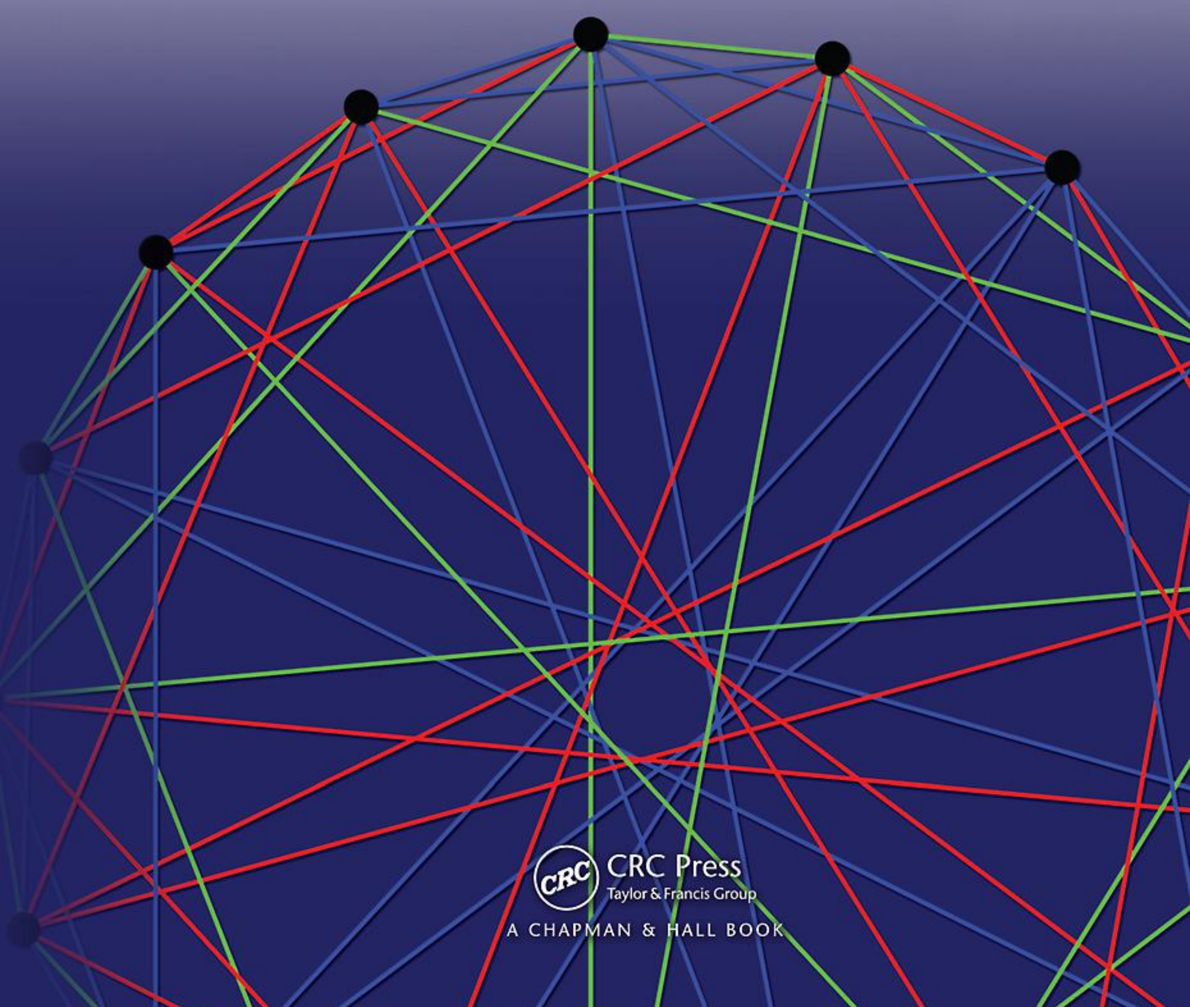
FRED S. ROBERTS • BARRY TESMAN

Applied Combinatorics

SECOND EDITION



CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Applied Combinatorics

SECOND EDITION

Applied Combinatorics

SECOND EDITION

FRED S. ROBERTS
BARRY TESMAN

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group an **informa** business
A CHAPMAN & HALL BOOK

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

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number: 978-1-4200-9982-9 (Hardback)

Library of Congress Cataloging-in-Publication Data

Roberts, Fred S.
Applied combinatorics/ Fred Roberts. -- 2nd ed. / Barry Tesman.
p. cm.
Originally published: 2nd ed. Upper Saddle River, N.J. : Pearson Education/Prentice-Hall, c2005.
Includes bibliographical references and index.
ISBN 978-1-4200-9982-9 (hardcover : alk. paper)
1. Combinatorial analysis. I. Tesman, Barry. II. Title.

QA164.R6 2009
511’.6--dc22 2009013043

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

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

To: Helen,
David, and
Sarah

-F.S.R.

To: Johanna,
Emma, and
Lucy

-B.T.

Contents

Preface

The second edition of *Applied Combinatorics* comes 20 years after the first one. It has been substantially rewritten with more than 200 pages of new material and significant changes in numerous sections. There are many new examples and exercises. On the other hand, the main philosophy of the book is unchanged. The following three paragraphs are from the preface of the first edition and these words still ring true today.

> Perhaps the fastest growing area of modern mathematics is combinatorics. A major reason for this rapid growth is its wealth of applications, to computer science, communications, transportation, genetics, experimental design, scheduling, and so on. This book introduces the reader to the tools of combinatorics from an applied point of view.
>
> Much of the growth of combinatorics has gone hand in hand with the development of the computer. Today's high-speed computers make it possible to implement solutions to practical combinatorial problems from a wide variety of fields, solutions that could not be implemented until quite recently. This has resulted in increased emphasis on the development of solutions to combinatorial problems. At the same time, the development of computer science has brought with it numerous challenging combinatorial problems of its own. Thus, it is hard to separate combinatorial mathematics from computing. The reader will see the emphasis on computing here by the frequent use of examples from computer science, the frequent discussion of algorithms, and so on. On the other hand, the general point of view taken in this book is that combinatorics has a wealth of applications to a large number of subjects, and this book has tried to emphasize the variety of these applications rather than just focusing on one.
>
> Many of the mathematical topics presented here are relatively standard topics from the rapidly growing textbook literature of combinatorics. Others are taken from the current research literature, or are chosen because they illustrate interesting applications of the subject. The book is distinguished, we believe, by its wide-ranging treatment of applications. Entire sections are devoted to such applications as switching functions, the use of enzymes to uncover unknown RNA chains, searching and sorting problems of information retrieval, construction of error-correcting codes, counting of chemical

compounds, calculation of power in voting situations, and uses of Fibonacci numbers. There are entire sections on applications of recurrences involving convolutions, applications of eulerian chains, applications of generating functions, and so on, that are unique to the literature.

WHAT'S NEW IN THIS EDITION?

Much of the appeal of this book has stemmed from its references to modern literature and real applications. The applications that motivate the development and use of combinatorics are expanding greatly, especially in the natural and social sciences. In particular, computer science and biology are primary sources of many of the new applications appearing in this second edition. Along with these additions, we have also added some major new topics, deleted some specialized ones, made organizational changes, and updated and improved the examples, exercises, and references to the literature.

Some of the major changes in the second edition are the following:

Chapter 1 (What Is Combinatorics?): We have added major new material on list colorings, expanding discussion of scheduling legislative committees. List colorings are returned to in various places in the book.

Chapter 2 (Basic Counting Rules): Section 2.16, which previously only discussed algorithmic methods for generating permutations, has been substantially expanded and broken into subsections. Section 2.18, which introduces the notion of "good algorithms" and NP-completeness, has been substantially rewritten and modernized. A new section on the pigeonhole principle has been added. The section consists of the material from Section 8.1 of the first edition and some of the material from Section 8.2 that deals with Ramsey theory. We have also added a substantial new section on the inversion distance between permutations and the study of mutations in evolutionary biology.

Chapter 3 (Introduction to Graph Theory): A major new subsection has been added to Section 3.3, the graph coloring section. This new subsection deals with the generalizations of graph coloring, such as set coloring, list coloring, and T-coloring, that have been motivated by practical problems such as mobile radio telephone problems, traffic phasing, and channel assignment. We have also introduced a major new subsection on phylogenetic tree reconstruction. Much of the material on Ramsey theory from Chapter 8 of the first edition, and not covered in Chapter 2, has been updated and presented in a new section.

Chapter 4 (Relations): This chapter is brand new. Concepts of binary relations are defined and connected to digraphs. Orders are introduced using digraphs and relations, and parts of the new chapter deal with linear and weak orders; partial orders, linear extensions and dimension; chains and comparability graphs; lattices; boolean algebras; and switching functions and gate networks. The chapter is closely tied to applications ranging from information theory to utility theory to searching and sorting, as well as returning to the earlier applications such as switching functions. This chapter includes some applications not widely discussed in the combinatorics literature, such as preference, search engines, sequencing by hybridization, and psychophysical scaling. Examples based on Chapter 4 concepts have also been added to many subsequent chapters. Coverage of Chapter 4 can be delayed until after Chapter 11.

Chapter 5 (Generating Functions and Their Applications): In the first edition, this was Chapter 4. Many new concepts and examples introduced in earlier chapters are revisited here, for example, weak orders from Chapter 4 and list colorings from Chapter 3.

Chapter 6 (Recurrence Relations): This was Chapter 5 in the first edition. New material on DNA sequence alignment has been added as has material on the "transposition average" of permutations.

Chapter 7 (The Principle of Inclusion and Exclusion): This was Chapter 6 in the first edition. We have added major new material on cryptography and factoring integers throughout the chapter (and revisited it later in the book).

Old Chapter 8 - 1st edition (Pigeonhole Principle): This chapter has been dropped, with important parts of the material added to Chapter 2 and other parts included from time to time throughout the book.

Chapter 8 (The Pólya Theory of Counting): This was Chapter 7 in the first edition. Some examples based on newly added Chapter 4 concepts such as weak order run through the chapter. A subsection on automorphisms of graphs has been added and returned to throughout the chapter.

Chapter 9 (Combinatorial Designs): Major additions to this chapter include a section on orthogonal arrays and cryptography, including authentication codes and secret sharing. There is also a new section on connections between modular arithmetic and the RSA cryptosystem and one on resolvable designs with applications to secret sharing. A new section on "Group Testing" includes applications to identifying defective products, screening diseases, mapping genomes, and satellite communication.

Chapter 10 (Coding Theory): There is a new subsection on "consensus decoding" with connections to finding proteins in molecular sequences and there are added connections of error-correcting codes to compact disks. Material on "reading" DNA to produce proteins is also new.

Chapter 11 (Existence Problems in Graph Theory): We have added new subsections to Section 11.2 that deal with the one-way street problem. These new subsections deal with recent results about orientations of square and annular grids reflecting different kinds of cities. We have added a new subsection on testing for connectedness of truly massive graphs, arising from modern applications involving telecommunications traffic and web data. There is also a new subsection on sequencing DNA by hybridization.

Chapter 12 (Matching and Covering): There are many new examples illustrating the concepts of this chapter, including examples involving smallpox vaccinations, sound systems, and oil drilling. We have introduced a new section dealing with stable marriages and their many modern applications, including the assignment of interns to hospitals, dynamic labor markets, and strategic behavior. A section on maximum-weight matching, which was in Chapter 13 of the first edition, has been moved to this chapter.

Chapter 13 (Optimization Problems for Graphs and Networks): We have introduced a new subsection on Menger's Theorems. There are also many new examples throughout the chapter, addressing such problems as building evacuation, clustering and data mining, and distributed computing.

Appendix (Answers to Selected Exercises): Answers to Selected Exercises was included in the 1st edition of the book but it has been greatly expanded in this edition.

CONTINUING FEATURES

While the second edition has been substantially changed from the first, this edition continues to emphasize the features that make this book unique:

- Its emphasis on applications from a variety of fields, the treatment of applications as major topics of their own rather than as isolated examples, and the use of applications from the current literature.
- Many examples, especially ones that tie in new topics with old ones and are revisited throughout the book.
- An emphasis on problem solving through a variety of exercises that test routine ideas, introduce new concepts and applications, or attempt to challenge the reader to use the combinatorial techniques developed. The book continues to be based on the philosophy that the best way to learn combinatorial mathematics, indeed any kind of mathematics, is through problem solving.
- A mix of difficulty in topics with careful annotation that makes it possible to use this book in a variety of courses at a variety of levels.
- An organization that allows the use of the topics in a wide variety of orders, reflecting the somewhat independent nature of the topics in combinatorics while at the same time using topics from different chapters to reinforce each other.

THE ORGANIZATION OF THE BOOK

The book is divided into four parts. The first part (Chapters 2, 3, and 4) introduces the basic tools of combinatorics and their applications. It introduces fundamental counting rules and the tools of graph theory and relations. The remaining three parts are organized around the three basic problems of combinatorics: the counting problem, the existence problem, and the optimization problem. These problems are discussed in Chapter 1. Part II of the book is concerned with more advanced tools for dealing with the counting problem: generating functions, recurrences, inclusion/exclusion, and Pólya Theory. Part III deals with the existence problem. It discusses combinatorial design, coding theory, and special problems in graph theory. It also begins a series of three chapters on graphs and networks (Chapters 11–13, spanning Parts III and IV) and begins an introduction to graph algorithms. Part IV deals with combinatorial optimization, illustrating the basic ideas through a continued study of graphs and networks. It begins with a transitional chapter on matching and covering that starts with the existence problem and ends with the optimization problem. Then Part IV ends with a discussion of optimization problems for graphs and networks. The division of the book into four parts is somewhat arbitrary, and many topics illustrate several different aspects of combinatorics, for instance both existence and optimization questions. However, dividing the book into four parts seemed to be a reasonable way to organize the large amount of material that is modern combinatorics.

PREREQUISITES

This book can be used at a variety of levels. Most of the book is written for a junior/senior audience, in a course populated by math and computer science majors and nonmajors. It could also be appropriate for sophomores with sufficient mathematical maturity. (Topics that can be omitted in elementary treatments are indicated throughout.) On the other hand, at a fast pace, there is more than enough material for a challenging graduate course. In the undergraduate courses for which the material has been used at Rutgers, the majority of the enrollees come from mathematics and computer science, and the rest from such disciplines as business, economics, biology, and psychology. At Dickinson, the material has been used primarily for junior/senior-level mathematics majors. The prerequisites for these courses, and for the book, include familiarity with the language of functions and sets usually attained by taking at least one course in calculus. Infinite sequences and series are used in Chapters 5 and 6 (though much of Chapter 6 uses only the most elementary facts about infinite sequences, and does not require the notion of limit). Other traditional topics of calculus are not needed. However, the mathematical sophistication attained by taking a course like calculus is a prerequisite. Also required are some tools of linear algebra, specifically familiarity with matrix manipulations. An understanding of mathematical induction is also assumed. (There are those instructors who will want to review mathematical induction in some detail at an early point in their course, and who will want to quickly review the language of sets.) A few optional sections of the book require probability beyond what is developed in the text. Other sections introduce topics in modern algebra, such as groups and finite fields. These sections are self-contained, but they would be too fast-paced for a student without sufficient background.

ALGORITHMS

Many parts of the book put an emphasis on algorithms. This is inevitable, as combinatorics is increasingly connected to the development of precise and efficient procedures for solving complicated problems, and because the development of combinatorics is so closely tied to computer science. Our aim is to introduce students to the notion of an algorithm and to introduce them to some important examples of algorithms. For the most part, we have adopted a relatively informal style in presenting algorithms. The style presumes little exposure to the notion of an algorithm and how to describe it. The major goal is to present the basic idea of a procedure, without attempting to present it in its most concise or most computer-oriented form. There are those who will disagree with this method of presenting algorithms. Our own view is that no combinatorics course is going to replace the learning of algorithms. The computer science student needs a separate course in algorithms that includes discussion of implementing the data structures for the algorithms presented. However, all students of combinatorics need to be exposed to the idea of algorithm, and to the algorithmic way of thinking, a way of thinking that is so central and basic to the subject. We realize that our compromise on how to present algorithms will not make everyone happy. However, it should be pointed out that for students with a background in computer science, it would make for interesting, indeed important, exercises to translate the informal algorithms of the text into more precise computer algorithms or even computer programs.

ROLE OF EXAMPLES AND APPLICATIONS

Applications play a central role in this book and are a feature that makes the book unique among combinatorics books. The instructor is advised to pick and choose among the applications or to assign them for outside reading. Many of the applications are presented as Examples that are returned to as the book progresses. It is not necessary for either the instructor or the student to be an expert in the area of application represented in the various examples and subsections of the book. They tend to be self-contained and, when not, should be readily understood with some appropriate searching of the Internet.

The connection between combinatorics and computer science is well understood and vitally important and does not need specific emphasis in this discussion.

Of particular importance in this book are examples from the biological sciences. Our emphasis on such examples stems from our observation that the connection between the biological and the mathematical sciences is growing extremely fast. Methods of mathematics and computer science have played and are playing a major role in modern biology, for example in the "human genome project" and in the modeling of the spread of disease. Increasingly, it is vitally important for mathematical scientists to understand such modern applications and also for students of the biological sciences to understand the importance for their discipline of mathematical methods such as combinatorics. This interdisciplinarity is reflected in the growing number of schools that have courses or programs at the interface between the mathematical and the biological sciences.

While less advanced than the connection between the mathematical and the biological sciences, the connection between the mathematical and the social sciences is also growing rapidly as more and more complex problems of the social sciences are tackled using tools of computer science and mathematical modeling. Thus, we have introduced a variety of applications that arise from the social sciences, with an emphasis on decisionmaking and voting.

PROOFS

Proving things is an essential aspect of mathematics that distinguishes it from other sciences. Combinatorics can be a wonderful mechanism for introducing students to the notion of mathematical proof and teaching them how to write good proofs. Some schools use the combinatorics course as the introduction to proofs course. That is not our purpose with this book. While the instructor using this book should include proofs, we tend to treat proofs as rather informal and do not put emphasis on writing them. Many of the harder proofs in the book are starred as optional.

EXERCISES

The exercises play a central role in this book. They test routine ideas, introduce new concepts and applications, and attempt to challenge the reader to use the combinatorial techniques developed in the text. It is the nature of combinatorics, indeed the nature of most of mathematics, that it is best mastered by doing many problems. We have tried to include a wide variety of both applied and theoretical exercises, of varying degrees of difficulty, throughout the book.

WAYS TO USE THE BOOK IN VARIOUS SETTINGS

This book is appropriate for a variety of courses at a variety of levels. We have both used the material of the book for several courses, in particular a one-semester course entitled Combinatorics and a one-semester course entitled Applied Graph Theory. The combinatorics course, taught to juniors and seniors, covers much of the material of Chapters 1, 2, 3, 5, 6, 7, 9, and 10, omitting the sections indicated by footnotes in the text. (These are often proofs.) At Rutgers, a faster-paced course that Fred Roberts has used with first-year graduate students puts more emphasis on proofs, includes many of the optional sections, and also covers the material of either Chapter 8 or Chapter 12. In an undergraduate or a graduate course, the instructor could also substitute for Chapters 9 and 10 either Chapter 8 or Chapter 11 and parts of Chapters 12 and 13. Including Chapter 11 is especially recommended at institutions that do not have a separate course in graph theory. Similarly, including parts of Chapter 13 is especially recommended for institutions that do not have a course in operations research. At Rutgers, there are separate (both undergraduate and graduate) courses that cover much of the material of Chapters 11 to 13.

Other one-semester or one-quarter courses could be designed from this material, as most of the chapters are relatively independent. (See the discussion below.) At Rutgers, the applied graph theory course that is taught is built around Chapters 3 and 11, supplemented with graph-theoretical topics from the rest of the book (Chapters 4, 12, and 13) and elsewhere. (A quick treatment of Sections 2.1 through 2.7, plus perhaps Section 2.18, is needed background.) Chapters 3, 11, 12, and 13 would also be appropriate for a course introducing graph algorithms or a course called Graphs and Networks. The entire book would make a very appropriate one-year introduction to modern combinatorial mathematics and its applications. A course emphasizing applications of combinatorics for those who have previously studied combinatorics could be constructed out of the applied subsections and examples in the text.

This book could be used for a one-semester or one-quarter sophomore-level course. Such a course would cover much of Chapters 1, 2, and 3, skip Chapters 4 and 5, and cover only Sections 6.1 and 6.2 of Chapter 6. It would then cover Chapter 7 and parts of Chapter 11. Starred sections and most proofs would be omitted. Other topics would be added at the discretion of the instructor.

DEPENDENCIES AMONG TOPICS AND ORDERS IN WHICH TO USE THE BOOK

In organizing any course, the instructor will wish to take note of the relative independence of the topics here. There is no well-accepted order in which to present an introduction to the subject matter of combinatorics, and there is no universal agreement on the topics that make up such an introduction. We have tried to write this book in such a way that the chapters are quite independent and can be covered in various orders.

Chapter 2 is basic to the book. It introduces the basic counting rules that are used throughout. Chapter 3 develops just enough graph theory to introduce the subject. It emphasizes graph-theoretical topics that illustrate the counting rules developed in Chapter 2. The ideas introduced in Chapter 3 are referred to in places throughout the book, and most heavily in Chapters 4, 11, 12, and 13. It is possible to use this book for a one-semester or one-quarter course in combinatorics without covering Chapter 3. However, in our opinion, at least the material on graph coloring (Sections 3.3 and 3.4) should be included. The major dependencies beyond Chapter 3 are that Chapter 4 depends on Chapter 3; Chapter 6 after Section 6.2 depends on Chapter 5; Chapter 7 refers to examples developed in Chapters 3 and 6; Chapters 11, 12, and 13 depend on Chapter 3; and Section 10.5 depends on Chapter 9. Ideas from Chapter 12 are used in Chapter 13, Section 13.3.8.

COMBINATORICS IS RAPIDLY CHANGING

Finally, it should be emphasized that combinatorics is a rapidly growing subject and one whose techniques are being rapidly developed and whose applications are being rapidly explored. Many of the topics presented here are close to the frontiers of research. It is typical of the subject that it is possible to bring a newcomer to the frontiers very quickly. We have tried to include references to the literature of combinatorics and its applications that will allow the interested reader to delve more deeply into the topics discussed here.

ACKNOWLEDGMENTS

Fred Roberts started on the first edition of this book in 1976, when he produced a short set of notes for his undergraduate course in combinatorics at Rutgers. Over the years that this book has changed and grown, he has used it regularly as the text for that course and for the other courses described earlier, as has Barry Tesman. It has also been a great benefit to the authors that others have used this material as the text for their courses and have sent extensive comments. They would particularly like to thank for their very helpful comments: Midge Cozzens, who used drafts of the first edition at Northeastern; Fred Hoffman, who used them at Florida Atlantic; Doug West, who used them at Princeton; Garth Isaak, who used drafts of the second edition at Lehigh; and Buck McMorris, who used drafts of the second edition at Illinois Institute of Technology.

We would especially like to thank present and former students who have helped in numerous ways in the preparation of both editions of the book, by proofreading, checking exercises, catching numerous mistakes, and making nasty comments. Fred Roberts wants to acknowledge the help of Midge Cozzens, Shelly Leibowitz, Bob Opsut, Arundhati Ray-Chaudhuri, Sam Rosenbaum, and Jeff Steif. Barry Tesman acknowledges the help of Kathy Clawson and John Costango. Both authors would also like to thank Aritanan Gruber, David Papp, and Paul Raff for their help with working out answers to selected exercises.

We have received comments on this material from many people. We would specifically like to thank the following individuals, who made extremely helpful comments on the first edition at various stages during the reviewing process for that edition, as well as at other times: John Cozzens, Paul Duvall, Marty Golumbic, Fred Hoffman, Steve Maurer, Ronald Mullin, Robert Tarjan, Tom Trotter, and Alan Tucker. As we were preparing the second edition, we received very helpful comments on the first edition from Steve Maurer. Jeff Dinitz gave us detailed comments on drafts of Chapters 9 and 10. For the second edition, we received extremely helpful comments from the following reviewers: Edward Allen, Martin Billik, John Elwin, Rodney W. Forcade, Kendra Killpatrick, Joachim Rosenthal, Sung-Yell Song, Vladimir Tonchev, and Cun-Quan Zhang. Although we have received a great deal of help with this material, errors will almost surely remain. We alone are responsible for them.

As the first edition of this book grew, it was typed and retyped, copied and recopied, cut (literally with scissors), pasted together (literally with glue), uncut, glued, and on and on. Fred Roberts had tremendous help with this from Lynn Braun, Carol Brouillard, Mary Anne Jablonski, Kathy King, Annette Roselli, and Dotty Westgate. It is quite remarkable how the business of publishing has changed. For the second edition, Barry Tesman did the typing, retyping, (electronic) cutting and pasting, etc. Without an electronic copy of the first edition, the task of scanning the complete first edition went to Barry Tesman's former student Jennifer Becker. He acknowledges her herculean task. He also would like to thank members of his department, LIS, and the administration at Dickinson for their support and contributions.

The authors would like to thank Pearson Education for permission to modify and reproduce material from Fred Roberts' book *Discrete Mathematical Models, with Applications to Social, Biological, and Environmental Problems*, 1976, Pearson Education, Inc. All rights are reserved by Pearson Education. In particular we have modified Tables 2.1 and 8.1, reproduced exercises from Section 3.6, and reproduced selections from pages 8–14, 21–23, 25–29, 31–32, 81–82, 84–86, 156–157, and 165–166.

Both of us would like to thank our families for their support. Those who have written a book will understand the number of hours it takes away from one's family: cutting short telephone calls to proofread, canceling trips to write, postponing outings to create exercises, stealing away to make just one more improvement. Our families have been extremely understanding and helpful. Fred Roberts would like to thank his late parents, Louis and Frances Roberts, for their love and support. He would like to thank his mother-in-law, the late Lily Marcus, for her assistance, technical and otherwise. He would like to thank his wife, Helen, who, it seems, is always a "book widow." She has helped not only by her continued support and guidance, and inspiration, but she has also co-authored one chapter of this book, and introduced him to a wide variety of topics and examples which she developed for her courses and which we have freely scattered throughout this book. Finally, he would like to thank Sarah and David. When the first edition was being written, their major contribution to it was to keep him in constant good humor. Remarkably, as his children have grown to adulthood, they have grown to contribute to his work in other ways: for instance, Sarah by introducing him to ideas of public health that are reflected in some of his current mathematical interests and in this book; and David by explaining numerous aspects of computer science, earning him in particular an acknowledgment in an important footnote later in the book. Fred Roberts does not need the counting techniques of combinatorics to count his blessings. Barry Tesman would like to thank his parents, Shirley and Harvey Tesman, for their love and support. He would like to thank his wife, Johanna, who was his silent partner in this undertaking and has been his (nonsilent) partner and best friend for the last 20 years. Finally, he would like to thank Emma and Lucy, for being Emma and Lucy.

Fred S. Roberts
froberts@dimacs.rutgers.edu

Barry Tesman
tesman@dickinson.edu

Notation

Set-theoretic Notation

$\cup$	union	$\emptyset$	empty set
$\cap$	intersection	$\{\cdots\}$	the set ...
$\subseteq$	subset (contained in)	$\{\cdots : \cdots\}$	the set of all ... such that ...
$\subsetneqq$	proper subset	A^c	complement of A
$\nsubseteq$	is not a subset	$A - B$	$A \cap B^c$
$\supseteq$	contains (superset)	$\lvert A\rvert$	cardinality of A, the number of elements in A
$\in$	member of		
$\notin$	not a member of		

Logical Notation

$\sim$	not
$\rightarrow$	implies
$\leftrightarrow$	if and only if (equivalence)
iff	if and only if

Miscellaneous

$\lfloor x \rfloor$	the greatest integer less than or equal to x	$[a, b]$	the closed interval consisting of all real numbers c with $a \leq c \leq b$
$\lceil x \rceil$	the least integer greater than or equal to x	$\approx$	approximately equal to
$f \circ g$	composition of the two functions f and g	$\equiv$	congruent to
$f(A)$	the image of the set A under the function f; that is, $\{f(a) : a \in A\}$	A^T	the transpose of the matrix A
(a, b)	the open interval consisting of all real numbers c with $a < c < b$	$\prod$	product
		$\sum$	sum
		$\int$	integral
		Re	the set of real numbers

Chapter 1

What Is Combinatorics?

1.1 THE THREE PROBLEMS OF COMBINATORICS

Perhaps the fastest-growing area of modern mathematics is combinatorics. Combinatorics is concerned with the study of arrangements, patterns, designs, assignments, schedules, connections, and configurations. In the modern world, people in almost every area of activity find it necessary to solve problems of a combinatorial nature. A computer scientist considers *patterns* of digits and switches to encode complicated statements. A shop supervisor prepares *assignments* of workers to tools or to work areas. An agronomist *assigns* test crops to different fields. An electrical engineer considers alternative *configurations* for a circuit. A banker studies alternative *patterns* for electronically transferring funds, and a space scientist studies such *patterns* for transferring messages to distant satellites. An industrial engineer considers alternative production *schedules* and workplace *configurations* to maximize efficient production. A university scheduling officer *arranges* class meeting times and students' *schedules*. A chemist considers possible *connections* between various atoms and molecules, and *arrangements* of atoms into molecules. A transportation officer *arranges* bus or plane *schedules*. A linguist considers *arrangements* of words in unknown alphabets. A geneticist considers *arrangements* of bases into chains of DNA, RNA, and so on. A statistician considers alternative *designs* for an experiment.

There are three basic problems of combinatorics. They are the *existence problem*, the *counting problem*, and the *optimization problem*. The existence problem deals with the question: Is there at least one arrangement of a particular kind? The counting problem asks: How many arrangements are there? The optimization problem is concerned with choosing, among all possible arrangements, that which is best according to some criteria. We shall illustrate these three problems with a number of examples.

Example 1.1 Design of Experiments Let us consider an experiment designed to test the effect on human beings of five different drugs. Let the drugs be labeled $1, 2, 3, 4, 5$. We could pick out five subjects and give each subject a different drug.

Table 1.1: A Design for a Drug Experiment[a]

		Day				
		M	Tu	W	Th	F
	A	1	2	3	4	5
	B	1	2	3	4	5
Subject	C	1	2	3	4	5
	D	1	2	3	4	5
	E	1	2	3	4	5

[a] The entry in the row corresponding to a given subject and the column corresponding to a given day shows the drug taken by that subject on that day.

Unfortunately, certain subjects might be allergic to a particular drug, or immune to its effects. Thus, we could get very biased results. A more effective use of five subjects would be to give each subject each of the drugs, say on five consecutive days. Table 1.1 shows one possible arrangement of the experiment. What is wrong with this arrangement? For one thing, the day of the week a drug is taken may affect the result. (People with Monday morning hangovers may never respond well to a drug on Monday.) Also, drugs taken earlier might affect the performance of drugs taken later. Thus, giving each subject the drugs in the same order might lead to biased results. One way around these problems is simply to require that no two people get the same drug on the same day. Then the experimental design calls for a 5×5 table, with each entry being one of the integers $1, 2, 3, 4, 5$, and with each row having all its entries different and each column having all its entries different. This is a particular kind of pattern. The crucial question for the designer of the drug experiment is this: Does such a design exist? This is the existence problem of combinatorics. ■

Let us formulate the problem more generally. We define a *Latin square*[1] as an $n \times n$ table that uses the numbers $1, 2, \ldots, n$ as entries, and does so in such a way that no number appears more than once in the same row or column. Equivalently, it is required that each number appear exactly once in each row and column. A typical existence problem is the following: Is there a 2×2 Latin square? The answer is yes; Table 1.2 shows such a square. Similarly, one may ask if there is a 3×3 Latin square. Again, the answer is yes; Table 1.3 shows one.

Our specific question asks whether or not there is a 5×5 Latin square. Table 1.4 shows that the answer is yes. (Is there an $n \times n$ Latin square for every n? The answer is known and is left to the reader.)

[1] The term "Latin square" comes from the fact that the elements were usually represented by letters of the Latin alphabet.

Table 1.2: A 2×2 Latin Square

1	2
2	1

Table 1.3: A 3×3 Latin Square

1	2	3
2	3	1
3	1	2

Table 1.4: A 5×5 Latin Square

1	2	3	4	5
2	3	4	5	1
3	4	5	1	2
4	5	1	2	3
5	1	2	3	4

Note that the Latin square is still not a complete solution to the problem that order effects may take place. To avoid any possible order effects, we should ideally have enough subjects so that each possible ordering of the 5 drugs can be tested. How many such orderings are there? This is the *counting problem*, the second basic type of problem encountered in combinatorics. It turns out that there are $5! = 120$ such orderings, as will be clear from the methods of Section 2.3. Thus, we would need 120 subjects. If only 5 subjects are available, we could try to avoid order effects by choosing the Latin square we use at random. How many possible 5×5 Latin squares are there from which to choose? We address this counting problem in Section 6.1.3.

As this very brief discussion suggests, questions of experimental design have been a major stimulus to the development of combinatorics.[2] We return to experimental design in detail in Chapter 9.

Example 1.2 Bit Strings and Binary Codes A *bit* or *binary digit* is a zero or a one. A *bit string* is defined to be a sequence of bits, such as 0001, 1101, or 1010. Bit strings are the crucial carriers of information in modern computers. A bit string can be used to encode detailed instructions, and in turn is translated into a sequence of on-off instructions for switches in the computer. A *binary code (binary block code)* for a collection of symbols assigns a different bit string to each of the symbols. Let us consider a binary code for the 26 letters in the alphabet. A typical such code is the Morse code which, in its more traditional form, uses dots for zeros and dashes for ones. Some typical letters in Morse code are given as follows:

$$\text{O}: 111, \ \text{A}: 01, \ \text{K}: 101, \ \text{C}: 1010.$$

If we are restricted to bit strings consisting of either one or two bits, can we encode all 26 letters of the alphabet? The answer is no, for the only possible strings are the following:

$$0, 1, 00, 01, 10, 11.$$

There are only six such strings. Notice that to answer the question posed, we had to *count* the number of possible arrangements. This was an example of a solution to a counting problem. In this case we counted by *enumerating* or listing all possible

[2] See Herzberg and Stanton [1984].

arrangements. Usually, this will be too tedious or time consuming for us, and we will want to develop shortcuts for counting without enumerating. Let us ask if bit strings of three or fewer bits would do for encoding all 26 letters of the alphabet. The answer is again no. A simple enumeration shows that there are only 14 such strings. (Can you list them?) However, strings of four or fewer bits will suffice. (How many such strings are there?) The Morse code, indeed, uses only strings of four or fewer symbols. Not every possible string is used. (Why?) In Section 2.1 we shall encounter a very similar counting problem in studying the genetic code. DNA chains encode the basic genetic information required to determine long strings of amino acids called proteins. We shall try to explain how long a segment in a DNA chain is required to be to encode for an amino acid. Codes will arise in other parts of this book as well, not just in the context of genetics or of communication with modern computers. For instance, in Chapter 10 we study the error-correcting codes that are used to send and receive messages to and from distant space probes, to fire missiles, and so on. ■

Example 1.3 The Best Design for a Gas Pipeline The flow of natural gas through a pipe depends on the diameter of the pipe, its length, the pressures at the endpoints, the temperature, various properties of the gas, and so on. The problem of designing an offshore gas pipeline system involves, among other things, decisions about what sizes (diameters) of pipe to use at various junctions or links so as to minimize total cost of both construction and operation. A standard approach to this problem has been to use "engineering judgment" to pick reasonable sizes of pipe and then to hope for the best. Any chance of doing better seems, at first glance, to be hopeless. For example, a modest network of 40 links, with 7 possible pipe sizes for each link, would give rise to 7^{40} possible networks, as we show in Section 2.1. Now 7^{40}, as we shall see, is a very large number. Our problem is to find the least expensive network out of these 7^{40} possibilities. This is an example of the third kind of combinatorial problem, an *optimization problem*, a problem where we seek to find the optimum (best, maximum, minimum, etc.) design or pattern or arrangement.

It should be pointed out that progress in solving combinatorial optimization problems has gone hand in hand with the development of the computer. Today it is possible to solve on a machine problems whose solution would have seemed inconceivable only a few years ago. Thus, the development of the computer has been a major impetus behind the very rapid development of the field of combinatorial optimization. However, there are limitations to what a computing machine can accomplish. We shall see this next.

Any finite problem can be solved in principle by considering all possibilities. However, how long would this particular problem take to solve by enumerating all possible pipeline networks? To get some idea, note that 7^{40} is approximately 6×10^{33}, that is, 6 followed by 33 zeros. This is a huge number. Indeed, even a computer that could analyze 1 billion (10^9) different pipeline networks in 1 second (one each nanosecond) would take approximately $1.9 \times 10^{17} = 190{,}000{,}000{,}000{,}000{,}000$ years to analyze all 7^{40} possible pipeline networks![3]

Much of modern combinatorics is concerned with developing procedures or *algorithms* for solving existence, counting, or optimization problems. From a practical point of view, it is a very important problem in computer science to analyze an algorithm for solving a problem in terms of how long it would take to solve or how much storage capacity would be required to solve it. Before embarking on a computation (such as trying all possibilities) on a machine, we would like to know that the computation can be carried out within a reasonable time or within the available storage capacity of the machine. We return to these points in our discussion of computational complexity in Sections 2.4 and 2.18.

The pipeline problem we have been discussing is a problem that, even with the use of today's high-speed computer tools, does not seem tractable by examining all cases. Any foreseeable improvements in computing speed would make a negligible change in this conclusion. However, a simple procedure gives rise to a method for finding the optimum network in only about $7 \times 40 = 280$ steps, rather than 7^{40} steps. The procedure was implemented in the Gulf of Mexico at a savings of millions of dollars. See Frank and Frisch [1970], Kleitman [1976], Rothfarb, *et al.* [1970], or Zadeh [1973] for references. This is an example of the power of techniques for combinatorial optimization. ■

Example 1.4 Scheduling Meetings of Legislative Committees Committees in a state legislature are to be scheduled for a regular meeting once each week. In assigning meeting times, the aide to the Speaker of the legislature must be careful not to schedule simultaneous meetings of two committees that have a member in common. Let us suppose that in a hypothetical situation, there are only three meeting times available: Tuesday, Wednesday, and Thursday mornings. The committees whose meetings must be scheduled are Finance, Environment, Health, Transportation, Education, and Housing. Let us suppose that Table 1.5 summarizes which committees have a common member. A convenient way to represent the information of Table 1.5 is to draw a picture in which the committees are represented by dots or points and two points are joined by an undirected line if and only if the corresponding committees have a common member. The resulting diagram is called a *graph*.

Figure 1.1 shows the graph obtained in this way for the data of Table 1.5. Graphs of this kind have a large number of applications, for instance in computer science, operations research, electrical engineering, ecology, policy and decision science, and in the social sciences. We discuss graphs and their applications in detail in Chapters 3 and 11 and elsewhere.

Our first question is this: Given the three available meeting times, can we find an assignment of committees to meeting times so that no member has to be at

[3] There are roughly 3.15×10^7 seconds per year, so $3.15 \times 10^7 \times 10^9$ or 3.15×10^{16} networks could be analyzed in a year. Then the number of years it takes to check 6×10^{33} networks is

$$\frac{6 \times 10^{33}}{3.15 \times 10^{16}} \approx 1.9 \times 10^{17}.$$

Table 1.5: Common Membership in Committees[a]

	Finance	Environment	Health	Transportation	Education	Housing
Finance	0	0	0	0	0	1
Environment	0	0	1	0	1	0
Health	0	1	0	1	1	1
Transportation	0	0	1	0	0	1
Education	0	1	1	0	0	1
Housing	1	0	1	1	1	0

[a]The i, j entry is 1 if committees i and j have a common member, and 0 otherwise. (The diagonal entries are taken to be 0 by convention.)

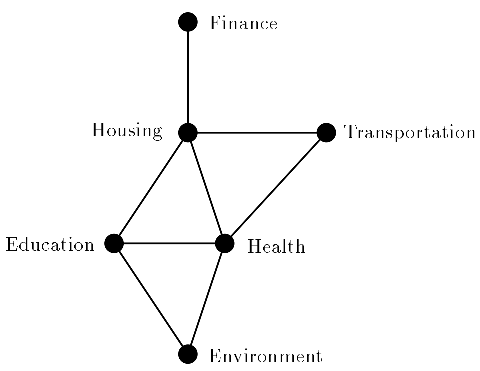


Figure 1.1: The graph obtained from the data of Table 1.5.

two meetings at once? This is an existence question. In terms of the graph we have drawn, we would like to assign a meeting time to each point so that if two points are joined by a line, they get different meeting times. Can we find such an assignment? The answer in our case, after some analysis, is yes. One assignment that works is this: Let the Housing and Environment committees meet on Tuesday, the Education and Transportation committees on Wednesday, and the Finance and Health committees on Thursday.

Problems analogous to the one we have been discussing arise in scheduling final exams or class meeting times in a university, in scheduling job assignments in a factory, and in many other scheduling situations. We shall return to such problems in Chapter 3 when we look at these questions as questions of *graph coloring* and think of the meeting times, for example, as corresponding to "colors."

The problem gets more realistic if each committee chair indicates a list of acceptable meeting times. We then ask if there is an assignment of committees to meeting times so that each committee is assigned an acceptable time and no member has to be at two meetings at once. For instance, suppose that the acceptable meeting times for Transportation are Tuesday and Thursday, for Education is Wednesday, and all other committees would accept any of the three days. It is not hard to show that there is no solution (see Exercise 13). We will then have solved the existence problem in the negative. This is an example of a scheduling problem known as a

Table 1.6: First Choice of Meeting Times

Committee	Finance	Environment	Health	Transportation	Education	Housing
Chair's first choice	Tuesday	Thursday	Thursday	Tuesday	Tuesday	Wednesday

list-coloring problem, a graph coloring problem where assigned colors (in this case representing "days of the week") are chosen from a list of acceptable ones. We return to this problem in Example 3.22. List colorings have been widely studied in recent years. See Alon [1993] and Kratochvíl, Tuza, and Voigt [1999] for recent surveys.

We might ask next: Suppose that each committee chair indicates his or her first choice for a meeting time. What is the assignment of meeting times that satisfies our original requirements (if there is such an assignment) and gives the largest number of committee chairs their first choice? This is an optimization question. Let us again take a hypothetical situation and analyze how we might answer this question. Suppose that Table 1.6 gives the first choice of each committee chair. One approach to the optimization question is simply to try to identify all possible satisfactory assignments of meeting times and for each to count how many committee chairs get their first choice. Before implementing any approach to a combinatorial problem, as we have observed before, we would like to get a feeling for how long the approach will take. How many possibilities will have to be analyzed? This is a counting problem. We shall solve this counting problem by enumeration. It is easy to see from the graph of Figure 1.1 that Housing, Education, and Health must get different times. (Each one has a line joining it to the other two.) Similarly, Transportation must get a different time from Housing and Health. (Why?) Hence, since only three meeting times are available, Transportation must meet at the same time as Education. Similarly, Environment must meet at the same time as Housing. Finally, Finance cannot meet at the same time as Housing, and therefore as Environment, but could meet simultaneously with any of the other committees. Thus, there are only two possible meeting patterns. They are as follows.

Pattern 1. Transportation and Education meet at one time, Environment and Housing at a second time, and Finance and Health meet at the third time.

Pattern 2. Transportation, Education, and Finance meet at one time, Environment and Housing meet at a second time, and Health meets at the third time.

It follows that Table 1.7 gives all possible assignments of meeting times. In all, there are 12 possible. Our counting problem has been solved by enumerating all possibilities. (In Section 3.4.1 we do this counting another way.) It should be clear from Example 1.3 that enumeration could not always suffice for solving combinatorial problems. Indeed, if there are more committees and more possible meeting times, the problem we have been discussing gets completely out of hand.

Having succeeded in enumerating in our example, we can easily solve the optimization problem. Table 1.7 shows the number of committee chairs getting their first choice under each assignment. Clearly, assignment number 7 is the best from this point of view. Here, only the chair of the Environment committee does not get his or her first choice. For further reference on assignment of meeting times for state legislative committees, see Bodin and Friedman [1971]. For work on other scheduling problems where the schedule is repeated periodically (e.g., every week), see, for instance, Ahuja, Magnanti, and Orlin [1993], Baker [1976], Bartholdi, Orlin, and Ratliff [1980], Chrétienne [2000], Crama, *et al.* [2000], Karp and Orlin [1981], Orlin [1982], or Tucker [1975]. For surveys of various workforce scheduling algorithms, see Brucker [1998], Kovalëv, *et al.* [1989], or Tien and Kamiyama [1982]. ■

This book is organized around the three basic problems of combinatorics that we have been discussing. It has four parts. After an introductory part consisting of Chapters 2 to 4, the remaining three parts deal with these three problems: the counting problem (Chapters 5 to 8), the existence problem (Chapters 9 to 11), and the optimization problem (Chapters 12 and 13).

1.2 THE HISTORY AND APPLICATIONS OF COMBINATORICS[4]

The four examples described in Section 1.1 illustrate some of the problems with which combinatorics is concerned. They were chosen from a variety of fields to illustrate the variety of applications of combinatorics in modern times.

Although combinatorics has achieved its greatest impetus in modern times, it is an old branch of mathematics. According to legend, the Chinese Emperor Yu (in approximately 2200 B.C.) observed a magic square on the back of a divine tortoise. (A *magic square* is a square array of numbers in which the sum of all rows, all columns, and both diagonals is the same. An example of such a square is shown in Table 1.8. The reader might wish to find a different 3×3 magic square.)

Permutations or arrangements in order were known in China before 1100 B.C. The binomial expansion [the expansion of $(a+b)^n$] was known to Euclid about 300 B.C. for the case $n = 2$. Applications of the formula for the number of permutations of an n-element set can be found in an anonymous Hebrew work, *Sefer Yetzirah*, written between A.D. 200 and 500. The formula itself was known at least 2500 years ago. In A.D. 1100, Rabbi Ibn Ezra knew the formula for the number of combinations of n things taken r at a time, the binomial coefficient. Shortly thereafter, Chinese, Hindu, and Arab works began mentioning binomial coefficients in a primitive way.

In more modern times, the seventeenth-century scholars Pascal and Fermat pursued studies of combinatorial problems in connection with gambling—among other things, they figured out odds. (Pascal's famous triangle was in fact known to Chu

[4]For a more detailed discussion of the history of combinatorics, see Biggs, Lloyd, and Wilson [1995] or David [1962]. For the history of graph theory, see Biggs, Lloyd, and Wilson [1976].

Table 1.7: Possible Assignments of Meeting Times

Assignment number	Tuesday	Wednesday	Thursday	Number of committee chairs getting their first choice
1	Transportation-Education	Environment-Housing	Finance-Health	4
2	Transportation-Education	Finance-Health	Environment-Housing	3
3	Environment-Housing	Transportation-Education	Finance-Health	1
4	Environment-Housing	Finance-Health	Transportation-Education	0
5	Finance-Health	Transportation-Education	Environment-Housing	2
6	Finance-Health	Environment-Housing	Transportation-Education	2
7	Transportation-Education-Finance	Environment-Housing	Health	5
8	Transportation-Education-Finance	Health	Environment-Housing	4
9	Environment-Housing	Transportation-Education-Finance	Health	1
10	Environment-Housing	Health	Transportation-Education-Finance	0
11	Health	Transportation-Education-Finance	Environment-Housing	1
12	Health	Environment-Housing	Transportation-Education-Finance	1

Table 1.8: A Magic Square

4	9	2
3	5	7
8	1	6

Shih-Chieh in China in 1303.) The work of Pascal and Fermat laid the groundwork for probability theory; in the eighteenth century, Laplace defined probability in terms of number of favorable cases. Also in the eighteenth century, Euler invented graph theory in connection with the famous Königsberg bridge problem and Bernoulli published the first book presenting combinatorial methods, *Ars Conjectandi*. In the eighteenth and nineteenth centuries, combinatorial techniques were applied to study puzzles and games, by Hamilton and others. In the nineteenth century, Kirchhoff developed a graph-theoretical approach to electrical networks and Cayley developed techniques of enumeration to study organic chemistry. In modern times, the techniques of combinatorics have come to have far-reaching, significant applications in computer science, transportation, information processing, industrial planning, electrical engineering, experimental design, sampling, coding, genetics, political science, and a variety of other important fields. In this book we always keep the applications close at hand, remembering that they are not only a significant benefit derived from the development of the mathematical techniques, but they are also a stimulus to the continuing development of these techniques.

EXERCISES FOR CHAPTER 1

1. Find a 4×4 Latin square.
2. Find all possible 3×3 Latin squares.
3. Describe how to create an $n \times n$ Latin square.
4. (Liu [1972]) Suppose that we have two types of drugs to test simultaneously, such as headache remedies and fever remedies. In this situation, we might try to design an experiment in which each type of drug is tested using a Latin square design. However, we also want to make sure that, if at all possible, all combinations of headache and fever remedies are tested. For example, Table 1.9 shows two Latin square designs if we have 3 headache remedies and 3 fever remedies. Also shown in Table 1.9 is a third square, which lists as its i, j entry the i, j entries from both of the first two squares. We demand that each entry of this third square be different. This is not true in Table 1.9.
   (a) Find an example with 3 headache and 3 fever drugs where the combined square has the desired property.
   (b) Find another example with 4 headache and 4 fever drugs. (In Chapter 9 we observe that with 6 headache and 6 fever drugs, this is impossible. The existence problem has a negative solution.) *Note*: If you start with one Latin square design for the headache drugs and cannot find one for the fever drugs so that the combined square has the desired property, you should start with a different design for the headache drugs.
5. Show by enumeration that there are 14 bit strings of length at most 3.
6. Use enumeration to find the number of bit strings of length at most 4.
7. Suppose that we want to build a *trinary code* for the 26 letters of the alphabet, using strings in which each symbol is 0, 1, or -1.

Table 1.9: A Latin Square Design for Testing Headache Drugs 1, 2, and 3, a Latin Square Design for Testing Fever Drugs a, b, and c, and a Combination of the Two.[a]

Subject	Headache Drugs, Day 1	Day 2	Day 3	Fever Drugs, Day 1	Day 2	Day 3	Combination, Day 1	Day 2	Day 3
1	1	2	3	a	b	c	$1, a$	$2, b$	$3, c$
2	2	3	1	b	c	a	$2, b$	$3, c$	$1, a$
3	3	1	2	c	a	b	$3, c$	$1, a$	$2, b$

[a] The third square has as its i, j entry the headache drug and the fever drug shown in the i, j entries of the first two squares, respectively.

Table 1.10: Overlap Data[a]

	English	Calculus	History	Physics
English	0	1	0	0
Calculus	1	0	1	1
History	0	1	0	1
Physics	0	1	1	0

[a] The i, j entry is 1 if the ith and jth courses have a common member, and 0 otherwise.

(a) Could we encode all 26 letters using strings of length at most 2? Answer this question by enumeration.

(b) What about using strings of length exactly 3?

8. The genetic code embodied in the DNA molecule, a code we describe in Section 2.1, consists of strings of symbols, each of which is one of the four letters T, C, A, or G. Find by enumeration the number of different codewords or strings using these letters and having length 3 or less.

9. Suppose that in designing a gas pipeline network, we have 2 possible pipe sizes, small (S) and large (L). If there are 4 possible links, enumerate all possible pipeline networks. (A typical one could be abbreviated $LSLL$, where the ith letter indicates the size of the ith pipe.)

10. In Example 1.3, suppose that a computer could analyze as many as 100 billion different pipeline networks in a second, a 100-fold improvement over the speed we assumed in the text. Would this make a significant difference in our conclusions? Why? (Do a computation in giving your answer.)

11. Tables 1.10 and 1.11 give data of overlap in class rosters for several courses in a university.

    (a) Translate Table 1.10 into a graph as in Example 1.4.

Table 1.11: More Overlap Data[a]

	English	Calculus	History	Physics	Economics
English	0	1	0	0	0
Calculus	1	0	1	1	1
History	0	1	0	1	1
Physics	0	1	1	0	1
Economics	0	1	1	1	0

[a]The i, j entry is 1 if the ith and jth courses have a common member, and 0 otherwise.

Table 1.12: Acceptable Exam Times

Course	English	Calculus	History	Physics
Acceptable exam times	Thur. AM	Wed. AM Thur. AM	Tues. AM Wed. AM	Tues. AM Wed. AM

(b) Repeat part (a) for Table 1.11.

12. (a) Suppose that there are only two possible final examination times for the courses considered in Table 1.10. Is there an assignment of final exam times so that any two classes having a common member get a different exam time? If so, find such an assignment. If not, why not?

    (b) Repeat part (a) for Table 1.10 if there are three possible final exam times.

    (c) Repeat part (a) for Table 1.11 if there are three possible final exam times.

    (d) Repeat part (a) for Table 1.11 if there are four possible final exam times.

13. Suppose that in the situation of Table 1.5, the acceptable meeting times for Transportation are Tuesday and Thursday, for Education is Wednesday, and for all others are Tuesday, Wednesday, and Thursday. Show that no assignment of meeting times is possible.

14. (a) Suppose that in the situation of Table 1.10, the acceptable exam time schedules for each course are given in Table 1.12. Answer Exercise 12(b) if, in addition, each exam must be scheduled at an acceptable time.

    (b) Suppose that in the situation of Table 1.11, the acceptable exam time schedules for each course are given in Table 1.13. Answer Exercise 12(d) if, in addition, each exam must be scheduled at an acceptable time.

15. Suppose that there are three possible final exam times, Tuesday, Wednesday, and Thursday mornings. Suppose that each instructor of the courses listed in Table 1.10 requests Tuesday morning as a first choice for final exam time. What assignment (assignments) of exam times, if any exist, gives the largest number of instructors their first choice?

Table 1.13: More Acceptable Exam Times

Course	English	Calculus	History	Physics	Economics
Acceptable exam times	Wed. AM	Tues. AM Wed. AM	Tues. AM Wed. AM	Tues. AM Thur. AM	Mon. AM Wed. AM

REFERENCES FOR CHAPTER 1

AHUJA, R. K., MAGNANTI, T. L., and ORLIN, J. B., *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.

ALON, N., "Restricted Colorings of Graphs," in K. Walker (ed.), *Surveys in Combinatorics*, Proceedings 14th British Combinatorial Conference, London Math. Soc. Lecture Note Series, Vol. 187, Cambridge University Press, Cambridge, 1993, 1–33.

BAKER, K. R., "Workforce Allocation in Cyclical Scheduling Problems," *Oper. Res. Quart., 27* (1976), 155–167.

BARTHOLDI, J. J., III, ORLIN, J. B., and RATLIFF, H. D., "Cyclic Scheduling via Integer Programs with Circular Ones," *Oper. Res., 28* (1980), 1074–1085.

BIGGS, N. L., LLOYD, E. K., and WILSON, R. J., *Graph Theory 1736–1936*, Oxford University Press, London, 1976.

BIGGS, N. L., LLOYD, E. K., and WILSON, R. J., "The History of Combinatorics," in R. L. Graham, M. Grötschel, and L. Lovász (eds.), *Handbook of Combinatorics*, Elsevier, Amsterdam, 1995, 2163–2198.

BODIN, L. D., and FRIEDMAN, A. J., "Scheduling of Committees for the New York State Assembly," Tech. Report USE No. 71–9, Urban Science and Engineering, State University of New York, Stony Brook, NY, 1971.

BRUCKER, P., *Scheduling Algorithms*, Springer-Verlag, Berlin, 1998.

CHRÉTIENNE, P., "On Graham's Bound for Cyclic Scheduling," *Parallel Comput., 26* (2000), 1163–1174.

CRAMA, Y., KATS, V., VAN DE KLUNDERT, J., and LEVNER, E., "Cyclic Scheduling in Robotic Flowshops," *Ann. Oper. Res., 96* (2000), 97–124.

DAVID, F. N., *Games, Gods, and Gambling*, Hafner Press, New York, 1962. (Reprinted by Dover, New York, 1998.)

FRANK, H., and FRISCH, I. T., "Network Analysis," *Sci. Amer., 223* (1970), 94–103.

HERZBERG, A. M., and STANTON, R. G., "The Relation Between Combinatorics and the Statistical Design of Experiments," *J. Combin. Inform. System Sci., 9* (1984), 217–232.

KARP, R. M., and ORLIN, J. B., "Parametric Shortest Path Algorithms with an Application to Cyclic Staffing," *Discrete Appl. Math., 3* (1981), 37–45.

KLEITMAN, D. J., "Comments on the First Two Days' Sessions and a Brief Description of a Gas Pipeline Network Construction Problem," in F. S. Roberts (ed.), *Energy: Mathematics and Models*, SIAM, Philadelphia, 1976, 239–252.

KOVALËV, M. YA., SHAFRANSKIJ, YA. M., STRUSEVICH, V. A., TANAEV, V. S., and TUZIKOV, A. V., "Approximation Scheduling Algorithms: A Survey," *Optimization, 20* (1989), 859–878.

KRATOCHVÍL, J., TUZA, Z., and VOIGT, M., "New Trends in the Theory of Graph Colorings: Choosability and List Coloring," in R. L. Graham, J. Kratochvíl, J.

Nešetřil, and F. S. Roberts (eds.), *Contemporary Trends in Discrete Mathematics*, DIMACS Series, Vol. 49, American Mathematical Society, Providence, RI, 1999, 183–197.

Liu, C. L., *Topics in Combinatorial Mathematics*, Mathematical Association of America, Washington, DC, 1972.

Orlin, J. B., "Minimizing the Number of Vehicles to Meet a Fixed Periodic Schedule: An Application of Periodic Posets," *Oper. Res., 30* (1982), 760–776.

Rothfarb, B., Frank, H., Rosenbaum, D. M., Steiglitz, K., and Kleitman, D. J., "Optimal Design of Offshore Natural-Gas Pipeline Systems," *Oper. Res., 18* (1970), 992–1020.

Tien, J. M., and Kamiyama, A., "On Manpower Scheduling Algorithms," SIAM *Rev., 24* (1982), 275–287.

Tucker, A. C., "Coloring a Family of Circular Arcs," SIAM *J. Appl. Math., 29* (1975), 493-502.

Zadeh, N., "Construction of Efficient Tree Networks: The Pipeline Problem," *Networks, 3* (1973), 1–32.

Chapter 2

Basic Counting Rules[1]

2.1 THE PRODUCT RULE

Some basic counting rules underlie all of combinatorics. We summarize them in this chapter. The reader who is already familiar with these rules may wish to review them rather quickly. This chapter also introduces a widely used tool for proving that a certain kind of arrangement or pattern exists. In reading this chapter the reader already familiar with counting may wish to concentrate on the variety of applications that may not be as familiar, many of which are returned to in later chapters.

Example 2.1 Bit Strings and Binary Codes (Example 1.2 Revisited) Let us return to our binary code example (Example 1.2), and ask again how many letters of the alphabet can be encoded if there are exactly 2 bits. Let us get the answer by drawing a tree diagram. We do that in Figure 2.1. There are 4 possible strings of 2 bits, as we noted before. The reader will observe that there are 2 choices for the first bit, and for each of these choices, there are 2 choices for the second bit, and 4 is 2×2. ■

Example 2.2 DNA The total of all the genetic information of an organism is its *genome*. It is convenient to think of the genome as one long deoxyribonucleic acid (DNA) molecule. (The genome is actually made up of pieces of DNA representing the individual chromosomes.) The DNA (or chromosomes) is composed of a string of building blocks known as nucleotides. The genome size can be expressed in terms of the total number of nucleotides. Since DNA is actually double-stranded with the two strands held together by virtue of pairings between specific bases (a base being one of the three subcomponents of a nucleotide), genome sizes are usually

[1]This chapter was written by Helen Marcus-Roberts, Fred S. Roberts, and Barry A. Tesman.

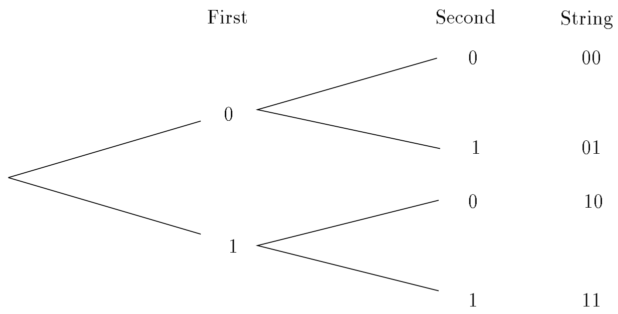


Figure 2.1: A tree diagram for counting the number of bit strings of length 2.

expressed in terms of base pairs (bp). Each base in a nucleotide is one of four possible chemicals: thymine, T; cytosine, C; adenine, A; guanine, G. The sequence of bases encodes certain genetic information. In particular, it determines long chains of amino acids which are known as proteins. There are 20 basic amino acids. A sequence of bases in a DNA molecule will encode one such amino acid. How long does a string of a DNA molecule have to be for there to be enough possible bases to encode 20 different amino acids? For example, can a 2-element DNA sequence encode for the 20 different basic amino acids? To answer this, we need to ask: How many 2-element DNA sequences are there? The answer to this question is again given by a tree diagram, as shown in Figure 2.2. We see that there are 16 possible 2-element DNA sequences. There are 4 choices for the first element, and for each of these choices, there are 4 choices for the second element; the reader will notice that 16 is 4×4. Notice that there are not enough 2-element sequences to encode for all 20 different basic amino acids. In fact, a sequence of 3 elements does the encoding in practice. A simple counting procedure has shown why at least 3 elements are needed. ■

The two examples given above illustrate the following basic rule.

PRODUCT RULE: If something can happen in n_1 ways, *and* no matter how the first thing happens, a second thing can happen in n_2 ways, then the two things together can happen in $n_1 \times n_2$ ways. More generally, if something can happen in n_1 ways, *and* no matter how the first thing happens, a second thing can happen in n_2 ways, *and* no matter how the first two things happen, a third thing can happen in n_3 ways, *and* ..., then all the things together can happen in

$$n_1 \times n_2 \times n_3 \times \cdots$$

ways.

Returning to bit strings, we see immediately by the product rule that the number of strings of exactly 3 bits is given by $2 \times 2 \times 2 = 2^3 = 8$ since there are two choices for the first bit (0 or 1), and no matter how it is chosen, there are two choices for

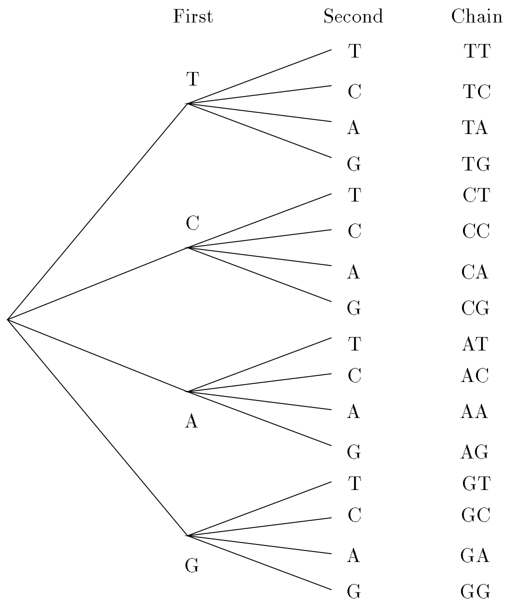


Figure 2.2: A tree diagram for counting the number of 2-element DNA sequences.

the second bit, and no matter how the first 2 bits are chosen, there are two choices for the third bit. Similarly, in the pipeline problem of Example 1.3, if there are 7 choices of pipe size for each of 3 links, there are

$$7 \times 7 \times 7 = 7^3 = 343$$

different possible networks. If there are 40 links, there are

$$7 \times 7 \times \cdots \times 7 = 7^{40}$$

different possible networks. Note that by our observations in Chapter 1, it is infeasible to count the number of possible pipeline networks by enumerating them (listing all of them). Some method of counting other than enumeration is needed. The product rule gives such a method. In the early part of this book, we shall be concerned with such simple methods of counting.

Next, suppose that A is a set of a objects and B is a set of b objects. Then the number of ways to pick one object from A and then one object from B is $a \times b$. This statement is a more precise version of the product rule.

To give one final example, the number of 3-element DNA sequences is

$$4 \times 4 \times 4 = 4^3 = 64.$$

That is why there are enough different 3-element sequences to encode for all 20 different basic amino acids; indeed, several different chains encode for the same

Table 2.1: The Number of Possible DNA Sequences for Various Organisms

Phylum	Genus and Species	Genome size (base pairs)	Number of possible sequences
Algae	*P. salina*	6.6×10^5	$4^{6.6 \times 10^5} > 10^{3.97 \times 10^5}$
Mycoplasma	*M. pneumoniae*	1.0×10^6	$4^{1.0 \times 10^6} > 10^{6.02 \times 10^5}$
Bacterium	*E. coli*	4.2×10^6	$4^{4.2 \times 10^6} > 10^{2.52 \times 10^6}$
Yeast	*S. cerevisiae*	1.3×10^7	$4^{1.3 \times 10^7} > 10^{7.82 \times 10^6}$
Slime mold	*D. discoideum*	5.4×10^7	$4^{5.4 \times 10^7} > 10^{3.25 \times 10^7}$
Nematode	*C. elegans*	8.0×10^7	$4^{8.0 \times 10^7} > 10^{4.81 \times 10^7}$
Insect	*D. melanogaster*	1.4×10^8	$4^{1.4 \times 10^8} > 10^{8.42 \times 10^7}$
Bird	*G. domesticus*	1.2×10^9	$4^{1.2 \times 10^9} > 10^{7.22 \times 10^8}$
Amphibian	*X. laevis*	3.1×10^9	$4^{3.1 \times 10^9} > 10^{1.86 \times 10^9}$
Mammal	*H. sapiens*	3.3×10^9	$4^{3.3 \times 10^9} > 10^{1.98 \times 10^9}$

Source: Lewin [2000].

amino acid—this is different from the situation in Morse code, where strings of up to 4 bits are required to encode for all 26 letters of the alphabet but not every possible string is used. In Section 2.9 we consider Gamow's [1954a,b] suggestion about which 3-element sequences encode for the same amino acid.

Continuing with DNA molecules, we see that the number of sequences of 4 bases is 4^4, the number with 100 bases is 4^{100}. How long is a full-fledged DNA molecule? Some answers are given in Table 2.1. Notice that in slime mold (*D. discoideum*), the genome has 5.4×10^7 bases or base pairs. Thus, the number of such sequences is

$$4^{5.4 \times 10^7},$$

which is greater than

$$10^{3.2 \times 10^7}.$$

This number is 1 followed by 3.2×10^7 zeros or 32 million zeros! It is a number that is too large to comprehend. Similar results hold for other organisms. By a simple counting of all possibilities, we can understand the tremendous possible variation in genetic makeup. It is not at all surprising, given the number of possible DNA sequences, that there is such an amazing variety in nature, and that two individuals are never the same. It should be noted once more that given the tremendous magnitude of the number of possibilities, it would not have been possible to count these possibilities by the simple expedient of enumerating them. It was necessary to develop rules or procedures for counting, which counted the number of possibilities without simply listing them. That is one of the three basic problems in combinatorics: developing procedures for counting without enumerating.

As large as the number of DNA sequences is, it has become feasible, in part due to the use of methods of combinatorial mathematics, to "sequence" and "map"

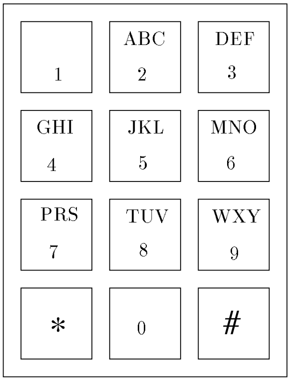


Figure 2.3: A telephone pad.

genomes of different organisms, including humans. A *gene* is a strip of DNA that carries the code for making a particular protein. "Mapping" the genome would require localizing each of its genes; "sequencing" it would require determining the exact order of the bases making up each gene. In humans, this involves approximately 100,000 genes, each with a thousand or more bases. For more on the use of combinatorial mathematics in mapping and sequencing the genome, see Clote and Backofen [2000], Congress of the United States [1988], Farach-Colton, *et al.* [1999], Gusfield [1997], Lander and Waterman [1995], Pevzner [2000], Setubal and Meidanis [1997], or Waterman [1995].

Example 2.3 Telephone Numbers At one time, a local telephone number was given by a sequence of two letters followed by five numbers. How many different telephone numbers were there? Using the product rule, one is led to the answer:

$$26 \times 26 \times 10 \times 10 \times 10 \times 10 \times 10 = 26^2 \times 10^5.$$

While the count is correct, it doesn't give a good answer, for two letters on the same place on the pad led to the same telephone number. The reader might wish to envision a telephone pad. (A rendering of one is given in Figure 2.3.) There are three letters on all digits, except that 1 and 0 have no letters. Hence, letters A, B, and C were equivalent; so were W, X, and Y; and so on. There were, in effect, only 8 different letters. The number of different telephone numbers was therefore

$$8^2 \times 10^5 = 6.4 \times 10^6.$$

Thus, there were a little over 6 million such numbers. In the 1950s and 1960s, most local numbers were changed to become simply seven-digit numbers, with the restriction that neither of the first two digits could be 0 or 1. The number of telephone numbers was still $8^2 \times 10^5$. Direct distance dialing was accomplished by adding a three-digit area code. The area code could not begin with a 0 or 1, and it had to have 0 or 1 in the middle. Using these restrictions, we compute that the number of possible telephone numbers was

$$8 \times 2 \times 10 \times 8^2 \times 10^5 = 1.024 \times 10^9.$$

Table 2.2: Two Switching Functions

Bit string x	$S(x)$	$T(x)$
00	1	0
01	0	0
10	0	1
11	1	1

That was enough to service over 1 billion customers. To service even more customers, direct distance dialing was changed to include a starting 1 as an 11th digit for long-distance calls. This freed up the restriction that an area code must have a 0 or 1 in the middle. The number of telephone numbers grew to

$$1 \times 8 \times 10 \times 10 \times 8^2 \times 10^5 = 5.12 \times 10^9.$$

With increasingly better technology, the telecommunications industry could boast that with the leading 1, there are no restrictions on the next 10 digits. Thus, there are now 10^{10} possible telephone numbers. However, demand continues to increase at a very fast pace (e.g., multiple lines, fax machines, cellular phones, pagers, etc.). What will we do when 10^{10} numbers are not enough? ■

Example 2.4 Switching Functions Let B_n be the set of all bit strings of length n. A *switching function (Boolean function) of n variables* is a function that assigns to each bit string of length n a number 0 or 1. For instance, let $n = 2$. Then $B_2 = \{00, 01, 10, 11\}$. Two switching functions S and T defined on B_2 are given in Table 2.2. The problem of making a detailed design of a digital computer usually involves finding a practical circuit implementation of certain functional behavior. A computer device implements a switching function of two, three, or four variables. Now every switching function can be realized in numerous ways by an electrical network of interconnections. Rather than trying to figure out from scratch an efficient design for a given switching function, a computer engineer would like to have a catalog that lists, for every switching function, an efficient network realization. Unfortunately, this seems at first to be an impractical goal. For how many switching functions of n variables are there? There are 2^n elements in the set B_n by a generalization of Example 2.1. Hence, by the product rule, there are $2 \times 2 \times \cdots \times 2$ different n-variable switching functions, where there are 2^n terms in the product. In total, there are 2^{2^n} different n-variable switching functions. Even the number of such functions for $n = 4$ is 65,536, and the number grows astronomically fast. Fortunately, by taking advantage of symmetries, we can consider certain switching functions equivalent as far as what they compute is concerned. Then we need not identify the best design for every switching function; we need do it only for enough switching functions so that every other switching function is equivalent to one of those for which we have identified the best design. While the first computers were being built, a team of researchers at Harvard painstakingly enumerated all possible switching functions of 4 variables, and determined which were equivalent. They discovered that it was possible to reduce every switching function to one of 222 types (Harvard Computation Laboratory Staff [1951]). In Chapter 8 we show how to derive results such as this from a powerful theorem due to George Pólya. For a more detailed discussion of switching functions, see Deo [1974, Ch. 12], Harrison [1965], Hill and Peterson [1968], Kohavi [1970], Liu [1977], Muroga [1990], Pattavina [1998], Prather [1976], or Stone [1973]. ■

Example 2.5 Food Allergies An allergist sees a patient who often develops a severe upset stomach after eating. Certain foods are suspected of causing the problem: tomatoes, chocolate, corn, and peanuts. It is not clear if the problem arises because of one of these foods or a combination of them acting together. The allergist tells the patient to try different combinations of these foods to see whether there is a reaction. How many different combinations must be tried? Each food can be absent or present. Thus, there are $2 \times 2 \times 2 \times 2 = 2^4 = 16$ possible combinations. In principle, there are 2^{2^4} possible manifestations of food allergies based on these four foods; each possible combination of foods can either bring forth an allergic reaction or not. Each person's individual sensitivity to combinations of these foods corresponds to a switching function $S(x_1, x_2, x_3, x_4)$ where x_1 is 1 if there are tomatoes in the diet, x_2 is 1 if there is chocolate in the diet, x_3 is 1 if there is corn in the diet, x_4 is 1 if there are peanuts in the diet. For instance, a person who develops an allergic reaction any time tomatoes are in the diet or any time both corn and peanuts are in the diet would demonstrate the switching function S which has $S(1,0,0,0) = 1$, $S(1,1,0,0) = 1$, $S(0,0,1,1) = 1$, $S(0,1,1,0) = 0$, and so on. In practice, it is impossible to know the value of a switching function on all possible bit strings if the number of variables is large; there are just too many possible bit strings. Then the practical problem is to develop methods to guess the value of a switching function that is only partially defined. There is much recent work on this problem. See, for example, Boros, *et al.* [1995], Boros, Ibaraki, and Makino [1998], Crama, Hammer, and Ibaraki [1988], and Ekin, Hammer, or Kogan [2000]. Similar cause-and-effect problems occur in diagnosing failure of a complicated electronic system given a record of failures when certain components fail (we shall have more to say about this in Example 2.21) and in teaching a robot to maneuver in an area filled with obstacles where an obstacle might appear as a certain pattern of dark or light pixels and in some situations the pattern of pixels corresponds to an object and in others it does not. For other applications, see Boros, *et al.* [2000]. ■

EXERCISES FOR SECTION 2.1[2]

1. The population of Carlisle, Pennsylvania, is about 20,000. If each resident has three initials, is it true that there must be at least two residents with the same initials? Give a justification of your answer.

[2] *Note to reader:* In the exercises in Chapter 2, exercises after each section can be assumed to use techniques of some previous (nonoptional) section, not necessarily exactly the techniques just introduced. Also, there are additional exercises at the end of the chapter. Indeed, sometimes an exercise is included which does not make use of the techniques of the current section. To understand a new technique, one must understand when it does not apply as well as when it applies.

2. A library has 1,700,000 books, and the librarian wants to encode each using a codeword consisting of 3 letters followed by 3 numbers. Are there enough codewords to encode all 1,700,000 books with different codewords?
3. (a) Continuing with Exercise 7 of Chapter 1, compute the maximum number of strings of length at most 3 in a trinary code.

   (b) Repeat for length at most 4.

   (c) Repeat for length exactly 4, but beginning with a 0 or 1.
4. In our discussion of telephone numbers, suppose that we maintain the original restrictions on area code as in Example 2.3. Suppose that we lengthen the local phone number, allowing it to be any eight-digit number with the restriction that none of the first three digits can be 0 or 1. How many local phone numbers are there? How many phone numbers are there including area code?
5. If we want to use bit strings of length at most n to encode not only all 26 letters of the alphabet, but also all 10 decimal digits, what is the smallest number n that works? (What is n for Morse code?)
6. How many $m \times n$ matrices are there each of whose entries is 0 or 1?
7. A musical band has to have at least one member. It can contain at most one drummer, at most one pianist, at most one bassist, at most one lead singer, and at most two background singers. How many possible bands are there if we consider any two drummers indistinguishable, and the same holds true for the other categories, and hence call two bands the same if they have the same number of members of each category? Justify your answer.
8. How many nonnegative integers less than 1 million contain the digit 2?
9. Enumerate all switching functions of 2 variables.
10. If a function assigns 0 or 1 to each switching function of n variables, how many such functions are there?
11. A switching function S is called *self-dual* if the value S of a bit string is unchanged when 0's and 1's are interchanged. For instance, the function S of Table 2.2 is self-dual, but the function T of that table is not. How many self-dual switching functions of n variables are there?
12. (Stanat and McAllister [1977]) In some computers, an integer (positive or negative) is represented by using bit strings of length p. The last bit in the string represents the sign, and the first $p-1$ bits are used to encode the integer. What is the largest number of distinct integers that can be represented in this way for a given p? What if 0 must be one of these integers? (The sign of 0 is $+$ or $-$.)
13. (Stanat and McAllister [1977]) Every integer can be represented (nonuniquely) in the form $a \times 2^b$, where a and b are integers. The *floating-point representation* for an integer uses a bit string of length p to represent an integer by using the first m bits to encode a and the remaining $p-m$ bits to encode b, with the latter two encodings performed as described in Exercise 12.

    (a) What is the largest number of distinct integers that can be represented using the floating-point notation for a given p?

    (b) Repeat part (a) if the floating-point representation is carried out in such a way that the leading bit for encoding the number a is 1.

    (c) Repeat part (a) if 0 must be included.

14. When acting on loan applications it can be concluded, based on historical records, that loan applicants having certain combinations of features can be expected to repay their loans and those who have other combinations of features cannot. As their main features, suppose that a bank uses:

    Marital Status: Married, Single (never married), Single (previously married).

    Past Loan: Previous default, No previous default.

    Employment: Employed, Unemployed (within 1 year), Unemployed (more than 1 year).

    (a) How many different loan applications are possible when considering these features?

    (b) How many manifestations of loan repayment/default are possible when considering these features?

2.2 THE SUM RULE

We turn now to the second fundamental counting rule. Consider the following example.

Example 2.6 Congressional Delegations There are 100 senators and 435 members of the House of Representatives. A delegation is being selected to see the President. In how many different ways can such a delegation be picked if it consists of one senator *and* one representative? The answer, by the product rule, is

$$100 \times 435 = 43{,}500.$$

What if the delegation is to consist of one member of the Senate *or* one member of the House? Then there are

$$100 + 435 = 535$$

possible delegations. This computation illustrates the second basic rule of counting, the sum rule. ■

SUM RULE: If one event can occur in n_1 ways and a second event in n_2 (different) ways, then there are $n_1 + n_2$ ways in which *either* the first event *or* the second event can occur (*but not both*). More generally, if one event can occur in n_1 ways, a second event can occur in n_2 (different) ways, a third event can occur in n_3 (still different) ways, ..., then there are

$$n_1 + n_2 + n_3 + \cdots$$

ways in which (exactly) one of the events can occur.

In Example 2.6 we have italicized the words "and" and "or." These key words usually indicate whether the sum rule or the product rule is appropriate. The word "and" suggests the product rule; the word "or" suggests the sum rule.

Example 2.7 Draft Picks A professional football team has two draft choices to make and has limited the choice to 3 quarterbacks, 4 linebackers, and 5 wide receivers. To pick a quarterback and linebacker there are $3 \times 4 = 12$ ways, by the product rule. How many ways are there to pick two players if they must play different positions? You can pick either a quarterback and linebacker, quarterback and wide receiver, or linebacker and wide receiver. There are, by previous computation, 12 ways of doing the first. There are 15 ways of doing the second (why?) and 20 ways of doing the third (why?). Hence, by the sum rule, the number of ways of choosing the two players from different positions is

$$12 + 15 + 20 = 47.$$

■

Example 2.8 Variables in BASIC and JAVA The programming language BASIC (standing for Beginner's All-Purpose Symbolic Instruction Code) dates back to 1964. Variable names in early implementations of BASIC could either be a letter, a letter followed by a letter, or a letter followed by a decimal digit, that is, one of the numbers $0, 1, \ldots, 9$. How many different variable names were possible? By the product rule, there were $26 \times 26 = 676$ and $26 \times 10 = 260$ names of the latter two kinds, respectively. By the sum rule, there were $26+676+260 = 962$ variable names in all.

The need for more variables was but one reason for more advanced programming languages. For example, the JAVA programming language, introduced in 1995, has variable name lengths ranging from 1 to 65,535 characters. Each character can be a letter (uppercase or lowercase), an underscore, a dollar sign, or a decimal digit except that the first character cannot be a decimal digit. By using the sum rule, we see that the number of possible characters is $26 + 26 + 1 + 1 + 10 = 64$ except for the first character which has only $64 - 10 = 54$ possibilities. Finally, by using the sum and product rules, we see that the number of variable names is

$$54 \cdot 64^{65,534} + 54 \cdot 64^{65,533} + \cdots + 54 \cdot 64 + 54.$$

This certainly allows for more than enough variables.[3] ■

In closing this section, let us restate the sum rule this way. Suppose that A and B are disjoint sets and we wish to pick exactly one element, picking it from A *or* from B. Then the number of ways to pick this element is the number of elements in A plus the number of elements in B.

EXERCISES FOR SECTION 2.2

1. How many bit strings have length 3, 4, or 5?
2. A committee is to be chosen from among 8 scientists, 7 psychics, and 12 clerics. If the committee is to have two members of different backgrounds, how many such committees are there?

[3] The value of just the first term in the sum, $54 \cdot 64^{65,534}$, is approximately $8.527 \times 10^{118,367}$. Arguably, there are at least on the order of 10^{80} atomic particles in the universe (e.g., see Dembski [1998]).

3. How many numbers are there which have five digits, each being a number in $\{1, 2, \ldots, 9\}$, and either having all digits odd or having all digits even?
4. Each American Express card has a 15-digit number for computer identification purposes. If each digit can be any number between 0 and 9, are there enough different account numbers for 10 million credit-card holders? Would there be if the digits were only 0 or 1?
5. How many 5-letter words either start with d or do not have the letter d?
6. In how many ways can we get a sum of 3 or a sum of 4 when two dice are rolled?
7. Suppose that a pipeline network is to have 30 links. For each link, there are 2 choices: The pipe may be any one of 7 sizes and any one of 3 materials. How many different pipeline networks are there?
8. How many DNA chains of length 3 have no C's at all or have no T's in the first position?

2.3 PERMUTATIONS

In combinatorics we frequently talk about n-element sets, sets consisting of n distinct elements. It is convenient to call these *n-sets.* A *permutation* of an n-set is an arrangement of the elements of the set in order. It is often important to count the number of permutations of an n-set.

Example 2.9 Job Interviews Three people, Ms. Jordan, Mr. Harper, and Ms. Gabler, are scheduled for job interviews. In how many different orders can they be interviewed? We can list all possible orders, as follows:

1. Jordan, Harper, Gabler
2. Jordan, Gabler, Harper
3. Harper, Jordan, Gabler
4. Harper, Gabler, Jordan
5. Gabler, Jordan, Harper
6. Gabler, Harper, Jordan

We see that there are 6 possible orders. Alternatively, we can observe that there are 3 choices for the first person being interviewed. For each of these choices, there are 2 remaining choices for the second person. For each of these choices, there is 1 remaining choice for the third person. Hence, by the product rule, the number of possible orders is

$$3 \times 2 \times 1 = 6.$$

Each order is a permutation. We are asking for the number of permutations of a 3-set, the set consisting of Jordan, Harper, and Gabler.

If there are 5 people to be interviewed, counting the number of possible orders can still be done by enumeration; however, that is rather tedious. It is easier to observe that now there are 5 possibilities for the first person, 4 remaining possibilities for the second person, and so on, resulting in

$$5 \times 4 \times 3 \times 2 \times 1 = 120$$

Table 2.3: Values of $n!$ for n from 0 to 10

n	0	1	2	3	4	5	6	7	8	9	10
$n!$	1	1	2	6	24	120	720	5,040	40,320	362,880	3,628,800

possible orders in all. ■

The computations of Example 2.9 generalize to give us the following result: The number of permutations of an n-set is given by

$$n \times (n-1) \times (n-2) \times \cdots \times 1 = n!$$

In Example 1.1 we discussed the number of orders in which to take 5 different drugs. This is the same as the number of permutations of a 5-set, so it is $5! = 120$. To see once again why counting by enumeration rapidly becomes impossible, we show in Table 2.3 the values of $n!$ for several values of n.

The number 25!, to give an example, is already so large that it is incomprehensible. To see this, note that

$$25! \approx 1.55 \times 10^{25}.$$

A computer checking 1 billion permutations per second would require almost half a billion years to look at 1.55×10^{25} permutations.[4] In spite of the result above, there are occasions where it is useful to enumerate all permutations of an n-set. In Section 2.16 we present an algorithm for doing so.

The number $n!$ can be approximated by computing $s_n = \sqrt{2\pi n}(n/e)^n$. The approximation of $n!$ by s_n is called *Stirling's approximation*. To see how good the approximation is, note that it approximates 5! as $s_5 = 118.02$ and 10! as $s_{10} = 3{,}598{,}600$. (Compare these with the real values in Table 2.3.) The quality of the approximation is evidenced by the fact that the ratio of $n!$ to s_n approaches 1 as n approaches ∞ (grows arbitrarily large). (On the other hand, the difference $n! - s_n$ approaches ∞ as n approaches ∞.) For a proof, see an advanced calculus text such as Buck [1965].

EXERCISES FOR SECTION 2.3

1. List all permutations of

   (a) $\{1, 2, 3\}$ (b) $\{1, 2, 3, 4\}$

[4] To see why, note that there are approximately 3.15×10^7 seconds in a year. Thus, a computer checking 1 billion $= 10^9$ permutations per second can check $3.15 \times 10^7 \times 10^9 = 3.15 \times 10^{16}$ permutations in a year. Hence, the number of years required to check 1.55×10^{25} permutations is

$$\frac{1.55 \times 10^{25}}{3.15 \times 10^{16}} \approx 4.9 \times 10^8.$$

2. How many permutations of $\{1, 2, 3, 4, 5\}$ begin with 5?
3. How many permutations of $\{1, 2, \ldots, n\}$ begin with 1 and end with n?
4. Compute s_n and compare it to $n!$ if

   (a) $n = 4$ (b) $n = 6$ (c) $n = 8$

5. How many permutations of $\{1, 2, 3, 4\}$ begin with an odd number?
6. (a) How many permutations of $\{1, 2, 3, 4, 5\}$ have 2 in the second place?
   (b) How many permutations of $\{1, 2, \ldots, n\}$, $n \geq 3$, have 2 in the second place and 3 in the third place?
7. How many ways are there to rank five potential basketball recruits of different heights if the tallest one must be ranked first and the shortest one last?
8. (Cohen [1978])
   (a) In a six-cylinder engine, the even-numbered cylinders are on the left and the odd-numbered cylinders are on the right. A good firing order is a permutation of the numbers 1 to 6 in which right and left sides are alternated. How many possible good firing orders are there which start with a left cylinder?
   (b) Repeat for a $2n$-cylinder engine.
9. Ten job applicants have been invited for interviews, five having been told to come in the morning and five having been told to come in the afternoon. In how many different orders can the interviews be scheduled? Compare your answer to the number of different orders in which the interviews can be scheduled if all 10 applicants were told to come in the morning.

2.4 COMPLEXITY OF COMPUTATION

We have already observed that not all problems of combinatorics can be solved on the computer, at least not by enumeration. Suppose that a computer program implements an algorithm for solving a combinatorial problem. Before running such a program, we would like to know if the program will run in a "reasonable" amount of time and will use no more than a "reasonable" (or allowable) amount of storage or memory. The time or storage a program requires depends on the input. To measure how expensive a program is to run, we try to calculate a *cost function* or a *complexity function.* This is a function f that measures the cost, in terms of time required or storage required, as a function of the size n of the input problem. For instance, we might ask how many operations are required to multiply two square matrices of n rows and columns each. This number of operations is $f(n)$.

Usually, the cost of running a particular computer program on a particular machine will vary with the skill of the programmer and the characteristics of the machine. Thus there is a big emphasis in modern computer science on comparison of algorithms rather than programs, and on estimation of the complexity $f(n)$ of an algorithm, independent of the particular program or machine used to implement the algorithm. The desire to calculate complexity of algorithms is a major stimulus for the development of techniques of combinatorics.

Example 2.10 The Traveling Salesman Problem A salesman wishes to visit n different cities, starting and ending his business trip at the first city. He does not care in which order he visits the cities. What he does care about is to minimize the total cost of his trip. Assume that the cost of traveling from city i to city j is c_{ij}. The problem is to find an algorithm for computing the cheapest route, where the cost of a route is the sum of the c_{ij} for links used in the route. This is a typical combinatorial optimization problem.

For the traveling salesman problem, we shall be concerned with the enumeration algorithm: Enumerate all possible routes and calculate the cost of each route. We shall try to compute the complexity $f(n)$ of this algorithm, where n is the size of the input, that is, the number of cities. We shall assume that identifying a route and computing its cost is comparable for each route and takes 1 unit of time.

Now any route starting and ending at city 1 corresponds to a permutation of the remaining $n-1$ cities. Hence, there are $(n-1)!$ such routes, so $f(n) = (n-1)!$ units of time. We have already shown that this number can be extremely high. When n is 26 and $n-1$ is 25, we showed that $f(n)$ is so high that it is infeasible to perform this algorithm by computer. We return to the traveling salesman problem in Section 11.5. ∎

It is interesting to note that the traveling salesman problem occurs in many guises. Examples 2.11 to 2.16 give some of the alternative forms in which this problem has arisen in practice.

Example 2.11 The Automated Teller Machine (ATM) Problem Your bank has many ATMs. Each day, a courier goes from machine to machine to make collections, gather computer information, and so on. In what order should the machines be visited in order to minimize travel time? This problem arises in practice at many banks. One of the first banks to use a traveling salesman algorithm to solve it, in the early days of ATMs, was Shawmut Bank in Boston.[5] ∎

Example 2.12 The Phone Booth Problem Once a week, each phone booth in a region must be visited, and the coins collected. In what order should that be done in order to minimize travel time? ∎

Example 2.13 The Problem of Robots in an Automated Warehouse The warehouse of the future will have orders filled by a robot. Imagine a pharmaceutical warehouse with stacks of goods arranged in rows and columns. An order comes in for 10 cases of aspirin, six cases of shampoo, eight cases of Band-Aids, and so on. Each is located by row, column, and height. In what order should the robot fill the order in order to minimize the time required? The robot needs to be programmed to solve a traveling salesman problem. (See Elsayed [1981] and Elsayed and Stern [1983].) ∎

Example 2.14 A Problem of X-Ray Crystallography In x-ray crystallography, we must move a diffractometer through a sequence of prescribed angles. There

[5] This example is from Margaret Cozzens (personal communication).

is a cost in terms of time and setup for doing one move after another. How do we minimize this cost? (See Bland and Shallcross [1989].) ■

Example 2.15 Manufacturing In many factories, there are a number of jobs that must be performed or processes that must be run. After running process i, there is a certain setup cost before we can run process j; a cost in terms of time or money or labor of preparing the machinery for the next process. Sometimes this cost is small (e.g., simply making some minor adjustments) and sometimes it is major (e.g., requiring complete cleaning of the equipment or installation of new equipment). In what order should the processes be run to minimize total cost? (For more on this application, see Example 11.5 and Section 11.6.3.) ■

Example 2.16 Holes in Circuit Boards In 1993, Applegate, Bixby, Chvátal, and Cook (see http://www.cs.rutgers.edu/~chvatal/pcb3038.html) found the solution to the largest TSPLIB[6] traveling salesman problem solved up until that time. It had 3,038 cities and arose from a practical problem involving the most efficient order in which to drill 3,038 holes to make a circuit board (another traveling salesman problem application). For information about this, see Zimmer [1993].[7] ■

The traveling salesman problem is an example of a problem that has defied the efforts of researchers to find a "good" algorithm. Indeed, it belongs to a class of problems known as *NP-complete* or *NP-hard problems*, problems for which it is unlikely there will be a good algorithm in a very precise sense of the word *good*. We return to this point in Section 2.18, where we define NP-completeness briefly and define an algorithm to be a *good algorithm* if its complexity function $f(n)$ is bounded by a polynomial in n. Such an algorithm is called a *polynomial algorithm* (more precisely, a polynomial-time algorithm).

Example 2.17 Scheduling a Computer System[8] A computer center has n programs to run. Each program requires certain resources, such as a compiler, a number of processors, and an amount of memory per processor. We shall refer to the required resources as a *configuration* corresponding to the program. The conversion of the system from the ith configuration to the jth configuration has a cost associated with it, say c_{ij}. For instance, if two programs require a similar configuration, it makes sense to run them consecutively. The computer center would like to minimize the total costs associated with running the n programs. The fixed cost of running each program does not change with different orders of running the programs. The only things that change are the conversion costs c_{ij}. Hence, the center wants to find an order in which to run the programs such that the total

[6] The TSPLIB (http://www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95/) is a library of 110 instances of the traveling salesman problem.

[7] All instances in the TSPLIB library have been solved. The largest instance of the traveling salesman problem in TSPLIB consists of a tour through 85,900 cities in a VLSI (Very Large-Scale Integration) application. For a survey about the computational aspects of the traveling salesman problem, see Applegate, *et al.* [1998]. See also the Traveling Salesman Problem home page (http://www.tsp.gatech.edu/index.html).

[8] This example is due to Stanat and McAllister [1977].

conversion costs are minimized. Similar questions arise in many scheduling problems in operations research. We discuss them further in Example 11.5 and Section 11.6.3. As in the traveling salesman problem, the algorithm of enumerating all possible orders of running the programs is infeasible, for it clearly has a computational complexity of $n!$. [Why $n!$ and not $(n-1)!$?] Indeed, from a formal point of view, this problem and the traveling salesman problem are almost equivalent—simply replace cities by configurations. Any algorithm for solving one of these problems is readily translatable into an algorithm for solving the other problem. It is one of the major motivations for using mathematical techniques to solve real problems that we can solve one problem and then immediately have techniques that are applicable to a large number of other problems, which on the surface seem quite different. ■

Example 2.18 Searching Through a File In determining computational complexity, we do not always know exactly how long a computation will take. For instance, consider the problem of searching through a list of n keys (identification numbers) and finding the key of a particular person in order to access that person's file. Now it is possible that the key in question will be first in the list. However, in the *worst case*, the key will be last on the list. The cost of handling the worst possible case is sometimes used as a measure of computational complexity called the *worst-case complexity*. Here $f(n)$ would be proportional to n. On the other hand, another perfectly appropriate measure of computational complexity is the *average* cost of handling a case, the *average-case complexity*. Assuming that all cases are equally likely, this is computed by calculating the cost of handling each case, summing up these costs, and dividing by the number of cases. In our example, the average-case complexity is proportional to $(n+1)/2$, assuming that all keys are equally likely to be the object of a search, for the sum of the costs of handling the cases is given by $1+2+\cdots+n$. Hence, using a standard formula for this sum, we have

$$f(n) = \frac{1}{n}(1+2+\cdots+n) = \frac{1}{n}\frac{n(n+1)}{2} = \frac{n+1}{2}. \qquad ■$$

In Section 3.6 we discuss the use of binary search trees for storing files and argue that the computational complexity of finding a file with a given key can be reduced significantly by using a binary search tree.

EXERCISES FOR SECTION 2.4

1. If a computer could consider 1 billion orders a second, how many years would it take to solve the computer configuration problem of Example 2.17 by enumeration if n is 25?

2. If a computer could consider 100 billion orders a second instead of just 1 billion, how many years would it take to solve the traveling salesman problem by enumeration if $n = 26$? (Does the improvement in computer speed make a serious difference in conclusions based on footnote 4 on page 26?)

3. Consider the problem of scheduling n legislative committees in order for meetings in n consecutive time slots. Each committee chair indicates which time slot is his or her first choice, and we seek to schedule the meetings so that the number of chairs receiving their first choice is as large as possible. Suppose that we solve this problem by enumerating all possible schedules, and for each we compute the number of chairs receiving their first choice. What is the computational complexity of this procedure? (Make an assumption about the number of steps required to compute the number of chairs receiving their first choice.)
4. Suppose that there are n phone booths in a region and we wish to visit each of them twice, but not in two consecutive times. Discuss the computational complexity of a naive algorithm for finding an order of visits that minimizes the total travel time.
5. Solve the traveling salesman problem by enumeration if $n = 4$ and the cost c_{ij} is given in the following matrix:
$$(c_{ij}) = \begin{pmatrix} - & 1 & 8 & 11 \\ 16 & - & 3 & 6 \\ 4 & 9 & - & 11 \\ 8 & 3 & 2 & - \end{pmatrix}.$$
6. Solve the computer system scheduling problem of Example 2.17 if $n = 3$ and the cost of converting from the ith configuration to the jth is given by
$$(c_{ij}) = \begin{pmatrix} - & 8 & 11 \\ 12 & - & 4 \\ 3 & 6 & - \end{pmatrix}.$$
7. Suppose that it takes 3×10^{-9} seconds to examine each key in a list. If there are n keys and we search through them in order until we find the right one, find

   (a) the worst-case complexity (b) the average-case complexity
8. Repeat Exercise 7 if it takes 3×10^{-11} seconds to examine each key.
9. (Hopcroft [1981]) Suppose that L is a collection of bit strings of length n. Suppose that A is an algorithm which determines, given a bit string of length n, whether or not it is in L. Suppose that A always takes 2^n seconds to provide an answer. Then A has the same worst-case and average-case computational complexity, 2^n. Suppose that $\widehat{L}$ consists of all bit strings of the form
$$x_1x_2 \cdots x_nx_1x_2 \cdots x_n,$$
where $x_1x_2 \cdots x_n$ is in L. For instance, if $L = \{00, 10\}$, then $\widehat{L} = \{0000, 1010\}$. Consider the following Algorithm B for determining, given a bit string $y = y_1y_2 \cdots y_{2n}$ of length $2n$, whether or not it is in $\widehat{L}$. First, determine if y is of the form $x_1x_2 \cdots x_nx_1x_2 \cdots x_n$. This is easy to check. Assume for the sake of discussion that it takes essentially 0 seconds to answer this question. If y is not of the proper form, stop and say that y is not in $\widehat{L}$. If y is of the proper form, check if the first n digits of y form a bit string in L.

