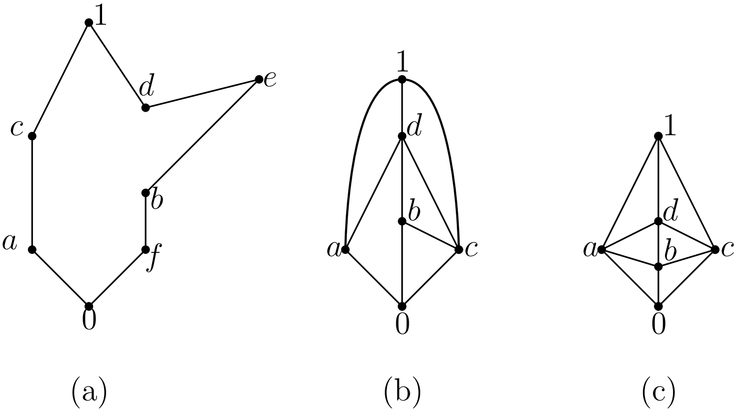


Figure 1.11: Three lattices

8. Show that the lattice of Fig. 1.11 (c) is not modular.

9. Show that the lattice of all subspaces of a vector space is not distributive.

10. Which of the following lattices are (i) distributive, (ii) modular, (iii) modular, but not distributive? (a) D_{10} (b) D_{20} (c) D_{36} (d) D_{60}.

11. Give a detailed proof of Theorem 1.10.7.

Bibliography

[1] K. J. Devlin, Sets, Functions and Logic, CRC Press, 2003.

[2] I. S. Luthar, Sets, Functions and Numbers, Alpha Science International Ltd, 2005.

[3] P. R. Halmos, Naive Set Theory, D. Von Nostrand Company, INC, 1960.

[4] M. N. Yiannis, Notes on Set Theory, UTM, Springer, 1994.

Chapter 2

Combinatorics

La Combinatoire est l'étude:

- d'une configuration
- d'une configuration inconnue
- de dénombrement exact de configurations
- de dénombrement approché de configurations
- d'énumération de configurations
- d'optimization

Claude Berge
Principes de Combinatoire

In this chapter on combinatorics, we start from the elementary rules of counting, then study permutations and combinations, binomial coefficients, binomial theorem, multinomial coefficients, multinomial theorem, Stirling numbers of the first and the second kind, Bell numbers, the Principle of Inclusion and Exclusion (simple and weighted versions), some applications of the Principle of Inclusion and Exclusion to number theory and the theory of permanents, generating function techniques and recurrence relations, Bernoulli numbers, Catalan numbers, and an algorithm for generating all the subsets of a given finite set.

2.1 What Is Combinatorics?

Combinatorics can be described as the study of properties of *finite* sets and *finite* structures (see [2]). Unless otherwise stated, all sets in this chapter are *finite*.

Combinatorics is traditionally associated with the permutation and combination of a finite number of objects. Combinatorics deals with the following two types of problems: Existence problems and enumeration problems (see [1]).

In the existence problem, the following question is raised: Does there exist a particular special structure or situation or phenomenon? If the answer to the existence problem is affirmative, then we raise the second question: How many such special structures exist? This is the enumeration problem. In fact, the second problem of enumeration can be viewed as a *generalization* of the existence problem. This is because, the existence of a special structure has an affirmative answer if and only if the number of such special structures is at least *one*.

The following examples illustrate the existence and enumeration problem in combinatorics.

Example 2.1.1 (Enumeration problem):
Consider a set A consisting of 4 integers 1,2,3,4. Symbolically, $A = \{1, 2, 3, 4\}$. Find the number of subsets of A consisting of just *two* elements. This is an enumeration problem. Here the existence problem is trivially solved by "exhibiting" a subset of A with two elements (for example, $\{1, 2\}$ is a 2-element subset of A.)

There are *six* subsets of A containing exactly 2 elements, namely, $\{1, 2\}$, $\{$1,3, $\}$, $\{1, 4\}$, $\{$2,3$\}$, $\{$2,4$\}$, $\{3, 4\}$.

Example 2.1.2 (Existence problem: Checkerboard problem):
Consider the familiar 8×8 chess board. From this chess board, we remove two diagonally opposite squares. The board thus obtained by the removal of two opposite squares is called a *checkerboard*. It has exactly 62 squares, the removed squares are marked with ×, "B" stands for a black square and "W" stands for a white square

(see Table 2.1).

Table 2.1: Checkerboard

×	B	W	B	W	B	W	B
B	W	B	W	B	W	B	W
W	B	W	B	W	B	W	B
B	W	B	W	B	W	B	W
W	B	W	B	W	B	W	B
B	W	B	W	B	W	B	W
W	B	W	B	W	B	W	B
B	W	B	W	B	W	B	×

Table 2.2: Dominoes

W	B

Example 2.1.3:
The question is the following: Is it possible to cover the entire checkerboard by using thirty-one 1×2 dominoes? This is *the existence problem.* The answer is "no" as the following simple "parity" argument proves:

In the conventional chess board, the squares are colored alternatively black and white. Hence there are 32 black squares and 32 white squares. Note that the two diagonally opposite squares are of the same color (either both black or both white). Suppose we have removed two white squares. Hence, our checkerboard has 30 white squares and 32 black squares. But a 1×2 rectangle of domino covers exactly one black square and one white square. The given 31 dominoes can cover 31 black squares and 31 white squares. But the checkerboard has an unequal number of black and white squares. Hence a complete covering of the checkerboard with 31 1×2 dominoes is impossible.

On the other hand, if we are given 32 dominoes of 1×2 rectangle, we can easily cover the *entire* chess board with 32 dominoes.

In this case, we are led to *the enumeration problem:* How many such coverings are possible?

EXAMPLE 2.1.4 (Existence problem: Euler's 36 officers problem): **Euler's 36 officers problem and the fallacy of Euler's conjecture [1][5]:**

Consider 6 regiments and from each regiment take 6 officers of 6 different ranks. Hence, we have a total of 36 officers.

Question (Euler): Is it possible to arrange these 36 officers in the form of a 6 × 6 matrix satisfying the following condition?

Each row and each column of the 6 × 6 matrix contains only one officer from each rank and only one officer from each column.

Euler conjectured that such an arrangement is impossible.

Representation: Let us denote the ranks and regiments by the integers 1,2,3,4,5,6. Then associate to each officer the ordered pair (i, j) with $1 \leq i, j \leq n$, where the first co-ordinate i represents the officer's rank and the second co-ordinate j, the officer's regiment. Then Euler's conjecture is equivalent to constructing a 6×6 matrix with distinct entries (i, j) with $1 \leq i, j \leq n$, in such a way that each row has all of the six numbers $1, 2, 3, 4, 5, 6$ in some order as the first co-ordinate and each column has all of the six numbers $1, 2, 3, 4, 5, 6$ in some order as the second co-ordinate.

Euler's 36 officers problem was verified by Tarry [5] by systematic enumeration.

Euler's conjecture: Euler further conjectured that such an arrangement/matrix is impossible for all square matrices of order $(4n + 2) \times (4n + 2)$ for all $n \geq 2$. This research earned the trio, Bose, Shrikhande, and Parker, the sobriquet Euler's spoiler, and created such a stir that even the *New York Times* brought the news out on its first page.

But Bose, Shrikhande, and Parker proved the existence of such a matrix for each $n \geq 2$. The following remarkable 10×10 matrix is a counterexample to Euler's conjecture for $n = 2$.

For convenience, let us use $a, b, c, d, e, f, g, h, i, j$ for the first co-ordinate instead of $1, 2, 3, 4, 5, 6, 7, 8, 9, 10$ and $A, B, C, D, E, F, G, H, I, J$ for the second co-ordinate in place of $1, 2, 3, 4, 5, 6, 7, 8, 9, 10$. For instance, gI denotes an officer of the rank 7 in the regiment 9 and aJ an officer of rank 1 of the regiment 10.

$$M = \begin{array}{c|cccccccccc|} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & jJ & fG & eH & dI & iA & hC & gE & aB & bD & cF \\ 2 & gF & aA & jG & fH & eI & iB & hD & bC & cE & dJ \\ 3 & hE & gJ & bB & aG & jH & fI & iC & cD & dF & eA \\ 4 & iD & hF & gA & cC & bG & aH & jI & dE & eJ & fB \\ 5 & aI & iE & hJ & gB & dD & cG & bH & eF & fA & jC \\ 6 & cH & bI & iF & hA & gC & eE & dG & fJ & jB & aD \\ 7 & eG & dH & cI & iJ & hB & gD & fF & jA & aC & bE \\ 8 & bA & cB & dC & eD & fE & jF & aJ & gG & hH & iI \\ 9 & dB & eC & fD & jE & aF & bJ & cA & hI & iG & gH \\ 10 & fC & jD & aE & bF & cJ & dA & eB & iH & gI & hG \end{array}$$

Each row and each column of the matrix M has all the letters $a, b, \ldots, j$ as first co-ordinate in some order. Similarly, each row and each column of the matrix M has all the letters $A, B, \ldots, J$ in some order as the second co-ordinate.

2.2 Elementary Counting Principles

First we recall the following elementary "sum rule" concerning sets. We denote by $|A|$, the number of elements in the set A.

FACT 2.2.0.1 (Sum rule):
Consider two disjoint finite sets A and B, that is, A and B are finite sets with $A \cap B = \emptyset$. The number of elements in the union $|A \cup B|$ is $|A| + |B|$. Symbolically,

$$|A \cup B| = |A| + |B|.$$

There is nothing sacred about the number two in the above fact. More generally, if $A_1, A_2, \cdots, A_n$ are n finite *pairwise* mutually disjoint sets (that is, $A_i \cap A_j = \emptyset$ for i, j with $1 \leq i < j \leq n$) then the number of elements in the union $A_1 \cup A_2 \cup \cdots \cup A_n$ is given by the equation

$$|A_1 \cup A_2 \cup \cdots \cup A_n| = |A_1| + |A_2| + \cdots + |A_n|.$$

The whole is equal to the sum of its parts.

Let us recall the idea of the *Cartesian product* of two or more given sets.

If A and B are two given sets then the Cartesian product or simply product of A and B is denoted by $A \times B$ and defined as the set of all ordered pairs (a, b) with $a \in A$ and $b \in B$. Symbolically,

$$A \times B = \{ (a, b) \mid a \in A, b \in B \}$$

More generally, the product of n sets $A_1, A_2, \cdots, A_n$ is denoted by $A_1 \times A_2 \times \cdots \times A_n$ and is defined as the set of all ordered n-tuples $(a_1, a_2, \cdots, a_n)$ with $a_i \in A_i$ for $i = 1, 2, \cdots, n$. a_i is called the ith *component* of the n-tuple $(a_1, a_2, \cdots, a_n)$

We declare the n-tuple $(a_1, a_2, \cdots, a_n)$ to be equal to the n-tuple $(a'_1, a'_2, \cdots, a'_n)$ if and only if $a_i = a'_i$ for all $i = 1, 2, \cdots, n$.

EXAMPLE 2.2.1 (Cartesian product):
Consider two sets A and B where $A = \{ 1, 2, 3 \}$ and $B = \{ 2, 4 \}$. Then the product of A and B is

$$A \times B = \{ (1, 2), (1, 4), (2, 2), (2, 4), (3, 2), (3, 4) \}$$

Its geometric representation is shown in Figure 2.1:

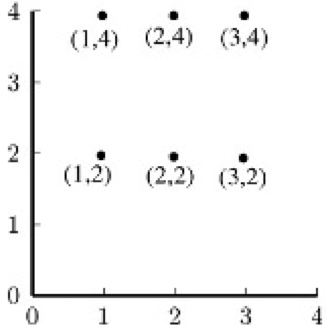


Figure 2.1: An example illustrating a Cartesian product

EXAMPLE 2.2.2 (Product of two intervals):
Consider the two-dimensional Cartesian plane xy. Consider two intervals $[1, 3] = \{ x \mid x$ real with $1 \leq x \leq 3 \}$ in the x axis and $[2, 3]\{ y \mid y$ real with $2 \leq y \leq 3 \}$ in the y axis. Then the geometric interpretation of the product $[1, 3] \times [2, 3]$ is the set of all points of the rectangle $ABCD$ in the xy plane where the co-ordinates of A,B,C,D are respectively $(1, 2), (1, 3), (3, 2), (3, 3)$ (see Figure 2.2).

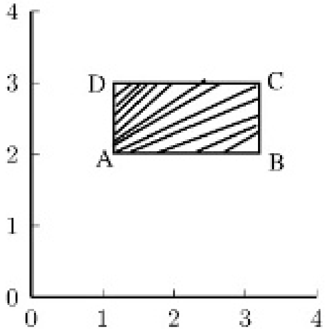


Figure 2.2: An example illustrating product of intervals

EXAMPLE 2.2.3 (Right circular cylinder as Cartesian product):
The product of a circle in the xy plane and a closed interval in the z axis is a right circular cylinder in the three dimensional space R^3.

Two Interpretations of functions

Consider a function $f : A \to B$ with $A = \{ 1, 2, 3, 4, 5 \}$ and $B = \{ a, b, , c, d \}$ with $f(1) = f(3) = a$,$f(2) = f(4) = b$,$f(5) = c$. This function is represented graphically in Figure 2.3:

This function f is denoted by

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ a & b & a & b & c \end{pmatrix}$$

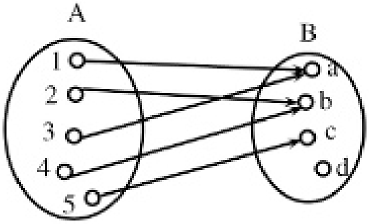


Figure 2.3: Graphical illustration of a function

First interpretation: Professors and offices/rooms

The elements of the *domain* set A stand for 5 professors and the elements of the co-domain set B are the rooms/offices occupied by these professors in the university.

Professors 1 and 2 share room "a," Professors 3 and 4 share room "b," whereas Professor 5 is all alone in office "c." Note that office "d" is *not* occupied by any of these five professors and hence unoccupied.

If every room is occupied by at least one professor then the function is *surjective or onto.* Hence the function f is not surjective because the room d is occupied by *no* one.

If every room is occupied by at most one person then the function is *injective or one to one.* Hence our function f is not injective.

Second interpretation: Cartesian product or word

The elements of the co-domain B are treated as *letters* and the set B as an *alphabet*.

The function f where

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ a & b & a & b & c \end{pmatrix}$$

is represented more compactly as $f = (a, b, a, b, c)$ if the domain set is *understood* to be the ordered 5-tuple $(1, 2, 3, 4, 5)$. We even remove commas in $f = (a, b, a, b, c)$ and write simply $f = (ababc)$. Hence, the function f can be viewed as an element of the Cartesian product $B^5 = \overbrace{B \times B \times \cdots \times B}^{5 \text{ times}}$.

The elements of the product B^5 are sometimes called *words* of *length* 5 of the alphabet B. (The number of letters in the word is its length). In this representation, the function is *identified* with a word whose length is equal to the number of elements of the *domain* of the function.

More generally, a function f from a set $A = \{1, 2, \ldots, n\}$ of n elements to a set B of m elements can be viewed as ordered n tuples of elements of the co-domain set B.

Equality of functions

The *equality of two functions* means the equality of the corresponding ordered tuples. In other words, the two functions have the same *effect* on each element of the domain set. If every letter of the alphabet appears at least once in a word then the function is surjective. If each letter appears at most once in a word then the function is injective.

In graphical terms:

A function is injective if and only if the number of arrows arriving at an element of the co-domain is at most one, that is, the in-degree of every element of the co-domain is ≤ 1.

A function is surjective if and only if the number of arrows arriving at each element of the co-domain is at least one, that is, the in-degree of every element of the co-domain is ≥ 1.

A function is bijective if and only if the number of arrows arriving at an element of a co-domain is exactly one, that is, the in-degree of every element of the co-domain is $= 1$.

The cardinality of two finite sets A and B can be compared by the following proposition whose proof is obvious.

Proposition 2.2.1:
Consider any two finite sets A and B. Then the following hold:

1. There is an injective function from the set A to the set B if and only if $|A| \leq |B|$.

2. There is a surjective function from the set A onto the set B if and only if $|A| \geq |B|$.

3. There is a bijective function or one-to-one correspondence from the set A onto the set B if and only if $|A| = |B|$.

For example, to prove that two finite sets A and B are of the same cardinality we may establish a one-to-one correspondence between the sets A and B.

We shall first see the following "fundamental principle"(also called the product rule) of combinatorics.

Proposition 2.2.2 (Fundamental principle or product rule):
Consider two finite sets A and B of cardinalities m and n respectively. Then the cardinality of the Cartesian product $A \times B$ is mn. In other words, $|A \times B| = |A||B|$.

The above fundamental principle is often stated in the following form:

Popular form of the fundamental principle

If a certain thing can be done in m ways, and when it has been done, a second thing can be done in n ways, then the total number of ways in which the two things can be done *together* is mn ways.

Example 2.2.4 (Product rule):
If there are 2 different routes r_1, r_2 to travel from Perpignan city to Lyon city and if there are three ways r_3, r_4, r_5 to go from Lyon city to Paris city then there are $2 \times 3 = 6$ different ways $r_1r_3, r_1r_4, r_1r_5, r_2r_3, r_2r_4, r_2r_5$ to travel from Perpignan city to Paris passing via Lyon (see Figure 2.4). In this figure, the three cities are represented as small circles and the route by curves/straight segments.

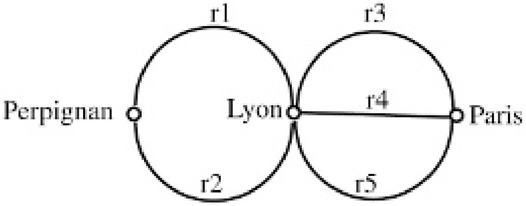


Figure 2.4: An example illustrating the product rule

The above product rule can be generalized to any number of k sets as below.

Proposition 2.2.3:
If $A_1, A_2, \dots, A_k$ are k sets with $|A_i| = n_i$ for all $i = 1, 2, \dots, k$, then the number of elements in the Cartesian product $A_1 \times A_2 \times \cdots \times A_k$ is $n_1 n_2 \cdots n_k$.

Example 2.2.5 (Product rule):
How many square matrices of order n can be formed with entries of either 0 or 1?. Such matrices are called $(0,1)$ matrices.

Each entry of a $(0,1)$ matrix has 2 possible choices of entries. But a square $(0,1)$ matrix of order n possesses $n \times n = n^2$ entries which are either 0 or 1. Hence by the product rule the number of possible $(0,1)$ matrices of order n is $\overbrace{2 \times 2 \times \cdots \times 2}^{n^2 \text{ times}} = 2^{n^2}$.

Example 2.2.6 (Product rule):
How many 3-letter words (words of length three like aaa, aab, $aac, \dots$, by not worrying about their meanings) can be formed from the English alphabet? How many 3-letter words on distinct letters (e.g. $abc, abd, \dots,$) can be formed?

We know that there are 26 letters in the English alphabet. The first letter of the word can be chosen in 26 ways, the second in 26 ways and the third also in 26 ways. Hence by the product rule (Proposition 2.2.3) the required number of 3-letter words is

$26 \times 26 \times 26 = 26^3$.

Let us now find the number of 3-letter words on distinct letters. The first letter can be chosen in 26 ways. Having chosen the first letter, the second one can be chosen in 25 ways. Having chosen the first and the second one, the third can be selected in 24 ways. Hence by the product rule, the number of 3-letter words on distinct letters is $26 \times 25 \times 24 = 15600$.

Example 2.2.7 (Counting the number of divisors of an integer): Find the number of positive divisors of the integer 600.

By *the fundamental theorem of arithmetic*, any integer > 1 can be written as the product of primes in only one way except for the order of prime factors. Let us write

$$600 = 2^3 \times 3 \times 5^2.$$

So each divisor of 600 must be of the form $2^i \times 3^j \times 5^k$, for all i, j, k with $0 \leq i \leq 3, \quad 0 \leq j \leq 1, \quad 0 \leq k \leq 2$. There are 4 possible values for i, 2 for j, and three for k. Hence by *the product rule*, the total number of possible positive divisors of 600 is $4 \times 2 \times 3 = 24$.

More generally, if $n = 2^{n_2} \times 3^{n_3} \times 5^{n_5} \times \cdots$, then the number of possible positive divisors of n is

$$(n_2 + 1)(n_3 + 1)(n_5 + 1) \cdots$$

Pigeon-hole principle

Of three *ordinary* people, two must have the same sex!

D. J. Kleitman

More generally, if $n+1$ letters are distributed among n letter boxes, then at least one of the letter boxes will have at least two letters.

In an abstract manner, if a set consisting of a “large” number of elements, is decomposed into a sufficiently small number of subsets then at least one of the subsets will have many elements of the set. More about this principle in Chapter 4.

Example 2.2.8:
Show that in any sequence of n integers, there is a subsequence of *consecutive* integers whose sum is exactly divisible by n.

Solution: We shall use the pigeon-hole principle to prove the result. Consider the sequence $(a_1, a_2, \ldots, a_n)$ of n integers and all of its possible n subsequences of consecutive integers (a_1), (a_1, a_2), (a_1, a_2, a_3), $, (a_1, a_2, \ldots, a_n)$. Let $s_1, s_2, \ldots, s_n$ be their respective sums, that is $s_i = \sum_{j=1}^{i} a_j$ for $1 \leq i \leq n$. If one of the $s_i' s$ is divisible by n then we are done. So we may suppose that *no* s_i is divisible by n (with remainder 0). Since the remainder 0 is excluded, the possible remainders of s_i's when divided by n must be among the integers $1, 2, \ldots, n-1$. Since there are n sums and only $n-1$ remainders, by *the pigeon-hole principle* (identify the sums with letters and the remainders with the letter boxes) there are integers p and q with $p < q$ such that the sums s_p and s_q leave the *same* remainder r when divided by n. Therefore we can write: $s_q = bn + r$ and $s_p = cn + r$. By subtraction, we have $s_q - s_p = (b - c)n$. This means that, $a_{p+1} + \cdots + a_q = (b - c)n$, a multiple of n.

The following example shows that *the decimal expansion* of a rational number either terminates or repeats. For example, the decimal representation of $1/2$ is 0.5, which terminates. On the other hand, the decimal representation of $1/3$ is $0.333\ldots$, which repeats indefinitely. The decimal representation of $1/7$ is $0.\overline{142857}$ where the bar indicates the repetition of the digits indefinitely.

More generally, if x is any real number, then x can be written as

$$x = n + 0.d_1 d_2 d_3 \ldots d_k \ldots$$

where n is an integer and d_i is a *digit* with $0 \leq d_i \leq 9$. To avoid ambiguity, we suppose that the sequence of digits doesn't end with infinitely many 9s; this is because, for example, $0.5 = 0.4999\ldots$. The above representation of x means that

$$n + \frac{d_1}{10} + \frac{d_2}{10^2} + \cdots + \frac{d_k}{10^k} \leq x < n + \frac{d_1}{10} + \frac{d_2}{10^2} + \cdots + \frac{d_k}{10^k} + \frac{1}{10^k}.$$

In other words, d_k is the *largest* digit satisfying the following inequality

$$n + \frac{d_1}{10} + \frac{d_2}{10^2} + \cdots + \frac{d_k}{10^k} \leq x \text{ for all integer } k > 0.$$

Example 2.2.9 (The decimal expansion of a rational number): Prove that the decimal expansion of a rational number a/b either terminates or repeats where a and b are positive integers.

Solution: By the *Euclidean division algorithm*, if we divide a by b we get the integer quotient q and the nonnegative integer remainder r, which is strictly less than the divisor b. Symbolically,

$$a = bq + r, \text{ with } 0 \leq r < b$$

If $r = 0$, then the division process terminates and we have $a/b = q$, in which case the decimal expansion is q, which is a terminating one.

If not, since we have exhausted the digits of a, we bring down 0. Note that the digit that we bring down is always 0. We continue the division process after bringing down 0 at each step. We must obtain one of the remainders $0, 1, \ldots, b-1$. If 0 is obtained after a finite number of divisions, then again the decimal expansion is a terminating one. If 0 is not obtained, then by the pigeon-hole principle after at most b steps one of the remainders must be repeated (since we are left with only $b-1$ possible remainders $1, 2, \ldots, b-1$). If the remainder repeats, the entire division process repeats.

We shall prove the following interesting theorem due to Erdös and Szekeres which may be considered as a generalization of the pigeon-hole principle.

Theorem 2.2.1 (Erdös and Szekeres):
Consider any sequence of $mn+1$ distinct integers $(a) = (a_1, a_2, \ldots, a_{mn+1})$. Then either the sequence (a) contains a (strictly) decreasing subsequence of $> m$ terms or the sequence (a) contains a (strictly) increasing subsequence of $> n$ terms.

Proof. We shall prove the theorem by contradiction. Suppose the theorem is false.

Let l_i^- be the number of integers in a longest decreasing subsequence of the given sequence (a) starting with the term a_i and let l_i^+ be the number of integers in a longest increasing subsequence of the sequence (a) beginning with the integer a_i. Since the theorem is assumed to be false, we have the inequalities: $l_i^- \leq m$ and $l_i^+ \leq n$ for each $i = 1, 2, \ldots, mn+1$.

We shall establish an injective function from the set $\{ a_1, a_2, \ldots, a_{mn+1} \}$ to the Cartesian product $\{ 1, 2, \cdots, m \} \times \{ 1, 2, \ldots, n \}$. To define an injective function f, we associate to each integer a_i the ordered pair (l_i^-, l_i^+), that is,

$$f(a_i) = (l_i^-, l_i^+) \text{ for each } i = 1, 2, \ldots, mn+1.$$

Since the theorem is assumed to be false, f is a function from the set $\{ a_1, a_2, \ldots, a_{mn+1} \}$ to the Cartesian product $\{ 1, 2, \cdots, m \} \times \{ 1, 2, \ldots, n \}$. We shall prove that f is injective. We have to show that if $a_i \neq a_j$ then $f(a_i) \neq f(a_j)$.

Consider two distinct integers a_i and a_j with $i < j$. We distinguish two cases:

Case 1: $a_i < a_j$.

By the definition of l_i^+ and l_j^+, we have $l_i^+ > l_j^+$ and hence $(l_i^-, l_i^+) \neq (l_j^-, l_j^+)$ (by the definition of equality of ordered n-tuples). That is, $f(a_i) \neq f(a_j)$.

Case 2: $a_i > a_j$.

By the definition of l_i^- and l_j^-, we have $l_i^- > l_j^-$ and hence $(l_i^-, l_i^+) \neq (l_j^-, l_j^+)$ (by the definition of equality of ordered n-tuples). That is, $f(a_i) \neq f(a_j)$.

Hence f is an injective function. But by the Proposition 2.2.1, we have

$$|\{ a_1, a_2, \ldots, a_{mn+1} \}| \leq |\{ 1, 2, \ldots, m \} \times \{ 1, 2, \ldots, n \}|$$

That is, $mn + 1 \leq mn$, a contradiction. ∎

The number of functions from an n-element set to an m element set:

Theorem 2.2.2:
Let A and B be two sets with $A = \{1, 2, \ldots, n\}$ and $B = \{1, 2, \ldots, m\}$. The number of functions from the set A to the set B is $|B|^{|A|} = m^n$.

Proof. A function f from A to B can be viewed as an element of the Cartesian product $B^n = \overbrace{B \times B \times \cdots \times B}^{n \text{ times}}$ (second interpretation of functions). Hence the set of all functions from A to B is the set of all ordered n-tuples of elements of B. In particular, the number of functions from A to B is $|B^n| = \overbrace{|B| \times |B| \times \cdots \times |B|}^{n \text{ times}} = m^n$. ■

Because of the above theorem, the set of all functions from a set A to a set B is denoted by B^A. Symbolically,

$$B^A = \{ f \mid f : A \to B \}.$$

A function from A to B is sometimes traditionally called a *permutation with repetition* of elements of B taken n at a time. Here n is the number of elements of the domain set A. An n-set is a set of n elements. An m-subset of a set A is a subset of A with exactly m elements.

Subtraction Rule:

Example 2.2.10 (Rule of subtraction):
Consider a biology class of 40 students in which 25 students are girls. Then the number of boys in the class is $40 - 25 = 15$. The rule of sum generalizes this simple example.

Fact 2.2.2.1:
Let S be a finite set and let A be a *subset* of S. Then the number of elements of the set S *not* in the subset A is

$$S \setminus A = |S| - |A|.$$

Example 2.2.11 (Subtraction rule):
Find the number of integers between 100 and 1000 in which there is at least one digit 3 present.

Solution: Any integer between 100 and 1000 containing a digit 3 must have 3 digits.

Total number of 3-digit integers with at least one digit 3 = Total number of 3-digit integers − Total number of 3-digit numbers in which 3 is not present at all.

Let us find the total number of 3-digit integers: The digits to be used are $0, 1, 2, \ldots, 9$. The hundred's place can be chosen in 9 ways (because it can't be 0). Each of the tenth and first place can be chosen in 10 ways. Hence by the product rule, the total number of 3-digit integers is equal to $9 \times 10 \times 10 = 900$.

Let us now find the total number of 3-digit integers in which the digit 3 is not present at all: We can only use the 9 digits $0, 1, 2, 4, 5, 6, 7, 8, 9$. The hundredth place can be chosen in 8 ways (because 0 must not be there). Each of the tenth and first place can be chosen in 9 ways. Hence the number of integers without the digit 3 is equal to $8 \times 9 \times 9 = 648$.

Hence the required number is $900 - 648 = 252$.

Division rule

Example 2.2.12 (Counting the number of cows by counting their legs):
The number of cows in a shed is *the same as* the total number of their legs divided by 4. (Assuming that every cow has exactly four legs!)

Example 2.2.13 (Division rule):
Seventy-five toffees are distributed equally among the children. If each child receives exactly 5 toffees then the total number of children present is

$$\frac{75}{5} = 15.$$

The division rule generalizes the above simple examples.

Fact 2.2.2.2:
Consider a multiset M of m elements in which each element is repeated k times (that is, each element has multiplicity k). Then the total number of *distinct* elements of the multiset M is given by the quotient

$$\frac{m}{k}.$$

Example 2.2.14:
Consider the multiset $M = \{\,3.a, 3.b, 3.c, 3.d\,\} = \{\,a, a, a, b, b, b, c, c, c, d, d, d\,\}$ consisting of 12 elements in which each element has multiplicity 3. Then the number of *distinct* elements of M is $\frac{12}{3} = 4$.

2.3 Permutations and Combinations

A *permutation* of an m-set B taken n at a time is simply an *injective function* from the set of n elements $\{\,1, 2, \cdots, n\,\}$ to the set B. According to the second interpretation of functions as Cartesian products, a permutation of an m-set B taken n at a time is an element of the Cartesian product/word of *length* n whose components (or letters) are *distinct* elements of the set B (B is the alphabet).

Example 2.3.1 (Permutation):
Consider the set $B = \{\,a, b, c\,\}$ of three elements. Let us find the set of *all* permutations of the elements of B taken 2 at a time. It is the set of *all* elements of the Cartesian product $B^2 = B \times B = \{\,aa, ab, ac, ba, bb, bc, ca, cb, cc\,\}$ whose components are distinct. Hence the required permutations or arrangements are:

$a\ b, a\ c, b\ a, b\ c, c\ a, c\ b$. They are six in number. In other words, there are 6 injective functions $f_1, f_2, \ldots, f_6$ where $f_1(1) = a, f_1(2) = b,\quad f_2(1) = a, f_2(2) = c,\quad f_3(1) = b, f_3(2) = a,$ $f_4(1) = b, f_4(2) = c,\quad f_5(1) = c, f_5(2) = a, f_6(1) = c, f_6(2) = b$.

Method of constructing permutations of n elements in an inductive manner:

We shall illustrate this method by a simple example.

Example 2.3.2 (Permutation construction):
The permutation of the 1-set $\{1\}$ is simply 1.

The permutations of the 2-element set $\{1, 2\}$ are 1 2, and 2 1. This is obtained by inserting the element 2 in all possible places (before 1 and after 1) of the permutation of the 1-set.

From these permutations of the 2-set, we now construct all permutations of the 3-set $\{1, 2, 3\}$ in the following way.

For each permutation of the set $\{1, 2\}$, construct three other permutations by inserting the number 3 in every possible place, getting

3 1 2, 1 3 2, 1 2 3
3 2 1, ,2 3 1 ,2 1 3.

To construct all permutations of the 4-set $\{1, 2, 3, 4\}$, we now insert the number 4 in all possible places in each of the permutations of the 3-set $\{1, 2, 3\}$. This process can be continued indefinitely.

The number of injective functions from an n-set to an m-set: P_{nm}

The following theorem gives the formula for the number of permutations of an m-set taken n at a time.

Theorem 2.3.1:
The number of injective functions from an n-set to an m-set ($n \leq m$ is $m(m-1)\cdots(m-n+1)$. In other words, the number of permutations of m objects taken n at a time is $P_{nm} = m(m-1)\cdots(m-n+1)$.

Proof. The number of injective functions from an n-set $\{1, 2, \ldots, n\}$ to an m-set $B = \{1, 2, \ldots, m\}$ is the number of ordered n tuples $(b_1, b_2, \ldots, b_n)$ of *distinct* elements of the set B.

The first component b_1 can be selected from the elements of B in m ways, since it can be either 1 or 2 or etc. or m. After having chosen the first component of B^n, we have $m-1$ choices for the second component b_2 (since the components must be distinct). Having chosen the first two components, there are $m-2$ choices for the third component, b_3, etc. Finally, we have $m-(n-1)$ choices for the last component b_n. Hence by the product rule (Proposition 2.2.2), the number of distinct ordered n tuples of elements is $m(m-1)\cdots(m-n+1)$. ■

The number $m(m-1)\cdots(m-n+1)$ is called a *falling factorial* and is denoted by $m_{(n)}$ or $m^{\underline{n}}$ (read m to the n falling) by Knuth. We define the empty product $m^{\underline{0}} = 1$ for $m \neq 0$. In particular, the number $m_{(m)} = m(m-1)\cdots 2.1$ is the product of all positive integers from 1 to m. It is called the factorial of m or m *factorial* and is denoted simply by $m!$ instead of $m_{(m)}$. For the sake of convenience we define $0! = 1$.

A *recurrence relation* allows us to calculate the terms of a sequence $(f_n)_{n=0}^{\infty}$ from the given initial values and the previously computed values.

The factorial function can be defined recursively as follows:

$$n! = \begin{cases} 1 & \text{if } n = 0 \text{ (basis or initialization)} \\ n(n-1)! & \text{if } n \geq 1 \text{ (recurrence)} \end{cases}$$

A bijective function/one-to-one correspondence from an n-set onto itself is called a *permutation* of the n-set.

Corollary 2.3.1.1:
The number of permutations of n elements is $n!$ In other words, the number of bijections from an n-set onto itself is $n!$

Proof. An injective function from an n-set to itself is necessarily surjective and hence bijective. By Theorem 2.3.1, the number of such bijections is $n^{\underline{n}}$ which is $n!$ ■

Stirling approximation for $n!$:
The factorial function grows very rapidly. For example, $0! = 1$, $1! = 1$, $2! = 2$, $3! = 6$, $4! = 24$, $5! = 120$, $6! = 720$, $,7! = 5040$, $8! = 40320$, $9! = 362880$,
$10! = 3628800$.

The following formula due to Stirling gives an approximation of $n!$

$$n! \approx n^n e^{-n}\sqrt{2n\pi} \tag{2.1}$$

The symbol ≈ means "approximately equal" and the number e denotes the base of the natural logarithm. In fact, e is an irrational number and is defined by the infinite series

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots \approx 2.718281 \cdots .$$

π is the "circle ratio"(ratio of the circumference of a circle to its diameter), which is an irrational number $\approx 3.14159 \cdots$.

Example 2.3.3 (Approximation for 8!):
The exact value of 8! is 40320. Its approximate value is (by formula 2.1) $8^8 e^{-8}\sqrt{2 \times 8 \times \pi} \approx 39902$.

Example 2.3.4 (Permutation):
The set of all permutations of the 3-set $\{a, b, c\}$ are $a\quad b\quad c$, $a\quad c\quad b$, $b\quad a\quad c$, $b\quad c\quad a$, $c\quad a\quad b$, $c\quad b\quad a$. There are $3! = 6$ permutations of a 3-set.

The number of k-subsets of an n-set: $\binom{n}{k}$ (read "n choose k")

We are interested in finding a formula for the number of k-subsets of an n-set. This *integer* is also called the number of combinations of n objects taken k at a time and is denoted by $\binom{n}{k}$ (read "n choose k"). The *integers* $\binom{n}{k}$ are called *binomial coefficients.*

Binomial coefficients in detail were known to Hindu Mathematicians Bhaskara Acharya (Bhaskara Acharya's book: *Lilavati*) and Halayudha's tenth-century commentary (Halayudha commentary on an ancient Hindu classic: *Pingala's Chanda-Sutra*). These are the earliest known detailed and clear discussions on binomial coefficients (see [1]). Note that in a set, the order of the elements is irrelevant whereas in the idea of a permutation, the order of the elements is taken into account. After all, permutations are certain Cartesian products.

Hence in the notion of combination, the order of elements is *irrelevant.* Let us find the number of 2-subsets of a 4-element set. The following example illustrates the method.

Example 2.3.5 (Combination):
Consider the 4-element set $A = \{1, 2, 3, 4\}$. We have to find the number of 2-subsets of A.

The number of permutations of the elements of A taken 2 at a time is $4_2 = 4 \times 3 = 12$. But the two permutations $a \quad b = (a, b)$ and $b \quad a = (b, a)$ of A taken 2 at a time defines only one 2-element subset $\{a, b\}$ for $a, b \in A$ with $a \neq b$ (because the order of elements in a set is irrelevant.) Conversely, each 2-subset of A defines 2!=2 permutations of A taken two at a time. Hence we have the equality $\binom{4}{2} = 12/2 = 6$. Hence there are six 2-subsets of a 4-set.

With this example as a model, we prove the following theorem.

Theorem 2.3.2:
The number of k-subsets of an n-set is $\binom{n}{k}$ $=n(n-1)\cdots(n-k+1)/k! = n!/k!(n-k)!$. $(0 \leq k \leq n)$

Proof. Consider an n-set $A = \{1, 2, \ldots, n\}$. By Theorem 2.3.1, the number of permutations of the set A taken k at a time is $n_{(k)} = n(n-1)\cdots(n-k+1)$. Each k-subset of A defines $k!$ permutations of 1,2,…,n taken k at a time, and conversely $k!$ permutations of $a_1, a_2, \ldots, a_k$ $(a_i \in A$ and $a_i \neq a_j)$ defines only one k-subset $\{a_1, a_2, \ldots, a_k\}$ of the set A. Hence we have the equality $\binom{n}{k} = n_{(k)}/k! = n(n-1)\ldots(n-k+1)/k!$. Multiplying the numerator and the denominator of the right-hand side by $(n-k)!$, we obtain $\binom{n}{k}= n!/k!(n-k)!$. ■

Note that $\binom{n}{k} = 0$ if $k > n$ by definition. Similarly, $\binom{n}{k} = 0$ if $k < 0$. $\binom{0}{0} = 0!/0!(0-0)! = 1$. $\binom{0}{k} = 0$ if $k > 0$.

Theorem 2.3.2 can be written (using 2.3.1 and 2.3.1.1) strikingly as follows:

The number of k-subsets of an n-set is equal to the number of injections from a k-set to the n-set divided by the number of bijections from a k-set onto itself.

Example 2.3.6 (Binomial coefficients):
Find the number of triangles that can be formed in a *convex polygon* of n sides.

Solution: A polygon is convex if the segment joining any two points on or inside the polygon lies entirely in the polygon. Intuitively, a convex set is one in which we cannot play *hide and seek*. Let the vertex set of the polygon be $V = \{ 1, 2, \ldots, n\}$. Note that in a convex polygon no three vertices are collinear. Hence a triangle is obtained by choosing/joining any three vertices of the set V. But by the definition of the binomial coefficient a three-element subset of an n-set can be chosen in $\binom{n}{3}$ ways. Hence the number of triangles formed is $\binom{n}{3} = n(n-1)(n-2)/3!$

Example 2.3.7 (Binomial coefficients):
Find the number of diagonals of a polygon of n sides.

Solution: Let the vertex set of the polygon be $V = \{ 1, 2, \ldots, n\}$. By joining any two distinct vertices of the polygon we get either a *side* of the polygon which can be any segment of the set $S = \{ 12, 23, 34, \ldots, (n-1)n, n1 \}$ or else a diagonal of the polygon. But the number of line segments that can be obtained by joining/choosing any two vertices is the same as the number of two-element subsets of the vertex set V which is the binomial coefficient $\binom{n}{2} = n(n-1)/2$. Among these $n(n-1)/2$ segments, exactly n line segments are the sides of the polygon. Hence the number of diagonals $= n(n-1)/2 - n = n(n-3)/2$.

Definition 2.3.1 (Extension of binomial coefficients):
The formula of Theorem 2.3.2, may be used to define binomial coefficients $\binom{n}{k}$ when n is *not* a non-negative integer. More precisely, for any real number x, and any non-negative integer k, we define

$$\binom{x}{k} = \frac{x(x-1)\cdots(x-k+1)}{k(k-1)\ldots(1)} = \frac{x^{\underline{k}}}{k!}$$

$$\binom{x}{k} = 0 \text{ if } k < 0$$

Corollary 2.3.2.1:
The product of k consecutive positive integers is divisible by $k!$.

Proof. Consider any k successive positive integers $n+1, n+2, \cdots,$ $n+k$ for $n \geq 0$. We have to prove that the product $(n+1) \times (n+2) \times \cdots \times \times (n+k)$ is divisible by $k!$. In other words, it is to be proved that $\frac{(n+1)\times(n+2)\times\cdots\times\times(n+k)}{k!}$ is an integer. But this quotient is equal to the binomial coefficient $\binom{n+k}{k}$(by Theorem 2.3.2.) But the binomial coefficients are integers by definition. Hence the corollary. ∎

A positive integer $p \geq 2$ is called a *prime number* if its only positive divisors are unity and itself.

Corollary 2.3.2.2:
If p is a prime number, then the binomial coefficients

$$\binom{p}{1}, \binom{p}{2}, \ldots, \binom{p}{p-1}$$

are all divisible by p.

Proof. Let k be an integer such that $1 \leq k \leq p-1$. We have to show that p divides $\binom{p}{k}$. By Corollary 2.3.2.1, $k!$ divides the product $p(p-1)\cdots(p-k+1)$. In other words, the quotient

$$p \times \frac{(p-1)(p-2)\cdots(p-k+1)}{k!}$$

is an integer. Since p is prime and $k < p$, we have, $k, k-1, \ldots, 2, 1$ are all relatively prime to p. In particular, $k!$ is relatively prime to p. Therefore in the above quotient, $k!$ divides the product $(p-1)(p-2)\cdots(p-k+1)$ (since if c divides ab and if a and c are relatively prime, then the integer c divides the integer b.). That is, $\frac{(p-1)(p-2)\cdots(p-k+1)}{k!}$ is an *integer.* But then, $\binom{p}{k}/p = \frac{(p-1)(p-2)\cdots(p-k+1)}{k!}$ Hence the corollary. ∎

Example 2.3.8 (Duality relation: Binomial coefficients):
$\binom{n}{0}$=$n!/0!(n-0)! = 1$(since $0! = 1$). Similarly, $\binom{n}{n} = 1$. $\binom{n}{n-k} =$

$n!/(n-k)!(n-(n-k))! = n!/k!(n-k)! = \binom{n}{k}$ Thus we have the duality relation of the binomial coefficients: $\binom{n}{k} = \binom{n}{n-k}$. There is a nice combinatorial interpretation of this duality relation: every time we select a k-subset of an n-set, we indirectly reject an $(n-k)$-subset (the complement of the k-subset we select) of the n-set.

Example 2.3.9:
If m horizontal parallel lines are cut by n vertical parallel lines, find the number of parallelograms that can be obtained.

Solution: To form a parallelogram, we need two horizontal parallel lines and two vertical parallel lines. Two horizontal lines can be chosen from among the m lines in $\binom{m}{2}$ ways and similarly two vertical lines can be chosen from n lines in $\binom{n}{2}$ ways. Hence by the product rule (Proposition 2.2.2), the number of parallelograms obtained is $\binom{m}{2} \times \binom{n}{2} = \frac{m(m-1)}{2} \times \frac{n(n-1)}{2} = \frac{mn(m-1)(n-1)}{4}$.

The number of k- multisubsets of an n-set:

The principal utility of the notion of a set is the following: Given an element, we should be able to say if the given element belongs to the set or not. So, in a set, the occurrence of an element is *not* repeated more than once.

If we allow the occurrence of an element more than once, the set is called a *multiset* and the number of occurrences of an element in a multiset is called its *multiplicity.*

Consider a finite set A. A set B is called a *multisubset* of A, if every element of B is in the set A and an element of B may occur more than once in B. If the multisubset B of A has exactly k elements (multiplicity of each element of B is counted), then B is a k-multiset of A. A set can be viewed as a multiset with the multiplicity of each element 1.

Example 2.3.10 (Multisubset):
Let $A = \{1, 2, 3, 4\}$. Then $B = \{1, 1, 1, 3, 4, 4\}$ is a 6-multisubset of A. The multiplicity of the element 1 is 3, of the element 3 is 1, and of the element 4 is 2. Another multiset (in fact simply a subset) of A is $\{1, 3\}$ where the multiplicity of each element is 1.

We are interested in finding a formula for the number of k-multisubsets of an n-set. We employ a technique called *bijective proof*. In this proof, we establish a one-to-one correspondence between the set of all k-multisubsets of an n-set and the set of all k-subsets of an $n+k-1$-set.

Theorem 2.3.3:
The number of k-multisubsets of an n-set is $\binom{n+k-1}{k}$.

Proof. (Bijective proof) Consider an n-set $A = \{1, 2, \ldots, n\}$. A k-multisubset B of A can be written in the form

$$B = \{b_1, b_2, \ldots, b_k\}$$

where the b_i elements of the set B satisfy the following inequalities:

$$1 \leq b_1 \leq b_2 \leq b_3 \cdots \leq b_k \leq n$$

Note that if the strict inequality holds everywhere then the set B becomes simply a k-subset of A. Consider the $n+k-1$-set

$$A' = \{1, 2, \ldots, n+k-1\}$$

We shall now establish a one-to-one correspondence between the k-multisubsets of A and k-subsets of A'.

Given a k-multisubset B of the n-set A, we shall produce a k-subset B' of the $n+k-1$-set A' by using B. Let $B = \{b_1, b_2, \ldots, b_k\}$ be a k-multisubset of A with

$$1 \leq b_1 \leq b_2 \leq \cdots \leq b_i \leq \cdots \leq b_k \leq n$$

Now the set $B' = \{b_1 + 0, b_2 + 1, \ldots, b_i + i - 1, \ldots, b_k + k - 1\}$ is certainly a k-multisubset of A', because $1 \leq b_i \leq n$ and hence $1 \leq b_i + i - 1 \leq n + i - 1 \leq n + k - 1$ for all i with $1 \leq i \leq k$. We now show that B' is a k-subset of A', that is, the elements of B' are *distinct*. If not, suppose $b_i + i - 1 = b_j + j - 1$ with $i < j$. Since $i < j$, we have $b_i \leq b_j$. But then the equation $b_i + i - 1 = b_j + j - 1$ implies $b_i - b_j = j - i$. This is impossible since the left-hand side of this equality $b_i - b_j$ is ≤ 0 while the expression on the right-hand side is > 0. Hence B' is a k-subset of A'.

Conversely, consider now a k-subset $B' = \{ b_1, b_2, \ldots, b_k \}$ of A' with

$$1 \leq b_1 < b_2 < \cdots < b_i \cdots < b_k \leq n + k - 1.$$

We shall construct a k-multisubset B of A with the help of the k-subset B'. Define $B' = \{ b_1 - 0, b_2 - 1, \ldots, b_i - (i-1), \ldots, b_k - (k-1) \}$. We shall prove that B' is a k-multisubset of A. It is enough if we prove $1 \leq b_i - (i-1) \leq n$ for all i with $1 \leq i \leq k$.

First we prove that $b_i - (i-1) \geq 1$ for all i with $1 \leq i \leq k$ by induction on i. This is clearly true for $i = 1$ for $b_1 - (1-1) = b_1 \geq 1$. Suppose $b_p - (p-1) \geq 1$ with $1 \leq p < k$. We shall prove it for $p+1$. That is, we have to prove $b_{p+1} - (p+1-1) \geq 1$. Since $b_p < b_{p+1}$, we have $b_p - p < b_{p+1} - p$ and hence $b_p - p + 1 \leq b_{p+1} - p$. By induction hypothesis, $b_p - p + 1 \geq 1$. Hence $b_{p+1} - p \geq 1$.

Next we prove $b_i - (i-1) \leq n$ for all i with $1 \leq i \leq k$. This is true for $i = k$ for $b_k \leq n + k - 1$ and hence $b_k - k + 1 \leq n$. Now $b_{k-1} < b_k$. Hence $b_{k-1} - k < b_k - k$. This implies that $b_{k-1} - k + 1 \leq b_k - k$ and therefore $b_{k-1} - k + 2 \leq b_k - k + 1 \leq n$. Hence the inequality is proved for $i = k - 1$. Similarly we prove the inequality successively for $i = k-2, k-3, \ldots, 1$.

Hence we have established a bijection from the set of all k-multisubsets of the n-set A onto the k-subsets of the $(n+k-1)$-set A'. But the number of k-subsets of the $(n+k-1)$-set is $\binom{n+k-1}{k}$ by definition. Hence by Proposition 2.2.1, the number of k-multisubsets of the n-set is also equal to $\binom{n+k-1}{k}$. ■

Example 2.3.11 (Number of multisubsets):
The number of 2-multisubsets of the 3-set $\{ 1, 2, 3 \}$ is $\binom{3+2-1}{2} = \binom{4}{2} = 4 \times 3/2! = 6$. These 2-multisubsets are $\{ 1, 2 \}, \{ 1, 3 \}, \{ 2, 3 \}, \{ 1, 1 \}, \{ 2, 2 \}, \{ 3, 3 \}$.

Example 2.3.12:
If a set of k-dice are thrown simultaneously, find the number of possible outcomes.

Solution: The faces of each die are marked with the integers 1,2,3,4,5,6. Hence an outcome of a single throw of a set of k-dice may be viewed as a k-multisubset of the 6-set $\{ 1, 2, 3, 4, 5, 6 \}$ (e.g., possible outcomes are $\overbrace{\{1,1,\dots,1\}}^{k\ 1's}, \overbrace{\{1,1,\dots,1,2\}}^{(k-1)\ 1's}$, etc.). Hence the number of possible outcomes is (Theorem 2.3.3) $\binom{6+k-1}{k} = \binom{k+5}{k} = \binom{k+5}{5}$ by the duality relation of the binomial coefficients (Example 2.3.8).

An interpretation of the binomial coefficients as the number of nonnegative integral solutions of a homogeneous linear equation:

Consider the following equation:

$$x_1 + x_2 + \cdots + x_n = k$$

where k is a positive integer. How many nonnegative integral solutions $(x_1, x_2, \dots, x_n)$ does the above equation possess?

We shall see that there is a one-to-one correspondence between the nonnegative integral solutions of the equation $x_1 + x_2 + \ldots + x_n = k$ and the k- multisubset of n-set $\{a_1, a_2, \dots, a_n\}$.

If $(x_1, x_2, \dots, x_n)$ is a nonnegative integral solution of the equation, we construct a k-multisubset of $\{a_1, a_2, \dots, a_n\}$ by choosing the element a_i with multiplicity x_i for $1 \leq i \leq n$. Conversely, given a k-multisubset S of $\{a_1, a_2, \dots, a_n\}$ we assign to the variable x_i the value of the multiplicity of the element a_i occurring in the k-multisubset S. Hence the number of nonnegative integral solutions of the equation $x_1 + x_2 + \cdots + x_n = k$ is $\binom{n+k-1}{k}$.

Example 2.3.13:

Find the number of *positive* integral solutions of the equation $x_1 + x_2 + x_3 + x_4 = 9$.

We have a formula for the number of *nonnegative* integral solutions of a linear homogeneous equation but here we have to determine the number of *positive* integral solutions. To use our formula, we perform the following change of variables:

$y_1 = x_1 - 1 \quad y_2 = x_2 - 1 \quad y_3 = x_3 - 1 \quad y_4 = x_4 - 1$. Plugging these into our initial equation, we get the equation $y_1+y_2+y_3+y_4 = 5$ where the y_i variables are nonnegative. Note that (x_1, x_2, x_3, x_4) is a positive integral solution of $x_1 + x_2 + x_3 + x_4 = 9$ if and only if (y_1, y_2, y_3, y_4) is a nonnegative integral solution of the equation

$y_1 + y_2 + y_3 + y_4 = 5$. Hence the number of desired solutions is $\binom{4+5-1}{5} = 56$.

Power set

The set of all subsets of a given set A is often called the power set of the set A and is denoted by $\mathcal{P}(A)$ or 2^A.

EXAMPLE 2.3.14 (Power set):
Let $A = \{1, 2, 3\}$. Then the set of all subsets of A is

$$\mathcal{P}(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$$

Note that each element of $\mathcal{P}(A)$ is a subset of the set A. There are $2^3 = 8$ subsets of A.

The following Table 2.3 summarizes our results for various kinds of selections—ordered and unordered, with and without repetition—of k-objects out of n-given objects.

Table 2.3: Number of ordered and unordered selections of k things out of n things

	Ordered/Functions/ Cartesian product	Unordered/subsets/ multisubsets
Without repetition	$n(n-1)\cdots(n-k+1)$	$\binom{n}{k}$
With repetition	n^k	$\binom{n+k-1}{k}$

Characteristic function associated with a subset S: χ_S
Consider a subset S of a given finite set A. Then the characteristic function of the set S, denoted by χ_S, is the function from the set A to the set $\{0, 1\}$ defined as follows:

$$\chi_S(a) = \begin{cases} 1 & \text{if } a \in S \\ 0 & \text{otherwise.} \end{cases}$$

Example 2.3.15 (Characteristic function):
Consider a non-empty finite set A. Then $\chi_\emptyset$ is the function which takes the value 0 on each element of the set A. On the other hand, χ_A is the characteristic function which assigns the value 1 to each element of the set A.

We shall prove the following theorem which establishes a bijection between the set of all characteristic functions on an n-set and the set of all ordered n-tuples of 0's and 1's.

Theorem 2.3.4:
There is a natural one-to-one correspondence between the set of all characteristic functions defined on an n-set and the set of all ordered n-tuples of 0's and 1's.

Proof. Consider the n-set $A = \{\,1, 2, \ldots, n\,\}$ and a subset S of A.

To the characteristic function χ_S, we associate an ordered n-tuple $(a_1, a_2, \ldots, a_n)$ where

$$a_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{otherwise.} \end{cases}$$

Conversely, given an ordered n-tuple $(a_1, a_2, \ldots, a_n)$ of 0's and 1's, we define a subset S of A by

$$S = \{\, i \mid a_i = 1\,\}.$$

Now we associate the characteristic function χ_S to the ordered n-tuple $(a_1, a_2, \ldots, a_n)$ This completes the proof. ■

Corollary 2.3.4.1:
The number of characteristic functions defined on an n-set is 2^n.

Proof. By the product rule (Proposition 2.2.2), the number of possible n-tuples of 0's and 1's is $\overbrace{2 \times 2 \times \cdots \times 2}^{n \text{ times}} = 2^n$ (since each of the n components of an n-tuple has 2 possible choices:0 or 1). But by Theorem 2.3.4, there is a one-to-one correspondence between the set of all characteristic functions on an n-set and the set of all ordered n-tuples of 0's and 1's. Hence the number of characteristic functions on an n-set is 2^n. ■

Corollary 2.3.4.2:
The total number of subsets of an n-set is 2^n.

Proof. Consider an n-set $A = \{1, 2, \ldots, n\}$. To each subset S of A we associate its characteristic function χ_S.

Conversely, to each characteristic function χ_S the subset S is obtained by $S = \{ i \mid a_i = 1 \}$.

This establishes a bijection between the subsets of A and the characteristic functions on A. In particular, the number of subsets of A is equal to the number of characteristic functions on A. By Corollary 2.3, the number of subsets of A is 2^n. ■

Because of the above corollary, the power set of a given finite set A is denoted by 2^A.

The following elementary identity is obtained by counting the number of subsets of an n-set in two different ways. This technique of counting the same quantity in at least two different ways is frequently employed in combinatorics.

Corollary 2.3.4.3:
For any positive integer n, we have $\binom{n}{0}+\binom{n}{1}+\cdots+\binom{n}{n}=2^n$.

Proof. Consider an n-set A. The number of subsets of A is equal to the sum of the number of 0-subsets of A + the number of 1-subsets of A + $\cdots$ + the number of n-subsets of A. But by definition, the number of k-subsets of an n-set is $\binom{n}{k}$ for each k with $0 \leq k \leq n$. Hence the total number of subsets of an n-set is the sum $\binom{n}{0}+\binom{n}{1}+\cdots+\binom{n}{n.}$

But by Corollary 2.3.4.2, the number of subsets of an n-set is 2^n. Therefore, equating the two expressions for the total number of subsets, we get

$$\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{n} = 2^n.$$

■

We now prove the following addition formula (recurrence relation) of binomial coefficients.

Lemma 2.3.1:
$\binom{n}{k-1} + \binom{n}{k} = \binom{n+1}{k}$ for all integers $n \geq 0$ and all integers k with $0 \leq k \leq n$.

Proof. Note that the identity is trivially true if $k = 0$, because $\binom{n}{-1} = 0$ and $\binom{n}{0} = 1$. So, suppose $k \geq 1$.

First proof:

The proof of the identity follows by using the formula $\binom{n}{k} = n!/k!(n-k)!$ to replace the expression on the left-hand side of the identity to obtain the right-hand side.

$$\begin{aligned}
\binom{n}{k-1} + \binom{n}{k} &= n!/(k-1)!(n-k+1)! + n!/k!(n-k)! \\
&= n!/k!(n-k)!\,((k/n-k+1)+1) \\
&\qquad \text{using } p! = p(p-1)! \\
&= n!/k!(n-k)!\,(n+1/n-k+1) \\
&= (n+1)!/k!(n-k+1)! \\
&\qquad \text{using again } p(p-1)! = p! \\
&= \binom{n+1}{k} \\
&\qquad\quad \text{using } p!/k!(p-k)! = \tbinom{p}{k}.
\end{aligned}$$

Second proof:

This proof uses the definition of $\binom{n}{k}$ and is combinatorial in nature. Consider an $(n+1)$-set $A = \{1, 2, \ldots, n+1\}$. By definition, $\binom{n+1}{k}$ is the number of k-subsets of the $(n+1)$-set A. We distinguish two cases:

Case 1: The k-subsets of A containing a particular element, say, $n+1$. The number of k-subsets of A containing the element $n+1$ is equal to the number of $(k-1)$-subsets of the n-set $\{1, 2, \dots, n\}$ (since a $(k-1)$-subset of $A \setminus \{n+1\}$ together with the singleton $n+1$ gives a k-subset of A and conversely the removal of the element $n+1$ from a k-subset of A containing the element $n+1$ gives a $k-1$ subset of $A \setminus \{n+1\}$). Hence the total number of k-subsets of A containing the particular element $n+1$ is $\binom{n}{k-1}$.

Case 2: The k-subsets of A *not* containing the particular element $n+1$.

The number of k-subsets of A not containing the element $n+1$ is the same as the number of k-subsets of the n-set $A \setminus \{n+1\} = \{1, 2, \dots, n\}$ which is by definition $\binom{n}{k}$.

Note that these two cases are mutually exclusive and they exhaust all the possibilities. Hence by the sum rule 2.2.0.1, we have

$$\binom{n}{k-1} + \binom{n}{k} = \binom{n+1}{k}$$

■

Lemma 2.3.1 can be used to find the binomial coefficients *recursively* starting from the initial conditions $\binom{n}{0} = 1$ and $\binom{n}{1} = n$. In Table 2.4, called *Pascal's triangle*, the nth line gives the binomial coefficients $\binom{n}{0}, \binom{n}{1}, \dots, \binom{n}{n}$ for $n = 0, 1, 2, 3$. (Note that the first line is the 0th line.) For example, the 5th line gives the binomial coefficients $\binom{4}{0}, \binom{4}{1}, \binom{4}{2}, \binom{4}{3}, \binom{4}{4}$. Each coefficient is obtained by adding the two coefficients found in the north-east and north-west direction. This table may be continued to any number of lines.

2.3.1 Sums and Products

Sums:

Consider a sequence of real numbers $a_1, a_2, \dots$. In fact, this sequence may be considered as a function a from the set of natural numbers to the set of real numbers. The value of the function a at the integer i is denoted by the *subscript notation* a_i instead $a(i)$.

Table 2.4: Pascal Triangle

$$\begin{array}{ccccccccc}
 & & & & 1 & & & & \\
 & & & 1 & & 1 & & & \\
 & & 1 & & 2 & & 1 & & \\
 & 1 & & 3 & & 3 & & 1 & \\
1 & & 4 & & 6 & & 4 & & 1 \\
 & & & & \vdots & & & &
\end{array}$$

The sum $a_1 + a_2 + \cdots + a_n$ is denoted by the symbolism $\sum_{i=1}^{n} a_i$ or $\sum_{1 \leq i \leq n} a_i$. The variable i in the summation is called a *dummy variable or index variable.* This index variable may be replaced by any other variable j. Hence $\sum_{i=1}^{n} a_i = \sum_{j=1}^{n} a_j$. If $n = 0$ then the sum is defined to be zero.

More generally, the symbolism

$$\sum_{P(i)} a_i$$

means the sum of all the numbers a_i, where i is an integer satisfying the property $P(i)$ ($P(i)$ is a property of the integer i). If there is no integer i satisfying the property $P(i)$, then the sum is defined to be zero.

Example 2.3.16 (On sums):

$$\sum_{\substack{1 \leq i \leq 15 \\ i,\ \text{prime}}} a_i$$

denotes the sum $a_2 + a_3 + a_5 + a_7 + a_{11} + a_{13}$. Here the property to be satisfied by the index variable i should be an integer such that $1 \leq i \leq 15$ and i must be a prime number.

The notation $\sum_{1 \leq i \leq n} a_i$ where $n = 3.14$ means $a_1 + a_2 + a_3$.

Double summation:
Interchange of the order of summation:

Consider the mn numbers a_{ij} for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. (These mn numbers may be viewed as a function a from the Cartesian product $\{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}$ to the set of real numbers. The value of the function a at the pair (i, j) is denoted by using the subscript notation a_{ij}.) Then the double summation

$\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}$ means $\sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} \right)$. In this summation, we first fix the value of the index i and vary the index j from 1 to n and then vary the index i from 1 to m. To understand this, let us consider the following special case.

If $m = 3$ and $n = 2$, then $\sum_{i=1}^{3} \sum_{j=1}^{2} a_{ij} = \sum_{i=1}^{3} \left(\sum_{j=1}^{2} a_{ij} \right) = \sum_{i=1}^{3} (a_{i1} + a_{i2}) = (a_{11} + a_{12}) + (a_{21} + a_{22}) + (a_{31} + a_{32})$.

We may view the mn numbers arranged in the form of the rectangular array (called "matrix") $A = (a_{ij})$ where the number a_{ij} is in the intersection of the ith line and the jth column of the matrix A.

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}$$

By summing the numbers a_{ij} of the matrix M in two different ways, first row by row and then summing the numbers (a_{ij}) column by column we obtain

$$\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} = \sum_{j=1}^{n} \sum_{i=1}^{m} a_{ij}$$

Thus we may interchange the order of summation in a double summation.

Example 2.3.17:

Given mn numbers a_{ij} for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$, prove that $\sum_{i=1}^{n} \sum_{j=1}^{i} a_{ij} = \sum_{j=1}^{n} \sum_{i=j}^{n} a_{ij}$.

Solution: View the mn numbers a_{ij} arranged in the form of an $m \times n$ matrix A.. In this matrix, the subscripts i, j of the numbers a_{ij} verify the following property: $i = j$ if a_{ij} is in the main diagonal of M, $i < j$ if a_{ij} is above the main diagonal and $i > j$ if a_{ij} is below the main diagonal.

Summing all the numbers of the matrix M which are below or on the main diagonal in two different ways, first line by line and then column by column, we get the desired equality.

Example 2.3.18 (Distributive law):
Consider the numbers a_1, a_2 and b_1, b_2, b_3. Then

$$\begin{aligned}\left(\sum_{i=1}^{2} a_i\right)\left(\sum_{j=1}^{3} b_j\right) &= (a_1 + a_2)(b_1 + b_2 + b_3) \\ &= (a_1b_1 + a_1b_2 + a_1b_3) + (a_2b_1 + a_2b_2 + a_2b_3) \\ &= \sum_{i=1}^{2}\left(\sum_{j=1}^{3} a_ib_j\right)\end{aligned}$$

More generally, we have the distributive law

$$\left(\sum_{i=1}^{m} a_i\right)\left(\sum_{j=1}^{n} b_j\right) = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} a_ib_j\right).$$

Change of variable:

Consider the sum $\sum_{i=1}^{n} a_i$. Let us replace the index variable i by $j + 1$. $i = 1$ implies that $j + 1 = 1$ and hence $j = 0$. $i = n$ implies that $j + 1 = n$ and therefore $j = n - 1$. Therefore we have the equality

$$\sum_{i=1}^{n} a_i = \sum_{j=0}^{n-1} a_{j+1}.$$

More generally, we may replace the index variable i by $p(i)$ where $p(i)$ is a permutation of the range of i. (See [1].)

Products:

The symbolism

$$\prod_{P(i)} a_i$$

means the product of all numbers a_i for which the integer i satisfies the property $P(i)$. If no such integer i exists, then we define the product to have the value 1. It is in this spirit that we have defined $0! = 1$.

Sum of the products:

The sum of the product of elements belonging to all k-subsets of the n-set $A = \{1, 2, \ldots, n\}$ is denoted by

$$\sum_{1 \leq i_1 < i_2 < \cdots i_k \leq n} i_1 \times i_2 \times \cdots \times i_k.$$

The above sum is taken over all integers $i_1, i_2, \ldots, i_k$ satisfying the property $1 \leq i_1 < i_2 < \cdots i_k \leq n$. Since the order of elements in a set is irrelevant, we have imposed the condition $1 \leq i_1 < i_2 < \cdots i_k \leq n$ on the elements i_j of the subset $\{i_1, i_2, \ldots, i_k\}$ in order to avoid duplication (because $\{1, 2\} = \{2, 1\}$).

To understand this notation, consider the following special case.

Example 2.3.19 (Sum of the products):
Find the sum of the products of integers belonging to all 3-element subsets of the set $\{1, 2, 3, 4, 5\}$.

Solution: The required sum is $\sum_{1 \leq i < j < k \leq 5} i \times j \times k$, which is equal to

$(1 \times 2 \times 3) + (1 \times 2 \times 4) + (1 \times 2 \times 5) + (1 \times 3 \times 4) + (1 \times 3 \times 5) + (1 \times 4 \times 5) + (2 \times 3 \times 4) + (2 \times 3 \times 5) + (2 \times 4 \times 5) + (3 \times 4 \times 5) = 6 + 8 + 10 + 12 + 15 + 20 + 24 + 30 + 40 + 60 = 225.$

2.4 Binomial Theorem

In school we learn the following elementary formulas:

$$(a+b)^2 = a^2 + 2ab + b^2; \qquad (a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$

The binomial theorem gives us a formula for $(a+b)^n$ for any positive integer n.

We observe that the expression on the right-hand side of $(a+b)^2$ is a *homogeneous expression* of degree 2 in a and b (that is, the total degree of each term in the expansion of $(a+b)^2$ is 2). Similarly, the expression on the right-hand side in the expansion of $(a+b)^3$ is a homogeneous expression of degree 3 in a and b. More generally, one can prove easily by induction on n the following observation.

Observation 2.4.1:
For any positive integer n, the expansion of $(a+b)^n$ is a homogeneous expression of degree n. Moreover, all terms of the form $a^k b^{n-k}$ for k with $0 \leq k \leq n$ appear in the expansion of $(a+b)^n$.

Proof. Proof is by induction on n.

Induction basis: Let us verify the observation for $n = 1$. $(a + b)^1 = a + b$ and clearly $a + b$ is a homogeneous expression and all possible terms of degree one appear.

Induction hypothesis: Suppose the observation for a fixed n with $(n \geq 1)$.

We shall prove the observation for $n + 1$.

$(a+b)^{n+1} = (a+b)^n(a+b)$. By induction hypothesis, $(a+b)^n$ is a homogeneous expression in a and b and all possible terms of the form $a^k b^{n-k}$ appear in its expansion. Denote this expression by E_n. Hence $(a+b)^{n+1} = E_n \times (a+b) = aE_n + bE_n$. Clearly $aE_n + bE_n$ is a homogeneous expression of degree $n+1$ and all possible terms of the form $a^k b^{n+1-k}$ for all k with $0 \leq k \leq n+1$ appear in it. This proves the observation. ■

The binomial theorem, Pingala's Chanda-Sutra (Sutra means "formula" in Sanskrit language and Chanda means "moon") gives us a formula for $(a+b)^n$ for any positive integer n.

Theorem 2.4.1:
For any positive integer n we have

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$$

$$= \binom{n}{0}a^0b^n + \binom{n}{1}a^1b^{n-1} + \binom{n}{2}a^2b^{n-2} + \cdots + \binom{n}{n}a^nb^0$$

Proof. By Observation 2.4.1, all terms of the form a^kb^{n-k} for k with $0 \le k \le n$ appear in the expansion of $(a+b)^n$. Our job is now to find the coefficient of a^kb^{n-k} in the expansion of $(a+b)^n$. Consider the product

$$(a+b)^n = \overbrace{(a+b) \times (a+b) \times \cdots \times (a+b)}^{n \text{ factors}}$$

A term of the form a^kb^{n-k} is obtained by choosing exactly k a's out of n factors $(a+b)$ (and hence $n-k$ b's out of the remaining $n-k$ factors $(a+b)$). By definition of the binomial coefficient, k a's can be chosen out of n factors $(a+b)$ in $\binom{n}{k}$ ways. Hence the coefficient of the term a^kb^{n-k} in the expansion of $(a+b)^n$ is $\binom{n}{k}$. This proves the theorem. ■

If we define $0^0 = 1$ (since $\lim_{x\to 0+} x^x = 1$), the above binomial theorem remains true even if $n = 0$. Setting $a = b = 0$ and $n = 0$ in the binomial formula, we obtain $0^0 = 1$ which is true, because $\binom{0}{0} = 1$.

In fact, the binomial formula can be extended to any real exponent. We state the following theorem without proof.

Theorem 2.4.2:
If r is a real number which is not a non-negative integer, then the infinite series

$$\sum_{k=0}^{\infty} \binom{r}{k} x^k$$

converges to $(1+x)^r$ for all real numbers x with $|x| < 1$.

The binomial theorem justifies the name "binomial coefficients" thanks to the numbers $\binom{n}{k}$ appearing as coefficients:

Example 2.4.1 (Binomial theorem):
Let us expand $(a+b)^5$. Here $n = 5$.

By Theorem 2.4.1,

$$\begin{aligned}(a+b)^5 &= \binom{5}{0}a^0b^5 + \binom{5}{1}a^1b^4 + \binom{5}{2}a^2b^3 + \binom{5}{3}a^3b^2 \\ &\quad + \binom{5}{4}a^4b^1 + \binom{5}{5}a^5b^0 \\ &= b^5 + 5ab^4 + 10a^2b^3 + 10a^3b^2 + 5a^4b^1 + 5a^5\end{aligned}$$

Since $\binom{n}{k} = \binom{n}{n-k}$ (duality relation of the binomial coefficients), we observe that in the expansion of $(a+b)^n$, the coefficients equidistant from the beginning and the end are equal. Furthermore, because of the following relations of the binomial coefficients,

$$\binom{2n}{0} < \binom{2n}{1} < \cdots < \binom{2n}{n}$$

$$\binom{2n-1}{0} < \binom{2n-1}{1} < \cdots < \binom{2n-1}{n-1} = \binom{2n-1}{n}$$

the coefficients in the expansion $(a+b)^n$ first strictly increase and then strictly decrease (this is known as the monotonicity of the binomial coefficients).

Remark 2.4.1 (Justification for the name: Chanda-Sutra):
As we have already said, the word “Chanda” means “moon” in the ancient Sacred Sanskrit language. Just like the rising and decaying moon, the binomial coefficients first increase and then decrease. Because of the above duality property, the name Chanda-Sutra=Moon Formula perfectly justifies the appellation The Binomial Theorem.

Binomial identities:
The binomial theorem is very useful for deriving identities involving binomial coefficients:

Corollary 2.4.2.1:
If n is a positive integer, then we have

$$(1+x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k = 1 + \binom{n}{1} x^1 + \binom{n}{2} x^2 + \cdots + \binom{n}{n} x^n,$$

$$\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{n} = 2^n,$$

and

$$\binom{n}{0} - \binom{n}{1} + \cdots + (-1)^n \binom{n}{n} = 0$$

Proof. By Theorem 2.4.1,

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$$

Setting $a = 1$ and $b = x$ and using $\binom{n}{0} = 1$ we get the first identity.

Setting $a = b = 1$ we obtain the second equation.

Setting $a = 1$ and $b = -1$ we obtain the third one. ■

As we have already noted, a technique usually employed in proving combinatorial identities is to compare the same quantity calculated in two different ways. The following example illustrates this technique.

Example 2.4.2 (Binomial identities):
Prove that

$$\binom{n}{0}^2 + \binom{n}{1}^2 + \binom{n}{2}^2 + \cdots + \binom{n}{n}^2 = \binom{2n}{n}$$

Proof. The right-hand side of the identity provides us a hint. The right-hand side integer $\binom{2n}{n}$ is the coefficient of x^n in the expansion of $(1+x)^{2n}$, since by replacing n by $2n$ in the first identity of Corollary 2.4.2.1, we have

$$(1+x)^{2n} = \sum_{k=0}^{2n} \binom{2n}{k} x^k$$

Let us now calculate the coefficient of x^n in the expansion $(1+x)^{2n}$ in a different way. By observing the left-hand side of the given identity, we get a hint. Let us write $(1+x)^{2n} = (1+x)^n \times (1+x)^n$. We shall show that the left-hand side of the identity is *also* the coefficient of x^n in the product

$$\begin{aligned}(1+x)^n \times (1+x)^n &= \left(1+\binom{n}{1}x^1+\binom{n}{2}x^2+\cdots+\binom{n}{n}x^n\right)\\ &\quad \times\left(1+\binom{n}{1}x^1+\binom{n}{2}x^2+\cdots+\binom{n}{n}x^n\right)\end{aligned}$$

A term in x^n is obtained by taking a term $\binom{n}{k}x^k$ from the first factor and a term $\binom{n}{n-k}x^{n-k}$ from the second factor of the right-hand side of the above expression for all k with $0 \le k \le n$. Hence the coefficient of x^n in the product is

$$\begin{aligned}\sum_{k=0}^{n}\binom{n}{k}\binom{n}{n-k} &= \binom{n}{0}\binom{n}{n}+\binom{n}{1}\binom{n}{n-1}+\binom{n}{2}\binom{n}{n-2}\\ &\quad +\cdots+\binom{n}{n}\binom{n}{0}\end{aligned}$$

By the duality relation of the binomial coefficient (see Example 2.3.8), $\binom{n}{n-k} = \binom{n}{k}$ for all k with $0 \le k \le n$. Substituting $\binom{n}{k}$ for $\binom{n}{n-k}$ in the above expression, we obtain the following expression for the coefficient of x^n in $(1+x)^n \times (1+x)^n$:

$$\binom{n}{0}^2+\binom{n}{1}^2+\binom{n}{2}^2+\cdots+\binom{n}{n}^2$$

This establishes the identity. ∎

Example 2.4.3 (Chu-Vandermonde formula):
Prove the Chu-Vandermonde formula

$$\sum_{k,\text{ an integer}}\binom{r}{k}\binom{s}{n-k} = \binom{r+s}{n}$$

Solution: Note that even though the summation on the left side is taken over all integers, only finitely many terms are non-zero because the binomial coefficient $\binom{r}{k} = 0$ if $k < 0$ or $k > r$.

We may assume that r and s are positive integers. The right-hand side gives us a clue to prove the formula. The right-hand side is the coefficient of x^n in the expansion of $(1+x)^{r+s}$ which is equal to $\sum_{k=0}^{r+s}\binom{r+s}{k}x^k$ by Corollary 2.4.2.1. Hence we prove this identity by finding the coefficient of x^n in $(1+x)^{r+s}$ in a second way.

Write

$$(1+x)^{r+s} = (1+x)^r \times (1+x)^s$$

Let us find the coefficient of x^{r+s} in the product

$$(1+x)^r \times (1+x)^s = \left(\sum_{k=0}^{r}\binom{r}{k}x^k\right) \times \left(\sum_{k=0}^{s}\binom{s}{k}x^k\right)$$

The coefficient of x^n is the sum of the product of the coefficient of x^k in $\left(\sum_{k=0}^{r}\binom{r}{k}x^k\right)$ and the coefficient of x^{n-k} in $\left(\sum_{k=0}^{s}\binom{s}{k}x^k\right)$ for all possible integer k which is equal to

$$\sum_{k, aninteger}\binom{r}{k}\binom{s}{n-k} = \binom{r+s}{n}.$$

Therefore the formula is proved.

A combinatorial interpretation of the above identity:

A nice combinatorial interpretation (proof) of the above Chu-Vandermonde identity is given in [1].

By the definition of the binomial coefficients, the right-hand side is the number of ways of selecting n people from among r men and s women; whereas each term on the left-hand side is the number of ways to select k men from among r men and $n-k$ women from among the s women for all possible values of k. By the product rule (Proposition 2.2.2), we obtain that the left-hand side of the formula is *also* the number of ways of selecting n people from among r men and s women.

Differentiation and integration may be used to derive certain binomial identities. The following examples illustrate the method.

Example 2.4.4 (Binomial identity using differentiation):
Prove that

$$\sum_{k=1}^{n} k\binom{n}{k} = n2^{n-1}$$

Proof. By Corollary 2.4.2.1, we have

$$(1+x)^n = 1 + \binom{n}{1}x^1 + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n$$

Differentiating both sides with respect to x using the formula $d/dx\,(x^k) = kx^{k-1}$, for $k \geq 0$, we obtain

$$n(1+x)^{n-1} = \binom{n}{1} + 2\binom{n}{2}x^1 + 3\binom{n}{3}x^2 \cdots + n\binom{n}{n}x^{n-1}$$

Setting $x = 1$ on both sides, we obtain the result. ■

Example 2.4.5 (Binomial identity using integration):
Prove that

$$1 + \binom{n}{1}/2 + \binom{n}{2}/3 + \cdots + \binom{n}{n}/(n+1) = \left(2^{n+1} - 1\right)/(n+1).$$

Proof. By Corollary 2.4.2.1, we have

$$(1+x)^n = 1 + \binom{n}{1}x^1 + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n$$

Integrating both sides between 0 and 1 with respect to x using the formula $\int x^k\,dx = x^{k+1}/k+1$ $(k \geq 0)$ we get,

$$\int_0^1 (1+x)^n = \int_0^1 \left(1 + \binom{n}{1}x^1 + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n\right)$$

$$\left[\frac{(1+x)^{n+1}}{n+1}\right]_0^1 = \left[x + \binom{n}{1}\frac{x^2}{2} + \binom{n}{2}\frac{x^3}{3} + \cdots + \binom{n}{n}\frac{x^{n+1}}{n+1}\right]_0^1$$

Setting $x = 1$ on both sides, we get the desired identity. ■

2.5 Multinomial Coefficients

Ordered and unordered partitions:

One of the important generalizations of the binomial coefficient $\binom{n}{k}$ is the *multinomial coefficient* $\binom{n}{n_1,n_2,\dots,n_k}$. To define the multinomial coefficient, we need the idea of an ordered partition of a given set.

Consider an n-set A and subsets $A_1, A_2, \dots, A_k$ of A. The subsets $A_1, A_2, \dots, A_k$ form a *partition* of the set A if the following two conditions are satisfied.

1. $A_i \cap A_j = \emptyset$ for all i, j with $1 \leq i < j \leq k$. In other words, the subsets $A_1, A_2, \dots, A_k$ are mutually disjoint.

2. $A_1 \cup A_2 \cup \cdots \cup A_k = A$. In other words, the union of the subsets $A_1, A_2, \dots, A_k$ is the whole set A.

Note that A_i may be the empty set . We distinguish two types of partition depending on the definition of the equality of two partitions.

A partition $(A_1, A_2, \dots, A_k)$ of the set A is an *ordered partition* if the equality of partition is defined as follows: Two partitions $(A_1, A_2, \dots, A_k)$ and $\left(A'_1, A'_2, \dots, A'_p\right)$ are declared equal if $k = p$ and $A_i = A'_i$ for all $i = 1, 2, \dots, k$.

For non-negative integers $n_1, n_2, \dots, n_k$, an $(n_1, n_2, \dots, n_k)$-ordered partition of an n-set A is an ordered partition $(A_1, A_2, \dots, A_k)$ of A with $|A_i| = n_i$ for $i = 1, 2, \dots, k$. Note that $n_1 + n_2 + \cdots + n_k = n$. A partition $\{A_1, A_2, \dots, A_k\}$ of the set A is an *unordered partition* if the equality of partition is defined as follows: Two partitions $\{\, A_1, A_2, \dots, A_k \,\}$ and $\left\{\, A'_1, A'_2, \dots, A'_p \,\right\}$ are declared equal if $k = p$ and each A_i is equal to some A'_j.

Notation: We use *parentheses ( ) for ordered partitions* and *braces* $\{\}$ *for unordered partitions.*

Example 2.5.1 (Ordered and unordered partitions):
Consider the set $A = \{\, 1, 2, 3, 4, 5 \,\}$. (A_1, A_2, A_3) where $A_1 = \{\, 3, 5 \,\}, A_2 = \{\, 2, 4 \,\}, A_3 = \{\, 1 \,\}$ is an ordered $(2, 2, 1)$-partition of A. Another ordered $(1, 2, 2)$-partition is (A'_1, A'_2, A'_3) where $A'_1 = \{\, 1 \,\}, A'_2 = \{\, 3, 5, \,\}, A'_3 = \{\, 2, 4 \,\}$. Note that $(A_1, A_2, A_3) \neq (A'_1, A'_2, A'_3)$. In other words, as ordered partitions (A_1, A_2, A_3) and

(A_1', A_2', A_3') are different but as unordered partitions $\{ A_1, A_2, A_3 \}$ and $\{ A_1', A_2', A_3' \}$ are one and the same.

Multinomial coefficients are defined in terms of the ordered $(n_1, n_2, \ldots, n_k)$ partitions of an n-set.

Definition 2.5.1 (Multinomial coefficients):
Consider an n-set and k non-negative integers $n_1, n_2, \ldots, n_k$ with $n_1 + n_2 + \cdots + n_k = n$. Then the multinomial coefficient $\binom{n}{n_1, n_2, \ldots, n_k}$ or k-nomial coefficient (read n choose $n_1, n_2, \ldots, n_k$) is the number of ordered $(n_1, n_2, \ldots, n_k)$ partitions of the n-set.

Example 2.5.2 (Binomial coefficient as the number of ordered (n_1, n_2)-partitions):
Consider an n-set A and an integer k with $1 \leq k \leq n$. Then the binomial coefficient $\binom{n}{k}$ is the number of k-subsets of the n-set. Every choice of a subset k-subset B uniquely defines another $(n-k)$-subset $B^c = A \setminus B$, the complement of the set B in the set A. Hence every k-subset B of A defines an ordered $(k, n-k)$-partition and conversely every ordered $(k, n-k)$-partition of the n-set A defines a k-subset of A. Hence the number of k-subsets of A is the same as the number of ordered $(k, n-k)$-partitions of A. Hence by the Proposition 2.2.1, we have the equality

$$\binom{n}{k, n-k} = \binom{n}{k}$$

Example 2.5.3:
Find the number of $\overbrace{(1, 1, \ldots, 1)}^{n\ 1's}$ ordered partitions of an n-set.

Each ordered $\overbrace{(1, 1, \ldots, 1)}^{n\ 1's}$ partition of an n-set defines a permutation on the n-set and conversely each permutation on the n-set defines a $\overbrace{(1, 1, \ldots, 1)}^{n\ 1's}$ partition of the n-set. Hence the required number is the number of permutations on the n-set which is $n!$ (by Proposition 2.2.1 and Corollary 2.3.1.1).

The following theorem gives a formula for the multinomial coefficients.

THEOREM 2.5.1:
The number of ordered $(n_1, n_2, \dots, n_k)$-partitions of an n-set ($n_i \geq 0$ for all $i = 1, 2, \dots, k$ and $\sum_{i=1}^{k} n_i = n$) is

$$\binom{n}{n_1, n_2, \dots, n_k} = \frac{n!}{n_1! n_2! \cdots n_k!}$$

Proof. Let $(A_1, A_2, \dots, A_k)$ be an ordered $(n_1, n_2, \dots, n_k)$ partition of an n-set. From the n-set, a subset A_1 consisting of n_1 elements can be chosen in $\binom{n}{n_1}$ ways (by Theorem 2.3.2). Having chosen a subset A_1, another subset A_2 consisting of n_2 elements can be chosen from the remaining $n - n_1$ elements in $\binom{n-n_1}{n_2}$ ways. More generally, having chosen the subsets A_1 with n_1 elements, A_2 with n_2 elements, $\dots, A_{p-1}$ with n_{p-1} elements, we can choose the p-th subset A_p from the remaining $n - n_1 - n_2 - \cdots - n_{p-1}$ in $\binom{n-n_1-n_2-\cdots-n_{p-1}}{n_p}$ ways $(1 \leq p \leq k)$.. Hence by the product rule (Proposition 2.2.3 the number of ordered $(n_1, n_2, \dots, n_k)$ partitions of an n-set is

$$\binom{n}{n_1}\binom{n-n_1}{n_2}\cdots\binom{n-n_1-n_2-\cdots-n_{p-1}}{n_p}\cdots$$

$$\binom{n-n_1-n_2-\cdots-n_{k-1}}{n_k}$$

which is equal to (by Theorem 2.3.2)

$$\frac{n!}{n_1!(n-n_1)!}\frac{(n-n_1)!}{n_2!(n-n_1-n_2)!}\frac{(n-n_1-n_2)!}{n_3!(n-n_1-n_2-n_3)!}\cdots$$

$$\frac{(n-n_1-\cdots-n_{k-1})!}{n_k!}$$

On simplifying, this number is equal to

$$\frac{n!}{n_1! n_2! \cdots n_k!}$$

This proves the result. ■

Example 2.5.4:
In a game of bridge, 52 cards are distributed equally among four players. Hence each player receives 13 cards. The order of the cards received by each player is irrelevant. Find the total number of situations possible.

Each hand can be viewed as an ordered $(13, 13, 13, 13)$ partition of a 52-set. Therefore the number of possible different hands is the multinomial coefficient $\binom{52}{13,13,13,13} = \frac{52!}{13!13!13!13!} = \frac{52!}{(13!)^4}$.

Another interpretation of multinomial coefficients in terms of permutations of multisets:
Consider a multiset $M = \{ 1^{n_1}, 2^{n_2}, \dots, k^{n_k} \}$ where the element 1 is repeated n_1 times, the element 2 is repeated n_2 times, etc., with $n = n_1 + n_2 + \cdots + n_k$. This multiset is also written as $M = \{ n_1.1, n_2.2, \dots, n_k.k \}$. Then the number of permutations of the elements of the set M is exactly

$$\binom{n}{n_1, n_2, \dots, n_k} = \frac{n!}{n_1!n_2! \cdots n_k!}$$

Example 2.5.5:
Find the number of *anagrams* that can be formed from the word "SRIRANGAM."

The number of anagrams is the *same as* the number of permutations of the multiset $\{ 1.S, 2.R, 2.A, 1.I, 1.N, 1.G, 1.M \}$ because the letters R,A, are repeated twice in the word and all other letters appear only once. Hence the required number is the multinomial coefficient

$$\binom{9}{1, 2, 2, 1, 1, 1, 1} = \frac{9!}{1!2!2!1!1!1!1!} = 90720.$$

Example 2.5.6 (Mutinomial coefficients):
Find the number of different words that can be formed from the word "MISSISSIPPI." Find the number of these words in which four I's do not occur together.

Solution: The words of "MISSISSIPPI" induce the multiset $\mathcal{M} = \{ 1.M, 4.S, 4.I, 2.P \}$. The number of elements of the multiset is 11. Hence the required number is the number of permutations of the multiset $\mathcal{M}$ which is equal to the multinomial (4-nomial) coefficient

$$\binom{11}{1,4,4,2} = \frac{11!}{1!4!4!2!} = 34650.$$

For the second question, consider the four I's as a single entity I', that is, $I' \equiv IIII$. Then we have the multiset $\mathcal{M}' = \{\,1.M, 4.S, 1.I', 2.P\,\}$ of 8 elements. The number of permutations of $\mathcal{M}'$ is the 4-nomial coefficient $\binom{8}{1,4,1,2}$ which is equal to $\frac{8!}{1!4!1!2!} = 840$. Therefore the number of permutations of the multiset $\mathcal{M}$ in which the four I's appear together is 840.

Hence, by *the subtraction rule*, the required number is $34650 - 840 = 33810$.

Example 2.5.7 (Unordered partition):
Find the number of *unordered* $\underbrace{\{\,2,2,\ldots,2\,\}}_{n\ 2's}$ partitions of the $2n$ set $A = \{\,1,2,\ldots,2n\,\}$. (For example, there are three $\{\,2,2,2\,\}$ unordered partitions of the 4-set $\{\,1,2,3,4\,\}$, namely,

$$\{\,\{\,1,2\,\},\{\,3,4\,\}\,\} \quad ,\{\,\{\,1,3\,\},\{\,2,4\,\}\,\} \quad ,\{\,\{\,1,4\,\},\{\,2,3\,\}\,\}).$$

The number of $\underbrace{(\,2,2,\ldots,2\,)}_{n\ 2's}$ *ordered* partitions of A of the $2n$-set is the multinomial coefficient $\binom{2n}{2,2,\ldots,2}$ (by definition).

Each *unordered* $\underbrace{\{\,2,2,\ldots,2\,\}}_{n\ 2's}$ partition $\{\,A_1, A_2, \ldots, A_n\,\}$ of the $2n$-set A gives rise to $n!$ $\underbrace{(\,2,2,\ldots,2\,)}_{n\ 2's}$ *ordered* partitions of A (because there are $n!$ possible permutations of $A_1, A_2, \ldots, A_n$.) Conversely, the $n!$ $\underbrace{(\,2,2,\ldots,2\,)}_{n\ 2's}$ *ordered* partitions $A_1, A_2, \ldots, A_n$ of A define only one *unordered* $\underbrace{\{\,2,2,\ldots,2\,\}}_{n\ 2's}$ partition $\{\,A_1, A_2, \ldots, A_n\,\}$ of A (since we disregard the order of sets in the partition.) Hence the number of *unordered* $\underbrace{\{\,2, 2, \dots, 2\,\}}_{n\ 2's}$ partitions is

$$\binom{2n}{2, 2, \dots, 2}/n! = \frac{(2n)!}{2^n n!}$$

Example 2.5.8 (Unordered partition):
Find the number of partitions of the n-set $A = \{\,1, 2, \dots, n\,\}$ into which j subsets can be partitioned such that there are k_i subsets having $i \geq 1$ elements.

Solution: The number of subsets into which A should be partitioned is $j = k_1 + k_2 + \cdots$ and the number of elements of the set A is $n = k_1 + 2k_2 + 3k_3 + \cdots$.

The number of *ordered*

$$\left(\overbrace{1, 1, \dots, 1}^{k_1\ 1's}, \overbrace{2, 2, \dots, 2}^{k_2\ 2's}, \overbrace{3, 3, \dots, 3}^{k_3\ 3's}, \dots\right)$$

partitions is the multinomial coefficient

$$\binom{n}{1, 1, \dots, 1, 2, 2, \dots, 2, 3, 3, \dots, 3, 4, \dots} = \frac{n!}{1!^{k_1} 2!^{k_2} 3!^{k_3} \cdots}.$$

Each *unordered*

$$\left\{\overbrace{1, 1, \dots, 1}^{k_1\ 1's}, \overbrace{2, 2, \dots, 2}^{k_2\ 2's}, \overbrace{3, 3, \dots, 3}^{k_3\ 3's}, \dots\right\}$$

partition gives rise to $k_1!k_2!k_3! \cdots$ *ordered*

$$\left(\overbrace{1, 1, \dots, 1}^{k_1\ 1's}, \overbrace{2, 2, \dots, 2}^{k_2\ 2's}, \overbrace{3, 3, \dots, 3}^{k_3\ 3's}, \dots\right)$$

partitions (because there are $k_1!$ ways to permute subsets with one element, $k_2!$ ways to permute subsets with 2 elements etc., hence there are $k_1!k_2!k_3! \cdots$ ways to permute all the subsets by the product rule (Proposition 2.2.2) and conversely. Hence the required number is

$$\binom{n}{1,1,\ldots,1,2,2,\ldots,2,3,3,\ldots,3,4,\ldots}/k_1!k_2!k_3!\cdots =$$

$$\frac{n!}{1!^{k_1}k_1!2!^{k_2}k_2!3!^{k_3}k_3!\cdots}.$$

The following example illustrates a special case of the above example.

Example 2.5.9:
Find the number of unordered $\{1,1,2\}$ partitions of the 4-set $\{1,2,3,4\}$.

By the formula of Example 2.5.8, the required number is $\frac{4!}{(1!)^2 2 (2!)^1 1!} = 6$. The six $\{1,1,2\}$ unordered partitions $P_i = \{A_1, A_2, A_3\}$ for $i = 1, 2, \ldots, 6$ are given in the table:

Table 2.5: Six $\{1,1,2\}$ partitions of the 4-set

Partitions	A_1	A_2	A_3
P_1	$\{1\}$	$\{2\}$	$\{3,4\}$
P_2	$\{1\}$	$\{3\}$	$\{2,4\}$
P_3	$\{1\}$	$\{4\}$	$\{2,3\}$
P_4	$\{2\}$	$\{3\}$	$\{1,4\}$
P_5	$\{2\}$	$\{4\}$	$\{1,3\}$
P_6	$\{3\}$	$\{4\}$	$\{1,2\}$

Example 2.5.10:
How many words of length 10 (that is 10-letter words) can be formed from 3 a's, 3 b's, 2 c's and 2 d's?

Note that 3+3+2+2=10. We shall use the bijective method.

Consider the 10-set $A = \{1, 2, \ldots, 10\}$.

We shall establish a one-to-one correspondence between the set of all ordered $(3,3,2,2)$ partitions of the set A and the set B of all words of length 10 with exactly 3 a's, 3 b's, 2 c's and 2 d's.

Consider an ordered $(3,3,2,2)$ partition $(\,A_1, A_2, A_3, A_4\,)$ of the set A. We associate to the partition (A_1, A_2, A_3, A_4) a unique word w of length 10 belonging to the set B in the following way: Denote by $w(i)$ the ith letter of the word w.

$$w(i) = \begin{cases} a & \text{if } i \in A_1 \\ b & \text{ if } i \in A_2 \\ c & \text{if } i \in A_3 \\ d & \text{ otherwise} \end{cases}$$

For example, to the partition $(\,A_1, A_2, A_3, A_4\,)$ where $A_1 = \{\,1,5,9,\,\}$,$A_2 = \{\,4,6,8\,\}$,$A_3 = \{\,2,7\,\}$ and $A_4 = \{\,3,10\,\}$ we associate the word *acdbabcbad*.

Conversely, given a word w of length 10 belonging to the set B, we construct a unique ordered $(3,3,2,2)$ partition $(\,A_1, A_2, A_3, A_4\,)$ of the set A in the following way. Let i be an integer such that $1 \leq i \leq 10$.

$$i \in \begin{cases} A_1 & \text{if } w(i) = a \\ A_2 & \text{if } w(i) = b \\ A_3 & \text{if } w(i) = c \\ A_4 & \text{otherwise} \end{cases}$$

For example, to the word $w = aabccdbabd$ the assigned partition is $A_1 = \{\,1,2,8\,\}$,$A_2 = \{\,3,7,9\,\}$,$A_3 = \{\,4,5\,\}$,$A_4 = \{\,6,10\,\}$.

Therefore a one-to-one correspondence is found from the set of all ordered $(3,3,2,2)$ partitions of the 10-set A onto the set B. But the number of ordered $(3,3,2,2)$ partitions of the 10-set A is the multinomial coefficient $\binom{10}{3,3,2,2}$. Hence (by the Proposition 2.2.1, the number of words of the set B is also equal to $\binom{10}{3,3,2,2}$, which is equal to $\frac{10!}{3!3!2!2!} = 25200$.

The above example motivates the following generalization and another interpretation of multinomial coefficients.

Theorem 2.5.2:

Consider an alphabet $A = \{\,a_1, a_2, \ldots, a_k\,\}$ consisting of k distinct letters. Let $n_1, n_2, \dots n_k$ be k non-negative integers with $n_1 + n_2 + \cdots + n_k = n$. Then the number of words of length n that can be formed using the letter a_1 exactly n_1 times, using the letter a_2 exactly n_2 times, etc., and using the letter a_k exactly n_k times is the multinomial coefficient

$$\binom{n}{n_1, n_2, \dots, n_k} = \frac{n!}{n_1! n_2! \cdots n_k!}.$$

Proof. It is enough if we imitate the proof of Example 2.5.10 by replacing the 10-set by the k-set A to establish a one-to-one correspondence between the ordered $(n_1, n_2, \dots, n_k)$ partitions of an n-set and the set of all words of length n from the alphabet A containing exactly n_1 times the letter a_1, containing exactly n_2 times the letter a_2, and finally containing exactly n_k times the letter a_k. ■

EXAMPLE 2.5.11:
Find the number of words of length 4 (4-letter words) that can be formed from the letters a, b, c in which the letter a appears at most three times, b appears at most once and c appears at most twice.

First we enumerate all the possible ordered partitions of 4 into three parts in which the first part is ≤ 3 (corresponding to the maximum number of appearances of a), the second part is ≤ 1 (corresponding to the maximum number of appearances of b) and the third part is ≤ 2 (corresponding to the maximum number of appearances of c).

The possible partitions of the integer 4 with the above conditions are: $(3, 1, 0)$,$(3, 0, 1)$,$(2, 1, 1)$,$(2, 0, 2)$,$(1, 1, 2)$.

Therefore, the number of words of length 4 with the imposed conditions is

$$\binom{4}{3,1,0} + \binom{4}{3,0,1} + \binom{4}{2,1,1} + \binom{4}{2,0,2} + \binom{4}{1,1,2}$$

which is equal to

$$\frac{4!}{3!1!0!} + \frac{4!}{3!0!1!} + \frac{4!}{2!1!1!} + \frac{4!}{2!0!2!} + \frac{4!}{1!1!2!} = 4 + 4 + 12 + 6 + 12 = 38.$$

Multinomial theorem:

The main use of the multinomial coefficient is the following generalization of the binomial Theorem 2.4.1.

THEOREM 2.5.3:

For any k real numbers $a_1, a_2, \ldots, a_k$ and for any positive integer n the following relation is satisfied:

$$(a_1 + a_2 + \cdots + a_k)^n = \sum_{\substack{n_1, n_2, \ldots, n_k \geq 0 \\ n_1 + n_2 + \cdots + n_k = n}} \binom{n}{n_1, n_2, \ldots, n_k} a_1^{n_1} a_2^{n_2} \cdots a_k^{n_k}$$

Proof. First of all, observe that the expansion of $(a_1 + a_2 + \cdots + a_k)^n$, is a homogeneous expression in $a_, a_2, \ldots, a_k$ of degree $n = n_1 + n_2 + \cdots + n_k$. Further, all possible terms of the form $a_1^{n_1} a_2^{n_2} \cdots a_k^{n_k}$ for $n_1, n_2, \ldots, n_k \geq 0$ with $n_1 + n_2 + \cdots + n_k = n$ are found in the expansion (see Observation 2.4.1). To get the expansion $(a_1 + a_2 + \cdots + a_k)^n$ it is enough if we find the coefficient (that is the number of occurrences) of each term $a_1^{n_1} a_2^{n_2} \cdots a_k^{n_k}$.

Now consider the k-set of alphabet $A = \{ a_1, a_2, \ldots, a_k \}$. A term of the form $a_1^{n_1} a_2^{n_2} \cdots a_k^{n_k}$ in the expansion of $(a_1 + a_2 + \cdots + a_k)^n$ can be viewed as a word of length $n_1 + n_2 + \cdots + n_k$ with the letter a_1 appearing exactly n_1 times, the letter a_2 appearing exactly n_2 times, etc., and finally the letter a_k appearing exactly n_k times. By Theorem 2.5.2, the number of such words is the multinomial coefficient

$$\binom{n}{n_1, n_2, \ldots, n_k}$$

Hence the relation is proved. ∎

EXAMPLE 2.5.12 (Multinomial theorem):

Find the expansion of $(a_1 + a_2 + a_3)^3$.

The possible ordered partitions of the integer 3 into nonnegative integers are: $(3,0,0), (0,3,0), (0,0,3), (2,1,0)(1,2,0), (2,0,1), (1,0,2), (0,2,1)$, $(0,1,2), (1,1,1)$.

The partition (n_1, n_2, n_3) corresponds to the term $a_1^{n_1} a_2^{n_2} a_3^{n_3}$ in the expansion of $(a_1 + a_2 + a_3)^3$ and its coefficient is $\binom{3}{n_1, n_2, n_3}$. The first three partitions give the same coefficient 1, the second three give the same coefficient 3 and the last partition gives the number 6. Hence

$$(a_1+a_2+a_3)^3 = a_1^3+a_2^3+a_3^3+3a_1^2a_2+3a_1a_2^2+3a_1^2a_3+3a_1a_3^2 \\ +3a_2^2a_3+3a_2a_3^2+6a_1a_2a_3$$

Example 2.5.13:
Find the coefficients of the terms $a_1^3a_3^2$ and $a_1^2a_2a_3^2$ in the expansion of $(3a_1+2a_2-4a_3)^5$.

The power of the term $a_1^3a_3^2 = a_1^3a_2^0a_3^2$ indicates the partition $(3+0+2)$ of the exponent 5, in $(3a_1+2a_2-4a_3)^5$. Hence the corresponding coefficient is $\binom{5}{3,0,2} \times (3)^3 2^0(-4)^2 = 1440$.

The partition induced by the term $a_1^2a_2a_3^2$ is $(2+1+2)$ and the corresponding coefficient is $\binom{5}{2,1,2} \times (3)^2 2^1(-4)^2 = 8640$.

Example 2.5.14:
Find the number of terms in the expansion of $(a_1+a_2+\cdots+a_k)^n$.

By the multinomial theorem, the number of terms in the expansion is the number of ordered partitions $(n_1, n_2, \cdots, n_k)$ of the exponent $n = \sum_{i=1}^k n_i$. As we have already seen, this number is the same as the number of nonnegative integral solutions of the homogeneous linear equation $x_1+x_2+\cdots+x_k = n$ which is the same as the number of n-multisubset of a k-set which is $\binom{n+k-1}{n}$.

In particular, the number of terms in the expansion of $(a+b)^n$ is $\binom{n+2-1}{n} = \binom{n+1}{n} = \binom{n+1}{1} = n+1$.

By comparing the coefficient of $a_1^{n_1}a_2^{n_2}\cdots a_k^{n_k}$ in the expansion of $(a_1+a_2+\cdots+a_k)^n$ in two different ways, the following corollary is proved.

Corollary 2.5.3.1:
We have:

$$\binom{n}{n_1, n_2, \ldots, n_k} = \sum_{\substack{n_1,n_2,\ldots,n_{i-1},n_{i+1},\ldots,n_k \geq 0, n_i>0 \\ n_1+n_2+\cdots+n_k=n}} \binom{n-1}{n_1, n_2, \ldots, n_{i-1}, n_i-1, n_{i+1}, \ldots, n_k}$$

Proof.

$$(a_+a_2+\cdots+a_k)^n=(a_1+a_2+\cdots+a_k)\times(a_1+a_2+\cdots+a_k)^{n-1}$$

The coefficient of $a_1^{n_1}a_2^{n_2}\cdots a_k^{n_k}$ in the left-hand side expression is $\binom{n}{n_1,n_2,\ldots,n_k}$, by the multinomial Theorem 2.5.3.

Let us now find the coefficient of $a_1^{n_1}a_2^{n_2}\cdots a_k^{n_k}$ in the right-hand expression. It is the coefficient of a_1 from the first bracket multiplied by the coefficient of $a_1^{n_1-1}a_2^{n_2}\cdots a_n^{n_k}$ from the second bracket + the coefficient of a_2 from the first bracket multiplied by the coefficient of $a_1^{n_1}a_2^{n_2-1}\cdots a_n^{n_k}$ from the second bracket $+\cdots+$ the coefficient of a_i from the first bracket multiplied by the coefficient of $a_1^{n_1}a_2^{n_2-1}\cdots a_i^{n_i-1}\cdots a_n^{n_k}$ from the second bracket $+\cdots+$ the coefficient of a_k from the first bracket multiplied by the coefficient of $a_1^{n_1}a_2^{n_2}\cdots a_n^{n_k-1}$ from the second bracket. By the multinomial Theorem 2.5.3, the coefficient of $a_1^{n_1}a_2^{n_2}\cdots a_k^{n_k}$ in the right-hand side is

$$\binom{n-1}{n_1-1,n_2,\ldots,n_k}+\binom{n-1}{n_1,n_2-1,n_3,\ldots,n_k}+\cdots$$
$$+\binom{n-1}{n_1,n_2,\ldots,n_{k-1},n_k-1}$$

which is equal to

$$\sum_{\substack{n_1,n_2,\ldots,n_{i-1},n_{i+1},\ldots,n_k\geq 0,n_i>0\\ n_1+n_2+\cdots+n_k=n}}\binom{n-1}{n_1,n_2,\ldots,n_{i-1},n_i-1,n_{i+1},\ldots,n_k}$$

■

The following corollary gives a formula for the sum of all possible k-nomial coefficients $\binom{n}{n_1,n_2,\ldots,n_k}$ for a given n and k.

Corollary 2.5.3.2:
We have:

$$\sum_{\substack{n_1,n_2,\ldots,n_k\geq 0\\ n_1+n_2+\cdots+n_k=n}}\binom{n}{n_1,n_2,\ldots,n_k}=k^n$$

Proof. Setting $a_1 = a_2 = \cdots = a_k = 1$ in the multinomial Theorem 2.5.3, the desired result is obtained. ■

By setting oddly subscripted a_i's to $+1$ and evenly subscripted a_i's to -1 in the multinomial formula 2.5.3, we obtain the following corollary.

Corollary 2.5.3.3:

$$\sum_{\substack{n_1,n_2,\ldots,n_k \geq 0\\ n_1+n_2+\cdots+n_k=n}} \binom{n}{n_1, n_2, \ldots, n_k} (-1)^{n_2+n_4+\cdots} = \frac{1-(-1)^k}{2}$$

Proof. Set $a_1 = a_3 = \cdots = 1$ and $a_2 = a_4 = \cdots = -1$ in the multinomial Theorem 2.5.3, we have

$$\sum_{\substack{n_1,n_2,\ldots,n_k \geq 0\\ n_1+n_2+\cdots+n_k=n}} \binom{n}{n_1, n_2, \ldots, n_k} (-1)^{n_2+n_4+\cdots} = \begin{cases} 0 & \text{if } k \text{ is even} \\ 1 & \text{if } k \text{ is odd} \end{cases}$$

Combining the two right-hand side piecewise equalities into one, we get the result. ■

2.6 Stirling Numbers

> Mathematical notations evolve like all languages do.
>
> *Donald Knuth*

There are two kinds of *Stirling numbers*: Stirling number of the first kind denoted by $\left[{n \atop k}\right]$ and the Stirling number of the second kind denoted by $\left\{{n \atop k}\right\}$. Berge denotes these numbers by s_n^k and S_n^k (see [3]). We shall first study the Stirling number of the first kind.

Stirling number of the first kind: Let us recall the falling factorial $x^{\underline{n}}$ (read, "x to the n falling") where

$$x^{\underline{n}} = x(x-1)\cdots(x-n+1) = \prod_{i=0}^{n-1}(x-i)$$

Note that the falling factorial $x^{\underline{n}}$ contains exactly n factors and is a polynomial of degree n in the variable x.

Definition 2.6.1 (Stirling numbers of the first kind):
The *absolute value of the coefficient* of x^k in the polynomial $x^{\underline{n}} = x(x-1)\cdots(x-n+1)$ is called the Stirling number of the first kind and is denoted by $\left[{n \atop k}\right]$ (read "n cycle k." This will be justified shortly; see [6]).

Hence we can write

$$\begin{aligned} x^{\underline{n}} &= x(x-1)\cdots(x-n+1) \\ &= \begin{bmatrix} n \\ n \end{bmatrix} x^n - \begin{bmatrix} n \\ n-1 \end{bmatrix} x^{n-1} + \cdots + (-1)^n \begin{bmatrix} n \\ 0 \end{bmatrix} x^0 \\ &= \sum_{k=0}^{n} (-1)^{n-k} \begin{bmatrix} n \\ k \end{bmatrix} x^k \end{aligned}$$

Remark 2.6.1 (Stirling numbers):
We could also define the Stirling number of the first kind $\left[{n \atop k}\right]$ as the coefficient of x^k in the *rising factorial* $x^{\overline{n}}$ (read "x to the n rising") by throwing out minus signs.

$$x^{\overline{n}} = x(x+1)(x+2)\cdots(x+n-1) = \sum_{k=0}^{n} \begin{bmatrix} n \\ k \end{bmatrix} x^k$$

The principal motivation for defining the Stirling number of the first kind is to convert falling factorial powers into ordinary powers.

The following examples illustrate the notion of Stirling numbers.

Example 2.6.1:
Find the Stirling numbers $\left[{n \atop k}\right]$ for $n = 4$ and $k = 0, 1, \ldots 4$. Solution:

$$x^{\underline{4}} = x(x-1)(x-2)(x-3) = x^4 - 6x^3 + 11x^2 - 6x$$

Hence $\left[{4 \atop 4}\right] = 1, \quad \left[{4 \atop 3}\right] = 6, \quad \left[{4 \atop 2}\right] = 11, \quad \left[{4 \atop 1}\right] = 6, \quad \left[{4 \atop 0}\right] = 0.$

Example 2.6.2:
Find $\left[{n+1 \atop 1}\right]$ for non-negative integer n.

Solution: By definition, $\left[{n+1 \atop 1} \right]$ is the coefficient of x^1 in the product $x(x+1)(x+2)\cdots(x+n)$, that is, the constant term in the product $(x+1)(x+2)\cdots(x+n)$ which is $1 \times 2 \times 3 \cdots \times n = n!$

Note that we always have the following relation by definition of Stirling numbers:

$$\left[{n \atop 0} \right] = 0 \text{ and } \left[{n \atop n} \right] = 1 \text{for non-negative integer } n.$$

Note also that $\left[{n \atop k} \right] = 0$ if $k > n$. We now prove the following simple addition formula (recurrence relation) for the Stirling numbers of the first kind.

Lemma 2.6.1 (Recurrence relation involving the Stirling number of the first kind):
For non-negative integers n and k we have the following relation:

$$\left[{n+1 \atop k} \right] = n \left[{n \atop k} \right] + \left[{n \atop k-1} \right]$$

Proof. By definition, the number $\left[{n+1 \atop k} \right]$ is the absolute value of the coefficient of x^k in the product $x(x-1)\cdots(x-n+1)(x-n)$. Now let us write this product as

$$\begin{aligned} x^{\underline{n+1}} &= (x-n) \times x(x-1)(x-2)\cdots(x-n+1) \\ &= x\left(x(x-1)\cdots(x-n+1)\right) - n\left(x(x-1)\cdots(x-n+1)\right) \end{aligned}$$

But then the absolute value of the coefficient of x^k on the right-hand side of the above writing is the absolute value of the coefficient of x^{k-1} in the product $x(x-1)\cdots(x-n+1) + n \times$ the absolute value of the coefficient of x^k in $x(x-1)\cdots(x-n+1)$ which by definition is $\left[{n \atop k-1} \right] + n \left[{n \atop k} \right]$. Hence the lemma. ∎

From the initial values $\left[{n \atop 0} \right] = 0$ and $\left[{n \atop n} \right] = 1$ for non-negative integer n and the recurrence relation $\left[{n+1 \atop k} \right] = n \left[{n \atop k} \right] + \left[{n \atop k-1} \right]$, we may find any Stirling number of the first kind. The following Stirling triangle of the first kind (similar to the Pascal triangle) constructs the Stirling numbers iteratively.

For example, the 4th line of the Stirling triangle gives the coefficients $\genfrac{[}{]}{0pt}{}{4}{1}$, $\genfrac{[}{]}{0pt}{}{4}{2}$, $\genfrac{[}{]}{0pt}{}{4}{3}$, $\genfrac{[}{]}{0pt}{}{4}{4}$ and $\genfrac{[}{]}{0pt}{}{4}{5}$. (Note that the first line is the 0th line.) To obtain the kth number in the nth line (assuming that the terms in the $(n-1)$st line have already been computed), we multiply by n the number found to the north-east of the number to be found and add it to the number to the north-west of the number to be calculated. The table may be continued to any number of lines.

Table 2.6: Stirling Triangle of the first kind

$$\begin{array}{ccccccccc}
 & & & & 1 & & & & \\
 & & & 1 & & 0 & & & \\
 & & 1 & & 1 & & 0 & & \\
 & 2 & & 3 & & 1 & & 0 & \\
4 & & 11 & & 6 & & 1 & & 0 \\
 & & & & \vdots & & & &
\end{array}$$

We shall now see a nice *combinatorial interpretation of the Stirling number* of the first kind:

Combinatorial interpretation of Stirling numbers of the first kind:

Permutations revisited:

As we have already seen, a (linear) permutation on an n-set $A = \{1, 2, \ldots, n\}$ is a one-to-one correspondence from the set A onto itself. There are three types of notation used to represent a permutation on an n-set. Each notation has its own advantage. We normally choose one suitable to the problem at hand. In the sequel, the three notations are illustrated.

EXAMPLE 2.6.3 (Three representations of a permutation): Consider a 7-set $A = \{1, 2, \ldots, 7\}$ and consider a bijection $p : A \rightarrow A$ defined as follows: $p(1) = 2, p(2) = 3, p(3) = 1, p(4) = 5, p(5) = 4, p(6) = 6, p(7) = 7$.

Two-line notation (matrix form or two-line notation or two-dimensional notation): In the two-line notation, the permutation p is written as

$$p = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 1 & 5 & 4 & 6 & 7 \end{pmatrix}$$

Note that in the first line of the matrix notation, the domain elements are listed in increasing order and the second line of the matrix gives the corresponding values of the bijective function p. Note that the interchange of any two columns of the matrix p leaves the function p unchanged.

One-line notation or one-dimensional notation: In the one-line notation of the permutation p, we write only the *values* of the bijective function p, that is, only the *second line of the matrix* form of p assuming that the domain elements are always *listed in the natural order* $1, 2, \cdots, n$. Hence we write

$$p = 2315467$$

Cycle notation: This notation uses the following well-known decomposition property of permutations:

Every permutation can be expressed in a unique manner, except for order, as a product (composition) of disjoint cycles.

To understand this, let us represent the permutation p graphically. In this graphical representation of p, we draw a directed arc from the point i to the point j if and only if $p(i) = j$. Here the points are the numbers $1, 2 \ldots, 7$ (see the Figure 2.5).

According to the figure, the permutation p can be written as the product of disjoint cycles as:

$$p = (123)(45)(6)(7).$$

More generally, a cycle $(x_1 x_2 \cdots x_k)$ of a permutation maps x_1 onto x_2,x_2 onto x_3,. . ., x_{k-1} onto x_k, and finally x_k onto x_1.

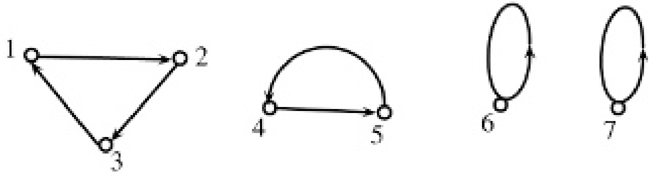


Figure 2.5: Permutation as a product of disjoint cycles

Reduced cycle notation: In the reduced form of cycle notation the single point cycles are omitted. Hence p can be represented as

$$p = (123)(45).$$

The disjoint cycle representation of a permutation is *not* unique. The same permutation p can be given as

$$p = (231)(54) = (312)(45) = (123)(45) = (45)(123)$$

because graphically, a cycle may start and end at any of its points! We shall see that the permutation p is indeed the product (composition) of disjoint cycles (123) and (45) because

$$p_1 = (123) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 1 & 4 & 5 & 6 & 7 \end{pmatrix}$$

Note that the elements i *not* present in the cycle (123) are fixed by the permutation p_1 that is, $p_(i) = i$. Similarly,

$$p_2 = (45) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 5 & 4 & 6 & 7 \end{pmatrix}$$

Now it is readily seen that the composition $p_1 \circ p_2 = p$ (for example, p_1 maps 1 onto 2 and p_2 maps 2 onto itself and hence $p_1 \circ p_2$ maps 1 onto 2, etc.).

We are now ready to interpret the number $\left[{n \atop k} \right]$ combinatorially.

Combinatorial meaning of $\left[{n \atop k}\right]$:
Consider the set of all $n!$ (linear) permutations of an n-set $A = \{1, 2, \ldots, n\}$ written in *cycle notation.* Then $\left[{n \atop k}\right]$ is the number of permutations on the n-set having k cycles. (Hence the reading n cycle k.)

The following example illustrates the combinatorial interpretation of $\left[{4 \atop 2}\right]$.

Example 2.6.4 (Combinatorial interpretation of the Stirling number of the first kind):
There are $4! = 24$ permutations that are possible on the 4-set $A = \{1, 2, 3, 4\}$. Among these 24 permutations, there are *eleven* different permutations which can be written in the cycle notation as the product of *two* disjoint cycles. The reader can verify that the permutations having exactly two cycles are:
(123)(4), (124)(3), (134)(2), (234)(1), (142)(3), (143)(2), (243)(1), (132)(4), (12)(34), (13)(24), (14)(23).
Hence $\left[{4 \atop 2}\right] = 11$.

Example 2.6.5 (Sum of all the Stirling numbers of the first kind):
Show that $\sum_{k=0}^{n} \left[{n \atop k}\right] = n!$ for $n \geq 0$.

Combinatorial proof: $\left[{n \atop k}\right]$ is the number of permutations of $[n]$ containing exactly k cycles. Hence the sum $\sum_{k=0}^{n} \left[{n \atop k}\right]$ accounts for the number of permutations of $[n]$ having $0, 1, \ldots, n$ cycles. This is nothing but the number of permutations of $[n]$ which is $n!$.

Algebraic proof: By definition,

$$x(x+1)\cdots(x+n-1) = \sum_{k=0}^{n} \left[{n \atop k}\right] x^k.$$

Setting $x = 1$, we get, $n! = \sum_{k=0}^{n} \left[{n \atop k}\right]$.

Consider the $n!$ permutations of the n-set $\{1, 2, \ldots, n\}$ expressed in cycle notation. The following example gives a formula for the total number of cycles in terms of the *harmonic number* h_n where

$$h_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \sum_{i=1}^{n} \frac{1}{i} \quad \text{for all } n \geq 1.$$

EXAMPLE 2.6.6 (The number of cycles):
For $n = 3$, the $3! = 6$ permutations in cycle notations are: $(1)(2)(3)$, $(1)(32)$, $(12)(3)$ (123), (132), $(13)(2)$. The total number of cycles is 11.

EXAMPLE 2.6.7 (Formula for the number of cycles):
Let us write all $n!$ permutations of the n-set $\{1, 2, \ldots, n\}$ in cycle notation. Then prove that the total number of cycles is $n!h_n$ where h_n is the *harmonic number* $h_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$.

Solution:

Let us first find how many times a special k-cycle $(a_1a_2 \ldots a_k)$ with $(1 \leq k \leq n)$, appears in these $n!$ permutations. By fixing the cycle $(a_1a_2 \ldots a_k)$, the remaining $n - k$ integers can be permuted in $(n - k)!$ ways.

Now let us compute the number of different possible k-cycles $(a_1a_2 \ldots a_k)$. The first element a_1 can be picked in n ways. Having selected the first element of the cycle, the second integer can be picked in $n - 2$ ways. More generally, the ith integer of the cycle may be picked in $(n - i)$ ways. Hence the number of different possible k-cycles $(a_1a_2 \ldots a_k)$ is $n(n-1)(n-2)\cdots(n-k)$. But then the k cycles $(a_1a_2 \ldots a_k)$, $(a_2a_3 \ldots a_ka_1)$, $(a_3a_4 \ldots a_ka_1a_2), \ldots,$ $(a_ka_1a_2 \ldots a_{k-1}$ are all one and the same cycle, since a cycle may start and end at any of its elements. Therefore, the total number of different k-cycles $(a_1a_2 \ldots a_k)$ among all the possible $n!$ permutations is $n(n-1)(n-2)\cdots(n-k)/k$.

By the product rule (Proposition 2.2.2), the total number of k-cycles is
$(n(n-1)(n-2)\cdots(n-k)/k) \times (n-k)! = n!/k$ for $(1 \leq k \leq n)$.

Hence the total number of cycles in all permutations is $\sum_{k=1}^{n} \frac{n!}{k} = n!\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right).$

EXAMPLE 2.6.8:
Consider the 3! permutations of the 3-set $\{1, 2, 3\}$ expressed in cycle notation. Then the total number of cycles among all the 6 permutations is $3!h_3 = 3!(1 + 1/2 + 1/3) = 6 + 3 + 2 = 11$ by Example 2.6.7.

Cyclic permutations or circular permutations:

DEFINITION 2.6.2:
A *cyclic permutation* on the n-set $A = \{1, 2, \ldots, n\}$ is a permutation on the set A whose cycle representation consists of only one cycle. Two cyclic permutations are declared equal if one can be obtained from the other by cyclic rotation in the clockwise/anti-clockwise direction. What is important is their positions relative to each other and *not* to their environment.

Let us recall that the permutation $(123 \ldots n)$ in cycle notation is represented as

$$\begin{pmatrix} 1 & 2 & 3 & \ldots & n \\ 2 & 3 & 4 & \ldots & 1 \end{pmatrix}$$

in two-line notation. Hence the following n cyclic permutations $(123 \ldots n)$; $(234 \ldots n1)$; $(345 \ldots 12) \ldots (n12 \ldots n-1)$ are all considered as equal.

The following example clarifies the definition.

EXAMPLE 2.6.9 (Cyclic permutations):
Find all the distinct cyclic permutations on the 4-set $\{1, 2, 3, 4\}$.

We have to write all distinct permutations of the 4-set having only one cycle. These are (1234); (1243); (1324); and their inverses (4321); (3421); (4231). They are $6 (= 3!)$ in number.

There is an n-to-1 correspondence between the set of all *linear permutations* on n symbols and the *circular permutation* on n symbols.

The following example illustrates the above statement.

EXAMPLE 2.6.10 (5-to-1 correspondence):
Consider the *linear* permutations on five integers $1, 2, 3, 4, 5$. There are in total $5! = 120$ linear permutations.

But the five linear permutations $12345, 23451, 34512, 45123, 51234$ all induce only one *circular* permutation 12345 because of the definition of the equality of two cyclic permutations. Hence the total number of cyclic permutations by the *division rule* is $5! = 24$.

Example 2.6.11 (Number of cyclic permutations on $(n+1)$-set): Find the number of cyclic permutations on an $(n+1)$-set ($n \geq 0$.)

Solution: By the combinatorial interpretation of the Stirling number of the first kind, $\left[{n+1 \atop 1}\right]$ is the number of permutations on an n-set having one cycle, that is, the number of cyclic permutations on an n-set. But by definition, $\left[{n+1 \atop 1}\right]$ is the coefficient of x^1 in the product $\overbrace{x(x+1)\cdots(x+n)}^{n\ factors}$, that is, the term independent of x in the product

$$(x+1)(x+2)\cdots(x+n)$$

which is $1 \times 2 \times 3 \cdots \times n = n!$. Therefore, the number of cyclic permutations on an $(n+1)$-set is $n!$.

Stirling number of the first kind as the sum of a product: Another equivalent way to define the Stirling number of the first kind is as the sum of the product of certain integers.

To define $\left[{n \atop k}\right]$, we consider the $n-1$-set $A = \{1, 2, \ldots, n-1\}$. Then $\left[{n \atop k}\right]$ is the sum of the product of integers in all $(n-k)$-subsets of the $(n-1)$-set A. Symbolically,

$$\begin{bmatrix} n \\ k \end{bmatrix} = \sum_{0<k_1<k_2<\cdots<k_{n-k}<n} k_1 k_2 \cdots k_{n-k}$$

Example 2.6.12 (Stirling number of first kind as the sum of a product):
What is the value of $\left[{5 \atop 2}\right]$?

Solution: By definition, $\left[{5 \atop 2}\right]$ is the sum of the product of integers in all $(5-2) = 3$-subsets of the 4-set $A = \{1, 2, 3, 4\}$. The four 3-subsets of A are $\{1, 2, 3\}$, $\{1, 2, 4\}$, $\{1, 3, 4\}$, and $\{2, 3, 4\}$. Hence

$$\begin{bmatrix} 5 \\ 2 \end{bmatrix} = 1 \times 2 \times 3 + 1 \times 2 \times 4 + 1 \times 3 \times 4 + 2 \times 3 \times 4 = 50.$$

Example 2.6.13:
Let p be a prime number. Then prove that $\left[{p \atop k} \right]$ for all k with $1 < k < p$.

Solution: In the modulo p world, the only relevant integers are $0, 1, \ldots, p-1$ and any other integer can be brought back to this list by taking the remainder on division by p. (Analogy: The clock follows modulo 12 arithmetic.) We have $x(x-1)\cdots(x-(p-1)) = 0$ for each $x = 0, 1, \ldots, p-1$. Hence we can write

$$x(x-1)\cdots(x-(p-1)) \equiv 0 \pmod{p}.$$

That is,

$$\sum_{k=1}^{p} \left[{p \atop k} \right] (-1)^{n-k} x^k \equiv 0 \pmod{p}.$$

By Fermat's theorem (see Chapter 3) we have,

$$x^p - x \equiv 0 \pmod{p}.$$

Subtracting the last two modular equalities, we get (since $\left[{p \atop p} \right] = 1$

$$\left[{p \atop p-1} \right] x^{p-1} - \left[{p \atop p-2} \right] x^{p-2} + \cdots - \left[{p \atop 2} \right] x^2 + \left[{p \atop 1} \right] x + x \equiv 0 \pmod{p}.$$

But $\left[{p \atop 1} \right] = (p-1)!$. Hence the coefficient of x in the above modular equality is $(p-1)! + 1$ which is $\equiv 0 \pmod{p}$ by Wilson's theorem (see Chapter 3). Hence we have

$$\left[{p \atop p-1} \right] x^{p-1} - \left[{p \atop p-2} \right] x^{p-2} + \cdots - \left[{p \atop 2} \right] x^2 \equiv 0 \pmod{p}.$$

This means that p divides each coefficient $\left[{p \atop k} \right]$ for $1 < k < p$.

2.7 Stirling Number of the Second Kind $\left\{ {n \atop k} \right\}$

The *Stirling number of the second kind* denoted by $\left\{ {n \atop k} \right\}$ is the number of *unordered partitions* of the n-set $A = \{1, 2, \ldots, n\}$ into

k mutually disjoint non-empty subsets. Note that the empty set is *not* permitted in the partition. (In the partitions corresponding to a multinomial coefficient, empty sets are allowed.) Such a partition is called a k-partition of the set A.

The curly braces in $\left\{ n \atop k \right\}$ (read n subset k) can be easily remembered, since curly braces denote sets. The following example illustrates the Stirling number of the second kind.

EXAMPLE 2.7.1 (Stirling number of the second kind):
What is the value of $\left\{ 4 \atop 2 \right\}$?

Solution: Consider the 4-set $A = \{1, 2, 3, 4\}$. The possible partitions of the set A into *two* non-empty subsets are: $\{1, 2, 3\}$ $\{4\}$; $\{1, 2, 4\}$ $\{3\}$; $\{1, 3, 4\}$ $\{2\}$; $\{2, 3, 4\}$ $\{1\}$;
$\{1, 2\}$ $\{3, 4\}$; $\{1, 3\}$ $\{2, 4\}$; $\{1, 4\}$ $\{2, 3\}$. There are 7 partitions of A into two non-empty subsets. Therefore by definition, $\left\{ 4 \atop 2 \right\} = 7$.

$$\left\{ n \atop 1 \right\} = 1 \text{ for } n \geq 1$$

because the number of unordered partitions of an n-set A into *one* mutually disjoint non-empty subset is 1(the set $\{A\}$ is the only partition of A). Again

$$\left\{ n \atop n \right\} = 1 \text{ for } n \geq 1$$

since the only unordered partition of the n-set $A = \{1, 2, \ldots, n\}$ into n mutually disjoint subsets is the partition $\{A_1, A_2, \ldots, A_n\}$ where $A_i = \{i\}$ for all $i = 1, 2, \ldots, n$.

We define $\left\{ 0 \atop 0 \right\} = 1$. Note that $\left\{ n \atop k \right\} = 0$ if $k > n$.

The initial and boundary values of the binomial coefficients, Stirling numbers of the first and second kind may be written succinctly with the help of the *Kronecker delta* δ_{ij} where

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Kronecker's famous citation:

> God created the natural numbers, all the rest is the work of man.
>
> *Kronecker*

Another interesting quotation:

> Man created the integers, all else is Dieudonné.
>
> *R. K. Guy*

Using the notation δ_{ij}, we write for non-negative integer n,

$$\binom{0}{n} = \left[{0 \atop n} \right] = \left\{ {0 \atop n} \right\} = \delta_{n0}.$$

Furthermore,

$$\binom{n}{n} = \left[{n \atop n} \right] = \left\{ {n \atop n} \right\} = 1 = \delta_{nn}.$$

We shall now prove the following addition formula (recurrence formula) involving Stirling numbers of the second kind.

Lemma 2.7.1 (Recurrence formula for Stirling number of the second kind):
For non-negative integers n and k we have:

$$\left\{ {n+1 \atop k} \right\} = k \left\{ {n \atop k} \right\} + \left\{ {n \atop k-1} \right\}$$

Proof. Consider the $(n+1)$-set $A = \{\, 1, 2, \ldots, n+1 \,\}$. By definition, $\left\{ {n+1 \atop k} \right\}$ is the number of unordered partitions of the set A into k mutually disjoint non-empty subsets of A. We distinguish two types of k-partitions of the $(n+1)$-set A.

Case 1: k-Partitions of A having the singleton set $\{\, n+1 \,\}$ as a member of the partition. (This set can be any other singleton set. If $1, 2, \ldots, n, n+1$ are professors in the Mathematics Department, then $n+1$ can be considered as the "Head" of the Mathematics Department.)

We are interested in counting the number of such partitions in terms of n. By definition, $\left\{ {n \atop k-1} \right\}$ is the number of unordered

$(k-1)$-partitions of the n-set $A\setminus\{\,n+1\,\}$. Each $(k-1)$-partition $\{\,A_1, A_2, \ldots, A_{k-1}\,\}$ of the n-set $A\setminus\{\,n+1\,\}$ defines the k-partition $\{\,A_1, A_2, \ldots, A_{k-1}\,\}\cup\{\,\{\,n+1\,\}\,\}$ of the $(n+1)$-set A. Conversely, each k-partition of the $(n+1)$-set A having the singleton set $\{\,n+1\,\}$ as a member, defines a $(k-1)$-partition (obtained by removing the set $\{\,n+1\,\}$ from the k-partition) of the n-set $A\setminus\{\,n+1\,\}$. Hence by Proposition 2.2.1, the number of k-partitions of A having the singleton set $\{\,n+1\,\}$ as a member of the partition is the same as the number of $(k-1)$-partitions of the n-set $A\setminus\{\,n+1\,\}$ which is by definition $\left\{ {n \atop k-1} \right\}$

Case 2: k-partitions of A having no singleton subset $\{\,n+1\,\}$ as a member, that is, partitions in which subsets containing the element $n+1$ must have at least two elements.

Let us count the number of such partitions in terms of n.

Consider a k-partition $\{\,A_1, A_2, \ldots, A_k\,\}$ of A such that the subset A_i containing the element $n+1$ and the number of elements of A_i is ≥ 2. Now the $\{\,A_1, A_2, \ldots, A_i\setminus\{\,n+1\,\}, \ldots, A_k\,\}$ is a k-partition of the n-set $A\setminus\{\,n+1\,\}$, since $A_i\setminus\{\,n+1\,\}\neq\emptyset$.

Conversely, consider a k-partition $\{\,A_1, A_2, \ldots, A_k\,\}$ of the n-set $A\setminus\{\,n+1\,\}$. This partition defines k k-partitions $\{\,A_1, A_2, \ldots,$ $A_i\cup\{\,n+1\,\}, \ldots, A_k\,\}$ of the set $(n+1)$-set A for $i = 1, 2, \ldots, k$.

Therefore the number of desired partitions in this case is $k\left\{ {n \atop k} \right\}$.

Since case 1 and case 2 are mutually disjoint, we have by the rule of sum (see Fact 2.2.0.1),

$$\left\{ {n+1 \atop k} \right\} = k\left\{ {n \atop k} \right\} + \left\{ {n \atop k-1} \right\}$$

Thus the proof is complete. ■

The above recurrence relation allows us to compute the numbers $\left\{ {n \atop k} \right\}$ starting from the initial conditions $\left\{ {0 \atop 0} \right\} = 0$ and $\left\{ {n \atop 1} \right\} = 1$ for $n \geq 1$. The following table, called the Stirling triangle of the second kind, constructs the numbers $\left\{ {n \atop k} \right\}$ iteratively. To obtain the k-th number in the $(n+1)$-th line, (assuming that n lines have already been constructed), we multiply by k the kth number of the nth line and add it to the $(k-1)$-th number of the n-th line. The table may be continued to any number of lines. Note that the first line is the 0-th line.

For example, the 4-th line gives the numbers $\left\{ {4 \atop 1} \right\}$; $\left\{ {4 \atop 2} \right\}$; $\left\{ {4 \atop 3} \right\}$; $\left\{ {4 \atop 4} \right\}$; and $\left\{ {4 \atop 5} \right\}$.

Table 2.7: Stirling triangle of the second kind

				1				
			1		0			
		1		1		0		
	1		3		1		0	
1		7		6		1		0
				⋮				

Example 2.7.2 (Stirling numbers of the first and second kinds): Prove that $\left\{ {n \atop k} \right\} \leq \left[{n \atop k} \right]$.

Combinatorial proof: Every partition of $[n]$ into nonempty subsets induces at least one cycle. For example, the singleton sets $\{ 1 \}$ and $\{ 1, 2 \}$ give only one cycle (1) and $(1, 2) = (2, 1)$ respectively. All other sets of at least 3 elements give more than one cycle. For example, the set of three elements $\{ 1, 2, 3 \}$ give two different cycles $(1, 2, 3)$ and $(1, 3, 2)$. Hence the claimed inequality.

Example 2.7.3 (Stirling numbers of the first and second kinds): Prove that

$$\left[{n \atop n-1} \right] = \left\{ {n \atop n-1} \right\} = \binom{n}{2}$$

Solution: By definition, $\left[{n \atop n-1} \right]$ is the coefficient of x^{n-1} in the product $x(x+1)(x+2)\cdots(x+n-1)$, that is, the coefficient of x^{n-2} in

$$\begin{aligned}&(x+1)(x+2)(x+3)\cdots(x+n-1) = x^{n-2}\\&+(1+2+3+\cdots+n-1)x^{n-2}+\cdots+(n-1)!\end{aligned}$$

Hence $\left[{n \atop n-1} \right] = 1 + 2 + \cdots + (n-1) = (n-1)n/2 = \binom{n}{2}$.

Now, by definition, $\left\{ {n \atop n-1} \right\}$ is the number of unordered partitions of the n-set $A = \{ 1, 2, \ldots, n \}$ into $(n-1)$ mutually disjoint non-empty subsets of A. The sets of such a partition are of the form $\{ A_1, A_2, \ldots, A_i, \ldots, A_{n-1} \}$ where exactly one of the A_i's is a two-element subset of A and all others are singleton subsets of A. Note that once we pick a 2-subset A_i of A, all other sets in the partition are uniquely determined.

Therefore, the number of such partitions is the same as the number of two-element subsets of the n-set, which by definition is $\binom{n}{2}$.

Example 2.7.4 (Property):
Let p be a prime number. Then prove that $\left\{ {p \atop k} \right\}$ for all k with $1 < k < p$.

Solution: Similar to Example 2.6.13.

Example 2.7.5 (Stirling number of the second kind):
Prove that for non-negative integer n,

$$\left\{ {n+1 \atop 2} \right\} = 2^n - 1.$$

Solution: By definition, $\left\{ {n+1 \atop 2} \right\}$ is the number of *unordered* partitions of the $(n+1)$-set $A = \{ 1, 2, \ldots, n+1 \}$ into two mutually disjoint non-empty subsets of A.

First we count the number of *ordered* 2-partitions of the set A (empty set allowed as a part of the partition.)

By definition of the multinomial coefficient, the number of *ordered* $(k, n-k)$ 2-partition of the n-set A ($0 \leq k \leq n$) is $\binom{n}{k, n-k} = \frac{n!}{k!(n-k)!} = \binom{n}{k}$. Hence the number of ordered 2-partitions of the $(n+1)$-set is

$$\sum_{k=1}^{n} \binom{n}{k} = 2^{n+1} \text{ by Corollary 2.4.1 }.$$

These 2^{n+1} ordered 2-partitions of A include the special two ordered partitions $(\emptyset, A)$ and $(A, \emptyset)$ of which one part is empty. If we remove these two partitions, then we have the number of ordered 2-partitions (A_1, A_2) (with $A_1 \neq \emptyset$ and $A_2 \neq \emptyset$) of the $(n+1)$-set which is $2^{n+1} - 2$.

But the two *ordered* 2-partitions (A_1, A_2) and (A_2, A_1) define *only one unordered* partition $\{ A_1, A_2 \}$. Therefore, the number of *unordered* partitions of the set A into two mutually disjoint non-empty subsets is $\frac{1}{2} \times (2^{n+1} - 2) = 2^n - 1$.

Number of surjective functions from an n-set onto an m-set ($n \geq m$): We shall now find the number of surjective functions from an n-set onto an m-set using the Stirling number of the second kind. The following example illustrates how a surjective function f from an n-set X onto an m-set Y induces an unordered partition of the domain set X into mutually disjoint non-empty subsets of X.

First we recall the idea of an inverse image of any function $f : X \to Y$. For an element $y \in Y$, $f^{-1}(y)$ is the set of all elements $x \in X$ whose image under f is y. Symbolically,

$$f^{-1}(y) = \{ x \in X \mid f(x) = y \}$$

Example 2.7.6 (Partition induced by a surjective function):
Let $X = \{ p_1, p_2, \ldots p_{10} \}$ be a set of professors in the Department of Mathematics and let $Y = \{ r_1, r_2, \ldots, r_6 \}$ be the rooms in the Department of Mathematics in a university. Consider the surjective function $f : X \to Y$ which assigns to each professor his room in the Department of Mathematics. f is written in the matrix form where

$$f = \begin{pmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} \\ r_3 & r_6 & r_3 & r_1 & r_2 & r_4 & r_5 & r_4 & r_3 & r_6 \end{pmatrix}$$

The function is clearly surjective because every room is occupied by at least one professor. Now for room $r \in Y$, $f^{-1}(r)$ is the set of professors occupying the room r and this set is non-empty because f is surjective. The partition induced by the function f is

$f^{-1}(r_1) \cup f^{-1}(r_2) \cup \cdots \cup f^(-1)(r_6) = \{ p_4 \} \cup \{ p_5 \} \cup \{ p_1, p_3, p_9 \} \cup \{ p_6, p_8 \} \{ p_7 \} \cup \{ p_2, p_{10} \}$.

With this example in mind, we state the following proposition.

Proposition 2.7.1:
Let $f : X \to Y$ be a surjective function from an n-set X onto an m-set Y. Then f induces an unordered partition of the domain set X into m mutually disjoint non-empty subsets of X.

Proof. If $Y = \{ y_1, y_2, \dots, y_m \}$, then the required partition of the set X into m mutually disjoint non-empty subsets is

$$\{ f^{-1}(y_1), f^{-1}(y_2), \dots, f^{-1}(y_m) \}$$

■

The following theorem gives the number of surjective functions from an n-set onto an m-set.

Theorem 2.7.1:
The number of surjective functions from an n-set onto an m-set $(n \geq m)$ is $\left\{ {n \atop m} \right\} m!$.

Proof. Let the domain set be the n-set $X = \{ x_1, x_2, \dots, x_n \}$ and let the range set be the m-set $Y = \{ y_1, y_2, \dots, y_m \}$. By Proposition 2.7.1, every surjective function from X onto Y induces a partition of X into mutually disjoint non-empty subsets of X.

Now consider an unordered m-partition $\{ X_1, X_2, \dots, X_m \}$ of the set X. Consider *any* permutation (written in one-line notation) $p_i = (y_{i_1} y_{i_2} \dots y_{i_m})$ of the m-set Y. This permutation defines a surjective function p_i from X onto Y where

$$p_i(x_j) = y_{i_j} \text{ if } x_j \in X_j,\ 1 \leq j \leq n$$

That is, p_i maps every element of the subset X_j onto the element y_{i_j}. Since there are $m!$ permutations possible on the m-set Y, each m-partition $\{ X_1, X_2, \dots, X_m \}$ of the set X induces $m!$ surjections from X onto Y. But the number partitions possible of the n set X into m mutually disjoint non-empty sets is the Stirling number of the second kind $\left\{ {n \atop m} \right\}$, by definition. Therefore by the product rule (see Proposition 2.2.2), the number of surjections from the n-set X onto the m-set Y is $\left\{ {n \atop m} \right\} m!$. ■

Example 2.7.7:
The number of surjective functions from the 4-set $X = \{ x_1, x_2, x_3, x_4 \}$ onto the 3-set $Y = \{ y_1, y_2, y_3 \}$ is $\left\{ 4 \atop 3 \right\} 3!$. From the Table 2.7, $\left\{ 4 \atop 3 \right\} = 6$. Hence the number of surjections is $6 \times 3! = 36$.

Example 2.7.8:
Find the number of ways to partition n elements in m different boxes $B_1, B_2, \ldots, B_m$ in such a way that k of these boxes are non-empty and $m - k$ of these boxes are empty.

Solution:

The number of ways to pick k boxes out of m boxes is $\binom{m}{k}$ ways. The number of ways to partition (unordered partition) n elements into k mutually disjoint subsets is the Stirling number $\left\{ n \atop k \right\}$. Hence, the number of ways to put n elements into k boxes $B_{i_1}, B_{i_2}, \ldots, B_{i_k}$ such that no box is empty is the same as number of *ordered* partitions of n-elements into k non-empty subsets which is $k! \left\{ n \atop k \right\}$ (since each unordered partition into k non-empty subsets gives rise to $k!$ ordered partitions). Therefore by the product rule of Proposition 2.2.2, the desired number is $k! \left\{ n \atop k \right\} \binom{m}{k}$.

Example 2.7.9:
Find the number of equivalence relations that can be defined on the set $[n] = \{ 1, 2, \ldots, n \}$.

Solution: We know that every equivalence relation on $[n]$ partitions the set $[n]$ into nonempty mutually disjoint subsets called the equivalence classes and given any partition of $[n]$ into nonempty mutually disjoint subsets, we can define an equivalence relation on $[n]$ whose equivalence classes are exactly the given partitions of the set.

Hence by the definition of the Stirling number of the second kind, the number of equivalence relations on $[n]$ is the number of possible partitions of $[n]$ into k $(1 \leq k \leq n)$ nonempty mutually disjoint subsets. This number is

$$\sum_{k=1}^{n} \left\{ n \atop k \right\}.$$

Corollary 2.7.1.1:
For positive integers m and n we have the equality

$$m^n = \sum_{k=1}^{n} \genfrac{\{}{\}}{0pt}{}{n}{k} m^{\underline{k}}$$

Proof. By Theorem 2.2.2, the left-hand side of the equality m^n is the number of functions from an n-set $X = \{ x_1, x_2, \dots, x_n \}$ to the m-set $Y = \{ y_1, y_2, \dots, y_m \}$.

Now let us compute the number of functions from X to Y in a different way. Let us first observe that any function $f : X \to Y$ becomes a surjective function if we restrict the co-domain set Y to the subset $f(X) = \{ f(x) \mid x \in X \}$, the set of all images of the elements of the domain set X. Furthermore, $1 \leq |f(X)| \leq n$.

Consider a k-subset T of Y where k is an integer such that $1 \leq k \leq n$. By Theorem 2.7.1, the number of surjections from X onto the k-set T is $\genfrac{\{}{\}}{0pt}{}{n}{k} k!$. But a k-subset T of the m-set Y can be chosen in $\binom{m}{k}$, by the definition of the binomial coefficient. Therefore, by the product rule (Proposition 2.2.2), the number of surjections from X onto a k-subset of Y is $\genfrac{\{}{\}}{0pt}{}{n}{k} k! \binom{m}{k}$. Since the integer k may assume any value between 1 and n, we have that the number of surjections from X onto a subset of Y is

$$\sum_{k=1}^{n} \genfrac{\{}{\}}{0pt}{}{n}{k} k! \binom{m}{k} = \sum_{k=1}^{n} \genfrac{\{}{\}}{0pt}{}{n}{k} m^{\underline{k}}$$

■

Example 2.7.10:
Verify the equality in the Corollary 2.7.1.1, for $n = 3$ and $m = 4$.

The left-hand side is $m^n = 4^3 = 64$.

The right-hand side of the equality is

$$\begin{aligned} \sum_{k=1}^{n} \genfrac{\{}{\}}{0pt}{}{n}{k} m^{\underline{k}} &= \sum_{k=1}^{3} \genfrac{\{}{\}}{0pt}{}{3}{k} 4^{\underline{k}} \\ &= \genfrac{\{}{\}}{0pt}{}{3}{1} 4 + \genfrac{\{}{\}}{0pt}{}{3}{2} (4)(3) + \genfrac{\{}{\}}{0pt}{}{3}{3} (4)(3)(2) \end{aligned}$$

$$\begin{aligned} &= (1)(4) + (3)(4)(3) + (1)(4)(3)(2) \\ &= 4 + 36 + 24 \\ &= 64 \end{aligned}$$

Consider the equality $m^n = \sum_{k=1}^{n} \left\{ {n \atop k} \right\} m^{\underline{k}}$ of the Corollary 2.7.1.1. This equation can be proved to be true for any real number x. To do this, we need the following simple lemma concerning polynomials. To prove the lemma, we need the Fundamental Theorem of Algebra, which we state without proof.

Theorem 2.7.2 (Fundamental theorem of algebra):
Consider a polynomial $p(z) = p_0 + p_1 z + p_2 z^2 + \cdots + p_n z^n$ of degree n with complex coefficients p_i. Then $p(z)$ has exactly n complex roots (distinct or coincident).

The following lemma is very useful to extend the validity of some equations from integers to all real numbers.

Lemma 2.7.2:
Let $p(x) = p_0 + p_1 x + \cdots + p_n x^n$ and $q(x) = q_0 + q_1 x + \cdots + q_n x^n$ be two non-zero polynomials of degree n. If $p(x) = q(x)$ for $n+1$ distinct real numbers $r_0, r_1, \ldots, r_n$ then the polynomials $p(x)$ and $q(x)$ are identically equal, symbolically, $p(x) \equiv q(x)$.

Proof. $p(x) = q(x)$ for $x = r_0, r_1, \ldots, r_n$. Therefore $r(x) = p(x) - q(x)$ is a polynomial of degree at most n. Moreover, the polynomial $r(x)$ of degree at most n is such that $s(r_i) = p(r_i) - q(r_i) = 0$ for $i = 0, 1, \ldots, n$. In other words, the polynomial $s(x)$ which is of degree at most n has $n+1$ distinct roots. Hence by the fundamental theorem of algebra 2.7.2, the polynomial $s(x)$ must be identically zero, that is, $s(x) = 0$ for all real numbers. That is, $p(x) - q(x) \equiv 0$, that is, $p(x) \equiv q(x)$. ■

Corollary 2.7.2.1:
If x is any real number and n is a positive integer then we have

the equality

$$x^n = \sum_{k=1}^{n} \left\{ {n \atop k} \right\} x^{\underline{k}}$$

Proof. By Theorem 2.7.1.1, we have the equality

$$m^n = \sum_{k=1}^{n} \left\{ {n \atop k} \right\} m^{\underline{k}} \text{ for positive integers } m, n.$$

The left-hand side and the right-hand side of the above equation may be viewed as polynomials of degree n in the variable m. These two polynomials of degree n in the variable m assume the same values for $n+1$ distinct values of m where $m = 0, 1, 2, \dots, n$ (since for $m = 0$, both polynomials are 0.) Hence by Lemma 2.7.2, the left-hand side is *identically* equal to the right-hand side. This proves the corollary. ■

THEOREM 2.7.3:

$$\left\{ {n+1 \atop k} \right\} = \sum_{p=0}^{n} \binom{n}{p} \left\{ {p \atop k-1} \right\}.$$

Proof. Consider the $(n+1)$-set $A = \{ 1, 2, \dots, n, n+1 \}$ and a partition $\mathcal{P} = \{ P_1, P_2, \dots, P_k \}$ of A into k mutually disjoint nonempty subsets.

Let P_i be the set containing the integer $n+1$. Then by *removing* the set P_i from $\mathcal{P}$, we get the partition $\{ P_1, P_2, \dots, P_{i-1}, P_{i+1}, P_k \}$ of the subset $A \setminus P_i$ into $k-1$ mutually disjoint subsets.

Conversely, given a subset S of A with p elements $(0 \le p \le n)$ not containing the integer $n+1$ and a partition $\mathcal{P} = \{ P_1, P_2, \dots, P_{k-1} \}$ of S into $k-1$ mutually disjoint subsets, we get a partition $\mathcal{P} \cup (A \setminus S)$ of A by *adding* the set $A \setminus S$ to $\mathcal{P}$.

But a subset S of A consisting of p elements not containing the integer $n+1$ may be picked in $\binom{n}{p}$ ways. Thus we have established a one-to-one correspondence between the family of all partitions of the set A and the family of all partitions of the subsets of A not containing the element $n+1$.

Therefore, by the Proposition 2.2.1, we have the equality

$$\left\{ {n+1 \atop k} \right\} = \sum_{p=0}^{n} \binom{n}{p} \left\{ {p \atop k-1} \right\}.$$

■

Stirling number of the second kind as the sum of a product:
Like Stirling number of the first kind, the Stirling number of the second kind may be defined as the sum of a product of certain integers.

Definition 2.7.1 (Stirling number of the second kind as the sum of a product):
To define $\left\{ {n \atop k} \right\}$, we consider the k-set $A = \{ 1, 2, \ldots, k \}$. Now $\left\{ {n \atop k} \right\}$ is the sum of the product of integers in all $n - k$-*multisubsets* of the set A. Symbolically,

$$\left\{ {n \atop k} \right\} = \sum_{1 \leq k_1 \leq k_2 \leq \cdots \leq k_{n-k} \leq k} k_1 k_2 \cdots k_{n-k}$$

Example 2.7.11 (Stirling number of the second kind as the sum of a product):
Find $\left\{ {5 \atop 3} \right\}$.

Solution: Consider the 3-set $A = \{ 1, 2, 3 \}$. The possible $(5 - 3) = 2$-multisubsets of the set A are: $\{ 1, 2 \}$; $\{ 1, 3 \}$; $\{ 2, 3 \}$; $\{ 1, 1 \}$; $\{ 2, 2 \}$; $\{ 3, 3 \}$. Hence

$$\left\{ {5 \atop 3} \right\} = (1 \times 2) + (1 \times 3) + (2 \times 3) + (1 \times 1) + (2 \times 2) + (3 \times 3) = 25.$$

2.8 Bell Numbers

The total number of unordered partitions of an n-set into mutually disjoint subsets is the n-th Bell number, denoted by B_n. Since $\left\{ {n \atop k} \right\}$ is the number of unordered partitions of an n-set into k mutually disjoint non-empty subsets, we have the following proposition.

Proposition 2.8.1:

$$B_n = \left\{ {n \atop 1} \right\} + \left\{ {n \atop 2} \right\} + \cdots + \left\{ {n \atop n} \right\}.$$

The following proposition gives a recurrence relation involving Bell numbers.

Proposition 2.8.2 (Recurrence relation involving Bell numbers):

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k.$$

Proof. Since $\binom{n}{k} = \left\{ {n \atop k} \right\} = 0$ for all $k > n$, we may write the equation of the Theorem 2.7.3 as

$$\left\{ {n+1 \atop k} \right\} = \sum_{p=0}^{\infty} \binom{n}{p} \left\{ {p \atop k-1} \right\} \tag{2.2}$$

Hence by the Proposition 2.8.1, we have

$$\begin{aligned} B_{n+1} &= \sum_{k=1}^{n+1} \left\{ {n+1 \atop k} \right\} \\ &= \sum_{k=1}^{\infty} \left\{ {n+1 \atop k} \right\} \text{ since } \left\{ {n \atop k} \right\} = 0 \text{ if } k > n \\ &= \sum_{k=1}^{\infty} \sum_{p=0}^{n} \binom{n}{p} \left\{ {p \atop k-1} \right\} \text{ by Theorem 2.7.3} \\ &= \sum_{p=0}^{n} \binom{n}{p} \left(\sum_{k=1}^{\infty} \left\{ {p \atop k-1} \right\} \right) \text{ by interchanging summation} \\ &= \sum_{p=0}^{n} \binom{n}{p} \left(\sum_{k=1}^{p} \left\{ {p \atop k-1} \right\} \right) \\ &= \sum_{p=0}^{n} \binom{n}{p} B_p \text{ by Theorem 2.7.3} \end{aligned}$$

■

2.9 The Principle of Inclusion and Exclusion

Consider a set A consisting of n elements. Let $A_1, A_2, \dots, A_m$ be m subsets of the set A. The inclusion and exclusion principle counts the number of elements of the set A which are in *exactly* k of the subsets $A_1, A_2, \dots, A_m$. Inclusion and exclusion principle is an important technique in enumeration problems.[4][5]

The principle can be considered as a generalization of the following example:

Example 2.9.1:
In a class of 40 students, 14 like mathematics, 16 like physics, and 15 like chemistry; 7 like both mathematics and physics, 9 both physics and chemistry, and 6 students like mathematics and chemistry. There are 5 who like all three subjects. How many students do not like any of the three subjects?

Solution: First we subtract from the total number of students, namely 40, the number of students who like mathematics, physics, chemistry respectively:

$$40 - 14 - 16 - 15.$$

In the above expression, a student who likes both mathematics and physics is subtracted *twice*; so we have to add them back, and similarly for the two other pairs of subjects: This gives the number $40 - 14 - 16 - 15 + 7 + 9 + 6$.

But now a student who likes all the three subjects is subtracted thrice and added thrice. In order to get the desired number we have to subtract the number of students who like all the three once more. Hence the desired integer is $40-14-16-15+7+9+6-5$.

Example 2.9.2:
Let $A = \{1, 2, \dots, 10\}$. Consider the three subsets A_1, A_2, A_3 of the set A where $A_1 = \{2, 4, 6, 7, 8\}$ $A_2 = \{1, 2, 4, 6, 8, 10\}$ $A_3 = \{1, 4, 5, 6, 8, 9, 10\}$ (see the shaded portion of Figure 2.6).

The set of elements of A lying in *exactly* two of the subsets is $\{1, 2, 10\}$ (because the element 1 lies in $A_2 \cap A_3$ and 2 in $A_1 \cap A_2$ but

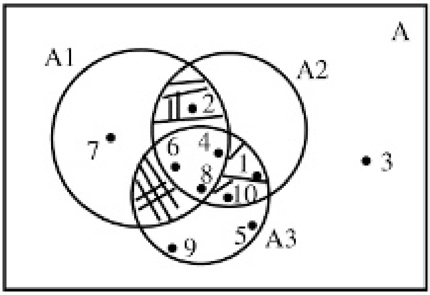


Figure 2.6: Inclusion-exclusion principle

$1, 2 \notin A_1 \cap A_2 \cap A_3$, the element $10 \in A_1 \cap A_3$ but $10 \notin A_1 \cap A_2 \cap A_3$). Hence the number of elements in *exactly* two of the subsets is 3.

Example 2.9.3:
Given two subsets A_1 and A_2 of a finite set A, express the number of elements in *exactly* one of the subsets A_1, A_2 in terms of $A_1, A_2,\ $ and$A_1 \cap A_2$.

Solution:

The set of elements belonging to *exactly* one of the subsets is

$$(A_1 \setminus (A_1 \cap A_2)) \cup (A_2 \setminus (A_1 \cap A_2)) \,.$$

But the two sets $(A_1 \setminus (A_1 \cap A_2))$ and $(A_2 \setminus (A_1 \cap A_2))$ are mutually disjoint. Therefore by the sum rule (see Fact 2.2.0.1),

$$\begin{aligned} & |\,(A_1 \setminus (A_1 \cap A_2)) \cup (A_2 \setminus (A_1 \cap A_2)) \\ | = \; & |\,(A_1 \setminus (A_1 \cap A_2))\,| + |\,(A_2 \setminus (A_1 \cap A_2))\,| \\ = \; & (|A_1| - |A_1 \cap A_2) + (|A_2| - |A_1 \cap A_2|) \\ = \; & |A_1| + |A_2| - 2|A_1 \cap A_2| \end{aligned}$$

Note that we have used the subtraction rule in the above derivation: if $A \subset B$, then $|B \setminus A| = |B| - |A|$.

The inclusion and exclusion principle generalizes the above example. Before stating and proving the formula, we prove a simple equation involving binomial coefficients.

FACT 2.9.0.1:

$$\binom{r}{s}\binom{p}{r} = \binom{p}{s}\binom{p-s}{p-r} \text{ where } s \leq r \leq p$$

Proof.

$$\begin{aligned}
\binom{r}{s}\binom{p}{r} &= \frac{r!}{s!(r-s)!} \times \frac{p!}{r!(p-r)!} \quad \text{by formula} \\
&= \frac{p!}{s!(p-s)!} \times \frac{(p-s)!}{(p-r)!((p-s)-(p-r)!)} \\
&\quad \text{multiplying and dividing by } (p-s)! \\
&= \binom{p}{s}\binom{p-s}{p-r} \quad \text{by formula}
\end{aligned}$$

■

THEOREM 2.9.1 (Inclusion and exclusion principle):
Consider a set S of n elements and m subsets $S_1, S_2, \ldots, S_m$ of the set S. For $i_{1,2}, \ldots, i_r$ with $1 \leq i_1 < i_2 < \cdots \leq m$, let $A(i_1, i_2, \ldots, i_r)$ be the number of elements in the intersection $S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_r}$. Then the number of elements of S lying in *exactly* k of the subsets $S_1, S_2, \ldots, S_m$ is given by the formula

$$E(k) = A_k - \binom{k+1}{k} A_{k+1} + \binom{k+2}{k} A_{k+2} - \cdots + (-1)^{m-k} \binom{m}{k} A_m$$

where

$$A_r = \sum_{1 \leq i_1 < i_2 < \cdots < i_r \leq m} A(i_1, i_2, \ldots, i_r).$$

Proof. Consider an element $s \in S$ belonging to *exactly* p of the subsets $S_1, S_2, \ldots, S_m$. We shall prove that the element s contributes *zero* to the right-hand side of the formula if either $p < k$ or $p > k$ and the element s contributes exactly *one* to the right-hand side if $p = k$.

Case 1: $p < k$.

Since the element s belongs to at most $k-1$ of the subsets $S_1, S_2, \ldots, S_m$, it contributes zero to $A_k, A_{k+1}, \ldots, A_m$. Hence the contribution of s to the right-hand side of the formula is zero.

Case 2: $p > k$.

Set $k+q = p$ where $q > 0$ is a positive integer.

Let us first observe that $A_r = \sum_{1 \leq 1 < i_1 < i_2 < \cdots < i_r \leq m} A(i_1, i_2, \ldots, i_r)$ has exactly $\binom{m}{r}$ terms (since the subscripts in the sum are taken over each r-subset of $\{1, 2, \ldots, m\}$). Therefore the contribution of the element s to the first term A_k is $\binom{p}{k}$, the contribution of s to the second term $-\binom{k+1}{k}A_{k+1}$ is $-\binom{k+1}{k}\binom{p}{k+1}$ etc. and finally the contribution of s to the $(k+q+1)$st term is $(-1)^q\binom{k+q}{k}\binom{s}{k+q}$. The element s contributes zero to A_r if $r > p$ by the definition of A_r. Therefore, the contribution of s to the right-hand side of the formula is

$$\binom{p}{k}\binom{k}{k} - \binom{k+1}{k}\binom{p}{k+1} + \binom{k+2}{k}\binom{p}{k+2} - \cdots$$
$$+(-1)^{p-k}\binom{p}{k}\binom{p}{p}$$

Using the fact that $\binom{r}{s}\binom{p}{r} = \binom{p}{s}\binom{p-s}{p-r}$ where $s \leq r \leq p$ (see Fact 2.9.0.1), the contribution of s to the right-hand side is

$$\binom{p}{k}\binom{p-k}{p-k} - \binom{p}{k}\binom{p-k}{p-(k+1)} + \binom{p}{k}\binom{p-k}{p-(k+2)} - \cdots$$
$$+(-1)^{p-k}\binom{p-k}{p-p}$$

which is equal to (by taking $\binom{p}{k}$ as a a common factor and using the duality relation of the binomial coefficient $\binom{p}{k} = \binom{p}{p-k}$ (see 2.3.8).

$$\binom{p}{k}\left(\binom{p}{0} - \binom{p}{1} + \binom{p}{2} - \cdots (-1)^{p-k}\binom{p-k}{p-k}\right)$$

But by the third equation of Corollary 2.4.2.1, the bracketed expression is zero. Hence the contribution of s to the right-hand side is zero in this case.

Case 3: $p = k$.

The contribution of s to the first term $A_k = A_p$ of the right-hand side of the formula is clearly 1. The contribution of s to $A_{k+1}, A_{k+2}\cdots$ is clearly zero, since s belongs to exactly k of the subsets. Therefore, the total contribution of s to the right side of the formula is $1 - 0 + 0 - \cdots = 1$.

From these three cases we conclude that each element belonging to exactly k of the subsets $S_1, S_2 \ldots, S_m$ is counted once to the right-hand side of the formula and each other element contributes zero. Hence right-hand side counts *exactly* the number of elements of A belonging to exactly k of the subsets. Hence the formula. ■

The alternating signs "+" and "-" ("inclusion for + and exclusion for −") in the inclusion and exclusion formula justifies its name. Also note that the formula express the "exact number of elements in k of the subsets " ("E" for exact) in terms of "number of elements in at least k of the subsets ("A" for at least) in at least $(k+1)$ of the subsets, etc.)."

As a special case of the above Theorem 2.9.1 we derive the following useful corollary.

Corollary 2.9.1.1:
Consider a set S of n elements and m subsets $S_1, S_2, \ldots, S_m$ of the set S. For $i_{1,2}, \ldots, i_r$ with $1 \leq i_1 < i_2 < \cdots \leq m$, let $A(i_1, i_2, \ldots, i_r) = |S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_r}|$. Then the number of elements in *none* of the subsets is

$$n - A_1 + A_2 - \cdots + (-1)^m A_m$$

where

$$A_r = \sum_{1 \leq 1 < i_1 < i_2 < \cdots < i_r \leq m} A(i_1, i_2, \ldots, i_r).$$

Proof. By the inclusion and exclusion principle (see Theorem 2.9.1), the number of elements in exactly k of the subsets $S_1, S_2, \ldots, S_m$ is

$$E(k) = A_k - \binom{k+1}{k} A_{k+1} + \binom{k+2}{k} A_{k+2} - \cdots + (-1)^{m-k} A_m$$

Setting $k = 0$ we have that the number of elements in none of the subsets $S_1, S_2, \ldots, S_m$ is

$$E(0) = A_0 - \binom{1}{0} A_1 + \binom{2}{0} A_2 - \cdots + (-1)^m \binom{m}{0} A_m$$

But $\binom{p}{0} = 1$ for any non-negative integer p. Moreover, A_0, by definition, is the number of elements of S belonging to at least zero of the subsets $S_1, S_2, \ldots, S_m$ which is the number of elements of S. Therefore, the number of elements in none of the subsets is

$$E(0) = n - A_1 + A_2 - \cdots + (-1)^m A_m.$$

■

2.9.1 Applications of Inclusion and Exclusion Principle

A formula for the number of derangements of n elements: A *derangement* is a permutation p on an n-set $A = \{1, 2, \ldots, n\}$ such that $p(i) \neq i$ for all $i = 1, 2, \ldots, n$. In other words, derangements are permutations with no *fixed elements*. (An element i is a fixed element of a permutation p if $p(i) = i$.) A derangement is a permutation in which no element occupies its original position.

A permutation written as a product of disjoint cycles is a derangement if it has *no* singleton cycle.

We are interested in finding a formula for the number of derangements. Before stating and proving the formula, we consider an example to illustrate the notion.

Example 2.9.4 (Derangements):
Find the number of derangements of a 3-element set.

Solution: Consider the 3-set $A = \{1, 2, 3\}$. There are $3! = 1 \times 2 \times 3 = 6$ permutations possible with the elements of A. Let us write down the 6 permutations in one-line notation. These are:

$$123; 132; 213; 231; 312; 321$$

Of these six permutations, the derangements, that is, permutations with no fixed elements are: $p = 231; q = 312$. That is, $p(1) = 2, p(2) = 3, p(3) = 1$ and $q(1) = 3, q(2) = 1, q(3) = 2$. Hence the number of derangements of a 3-set is 2.

In cycle notation $p = (123)$ and $q = (132)$ and we see p and q have no singleton cycles. Hence p and q are the only two derangements.

Theorem 2.9.2 (Formula for the number of derangements): The number of derangements d_n on n elements is

$$d_n = n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + (-1)^n \frac{1}{n!}\right)$$

Proof. Let S be the set of all $n!$ permutations of the n-set $\{1, 2, \ldots, n\}$. For $i = 1, 2, \ldots, n$ let S_i be the subset of S consisting of all permutations on $1, 2, \ldots, n$ for which the element i is a fixed element. In other words, S_i consists of all permutations written in cycle notation for which the element i forms a singleton cycle.

Therefore, the number of derangements d_n is the number of elements of S which are in none of the subsets $S_1, S_2, \ldots, S_n$. By Corollary 2.9.1.1, the number of elements of S which are in none of the subsets $S_1, S_2, \ldots, S_n$ is

$$E(0) = A_0 - \binom{1}{0}A_1 + \binom{2}{0}A_2 - \cdots + (-1)^n \binom{n}{0}A_n$$

Our problem reduces to calculating A_r for $r = 1, 2, \ldots, n$. By definition of S_i, $A_r = \sum_{1 \le i_1 < i_2 < \cdots < i_r \le n} |S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_r}|$ is the number of permutations in the set S in which the elements $i_1, i_2, \ldots, i_r$ form singleton cycles/fixed elements. In order to calculate A_r, we first calculate the general term of A_r which is $|S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_r}|$. To find $|S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_r}|$, fix the r elements $i_1, i_2, \ldots, i_r$ in their respective positions. The remaining $(n - r)$ elements may be permuted in $(n - r)!$ ways. Hence $|S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_r}| = (n - r)!$ Since there are $\binom{n}{r}$ terms in the sum A_r, we have $A_r = \binom{n}{r}(n - r)!$.

Substituting A_r for $r = 0, 1, \ldots, n$ in

$$E(0) = A_0 - \binom{1}{0}A_1 + \binom{2}{0}A_2 - \cdots + (-1)^n\binom{n}{0}A_n,$$

we get

$$\begin{aligned} E(0) &= \binom{n}{0}(n-0)! - \binom{n}{1}(n-1)! + \binom{n}{2}(n-2)! \\ & \quad -\binom{n}{3}(n-3)! + \cdots + (-1)^n\binom{n}{n}(n-n)! \\ &= \frac{n!}{0!} - \frac{n!}{(n-1)!} + \frac{n!}{(n-2)!} - \frac{n!}{(n-3)!} \\ & \quad + \cdots + (-1)^n\frac{n!}{(n-n)!} \quad \text{using } \binom{n}{k} = \frac{n!}{k!(n-k)!} \\ &= n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots (-1)^n\frac{1}{n!}\right) \end{aligned}$$

■

Remark 2.9.1 (Number of derangements):
By Theorem 2.9.2, the number of derangements d_n on n symbols is

$$d_n = n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + (-1)^n\frac{1}{n!}\right)$$

But we know that the inverse of *the number e* is given by the alternating infinite series

$$e^{-1} = 1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots (-1)^n\frac{1}{n!} + (-1)^{n+1}\frac{1}{(n+1)!} + \cdots$$

Multiplying both sides by $n!$ we get,

$$\begin{aligned} n!e^{-1} &= n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots (-1)^n\frac{1}{n!}\right) \\ & \quad +(-1)^{n+1}\frac{1}{n+1} + (-1)^{n+2}\frac{1}{(n+1)(n+2)} + \cdots \\ &= d_n + (-1)^{n+1}\frac{1}{n+1} + (-1)^{n+2}\frac{1}{(n+1)(n+2)} + \cdots \end{aligned}$$

If n is sufficiently large, the sum $(-1)^{n+1}\frac{1}{n+1}+(-1)^{n+2}\frac{1}{(n+1)(n+2)}+\cdots$ can be made arbitrarily small. Hence we may take $\frac{n!}{e}$ as a good approximation for the number of derangements. In fact, $\lfloor\frac{n!}{e}\rceil$, the *nearest integer* to the number $\frac{n!}{e}$ is the exact value of d_n. (For example, $\lfloor 5.5\rceil = \lfloor 5.9\rceil = 6$ whereas $\lfloor 5.4\rceil = 5$.)

Notation for the number of derangements on n symbols. This number is called n subfactorial in the book (see [6]) and denoted by n followed by the exclamation symbol inverted like factorial inverted.

Example 2.9.5:
Find the number of derangements on 3 symbols and on 4 symbols.

Solution: The number of derangements on 3 symbols is $3!(1-\frac{1}{1!}+\frac{1}{2!}-\frac{1}{3!}) = 6-6+3-1 = 2$.

By Remark 2.9.1, the number of derangements on 4 symbols is $\lfloor\frac{4!}{e}\rceil = \lfloor\frac{24}{2.7}\rceil = \lfloor 8.8\rceil = 9$ (since the real number $e \approx 2.7$).

Example 2.9.6 (A problem in probability):
(See Chapter 6 for more details.) Consider n gentlemen attending a party. Their n hats are placed in a checkroom. Then the n hats are mixed and returned at random to the gentlemen after the party is over. Find the probability that no one receives his own hat.

Solution: By definition

$$\text{The probability} = \frac{\text{The number of favorable cases}}{\text{The total number of possible cases}}$$

The total number of possible cases is the total number of ways in which the n hats can be mixed which is the number of permutations of n hats. Hence the total number of possible cases is $n!$.

The number of favorable cases is the number of ways in which no gentleman gets his *own* hat, which is nothing but the number of permutations of n hats in which the i-th man will not get his *own* hat for all $i = 1, 2, \ldots, n$. This is the number of derangements of n hats, which by Theorem 2.9.2 is the number

$d_n = n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + (-1)^n \frac{1}{n!}\right)$. Hence the required probability is $\frac{d_n}{n!} = 1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + (-1)^n \frac{1}{n!} \approx \frac{1}{e}$ by Remark 2.9.1, if n is sufficiently large.

Example 2.9.7 (A chess board problem):
Consider the usual 8 × 8 chess board. Suppose we are given eight rooks. Find the number of ways in which these eight rooks can be placed on the board in such a way that no rook can attack another (that is, no two rooks are in the same row (line) or in the same column) and the second main diagonal/the black diagonal (diagonal from the bottom left square to the top right square) is free of rooks.

Solution: (Bijective proof)

Let the eight rooks be $1, 2, \ldots, 8$. Consider a derangement $i_1 i_2 \cdots i_8$ on the 8-set $\{1, 2, \ldots, 8\}$ where $i_k \neq k$ for each $k = 1, 2, \ldots, 8$. To this derangement, we associate a board position for 8 rooks as follows:

We place the rook k in the j-th line of the k-th column of the chess board, if $i_k = j$ for $k = 1, 2, \ldots, 8$. (For example, if the derangement is 87654321, then we place the rook 1 in the 8th line of the first column, the rook 2 in the 7th line of the second column, etc.) Since $i_k \neq k$, no rook will occupy the main diagonal and no rook will attack another (8 rooks are placed in different lines and different columns).

Conversely, consider a position of 8 rooks in the board according to the condition imposed in the example. We construct a derangement $i_1 i_2 \cdots i_8$ from the given position of rooks as follows: i_k is equal to the integer j if the rook in the kth column occupies the jth line. Since no two rooks are in the same column, i_k is well defined. Then $i_1 i_2 \cdots i_8$ is a derangement, since $i_k \neq k$ (because the second main diagonal is free of rooks) and $i_1 i_2 \cdots i_8$ is a permutation of $1, 2, \ldots, 8$ (because no two rooks are in the same line.)

Thus we have established a bijection between the derangements on the 8-set $\{1, 2, \ldots, 8\}$ and the set of possible positions of 8 rooks on the board such that no rook can attack another with the second main diagonal being free of rooks. Therefore, by Proposition 2.2.1, the number possible positions of the 8 rooks is the number of derangements of an 8-set which is $d_8 = 8!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + \frac{1}{8!}\right) = 14833$.

2.9.2 Application of Inclusion and Exclusion Principle to Elementary Number Theory

(See Chapter 3, for more information.)

The Euler ϕ-function $\phi(n)$: For a positive integer n, we define $\phi(n)$, called the Euler ϕ-function, as the number of positive integers $\leq n$ and relatively prime to the integer n. (Two positive integers p and q are *relatively prime*, denoted by $p \perp q$, if their only common positive divisor is unity.)

Example 2.9.8:
$\phi(1) = 1$, $\phi(2) = 1$,$\phi(3) = 2$.

$\phi(8) = 4$, because there are four positive integers ≤ 8 and relatively prime to 8. These are $1, 3, 5, 7$.

$\phi(9) = 6$, because there are six positive integers ≤ 8 and relatively prime to 8. These are $1, 2, 4, 5, 7, 8$.

$\phi(p) = p - 1$, for any prime number p, since any positive integer $< p$ is relatively prime to p.

We are interested in finding a formula for $\phi(n)$ in terms of n and its prime factors.

We recall the fundamental theorem of arithmetic, also called the unique factorization theorem. A positive integer $n > 1$ is a *prime number* if its only positive divisors are unity and itself.

Theorem 2.9.3 (Fundamental theorem of arithmetic):
Any positive integer $n > 1$ can be factored in a unique way as

$$n = p_1^{k_1} p_2^{k_2} \cdots p_m^{k_m}$$

where $p_1 < p_2 \cdots < p_m$ are prime numbers and $k_i > 0$ are integers for $i = 1, 2, \ldots, m$.

Floor function: For a real number x, $\lfloor x \rfloor$ (read "floor of x") denotes the greatest integer $\leq x$. Symbolically,

$$\lfloor x \rfloor = \max_{n, \text{an integer}} n \leq x.$$

Example 2.9.9 (Floor function):
For example, $\lfloor 5.99 \rfloor = \lfloor 5 \rfloor = 5$; $\lfloor -5.4 \rfloor = -6$. $\lfloor \frac{16}{3} = 5 \rfloor$.

Note that for positive integers n and k, $\lfloor \frac{n}{k} \rfloor$ is the quotient obtained by dividing n by k. In other words, $\lfloor \frac{n}{k} \rfloor$ is the number of integers in the set $\{1, 2, \ldots, n\}$ which are multiples of the integer k (that is divisible by k).

We now state and prove a formula for $\phi(n)$.

Theorem 2.9.4:
Let n be a positive integer. Then the number of positive integers $\leq n$ and relatively prime to n is

$$\phi(n) = n \prod_{\substack{p, \text{a prime number} \\ p|n}} \left(1 - \frac{1}{p}\right).$$

That is, the above product is taken over all prime divisors p of n.

Proof. Let $\{p_1, p_2, \ldots, p_m\}$ be the set of distinct prime divisors of the integer n. Consider the set $A = \{1, 2, \ldots, n\}$ For $i = 1, 2, \ldots, m$, denote by S_i the set of integers belonging to the set A which are divisible by the prime number p_i.

We observe that the number of positive integers $\leq n$ and relatively prime to n is the same as the number of integers in the set A which are not divisible by $p_1, p_2, \ldots, p_m$. (For if k is an integer with $1 \leq k \leq n$ and relatively prime to n then k is not divisible by any of the numbers p_i for $i = 1, 2, \ldots, m$. Conversely, if $k \in A$ is not divisible by any of the p_i's, then k is relatively prime to n.)

Hence we calculate the number of integers in the set A not divisible by any of the p_i's. By our definition of the sets S_i's, this is nothing but the number of elements of A belonging to none of the sets S_i for $i = 1, 2, \ldots, m$.

By Corollary 2.9.1.1, this number is

$$E(0) = n - A_1 + A_2 - \cdots + (-1)^m A_m$$

where

$$A_r = \sum_{1 \leq 1 < i_1 < i_2 < \cdots < i_r \leq m} |S_{i_1} \cap S_{i_2} \cap \cdots S_{i_r}|.$$

$|S_{i_1} \cap S_{i_2} \cap \cdots S_{i_r}|$ is the number of elements of A divisible by $p_{i_1}, p_{i_2}, \ldots, p_{i_r}$, that is, divisible by the product $p_{i_1} p_{i_2} \cdots p_{i_r}$ (since $p_{i_1}, p_{i_2}, \ldots, p_{i_r}$ are primes). But this number is the floor function $\lfloor \frac{n}{p_{i_1} p_{i_2} \cdots p_{i_r}} \rfloor = \frac{n}{p_{i_1} p_{i_2} \cdots p_{i_r}}$ (since the product $p_{i_1} p_{i_2} \cdots p_{i_r}$ divides n, see Example 2.9.9). Therefore,

$$A_r = \sum_{1 \leq i_1 < i_2 < \cdots < i_r \leq m} \frac{n}{p_{i_1} p_{i_2} \cdots p_{i_r}}$$

Substituting A_r in $E(0) = n - A_1 + A_2 - \cdots + (-1)^m A_m$ we get the number of required integers is

$$\begin{aligned} & E(0) \\ = & n - \sum_{1 \leq i_1 \leq m} \frac{n}{p_{i_1}} + \sum_{1 \leq i_1 < i_2 \leq m} \frac{n}{p_{i_1} p_{i_2}} - \cdots (-1)^m \frac{n}{p_1 p_2 \cdots p_m} \\ = & n \left(1 - \sum_{1 \leq i_1 \leq m} \frac{1}{p_{i_1}} + \sum_{1 \leq i_1 < i_2 \leq m} \frac{1}{p_{i_1} p_{i_2}} - \cdots (-1)^m \frac{1}{p_1 p_2 \cdots p_m} \right) \\ = & n \left(1 - \frac{1}{p_1} \right) \left(1 - \frac{1}{p_2} \right) \cdots \left(1 - \frac{1}{p_m} \right) \\ = & n \prod_{\substack{p \\ \text{p is a prime divisor of } n}} \left(1 - \frac{1}{p} \right) \end{aligned}$$

■

Example 2.9.10:
Find $\phi(30)$.

Solution: The prime divisors of 30 are $2, 3, 5$. Hence by Theorem 2.9.4, $\phi(30) = 30(1 - \frac{1}{2})(1 - \frac{1}{3})(1 - \frac{1}{5}) = 30 \times \frac{1}{2} \times \frac{2}{3} \times \frac{4}{5} = 8.$

The formula of the Theorem 2.9.4 can be written elegantly using the *Möbius function* $\mu(n)$. For a positive integer n, the function $\mu(n)$ is defined as

$$\mu(n) = \begin{cases} 1 & \text{if } n = 1 \\ 0 & \text{if } n \text{ is divisible by a square of a prime number} \\ (-1)^k & \text{if } n = p_1 p_2 \cdots p_k \text{ where } p_i\text{'s are distinct primes} \end{cases}$$

For example, $\mu(16) = 0$ since 2^2 divides 16. $\mu(30) = \mu(2 \times 3 \times 5) = (-1)^3 = -1$.

Theorem 2.9.5 (Euler ϕ function using Möbius function μ): For a positive integer n,

$$\phi(n) = n \sum_{d} \frac{\mu(d)}{d}$$

where the sum is taken over all positive divisors d of n.

Proof. Consider the sum

$$n \sum_{d,\text{ a divisor of } n} \frac{\mu(d)}{d} \tag{2.3}$$

By the *fundamental theorem of arithmetic* 2.9.3, the integer n can be written as the product

$$n = p_1^{k_1} p_2^{k_2} \cdots p_m^{k_m}$$

where $p_1 < p_2 \cdots < p_m$ are prime numbers and $k_i > 0$ are integers for $i = 1, 2, \ldots, m$. Therefore any positive divisors d of n may be written in the form $d = p_1^{q_1} p_2^{q_2} \cdots p_m^{q_m}$ with $0 \leq q_i \leq k_i$. By the definition of the Möbius function $\mu(d)$, $\mu(d) = 0$ if there is an i with $q_i \geq 2$ (since the prime square p_i^2 divides d.) Hence such divisors d contribute zero to the sum (2.3). Therefore only the divisors d of the form $d = p_1^{q_1} p_2^{q_2} \cdots p_m^{q_m}$ with $q_i = 0$ or 1 contribute to the sum (2.3), that is, only the divisors d which are either 1 or the product of distinct prime numbers are counted in the sum (2.3). By definition of $\mu(d)$,

$$\mu(d) = \begin{cases} 1 & \text{if } d \text{ is a product of even number of some distinct } p_i\text{'s} \\ -1 & \text{if } d \text{ is a product of odd number of some distinct } p_i\text{'s} \end{cases}$$

Hence the sum 2.3 can be written as

$$\begin{aligned}
& n \sum_{d, \text{ a divisor of } n} \frac{\mu(d)}{d} \\
= & \; n\left(\frac{\mu(1)}{1} + \sum_{d, \text{ a product of distinct primes } p_i\text{'s}} \frac{\mu(d)}{d}\right) \\
= & \; n\left(1 + \sum_{1\le i\le m} \frac{\mu(p_i)}{p_i} + \sum_{1\le i<j\le m} \frac{\mu(p_ip_j)}{p_ip_j} + \cdots + \frac{\mu(p_1p_2\cdots p_m)}{p_1p_2\cdots p_m}\right) \\
= & \; n\left(1 - \sum_{1\le i\le m} \frac{1}{p_i} + \sum_{1\le i\le m} \frac{1}{p_ip_j} - \cdots + (-1)^m \frac{1}{p_1p_2\cdots p_m}\right) \\
= & \; n\left(1-\frac{1}{p_1}\right)\left(1-\frac{1}{p_2}\right)\cdots\left(1-\frac{1}{p_m}\right) \\
= & \; n \prod_{\substack{p \\ \text{p is a prime divisor of } n}} \left(1-\frac{1}{p}\right) \\
= & \; \phi(n)
\end{aligned}$$

■

Sieve of Eratosthenes:
For a real number x, let us denote by $\pi(x)$ the number of prime numbers $\le x$. For example, $\pi(23) = 9$ because there are nine prime numbers, namely, $2, 3, 5, 7, 11, 13, 17, 19, 23$ are ≤ 23. Apparently, there is no closed formula for $\pi(x)$. However there is an asymptotic formula for $\pi(x)$ which says that

$$\lim_{x\to\infty} \frac{\pi(x)}{(x/\log_e x)} = 1.$$

The above equation is called the *Prime number theorem.* We first prove the following simple fact concerning a sufficient condition to test primality. A divisor d of an integer n is a prime divisor if d is prime.

Fact 2.9.5.1 (A sufficient condition to test primality):
Let $n \geq 2$ be a positive integer. If n has no prime divisors $\leq \sqrt{n}$, then n is a prime number.

Proof. If n were not a prime, $n = ab$ for integers $a \geq 2$, $b \geq 2$. Clearly, both a and b cannot be strictly greater than $\sqrt{n}$, for otherwise their product will be $> n$. So at least one of a and b, say a must be $\leq \sqrt{n}$. But then, any prime factor of a, which is also a prime factor of n, must be $\leq \sqrt{n}$. ■

Remark 2.9.2 (Algorithm: Sieve of Eratosthenes):
The above Fact 2.9.5.1 gives us a method (or algorithm) to find all prime numbers $\leq n$ if we know the prime numbers $\leq \sqrt{n}$.

Algorithm: Write down the list of all integers from 2 to n. Let $p_1 = 2, p_2 = 5, \ldots, p_m$ be the known prime numbers $\leq \sqrt{n}$. Now strike out the numbers of the list which are divisible by p_1, then strike out the numbers divisible by p_2, and so on, and finally strike out the numbers divisible by p_m. By the Fact 2.9.5.1, the numbers remaining in the list are all prime numbers $> \sqrt{n}$ and $\leq n$. This method of generating prime numbers is known as the "sieve of Eratosthenes."

Theorem 2.9.6:
If n is a positive integer then

$$\pi(n) - \pi(\sqrt{n}) = -1 + n \sum_{d} \frac{\mu(d)}{d}$$

where the sum is taken over all divisors d of n.

Proof. Let $p_1, p_2, \ldots, p_k$ be the prime numbers $\leq \sqrt{n}$. Let $S = \{2, 3, \ldots, n\}$. Note that $|S| = n - 1$. For $i = 1, 2, \ldots, p_m$, let S_i be the subset consisting of all integers in S which are divisible by the prime p_i. By Remark 2.9.2, the integers of the set S which do not belong to any of the sets S_i for $i = 1, 2, \ldots, n$ are prime numbers $> \sqrt{n}$ and $\leq n$. Using the notation $\pi(x)$, the number of integers which are in none of the sets S_i is $\pi(n) - \pi(\sqrt{n})$.

But this number can be computed in a second way using the principle of inclusion and exclusion. By Corollary 2.9.1.1, the number of elements in none of the sets is

$$(n-1) - A_1 + A_2 - \cdots + (-1)^m A_m \tag{2.4}$$

where $A_r = \sum_{1 \leq 1 < i_1 < i_2 < \cdots < i_r \leq m} |S_{i_1} \cap S_{i_2} \cap \cdots S_{i_r}|$. By the definition of S_i, $|S_{i_1} \cap S_{i_2} \cap \cdots S_{i_r}|$ is the number of integers in the set A which are divisible by $p_{i_1}, p_{i_2}, \ldots, p_{i_r}$, that is divisible by the product $p_{i_1} p_{i_2} \cdots p_{i_r}$ (since p_{i_j}'s are primes) (refer to Example 2.9.9). This number is the quotient $\frac{n}{p_{i_1} p_{i_2} \cdots p_{i_r}}$. Substituting the values of A_r $(r = 1, 2, \ldots, m)$ in Equation 2.4, we get

$$\begin{aligned}
\pi(n) - \pi(\sqrt{n}) &= -1 + n - \sum_{1 \leq i \leq m} \frac{n}{p_i} + \sum_{1 \leq i < j \leq m} \frac{n}{p_i p_j} - \cdots \\
&\quad +(-1)^m \frac{n}{p_1 p_2 \cdots p_m} \\
&= -1 + n \prod_{\substack{p \\ \text{a prime divisor of } n}} \left(1 - \frac{1}{p}\right) \\
&= -1 + n \sum_{d} \frac{\mu(d)}{d} \quad \text{by Theorem 2.9.5}
\end{aligned}$$

■

2.9.3 Applications to Permanents

The *permanent* of a square matrix is the sum of the products of the entries of the matrix taken only one entry from each row and each column. Thus the permanent of a square matrix has the same expansion as the *determinant* of the matrix except that each term of the permanent is assigned a plus sign while the determinant alternates between a plus sign and minus sign. We denote the permanent of a matrix M by $\text{per}(M)$.

Hence permanents can be expanded similar to the *Laplace expansion for determinants*. The following examples illustrate the concept.

Example 2.9.11:
The permanent of $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is $ad + bc$ whereas its determinant is $ad - bc$.

Example 2.9.12 (Determinants and permanents):
The permanent of the matrix (we expand similar to the Laplace expansion of determinants)

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

is $a(ei + fh) + b(di + gf) + c(dh + ge) = aei + afh + bgf + cdh + bdi + cge$. Its determinant is $a(ei - fh) - b(di - gf) + c(dh - ge) = aei - afh + bgf - bdi + cdh - cge$.

In a formal manner, the permanent of the square matrix

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

is defined as the number

$$\sum_{p} a_{1p_1} a_{2p_2} \cdots a_{np_n}$$

where the sum is taken over all possible permutations $p = (p_1, p_2, \dots, p_n)$ on the set $\{1, 2, \dots, n\}$. Hence the permanent of an $n \times n$ square matrix possesses exactly $n!$ terms (since there are $n!$ permutations possible on an n-set) and is the sum of the products of all possible permutations of the set $[n]$. Note that the action of the permutation p on the element i is denoted conveniently by p_i instead of $p(i)$.

Example 2.9.13:
Find the permanent of the 3×3 matrix

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Solution: Each term of the permanent is the sum of the product taken only one element from each row and each column. Hence each term of the permanent is $1 \times 1 \times 1 = 1$ and the permanent has exactly $3!$ terms. Hence the value is $3! = 6$.

More generally, the permanent of an $n \times n$ matrix with each entry 1 is $n!$.

Example 2.9.14:
Find the permanent of the $n \times n$ matrix (a_{ij}) with each entry in the main diagonal is 0 and all other entries are 1.

Solution: By definition the permanent is

$$\sum_p a_{1p_1} a_{2p_2} \cdots a_{np_n}$$

where $p = (p_1, p_2, \ldots, p_n)$ is a permutation of $1, 2, \ldots, n$. Since the diagonal entries a_{ii} are zero, for permutations p for which $p(i) = p_i = i$ for some i, we have $a_{1p_1} a_{2p_2} \cdots a_{np_n} = 0$. Therefore only the permutations p for which $p_i \neq i$ for all $i = 1, 2, \ldots, n$ contribute in the evaluation of the permanent. But by definition such permutations are called derangements. Since $a_{ij} = 1$ if $i \neq j$, we have that the permanent is equal to

$$\sum_p \overbrace{1 \times 1 \cdots \times 1}^{n\, ones}$$

where the sum is taken over all derangements of the set $\{1, 2 \ldots, n\}$.

Since each term of the sum is 1, the permanent is the number of derangements on the n-set $\{1, 2 \ldots, n\}$ which is $d_n = n! \left(1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + (-1)^n \frac{1}{n!}\right)$. (See Theorem 2.9.2.)

Property 2.9.1 (Permanents and determinants):
The permanents satisfy the following properties:

1. If A is an $n \times n$ square matrix with real entries, then $\operatorname{per}(A) = \operatorname{per}(A^t)$ where A^t is the transpose of A. (The transpose of A is the matrix obtained by interchanging the rows and columns of A.)

2. If A, B are two square matrices of order $n \times n$, then in general

$$\operatorname{per}(A \times B) \neq \operatorname{per}(A) \times \operatorname{per}(B)$$

(unlike in determinants).

3. There is no known reasonable way (technically a "polynomial time") to calculate the permanent of a matrix (unlike determinants).

4. The addition of a constant multiple of a row (column) to another row (column) does modify the value of the permanent (unlike the determinant).

Example 2.9.15 (Permanents and product of the row sums of a square matrix):
Consider the 2×2 matrix $\begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}$. The product of the row sums of the above matrix is $(m_{11} + m_{12})(m_{21} + m_{22}) = m_{11}m_{21} + m_{11}m_{22} + m_{12}m_{21} + m_{12}m_{22}$

The above product of the row sums can be viewed as

$$\sum_{j} m_{1j_1} m_{2j_2}$$

where the sum is taken over all possible functions j from the set $\{1, 2\}$ to itself. Note that the value of the function j on 1 is denoted by j_1 instead of $j(1)$. Similarly for j_2. The ($2^2 = 4$) different functions from the set $\{1, 2\}$ to itself are $\begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 \\ 2 & 2 \end{pmatrix}$.

The four functions will be written simply as $(1, 1)$;$(1, 2)$;$(2, 1)$;$(2, 2)$.

More generally, the product of the row sums of an $n \times n$ matrix, $M = (m_{ij})$ is

$$\sum_j m_{1j_1} m_{2j_2} \cdots m_{nj_n},$$

where the sum is taken over all possible functions $j = (j_1, j_2, \ldots, j_n)$ from the set $\{1, 2, \ldots, n\}$ to itself. By Theorem 2.2.2, the number of terms in the above sum is n^n.

If we restrict the functions $j = (j_1, j_2, \ldots, j_n)$ from the set $\{1, 2, \ldots, n\}$ to itself, that is, to bijections (permutations) we get, by definition, the value of the permanent of the matrix (m_{ij}).

Now consider the matrix, M_r, obtained by replacing the entries of some r columns, say the columns $1, 2, \ldots, r$ of the matrix M, by zeros. That is,

$$M_r = \begin{pmatrix} 0 & 0 & \ldots & 0 & \ldots & m_{1,r+1} & \ldots & m_{1n} \\ 0 & 0 & \ldots & 0 & \ldots & m_{2,r+1} & \ldots & m_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & \ldots & m_{n,r+1} & \ldots & m_{nn} \end{pmatrix}$$

The sum of the product of the row sums of the matrix M_r is denoted by $s(M_r)$ where

$$s(M_r) = \prod_{i=1}^{n} (m_{i,r+1} + m_{i,r+2} + \cdots + m_{in}).$$

By multiplying term by term the above product of the sums, we obtain

$$s(M_r) = \sum_j m_{1j_1} m_{2,j_2} \cdots m_{n,j_n}$$

where the sum is taken over all functions $j = (j_1, j_2, \ldots, j_n)$ from the n-set $\{1, 2, \ldots, n\}$ to $\{r+1, r+2, \ldots, n\}$, that is, the sum is taken over all ordered n-tuples $(j_1, j_2, \ldots, j_n)$ not containing the components $r+1, r+2, \ldots, n$.