   (a) Compute the worst-case complexity of Algorithm B.

   (b) Compute the average-case complexity of Algorithm B.

   (c) Do your answers suggest that average-case complexity might not be a good measure? Why?

2.5 r-PERMUTATIONS

Given an n-set, suppose that we want to pick out r elements and arrange them in order. Such an arrangement is called an *r-permutation of the n-set.* $P(n, r)$ will count the number of r-permutations of an n-set. For example, the number of 3-letter words without repeated letters can be calculated by observing that we want to choose 3 different letters out of 26 and arrange them in order; hence, we want $P(26, 3)$. Similarly, if a student has 4 experiments to perform and 10 periods in which to perform them (each experiment taking one period to complete), the number of different schedules he can make for himself is $P(10, 4)$. Note that $P(n, r) = 0$ if $n < r$: There are no r-permutations of an n-set in this case. In what follows, it will usually be understood that $n \geq r$.

To see how to calculate $P(n, r)$, let us note that in the case of the 3-letter words, there are 26 choices for the first letter; for each of these there are 25 remaining choices for the second letter; and for each of these there are 24 remaining choices for the third letter. Hence, by the product rule,

$$P(26, 3) = 26 \times 25 \times 24.$$

In the case of the experiment schedules, we have 10 choices for the first experiment, 9 for the second, 8 for the third, and 7 for the fourth, giving us

$$P(10, 4) = 10 \times 9 \times 8 \times 7.$$

By the same reasoning, if $n \geq r$,[9]

$$P(n, r) = n \times (n - 1) \times (n - 2) \times \cdots \times (n - r + 1).$$

If $n > r$, this can be simplified as follows:

$$P(n, r) = \frac{[n \times (n-1) \times \cdots \times (n - r + 1)] \times [(n - r) \times (n - r - 1) \times \cdots \times 1]}{(n - r) \times (n - r - 1) \times \cdots \times 1}.$$

Hence, we obtain the result

$$P(n, r) = \frac{n!}{(n - r)!}. \tag{2.1}$$

We have derived (2.1) under the assumption $n > r$. It clearly holds for $n = r$ as well. (Why?)

Example 2.19 CD Player We buy a brand new CD player with many nice features. In particular, the player has slots labeled 1 through 5 for five CDs which it plays in that order. If we have 24 CDs in our collection, how many different ways can we load the CD player's slots for our listening pleasure? There are 24 choices

[9] This formula even holds if $n < r$. Why?

for the first slot, 23 choices for the second, 22 choices for the third, 21 choices for the fourth, and 20 choices for the fifth, giving us

$$P(24,5) = 24 \times 23 \times 22 \times 21 \times 20.$$

Alternatively, using Equation (2.1), we see again that

$$P(24,5) = \frac{24!}{(24-5)!} = \frac{24!}{19!} = 24 \times 23 \times 22 \times 21 \times 20. \qquad \blacksquare$$

EXERCISES FOR SECTION 2.5

1. Find:

   (a) $P(3,2)$ (b) $P(5,3)$ (c) $P(8,5)$ (d) $P(1,3)$

2. Let $A = \{1, 5, 9, 11, 15, 23\}$.

   (a) Find the number of sequences of length 3 using elements of A.

   (b) Repeat part (a) if no element of A is to be used twice.

   (c) Repeat part (a) if the first element of the sequence is 5.

   (d) Repeat part (a) if the first element of the sequence is 5 and no element of A is used twice.

3. Let $A = \{a, b, c, d, e, f, g, h\}$.

   (a) Find the number of sequences of length 4 using elements of A.

   (b) Repeat part (a) if no letter is repeated.

   (c) Repeat part (a) if the first letter in the sequence is b.

   (d) Repeat part (a) if the first letter is b and the last is d and no letters are repeated.

4. In how many different orders can we schedule the first five interviews if we need to schedule interviews with 20 job candidates?

5. If a campus telephone extension has four digits, how many different extensions are there with no repeated digits:

   (a) If the first digit cannot be 0?

   (b) If the first digit cannot be 0 and the second cannot be 1?

6. A typical combination[10] lock or padlock has 40 numbers on its dial, ranging from 0 to 39. It opens by turning its dial clockwise, then counterclockwise, then clockwise, stopping each time at specific numbers. How many different padlocks can a company manufacture?

---

[10] In Section 2.7 we will see that the term "combination" is not appropriate with regard to padlocks; "r-permutation" would be correct.

2.6 SUBSETS

Example 2.20 The Pizza Problem A pizza shop advertises that it offers over 500 varieties of pizza. The local consumer protection bureau is suspicious. At the pizza shop, it is possible to have on a pizza a choice of any combination of the following toppings:

pepperoni, mushrooms, peppers, olives, sausage,
anchovies, salami, onions, bacon.

Is the pizza shop telling the truth in its advertisements? We shall be able to answer this question with some simple applications of the product rule. ■

To answer the question raised in Example 2.20, let us consider the set $\{a, b, c\}$. Let us ask how many subsets there are of this set. The answer can be obtained by enumeration, and we find that there are 8 such subsets:

$$\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}.$$

The answer can also be obtained using the product rule. We think of building up a subset in steps. First, we think of either including element a or not. There are 2 choices. Then we either include element b or not. There are again 2 choices. Finally, we either include element c or not. There are again 2 choices. The total number of ways of building up the subset is, by the product rule,

$$2 \times 2 \times 2 = 2^3 = 8.$$

Similarly, the number of subsets of a 4-set is

$$2 \times 2 \times 2 \times 2 = 2^4 = 16,$$

and the number of subsets of an n-set is

$$\underbrace{2 \times 2 \times \cdots \times 2}_{n \text{ times}} = 2^n.$$

Do these considerations help with the pizza problem? We can think of a particular pizza as a subset of the set of toppings. Alternatively, we can think, for each topping, of either including it or not. Either way, we see that there are $2^9 = 512$ possible pizzas. Thus, the pizza shop has not advertised falsely.

EXERCISES FOR SECTION 2.6

1. Enumerate the 16 subsets of $\{a, b, c, d\}$.
2. A magazine subscription service deals with 35 magazines. A subscriber may order any number of them. The subscription service is trying to computerize its billing procedure and wishes to assign a different computer key (identification number) to two different people unless they subscribe to exactly the same magazines. How much storage is required; that is, how many different code numbers are needed?

3. If the pizza shop of Example 2.20 decides to always put onions and mushrooms on its pizzas, how many different varieties can the shop now offer?
4. Suppose that the pizza shop of Example 2.20 adds a new possible topping, sardines, but insists that each pizza either have sardines or have anchovies. How many possible varieties of pizza does the shop now offer?
5. If A is a set of 10 elements, how many nonempty subsets does A have?
6. If A is a set of 8 elements, how many subsets of more than one element does A have?
7. A *value function* on a set A assigns 0 or 1 to each subset of A.
   (a) If A has 3 elements, how many different value functions are there on A?
   (b) What if A has n elements?
8. In a simple game (see Section 2.15), every subset of players is identified as either winning or losing.
   (a) If there is no restriction on this identification, how many distinct simple games are there with 3 players?
   (b) With n players?

2.7 r-COMBINATIONS

An *r-combination of an n-set* is a selection of r elements from the set, which means that order does not matter. Thus, an r-combination is an r-element subset. $C(n, r)$ will denote the number of r-combinations of an n-set. For example, the number of ways to choose a committee of 3 from a set of 4 people is given by $C(4, 3)$. If the 4 people are Dewey, Evans, Grange, and Howe, the possible committees are

$$\{\text{Dewey, Evans, Grange}\}, \{\text{Howe, Evans, Grange}\},$$
$$\{\text{Dewey, Howe, Grange}\}, \{\text{Dewey, Evans, Howe}\}.$$

Hence, $C(4, 3) = 4$. We shall prove some simple theorems about $C(n, r)$. Note that $C(n, r)$ is 0 if $n < r$: There are no r-combinations of an n-set in this case. Henceforth, $n \geq r$ will usually be understood. It is assumed in all of the theorems in this section.

Theorem 2.1

$$P(n, r) = C(n, r) \times P(r, r).$$

Proof. An ordered arrangement of r objects out of n can be obtained by first choosing r objects [this can be done in $C(n, r)$ ways] and then ordering them [this can be done in $P(r, r) = r!$ ways]. The theorem follows by the product rule. Q.E.D.

Corollary 2.1.1

$$C(n, r) = \frac{n!}{r!(n - r)!}. \tag{2.2}$$

Proof.

$$C(n,r) = \frac{P(n,r)}{P(r,r)} = \frac{n!/(n-r)!}{r!/(r-r)!} = \frac{n!}{r!(n-r)!}.$$

Q.E.D.

Corollary 2.1.2

$$C(n,r) = C(n,n-r).$$

Proof.

$$C(n,r) = \frac{n!}{r!(n-r)!} = \frac{n!}{(n-r)!r!} = \frac{n!}{(n-r)![n-(n-r)]!} = C(n,n-r).$$

Q.E.D.

For an alternative "combinatorial" proof, see Exercise 20.

Note: The number

$$\frac{n!}{r!(n-r)!}$$

is often denoted by

$$\binom{n}{r}$$

and called a *binomial coefficient*. This is because, as we shall see below, this number arises in the binomial expansion (see Section 2.14). Corollary 2.1.2 states the result that

$$\binom{n}{r} = \binom{n}{n-r}.$$

In what follows we use $C(n,r)$ and $\binom{n}{r}$ interchangeably.

Theorem 2.2

$$C(n,r) = C(n-1,r-1) + C(n-1,r).$$

Proof. Mark one of the n objects with a $*$. The r objects can be selected either to include the object $*$ or not to include it. There are $C(n-1,r-1)$ ways to do the former, since this is equivalent to choosing $r-1$ objects out of the $n-1$ non-$*$ objects. There are $C(n-1,r)$ ways to do the latter, since this is equivalent to choosing r objects out of the $n-1$ non-$*$ objects. Hence, the sum rule yields the theorem. Q.E.D.

Note: This proof can be described as a "combinatorial" proof, i.e, relying on counting arguments. This theorem can also be proved by algebraic manipulation, using the formula (2.2). Here is such an "algebraic" proof.

Second Proof of Theorem 2.2.

$$\begin{aligned}
C(n-1,r-1)+C(n-1,r) &= \frac{(n-1)!}{(r-1)![(n-1)-(r-1)]!}+\frac{(n-1)!}{r![(n-1)-r]!}\\
&= \frac{(n-1)!}{(r-1)!(n-r)!}+\frac{(n-1)!}{r!(n-r-1)!}\\
&= \frac{r(n-1)!}{r!(n-r)!}+\frac{(n-r)(n-1)!}{r!(n-r)!}\\
&= \frac{r(n-1)!+(n-r)(n-1)!}{r!(n-r)!}\\
&= \frac{(n-1)![r+n-r]}{r!(n-r)!}\\
&= \frac{n!}{r!(n-r)!}\\
&= C(n,r).
\end{aligned}$$

Q.E.D.

Let us give some quick applications of our new formulas and our basic rules so far.

1. In the pizza problem (Example 2.20), the number of pizzas with exactly 3 different toppings is

$$C(9,3)=\frac{9!}{3!6!}=84.$$

2. The number of pizzas with at most 3 different toppings is, by the sum rule,

$$C(9,0)+C(9,1)+C(9,2)+C(9,3)=130.$$

3. If we have 6 drugs being tested in an experiment and we want to choose 2 of them to give to a particular subject, the number of ways in which we can do this is

$$C(6,2)=\frac{6!}{2!4!}=15.$$

4. If there are 7 possible meeting times and a committee must meet 3 times, the number of ways we can assign the meeting times is

$$C(7,3)=\frac{7!}{3!4!}=35.$$

5. The number of 5-member committees from a group of 9 people is

$$C(9,5)=126.$$

6. The number of 7-member committees from the U.S. Senate is
$$C(100, 7).$$

7. The number of delegations to the President consisting of 2 senators and 2 representatives is
$$C(100, 2) \times C(435, 2).$$

8. The number of 9-digit bit strings with 5 1's and 4 0's is
$$C(9, 5) = C(9, 4).$$
To see why, think of having 9 unknown digits and choosing 5 of them to be 1's (or 4 of them to be 0's).

A convenient method of calculating the numbers $C(n, r)$ is to use the array shown in Figure 2.4. The number $C(n, r)$ appears in the nth row, rth diagonal. Each element in a given position is obtained by summing the two elements in the row above it which are just to the left and just to the right. For example, $C(5, 2)$ is given by summing up the numbers 4 and 6, which are circled in Figure 2.4. The array of Figure 2.4 is called *Pascal's triangle*, after the famous French philosopher and mathematician Blaise Pascal. Pascal was one of the inventors of probability theory and discovered many interesting combinatorial techniques.

Why does Pascal's triangle work? The answer is that it depends on the relation
$$C(n, r) = C(n-1, r-1) + C(n-1, r). \tag{2.3}$$
This is exactly the relation that was proved in Theorem 2.2. The relation (2.3) is an example of a *recurrence relation.* We shall see many such relations later in the book, especially in Chapter 6, which is devoted entirely to this topic. Obtaining such relations allows one to reduce calculations of complicated numbers to earlier steps, and therefore allows the computation of these numbers in stages.

EXERCISES FOR SECTION 2.7

1. How many ways are there to choose 5 starters (independent of position) from a basketball team of 10 players?
2. How many ways can 7 award winners be chosen from a group of 50 nominees?
3. Compute:

   (a) $C(6, 3)$ (b) $C(7, 4)$ (c) $C(5, 1)$ (d) $C(2, 4)$
4. Find $C(n, 1)$.
5. Compute $C(5, 2)$ and check your answer by enumeration.
6. Compute $C(6, 2)$ and check your answer by enumeration.

	$r = 0$	$r = 1$	$r = 2$	$r = 3$	$r = 4$	$r = 5$	$r = 6$
$n = 0$	1						
$n = 1$	1	1					
$n = 2$	1	2	1				
$n = 3$	1	3	3	1			
$n = 4$	1	4	6	4	1		
$n = 5$	1	5	10	10	5	1	
$n = 6$	1	6	15	20	15	6	1

Figure 2.4: Pascal's triangle. The circled numbers are added to give $C(5, 2)$.

7. Check by computation that:

   (a) $C(7, 2) = C(7, 5)$ (b) $C(6, 4) = C(6, 2)$

8. Extend Figure 2.4 by adding one more row.
9. Compute $C(5, 3)$, $C(4, 2)$, and $C(4, 3)$ and verify that formula (2.3) holds.
10. Repeat Exercise 9 for $C(7, 5)$, $C(6, 4)$, and $C(6, 5)$.
11. (a) In how many ways can 8 blood samples be divided into 2 groups to be sent to different laboratories for testing if there are 4 samples in each group? Assume that the laboratories are distinguishable.

    (b) In how many ways can 8 blood samples be divided into 2 groups to be sent to different laboratories for testing if there are 4 samples in each group? Assume that the laboratories are indistinguishable.

    (c) In how many ways can the 8 samples be divided into 2 groups if there is at least 1 item in each group? Assume that the laboratories are distinguishable.
12. A company is considering 6 possible new computer systems and its systems manager would like to try out at most 3 of them. In how many ways can the systems manager choose the systems to be tried out?
13. (a) In how many ways can 10 food items be divided into 2 groups to be sent to different laboratories for purity testing if there are 5 items in each group?

    (b) In how many ways can the 10 items be divided into 2 groups if there is at least 1 item in each group?
14. How many 8-letter words with no repeated letters can be constructed using the 26 letters of the alphabet if each word contains 3, 4, or 5 vowels?
15. How many odd numbers between 1000 and 9999 have distinct digits?
16. A fleet is to be chosen from a set of 7 different make foreign cars and 4 different make domestic cars. How many ways are there to form the fleet if:

    (a) The fleet has 5 cars, 3 foreign and 2 domestic?

(b) The fleet can be any size (except empty), but it must have equal numbers of foreign and domestic cars?

(c) The fleet has 4 cars and 1 of them must be a Chevrolet?

(d) The fleet has 4 cars, 2 of each kind, and a Chevrolet and Honda cannot both be in the fleet?

17. (a) A computer center has 9 different programs to run. Four of them use the language C++ and 5 use the language JAVA. The C++ programs are considered indistinguishable and so are the JAVA programs. Find the number of possible orders for running the programs if:

    i. There are no restrictions.

    ii. The C++ programs must be run consecutively.

    iii. The C++ programs must be run consecutively and the JAVA programs must be run consecutively.

    iv. The languages must alternate.

    (b) Suppose that the cost of switching from a C++ configuration to a JAVA configuration is 10 units, the cost of switching from a JAVA configuration to a C++ configuration is 5 units, and there is no cost to switch from C++ to C++ or JAVA to JAVA. What is the most efficient (least cost) ordering in which to run the programs?

    (c) Repeat part (a) if the C++ programs are all distinguishable from each other and so are the JAVA programs.

18. A certain company has 30 female employees, including 3 in the management ranks, and 150 male employees, including 12 in the management ranks. A committee consisting of 3 women and 3 men is to be chosen. How many ways are there to choose the committee if:

    (a) It includes at least 1 person of management rank of each gender?

    (b) It includes at least 1 person of management rank?

19. Consider the identity
$$\binom{n}{m}\binom{m}{k} = \binom{n}{k}\binom{n-k}{m-k}.$$

    (a) Prove this identity using an "algebraic" proof.

    (b) Prove this identity using a "combinatorial" proof.

20. Give an alternative "combinatorial" proof of Corollary 2.1.2 by using the definition of $C(n,r)$.

21. How would you find the sum $\binom{n}{0}+\binom{n}{1}+\binom{n}{2}+\cdots+\binom{n}{n}$ from Pascal's triangle? Do so for $n = 2$, 3, and 4. Guess at the answer in general.

22. Show that
$$\binom{n}{0}+\binom{n+1}{1}+\cdots+\binom{n+r}{r} = \binom{n+r+1}{r}.$$

23. Prove the following identity (using a combinatorial proof if possible). The identity is called *Vandermonde's identity.*

$$\binom{n+m}{r} = \binom{n}{0}\binom{m}{r} + \binom{n}{1}\binom{m}{r-1} + \binom{n}{2}\binom{m}{r-2} + \cdots + \binom{n}{r}\binom{m}{0}.$$

24. Following Cohen [1978], define $\left\langle {n \atop r} \right\rangle$ to be $\binom{n+r-1}{r}$. Show that

$$\left\langle {n \atop r} \right\rangle = \left\langle {n \atop r-1} \right\rangle + \left\langle {n-1 \atop r} \right\rangle$$

(a) using an algebraic proof (b) using a combinatorial proof

25. If $\left\langle {n \atop r} \right\rangle$ is defined as in Exercise 24, show that

$$\left\langle {n \atop r} \right\rangle = \frac{n}{r}\left\langle {n+1 \atop r-1} \right\rangle = \frac{n+r-1}{r}\left\langle {n \atop r-1} \right\rangle.$$

26. A sequence of numbers $a_0, a_1, a_2, \ldots, a_n$ is called *unimodal* if for some integer t, $a_0 \leq a_1 \leq \cdots \leq a_t$ and $a_t \geq a_{t+1} \geq \cdots \geq a_n$. (Note that the entries in any row of Pascal's triangle increase for awhile and then decrease and thus form a unimodal sequence.)

(a) Show that if $a_0, a_1, a_2, \ldots, a_n$ is unimodal, t is not necessarily unique.

(b) Show that if $n > 0$, the sequence $\binom{n}{0}, \binom{n}{1}, \binom{n}{2}, \ldots, \binom{n}{n}$ is unimodal.

(c) Show that the largest entry in the sequence in part (b) is $\binom{n}{\lfloor n/2 \rfloor}$ where $\lfloor x \rfloor$ is the greatest integer less than or equal to x.

2.8 PROBABILITY

The history of combinatorics is closely intertwined with the history of the theory of probability. The theory of probability was developed to deal with uncertain events, events that might or might not occur. In particular, this theory was developed by Pascal, Fermat, Laplace, and others in connection with the outcomes of certain gambles. In his *Théorie Analytique des Probabilités*, published in 1812, Laplace defined probability as follows: The *probability* of an event is the number of possible outcomes whose occurrence signals the event divided by the total number of possible outcomes. For instance, suppose that we consider choosing a 2-digit bit string at random. There are 4 such strings, 00, 01, 10, and 11. What is the probability that the string chosen has a 0? The answer is $\frac{3}{4}$, because 3 of the possible outcomes signal the event in question, that is, have a 0, and there are 4 possible outcomes in all. This definition of Laplace's is appropriate only if all the possible outcomes are equally likely, as we shall quickly observe.

Let us make things a little more precise. We shall try to formalize the notion of probability by thinking of an *experiment* that produces one of a number of possible outcomes. The set of possible outcomes is called the *sample space*. An *event* corresponds to a subset of the set of outcomes, that is, of the sample space; it corresponds to those outcomes that signal that the event has taken place. An event's *complement* corresponds to those outcomes that signal that the event has *not* taken place. Laplace's definition says that if E is an event in the sample space S and E^c is the complement of E, then

$$\text{probability of } E = \frac{n(E)}{n(S)} \text{ and probability of } E^c = \frac{n(S) - n(E)}{n(S)} = 1 - \frac{n(E)}{n(S)},$$

where $n(E)$ is the number of outcomes in E and $n(S)$ is the number of outcomes in S. Note that it follows that the probability of E is a number between 0 and 1.

Let us apply this definition to a gambling situation. We toss a die—this is the experiment. We wish to compute the probability that the outcome will be an even number. The sample space is the set of possible outcomes, $\{1, 2, 3, 4, 5, 6\}$. The event in question is the set of all outcomes which are even, that is, the set $\{2, 4, 6\}$. Then we have

$$\text{probability of even } = \frac{n(\{2,4,6\})}{n(\{1,2,3,4,5,6\})} = \frac{3}{6} = \frac{1}{2}.$$

Notice that this result would not hold unless all the outcomes in the sample space were equally likely. If we have a weighted die that always comes up 1, the probability of getting an even number is not $\frac{1}{2}$ but 0.[11]

Let us consider a family with two children. What is the probability that the family will have at least one boy? There are three possibilities for such a family: It can have two boys, two girls, or a boy and a girl. Let us take the set of these three possibilities as our sample space. The first and third outcomes make up the event "having at least one boy," and hence, by Laplace's definition,

$$\text{probability of having at least one boy } = \frac{2}{3}.$$

Is this really correct? It is not. If we look at families with two children, more than $\frac{2}{3}$ of them have at least one boy. That is because there are four ways to build up a family of two children: we can have first a boy and then another boy, first a girl and then another girl, first a boy and then a girl, or first a girl and then a boy. Thus, there are more ways to have a boy and a girl than there are ways to have two boys, and the outcomes in our sample space were not equally likely. However, the

[11]It could be argued that the definition of probability we have given is "circular" because it depends on the notion of events being "equally likely," which suggests that we already know how to measure probability. This is a subtle point. However, we can make comparisons of things without being able to measure them, e.g., to say that this person and that person seem equally tall. The theory of measurement of probability, starting with comparisons of this sort, is described in Fine [1973] and Roberts [1976, 1979].

outcomes BB, GG, BG, and GB, to use obvious abbreviations, are equally likely,[12] so we can take them as our sample space. Now the event "having at least one boy" has 3 outcomes in it out of 4, and we have

$$\text{probability of having at least one boy } = \frac{3}{4}.$$

We shall limit computations of probability in this book to situations where the outcomes in the sample space are equally likely. Note that our definition of probability applies only to the case where the sample space is finite. In the infinite case, the Laplace definition obviously has to be modified. For a discussion of the not-equally-likely case and the infinite case, the reader is referred to almost any textbook on probability theory, for instance Feller [1968], Parzen [1992], or Ross [1997].

Let us continue by giving several more applications of our definition. Suppose that a family is known to have 4 children. What is the probability that half of them are boys? The answer is not $\frac{1}{2}$. To obtain the answer we observe that the sample space is all sequences of B's and G's of length 4; a typical such sequence is BGGB. How many such sequences have exactly 2 B's? There are 4 positions, and 2 of these must be chosen for B's. Hence, there are $C(4,2)$ such sequences. How many sequences are there in all? By the product rule, there are 2^4. Hence,

$$\text{probability that half are boys } = \frac{C(4,2)}{2^4} = \frac{6}{16} = \frac{3}{8}.$$

The reader might wish to write out all 16 possible outcomes and note the 6 that signal the event having exactly 2 boys.

Next, suppose that a fair coin is tossed 5 times. What is the probability that there will be at least 2 heads? The sample space consists of all possible sequences of heads and tails of length 5, that is, it consists of sequences such as HHHTH, to use an obvious abbreviation. How many such sequences have at least 2 heads? The answer is that $C(5,2)$ sequences have exactly 2 heads, $C(5,3)$ have exactly 3 heads, and so on. Thus, the number of sequences having at least 2 heads is given by

$$C(5,2) + C(5,3) + C(5,4) + C(5,5) = 26.$$

The total number of possible sequences is $2^5 = 32$. Hence,

$$\text{probability of having at least two heads } = \frac{26}{32} = \frac{13}{16}.$$

Example 2.21 Reliability of Systems Imagine that a system has n components, each of which can work or fail to work. Let x_i be 1 if the ith component works and 0 if it fails. Let the bit string $x_1x_2\cdots x_n$ describe the system. Thus, the bit string 0011 describes a system with four components, with the first two failing

[12] Even this statement is not quite accurate, because it is slightly more likely to have a boy than a girl (see Cummings [1997]). Thus, the four events we have chosen are not exactly equally likely. For example, BB is more likely than GG. However, the assertion is a good working approximation.

Table 2.4: The Switching Function F That is 1 if and Only if Two or Three Components of a System Work

$x_1x_2x_3$	111	110	101	100	011	010	001	000
$F(x_1x_2x_3)$	1	1	1	0	1	0	0	0

and the third and fourth working. Since many systems have built-in redundancy, the system as a whole can work even if some components fail. Let $F(x_1x_2\cdots x_n)$ be 1 if the system described by $x_1x_2\cdots x_n$ works and 0 if it fails. Then F is a function from bit strings of length n to $\{0,1\}$, that is, an n-variable switching function (Example 2.4). For instance, suppose that we have a highly redundant system with three identical components, and the system works if and only if at least two components work. Then F is given by Table 2.4. We shall study other specific examples of functions F in Section 3.2.4 and Exercise 22, Section 13.3. Suppose that components in a system are equally likely to work or not to work.[13] Then any two bit strings are equally likely to be the bit string $x_1x_2\cdots x_n$ describing the system. Now we may ask: What is the probability that the system works, that is, what is the probability that $F(x_1x_2\cdots x_n)=1$? This is a measure of the *reliability* of the system. In our example, 4 of the 8 bit strings, 111, 110, 101, and 011, signal the event that $F(x_1x_2x_3)=1$. Since all bit strings are equally likely, the probability that the system works is $\frac{4}{8}=\frac{1}{2}$. For more on this approach to system reliability, see Karp and Luby [1983] and Barlow and Proschan [1975].

The theory of reliability of systems has been studied widely for networks of all kinds: electrical networks, computer networks, communication networks, and transportation routing networks. For a general reference on the subject of reliability of networks, see Hwang, Monma, and Roberts [1991] or Ball, Colbourn, and Provan [1995]. ■

Before closing this section, we observe that some common statements about probabilities of events correspond to operations on the associated subsets. Thus, we have:

Probability that event E does not occur is the probability of E^c.
Probability that event E or event F occurs is the probability of $E\cup F$.
Probability that event E and event F occur is the probability of $E\cap F$.

It is also easy to see from the definition of probability that

$$\text{probability of } E^c = 1 - \text{ probability of } E. \tag{2.4}$$

If E and F are disjoint,

$$\text{probability of } E\cup F = \text{ probability of } E + \text{probability of } F, \tag{2.5}$$

[13] In a more general analysis, we would first estimate the probability p_i that the ith component works.

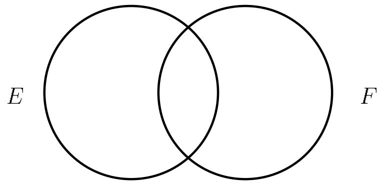


Figure 2.5: A Venn diagram related to Equation (2.6).

and, in general,

$$\begin{aligned}\text{probability of } E \cup F = & \text{ probability of } E + \text{probability of } F \\ & - \text{ probability of } E \cap F. \end{aligned} \tag{2.6}$$

To see why Equation (2.6) is true, consider the Venn diagram in Figure 2.5. Notice that when adding the probability of E and the probability of F, we are adding the probability of the intersection of E and F twice. By subtracting the probability of the intersection of E and F from the sum of their probabilities, Equation (2.6) is obtained.

To illustrate these observations, let us consider the die-tossing experiment. Then the probability of not getting a 3 is 1 minus the probability of getting a 3; that is, it is $1 - \frac{1}{6} = \frac{5}{6}$. What is the probability of getting a 3 or an even number? Since $E = \{3\}$ and $F = \{2, 4, 6\}$ are disjoint, (2.5) implies it is probability of E plus probability of $F = \frac{1}{6} + \frac{3}{6} = \frac{2}{3}$. Finally, what is the probability of getting a number larger than 4 or an even number? The event in question is the set $\{2, 4, 5, 6\}$, which has probability $\frac{4}{6} = \frac{2}{3}$. Note that this is not the same as the probability of a number larger than 4 plus the probability of an even number $= \frac{2}{6} + \frac{3}{6} = \frac{5}{6}$. This is because $E = \{5, 6\}$ and $F = \{2, 4, 6\}$ are not disjoint, for $E \cap F = \{6\}$. Applying (2.6), we have probability of $E \cup F = \frac{2}{6} + \frac{3}{6} - \frac{1}{6} = \frac{2}{3}$, which agrees with our first computation.

Example 2.22 Food Allergies (Example 2.5 Revisited) In Example 2.5 we studied the switching functions associated with food allergies brought on by some combination of four foods: tomatoes, chocolate, corn, and peanuts. We saw that there are a total of $2^4 = 16$ possible food combinations. We considered the situation where a person develops an allergic reaction any time tomatoes are in the diet or corn and peanuts are in the diet. What is the probability of not having such an allergic reaction?

To find the probability in question, we first calculate the probability that there is a reaction. Note that there is a reaction if the foods present are represented by the bit string $(1, y, z, w)$ or the bit string $(x, y, 1, 1)$, where x, y, z, w are (binary) 0-1 variables. Since there are three binary variables in the first type and two binary variables in the second type, there are $2^3 = 8$ different bit strings of the first type and $2^2 = 4$ of the second type. If there was no overlap between the two types, then (2.5) would allow us to merely add probabilities. However, there is overlap when x, z, and w are all 1. In this case y could be 0 or 1. Thus, by (2.6), the probability of a food reaction is

$$\frac{8}{16} + \frac{4}{16} - \frac{2}{16} = \frac{10}{16} = \frac{5}{8}.$$

By (2.4), the probability of no reaction is $1 - \frac{5}{8} = \frac{3}{8}$. ■

In the Example 2.22, enumeration of the possible combinations would be an efficient solution technique. However, if as few as 10 foods are considered, then enumeration would begin to get unwieldy. Thus, the techniques developed and used in this section are essential to avoid enumeration.

EXERCISES FOR SECTION 2.8

1. Are the outcomes in the following experiments equally likely?
   (a) A citizen of California is chosen at random and his or her town of residence is recorded.
   (b) Two drug pills and three placebos (sugar pills) are placed in a container and one pill is chosen at random and its type is recorded.
   (c) A snowflake is chosen at random and its appearance is recorded.
   (d) Two fair dice are tossed and the sum of the numbers appearing is recorded.
   (e) A bit string of length 3 is chosen at random and the sum of its digits is observed.
2. Calculate the probability that when a die is tossed, the outcome will be:
   (a) An odd number (b) A number less than or equal to 2
   (c) A number divisible by 3
3. Calculate the probability that a family of 3 children has:
   (a) Exactly 2 boys (b) At least 2 boys
   (c) At least 1 boy and at least 1 girl
4. If black hair, brown hair, and blond hair are equally likely (and no other hair colors can occur), what is the probability that a family of 3 children has at least two blondes?
5. Calculate the probability that in four tosses of a fair coin, there are at most three heads.
6. Calculate the probability that if a DNA chain of length 5 is chosen at random, it will have at least four A's.
7. If a card is drawn at random from a deck of 52, what is the probability that it is a king or a queen?
8. Suppose that a card is drawn at random from a deck of 52, the card is replaced, and then another card is drawn at random. What is the probability of getting two kings?
9. If a bit string of length 4 is chosen at random, what is the probability of having at least three 1's?
10. What is the probability that a bit string of length 3, chosen at random, does not have two consecutive 0's?

11. Suppose that a system has four independent components, each of which is equally likely to work or not to work. Suppose that the system works if and only if at least three components work. What is the probability that the system works?
12. Repeat Exercise 11 if the system works if and only if the fourth component works and at least two of the other components work.
13. A medical lab can operate only if at least one licensed x-ray technician is present and at least one phlebotomist. There are three licensed x-ray technicians and two phlebotomists, and each worker is equally likely to show up for work on a given day or to stay home. Assuming that each worker decides independently whether or not to come to work, what is the probability that the lab can operate?
14. Suppose that we have 10 different pairs of gloves. From the 20 gloves, 4 are chosen at random. What is the probability of getting at least one pair?
15. Use rules (2.4)–(2.6) to calculate the probability of getting, in six tosses of a fair coin:

    (a) Two heads or three heads (b) Two heads or two tails

    (c) Two heads or a head on the first toss

    (d) An even number of heads or at least nine heads

    (e) An even number of heads and a head on the first toss

16. Use the definition of probability to verify rules:

    (a) (2.4) (b) (2.5) (c) (2.6)

17. Repeat the problem in Example 2.22 when allergic reactions occur only in diets:

    (a) Containing either tomatoes and corn or chocolate and peanuts

    (b) Containing either tomatoes or all three other foods

2.9 SAMPLING WITH REPLACEMENT

In the National Hockey League (NHL), a team can either win (W), lose (L), or lose in overtime (OTL) each of its games. In an 82-game schedule, how many different seasons[14] can a particular team have? By the product rule, the answer is 3^{82}. There are three possibilities for each of the 82 games: namely, W, L, or OTL. We say that we are *sampling with replacement.* We are choosing an 82-permutation out of a 3-set, {W, L, OTL}, but with replacement of the elements in the set after they are drawn. Equivalently, we are allowing repetition. Let $P^R(m, r)$ be the number of r-permutations of an m-set, with replacement or repetition allowed. Then the product rule gives us

$$P^R(m, r) = m^r. \tag{2.7}$$

The number $P(m, r)$ counts the number of r-permutations of an m-set if we are sampling without replacement or repetition.

---

[14]Do not confuse "seasons" with "records." Records refer to the final total of wins, losses, and ties while seasons counts the number of different ways that each record could be attained.

We can make a similar distinction in the case of r-combinations. Let $C^R(m,r)$ be the number of r-combinations of an m-set if we sample with replacement or repetition. For instance, the 4-combinations of a 2-set $\{a,b\}$ if replacement is allowed are given by

$$\{a,a,a,a\},\{a,a,a,b\},\{a,a,b,b\},\{a,b,b,b\},\{b,b,b,b\}.$$

Thus, $C^R(2,4)=5$. We now state a formula for $C^R(m,r)$.

Theorem 2.3

$$C^R(m,r)=C(m+r-1,r).$$

We shall prove Theorem 2.3 at the end of this section. Here, let us illustrate it with some examples.

Example 2.23 The Chocolate Shoppe Suppose that there are three kinds of truffles available at a chocolate shoppe: cherry (c), orange (o), and vanilla (v). The store allows a customer to design a box of chocolates by choosing a dozen truffles. How many different truffle boxes are there? We can think of having a 3-set, $\{c,o,v\}$, and picking a 12-combination from it, with replacement. Thus, the number of truffle boxes is

$$C^R(3,12)=C(3+12-1,12)=C(14,12)=91. \qquad \blacksquare$$

Example 2.24 DNA Strings: Gamow's Encoding In Section 2.1 we studied DNA strings on the alphabet $\{\mathrm{A},\mathrm{C},\mathrm{G},\mathrm{T}\}$ and the minimum length required for such a string to encode for an amino acid. We noted that there are 20 different amino acids, and showed in Section 2.1 that there are only 16 different DNA strings of length 2, so a string of length at least 3 is required. But there are $4^3=64$ different strings of length 3. Gamow [1954a,b] suggested that there was a relationship between amino acids and the rhombus-shaped "holes" formed by the bases in the double helix structure of DNA. Each rhombus (see Figure 2.6) consists of 4 bases, with one base located at each corner of the rhombus. We will identify each rhombus with its 4-base sequence $xyzw$ that starts at the top of the rhombus and continues clockwise around the rhombus. For example, the rhombus in Figure 2.6 would be written GTTA. Due to base pairing in DNA, the fourth base, w, in the sequence is always fixed by the second base, y, in the sequence. If the second base is T, then the fourth base is A (and vice versa), or if the second base is G, then the fourth base is C (and vice versa).

Gamow proposed that rhombus $xyzw$ encodes the same amino acid as (a) $xwzy$ and (b) $zyxw$. Thus, GTTA, GATT, TTGA, and TAGT would all encode the same amino acid. If Gamow's suggestion were correct, how many amino acids could be encoded using 4-base DNA rhombuses? In the 4-base sequence $xyzw$ there are only 2 choices for the y-w pair: A-T (or equivalently, T-A) and G-C (or equivalently, C-G). Picking the other two bases is an application of Theorem 2.3. We have $m=4$ objects, we wish to choose $r=2$ objects, with replacement, and order doesn't matter. This can be done in

$$C(m+r-1,r)=C(5,2)=10$$

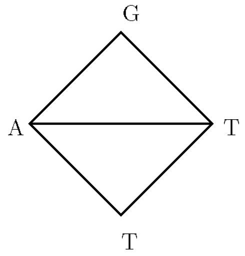


Figure 2.6: A 4-base DNA rhombus.

Table 2.5: Choosing a Sample of r Elements from a Set of m Elements

Order counts?	Repetition allowed?	The sample is called:	Number of ways to choose the sample	Reference
No	No	r-combination	$C(m,r) = \dfrac{m!}{r!(m-r)!}$	Corollary 2.1.1
Yes	No	r-permutation	$P(m,r) = \dfrac{m!}{(m-r)!}$	Eq. (2.1)
No	Yes	r-combination with replacement	$C^R(m,r) = C(m+r-1,r)$	Theorem 2.3
Yes	Yes	r-permutation with replacement	$P^R(m,r) = m^r$	Eq. (2.7)

different ways. Thus, it would be possible to encode $2 \times 10 = 20$ different amino acids using Gamow's 4-base DNA rhombuses which is precisely the correct number. Unfortunately, it was later discovered that this is not the way the coding works. See Golomb [1962] for a discussion. See also Griffiths, *et al.* [1996]. ∎

Our discussion of sampling with and without replacement is summarized in Table 2.5.

Example 2.25 Voting Methods In most elections in the United States, a number of candidates are running for an office and each registered voter may vote for the candidate of his or her choice. The winner of the election is the candidate with the highest vote total. (There could be multiple winners in case of a tie, but then tie-breaking methods could be used.) This voting method is called *plurality voting.* Suppose that 3 juniors are running for student class president of a class of 400 students. How many different results are possible if everyone votes? By "different results" we are referring to the number of different "patterns" of vote totals obtained by the 3 candidates. A *pattern* is a sequence (a_1, a_2, a_3) where a_i is the number of votes obtained by candidate i, $i = 1, 2, 3$. Thus, $(6, 55, 339)$ is different from $(55, 339, 6)$, and $(6, 55, 338)$ is not possible. [We will make no distinction among the voters (i.e., who voted for whom), only in the vote totals for each candidate.] Again, this is an application of Theorem 2.3. We have $m = 3$ objects and we wish to choose $r = 400$ objects, with replacement (obviously). This can be done in

$$C(m+r-1, r) = C(402, 400) = 80{,}601$$

different ways. This answer assumes that each voter voted. Exercise 10 addresses the question of vote totals when not all voters necessarily vote.

Another voting method, called *cumulative voting*, can be used in elections where more than one candidate needs to be elected. This is the case in many city council, board of directors, and school board elections. (Cumulative voting was used to elect the Illinois state legislature from 1870 to 1980.) With cumulative voting, voters cast as many votes as there are open seats to be filled and they are not limited to giving all of their votes to a single candidate. Instead, they can put multiple votes on one or more candidates. In such an election with p candidates, q open seats, and r voters, a total of qr votes are possible. The winners, analogous to the case of plurality voting, are the candidates with the q largest vote totals. Again consider the school situation of 3 candidates and 400 voters. However, now suppose that the students are not voting to elect a junior class president but two co-presidents. Under the cumulative voting method, how many different vote totals are possible? If, as in the plurality example above, each voter is required to vote for at least one candidate, then at least 400 votes and at most $2 \cdot 400 = 800$ votes must be cast. Consider the case of j votes being cast where $400 \leq j \leq 800$. By Theorem 2.3, there are

$$C^R(3,j) = \binom{3+j-1}{3-1} = \binom{2+j}{2}$$

different vote totals. Since j can range from 400 to 800, using the sum rule, there are a total of

$$\binom{2+400}{2} + \binom{2+401}{2} + \cdots + \binom{2+800}{2} = 75{,}228{,}001$$

different vote totals. Cumulative voting with votes not required and other voting methods are addressed in the exercises. For a general introduction to the methods and mathematics of voting see Aumann and Hart [1998], Brams [1994], Brams and Fishburn [1983], Farquharson [1969], or Kelly [1987]. ∎

Proof of Theorem 2.3.[15] Suppose that the m-set has elements $a_1, a_2, \ldots, a_m$. Then any sample of r of these objects can be described by listing how many a_1's are in it, how many a_2's, and so on. For instance, if $r = 7$ and $m = 5$, typical samples are $a_1a_1a_2a_3a_4a_4a_5$ and $a_1a_1a_1a_2a_4a_5a_5$. We can also represent these samples by putting a vertical line after the last a_i, for $i = 1, 2, \ldots, m-1$. Thus, these two samples would be written as $a_1a_1 \mid a_2 \mid a_3 \mid a_4a_4 \mid a_5$ and $a_1a_1a_1 \mid a_2 \mid\mid a_4 \mid a_5a_5$, where in the second case we have two consecutive vertical lines since there is no a_3. Now if we use this notation to describe a sample of r objects, we can omit the subscripts. For instance, $aa \mid aa \mid\mid\mid aaa$ represents $a_1a_1 \mid a_2a_2 \mid\mid\mid a_5a_5a_5$. Then the number of samples of r objects is just the number of different arrangements of r letters a and $m-1$ vertical lines. Such an arrangement has $m+r-1$ elements, and we determine the arrangement by choosing r positions for the a's. Hence, there are $C(m+r-1, r)$ such arrangements. Q.E.D.

[15]The proof may be omitted.

EXERCISES FOR SECTION 2.9

1. If replacement is allowed, find all:

   (a) 5-permutations of a 2-set
   (b) 2-permutations of a 3-set
   (c) 5-combinations of a 2-set
   (d) 2-combinations of a 3-set

2. Check your answers in Exercise 1 by using Equation (2.7) or Theorem 2.3.
3. If replacement is allowed, compute the number of:

   (a) 7-permutations of a 3-set
   (b) 7-combinations of a 4-set

4. In how many ways can we choose eight concert tickets if four concerts are available?
5. In how many different ways can we choose 12 microwave desserts if 5 different varieties are available?
6. Suppose that a codeword of length 8 consists of letters A, B, or C or digits 0 or 1, and cannot start with 1. How many such codewords are there?
7. How many DNA chains of length 6 have at least one of each base T, C, A, and G? Answer this question under the following assumptions:

   (a) Only the number of bases of a given type matter.
   (b) Order matters.

8. In an 82-game NHL season, how many different final records[16] are possible:

   (a) If a team can either win, lose, or overtime lose each game?
   (b) If overtime losses are not possible?

9. The United Soccer League in the United States has a shootout if a game is tied at the end of regulation. So there are wins, shootout wins, losses, or shootout losses. How many different 12-game seasons are possible?
10. Calculate the number of different vote totals, using the plurality voting method (see Example 2.25), when there are m candidates and n voters and each voter need not vote.
11. Calculate the number of different vote totals, using the cumulative voting method (see Example 2.25), when there are m candidates, n voters, l open seats, and each voter need not vote.

2.10 OCCUPANCY PROBLEMS[17]

2.10.1 The Types of Occupancy Problems

In the history of combinatorics and probability theory, problems of placing *balls* into *cells* or *urns* have played an important role. Such problems are called *occupancy problems.* Occupancy problems have numerous applications. In classifying

[16] See footnote on page 47.

[17] For a quick reading of this section, it suffices to read Section 2.10.1.

Table 2.6: The Distributions of Two Distinguishable Balls to Three Distinguishable Cells

		Distribution								
		1	2	3	4	5	6	7	8	9
Cell	1	ab			a	a		b	b	
	2		ab		b		a	a		b
	3			ab		b	b		a	a

types of accidents according to the day of the week in which they occur, the balls are the types of accidents and the cells are the days of the week. In cosmic-ray experiments, the balls are the particles reaching a Geiger counter and the cells are the counters. In coding theory, the possible distributions of transmission errors on k codewords are obtained by studying the codewords as cells and the errors as balls. In book publishing, the possible distributions of misprints on k pages are obtained by studying the pages as cells and the balls as misprints. In the study of irradiation in biology, the light particles hitting the retina correspond to balls, the cells of the retina to the cells. In coupon collecting, the balls correspond to particular coupons, the cells to the types of coupons. We shall return in various places to these applications. See Feller [1968, pp. 10–11] for other applications.

In occupancy problems, it makes a big difference whether or not we regard two balls as distinguishable and whether or not we regard two cells as distinguishable. For instance, suppose that we have two distinguishable balls, a and b, and three distinguishable cells, 1, 2, and 3. Then the possible distributions of balls to cells are shown in Table 2.6. There are nine distinct distributions. However, suppose that we have two indistinguishable balls. We can label them both a. Then the possible distributions to three distinguishable cells are shown in Table 2.7. There are just six of them. Similarly, if the cells are not distinguishable but the balls are, distributions 1–3 of Table 2.6 are considered the same: two balls in one cell, none in the others. Similarly, distributions 4–9 are considered the same: two cells with one ball, one cell with no balls. There are then just two distinct distributions. Finally, if neither the balls nor the cells are distinguishable, then distributions 1–3 of Table 2.7 are considered the same and distributions 4–6 are as well, so there are two distinct distributions.

It is also common to distinguish between occupancy problems where the cells are allowed to be empty and those where they are not. For instance, if we have two distinguishable balls and two distinguishable cells, then the possible distributions are given by Table 2.8. There are four of them. However, if no cell can be empty, there are only two, distributions 3 and 4 of Table 2.8.

The possible cases of occupancy problems are summarized in Table 2.9. The notation and terminology in the fourth column, which has not yet been defined, will be defined below. We shall now discuss the different cases.

Table 2.7: The Distributions of Two Indistinguishable Balls to Three Distinguishable Cells

		Distribution					
		1	2	3	4	5	6
Cell	1	aa			a	a	
	2		aa		a		a
	3			aa		a	a

Table 2.8: The Distributions of Two Distinguishable Balls to Two Distinguishable Cells

		Distribution			
		1	2	3	4
Cell	1	ab		a	b
	2		ab	b	a

2.10.2 Case 1: Distinguishable Balls and Distinguishable Cells

Case 1a is covered by the product rule: There are k choices of cells for each ball. If $k = 3$ and $n = 2$, we get $k^n = 9$, which is the number of distributions shown in Table 2.6. Case 1b is discussed in Section 2.10.4.

2.10.3 Case 2: Indistinguishable Balls and Distinguishable Cells[18]

Case 2a follows from Theorem 2.3, for we have the following result.

Theorem 2.4 The number of ways to distribute n indistinguishable balls into k distinguishable cells is $C(k + n - 1, n)$.

Proof. Suppose that the cells are labeled $C_1, C_2, \ldots, C_k$. A distribution of balls into cells can be summarized by listing for each ball the cell into which it goes. Then, a distribution corresponds to a collection of n cells with repetition allowed. For instance, in Table 2.7, distribution 1 corresponds to the collection $\{C_1, C_1\}$ and distribution 5 to the collection $\{C_1, C_3\}$. If there are four balls, the collection $\{C_1, C_2, C_3, C_3\}$ corresponds to the distribution that puts one ball into cell C_1, one ball into cell C_2, and two balls into cell C_3. Because a distribution corresponds to a collection $C_{i_1}, C_{i_2}, \ldots, C_{i_n}$, the number of ways to distribute the balls into cells is the same as the number of n-combinations of the k-set $\{C_1, C_2, \ldots, C_k\}$ in which repetition is allowed. This is given by Theorem 2.3 to be $C(k + n - 1, n)$. Q.E.D.

[18] The rest of Section 2.10 may be omitted.

Table 2.9: Classification of Occupancy Problems

	Distinguished balls?	Distinguished cells?	Can cells be empty?	Number of ways to place n balls into k cells:
Case 1				
1a	Yes	Yes	Yes	k^n
1b	Yes	Yes	No	$k!S(n,k)$
Case 2				
2a	No	Yes	Yes	$C(k+n-1,n)$
2b	No	Yes	No	$C(n-1,k-1)$
Case 3				
3a	Yes	No	Yes	$S(n,1)+S(n,2)+\cdots+S(n,k)$
3b	Yes	No	No	$S(n,k)$
Case 4				
4a	No	No	Yes	Number of partitions of n into k or fewer parts
4b	No	No	No	Number of partitions of n into exactly k parts

Theorem 2.4 is illustrated by Table 2.7. We have $k=3, n=2$, and $C(k+n-1,n)=C(4,2)=6$. The result in case 2b now follows from the result in case 2a. Given n indistinguishable balls and k distinguishable cells, we first place one ball in each cell. There is one way to do this. It leaves $n-k$ indistinguishable balls. We wish to place these into k distinguishable cells, with no restriction as to cells being nonempty. By Theorem 2.4 this can be done in

$$C(k+(n-k)-1,n-k)=C(n-1,k-1)$$

ways. We now use the product rule to derive the result for case 2b of Table 2.9. Note that $C(n-1,k-1)$ is 0 if $n<k$. There is no way to assign n balls to k cells with at least one ball in each cell.

2.10.4 Case 3: Distinguishable Balls and Indistinguishable Cells

Let us turn next to case 3b. Let $S(n,k)$ be defined to be the number of ways to distribute n distinguishable balls into k indistinguishable cells with no cell empty. The number $S(n,k)$ is called a *Stirling number of the second kind.*[19] In Section 5.5.3 we show that

$$S(n,k)=\frac{1}{k!}\sum_{i=0}^{k}(-1)^i\binom{k}{i}(k-i)^n. \tag{2.8}$$

[19] A Stirling number of the first kind exists and is found in other contexts. See Exercise 24 of Section 3.4.

To illustrate this result, let us consider the case $n = 2, k = 2$. Then

$$S(n, k) = S(2, 2) = \frac{1}{2}[2^2 - 2 \cdot 1^2 + 0] = 1.$$

There is only one distribution of two distinguishable balls a and b to two indistinguishable cells such that each cell has at least one ball: one ball in each cell.

The result in case 3a now follows from the result in case 3b by the sum rule. For to distribute n distinguishable balls into k indistinguishable cells with no cell empty, either one cell is not empty or two cells are not empty or .... The result in case 1b now follows also, since putting n distinguishable balls into k distinguishable cells with no cells empty can be accomplished by putting n distinguishable balls into k indistinguishable cells with no cells empty [which can be done in $S(n, k)$ ways] and then labeling the cells (which can be done in $k!$ ways). For instance, if $k = n = 2$, then by our previous computation, $S(2, 2) = 1$. Thus, the number of ways to put two distinguishable balls into two distinguishable cells with no cells empty is $2!S(2, 2) = 2$. This is the observation we made earlier from Table 2.8.

2.10.5 Case 4: Indistinguishable Balls and Indistinguishable Cells

To handle cases 4a and 4b, we define a *partition* of a positive integer n to be a collection of positive integers that sum to n. For instance, the integer 5 has the partitions

$$\{1, 1, 1, 1, 1\}, \{1, 1, 1, 2\}, \{1, 2, 2\}, \{1, 1, 3\}, \{2, 3\}, \{1, 4\}, \{5\}.$$

Note that $\{3, 2\}$ is considered the same as $\{2, 3\}$. We are interested only in what integers are in the collection, not in their order. The number of ways to distribute n indistinguishable balls into k indistinguishable cells is clearly the same as the number of ways to partition the integer n into at most k parts. This gives us the result in case 4a of Table 2.9. For instance, if $n = 5$ and $k = 3$, there are five possible partitions, all but the first two listed above. If $n = 2$ and $k = 3$, there are two possible partitions, $\{1, 1\}$ and $\{2\}$. This corresponds in Table 2.7 to the two distinct distributions: two cells with one ball in each or one cell with two balls in it. The result in case 4b of Table 2.9 follows similarly: The number of ways to distribute n indistinguishable balls into k indistinguishable cells with no cell empty is clearly the same as the number of ways to partition the integer n into exactly k parts. To illustrate this, if $n = 2$ and $k = 3$, there is no way.

We will explore partitions of integers briefly in the exercises and return to them in the exercises of Sections 5.3 and 5.4, where we approach them using the method of generating functions. For a detailed discussion of partitions, see most number theory books, for instance, Niven [1991] or Hardy and Wright [1980]. See also Berge [1971] or Riordan [1980].

2.10.6 Examples

We now give a number of examples, applying the results of Table 2.9. The reader should notice that whether or not balls or cells are distinguishable is often a matter of judgment, depending on the interpretation and in what we are interested.

Example 2.26 Hospital Deliveries Suppose that 80 babies are born in the month of September in a hospital and we record the day each baby is born. In how many ways can this event occur? The babies are the balls and the days are the cells. If we do not distinguish between 2 babies but do distinguish between days, we are in case 2, $n = 80, k = 30$, and the answer is given by $C(109, 80)$. The answer is given by $C(79, 29)$ if we count only the number of ways this can happen with each day having at least 1 baby. If we do not care about what day a particular number of babies is born but only about the number of days in which 2 babies are born, the number in which 3 are born, and so on, we are in case 4 and we need to consider partitions of the integer 80 into 30 or fewer parts. ■

Example 2.27 Coding Theory In coding theory, messages are first encoded into coded messages and then sent through a transmission channel. The channel may be a telephone line or radio wave. Due to noise or weak signals, errors may occur in the received codewords. The received codewords must then be decoded into the (hopefully) original messages. (An introduction to cryptography with an emphasis on coding theory is contained in Chapter 10.)

In monitoring the reliability of a transmission channel, suppose that we keep a record of errors. Suppose that 100 coded messages are sent through a transmission channel and 30 errors are made. In how many ways could this happen? The errors are the balls and the codewords are the cells. It seems reasonable to disregard the distinction between errors and concentrate on whether more errors occur during certain time periods of the transmission (because of external factors or a higher load period). Then codewords are distinguished. Hence, we are in case 2, and the answer is given by $C(129, 30)$. ■

Example 2.28 Gender Distribution Suppose that we record the gender of the first 1000 people to get a degree in computer science at a school. The people correspond to the balls and the two genders are the cells. We certainly distinguish cells. If we distinguish individuals, that is, if we distinguish between individual 1 being male and individual 2 being male, for example, then we are in case 1. However, if we are interested only in the number of people of each gender, we are in case 2. In the former case, the number of possible distributions is 2^{1000}. In the latter case, the number of possible distributions is given by $C(1001, 1000) = 1001$. ■

Example 2.29 Auditions A director has called back 24 actors for 8 different touring companies of a "one-man" Broadway show. (More than one actor may be chosen for a touring company in case of the need for a stand-in.) The actors correspond to the balls and the touring companies to the cells. If we are interested only in the actors who are in the same touring company, we can consider the balls distinguishable and the cells indistinguishable. Since each touring company needs at least one actor, no cell can be empty. Thus we are in case 3. The number of possible distributions is given by $S(24, 8)$. ■

Example 2.30 Statistical Mechanics In statistical mechanics, suppose that we have a system of t particles. Suppose that there are p different states or levels (e.g., energy levels), in which each of the particles can be. The state of the system is described by giving the distribution of particles to levels. In all, if the particles are distinguishable, there are p^t possible distributions. For instance, if we have 4 particles and 3 levels, there are $3^4 = 81$ different arrangements. One of these has particle 1 at level 1, particle 2 at level 3, particle 3 at level 2, and particle 4 at level 3. Another has particle 1 at level 2, particle 2 at level 1, and particles 3 and 4 at level 3. If we consider any distribution of particles to levels to be equally likely, then the probability of any given arrangement is $1/p^t$. In this case we say that the particles obey the *Maxwell-Boltzmann statistics.* Unfortunately, apparently no known physical particles exhibit these Maxwell-Boltzmann statistics; the p^t different arrangements are not equally likely. It turns out that for many different particles, in particular photons and nuclei, a relatively simple change of assumption gives rise to an empirically accurate model. Namely, suppose that we consider the particles as indistinguishable. Then we are in case 2: Two arrangements of particles to levels are considered the same if the same number of particles is assigned to the same level. Thus, the two arrangements described above are considered the same, as they each assign one particle to level 1, one to level 2, and two to level 3. By Theorem 2.4, the number of distinguishable ways to arrange t particles into p levels is now given by $C(p+t-1, t)$. If we consider any distribution of particles to levels to be equally likely, the probability of any one arrangement is

$$\frac{1}{C(p+t-1,t)}.$$

In this case, we say that the particles satisfy the *Bose-Einstein statistics.* A third model in statistical mechanics arises if we consider the particles indistinguishable but add the assumption that there can be no more than two particles at a given level. Then we get the *Fermi-Dirac statistics* (see Exercise 21). See Feller [1968] or Parzen [1992] for a more detailed discussion of all the cases we have described. ■

EXERCISES FOR SECTION 2.10

Note to the reader: When it is unclear whether balls or cells are distinguishable, you should state your interpretation, give a reason for it, and then proceed.

1. Write down all the distributions of:

   (a) 3 distinguishable balls a, b, c into 2 distinguishable cells $1, 2$

   (b) 4 distinguishable balls a, b, c, d into 2 distinguishable cells $1, 2$

(c) 2 distinguishable balls a, b into 4 distinguishable cells $1, 2, 3, 4$

(d) 3 indistinguishable balls a, a, a into 2 distinguishable cells $1, 2$

(e) 4 indistinguishable balls a, a, a, a into 2 distinguishable cells $1, 2$

(f) 2 indistinguishable balls a, a into 4 distinguishable cells $1, 2, 3, 4$

2. In Exercise 1, which of the distributions are distinct if the cells are indistinguishable?
3. Use the results of Table 2.9 to compute the number of distributions in each case in Exercise 1 and check the result by comparing the distributions you have written down.
4. Repeat Exercise 3 if the cells are indistinguishable.
5. Use the results of Table 2.9 to compute the number of distributions with no empty cell in each case in Exercise 1. Check the result by comparing the distributions you have written down.
6. Repeat Exercise 5 if the cells are indistinguishable.
7. Find all partitions of:

   (a) 4 (b) 7 (c) 8

8. Find all partitions of:

   (a) 9 into four or fewer parts (b) 11 into three or fewer parts

9. Compute:

   (a) $S(n, 0)$ (b) $S(n, 1)$ (c) $S(n, 2)$
   (d) $S(n, n-1)$ (e) $S(n, n)$

10. In checking the work of a proofreader, we look for 5 kinds of misprints in a textbook. In how many ways can we find 12 misprints?
11. In Exercise 10, suppose that we do not distinguish the types of misprints but we do keep a record of the page on which a misprint occurred. In how many different ways can we find 25 misprints in 75 pages?
12. In Example 2.27, suppose that we pinpoint 30 kinds of errors and we want to find out whether these errors tend to appear together, not caring in which codeword they appear together. In how many ways can we find 30 kinds of errors in 100 codewords if each kind of error is known to appear exactly once in some codeword?
13. An elevator with 9 passengers stops at 5 different floors. If we are interested only in the passengers who get off together, how many possible distributions are there?
14. If lasers are aimed at 5 tumors, how many ways are there for 10 lasers to hit? (You do not have to assume that each laser hits a tumor.)
15. A Geiger counter records the impact of 6 different kinds of radioactive particles over a period of time. How many ways are there to obtain a count of 30?
16. Find the number of ways to distribute 10 customers to 7 salesmen so that each salesman gets at least 1 customer.
17. Find the number of ways to pair off 10 students into lab partners.
18. Find the number of ways to assign 6 jobs to 4 workers so that each job gets a worker and each worker gets at least 1 job.

19. Find the number of ways to partition a set of 20 elements into exactly 4 subsets.
20. In Example 2.30, suppose that there are 8 photons and 4 energy levels, with 2 photons at each energy level. What is the probability of this occurrence under the assumption that the particles are indistinguishable (the Bose-Einstein case)?
21. Show that in Example 2.30, if particles are indistinguishable but no two particles can be at the same level, then there are $C(p, t)$ possible arrangements of t particles into p levels. (Assume that $t \leq p$.)
22. (a) Show by a combinatorial argument that
$$S(n, k) = kS(n-1, k) + S(n-1, k-1).$$
    (b) Use the result in part (a) to describe how to compute Stirling numbers of the second kind by a method similar to Pascal's triangle.
    (c) Apply your result in part (b) to compute $S(6, 3)$.
23. Show by a combinatorial argument that
$$S(n+1, k) = C(n, 0)S(0, k-1) + C(n, 1)S(1, k-1) + \cdots + C(n, n)S(n, k-1).$$
24. (a) If order counts in a partition, then $\{3, 2\}$ is different from $\{2, 3\}$. Find the number of partitions of 5 if order matters.
    (b) Find the number of partitions of 5 into exactly 2 parts where order matters.
    (c) Show that the number of partitions of n into exactly k parts where order matters is given by $C(n-1, k-1)$.
25. The *Bell number* B_n is the number of partitions of a set of n elements into nonempty, indistinguishable cells. Note that
$$B_n = S(n, 0) + S(n, 1) + \cdots + S(n, n).$$
Show that
$$B_n = \binom{n-1}{0} B_0 + \binom{n-1}{1} B_1 + \cdots + \binom{n-1}{n-1} B_{n-1}.$$

2.11 MULTINOMIAL COEFFICIENTS

2.11.1 Occupancy Problems with a Specified Distribution

In this section we consider the occupancy problem of distributing n distinguishable balls into k distinguishable cells. In particular, we consider the situation where we distribute n_1 balls into the first cell, n_2 into the second cell, ..., n_k into the kth cell. Let
$$C(n; n_1, n_2, \ldots, n_k)$$
denote the number of ways this can be done. This section is devoted to the study of the number $C(n; n_1, n_2, \ldots, n_k)$, which is sometimes also written as
$$\binom{n}{n_1, n_2, \ldots, n_k}$$
and called the *multinomial coefficient.*

Example 2.31 Campus Registration The university registrar's office is having a problem. It has 11 new students to squeeze into 4 sections of an introductory course: 3 in the first, 4 each in the second and third, and 0 in the fourth (that section is already full). In how many ways can this be done? The answer is $C(11;3,4,4,0)$. Now there are $C(11,3)$ choices for the first section; for each of these there are $C(8,4)$ choices for the second section; for each of these there are $C(4,4)$ choices for the third section; for each of these there are $C(0,0)$ choices for the fourth section. Hence, by the product rule, the number of ways to assign sections is

$$\begin{aligned} C(11;3,4,4,0) &= C(11,3)\times C(8,4)\times C(4,4)\times C(0,0) \\ &= \frac{11!}{3!8!}\times\frac{8!}{4!4!}\times\frac{4!}{4!0!}\times\frac{0!}{0!0!} = \frac{11!}{3!4!4!}, \end{aligned}$$

since $0! = 1$. Of course, $C(0,0)$ always equals 1, so the answer is equivalent to $C(11;3,4,4)$. Additionally, $C(4,4)=1$, so the answer is also equivalent to $C(11,3)\times C(8,4)$. The reason for this is that once the 3 students for the first section and 4 students for the second section have been chosen, there is only one way to choose the remaining 4 for the third section.

Note that if section assignments for 11 students are made at random, there are 4^{11} possible assignments: For each student, there are 4 choices of section. Hence, the probability of having 3 students in the first section, 4 each in the second and third sections, and 0 in the fourth is given by

$$\frac{C(11;3,4,4,0)}{4^{11}}.$$

In general, suppose that $\Pr(n;n_1,n_2,\ldots,n_k)$ denotes the probability that if n balls are distributed at random into k cells, there will be n_i balls in cell i, $i=1,2,\ldots,k$. Then

$$\Pr(n;n_1,n_2,\ldots,n_k) = \frac{C(n;n_1,n_2,\ldots,n_k)}{k^n}.$$

(Why?) Note that when calculating the multinomial coefficient, the acknowledgment of empty cells does not affect the calculation. This is because

$$C(n;n_1,n_2,\ldots,n_j,0,0,\ldots,0) = C(n;n_1,n_2,\ldots,n_j).$$

However, the probability of a multinomial distribution *is* affected by empty cells as the denominator is based on the number of cells, both empty and nonempty.

Continuing with our example, suppose that suddenly, spaces in the fourth section become available. The registrar's office now wishes to put 3 people each into the first, second, and third sections, and 2 into the fourth. In how many ways can this be done? Of the 11 students, 3 must be chosen for the first section; of the remaining 8 students, 3 must be chosen for the second section; of the remaining 5 students, 3 must be chosen for the third section; finally, the remaining 2 must be put into the fourth section. The total number of ways of making the assignments is

$$\begin{aligned} C(11;3,3,3,2) &= C(11,3)\times C(8,3)\times C(5,3)\times C(2,2) \\ &= \frac{11!}{3!8!}\times\frac{8!}{3!5!}\times\frac{5!}{3!2!}\times\frac{2!}{2!0!} = \frac{11!}{3!3!3!2!}. \end{aligned}$$ ■

Let us derive a formula for $C(n; n_1, n_2, \ldots, n_k)$. By reasoning analogous to that used in Example 2.31,

$$\begin{aligned} C(n; n_1, n_2, \ldots, n_k) &= C(n, n_1) \times C(n-n_1, n_2) \times C(n-n_1-n_2, n_3) \times \cdots \\ &\quad \times C(n-n_1-n_2-\cdots-n_{k-1}, n_k) \\ &= \frac{n!}{n_1!(n-n_1)!} \times \frac{(n-n_1)!}{n_2!(n-n_1-n_2)!} \times \frac{(n-n_1-n_2)!}{n_3!(n-n_1-n_2-n_3)!} \\ &\quad \times \cdots \times \frac{(n-n_1-n_2-\cdots-n_{k-1})!}{n_k!(n-n_1-n_2-\cdots-n_k)!} \\ &= \frac{n!}{n_1!n_2!\cdots n_k!(n-n_1-n_2-\cdots-n_k)!}. \end{aligned}$$

Since $n_1 + n_2 + \cdots + n_k = n$, and since $0! = 1$, we have the following result.

Theorem 2.5

$$C(n; n_1, n_2, \ldots, n_k) = \frac{n!}{n_1!n_2!\cdots n_k!}.$$

We now give several illustrations and applications of this theorem.

1. The number of 4-digit bit strings consisting of three 1's and one 0 is $C(4; 3, 1)$: Out of four places, we choose three for the digit 1 and one for the digit 0. Hence, the number of such strings is given by

   $$C(4; 3, 1) = \frac{4!}{3!1!} = 4.$$

   The four such strings are 1110, 1101, 1011, and 0111.

2. The number of 5-digit numbers consisting of two 2's, two 3's, and one 1 is

   $$C(5; 2, 2, 1) = \frac{5!}{2!2!1!} = 30.$$

3. Notice that $C(n; n_1, n_2) = C(n, n_1)$. Why should this be true?

4. An NHL hockey season consists of 82 games. The number of ways the season can end in 41 wins, 27 losses, and 14 overtime losses is

   $$C(82; 41, 27, 14) = \frac{82!}{41!27!14!}.$$

5. RNA is a messenger molecule whose links are defined from DNA. An RNA chain has, at each link, one of four bases. The possible bases are the same as those in DNA (Example 2.2), except that the base uracil (U) replaces the base thymine (T). How many possible RNA chains of length 6 are there consisting of 3 cytosines (C) and 3 adenines (A)? To answer this question, we think of

6 positions in the chain, and of dividing these positions into four sets, 3 into the C set, 3 into the A set, and 0 into the T and G sets. The number of ways this can be done is given by

$$C(6;3,3,0,0) = C(6;3,3) = 20.$$

6. There are 4^6 possible RNA chains of length 6. The probability of obtaining one with 3 C's and 3 A's if the RNA chain were produced at random (i.e., all possibilities equally likely) would be

$$\Pr(6;3,3,0,0) = \frac{C(6;3,3,0,0)}{4^6} = \frac{C(6;3,3)}{4^6} = \frac{20}{4096} = \frac{5}{1024} \approx .005.$$

Note that this is not $\Pr(6;3,3)$.

7. The number of 10-link RNA chains consisting of 3 A's, 2 C's, 2 U's, and 3 G's is

$$C(10;3,2,2,3) = 25{,}200.$$

8. The number of RNA chains as described in the previous example which end in AAG is

$$C(7;1,2,2,2) = 630,$$

since there are now only the first 7 positions to be filled, and two of the A's and one of the G's are already used up. Notice how knowing the end of a chain can reduce dramatically the number of possible chains. In the next section we see how, by judicious use of various enzymes which decompose RNA chains, we might further limit the number of possible chains until, by a certain amount of detective work, we can uncover the original RNA chain without actually observing it.

2.11.2 Permutations with Classes of Indistinguishable Objects

Applications 1, 2, and 5–8 suggest the following general notion: Suppose that there are n objects, n_1 of type 1, n_2 of type 2, ..., n_k of type k, with $n_1 + n_2 + \cdots + n_k = n$. Suppose that objects of the same type are indistinguishable. The number of distinguishable permutations of these objects is denoted $P(n;n_1,n_2,\ldots,n_k)$. We use the word "distinguishable" here because we assume that objects of the same type are indistinguishable. For instance, suppose that $n = 3$ and there are two type 1 objects, a and a, and one type 2 object, b. Then there are $3! = 6$ permutations of the three objects, but several of these are indistinguishable. For example, baa in which the first of the two a's comes second is indistinguishable from baa in which the second of the two a's comes second. There are only three distinguishable permutations, baa, aba, and aab.

Theorem 2.6

$$P(n; n_1, n_2, \ldots, n_k) = C(n; n_1, n_2, \ldots, n_k).$$

Proof. We have n positions or places to fill in the permutation, and we assign n_1 of these to type 1 objects, n_2 to type 2 objects, and so on. Q.E.D.

We return to permutations with classes of indistinguishable objects in Section 2.13.

EXERCISES FOR SECTION 2.11

1. Compute:

   (a) $C(7; 2, 2, 2, 1)$ (b) $C(9; 3, 3, 3)$ (c) $C(8; 1, 2, 2, 2, 1)$
   (d) $\Pr(6; 2, 2, 2)$ (e) $\Pr(10; 2, 1, 1, 2, 4)$ (f) $\Pr(8; 4, 2, 2, 0, 0)$
   (g) $P(9; 6, 1, 2)$ (h) $P(7; 3, 1, 3)$ (i) $P(3; 1, 1, 1)$

2. Find $C(n; 1, 1, 1, \ldots, 1)$.
3. Find:

   (a) $P(n; 1, n-1)$ (b) $\Pr(n; 1, n-1)$ (c) $\Pr(n; 1, n-1, 0)$

4. A code is being written using the five symbols $+$, $\natural$, $\bowtie$, ∇, and $\otimes$.

   (a) How many 10-digit codewords are there that use exactly 2 of each symbol?
   (b) If an 10-digit codeword is chosen at random, what is the probability that it will use exactly 2 of each symbol?

5. In a kennel that is short of space, 12 dogs must be put into 3 cages, 4 in cage 1, 5 in cage 2, and 3 in cage 3. In how many ways can this be done?
6. A code is being written using three symbols, a, b, and c.

   (a) How many 7-digit codewords can be written using exactly 4 a's, 1 b, and 2 c's?
   (b) If a 7-digit codeword is chosen at random, what is the probability that it will use exactly 4 a's, 1 b, and 2 c's?

7. A code is being written using the five digits 1, 2, 3, 4, and 5.

   (a) How many 15-digit codewords are there that use exactly 3 of each digit?
   (b) If a 15-digit codeword is chosen at random, what is the probability that it will use exactly 3 of each digit?

8. How many RNA chains have the same makeup of bases as the chain

   UGCCAUCCGAC?

9. (a) How many different "words" can be formed using all the letters of the word *excellent*?
   (b) If a 9-letter "word" is chosen at random, what is the probability that it will use all the letters of the word *excellent*?

10. How many ways are there to form a sequence of 10 letters from 4 a's, 4 b's, 4 c's, and 4 d's if each letter must appear at least twice?
11. How many distinguishable permutations are there of the symbols $a, a, a, a, a, b, c, d, e$ if no two a's are adjacent?
12. Repeat Exercise 20 of Section 2.10 under the assumption that the particles are distinguishable (the Maxwell-Boltzmann case).
13. (a) Suppose that we distinguish 5 different light particles hitting the retina. In how many ways could these be distributed among three cells, with three hitting the first cell and one hitting each of the other cells?
    (b) If we know there are 5 different light particles distributed among the three cells, what is the probability that they will be distributed as in part (a)?
14. Suppose that 35 radioactive particles hit a Geiger counter with 50 counters. In how many different ways can this happen with all but the 35th particle hitting the first counter?
15. Suppose that 6 people are invited for job interviews.
    (a) How many different ways are there for 2 of them to be interviewed on Monday, 2 on Wednesday, and 2 on Saturday?
    (b) Given the 6 interviews, what is the probability that the interviews will be distributed as in part (a) if the 6 people are assigned to days at random?
    (c) How many ways are there for the interviews to be distributed into 3 days, 2 per day?
16. Suppose that we have 4 elements. How many distinguishable ways are there to assign these to 4 distinguishable sets, 1 to each set, if the elements are:

    (a) a, b, c, d? (b) a, b, b, b? (c) a, a, b, b? (d) a, b, b, c?

2.12 COMPLETE DIGEST BY ENZYMES[20]

Let us consider the problem of discovering what a given RNA chain looks like without actually observing the chain itself (RNA chains were introduced in Section 2.11). Some enzymes break up an RNA chain into fragments after each G link. Others break up the chain after each C or U link. For example, suppose that we have the chain

CCGGUCCGAAAG.

Applying the G enzyme breaks the chain into the following fragments:

G fragments: CCG, G, UCCG, AAAG.

[20]This section may be omitted without loss of continuity. The material here is not needed again until Section 11.4.4. However, this section includes a detailed discussion of an applied topic, and we always include it in our courses.

We then know that these are the fragments, but we do not know in what order they appear. How many possible chains have these four fragments? The answer is that $4! = 24$ chains do: There is one chain corresponding to each of the different permutations of the fragments. One such chain (different from the original) is the chain

UCCGGCCGAAAG.

Suppose that we next apply the U, C enzyme, the enzyme that breaks up a chain after each C link or U link. We obtain the following fragments:

U, C fragments: C, C, GGU, C, C, GAAAG.

Again, we know that these are the fragments, but we do not know in what order they appear. How many chains are there with these fragments? One is tempted to say that there are 6! chains, but that is not right. For example, if the fragments were

C, C, C, C, C, C,

there would not be 6! chains with these fragments, but only one, the chain

CCCCCC.

The point is that some of the fragments are indistinguishable. To count the number of distinguishable chains with the given fragments, we note that there are six fragments. Four of these are C fragments, one is GGU, and one is GAAAG. Thus, by Theorem 2.6, the number of possible chains with these as fragments is

$$P(6;4,1,1) = C(6;4,1,1) = \frac{6!}{4!1!1!} = 30.$$

Actually, this computation is still a little off. Notice that the fragment GAAAG among the U, C fragments could not have appeared except as the terminal fragment because it does not end in U or C. Hence, we know that the chain ends

GAAAG.

There are five remaining U, C fragments: C, C, C, C, and GGU. The number of chains (beginning segments of chains) with these as fragments is

$$C(5;4,1) = 5.$$

The possible chains are obtained by adding GAAAG to one of these 5 beginning chains. The possibilities are

CCCCGGUGAAAG
CCCGGUCGAAAG
CCGGUCCGAAAG
CGGUCCCGAAAG
GGUCCCCGAAAG.

We have not yet combined our knowledge of both G and U, C fragments. Can we learn anything about the original chain by using our knowledge of both? Which of the 5 chains that we have listed has the proper G fragments? The first does not, for it would have a G fragment CCCCG, which does not appear among the fragments when the G enzyme is applied. A similar analysis shows that only the third chain,

CCGGUCCGAAAG,

has the proper set of G fragments. Hence, we have recovered the initial chain from among those that have the given U, C fragments.

This is an example of recovery of an RNA chain given a *complete enzyme digest*, that is, a split up after every G link and another after every U or C link. It is remarkable that we have been able to limit the large number of possible chains for any one set of fragments to only one possible chain by considering both sets of fragments. This result is more remarkable still if we consider trying to guess the chain knowing just its bases but not their order. Then we have

$$C(12; 4, 4, 3, 1) = 138{,}600 \text{ possible chains!}$$

Let us give another example. Suppose we are told that an RNA chain gives rise to the following fragments after complete digest by the G enzyme and the U, C enzyme:

G fragments: UG, ACG, AC
U, C fragments: U, GAC, GAC.

Can we discover the original chain? To begin with, we ask again whether or not the U, C fragments tell us which part of the chain must come last. The answer is that, in this case, they do not. However, the G fragments do: AC could only have arisen as a G fragment if it came last. Hence, the two remaining G fragments can be arranged in any order, and the possible chains with the given G fragments are

UGACGAC and ACGUGAC.

Now the latter chain would give rise to AC as one of the U, C fragments. Hence, the former must be the correct chain.

It is not always possible to recover the original RNA chain completely knowing the G fragments and U, C fragments. Sometimes the complete digest by these two enzymes is ambiguous in the sense that there are two RNA chains with the same set of G fragments and the same set of U, C fragments. We ask the reader to show this as an exercise (Exercise 8).

The "fragmentation stratagem" described in this section was used by R. W. Holley and his co-workers at Cornell University (Holley, *et al.* [1965]) to determine the first nucleic acid sequence. The method is not used anymore and indeed was used only for a short time before other, more efficient, methods were adopted. However, it has great historical significance and illustrates an important role for mathematical methods in biology. Nowadays, by the use of radioactive marking and high-speed computer analysis, it is possible to sequence long RNA chains rather quickly.

The reader who is interested in more details about complete digests by enzymes should read Hutchinson [1969], Mosimann [1968], or Mosimann, *et al.* [1966]. We return to this problem in Section 11.4.4.

EXERCISES FOR SECTION 2.12

1. An RNA chain has the following fragments after being subjected to complete digest by G and U, C enzymes:

   G fragments: CUG, CAAG, G, UC
   U, C fragments: C, C, U, AAGC, GGU.

   (a) How many RNA chains are there with these G fragments?

   (b) How many RNA chains are there with these U, C fragments?

   (c) Find *all* RNA chains that have these G and U, C fragments.

2. In Exercise 1, find the number of RNA chains with the same bases as those of the chains with the given G fragments.
3. Repeat Exercise 1 for the following G and U, C fragments:

   G fragments: G, UCG, G, G, UU
   U, C fragments: GGGU, U, GU, C.

4. In Exercise 3, find the number of RNA chains with the same bases as those of the chains with the given G fragments.
5. Repeat Exercise 1 for the following G and U, C fragments:

   G fragments: G, G, CC, CUG, G
   U, C fragments: GGGC, U, C, GC.

6. In Exercise 5, find the number of RNA chains with the same bases as those of the chains with the given G fragments.
7. A bit string is broken up after every 1 and after every 0. The resulting pieces (not necessarily in proper order) are as follows:

   break up after 1: 0, 001, 01, 01
   break up after 0: 0, 10, 0, 10, 10.

   (a) How many bit strings are there which have these pieces after breakup following each 1?

   (b) After each 0?

   (c) Find all bit strings having both of these sets of pieces.

8. Find an RNA chain which is ambiguous in the sense that there is another chain with the same G fragments and the same U, C fragments. (Can you find one with six or fewer links?)
9. What is the shortest possible RNA chain that is ambiguous in the sense of Exercise 8?
10. Can a bit string be ambiguous if it is broken up as in Exercise 7? Why?

2.13 PERMUTATIONS WITH CLASSES OF INDISTINGUISHABLE OBJECTS REVISITED

In Sections 2.11 and 2.12 we encountered the problem of counting the number of permutations of a set of objects in which some of the objects were indistinguishable. In this section we develop an alternative procedure for counting in this situation.

Example 2.32 "Hot Hand" A basketball player has observed that of his 10 shots attempted in an earlier game, 4 were made and 6 were missed. However, all 4 made shots came first. The basketball player's observation can be abbreviated as

$$\text{XXXXOOOOOO},$$

where X stands for a made shot and O for a missed one. Is this observation a coincidence, or does it suggest that the player had a "hot hand"? A hot hand assumes that once a player makes a shot, he or she has a higher-than-average chance of making the next shot. (For a detailed analysis of the hot hand phenomenon, see Tversky and Gilovich [1989].) To answer this question, let us assume that there is no such thing as having a hot hand, that is, that a shot is no more (or less) likely to go in when it follows a made (or missed) shot. Thus, let us assume that the made shots occur at random, and each shot has the same probability of being made, independent of what happens to the other shots. It follows that all possible orderings of 4 made shots and 6 missed shots are equally likely.[21] How many such orderings are there? The answer, to use the notation of Section 2.11, is

$$P(10;6,4) = C(10;6,4) = \frac{10!}{4!6!} = 210.$$

To derive this directly, note that there are 10 positions and we wish to assign 4 of these to X and 6 to O. Thus, the number of such orderings is $C(10;6,4) = 210$. If all such orderings are equally likely, the probability of seeing the specific arrangement

$$\text{XXXXOOOOOO}$$

is 1 out of 210. Of course, this is the probability of seeing any one given arrangement. What is more interesting to calculate is the probability of seeing 4 made shots together out of 10 shots in which exactly 4 are made. In how many arrangements of 10 shots, 4 made and 6 missed, do the 4 made ones occur together? To answer this, let us consider the 4 made shots as one unit X*. Then we wish to consider the number of orders of 1 X* and 6 O's. There are

$$C(7;1,6) = \frac{7!}{1!6!} = 7$$

[21] It does not follow that all orderings of 10 made and missed shots are equally likely. For instance, even if making a shot occurs at random, if making a shot is very unlikely, then the sequence OOOOOOOOOO is much more likely than the sequence XXXXXXXXXX.

such orders. These correspond to the orders

XXXXOOOOOO	(which is X*OOOOOO)
OXXXXOOOOO	(which is OX*OOOOO)
OOXXXXOOOO	(which is OOX*OOOO)
OOOXXXXOOO	(which is OOOX*OOO)
OOOOXXXXOO	(which is OOOOX*OO)
OOOOOXXXXO	(which is OOOOOX*O)
OOOOOOXXXX	(which is OOOOOOX*).

The probability of seeing 4 made shots and 6 missed shots with all the made shots together, given that there are 4 made shots and 6 missed ones, is therefore 7 out of 210, or $\frac{1}{30}$. This is quite small. Hence, since seeing all the made shots together is unlikely, we expect that perhaps this is not a random occurrence, and there is such a thing as a hot hand. (We return to the question of hot hand in Example 6.8.)

Before leaving this example, it is convenient to repeat the calculation of the number of ways of ordering 4 made shots and 6 missed ones. Suppose we label the shots so that they are distinguishable:

$$\mathrm{X}_a, \mathrm{X}_b, \mathrm{X}_c, \mathrm{X}_d, \mathrm{O}_a, \mathrm{O}_b, \mathrm{O}_c, \mathrm{O}_d, \mathrm{O}_e, \mathrm{O}_f.$$

There are 10! permutations of these 10 labels. For each such permutation, we can reorder the 4 X's arbitrarily; there are 4! such reorderings. Each reordering gives rise to an ordering which is considered the same as far as we are concerned. Similarly, we can reorder the 6 O's in 6! ways. Thus, groups of $4! \times 6!$ orderings are the same, and each permutation corresponds to $4! \times 6!$ similar ones. The number of indistinguishable permutations is

$$\frac{10!}{4!6!}. \qquad \blacksquare$$

The reasoning in Example 2.32 generalizes to give us another proof of the result (Theorem 2.6) that $P(n; n_1, n_2, \ldots, n_k) = C(n; n_1, n_2, \ldots, n_k)$.

EXERCISES FOR SECTION 2.13

1. Suppose a researcher observes that of 12 petri dishes, 4 have growths and 8 do not. The 4 dishes with growths are next to each other. Assuming that 4 of the 12 petri dishes have growths and that all orderings of these dishes are equally likely, what is the probability that the 4 dishes with growths will all be next to each other?
2. A market researcher observes that of 11 cars on a block, 4 are foreign and 7 are domestic. The 4 foreign cars are next to each other. Assuming that 4 of the 11 cars are foreign and that all orderings of these 11 cars are equally likely, what is the probability that the 4 foreign cars are next to each other?
3. Suppose a forester observes that some trees are sick (S) and some well (W), and that of 13 trees in a row, the first 5 are S and the last 8 are W.

(a) Assuming that sickness occurs at random and all trees have the same probability of being sick, independent of what happens to the other trees, what is the probability of observing the given order?

(b) What is the probability that out of 13 trees, 5 sick and 8 well, the 5 sick ones occur together?

4. Suppose a forester observes that some trees are sick (S), some well (W), and some questionable (Q). Assuming that of 30 trees, 10 are sick or questionable, what is the probability that these 10 appear consecutively? Assume that all sequences of sick, well, and questionable with 10 sick or questionable are equally likely.

5. Suppose that of 11 houses lined up in one block, 6 are infested with termites.

   (a) In how many ways can the presence or absence of termites occur so that these 6 houses are next to each other?

   (b) In how many ways can this occur so that none of these 6 houses are next to each other?

   (c) In how many ways can we schedule an order of visits that go to two of these houses in which there are no termites?

   (d) In how many ways can we schedule an order of visits that go to two of these houses if at most one house that we visit can have termites?

6. In how many different orders can a couple produce 9 children with 5 boys so that a boy comes first and all 4 girls are born consecutively?

7. How many distinct ways are there to seat 8 people around a circular table? (Clarify what "distinct" means here.)

8. If an RNA chain of length 4 is chosen at random, what is the probability that it has:

   (a) At least three consecutive C's? (b) At least two consecutive C's?

   (c) A consecutive AG? (d) A consecutive AUC?

9. How many bit strings of length 21 have every 1 followed by 0 and have seventeen 0's and four 1's?

10. How many RNA chains of length 20 have five A's, four U's, five C's, and six G's and have every C followed by G?

11. There are 20 people whose records are stored in order in a file. We want to choose 4 of these at random in performing a survey, making sure not to choose two consecutive people. In how many ways can this be done? (*Hint:* Either we choose the last person or we do not.)

2.14 THE BINOMIAL EXPANSION

Suppose that 6-position license plates are being made with each of the six positions being either a number or a letter. Referring back to the sum and product rules, there are

$$(10 + 26)^6 = 2{,}176{,}782{,}336$$

possible license plates. Generalizing, suppose that the license plates need to be n digits with each position being either one of a things or one of b things. Then there are

$$(a+b)^n$$

license plates. As an application of the ideas considered in this chapter, let us develop a useful formula for $(a+b)^n$.

Theorem 2.7 (Binomial Expansion) For $n \geq 0$,

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

Proof.[22] Note that

$$(a+b)^n = \underbrace{(a+b)(a+b)\cdots(a+b)}_{n \text{ times}}.$$

In multiplying out, we pick one term from each factor $(a+b)$. Hence, we only obtain terms of the form $a^k b^{n-k}$. To find the coefficient of $a^k b^{n-k}$, note that to obtain $a^k b^{n-k}$, we need to choose k of the terms from which to choose a. This can be done in $\binom{n}{k}$ ways. Q.E.D.

In particular, we have

$$\begin{aligned}
(a+b)^2 &= \binom{2}{0}a^2 + \binom{2}{1}ab + \binom{2}{2}b^2 = a^2 + 2ab + b^2 \\
(a+b)^3 &= \binom{3}{0}a^3 + \binom{3}{1}a^2b + \binom{3}{2}ab^2 + \binom{3}{3}b^3 = a^3 + 3a^2b + 3ab^2 + b^3 \\
(a+b)^4 &= \binom{4}{0}a^4 + \binom{4}{1}a^3b + \binom{4}{2}a^2b^2 + \binom{4}{3}ab^3 + \binom{4}{4}b^4 \\
&= a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4.
\end{aligned}$$

The reader might wish to compare the coefficients here with those numbers appearing in Pascal's triangle (Figure 2.4). Do you notice any similarities?

It is not hard to generalize the binomial expansion of Theorem 2.7 to an expansion of

$$(a+b+c)^n$$

and more generally of

$$(a_1 + a_2 + \cdots + a_k)^n.$$

We leave the generalizations to the reader (Exercises 5 and 8).

Let us give a few applications of the binomial expansion here. The coefficient of x^{20} in the expansion of $(1+x)^{30}$ is obtained by taking $a=1$ and $b=x$ in Theorem 2.7. We are seeking the coefficient of $1^{10}x^{20}$, that is, $C(30,10)$.

[22] For an alternative proof, see Exercise 12.

Theorem 2.8 For $n \geq 0$,

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

Proof. Note that

$$2^n = (1+1)^n.$$

Hence, by the binomial expansion with $a = b = 1$,

$$2^n = \sum_{k=0}^{n} \binom{n}{k} 1^k 1^{n-k}.$$

Q.E.D.

Another way of looking at Theorem 2.8 is the following. The number 2^n counts the number of subsets of an n-set. Also, the left-hand side of Theorem 2.8 counts the number of 0-element subsets of an n-set plus the number of 1-element subsets plus ... plus the number of n-element subsets. Each subset is counted once and only once in this way.

Theorem 2.9 For $n > 0$,

$$\binom{n}{0} - \binom{n}{1} + \binom{n}{2} - \cdots + (-1)^k \binom{n}{k} + \cdots + (-1)^n \binom{n}{n} = 0.$$

Proof.

$$0 = (1-1)^n = (-1+1)^n = \sum_{k=0}^{n} \binom{n}{k} (-1)^k (1)^{n-k}.$$

Q.E.D.

Corollary 2.9.1 For $n > 0$,

$$\binom{n}{0} + \binom{n}{2} + \cdots = \binom{n}{1} + \binom{n}{3} + \cdots.$$

The interpretation of this corollary is that the number of ways to select an even number of objects from n equals the number of ways to select an odd number.

EXERCISES FOR SECTION 2.14

1. Write out:

   (a) $(x+y)^5$ (b) $(a+2b)^3$ (c) $(2u+3v)^4$

2. Find the coefficient of x^{11} in the expansion of:

   (a) $(1+x)^{16}$ (b) $(2+x)^{14}$ (c) $(2x+4y)^{11}$

3. What is the coefficient of x^9 in the expansion of $(1+x)^{12}(1+x)^4$?
4. What is the coefficient of x^8 in the expansion of $(1+x)^{10}(1+x)^6$?

5. Find a formula for $(a+b+c)^n$.
6. Find the coefficient of $a^2b^2c^2$ in the expansion of $(a+b+c)^6$.
7. Find the coefficient of xyz^3 in the expansion of $(x+y+z)^5$.
8. Find a formula for $(a_1+a_2+\cdots+a_k)^n$.
9. What is the coefficient of a^3bc^2 in the expansion of $(a+b+c)^6$?
10. What is the coefficient of xy^2z^2w in the expansion of $(x+y+z+2w)^6$?
11. What is the coefficient of $a^3b^2cd^6$ in the expansion of $(a+5b+2c+2d)^{12}$?
12. Prove Theorem 2.7 by induction on n.
13. Find $\binom{12}{0}+\binom{12}{2}+\binom{12}{4}+\binom{12}{6}+\binom{12}{8}+\binom{12}{10}+\binom{12}{12}$.
14. Prove that the number of even-sized subsets of an n-set equals 2^{n-1}.
15. A bit string has *even parity* if it has an even number of 1's. How many bit strings of length n have even parity?
16. Find:

    (a) $\sum_{k=0}^{n} 2^k\binom{n}{k}$ (b) $\sum_{k=0}^{n} 4^k\binom{n}{k}$

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

    (e) $\sum_{k=1}^{n} k\binom{n}{k}$ [*Hint:* Differentiate the expansion of $(x+1)^n$ and set $x=1$.]
17. Show that:

    (a) $\sum_{k=1}^{n} k\binom{n}{k}2^{k-1}2^{n-k}=n(4^{n-1})$ (b) $\sum_{k=1}^{n} k\binom{n}{k}2^{n-k}=n(3^{n-1})$

2.15 POWER IN SIMPLE GAMES[23]

2.15.1 Examples of Simple Games

In this section we apply to the analysis of multiperson games some of the counting rules described previously. Now in modern applied mathematics, a game has come to mean more than just Monopoly, chess, or poker. It is any situation where a group of players is competing for different rewards or payoffs. In this sense, politics is a game, the economic marketplace is a game, the international bargaining arena is a game, and so on. We shall take this broad view of games here.

Let us think of a game as having a set I of n players. We are interested in possible cooperation among the players and, accordingly, we study *coalitions* of players, which correspond to subsets of the set I. We concentrate on *simple games*,

[23]This section may be omitted without loss of continuity. The formal prerequisites for this section are Sections 2.1–2.8.

games in which each coalition is either winning or losing. We can define a simple game by giving a value function v which assigns the number 0 or 1 to each coalition $S \subseteq I$, with $v(S)$ equal to 0 if S is a losing coalition and 1 if S is a winning coalition. It is usually assumed in game theory that a subset of a losing coalition cannot be winning, and we shall make that assumption. It is also usually assumed that for all S, either S or $I - S$ is losing. We shall assume that.

Very important examples of simple games are the weighted majority games. In a *weighted majority game*, there are n players, player i has v_i votes, and a coalition is winning if and only if it has at least q votes. We denote this game by

$$[q; v_1, v_2, \ldots, v_n].$$

The assumption that either S or $I - S$ loses places restrictions on the allowable $q, v_1, v_2, \ldots, v_n$. For instance, $[3; 4, 4]$ does not satisfy this requirement. Weighted majority games arise in corporations, where the players are the stockholders and a stockholder has one vote for each share owned. Most legislatures are weighted majority games of the form $[q; 1, 1, \ldots, 1]$, where each player has one vote. However, some legislatures give a legislator a number of votes corresponding to the population of his or her district. For example, in 1964 the Nassau County, New York, Board of Supervisors was the weighted majority game $[59; 31, 31, 21, 28, 2, 2]$ (Banzhaf [1965]).

Another example is the Council of the European Union. This body, made up of 27 member states, legislates for the Union. In most of its cases, the Council decides by a "qualified majority vote" from its member states carrying the following weights:

Member Countries	Votes
Germany, France, Italy, and the United Kingdom	29
Spain and Poland	27
Romania	14
The Netherlands	13
Belgium, Czech Republic, Greece, Hungary, and Portugal	12
Austria, Bulgaria, and Sweden	10
Denmark, Ireland, Lithuania, Slovakia, and Finland	7
Cyprus, Estonia, Latvia, Luxembourg, and Slovenia	4
Malta	3
Total	345

At least 255 votes are required for a qualified majority. [In addition, a majority of member states (and in some cases a two-thirds majority) must approve for a qualified majority to be reached. We will not consider this criterion.] Therefore, this weighted majority game is

$$[255; 29, 29, 29, 29, 27, 27, 14, 13, 12, 12, 12, 12, 12, 10, 10, 10, 7, 7, 7, 7, 7, 4, 4, 4, 4, 4, 3].$$

Perhaps the most elementary weighted majority game is the game $[2; 1, 1, 1]$. In this game there are three players, each having one vote, and a simple majority of the players forms a winning coalition. Thus, the winning coalitions are the sets

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

Another example of a simple game is the U.N. Security Council. Here there are 15 players: 5 permanent members (China, France, Russia, the United Kingdom, the United States) and 10 nonpermanent members. For decisions on substantive matters, a coalition is winning if and only if it has all 5 permanent members, since they have veto power,[24] and at least 4 of the 10 nonpermanent members. (Decisions on procedural matters are made by an affirmative vote of at least nine of the 15 members.) It is interesting to note that for substantive decisions the Security Council can be looked at as a weighted majority game. Consider the game

$$[39; 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],$$

where the first five players correspond to the permanent members. The winning coalitions in this game are exactly the same as those in the Security Council, as is easy to check. Hence, even though weighted votes are not explicitly assigned in the Security Council, it can be considered a weighted majority game. (The reader might wish to think about how to obtain numbers, such as 39, 7, and 1, which translate a simple game into a weighted majority game.)