With Example 2.9.15 in mind, we are going to derive an expression for the permanent of a square matrix using the *weighted version of the exclusion and inclusion principle* which we shall state and prove now.

Weighted version of the inclusion and exclusion principle:

THEOREM 2.9.7:
Consider an n-set S. Let each element $s \in S$ be associated to a real number $w(s)$, called the weight of the element s. Consider m subsets $S_1, S_2, \ldots, S_m$ of the set S. Let $w(i_1, i_2, \ldots, i_r)$ be the sum of the weights of all the elements belonging to the intersection $S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_r}$. Then the sum of the weights of the elements of the set S which are in exactly k of the subsets $S_1, S_2, \ldots, S_m$ is

$$E(k) = A_k - \binom{k+1}{k} A_{k+1} + \binom{k+2}{k} A_{k+2} - \cdots + (-1)^{m-k} \binom{m}{k} A_m \tag{2.5}$$

where $A_0 =$ sum of the weights of all the elements of S and $A_r = \sum w(i_1, i_2, \ldots, i_r)$ where the sum is taken over all r-subsets $\{\imath_1, i_2, \ldots, i_r\}$ of $\{1, 2, \ldots, n\}$ with $1 \leq < i_1 < i_2 < \cdots < i_r \leq m$.

(Note that if $w(s) = 1$ for all $s \in S$, we get our Theorem 2.9.1.)

Proof. Consider an element $s \in S$ belonging to *exactly* p of the subsets $S_1, S_2, \ldots, S_m$. We shall prove that the element s contributes *zero* to the right-hand side of the formula 2.5 for E_k if either $p < k$ or $p > k$ and the element s contributes exactly $w(s)$ to the right-hand side of the formula 2.5 for E_k if $p = k$.

The proof is essentially the same as the proof of the Theorem 2.9.1 with the following slight modification which takes into account the weight of $s \in S$.

Case 1 remains the same.

In Case 2, multiply the expression

$$\binom{p}{k}\binom{k}{k} - \binom{k+1}{k}\binom{p}{k+1} + \binom{k+2}{k}\binom{p}{k+2} - \cdots$$
$$+(-1)^{p-k}\binom{p}{k}\binom{p}{p}$$

by $w(s)$ and proceed.

In Case 3, replace "1" by $w(s)$. That is, read the first sentence of Case 3 as "the contribution of s to the first term $A_k = A_p$ of the right-hand side of the formula is clearly $w(s)$ and proceed. ∎

By setting $k = 0$ in the above theorem we get the following corollary.

Corollary 2.9.7.1:
The sum of the weights of the elements of the set S which are in none of the k subsets $S_1, S_2, \dots, S_m$ is

$$E(0) = A_0 - A_1 + A_2 - \cdots + (-1)^m A_m$$

where

$$A_r = \sum_{1 \le i_1 < i_2 < \cdots < i_r \le m} w(i_1, i_2, \dots, i_r)$$

with $w(i_1, i_2, \dots, i_r) = \sum_{s \in S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_r}} w(s)$.

Theorem 2.9.8:
Consider an $n \times n$ matrix $M = (m_{ij}.)$ Let M_r be a matrix obtained from the matrix M by replacing some r columns of M by zeros. Let $s(M_r)$ denote the product of the row sums of the matrix M_r and let s_r be the sum of the $s(M_r)$ taken over all possible choices of M_r. That is $s_r = \sum_{M_r} s(M_r)$. We then have

$$\text{per(M)} = s_0 - s_1 + s_2 - \cdots (-1)^{n-1} s_{n-1}.$$

Proof. We shall apply the weighted version of the principle of inclusion and exclusion (see Corollary 2.9.7.1) to prove the expression.

To do this, let S be the set of all functions $j = (j_1, j_2, \dots, j_n)$ from the n-set $\{1, 2, \dots, n\}$ onto itself. (Note that we denote $j(1)$ by j_1, $j(2)$ by j_2, etc.) Hence a function from the n-set $\{1, 2, \dots, n\}$ onto itself is nothing but an ordered n-tuple of elements from $\{1, 2, \dots, n\}$. (See the first interpretation of functions in the beginning of the chapter.) To each element $j = (j_1, j_2, \dots, j_n)$ of the set S, we associate a weight $w(j) = m_{1j_1} m_{2j_2} \cdots m_{nj_n}$.

Then per(M), by definition, is the sum of the weights of all the elements $w(j)$, with $j \in S$ and j is a permutation on the set $\{1, 2, \dots, n\}$. Define n subsets $S_1, S_2, \dots, S_n$ as follows. For $r = 1, 2, \dots, n$, let S_r be the subset of S consisting of all ordered n-tuples $(j_1, j_2, \dots, j_n)$ *not* containing the integer r as its component, that is, $j_k \neq r$ for all $k = 1, 2, \dots, n$.

Now let us compute A_r for $r = 0, 1, \dots, n$ where

$$A_r = \sum_{1 \leq j_1 < j_2 < \cdots \leq n} \left(\sum_{s \in S_{j_1} \cap S_{j_2} \cdots \cap S_{j_r}} w(s) \right).$$

But by the definition of S_i's, an ordered n-tuple $s \in S_{j_1} \cap \S_{j_2} \cdots \cap S_{j_r}$ does *not* contain the components $j_1, j_2, \dots, j_r$. Hence $\sum_{s \in S_{j_1} \cap S_{j_2} \cdots \cap S_{j_r}} w(s)$ is the product of the row sums of the coefficients of the matrix M_r obtained from the matrix M by replacing the columns $j_1, j_2, \dots, j_r$ by the zero entries which is $s(M_r)$ (by Example 2.9.15). Hence $A_r = \sum_{1 \leq j_1 < j_2 < \cdots \leq n} s(M_r)$ where the sum is taken over all possible choices of M_r. But this sum is equal to s_r.

By Corollary 2.9.7.1, the sum of the weights of the elements of S belonging to none of the sets $S_1, S_2, \dots, S_n$ is

$$\begin{aligned} E(0) &= A_0 - A_1 + A_2 - \cdots + (-1)^n A_n \\ &= s_0 - s_1 + s_2 - \cdots + (-1)^n s_n \\ &= s_0 - s_1 + s_2 - \cdots (-1)^{n-1} s_{n-1}. \\ &\quad (\quad \text{since } M_n \text{ is the zero matrix, } s_n = 0) \end{aligned}$$

Hence the formula. ■

We now prove two corollaries.

Corollary 2.9.8.1:
For all non-negative integers n we have the equality

$$n! = \sum_{r=0}^{n-1} (-1)^r \binom{n}{r} (n-r)^n$$

Proof. Consider the $n \times n$ matrix $M = (m_{ij}$ where each entry m_{ij} is 1. Then by Example 2.9.3, the permanent of the matrix M is $n!$.

We shall now compute the permanent of the matrix M in a second way using the formula of Theorem 2.9.8

$$\text{per(M)} = s_0 - s_1 + s_2 - \cdots (-1)^{n-1} s_{n-1} \tag{2.6}$$

For $r = 0, 1, \ldots, n-1$, let M_r be the matrix obtained by replacing the columns $j_1', j_2', \ldots, j_r'$ of the matrix M by the zero entries. Then $s(M_r)$ is the product of the row sums of the matrix M_r. Since the sum of each row of M_r is $(n-r)$, we have the product of the row sums of M_r is $(n-r)^n$, that is, $s(M_r) = (n-r)^n$. Now by definition, $s_r = \sum_{M_r} s(M_r)$ where the sum is taken over all possible choices of M_r. But r columns can be chosen from n columns in $\binom{n}{r}$ ways. Hence $s_r = \sum_{M_r} s(M_r) = \binom{n}{r}(n-r)^n$. Substituting the values of s_r in the right-hand side of Equation (2.6), we get,

$$\begin{aligned}
\text{per}(M) &= s_0 - s_1 + s_2 - \cdots (-1)^{n-1} s_{n-1} \\
&= \binom{n}{0}(n-0)^n - \binom{n}{1}(n-1)^n + \cdots \\
&\quad (-1)^{n-1}\binom{n}{n-1}(n-(n-1))^{(n-(n-1)} \\
&= \sum_{r=0}^{n-1}(-1)^r \binom{n}{r}(n-r)^n
\end{aligned}$$

Thus the corollary is proved. ∎

Corollary 2.9.8.2:
The number of derangements d_n on an n-set is

$$d_n = \sum_{r=0}^{n-1}(-1)^r \binom{n}{r}(n-r)^r (n-r-1)^{n-r}.$$

Proof. Consider the matrix $M = (m_{ij})$ with each entry in the main diagonal m_{ii} zero and all other entries 1. By Example 2.9.14, the permanent of the matrix M is d_n.

Now let us calculate the permanent of the matrix M in a different way using the formula of Theorem 2.9.8

$$\text{per(M)} = s_0 - s_1 + s_2 - \cdots (-1)^{n-1} s_{n-1} \qquad (2.7)$$

For $r = 0, 1, \ldots, n-1$, let M_r be the matrix obtained by replacing the columns $j'_1, j'_2, \ldots, j'_r$ of the matrix M by the zero entries. Then $s(M_r)$ is the product of the row sums of the matrix M_r. The row sum $s(l_i)$ of the entries in the ith line of the matrix M_r is given by

$$s(l_i) = \begin{cases} n-r & \text{if } i \in \{ j'_1, j'_2, \ldots, j'_r \} \\ n-r-1 & \text{otherwise} \end{cases}$$

In other words, the row sum of M_r is $n-r$ for the rows $j'_1, j'_2, \ldots, j'_r$ and it is equal to $n-r-1$ for all other $n-r$ rows. Hence the product of the row sums of the matrix M_r is $s(M_r) = (n-r)^r(n-r-1)^{n-r}$. Therefore,

$$\begin{aligned} s_r &= \sum_{M_r} s(M_r) = \sum_{M_r} (n-r)^r (n-r-1)^{n-r} \\ &= \binom{n}{r} (n-r)^r (n-r-1)^{n-r} \end{aligned}$$

(since there are $\binom{n}{r}$ possible choices of r columns out of n available columns). Substituting the values of s_r to the right-hand side of Equation (2.7), we obtain

$$\begin{aligned} \text{per}(M) &= s_0 - s_1 + s_2 - \cdots (-1)^{n-1} s_{n-1} \\ &= \sum_{r=0}^{n-1} (-1)^r s_r \\ &= \sum_{r=0}^{n-1} (-1)^r \binom{n}{r} (n-r)^r (n-r-1)^{n-r} \end{aligned}$$

This proves the corollary. ■

2.10 Generating Functions and Recurrence Relations

Consider an infinite sequence of numbers $(a_i)_{i \geq 0} = (a_0, a_1, \ldots)$. Then the infinite sum in the "indeterminate" z (called the power series)

$$g(z) = a_0 + a_1 z + a_2 z^2 + \cdots = \sum_{i=0}^{\infty} a_i z^i$$

is called an *ordinary generating function* or simply the *generating function* of the sequence (a_i). The function $g(z)$ "encapsulates" the entire infinite sequence (a_i).

A note on the symbol "x":
The symbol "x" is used very often in mathematics. But the meaning of "x" varies according to the subtle context. In elementary algebra "x" denotes an unknown number. For example, if we want to solve the equation $x^2 - x - 1 = 0$, then "x" here represents an *"unknown."*

The same equation $x^2-x-1 = 0$ can be solved using the graphical method: In this method, we draw the graph of the equation $y = x^2$ and find its intersection with the graph of the straight line $y - x - 1 = 0$. In this graphical method, "x" is no longer an unknown but is a *variable*.

Now there is another generalization of the symbol "x" in which "x" represents an *indeterminate*. It is simply a symbol with no condition imposed on it, except that the symbol should satisfy the fundamental laws of algebra. This means that with "x" and with other numbers or other indeterminates, we can perform the operation of addition, subtraction, multiplication and perhaps division and the operations of addition and multiplication are commutative and multiplication is distributive over addition. The unknowns and the variables also obey these laws but the idea of an indeterminate is more general because we do not assume that an indeterminate does belong to a number system (see the book, *The Skeleton Key of Mathematics*, by D. E. Littlewood).

The following examples illustrate the concept of a generating function.

Example 2.10.1 (Geometric series):
Consider the sequence $(a_i)_{i=0}^{\infty} = (1, 1, 1, \ldots)$. Then the generating

function of the sequence (a_i) is

$$g(z) = 1 + z + z^2 + z^3 + \cdots .$$

We know that the right-hand side of the above equation is a geometric series which converges to $\frac{1}{1-z}$ for all z with the property that $|z| < 1$. Hence the function $\frac{1}{1-z}$ "encapsulates" the whole sequence (a_i).

Example 2.10.2 (Binomial theorem):
The generating function of the sequence (a_i) where $a_0 = 1$,$a_1 = r$,$a_2 = r(r-1)/2!$,$a_3 = r(r-1)(r-2)/3!$, etc., is

$$1 + rz + r(r-1)z^2/2! + r(r-1)(r-2)z^3/3! + \cdots$$

From the binomial Theorem 2.4.2, the above series becomes an infinite series if r is not a non-negative integer and it converges to $(1+z)^r$ for all z satisfying the inequality $|z| < 1$.

If r is a non-negative integer, then the series becomes a finite series because $a_{r+1} = a_{r+2} = \cdots = 0$ and its value is again $(1+z)^r$ for all values of z.

The behavior of the geometric series of Example 2.10.1 which converges for all z with $|z| < 1$ and diverges for all z with $|z| > 1$ is typical. In fact, every power series $a_0 + a_1 z + a_2 z^2 + \cdots$ converges inside a circle $|z| < r$, called *the circle of convergence*, and diverges outside this circle except in the following two cases. The series converges for all values of z or it converges only for $z = 0$.

We state the following theorem due to Abel without proof. (For a proof, see [8]).

Theorem 2.10.1:
Consider the power series $a_0 + a_1 z + a_2 z^2 + \cdots$ where the coefficients a_i's are real or complex numbers. Then there exists a real number r, called the radius of convergence, such that $0 \leq r \leq \infty$, satisfying the following properties:

1. The series converges absolutely for all z satisfying the inequality $|z| < r$. If s is a real number such that $0 \leq s < r$ then the

series converges uniformly for all z with $|z| \leq s$. 2. For all z with $|z| > r$, the series is divergent.

3. For all z with $|z| < r$, the sum of the series is an analytic function. We can differentiate term by term the series $a_0 + a_1 z + a_2 z^2 + \cdots$ and the derived series has the same radius of convergence as the original series.

The circle $\{ z : |z| = r \}$ is called the circle of convergence of the series $a_0 + a_1 z + a_2 z^2 + \cdots$. Nothing can be said about the convergence of the series for values of z on the circle of convergence. The following due to Hadamard, gives a formula for the radius of convergence r.

$$1/r = \limsup_{n \to \infty} |a_n|^{1/n}.$$

If the series is convergent only for $z = 0$, then it may be possible to obtain a convergent series for the sequence $(a_i/i!)$. This motivates the following definition.

Exponential generating function

Consider the infinite sequence $(a_i)_{i=0}^{\infty}$. Then the infinite sum

$$g(z) = a_0 z + a_1 z/1! + a_2 z^2/2! + a_3 z^3/3! + \cdots$$

is called the exponential generating function of the sequence (a_i).

Example 2.10.3 (Exponential generating function):
The exponential generating function of the sequence $(a_i) = (1, 1, \ldots)$ is

$$1 + z/\,1! + z^2/2! + z^3/3! + \cdots$$

which is the familiar exponential function e^z. The series converges for all values of z.

Example 2.10.4 (Bernoulli numbers):
The exponential generating function of the Bernoulli numbers $(B_i)_{i=0}^{\infty}$ is the function $\frac{z}{e^z - 1}$.

Let us expand $\frac{z}{e^z-1}$. as an infinite series. $e^z = 1 + z + z^2/2! + z^3/3! + \cdots$. Hence

$$\frac{z}{e^z-1} = \frac{1}{1+z/2!+z^2/3!+\cdots} = \left(1+z/2!+z^2/3!+\cdots\right)^{-1}.$$

But by binomial Theorem 2.4.2, $(1+z)^{-1} = 1 - z + z^2 - z^3 + \cdots$. Expanding $(1+z/2!+z^2/3!+\cdots)^{-1}$ with the help of the binomial theorem, we get

$$\frac{z}{e^z-1} = 1 - \frac{1}{2}z + \frac{1}{12}z^2 + \cdots = \sum_{i=0}^{\infty} \frac{B_i z^i}{i!}.$$

Hence $B_0 = 1$, $B_1 = -1/2$, $B_2 = 1/12$ etc. The numbers B_i are known as Bernoulli numbers.

Remark 2.10.1 (On the convergence of generating functions): When we deal with generating functions, we need not worry about the convergence of the series, because we are only examining possible approaches to the solution of some problem (see Knuth [1].) When a solution is found by some means, it may be possible to justify the solution by other means (e.g., proof by induction).

Addition and scalar multiplication of power series

We add two power series by adding their corresponding coefficients.

More formally, if $g(z) = \sum_{i=0}^{\infty} a_i z^i$ is the generating function of the sequence (a_i) and $h(z) = \sum_{i=0}^{\infty} b_i z^i$ is the generating function of the sequence (b_i), then $g(z)+h(z) = \sum_{i=0}^{\infty} c_i z^i$ where $c_i = a_i + b_i$ for all $i = 0, 1, \ldots$.

For a constant c, we define $cg(z) = \sum_{i=0}^{\infty}(ca_i)z^i$.

Multiplication of two power series

To multiply two generating functions, we multiply term by term the symbols formally, use the relation $z^r \times z^s = z^{r+s}$, and collect like terms.

In a formal manner, if $g(z) = \sum_{i=0}^{\infty} a_i z^i$ is the generating function of the sequence (a_i) and $h(z) = \sum_{i=0}^{\infty} b_i z^i$ is the generating function of the sequence (b_i), then

$$\begin{aligned} g(z)h(z) &= (a_0 + a_1 z + a_2 z^2 + \cdots)(b_0 + b_1 z + b_2 z^2 \cdots) \\ &= (a_0 + b_0) + (a_0 b_1 + a_1 b_0) z \\ &\quad +(a_0 b_2 + a_1 b_1 + a_2 b_0) z^2 + \cdots \\ &= c_0 + c_1 z + c_2 z^2 + \cdots \end{aligned}$$

where $c_k = a_k b_0 + a_{k-1} b_1 + a_{k-2} b_2 + \cdots + a_0 b_k = \sum_{i=0}^{k} a_{k-i} b_i$ for all $k = 0, 1, \ldots$.

2.10.1 Solving Recurrence Relations Using Generating Function Techniques

Gopala, Hemachandra, Fibonacci sequence:
Consider the sequence of integers $(f_n)_{n=0}^{\infty} = (0, 1, 1, 2, 3, 5, 8, 13, \ldots$ defined recursively as follows: The first two terms of the sequence are 0 and 1. Every other term is the sum of the two previous terms. In a formal manner,

$$f_n = \begin{cases} 0 & \text{if } n = 0(\text{ basis}) \\ 1 & \text{if } n = 1(\text{ basis}) \\ f_{n-1} + f_{n-2} \text{ if } n \geq 2(\text{ recurrence}) \end{cases}$$

The above sequence, called a Fibonacci sequence, was actually known much earlier to Hindu mathematicians Gopala and Hemachandra of the 12th century (see Knuth [1].)

Example 2.10.5 (Closed formula for Gopala-Hemachandra-Fibonacci sequence):
Our problem is to find a closed formula for the nth term of the sequence (f_n).

Solution:

We consider the generating function of the sequence

$$\begin{aligned} g(z) &= f_0 + f_1 z + f_2 z^2 + f_3 z^3 + \cdots \\ &= z + z^2 + 2z^3 + 3z^4 \cdots \end{aligned}$$

Our aim, if possible, is to find a "closed formula" for $g(z)$ like our Example 2.10.1 of the geometric series. To this end, we shall use the relation $f_n - f_{n-1} - f_{n-2} = 0$ for all n with $n \geq 2$. Replacing n by $n+2$ we get $f_{n+2} - f_{n+1} - f_n = 0$ for all n with $n \geq 0$. Now

$$\begin{aligned} g(z) &= f_0 + f_1 z + f_2 z^2 + f_3 z^3 + f_4 z^4 \cdots \\ zg(z) &= \qquad f_0 z + f_1 z^2 + f_2 z^3 + \cdots \\ z^2 g(z) &= \qquad\qquad f_0 z^2 + f_1 z^3 + f_2 z^4 \end{aligned}$$

Subtracting vertically, we get,

$$(1-z-z^2)g(z) = f_0 + (f_1 - f_0)z + (f_2 - f_1 - f_0)z^2 + (f_3 - f_2 - f_1)z^3$$
$$+(f_4 - f_3 - f_2)z^4 + \cdots$$

But $f_0 = 0$ and $f_{n+2} - f_{n+1} - f_n = 0$ for all values of n with $n \geq 0$. Therefore all terms of the above series vanish except the second term. Hence, we have the closed expression for $g(z)$ (if it exists) of $(1 - z - z^2)g(z) = (f_1 - f_0)z = z$ or

$$g(z) = z/(1 - z - z^2).$$

We now split the rational expression $z/(1 - z - z^2)$ into partial fractions. Factoring the denominator (or solving the quadratic equation $1 - z - z^2 = 0$) we get

$$1 - z - z^2 = (1 - \phi z)(1 - \phi' z)$$

where the real number $\phi = \frac{\sqrt{5}+1}{2}$ is the "golden ratio" and $\phi' = 1 - \phi$. Now splitting $g(z)$ into partial fractions we get,

$$g(z) = z/(1 - z - z^2) = \frac{1}{\sqrt(5)}\left(\frac{1}{1 - \phi z} - \frac{1}{1 - \phi' z}\right).$$

By Example 2.10.1 of the geometric series,

$$\frac{1}{1 - z} = 1 + z + z^2 + z^3 + \cdots$$

Therefore,

$$g(z) = \frac{1}{\sqrt(5)}(1 + \phi z + \phi^2 z^2 + \cdots - 1 - \phi' z - \phi'^2 z^2 - \cdots$$

That is,

$$f_0+f_1z+f_2z^2+\cdots=\frac{1}{\sqrt{5}}(1+\phi z+\phi^2z^2+\cdots-1-\phi' z-\phi'^2z^2-\cdots)$$

Comparing the coefficient of z^n on both sides of the above equation, we get,

$$f_n=\frac{1}{\sqrt{5}}(\phi^n-\phi'^n).$$

This is the desired closed form of the n-th term of the sequence (f_n).

Remark 2.10.2:
The formula $f_n=\frac{1}{\sqrt{5}}(\phi^n-\phi'^n)$ can now be proved by induction on the parameter n. Note that to prove something by induction, we must know the result in advance! The generating function technique gives a possible approach to find a formula without worrying about the convergence of the generating function.

Example 2.10.6 (Application):
(See [6].) Show that the sum of the infinite numbers:
.1 + .01 + .002 + .0003 + .00005 + .000008 + .0000013 + $\cdots$
converges to a rational number.

Solution: Observe that the nonzero digits form the Fibonacci sequence $(1,1,2,3\dots)$. The generating function for the Fibonacci sequence is

$$g(z)=\sum_{n=0}^{\infty}f_nz^n=z/(1-z-z^2)$$

(see Example 2.10.5). Setting $n=1/10$, we get the desired sum is equal to 10/89.

Example 2.10.7:
Prove by induction that the nth term of the sequence (f_n) where $f_0=0$, $f_1=1$ (basis) and $f_n=f_{n-1}+f_{n-2}$ (recurrence) for all $n\geq 2$ is

$$f_n=\frac{1}{\sqrt{5}}(\phi^n-\phi'^n)$$

where the "golden ratio" $\phi = \frac{\sqrt{5}+1}{2}$ and $\phi' = 1 - \phi$ are the roots of the equation $x^2 - x - 1 = 0$.

Solution:

Induction basis: Setting $n = 0$ in $f_n = \frac{1}{\sqrt{5}}(\phi^n - \phi'^n)$ we get $f_0 = \frac{1}{\sqrt{5}}(\phi^0 - \phi'^0) = 0$, which is true. By setting $n = 1$, $f_1 \frac{1}{\sqrt{5}}(\phi^1 - \phi'^1) = 1$.

Induction hypothesis: Suppose $f_n = \frac{1}{\sqrt{(5)}}(\phi^n - \phi'^n)$ for all integers $\leq n$. We shall prove that $f_{n+1} = \frac{1}{\sqrt{5}}(\phi^{n+1} - \phi'^{n+1})$. By recurrence, we have

$$f_{n+1} = f_n + f_{n-1} \tag{2.8}$$

By the induction hypothesis, $f_n = \frac{1}{\sqrt{5}}(\phi^n - \phi'^n)$ and $f_{n-1} = \frac{1}{\sqrt{5}}(\phi^{n-1} - \phi'^{n-1})$. Substituting these in Equation 2.8, and using the fact that ϕ and ϕ' are the roots of the equation $1 + x = x^2$ we obtain

$$\begin{aligned}
f_{n+1} &= f_n + f_{n-1} \\
&= \frac{1}{\sqrt{5}}(\phi^n - \phi'^n) + \frac{1}{\sqrt{5}}(\phi^{n-1} - \phi'^{n-1}) \\
&= \frac{1}{\sqrt{5}}\left(\phi^{n-1}(1+\phi) - \phi' n - 1(1+\phi')\right) \\
&= \frac{1}{\sqrt{5}}\left(\phi^{n-1}\phi^2 - \phi' n - 1\phi'^2)\right) \\
&\quad \text{since } 1+\phi = \phi^2 \text{ and } 1+\phi' = 1+\phi'^2 \\
&= \frac{1}{\sqrt{5}}(\phi^{n+1} - \phi'^{n+1})
\end{aligned}$$

2.10.2 Catalan Numbers

Catalan numbers give the number of ways to parenthesize a string of n-symbols. A string of n-symbols on a set A is simply an ordered n-tuple of elements of A.

Example 2.10.8 (Matrix chain multiplication):
Consider n square matrices $M_1, M_2, \ldots, M_n$ of the same order. We know that the matrix multiplication is associative. The number

of ways to perform the multiplication $M_1 \times M_2 \times \cdots \times M_n$ is the same as the number of ways to parenthesize a string of n symbols.

Example 2.10.9 (Catalan numbers):
Consider a string of 3-letters abc. This can be parenthesized in the following two ways: $((ab)c)$ and $(a(bc))$.

Take a string of 4 letters, $abcd$. This string can be parenthesized in the following ways: $(((ab)c)d)$, $((ab)(cd))$, $((a((bc)d))$, $(a(b(cd)))$, $((a(bc))d)$. Hence the number of ways to parenthesize 4 letters is 5. In other words, by Example 2.10.8, the number of ways to multiply 4 square matrices of the same order is 5.

We are interested in finding a "closed formula" for the number of ways to parenthesize a string of n-symbols using the generating function techniques. The following example illustrates the method.

Example 2.10.10 (A closed formula for Catalan numbers):
Find a closed formula for the Catalan numbers with the help of the generating function technique.

Solution:

In Example 2.10.5, we had a recurrence relation at our disposal which indeed facilitated our job in finding a closed formula for f_n. In a similar manner, we first derive a recurrence relation involving the sequence of Catalan numbers $(c_i)_{i=1}^{\infty}$. For the basis of the recurrence, we easily see the following: $c_1 = 1$ (since there is only one way to parenthesize the symbol a which is (a).

Now for the recurrence relation involving c_i's. Consider a string of n symbols $a_1a_2a_3 \cdots a_n$ with $n \geq 2$. The number of ways to parenthesize the first string of length k, $a_1a_2 \cdots a_k$ where $(1 \leq k \leq n)$ is by definition c_k. The number of ways to parenthesize the remaining string of length $n-k$, $a_{k+1}a_{k+2} \cdots a_n$ is c_{n-k}. Hence by the product rule of Proposition 2.2.2, the number of ways to parenthesize together the first string of length k, $a_1a_2 \cdots a_k$ and then the following string of length $n-k$, $a_{k+1}a_{k+2} \cdots a_n$ is the product $c_k \times c_{n-k}$. Since the integer k can assume any value between 1 and n, we have by the sum rule 2.2.0.1 the following recurrence relation:

$$c_n = c_1c_{n-1} + c_2c_{n-2} + \cdots + c_{n-1}c_1 = \sum_{i=1}^{n-1} c_ic_{n-i} \text{ for } n \geq 2. \quad (2.9)$$

Thus we have derived a recurrence relation involving $c_i's$. Now we proceed as in Example 2.10.5 to find a closed form for c_n. The sequence (c_i) starts with c_1. To facilitate the computation, define the new sequence $(b_i)_{i=0}^{\infty}$ where $b_i = c_{i+1}$ for all $i = 0, 1, \ldots$. Hence the new sequence $(b_i) = (b_0, b_1, b_2, \ldots = (1, 1, 2, \ldots)$ where $b_0 = 1$ and the recurrence relation 2.9 written in terms of $b_i's$ using $c_i = b_{i-1}$ is

$$b_{n-1} = b_0b_{n-2} + b_1b_{n-3} + \cdots + b_{n-2}b_0 \text{ for } n \geq 2.$$

Substituting $n + 1$ for n we get,

$$b_n = b_0b_{n-1} + b_1b_{n-2} + \cdots + b_{n-1}b_0 \text{ for } n \geq 1. \quad (2.10)$$

We first set up the generating function $g(z)$ of the sequence (b_i) where

$$g(z) = b_0z + b_1z + b_2z^2 + \cdots = \sum_{i=0}^{\infty} b_iz^i.$$

Our first aim is to find a closed form for $g(z)$ using the recurrence relation 2.9.

$$\begin{aligned}
g(z) &= b_0 + b_1z + b_2z^2 + b_3z^3 + \cdots \\
g(z)^2 &= b_0b_0 + (b_0b_1 + b_1b_0)z + (b_0b_2 + b_1b_1 + b_2b_0)z^2 \\
&\quad +(b_0b_3 + b_1b_2 + b_2b_1 + b_3b_0)z^3 + \cdots \\
&= b_1 + b_2z + b_3z^2 + b_4z^3 + \cdots \\
&\quad \text{ by the Equation (2.10) and the fact that } b_1 = b_0. \\
zg(z)^2 &= b_1z + b_2z^2 + b_3z^3 + b_4z^4 \cdots \\
&= g(z) - 1.
\end{aligned}$$

Therefore, the generating function satisfies the relation $zg(z)^2 = g(z) - 1$. Solving this quadratic equation in "g(z)" we get the closed form

$$g(z) = \frac{1}{2z}\left(1 - \pm\sqrt{1-4z}\right).$$

If $g(z) = \frac{1}{2z}\left(1+\sqrt{1-4z}\right)$, then $1 = b_0 = g(0) \to \infty$, which is impossible. Hence

$$g(z) = \frac{1}{2z}\left(1-\sqrt{1-4z}\right).$$

Expanding $\sqrt{1-4z} = (1-4z)^{1/2}$ by the binomial Theorem 2.4.2, we get,

$$\begin{aligned}
g(z) &= \frac{1}{2z}\left(1-(1-4z)^{1/2}\right)\\
&= \frac{1}{2z} - \frac{1}{2z}\sum_{k\geq 0}\binom{\frac{1}{2}}{k}(-4z)^k \text{ with } |z| < 1/4\\
&= 2\left(\frac{1}{4z} - \frac{1}{4z}\sum_{k\geq 0}\binom{\frac{1}{2}}{k}(-4z)^k\right)\\
&= 2\left(\frac{1}{4z} + \sum_{k\geq 0}\binom{\frac{1}{2}}{k}(-4z)^{k-1}\right)\\
&= 2\left(\frac{1}{4z} - \frac{1}{4z} + \sum_{k\geq 1}\binom{\frac{1}{2}}{k}(-4z)^k\right)\\
&= 2\sum_{k\geq 1}\binom{\frac{1}{2}}{k}(-4z)^{k-1}\\
&= 2\sum_{n\geq 0}\binom{\frac{1}{2}}{n+1}(-4z)^n\\
&\ \text{by setting } k = n+1 \text{ and changing limit accordingly}\\
&= 2\sum_{n\geq 0}\frac{1}{2(n+1)}\binom{\frac{-1}{2}}{n}(-4z)^n \text{using } \binom{n}{r} = \frac{n}{r}\binom{n-1}{r-1}\\
&= \sum_{n\geq 0}\binom{\frac{-1}{2}}{n}\frac{(-4z)^n}{n+1}\\
&= \sum_{n\geq 0}\frac{(-1/2)(-1/2-1)(-1/2-2)\cdots(-1/2-n+1)}{n!}(-4)^n z^n/(n+1)\\
&= \sum_{n\geq 0}\frac{(-4\times -1/2)((-4)(-1/2-1))((-4)(-1/2-2))\cdots((-4)(-1/2-n+1))}{n!}z^n/(n+1)\\
&= \sum_{n\geq 0}\frac{2.6.10.14\cdots(4n-2)}{n!}z^n/(n+1)\\
&= \sum_{n\geq 0}\frac{2^n 1.3.5\cdots(2n-1)}{n!}z^n/(n+1)\\
&= \sum_{n\geq 0}\frac{2^n n! 1.3.5\cdots(2n-1)}{n!n!}z^n/(n+1)\\
&= \sum_{n\geq 0}\frac{2.4.6\cdots(2n).1.3.5\cdots(2n-1)}{n!n!}z^n/(n+1)\\
&= \sum_{n\geq 0}\binom{2n}{n}z^n/(n+1).
\end{aligned}$$

But $g(z) = b_0z + b_1z + b_2z^2 + \cdots = \sum_{i=0}^{\infty} b_i z^i$ and hence by comparing the coefficient of z^n on both sides of the equation, we get

$$b_n = \frac{1}{n+1}\binom{2n}{n}.$$

But $b_n = c_{n+1}$. Hence the Catalan number $c_{n+1} = \frac{1}{n+1}\binom{2n}{n}$.

We give here the first few Catalan numbers:
$c_1 = 1, c_2 = 1, c_3 = 2, c_3 = 5, c_4 = 14, c_5 = 42, c_6 = 132.$

> Each equation I included in my book would halve the sales.
>
> *Stephan Hawking*

EXAMPLE 2.10.11:
Find the number of ways to multiply a chain of six $n \times n$ matrices $M_1 \times M_2 \times \cdots \times M_6$.

Solution:

The required number is the number of ways to parenthesize a word of length 6, which is $c_{5+1} = c_6 = \frac{1}{6}\binom{10}{5} = 42$.

EXAMPLE 2.10.12 (Generating function for a finite sum):
Let $g(z)$ be a generating function for the sequence $(a_i)_{i=0}^{\infty}$. Then prove that the generating function for the *partial sum* sequence (s_i) is $\frac{g(z)}{1-z}$ where $s_i = a_0 + a_1 + \cdots + a_i$ for all $i = 0, 1, \ldots$.

Solution:

We shall use the fact that $\frac{1}{1-z} = 1 + z + z^2 + z^3 \cdots$ (see 2.10.1)

$$\begin{aligned} g(z) &= a_0 + a_1 z + a_2 z^2 + \cdots, \\ \text{and hence } \frac{g(z)}{1-z} &= (a_0 + a_1 z + a_2 z^2 + \cdots)(1 + z + z^2 + \cdots) \\ &= \big(a_0 + (a_0 + a_1)z + (a_0 + a_1 + a_2)z^2 + \cdots\big) \\ &= s_0 + s_1 z + s_2 z^2 + \cdots \end{aligned}$$

where $s_i = a_0 + a_1 + \cdots + a_i$. This shows that $\frac{g(z)}{1-z}$ is the generating function for the partial sum sequence (s_i).

In the following example, we use *differentiation* to find the generating function.

Example 2.10.13:
Find the sum $1^2 + 2^2 + 3^2 + \cdots + n^2$.

Solution:

First we shall find the generating function for the sequence $(i^2)_{i=0}^{\infty}$.

$$\begin{aligned}
\frac{1}{1-z} &= 1 + z + z^2 + z^3 + \cdots . \text{ by 2.10.1} \\
\frac{d}{dz}\left(\frac{1}{1-z}\right) &= \frac{d}{dz}(1 + z + z^2 + z^3 + \cdots) \\
&\quad \text{by differentiating both sides w.r.t. } z \\
\Rightarrow \frac{1}{(1-z)^2} &= 1 + 2z + 3z^2 + 4z^3 + \cdots \\
\Rightarrow \frac{z}{(1-z)^2} &= z + 2z^2 + 3z^3 + 4z^4 + \cdots \\
\Rightarrow \frac{d}{dz}\left(\frac{z}{(1-z)^2}\right) &= 1^2 + 2^2 z + 3^2 z^2 + 4^2 z^3 + \cdots \\
\Rightarrow z\frac{d}{dz}\left(\frac{z}{(1-z)^2}\right) &= 0^2 + 1^2 z + 2^2 z^2 + 3^2 z^3 + 4^2 z^4 + \cdots
\end{aligned}$$

Hence, the generating function of the sequence (i^2) is $z\frac{d}{dz}\left(\frac{1}{(1-z)^2}\right)$ which on differentiation gives $g(z) = \frac{z(1+z)}{(1-z)^3}$.

By Example 2.10.12, the generating function for the partial sum (s_i) where $s_i = \sum_{j=0}^{i} j^2$ is $g(z)/(1-z)$. Therefore the sum $1^2 + 2^2 + 3^2 + \cdots + n^2$ is the coefficient of z^n in the expansion of $g(z)/(1-z) = \frac{z(1+z)}{(1-z)^4}$. By binomial Theorem 2.4.2,

$$\begin{aligned}
z(1+z)/(1-z)^4 &= (z+z^2)(1-z)^{-4} \\
&= (z+z^2)\left(\sum_{k\geq 0}\binom{-4}{k}(-z)^k\right)
\end{aligned}$$

Therefore the coefficient of z^n in $g(z)/(1-z)$ is the coefficient of z^{n-1} in $(1-z)^{-4}$+ the coefficient of z^{n-2} in $(1-z)^{-4}$.

The coefficient of z^n in $(1-z)^{-4}$ is $\frac{(-4)(-4-1)(-4-2)\cdots(-4-n+1)}{(}-1)^n n! = \frac{4\times 5\times 6\cdots\times(n+3)}{n!}$ which is equal to $\frac{(n+1)(n+2)(n+3)}{1.2.3}$. Therefore the required coefficient of z^n in $g(z)/(1-z)$ is $\frac{n(n+1)(n+2)}{1.2.3} + \frac{(n-1)n(n+1)}{1.2.3} = \frac{n(n+1)(2n+1)}{6}$.

2.11 Generating Subsets

Generating all the subsets of an n element set:

We shall study an *algorithm* (a step-by-step procedure to solve a problem) to generate all the possible subsets $\mathcal{P}[n]$ of the set $\{ n, n-1, \ldots, 2, 1 \}$. Note that the elements are written in decreasing order in the set $[n]$, the reason will be seen shortly. We have already seen that there are 2^n subsets of the set $[n]$. Consider the binary set $B = \{ 0, 1 \}$. The cardinality of the set B^n of all ordered n tuples of elements of B is also 2^n by the product rule. (An ordered n tuple written without parentheses and commas between elements is referred to as a *string or words of length n.* We also refer to the elements of B as *letters* and the set B as an *alphabet.* For example, the ordered triple $(0, 1, 1)$ is simply written as the string 011 whose length is 3.)

Since the two sets $\mathcal{P}[n]$ and B^n have the same number of elements, there are bijections from the set B^n onto the set $\mathcal{P}[n]$. In particular, we choose the *"natural bijection"* which associates to each element of B^n its corresponding *characteristic vector*, that is, the ith letter(component) of the string corresponds to the integer $n-i+1$ of the set $[n]$ with the meaning that the 0 value in the i-th letter implies the absence of the integer $n-i+1$ in the corresponding subset and 1 implies its presence in the subset. That is, the first coordinate of the string corresponds to the integer n, the second to $n-1$ and so on. Further, each characteristic vector of length n can be viewed as a number in base 2 and hence has a decimal value which lies between 0 and $2^n - 1$. Note that the decimal 0 corresponds to the binary number $(\overbrace{00\ldots0}^{n\ zeros})_2)$ and the decimal 1 corresponds to the binary $(\ \overbrace{00\ldots1}^{(n-1)\ zeros}\ )_2)$ (the subscript 2 represents the number in base 2). Note that the leading 0's are written in order to make the length of the string n.

Equality of strings: We declare two string $a_1a_2\ldots a_n$ and

$b_1 b_2 \dots b_n$ are equal if and only if $a_i = b_i$ for all possible subscripts i.

The algorithm we present uses this natural bijection to generate all the subsets of $[n]$. In other words the algorithm simply generates all the possible strings of 0's and 1's with length n.

The following example illustrates the "natural bijection" for the set $[3] = \{3, 2, 1\}$.

Example 2.11.1 (Natural bijection):

subsets	strings of length 3	decimal value
$\emptyset$	000	0
$\{1\}$	001	1
$\{2\}$	010	2
$\{2, 1\}$	011	3
$\{3\}$	100	4
$\{3, 1\}$	101	5
$\{3, 2\}$	110	6
$\{3, 2, 1\}$	111	7

The first column gives all possible subsets of [3], the second column the corresponding strings of 0's and 1's of length 3. The third column uses the *formula to convert binary into decimal*:

$$(b_{n-1} b_{n-2} \dots b_2 b_1 b_0)_2 = \sum_{i=0}^{n-1} b_i \times 2^i$$

For example, $(110)_2 = 0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 = 0 + 2 + 4 = 6$ with $n = 3$.

Conversion of integer into binary :

Conversely, to convert a nonnegative decimal integer n into a binary representation, we divide the integer n successively by 2 (by recording the remainders obtained during the division processes) *till we arrive at a quotient 0.* Then we write the sequence of remainders found in the reverse order.

The following example illustrates the conversion.

Example 2.11.2 (Conversion of decimal into binary):
Let us convert the integer $n = 75$ into decimal. The sequence of remainders obtained during the division is given in the following table.

Integer n	Quotient q	Remainder r
75	37	1
37	18	1
18	9	0
9	4	1
4	2	0
2	1	0
1	0	1

The sequence of remainders obtained is $(1, 1, 0, 1, 0, 0, 1)$ and writing this sequence in reverse order we obtain 1001011. Hence $75 = (1001011)_2$.

Verification: $(1 \times 2^0) + (1 \times 2^1) + (0 \times 2^2) + (1 \times 2^3) + (0 \times 2^4) + (0 \times 2^5) + (1 \times 2^6) = 75$.

Let us observe the order of listing of the strings in the second column of Example 2.11.1. They are listed in the *lexicographic order* just like the word "algorithm" comes before the word "almanac" in the dictionary. More generally, the string $a_1a_2 \dots a_p$ *lexicographically precedes* the string $b_1b_2 \dots b_p$ denoted by $a_1a_2 \dots a_p < b_1b_2 \dots b_p$ if there is some k, $1 \leq k \leq n$ such that $a_j = b_j$ for $1 \leq j < k$ but $a_k < b_k$. Geometrically, the initial segment of first $k-1$ characters of the two strings are equal and strict inequality holds in the kth character. Here we suppose $a_i's$ and $b_i's$ are integers. This is the reason why we list the elements of the set $[n]$ in the decreasing order. Note however that in the first column of Example 2.11.1 the subsets are listed *not* in lexicographic order but in the order called *squashed ordering* of the subsets of [3]. In this squashed ordering, we write first all the possible subsets of the current elements before considering a new element.

Example 2.11.3 (Correspondence: Subsets and strings):
Consider the set $[5] = \{ 5, 4, 3, 2, 1 \}$. The subset corresponding to the string 11001 of length 5 is $\{ 5, 4, 1 \}$ and the string of length 5

corresponding to the subset $\{4, 2\}$ is 01010.

Therefore the problem of generating all the subsets of a set of n elements is the *same as* the problem of generating in *the lexicographic order* all possible strings of 0's and 1's with length n. The first string in the lexicographic order is $(\overbrace{00\ldots0}^{n\ zeros})_2)$ and the last string is $(\overbrace{11\ldots1}^{n\ ones})_2)$. In Example 2.11.1 every new string is obtained by adding 1 (in base 2 arithmetic) to the previous string. So the algorithm proceeds as follows:

```
    We start with the string $(b_{n-1}b_{n-2}\ldots b_2b_1b_0) = (\overbrace{00\ldots0}^{n\ zeros})_2)$.
    while the string is not equal to $(\overbrace{11\ldots1}^{n\ ones})_2)$ do
begin
   add 1 to the string $(b_{n-1}b_{n-2}\ldots b_2b_1b_0)$
   write the string thus obtained;
end while;
```

The following example illustrates the operation of adding the bit (*binary digit*) 1 to a string of 0's and 1's:

Example 2.11.4 (Addition of bit 1 to a string):
The addition uses the four equations: $0+0=0, \quad 0+1=1+0=1, \quad 1+1=10$.

Let us add 1 to the string 1010111. By using the equations and carrying out the bit, we obtain $1010111+1=1011000$.

We observe that to add 1 to 1010111, it is enough to replace the *first* bit 0 from the *right* by 1 and all other 1's to the right of the first 0 from right, (which by definition are all 1) by 0's.

Thus, from the above example, we get the general rule for adding the bit 1 to a string of 0's and 1's $(\neq (\overbrace{11\ldots1}^{nones}))$. The rule is: Replace the *first* bit 0 from the right of the string by 1 and all other bits to the right of the first 0 from the right by 0's.

We are now ready to give the complete algorithm to generate all strings of 0's and 1's.

Algorithm to generate all strings of 0's and 1's with length n:

```
(*INITIALIZATION of $(b_{n-1}b_{n-2}\ldots b_2b_1b_0)$*)
$(b_{n-1}b_{n-2}\ldots b_2b_1b_0) \leftarrow \overbrace{00\ldots0}^{n\ zeros})$
(*ITERATION*)
while the string $(b_{n-1}b_{n-2}\ldots b_2b_1b_0)$ is not equal to $(\overbrace{11\ldots1}^{n\ ones})_2)$ do
begin while
     Replace the *first* bit 0 from the *right* of the string $(b_{n-1}b_{n-2}\ldots b_2b_1b_0)$ by 1 and all other bits to the right of the first 0 from right, by 0's;
    write the newly obtained string $(b_{n-1}b_{n-2}\ldots b_2b_1b_0)$;
end while;
```

Example 2.11.5:
Consider the set $[6] = \{ 6, 5, 4, 3, 2, 1 \}$. In the above algorithm for generating the subsets, let us find the subset following immediately the subset $\{ 6, 4, 3, 1 \}$.
The subset $\{ 6, 4, 3, 1 \}$ is represented by the string 101101 of length 6. To find the next string, replace the first 0 from right by the bit 1 and replace all other bits to the right of this 0 by 1's. We obtain the string 101110. The subset corresponding to the string 101110 is $\{ 6, 4, 3, 2 \}$, which is the set following $\{ 6, 4, 3, 1 \}$.

Algorithm to generate all possible k-subsets of an n-set in lexicographic order:

Let us generate all possible k-subsets of the n-set $n = \{ 1, 2, \ldots, n \}$. Note that the elements of $[n]$ are written in the natural strictly increasing order, in *contrast* to the strictly decreasing order used in the algorithm for generating all possible subsets of $[n]$. Let us first agree to write a subset as a string/word of strictly increasing letters/elements. Evidently, in this writing the letters are distinct. The following example clarifies our convention.

Example 2.11.6 (Subsets in lexicographic order):
Let [6] = $\{1,2,3,4,5,6\}$ and let us write the 4-subsets of [6] in lexicographic order as a string of increasing letters.

4-subsets	string notation	4-subsets	string notation
$\{1,2,3,4\}$	1234	$\{1,3,5,6\}$	1356
$\{1,2,3,5\}$	1235	$\{1,4,5,6\}$	1456
$\{1,2,3,6\}$	1236	$\{2,3,4,5\}$	2345
$\{1,2,4,5\}$	1245	$\{2,3,4,6\}$	2346
$\{1,2,4,6\}$	1246	$\{2,3,5,6\}$	2356
$\{1,2,5,6\}$	1256	$\{2,4,5,6\}$	2456
$\{1,3,4,5\}$	1345	$\{3,4,5,6\}$	3456
$\{1,3,4,6\}$	1346		

We observe that the strings of length 4 (equivalently subsets of cardinality 4) in the second column of Example 2.11.6 are listed in lexicographic order. The first one in the list is 1234 and the last one is 3456.

More generally, the *first* string in lexicographic order of k distinct elements of the n-set $[n]$ is $12\ldots k$ and the *last* string is $(n-k+1)(n-k+2)\ldots n$.

Question: How can we find a string of length k in $[n]$ immediately following the k-string $a_1a_2\ldots a_k \neq (n-k+1)(n-k+2)\ldots n$ in lexicographic order?

To answer the question, let us again refer to Example 2.11.6. By studying this example closely, we arrive at the following two cases.

Let us first dispose of the trivial case.

Case 1: The last letter a_4 in the string is $\neq 6$. For example, the string 1245 with $a_4 = 5$.

In this case, by the definition of lexicographic ordering, the string immediately following is obtained by replacing the last letter a_4 by $a_4 + 1$ while keeping the other letters fixed. For example, the string immediately following 1245 is 1246.

Case 2: The last letter of the string $a_4 = 6$. For example, consider the string 1256 with $a_4 = 6$.

In this case, find the largest length of consecutive integers from the right of the string (also called *the largest length of consecutive* *suffix*). In the example, in the string 1256, the largest suffix of consecutive integers is 56 and its length is 2. Then replace the third ($= 2+1$) integer from the right of the string by its immediate successor. The string obtained is 1345.

We generalize our example, to give the following rule to find the immediate successor in the lexicographic order of the string $a_1a_2\ldots a_k \neq (n-k+1)(n-k+2)\ldots n$ with $a_1a_2 < \cdots < a_k$ in the set $[n]$.

Case 1: The last letter a_k in the string is $\neq n$.

In this case, by the definition of lexicographic ordering, the string immediately following is obtained by replacing the last letter a_k by a_k+1 while keeping the other letters fixed.

Case 2: The last letter of the string $a_k = n$. For example, consider the string 1256 with $a_n = 6$.

In this case, find the largest length l of consecutive integers from the right of the string (also called *the largest length of consecutive suffix*). Then replace the $l+1$-st integer from the right of the string with its immediate successor.

We are now ready to write the algorithm in its general form:

Algorithm to generate all the k-subsets of an n-set $[n]$

```
(* INITIALIZATION*)
$a_1a_2\ldots a_k \leftarrow 12\ldots k$;
(* ITERATION *)
while ($a_1a_2\ldots a_k \neq (n-k+1)(n-k+2)\ldots n$) do
begin

    if $a_k \neq n$
then $c \leftarrow a_1a_2\ldots(a_k+1)$
    else
    begin
        $l \leftarrow$ the length of the largest suffix of $a_1a_2\ldots a_k$;
        $a_1a_2\ldots a_l\ldots a_k \leftarrow a_1a_2\ldots(a_l+1)\ldots a_k$
    end
end
```

Example 2.11.7:
Let [8] = $\{1, 2, \ldots, 8\}$ and let us find the subset immediately following the 5-set 125678 in lexicographic order.

The last integer of the string is equal to 8. Hence we are in Case 2. The largest suffix of consecutive integers is 5678 and its length is 4. So by our rule, we replace the fifth integer 2 from the right with its successor 3. Hence the required string is 135678.

On the other hand, the string coming just after 2567 is 2368 (we are in Case 1, because the last integer in the string is $7 \neq 8$ and we add 1 to the last integer 7 to get 8).

We are now interested in finding the place of occurrence of the k-subset $a_1 a_2 \ldots a_l \ldots a_k$ among all the k-subsets of the set $[n]$. Let us again refer to Example 2.11.6. For example, the string 1236 occurs in the 3rd place and the string 2345 occurs in the 11th place.

More generally, we are going to prove a formula for the place of occurrence of the string $a_1 a_2 \ldots a_l \ldots a_k$ in the list of all possible k-strings of the set $[n]$.

Theorem 2.11.1:
Consider the k-string/subset $a_1 a_2 \ldots a_k$ of the $[n]$-set $\{1, 2, \ldots, n\}$. The place of occurrence of the k-subset/string $a_1 a_2 \ldots a_k$ in lexicographic order among the list of all possible k-subsets of $[n]$ is given by

$$\binom{n}{k} - \binom{n-a_1}{k} - \binom{n-a_2}{k-1} - \cdots - \binom{n-a_{k-1}}{2} - \binom{n-a_k}{1}.$$

Proof. Our proof goes as follows: We use the subtraction rule. First we find the number of k-subsets coming *after* (in lexicographic order) the k-subset $a_1 a_2 \ldots a_k$. Then we subtract this number from the total number of k-subsets of $[n]$, which is the binomial coefficient $\binom{n}{k}$ to obtain the desired place. Let us recall that a k-subset is written as a string $a_1 a_2 \ldots a_k$ with $a_1 < a_2 < \cdots < a_k$.

Therefore let us first find the number of k-subsets following the k-subset $a_1 a_2 \ldots a_k$. This number is split into k numbers:

1. The number of k-subsets coming after $a_1a_2\dots a_k$ whose first element is strictly greater than a_1.

2. The number of k-subsets coming after $a_1a_2\dots a_k$ whose first element (prefix) is a_1 but the second one is strictly greater than a_2.

$\vdots$

i. The number of k-subsets coming after $a_1a_2\dots a_k$ whose prefix (initial segment) is $a_1a_2\dots a_{i-1}$ but whose i-th element is strictly greater than a_i.

$\vdots$

k. The number of k-subsets coming after $a_1a_2\dots a_k$ whose prefix (initial segment) is $a_1a_2\dots a_{k-1}$ but the k-th one is strictly greater than a_k.

These k numbers are, respectively, $\binom{n-a_1}{k}$, $\binom{n-a_2}{k-1}$ $\dots$ $\binom{n-a_i}{k-(i-1)}$ $\binom{n-a_k}{k-(k-1)}$.

Therefore the number of k-subsets coming after the k-subset $a_1a_2\dots a_k$ is the sum

$$\binom{n-a_1}{k}+\binom{n-a_2}{k-1}+\cdots+\binom{n-a_{k-1}}{2}+\binom{n-a_k}{1}$$

Subtracting the above sum from the total number $\binom{n}{k}$ of k-subsets of $[n]$ we get the formula of the theorem. ■

Example 2.11.8:

Let us find the place of occurrence of the 4-subset 2346 among the list of 4-subsets of [6] written in lexicographic order.

Here $n = 6$ and $k = 4$. According to the above formula the place of occurrence of the string 2346 is

$$\binom{6}{4}-\binom{6-2}{4}-\binom{6-3}{3}-\binom{6-4}{2}-\binom{6-6}{1}$$

which is equal to $15-1-1-1-0=12$.

The subset/string 2346 occurs in the 12th place as can be seen from Example 2.11.6.

2.12 Exercises

1. Consider any sequence of 13 distinct integers. Verify that your sequence always contains either a strictly increasing subsequence of 4 integers or strictly decreasing subsequence of 3 integers.

2. What is the *minimum* number of elements of a sequence of distinct integers to ensure that the sequence will always contain either a strictly increasing subsequence of 4 integers or a strictly decreasing subsequence of 4 integers.

3. Consider any sequence of 10 integers. Find a subsequence of consecutive integers of your sequence whose sum is divisible by 10.

4. Prove or disprove (n, a non-negative integer): $n!$ is always even.

5. Prove or disprove (n, a non-negative integer): $n!$ is always divisible by 10 if $n \geq 5$.

6. Prove that $(n! + 1)$ is not divisible by any integer d where $2 \leq d \leq n$.

7. Prove that

$$\frac{2n!}{n!} = (1 \times 3 \times 5 \times \cdots (2n - 1)) \times 2^n$$

8. Find the *greatest common divisor* (gcd) of the three numbers $6!, 7!, 9!$ What is their *least common multiple*?

9. Find the number of words of length 5 which can be formed from the letters of the word PARIS when

   (a) the letters are distinct

   (b) the repetition of the letters is allowed

10. Find the number of ways in which 6 letters can be distributed among 4 letter boxes.

11. Find the number of positive divisors of

    (a) 10^3

    (b) $2^3 \times 3^5 \times 7^4$

    (c) 1729 (Ramanujan number)

12. Find the number of words of 4 distinct letters that can be made from the English alphabet.

13. Find the number of integers between 100 and 1000 such that the digit 5 is in the first (unit's) place.

14. Find the number of integers between 100 and 1000 such that at least one of their digits is 3.

15. Find the ordered two partition of n distinct objects or in other words, find the number of ways in which n objects can be put into two *labeled boxes* B_1 and B_2 (a box can be possibly empty).

16. Find the number of ways of putting n objects into two *labeled boxes* B_1 and B_2 in which each box contains at least one element.

17. In how many ways can 4 boys and 5 girls be seated in a row so that boys are always seated in even places?

18. Find the number of five-digit integers with distinct digits.

19. Find the number of integers less than 1000 in which all digits are distinct.

20. Find the number of ways of seating 5 boys and 3 girls in a row so that no two girls are together.

21. Find the number of ways of seating 4 boys and 4 girls in a row so that:

    (a) No two girls sit together.

    (b) All the girls sit together and all the boys sit together.

(c) All the girls are never together.

22. Find the number of *nonnegative* integral solutions of the equation
$$x_1 + x_2 + x_3 + x_4 = 18.$$

23. Find the number of *positive* integral solution of the equation
$$x_1 + x_2 + x_3 + x_4 = 10$$

24. Find the number of integral solutions of the equation
$$x_1 + x_2 + x_3 + x_4 = 18$$
where $x_1 \geq 3, \quad x_2 \geq 1, \quad x_3 \geq 0, \quad x_4 \geq 5$

25. Give the number of functions from a 3-set to a 5-set.

26. Find the number of relations from a 3-set to a 5-set.

27. Using the Stirling number of the second kind, find the number of surjections from a 5-set onto the 3-set.

28. Find the number of injections from a 4-set into a 6-set.

29. Give the number of bijections from a 5-set onto a 5-set.

30. Give the number of *circular/cyclic* permutations of a 5-set.

31. Give the number of *linear* permutations of a 5-set.

32. Using the Stirling number of the first kind, find the number of permutations of a 5-set having two cycles.

33. Find the number of derangements of a 5-set.

34. List all the possible derangements of a 4-set.

35. Describe the sum of the Stirling numbers of the second kind $\sum_{k=0}^{n} \left\{ {n \atop k} \right\}$ in terms of the notion of equivalence relations on the set $\{ 1, 2, \ldots, n \}$.

36. What value of k makes the binomial coefficient $\binom{10}{k}$ a maximum?

37. What value of k makes the binomial coefficient $\binom{11}{k}$ a maximum?

38. What value of k makes the binomial coefficient $\binom{n}{k}$ a minimum?

39. Find the number of terms in the expansion of $(x_1 + x_2 + x_3 + x_4)^5$.

40. What is the sum of all the coefficients in the expansion of $(x_1 + x_2 + x_3 + x_4)^5$?

41. Find the number of 4-multisubsets of a 3-set.

42. Find the number of increasing functions (not necessarily strictly increasing) from the 5-set $[5] = \{ 1, 2, 3, 4, 5 \}$ into the 3-set $[3] = \{ 1, 2, 3 \}$.

43. Using the Stirling number of the second kind, find the number of equivalence relation that can be defined on the set $[5] = \{ 1, 2, 3, 4, 5 \}$.

44. Using the algorithm studied in the text, generate all four subsets of the 6-set $[6] = \{ 1, 2, \ldots, 6 \}$ in lexicographic order.

45. Using the algorithm studied in the text, generate all 4-tuples of 0's and 1's in lexicographic order.

46. Use the binomial theorem to find the value of 11^5.

47. Find the number of 11-permutations of the multiset $\{ 3.a, 4.b, 5.c \}$.

48. Show that any five points selected within a square of side length 2, there must exist two points such that their Euclidean distance is strictly less than $\sqrt{2}$.
Hint: Divide the rectangle into four equal parts and apply the pigeon-hole principle.

49. Choose any ten points inside an equilateral triangle of side length 1. Show that there exist two points whose Euclidean distance is strictly less than $1/3$.

50. Prove that if $n+1$ integers are chosen from the set $\{1, 2, \ldots, 2n\}$, then there must be two integers differing by 1.

51. Prove that if $n+1$ integers are chosen from the set $\{1, 2, \ldots, 3n\}$, then there must be two integers differing by at most 2.

52. How many points must be chosen inside an equilateral triangle of side length 1 to ensure two points at Euclidean distance strictly less than $1/n$?

53. Consider the n-set $[n] = \{1, 2, \ldots, n\}$ and a collection $\mathcal{A}$ of subsets of $[n]$ such that no two subsets in $\mathcal{A}$ are disjoint. Prove that $|\mathcal{A}| \leq 2^{n-1}$.

54. In a class of 25 students, 15 like mathematics, 12 like physics, 11 like chemistry, 5 like both mathematics and chemistry, 9 like both mathematics and physics, 4 like both physics and chemistry, and 3 like all the three subjects. Find the number of students who like

    (a) only chemistry

    (b) only mathematics

    (c) only physics

    (d) physics and chemistry but not mathematics

    (e) mathematics and physics but not chemistry

    (f) only one of the subjects

    (g) at least one of the three subjects

    (h) none of the subjects

55. Let S be a finite set and let A_1, A_2, A_3 be three subsets of the set S. Give a formula for the following:

(a) The number of elements in *exactly two* of the sets A_1, A_2, A_3 in terms of the intersections of some of the sets A_1, A_2, A_3.

(b) The number of elements in *exactly one* of the sets A_1, A_2, A_3 in terms of A_1, A_2, A_3 and the intersections of the some of them.

Bibliography

[1] D. E. Knuth, Art of Computer Programming, Fundamental Algorithms, Vol. 1, Addison-Wesley, Reading, Mass (1968).

[2] L. Lovász, Combinatorial Problems and Exercises, 2nd Edition, Elsevier, North-Holland Publishing Company, Amsterdam, 2003.

[3] C. Berge, The Principles of Combinatorics, Academic Press, New York, 1971.

[4] V. Krishnamurthy, Combinatorics: Theory and Applications, Affiliated East-West Press, 1985.

[5] H. J. Ryser, Combinatorial Mathematics, The Carus Mathematical Monographs, The Mathematical Association of America, Distributed by John Wiley and Sons, Inc., 1963.

[6] R. L. Graham, D. E. Knuth, O. Patashnik, Concrete Mathematics, Pearson Education, Inc. and Dorling Kindersley Publishing, Inc. (1994).

[7] D. E. Littlewood, The Skeleton Key of Mathematics, Dover Publications, Inc. Mineola, NY (2002).

[8] L.V. Ahlfors, Complex Analysis, Third edition, McGraw-Hill Book Company, 1979.

Chapter 3

Basics of Number Theory

Summary

As the title indicates, this chapter deals with basics of number theory. To begin with, the greatest common divisor (gcd) and the least common multiple (lcm) of two nonzero numbers are defined. The Euclidean algorithm and the extended Euclidean algorithm for any two nonzero numbers are then established.

The next section deals with the property of primes. The unique factorization theorem which states that any positive integer is expressible uniquely, up to order, as a product of primes is established. Linear congruences and complete residue systems are then explained.

In the following sections, some of the fundamental theorems in basic number theory, namely, Euler's theorem, Wilson's theorem, Chinese remainder theorem for linear congruences are established. These are followed by a discussion on lattice points visible from the origin. The chapter closes with a discussion of some arithmetical functions, a brief discussion of big O notation and polynomial time algorithms, and a proof of the fact that Euclid's algorithm for finding the greatest common divisor of two (nonzero) integers is a polynomial time algorithm. Some concepts found here might have found a place in Chapter 2 as well, but in passing.

3.1 Introduction

In this chapter, we present some basics of number theory. These include divisibility, primes, congruences, some number-theoretic functions and the Euclidean algorithm for finding the gcd of two numbers. We also explain the big O notation and polynomial time algorithms. We show, as examples, that the Euclidean algorithm and the modular exponentiation algorithm are polynomial time algorithm. We denote by $\mathbf{Z}$, the set of integers and $\mathbf{N}$, the set of positive integers.

3.2 Divisibility

Definition 3.2.1:
Let a and b be any two integers and $a \neq 0$. Then b is divisible by a (equivalently, a is a divisor of b) if there exists an integer c such that $b = ac$.

If a divides b, it is denoted by $a \mid b$. If a does not divide b, we denote it by $a \nmid b$.

Theorem 3.2.2:

1. *$a \mid b$ implies that $a \mid bc$ for any integer c.*
2. *If $a \mid b$ and $b \mid c$, then $a \mid c$.*
3. *If a divides $b_1, b_2, \ldots, b_n$, then a divides $b_1x_1 + \cdots + b_nx_n$ for integers $x_1, \ldots, x_n$.*
4. *$a \mid b$ and $b \mid a$ imply that $a = \pm b$.*

The proofs of these results are trivial and are left as exercises. (For instance, to prove (iii), we note that if a divides $b_1, \ldots, b_n$, there exist integers c_1, $c_2, \ldots, c_n$ such that $b_1 = ac_1$, $b_2 = ac_2$, $\ldots$, $b_n = ac_n$. Hence $b_1x_1 + \cdots + b_nx_n = a(c_1x_1 + \cdots + c_nx_n)$, and so a divides $b_1x_1 + \cdots b_nx_n$).

Theorem 3.2.3 (Division algorithm)**:**
Given any integers a and b with $a \neq 0$, there exist unique integers q and r such that

$$b = qa + r, \quad 0 \leq r < |a|, \text{ where } |a| = max(a, -a).$$

Proof. Consider the arithmetic progression

$$\dots, b - 3|a|, \; b - 2|a|, \; b - |a|, \; b, \; b + |a|, \; b + 2|a|, \; b + 3|a|, \dots$$

with common difference $|a|$ and extended infinitely in both the directions. Certainly, this sequence contains a least non-negative integer r. Let this term be $b + q|a|$, $q \in \mathbf{Z}$. Thus

$$b + q|a| = r \tag{3.1}$$

and, its previous term, namely, $r - |a| < 0$ so that $r < |a|$. If $a > 0$, then (3.1) gives

$$b = -qa + r, \quad 0 \leq r < |a| = a,$$

while if $a < 0$, $|a| = -a$ so that $b = qa + r$, $0 \leq r < |a|$. It is clear that the numbers q and r are unique. ■

Theorem 3.2.3 gives the division algorithm, that is, the process by means of which division of one integer by a nonzero integer is made in the set of integers. An algorithm is a step-by-step procedure to solve a given mathematical problem. We next present Euclid's algorithm to determine the gcd of two numbers a and b. Euclid's algorithm (300 BCE) is the first known algorithm in the mathematical literature. It is just the usual algorithm taught in high school algebra.

3.3 The Greatest Common Divisor (gcd) and the Least Common Multiple (lcm) of Two Integers

Definition 3.3.1:
Let a and b be two integers, at least one of which is not zero. A common divisor of a and b is an integer $c(\neq 0)$ such that $c \mid a$ and $c \mid b$. The greatest common divisor of a and b is the greatest of the common divisors of a and b. It is denoted by (a, b).

If c divides a and b, then so does $-c$. Hence $(a,\, b) > 0$ and is uniquely defined. Moreover, if c is a common divisor of a and b, that is, if $c \mid a$ and $c \mid b$, then

$$a = a'c \quad \text{and} \quad b = b'c$$

for integers a' and b'. Hence

$$(a, b) = c(a', b')$$

so that $c \mid (a, b)$. Thus any common divisor of a and b divides the gcd of a and b. Hence (a, b) is the least positive common divisor of a and b that is divisible by every common divisor of a and b. Moreover, $(a, b) = (\pm a, \pm b)$.

Proposition 3.3.2:
If $c \mid ab$ and $(c, b) = 1$, then $c \mid a$.

Proof. By hypothesis $c \mid ab$. Trivially $c \mid (ac)$. Hence c is a common divisor of ab and ac. Hence c is a divisor of $(ab,\, ac) = a(b,\, c) = a$, as $(b,\, c) = 1$. ∎

Definition 3.3.3:
If a, b and c are nonzero integers and if $a \mid c$ and $b \mid c$, then c is called a common multiple of a and b.

The least common multiple (lcm) of a and b is the smallest of the positive common multiples of a and b and is denoted by $[a,\, b]$. As in the case of gcd, $[a,\, b] = [\pm a,\, \pm b]$.

Euclid's Algorithm

Since $(\pm a,\, \pm b) = (a,\, b)$, we may assume without loss of generality that $a > 0$ and $b > 0$ and that $a > b$ (if $a = b$, then $(a,\, b) = (a,\, a) = a$). By the Division Algorithm (Theorem 3.2.3), there exist integers q_1 and r_1 such that

$$a = q_1 b + r_1, \quad 0 \le r_1 < b. \tag{3.2}$$

If $r_1 \ne 0$, divide b by r_1 and get

$$b = q_2 r_1 + r_2, \quad 0 \le r_2 < r_1. \tag{3.3}$$

If $r_2 \ne 0$, divide r_1 by r_2 and get

$$r_1 = q_3 r_2 + r_3, \quad 0 \le r_3 < r_2. \tag{3.4}$$

Proceeding in this way, at the $(i+2)$-th stage, we get the equation

$$r_i = q_{i+2} r_{i+1} + r_{i+2}, \quad 0 \le r_{i+2} < r_{i+1}. \tag{3.5}$$

Since the sequence of remainders $r_1,\, r_2, \ldots$ is strictly decreasing, this procedure must stop at some stage, say,

$$r_{j-1} = q_{j+1} r_j. \tag{3.6}$$

Then $(a, b) = r_j$.

Proof. First we show that r_j is a common divisor of a and b. To see this we observe from Equation (3.6) that $r_j \mid r_{j-1}$. Now Equation (3.5) for $i = j - 2$ is

$$r_{j-2} = q_j r_{j-1} + r_j. \tag{3.7}$$

Since $r_j \mid r_{j-1}$, r_j divides the expression on the right side of (3.7), $r_j \mid r_{j-2}$. Going backward, we get successively that r_j divides $r_{j-1}, r_{j-2}, \ldots, r_1$, b and a. Thus $r_j \mid a$ and $r_j \mid b$.

Next, let $c \mid a$ and $c \mid b$. Then from Equation (3.2), $c \mid r_1$, and this when substituted in Equation (3.3), gives $c \mid r_2$. Thus the successive equations of the algorithm yield $c \mid a$, $c \mid b$, $c \mid r_1$, $c \mid r_2$, $\ldots$, $c \mid r_j$. Thus any common divisor of a and b is a divisor of r_j.

Consequently, $r_j = (a, b)$. ■

Extended Euclidean Algorithm

Theorem 3.3.4:
If $r_j = (a, b)$, then it is possible to find integers x and y such that

$$ax + by = r_j \tag{3.8}$$

Proof. The equations preceding (3.6) are

$$\begin{aligned} r_{j-3} &= q_{j-1} r_{j-2} + r_{j-1}, \quad \text{and} \\ r_{j-2} &= q_j r_{j-1} + r_j. \end{aligned} \tag{3.9}$$

Equation (3.9) expresses r_j in terms of r_{j-1} and r_{j-2} while the equation preceding it expresses r_{j-1} in terms of r_{j-2} and r_{j-3}. Thus

$$\begin{aligned} r_j &= r_{j-2} - q_j r_{j-1} \\ &= r_{j-2} - q_j \left(r_{j-3} - q_{j-1} r_{j-2} \right) \\ &= \left(1 + q_j q_{j-1} \right) r_{j-2} - q_j r_{j-3}. \end{aligned}$$

Thus we have expressed r_j as a linear combination of r_{j-2} and r_{j-3}, the coefficients being integers. Working backward, we get r_j as a linear combination of a, b with the coefficients being integers. ∎

The process given in the proof of Theorem 3.8 is known as the *Extended Euclidean Algorithm.*

Corollary 3.3.5:
If $(a, m) = 1$, then there exists an integer u such that $au \equiv 1 \pmod{m}$ and any two such integers are congruent modulo m.

Proof. By the Extended Euclidean Algorithm, there exist integers u and v such that $au + mv = 1$. This however means that $au \equiv 1 \pmod{m}$. The second part is trivial as $(a, m) = 1$. ∎

Example 3.3.6:
Find integers x and y so that $120x + 70y = 10$.

We apply Euclid's algorithm to $a = 120$ and $b = 70$. We have

$$120 = 1 \cdot 70 + 50$$

$$\begin{aligned} 70 &= 1 \cdot 50 + 20 \\ 50 &= 2 \cdot 20 + 10 \\ 20 &= 2 \cdot 10 \\ \text{Hence} \quad \gcd(120,\, 70) &= 10. \end{aligned}$$

Now starting from the penultimate equation and going backward, we get

$$\begin{aligned} 10 &= 50 - 2 \cdot 20 \\ &= 50 - 2(70 - 1 \cdot 50) \\ &= 3 \cdot 50 - 2 \cdot 70 \\ &= 3 \cdot (120 - 1 \cdot 70) - 2 \cdot 70 \\ &= 3 \cdot 120 - 5 \cdot 70. \end{aligned}$$

Therefore $x = 3$ and $y = -5$ fulfill the requirement. Note that we have obtained the desired result by replacing the successive remainders by going backward.

3.4 Primes

Definition 3.4.1:
An integer $n > 1$ is a prime if its only positive divisors are 1 and n. A natural number greater than 1 which is not a prime is a composite number.

Naturally, 2 is the only even prime. 3, 5, 7, 11, 13, 17, ... are all odd primes. The composite numbers are 4, 6, 8, 9, 10, ...

Theorem 3.4.2:
Every integer $n > 1$ can be expressed as a product of primes.

Proof. The result is obvious for $n = 2$, 3 and 4. So assume that $n > 4$ and apply induction. If n is a prime, there is nothing to prove. If n is not a prime, then $n = n_1 n_2$, where $1 < n_1 < n$ and $1 < n_2 < n$. By induction hypothesis, both n_1 and n_2 are products of primes. Hence n itself is a product of primes. (Note that the prime factors of n need not all be distinct.) ■

Suppose in a prime factorization of n, the distinct prime factors are $p_1, p_2, \ldots, p_r$, and that p_i is repeated α_i times, $1 \leq i \leq r$,

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_r^{\alpha_r}. \tag{3.10}$$

We now show that this factorization of n is unique in the sense that in any prime factorization, the prime factors that occur are the same and that the prime powers are also the same except for the order of the prime factors. For instance, $200 = 2^3 \times 5^2$, and the only other way to write it in the form (3.10) is $5^2 \times 2^3$.

Theorem 3.4.3 (Unique factorization theorem):
Every positive integer $n > 1$ can be expressed uniquely in the form

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_r^{\alpha_r}$$

where p_i, $1 \leq i \leq r$, are distinct primes; the above factorization is unique except for the order of the primes.

To prove Theorem 3.4.3, we need a property of primes.

Lemma 3.4.4:
If p is a prime such that $p \mid (ab)$, but $p \nmid a$, then $p \mid b$.

Proof. As $p \nmid a$, $(p, a) = 1$. Now apply Proposition 3.3.2. ■

Note 3.4.5:
Theorem 3.4.4 implies that if p is a prime and $p \nmid a$ and $p \nmid b$, then $p \nmid (ab)$. More generally, $p \nmid a_1$, $p \nmid a_2$, $\ldots p \nmid a_n$ imply that $p \nmid (a_1 a_2 \cdots a_n)$. Consequently if $p \mid (a_1 a_2 \cdots a_n)$, then p must divide at least one a_i, $1 \leq i \leq n$.

Proof. (of Theorem 3.4.3) Suppose

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_r^{\alpha_r} = q_1^{\beta_1} q_2^{\beta_2} \cdots q_j^{\beta_j} \cdots q_s^{\beta_s} \tag{3.11}$$

are two prime factorizations of n, where the p_i's and q_i's are all primes. As $p_1 \mid (p_1^{\alpha_1} \cdots p_r^{\alpha_r})$, $p_1 \mid \left(q_1^{\beta_1} \cdots q_s^{\beta_r}\right)$. Hence by Lemma 3.4.4, p_1 must divide some q_j. As p_1 and q_j are primes and $p_1 \mid q_j$, $p_1 = q_j$. Cancelling p_1 on both sides, we get

$$p_1^{\alpha_1 - 1} p_2^{\alpha_2} \cdots p_r^{\alpha_r} = q_1^{\beta_1} q_2^{\beta_2} \cdots q_j^{\beta_j - 1} \cdots q_s^{\beta_s}. \tag{3.12}$$

Now argue as before with p_1 if $\alpha_1 - 1 \geq 1$. If $\alpha_1 < \beta_j$, this procedure will result in the relation

$$p_2^{\alpha_2} \cdots p_r^{\alpha_r} = q_1^{\beta_1} q_2^{\beta_2} \cdots q_j^{\beta_j - \alpha_1} \cdots q_s^{\beta_s}. \tag{3.13}$$

Now p_1 divides the right hand expression of (3.13) and so must divide the left-hand expression of (3.13). But this is impossible as the p_i's are distinct primes. Hence $\alpha_1 = \beta_j$. Cancellation of $p_1^{\alpha_1}$ on both sides of (3.11) yields

$$p_2^{\alpha_2} \cdots p_r^{\alpha_r} = q_1^{\beta_1} q_2^{\beta_2} \cdots q_{j-1}^{\beta_{j-1}} q_{j+1}^{\beta_{j+1}} \cdots q_s^{\beta_s}.$$

Repetition of our earlier argument gives $p_2 =$ one of the q_i's, $1 \leq i \leq s$, say, q_k, $k \neq j$. Hence $\alpha_2 = \beta_k$ and so on. This shows that each $p_i =$ some q_t and that $\alpha_i = \beta_t$ so that $p_i^{\alpha_i} = q_t^{\alpha_t}$. Cancellation of $p_1^{\alpha_1}$ followed by $p_2^{\alpha_2}, \ldots, p_r^{\alpha_r}$ on both sides will leave 1 on the left side expression of (3.11) and so the right side expression of (3.11) should also reduce to 1. ■

The unique factorization of numbers enables us to compute the gcd and lcm of two numbers. Let a and b be any two integers ≥ 2. Let $p_1, \ldots, p_r$ be the primes, each of which divides either a or b. Then a and b can be written uniquely in the form

$$\begin{aligned} a &= p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_r^{\alpha_r} \\ b &= p_1^{\beta_1} p_2^{\beta_2} \cdots p_r^{\beta_r}, \end{aligned}$$

where α_i's and β_j's are non-negative integers. (Taking α_i's and β_j's to be nonnegative integers in the prime factorizations of a and b, instead of taking them to be positive, enables us to use the same prime factors for both a and b. For instance, if $a = 72$ and $b = 45$, we can write a and b as,

$$\begin{aligned} a &= 2^3 \cdot 3^2 \cdot 5^0 \\ \text{and} \quad b &= 2^0 \cdot 3^2 \cdot 5^1.) \end{aligned}$$

Then clearly,

$$\begin{aligned} (a,\, b) &= \prod_{i=1}^{r} p_i^{\min(\alpha_i,\, \beta_i)}, \quad \text{and} \\ [a,\, b] &= \prod_{i=1}^{r} p_i^{\max(\alpha_i,\, \beta_i)} \end{aligned}$$

We next establish two important properties of prime numbers.

Theorem 3.4.5 (Euclid):
The number of primes is infinite.

Proof. The proof is by contradiction. Suppose there are only finitely many primes, say, $p_1, p_2, \ldots, p_r$. Then the number $n = 1 + p_1 p_2 \cdots p_r$ is larger than each p_i, $1 \leq i \leq r$, and hence composite. Now any composite number is divisible by some prime. But none of the primes p_i, $1 \leq i \leq r$, divides n. (For, if p_i divides n, then $p_i \mid 1$, an impossibility). Hence the number of primes is infinite. ■

Theorem 3.4.6 (Nagell):
There are arbitrarily large gaps in the sequence of primes. In other words, for any positive integer $k \geq 2$, there exist k consecutive composite numbers.

Proof. The k numbers

$$(k+1)! + 2,\ (k+1)! + 3, \ldots, (k+1)! + k,\ (k+1)! + (k+1)$$

are consecutive. They are all composite since $(k+1)!+2$ is divisible by 2, $(k+1)!+3$ is divisible by 3 and so on. In general $(k+1)!+j$ is divisible by j for each j, $2 \leq j \leq k+1$. ■

Definition 3.4.7:
Two numbers a and b are coprime or relatively prime or prime to each other, if $gcd(a, b) = 1$.

3.5 Exercises

1. For any integer n, show that $n^2 - n$ is divisible by 2, $n^3 - n$ by 6 and $n^5 - n$ by 30.

2. Show that $(n,\, n+1) = 1$ and that $[n,\, n+1] = n(n+1)$.

3. Use the unique factorization theorem to prove that for any two positive integers a and b, $(a,\, b)[a,\, b] = ab$, and that $(a,\, b) \mid [a,\, b]$. (Remark: This shows that if $(a,\, b) = 1$, then $[a,\, b] = ab$. More generally, if $\{a_1,\, a_2,\, \ldots, a_r\}$ is any set of positive integers, then $(a_1,\, a_2, \ldots,\, a_r)$ divides $[a_1,\, a_2, \ldots,\, a_r]$. Here $(a_1,\, a_2, \ldots, a_r)$ and $[a_1,\, a_2, \ldots, a_r]$ denote, respectively, the gcd and lcm of the numbers $a_1,\, a_2, \ldots, a_r$.

4. Prove that no integers x, y exist satisfying $x + y = 100$ and $(x,\, y) = 3$. Do x, y exist if $x + y = 99$, $(x,\, y) = 3$?

5. Show that there exist infinitely many pairs $(x,\, y)$ such that $x + y = 72$ and $(x,\, y) = 9$.

   Hint: One choice for $(x,\, y)$ is $(63,\, 9)$. Let p be any prime strictly greater than 72. Then $(p, 72) = 1$. Let $x = 72 - 9p$ and $y = 9p$. Then $x + y = 72$ $(x, y) = 9$. As p has infinitely many choices, the result is clear.

6. If $a + b = c$, show that $(a,\, c) = 1$, iff $(b,\, c) = 1$. Hence show that any two consecutive Fibonacci numbers are coprime. (The Fibonacci numbers F_n are defined by the recursive relation $F_n = F_{n-1} + F_{n-2}$, where $F_0 = 1 = F_1$. Hence the Fibonacci sequence is $\{1, 1, 2, 3, 5, 8, \ldots\}$.

7. Find (i) gcd (2700, 15120). (ii) lcm [2700, 15120].

8. Determine integers x and y so that

(a) $180x + 72y = 36$.

(b) $605x + 96y = 67$.

9. For a positive integer n, show that there exist integers a and b such that n is a multiple of $(a,\, b) = d$ and $ab = n$ iff $d^2 \mid n$.

   (Hint: By Exercise 3 above, $(a,\, b)[a,\, b] = ab = n$ and that $(a,\, b) \mid [a,\, b]$. Hence $d^2 \,|\, n$. Conversely, if $d^2 \,|\, n$, $n = d^2c$. Now take $d = a$ and $dc = b$.)

10. Prove that $a^{mn} - 1$ is divisible by $a^m - 1$, when m and n are positive integers. Hence show that if $a^n - 1$ is prime, $a \geq 2$, then n must be prime.

11. Prove that $\sum\limits_{j=1}^{2n-1} \dfrac{1}{2j-1}$ is not an integer for $n > 1$. (Hint: Let $S = \{1, 3, \ldots, (2n-1)\}$, and let r be the largest positive integer with $3^r \in S$. Then 3^r divides no number in $S \setminus \{3^r\}$. Suppose $\frac{1}{1}+\frac{1}{3}+\cdots+\frac{1}{2n-1}$ is an integer q. Then $q - \frac{1}{3^r} = \frac{1}{3^{r-1}} \cdot \frac{a}{b}$, where a and b are integers with $(3, b) = 1$).

12. Show that the sum $s = \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ is never an integer for any $n \geq 2$.

13. Prove that if $a_n = 2^{2^n} + 1$, then $(a_n,\, a_n + 1) = 1$ for each $n \geq 1$. (Hint: Set $2^{2^n} = x$).

14. If $n = p_1^{a_1} \cdots p_r^{a_r}$ is the prime factorization of n, show that $d(n)$, the number of distinct divisors of n is $(a_1 + 1) \cdots (a_r + 1)$.

3.6 Congruences

A congruence is a division with reference to a number or a function. The congruence relation has a notational convenience that could be employed in making addition, subtraction, multiplication by constants and division in some special cases.

Definition 3.6.1:
Given integers a, b and $n\,(\neq 0)$, a is said to be congruent to b modulo n, if $a-b$ is divisible by n, that is, $a-b$ is a multiple of n. In symbols, it is denoted by $a \equiv b \pmod{n}$, and is read as "a is congruent to b modulo n." The number n is the modulus *of the congruence.*

Definition 3.6.2:
If $f(x)$, $g(x)$ and $h(x)\,(\neq 0)$ are any three polynomials with real coefficients then by $f(x) \equiv g(x) \pmod{h}(x)$, we mean that $f(x)-g(x)$ is divisible by $h(x)$ over $\mathbf{R}$, that is to say, there exists a polynomial $q(x)$ with real coefficients such that

$$f(x) - g(x) = q(x)h(x).$$

The congruence given in Definition 3.6.1 is *numerical congruence* while that given in Definition 3.6.2 is *polynomial congruence*. We now concentrate on numerical congruences. Trivially, $a \equiv b \pmod{m}$, iff $a \equiv b \pmod{-m}$. Hence we assume without loss of generality that the modulus of any numerical congruence is a positive integer.

Proposition 3.6.3: 1. $a \equiv b \pmod{m}$ iff $b \equiv a \pmod{m}$.

2. If $a \equiv b \pmod{m}$, and $b \equiv c \pmod{m}$, then $a \equiv c \pmod{m}$.

3. If $a \equiv b \pmod{m}$, then for any integer k, $ka \equiv kb \pmod{m}$. In particular, taking $k = -1$, we have $-a \equiv -b \pmod{m}$, whenever $a \equiv b \pmod{m}$.

4. If $a \equiv b \pmod{m}$, and $c \equiv d \pmod{m}$, then

   (a) $a \pm c \equiv b \pm d \pmod{m}$, and

   (b) $ac \equiv bd \pmod{m}$.

Proof. We prove only 3 and 4; the rest follow immediately from the definition.

3. If $a \equiv b \pmod{m}$, then $a - b$ is divisible by m, and hence so is $k(a - b) = ka - kb$. Thus $ka \equiv kb \pmod{m}$.

4. If $a - b$ and $c - d$ are multiples of m, say, $a - b = km$ and $c - d = k'm$ for integers k and k', then $(a + c) - (b + d) = (a - b) + (c - d) = km + k'm = (k + k')m$, a multiple of m. This means that $a + c \equiv b + d \pmod{m}$. Similarly $a - c \equiv b - d \pmod{m}$. Next, if $a \equiv b \pmod{m}$, then by (3), $ac \equiv bc \pmod{m}$. But then $c \equiv d \pmod{m}$ gives $bc \equiv bd \pmod{m}$. Hence $ac \equiv bd \pmod{m}$ by (1).

■

Proposition 3.6.4:
If $ab \equiv ac \pmod{m}$, and $(a,\, m) = 1$, then $b \equiv c \pmod{m}$.

Proof.

$$ab \equiv ac \pmod{m} \implies m|a(b - c)$$

$$\implies m|(b - c) \text{ as } (a, m) = 1 \text{ (by Proposition 3.3.2)}$$

$$b \equiv c \pmod{m}.$$

■

Corollary 3.6.5:
If $ab \equiv ac \pmod{m}$, then $b \equiv c$ (mod $\frac{)}{m}d$, where $d = (a,\, m)$.

Proof. As $d = (a,\, m)$, $d \mid a$ and $d \mid m$. Therefore, $a = da'$ and $m = dm'$, where $(a',\, m') = 1$. Then $ab \equiv ac \pmod{m}$ gives that

$$da'b \equiv da'c \pmod{m}, \tag{3.14}$$

$$\text{that is} \quad da'b \equiv da'c \pmod{dm'} \tag{3.15}$$

and therefore $a'b \equiv a'c$. (mod m'). But $(a',\, m') = 1$. Hence by Proposition 3.6.4, $b \equiv c \pmod{m'}$. ■

Proposition 3.6.6:
If $(a, m) = (b, m) = 1$, then $(ab, m) = 1$.

Proof. Suppose $(ab, m) = d > 1$ and p is a prime divisor of d. Then $p \mid m$ and $p \mid ab$. But $p \mid ab$ means that $p \mid a$ or $p \mid b$ as p is prime. If $p \mid a$, then $(a,\, m) \geq p > 1$, while if $p \mid b$, $(b,\, m) \geq p > 1$. This shows that $(ab,\, m) \geq p > 1$, a contradiction. Hence $(ab, m) = 1$. ■

Proposition 3.6.7:
If $ax \equiv 1 \pmod{m}$ and $(a,\, m) = 1$, then $(x,\, m) = 1$.

Proof. Suppose $(x,\, m) = d > 1$. Let p be a prime divisor of d. Then $p \mid x$ and $p \mid m$. This however means, since $ax - 1 = km$ for some $k \in \mathbf{Z}$, that $p \mid 1$, a contradiction. ■

Proposition 3.6.8:
If $a \equiv b \pmod{m_i}$, $1 \leq i \leq r$, then $a \equiv b \pmod{[m_1, \ldots, m_r]}$, where $[m_1, \ldots, m_r]$ stands for the lcm of $m_1, \ldots, m_r$.

Proof. The hypothesis implies that $a - b$ is a common multiple of $m_1, \ldots, m_r$ and hence it is a multiple of the least common multiple of $m_1, \ldots m_r$. (Because if $a - b = \alpha_i m_i$, $1 \leq i \leq r$, and m_i has the prime factorization $m_i = p_1^{\alpha_1^i} p_2^{\alpha_2^i} \cdots p_t^{\alpha_t^i}$, $1 \leq i \leq r$, then $p_j^{\alpha_j^i} \mid m_i$ for each j, in $1 \leq j \leq t$ and for each i, $1 \leq i \leq r$. (Here the exponents α_j^i are nonnegative integers. This enables us to take the same prime factors $p_1, \ldots, p_t$ for all the m_i's. See Section 3.4.) Hence m_i is divisible by $\left(p_j^{\max_i \alpha_j^i}\right)$ for each j in $1 \leq j \leq t$. Thus each of these t numbers divides $a-b$ and they are pairwise coprime. Hence $a - b$ is divisible by their product. But their product is precisely the lcm of $m_1, \ldots, m_r$. (See Section 3.4.) ■

3.7 Complete System of Residues

A number b is called a residue of a number a modulo m if $a \equiv b \pmod{m}$. Obviously, b is a residue of b modulo m.

Definition 3.7.1:
Given a positive integer m, a set $S = \{x_1, \ldots, x_m\}$ of m numbers is called a complete system of residues modulo m *if for any integer x, there exists a unique $x_i \in S$ such that $x \equiv x_i \pmod{m}$.*

We note that no two numbers x_i and x_j, $i \neq j$, in S are congruent modulo m. For if $x_i \equiv x_j \pmod{m}$, then since $x_i \equiv x_i \pmod{m}$ trivially, we have a contradiction to the fact that S is a complete residue system modulo m. Conversely, it is easy to show that any set of m numbers, no two of which are congruent modulo m, forms a complete residue system modulo m. In particular, the set $\{0, 1, 2, \ldots, m-1\}$ is a complete residue system modulo m.

Next, suppose that $(x, m) = 1$ and $x \equiv x_i \pmod{m}$. Then x_i is also prime to m. (If x_i and m have a common factor $p > 1$ then $p \mid x$ as $x - x_i = km$, $k \in \mathbf{Z}$. This however means that $(x, m) \geq p > 1$, a contradiction to our assumption.) Thus if $S = \{x_1, \ldots, x_m\}$ is a complete system of residues modulo m and $x \equiv x_i \pmod{m}$, then $(x, m) = 1$ iff $(x_i, m) = 1$. Then deleting the numbers x_j of S that are not coprime to m, we get a subset S' of S consisting of a set of residues modulo m each of which is relatively prime to m. Such a system is called a *reduced system of residues modulo m*. For instance, taking $m = 10$, $\{0, 1, 2, \ldots, 9\}$ is a complete system of residues modulo 10, while $S' = \{1, 3, 7, 9\}$ is a reduced system of residues modulo 10. The numbers in S' are all the numbers that are less than 10 and prime to 10.

Euler's ϕ-Function

Definition 3.7.2:
The Euler function $\phi(n)$ *(also called the* totient function*) is defined to be the number of positive integers less than n and prime to n. It is also the cardinality of a reduced residue system modulo n.*

We have seen earlier that $\phi(10) = 4$. We note that $\phi(12)$ is also equal to 4 since 1, 5, 7, 11 are all the numbers less than 12 and prime to 12. If p is a prime, then all the numbers in $\{1, 2, \ldots, p-1\}$ are less than p and prime to p and so $\phi(p) = p-1$.

We now present Euler's theorem on the ϕ-function.

Theorem 3.7.3 (Euler)**:**
If $(a,\, n) = 1$, *then*

$$a^{\phi(n)} \equiv 1 \pmod{n}.$$

Proof. Let $\left\{r_1, \ldots, r_{\phi(n)}\right\}$ be a reduced residue system modulo n. Now $(r_i,\, n) = 1$ for each i, $1 \leq i \leq \phi(n)$. Further, as $(a,\, n) = 1$, by Proposition 3.6.6, $(ar_i,\, n) = 1$. Moreover, if $i \neq j$, $ar_i \not\equiv ar_j \pmod{n}$. For, $ar_i \equiv ar_j \pmod{n}$ implies (as $(a,\, n) = 1$), by virtue of Proposition 3.6.4, that $r_i \equiv r_j \pmod{n}$, a contradiction to the fact that $\left\{r_1, \ldots, r_{\phi(n)}\right\}$ is a reduced residue system modulo n. Hence $\left\{ar_i, \ldots, ar_{\phi(n)}\right\}$ is also a reduced residue system modulo n and

$$\prod_{i=1}^{\phi(n)} (ar_i) \equiv \left(\prod_{j=1}^{\phi(n)} r_j\right) \pmod{n}.$$

This gives that $a^{\phi(n)} \prod_{i=1}^{\phi(n)} r_i \equiv \prod_{j=1}^{\phi(n)} r_j \pmod{n}$. Further $(r_i,\, n) = 1$ for each $i = 1,\, 2,\, \ldots, \phi(n)$ gives that $\left(\prod_{i=1}^{\phi(n)} r_i,\, n\right) = 1$ by Proposition 3.6.6. Consequently, by Proposition 3.6.4,

$$a^{\phi(n)} \equiv 1 \pmod{n}. \qquad \blacksquare$$

Corollary 3.7.4 (Fermat's little theorem)**:**
If n *is a prime and* $(a,\, n) = 1$, *then* $a^{n-1} \equiv 1 \pmod{n}$.

Proof. If n is a prime, then $\phi(n) = n - 1$. Now apply Euler's theorem (Theorem 3.7.3). ■

We see more properties of the Euler function $\phi(n)$ in Section 3.11. Another interesting theorem in elementary number theory is Wilson's theorem.

Theorem 3.7.5:
If $u \in [1,\, m-1]$ *is a solution of the congruence* $ax \equiv 1 \pmod{m}$, *then all the solutions of the congruence are given by* $u + km$, $k \in \mathbf{Z}$.

Proof. Clearly, $u+km$, $k \in \mathbf{Z}$, is a solution of the congruence $ax \equiv 1 \pmod{m}$ because $a(u+km) = au + akm \equiv au \equiv 1 \pmod{m}$. Conversely, let $ax_0 \equiv 1 \pmod{m}$. Then $a(x_0 - u) \equiv 0 \pmod{m}$, and therefore by Proposition 3.6.7, $(x_0 - u) \equiv 0 \pmod{m}$ as $(a, m) = 1$ in view of $au \equiv 1 \pmod{m}$. Hence $x_0 = u + km$ for some $k \in \mathbf{Z}$. ■

Theorem 3.7.7 (Wilson's theorem)**:**
If p is a prime, then

$$(p-1)! \equiv -1 \pmod{p}.$$

Proof. The result is trivially true if $p = 2$ or 3. So let us assume that prime $p \geq 5$. We look at $(p-1)! = 1 \cdot 2 \cdots (p-1)$. Now

$$1 \equiv 1 \pmod{p}, \quad \text{and}$$
$$p - 1 \equiv -1 \pmod{p}.$$

Hence it is enough if we prove that

$$2 \cdot 3 \cdots (p-2) \equiv 1 \pmod{p},$$

since the multiplication of the three congruences (See Proposition 3.6.3) will yield the required result.

Now, as $p\,(\geq 5)$ is an odd prime, the cardinality of L is even, where $L = \{2, 3, \ldots, p-2\}$. For each $i \in L$, by virtue of Corollary 3.3.5, there exists a unique j, $1 \leq j \leq p-1$ such that $ij \equiv 1 \pmod{p}$. Now $j \neq 1$, and $j \neq p-1$. $\big($If $j = p-1$, then $ij = i(p-1) \equiv -i \pmod{p}$ and therefore $-i \equiv 1 \pmod{p}$. This means that $p \mid (i+1)$. This is not possible as $i \in L\big)$. Also $ij = ji$. Moreover $j \neq i$ since $j = i$ implies that $i^2 \equiv 1 \pmod{p}$ and therefore $p \mid (i-1)$ or $p \mid (i+1)$. However this is not possible as this will imply that $i = 1$ or $(p-1)$. Thus each $i \in L$ can be paired off with a unique $j \in L$ such that $ij \equiv 1 \pmod{p}$. In this way we get $\dfrac{p-3}{2}$ congruences. Multiplying these $\dfrac{p-3}{2}$ congruences, we get

$$2 \cdot 3 \cdots (p-2) \equiv 1 \cdots 1 \equiv 1 \pmod{p} \tag{3.16}$$

■

Example 3.7.8:
As an application of Wilson's theorem, we prove that $712! + 1 \equiv 0 \pmod{719}$.

Proof. Since 719 is a prime, Wilson's theorem implies that $718! + 1 \equiv 0 \pmod{719}$. We now rewrite $718!$ in terms of 712 as

$$\begin{aligned}718! &= (712)! \times 713 \times 714 \times 715 \times 716 \times 717 \times 718\\ &= 712!(719-6)(719-5)\cdots(719-1)\\ &= 712!\left(M(719)+6!\right).\end{aligned}$$

$\big(M(719)$ stands for a multiple of 719.$\big)$

$$\begin{aligned}&\equiv \big(712! \times 6!\big) \pmod{719}\\ &\equiv \big(712! \times 720\big) \pmod{719}.\\ &\equiv \big(712! \times (719+1)\big) \pmod{719}.\\ &\equiv \big((712! \times 719) + 712!\big) \pmod{719}\\ &\equiv \big(712!\big) \pmod{719}.\end{aligned}$$

Thus $712! + 1 \equiv 718! + 1 \equiv 0 \pmod{719}$ ■

If $a \equiv b \pmod{m}$, then by Proposition 3.6.3, $a^2 \equiv b^2 \pmod{m}$, and so $a^3 \equiv b^3 \pmod{m}$ and so on. In general, $a^r \equiv b^r \pmod{m}$ for every positive integer r and hence, again by Proposition 3.6.3, $ta^r \equiv tb^r$ for every integer t. In particular, if $f(x) = a_0 + a_1x + \cdots + a_nx^n$ is any polynomial in x with integer coefficients, then $f(a) = a_0 + a_1 \cdot a + a_2 \cdot a^2 + \cdots + a_n \cdot a^n \equiv a_0 + a_1 \cdot b + a_2 \cdot b^2 + \cdots + a_n \cdot b^n = f(b) \pmod{m}$. We state this result as a theorem.

Theorem 3.7.9:
If $f(x)$ is a polynomial with integer coefficients, and $a \equiv b \pmod{m}$, then $f(a) \equiv f(b) \pmod{m}$.

3.8 Linear Congruences and the Chinese Remainder Theorem

Let $f(x)$ be a polynomial with integer coefficients. By a *solution* of the polynomial congruence

$$f(x) \equiv 0 \pmod{m}, \tag{3.17}$$

we mean an integer x_0 with $f(x_0) \equiv 0 \pmod{m}$. If $x_0 \equiv y_0 \pmod{m}$, by Theorem 3.7.9, $f(x_0) \equiv f(y_0) \pmod{m}$, and hence y_0 is also a solution of the congruence (3.17). Hence, when we speak of all the solutions of (3.17), we consider congruent solutions as forming a class. Hence by the *number* of solutions of a polynomial congruence, we mean the number of distinct congruence classes of solutions. Equivalently, it is the number of incongruent solutions modulo m of the congruence. Since any set of incongruent numbers modulo m is of cardinality at most m, the number of solutions of any polynomial congruence modulo m is at most m.

The congruence (3.17) is linear if $f(x)$ is a linear polynomial. Hence a linear congruence is of the form

$$ax \equiv b \pmod{m} \tag{3.18}$$

It is not always necessary that a congruence modulo m has m solutions. In fact, a congruence modulo m may have no solution or less than m solutions. For instance, the congruence $2x \equiv 1 \pmod{6}$ has no solution since $2x - 1$ is an odd integer and hence cannot be a multiple of 6. The congruence $x^3 \equiv 1 \pmod{7}$ has exactly 3 solutions given by $x \equiv 1, 2, 4 \pmod{7}$.

Theorem 3.8.1:
Let $(a, m) = 1$ and b an integer. Then the linear congruence

$$ax \equiv b \pmod{m} \tag{3.19}$$

has exactly one solution.

Proof. (See also Corollary 3.3.5.) The numbers $1, 2, \ldots, m$ form a complete residue system modulo m. Hence, as $(a, m) = 1$, the numbers $a \cdot 1$, $a \cdot 2$, $\cdots$, $a \cdot m$ also form a complete residue system modulo m. Now any integer is congruent modulo m to a unique integer in a complete residue system modulo m (by definition of a complete residue system). Hence b is congruent modulo m to a unique $a \cdot i$, $1 \leq i \leq m$. Thus there exists a unique $x \in \{1,\, 2,\, \ldots m\}$ such that

$$ax \equiv b \pmod{m} \tag{3.20}$$

■

If $(a,\, m) = 1$, taking $b = 1$ in Theorem 3.8.1, we see that there exists a unique x in $\{1, 2, \ldots m\}$ such that

$$ax \equiv 1 \pmod{m}$$

This unique x is called the *reciprocal* of a modulo m.

We have seen in Theorem 3.8.1 that if $(a,\, m) = 1$, the congruence has exactly one solution. What happens if $(a,\, m) = d$?

Theorem 3.8.2:
Let $(a,\, m) = d$. Then the congruence

$$ax \equiv b \pmod{m} \tag{3.21}$$

has a solution iff $d \mid b$. If $d \mid b$, the congruence has exactly d solutions. The d solutions are given by

$$x_0,\, x_0 + m/d,\, x_0 + 2m/d,\, \ldots, x_0 + (d-1)m/d, \tag{3.22}$$

where x_0 is the unique solution in $\{1, 2, \ldots, m/d\}$ of the congruence

$$\frac{a}{d}x \equiv \frac{b}{d} \quad (\text{mod } \frac{m}{d}).$$

Proof. Suppose x_0 is a solution of the congruence (3.21). Then $ax_0 = b + km$, $k \in \mathbf{Z}$. As $d \mid a$ and $d \mid m$, $d \mid b$. Conversely, if $d \mid b$, let $b = db_0$. Further, if $a = da_0$ and $m = dm_0$, the congruence (3.21) becomes

$$a_0 dx \equiv db_0 \pmod{dm_0},$$

and therefore

$$a_0x \equiv b_0 \pmod{m_0} \tag{3.23}$$

where $(a_0, m_0) = 1$. But the latter has a unique solution $x_0 \in \{1, 2, \ldots m_0 = m/d\}$. Hence $a_0x_0 \equiv b_0 \pmod{m_0}$. So x_0 is also a solution of (3.21).

Assume now that $d \mid b$. Let y be any solution of (3.21). Then $ay \equiv b \pmod{m}$. Also $ax_0 \equiv b \pmod{m}$. Hence $ay \equiv ax_0 \pmod{m}$ so that $a_0dy \equiv a_0dx_0 \pmod{(m/d)d}$. Hence $a_0y \equiv a_0x_0 \pmod{m/d}$. As $d = (a, m)$, $(a_0, m/d) = 1$. So by Proposition 3.6.4, $y \equiv x_0 \pmod{m/d}$ and so $y = x_0 + k(m/d)$ for some integer k. But $k \equiv r \pmod{d}$ for some r, $0 \leq r < d$. This gives $k\frac{m}{d} \equiv r\frac{m}{d} \pmod{m}$. Thus $x_0 + k(m/d) \equiv x_0 + r(m/d) \pmod{m}$, $0 \leq r < d$ and so y is congruent modulo m to one of the numbers in (3.22). ∎

Chinese Remainder Theorem

Suppose there is more than one linear congruence. In general, they need not possess a common solution. (In fact, as seen earlier, even a single linear congruence may not have a solution.) The Chinese remainder theorem ensures that if the moduli of the linear congruences are pairwise coprime, then the simultaneous congruences all have a common solution. To start with, consider congruences of the form $x \equiv b_i \pmod{m_i}$.

Theorem 3.8.3:
Let $m_1, \ldots, m_r$ be positive integers that are pairwise coprime, that is, $(m_i, m_j) = 1$ whenever $i \neq j$. Let $b_1, \ldots, b_r$ be arbitrary integers. Then the system of congruences

$$\begin{aligned} x &\equiv b_1 \pmod{m_1} \\ &\vdots \\ x &\equiv b_r \pmod{m_r} \end{aligned}$$

has exactly one solution modulo $M = m_1 \cdots m_r$.

Proof. Let $M_i = M/m_i$, $1 \le i \le r$. Then, by hypothesis,

$$(M_i,\, m_i) = \Big(\prod_{\substack{k=1\\k\ne i}}^{r} m_k,\, m_i\Big) = 1.$$

Hence each M_i has a unique reciprocal M_i' modulo m_i, $1 \le i \le r$. Let

$$x = b_1 M_1 M_1' + \cdots + b_r M_r M_r'. \tag{3.24}$$

Now m_i divides each M_j, $j \ne i$. Hence, taking modulo m_i on both sides of (3.24), we get

$$\begin{aligned} x &\equiv b_i M_i M_i' \pmod{m_i} \\ &\equiv b_i \pmod{m_i} \quad \text{as } M_i M_i' \equiv 1 \pmod{m_i}. \end{aligned}$$

Hence x is a common solution of all the r congruences.

We now show that x is unique modulo M. In fact, if y is another common solution, we have

$$y \equiv b_i \pmod{m_i}, \quad 1 \le i \le r,$$

and, therefore,

$$x \equiv y \pmod{m_i}, \quad 1 \le i \le r.$$

This means, as the m_i's are pairwise coprime, that

$$x \equiv y \pmod{M}. \qquad \blacksquare$$

We now present the general form of the Chinese remainder theorem.

Theorem 3.8.4 (Chinese remainder theorem)**:**
Let $m_1, \ldots, m_r$ be positive integers that are pairwise coprime. Let $b_1, \ldots, b_r$ be arbitrary integers and let integers $a_1, \ldots, a_r$ satisfy $(a_i,\, m_i) = 1$, $1 \le i \le r$. Then the system of congruences

$$\begin{aligned} a_1 x &\equiv b_1 \pmod{m_1} \\ &\vdots \\ a_r x &\equiv b_r \pmod{m_r} \end{aligned}$$

has exactly one solution modulo $M = m_1 m_2 \cdots m_r$.

Proof. As $(a_i,\, m_i) = 1$, a_i has a unique reciprocal a_i' modulo m_i so that $a_i a_i' \equiv 1 \pmod{m_i}$. Then the congruence $a_i x \equiv b_i \pmod{m_i}$ is *equivalent to* $a_i' a_i x \equiv a_i' b_i \pmod{m_i}$, that is, to $x \equiv a_i' b_i \pmod{m_i}$, $1 \le i \le r$. By Theorem 3.8.3, these congruences have a common unique solution x modulo $M = m_1 \cdots m_r$. Because of the equivalence of the two sets of congruences, x is a common solution to the given set of r congruences as well. ■

3.9 Lattice Points Visible from the Origin

A lattice point of the plane is a point both of whose Cartesian coordinates (with reference to a pair of rectangular axes) are integers. For example, (2, 3) is a lattice point while (2.5, 3) is not. A lattice point $(a,\, b)$ is said to be visible from another lattice point $(a',\, b')$ if the line segment joining $(a',\, b')$ with $(a,\, b)$ contains no other lattice point. In other words, it means that there is no lattice point that obstructs the view of $(a,\, b)$ from $(a',\, b')$. It is clear that $(\pm 1,\, 0)$ and $(0,\, \pm 1)$ are the only lattice points on the coordinate axes visible from the origin. Further, the point (2, 3) is visible from the origin, but (2, 2) is not (see Figure 3.1). Hence we consider here lattice points $(a,\, b)$ not on the coordinate axes but visible from the origin. Without loss of generality, we may assume that $a \ge 1$ and $b \ge 1$.

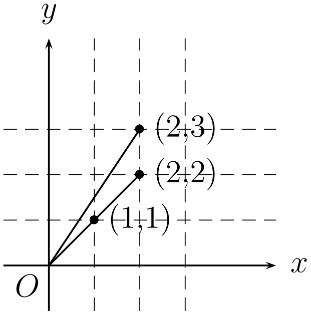


Figure 3.1: Lattice points visible from the origin

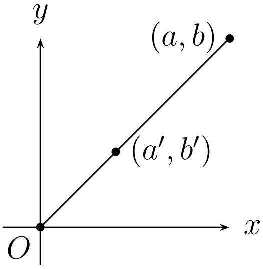


Figure 3.2: Illustration

Lemma 3.9.1:
The lattice point $(a,\, b)$ (not belonging to any of the coordinate axes) is visible from the origin iff the $gcd(a,\, b) = 1$.

Proof. As mentioned earlier, assume without loss of generality, that $a \geq 1$ and $b \geq 1$. A similar argument applies in the other cases.

Suppose $gcd(a,\, b) = 1$. Then $(a,\, b)$ must be visible from the origin. If not, there exists a lattice point $(a',\, b')$ with $a' < a$ and $b' < b$ in the segment joining $(0,\, 0)$ with $(a,\, b)$. (See Figure 3.2.) Then $\frac{b}{a} = \frac{b'}{a'}$ (= slope of the line joining $(0,\, 0)$ with $(a,\, b)$) so that $ba' = b'a$. Now $a \mid b'a$ and so $a \mid ba'$. But $(a,\, b) = 1$ and hence by Proposition 3.3.2, $a \mid a'$. But this is a contradiction since $a' < a$.

Next assume that $gcd(a,\, b) = d > 1$. Then $a = da'$, $b = db'$ for positive integers a', b'. Then the lattice point $(a',\, b')$ lies on the segment joining $(0,\, 0)$ with $(a,\, b)$, and since $a' < a$ and $b' < b$, $(a,\, b)$ is not visible from the origin. ■

Corollary 3.9.5:
The lattice point $(a,\, b)$ is visible from the lattice point $(c,\, d)$ iff $(a-c,\, b-d) = 1$.

Proof. Shift the origin to $(c,\, d)$ through parallel axes. Then, the new origin is $(c,\, d)$ and the new coordinates of the original point $(a,\, b)$ with respect to the new axes are $(a-c,\, b-d)$. Now apply Lemma 3.9.1. ■

We now give an application of the Chinese remainder theorem to the set of lattice points visible from the origin.

Theorem 3.9.6:
The set of lattice points visible from the origin contains arbitrarily large square gaps. That is, given any positive integer k, there exists a lattice point $(a,\, b)$ such that none of the lattice points

$$(a+r,\, b+s), \quad 1 \leq r \leq k, \quad 1 \leq s \leq k,$$

is visible from the origin.

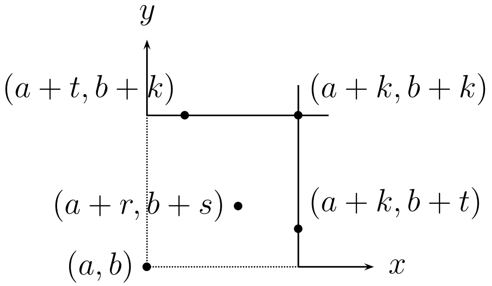


Figure 3.3: Lattice points not visible from (0, 0)

Proof. Let $\{p_1, p_2, \ldots\}$ be the sequence of primes. Given the positive integer k, construct a k-by-k matrix M whose first row is the sequence of first k primes $p_1, p_2, \ldots, p_k$, the second row is the sequence of next k primes, namely $p_{k+1}, \ldots, p_{2k}$, and so on.

$$M : \begin{bmatrix} p_1 & p_2 & \cdots & p_s & \cdots & p_k \\ p_{k+1} & p_{k+2} & \cdots & p_{k+s} & \cdots & p_{2k} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \end{bmatrix}$$

Let m_i (resp. M_i) be the product of the k primes in the i-th row (resp. column) of M. Then for $i \neq j$, $(m_i, m_j) = 1$ and $(M_i, M_j) = 1$ because in the products m_i and m_j (resp. M_i and M_j), there is no repetition of any prime. Now by Chinese remainder theorem, the set of congruences

$$\begin{aligned} x &\equiv -1 \pmod{m_1} \\ x &\equiv -2 \pmod{m_2} \\ &\vdots \\ x &\equiv -k \pmod{m_k} \end{aligned}$$

has a unique common solution a modulo $m_1 \cdots m_k$. Similarly, the system

$$y \equiv -1 \pmod{M_1}$$

$$\begin{aligned} y &\equiv -2 \pmod{M_2} \qquad\qquad \vdots \\ y &\equiv -k \pmod{M_k} \end{aligned}$$

has a unique common solution b modulo $M_1 \cdots M_k$. Then $a \equiv -r \pmod{m_r}$, and $b \equiv -s \pmod{M_s}$, $1 \leq r,\ s \leq k$. Hence $a + r$ is divisible by the product of all the primes in the r-th row of M, and similarly $b + s$ is divisible by the product of all the primes in the s-th column of M. Hence the prime common to the r-th row and s-th column of M divides both $a + r$ and $b + s$. In other words $(a + r,\, b + s) \neq 1$. So by Lemma 3.9.1, the lattice point $(a + r,\, b + s)$ is not visible from the origin. Now any lattice point inside the square is of the form $(a+r,\, b+s)$, $0 < r < k$, $0 < s < k$. (For $1 \leq r \leq k - 1$, $1 \leq s \leq k - 1$, we get lattice points inside the square while for $r = k$, or $s = k$, we get lattice points on the boundary of the square.) ■

3.10 Exercises

1. If a is prime to m, show that $0 \cdot a,\ 1 \cdot a,\ 2 \cdot a, \ldots, (m - 1) \cdot a$ form a complete system of residues $\pmod{m}$. Hence show that for any integer b and for $(a,\, m) = 1$, the set $\{b,\ b + a,\ b + 2a, \ldots, b + (m - 1)a\}$ forms a complete system of residues modulo m.

2. Show that the sum of the numbers less than n and prime to n is $\frac{1}{2}n\phi(n)$.

3. Prove that $18! + 1$ is divisible by 437.

4. If p and q are distinct primes, show that $p^{q-1} + q^{p-1} - 1$ is divisible by pq.

5. If p is a prime, show that $2(p - 3)! + 1 \equiv 0 \pmod{p}$. [Hint: $(p - 1)! = (p - 3)!(p - 2)(p - 1) \equiv 2(p - 3)!$].

6. Solve: $5x \equiv 2 \pmod{6}$. [Hint: See Theorem 3.8.1].

7. Solve: $3x \equiv 2 \pmod 6$. [Hint: Apply Theorem 3.8.2].

8. Solve: $5x \equiv 10 \pmod 7$15. [Hint: Apply Chinese remainder theorem].

9. Solve the simultaneous congruences:

   (a) $x \equiv 1 \pmod 3$, $x \equiv 2 \pmod 4$; $2x \equiv 1 \pmod 5$.

   (b) $2x \equiv 1 \pmod 3$, $3x \equiv 1 \pmod 4$; $x \equiv 2 \pmod 5$.

10. Prove the converse of Wilson's theorem, namely, if $(n-1)!+1 \equiv 0 \pmod n$, then n is prime. [Hint: Prove by contradiction]. ■

3.11 Some Arithmetical Functions

Arithmetical functions are real or complex valued functions defined on $\mathbf{N}$, the set of positive integers. We have already come across the arithmetical function $\phi(n)$, Euler's totient function. In this section, we look at the basic properties of the arithmetical functions $\phi(n)$, the Möbius function $\mu(n)$ and the divisor function $d(n)$.

Definition 3.11.1:
An arithmetical function *or a number-theoretical function is a function whose domain is the set of natural numbers and codomain is the set of real or complex numbers.*

The Möbius function $\mu(n)$

Definition 3.11.2:
The Möbius function $\mu(n)$ is defined as follows:

$$\mu(1) = 1;$$

If $n > 1$, and $n = p_1^{a_1} \cdots p_r^{a_r}$ is the prime factorization of n,

$$\mu(n) = \begin{cases} (-1)^r & \text{if } a_1 = a_2 = \cdots = a_r = 1 \\ \text{(that is, if } n \text{ is a product of } r \text{ distinct primes)} \\ 0 \quad \text{otherwise (that is, } n \text{ has a square factor} > 1). \end{cases}$$

For instance, if $n = 5 \cdot 11 \cdot 13 = 715$, a product of three distinct primes, $\mu(n) = (-1)^3 = -1$, while if $n = 5^2 \cdot 11 \cdot 13$ or $n = 7^3 \cdot 13$, $\mu(n) = 0$. Most of these arithmetical functions have nice relations connecting n and the divisors of n.

Theorem 3.11.2:
If $n \geq 1$, we have

$$\sum_{d|n} \mu(d) = \left\lfloor \frac{1}{n} \right\rfloor = \begin{cases} 1 & \textit{if } n = 1 \\ 0 & \textit{if } n > 1. \end{cases} \tag{3.25}$$

(Recall that for any real number x, $\lfloor x \rfloor$ stands for the floor of x, that is, the greatest integer not greater than x; for example, $\lfloor \frac{15}{2} \rfloor = 7$.)

Proof. If $n = 1$, $\mu(1)$ is, by definition, equal to 1 and hence the relation (3.25) is valid. Now assume that $n > 1$ and that $p_1, \ldots, p_r$ are the distinct prime factors of n. Then any divisor of n is of the form $p_1^{a_1} \cdots p_r^{a_r}$, where each $a_i \geq 0$ and hence the divisors of n for which the μ-function has nonzero values are the numbers in the set $\{p_1^{\sigma_1} \cdots p_r^{\sigma_r}\colon\ \sigma_i = 0 \text{ or } 1,\ 1 \leq i \leq r\} = \{1;\, p_1, \ldots, p_r;\, p_1p_2, p_1p_3, \ldots, p_{r-1}p_r;\, \ldots;\, p_1p_2 \cdots p_r\}$. Now $\mu(1) = 1$; $\mu(p_i) = (-1)^1 = -1$; $\mu(p_ip_j) = (-1)^2 = 1$; $\mu(p_ip_jp_k) = (-1)^3 = -1$ and so on. Further the number of terms of the form p_i is $\binom{n}{1}$, of the form p_ip_j is $\binom{n}{2}$, and so on. Hence if $n > 1$,

$$\begin{aligned} \sum_{d|n} \mu(d) &= 1 - \binom{n}{1} + \binom{n}{2} - \cdots + (-1)^r \binom{n}{r} \\ &= (1-1)^r = 0 \end{aligned}$$

■

A relation connecting ϕ and μ

Theorem 3.11.3:
If $n \geq 1$, we have

$$\phi(n) = \sum_{d|n} \mu(d)\frac{n}{d} = n\sum_{d|n}\frac{\mu(d)}{d} \tag{3.26}$$

Proof. If $(n,\, k) = 1$, then $\left\lfloor \frac{1}{(n,k)} \right\rfloor = \left\lfloor \frac{1}{1} \right\rfloor = 1$, while if $(n,\, k) > 1$, $\left\lfloor \frac{1}{(n,k)} \right\rfloor = \lfloor \text{a positive number less than } 1 \rfloor = 0$. Hence

$$\phi(n) = \sum_{k=1}^{n} \left\lfloor \frac{1}{(n,\, k)} \right\rfloor.$$

Now replacing n by $(n,\, k)$ in Theorem 3.11.2, we get

$$\sum_{d|(n,k)} \mu(d) = \left\lfloor \frac{1}{(n,\, k)} \right\rfloor.$$

Hence

$$\begin{aligned} \phi(n) &= \sum_{k=1}^{n} \sum_{d|(n,k)} \mu(d) \\ &= \sum_{k=1}^{n} \sum_{\substack{d|n \\ d|k}} \mu(d). \end{aligned} \tag{3.27}$$

For a fixed divisor d of n, we must sum over all those k in the range $1 \leq k \leq n$ which are multiples of d. Hence if we take $k = qd$, then $1 \leq q \leq n/d$. Therefore (3.27) reduces to

$$\begin{aligned} \phi(n) &= \sum_{d|n} \sum_{q=1}^{n/d} \mu(d) \\ &= \sum_{d|n} \mu(d) \sum_{q=1}^{n/d} 1 = \sum_{d|n} \mu(d)\frac{n}{d} \end{aligned}$$

■

Theorem 3.11.4:
If $n \geq 1$, we have $\sum_{d|n} \phi(d) = n$.

Proof. For each divisor d of n, let $A(d)$ denote those numbers k, $1 \leq k \leq n$, such that $(k,\, n) = d$. Clearly, the sets $A(d)$ are pairwise disjoint and their union is the set $\{1,\, 2,\, \ldots, n\}$. (For example, if $n = 6$, $d = 1,\, 2,\, 3$ and 6.) Moreover $A(1) = \{k : (k,\, n) = 1\} = \{\text{set of numbers} \leq n \text{ and prime to } n\} = \{1,\, 5\}$. Similarly, $A(2) = \{2,\, 4\}$, $A(3) = \{3\}$, $A(4) = \phi = A_5$, and $A_6 = \{6\}$. Clearly, the sets $A(1)$ to $A(6)$ are pairwise disjoint and their union is the set $\{1,\, 2,\, 3,\, 4,\, 5,\, 6\}$). Then if $|A(d)|$ denotes the cardinality of the set $A(d)$,

$$\sum_{d|n} \mid A(d) \mid = n \tag{3.28}$$

But $(k,\, n) = d$ iff $\left(\frac{k}{d}, \frac{n}{d}\right) = 1$ and $0 < k \leq n$. Hence if we set $q = \frac{k}{d}$, there is a $1{-}1$ correspondence between the elements in $A(d)$ and those integers q satisfying $0 < q \leq \frac{n}{d}$, where $(q,\, n/d) = 1$. The number of such q's is $\phi(n/d)$. Note: If $q = n/d$, then, $(q,\, n/d) = (n/d,\, n/d) = n/d = 1$ iff $d = n$. In this case, $q = 1 = \phi(1) = \phi\left(\frac{n}{d}\right)$. Thus $\mid A(d) \mid = \phi(n/d)$ and (3.28) becomes

$$\sum_{d|n} \phi(n/d) = n.$$

But this is equivalent to $\sum_{d|n} \phi(d) = n$ since as d runs through all the divisors of n, so does n/d. ■

As an example, take $n = 12$. Then d runs through 1, 2, 3, 4, 6 and 12. Now $\phi(1) = \phi(2) = 1$, $\phi(3) = \phi(4) = \phi(6) = 2$, and $\phi(12) = 4$ and $\sum_{d|n} \phi(d) = \phi(1) + \phi(2) + \phi(3) + \phi(4) + \phi(6) + \phi(12) = (1+1) + (2+2+2) + 4 = 12 = n$.

A product formula for $\phi(n)$

We now present the well-known product formula for $\phi(n)$ expressing it as a product extended over all the distinct prime factors of

n.

Theorem 3.11.5:
For $n \geq 2$, we have

$$\phi(n) = n \prod_{\substack{p|n \\ p=\text{a prime}}} \left(1 - \frac{1}{p}\right) \tag{3.29}$$

Proof. We use the formula $\phi(n) = \sum_{d|n} \mu(d)\frac{n}{d}$ of Theorem 3.11.3 for the proof. Let $p_1, \ldots, p_r$ be the distinct prime factors of n. Then

$$\prod_{\substack{p|n \\ p=\text{a prime}}} \left(1 - \frac{1}{p}\right) = \prod_{i=1}^{r} \left(1 - \frac{1}{p_i}\right) = 1 - \sum_i \frac{1}{p_i} + \sum_{i \neq j} \frac{1}{p_i p_j} - \sum \frac{1}{p_i p_j p_k} + \cdots, \tag{3.30}$$

where, for example, the sum $\sum \frac{1}{p_i p_j p_k}$ is formed by taking *distinct* prime divisors p_i, p_j and p_k of n. Now, by definition of the μ-function, $\mu(p_i) = -1$, $\mu(p_i p_j) = 1$, $\mu(p_i p_j p_k) = -1$ and so on. Hence the sum on the right side of (3.30) is equal to

$$1 + \sum_{p_i} \frac{\mu(p_i)}{p_i} + \sum_{p_i, p_j} \frac{\mu(p_i p_j)}{p_i p_j} + \cdots = \sum_{d|n} \frac{\mu(d)}{d},$$

since all the other divisors of n, that is, divisors which are not products of distinct primes, contain a square and hence their μ-values are zero. Thus

$$n \prod_{\substack{p|n \\ p=\text{a prime}}} \left(1 - \frac{1}{p}\right) = \sum_{d|n} \mu(d)\frac{n}{d} = \phi(n) \quad \text{(by Theorem 3.11.3)}$$

■

The Euler ϕ-function has the following properties.

Theorem 3.11.6:

1. *$\phi(p^r) = p^r - p^{r-1}$ for prime p and $r \geq 1$.*
2. *$\phi(mn) = \phi(m)\phi(n)(d/\phi(d))$, where $d = (m, n)$.*

3. *$\phi(mn) = \phi(m)\phi(n)$, if m and n are relatively prime.*

4. *$a \mid b$ implies that $\phi(a) \mid \phi(b)$.*

5. *$\phi(n)$ is even for $n \geq 3$. Moreover, if n has k distinct odd prime factors, then $2^k \mid \phi(n)$.*

Proof. 1. By the product formula,

$$\phi(p^r) = p^r\left(1 - \frac{1}{p}\right) = p^r - p^{r-1}.$$

2. We have $\phi(mn) = mn\prod_{\substack{p|mn \\ p=\text{a prime}}}\left(1 - \frac{1}{p}\right)$. If p is a prime that divides mn, then p divides either m or n. But then there may be primes which divide both m and n and these are precisely the prime factors of (m, n). Hence if we look at the primes that divide both m and n separately, the primes p that divide $(m, n) = d$ occur twice. Therefore

$$\begin{aligned}\prod_{p|mn}\left(1 - \frac{1}{p}\right) &= \frac{\prod_{p|m}\left(1 - \frac{1}{p}\right) \cdot \prod_{p|n}\left(1 - \frac{1}{p}\right)}{\prod_{p|(m,\,n)}\left(1 - \frac{1}{p}\right)} \\ &= \frac{\dfrac{\phi(m)}{m} \cdot \dfrac{\phi(n)}{n}}{\dfrac{\phi(d)}{d}} \quad \text{(by the product formula)} \\ &= \frac{1}{mn}\phi(m)\phi(n)\frac{d}{\phi(d)}\end{aligned}$$

This gives the required result since the term on the left is $\dfrac{1}{mn}\phi(mn)$.

3. If $(m, n) = 1$, then d in (ii) is 1. Now apply (ii).

4. If $a \mid b$, then every prime divisor of a is a prime divisor of b. Hence $a\prod_{p|a}\left(1 - \frac{1}{p}\right) \mid b\prod_{p|b}\left(1 - \frac{1}{p}\right)$. This however means that $\phi(a) \mid \phi(b)$.

5. If $n \geq 3$, either n is a power of 2, say, 2^r, $r \geq 2$, or else $n = 2^s m$, where $m \geq 3$ is odd. If $n = 2^r$, $\phi(n) = 2^{r-1}$, an even number. In the other case, by (iii), $\phi(n) = \phi(2^s)\phi(m)$. Now if $m = p_1^{k_1} \cdots p_s^{k_s}$ is the prime factorization of m, by (iii) $\phi(m) = \prod_{i=1}^s \phi(p_i^{k_i}) = \big(\text{by (i)}\big) \prod_{i=1}^s p_i^{k_i-1} \cdot (p_i - 1)$. Now each $p_i - 1$ is even and hence 2^s is a factor of $\phi(n)$.

■

There are other properties of the three arithmetical functions described above and also other arithmetical functions not described here. The interested reader can consult the books given in the reference.

We now present an application of Theorem 3.11.4.

An application

We prove that the determinant

$$\begin{vmatrix} (1,1) & (1,2) & \dots & (1,n) \\ (2,1) & (2,2) & \dots & (2,n) \\ \vdots & \vdots & \ddots & \vdots \\ (n,1) & (n,2) & \dots & (n,n) \end{vmatrix} = \phi(1)\phi(2)\cdots\phi(n).$$

Proof. Let D be the diagonal matrix

$$\begin{bmatrix} \phi(1) & 0 & \dots & 0 \\ 0 & \phi(2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & & & \phi(n) \end{bmatrix}$$

Then $\det D = \phi(1)\phi(2)\cdots\phi(n)$. Define the n by n matrix $A = (a_{ij})$ by

$$a_{ij} = \begin{cases} 1 & \text{if } i \mid j \\ 0 & \text{otherwise.} \end{cases}$$

Then A is a unit upper triangular matrix. (Recall that a square matrix $A = (a_{ij})$ is upper triangular if $a_{ij} = 0$ for $i > j$, that is, the entries below the principal diagonal of A are all zero) with all diagonal entries equal to 1.) Hence $\det A = 1 = \det A^t$. Set $S = A^t DA$. Then

$$\begin{aligned}\det S &= \det A^t \cdot \det D \cdot \det A\\ &= 1 \cdot (\phi(1)\phi(2)\cdots\phi(n)) \cdot 1\\ &= \phi(1)\phi(2)\cdots\phi(n).\end{aligned}$$

We now show that $S = (s_{ij})$, where $s_{ij} = (i,\, j)$. This would prove our statement.

Now $A^t = (b_{ij})$, where $b_{ij} = a_{ji}$. Hence if D is the matrix $(d_{\alpha\beta})$, then the $(i,\, j)$-th entry of S is given by:

$$s_{ij} = \sum_{\alpha=1}^{n}\sum_{\beta=1}^{n} b_{i\alpha} d_{\alpha\beta} a_{\beta j}.$$

Now $d_{\alpha\beta} = 0$ if $\alpha \neq \beta$, and $d_{\alpha\alpha} = \phi(\alpha)$ for each α. Therefore

$$\begin{aligned}s_{ij} &= \sum_{\alpha=1}^{n} b_{i\alpha} d_{\alpha\alpha} a_{\alpha j}\\ &= \sum_{\alpha=1}^{n} b_{i\alpha} a_{\alpha j}\phi(\alpha)\\ &= \sum_{\alpha=1}^{n} a_{\alpha i} a_{\alpha j}\phi(\alpha)\end{aligned}$$

Now by definition $a_{\alpha i} = 0$ iff $\alpha \nmid i$ and $a_{\alpha i} = 1$ if $\alpha \mid i$. Hence the nonzero terms of the last sum are given by those α that divide i as well as j. Now $\alpha \mid i$ and $\alpha \mid j$ iff $\alpha \mid (i,\, j)$. Thus $s_{ij} = \sum\limits_{\alpha \mid (i,j)} \phi(\alpha)$. But the sum on the right is, by Theorem 3.11.4, $(i,\, j)$. ■

3.12 Exercises

1. Find those n for which $\phi(n) \mid n$.

2. *An arithmetical function f is called multiplicative if it is not identically zero and $f(mn) = f(m)f(n)$ whenever $(m, n) = 1$. It is completely multiplicative if $f(mn) = f(m)f(n)$ for all positive integers m and n.* Prove the following:

   (a) ϕ is multiplicative but not completely multiplicative.

   (b) μ is multiplicative but not completely multiplicative.

   (c) If n is even, prove that $\sum_{d|n} \mu(d)\phi(d) = 0$.

3. Prove that the sum of the positive integers less than n and prime to n is $\frac{1}{2}n\phi(n)$.

4. Let $\sigma(n)$ denote the sum of the divisors of n. Prove that $\sigma(n)$ is multiplicative. Hence prove that if $n = p_1^{a_r} \cdots p_r^{a_r}$ is the prime factorization of n, then

$$\sigma(n) = \prod_{i=1}^{r} \frac{p_i^{a_i+1} - 1}{p_i - 1}.$$

3.13 The Big O Notation

The big O notation is used mainly to express an upper bound for a given arithmetical function in terms of another simpler arithmetical function.

Definition 3.13.1:
Let $f : \mathbf{N} \to \mathbf{C}$ be an arithmetical function. Then $f(n)$ is $O(g(n))$ (read big O of $g(n)$), where $g(n)$ is another arithmetical function provided that there exists a constant $K > 0$ such that

$$|f(n)| \le K|g(n)| \quad \textit{for all} \quad n \in \mathbf{N}.$$

More generally, we have the following definition for any complex-valued function.

Definition 3.13.2:
Let $f : \mathbf{R} \to \mathbf{C}$ be a complex valued function. Then $f(x) = O\big(g(x)\big)$, where $g : \mathbf{R} \to \mathbf{C}$ is another function if there exists a constant $K > 0$ such that

$$|f(x)| \le K|g(x)| \quad \textit{for each } x \textit{ in } \mathbf{R}.$$

An equivalent formulation of Definition 3.13.1 is the following.

Definition 3.13.3:
Let $f : \mathbf{N} \to \mathbf{C}$ be an arithmetical function. Then $f(n)$ is $O(g(n))$, where $g(n)$ is another arithmetical function if there exists a constant $K > 0$ such that

$$|f(n)| \le K|g(n)| \quad \textit{for all} \quad n \ge n_0, \quad \textit{for some positive integer } n_0. \tag{3.31}$$

Clearly, Definition 3.13.1 implies Definition 3.13.3. To prove the converse, assume that (3.31) holds. Choose positive numbers $c_1, c_2, \ldots, c_{n_0-1}$ such that $|f(1)| < c_1|g(1)|$, $|f(2)| < c_2|g(2)| \ldots$, $|f(n-1)| < c_{n-1}|g(n-1)|$. Let K_0 =max $(c_1, c_2, \ldots, c_{n_0-1}, K)$. Then

$$|f(n)| \le K_0|g(n)| \quad \text{for each} \quad n \in \mathbf{N}.$$

This is precisely Definition 3.13.1.

The time complexity of an algorithm is the number of bit operations required to execute the algorithm. If there is just one input, and n is the size of the input, the time complexity is a function $T(n)$ of n. In order to see that $T(n)$ is not very unwieldy, usually, we try to express $T(n) = O(g(n))$, where $g(n)$ is a known less complicated function of n. The most ideal situation is where $g(n)$ is a polynomial in n. Such an algorithm is known as a polynomial time algorithm. We now give the definition of a polynomial time algorithm where there are, not just one, but k inputs.

Definition 3.13.4:
Let $n_1, \dots, n_r$ be positive integers and let n_i be a k_i-bit integer (so that the size of n_i is k_i), $1 \leq i \leq r$. An algorithm to perform a computation involving $n_1, \dots, n_r$ is said to be a polynomial time *algorithm if there exist nonnegative integers $m_1, \dots m_r$ such that the number of bit operations required to perform the algorithm is $O(k_1^{m_1} \dots k_r^{m_r})$.*

Recall that the size of a positive integer is the number of bits in it. For instance, $8 = (1000)$ and $9 = (1001)$. So both 8 and 9 are 4 bits. In fact all numbers n such that $2^{k-1} \leq n < 2^k$ are k-bits. Taking logarithms with respect to base 2, we get $k-1 \leq \log_2 n < k$ and hence $k-1 \leq \lfloor \log_2 n \rfloor < k$, so that $\lfloor \log_2 n \rfloor = k-1$. Thus $k = 1 + \lfloor \log_2 n \rfloor$ and hence k is $O(\log_2 n)$. Thus we have proved the following result.

Theorem 3.13.5:
The size of n is $O(\log_2 n)$.

Note that in writing $O(\log n)$, the base of the logarithm is immaterial. For, if the base is b, then any number that is $O(\log_2 n)$ is $O(\log_b n)$ and vice verse. This is because $\log_2 n = \log_b n \cdot log_2 b$, and $\log_2 b$ can be absorbed in the constant K of Definition 3.13.1.

Example 3.13.1:
Let $g(n)$ be a polynomial of degree t. Then $g(n)$ is $O(n^t)$.

Proof. Let $g(n) = a_0 n^t + a_1 n^{t-1} + \dots + a_t$, $a_i \in \mathbf{R}$. Then

$$\begin{aligned} |g(n)| \leq & |a_0| n^t + |a_1| n^{t-1} + \ldots + |a_t| \\ \leq & n^t (|a_0| + |a_1| + \ldots + |a_t|) \\ = & K n^t, \quad \text{where} \quad K = \sum_{i=0}^{t} |a_i|. \end{aligned}$$

Thus $g(n)$ is $O(n^t)$. ∎

We now present two examples for a polynomial time algorithm.

We have already described (see Section 3.3) Euclid's method of computing the gcd of two positive integers a and b.

Theorem 3.13.6:
Euclid's algorithm is a polynomial time algorithm.

Proof. We show that Euclid's algorithm of computing the gcd (a, b), $a > b$, can be performed in time $O(\log^3 a)$.

Adopting the same notation as in (3.5), we have

$$r_j = q_{j+2}r_{j+1} + r_{j+2}, \quad 0 \le r_{j+2} < r_{j+1}.$$

Now the fact that $q_{j+2} \ge 1$ gives,

$$r_j \ge r_{j+1} + r_{j+2} > 2r_{j+2}.$$

Hence $r_{j+2} < \frac{1}{2}r_j$ for each j. This means that the remainder in every other step in the Euclidean algorithm is less than half of the original remainder. Hence if $a = O(2^k)$, then there are at most $k = O(\log a)$ steps in the Euclidean algorithm. Now, how about the number of arithmetic operations in each step? In Equation (3.5), the number r_j is divided by r_{j+1} and the remainder r_{j+2} is computed. Since both r_j and r_{j+1} are numbers less than a they are $O(\log a)$. Hence Equation (3.5) involves $O(\log^2 a)$ bit operations. Since there are $k = O(\log a)$ such steps, the total number of bit operations is $(O(\log^3 a))$. ■

Next we show that the modular exponentiation $a^c \pmod{m}$ for positive integers a, c and m can be performed in polynomial time. Here we can take without loss of generality that $a < m$ (because if $a \equiv a' \pmod{m}$, then $a^c \equiv (a')^c \pmod{m}$, where a' can be taken to be $< m$). We now show that $a^c \pmod{m}$ can be computed in $O(\log a \log^2 m)$ bit operations. Note that $O(\log a)$ is $O(\log m)$.

Write c in binary. Let

$$c = (b_{k-1}b_{k-2}\cdots b_1b_0) \quad \text{in the binary scale.}$$

Then $c = b_{k-1}2^{k-1} + b_{k-2}2^{k-2} + \cdots + b_0 2^0$, and therefore $a^c = a^{b_{k-1}2^{k-1}} \cdot a^{b_{k-2}2^{k-2}} \cdots a^{b_1 2} \cdot a^{b_0}$, where each $b_i = 0$ or 1. We now compute $a^c \pmod{m}$ recursively by reducing the number computed at each step by $\pmod{m}$. Set

$$\begin{aligned}
y_0 &= a^{b_{k-1}} = a\\
y_1 &= y_0^2 a^{b_{k-2}} = a^{b_{k-1}\cdot 2} a^{b_{k-2}}\\
y_2 &= y_1^2 a^{b_{k-3}} = a^{b_{k-1}2^2} a^{b_{k-2}2} a^{b_{k-3}}\\
&\vdots\\
y_{i+1} &= y_i^2 a^{b_{k-i-2}} = a^{b_{k-1}2^i} a^{b_{k-2}2^{i-1}} \cdots a^{b_{k-i-2}}\\
&\vdots\\
y_{k-1} &= y_{k-2}^2 a^{b_{k-k}} = a^{b_{k-1}2^{k-1}} a^{b_{k-2}2^{k-2}} \cdots a^{b_0}\\
&= a^{(b_{k-1}2^{k-1} + b_{k-2}2^{k-2} + \cdots + b_0)} = a^c \pmod{m}.
\end{aligned}$$

There are $k-1$ steps in the algorithm. Note that y_{i+1} is computed by squaring y_i and multiplying the resulting number by 1 if $b_{k-i-2} = 0$ or else multiplying the resulting number by a if $b_{k-i-2} = 1$. Now $y_i \pmod{m}$ being a $O(\log m)$ number, to compute y_i^2, we make $O(\log^2 m) = O(t^2)$ where $t = O(\log_2 m)$ bit operations. y_i being a t-bit, y_i^2 is a $2t$ or $(2t+1)$ bit number and so it is also an $O(t)$-bit number. Now we reduce y_i^2 modulo m, that is, we divide the $O(t)$ number y_i^2 by the $O(t)$ number m. Hence this requires an additional $O(t^2)$ bit operations. Thus in all we have performed until now $O(t^2) + O(t^2)$ bit operations, that is $O(t^2)$ bit operations. Having computed $y_i^2 \pmod{m}$, we next multiply it by a^0 or a^1. As a is $O(\log_2 m) = O(t)$, this requires $O(t^2)$ bit operations. Thus in all, computation of y_{i+1} from y_i requires $O(t^2)$ bit operations. But then there are $k-1 = O(\log_2 c)$ steps in the algorithm. Thus the number of bit operations in the computation of $a^c \pmod{m}$ is $O(\log_2 c \log_2^2 m) = O(kt^2)$. Thus the algorithm is a polynomial time algorithm. ■

Bibliography

[1] G. Andrews, Number Theory, Hindustan Publishing Corporation (India), 1992.

[2] I. Niven and H. S. Zuckerman, An Introduction to the Theory of Numbers, John Wiley and Sons, Inc., 1960.

[3] G. H. Hardy and E. M. Wright, An Introduction to the Theory of Numbers, 4^{th} ed., Oxford, Clarendon Press, 1960.

[4] T. Nagell, Introduction to Number Theory, John Wiley and Sons, New York, 1951.

[5] C. V. Hsiung, Elementary Theory of Numbers, World Scientific Pub. Co. 1992.

Chapter 4

Introduction to Graph Theory

> Modélisation: Remplacer du visible compliqué par l'invisible simple.
>
> *J. Perrin*

> Mathematics is a reflection of the real-world in the mirror of our thinking.
>
> *G. Polya*

> One picture is worth more than a long discourse.
>
> *Napoléon Bonaparte*

The chapter starts with the idea of a graph and basic definitions. An informal, intuitive introduction to NP-complete problems is given. Then multigraphs, simple graphs, degree sequences, some special graphs, graphs and subgraphs, walks, paths and cycles are introduced. Then we study graphs and puzzles, and Ramsey numbers. Finally, graph algebra is studied.

4.1 The Idea of a Graph

Intuitively, a graph is a diagram consisting of a finite set of *points* (called *vertices*) together with a finite set of *arrows*, each arrow joining a certain ordered pair of vertices. Here, vertices and arrows are undefined terms like points and lines in geometry. Many real-world situations can be abstracted by means of the concept of the graph. Graphs can be used as *mathematical models* for systems involving *binary relations.* A mathematical model is a reflection of some real-world situation. A model must mirror certain *essential features* of a real situation, but not all. A mathematical model is also a simplification that allows us to study the relevant properties and ignore the others. A model can be smaller or larger or roughly the same size as the thing it represents.

Vertices of a graph may be interpreted as cities and arcs as highways joining the corresponding pairs of cities or vertices as computers and arcs as possible direct communication between the corresponding computers.

More formally, we define a graph as follows [1]:

> The number of systems of terminology presently used in graph theory is equal, to a close approximation, to the number of graph theorists.
>
> *Richard P. Stanley*

Definition 4.1.1:
A *graph* G is an *ordered* pair $(X,\, U)$ where X is a finite set of *points* (called *vertices*), and U is a finite *unordered list/multiset* of arcs (arrows), and each arc is an *ordered* pair of vertices of G.

A lot of nomenclature and classification is associated with the theory of graphs. Since U is a *list*, an arc in U may appear several times in U. A *p-graph* (where p is an integer ≥ 0) is one, in which the same arc cannot appear more than p times in U. If G is a 1-graph, then each element of U will not occur more than once and hence U becomes the *set* of arcs of the graph G.

The number of vertices of the graph G is denoted by $n(G)$ (called the *order* of the graph) or simply n, if there is no possibility of confusion, and the number of arcs by $m(G)$ or simply m. One of the appealing features of graph theory is that graphs admit *geometrical/graphical representation*, by which we understand several of their properties. Hence the name "graph."

The following example illustrates the concept of a graph and its geometrical representation. *We often identify a graph with its geometrical representation.*

EXAMPLE 4.1.1:
Consider a graph $G = (X, U)$ where $X = \{x_1, x_2, \ldots, x_7\}$ and $U = (u_1, u_2, \ldots, u_{12})$ where $u_1 = (x_7, x_7), u_2 = (x_2, x_2), u_3 = (x_3, x_2), u_4 = (x_2, x_1), u_5 = (x_1, x_4), u_6 = (x_4, x_1), u_7 = (x_4, x_5), u_8 = (x_3, x_4), u_9 = (x_3, x_5), u_{10} = (x_6, x_7), u_{11} = (x_7, x_6), u_{12} = (x_7, x_6)$. This is a *set theoretic representation* of a graph.

The arcs u_{11} and u_{12} are *multiple arcs* since they join the same ordered pair of vertices. The arcs u_5 and u_6 are *symmetric arcs.*

The same graph can be represented geometrically as follows: (see Figure 4.1).

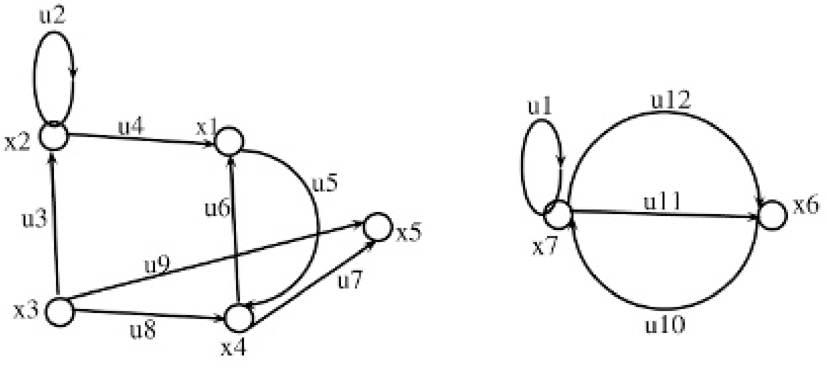


Figure 4.1: An example of a graph

In this graphical representation, vertices are depicted as small circles, and an arc as a continuous curve joining from its first co-ordinate vertex to its second co-ordinate vertex. Note that the direction of an arc is indicated at its second co-ordinate vertex. We assume that no arc intersects itself and passes through a vertex other than its first and second co-ordinate vertices. Such a representation is always possible. The arcs u_{11} and u_{12} are of the same form (x_7, x_6) and hence U is a multiset. Such arcs are called *multiple arcs*.

Loop

For an arc, its first co-ordinate vertex is simply called its *initial* vertex or *tail* and its second co-ordinate vertex its *final* vertex or *head or tip*. An arc of the form (x, x) is a *loop*, that is, a loop joins a vertex to itself. An *extremity* of an arc is either its initial vertex or its final vertex. For a loop, initial and final vertices coincide.

Let us refer to Example 4.1.1 (see Figure 4.1). It is a 2-graph, because the arcs u_{11} and u_{12} are of the same form (x_7, x_6). There are two loops, u_1 and u_2. The parameters n and m are respectively 7, 12.

Adjacency and incidence

Two arcs u_1 and u_2 are *adjacent* whenever they share an extremity in common. Two vertices are *adjacent* whenever there is an arc joining them, the direction of the arc being immaterial. For an arc $u = (x, y)$ with $x \neq y$, the arc u is said to be *incident out* of the vertex x and u is said to be *incident into* the vertex y. A loop $u = (x, x)$ is at the same time incident into the vertex x and incident out of x. A vertex is simply *incident* with an arc if it is either an initial or final vertex of the arc. Let us note that the relation of *incidence* is defined between an arc and a vertex whereas the relation of *adjacency* is defined either between two vertices or between two arcs.

Predecessor and successor

The vertex x is a *predecessor* of the vertex y or dually y is a *successor* of x. if (x, y) is an arc. The set of all successors of a vertex x is denoted by $\Gamma^+(x)$ and the set of all predecessors of x by $\Gamma^-(x)$.

In-degree, out-degree, degree

The in-degree of a vertex x, denoted by $d^{-}(x)$, is the number of arcs having the vertex x as its final vertex or geometrically it is the number of arcs leading *into* the vertex x. Dually, the out-degree of a vertex x, denoted by $d^{+}(x)$, is the number of arcs having x as its initial vertex or geometrically it is the number of arcs leading *away* from the vertex x. A loop attached to the vertex x is considered as leading into and away from the vertex x. In geometrical representation of a graph, we may omit the orientation on a loop. Finally, the degree of a vertex x, denoted by $d(x)$, is the sum of its in-degree and its out-degree, that is, $d(x) = d^{+}(x) + d^{-}(x)$. A vertex of degree zero is an *isolated vertex* and a vertex of degree one is an *end-vertex*. Let us note that each loop at the vertex x is counted twice in the calculation of $d(x)$.

Let us pause and look at an example.

Example 4.1.2:
Let us again refer to the graph of Example 4.1.1. The vertices x_3 and x_5 are adjacent whereas x_2 and x_5 are *non*-adjacent. The vertex x_3 is a predecessor of x_4 and x_2 is a successor of x_3. The arcs u_6 and u_8 are adjacent whereas u_5 and u_9 are *non*-adjacent. $d^{+}(x_2) = 2$, $d^{-}(x_2) = 2$, $d(x_7) = d^{+}(x_7) + d^{-}(x_7) = 2 + 3 = 5$. $\Gamma^{+}(x_7) = \Gamma^{-}(x_7) = \{ x_6, x_7 \}$

Until now, we have introduced several parameters of a graph namely, n, the number of vertices, m, the number of arcs, $d^{+}(x)$, the out-degree of the vertex x, and $d^{-}(x)$ the in-degree of the vertex x. Now it is time to prove a concrete result involving these parameters. Axiomatically, mathematics is concerned with the relations among undefined objects.

Before stating the theorem, let us again refer to the graph of Example 4.1.1 (see Figure 4.1). In this graph, the in-degrees of the vertices $x_1, x_2, \cdots, x_7$ are respectively $2, 2, 0, 2, 2, 1, 3$, and their sum is 12, which is equal to the number of arcs in the graph. Similarly, the sum of the out-degrees of the vertices is again equal to the number of arcs. This is not a coincidence as we see in the following elementary but important result.

The following Theorem 4.1.1 may be compared with the following analogy:

Analogy: The number of monkeys in a cage is equal to the number of heads of the monkeys, which is also equal to the number of tails of the monkeys (assuming that each monkey has exactly one head and exactly one tail!).

THEOREM 4.1.1:
In any graph, the sum of the in-degrees of all vertices is equal to the sum of the out-degrees of all vertices. Moreover, each sum equals the number of arcs of the graph. In notation,

$$\sum_{i=1}^{n} d^{+}(x_i) = \sum_{i=1}^{n} d^{-}(x_i) = m.$$

Proof. In fact, a little more is true.

Let $G = (X, U)$ be a graph with vertex set $X = \{x_1, x_2, \cdots, x_n\}$
and $U = (\, u_1, u_2, \ldots, \, u_m)$.

Let U_i^{+} be the set/multiset of arcs leading away from the vertex x_i, for $i = 1, 2, \ldots, n$. (In Example 4.1.1, $U_1^{+} = \{\, u_5 \,\}, U_2^{+} = \{\, u_2, u_4 \,\}$, etc.) By the definition of out-degree, $|U_i^{+}| = d^{+}(x_i)$. Clearly,

$$U = U_1^{+} \cup U_2^{+} \cup \cdots \cup U_n^{+} \text{ and } \; U_i^{+} \cap U_j^{+} = \emptyset \text{ for } i \neq j.$$

Let us note that U_i^{+} may be the empty set. In particular, we have

$$\begin{aligned} |U| &= |U_1^{+} \cup U_2^{+} \cup \cdots \cup U_n^{+}| \\ &= \sum_{i=1}^{n} |U_i^{+}| \\ &= \sum_{i=1}^{n} d^{+}(x_i) \end{aligned}$$

Therefore, $\sum_{i=1}^{n} d^{+}(x_i) = m$, the number of arcs of G.

Similarly, by defining the set U_i^{-} as the set of arcs leading into the vertex x_i, we can prove $\sum_{i=1}^{n} d^{-}(x_i) = m$. ■

(Out-degree,in-degree) sequence of a graph: Consider a graph G with vertices $1, 2, \ldots, n$ and suppose $d^+(i) = r_i$ and $d^-(i) = s_i$ for all $1 \leq i \leq n$. The sequence $(r_1, s_1), (r_2, s_2), \ldots, (r_n, s_n)$ is called the (out-degree,in-degree) sequence of G or simply (out,in) degree-sequence of G.

Conversely, a sequence $(r_1, s_1), (r_2, s_2), \ldots, (r_n, s_n)$ of a pair of non-negative integers is said to be a *realizable sequence* if there is a graph with n vertices $1, 2, \ldots, n$ satisfying the property: $d^+(i) = r_i$ and $d^-(i) = s_i$ for all $1 \leq i \leq n$. Otherwise, the sequence is referred to as *non-realizable*. The sequence is *p-realizable* if there is a p-graph satisfying the above property. The above theorem can be used to say that a given sequence is non-realizable.

Example 4.1.3:

Does there exist a digraph with five vertices $1, 2, 3, 4, 5$ with (out-degree, in-degree) sequence: $(2, 1), (1, 2), (3, 2), (2, 2), (3, 3)$? If the answer is "yes," construct a digraph realizing the pair of sequences. If "no," justify your answer.

Solution: Since $\sum_{i=1}^{5} d^+(i) = 11 \neq \sum_{i=1}^{5} d^-(i) = 10$, there exists *no* graph on 5 vertices realizing the sequence (by Theorem 4.1.1.)

On the other hand, the sequence $(2, 1), (2, 1), (2, 3), (1, 2)$ is 2-realizable because the graph with vertex set $1, 2, 3, 4$ and arc set

$$U = \{\,(1, 4), (1, 4), (2, 1), (2, 3), (3, 2), (3, 3), (4, 3)\,\}$$

has (out, in) degree sequence $(2, 1), (2, 1), (2, 3), (1, 2)$ and no arc is repeated more than twice.

Corollary 4.1.1.1:

In any graph, the sum of the degrees of all vertices is equal to twice the number of arcs. In symbol,

$$\sum_{i=1}^{n} d(x_i) = 2m.$$

Proof.

$$\begin{aligned}\sum_{i=1}^{n} d(x_i) &= \sum_{i=1}^{n}(d^+(x_i) + d^-(x_i)) \\ &= \sum_{i=1}^{n} d^+(x_i) + \sum_{i=1}^{n} d^-(x_i) \\ &= m + m \\ &= 2m\end{aligned}$$

by Theorem 4.1.1 ■

Corollary 4.1.1.1 may be stated in the following popular terms:

Analogy: The number of students in a classroom is equal to the total number of hands of all the students divided by 2 (assuming that each student has exactly 2 hands!).

Handshaking lemma: The above corollary is popularly known as the *handshaking lemma.* This lemma says that if in a group of people several people shake hands, then the total number of hands shaken should be an even integer, just because exactly two hands are involved in each handshake. (We identify hands of all the people in the group as the vertices of a graph and each handshake as an edge joining the corresponding vertices.)

Corollary 4.1.1.2:
In any graph, the number of vertices of odd degree, that is, vertices whose degree is an odd integer, is always even.

Proof. Let X_1 consist of all vertices of odd degree in G and let X_2 consist of all vertices of even degree. Then

$$X = X_1 \cup X_2 \text{ with } X_1 \cap X_2 = \emptyset$$

where X is the set of vertices of the graph. We have to prove that $|X_1|$ is even.

We have, $\sum_{x\in X} d(x) = \sum_{x\in X_1} d(x) + \sum_{x\in X_2} d(x)$.

But by definition of X_2, $d(x)$ is even for each $x \in X_2$. Hence by Corollary 4.1.1.1, $\sum_{x\in X_1} d(x) = 2m$—an even integer = an even integer. Since the degree of each vertex in X_1 is an odd integer, we must have $|X_1|$, an even integer. ■

The graph as defined in this section is usually called the *directed graph* or simply *digraph*, because the arcs are ordered pairs of vertices. Such graphs are used to represent real-world situations like a traffic network/city map in which certain routes are one-way for motorists. There are situations in which the directions of the arcs are irrelevant.

Examples are graphs whose vertices represent the set of people in a gathering and two vertices are joined by a straight line segment if the corresponding persons know each other. Graphs obtained by ignoring the directions of the arcs are called "undirected graphs." When the direction of the arc is ignored, the arc is called an *edge* of the graph.

Another important example concerning the famous *four-color conjecture* has now become the following theorem:

Four-Color theorem

The vertices of a graph represent the set of different countries in any map in a plane (a country consists of a single connected region) and two vertices are joined by a line segment if the corresponding countries share a common boundary, which is not a single point. The *four-color theorem* states that the vertices (countries) of such a graph can be colored using only four colors such that two vertices joined by a line segment receive different colors, like in our globe, France and Switzerland are colored differently, otherwise we would be unable to distinguish their border visually. We may color India and France with the same color since they don't share a common boundary. Nearly 125 years since its birth in the year 1852 as a conjecture and resisting attempts by many mathematicians, the conjecture has been finally proved with the massive aid of the computer (1200 hours of computer time on three computers) by Appel and Haken (see [7] for an interesting history of the four-color problem).

EXAMPLE 4.1.4 (Illustrating the four-color theorem):
In Figure 4.2, the map M consists of five countries C_1, C_2, C_3, C_4, C_5. The graph corresponding to the map M is denoted by G. In the graph G, the vertex C_1 is joined to all other vertices since the corresponding country C_1 in the map M shares a common boundary with all other countries. The vertices C_2 and C_3 are not joined by an edge because the countries C_2 and C_3 don't have a common boundary. Similarly, the countries C_4 and C_5 have no common boundary.

A coloring of the graph G with three colors B (for blue), y (for yellow), R (for red) with the property that no two vertices joined by an edge (or line segment) receive the same color (see color e-book). The reader may observe that 3 is the minimum number of colors needed to color the vertices of the graph G with the constraint imposed. In other words, three is the *chromatic number* of the graph G. Notice an important property of the graph: no two edges intersect at a point of the plane except at a vertex of the graph G. A graph admitting such a drawing is called a *planar graph*. For example, the graph of Figure 4.1 is planar (by redrawing the arc u_9 without cutting u_6 and u_7). We now state the four-color theorem:

THEOREM 4.1.2 (The four-color theorem):
The vertices of any planar graph can be colored in four or fewer colors in such a way that the vertices joined by an edge receive different colors.

4.2 An Informal and Intuitive Introduction to NP-Complete Problems

What is a *problem*? A problem is a *question* to be answered or a structure to be constructed in response to various input *parameters or free variables*. These parameters of the problem are described (like integer, real, graph, $n \times n$ matrix of real entries) without actually specifying their values.

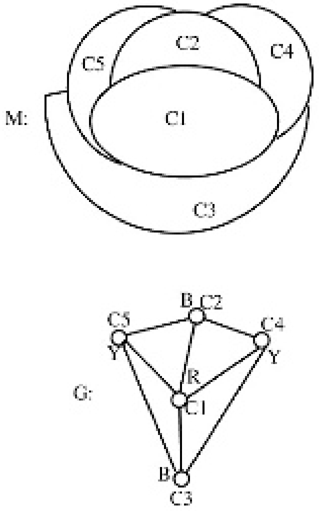


Figure 4.2: An example illustrating the four-color theorem

An *instance* of a problem is obtained by giving specific values to the parameters of the problem.

A *decision problem* is one which requires an answer, either "yes" or "no."

Example (Decision problem):

Instance: A positive integer $k \geq 2$.

Question: Is k a prime number?

Let us see an example from graph theory.

Instance: An undirected graph $G = (X, E)$.

Question: Is G 3-colorable? That is, can we *paint* all the vertices of G using 3 colors in such a way that the vertices joined by an edge receive different colors?

An *algorithm* is a step-by-step method to solve a problem. Let us recall that one of the oldest algorithms is *Euclid's algorithm* to find the greatest common divisor (gcd) of two positive integers; Knuth calls it the "granddad of algorithms." An algorithm solves a decision problem if it always gives an answer either "yes" or "no" to any instance of the decision problem.

Note that the problem of finding the gcd(m, n) of two positive integers m, n can be converted (more generally any optimization problem) into a sequence of decision problems as follows:

Since $\gcd(m, n)$ is always less than or equal to $\min(m, n)$, we start by setting $k = \min(m, n)$.

Does k divide both m and n? If the answer is "yes" then $k = \gcd(m, n)$. If the answer is "no" then we ask: Does $k - 1$ divide m and n. If the answer is "yes" then $k - 1 = \gcd(m, n)$. Otherwise, we continue asking if $k-2$ divides both m and n and so on till we get an integer d dividing both m and n. In this case, the $\gcd(m, n) = d$. Such an algorithm is called a *brute-force algorithm*, because it examines all possible divisors of m and n.

To each problem, we associate an integer n called the *size of the problem*, which is a measure of the quantity of input data (see Chapter 4). For example, the size of a graph may be its number of vertices or maximum of the number of vertices and edges or the sum of the number of vertices and edges, and the size of the matrix multiplication problem will be the largest dimension of the matrices to be multiplied.

The *time complexity of an algorithm*, expressed as a function of the size of the problem it solves, is the *maximum* number of elementary steps executed by the algorithm in solving any instance of that size.

An algorithm is said to be a *polynomial time algorithm or good algorithm* if its time complexity function is bounded above by a polynomial on the size of the input data.

An algorithm whose time complexity function is *not* bounded above by a polynomial function of the input size is said to be an *exponential time algorithm.*

Why are polynomial algorithms considered good?

Polynomials have the following two important properties:

1. Closure property of polynomials: They are closed under addition, multiplication and composition. In other words, the sum, the product and the composition of two polynomials are still polynomials. No other smaller class of functions (for example, the logarithmic functions) having useful complexity bounds has this property. Intuitively, polynomial algorithms become "useless" in practice little by little as the size of the problem increases, whereas

exponential algorithms become “useless” all of a sudden.

2. The formal computational models RAM (Random Access Machine), RASP (Random Access Stored Program Machine), and the Turing Machine are all polynomially equivalent under logarithmic cost (see [9]).

Class-P consists of all decision problems which can be solved in polynomial time. *Class-NP* is the set of all decision problems for which the “YES-ANSWER” can be “exhibited” in polynomial time. “NP” stands for the *nondeterministic polynomial time*. We shall give an example to illustrate the word “exhibit.”

Example 4.2.1 (An NP decision problem):
Instance: An integer $k \geq 4$.

Question: Is k a composite number? Specifically, are there integers $p \geq 2$ and $q \geq 2$ such that $k = pq$?

This problem is in the class-NP. Why? If the answer is “yes,” then the “yes-answer” can be “exhibited” or “proven” quickly, that is, in polynomial time. But how can we do this?

All we have to do is to provide two integers $p \geq 2$ and $q \geq 2$. Once the two integers are furnished, then these two integers can be multiplied together to see if their product is *indeed* equal to the given integer k. Note that the usual algorithm for the multiplication of two integers p and q demands only polynomial time. It is important to note that the time needed to find the factors p and q is *not* at all taken into account. Only the time for the “verification” of the “yes” answer is calculated and this “verification time” should be a polynomial in the size of the input.

The chromatic number $\gamma(G)$ of an undirected graph G is the *minimum* number of colors needed to *paint* the vertices of G in such a way that two vertices joined by an edge are assigned different colors. Note that the vertices not joined by an edge may or may not receive the same color. An undirected graph G is *k-colorable* (k, a nonnegative integer) if k is greater than or equal to the chromatic number $\gamma(G)$. The integers $1, 2, \ldots, k$ are used for the colors. Note that in a k-coloring of a graph G, the set of vertices colored with the same color i are mutually nonadjacent, and this set is called the color class i under the k coloring. A set of vertices of G in which *no* two vertices are joined/adjacent is called an **independent set or stable set**. A set of vertices of G in which any two vertices are adjacent is called a *clique of* G.

THEOREM 4.2.1 (Brooks's theorem):
Brooks's theorem asserts that if G is a simple graph that is neither a complete graph nor an odd elementary cycle, then its chromatic number satisfies the inequality $\gamma(G) \leq \Delta(G)$. ($\Delta(G)$ is the maximum degree of any vertex of G.) (For a proof see [3],[5].)

Determining the chromatic number of an arbitrary graph belongs to a large class of "difficult" problems called *NP-Complete problems* (NP stands for *nondeterministic polynomial time*, for which all known solutions are of the type "brute-force method," that is, "try all possibilities." All problems in the NP-Complete class are "equivalent in computational difficulty."

What is the "brute-force method" for finding the chromatic number of a graph G? We are interested in finding a *smallest* integer k such that G is k colorable. Clearly, every graph G is n colorable (n, the number of vertices of G). Then, we try to color the graph with $(n-1)$ colors. If we don't succeed with $(n-1)$ colors then we conclude that $\gamma = n$. Otherwise, try to color the graph by using $(n-2)$ colors. If the attempt is not successful, then $\gamma = n-1$. If we succeed in coloring G with the help of $n-2$ colors, then try to assign $n-3$ colors and so on until we arrive at a situation where we can color the graph with k colors but not with $k-1$ colors. At this point, we can conclude that $\gamma(G) = k$.

Evidently, this *brute-force algorithm or exhaustive method* takes an enormous amount of time. Technically, this "try all possibilities method" takes an *exponential time* (or intuitively "slow algorithm") in the length of the input (as opposed to a *polynomial time algorithm*, or intuitively "quick or fast algorithm") which is informally the number n of vertices of the graph. In a formal manner, the input length is the number of bits necessary to encode the given instance.

In fact, the class of NP-Complete decision problems share the following two properties:

1. Currently, *no* polynomial time algorithm is *known* to solve any problem belonging to this class. In other words, all the *known* algorithms to solve a problem of this class take exponential time in the worst case.

2. If one of the problems of this class has a polynomial time algorithm, then every other problem of the class also possesses a polynomial time algorithm.

Many real-world problems like the *Traveling Salesman problem or TSP*, and the *Integer programming problem* belong to this class. Note that the TSP is NP-Complete in its decision version whereas the corresponding optimization version is NP-Hard. A decision problem is NP-Hard if any problem in NP is *polynomially reducible* to it. A problem π is polynomially reducible to a problem π' if there is a many-one transformation that carries a positive instance (yes-instance) of π into a positive instance of π' and a negative instance (no-instance) of π into a negative instance of π'.

We now introduce the notion of a *multigraph* or *undirected graph*.

4.3 Multigraph or Undirected Graph

A multigraph is a graph in which the directions/orientations of its arcs are ignored. In a formal manner, a multigraph is defined as follows:

Definition 4.3.1:
A *multigraph or an undirected graph* G is an ordered pair (X, E) where X is a finite set of points (called *vertices*) and E is a finite multiset of *unordered* pairs of vertices (called *edges*).

The following example serves to clarify the definition of the multigraph.

Example 4.3.1 (Multigraph):
Consider the graph obtained by "forgetting"(called the *underlying*

graph) the directions of the arcs of the graph in Example 4.1.1. Such a graph is referred to as the *underlying multigraph* of Example 4.1.1. It is a graph with a vertex set that is the same as that of Example 4.1.1, and its edge set $E = (e_1, e_2, \ldots, e_{12})$ where $e_1 = x_7x_7$, $e_2 = x_2x_2$, $e_3 = x_3x_2$, $e_4 = x_2x_1$, $e_5 = x_1x_4$, $e_6 = x_4x_1$, $e_7 = x_4x_5$, $e_8 = x_3x_4$, $e_9 = x_3x_5$, $e_{10} = x_6x_7$, $e_{11} = x_7x_6$, $e_{12} = x_6x_7$. This is a set theoretic representation of the multigraph.

The following is a geometric representation (see Figure 4.3).

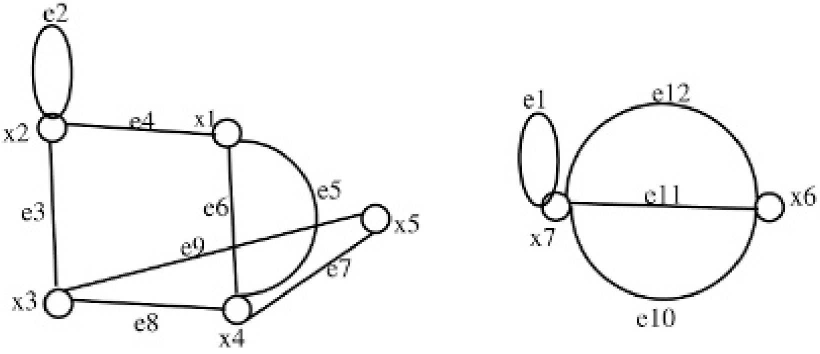


Figure 4.3: An example of a multigraph

In an undirected graph, the concepts of in-degree, out-degree, edge leading out of a vertex, edge leading into the vertex, etc., have no meaning. We should only speak of simply the degree of a vertex, an edge incident with a vertex, etc. The set of all vertices joined (*adjacent*) to a given vertex x is denoted by $\Gamma(x)$, that is,

$$\Gamma(x) = \{\, y \in X \mid xy \text{ is an edge of the graph}\}.$$

Example 4.3.2:

In Example 4.3.1, the degree $d(x_2) = 4$, the number of edges incident with the vertex x_2, the loop at x_2 being counted twice. The edges e_7 and e_9 are adjacent whereas the edges e_3 and e_6 are nonadjacent. The vertices x_4 and x_3 are adjacent whereas the vertices x_2 and x_4 are nonadjacent. The vertices x_3 and x_5 are the *extremities or the end vertices* of the edge e_9. The edges e_1 and e_2 are loops. An edge (called a *link*) which is not a loop may be regarded as a two-element subset of the vertex set. e_5 and e_6 are multiple edges as are e_{10} and e_{12}.

Simple graph

A *simple graph* is a multigraph which does not admit loops and multiple edges. For example, a graph with vertices representing all people in a gathering and edges joining two vertices if the corresponding persons know each other is a simple graph. Thus every edge of a simple graph can be regarded as a two-element subset of the vertex set. The following is the *underlying simple graph* obtained from the multigraph of Example 4.3.1 by deleting all loops and removing all but exactly one edge between two adjacent vertices.

Example 4.3.3 (Example of a simple graph):
(See Figure 4.4.)

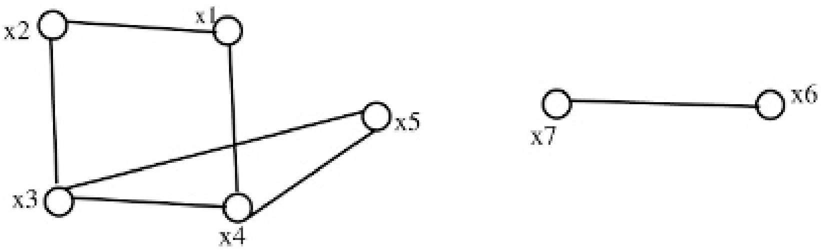


Figure 4.4: An example of a simple graph

Graphical sequence

If G is a multigraph with n vertices $1, 2, \dots, n$ then the sequence $d = (d(1), d(2), \dots, d(n))$ where $d(i)$ = the degree of the vertex i in the graph G, is called the *degree sequence* of the graph G.

A sequence $d = (d_1, d_2, \dots, d_n)$ of non-negative integers is said to be *graphical* if there is a *simple* graph G with n vertices $1, 2, \dots, n$ such that the degree of the vertex i is exactly the integer d_i for all $1 \leq i \leq n$. We also say that the graph G *realizes* the sequence $d = (d_1, d_2, \dots, d_n)$

Example 4.3.4 (Graphical sequence):
The sequence $(2, 2, 3, 3, 2, 1, 1)$ is graphical as the simple graph of Figure 4.4 has seven vertices $x_1, x_2, \dots, x_7$ with $d(x_1) = 2$, $d(x_2) = 2$,$d(x_3) = 3$,$d(x_4) = 3$,$d(x_5) = 2$,$d(x_6) = 1$,$d(x_7) = 1$.

Example 4.3.5 (Non-graphical sequence):
Show that the sequence $(6, 6, 5, 4, 3, 3, 1)$ is *not* graphical.

Solution: If the sequence is graphical, then there is a *simple* graph G with vertices $1, 2, \dots, 7$ with $d(1) = 6$,$d(2) = 6$,$d(3) = 5$,$d(4) = 4$,$d(5) = 3$,$d(6) = 3$,$d(7) = 1$. Since $d(1) = 6$ and $d(2) = 6$ the vertex 1 is joined to all *other* vertices and the vertex 2 is also joined to all *other* vertices. (See Figure 4.5.) But in the process of constructing a graph realizing the sequence, the degree of each vertex is *already* ≥ 2 and hence we can't have a vertex with degree 1 in the graph. Hence the sequence is non-graphical.

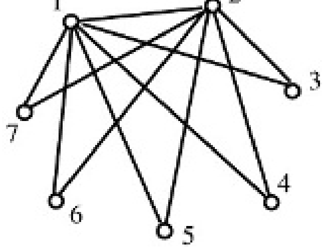


Figure 4.5: A step in the proof that $(2, 2, 3, 3, 2, 1, 1)$ is non-graphical

An application to geometry

Consider n points in the plane such that the Euclidean distance between any two points is at least (≥ 1) one. Prove that there are at most ($\leq$) $3n$ pairs of points at distance exactly equal to 1.

Proof. Let us model this situation by a simple graph G. The vertices correspond to n points of the plane and two vertices are joined by an edge if the distance between its end vertices is *exactly* one. We must show that the number of edges m of this graph is at most $3n$.

Let us suppose that we have already proved $d(x) \leq 6$ for each vertex $x \in X$. (Why 6? The reason will be clear soon.)

Then by Corollary 4.1.1.2,

$$\begin{aligned} 2m &= \sum_{i=1}^{n} d(x) \\ &\leq \sum_{i=1}^{n} 6 \\ &= 6n \end{aligned}$$

Hence, $m \leq 3n$. It remains to show that $d(x) \leq 6$ for each vertex $x \in X$. The proof is by contradiction. If not, there is a vertex x such that $d(x) \geq 7$. Consider some seven vertices $x_1, x_2, \ldots, x_7$ joined to the vertex x (see Figure 4.6).

Since the distance between x and x_i, $1 \leq i \leq 7$, is exactly one, we can draw a circle with x as its center and passing through the vertices $x_1, x_2, \ldots, x_7$. The circumference of this circle is 2π which is $\approx 2 \times 3.14 = 6.28 < 7$. But then there are seven points on the circumference of the circle, hence by the pigeon-hole principle, there are two points at *circular* distance < 1. Therefore, there are two points whose Euclidean distance is < 1 (because a straight line is the shortest distance between two points), a contradiction to our assumption that the distance between two points is ≥ 1. ∎

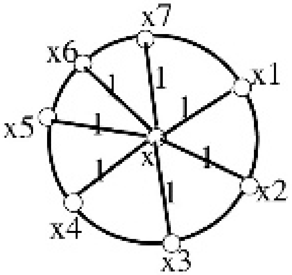


Figure 4.6: A step in the proof

4.4 Some Special Graphs

Complete graph

A directed graph is a *complete graph*, if between any two distinct vertices x and y, there is at least one arc joining them. Hence, a 1-graph is complete if (x, y) is not an arc then (y, x) is an arc.

Example 4.4.1 (A complete graph):
The following graph is a complete graph on four vertices (see Figure 4.7):

A *simple complete* graph on n vertices is denoted by K_n. Since any two vertices of K_n is joined by exactly one edge, the number of edges of K_n is the same as the number of two-element subsets of a set of n elements, which is the binomial coefficient $\binom{n}{2.}$ In notation, $m(K_n) = \binom{n}{2} = \frac{n(n-1)}{2}$.

Example 4.4.2 (Complete graphs K_4 and K_5):
In the following diagrams, the complete graphs on four and five vertices are drawn (see Figure 4.8).

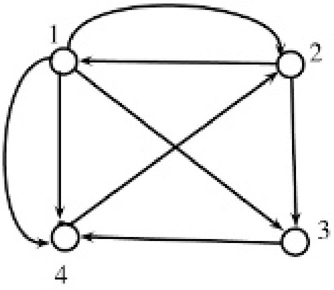


Figure 4.7: A complete graph on four vertices

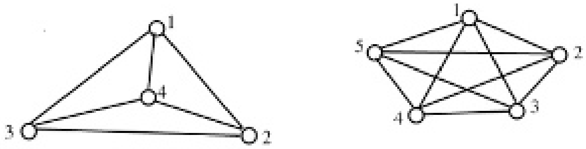


Figure 4.8: The graphs K_4 and K_5

Bipartite graph

A graph G is a *bipartite graph or 2-colorable graph* (briefly, bigraph) if its vertex set X admits a partition X_1 and X_2 into *two* parts, such that each arc/edge of G joins a vertex of X_1 and a vertex of X_2. Stated differently, there are no arcs/edges between two vertices of X_1 or between two vertices of X_2. Such a partition is called a *bipartition* of the vertex set. In particular, a bipartite graph cannot contain a loop.

A *complete* bipartite graph is a bipartite graph in which there is at least one arc/edge joining a vertex of X_1 and a vertex of X_2. A complete *simple* bipartite graph with parts consisting of p and q vertices is denoted by $K_{p,q}$. The number of edges of the complete bipartite graph $K_{p,q}$ is pq.

Example 4.4.3 (A bipartite graph and the bipartite graph $K_{3,3}$): In the following diagrams, we see a bipartite graph and the complete bipartite graph $K_{3,3}$ (see Figure 4.9). The bipartitions of the graphs are $X_1 = \{1, 2, 3\}$ and $X_2 = \{4, 5, 6\}$. There are no edges between the vertices of the subset X_1 and the subset X_2. Such subsets of vertices of a graph are called *stable sets or independent sets*.

On the other hand, the complete graph K_3 is *not* bipartite.

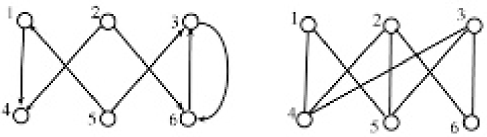


Figure 4.9: A bigraph and the graph $K_{3,3}$

Example 4.4.4 (Star graph $K_{1,3}$.):
A star graph is a complete bipartite graph $K_{1,p}$. The case of $p = 3$ is an important special case and the graph $K_{1,3}$ is drawn below (see Figure 4.10).

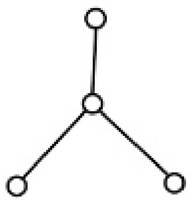


Figure 4.10: The star graph $K_{1,3}$

k-partite graph or k-colorable graph

There is nothing sacrosanct about the number "two" in the above definition of the bipartite graph. A *k-partite graph or k-colorable graph* is a graph whose vertex set can be partitioned into k subsets such that each arc/edge has one extremity/end in one subset and another extremity/end in another subset. Put differently, we must not have an arc/edge joining two vertices of the same subset of the partition.

A *complete* k-partite graph is a k-partite graph in which there is at least one arc/edge joining two vertices lying in two different subsets of the partition.

Definition 4.4.1 (Integer functions):
For a real number x, the *floor of* x denoted by $\lfloor x \rfloor$ is the *greatest* integer less than or equal to x. For example, $\lfloor 4.7 \rfloor = 4$, $\lfloor -4.7 \rfloor = -5$,$\lfloor \pi \rfloor = 3$, where π is *the circle ratio*, $\lfloor \mathrm{e} \rfloor = 2$ where e is the *base of the natural logarithm* .

For a real number x, the *ceiling of* x denoted by $\lceil x \lceil$ is the *least* integer greater than or equal to the real number x. For example, $\lceil 4.7 \rceil = 5$, $\lceil -4.7 \rceil = -4$,$\lceil \pi \rceil = 4$, where π is *the circle ratio*, $\lceil \mathrm{e} \rceil = 3$.

Note that for any real number x, we have the double inequality, $0 \leq \lfloor x \rfloor \leq \lceil x \rceil \leq 1$ and $\lceil x \rceil = \lfloor x \rfloor$ if and only if x is an integer.

The name "entire" is derived from the French word "Entier" which means "integer."

A *simple complete k-partite* graph with the parts consisting of $n_1, n_2, \ldots, n_k$ vertices is denoted by $K_{n_1,n_2,\ldots,n_k}$. The number of edges of $K_{n_1,n_2,\ldots,n_k}$ is $\sum_{1 \leq i < j \leq k} n_i n_j$.

A graph $K_{n_1,n_2,\ldots,n_k}$ satisfying the condition $|n_i - n_j| \ \leq 1$ for $i \leq i, j \leq k$ is called a Turán graph, that is, $n_i = \lfloor \frac{n}{k} \rfloor$ or $\lceil \frac{n}{k} \rceil$ for $1 \leq i \leq k$. The Turán graph is denoted by $T_{n,k}$.

We shall see later that bipartite graphs can be recognized in *polynomial time*, whereas no such algorithm is known for k-partite graphs for $k \geq 3$.

EXAMPLE 4.4.5 (The Turàn graph $T_{6,3} = K_{2,2,2}$):
The following is the Turàn graph $K_{2,2,2}$ (see Figure 4.11).

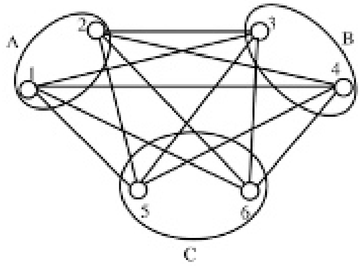


Figure 4.11: The Turàn graph $T_{6,3} = K_{2,2,2}$

Regular graph

A graph is a *regular graph* if the degree of every vertex is the same integer. A k-regular graph is one for which the degree of each vertex is k. A 3-regular graph is often called a *cubic graph*. The complete graph K_n is an $(n-1)$-regular graph.

EXAMPLE 4.4.6 (Modeling using bipartite regular graphs):
In a certain college, each girl "knows" exactly k boys and each boy "knows" exactly k girls ($k > 0$). Prove that the number of boys in the college is equal to the number of girls in the college. (We assume that "knowing" is a *symmetric relation.*)

Solution: We construct a bipartite graph $G = (X_1, X_2; E)$ as follows: X_1 is the set of all girls, and X_2 is the set of all boys of the college and a girl "g" is joined to a boy "b" if and only if the girl "g" and the boy "b" know each other. Thus G is a bipartite regular graph of degree $k > 0$ with bipartition (X_1, X_2) and we have to show that $|X_1| = |X_2|$.

We shall calculate the number of edges of G in *two* different ways and equate them.

1. Since there are no edges between two vertices of X_1 and two vertices of X_2, the number of edges m of G is (by viewing from X_1)

$$m = k|X_1|.$$

2. The number of edges of G by viewing from X_2 is

$$m = k|X_2|.$$

Equating the two values of m, we have $k|X_1| = k|X_2|$. Since $k \neq 0$, we have $|X_1| = |X_2|$.

Example 4.4.7 (Petersen graph):
The following graph is called the Petersen graph. This graph possesses many remarkable properties. The Petersen graph is a 3-regular graph (see Figure 4.12). It cannot be drawn in the plane in such a way that two of its edges avoid intersecting at a point other than at one of its vertices. Such a graph is called a *non-planar graph*. If there is a drawing of a graph in the plane such that no two of its edges/arcs intersect at a point other than a vertex then such a graph is referred to as a *planar graph*.

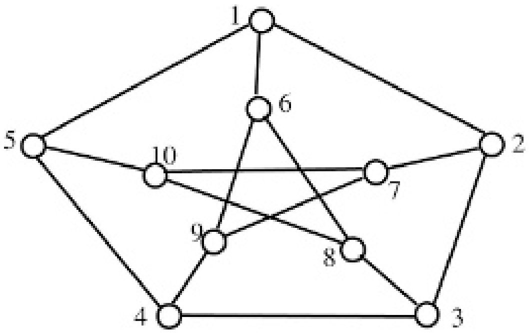


Figure 4.12: The Petersen graph

Definition 4.4.2 (Strongly regular graph):
A simple k-regular graph ($k > 0$) G on n vertices which is not a complete graph is *strongly regular* if every pair of adjacent vertices have λ common neighbors and every pair of nonadjacent vertices have μ common neighbors. The numbers n, k, λ, μ are the parameters of the strongly regular graph.

Again, the graph of Petersen is strongly regular with parameters $10, 3, 0, 1$.

Definition 4.4.3 (Line graph):
Consider a simple graph $G = (X, E)$ with at least one edge. Then the *line graph of* G, denoted by $L(G)$, is defined as follows: The vertex set of $L(G)$ is the set of edges of the base graph G and two vertices e_i and e_j of $L(G)$ are adjacent in $L(G)$ if and only if the edges e_i and e_j share a common vertex in the graph G, that is, e_i and e_j are adjacent edges of G.

Example 4.4.8 (A graph and its line graph):
The graph G with its edges labeled e_1, e_2, e_3, e_4, e_5 and its line graph $L(G)$ with its vertices labeled e_1, e_2, e_3, e_4, e_5 (see Figure 4.13).

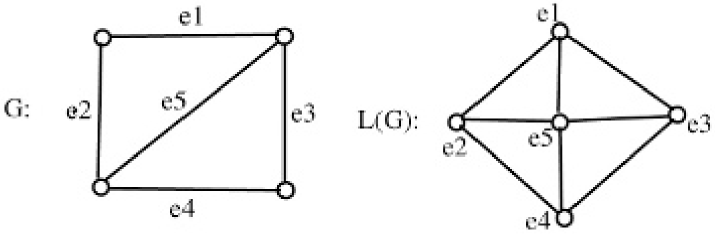


Figure 4.13: A graph and its line graph

Theorem 4.4.1:
Let G be a simple graph with n vertices $1, 2, \ldots, n$ and m edges.

Let the degree of the vertex i be d_i for $1 \leq i \leq n$. Then the number of vertices of the line graph $L(G)$ is m and the number of edges of $L(G)$ is given by the formula

$$-m + \frac{1}{2}\sum_{i=1}^{n} d_i^2.$$

Proof. By the definition of the line graph $L(G)$, the number of vertices of $L(G)$ is m. Now we have to find the number of edges of $L(G)$ in terms of d_i's. Take the vertex i of G. An edge of $L(G)$ is obtained from two adjacent edges of G. Since the degree of i is d_i, there are exactly d_i edges incident at i and these d_i edges are *mutually* adjacent to each other (in other words, the star K_{i,d_i} is a subgraph of G at the vertex i) and hence form a complete graph K_{d_i} while constructing $L(G)$. We know that the number of edges of the complete graph K_{d_i} is the binomial coefficient $\binom{d_i}{2}$. This argument is valid for any vertex i of G. Hence the number of edges of $L(G)$ is $\sum_{i=1}^{n}\binom{d_i}{2} = \sum_{i=1}^{n} d_i(d_i-1)/2 = \frac{1}{2}\sum_{i=1}^{n}(d_i^2 - d_i)$ which is equal to $\frac{1}{2}\sum_{i=1}^{n} d_i^2 - \frac{1}{2}\sum_{i=1}^{n} d_i$. But by Corollary 4.1.1.1, $\sum_{1=1}^{n} d_i = 2m$. By plugging this sum into the number of lines of $L(G)$, the desired formula is obtained. ■

Definition 4.4.4 (The chromatic index of a graph):
A multigraph G is k *edge colorable* if its edges can be colored in k or fewer colors in such a way that no two edges sharing a common vertex receive the same color. The *edge chromatic number* of G denoted by $q(G)$ is the *minimum* integer k for which G is k edge colorable. Let us note that in a k-edge-coloring of G the edges receiving the same color are mutually nonadjacent. A set of mutually nonadjacent edges in a multigraph is often called a *matching* in G. A matching consisting of $n/2$ (n even) edges of G is called a *perfect matching in* G. The chromatic index of a graph can be viewed as follows: $q(G)$ is the smallest integer k such that the edge set E of G can be partitioned into the union of k mutually disjoint matchings, that is,

$$E = M_1 \cup M_2 \cup \cdots \cup M_k \text{ where each } M_i \text{ is a matching in } G$$

and $M_i \cap M_j \neq \emptyset$ for all $i, j, i \neq j$.

By the definition of the line graph it is seen that $q(G) = \gamma(L(G))$, the chromatic number of $L(G)$.

Example 4.4.9 (Chromatic index):
The graph G with its edges colored with the colors $1, 2, 3, 4$. (see Figure 4.14). The reader can verify that the chromatic index of G is 4.

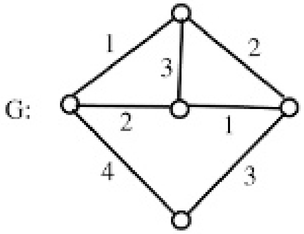


Figure 4.14: A graph with chromatic index number 4

The following theorem is due to Gupta and independently by Vizing (for a proof, see [1] or [5]).

Theorem 4.4.2 (Gupta, Vizing):
If G is a loopless multigraph with multiplicity p (the number of edges joining any two vertices is less than or equal to the integer p) then its chromatic index satisfies the double inequality

$$\Delta(G) \leq q(G) \leq \Delta(G) + p.$$

Definition 4.4.5 (Total graph):
(See the book by Harary [2].) Consider a simple graph G with vertices $x_1, x_2, \ldots, x_n$ and with edges $e_1, e_2, \ldots, e_m$. Then the total graph of G denoted by $T(G)$ is a graph with vertex set

$\{ x_1, x_2, \ldots, x_n, e_1, e_2, \ldots, e_m \}$ and two vertices of $T(G)$ are adjacent whenever the following conditions are verified:

1. If the vertices of $T(G)$ are x_i and x_j $(i \neq j)$ then we join x_i and x_j if these two vertices are adjacent in the base graph G;

2. If the vertices of $T(G)$ are e_i and e_j $(i \neq j)$ then we join e_i and e_j if these two edges are adjacent in the base graph G;

3. If the vertices of $T(G)$ are x_i and e_j $(i \neq j)$ then we join x_i and e_j if the vertex x_i is incident with the edge e_j in the base graph G.

Example 4.4.10 (A graph and its total graph):
The graph G with its vertices labeled x_1, x_2, x_3 and its edges labeled e_1, e_2 and its total graph $T(G)$ with its vertices labeled x_1, x_2, x_3, e_1, e_2 (see Figure 4.15).

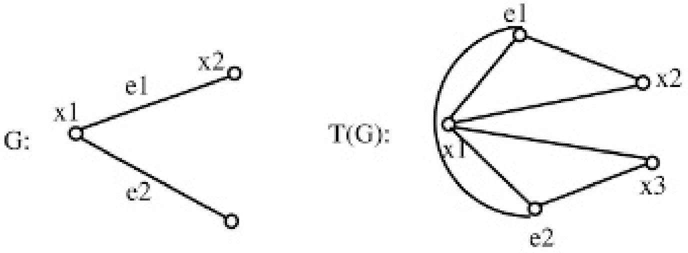


Figure 4.15: A graph and its total graph

Definition 4.4.6 (Complement of a simple graph):
Consider a simple graph $G = (X, E)$. *The complement of the graph* G denoted by $G^c = (X, E^c)$ is the simple graph with the same vertex set X as G and two vertices are adjacent in G^c if and only if they are *not* adjacent in the graph G, that is, $E^c = \{ ij \mid i \neq j \text{ and } ij \notin E \}$.

For example, the complement of the complete graph K_n^c is the graph consisting of n isolated vertices and the complement of the complete bipartite graph $K_{p,q}$ is the disjoint union of two complete graphs K_p and K_q.

Example 4.4.11 (Complement of a graph):
In Figure 4.16 the graph H is the complement of the graph G.

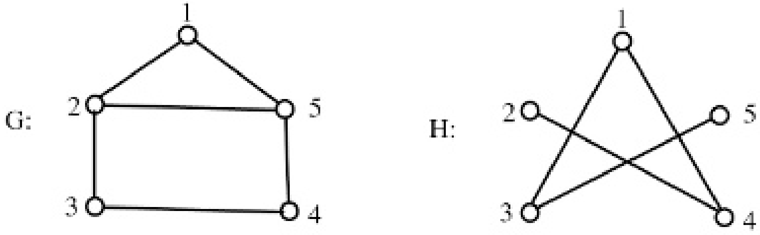


Figure 4.16: A graph G and its complement $H = G^c$

An extremal result

Extremal structures exhibit interesting properties.

What is the *maximum* number of edges in a simple bipartite graph on n vertices? What is the *maximum* number of edges of a simple graph without containing a given induced subgraph ? Such questions are studied in the part of graph theory called extremal graph theory.

Our first question can be answered easily. Consider the complete bipartite graph $K_{3,3}$ on 6 vertices. The number of its edges is 3×3. If we partition the set of 6 vertices into two parts consisting of 2 and 4 vertices, we have the complete bipartite graph $K_{2,4}$ whose number of edges is only 2×4. So, intuitively, we feel that to have the maximum number of edges in a bipartite graph, the bipartition should be as equal as possible. This intuitive feeling is asserted by the following theorem due to Turán.

In fact, Turán's theorem is nothing more than the following simple observation in elementary number theory.

Observation 4.4.1:
Consider a partition of a positive integer n into the sum of two positive integers, n_1 and n_2, that is, $n = n_1 + n_2$ (for example,

13=6+7 or 13=5+8 or 13=4+9). Then the product of the partition numbers is *maximum* if and only if the partition numbers are *as equal as possible.* For example, $6 \times 7 > 5 \times 8 > 4 \times 9$.