A similar situation arises for the Australian government. In making national decisions, 6 states and the federal government play a role. In effect, a measure passes if and only if it has the support of at least 5 states or at least 2 states and the federal government. As is easy to see, this simple game corresponds (in the sense of having the same winning coalitions) to the game $[5; 1, 1, 1, 1, 1, 1, 3]$, where the seventh player is the federal government.

Not every simple game is a weighted majority game. A bicameral legislature is an example (see Exercise 5).

2.15.2 The Shapley-Shubik Power Index

We shall be concerned with measuring the *power* of a player in a simple game: his or her ability to maneuver into a winning coalition. Note first that power is not necessarily proportional to the number of votes a player has. For example, compare the two games $[2; 1, 1, 1]$ and $[51; 49, 48, 3]$. In each game, there are 3 players, and any coalition of 2 or more players wins. Thus, in both games, player 3 is in the same winning coalitions, and hence has essentially the same power. These two games might be interpreted as a legislature with 3 parties. In the first legislature, there are 3 equal parties and 2 out of 3 must go along for a measure to pass. In the second legislature, there are 2 large parties and a small party. However, assuming that party members vote as a bloc, it is still necessary to get 2 out of 3 parties to go along to pass a measure, so in effect the third small party has as much power as it does in the first legislature.

There have been a number of alternative approaches to the measurement of power in simple games. These include power indices proposed by Banzhaf [1965], Coleman [1971], Deegan and Packel [1978], Johnston [1978], and Shapley and Shubik [1954]. We shall refer to the Banzhaf and Coleman power indices in Section 5.7.

[24] This is the rule of "great power unanimity."

The former has come to be used in the courts in "one-person, one-vote" cases. Here we concentrate on the Shapley-Shubik power index, introduced in its original form by Shapley [1953] and in the form we present by Shapley and Shubik [1954]. (In its more general original form, it is called the *Shapley value*.) For a survey of the literature on the Shapley-Shubik index, see Shapley [1981]. For a survey of the literature on power measures, see Lucas [1983].

Let us think of building up a coalition by adding one player at a time until we reach a winning coalition. The player whose addition throws the coalition over from losing to winning is called *pivotal*. More formally, let us consider any permutation of the players and call a player i *pivotal* for that permutation if the set of players preceding i is losing, but the set of players up to and including player i is winning. For example, in the game $[2;1,1,1]$, player 2 is pivotal in the permutation 1, 2, 3 and in the permutation 3, 2, 1. The *Shapley-Shubik power index* p_i for player i in a simple game is defined as follows:

$$p_i = \frac{\text{number of permutations of the players in which } i \text{ is pivotal}}{\text{number of permutations of the players}}.$$

If we think of one permutation of the players being chosen at random, the Shapley-Shubik power index for player i is the probability that player i is pivotal. In the game $[2;1,1,1]$, for example, there are three players and hence 3! permutations. Each player is pivotal in two of these. For example, as we have noted, player 2 is pivotal in 1, 2, 3 and 3, 2, 1. Hence, each player has power $2/3! = 1/3$. In the game $[51;49,48,3]$, player 1 is pivotal in the permutations 2, 1, 3 and 3, 1, 2. For in the first he or she brings in enough votes to change player 2's 48 votes into 97 and in the second he or she brings in enough votes to change player 3's 3 votes to 52. Thus,

$$p_1 = \frac{2}{3!} = \frac{1}{3}.$$

Similarly, player 2 is pivotal in permutations 1, 2, 3 and 3, 2, 1, and player 3 in permutations 1, 3, 2 and 2, 3, 1. Hence,

$$p_2 = p_3 = \frac{1}{3}.$$

Thus, as anticipated, the small third party has the same power as the two larger parties.

In the game $[51;40,30,15,15]$, the possible permutations are shown in Table 2.10, and the pivotal player in each is circled. Player 1 is pivotal 12 times, so his or her power is $12/4! = 1/2$. Players 2, 3, and 4 are each pivotal 4 times, so they each have power $4/4! = 1/6$.

Let us compute the Shapley-Shubik power index for the Australian government, the game $[5;1,1,1,1,1,1,3]$. In this and the following examples, the enumeration of all permutations is not the most practical method for computing the Shapley-Shubik power index. We proceed by another method. The federal government (player 7) is pivotal in a given permutation if and only if it is third, fourth, or fifth. By symmetry, we observe that the federal government is picked in the ith position

Table 2.10: All Permutations of the Players in the Game $[51; 40, 30, 15, 15]$, with Pivotal Player Circled

1 2 3 4	2 1 3 4	3 1 2 4	4 1 2 3
1 2 4 3	2 1 4 3	3 1 4 2	4 1 3 2
1 3 2 4	2 3 1 4	3 2 1 4	4 2 1 3
1 3 4 2	2 3 4 1	3 2 4 1	4 2 3 1
1 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
1 4 3 2	2 4 3 1	3 4 2 1	4 3 2 1

in a permutation of the 7 players in exactly 1 out of every 7 permutations. Hence, it is picked third, fourth, or fifth in exactly 3 out of every 7 permutations. Thus, the probability that the federal government is pivotal is 3/7; that is,

$$p_7 = \frac{3}{7}.$$

We can also see this by observing that the number of permutations of the seven players in which the federal government is third is 6!, for we have to order the remaining players. Similarly, the number of permutations in which the federal government is fourth (or fifth) is also 6!. Thus,

$$p_7 = \frac{3 \cdot 6!}{7!} = \frac{3}{7}.$$

Now it is easy to see that

$$p_1 + p_2 + \cdots + p_7 = 1.$$

It is always the case that

$$\sum_i p_i = 1.$$

(Why?) Hence,

$$p_1 + p_2 + p_3 + p_4 + p_5 + p_6 = 1 - \frac{3}{7} = \frac{4}{7}.$$

Since by symmetry

$$p_1 = p_2 = \cdots = p_6,$$

each of these numbers is equal to

$$\frac{4}{7} \div 6 = \frac{2}{21}.$$

Thus, although the federal government has only 3 times the number of votes of a state, it has $4\frac{1}{2}$ times the power (9/21 vs. 2/21).

2.15.3 The U.N. Security Council

Let us turn next to the Shapley-Shubik power index for the U.N. Security Council. Let us fix a nonpermanent player i. Player i is pivotal in exactly those permutations in which all permanent players precede i and exactly three nonpermanent players precede i. How many such permutations are there? To find such a permutation, we first choose the three nonpermanent players who precede i; for each such choice, we order all eight players who precede i (the five permanent and three nonpermanent ones); for each choice and ordering, we order the remaining six nonpermanent players who follow i. The number of ways to make the first choice is $C(9,3)$, the number of ways to order the preceding players is $8!$, and the number of ways to order the following players is $6!$. Thus, by the product rule, the number of permutations in which i is pivotal is given by

$$C(9,3) \times 8! \times 6! = \frac{9!}{3!6!} \times 8! \times 6! = \frac{9!8!}{3!}.$$

Thus, since the total number of permutations of the 15 players is $15!$,

$$p_i = \frac{9!8!}{3!} \div 15! \approx .001865.$$

It follows that the sum of the powers of the nonpermanent players is 10 times this number; that is, it is .01865. Thus, since all the powers add to 1, the sum of the powers of the permanent players is .98135. There are five permanent players, each of whom, by symmetry, has equal power. It follows that each has power

$$p_j = \frac{.98135}{5} = .1963.$$

Hence, permanent members have more than 100 times the power of nonpermanent members. (The idea of calculating power in the U.N. Security Council and other legislative bodies in this manner was introduced in Shapley and Shubik [1954].)

2.15.4 Bicameral Legislatures

To give a more complicated example, suppose that we have a bicameral legislature with n_1 members in the first house and n_2 members in the second house.[25] Suppose that a measure can pass only if it has a majority in each house of the legislature, and suppose for simplicity that n_1 and n_2 are both odd. Let I be the union of the sets of members of both houses, and let π be any permutation of I. A player i in the jth house is pivotal in π if he is the $[(n_j+1)/2]$th player of his house in π and a majority of players in the other house precede him. However, for every permutation

[25] This example is also due to Shapley and Shubik [1954].

π in which a player in the first house is pivotal, the reverse permutation makes a player in the second house pivotal. (Why?) Moreover, every permutation has some player as pivotal. Thus, some player in house number 1 is pivotal in exactly $\frac{1}{2}$ of all permutations. Since all players in house number 1 are treated equally, any one of these players is pivotal in exactly $1/(2n_1)$ of the permutations. Similarly, any player of house number 2 will be pivotal in exactly $1/(2n_2)$ of the permutations. Thus, each player of house number 1 has power $1/(2n_1)$ and each player of house number 2 has power $1/(2n_2)$. In the U.S. House and Senate, $n_1 = 435$ and $n_2 = 101$, including the Vice-President who votes in case of a tie. According to our calculation, each representative has power $1/870 \approx .0011$ and each senator (including the Vice-President) has power $1/202 \approx .005$. Thus, a senator has about five times as much power as a representative.

Next, let us add an executive (a governor, the President) who can veto the vote in the two houses, but let us assume that there is no possibility of overriding the veto. Now there are $n_1 + n_2 + 1$ players and a coalition is winning if and only if it contains the executive and a majority from each house. Assuming that n_1 and n_2 are large, Shapley and Shubik [1954] argue that the executive will be pivotal in approximately one-half of the permutations. (This argument is a bit complicated and we omit it.) The two houses divide the remaining power almost equally. Finally, if the possibility of overriding the veto with a two-thirds majority of both houses is added, a similar discussion implies that the executive has power approximately one-sixth, and the two houses divide the remaining power almost equally. The reader is referred to Shapley and Shubik's paper for details.

Similar calculations can be made for the relative power that various states wield in the electoral college. Mann and Shapley [1964a,b] calculated this using the distribution of electoral votes as of 1961. New York had 43 out of the total of 538 electoral votes, and had a power of .0841. This compared to a power of .0054 for states like Alaska, which had three electoral votes. According to the Shapley-Shubik power index, the power of New York exceeded its percentage of the vote, whereas that of Alaska lagged behind its percentage.

Similar results for the distribution of electoral votes as of 1972 were obtained by Boyce and Cross [unpublished observations, 1973]. In the 1972 situation, New York had a total of 41 electoral votes (the total was still 538) and a power of .0797, whereas Alaska still had three electoral votes and a power of .0054. For a more comprehensive discussion of power in electoral games, see Brams, Lucas, and Straffin [1983], Lucas [1983], Shapley [1981], and Straffin [1980].

2.15.5 Cost Allocation

Game-theoretic solutions such as the Shapley-Shubik power index and the more general Shapley value have long been used to allocate costs to different users in shared projects. Examples of such applications include allocating runway fees to different users of an airport, highway fees to different-size trucks, costs to different colleges sharing library facilities, and telephone calling charges among users. See Lucas [1981a], Okada, Hashimoto, and Young [1982], Shubik [1962], Straffin and

Heaney [1981], and Young [1986].

These ideas have found fascinating recent application in multicast transmissions, for example, of movies over the Internet. In unicast routing, each packet sent from a source is delivered to a single receiver. To send the same packet to multiple sites requires the source to send multiple copies of the packet and results in a large waste of bandwidth. In multicast routing, we use a "directed tree" connecting the source to all receivers, and at branch points a packet is duplicated as necessary. The bandwidth used by a multicast transmission is not directly attributable to a single receiver and so one has to find a way to distribute the cost among various receivers. Feigenbaum, Papadimitriou, and Shenker [2000], Herzog, Shenker, and Estrin [1997], and Jain and Vazirani [2001] applied the Shapley value to determining cost distribution in the multicasting application and studied the computational difficulty of implementing their methods.

2.15.6 Characteristic Functions

We have concentrated in this section on simple games, games that can be defined by giving each coalition S a *value* $v(S)$ equal to 0 or 1. The value is often interpreted as the best outcome a coalition can guarantee itself through cooperation. If the value function or *characteristic function* $v(S)$ can take on arbitrary real numbers as values, the game is called a game in *characteristic function form.* Such games have in recent years found a wide variety of applications, such as in water and air pollution, disarmament, and bargaining situations. For a summary of applications, see Brams, Schotter, and Schwödianer [1979] or Lucas [1981a]. For more information about the theory of games in characteristic function form, see Fudenberg and Tirole [1991], Myerson [1997], Owen [1995], or Roberts [1976].

For more on game theory in general, see, for example, Aumann and Hart [1998], Fudenberg and Tirole [1991], Jones [1980], Lucas [1981b], Myerson [1997], Owen [1995], or Stahl [1998].

EXERCISES FOR SECTION 2.15

1. For each of the following weighted majority games, describe all winning coalitions.
   (a) $[65; 50, 30, 20]$ (b) $[125; 160, 110, 10]$
   (c) The Board of Supervisors, Nassau County, NY, 1964: $[59; 31, 31, 21, 28, 2, 2]$
   (d) $[80; 44, 43, 42, 41, 5]$ (e) $[50; 35, 35, 35, 1]$
2. For the following weighted majority games, identify all *minimal* winning coalitions, that is, winning coalitions with the property that removal of any player results in a losing coalition.
   (a) $[14; 6, 6, 8, 12, 2]$ (b) $[60; 58, 7, 1, 1, 1, 1]$
   (c) $[20; 6, 6, 6, 6]$ (d) All games of Exercise 1

3. Calculate the Shapley-Shubik power index for each player in the following weighted majority games.
   (a) $[51; 49, 47, 4]$
   (b) $[201; 100, 100, 100, 100, 1]$
   (c) $[151; 100, 100, 100, 1]$
   (d) $[51; 26, 26, 26, 22]$
   (e) $[20; 8, 8, 4, 2]$ (*Hint:* Is player 4 ever pivotal?)
4. Calculate the Shapley-Shubik power index for the following games.
   (a) $[16; 9, 9, 7, 3, 1, 1]$. (This game arose in the Nassau County, New York, Board of Supervisors in 1958; see Banzhaf [1965].)
   (b) $[59; 31, 31, 21, 28, 2, 2]$. (This game arose in the Nassau County, New York, Board of Supervisors in 1964; again see Banzhaf [1965].)
5. Consider a conference committee consisting of three senators, x, y, and z, and three members of the House of Representatives, a, b, and c. A measure passes this committee if and only if it receives the support of at least two senators and at least two representatives.
   (a) Identify the winning coalitions of this simple game.
   (b) Show that this game is not a weighted majority game. That is, we cannot find votes $v(x), v(y), v(z), v(a), v(b)$, and $v(c)$ and a quota q such that a measure passes if and only if the sum of the votes in favor of it is at least q. (*Note:* A similar argument shows that, in general, a bicameral legislature cannot be thought of as a weighted majority game.)
6. Which of the following defines a weighted majority game in the sense that there is a weighted majority game with the same winning coalitions? Give a proof of your answer.
   (a) Three players, and a coalition wins if and only if player 1 is in it.
   (b) Four players, a, b, x, y; a coalition wins if and only if at least a or b and at least x or y is in it.
   (c) Four players and a coalition wins if and only if at least three players are in it.
7. Suppose that a country has 3 provinces. The number of representatives of each province in the state legislature is given as follows: Province A has 6, province B has 7, and province C has 2. If all representatives of a province vote alike, and a two-thirds majority of votes is needed to win, find the power of each province using the Shapley-Shubik power index.
8. Calculate the Shapley-Shubik power index for the conference committee (Exercise 5).
9. Prove that in a bicameral legislature, for every permutation in which a player in the first house is pivotal, the reverse permutation makes a player in the second house pivotal.
10. (Lucas [1983]) In the original Security Council, there were five permanent members and only six nonpermanent members. The winning coalitions consisted of all five permanent members plus at least two nonpermanent members.
    (a) Formulate this as a weighted majority game.
    (b) Calculate the Shapley-Shubik power index.

11. (Lucas [1983]) It has been suggested that Japan be added as a sixth permanent member of the Security Council. If this were the case, assume that there would still be 10 nonpermanent members and winning coalitions would consist of all six permanent members plus at least four nonpermanent members.

    (a) Formulate this as a weighted majority game.

    (b) Calculate the Shapley-Shubik power index.

12. Compute the Shapley-Shubik power index of a player with 1 vote in the game in which 6 players have 11 votes each, 12 players have 1 vote each, and 71 votes are needed to win.

13. If we do not require that every subset of a losing coalition is a losing coalition or that for all S, either S or $1-S$ is losing, then how many different simple games are there on a set of n players?

14. In a simple game, if p_i is the Shapley-Shubik power index for player i and $\sum_{i \in S} p_i$ is greater than $\frac{1}{2}$, is S necessarily a winning coalition? Why?

15. Suppose that $v(S)$ gives 1 if coalition S is winning and 0 if S is losing. If p_i is the Shapley-Shubik power index for player i, show that

$$p_i = \sum \{\gamma(s)[v(S) - v(S - \{i\})] : S \text{ such that } i \in S\}, \tag{2.9}$$

    where

$$s = |S| \text{ and } \gamma(s) = \frac{(s-1)!(n-s)!}{n!}.$$

16. Apply formula (2.9) in Exercise 15 to compute the Shapley-Shubik power index for each of the weighted majority games in Exercise 3.

17. It is usually assumed that if v is a characteristic function, then

$$v(\emptyset) = 0 \tag{2.10}$$

$$v(S \cup T) \geq v(S) + v(T) \text{ if } S \cap T = \emptyset. \tag{2.11}$$

    Which of the following characteristic functions on $I = \{1, 2, \ldots, n\}$ have these two properties?

    (a) $n = 3$, $v(\emptyset) = 0$, $v(\{1\}) = v(\{2\}) = v(\{3\}) = -1$, $v(\{1,2\}) = 3$, $v(\{1,3\}) = 3$, $v(\{2,3\}) = 4$, $v(\{1,2,3\}) = 2$.

    (b) $n = 3$, $v(\emptyset) = 0$, $v(\{1\}) = v(\{2\}) = 0$, $v(\{3\}) = -1$, $v(\{1,2\}) = 1$, $v(\{1,3\}) = 0$, $v(\{2,3\}) = -1$, $v(\{1,2,3\}) = 0$.

    (c) Arbitrary n, $v(S) = -|S|$, for all S.

18. Show that the following characteristic function on $I = \{1, 2, 3, 4\}$ satisfies conditions (2.10) and (2.11) of Exercise 17.[26]

[26] This and the next exercise are unpublished exercises due to A. W. Tucker.

$$
\begin{aligned}
v(\emptyset) &= 0 \\
v(\{i\}) &= 0 && \text{(all } i\text{)} \\
v(\{i,j\}) &= \frac{i+j}{10} && \text{(all } i \neq j\text{)} \\
v(\{i,j,k\}) &= \frac{i+j+k}{10} && \text{(all distinct } i,j,k\text{)} \\
v(\{1,2,3,4\}) &= \frac{1+2+3+4}{10}.
\end{aligned}
$$

19. Generalizing Exercise 18, let $I = \{1, 2, \ldots, 2n\}$, and let

$$
\begin{aligned}
v(\emptyset) &= 0 \\
v(\{i\}) &= 0 && \text{(all } i\text{)} \\
v(S) &= \sum_{i \in S} c_i && \text{(for each } S \text{ with } |S| > 1,
\end{aligned}
$$

where the c_i are $2n$ positive constants with sum equal to 1. Verify that v satisfies conditions (2.10) and (2.11) of Exercise 17.

20. In the game called *deterrence*, each of the n players has the means to destroy the wealth of any other player. If w_i is the wealth of player i, then $v(S)$ is given by

$$
v(S) = \begin{cases} -\sum_{i \in S} w_i & \text{(if } |S| < n\text{)} \\ 0 & \text{(if } |S| = n\text{).} \end{cases}
$$

Show that $v(S)$ satisfies conditions (2.10) and (2.11) of Exercise 17.

21. In the game called *pure bargaining*, a private foundation has offered n states a total of d dollars for development of water pollution abatement facilities provided that the states can agree on the distribution of the money. In this game $v(S)$ is given by

$$
v(S) = \begin{cases} 0 & \text{(if } |S| < n\text{)} \\ d & \text{(if } |S| = n\text{).} \end{cases}
$$

Show that $v(S)$ satisfies conditions (2.10) and (2.11) of Exercise 17.

22. In the game called *two-buyer market*, player 1 owns an object worth a units to him. Player 2 thinks the object is worth b units and player 3 thinks it is worth c units. Assuming that $a < b < c$, the characteristic function of this game is given by

$$
\begin{aligned}
v(\{1\}) &= a & v(\{2\}) &= 0 & v(\{3\}) &= 0 & v(\{1,2\}) &= b \\
v(\{1,3\}) &= c & v(\{2,3\}) &= 0 & v(\{1,2,3\}) &= c.
\end{aligned}
$$

Show that $v(S)$ satisfies conditions (2.10) and (2.11) of Exercise 17.

23. (Shapley and Shubik [1969]) In the *garbage game*, each of n players has a bag of garbage that he or she *must* drop in someone else's yard. The utility or worth of b bags of garbage is $-b$. Then

$$
v(S) = \begin{cases} 0 & \text{(if } s = 0\text{)} \\ -(n-s) & \text{(if } 0 < s < n\text{)} \\ -n & \text{(if } s = n\text{),} \end{cases}
$$

where $s = |S|$. Show that $v(S)$ satisfies conditions (2.10) and (2.11) of Exercise 17.

24. Formula (2.9) of Exercise 15 can be used as the definition of the Shapley-Shubik power index for a game in characteristic function form. Using this formula:
    (a) Calculate the Shapley-Shubik power index for the game of Exercise 20.
    (b) Calculate the Shapley-Shubik power index for the game of Exercise 21.
    (c) Calculate the Shapley-Shubik power index for the game of Exercise 22 if $a = 3$, $b = 5$, and $c = 10$.
    (d) Calculate the Shapley-Shubik power index for the game of Exercise 23.
    (e) Calculate the Shapley-Shubik power index for the game of Exercise 19.

2.16 GENERATING PERMUTATIONS AND COMBINATIONS[27]

In Examples 2.10 and 2.17 we discussed algorithms that would proceed by examining every possible permutation of a set. (Other times, we may be interested in every r-combination or every subset.) We did not comment there on the problem of determining in what order to examine the permutations, because we were making the point that such algorithms are not usually very efficient. However, there are occasions when such algorithms are useful. In connection with them, we need a procedure to generate all permutations of a set and in general, all members of a certain class of combinatorial objects. In this section we describe such procedures.

2.16.1 An Algorithm for Generating Permutations

A natural order in which to examine permutations is the *lexicographic order*. To describe this order, suppose that $\pi = \pi_1\pi_2\pi_3$ and $\sigma = \sigma_1\sigma_2\sigma_3$ are two permutations of the set $\{1, 2, 3\}$. We say that π *precedes* σ if $\pi_1 < \sigma_1$ or if $\pi_1 = \sigma_1$ and $\pi_2 < \sigma_2$. (Note that if $\pi_1 = \sigma_1$ and $\pi_2 = \sigma_2$, then $\pi_3 = \sigma_3$.) For instance, $\pi = 123$ precedes $\sigma = 231$ since $\pi_1 = 1 < 2 = \sigma_1$, and $\pi = 123$ precedes $\sigma = 132$ because $\pi_1 = 1 = \sigma_1$ and $\pi_2 = 2 < 3 = \sigma_2$. More generally, suppose that $\pi = \pi_1\pi_2\cdots\pi_n$ and $\sigma = \sigma_1\sigma_2\cdots\sigma_n$ are two permutations of the set $\{1, 2, \ldots, n\}$. Then π *precedes* σ if $\pi_1 < \sigma_1$ or if $\pi_1 = \sigma_1$ and $\pi_2 < \sigma_2$ or if $\pi_1 = \sigma_1$ and $\pi_2 = \sigma_2$ and $\pi_3 < \sigma_3$ or ... or if $\pi_1 = \sigma_1$ and $\pi_2 = \sigma_2$ and ... and $\pi_k = \sigma_k$ and $\pi_{k+1} < \sigma_{k+1}$ or if .... Thus, $\pi = 42135$ precedes $\sigma = 42153$ because $4 = 4, 2 = 2, 1 = 1$, and $3 < 5$. In this lexicographic order, we order as we do words in a dictionary, considering first the first "letter," then in case of ties the second "letter," and so on. The following lists all permutations of $\{1, 2, 3\}$ in lexicographic order:

$$123, 132, 213, 231, 312, 321. \tag{2.12}$$

Notice that in terms of permutations of a set of numbers $\{1, 2, \ldots, n\}$, if $n \leq 9$ the permutations can be thought of "numbers" themselves. For example, the permutation 3241 can be thought of as three-thousand-two-hundred-forty-one. In

[27] This section may be omitted.

these cases, the lexicographic order of the permutations will be equivalent to the increasing order of the "numbers." In (2.12), the permutations in lexicographic order increase from the number one-hundred-twenty-three to three-hundred-twenty-one.

We shall describe an algorithm for listing all permutations in lexicographic order. The key step is to determine, given a permutation $\pi = \pi_1\pi_2\cdots\pi_n$, what permutation comes next. The last permutation in the lexicographic order is $n(n-1)(n-2)\ldots 21$. This has no next permutation in the order. Any other permutation π has $\pi_i < \pi_{i+1}$ for some i. If $\pi_{n-1} < \pi_n$, the next permutation in the order is obtained by interchanging π_{n-1} and π_n. For instance, if $\pi = 43512$, then the next permutation is 43521. Now suppose that $\pi_{n-1} > \pi_n$. If $\pi_{n-2} < \pi_{n-1}$, we rearrange the last three entries of π to obtain the next permutation in the order. Specifically, we consider π_{n-1} and π_n and find the smallest of these which is larger than π_{n-2}. We put this in the $(n-2)$nd position. We then order the remaining two of the last three digits in increasing order. For instance, suppose that $\pi = 15243$. Then $\pi_{n-1} = 4 > 3 = \pi_n$ but $\pi_{n-2} = 2 < 4 = \pi_{n-1}$. Both π_{n-1} and π_n are larger than π_{n-2} and 3 is the smaller of π_{n-1} and π_n. Thus, we put 3 in the third position and put 2 and 4 in increasing order, obtaining the permutation 15324. If π is 15342, we switch 4 into the third position, not 2, since $2 < 3$, and obtain 15423.

In general, if $\pi \neq n, n-1, n-2, \ldots, 2, 1$, there must be a rightmost i so that $\pi_i < \pi_{i+1}$. Then the elements from π_i and on must be rearranged to find the next permutation in the order. This is accomplished by examining all π_j for $j > i$ and finding the smallest such π_j that is larger than π_i. Then π_i and π_j are interchanged. Having made the interchange, the numbers following π_j after the interchange are placed in increasing order. They are now in decreasing order, so simply reversing them will suffice. For instance, suppose that $\pi = 412653$. Then $\pi_i = 2$ and $\pi_j = 3$. Interchanging π_i and π_j gives us 413652. Then reversing gives us 413256, which is the next permutation in the lexicographic order.

The steps of the algorithm are summarized as follows.

Algorithm 2.1: Generating All Permutations of $1, 2, \ldots, n$

Input: n.
Output: A list of all $n!$ permutations of $\{1, 2, \ldots, n\}$, in lexicographic order.

Step 1. Set $\pi = 12\cdots n$ and output π.

Step 2. If $\pi_i > \pi_{i+1}$ for all i, stop. (The list is complete.)

Step 3. Find the largest i so that $\pi_i < \pi_{i+1}$.

Step 4. Find the smallest π_j so that $i < j$ and $\pi_i < \pi_j$.

Step 5. Interchange π_i and π_j.

Step 6. Reverse the numbers following π_j in the new order, let π denote the resulting permutation, output π, and return to step 2.

Note that Algorithm 2.1 can be modified so that as a permutation is generated, it is examined for one purpose or another. For details of a computer implementation of Algorithm 2.1, see, for example, Reingold, Nievergelt, and Deo [1977].

2.16.2 An Algorithm for Generating Subsets of Sets

In Section 2.6 we considered the problem of finding all possible pizzas given a particular set of toppings. This was tantamount to finding all subsets of a given set. In this section we describe an algorithm for doing so.

We start by supposing that S is a subset of the set $\{1, 2, \ldots, n\}$. An equivalent way to denote S is by a bit string B of length n, where a 1 in B's ith spot indicates that i is in S and a 0 in B's ith spot indicates that i is not. For instance, if $S = \{1, 3, 4, 6\} \subseteqq \{1, 2, 3, 4, 5, 6, 7\}$ then $B = 1011010$. Thus, the problem of generating all subsets of an n-set becomes the problem of generating all bit strings of length n.

An ordering similar to the lexicographic ordering that we introduced for permutations will be used for these bit strings. Suppose that $\alpha = \alpha_1\alpha_2\alpha_3$ and $\beta = \beta_1\beta_2\beta_3$ are two bit strings of length 3. We say that α *precedes* β if $\alpha_1 < \beta_1$ or if $\alpha_1 = \beta_1$ and $\alpha_2 < \beta_2$ or if $\alpha_1 = \beta_1$, $\alpha_2 = \beta_2$, and $\alpha_3 < \beta_3$. For instance, $\alpha = 001$ precedes $\beta = 010$ since $\alpha_1 = 0 = \beta_1$ and $\alpha_2 = 0 < 1 = \beta_2$. More generally (and more succinctly), suppose that $\alpha = \alpha_1\alpha_2\cdots\alpha_n$ and $\beta = \beta_1\beta_2\cdots\beta_n$ are two bit strings of length n. Then α precedes β if $\alpha_i < \beta_i$ for the smallest i in which α_i and β_i differ. Thus, $\alpha = 01010$ precedes $\beta = 01100$ because $\alpha_1 = 0 = \beta_1$, $\alpha_2 = 1 = \beta_2$, and $\alpha_3 = 0 < 1 = \beta_3$. In this "lexicographic" order, we again order as we do words in a dictionary; however, in this case we are dealing with a restricted "alphabet," 0 and 1. The following lists all bit strings of length 3 in lexicographic order:

$$000, 001, 010, 011, 100, 101, 110, 111. \tag{2.13}$$

Treating bit strings like numbers and using an increasing ordering is another way to think about a lexicographic ordering of subsets of a set. Notice that in (2.13), the length 3 bit strings go from zero to one-hundred-eleven and increase in between.

Next, we describe an algorithm for listing all bit strings in lexicographic order. Given a bit string, what bit string comes next? Since $11\cdots1$ has no 0's, it will not precede any other bit string by our definition of *precede* above and will thus be last in the order. Any other bit string β has $\beta_i = 0$ for some i. The next bit string after β in the order is obtained by starting at β_n and working backwards, changing all occurrences of 1's to 0's, and vice versa. By stopping the process the first time a 0 is changed to a 1, the next bit string in the order is obtained. For instance, suppose that $\beta = 1001011$. We change the places of β_5, β_6, and β_7 (which are in bold below) to obtain the next bit string, i.e.,

$$1001\mathbf{011} \to 1001\mathbf{100}.$$

Alternatively, if we think of the bit strings as "numbers," then the next bit string after β is the next-largest bit string. This can be found by adding 1 to β. It is not hard to see that adding 1 to β will have the same effect as what was described above.

The steps of this algorithm are summarized as follows.

Algorithm 2.2: Generating All Bit Strings of Length n

Input: n.
Output: A list of all 2^n bit strings of length n, in lexicographic order.

Step 1. Set $\beta = 00\cdots 0$ and output β.

Step 2. If $\beta_i = 1$ for all i, stop. (The list is complete.)

Step 3. Find the largest i so that $\beta_i = 0$.

Step 4. Change β_i to 1 and $\beta_{i+1}, \beta_{i+2}, \ldots, \beta_n$ to 0, let β denote the resulting bit string, output β, and return to step 2.

There are certainly other orderings for bit strings of length n (and permutations of $\{1, 2, \ldots, n\}$) than the ones of the lexicographic type. We describe another ordering of all of the bit strings of length n. This new order in which we will examine these bit strings is called the *binary-reflected Gray code order*.[28] (The reason for the term *binary-reflected* should become clear when we describe the ordering below. The use of the word *code* comes from its connection to coding theory; see Chapter 10.)

The binary-reflected Gray code order for bit strings of length n, denoted $G(n)$, can easily be defined recursively. That is, we will define the binary-reflected Gray code order for bit strings of length n in terms of the binary-reflected Gray code order for bit strings of length less than n. We will use the notation $G_i(n)$ to refer to the ith bit string in the ordering $G(n)$. Normally, a binary-reflected Gray code order begins with the all-0 bit string, and recall that the number of subsets of an n-element set is 2^n. Thus, $G(1)$ is the order that starts with the bit string $0 = G_1(1)$ and ends with the bit string $1 = G_2(1)$, i.e.,

$$G(1) = 0, 1.$$

To find $G(2)$, we list the elements of $G(1)$ and attach a 0 at the beginning of each element. Then list the elements of $G(1)$ in "reverse order" and attach a 1 at the beginning of each of these elements. Thus,

$$G(2) = 0G_1(1), 0G_2(1), 1G_2(1), 1G_1(1) = 00, 01, 11, 10. \tag{2.14}$$

This same procedure is used whether we are going from $G(1)$ to $G(2)$ or from $G(n)$ to $G(n+1)$.

$$G(n+1) = 0G_1(n), 0G_2(n), \ldots, 0G_{2^n}(n), 1G_{2^n}(n), 1G_{2^n-1}(n), \ldots, 1G_1(n). \tag{2.15}$$

Letting $G(n)^R$ be the reverse order of $G(n)$ and with a slight abuse of notation, $G(n+1)$ can be defined as

$$G(n+1) = 0G(n), 1G(n)^R. \tag{2.16}$$

[28] This order is based on work due to Gray [1953].

Thus,

$$G(3) = 0G(2), 1G(2)^R = 000, 001, 011, 010, 110, 111, 101, 100.$$

Notice that we doubled the number of elements in going from $G(1)$ to $G(2)$ and $G(2)$ to $G(3)$. This is not an anomaly. $|G(2)| = 2^2 = 2 \cdot 2^1 = 2|G(1)|$ and in general

$$|G(n+1)| = 2^{n+1} = 2 \cdot 2^n = 2|G(n)|.$$

$G(n+1)$ as defined in Equation (2.15) has $2 \cdot 2^n = 2^{n+1}$ elements and no duplicate elements (Why?). Therefore, the recursively defined binary-reflected Gray code order $G(i)$ is in fact an ordering of all of the bit strings of length i, $i = 1, 2, \ldots$. It is left to the reader (Exercise 15) to find an algorithm for listing all terms in $G(n)$ directly, as opposed to recursively. Again, the key step will be to produce, given a length n bit string $\beta = \beta_1\beta_2\cdots\beta_n$, the length n bit string that comes next in $G(n)$.

Although the lexicographic order for this problem is probably more intuitive, the binary-reflected Gray code order is better in another sense. It is sometimes important that the change between successive elements in an ordering be kept to a minimum. In this regard, the binary-reflected Gray code order is certainly more efficient than the lexicographic order. Successive elements in the binary-reflected Gray code order differ in only one spot. This is obviously best possible. The lexicographic order for bit strings of length n will always have n changes for some pair of successive elements. (Why?) Notice that in (2.13), there are $n = 3$ spots which change when going from 011 to 100.

2.16.3 An Algorithm for Generating Combinations

The next combinatorial objects that we defined in this chapter after permutations and subsets were the r-combinations of an n-set. In terms of the preceding section's subsets of a set, these can be thought of as those subsets of $\{1, 2, \ldots, n\}$ of size exactly r or those bit strings of length n with exactly r 1's. Because of this association, the algorithm for their generation follows closely from Algorithm 2.2.

Suppose that we are interested in generating the 3-combinations of a 5-set. Since we are dealing with subsets, order does not matter. Thus,

$$\{1,2,5\}, \{1,5,2\}, \{2,1,5\}, \{2,5,1\}, \{5,1,2\}, \{5,2,1\}$$

are all considered the same. Therefore, as a matter of choice, our algorithm will generate the r-combinations with each subset's elements in increasing order. So, of the 6 identical subsets listed above, our algorithm will generate $\{1, 2, 5\}$. There are $\binom{5}{3} = 10$ 3-combinations of our 5-set which in increasing lexicographic order are

$$\{1,2,3\}, \{1,2,4\}, \{1,2,5\}, \{1,3,4\}, \{1,3,5\}, \{1,4,5\}, \{2,3,4\}, \{2,3,5\}, \{2,4,5\}, \{3,4,5\}.$$

Our algorithm for generating r-combinations of an n-set works in the following way. Given an r-combination $\gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_r\}$, find the largest i such that $\{\gamma_1, \gamma_2, \ldots, \gamma_{i-1}, \gamma_i + 1, \gamma_{i+1}, \ldots, \gamma_r\}$ is an r-combination whose elements are still in increasing order. Then reset $\gamma_{i+1}, \gamma_{i+2}, \ldots, \gamma_r$ to their lowest possible values. For example, suppose that $\gamma = \{1, 3, 4, 7, 8\}$ is a 5-combination of the 8-set $\{1, 2, \ldots, 8\}$ whose elements are in increasing order. Then, $i = 3$ and $\gamma_i = 4$. Incrementing γ_i by one yields $\{1, 3, 5, 7, 8\}$. Then γ_4 and γ_5 can be reset to 6 and 7, respectively, giving us $\{1, 3, 5, 6, 7\}$, which is the next 5-combination in the increasing lexicographic order. The algorithm starts with $\{1, 2, \ldots, r\}$ and ends with $\{n - r + 1, n - r + 2, \ldots, n\}$ since no element in the latter can be incremented.

This algorithm, like our previous lexicographic algorithms, is not efficient in terms of minimizing changes between successive terms in the order. A binary-reflected Gray code order, similar to the one in the preceding section, is most efficient in this regard. Recall that $G(n+1)$ is defined recursively by (2.16). Next, let $G(n, r)$, $0 \le r \le n$, be the binary-reflected Gray code order for bit strings of length n with exactly r 1's. $G(n, r)$, $0 < r < n$, can be defined recursively by

$$G(n, r) = 0G(n-1, r), 1G(n-1, r-1)^R$$

and

$$G(n, 0) = 0^n = \underbrace{00 \cdots 0}_{n \text{ of them}} \quad \text{and} \quad G(n, n) = 1^n = \underbrace{11 \cdots 1}_{n \text{ of them}}.$$

In the rest of the cases, the first and last bit strings in $G(n, r)$ will be

$$0^{n-r}1^r \text{ and } 10^{n-r}1^{r-1},$$

respectively. For example,

$$\begin{array}{ll}
G(1,0) = 0 & G(2,0) = 00 \\
G(1,1) = 1 & G(2,1) = 01, 10 \\
 & G(2,2) = 11 \\
 & \\
G(3,0) = 000 & G(4,0) = 0000 \\
G(3,1) = 001, 010, 100 & G(4,1) = 0001, 0010, 0100, 1000 \\
G(3,2) = 011, 110, 101 & G(4,2) = 0011, 0110, 0101, 1100, 1010, 1001 \\
G(3,3) = 111 & G(4,3) = 0111, 1101, 1110, 1011 \\
 & G(4,4) = 1111.
\end{array}$$

Since each bit string must contain exactly r 1's, the best that could be hoped for between successive terms of an order is that at most two bits differ. The binary-reflected Gray code order for bit strings of length n with exactly r 1's does in fact attain this minimum.

See Reingold, Nievergelt, and Deo [1977] for more algorithms for generating permutations, subsets, and combinations, in addition to compositions and partitions. An early but comprehensive paper on generating permutations and combinations is Lehmer [1964].

EXERCISES FOR SECTION 2.16

1. For each of the following pairs of permutations, determine which comes first in the lexicographic order.

(a) 3412 and 2143 (b) 3124 and 3214
(c) 234651 and 235164 (d) 76813254 and 76813524

2. For each of the following pairs of bit strings, determine which comes first in the lexicographic order.

   (a) 1001 and 1101 (b) 0101 and 0110
   (c) 11011 and 11101 (d) 110110101 and 110101100

3. For each of the following pairs of combinations, determine which comes first in the increasing lexicographic order.

   (a) $\{1,2,3,4\}$ and $\{1,2,3,5\}$ (b) $\{1,2,6,7,9,16\}$ and $\{1,2,6,7,10,13\}$
   (c) $\{2,4,5,6\}$ and $\{1,3,5,6\}$ (d) $\{8,10,11,12,13\}$ and $\{8,9,10,11,13\}$

4. For each of the following permutations, find the permutation immediately following it in the lexicographic order.

   (a) 123456 (b) 3457621 (c) 152463
   (d) 82617543 (e) 4567321 (f) 54328716

5. For each of the following bit strings, find the bit string immediately following it in the binary-reflected Gray code order.

   (a) 011010 (b) 0101001 (c) 0000
   (d) 11111 (e) 1110111 (f) 011111

6. For each of the following r-combinations of an 8-set, find the combination immediately following it in the increasing lexicographic order.

   (a) $\{1,3,5,6,8\}$ (b) $\{2,3,4,7\}$
   (c) $\{1,3,4,5,6,7\}$ (d) $\{4,5,6,7,8\}$

7. List all permutations of $\{1,2,3,4\}$ in lexicographic order.
8. List all permutations of $\{u,v,x,y\}$ in lexicographic order.
9. List all bit stings of length 4 in lexicographic order.
10. In JAVA (Example 2.8), consider all variables of length 3. If a letter precedes an underscore, an underscore precedes a dollar sign, and all of these precede a decimal digit, find the first and last 3-character variables in lexicographic order.
11. List all 2-combinations of $\{1,2,3,4,5\}$ in increasing lexicographic order.
12. Find a bit string of length n which changes in all n of its spots when producing the next one in lexicographic order.
13. Find $G(4)$.
14. Using the recursive definitions in Equations (2.14) and (2.15), prove that $G(n)$ has no duplicate terms in the order.
15. Find an algorithm for producing all bit strings of length n in binary-reflected Gray code order.
16. Using Algorithm 2.2, prove that the lexicographic order for generating all subsets of a given n-set will always have n changes for some pair of successive terms.
17. Develop an algorithm for generating all permutations of the set $\{1,2,\ldots,n\}$ such that i does not appear in the ith spot of any permutation for all i.

18. Explain the reason for the use of the term "binary-reflected" from the binary-reflected Gray code order. [*Hint:* Refer to the procedure for finding $G(n)$ given by Equation (2.15).]
19. Prove that any two successive bit strings in a binary-reflected Gray code order differ in exactly one position.
20. Find $G(5, r)$, $0 \leq r \leq 5$.
21. Let $f_n(\pi)$ be i if π is the ith permutation in the lexicographic order of all permutations of the set $1, 2, \ldots, n$. Compute:

    (a) $f_2(21)$ (b) $f_3(231)$

    (c) $f_5(15243)$ (d) $f_6(654321)$
22. Suppose that $f_n(\pi)$ is defined as in Exercise 21 and that permutation π' is obtained from permutation $\pi = \pi_1\pi_2 \cdots \pi_n$ by deleting π_1 and reducing by 1 all elements π_j such that $\pi_j > \pi_1$. Show that $f_n(\pi) = (\pi_1 - 1)(n-1)! + f_{n-1}(\pi')$.
23. Find an algorithm for generating all of the r-permutations of an n-set.
24. Using Algorithm 2.1 and the note following the algorithm, find another algorithm for generating all r-combinations of an n-set.
25. Recall the idea of a complexity function from Section 2.4. Calculate:

    (a) The complexity function for Algorithm 2.1

    (b) The complexity function for Algorithm 2.2

    (c) The complexity function for the algorithm which produces all r-combinations of an n-set in increasing lexicographic order

2.17 INVERSION DISTANCE BETWEEN PERMUTATIONS AND THE STUDY OF MUTATIONS[29]

Mutations are a key process by which species evolve. These mutations can occur in the sequences representing DNA. Sometimes the mutations can involve *inversions* (where a segment of DNA is reversed), *transpositions* (where two adjacent segments of DNA exchange places), *translocations* (where the ends of two chromosomes are exchanged), and sometimes they are more complicated. Much recent work has focused on algorithmic analysis of these mutations. For example, see Ferretti, Nadeau, and Sankoff [1996], Hannenhalli and Pevzner [1996], and Kaplan, Shamir, and Tarjan [1997]. We concentrate on inversions here.

Inversions seem to be the dominant form of mutation in some species. For example, inversions play a special role in the evolution of fruit flies (Dobzhansky [1970]), pea plants (Palmer, Osorio, and Thompson [1988]), and certain bacteria (O'Brien [1993]). At a large level, we can study inversions of genes, where genes correspond to subsequences of DNA. Genes are arranged on chromosomes and it is a reasonable starting assumption that genes on a given chromosome are distinguishable, so they

[29] This section closely follows Gusfield [1997]. It may be omitted without loss of continuity.

can be thought of as labeled $1, 2, \ldots, n$ on a given chromosome and as a permutation of $\{1, 2, \ldots, n\}$ after a series of inversions. An inversion then reverses a subsequence of the permutation. For instance, starting with the identity permutation, 123456, we can invert subsequence 2345 to get 154326, then subsequence 15 to get 514326, then subsequence 432 to get 512346. In the study of evolution, we know the current species or DNA sequence or sequence of genes on a chromosome, and try to reconstruct how we got to it. A natural question that occurs, then, is how to find the way to get from one permutation, e.g., 123456, to another, e.g., 512346, in the smallest number of steps, or in our case, the smallest number of inversions. This is called the *inversion distance* of permutation 512346 from permutation 123456, or simply the inversion distance of 512346. We have seen how to get 512346 from 123456 in three inversions. In fact, the inversion distance is at most 2: Invert subsequence 12345 to get 543216 and invert subsequence 4321 to get 512346. Thus, the inversion distance of 512346 equals 2 since it can't be 1. (Why?) We shall describe a heuristic algorithm that looks for the smallest number of inversions of the identity permutation $12 \cdots n$ required to obtain a permutation π of $\{1, 2, \ldots, n\}$, i.e., computes the inversion distance of π.

If π represents a permutation, then π_i represents the number in the ith position in π. In other words, $\pi = \pi_1\pi_2\cdots\pi_n$. For instance, $\pi_3 = 2$ in the permutation 512346. A *breakpoint* in π occurs between two numbers π_i and π_{i+1} if

$$|\pi_i - \pi_{i+1}| \neq 1.$$

Additionally, π has a breakpoint at its front if $\pi_1 \neq 1$ and at its end if $\pi_n \neq n$. So, $\pi = 143265$ has breakpoints between 1 and 4, between 2 and 6, and at its end.

Let $\Phi(\pi)$ equal the total number of breakpoints in π.

Theorem 2.10 The inversion distance of any permutation π is at least $\left\lceil \frac{\Phi(\pi)}{2} \right\rceil$.

Proof. If $\pi = \pi_1\pi_2\cdots\pi_n$ and the subsequence $\pi_i\pi_{i+1}\cdots\pi_{i+j}$ is inverted, then at most two new breakpoints can be created by this inversion. These two new breakpoints could occur between π_{i-1} and π_{i+j} and/or between π_i and π_{i+j+1} in the new permutation

$$\pi^* = \pi_1\pi_2\cdots\pi_{i-1}\pi_{i+j}\pi_{i+j-1}\cdots\pi_{i+1}\pi_i\pi_{i+j+1}\pi_{i+j+2}\cdots\pi_n.$$

Since the identity permutation has no breakpoints, at least $\lceil\Phi(\pi)/2\rceil$ inversions are needed to transform the identity permutation into π (or vice versa). Q.E.D.

It was noted above that permutation 143265 has three breakpoints. Therefore, by Theorem 2.10, its inversion distance is at least $\lceil 3/2 \rceil = 2$.

The subsequence in a permutation between a breakpoint and (a) the front of the permutation, (b) another breakpoint, or (c) the end of the permutation, with no other breakpoint between them, is called a *strip*. If the numbers in the strip are increasing (decreasing), the strip is called *increasing* (*decreasing*). Strips consisting of a single number are defined to be decreasing. For example, the permutation

541236 has 3 breakpoints and 3 strips. The two decreasing strips are 54 and 6, while the increasing strip is 123.

The problem of finding the inversion distance of a permutation has been shown to be hard.[30] Sometimes, methods that come close to the inversion distance—or at least within a fixed factor of it—can come in handy. The next two lemmas can be used to give an algorithm for transforming any permutation into the identity permutation using a number of inversions that is at most four times the inversion distance.

Lemma 2.1 If permutation π contains a decreasing strip, then there is an inversion that decreases the number of breakpoints.

Proof.[31] Consider the decreasing strip with the smallest number π_i contained in any decreasing strip. By definition, π_i is at the right end of this strip. If $\pi_i = 1$, then $\pi_1 \neq 1$, in which case there must be breakpoints before π_1 and after π_i. Inverting $\pi_1\pi_2\cdots\pi_i$ reduces the number of breakpoints by at least one since 1 moves into the first spot of the inverted permutation.

Suppose that $\pi_i \neq 1$. Consider π_{i+1}, if it exists. It cannot be $\pi_i - 1$ since otherwise it would be in a decreasing strip. It also cannot be $\pi_i + 1$, for otherwise π_i would not be in a decreasing strip. So there must be a breakpoint between π_i and π_{i+1} or after π_i (i.e., at the end of the entire permutation) if π_{i+1} doesn't exist. By similar reasoning, there must be a breakpoint immediately to the right of $\pi_i - 1$.

If $\pi_i - 1$ is located to the right of π_i then invert $\pi_{i+1}\pi_{i+2}\cdots\pi_i - 1$. And if $\pi_i - 1$ is located to the left of π_i then invert starting with the term immediately to the right of $\pi_i - 1$ through π_i. In either case, we are inverting a subsequence with breakpoints at both its ends and reducing the number of breakpoints by at least one since $\pi_i - 1$ and π_i are now consecutive elements in the inverted permutation. Q.E.D.

For example, 54 in the permutation 541236 is the decreasing strip with the smallest number, 4. Locate the number 3 and, as must be the case, it is in an increasing sequence with a breakpoint immediately to its right. Invert 123 to get 543216. This new permutation has 2 breakpoints whereas the original had 3.

Lemma 2.2 If permutation π is not the identity and contains no decreasing strips, then there is an inversion that does not increase the number of breakpoints but creates a decreasing strip.

Proof.[32] Since there are no decreasing strips, every strip must be increasing. If $\pi_1 \neq 1$ or $\pi_n \neq n$, then inverting the increasing strip leading from π_1 or leading to π_n, respectively, will satisfy the lemma. Otherwise, find the first and second breakpoints after $\pi_1 = 1$. These exist since π is not the identity permutation. The subsequence between these two breakpoints satisfies the lemma. Q.E.D.

---

[30] Inversion distance calculation is NP-hard (Caprara [1997]), using the language of Section 2.18.

[31] The proof may be omitted.

[32] The proof may be omitted.

Consider the *inversion algorithm* that works as follows:

Step 1. If there is a decreasing strip, use the inversion of Lemma 2.1. Repeat until there is no decreasing strip.

Step 2. Use the inversion of Lemma 2.2 and return to Step 1.

The number of breakpoints is decreased at least once every two inversions. By Theorem 2.10, we have the following theorem.

Theorem 2.11 (Kececioglu and Sankoff [1994]) The inversion algorithm transforms any permutation into the identity permutation using a number of inversions that is at most four times the inversion distance.

The bound of using at most four times the number of inversions as the optimal is not the best known. Kececioglu and Sankoff [1995] were able to reduce the error bound in half by proving the following lemma.

Lemma 2.3 Let π be a permutation with a decreasing strip. If every inversion that reduces the number of breakpoints of π leaves a permutation with no decreasing strips, then π has an inversion that reduces the number of breakpoints by two.

With this new lemma, we know that, essentially, there always exist two successive inversions that reduce the number of breakpoints by 2. This means that we can reach the identity permutation with at most $\Phi(\pi)$ inversions, which is at most twice the inversion distance (by Theorem 2.10). Bafna and Pevzner [1996] have lowered the Kececioglu and Sankoff [1995] bound to 1.75 by considering the effects of an inversion on future inversions.

Inversion is only one type of mutation but an important one, especially in organisms of one chromosome (Sessions [1990]). Other inversion variants are addressed in the exercises. Transpositions and translocations are also interesting ways to modify a permutation. We say a few words about the former. A transposition $\pi_i\pi_{i+1}$ of the permutation $\pi = \pi_1\pi_2\cdots\pi_n$ results in the permutation $\pi_1\pi_2\cdots\pi_{i-1}\pi_{i+1}\pi_i\pi_{i+2}\cdots\pi_n$. Changing the identity permutation into another permutation (or vice versa) by transpositions of this form is a well-studied problem in combinatorics. The number of transpositions needed to do this is readily established. According to a well-known formula (see, e.g., Jerrum [1985]), the number of such transpositions required to switch an identity permutation into the permutation π of $\{1, 2, \ldots, n\}$ is given by

$$J(\pi) = |\{(i,\ j) : 1 \leq i < j \leq n \ \&\ \pi_i > \pi_j\}|. \tag{2.17}$$

The proof of this is left to the reader (Exercise 8). To illustrate the result, we note that the number of transpositions needed to change the permutation $\pi = 143652$ into the identity is 6 since $J(\pi) = |\{(2,3),(2,6),(3,6),(4,5),(4,6),(5,6)\}|$. More general work, motivated by considerations of mutations, allows transpositions of entire segments of a permutation. For example, we could transform 15823647 into 13658247 by transposing segments 582 and 36. For references, see the papers by Bafna and Pevzner [1998], Christie [1996], and Heath and Vergara [1998]. Transpositions have also arisen in the work of Mahadev and Roberts [2003] in applications to channel assignments in communications and physical mapping of DNA.

EXERCISES FOR SECTION 2.17

1. Which of the following permutations have inversion distance 1?

   (a) 4567123 (b) 12354678 (c) 54321

2. Which of the following permutations have inversion distance 2?

   (a) 4567123 (b) 23456781 (c) 54321

3. Give an example of a permutation with 3 breakpoints but inversion distance 3. (Note that this is another example of the fact that the bound in Theorem 2.10 is not the best possible.)
4. Prove that a strip is always increasing or decreasing.
5. Consider the following greedy algorithm for inversion distance. First find and apply the inversion that brings 1 into position π_1. Next find and apply the inversion that brings 2 into position π_2. And so on.

   (a) Apply this greedy algorithm to 512346.

   (b) Prove that this algorithm ends in at most $\Phi(\pi)$ inversions.

   (c) Find a permutation of $1, 2, \ldots, n$ that requires $n-1$ inversions using this greedy algorithm.

   (d) What is the inversion distance of your permutation in part (c)?

6. (Kececioglu and Sankoff [1995]) In the *signed inversion problem*, each number in a permutation has a sign ($+$ or $-$) that changes every time the number is involved in an inverted subsequence. For example, starting with the permutation $+1\ -5\ +4\ +3\ -2\ -6$, we can invert 543 to get $+1\ -3\ -4\ +5\ -2\ -6$. The signed inversion distance problem is to use the minimum number of inversions to transform a signed permutation into the identity permutation whose numbers have positive sign.

   (a) Give a series of inversions that transform $-3\ -4\ -6\ +7\ +8\ +5\ -2\ -1$ into the all-positive identity permutation.

   (b) In a signed permutation π, an *adjacency* is defined as a pair of consecutive numbers of the form $+i\ +(i+1)$ or $-(i+1)\ -i$. A *breakpoint* is defined as occurring between any two consecutive numbers that do not form an adjacency. Also, there is a breakpoint at the front of π unless the first number is $+1$, and there is a breakpoint at the end of π unless the last number is $+n$.

      i. How many breakpoints does $-3\ -4\ -6\ +7\ +8\ +5\ -2\ -1$ have?

      ii. Describe a bound for signed inversion distance in terms of breakpoints analogous to Theorem 2.10.

7. Find the minimum number of transpositions that transform the following permutations into the identity and identify which transpositions achieve the minimum in each case.

   (a) 54321 (b) 15423 (c) 625143

8. Prove (2.17) (Jerrum's formula).

9. Recall that *mutations due to transpositions* occur when two adjacent *segments* of DNA exchange places. Find segment transpositions that transform the following permutations into the identity permutation. What is the minimum number of such transpositions?

   (a) 5123746 (b) 987123456

10. Suppose that an organism has a circular chromosome. In this case, the permutation $\pi_1, \pi_2, \ldots, \pi_n$ is equivalent to $\pi_i, \pi_{i+1}, \ldots, \pi_n, \pi_1, \pi_2, \ldots, \pi_{i-1}$. We call such a permutation *connected*. Find a series of inversions that transform the following connected permutations into the identity permutation. What is the minimum number of inversions?

    (a) 5123746 (b) 987123456

2.18 GOOD ALGORITHMS[33]

2.18.1 Asymptotic Analysis

We have already observed in Section 2.4 that some algorithms for solving combinatorial problems are not very good. In this section we try to make precise what we mean by a good algorithm. As we pointed out in Section 2.4, the cost of running a particular computer program on a particular machine will vary with the skill of the programmer and the characteristics of the machine. Thus, in the field of computer science, the emphasis is on analyzing algorithms for solving problems rather than on analyzing particular computer programs, and that will be our emphasis here.

In analyzing how good an algorithm is, we try to estimate a complexity function $f(n)$, to use the terminology of Section 2.4. If n is relatively small, then $f(n)$ is usually relatively small, too. Most any algorithm will suffice for a small problem. We shall be mainly interested in comparing complexity functions $f(n)$ for n relatively large.

The crucial concept in the analysis of algorithms is the following. Suppose that F is an algorithm with complexity function $f(n)$ and that $g(n)$ is any function of n. We write that F or f is $O(g)$, and say that F or f is "big oh of g" if there is an integer r and a positive constant k so that for all $n \geq r$, $f(n) \leq k \cdot g(n)$. [If f is $O(g)$, we sometimes say that g *asymptotically dominates* f.] If f is $O(g)$, then for problems of input size at least r, an algorithm with complexity function f will never be more than k times as costly as an algorithm with complexity function g. To give some examples, $100n$ is $O(n^2)$ because for $n \geq 100$, $100n \leq n^2$. Also, $n + 1/n$ is $O(n)$, because for $n \geq 1$, $n + 1/n \leq 2n$. An algorithm that is $O(n)$ is called *linear*, an algorithm that is $O(n^2)$ is called *quadratic*, and an algorithm that is $O(g)$ for g a polynomial is called *polynomial*. Other important classes of algorithms in computer science are algorithms that are $O(\log n)$, $O(n \log n)$, $O(c^n)$ for $c > 1$, and $O(n!)$. We discuss these below or in the exercises.

[33] This section should be omitted in elementary treatments.

An algorithm whose complexity function is $c^n, c > 1$, is called *exponential.* Note that every exponential algorithm is $O(c^n)$, but not every $O(c^n)$ algorithm is exponential. For example, an algorithm whose complexity function is n is $O(c^n)$ for any $c > 1$. This is because $n \leq c^n$ for n sufficiently large.

A generally accepted principle is that an algorithm is *good* if it is polynomial. This idea is originally due to Edmonds [1965]. See Garey and Johnson [1979], Lawler [1976], or Reingold, Nievergelt, and Deo [1977] for a discussion of good algorithms. We shall try to give a very quick justification here.[34]

Since we are interested in $f(n)$ and $g(n)$ only for n relatively large, we introduce the constant r in defining the concept "f is $O(g)$." But where does the constant k come from? Consider algorithms F and G whose complexity functions are, respectively, $f(n) = 20n$ and $g(n) = 40n$. Now clearly algorithm F is preferable, because $f(n) \leq g(n)$ for all n. However, if we could just improve a particular computer program for implementing algorithm G so that it would run in $\frac{1}{2}$ the time, or if we could implement G on a faster machine so that it would run in $\frac{1}{2}$ the time, then $f(n)$ and $g(n)$ would be the same. Since the constant $\frac{1}{2}$ is independent of n, it is not farfetched to think of improvements by this constant factor to be a function of the implementation rather than of the algorithm. In this sense, since $f(n)/g(n)$ equals a constant, that is, since $f(n) = kg(n)$, the functions $f(n)$ and $g(n)$ are considered the same for all practical purposes.

Now, to say that f is $O(g)$ means that $f(n) \leq kg(n)$ (for n relatively large). Since $kg(n)$ and $g(n)$ are considered the same for all practical purposes, $f(n) \leq kg(n)$ says that $f(n) \leq g(n)$ for all practical purposes. Thus, to say that f is $O(g)$ says that an algorithm of complexity g is no more efficient than an algorithm of complexity f.

Before justifying the criterion of polynomial boundedness, we summarize some basic results in the following theorem.

Theorem 2.12

(a) If c is a positive constant, then f is $O(cf)$ and cf is $O(f)$.

(b) n is $O(n^2)$, n^2 is $O(n^3), \ldots, n^{p-1}$ is $O(n^p), \ldots$. However, n^p is not $O(n^{p-1})$.

(c) If $f(n) = a_q n^q + a_{q-1}n^{q-1} + \cdots + a_0$ is a polynomial of degree q, with $a_q > 0$, and if $a_i \geq 0$, all i, then f is $O(n^q)$.

(d) If $c > 1$ and $p \geq 0$, then n^p is $O(c^n)$. Moreover, c^n is not $O(n^p)$.

Part (a) of Theorem 2.12 shows that just as we have assumed, algorithms of complexity f and cf are considered equally efficient. Part (b) asserts that an $O(n^p)$ algorithm is more efficient the smaller the value of p. Part (c) asserts that the degree of the polynomial tells the relative complexity of a polynomial algorithm. Part (d) asserts that polynomial algorithms are always more efficient than exponential algorithms. This is why polynomial algorithms are treated as *good*, whereas

[34] The reader who only wants to understand the definition may skip the rest of this subsection.

Table 2.11: Growths of Different Complexity Functions

	Complexity function $f(n)$			
Input size n	n	n^2	$10n^2$	2^n
5	5	25	250	32
10	10	10^2	10^3	$1{,}024 \approx 1.02 \times 10^2$
20	20	400	4,000	$1{,}048{,}576 \approx 1.05 \times 10^6$
30	30	900	9,000	$\approx 1.07 \times 10^9$
50	50	2,500	25,000	$\approx 1.13 \times 10^{15}$
$100 = 10^2$	10^2	10^4	10^5	$\approx 1.27 \times 10^{30}$
$1{,}000 = 10^3$	10^3	10^6	10^7	$> 10^{300}$
$10{,}000 = 10^4$	10^4	10^8	10^9	$> 10^{3000}$

exponential ones are not. The results of Theorem 2.12 are vividly demonstrated in Table 2.11, which shows how rapidly different complexity functions grow. Notice how much faster the exponential complexity function 2^n grows in comparison to the other complexity functions.

Proof of Theorem 2.12.

(a) Clearly, cf is $O(f)$. Take $k = c$. Next, f is $O(cf)$ because $f(n) \leq (1/c)cf(n)$ for all n.

(b) Since $n^p \geq n^{p-1}$ for $n \geq 1$, n^{p-1} is $O(n^p)$. Now, n^p is not $O(n^{p-1})$. For $n^p \leq cn^{p-1}$ only for $n \leq c$.

(c) Note that since $a_i \geq 0$, $a_i n^i \leq a_i n^q$, for all i and all $n \geq 1$. Hence, it follows that $f(n) \leq (a_0 + a_1 + \cdots + a_q)n^q$, for all $n \geq 1$.

(d) This is a standard result from calculus or advanced calculus. It can be derived by noting that $n^p/c^n \to 0$ as $n \to \infty$. This result is obtained by applying l'Hôspital's rule (from calculus) p times. Since $n^p/c^n \to 0$, $n^p/c^n \leq k$ for n sufficiently large. A similar analysis shows that $c^n/n^p \to \infty$ as $n \to \infty$, so c^n could not be $\leq kn^p$ for $n \geq r$ and constant k. Q.E.D.

The proof of part (d) alludes to the fact that limits can be used to prove whether or not f is "big oh" of g. It is important to note that $f(n)$ and $g(n)$ should not be considered general functions of n but as nonnegative functions of n. This is the case since they measure the cost or complexity of an algorithm.

Theorem 2.13 If $g(n) > 0$ and $\displaystyle\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists, then f is $O(g)$.

Proof. Suppose that $\lim_{n\to\infty} \frac{f(n)}{g(n)} = L$. Then for every $\epsilon > 0$, there exists $N > 0$, such that $\left|\frac{f(n)}{g(n)} - L\right| < \epsilon$ whenever $n > N$. Note that

$$\begin{gathered}
\left|\frac{f(n)}{g(n)} - L\right| < \epsilon \\
\Downarrow \\
\frac{f(n)}{g(n)} - L < \epsilon \\
\Downarrow \\
f(n) < \epsilon g(n) + Lg(n) \\
\Downarrow \\
f(n) < (\epsilon + L)g(n).
\end{gathered}$$

Using the definition of "big oh," the proof is completed by letting $r = N$ and $k = \epsilon + L$. Note that the proof uses the fact that $g(n) > 0$. Q.E.D.

Using Theorem 2.13 and l'Hôspital's rule from calculus, we see that $f(n) = 7\log n$ is $O(n^5)$ since

$$\lim_{n\to\infty} \frac{7\log n}{n^5} = \lim_{n\to\infty} \frac{7\left(\frac{1}{n}\right)}{5n^4} = \lim_{n\to\infty} \frac{7}{5n^5} = 0.$$

If the limit does not exist in Theorem 2.13, no conclusion can be drawn. Consider $f(n) = \sin n + 1$ and $g(n) = \cos n + 3$. Then $\lim_{n\to\infty} \frac{\sin n + 1}{\cos n + 3}$ doesn't exist. However, it is easy to see that

$$\sin n + 1 \leq k(\cos n + 3)$$

for $k = 1$ and all $n \geq 1$. Thus, f is $O(g)$. On the other hand, as we saw in the proof of Theorem 2.12(d), $\lim_{n\to\infty} \frac{c^n}{n^p} \to \infty$ as $n \to \infty$, which implies that c^n could not be $\leq kn^p$ for $n \geq r$ and constant k.

Before closing this subsection, we should note again that our results depend on the crucial "equivalence" between algorithms of complexities f and cf, and on the idea that the size n of the input is relatively large. In practice, an algorithm of complexity $100n$ is definitely worse than an algorithm of complexity n. Moreover, it is also definitely worse, for small values of n, than an algorithm of complexity 2^n. Thus, an $O(n)$ algorithm, in practice, can be worse than an $O(2^n)$ algorithm. The results of this section, and the emphasis on polynomial algorithms, must be interpreted with care.

2.18.2 NP-Complete Problems

In studying algorithms, it is convenient to distinguish between deterministic procedures and nondeterministic ones. An algorithm may be thought of as passing from state to state(s). A *deterministic algorithm* may move to only one new state at a time, while a *nondeterministic algorithm* may move to several new states at once. That is, a nondeterministic algorithm may explore several possibilities simultaneously. In this book we concentrate exclusively on deterministic algorithms, and indeed, when we use the term *algorithm*, we shall mean deterministic. The class of problems for which there is a deterministic algorithm whose complexity is polynomial is called P. The class of problems for which there is a nondeterministic algorithm whose complexity is polynomial is called NP. Clearly, every problem in P is also in NP. To this date, no one has discovered a problem in NP that can be shown not to be in P. However, there are many problems known to be in NP that may or may not be in P. Many of these problems are extremely common and seemingly difficult problems, for which it would be very important to find a deterministic polynomial algorithm. Cook [1971] discovered the remarkable fact that there were some problems L, known as NP-*hard* problems, with the following property: If L can be solved by a deterministic polynomial algorithm, then so can every problem in NP. The traveling salesman problem discussed in Example 2.10 is such an NP-hard problem. Indeed, it is an NP-*complete* problem, an NP-hard problem that belongs to the class NP. Karp [1972] showed that there were a great many NP-complete problems. Now many people doubt that every problem for which there is a nondeterministic polynomial algorithm also will have a deterministic polynomial algorithm. Hence, they doubt whether it will ever be possible to find deterministic polynomial algorithms for such NP-hard (NP-complete) problems as the traveling salesman problem. Thus, NP-hard (NP-complete) problems are hard in a very real sense. See Garey and Johnson [1979] for a comprehensive discussion of NP-completeness. See also Reingold, Nievergelt, and Deo [1977].

Since real-world problems have to be solved, we cannot simply stop seeking a solution when we find that a problem is NP-complete or NP-hard. We make compromises, for instance by dealing with special cases of the problem that might not be NP-hard. For example, we could consider the traveling salesman problem only when the two cheapest links are available when leaving any city; or when, upon leaving a city, only the five closest cities are considered. Alternatively, we seek good algorithms that approximate the solution to the problem with which we are dealing. An increasingly important activity in present-day combinatorics is to find good algorithms that come close to the (optimal) solution to a problem.

EXERCISES FOR SECTION 2.18

1. In each of the following cases, determine if f is $O(g)$ and justify your answer *from the definition.*

   (a) $f = 2^n$, $g = 5^n$ (b) $f = 6n + 2/n$, $g = n^2$
   (c) $f = 10n$, $g = n^2$ (d) $f = 4^{2n}$, $g = 4^{5n} - 50$
   (e) $f = \frac{1}{5}n^4$, $g = n^2 + 7$ (f) $f = \cos n$, $g = 4$

2. Use limits to determine if f is $O(g)$.

(a) $f = 3^n$, $g = 4^n$ (b) $f = n + 2/n$, $g = n^3$
(c) $f = 10n$, $g = n^2$ (d) $f = 3^{2n}$, $g = 3^5 n - 100$
(e) $f = n!$, $g = e^n$ (*Hint:* Recall Stirling's approximation from Section 2.3.)

3. Prove that if f is $O(h)$ and g is $O(h)$, then $f + g$ is $O(h)$.
4. Prove that if f is $O(g)$ and g is $O(h)$, then f is $O(h)$.
5. Prove that if f is $O(g)$ and g is $O(f)$, then for all h, h is $O(f)$ if and only if h is $O(g)$.
6. (a) Show that $\log_2 n$ is $O(n)$. (b) Show that n is not $O(\log_2 n)$.
7. Suppose that $c > 1$.

   (a) Show that c^n is $O(n!)$. (b) Show that $n!$ is not $O(c^n)$.
8. (a) Is it true that n is $O(n \log_2 n)$? Why?

   (b) Is it true that $n \log_2 n$ is $O(n)$? Why?
9. For each of the following functions f and g, determine if f is $O(g)$ and justify your answer. You may use the definition, limits, a theorem from the text, or the result of a previous exercise.

   (a) $f = 5n$, $g = n^3$ (b) $f = n^3$, $g = 5n$
   (c) $f = 6n^2 + 2n + 1$, $g = n^2$ (d) $f = n^3$, $g = 2^n$
   (e) $f = n^3$, $g = n^5$ (f) $f = 7n^7$, $g = 3^n$
   (g) $f = 10 \log_2 n$, $g = n$ (h) $f = n^2 + 2^n$, $g = 4^n$
   (i) $f = \log_2 n + 25$, $g = n^3$ (j) $f = n^2$, $g = 2^n + n$
   (k) $f = 7^n$, $g = 4n^{50} + 25n^{10} + 100n$ (l) $f = 3n^4 + 2n^2 + 1$, $g = 2^n$
   (m) $f = n + n \log_2 n$, $g = 2^n$ (n) $f = n^3 + 3^n$, $g = n!$
10. Explain what functions are $O(1)$, that is, $O(g)$ for $g \equiv 1$.
11. Let $f(n) = \sum_{i=1}^{n} i$. Show that f is $O(n^2)$.
12. Which of the following functions are "big oh" of the others: $4n \log_2 n + \log_2 n$, $n^2 + 3/n^2$, $\log_2 \log_2 n$.
13. Repeat Exercise 12 for $(\log_2 n)^2$, 2^n, and $4n^2 \log_2 n + 5n$.

2.19 PIGEONHOLE PRINCIPLE AND ITS GENERALIZATIONS

2.19.1 The Simplest Version of the Pigeonhole Principle

In combinatorics, one of the most widely used tools for proving that a certain kind of arrangement or pattern *exists* is the *pigeonhole principle*. Stated informally, this principle says the following: If there are "many" pigeons and "few" pigeonholes, then there must be two or more pigeons occupying the same pigeonhole. This principle is also called the *Dirichlet drawer principle*,[35] the *shoebox principle*, and by

[35] Although the origin of the pigeonhole principle is not clear, it was widely used by the nineteenth-century mathematician Peter Dirichlet.

other names. It says that if there are many objects (shoes) and few drawers (shoeboxes), then some drawer (shoebox) must have two or more objects (shoes). We present several variants of this basic combinatorial principle and several applications of it. Note that the pigeonhole principle simply states that there must *exist* two or more pigeons occupying the same pigeonhole. It does not help us to identify such pigeons.

Let us start by stating the pigeonhole principle more precisely.

Theorem 2.14 (Pigeonhole Principle) If $k + 1$ pigeons are placed into k pigeonholes, then at least one pigeonhole will contain two or more pigeons.

To illustrate Theorem 2.14, we note that if there are 13 people in a room, at least two of them are sure to have a birthday in the same month. Similarly, if there are 677 people chosen from the telephone book, then there will be at least two whose first and last names begin with the same letter. The next two examples are somewhat deeper.

Example 2.33 Scheduling Meetings of Legislative Committees (Example 1.4 Revisited) Consider the meeting schedule problem of Example 1.4. A *clique* consists of a set of committees each pair of which have a member in common. The *clique number* corresponding to the set of committees is the size of the largest clique. Given the data of Table 1.5, the largest clique has size 3. The cliques of size 3 correspond to the triangles in the graph of Figure 1.1. Since all committees in a clique must receive different meeting times, the pigeonhole principle says that the number of meeting times required is at least as large as the size of the largest clique. To see why, let the vertices of a clique be the pigeons and the meeting times be pigeonholes. (In the language of Chapter 3, this conclusion says that the chromatic number of a graph is always at least as big as the clique number.) ■

Example 2.34 Manufacturing Personal Computers A manufacturer of personal computers (PCs) makes at least one PC every day over a period of 30 days, doesn't start a new PC on a day when it is impossible to finish it, and averages no more than $1\frac{1}{2}$ PCs per day. Then there must be a period of consecutive days during which *exactly* 14 PCs are made. To see why, let a_i be the number of PCs made through the end of the ith day. Since at least one PC is made each day, and at most 45 PCs in 30 days, we have

$$\begin{aligned} a_1 &< a_2 < \cdots < a_{30}, \\ a_1 &\geq 1, \\ a_{30} &\leq 45. \end{aligned}$$

Also,

$$a_1 + 14 < a_2 + 14 < \cdots < a_{30} + 14 \leq 45 + 14 = 59.$$

Now consider the following numbers:

$$a_1, a_2, \ldots, a_{30}, a_1 + 14, a_2 + 14, \ldots, a_{30} + 14.$$

These are 60 numbers, each between 1 and 59. By the pigeonhole principle, two of these numbers are equal. Since $a_1, a_2, \ldots, a_{30}$ are all different and $a_1 + 14, a_2 + 14, \ldots, a_{30} + 14$ are all different, there exist i and j $(i \neq j)$ so that $a_i = a_j + 14$. Thus, between days i and j, the manufacturer makes exactly 14 PCs. ■

2.19.2 Generalizations and Applications of the Pigeonhole Principle

To begin with, let us state some stronger versions of the pigeonhole principle. In particular, if $2k + 1$ pigeons are placed into k pigeonholes, at least one pigeonhole will contain more than two pigeons. This result follows since if every pigeonhole contains at most two pigeons, then since there are k pigeonholes, there would be at most $2k$ pigeons in all. By the same line of reasoning, if $3k + 1$ pigeons are placed into k pigeonholes, then at least one pigeonhole will contain more than three pigeons.

Speaking generally, we have the following theorem.

Theorem 2.15 If m pigeons are placed into k pigeonholes, then at least one pigeonhole will contain more than

$$\left\lfloor \frac{m-1}{k} \right\rfloor$$

pigeons.

Proof. If the largest number of pigeons in a pigeonhole is at most $\lfloor (m-1)/k \rfloor$, the total number of pigeons is at most

$$k \left\lfloor \frac{m-1}{k} \right\rfloor \leq m - 1 < m.$$

Q.E.D.

To illustrate Theorem 2.15, note that if there are 40 people in a room, a group of more than three will have a common birth month, for $\lfloor 39/12 \rfloor = 3$. To give another application of this result, suppose we know that a computer's memory has a capacity of 8000 bits in eight storage locations. Then we know that we have room in at least one location for at least 1000 bits. For $m = 8000$ and $k = 8$, so $\lfloor (m-1)/k \rfloor = 999$. Similarly, if a factory has 40 electrical outlets with a total 9600-volt capacity, we know there must be at least one outlet with a capacity of 240 or more volts.

These last two examples illustrate the following corollary of Theorem 2.15, whose formal proof is left to the reader (Exercise 21).

Corollary 2.15.1 The average value of a set of numbers is between the smallest and the largest of the numbers.

An alternative way to state this corollary is:

Corollary 2.15.2 Given a set of numbers, there is always a number in the set whose value is at least as large (at least as small) as the average value of the numbers in the set.

Example 2.35 Scheduling Meetings of Legislative Committees (Example 2.33 Revisited) To continue with Example 2.33, let us say that a set of committees is an *independent set* if no pair of them have a member in common. An assignment of meeting times partitions the n committees into k disjoint groups each of which is an independent set. The average size of such an independent set is n/k. Thus, there is at least one independent set of size at least n/k. In the data of Table 1.5, $n = 6$, $k = 3$, and $n/k = 2$. If there are 8 committees and 3 meeting times, we know that at least one meeting time will have at least 8/3, i.e., at least 3, committees. ■

Example 2.36 Web Servers A company has 15 web servers and 10 Internet ports. We never expect more than 10 web servers to need a port at any one time. Every 5 minutes, some subset of the web servers requests ports. We wish to connect each web server to some of the ports, in such a way that we use as few connections as possible, but we are always sure that a web server will have a port to access. (A port can be used by at most one web server at a time.) How many connections are needed? To answer this question, note that if there are fewer than 60 connections, the average port will have fewer than six connections, so by Corollary 2.15.2, there will be some port that will be connected to five or fewer web servers. If the remaining 10 web servers were used at one time, there would be only 9 ports left for them. Thus, at least 60 connections are required. It is left to the reader (Exercise 19) to show that there is an arrangement with 60 connections that has the desired properties. ■

Another application of Theorem 2.15 is a result about increasing and decreasing subsequences of a sequence of numbers. Consider the sequence of numbers $x_1, x_2, \ldots, x_p$. A *subsequence* is any sequence $x_{i_1}, x_{i_2}, \ldots, x_{i_q}$ such that $1 \leq i_1 < i_2 < \cdots < i_q \leq p$. For instance, if $x_1 = 9, x_2 = 6, x_3 = 14, x_4 = 8$, and $x_5 = 17$, then we have the sequence 9, 6, 14, 8, 17; the subsequence x_2, x_4, x_5 is the sequence 6, 8, 17; the subsequence x_1, x_3, x_4, x_5 is the sequence 9, 14, 8, 17; and so on. A subsequence is *increasing* if its entries go successively up in value, and *decreasing* if its entries go successively down in value. In our example, a longest increasing subsequence is 9, 14, 17, and a longest decreasing subsequence is 14, 8.

To give another example, consider the sequence

$$12, 5, 4, 3, 8, 7, 6, 11, 10, 9.$$

A longest increasing subsequence is 5, 8, 11 and a longest decreasing subsequence is 12, 11, 10, 9. These two examples illustrate the following theorem, whose proof depends on Theorem 2.15.

Theorem 2.16 (Erdős and Szekeres [1935]) Given a sequence of n^2+1 *distinct* integers, either there is an increasing subsequence of $n+1$ terms or a decreasing subsequence of $n + 1$ terms.

Note that $n^2 + 1$ is required for this theorem; that is, the conclusion can fail for a sequence of fewer than $n^2 + 1$ integers. For example, consider the sequence

$$3, 2, 1, 6, 5, 4, 9, 8, 7.$$

This is a sequence of 9 integers arranged so that the longest increasing subsequences and the longest decreasing subsequences are 3 terms long.

Proof of Theorem 2.16. Let the sequence be

$$x_1, x_2, \ldots, x_{n^2+1}.$$

Let t_i be the number of terms in the longest increasing subsequence beginning at x_i. If any t_i is at least $n + 1$, the theorem is proved. Thus, assume that each t_i is between 1 and n. We therefore have $n^2 + 1$ pigeons (the $n^2 + 1$ t_i's) to be placed into n pigeonholes (the numbers $1, 2, \ldots, n$). By Theorem 2.15, there is a pigeonhole containing at least

$$\left\lfloor \frac{(n^2+1)-1}{n} \right\rfloor + 1 = n + 1$$

pigeons. That is, there are at least $n + 1$ t_i's that are equal. We shall show that the x_i's associated with these t_i's form a decreasing subsequence. For suppose that $t_i = t_j$, with $i < j$. We shall show that $x_i > x_j$. If $x_i \leq x_j$, then $x_i < x_j$ because of the hypothesis that the $n^2 + 1$ integers are all distinct. Then x_i followed by the longest increasing subsequence beginning at x_j forms an increasing subsequence of length $t_j + 1$. Thus, $t_i \geq t_j + 1$, which is a contradiction. Q.E.D.

To illustrate this proof, let us consider the following sequence of 10 distinct integers:

$$10, 3, 2, 1, 6, 5, 4, 9, 8, 7.$$

Here $n = 3$ since $10 = 3^2 + 1$, and we have

i	x_i	t_i	Sample subsequence
1	10	1	10
2	3	3	3, 6, 7
3	2	3	2, 6, 7
4	1	3	1, 6, 7
5	6	2	6, 7

i	x_i	t_i	Sample subsequence
6	5	2	5, 7
7	4	2	4, 7
8	9	1	9
9	8	1	8
10	7	1	7

Hence, there are four 1's among the t_i's, and the corresponding x_i's, namely x_1, x_8, x_9, x_{10}, form a decreasing subsequence, 10, 9, 8, 7.

We close this section by stating one more generalization of the pigeonhole principle, whose proof we leave to the reader (Exercise 22).

Theorem 2.17 Suppose that $p_1, p_2, \ldots, p_k$ are positive integers. If

$$p_1 + p_2 + \cdots + p_k - k + 1$$

pigeons are put into k pigeonholes, then either the first pigeonhole contains at least p_1 pigeons, or the second pigeonhole contains at least p_2 pigeons, or $\ldots$, or the kth pigeonhole contains at least p_k pigeons.

2.19.3 Ramsey Numbers

One simple application of the version of the pigeonhole principle stated in Theorem 2.15 is the following.

Theorem 2.18 Assume that among 6 persons, each pair of persons are either friends or enemies. Then either there are 3 persons who are mutual friends or 3 persons who are mutual enemies.

Proof. Let a be any person. By the pigeonhole principle, of the remaining 5 people, either 3 or more are friends of a or 3 or more are enemies of a. (Take $m = 5$ and $k = 2$ in Theorem 2.15.) Suppose first that b, c, and d are friends of a. If any 2 of these persons are friends, these 2 and a form a group of 3 mutual friends. If none of b, c, and d are friends, then b, c, and d form a group of 3 mutual enemies. The argument is similar if we suppose that b, c, and d are enemies of a. Q.E.D.

Theorem 2.18 is the simplest result in the combinatorial subject known as Ramsey theory, dating back to the original article by Ramsey [1930]. It can be restated as follows:

Theorem 2.19 Suppose that S is any set of 6 elements. If we divide the 2-element subsets of S into two classes, X and Y, then either

1. there is a 3-element subset of S all of whose 2-element subsets are in X,

or

2. there is a 3-element subset of S all of whose 2-element subsets are in Y.

Generalizing these conclusions, suppose that p and q are integers with $p,\ q\ \geq 2$. We say that a positive integer N has the (p, q) *Ramsey property* if the following holds: Given any set S of N elements, if we divide the 2-element subsets of S into two classes X and Y, then either

1. there is a p-element subset of S all of whose 2-element subsets are in X,

or

2. there is a q-element subset of S all of whose 2-element subsets are in Y.

For instance, by Theorem 2.19, the number 6 has the $(3, 3)$ Ramsey property. However, the number 3 does not have the $(3, 3)$ Ramsey property. For consider the set $S = \{a, b, c\}$ and the division of 2-element subsets of S into $X = \{\{a, b\}, \{b, c\}\}$ and $Y = \{\{a, c\}\}$. Then clearly there is no 3-element subset of S all of whose 2-element subsets are in X or 3-element subset of S all of whose 2-element subsets are in Y. Similarly, the numbers 4 and 5 do not have the $(3, 3)$ Ramsey property.

Note that if the number N has the (p, q) Ramsey property and $M > N$, the number M has the (p, q) Ramsey property. (Why?) The main theorem of Ramsey theory states that the Ramsey property is well defined.

Table 2.12: The Known Ramsey Numbers $R(p,q)^a$

p	q	$R(p,q)$	Reference(s)
2	n	n	
3	3	6	Greenwood and Gleason [1955]
3	4	9	Greenwood and Gleason [1955]
3	5	14	Greenwood and Gleason [1955]
3	6	18	Kéry [1964]
3	7	23	Kalbfleisch [1966], Graver and Yackel [1968]
3	8	28	Grinstead and Roberts [1982], McKay and Min [1992]
3	9	36	Kalbfleisch [1966], Grinstead and Roberts [1982]
4	4	18	Greenwood and Gleason [1955]
4	5	25	Kalbfleisch [1965], McKay and Radziszowski [1995]

[a] Note that $R(p,q) = R(q,p)$.

Theorem 2.20 (Ramsey's Theorem[36]) If p and q are integers with $p, q \geq 2$, there is a positive integer N which has the (p,q) Ramsey property.

For a proof, see Graham, Rothschild, and Spencer [1990].

One of the key problems in the subject known as Ramsey theory is the identification of the *Ramsey number*, $R(p,q)$, which is the smallest number that has the (p,q) Ramsey property. Note that by Theorem 2.19, $R(3,3) \leq 6$. In fact, $R(3,3) = 6$. The problem of computing a Ramsey number, $R(p,q)$, is an example of an optimization problem. So, in trying to compute Ramsey numbers, we are working on the third basic type of combinatorics problem, the optimization problem. Computation of Ramsey numbers is in general a difficult problem. Very few Ramsey numbers are known explicitly. Indeed, the only known Ramsey numbers $R(p,q)$ are given in Table 2.12. (Some of these entries are verified, at least in part, in Section 3.8.) For a comprehensive survey article on Ramsey numbers, the reader is referred to Radziszowski [2002]. See also Graham [1981], Graham, Rothschild, and Spencer [1990], and Chung and Grinstead [1983].

Ramsey's Theorem (Theorem 2.20) has various generalizations. Some are discussed in Section 3.8. For others, see Graham, Rothschild, and Spencer [1990].

Ramsey theory has some intriguing applications to topics such as confusion graphs for noisy channels, design of packet-switched networks, information retrieval, and decisionmaking. A decisionmaking application will be discussed in Section 4.3.3. An overview of some applications of Ramsey theory can be found in Roberts [1984].

[36] This theorem is essentially contained in the original paper by Ramsey [1930], which was mainly concerned with applications to formal logic. The basic results were rediscovered and popularized by Erdős and Szekeres [1935]. See Graham, Rothschild, and Spencer [1990] for an account.

EXERCISES FOR SECTION 2.19

1. How many people must be chosen to be sure that at least two have:
   (a) The same first initial?
   (b) A birthday on the same day of the year?
   (c) The same last four digits in their social security numbers?
   (d) The same first three digits in their telephone numbers?
2. Repeat Exercise 1 if we ask for at least three people to have the desired property.
3. If five different pairs of socks are put unsorted into a drawer, how many individual socks must be chosen before we can be sure of finding a pair?
4. (a) How many three-digit bit strings must we choose to be sure that two of them agree on at least one digit?
   (b) How many n-digit bit strings must we choose?
5. Final exam times are assigned to 301 courses so that two courses with a common student get different exam times and 20 exam times suffice. What can you say about the largest number of courses that can be scheduled at any one time?
6. If a rental car company has 95 cars with a total of 465 seats, can we be sure that there is a car with at least 5 seats?
7. If a school has 400 courses with an average of 40 students per course, what conclusion can you draw about the largest course?
8. If a telephone switching network of 20 switching stations averages 65,000 connections for each station, what can you say about the number of connections in the smallest station?
9. There are 3 slices of olive pizza, 5 slices of plain pizza, 7 slices of pepperoni pizza, and 8 slices of anchovy pizza remaining at a pizza party.
   (a) How many slices need to be requested to assure that 3 of at least one kind of pizza are received?
   (b) How many slices need to be requested to assure that 5 slices of anchovy are received?
10. A building inspector has 77 days to make his rounds. He wants to make at least one inspection a day, and has 132 inspections to make. Is there a period of consecutive days in which he makes exactly 21 inspections? Why?
11. A researcher wants to run at least one trial a day over a period of 50 days, but no more than 75 trials in all.
    (a) Show that during those 50 days, there is a period of consecutive days during which the researcher runs exactly 24 trials.
    (b) Is the conclusion still true for 30 trials?
12. Give an example of a committee scheduling problem where the size of the largest clique is smaller than the number of meeting times required.
13. Find the longest increasing and longest decreasing subsequences of each of the following sequences and check that your conclusions verify the Erdős-Szekeres Theorem.

(a) 6,5,7,4,1 (b) 6,5,7,4,1,10,9,11,14,3 (c) 4,12,3,7,14,13,15,16,10,8

14. Give an example of a sequence of 16 distinct integers that has neither an increasing nor a decreasing subsequence of 5 terms.
15. An employee's time clock shows that she worked 81 hours over a period of 10 days. Show that on some pair of consecutive days, the employee worked at least 17 hours.
16. A modem connection is used for 300 hours over a period of 15 days. Show that on some period of 3 consecutive days, the modem was used at least 60 hours.
17. There are 25 workers in a corporation sharing 12 cutting machines. Every hour, some group of the workers needs a cutting machine. We never expect more than 12 workers to require a machine at any given time. We assign to each machine a list of the workers cleared to use it, and make sure that each worker is on at least one machine's list. If the number of names on each of the lists is added up, the total is 95. Show that it is possible that at some hour some worker might not be able to find a machine to use.
18. Consider the following sequence:

$$9, 8, 4, 3, 2, 7, 6, 5, 10, 1.$$

Find the numbers t_i as defined in the proof of Theorem 2.16, and use these t_i's to find a decreasing subsequence of at least four terms.

19. In Example 2.36, show that there is an arrangement with 60 connections that has the properties desired.
20. Suppose that there are 10 people at a party whose (integer) ages range from 0 to 100.
    (a) Show that there are two distinct, but not necessarily disjoint, subsets of people that have exactly the same total age.
    (b) Using the two subsets from part (a), show that there exist two *disjoint* subsets that have exactly the same total age.
21. Prove Corollary 2.15.1 from Theorem 2.15.
22. Prove Theorem 2.17.
23. Show that if $n+1$ numbers are selected from the set $1, 2, 3, \ldots, 2n$, one of these will divide a second one of them.
24. Prove that in a group of at least 2 people, there are always 2 people who have the same number of acquaintances in the group.
25. An interviewer wants to assign each job applicant interviewed a rating of P (pass) or F (fail). She finds that no matter how she assigns the ratings, at least 3 people receive the same rating. What is the least number of applicants she could have interviewed?
26. Repeat Exercise 25 if she always finds 4 people who receive the same rating.
27. Given a sequence of p integers $a_1, a_2, \ldots, a_p$, show that there exist consecutive terms in the sequence whose sum is divisible by p. That is, show that there are i and j, with $1 \le i \le j \le p$, such that $a_i + a_{i+1} + \cdots + a_j$ is divisible by p.

28. Show that given a sequence of $R(n+1, n+1)$ distinct integers, either there is an increasing subsequence of $n+1$ terms or a decreasing subsequence of $n+1$ terms.

29. Show by exhibiting a division X and Y that:

    (a) The number 4 does not have the $(3,3)$ Ramsey property

    (b) The number 5 does not have the $(3,4)$ Ramsey property

    (c) The number 6 does not have the $(4,4)$ Ramsey property

30. Find the following Ramsey numbers.

    (a) $R(2,2)$ (b) $R(2,8)$ (c) $R(7,2)$

31. Show that if the number N has the (p,q) Ramsey property and $M > N$, the number M has the (p,q) Ramsey property.

32. Consider a group of 10 people, each pair of which are either friends or enemies.

    (a) Show that if some person in the group has at least 4 friends, there are 3 people who are mutual friends or 4 people who are mutual enemies.

    (b) Similarly, if some person in the group has at least 6 enemies, show that either there are 3 people who are mutual friends or 4 people who are mutual enemies.

    (c) Show that by parts (a) and (b), a group of 10 people, each pair of which are either friends or enemies, has either 3 people who are mutual friends or 4 people who are mutual enemies.

    (d) Does part (c) tell you anything about a Ramsey number?

33. Suppose that p, q, and r are integers with $p \geq r$, $q \geq r$, and $r \geq 1$. A positive integer N has the $(p,q;r)$ *Ramsey property* if the following holds: Given any set S of N elements, if we divide the r-element subsets of S into two classes X and Y, then either:

    1. There is a p-element subset of S all of whose r-element subsets are in X,

    or

    2. There is a q-element subset of S all of whose r-element subsets are in Y.

    The *Ramsey number* $R(p,q;r)$ is defined to be the smallest integer N with the $(p,q;r)$ Ramsey property.[37] For a proof that $R(p,q;r)$ is well defined, i.e., that there is always such an N, see Graham, Rothschild, and Spencer [1990].

    (a) Show that $R(p,q;1) = p+q-1$.

    (b) Show that $R(p,r;r) = p$ and $R(r,q;r) = q$.

[37] The (p,q) Ramsey property is the same as the $(p,q;2)$ Ramsey property.

ADDITIONAL EXERCISES FOR CHAPTER 2

1. There are 1000 applicants for admission to a college that plans to admit 300. How many possible ways are there for the college to choose the 300 applicants admitted?
2. An *octapeptide* is a chain of 8 amino acids, each of which is one of 20 naturally occurring amino acids. How many octapeptides are there?
3. In an RNA chain of 15 bases, there are 3 A's, 6 U's, 5 G's, and 1 C. If the chain begins with GU and ends with ACU, how many such chains are there?
4. How many functions are there each of which assigns a number 0 or a number 1 to each $m \times n$ matrix of 0's and 1's?
5. How many switching functions of 5 variables either assign 1 to all bit strings that start with a 1 or assign 0 to all bit strings that start with a 1?
6. In scheduling appliance repairs, eight homes are assigned the morning and nine the afternoon. In how many different orders can we schedule repairs?
7. If campus telephone extensions have 4 digits with no repetitions, how many different extensions are possible?
8. In an RNA chain of 20 bases, there are 4 A's, 5 U's, 6 G's, and 5 C's. If the chain begins either AC or UG, how many such chains are there?
9. How many distinguishable permutations are there of the letters in the word *reconnaissance*?
10. A chain of 20 amino acids has 5 histidines, 6 arginines, 4 glycines, 1 aspargine, 3 lysines, and 1 glutamic acid. How many such chains are there?
11. A system of 10 components works if at least 4 of the first 5 components work and at least 4 of the second 5 components work. In how many ways can the system work?
12. Of 15 paint jobs to be done in a day, 5 of them are short, 4 are long, and 6 are of intermediate length. If the 15 jobs are all distinguishable, in how many different orders can they be run so that:

    (a) All the short jobs are run at the beginning?

    (b) All the jobs of the same length are run consecutively?

13. A person wishes to visit 6 cities, each exactly twice, and never visiting the same city twice in a row. In how many ways can this be done?
14. A family with 9 children has 2 children with black hair, 3 with brown hair, 1 with red hair, and 3 with blond hair. How many different birth orders can give rise to such a family?
15. An ice cream parlor offers 29 different flavors. How many different triple cones are possible if each scoop on the cone has to be a different flavor?
16. A man has 6 different suits. In how many ways can he choose a jacket and a pair of pants that do not match?
17. Suppose that of 11 houses on a block, 6 have termites.

    (a) In how many ways can the presence or absence of termites occur so that the houses with termites are all next to each other?

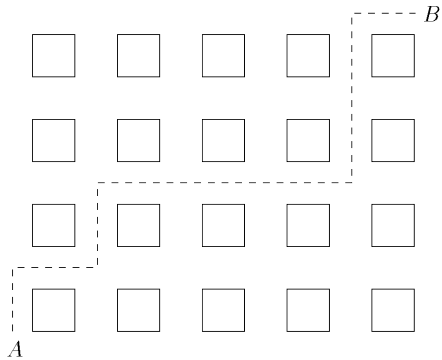


Figure 2.7: A grid of city streets. The dashed line gives a typical route.

(b) In how many ways can the presence or absence of termites occur so that none of the houses with termites are next to each other?

18. How many ways are there to distribute 8 gifts to 5 children?

19. How many ways are there to run an experiment in which each of 5 subjects is given some pills to take if there are 20 pills and each subject must take at least 1?

20. How many ways are there to prepare 20 recipes in a 7-day period if there is a list of 75 recipes to choose from, if the order in which the recipes are made does not matter, and if duplicate recipes are acceptable?

21. If a person owns 5 mutual funds holding a total of 56 stocks, what can you say about the number of stocks in the largest fund?

22. A web site was "hit" 300 times over a period of 15 days. Show that over some period of 3 consecutive days, it was "hit" at least 60 times.

23. How many steps does it take to change the permutation 84316275 into the identity permutation if:

    (a) We only allow interchanges between the ith and $(i + 1)$st element?

    (b) We only allow interchanges that replace a subsequence (like 3162) by its reverse (2613)?

24. (Pólya) Consider a city with a grid of streets as shown in Figure 2.7. In an experiment, each subject starts at corner A and is told to proceed to corner B, which is five blocks east and four blocks north of A. He or she is given a route that takes exactly nine blocks to walk. The experimenter wants to use 100 subjects and give each one a different route to walk. Is it possible to perform this experiment? Why?

25. (Liu [1968]) We print one 5-digit number on a slip of paper. We include numbers beginning with 0's, for example, 00158. Since the digits 0,1, and 8 look the same upside down, and since 6 and 9 are interchanged when a slip of paper is turned upside down, 5-digit numbers such as 61891 and 16819 can share the same slip of paper. If we want to include all possible 5-digit numbers but allow this kind of sharing, how many different slips of paper do we need?

REFERENCES FOR CHAPTER 2

Applegate, D., Bixby, R., Chvátal, V., and Cook, W., "On the Solution of Traveling Salesman Problems," *Doc. Math., Extra Vol. III*, Proceedings of the ICM (1998), 645–656.

Aumann, R. J., and Hart, S. (eds.), *Handbook of Game Theory with Economic Applications*, Elsevier Science, New York, 1998.

Bafna, V., and Pevzner, P. A., "Genome Rearrangements and Sorting by Reversal," *SIAM J. Comput., 25* (1996), 272–289.

Bafna, V., and Pevzner, P. A., "Sorting by Transpositions," *SIAM J. Discrete Math., 11* (1998), 224–240.

Ball, M. O., Colbourn, C. J., and Provan, J. S., "Network Reliability," in M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser (eds.), *Handbooks in Operations Research and Management Science, 7*, North-Holland Publishing Co., Amsterdam, 1995, 673–762.

Banzhaf, J. F., III, "Weighted Voting Doesn't Work: A Mathematical Analysis," *Rutgers Law Rev., 19* (1965), 317–343.

Barlow, R. E., and Proschan, F., *Statistical Theory of Reliability and Life Testing: Probability Models*, Holt, Rinehart and Winston, New York, 1975.

Berge, C., *Principles of Combinatorics*, Academic Press, New York, 1971.

Bland, R. G., and Shallcross, D. F., "Large Traveling Salesman Problems Arising from Experiments in X-Ray Crystallography: A Preliminary Report on Computation," *Oper. Res. Letters, 8* (1989), 125–128.

Boros, E., Gurvich, V., Hammer, P. L., Ibaraki, T., and Kogan, A., "Decomposability of Partially Defined Boolean Functions," *Discrete Appl. Math., 62* (1995), 51–75.

Boros, E., Ibaraki, T., and Makino, K., "Error-free and Best-Fit Extensions of Partially Defined Boolean Functions," *Inform. Comput., 140* (1998), 254–283.

Boros, E., Hammer, P. L., Ibaraki, T., Kogan, A., Mayoraz, E., and Muchnik, I., "An Implementation of Logical Analysis of Data," *IEEE Trans. Knowl. Data Eng., 12* (2000), 292–306.

Boyce, D. M., and Cross, M. J., "An Algorithm for the Shapley-Shubik Voting Power Index for Weighted Voting," unpublished Bell Telephone Laboratories manuscript, 1973.

Brams, S. J., *Voting Procedures*, North-Holland, Amsterdam, 1994.

Brams, S. J., and Fishburn, P. C., *Approval Voting*, Birkhauser, Boston, 1983.

Brams, S. J., Lucas, W. F., and Straffin, P. D. (eds.), *Political and Related Models*, Vol. 2 of *Modules in Applied Mathematics*, Springer-Verlag, New York, 1983.

Brams, S. J., Schotter, A., and Schwödianer, G. (eds.), *Applied Game Theory*, IHS-Studies No. 1, Physica-Verlag, Würzburg, 1979.

Buck, R. C., *Advanced Calculus*, 2nd ed., McGraw-Hill, New York, 1965.

Caprara, A., "Sorting by Reversals Is Difficult," *Proc. of RECOMB 97: The First International Conf. on Comp. Biology*, ACM Press, New York, 1997, 75–83.

Christie, D. A., "Sorting Permutations by Block-Interchanges," *Inform. Process. Lett., 60* (1996), 165–169.

Chung, F. R. K., and Grinstead, C. M., "A Survey of Bounds for Classical Ramsey Numbers," *J. Graph Theory, 7* (1983), 25–37.

Clote, P., and Backofen, R., *Computational Molecular Biology: An Introduction*, Wiley, New York, 2000.

COHEN, D. I. A., *Basic Techniques of Combinatorial Theory*, Wiley, New York, 1978.

COLEMAN, J. S., "Control of Collectivities and the Power of a Collectivity to Act," in B. Lieberman (ed.), *Social Choice*, Gordon and Breach, New York, 1971, 269–298.

CONGRESS OF THE UNITED STATES, OFFICE OF TECHNOLOGY ASSESSMENT STAFF, *Mapping Our Genes; Genome Projects - How Big, How Fast?*, Johns Hopkins University Press, Baltimore, MD, 1988.

COOK, S. A., "The Complexity of Theorem-Proving Procedures," *Proceedings of the Third ACM Symposium on Theory of Computing*, Association for Computing Machinery, New York, 1971, 151–158.

CRAMA, Y., HAMMER, P. L., and IBARAKI, T., "Cause-Effect Relationships and Partially Defined Boolean Functions," *Annals of Oper. Res., 16* (1988), 299–325.

CUMMINGS, M. R., *Human Heredity: Principles and Issues*, West Publishing Company, St. Paul, MN, 1997.

DEEGAN, J., and PACKEL, E. W., "A New Index of Power for Simple n-Person Games," *International Journal of Game Theory, 7* (1978), 113–123.

DEMBSKI, W. A., *The Design Inference: Eliminating Chance through Small Probabilities*, Cambridge University Press, New York, 1998.

DEO, N., *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall, Englewood Cliffs, NJ, 1974.

DOBZHANSKY, T., *Genetics of the Evolutionary Process*, Columbia University Press, New York, 1970.

EDMONDS, J., "Paths, Trees, and Flowers," *Canadian J. Mathematics, 17* (1965), 449–467.

EKIN, O., HAMMER, P. L., and KOGAN, A., "Convexity and Logical Analysis of Data," *Theoret. Comput. Sci., 244* (2000), 95–116.

ELSAYED, E. A., "Algorithms for Optimal Material Handling in Automatic Warehousing Systems," *Int. J. Prod. Res., 19* (1981), 525–535.

ELSAYED, E. A., and STERN, R. G., "Computerized Algorithms for Order Processing in Automated Warehousing Systems," *Int. J. Prod. Res., 21* (1983), 579–586.

ERDŐS, P., and SZEKERES, G., "A Combinatorial Problem in Geometry," *Composito Math., 2* (1935), 464–470.

FARACH-COLTON, M., ROBERTS, F. S., VINGRON, M., and WATERMAN, M. S. (eds.), *Mathematical Support for Molecular Biology*, Vol. 47, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Providence, RI, 1999.

FARQUHARSON, R., *Theory of Voting*, Yale University Press, New Haven, CT, 1969.

FEIGENBAUM, J., PAPADIMITRIOU, C. H., and SHENKER, S., "Sharing the Cost of Multicast Transmissions," to appear in *J. Comput. System Sci.* [See also *Proceedings 32nd ACM Symposium on Theory of Computing*, (2000), 218–227.]

FELLER, W., *An Introduction to Probability Theory and Its Applications*, 3rd ed., Wiley, New York, 1968.

FERRETTI, V., NADEAU, J., and SANKOFF, D., "Original Synteny," *Proc. 7th Symp. Combinatorial Pattern Matching, Springer LNCS 1075* (1996), 149–167.

FINE, T. L., *Theories of Probability*, Academic Press, New York, 1973.

FUDENBERG, D., and TIROLE, J., *Game Theory*, MIT Press, Cambridge, MA, 1991.

GAMOW, G., "Possible Mathematical Relation between Deoxyribonucleic Acid and Proteins," *K. Dan. Vidensk. Selsk. Biol. Medd., 22* (1954), 1–13. (a)

GAMOW, G., "Possible Relations between Deoxyribonucleic Acid and Protein Structures," *Nature, 173* (1954), 318. (b)

Garey, M. R., and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

Golomb, S. W., "Efficient Coding for the Deoxyribonucleic Channel," in *Mathematical Problems in the Biological Sciences*, Proceedings of Symposia in Applied Mathematics, Vol. 14, American Mathematical Society, Providence, RI, 1962, 87–100.

Graham, R. L., *Rudiments of Ramsey Theory*, CBMS Regional Conference Series in Mathematics, No. 45, American Mathematical Society, Providence, RI, 1981.

Graham, R. L., Rothschild, B. L., and Spencer, J. H., *Ramsey Theory*, 2nd ed., Wiley, New York, 1990.

Graver, J. E., and Yackel, J., "Some Graph Theoretic Results Associated with Ramsey's Theorem," *J. Comb. Theory, 4* (1968), 125–175.

Gray, F., "Pulse Code Communication," U.S. Patent 2,632,058, March 17, 1953.

Greenwood, R. E., and Gleason, A. M., "Combinatorial Relations and Chromatic Graphs," *Canadian J. Math., 7* (1955), 1–7.

Griffiths, A. J. F., Miller, J. H., Suzuki, D. T., Lewontin, R. C., and Gelbart, W. M., *An Introduction to Genetic Analysis*, 6th ed., Freeman, New York, 1996.

Grinstead, C. M., and Roberts, S. M., "On the Ramsey Numbers $R(3,8)$ and $R(3,9)$," *J. Comb. Theory, Series B, 33* (1982), 27–51.

Gusfield, D., *Algorithms on Strings, Trees and Sequences; Computer Science and Computational Biology*, Cambridge University Press, New York, 1997.

Hannenhalli, S., and Pevzner, P. A., "To Cut or Not to Cut: Applications of Comparative Physical Maps in Molecular Evolution," *Proc. 7th ACM-SIAM Symp. Discrete Algs.* (1996), 304–313.

Hardy, G. H., and Wright, E. M., *An Introduction to the Theory of Numbers*, 5th ed., Oxford University Press, New York, 1980.

Harrison, M. A., *Introduction to Switching and Automata Theory*, McGraw-Hill, New York, 1965.

Harvard Computation Laboratory Staff, *Synthesis of Electronic Computing and Control Circuits*, Harvard University Press, Cambridge, MA, 1951.

Heath, L. S., and Vergara, J. P. C., "Sorting by Bounded Block-Moves," *Discrete Appl. Math., 88* (1998), 181–206.

Herzog, S., Shenker, S., and Estrin, D., "Sharing the 'Cost' of Multicast Trees: An Axiomatic Analysis," *IEEE/ACM Trans. Networking, 5* (1997), 847–860.

Hill, F. J., and Peterson, G. R., *Switching Theory and Logical Design*, Wiley, New York, 1968.

Holley, R. W., Everett, G. A., Madison, J. T., Marquisee, M., and Zamir, A., "Structure of a Ribonucleic Acid," *Science, 147* (1965), 1462–1465.

Hopcroft, J. E., "Recent Developments in Random Algorithms," paper presented at SIAM National Meeting, Troy, NY, June 1981.

Hutchinson, G., "Evaluation of Polymer Sequence Fragment Data Using Graph Theory," *Bull. Math. Biophys., 31* (1969), 541–562.

Hwang, F. K., Monma, C., and Roberts, F. S. (eds.), *Reliability of Computer and Communication Networks*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 5, American Mathematical Society and Association for Computing Machinery, Providence, RI, 1991.

Jain, K., and Vazirani, V. V., "Applications of Approximation Algorithms to Cooperative Games," *Proceedings 33nd ACM Symposium on Theory of Computing*, (2001), 364–372.

Jerrum, M. R., "The Complexity of Finding Minimum-Length Generator Sequences," *Theoret. Comput. Sci., 36* (1985), 265–289.

Johnston, R. J., "On the Measurement of Power: Some Reactions to Laver," *Environment and Planning A, 10* (1978), 907–914.

Jones, A. J., *Game Theory: Mathematical Models of Conflict*, Wiley, New York, 1980.

Kalbfleisch, J. G., "Construction of Special Edge-Chromatic Graphs," *Canadian Math. Bull., 8* (1965), 575–584.

Kalbfleisch, J. G., "Chromatic Graphs and Ramsey's Theorem," Ph.D. thesis, University of Waterloo, Waterloo, Ontario, Canada, January 1966.

Kaplan, H., Shamir, R., and Tarjan, R. E., "Faster and Simpler Algorithms for Sorting Signed Permutations by Reversal," *Proc. 8th ACM-SIAM Symp. Discrete Algs.* (1997).

Karp, R. M., "Reducibility among Combinatorial Problems," in R. E. Miller and J. W. Thatcher (eds.), *Complexity of Computer Computations*, Plenum Press, New York, 1972, 85–103.

Karp, R. M., and Luby, M. G., "Monte-Carlo Algorithms for Enumeration and Reliability Problems," *24th Annual ACM Symposium on Theory of Computing* (1983), 56–64.

Kececioglu, J. D., and Sankoff, D., "Efficient Bounds for Oriented Chromosome Inversion Distance," *Proc. 5th Symp. Combinatorial Pattern Matching, Springer LNCS 807* (1994), 307–325.

Kececioglu, J. D., and Sankoff, D., "Exact and Approximation Algorithms for Sorting by Reversal," *Algorithmica, 13* (1995), 180–210.

Kelly, J. S., *Social Choice Theory: An Introduction*, Springer-Verlag, New York, 1987.

Kéry, G., "On a Theorem of Ramsey," *Matematikai Lapok, 15* (1964), 204–224.

Kohavi, Z., *Switching and Finite Automata Theory*, McGraw-Hill, New York, 1970.

Lander, E. S., and Waterman, M. S. (eds.), *Calculating the Secrets of Life; Contributions of the Mathematical Sciences to Molecular Biology*, National Academy Press, Washington, DC, 1995.

Lawler, E. L., *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

Lehmer, D. H., "The Machine Tools of Combinatorics," in E. F. Beckenbach (ed.), *Applied Combinatorial Mathematics*, Wiley, New York, 1964, 5–31.

Lewin, B., *Genes VII*, Oxford University Press, New York, 2000.

Liu, C. L., *Introduction to Combinatorial Mathematics*, McGraw-Hill, New York, 1968.

Liu, C. L., *Elements of Discrete Mathematics*, McGraw-Hill, New York, 1977.

Lucas, W. F., "Applications of Cooperative Games to Equitable Allocation," in W. F. Lucas (ed.), *Game Theory and Its Applications*, Proceedings of Symposia in Applied Mathematics, Vol. 24, American Mathematical Society, Providence, RI, 1981, 19–36. (a)

Lucas, W. F. (ed.), *Game Theory and Its Applications*, Proceedings of Symposia in Applied Mathematics, Vol. 24, American Mathematical Society, Providence, RI, 1981. (b)

Lucas, W. F., "Measuring Power in Weighted Voting Systems," in S. J. Brams, W. F. Lucas, and P. D. Straffin (eds.), *Political and Related Models*, Vol. 2 of *Modules in Applied Mathematics*, Springer-Verlag, New York, 1983, 183–238.

Mahadev, N. V. R., and Roberts, F. S., "Consensus List Colorings of Graphs and Physical Mapping of DNA," in M. Janowitz, F.-J. LaPointe, F. R. McMorris, B. Mirkin, and F. S. Roberts (eds.), *Bioconsensus*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 61, American Mathematical Society,

Providence, RI, 2003, 83–95.
Mann, I., and Shapley, L. S., "Values of Large Games IV: Evaluating the Electoral College by Monte Carlo Techniques," RAND Corporation Memorandum RM-2651, September 1960; reproduced in M. Shubik (ed.), *Game Theory and Related Approaches to Social Behavior*, Wiley, New York, 1964. (a)
Mann, I., and Shapley, L. S., "Values of Large Games Vl: Evaluating the Electoral College Exactly," RAND Corporation Memorandum RM-3158-PR, May 1962; reproduced in part in M. Shubik (ed.), *Game Theory and Related Approaches to Social Behavior*, Wiley, New York, 1964. (b)
McKay, B. D., and Min, Z. K., "The Value of the Ramsey Number $R(3,8)$," *J. Graph Theory, 16* (1992), 99–105.
McKay, B. D., and Radziszowski, S. P., "$R(4,5) = 25$," *J. Graph Theory, 19* (1995), 309–322.
Mosimann, J. E., *Elementary Probability for the Biological Sciences*, Appleton-Century-Crofts, New York, 1968.
Mosimann, J. E., Shapiro, M. B., Merril, C. R., Bradley, D. F., and Vinton, J. E., "Reconstruction of Protein and Nucleic Acid Sequences IV: The Algebra of Free Monoids and the Fragmentation Stratagem," *Bull. Math. Biophys., 28* (1966), 235–260.
Muroga, S., *Logic Design and Switching Theory*, Krieger Publishing Company, Melbourne, 1990.
Myerson, R. B., *Game Theory*, Harvard University Press, Cambridge, MA, 1997.
Niven, I., *An Introduction to the Theory of Numbers*, Wiley, New York, 1991.
O'Brien, S. J. (ed.), *Genetic Maps: Locus Maps of Complex Genomes*, 6th ed., Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY, 1993.
Okada, N., Hashimoto, T., and Young, P., "Cost Allocation in Water Resources Development," *Water Resources Research, 18* (1982), 361–373.
Owen, G., *Game Theory*, 3rd ed., Academic Press, San Diego, CA, 1995.
Palmer, J. D., Osorio, B., and Thompson, W. F., "Evolutionary Significance of Inversions in Legume Chloroplast DNAs," *Current Genetics, 14* (1988), 65–74.
Parzen, E., *Modern Probability Theory and Its Applications*, Wiley, New York, 1992.
Pattavina, A., *Switching Theory*, Wiley, New York, 1998.
Pevzner, P. A., *Computational Molecular Biology: An Algorithmic Approach*, MIT Press, Cambridge, MA, 2000.
Prather, R. E., *Discrete Mathematical Structures for Computer Science*, Houghton Mifflin, Boston, 1976.
Radziszowski, S. P., "Small Ramsey Numbers," *Electron. J. Combinat., DS1* (2002), 35 pp.
Ramsey, F. P., "On a Problem of Formal Logic," *Proc. Lond. Math. Soc., 30* (1930), 264–286.
Reingold, E. M., Nievergelt, J., and Deo, N., *Combinatorial Algorithms: Theory and Practice*, Prentice Hall, Englewood Cliffs, NJ, 1977.
Riordan, J., *An Introduction to Combinatorial Analysis*, Princeton University Press, Princeton, NJ, 1980.
Roberts, F. S., *Discrete Mathematical Models, with Applications to Social, Biological, and Environmental Problems*, Prentice Hall, Englewood Cliffs, NJ, 1976.
Roberts, F. S., *Measurement Theory, with Applications to Decisionmaking, Utility, and the Social Sciences*, Addison-Wesley, Reading, MA, 1979. (Digitally printed version, Cambridge University Press, Cambridge, UK, 2009.)

Roberts, F. S., "Applications of Ramsey Theory," *Discrete Appl. Math., 9* (1984), 251–261.

Ross, S. M., *A First Course in Probability*, 5th ed., Prentice Hall, Upper Saddle River, 1997.

Sessions, S. K., "Chromosomes: Molecular Cytogenetics," in D. M. Hillis and C. Moritz (eds.), *Molecular Systematics*, Sinauer, Sunderland, MA, 1990, 156–204.

Setubal, J. C., and Meidanis, J., *Introduction to Computational Molecular Biology*, PWS Publishers, Boston, 1997.

Shapley, L. S., "A Value for n-Person Games," in H. W. Kuhn and A. W. Tucker (eds.), *Contributions to the Theory of Games*, Vol. 2, Annals of Mathematics Studies No. 28, Princeton University Press, Princeton, NJ, 1953, 307–317.

Shapley, L. S., "Measurement of Power in Political Systems," in W. F. Lucas (ed.), *Game Theory and Its Applications*, Proceedings of Symposia in Applied Mathematics, Vol. 24, American Mathematical Society, Providence, RI, 1981, 69–81.

Shapley, L. S., and Shubik, M., "A Method for Evaluating the Distribution of Power in a Committee System," *Amer. Polit. Sci. Rev., 48* (1954), 787–792.

Shapley, L. S., and Shubik, M., "On the Core of an Economic System with Externalities," *Amer. Econ. Rev., 59* (1969), 678–684.

Shubik, M., "Incentives, Decentralized Control, the Assignment of Joint Costs and Internal Pricing," *Management Sci., 8* (1962), 325–343.

Stahl, S., *A Gentle Introduction to Game Theory*, American Mathematical Society, Providence, RI, 1998.

Stanat, D. F., and McAllister, D. F., *Discrete Mathematics in Computer Science*, Prentice Hall, Englewood Cliffs, NJ, 1977.

Stone, H. S., *Discrete Mathematical Structures and Their Applications*, Science Research Associates, Chicago, 1973.

Straffin, P. D., Jr., *Topics in the Theory of Voting*, Birkhäuser-Boston, Cambridge, MA, 1980.

Straffin, P. D., and Heaney, J. P., "Game Theory and the Tennessee Valley Authority," *Internat. J. Game Theory, 10* (1981), 35–43.

Tversky, A., and Gilovich, T., "The Cold Facts about the 'Hot Hands' in Basketball," *Chance, 2* (1989), 16–21.

Waterman, M. S., *Introduction to Computational Biology; Maps, Sequences and Genomes*, CRC Press, Boca Raton, FL, 1995.

Young, H. P., *Cost Allocation: Methods, Principles, Applications*, Elsevier Science, New York, 1986.

Zimmer, C., "And One for the Road," *Discover*, January 1993, 91–92.

Chapter 3

Introduction to Graph Theory

3.1 FUNDAMENTAL CONCEPTS[1]

3.1.1 Some Examples

In Example 1.4 we introduced informally the notion of a graph. In this chapter we study graphs and directed analogues of graphs, called digraphs, and their numerous applications. Graphs are a fundamental tool in solving problems of combinatorics. In turn, many of the counting techniques of combinatorics are especially useful in solving problems of graph theory. The theory of graphs is an old subject that has been undergoing a tremendous growth in interest in recent years. From the beginning, the subject has been closely tied to applications. It was invented by Euler [1736] in the process of settling the famous Königsberg bridge problem, which we discuss in Section 11.3.1. Graph theory was later applied by Kirchhoff [1847] to the study of electrical networks, by Cayley [1857, 1874] to the study of organic chemistry, by Hamilton to the study of puzzles, and by many mathematicians and nonmathematicians to the study of maps and map-coloring. In the twentieth cen-

[1]The topics in graph theory introduced in this chapter were chosen for three reasons. First, they illustrate quickly the nature and variety of applications of the subject. Second, they can be used to illustrate the counting techniques introduced in Chapter 2. Third, they will be used to illustrate the counting and existence results in Chapters 5–8. We return to graph theory more completely in Chapter 11, which begins a sequence of three chapters on graphs and networks and begins an introduction to the algorithmic aspects of graph theory. In the undergraduate combinatorics course at Rutgers taught by Fred Roberts, he does not cover much graph theory, as there is a separate undergraduate graph theory course. Accordingly, he goes through this chapter very rapidly. He covers Sections 3.1.1 and 3.1.2; all of Section 3.2 (but in about 30 minutes, with little emphasis on the exercises); 3.3.1, 3.3.3; 3.4.1, 3.4.2; 3.5.1, 3.5.2, 3.5.4, 3.5.5 (only the proof of Theorem 3.16); and 3.5.6 (without sketching the proof of Cayley's Theorem). Other sections can be added to expand on the material covered. In a graph theory course or in a combinatorics course with more emphasis on graphs or on computing or on theory, more material from this chapter should be included.

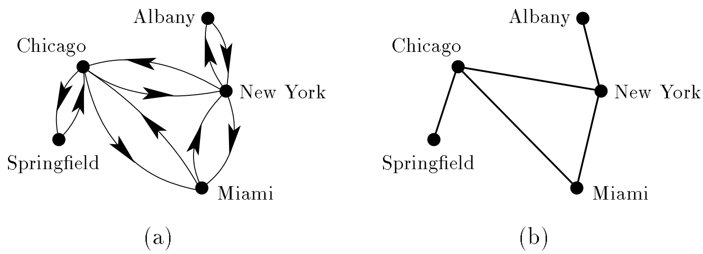


Figure 3.1: Direct air links.

tury, graph theory has been used increasingly in electrical engineering, computer science, chemistry, political science, ecology, molecular biology, transportation, information processing, and a variety of other fields.

To illustrate applications of graph theory, and to motivate the formal definitions of graph and digraph that we introduce, let us consider several examples.

Example 3.1 Transportation Networks Graphs and digraphs arise in many transportation problems. For example, consider any set of locations in a given area, between which it is desired to transport goods, people, cars, and so on. The locations could be cities, warehouses, street corners, airfields, and the like. Represent the locations as points, as shown in the example of Figure 3.1(a), and draw an arrow or directed line (or curve) from location x to location y if it is possible to move the goods, people, and so on, directly from x to y. The situation where all the links are two-way can be more simply represented by drawing a single undirected line between two locations that are directly linked rather than by drawing two arrows for each pair of locations [see Figure 3.1(b)]. Interesting questions about transportation networks are how to design them to move traffic efficiently, how to make sure that they are not vulnerable to disruption, and so on. ■

Example 3.2 Communication Networks Graphs are also used in the study of communications. Consider a committee, a corporate body, or any similar organization in which communication takes place. Let each member of the organization be represented by a point, as in Figure 3.2, and draw a line with an arrow from member x to member y if x can communicate directly with y. For example, in the police force of Figure 3.2, the captain can communicate directly with the dispatcher, who in turn can reach the captain via either of the lieutenants.[2] Typical questions asked about such a "communication network" are similar to questions about transportation networks: How can the network be designed efficiently, how easy is it to disrupt communications, and so on? The modern theory of communication networks is often concerned with networks of interacting communications and computing devices, and the graphs that arise are huge. ■

[2] See Kemeny and Snell [1962, Ch. 8] for a more detailed discussion of a similar communication network of a police force.

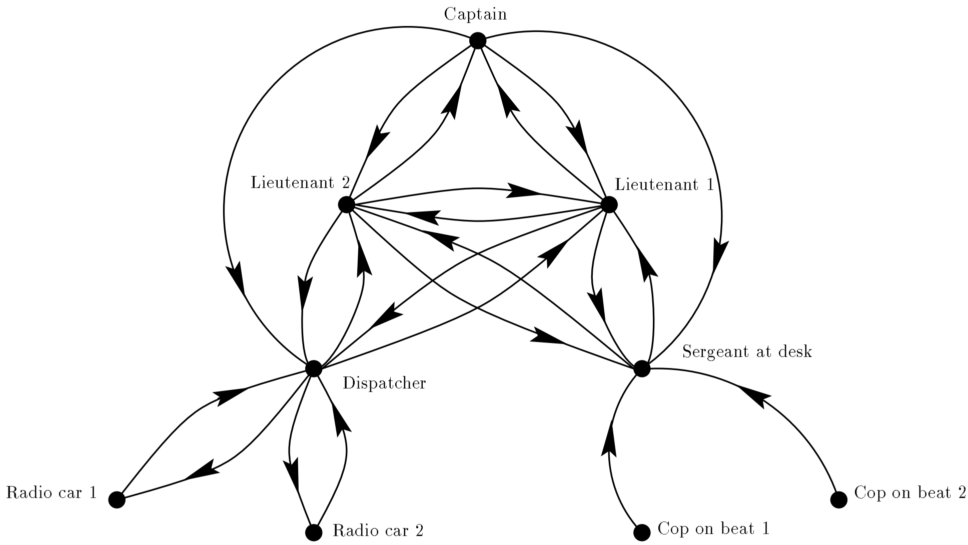


Figure 3.2: Communication network of part of a police force.

Example 3.3 Physical Networks Graphs often correspond to physical systems. For example, an electrical network can be thought of as a graph, with points as the electrical components, and two points joined by an undirected line if and only if there is a wire connecting them. Similarly, in telephone networks, we can think of the switching centers and the individual telephones as points, with two points joined by a line if there is a direct telephone line between them. Oil or gas pipelines of the kind studied in Example 1.3 can also be translated into graphs in this way. We usually seek the most efficient or least expensive design of a network meeting certain interconnection requirements, or seek a network that is least vulnerable to disruption. ■

Example 3.4 Reliability of Networks If a transportation, communication, electrical, or computer network is modeled by a graph, failures of components of the network can correspond to failures of points or lines. Good networks are designed with redundancy so that failures of some components do not lead to failure of the whole network (see Example 2.21). For instance, suppose that we have a network of six computers as shown in Figure 3.3 with direct links between them indicated by lines in the figure. Suppose there is a certain probability that each link will fail. What is the probability that every pair of computers in the system will be able to communicate (at least indirectly, possibly through other computers) if some of the links can fail? Could the network have been designed differently with the same number of links so as to increase this probability? ■

Example 3.5 Searching for Information on the Internet Because of the dramatic growth in the number of web sites and pages on the Internet, search engines

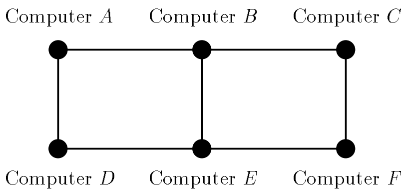
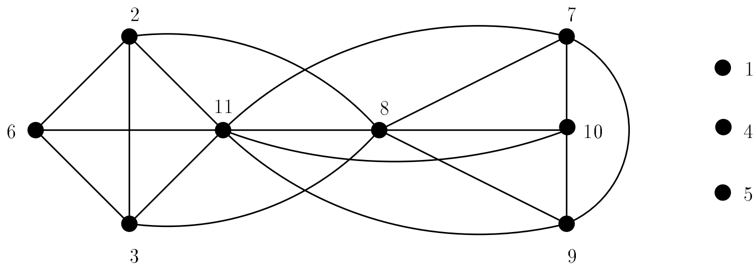


Figure 3.3: A network.

such as Google were introduced. The systems construct a database of web pages that has to be updated continually. The updating is done by robots that periodically traverse the web's hypertext structure. Let each web site be represented by a point and draw a line with an arrow in it from one web site to another if a robot can visit the second after the first. In what order should the web sites be visited so as to minimize the fraction of time that pages in the database are out of date? The answer depends in part on estimates of probabilities that given web pages are updated. Other applications of graph theory in searching on the Internet arise when we see the "most relevant" web site for a user query—this is the matching problem discussed in Chapter 12—or a set of web sites with at least a sufficiently high "relevance score." ■

Example 3.6 Analysis of Computer Programs Graphs have extensive applications in the analysis of computer programs. One approach is to subdivide a large program into subprograms, as an aid to understanding it, documenting it, or detecting structural errors. The subprograms considered are program blocks, or sequences of computer instructions with the property that whenever any instruction in the sequence is executed, all instructions in the sequence are executed. Let each program block be represented by a point, much as we represented cities in Figure 3.1. If it is possible to transfer control from the last instruction in program block x to the first instruction in program block y, draw a line with an arrow from x to y. The resulting diagram is called a *program digraph.* (A flowchart is a special case where each program block has one instruction.) Certain program blocks are designated as starting and stopping blocks. To detect errors in a program, we might use a compiler to ask if there are points (program blocks) from which it is never possible to reach a stopping point by following arrows. Or we might use the compiler to ask if there are points that can never be reached from a starting point. (If so, these correspond to subroutines which are never called.) The program digraph can also be used to estimate running time for the program. Graphs have many other uses in computer science, for instance in the design and analysis of computers and digital systems, in data structures, in the design of fault-tolerant systems, and in the fault diagnosis of digital systems. ■

Example 3.7 Competition Among Species Graphs also arise in the study of ecosystems. Consider a number of species that make up an ecosystem. Represent the species (or groups of species) as points, as in Figure 3.4, and draw an undirected line between species x and species y if and only if x and y compete. The resulting diagram is called a *competition graph* (or *niche overlap graph*). Questions one can



Key:

1. Canopy: leaves, fruit, flowers
2. Large ground animals: large mammals, birds
3. Insects
4. Trunk, fruit, flowers
5. Ground: roots, fallen fruit, leaves, trunks
6. Fungi
7. Middle-zone scansorial animals: mammals in both canopy and ground zones
8. Canopy animals: birds, fruit bats, other mammals
9. Upper air mammals: birds and bats, insectivorous
10. Middle-zone flying animals: birds, insectivorous bats
11. Small ground animals: birds, small mammals

Figure 3.4: A competition graph for species in a Malaysian rain forest. (From data of Harrison [1962], as adapted by Cohen [1978]. Graph from Roberts [1978].)

ask about competition graphs include questions about their structural properties (e.g., how "connected" are they, and what are their connected "pieces"); and about the "density" of lines (ratio of the number of lines present to the number of lines possible). ■

Example 3.8 Tournaments To give yet another application of graphs, consider a round-robin tournament[3] in tennis, where each player must play every other player exactly once, and no ties are allowed. One can represent the players as points and draw an arrow from player x to player y if x "beats" y, as in Figure 3.5. Similar tournaments arise in a surprisingly large number of places in the social, biological, and environmental sciences. Psychologists perform a pair comparison preference experiment on a set of alternatives by asking a subject, for each pair of alternatives, to state which he or she prefers. This defines a tournament, if we think of the alternatives as corresponding to the players and "prefers" as corresponding to "beats." Tournaments also arise in biology. In the farmyard, for every pair of chickens, it has been found that exactly one "dominates" the other. This "pecking order" among chickens again defines a tournament. (The same is true of other species of animals.) In studying tournaments, a basic problem is to decide on the "winner" and to rank the "players." Graph theory will help with this problem, too. ■

Example 3.9 Information Retrieval[4] In an information retrieval system on a computer, each document being indexed is labeled with a number of *index terms*

[3] This is not the more common elimination tournament.
[4] This example is due to Deo [1974].

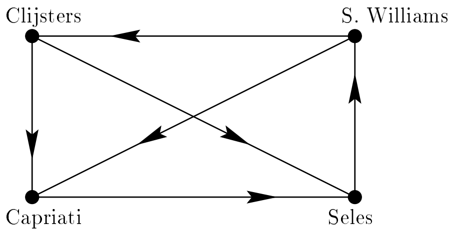


Figure 3.5: A tournament.

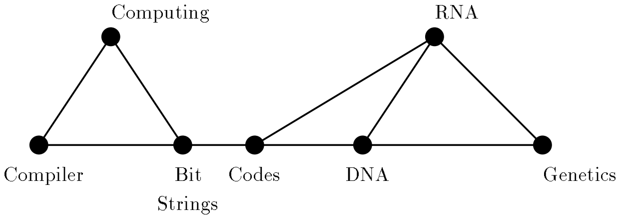


Figure 3.6: Part of a similarity graph.

or *descriptors*. Let the index terms be drawn as points and join two points with a line if the corresponding index terms are closely related, as in Figure 3.6. The diagram that results is called a *similarity graph*. It can be used to produce a classification of documents and to help in information retrieval: One provides some index terms and the information retrieval system produces a list of related terms and the corresponding documents. In sophisticated information retrieval applications in large databases such as the World Wide Web (as in Example 3.5), we might measure the relevance of a document or web page for a particular query term and seek to find a web page that has maximum relevance. ■

3.1.2 Definition of Digraph and Graph

These examples all give rise to directed or undirected graphs. To be precise, let us define a *directed graph* or *digraph* D as a pair (V, A), where V is a nonempty set and A is a set of ordered pairs of elements of V. V will be called the set of *vertices* and A the set of *arcs*. (Some authors use the terms *node, point*, and so on, in place of *vertex*, and the terms *arrow, directed line, directed edge*, or *directed link* in place of *arc*.) If more than one digraph is being considered, we will use the notation $V(D)$ and $A(D)$ for the vertex set and the arc set of D, respectively. Usually, digraphs are represented by simple diagrams such as those of Figure 3.7. Here, the vertices are represented by points and there is a directed line (or arrow, not necessarily straight) heading from u to v if and only if (u, v) is in A. For example, in the digraph D_1 of Figure 3.7, V is the set $\{u, v, w, x\}$ and A is the set

$$\{(u, v), (u, w), (v, w), (w, x), (x, u)\}.$$

If there is an arc from vertex u to vertex v, we shall say that u is *adjacent* to v.

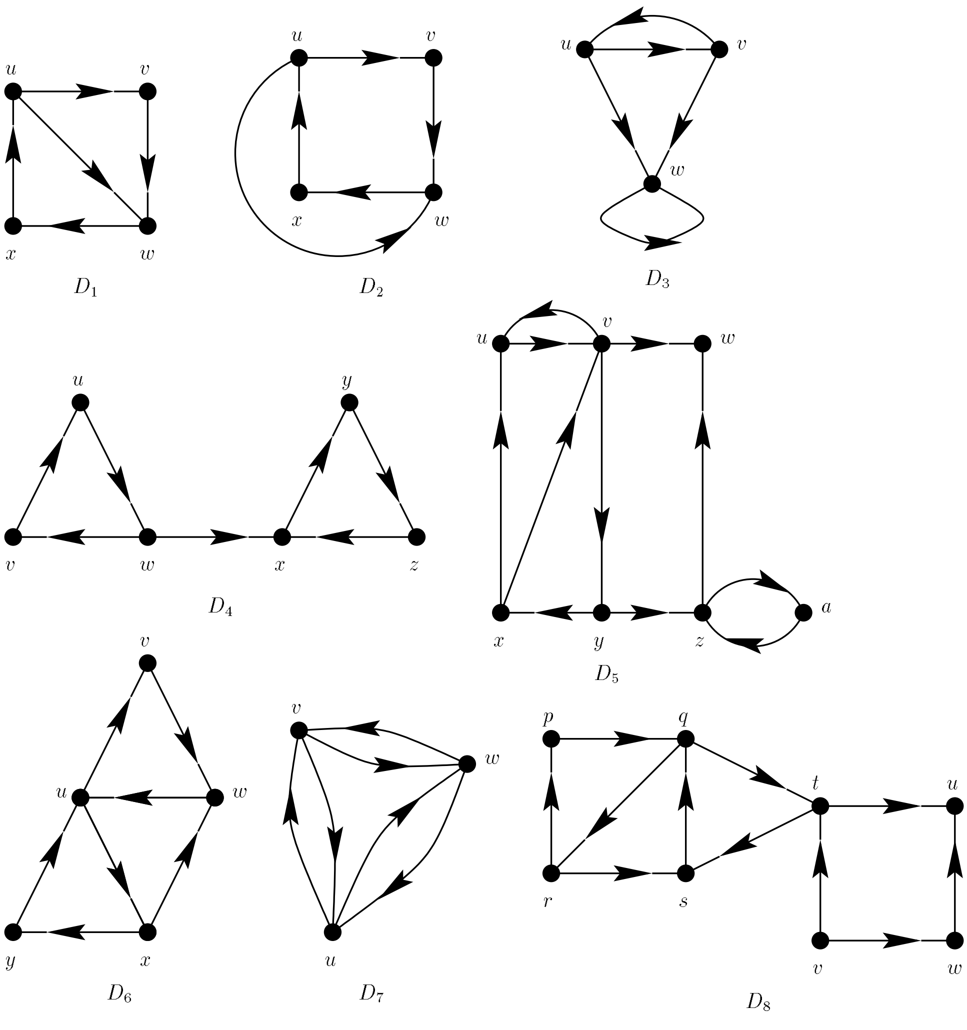


Figure 3.7: Digraphs.

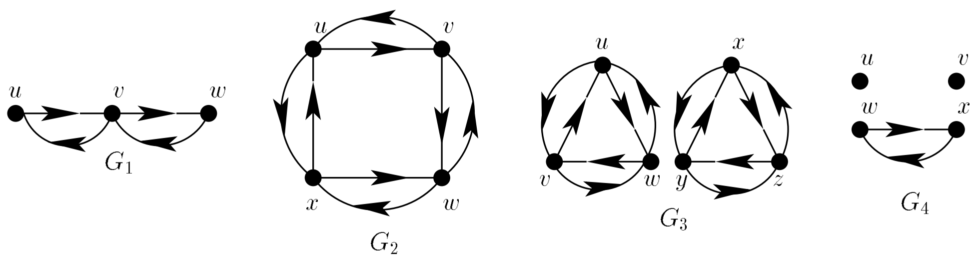


Figure 3.8: Graphs.

Thus, in Figure 3.7, in digraph D_1, u is adjacent to v and w, w is adjacent to x, and so on.

Note: The reader should notice that the particular placement of the vertices in a diagram of a digraph is unimportant. The distances between the vertices have no significance, the nature of the lines joining them is unimportant, and so on. Moreover, whether or not two arcs cross is also unimportant; the crossing point is not a vertex of the digraph. All the information in a diagram of a digraph is included in the observation of whether or not a given pair of vertices is joined by a directed line or arc and in which direction the arc goes.[5] Thus, digraphs D_1 and D_2 of Figure 3.7 are the same digraph, only drawn differently. In the next subsection, we shall say that D_1 and D_2 are isomorphic.

In a digraph, it is perfectly possible to have arcs in both directions, from u to v and from v to u, as shown in digraph D_3 of Figure 3.7, for example. It is also possible to have an arc from a vertex to itself, as is shown with vertex w in digraph D_3. Such an arc is called a *loop*. It is not possible, however, to have more than one arc from u to v. Often in the theory and applications of digraphs, such multiple arcs are useful—this is true in the study of chemical bonding, for example—and then one studies *multigraphs* or better, *multidigraphs*, rather than digraphs.

Very often, there is an arc from u to v whenever there is an arc from v to u. In this case we say that the digraph (V, A) is a *graph*. Figure 3.8 shows several graphs. In the drawing of a graph, it is convenient to disregard the arrows and to replace a pair of arcs between vertices u and v by a single nondirected line joining u and v. (In the case of a directed loop, it is replaced by an undirected one.) We shall call such a line an *edge* of the graph and think of it as an unordered pair of vertices $\{u, v\}$. (The vertices u and v do not have to be distinct.) If there is an edge $\{u, v\}$ in the graph, we call u and v *neighbors*. The graph drawings obtained from those of Figure 3.8 in this way are shown in Figure 3.9. Thus, a graph G may be defined as a pair (V, E), where V is a set of vertices and E is a set of unordered pairs of elements from V, the edges. If more than one graph is being considered, we will use the notation $V(G)$ and $E(G)$ for the vertex set and the edge set of G, respectively.

At this point, let us make explicit several assumptions about our digraphs and graphs. Many graph theorists make explicit the following assumption: There are no

[5] This isn't quite true. Some properties of digraphs may be associated with the actual placement of vertices and arcs in a diagram, as we shall see later in the text.

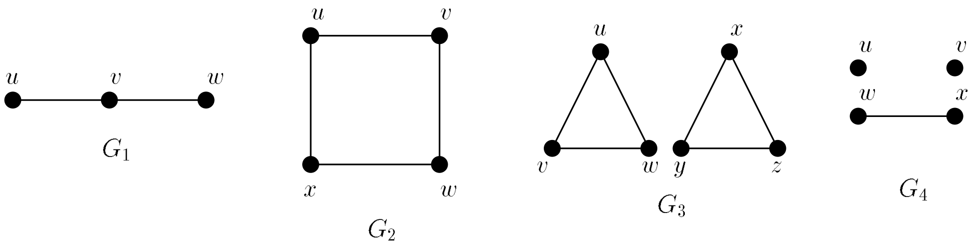


Figure 3.9: The graphs of Figure 3.8 with arcs replaced by edges.

multiple arcs or edges, that is, no more than one arc or edge from vertex u to vertex v. For us, this assumption is contained in our definition of a digraph or graph. We shall assume, at least at first, that digraphs and graphs have no loops. (Almost everything we say for loopless digraphs and graphs will be true for digraphs and graphs with loops.) We shall also limit ourselves to digraphs or graphs with finite vertex sets. Let us summarize these assumptions as follows:

Assumptions: Unless otherwise specified, all digraphs and graphs referred to in this book have finite vertex sets, have no loops, and are not allowed to have multiple arcs or edges.

Let the *degree* of vertex u of graph G, $\deg(u)$ or $\deg_G(u)$, count the number of neighbors of u. Note that if we sum up the degrees of all vertices of G, we count each edge twice, once for each vertex on it. Thus, we have the following theorem.

Theorem 3.1 If G is any graph of e edges,

$$\sum_{u \in V(G)} \deg(u) = 2e.$$

3.1.3 Labeled Digraphs and the Isomorphism Problem[6]

A *labeled digraph* or *graph* of n vertices is a digraph or graph which has the integers $1, 2, \ldots, n$ assigned, one to each vertex. Two labeled digraphs or graphs can be, for all practical purposes, the same. For instance, Figure 3.10 shows an unlabeled graph G and three labelings of the vertices of G. The first two labelings are considered the same in the following sense: Their edge sets are the same. However, the first and third labelings are different, because, for instance, the first has an edge $\{3,4\}$ whereas the third does not.

As a simple exercise in counting, let us ask how many distinct labeled graphs there are which have n vertices, for $n \geq 2$. The answer is most easily obtained if we observe that a labeled graph with n vertices can have at most $C(n,2)$ edges, for $C(n,2)$ is the number of unordered pairs of vertices from the n vertices. Let us suppose that the graph has e edges. Then we must choose e edges out of these

[6]This subsection is optional. It is placed here as a good application of the counting techniques of Chapter 2, and the concepts are used occasionally. The material can be returned to when it is needed.

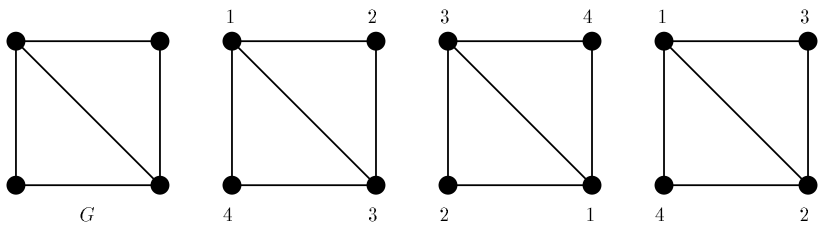


Figure 3.10: A graph G and three labelings of its vertices.

$C(n,2)$ possible edges. Hence, we see that the number $L(n,e)$ of labeled graphs with n vertices and e edges is given by

$$L(n,e) = C\left(C(n,2),e\right) = \binom{\binom{n}{2}}{e}. \tag{3.1}$$

Thus, by the sum rule and Equation (3.1), the number $L(n)$ of labeled graphs of n vertices is given by

$$L(n) = \sum_{e=0}^{C(n,2)} L(n,e) = \sum_{e=0}^{C(n,2)} C\left(C(n,2),e\right). \tag{3.2}$$

For instance, if $n = 3$, then $L(3,0) = 1, L(3,1) = 3, L(3,2) = 3, L(3,3) = 1$, and $L(3) = 8$. Figure 3.11 shows the eight labeled graphs of 3 vertices.

Note that Equation (3.2) implies that the number of distinct labeled graphs grows very fast as n grows. To see that, note that if $r = C(n,2)$, then using Theorem 2.8,

$$L(n) = \sum_{e=0}^{r} C(r,e) = 2^r,$$

so

$$L(n) = 2^{n(n-1)/2}. \tag{3.3}$$

The number given by (3.3) grows exponentially fast as n grows. There are just too many graphs to answer most graph-theoretical questions by enumerating graphs.

Two labeled digraphs are considered *the same* if their arc sets are the same. How many different labeled digraphs are there with n vertices? Since any arc is an ordered pair of vertices and loops are not allowed, there are, by the product rule, $n(n-1)$ possible arcs. The number $M(n,a)$ of labeled digraphs with n vertices and a arcs is given by

$$M(n,a) = C\left(n(n-1),a\right) = \binom{n(n-1)}{a}. \tag{3.4}$$

Again by the sum rule, the number $M(n)$ of labeled digraphs with n vertices is thus given by

$$M(n) = \sum_{a=0}^{n(n-1)} M(n,a) = \sum_{a=0}^{n(n-1)} C\left(n(n-1),a\right). \tag{3.5}$$

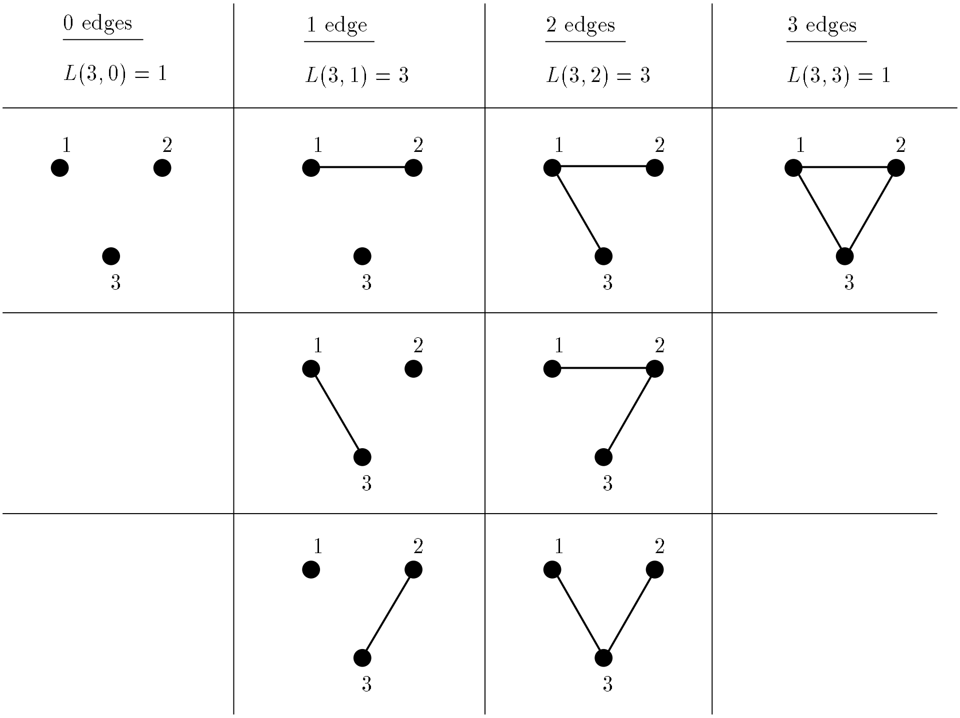


Figure 3.11: The eight different labeled graphs of 3 vertices.

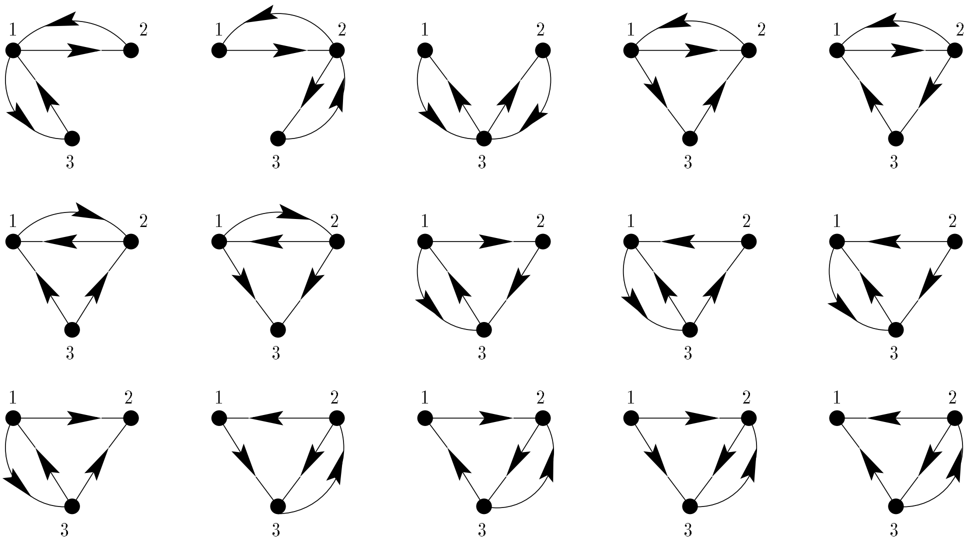


Figure 3.12: The 15 labeled digraphs of 3 vertices and 4 arcs.

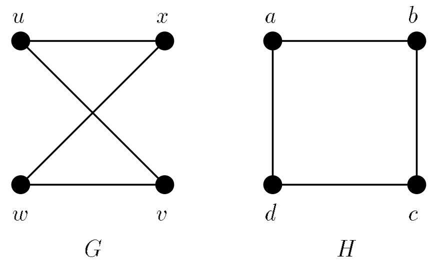


Figure 3.13: Graphs G and H are isomorphic.

For instance, if $n = 3$, then $M(3,0) = 1, M(3,1) = 6, M(3,2) = 15, M(3,3) = 20, M(3,4) = 15, M(3,5) = 6, M(3,6) = 1$, and $M(3) = 64$. Figure 3.12 shows the 15 labeled digraphs with 3 vertices and 4 arcs.

Using an argument similar to the one given above for $L(n)$, it can be shown that

$$M(n) = 2^{n(n-1)}. \tag{3.6}$$

The proof of Equation (3.6) is left to the exercises.

Two unlabeled graphs (digraphs) G and H, each having n vertices, are considered *the same* if the vertices of both can be labeled with the integers $1, 2, \ldots, n$ so that the edge sets (arc sets) consist of the same unordered (ordered) pairs, that is, if the two graphs (digraphs) can each be given a labeling that shows them to be the same as labeled graphs (digraphs). If this can be done, we say that G and H are *isomorphic*.

For instance, the graphs G and H of Figure 3.13 are isomorphic, as is shown by labeling vertex u as 1, v as 2, w as 3, x as 4, a as 1, b as 2, c as 3, and d as 4. The digraphs D_1 and D_2 of Figure 3.7 are also isomorphic; labeling u as 1, v as 2, w as 3, and x as 4 in both digraphs demonstrates this.

Although it is easy to tell whether or not two labeled graphs or digraphs are the same, the problem of determining whether or not two unlabeled graphs or digraphs are the same, that is, isomorphic, is a very difficult one indeed. This is called the *isomorphism problem*, and it is one of the most important problems in graph theory. The most naive algorithm for determining if two graphs G and H of n vertices are isomorphic would fix a labeling of G using the integers $1, 2, \ldots, n$, and then try out all possible labelings of H using these integers. Thus, this algorithm has computational complexity $f(n) = n!$, and we have already seen in Sections 2.3 and 2.4 that even for moderate n, such as $n = 25$, considering this many cases is infeasible. Although better algorithms are known, the best algorithms known for solving the isomorphism problem have computational complexity which is exponential in the size of the problem; that is, they require an unfeasibly large number of steps to compute if the number of vertices gets moderately large. See Reingold, Nievergelt, and Deo [1977, Sec. 8.5], Deo [1974, Sec. 11-7], and Kreher and Stinson [1998] for some discussion. Polynomial algorithms have been found when the graphs in question have certain properties. See, for example, Luks [1982], Bodlaender [1990], and Ponomarenko [1984, 1992].

EXERCISES FOR SECTION 3.1

1. In the digraph of Figure 3.1, identify:

   (a) The set of vertices (b) The set of arcs

2. Repeat Exercise 1 for the digraph of Figure 3.5.
3. Repeat Exercise 1 for the digraph D_4 of Figure 3.7.
4. In each of the graphs of Figure 3.9, identify:

   (a) The set of vertices (b) The set of edges

5. In digraph D_5 of Figure 3.7, find a vertex adjacent to vertex y.
6. In the graph of Figure 3.1, find all neighbors of the vertex New York.
7. Draw a transportation network with the cities New York, Paris, Vienna, Washington, DC, and Algiers as vertices, and an edge joining two cities if it is possible to travel between them by road.
8. Draw a communication network for a team fighting a forest fire.
9. Draw a digraph representing the following football tournament. The teams are Michigan, Ohio State, and Northwestern. Michigan beats Ohio State, Ohio State beats Northwestern, and Northwestern beats Michigan.
10. Draw a program digraph for a computer program of your choice.
11. Draw a similarity graph involving some terms related to ecology.
12. A *food web* is a digraph whose vertices are some species in an ecosystem and which has an arc from x to y if x preys on y. Draw a food web for the set of species {deer, mountain lion, eagle, mouse, fox, grass}.
13. Sometimes, we say that two species in an ecosystem *compete* if they have a common prey. We can build a competition graph (Example 3.7) from a food web in this way. Find the competition graph for the food web of Exercise 12.
14. Generalizing Exercise 13, we can define the competition graph G corresponding to any digraph D by letting $V(G) = V(D)$ and letting $\{x, y\} \in E(G)$ if and only if there is $a \in V(D)$ so that (x, a) and (y, a) are in $A(D)$. Find the competition graph corresponding to each of the digraphs of Figure 3.7.
15. Show that in a graph G with n vertices and e edges, there is a vertex of degree at least $2e/n$.
16. Can the number of vertices of odd degree in a graph be odd? Why?
17. Figure 3.14 shows a graph and three labelings of its vertices.

    (a) Are the first two labelings the same? Why?

    (b) Are the first and the third? Why?

18. Figure 3.15 shows a digraph and three labelings of its vertices.

    (a) Are the first two labelings the same? Why?

    (b) Are the first and the third? Why?

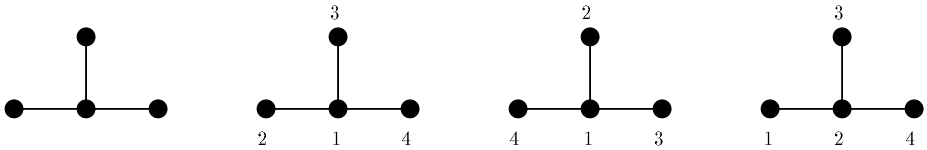


Figure 3.14: A graph and three labelings of its vertices.

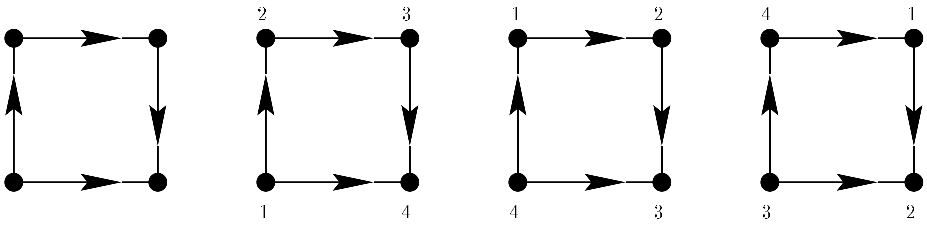


Figure 3.15: A digraph and three labelings of its vertices.

19. Find the number of labeled graphs with 4 vertices and 2 edges by using Equation (3.1). Check by drawing all such graphs.
20. How many different labeled graphs are there with 4 vertices and an even number of edges?
21. Find the number of labeled digraphs with 4 vertices and 2 arcs by using Equation (3.4). Check by drawing all such digraphs.
22. Prove Equation (3.6).
23. Are the graphs of Figure 3.16(a) isomorphic? Why?
24. Are the graphs of Figure 3.16(b) isomorphic? Why?
25. Are the digraphs of Figure 3.17(a) isomorphic? Why?
26. Are the digraphs of Figure 3.17(b) isomorphic? Why?
27. Are the digraphs of Figure 3.17(c) isomorphic? Why?
28. An *orientation* of a graph arises by replacing each edge $\{x, y\}$ by one of the arcs (x, y) or (y, x). For instance, the digraph of Figure 3.15 is an orientation of graph H of Figure 3.13. For each of the graphs of Figure 3.16(a), find all nonisomorphic orientations.

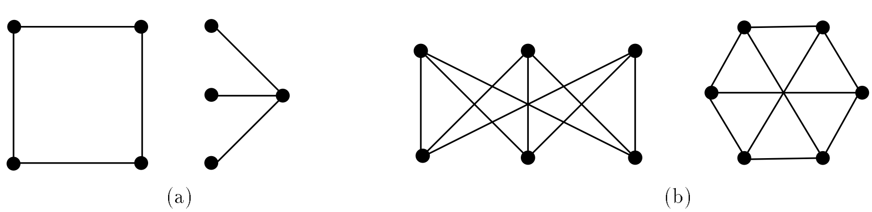


Figure 3.16: Graphs for Exercises 23, 24, 28, Section 3.1.

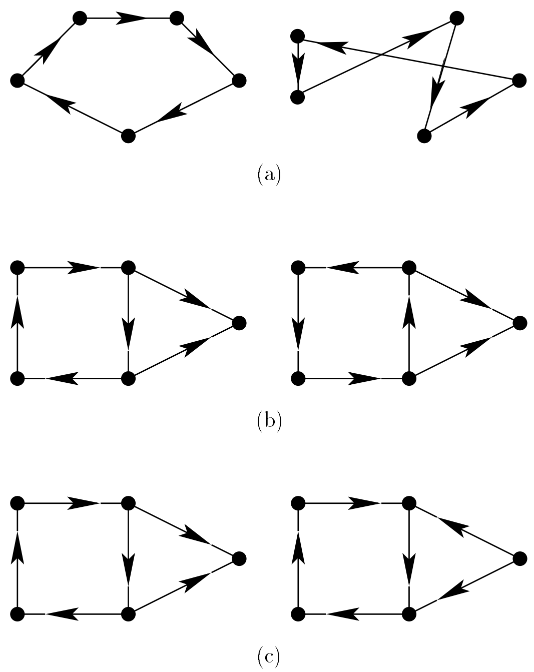


Figure 3.17: Digraphs for Exercises 25, 26, 27, Section 3.1.

29. Suppose that G and H are two graphs with the same number of vertices and the same number of edges. Suppose that α_k is the number of vertices in G with exactly k neighbors, and β_k is the number of vertices in H with exactly k neighbors. Suppose that $\alpha_k = \beta_k$ for all k. Are G and H necessarily isomorphic? Why?

30. Repeat Exercise 29 if $\alpha_2 = \beta_2 = |V(G)| = |V(H)|$ and $\alpha_k = \beta_k = 0$ for $k \neq 2$.

3.2 CONNECTEDNESS

3.2.1 Reaching in Digraphs

In a communication network, a natural question to ask is: Can one person initiate a message to another person? In a transportation network, an analogous question is: Can a car move from location u to location v? In a program digraph, we are interested in determining if from every vertex it is possible to follow arcs and ultimately hit a stopping vertex. All of these questions have in common the following idea of reachability in a digraph $D = (V, A)$: Can we reach vertex v by starting at vertex u and following the arcs of D?

To make this concept precise, let us introduce some definitions. A *path* in D is a sequence

$$u_1, a_1, u_2, a_2, \ldots, u_t, a_t, u_{t+1}, \tag{3.7}$$

Table 3.1: Reaching and Joining

Digraph D	Graph G
$u_1, a_1, u_2, a_2, \ldots, u_t, a_t, u_{t+1}$	$u_1, e_1, u_2, e_2, \ldots, u_t, e_t, u_{t+1}$
Reaching	Joining
Path:	*Chain*:
a_i is (u_i, u_{i+1})	e_i is $\{u_i, u_{i+1}\}$
Simple path:	*Simple chain*:
Path and	Chain and
u_i distinct	u_i distinct
Closed path:	*Closed chain*:
Path and	Chain and
$u_{t+1} = u_1$	$u_{t+1} = u_1$
Cycle (simple closed path):	*Circuit (simple closed chain)*:
Path and	Chain and
$u_{t+1} = u_1$ and	$u_{t+1} = u_1$ and
u_i distinct, $i \leq t$ and	u_i distinct, $i \leq t$ and
(a_i distinct)[a]	e_i distinct

[a]This follows from u_i distinct, $i \leq t$.

where $t \geq 0$, each u_i is in V, that is, is a vertex, and each a_i is in A, that is, is an arc, and a_i is the arc (u_i, u_{i+1}). That is, arc a_i goes from u_i to u_{i+1}. Since t might be 0, u_1 alone is a path, a path from u_1 to u_1. The path (3.7) is called a *simple path* if we never use the same vertex more than once.[7] For example, in digraph D_5 of Figure 3.7, $u, (u, v), v, (v, w), w$ is a simple path and $u, (u, v), v, (v, y), y, (y, x), x, (x, v), v, (v, w), w$ is a path that is not a simple path since it uses vertex v twice. Naming the arcs is superfluous when referring to a path, so we simply speak of (3.7) as the path $u_1, u_2, \ldots, u_t, u_{t+1}$.

A path (3.7) is called *closed* if $u_{t+1} = u_1$. In a closed path, we end at the starting point. If the path (3.7) is closed and the vertices $u_1, u_2, \ldots, u_t$ are distinct, then (3.7) is called a *cycle* (a simple closed path[8]). (The reader should note that if the vertices $u_i, i \leq t$, are distinct, the arcs a_i must also be distinct.)

To give some examples, the path u, v, w, x, u in digraph D_1 of Figure 3.7 is a cycle, as is the path u, v, w, x, y, u in digraph D_6. But the closed path u, v, y, x, v, y, x, u of D_5 is not a cycle, since there are repeated vertices. In general, in counting or listing cycles of a digraph, we shall not distinguish two cycles that use the same vertices and arcs in the same order but start at a different vertex. Thus, in digraph D_6 of Figure 3.7, the cycle w, x, y, u, v, w is considered the same as the cycle

[7]One of the difficulties in learning graph theory is the large number of terms that have to be mastered early. To help the reader overcome this difficulty, we have included the terms *path, simple path*, and so on, in succinct form in Table 3.1

[8]A simple closed path is, strictly speaking, not a simple path.

u, v, w, x, y, u. The *length* of a path, simple path, cycle, and so on is the number of arcs in it. Thus, the path (3.7) has length t. In digraph D_5 of Figure 3.7, u, v, y, x, v is a path of length 4, u, v, y, z is a simple path of length 3, and u, v, y, x, u is a cycle of length 4. We say that v is *reachable* from u if there is a path from u to v. Thus, in D_5, z is reachable from u. However, u is not reachable from z.

A digraph $D = (V, A)$ is called *complete symmetric* if for every $u \neq v$ in V, the ordered pair (u, v) is an arc of D. For instance, the digraph D_7 of Figure 3.7 is complete symmetric. A complete symmetric digraph on n vertices has $n(n-1)$ arcs. Let us ask how many simple paths it has of a given length k, if $k \leq n$, the number of vertices. The answer is that to find such a path, we choose any $k+1$ vertices, and then order them. Thus, we have $C(n, k+1) \cdot (k+1)! = P(n, k+1)$ such paths.

3.2.2 Joining in Graphs

Suppose that $G = (V, E)$ is a graph. Terminology analogous to that for digraphs can be introduced. A *chain* in G is a sequence

$$u_1, e_1, u_2, e_2, \ldots, u_t, e_t, u_{t+1}, \tag{3.8}$$

where $t \geq 0$, each u_i is a vertex, and each e_i is the edge $\{u_i, u_{i+1}\}$. A chain is called *simple* if all the u_i are distinct and *closed* if $u_{t+1} = u_1$. A closed chain (3.8) in which $u_1, u_2, \ldots, u_t$ are distinct and $e_1, e_2, \ldots, e_t$ are distinct is called a *circuit* (a simple closed chain[9]). The *length* of a chain, circuit, and so on of form (3.8) is the number of edges in it. We say that u and v are *joined* if there is a chain from u to v.

To give some examples, in the graph of Figure 3.18,

$$r, \{r,t\}, t, \{t,w\}, w, \{w,u\}, u, \{u,t\}, t, \{t,r\}, r, \{r,s\}, s, \{s,u\}, u, \{u,t\}, t, \{t,w\}, w$$

is a chain. This chain can be written without reference to the edges as r, t, w, u, t, r, s, u, t, w. A simple chain is given by r, t, u, w, x. A circuit is given by r, t, u, s, r. Finally, $p, \{p, r\}, r, \{r, p\}, p$ is not considered a circuit, since $e_1 = e_t$. (The edges are unordered, so $\{p, r\} = \{r, p\}$.) Note that without the restriction that the edges be distinct, this would be a circuit. In the analogous case of digraphs, we did not have to assume arcs distinct for a cycle, since that follows from vertices distinct.

The *complete graph* on n vertices, denoted K_n, is defined to be the graph in which every pair of vertices is joined by an edge. In K_n, there are $C(n, 2)$ edges. The number of simple chains of length k in K_n is given by $P(n, k+1)$. (Why?)

3.2.3 Strongly Connected Digraphs and Connected Graphs

One reason graph theory is so useful is that its geometric point of view allows us to define various structural concepts. One of these concepts is connectedness. A

[9] We shall see below why the restriction that the edges be distinct is added. Also, it should be noted that a simple closed chain is, strictly speaking, not simple.

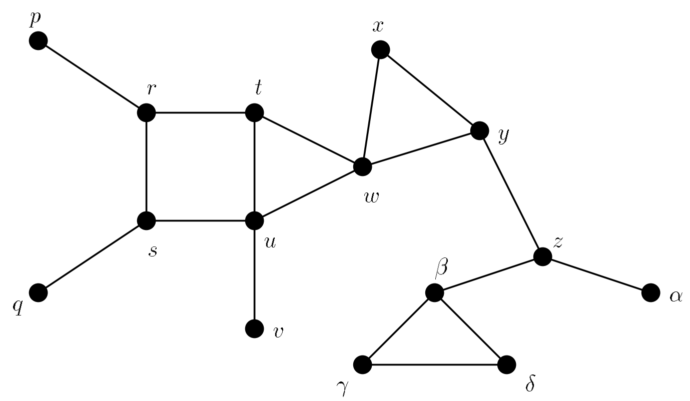


Figure 3.18: A graph.

digraph is said to be *strongly connected* if for every pair of vertices u and v, v is reachable from u and u is reachable from v. Thus, digraph D_6 of Figure 3.7 is strongly connected, but digraph D_5 is not. If a communication network is strongly connected, every person can *initiate* a communication to every other person. If a transportation network is not strongly connected, there are two locations u and v so that one cannot go from the first to the second, or vice versa. In Section 11.2 we study how to obtain strongly connected transportation networks. A program digraph is never strongly connected, for there are no arcs leading out of a stopping vertex. A tournament can be strongly connected (see Figure 3.5); however, it does not have to be. In a strongly connected tournament, it is hard to rank the players, since we get situations where u_1 beats u_2, who beats $u_3, \ldots$, who beats u_t, who beats u_1.

We say that a graph is *connected* if between every pair of vertices u and v there is a chain. This notion of connectedness coincides with the one used in topology: The graph has one "piece." In Figure 3.9, graphs G_1 and G_2 are connected while G_3 and G_4 are not. Physical networks (electrical, telephone, pipeline) are usually connected. Indeed, we try to build them so that an outage at one edge does not result in a disconnected graph.

Algorithms to test whether or not a graph is connected have been designed in a variety of ways. The fastest are very good. They have computational complexity that is linear in the number of vertices n plus the number of edges e. In the language of Section 2.18, they take on the order of $n + e$ steps. [They are $O(n + e)$.] Since a graph has at most $\binom{n}{2}$ edges, we have

$$e \leq \binom{n}{2} = \frac{n(n-1)}{2} \leq n^2.$$

Thus, $n + e \leq n + n^2$, so these algorithms take a number of steps on the order of $n + n^2$, which is a polynomial in n. [In the notation of Section 2.18, they are $O(n^2)$.

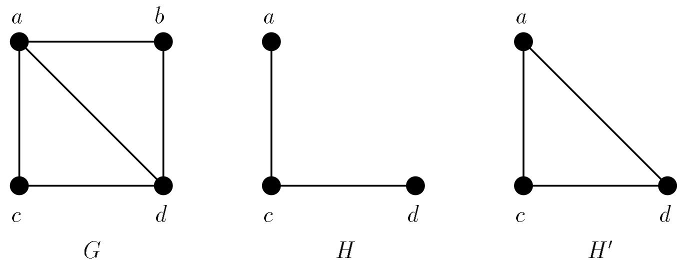


Figure 3.19: H is a subgraph of G and H' is a generated subgraph.

We say they are quadratic in n.] Similar results and algorithms apply to digraphs and the concept of strong connectedness. We discuss some of these algorithms in Section 11.1. See also Aho, Hopcroft, and Ullman [1974], Even [1979], Frank [1995], Gibbons [1985], Golumbic [1980], Reingold, Nievergelt, and Deo [1977], or West [2001].

3.2.4 Subgraphs

In what follows it will sometimes be useful to look at parts of a graph. Formally, suppose that $G = (V, E)$ is a graph. A *subgraph* $H = (W, F)$ is a graph such that W is a subset of V and F is a set of unordered pairs of vertices of W which is a subset of E. Thus, to define a subgraph of G, we choose from G some vertices and some edges joining the chosen vertices. For instance, in Figure 3.19, graphs H and H' are both subgraphs of graph G. In H', the edge set consists of all edges of G joining vertices of $W = \{a, c, d\}$. In such a case, we say that H' is the *subgraph generated* or *induced* by the vertices of W.

Similar concepts apply to digraphs. If $D = (V, A)$ is a digraph, then a *subgraph* (or *subdigraph*) $J = (W, B)$ of D is a digraph with W a subset of V and B a set of ordered pairs of vertices of W which is a subset of A. J is a *generated subgraph* if B is all arcs of D that join vertices in W. For instance, in Figure 3.20, digraph J is a subgraph of digraph D and digraph J' is the subgraph generated by the vertices a, c, and d.

As a simple application of these ideas, let us ask how many subgraphs of k vertices there are if we start with the complete symmetric digraph on n vertices. To find such a subgraph, we first choose the k vertices; this can be done in $C(n, k)$ ways. These vertices are joined by $k(k-1)$ arcs in D. We may choose any subset of this set of arcs for the subgraph; that is, we may choose arcs for the subgraph in $2^{k(k-1)}$ ways. Thus, by the product rule, there are

$$C(n, k) \cdot 2^{k(k-1)}$$

subgraphs of k vertices.

Example 3.10 Reliability of Systems (Example 2.21 Revisited) In Example 2.21 we studied systems consisting of components that might or might not work,

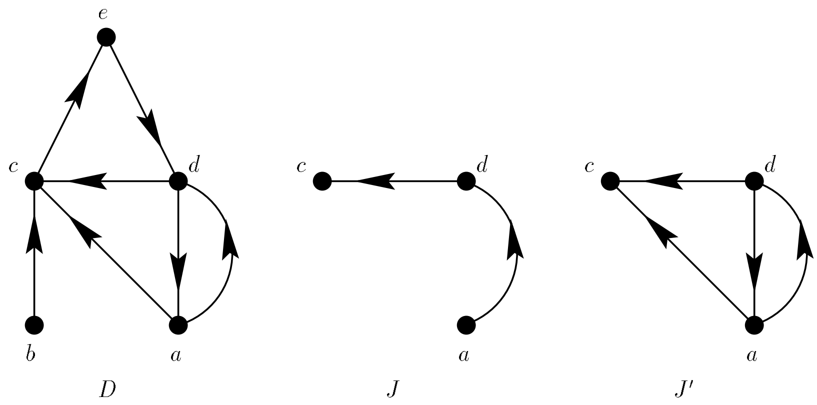


Figure 3.20: J is a subgraph of D and J' is the subgraph generated by vertices a, c, and d.

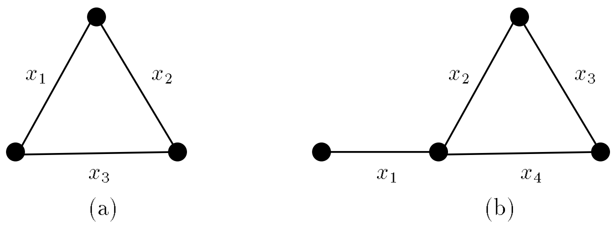


Figure 3.21: Two systems.

and introduced rules for determining, given which components are working, whether or not the system works. In studying reliability of such systems, we commonly represent a system by a graph G and let each edge correspond to a component. Then in many situations (see Example 3.4) it makes sense to say that the system works if and only if every pair of vertices is joined by a chain of working components, i.e., if and only if the subgraph H consisting of all vertices of G and the working edges of G is connected. Consider, for example, the graph G of Figure 3.21(a). There are three components, labeled x_1, x_2, and x_3. Clearly, the system works if and only if at least two of these components work. Similarly, if G is as in Figure 3.21(b), the system works if and only if component x_1 works and at least two of the remaining three components work. ■

3.2.5 Connected Components

Suppose that $G = (V, E)$ is a graph. A *connected component* or a *component* of G is a connected, generated subgraph H of G which is maximal in the sense that no larger connected generated subgraph K of G contains all the vertices of H. For example, in the graph G of Figure 3.22, the subgraph generated by the vertices a, b,

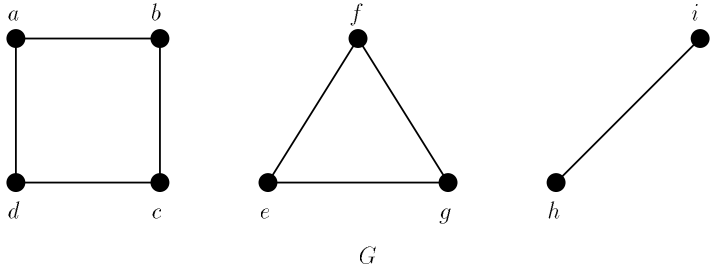


Figure 3.22: There are three components, the subgraphs generated by vertices a, b, c, and d, by vertices e, f, and g, and by vertices h and i.

and c is connected, but it is not a component since the subgraph generated by the vertices a, b, c, and d is a connected generated subgraph containing all the vertices of the first subgraph. This second subgraph is a component. There are three components in all, the other two being the subgraphs generated by vertices e, f, and g and by h and i. These components correspond to the "pieces" of the graph. Clearly, connected graphs have exactly one component. In the information retrieval situation of Example 3.9, to give one simple application, components produce a natural classification of documents. In the competition graph of Figure 3.4, there are four components (three consisting of one vertex each). Real-world competition graphs tend to have at least two components. Concepts for digraphs analogous to connected components are studied in the exercises.

EXERCISES FOR SECTION 3.2

1. For the digraph D_8 of Figure 3.7:
   (a) Find a path that is not a simple path.
   (b) Find a closed path.
   (c) Find a simple path of length 4.
   (d) Determine if $q, (q, t), t, (t, s), s, (s, q), q$ is a cycle.
   (e) Find a cycle of length 3 containing vertex p.
2. For the graph of Figure 3.18:
   (a) Find a closed chain that is not a circuit.
   (b) Find the longest circuit.
   (c) Find a chain different from the one in the text which is not simple.
   (d) Find a closed chain of length 6.
3. Give an example of a digraph and a path in that digraph which is not a simple path but has no repeated arcs.
4. Give an example of a graph in which the shortest circuit has length 5 and the longest circuit has length 8.

5. For each digraph of Figure 3.7, determine if it is strongly connected.
6. Which of the graphs of Figure 3.23 are connected?
7. For each digraph of Figure 3.24:

   (a) Find a subgraph that is not a generated subgraph.

   (b) Find the subgraph generated by vertices 5, 8, and 9.

   (c) Find a strongly connected generated subgraph.

8. For the graph of Figure 3.25:

   (a) Find a subgraph that is not a generated subgraph.

   (b) Find a generated subgraph that is connected but not a connected component.

   (c) Find all connected components.

9. A digraph is *unilaterally connected* if for every pair of vertices u and v, either v is reachable from u or u is reachable from v, but not necessarily both.

   (a) Give an example of a digraph that is unilaterally connected but not strongly connected.

   (b) For each digraph of Figure 3.7, determine if it is unilaterally connected.

10. A digraph is *weakly connected* if when all directions on arcs are disregarded, the resulting graph (or possibly multigraph) is connected.

    (a) Give an example of a digraph that is weakly connected but not unilaterally connected.

    (b) Give an example of a digraph that is not weakly connected.

    (c) For each digraph of Figure 3.7, determine if it is weakly connected.

11. Prove that if v is reachable from u in digraph D, there is a simple path from u to v in D.
12. Suppose that a system defined by a graph G works if and only if the vertices of G and the working edges form a connected subgraph of G. Under what circumstances does each of the systems given in Figure 3.26 work?
13. In a digraph D, a *strong component* is a strongly connected, generated subgraph which is maximal in the sense that it is not contained in any larger, strongly connected, generated subgraph. For example, in digraph D_5 of Figure 3.7, the subgraph generated by vertices x, y, v is strongly connected, but not a strong component since the subgraph generated by x, y, v, u is also strongly connected. The latter is a strong component. So is the subgraph generated by the single vertex w and the subgraph generated by the vertices z and a. There are no other strong components. (For applications of strong components to communication networks, to energy demand, and to Markov models of probabilistic phenomena, see Roberts [1976].) Find all strong components of each digraph of Figure 3.7.
14. Find all strong components for the police force of Figure 3.2.

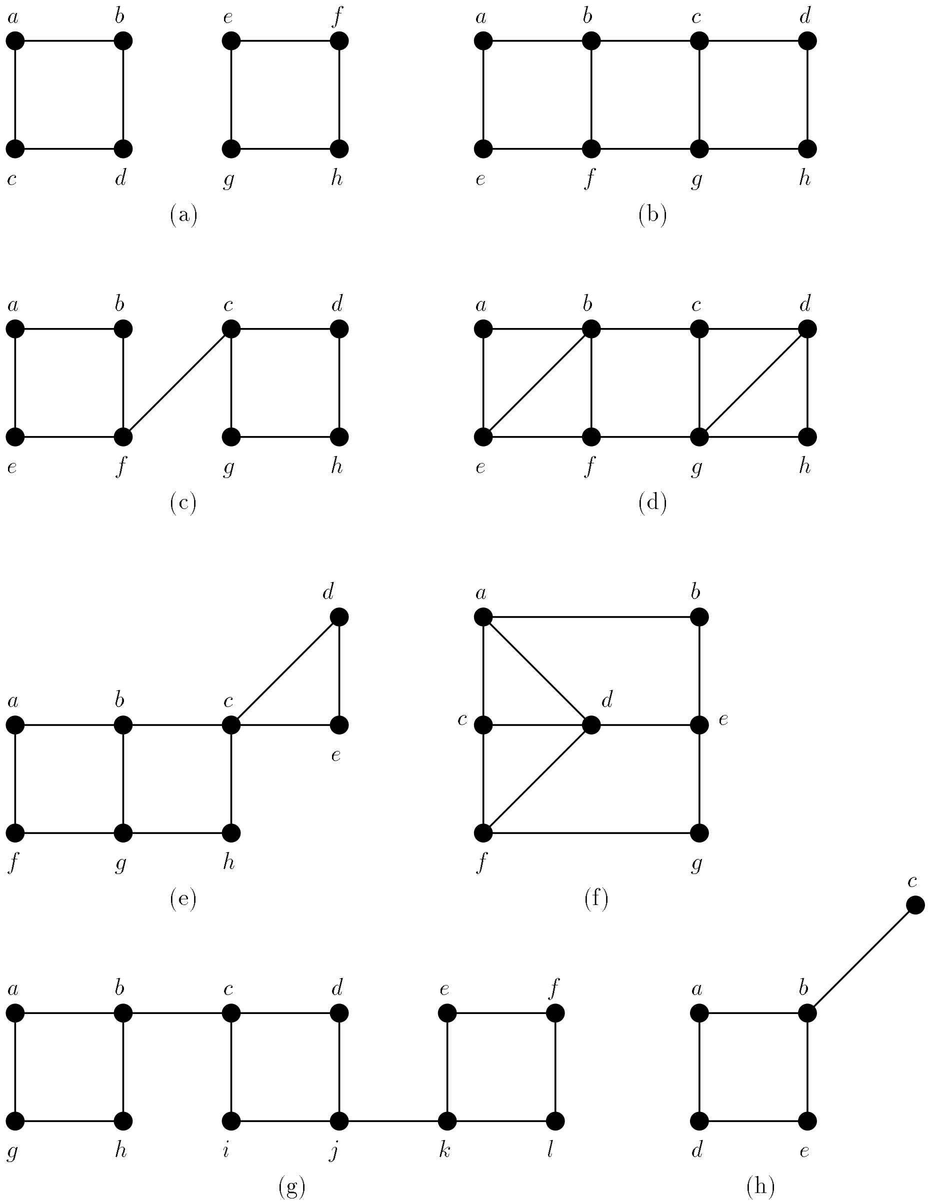


Figure 3.23: Graphs for exercises of Section 3.2.

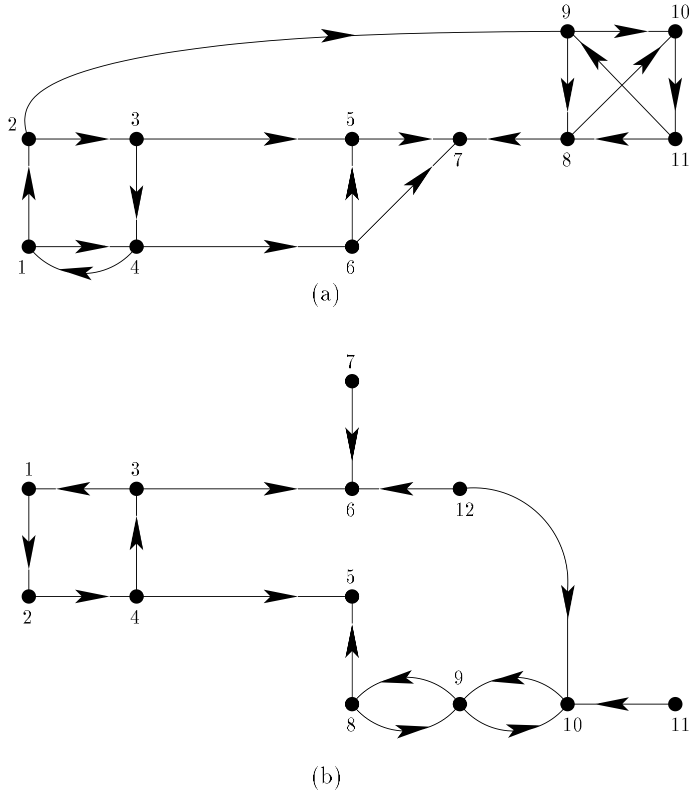


Figure 3.24: Digraphs for exercises of Section 3.2.

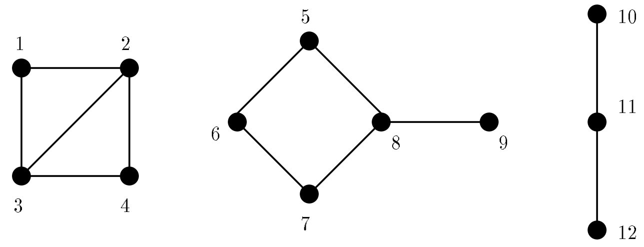


Figure 3.25: Graph for exercises of Section 3.2.

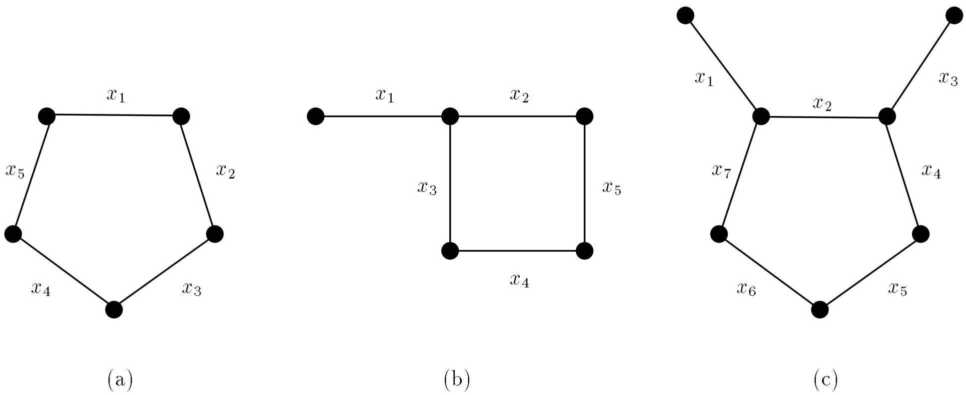


Figure 3.26: Systems for Exercise 12, Section 3.2.

15. In a digraph D, show that:

    (a) Every vertex is in some strong component.

    (b) Every vertex is in at most one strong component.

16. Show that a graph is connected if and only if it has a chain going through all the vertices.

17. Prove that a digraph is strongly connected if and only if it has a closed path going through all the vertices.

18. Prove that in a unilaterally connected digraph D, in any set of vertices, there is a vertex that can reach (using arcs of D) all others in the set.

19. In a communication network, suppose that an arc from x to y means that a message can be sent directly from x to y. If we want to place a message with as small a set of vertices as possible, so that it is possible for the message to reach all other vertices (perhaps in more than one step), what is the smallest number of vertices needed if the communication network is:

    (a) Strongly connected?

    (b) Unilaterally connected?

20. In Exercise 19, if the communication network is weakly connected, can we always place a message with at most half of the vertices to guarantee that it can reach all other vertices?

21. Show from the result of Exercise 18 that a digraph is unilaterally connected if and only if it has a path going through all the vertices.

22. (a) Give an example of a strongly connected digraph that has no cycle through all the vertices.

    (b) Does every unilaterally connected digraph have a simple path through all the vertices?

23. A *weak component* of a digraph is a maximal, weakly connected, generated subgraph. For each digraph of Figure 3.24, find all weak components.

24. A *unilateral component* of a digraph is a maximal, unilaterally connected, generated subgraph.

    (a) Find a unilateral component with five vertices in digraph (b) of Figure 3.24.

    (b) Is every vertex of a digraph in at least one unilateral component?

    (c) Can it be in more than one?

25. A digraph is *unipathic* if whenever v is reachable from u, there is exactly one simple path from u to v.

    (a) Is the digraph D_4 of Figure 3.7 unipathic?

    (b) What about the digraph of Figure 3.15?

26. For a digraph that is strongly connected and has n vertices, what is the least number of arcs? What is the most? (Observe that a digraph which is strongly connected with the least number of arcs is very vulnerable to disruption. How many links is it necessary to sever in order to disrupt communications?)

27. (Harary, Norman, and Cartwright [1965]) Refer to the definition of unipathic in Exercise 25. Can two cycles of a unipathic digraph have a common arc? (Give a proof or counterexample.)

28. (Harary, Norman, and Cartwright [1965]) If D is strongly connected and has at least two vertices, does every vertex have to be on a cycle? (Give a proof or counterexample.)

29. Suppose that a digraph D is not weakly connected.

    (a) If D has four vertices, what is the maximum number of arcs?

    (b) What if D has n vertices?

30. Do Exercise 29 for digraphs that are unilaterally connected but not strongly connected.

31. Do Exercise 29 for digraphs that are weakly connected but not unilaterally connected.

32. The reliability of a network modeled as a digraph D can be measured by how much its connectedness changes when a single arc or vertex fails. Let $D-u$ be the subgraph generated by vertices different from u. Give examples of digraphs D and vertices u with the following properties, or show that there are no such digraphs:

    (a) D is strongly connected and $D-u$ is unilaterally but not strongly connected.

    (b) D is strongly connected and $D-u$ is not unilaterally connected.

    (c) D is unilaterally but not strongly connected and $D-u$ is not unilaterally connected.

33. Repeat Exercise 32 for $D-a$, where a is an arc of D and $D-a$ is the subgraph of D obtained by removing arc a.

34. (Harary, Norman, and Cartwright [1965]) If D is a digraph, define the *complementary digraph* D^c as follows: $V(D^c) = V(D) = V$ and an ordered pair (u,v) from $V \times V$ (with $u \neq v$) is in $A(D^c)$ if and only if it is not in $A(D)$. For example, if D is the digraph of Figure 3.27, then D^c is the digraph shown. Give examples of digraphs D that are weakly connected, not unilaterally connected, and such that:

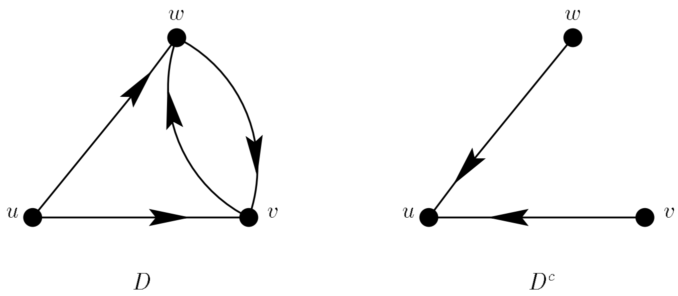


Figure 3.27: A digraph D and its complementary digraph D^c.

(a) D^c is strongly connected.

(b) D^c is unilaterally connected but not strongly connected.

(c) D^c is weakly connected but not unilaterally connected.

35. Find the number of distinct cycles of length k in the complete symmetric digraph of n vertices if two cycles are considered the same if one can be obtained from the other by changing the starting vertex.

3.3 GRAPH COLORING AND ITS APPLICATIONS

3.3.1 Some Applications

In Example 1.4 we considered the problem of scheduling meetings of committees in a state legislature and translated that into a problem concerning graphs. In this section we formulate the graph problem as a problem of coloring a graph. We remark on a number of applications of graph coloring. In the next section we apply the counting tools of Chapter 2 to count the number of graph colorings.

Example 3.11 Scheduling Meetings of Legislative Committees (Examples 1.4, 2.33, 2.35 Revisited) In the committee scheduling problem, we draw a graph G where the vertices of G are all the committees that need to be assigned regular meeting times and two committees are joined by an edge if and only if they have a member in common. Now we would like to assign a meeting time to each committee in such a way that if two committees have a common member, that is, if the corresponding vertices are joined by an edge in G, then the committees get different meeting times. Instead of assigning meeting times, let us think of assigning a color (corresponding to a meeting time) to each vertex of G, in such a way that if two vertices are joined by an edge, they get a different color. Committee scheduling is a prime example of *graph coloring* (*vertex coloring* or *coloring* are also used): Coloring the vertices of a graph so that adjacent vertices get different colors. (Edge coloring a graph will be addressed in the exercises.) If such an assignment can be carried out for G using at most k colors, we call it a *k-coloring* of G and

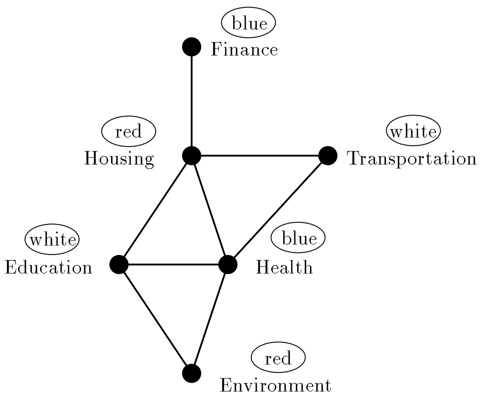


Figure 3.28: A 3-coloring for the graph G of Figure 1.1. The color assigned to a vertex is circled.

say G is *k-colorable*. The smallest number k such that G is k-colorable is called the *chromatic number* of G and is denoted $\chi(G)$. To illustrate these ideas, let us return to the graph of Figure 1.1 and call that graph G. A 3-coloring of G is shown in Figure 3.28. The three colors used are red, white, and blue. Note that this graph also has a 4-coloring. Indeed, by our definition, this figure shows a 4-coloring—we do not require that each color be used. A 4-coloring that uses four colors would be obtained from this coloring by changing the color of the Finance vertex (or for that matter any vertex) to green. We can always increase the number of colors used (up to the number of vertices). Thus, the emphasis is on finding the smallest number of colors we can use, that is, $\chi(G)$. Here, $\chi(G)$ obviously equals 3, for the three vertices Education, Housing, and Health must all get different colors. In this section we describe applications of graph coloring. As we remarked in Example 1.4, other applications with the same flavor as scheduling committee meetings involve scheduling final exam and class meeting times in a university, scheduling job assignments in a factory, and many other such problems. ■

Example 3.12 Index Registers and Optimizing Compilers (Tucker [1984]) In an optimizing compiler, it is more efficient to temporarily store the values of frequently used variables in index registers in the central processor, rather than in the regular memory, when computing in loops in a program. We wish to know how many index registers are required for storage in connection with a given loop. We let the variables that arise in the loop be vertices of a graph G and draw an edge in G between two variables if at some step in the loop, they will both have to be stored. Then we wish to assign an index register to each variable in such a way that if two variables are joined by an edge in G, they must be assigned to different registers. The minimum number of registers required is then given by the chromatic number of G. ■

Example 3.13 Channel Assignments Television transmitters in a region are to be assigned a channel over which to operate. If two transmitters are within 100 miles of each other, they must get different channels. The problem of assigning channels can be looked at as a graph coloring problem. Let the vertices of a graph G be the transmitters and join two transmitters by an edge if and only if they are within 100 miles of each other. Assign a color (channel) to each vertex so that if two vertices are joined by an edge, they get different colors. How few channels are needed for a given region? This is the chromatic number of G. (For more information on applications of graph coloring to television or radio-frequency assignments, see, for example, Cozzens and Roberts [1982], Hale [1980], Opsut and Roberts [1981], Roberts [1991], van den Heuvel, Leese, and Shepherd [1998], or Welsh and Whittle [1999].) ■

Example 3.14 Routing Garbage Trucks Let us next consider a routing problem posed by the Department of Sanitation of the City of New York (see Beltrami and Bodin [1973] and Tucker [1973]).[10] It should be clear that techniques like those to be discussed can be applied to other routing problems, for example milk routes and air routes. A garbage truck can visit a number of sites on a given day. A *tour* of such a truck is a schedule (an ordering) of sites it visits on a given day, subject to the restriction that the tour can be completed in one working day. We would like to find a set of tours with the following properties:

1. Each site i is visited a specified number k_i times in a week.
2. The tours can be partitioned among the six days of the week (Sunday is a holiday) in such a way that (a) no site is visited twice on one day[11] and (b) no day is assigned more tours than there are trucks.
3. The total time involved for all trucks is minimal.