> A single drop of water added to a full cup of water, makes the cup overflow.

Theorem 4.4.3:
The number of edges of a bipartite simple graph on n vertices is $\leq \lfloor \frac{n^2}{4} \rfloor$.

Proof. We give a simple proof using differential calculus! A combinatorial proof can be found in books [2], [5].

Consider a bipartition of the vertex set X into two parts consisting of p and $n-p$ vertices. In order to obtain the maximum number of edges, the bipartite graph should be a *complete* bipartite graph. Then the number of edges of the complete bipartite graph $K_{p,n-p}$ is $p(n-p)$. Let us regard p as a continuous variable in the closed interval $[1,n]$. The we have to *maximize* the following function m which is a function of the one variable p

$$m = p(n-p). \tag{4.1}$$

Differentiating with respect to p, $dm/dp = n - 2p$ and $dm/dp = 0$ gives the value of $p = n/2$.

The second derivative $d^2m/dp^2 = -2 < 0$. Hence m attains the maximum value if $p = n/2$ and the maximum value is obtained by substituting the value of $p = n/2$ in Equation 4.1.

Hence, $m \leq \ n^2/4$. Since m is an integer, we have $m \leq \lfloor \frac{n^2}{4} \rfloor$. ■

Remark 4.4.1 (Contrapositive of Theorem 4.4.3):
(See Chapter 6 for more details on contrapositive: The contrapositive of the statement "p implies q" is: "not q implies not p." They are logically equivalent.)

Theorem 4.4.3 can be stated as "if a simple graph contains more than $\leq \lfloor \frac{n^2}{4} \rfloor$ edges, then the graph is *not* bipartite. In fact, the graph contains a triangle, an elementary cycle of length 3.

Isomorphism

> Mathematicians do not study objects but the relations among the objects; it is therefore irrelevant for them to replace these objects by others provided the relations are preserved.
>
> The matter is not important to them; only the *form* is relevant to them.
>
> *H. Poincaré*

> Are the great new discoveries really just recognitions that two things are the same?
>
> *Paul Halmos*

In mathematics, we declare two objects to be *equal* if they possess some common properties, that is, we measure the equality of objects with respect to some properties. We rarely ask for *identity* of two objects. This is even true of real-life situations. When we say that two candidates are equally good for a post of a professor at a university, we mean that the quality of their dossiers are the same; one may be taller than the other, etc. An elementary example from arithmetic is the following: We consider the two rational numbers 1/2 and 5/10 as equal or equivalent, even though they are not identical.

When do we declare two graphs to be equal? Two graphs G_1 and G_2 are *isomorphic* if we can draw the graph G_2 so as to "resemble" the graph G_1. This means that in the pictorial representation of a graph/multigraph, the relative position of vertices and the forms of arcs/edges joining the pairs of vertices is immaterial. We are only interested in the pair of vertices which are joined by an arc/edge and how many times they are joined. Isomorphic graphs possess the same structural properties. Thus two graphs differing only in the labels of their vertices but not in the structure of their arcs are equal.

Example 4.4.12 (A graph isomorphic to the Petersen graph): For example, the following graph H is another way of drawing the Petersen graph (see Figure 4.17). In other words, the Petersen graph is isomorphic to the following graph H. The vertices and the edges of the graph H are not labeled. Such a graph is called an *unlabeled graph.* We use labels to vertices and edges only for the purpose of referring to them.

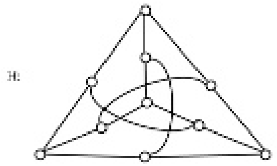


Figure 4.17: Another drawing of the Petersen graph

In a formal manner, we give the definition of isomorphism of two graphs. Before giving the definition, we need a useful notation.

Given a graph G and two distinct vertices x and y, $m^+(x,y)$ denotes the number of arcs having the vertex x as its tail and the vertex y as its head. Also, $m^-(x,y)$ denotes the number of arcs with x as its head and y as its tail. We set

$$m(x,y) = m^+(x,y) + m^-(x,y).$$

For each loop attached to the vertex x, $m^+(x,x) = m^-(x,x) = 1$, $m(x,x) = 2$.

More generally, for disjoint subsets S and T of the vertex set of a graph, $U(S,T)$ denotes the multiset/set of all arcs with tail in the set S and head in the set T; we set $m(S,T) = |U(S,T)|$.

Similarly, the notation $E(S,T)$ means the set of all edges of a multigraph with one end in S and another end in T, where S and T are disjoint subsets of the vertex set.

Definition 4.4.7:
Consider two graphs $G_1 = (X_1, U_1)$ and $G_2 = (X_2, U_2)$. We say that the graph G_1 is *isomorphic* to the graph G_2, denoted by $G_1 \cong G_2$, if there is a one-to-one correspondence f from the set X_1 onto the set X_2 satisfying the following property: The number of arcs with the vertex x as its tail and the vertex y as its head in

the graph G_1 is the same as the number of arcs having the vertex $f(x)$ as its tail and the vertex $f(y)$ as its head in the graph G_2. In notation,

$$m^+_{G_1}(x,y) = m^+_{G_2}(f(x), f(y)) \text{ for every } x, y \in X_1.$$

Similar definition holds for isomorphism of a multigraph if we replace m^+ simply by m in the above definition.

The definition becomes somewhat simpler in the case of a simple graph.

Two simple graphs G_1 and G_2 are isomorphic if there is a bijection from the vertex set of G_1 onto the vertex set of G_2 preserving adjacency and non-adjacency.

To prove that two graphs G and H are isomorphic, we must find an isomorphism between the graphs G and H. The following example illustrates this.

Example 4.4.13 (Isomorphism):
Consider the two labeled graphs G and H of Figure 4.18. The function $f : \{1, 2, \ldots, 10\} \to \{1, 2, \ldots, 10\}$ defined below establishes an isomorphism between the graph G and the graph H.

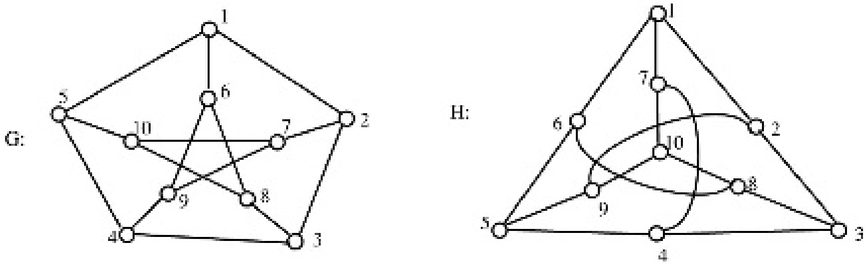


Figure 4.18: Isomorphic graphs

$$f(1) = 1, f(2) = 7, f(3) = 4, f(4) = 5, f(5) = 6, f(6) = 2,$$

$$f(7) = 10, f(8) = 3, f(9) = 9, f(10) = 8.$$

The reader can verify that ij is an edge of the graph G if and only if $f(i)f(j)$ is an edge of H. The function f is a *succinct certificate* to "exhibit" that $G \cong H$.

Definition 4.4.8 (Self-complementary graph):
A simple graph G is *self-complementary* if it is isomorphic to its complement G^c; symbolically,

$$G \cong G^c.$$

Example 4.4.14 (Self-complementary graph):
In Figure 4.19 the graph G is a self-complementary graph. The reader is asked to draw the complement G^c and verify that $G \cong G^c$.

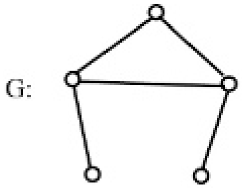


Figure 4.19: A self-complementary graph

Example 4.4.15:
Prove that the number of vertices of a self-complementary graph is of the form either $4p$ or $4p+1$.

Solution: Let $G = (X, E)$ be a self-complementary graph. Then by definition, $G^C = (X, E^c) \cong G$. In particular, the number of edges of G is equal to the number of edges of G^c. Symbolically, $m(G) = m(G^c)$.

Also by the definition of G^c, we have $K_n = (X, E \cup E^c)$ where K_n is the complete graph on n vertices. Hence we have,

$$\begin{aligned} |E| + |E^c| &= m(K_n) \\ \text{that is } m(G) + m(G^c) &= n(n-1)/2 \text{ since } m(K_n) = n(n-1)/2 \\ \text{implies } 2m(G) &= n(n-1)/2 \text{ since } m(G) = m(G^c) \end{aligned}$$

$$\text{implies } m(G) = n(n-1)/4.$$

Since $m(G)$ is an integer, this last equality implies that 4 divides $n(n-1)$. But $n-1$ and n are consecutive integers and hence exactly one of them must be even and the other odd. Therefore, 4 divides either n or else $n-1$. Hence n must be either of the form $4p$ or $4p+1$.

To establish that two graphs G and H are non-isomorphic, it is enough if we find a *property* P of a graph which holds in the graph G but which does *not* hold in the graph H. The graph in Figure 4.20 illustrates this.

Example 4.4.16 (Non-isomorphism):
Consider the two graphs of Figure 4.20. We know that by the

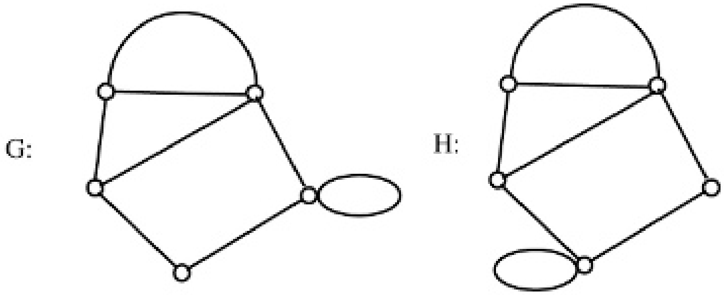


Figure 4.20: Non-isomorphic graphs

definition of isomorphism, if two graphs differ either in the number of vertices or in the number of edges then they are *not* isomorphic. But the two graphs G and H have the same number of vertices and edges. So we have to find some property P which holds in G but which fails in H.

The property P is the adjacency of vertices of degree 4.

The graphs are not isomorphic because in the graph G, the only two vertices of degree 4 are adjacent whereas in the graph H, the only two vertices of degree 4 are *not* adjacent.

In general, to prove the non-isomorphism of two graphs on n vertices we must examine all the possible $n!$ ($10! > 3.5$ million) bijections to say that no bijection preserves the adjacency and non-adjacency.

Clearly isomorphic graphs/multigraphs possess the same number of vertices and arcs/edges but the converse is not true. Determining if two graphs are isomorphic is an *intermediately difficult* unsolved problem. The graph isomorphism is believed to belong to the *class-NPI.* Here the letter "I" stands for the word *intermediate* difficulty. The class NPI "lies" between the *Class-P* (the class of decision problems solvable in a polynomial time algorithm) and NP-complete, that is, intuitively, Class-P $\leq$ Class-NPI $\leq$ NP-Complete. Here, the symbol $\leq$ is employed to indicate the "difficulty" of the classes. *The isomorphism problem* is particularly intriguing for its unsettled complexity status.

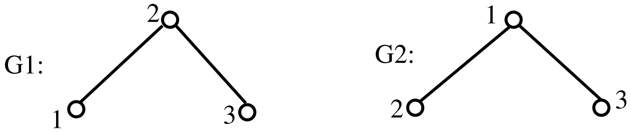
Isomorphism is an equivalence relation on the set of all graphs. The relation of isomorphism partitions the set of all graphs into disjoint equivalence classes. Two graphs are isomorphic if they belong to the same equivalence class, otherwise they are not isomorphic. We attach labels to the vertices and arcs/edges to facilitate the reference to them. A graph without any labeling in the geometric representation may be thought of as a representative of an equivalence class of isomorphic graphs.

Label isomorphism

Consider two *labeled* graphs/multigraphs G_1 and G_2 having the same vertex set. Two graphs are *label isomorphic* or *identical* if the *identity* function $f(x) = x$ from the vertex of G_1 onto the vertex set of G_2 is an isomorphism of G_1 onto G_2.

Isomorphism does not imply label isomorphism. The following example illustrates this statement:

Example 4.4.17:



Example 4.4.18 (A graph isomorphic to the Petersen graph): For example, the graph H of Figure 4.21 is another way of drawing the Petersen graph (see Figure 4.18). In other words, the Petersen graph is isomorphic to the following graph H. The vertices and the edges of the graph H are not labeled. Such a graph is called an *unlabeled graph*. We use labels to vertices and edges only for the purpose of referring to them.

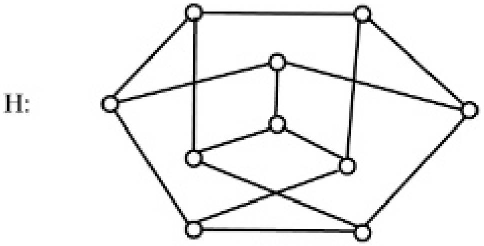


Figure 4.21: Yet another drawing of the Petersen graph

A simple result in enumerative graph theory

Enumerative graph theory is concerned with the study of the number of graphs on n vertices satisfying some conditions. We prove the following simple combinatorial result on the number of simple labeled graphs on n vertices, that is, the number of graphs on n vertices $1, 2, \cdots, n$ up to label isomorphism.

Theorem 4.4.4:
The number of simple labeled graphs on n vertices is $2^{\binom{n}{2}}$. More precisely, the number of *non*label isomorphic (or non-identical) simple graphs on the given set of n vertices is exactly $2^{\binom{n}{2}}$.

Proof. Let the vertex set be $X = \{1, 2, \ldots, n\}$. Any two-element subset of X determines an edge and conversely an edge of a simple graph can be considered as a two-element subset of its vertex set. Hence the set of all *possible* edges is $\mathcal{P}_2(X) = \{ ij \mid 1 \leq i < j \leq n \}$ (to avoid repetition of an edge 12 as 21, we have written $i < j$) and their number is the binomial coefficient $\binom{n}{2}$.

Any subset of $\mathcal{P}_2(X)$ determines a graph with the vertex set X and there are exactly $2^{\binom{n}{2}}$ subsets of $\mathcal{P}_2(X)$. (Since the number of subsets of a set of k elements is exactly 2^k.) ■

The following example illustrates the above theorem.

Example 4.4.19 (Labeled graphs on three vertices):
There are $2^3 = 8$ different labeled simple graphs on 3 vertices. These graphs are (see Figure 4.22):

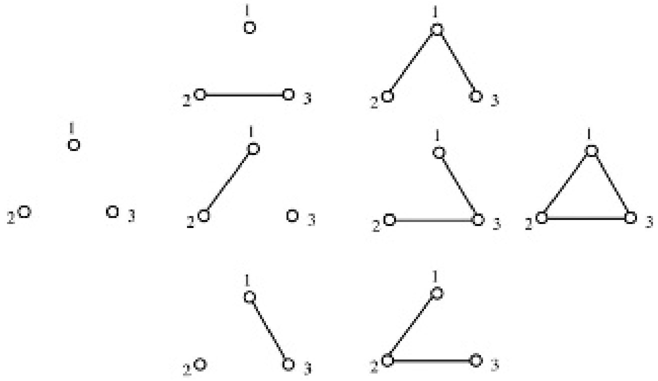


Figure 4.22: Labeled graphs on 3 vertices

In the above example, there are *only four* graphs which are *non-isomorphic.*

In general, it is a difficult unsolved problem to determine the number of non-isomorphic graphs on n vertices.

Corollary 4.4.4.1:
The number of simple labeled graphs on n vertices and with m edges is $\binom{\binom{n}{2}}{m}$.

Proof. Let us first recall that the maximum number of edges of a simple graph on n vertices is the *binomial coefficient* $\binom{n}{2}$.

Each choice of m edges from among the $\binom{n}{2}$ possible edges determines a graph of m edges on n vertices. Hence the number of labeled graphs on n vertices and m edges is the same as the number of possible subsets of m elements from a set of $\binom{n}{2}$ elements. Therefore, the desired number is $\binom{\binom{n}{2}}{m}$. ■

The following example gives a lower bound on the number of non-isomorphic simple graphs on n vertices.

Example 4.4.20:
Show that the number of non-isomorphic simple graphs on n vertices is $\geq 2^{n(n-1)/2}/n!$

Solution: By Theorem 4.4.4, the number of labeled simple graphs on n vertices is $2^{\binom{n}{2}}$. Let k be the number of isomorphic graphs on n vertices.

Now consider any simple *labeled* graph G on n vertices. The vertices of G can be permuted in $n!$ ways, each permutation resulting in the graph isomorphic to G. This is because in any permutation of the vertices of G, the *combinatorial structure* of the edges is preserved. But in the resulting $n!$ graphs, all need *not* be non-identical. For example, if G is the complete graph K_n, any permutation of the vertices of K_n results in a graph identical to K_n. Hence, we have the inequality,

$$k \times n! \geq 2^{n(n-1)/2}.$$

Therefore, $k \geq 2^{n(n-1)/2}/n!$.

4.5 Graphs and Subgraphs

A *subgraph* of a graph/multigraph G is a graph having all of its vertices and arcs/edges in the graph G.

Types of subgraphs

Induced subgraph

Intuitively, an *induced subgraph* on a subset A of vertices of a graph G, is just a "photographic image" of the part of the graph G focused on the subset of vertices A. Consider a graph/multigraph G with the vertex set X and a vertex x of G. The vertex deleted subgraph, $G \setminus x$, is the graph whose vertex set is $X \setminus x$ and with an arc/edge multiset with all arcs/edges of G having no extremity in x. Stated differently, the graph $G \setminus x$ consists of all arcs/edges of G possessing both extremities in the set $X - x$.

In a formal manner, for a subset A of the vertex set X, the induced subgraph of A, denoted by $G(A)$ or $< A >$ is a graph obtained from G by deleting the vertices of $X \setminus A$ in succession. Put differently, $G(A)$ is the graph with vertex set A and the arc/edge multiset consisting of all edges of G having both extremities in the set A. Let us note that the graph $G(A)$ is the *maximal* subgraph of G with the vertex set A (maximal with respect to the set inclusion). Let us refer to the following example.

Example 4.5.1 (A graph and its subgraphs):
Let us refer to the graphs of Figure 4.23. The graph G_2 is an induced subgraph of the graph G. Note that the graph G_2 can also be obtained by removing the vertex 1 from the graph G, that is, $G_2 = G - 1$. The graph G_1 is *not* an induced subgraph of G because the directed arc $(4, 1)$ is present in G but missing in G_1. Similarly, G_3 is *not* an induced subgraph of G.

The following example gives a "somewhat" obvious necessary condition for a sequence to be graphic.

Example 4.5.2 (A necessary condition for a sequence to be graphic):
Consider a sequence $d = (d_1, d_2, \ldots, d_n)$ of non-negative integers with $d_1 \geq d_2 \geq \cdots \geq d_n$, If d is graphic, then

1. $\sum_{i=1}^{n} d_i$ is an even integer.

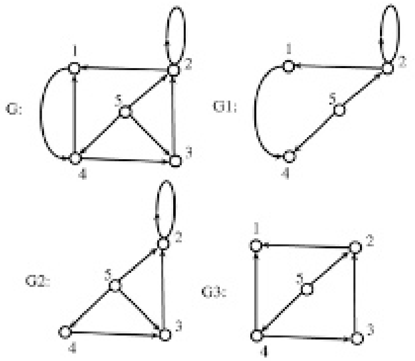


Figure 4.23: A graph and its subgraphs

2. $\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(k, d_i)$ for $1 \leq k \leq n$.

   Solution: Since d is graphic, there is a graph $G = (X, E)$ with vertex set $[n] = \{1, 2, \ldots, n\}$ and degree $d(i) = d_i$ for all $1 \leq i \leq n$. Then by Corollary 4.1.1.1, we have $\sum_{i=1}^{n} d_i = 2m$, which is an even integer.

   Now, we have to prove the inequality $\sum_{i=1}^{k} d_i \leq k(k-1) + \min(k, d_i)$ for $1 \leq k \leq n$.

   Observe that the above inequality gives an *upper bound* on the sum of the k *largest* degrees of the vertices of the graph G. This means that we have to *maximize* the sum $\sum_{i=1}^{k} d_i$. We split this sum of degrees into two disjoint parts: (Divide and Prove!) 1. The sum of the edges arising by joining the k vertices $[k] = \{1, 2, \ldots, k\}$ with each other *plus* 2. The sum of the edges joining the remaining set of vertices $\{k+1, k+2, \ldots, n\}$ with the set of vertices $\{1, 2, \ldots, k\}$. Part 1. To maximize this sum, we must assume that the k vertices $1, 2, \ldots, k$ *induce* a *complete subgraph* K_k. In this case, the degree of each vertex i, $1 \leq i \leq k$ *in* the subgraph K_k is $k-1$.

   Part 2. Let us first recall the notation $m(S, T)$ for *disjoint* subsets S and T of the vertex set X of the graph G. $m(S, T)$

is the number of edges of G with one end in the set S and the other end in the set T. In particular, $m(i, T)$ for a vertex $i \notin T$, stands for the number of edges with one end incident with i and the other end in the set T. We also use the following simple observation: if $x \leq a$ and $x \leq b$, then $x \leq \min(a, b)$. Now we have to maximize the number of edges joining the remaining $n - k$ vertices $k + 1, k + 2, \ldots, n$ with the vertices $1, 2, \ldots, k$ which is by our notation $m([k], X \setminus [k])$. Consider an arbitrary vertex i with $k + 1 \leq i \leq n$.

On the one hand the vertex i can be joined to maximum of *all* the k vertices $1, 2, \ldots, k$, that is, $m(i, X \setminus [k]) \leq k)$, but on the other hand it can't be joined to more than d_i vertices of K_k, that is, $m(i, X \setminus [k]) \leq d_i)$. Combining the two inequalities we have : i is joined to at most $\min(k, d_i)$ vertices of K_k for all $k + 1 \leq i \leq n$. In notation, $m(i, X \setminus [k]) \leq \min(k, d_i)$.

Combining the two parts, we have the desired upper bound for the sum:

$$\begin{aligned} \sum_{i=1}^{k} d_i &\leq \sum_{i=1}^{k}(k-1) + \sum_{i=k+1}^{n} \min(k, d_i), \text{ for } 1 \leq k \leq n \\ &\leq k(k-1) + \sum_{i=k+1}^{n} \min(k, d_i), \text{ for } 1 \leq k \leq n. \end{aligned}$$

The above (see Example 4.5.2) “somewhat” obvious *necessary* condition for a sequence to be graphic has also been proved to be *sufficient* by Erdös and Gallai (see [1]). Such characterizations are called *TONCAS*, meaning “the obvious necessary condition is also sufficient.” For a simple and elegant proof of the theorem using Tutte's f-factor theorem (stated below), see [3].

Spanning subgraph

Consider a graph/multigraph G. The graph obtained by deleting an arc/edge of G is the graph whose vertex set is the same as that of G, and whose arc/edge multiset consisting of all arcs/edges of G except the one deleted.

Let us note that when we delete an arc/edge of a graph, the extremities of the deleted arc/edge still remain in the graph whereas the removal of a vertex suppresses not only the concerned vertex but also all arcs/edges having an extremity with the deleted vertex.

A *spanning subgraph* of a graph/multigraph G is a subgraph with the same vertex set as that of G. Let us note that a spanning subgraph can be obtained by deleting some subset of arcs/edges of G, one by one in succession.

For a subset U_1/E_1 of arcs/edges of a graph G, the arc/edge induced subgraph $G(U_1)/G(E_1)$ is a graph with the vertex set consisting of all extremities of U_1/E_1 and the arc/edge multiset U_1/E_1.

Finally, a *subgraph* of a graph/multigraph is one which can be obtained by removing some subset of vertices and/or some subset of arcs/edges of G. Note that such a subgraph *may neither be spanning nor induced* in G.

Example 4.5.3 (Spanning subgraph):
Let us again see the graph of Figure 4.23. The graph G_3 is a spanning subgraph of the graph G. Note that the subgraph G_3 can be obtained from the graph G by deleting the arc set $\{ (1,4), (5,3), (2,2) \}$. The subgraph G_1 is *neither an induced nor a spanning subgraph* of the graph G. Note that the graph G_1 can be obtained from the graph G by deleting successively the vertex 3 and the directed edge $(4,1)$.

Example 4.5.4:
Let $G = (X, E)$ be a simple graph with n vertices and m edges. Then find the following:

a) the number of *spanning* subgraphs of G

b) the number of *induced* subgraphs of G

Solution: a) The number of subsets of a set of p elements is exactly 2^p. By definition, a spanning subgraph H of G is a subgraph of the form $H = (X, F)$ where F is any subset of the edge set E. Since there are exactly 2^m subsets of E (because $|E| = m$), there are exactly 2^m spanning subgraphs, one for each subset of E.

b) By definition, an induced subgraph H of G is a subgraph of the form $H = (S, F)$ where S is a subset of the vertex set X and F is the set of all edges of G having both of its end vertices in the set S. Since there are 2^n subsets of X(because $|X| = n$), there are 2^n induced subgraphs, one for each subset of X. If we exclude the *empty graph* then the number of induced subgraphs is $2^n - 1$.

f-factors

Consider a multigraph $G = (X, E)$ and a function f from the vertex set to the set of non-negative integers. An f-factor of G is a spanning subgraph H of G such that $d_H(x) = f(x)$ for all x in X. If $f(x) = 1$ for all vertices x then the f-factor becomes a 1-factor or perfect matching.

Graphs containing f-factors were characterized by Tutte (see [3]). Erdös' conjecture concerning the existence of spanning biregular subgraphs in a regular graph was elegantly proved by Tutte using his *f-factor theorem.*

THEOREM 4.5.1 (Conjectured by Erdös and proved by Tutte): Let G be a k-regular graph and let r an integer such that $r \leq k$. Then G contains a spanning graph whose degrees are either r or $r + 1$.

A slight extension of the above theorem was proved by Sriraman Sridharan using Lovasz's (g, f)-factor theorem.

(g,f)-factors

Consider a multigraph $G = (X, E)$ and two functions f and g from the vertex set to the set of non-negative integers. A (g, f)-factor of G is a spanning subgraph H of G such that $g(x) \leq d_H(x) \leq f(x)$ for all x in X. If $f(x) = g(x)$ for all vertices x then the (g, f)-factor becomes an f-factor or perfect f-matching.

Graphs containing (g, f)-factors were characterized by Lovasz. However, Tutte deduced Lovasz's theorem as a consequence of his f-factor theorem (see [3]).

Theorem 4.5.2 (Thomassen and independently by Sriraman Sridharan):
Let G be a graph whose degrees are either k or $k+1$ and let r be an integer such that $r < k$. Then G contains a spanning graph whose degrees are either r or $r+1$.

(For proofs of these two theorems, see the book by K.R. Parthasarathy [3]).

Given a loopless multigraph of m edges, is there any nontrivial lower bound on the number of edges of a *spanning* bipartite subgraph? The question is answered by the following theorem due to Erdös:

Theorem 4.5.3:
A loopless multigraph G of m edges contains a spanning bipartite subgraph of at least $m/2$ edges.

Variational Proof. If the theorem is true, then the *maximum* number of edges of a spanning bipartite subgraph will have at least $m/2$ edges. Let H be such a bipartite graph with bipartition X_1 and X_2. Since H has the maximum number of edges, all the edges of the graph G having one end vertex in X_1 and another end vertex in X_2 must be present in the graph H.

Assume that we have proved the following: The degree of each vertex x in the graph H is at least half the degree of x in the graph G, that is,

$$d_H(x) \geq (1/2)d_G(x).$$

Then by Corollary 4.1.1.1

$$\begin{aligned} 2m(H) &= \sum_{x \in X} d_H(x) \\ &\geq (1/2)\sum_{x \in X} d_G(x) \end{aligned}$$

$$\begin{aligned} &\geq (1/2)(2m(G)) \\ &\geq m(G) \end{aligned}$$

Therefore, $m(H) \geq (1/2)m(G)$.

Hence it remains to show that $d_H(x) \geq (1/2)d_G(x)$. If not, there is a vertex x with $d_H(x) < (1/2)d_G(x)$. We may assume that x is in X_1. This means that the degree of the vertex x in the induced subgraph

$$d_{G(X_1)}(x) > 1/2d_G(x) \tag{4.2}$$

We can now form a new spanning bipartite graph H' of G with the bipartition $X_1 \setminus x$ and $X_2 \cup x$, that is, the new bipartition is obtained by removing the vertex x from X_1 and adding the vertex x in X_2. But then, $m(H') > m(H)$ by the inequality 4.2, which is a contradiction to the maximality of the graph H. ∎

Remark 4.5.1 (Open problem):
Can we find a result "similar" to Theorem 4.5.3 for an induced bipartite graph? More precisely, what is the maximum number of vertices of an induced bipartite graph in a given graph?

4.6 Walks, Paths, Cycles

We would like to "navigate" in a graph from one vertex to another along the edges of the graph. To this end, we shall study the notions of walks, paths, and cycles in a graph.

Walk

A *walk* in a graph is a finite sequence of arcs $(u_1, u_2, \ldots, u_p)$ such that one of the extremities of u_i coincides with one of the extremities of u_{i+1} for $1 \leq i < p$. Put differently, in the sequence, the arcs u_i and u_{i+1} are adjacent for $1 \leq i < p$.

In the graph of Example 4.1.1, the sequence $(u_4, u_6, u_5, u_6, u_7)$ is a walk whose *origin or departure or initial vertex* is the vertex x_2 and whose *terminus or arrival or final vertex* is the vertex x_5.

In the definition of a walk, let us observe that the direction of arcs is immaterial. We can imagine a walk of a pedestrian from one point, a street intersection, to another point in a city map in which there are *no* one ways for pedestrians but having one ways for motorists, that is, the walk is taken in the underlying multigraph obtained by ignoring the directions of the arcs of the given graph.

In the case of 1-graph/simple graph, a walk can be specified as a sequence of vertices met by the walk, since between two vertices of a 1-graph/simple graph there is at most one arc/edge joining the two vertices.

Simple walk

A *simple walk* (also called *trail*) is a walk in which *no* two arcs are repeated. In other words, a pedestrian taking a walk from one point to another in a city map does not traverse the same *street* twice.

For example, in the graph of Example 4.1.1, the sequence (u_4, u_6, u_5) is a simple walk with its origin at the vertex x_2 and the terminus at the vertex x_1.

Elementary walk

An *elementary walk* (also called a *path*) is a walk in which no two vertices are repeated except possibly the origin and the terminus of the walk. Stated differently, a pedestrian taking a walk from one point to another in a city does not traverse the same *point* twice, except perhaps for coming back to his departure point.

A *cycle* is a trail in which the departure vertex and the arrival vertex coincide. In other words, a cycle is a *closed trail*.

For example, $(u_5, u_6, u_7, u_9, u_8)$ is a cycle in the graph of Example 4.1.1 with origin = terminus = x_4.

An *elementary cycle* is an elementary path in which the origin and terminus coincide.

The sequence (u_7, u_9, u_8) is an elementary cycle of Example 4.1.1.

Directed walk, directed path, directed cycle

A *directed walk* or simply *diwalk* in graph is a walk in which the directions of the arcs are taken into account, that is, it is a sequence of arcs $(u_1, u_2, \ldots, u_p)$ such that the extremity terminal of u_i coincides with the extremity initial of u_{i+1} for $1 \leq i < p$.

$(u_4, u_5, u_6, u_5, u_7)$ is diwalk in the graph of Example 4.1.1. Its origin is x_2 and its terminus is x_5. For a diwalk, one can imagine a motorist traveling from one point of a city to another by *respecting* the one-way streets.

A *simple diwalk* (called also directed trail or ditrail) is a diwalk in which no arc is repeated.

The sequence (u_4, u_5, u_6) is a directed trail of the graph of Example 4.1.1 with origin at x_2 and the terminus at x_1. A *directed path or elementary directed walk* or simply *dipath* is a diwalk in which no two vertices are repeated except possibly the departure and the arrival.

The sequence (u_3, u_4, u_5, u_7) is a dipath of the graph of Example 4.1.1.

A *circuit or dicycle* is a simple closed diwalk, that is, it is a simple walk in which the origin and the terminus coincide.

In Example 4.1.1, (u_1, u_{11}, u_{10}) is a circuit.

An *elementary circuit* is a directed path in which the origin and the terminus coincide.

For example, (u_1) and (u_5, u_6) are elementary circuits in Example 4.1.1. A shortest possible path/directed path with vertex x as origin and vertex y as terminus, if it exists, is called a $x - y$ *geodesic*. We consider the sequence (x) where x is any vertex of a graph as a walk/diwalk of *length* zero or empty walk.

Remark 4.6.1:

Note that an elementary cycle/circuit may start and end at any of its vertices. This is because there is exactly one elementary cycle/-circuit of length l up to isomorphism. For example, in the graph of Example 4.1.1, (u_7, u_9, u_8) is an elementary cycle whose origin and terminus are the vertex x_4. This cycle can also be specified by the sequence (u_9, u_8, u_7) with origin and terminus at x_5 or by the sequence (u_8, u_7, u_9). More generally, a *cyclic permutation* of the arcs/edges of a cycle/circuit does not affect the cycle.

Table 4.1 summarizes the concepts of walk/directed walk in digraphs/multigraphs.

Table 4.1: Walks/diwalks in multigraphs/digraphs

Multigraph/Simple Graph (Direction of arcs immaterial)	Directed Graph (Direction of arcs matters)
Walk (vertices and edges may be repeated)	Directed Walk (vertices and arcs may be repeated)
Simple Walk (edges are distinct)	Directed trail or Simple Directed walk (arcs are distinct)
Elementary Walk or Path (vertices and hence edges are distinct)	Directed Path (vertices and hence arcs are distinct)
Cycle (edges are distinct)	Circuit (arcs are distinct)
Elementary Cycle (vertices are distinct)	Elementary Circuit (vertices are distinct)

We now prove a simple proposition concerning walk.

Proposition 4.6.1:
If there is a directed walk from vertex x to vertex y in a directed graph, then there is an *elementary* directed path from vertex x to vertex y.

Proof. Intuitively, if we remove all the "redundant" arcs from a directed walk with origin x and terminus y, we will be left out with a directed elementary path. In fact, a geodesic (shortest dipath) from the vertex x to the vertex y satisfies the proposition.

Let $W = (u_1, u_2, \ldots, u_p)$ be a directed walk of *minimum* possible length p from $x = x_1$ to $y = x_{p+1}$ with $u_i = (x_i, x_{i+1})$ for $1 \leq i \leq p$. If W is not an elementary walk, then a vertex must be repeated in the walk W. Let x_k be a repeated vertex in W (see

Figure 4.24). Note that a *shortest* repeated portion of W from x_k to x_k is a directed elementary cycle. Now the removal of all the arcs of this directed cycle together with all vertices of the directed cycle except x_k still gives a diwalk from x to y whose length is $< p$, a contradiction to the minimality of p. ■

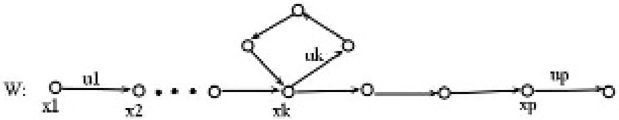


Figure 4.24: A walk W with repeated vertex x_k

Proposition 4.6.2:
Every simple graph with the degree of each vertex of at least two ($\delta \geq 2$) contains an elementary cycle of length $\delta + 1$.

Proof. Consider a *longest* possible (elementary) path P in the graph. (The length of a path is the number of edges in the path.) Let x_0 be the initial vertex of the path $P = (x_0, x_1, \ldots, x_l)$ and consider the vertices of the graph adjacent to the initial vertex x_0. Since the degree of $x_0 \geq 2$, there is a vertex $y \neq x_1$ such that x_0y is an edge of the graph.

If $y \neq x_i$ for $2 \leq i \leq l$, then we have a path $(y, x_0, x_1, \ldots, x_l)$ whose length $l + 1$ exceeds that of P, a contradiction.

Then we must have $y = x_i$ for some i with $2 \leq i \leq l$ with x_0x_i is an edge of the graph. But then we have the elementary cycle $(x_0, x_1, \ldots, x_i)$ (since x_0x_i is an edge of the graph). In fact, if i is the largest such index, then $i \geq \delta$. But then $(x_0, x_1, \ldots, x_i)$ is an elementary cycle of length at least $\delta + 1$. ■

4.7 Connectedness

Intuitively, a graph is connected if one draws the whole graph without lifting the pen from the paper, in other words, the graph may be drawn in one continuous pen stroke, a vertex and an arc/ edge may be traced more than once. If the vertices represent some objects and arcs/edges represent strings, then a connected graph will remain as *one piece* if picked by any one of the vertices. For example, in an electric circuit, circuit elements like transistors, resistors, and capacitors are connected by wires in various ways. We may ask the question: Is everything connected?

More formally, a graph is *connected* if there is an elementary path between any two vertices of the graph. A *disconnected graph* G consists of more than one vertex disjoint connected subgraphs, called the *connected components* of G. The connected components of a graph G *partition* the graph into maximal connected subgraphs of G. Note that for connectivity of a graph, the orientations of arcs are immaterial.

Example 4.7.1 (A disconnected graph):
The graph G of Figure 4.25 is a disconnected multigraph with four connected components. On the other hand, the Petersen graph is a connected simple graph.

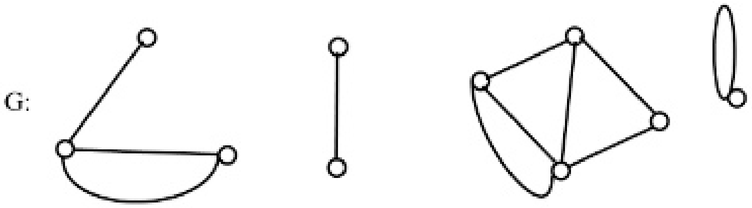


Figure 4.25: A disconnected multigraph graph with 4 components

Remark 4.7.1:
While proving theorems on graph theory, we normally use the following sentence: If the graph G is not connected then we can consider the connected components of G separately. However, there are exceptions like the famous *reconstruction conjecture* of Ulam (see [3]).

Distance in graphs/multigraphs

In a graph/multigraph, the distance between two vertices, $d(x, y)$, is the *minimum* number of arcs/edges of a path/directed path from the vertex x to the vertex y. If no such path/directed path exists then we set $d(x, y) = \infty$.

In a connected multigraph G, distance is a metric; that is, for all vertices x, y, and z, we have the following three properties:

1. $d(x, x) = 0$ and $d(x, y) > 0$ if $x \neq y$.
2. $d(x, y) = d(y, x)$.
3. $d(x, y) + d(y, z) \geq d(x, z)$ (triangle inequality).

The *diameter*, $d(G)$, of a graph G is $\max(d(x, y) \mid x, y \in X)$, in other words, the diameter of G is the length of any *longest geodesic* in G.

A simple graph is called a *geodetic graph* if there is a unique shortest path between any two vertices of the graph. For example, the reader can verify that the Petersen graph is geodetic whereas an elementary cycle of length 4 is *not* geodetic. More generally, an elementary cycle of even length is *not* geodetic whereas an elementary cycle of *odd* length is geodetic.

Geodetic graphs have been characterized by Parthasarathy and Srinivasan (see [3]).

Example 4.7.2:

If G is a simple graph on n vertices with its minimum degree $\delta(G) \geq (n - 1)/2$, then prove that the diameter of G is less than or equal to 2.

Solution: Consider any two distinct vertices x, y of G. We shall show that the distance between them is less than or equal to 2. We distinguish two cases.

Case 1: The vertices x and y are adjacent in G.

In this case, the distance $d(x, y) = 1$ and we are done.

Case 2: The vertices x and y are *not* adjacent.

In this case, we prove the existence of a third vertex z such that xz and zy are edges of G, that is, $z \in \Gamma(x) \cap \Gamma(y)$. If not, we have $\Gamma(x) \cap \Gamma(y) = \emptyset$. This means that the number of vertices of the graph G is at least

$$|\Gamma(x)| + |\Gamma(y)| + |\{ x \}| + |\{ y \}|.$$

Since the minimum degree of G is $\geq (n-1)/2$, and $|\Gamma(v)| = d(v)$ for all v, we get the number of vertices of G is at least $(n-1)/2 + (n-1)/2 + 1 + 1 = n + 1$, which is impossible. Therefore there must be a vertex z joined to both x and y and the elementary path (x, z, y) is a path of length 2 from x to y. This means that $d(x, y) = 2$.

All cases have been considered and the proof is complete.

Example 4.7.3:
Prove that if $G = (X, E)$ is a simple graph of diameter > 3 then the diameter of the complement graph G^c is < 3.

Solution: Consider any two distinct vertices x, y of G. We shall show that the distance between them is less than or equal to 2 *in* the complement G^c. Two cases can arise:

Case 1. The vertices x and y are *not* adjacent *in* the graph G. Then, by the definition of the complement, the two vertices x and y are joined by an edge in G^c and hence $d(x, y) = 1$ *in* G^c.

Case 2. The vertices x and y are adjacent *in* G.

We shall show that there is a third vertex z such that both xz and zy are *not* edges *in* G. Suppose the contrary. Then every vertex z distinct from x and y is joined to *at least* one of the vertices x and y *in* G. This means that any two vertices of G are joined by an elementary path of length ≤ 3 and hence the diameter of G is less than or equal to 3, a contradiction to our hypothesis. Hence there is a third vertex z such that both xz and zy are not edges in G. By the definition of the complement, xz and zy are edges *in* G^c. This means that the elementary path (x, z, y) *in* G^c is a path of length 2 and $d(x, y) = 2$. Therefore the diameter of G^c in this case is exactly 2.

A good characterization of non-bipartite graphs

Consider Figure 4.26:

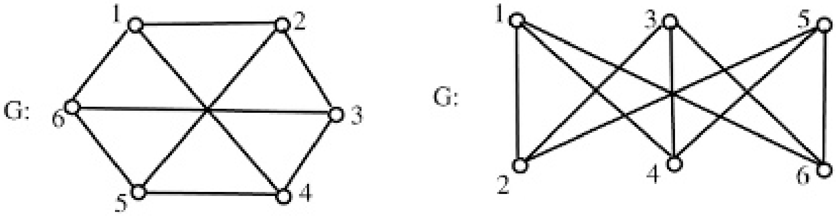


Figure 4.26: A bigraph and its bipartite drawing

From the geometric representation of the first graph G is it possible to assert "quickly" that the first graph G of Figure 4.26 bipartite? Of course, if we "exhibit" the *same* graph as indicated by the second graph G, it is clearly (by using our eye-ball!) bipartite because every edge of the graph possesses one end in $X_1 = \{1, 3, 5\}$ and another end in $X_2 = \{2, 4, 6\}$.

Conversely, we raise the following question: given a graph can we assert "quickly," that the graph is non-bipartite? For example, it can be verified easily that an elementary cycle of length three (a triangle) is not bipartite. More generally, one can prove that an elementary cycle of *odd* length (also called an odd elementary cycle) is non-bipartite.

The following theorem of König tells us that an elementary odd cycle is the only type of graph that a given graph G should not possess as a subgraph, in order to be bipartite. Put differently, *odd* elementary cycles are the only *forbidden* structures of a bipartite graph.

Theorem 4.7.1:
A graph G is *not* bipartite if and only if it contains an odd elementary cycle. Equivalently, a graph G is bipartite if and only if

each of its elementary cycle is of even length.

Proof. If G is a bipartite graph, then we have to prove that the graph contains *no* odd elementary cycle; in other words, we must prove that every elementary cycle must be of *even* length.

Since G is bipartite, by definition, its vertex set can be partitioned into two subsets X_1 and X_2 such that every edge has one end in X_1 and another end in X_2.

Let $C = (e_1, e_2, \dots, e_p)$ be an elementary cycle of length p. Then we have to show that p is an even integer. Let the edge e_i join the vertices x_i and x_{i+1}, that is, $e_i = x_i x_{i+1}$ for $1 \leq i \leq p$. Since C is a cycle, we have $x_1 = x_{p+1}$, that is, the origin and terminus of C coincide. We may assume that the origin x_1 of the cycle C is in the set X_1. Then, since G is bipartite, x_2 must be in X_2, x_3 must be in X_1, etc., because $x_i x_{i+1}$ is an edge, for $1 \leq i \leq p$. This means that the oddly subscripted vertices are in X_1, and the evenly subscripted vertices are in X_2. Since $x_1 = x_{p+1}$ is in the set X_1, we must have $p+1$ an odd integer and hence p is an even integer.

Now for the second part of the theorem, assume that the graph $G = (X, E)$ contains no odd elementary cycle. We have to prove that G is bipartite.

We may assume without loss of generality that the graph is connected, for otherwise, we can apply the following argument for the connected components of G separately (see Remark 4.7.1). (If $G_i = (X_i^{(1)} \cup X_i^{(2)}, E_i)$ are vertex disjoint bipartite graphs for $1 = 1, 2, \dots, k$, then their union is a bipartite graph with bipartition $\cup_{i=1}^{k} X_i^{(1)}$ and $\cup_{i=1}^{k} X_i^{(2)}$.)

The proof is *constructive* in nature, that is, we are going to give an *algorithm* to find a bipartition of the graph G. Consider any vertex x_1 in X. Define a bipartition of the vertex set X as follows: Let X_1 consist of all vertices at *even* distance from the vertex x_1 and $X_2 = X \setminus X_1$, in notation,

$$X_1 = \{\, x \in X \mid d(x_1, x) \text{ is even} \,\} \text{ and } X_2 = X \setminus X_1 =$$

$$\{\, x \in X \mid d(x_1, x) \text{ is odd} \,\}.$$

We claim that X_1 and X_2 is a desired bipartition. We shall show that every edge of G joins a vertex of X_1 and a vertex of X_2. For suppose there is an edge $e = xy$ joining two points of X_1. We will obtain a contradiction.

Consider the smallest possible length paths P_1 from x_1 to x, and P_2 from x_1 to y. By the definition of X_1, $d(x_1, x)$ and $d(x_1, y)$ are both even, that is, the number of edges in the paths P_1 and P_2 are even. These paths may have vertices and/or edges in common. Let z be the *last* vertex common to the paths P_1 and P_2. This last vertex may be the vertex x_1. *A portion of a smallest path is always a smallest path.* Hence the subpath of P_1 from x_1 to z and the subpath from x_1 to z of P_2 have the same length; put differently, $d(x_1, z)$ along the path P_1 and $d(x_1, z)$ along the path P_2 are equal. Since both $d(x_1, x)$ and $d(x_1, y)$ are even, we have both $d(z, x)$ and $d(z, y)$ have the *same parity.* But then, the edge-induced subgraph induced by the subpath of P_1 from z to x and the subpath of P_2 from z to y and the edge xy is an odd elementary cycle of G, a contradiction to the fact that G has no elementary cycle of odd length. Similarly, we can prove that there are no edges between any two vertices of X_2. ■

We have the intuitive feeling that if a *simple* graph contains a "lot" of edges, then it must be connected. The following proposition captures this intuition.

Proposition 4.7.1:
A simple graph on n vertices and $m > \binom{n-1}{2}$ edges is connected. Moreover the result is the best possible in the sense that for every $n > 1$ there is a disconnected graph on n vertices with exactly $\binom{n-1}{2}$ edges.

Proof. The proof is by contradiction.

If the graph G is disconnected, then it is the disjoint union of mutually vertex-disjoint connected subgraphs of G. Let the connected components of G be G_1, G_2, …, G_k where $k \geq 2$.

Set $n(G_i) = n_i$, the number of vertices of G_i and $m(G_i)) = m_i$, the number of edges of G_i for $1 \leq i \leq k$.

Then clearly, $n = n_1 + n_2 + \cdots + n_k$ and $m = m_1 + m_2 + \cdots + m_k$.

A simple graph on p vertices has at most $\binom{p}{2}$ edges, since every simple graph on p vertices is a spanning subgraph of the complete graph K_p. Hence $m_i \leq \binom{n_i}{2}$ for $1 \leq i \leq k$.

Therefore, $m = \sum_{i=1}^{k} m_i \leq \sum_{i=1}^{k} \binom{n_i}{2}$.

We shall now prove that $\sum_{i=1}^{k} \binom{n_i}{2} \leq \binom{n-1}{2}$ for $k \geq 2$ and $n_i \geq 1$.

To prove the inequality $\sum_{i=1}^{k} \binom{n_i}{2} \leq \binom{n-1}{2}$, we proceed by induction on k.

Basis: $k = 2$.

We have to prove that $\binom{n_1}{2} + \binom{n_2}{2} \leq \binom{n_1+n_2-1}{2}$.

That is, to prove, $n_1(n_1 - 1)/2 + n_2(n_2 - 1)/2 \leq (n_1 + n_2 - 1)(n_1 + n_2 - 2)/2$.

That is, to prove, $n_1 + n_2 \leq n_1 n_2 + 1$, which is always true because $n_1 \geq 1$ and $n_2 \geq 1$, that is, the sum of two positive integers is always less than or equal to their product plus one.

Induction hypothesis: Assume

$$\sum_{i=1}^{k} \binom{n_i}{2} \leq \binom{\sum_{i=1}^{k} n_i - 1}{2}.$$

We shall prove the inequality for $k + 1$.

$$\begin{array}{rcl} \displaystyle\sum_{i=1}^{k+1} \binom{n_i}{2} & = & \displaystyle\sum_{i=1}^{k} \binom{n_i}{2} + \binom{n_{k+1}}{2} \\ \textit{by induction hypothesis} & \leq & \displaystyle\binom{\sum_{i=1}^{k} n_i - 1}{2} + \binom{n_{k+1}}{2} \\ \textit{by basis} & \leq & \displaystyle\binom{\sum_{i=1}^{k} n_i - 1 + n_{k+1} - 1}{2} \\ & \leq & \displaystyle\binom{\sum_{i=1}^{k+1} n_i - 2}{2} \\ \textit{which is clearly} & \leq & \displaystyle\binom{\sum_{i=1}^{k+1} n_i - 1}{2}. \end{array}$$

Finally, the graph $K_{n-1} \cup \{ x_n \}$, the complete graph on $n - 1$ vertices plus an isolated vertex x_n is disconnected with two connected components for $n > 1$ and its number of edges is $\binom{n-1}{2}$. ■

4.8 Graphs and Puzzles

Graph theory and recreational mathematics were intimately related in the beginning of the subject. In fact, the origin of graph theory can be traced to the famous *Königsberg Bridge problem* solved by Euler in the year 1736. The problem can be stated as follows:

Königsberg bridge problem

In the city of Königsberg, now called Kaliningrad, there were two islands I_1, I_2 and two banks B_1 and B_2 of the River Pregel connected by seven bridges (see Figure 4.27). The problem is the following: Is it possible for a pedestrian starting at any of the four land areas I_1, I_2, B_1, B_2 to cross each of the seven bridges *exactly once* (no condition is imposed on coming back to the starting point)?

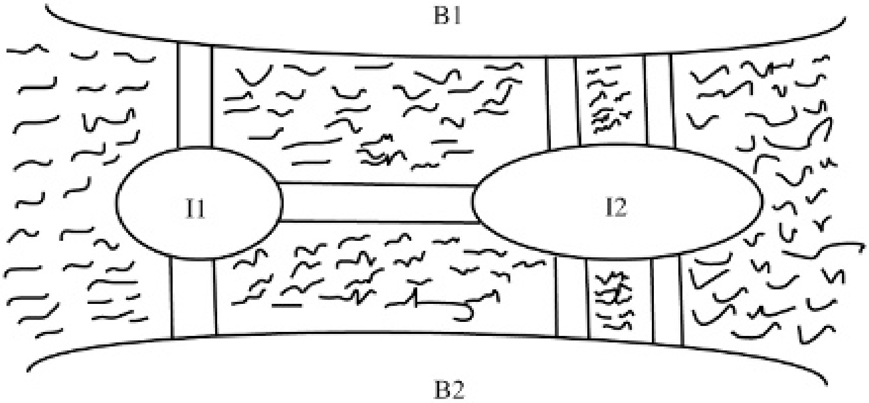


Figure 4.27: Seven bridges of Königsberg

We can model the problem by means of a multigraph. We represent the four land areas by four distinct vertices and the bridges by the edges joining the corresponding land areas (see Figure 4.28).

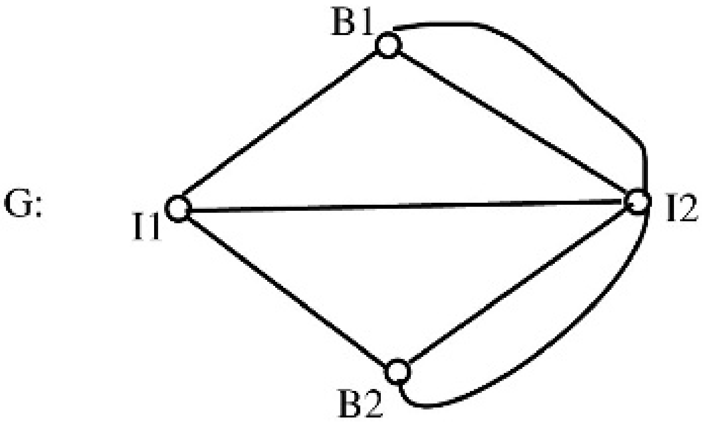


Figure 4.28: The graph corresponding to the seven bridges of Königsberg

In terms of the above multigraph, the problem can be reformulated as follows:

Can we draw the graph of Figure 4.28 at one single stroke? Stated differently, is it possible to draw the graph of Figure 4.28 without lifting the pen and without going through the same edge more than once? The correspondence between taking a walk around the bridges, without crossing a bridge more than once and drawing the corresponding multigraph without lifting the pen and without tracing the same edge more than once is clear.

After some trial-and-error attempts, we may conclude that such a drawing is impossible. Why? Is it possible to convince somebody that such a drawing is impossible, by merely looking at the graph? To our surprise the answer is "yes," because of the following result due to Euler (see Berge [1]).

THEOREM 4.8.1 (Euler):
Such a drawing is *impossible*, if and only if either the graph is *not* connected or there are at least four vertices of *odd* degree.

In fact, the multigraph of Figure 4.28 associated with the

Königsberg bridges have all of its four vertices possessing an odd degree. Hence, a walk around the seven bridges of Königsberg is impossible.

Good characterization

Euler's characterization given above is an example of a *good characterization* of multigraphs without admitting a drawing at one single pen stroke. If a multigraph can be drawn at one stroke, then the possibility of such a drawing can be "exhibited" by drawing the graph G with the stipulated condition. If it can't be drawn, then the impossibility of such a drawing can be proved by finding four vertices of a multigraph having odd degree.

Thus, we are able to convince somebody "quickly," that is, in polynomial time, of a possibility of such a drawing or the impossibility of such a drawing. Such a characterization is called a *"good characterization."* In somewhat more formal manner, let us define the notion of a good characterization:

Definition 4.8.1 (Good characterization):
Consider a property "P" of a graph or concerning any mathematical structure. The property "P" is referred to as an *"NP-property"* whenever the existence of the property "P" can be exhibited "easily" (in polynomial time in the size of the input) for any graph/mathematical structure verifying "P." For example, the compositeness of an integer $n \geq 4$ is an NP-property because to "convince" someone that n is composite, it is enough to furnish him or her with a divisor d of n other than 1 and n. The divisor d is referred to as a *succinct certificate* for the compositeness of n. "Planarity" of a given graph G is an NP-property for if G is planar then it can be easily exhibited by drawing G in the plane in such a way that the edges do not intersect at a point other than at a vertex of the graph. A good characterization is one which establishes an *equivalence* between an NP-property and a *negation* of another NP-property. Finally, a property P is *well-characterized* if we can exhibit *both* the existence of P *and* the non-existence of

P "easily."

Example 4.8.1:
In the example (see Figure 4.29) the first graph G_1 is impossible to draw at one stroke whereas the other two graphs G_2 and G_3 admit such a drawing.

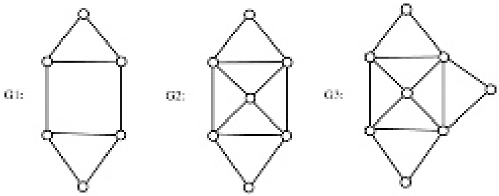


Figure 4.29: G_1, impossible to draw at one stroke, whereas G_2, G_3 admit such a drawing

If a graph/multigraph can be drawn at one stroke, then such a drawing is called a Eulerian directed trail/trail, that is, a Eulerian directed trail/Eulerian trail is a directed trail/trail passing through each arc/edge exactly once. If in addition, the directed trail/trail is closed, that is, the origin and terminus coincide, then such a directed trail/trail is called a *Eulerian circuit/Eulerian cycle*. A graph/multigraph admitting a Eulerian circuit/Eulerian cycle is called a Eulerian graph/multigraph.

The following is a characterization of graphs possessing Eulerian circuits and is due to I.J. Good (see Knuth [8]).

Theorem 4.8.2 (Good):
A graph possesses a Eulerian circuit if and only if it is connected and every vertex has the same in-degree as its out-degree.

Proof. One part of the theorem is easy to prove.

If G possesses a circuit passing through each arc exactly once, then every vertex x lies in the circuit. Hence, in the circuit for every arc leading into the vertex x, there is a different arc leading away from the vertex x. Therefore there is a *bijection* between the set of all arcs leading into x onto the set of all arcs leading away from x. In particular, $d^+(x) = d^-(x)$.

Conversely, assume that the graph G is connected and $d^+(x) = d^-(x)$ for each vertex x. We shall show that G possesses a Eulerian circuit. If a Eulerian circuit exists, then it must be a directed *trail* (simple walk) of *maximum* possible length m, because a Eulerian circuit passes through each arc exactly once. Let

$$T = (u_1, u_2, \ldots, u_p)$$

be a directed trail of *longest* possible length. Note that $head(u_{i-1}) = tail(u_i)$ for all i, with $2 \leq i \leq p$. We shall first prove that T is closed, that is, we shall prove that the head of last arc u_p is the same as the tail of first arc u_1.

If x is the head of the arc u_p, then all the $k = d^+(x)$ arcs leading out of the vertex x should appear in the directed trail T; since otherwise, we could add that arc u does not appear in the directed trail T at the terminus of T to get a longer directed trail than T. Since the integer p is the *largest* subscript such that arc u_p lies in T, we have the subscript i of every arc u_i leading out of the vertex x satisfies the inequality $1 \leq i \leq (p-1)$. If u_1 is one such arc, then T is closed. Otherwise, the subscript i of every arc u_i leading out of the vertex x satisfies the inequality $2 \leq i \leq (p-1)$. But for every such arc u_i, u_{i-1} exists because $i-1 \geq 1$ and $\leq (p-2)$ and the head of u_{i-1} coincides with the tail of u_i. Since the arc u_p leads into the vertex x, and p is the maximum index, these imply that the in-degree of the vertex x is at least $k+1$, a contradiction to the fact that $d^+(x) = d^-(x)$. Hence T is a closed directed trail, that is, a closed circuit.

It remains to prove that the circuit T includes *all* the arcs of the graph G. By Remark 4.6.1, a circuit may start and end at any if its vertices, that is, we can take any of its vertex vertices as origin and terminus. Assume an arc $u = (s, t)$ of the graph G not in the circuit T. Then this arc has neither its head t nor its tail s in T, since otherwise we can add this arc to T to have a longer directed trail than T. So if the circuit T does not include all the arcs of the graph G, then G is not connected. ∎

4.8.1 An Application

PROPOSITION 4.8.1:
Consider a connected digraph G with arcs $u_0, u_1, \ldots, u_m$. Let $p_0, p_1, \ldots, p_m$ be the positive integers. Let p_i be the weight associated with the arc u_i for $0 \leq i \leq m$. Assume that the sum of the weights of the arcs leading into each vertex x is equal to the sum of the weights of the arcs leading away from the vertex x, that is,

$$\sum_{head(u_i)=x} p_i = \sum_{tail(u_i)=x} p_i, \text{ for each } \text{ vertex } x \text{ of } G.$$

The above condition is called *Kirchoff's law.*

Assume further that $p_0 = 1$. Show that there is a directed walk (neither elementary nor simple) in G from the head of u_0 to the tail of u_0 such that the arc does not occur in the path, and for $1 \leq i \leq m$ arc u_i occurs exactly p_i times.

Proof. We are going to apply Theorem 4.8.2 to a suitably constructed digraph.

Construct a new directed graph G' having the same vertex set as that of G and having p_i copies of the arc u_i for $0 \leq i \leq m$, that is, each arc u_i of G is replicated p_i times in the new graph G' (see Figure 4.30).

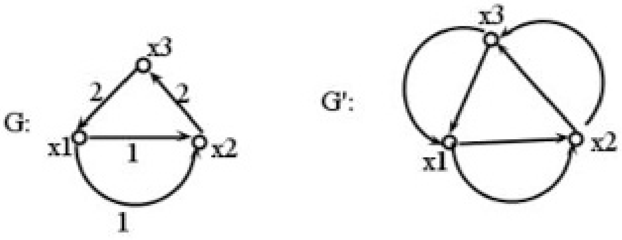


Figure 4.30: Graph G and G'. Weights are written on the edges of G.

Because of Kirchoff's condition in the graph G, in the newly constructed digraph G', every vertex has the same in-degree as its out-degree. Hence G' is a connected digraph in which $d^{+}(x) = d^{-}(x)$ for each vertex x. Therefore by Theorem 4.8.2 G' possesses a Eulerian circuit $(u_0, \ldots,)$ (recall that a circuit may start and end at any of its arcs) and u_0 occurs exactly once on this Eulerian circuit (since $p_0 = 1$).

The desired directed path is obtained by removing the arc u_0 from this Eulerian circuit. ■

4.8.2 Two Friendship Theorems

Proposition 4.8.2:
Show that, in any group of two or more people, there are always two people having the same number of friends inside the group. We assume friendship is a symmetric relation.

Proof. This problem can be modeled using a *simple* graph.

We construct a simple graph where the vertices correspond to different people and two vertices are adjacent if the corresponding people are friends. In this graph, the degree of a vertex x represents the number of friends of the person represented by x.

In graphical terms, the proposition is stated as follows:

Prove that, in any simple graph, there exist two distinct vertices x and y having the same degree, that is, $d(x) = d(y)$. This is a direct application of the pigeon-hole principle.

> Pigeon-hole principle: Of three *ordinary* people, two must have the same sex!
>
> *D. Kleitman*

More generally, if $(n + 1)$ letters are distributed among n letter boxes, then at least one letter box will have at least two letters.

Let us first observe that in a simple graph on n vertices, the degree of a vertex can assume at most n integers, $0, 1, \ldots, n - 1$.

We distinguish two cases:

Case 1: There is a vertex x such that $d(x) = 0$.

Then, we cannot have a vertex of degree $n-1$, that is, a vertex joined to all other vertices of the graph. More precisely, we must have $0 \leq d(y) \leq n-2$, for each vertex y. But then the possible degrees of a vertex are $n-1$ in number and we have n vertices. Hence by the pigeon-hole principle, there are two vertices having the same degree. (Here, the letters are the vertices and the n letter boxes labeled with $0, 1, \ldots, n-1$. A vertex of degree i goes into the letter box labeled i.)

Case 2: The degree of each vertex is > 0.

In this case also the possible degrees that a vertex can assume is $n-1$ in number. More precisely, the possible degrees of a vertex are: $1, 2, \ldots, n-1$. But the number of vertices is n. Therefore, there are two vertices possessing the same degree. (Here again the letters are the vertices and the letter boxes are labeled with the integers: $1, 2, \ldots, n-1$. A vertex of degree i goes into the letter box labeled i.) ■

Proposition 4.8.3:
Prove that in any group of six people, there are either three mutual friends or three mutual strangers. Here again we assume that the friendship is a symmetric relation. (Note that three mutual friends and three mutual strangers are not complementary to each other!)

Proof. As in the proof of the previous theorem, we construct a simple graph as follows:

We construct a simple graph G on six vertices: The six vertices represent the six people and two vertices are joined by a *solid* edge if the corresponding people know each other and we join two vertices by a *dotted* edge if the corresponding people *do not* know each other.

The puzzle is equivalent to the following: Any simple graph on six vertices contains either a solid elementary cycle of length three, that is, a solid triangle, or contains a dotted elementary cycle of length three, that is, a dotted triangle.

Consider a vertex x. Clearly, the number of solid edges incident with x plus the number of dotted edges incident with the same vertex x is exactly 5. Hence, either the number of solid edges at x

must be ≥ 3 or the number of dotted edges at x must be ≥ 3, for if not, the number of solid edges at x *plus* the number of dotted edges at x is ≤ 4. We may assume that there are at least three solid edges incident with the vertex x (See Figure 4.31); the case of at least three dotted edges can be treated similarly.

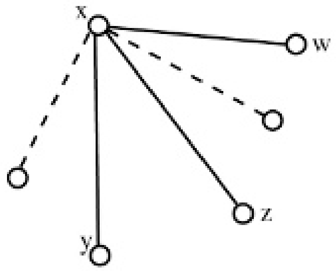


Figure 4.31: A step in the proof

If any two of the vertices, say, y and z in the neighbor set $\Gamma(x)$, are joined by an edge, then the vertices x, y, z induce a solid triangle (see Figure 4.32).

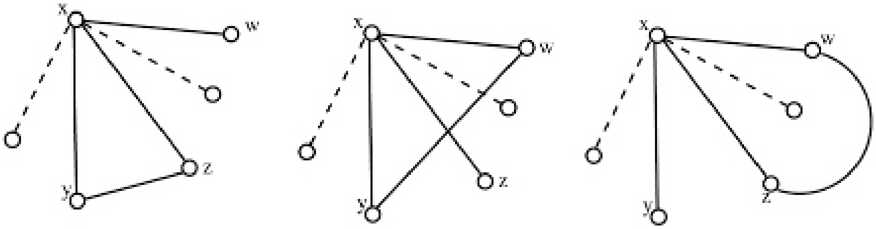


Figure 4.32: A step in the proof

If no two of the vertices in the neighbor set of x are adjacent, then, clearly the induced subgraph $G(\Gamma(x))$ contains a dotted triangle (see Figure 4.33). ■

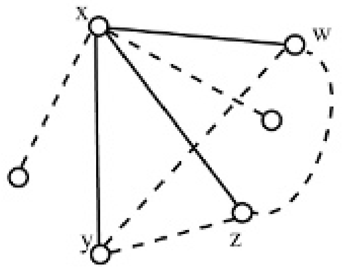


Figure 4.33: A step in the proof

4.8.3 Pandava Princes Problem and 3 Houses, 3 Utilities Problem

Example 4.8.2 (Pandava princes problem):
After the Kurushetra War in Bharat (now called India, this war dates back to 3000 BC according to historians), the five Pandava princes who won the war wanted to divide the kingdom, which is a single connected piece, into five "connected regions," in such a way that each region must have a common frontier (which is not a single point) with four other regions. Is it possible to share the kingdom in such a way? If so, how?

Solution: Draw a simple graph with five vertices corresponding to five provinces and two vertices are joined by an edge if and only if the corresponding regions have a common frontier. If such a sharing is possible, then in the graph thus constructed, each vertex is joined to four other vertices. This is nothing but a complete graph K_5 on five vertices.

The problem is now equivalent to drawing the complete graph on five vertices K_5 in a plane such that no two edges cross apart from their vertices.

After some trial and error you will see that such a drawing is impossible (see [3][1][5])! In other words, K_5 is a non-planar graph.

Example 4.8.3 (3 houses and 3 utilities problem):
We would like to connect all three houses (villas) a,b,c to all three factories e,g,w producing, respectively, electricity, gas and water. Is it possible to connect all three houses to all the three factories, so that the connecting lines/pipes (no underground connection!) do not cross in between? Assume that the houses and the factories are in the same plane.

Solution: As in Example 4.8.2, construct a graph with six vertices a,b,c,e,g,w and join two vertices if and only if one vertex is a house and another vertex is a factory. Since all three services, electricity, gas and water, are required by each house, we get a complete bipartite graph $K_{3,3}$. The problem is now: Is it possible to draw the graph $K_{3,3}$ in plane such that two edges do not cross, other than at a vertex? After some trial and error you will see that such a drawing is impossible. In other words, the graph $K_{3,3}$ is non-planar (see [1][3][5]).

4.9 Ramsey Numbers

Ramsey numbers can be considered as a profound generalization of the *pigeon-hole principle*:

If we partition a set of a sufficiently large number of elements into a sufficiently small number of subsets, then at least one of the sets of the partition must contain many of the elements of the original set.

The above proposition states that in any graph with *at least* six vertices, either there is a *stable set* of three vertices or *a clique* of three vertices, that is, an elementary cycle of length 3 (since a graph of at least six vertices contains a subgraph of exactly six vertices.)

Definition 4.9.1 (Ramsey numbers):
We may raise the following question: What is the *minimum* integer $r(p,q)$ such that *every* simple graph on $r(p,q)$ vertices contains either *a stable set* of p vertices or *a clique* of q vertices? The existence of such numbers $r(p,q)$ for all positive integers p and q

was proved by *Ramsey* and the numbers $r(p, q)$ are called *Ramsey numbers*.

In the following example we show that $r(3, 3) = 6$.

Example 4.9.1:
Consider the elementary cycle C_5. The reader can verify that C_5 contains neither a clique of three vertices nor a stable set of three vertices. Hence by the definition of $r(3, 3)$ we have $r(3, 3) \geq 6$.

But the above Proposition 4.8.3 implies that $r(3, 3) \leq 6$.

Combining the two inequalities, we have the result.

Only a few Ramsey numbers are known and in general finding the Ramsey numbers is a difficult unsolved problem.

Example 4.9.2 (Symmetry of Ramsey numbers):
Show that $r(p, q) = r(q, p)$.

Solution: Let G be a simple graph on $r(p, q)$ vertices. Then by the definition of $r(p, q)$, G contains either a stable set of p vertices or a clique of q vertices and the number of vertices of G is a graph of the *minimum* number of vertices satisfying this property. Now consider the complement G^c of the graph G. By the definition of G^c, a set S of vertices in G is a stable set in G, if and only if it is a clique in G^c and a set K of vertices in G is a clique in G, if and only if it is a stable set in G^c. Since G and G^c have the same set of vertices, we have by the definition of the Ramsey numbers, $r(p, q) = r(q, p)$.

Because of the symmetry of the Ramsey numbers, we can interchange the role of stable set and clique in the definition of Ramsey numbers.

The following theorem shows the existence of the Ramsey numbers. To prove the existence, we use induction. The following simple identity, the so-called Pascal identity involving binomial coefficients, is used as an *induction jump.*

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k} \; k \geq 1 \text{ and } n \geq 0$$

The following theorem can be viewed as a profound generalization of the pigeon-hole principle. The theorem intuitively states: A complete disorder is impossible.

THEOREM 4.9.1 (Ramsey):
Let G be any simple graph on $n = \binom{p+q}{p}$ vertices. Then G contains either a clique of $p+1$ vertices or a stable set of $q+1$ vertices.

Proof. The theorem is proved by induction on the sum $p+q$.

Induction Basis. For $p+q = 1$, we must have either $p = 0$ or $p = 1$. Then the number of vertices $n = 1$ and G consists of only one vertex and no edge. Clearly, G satisfies the theorem in this case. If $p = q = 1$ then the graph G has exactly two vertices. There are only two simple graphs with two vertices: 1. The complete graph K_2 on two vertices 2. The graph with two vertices and no edge. These two graphs satisfy the theorem. So we may suppose that $p > 1$ and $q > 1$.

Induction hypothesis. Suppose the theorem is true for all simple graphs with strictly less than $\binom{p+q}{p}$ vertices.

Take a simple graph on exactly $n = \binom{p+q}{p}$ vertices and a vertex x of G. By the definition of the complement of the graph G, we have

$$d_G(x) + d_{G^c}(x) = n - 1 = \binom{p+q}{p} - 1.$$

By the Pascal identity involving binomial coefficients, we have

$$d_G(x) + d_{G^c}(x) = \binom{p+q}{p} - 1 = \binom{p+q-1}{p-1} + \binom{p+q-1}{p} - 1.$$

Hence, we must have either $d_G(x) \geq \binom{p+q-1}{p-1}$ or $d_{G^c}(x) \geq \binom{p+q-1}{p}$ for, if both are not true, then $d_G(x) \leq \binom{p+q-1}{p-1} - 1$ and $d_{G^c}(x) \leq \binom{p+q-1}{p} - 1$, then by addition, we obtain $d_G(x) + d_{G^c}(x) \leq \binom{p+q-1}{p-1} + \binom{p+q-1}{p} - 2$, which is a contradiction.

Suppose $d_G(x) \geq \binom{p+q-1}{p-1}$. Take a subset S of the *neighbors* of the vertex x containing exactly $\binom{(p-1)+q}{p-1}$ vertices. Then by the induction hypothesis, the induced subgraph $< S >$ contains either a clique of $p-1$ vertices or a stable set of q vertices. But then the induced subgraph $< S \cup \{ x \} >$ contains a clique of p vertices or a stable set of q vertices because x is joined to all the vertices of S.

Now suppose $d_{G^c}(x) \geq \binom{p+q-1}{p}$. Take a subset T of vertices belonging to the neighbors of x in the complement graph G^c consisting of exactly $\binom{p+q-1}{p}$ vertices, which is, by the duality relation of the binomial coefficients, equal to $\binom{(q-1)+p}{q-1}$. (Duality relation of binomial coefficients: $\binom{n}{k} = \binom{n}{n-k}$.) By induction hypothesis, the induced subgraph $< T >$ of G^c contains either a clique of $q-1$ vertices or a stable set of p vertices. Hence, $T \cup \{ x \}$ contains either a clique of q vertices or a stable set of p vertices since x is joined to all the vertices of T in G^c. That is, G contains either a clique of p vertices or a stable set of q vertices. ■

We give below a table of some known Ramsey numbers (see [6], [4]).

Table 4.2: Some known Ramsey numbers $r(p, q)$

q \ p	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2	1	2	3	4	5	6	7
3	1	3	6	9	14	18	23
4	1	4	9	18	25		
5	1	5	14	25			
6	1	6	18				
7	1	7	23				

It is proved that $43 \leq r(5, 5) \leq 49$. However, the *exact* value of $r(5, 5)$ is not known. The numbers $r(p, p)$ are called the *diagonal Ramsey numbers*. For example, to prove that $r(5, 5) = k$ we have to prove that every simple graph on k vertices contains either a *stable set* of k vertices or a *clique* of k vertices *and* construct or prove the existence of a simple graph on $k-1$ vertices which contains neither a stable set of k vertices nor a clique of k vertices.

We now prove the following simple recurrence inequality involving Ramsey numbers:

Example 4.9.3 (Trivial Ramsey numbers):
Show that $r(p,1) = r(1,q) = 1$ and $r(2,q) = q \qquad r(p,2) = p$.

Solution: Consider the trivial graph G with exactly one vertex and no edges. Clearly, G itself is a clique of one vertex and a stable set of one vertex. The empty graph has neither a clique of one vertex nor a stable set of one vertex. By the definition of Ramsey numbers, $r(p,1) = r(1,q) = 1$.

We shall now prove that $r(2,q) = q$. Consider any simple graph on q vertices. If G has at least one edge $e = xy$ then $\{\, x, y \,\}$ is a clique of two vertices. If G does not contain any edge then the graph itself forms a stable set of q vertices. On the other hand, the graph consisting of $q-1$ vertices and no edges contains neither a clique of two vertices nor a stable set of q vertices. Therefore, $r(2,q) = q$. Similarly, $r(p,2) = p$.

Theorem 4.9.2 (Erdös-Szekeres):
The Ramsey numbers satisfy the recurrence relation

$$r(p,q) \leq r(p,q-1) + r(p-1,q) \text{ for all } p \geq 2 \text{ and } q \geq 2$$

Moreover, if both the numbers $r(p,q-1)$ and $r(p-1,q)$ are even integers, then the above inequality is strict.

Proof. By the *definition* of the Ramsey numbers, we have to show that every simple graph on $r(p,q-1)+r(p-1,q)$ vertices contains either a clique of p vertices or a stable set of q vertices.

Let G be a simple graph on $n = r(p,q-1)+r(p-1,q)$ vertices and take any vertex x of G. By the definition of the complement G^c of the graph G we have the equality $d_G(x)+d_{G^c}(x) = n-1$ for every vertex x of G, that is, in words, the number of vertices adjacent to x plus the number of vertices nonadjacent to x is always $n-1$. Hence the vertex x is either *nonadjacent* to a set S of $\geq r(p,q-1)$ vertices or else x is *adjacent* to a set K of $\geq r(p-1.q)$ vertices in G.

Because of the above observation, two cases can arise:

Case 1. x is nonadjacent to a set S of vertices with $|S| \geq r(p, q-1)$.

By the *minimality* of the integer in the definition of $r(p, q-1)$, we have: The induced subgraph $< S >$ of G contains either a clique of p vertices or a stable set of $q-1$ vertices and therefore the induced subgraph $< S \cup \{ x \} >$ of G (and hence also G) contains either a clique of p vertices or a stable set of q vertices.

We now consider the *negation* of the case 1. Case 2. x is adjacent to a set K of vertices with $|K| \geq r(p-1, q)$. By the *minimality* of the integer in the definition of $r(p-1, q)$, we have: The induced subgraph $< K >$ of G contains either a clique of $p-1$ vertices or a stable set of q vertices and therefore the induced subgraph $< K \cup \{ x \} >$ of G (and hence also G) contains either a clique of p vertices or a stable set of q vertices.

Assume now that both $r(p-1, q)$ and $r(p, q-1)$ are both *even* integers. We have to show that every graph G on $n = r(p, q-1) + r(p-1, q) - 1$ contains either a clique of p vertices or a stable set of q vertices.

Let G be a graph on n vertices. Since n is an odd integer, by Corollary 4.1.1.2, G must have a vertex x of even degree. In particular, $d(x) \neq r(p-1, q) - 1$, that is, x can't be adjacent to *exactly* $r(p-1, q) - 1$ vertices. As above, the following two cases arise:

Case 1. x is nonadjacent to a set S of vertices with $|S| \geq r(p, q-1)$.

Case 2. x is adjacent to a set K of vertices with $|K| \geq r(p, q-1) - 1$. But the equality is excluded. Hence the case 2 becomes: x is nonadjacent to a set S of vertices with $|S| \geq r(p, q-1)$. By repeating the same argument, G contains either a clique of p vertices or a stable set of q vertices. Therefore

$$r(p, q) \leq r(p-1, q) + r(p, q-1) - 1.$$

■

Definition 4.9.2 ((p, q)-Ramsey graph):
A (p, q)-Ramsey graph is a simple graph G on $r(p, q) - 1$ vertices. By the definition of the number $r(p, q)$, the graph G contains neither a clique of p vertices nor a stable set of q vertices. For example, the elementary cycle of length five is a $(3, 3)$-Ramsey graph, because it contains neither a triangle nor a stable set of three vertices and $r(3, 3) = 6$ (see Example 4.9.1).

We have already proved that $r(3, 3) = 6$ (see Example 4.9.1). As an application of the above recurrence inequality, we shall show in the following example that $r(3, 4) = 9$.

Example 4.9.4 (Ramsey number $r(3, 4)$):
Show that $r(3, 4) = 9$.

Solution: We shall first construct a simple graph on 8 vertices containing no clique of three vertices and no stable set of four vertices (see the graph of Figure 4.34 which has 8 vertices).

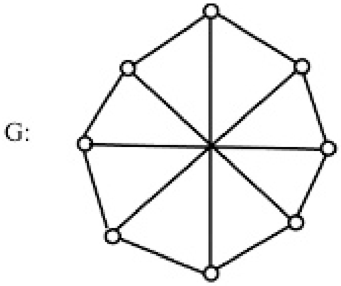


Figure 4.34: (3,4)-Ramsey graph

The reader can verify that the graph of Figure 4.34 contains no clique of three vertices and no stable set of four vertices. Hence by the definition of the Ramsey number $r(3, 4)$ we have the inequality

$$r(3, 4) \geq 9 \tag{4.3}$$

We shall now prove that $r(3, 4) \leq 9$. By the inequality of Theorem

4.9.2 we have

$$r(3,4) \leq r(2,4) + r(3,3).$$

But $r(2,4) = 4$ because every graph on four vertices contains either a clique of two vertices, that is an edge, or else it contains *no* edges and hence the whole graph itself is a stable set of four vertices. Further, the graph consisting of three vertices and *no* edges contains neither a clique of two vertices nor a stable set of four vertices.

Since both $r(3,3) = 6$ (see Example 4.9.1) and $r(2,4)$ are even, we have strict inequality in Theorem 4.9.2. Hence we have

$$r(3,4) \leq r(2,4) + r(3,4) - 1 \leq 4 + 6 - 1 \leq 9 \tag{4.4}$$

By combining the two inequalities 4.3 and 4.4, we have $r(3,4) = 9$.

The following corollary gives an upper bound on the Ramsey number $r(p,q)$ in terms of binomial coefficients.

Corollary 4.9.2.1:
The Ramsey number $r(p,q)$ satisfies the inequality

$$r(p,q) \leq \binom{p+q-2}{p-1}.$$

Proof. The proof proceeds by induction on the sum $p+q$.

Induction Basis: We verify the inequality for $p+q \leq 5$.

We have $r(p,1) = 1$ for all $p \geq 1$ (by Example 4.9.3), and

$r(p,1) = 1 \leq \binom{p+1-2}{p-1} = 1$. Moreover, $r(p,2) = p$ (by Example 4.9.3) and $r(p,2) = p \leq \binom{p+2-2}{p-1} = p$. In particular, the inequality is true for all $p \leq 4$. Hence by the symmetry of the Ramsey numbers, the induction basis is true for $p+q \leq 5$.

Induction Hypothesis: Suppose the inequality for all p and q with $5 \leq p+q < a+b$.

We shall prove the inequality for $r(a,b)$.

By Theorem, 4.9.2, we have

$$r(a,b) \quad \leq \quad r(a-1,b) + r(a,b-1)$$

$$\begin{aligned}
&\le \binom{a-1+b-2}{b-1} + \binom{a+b-1-2}{b-2} \\
&\quad \text{by the induction hypothesis} \\
&= \binom{a+b-3}{b-1} + \binom{a+b-3}{b-2} \\
&= \binom{a+b-2}{b-1} \\
&\quad \text{since } \tbinom{n}{k-1} + \tbinom{n}{k} = \tbinom{n+1}{k} \text{ “Pascal identity”}
\end{aligned}$$

■

Probabilistic Method

We shall now prove an inequality concerning the diagonal Ramsey numbers $r(k, k)$ using a powerful method in combinatorics called the Probabilistic method. This method is used to prove the existence of a mathematical object (in our case, a graph) with certain properties. Before presenting the inequality due to Erdös let us briefly review some basics of probability theory.

First of all, we recall some basics of probability theory (see Chapter 6 for more details).

Definition 4.9.3 (Sample space, events):
The set of all possible outcomes of an experiment is called a *sample space.* An *event* is a set of outcomes. Put differently, an event is a subset of the sample space.

The following example clarifies the notions of sample space and events.

Example 4.9.5 (Sample space, events):
Consider the experiment of tossing a fair coin twice. Let us note the head by H and the tail by T. Then the sample space of this experiment is the set S consisting of following four elements:

$$S = \{ HH, HT, TH, TT \}.$$

Note that the set S is the Cartesian product: $S = \{H, T\} \times \{H, T\}$. Let A be the event of getting at least one head. The subset A of S is $A = \{HH, HT, TH\}$. The event B of obtaining only tails is the singleton set $B = \{TT\}$.

Definition 4.9.4 (Probability of an event):
Let S be a finite sample space and let E be an event. Then the probability of the event E is written as $\Pr(E)$ and is defined as the ratio

$$\Pr(E) = \frac{|E|}{|S|}.$$

In other words,

$$\Pr(E) = \frac{\text{Number of outcomes favorable to } E}{\text{Total number of possible outcomes}}$$

In the above Example 4.9.5, $\Pr(A) = \frac{3}{4}$ and $\Pr(B) = \frac{1}{4}$.

What is a probabilistic method?

Let us describe in an informal and intuitive manner the probabilistic method: Consider a finite set $\mathcal{G}$ of graphs. If we can prove that an appropriately defined random graph would have the property "P" with *positive* probability, then we can conclude that there must exist a graph G in the set $\mathcal{G}$ with the property "P." This is because the probability is *zero* only in the case of the event $E = \emptyset$.

Let us see a simple probabilistic proof of Theorem 4.5.3 (already proved).

Theorem 4.9.3:
A loopless multigraph G of m edges contains a spanning bipartite subgraph of at least $m/2$ edges.

Proof. Let $H = (X_1, X_2)$ be any randomly chosen spanning bipartite subgraph of G. Take any edge $e = xy$ of G. Then there are *two* possibilities: either both ends of e are in the same partition of H or in different partitions of H. Hence the probability of having one end of e in X_1 and another in X_2 is $1/2$ and this is true of each edge of G. Hence the expected ("average") number of edges of the graph H is (see Chapter 6 for the definition of expectation) $\sum_{i=1}^{m} 1/2 = m/2$. Therefore, there must be at least one bipartite graph having at least $m/2$ edges. Analogy: If the average mark of a class is 55, there must be at least one student having at least 55 marks. ■

Analogy: If the average salary of a professor in a university is 4,000 euros, there must be at least one professor whose salary is at least (≥) 4,000 euros.

Theorem 4.9.4 (Erdös):
The diagonal Ramsey 4,000 numbers $r(k, k)$ satisfy the inequality

$$r(k, k) \geq 2^{k/2} \text{ for all } k \geq 2.$$

Proof. Since $r(2, 2) = 2 \geq 2^{2/2}$, and $r(3, 3) = 6 \geq 2^{3/2}$, we can assume that $k \geq 4$.

Consider the set $\mathcal{G}_n$ of all possible *non-identical* simple graphs on the vertex set $[n] = \{ x_1, x_2, \ldots, x_n \}$. By Theorem 4.4.4, the cardinality of the set $\mathcal{G}_n$ is given by

$$|\mathcal{G}_n| = 2^{\binom{n}{2}}.$$

Now we are interested in finding an upper bound for the number of graphs in the set $\mathcal{G}_n$ possessing a *clique* of k vertices.

Let us first fix a set $[k] = \{ 1, 2, \ldots, k \}$. We shall first find the number of graphs in $\mathcal{G}_n$ possessing the set $[k]$ as a clique. Since there are $\binom{n}{2}$ possible edges on the vertex set $\{ x_1, x_2, \ldots, x_n \}$ and $\binom{k}{2}$ edges of the induced clique $[k]$ are already fixed, we have the freedom to choose *any subset* of the remaining edge set of cardinality $\binom{n}{2} - \binom{k}{2}$ to have a graph with the clique set $[k]$. Since a set of cardinality p contains exactly 2^p subsets, the number of graphs in $\mathcal{G}$ with the clique $[k]$ is

$$2^{\binom{n}{2}-\binom{k}{2}}.$$

Let us now *vary* the k-subset $[k]$. We know that the number of k-subsets of the n-set $[n] = \{ x_1, x_2, \ldots, x_n \}$ is the binomial coefficient $\binom{n}{k}$. Hence the subset $\mathcal{G}_n^{<k>}$ of graphs in $\mathcal{G}_n$ possessing a clique of k-vertices satisfies the inequality

$$|\mathcal{G}_n^{<k>}| \leq \binom{n}{k} \times 2^{\binom{n}{2}-\binom{k}{2}}.$$

Notice the sign of inequality above, because while finding the set $\mathcal{G}_n^{<k>}$ the same graph may be generated many times (for instance, the complete graph K_n.) Therefore

$$\begin{array}{rcl} & & \text{Pr(A graph } G \text{ in } \mathcal{G}_n \text{ has a } k\text{-clique)} \\ & \leq & \dfrac{|\mathcal{G}_n^{<k>}|}{|\mathcal{G}_n|} \\ & \leq & \dbinom{n}{k} 2^{-\binom{k}{2}} \\ & \leq & \dfrac{n(n-1)\cdots(n-k+1)}{k!} \times 2^{-\binom{k}{2}} \\ & < & \dfrac{n^k 2^{-\binom{k}{2}}}{k!} \text{ (since } n-i < n \text{ for } i \geq 1.) \\ \text{(now for } n < 2^{k/2}) & < & \dfrac{2^{k^2/2} 2^{-\binom{k}{2}}}{k!} \\ & = & \dfrac{2^{k/2}}{k!} \\ & < & \dfrac{1}{2} \text{ (since } k! > 2^{k-1} \geq 2^{(k+2)/2} \end{array}$$

Hence, strictly less than fifty percent of the graphs in $\mathcal{G}_n$ contain a *clique* of k vertices.

Similar arguments as above can be applied to prove that strictly less than fifty percent of the graphs in $\mathcal{G}_n$ contain a *stable set* of k-vertices. Therefore, there must be a graph (with n vertices) in $\mathcal{G}_n$ which has neither a clique of k vertices nor has a stable set of k vertices. By the definition of $r(k,k)$ and our assumption on the number of vertices n, $n < 2^{k/2}$, we obtain $r(k,k) \geq n+1 \geq 2^{k/2}$

by choosing n as follows:

$$n = \begin{cases} 2^{k/2} - 1 & \text{if } 2^{k/2} \text{ is an integer,} \\ \lfloor 2^{k/2} \rfloor & \text{otherwise.} \end{cases}$$

■

Corollary 4.9.4.1:
The Ramsey numbers $r(p,q)$ satisfy the inequality

$$r(p,q) \geq 2^{\frac{\min(p,q)}{2}}.$$

Proof. Set $k = \min(p,q)$. Consider *any* simple graph G on $r(p,q)$ vertices. By the definition of $r(p,q)$, G contains either a clique of p vertices or a stable set of q vertices. But any clique of p vertices contains also a clique of k vertices and any stable set of q vertices contains also a stable set of k vertices. This is because $k \leq p$ and $k \leq q$. Hence the graph G contains a clique of k vertices or a stable set of k vertices. By the minimality of the number $r(k,k)$, we have

$$r(p,q) \geq r(k,k) \geq 2^{k/2} \text{ by Theorem 4.9.4.}$$

■

A generalization of Ramsey numbers

The Ramsey number $r(p,q)$ may be viewed in terms of a two partition of the edge set of the complete graph K_n as described below.

Consider the complete graph K_n and any two partition (E_1, E_2) of the edge set E of K_n, that is, $E = E_1 \cup E_2$ and $E_1 \cap E_2 = \emptyset$. Notice that E_i may be empty for some i. The edges of E_1 are said to be colored with color 1 and the edges of E_2 are said to be colored with color 2.

Definition 4.9.5:
The Ramsey number $r(p,q)$ is the *smallest* integer n such that any two partition (E_1, E_2) of the edge set of the complete graph

K_n contains either a complete subgraph on p vertices all of whose edges are colored with color 1 or contains a complete subgraph on q vertices all of whose edges are colored with color 2.

This is because, if $G = (X, E)$ is a simple graph on n vertices then $K_n = (X, E \cup E^c)$.

The above point of view enables us to generalize the Ramsey numbers as a function of k arguments $k \geq 2$ instead of two arguments.

Consider the complete graph K_n and any k partition $(E_1, E_2, \ldots, E_k)$ of the edge set E of K_n, that is, $E = E_1 \cup E_2 \cdots \cup E_k$ and $E_i \cap E_j = \emptyset$ for all $1 \leq i < j \leq n$. Notice that E_i may be empty for some i. The edges of E_i are said to be colored with color i for $1 \leq i \leq n$.

Definition 4.9.6 (Generalization of Ramsey numbers): The Ramsey number $r(p_1, p_2, \ldots, p_k)$ is the *minimum* integer n such that any k partition $(E_1, E_2, \ldots, E_k)$ of the edge set E of the complete graph K_n contains, for some i, a complete subgraph on p_i vertices, all of whose edges are colored with the color i.

Instant insanity puzzle

We are given four cubes of the same dimension. Denote the four cubes by *cube 1, cube 2, cube 3 and cube 4.* The six faces of each of the cubes are variously colored with the colors Yellow, Green, Red and Blue.

The puzzle poses the following question: Is it possible to stack the four cubes, one on top of another, in such a way that *all the four colors* appear on each of the four sides, east side, west side, north side, south side, of the resulting 4 × 1 stack?

See the illustration below: The cubes are unfolded in Figure 4.35.

One possible solution to the problem is to "try all possibilities" (brute-force method) of the stacking of the cubes until we find the right one or conclude that no such stacking is possible. The problem with this case-by-case exhaustive analysis is that the maxi-

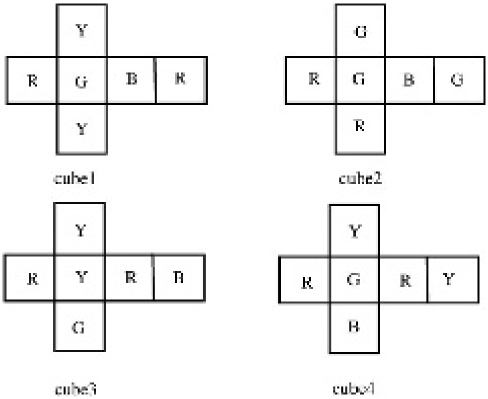


Figure 4.35: Unfolded cubes

mum number of possible trials will be 3 × 24 × 24 × 24 = 41472.

To find a solution to the puzzle in a reasonable amount of time, we construct a multigraph G as follows:

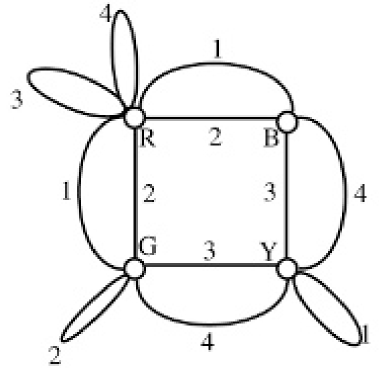
The vertices of the multigraph are the four different colors: Yellow(Y), Green(G), Red(R) and Blue(B). Two vertices are joined by an edge labeled by the integer i where $1 \leq i \leq 4$, if the corresponding colors form the opposite faces of the cube i.

Now let us proceed backwards, that is, assume that we are able to stack up the cubes, one on top of another, in such a way that all the four colors appear on the east side, west side, north side and south side. Construct the two subgraphs G_1 and G_2 of the graph G *induced* by the east-west faces and north-south faces respectively (see Figure 4.36). These two subgraphs possess the following properties:

1. G_1 and G_2 are *spanning* subgraphs of G; because all four colors are represented on each face of the stack.

2. Each of the two subgraphs has exactly four edges and these edges are labeled differently with $1, 2, 3, 4$; because by joining four opposite faces of the stack of four cubes we will have four edges labeled differently.

3. The subgraphs G_1 and G_2 are edge disjoint because an east-west edge is different from a north-south edge.



The Graph G

Figure 4.36: Graphs induced by east-west faces and north-south faces

4. G_1 and G_2 are regular subgraphs of degree 2; because each color appears exactly *twice* on east plus west sides of the stack and exactly *twice* on north plus south sides of the stack.

These four *necessary* conditions to have a solution to the puzzle can be easily seen to be also *sufficient*! That is, we may stack the cubes one on top of another if and only if the graph G admits two subgraphs G_1 and G_2 satisfying the above four stated properties.

Let us come back to our illustration: The graph G admits two subgraphs G_1 and G_2 satisfying all the four properties. (See the graphs G_1, G_2 and a right stacking of the cubes of Figure 4.37.)

4.10 Graph Algebra

In this section, we study some basic operations on graphs. These include the *union of graphs*, *join of graphs*, *Cartesian product of graphs*, *contraction of an edge of G* and *minor of a graph*. These operations are defined here only for *simple* graphs but these definitions can be naturally extended to multigraphs and digraphs as

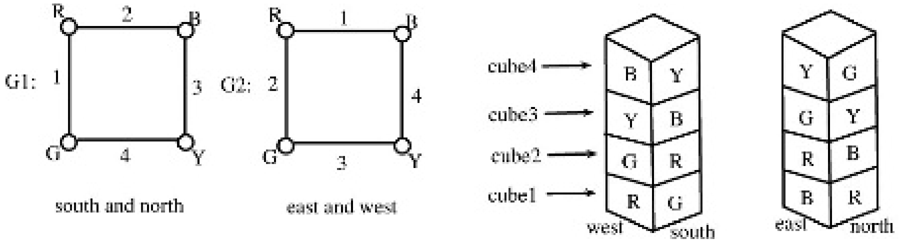


Figure 4.37: Graphs and desired stacking of cubes

well.

Consider two simple graphs $G_1 = (X_1, E_1)$ and $G_2 = (X_2, E_2)$ with disjoint vertex sets and edge sets.

Then *the union of* G_1 *and* G_2 denoted as usual by $G_1 \cup G_2$ is defined naturally as follows: The vertex set of $G_1 \cup G_2$ is the union $X_1 \cup X_2$ and the edge set is the union $E_1 \cup E_2$.

The join of G_1 *and* G_2 is denoted by $G_1 + G_2$ and is composed of the union $G_1 \cup G_2$ and all *possible* edges joining the vertex sets X_1 and X_2.

The Cartesian product of G_1 *and* G_2 denoted as expected by $G_1 \times G_2$ is defined intuitively as follows:

$G_1 \times G_2$ is obtained by substituting for each vertex of G_1 a copy of the graph G_2 (this operation of substitution is sometimes called a *blowing-up operation or expansion or splitting* as opposed to the operation of *contraction*) and replacing each edge of G_1 by the edges between corresponding vertices of the appropriate copies.