In one method proposed for solving this problem, one starts with any given set of tours and improves the set successively as far as total time is concerned. (In the present state of the art, the method comes close to a minimal set, but does not always reach one.) At each step, the given improved collection of tours must be tested to see if it can be partitioned in such a way as to satisfy condition (2a), that is, partitioned among the six days of the week in such a way that no site is visited twice on one day. Thus, we need an efficient test for "partitionability" which can be applied over and over. Formulation of such a test reduces to a problem in graph coloring, and that problem will be the one on which we concentrate. (The reader is referred to Beltrami and Bodin [1973] and to Tucker [1973] for a description of the treatment of the total problem.)

To test if a given collection of tours can be partitioned so as to satisfy condition (2a), let us define a graph G, the *tour graph*, as follows. The vertices of G are

[10] Other applications of graph theory to sanitation are discussed in Section 11.4.3.

[11] Requirement (a) is included to guarantee that garbage pickup is spread out enough to make sure that there is no accumulation.

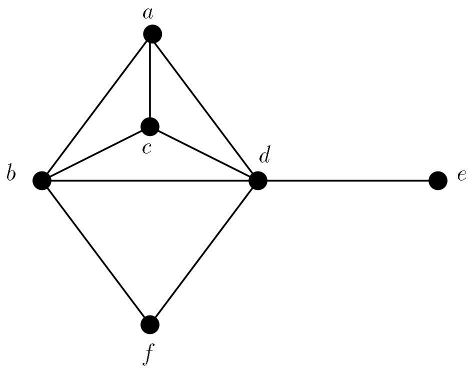


Figure 3.29: A graph of clique number 4.

the tours in the collection, and two distinct tours are joined by an edge if and only if they service some common site. Then the given collection of tours can be partitioned into six days of the week in such a way that condition (2a) is satisfied if and only if the collection of vertices $V(G)$ can be partitioned into six classes with the property that no edge of G joins vertices in the same class. It is convenient to speak of this question in terms of colors. Each class in the partition is assigned one of six colors and we ask for an assignment of colors to vertices such that no two vertices of the same color are joined by an edge.[12] The question about tours can now be rephrased as follows: Is the tour graph 6-colorable? ■

Example 3.15 Fleet Maintenance Vehicles (cars, trucks, ships, planes) come into a maintenance facility at scheduled times for regular maintenance. Two vehicles in the facility in overlapping time periods must be assigned different spaces. How many spaces does the facility require? This problem can also be formulated as a graph coloring problem. Let the vertices of a graph G be the vehicles scheduled for maintenance and join two vehicles by an edge if they are scheduled at overlapping times. Assign a color (space) to each vertex so that if two vertices are joined by an edge, they get different colors (spaces). The answer to our question is given by $\chi(G)$. ■

Example 3.16 Clique Number and Chromatic Number Suppose that G is a graph. A *clique* in G is a collection of vertices, each joined to the other by an edge. For instance, in the graph G of Figure 3.29, $\{a, b, c\}, \{d, e\}$, and $\{a, b, c, d\}$ are cliques. The *clique number* of G, $\omega(G)$, is the size of the largest clique of G. In our example, $\omega(G) = 4$. Since all vertices in a clique must receive different colors, this implies that $\chi(G) \geq \omega(G)$. Can you give an example of a graph in which $\chi(G) > \omega(G)$? ■

[12] This idea is due to Tucker [1973].

Example 3.17 Chromatic Number and Independence Number Suppose that G is a graph and W is a subset of the vertex set of G. W is called an *independent set* of G if no two vertices of W are joined by an edge. The *independence number* $\alpha(G)$ is the size of a largest independent set of G. For instance, in the graph of Figure 3.29, $\{a, f\}$ and $\{e, c, f\}$ are independent sets. There is no independent set of four vertices, so $\alpha(G) = 3$. Let us suppose that the vertices of G have been colored. There can be no edges between vertices of the same color, so all vertices of a given color define an independent set. Thus, a coloring of the $n = |V(G)|$ vertices of G in $\chi(G)$ colors partitions the vertices into $k = \chi(G)$ "color classes," each defining an independent set. The average size of such an independent set is $n/k = |V(G)|/\chi(G)$. Thus, by an application of the pigeonhole principle (Corollary 2.15.2), there is at least one independent set of size at least n/k, that is,

$$\alpha(G) \geq \frac{|V(G)|}{\chi(G)}$$

or

$$\chi(G)\alpha(G) \geq |V(G)|. \tag{3.9}$$ ■

Example 3.18 Course Scheduling Suppose that a university lets its professors schedule their classes at any time(s) during the week. Let us assume that any one course must be scheduled to consume three hours of classroom time. For example, one semester of classes might look as follows:

FALL 1999

Math 027: {Monday 3–4, Thursday 1–2:15, Friday 3–3:45}
Econ 321: {Tuesday 3–4:30, Thursday 3–4:30}
Span 114: {Monday 8–9, Wednesday 8–9, Friday 8–9}
⋮

How does a student pick classes that don't overlap? The university can build a graph where each vertex represents a different course at their school. If two courses have any time in which they overlap, draw an edge between those two vertices. A student wanting n courses must then pick n vertices which form an independent set in the graph. ■

Example 3.19 Map-coloring The problem of coloring maps is an old and important problem which has been one of the prime stimulants for the development of graph theory. To explain the map-coloring problem, let us consider the map of Figure 3.30. It is desired to color the countries on the map in such a way that if two countries share a common boundary, they get a different color. Of course, each country can be colored in a different color. However, for many years, cartographers have been interested in coloring maps with a small number of colors, if possible. We can start coloring the countries in the map of Figure 3.30 by coloring country 1 red (see Figure 3.31). Then country 2, which shares a boundary with country 1, must get a different color, say blue. Country 3 shares a boundary with each of the other

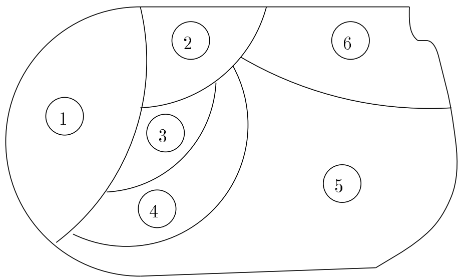


Figure 3.30: A map.

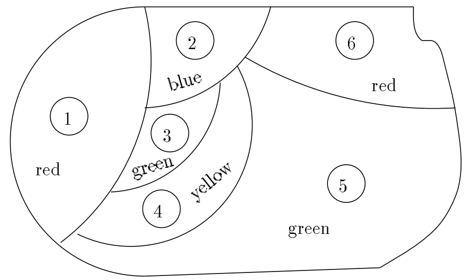


Figure 3.31: A coloring of the map in Figure 3.30.

countries colored so far, so it must get still a different color, say green. Country 4 shares a boundary with all of the first three countries, so it must get still a fourth color, say yellow. Country 5 shares a boundary with countries 1, 2, and 4, but not with 3. Thus, it is possible to color country 5 green. Finally, country 6 cannot be blue or green. In Figure 3.31, we have colored it red. Notice that the map has been colored with four colors. No one has ever found a map for which more than four colors are needed, provided that "map" and "boundary" are defined precisely so as to eliminate such things as countries having two pieces, countries whose common boundary is a single point, and so on. For more than 100 years, it was conjectured that every map could be colored in four or fewer colors. However, despite the work of some of the best mathematical minds in the world, this *four-color conjecture* was neither proved nor disproved, and the four-color problem remained unsolved. Finally, in 1977, the four-color conjecture was proved (see Appel and Haken [1977], Appel, Haken, and Koch [1977]). The original proof of the four-color theorem involved the use of high-speed computers to check certain difficult cases and involved some 1200 hours of computer time. (Recent work has led to "simpler" proofs of the four-color theorem; see, for example, Robertson, *et al.* [1997].)

One of the major steps in handling the map-coloring problem and k-colorings of maps was to translate the map-coloring problem into an equivalent but somewhat more tractable problem. Let the nation's capital of each country be represented by a point. Join two of these capitals by a (dashed) line if the corresponding countries share a common boundary. This gives rise to the lines of Figure 3.32. In Figure 3.33 the diagram is redrawn with only the capitals and the lines joining them remaining. This diagram defines a graph. Instead of coloring a whole country, we can think of just coloring its capital. In terms of a graph such as that in Figure 3.33, the requirement is that if two capitals or vertices are joined by an edge, they must get different colors. Thus, a map is colorable in k colors if and only if the corresponding graph is k-colorable. ■

Graph coloring and its generalizations have numerous applications in addition to those described here, for instance to time sharing in computer science, phasing traffic lights in transportation science, and various scheduling and maintenance

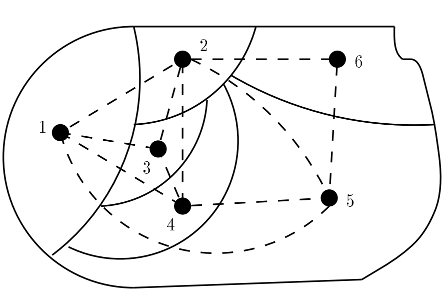


Figure 3.32: A dashed line joins two capitals if and only if their corresponding countries share a common boundary.

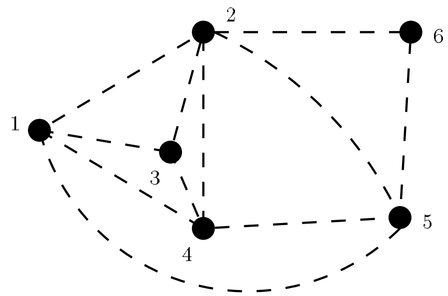


Figure 3.33: The graph of linked capitals from the map of Figure 3.32.

problems in operations research. See Opsut and Roberts [1981] and Roberts [1991] for descriptions of some of these problems.

3.3.2 Planar Graphs

The graph of Figure 3.33 has the property that no two edges cross except at vertices of the graph. A graph that has this property, or which has an equivalent (isomorphic) redrawing with this property, is called *planar*. Every map gives rise to a planar graph, and, conversely, every planar graph comes from a map. Thus, the four-color theorem can be stated as the following theorem in graph theory: Every planar graph is 4-colorable. The first graph of Figure 3.34 is planar, even though its drawing has edges crossing. The second graph of Figure 3.34, which is equivalent (isomorphic) to the first graph, is drawn without edges crossing. The first graph in Figure 3.34 is the complete graph K_4. Thus, K_4 is planar. The graph of Figure 3.35(a), which is K_5, is not planar. No matter how you locate five points in the plane, it is impossible to connect them all with lines without two of these lines crossing. The reader is encouraged to try this. The graph of Figure 3.35(b) is another example of a graph that is not planar. This graph is called the *water-light-gas graph* and is denoted by $K_{3,3}$. We think of three houses and three utilities and try to join each house to each utility. It is impossible to do this without some lines crossing. Again, the reader is encouraged to try this. The problem of determining if a graph is planar has a variety of applications. For instance, in electrical engineering, the planar graphs correspond exactly to the possible printed circuits. In Section 11.6.4 we show the use of planar graphs in a problem of facilities design. Kuratowski [1930] showed that in some sense, K_5 and $K_{3,3}$ are the only nonplanar graphs.[13]

To make precise the sense in which K_5 and $K_{3,3}$ are the only nonplanar graphs, let us say that graph G' is obtained from graph G by *subdivision* if we obtain G' by adding vertices on one edge of G. In Figure 3.36, graph G_i' is always obtained

[13] The rest of this subsection may be omitted.

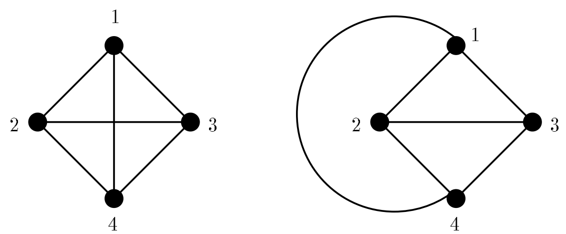


Figure 3.34: The first graph is planar, as is demonstrated by the second graph and the isomorphism shown by the vertex labelings.

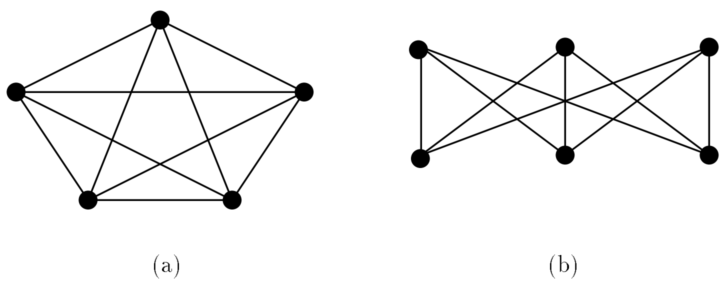


Figure 3.35: Two nonplanar graphs, K_5 and $K_{3,3}$.

from graph G_i by subdivision. Two graphs G and G' are called *homeomorphic* if both can be obtained from the same graph H by a sequence of subdivisions. For example, any two simple chains are homeomorphic. Figure 3.37 shows two graphs G and G' obtained from a graph H by a sequence of subdivisions. Thus, G and G' are homeomorphic (and, incidentally, homeomorphic to H).

Theorem 3.2 (Kuratowski [1930]) A graph is planar if and only if it has no subgraph[14] homeomorphic to K_5 or $K_{3,3}$.

For a proof of Theorem 3.2, we refer the reader to Harary [1969], Bondy and Murty [1976], or Makarychev [1997]. According to Kuratowski's Theorem, the graph G of Figure 3.38 is not planar because it is homeomorphic to K_5 and the graph G' is not planar because it has a subgraph H homeomorphic to $K_{3,3}$.

Suppose that $e = \{x, y\}$ is an edge of G. *Contracting* e means identifying the two vertices x and y. The new combined vertex is joined to all those vertices to which either x or y were joined. If both x and y were joined to a vertex z, only one of the edges from the combined vertex to z is included. In Figure 3.36, graph G_i'' is always obtained from graph G_i by contraction of edge e. *Contracting* G to G' means contracting a sequence of edges of G to obtain G'. Another characterization of planar graphs using contraction is given by the following theorem, due to Halin [1964], Harary and Tutte [1965], and Wagner [1937].

[14] Not necessarily a generated subgraph.

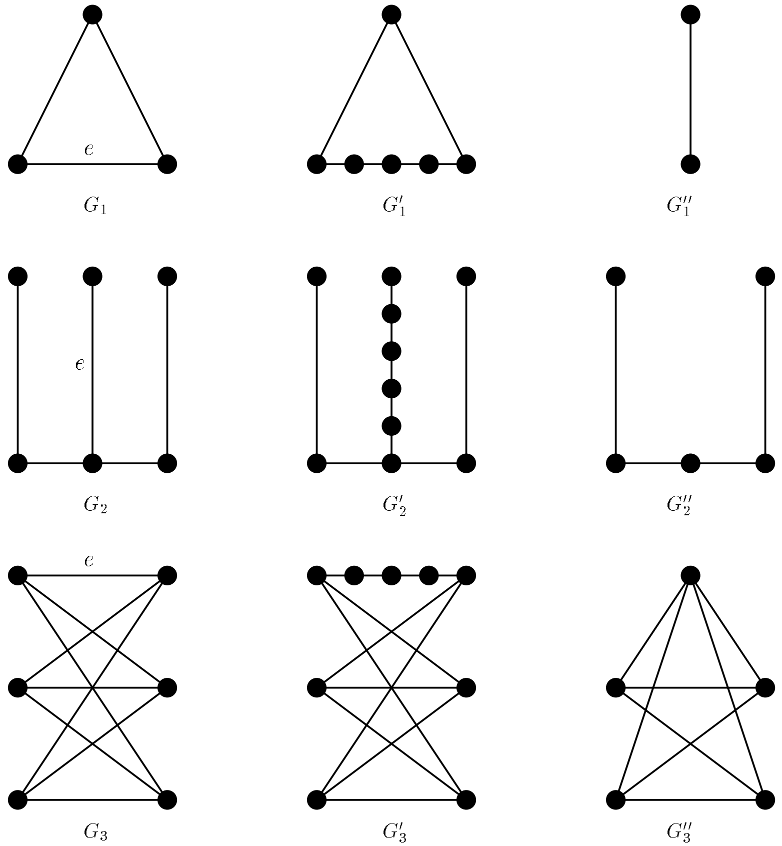


Figure 3.36: Graph G_i' is obtained from graph G_i by subdivision of edge e. Graph G_i'' is obtained from graph G_i by contraction of edge e.

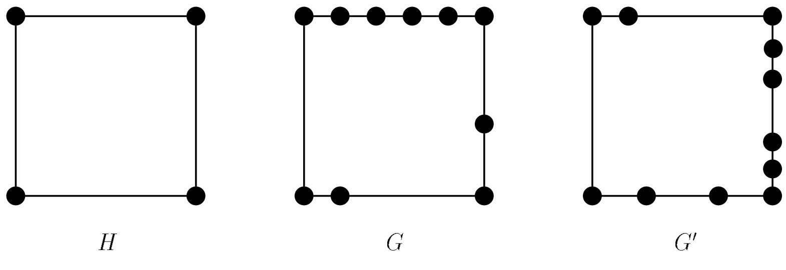


Figure 3.37: G and G' are homeomorphic because they are each obtained from H by subdivision.

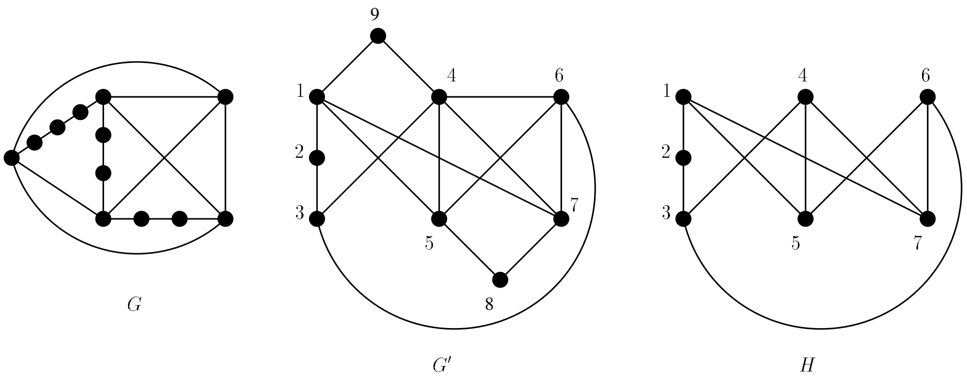


Figure 3.38: G is homeomorphic to K_5 and G' has the subgraph H which is homeomorphic and contractible to $K_{3,3}$.

Theorem 3.3 A graph G is planar if and only if no subgraph of G can be contracted to K_5 or $K_{3,3}$.

For a proof of Theorem 3.3, see Harary [1969], Bondy and Murty [1976], or Makarychev [1997]. By Theorem 3.3, G' of Figure 3.38 is not planar because it has a subgraph H that can be contracted to $K_{3,3}$ by contracting edge $\{1, 2\}$ or $\{2, 3\}$.

Before closing this subsection, we note that Kuratowski's Theorem does not give a good algorithm for testing a graph for planarity. However, there are such algorithms. For a good discussion of them, see Even [1979]. In particular, see Demourcron, Malgrance, and Pertuiset [1964] and Klotz [1989] for quadratic $[O(n^2)]$ algorithms or Hopcroft and Tarjan [1974] and Booth and Lueker [1976] for linear $[O(n)]$ algorithms.

3.3.3 Calculating the Chromatic Number

Let us study the colorability of various graphs. The graph K_4 is obviously colorable in four colors but not in three or fewer. Thus, $\chi(K_4) = 4$. The graph K_5 is colorable in five colors, but not in four or fewer (why?). Thus, $\chi(K_5) = 5$. (This is not a counterexample to the four-color theorem, since we have pointed out that K_5 is not a planar graph and so could not arise from a map.) $K_{3,3}$ in Figure 3.35 is colorable in two colors: Color the top three vertices red and the bottom three blue. Since clearly two colors are needed, the chromatic number of $K_{3,3}$ is 2.

Let us return briefly to the tour graph problem of Example 3.14. In general, to apply the procedure for finding a minimal set of tours, one has to have an algorithm, which can be applied quickly over and over again, for deciding whether a given graph is k-colorable. Unfortunately, there is not always a "good," that is, polynomial, algorithm for solving this problem. Indeed, in general, it is not known whether there is a "good" algorithm (in the sense of Section 2.4) for deciding if a given graph is k-colorable. This problem is NP-complete in the sense of Section 2.18, so is difficult in a precise sense. The garbage truck routing problem is thus reduced to a difficult mathematical question. However, formulation in precise mathematical terms has made it clear why this is a hard problem, and it has also given us many tools to use in solving it, at least in special cases. Let us remark that in a real-world situation, it is not sufficient to say that a problem is unsolvable or hard. Imagine a $500,000 consultant walking into the mayor's office and reporting that after careful study, he or she has concluded that the problem of routing garbage trucks is hard! Garbage trucks must be routed. So what can you do in such a situation? The answer is, you develop partial solutions, you develop solutions that are applicable only to certain special situations, you modify the problem, or in some cases, you even "lie." You lie by using results that are not necessarily true but seem to work. One such result is the Strong Perfect Graph Conjecture or the Strong Berge Conjecture, which goes back to Claude Berge [1961, 1962]. (To understand the following reasoning, it is not important to know what this conjecture says. See Golumbic [1980] for a detailed treatment of the conjecture or Roberts [1976, 1978] or Tucker [1973] for a treatment of the conjecture and its applications to garbage trucks and routing.) As Tucker [1973] pointed out, if the conjecture is true, there is an efficient algorithm for determining if a given graph is colorable in a given number of colors, at least in the context of the tour graph problem, where the tour graph is changed each time only locally and not globally. Thus, Tucker argued, it pays to "lie" and to use the Strong Berge Conjecture in routing garbage trucks. What could go wrong? The worst thing that could happen, said Tucker, is the following. One applies the conjecture to garbage truck routing and finds a routing that is supposedly assignable to the 6 days of the week, but which in fact cannot be so assigned. In this worst case, think of the boon to mathematics: We would have found a counterexample to the Strong Berge Conjecture! This remarkable argument, however, is no longer necessary. After over 40 years, the Strong Berge Conjecture was proved by Chudnovsky, *et al.* [2002] (see Mackenzie [2002]).

3.3.4 2-Colorable Graphs

Let us note next that there is one value of k for which it is easy to determine if G is k-colorable. This is the case $k = 2$.[15] A graph is 2-colorable if and only if the vertices can be partitioned into two classes so that all edges in the graph join vertices in the two different classes. (Why?) A graph with this kind of a partition is called *bipartite*. The depth-first search procedure to be described in Section 11.1 gives a polynomial algorithm for testing if a graph is bipartite (see Reingold, Nievergelt, and Deo [1977, pp. 399–400]).

There is also a useful characterization of 2-colorable graphs, which we state next. Let Z_p be the graph that consists of just a single circuit of p vertices. Figure 3.39 shows Z_3, Z_4, Z_5, and Z_6. It is easy enough to show that Z_4 and Z_6 are 2-colorable. A 2-coloring for each is shown. Clearly, Z_3 is not 2-colorable. Z_5 is also not 2-colorable. This takes a little proving, and we leave the proof to the reader. In

[15]For a discussion of other cases where there are good algorithms for determining if G is k-colorable, see Garey and Johnson [1979], Golumbic [1980], and Jensen and Toft [1995].

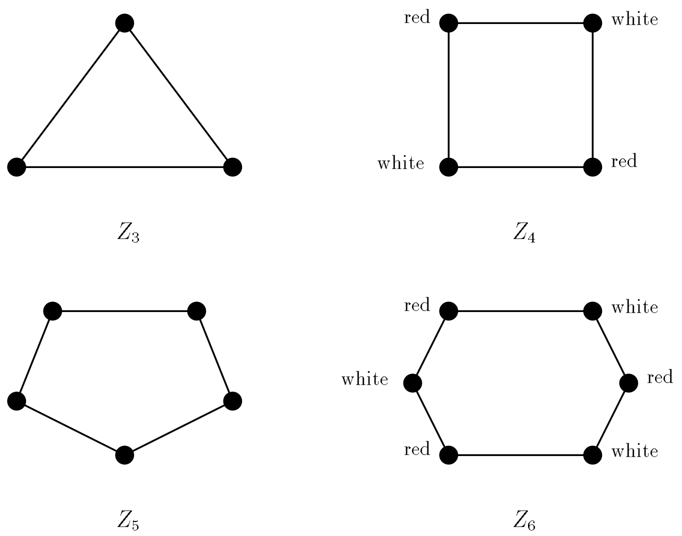


Figure 3.39: The graphs Z_p for $p = 3, 4, 5$, and 6.

general, it is easy to see that Z_p is 2-colorable if and only if p is even.

Now suppose that we start with Z_5, possibly add some edges, and then add new vertices and edges joining these vertices or joining them to vertices of Z_5. We might get graphs such as those in Figure 3.40. Now none of these graphs is 2-colorable. For a 2-coloring of the whole graph would automatically give a 2-coloring of Z_5. This is a general principle: A k-coloring of any graph G is a k-coloring of all subgraphs of G. Thus, any graph containing Z_5 as a subgraph is not 2-colorable. The same is true for Z_3, Z_7, Z_9, and so on. If G has any circuit of odd length, the circuit defines a subgraph of the form Z_p, for p odd; thus G could not be 2-colorable. The converse of this statement is also true, and we formulate the result as a theorem.

Theorem 3.4 (König [1936]) A graph is 2-colorable if and only if it has no circuits of odd length.

To prove the converse part of Theorem 3.4, we start with a graph G with no circuits of odd length and we present an algorithm for finding a 2-coloring of G. We may suppose that G is connected. (Otherwise, we can color each connected component separately.) Pick an arbitrary vertex x. Color x blue. Color all neighbors of x red. For each of these neighbors, color its uncolored neighbors blue. Continue in this way until all vertices are colored. The algorithm is illustrated by the graph of Figure 3.41, which is connected and has no odd-length circuits. Here, x is chosen to be a and the 2-coloring is shown.

To implement this algorithm formally, vertices that have been colored are saved in an ordered list called a *queue*. At each stage of the algorithm, we find the first vertex y in the queue and remove (pop) it from the queue. We find its uncolored neighbors and color them the opposite color of y. We then add these neighbors

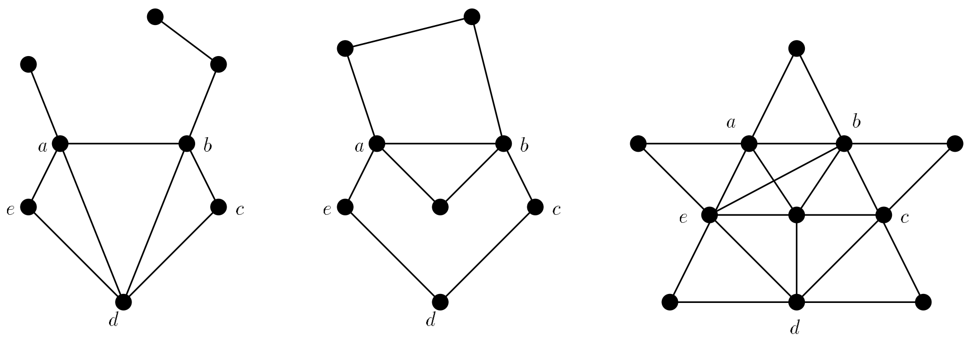


Figure 3.40: Graphs containing Z_5 as a subgraph. The vertices of Z_5 are labeled a, b, c, d, and e.

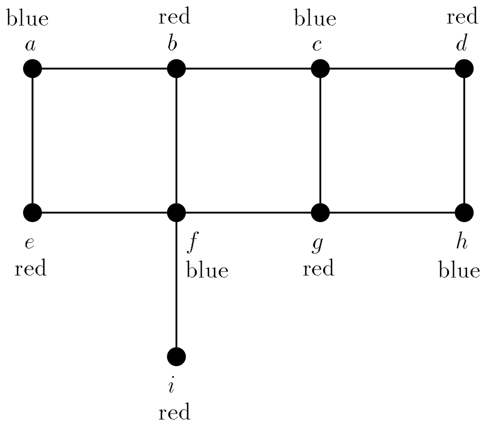


Figure 3.41: A connected graph without odd-length circuits. A 2-coloring obtained by first coloring vertex a is shown.

Table 3.2: Applying Algorithm 3.1 to the Graph of Figure 3.41

Vertices currently considered	Vertices colored	New queue Q
	a (blue)	a
a	b, e (red)	b, e
b	c, f (blue)	e, c, f
e	none	c, f
c	d, g (red)	f, d, g
f	i (red)	d, g, i
d	h (blue)	g, i, h

to the end of the queue. We continue until all vertices have been colored. The algorithm is formally stated as Algorithm 3.1. Here, Q is the queue.

Algorithm 3.1: Two-Coloring

Input: A graph $G = (V, E)$ that is connected and has no odd-length circuits.
Output: A coloring of the vertices of G using the two colors blue and red.

Step 1. Initially, all vertices of V are uncolored and Q is empty.

Step 2. Pick x in V, color x blue, and put x in Q.

Step 3. Let y be the first vertex in Q. Remove y from Q.

Step 4. Find all uncolored neighbors of y. Color each in the color opposite that used on y. Add them to the end of Q in arbitrary order.

Step 5. If all vertices are colored, stop. Otherwise, return to Step 3.

To illustrate Algorithm 3.1, consider the graph of Figure 3.41. The steps are summarized in Table 3.2. Pick x to be vertex a, color a blue, and put a into the queue. Find the uncolored neighbors of a, namely b and e. Color them red (that is, the opposite of a's color). Remove a from the queue Q and add b and e, say in the order b, e. Pick the first vertex in Q; here it is b. Remove it from Q. Find its uncolored neighbors; they are c and f. Color these the opposite color of that used for b, namely, blue. Add them to the end of Q in arbitrary order. If c is added first, then Q now is e, c, f. Remove the first vertex in Q, namely e. It has no uncolored neighbors. Q is now c, f. Go to the first vertex in Q, namely c, and remove it. Find its uncolored neighbors, d and g, and color them the opposite of c, namely, red. Add d and g to the end of Q, say d first. Q is now f, d, g. Next remove f from the head of Q, color its uncolored neighbor i red, and add i to the end of Q. Q is now d, g, i. Finally, remove d from the head of Q, color its uncolored neighbor h blue, and add h to the end of Q. Stop since all vertices have now been colored.

The procedure we have used to visit all the vertices is called *breadth-first search*. It is a very efficient computer procedure that has many applications in graph theory.

We shall return to breadth-first search, and the related procedure called depth-first search, in Section 11.1 when we discuss algorithms for testing a graph for connectedness. Algorithm 3.1 is a "good" algorithm in the sense of Sections 2.4 and 2.18. It is not hard to show that its complexity is of the order $n+e$, where n is the number of vertices of the graph and e is the number of edges. Since a graph has at most $\binom{n}{2}$ edges, we reason as in Section 3.2.3 to conclude that

$$e \leq \binom{n}{2} = \frac{n(n-1)}{2} \leq n^2.$$

Thus, Algorithm 3.1 takes at most a number of steps on the order of $n+n^2$, which is a polynomial in n. [In the notation of Section 2.18, the algorithm is $O(n^2)$.]

To show that Algorithm 3.1 works, we have to show that every vertex eventually gets colored and that we attain a graph coloring this way. Connectedness of the graph G guarantees that every vertex eventually gets colored. (We omit a formal proof of this fact.) To show that we get a graph coloring, suppose that u and v are neighbors in G. Could they get the same color? The easiest way to see that they could not is to define the distance $d(a,b)$ between two vertices a and b in a connected graph to be the length of the shortest chain between them. Then one can show that vertex z gets colored red if $d(x,z)$ is odd and blue if $d(x,z)$ is even.[16] (The proof is left as Exercise 31.) Now if two neighbors u and v are both colored red, there is a shortest chain C_1 from x to u of odd length and a shortest chain C_2 from x to v of odd length. It follows that C_1 plus edge $\{u,v\}$ plus C_2 (followed backwards) forms a closed chain from x to x of odd length. But if G has an odd-length closed chain, it must have an odd-length circuit (Exercise 32). Thus, we have a contradiction. We reach a similar contradiction if u and v are both colored blue.

Remember that Algorithm 3.1 and Theorem 3.4 apply only to 2-colorings. The general problem of graph coloring is an NP-complete problem; there is no known polynomial algorithm that determines k-colorability, for any fixed $k \geq 3$, let alone finding an actual optimal coloring.

3.3.5 Graph-Coloring Variants

There are many variations of graph (vertex) coloring. The following three examples only begin to scratch the surface of this burgeoning area in graph theory.

Example 3.20 Channel Assignment and the *T*-Coloring Problem Recall the problem of channel assignment from Example 3.13. The graph associated with this problem had the transmitters as vertices and an edge between vertices if the corresponding transmitters were within 100 miles of one another. Coloring this graph with as few colors as possible is tantamount to solving the channel assignment problem.

However, in certain situations not only can't "close" transmitters get the same channel but they can't get channels that differ by certain values. We call this set of

[16] The shortest proof of Theorem 3.4 is simply to *define* the coloring this way.

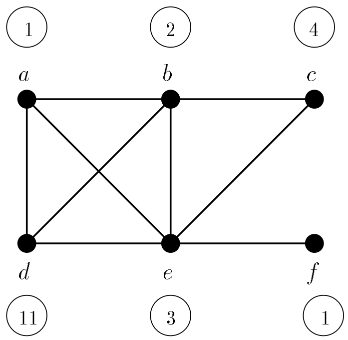


Figure 3.42: A T-coloring of a graph with T-set $T = \{0, 3, 4, 7\}$.

"forbidden" differences a *T-set* and denote it by T. For example, in UHF television, transmitters within 55 miles of one another can't have channels that differ by any value in the T-set $T = \{0, 1, 7, 14, 15\}$. Given a T-set T and a graph G, a vertex coloring of G in which adjacent vertices don't have colors whose absolute difference is in T is called a *T-coloring* of G. (*Absolute difference* refers to $|c_i - c_j|$ where c_i, c_j are colors.) Figure 3.42 gives an example of a T-coloring of a graph with a T-set $T = \{0, 3, 4, 7\}$. Notice that absolute differences in the colors of adjacent vertices equal 1, 2, 8, 9, and 10, none of which are in the T-set. We make the assumption that all T-sets contain 0. Otherwise, the very uninteresting case of all vertices being colored the same color would constitute a T-coloring. ■

In Example 3.20, we could have just as easily colored the vertices of Figure 3.42 with the colors 1, 100, 500, 1000, and 2000 (replacing 1, 2, 3, 4, and 11, respectively) to produce a T-coloring. However, this would not have been an efficient T-coloring. So, what do we mean by an efficient T-coloring? Just as minimizing the number of colors is primarily used as the criterion for efficient graph coloring, we need criteria for efficient T-colorings. With regard to channel assignment, sometimes we are concerned with the total number of channels used and other times we are concerned with the range of the channels used. Thus, sometimes we are interested in the total number of colors used in a T-coloring and sometimes we are more concerned with the range of colors used.

The *order* of a T-coloring refers to the total number of colors used. The order of the T-coloring in Figure 3.42 equals 5 since 5 distinct colors are used; 1, 2, 3, 4, and 11. The minimum number of colors needed to T-color a graph G (i.e., the minimum order) is called the *T-chromatic number of G* and is denoted $\chi_T(G)$. The T-coloring used in Figure 3.42 is not most efficient in this sense since G can be colored using 4 colors (but not 3). Vertices a, b, d, and e all need different colors since any two are joined by an edge. Thus, $\chi_T(G) \geq 4$. Vertices c and f can each be colored the same as a (or d), producing an order 4 T-coloring. So if G is the graph of Figure 3.42 and $T = \{0, 3, 4, 7\}$, then $\chi_T(G) = 4$.

The minimum order of a T-coloring of a graph G, $\chi_T(G)$, is not a new parameter, as the next theorem shows.

Theorem 3.5 (Cozzens and Roberts [1982]) For all graphs G and any T-set T,

$$\chi_T(G) = \chi(G).$$

Proof. Since we have assumed that 0 is contained in every T-set, any T-coloring of G will be (at the very least) a graph coloring. Thus, $\chi_T(G) \geq \chi(G)$.

Next, any graph coloring of G using j colors can be turned into a T-coloring using j colors in the following way. Without loss of generality, we can assume that the j colors in the graph coloring are the colors $1, 2, \ldots, j$. Replace color i with $i \cdot (t + 1)$, where t is the largest element in the T-set T. These new colors form a T-coloring of G (see Exercise 37). Therefore, a graph coloring using $\chi(G)$ colors can be turned into a T-coloring using the same number of colors. Thus, $\chi_T(G) \leq \chi(G)$.

We have shown that $\chi_T(G) \geq \chi(G)$ and $\chi_T(G) \leq \chi(G)$. Therefore,

$$\chi_T(G) = \chi(G). \qquad \text{Q.E.D.}$$

A more natural criterion for efficiency has to do with the range of colors used in a T-coloring. Many times in channel assignment it is not how many channels are assigned that's important but whether or not all requests for channels can fit into the allocated bandwidth. We define the *span* of a T-coloring to be the difference between the largest and smallest colors used. The span of the T-coloring in Figure 3.42 equals 10, which comes from $11 - 1$, 11 being the largest color and 1 being the smallest color used in the T-coloring. The smallest span over all T-colorings of a graph G is called the *T-span of G* and is denoted $sp_T(G)$. The T-coloring used in Figure 3.42 is most efficient with regard to span. Assuming that 1 is the smallest color used in any T-coloring, restricting yourself to the colors $1, 2, \ldots, 10$ will not allow for a T-coloring of this graph (see Exercise 38).

Both the T-chromatic number and T-span of a graph are not easy values to ascertain. In the notation of Section 2.18, calculation of each is in the class of NP-complete problems. However, for certain classes of graphs and specific T-sets, exact values for the T-span have been found. (See, for example, Bonias [1991], Liu [1991], Raychaudhuri [1985], Tesman [1993], or Wang [1985].)

Example 3.21 Task Assignment and the Set Coloring Problem[17] A large and complicated task, such as building an airplane, is actually made up of many subtasks. Some of these subtasks are incompatible and may not be performed at the same time. For example, some of the subtasks may require the same tools, resources, hangar space, and so on. The task assignment problem is the problem of scheduling subtasks so that only compatible subtasks may be scheduled in overlapping time periods.

To formulate this problem graph-theoretically, we let each vertex represent one of the subtasks of the larger task. Put an edge between two subtasks if they are

[17] From Opsut and Roberts [1981].

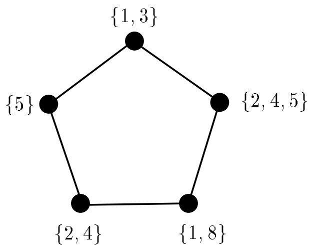


Figure 3.43: A set coloring for Z_5.

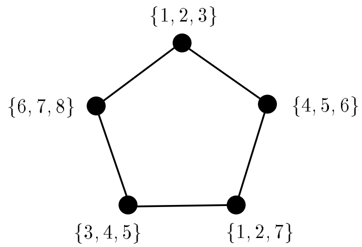


Figure 3.44: A 3-tuple coloring of Z_5.

incompatible. Then since each subtask will need a "time period," instead of assigning just one color to each vertex, we assign a set of colors which represent the times that the subtask will need. If x is a vertex, we denote the set assigned to x by $S(x)$. As before, we will require that adjacent vertices not receive the same colors. By this we mean that if x and y are adjacent, then $S(x)$ and $S(y)$ must not have any common members, that is,

$$S(x) \cap S(y) = \emptyset.$$

Such a coloring will be called a *set coloring* of the graph. Figure 3.43 gives an example of a set coloring for the graph Z_5. Note that the sets do not have to have the same size, which they don't in this example. As with vertex coloring, minimizing the total number of colors used will be the criterion for efficiency in set colorings. ■

To give a little more structure to set colorings, we consider the special case when all assigned sets are the same size. (Gilbert [1972] introduced this idea in connection with the frequency assignment problem involving mobile radios.) A *k-tuple coloring* will refer to a set coloring where each vertex is assigned a set of k colors. (If each set contains only one element, i.e., if we have a 1-tuple coloring, the set coloring is an ordinary graph coloring.) Figure 3.44 gives an example of a 3-tuple coloring of the graph Z_5 using 8 colors.

How many colors are needed to k-tuple color a graph? If the graph has n vertices, a crude upper bound would be kn colors. (Give each vertex a completely different set of k colors.) We can do much better than this, however. The following theorem gives an upper bound based on the chromatic number of the graph as opposed to the size of its vertex set. The proof of Theorem 3.6 is left as an exercise (Exercise 59).

Theorem 3.6 A graph G can be k-tuple colored in at most $k \cdot \chi(G)$ colors.

The minimum number of colors needed to k-tuple color a graph G is called the *k-tuple chromatic number* of G and is denoted by $\chi_k(G)$. From Theorem 3.6 we know that $\chi_k(G) \leq k \cdot \chi(G)$. This upper bound on $\chi_k(G)$ is, in fact, the exact value of $\chi_k(G)$ for many different classes of graphs, but it certainly does not hold in general. We have shown that we can 3-tuple color Z_5 using 8 colors, which is already better than $3 \cdot \chi(Z_5) = 3 \cdot 3 = 9$. (Irving [1983] has shown that finding

Table 3.3: A List Coloring for the List Assignment and Graph in Figure 3.45(b)

Vertex:	a	b	c	d	e	f
List:	$\{1,2\}$	$\{1,3\}$	$\{2,4\}$	$\{1,2\}$	$\{2,3\}$	$\{3,4\}$
Choose:	1	1	4	2	3	3

$\chi_k(G)$ is an NP-complete problem.) In fact, 8 is the 3-tuple chromatic number for Z_5. Let a and b be any pair of adjacent vertices. Then a and b cannot have any common colors. Without loss of generality, color vertex a with the colors 1, 2, 3 and color vertex b with the colors 4, 5, 6. It is then easy to show that the remaining vertices cannot be 3-tuple colored by using only one more additional color.

One class of graphs for which the bound in Theorem 3.6 is correct for determining the k-tuple chromatic number is the class of complete graphs. Since any two vertices are adjacent in a complete graph, no color can be repeated in a k-tuple coloring. Thus, we have the following theorem.

Theorem 3.7 For any n and k, $\chi_k(K_n) = k \cdot \chi(K_n) = kn$.

For other references to k-tuple colorings, see Brigham and Dutton [1982], Geller [1976], Roberts [1991], and Scott [1975].

Example 3.22 Scheduling Meetings of Legislative Committees (Example 1.4 Revisited): List Colorings In Example 1.4 we considered a scheduling problem where each committee chair provides a list of acceptable meeting times. The scheduling problem without this extra constraint was translated into a graph coloring problem in Example 3.11. With this constraint, it becomes the problem of finding a graph coloring in which the color assigned to a vertex is chosen from a list of acceptable colors. The same kind of problem arises in many other applications. For example, in channel assignment, the user of a given transmitter might specify a list of acceptable channels. Let $L(x)$ denote the list of colors assigned to vertex x. L is called a *list assignment* of G. A vertex coloring for G such that the color that you assign to a vertex x comes from the list $L(x)$ is called an L-*list coloring.*

Figure 3.45(a) shows an example of the graph $K_{3,3}$ with a list assignment. Can you find a list coloring for this assignment? Vertex a must be colored either 1 or 2. If it is colored 1, then vertex d must be colored 3 since there is an edge between vertex a and vertex d. Similarly, vertex f must be colored 2. But then there is no color left for vertex c. A similar problem occurs if vertex a is colored 2. Therefore, we have shown that this graph is not list colorable for this list assignment. However, if the lists assigned to the vertices were as in Figure 3.45(b) then choosing colors as in Table 3.3 gives selections that would work as a list coloring. ■

Example 3.23 List Colorings with Modified Lists In the committee scheduling example (Example 3.22), what would we do if there were no list coloring? We might ask some people to accept colors not on their original lists. One simple way

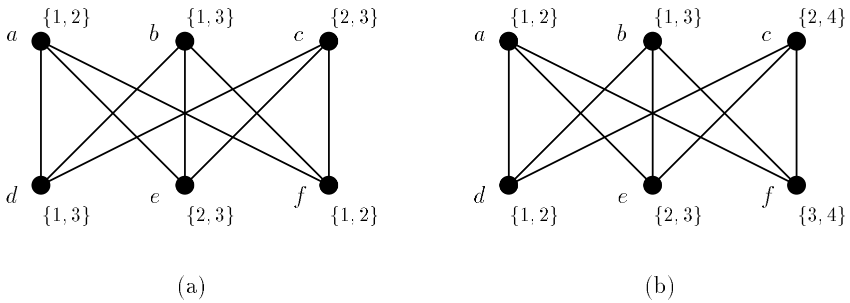


Figure 3.45: Two list assignments for $K_{3,3}$.

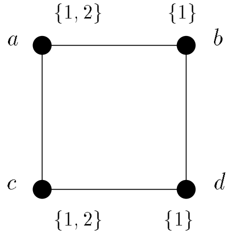


Figure 3.46: A 1-addable graph G.

to think of this is to allow some people x to expand their lists $L(x)$ by adding an additional color (from the available colors). What is the smallest number of people that have to do this? We say that G with list assignment L is *p-addable* if we can identify p distinct vertices $x_1, x_2, \ldots, x_p$ in G and (not necessarily distinct) colors $c_1, c_2, \ldots, c_p$ in $\bigcup L(x)$ so that if $L'(x_i) = L(x_i) \cup \{c_i\}$ for $i = 1, 2, \ldots, p$ and $L'(x) = L(x)$ otherwise, then there is a list coloring of G with list assignment L'. We are interested in calculating $I(G, L)$, the smallest p for which G with L is p-addable. To give a simple example, consider the graph G and list assignment L shown in Figure 3.46. Then there is no list coloring because vertices b and d must get different colors in any list coloring. However, adding color 2 to $L(d)$ makes this list colorable. Thus, G with L is 1-addable and $I(G, L) = 1$.

Consider the graph $G = K_{10,10}$, which has two classes, A and B, with 10 vertices each and edges between every vertex x in A and every vertex y in B. On vertices of A, use the ten 2-element subsets of $\{1, 2, 3, 4, 5\}$ as sets $L(x)$, and do the same on B. We shall show that $I(K_{10,10}, L) = 4$. Suppose we add a color to some sets $L(x)$ to get $L'(x)$ so there is a list coloring for $K_{10,10}$ with L'. Suppose the list coloring uses r colors on A and s colors on B. Then, of course, $r + s \leq 5$. Now $\binom{5-r}{2}$ sets on A do not use these r colors, so at least $\binom{5-r}{2}$ sets on A need a color added. Similarly, at least $\binom{5-s}{2}$ sets on B need a color added. This number of additions will work since all other sets on A have one of the r colors and similarly for B. It follows that

$$I\left(K_{10,10}, L\right) \leq \binom{5-r}{2} + \binom{5-s}{2},$$

with equality for some r and s. In fact, $r = 3$ and $s = 2$ give equality. Thus,

$$I\left(K_{10,10}, L\right) = \binom{5-3}{2} + \binom{5-2}{2} = 4.$$

Mahadev and Roberts [2003] prove that there are G and L so that $I(G, L)/|V(G)|$ is arbitrarily close to 1. This theorem has the interpretation that there are situations where almost everyone has to accept a color not on their original list! ■

Although the lists $L(x)$ may vary in size, let us consider the situation where the lists are all the same size. Then an important question for list coloring is: Given a graph G, what is the smallest c so that no matter what lists of size c are assigned to the vertices, you can always find a list coloring? If a graph G can be L-list colored for any lists of size c, we say that G is *c-choosable*. The smallest c for which G is c-choosable is defined to be the *choice number* of G and is denoted by $ch(G)$. As we saw in Figure 3.45, the graph $K_{3,3}$ is not 2-choosable. Even though the list assignment in Figure 3.45(b) can produce a list coloring, the list assignment in Figure 3.45(a) cannot. In Exercise 60 we ask the reader to show that $K_{3,3}$ is 3-choosable. Thus, $ch(K_{3,3}) = 3$.

Calculating the choice number of a graph is, like many of the other graph coloring problems we have studied, NP-complete. This was proven by Gravier [1996]. But like other coloring problems, bounds for the choice number in general and exact values for special classes of graphs can be obtained.

By definition, if a graph G is c-choosable, a list coloring exists for any list assignment of c colors to each vertex. In particular, a list coloring (which must also be a proper graph coloring) must exist if we assign the set $\{1, 2, \ldots, c\}$ to every vertex. Therefore, the chromatic number of G is at most c. Thus, we have shown that

$$ch(G) \geq \chi(G). \tag{3.10}$$

Equality in Equation (3.10) is sometimes attained for certain graphs. Consider the complete graph with n vertices, K_n. Recall that $\chi(K_n) = n$; every vertex is adjacent to every other vertex, thus requiring n colors. What is the choice number of K_n, $ch(K_n)$? It is easy to see that $ch(K_n) \leq n$. For when coloring a vertex by choosing an element from its list, the only colors not allowed are those that have been assigned to adjacent vertices, and there are $n - 1$ of these. Combining the observation that $ch(K_n) \leq n$ with Equation (3.10) shows that

$$ch(K_n) = n = \chi(K_n).$$

However, we certainly don't always attain equality in Equation (3.10). In fact, Erdős, Rubin, and Taylor [1979] have shown that the difference between the choice number and chromatic number can be arbitrarily large. See Alon [1993] and Kratochvíl, Tuza, and Voigt [1999] for further reading in this area.

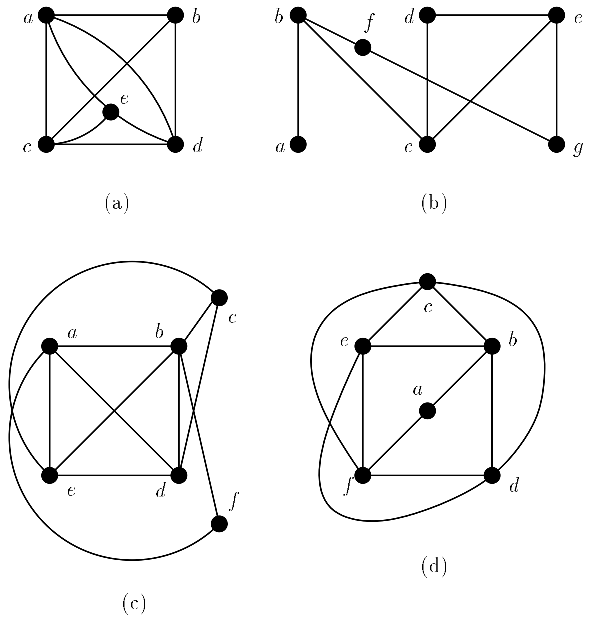


Figure 3.47: Graphs for exercises of Section 3.3.

EXERCISES FOR SECTION 3.3

1. Consider the following four tours of garbage trucks on the West Side of New York City. Tour 1 visits sites from 21st to 30th Streets, tour 2 visits sites from 28th to 40th Streets, tour 3 visits sites from 35th to 50th Streets, and tour 4 visits sites from 80th to 110th Streets. Draw the corresponding tour graph.
2. In Exercise 1, can the tours each be scheduled on Monday or Tuesday in such a way that no site is visited twice on the same day?
3. For each graph of Figure 3.47:

   (a) Determine if it is 3-colorable. (b) Determine its chromatic number $\chi(G)$.

4. A local zoo wants to take visitors on animal feeding tours and has decided on the following tours. Tour 1 visits the lions, elephants, and ostriches; tour 2 the monkeys, birds, and deer; tour 3 the elephants, zebras, and giraffes; tour 4 the birds, reptiles, and bears; and tour 5 the kangaroos, monkeys, and seals. If animals should not get fed more than once a day, can these tours be scheduled using only Monday, Wednesday, and Friday?
5. The following tours of garbage trucks in New York City are being considered (behind the mayor's back). Tour 1 picks up garbage at the Empire State Building, Madison Square Garden, and Pier 42 on the Hudson River. Tour 2 visits Greenwich Village, Pier 42, the Empire State Building, and the Metropolitan Opera House. Tour 3 visits Shea Stadium, the Bronx Zoo, and the Brooklyn Botanical Garden. Tour 4 goes to the Statue of Liberty and Pier 42; tour 5 to the Statue of Liberty, the

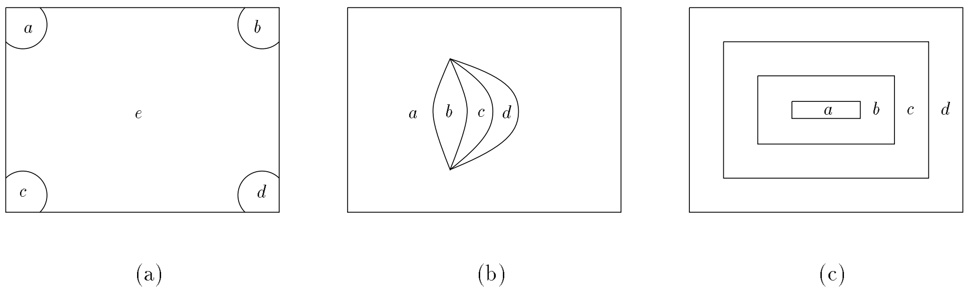


Figure 3.48: Maps.

New York Stock Exchange, and the Empire State Building; tour 6 to Shea Stadium, Yankee Stadium, and the Bronx Zoo; and tour 7 to the New York Stock Exchange, Columbia University, and the Bronx Zoo. Assuming that sanitation workers refuse to work more than three days a week, can these tours be partitioned so that no site is visited more than once on a given day?

6. The following committees need to have meetings scheduled.

$$\begin{aligned}
A &= \{\text{Smith, Jones, Brown, Green}\} \\
B &= \{\text{Jones, Wagner, Chase}\} \\
C &= \{\text{Harris, Oliver}\} \\
D &= \{\text{Harris, Jones, Mason}\} \\
E &= \{\text{Oliver, Cummings, Larson}\}.
\end{aligned}$$

Are three meeting times sufficient to schedule the committees so that no member has to be at two meetings simultaneously? Why?

7. In assigning frequencies to mobile radio telephones, a "zone" gets a frequency to be used by all vehicles in that zone. Two zones that interfere (because of proximity or for meteorological reasons) must get different frequencies. How many different frequencies are required if there are 6 zones, a, b, c, d, e, and f, and zone a interferes with zone b only; b with a, c, and d; c with b, d, and e; d with b, c, and e; e with c, d, and f; and f with e only?

8. In assigning work areas to workers, we want to be sure that if two such workers will interfere with each other, they will get different work areas. How many work areas are required if there are six workers, a, b, c, d, e, and f, and worker a interferes with workers b, e, and f; worker b with workers a, c, and f; worker c with b, d, and f; worker d with c, e, and f; e with a, d, and f; and f with all other workers?

9. In a given loop of a program, six variables arise. Variable A must be stored in steps 1 through 4, variable B in steps 3 through 6, variable C in steps 4 through 7, variable D in steps 6 through 9, variable E in steps 8 and 9, and variable F in steps 9 and 10. How many index registers are required for storage?

10. Find the graphs corresponding to the maps of Figure 3.48. Note that a single common point does not qualify as a common boundary.

11. Translate the map of Figure 3.49 into a graph G and calculate $\chi(G)$.

12. For each graph of Figure 3.50:

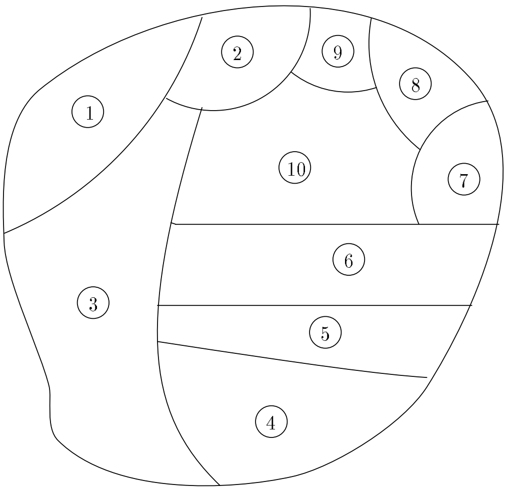


Figure 3.49: Map for exercises of Section 3.3.

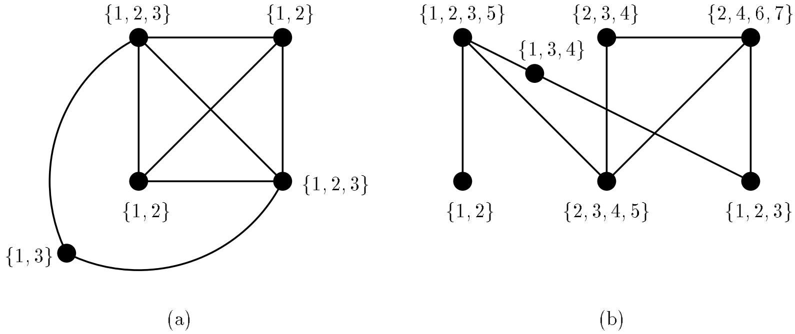


Figure 3.50: Graphs for exercises.

(a) Find $\omega(G)$. (b) Find $\alpha(G)$.

(c) Find $\chi(G)$ and verify the inequality (3.9).

13. Let G be any graph of 17 vertices and chromatic number 4.

    (a) Does G necessarily have an independent set of size 4?

    (b) Does G necessarily have an independent set of size 5?

14. If the vertices of the circuit of length 11, Z_{11}, are colored in four colors, what can you say about the size of the largest set of vertices each of which gets the same color?

15. Give examples of graphs G so that:

    (a) $\chi(G) = \omega(G)$ (b) $\chi(G) > \omega(G)$

16. Let $\theta(G)$ be the size of the smallest set S of cliques of G so that every vertex is in some clique of S. What is the relation between $\theta(G)$ and $\chi(G)$?

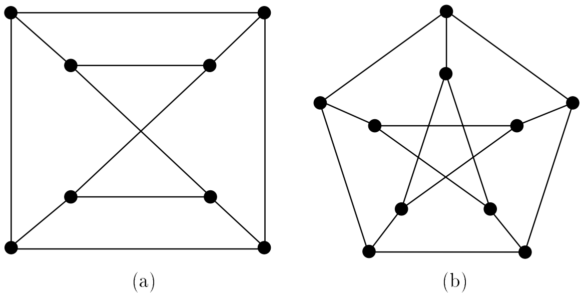


Figure 3.51: Two nonplanar graphs.

17. Can any graph containing K_5 as a subgraph be planar? Why?
18. Suppose that there are four houses and four utilities, and each house is joined by an edge to each utility. Is the resulting graph planar? Why?
19. Could K_6 come from a map? Why?
20. (a) Can K_3 be obtained by subdivision from K_2?
    (b) Are K_2 and K_3 homeomorphic?
21. Which of the graphs of Figure 3.47 are planar?
22. Prove that graph (a) of Figure 3.51 is nonplanar using:
    (a) Kuratowski's Theorem (Theorem 3.2) (b) Theorem 3.3
23. Repeat Exercise 22 for graph (b) of Figure 3.51.
24. For all graphs of Figure 3.47, find the graph obtained by contracting the edge $\{c, e\}$.
25. For all graphs of Figure 3.47, find the graph obtained by contracting edges $\{a, b\}$, $\{b, c\}$, and $\{c, d\}$ in that order.
26. Let $\deg(u)$, the *degree of vertex u* in a graph G, be defined to be the number of neighbors of u. Let $\Delta(G)$ be the maximum over all vertex degrees in G. Show that $\chi(G) \leq 1 + \Delta(G)$.
27. Give an example of a graph G such that $\chi(G) < 1 + \Delta(G)$, where $\Delta(G)$ is as defined in Exercise 26.
28. A graph G in which $\chi(G) = \omega(G)$ is called *weakly γ-perfect*. Give an example of a graph that is weakly γ-perfect and a graph that is not weakly γ-perfect.
29. Show that every 2-colorable graph is weakly γ-perfect.
30. Illustrate Algorithm 3.1 on graphs (c) and (g) of Figure 3.23.
31. Show that under Algorithm 3.1, vertex z gets colored red if $d(x, z)$ is odd and blue if $d(x, z)$ is even where x is the starting vertex.
32. Show that if a graph has an odd-length closed chain, it has an odd-length circuit.
33. Prove that $\chi(Z_p) = 3$ if and only if p is odd.
34. (a) Find a T-coloring for K_4 using $T = \{0, 1, 2\}$.
    (b) Find a T-coloring for Z_6 using $T = \{0, 2, 4\}$.
    (c) Find a T-coloring for the graph in Figure 3.52 using $T = \{0, 1, 2, 4, 8, 15\}$ and having order 6.

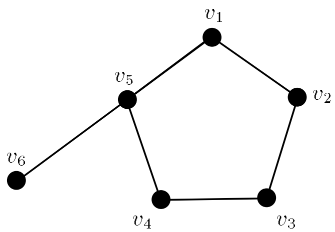


Figure 3.52: Graph for Section 3.3 exercises.

(d) Find a T-coloring for K_4 using $T = \{0, 1, 3\}$ and so that the span of the T-coloring is 7.

35. Find the T-span of K_4 when using the T-set $\{0, 1, 3, 4\}$.

36. When finding an efficient T-coloring, minimizing the order and minimizing the span may require different T-colorings. Recall that $\chi(Z_5) = 3$ and by Theorem 3.8, $\chi_T(Z_5) = 3$ for any T-set.

    (a) Find a T-coloring of Z_5 with the T-set $\{0, 1, 4, 5\}$ using 3 colors that has smallest possible span.

    (b) Find a T-coloring of Z_5 with the T-set $\{0, 1, 4, 5\}$ using 4 colors that has smallest possible span.

    (c) Find a T-coloring of Z_5 with the T-set $\{0, 1, 4, 5\}$ using 5 colors that has smallest possible span.

    (d) Is there *one* T-coloring of Z_5 with the T-set $\{0, 1, 4, 5\}$ that can minimize both order and span?

37. Prove that from a vertex coloring of a graph G using the colors $1, 2, \ldots, j$, a T-coloring will be produced by replacing color i with $i \cdot (t + 1)$, where t represents the largest color in the given T-set.

38. Prove that the T-coloring in Figure 3.42 is most efficient with regard to span. That is, prove that the graph of Figure 3.42 cannot be T-colored, with $T = \{0, 3, 4, 7\}$, using the colors $1, 2, \ldots, n$ where $n \leq 10$.

39. Find a minimum set coloring for the graph in Figure 3.52 which has two colors assigned to each vertex of even subscript and three colors assigned to each vertex of odd subscript.

40. Find a 2-tuple coloring of Z_4 using 5 colors.

41. Recall that $\chi_k(G)$ is the smallest m so that G has a k-tuple coloring using m colors. Find:

    (a) $\chi_2(Z_4)$ (b) $\chi_2(Z_3)$ (c) $\chi_2(K_4)$
    (d) $\chi_3(Z_4)$ (e) $\chi_3(Z_3)$ (f) $\chi_3(K_4)$

42. Find $\chi_2(G)$ for G the graph of Figure 3.52.

43. If G is 2-colorable and has at least one edge, show that $\chi_m(G) = 2m$.

44. Prove that $\chi_3(Z_5) = 8$.

45. For each graph of Figure 3.50, determine if it is list colorable with the given list assignment.

46. Show that (G, L) is p-addable for some p if and only if

$$\left|\bigcup\{L(x) : x \in V\}\right| \geq \chi(G).$$

47. The graph $K_{\binom{m}{2},\binom{m}{2}}$ has two classes of $\binom{m}{2}$ vertices, A and B, and every vertex x in A is adjacent to every vertex y in B. Let L give all 2-element subsets of $\{1, 2, \ldots, m\}$ to vertices of A and similarly for vertices of B. Find

$$I\left(K_{\binom{m}{2},\binom{m}{2}}, L\right).$$

48. Let $K_{7,7}$ be defined analogously to $K_{10,10}$ in Example 3.23. Let $|L(x)| = 3$ for all x and $|\bigcup L(x)| = 6$. Show that $K_{7,7}$ with L is 1-addable.

49. Show that:

    (a) Z_3 is not 2-choosable. (b) Z_3 is 3-choosable.

50. (a) Determine the choice number for Z_4.

    (b) Determine the choice number for Z_n, n even.

51. Suppose that a k-tuple coloring of a graph with n vertices is given. If we consider the k-tuple coloring as a list assignment of the graph, how many different list colorings are possible?

52. A graph G is *k-edge-colorable* if you can color the edges with k colors so that two edges with a common vertex get different colors. Let $\chi'(G)$, the *edge chromatic number*, be the smallest k so that G is k-edge-colorable. State a relation between $\chi'(G)$ and the number $\Delta(G)$ defined in Exercise 26. (For applications of edge coloring, see Fiorini and Wilson [1977].)

53. If G has n vertices, show that

$$\frac{n}{\alpha(G)} \leq \chi(G) \leq n - \alpha(G) + 1.$$

54. A graph G is called *k-critical* if $\chi(G) = k$ but $\chi(G-u) < k$ for each vertex $u \in V(G)$.

    (a) Find all 2-critical graphs.

    (b) Give an example of a 3-critical graph.

    (c) Can you identify *all* 3-critical graphs?

55. If $G = (V, E)$ is a graph, its *complement* G^c is the graph with vertex set V and an edge between $x \neq y$ in V if and only if $\{x, y\} \notin E$.

    (a) Comment on the relationship between the clique number $\omega(G)$ and the vertex independence number $\alpha(G^c)$.

    (b) Recall that $\theta(G)$ is the smallest number of cliques which cover all vertices of G. Show that $\chi(G) = \theta(G^c)$.

(c) Say that G is *weakly* α*-perfect* if $\theta(G) = \alpha(G)$. Give an example of a graph that is weakly α-perfect and an example of a graph that is not weakly α-perfect.

56. G is said to be γ*-perfect* (α*-perfect*) if every generated subgraph of G is weakly γ-perfect (weakly α-perfect). Give examples of graphs that are:

(a) γ-perfect (b) weakly γ-perfect but not γ-perfect

(c) α-perfect (d) weakly α-perfect but not α-perfect

57. Lovász [1972a,b] shows that a graph G is γ-perfect if and only if it is α-perfect. Hence, a graph that is γ-perfect (or α-perfect) is called *perfect*. For more on perfect graphs and their many applications, see Golumbic [1980].

(a) Show that it is not true that G is weakly γ-perfect if and only if G is weakly α-perfect.

(b) Show that G is γ-perfect if and only if G^c is γ-perfect. (You may use Lovász's result.)

58. (Tutte [1954], Kelly and Kelly [1954], Zykov [1949]) Show that for any integer $k > 1$, there is a graph G such that $\omega(G) = 2$ and $\chi(G) = k$.

59. Prove Theorem 3.6.

60. Figure 3.45(a) gave a list assignment to $K_{3,3}$ showing that $K_{3,3}$ was not 2-choosable. This exercise gives a proof that $K_{3,3}$ is 3-choosable. Let L be a list assignment of $K_{3,3}$ where each list is of size 3.

(a) Suppose that two nonadjacent vertices' lists share a common color. Show that an L-list coloring exists by using the common color.

(b) Suppose that every pair of nonadjacent vertices' lists do not share a common color. Show that an L-list coloring exists.

(c) Show why $ch(K_{3,3}) = 3$.

3.4 CHROMATIC POLYNOMIALS[18]

3.4.1 Definitions and Examples

Suppose that G is a graph and $P(G,x)$ counts the number of ways to color G in at most x colors. The related idea of counting the number of ways to color a map (see Exercise 3) was introduced by Birkhoff [1912] in an attack on the four-color conjecture—we discuss this below. The numbers $P(G,x)$ were introduced by Birkhoff and Lewis [1946]. Note that $P(G,x)$ is 0 if it is not possible to color G using x colors. Now $\chi(G)$ is the smallest positive integer x such that $P(G,x) \neq 0$. One of the primary reasons for studying the numbers $P(G,x)$ is to learn something about $\chi(G)$. In this section we study the numbers $P(G,x)$ in some detail, making heavy use of the counting techniques of Chapter 2.

[18]This section is not needed for what follows, except in Section 7.1.6. However, the reader is strongly encouraged to include it, for it provides many applications of the counting techniques of Chapter 2.

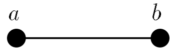


Figure 3.53: The graph K_2.

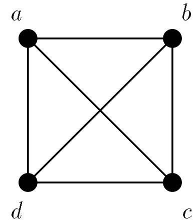


Figure 3.54: The graph K_4.

Table 3.4: Colorings of Graph K_2 of Figure 3.53 with Colors Red (R), Green (G), Blue (B), and Yellow (Y)

a	R	R	R	G	G	G	B	B	B	Y	Y	Y
b	G	B	Y	R	B	Y	R	G	Y	R	G	B

Consider first the graph K_2 shown in Figure 3.53. If x colors are available, any one of them can be used to color vertex a, and any one of the remaining $x-1$ colors can be used to color vertex b. Hence, by the product rule,

$$P(K_2, x) = x(x-1) = x^2 - x.$$

In particular,

$$P(K_2, 4) = 16 - 4 = 12.$$

The 12 ways of coloring K_2 in at most 4 colors are shown in Table 3.4.

Consider next the graph K_4 of Figure 3.54. If x colors are available, there are x choices of color for vertex a. For each of these choices, there are $x-1$ choices for vertex b, since b has an edge to a; for each of these there are $x-2$ choices for vertex c, since c has edges to both a and b; for each of these there are $x-3$ choices for vertex d, since d has edges to each of a, b, and c. Hence,

$$P(K_4, x) = x(x-1)(x-2)(x-3) = x^4 - 6x^3 + 11x^2 - 6x.$$

To color the graph of Figure 3.55 in x or fewer colors, there are x choices for vertex f. Then there are $x-1$ choices left for vertex a, and for each of these also $x-1$ choices for vertex b (since b may get the same color as a), and for each of these $x-1$ choices for vertex c, and for each of these $x-1$ choices for vertex d, and for each of these $x-1$ choices for vertex e. Hence,

$$P(G, x) = x(x-1)^5 = x^6 - 5x^5 + 10x^4 - 10x^3 + 5x^2 - x.$$

Let us turn now to the graph Z_4 of Figure 3.56. Here a and b must get different colors, but a and c could get the same color. Similarly, b and d could get the same color. And so on. If x colors are available, there are x choices of color for a. Then b and d can get either the same color or a different one. It is convenient to think of two cases.

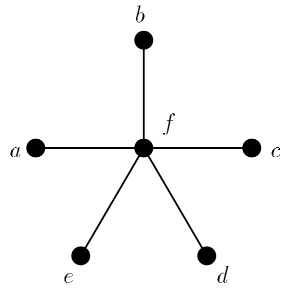


Figure 3.55: A graph.

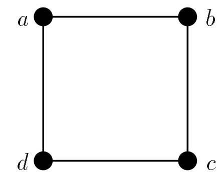


Figure 3.56: The graph Z_4.

Case 1. b and d get the same color.

Case 2. b and d get different colors.

In Case 1, c can get any of the colors not used for b and d. Hence, there are x choices for a, for each of these $x-1$ choices for the common color for b and d, and for each of these $x-1$ choices for the color for c. Hence, the number of colorings with x or fewer colors in which b and d get the same color is

$$x(x-1)^2.$$

In Case 2, there are x choices for a, $x-1$ for b, $x-2$ for d (since it must get a different color than b), and then $x-2$ choices for c (since it cannot receive either of the distinct colors used on b and d, but it can receive a's color). Hence, the number of colorings in which b and d get different colors is

$$x(x-1)(x-2)^2.$$

Since either Case 1 or Case 2 holds, the sum rule gives us

$$P(Z_4, x) = x(x-1)^2 + x(x-1)(x-2)^2 = x^4 - 4x^3 + 6x^2 - 3x. \tag{3.11}$$

The reader will notice that in each of the examples we have given, $P(G, x)$ is a polynomial in x. This will always be the case. Hence, it makes sense to call $P(G, x)$ the *chromatic polynomial.*

Theorem 3.8 $P(G, x)$ is always a polynomial.[19]

We will prove this theorem below. Recall that $\chi(G)$ is the smallest positive integer x such that $P(G, x) \neq 0$, that is, such that x is not a root of the polynomial $P(G, x)$. Thus, the chromatic number can be estimated by finding roots of a polynomial. Birkhoff's approach to the four-color problem was based on the idea of trying to characterize what polynomials were chromatic polynomials, in particular, of maps (or planar graphs), and then seeing if 4 was a root of any of these polynomials. To this day, the problem of characterizing the chromatic polynomials is not yet solved. We return to this problem below.

[19] This theorem was discovered for maps (equivalently, for graphs arising from maps) by Birkhoff [1912]. For all graphs, it is due to Birkhoff and Lewis [1946].

Example 3.24 Scheduling Meetings of Legislative Committees (Example 3.11 Revisited) Let us count the number of colorings of the graph G of Figure 3.28 using three or fewer colors. We start by computing $P(G, x)$. If there are x choices for the color of Education, there are $x - 1$ for the color of Housing, and then $x - 2$ for the color of Health. This leaves $x - 2$ choices for the color of Transportation, and $x - 2$ choices for the color of Environment, and finally, $x - 1$ choices for the color of Finance. Hence,

$$P(G, x) = x(x-1)^2(x-2)^3$$

and

$$P(G, 3) = 12.$$

This result agrees with the conclusion in our discussion of Example 1.4. There we described in Table 1.7 the 12 possible colorings in three or fewer colors. ■

Next, we state two simple but fundamental results about chromatic polynomials. One of these is about the graph I_n with n vertices and no edges, the *empty graph.*

Theorem 3.9 (a) If G is K_n, then

$$P(G, x) = x(x-1)(x-2)\cdots(x-n+1). \tag{3.12}$$

(b) If G is I_n, then $P(G, x) = x^n$.

Proof. (a) There are x choices for the color of the first vertex, $x - 1$ choices for the color of the second vertex, and so on.

(b) There are x choices for the color of each of the n vertices. Q.E.D.

The expression on the right-hand side of (3.12) will occur so frequently that it is convenient to give it a name. We call it $x^{(n)}$.

3.4.2 Reduction Theorems

A very common technique in combinatorics is to reduce large computations to a set of smaller ones. We employ this technique often in this book, and in particular in Chapter 6 when we study recurrence relations and reduce calculation of x_n to values of x_k for k smaller than n. This turns out to be a very useful technique for the computation of chromatic polynomials. In this subsection we develop and apply a number of reduction theorems that can be used to reduce the computation of any chromatic polynomial to that of a chromatic polynomial for a graph with fewer edges, and eventually down to the computation of chromatic polynomials of complete graphs and/or empty graphs.

We are now ready to state the first theorem.

Theorem 3.10 (The Two-Pieces Theorem) Let the vertex set of G be partitioned into disjoint sets W_1 and W_2, and let G_1 and G_2 be the subgraphs generated by W_1 and W_2, respectively. Suppose that in G, no edge joins a vertex of W_1 to a vertex of W_2. Then

$$P(G, x) = P(G_1, x)P(G_2, x).$$

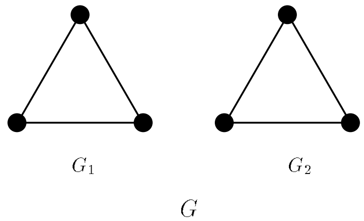


Figure 3.57: A graph with two pieces.

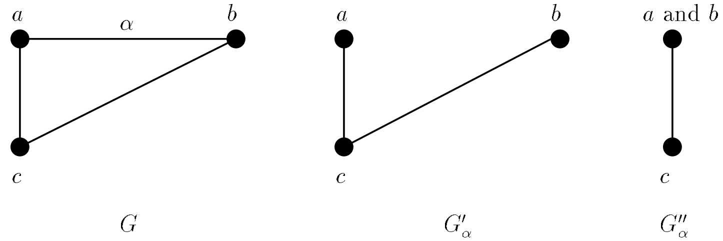


Figure 3.58: The graphs G'_α and G''_α.

Proof. If x colors are available, there are $P(G_1, x)$ colorings of G_1; for each of these there are $P(G_2, x)$ colorings of G_2. This is because a coloring of G_1 does not affect a coloring of G_2, since there are no edges joining the two pieces. The theorem follows by the product rule. Q.E.D.

To illustrate the theorem, we note that if G is the graph shown in Figure 3.57, then

$$P(G, x) = (x^{(3)})(x^{(3)}) = (x^{(3)})^2 = [x(x-1)(x-2)]^2.$$

To state our next reduction theorem, the crucial one, suppose that α is an edge of the graph G, joining vertices a and b. We define two new graphs from G. The graph G'_α is obtained by deleting the edge α but retaining the vertices a and b. The graph G''_α is obtained by identifying the two vertices a and b. In this case, the new combined vertex is joined to all those vertices to which either a or b were joined. (If both a and b were joined to a vertex c, only one of the edges from the combined vertex is included.) (In Section 3.3.2 we said that G''_α is obtained from G by contracting the edge α.) Figures 3.58 and 3.59 illustrate the two new graphs.

Theorem 3.11 (Fundamental Reduction Theorem[20])

$$P(G, x) = P(G'_\alpha, x) - P(G''_\alpha, x). \tag{3.13}$$

Proof. Suppose that we use up to x colors to color G'_α where edge α joins vertices a and b in G. Then either a and b receive different colors or a and b receive the same color. The number of ways of coloring G'_α so that a and b receive different

[20] This theorem is due to Birkhoff and Lewis [1946].

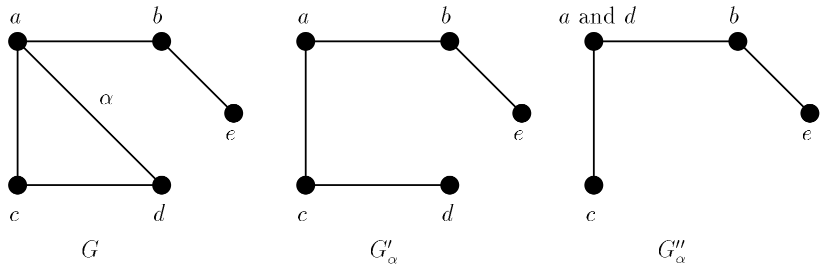


Figure 3.59: Another example of graphs G'_α and G''_α.

colors is simply the same as the number of ways of coloring G, that is, $P(G,x)$. The number of ways of coloring G'_α so that a and b get the same color is the same as the number of ways of coloring G''_α, namely $P(G''_\alpha, x)$. For we know that in G''_α, a and b would get the same color, and moreover the joint vertex a and b is forced to get a different color from that given to a vertex c if and only if one of a and b, and hence both, is forced to get a different color from that given to vertex c. The result now follows by the sum rule:

$$P(G'_\alpha, x) = P(G,x) + P(G''_\alpha, x). \qquad \text{Q.E.D.}$$

To illustrate the theorem, let us consider the graph G of Figure 3.60 and use the Fundamental Reduction Theorem to calculate its chromatic polynomial. We choose the edge between vertices 1 and 3 to use as α and so obtain G'_α and G''_α as shown in the figure. Now the graph G''_α of Figure 3.60 is the complete graph K_2, and hence by Theorem 3.9 we know that

$$P(G''_\alpha, x) = x(x-1). \tag{3.14}$$

The graph G'_α has two pieces, K_1 and K_2. By the Two-Pieces Theorem,

$$P(G'_\alpha, x) = P(K_1, x)P(K_2, x). \tag{3.15}$$

By Theorem 3.9, the first expression on the right-hand side of (3.15) is x and the second expression is $x(x-1)$. Hence,

$$P(G'_\alpha, x) = x \cdot x(x-1) = x^2(x-1). \tag{3.16}$$

Substituting (3.14) and (3.16) into (3.13), we obtain

$$\begin{aligned} P(G,x) &= x^2(x-1) - x(x-1) \\ &= x(x-1)(x-1) \\ &= x(x-1)^2. \end{aligned}$$

This expression could of course have been derived directly. However, it is a good illustration of the use of the Fundamental Reduction Theorem. Incidentally, applying the Fundamental Reduction Theorem again to $G''_\alpha = K_2$, we have

$$P(G''_\alpha, x) = P(K_2, x) = P(I_2, x) - P(I_1, x), \tag{3.17}$$

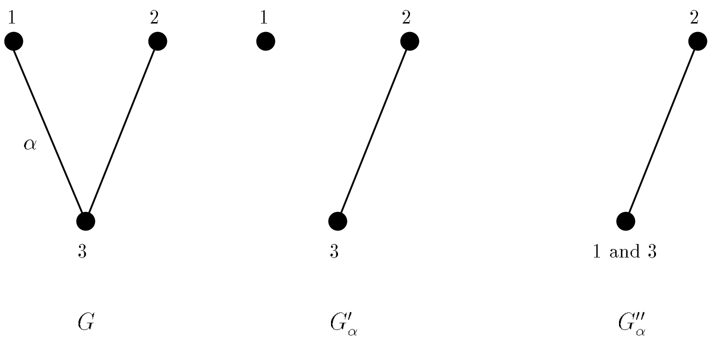


Figure 3.60: An application of the Fundamental Reduction Theorem.

as is easy to verify. Also, since $K_1 = I_1$, (3.15) and (3.17) give us

$$P(G'_\alpha, x) = P(I_1, x)[P(I_2, x) - P(I_1, x)]. \tag{3.18}$$

Finally, plugging (3.17) and (3.18) into (3.13) gives us

$$P(G, x) = P(I_1, x)[P(I_2, x) - P(I_1, x)] - [P(I_2, x) - P(I_1, x)]. \tag{3.19}$$

We have reduced $P(G, x)$ to an expression (3.19) that requires only knowledge of the polynomials $P(I_k, x)$ for different values of k.

As a second example, let us use the Fundamental Reduction Theorem to calculate $P(K_3, x)$. Note from Figure 3.61 that if $H = K_3$, then H'_α is the graph G of Figure 3.60 and H''_α is K_2. Thus,

$$\begin{aligned} P(K_3, x) &= P(G, x) - P(K_2, x) \\ &= x(x-1)^2 - x(x-1), \end{aligned}$$

where the first expression arises from our previous computation, and the second from the formula for $P(K_2, x)$. Simplifying, we obtain

$$\begin{aligned} P(K_3, x) &= x(x-1)[(x-1) - 1] \\ &= x(x-1)(x-2) \\ &= x^{(3)}, \end{aligned}$$

which agrees with Theorem 3.9.

The reader should note that each application of the Fundamental Reduction Theorem reduces the number of edges in each graph that remains. Hence, by repeated uses of the Fundamental Reduction Theorem, we must eventually end up with graphs with no edges, namely graphs of the form I_k. We illustrated this point with our first example. In any case, this shows that Theorem 3.8 must indeed hold, that is, that $P(G, x)$ is always a polynomial in x. For $P(I_k, x) = x^k$. Hence, we eventually reduce $P(G, x)$ to an expression that is a sum, difference, or product of terms each of which is of the form x^k, for some k. (The proof may be formalized by arguing by induction on the number of edges in the graph.)

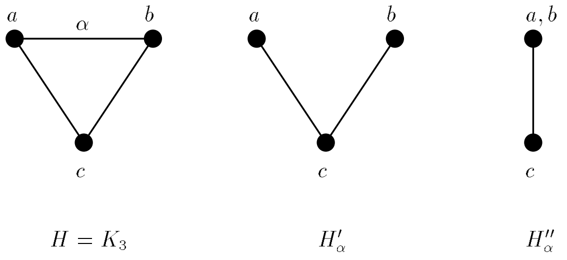


Figure 3.61: A second application of the Fundamental Reduction Theorem.

Figure 3.62 gives one final illustration of the Fundamental Reduction Theorem. In that figure, we simply draw a graph to stand for the chromatic polynomial of the graph. The edge α is indicated at each step.

3.4.3 Properties of Chromatic Polynomials[21]

We have already pointed out that one of Birkhoff's hopes in introducing chromatic polynomials was to be able to tell what polynomials were chromatic and then to study the roots of those polynomials that were chromatic. In this section we study the properties of the chromatic polynomials. We shall discover that we can learn a great deal about a graph by knowing its chromatic polynomial. Exercises 13–16 outline the proofs of the theorems stated here and present further properties of chromatic polynomials.

The first theorem summarizes some elementary properties of chromatic polynomials. These can be checked for all the examples of Sections 3.4.1 and 3.4.2.

Theorem 3.12 (Read [1968]) Suppose that G is a graph of n vertices and

$$P(G,x) = a_p x^p + a_{p-1}x^{p-1} + \cdots + a_1 x + a_0.$$

Then:

(a) The degree of $P(G,x)$ is n, that is, $p = n$.

(b) The coefficient of x^n is 1, that is, $a_n = 1$.

(c) The constant term is 0, that is, $a_0 = 0$.

(d) Either $P(G,x) = x^n$ or the sum of the coefficients in $P(G,x)$ is 0.

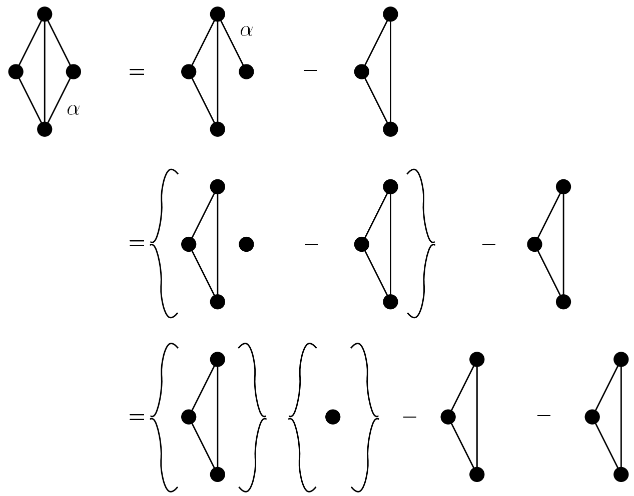
Theorem 3.13 (Whitney [1932]) $P(G,x)$ is the sum of consecutive powers of x and the coefficients of these powers alternate in sign. That is, for some I,

$$P(G,x) = x^n - \alpha_{n-1}x^{n-1} + \alpha_{n-2}x^{n-2} \mp \cdots \pm \alpha_0, \tag{3.20}$$

with $\alpha_i > 0$ for $i \geq I$ and $\alpha_i = 0$ for $i < I$.

---

[20] This subsection may be omitted.



$$= P(K_3, x)P(K_1, x) - P(K_3, x) - P(K_3, x)$$
$$= x(x-1)(x-2)x - x(x-1)(x-2) - x(x-1)(x-2)$$
$$= x(x-1)(x-2)^2$$

Figure 3.62: Another application of the Fundamental Reduction Theorem.

Theorem 3.14 (Read [1968]) In $P(G, x)$, the absolute value of the coefficient of x^{n-1} is the number of edges of G.

Unfortunately, the properties of chromatic polynomials that we have listed in Theorems 3.12 and 3.13 do not characterize chromatic polynomials. There are polynomials $P(x)$ that satisfy all these conditions but that are not chromatic polynomials of any graph. For instance, consider

$$P(x) = x^4 - 4x^3 + 3x^2.$$

Note that the coefficient of x^n is 1, the constant term is 0, the sum of the coefficients is 0, and the coefficients alternate in sign until, from the coefficient of x^1 and on, they are 0. However, $P(x)$ is not a chromatic polynomial of any graph. If it were, the number of vertices of the graph would have to be 4, by part (a) of Theorem 3.12. The number of edges would also have to be 4, by Theorem 3.14. No graph with four vertices and four edges has this polynomial as its chromatic polynomial, as is easy to check. Or see Lehmer [1985], who found the chromatic polynomials for all graphs on 6 or fewer vertices.

For further results on chromatic polynomials, see, for example, Jensen and Toft [1995], Liu [1972], or Read [1968].

EXERCISES FOR SECTION 3.4

1. Find the chromatic polynomial of each graph in Figure 3.63.

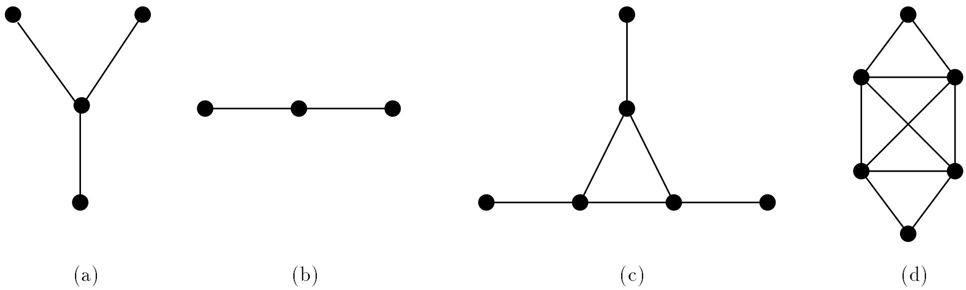


Figure 3.63: Graphs for exercises of Section 3.4.

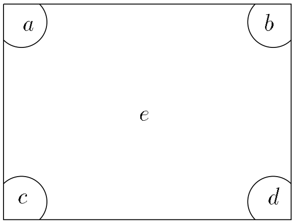


Figure 3.64: Map for exercises of Section 3.4.

2. For each graph in Figure 3.63, find the number of ways to color the graph in at most three colors.

3. The chromatic polynomial $P(M,x)$ of a map M is the number of ways to color M in x or fewer colors. Find $P(M,x)$ for the map of Figure 3.64.

4. Let L_n be the graph consisting of a simple chain of n vertices. Find a formula for $P(L_n,x)$.

5. For each of the graphs of Figure 3.65, find the chromatic polynomial using reduction theorems. (You may reduce to graphs with previously known chromatic polynomials.)

6. If G is the graph of Figure 3.65(a), express $P(G,x)$ in terms of polynomials $P(I_k,x)$ for various k.

7. If L_n is as defined in Exercise 4, what is the relation among $P(Z_n,x), P(Z_{n-1},x)$, and $P(L_n,x)$?

8. Use reduction theorems to find the chromatic polynomial of the map of Figure 3.66 (see Exercise 3). You may use the result of Exercise 4.

9. Let $N(G,x)$ be the number of ways of coloring G in exactly x colors. Find $N(G,x)$ for each of the following graphs G and the given values of x.

   (a) $Z_5, x=4$ (b) $K_5, x=6$ (c) $L_5, x=3$

   (d) Find an expression for $P(G,x)$ in terms of the numbers $N(G,r)$ for $r \leq x$.

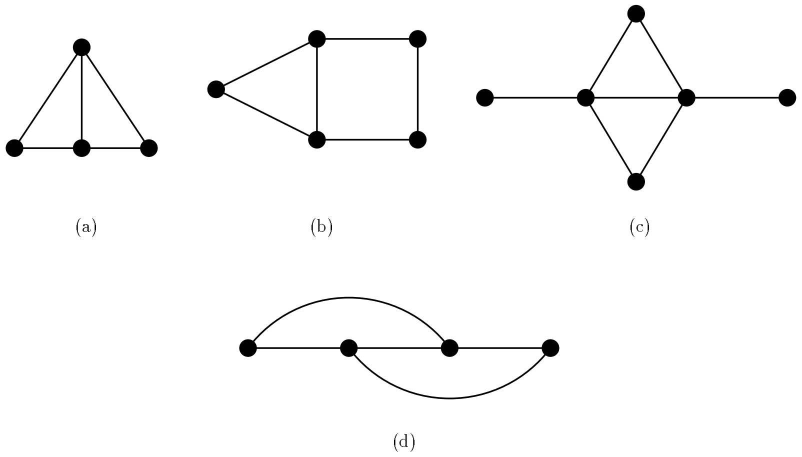


Figure 3.65: Graphs for exercises of Section 3.4.

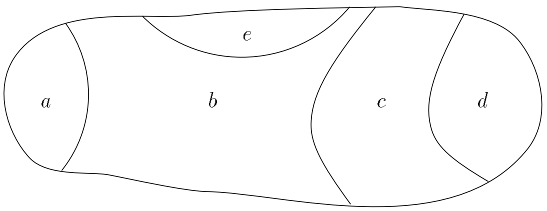


Figure 3.66: Map for exercises of Section 3.4.

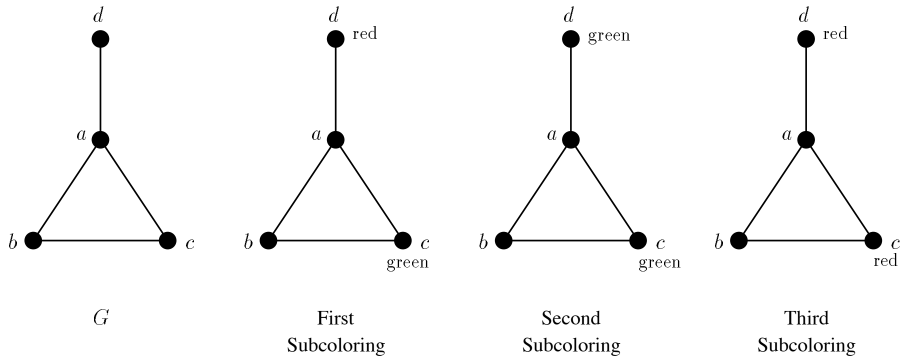


Figure 3.67: A graph G and three subcolorings of G.

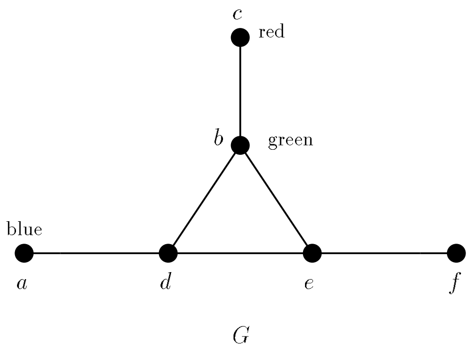


Figure 3.68: A graph and a subcoloring.

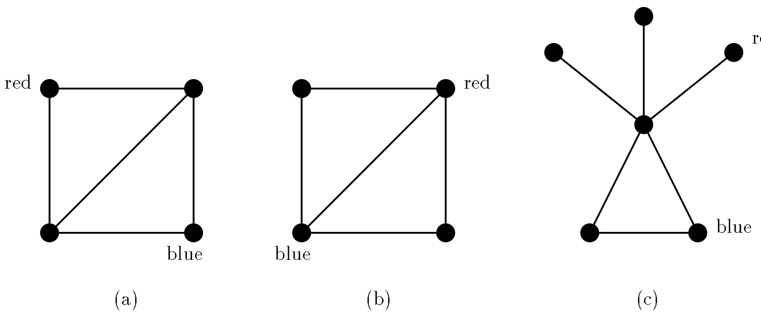


Figure 3.69: Graphs and subcolorings.

10. If we have a coloring of some vertices of G, we call this a *subcoloring* of G. A coloring of all the vertices of G that agrees with a subcoloring of some of the vertices of G is called an *extension* of the subcoloring. Figure 3.67 shows a graph G and three subcolorings of G. If there is just one more color available, say blue, then the first subcoloring can be extended to G in just one way, namely by coloring vertex a blue and vertex b red. However, the second subcoloring can be extended to a subcoloring of G in two ways, by coloring a blue and b red, or by coloring a red and b blue.

    (a) How many extensions are there of the third subcoloring shown in Figure 3.67?

    (b) Consider the graph G of Figure 3.68 and the subcoloring of the vertices a, b, and c shown in that figure. How many extensions are there of this subcoloring to all of G if only the colors green, red, blue, and brown are available?

    (c) Consider the graphs of Figure 3.69 and the subcolorings shown in that figure. Find the number of extensions of each subcoloring to a coloring of the whole graph in three or fewer colors, if red, blue, and green are the three colors available.

11. Repeat Exercise 10(c), finding the number of extensions using at most x colors, $x \geq 3$.

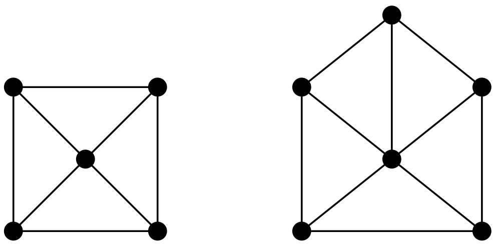
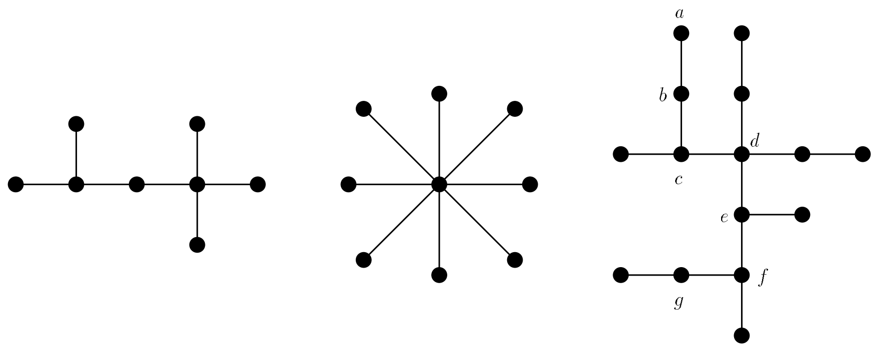


Figure 3.70: The wheels W_4 and W_5.

12. Show that the following could not be chromatic polynomials.
    (a) $P(x) = x^8 - 1$
    (b) $P(x) = x^5 - x^3 + 2x$
    (c) $P(x) = 2x^3 - 3x^2$
    (d) $P(x) = x^3 + x^2 + x$
    (e) $P(x) = x^3 - x^2 + x$
    (f) $P(x) = x^4 - 3x^3 + 3x^2$
    (g) $P(x) = x^9 + x^8 - x^7 - x^6$
13. Prove parts (c) and (d) of Theorem 3.12.
14. Prove parts (a) and (b) of Theorem 3.12 together, by induction on the number e of edges and by use of the Fundamental Reduction Theorem.
15. Prove Theorem 3.13 by induction on the number e of edges and by use of the Fundamental Reduction Theorem.
16. Prove Theorem 3.14 from Theorem 3.13, by induction on the number e of edges.
17. Prove that $P(G, q) \leq q(q-1)^{n-1}$ for any positive integer q, if G is connected with n vertices.
18. Prove that $P(G, \lambda) \neq 0$ for any $\lambda < 0$.
19. (a) If G has k connected components, show that the smallest i such that x^i has a nonzero coefficient in $P(G, x)$ is at least k.
    (b) Is this smallest i necessarily equal to k? Why?
    (c) Prove that if $P(G, x) = x(x-1)^{n-1}$, then G is connected.
20. Suppose that W_n is the *wheel* of $n+1$ vertices, that is, the graph obtained from Z_n by adding one vertex and joining it to all vertices of Z_n. W_4 and W_5 are shown in Figure 3.70. Find $P(W_n, x)$. You may leave your answer in terms of $P(Z_n, x)$.
21. If Z_n is the circuit of length n:
    (a) Show that for $n \geq 3, (-1)^n[P(Z_n, x) - (x-1)^n]$ is constant, independent of n.
    (b) Solve for $P(Z_n, x)$ by evaluating the constant in part (a).
22. Suppose that H is a clique of G and that we have two different subcolorings of H in at most x colors. Show that the number of extensions to a coloring of G in at most x colors is the same for each subcoloring.



(a) (b) (c)

Figure 3.71: Some trees.

23. The following is another reduction theorem. Suppose that H and K are generated subgraphs of G, with $V(G) = V(H) \cup V(K)$ and $E(G) = E(H) \cup E(K)$, and that $V(H) \cap V(K)$ is a clique of G of p vertices. Then

$$P(G,x) = \frac{P(H,x)P(K,x)}{x^{(p)}}.$$

   (a) Illustrate this result on the graph G of Figure 3.68 if H is the subgraph generated by $\{a,d,e,f\}$ and K the subgraph generated by $\{c,b,d,e\}$. (Disregard the subcoloring.)

   (b) Make use of the result of Exercise 22 to prove the theorem.

24. If the chromatic polynomial $P(K_n,x)$ is expanded out, the coefficient of x^k is denoted $s(n,k)$ and called a *Stirling number of the first kind.* Exercises 24–26 will explore these numbers. Find:

   (a) $s(n,0)$ (b) $s(n,n)$ (c) $s(n,1)$ (d) $s(n,n-1)$

25. Show that

$$|s(n,k)| = (n-1)|s(n-1,k)| + |s(n-1,k-1)|.$$

26. Use the result in Exercise 25 to describe how to compute Stirling numbers of the first kind by a method similar to Pascal's triangle and apply your ideas to compute $s(6,3)$.

3.5 TREES

3.5.1 Definition of a Tree and Examples

In this section and the next we consider one of the most useful concepts in graph theory, that of a tree. A *tree* is a graph T that is connected and has no circuits. Figure 3.71 shows some trees.

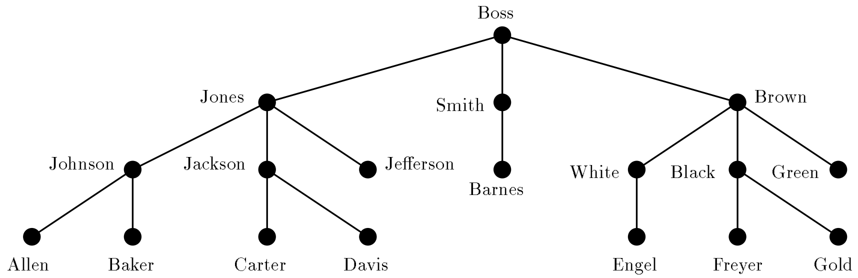


Figure 3.72: A telephone chain.

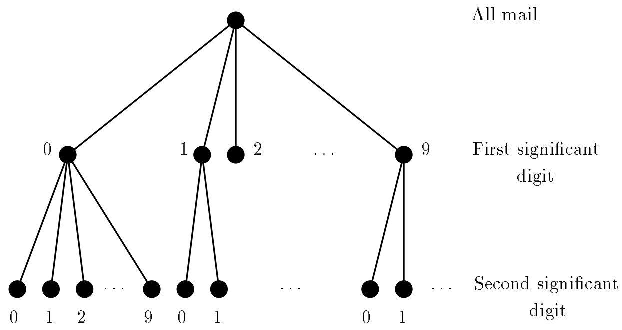


Figure 3.73: Part of the sort tree for sorting mail by ZIP code.

Example 3.25 Telephone Trees Many companies and other organizations have prearranged telephone chains to notify their employees in case of an emergency, such as a snowstorm that will shut down the company. In such a telephone chain, a person in charge makes a decision (e.g., to close because of snow) and calls several designated people, who each call several designated people, who each call several designated people, and so on. We let the people in the company be vertices of a graph and include an edge from a to b if a calls b (we include an undirected edge even though calling is not symmetric). The resulting graph is a tree, such as that shown in Figure 3.72. ■

Example 3.26 Sorting Mail[22] Mail intended for delivery in the United States carries on it, if the sender follows Postal Service instructions, a ZIP code consisting of a certain number of decimal digits. Mail arriving at a post office is first sorted into 10 piles by the most significant digit. Then each pile is divided into 10 piles by the next most significant digit. And so on. The sorting procedure can be summarized by a tree, part of which is shown in Figure 3.73. To give a simpler example, suppose that we sort mail within a large organization by giving a mail code not unlike a ZIP code, but consisting of only three digits, each being 0 or 1. Then the sort tree is shown in Figure 3.74. ■

[22] This example is based on Deo [1974].

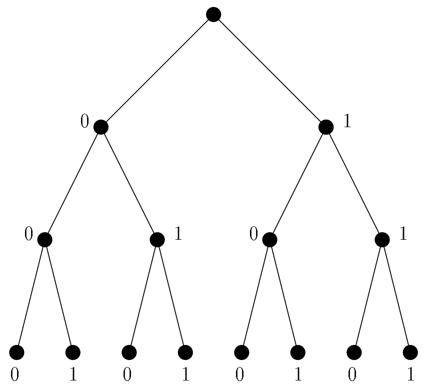


Figure 3.74: The sort tree for mail coded by three binary digits.

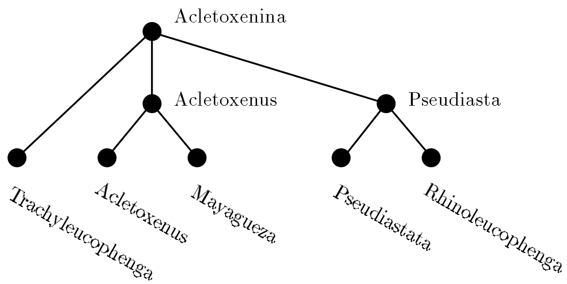


Figure 3.75: A (partial) phylogenetic tree of genera in the Drosophilidae (Diptera). (Based on data in Grimaldi [1990].)

Example 3.27 Phylogenetic Trees A widely held view in modern biology is that all existing organisms derive from common ancestors and that new species arise when a given population splits into two or more populations by some process, such as mutation. Trees are often used to model the evolutionary process in which species evolve into new species. The vertices of the tree are the species and an edge leads from a species to an immediate descendant. Figure 3.75 shows an evolutionary tree. Evolutionary trees also arise in subjects other than biology. For example, in linguistics, we study the evolution of words over time, through "mutations" in spelling. A common problem in phylogeny, the scientific discipline that studies evolution, is to reconstruct an evolutionary or *phylogenetic tree* from information about presently existing species. We have more to say about this in Section 3.5.7. ■

In this section and the next we consider a variety of other applications of trees, in particular emphasizing applications to organic chemistry, phylogenetic tree reconstruction, and to searching and sorting problems in computer science.

3.5.2 Properties of Trees

A fundamental property of trees is obtained by noting the relationship between the number of vertices and the number of edges of a tree.

Theorem 3.15 If T is a tree of n vertices and e edges, then $n = e + 1$.

Theorem 3.15 is illustrated by any of the trees we have drawn. It will be proved in Section 3.5.3.

Note that the property $n = e + 1$ does not characterize trees. There are graphs that have $n = e + 1$ and are not trees (Exercise 6). However, we have the following result, which will be proved in Section 3.5.5.

Theorem 3.16 Suppose that G is a graph of n vertices and e edges. Then G is a tree if and only if G is connected and $n = e + 1$.

We next note an interesting consequence of Theorem 3.15. The result will be used in counting the number of trees.

Theorem 3.17 If T is a tree with more than one vertex, there are at least two vertices of degree 1.

Proof. Since T is connected, every vertex must have degree ≥ 1. (Why?) Now by Theorems 3.15 and 3.1, $\sum \deg(u) = 2e = 2n - 2$. If $n - 1$ vertices had degree ≥ 2, the sum of the degrees would have to be at least $2(n-1) + 1 = 2n - 1$, which is greater than $2n - 2$. Thus, no more than $n - 2$ vertices can have degree ≥ 2. Q.E.D.

3.5.3 Proof of Theorem 3.15[23]

Theorem 3.18 In a tree T, if x and y are any two vertices, then there is one and only one simple chain joining x and y.

Proof. We know there is a chain between x and y, since T is connected. Recall that a simple chain has no repeated vertices. A shortest chain between x and y must be a simple chain. For if $x, u, \ldots, v, \ldots, v, \ldots, w, y$ is a shortest chain between x and y with a repeated vertex v, we can skip the part of the chain between repetitions of v, thus obtaining a shorter chain from x to y. Hence, there is a simple chain joining x and y. Suppose next that $C_1 = x, x_1, x_2, \ldots, x_k, y$ is a shortest simple chain joining x and y. We show that there can be no other simple chain joining x and y. Suppose that C_2 is such a chain. Let x_{p+1} be the first vertex of C_1 on which C_1 and C_2 differ, and let x_q be the next vertex of C_1 following x_p on which C_1 and C_2 agree. Then we obtain a circuit by following C_1 from x_p to x_q and C_2 back from x_q to x_p, which contradicts the fact that T is a tree. Q.E.D.

[23] This subsection may be omitted.

To illustrate this theorem, we note that in tree (c) of Figure 3.71, the unique simple chain joining vertices a and g is given by a, b, c, d, e, f, g. Now we return to the proof of Theorem 3.15.

Proof of Theorem 3.15. The proof is by induction on n. If $n = 1$, the result is trivial. The only tree with one vertex has no edges. Now suppose that the result is true for all trees of fewer than n vertices and that tree T has n vertices. Pick any edge $\{u, v\}$ in T. By Theorem 3.18, u, v is the only simple chain between u and v. If we take edge $\{u, v\}$ away from G (but leave vertices u and v), we get a new graph H. Now in H, there can be no chain between u and v. For if there is, it is easy to find a simple chain between u and v. But this is also a simple chain in G, and G has only one simple chain between u and v.

Now since H has no chain between u and v, H is disconnected. It is not difficult to show (Exercise 22) that H has exactly two connected components. Call these H_1 and H_2. Since each of these is connected and can have no circuits (why?), each is a tree. Moreover, each has fewer vertices than G. By the inductive hypothesis, if n_i and e_i are the number of vertices and edges of H_i, we have $n_1 = e_1 + 1$ and $n_2 = e_2 + 1$. Now $n = n_1 + n_2$ and $e = e_1 + e_2 + 1$ (add edge $\{u, v\}$). We conclude that

$$n = n_1 + n_2 = (e_1 + 1) + (e_2 + 1) = (e_1 + e_2 + 1) + 1 = e + 1. \qquad \text{Q.E.D.}$$

3.5.4 Spanning Trees[24]

Suppose that $G = (V, E)$ is a graph and $H = (W, F)$ is a subgraph. We say H is a *spanning subgraph* if $W = V$. A spanning subgraph that is a tree is called a *spanning tree*. For example, in the graph G of Figure 3.76, H and K as shown in the figure are spanning subgraphs because they have the same vertices as G. K is a spanning tree. Spanning trees have a wide variety of applications in combinatorics, which we shall investigate shortly. Analysis of an electrical network reduces to finding all spanning trees of the corresponding graph (see Deo [1974]). Spanning trees can also be used, through the program digraph (see Example 3.6), to estimate program running time (Deo [1974], p. 442]). They arise in connection with seriation problems in political science and archaeology (Roberts [1979], Wilkinson [1971]). They also form the basis for a large number of algorithms in network flows and the solution of minimal cost problems in operations research (Chapter 13). Graham and Hell [1985] mention applications of spanning trees to design of computer and communication networks, power networks, leased-line telephone networks, wiring connections, links in a transportation network, piping in a flow network, network reliability, picture processing, automatic speech recognition, clustering and classification problems, and so on. Their paper also gives many references to the literature of spanning trees and their applications.

We now give several applications of spanning trees in more detail.

[24] This subsection may be omitted until just before Section 11.1 or Section 13.1.

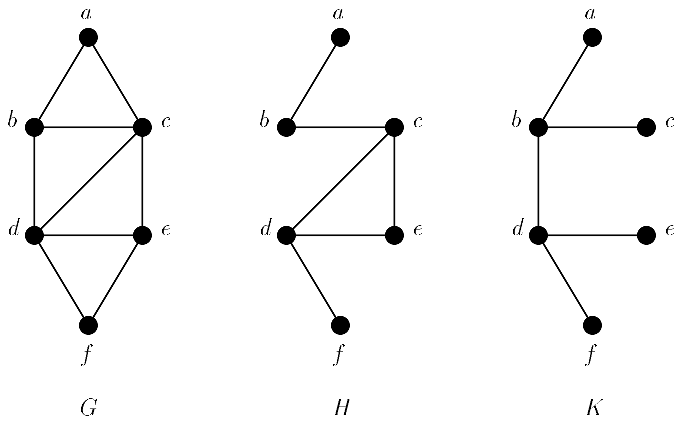


Figure 3.76: H and K are spanning subgraphs of G, and K is also a spanning tree.

Example 3.28 Highway Construction Suppose that a number of towns spring up in a remote region where there are no good highways. It is desired to build enough highways so that it is possible to travel from any of the towns to any other of the towns by highway either directly or by going through another town. Let the towns be vertices of a graph G and join each pair of towns by an edge. We wish to choose a subset F of the edges of G representing highways to be built, so that if we use the vertices of G and the edges of F, we have a connected graph. Thus, we seek a connected, spanning subgraph of G. If we wish to minimize costs, then certainly the highways we choose to build correspond to a connected spanning subgraph H of G with the property that removal of any edge leads to a spanning subgraph that is no longer connected. But in any connected spanning subgraph H of G, if there is a circuit, removal of any edge of the circuit cannot disconnect H. (Why?) Thus, H is connected and has no circuits; that is, H is a spanning tree. If, in addition, each edge of G has a weight or number on it, representing the cost of building the corresponding highway, then we wish to find a spanning tree F which is *minimum* in the sense that the sum of the weights on its edges is no larger than that on any other spanning tree. We study algorithms for finding minimum spanning trees in Section 13.1. We shall sometimes also be interested in finding maximum spanning trees. The procedure for finding them is analogous to the procedure for finding minimum spanning trees. It should be noted that a similar application of minimum spanning trees arises if the links are gas pipelines, electrical wire connections, railroads, sewer lines, and so on. ■

The approach taken in Example 3.28 assumes that we do not allow highways to link up at points other than the towns in question. This is implicit in the assumption that we want to go either directly between two towns or link them up through another town. If we allowed highways to link up at points other than the towns in question, additional savings could possibly be realized when minimizing costs. These cost-saving "additional" points are called *Steiner points*. For more information on Steiner points, see Bern and Graham [1989] or Cieslik [1998].

Example 3.29 Telephone Lines In a remote region, isolated villages are linked by road, but there is not yet any telephone service. We wish to put in telephone lines so that every pair of villages is linked by a (not necessarily direct) telephone connection. It is cheapest to put the lines in along existing roads. Along which stretches of road should we put telephone lines so as to make sure that every pair of villages is linked and the total number of miles of telephone line (which is probably proportional to the total cost of installation) is minimized? Again we seek a minimum spanning tree. (Why?) ■

Example 3.30 Computer Hardware (Ahuja, Magnanti, and Orlin [1993]) A digital computer has a variety of components to be connected by high-frequency circuitry. To reduce capacitance and delay line effects, it is important to minimize the length of wires between components. We need to connect all components and so seek a minimum spanning tree. ■

Example 3.31 Data Storage (Ahuja, Magnanti, and Orlin [1993], Kang, *et al.* [1977]) In many applied problems, data are stored in a two-dimensional array. When the rows of the array have many similar entries and differ only in a few places, we can save memory by storing only one row completely and then storing the differences of the other rows from this *reference row*. Build a complete graph G with the weight on the edge between rows i and j being the number of changes in entries required to switch from row i to row j, or vice versa. We can minimize data storage by choosing as the reference row that row with the minimum amount of data to store and then finding a minimum spanning tree in the graph G. Why? ■

Example 3.32 Measuring the Homogeneity of Bimetallic Objects[25] Minimum spanning trees have found applications at the U.S. National Bureau of Standards and elsewhere, in determining to what extent a bimetallic object is homogeneous in composition. Given such an object, build a graph by taking as vertices a set of sample points in the material. Measure the composition at each sample point and join physically adjacent sample points by an edge. Place on the edge $\{x, y\}$ a weight equal to the physical distance between sample points x and y multiplied by a homogeneity factor between 0 and 1. This factor is 0 if the composition of the two points is exactly alike and 1 if it is dramatically opposite, and otherwise is a number in between. In the resulting graph, find a minimum spanning tree. The sum of the weights in this tree is 0 if all the sample points have exactly the same composition. A high value says that the material is quite inhomogeneous. This statistic, the sum of the weights of the edges in a minimum spanning tree, is a very promising statistic in measuring homogeneity. According to Goldman [1981], it will probably come into standard use over an (appropriately extended) period of time. ■

[25] From Filliben, Kafadar, and Shier [1983], Goldman [1981], and Shier [1982].

We now present one main result about spanning trees.

Theorem 3.19 A graph G is connected if and only if it has a spanning tree.

Proof. Suppose that G is connected. Then there is a connected spanning subgraph H of G with a minimum number of edges. Now H can have no circuits. For if C is a circuit of H, removal of any edge of C (without removing the corresponding vertices) leaves a spanning subgraph of G that is still connected and has one less edge than H. This is impossible by choice of H. Thus, H has no circuits. Also, by choice, H is connected. Thus, H is a spanning tree.

Conversely, if G has a spanning tree, it is clearly connected. Q.E.D.

The proof of Theorem 3.19 can be reworded as follows. Suppose that we start with a connected graph G. If G has no circuits, it is already a tree. If it has a circuit, remove any edge of the circuit and a connected graph remains. If there is still a circuit, remove any edge of the circuit and a connected graph remains, and so on. Note that Theorem 3.19 gives us a method for determining if a graph G is connected: Simply test if G has a spanning tree. Algorithms for doing this are discussed in Section 13.1.

3.5.5 Proof of Theorem 3.16 and a Related Result[26]

We now present a proof of Theorem 3.16 and then give a related theorem.

Proof of Theorem 3.16. We have already shown, in Theorem 3.15, one direction of this equivalence. To prove the other direction, suppose that G is connected and $n = e + 1$. By Theorem 3.19, G has a spanning tree T. Then T and G have the same number of vertices. By Theorem 3.15, T has $n - 1$ edges. Since G also has $n - 1$ edges, and since all edges of T are edges of $G, T = G$. Q.E.D.

We also have another result, similar to Theorem 3.16, which will be needed in Chapter 13.

Theorem 3.20 Suppose that G is a graph with n vertices and e edges. Then G is a tree if and only if G has no circuits and $n = e + 1$.

Proof. Again it remains to prove one direction of this theorem. Suppose that G has no circuits and $n = e + 1$. If G is not connected, let $K_1, K_2, \ldots, K_p$ be its connected components, $p > 1$. Let K_i have n_i vertices. Then K_i is connected and has no circuits, so K_i is a tree. Thus, by Theorem 3.15, K_i has $n_i - 1$ edges. We conclude that the number of edges of G is given by

$$(n_1 - 1) + (n_2 - 1) + \cdots + (n_p - 1) = \sum n_i - p = n - p.$$

But since $p > 1, n - p < n - 1$, so $n \neq e + 1$. This is a contradiction. Q.E.D.

[26] This subsection may be omitted.

3.5.6 Chemical Bonds and the Number of Trees

In 1857, Arthur Cayley discovered trees while he was trying to enumerate the isomers of the saturated hydrocarbons, chemical compounds of the form C_kH_{2k+2}. This work was the forerunner of a large amount of graph-theoretical work in chemistry and biochemistry. We present Cayley's approach here.

Combinatorial methods are used in chemistry to represent molecules, clusters, and reaction paths. Coding theory (to be studied in Chapter 9) is basic in systematizing the enormous amount of chemical data available and in enumerating molecules, isomers, and families having various properties. Graph theory is used to understand the structure of chemical compounds, proteins, and so on, and symmetries in chemical structures, and it is also useful in developing systematic approaches to chemical nomenclature. All of these techniques are used increasingly by industry in the rapidly expanding fields of computer-assisted molecular design and combinatorial chemistry. For good references on graph theory and chemistry, see Balaban [1976], Hansen, Fowler, and Zheng [2000], McKee and Beineke [1997], and Rouvray and Balaban [1979]. See also the September 1992 issue of *The Journal of Chemical Education*, which featured graph theory in chemistry. We give other applications of combinatorics to chemistry in Section 6.4 and Chapter 8.

The isomers C_kH_{2k+2} can be represented by representing a carbon atom with the letter C and a hydrogen atom with the letter H, and linking two atoms if they are bonded in the given compound. For example, methane and ethane are shown in Figure 3.77. We can replace these diagrams with graphs by replacing each letter with a vertex, as shown in Figure 3.78. We call these graphs *bond graphs*. Note that given a bond graph of a saturated hydrocarbon, the vertices can be relabeled with letters C and H in an unambiguous fashion: A vertex is labeled C if it is bonded to four other vertices (carbon has chemical valence 4), and H if it is bonded to one other vertex (hydrogen has valence 1). Every vertex of one of our bond graphs is bonded to either one or to four other vertices. The bond graphs of some other saturated hydrocarbons are shown in Figure 3.79.

The graphs of Figures 3.78 and 3.79 are all trees. We shall show that this is no accident, that is, that the bond graph of every saturated hydrocarbon is a tree.

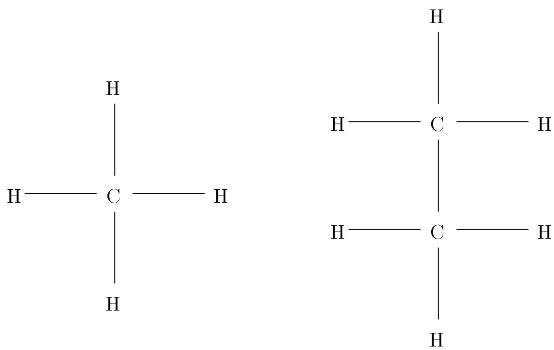
Recall that the degree of a vertex in a graph is the number of its neighbors, and by Theorem 3.1, the sum of the degrees of the vertices is twice the number of edges. Now in the bond graph for C_kH_{2k+2}, there are

$$k + 2k + 2 = 3k + 2$$

vertices. Moreover, each carbon vertex has degree 4 and each hydrogen vertex has degree 1. Thus, the sum of the degrees is

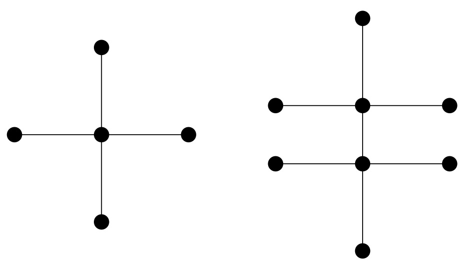
$$4k + 1(2k + 2) = 6k + 2.$$

The number of edges is half this number, or $3k + 1$. Thus, the number of edges is exactly one less than the number of vertices. Since the bond graph for a chemical compound C_kH_{2k+2} must be connected, it follows by Theorem 3.16 that it is a tree.



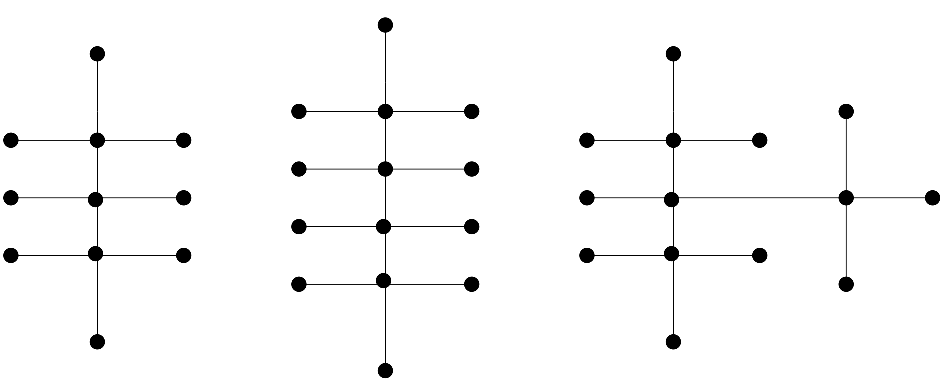
Methane Ethane

Figure 3.77: Two saturated hydrocarbons.



Methane Ethane

Figure 3.78: Bond graphs for the saturated hydrocarbons of Figure 3.77.



Propane Butane Isobutane

Figure 3.79: More bond graphs.

Now Cayley abstracted the problem of enumerating all possible saturated hydrocarbons to the problem of enumerating all trees in which every vertex has degree 1 or 4. He found it easier to begin by enumerating all trees. In the process, he discovered abstractly the bond graphs of some saturated hydrocarbons which were previously unknown, and predicted their existence. They were later discovered.

It clearly makes a big difference in counting the number of distinct graphs of a certain type whether or not we consider these graphs as labeled (see the discussion in Section 3.1.3). In particular, Cayley discovered that for $n \geq 2$, there were n^{n-2} distinct labeled trees with n vertices. We shall present one proof of this result here. For a survey of other proofs, see Moon [1967] (see also Harary and Palmer [1973], Shor [1995], and Takács [1990]).

Theorem 3.21 (Cayley [1889]) If $n \geq 2$, there are n^{n-2} distinct labeled trees of n vertices.

Let $N(d_1, d_2, \ldots, d_n)$ count the number of labeled trees with n vertices with the vertex labeled i having degree $d_i + 1$. We first note the following result, which is proved in Exercise 34.

Theorem 3.22 If $n \geq 2$ and all d_i are nonnegative integers, then

$$N(d_1, d_2, \ldots, d_n) = \begin{cases} 0 & \text{if } \sum_{i=1}^{n} d_i \neq n-2 \\ \\ C(n-2; d_1, d_2, \ldots, d_n) & \text{if } \sum_{i=1}^{n} d_i = n-2. \end{cases} \tag{3.21}$$

In this theorem,

$$C(n-2; d_1, d_2, \ldots, d_n) = \frac{(n-2)!}{d_1! d_2! \cdots d_n!}$$

is the multinomial coefficient studied in Section 2.11.

To illustrate Theorem 3.22, note that

$$N(1,3,0,0,0,0) = C(4;1,3,0,0,0,0) = \frac{4!}{1!3!0!0!0!0!} = 4.$$

There are four labeled trees of six vertices with vertex 1 having degree 2, vertex 2 having degree 4, and the remaining vertices having degree 1. Figure 3.80 shows the four trees.

Cayley's Theorem now follows as a corollary of Theorem 3.22. For the number $T(n)$ of labeled trees of n vertices is given by

$$T(n) = \sum \left\{ N(d_1, d_2, \ldots, d_n) : d_i \geq 0, \sum_{i=1}^{n} d_i = n-2 \right\},$$

which is the same as

$$T(n) = \sum \left\{ C(n-2; d_1, d_2, \ldots, d_n) : d_i \geq 0, \sum_{i=1}^{n} d_i = n-2 \right\}.$$

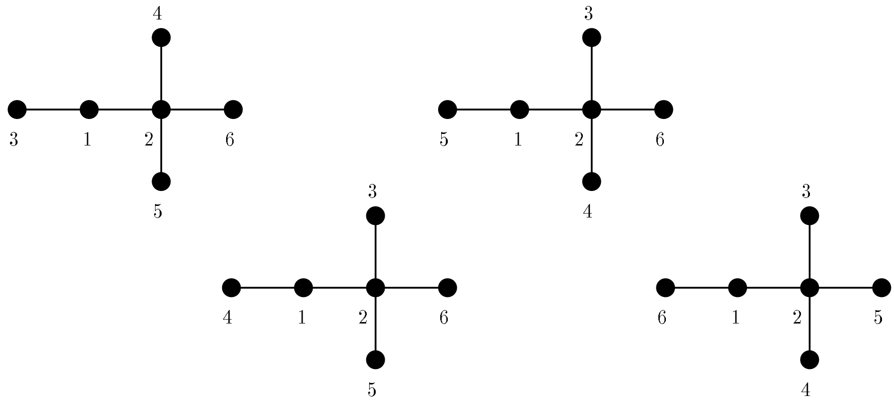


Figure 3.80: The four labeled trees with degrees 1+1, 3+1, 0+1, 0+1, 0+1, 0+1, respectively.

Now it is easy to see that the binomial expansion (Theorem 2.7) generalizes to the following *multinomial expansion*: If p is a positive integer, then

$$(a_1 + a_2 + \cdots + a_k)^p = \sum \left\{ C(p; d_1, d_2, \ldots, d_k) a_1^{d_1} a_2^{d_2} \cdots a_k^{d_k} : d_i \geq 0, \ \sum_{i=1}^{k} d_i = p \right\}.$$

Thus, taking $k = n$ and $p = n - 2$ and $a_1 = a_2 = \cdots = a_k = 1$, we have

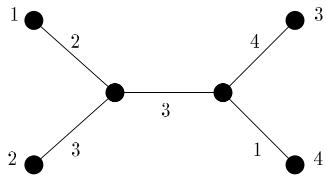
$$T(n) = (1 + 1 + \cdots + 1)^{n-2} = n^{n-2}.$$

This is Cayley's Theorem.

3.5.7 Phylogenetic Tree Reconstruction

In Example 3.27, we introduced the concept of an evolutionary or phylogenetic tree. A more general phylogenetic tree has weights of real numbers on its edges, representing the passage of time between two evolutionary events. An important problem in biology (and in other areas such as linguistics) is to reconstruct a phylogenetic tree from some information about present species. A general introduction to this topic from the biological point of view can be found in Fitch [1999] and an introduction from a combinatorial point of view can be found in Gusfield [1997]. Common methods for phylogenetic tree reconstruction start with the DNA sequences of presently existing species. *Sequence-based methods* use these sequences directly to reconstruct the tree. *Distance-based methods*, by contrast, first compute distances between all pairs of sequences and then use these distances to reconstruct the tree. More particularly, they start with n species and an $n \times n$ symmetric matrix D whose i, j entry $D(i, j)$ gives the calculated distance between the two species i and j. They then seek a tree in which the given species are the *leaves*, i.e., the vertices with only one neighbor, and $D(i, j)$ is equal to or closely approximated by the distance $d(i, j)$

$$D = \begin{pmatrix} 0 & 5 & 9 & 6 \\ 5 & 0 & 10 & 7 \\ 9 & 10 & 0 & 5 \\ 6 & 7 & 5 & 0 \end{pmatrix}$$



T

Figure 3.81: An additive distance matrix D and corresponding weighted tree T.

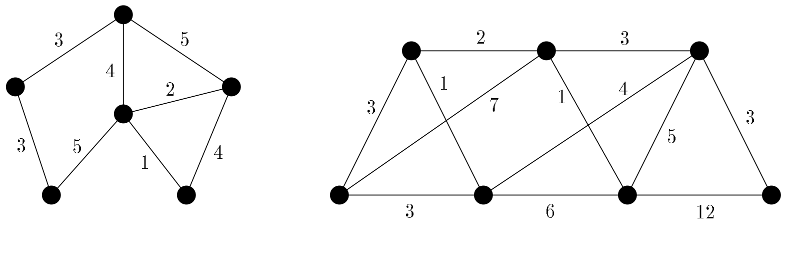
between i and j in the tree. Here, we measure distance $d(i,j)$ between vertices i and j in a tree with edge weights by summing up the weights in the unique simple chain from i to j in the tree. (Theorem 3.18 says that in any tree, there is always a unique simple chain between two given vertices.)

If D is a symmetric $n \times n$ matrix with 0's on the diagonal, we say that D is an *additive distance matrix* if we can find an edge-weighted tree with n leaves, corresponding to the given species, so that $d(i,j) = D(i,j)$. For instance, the matrix D of Figure 3.81 is an additive distance matrix and T of that figure is the corresponding weighted tree. For example, in the tree, the weights on the unique simple chain from 1 to 4 are 2, 3, and 1, so $d(1,4) = 6$, which corresponds to $D(1,4)$. How can we tell if a matrix is an additive distance matrix?

Theorem 3.23 (Buneman [1971]) A symmetric matrix D with 0's on the diagonal is an additive distance matrix if and only if for all i,j,k,l, of the three pairwise sums $D(i,j)+D(k,l), D(i,k)+D(j,l), D(i,l)+D(j,k)$, the largest two are identical.

To illustrate this theorem, we note that in Figure 3.81, for example, $D(1,2)+D(3,4) = 9, D(1,3)+D(2,4) = 15, D(1,4)+D(2,3) = 15$. There are good algorithms, in fact $O(n^2)$ algorithms, for reconstructing, if possible, an edge-weighted tree corresponding exactly to an additive distance matrix. See Gusfield [1997] for a variety of references.

If a matrix is not an additive distance matrix, we might wish to find a "closest" edge-weighted tree. This problem can be made precise in a variety of ways and almost all of them have been shown to be difficult to solve exactly (more precisely, NP-hard). One such problem is to find an additive distance matrix D' so that $\max_{i,j} |D(i,j) - D'(i,j)|$ is as small as possible. (This problem is NP-hard.) Agarwala, *et al.* [1996] found an algorithm that approximates an optimal solution, finding an edge-weighted tree so that $\max_{i,j} |D(i,j) - D'(i,j)|$ is guaranteed to be at most three times the optimal. This algorithm was improved by Cohen and Farach [1997]. Other algorithms, based on the theorem of Buneman (Theorem 3.23), are described in Erdős, *et al.* [1997, 1999].



(a) (b)

Figure 3.82: Graphs with weights on edges.

EXERCISES FOR SECTION 3.5

1. Draw the sort tree for sorting mail if the "ZIP code" consists of four digits, each being 0, 1, or 2.
2. Find all nonisomorphic trees of four vertices.
3. Find all nonisomorphic trees of five vertices.
4. Find:
   (a) The number of vertices in a tree of 10 edges
   (b) The number of edges in a tree of 10 vertices
5. Check Theorem 3.1 for every graph of Figure 3.76.
6. Give an example of a graph G with $n = e + 1$ but such that G is not a tree.
7. For each graph of Figure 3.23, either find a spanning tree or argue that none exists.
8. In each graph with weights on edges shown in Figure 3.82, find a spanning tree with minimum total weight.
9. A *forest* is a graph each of whose connected components is a tree. If a forest has n vertices and k components, how many edges does it have?
10. A simpler "proof" of uniqueness in Theorem 3.18 would be as follows. Suppose that C_1 and C_2 are two distinct simple chains joining x and y. Then C_1 followed by C_2 is a closed chain. But if a graph has a closed chain, it must have a circuit. Show that the latter statement is false.
11. In a connected graph with 15 edges, what is the maximum possible number of vertices?
12. In a connected graph with 25 vertices, what is the minimum possible number of edges?
13. What is the maximum number of vertices in a graph with 15 edges and three components?
14. Prove the converse of Theorem 3.18; that is, if G is any graph and any two vertices are joined by exactly one simple chain, then G is a tree.

15. Prove that if two nonadjacent vertices of a tree are joined by an edge, the resulting graph will have a circuit.
16. Prove that if any edge is deleted from a tree, the resulting graph will be disconnected.
17. If we have an (connected) electrical network with e elements (edges) and n nodes (vertices), what is the minimum number of elements we have to remove to eliminate all circuits in the network?
18. If G is a tree of n vertices, show that its chromatic polynomial is given by
$$P(G,x) = x(x-1)^{n-1}.$$
19. Use the result of Exercise 18 to determine the chromatic number of a tree.
20. Find the chromatic number of a tree by showing that every tree is bipartite.
21. Show that the converse of the result in Exercise 18 is true, that is, if
$$P(G,x) = x(x-1)^{n-1},$$
then G is a tree.
22. Suppose that G is a tree, $\{u, v\}$ is an edge of G, and H is obtained from G by deleting edge $\{u, v\}$, but not vertices u and v. Show that H has exactly two connected components. (You may not assume any of the theorems of this section except possibly Theorem 3.18.)
23. (Peschon and Ross [1982]) In an electrical distribution system, certain locations are joined by connecting electrical lines. A system of switches is used to open or close these lines. The collection of open lines has to have two properties: (1) every location has to be on an open line, and (2) there can be no circuits of open lines, for a short on one open line in an open circuit would shut down all lines in the circuit. Discuss the mathematical problem of finding which switches to open.
24. (Ahuja, Magnanti, and Orlin [1993], Prim [1957]) Agents of an intelligence agency each know how to contact each other. However, a message passed between agents i and j has a certain probability p_{ij} of being intercepted. If we want to make sure that all agents get a message but minimize the probability of the message being intercepted, which agents should pass the message to which agents? Formulate this as a spanning tree problem. (*Hint:* You will need to use logarithms.)
25. Here is an algorithm for finding a spanning tree of a connected graph G. Pick any vertex and mark it. Include any edge to an unmarked neighboring vertex and mark that vertex. At each step, continue adding an edge from the last marked vertex to an unmarked vertex until there is no way to continue. Then go back to the most recent marked vertex from which it is possible to continue. Stop when all vertices have been marked. (This procedure is called *depth first search* and we return to it in Section 11.1.)
    (a) Illustrate the algorithm on the graphs of Figure 3.82 (disregarding the weights).
    (b) Show that there is a spanning tree of a graph that cannot be found this way.
26. Suppose that a chemical compound C_kH_m has a bond graph that is connected and has no circuits. Show that m must be $2k + 2$.
27. Find the number of spanning trees of K_n.

28. Check Cayley's Theorem by finding all labeled trees of:

    (a) Three vertices (b) Four vertices

29. Is there a tree of seven vertices:

    (a) With each vertex having degree 1?

    (b) With two vertices having degree 1 and five vertices having degree 2?

    (c) With five vertices having degree 1 and two vertices having degree 2?

    (d) With vertices having degrees 2, 2, 2, 3, 1, 1, 1?

30. Is there a tree of five vertices with two vertices of degree 3?

31. In each of the following cases, find the number of labeled trees satisfying the given degree conditions by our formula and draw the trees in question.

    (a) Vertices 1, 2, and 3 have degree 2, and vertices 4 and 5 have degree 1.

    (b) Vertex 1 has degree 2, vertex 2 has degree 3, and vertices 3, 4, and 5 have degree 1.

    (c) Vertex 1 has degree 3, vertices 2 and 3 have degree 2, and vertices 4, 5, and 6 have degree 1.

32. Find the number of labeled trees of:

    (a) Six vertices, four having degree 2

    (b) Eight vertices, six having degree 2

    (c) Five vertices, exactly three of them having degree 1

    (d) Six vertices, exactly three of them having degree 1

33. Prove that $N(d_1, d_2, \ldots, d_n) = 0$ if $\sum_{i=1}^{n} d_i \neq n - 2$. (*Hint:* Count edges.)

34. This exercise sketches a proof of Theorem 3.22. Define $M(d_1, d_2, \ldots, d_n)$ by the right-hand side of Equation (3.21). It suffices to prove that if $n \geq 2$ and all d_i are nonnegative and $\sum_{i=1}^{n} d_i = n - 2$, then

$$N(d_1, d_2, \ldots, d_n) = M(d_1, d_2, \ldots, d_n). \tag{3.22}$$

    (a) Under the given assumptions, verify (3.22) for $n = 2$.

    (b) Under the given assumptions, show that $d_i = 0$, for some i.

    (c) Suppose that i in part (b) is n. Show that

$$\begin{aligned} N(d_1, d_2, \ldots, d_{n-1}, 0) = {} & N(d_1 - 1, d_2, d_3, \ldots, d_{n-1}) + \\ & N(d_1, d_2 - 1, d_3, \ldots, d_{n-1}) + \cdots + N(d_1, d_2, d_3, \ldots, d_{n-2}, d_{n-1} - 1), \end{aligned} \tag{3.23}$$

    where a term $N(d_1, d_2, \ldots, d_{k-1}, d_k - 1, d_{k+1}, \ldots, d_{n-1})$ appears on the right-hand side of (3.23) if and only if $d_k > 0$.

    (d) Show that M also satisfies (3.23).

    (e) Verify (3.22) by induction on n. (In the language of Chapter 6, the argument essentially amounts to showing that if M and N satisfy the same recurrence and the same initial condition, then $M = N$.)

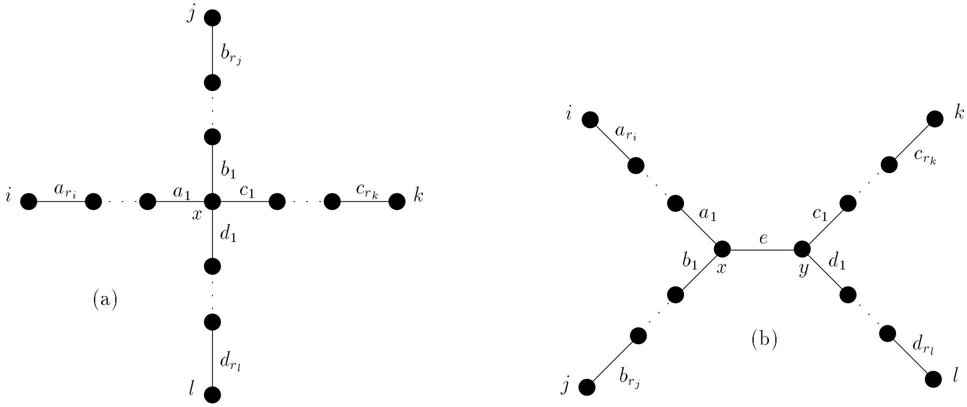


Figure 3.83: Edge-weighted trees. In these trees, the lengths of the chains from x to i, to j, to k, and to l in (a), from x to i and to j, and y to k and to l in (b) are arbitrary. Also, arbitrary positive weights a_i, b_i, c_i, d_i, e can be assigned to the edges.

35. Determine which of the following are additive distance matrices.

(a) $\begin{pmatrix} 0 & 1 & 3 \\ 1 & 0 & 2 \\ 3 & 2 & 0 \end{pmatrix}$ (b) $\begin{pmatrix} 0 & 1 & 4 \\ 1 & 0 & 2 \\ 4 & 2 & 0 \end{pmatrix}$

(c) $\begin{pmatrix} 0 & 1 & 3 & 9 \\ 1 & 0 & 2 & 2 \\ 3 & 2 & 0 & 1 \\ 9 & 2 & 1 & 0 \end{pmatrix}$

36. (a) Show that in tree (a) of Figure 3.83, all three sums in Theorem 3.23 are equal.

(b) Show that this conclusion fails for tree (b) of Figure 3.83.

37. Suppose that T is an edge-weighted tree and

$$d(i,j) + d(k,l) \leq d(i,k) + d(j,l) = d(i,l) + d(j,k).$$

Show that the vertices i, j, k, l are located in either configuration (a) or configuration (b) of Figure 3.83.

38. Suppose that T is a phylogenetic tree and each vertex has at most three neighbors. This occurs when each "evolutionary event" involves the split of a population into two new ones. Conclude from Exercise 37 that for all i, j, k, l, one of the three sums in Theorem 3.23 is strictly less than the other two.

3.6 APPLICATIONS OF ROOTED TREES TO SEARCHING, SORTING, AND PHYLOGENY RECONSTRUCTION[27]

3.6.1 Definitions

Using trees to search through a table or a file is one of the most important operations in computer science. In Example 2.18 we discussed the problem of searching through a file to find the key (identification number) of a particular person, and pointed out that there were more efficient ways to search than to simply go through the list of keys from beginning to end. In this section we show how to do this using search trees. Then we show how trees can be used for another important problem in computer science: sorting a collection into its natural order, given a list of its members. Finally, we use keyword ideas to formulate a different approach to the phylogenetic tree reconstruction problem discussed in Section 3.5.7.

Before defining search trees, let us note that each of the examples of trees in Figures 3.72–3.74 is drawn in the following way. Each vertex has a *level*, $0, 1, 2, \ldots, k$. There is exactly one vertex, the *root*, which is at level 0. All adjacent vertices differ by exactly one level and each vertex at level $i + 1$ is adjacent to exactly one vertex at level i. Such a tree is called a *rooted tree*. (It is not hard to show that every tree, after designating one vertex as the root, can be considered a rooted tree; see Exercise 5.) The number k is called the *height* of the rooted tree. In the example of Figure 3.72, the level 0 (root) vertex is Boss; the level 1 vertices are Jones, Smith, and Brown; the level 2 vertices are Johnson, Jackson, and so on; and the level 3 vertices are Allen, Baker, and so on. The height is 3. In a rooted tree, all vertices adjacent to vertex u and at a level below u's are called the *children* of u. For instance, in Figure 3.72, the children of Brown are White, Black, and Green. All vertices that are joined to u by a chain of vertices at levels below u's in the tree are called *descendants* of u. Thus, in our example, the descendants of Brown are White, Black, Green, Engel, Freyer, and Gold.

A rooted tree is called *m-ary* if every vertex has m or fewer children. A 2-ary rooted tree is called a *binary tree*. A rooted tree is called *complete m-ary* if every vertex has either 0 or m children. Figure 3.84 shows a complete binary tree. In a binary tree, we shall assume that any child of a vertex is designated as a *left child* or a *right child*.

Example 3.33 Code Trees When data need to be transmitted, each symbol is *encoded* or assigned a binary string. Suppose that the data consist of grades: 15 A's, 27 B's, 13 C's, 4 D's, and 1 F. If each bit string is the same length, then we need bit strings of length 3 (or more) to represent these five symbols. Therefore, this dataset requires at least

$$(15 + 27 + 13 + 4 + 1) \cdot 3 = 180 \text{ bits}.$$

[27] This section may be omitted.

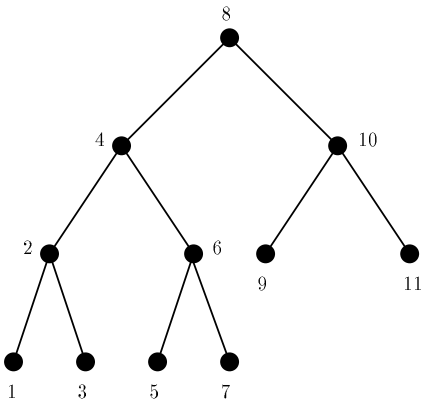


Figure 3.84: A complete binary search tree.

If we were trying to minimize the number of bits to transmit, bit strings of length 4 or more would not be used.

If we are allowed to use variable-length bit strings, substantial savings can be reaped. Suppose that we represent the grades as follows: A: 01, B: 1, C: 111, D: 10, F: 010. We now require only

$$(15 \cdot 2) + (27 \cdot 1) + (13 \cdot 3) + (4 \cdot 2) + (1 \cdot 3) = 107 \text{ bits}.$$

Although this is a savings in bits sent, there is a problem with this encoding. Consider the following transmission: $01011 \cdots$. When the receiver begins to decode, what is the first grade? An A might be assumed since the transmission begins with 01. However, an F could also be construed since the transmission also begins with 010. This problem arises since a bit string is the prefix of some other bit string. We need to find an encoding where this doesn't happen. Such an encoding is called a *prefix code*.

Prefix codes are easy to find. (In fact, the length 3 bit string encoding from above is a prefix code.) Consider the binary tree in Figure 3.85. Left branches are labeled with 0 and right branches with 1. This generates a bit string to associate with each vertex by looking at the unique chain from the root to that vertex. The vertices representing the various grades are labeled accordingly. Notice that the unique chain to the vertex labeled B (010) starts at the root and goes left (0), then right (1), then left (0).

This prefix code yields a savings in bits over the code assigning bit strings of length 3 to each grade. But is this the best that we could do? The trees associated, as above, with prefix codes are called *code trees*. Figure 3.85 is but one example of a *code tree* for this example. The problem of minimizing the number of bits to transmit data is equivalent to finding an optimal code tree. Given a set of symbols and their frequencies, Huffman [1952] provides an algorithm for the construction of an optimal code tree. The optimal code tree output of the following algorithm is sometimes called a *Huffman tree*.

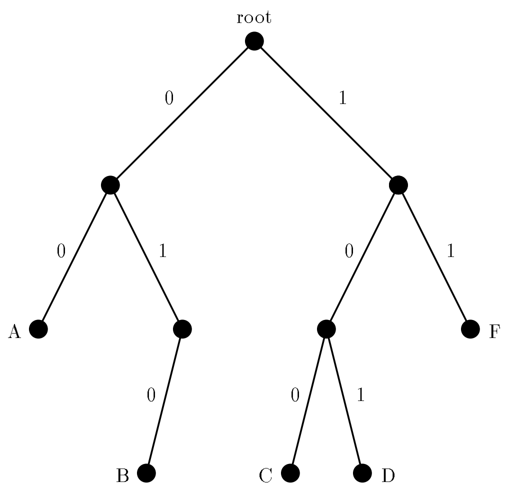


Figure 3.85: A code tree.

Algorithm 3.2: Optimal Code Tree

Input: A forest S of n isolated vertices (rooted trees) with labels the symbols $\{s_1, s_2, \ldots, s_n\}$ and weights the frequencies $\{f_1, f_2, \ldots, f_n\}$, respectively.
Output: An optimal code tree.

Step 1. Find two trees T, T' in S with the two smallest root weights f_i, f_j.

Step 2. Create a new complete binary tree T'' with root s_{ij} and weight $f_{ij} = f_i + f_j$ and having T and T' as its children.

Step 3. Replace T and T' in S with T''. If S is a tree, stop and output S. If not, return to step 1.

To give an example, suppose that the following data need to be transmitted: 15 A's, 4 B's, 10 C's, 15 D's, 11 Passes, and 3 Fails. Figure 3.86 shows the construction of the optimal code tree using Algorithm 3.2. Note that 6 distinct pieces of information (A, B, C, D, Pass, Fail) would need, at least, length 3 bit strings if all bit strings must be of the same length. The total number of bits for transmission would thus equal $(15 + 4 + 10 + 15 + 11 + 3) \cdot 3 = 174$. Using Algorithm 3.2, we find that only

$$(15 \cdot 2) + (4 \cdot 4) + (10 \cdot 3) + (15 \cdot 2) + (11 \cdot 2) + (3 \cdot 4) = 140$$

bits are needed, a savings of almost 20 percent. The optimality of the algorithm is proven in the exercises (see Exercise 20). For more on codes see Chapter 10. ■

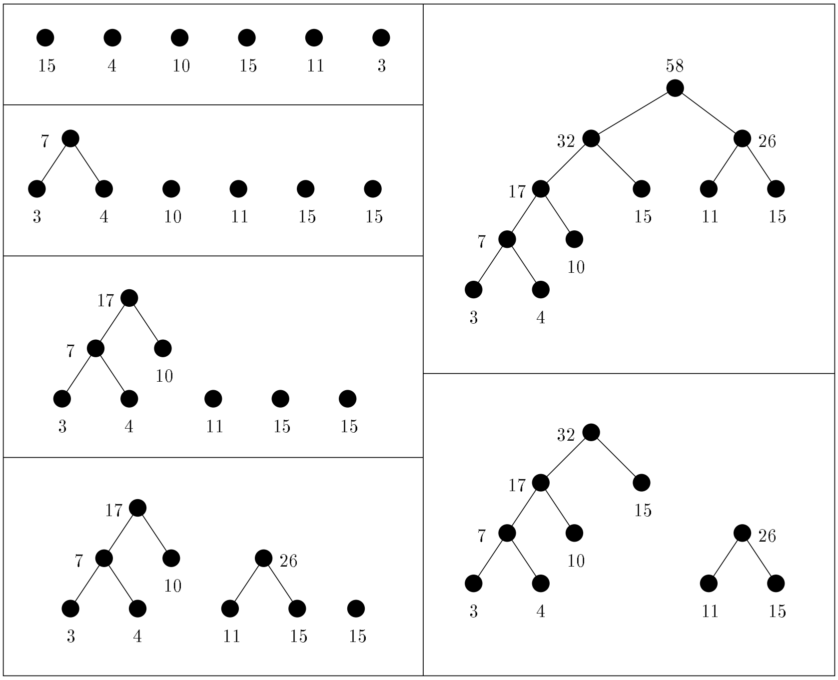


Figure 3.86: Optimal code tree using Algorithm 3.2 for transmitting 15 A's, 4 B's, 10 C's, 15 D's, 11 Passes, and 3 Fails. (Figure/algorithm starts in upper left-hand corner and goes counterclockwise.)

3.6.2 Search Trees

Suppose that we have a file of n names. Let T be any (rooted) binary tree. Label each vertex u of T with a key (a real number) in such a way that if u has a child, its left child and all descendants of its left child get keys that are lower numbers than the key of u, and its right child and all descendants of its right child get keys that are higher numbers than the key of u. A binary tree with such a labeling will be called a *binary search tree.* (Such a labeling can be found for every binary tree; see below.) Figure 3.84 shows a complete binary search tree, where the keys are the integers $1, 2, \ldots, 11$. Now given a particular key k, to locate it we search through the binary search tree, starting with the root. At any stage we look at the key of the vertex x being examined. At the start, x is the root. We ask if k is equal to, less than, or greater than the key of x. If equal, we have found the desired file. If less, we look next at the left child of x; if greater, we look next at the right child of x. We continue the same procedure. For instance, to find the key 7 using the binary search tree of Figure 3.84, we start with key 8, that of the root. Then we go to key 4 (since 7 is less than 8), then to key 6 (since 7 is higher than 4), then to key 7 (since 7 is higher than 6). Notice that our search took four steps rather than the seven steps it would have taken to go through the list $1, 2, \ldots, 11$ in order. See

Exercise 23 for another application of binary search trees.

Suppose that we can find a binary search tree. What is the computational complexity of a file search in the worst case? It is the number of vertices in the longest chain descending from the root to a vertex with no children; that is, it is one more than the height of the binary search tree. Obviously, the computational complexity is minimized if we find a binary search tree of minimum height. Now any binary tree can be made into a binary search tree. For a proof and an algorithm, see, for example, Reingold, Nievergelt, and Deo [1977]. Hence, we are left with the following question. Given n, what is the smallest h such that there is a binary tree on n vertices with height h? We shall answer this question through the following theorem, which is proved in Section 3.6.3. Recall that $\lceil a \rceil$ is the least integer greater than or equal to a.

Theorem 3.24 The minimum height of a binary tree on n vertices is equal to $\lceil \log_2(n+1) \rceil - 1$.

The binary tree of Figure 3.84 is a binary tree on 11 vertices that has the minimum height 3 since

$$\lceil \log_2(n+1) \rceil = \lceil \log_2 12 \rceil = \lceil 3.58 \rceil = 4.$$

In sum, Theorem 3.24 gives a logarithmic bound

$$\lceil \log_2(n+1) \rceil - 1 + 1 = \lceil \log_2(n+1) \rceil$$

on the computational complexity of file search using binary search trees. This bound can of course be attained by finding a binary search tree of minimum height, which can always be done (see the proof of Theorem 3.24). To summarize:

Corollary 3.24.1 The computational complexity of file search using binary search trees is $\lceil \log_2(n+1) \rceil$.

This logarithmic complexity is in general a much better complexity than the complexity n we obtained in Example 2.18 for file search by looking at the entries in a list in order. For $\lceil \log_2(n+1) \rceil$ becomes much less than n as n increases.

3.6.3 Proof of Theorem 3.24[28]

To prove Theorem 3.24, we first prove the following.

Theorem 3.25 If T is a binary tree of n vertices and height h, then $n \leq 2^{h+1} - 1$.

Proof. There is one vertex at level 0, there are at most $2^1 = 2$ vertices at level 1, there are at most $2^2 = 4$ vertices at level 2, there are at most $2^3 = 8$ vertices at level 3, ..., and there are at most 2^h vertices at level h. Hence,

$$n \leq 1 + 2^1 + 2^2 + 2^3 + \cdots + 2^h. \tag{3.24}$$

[28] This subsection may be omitted.

Now we use the general formula

$$1 + x + x^2 + \cdots + x^h = \frac{1 - x^{h+1}}{1 - x}, \quad x \neq 1, \tag{3.25}$$

which will be a very useful tool throughout this book. Substituting $x = 2$ into (3.25) and using (3.24), we obtain

$$n \leq \frac{1 - 2^{h+1}}{1 - 2} = 2^{h+1} - 1.$$

Q.E.D.

Proof of Theorem 3.24. We have

$$\begin{aligned} 2^{h+1} &\geq n + 1, \\ h + 1 &\geq \log_2(n + 1), \\ h &\geq \log_2(n + 1) - 1. \end{aligned}$$

Since h is an integer,

$$h \geq \lceil \log_2(n + 1) \rceil - 1.$$

Thus, every binary tree on n vertices has height at least $\lceil \log_2(n + 1) \rceil - 1$. It is straightforward to show that there is always a binary tree of n vertices whose height is exactly $p = \lceil \log_2(n+1) \rceil - 1$. Indeed, any complete binary tree in which the only vertices with no children are at level p or $p - 1$ will suffice to demonstrate this.[29]

Q.E.D.

3.6.4 Sorting

A basic problem in computer science is the problem of placing a set of items in their natural order, usually according to some numerical value. We shall call this problem the *sorting problem*. We shall study the problem of sorting by making comparisons of pairs of items.

Any algorithm for sorting by comparisons can be represented by a (complete) binary tree called a *decision tree*. Figure 3.87 shows a decision tree for a computer program that would sort the three numbers a, b, and c. In each case, a vertex of the tree corresponds to a test question or an output. At a test question vertex, control moves to the left child if the question is answered "yes" and to the right child if "no." Output vertices are shown by squares, test vertices by circles. The complexity of the algorithm represented by the decision tree T of Figure 3.87 is the number of steps (comparisons) required to reach a decision in the worst case. Since outputs correspond to vertices with no children, the complexity is obtained by finding one less than the number of vertices in the longest chain from the root to a vertex of T with no children, that is, by finding the height of the binary tree T. In our example, the height is 3.

In a rooted tree, let us call vertices with no children *leaves*. (These correspond to the leaves defined in Section 3.5.7.) Note that to sort a set of p (distinct) items,

[29] Such a binary tree is called *balanced*. Figure 3.84 is an example of a balanced binary tree.

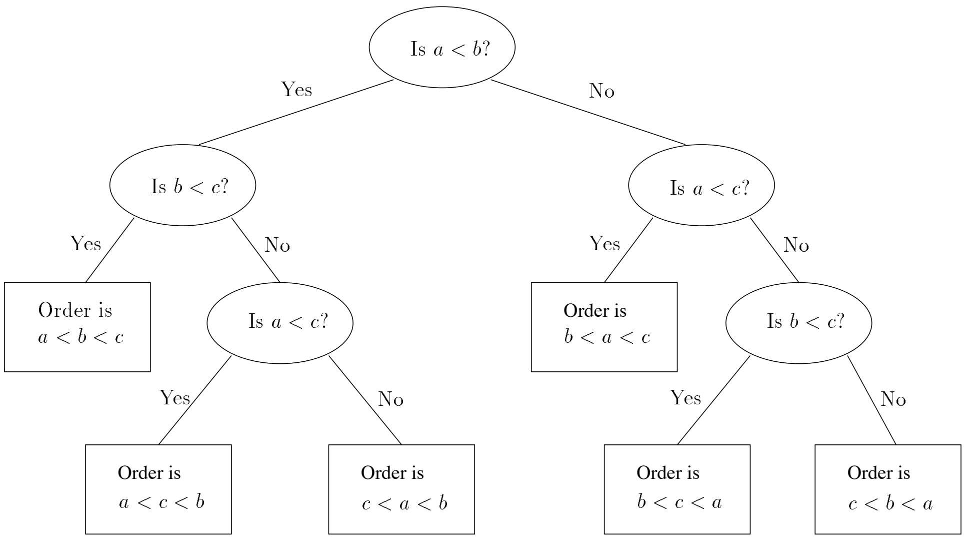


Figure 3.87: A decision tree T for sorting the set of numbers $\{a, b, c\}$.

there are $p!$ possible orders, so a decision tree for the sorting will have at least $p!$ leaves. (We say "at least" because several chains may lead from a root to the same order.) Now we have the following theorem, whose proof is left as an exercise (Exercise 33).

Theorem 3.26 A binary tree of height h has at most 2^h leaves.

This theorem is easily illustrated by the binary trees of Figure 3.74 or 3.84.

Theorem 3.27 Any algorithm for sorting $p \geq 4$ items by pairwise comparisons requires in the worst case at least $cp \log_2 p$ comparisons, where c is a positive constant.

Proof. We have already observed that any decision tree T that sorts p items must have at least $p!$ leaves. Thus, the number of comparisons in the worst case, which is the height of the tree T, has to be at least $\log_2 p!$ (for $2^{\log_2 p!} = p!$). Now for $p \geq 1$,

$$p! \geq p(p-1)(p-2)\cdots\left(\left\lceil\frac{p}{2}\right\rceil\right) \geq \left(\frac{p}{2}\right)^{p/2}.$$

Thus, for $p \geq 4$,

$$\log_2 p! \geq \log_2\left(\frac{p}{2}\right)^{p/2} = \frac{p}{2}\log_2\left(\frac{p}{2}\right) \geq \frac{p}{4}\log_2 p = \frac{1}{4}p\log_2 p.$$

Q.E.D.

There are a variety of sorting algorithms that actually achieve the bound in Theorem 3.27, that is, can be carried out in a constant times $p \log_2 p$ steps. Among the better-known ones is heap sort. In the text and exercises we discuss two well-known sorting algorithms, bubble sort and quik sort, which do not achieve the bound. For careful descriptions of all three of these algorithms, see, for example, Baase [1992], Brassard and Bratley [1995], or Manber [1989]. Note that $cp\log_2 p \leq cp^2$, so an algorithm that takes $cp\log_2 p$ steps is certainly a polynomially bounded algorithm.

In the algorithm known as *bubble sort*, we begin with an ordered set of p (distinct) items. We wish to put them in their proper (increasing) order. We successively compare the ith item to the $(i+1)$st item in the list, interchanging them if the ith item is larger than the $(i+1)$st. The algorithm is called bubble sort because the larger items rise to the top much like the bubbles in a glass of champagne. Here is a more formal statement of the algorithm.

Algorithm 3.3: Bubble Sort

Input: An ordered list $a_1a_2\ldots a_p$ of p items.
Output: A listing of $a_1, a_2, \ldots, a_p$ in increasing order.

Step 1. Set $m = p - 1$.

Step 2. For $i = 1, 2, \ldots, m$, if $a_i > a_{i+1}$, interchange a_i and a_{i+1}.

Step 3. Decrease m by 1. If m is now 0, stop and output the order. If not, return to step 2.

To illustrate bubble sort, suppose that we start with the order 516423. We first set $m = p - 1 = 5$. We compare 5 to 1, and interchange them, getting 156423. We then compare 5 to 6, leaving this order as is. We compare 6 to 4, and interchange them, getting 154623. Next, we compare 6 to 2, and interchange them, getting 154263. Finally, we compare 6 to 3, interchange them, and get 154236. We decrease m to 4 and repeat the process, getting successively 154236, 145236, 142536, and 142356. Note that we do not have to compare 5 to 6, since m is now 4. We decrease m to 3 and repeat the process, getting 142356, 124356, and 123456. Then m is set equal to 2 and we get 123456 and 123456. Note that no more interchanges are needed. Next, we set $m = 1$. No interchanges are needed. Finally, we set $m = 0$, and we output the order 123456.

Part of the decision tree for bubble sort on an order a, b, c, d is shown in Figure 3.88.

Note that bubble sort requires $p(p-1)/2$ comparisons. For at the mth iteration or repetition of the procedure, m comparisons are required, and m takes on the values $p-1, p-2, \ldots, 1$. Thus, using the standard formula for the sum of an arithmetic progression, we see that a total of

$$(p-1) + (p-2) + \cdots + 1 = \frac{p(p-1)}{2}$$

steps are needed. In the language of Section 2.18, the algorithm bubble sort is not as efficient as an algorithm that requires $cp\log_2 p$ steps. For bubble sort is an $O(p^2)$ algorithm and p^2 is not $O(p\log_2 p)$.

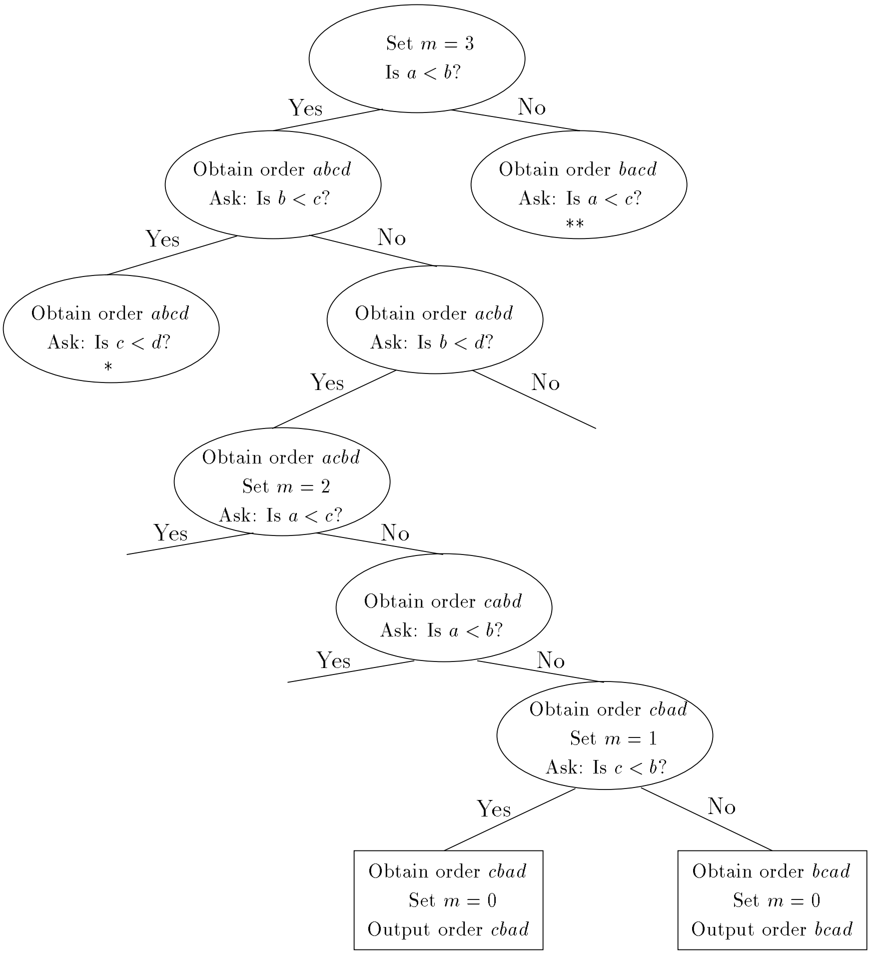


Figure 3.88: Part of the decision tree for bubble sort on an ordered set of four items, $abcd$. (The labels * and ** are for Exercise 28.)

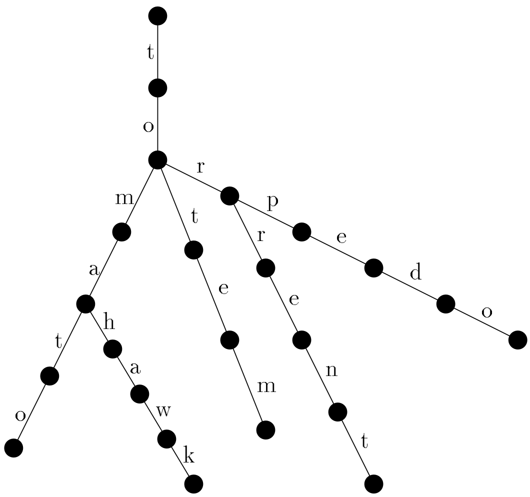


Figure 3.89: A keyword tree.

3.6.5 The Perfect Phylogeny Problem

In this section we describe a different model than that given in Section 3.5.7 for reconstructing the phylogenetic tree corresponding to a given set of presently existing species.

Let Σ be an alphabet (set of "letters"). A *pattern* from Σ is a sequence of letters from the alphabet. For example, if Σ consists of the 26 letters of the alphabet, "tomato" is a pattern, and if $\Sigma = \{0, 1\}$, a bit string is a pattern. Let P be a finite set of patterns from Σ. A *keyword tree* for P is a rooted tree satisfying the following conditions:

- Every edge is labeled with exactly one letter from Σ.
- Any two edges out of a given vertex have distinct labels.
- Every pattern in P corresponds to exactly one leaf so that the letters on the path from the root to the leaf, in that order, give the pattern.

To give an example, Figure 3.89 shows the keyword tree for the set of patterns {tomato, tomahawk, totem, torrent, torpedo}. Algorithms for constructing keyword trees are described in Gusfield [1997].

A way to reconstruct evolutionary history is to study characters that biological objects, such as species, might or might not have. These can be features such as possessing a backbone or having feathers or walking upright. Or they can be more subtle things such as "protein A enhances the expression of protein B" or that the species has a given nucleotide (e.g., adenine, A) in the pth position in its DNA sequence. In all of these cases, we say that we have *binary characters* where objects, in particular species, either have the character or don't. Suppose that we

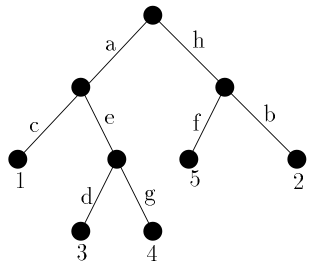


Figure 3.90: A phylogenetic tree.

are interested in a set S of n species and a set C of m characters. We can think of evolution as taking place from an ancestor (the root of the tree) that has none of the given characters in the set C. At some point, a character becomes evident. We then label the edge leading from a given vertex without a given character to its child by putting the name of the character on the edge. For the sake of simplicity, we assume that once a character becomes evident, it never disappears. Then by the time we get to a leaf of the tree, the path from the root to that leaf gives the names of all of the characters displayed by that species. Figure 3.90 gives an example. Here the species 1 has the characters a and c, the species 3 has the characters a, e, d, and so on.

In practice, we want to reconstruct a tree like that in Figure 3.90 given information about what characters each species has. To formulate this problem precisely, let us take M to be an $n \times m$ matrix of 0's and 1's, with its (i, j) entry equal to 1 if and only if species i has character j. We say that a *phylogenetic tree* for M is a rooted tree with exactly n leaves and exactly m edges with the following properties:

- Each of the n species in the set S labels exactly one leaf.
- Each of the m characters in the set C labels exactly one edge.
- For any species, the characters that label the edges along the path from the root to the leaf labeled i specify all of the characters j such that $M(i, j) = 1$.

We would like to know whether or not, given a matrix M, we can construct a phylogenetic tree for M. This is known as the *perfect phylogeny problem*. For instance, consider the matrix M given by

$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \!\!\begin{array}{c} \begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \end{array}.$$

Then a phylogenetic tree for M is given by the tree of Figure 3.90. In particular, a phylogenetic tree for M is a kind of keyword tree for the sets of patterns defined by the characters that a given species has. Not every M has a phylogenetic tree. For instance, the following matrix does not:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

We will see from the next theorem why there is no phylogenetic tree. However, the reader might wish to experiment with this small example to try to construct such a tree before reading the theorem.

To solve the perfect phylogeny problem, let O_j be the set of species with a 1 in column j of M.

Theorem 3.28 (Estabrook, Johnson, and McMorris [1975, 1976a,b]) A matrix M has a phylogenetic tree if and only if for j and k, either O_j and O_k are disjoint or one contains the other.

An algorithm for constructing a phylogenetic tree in $O(mn)$ steps was given by Gusfield [1991]. See Gusfield [1997] for more details. The problem becomes more complicated if each character can take on more than two possible states. For instance, we might say that the pth position in the DNA chain can take on states A, T, C, or G. The perfect phylogeny problem reformulated in this more general setting is studied by, among others, Agarwala and Fernandez-Baca [1994], Bodlaender, Fellows, and Warnow [1992], Kannan and Warnow [1994, 1995], and Steel [1992].

EXERCISES FOR SECTION 3.6

1. In the rooted tree of Figure 3.91, find the level of each vertex.
2. In each rooted tree of Figure 3.92, find the level of each vertex.
3. In each rooted tree of Figure 3.92, find the height of the tree.
4. In each rooted tree of Figure 3.92, find all descendants of the vertex b.
5. For each tree in Figure 3.71, label each vertex and select a vertex for the root. Determine the level of the remaining vertices and the height of the tree.
6. Is the tree of Figure 3.91 a binary search tree? If so, describe how to find the key 7.
7. Find a balanced binary tree (as defined in footnote 29) with n vertices where:

   (a) $n = 5$ (b) $n = 8$ (c) $n = 12$ (d) $n = 15$

8. Find a complete binary tree of:

   (a) Height 3 and 8 leaves (b) Height 3 and fewer than 8 leaves

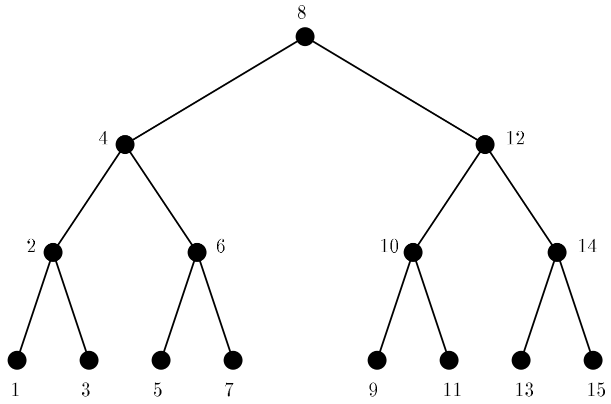


Figure 3.91: Rooted tree for Exercises of Section 3.6.

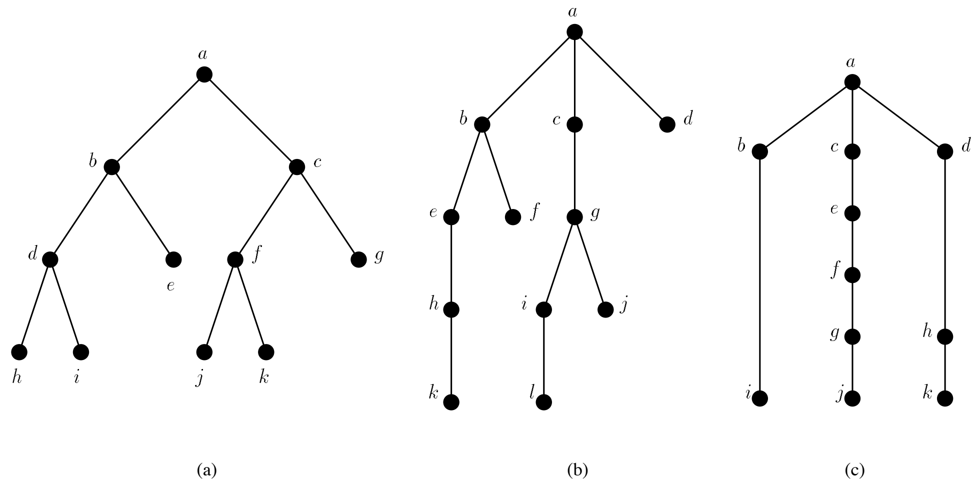


Figure 3.92: Rooted trees for Exercises of Section 3.6.

9. Find a complete binary tree of 11 vertices with:

   (a) As large a height as possible (b) As small a height as possible

10. Find a binary search tree with n vertices where n is:

    (a) 10 (b) 14 (c) 18 (d) 20

11. Find the minimum height of a binary tree of n vertices where n is:

    (a) 74 (b) 512 (c) 4095

12. A complete binary tree can be used to encode bit strings of n bits as follows. At any vertex, its left child is labeled 0 and its right child is labeled 1. A bit string then corresponds to a simple chain from the root to a vertex with no children. Draw such a tree for $n = 4$ and identify the simple chain corresponding to the string 1011.

13. Of all ternary (3-ary) trees on n vertices, what is the least possible height?

14. Using the code tree in Figure 3.85, decode the following transmissions:

    (a) 0100100010110011 (b) 00010111001010001011100101

15. Suppose that a dataset D consists of the four suits in a deck of cards.

    (a) Find a prefix code for D whose longest bit string is length 2. Draw the associated code tree.

    (b) Find a prefix code for D with one bit string of length 3 and the rest shorter. Draw the associated code tree.

16. Find bit strings of length 4 or less for g and h to produce a prefix code for $\{a, b, c, d, e, f, g, h\}$, where the following bit strings have already been assigned:

    $$a : 00 \quad b : 011 \quad c : 10 \quad d : 1100 \quad e : 1101 \quad f : 111.$$

    (*Hint:* Draw the code tree.)

17. Using Algorithm 3.2, find an optimal code tree for transmitting the following grades:

    (a) 4 A's, 4 B's, 4 B−'s, 4 C's, 4 C−'s, 4 D's, 4 D−'s, 4 F's

    (b) 8 A's, 6 A−'s, 4 B+'s, 1 B, 5 B−'s, 9 C+'s, 11 C's, 5 C−'s, 3 D+'s, 3 D's, 5 D−'s, 1 F

18. Consider the following data: 3 A's, 4 B's, 7 C's, 14 D's, 14 F's.

    (a) Show that Algorithm 3.2 can produce optimal code trees of different heights.

    (b) How can you modify Algorithm 3.2 to produce minimum-height optimal code trees?

19. Find a set of data for which the tree of Figure 3.93 is an optimal code tree.

20. This exercise proves the optimality of Algorithm 3.2's output.

    (a) Show that an optimal code tree for a given set of data is always a complete binary tree.

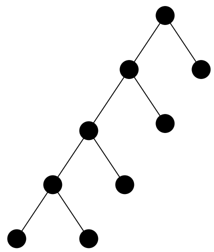


Figure 3.93: An optimal code tree.

(b) Show that Algorithm 3.2's output is optimal for datasets containing two symbols.

(c) Assume that Algorithm 3.2's output is optimal for datasets containing $n-1$ symbols, $n > 2$. Then, given a dataset D containing n symbols, remove the two symbols s_i and s_j with smallest frequencies (weights) f_i and f_j, and replace them with a single symbol s_{ij} with frequency $f_{ij} = f_i + f_j$. What can you say about Algorithm 3.2's output on these new data?

(d) Attach to vertex s_{ij} two children s_i and s_j with weights f_i and f_j, respectively. Explain why this new tree would be produced by Algorithm 3.2.

(e) Explain why this new tree is an optimal code tree for a dataset of n symbols.

21. Suppose that we have a *fully balanced binary search tree*, that is, a balanced binary search tree (see footnote 29 on page 207) in which every vertex with no children is at the same level. Assume that a file is equally likely to be at any of the n vertices of T. What is the computational complexity of file search using T if we measure complexity using the average number of steps to find a file rather than the largest number of steps to find one.

22. In a complete m-ary rooted tree, if a vertex is chosen at random, show that the probability that it has a child is about $1/m$.

23. (Tucker [1984]) In a compiler, a control word is stored as a number. Suppose that the possible control words are GET, DO, ADD, FILL, STORE, REPLACE, and WAIT, and these are represented by the numbers 1, 2, 3, 4, 5, 6, and 7 (in binary notation), respectively. Given an unknown control word X, we wish to test it against the possible control words on this list until we find which word it is. One approach is to compare X's number in order to the numbers $1, 2, \ldots, 7$ corresponding to the control words. Another approach is to build a binary search tree. Describe how the latter approach would work and build such a tree.

24. The *binary search algorithm* searches through an ordered file to see if a key x is in the file. The entries of the file are n numbers, $x_1 < x_2 < \cdots < x_n$. The algorithm compares x to the middle entry x_i in the file, the $\lceil n/2 \rceil$th entry. If $x = x_i$, the search is done. If $x < x_i$, then $x_i, x_{i+1}, \ldots, x_n$ are eliminated from consideration, and the algorithm searches through the file $x_1, x_2, \ldots, x_{i-1}$, starting with the middle entry. If $x > x_i$, then $x_1, x_2, \ldots, x_i$ are eliminated from consideration, and the algorithm searches through the file $x_{i+1}, x_{i+2}, \ldots, x_n$, starting with the middle entry. The procedure is repeated iteratively. Table 3.5 shows two examples. (See Knuth [1973]

Table 3.5: Two Binary Searches[a]

Searching for 180												
71	97	164	180	285	<u>436</u>	513	519	522	622	663	687	Compare 180 to 436
71	97	<u>164</u>	180	285								Compare 180 to 164
			<u>180</u>	285								Compare 180 to 180 180 has been found
Searching for 515												
71	97	164	180	285	<u>436</u>	513	519	522	622	663	687	Compare 515 to 436
						513	519	<u>522</u>	622	663	687	Compare 515 to 522
						<u>513</u>	519					Compare 515 to 513
							<u>519</u>					Compare 515 to 519 Conclude that 515 is not in the file

[a] In each case, the first list contains the whole file, the underlined number is the middle entry, and each subsequent line shows the remaining file to be searched.

for details.) Apply the binary search algorithm to search for 150 in the following ordered files:

(a) 22 36 83 148 150 190 201 225 216 221 301

(b) 18 29 149 301 399 625 648 833 901 949

25. Compute the computational complexity of the binary search algorithm defined in Exercise 24. (*Hint:* Show that a binary search tree is being used.)

26. Apply bubble sort to the order 4312 and illustrate the steps required to put this in proper order.

27. Apply bubble sort to the following orders.

    (a) 516324 (b) 346152

28. In the partial decision tree for bubble sort in Figure 3.88, fill in the part beginning at the vertex labeled:

    (a) * (b) **

29. Compare the worst-case complexity of the bubble sort algorithm to the bound in Theorem 3.27 for p equal to:

    (a) 7 (b) 15

30. In the sorting algorithm known as *quik sort*, we start with an ordered list of p items and try to find the proper order. We select the first item from the list and divide the remaining items into two groups, those less than the item selected and those greater than it. We then put the item chosen in between the two groups and repeat the process on each group. We eventually get down to groups of one element, and we stop. For instance, given the order 31564827, we select item 3. Then the items before 3 are listed as 12, those after 3 as 56487. We now apply the algorithm to

sort these two groups. For instance, choosing 5 from the second group gives us the two subgroups 4 and 687. We now order these. And so on. Apply quik sort to the following ordered lists:

(a) 5176324 (b) 941258376

31. How many steps (comparisons) does the algorithm quik sort (Exercise 30) require in the worst case if we start with a list $123 \cdots p$ and $p = 5$?
32. Repeat Exercise 31 for arbitrary p.
33. Prove Theorem 3.26 by induction on h.
34. Find the keyword tree corresponding to the set of patterns {sentry, seldom, spackle, spanking, spanning, seller}.
35. Find the keyword tree corresponding to the set of patterns {ATTCG, AATGC, ATGCC, AATTT, AATGTG} on the alphabet $\Sigma = \{$A, T, G, C$\}$.
36. For each of the following matrices M of 0's and 1's, determine if the perfect phylogeny problem has a solution and, if so, find a phylogenetic tree for M.

(a) $M = \begin{array}{c|cccc} & a & b & c & d \\ \hline 1 & 1 & 0 & 1 & 0 \\ 2 & 1 & 1 & 0 & 0 \\ 3 & 1 & 0 & 1 & 0 \\ 4 & 0 & 0 & 0 & 1 \end{array}$

(b) $M = \begin{array}{c|ccccc} & a & b & c & d & e \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 \\ 3 & 1 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 1 & 0 \\ 5 & 1 & 0 & 1 & 0 & 0 \end{array}$

(c) $M = \begin{array}{c|cccc} & a & b & c & d \\ \hline 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 1 & 1 & 0 \\ 3 & 0 & 1 & 0 & 0 \\ 4 & 0 & 0 & 0 & 1 \\ 3 & 1 & 0 & 1 & 0 \end{array}$

37. Prove that if the matrix M of 0's and 1's has a phylogenetic tree, then for j and k, either O_j and O_k are disjoint or one contains the other.
38. In the *dictionary problem*, a set of patterns forming a dictionary is known. When a pattern is presented, we want to find out if it is in the dictionary. Explain how keyword trees might help with this problem.
39. Suppose that a character can take any one of the states $0, 1, 2, \ldots, s$. Let M be an $n \times m$ matrix with entries from $\{0, 1, 2, \ldots, s\}$ and $M(i, j) = p$ if species i has character j in state p. A *perfect phylogeny* for M is a rooted tree where each species labels exactly one leaf, and edges are labeled with ordered triples (j, p, q), meaning that along that edge, character j changes state from p to q. Assume that the starting state for each character at the root is given and that for any character j and any y in $\{0, 1, 2, \ldots, s\}$, there is at most one edge on any chain from the root to a leaf that has the form (j, x, y) for some y. (A character may only change to state y once on this chain.)

(a) If $s = 2$ and $M = \begin{array}{c|cc} & a & b \\ \hline 1 & 2 & 0 \\ 2 & 1 & 2 \\ 3 & 1 & 1 \\ 4 & 2 & 2 \end{array}$, find a perfect phylogeny for M.

(b) If $s = 2$ and $M = \begin{array}{c c} & \begin{array}{ccc} a & b & c \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{pmatrix} 1 & 0 & 0 \\ 2 & 2 & 2 \\ 0 & 0 & 0 \\ 0 & 1 & 2 \\ 2 & 2 & 0 \end{pmatrix} \end{array}$, find a perfect phylogeny for M.

40. Suppose that D is a digraph. A *level assignment* is an assignment of a level L_i to each vertex i so that if (i, j) is an arc, then $L_i < L_j$. Show that a digraph has a level assignment if and only if it has no cycles.

41. A *graded level assignment* for a digraph D is a level assignment (Exercise 40) such that if there is an arc (i, j) in D, then $L_j = L_i + 1$. Show that if $D = (V, A)$ has a graded level assignment, then it is *equipathic*; that is, for all u, v in V, all simple paths from u to v have the same length.

42. Suppose that T is a rooted tree. Orient T by directing each edge $\{u, v\}$ from a lower level to a higher level. The resulting digraph is called a *directed rooted tree.*

    (a) Show that every directed rooted tree has exactly one vertex from which every other vertex is reachable by a path.

    (b) Can a directed rooted tree be unilaterally connected?

    (c) Show that every directed rooted tree has a level assignment.

    (d) Suppose that D is a directed rooted tree. Show that D has a graded level assignment if and only if D is equipathic.

43. Find the number of rooted, labeled trees of:

    (a) Five vertices, two having degree 2

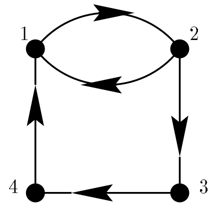
    (b) Five vertices in which the root has degree 2

    (c) Four vertices in which the root has degree 1

3.7 REPRESENTING A GRAPH IN THE COMPUTER

Many problems in graph theory cannot be solved except on the computer. Indeed, the development of high-speed computing machines has significantly aided the solution of graph-theoretical problems—for instance, the four-color problem—and it has also aided the applications of graph theory to other disciplines. At the same time, the development of computer science has provided graph theorists with a large number of important and challenging problems to solve. In this section we discuss various ways to represent a digraph or graph as input to a computer program. It also allows us to demonstrate how another area of mathematics, matrix theory, can aid in graph-theoretic problems.

Note that a diagram of a digraph or graph such as the diagrams we have used is not very practical for large digraphs or graphs and also not very amenable for input into a computer. Rather, some alternative means of entering digraphs and graphs are required. The best way to input a digraph or graph depends on the properties



$$\mathbf{A}(D) = \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

D

Figure 3.94: A digraph with its adjacency matrix $\mathbf{A}(D)$.

of the digraph or graph and the use that will be made of it. The efficiency of an algorithm depends on the choice of method for entering a digraph or graph, and the memory storage required depends on this as well. Here, we shall mention a number of different ways to input a digraph or graph.

Let us concentrate on entering digraphs, recalling that graphs can be thought of as special cases. One of the most common approaches to entering a digraph D into a computer is to give its *adjacency matrix* $\mathbf{A} = \mathbf{A}(D)$. This matrix is obtained by labeling the vertices of D as $1, 2, \ldots, n$, and letting the i, j entry of $\mathbf{A}$ be 1 if there is an arc from i to j in D, and 0 if there is no such arc. Figure 3.94 shows a digraph and its associated adjacency matrix.[30] Note that the adjacency matrix can be entered as a two-dimensional array or as a bit string with n^2 bits, by listing one row after another. Thus, n^2 bits of storage are required, $n(n-1)$ if the diagonal elements are assumed to be 0, which is the case for loopless digraphs. The storage requirements are less for a graph, thought of as a symmetric digraph, since its adjacency matrix is symmetric. Specifically, if there are no loops, we require

$$(n-1) + (n-2) + \cdots + 1 = \frac{n(n-1)}{2}$$

bits of storage. For we only need to encode the entries above the diagonal. If a digraph is very sparse, that is, has few arcs, the adjacency matrix is not a very useful representation.

Another matrix that is useful for entering graphs, although not digraphs, is called the incidence matrix. In general, suppose that S is a set and $\mathcal{F}$ is a family of subsets of S. Define the *point-set incidence matrix* $\mathbf{M}$ as follows. Label the elements of S as $1, 2, \ldots, n$ and the sets in $\mathcal{F}$ as $S_1, S_2, \ldots, S_m$. Then $\mathbf{M}$ is an $n \times m$ matrix and its i, j entry is 1 if element i is in set S_j and 0 if element i is not in set S_j. For example, if $S = \{1, 2, 3, 4\}$ and

$$\mathcal{F} = \{\{1, 2\}, \{2, 3\}, \{1, 4\}\},$$

[30] Of course, the specific matrix obtained depends on the way we list the vertices. But by an abuse of language, we call any of these matrices *the* adjacency matrix.

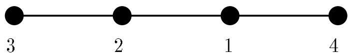


Figure 3.95: The graph whose incidence matrix is given by (3.26).

then $\mathbf{M}$ is given by

$$\mathbf{M} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{ccc} \{1,2\} & \{2,3\} & \{1,4\} \end{array} \\ \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}. \tag{3.26}$$

If G is a graph, its *incidence matrix* is the point-set incidence matrix for the set $S = V(G)$ and the set $\mathcal{F} = E(G)$. The matrix $\mathbf{M}$ of (3.26) is the incidence matrix for the graph of Figure 3.95. An incidence matrix requires $n \times e$ bits of storage, where e is the number of edges. This tends to be larger than $\frac{1}{2}n(n-1)$, the storage required for the adjacency matrix, and it is always at least $n(n-1) = n^2 - n$ for connected graphs. (Why?) Incidence matrices are widely used in inputting electrical networks and switching networks. It is possible to define an incidence matrix for digraphs as well (see Exercise 20).

Still another approach to inputting a digraph D is simply to give an *arc list*, a list of its arcs. How much storage is required here? Suppose that each vertex label is encoded by a bit string of length t. Then if D has a arcs, since each arc is encoded as a pair of codewords, $2at$ bits of storage are required. Note that t must be large enough so that 2^t, the number of bit strings of length t, is at least n, the number of vertices. Thus, $t \geq \log_2 n$. In fact,

$$t \geq \lceil \log_2 n \rceil,$$

where $\lceil x \rceil$ is the least integer greater than or equal to x. Hence, we require at least $2a\lceil \log_2 n \rceil$ bits of storage. If there are not many arcs, that is, if the adjacency matrix is sparse, this can be a number smaller than n^2 and so require less storage than the adjacency matrix.

There are two variants on the arc list. The first is to input two linear orders or arrays, each having a codewords where a is the size of the arc set. If these arrays are $(h_1, h_2, \ldots, h_a)$ and $(k_1, k_2, \ldots, k_a)$, and h_i encodes for the vertex x and k_i for the vertex y, then (x, y) is the ith arc on the arc list. The storage requirements are the same as for an arc list.

Another variant on the arc list is to give n *adjacency lists*, one for each vertex x of D, listing the vertices y such that (x, y) is an arc. The n lists are called an *adjacency structure*. An adjacency structure requires $(n+a)t$ bits of storage, where t is as for the arc list. This is because the adjacency list for a vertex x must be encoded by encoding x and then encoding all the y's so that x is adjacent to y.

As we have already pointed out, the best way to input a digraph or a graph into a computer will depend on the application for which we use the input. To give a simple example, suppose that we ask the question: Is (u_i, u_j) an arc? This question

Table 3.6: An Adjacency Structure for a Digraph

Vertex x:	1	2	3	4	5	6
Vertices adjacent to x:	2, 4	1, 3, 4	2	1, 5, 6	3	

can be answered in one step from an adjacency matrix (look at the i, j entry). To answer it from an adjacency structure requires $\deg(u_i)$ steps in the worst case if u_j is the last vertex to which there is an arc from u_i. On the other hand, let us consider an algorithm that requires marking all vertices to which there is an arc from u_i. This requires n steps from an adjacency matrix (look at all entries in the ith row), but only $\deg(u_i)$ steps from an adjacency structure.

EXERCISES FOR SECTION 3.7

1. For each digraph of Figure 3.7, find:
   (a) Its adjacency matrix
   (b) An arc list
   (c) Two linear arrays that can be used to input the arcs
   (d) An adjacency structure
2. For each graph of Figure 3.23, find an incidence matrix.
3. Draw the digraph whose adjacency matrix is given by

$$\text{(a) } \mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}; \qquad \text{(b) } \mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

4. Find the digraph whose vertices are a, b, c, d, e, f and whose arc list is

$$\{(b, a), (f, a), (b, d), (e, c), (e, b), (c, a), (c, e), (a, d)\}.$$

5. Find the digraph whose vertices are a, b, c, d, e and whose arcs are encoded by the two linear arrays (a, b, c, c, d, d, e), (b, d, a, d, c, e, a).
6. Find the digraph whose adjacency structure is given by Table 3.6.
7. Calculate the adjacency matrix for each of the graphs of Figure 3.9. (The adjacency matrix is the adjacency matrix of the corresponding digraph.)
8. Find the point-set incidence matrix for graph G of Figure 3.76 if:
   (a) $S = V(G)$ and $\mathcal{F}$ = all spanning trees of G
   (b) $S = E(G)$ and $\mathcal{F}$ = all spanning trees of G

9. If D^c is the complementary digraph of D (Exercise 34, Section 3.2), what is $\mathbf{A}(D) + \mathbf{A}(D^c)$?

10. Suppose that it requires one step to scan an entry in a list or an array. Suppose that a digraph is stored as an adjacency matrix.

    (a) How many steps are required to mark all vertices x adjacent to a particular vertex y, that is, such that (x, y) is an arc?

    (b) How many steps are required to mark or count all arcs?

11. Repeat Exercise 10 if the digraph is stored as an arc list.

12. Repeat Exercise 10 if the digraph is stored as an adjacency structure.

13. If D is a digraph of n vertices, its *reachability matrix* is an $n \times n$ matrix $\mathbf{R}$ whose i, j entry r_{ij} is 1 if vertex j is reachable from vertex i by a path, and 0 otherwise. For each digraph of Figure 3.7, find its reachability matrix. (Note that i is always reachable from i.)

14. What is the relationship between the reachability matrix of D, $\mathbf{R}(D)$, and $\mathbf{R}(D^c)$?

15. Show that D is strongly connected if and only if its reachability matrix (Exercise 13) is $\mathbf{J}$, the matrix of all 1's.

16. If D is a digraph with adjacency matrix $\mathbf{A}$, show by induction on k that the i, j entry of $\mathbf{A}^k$ gives the number of paths of length k in D that lead from i to j.

17. If G is a graph with adjacency matrix $\mathbf{A}$ (Exercise 7), what is the interpretation in graph (as opposed to digraph) language of the i, j entry of $\mathbf{A}^k$ (Exercise 16)?