In a formal manner, the vertex set of the Cartesian product $G_1 \times G_2$ is the set of all possible ordered pairs (x_1, x_2) with x_1 *in* X_1 and x_2 *in* $X_.$ The edge set of $G_1 \times G_2$ is obtained by joining the vertices (x_1, x_2) and (x_1, x_2') whenever x_2x_2' is an edge of the graph G_2 together with the edges drawn by joining the vertices (x_1, x_2) and (x_1', x_2) whenever x_1x_1' is an edge of the graph G_1. Note that a minimal condition to have an edge in the product is that either the first components of the vertex must be equal or else the second components are equal.

We illustrate these operations by the following examples:

Example 4.10.1 (Graph union and join):
The graph of Figure 4.38 illustrates the operation of union and join of two graphs.

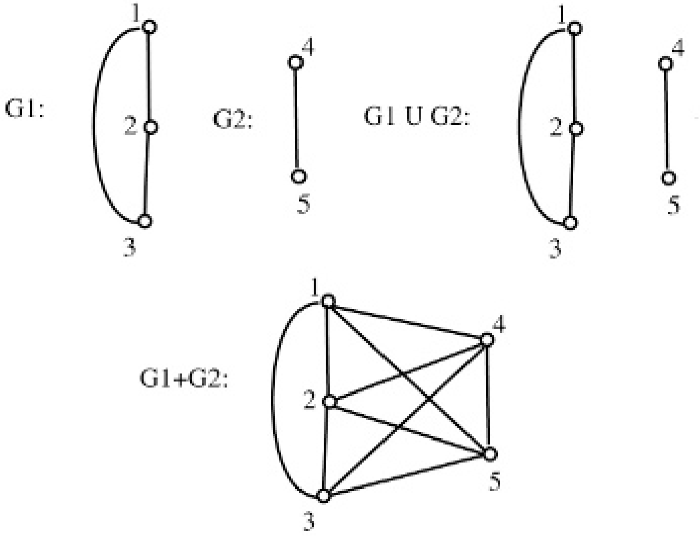


Figure 4.38: Graph union and join

Example 4.10.2 (Graph product):
The graph of Figure 4.39 illustrates the operation of the product of two graphs.

For other graph products like composition of graphs, the reader can see the books [1],[5].

Operation of contraction of an edge: Consider a simple graph G and an edge $e = xy$ in G. The simple graph obtained by *identifying the ends x and y of the edge e* and is denoted by $G.e$ or $G[x = y]$. In a formal manner, the graph $G.[x = y]$ is obtained from the graph G by the following the two sequences of operations:

1. The removal of the vertices x and y.

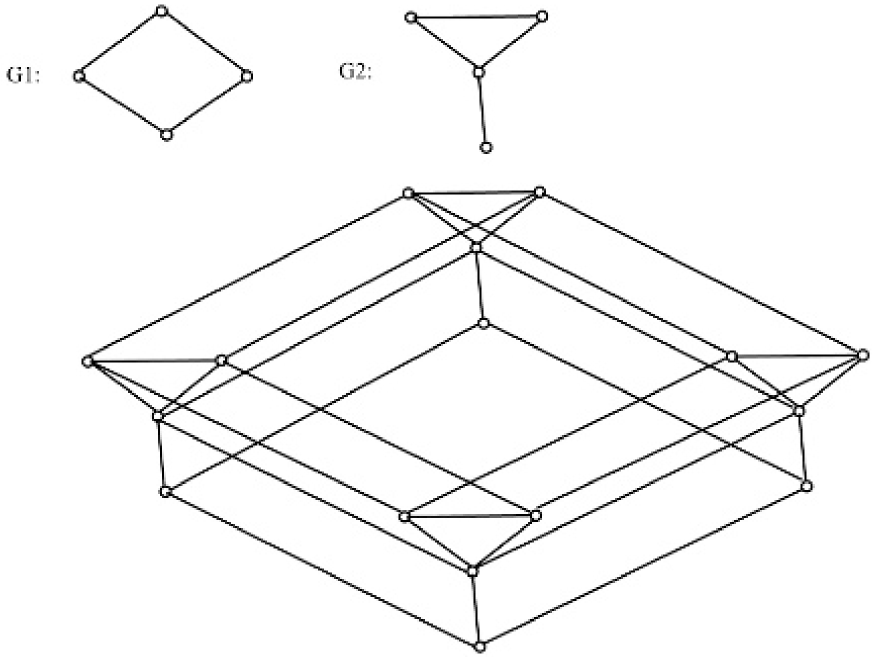


Figure 4.39: Graph product $G_1 \times G_2$

2. The addition of a *new* vertex denoted by z which is $[x = y]$ and joining the new vertex z to the vertices $v \neq x, y$ of G already joined either to x or to y or to both in the base graph G. (Replace parallel edges form z by a single edge.)

We can view the contraction $G.e$ where $e = xy$ as the graph obtained (up to isomorphism) by joining the vertex x in the vertex deleted subgraph graph $G - y$ to all the neighbors of the vertex y in the base graph G not already joined to x. By symmetry, $G.e$ is also the graph obtained by joining the vertex y in the vertex deleted graph $G - x$ to all the neighbors of the vertex x in the original graph G not already joined to y.

The following example clarifies the notion of contraction:

Example 4.10.3 (A graph and its contraction):
Let us refer to the graph of Figure 4.40 which illustrates the con-

traction of edge 14 of the graph G.

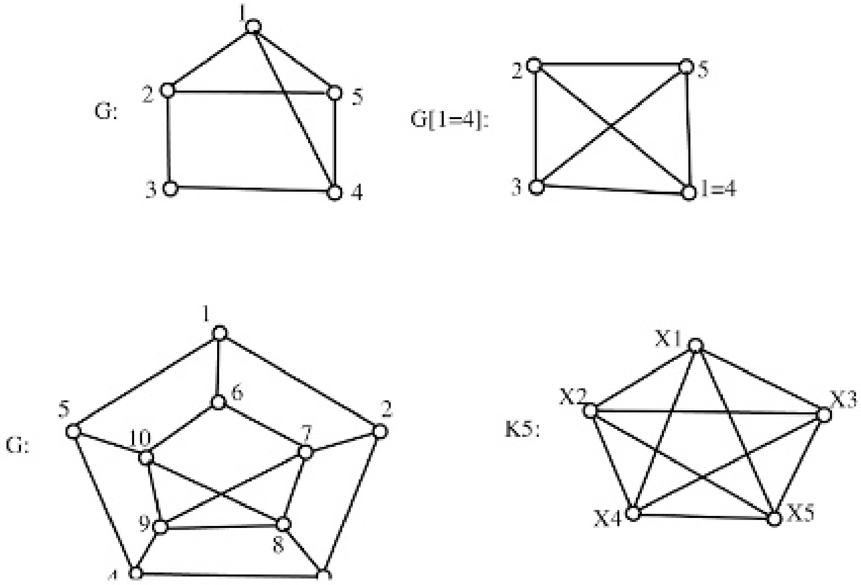


Figure 4.40: A graph and its contraction

A simple way of describing a sequence of edge contractions:

There is a *simple* way to describe a sequence of edge contractions in a graph: Suppose we start with a simple graph G. Let G_1 be the graph obtained by contracting an edge e_1 of G and let G_2 be the graph obtained by contracting an edge e_2 in G_1, and so on until we arrive finally at a graph H. Then such a graph H can be described in a simple manner as follows: Consider a partition $X_1 \cup X_2 \cup \cdots X_k$ of the vertex set X of G such that each induced subgraph $< X_i >$ is connected for all $1 \leq i \leq k$. Define the graph H as follows: The vertex set of H is $\{ X_1, X_2, \ldots, X_k \}$ and we join the vertices X_i and X_j in H if there is an edge joining a vertex of X_i with a vertex of X_j *in* the base graph G.

The following example illustrates the sequence of edge contractions:

Example 4.10.4 (A graph and its sequence of edge contraction): The complete graph K_5 is obtained from the graph G by a sequence of edge contractions. The partition of the vertex set of G is as follows: $X_1 = \{1, 2, 3, 4, 5\}$, $X_2 = \{10\}$, $X_3 = \{6, 7\}$, $X_4 = \{9\}$, $X_5 = \{8\}$ (see Figure 4.41).

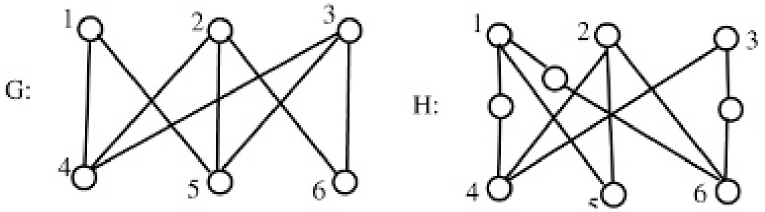


Figure 4.41: A graph and its sequence of edge contractions

Minor of a graph

Definition 4.10.1:
A minor of a simple graph G is any graph obtainable from G by a sequence of vertex deletions, edge deletions and edge contractions (in any order).

Example 4.10.5 (Minor of a graph):
As an example, consider again the Petersen graph. The complete graph on 5 vertices K_5 is a minor of the Petersen graph, because K_5 is obtained by contracting the five edges joining the pentagon (the elementary cycle $(1, 2, 3, 4, 5, 1)$ and the pentagram (the elementary cycle $(6, 8, 10, 7, 9, 6)$ (see the Petersen graph drawn in the text). Hence by the following theorem, the Petersen graph is *nonplanar.*

We now state the following *theorem due to Wagner* characterizing planar graphs (see the book by Parthasarathy [3]).

Theorem 4.10.1:
A graph G is planar if and only if G contains neither the complete graph K_5 nor the complete bipartite graph $K_{3,3}$ as a minor.

The nonexistence of a particular graph as a *minor* of a given graph influences profoundly the structure of the given graph. The graph minor theory has been developed by Robertson and Seymour (see [7] for an interesting history).

As an example we state the following theorem:

Theorem 4.10.2 (Graph minor theorem, Robertson and Seymour):
For any infinite set $\mathcal{S}$ of simple graphs, there exist two distinct graphs in the set $\mathcal{S}$, such that one of these graphs is a minor of the other.

4.11 Exercises

1. If G is any digraph, then prove the following three double inequalities:

$$\begin{array}{rcl} \delta^-(G) & \leq \frac{m}{n} & \leq \Delta^-(G) \\ \delta^+(G) & \leq \frac{m}{n} & \leq \Delta^+(G) \\ \delta^{(}G) & \leq \frac{2m}{n} & \leq \Delta^{(}G) \end{array}$$

   where $\delta^-(G), \delta^+(G), \Delta^-(G), \Delta^+(G), \delta(G), \Delta(G)$ are, respectively, the minimum in-degree, the minimum out-degree, the maximum in-degree, andthe maximum out-degree, the minimum degree, and the maximum degree of the graph G.

2. Show that in any digraph G without containing any directed circuit, we always have the equalities $\delta^-(G) = \delta^+(G) = 0$.

3. Consider any multigraph G. By assigning a direction to each edge of the graph, we obtain a digraph denoted by $\vec{G}$ which

is referred to as an *orientation* of the graph G. The graph G becomes the *underlying undirected graph of the digraph* $\vec{G}$. Show that any multigraph G has an orientation $\vec{G}$ such that $|d^+(x) - d^-(x)| \leq 1$.

4. Consider a 1-graph G without any loops with the property that $\max(\delta^-(G), \delta^+(G)) = p > 0$. Prove that G contains a directed cycle of length at least $p + 1$.

5. Show that if G is a simple graph with the minimum degree $\delta(G) \geq 2$, then G contains an elementary cycle.

6. Let G be a k-regular ($k > 0$) bipartite graph with bipartition (X_1, X_2). Then show that $|X_1| = |X_2|$.

7. Prove or disprove:

   (a) Every simple graph on p vertices is a subgraph of the complete graph $K_n(p \leq n)$.

   (b) Every simple graph on n vertices is a spanning subgraph of the complete graph K_n.

8. Construct the digraph $G = (X, U)$ where the vertex set $X = \{1, 2, 3, 4\}$ and the arc set is defined by the binary relation U on the set X as $U = \{(i, j) \mid i + j = 4 \text{ or } i + j = 7\}$. Find the number of connected components of the graph so constructed.

9. Construct the directed graph $G = (X, U)$ with the vertex set $X = \{1, 2, 3, 4\}$ and the arc set $U = \{(i, j) \mid i < j\}$. Does the graph contain a directed circuit? Is it a complete graph? Is G a *transitive graph*? (A digraph G is transitive if (x, y) and $(y, z) \in U$ imply that (x, z) also $\in U$.)

10. Prove or disprove:

    There exists a directed graph on four vertices $1, 2, 3, 4$ with the degrees verifying the properties $d^+(1) = 1, d^+(2) = 2$, $d^+(3) = 1, d^+(4) = 1$ and $d^-(1) = 2, d^-(2) = 1, d^-(3) = 1, d^-(4) = 2$. If the answer is "yes," then construct one such graph.

11. Prove or disprove:

    There exists a directed graph on four vertices $1, 2, 3, 4$ with the degrees satisfying the properties $d^+(1) = 1, d^+(2) = 2, , d^+(3) = 1, d^+(4) = 1$ and $d^-(1) = 2, d^-(2) = 1, d^-(3) = 1, d^-(4) = 1$. If the answer is "yes," then construct such a graph.

12. A sequence of n $(n \geq 1)$ non-negative integers $(d_1, d_2, \ldots, d_n)$ is said to be *multigraphic* if there is a multigraph on n vertices $1, 2, \ldots, n$ with $d(i) = d_i$ for all i with $1 \leq i \leq n$.

    Prove that a sequence of non-negative integers $(d_1, d_2, \ldots, d_n)$ is multigraphic if and only if $\sum_{i=1}^{n} d_i$ is an even integer.

    Give an *algorithm* to construct a multigraph on n vertices $1, 2, \ldots, n$ with $d(i) = d_i$ for all i with $1 \leq i \leq n$ whenever the sequence $(d_1, d_2, \ldots, d_n)$ is multigraphic.

13. Draw all the eleven non-isomorphic simple graphs on four vertices.

14. How many non-identical simple graphs on four vertices exist?

15. Construct two non-isomorphic simple cubic graphs on six vertices.

16. Let us refer to two different drawings of the Petersen graph in the text. Find an isomorphism between these different drawings of the Petersen graph.

17. Draw the line graphs of star $K_{1,3}$ and the complete graph K_3.

18. What is the line graph of an elementary cycle C_n on n vertices?

19. Draw the total graph of the elementary cycle C_4. Describe the total graph of the elementary cycle C_n.

20. True or false:

(a) The total graph $T(G)$ of G contains the graph G as an induced subgraph.

(b) The total graph $T(G)$ of G contains the line graph $L(G)$ as an induced subgraph.

21. Let G be a simple graph with n vertices $1, 2, \ldots, n$ and m edges and let the degree of the vertex i be d_i for $1 \leq i \leq n$. Express the number of vertices and edges of the total graph $T(G)$ of G in terms of n, m, d_i.

22. Find the chromatic index of the complete graphs K_4 and K_5. Guess a formula for the chromatic index of the complete graph K_n with n odd and even.

23. An elementary cycle of length n in a simple graph G on n vertices is called a *Hamilton cycle*. A Hamiltonian cycle doesn't always exist. A graph containing a Hamiltonian cycle is called a *Hamiltonian graph.*

    Show that the Petersen graph is *not* a Hamiltonian graph but becomes a Hamiltonian graph if we delete any vertex. Such a graph is called a *hypo-Hamiltonian graph.*

24. Construct a cubic simple graph on $2p$ $(p \geq 3)$ vertices without containing the elementary cycle C_3.

25. Construct two geodetic graphs of diameter 2 other than the Petersen graph.

26. The *kth power* $G^k (k \geq 1)$ of a simple graph $G = (X, E)$ is defined in terms of the distance between two vertices as follows:

    G^k has the same vertex set X as the original graph G and two distinct vertices x and y are joined in the kth power G^k if and only if the distance between them *in the graph* G is less than or equal to the integer k.

    (a) Describe the square of the Petersen graph.

(b) Describe the dth power G^d of a simple graph with diameter d.

(c) If the distance between two vertices x and y, $d(x, y) = p$ in the simple graph G, find $d(x, y)$ in the kth power G^k.

27. Consider a multigraph $G = (X, E)$ and consider an edge xy of G. The edge xy is said to be *subdivided* when it is replaced by an elementary path $(x, x_1, x_2, \ldots, x_s, y)$ where $x_1, x_2, \ldots, x_s$ are s *newly* introduced vertices *not* in the vertex set X. A graph H is a *subdivision graph* of G whenever H can be obtained from G by a sequence of subdivisions of a subset of the edges (in any order) of G.

    H is obtained by subdividing some of the edges of the graph $G = K_{3,3}$. The newly introduced vertices during the operation of subdivision are not labeled in the diagram (see Figure 4.42).

    (In fact, a famous *theorem of Kuratowski* characterizes nonplanar graphs: A multigraph G is nonplanar if and only if there is a subgraph isomorphic to a subdivision of either the complete bigraph $K_{3,3}$ or the complete graph K_5. See page 152 of the book by Parthasarathy [3] or Balakrishnan and Ranganathan [5].)

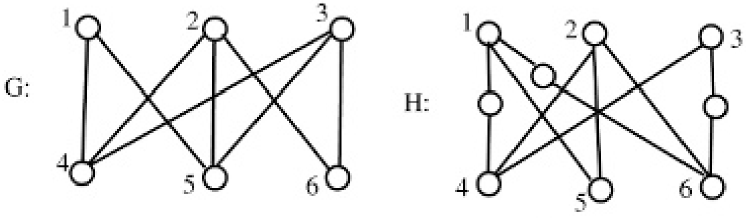


Figure 4.42: $K_{3,3}$ and its subdivision

28. Find a subgraph of the Petersen graph which is isomorphic to a subdivision of the complete bigraph $K_{3,3}$.

(a) Does there exist a self-complementary graph on n vertices where $n \leq 3$?

(b) Construct a self-complementary graph on 4 and 5 vertices.

(c) Prove that the number of vertices n of a self-complementary graph has the form $4p$ or $4p+1$ for some integer p.

(d) Prove that S is a stable set of G if and only if S is a clique of the complement G^c.

(e) Show that G and its complement G^c both cannot be disconnected simultaneously.

(f) Show that a self-complementary graph on $n > 1$ vertices has diameter either 2 or 3.

(g) Show that the sequence $(d_1, d_2, \ldots, d_n)$ is a graphic sequence if and only if the sequence $(n - d_1 - 1, n - d_2 - 1, \ldots, n - d_n - 1)$ is a graphic sequence.

(h) Show that Ramsey numbers are symmetric, that is, $r(p,q) = r(q,p)$ for positive integers p and q.

(i) What is the value of the Ramsey numbers $r(2,q)$?

(j) Prove or disprove the following equations involving two simple graphs G_1 and G_2.

i. $(G_1 + G_2)^c = G_1^c + G_2^c$

ii. $(G_1 \times G_2)^c = G_1^c \times G_2^c$.

(k) Show that the edge graph of the complete graph K_5 is isomorphic to the complement of the Petersen graph.

29. Let us refer to the definition of the *Turán graph* $T_{n,k}$ in the text.

a) Show that the number of edges in the Turán graph $T_{n,k}$ is exactly equal to

$$\binom{n-q}{2} + (k-1)\binom{q+1}{2} \text{ where } q = \lfloor \frac{n}{k} \rfloor,$$

the quotient of n divided by k.

b) If $G = K_{n_1,n_2,\ldots,n_k}$ is *any* complete $k-$partite graph on n vertices then show that $m(G) \leq m(T_{n,k})$ and that equality holds if and only if $G \cong T_{n,k}$.

30. If G is a simple graph on n vertices with its minimum degree $\delta(G) \geq (n-1)/2$ then prove that the graph G is connected.

31. Show that the Instant Insanity Puzzle has no solution if the four unfolded cubes are colored as shown in Figure 4.43.

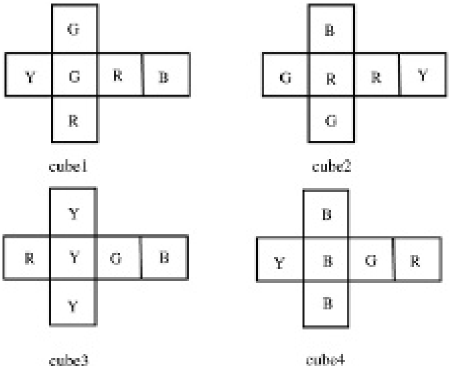


Figure 4.43: Four unfolded cubes

32. Find a solution to the four cubes problem if the four unfolded cubes are colored as shown in Figure 4.44.

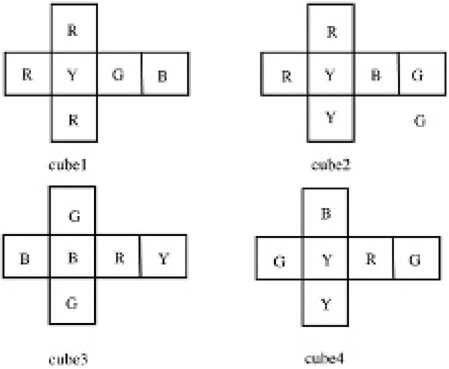


Figure 4.44: Four unfolded cubes

33. If G_1 and G_2 are two simple graphs with the number of vertices and edges n_1, m_1 and n_2, m_2 respectively, then find the number of vertices and edges of the Cartesian product $G_1 \times G_2$.

34. Draw the Cartesian product $K_2 \times C_5$ where K_2 is the complete simple graph on 2 vertices and C_5 is the elementary cycle of 5 vertices.

35. Can you write the Petersen graph as the product $K_2 \times G$ for some suitably chosen graph G. Justify your answer.

36. The graph *k-cube* $(k \geq 1)$ denoted by Q_k and is defined in a recursive manner as follows:

    $Q_1 = K_2$ (basis of the recursion) and $Q_k = K_2 \times Q_{k-1}$ for $k \geq 2$ (recursion).

    (a) Draw the graphs Q_2 and Q_3.

    (b) Find the number of vertices and edges of the k-cube Q_k in terms of k.

    (c) Prove that Q_k is a bipartite graph.

    (d) Find a bipartite graph which is not a subgraph of any Q_k. (Equivalently, the k-cube Q_k is defined as follows: The vertex set of the k-cube is the set of all possible binary k-tuples $b_1 b_2 \cdots b_k$ with each $b_i \in \{0, 1\}$ and two vertices of Q_k are adjacent if their corresponding binary k-tuples differ at exactly one co-ordinate.)

37. Prove or disprove: If G_1 and G_2 are two regular simple graphs, then $G_1 + G_2$ is also regular $G_1 \times G_2$ is still regular.

38. Prove or disprove: If G_1 and G_2 are two simple bigraphs, then $G_1 + G_2$ is also a bigraph $G_1 \times G_2$ is still a bigraph.

39. Prove that in a connected simple graph, any two longest elementary paths have a vertex in common.

40. Construct a k-regular simple graph on $k^2 + 1$ vertices and diameter 2 for $k = 2, 3$. (Such a graph exists only for $k = 2, 3, 7$ and possibly 57.) For $k = 2$, it is the elementary cycle of length 5. For $k = 3$, it is the Petersen graph. For $k = 7$, Hoffman and Singleton constructed such a graph (see [4]).

Bibliography

[1] C. Berge, The Theory of Graphs and Its Applications, Wiley, New York, 1958; Graphs and Hypergraphs, North Holland, 1973.

[2] F. Harary, Graph Theory, Addison-Wesley, Reading, Mass, 1969.

[3] K. R. Parthasarathy, Basic Graph Theory, Tata-McGraw Hill Publishing Company Limited, New Delhi (1994).

[4] J. A. Bondy and U. S. R. Murty, Graph Theory with Applications, MacMillion, 1976.

[5] R. Balakrishnan and K. Ranganathan, A Textbook of Graph Theory, 2nd edition,Springer, New York (2012).

[6] D. B. West, Introduction to Graph Theory, 2nd edition, Prentice Hall, New Jersey (2001).

[7] G. Chartrand and P. Zhang, Introduction to Graph Theory, Tata McGraw-Hill Publishing Company Limited, New Delhi (2006).

[8] D. E. Knuth, Art of Computer Programming, Fundamental Algorithms, Vol 1, Addison-Wesley, Reading, Mass (1968).

[9] A. V. Aho, J. E. Hopcroft and J. D. Ullman, Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

Chapter 5

Introduction to Algorithms and Data Structures

Algorithms + Data Structures = Programs.

N. Wirth

Mathematical models have been a crucial inspiration for all scientific activity even though they are only approximate idealizations of the real-world phenomena. Inside computers such models are more relevant than ever before, because computer programs create artificial worlds in which mathematical models often apply precisely.

Donald Knuth

This chapter begins with the concept of an algorithm. Then the complexity of algorithms is studied. After a brief introduction to computer architecture, an introduction to programming languages is given. Since data structures go in parallel with algorithms, fundamental data structures like lists, stacks and queues are studied. In the last section, examples of algorithms with different complexity functions encountered in practice are illustrated.

5.1 Algorithms

The word *algorithm* is at the very heart of computer science. The word algorithm signifies approximately the same meaning as the following words: method, procedure, function, technique, strategy, process, recipe, etc. (See [1] and [8].)

An algorithm can be described as a systematic and progressive method to solve a problem; that is, it is a step-by-step procedure to solve a problem.

Mathematically, it can be viewed as a function acting on a specified set of arguments/inputs to produce a specified set of results/outputs. It transforms the "raw" data into the "refined" data (just like a kitchen recipe transforms flour, sugar, milk, etc., into a cake).

The modern description of algorithm is a method of solving a problem adapted to the computer implementation.

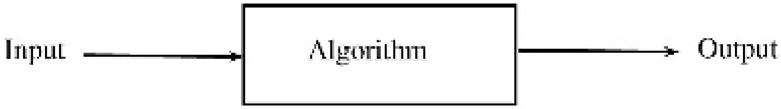


Figure 5.1: Algorithm as a black box

The concept of an algorithm is illustrated by Example 5.1.1, one of the oldest known algorithms due to Euclid (see also Chapter 3). Knuth calls Euclid's algorithm the granddad of all algorithms.

Example 5.1.1 (Euclid's greatest common divisor algorithm): Greatest Common Divisor of Two Integers (gcd):

Arguments or Inputs: Two positive integers m and n.

Result or output: The gcd(m, n).

Algorithm: If the two input integers m and n are equal, then the result is either m or n (basis).

Otherwise, that is, if $m \neq n$, then replace the greater integer by the difference, that is, substitute for the greater integer

Table 5.1: Execution of Euclid's gcd algorithm

m	n	is $m = n$?
25	35	*no*
25	$35 - 25 = 10$	*no*
$25 - 10 = 15$	10	*no*
$15 - 10 = 5$	10	*no*
5	$10 - 5 = 5$	*yes*

(the greater integer) − (the smaller integer), until we get equality (iteration).

When the equality is obtained, then one of the two equal integers is the desired result.

An algorithm must be seen to be believed! So, let us perform the algorithm of Example 5.1.1 on a sample input. This is called the hand simulation of algorithm. The following table illustrates a step-by-step execution of Euclid's algorithm for two integers $m = 25$ and $n = 35$.

> I hear, I forget;
> I see, I remember;
> I do, I understand.
>
> In Automathography, *Paul Halmos*

In the last line of the above table, the value of m is equal to 5 which is equal to the value of n. Hence the $\gcd(25, 35) = 5$.

An algorithm must possess the following five properties:

1. Finiteness or Termination: An algorithm must terminate after a finite number of operations. This need not be with a procedure or method.

2. Preciseness: Each instruction of an algorithm must be written in an unambiguous manner.

3. Input: An algorithm has zero or more inputs.

4. Output: An algorithm must produce at least one result/output.

5. Realizability or Hand Simulation: Hand simulation of the algorithm must be possible; that is, a human should be able to play the role of a computer. Stated differently, each instruction of an algorithm must be sufficiently elementary that a human being should be able to perform the algorithm in finite time and in an exact manner using paper and pencil.

The following is an example of a *method* which is a geometric construction that does *not* terminate:

Example 5.1.2 (gcd of two numbers: A Procedure which does not terminate):
The greatest common number of two incommensurable positive real numbers (that is, the ratio of the two numbers is not a rational number).

Input: Two line segments such that their ratio is not a rational number; that is, the given lengths are incommensurable.

Result: The greatest common measure of the two line segments.

Method: Essentially the procedure is the same as the algorithm of Example 5.1.1, interpreting "subtraction" as taking away a smaller line segment from the left end of the larger one, which is nothing but subtraction.

To see a solution to some "problems" with the help of *existing* algorithms, we may have to wait beyond our lifetime! Such algorithms are generally obtained by *exhaustive method*, by trying all possible possibilities. They are called *brute-force algorithms* . This is like playing all possible combinations/buying all tickets to win a national lottery! The following example clarifies the idea of the brute-force method.

Example 5.1.3 (Chess board problem):
Given a certain position of white and black pieces on a conventional 8×8 chess board, there is an algorithm to predict if the white or black will win the game! So, this algorithm is better than any human Grand Master. This algorithm examines all consequences of all the moves possible according to the rules of chess. This brute-force method takes a very long time, and we may not know the result in our lifetime!

We have described an algorithm as a systematic and progressive method to solve a "problem." Let us now describe the idea of a problem.

Problem

A problem is a question to be answered by "yes" or "no" (example: whether a given integer $n \geq 2$ is a prime number or not), or a structure to be found in response to several *input parameters* (example: in Example 5.1.1 the parameters are two positive integers m and n). These input parameters are usually described by fixing the set of values they can assume but their explicit values are not specified. The result of a problem is sometimes characterized by some properties.

Decision problem

A *decision problem* is one for which the answer is either "yes" or "no."

Undecidable Problems

A decision problem is an *undecidable problem* if there is *no* algorithm at all to solve the problem.

Example 5.1.4:
A halting problem is undecidable:

Input: An arbitrary computer program and an arbitrary input to that program.

Output: "Yes" if the program halts, that is, it does not contain an infinite loop. "No" if the program does not halt, that is, the program contains an infinite loop.

No algorithm can be given to solve this problem.

Another example is *Hilbert's tenth problem.* This problem concerns the solvability of polynomial equations in integers.

Theorem (Yuri Matijasevich): There is *no* algorithm to decide whether a given polynomial in several variables with integral coefficients has a solution in integers.

5.2 Complexity of Algorithms

Very often, we have two or more algorithms to solve the same problem. Among these algorithms to solve the same problem, how do we choose the best one? How do we judge the quality of an algorithm? An algorithm uses two main resources of a computer:

1. Time complexity: Time taken to execute the algorithm.
2. Space complexity: Memory space needed to perform the algorithm.

Time complexity

Intuitively, the time complexity of an algorithm or simply the complexity of an algorithm is the number of *elementary or basic statements/instructions* executed by the algorithm. Time complexity is expressed as a function of the *size* of the problem. Intuitively, the size of a problem is the number of times we have to type the keys in order to enter the input. For example, if we want to arrange or sort n given integers in non-decreasing order, the size of the problem is taken as n. The size of a graph problem may be n, the number of vertices or m, the number of edges or $\max(m, n)$.

In a formal manner, the size of a problem is the number of bits (binary digits) needed to encode the problem.

If an algorithm A processes an input of size n in $c.n^2$ units of time (one unit of time may be imagined as 10^{-6} second or one micro second), then we say that the complexity of the algorithm is of order n^2, and we write $T_A(n) = n^2$, that is, the time taken to find the solution of the problem of size n using algorithm A is of order n^2. Here c is a positive constant whose value depends on the machine, compiler, and data structures chosen to implement the algorithm.

Worst-Case complexity

This complexity is for "pessimists!" For a given size n of a problem, we may have several inputs meeting the size n. For many algorithms, the complexity does not depend on the size n but on a particular input of size n.

The *worst case complexity* is the *maximum* time needed, among all possible inputs of size n, to solve the problem. Stated differently, the worst case tells us the *slowest* an algorithm can run on an input of size n. In symbol, if $T_A(n)$ is the maximum time taken to solve a problem π of size n using algorithm A, then

$$T_A(n) = \max\{\, T(I_n) | I_n \in \pi_n \,\}$$

where π_n is the set of all inputs of size n and $T(I_n)$ is the time taken (number of basic statements executed) by the algorithm A on the input I_n.

We will be mostly interested in the worst-case time complexity since it has wide applicability. For example, if we are given *one* hour of computer time, the worst-case complexity of an algorithm determines how big a problem we can solve in an hour using a given algorithm.

To express the complexity, we need the following useful notation (see also Chapter 1).

“Big oh” notation

Intuitively, the equation $T(n) = O(f(n))$ (read: T of n is big oh of f of n) if the value function $T(n)$ is at most a constant times the value of the function $f(n)$, if n is sufficiently large; that is, the function $T(n)$ is *dominated* by the function $cg(n)$ (c is a positive constant) if n is sufficiently large. In other words, the quotient $\frac{T(n)}{f(n)}$ is bounded *above* by a constant if n is sufficiently large (assuming $f(n) \neq 0$).

Before studying examples illustrating big oh notations, let us see a remark on how to manipulate inequalities:

REMARK 5.2.1 (How to manipulate inequalities: transitivity, addition and positive scaling):
Given any two real numbers x, y, we have exactly one of the following relations:

1. $x = y$
2. $x < y$
3. $x > y$

Transitivity: If $x < y$ and $y < z$ then $x < z$. (x, y, z are reals.)

Addition: We may add two inequalities provided they “point in the same direction,” that is, if $x < y$ and if $z < t$ then $x+z < y+t$.

Positive scaling: If $x < y$ then $cx < cy$ if $c > 0$ and $cx > cy$ if $c < 0$. In short, multiplication by a negative number reverses the direction of the inequality and by a positive number preserves the inequality.

Reciprocal law: Let $x > 0$ and $y > 0$ be such that $x > y$. Then $1/x < 1/y$.

Let $x > y > 0$ and $s > t > 0$. Then $xs > yt$.

$|xy| = |x||y|$ and $|x/y| = |x|/|y|$ ($y \neq 0$.)

Triangle inequality: $|x + y| \leq |x| + |y|$. More generally,

$$|x_1 + x_2 + \cdots + x_n| \leq |x_1| + |x_2| + \cdots + |x_n|.$$

The following examples clarify the big oh notation:

Example 5.2.1 (Big oh):
We can write

$$1+2+\ldots+n=O(n^2).$$

because we know that

$$1+2+\ldots+n=n(n+1)/2.$$

So, it is enough if we prove that the value of $n(n+1)/2$ is at most a constant times the value of n^2, if n is sufficiently large.

Now $n(n+1)/2=(n^2+n)/2=n^2/2+n/2 \leq n^2+n^2=2\dot{n}^2$ if $n \geq 1$. Here the constant in question is 2 and we are not interested in finding the least constant satisfying the inequality. The lower bound for n for which the inequality holds is 1.

Example 5.2.2 (Big oh):
Prove that

$$1^2+2^2+\ldots+n^2=O(n^3).$$

Proof: We know that

$$1^2+2^2+\ldots+n^2=n(n+1)(2n+1)/6.$$

We have to show that $n(n+1)(2n+2)$ is at most a constant times n^3 if n is sufficiently large.

Now $n(n+1)(2n+1)/6=(2n^3+3n^2+n)/6=2n^3/6+3n^2/6+n/6 \leq n^3+n^3+n^3=3n^3$ for all $n \geq 1$. Here the multiplicative constant is 3 and lower bound for n for which the inequality is valid is 1.

We now state the formal general definition of the "Big Oh" Notation. The advantage of this notation is that it enables us to replace certain *inequalities* by *equations*.

Definition 5.2.1:
Consider two real-valued functions f and g defined on the set of positive integers. We write $f(n)=O(g(n)$ if there exist two constants $c>0$ and a positive integer n_0 such that the inequality $|f(n)| \leq c|g(n)|$ holds for $n \geq n_0$.

Since $n^2 = O(n^2)$ (because $n^2 \leq n^2$), $n \log n = O(n^2)$ (because $\log n \leq n$), $n = O(n^2)$, we may regard $O(n^2)$ as the set of all functions dominated by a constant times n^2. More generally, $O(f(n))$ is the set of all functions dominated by a constant times $f(n)$.

Average complexity or expected complexity

This complexity is for "probabilists!" The *average complexity* of an algorithm is the average time taken, over all inputs of size n. The average time complexity is more difficult to calculate than the worst-case complexity. To find the average complexity of an algorithm, we have to make some assumptions on the distribution of inputs and the realistic assumptions are difficult to tract mathematically.

If π_n is the set of all possible inputs of size n, and if each input is *equally likely* to occur, then the expected complexity is defined as

$$T_{av}(n) = \frac{1}{|\pi_n|} \sum_{I_n \in \pi_n} T(I_n)$$

where $T(I_n)$ is the time (or the number of elementary operations) taken to execute the algorithm on the input I_n.

If the assumption of equally likely occurrence on the inputs is dropped, then

$$T_{av}(n) = \sum_{I_n \in \pi_n} \Pr(I_n) \times T(I_n)$$

where $\Pr(I_n)$ is the probability that the input I_n occurs. As before, π_n is the set of all possible inputs of size n.

Best case complexity

This complexity is for "optimists!" The *best case complexity* of an algorithm is the fastest the algorithm can run on an input of size n, that is, it is the *minimum* time needed among all inputs of size n to solve the problem.

In symbol, if $T_A(n)$ is the *minimum* time taken to solve a problem π of size n using algorithm A, then

$$T_A(n) = \min\{\, T(I_n) | I_n \in \pi_n \,\}$$

where π_n is the set of all inputs of size n and $T(I_n)$ is the time taken by the algorithm A to execute input I_n.

To express the best-case complexity of an algorithm, the following notation will be useful:

Big omega notation

Consider two real-valued functions f and g defined on the set of natural numbers. The notation $f(n) = \Omega(g(n))$ (read: f of n is big omega of g of n) means that the function $|f(n)|$ *dominates* the function $c|g(n)|$ where $c > 0$ is a constant, that is, $|f(n)|$ is *at least* a constant times $|g(n)|$ if n is sufficiently large.

Symbolically, there exist two constants $c > 0$ and a natural number n_0 such that the inequality $|f(n)| \geq c|g(n)|$ holds if $n \geq n_0$. In other words, the quotient, $\frac{|f(n)|}{|g(n)|}$ is bounded *below* by a constant if n is sufficiently large and $g(n) \neq 0$.

Example 5.2.3 (Big omega):
We can write

$$1 + 2 + \ldots + n = \Omega(n^2).$$

because we know that

$$1 + 2 + \ldots + n = n(n+1)/2.$$

So, it is enough if we prove that the value of $n(n+1)/2$ is at least a constant times the value of n^2 if n is sufficiently large. Now $n(n+1)/2 = (n^2+n)/2 = n^2/2 + n/2 \geq n^2/2 = (1/2)n^2$ if $n \geq 1$. Here the constant c in question is $1/2$ and we are not interested in finding the maximum constant satisfying the inequality. The lower bound for n for which the inequality holds is 1.

The following simple proposition gives the relation between the "big oh" and "big omega" notations:

Proposition 5.2.1:
$f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$.

Proof. By definition, $f(n) = O(g(n)$ if and only if there exist two constants $c > 0$ and a positive integer n_0 such that the inequality
$|f(n)| \leq c|g(n)|$ for all $n \geq n_0$, that is, if and only if $|g(n)| \geq (1/c)|f(n)|$ for all $n \geq n_0$, that is, if and only if $g(n) = \Omega(f(n))$. ■

Big theta notation

Big theta notation is used to express the *exact* time complexity function. We write $f(n) = \Theta(g(n))$ (read: f of n is big theta of g of n) means the function $f(n)$ at the same time dominates and is dominated by a constant multiple of $g(n)$, that is, $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

In other words, there exist two constants $c_1 > 0$, $c_2 > 0$ and a natural number n_0 such that the inequality

$$c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ holds for all } n \geq n_0.$$

Example 5.2.4 (Big theta):
By Examples 5.2.2 and 5.2.3, it is clear that

$$1 + 2 + \ldots + n = \Theta(n^2).$$

An improved version of Euclid's algorithm

Consider the algorithm of Example 5.1.1. In this algorithm, if $m = 1000$ and $n = 1$, then the algorithm will take 999 steps/iterations to output the result 1; ($\gcd(1000, 1) = \gcd(999, 1) = \gcd(998, 1) = \cdots = \gcd(1, 1) = 1$). Since division is nothing but repetitive subtraction, we obtain the following improvement of algorithm of Example 5.1.1.

Example 5.2.5:
Input: Two positive integers m and n.

Output: The $\gcd(m, n)$.

Table 5.2: Execution of an improved version of Euclid's algorithm

m	n	$m \bmod n$	Is $m \bmod n = 0$?
132	276	132	*no*
276	132	12	*no*
132	12	0	*yes*

Algorithm: Replace the greater (not necessarily strict inequality; if $m = n$, the algorithm substitutes for m, the remainder $m \bmod n$ which is 0) of the two integers by the remainder of the division of two integers till the remainder of the division of the two integers becomes 0. Symbolically,

$$\gcd(m,n) = \begin{cases} n & \text{if } m \bmod n = 0 \text{ (basis).} \\ \gcd(n, m \bmod n) & \text{Otherwise (iteration).} \end{cases}$$

Let us execute the improved version of Euclid's algorithm on a sample set of data with $m = 276$ and $n = 132$. Table 5.2 traces each step of the execution.

The algorithm terminates when $m \bmod n = 0$. The greatest common divisor is the final value of n, which is 12.

The reader may easily verify that for the sample data with $m = 1000$ and $n = 1$ the improved algorithm 5.2.5 takes only one step whereas Algorithm 5.1.1 will go through 999 steps/subtractions!

Extended version of Euclid's algorithm

The following theorem is the basis for the extended version of Euclid's algorithm

Theorem 5.2.1 (Bezhout):
If m and n are positive integers, then we can find integers a and b such that

$$am + bn = \gcd(m,n).$$

Moreover, $m \perp n$ (m relatively prime to n or m and n are coprime) if and only if there are integers a, b such that $am + bn = 1$.

EXAMPLE 5.2.6 (Extended version of Euclid's algorithm):
Let $m = 18$ and $n = 12$. Then $\gcd(m, n) = 6$. We may take $a = 1$ and $b = -1$, so

$$am + bn = \gcd(m, n).$$

Let $m = 1769$, $n = 551$. Then by applying Euclid's gcd algorithm, we find $\gcd(1769, 551) = 29$. We may take $a = 5$ and $b = -16$ because

$$(5 \times 1769) - (16 \times 551) = 29.$$

In fact, the following min-max equality holds:

$$\min(am + bn > 0 | a, b \text{ are integers}) = \max_{d \text{ divides } m, d \text{ divides } n} d.$$

Given two positive integers m and n, the following algorithm finds two integers a and b such that

$$am + bn = \gcd(m, n).$$

Extended version of Euclid's algorithm

Input: Two positive integers m and n.

Output:Two integers a,b and their gcd d satisfying the equality:

$$am + bn = d.$$

Algorithm:

Let us write a fragment of the program in C (see Table 5.3):

Invariant of the above loop :

The algorithm works thanks to the two equations $am + bn = d$ and $ap * m + bp * n = c$ and the assignments inside the loop do not modify these two equations. These two equations constitute the "loop invariant" (we will study this concept later in this chapter). This loop invariant concept is used to prove the correctness of the algorithm.

Table 5.3: Implementation of extended version of Euclid's algorithm in C

```
int a,ap,b,bp.c,d,r,q,t;
//Initialization of a,ap,b,bp, c,d. ap for a prime. bp for b
prime. We work with c,d
a=0;ap=1;b=1;bp=0;c=m;d=n;
// we have: $am + bn = d$ and $ap * m + bp * n = c$.
r=c%d;q=(c-r)/d;//r, the remainder. q, the quotient.
$c = qd + r, 0 \leq r < d$.
//Observe that if $r = 0$, the program ends; we have in this
case: $am + bn = d$ as desired
while ($r > 0$) {
    c=d;d=r;
    t=ap;ap=a;a=t-qa;
    t=bp;bp=b;b=t-qb;//Still we have:$am + bn = d$ and
$ap * m + bp * n = c$.
};//end while.
//We quit the loop with r=0 and we have $am + bn = d$.
```

Let us execute the above program with $m = 55$ and $n = 25$ (see Table 5.4).

In Table 5.4, from the last row $d = 5$, $a = 1$, $b = -2$ and we have the equality $(1 \times 55) - (2 \times 25) = 5$.

Application of the extended version of Euclid's algorithm to partial fractions:

To simplify the sum and difference of rational numbers like $\frac{1}{5} - \frac{1}{2} + \frac{1}{3}$, we first find the lcm (least common multiple) of the denominators $5, 2, 3$, which is 30, and then simplify the expression

Table 5.4: Execution of extended version of Euclid's algorithm

ap	a	bp	b	c	d	q	r
1	0	0	1	55	25	2	5
0	1	1	-2	25	5	5	0

into $\frac{1}{30}$.

We now consider the inverse problem of writing $\frac{1}{30}$ into the sum or difference of rational numbers in which the denominators are either prime factors of 30 or squares of the prime factors.

The method is illustrated by Example 5.2.7.

Example 5.2.7:

[Application to partial fractions] Write $\frac{1}{30}$ in partial fractions.

Solution: First decompose 30 as the product of prime numbers. This is always possible by the fundamental theorem of arithmetic). We have $30 = 2\times3\times5$. Now the factors 2 and 3 are relatively prime, that is, $\gcd(3, 2) = 1$. By applying the extended version of Euclid's algorithm, we find two integers a and b such that $3a + 2b = 1$. For example, we may take, $a = 1$ and $b = -1$.

That is, $(1\times3)-(1\times2) = 1$. By dividing both sides by $6 = 3\times2$ we get,

$$\frac{1}{2} - \frac{1}{3} = \frac{1}{6}.$$

Now $6 \perp 5$, that is, $6 = 3 \times 2$ is relatively prime to the factor 5. Again by applying the extended version of Euclid's algorithm, we can find two integers a and b (for example, $a = 1$ and $b = -1$) such that $6a + 5b = 1$, that is, $(1 \times 6) - (1 \times 5) = 1$.

By dividing both sides of this equation by 30 $(30 = 6 \times 5)$ we have,

$\frac{1}{5} - \frac{1}{6} = \frac{1}{30}$. Now we plug $\frac{1}{6} = \frac{1}{2} - \frac{1}{3}$ into the preceding equation to obtain

$$\frac{1}{5} - \frac{1}{2} + \frac{1}{3} = \frac{1}{30}.$$

As another example, consider the rational number $\frac{1}{12}$. $12 = 2\times2\times3$. $\gcd(2, 2) = 2$. By the extended version of Euclid's algorithm, we may take $a = 2$ and $b = 1$ and $(2 \times 2) - (1 \times 2) = 2$. Divide by 4 to get $\frac{1}{2} - \frac{1}{4} = \frac{1}{4}$.

Now $4 \perp 3$. We may take $a = 1$ and $b = -1$. Divide 12 to get,

$\frac{1}{3} - \frac{1}{4} = \frac{1}{12}$. Now plug $\frac{1}{4} = \frac{1}{2} - \frac{1}{4}$ into the preceding equation, and we get

$$\frac{1}{3} - \frac{1}{2} + \frac{1}{4} = \frac{1}{12}.$$

Definition 5.2.2 (polynomial function):
A function $p(z)$ is a polynomial function if it can be written in the form

$$p(z) = a_k z^k + a_{k-1} z^{k-1} + \cdots + a_1 z + a_0$$

where $a_0, a_1, \ldots, a_k$ are real or complex numbers, and k is an integer ≥ 0. If $a_k \neq 0$, then k is called the degree of the polynomial.

Good algorithm or polynomial time algorithm

Consider a problem P and an algorithm A for solving P. The algorithm A is said to be a *good or polynomial time algorithm* if its time complexity is $O(p(n))$ where $p(n)$ is a polynomial in n and as usual n is the size of the problem P.

Exponential time algorithm

An algorithm whose time complexity is *not* bounded above by a polynomial function of the input size n is called an *exponential time algorithm.*

For example, Algorithm 5.1.1 is an exponential time algorithm, whereas the improved version of Euclid's algorithm 5.2.5 is a polynomial time algorithm as we will see later (to represent the positive integer n in binary notation, we need only about $\log_2 n$ not n bits, so we want an algorithm which is polynomial in $\log_2 n$ and not in n).

We shall now prove the following results involving big oh notation. These are useful when finding the complexity of algorithms:

Theorem 5.2.2:
If $T(n)$ is a polynomial of degree m with real coefficients, then $T(n) = O(n^m)$.

Proof. Let $T(n) = a_0 + a_1 n + a_2 n^2 + \cdots + a_m n^m$ where a_i's are real numbers.

We have to show that $|T(n)| \leq cn^m$ for some positive constant c and for $n \geq n_0$.

$$|T(n)| \;=\; |a_0 + a_1 n + \cdots + a_m n^m|$$

$$\begin{aligned}
(\textit{by triangle inequality}) &\leq |a_0| + |a_1 n| + \cdots + |a_m n^m| \\
(\textit{for } n \geq 1) &\leq |a_0| + |a_1|n + \cdots + |a_m|n^m \\
&\leq n^m(\frac{|a_0|}{n^m} + \frac{|a_1|}{n^{m-1}} + \cdots + |a_m|) \\
&\leq n^m(|a_0| + |a_1| + \cdots + |a_m|) = c.n^m.
\end{aligned}$$

In the last step, we have used the fact that $\frac{|a_i|}{n^{m-i}} \leq |a_i|$ for $n \geq 1$.

The constant c is $|a_0| + |a_1| + \cdots + |a_m|$. ■

The rule of sum

PROPOSITION 5.2.2:
Let $T_1(n) = O(f_1(n)$ and $T_2(n) = O(f_2(n))$ then $T_1(n) + T_2(n) = O(\max(|f_1(n)|, |f_2(n)|)$.

Proof. We have to prove that $|T_1(n) + T_2(n)| \leq c \max(|f_1(n)|, |f_2(n)|)$ for $c > 0$ and for all $n \geq n_0$. $T_1(n) = O(f_1(n))$. Therefore by definition, $|T_1(n)| \leq c_1|f_1(n)|$ for $c_1 > 0$ and for all $n \geq n_1$.

$T_2(n) = O(f_2(n))$. Therefore by definition, $|T_2(n)| \leq c_2|f_2(n)|$ for $c_2 > 0$ and for all $n \geq n_2$.

If $n_0 = \max(n_1, n_2)$, the above two inequalities are simultaneously satisfied for $n \geq n_0$.

Hence by triangle inequality,

$$|T_1(n) + T_2(n) \leq |T_1(n)| + |T_2(n)| \leq c_1|f_1(n)| + c_2|f_2(n)|$$

Set $c = \max(c_1, c_2)$. Since $c_1 \leq c$ and $c_2 \leq c$, we have $|T_1(n) + T_2(n)| \leq c(|f_1(n)| + |f_2(n)|)$

Now set $f(n) = \max(|f_1(n)|, |f_2(n)|)$ for $n \geq n_0$. Since $f_1(n) \leq |f(n)|$ and $f_2(n) \leq |f(n)|$ for $n \geq n_0$, we have $|T_1(n) + T_2(n)| \leq 2cf(n)$. ■

We have proved the rule of sum for two functions T_1 and T_2. There is nothing special about the number 2, that is, the rule of sum can be extended to k functions for any $k \geq 2$.

In essence the rule says the following:

The time complexity of the sequence of k instructions I_1, $I_2, \cdots, I_k$ is $O(\max(T(I_1), T(I_2), \cdots, T(I_k)))$ where $T(I_j)$ is the time needed to perform the instruction I_j. Intuitively, we can ignore lower-order terms and the multiplicative constants in a sum, when we calculate the complexity.

> In big O-notation, a big term subsumes the smaller ones, that is, the big one eats away the smaller ones!

EXAMPLE 5.2.8:
Rule of sum:

$100 \log n + 40\sqrt{n} + 1000n \log n + 4n^2 = O(n^2)$ as lower-order terms and the multiplicative constants can be ignored, thanks to big oh notation.

EXAMPLE 5.2.9:
Consider the functions $T_1(n)$ and $T_2(n)$ defined as below:

$$T_1(n) = \begin{cases} n^4 & if\ n\ if\ n\ is\ even \\ n^2 & if\ n\ if\ n\ is\ odd \end{cases}$$

$$T_2(n) = \begin{cases} n^2 & if\ n\ if\ n\ is\ even \\ n^3 & if\ n\ if\ n\ is\ odd \end{cases}$$

Then by the sum rule, $T_1(n) + T_2(n) = O(\max(n^4, n^3))$, which is n^4 if n is even, n^3 if n is odd.

Rule of product

PROPOSITION 5.2.3:
Let $T_1(n) = O(f_1(n)$ and $T_2(n) = O(f_2(n))$ then $T_1(n)T_2(n) = O(f_1(n)f_2(n))$.

Proof. We have to prove that $|T_1(n)T_2(n)| \leq c|f_1(n)f_2(n)|$ for some constant $c > 0$ and for all $n \geq n_0$. $T_1(n) = O(f_1(n))$. Therefore by definition, $|T_1(n)| \leq c_1|f_1(n)|$ for $c_1 > 0$ and for all $n \geq n_1$.

$T_2(n) = O(f_2(n))$. Therefore by definition, $|T_2(n)| \leq c_2|f_2(n)|$ for $c_2 > 0$ and for all $n \geq n_2$.

For $n_0 = \max(n_1, n_2)$, the above two inequalities are simultaneously satisfied.

Hence for $n \geq n_0$, we have

$$\begin{aligned}|T_1(n)T_2(n)| &= |T_1(n)||T_2(n)| \leq c_1c_2|f_1(n)||f_2(n)| \\ &\leq c_1c_2|f_1(n)f_2(n)|.\end{aligned}$$

■

One should be careful in manipulating big oh notation. First of all, the equality "=" used in big oh notation is *not* a symmetric relation. For example, $4n^2 = O(n^2)$, but we should not write $O(n^2) = n^2/2$, otherwise we will have the absurd relation $4n^2 = n^2/2$. Example 5.2.10 illustrates one possible trap.

Example 5.2.10:
[Fallacy in O-symbol manipulation][4]. Find the error in the following argument: We know that a linear function in n, $an + b$ is $O(n)$. Therefore, $n = O(n)$, $2n = O(n)$,$\cdots$, .5.2.2. Hence we can write, $\sum_{k=1}^{n} kn = \sum_{i=1}^{n} O(n) = O(n^2)$.

Answer: Here, n, is a variable (the size of the input). We have replaced kn by a single O-symbol, that is, $|kn| \leq cn$, which is not true if k is a function of n, for example, $k = n$ or $k = n/2$. In fact, the sum is, $n(1 + 2 + \cdots + n)$= $n\sum_{k=1}^{n} k = n \times n(n + 1)/2$, which is a polynomial of degree 3, and hence $O(n^3)$.

We now present a formal definition of an exponential function.

Definition 5.2.3 (Exponential function):
$f(n)$ is an exponential function if there are constants $c_1 > 0, k_1 > 1, c_2 > 0, k_2 > 1$ such that

$$c_1k_1^n \leq f(n) \leq c_2k_2^n$$

for all but a finite number of values of n.

Table 5.5 describes the commonly encountered complexity functions and the corresponding algorithms. These algorithms will be treated in the text. As usual, the integer n represents the size of the problem. The logarithms in Table 5.5 are to the base 2.

Table 5.5: Table of complexity functions and their corresponding algorithms

Complexity Functions	*Algorithms*
$\log n$	Bisection Method or searching in a sorted array
n	Linear search in an unsorted array
$n \log n$	Quick sort (Expected complexity)
n^2	Insertion sort
n^3	Multiplication of two matrices of order n
2^n	Brute force or exhaustive search
$n!$	Generating all permutations of n elements

REMARK 5.2.2:
The most convenient base of logarithms in mathematics is the real number e, perhaps because the derivative of e^x is the same function e^x, but in computer science the most suitable base of logarithms is the integer 2 because computer programs often make two way alternatives (just like in trigonometry, the convenient measure of angles for practical purpose is the angle measure in degrees but for theoretical purpose, the convenient measure is the radian measure)[4]. The base of the logarithm modifies the constant c of the complexity function but not the function. Further the numbers are represented in base two notation.

> The logarithms of the same number with two different bases are proportional to each other.

In Table 5.5, except the functions $2^n$and $n!$, all other functions are polynomials. One may think that by designing more and more powerful processors, the computational complexity of algorithms can be ignored. But just the opposite of this statement is true.

Consider a problem P and five different algorithms $A_1, A_2, \ldots,$ A_5 to solve the problem P. Let the respective complexities of the algorithms be $\log n, n, n \log n, n^2, 2^n$. Here the complexity functions express the execution time in terms of microseconds, that is, one unit of time is $10^{-6} sec$. For example, the algorithm A_4 processes an input of size n in n^2 units of time. Suppose we are given *one second* of computer time.

Table 5.6 gives the size of problems one can solve using these different five algorithms.

For example, the entries in the third column are obtained by solving the equations $\log n = 1000000$, $n = 1000000$, $n \log n = 1000000$, $n^2 = 1000000$,$2^n = 1000000$ for n.

Table 5.6: Comparison of polynomial and exponential time complexity functions

Algorithms	*Complexity functions*	*Maximum size processed in 1 second*
A_1	$\log n$	$2^{1000000}$
A_2	n	1000000
A_3	$n \log n$	≈ 62746
A_4	n^2	1000
A_5	2^n	≈ 20

> The function 2^n is the paradigm for non-polynomial growth.

EXAMPLE 5.2.11:
Prove that if k is any integer, then $e^n \neq= O(n^k)$. In particular, $2^n \neq= O(n^k)$.

Solution: $e^n = 1 + n/1! + n^2/2! + \cdots \infty$ (by definition). If $k > 0$, then $e^n > n^{k+1}/(k+1)$. Hence $e^n/n^k > n/(k+1)$. This means that e^n/n^k is not bounded by any constant. This proves the first assertion. Since $2 < e(= 2.718281...)$, $2^n \leq e^n$ for all $n \geq 0$. Hence, $O(2^n) = O(e^n) \neq= O(n^k)$.

Table 5.6 illustrates that with an exponential time algorithm 2^n, one can process/execute in one second a problem of size only 20, whereas with the quadratic polynomial function n^2 the size becomes 1000 and with the logarithmic function the size reaches $2^{1000000}$.

Even more interesting will be the study of the increase in the size of the problem with the corresponding increase in the speed of the processor.

Suppose we have a new computer which runs 1000 times faster than our present computer, that is, with the new computer one unit of time will be $10^{10^{-9}}$sec. Table 5.7 illustrates the increase in size we can achieve with the new machine, given a fixed computer time, say, one hour. In Table 5.7, for example, the fourth entry of the last column is obtained by solving for N_4, the new size, the equation $N_4^2 = 1000 s_4^2$, and the fifth entry of the last column is obtained by solving for N_5, the new size, the equation $2^{N_5} = 1000 \times 2^{s_5}$.

Table 5.7: Effect of new machine on polynomial and exponential time algorithms

Complexity functions	Size with present computer	With computer 1000 times faster
$\log n$	s_1	s_1^{1000}
n	s_2	$1000s_1$
$n \log n$	s_3	$\approx 1000s_3$ for large s_3
n^2	s_4	$\sqrt{1000}s_4 = 31.6s_4$
2^n	s_5	$s_5 + 9.97$

Let us note that from Table 5.7, with the linear algorithm n, the size of the problem that can be solved with the new machine is *multiplied* by 1000 whereas with the exponential algorithm 2^n the size is only *increased* by about 10. In general, with polynomial time algorithms, the *new* size of the problem is obtained by *multiplying* the old size by some constant factor, whereas with exponential time algorithm the size is obtained by only *adding* a constant term to the old size.

This is the reason why computer scientists consider polynomial algorithms as good or efficient algorithms.

Polynomial time algorithms are designed by gaining some deeper insight into the structure of the problem, whereas exponential algorithms are often obtained by variations of brute-force or exhaustive search methods. We call a problem *intractable* if there is *no* polynomial time algorithm to solve the problem.
Why are polynomial algorithms considered good?

Polynomials have the following properties:

1. Closure property of polynomials: They are closed under addition, multiplication and composition. In other words, the sum,

the product and the composition of two polynomials are still polynomials. No other smaller class of functions (for example, the logarithmic functions) having useful complexity bounds has this property. Intuitively, polynomial algorithms become "useless" in practice little by little as the size of the problem increases, whereas exponential algorithms become "useless" all of a sudden.

2. The formal computational models RAM (Random Access Machine), RASP (Random Access Stored Program Machine), and the Turing Machine are all polynomially equivalent under logarithmic cost (see [3]).

3. Polynomial time algorithms can be programmed using a given number of cycles with no jumps, whereas an exponential time algorithm does not have this property.

Remark 5.2.3:
The simplex algorithm to solve a linear programming problem takes exponential time in the worst case (Klee-Minty examples), but this algorithm behaves very well in practice, that is, it works as if it is an efficient algorithm. The simplex method takes linear time on the average (see [9]). Further, an exponential time algorithm may be useful if the size of the problem is bounded above by a suitable constant, that is, n is sufficiently small. To illustrate this assertion, consider two algorithms of time complexity n^3 and 2^n to solve the same problem. Since $2^n \leq n^3$ for $n \leq 9$, we may use the exponential time algorithm to solve problems of size less than or equal to 9. Here we have ignored the constants of proportionality c. If we have two algorithms of complexity $1000n^2$ and $2n^3$ to solve the same problem, then the $2n^3$ algorithm will be cheaper for all problems of size less than 500.

Example 5.2.12 ($n!$ is an exponential function):
Prove that $n^{n/2} \leq n! \leq \left(\frac{n+1}{2}\right)^n$.

Solution:

$$n! = 1.2 \cdots n = \prod_{p=1}^{n} p$$

Writing the product in the reverse order,

$$n! = n.(n-1)\cdots 1 = \prod_{p=1}^{n}(n-p+1)$$

Hence by multiplying the two factorials, we have

$$(n!)^2 = \prod_{p=1}^{n} p(n-p+1).$$

We shall prove that $n \le p(n-p+1) \le \left(\frac{n+1}{2}\right)^2$ for all p with $1 \le p \le n$.
We shall first prove $p(n-p+1) \ge n$. The inequality is clearly true if $p = 1$, for $1(n-1+1) \ge n$. Suppose $p \ne 1$. Then, $p(n-p+1) \ge n$, if and only if $n(p-1) \ge p(p-1)$, if and only if $n \ge p$ (since $p \ne 1$) which is true.
To prove $p(n-p+1) \le \frac{(n+1)^2}{4}$, we may also use differential calculus! Set $f(p) = p(n-p+1)$. Then $f'(p) = n-2p+1$ and $f'(p) = 0$ gives $p = \frac{n+1}{2}$. Now $f''(p) = -2 < 0$ and hence $f(p)$ has a maximum at $p = \frac{n+1}{2}$ and the maximum value is $f(\frac{n+1}{2}) = \frac{(n+1)^2}{4}$. But

$$\min_{1\le p\le n} f(p) \le f(p) \le \max_{1\le p\le n} f(p)$$

Thus, $n \le p(n-p+1) \le \frac{(n+1)^2}{4}$ and therefore,

$$\prod_{p=1}^{n} n \le \prod_{p=1}^{n} p(n-p+1) \le \prod_{p=1}^{n} \frac{(n+1)^2}{4}$$

$$\text{implies } n^n \le (n!)^2 \le \left(\frac{n+1}{2}\right)^{2n}.$$

Taking the square root, we obtain

$$n^{n/2} \le n! \le \left(\frac{n+1}{2}\right)^{n}.$$

EXAMPLE 5.2.13:
Calculating the complexity:
Consider the following procedure "mystery" written in Pascal:

```
procedure mystery( n : Integer);
var i, j, x, y : integer;
begin
  for i:= 1 to n do
    if i mod 2 = 1 then
    begin
      for j: = i to n do x: =x+1;
      for j:= 1 to i do y:= y+1;
    end;
end;
```

Let $A(n)$ denote the number of assignments performed by the procedure "mystery" to the variables x or y.

Express the function $A(n)$ using the big oh notation and big theta notation.

Answer:

We use the two integer functions: the lower integer function $\lfloor x \rfloor$ which is the greatest integer $\leq$ the real number x and the upper integer function $\lceil x \rceil$ which is the minimum integer $\geq$ the real number x. For example, $\lfloor 3.61 \rfloor = 3$ whereas $\lceil 3.61 \rceil = 4$.

The outermost loop is executed as many number of times as there are odd integers between 1 and n. If n is even, then there are $n/2$ odd integers between 1 and n and if n is odd there are $(n+1)/2$ odd integers between 1 and n. Combining, there are exactly $\lfloor (n+1)/2 \rfloor$ odd integers.

In the first inner loop, there are clearly $n-i+1$ assignments for the variable x and in the second inner loop, there are exactly i assignments to the variable y.

Hence $A(n) = \lfloor (n+1)/2 \rfloor (n-i+1+i) = \lfloor (n+1)/2 \rfloor (n+1)$.

Clearly, $A(n) \leq \frac{n+1}{2}(n+1)$ since $\lfloor x \rfloor \leq x$.

Now $A(n)$ is bounded by a polynomial of degree 2 and hence by Theorem 5.2.2 $A(n) = O(n^2)$. Further, $A(n) \geq \frac{n}{2}(n+1)$ since $\lfloor (n+1)/2 \rfloor \geq n/2$.

Hence $A(n) = \Theta(n^2)$.

Encoding integers: In the study of complexity, three types of integer encodings are used: *Unary, Binary and Uniform, or Arithmetic.* Consider a non-negative integer n.

Unary encoding: If we assume that to store the integer n in the memory, we need n computer words (a word is an addressable unit of memory), then we say that the coding is unary. For example, the roman numeral to represent 3 uses three strokes iii of i to represent the integer 3.

Binary encoding: This encoding is more realistic. If we use $l(n)$ bits (binary digits: 0, 1) to represent, then we say that the encoding is binary where $l(n)$ is defined as follows:

$$l(n) = \begin{cases} 1 & \text{if } n = 0 \\ \lfloor \log_2 n \rfloor + 1 & \text{if } n \neq 0 \end{cases}$$

In fact, $l(n)$ is the number of bits necessary to write the integer n in binary notation.

Remark 5.2.4 (common, binary, natural logarithms):
Logarithms with bases 10, 2, e($=\approx 2.71$) are called, respectively, common, binary, and natural logarithms. In computer science, binary logarithms are used.

Uniform or arithmetic encoding: If we use only one memory word to store the integer n, then the encoding is called uniform encoding. If all numbers in the calculation can be stored in one computer word, we use this encoding.

Unless otherwise mentioned, we use uniform encoding of integers.

Example 5.2.14 (Number of bits in n):
Show that the number of bits in the binary expansion of the integer $n > 0$ is exactly $\lfloor \log_2 n \rfloor + 1$.

Solution:

Let the binary writing of n consist of k bits. We have to prove that $k = \lfloor \log_2 n \rfloor + 1$.

The minimum value of a number written in binary with k digits is $1\overbrace{00\ldots0}^{n-1}$ and the maximum value of the binary with k bits is $\overbrace{11\ldots1}^{n}$.

Their corresponding values in decimals are

$$2^{k-1}$$

and

$$2^0 + 2^1 + \cdots + 2^{k-1} = 2^k - 1.$$

Hence, $2^{k-1} \leq n \leq 2^k - 1$. To remove 1 from $2^k - 1$ form of the inequality, we write: $2^{k-1} \leq n < 2^k$ (note the strict inequality on the right). Applying $\log_2$ to the previous inequalities, we obtain $k - 1 \leq \log_2 n < k$. This means that $\log_2 n$ is between two *successive* integers $k-1$ and k. This implies that $\lfloor \log_2 n \rfloor = k-1$.

Typical complexity functions and algorithms

`Constant function` c: The notation O(1) represents some constant. If every statement of an algorithm is executed at most some constant times, then the complexity is O(1). This is an ideal situation.

`Logarithmic function O(`$\log n$`):` This complexity arises in algorithms which convert a problem of size n into a similar subproblem of size $n/2$ such that the solution of the problem of size n is the same as the solution to the problem size $n/2$. This is called the *Divide and Conquer Technique.* The amount of "effort" needed to divide the given problem of size n into a similar subproblem of size approximately $n/2$ must be a constant. The successive sizes of sub-problems are approximately $n/2, n/2^2, \ldots, n/2^k$ respectively.

`Linear function O(n):` When an algorithm processes each input at most a constant times, the time complexity will be linear. This typically arises in an algorithm processing each input in a single loop consisting of a statement requiring $O(1)$ time.

`The function O(`$n \log n$`):` This function is obtained by employing the Divide-and-Conquer method to solve the problem. We divide a problem of size n into two similar sub-problems of size approximately $n/2$, solve the subproblems separately, and finally combine the solutions of the sub-problems into the solution of the original given problem. The amount of work necessary to divide the original problem into two similar subproblems should be a linear function in n.

`Quadratic function O(`n^2`):` This function is obtained in an algorithm which processes each pair of inputs (typically in a double-nested loop). For example, in a program with a double-nested loop consisting of a statement requiring O(1) time.

`Cubic function O(`n^3`):` We will have this complexity in an algorithm which processes all triples of inputs. For example, in a program with a triple-nested loop consisting of a statement requiring O(1) time.

`Exponential function O(`2^n`):` This complexity arises as a result of *brute-force or exhaustive search methods*.

5.3 An Overview of a Classical Computer

Algorithms exist independent of any computer. In the absence of a computer it is the human being who plays the role of a machine. Of course, humans are subject to errors, fatigue, etc., whereas computers can execute long sequences of instructions with reliability, and with thoughtless obedience.

The modern digital computer was invented to facilitate difficult, long and tedious calculations. A computer is only a machine which manipulates a material called “information.” *Information* can be described as a finite sequence of letters, digits, and some special characters like *, +, :, etc.

The information manipulated by a computer must be:

1. **Stored**: This is the role of the memory.

2. **Manipulated**: This is the role of the Central Processing Unit.

3. **Exchanged** (with a programmer or with other machines): This is the role of the input/output peripheral units like the keyboard and monitor.

In the figure below, a model of a classical computer is depicted (see [7]).

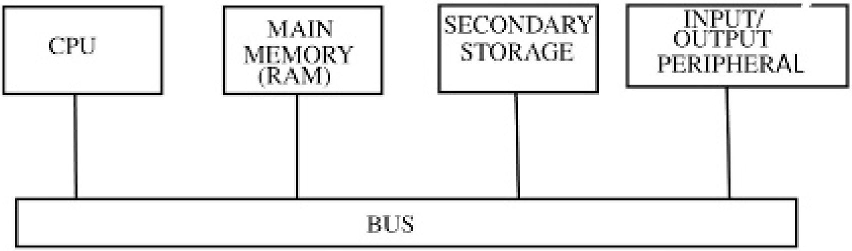


Figure 5.2: A model of a classic computer

Main memory

Information is stored in memory as a sequence of *bits* (*binary digits*). A binary digit is either 0 or 1. In fact, the information manipulated by a computer is materialized by two different voltages (0 means a high voltage, 1 means a low voltage, for example) in the computer's circuitry. The smallest storage unit of memory is referred to as a cell or a *byte.* We may treat a byte as a box containing a small chunk of information. Each byte possesses an *address* which distinguishes it from the other bytes. In fact, the addresses of the bytes are the non-negative integers starting from 0,1,... and ending with $n-1$ where n is the number of bytes in the *main memory.* Thus a byte is the smallest storage unit which has its own address, that is, the smallest addressable unit of memory. Usually a byte is a sequence of 8 bits.

It is generally 8, 16, 24 or 32 bits under high/low voltage. The set of all bytes is referred to as the main memory. In the main memory, the computer stocks not only the sequence of statements comprising a program, indicating the manipulation/computation to be carried out on the *data*, but also the data on which these statements of the program will be executed. The following parameters are associated with the memory: 1. The capacity or size of the memory. 2. The speed with which the instructions can be stocked and retrieved. The main memory is also called the *Random Access Memory* or briefly, RAM.

Central processing unit:

This is the "brain" behind the computer. All processing takes place in the *Central Processing Unit* or briefly, CPU. The CPU (in fact, the control unit of the CPU orders the execution) executes the instructions as follows:

1. *Fetch instruction*: The next instruction to be executed is brought by the processor from the RAM and stored in the appropriate *registers*. Registers are the CPU's own storage space.

2. *Decode instruction*: The instruction in the register is recognized or decoded.

3. *Fetch data/operands*: The data/operands on which the instruction is to be executed is brought from the main memory or register and stored in the appropriate register.

4. *Execute*: The instruction is executed.

5. *Depositing the result*: The result of the execution is written back to wherever it is supposed to go—either stored in the main memory or in the appropriate register. The following example illustrates the CPU's actions:

CPU actions

Example 5.3.1:
To evaluate the expression $x * y + t - z$ where x, y, z, t are integer variables.

Let $r1$ and $r2$ be two registers and let a be an integer variable used to store the intermediate/temporary result. Then the evaluation may proceed as follows:

```
$r1 := x;$
$r2 := y;$
$r1 := r1 * r2;$
$a := r1;$
```

$r1 := t;$
$r2 := z;$
$r1 := r1 - r2;$
$r2 := a;$
$r1 := r1 + r2;$
The final result is in the register $r1$.

The CPU is divided into two parts:

1. *The control unit*: The control unit orders the execution to be carried out.

2. *The arithmetic and logic unit*: The arithmetic and logic unit (briefly, ALU) executes instructions (see Figure 5.3). The arithmetic operations are addition, subtraction, multiplication and division and the logical operations are "or," "and," and "not." These operations are executed on the data stored in the main memory.

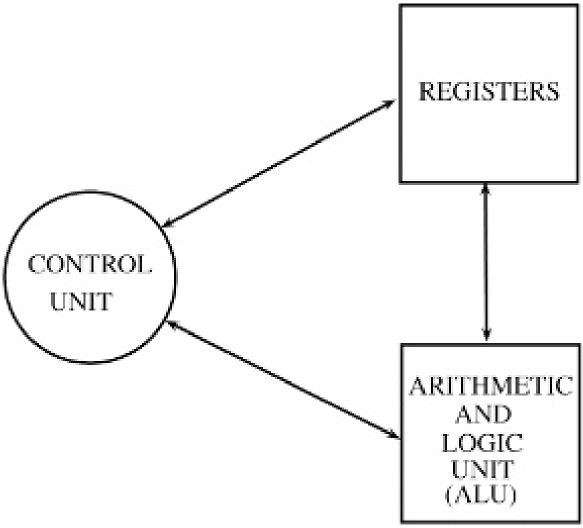


Figure 5.3: A view of the central processing unit

Hierarchy of memories:

In fact there are four levels of *memory hierarchy*:

1. Registers

2. Cache

3. Random access main memory (RAM)

4. Secondary or disk memory

Each level has a larger storage capacity than the preceding level but access time is greater than the preceding one. As we have already seen, registers are the processors' own storage elements. Access to the register is extremely fast. Access to the register is faster than access to the main memory.

The next memory level in the hierarchy is *cache*. Cache contains a small amount of main memory and it is a subset of *Static Random Access Memory* (briefly, SRAM). It is hoped that the cache contains the right contents of the main memory at the right time.

The next level is DRAM, *Dynamic Random Access Memory.* The adjectives "static" and "dynamic" are used according to the technology used to design memory cells. For example, each time we read information from the DRAM, the information read is lost and has to be rewritten. This is not the case with SRAM.

The last level in the memory hierarchy is the *auxiliary memory.* This is a peripheral device or a collection of devices such as disks. These devices can store much more information than the main memory but its access speed is slow compared to the main memory. The information stored in the secondary memories are permanent, in the sense that the data will be available after the computer is switched off and later switched on. The computer tries to keep the data it needs frequently in the main memory because of the slow access time of the peripheral memories.

Coding the information

The information manipulated by the computer must be *first* keyboarded, that is, captured by the computer, then manipulated/processed, and finally exchanged with the outside world. During each of these stages, the information is coded. A *coding* is simply a correspondence between two ways of representing the same information.

Example 5.3.2:
Coding the information:

For example, the different names of the months of the year January, February, . . . , December are usually written for simplicity using the integers $1, 2, \ldots, 12$. This way of writing the names of the months using the integers can be considered as a "code."

The widely employed coding of characters (10 digits, 11 to 25 special characters, some command characters) is the *ASCII code.* Its expansion is American Standard Code for Information Interchange. As its expansion indicates, this character set is used to facilitate exchange of information among the user, machine, printing device, etc. It is proposed by the *ANSI* (American National Standards Institute). In the ASCII code, each character is coded or represented by a unique sequence of 7 bits. This one-to-one correspondence or bijection between the ASCII code and the set of 128 characters is called a code. Therefore ASCII is referred to as seven-bit code. For example, the uppercase letter "A" is represented by the sequence 1000001. Its value in decimal is 65. Since $2^7 = 128$, using the ASCII code one can represent 128 characters.

The ASCII alphabet is partitioned into *printable characters* and *control characters.* There are 95 graphic/printable characters and 33 control characters. The control characters are used in data transmission and the control of printing equipment. The ordinal numbers from 0 to 31 and 127 represent control characters. For example, **cr** means "carriage return" and lf means "line feed." When we transmit the data to the printer, "cr" instructs the printer to begin a new line. The restricted ASCII code which uses only uppercase letters is used on commercial devices.

The ASCII code for the character 0 is 48 written in binary, the code for the character 1 is 49, etc. The code for the uppercase A is 65, again written in binary, the code for B is 66, etc. The ASCII code is *coherent and contiguous.* The other computer alphabet is EBCDIC (*Extended Binary Coded Decimal Interchange Code*) proposed by IBM (International Business Machine) for its 360 line of computers designed by Amdhal. It is an 8-bit code and hence can represent $2^8 = 256$ different characters.

Information processing and the hierarchy of languages

Computers work exclusively with binaries (machine language), a tedious language for human programmers. (There are also decimal computers!) A *machine language* is one which can be directly executed/interpreted by the computer.

> The machine language is the mother tongue of a computer.

Since machine languages are not suitable for humans, computer scientists have invented "high-level" languages which are in between the natural languages like French, English, etc. (which are imprecise and ambiguous for a machine, which acts exactly as it is told to do, no more or less) and machine language, which is cumbersome for human beings. Some "high-level" languages are Pascal, C, ADA, etc. A *program* is a sequence of elementary instructions. These high-level languages are translated by a program, called a *compiler*, into an equivalent language (see [10]).

The language translated by a compiler may be an *assembly language* or relocatable machine language. An assembly language is a symbolic and mnemonic version of the machine language in which names are used instead of machine/binary codes for operations. Names are also given to machine addresses. In the assembly language, words like "store," "add," "load," etc., replace the machine codes for operations. This assembly language will be further processed by the assembler to produce relocatable machine code. This relocatable machine code is finally passed to the load/link editor to produce "absolute machine code," which can be directly interpreted by the machine's circuitry. The role of the linker is to bind

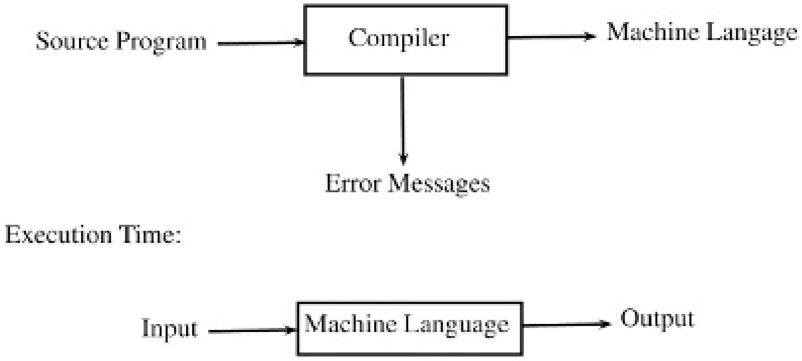


Figure 5.4: A view of a compiler

together the compiled parts (see Figure 5.4). The other method of executing a program written in a high-level language is by using "interpreters." *Interpreters* are used in command languages like Maple and MATLAB®. Operations executed in a command language may be an invocation of a large and complex program such as an editor or a compiler. An interpreter takes together a program and its input and furnishes the output. We see in the computer system, there are two different entities:

1. *Hardware*, which is the computer's circuitry.

2. *Software*, which is the collection of programs which facilitates the writing of other programs and which extends the capacities of the machine.

Finally, one should not forget the "humanware."

5.4 Introduction to Programming

The transformation of an algorithm in a precise formalized computer language is called a *program.* In a computer language, each instruction has a very definite meaning. A programming language consists of two components:

1. The *syntax/grammar* of the language, which describes what the program will look like.

> To command a programming language, we must obey its syntax and semantics.

2. The *semantics* of the language, which describes the meaning of each statement in a program.

We will be using a "pseudo-language," which is obtained by combining the constructs of a programming language Pascal together with some informal English statements. We will now describe briefly a few basic concepts of imperative programming language and the notation we will be using to express our algorithms:

Variable or Box:

> Once a person has understood the way in which the variables are used in a programming language, he has understood the quintessence of programming language.
>
> The concept of a variable represents an abstraction from its current value.
>
> *E. Dijkstra*

A *variable* or *identifier* is a symbol to which we can attribute several values of a given type. It can be considered as a "box" of suitable size in the memory of the computer whose name is the variable. We may not use the *keywords* of a language as identfiers as these keywords have special meaning for the language. For example, to declare a variable n of type integer in the language C we write: int n; in the language Pascal it is written as:

var n : integer;

A variable may be compared with a blackboard in a classroom. The content of the variable is equivalent to the writings on the board. Its value (contents) can be read or copied as many times as we like but once its content is erased by a new assignment (eraser), its previous value is lost.

Type:

There are two data types:

1. Built-in data types or pre-defined data types. (For example, integer, real. These basic types get machine support.)

2. User-defined data types, also called data structures, which we will study later.

A *type* is the set of values a variable can assume together with some operations to combine them. The type can be viewed as the trademark of a variable. For example, with the type integer, the typical operations in Pascal are : $+,-,*$, mod, div, etc. In the language C, the operations with the integers are: $+$, $-$, $*$, % (modulo operator), etc. The basic types in the language C are int (integer), float (real), char (character), etc.

All programs will be finally translated into a sequence of bits. Writing programs using only the bits would be cumbersome indeed. The concept of type permits us to specify the manner in which a particular set of bits will be treated and the functions allow us to describe the operations that will be performed on the data. We can now describe the concept of a variable in a programming language in all of its facets. In fact a variable can be viewed as consisting of four different components:

1. Name
2. A set of attributes
3. A reference or a box in memory
4. The content of the box (see Figure 5.5).

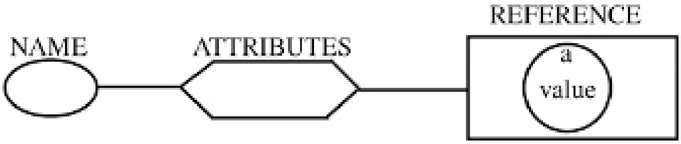


Figure 5.5: The four components of a variable

Unlike in mathematics, in computer science, the type integer represents only a *finite subset* of the set of integers. Of course, this set will differ from one computer to another. Similarly, the type real in a programming language represents a *finite* set of representatives of intervals in the real continuum. This is because the computer memory is finite. More precisely, to each real number x, there is associated a computer real number $r(x)$ called the representative of x (somewhat like a parliament member representing the people of his/her constituency). Every $r(x)$ represents many real numbers, with the condition that the set represented by $r(x)$ is a coherent interval on the real axis. The most commonly used format for reals is the IEEE format (Institute of Electronic and Electrical Engineers). The operations on the variables of the type integer are supposed to be exact except for the phenomenon of overflow. On the other hand, operations involving arguments of type real are "inexact." Two real numbers x and y are considered equal by a computer if $r(x) = r(y)$, that is, the computer real number of x is equal to the computer real number of y. In computer programs, the equality of two real numbers x and y is tested by the inequality $|x - y| < \epsilon$, where ϵ is a very small number. Note that this test of equality of two real numbers does not even satisfy the relation of transitivity; that is, $x = y$ and $y = z$ need not imply that $x = z$.

> All exact sciences are dominated by the idea of approximation.
>
> *B. Russel*

> Approximately equal is not an equivalence relation.
>
> In computer arithmetic, the usual associative and distributive laws are generally not valid.

Remark 5.4.1:
Inside a computer, the bit/internal structure representing an integer is usually different from the one representing a real number even though the integer and the real number denote the same value.

Assignment statement

The most important elementary instruction in a programming language is the *assignment* (or substitution or replacement) statement. If n and m are two integer variables, the substitution of the value of the variable m for the variable n is written as:

$n := m$; in Pascal.

In C we write this assignment as $n = m$;.

In writing algorithms, we write this as $n \leftarrow m$;. It should be noted that in an assignment statement, the right-hand side variable represents the content of the box (also referred to as the *r-value*) whose name is m and the left-hand side represents the reference component of the variable (also called the *l-value*) n.

Sometimes we use the term "*dereferencing*" for "r-value." The term dereferencing means the following: taking a reference and returning its associated value. This is also called the *contents operator*. In general, an assignment statement can be written as

"variable := formula";

where "formula" is an expression which gives a single value and this value will be assigned to the variable on the left-hand side.

Compound statement

A *compound statement* is a sequence of at least two elementary statements. In Pascal, a compound statement is enclosed between the two delimiters begin, end. In the language C, we use {, } as delimiters (see Figure 5.6). S, S_1, S_2, I represent statements or instructions.

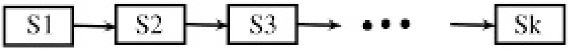


Figure 5.6: A compound statement

Conditional statement

If the traffic signal is green then the car moves; otherwise the car stops. If the student passes the final examination, then he goes on vacation during the summer; otherwise, he sits at home and revises for his second sitting.

The computer is capable of receiving instructions of this kind. In Pascal this conditional statement is expressed as follows:

$$\text{if } B \text{ then } S_1 \text{ else } S_2;$$

Here B is a Boolean expression which has a value either true or false. S_1 and S_2 are statements. If the result of evaluation of the Boolean expression is true then the statement S_1 is executed; otherwise the statement S_2 is executed.

The corresponding statement in C is exactly the same except for the word "then" and the semicolon may be written after the statement S_1 (see Figure 5.7).

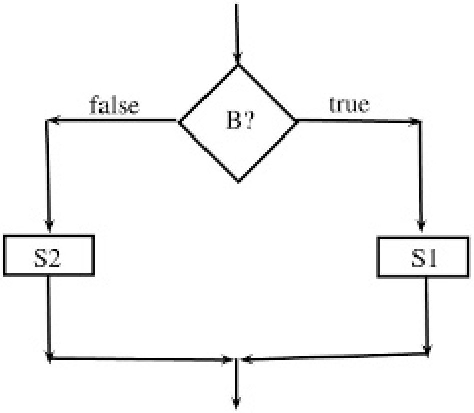


Figure 5.7: A flowchart for if-then-else statement

A variation of the above conditional statement is one in which the else part is absent. It is written in Pascal as:

if B then S;
In C we write this as:
if (B) S;
(See Figure 5.8 for conditional statement without “else” part.)

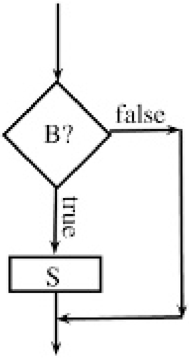


Figure 5.8: A flowchart for if-then statement

Example 5.4.1:
Conditional statement in Pascal:

if (n mod 2 = 0) then writeln(n, ’ is even’) else writeln(n, ’ is odd’);

Conditional statement in C:

if (n % 2 ==0) printf(”%d is even”, n) else printf(”%d is odd”, n); %d inside quote says n should be printed as a decimal number.

Repetitive statement or loop with precondition

Computers are well suited for performing a sequence of operations until a condition is satisfied. Machines can execute the same sequence of instructions thousands of times with absolute, thoughtless obedience and accuracy. This is where a human operator fails. A human being is subject to errors and fatigue which is not the case with an automaton.

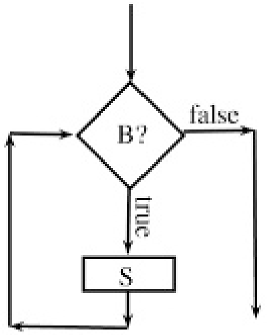


Figure 5.9: A flowchart for while-do loop

In Pascal the *repetitive or iterative* statement with precondition is expressed as below:

while B do S;

In C the construction is as follows:

while (B) S

Note that "do" is present in Pascal notation after while but not in C notation. (See Figure 5.9.)

Its meaning is the following: The Boolean expression B is evaluated first. If its value is true, then the statement S is executed. Again, the expression B is evaluated and if its value is true then we execute the statement S; if the value of B is false, then we quit the loop.

In brief, the statement S is executed as long as the value of the Boolean expression is true. This implicitly implies that the execution of the statement S should modify one or more variables present in the expression B, in order to terminate the loop. Let us note that if the first evaluation of B leads to a false value, the loop is *not* performed at all.

Example 5.4.2:
Iterative statement:

Let us write Euclid's algorithm 5.1.1 in Pascal and in C:

```
program euclid;
var m, n, x, y : integer;(* find the gcd of m and n*)
begin
  writeln(' enter two integers ');
  readln(m, n);(* read m and n*)
  x : = m; y: = n; (* we manipulate x, y, copies of m,n*)
  while x <> y do (*loop*)
  if x > y then x:= x - y
            else y:= y - x;
  (* we quit the loop with x = y*)
  writeln(' The gcd of ', m, 'and ', n, 'is ', x):
end;
```

The sentences between the delimiters (* and *) are comments to follow the program easily and these are ignored by the compiler. Let us write the same program in C:

```
#include <stdio.h> /* reference to standard input-output library*/
int main( )
{
  int m, n, x, y;
  printf(" enter two integers\n");/* \n is the new line character*/
  scanf(" %d %d", &m, &n);/* read m and n*/
  x = m; y = n;/* we manipulate x and y*/
  while ( x != y)
    if (x > y ) x = x - y; else y = y - x;
  /* we quit the loop with x = y*/
  printf(" The gcd of %d and %d is  %d", m, n, x);
}
```

The sentences within the delimiters /* and */ are comments to understand the program. These comments are ignored by the compiler. scanf reads the input line and assigns the values found there to the variables m and n. The character &, just before m and n in scanf, means that scanf accesses the variables indirectly.

Also note that the above programs do not terminate, that is, they have an *infinite loop* if the variable m is strictly positive and $n = 0$.

Loop with postcondition

In this kind of loop, the condition to stay in the loop is tested at the bottom of the loop. Note that this implies that the loop is executed at least once. This kind of loop should be used only if we know in advance that the loop will be performed at least once.

In Pascal, this loop is written as follows:

repeat S until condition;

Its meaning is as follows: the loop is executed until the condition becomes "true." In other words, the loop is performed as long as the condition remains "false" (see Figure 5.10).

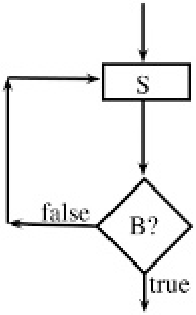


Figure 5.10: A flowchart for repeat-until loop

In C, the loop with postcondition is written as below:

do S while condition;

The loop is executed as long as the condition remains "true" (see Figure 5.11).

A program represents a class of computations and determines a model of behavior for an indeterminate, often infinite number of possible processes. Now the question is the following: How do we know that each possible computation evoked by a computer program will give the exact result? One brute-force answer is the following: Execute the program with all possible legal inputs and

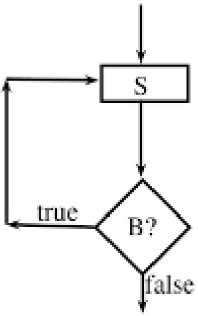


Figure 5.11: A flowchart for do-while loop

compare the answers obtained by some other means, for example, by hand calculation. This is humanly impossible taking into consideration the large amount of inputs involved and the time needed. So, the exhaustive method is not helpful in proving programs.

> Program testing (debugging) may be used to find
> the presence of errors, but never to prove their absence.
>
> *E. Dijkstra*

In order to prove that the loop calculates the desired value correctly on exiting, we now introduce the important concept of *loop invariant or general snapshot.*

Loop invariant

A *loop invariant* (or general snapshots or inductive assertion) is an assertion or a relation among variables of a program, which is always true at a given point in the loop, irrespective of the number of executions of the loop performed before. Mathematical induction and program proving are intimately related. To show that a loop gives the desired result on exiting, we use induction on the number of executions of the loop or some sort of induction on the values of an input variable. Invariants hold at particular points of a program text and they can be considered as a "bridge" between the *static* program text (spread out in text space) and the *dynamic* process of computation. In fact, the proof of a loop/inductive assertion involves the following three constituents:

1. Initialization: The invariant is true at/before the entry of the loop. (Induction basis)

2. Preservation: The assertion is *conserved* one execution after another. (Induction leap)

3. Termination/Finiteness : The loop terminates after a finite number of executions.

The following example illustrates this idea on Euclid's algorithm.

Example 5.4.3:
Loop invariant and termination:

Let us refer to the program of Example 5.4.2 written in Pascal implementing Algorithm 5.1.1. What is the invariant of the "while loop" of this program ?

The relation which holds at the *entry* of the loop is the following:

$$\gcd(m, n) = \gcd(x, y).$$

Let us prove this relation:

1. Basis: Since $x := m$; and $y := n$;, the relation is true before entering into the loop.

2. Induction leap: It is enough if we show that $\gcd(m, n) = \gcd(m - n, n)$ if $m > n$.

We will prove first that the set of all divisors of m and n is the same as the set of all divisors of $m - n$ and n.

If d is a divisor of m and n, then d divides the *difference* $m - n$ and n. Hence the set of divisors of m and n is a subset of the set of divisors of $m - n$ and n. Conversely, if d divides $m - n$ and n, then d divides the *sum* $m - n + n = m$ and n. Hence the set of divisors of $m - n$ and n is contained in the set of divisors of m and n. Hence the equality of the two sets of divisors.

In particular, we have $\max_{d|m,d|n} d = \max_{d|m-n,d|n} d$. Thus the induction leap is proved (read: $d|m$ as d divides m).

3. Termination: Initially, x and y are strictly positive integers. Set $N = \max(x, y)$. Inside the loop, the strictly greater of the two integers x and y is replaced by the difference. Thus the function N forms a strictly decreasing sequence of positive integers, which must obviously terminate.

Loop termination

To prove the termination or convergence of a loop,

$$\text{while } B \text{ do } S;$$

we normally find an integer function N depending on some of the variables of the program with the following properties:

1. If B is true then $N > 0$ on entry of the loop.

2. Each execution of the loop decreases the value of the integer function N.

Example 5.4.4 (Infinite loop):
If a C program has the following loop, then it will not terminate:

```
while (1) { }
```

Loop with post condition

In the while loop "while B do S" discussed above, if the Boolean condition B assumes the "false" value at the beginning, then the loop is *not* executed at all. Of course, any loop can be expressed with this "while" construction. For reasons of convenience, we see below a variant of this loop:

In Pascal, this loop is written as:

$$\text{repeat } S \text{ until } B;$$

The statement S is executed till the condition B takes the value "true" (*not* "false"). Note that, since the Boolean condition is tested at the end of the loop, the statement S is performed at least once.

In the language C, it is written as:

$$\text{do } S \text{ while } (B);$$

The statement S is executed as long as the condition B remains "true." This form of loop is executed at least once, since the condition to stay in the loop is tested at the end of the loop.

For loop:

In the case of the "while loop," it is difficult to predict in advance the number of times the loop will be performed, because it depends in general on the particular input. If we can compute in advance, the number of executions of the "while loop," the following construction will be very handy.

In Pascal, the "for loop" is written as:

$$\text{for } i := 1 \text{ to } n \text{ do } S \text{ ;}$$

The statement is executed exactly n times where n is a positive integer. If n is a non-positive integer, the loop is executed zero times.

In the language C, it is written as:

$$\text{for}(i = 1;\, i \leq n;\, i + +)\ S$$

(See Figure 5.12.)

Table 5.8 summarizes the notations of Pascal and C.

5.4.1 Parameter Passing

Modularizing programs into coherent pieces:

A module of a program is a sub-program (subroutine, procedure, function, etc.) which realizes some processes, which will be used to write the main program. Procedures are user-defined operations in contrast to built-in operations like addition among integers. This idea of *modules* of a program is equivalent to the concept of a lemma in mathematics. (A lemma is a result used to prove a theorem, which is comparable to the whole program.) This

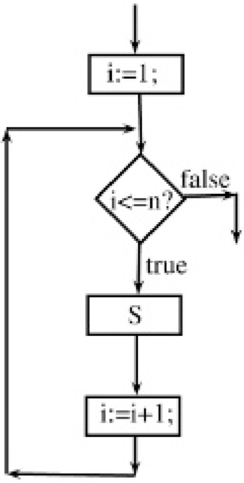


Figure 5.12: A flowchart for a “for loop”

concept facilitates the understanding and management of long and complex programs. Another advantage is avoiding the repetition of the same type of calculation in different parts of a program. Even if the program is short, it is advisable to modularize the code into coherent, and as far as possible into independent, pieces. Procedures/functions specify *what* is done without telling *how* it is done.