18. Suppose that a square 0-1 symmetric matrix has 0's on its diagonal. Is it necessarily the adjacency matrix of some graph?

19. For digraphs D_1, D_2, and D_7 of Figure 3.7, use the result of Exercise 16 to find the number of paths of length 3 from u to v. Identify the paths.

20. If D is a digraph, its *incidence matrix* has rows corresponding to vertices and columns to arcs, with i, j entry equal to 1 if j is the arc (i, k) for some k, -1 if j is the arc (k, i) for some k, and 0 otherwise.

    (a) Find the incidence matrix for each digraph of Figure 3.7.

    (b) How many bits of storage are required for the incidence matrix?

    (c) If $\mathbf{M} = (m_{ij})$ is the incidence matrix of digraph D, what is the significance of the matrix $\mathbf{N} = (n_{ij})$, where $n_{ij} = \sum_k m_{ik} m_{jk}$?

    (d) If D is strongly connected, can we ever get away with fewer bits of storage than are needed for the adjacency matrix?

21. If $\mathbf{M} = (m_{ij})$ is any matrix of nonnegative entries, let $B(\mathbf{M})$ be the matrix whose i, j entry is 1 if $m_{ij} > 0$ and 0 if $m_{ij} = 0$. Show that if D is a digraph of n vertices with reachability matrix $\mathbf{R}$ and adjacency matrix $\mathbf{A}$, and $\mathbf{I}$ is the identity matrix, then:

    (a) $\mathbf{R} = B[\mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \cdots + \mathbf{A}^{n-1}]$ (b) $\mathbf{R} = B[(\mathbf{I} + \mathbf{A})^{n-1}]$

22. Check the results of Exercise 21 on the digraphs D_1 and D_7 of Figure 3.7.

23. (a) Show that D is unilaterally connected (Exercise 9, Section 3.2) if and only if $B(\mathbf{R} + \mathbf{R}^T) = \mathbf{J}$, where B is defined in Exercise 21, $\mathbf{R}$ in Exercise 13, and $\mathbf{J}$ in Exercise 15, and where $\mathbf{R}^T$ is the transpose of $\mathbf{R}$.

    (b) Show that D is weakly connected (Exercise 10, Section 3.2) if and only if we have $B[(\mathbf{I} + \mathbf{A} + \mathbf{A}^T)^{n-1}] = \mathbf{J}$.

24. Suppose that D is a digraph with reachability matrix (Exercise 13) $\mathbf{R} = (r_{ij})$ and that $\mathbf{R}^2$ is the matrix (s_{ij}). Show that:

    (a) The strong component (Exercise 13, Section 3.2) containing vertex i is given by the entries of 1 in the ith row of $\mathbf{T} = (t_{ij})$, where $t_{ij} = r_{ij} \times r_{ij}^{(T)}$ and $r_{ij}^{(T)}$ is the i, j entry of the transpose of $\mathbf{R}$.

    (b) The number of vertices in the strong component containing i is s_{ii}.

25. For each digraph of Figure 3.7, use the results of Exercise 24 to find the strong components.

26. If $\mathbf{R}$ is the reachability matrix of a digraph and $c(i)$ is the ith column sum of $\mathbf{R}$, what is the interpretation of $c(i)$?

27. If G is a graph, how would you define directly its reachability matrix $\mathbf{R}(G)$?

28. If G is a graph, $\mathbf{R} = \mathbf{R}(G)$ is its reachability matrix (Exercise 27), and $\mathbf{T}$ is as defined in Exercise 24:

    (a) Show that $\mathbf{T} = \mathbf{R}$.

    (b) What is the interpretation of the 1, 1 entry of $\mathbf{R}^2$?

29. Suppose that $\mathbf{R}$ is a matrix of 0's and 1's with 1's down the diagonal (and perhaps elsewhere). Is $\mathbf{R}$ necessarily the reachability matrix of some digraph? (Give a proof or counterexample.)

30. (Harary [1969]) Suppose that $\mathbf{B}$ is the incidence matrix of a graph G and $\mathbf{B}^T$ is the transpose of $\mathbf{B}$. What is the significance of the i, j entry of the matrix $\mathbf{B}^T\mathbf{B}$?

31. (Harary [1969]) Let G be a graph. The *circuit matrix* $\mathbf{C}$ of G is the point-set incidence matrix with S the set of edges of G and $\mathcal{F}$ the family of circuits of G. Let $\mathbf{B}$ be the incidence matrix of G. Show that every entry of $\mathbf{BC}$ is $\equiv 0 (\text{mod } 2)$.

32. Can two graphs have the same incidence matrix and be nonisomorphic? Why?

33. Can two graphs have the same circuit matrix and be nonisomorphic? Why? What if every edge is on a circuit?

3.8 RAMSEY NUMBERS REVISITED

Ramsey theory and, in particular, Ramsey numbers $R(p, q)$ were introduced in Section 2.19.3. To study Ramsey numbers, it is convenient to look at them using graph theory. If G is a graph, its *complement* G^c is the graph with the same vertex set as G, and so that for all $a \neq b$ in $V(G)$, $\{a, b\} \in E(G^c)$ if and only if $\{a, b\} \notin E(G)$. In studying the Ramsey numbers $R(p, q)$, we shall think of a set S as the vertex set of a graph, and of a set of 2-element subsets of S as the edge set of this graph. Then, to say that a number N has the (p, q) Ramsey property means that whenever S is a set of N elements and we have a graph G with vertex set S, then if we divide the 2-element subsets of S into edges of G and edges of G^c (edges not in G), either there are p vertices all of which are joined by edges in G or there are q vertices all of which are joined by edges in G^c. Put another way, we have the following theorem.

Theorem 3.29 A number N has the (p, q) Ramsey property if and only if for every graph G of N vertices, either G has a complete p-gon (complete subgraph of p vertices, K_p) or G^c has a complete q-gon.

Corollary 3.29.1 $R(p, 2) = p$ and $R(2, q) = q$.

Proof. For every graph of p vertices, either it is complete or its complement has an edge. This shows that $R(p, 2) \leq p$. Certainly, $R(p, 2) \geq p$. (Why?) Q.E.D.

In the language of Theorem 3.29, Theorem 2.18 says that if G is a graph of 6 (or more) vertices, then either G has a triangle or G^c has a triangle. (The reader should try this out on a number of graphs of his or her choice.) More generally, every graph of (at least) $R(p, q)$ vertices has either a complete p-gon (K_p) or its complement has a complete q-gon (K_q).

Consider now the graph $G = Z_5$, the circuit of length 5. Now G^c is again (isomorphic to) Z_5. Thus, neither G nor G^c has a triangle. We conclude that the number 5 does not have the $(3, 3)$ Ramsey property, so $R(3, 3) > 5$. This completes the proof of the following theorem.

Theorem 3.30 $R(3, 3) = 6$.

To use the terminology of Section 3.3.1, a complete p-gon in G is a clique of p vertices. A complete q-gon in G^c corresponds to an independent set of q vertices in G. Thus, we can restate Theorem 3.29 as follows.

Theorem 3.31 A number N has the (p, q) Ramsey property if and only if for every graph G of N vertices, either G has a clique of p vertices or G has an independent set of q vertices.

There is still another way to restate Theorem 3.29, which is as follows.

Theorem 3.32 A number N has the (p, q) Ramsey property if and only if whenever we color the *edges* of K_N, the complete graph on N vertices, with each edge being colored either red or blue, then K_N has a complete red p-gon or K_N has a complete blue q-gon.

Proof. Given an edge coloring, let G be the graph whose vertices are the same as those of K_N and whose edges are the red edges. Q.E.D.

We have already observed that Ramsey numbers are difficult to compute. The few Ramsey numbers that have been determined are given in Table 2.12. To verify (at least partly) some of the numbers in Table 2.12, one can derive some bounds

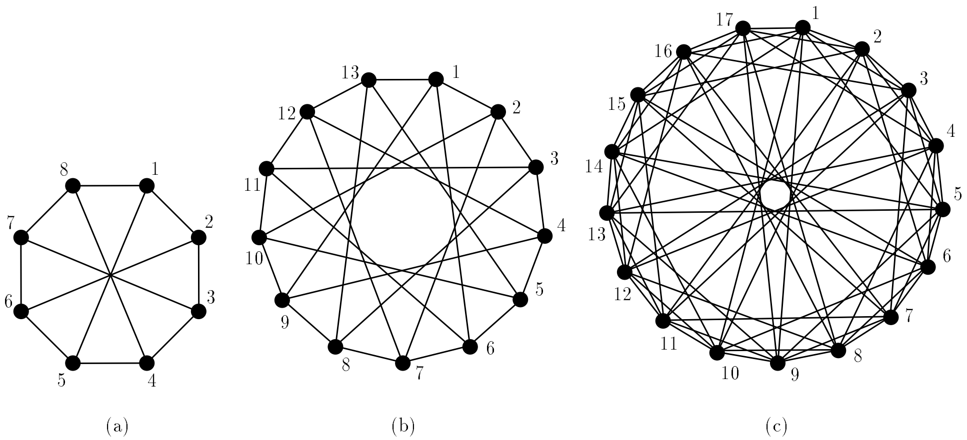


Figure 3.96: Graphs that demonstrate (a) $R(3,4) \geq 9$, (b) $R(3,5) \geq 14$, and (c) $R(4,4) \geq 18$.

on the Ramsey numbers. Consider graph (a) of Figure 3.96. This graph has 8 vertices. It also has no triangle (3-gon) and no independent set of 4 vertices. Thus, by Theorem 3.31, the number 8 does not have the $(3,4)$ Ramsey property. It follows that $R(3,4) \geq 9$. Similarly, graphs (b) and (c) of Figure 3.96 show that $R(3,5) \geq 14$ and $R(4,4) \geq 18$ (Exercises 2 and 3).

Finally, variants of Ramsey numbers present difficult challenges in graph theory. For discussion of one such topic, that of graph Ramsey numbers, see Exercises 11–13.

EXERCISES FOR SECTION 3.8

1. Show that $R(p,2) \geq p$.
2. Use Figure 3.96(b) to show that $R(3,5) \geq 14$.
3. Use Figure 3.96(c) to show that $R(4,4) \geq 18$.
4. Let G be a complete graph on 25 vertices and let the edges of G be colored either brown or green. If there is no green triangle, what is the largest complete brown m-gon you can be sure G has?
5. For each of the graphs of Figure 3.97, either find a clique of 3 vertices or an independent set of 3 vertices, or conclude that neither of these can be found.
6. Let G be any graph of 11 vertices and chromatic number 3.
   (a) Does G necessarily have either a clique of 3 vertices or an independent set of 3 vertices?
   (b) Does G necessarily have either a clique of 4 vertices or an independent set of 3 vertices?
7. Let G be a graph of 16 vertices and largest clique of size 3.

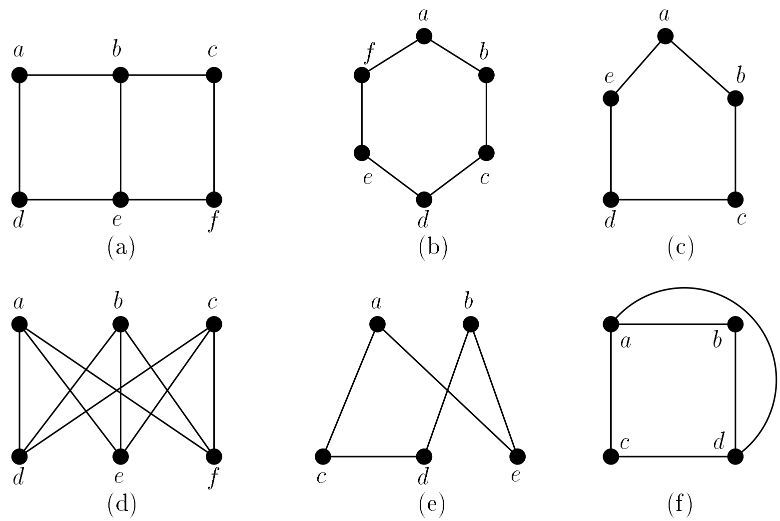


Figure 3.97: Graphs for Exercise 5, Section 3.8.

(a) Does G necessarily have an independent set of 4 vertices?

(b) Of 5 vertices?

8. Color the edges of K_{10} either red or blue.

   (a) Show that if there are at least 4 red edges from one vertex, there are 3 vertices all joined by red edges or 4 vertices all joined by blue edges.

   (b) Similarly, if there are at least 6 blue edges from a vertex, show that either there are 3 vertices all joined by red edges or 4 vertices all joined by blue edges.

   (c) Show that by parts (a) and (b), K_{10} has in any coloring of its edges with red or blue colors either 3 vertices all joined by red edges or 4 vertices all joined by blue edges.

   (d) Does part (c) tell you anything about a Ramsey number?

9. Let G be a tree of 20 vertices.

   (a) Does G necessarily have an independent set of 5 vertices?

   (b) Of 6 vertices?

10. Color the edges of the graph K_{17} in red, white, or blue. This exercise will argue that there are 3 vertices all joined by edges of the same color.

    (a) Fix one vertex a. Show that of the edges joining this vertex, at least 6 must have the same color.

    (b) Suppose that the 6 edges in (a) are all red. These lead from a to 6 vertices, b, c, d, e, f, and g. Argue from here that either K_{17} has a red triangle, a blue triangle, or a white triangle.

(c) What does the result say about the Ramsey numbers $R(p, q; r)$ defined in Exercise 33 of Section 2.19.

11. Let G_1 and G_2 be graphs. An integer N is said to have the *graph Ramsey property* (G_1, G_2) if every coloring of the edges of the complete graph K_N in the colors 1 and 2 gives rise, for some i, to a subgraph that is (isomorphic to) G_i and is colored all in color i, that is, to a monochromatic G_i, for $i = 1$ or 2. The *graph Ramsey number* $R(G_1, G_2)$ is the smallest N with the graph Ramsey property (G_1, G_2). (It is not hard to show that this is well defined. See Chartrand and Lesniak [1996] or Graham, Rothschild, and Spencer [1990].) If L_p is the chain of p vertices and Z_q is the circuit of q vertices:

    (a) Show that $R(L_3, L_3) = 3$.
    (b) Show that $R(L_4, L_4) = 5$.
    (c) Find $R(L_3, L_4)$.
    (d) Find $R(L_3, Z_4)$.
    (e) Find $R(L_4, Z_4)$.
    (f) Find $R(Z_4, Z_4)$.

12. (Chvátal and Harary [1972]) Let $c(G)$ be the size of the largest connected component of G. Show that
$$R(G, H) \geq (\chi(G) - 1)(c(H) - 1) + 1.$$

13. (Chvátal [1977]) If T_m is a tree on m vertices, show that
$$R(T_m, K_n) = 1 + (m - 1)(n - 1).$$

REFERENCES FOR CHAPTER 3

Agarwala, R., and Fernandez-Baca, D., "A Polynomial-time Algorithm for the Perfect Phylogeny Problem When the Number of Character States Is Fixed," *SIAM J. Comput., 23* (1994), 1216–1224.

Agarwala, R., Bafna, V., Farach, M., Narayanan, B., Paterson, M., and Thorup, M., "On the Approximability of Numerical Taxonomy: Fitting Distances by Tree Metrics," *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1996.

Aho, A. V., Hopcroft, J. E., and Ullman, J. D., *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B., *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.

Alon, N., "Restricted Colorings of Graphs," in K. Walker (ed.), *Surveys in Combinatorics*, London Mathematical Society Lecture Note Series, 187, Cambridge University Press, Cambridge, 1993, 1–33.

Appel, K., and Haken, W., "Every Planar Map Is Four Colorable. Part I: Discharging," *Ill. J. Math., 21* (1977), 429–490.

Appel, K., Haken, W., and Koch, J., "Every Planar Map Is Four Colorable. Part II: Reducibility," *Ill. J. Math., 21* (1977), 491–567.

Baase, S., *Computer Algorithms*, 2nd ed., Addison-Wesley Longman, Reading, MA, 1992.

Balaban, A. T. (ed.), *Chemical Applications of Graph Theory*, Academic Press, New York, 1976.

Beltrami, E. J., and Bodin, L. D., "Networks and Vehicle Routing for Municipal Waste Collection," *Networks, 4* (1973), 65–94.

Berge, C., "Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind," *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg, Math.-Naturwiss. Reihe, 10* (1961), 114.

Berge, C., "Sur une conjecture relative au problème des codes optimaux," *Commun. 13ème Assemblèe Gènèrale de l'URSI* (International Scientific Radio Union), Tokyo, 1962.

Bern, M. W., and Graham, R. L., "The Shortest-Network Problem," *Scientific American, 260* (1989), 84–89.

Birkhoff, G. D., "A Determinant Formula for the Number of Ways of Coloring a Map," *Ann. Math., 14* (1912), 42–46.

Birkhoff, G. D., and Lewis, D. C., "Chromatic Polynomials," *Trans. Amer. Math. Soc., 60* (1946), 355–451.

Bodlaender, H. L., "Polynomial Algorithms for Graph Isomorphism and Chromatic Index on Partial k-Trees," *J. Algorithms, 11* (1990), 631–643.

Bodlaender, H. L., Fellows, M., and Warnow, T., "Two Strikes Against Perfect Phylogeny," *Proc. of the 19th Int. Colloq. on Automata, Languages, and Programming*, (1992), 273–283.

Bondy, J. A., and Murty, U. S. R., *Graph Theory with Applications*, Elsevier, New York, 1976.

Bonias, I., "T-colorings of Complete Graphs," Ph.D. thesis, Northeastern University, 1991.

Booth, K. S., and Lueker, G. S., "Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-tree Algorithms," *J. Comp. Syst. Sci., 13* (1976), 335–379.

Brassard, G., and Bratley, P., *Algorithmics: Theory and Practice*, Prentice Hall, Upper Saddle River, NJ, 1995.

Brigham, R. C., and Dutton, R. D., "Generalized k-Tuple Colorings of Cycles and Other Graphs," *J. Comb. Theory, Series B, 32* (1982), 90–94.

Buneman, P., "The Recovery of Trees from Measures of Dissimilarity," in F. R. Hodson, D. G. Kendall, and P. Tautu (eds.), *Mathematics in the Archaeological and Historical Sciences*, Edinburgh University Press, Edinburgh, 1971, 387–395.

Cayley, A., "On the Theory of the Analytical Forms Called Trees," *Philos. Mag., 13* (1857), 172–176. [Also *Math. Papers*, Cambridge, *3* (1891), 242–246.]

Cayley, A., "On the Mathematical Theory of Isomers," *Philos. Mag., 67* (1874), 444–446. [Also *Math. Papers*, Cambridge, *9* (1895), 202–204.]

Cayley, A., "A Theorem on Trees," *Quart. J. Math., 23* (1889), 376–378. [Also *Math. Papers*, Cambridge, *13* (1897), 26–28.]

Chartrand, G., and Lesniak, L., *Graphs and Digraphs*, 3rd ed., CRC Press, Boca Raton, 1996.

Chudnovsky, M., Robertson, N., Seymour, P. D., and Thomas, R., "The Strong Perfect Graph Theorem," manuscript, (2002).

Chvátal, V., "Tree-Complete Graph Ramsey Numbers," *J. Graph Theory, 1* (1977), 93.

Chvátal, V., and Harary, F., "Generalized Ramsey Theory for Graphs," *Bull. Amer. Math. Soc., 78* (1972), 423–426.

Cieslik, D., *Steiner Minimal Trees*, Kluwer Academic Publishers, Norwell, MA, 1998.

Cohen, J. E., *Food Webs and Niche Space*, Princeton University Press, Princeton, NJ, 1978.

Cohen, J., and Farach, M., "Numerical Taxonomy on Data: Experimental Results," *J. Comput. Biol., 4* (1997), 547–558.

Cozzens, M. B., and Roberts, F. S., "T-Colorings of Graphs and the Channel Assignment Problem," *Congressus Numerantium, 35* (1982), 191–208.

Demourcron, G., Malgrance, V., and Pertuiset, R., "Graphes planaires: reconnaissance et construction des représentations planaires topologiques," *Rev. Française Recherche Opérationnelle, 8* (1964), 33–47.

Deo, N., *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall, Englewood Cliffs, NJ, 1974.

Erdős, P., Rubin, A. L., and Taylor, H., "Choosability in Graphs," *Congressus Numerantium, 26* (1979), 125–157.

Erdős, P. L., Steel, M. A., Székely, L. A., and Warnow, T. J., "Constructing Big Trees from Short Sequences," in P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela (eds.), *ICALP'97, 24th International Colloquium on Automata, Languages, and Programming (Silver Jubilee of EATCS)*, Bologna, Italy, July 7–11, 1997, *Lecture Notes in Computer Science*, Vol. 1256, Springer-Verlag, Berlin, 1997, 827–837.

Erdős, P. L., Steel, M. A., Székely, L. A., and Warnow, T. J., "A few logs suffice to build (almost) all trees. I," *Random Structures Algorithms, 14* (1999), 153–184.

Estabrook, G., Johnson, C., and McMorris, F. R., "An Idealized Concept of the True Cladistic Character," *Math. Bioscience, 23* (1975), 263–272.

Estabrook, G., Johnson, C. and McMorris, F. R., "A Mathematical Foundation for the Analysis of Cladistic Character Compatibility," *Math. Bioscience, 29* (1976), 181–187. (a)

Estabrook, G., Johnson, C., and McMorris, F. R., "An Algebraic Analysis of Cladistic Characters," *Discrete Math., 16* (1976), 141–147. (b)

Euler, L., "Solutio Problematis ad Geometriam Situs Pertinentis," *Comment. Acad. Sci. 1 Petropolitanae, 8* (1736), 128–140. [Reprinted in *Opera Omnia*, Series 1–7 (1766), 1–10.]

Even, S., *Graph Algorithms*, Computer Science Press, Potomac, MD, 1979.

Filliben, J. J., Kafadar, K., and Shier, D. R., "Testing for Homogeneity of Two-Dimensional Surfaces," *Math Modeling, 4* (1983), 167–189.

Fiorini, S., and Wilson, R. J., *Edge Colorings of Graphs*, Pitman, London, 1977.

Fitch, W. M., "An Introduction to Molecular Biology for Mathematicians and Computer Programmers," in M. Farach-Colton, F. S. Roberts, M. Vingron, and M. S. Waterman (eds.), *Mathematical Support for Moleclar Biology*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 47, American Mathematical Society, Providence, RI, 1999, 1–31.

Frank, A., "Connectivity and Network Flows," in R. L. Graham, M. Grötschel, and L. Lovász (eds.), *Handbook of Combinatorics*, Elsevier, Amsterdam, 1995, 111–177.

Garey, M. R., and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

Geller, D. P., "r-Tuple Colorings of Uniquely Colorable Graphs," *Discrete Mathematics, 16* (1976), 9–12.

Gibbons, A., *Algorithmic Graph Theory*, Cambridge University Press, Cambridge, 1985.

Gilbert, E. N., Unpublished Technical Memorandum, Bell Telephone Labs, Murray Hill, NJ, 1972.

Goldman, A. J., "Discrete Mathematics in Government," lecture presented at SIAM Symposium on Applications of Discrete Mathematics, Troy, NY, June 1981.

Golumbic, M. C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New

York, 1980.

GRAHAM, R. L., and HELL, P., "On the History of the Minimum Spanning Tree Problem," *Annals of the History of Computing, 7* (1985), 43–57.

GRAHAM, R. L., ROTHSCHILD, B. L., and SPENCER, J. H., *Ramsey Theory*, 2nd ed., Wiley, New York, 1990.

GRAVIER, S., "Coloration et Produits de Graphes," Ph.D. thesis, Université Joseph Fourier, Grenoble, France, 1996.

GRIMALDI, D. A., *A Phylogenetic, Revised Classification of Genera in the Drosophilidae (Diptera)*, Bulletin of the American Museum of Natural History, American Museum of Natural History, New York, 1990.

GUSFIELD, D., "Efficient Algorithms for Inferring Evolutionary History," *Networks, 21* (1991), 19–28.

GUSFIELD, D., *Algorithms on Strings, Trees, and Sequences*, Cambridge University Press, New York, 1997.

HALE, W. K., "Frequency Assignment: Theory and Applications," *Proc. IEEE, 68* (1980), 1497–1514.

HALIN, R., "Bemerkungen Über Ebene Graphen," *Math. Ann., 53* (1964), 38–46.

HANSEN, P., FOWLER, P., and ZHENG, M. (eds.), *Discrete Mathematical Chemistry*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 51, American Mathematical Society, Providence, RI, 2000.

HARARY, F., *Graph Theory*, Addison-Wesley, Reading, MA, 1969.

HARARY, F., NORMAN, R. Z., and CARTWRIGHT, D., *Structural Models: An Introduction to the Theory of Directed Graphs*, Wiley, New York, 1965.

HARARY, F., and PALMER, E. M., *Graphical Enumeration*, Academic Press, New York, 1973.

HARARY, F., and TUTTE, W. T., "A Dual Form of Kuratowski's Theorem," *Canad. Math. Bull., 8* (1965), 17–20.

HARRISON, J. L., "The Distribution of Feeding Habits among Animals in a Tropical Rain Forest," *J. Anim. Ecol., 31* (1962), 53–63.

HOPCROFT, J. E., and TARJAN, R. E., "Efficient Planarity Testing," *J. ACM, 21* (1974), 549–568.

HUFFMAN, D. A., "A Method for the Construction of Minimum Redundancy Codes," *Proc. Inst. Rail. Eng., 40* (1952), 1098–1101.

IRVING, R. W., "NP-completeness of a Family of Graph-Colouring Problems," *Discrete Appl. Math., 5* (1983), 111–117.

JENSEN, T. R., and TOFT, B., *Graph Coloring Problems*, Wiley, New York, 1995.

KANG, A. N. C., LEE, R. C. T., CHANG, C. L., and CHANG, S. K., "Storage Reduction through Minimal Spanning Trees and Spanning Forests," *IEEE Trans. Comput., C-26* (1977), 425–434.

KANNAN, S., and WARNOW, T., "Inferring Evolutionary History from DNA Sequences," *SIAM J. Comput., 23* (1994), 713–737.

KANNAN, S., and WARNOW, T., "A Fast Algorithm for the Computation and Enumeration of Perfect Phylogenies When the Number of Character States Is Fixed," *6th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1995, 595–603.

KELLY, J. B., and KELLY, L. M., "Paths and Circuits in Critical Graphs," *Amer. J. Math., 76* (1954), 786–792.

KEMENY, J. G., and SNELL, J. L., *Mathematical Models in the Social Sciences*, Blaisdell, New York, 1962. (Reprinted by MIT Press, Cambridge, MA, 1972.)

KIRCHHOFF, G., "Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Ströme geführt wird," *Ann. Phys. Chem., 72* (1847), 497–508.

KLOTZ, W., "A Constructive Proof of Kuratowski's Theorem," *Ars Combinatoria, 28* (1989), 51–54.

KNUTH, D. E., *The Art of Computer Programming*, Vol. 3: *Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.

KÖNIG, D., *Theorie des endlichen und unendlichen Graphen*, Akademische Verlagsgesellschaft, Leipzig, 1936. (Reprinted by Chelsea, New York, 1950.)

KRATOCHVÍL, J., TUZA, Z., and VOIGT, M., "New Trends in the Theory of Graph Colorings: Choosability and List Coloring," in R. L. Graham, J. Kratochvíl, J. Nešetřil, and F. S. Roberts (eds.), *Contemporary Trends in Discrete Mathematics*, DIMACS Series, 49, American Mathematical Society, Providence, RI, 1999, 183–195.

KREHER, D. L., and STINSON, D. R., *Combinatorial Algorithms: Generation, Enumeration, and Search*, CRC Press, Boca Raton, FL, 1998.

KURATOWSKI, K., "Sur le Problème des Courbes Gauches en Topologie," *Fund. Math., 15* (1930), 271–283.

LEHMER, D. H., "The Chromatic Polynomial of a Graph," *Pacific J. Math., 118* (1985), 463–469.

LIU, C. L., *Topics in Combinatorial Mathematics*, Mathematical Association of America, Washington, DC, 1972.

LIU, D.-F., "Graph Homomorphisms and the Channel Assignment Problem," Ph.D. thesis, University of South Carolina, 1991.

LOVÁSZ, L., "Normal Hypergraphs and the Perfect Graph Conjecture," *Discrete Math., 2* (1972), 253–267. (a)

LOVÁSZ, L., "A Characterization of Perfect Graphs," *J. Comb. Theory B, 13* (1972), 95–98. (b)

LUKS, E. M., "Isomorphism of Graphs of Bounded Valence Can Be Tested in Polynomial Time," *J. Comput. System Sci., 25* (1982), 42–65.

MACKENZIE, D., "Graph Theory Uncovers the Roots of Perfection," *Science, 297* (2002), 38.

MAHADEV, N. V. R., and ROBERTS, F. S., "Consensus List Colorings of Graphs and Physical Mapping of DNA," in M. Janowitz, F. R. McMorris, B. Mirkin, and F. S. Roberts (eds.), *Bioconsensus*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 61, American Mathematical Society, Providence, RI, 2003, 83–95.

MAKARYCHEV, Y., "A Short Proof of Kuratowski's Graph Planarity Criterion," *J. Graph Theory, 25* (1997), 129–131.

MANBER, U., *Introduction to Algorithms: A Creative Approach*, Addison-Wesley Longman, Reading, MA, 1989.

MCKEE, T. A., and BEINEKE, L. W., *Graph Theory in Computer Science, Chemistry, and Other Fields*, Pergamon Press, Exeter, UK, 1997.

MOON, J. W., "Various Proofs of Cayley's Formula for Counting Trees," in F. Harary (ed.), *A Seminar on Graph Theory*, Holt, Rinehart and Winston, New York, 1967, 70–78.

OPSUT, R. J., and ROBERTS, F. S., "On the Fleet Maintenance, Mobile Radio Frequency, Task Assignment, and Traffic Phasing Problems," in G. Chartrand, Y. Alavi, D. L. Goldsmith, L. Lesniak-Foster, and D. R. Lick (eds.), *The Theory and Applications of Graphs*, Wiley, New York, 1981, 479–492.

PESCHON, J., and ROSS, D., "New Methods for Evaluating Distribution, Automation, and Control (DAC) Systems Benefits," *SIAM J. Algebraic Discrete Methods, 3* (1982), 439–452.

Ponomarenko, I. N., "A Polynomial Isomorphism Algorithm for Graphs Not Contractible to $K_{3,g}$ (Russian; English summary)," *Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI), 137* (1984), 99–114.

Ponomarenko, I. N., "Polynomial Time Algorithms for Recognizing and Isomorphism Testing of Cyclic Tournaments," *Acta Appl. Math., 29* (1992), 139–160.

Prim, R. C., "Shortest Connection Networks and Some Generalizations," *Bell Syst. Tech. J., 36* (1957), 1389–1401.

Raychaudhuri, A., "Intersection Assignments, T-Coloring, and Powers of Graphs," Ph.D. thesis, Rutgers University, 1985.

Read, R. C., "An Introduction to Chromatic Polynomials," *J. Comb. Theory, 4* (1968), 52–71.

Reingold, E. M., Nievergelt, J., and Deo, N., *Combinatorial Algorithms: Theory and Practice*, Prentice Hall, Englewood Cliffs, NJ, 1977.

Roberts, F. S., *Discrete Mathematical Models, with Applications to Social, Biological, and Environmental Problems*, Prentice Hall, Englewood Cliffs, NJ, 1976.

Roberts, F. S., *Graph Theory and Its Applications to Problems of Society*, NSF-CBMS Monograph No. 29, SIAM, Philadelphia, 1978.

Roberts, F. S., "Indifference and Seriation," *Ann. N.Y. Acad. Sci., 328* (1979), 173–182.

Roberts, F. S., "From Garbage to Rainbows: Generalizations of Graph Coloring and their Applications," in Y. Alavi, G. Chartrand, O. R. Oellermann, and A. J. Schwenk (eds.), *Graph Theory, Combinatorics, and Applications*, Vol. 2, Wiley, New York, 1991, 1031–1052.

Robertson, N., Sanders, D. P., Seymour, P. D., and Thomas, R., "The Four Colour Theorem," *J. Comb. Theory, Series B, 70* (1997), 2–44.

Rouvray, D. H., and Balaban, A. T., "Chemical Applications of Graph Theory," in R. J. Wilson and L. W. Beinecke (eds.), *Applications of Graph Theory*, Academic Press, London, 1979, 177–221.

Scott, S. H., "Multiple Node Colourings of Finite Graphs," doctoral dissertation, University of Reading, England, March 1975.

Shier, D. R., "Testing for Homogeneity using Minimum Spanning Trees," *UMAP J., 3* (1982), 273–283.

Shor, P. W., "A New Proof of Cayley's Formula for Counting Labeled Trees," *J. Combin. Theory, Ser. A, 71* (1995), 154–158.

Steel, M. A., "The Complexity of Reconstructing Trees from Qualitative Characters and Subtrees," *J. Classification, 9* (1992), 91–116.

Takács, L., "On Cayley's Formula for Counting Forests," *J. Combin. Theory, Ser. A, 53* (1990), 321–323.

Tesman, B. A., "Complete Graph T-Spans," *Congressus Num., 35-A* (1993), 161–173.

Tucker, A. C., "Perfect Graphs and an Application to Optimizing Municipal Services," *SIAM Rev., 15* (1973), 585–590.

Tucker, A. C., *Applied Combinatorics*, 2nd ed., Wiley, New York, 1984.

Tutte, W. T. (alias B. Descartes), "Solution to Advanced Problem No. 4526," *Amer. Math. Monthly, 61* (1954), 352.

van den Heuvel, J., Leese, R. A., and Shepherd, M. A., "Graph Labeling and Radio Channel Assignment," *J. Graph Theory, 29* (1998), 263–283.

Wagner, K., "Über Eine Eigenschaft der Ebene Komplexe," *Math. Ann., 114* (1937),

570–590.

WANG, D.-I., "The Channel Assignment Problem and Closed Neighborhood Containment Graphs," Ph.D. thesis, Northeastern University, 1985.

WELSH, D., and WHITTLE, G. P., "Arrangements, Channel Assignments, and Associated Polynomials," *Adv. Appl. Math., 23* (1999), 375–406.

WEST, D. B., *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River, NJ, 2001.

WHITNEY, H., "The Coloring of Graphs," *Ann. Math., 33* (1932), 688–718.

WILKINSON, E. M., "Archaeological Seriation and the Traveling Salesman Problem," in F. R. Hodson, *et al.* (eds.), *Mathematics in the Archaeological and Historical Sciences*, Edinburgh University Press, Edinburgh, 1971.

ZYKOV, A. A., "On Some Properties of Linear Complexes (Russian)," *Mat. Sbornik N.S., 24* (1949), 163–188. (English transl.: *Amer. Math. Soc. Transl., 1952*, (1952), 33 pp.)

Chapter 4

Relations[1]

4.1 RELATIONS

A fundamental idea in science as well as in everyday life is to see how two objects, items, or alternatives are related. We might say that a is bigger than b, a is louder than b, a is a brother of b, a is preferred to b, or a and b are equally talented. In this chapter we make precise the idea of a relation between objects, in particular a binary relation, and then note that the study of binary relations is closely related to the study of digraphs from Chapter 3. We pay special attention to those relations that define what are called order relations, and apply them to problems arising from such fields as computer science, economics, psychophysics, biology, and archaeology.

4.1.1 Binary Relations

Suppose that X and Y are sets. The *cartesian product* of X with Y, denoted $X \times Y$, is the set of all ordered pairs (a, b) where a is in X and b is in Y. A *binary relation* R on a set X is a subset of the cartesian product $X \times X$, that is, a set of ordered pairs (a_1, a_2) where a_1 and a_2 are in X. To emphasize the importance of the underlying set, we often speak of the binary relation (X, R) rather than just the binary relation R. If X is the set $\{1, 2, 3, 4\}$, examples of binary relations on X are given by

$$R = \{(1,1), (1,2), (2,1), (3,1), (3,2), (3,3), (3,4)\} \tag{4.1}$$

and

$$S = \{(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)\}. \tag{4.2}$$

Looking back to Section 3.1.2, we see that binary relations are defined exactly the same way as digraphs. Recall that a digraph was defined as a pair (V, A) where

[1] This chapter may be omitted. Section 3.1.2 is suggested as a prerequisite. Alternatively, this chapter may be skipped at this point and returned to later. Ideas from this chapter are needed only in selected examples in a few sections of the book, in particular in parts of Chapters 8, 12, and 13.

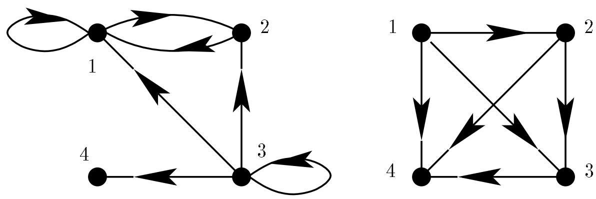


Figure 4.1: Digraph representations of binary relations (4.1) and (4.2).

V is a set and A is a set of ordered pairs of elements of V; this is just another way of saying that A is a binary relation on a set V; that is, A is a subset of the cartesian product $V \times V$. The digraphs corresponding to the binary relations defined from (4.1) and (4.2) can be seen in Figure 4.1. Note that since all digraphs in this book have finite vertex sets, we will only talk about digraphs of binary relations (X, R) for X a finite set.

In the case of a binary relation R on a set X, we shall usually write aRb to denote the statement that $(a, b) \in R$ or that there is an arc from a to b in the digraph of R. Thus, for example, if S is the relation[2] from (4.2), then $1S4$ and $2S3$ but not $3S1$. We shall also use $\sim aRb$ to denote the statement that (a, b) is not in R or that there is no arc from a to b in the digraph of R.

As the name suggests, a binary relation represents what it means for two elements to be related, and in what order. Binary relations arise very frequently from everyday language. For example, if X is the set of all people in the world, then the set

$$F = \{(a, b) : a \in X \text{ and } b \in X \text{ and } a \text{ is the father of } b\}$$

defines a binary relation on X, which we may call, by a slight abuse of language, "father of."

Example 4.1 Preference Suppose that X is any collection of alternatives among which you are choosing, for example, a menu of dinner items or a set of presidential candidates or a set of job candidates or a set of software packages. Suppose that

$$P = \{(a, b) \in X \times X : \text{ you strictly prefer } a \text{ to } b\}.$$

Then P may be called your relation of *strict preference* on the set X. Strict preference is to be distinguished from *weak preference*: The former means "better than" and the latter means "at least as good as." We will normally qualify preference as either being strict or weak. The relation (X, P) is widely studied in economics, political science, psychology, and other fields. To give a concrete example, suppose that you are considering preferences among alternative vacation destinations, your set of possible destinations is $X = \{$San Francisco, Los Angeles, New York, Boston, Miami, Atlanta, Phoenix$\}$, and your strict preference relation is given by

[2]We will often use the term "relation" to mean "binary relation." More generally, a relation is a subset of the cartesian product $X \times X \times \cdots \times X$.

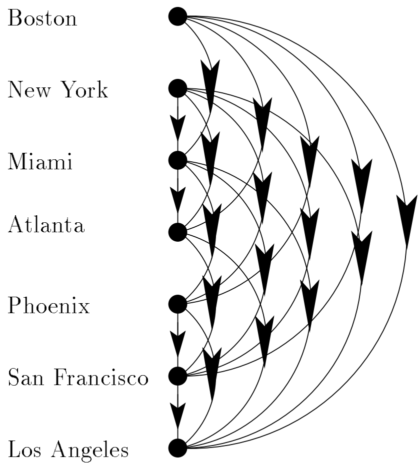


Figure 4.2: Preference digraph for data of Example 4.1.

$P = \{$(Boston, Atlanta), (Boston, Phoenix), (Boston, San Francisco), (Boston, Miami), (Boston, Los Angeles), (New York, Atlanta), (New York, Phoenix), (New York, San Francisco), (New York, Miami), (New York, Los Angeles), (San Francisco, Los Angeles), (Atlanta, San Francisco), (Atlanta, Los Angeles), (Miami, San Francisco), (Miami, Los Angeles), (Miami, Atlanta), (Miami, Phoenix), (Phoenix, Los Angeles), (Phoenix, San Francisco)$\}$. Thus, for example, you strictly prefer Miami to Atlanta. The digraph corresponding to this (X, P) is shown in Figure 4.2. ∎

Example 4.2 Psychophysical Scaling The study of the relationship between the physical properties of stimuli and their psychological properties is called *psychophysics*. In psychophysics, for instance, we try to relate the psychological response of loudness or brightness or sweetness to the physical properties of a sound, light, or food. (See Falmagne [1985] for an introduction to psychophysics from a mathematical point of view.) We often start by making comparisons. For example, if X is a set of sounds, such as coming from different airplanes at different distances from us, we might say that one "sounds louder than" another. If aLb means that "a sounds louder than b," then (X, L) is a binary relation. For example, let a be a Boeing 747 at 2000 feet, a' be a Boeing 747 at 3000 feet, b be a Boeing 757 at 2000 feet, b' be a Boeing 757 at 3000 feet, c be a Boeing 767 at 2000 feet, and c' be a Boeing 767 at 3000 feet. Suppose that

$$L = \{(a,a'),(a,b),(a,b'),(a,c),(a,c'),(a',b),(a',b'),(a',c'),(b,b'),(b,c'),(b',c'),(c,c')\}.$$

Then, for example, a sounds louder than c. The digraph corresponding to this (X, L) is shown in Figure 4.3. ∎

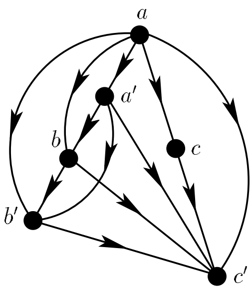


Figure 4.3: "Sounds louder than" digraph for psychophysical scaling. Digraph corresponds to relation (X, L) of Example 4.2.

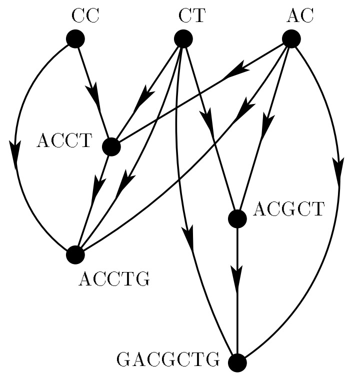


Figure 4.4: Digraph corresponding to relation (X, S) of Example 4.3.

Example 4.3 The Substring Problem In both biology and computer science, we deal with strings of symbols from some alphabet. We are often interested in whether one string appears as a consecutive substring of another string. This is very important in molecular biology, where we seek "patterns" in large molecular sequences such as DNA or RNA sequences, patterns being defined as small, consecutive substrings. We return to a related idea in Example 11.2 and Section 11.6.5.

Suppose that X is a collection of strings. Let us denote by aSb the observation that string a appears as a consecutive substring of string b. This defines a binary relation (X, S). To give a concrete example, let

$$X = \{\text{CC, CT, AC, ACCT, ACCTG, ACGCT, GACGCTG}\}.$$

Then we have (CC, ACCT) $\in S$, (ACCT, ACCTG) $\in S$, and so on. The corresponding digraph is shown in Figure 4.4.

The binary relation (X, S) plays an important role in computer science. One is often given a fixed set of strings and asked to determine whether a given string is a consecutive substring of any string in the set. The data structures known as suffix trees play an important role in solving this problem. See Gusfield [1997] for a discussion of this problem. ■

Example 4.4 Search Engines Web search engines, such as Google, use measures of relevance between a query and a web page. Thus, we start with a set X of web pages. Of course, because the Internet is constantly growing and changing, the set X is changing, but at any given instant, let us consider it fixed. One of the challenges for search engines is to find the set X at any given time. Let q be a query, x be a web page, and $r(x, q)$ be a measure of the relevance of web page x to query q. Another challenge for search engines is to figure out how to measure $r(x, q)$. Let us say that x is ranked over y if $r(x, q) > r(y, q)$. In this case, we write

Table 4.1: Web Pages x and Their Relevance r to a Query q

x	a	b	c	d	e	f
$r(x,q)$	7	8	5	7	8	4

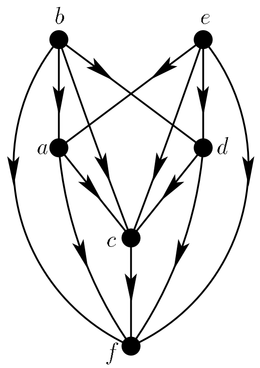


Figure 4.5: Digraph corresponding to relation (X,R) of Example 4.4.

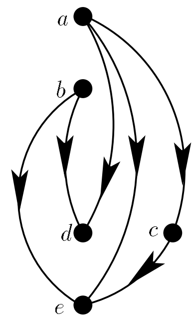


Figure 4.6: Precedence digraph for sequencing in archaeology. Digraph corresponds to relation (X,Q) of Example 4.5

xRy. Consider, for instance, the values of $r(x,q)$ in Table 4.1. Thus,

$$R = \{(b,a),(b,c),(b,d),(b,f),(e,a),(e,c),(e,d),(e,f),(a,c),(a,f),(d,c),(d,f),(c,f)\}.$$

The corresponding digraph is shown in Figure 4.5. ■

Example 4.5 Sequencing in Archaeology A common problem in many applied contexts involves placing items or individuals or events in a natural order based on some information about them. For instance, in archaeology, several types of pottery or other artifacts are found in different digs. We would like to place the artifacts in some order corresponding to when they existed in historical times. We know, for instance, that artifact a preceded artifact b in time. Can we reconstruct an order for the artifacts? This problem, known as the problem of *sequence dating* or *sequencing* or *seriation*, goes back to the work of Flinders Petrie [1899, 1901]. Some mathematical discussion of sequence dating can be found in Kendall [1963, 1969a,b] and Roberts [1976, 1979a]. To give a concrete example, suppose that X consists of five types of pottery, a, b, c, d, e, and we know that a preceded c, d, and e, b preceded d and e, and c preceded e. Then if xQy means that x preceded y, we have

$$Q = \{(a,c),(a,d),(a,e),(b,d),(b,e),(c,e)\}.$$

The digraph corresponding to (X,Q) is shown in Figure 4.6. ■

Table 4.2: Properties of Relations

A binary relation (X, R) is:	Provided that:
Reflexive	aRa, all $a \in X$
Nonreflexive	it is not reflexive
Irreflexive	$\sim aRa$, all $a \in X$
Symmetric	$aRb \Rightarrow bRa$, all $a, b \in X$
Nonsymmetric	it is not symmetric
Asymmetric	$aRb \Rightarrow \sim bRa$, all $a, b \in X$
Antisymmetric	aRb & $bRa \Rightarrow a = b$, all $a, b \in X$
Transitive	aRb & $bRc \Rightarrow aRc$, all $a, b, c \in X$
Nontransitive	it is not transitive
Negatively transitive	$\sim aRb$ & $\sim bRc \Rightarrow \sim aRc$, all $a, b, c \in X$ or $aRc \Rightarrow aRb$ or bRc, all $a, b, c \in X$
Strongly complete	aRb or bRa, all $a, b \in X$
Complete	aRb or bRa, all $a \neq b \in X$

4.1.2 Properties of Relations/Patterns in Digraphs

There are a number of properties that are common to many naturally occurring relations. In this section we discuss some of these properties and their representation when considering the digraphs of relations with these properties. These properties are summarized in Table 4.2.

A binary relation (X, R) is *reflexive* if for all $a \in X$, aRa. Thus, for example, if X is a set of numbers and R is the relation "equality" on X, then (X, R) is reflexive because a number is always equal to itself. However, if $X = \{1, 2, 3, 4\}$, the relation R from (4.1) is not reflexive, since $2R2$ (and $4R4$) does not hold. In this case the binary relation is called *nonreflexive*, which simply means "not reflexive." Again if $X = \{1, 2, 3, 4\}$, the relation S from (4.2) is nonreflexive, since $\sim 1S1$ (and $\sim 2S2$ and $\sim 3S3$ and $\sim 4S4$). When a binary relation is as nonreflexive as this relation, it is called *irreflexive*. That is, (X, R) is irreflexive if $\sim aRa$ for all $a \in X$. In this sense, the relation "father of" on a set of people is irreflexive. So are the relations (X, P), (X, L), (X, S), (X, R), (X, Q) of Examples 4.1, 4.2, 4.3, 4.4, and 4.5, respectively. What does this all mean with regard to digraphs? If a binary relation is reflexive, its digraph has loops at every vertex. In a nonreflexive relation, at least one loop is not present, and in an irreflexive relation, no loop is present. Thus, existence or nonexistence of these three properties can easily be discovered from the digraph of a relation. Consider the digraphs in Figure 4.7. We can quickly ascertain that digraph (a) is irreflexive and nonreflexive, digraph (b) is only nonreflexive, and digraph (c) is reflexive.

In Section 3.1.2 we defined a graph from a digraph by checking to see whether or not there is an arc from u to v whenever there is an arc from v to u. In the

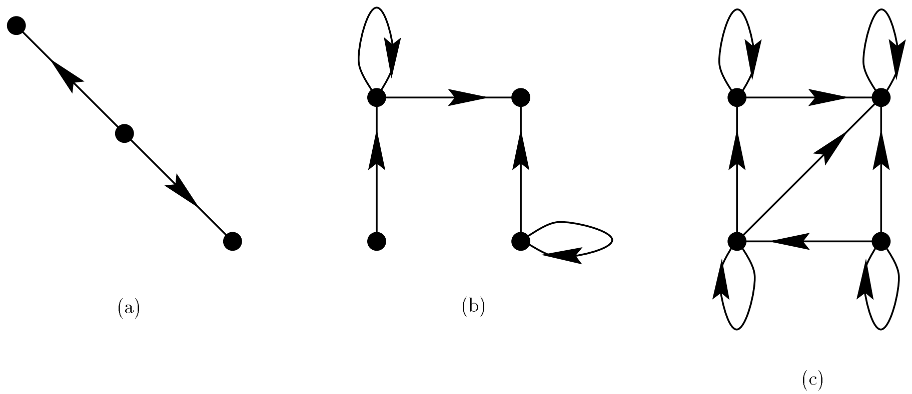


Figure 4.7: Examples of digraphs.

affirmative case we say the digraph is a graph and replace each pair of arcs between vertices by a single nondirected line and call it an edge. This condition of having an arc from u to v whenever there is an arc from v to u is exactly the condition of our next property. A binary relation (X, R) is called *symmetric* if for all $a, b \in X$,

$$aRb \Rightarrow bRa.$$

That is, (X, R) is symmetric if $(b, a) \in R$ whenever $(a, b) \in R$. So, by our discussion above, any graph represents a symmetric binary relation. (Note that a symmetric digraph may or may not have loops.) The relation "equality" on any set of numbers is symmetric. So is the relation "brother of" on the set of all males in the United States. However, the relation "brother of" on the set of all people in the United States is not symmetric, for if a is the brother of b, it does not necessarily follow that b is the brother of a. (Why?) This shows why it is important to speak of the underlying set when defining a relation and studying its properties.

Other examples of *nonsymmetric* (not symmetric) relations are the relation "father of" on the set of people of the world and the relations (X, P), (X, L), (X, S), (X, R), and (X, Q) of Examples 4.1, 4.2, 4.3, 4.4, and 4.5, respectively. These six relations are all highly nonsymmetric. In fact, they are called *asymmetric* because they satisfy the rule

$$aRb \Rightarrow \sim bRa.$$

Other asymmetric relations include the relation "greater than," $>$, on a set of real numbers, "strictly contained in," $\subsetneqq$, on any collection of sets, and the relation S from (4.2) on the set $X = \{1, 2, 3, 4\}$. What properties of the corresponding digraph capture the idea that a relation is asymmetric? One interpretation is that the digraph of an asymmetric relation will have no loops and that for all vertices u and v,

$$d(u, v) + d(v, u) \neq 2. \tag{4.3}$$

Some relations (X, R) are not quite asymmetric, but are almost asymmetric in the sense that loops are allowed but for vertices $u \neq v$, Equation (4.3) holds. Let us say that (X, R) is *antisymmetric* if for all $a, b \in X$,

$$aRb \ \& \ bRa \Rightarrow a = b.$$

So, an antisymmetric digraph is like an asymmetric digraph which allows loops. In many examples, being antisymmetric versus asymmetric means that "equality of elements" is allowed. For example, the relation "greater than or equal to," $\geq$, on a set of real numbers, "contained in," $\subseteqq$, on any collection of sets, and "at least as tall as" on any set of people no two of whom have the same height are three examples of antisymmetric relations. It is easy to show that every asymmetric binary relation is antisymmetric but the converse is false (Exercise 15).

A relation (X, R) is called *transitive* if for all $a, b, c \in X$, whenever aRb and bRc, then aRc. That is, (X, R) is transitive if for all $a, b, c \in X$,

$$aRb \ \& \ bRc \Rightarrow aRc.$$

Examples of transitive relations are the relations $=$ and $>$ on a set of real numbers, "implies" on a set of statements, and the relation (X, S) where $X = \{1, 2, 3, 4\}$ and S is given by (4.2). The relations (X, P), (X, L), (X, S), (X, R), and (X, Q) of Examples 4.1, 4.2, 4.3, 4.4, and 4.5, respectively, are all transitive. This is clearly the case for (X, S) and (X, R), which are defined very precisely, and it seems reasonable for the other three examples as well. Thus, it seems reasonable to assume that the relation of strict preference among alternative vacation destinations is always transitive, for if you prefer a to b and b to c, you should be expected to prefer a to c. Similarly, it seems reasonable to assume that the relation "sounds louder than" on a set of airplanes is always transitive and similarly for the relation "preceded" on a set of artifacts. As reasonable as these last three examples appear, only with empirical data may verification be obtained. In strict preferences arising in real applications, we sometimes find transitivity violated. If $X = \{1, 2, 3, 4\}$ and R is given by (4.1), then (X, R) is *nontransitive*, i.e, not transitive, because $2R1$ and $1R2$ but $\sim 2R2$. Another relation that is not transitive is the relation "father of" on the set of people in the world. How does one tell from a digraph whether or not its associated binary relation is transitive? To be transitive means that if there is an arc from vertex u to vertex v and an arc from vertex v to vertex w, then there will be an arc from vertex u to vertex w. Transitivity can also be defined by a restriction on the distance function for a digraph (see Exercise 28).

Our next property is similar to transitivity but in a negative sense. A binary relation (X, R) is called *negatively transitive* if for all $a, b, c \in X$, $\sim aRb$ and $\sim bRc$ imply that $\sim aRc$. A binary relation (X, R) is negatively transitive if the relation "not in R," defined on the set X, is transitive. To give an example, the relation $R =$ "greater than" on a set of real numbers is negatively transitive, for "not in R" is the relation "not greater than" or "less than or equal to," which is certainly transitive. It is easy to show that if $X = \{1, 2, 3, 4\}$, the relation S from (4.2) is negatively transitive. So are the relations (X, S) and (X, R) from Examples 4.3 and 4.4. Similarly, strict preference on a set of alternatives, "sounds louder than" on a set of sounds, and preceded on a set of artifacts are probably negatively transitive. (When

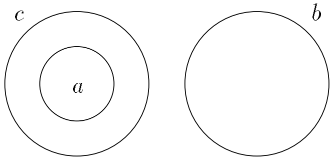


Figure 4.8: The relation "contained in" is not necessarily negatively transitive.

dealing with economics, political science, psychology, or psychophysics, terms like "probably" are often the best that can be expected.) Verifying negative transitivity can be annoyingly confusing. It is often easier to test the contrapositive (and hence equivalent) condition: For all $a, b, c \in X$, if aRc, then aRb or bRc. Using this notion, we see easily that the relation "greater than" is negatively transitive, for if $a > c$, then for all b, either $a > b$ or $b > c$. Similarly, one sees that "contained in" is not negatively transitive, for if a is contained in c, there may very well be a b so that a is not contained in b and b is not contained in c. (See Figure 4.8 for an example.) The relation "father of" on the set of all people in the world is not negatively transitive, nor is the relation (X, R) where $X = \{1, 2, 3, 4\}$ and R is given by (4.1). To see the latter, note that $2R1$ but not $2R4$ and not $4R1$.

Checking to see if a digraph is negatively transitive can be a nontrivial task. In the worst case, for every arc (u, v), two arcs must be searched for, (u, w) and (w, v), for every vertex w, $w \neq u, v$. Thus, it is possible that checking for negative transitivity could take $2a(n-2)$ searches, where n and a are the sizes of the vertex set and arc set of the digraph, respectively.

Let us say that a binary relation (X, R) is *strongly complete* if for all $a, b \in X$, aRb or bRa. Thus, $\geq$, "greater than or equal to," on a set of numbers is strongly complete. However, "strict containment" on a family of sets may not be as it is possible that for two sets, neither is strictly contained in the other. Similarly, "father of" on the set of all people in the world is not strongly complete. (Why?) The substring relation of Example 4.3 is also not strongly complete. One digraph test for strong completeness involves its underlying graph (see Exercise 6).

Notice that "greater than" on a set of numbers is not strongly complete since if $a = b$, then $\sim a > b$ and $\sim b > a$. This relation is almost strongly complete in the sense that for all $a \neq b$, aRb or bRa. A binary relation satisfying this condition is called *complete*. The relation R on $X = \{1, 2, 3, 4\}$ defined by Equation (4.1) is neither complete nor strongly complete (why?), but the relation S on X defined by Equation (4.2) is complete but not strongly complete. The relation (X, R) of Example 4.4 is neither complete nor strongly complete. (Again, see Exercise 6 for a digraph representation of the property of strong completeness.)

EXERCISES FOR SECTION 4.1

1. (a) Consider the binary relation "a divides b" on the set of positive integers. Which of the following properties does this relation have: Reflexive, irreflexive, symmetric, asymmetric, antisymmetric, transitive, negatively transitive?

   (b) Repeat part (a) for the relation "uncle of" on a set of people.

(c) For the relation "has the same weight as" on a set of mice.

(d) For the relation "feels smoother than" on a set of objects.

(e) For the relation "admires" on a set of people.

(f) For the relation "has the same blood type as" on a set of people.

(g) For the relation "costs more than" on a set of cars.

(h) For the relation (X, R) where X is the set of all bit strings and aRb means some proper suffix of a is a proper prefix of b. (A *proper suffix* of a string $b_1b_2\cdots b_n$ is a string of the form $b_ib_{i+1}\cdots b_n$, $n \geq i > 1$. A proper prefix is defined similarly.)

2. Show that the binary relation "brother of" on the set of all people in the world is not symmetric.

3. Show that the binary relation "father of" on the set of all people in the world is not strongly complete.

4. Find an example of "everyday language" that can be used to describe the binary relation of (4.2).

5. (Stanat and McAllister [1977]) We are given a library of documents comprising a set Y, and develop a set Z of "descriptors" (e.g., keywords) to describe the documents. Let $X = Y \cup Z$ and let aRb hold if descriptor b applies to document a. Document retrieval systems use the relation (X, R) to find relevant documents for users. Which of the properties of Exercise 1(a) are satisfied by the relation (X, R)?

6. (a) Describe a test for a digraph D's underlying graph that must be satisfied if and only if D is strongly complete.

   (b) How would your answer to part (a) change if strongly complete is replaced by complete?

7. If (X, R) is a binary relation, the *converse* of R is the relation R^{-1} on X defined by

$$aR^{-1}b \text{ iff } bRa.$$

   (a) Describe the digraph of R^{-1} as compared to R.

   (b) Identify the converse of the binary relation "uncle of" on the set of all people in Sweden.

8. If (X, R) is a binary relation, the *complement* of R is the relation R^c on X defined by

$$aR^cb \text{ iff } \sim aRb.$$

   (a) Describe the digraph of R^c as compared to R.

   (b) Prove or give a counterexample to the statement: If R is symmetric, then R^c is symmetric.

   (c) Identify the complement of the binary relation "father of" on the set of all people in Sri Lanka.

9. If (X, R) and (X, S) are binary relations, the *intersection* relation $R \cap S$ on X is defined by

$$R \cap S = \{(a, b) : aRb \text{ and } aSb\}.$$

(a) Identify $(X, R \cap S)$ when X = the set of all people in Ireland, R = "older than," and S = "father of." (Be as succinct as possible.)

(b) Draw the digraph of $(X, R \cap S)$ where $X = \{1, 2, 3, 4\}$ and R and S are as given in (4.1) and (4.2).

10. Suppose that (X, R) and (X, S) are binary relations. For each property listed in Exercise 1(a), assume that (X, R) and (X, S) each have the stated property. Then either prove that $(X, R \cap S)$ also has the property or give an example to show that it may not.

11. Repeat Exercise 10 for (X, R^{-1}).

12. Repeat Exercise 10 for (X, R^c).

13. Which of the properties in Exercise 1(a) hold for $(X, \emptyset)$?

14. Which of the properties in Exercise 1(a) hold for $(X, X \times X)$?

15. (a) Show that every asymmetric relation is antisymmetric.

    (b) Show that the converse of part (a) does not hold.

16. Prove that an asymmetric binary relation will be irreflexive.

17. Show that a binary relation is asymmetric and transitive if and only if it is irreflexive, antisymmetric, and transitive.

18. (a) Show that it is not possible for a binary relation to be both symmetric and asymmetric.

    (b) Show that it is possible for a binary relation to be both symmetric and antisymmetric.

19. Show that there are binary relations that are:

    (a) Transitive but not negatively transitive

    (b) Negatively transitive but not transitive

    (c) Neither negatively transitive nor transitive

    (d) Both negatively transitive and transitive

20. A binary relation (X, R) is an *equivalence relation* if it is reflexive, symmetric, and transitive. Which of the following binary relations (X, R) are equivalence relations?

    (a) $(Re, =)$ where Re is the set of real numbers

    (b) $(Re, \geq)$ (c) $(Re, >)$

    (d) X = a set of people, aRb iff a and b have the same weight

    (e) $X = \{0, 1, 2, \ldots, 22\}$, aRb iff $a \equiv b (\text{mod } 5)$

    (f) X is the collection of all finite sets of real numbers and aRb iff $a \cap b \neq \emptyset$

    (g) $X = \{(1, 1), (2, 3), (3, 8)\}$ and $R = \{\langle(1, 1), (1, 1)\rangle, \langle(2, 3), (2, 3)\rangle, \langle(3, 8), (3, 8)\rangle, \langle(1, 1), (2, 3)\rangle, \langle(2, 3), (1, 1)\rangle\}$

21. If (X, R) and (X, S) are equivalence relations (see Exercise 20), is

    (a) $(X, R \cap S)$ where $R \cap S = \{(a, b) : aRb \text{ and } aSb\}$?

    (b) $(X, R \cup S)$ where $R \cup S = \{(a, b) : aRb \text{ or } aSb\}$?

(c) $(X, R/S)$ where $R/S = \{(a,b) |$ for some $c \in X, aRc$ and $cSb\}$?

22. To show that all the properties of an equivalence relation are needed, give an example of a binary relation that is:

    (a) Reflexive, symmetric, and not transitive

    (b) Reflexive, transitive, and not symmetric

    (c) Symmetric, transitive, and not reflexive

23. If (X, R) is an equivalence relation (see Exercise 20), let $C(a) = \{b \in X | aRb\}$. This is called the *equivalence class containing* a. For example, if $X = \{1,2,3,4,5\}$ and $R = \{(1,1),(1,2),(2,1),(2,2),(3,3),(4,4),(5,5)\}$, then (X, R) is an equivalence relation. The equivalence classes are $C(1) = C(2) = \{1,2\}$, $C(3) = \{3\}$, $C(4) = \{4\}$, $C(5) = \{5\}$.

    (a) Find all equivalence classes in equivalence relation (X, R) of Exercise 20(e).

    (b) Show that two equivalence classes $C(a)$ and $C(b)$ are either disjoint or identical.

    (c) Give an example of an equivalence relation with three distinct equivalence classes.

    (d) Give an example of an equivalence relation with two distinct equivalence classes, one of which has three elements and the other two.

24. Suppose that $X = Re$ and
$$aRb \text{ iff } a > b + 1.$$
Which of the properties in Exercise 1a hold for (X, R)?

25. Consider the binary relation (X, S) where $X = Re$ and
$$aSb \text{ iff } |a - b| \leq 1.$$
This relation is closely related to the binary relation (X, R) of Exercise 24. Which of the properties in Exercise 1(a) hold for (X, S)?

26. If (X, R) is a binary relation, the *symmetric complement* of R is the binary relation S on X defined by
$$aSb \text{ iff } (\sim aRb \;\&\; \sim bRa).$$
Note that if R is strict preference, then S is indifference; you are *indifferent* between two alternatives if and only if you prefer neither.

    (a) Show that the symmetric complement is always symmetric.

    (b) Show that if (X, R) is negatively transitive, the symmetric complement is transitive.

    (c) Show that the converse of part (b) is false.

    (d) If $X = Re$ and R is as defined in Exercise 25, find an inequality to describe the symmetric complement of R.

    (e) Identify the symmetric complement of the following relations:

        i. $(Re, >)$

        ii. $(Re, =)$

iii. (N, R), where N is the set of positive integers and xRy means that x does not divide y

27. Compute the number of binary relations on a set X having n elements.
28. Given a digraph D, prove that D is transitive if and only if $d(u, v) \neq 2$ whenever v is reachable from u.

4.2 ORDER RELATIONS AND THEIR VARIANTS

In this section we study the special binary relations known as order relations and their variants.

4.2.1 Defining the Concept of Order Relation

Example 4.6 Utility Functions Suppose that (X, P) is the strict preference relation of Example 4.1. In economics or psychology, we sometimes seek to reflect preferences by a numerical value so that the higher the value assigned to an object, the more preferred that object is. Thus, we might ask if we can assign a value $f(a)$ to each alternative a in X so that a is strictly preferred to b if and only if $f(a) > f(b)$:

$$aPb \Leftrightarrow f(a) > f(b). \tag{4.4}$$

If this can be done, f is sometimes called a *utility function* (*ordinal utility function*). Utility functions are very useful in decisionmaking applications because they give us a single numerical value on which to base our choices and they provide an *order* for the alternatives. We often choose courses of action that maximize our utility (or "expected" utility). In Example 4.1, a utility function satisfying (4.4) can be found. One example of such a function is

$$\begin{gathered} f(\text{Boston}) = 5, \quad f(\text{New York}) = 5, \quad f(\text{Miami}) = 4, \\ f(\text{Atlanta}) = 2, \quad f(\text{Phoenix}) = 2, \quad f(\text{San Francisco}) = 1, \\ f(\text{Los Angeles}) = 0. \end{gathered} \tag{4.5}$$

This gives us an order for X: Boston and New York tied for first, Miami next, then Atlanta and Phoenix tied, then San Francisco, finally Los Angeles. The relation (X, P) is both transitive and antisymmetric. (In fact, it is asymmetric.) Indeed, any binary relation (X, P) for which there is a function satisfying (4.4) is transitive and antisymmetric (even asymmetric). To see why, note that if aPb and bPc, then (4.4) implies that $f(a) > f(b)$ and $f(b) > f(c)$. Therefore, $f(a) > f(c)$ and, by (4.4), aPc. Antisymmetry is proven similarly.

The notion of utility goes back at least to the eighteenth century. Much of the original interest in this concept goes back to Jeremy Bentham [1789]. According to Bentham: "By utility is meant that property in any object, whereby it tends to produce benefit, advantage, pleasure, good, or happiness ...." Bentham formulated procedures for measuring utility, for he thought that societies should strive for "the greatest good for the greatest number"—that is, maximum utility. The problem of

Table 4.3: Order Relations and their Variants[a]

	Relation Type:						
Defining Property:	Order Relation	Weak Order	Strict Weak Order	Linear Order	Strict Linear Order	Partial Order	Strict Partial Order
Reflexive						$\surd$	
Symmetric							
Transitive	$\surd$	$\surd$		$\surd$	$\surd$	$\surd$	$\surd$
Asymmetric			$\surd$		$\surd$		$\surd$
Antisymmetric	$\surd$			$\surd$		$\surd$	
Negatively transitive			$\surd$				
Strongly complete		$\surd$		$\surd$			
Complete					$\surd$		

[a]A given type of relation can satisfy more of these properties than those indicated. Only the defining properties are indicated.

how to measure utility is a complex one and much has been written about it. See, for example, Barberà, Hammond, and Seidel [2004], Fishburn [1970b], Keeney and Raiffa [1993], Luce [2000], or Roberts [1979b] for discussions. Utilities are used in numerous applications. They can help in business decisions such as when to buy or sell or which computer system to invest in, personal decisions such as where to live or which job offer to accept, in choice of public policy such as new environmental regulations or investments in homeland security initiatives, and so on. ■

A binary relation (X, R) satisfying transitivity and antisymmetry will be called an *order relation* and we say that X is *ordered* by R. Thus, for example, "contained in" on a family of sets is an order relation, as is "strictly contained in," and so is "descendant of" on the set of people in the world and $\geq$ on a set of numbers. Notice that in the digraph of Figure 4.2, every arc is drawn heading downward. If (X, R) is a transitive relation and its corresponding digraph has this kind of a drawing, with the possible exception of loops, then antisymmetry follows and it is an order relation. Notice that, by transitivity, (a, b) is in the relation whenever there is a path from a to b with each arc heading down.[3]

In this section we define a variety of order relations and relations that are closely related to order relations. We summarize the definitions in Table 4.3. The properties defining a type of order relation are not the only properties the relation has. However, in mathematics, we try to use a minimal set of properties in making a definition.

Figure 4.9 shows a number of different examples of order relations. Notice that even though there are ascending arcs, digraph (c) of Figure 4.9 is an order relation. A redrawing exists with all arcs heading down (see Exercise 23). In fact, it is not

[3]Up to now, the position of vertices in a graph or digraph was unimportant—only the adjacencies mattered. Now, position will matter. Actually, we have previously seen this idea when we introduced rooted trees, but the positioning of vertices was not emphasized in that discussion.

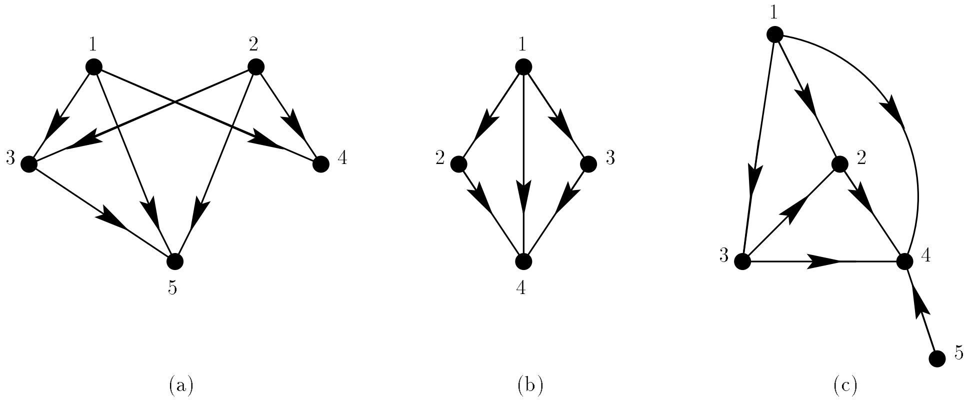


Figure 4.9: Examples of order relations.

hard to show the following:

Theorem 4.1 A transitive binary relation (X, R) is an order relation if and only if its digraph can be drawn so that all arcs (other than loops) head down.[4]

This shows that relations (X, P), (X, L), (X, S), (X, R), and (X, Q) whose corresponding digraphs are shown in Figures 4.2, 4.3, 4.4, 4.5, and 4.6, respectively, are order relations.

One interesting consequence of the two defining properties for order relations is the following theorem.

Theorem 4.2 The digraph of an order relation has no cycles (except loops).

Proof. Suppose that (X, R) is an order relation and that $C = a_1, a_2, \ldots, a_j, a_1$ is a cycle in the corresponding digraph. Thus, $a_1 \neq a_2$. Since (X, R) is transitive and antisymmetric and by the definition of C we know that $(a_1, a_2), (a_2, a_3), \ldots (a_j, a_1)$ are arcs in the digraph. Using the arcs (a_2, a_3) and (a_3, a_4), we apply transitivity to show that $(a_2, a_4) \in R$. Then, since $(a_2, a_4) \in R$ and $(a_4, a_5) \in R$, transitivity implies that $(a_2, a_5) \in R$. Continuing in this way, we conclude that $(a_2, a_j) \in R$. This plus $(a_j, a_1) \in R$ gives us $(a_2, a_1) \in R$. Thus, (a_1, a_2) and (a_2, a_1) are arcs of the digraph, which contradicts the fact that the digraph is antisymmetric, since $a_1 \neq a_2$. Q.E.D.

A binary relation for which we can find a numerical representation satisfying (4.4) has more properties than just transitivity and antisymmetry. For example, it is negatively transitive, to give just one example of another property. (Why?) In later subsections, we will define stronger types of order relations by adding properties that they are required to satisfy.

Besides transitivity and antisymmetry, many of the order relations that we will introduce will be either reflexive or irreflexive. These two possibilities are usually

[4]Although we do not make it explicit, this theorem assumes the hypothesis that X is finite.

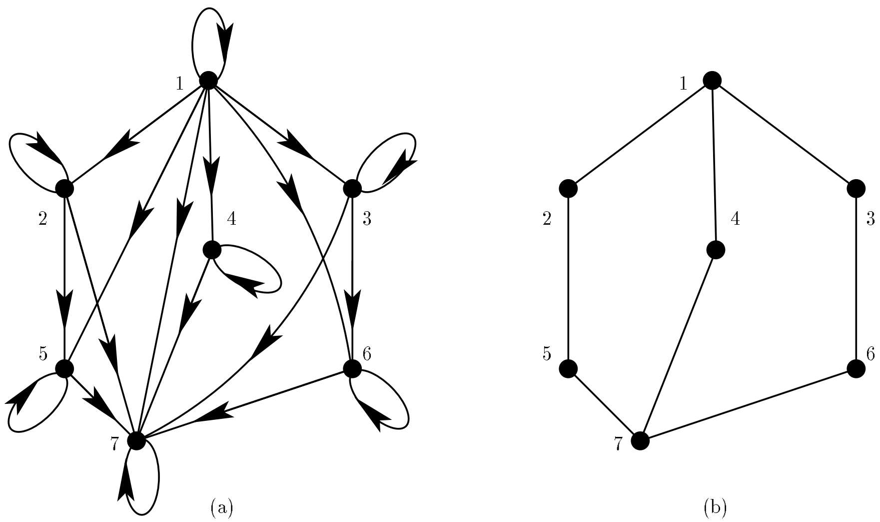


Figure 4.10: A digraph (a) that is a partial order and its associated diagram (b).

based on the context of the problem at hand. If we assume reflexivity, we call our order relation (X, R) a *partial order* or X a *partially ordered set* or *poset*.[5] Figure 4.10(a) shows a partial order. If irreflexivity is assumed, then the adjective "strict" will be used; (X, R) is a *strict partial order* if it is irreflexive, antisymmetric, and transitive. Figures 4.9 (a), (b), (c), 4.2, 4.3, 4.4, 4.5, and 4.6 show strict partial orders. A strict partial order is sometimes defined more succinctly as an asymmetric and transitive binary relation. This is because we have the following theorem.

Theorem 4.3 A binary relation is irreflexive, transitive, and antisymmetric if and only if it is transitive and asymmetric.

Proof. Suppose that (X, R) is irreflexive, transitive, and antisymmetric. Suppose that aRb and bRa. Then $a = b$ by antisymmetry. But aRa fails by irreflexivity. Thus, aRb and bRa cannot both hold and (X, R) is asymmetric.

Conversely, suppose that (X, R) is transitive and asymmetric. Then aRb and bRa cannot both hold, so (X, R) is (vacuously) antisymmetric. Moreover, asymmetry implies irreflexivity, since aRa implies that aRa and aRa, which cannot be the case by asymmetry. Q.E.D.

4.2.2 The Diagram of an Order Relation

Consider an order relation R on a set X. Since antisymmetry and transitivity are defining properties, these can be used to simplify the digraph of the relation.

[5] Some authors use "ordered set" and "partially ordered set" interchangeably. We shall make the distinction to allow for nonreflexive order relations.

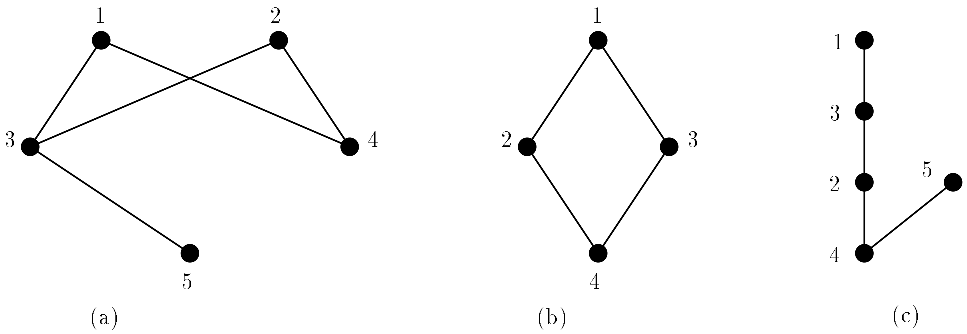


Figure 4.11: Diagrams of the order relations (a), (b), and (c) from Figure 4.9.

We illustrate with digraph (a) of Figure 4.10. Since all arcs (except loops) point downward, why not remove the arrowheads from the arcs, thus turning them into edges? In addition, this digraph is reflexive, so drawing a loop at each vertex is unnecessary. These two changes transform the digraph into a loopless graph. Next, consider the edges from 3 to 6, 6 to 7, and 3 to 7. Since transitivity is known to hold, the edge from 3 to 7 is unnecessary. In general, we can remove all edges that are implied by transitivity. This will simplify the digraph (graph) of the order relation dramatically. The graph produced by removing loops, arrowheads, and arcs implied by transitivity will be called the *diagram* (*order diagram* or *Hasse diagram*) of the order relation. Figure 4.10(b) shows the diagram of the order relation from Figure 4.10(a). In a diagram, aRb if there is a descending chain from a to b. For example, in Figure 4.10(b), $1R5$ and $3R7$ but $\sim 3R5$. The same kinds of simplifications can be made for the digraphs of order relations which are irreflexive. The graphs resulting from these same simplifications of the digraphs of an irreflexive order relation will also be called diagrams. The diagrams associated with digraphs (a), (b), and (c) of Figure 4.9 are shown in Figure 4.11.

Similarly, any loopless graph can be reduced to a diagram, i.e., an order relation, as long as no edge is horizontal. Edges (arcs) that can be assumed by transitivity in the graph can be removed. A diagram will never contain loops, so we will not be able to ascertain reflexivity of the relation from the diagram. Only through the definition of the relation or the context of the presentation can reflexivity be determined.

Consider an order relation R on a set X. We say that x *covers* y or xKy if xRy and there is no z for which xRz and zRy. The binary relation (X, K) is called the *cover relation* of (X, R). We can define the *cover graph* G_K associated with R as follows: (a) $V(G_K)$ is the set X; (b) $\{x, y\} \in E(G_K)$ if x covers y. Hence, the drawing of the cover graph associated with an order relation is actually the diagram of the order relation with y lower than x whenever x covers y. Alternatively, any diagram is the cover graph of an order relation. Consider diagram (b) of Figure 4.11. This diagram defines the cover graph of the relation (X, R) defined by Figure 4.9(b). Here, $K = \{(1, 2), (1, 3), (2, 4), (3, 4)\}$.

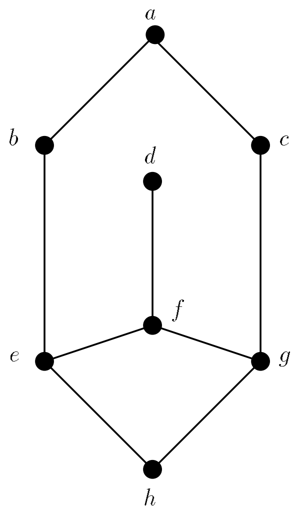


Figure 4.12: Another diagram.

Again, consider an order relation R on a set X. We will use xSy to mean that xRy and $x \neq y$. If xSy or ySx, then x and y are said to be *comparable* and we write xCy. Alternatively, x and y are said to be *incomparable*, written xIy, if neither xRy nor yRx.

The diagram implications of S, C, and I are also straightforward. xSy if and only if there is a descending chain from x to y. Either a descending chain from x to y or y to x means that xCy, while for $x \neq y$, xIy implies that either there is no chain between x and y or that the only chains are neither strictly ascending nor strictly descending. In the diagram of Figure 4.10(b) we see immediately that $3C7$, $3S7$, $\sim 2C6$, and $2I3$.

An element x is called *maximal* in an ordered set (X, R) if there is no $y \in X$ such that ySx. If there is only one maximal element, that element is called *maximum* and denoted $\widehat{1}$. We let $\max(X, R)$ be the set of maximal elements in (X, R). Similar definitions can be made for the terms *minimal*, *minimum*, and $\min(X, R)$. If a minimum element exists, it is denoted $\widehat{0}$. Consider the diagram of Figure 4.12. Here d is a maximal element, h is minimum, $\max(X, R) = \{a, d\}$, and $\min(X, R) = \{h\}$ (since h is minimum). While every order relation (on a finite set) has at least one minimal and at least one maximal element [see Exercise 24(a)], it may not necessarily have either a maximum or a minimum element.

4.2.3 Linear Orders

In many applications, we seek an order relation that "ranks" alternatives, i.e., it gives a first choice, second choice, third choice, etc. Such an order is an example of what we will call a strict linear order. Any time we need to "line up" a set of elements, we are creating a strict linear ordering for that set. The way patients are seen by a doctor at a clinic, when schoolchildren need to line up in single file, and the way television programs are scheduled for airing on a certain channel are all

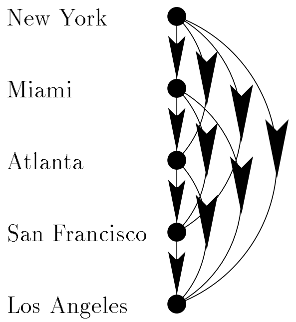


Figure 4.13: A generated subgraph of the digraph of Figure 4.2.

examples of strict linear orders. When we rank alternative political candidates or software packages according to "quality," we are aiming to produce a strict linear order. A binary relation is called a *strict linear order* if it is transitive, asymmetric, and complete. The order relation drawn in Figure 4.13 is an example of a strict linear order. (The terms *strict total order* and *strict simple order* are also used.) A strict linear order can and will satisfy more of our properties but only the three properties given are needed to define it. (It is left to the reader to show that none of the properties of the definition are superfluous; see Exercise 21.) In fact, a strict linear order will also be irreflexive, antisymmetric, and negatively transitive. Asymmetry implies both irreflexivity and antisymmetry (see proof of Theorem 4.3), while transitivity with completeness implies negative transitivity. (The proof of the latter is left to the reader; see Exercise 22.)

The prototype of strict linear orders is the relation $>$ on any set of real numbers. By Theorem 4.1 and the completeness property, in a strict linear order R on a finite set X, the diagram of R consists of the elements of X laid out on a vertical line, i.e., aRb if and only if a is above b. On the other hand, a diagram in the shape of a vertical line will always be a strict linear order if we assume irreflexivity (no loops); completeness is the only property that needs to be checked, since transitivity and asymmetry follow from the definition of order relations under the assumption of irreflexivity. To show completeness, consider any two elements $a \neq b$ in X. Since one must be above the other in a vertical line diagram, by the definition of a diagram, the higher element is "R" to the lower element. Therefore, it must be the case that either aRb or bRa. Since drawing strict linear orders (vertical line graphs) is unilluminating, we will use the notation of Trotter [1992] to describe strict linear orders succinctly. Let $L_S = [a_{i_1}, a_{i_2}, \ldots, a_{i_n}]$ denote the strict linear order S on the set $X = \{a_1, a_2, \ldots, a_n\}$, where $a_{i_j} S a_{i_k}$ whenever a_{i_j} precedes a_{i_k} in L_S. In this notation, the strict linear order of Figure 4.13 is given by [New York, Miami, Atlanta, San Francisco, Los Angeles].

Recall that the term "strict" in strict orders refers to the fact that the relation is irreflexive, whereas the "nonstrict" version is reflexive. We define a *linear order* by the antisymmetric, transitive, and strongly complete properties. It is simple to show that the same things can be said for linear orders as strict linear orders except for the fact that linear orders are reflexive. In particular, the prototype of linear orders is the relation $\geq$ on any set of real numbers. The strict linear order notation, $[a_{i_1}, a_{i_2}, \ldots, a_{i_n}]$, will also be used for a linear order R, with the only change being that $a_j R a_j$ holds for all j.

Example 4.7 Lexicographic Orders (Stanat and McAllister [1977]) Let Σ be a finite alphabet and let R be a strict linear order of elements of Σ. Let X be the set of strings from Σ and define the lexicographic (dictionary) order S on X as follows. First, we take xSy if x is a prefix of y. Second, suppose that $x = zu$, $y = zv$, and z is the longest prefix common to x and y. Then we take xSy if the first symbol of u precedes the first symbol of v in the strict linear order R. For example, if Σ is the alphabet $\{a, b, c, \ldots, z\}$ and R is the usual alphabetical order, then $abSabht$. Also, $abdtSabeab$ since we have $z = ab$, $u = dt$, $v = eab$, and the first symbol of u, i.e., d, precedes the first symbol of v, i.e., e. The binary relation (X, S) corresponds to the usual ordering used in dictionaries. It defines a strict linear order since it is transitive, asymmetric, and complete (see Exercise 15).

Let us continue with the case where Σ is the alphabet $\{a, b, c, \ldots, z\}$ and R is the usual alphabetical order. In the language of Section 4.2.2, if x is any element of X, x covers xa since $xSxa$ and there is no string z in X such that $xSzSxa$. On the other hand, no element covers xb. Why? (See Exercise 17.) This example has an infinite set X. A finite example arises if we take all strings in Σ of at most a given length. ■

4.2.4 Weak Orders

When we are ranking alternatives as first choice, second choice, and so on, we may want to allow ties. We next consider relations called weak orders that are like linear orders except that ties are allowed. A *weak order* is a binary relation that is transitive and strongly complete. Note that since antisymmetry is not assumed, a weak order may not be an order relation. Since antisymmetry is the one defining property of linear orders not necessarily assumed for weak orders, we can have aRb and bRa for $a \neq b$ for weak orders R. In this case, we can think of a and b as "tied" in R. Figure 4.14 shows a "diagram" of a typical weak order. This is not a diagram in the sense that we have defined diagrams. Here, each element has a horizontal level, all elements a and b at the same horizontal level satisfy aRb and bRa, and, otherwise, aRb if and only if a is at a higher level than b. Thus, in the weak order corresponding to Figure 4.14, R is given by

$$x_i R y_j \text{ iff } x = y \text{ or } x \text{ precedes } y \text{ in the alphabet,}$$

where x and y are a, b, c, d, e, or f. For example, $b_2 R e_3$ and $a_1 R f_2$. One can show (see Roberts [1976, 1979b]) that every weak order (on a finite set) arises this way.

It is sometimes useful to consider *strict weak orders*, binary relations that arise in the same way as weak orders except that for elements a and b at the same horizontal level, we do not have aRb. Thus, in a figure like Figure 4.14, we have $x_i R y_j$ if and only if x precedes y in the alphabet. The relation R defined in this way from such

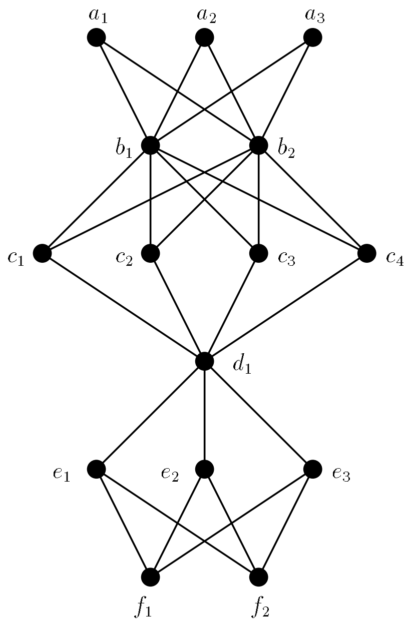


Figure 4.14: This figure defines a "typical" weak order/strict weak order.

a "diagram" is easily seen to be asymmetric and negatively transitive. Conversely, if X is finite, a relation (X, R) that is asymmetric and negatively transitive can be seen to come from a "diagram" like Figure 4.14 in this way (see Roberts [1976, 1979b]). A relation that is asymmetric and negatively transitive is called a *strict weak order*. It is simple to show that weak orders are to strict weak orders as linear orders are to strict linear orders. The only difference is for elements at the same horizontal level, i.e., "tied."

Note that whereas weak orders may not be order relations according to our definition, strict weak orders always are (why?). Note also that strict weak orders allow incomparable elements but only in a special way. The digraph of Figure 4.2 defines a strict weak order. It is easy to see that it can be redrawn with Boston and New York at the top level, then Miami, then Atlanta and Phoenix, then San Francisco, and, finally, Los Angeles. The levels are readily obtained from the function f of (4.5). Recall that function f is a utility function, defined from (4.4). Conversely, if a strict preference relation P is defined from a utility function f by (4.4), it is easy to see that it is asymmetric and negatively transitive, i.e., a strict weak order. The proof is left as an exercise (Exercise 20).

For the same reason that the digraph of Figure 4.2 defines a strict weak order, so does the digraph of Figure 4.5. If we let $f(x) = r(x, q)$ as in Example 4.4, then we have

$$xRy \Leftrightarrow r(x, q) > r(y, q) \Leftrightarrow f(x) > f(y),$$

which gives us the equivalent of (4.4).

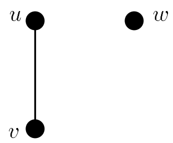


Figure 4.15: A diagram not found in a strict weak order.

Table 4.4: Preference Orderings (Strict Linear Orderings) for a Size 4 Stable Marriage Problem.

Men's Preferences	Women's Preferences
$m_1 : [w_1, w_2, w_3, w_4]$	$w_1 : [m_4, m_3, m_2, m_1]$
$m_2 : [w_2, w_1, w_4, w_3]$	$w_2 : [m_3, m_4, m_1, m_2]$
$m_3 : [w_3, w_4, w_1, w_2]$	$w_3 : [m_2, m_1, m_4, m_3]$
$m_4 : [w_4, w_3, w_2, w_1]$	$w_4 : [m_1, m_2, m_3, m_4]$

It is not hard to show that the digraphs of Figures 4.3 and 4.6 are not strict weak orders. Thus, a function satisfying (4.4) does not exist in either case. To see why, we consider the diagram in Figure 4.15. Notice that elements u and w are incomparable and elements w and v are incomparable. In particular, $\sim uRw$ and $\sim wRv$ follows. However, uRv. Thus, this diagram is not negatively transitive, nor can it be part of a larger negatively transitive diagram. This diagram is essentially the definition of not negatively transitive with regard to diagrams. An order relation is a strict weak order if and only if its diagram does not contain the diagram in Figure 4.15.

4.2.5 Stable Marriages[6]

Suppose that n men and n women are to be married to each other. Before we decide on how to pair up the couples, each man and each woman supplies a preference list of the opposite sex, a strict linear order. A *set of stable marriages* (or a *stable matching*) is a pairing (or matching) of the men and women so that no man and woman would both be better off (in terms of their preferences) by leaving their assigned partners and marrying each other. This problem and a number of its variations were introduced in Gale and Shapley [1962].

Consider the case where $n = 4$ and the preferences are given by the strict linear orders in Table 4.4. Note that $M_1 = \{m_1 - w_4, m_2 - w_3, m_3 - w_1, m_4 - w_2\}$ is a stable set of marriages. To see why, note that w_4 and w_3 married their first choice; so neither would be willing to leave their partner, m_1, m_2, respectively. Also, m_3 and m_4 are getting their first choice among the other women, namely, w_1 and w_2, respectively. In all, there are 10 stable matchings for this problem. Two other obvious stable matchings are $M_9 = \{m_1 - w_1, m_2 - w_2, m_3 - w_3, m_4 - w_4\}$ and

[6] This subsection is based on Gusfield and Irving [1989].

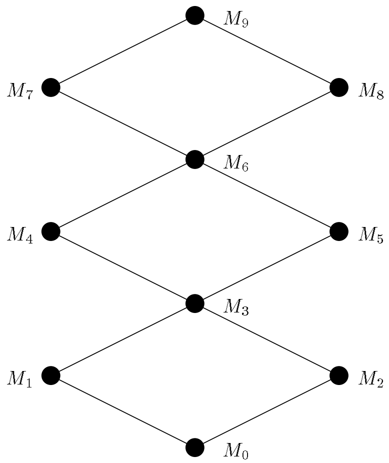


Figure 4.16: The partial order of the man-oriented dominance relation on stable matchings.

$M_0 = \{m_1 - w_4, m_2 - w_3, m_3 - w_2, m_4 - w_1\}$. These are both stable since each man (woman) got his (her) first choice. The full list of 10 stable matchings is

$$\begin{array}{lcl} M_0 & = & \{m_1 - w_4, m_2 - w_3, m_3 - w_2, m_4 - w_1\} \\ M_1 & = & \{m_1 - w_4, m_2 - w_3, m_3 - w_1, m_4 - w_2\} \\ M_2 & = & \{m_1 - w_3, m_2 - w_4, m_3 - w_2, m_4 - w_1\} \\ M_3 & = & \{m_1 - w_3, m_2 - w_4, m_3 - w_1, m_4 - w_2\} \\ M_4 & = & \{m_1 - w_2, m_2 - w_4, m_3 - w_1, m_4 - w_3\} \\ M_5 & = & \{m_1 - w_3, m_2 - w_1, m_3 - w_4, m_4 - w_2\} \\ M_6 & = & \{m_1 - w_2, m_2 - w_1, m_3 - w_4, m_4 - w_3\} \\ M_7 & = & \{m_1 - w_2, m_2 - w_1, m_3 - w_3, m_4 - w_4\} \\ M_8 & = & \{m_1 - w_1, m_2 - w_2, m_3 - w_4, m_4 - w_3\} \\ M_9 & = & \{m_1 - w_1, m_2 - w_2, m_3 - w_3, m_4 - w_4\}. \end{array}$$

Given this set of all 10 stable matchings, person x would prefer one stable matching M_i over another M_j if x prefers his/her partner in M_i to his/her partner in M_j. We can then define the *man-oriented dominance relation* as follows: M_i dominates M_j if every man prefers M_i to M_j or is indifferent between them. It is not hard to show that man-oriented dominance is a partial order. Its diagram is shown in Figure 4.16. (See Section 12.8 for a more complete treatment of the stable marriage problem.)

EXERCISES FOR SECTION 4.2

1. Which of the following are order relations?

   (a) $\subsetneq$ on the collection of subsets of $\{1, 2, 3, 4\}$

(b) (X, P), where $X = Re \times Re$ and

$$(a, b)P(s, t) \text{ iff } (a > s \text{ and } b > t)$$

(c) (X, Q), where X is a set of n-dimensional alternatives, $f_1, f_2, \ldots, f_n$ are real-valued scales on X, and Q is defined by

$$aQb \Leftrightarrow [f_i(a) > f_i(b) \text{ for each } i]$$

(d) (X, Q), where X is as in part (c) and

$$aQb \Leftrightarrow [f_i(a) \geq f_i(b) \text{ for each } i \text{ and } f_i(a) > f_i(b) \text{ for some } i]$$

2. Which of the binary relations in Exercise 1 are strict partial orders?
3. Which of the binary relations in Exercise 1 are linear orders?
4. Which of the binary relations in Exercise 1 are strict linear orders?
5. Which of the binary relations in Exercise 1 are weak orders?
6. Which of the binary relations in Exercise 1 are strict weak orders?
7. Which of the following are linear orders?
   (a) $\{(a, a), (b, b), (c, c), (d, d), (c, d), (c, b), (c, a), (d, b), (d, a), (b, a)\}$ on set $X = \{a, b, c, d\}$
   (b) $\{(a, a), (a, b), (a, d), (b, b), (b, d), (c, a), (c, b), (c, d), (d, d)\}$ on set $X = \{a, b, c, d\}$
   (c) $\{(a, a), (a, b), (a, c), (a, d), (b, b), (b, c), (b, d), (c, a), (c, c), (c, d), (d, d)\}$ on set $X = \{a, b, c, d\}$
   (d) $\{(\lambda, \lambda), (\lambda, \xi), (\lambda, \zeta), (\lambda, \varphi), (\xi, \xi), (\xi, \zeta), (\xi, \varphi), (\rho, \lambda), (\rho, \xi), (\rho, \rho), (\rho, \zeta), (\rho, \varphi), (\zeta, \zeta), (\zeta, \varphi), (\varphi, \varphi)\}$ on set $X = \{\lambda, \xi, \zeta, \varphi, \rho\}$
8. Which of the following are strict linear orders?
   (a) $\{(c, d), (c, b), (c, a), (d, b), (d, a), (b, a)\}$ on set $X = \{a, b, c, d\}$
   (b) $\{(a, b), (c, b), (a, d), (c, d), (b, d), (c, a), (b, c)\}$ on set $X = \{a, b, c, d\}$
   (c) $\{(\lambda, \delta), (\lambda, \xi), (\lambda, \zeta), (\lambda, \varphi), (\xi, \delta), (\xi, \zeta), (\xi, \varphi), (\xi, \xi), (\rho, \lambda), (\rho, \xi), (\rho, \delta), (\rho, \zeta), (\rho, \varphi), (\zeta, \delta), (\zeta, \varphi), (\varphi, \delta)\}$ on set $X = \{\lambda, \xi, \zeta, \varphi, \rho\}$
   (d) $\{(5, 1), (5, 2), (5, 4), (1, 2), (1, 4), (3, 5), (3, 1), (3, 2), (3, 4), (2, 4)\}$ on set $X = \{1, 2, 3, 4, 5\}$
9. Which of the following are weak orders?
   (a) $\{(a, a), (b, b), (c, c), (d, d), (b, a), (b, c), (b, d), (c, a), (c, d), (a, d)\}$ on set $X = \{a, b, c, d\}$
   (b) $\{(1,1), (1,2), (1,3), (1,4), (1,5), (1,6), (1,7), (1,8), (1,9), (2,2), (2,5), (2,6), (2,7), (2,8), (2,9), (3,3), (3,5), (3,6), (3,7), (3,8), (3,9), (4,4), (4,5), (4,6), (4,7), (4,8), (4,9), (5,5), (5,7), (5,8), (5,9), (6,6), (6,7), (6,8), (6,9), (7,7), (7,9), (8,8), (8,9), (9,9)\}$ on set $X = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
   (c) $\{(\Gamma, \Delta), (\Delta, \Psi), (\Gamma, \Upsilon), (\Delta, \Upsilon), (\Gamma, \Omega), (\Upsilon, \Psi), (\Gamma, \Psi), (\Psi, \Omega), (\Delta, \Omega), (\Upsilon, \Omega)\}$ on set $X = \{\Gamma, \Delta, \Psi, \Upsilon, \Omega\}$
10. Which of the binary relations of Exercise 9 are strict weak orders?