A procedure/function consists of four constituents:

1. Its name

2. The formal parameters

3. Type of value calculated by the procedure/function

4. The body, indicating how the computation is carried out in terms of parameters and other variables.

A subprogram which computes and returns a single value of type integer, real, char (scalar type or unstructured), pointer, etc. is called a “function” in Pascal, whereas a subprogram which is not

Table 5.8: Table of Pascal and C notations

Pascal Notation	*C Notation*
:= (Assignment)	= (Assignment)
<> (Not equal to)	!= (Not equal to)
= (Comparison)	== (Comparison)
or(logical or)	——(logical or)
and(logical and)	&&(logical and)
not(logical negation)	!(logical negation)
mod(remainder operator)	%(operator remainder)
readln (Read statement)	scanf (Read statement)
writeln (Write statement)	printf (Write statement)
if B then S_1 else S_2; (Conditional statement)	if (B) S_1; else S_2; (Conditional statement)
;(statement separator)	;(statement terminator)
while B do S (Loop)	while (B) S (Loop)
for i:= 1 to n do S (Loop)	for $(i = 1; i \leq n; i++)$ S; (Loop)
no ; before else	; before else

a function, that is, which does not compute a single value, is called a "procedure" or "subroutine." In the language C, analogous to the concept of a "procedure" in Pascal, is a function which returns *no* value or whose return type is "void." Let us illustrate this by writing a function computing the gcd in Pascal and in C.

Example 5.4.5:
Functions in Pascal and C computing the gcd using the improved version of Euclid's algorithm 5.2.5:

Pascal function for gcd:

```
function gcd( m, n : integer): integer;
(* m , n are formal parameters.
The types of parameters are integers.
The name of the function is gcd.
 The return type is integer,
which is indicated at the right end*)
var r: integer;(* local variable, representing remainder*)
```

```
begin
  r : = m mod n;
  while r < > 0 do
  begin
    m : = n;
    n : = r;
    r : = m mod n;
  end;(* while *)
  gcd: = n;
end;
Note how the final value computed is assigned
to the name of the function gcd.

$C$ function for gcd:\\
int gcd( int m, int n)
/* note the return type "int" is written
at the beginning of the function heading*/
{ int r;/* local or automatic variable*/
  r = m % n;/* % means mod*/
  while ( r ! = 0)
  { m = n;
    n = r;
    r = m % n;
  }
  return n
}
Note the final value is written after the return statement.
```

Of course, the functions in the above examples should be incorporated in the main program and *invoked* (called or activated) in order to obtain an executable program. Generally, there are *two* ways in which a function/procedure can establish a relation with other functions/procedures.

1. *Global variables*: These are the variables declared in the main program or in the environment surrounding the subprograms. The *environment* of a module is the set of variables defined outside the body of the module and which may be used or modified by the module at run time. The environment of the main program is the *operating system* of the computer, where standard objects are all predefined.

2. *Parameters*: The other way of establishing contacts between modules is with the help of parameters. The parameters introduced in the procedure/function heading, during the declaration are called the *formal parameters*. They exist only within the body of the subprogram and are local to it. During the activation

of a subprogram, the arguments replacing the formal parameters of a subprogram are called the *actual parameters*. There are *three* ways in which actual parameters can be substituted in place of the *corresponding* formal parameters.

Local variables: The variables declared inside a procedure/-function are local to the procedure/function and are unknown out of the text of the procedure/function. Such variables are called *automatic variables*, since they are *created* when the procedure/-function is called and are *destroyed* when we exit the procedure/-function. However, in the language C, if the keyword "static" appears before a local variable, such variables do retain their values from one call of the procedure to another. The word "static" means "only one copy."

We will describe each of these methods of substitution below.

Recall that a variable consists of four components:

1. Name,
2. Type,
3. Reference,
4. Content.

Value substitution: During the activation of a subprogram, if the value or content of the actual parameter is substituted for the corresponding formal parameters, then this substitution is called the *value substitution*. If the actual parameter is an expression/ formula, this expression is evaluated and the resulting value is assigned to the corresponding formal parameters. After this substitution, however, there no longer exists any connection between actual and formal parameters. In the value substitution, the formal parameters are considered as local variables of the subprogram and are initialized to the values of the actual parameters.

When do we use a value substitution?

When a parameter acts as an input to a subprogram and *not* a result of the subprogram, then the value substitution is generally appropriate. Of course there are exceptions in the case of an actual parameter representing a big structure like a matrix. If a big structure is *not* modified by the called subprogram, that is, none of its entries will be assigned a new value during the invocation, then we pass such a structure by variable substitution. This is done, because we don't want to waste time copying a big structure. In Pascal and C, the value substitution is made by default, that is, if there is no specification on the part of the programmer. In the language ADA, this kind of parameter passing is called *in* parameter.

Variable/reference substitution: In this case, the actual parameter must be a variable. The possible presence of indices in the variable are evaluated. The variable thus obtained is substituted for the corresponding formal parameters.

When do we use a variable substitution?

This is used if the parameter represents a result of a subprogram. In the language Pascal, this kind of substitution is indicated by the symbol *var* preceding the formal parameter. In the language C, the only parameter-passing mechanism is by value. However, the effect of variable substitution can be achieved by means of *pointers*. A pointer is a reference to some object. Putting it differently, a pointer is a variable whose value/content represents another variable. For example, a pointer concerning a human may be his email address or his telephone number or his residing address. In the language ADA, this kind of parameter passing is called *out* parameter or *in out* parameter.

A subroutine or procedure achieves its task either by changing one or more of its variable/reference parameters or by modifying some variables in the environment of the procedure or by modifying both.

Name substitution: The actual parameter is substituted *literally* for every occurrence of the corresponding formal parameters. We don't use this kind of parameter passing in our book.

Table 5.9 illustrates the notations used in mathematics, in Pascal and in C. The following examples clarify the parameter-passing mechanism:

Table 5.9: Notations in mathematics, Pascal, and C

Mathematics	Pascal	C
$f(a,b)=c$. (a,b,c are integers. For example,f can be gcd or lcm function etc)	function $f(a,b$:integer):integer;	int f(int a,int b)
$f(a_0,a_1,\ldots,a_{n-1})=b$ $a_0,a_1,\ldots,a_n,b$ are integers. For example, f may find the maximum, or minimum of $a_0,\ldots,a_{n-1}$. f does not change $a_0,\ldots,a_n$.	procedure f(var a:T;var b:integer); (To avoid copying, a is declared as a variable parameter; T is type array of n integers;	void f(int a[ ],int n,int *b) a is a pointer to a[0], the first element of array a. b is a pointer to integer.

Example 5.4.6:
Value and variable substitution:

Let us write a subprogram computing the product of two positive integers m and n using *only* the operation of addition, and store the resulting product in the variable p.

The algorithm behind this procedure is very simple: We have to decompose the product as a sequence of additions: Symbolically,

$$m \times n = \overbrace{m + m + \cdots + m}^{n\ terms}.$$

```
A Pascal procedure:\\
procedure multiply( m , n : integer; var p :integer);
(* m and n are value parameters by default.
p is the variable parameter*)
var i : integer;(* i an index variable *)
begin
  p : = 0; (* initialization*)
  for i : = 1 to n do
    p : = p + m;
end;(* multiply *)

To find the product of 7 and 5, we may write
in the main program, the following statements:
 a : = 7; b : = 5; (* a, b, c are global variables *)
 multiply( a , b, c );(* A call to multiply.
 m takes the value of 7 and n takes 5.
 c is identified with p. Finally, c = 35.
 The final result is made available in the variable c.*)
```

Example 5.4.7:
Variable substitution:

Let us write a procedure/function exchanging the contents of two integer variables, that is, if $m = 5$ and $n = 3$ initially, then after execution $m = 3$ and $n = 5$. What is the algorithm?

Consider the problem exchanging the contents of two bottles of wine (one liter each) without mixing them. We need a third empty bottle of capacity one liter. First, we pour the contents of the first bottle into the third empty bottle, then pour the contents of the second bottle into the now empty first bottle, and finally pour the contents of the third into the now empty second bottle. This is the algorithm. A Pascal procedure to exchange two variables:

```
   procedure permute( var x, y : integer );
   (* x and y are variable parameters. *)
   var temp : integer; (* temp is a local variable *)
   begin
     temp : = x; x : = y; y : = temp;
   end;

   In the main program, we write the following instructions:
   m := 5; n : = 3; (* m, n are global*)
   permute( m, n); (* A call to permute.
   m is identified with x and n with y *)
   After this call: m = 3 and n = 5.

   Function in C to exchange two variables:

   void permute( int *x, int *y)
   { /* A prefix *  indicates a pointer */
     int temp; /* temp automatic or local  variable */
     temp = *x; *x = *y; *y = temp;
   }

   In the main program, we write the following statements:
   m = 5; n = 3; /* m, n are global integer variables*/
   permute (&m , &n); /* Invoking permute. x is identified with &m,
   similarly y with &n. In C, *&m is the same as m, *&n is n */
   After the invocation, m = 3 and n = 5.

   On the other hand, consider the following function.
    void permute( int x, int y)
   { int temp; /* temp is local */
     temp = x; x = y; y = temp;
   }
   The main program has instructions: m = 5; n = 3;
   The call permute(m,n) does nothing as it swaps only the parameters
x and y and not the variables m and n. It leaves the value of
m,n unchanged.
```

5.4.2 Recursion

A *recurrence relation* is used to compute the terms of a sequence from the given initial values and the previously computed values. In mathematics, a function is said to be a *recursive function* if it is defined partially in terms of itself.

Similarly, in programming, a function which calls (invokes) itself either directly or indirectly is referred to as a recursive function. A function F invoking itself, that is, there is a statement in the body of the function F referring to itself, is called a *direct recursion.* This means that in graphical terminology, we have a loop around the vertex F. On the other hand, if the function F invokes the function G, which in turn invokes the function H, which again calls the function F, then such a situation is referred to as an *indirect recursion.* This means that we have a circuit of length three (F, G, H, F). Each arc is interpreted as an invocation. Recursion allows us to write compact and concise programs.

Example 5.4.8 (Recursive functions):
Recursive function:

The example which comes to our mind first is the *factorial function* defined as follows:

For a positive integer n, factorial n is the product of all first n natural numbers. More precisely,

$$n! = \begin{cases} 1 & \text{if } \ n = 0 \text{ (basis)} \\ n(n-1)! & \text{if } \ n \geq 1 \text{ (recursion.)} \end{cases}$$

Let us write a function computing $n!$ in Pascal and in C.

Factorial function in Pascal:

```
function fact( n : integer) : integer;
begin
  if n = 0 then fact : = 1 (* basis*)
           else fact : = n * fact (n - 1); (* recursive call*)
end;

Factorial function in $C$:

int fact(int n)
{ if (n == 0) return 1; /* basis */
   return n*fact(n - 1);/* recursive call */
}
```

Let us write a recursive function in C to find the gcd of two integers m, n implementing the following algorithm:

$$\mathrm{gcd}(m,n) = \begin{cases} m & \text{if } n = 0 \text{ (basis)} \\ \mathrm{gcd}(n, m \bmod n) & \text{if } n > 0 \text{ (recursion).} \end{cases}$$

Recursive gcd function in C :

```
int gcd(int m,int n)//m>0,n>=0
{    if (n==0) return m;//basis
return gcd(n,m%n);//recursive call
}
```

We notice that these functions, “fact” and “gcd,” are a direct translation of the mathematical definition of the factorial function and the gcd algorithm. Further, we don’t need to declare any auxiliary variable to write these functions!

Let us calculate the complexity of the function fact in the above example:

Let T(n) be the time needed to compute the n! Then we have the recurrence relation involving $T(n)$:

$$T(n) = \begin{cases} 1 & \text{if } \ n = 0 \\ T(n-1) + 1 & \text{if } \ n \geq 1. \end{cases}$$

Notice that the basis case costs one unit of time, whereas the costs of the statement involving recursion is $T(n-1)+1$, because the cost of fact$(n-1)$ if $T(n-1)$ and the cost of multiplication * is one unit.

Let us now solve the above recurrence relation.

$$T(n) = T(n-1) + 1$$

Replacing n by $n-1$, we get $T(n-1) = T(n-1-1)+1 = T(n-2)+1$. Substituting $T(n-1) = T(n-2)+1$ in the equation

$$T(n) = T(n-1) + 1$$

we obtain,

$$T(n) = T(n-2) + 2$$

Continuing like this, we get after n steps, $T(n) = T(n-n) + n = T(0) + n = 1 + n$ by using $T(0) = 1$. $T(n)$ is a linear function, hence $T(n) = O(n)$.

Let us write a non-recursive function computing $n!$.

Example 5.4.9 (Non-recursive factorial functions in Pascal and C):
Non-recursive factorial function: Pascal function:

```
function fact (n: integer);
var i, f : integer; (* i an index variable*)
begin
  (* initialization of i and f*)
  i := 0; f : = 1; (* 0! = 1*)
  while i < n do  (* iteration*)
  begin
    i : = i + 1; f : = i * f;(* i! = i (i-1)!*)
  end;
  fact := f ;
end;

C factorial function:

int fact( int n)
{ int i, f;
  i = 0; f = 1; /* initialization*/
  while (i < n ) /* iteration */
  {  i = i + 1; f = i * f;
  }
  return f
}
```

Remark 5.4.2:
During the initialization in programming, an empty sum is defined as *zero* because $sum + 0 = sum$ whereas, an empty product is defined as *one*, *not zero*, because $product \times 1 = product$. That is why we define $0! = 1$, $2^0 = 1$.

Example 5.4.10 (Fibonacci sequence):
The Fibonacci sequence $(f_n)_{n\geq 0}$ is defined as follows:

$$f_n = \begin{cases} 0 & \text{if } \ n = 0 \\ 1 & \text{if } \ n = 1 \\ f_{n-1} + f_{n-2} & \text{otherwise.} \end{cases}$$

Let us write a function in C computing the nth Fibonacci number. The function f_n has multiple (two) bases.

```
int fib( int n)
{ if (n == 0) return 0;/* basis*/
  if (n == 1) return 1; /* basis */
  return fib (n - 1) + fib (n - 2); /* two recursive calls*/
}
```

The above program is concise and compact but its complexity is exponential! The reason is the following: terms of the sequence are computed independently many times! (See the calling tree in Figure 5.13.)

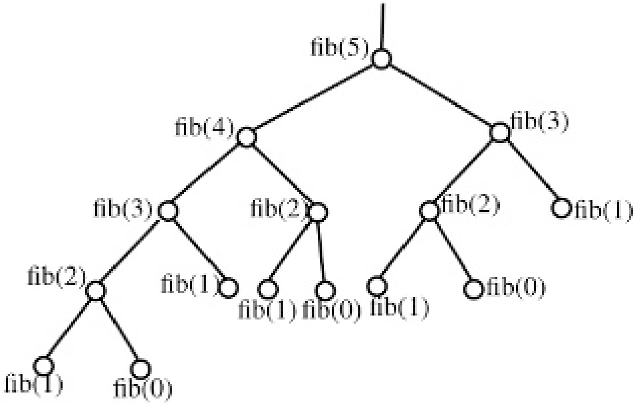


Figure 5.13: Tree illustrating the call fib(5)

This is an example to show that the recursion must not be used blindly. Let us calculate the complexity of the function fib(n). Let

$T(n)$ be the time needed to compute the fib(n). Then we have the following recurrence relation:

$$T(n) = \begin{cases} 1 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ T(n-1) + T(n-2) + 1 & \text{otherwise.} \end{cases}$$

Let us note that the two bases cost one unit each, the cost of the call fib$(n-1)$ is $T(n-1)$, the cost of the call fib$(n-2)$ is $T(n-2)$, and the cost of addition is one unit. We have to solve this recurrence relation, that is, we have to express $T(n)$ in terms of n.

To this end, we introduce the displacement operator E defined as below:

$E^0T(n) = T(n)$ and $ET(n) = T(n+1)$. We define the composition $E^kT(n) = E^{k-1}(ET(n))$ for $k \geq 2$. $E{-1}$ is the inverse of E, $E{-2}$ is the inverse of E^2 etc.

Let us come back to the recurrence relation $T(n) = T(n-1) + T(n-2) + 1$. For convenience, replacing n by $n+2$, we get:

$T(n+2) = T(n+1) + T(n) + 1$. Equivalently, using the operator E, $E^2T(n) = ET(n) + T(n) + 1$ which is

$$(E^2 - E - 1)T(n) = 1$$

The *characteristic equation* of the above difference equation is $x^2 - x - 1 = 0$. Solving this quadratic equation we get, $x = \frac{1 \pm \sqrt{5}}{2}$. Hence the two roots are $\phi = \frac{1+\sqrt{5}}{2}$, which is known as the *golden ratio* and $\hat{\phi} = 1 - \phi$.

Therefore, a solution of the recurrence relation $T(n+2) - T(n+1) - T(n) = 0$ is $T(n) = A\phi^n + B\hat{\phi}^n$ where A and B are constants which can be found from the initial conditions. Since the multiplicative constants can be ignored while using the big oh notation, we don't bother to find them explicitly.

Now we have to find a particular solution. A particular solution is $T(n) = \frac{1}{E^2-E-1}1 = -(1 + E - E^2)^{-1}1$.

Using the fact that $(1+x)^{-1} = 1 - x + x^2 - x^3 + \cdots$ we get $T(n) = -(1 - (E - E^2) + (E - E^2)^2 - \cdots)1$. But $E1 = 1$. Hence a particular solution is -1 (the reader may verify this by direct substitution in the equation) and therefore the general solution of the recurrence relation is

$$T(n) = A\phi^n + B\hat{\phi}^n - 1.$$

But $\lim_{n\to\infty} \hat{\phi}^n = 0$ because $|\phi| \approx |-0.6| < 1$. Hence $T(n) = O(\phi^n) = O(2^n)$. (Recall that we may ignore the terms of lower order while calculating the complexity.)

A non-recursive program to compute f_n

The recursive program to compute the n-th Fibonacci number takes exponential time, as we have seen. We give below a simple non-recursive program to compute f_n which takes only linear time.

Example 5.4.11 (A non-recursive function in *Pascal* to compute f_n:):
Non-recursive function for f_n

```
function fib( n : integer ): integer;
var i, c_t, p_t, temp : integer;(* c_t, the current term and
p_t, the preceding term, temp, a temporary variable*)
begin
  i := 1; c_t : =1 ; p_t : = 0;(* initialization*)
  while  i < n do
  begin
    temp : = c_t; i := i + 1;
    c_t : = p_t + c_t; (* each term is the sum of the
    two previous terms *)
    p_t:= temp;
  end;
  fib := c_t;
end;
```

Clearly the complexity of this non-recursive function is $O(n)$, since there is only one dominating loop which is executed exactly $(n-1)$ times. The other statements inside the loop together require time $O(1)$. Hence the complexity is $O(n)$.

The following Table 5.10 summarizes the complexities of various constructs in programming, that will occur frequently in practice:

Table 5.10: Summary of complexities of various constructs

Statement	*Complexity*
Assignment: $x := y$; (x, y are unstructured variables like integer, real, Boolean, char etc.)	O(1)
Compound statement: begin $S_1, S_2, \ldots, S_k$end;	O(max(Cost (S_1), Cost $(S_2), \ldots,$ Cost (S_k))
Conditional statement: if B then S_1 else S_2;	O(Cost of evaluation of B + max(cost(S_1), cost(S_2)))
Iterative statement: while B do S	O(Number of executions of the loop* cost(S)) (assuming the cost of evaluation of B = O(1))
for i:= 1 to n do S	O(n)(assuming the cost of the statement S = O(1))

for i:= 1 to n do
for j:= 1 to n do
S
	O(n^2)(assuming the cost of the statement S = O(1))

for i:= 1 to n do
for j:=1 to n do
for k:=1 to n do
S
	O(n^3)(assuming the cost of the statement S = O(1))

Reading an unstructured variable or Writing an unstructured variable	O(1)
Writing an expression	O(The cost of evaluating the expression)

5.5 Introduction to Data Structures

> What are we to think of this question: Is Euclidean geometry true? This question has no meaning. We could as well ask: If Cartesian coordinates are true and polar coordinates false? One geometry cannot be more true than another, it can only be more convenient.
>
> *Henri Poincaré*

> Algorithms + Data Structures = Programs
>
> *N. Wirth* [5]

Data structures are user-defined data types in contrast to built-in data types like integer, real, Boolean. Built-in data structures get machine support.

The problem-solving steps:

We are going to describe a method of solving problems known as the *top-down method*, that is from the whole to the parts. The top-down method leads to programs with inherent structure. The dual method is referred to as the *bottom-up method*, that is from the parts to the whole. There are three steps involved in the top-down method.

1. In the top-down method, we have to find first of all, a suitable mathematical model (other models are physical models,

logical models, etc.) representing the real-world problem. Some examples of mathematical models are sequences, sets, trees, and graphs. Once a suitable model is found, we try to find a solution in terms of this mathematical model. Results concerning the model can be used to solve the problem. We are interested in finding a solution in the form of an algorithm. During this first step, we express our algorithm in an informal way. This informal algorithm will not contain many details.

2. In the second step, we progressively transform our informal algorithm into a pseudo-program. A pseudo-program will have all the essence of a working program without taking into consideration the declarations of variables, types and syntax of a particular language like C. It may also have statements like "Let m be a minimum integer of the sequence S, etc." This pseudo-code will have more details compared to the informal algorithm. As we refine the pseudo-code by taking into consideration more details of the problem, we can identify the basic operations to be performed on the different data types. (A data type is a collection of objects together with various operations on the objects.) Once the operations on data types are identified, we create appropriate *abstract data types.*

> Mathematical Model + Various Operations defined on the Model = Abstract Data Type.

Stacks and Queues are classic examples of abstract data types. The idea of an abstract data type appears in the class type of the language SIMULA67.

3. In the third and final step of the stepwise refinement, we have to implement each of our abstract data types in a programming language like Pascal or C. We must also refine each informal instruction of the pseudo-code into a legal instruction of the language in which we like to execute our program.

> Abstract Data Type + Implementation = Data Structure

The data or information (data/information is a finite sequence of digits, letters, special symbols, etc.) to be manipulated by the computer, represents an abstraction on the part of the real world. For example, the data concerning a student in a university are perhaps his name, surname, date of birth, mailing address, courses, etc. Most probably this data may *not* include details like color of his hair, eyes, etc. The data we have at hand to manipulate inside a computer should represent relevant details on the part of the real world and must ignore other properties which are irrelevant to the problem at hand.

We would like to organize or structure the data so that operations on the data can be efficiently carried out, because our goal is to minimize the time taken to execute an algorithm. The operations that we would like to perform on the data are *storing* information in the memory, *retrieval* of information found in the memory, and *modification* of the information.

For example, consider a huge science library containing thousands of books, scientific journals, reviews, indices, etc. In order to use the library in an efficient way, the books should be arranged according to the different disciplines (mathematics, computer science, etc.). Further, the mathematics books may be divided into various topics like topology, differential equations, etc. and inside each such division the books are arranged according to the lexicographic order (like words in a dictionary) of the authors. Of course we need library people who manage the library!

In a similar manner, the raw data we have at hand must be given some structure in order to carry out the operations on the data. In fact, data representation is a mapping of abstract structure into the computer store/memory. Of course this structuring depends on the operations that will be performed by the algorithm.

The most basic way of organizing/representing a sequence or list of n elements is by the idea of an *array.* The mathematical counterpart of the concept of an array is the vector or a function whose domain is an index set. A *list* is a mathematical object or *data model* which is defined as follows.

A list is a sequence of zero or more elements $(a_1, a_2, \ldots, a_n)$ where each element a_i belongs to a given type. If $n = 0$, then the list is an empty list. The operations we would like to perform on this list are the following: Finding the length or the number of elements of the list, inserting a new element in the $i-$th place, suppressing the element in the j-th place, suppressing all the elements of the list to get the empty list, etc.

Example 5.5.1 (List operations):
Consider the list $L = (2, 1, 5, 0, 7)$ of integers. The length of the list is 5. Insertion of the integer 4 in the 3rd place of L (before the integer 5) modifies the list as $L = (2, 1, 4, 5, 0, 7)$ and its length becomes 6. Deleting or suppressing the last element, that is, the 6th element of L, gives $L = (2, 1, 4, 5, 0)$. Deleting all the elements of L leads to the empty list $L = ()$.

The data model list is principally represented by the two data structures called a one-dimensional array/vector (sequential allocation of memory) and a *linked list*, using a pointer mechanism. A pointer is a reference to an object usually implemented as a single machine address.

Representation of a list by array: An array is an aggregate of *homogeneous* data elements. Mathematically, an array is a mapping defined as follows in Pascal:

var a: array[domain_type] of range_type;

Quite often, the domain type of an array is a subset of integers. In fact, the entire random access memory of a computer can be viewed as an array where: memory: array[addresses] of word; where *word* is an addressable unit of memory, that is, which has its own address.

An array is represented as a finite number of contiguous memory spaces. Let us see how an array is declared in Pascal and C. In Pascal an array "a" of 5 integers (here the elements are integers) is defined as follows: var a: array[1..5] of integer;

The different integers of the array are $a[1], \ldots, a[5]$. In the language C, the same array is defined as follows: int $a[4]$; the different integers of the array are $a[0], \ldots, a[4]$. Note that in the language C, the lower index of the array starts from zero. Further, in the language C, the name of the array "a" also represents a pointer to the first element of the array, which is $a[0]$. Hence, there are two representations of the ith integer of the array "a," namely, $a[i]$ and $*(a+i)$. Mathematically, $a[i] = *(a+i)$.

Two-dimensional array or matrix

In mathematics, a *matrix* as we know is a rectangular array of elements arranged into rows and columns. The elements usually come from a field(real or complex). In the language Pascal, a square matrix m consisting of n rows and n columns with integer entries is defined as follows.

Matrix declaration in Pascal:
const $n = 10$;
var m: array[1..n,1..n] of integer;

The entry in the intersection of the ith row and jth column is denoted by $m[i,j]$. The same matrix m is declared in the language C as below.

Matrix declaration in C:
#define $n = 10$
int m[n-1][n-1];

The (i,j) entry of the matrix m is denoted by $m[i][j]$; Note that the entries of m are $m[0][0], m[0][1], \ldots, m[n-1][n-1]$. As an example of matrix manipulation, we shall study an algorithm to construct a *magic square* of odd orders.

Algorithm to construct a magic square of odd order:

A *magic square* is an $n \times n$ square matrix m satisfying the following properties.

1. Each entry belongs to the set $\{1, 2, \ldots, n^2\}$
2. The entries are all distinct.
3. The sum of the entries in each row and in each column and in each of the two diagonals is the *same*.

The following example illustrates the magic square.

Example 5.5.2 (Magic square):
A 3 × 3 magic square is given below.

8	1	6
3	5	7
4	9	2

The sum of the entries of each row and each column and each diagonal is 15.

We shall now describe an algorithm to construct a magic square of *odd* order.

Input: An empty $n \times n$ matrix m where n is an odd integer.

Output: An $n \times n$ magic square.

Algorithm: First of all, let us imagine three *identical copies* of the matrix m, one copy in the direction north of the matrix m, one copy in the direction east of the matrix m and one copy in the direction north-east of the matrix m as indicated in the following figure.

COPY	COPY
m	COPY

Now assign the integer 1 to the middle entry/cell of the 1st row/top row of the matrix m. Then assign the integers $2, 3, \ldots, n^2$ successively in this order to the cells of the matrix m by always traveling in the north-east direction according to the following conditions.

(1) If we fall off the matrix m by traveling through north-east $\nearrow$ (since we travel always in the north-east direction $\nearrow$, if we fall off the *original* matrix m, then we must land in one of the cells of a *copy* of the matrix m). Assign the next integer to the *corresponding* cell of the *original* matrix, if the corresponding cell is not yet filled. Otherwise, assign the next integer to the cell found just below the most recently filled cell.

(2) If we meet a cell of the original matrix which is already filled while traveling in the north-east direction $\nearrow$, then assign the next integer to the cell found just below $\downarrow$ the most recently filled cell (see Example 5.5.2).

Let us write a program to construct a magic square of odd order in the language C (see Figure 5.14). The program does not use the row zero and the column zero of the matrix $m[n][n]$. Now we will study the representation by linked list.

Representation of data model list by linked list

Each variable/node in the above figure consists of two fields: The integer field and the pointer field. Such a variable can be defined in Pascal by using the construction called "Record," and in the language C it is defined by means of the construction called "struct" (structure). Its mathematical counterpart is the Cartesian product. An array is an aggregate of *homogeneous* data elements whereas a record or structure is an aggregate of possibly *heterogeneous* data elements. In other words, it is a group of variables which are individually accessed by their field names. The field names are usually not integers as is the case with the arrays. This allows us to use intuitive names for the fields.

Example 5.5.3:
Heterogeneous data types:

Let us define the data type of complex numbers in Pascal. A *complex number* z is an expression of the form $x + iy$ where x and y are real numbers with $i = \sqrt{-1}$. x is the real part of z and y is the imaginary part of z. We now define the type "complex" in Pascal.

```
type complex = record
           rp : ;(* rp for real part *)
           ip : real; (* ip for imaginary part *)
end;(see figure below)\\
real
```

Note that there is no "begin" for the type complex but "end" is present. Let us define a variable z of type complex.

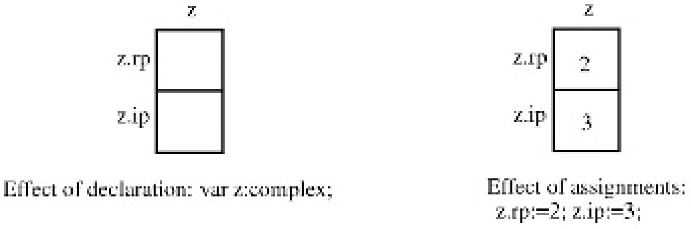
var z : complex; Now the variable z consists of two fields: rp and ip. The real part of z is denoted by $z.rp$ and the imaginary

Figure 5.14: Program to construct a magic square of *odd* order

```
#include<stdio.h>
#define n=6
{int m[n][n],i,j,nei,nej,key;
//(i, j) the current cell;(nei, nej)
//cell in the north-east ↗ of (i, j)
//initialization of matrix m = 0
//m[i][j] = 0 if and only if the cell (i, j) is not yet filled
for (i = 1;i <= n,i + +)
   for(j = 1;j <= n;j++)
       m[i][j] = 0;
//assign 1 to the middle cell of top row
m[1][n/2] = 1;i = 1; j = n/2;
(i, j) current cell
for (key = 2;key <= n ∗ n;key + +)
{//calculate the north-east cell of (i, j)
   if (i == 1) //fall off m
    nei = n;
    else i = i − 1;
    if (j == n) /:fall off m
    nej = 1;
    else j = j + 1;
    if (m[nei][nej] == 0)//(nei, nej) not yet filled
    {i = nei; j = nej;}
    else i = i + 1;//just get down one cell south ↓ of (i, j)
    m[i][j] = key;
}
//print the magic square m
for (i = 1;i <= n,i + +)
{
   {for(j = 1;j <= n;j++)
       printf("%4d",m[i][j]);
   printf("\n");

}
}
```

part of z is accessed by $z.ip$. A field is accessed by the name of the variable which is z and the name of the field which can be either rp or ip separated by a period. For example, the assignments $z.rp := 2$ and $z.ip := 3$ set the complex number variable z to $z = 2 + 3i$.



Effect of declaration: var z:complex;

Effect of assignments:
z.rp:=2; z.ip:=3;

Figure 5.15: Illustration of record variable z

Let us now write a function in Pascal to find the modulus of the complex number z where the modulus of z is denoted by $|z|$ and is equal to $\sqrt{x^2 + y^2}$.

```
function mod( z : complex): real;
(* The input parameter is z; The output type is real.
The name of the function is mod *)
begin
   mod : = sqrt(z.rp*z.rp + z.ip*z.ip);
end;
```

The type complex is defined in the language C as follows:

```
  struct complex { float rp; float ip };
/* note the semicolon after the right brace */
```

We now define a variable z of type complex as below:
complex z; The real part of z is denoted by $z.rp$ and the imaginary part is $z.ip$. The assignments $z.rp = 2$ and $z.ip = 3$ set the complex number $z = 2+3i$. The modulus function is written in C as follows:

```
float mod( float $z$)
```

/* The input parameter is $z$. The return type is float. The function name is mod.*/ return sqrt($z.rp * z.rp + z.ip.z.ip$) The square root function sqrt is found in the library routine `<math.h.>`

Let us declare a type consisting of nodes in Pascal where each node has two fields: a variable x of type integer and the second variable called "next" which is a *pointer* or address or reference to a variable of type node. A pointer is a reference to some object, that is, a pointer variable is one whose value represents another variable. Pointers are usually implemented as a single machine address independent of what they point to. Pointers are used for indirect addressing. Several pointers can point to the same variable, but a pointer cannot point to two different variables.

Example 5.5.4:
Heterogeneous data elements:

```
type node = record
var x: integer;
next: ^node;
(* ^ is the notation of pointer in pascal.
 ^ is read as : points to *)
end;
```

This type node is of a *recursive type*, that is, to define the type node we again use the node as its constituent. Note that the keyword "end" is present without the keyword "begin." This declaration describes the formats of a variable of type "node." A variable $v$ of type node may now be defined as follows:

```
var $v$: node;
```

The variable v has two fields: (1) x and (2) next. The "x" field of the variable v is denoted by $v.x$ and the next field of v is denoted by v.next. In other words, a field is accessed by the name of the variable and the name of the field separated by a period, or dot. The node pointed to by the pointer variable v.next is denoted by `v.next^`. Note that the pointer notation `‘‘^’’` is written as a postfix to the pointer variable v.next. There is a pointer constant called "nil" which points nowhere. This is equivalent to the integer 0 among the set of integers and the empty set $\emptyset$ in the theory of sets.

A variable of type `^node` is declared as: var p: `^node;` To create a node to be pointed by the variable p, we invoke the predefined procedure "new." The instruction to create a new node to be pointed (referenced) by p is new(p). new(p) creates a variable of type node and puts a pointer to it in the variable p. The node created by invoking the procedure new comes from a special region of memory usually referred to as *heap*. The node just allocated by the system is denoted by $p\hat{.}$ This is called the *dynamic allocation* of variable p^.

Pictorially, to indicate that the variable p is a pointer to a variable p, we draw an arrow from the variable p to the variable p (see Figure 5.16).

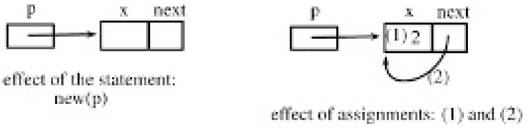


Figure 5.16: Illustration of pointer variable

Note that the notation ^ is used in two different ways according to the contexts. For example, if T is a type, then ^T represents the *pointer type*; and if p is a variable of type ^T, then p^, called the *dereferencing operator*, denotes the variable of type T pointed by the pointer p.

Let us pause and look at an example of a linked list construction.

Example 5.5.5:
Linked list construction in Pascal:

Let us construct a linked list in Pascal representing the sequence $(1, 2, \ldots, n)$. The comma "," between two successive integers can be interpreted as "pointers." This list will consist of exactly n nodes. If $n = 0$ then the list is empty. Finally, the program prints the contents of the linked list. If the list is empty, it prints nothing.

Algorithm: We create a node and store n, in the x field of this node. Then another new node is created and $n - 1$ is assigned to the x field of this new node and this node containing $n - 1$ is attached to the node containing n. This process is repeated until we arrive at 0. To print the list, we simply go through the list node by node writing the contents until we arrive at the end of the list. The end of the list is indicated by the pointer nil which can be compared to the back door of the last compartment of a train (don't open it!).

```
program ptr_list;
type pointer = ^node
       node = record
         x:integer;
         next: ^node;
end;
var head, t : pointer;(* head points to
the first node of the list
under construction.*)
begin
  writeln(’enter a nonnegative integer’);
  readln(n);
  head := nil; (* initialize empty list *)
  while n > 0 do
  begin
    new(t);
    (* create a new node to be pointed by t *)
```

```
      t^.x := n; (* store n in the x field of t^*)
      t^.next := head; (* attach the node t^
      as the first node of the list pointed by head *)
      head := t; (* update head *)
      n : = n - 1;
    end; (* list is constructed*)
  (* print the contents of the list *)
  (* if the list is nonempty, then head=t  *)
  while head < > nil do
    begin
      writeln( t^.x, ' ');(* write x field t^ *)
      t := t^.next;
      (* move t one step forward in the list *)
    end;
  end.
```

In *C*, the type node is defined as follows using the keyword “struct”:

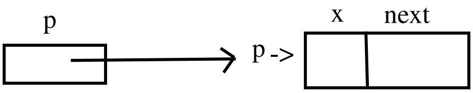
```
struct node \{int  $x$; struct node *next;\};
```

Note the semicolon after the closing brace }. An asterisk * appearing as a prefix to a variable means a pointer variable, that is, next is a pointer variable that will be referencing a variable of type node.

A pointer variable t referencing a variable of type “node” can be defined as follows:

```
    struct node $*t$;\\
```

Now to create a variable of type node to be pointed by the variable t, we use the following somewhat complicated instruction: $t =$ (struct node *) malloc(sizeof $*t$); (see the figure below)



The instruction malloc (**m**emory *alloc*ation) creates a variable of type node and puts a reference to it in the variable *t*. These nodes come from a special region of memory called *heap.* The node just created by invoking the malloc function is denoted by $*t$ or $t- >$ (a minus sign followed by the greater-than sign). The x field of the node $t- >$ is denoted by $t- > x$ and the pointer field by $t- >$next. Finally, the pointer constant which points nowhere is denoted by NULL (equivalent to 0 or $\emptyset$.) Note that several pointers can point to the same node. Note that there are two dereferencing operator notations. A prefix * attached to the pointer variable t and a suffix $- >$ written after the variable t. ($- >$ is the minus sign followed by the symbol greater than.)

In order to clarify these notations, let us now write a program in C to represent the sequence $(1, 2, \ldots, n)$ by a linked allocation. Once the linked list is constructed, the program prints the contents of the list. The algorithm is the same as that of Example 5.5.5.

Example 5.5.6 (Linked list in C:):

```
#include <stdio.h>
struct node {int x; struct node *next;};
main()
{ int n;
  struct node *head, *t;/* head points to the first node
  of the list under construction*/
  printf("enter a nonnegative integer\n");
  scanf("%d",&n);
  head = NULL; /* initialize empty list*/
  while (n > 0)
    {
    t = (struct node *) malloc(sizeof *t); /*create
    a node to be referenced by t*/
    t->x = n; /* assign n to the x field of t->*/
    t->next = head; /*attach the node t->
    at the head of the list pointed by head*/
    head = t; /*update head*/
    n = n - 1;
    }/* list construction is over*/
    /* print the list. t = head if n > 0*/
  while (head != NULL)
    {
    printf("%d", t->x, " ");/* print the x field of t-> */
    t = t->next; /* move t one step forward in the list*/
    }
  }
```

Remark 5.5.1:
A linked list is completely determined by a pointer to the first node of the list.

Advantages of array representation:

Direct access: Access time to the ith element of the array is independent of the index of the array i. In other words, the time needed to attain any given element is a constant which is denoted by O(1).

Calculation on indices: Since the index type of an array is often a subset of the type integer, we can perform calculation on the set of indices of the array.

Disadvantages of array representation:

Fixed size: In Pascal, the size of the array is fixed during the declaration and cannot be changed thereafter. In the language C, however, a programmer can fix the size of an array during execution (dynamic allocation of size). In the array implementation, the programmer has to predict in advance the maximum number of elements that the array will contain at any time during execution and fix the maximum size of the array accordingly. This may lead to wasted memory space.

Insertion and suppression: The operation of inserting a new element in the ith place of the array, takes in the worst case O(n) time, where n is the number of elements currently in the array. This is because, to insert an element in the ith place, we have to move the elements in places $n, n-1, \ldots, i$ one position to the right (by imagining the array is drawn horizontally with the smallest index at the far left), so that the ith cell of the array will be free to receive the new element. Similarly, the operation of performing suppression of the ith element of the array takes O(n) time in the worst case. In the case of suppression of the ith element, we have to move elements in positions $i+1, i+2, \ldots, n$ one position to the left (again we assume that the array is drawn from left to right with the least index at the far left). The following example illustrates the insertion and suppression operation.

Example 5.5.7:
Insertion and suppression in an array:

Consider an array a of size n_max. Assume that n is the number of elements currently in the array a and that the smallest index is 1.

The following procedure in Pascal illustrates the operation of inserting a new element x in the ith place. We suppose the following type declaration:

const n_max = 1000; type vector = array[1..n_max] of integer;

```
procedure insert(i,n: integer; var a : vector);
(* i,n are value parameters. a is the variable parameters.
a is modified by the procedure *)
var j : integer;
begin
  if ( n = n_max )
  then writeln( 'array is full')
  else if ( i > n + 1 ) or ( i < 1)
       then writeln(' position does not exist')
       else
       begin
         (* move integers at  n, n - 1, ...,i one position
         to the right *)
         for  j : = n downto i  do
           a[j + 1] : = a[j];
         n : = n + 1; (* update n*)
         (* insert x *)
         a[i] := x;
       end;
end;
```

Example 5.5.8 (Deleting an element from an array):
The following procedure deletes the integer at position i.

```
procedure suppression( i :integer; var a:vector);
(* i, the value parameter. a, the variable parameter.\\
suppression modifies a*)
var j:integer;
begin
  if (i > n) or (i < 1)
  then
    writeln('position does not exist')
  else
  begin
    n : = n - 1;(* update n*)
    for j:= i to n do
      (* move integers at i+1,i+2,... one position to the left*)
      a[j]:= a[j+1];
  end;
  end;
```

Advantages of linked list representation of a list:

A list is a "living thing," since it is created (birth of the list), then grows, shrinks, and finally destroyed during its lifetime. So in the pointer implementation of the linked list, we need not impose a bound on the number of elements in the sequence. The linked list representation needs only as much space as the number of elements currently on the list, but needs space for the pointer in each node. The operations of insertion and suppression of a node in the list can be performed in a constant amount of time, that is, the time complexity of these operations is O(1).

Insertion in a linked list: Suppose we are interested in adding a new node just *after* the node of the list pointed by a pointer t. Let us suppose the declaration:

```
type pointer = ^node
        node = record
          x:integer;
          next: ^node;
 end;
```

Let p be a pointer to a variable of type node. Then to add a new node (with its x field containing the integer n), just after the node of the list referenced by t, the assignments found in Table 5.11 are performed. (See Figure 5.17.)

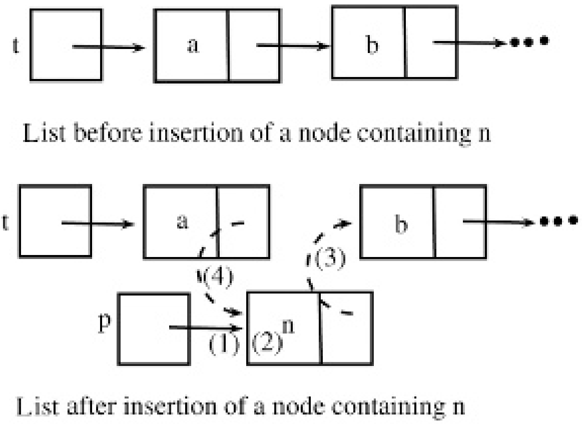
Table 5.11: Linked list insertion

Statements	Effects
(1) new(p);	This instruction creates a node referenced by p.
(2) pˆ : = n;	Assign the integer n to the x field of pˆ.
(3) pˆ. x next := tˆ.next; (4) tˆ.next = p;	Attach the node pˆ just after the node tˆ

Deleting a node from a linked list:

Consider a linked list whose nodes are defined as follows:

```
type pointer = ^node
```



List before insertion of a node containing n

List after insertion of a node containing n

Figure 5.17: Linked list insertion

```
          node = record
            x:integer;
            next: ^node;
   end;
```

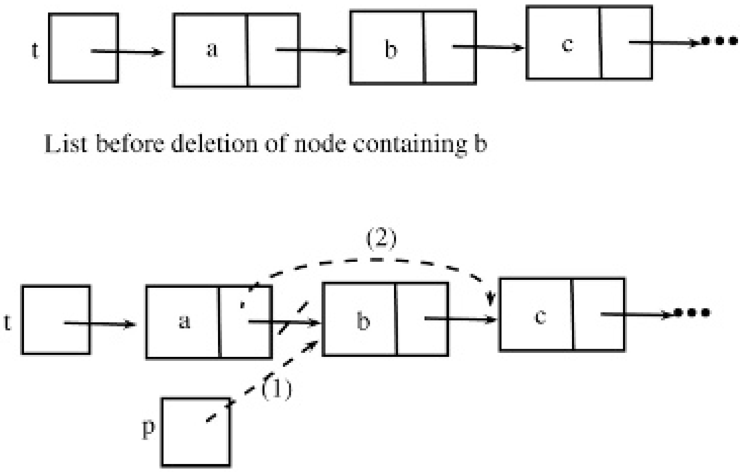
We are interested in deleting the node *after* the node referenced by a pointer t. Note that to suppress a node, we must know a pointer to the node *preceding* it. To do this, we perform the following two assignments of pointers:

p:= t^.next; (here, p is a pointer variable)

t^.next := p^.next; dispose(p);

The predefined/built-in procedure *"dispose"* in Pascal, is the inverse of the procedure "new." The procedure dispose(p) allows the system to claim the memory space occupied by the node referenced by the pointer variable p. The corresponding procedure in C is *"free"* (see Figure 5.18).

Disadvantages of linked list representation: We have no direct access to the ith node of the list. To access the ith node of the list,



List before deletion of node containing b

List after deletion of node containing b

Figure 5.18: Linked list deletion

we have to move from the first node of the list in a step-by-step manner. The complexity of performing such an operation in the worst case is O(n) where n is the number of nodes currently in the list. Also, no manipulation of field names is possible since the field names are strings of characters.

Another disadvantage of the pointer implementation of a list is that it needs *extra* space for the pointer in each node.

What implementation do we choose to represent a list? The answer depends on the algorithm we use to process the list. We have to organize the data in the memory of a computer so that the operations of *storage, retrieval and modification* can be carried out efficiently. If the principal operations of an algorithm on a list are scanning the list, finding the ith element of the list, and swapping the ith and jth elements of the list like in sorting algorithms, we have to use an array implementation.

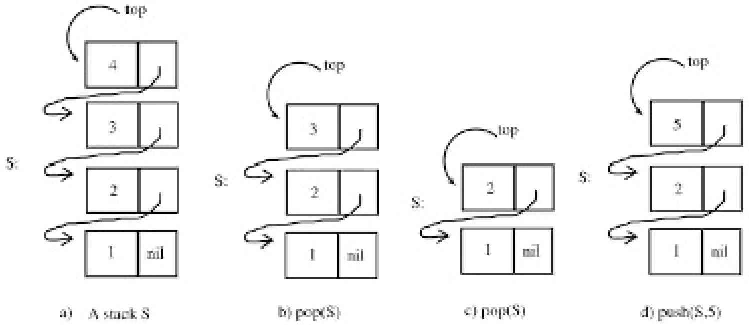
On the other hand, if the main operations of the algorithm on the list are insertions and deletions of elements in arbitrary positions, then the linked list implementation will be more appropriate.

5.5.1 Access Restricted Lists

> Besides the actual universe, I can set in imagination other universes in which the laws are completely different.
>
> *J.L. Synge*

Stacks: A *stack* is a special case of a list in which the operations of insertion and suppression of an element of the list are performed at one end. (If the elements of a list are written from left to right, then the list has two ends, the left end and right end.) The end in which the operations of insertion and deletion are done is called the *top* of the stack. An example of a stack is the set of dishes in a cafeteria, where it is easy to remove the top dish or add a new dish at the top of the pile. A stack is also referred to as a "LIFO" (last-in-first-out) list. The operation of adding a new element at the top is referred to as "push" and the reverse operation of deleting the element at the top of the stack is called "pop." The main application of stacks is in the implementation of recursion in programming languages. To convert a recursive program into a non-recursive one, the programmer has to manage the stack explicitly.

Example 5.5.9 (Stack operations):
Consider the stack $S = (4, 0, 2, 4, 3)$. The stack grows from left to right, that is, the top element of the stack is the rightmost element 3 and the bottom element is the leftmost integer 4. Now the operation of "push(S, 7)" gives the stack $S = (4, 0, 2, 4, 3, 7)$. The operation of "pop(S)" gives the stack $S = (0, 2, 4, 3, 7)$. If we now perform the operation of pop five times successively, we are left with the empty stack $S = ()$. If we now try to apply the command pop(S), this is an error (like taking an element from an empty set).



a) A stack S

b) pop(S)

c) pop(S)

d) push(S,5)

Figure 5.19: Stack operations

Remark 5.5.2:
We use, for the sake of convenience, the top-down notation for stacks, that is, the stacks grow from bottom to top.

Let us write in Pascal the elementary operations associated with a stack of integers, that is, the contents of the stack are integers. We use the linked list representation of stack.

Example 5.5.10 (Elementary stack operations in Pascal using a linked list):
Stack operations in Pascal

```
We assume the following types and variables:
type pointer = ^node
        node = record
           x:  integer;
        next: pointer;
end;
var head : pointer;(* head points to the top
of the stack which is under construction*)
We work with a global stack. Otherwise, the stack must be
passed as a variable parameter.

(* initialize empty stack*)
procedure init_stack;
begin
  head : = nil;
end;
```

```
We now write a function testing if the stack is empty.

function is_stack_empty : boolean;
begin
  if head = nil then is_stack_empty := true
                else is_stack_empty := false;
end;

function stack_top : integer;
(* returns the ‘x’ field of the node at the top*)
begin
  stack_top := head^.x;
end;

Of course the function ‘stack_top’ should be called only if
the stack is nonempty, because the test for nonempty
stack is not found in the function ‘stack_top.’
We now write the ‘push’ and ’pop’ operations.

procedure push( y : integer);
var t: pointer;
begin
  new(t); (* create a new node pointed by t*)
  t^.x = y;(*assign y to the x field of t^*)
  t^.next := head; (*attach node t^
  at the top of the stack*)
  head := t;(*update head*)
end;

function pop: integer;
(*remove node at the top and return the x field
of the removed node*)
var t: pointer; y:integer;
begin
  t := head; y:= t^.x;
  head := t^.next;(* delete the node at the top*)
  dispose(t);
  pop := y;
end;

Of course this procedure should be called only if
the stack is nonempty, because the test for nonempty
stack is not found in the function ‘pop.’
```

Let us write the elementary stack operations in *C* with a linked list. The items of the stack are integers.

Example 5.5.11 (Stack operations in *C* with a linked list):
We use the following definitions of type and variables:

```
struct node
```

```
  {int x; struct node *next; };
struct node *head;

/*initialize empty stack*/
void init_stack ( )
{ head = NULL;
}

int is_stack_empty ( )
{ return head == NULL;
}

int stack_top ( )
{return head->x
}

Of course, the function ‘‘stack_top’’ must be called only when
the stack is nonempty, because the test for nonempty
 stack is not inside the function.

 void push (int y)
 { struct node *t;
   /*create a node to be pointed by t*/
   t = (struct node *) malloc(sizeof *t);
   t->x = y; /*assign y to the x field of t->*/
   t->next = head; /*attach t-> at the top of the stack*/
   head = t;/*update head*/
 }

 int pop ()
 { struct node *t; int y;
   t = head;
   head = t->next;/*delete the node at the top*/
   y = t->x;/*assign x field of t-> to y*/
   free (t);
   return y;
 }
```

The following example illustrates the array implementation of a stack of integers in C.

Example 5.5.12 (Array implementation of stack of integers in C): Let us recall that to represent a stack by an array, a programmer should predict somehow in advance the maximum size to which the stack can grow during the execution of the program.

```
#define max_size = 50 /*stack of size max_size*/
int stack[max_size+1], top;

void init_stack()
{ top = 0;/*bottom most element at stack[1]*/
}

int is_stack_empty ( )
{ return top == 0;
}

void push (int y)
{ stack[++top]=y;
}

Of course, the function should be called only when
the stack is not full, that is, if top is not max_size.
int pop ( )
{ return stack[top--];
}

Of course, this function must be called only when
the stack is non empty,that is, if top is not 0.

int top_stack ()
{return stack[top]
}

This function should be called only when
the stack is non empty, that is,if top is not 0.
```

Data model queue

We shall now study another access restricted data structure called "*queue.*" A common example of a queue is persons waiting at the booking counter in a railway station. A queue is a special kind of list in which the operation of insertion is performed at *one* end of the list called the "end" or "rear" or "tail" of the list, and the operation of deletion is performed at the *other* end of the list, called the "front" or "head" of the list. The elementary operations associated with a queue are the following: Initializing an empty queue, finding the element at the head of the queue, insertion of a new element at the end of the queue (enqueue), deletion of the element at the front of the queue (dequeue), etc. As for stacks, we study two queue implementations: 1. implementation by linked list 2. Implementation by *circular array.*

Implementation of queue operations by linked list:

We draw the elements of the queue horizontally from left to right (as we write) and consider the leftmost element as the head of the queue and the rightmost element as the end of the queue. This is for the sake of convenience of implementation. As we have seen, a stack is completely characterized by a pointer to the node to the top of the stack, whereas a queue can be conveniently represented by two pointers called "head" and "tail." The following example illustrates the basic operations of queues in Pascal using a linked list.

Example 5.5.13 (Queue operations):
Consider the queue $Q = (2, 1, 0, 5, 6)$ of integers. The head of the queue points to the leftmost integer 2 and the tail of Q indicates the integer 6. The operation of "enqueue(Q, 2)" modifies the queue as $Q = (2, 1, 0, 5, 6, 2)$. The operation "dequeue(Q)" gives the queue $Q = (1, 0, 5, 6)$. If the procedure "dequeue(Q)" is invoked successively 5 times, then we will be left with the empty queue $Q = ()$. Now applying the command "dequeue(Q)" will lead to an error message.

Example 5.5.14 (Linked list implementation of queues of integers in Pascal):
We assume the following definition of types and variables:

```
type pointer = ^node
        node = record
           x : integer;
         next: pointer;
end;
var head, tail : pointer;
Let us now write the basic operations of queues:

procedure init_queue;(* initialize empty queue*)
begin
  head := nil;
end;

function is_queue_empty:boolean;
(* test if the queue is empty*)
begin
  if head = nil then is_empty_queue := true
              else is_empty_queue := false;
end;

function front: integer;
(* returns the x field of the node at the front*)
begin
  front := head^.x;
end;
```

```
This function ‘‘front’’ should be called only if
the queue is nonempty.

procedure enqueue(y :integer);
(* enqueue adds a new node with its x field y
at the end of the queue*)
var t :pointer;
begin
  new(t); (* creates a new node*)
  t^.x := y; (* x field of t^ receives y*)
  t^.next := nil;
  if head = nil
  then
  begin
    head := t;
    tail := head;
  end
  else
  tail^.next := t; (*attach t^ at the rear of the queue*)
  tail := t; (* update tail*)
end;

Note that if the queue has exactly one node then
head and tail point to the only node of the queue.

function dequeue: integer;
(* deletes the front node and returns
its x field*)
var t: pointer; y :integer;
begin
  t := head;
  if head = tail (* queue with only one node*)
  then tail := nil;
  head:= t^.next;(* delete node at front*)
  y := t^.x;
  dispose (t);
  dequeue := y;
end;
The above function must be called only
if the queue is non empty, that is, if
the head is not nil. Note also that
in the above implementation of queue commands,
there are two cases in which head and tail may
be equal: when the queue is empty and when
the queue consists of only one node.
```

Circular array representation of queues

A *circular array* is simply a linear array in which we imagine the first element of the array follows the last element of the array. There is an *advantage* in using the circular array instead of the linear array to represent a queue.

Let us clarify this with an example. Consider, for example, a queue (linear but not circular) Q of 10 integers represented by an array a, that is, the head of the queue is $a[1]$ and the tail of the queue is $a[10]$. Let us apply the operation "dequeue(Q)" 6 times repetitively (draw a figure to follow). Then the head of Q points to the index 7 and the tail remains unchanged. Now if we apply the operation "enqueue" once, then the procedure reaches the end of the array (remember the tail points to 10) and leads to the error called "index out of bound" or "queue is full," even though the space is available at the beginning of the array! So in order to make space, with the linear array representation of queues, we have to move/shift the elements of the array regularly to the left and the complexity of this operation in the worst case is $\mathrm{O}(n)$.

In the circular array representation of queues, we need not shift the element regularly to the left of the array.

Let us illustrate the circular array representation by an example.

Example 5.5.15 (Circular array representation of queue): Consider the $Q = (2, 1, 3, 4)$ represented by a circular array a of length 10. The queue grows in a clockwise direction and the integers are stored in consecutive positions in the circular array. We may imagine the four integers stored in places with indices, for example, 5, 6, 7, 8, that is, $a[5] = 2, \ldots, a[8] = 4$. The head of the queue is the index 5 and the tail is 8 (see Figure 5.20).

Now to perform "enqueue(Q, 7)," we simply increment the tail by 1 and assign the integer 7 to $a[tail]$. Now tail $= 9$. Performing another "enqueue(Q, 6)" makes the value of the tail equal to 10. Finally, "enqueue(Q,0)" makes the value of the tail equal to 1, because we imagine the first element follows the last, whose index is 10. To perform the operation "dequeue," we simply increment the index "head" of the queue.

If we apply the operation of "dequeue" 7 times, then we are left with the empty queue and head $= 2$ and tail is 1. This is exactly the same *relative* position as when the queue had 10 elements, that is, as when the queue is full. Hence we cannot distinguish between

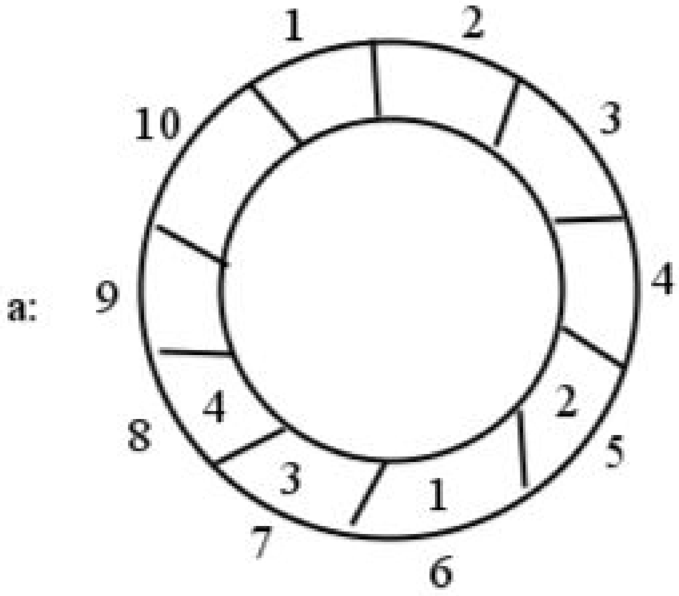


Figure 5.20: A circular list

an empty queue and a full queue. To avoid this problem, we allow the queue to contain at most 9(= 10−1) elements.

From the above example, it is clear that one way to distinguish the full queue from the empty queue is to permit only $n - 1$ elements in the queue where n is the maximum size of the circular array. Let us now write the basic commands of a queue represented by a circular array.

Example 5.5.16 (Basic operations of a queue in Pascal using a circular array):
We assume the following declarations:

```
const n = 100;
var a : array[1..n] of integer;
    head, tail : integer;

(*Even though the capacity of the array
is n, we allow only the maximum of n - 1 elements. This is
to avoid ambiguity in case of empty queue and the full queue.
We work with one global queue. Otherwise, the queue should
be passed as a variable parameter. We now write the basic operations:*)

procedure init_queue;
begin
  head := 1; tail := n;(* initialize empty queue *)
end;

function is_empty_queue : boolean;
begin
  if ((tail mod n) + 1 ) = head
  then is_empty_queue := true
  else is_empty_queue := false;
end;

function is_queue_full : boolean ;
begin
  if ((tail mod n) + 1) mod n + 1 = head
  then is_queue_full := true
  else is_queue_full := false;
end;

function front : integer;
(* returns the integer at the head of the queue*)
begin
  front := a[head];
end;
This function ‘‘front’’ must be invoked only when the queue is non empty.

procedure enqueue( y : integer);
(* add y at the rear of the queue*)
begin
  tail : = tail mod n + 1;
  a[tail] := y;
end;
This procedure ‘‘enqueue’’ must be called only if the
queue is not full.

procedure dequeue;
begin
  head := head mod n + 1;
end;
This procedure must be activated only if the queue is not empty.
```

For other data structures like binary trees, balanced trees, etc., the reader is referred to [3][6][5].

In the following section, we study examples of algorithms with different complexity functions.

5.6 Examples of Algorithms with Different Complexity Functions

$O(n); O \log n); O(n \log n); O(n^2); O(n^3); O(2^n); O(n!)$:

Example of an algorithm which takes $O(n)$ time:

Sequential search in an array:

Input: An array $a[1..n]$ of integers and an integer x to be searched in the array.

Output: i if $a[i] = x$, that is, the integer x is in the ith position of the array. -1 if x is not in the array.

Algorithm: Scan the array by comparing each element of the array against x. If x is found in the ith position, then stop. If the end of the array is reached without finding the element x, then write -1.

The function in C implementing the sequential search algorithm is given in Table 5.12. The complexity of the function in

Table 5.12: Sequential search in an array

```
int seq_search(int a[ ],int l, int r)//l, left index, r, right
{for(int i = 1; i <= n; i++)
    if (a[i] ==x) return i;
//if we reach here, then x not found in the array.
return -1;
}
```

Table 5.12:

In the worst case (case in which x is not in the array or in the last position of the array), we make n comparisons. Hence the complexity is $O(n)$.

Example of an algorithm which takes $O(\log n)$ time:

Divide-and-Conquer Algorithms: Idea: To solve a problem P of size n, we divide the problem P into two similar subproblems P_1 and P_2 of sizes approximately $n/2$ in such a way that the solution of the original problem P is either the solution of the subproblem P_1 or the solution of the subproblem P_2. At each successive step, the size of the subproblem is divided by 2. Hence, the successive sizes of the subproblems are $n/2, n/2^2, n/2^3, \ldots, 1$. Here $n/2$ is taken either as the lower integer part $\lfloor n/2 \rfloor$ or as the upper integer part $\lceil n/2 \rceil$.

The following method, called the "bisection method," is prototypical of the divide-and-conquer algorithm.

Binary search or bisection method in an array:

Input: A sorted array $a[1..n]$ of integers with $a[1] \leq a[2] \leq \cdots \leq a[n]$ and an integer x to be searched in the array.

Output: i if $a[i] = x$, that is, the integer x is in the ith position of the array. -1 if x is not in the array.

Algorithm: Set m=(1+n)/2; Compare the element $a[m]$ against x. If equality holds then write i and stop. Otherwise, we must have either $a[m] < x$ or $a[m] > x$, since the array is sorted. If $a[m] > x$, then try searching x in the first half of the array $a[1..m-1]$. Otherwise we search x in the second half of the array $a[m+1..n]$ and continue till we find the element x or the size of the array searched is negative!

The function in C implementing the binary search algorithm is given in Table 5.13. The complexity of the function in Table 5.13:

Table 5.13: Binary search in an array

```
int bin_search(int a[ ],int l, int r)//l for left index. r for right.
{ int m;
//m, the middle index
while(l <= r)
{
    m=(l+r)/2;
    if (a[m] == x) return m;
    if (a[m] > x) r = m - 1; else l = m + 1;
}
//l > r. if we reach here, x is not found in the array.
return -1;
}
```

Let $T(n)$ be the time required to search x in an array of n elements. If the array consists of one element then we compare x

against the unique element. Otherwise, we find the middle index and restrict our search either in the first half of the array or in the second half of the array. Thus we have the following recurrence relation for $T(n)$.

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n/2) + 1 & \text{if } \ n > 1 \end{cases}$$

In the equation $T(n) = T(n/2)+1$, $T(n/2)$ is the cost of searching in a half array and $+1$ is the cost of halving the array $(m = (l+r)/2.)$

Let us solve the equation in the special case where $n = 2^k$, that is, n is a power of 2.

We have to solve the equation

$$T(2^k) = T(2^{k-1}) + 1$$

Replacing k by $k-1$, we get,

$$T(2^{k-1}) = T(2^{k-2}) + 1$$

Plugging $T(2^{k-1}) = T(2^{k-2}) + 1$ into the right-hand side of the equation $T(2^k) = T(2^{k-1}) + 1$, we get

$$T(2^k) = T(2^{k-2}) + 2$$

Similarly, we can get the next recurrence equation

$$T(2^k) = T(2^{k-3}) + 3$$

Continuing like this, we get finally,

$$T(2^k) = T(2^{k-k}) + k = T(1) + k = 1 + k$$

But $2^k = n$. Hence $k = \log_2 n$.

Therefore, $T(n) = \log_2 n + 1 = O(\log n)$.

Example of an algorithm with complexity $O(n \log n)$.

Divide-and-conquer algorithm:

Consider a problem P of size n. This complexity arises in the following case. We divide the problem P of size n into two similar

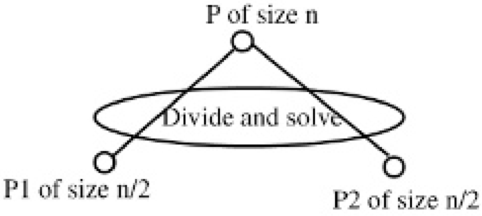


Figure 5.21: Illustration of the divide-and-conquer method

subproblems P_1 and P_2, each of size approximately $n/2$. Then we solve the two subproblems independently and combine the solutions of the subproblems P_1 and P_2 into a solution of the original problem P (see Figure 5.21).

Paradigm of the divide-and-conquer recurrence relation [3]:

We first prove that the following recurrence relation concerning the complexity of divide-and-conquer algorithms. The complexity depends on the following factors:

1. The number of subproblems
2. The size of the subproblems
3. Effort necessary to split the original problem into subproblems

Theorem 5.6.1 (Divide-and-conquer paradigm):
Consider three nonnegative constants a, b, c. Suppose the recurrence relation

$$T(n) = \begin{cases} b & \text{if } n = 1 \\ aT(n/c) + bn & \text{if } \ n > 1 \end{cases}$$

If n is a power of c then the solution to the recurrence relation is

$$T(n) = \begin{cases} O(n) & \text{if } a < c \\ O(n \log n) & \text{if } a = c \\ O(n^{\log_c a}) & \text{if } a > c. \end{cases}$$

Proof. Since n is a power of c, set $n = c^k$ or $k = \log_c n$. Then

$$T(c^k) = aT(c^{k-1}) + bc^k$$

Replace k by $k-1$, we get, $T(c^{k-1}) = aT(c^{k-2}) + bc^{k-1}$. Plugging $T(c^{k-1}) = aT(c^{k-2})+bc^{k-1}$ into the right-hand side of the equation $T(c^k) = aT(c^{k-1}) + bc^k$ we get,

$$T(c^k) = a^2T(c^{k-2}) + abc^{k-1} + bc^k$$

Replacing k by $k-2$ in $T(c^k) = aT(c^{k-1})+bc^k$ and substituting on the right-hand side of the equation $T(c^k) = a^2T(c^{k-2})+abc^{k-1}+bc^k$ we get,

$$T(c^k) = a^3T(c^{k-3}) + a^2bc^{k-2} + abc^{k-1} + bc^k$$

$$\vdots$$

$$T(c^k) = a^kT(c^{k-k}) + a^{k-1}bc + a^{k-2}bc^2 + \cdots + a^{k-k}bc^k$$

$$T(c^k) = bc^k \left((a/c)^k + (a/c)^{k-1} + \cdots + (a/c) + 1\right)$$

The expression inside the bracket is a geometric series. Set the common ratio $r = a/c$.

$$T(n) = bn \left(r^k + r^{k-1} + \cdots + r + 1\right)$$

Case 1. $r = a/c < 1$. Then the geometric series $\sum_{i=0}^{\infty} r^i$ converges to $1/(1-r)$. Hence in this case, $T(n) < bn/(1-r) = O(n)$ (since b and r are constants).

Case 2. $a = c$. Then $a/c = 1$ and $T(n) = bn(k+1) = bn(\log_c n + 1) = O(n \log n)$.

Case 3. $a > c$.

$T(n) = bn\frac{r^{k+1}-1}{r-1}$. Now $r^{k+1} = r \times r^k = r \times (a/c)^k = r \times \left(a^k/c^k\right) = r \times \left(a^k/n\right)$. Plugging in $r^{k+1} = r \times \left(a^k/n\right)$ and simplifying we get $T(n) = b/(r-1) \times \left(ra^k - n\right)$. $T(n) = O(n^{\log_c a})$. Since r, b are constants, $a^k = a^{\log_c n} = n^{log_c a}$, $\log_c a > 1$, and by the rule of sum we can ignore n before $n^{log_c a}$. ■

As an example of an $O(n \log n)$ divide-and-conquer algorithm (in the average case), we study Quick Sort invented by Hoare.

Quick sort or partition sort:
Input: An array $a[1..n]$ of integers.

Output: A permutation of the elements of the array such that

$$a[1] \leq a[2] \leq \cdots \leq a[n]$$

Algorithm: (Quick sort)

Quick sort is based on the following partition method. We choose an element of the array as the pivot element. Following Sedgewick [2], we choose the rightmost element $a[n]$ as the pivot element. Then we split the array satisfying the following three properties:

1. The pivot element is in its final position, that is, had the array been sorted, the pivot element would have occupied the current position after the split.

2. All the elements to the left of the pivot element are less than or equal to the pivot element.

3. All the elements to the right of the pivot elements are greater than or equal to the pivot element.

Assume that the pivot element occupies the kth position after the split. Then we have: $a[1], a[2], \ldots, a[k-1]$ all are $\leq a[k]$ and $a[k+1], a[k+2], \ldots, a[n]$ are all $\geq a[k]$.

Then we apply the partitioning process *recursively* to the elements left of $a[k]$ and to the elements right of $a[k]$ and so on till we arrive at a single element array which is clearly sorted. The following example illustrates the partitioning process.

Example 5.6.1 (Quick sort):
Consider the array $a = (5, 2, 6, 5, 2, 3, 4)$.

The size of the array is $n = 7$. We are going to apply the partitioning process to the array a. Consider as the pivot element the rightmost element $a[7] = 4$. Scan the array from left to right comparing the array elements with the pivot element 4. During the scan, if we come across an integer which is greater than or equal to the pivot element we stop the scan. In the example, the very first element of the array $a[1] = 5$ stops the scan.

Now we scan the array from $(n - 1)$st to the left. If we come across an integer less than or equal to the pivot integer we stop the scan. In the example, the element $a[6] = 3$ stops the scan.

Now we exchange the two elements which stopped the scans, that is, we permute the elements $a[1]$ and $a[6]$ to get the array $a = (3, 2, 6, 5, 2, 5, 4)$.

The scan from the left continues from the second element. Since $2 \not> 4$, we move to the next element 6. But $6 > 4$. Hence the left scan stops at $a[3]$.

The scan from the right continues from the 5th element. Since $2 \not> 4$, the right scan stops at $a[5]$.

We now exchange the elements $a[3]$ and $a[5]$ to get the array $a = (3, 2, 2, 5, 6, 5, 4)$.

We continue the left scan from the 4th element of the array. Since $5 > 4$ the left scan stops at $a[4]$. We continue the right scan from the 4th element. Since $5 > 4$, we move to the third element. But $a[3] = 2 < 4$, and the right scan stops at $a[3]$.

The scans stop now since the left-to-right scan and the right-to-left scan pointers crossed each other (left scan stopped at the index 4 and the right at the index 3) and the condition to stop the scans is that the right index is less than or equal to the left index, which is $3 \leq 4$. Finally, we exchange the element where the left scan stopped with the pivot element to complete the partition. That is, exchange the elements $a[4]$ and $a[7]$ to get the partitioned array $a = (3, 2, 2, 4, 6, 5, 5)$.

In the above array, we see that 4 is in the final position and all elements to the left of $a[4]$ are less than or equal to $a[4] = 4$ and all elements to the right of $a[4]$ are greater than or equal to $a[4]$.

Now we apply the partitioning process to the left array $(3, 2, 2)$ and the right array $(6, 5, 5)$. Note that we have partitioned the initial array of size 7 into two subarrays of size $3 = 7/2$ each and an array consisting of singleton pivot element. We continue the partitioning process till we obtain an array consisting of only one element which is clearly sorted.

Let us write the quick sort algorithm in C. We write a recursive function quicksort in C. The function takes array a as a parameter. It also takes two integer parameters l for the left end of the array (lower limit) and r for thee right end (upper limit) of the array (see Table 5.14). It remains to write the function "split."

Table 5.14: Quick sort algorithm in C

```
void quicksort(int a[ ],int l, int r)
{
int k;
if (r <= l) return;//if array 'a' has fewer than 2 elements, do
nothing
k=split(a,l,r);//call to the function split
quicksort(a,l,k-1);//recursive call to the left subarray
quicksort(a,k+1,r);//recursive call to the right subarray
}
```

Table 5.15: Partition function in C

```
int split(int a[ ],int l, int r)
{int pivot,t,k,j;
//index k scans array a from left to right. j scans from right
to left
   pivot=a[r];k=l-1;j=r;
    do{
      do k++; while(a[k] < pivot);//pivot element acts as
sentinel
      do j–; while(a[j] > pivot)&&(j! = l);
//(j! = l)?:to stop the scan when pivot is the smallest in the
array
      t=a[k];a[k]=a[j];a[j]=t;//
    while (j > k)}//quit loop with j ≤ k.
a[j]=a[k];a[k]=a[r];a[r]=t;// return k; }
```

The function split in C: An additional exchange of $a[k]$ and $a[j]$ is performed when $j < k$ just after the two scan indices j

and k cross but before the crossing is found and the outer most "do loop" is quit. The three assignment statements before the "return k" permutes $a[k]$ and $a[j]$ (to compensate the extra permutation performed) and $a[k]$ with $a[r]$ to put the pivot into the final position.

The complexity of the quick sort algorithm

Best case (the most favorable case): If we are "extremely lucky," then the pivot element will always partition the array in the middle. In this most favorable case, we have the following recurrence relation involving $T(n)$, the time to perform the quick sort.

$$T(1) = 1$$

$$T(n) = 2T(n/2) + n \text{ if } n > 1$$

The recurrence relation $T(n) = 2T(n/2)+n$ says that the problem of size n is split into two subproblems of size $n/2$ (the term $2T(n/2)$ reflects this) with an effort of split n (the term $+n$ reflects this.) In fact, the effort needed to split the array is exactly $n-1$ but $n-1 = O(n)$. This is done for the sake of convenience.

By Theorem 5.6.1, the solution of the recurrence is $T(n) = O(n \log n)$ since $a = c = 2$ and $b = 1$.

Worst case (the most unfavorable case): In the most unfavorable case (for example, if the pivot element happens to be the largest element of the array), the array will be split into two subarrays of size $n-1$ and 0 because the pivot element occupies the last position of the array. In this way, every scan from left to right and right to left fixes only one element. The cost of such a scan is $O(n)$. Hence to fix n elements, the complexity is $T(n) = O(n^2)$.

Average case: The main power of quick sort is that its expected complexity is $O(n \log n)$.

Theorem 5.6.2 (Average complexity of quick sort):
The program of Table 5.14 takes $O(n \log n)$ expected time to sort an array of n integers.

Proof. In the time analysis, we shall use the fact that the `harmonic discrete sum` $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ can be approximated by the integral $\int_1^n \frac{1}{x}\, dx$; that is,

$$\int_1^n \frac{1}{x}\, dx \approx 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

In order to simplify the time analysis, let us assume that all the integers of the array are *distinct*.

Let $T(n)$ be the average time taken by the program 5.14.

If the array consists of fewer than two integers, we do nothing. Hence

$$T(0) = T(1) = 1 \text{ (basis of the recursion)}$$

Now for the recursion step. The call split(l, r) (with $l = 1$ and $r = n$) puts the pivot element in its *final* position k, after comparing all other elements of the array with the pivot. Since the size of the array is n, this scan requires exactly $n - 1$ comparisons but $n - 1 = O(n)$.

The two recursive calls quicksort$(l, k-1)$ and quicksort$(k+1, r)$ (with $l = 1$ and $r = n$) of the program 5.14 have an expected complexity of

$$T(k-1) \text{ and } T(n-k) \text{ respectively}$$

Since the final index k of the pivot element is *equally likely* to take any value between 1 and n, the probability that the pivot occupies the kth index is $1/n$ for each k with $1 \leq k \leq n$. Hence we have the recurrence relationship

$$T(n) = n + \frac{1}{n}\sum_{k=1}^{n}[T(k-1) + T(n-k)], \text{ for } n \geq 2 \text{ (recurrence)}$$

We have to solve the above recurrence relation. First let us note that, $\sum_{k=1}^n T(k-1) = T(0) + T(1) + \cdots + T(n-1)$ and $\sum_{k=1}^n T(n-k) = T(n-1) + T(n-2)) + \cdots + T(0)$. Hence, $\sum_{k=1}^n T(k-1) = \sum_{k=1}^n T(n-k)$. With this algebraic manipulation, the recurrence relation becomes,

$$T(n) = n + \frac{2}{n}\sum_{k=1}^{n} T(k-1)$$

To eliminate the denominator n in the above relation, we multiply both sides by n, to obtain,

$$nT(n) = n^2 + 2\sum_{k=1}^{n} T(k-1)$$

Now to eliminate the sum $\sum$, we replace n by $n-1$ in the above, and subtract the two recurrences: $nT(n)-(n-1)T(n-1) = n^2 - (n-1)^2 + 2[(T(0)+T(1)+\cdot+T(n-1))-(T(0)+T(1)+\cdots+T(n-2))]$ which simplifies to $nT(n) = (n+1)T(n-1)+2n-1$. Divide both sides of the above equation by $n(n+1)$, and we get the recurrence which telescopes:

$$\begin{array}{rcl}
\dfrac{T(n)}{n+1} & = & \dfrac{T(n-1)}{n} + \dfrac{2}{n+1} - \dfrac{1}{n(n+1)} \\
 & = & \dfrac{T(n-1)}{n} + \dfrac{2}{n+1} - \dfrac{1}{n} + \dfrac{1}{n+1} \\
 & = & \dfrac{T(n-1)}{n} + \dfrac{3}{n+1} - \dfrac{1}{n} \\
(n \leftarrow n-1) & = & \dfrac{T(n-2)}{n-1} + \dfrac{3}{n} - \dfrac{1}{n-1} + \dfrac{3}{n+1} - \dfrac{1}{n} \\
 & = & \vdots \\
 & = & \dfrac{T(n-n)}{n-(n-1)} + 3\displaystyle\sum_{k=2}^{n+1}\dfrac{1}{k} - \sum_{k=1}^{n}\dfrac{1}{k} \\
 & \approx & T(0) + 3\displaystyle\int_1^n (1/x)\,dx - \int_1^n (1/x)\,dx \\
(T(0)=1) & \approx & 1 + 3\log n - \log n \\
 & \approx & 1 + 2\log n
\end{array}$$

Hence, $T(n) \approx (1+2\log n)(n+1) = O(n\log n)$ by the sum rule of big O-notation. (Recall that in the O-notation, a “big” term eats away the “small” ones.) ■

Example of an algorithm with complexity $O(n\log n)$. Complexity of the greatest common divisor algorithm:

Let us recall the gcd algorithm:

Table 5.16: gcd function in C.

```
Function in C :
int gcd(int m, int n)
{ if (m%n == 0) return n;//basis
return gcd(n, m%n);//recursion
}
```

Input: Two positive integers m, n with $m > n$.
Output: The gcd of m and n.
Algorithm:

$$\gcd(m,n) = \begin{cases} n & \text{if } m \bmod n = 0 \\ \gcd(n, m \bmod n) & \text{otherwise.} \end{cases}$$

We shall prove that the complexity of the function in Table 5.16 is $O(\log m)$. We shall first prove that after two recursive calls, the size of the problem m is reduced by at least 2. For this, we show that $m\%n \leq m/2$. If $n < m/2$, we have $m\%n < n$ (because the remainder is always ¡ the divisor). If $n > m/2$, the quotient of the division of m by n is $m - n$. But $n > m/2$, hence, $m - n < m/2$. The case $n = m/2$ is not taken into account, because in this case $m\%n = 0$ and hence $\gcd(m \bmod n) = n$ and the program terminates immediately with complexity $O(1)$.

After two calls we have : $\gcd(m,n) = \gcd(n, m\%n) = \gcd(m\%n, n\%(m\%n))$. Since $m\%n < m/2$ (as we have already shown), the original value of m is at least reduced by 2. Therefore we have the following recurrence relation:

$$T(m) \leq \begin{cases} 1 & \text{if } m \bmod n = 0 \text{ basis} \\ 2 + T(m/2) & \text{otherwise.} \end{cases}$$

In the expression $2 + T(m/2)$, the 2 represents the cost of two calls. Let us solve this in the special case where m is a power of 2, say $m = 2^k$. In the worst case, we have the equality instead of inequality in the recurrence. Plugging $m = 2^k$, we get,

$$T(2^k) = 2 + T(2^{k-1})$$

$$\begin{aligned}
\text{changing } k \text{ into } k-1, &= 2+T(2^{k-2}) \\
\text{plugging the second equation into first } T(2^k) &= 2\times 2+T(2^{k-2}) \\
\text{similarly } T(2^k) &= 3\times 2+T(2^{k-2}) \\
&\vdots \\
T(2^k) &= 2k+T(2^{k-k}=1)
\end{aligned}$$

But $T(1) = 1$. Hence $T(m) = 2k + 1$. Since $2^k = m$, we have $k = \log_2 m$. Therefore $T(m) = 2\log_2 m + 1 = O(\log_2 m)$. This is a polynomial time algorithm, because we express the integer m in binary we need only $\log_2 m$ bits not m bits.

Generating prime numbers: (Sieve of Eratosthenes)
Let us write a program in C to generate all prime numbers less than or equal to a given integer (see Chapter 2).

Program to generate all primes $\leq n$. We use an array $a[1:n]$ of integers.

Initialization: We initialize the array as $a[i] = 1$ for all i from 2 to n. This is equivalent (simulation) to writing all the numbers from 2 to n in a list.

Iteration: First we erase all the multiples of 2 from the list, that is, we erase 4,6,... (all even numbers from 4), next we erase all multiples of 3, that is, 6,9,..., then all multiples of 5 (since 4 is already erased, we don't consider its multiples which are already erased) that is, 10,15,..., till we arrive at the middle of the list which is $n/2$. This phase of erasing is simulated by setting to zero the array elements which correspond to indices which are known to be composite.

The complexity of sieve algorithm: The size of the problem is N. Initialization takes $O(N)$, because there is one loop from 2 to $N-1$ and $N-2 = O(N)$. The inner for loop is executed $N/2, N/3, N/4, \ldots, 2$ times respectively. Hence the complexity is

$$\begin{aligned}
T(N) &= N \text{ (for initialization)} + N/2 + N/3 + \cdots + 2 \\
&\leq N + N/2 + N/3 + \cdots + 2 + 1 \\
&\leq N(1 + 1/2 + 1/3 + \cdots + 1/N)
\end{aligned}$$

But $1+1/2+1/3+\cdots+1/N \approx \int_1^N 1/x\,dx = \log_e N$. Hence $T(N) = N log\, N$. In fact by optimizing the program, and using a some-

Table 5.17: Program to generate all primes ≤ n.

```
#include<stdio.h>//reference to input-output library
#define N 10000// constant N
int main()
{
int i,j,a[N];
//initialization of array: list numbers from 2 to n
for(i = 2;i < N;i + +) a[i] = 1;
//iteration
for(i = 2;i < N/2;i + +)
//erase multiples of i
   for(j = 2;j < N/i;j + +) //N × N/i = N
       a[i ∗ j] = 0;//i × j is composite
//write the primes 10 per row
j = 0;//j counter
for(i = 2;i < N;i + +)
{
    if (a[i] == 1)
        {printf(”%5d”, a[i]); j + +;}/*print the prime and
update counter*/
    if (j == 10) printf(”\n”);//new line
}
return 0; }
```

what deep result of number theory, it can be proved that $T(N) = O(N \log(\log N))$ (because $\sum^{N}_{\substack{i=1\\ \text{i, prime}}} 1/i = O(N \log(\log N))$).

Example of an algorithm with complexity $O(n^2)$:

Elementary sorting: (Sorting by selection)

Input: An array $a[1..n]$ of integers.

Output: A permutation of the array elements with the property $a[1] \le a[2] \le a[3] \cdots \le a[n]$.

Algorithm: (Sorting by selection)

Find the smallest of the array $a[1..n]$ and exchange it with the first element $a[1]$.

Find the smallest element in the array $a[2..n]$ and exchange it with $a[2]$, etc.

More generally, find the smallest element in $a[i..n]$ and exchange it with $a[i]$.

Finally, find the smallest element in $a[n-1..n]$ and exchange it with $a[n-1]$. We present the algorithm in pseudo-code:

```
For i from 1 to n-1 do
begin
let $a[k] = \min a[i..n]$
exchange $(a[k], a[i])$
end
```

Let us illustrate the algorithm by an example.

Example 5.6.2 (Selection sort):
Consider the array $a = (7, 8, 6, 2, 5, 2))$ as input. Here $n = 6$. A smallest element in $a[1..6]$ is $a[4] = 2$ and exchange $a[4]$ and $a[1]$ to get the array $a = (2, 8, 6, 7, 5, 2)$. The smallest element in $a[2..6]$ is $a[6] = 2$ and exchange $a[2]$ and $a[6]$ to get the array $a = (2, 2, 6, 7, 5, 8)$. The successive arrays are $a = (2, 2, 5, 7, 6, 8)$, $a = (2, 2, 5, 6, 7, 8)$, $a = (2, 2, 5, 6, 7, 8)$

There are five iterations.

Let us write a function in C implementing the selection sort (see Table 5.18).

Complexity of the selection sort:

Let $T(n)$ be the time to sort n integers by the selection sort function of Table 5.18. The instructions inside the inner loop take $O(1)$ (recall that $O(1)$ represents a constant). There are 3 assignments and each costs a unit of time. Hence the total cost is $3 = O(1)$.

The inner loop is executed $n - i$ times. The external loop is executed $n - 1$ times. Hence,

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=1}^{n-i} 1$$

Table 5.18: Selection sort

```
void selection(int a[ ],int l, int r)//l, left index, r, right
{
int i,j,k,t;//i,j for loops.
for (i = 1; i < n; i + +)
   {//initialization of k
   k=i;
    for (j = i + 1; j ≤ n; j + +)
       if (a[j] < a[k]) k=j;
//exchange a[k] and a[i]
    t=a[k];a[k]=a[i];a[i]=t;}
}
```

$$T(n) = \sum_{i=1}^{n-1}(\overbrace{1+1+\cdots+1}^{(n-i) terms})$$

$$T(n) = \sum_{i=1}^{n-1}(n-i)$$

$$T(n) = (n-1) + (n-2) + \cdots + 1 = \frac{(n-1)n}{2}$$

$$= \text{A polynomial of degree } 2 = O(n^2).$$

Example of an algorithm of complexity $O(n^3)$:

Matrix multiplication:

Input: Two matrices $A_{n\times n} = (a_{ij})$ and $B_{n\times n} = (b_{ij})$ with integer entries.

Output: The product matrix $C_{n\times n} = A \times B$ where $C = (c_{ij})$.

Algorithm: Just use the definition of the product of two matrices.

$$c_{ij} = \sum_{k=1}^{n} a_{ik} * b_{kj} \text{ for all } i, j = 1, 2, \ldots, n.$$

We give a fragment of the program in C (see Table 5.19).

The complexity of the matrix multiplication program:

Table 5.19: Matrix multiplication program

```
int i,j,k,s;
//initialization of C
for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
        c[i][j]=0;
//end of initialization of C
for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
        for (k = 1; k <= n; k++)
            c[i][j]=c[i][j]+a[i][k]*b[k][j];
```

Let $T(n)$ be the cost of multiplying two $n \times n$ matrices. The instruction c[i][j]=c[i][j]+a[i][k]*b[k][j] costs one unit of time.

Hence

$$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} 1$$

$$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} n$$

$$T(n) = \sum_{i=1}^{n} (\overbrace{n + n + \cdots + n}^{n\ terms})$$

$$T(n) = \sum_{i=1}^{n} n^2$$

$$T(n) = (\overbrace{n^2 + n^2 + \cdots + n^2}^{n\ terms}) = n^3 = O(n^3)$$

Example of an algorithm with complexity $O(2^n)$:
(See the excellent book by Sedgewick [2]).

Towers of Brahma-Hanoi puzzle:

The puzzle consists of three pegs A, B, and C. We assume that A, B, and C are arranged in a clockwise direction in the form of a triangle. Initially, peg A has on it some number n of disks, starting with the largest one on the bottom and successively smaller ones on top, as shown in Figure 5.23.

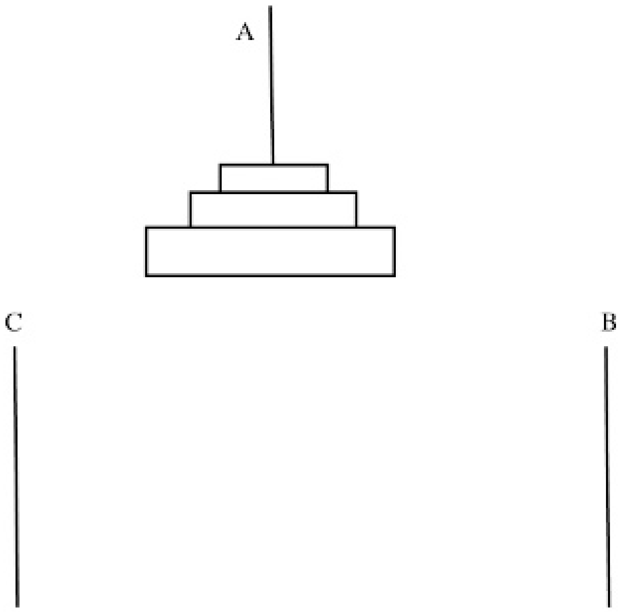


Figure 5.22: Illustration of the Brahma-Hanoi puzzle

The aim of the puzzle is to move the disks one at a time from peg to peg, never placing a larger disk on top of a smaller one, finally arriving with all the disks on peg B. Peg C may be used as an auxiliary peg. This problem is due to the French mathematician Edouard Lucas (1883). The tower of Brahma has 64 disks and the tower of Hanoi has 8 disks. According to a romantic legend by Lucas, Lord Brahma in the beginning of the creation of the universe, placed these 64 golden disks stacked in decreasing size on peg A. Brahma then ordained a group of priests to transfer all the 64 disks from peg A to peg B. When the priest finished the transfer of all the disks, the world will end (see [4]).

Algorithm (divide-and-conquer):
Suppose we know how to transfer $n-1$ disks from one peg to another. Divide the problem of moving n disks into two similar subproblems of size $n-1$. First transfer the $n-1$ smallest disks from peg A to peg C leaving the largest disk on peg A.

Transfer the largest disk from A to B. Then move the $n-1$ smallest disk from C to B.

Although the exact transfer of individual disks is not clear, and hand simulation is difficult because of the successive and deep recursive calls, the algorithm is easy to understand. We can also prove easily that the algorithm works correctly.

Let us write a recursive program in C (see Table 5.20):

Imagine the pegs arranged in the form of a triangle.

Table 5.20: C function for Brahma_Hanoi puzzle

```
void Brahma_Honoi(int n, int d)
{//d for direction. d=+1 clockwise transfer of disk, d=-1 anti-
clockwise move
if (n==0) return;// 0 disks. do nothing
Brahma_Honoi(n-1, -d);//move n − 1 disks from peg A to peg
C
printf("%d ", n*d);//move nth disk from peg A to peg B
Brahma_Honoi(n-1,-d);//move n − 1 disks from peg C to peg B
}
```

A call to Brahma_Hanoi(3,+1) will print the following sequence of moves (n=3,d=+1): (see the tree of Figure 5.23 which illustrates the different calls.

$+1-2+1+3+1-2+1$

A call to Brahma_Hanoi(2,-1) will print the following sequence of moves (n=2,d=-1):

+1 -2 +1

Note that the two disks are transferred to peg C.
A call to Brahma_Hanoi(2,+1) will print the following sequence of moves (n=2,d=+1):

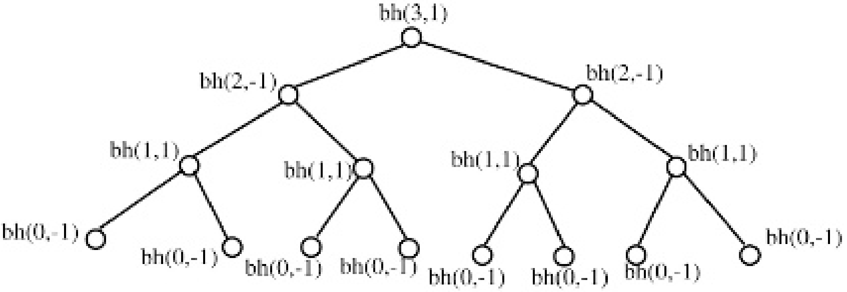


Figure 5.23: Tree illustration of call Brahma_Hanoi (3,1)

-1 +2 -1

Here the two pegs are transferred to peg B.

Interpretation of the output: +1 means move the smallest disk one position clockwise. −2 means move the second smallest disk one position counterclockwise, +3 means transfer the third smallest disk (that is, the largest disk) one position clockwise. Note the pegs A, B, C are arranged in the form of a triangle with A, B, and C in clockwise order. The reader is asked to verify the sequence of moves indicated by the sequence to convince herself that the transfers indeed satisfy the conditions of the puzzle.

By observing this output, we deduce the individual moves of the disks as follows: Complexity of the Brahma_Hanoi puzzle:

Let $T(n)$ be the number of disk moves performed by the algorithm. Then we have the following recurrence relation:

$$T(n) = \begin{cases} 0 & \text{if } n = 0 \\ 2T(n-1)+1 & \text{otherwise.} \end{cases}$$

The recurrence equation $T(n) = 2T(n-1)+1$ reflects the following fact. We convert the problem of size n into two similar subproblems of size $n-1$. The term +1 indicates the effort/labor/cost necessary to divide the problem of size n into two similar subproblems of size $n-1$.

Table 5.21: Recursive algorithm for Brahma_Hanoi puzzle

Algorithm (Revisited): (Iterative method)
We distinguish two cases:
Case 1: n, the number of disks is odd.
On odd-numbered moves, move the smallest disk one peg *clockwise.*
On even-numbered moves, make the only legal move not involving the smallest disk.
Case 2: n, the number of disks is even.
On odd-numbered moves, move the smallest disk one peg *anti-clockwise.*
On even-numbered moves, make the only legal move not involving the smallest disk.

Let us now solve the recurrence relation

$$T(n) = 2T(n-1) + 1$$

Adding 1 on both sides, we get,

$$T(n) + 1 = 2T(n-1) + 2 = 2(T(n-1) + 1)$$

Now set, $S(n) = T(n)+1$. Then $S(n-1) = T(n-1)+1$. Therefore the equation $T(n) + 1 = 2T(n-1) + 2 = 2(T(n-1) + 1)$ becomes

$$S(n) = 2S(n-1).$$

Replacing n with $n-1$ we get,

$$S(n-1) = 2S(n-2).$$

Plugging $S(n-1) = 2S(n-2)$ into the equation $S(n) = 2S(n-1)$ we obtain,

$$S(n) = 2^2 S(n-2)$$

More generally,

$$S(n) = 2^k S(n-k)$$

Setting $k = n$, we obtain

$$S(n) = 2^n S(n-n) = 2^n S(0) = 2^n (T(0) + 1) = 2^n.$$

Hence $T(n) + 1 = 2^n$.

$$T(n) = 2^n - 1 = O(2^n)$$

If $n = 10$, then 1,023 moves are performed. If $n = 20$, then about 1 million moves are performed. If $n = 30$, then about 1 billion moves are done. Even if the priest transfers 1 million disks per second, it will take 5000 centuries to transfer 64 disks! (See the book "Concrete Mathematics" by Graham, Knuth, Pathashnick [4].)

Example of an algorithm of complexity $O(n!)$

Algorithm for permutation generation

Input: A set of n integers $\{1, 2, \ldots, n\}$.
Output: List of all the $n!$ permutations of $\{1, 2, \ldots, n\}$.
Algorithm: The algorithm is recursive.

If $n = 1$, then the *only one* permutation of the singleton $\{1\}$ is 1 (basis of the recursion).

Now for the recursion step: If $n > 1$, then we proceed as follows: n-set permutations will be expressed in terms of $(n-1)$-set permutations. Fix the last integer n, and append before integer n, the list of all the permutations of the first $(n-1)$ integers. For example, if $n = 3$, then we fix the integer 3, and the list of permutations of the first $n - 1 = 2$ integers are 12, , 21. By appending these permutations of $\{1, 2\}$ before 3, we obtain, the list: 123, , 213.

Next, we swap the first and the last integer of the list $(1, 2, \ldots, n)$ to obtain the list $(n, 2, 3, \ldots, n-1, 1)$. As before, fix the last integer of the list 1, and append before 1, the list of all the permutations of $(n-1)$ integers $(n, 2, 3, \ldots, n-1)$ integers. For example, if $n = 3$, by swapping 1 and 3, we get the list $(3, 2, 1)$. Fixing 1 and appending the list of permutations of the list $(3, 2)$, we obtain:

321, 231. We now swap the second and the last element of the *original list* $(1, 2, \ldots, n)$, and proceed as before. The procedure stops after swapping the element $n-1$ and n of the list $(1, 2, \ldots, n)$ and appending before $(n-1)$ all the permutations of $(1, 2, \ldots, (n-2), n)$.

Table 5.22: Permutation generation

```
#include<stdio.h>
#define N 4
int a[N+1]={ 0,1,2,3,4 };
void print()
{ int i;
for ( i=1; i<=N; i++)
    printf(”%3d”, a[i]);
printf("\n\n");
}
void perm(int k)
{ int i,x;
if (k==1)
    print();
else
    {
    perm(k-1);/*recursive call*/
    for (i=1; i < k; i++)
       {x=a[i];a[i]=a[k];a[k]=x;/* swap a[i] and a[k]*/
       perm(k-1);/*recursive call*/
       x=a[i];a[i]=a[k];a[k]=x;/*restore array*/
       }
    }
}
int main()
{ perm(N);
return 0;
}
```

For example, with $n = 3$, we stop after listing the permutations:

132, 321.

We write the complete program for permutation generation in C (see Table 5.22).

Complexity of the permutation generation

Let $C(n)$ be the number of calls of perm(n) to generate all the permutations of the list $(1, 2. \dots, n)$. Then

$$C(n) = \begin{cases} 1 & \text{if } n = 1 \\ nC(n-1) & \text{otherwise} \end{cases}$$

$C(n) = nC(n-1)$ for $n > 1$, since there are n calls to perm(n-1). Let us solve the recurrence, $C(n) = nC(n-1)$. Replace n by $n-1$, we get: $C(n-1) = (n-1)C(n-2)$. Substituting $C(n-1) = (n-1)C(n-2)$ in the recurrence $C(n) = nC(n-1)$, we obtain: $C(n) = n(n-1)C(n-2)$. Continuing like this, the recurrence telescopes to: $C(n) = n(n-1)(n-2)(n-3)\cdots 2C(1)$ Plugging in $C(1) = 1$, we have $C(n) = n!$.

Let $W(n)$ be the time to write all the $n!$ permutations. Since the function print() has a loop which is executed n times to write each permutation, and there are $n!$ permutations, $W(n) = n \times n!$ Hence the time complexity to generate all the permutations on n symbols is $T(n) = C(n) + W(n) = n! + n \times n! \leq 2 \times (n+1)! = O(n+1)!$

5.7 Exercises

1. Execute Euclid's gcd algorithm on the following pairs of integers:

   a) $(35, 25)$ b) $(57, 33)$ c) $(1729, 13)$ d) $(47, 17)$ e) $(100, 1)$ f) (551,1769)

2. Execute the improved version of Euclid's algorithm on the following pairs of integers:

   a) $(1729, 13)$ b) $(17, 47)$ c) $(27, 81)$ d) $(1, 1000)$ e) (551,1769)

3. Execute the extended version of Euclid's algorithm with $m = 1769$ and $n = 551$, that is, find integers a and b such that

$$am + bn = \gcd(m, n).$$

4. Write a program in C to find the least common multiple (lcm) of two positive integers.

5. Using extended version of Euclid's algorithm, write the following rational numbers into partial fractions:

   a) $\frac{1}{45}$ b) $\frac{1}{120}$ c) $\frac{5}{42}$

6. Prove the min-max equality

   $$\min(am{+}bn > 0|a, b\,\text{are integers}) = \max_{d\text{ divides } m, d \text{ divides } n} d.$$

7. Prove by induction on n the following equalities:

   1. $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$.
   2. $1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$.
   3. $1^3 + 2^3 + \cdots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$.

8. True or false: If $P(n)$ is a polynomial of degree k, then $P(n) = \Omega(n^k)$. If true, prove the result. If false, give a counterexample.

9. True or false: If $P(n)$ is a polynomial of degree k, then $P(n) = \Theta(n^k)$. If true, prove the result. If false, give a counterexample.

10. Prove that

    $$\sum_{i=1}^{n} i^k = O(n^{k+1}).$$

11. True or false: Justify your answer.

    $$\sum_{i=1}^{n} i^k = \Omega(n^{k+1}).$$

12. If $P(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$ with $a_k > 0$, then prove that $P(n) = \Omega(n^k)$.

13. Prove that $n! = O(n^n)$.

14. True or false: Justify your answer.

    a) $5n^2 + 8 = O(n)$.

    b) $n^2 \log n = \Theta(n^2)$.

    c) $n^2 \log n = O(n^3)$.

    d) $O(f(n)) - O(f(n)) = 0$.

15. State and prove the sum rule for big Ω notation.

16. State and prove the product rule for big Ω notation.

17. Write a program in C to find the quotient and the remainder of two integers $m \geq 0$ and $n > 0$. Use only the operations of subtraction, addition and comparison of two integers.

    (Hint: Division is a repeated subtraction, the number of subtractions being the quotient.)

18. Write in C the elementary operations of a stack (pop, push, is_empty, is_full) of integers. Assume the stack represented is an array of integers.

19. Write in C the elementary operations of a stack (pop, push, is_empty, is_full) of integers. Assume the stack represented is a linked list.

20. Write in C the elementary operations of a queue (enqueue, dequeue, is_empty, is_full) of integers. Assume the queue represented is an array of integers.

21. Write in C the elementary operations of a queue (enqueue, dequeue, is_empty, is_full) of integers. Assume the queue as a linked list.

22. Write in C the elementary operations of a queue (enqueue, dequeue, is_empty, is_full) of integers. Assume the queue represented as a circular list.

23. Write a program in C to construct a magic square of order n where n is an odd integer. Test your program with $n = 3$ and $n = 5$.

24. True or false: Justify your answer.

    There is no magic square of order 2.

25. True or false: Justify your answer.

    A magic square $M_{3\times 3}$ of order 3 must have the integer 3 in the middle position; that is, $M[2,2] = 3$.

26. Prove that the sum of entries in each row, in each column, and in each of the two diagonals of a magic square of order n is $\frac{n(n^2+1)}{2}$.

27. Write an iterative program in C to find the sum of the harmonic series
    $$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}.$$

28. Write a recursive program in C to find the sum of the harmonic series
    $$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}.$$

29. Consider a matrix $M = (m_{ij})_{n\times n}$ with $m_{ij} = 0$ if $i < j$. That is, M is a lower triangular matrix. Show how to represent the matrix M of n^2 entries as a one-dimensional array a by *not* representing all the entries zero which are above the principal diagonal of the matrix M. In this way, to represent the matrix M, instead of $O(n^2)$ space, we use only $O(\frac{n(n+1)}{2})$ space. Show how to obtain the entry $M[i,j]$ with $i > j$ in the array a.

30. Consider a tridiagonal matrix $M = (m_{ij})_{n\times n}$, that is, $m_{ij} = 0$ if $|i - j| \geq 2$. Show how to represent the matrix M of n^2 entries as another matrix $N_{n\times 3}$ by *not* representing all the entries m_{ij} with $|i - j| \geq 2$. In this way, to represent the matrix M, instead of $O(n^2)$ space, we use only $O(n)$ space. Show how to obtain the entry $M[i,j]$ with $|i - j| \leq 1$ in the new matrix N.

31. Execute the quick sort algorithm on the array $a = (44, 55, 12, 42, 94, 6, 18, 67)$, by taking the rightmost integer of the array as the pivot integer.

32. Execute the quick sort algorithm on the array $a = (44, 55, 12, 42, 94, 6, 18, 67)$, by taking the leftmost integer of the array as the pivot integer.

33. Execute the selection sort algorithm on the array $a = (44, 55, 12, 42, 94, 6, 18, 67)$.

34. Implement the quick sort algorithm by taking the leftmost element of the array as the pivot element.

35. Implement the quick sort algorithm by taking the middle element of the array as the pivot element.

36. Draw the binary trees corresponding each of the different calls of Brahma_Hanoi(3,-1) and Brahma_Hanoi(2,+1). Write the respective output sequences.

37. Prove that the number of moves made by the k-th disk in the Brahma_Honoi tower puzzle is exactly 2^{n-k}, where n is the total number of disks initially placed on peg A.

38. Using the iterative algorithm for the Brahma_Honoi puzzle given in the text (see Table 5.21), write an iterative program in C.

39. Solve the following recurrence relation:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 3T(n/2) + n & \text{if } \ n > 1 \end{cases}$$

Bibliography

[1] D. E. Knuth, The Art of Computer Programming. Volume 1: Fundamental Algorithms, second edition, Addison-Wesley, Reading, MA, 1973; Volume 2, Seminumerical Algorithms, second edition, Addison-Wesley, Reading, MA, 1981; Volume 3: Sorting and Searching, second printing, Addison-Wesley, Reading, MA, 1975.

[2] R. Sedgewick, Algorithms, Addison-Wesley, Reading, MA, 1989; Algorithms in C, Addison-Wesley, Reading, MA,1990; Algorithms in C++, Addison-Wesley, Reading, MA,1998.

[3] A. V. Aho, J.E. Hopcroft, J. D. Ullman, The Design and Analysis of Algorithms, Addison-Wesley, Reading, MA, 1975; Data Structures and Algorithms, Addison-Wesley, Reading, MA, 1983.

[4] R. L. Graham, D. E. Knuth, O. Pathashnik, Concrete Mathematics, Addison-Wesley, Reading, MA, 1988.

[5] N. Wirth, Algorithms + Data Structures = Programs, Prentice-Hall, Inc., New Jersey, 1976; Systematic Programming, An Introduction, Prentice-Hall, Inc., New Jersey, 1973.

[6] D. Beauquier, J. Berstel et Ph. Chrétienne, Éléments d'algorithmique, Masson, 1992.

[7] Alfred V. Aho, Jeffrey D. Ullman, Foundations of Computer Science (Principles of Computer Science series), Addison-Wesley, Reading, MA, February, 1992.

[8] K. Thulasiraman, M. N. S. Swamy, Graphs: Theory and Practice, John-Wiley & Sons, Inc., 1992.

[9] V. Chvatal, Linear Programming, Freeman, 1983.

[10] Ravi Sethi, Programming Languages: Concepts and Constructs, Addison-Wesley, Reading, MA, 1998.

Chapter 6

Introduction to Logic and Probability

> C'est par l'intuition qu'on invente, mais par la logique qu'on démontre.
>
> *Henri Poincaré*

In this brief chapter, we shall study mathematical logic and elementary probability. The first section starts with the concepts of statements, truth assignments to statements, implication (conditional statement), logically equivalent statements (if and only if statements), truth tables, tautology, contradiction, valid arguments, arguments with fallacy (invalid arguments), and proof techniques in mathematics.

In the next section, we study elementary probability theory. We introduce sample space, events, probability, random variables (discrete and continuous), conditional probability, independence of events, expectation, variance, and Bernoulli, binomial, and Poisson distributions [1],[2],[3].

6.1 Introduction

In mathematics and even in real life, we deal with statements (also called propositions or assertions) which may be true or false but not both, like the following:

1. New Delhi is the capital of India.
2. New York is the capital of the United States.
3. $x = 1$ is a solution of the equation $x^2 - 3x + 2 = 0$.
4. What are you doing?
5. $2 \leq 3$.

The first, third, and fifth are true statements but the second one is false. The fourth one is not a proposition at all. So to every statement, we associate either the value true (also noted by 1) or the value false (also denoted by 0). These are called the *truth values* of the statements.

6.2 Algebra of Propositions

Consider the following compound propositions:

1. New Delhi is the capital of India or $2 > 3$.
2. New York is the capital of the United States and Paris is the capital of France.
3. It is not true that $x = 1$ is a solution of the equation $x^2 - 3x + 1 = 0$.

The three propositions are made of subpropositions using the operations or, and, and, not. We associate the value true with the first and third and false with the second. Propositions can be combined by the operation or, denoted by $\vee$ (read meet or disjunction), and, denoted by $\wedge$ (read join or conjunction), and not (denoted by $\neg$) (read not or negation). These operations are similar to the set union $\cup$, set intersection $\cap$, and complement of a set A, denoted by A'. Tables 6.1 and 6.2 are called *truth tables* and define the three operations. p, q represent the propositions and true=1 and false=0. In this chapter, p, q, r denote propositions (elementary or compound).

Table 6.1: Truth table for or, and

p	q	$p \vee q$	$p \wedge q$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Table 6.2: Truth table for not

p	$\neg p$
0	1
1	0

Note that for 2 variables (propositions) p, q, the table has $2^2 = 4$ lines and in general for n variables, the table will have 2^n lines.

A compound proposition is called a *tautology* if its truth value is always true irrespective of the truth values of its constituents. For example, $p \vee \neg p$ is always true, hence it is a tautology. If the propositional expression always takes the truth value false, then it is called a *contradiction*. For example, $p \wedge \neg p$ is a contradiction.

Priority of operators

The operator not ($\neg$) has higher precedence than and ($\wedge$) and ($\wedge$) has higher precedence than or ($\vee$). Therefore, the expression $\neg p \wedge q$ is interpreted as $(\neg p) \wedge q$ but not as $\neg(p \wedge q)$.

Implication

For example, if the signal is green then a car can cross the signal. An implication is also a proposition. An implication is a statement of the form: if p then q. Equivalent forms are:

1. $p \to q$ (read p implies q)

2. q is necessary for p

3. p only if q

4. p is sufficient for q.

p is called the *hypothesis* and q is the *conclusion*. Table 6.3 gives the truth table for an implication. Note that if the hypothesis is false then the implication is always true! One can easily remember this table by interpreting $\rightarrow$ as $\leq$. The *converse* of the statement $p \rightarrow q$ is $q \rightarrow p$. There is no relation between an implication and its converse, in the sense that one can be true and the other false or both can be true/false simultaneously. This is seen by truth Table 6.4.

Table 6.3: Truth table for implication

p	q	$p \rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

Table 6.4: Truth table for implication and its converse

p	q	$p \rightarrow q$	$q \rightarrow p$
0	0	1	1
0	1	1	0
1	0	0	1
1	1	1	1

Two statements p and q are *logically equivalent* if they have identical truth tables, denoted by $p \leftrightarrow q$ or $p \equiv q$, or simply $p = q$. In other words, p and q are logically equivalent if and only if $p \rightarrow q$ and $q \rightarrow p$. Other forms of equivalent propositions are p if and only if q, and a necessary and sufficient condition for p is q. The following example illustrates the logically equal statements (see Table 6.5).

Example 6.2.1 (Logically equivalent propositions: Contrapositive):
As an example, we prove that $p \to q$ is logically equivalent to $\neg q \to \neg p$ (see Table 6.5). Note that the third and sixth columns are identical. Hence the equivalence is established. This equivalence is called a *contrapositive.*

Table 6.5: $p \to q \equiv \neg q \to \neg q$

p	q	$p \to q$	$\neg q$	$\neg p$	$\neg q \to \neg p$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	0	0
1	1	1	0	0	1

Algebra of propositions

As we already remarked, propositions satisfy the same laws as the algebra of sets.

> The laws governing the propositions are exactly the same as the laws of the algebra of sets!

Let us write $p + q$ for $p \vee q$, pq for $p \wedge q$ and p' for $\neg p$. With this notation, we state the laws of the algebra of propositions:

1. $p + p = p, pp = p$ (idempotence)

2. $p + q = q + p, pq = qp$ (commutative laws)

3. $p + (q + r) = (p + q) + r$, $p(qr) = (pq)r$ (associative laws)

4. $p(q + r) = pq + pr$, $p + (qr) = (p + q)(p + r)$ (distributive laws/factorization laws)

5. $p + 1 = 1, p0 = 0$ (absorption laws)

6. $p1 = p$, $p + 0 = p$ (identity laws)

7. $1' = 0$, $0' = 1$, $p + p' = 1$, $pp' = 0$ (complement laws)

8. $(p')' = p$ (involution law)

9. $(p + q)' = p'q'$, $(pq)' = p' + q'$ (DeMorgan's laws).

Note that the only law which is usually not familiar to the reader is the distributive law $p+(qr) = (p+q)(p+r)$, which will be clear to the reader if we write $+$ as $\cup$ (juxtaposition) as $\cap$, and p,q,r as sets P, Q, R.

Valid and invalid arguments

Let $p_1, p_2, \ldots, p_n, q$ be propositions. Then an *argument* is an assertion that the propositions $p_1, p_2, \ldots, p_n$, called the hypothesis, yields the proposition q, called the conclusion. The argument is denoted by

$$p_1, p_2, \ldots, p_n \vdash q.$$

An argument $p_1, p_2, \ldots, p_n \vdash q$ is said to be a *valid argument* if the proposition q is true whenever all the propositions $p_1, p_2, \ldots, p_n$ are simultaneously true. Otherwise the argument is called a *fallacy*. We state the following theorem, which is easy to prove.

Theorem 6.2.1:
An argument $p_1, p_2, \ldots, p_n \vdash q$ is a valid argument if and only if the implication $p_1 \wedge p_2 \wedge \ldots \wedge p_n \to q$ is a tautology.

Example 6.2.2 (Algebra of propositions, valid arguments):
Prove by the algebraic method that the argument $p \to q$, $q \to r \vdash p \to r$ is a valid argument.

Proof: By Theorem 6.2.1, we must show that the proposition $[(p \to q) \wedge (q \to r)] \to p \to r$ is a tautology. To apply laws of algebra easily, let us write $p + q$ for $p \vee q$, pq for $p \wedge q$ and p' for $\neg p$. With this notation, we must prove that $[(p' + q)(q' + r) \to (p' + r) = 1$. Note that we have replaced $(p \to q)$ by $p' + q$ by contrapositiveness, etc. (see Table 6.5). Once again, by applying contrapositiveness, we must prove that

$$[(p' + q)(q' + r)]' + (p' + r) = 1.$$

Now, $[(p'+q)(q'+r)]'+(p'+r)=$ $(p'+q)'+(q'+r)'+p'+r$ (by DeMorgan's laws)$=(p')'q'+(q')'r'+p'+r$ (by DeMorgan's laws)$=pq'+qr'+p'+r$ (by involution law)$=(p'+pq')+(r+qr')$ (commutativity)$=(p'+p)(p'+q')+(r+r')(r+q)$ (factorization law)$=p'+q'+r+q$ (because $p+p'=1=q+q'$ and $1p=p$)$=q'+q+p'+r$ (commutativity)$=1+(p'+r)$=1 (since p+p'=1 and $1+p=1$). Hence the proof.

Example 6.2.3 (Invalid argument):
Prove using the truth table that the argument $p \to q, q \vdash p$ is a fallacy.

Proof. By Theorem 6.2.1, we shall prove that $[[p \to q] \wedge q]] \to p$ is not a tautology (see Table 6.6).

Table 6.6: Example of an invalid argument

p	q	$p \to q$	$[[p \to q] \wedge q]]$	$[[p \to q] \wedge q]] \to p$
0	0	1	0	1
0	1	1	1	0
1	0	0	0	1
1	1	1	1	1

In the last column, which gives the possible values of $[p \to q \wedge q] \to p$, we find one entry 0 in the second line. Hence, the argument is not valid.

Exclusive or

The word "or" is used in two senses. Normally, p or q means either p or q or both, that is, at least one of p and q occurs. The word "or" is also used in the following sense: p or q but not both. For example, Ram will prepare his PhD thesis either in the University of Paris 6 or in the University of Perpignan Via Domitia, is used in the latter sense, that is, "exclusive or" sense. The notation for this is $\oplus$. The following table gives the definition of $p \oplus q$. From the Table 6.7, it is seen that $p \oplus q$ is equal to addition modulo 2 of p and q.

Table 6.7: Truth table for “exclusive or”

p	q	$p \oplus q$=$(p+q)$ mod 2
0	0	0
0	1	1
1	0	1
1	1	0

6.3 Proofs in Mathematics

> An implication whose truth value is “true” is called a theorem.

What is a theorem? An implication whose truth value is true is called a *theorem*. In other words, it is a true statement of the form: if p then q where p and q are statements which are true or false but not both. How do we prove a theorem? One possible answer is the following: We must read and understand the proofs of other mathematicians. Then we can mimic their proof techniques to prove our theorem. In this brief section, we shall see some methods of proof often employed to prove theorems: proof by induction, direct proof, and indirect proof (or proof by contradiction).

Proof by induction

This is a principle of inheritance. If every man/woman transmits some genetic property g to his descendants, then *all* the descendants of a man/woman with this genetic property will have the generic property g. It is equivalent to König’s infinity lemma in graph theory (see Art of Computer Programming, Vol 1, by D. Knuth)[4].

Mathematical induction

Let $S(n)$ be a statement about the integer n. For example, $S(n)$ could be the statement $n(n+1)$ is an even integer. We are interested in proving that the statement $S(n)$ is true for all positive integers. For this, we proceed in two steps:

1. Prove that $S(1)$ is true. This is called the basis of induction.

2. Prove that if $S(1)S(2), \ldots, S(n)$ are all true, then $S(n+1)$ is also true. This is called the induction step and the supposition that $S(1)S(2), \ldots, S(n)$ are all true is called the induction hypothesis.

Example 6.3.1 (Proof by induction):
If n is a positive integer then prove that $n(n+1)(2n+1)$ is divisible by 6.

Proof. Induction basis: We prove the assertion for $n = 1$. If $n = 1$ then the product becomes $1 \times 2 \times 3 = 6$, which trivially proves the basis of the induction.

Induction hypothesis: Suppose $S(n) = n(n+1)(2n+1)$ is divisible by 6. Then we must prove that $S(n+1)$ is also divisible by 6.

Now $S(n+1) = (n+1)(n+1+1)(2(n+1)+1)= (n+1)((n+1)+1)((2n+1)+2)$. We must write this product in such a way to apply the induction hypothesis. By multiplying and simplifying, we have $S(n+1) = n(n+1)(2n+1)+6(n^2+2n+1)=S(n)+6(n^2+2n+1)$. By induction hypothesis, $S(n)$ is divisible by 6. Note the factor 6 in the second term of $S(n+1)$. Hence $S(n+1)$ is also divisible by 6.

Direct proof

To prove the statement, "if p then q," we start from the hypothesis p and then arrive at the conclusion q by a sequence of logical steps.

Example 6.3.2 (Direct proof):
Let us refer to Example 6.3.1. We shall prove the same statement by the direct proof method.

Proof. $n(n+1)(2n+1) = n(n+1)(2(n+2)-3)=2n(n+1)(n+2) - 3n(n+1)$.

But $n(n+1)(n+2)$ is divisible by $3! = 6$ because $n(n+1)(n+2)/3! = \binom{n+2}{3}$ which is an integer. For a similar reason, $3n(n+1)$ is divisible by $3 \times 2! = 6$, because $n(n+1)/2 = \binom{n+1}{2}$, which is an integer. Hence, the difference is also divisible by 6.

Indirect proof

To prove the proposition, "if p then q," we prove its logically equivalent statement: "not q implies not p." Another variation of the indirect proof is the *proof by contradiction.* In this method, we suppose "p is true" and "q is false" and then try to arrive at "p is false" or some sort of absurd result. Since p can't be true and false simultaneously, we are done.

> Reductio ad absurdum (Reduction to absurdity), which Euclid loved so much is one of a mathematician's finest weapons. It is a far finer gambit than any chess gambit: a chess player may offer the sacrifice of a pawn or even a piece, but a mathematician offers the game.
>
> *G. H. Hardy*

EXAMPLE 6.3.3 (Indirect proof):
Let us again refer to Example 6.3.1. We shall prove the same statement by the *indirect proof* method.

Proof. Suppose $S(n) = n(n+1)(2n+1)$ is not a multiple of 6 for all n positive integers. This means that there is a positive integer n_0 for which $S(n_0)$ is not a multiple of 6. Choose an integer n_0 as small as possible with this property. Then by dividing $S(n_0)$ by 6, we get a remainder r with $0 < r < 6$ and a quotient $P_3(n_0)$, a polynomial of degree 3. (Since $S(n_0)$ may be viewed as a polynomial of degree 3 in n_0, and 6 is a constant which is polynomial of degree 0.) Therefore,

$$S(n_0) = 6P_3(n_0) + r$$

Since n_0 is the smallest integer with the stated property, $S(n_0 - 1)$ is divisible by 6 (with remainder zero). Therefore, we can write

$$S(n_0 - 1) = 6P_3(n_0 - 1).$$

Subtracting, $S(n_0) - S(n_0 - 1) = n_0(n_0 + 1)(2n_0 + 1) - 6P_3(n_0) - [(n_0 - 1)(n_0)(2n_0 - 1) - 6P_3(n_0 - 1)] = r$. Simplifying, we get,

$6n_0^2 - 6P_2(n_0) = r$, where $P_2(n_0)$ is the "polynomial of degree 2 in n_0" obtained by simplifying the square bracket term. But then the left-hand side is divisible by 6, but the right-hand side is not, since $0 < r < 6$. This contradiction proves the result.

Example 6.3.4 (Erroneous proof):
Find the fallacy in the following argument: If a divides m and n, then $m = n$ $(m, n, a$ are integers).

Proof: a divides m, hence $m = qa$. a divides n, hence $n = qa$. Therefore we have, $m = n$.

Answer: If Ram and Arjun are married, that doesn't mean that they have the same wife! Still don't see?

6.3.1 Universal and Existential Quantifiers

Consider a set A. Let $p(x)$ be a statement (proposition) concerning an element of the set A. For example, if A is the set of all natural numbers, then $p(x)$ can be "x is a prime number." The truth set of $p(x)$ is denoted by T_p and is defined as

$$T_p = \{\, x \in\ A \mid p(x) \text{ is true} \,\}.$$

For example, if A is the set of natural numbers, and $p(x)$ is "x is prime" then the truth set

$$T_p = \{\, 2, 3, 5, 7, 11, \dots \,\}.$$

$p(x)$ may be an equation or inequality.

The notation $\forall$ stands for "for all" (called a universal quantifier) and $\exists$ denotes "there exists" (existential quantifier). For example, we may write

$$\forall n \in\ \mathbf{N}\ n(n+1) \text{ is } \text{ even}.$$

It is read as for natural number n, $n(n+1)$ is an even integer. More generally, the statement $\forall\ x \quad \in\ A\ p(x)$ is called the *universal statement* and $\exists\ x\ \in\ A\ p(x)$ is an *existential statement.* Note that the statement $p(x)$ can be either true or false.

Example 6.3.5 (Universal and existential statement):
Prove that the statement $\exists\ n\ \in\ \mathbf{N}\ n^2+n+41$ is composite is true.

Solution: We substitute $n = 1, 2, 3\cdots$ in the expression n^2+n+41. The values are respectively, 43, 47, 53 which are all prime numbers. We now study the given expression carefully. To get a composite number by a suitable substitution of n, we should be able to factorize the expression after a suitable substitution. The value we find is $n = 41$, which on substitution gives $41^2+41+41 = 41(41+1+1)$ which is composite.

We may also say (dually) that the expression $\forall n \in\ \mathbf{N}\ n^2+n+41$ is prime is false, and $n = 41$ is a *counter-example* to the universal statement $\forall n \in\ \mathbf{N}\ n^2+n+41$ is prime.

6.4 Probability

In real-life, we sometimes say on observing the sky: It may probably rain. This means roughly that the probability/chance of rain fall is more than 50%, that is, more than 1/2. The probability theory makes such statements precise [1],[2].

Experiments, outcomes, sample space, events

Consider tossing a fair coin. We may get either a head (briefly, h) or tail (briefly, t).(We suppose that the coin does not stand on its side!) Tossing a coin is called an *experiment* and h and t are called *outcomes*. The set $\{\,h, t\,\}$ is called the *sample space* of the experiment. h and t are called the sample points. (If the possibility of a coin standing on its side is allowed, then the sample space will have 3 elements $\{\,h, t\,s\}$, where 's' stands for the side.)

Definition 6.4.1:
In general, the set of all possible results (also called outcomes) of an experiment is called the *sample space* of the experiment. A sample space is made up of sample points. Any subset of a sample space is called an *event*. Hence if a sample space consists of n

outcomes then 2^n events are possible. The null set $\emptyset$, which is a subset of every set, is an event called the impossible event and the sample space itself is called the sure event. A singleton subset of a sample space is an elementary event. Two events E_1 and E_2 are called *mutually exclusive*, if they have no elements in common. An event is said to *occur*, if any one of its sample points occurs.

The following example illustrates the union, intersection, and complement of events.

> The laws governing the algebra of events are the same as the laws of the algebra of sets!

Example 6.4.1 (Union, intersection, complements of events): Consider the experiment of throwing a die and observe the number appearing on the top. The faces of the die are marked with $1, 2, 3, 4, 5, 6$. The sample space $S = \{1, 2, 3, 4, 5, 6\}$, since any one of the numbers from 1 to 6 can appear on the top. Let E_1, E_2, E_3 be three events defined as follows.

E_1 is the event that an odd number appears. E_2 is the event that an even number appears. E_3 is the event that a multiple of 3 appears. Then $E_1 = \{1, 3, 5\}$;$E_2 = \{2, 4, 6\}$;$E_3 = \{3, 6\}$. $E_1 \cup E_3 = \{1, 3, 5, 6\}$ is the event that either an odd number or a multiple of 3 occurs. $E_2 \cap E_3 = \{6\}$ is the event that an even number which is a multiple of 3 occurs. $E_3' = \{1, 2, 4, 5\}$ is the event that a multiple of 3 does not occur. $E_1 \cap E_2 = \emptyset$ is the impossible event, that is, an odd number and an even number cannot occur simultaneously.

Example 6.4.2:
Consider the experiment of tossing a fair coin twice. The sample space of this experiment is $\{hh, ht, th, tt\}$ where hh means two heads appear, ht means the first toss we get a head and on the second toss we get a tail, etc. Let E be the event of obtaining at least one head. Then $E = \{hh, ht, th\}$. The event of getting an equal number of heads and tails is $\{ht, th\}$.

Until now, our sample spaces have consisted only of a finite number of sample points. The following example illustrates a sample space which is infinite.

Example 6.4.3 (Infinite sample space):
Consider the experiment of shooting an arrow until it hits the target. The sample space of this experiment is

$$\{ s, fs, ffs, fffs, \dots \}$$

where s denotes the success (hitting the target), fs denotes a failure (missing the target) immediately followed by a success, ffs means two successive failures followed immediately by a success, etc. The event of getting a success on or before the kth shoot ($k \geq 1$) is $\{ s, fs, ffs, \dots, f^{k-1}s \}$ where f^i means a string of i f's.

Finite sample space and probability model

Unless stated otherwise, we deal with sample spaces which are finite, that is, their number of points/elements is an integer.

We define the finite probability space (probability model) as follows:

Definition 6.4.2 (Probability space):
A probability model (probability space) is simply a sample space with a non-negative real number associated with each point of the sample space with the condition that the sum of all the numbers associated with the points of the sample space is 1. The non-negative number associated with a point is called the *probability* of that point. We denote by $p(s)$, the probability of the point s in S. The probability of an event $E \subset S$ is denoted by $p(E)$ and is the sum of all the probabilities of the points of E, that is, $p(E) = \sum_{einE} p(e)$.

> A probability model distributes the total probability of 1 among its elementary events.

The following examples illustrate the concept.

Example 6.4.4:
Consider the experiment of tossing a coin once. The sample space of this experiment is $S = \{ h, t \}$ where h denotes head and t denotes tail. Then the assignments $p(h) = 1/3$ and $p(t) = 2/3$ make S, a probability space, because the numbers associated are non-negative and their sum is 1.

We may also set (imagine the coin so "designed"): $p(h) = 1/2, p(t) = 1/2$ or $p(h) = 0.23, p(t) = 0.77$ to obtain another probability model. There are infinitely many such assignments possible.

Remark 6.4.1 (perfectly balanced coin and probability model):
There is no perfectly balanced coin (perfectly unbiased coin), otherwise we can't distinguish between the head and the tail! When we associate the probabilities: $p(h) = p, p(t) = 1 - p$ for $p > 0$, we imagine the coin designed in such a way that if the experiment of tossing the coin is repeated a very large number of times, $p(h)$ will tend to p.

Example 6.4.5:
Consider the experiment of throwing a die twice. The sample space S consists of 36 ordered pairs $S = \{ (i, j) | 1 \leq i, j \leq 6 \}$. This corresponds to the Cartesian product of the set consisting of $1, 2, \ldots, 6$ with itself. $|S| = 36$. We denote the ordered pair (i, j) simply as ij. Then the assignments $p(ij) = 1/36$ for all ij in S make S a probability model. Such a space is called an equiprobable space.

If each sample point of a sample space is assigned the *same* probability, then the space is called an *equiprobable space.*

Definition 6.4.3 (Probability model of an equiprobable space):
Consider a finite sample space consisting of n points. If the probability of each point is $1/n$, then S is an equiprobable space. If an event $E \subset S$ consists of k elements, then $p(E) = k/n$. Hence,

$$P(E) = \frac{\text{number of elements in E}}{\text{number of elements in S}}$$

or

$$P(E) = \frac{\text{number of outcomes favorable to E}}{\text{total number of all possible outcomes of the experiment}}$$

As we already quoted, the algebra of probability of events and the algebra of sets are similar. The following table illustrates this analogy. E, E_1, E_2, E_3 denote events in a sample space S. By the Table 6.8, we have to simply attach the symbol of probability p before the set notation to manipulate the probabilities. For example, we have the following relations for a sample space S and its events E_1, E_2, E_3:

Table 6.8: Similarity between probability of events and set algebra

Description	Notation
At least one of E_1 or E_2(E_1orE_2)	$E_{\cup}E_2$
E_1 and E_2	$E_1 \cap E_2$
not E	E'
E_1 but not E_2	$E_1 \setminus E_2$
Neither E_1 nor E_2	$E_1' \cap E_2'$
At least one of E_1, E_2, E_3	$E_1 \cup E_2 \cup E_3$
Exactly one of E_1 and E_2	$(E_1 \cup E_2) \setminus (E_1 \cap E_2)$
All three of E_1, E_2, E_3	$E_1 \cap E_2 \cap E_3$
Impossible event	$\emptyset$
Sure event	S, the sample space.

- $p(\emptyset) = 0$. This corresponds to the number of elements in the empty set is 0.

- $p(S) = 1$. This corresponds to the fact that S is the universal set.

- $p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1 \cap E_2)$. This corresponds to $|A \cup B| = |A| + |B| - |A \cap B|$.

- $p(E_1 \setminus E_2) = p(E_1) - p(E_1 \cap E_2)$. This corresponds to $|A \setminus B| = |A| - |A \cap B|$.

- if $E_1 \subset E_2$ then $p(E_1) \leq p(E_2)$. This corresponds to if $A \subset B$ then $|A| \leq |B|$.

In particular, if two events E_1, E_2 are *mutually exclusive*, that is, the occurrence of one of the events excludes the occurrence of the other, or $E_1 \cap E_2 = \emptyset$, then $p(E_1 \cup E_2) = p(E_1) + p(E_2)$.

Independent events

Two events E_1, E_2 are said to be *independent events* if $p(E_1 \cap E_2) = p(E_1)p(E_2)$. This says that the occurrence of one of the events does not in any way influence the occurrence of the other event. If the events are not independent, then they are called dependent events.

Example 6.4.6 (Independent events):
Consider the experiment that a coin is tossed twice. The sample space is $S = \{ hh, ht, th, tt \}$. Note that we suppose the space is equiprobable unless stated otherwise. The probability of each sample point is 1/4. Consider the events E_1, E_2, E_3 where E_1 is the event that a head appears in the first toss, E_2 is the one with a head in the second toss, E_3 is the one with two heads. Then

$E_1 = \{ hh, ht \}$;$E_2 = \{ hh, th \}$;$E_3 = \{ hh \}$.

Clearly, by the wording of the definitions of E_1 and E_2, E_1 and E_2 are independent. Let us verify this. $p(E_1) = |E_1|/|S| = 2/4 = 1/2$. Similarly $p(E_2) = 1/2$. $p(E_1 \cap E_2) = p(\{ hh \}) = 1/4$. Therefore $p(E_1 \cap E_2) = p(E_1)p(E_2)$.

The events E_1 and E_3 are not independent. Let us verify this fact. $p(E_3) = 1/4$. $p(E_1 \cap E_3) = p(\{ hh \}) = 1/4$.

We see that $p(E_1)p(E_3) = 1/2 \times 1/4 = 1/8 \neq p(E_1 \cap E_3) = 1/4$.

Conditional probability

Consider an experiment and the sample space S associated with this experiment. S need not be equiprobable space. Consider two events E and F. Note that E and F are subsets of S. Suppose $p(F) > 0$. Then the *conditional probability* of the event E relative to the event F or the probability of the occurrence of E once F

has occurred is denoted by $p(E/F)$, and is defined as

$$p(E/F) = p(E \cap F)/p(F).$$

This measures the probability of the event E *with respect to* the reduced subspace F instead of the whole sample space S. If the space is equiprobable ($F \neq \emptyset$) then we have

$$p(E/F) = p(E \cap F)/p(F) = |E \cap F|/|F|.$$

Example 6.4.7 (Conditional probability):
Consider the experiment of tossing a die. Consider two events E and F of the sample space S, defined as: E is the event that an odd number appears, that is, $E = \{1, 3, 5\}$. F is the event that a prime number appears, that is, $F = \{2, 3\}$. Then the conditional probability of E relative to the event F is $p(E/F) = |E \cap F|/|F| = 1/2$ since $E \cap F = \{3\}$.

Note that $p(E) = 3/6 = 1/2$. $p(E/S) = |E \cap S|/|S| = |E|/|S| = 1/2 = p(E)$.

Note that if the events E and F are independent, then we have

$$p(E/F) = p(E \cap F)/p(F) = p(E) \times p(F)/p(F) = p(E),$$

or equivalently,

$$p(F/E) = p(F \cap E)/p(E) = p(F) \times p(E)/p(E) = p(F).$$

These are the definitions of independence in terms of conditional probability. Of course, we assume that the denominator is > 0.

Random variables

A *random variable* is simply a function from a sample space to the set of real numbers. In other words, if we associate to each sample point of a sample space a number, this association is called a random variable. This allows us to transform the sample points of a sample space (which are not necessarily numbers) into numbers.

Even if the sample points are numbers (like in the toss of a die), we associate real numbers to sample points.

Conventionally, a random variable is denoted by the letters X, Y, Z.

Example 6.4.8 (Random variable):
Consider the experiment of tossing a coin twice. The sample space is $S = \{\, hh, ht, th, tt \,\}$. Define a random variable X by associating to each sample point the number of heads in it. Hence, $X(hh) = 2, X(ht) = 1, X(th) = 1, X(tt) = 0$.

Define another random variable Y by assigning to each point the number of tails. Therefore, $Y(hh) = 0, Y(ht) = 1, Y(th) = 1, Y(tt) = 2$.

Define yet another random variable Z by associating to each point 1 if the first appearance is h, and 0 otherwise. Therefore, $Z(hh) = Z(ht) = 1$ and $Z(th) = Z(tt) = 0$.

Example 6.4.9 (Random variable):
Consider throwing a die twice. The sample space S consists of 36 ordered pairs $S = \{\, (i, j) | 1 \leq i, j \leq 6 \,\}$. This corresponds to the Cartesian product of the set consisting of $1, 2, \ldots, 6$ with itself. $|S| = 36$. We denote the ordered pair (i, j) simply as ij. Define a random variable X as $X(ij) = i + j$ for each $ij \in S$.

Define another random variable Y as $Y(ij) = \max(i, j)$.

Probability distribution of a random variable

Let X be a random variable defined on a probability sample space S (not necessarily equiprobable). Let $R(X) = \{\, x_1, x_2, \ldots, x_m \,\}$ be the set of values taken by the random variable. $R(X)$ is called the range of values of the random variable. X induces the probability space on the set $R(X)$ by assigning to each point x_i of $R(X)$, the probability $p(x_i)$ as follows.

$X^{-1}(x_i) =$ the inverse image of x_i under the function X=$\{\, s \in S \mid X(s) = x_i \,\}$. This is the set of all points of the sample point of S for which the value x_i is assigned by the random variable X. Clearly $X^{-1}(x_i)$ is a subset of S and hence it is an event of the sample space. $p(x_i)$ is defined as $p(X^{-1}(x_i)$, that is, the sum of the probabilities of all the points of S for which the image is x_i (see Table 6.9).

Table 6.9: Probability distribution of a random variable

x_1	x_2	$\dots$	x_m
p_1	p_2	$\dots$	p_m

The ordered pairs $(x_1, p_1), (x_2, p_2), \dots, (x_m, p_m)$ given by Table 6.9 is called the *probability distribution of* X. If the space S is equiprobable, then

$$p(X = x_i) = p(x_i) = |X^{-1}(x_i)|/|S|$$

$$= \frac{\text{number of points of } S \text{ whose image is } x_i}{\text{number of points of } S}$$

Example 6.4.10 (Probability distribution of X):
Consider the experiment of tossing a fair coin twice. The sample space is $S = \{\, hh, ht, th, tt \,\}$. Define a random variable X by associating to each sample point the number of heads in it. Hence, $X(hh) = 2, X(ht) = 1, X(th) = 1, X(tt) = 0$. Find the distribution of X.

S is equiprobable, therefore $p(hh) = p(ht) = p(th) = p(tt) = 1/4$. The range of X is $R(X) = \{\, 0, 1, 2 \,\}$. Note that in a set the elements are not repeated and 1 is written only once, even though it appears twice as an image of X. Now to find $p(0)$, we first find the inverse image of 0, that is, $X^{-1}(0)$, which is $\{\, tt \,\}$ and $p(0) = p(\{\, tt \,\}) = 1/4$. To find $p(1)$, let us find first $X^{-1}(1)$, which is $\{\, ht, th \,\}$, and $p(1) = p(\{\, ht, th \,\}) = 2/4 = 1/2$. Similarly, $p(2) = 1/4$. This distribution is given in Table 6.10. Note that the sum of the numbers in the second row is 1 and the numbers are ≥ 0.

Table 6.10: Probability distribution of a random variable

$X = x_i$	0	1	2
$p(X = x_i) = p_i$	1/4	1/2	1/4

Random variable $f(X)$

If X is a random variable defined on a sample space S, and if $f(X)$ is a function of X, for example, $f(X) = X^2 + 1)$ with values in the real number set for any value of its range $R(X)$, then we define **the random variable** $f(X)$ as follows:

$$f(X)(s) = f(X(s)) \ \forall \ s \in S$$

with *the identical corresponding probabilities as the random variable X.*

Example 6.4.11 (Random variable $f(X)$):
For example, let us refer to the previous Example 6.4.10 and now define $f(X) = 2X - 3$. Then the new random variable $f(X)$ takes the values $f(X)(hh) = f(X(hh)) = f(2) = 1; f(X)(ht) = f(X(ht)) = f(1) = -1; f(X)(th) = -1$ and $f(X)(tt) = f(X(tt)) = f(0) = -3$. Note that by definition of $f(X)$, the probabilities are the same as in Example 6.4.10.

Expectation of a random variable

There are two parameters connected with a random variable: *Expectation of a random variable X* (also called the mean or average of X (because it coincides with the average value of a sequence of numbers where the probability is defined as the relative frequency of a number in the sequence) denoted by $E(X)$ or $\mu(X)$ or simply μ and the *standard deviation* of X, denoted by σ_X of σ. σ_X gives us a measure of "dispersion" of the values of the random variable from the mean.

Definition 6.4.4 (Expectation):
Consider Table 6.11 giving the probability distribution of a random variable X. Then the expectation of X, $E(X)$ is defined as the sum of the products of the values of the random variable and its corresponding probability, that is,

Table 6.11: Probability distribution of a random variable

x_1	x_2	$\dots$	x_m
p_1	p_2	$\dots$	p_m

$$E(X) = \sum_{i=1}^{m} x_i p_i.$$

Example 6.4.12 (Expectation):
Consider the probability distribution of a random variable of Table 6.12. Then $E(X) = 0 \times 1/4 + 1 \times 1/2 + 2 \times 1/4 = 1$.

Table 6.12: Probability distribution of a random variable

$X = x_i$	0	1	2
$p(X = x_i) = p_i$	1/4	1/2	1/4

Variance and standard deviation of a random variable

Consider Table 6.13, giving the probability distribution of a random variable X. Then the *variance* of X, denoted by $var(X)$, is defined in terms of expectation as:

$$var(X) = E[(X - E(X))^2] = \sum_{i=1}^{m} (x_i - \mu)^2 p_i = \left(\sum_{i=1}^{m} p_i xi^2 \right) - \mu^2$$

Table 6.13: Probability distribution of a random variable

x_1	x_2	$\dots$	x_m
p_1	p_2	$\dots$	p_m

where $\mu = E(X)$. Here we have used the definition of the new random variable $f(X) = (X - \mu)^2$.

The standard deviation $\sigma = \sqrt{var(X)}$.

EXAMPLE 6.4.13 (Standard deviation of a random variable): Let us refer to the probability distribution of the random variable given by Table 6.14. Then $\mu = E(X) = 0 \times 1/6 + 1 \times 1/6 + 2 \times 1/12 + 3 \times 1/24 + 4 \times 1/8 + 5 \times 5/12 = 73/24$.

Table 6.14: Probability distribution of a random variable

.

x_i	0	1	2	3	4	5
p_i	1/6	1/6	1/12	1/24	1/8	5/12

$var(X) = \left(\sum_{i=1}^{6} xi^2 p_i\right) - \mu^2$, because there are $m = 6$ entries in each row.

$var(X) = 0^2 \times 1/6 + 1^2 \times 1/6 + 2^2 \times 1/12 + 3^2 \times 1/24 + 4^2 \times 1/8 + 5^2 \times 5/12 - (73/24)^2 = 2327/576$.

$\sigma = \sqrt{(2327/576)} \approx 2.008$.

Bernoulli distribution

Consider an experiment (sometimes called a trial) with only two possible outcomes, like tossing a coin (not necessarily an unbiased coin). Let the probability of heads be p (p need not be $1/2$). Then the probability of tails is $q = 1 - p$. Such a distribution is called a *Bernoulli distribution* with parameter p. In terms of a random variable, it can be described by Table 6.15. In Table 6.15, the value 1 for the random variable represents a success and the value

Table 6.15: Bernoulli distribution of a random variable X

x_i	1	0
p_i	p	$q = 1 - p$

0, a failure. Such a random variable is called an *indicator random variable.*

Binomial distribution

An n $(n \geq 1)$ repeated Bernoulli distribution is called the binomial distribution with parameter n and p and is denoted by $B(n, p)$. Its distribution is given by Table 6.16. The second row of Table 6.16 is nothing but the different terms of the binomial expansion of $(q + p)^n$ and hence the name *binomial distribution.*

Table 6.16: Binomial distribution of a random variable X

x_i	0	1	2	$\dots$	k	$\dots$	n
p_i	q^n	$\binom{n}{1}q^{n-1}p$	$\binom{n}{2}q^{n-2}p^2$	$\dots$	$\binom{n}{k}q^{n-k}p^k$	$\dots$	p^n

The probability of getting k successes (that is k heads) in an independent n tossing of a coin is given by the formula:

$$p_k = p(X = k) = \binom{n}{k} q^{n-k} p^k, \text{ for k=0,1,\dots,n}$$

Example 6.4.14 (Binomial distribution):
Ram plays a game in which the probability of his success is 1/4. He repeats the same game 6 times. What is the probability that Ram wins the game (I) exactly twice (II) strictly more than four times (III) at least once.

Solution: This experiment follows the binomial distribution with parameters $n = 6$ and $p = 1/4$. Hence $q = 1 - p = 3/4$. By the binomial distribution,

$$p_k = p(X = k) = \binom{n}{k} q^{n-k} p^k, \text{for k=0,1,. . . ,n.}$$

(I) The probability of having 2 successes $= p_2 = p(X = 2) = \binom{6}{2}(1/4)^2(3/4)^4 \approx 0.29$.

(II) The probability of having > 4 successes $= p(X > 4) = p(X = 5) + p(X = 6) = \binom{6}{5}(1/4)^5(3/4)^1 + (1/4)^6 \approx 0.004$.

(III) The probability of having no successes $= p_0 = p(X = 0) = (3/4)^6$

Hence, the probability of having at least one success $= p(X > 0) = 1 - p(X = 0) = 1 - (3/4)^6 =\approx 0.82$ since $p(E') = 1 - p(E)$.

Poisson distribution

This distribution is an infinite distribution. The general form of the Poisson distribution with parameter m is given by

$$p_k = p(X = k) = \frac{m^k e^{-m}}{k!}, \text{for } k = 0, 1, \ldots, \infty.$$

We can show that if n is large and p is small (with $np = m$), then the binomial distribution ($\binom{n}{k} p^k q^{n-k}$ is difficult to calculate) tends to the Poisson distribution.

Example 6.4.15 (Poisson distribution):
An industry produces spare parts for cars. According to the quality control department, among every 1000 produced, 5 are defective. We check 800 spare parts. What is the probability that among the 800 parts checked, strictly more than 4 are defective?

Solution: Here $n = 800$ (a "large" integer) and $p = 5/1000 = 0.005$ (a small "number"). So we use the Poisson distribution. The parameter $m = np = 800 \times 5/1000 = 4$. We seek the number $p(X > 4) = 1 - p(X \leq 3) = p(X = 0) + p(X = 1) + p(X = 2) + p(X = 3)$(because $p(E') = 1 - p(E)$.)

We calculate each of the four terms using $p(X = k) = \frac{m^k e^{-m}}{k!}$: $p(X = 0) = 4^0 e^{-4}/0! = e^{-4}$. Now use the fact that $e \approx 2.71$.

We may use a calculator to find the values of each of the three other terms $p(X = 1)$, $p(X = 2)$, $p(X = 3)$. Finally, we get $p(x > 4) \approx 0.371$. We leave the details of the calculation to the reader.

EXERCISES:

1. Simplify the logical expression: $(p \vee q) \wedge (p \vee q')$ where p, q are statements.

2. Prove by using the truth table that $[(p \to q) \wedge (q \to r)] \to (p \to r)]$ is a tautology. The truth table will have eight rows. The text proves this using algebraic methods.

3. Prove by algebraic method that $p \vee \neg(p \wedge q)$ is a tautology.

4. Prove that the argument $p \to q, \neg p \vdash \neg q$ is a fallacy.

5. Disprove the following statement by finding a counterexample. For all positive integers n, n^2+n+41 is a prime number.

6. Write the definition of $\lim_{n\to\infty} a_n = l$ and its negation.

7. Write the definition of $\lim_{x\to a} f(x) = l$ and its negation.

8. Write the converse of the following theorem: (Lagrange's theorem): The order of the subgroup of a finite group divides the order of the group.

9. Let us consider the following statement and its proof.

   Proposition: If 5 is a multiple of 3, then 20 is also a multiple of 3.

   Proof of the proposition: is a multiple of 3. Hence we can write : $5 = 3 \times k$ for some integer k. Now $20 = 4 \times 5$. Plugging $5 = 3k$ in the previous equation, we get, $20 = 4 \times 3k = 3 \times (4k)$. Therefore, 20 is also a multiple of 3 (since $3k$ is an integer).

   What is wrong with the above proof? Find the errors, if there are errors. Your conclusion?

10. True or false. Justify your answer. If 250 is a perfect square then 1000 is also a perfect square.

11. Prove that $p \to (q \vee r)$ is logically equivalent to $(p \to q) \vee (p \to r)$ either by algebraic method or by truth table.

12. A pair of unbiased dice is tossed. Find the probability of obtaining a minimum of 2 in one of the dice.

13. Find $p(A/B)$ if $A \cap B = \emptyset$. (assuming $p(B) > 0$).

14. Find $p(A/B)$ if $B \subseteq A$.(assuming $p(B) > 0$).

15. An urn contains 8 items of which three are defective. Three items are drawn from the urn one by one without replacement. Find the probability that all the three items drawn are defective.

16. A fair coin is tossed four times; let X be the random variable associated with the number of heads that appeared. Write the distribution of X and find the expectation and the variance of X.

17. A pair of two fair dice is tossed. Let X be the random variable assigning the maximum of the number that appeared on the two dice. Write the distribution of X and find its expectation and its variance.

18. A university secretary writes n letters to n professors. The n letters are handed over at random to each professor. Find the expectation that each professor receives his/her own letter. (Hint: Use the indicator random variable).

19. Find the expectation and variance of the Poisson distribution with parameter m.

20. Prove that the mean and the variance of the binomial distribution with parameters n and p are np and npq, respectively, where $q = 1 - p$.

Bibliography

[1] W. Feller, An Introduction to Probability Theory and Its Applications, Vol 1, John Wiley, New York, 1968.

[2] W. Feller, An Introduction to Probability Theory and Its Applications, Vol 2, John Wiley, New York, 1968.

[3] S. Lipschutz and M. Lipson, Discrete Mathematics, Tata McGraw-Hill Publishing Company Limited, New Delhi, 2002.

[4] D. E. Knuth, The Art of Computer Programming. Volume 1: Fundamental Algorithms, second edition, Addison- Wesley, Reading, MA, 1973.

Appendices

Answers to Even-Numbered Exercises

Appendix A

Answers to Chapter 1

Section 1.7

Exercise 2:

(i) Let $y \in f(X_1 \cup X_2)$, then there exists an $x \in X_1 \cup X_2$ such that $y = f(x)$. As $x \in X_1 \cup X_2, x \in X_1 \, \text{or} \, x \in X_2$(or both), say, $x \in X_1$. Then $f(x) = y \in f(X_1) \subseteq f(X_1) \cup f(X_2)$ and argue as before.

(ii) Similar to (i).

(iii) Similar to (i).

Exercise 4:

(i) Let $\{X_1, X_2, \dots\}$ be a denumerable sequence of denumerable sets. We want to show that $X = \cup_{i \in \mathbb{N}} X_i$ is denumerable (that is, its elements can be enumerated as a sequence). Set

$$\begin{aligned} X_1 &= \{x_{11}, x_{12}, x_{13}, \dots\}, \\ X_2 &= \{x_{21}, x_{22}, x_{23}, \dots\}, \\ X_3 &= \{x_{31}, x_{32}, x_{33}, \dots\}, \dots, \end{aligned}$$

Then we enumerate the elements of X as a sequence using Cantor's diagonalization process. We take
$X = \{x_{11}; x_{12}, x_{21}; x_{13}, x_{22}; x_{31}; \dots\}$. Then any element

of $\cup_{i\in\mathbb{N}}X_i$ is the n-th term of this sequence for some unique n. Hence X is denumerable.

(ii) Follows from the fact that a subset of a union of denumerable sets cannot be finite.

Exercise 6:

(i) $n \in M_n$ as n is a multiple of n. Hence $\mathbb{N} \subseteq \cup_{n\in\mathbb{N}}M_n$, Trivially, $\cup_{n\in\mathbb{N}}M_n \subseteq \mathbb{N}$.

(ii) Let $p \in M_{n_1} \cap M_{n_2}$. Then p is a multiple of n_1 as well as n_2, and so, p is a multiple of $[n_1, n_2]$, the lcm of n_1 and n_2. Conversely, any multiple of $[n_1, n_2]$ is a multiple of both n_1 and n_2, and hence belongs to $M_{n_1} \cap M_{n_2}$. Thus $M_{n_1} \cap M_{n_2} = M_{[n_1,n_2]}$.

(iii) Let $n \in \mathbb{N}$. Then $n \notin M_{n+1}$ and hence $n \notin \cap_{q\in\mathbb{N}}M_q \Rightarrow \cap_{q\in\mathbb{N}}M_q = \emptyset$.

(iv) Let $n \in \mathbb{N} \setminus \{1\}$. Then n has some prime divisor p and hence $n \in M_p$, Thus $\cup_{p=\text{a prime}}M_p = \mathbb{N} \setminus \{1\}$.

Exercise 8:

(i) On the set of reals, set aRb if $d(a, b) \leq 1$ where d stands for the distance.

(ii) On the set $\mathbb{Z}$ of integers, set aRb if $a - b \geq 0$. Then $\mathbb{R}$ is reflexive, transitive but not symmetric.

Exercise 10: Let $X = \{X_1, X_2, \ldots, X_r\}$ be a collection of r sets, each X_i being finite. Suppose $|X_i| = n_i$. If the sets X_i are pairwise disjoint enumerate the elements of X as $x_{11}, x_{12}, \ldots, x_{1n_1}$; $x_{n_1+1}, x_{n_1+2}, \ldots, x_{n_1+n_2}; \ldots; x_{n_1n_2+\cdots+n_{r-1}}, \ldots, x_{n_1n_2+\cdots+n_r}$, and hence X has cardinality $\alpha = \sum\limits_{i=1}^{r} n_i$, which is finite. If the sets are not pairwise disjoint, then $|X| < \alpha$, and therefore, X is again finite.

Section 1.12

Exercise 2: Take $a \wedge b = inf(a, b)$, and $a \vee b = sup(a, b)$.

Exercise 4:

(i)

$$\begin{aligned}(a \wedge b) \wedge c &= a \wedge (b \wedge c)\,\text{(by associativity)}\\ &= a \wedge (c \wedge b)\,\text{(by commutativity)}\\ &= (c \wedge b) \wedge a\,\text{(again by commutativity)}\end{aligned}$$

NOTE: We can as well denote the LHS by $a \wedge b \wedge c$ without any ambiguity.

(ii) Similar to (i).

Exercise 6: Suppose the lattice is a chain. Then $a \vee (b \wedge c)$ yields, if for instance, $a < b < c, a \vee b = b$, while $a \vee c = c$ and $b \wedge c = b$. Hence $a \vee (b \wedge c) = b = (a \vee b) \wedge (a \vee c)$. By duality, $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ (in fact, both sides are equal to a). Similar verifications apply for other orderings like $a < c < b$, etc.

Exercise 8: Apply Theorem 1.9.0.16.

Exercise 10: D_{10} : Contains neither a diamond nor pentagonal as a sublattice. Therefore distributive and hence modular.
D_{20} : Distributive.
D_{36} : Contains the pentagonal lattice with vertices $3, 6, 12, 36, 9$ in cyclic order and hence not modular and therefore not distributive.
$D_{60:}$ Contains neither a diamond nor a pentagon. Hence the lattice is distributive.

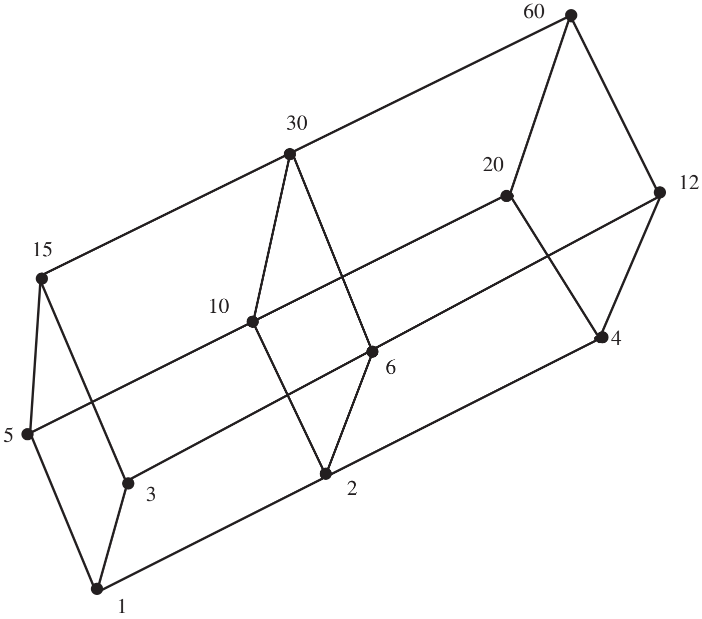


Figure A.1: Lattice D_{60}

Appendix B

Answers to Chapter 2

Section 2.12

Exercise 2: The minimum number is $4 \times 4 + 1 = 17$.

Exercise 4: $0! = 1! = 1$ are the only odd factorials.

Exercise 6: The remainder is 1 and the quotient is $n!/d$.

Exercise 8: The gcd $= 6!$ and the $lcm = 9!$.

Exercise 10: 4^6.

Exercise 12: $26 \times 25 \times 24 \times 23$.

Exercise 14: The number is equal to the (total number of 3-digit numbers) $-$ (total number of 3-digit numbers in which 7 is absent). Note that 0 cannot occupy the hundred's place. Hence the number sought is $9 \times 10 \times 10 - 8 \times 9 \times 9 = 252$.

Exercise 16: The required number is $\binom{n}{1} + \binom{n}{2} + \ddots + \binom{n}{n-1} = 2^n - 2$. Another way: the number of subsets is 2^n. By excluding the null set from B_1 and B_2, we obtain the number $2^n - 2$. ($n \geq 2$.)

Exercise 18: $9 \times 9 \times 8 \times 7 \times 6$. (Note that 0 cannot be in ten thousandth place.)

Exercise 20: 5 boys can be seated in 5! ways. A girl can be seated before all of the boys or at the end of all the boys or in between two boys. Hence the first girl has 6 places to be seated and the second has 5 places and the third has 4 places. Hence the number is $5! \times 6 \times 5 \times 4 = 14400$.

Exercise 22: The number is the same as the number of multisubsets of 18 elements from a set of 4 elements. This number is $\binom{4+18-1}{18}$, which is $\binom{21}{3} = 2660$.

Exercise 24: We transform the problem into an equivalent problem of finding nonnegative integral solutions. To this end, substitute, $y_1 = x_1 - 3$, $y_2 = x_2 - 1$, $y_3 = x_3$, $y_4 = x_4 - 5$. The initial equation becomes, $y_1 + y_2 + y_3 + y_4 = 9$. The number of solutions is the number of multisubsets of 9 elements from a set of 4 elements. This number is $\binom{4+9-1}{9} = \binom{12}{9} = 440$.

Exercise 26: The number of elements in the Cartesian product of a 3-set and a 5-set is $3 \times 5 = 15$. Any subset of this 15-element set gives us a relation and the number of subsets is 2^15.

Exercise 28: The number is $6 \times 5 \times 4 \times 3 = 360$.

Exercise 30: $4! = 24$.

Exercise 32: The number is $\substack{[4\\2]}$ which is the coefficient of x^2 in the product $x(x-1)(x-2)(x-3)$. The coefficient is 11.

Exercise 34: 2143, 2341, 2413, 3142, 3412, 3421, 4123, 4312,4321. They are 9 in number.

Exercise 36: $k = 5$.

Exercise 38: $k = 0$ or $k = n$.

Exercise 40: $4^5 = 1280$.

Exercise 42: The answer is the number of multisets of 5 elements from the set $\{1, 2, 3\}$, which is $\binom{3+5-1}{5} = \binom{7}{5}$=21.

Exercise 44: 1234, 1235, 1236, 1245, 1246, 1256, 1345, 1346 1356, 1456, 2345, 2346, 2356, 2456, 3456.

Exercise 46: $11^5 = (1+10)^5 = 1+\binom{5}{1}10+\binom{5}{2}10^2+\binom{5}{3}10^3+\binom{5}{4}10^4+10^5 = 1 + 50 + 1000 + 10000 + 50000 + 100000 = 161051$.

Exercise 48: Join the midpoints of the opposite sides of the square. We obtain 4 equal squares each of one unit side. There are five points and four squares of side one unit. Hence, two of the points must be in one of the four squares, by the pigeon-hole principle. Since the diagonal of each four square measures $\sqrt{2}$ (by the Pythagorean theorem), the distance between two points must be less then $\sqrt{2}$.

Exercise 50: If not, let S be a subset of $\{a_1, a_2, \ldots, a_{n+1}\}$ with $a_1 < a_2 < \ldots < a_{n+1}$, $a_{i+1} - a_i \geq 2$ for each i, $1 \leq i \leq n+1$. Since $a_1 \geq 1$, $a_{i+1} - a_i \geq 2$, and there are $n+1$ elements in the set S, we must have $a_{n+1} \geq 1 + 2n$, which is impossible.

Exercise 52: We have already seen in the text that any five points inside an equilateral triangle of unit side will ensure that two of the points will be at distance less than 1/2. Similarly, any ten points inside an equilateral triangle of unit side will ensure that two of the points will be at distance less than 1/3 (for this divide each side into 1/3rd of unity and form 9 triangles each of side 1/3). Generalize these. Still don't see the answer? The answer is n^2+1.

Exercise 54: To follow the argument, draw a Venn diagram. Let M, P, C denote the set of students liking (at least) mathematics, (at least), physics, and (at least), chemistry, respectively. By the principle of inclusion-exclusion, the number of students liking at least one of the subjects is,

$$|M \cup P \cup C| = |M| + |P| + |C| - |M \cap P| - |P \cap C| - |C \cap M| + |M \cap P \cap C|$$

which is equal to $15+12+11-9-4-5+3 = 23$. This gives the answer to part (g). Now let us find the number of students liking *only* mathematics and physics. This number is the number of students liking *at least* one of the subjects mathematics or physics minus the number of students liking all the three subjects. This number is $9 - 3 = 6$. Similarly, the number of students liking only physics and chemistry is $4 - 3 = 1$, the number of students liking only mathematics and chemistry is $5 - 3 = 2$. The number of students liking *only* mathematics is the number of students liking mathematics minus the number of students liking only mathematics and physics minus the number liking only chemistry and mathematics minus the number liking all the three $=15 - 6 - 2 - 3 = 4$. Similarly, the number liking only physics is 2 and the number liking only chemistry is 5. Finally, the number liking none of the subjects is the total number of students minus the number liking at least one of the subjects, which is $= 25 - 23 = 2$.

Appendix C

Answers to Chapter 3

Section 3.5

Exercise 2: If prime p divides n, it cannot divide $(n+1)$. Hence $(n, n+1) = 1$. As $(n, n+1) = 1$, $[n, n+1] = n(n+1)$. Indeed, for any two positive integers, if $(p, q) = 1$, then $[p, q] = pq$.

Exercise 4: As $(x, y) = 3$, $3|x$ and $3|y$. Hence if $x + y = 100$, then 3 must divide 100 which is not the case.
Second part: Yes; take $x = 96, y = 3$.

Exercise 6: If $(b, c) \neq 1$, let p be a prime divisor of (b, c). Then $p|b$ and $p|c$, and as $a + b = c$, $p|a \implies (a, c) \neq 1$, a contradiction. Proof for the converse is similar. Result for Fibonacci numbers is now trivial.

Exercise 8:

(i) Dividing the given equation by 11, we get $311x + 11y = 1$. Now $(311, 11) = 1$. So we can express 1 as a linear combination of 311 and 11. We have

$$\text{Divide } 311 \text{ by } 11 : 311 = 28 \cdot 11 + 3$$
$$\text{Divide } 11 \text{ by } 3 : 11 = 3 \cdot 3 + 2$$
$$\text{Divide } 3 \text{ by } 2 : 3 = 1 \cdot 2 + 1$$

$$2 = 1 \cdot 2$$

Hence $(311, 11) = 1$, and therefore, we can express 1 as a linear combination of 311 and 11. Indeed, (by substituting for the successive remainders)

$$\begin{aligned} 1 &= 3 - 2 \\ &= 3 - (11 - 3 \cdot 3) \\ &= 4(311 - 28 \cdot 11) - 11 \\ &= 4 \cdot 3 - 11 \\ &= 4 \cdot 311 - 113 \cdot 11 \end{aligned}$$

Hence $x = 311 \cdot 11 = 3421$ and $y = -113 \cdot 11 = -1243$.

(ii) Similar to (i) Answer : $x = -4730$ and $y = 271$.

Exercise 10: Set $a^m = b$. Then $a^{mn} - 1 = b^n - 1$. Clearly, $b^n - 1$ is divisible by $b - 1$. Therefore $a^{mn} - 1$ is divisible by $a^m - 1$.
Next part: if n were not a prime, then $n = pq$, where p and q are positive integers greater than or equal to 2. So $a^n - 1 = a^{pq} - 1$ is divisible by a^{p-1} and also by $a^q - 1$ and hence it is not a prime, a contradiction.

Exercise 12: Let k be the largest power of 2 that occurs in the denominators, namely, $2, 3, 4 \ldots, n$. Suppose $s = \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{2k} + \cdots + \frac{1}{n}$ is an integer. We shall arrive at a contradiction. Multiply the above equation by 2^{k-1}. This gives $-\frac{1}{2} = -s2^{k-1} + 2^{k-2} + \frac{2^{k-1}}{3} + 2^{k-2} + \frac{2^{k-1}}{5} + \cdots$, where every denominator on the RHS is an odd integer. Hence on simplification, we get a fraction with an odd denominator, a contradiction to the fact that the LHS is equal to $-\frac{1}{2}$. (Note: For the same reason, the odd denominator cannot be equal to 1.)

Exercise 14: Any factor of n is of the form $\prod_{i=1}^{r} p_i^{k_i}$, $0 \leq k_i \leq a_i$. Thus k_i takes $(a_i + 1)$ values. Hence the result.

Section 3.10

Exercise 2: By definition, $\phi(n)$ = number of positive integers less than n and prime to n. Denote them by

$$a_1, a_2, \ldots, a_{\phi(n)} \ldots (i)$$

Now, as a_i is prime to n, and $n - a_i$ is also prime to n. Hence the numbers

$$n - a_1, n - a_2, \ldots, n - a_{\phi(n)} \ldots (ii)$$

are just the same as the numbers in (i). Thus

$$\begin{aligned}\sum_{i=1}^{\phi(n)} a_i &= \frac{1}{2}\Big[\sum_{i=1}^{\phi(n)} a_i + \sum_{i=1}^{\phi(n)} (n - a_i)\Big]\\ &= \frac{1}{2}\phi(n)n.\end{aligned}$$

Exercise 4: By Euler's Theorem, as $(p, q) = 1, p^{\phi(q)} \equiv 1(mod\, q)$, that is, $p^{q-1} \equiv 1(mod\, q)$, as q is a prime. Similarly, $q^{p-1} \equiv 1(mod\, p)$. Therefore

$$\begin{aligned}(p^{q-1} - 1)(q^{p-1} - 1) &\equiv 0(mod\ pq)\\ \implies p^{q-1}q^{p-1} - (p^{q-1} + q^{p-1}) + 1 &\equiv 0(mod\ pq)\\ \implies p^{q-1} + q^{p-1} &\equiv 1(mod\ pq).\end{aligned}$$

Exercise 6: As $(5, 6) = 1$, the congruence has exactly one solution. It is $x = 4$.

Exercise 8: As $(5, 715) = 5$, the congruence has 5 solutions. We apply Theorem 3.8.11. Dividing by 5, we get the congruence, $x \equiv 2\,(mod\ 143)$ and this gives the unique solution $x_0 = 145$. Here $m = 715, d = 5$. Hence the 5 solutions are $x_0 = 145, x_0 + \frac{m}{d} = 145 + 143 = 248, x_0 + 2\frac{m}{d} = 145 + 286 = 431, x_0 + 3\frac{m}{d} = 145 + 429 = 574$, and $x_4 = x_0 + 4\frac{m}{d} = 145 + 572 = 717$.

Exercise 10: Suppose n were not a prime. Then n has a prime divisor p (where $1 < p < n$). By assumption, $(n-1)! \equiv -1\,(mod\ n)$ and hence $(n-1)! \equiv -1\,(mod\ p)$. Also $(n-1)! \equiv 0\,(mod\ p) \implies -1 \equiv 0\,(mod\ p) \implies p|(p-1)$, a contradiction. Hence n must be a prime.

Section 3.12

Exercise 2:

(a) See Theorem 3.11.6 for the first part. If $(m, n) \neq 1, \phi(mn)$ need not be equal to $\phi(m)\phi(n)$. For instance, take $m = n = 2$. Then $\phi(mn) = 2$ while $\phi(m) = 1 = \phi(n)$.

(b) If $(m, n) = 1$, the prime factors of m and n are distinct. If m is a product of r distinct primes, and n is a product of s distinct primes, then mn is a product of $r+s$ distinct primes. Hence $\mu(mn) = (-1)^{r+s} = (-1)^r(-1)^s = \mu(m)\mu(n)$. Case when $(m, n) \neq 1$. For example, take $m = 5, n = 15$. Then $\mu(m) = -1, \mu(n) = 1$ while $\mu(mn) = 0$.

(c) In the sum $\sum\limits_{d|n} \mu(d)\phi(d)$, it suffices to take only square-free prime divisors r and 1 (since $\mu(p^r) = 0$ for $r \geq 2$). Let $p_1 < \cdots < p_r$ be the distinct prime divisors of n. Then

$$\begin{aligned}\sum_{d|n} \mu(d)\phi(d) &= \mu(1)\phi(1) + \sum_i \mu(p_i)\phi(p_i)\\ &\quad + \sum_{i<j} \mu(p_ip_j)\phi(p_ip_j) + \cdots\\ &= 1 - \sum_i \phi(p_i) + \sum_{i<j} \phi(p_ip_j) - \cdots\\ &= \prod_{i=1}^{n} [1 - \phi(p_i)].\end{aligned}$$

But as n is even, $p_1 = 2$, and $\phi(p_i) = 1$. Therefore the product on the right is 0.

Appendix D

Answers to Chapter 4

Section 4.1

Exercise 2: Consider a longest directed path P in G. If x_0 is the origin of the path P, then the in-degree of x_0 is 0, otherwise either the graph will have a path of length greater than $|P|$ or the graph will contain a directed circuit which is impossible. Similarly, the terminus of the path will have out-degree 0.

Exercise 4: Consider a longest path $P = (x_0, x_1, \dots, x_l)$ in G. The length of P is l. Suppose that $\max(\delta^-, \delta^+) = \delta^+ = p > 0$. Since P is a longest path, there can't be a vertex $y \neq x_i$ for $i = 0, 1, 2, \dots, l$ in G with an arc (x_l, y). Otherwise, $P' = (x_0, x_1, \dots, x_l, y)$ will be a path of length $l + 1$. Hence all the successors of the terminal vertex x_l of the path P must be a subset of $\{\, x_{l-1}, x_{l-2}, \dots, x_0 \,\}$. Since the out-degree of x_l is at least p, (x_l, x_k) must be an arc for some $k \leq l - p$ where x_k is a vertex of the path P. This implies that $(x_k, x_{k+1}, \dots, x_l, x_k)$ is a cycle of length $l - k + 1$ which is $\geq p + 1$. The argument is similar if $\max(\delta^-, \delta^+) = \delta^- = p > 0$.

Exercise 6: The number of edges of G is on the one hand $k|X_1|$ and on the other hand it is $k|X_2|$, since G is bipartite. Hence, $k|X_1| = k|X_2|$ which implies $|X_1| = |X_2|$.

Exercise 8: The arc set $U = \{\, (1,3), (3,1)(2,2), (3,4), (4,3) \,\}$.

There are two connected components and their vertex sets are: $\{1,3,4\}$ and $\{2\}$.

Exercise 10: No such graph exists because the sum of the out degrees is 5 whereas the sum of the in degrees is 6.

Exercise 12: In any graph, the sum of the degrees of the vertices is always even. To prove the other part, form the sequences $(r_1, r_2, \dots, r_n)$ and $(q_1, q_2, \dots, q_n)$ where q_i is the quotient of the division of d_i by 2 and r_i is the remainder of the division of d_i by 2. Since the sum of all the d_i's is even, the sequence $(r_1, r_2, \dots, r_n)$ will have an even number of ones, say, $r_{i_1}, r_{i_2}, \dots, r_{i_{2k}}$. We shall now construct a multigraph with the given sequence (d_i). Take n vertices $1, 2, \dots, n$. Join r_{i_1}, r_{i_2} by an edge, r_{i_3}, r_{i_4} by an edge and finally, join $r_{i_{2k-1}}, r_{i_2k}$ by an edge. Now attach to each vertex i, q_i loops. Take an example of a sequence and execute the algorithm to convince yourself.

Exercise 14: The number is $2^{\binom{4}{2}} = 2^6 = 64$.

Exercise 16: Hint: Label the Petersen graph and the other isomorphic graphs of Petersen. Redraw each of the other graphs so as to resemble the Petersen graph. Once this is done, an isomorphism will be evident.

Exercise 18: C_n itself.

Exercise 20: Both are true.

Exercise 22: The chromatic index of $K_4 = \Delta(K_4) = 3$ and that of $K_5 = \Delta(K_5) + 1 = 5$. More generally, $K_{2n} = \Delta(K_{2n}) = 2n - 1$ and $K_{2n+1} = \Delta(K_{2n+1}) + 1 = 2n + 1$.

Exercise 24: We distinguish two cases: $p = 3$. Take an elementary cycle C_6 of length 6, $(1,2,3,4,5,6,1)$ where successive vertices in the sequence are joined by an edge. Add the edges 25, 36, 41. We obtain a cubic graph with 6 vertices and no triangle. The graph obtained is nothing but the complete bipartite graph $K_{3,3}$. Now let $p \geq 4$. Take two copies of elementary cycles C_6, $(1, 2, \ldots, p, 1)$, and $(1', 2', \ldots, p', 1')$ where successive vertices in the sequence are joined in each copy. Now add the p edges $11', 22', \ldots, pp')$. This gives us a 3-regular graph without a triangle.

Exercise 26: The square of the Petersen graph is the complete graph K_{10}, because the diameter of the Petersen graph is 2.
If G is of diameter d, then G^d is the complete graph on n vertices where n is the number of vertices of G.
The distance $d(x, y)$ in G^k is $\lceil n/k \rceil$.

Exercise 28: a) The graph consists of only one vertex, no edges are self-complementary. b) An elementary path of length (consisting of 4 vertices and 3 successive edges) is self-complementary. The elementary cycle of length 5, C_5 is self-complementary. c) Note that $m(G) + m(G^c = m(K_n) = n(n-1)/2$. Since G is self-complementary, $m(G) = m(G^c)$. Hence, the number of edges of G is $n(n-1)/4$. This implies that $n(n-1)$ is a multiple of 4. This means that either n is a multiple of 4 or else $n-1$ is a multiple of 4. d) This follows by definition of the complement of a graph. e) We shall show that if G is disconnected, then G^c is connected. Let the components of G be $G_1, G_2, \ldots, G_k$ with $k \geq 2$. Let the number of vertices of G_i be n_1 for $1 = 1, 2, \ldots, k$. Then, the complete k partite graph $K_{n_1, n_2, \ldots, n_k}$ (which is connected) is a spanning subgraph of G^c. Hence G^c is connected. f) We shall show that if G of diameter > 3, then G^c is of diameter < 3. Take two vertices x and y in G. If xy is not an edge, then xy is an edge in G^c and $d(x, y) = 1$ in G^c. Suppose xy is an edge in G. We shall show that $d(x, y)$ in G^c is 2. Since G is of diameter ¿3, there must be a vertex z which is joined to neither x nor y in G. But then in G^c, every vertex $z \not x, y$ is joined to either x or y. This means that the distance between x and y is 2. Therefore, the diameter of G^c is 2. If G is totally disconnected, then G^c is complete and its diameter is 1. g) This follows the equation $d_G(x) + d_{G^c}(x) = n - 1$ for every vertex x and hence $d_{G^c}(x) = n - 1 - d_G(x)$. h) A set of vertices S is a stable set in G if and only if S is a clique in G^c and S is a clique in G if and only if S is a stable set in G. Hence by definition of the ramsey numbers, we have $r(p,q) = r(q,p$. i) $r(2,q) = q$. j) No. $(K_2 + K_2)^c \neq K_2^c + K_2^c$ and $(K_2 \times K_2)^c \neq K_2^c \times K_2^c$. h). Draw the edge graph of K_5 and then redraw the edge graph so as to resemble the Petersen graph.

Exercise 30: We shall prove a stronger result the diameter of the graph is less than or equal to 2. Consider any two distinct vertices x, y. We have to prove that $d(x,y) \leq 2$. If xy is an edge, then $d(x,y) = 1$. Suppose xy is not an edge of G. We claim that there is a vertex z which is joined to both x and y. If not, $\Gamma(x) \cap \Gamma(y) = \emptyset$. Since, $\delta \geq (n-1)/2$, we have the inequality $d(x)+d(y)+1+1 \geq n$, that is, $n+1 \geq n$, which is impossible. Hence there is a vertex z and both xz and zy are edges. This implies that $d(x,y) = 2$.

Exercise 32: (Draw graphs at each stage to follow the argument.) Cube 1 corresponds to the multigraph graph with four vertices R, B, G, Y with edges RR, RG, BY, cube 2 to the graph with four vertices R, B, G, Y with edges RB, RY, GY, cube 3 to the graph with four vertices R, B, G, Y with edges RB, BY, GG, and cube 4 to the graph with four vertices R, B, G, Y with edges RG, GY, YB. Now label the edges of cube 1 by 1, cube 2 by 2, cube 3 by 3, and cube 4 by 4. By superimposing these four graphs, we obtain a multigraph graph with four vertices R, B, G, Y with edges labeled $RR(1), GG(3), RG(1)$, $RG(1)$, $RB(2)$, $RB(3)$, $BY(1)$, $BY(3)$, $BY(4)$, $YG(2)$, $YG(4)$, $RY(2)$. The labels of the edges are indicated within parentheses. The superimposed graph has two edge disjoint spanning subgraphs G_1 with edge set $RB(3), BY(1)$, $YG(2), GR(4)$ (corresponding to the front and back side) and G_2 with edge set $RB(2), BY(3)$, $YG(4), GR(1)$ (corresponding to the left and right side). Now transform these two graphs G_1 and G_2 into a solution so that all four colors appear on each side of the 4×1 stack.

Exercise 34: Draw two five cycles $(1,2,3,4,5,1)$ and $(1', 2', 3'$, $4', 5', 1')$ where successive vertices of the sequences are joined by an edge. Now draw the edges $11', 22', 33', 44', 55'$. (Draw the graph.)

Exercise 36: a) Q_2 is the elementary cycle of length 4. To get Q_3, take two copies of Q_2 and join the corresponding vertices in each copy by a new edge. This graph is the usual 3-dimensional cube. (Draw the graphs.) b) The number of vertices of Q_k is 2^k. To find the number of edges, observe that a Q_k is a k-regular graph. Hence the sum of the degrees of the vertices is $k2^k$ which is $2m$. Hence the number of edges is $k2^{k-1}$. c) The proof is induction on k. If $k = 1$, then $Q_1 = K_2$ is clearly bipartite. Assume Q_k is bipartite for $k \geq 1$. We shall show that Q_{k+1} is bipartite. By definition, $Q_{k+1} = Q_k \times K_2$ is obtained by taking two copies of Q_k and joining the corresponding vertices of the copies by a new edge. Let S and T be a bi-partition of Q_k and let S' and T' be a bi-partition of its copy. Then clearly, $S \cup T'$ and $S' \cup T$ are bi-partitions of Q_{k+1}. d). The complete bipartite graph $K_{2,3}$. (Prove!)

Exercise 38: $K_2 + K_2$ is not a bipartite graph, although K_2 is bipartite. To prove $G_1 \times G_2$ is a bipartite graph if G_1 and G_2 are bipartite, follow a similar argument of the answer to Exercise 36c.

Appendix E

Answers to Chapter 5

Section 5.1

Exercise 2: a) $\gcd(1729, 13) = 13$, because 13 divides 1729. b) $\gcd(17, 47) = \gcd(47, 17) = \gcd(17, 13) = \gcd(13, 4) = \gcd(4, 1) = 1$. c) $\gcd(27, 81) = \gcd(81, 27) = 27$. d) $\gcd(1, 1000) = \gcd(1000, 1) = 1$. e) $\gcd(551, 1769) = \gcd(1769, 551) = \gcd(551, 116) = \gcd(116, 87) = \gcd(87, 29) = 3$.

Exercise 4: We shall write a function in pseudo-code and the reader is asked to translate it into C:

int lcm(int m, int n) { int min,max,t; if $(m > n)\{\max = m; \min = n;\}$; else $\{\max = n; \min = m;\}$
$t = 1$; while $(t * \max \bmod \min\ != 0)$ $t++$; printf(“%d'', $t * \max$)}

Exercise 6: Let the minimum of the positive integral linear combination of m and n be $p = a'm+b'n$. Any divisor of m and n divides any integral linear combination of m and n and hence p. In particular, the $\gcd(m, n)$ divides p. Hence $\gcd(m, n) \leq p$. It remains to show that $\gcd(m, n) \geq p$. We shall show that p divides both m and n. If, say, p did not divide m, then by the division algorithm, we can write $m = qp + r$ where q is the quotient and r the remainder with $0 < r < p$. This means that $r = m - qp = m - q(a'm + b'n)$. Since $0 < r < p$, r would be a smaller positive integral linear combination of m and n, which is a contradiction. Hence p divides both m and n. Therefore, $p \leq \gcd(m, n)$.

Exercise 8: True. Let $P(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + +a_1 n + a_0$. Then $\lim_{n\to\infty} |P(n)/n^k| = |a_k|$. This means that in particular for $\epsilon = |a_k|/2 > 0$, there exists n_0 such that $|P(n)/n^k| - |a_k| > |a_k| - |a_k|/2 = |a_k|/2$ for all $n \geq n_0$. (Since, by definition, if $a_n \to l$ as $n \to$, then, for every $\epsilon > 0$, there exists an n_0 such that $l - \epsilon < a_n < l + \epsilon$ for all $n \geq n_0$. Hence $|P(n)/n^k| > |a_k|/2$ or $|P(n)| > cn^k$ for $n \geq n_0$ with $c = |a_k|/2$.

Exercise 10: We can approximate a sum by integral by using the formula: $\sum_{i=1}^{n-1} f(i) \approx \int_1^n f(x)\,dx$ function , assuming that $f(x)$ is a differentiable function. Hence the given sum is equal to $O(\int_1^{n+1} x^k\,dx)$, whose value is $(n+1)^{k+1}/k + 1 - 1/k + 1$ which is $O(n^{k+1})$. (See *Euler's Summation Formula, Art of Computer Programming*, Vol 1, by D. Knuth.)

Exercise 12: Special case of Exercise 8.

Exercise 14: a) False, because the ratio $(5n^2+8)/n$ tends to infinity and hence the ratio is not bounded. b) False, for the same reason. c) True, since $n^2 = O(n^2)$ and $\log n = O(n)$ and by the product rule, we get the result. d) False, in general, $O(f(n)) - O(f(n)) = O(f(n))$. Proof. Note that each O symbol may represent a different function. Hence let $g(n) = O(f(n)$ and $h(n) = O(f(n)$. This means that, $|g(n)| \leq c_1|f(n)|$ for all $n \geq n_1$ and $|h(n)| \leq c_1|f(n)|$ for all $n \geq n_2$. Hence for $n \geq \max(n_1, n_2)$, both inequalities are valid simultaneously. Therefore, $|g(n) - h(n)| \leq |g(n)| + |h(n)| \leq (c_1 + c_2)|f(n)|$.

Exercise 16: If $g_1(n) = \Omega(f_1(n))$ and $g_2(n) = \Omega(f_2(n))$, then $g_1(n)g_2(n) = \Omega(f_1(n)f_2(n)$. Proof. Since $g_1(n) = \Omega(f_1(n))$ and $g_2(n) = \Omega(f_2(n))$, there are positive constants c_1, c_2 and positive integers n_1, n_2 such that $|g_1(n)| \geq c_1|f_1(n)|$ for all $n \geq n_1$ and $|g_2(n)| \geq c_2|f_2(n)|$ for all $n \geq n_2$. Hence for $n \geq \max(n_1, n_2)$ both inequalities are satisfied. Therefore,

$|g_1(n)g_2(n)| = |g_1(n)||g_2(n)| \geq c_1c_2|f_1(n)f_2(n)|$ for all $n \geq \max(n_1, n_2)$. Hence by the definition of Ω, we have $g_1(n)g_2(n) = \Omega(f_1(n)f_2(n)$.

Exercise 18: We use the constant N as the maximum number of integers the stack can hold. For example, it could be

```
#define N 100
```

stackInitialize( ){ h=0 }. int stackEmpty() { return h==0;}. void push(int x){ s[h++]=x;}. int pop(){return s[--h]}. stackFull(){ return h==N+1}.

Exercise 20: 5 boys can be seated in 5! ways. A girl can be seated before all of the boys or at the end of all the boys or in between two boys. Hence the first girl has 6 places to be seated and the second has 5 places and the third has 4 places. Hence the number is $5! \times 6 \times 5 \times 4 = 14400$.

Exercise 22: The number is the same as the number of multisubsets of 18 elements from a set of 4 elements. This number is $\binom{4+18-1}{18}$, which is $\binom{21}{3} = 2660$.

Exercise 24: True. Since if there were a magic square of order $n = 2$, then the sum of each line, column, and diagonals must be $n(n^2+1)/2 = 5$. For all the other values of n, a magic square exists.

Exercise 26: Let the common sum be s. Let us find the value of the sum of all the entries of magic square in two different ways and equate. On the one hand, the sum of all the entries is $+2+\cdots+n^2 = n^2(n^2+1)/2$, and on the other hand this sum is ns. Equating we get $ns = n^2(n^2+1)/2$. Hence, $s = n(n^2+1)/2$.

Exercise 28: Float harmonic(int n) { if (n==1) return 1; return (harmonic(n-1) + 1/n);}

Exercise 30: Define the (in pascal notation) matrix N: array[1..n,-1..1] of real; where the coefficients $m_{ij} = M[i,j]$ are N[i,j-i]. In this manner, the coefficients which are identical to zero in the matrix M do not occupy any storage space. The storage space needed for the old matrix M is $O(n^2)$, whereas the space needed for the new matrix is only $3cn = O(n)$ where $c > 0$ is a constant.

Exercise 32: During the first scan, the integers 55 and 18 are exchanged. In the next scan, left and right pointers completely cross at 94 and 6 respectively. We exchange 6 (right pointer integer) and the pivot 44 to get the sequence $(6, 18, 12, 42, 44, 94, 55, 67)$. We are left with two files $(6, 18, 12, 42)$ and $(94, 55, 67)$. Consider the right file. The first scan of this file leave the left and right pointers towards 67. Exchange 67 and 94. There is no right file now . The left file is $(67, 55)$. The first scan exchanges 55 and 67. Similarly, apply the scan and exchange to the initial left file $(6, 18, 12, 42)$ to get $(6, 12, 18, 42)$. By combining the two files , the array is sorted.

Exercise 34: See the text for a program in C 5.15 taking the rightmost integer as the pivot. We need to modify only the function split. Now the pivot is a[l] instead of a[r].

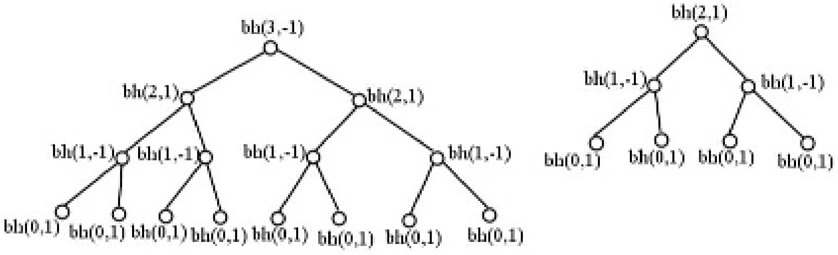


Figure E.1: Binary trees corresponding to calls: bh(3,-1) and bh(2,1)

Exercice 36: The output to call bh(3,-1) is: -1,+2,-1,-3,-1,+2,-1 and the output for the call bh(2,1) is -1,+2,-1.

Exercise 38: Hint: We shall use three stacks (represented by arrays) a,b,c of integers corresponding to three pegs. Let ha,hb,hc

Table E.1: Partition function in C with leftmost as pivot

```
int split(int a[], int l, int r)
{int pivot,t,k,j;
//index k scans array a from left to right. j scans from right to
left
pivot=a[l];k=l;j=r+1;
   do{
      do k++; while(a[k] < pivot)&&(k! = r);
      do j–; while(a[j] > pivot);//pivot element acts as sentinel
//(k! = r)?:to stop the scan when pivot is the greatest in the
array
      t=a[j];a[j]=a[k];a[k]=t;//
    while (j > k)}
a[k]=a[j];a[j]=a[l];a[l]=t;
return k;
}
```

be three integer variables pointing to the current integers at the top of the stacks in a,b,c respectively. Initially all the disks are stacked on the stack a where $a[i] = 1$ for $i = 1, 2, \ldots, n$ and the disks are in the order $n, n-1, \cdots, 2, 1$ from bottom to top. The arrays b, c are empty. Initially, $ha = n, hb = hc = 0$. Use the algorithm: Imagine the arrays a, b, c in the form of a triangle. On odd numbered moves, move the smallest disk 1 one position in the clockwise direction. On even numbered moves, make the only legal move not involving disk 1.

Appendix F

Answers to Chapter 6

Section 6.1

Exercise 2:

Table F.1: Truth Table

p	q	r	$p \to q$	$q \to r$	$[(p \to q) \land (q \to r)]$	$p \to r$	$[(p \to q) \land (q \to r)]$ $\to (p \to r)$
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	1	0	1	0	0	1	1
0	1	1	1	1	1	1	1
1	0	0	0	1	0	0	1
1	0	1	0	1	0	1	1
1	1	0	1	0	0	0	1
1	1	1	1	1	0	1	1

Exercise 4: We shall prove that $E = p \to q, \neg p \vdash \neg q$ is a fallacy by the algebraic method. We use $p \to q = \neg p \lor q)$.
Equivalently, we shall prove that $E = (p \to q) \land (\neg p) \to \neg q$ is not a tautology.
$E = (p' + q)p' \to q'$ (denoting $\neg$ by ', $\land$ by juxtaposition, $\lor$ by +)
$E = (p' + p'q)' + q' = p + p + q' + q'$ (by DeMoran's law and involution)$=p + q'$ which has the value 0 when $p = 0$ and $q = 1$. Hence it is not a tautology.

Exercise 6: $\lim_{n\to\infty} a_n = l$ means $\forall\epsilon > 0, \exists n_0 \in N$ such that $|a_n - l| < \epsilon$, $\forall n > n_0$.
$\lim_{n\to\infty} a_n \neq l$ means $\exists\epsilon > 0$ such that $\forall n_0 \in N$, $\exists n > n_0$, $|a_n - l| \geq \epsilon$.

Exercise 8: Statement of the Converse of Lagrange's Theorem: Let G be a finite group. If an integer p divides the order of G, then G has a subgroup of order p. (This converse is false.)

Exercise 10: True: Justification: It is given that 250 is a perfect square. This means that there exists an integer k such that $250 = k^2$. Now, $1000 = 250 \times 4$. Plugging in k^2 for 250, we get, $1000 = k^2 \times 4 = (2k)^2$. Hence, 1000 is a perfect square. (Note that the statement "250 is a perfect square" is false and the statement "1000 is a perfect square" is false, but the implication is true! See the truth table for implication.)

Exercise 12: The sample space consists of all 36 ordered pairs (i, j) with $1 \leq i, j \leq 6$. The number of favorable cases is the number of elements of the set $\{\, (i, j) \mid i \geq 2 or j \geq 2 \,\}$, which is $35 (= 36 - 1)$ (all the sample points except $(1, 1)$). Hence the probability is $35/36$.

Exercise 14: $p(A/B) = p(A \cap B)/p(B) = p(B)/p(B) = 1$.

Exercise 16: Note that the numerators in the p_i line are binomial coefficients $\binom{4}{i}$ for $0 \leq i \leq 4$.

Table F.2: The distribution of the random variable X

x_i	0	1	2	3	4
p_i	1/16	4/16	6/16	4/16	1/16

$E(X) = 0 \times 1/16 + 1 \times 4/16 + 2 \times 6/16 + 3 \times 4/16 + 4 \times 1/16 = 2$.
$var(X) = E(X^2) - (E(X))^2 = 0^2 \times 1/16 + 1^2 \times 4/16 + 2^2 \times 6/16 + 3^2 \times 4/16 + 4^2 \times 1/16 - 2^2 = 1$.

Exercise 18: For $i = 1, 2, \cdots, n$ define the random variables $X_1, X_2, \cdots, X_n$ as follows: Let $X_i = 1$ if the ith professor gets his/her own letter, and $X_i = 0$, otherwise. Let $X = \sum_{i=1}^{n} X_i$. We want to find $E(X)$. $E(X) = E(\sum_{i=1}^{n} X_i) = \sum_{i=1}^{n} E(X_i)$, since the expectation is a linear function. Since the letters are given at random to different professors, the probability that the ith professor gets his/her own letter is $1/n$. Hence $E(X_i) = 1 \times 1/n + 0 \times (1 - 1/n) = 1/n$. Substituting, $E(X) = 1$.

Exercise 20: $E(X) = \sum_{i=0}^{n} i \times \binom{n}{i} p^i q^{n-i}$. Writing, $\binom{n}{i} = n!/i!(n-i)!$ and $n! = (n-1)! \times n$ and taking np as a common factor, we get $E(X) = np(q+p)^{n-1} = np1^{n-1} = np$. (By the binomial theorem and $q + p = 1$.) $var(X) = E(X^2) - (E(X))^2$. $E(X^2) = \sum_{i=0}^{n} x_i^2 p_i$. Now, $\sum_{i=1}^{n} (x_i(x_i - 1)p_i) = (\sum_{i=1}^{n} x_i^2 p_i) - E(X)$. Therefore, $\sum_{i=1}^{n} x_i^2 p_i = \sum_{i=1}^{n} (x_i(x_i - 1)p_i) + E(X)$. We first calculate $\sum_{i=1}^{n} (x_i(x_i - 1)p_i)$. Substituting for p_i and x_i and taking $n(n-1)p^2$ as a common factor, we obtain, by the binomial theorem, $\sum (x_i(x_i - 1)p_i) = n(n-1)p^2(q+p)^{n-2} = n(n-1)p^2$. But $E(X) = np$. Hence, $var(X) = n(n-1)p^2 + np - n^2p^2 = np(1-p) = npq$.

Index