11. Draw the diagrams corresponding to the strict partial orders of:

    (a) Figure 4.3 (b) Figure 4.4 (c) Figure 4.6

12. Draw the diagram for the strict weak order of Figure 4.5 in the same way as Figure 4.14, with "tied" elements at the same horizontal level.

13. Suppose that $X = \{1, 2, 3, 4\}$ and

$$R = \{(1,1), (2,2), (3,3), (4,4), (1,3), (1,4), (2,3), (2,4), (3,4)\}.$$

    (a) Show that (X, R) is a partial order.

    (b) Draw the diagram for (X, R).

    (c) Find (X, K), the cover relation associated with (X, R).

14. Suppose that $X = \{t, u, v, w, x, y, z\}$ and

$$R = \{(t,u), (t,v), (u,v), (w,v), (w,x), (w,y), (w,z), (x,v), (x,z), (y,v), (y,z), (z,v)\}.$$

    (a) Show that (X, R) is a strict partial order.

    (b) Draw the diagram for (X, R).

    (c) Find (X, K), the cover relation associated with (X, R).

15. Show that the binary relation (X, S) of Example 4.7 is:

    (a) Transitive (b) Asymmetric (c) Complete

16. From Example 4.7, write out S for the strict linear order (X, S), where X is the set of all strings of length at most 4 and:

    (a) $\Sigma = \{a, b\}$ if aRb (b) $\Sigma = \{a, b, c\}$ if aRb, aRc, and bRc

17. From Example 4.7, if $\Sigma = \{a, b, c, \ldots, z\}$ and R is the usual alphabetical order, explain why $xb \in X$ has no cover.

18. Suppose that $X = \{a, b, c, d, e, f\}$ and

$$R = \{(a,c), (a,f), (b,c), (b,f), (d,a), (d,b), (d,c), (d,e), (d,f), (e,c), (e,f)\}.$$

    (a) Show that (X, R) is a strict weak order.

    (b) Draw the diagram for (X, R).

19. Consider the strict linear order $L_S = [x_1, x_2, \ldots, x_n]$.

    (a) Find $L_{S^{-1}}$. (b) Find $L_S \cap L_{S^{-1}}$. (c) Find $L_S \cup L_{S^{-1}}$.

20. If (X, P) is a strict preference relation defined from a utility function f by (4.4), show that P is asymmetric and negatively transitive.

21. Prove that no two of the following three properties imply the third: transitive, complete, asymmetric.

22. Prove that a transitive and complete binary relation will be negatively transitive.

23. Redraw digraph (c) in Figure 4.9 to prove that it is an order relation. That is, redraw the digraph so that all arcs are descending.

24. (a) Prove that every order relation (on a finite set) has at least one maximal and one minimal element.
    (b) Prove that every (strict) linear order has a maximum and a minimum element.
25. (a) Is the converse R^{-1} of a strict partial order necessarily a strict partial order? (For the definition of converse, see Exercise 7, Section 4.1.)
    (b) Is the converse R^{-1} of a partial order necessarily a partial order?
26. Show that every strict weak order is a strict partial order.
27. If (X, R) is strict weak, define S on X by

$$aSb \text{ iff } (aRb \text{ or } a = b).$$

    Show that (X, S) is a partial order.
28. Draw the diagram of the converse (Exercise 7, Section 4.1) of the strict partial order defined by Figure 4.14.
29. (a) Is the converse of a strict weak order necessarily a strict weak order? Why?
    (b) Is the converse of a strict partial order necessarily a strict partial order? Why?
30. (a) Is the complement of a strict weak order necessarily a strict weak order? (For the definition of complement, see Exercise 8, Section 4.1.)
    (b) Is the complement of a strict partial order necessarily a strict partial order?
31. Prove that the man-oriented dominance relation of Section 4.2.5 is a partial order.
32. Consider the set of stable matchings of Section 4.2.5.
    (a) Write the definition for a woman-oriented dominance relation on a set of stable matchings.
    (b) Draw the diagram for the woman-oriented dominance relation on this set of stable matchings.
    (c) How does the diagram for man-oriented dominance compare with the diagram for woman-oriented dominance?
33. (a) Explain why

$$\{m_1 - w_1, m_2 - w_4, m_3 - w_2, m_4 - w_3\}$$

    is not a set of stable marriages for the preference orderings of Table 4.4.
    (b) How many sets of marriages are not stable for the preference orderings of Table 4.4?
    (c) Prove that $M_0, M_1, \ldots, M_9$ are the only sets of stable marriages for the preference orderings of Table 4.4.

4.3 LINEAR EXTENSIONS OF PARTIAL ORDERS

4.3.1 Linear Extensions and Dimension

Example 4.8 Mystery Novels and Linear Extensions When writing a good murder mystery novel, the author gives clues that help the reader figure out "who

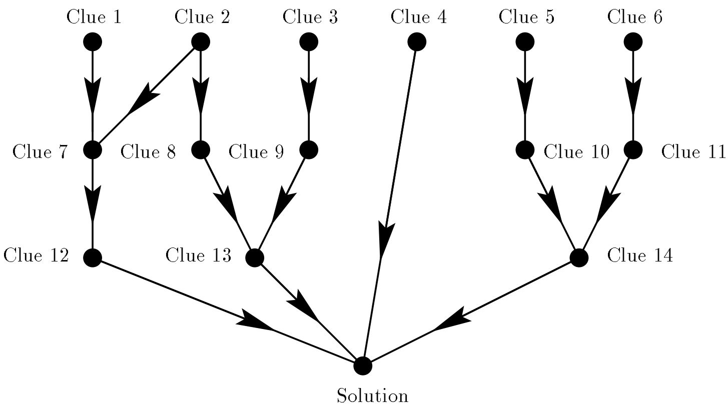


Figure 4.17: "Clue" digraph for a novel.

done it." Some clues depend on earlier clues to be understood. Consider Figure 4.17. The vertices of the digraph represent the clues and "Solution" in some novel, and the arcs represent dependency of one clue on another. For example, the arc from Clue 3 to Clue 9 represents the fact that Clue 3 is needed to understand Clue 9. Clearly, we can assume transitivity, so we do not have to draw all of the arcs between clues. Also, certainly antisymmetry holds, so our digraph defines an order relation. The arcs from some clues to the Solution vertex represent the fact that those clues are needed to figure out the mystery. Thus, by transitivity, all clues are needed to figure out the mystery.

The task facing the author is in what order to present the clues in the novel in a "coherent" way. By coherent we mean that no clue is presented until all of its dependent clues are given first. Depending on the digraph, there could be lots of ways to present the clues. Figure 4.18 shows four ways in which to present the clues for the digraph of Figure 4.17. Again, transitivity is assumed in these digraphs. The digraphs of Figure 4.18 illustrate the idea of a linear extension.

Consider strict partial orders R and S on the same set X. If aRb whenever aSb, then R is an *extension* of S. Put another way, R is an *extension* of S if all of the ordered pairs that define S are found in R. If R is a strict linear order, R is called a *linear extension* of S. The digraph of Figure 4.17 is in fact a strict partial order (if we recall that the arcs implied by transitivity are omitted) and each digraph of Figure 4.18 gives a linear extension of this order relation. A linear extension is just the thing the author of the mystery novel is searching for. Because of the linear nature in which the words of a book are written, the clues must be presented in the order given by a linear extension. ■

Example 4.9 Topological Sorting Linear extensions of strict partial orders arise naturally when we need to enter a strict partial order (X, R) into a computer. We enter the elements of X in a sequential order and want to enter them in such a way that the strict partial order is preserved. The process generalizes to finding a

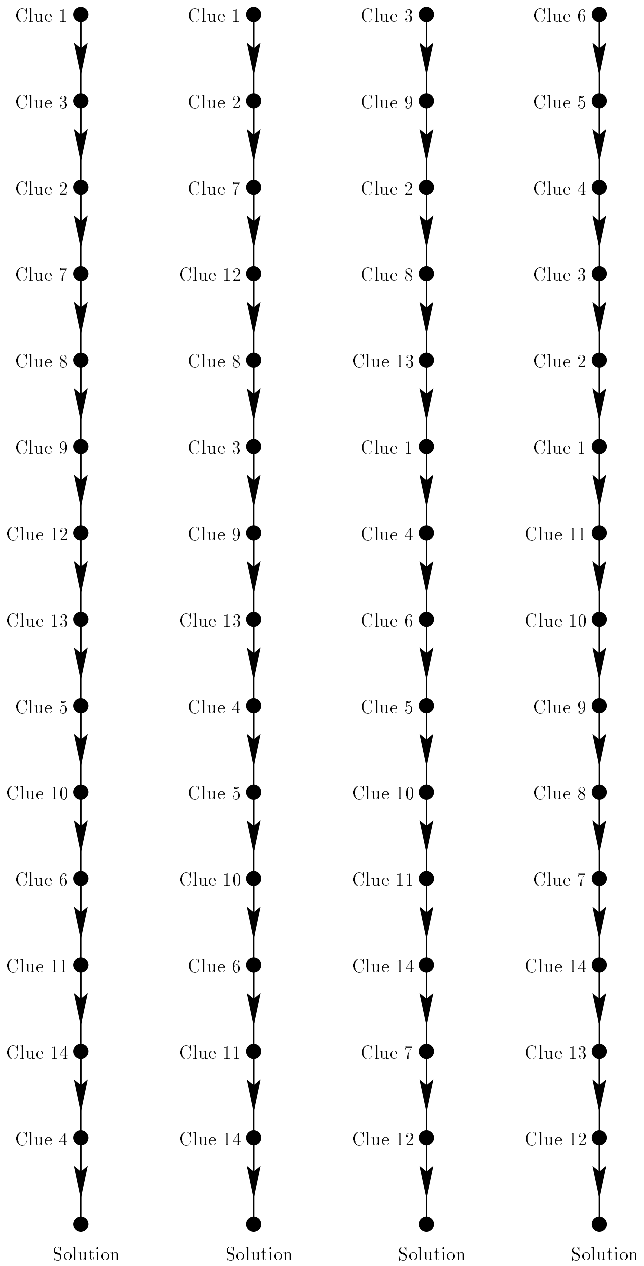


Figure 4.18: Possible clue presentation models.

way to label the vertices of an arbitrary digraph D with the integers $1, 2, \ldots, n$ so that every arc of D goes from a vertex with a smaller label to a vertex with a larger label. This process is called *topological sorting*. We return to this in Sections 11.6.2–11.6.4, where we describe applications of the idea to project planning, scheduling, and facilities design. ■

The following theorem says that linear extensions don't happen by chance.

Theorem 4.4 (Szpilrajn's Extension Theorem [1930]) Every strict partial order has a linear extension. Moreover, if (X, R) is a strict partial order and xIy for distinct $x, y \in X$ (i.e., x and y are incomparable), there is at least one linear extension L with xLy.

It is not hard to show that the following algorithm will find a linear extension of a given strict partial order. In fact, it can be used in such a way as to find all of the linear extensions of a strict partial order. The algorithm uses the notion of a suborder. If (X, R) is a strict partial order, D is its corresponding digraph, and $Y \subseteqq X$, the subgraph generated by Y is again the digraph of a strict partial order since it is asymmetric and transitive. We call this relation a *suborder* and denote it by (Y, R). (Technically speaking, this is not proper notation. We mean all ordered pairs in $Y \times Y$ that are contained in R.) Figure 4.13 is the suborder of the order relation of Figure 4.2 generated by the set

$$Y = \{\text{New York, Miami, Atlanta, San Francisco, Los Angeles}\}.$$

Algorithm 4.1: Linear Extension

Input: A strict partial order (X, R).
Output: A linear extension of (X, R).

Step 1. Set $m = |X|$.

Step 2. Find a minimal element x in (X, R).

Step 3. Let $X = X - x$, $R = R \cap \{(X - x) \times (X - x)\}$, and $b_m = x$.

Step 4. Decrease m by 1. If m is now 0, stop and output the digraph B whose vertex set is X and where (b_i, b_j) is an arc if $i < j$. If not, return to Step 2.

Here are a few remarks about the algorithm: (1) By Theorem 4.2, a strict partial order will always have a minimal element, which is needed in Step 2. (2) $X - x$ with R as redefined in Step 3 is the suborder we are denoting $(X - x, R)$. It is again a strict partial order and hence has a minimal element. (3) B is easily seen to be a linear extension of (X, R).

To illustrate Algorithm 4.1, consider the strict partial order in Figure 4.19. Step 1 sets $m = 4$ and we can choose the minimal element a_4 for Step 2. Then we consider the strict partial order with a_4 removed and let $b_4 = a_4$; this is Step 3. Next, we can choose the minimal element a_3, let $b_3 = a_3$, and consider the strict partial order with a_3 (and a_4) removed. We can then choose the minimal element

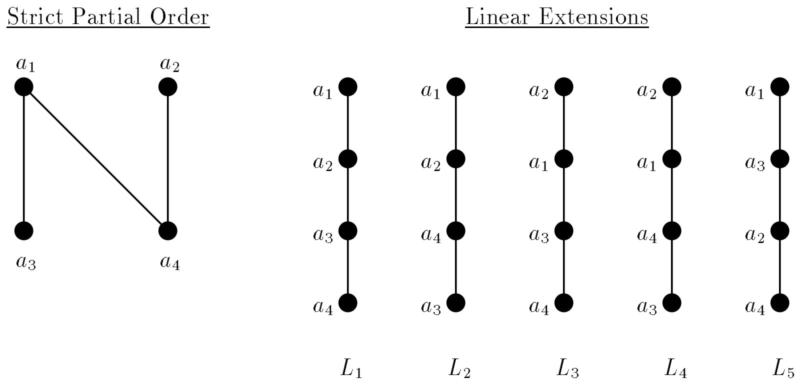


Figure 4.19: A strict partial order on four elements and its linear extensions.

a_2, let $b_2 = a_2$, and consider the strict partial order with a_2 (a_3 and a_4) removed. Finally, we choose the remaining (minimal) element a_1 and let $b_1 = a_1$. The digraph B defined in Step 4 corresponds to the linear extension L_1 in the figure.

Let us consider the set $\mathcal{F}$ of all linear extensions of a strict partial order (X, R). If xRy, the same must be true in every linear extension of $\mathcal{F}$. If xIy, there is at least one linear extension in $\mathcal{F}$ that has x above y and another that has y above x, by Szpilrajn's Extension Theorem. Thus, we get the following result.

Theorem 4.5 If (X, R) is a strict partial order and $\mathcal{F}$ is the set of linear extensions of (X, R), then

$$(X, R) = \bigcap_{L \in \mathcal{F}} L.$$

Since a strict partial order is the intersection of all its linear extensions, a natural question to ask is: How many linear extensions are needed before their intersection is the original strict partial order? Given a strict partial order (X, R), the size of the smallest set of linear extensions whose intersection is (X, R) is called the *dimension* of (X, R), written $\dim(X, R)$. (This idea was introduced by Dushnik and Miller [1941].) Figure 4.19 gives all five linear extensions, L_1, L_2, L_3, L_4, L_5, of the strict partial order (X, R) given there. (Recall that for a set of four elements, there are 4! orderings of the elements. In this case, only five of the $4! = 24$ possible orderings are linear extensions.) Since (X, R) is not a linear order, $\dim(X, R) > 1$, and by Theorem 4.5, $\dim(X, R) \leq 5$. It is easy to see that $L_4 \cap L_5$ equals (X, R). [For example, $a_3 I a_4$ in (X, R) and $a_3 S a_4$ in L_5 while $a_4 S a_3$ in L_4.] Therefore, $\dim(X, R) = 2$. Finding the dimension of an arbitrary order relation is not an easy problem. Given an order relation (X, R), Yannakakis [1982] showed that testing for $\dim(X, R) \leq t$ is NP-complete, for every fixed $t \geq 3$. See Trotter [1996] for a survey article on dimension. Due to the Yannakakis result, most of the results related to dimension take the form of bounds or exact values for specific classes of ordered sets. We present a few of them in the next section.

Example 4.10 Multidimensional Utility Functions Suppose that (X, P) is the strict preference relation of Example 4.1. In Example 4.6, we talked about a function f measuring the value $f(a)$ of each alternative a in X, with a strictly preferred to b if and only if the value of a is greater than the value of b; that is,

$$aPb \leftrightarrow f(a) > f(b).$$

What if we use several characteristics or dimensions, say value $f_1(a)$, beauty $f_2(a)$, quality $f_3(a)$, and so on? We might express strict preference for a over b if and only if a is better than b according to each characteristic; that is,

$$aPb \leftrightarrow [f_1(a) > f_1(b)] \,\&\, [f_2(a) > f_2(b)] \,\&\, \cdots \,\&\, [f_t(a) > f_t(b)]. \tag{4.6}$$

Such situations often arise in comparisons of software and hardware. For instance, we might rate two different software packages on the basis of cost, speed, accuracy (measured say by 0 = poor, 1 = fair, 2 = good, 3 = excellent), and ease of use (again using poor, fair, good, excellent). Then we might definitely strictly prefer one package to another if and only if it scores higher on each "dimension." (To be precise, this works only if we use 1/cost rather than cost. Why?) We might also limit our decision on packages to which no other package is strictly preferred. A detailed example is described by Fenton and Pfleeger [1997, p. 225].

If P is defined using (4.6), it is easy to see that (X, P) is a strict partial order. The converse problem is of importance in preference theory. Suppose that we are given a strict partial order (X, P). Can we find functions $f_1, f_2, \ldots, f_t$, each f_i assigning a real number to each a in X, so that (4.6) holds? In fact, if (X, P) has dimension t, with $P = \bigcap_{i=1}^{t} L_i$, we can define f_i so that

$$aL_ib \leftrightarrow f_i(a) > f_i(b).$$

Then (4.6) follows. Hence, we can always find $f_1, f_2, \ldots, f_t$ for sufficiently large t, and the smallest number t for which we can find t such functions is at most the dimension of P. In fact, the smallest number equals the dimension except if the dimension is 2, in which case we might be able to find one function f_1 satisfying (4.6). This occurs for example if (X, P) is a strict weak order. See Baker, Fishburn, and Roberts [1972] for a discussion of the connection between strict partial orders and the multidimensional model for preference given by (4.6). For more on multidimensional utility functions, see Keeney and Raiffa [1993] or Vincke [1992]. ■

4.3.2 Chains and Antichains

In a strict partial order (X, R), a suborder that is also a strict linear order is called a *chain*. Thus, a chain is a suborder (Y, R) of (X, R) where any two distinct elements of Y are comparable. The *length* of a chain (Y, R) equals $|Y| - 1$, which is the same as the number of edges in the diagram of (Y, R). On the other hand, a suborder in which none of the elements are comparable is called an *antichain*. $L_1 = [a_1, c_2, d_1]$ and $L_2 = [a_3, b_1, c_2, d_1, e_1, f_2]$ are examples of chains in the strict partial order whose diagram is given in Figure 4.14, while (Y, R), where $Y = \{c_1, c_3, c_4\}$, is an example of an antichain. The terms "chain" and "antichain" sometimes refer only to the subset and not the suborder of the strict partial order. A chain or antichain is *maximal* if it is not part of a longer chain or antichain, respectively. Note that chain L_1 is not maximal but L_2 is maximal. Y is not a maximal antichain but $Y \cup \{c_2\}$ is maximal.

Two of the most famous theorems on the relationship between chains and antichains are Dilworth's Theorems.

Theorem 4.6 (Dilworth [1950]) If (X, R) is a strict partial order and the number of elements in a longest chain is j, there are j antichains, X_1, X_2, $\ldots, X_j$, such that

$$X = X_1 \cup X_2 \cup \cdots \cup X_j$$

and

$$X_i \cap X_k = \emptyset$$

for $i \neq k$.

Proof. Let $X_1 = \max(X, R)$, $X_2 = \max(X - X_1, R)$, $X_3 = \max(X - X_1 - X_2, R)$, $\ldots$. Continue this process until $X - X_1 - X_2 - \cdots - X_p = \emptyset$. By the definition of maximal elements, each X_i will be an antichain and

$$X_i \cap X_k = \emptyset$$

for $i \neq k$. To finish the proof we need to show that $j = p$.

Each element in a longest chain must be in a different antichain X_i. Thus, $p \geq j$. Consider X_i and X_{i+1}. For each element x in X_{i+1}, there is an element y in X_i such that yRx. Otherwise, x would have been maximal in $X - X_1 - X_2 - \cdots - X_{i-1}$ and would have been in X_i. Therefore, we can construct a chain using one element from each X_i, so $p \leq j$. Q.E.D.

The proof of the next theorem is similar and is left as an exercise (Exercise 19).

Theorem 4.7 (Dilworth [1950]) If (X, R) is a strict partial order and the size of a largest antichain is j, then there are j chains, $C_1, C_2, \ldots, C_j$, such that

$$X = C_1 \cup C_2 \cup \cdots \cup C_j$$

and

$$C_i \cap C_k = \emptyset$$

for $i \neq k$.

Example 4.11 Subset Containment Let Δ be the set $\{1, 2, \ldots, n\}$. Consider the strict partial order $\subsetneq$ on the set $S =$ subsets of Δ. What is the largest collection of subsets of Δ with the property that no member is contained in another? This is the same as asking for the largest antichain in $(S, \subsetneqq)$.

Clearly, the set of all subsets of a given size k, for any k, is an antichain of size $\binom{n}{k}$. Of these antichains, which is largest? It is easy to prove that the largest antichain of this type occurs when k is "half" of n (see Exercise 26 of Section 2.7). That is, for $0 \leq k \leq n$,

$$\binom{n}{k} \leq \binom{n}{\left\lfloor \frac{n}{2} \right\rfloor}. \tag{4.7}$$

Thus, there exists an antichain of size $\binom{n}{\left\lfloor \frac{n}{2} \right\rfloor}$. Are there any bigger ones? Sperner [1928] proved that there weren't. Consider the maximal chains in $(S, \subsetneqq)$. A maximal chain is made up of the null set, a subset of size 1, a subset of size 2, ..., and finally the set itself. There are n choices for the subset of size 1. The subset of size 2 must contain the subset of size 1 that was picked previously. Thus, there are $n-1$ choices for the second element in the subset. Continuing to build up the maximal chain in this way, we see that there are $n!$ maximal chains. Suppose that A is an antichain. If $s \in A$ and $|s| = k$, then s belongs to $k!(n-k)!$ maximal chains. (Why? See Exercise 14.) Since at most one member of A belongs to any chain, we get

$$\sum_{s \in A} |s|!(n - |s|)! \leq n!.$$

Dividing through by $n!$ we get

$$\sum_{s \in A} \frac{1}{\binom{n}{|s|}} \leq 1. \tag{4.8}$$

Combining (4.7) and (4.8) yields

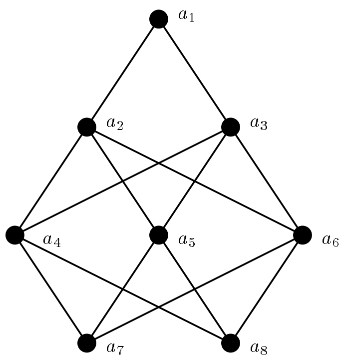
$$\sum_{s \in A} \frac{1}{\binom{n}{\left\lfloor \frac{n}{2} \right\rfloor}} \leq 1. \tag{4.9}$$

Since the left-hand side of (4.9) has no summands with s in them,

$$\sum_{s \in A} \frac{1}{\binom{n}{\left\lfloor \frac{n}{2} \right\rfloor}} = \frac{1}{\binom{n}{\left\lfloor \frac{n}{2} \right\rfloor}} \sum_{s \in A} 1 = \frac{1}{\binom{n}{\left\lfloor \frac{n}{2} \right\rfloor}} |A|.$$

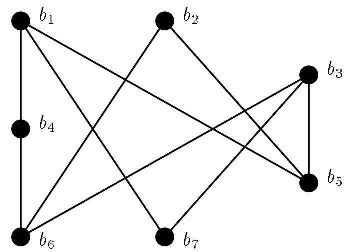
Combining this with (4.9), we get

$$\frac{|A|}{\binom{n}{\left\lfloor \frac{n}{2} \right\rfloor}} \leq 1 \quad \text{or} \quad |A| \leq \binom{n}{\left\lfloor \frac{n}{2} \right\rfloor}.$$



$H = 4, W = 3, D = 2$

(a)



$H = 3, W = 3, D = 3$

(b)

Figure 4.20: Strict partial orders and their height H, width W, and dimension D.

Therefore, all subsets of size $\left\lfloor \frac{n}{2} \right\rfloor$ form the largest antichain in the strict partial order $(S, \subsetneq)$: an antichain of size $\binom{n}{\left\lfloor \frac{n}{2} \right\rfloor}$. ■

The definitions of chain and antichain lead to parameters, *width* and *height*, that can be used to bound the dimension of a strict partial order. The *width* of (X, R), $W(X, R)$, is the size of a maximum-sized antichain, while the *height* of (X, R), $H(X, R)$, equals the length of a maximum-sized chain plus one. The strict partial orders in Figure 4.20 provide examples of these parameters.

The following lemma will be used to provide an upper bound on dimension of a strict partial order using its width.

Lemma 4.1 (Hiraguchi [1955]) Let (X, R) be a strict partial order and $C \subseteq X$ be a chain. Then there is a linear extension L'_C of R such that xL'_Cy for every $x, y \in X$, $x \in C$, $y \notin C$, and xIy.

Proof. Using Algorithm 4.1 to produce L'_C, always choose an element not in C whenever possible. The linear extensions produced in these cases will certainly satisfy the necessary conditions of the theorem. Q.E.D.

Theorem 4.8 (Dilworth [1950]) Given a strict partial order (X, R),

$$\dim(X, R) \leq W(X, R).$$

Proof. Let $w = W(X, R)$. By Theorem 4.7, there exist w chains, $C_1, C_2, \ldots, C_w$, such that $X = C_1 \cup C_2 \cup \cdots \cup C_w$ and $C_i \cap C_k = \emptyset$, for $i \neq k$. We will use these

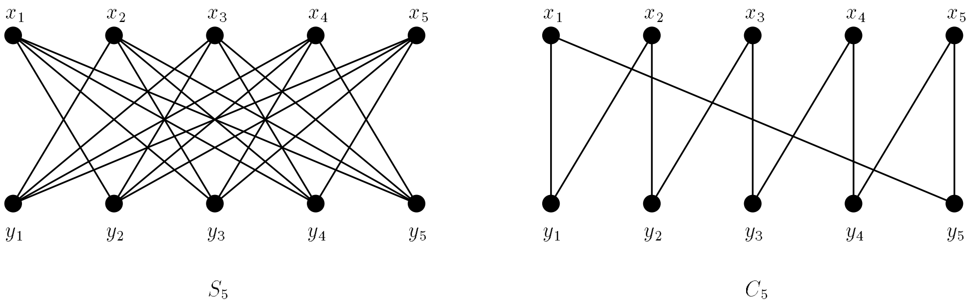


Figure 4.21: Strict partial orders S_5 and C_5.

chains to construct w linear extensions, $L_1, L_2, \ldots, L_w$, whose intersection equals (X, R).

Let L_i equal the linear extension L'_{C_i} from Lemma 4.1 using the chain C_i. Since the chains $C_1, C_2, \ldots, C_w$ are nonoverlapping, any element of X is in one and only one chain. If xIy, there must be distinct chains containing x and y. Call them C_j and C_k, respectively. Then L_j will have xL_jy and L_k will have yL_kx. This along with the fact that the linear extensions contain R assures that $(X, R) = \cap_{i=1}^{w} L_i$. Q.E.D.

Consider the two strict partial orders S_5 and C_5 in Figure 4.21, each on a set of 10 elements. The width of S_5 equals 5. So, by Theorem 4.8, the dimension of S_5 is at most 5. In fact, it equals 5, showing that the bound of Theorem 4.8 can be attained. To prove this, we will show that no four linear extensions will intersect to produce S_5. Suppose that $L_1 \cap L_2 \cap L_3 \cap L_4 = S_5$. Note that x_iIy_i, for $i = 1, 2, 3, 4, 5$. Thus, y_i precedes x_i in at least one of the linear extensions. By the pigeonhole principle, at least two of these precedences must appear in the same linear extension. Without loss of generality, suppose that y_1 precedes x_1 and y_2 precedes x_2 in L_1. Since x_2 precedes y_1 and x_1 precedes y_2 in S_5, this must also be true in L_1. Therefore, in L_1, we have $y_1L_1x_1$, $x_1L_1y_2$, $y_2L_1x_2$, $x_2L_1y_1$, and transitivity gives us $y_1L_1y_1$, a contradiction.

S_5 is just one example of a whole class of strict partial orders which have equal dimension and width. Generalizing S_5, for $n \geq 3$, let $S_n = (X, R)$ be a strict partial order with $X = \{x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n\}$. The maximal elements of S_n are $\{x_1, x_2, \ldots, x_n\}$, the minimal elements of S_n are $\{y_1, y_2, \ldots, y_n\}$, both sets are antichains, and x_iRy_j if and only if $i \neq j$. S_n will have dimension n and width n. The proof is analogous to that used for S_5.

In other cases, the bound in Theorem 4.8 may be as far from attained as desired. The second strict partial order in Figure 4.21, C_5, has width 5 and dimension 3. The proofs of these facts are left to the exercises [Exercises 10(a) and 10(b)]. Baker, Fishburn, and Roberts [1972] used the term *crown* to refer to this type of strict partial order. In general, they defined a crown order relation $C_n = (X, R)$, $n \geq 3$, as follows. Let $X = \{x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n\}$. The maximal elements of C_n are $\{x_1, x_2, \ldots, x_n\}$, the minimal elements of C_n are $\{y_1, y_2, \ldots, y_n\}$, both sets are antichains, and $x_i R y_j$ if and only if $i = j$ or $j \equiv i - 1 (\text{mod } n)$. C_n has dimension n and width 3. The proof of this fact is left to the exercises (Exercise 16). Other bounds for dimension are given in the exercises.

4.3.3 Interval Orders

The dimension of many important strict partial orders has been computed. We close this section by commenting on the dimension of one very important class of strict partial orders, the interval orders. To get an interval order, imagine again that there is a collection of alternatives among which you are choosing. You do not know the exact value for each alternative a but you estimate a range of possible values given by a closed interval $J(a) = [\alpha(a), \beta(a)]$. Then you prefer a to b if and only if you are sure that the value of a is greater than the value of b, that is, if and only if $\alpha(a) > \beta(b)$. It is easy to show (Exercise 20) that the corresponding digraph gives a strict partial order, i.e., that it is asymmetric and transitive. (In this digraph, the vertices are a family of closed real intervals, and there is an arc from interval $[a, b]$ to interval $[c, d]$ if and only if $a > d$.) Any strict partial order that arises this way is called an *interval order*. The notion of interval order is due to Fishburn [1970a]. He showed the following.

Theorem 4.9 A digraph $D = (V, A)$ is an interval order if and only if D has no loops and whenever $(a, b) \in A$ and $(c, d) \in A$, then either $(a, d) \in A$ or $(c, b) \in A$.

To illustrate Theorem 4.9, note that strict partial order C_5 of Figure 4.21 is not an interval order since (x_1, y_1) and (x_3, y_3) are arcs of C_5 but (x_1, y_3) and (x_3, y_1) are not.

In studying interval orders, which are somehow one-dimensional in nature, it came as somewhat of a surprise that their dimension as strict partial orders could be arbitrarily large. (This is the content of the following theorem, whose proof uses a generalization of the Ramsey theory results from Section 2.19.3.) This implies that if preferences arise in the very natural way that defines interval orders, we might need very many dimensions or characteristics to explain preference in the sense of (4.6) from Example 4.10.

Theorem 4.10 (Bogart, Rabinovitch, and Trotter [1976]) There are interval orders of arbitrarily high dimension.

EXERCISES FOR SECTION 4.3

1. Does the intersection of the four linear extensions in Figure 4.18 equal the order relation in Figure 4.17?
2. (a) Find two linear extensions whose intersection is the strict partial order (a) in Figure 4.20.

   (b) Find three linear extensions whose intersection is the strict partial order (b) in Figure 4.20.

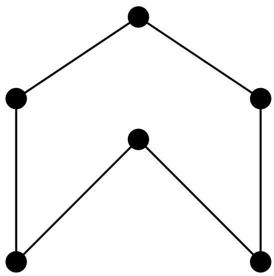


Figure 4.22: A strict partial order.

3. Six software packages (SP), A, B, C, D, E, and F, are rated on the basis of "1/cost," "speed," "accuracy," and "ease of use" using the functions f_1, f_2, f_3, and f_4, respectively. Suppose that the following data have been collected:

SP	f_1	f_2	f_3	f_4
A	1	1	2	2
B	0	0	1	0
C	3	3	2	3
D	3	2	3	3
E	0	1	2	1
F	2	3	3	2

   (a) Use (4.6) to determine the relation P of strict preference.

   (b) Define a new relation P' where $aP'b$ if and only if a scores higher than b on 3 out of 4 of the rating functions. Find P'.

4. Find the dimension of the following strict partial orders.

   (a) Figure 4.22 (b) Figure 4.23(a) (c) Figure 4.23(b)
   (d) Figure 4.23(c) (e) P_1 of Figure 4.24 (f) P_2 of Figure 4.24

5. (a) Give an example of a strict partial order that has exactly two linear extensions.

   (b) Is it possible to give an example of a strict partial order that has exactly three linear extensions?

   (c) If a strict partial order (X, R) has exactly two linear extensions, find $\dim(X, R)$.

6. (a) Use Algorithm 4.1 on the order relation S_5 in Figure 4.21 to find a linear extension. Show the results of each step of the algorithm.

   (b) How many linear extensions exist for S_5?

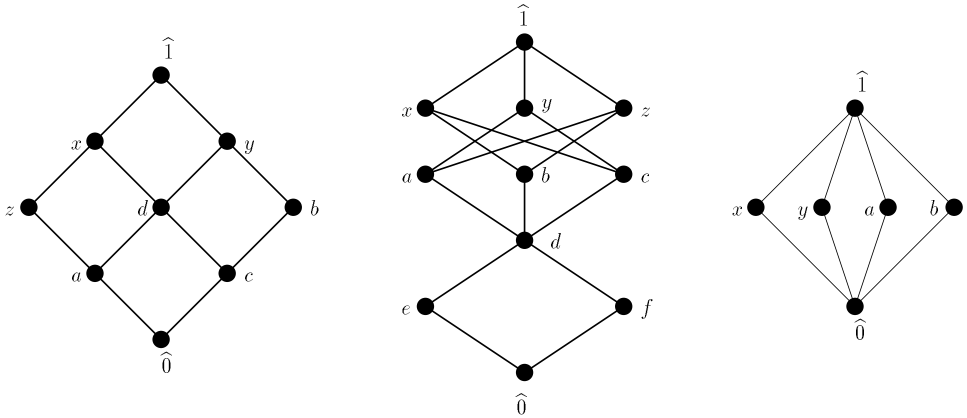
7. Find the family of linear extensions for the strict partial order (X, R), where $X = \{1, 2, 3, 4, 5, 6\}$ and $R = \{(1,3), (1,4), (2,3), (2,4), (3,4), (1,5), (2,5), (3,5), (4,5), (1,6), (2,6), (3,6), (4,6)\}$.

8. Find the height and width for each of the strict partial orders in Figure 4.23.

9. (a) Let C be the chain $[\widehat{1}, d, a]$ in the strict partial order (a) of Figure 4.23. Find a linear extension L'_C as in Lemma 4.1.

   (b) Repeat part (a) with $C = [\widehat{1}, x, d, a]$.

   (c) Repeat part (a) with $C = [y, d, \widehat{0}]$ in the strict partial order (b) of Figure 4.23.



(a) (b) (c)

Figure 4.23: Strict partial orders (that are also lattices).

(d) Repeat part (a) with $C = [x]$ in the strict partial order (c) of Figure 4.23.

10. For the strict partial order C_5 of Figure 4.21:

    (a) Prove that the width is 5.

    (b) Prove that the dimension is 3.

    (c) Show that any suborder of C_5 has dimension 2.

11. If X is the set of all subsets of $\{1, 2, 3\}$ and R is the strict partial order $\subsetneqq$, show that (X, R) has dimension 3. (Komm [1948] proved that the strict partial order $\subsetneqq$ on the set of subsets of a set X has dimension $|X|$.)

12. Hiraguchi [1955] showed that if (X, R) is a strict partial order with $|X|$ finite and at least 4, then $\dim(X, R) \leq |X|/2$. Show that dimension can be less than $\lfloor |X|/2 \rfloor$.

13. Show that every strict weak order has dimension at most 2.

14. Suppose that A is an antichain in the subset containment order of Example 4.11. If $s \in A$ and $|s| = k$, prove that s belongs to $k!(n-k)!$ maximal chains.

15. Recall the definition of strict partial order S_n on page 269. Prove that S_n has dimension and width both equaling n.

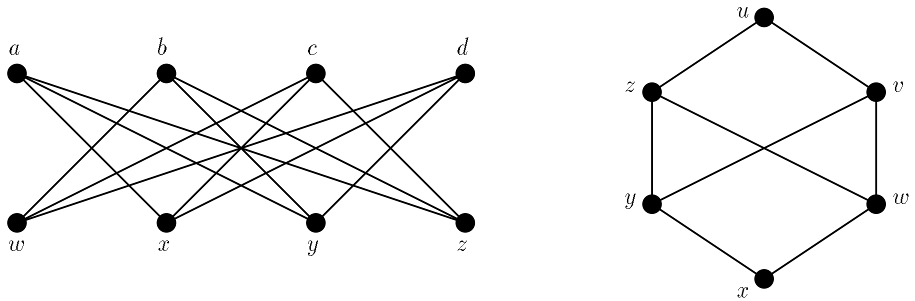
16. Recall the definition of the crown strict partial order C_n on page 269. Prove that C_n has dimension 3 and width n.

17. Find "j" chains that satisfy Dilworth's Theorem (Theorem 4.7) for the strict partial order P_1 of Figure 4.24.

18. Find an antichain of size "j" that satisfies Dilworth's Theorem (Theorem 4.6) for the strict partial order P_2 of Figure 4.24.

19. Prove Theorem 4.7.

20. Show that if V is any set of closed intervals and there is an arc from $[a, b]$ to $[c, d]$ if and only if $a > d$, then the resulting digraph is a strict partial order.



P_1 P_2

Figure 4.24: Two strict partial orders.

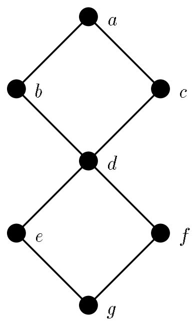


Figure 4.25: A planar strict partial order.

21. (a) Show that if a digraph is defined as in Exercise 20, the necessary condition of Theorem 4.9 is satisfied.

    (b) Use Theorem 4.9 to determine which of the digraphs of Figures 4.19–4.25 are interval orders.

22. (Trotter [1992]) A strict partial order is *planar* if its diagram can be drawn without edges crossing. [While there are planar strict partial orders with arbitrarily large dimension, Trotter [1992] cites a result of Baker showing that a lattice (as defined in Section 4.4) is planar if and only if its dimension is at most 2.]

    (a) Show that strict partial order P_1 in Figure 4.24 is a planar strict partial order by redrawing its diagram without edge crossings.

    (b) Unlike graphs, suborders of planar strict partial orders are not necessarily planar. Find a nonplanar suborder of the planar strict partial order P in Figure 4.25.

23. Recall that the Ramsey number $R(a,b)$ was defined in Section 2.19.3 and revisited in Section 3.8. Show that if a strict partial order has at least $R(a+1,b+1)$ vertices, then it either has a path of $a+1$ vertices or a set of $b+1$ vertices, no two of which are joined by arcs. (A famous theorem of Dilworth [1950] says that the same conclusion holds as long as the strict partial order has at least $ab+1$ vertices.)

4.4 LATTICES AND BOOLEAN ALGEBRAS

4.4.1 Lattices

Let (X, R) be a strict partial order. Throughout this section we use R^* to denote the binary relation on X defined by $aR^*b \Leftrightarrow a = b$ or aRb. An *upper bound* of a subset $U \subseteq X$ is an element $a \in X$ such that aR^*x for all $x \in U$. If a is an upper bound of U and bRa for all other upper bounds b of U, then a is called the *least upper bound* of U, *lub* U. The terms $\sup U$, $\vee U$, and *join* of U are also used. Similarly, we define a *lower bound* of a subset $U \subseteq X$ to be an element $a \in X$ such that xR^*a for all $x \in U$. If a is a lower bound of U and aRb for all other lower bounds b of U, then a is called the *greatest lower bound* of U, *glb* U. The terms $\inf U$, $\wedge U$, and *meet* of U are also used. If U is only a pair of elements, say $\{x, y\}$, a is the glb of U, and b is the lub of U, we can write

$$a = x \wedge y, \quad b = x \vee y.$$

Consider the strict partial order P_2 in Figure 4.24. The set $\{y, w\}$ has no lub and the set $\{z, v\}$ has no glb. However, $\{z, v, y\}$ has a lub, namely u, while the glb of $\{y, w\}$ is x. Note that u is also the lub of $\{u, z, v\}$.

Example 4.12 Lexicographic Orders (Example 4.7 Revisited) (Stanat and McAllister [1977]) In Example 4.7 we introduced a lexicographic order (X, S). Consider the situation where Σ is the alphabet $\{a, b\}$ and R is the strict linear order defined by $R = \{(a, b)\}$. Then the set of strings of the form $a^m b = aa \cdots ab$ with m a's, $m \geq 0$, has no lub. That is because the only upper bounds are strings of the form $a^m = aa \cdots a$, with m a's and, for any $n > m$, $a^m S a^n$. ■

A *lattice* is a strict partial order in which every pair of elements has a glb and a lub. Sometimes lattices are defined as strict partial orders in which every nonempty subset of elements has a glb and a lub. These definitions are equivalent when dealing with strict partial orders on finite sets. (Why? See Exercise 4.) In what follows we consider only finite lattices, so either definition is acceptable.

Some examples of lattices are:

- The subsets of a given set ordered by strict containment, $\subsetneqq$.
- The divisors of a given integer ordered by "proper divisor of."
- The transitive binary relations on a given set ordered by strict containment, $\subsetneqq$.
- The stable matchings for a given set of preference orderings ordered by "dominance."

More examples can be found in Figure 4.23.

How many maximal elements does a lattice have? Consider all of the maximal elements of a lattice (X, R), i.e., $\max(X, R)$. Then a lub $\max(X, R)$ exists and

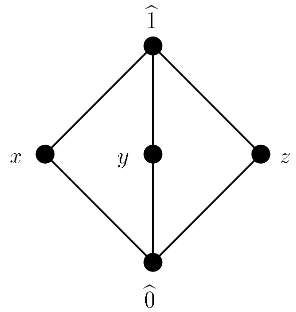


Figure 4.26: A nondistributive, complemented lattice.

must follow all of the elements in $\max(X, R)$. Therefore, $|\max(X, R)| = 1$. A similar argument applies to the minimal elements of (X, R). Thus, we have the following theorem.

Theorem 4.11 Every lattice has a maximum element and a minimum element.

As before, we denote by $\widehat{1}$ the maximum element of a lattice and by $\widehat{0}$ the minimum element. In the lattices of Figure 4.23, $\widehat{1}$ and $\widehat{0}$ are shown.

The following is a list of some basic properties of lattices whose proofs are left for the exercises. Consider a lattice (X, R). For all $a, b, c, d \in X$:

- If aRb and cRd, then
$$(a \wedge c)R(b \wedge d) \text{ and } (a \vee c)R(b \vee d) \qquad (\textit{order preserving}).$$
- $a \vee b = b \vee a$ and $a \wedge b = b \wedge a$ (*commutative*).
- $(a \vee b) \vee c = a \vee (b \vee c)$ and $(a \wedge b) \wedge c = a \wedge (b \wedge c)$ (*associative*).
- $a \wedge (a \vee b) = a \vee (a \wedge b) = a$ (*absorptive*).
- $a \vee a = a \wedge a = a$ (*idempotent*).

Notice that a distributive property is not a part of this list. The reason is that not all lattices have such a property. Two *distributive properties* of interest are

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c) \tag{4.10}$$

and

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) \tag{4.11}$$

For lattices, conditions (4.10) and (4.11) are equivalent (see Exercise 7). Lattices satisfying either of these conditions are called *distributive lattices.* For example, in Figure 4.26, $z \wedge (x \vee y) = z$ while $(z \wedge x) \vee (z \wedge y) = \widehat{0}$. Thus, this lattice is not distributive because condition (4.10) is violated.

Another property that lattices may or may not have is called complemented. An element x of a lattice with a maximum element $\widehat{1}$ and minimum element $\widehat{0}$ has a *complement* c if

$$x \vee c = \widehat{1} \text{ and } x \wedge c = \widehat{0}.$$

For example, in Figure 4.26, x has a complement in both y and z. Since $\widehat{0}$ and $\widehat{1}$ always have each other as complements in any lattice, it is the case that every element of this lattice has a complement. If every element of a lattice has a complement, the lattice is said to be *complemented.* A lattice that is both complemented and distributive is called a *Boolean algebra.* This area of lattice theory, Boolean algebras, has a number of important applications, such as to the theory of electrical switching circuits. See Gregg [1998] or Greenlaw and Hoover [1998]. We turn to it next.

4.4.2 Boolean Algebras

We have seen that an element x can have more than one complement. However, this cannot happen in a Boolean algebra.

Theorem 4.12 In a Boolean algebra, each element has one and only one complement.

Proof. Suppose that (X, R) is a Boolean algebra and that $x \in X$. Assume that x has two distinct complements y and z and use the distributive property to reach a contradiction. Details are left as an exercise (Exercise 13). Q.E.D.

If (X, R) is a Boolean algebra and $x \in X$, we let x' denote the complement of x.

Example 4.13 The $\{0,1\}$-Boolean Algebra Let $X = \{0,1\}$ and define R on X by $R = \{(1,0)\}$. Then (X, R) defines a lattice with

$$0 \vee 0 = 0,\ 0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1,\ 0 \wedge 0 = 0 \wedge 1 = 1 \wedge 0 = 0,\ 1 \wedge 1 = 1. \quad (4.12)$$

We can summarize (4.12) with the following tables:

$$\begin{array}{c|cc} \vee & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 1 \end{array} \qquad \begin{array}{c|cc} \wedge & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \end{array} \quad . \quad (4.13)$$

We have $\widehat{1} = 1$, $\widehat{0} = 0$, and, moreover, the complement of 1 is 0 and the complement of 0 is 1, which shows that (X, R) is complemented. We can summarize the latter observation by the table

$$\begin{array}{c|c} ' & \\ \hline 0 & 1 \\ 1 & 0 \end{array} \quad . \quad (4.14)$$

The distributive property is easy to demonstrate and is left to the reader (Exercise 9). Hence, (X, R) defines a Boolean algebra. ■

Example 4.14 Truth Tables Think of 0 as representing the idea that a statement is false (F) and 1 as representing the idea that a statement is true (T). We can think of $\vee$ as standing for the disjunction "or" and $\wedge$ as standing for the conjunction "and." We can replace the tables in (4.13) by the following *truth tables*:

or	F	T
F	F	T
T	T	T

and	F	T
F	F	F
T	F	T

(4.15)

The first table corresponds to the fact that the statement "p or q" (sometimes written "$p \vee q$") is true if and only if either p is true, q is true, or both are true, while the second table corresponds to the fact that the statement "p and q" (sometimes written "$p \wedge q$") is true if and only if both p and q are true. For instance, using these truth tables, we can conclude that the following statements are true:

- $65 > 23$ or Washington, DC is the capital of the United States.
- $35 + 29 = 64$ and basketball teams play only 5 players at a time.

However, the statement

$$2 + 3 = 6 \quad \text{or} \quad 13 \text{ inches} = 1 \text{ foot}$$

is false. If complement $'$ corresponds to the negation "not," table (4.14) can be written as

not	
F	T
T	F

(4.16)

Using (4.15) and (4.16), we can analyze situations in which complex statements are given. This analysis will take the form of a larger *truth table*. For instance, consider the statement "(p or q) and p" [sometimes written "$(p \vee q) \wedge p$"]. We can analyze this statement with a truth table as follows:

p	q	p or q	(p or q) and p
F	F	F	F
F	T	T	F
T	F	T	T
T	T	T	T

The first two columns give all combinations of F and T for statements p and q. The third column shows that "p or q" is true in the case where p, q is F,T or T,F or T,T, respectively, as given by the first table of (4.15). Now in the latter two cases, both "p or q" and p are T, which makes "(p or q) and p" T by the second table of (4.15). A similar analysis can be made of more complex statements. For instance, consider the statement "John lies and (Mary lies or John tells the truth)." Let $p =$ "John lies" and $q =$ "Mary lies." The truth table analysis of our statement gives us

p	q	p'	q or p'	p and (q or p')
F	F	T	T	F
F	T	T	T	F
T	F	F	F	F
T	T	F	T	T

This shows that our statement is true only in the case where both John and Mary lie.

Finally, consider the statement "p' or q." The truth table for this statement is given by

p	q	p' or q
F	F	T
F	T	T
T	F	F
T	T	T

(4.17)

This truth table also describes the logical meaning of the *conditional* statement "if p then q." For when p is true, "if p then q" can only be true when q is also true. When p is false, "if p then q" must be true since the "if" part of the statement is false. Since conditional statements arise often in various contexts, "$p \to q$" is used in place of "p' or q," i.e., "$p' \vee q$," for notational facility. ∎

Example 4.15 Logic Circuits The Boolean algebra of Example 4.13 is critical in computer science. We can think of electrical networks as designed from wires that carry two types of voltages, "high" (1) or "low" (0). (Alternatively, we can think of switches that are either "on" or "off," respectively.) We can think of combining inputs with certain kinds of "gates." An *or-gate* takes two voltages x and y as inputs and outputs a voltage $x \vee y$, while an *and-gate* takes x and y as inputs and outputs voltage $x \wedge y$, where $\vee$ and $\wedge$ are defined by (4.13). For example, an and-gate turns two high voltages into a high voltage and turns one high voltage and one low voltage into a low voltage. An *inverter* receives voltage x as input and outputs voltage x', i.e., it turns a high voltage into a low one and a low voltage into a high one.

Figure 4.27 shows a schematic or-gate, and-gate, and inverter. Figure 4.28 shows a circuit diagram. Here, we receive three inputs x_1, x_2, x_3 and calculate the output $(x_1 \vee x_2) \wedge (x_2' \wedge x_3)$. We can think of the computer corresponding to this circuit diagram as calculating the switching function

$$f(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge (x_2' \wedge x_3).$$

Using (4.13) and (4.14), we can calculate f as follows:

x_1	x_2	x_3	$(x_1 \vee x_2) \wedge (x_2' \wedge x_3)$
1	1	1	0
1	1	0	0
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

∎

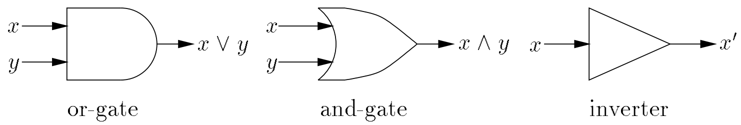


Figure 4.27: Schematic or-gate, and-gate, and inverter.

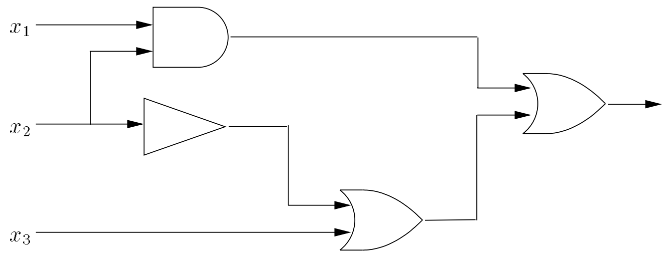


Figure 4.28: A circuit diagram that calculates $(x_1 \vee x_2) \wedge (x_2' \wedge x_3)$.

Example 4.16 Overhead Lighting Consider a room with an overhead light. Each of the three doorways leading to the room has a switch that controls the overhead light. Whenever one of the switches is changed, the light goes off if it was on and on if it was off. How are the switches wired? In most cases, the light and switches are connected by a type of electrical circuit called an *and-or circuit.* And-or circuits are logic circuits.

Switches in the room would correspond to the inputs of the and-or circuit, and the light would correspond to a lone output of the circuit. (More than one output corresponding to multiple lights could be present, but we consider only single-output circuits.) The light is on when the output of the circuit is 1 and off when it is 0. Suppose that the switches in our room are denoted by x, y, and z. Consider an and-or circuit that outputs $(x \wedge y \wedge z) \vee (x \wedge y' \wedge z') \vee (x' \wedge y \wedge z') \vee (x' \wedge y' \wedge z)$. All the possible switch voltages and their corresponding overhead light result are summarized by

x	y	z	$(x \wedge y \wedge z) \vee (x \wedge y' \wedge z') \vee (x' \wedge y \wedge z') \vee (x' \wedge y' \wedge z)$
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

(4.18)

Pick any row in (4.18). It is easy to check that a single change in x, y, or z will result in the output changing from high voltage to low voltage or low voltage to high voltage. Therefore, this and-or circuit is precisely what is needed for our overhead light example. ■

EXERCISES FOR SECTION 4.4

1. Consider the strict partial orders of Figure 4.24.
   (a) Does every pair of elements have a lub?
   (b) Does every pair of elements have a glb?
2. In each lattice of Figure 4.23, find:
   (a) lub $\{a, b, x\}$ (b) glb $\{a, b, x\}$
   (c) $x \vee y$ (d) $a \wedge b$
3. (a) Which of the strict partial orders in Figure 4.20 are lattices?
   (b) Which of the strict partial orders in Figure 4.21 are lattices?
4. Suppose that (X, R) is a strict partial order on a finite set X. Prove that every pair of elements of X has a glb and a lub if and only if every nonempty subset of X has a glb and a lub.
5. Let X be an arbitrary set and $\bigtriangledown, \bigtriangleup$ be binary operations such that
   - $\bigtriangledown$ and $\bigtriangleup$ are commutative.
   - $\bigtriangledown$ and $\bigtriangleup$ are associative.
   - absorption holds with $\bigtriangledown$ and $\bigtriangleup$.
   - $x \bigtriangledown x = x \bigtriangleup x = x$.

   Prove that (X, R) is a strict partial order where xRy if $x \bigtriangledown y = y$, for all $x, y \in X$.
6. Prove that every lattice is:
   (a) Commutative (b) Associative
   (c) Absorptive (d) Idempotent
7. For lattices, prove that conditions (4.10) and (4.11) are equivalent.
8. Suppose that (X, R) is a lattice and aRb and cRd.
   (a) Show that $(a \wedge c)R(b \wedge d)$. (b) Show that $(a \vee c)R(b \vee d)$.
9. Let (X, R) be the lattice in Example 4.13. Show that (X, R) is distributive.
10. Decide whether or not each lattice in Figure 4.23 is complemented.
11. Decide whether or not each lattice in Figure 4.23 is distributive.
12. Suppose that (X, R) is a distributive lattice. Show that if yRx, then $y \vee (x \wedge z) = x \wedge (y \vee z)$, for all $z \in X$. (Lattices with this property are called *modular*.)
13. Prove that in a Boolean algebra, each element has a unique complement. (*Hint:* Use the distributive property to show that a complement of an element must be unique.)
14. Construct the truth table for the following statements:
    (a) $p \wedge q'$ (b) $(p \wedge q) \vee (p' \wedge q')$
    (c) $(p \wedge q') \to q$ (d) $(p \wedge q) \to (p \wedge r)$

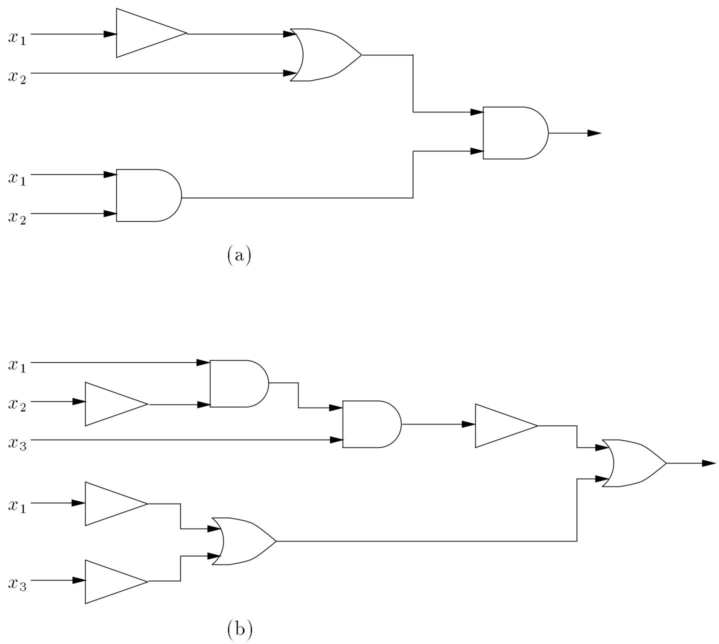


Figure 4.29: Two logic circuits.

15. Find a symbolic form and then construct the truth table for the following statements:

    (a) If Pete loves Christine, then Christine loves Pete.

    (b) Pete and Christine love each other.

    (c) It is not true that Pete loves Christine and Christine doesn't love Pete.

16. Two statements are said to be *equivalent* if one is true if and only if the other is true. One can demonstrate equivalence of two statements by constructing their truth tables and showing that there is a T in the corresponding rows. Use this idea to check if the following pairs of statements are equivalent:

    (a) $p' \wedge q'$ and $(p \vee q)'$ (b) $p \vee (q \wedge r)$ and $(p \vee q) \wedge (p \vee r)$

    (c) $p \vee (p \wedge q)$ and p (d) $(p \to q) \to r$ and $p \to (q \to r)$

    (e) $(p \vee q) \to (p \wedge q)$ and $((p \vee q) \wedge (p \wedge q)') \to (p \vee q)'$

17. Consider the conditional statement $p \to q$ and these related statements: $q \to p$ (converse), $p' \to q'$ (inverse), and $q' \to p'$ (contrapositive). Which pairs of these four statements are equivalent? (See Exercise 16.)

18. Give the switching function for the following logic circuits:

    (a) Figure 4.29(a) (b) Figure 4.29(b)

19. Draw a logic circuit for the following switching functions:

    (a) $(p \wedge q) \vee (p' \vee q)'$ (b) $(p \vee (q \wedge r)) \wedge ((p \vee q) \wedge (p \vee r))$

(c) $(p \to q) \vee q$ [see (4.17)]

20. Find an and-or circuit to model an overhead light with two switches. (*Hint:* Consider the statement A $\vee$ B, where A and B are each one of $p \wedge q$, $p' \wedge q$, $p \wedge q'$, or $p' \wedge q'$.)

REFERENCES FOR CHAPTER 4

Baker, K. A., Fishburn, P. C., and Roberts, F. S., "Partial Orders of Dimension 2, Interval Orders and Interval Graphs," *Networks, 2* (1972), 11–28.

Barberà, S., Hammond, P. J., and Seidel, C. (eds.), *Handbook of Utility Theory*, Vol. 1, Kluwer Academic Publishers, Boston, 2004.

Bentham, J., *The Principles of Morals and Legislation*, London, 1789. (Available online at: http://www.la.utexas.edu/research/poltheory/bentham/ipml/)

Bogart, K. P., Rabinovitch, I., and Trotter, W. T., "A Bound on the Dimension of Interval Orders," *J. Comb. Theory, 21* (1976), 319–328.

Dilworth, R. P., "A Decomposition Theorem for Partially Ordered Sets," *Ann. Math., 51* (1950), 161–166.

Dushnik, B., and Miller, E. W., "Partially Ordered Sets," *Amer. J. Math., 63* (1941), 600–610.

Falmagne, J.-C., *Elements of Psychophysical Theory*, Oxford University Press, New York, 1985.

Fenton, N. E., and Pfleeger, S. L., *Software Metrics*, 2nd ed., PWS Publishing Co., Boston, 1997.

Fishburn, P. C., "Intransitive Indifference with Unequal Indifference Intervals," *J. Math. Psychol., 7* (1970), 144–149. (a)

Fishburn, P. C., *Utility Theory for Decisionmaking*, Wiley, New York, 1970. (b)

Gale, D., and Shapley, L. S., "College Admissions and the Stability of Marriage," *Amer. Math. Monthly, 69* (1962), 9–15.

Greenlaw, R., and Hoover, J., *Fundamentals of the Theory of Computation: Principles and Practice*, Morgan Kaufmann Publishers, San Francisco, 1998.

Gregg, J., *Ones and Zeros: Understanding Boolean Algebra, Digital Circuits and the Logic of Sets*, IEEE, Piscataway, NJ, 1998.

Gusfield, D., *Algorithms on Strings, Trees and Sequences; Computer Science and Computational Biology*, Cambridge University Press, New York, 1997.

Gusfield, D., and Irving, R. W., *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, Cambridge, MA, 1989.

Hiraguchi, T., "On the Dimension of Orders," *Sci. Rep. Kanazawa Univ., 4* (1955), 1–20.

Keeney, R. L., and Raiffa, H., *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, Cambridge University Press, New York, 1993.

Kendall, M. G., "A Statistical Approach to Flinders Petrie's Sequence Dating," *Bull. Intern. Statist. Inst., 40* (1963), 657–680.

Kendall, M. G., "Incidence Matrices, Interval Graphs, and Seriation in Archaeology," *Pacific J. Math., 28* (1969), 565–570. (a)

Kendall, M. G., "Some Problems and Methods in Statistical Archaeology," *World Archaeology, 1* (1969), 61–76. (b)

Komm, H., "On the Dimension of Partially Ordered Sets," *Amer. J. Math., 70* (1948), 507–520.

Luce, R. D., *Utility of Gains and Losses*, Lawrence Erlbaum Associates, Mahwah, NJ, 2000.

Petrie, W. M. F., "Sequences in Prehistoric Remains," *J. Anthrop. Inst., N.S., 29* (1899), 295–301.

Petrie, W. M. F., *Diospolis Parra*, Egypt Exploration Fund, London, 1901.

Roberts, F. S., *Discrete Mathematical Models, with Applications to Social, Biological, and Environmental Problems*, Prentice Hall, Englewood Cliffs, NJ, 1976.

Roberts, F. S., "Indifference and Seriation," *Ann. N.Y. Acad. Sci., 328* (1979), 173–182. (a)

Roberts, F. S., *Measurement Theory, with Applications to Decisionmaking, Utility, and the Social Sciences*, Addison-Wesley, Reading, MA, 1979. (b) (Digitally printed version, Cambridge University Press, Cambridge, UK, 2009.)

Sperner, E., "Ein Satz über Untermengen einer endlichen Menge," *Math. Zeit., 27* (1928), 544–548.

Stanat, D. F., and McAllister, D. F., *Discrete Mathematics in Computer Science*, Prentice Hall, Englewood Cliffs, NJ, 1977.

Szpilrajn, E., "Sur l'Extension de l'Ordre Partiel," *Fund. Math., 16* (1930), 386–389.

Trotter, W. T., *Combinatorics and Partially Ordered Sets*, The Johns Hopkins University Press, Baltimore, MD, 1992.

Trotter, W. T., "Graphs and Partially Ordered Sets: Recent Results and New Directions," *Congressus Num., 116* (1996), 253–278.

Vincke, P., *Multicriteria Decision-Aid*, Wiley, Chichester, UK, 1992.

Yannakakis, M., "The Complexity of the Partial Order Dimension Problem," *SIAM J. Algebraic Discrete Methods, 3* (1982), 351–358.

PART II. The Counting Problem

Chapter 5

Generating Functions and Their Applications[1]

5.1 EXAMPLES OF GENERATING FUNCTIONS

Much of combinatorics is devoted to developing tools for counting. We have seen that it is often important to count the number of arrangements or patterns, but in practice it is impossible to list all of these arrangements. Hence, we need tools to help us in counting. In the next four chapters we present a number of tools that are useful in counting. One of the most powerful tools that we shall present is the notion of the generating function. This chapter is devoted to generating functions.

Often in combinatorics, we seek to count a quantity a_k that depends on an input or a parameter, say k. This is true, for instance, if a_k is the number of steps required to perform a computation if the input has size k. We can formalize the dependence on k by speaking of a sequence of unknown values, $a_0, a_1, a_2, \ldots, a_k, \ldots$. We seek to determine the kth term in this sequence. Generating functions provide a simple way to "encode" a sequence such as $a_0, a_1, a_2, \ldots, a_k, \ldots$, which can readily be "decoded" to find the terms of the sequence. The trick will be to see how to compute the encoding or generating function for the sequence without having the sequence. Then we can decode to find a_k. The method will enable us to determine the unknown quantity a_k in an indirect but highly effective manner.

The method of generating functions that we shall present is an old one. Its roots are in the work of De Moivre around 1720, it was developed by Euler in 1748 in connection with partition problems, and it was treated extensively in the late eighteenth and early nineteenth centuries by Laplace, primarily in connection with probability theory. In spite of its long history, the method continues to have

[1] In an elementary treatment, this chapter should be omitted. Chapters 5 and 6 are the only chapters that make use of the calculus prerequisites, except for assuming a certain level of "mathematical maturity" that comes from taking a calculus course.

widespread application, as we shall see. For a more complete treatment of generating functions, see Lando [2003], MacMahon [1960], Riordan [1980], Srivastava and Manocha [1984], or Wilf [2006]. (See also Riordan [1964].)

5.1.1 Power Series

In this chapter we use a fundamental idea from calculus, the notion of power series. The results about power series we shall need are summarized in this subsection. The reader who wants more details, including proofs of these results, can consult most any calculus book.

A *power series* is an infinite series of the form $\sum_{k=0}^{\infty} a_k x^k$. Such an infinite series always converges for $x = 0$. Either it does not converge for any other value of x, or there is a positive number R (possibly infinite) so that it converges for all x with $|x| < R$. In the latter case, the largest such R is called the *radius of convergence*. In the former case, we say that 0 is the radius of convergence. A power series $\sum_{k=0}^{\infty} a_k x^k$ can be thought of as a function of x, $f(x)$, which is defined for those values of x for which the infinite sum converges and is computed by calculating that infinite sum. In most of this chapter we shall not be concerned with matters of convergence. We simply assume that x has been chosen so that $\sum_{k=0}^{\infty} a_k x^k$ converges.[2]

Power series arise in calculus in the following way. Suppose that $f(x)$ is a function which has derivatives of all orders for all x in an interval containing 0. Then

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!} x^k = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \cdots. \tag{5.1}$$

The power series on the right-hand side of (5.1) converges for some values of x, at least for $x = 0$. The power series is called the *Maclaurin series expansion* for f or the *Taylor series expansion* for f about $x = 0$.

Some of the most famous and useful Maclaurin expansions are the following:

$$\frac{1}{1-x} = \sum_{k=0}^{\infty} x^k = 1 + x + x^2 + x^3 + \cdots, \text{ for } |x| < 1, \tag{5.2}$$

$$e^x = \sum_{k=0}^{\infty} \frac{1}{k!} x^k = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots, \text{ for } |x| < \infty, \tag{5.3}$$

$$\sin x = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)!} x^{2k+1} = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \cdots, \text{ for } |x| < \infty, \tag{5.4}$$

$$\ln(1+x) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} x^k = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \cdots, \text{ for } |x| < 1. \tag{5.5}$$

[2]This assumption can be made more precise by thinking of $\sum_{k=0}^{\infty} a_k x^k$ as simply a formal expression, a *formal power series*, rather than as a function, and by performing appropriate manipulations on these formal expressions. For details of this approach, see Niven [1969].

To show, for instance, that (5.3) is a special case of (5.1), it suffices to observe that if $f(x) = e^x$, then $f^{(k)}(x) = e^x$ for all k, and $f^{(k)}(0) = 1$. Readers should check for themselves that Equations (5.2), (5.4), and (5.5) are also special cases of (5.1).

One of the reasons that power series are so useful is that they can easily be added, multiplied, divided, composed, differentiated, or integrated. We remind the reader of these properties of power series by formulating several general principles.

Principle 1. Suppose that $f(x) = \sum_{k=0}^{\infty} a_k x^k$ and $g(x) = \sum_{k=0}^{\infty} b_k x^k$. Then $f(x)+g(x)$, $f(x)g(x)$, and $f(x)/g(x)$ can be computed by, respectively, adding term by term, multiplying out, or using long division. [This is true for division only if $g(x)$ is not zero for the values of x in question.] Specifically,

$$\begin{aligned}
f(x)+g(x) &= \sum_{k=0}^{\infty}(a_k+b_k)x^k \\
&= (a_0+b_0)+(a_1+b_1)x+(a_2+b_2)x^2+\cdots, \\
f(x)g(x) &= a_0\sum_{k=0}^{\infty} b_k x^k + a_1 x\sum_{k=0}^{\infty} b_k x^k + a_2x^2\sum_{k=0}^{\infty} b_k x^k + \cdots \\
&= a_0(b_0+b_1x+b_2x^2+\cdots)+a_1x(b_0+b_1x+b_2x^2+\cdots) \\
&\quad + a_2x^2(b_0+b_1x+b_2x^2+\cdots)+\cdots, \\
\frac{f(x)}{g(x)} &= \frac{a_0}{\sum_{k=0}^{\infty} b_k x^k} + \frac{a_1x}{\sum_{k=0}^{\infty} b_k x^k} + \frac{a_2x^2}{\sum_{k=0}^{\infty} b_k x^k} + \cdots \\
&= \frac{a_0}{b_0+b_1x+b_2x^2+\cdots} + \frac{a_1x}{b_0+b_1x+b_2x^2+\cdots} \\
&\quad + \frac{a_2x^2}{b_0+b_1x+b_2x^2+\cdots} + \cdots.
\end{aligned}$$

If the power series for $f(x)$ and $g(x)$ both converge for $|x| < R$, so do $f(x)+g(x)$ and $f(x)g(x)$. If $g(0) \neq 0$, then $f(x)/g(x)$ converges in some interval about 0.

For instance, using (5.2) and (5.3), we have

$$\begin{aligned}
\frac{1}{1-x}+e^x &= (1+x+x^2+x^3+\cdots)+\left(1+x+\frac{1}{2!}x^2+\frac{1}{3!}x^3+\cdots\right) \\
&= (1+1)+(1+1)x+\left(1+\frac{1}{2!}\right)x^2+\left(1+\frac{1}{3!}\right)x^3+\cdots \\
&= \sum_{k=0}^{\infty}\left(1+\frac{1}{k!}\right)x^k.
\end{aligned}$$

Also,

$$\begin{aligned}
\frac{1}{1-x}e^x &= (1+x+x^2+x^3+\cdots)\left(1+x+\frac{1}{2!}x^2+\frac{1}{3!}x^3+\cdots\right) \\
&= 1\left(1+x+\frac{1}{2!}x^2+\cdots\right)+x\left(1+x+\frac{1}{2!}x^2+\cdots\right) \\
&\quad +x^2\left(1+x+\frac{1}{2!}x^2+\cdots\right)+\cdots
\end{aligned}$$

$$= \quad 1 + 2x + \frac{5}{2}x^2 + \cdots.$$

Power series are also easy to compute under composition of functions.

Principle 2. If $f(x) = g(u(x))$ and if we know that $g(u) = \sum_{k=0}^{\infty} a_k u^k$, we have $f(x) = \sum_{k=0}^{\infty} a_k [u(x)]^k$.[3]

Thus, setting $u = x^4$ in Equation (5.5) gives us

$$\ln(1 + x^4) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} x^{4k}.$$

Principle 2 generalizes to the situation where we have a power series for $u(x)$.

Principle 3. If a power series $f(x) = \sum_{k=0}^{\infty} a_k x^k$ converges for all $|x| < R$ with $R > 0$, the derivative and antiderivative of $f(x)$ can be computed by differentiating and integrating term by term. Namely,

$$\frac{df}{dx}(x) = \frac{d}{dx}\left(\sum_{k=0}^{\infty} a_k x^k\right) = \sum_{k=0}^{\infty} \frac{d}{dx}(a_k x^k) = \sum_{k=1}^{\infty} k a_k x^{k-1} \tag{5.6}$$

and

$$\int_0^x f(t)\,dt = \int_0^x \left(\sum_{k=0}^{\infty} a_k t^k\right) dt = \sum_{k=0}^{\infty} \int_0^x a_k t^k \,dt = \sum_{k=0}^{\infty} \frac{1}{k+1} a_k x^{k+1}. \tag{5.7}$$

The power series in (5.6) and (5.7) also converge for $|x| < R$.

For instance, since

$$\frac{1}{(1-x)^2} = \frac{d}{dx}\left[\frac{1}{1-x}\right],$$

we see from (5.2) and (5.6) that

$$\frac{1}{(1-x)^2} = \sum_{k=1}^{\infty} k x^{k-1} = 1 + 2x + 3x^2 + 4x^3 + \cdots.$$

5.1.2 Generating Functions

Suppose that we are interested in computing the kth term in a sequence (a_k) of numbers. We shall use the convention that (a_k) refers to the sequence and a_k, written without parentheses, to the kth term. The *(ordinary) generating function* for the sequence (a_k) is defined to be

$$G(x) = \sum_{k=0}^{\infty} a_k x^k = a_0 x^0 + a_1 x^1 + a_2 x^2 + \cdots. \tag{5.8}$$

[3] If the power series for $g(u)$ converges for $|u| < S$ and $|u(x)| < S$ whenever $|x| < R$, then the power series for $f(x)$ converges for all $|x| < R$.

The sum is finite if the sequence is finite and infinite if the sequence is infinite. In the latter case, we will think of x as having been chosen so that the sum in (5.8) converges.

Example 5.1 Suppose that $a_k = \binom{n}{k}$, for $k = 0, 1, \ldots, n$. Then the ordinary generating function for the sequence (a_k) is

$$G(x) = \binom{n}{0} + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n.$$

By the binomial expansion (Theorem 2.7),

$$G(x) = (1+x)^n.$$

The advantage of what we have done is that we have expressed $G(x)$ in a simple, closed form (encoded form). Knowing this simple form for $G(x)$, one can now possibly derive a_k simply by remembering this closed form for $G(x)$ and decoding, that is, expanding out, and searching for the coefficient of x^k. Even more useful is the fact that, as we have observed before, often we are able to find $G(x)$ without knowing a_k and then to solve for a_k by expanding out. ■

Example 5.2 Suppose that $a_k = 1$, for $k = 0, 1, 2, \ldots$. Then

$$G(x) = 1 + x + x^2 + \cdots.$$

By Equation (5.2),

$$G(x) = \frac{1}{1-x}$$

provided that $|x| < 1$. Again the reader will note the closed form for $G(x)$. ■

Example 5.3 Often we will know the generating function but not the sequence. We will try to "recover" the sequence from the generating function. For example, suppose that

$$G(x) = \sum_{k=0}^{\infty} a_k x^k$$

and we know

$$G(x) = \frac{x^2}{1-x}.$$

What is a_k? Using Equation (5.2), we have for $|x| < 1$,

$$\begin{aligned} G(x) &= x^2\left[\frac{1}{1-x}\right] \\ &= x^2(1 + x + x^2 + \cdots) \\ &= x^2 + x^3 + x^4 + \cdots. \end{aligned}$$

Hence,

$$(a_k) = (0, 0, 1, 1, 1, \ldots).$$

In this chapter and Chapter 6 we study a variety of techniques for expanding out $G(x)$ to obtain the desired sequence (a_k). ■

Example 5.4 Suppose that $a_k = 1/k!$, for $k = 0, 1, 2, \ldots$. Then

$$G(x) = \frac{1}{0!} + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots.$$

By Equation (5.3), $G(x) = e^x$ for all values of x. ■

Example 5.5 Suppose that $G(x) = x\sin(x^2)$ is the ordinary generating function for the sequence (a_k). To find a_k, use Equation (5.4), substitute x^2 for x, and multiply by x, to find that

$$\begin{aligned} G(x) &= x\left[x^2 - \frac{1}{3!}(x^2)^3 + \frac{1}{5!}(x^2)^5 - \cdots\right] \\ &= x^3 - \frac{1}{3!}x^7 + \frac{1}{5!}x^{11} - \cdots. \end{aligned}$$

Thus, we see that a_k is the kth term of the sequence

$$\left(0, 0, 0, 1, 0, 0, 0, -\frac{1}{3!}, 0, 0, 0, \frac{1}{5!}, 0, \ldots\right).$$

■

Example 5.6 Suppose that $G(x) = \cos x$ is the ordinary generating function for the sequence (a_k). Since $G(x)$ has derivatives of all orders, we can expand out in a Maclaurin series, by calculating $f^{(k)}(0)$ for all k, and we see that

$$G(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!}x^{2k} = 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \cdots. \tag{5.9}$$

The verification of this is left as an exercise. An alternative approach is to observe that $G(x) = d(\sin x)/dx$, and so to use Equation (5.4). Then we see that

$$\begin{aligned} G(x) &= \frac{d}{dx}(\sin x) \\ &= \frac{d}{dx}\left[x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \cdots\right] \\ &= \frac{d}{dx}[x] + \frac{d}{dx}\left[-\frac{1}{3!}x^3\right] + \frac{d}{dx}\left[\frac{1}{5!}x^5\right] + \cdots \\ &= 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \cdots, \end{aligned}$$

which agrees with Equation (5.9). ■

Example 5.7 If $(a_k) = (1, 1, 1, 0, 1, 1, \ldots)$, the ordinary generating function is given by

$$\begin{aligned} G(x) &= 1 + x + x^2 + x^4 + x^5 + \cdots \\ &= (1 + x + x^2 + x^3 + x^4 + x^5 + \cdots) - x^3 \\ &= \frac{1}{1-x} - x^3. \end{aligned}$$

■

Example 5.8 If $(a_k) = (1/2!, 1/3!, 1/4!, \ldots)$, the ordinary generating function is given by

$$\begin{aligned} G(x) &= \frac{1}{2!} + \frac{1}{3!}x + \frac{1}{4!}x^2 + \cdots \\ &= \frac{1}{x^2}\left(\frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \cdots\right) \\ &= \frac{1}{x^2}\left(e^x - 1 - x\right). \end{aligned}$$

■

Example 5.9 The Number of Labeled Graphs In Section 3.1.3 we counted the number $L(n, e)$ of labeled graphs with n vertices and e edges, $n \geq 2, e \leq C(n, 2)$. If n is fixed and we let $a_k = L(n, k), k = 0, 1, \ldots, C(n, 2)$, let us consider the generating function

$$G_n(x) = \sum_{k=0}^{C(n,2)} a_k x^k.$$

Note that in Section 3.1.3 we computed $L(n, e) = C(C(n, 2), e)$. Hence, if $r = C(n, 2)$,

$$G_n(x) = \sum_{k=0}^{r} C(r, k)x^k.$$

By the binomial expansion (Theorem 2.7), we have

$$G_n(x) = (1 + x)^r = (1 + x)^{C(n,2)}, \tag{5.10}$$

which is a simple way to summarize our knowledge of the numbers $L(n, e)$. In particular, from (5.10) we can derive a formula for the number $L(n)$ of labeled graphs of n vertices. For

$$L(n) = \sum_{k=0}^{r} C(r, k),$$

which is $G_n(1)$. Thus, taking $x = 1$ in (5.10) gives us

$$L(n) = 2^{C(n,2)},$$

which is the result we derived in Section 3.1.3. ■

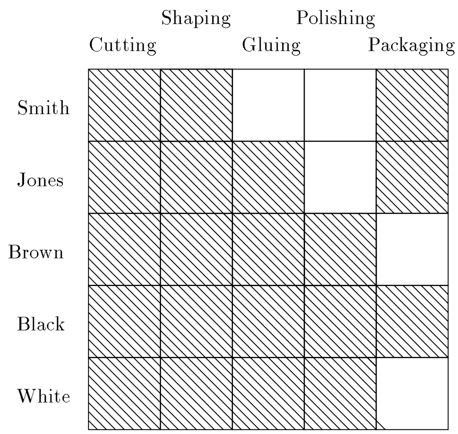


Figure 5.1: A board corresponding to a job assignment problem.

Example 5.10 Rook Polynomials: Job Assignments and Storing Computer Programs Suppose that B is any $n \times m$ board such as those in Figures 5.1 and 5.2, with certain squares forbidden and others acceptable, the acceptable ones being darkened. Let $r_k(B)$ be the number of ways to choose k acceptable (darkened) squares, no two of which lie in the same row and no two of which lie in the same column. We can think of B as part of a chess board. A *rook* is a piece that can travel either horizontally or vertically on the board. Thus, one rook is said to be able to *take* another if the two are in the same row or the same column. We wish to place k rooks on B in acceptable squares in such a way that no rook can take another. Thus, $r_k(B)$ counts the number of ways k nontaking rooks can be placed in acceptable squares of B.

The 5×5 board in Figure 5.1 arises from a job assignment problem. The rows correspond to workers, the columns to jobs, and the i, j position is darkened if worker i is suitable for job j. We wish to determine the number of ways in which each worker can be assigned to one job, no more than one worker per job, so that a worker only gets a job to which he or she is suited. It is easy to see that this is equivalent to the problem of computing $r_5(B)$.

The 5×7 board in Figure 5.2 arises from a problem of storing computer programs. The i, j position is darkened if storage location j has sufficient storage capacity for program i. We wish to assign each program to a storage location with sufficient storage capacity, at most one program per location. The number of ways this can be done is again given by $r_5(B)$. We shall compute $r_5(B)$ in these two examples in the text and exercises of Section 7.1.

The expression

$$R(x, B) = r_0(B) + r_1(B)x + r_2(B)x^2 + \cdots$$

is called the *rook polynomial* for the board B. The rook polynomial is indeed a polynomial in x, since $r_k(B) = 0$ for k larger than the number of acceptable squares.

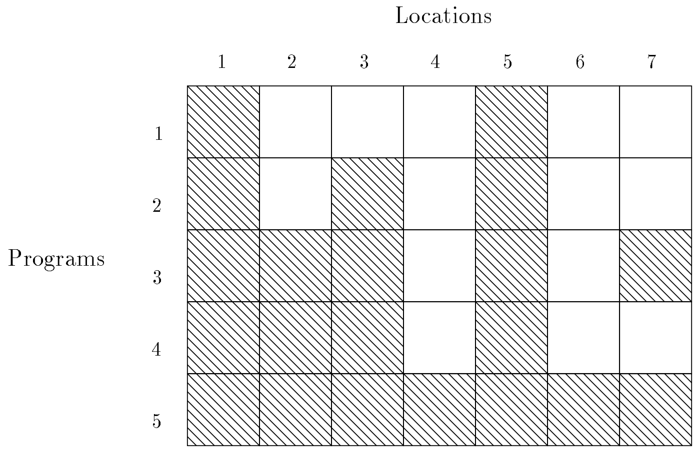


Figure 5.2: A board corresponding to a computer storage problem.

The rook polynomial is just the ordinary generating function for the sequence

$$(r_0(B), r_1(B), r_2(B), \ldots).$$

As with generating functions in general, we shall find methods for computing the rook polynomial without explicitly computing the coefficients $r_k(B)$, and then we shall be able to compute these coefficients from the polynomial.

To give some examples, consider the two boards B_1 and B_2 of Figure 5.3. In board B_1, there is one way to place no rooks (this will be the case for any board), two ways to place one rook (use either darkened square), one way to place two rooks (use both darkened squares), and no way to place more than two rooks. Thus,

$$R(x, B_1) = 1 + 2x + x^2.$$

In board B_2, there is again one way to place no rooks, four ways to place one rook (use any darkened square), two ways to place two rooks (use the diagonal squares or the nondiagonal squares), and no way to place more than two rooks. Thus,

$$R(x, B_2) = 1 + 4x + 2x^2.$$ ■

EXERCISES FOR SECTION 5.1

1. For each of the following functions, find its Maclaurin expansion by computing the derivatives $f^{(k)}(0)$.

   (a) $\cos x$ (b) e^{3x} (c) $\sin(2x)$ (d) $x^3 + 4x + 7$
   (e) $x^2 + e^x$ (f) xe^x (g) $\ln(1 + 4x)$ (h) $x \sin x$

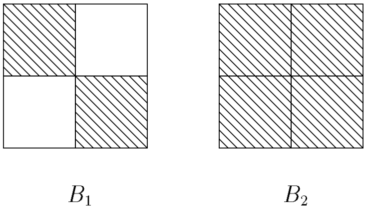


Figure 5.3: Two boards.

2. For each of the following functions, use known Maclaurin expansions to find the Maclaurin expansion, by adding, composing, differentiating, and so on.

(a) $x^3 + \frac{1}{1-x}$ (b) $x^2 \frac{1}{1-x}$ (c) $\sin(x^4)$ (d) $\frac{1}{4-x}$

(e) $\sin(x^2+x+1)$ (f) $\frac{1}{(1-x)^3}$ (g) $5e^x + e^{3x}$ (h) $\ln(1-x)$

(i) $\ln(1+3x)$ (j) $x^3 \sin(x^5)$ (k) $\ln(1+x^2)$ (l) $\frac{1}{1-2x}e^x$

3. For the following sequences, find the ordinary generating function and simplify if possible.

(a) $(1,1,1,0,0,\ldots)$ (b) $(1,0,2,3,4,0,0,\ldots)$ (c) $(5,5,5,\ldots)$

(d) $(1,0,0,1,1,\ldots)$ (e) $(0,0,1,1,1,1,1,\ldots)$ (f) $(0,0,4,4,4,\ldots)$

(g) $(1,1,2,1,1,1,\ldots)$ (h) $(a_k) = \left(\frac{3}{k!}\right)$ (i) $(a_k) = \left(\frac{3^k}{k!}\right)$

(j) $\left(0,0,\frac{1}{2!},\frac{1}{3!},\frac{1}{4!},\cdots\right)$ (k) $\left(1,1,\frac{1}{2!},\frac{1}{3!},\frac{1}{4!},\cdots\right)$ (l) $\left(\frac{1}{3!},\frac{1}{4!},\frac{1}{5!},\cdots\right)$

(m) $\left(3,-\frac{3}{2},\frac{3}{3},-\frac{3}{4},\cdots\right)$ (n) $(1,0,1,0,1,0,\ldots)$ (o) $(0,1,0,3,0,5,\ldots)$

(p) $\left(2,0,-\frac{2}{3!},0,\frac{2}{5!},\cdots\right)$ (q) $\left(1,-1,\frac{1}{2!},-\frac{1}{3!},\frac{1}{4!},\cdots\right)$ (r) $\left(0,-1,\frac{1}{2},-\frac{1}{3},\frac{1}{4},\cdots\right)$

4. Find the sequence whose ordinary generating function is given as follows:

(a) $(x+5)^2$ (b) $(1+x)^4$ (c) $\frac{x^3}{1-x}$

(d) $\frac{1}{1-3x}$ (e) $\frac{1}{1+8x}$ (f) e^{6x}

(g) $1+\frac{1}{1-x}$ (h) $5+e^{2x}$ (i) $x\sin x$

(j) $x^3+x^4+e^{2x}$ (k) $\frac{1}{1-x^2}$ (l) $2x+e^{-x}$

(m) e^{-2x} (n) $\sin 3x$ (o) $x^2\ln(1+2x)+e^x$

(p) $\frac{1}{1+x^2}$ (q) $\cos 3x$ (r) $\frac{1}{(1+x)^2}$

5. Suppose that the ordinary generating function for the sequence (a_k) is given as follows. In each case, find a_3.

(a) $(x-7)^3$ (b) $\frac{14}{1-x}$ (c) $\ln(1-2x)$ (d) e^{5x}

6. In each case of Exercise 5, find a_4.
7. Professor Jones wants to teach Calculus I or Linear Algebra, Professor Smith wants to teach Linear Algebra or Combinatorics, and Professor Green wants to teach Calculus I or Combinatorics. Each professor can be assigned to teach at most one course, with no more than one professor per course, and a professor only gets a course that he or she wants to teach. Set up a generating function and use it to answer the following questions.

   (a) In how many ways can we assign one professor to a course?

   (b) In how many ways can we assign two professors to courses?

   (c) In how many ways can we assign three professors to courses?

8. Suppose that worker a is suitable for jobs 3, 4, 5, worker b is suitable for jobs 2, 3, and worker c is suitable for jobs 1, 5. Also, each worker can be assigned to at most one job, no more than one worker per job, and a worker only gets a job to which he or she is suited. Set up a generating function and use it to answer the following questions.

   (a) In how many ways can we assign one worker to a job?

   (b) In how many ways can we assign two workers to jobs?

   (c) In how many ways can we assign three workers to jobs?

9. Suppose that T_n is the number of rooted (unlabeled) trees of n vertices. The ordinary generating function $T(x) = \sum T_n x^n$ is given by

$$T(x) = x + x^2 + 2x^3 + 4x^4 + 9x^5 + 20x^6 + \cdots.$$

   (Riordan [1980] computes T_n for $n \leq 26$. More values of T_n are now known.) Verify the coefficients of x, x^2, x^3, x^4, x^5, and x^6.

10. Let $M(n, a)$ be the number of labeled digraphs with n vertices and a arcs and let $M(n)$ be the number of labeled digraphs with n vertices (see Section 3.1.3). If n is fixed, let $b_k = M(n, k)$ and let

$$D_n(x) = \sum_{k=0}^{n(n-1)} b_k x^k.$$

   (a) Find a simple, closed-form expression for $D_n(x)$.

   (b) Use this expression to derive a formula for $M(n)$.

11. Suppose that $c_k = R(n, k)$ is the number of labeled graphs with a certain property P and having n vertices and k edges, and $R(n)$ is the number of labeled graphs with property P and n vertices. Suppose that

$$G_n(x) = \sum_{k=0}^{\infty} c_k x^k$$

   is the ordinary generating function and we know that $G_n(x) = (1 + x + x^2)^n$. Find $R(n)$.

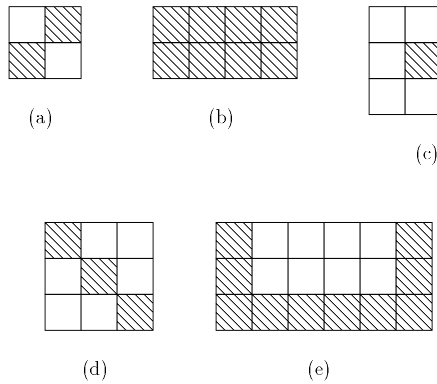


Figure 5.4: Boards for Exercise 13, Section 5.1.

1	2	3	4	5
2	3	1	5	4

Figure 5.5: A 2×5 Latin rectangle.

12. Suppose that $d_k = S(n, k)$ is the number of labeled digraphs with a certain property Q and having n vertices and k arcs, and $S(n)$ is the number of labeled digraphs with property Q and n vertices and at least two arcs. Let the ordinary generating function be given by
$$H_n(x) = \sum_{k=0}^{\infty} d_k x^k.$$
Suppose we know that $H_n(x) = (1 + x^2)^{n+5}$. Find $S(n)$.

13. Compute the rook polynomial for each of the boards of Figure 5.4.

14. Compute the rook polynomial for the $n \times n$ chess board with all squares darkened if n is

    (a) 3 (b) 4 (c) 6 (d) 8.

15. A *Latin rectangle* is an $r \times s$ array with entries $1, 2, \ldots, n$, so that no two entries in any row or column are the same. A Latin square (Example 1.1) is a Latin rectangle with $r = s = n$. One way to build a Latin square is to build it up one row at a time, adding rows successively to Latin rectangles. In how many ways can we add a third row to the Latin rectangle of Figure 5.5? Set this up as a rook polynomial problem by observing what symbols can still be included in the jth column. You do not have to solve this problem.

16. *Use rook polynomials* to count the number of permutations of $1, 2, 3, 4$ in which 1 is not in the second position, 2 is not in the fourth position, and 3 is not in the first or fourth position.

17. Show that if board B' is obtained from board B by deleting rows or columns with no darkened squares, then $r_k(B) = r_k(B')$.

5.2 OPERATING ON GENERATING FUNCTIONS

A sequence defines a unique generating function and a generating function defines a unique sequence; we will be able to pass back and forth between sequences and generating functions. It will be useful to compile a "library" of basic generating functions and their corresponding sequences. Our list can start with the generating functions $1/(1-x)$, e^x, $\sin x$, $\ln(1+x)$, whose corresponding sequences can be derived from Equations (5.2)–(5.5). By operating on these generating functions as in Section 5.1.1, namely by adding, multiplying, dividing, composing, differentiating, or integrating, we can add to our basic list. In this section we do so.

In this section we observe how the various operations on generating functions relate to operations on the corresponding sequences. We start with some simple examples. Suppose that (a_k) is a sequence with ordinary generating function $A(x) = \sum_{k=0}^{\infty} a_k x^k$. Then multiplying $A(x)$ by x corresponds to shifting the sequence one place to the right and starting with 0. For $xA(x) = \sum_{k=0}^{\infty} a_k x^{k+1}$ is the ordinary generating function for the sequence $(0, a_0, a_1, a_2, \ldots)$. Similarly, multiplying $A(x)$ by $1/x$ and subtracting a_0/x corresponds to shifting the sequence one place to the left and deleting the first term, for

$$\frac{1}{x}A(x) - \frac{a_0}{x} = \sum_{k=0}^{\infty} a_k x^{k-1} - \frac{a_0}{x} = \sum_{k=1}^{\infty} a_k x^{k-1} = \sum_{k=0}^{\infty} a_{k+1} x^k.$$

To illustrate these two results, note that since $A(x) = e^x$ is the ordinary generating function for the sequence

$$\left(1, 1, \frac{1}{2!}, \frac{1}{3!}, \cdots\right),$$

xe^x is the ordinary generating function for the sequence

$$\left(0, 1, 1, \frac{1}{2!}, \frac{1}{3!}, \cdots\right),$$

and $(1/x)e^x - 1/x$ is the ordinary generating function for the sequence

$$\left(1, \frac{1}{2!}, \frac{1}{3!}, \cdots\right).$$

Similarly, by Equation (5.6), differentiating $A(x)$ with respect to x corresponds to multiplying the kth term of (a_k) by k and shifting by one place to the left. Thus, we saw in Section 5.1.1 that since

$$\frac{1}{(1-x)^2} = \frac{d}{dx}\left[\frac{1}{1-x}\right],$$

$1/(1-x)^2$ is the ordinary generating function for the sequence $(1, 2, 3, \ldots)$.

Suppose that (a_k) and (b_k) are sequences with ordinary generating functions $A(x) = \sum_{k=0}^{\infty} a_k x^k$ and $B(x) = \sum_{k=0}^{\infty} b_k x^k$, respectively. Since two power series can be added term by term, we see that $C(x) = A(x) + B(x)$ is the ordinary generating function for the sequence (c_k) whose kth term is $c_k = a_k + b_k$. This sequence (c_k) is called the *sum* of (a_k) and (b_k) and is denoted $(a_k) + (b_k)$. Thus,

$$\frac{1}{1-x} + e^x$$

is the ordinary generating function for

$$(1, 1, 1, \ldots) + \left(1, 1, \frac{1}{2!}, \frac{1}{3!}, \cdots\right) = \left(2, 2, 1 + \frac{1}{2!}, 1 + \frac{1}{3!}, \cdots\right).$$

From the point of view of combinatorics, the most interesting case arises from multiplying two generating functions. Suppose that

$$C(x) = A(x)B(x), \tag{5.11}$$

where $A(x), B(x)$, and $C(x)$ are the ordinary generating functions for the sequences $(a_k), (b_k)$, and (c_k), respectively. Does it follow that $c_k = a_k b_k$ for all k? Let $A(x) = 1 + x$ and $B(x) = 1 + x$. Then $C(x) = A(x)B(x)$ is given by $1 + 2x + x^2$. Now $(c_k) = (1, 2, 1, 0, 0, \ldots)$ and $(a_k) = (b_k) = (1, 1, 0, 0, \ldots)$, so $c_0 = a_0 b_0$ but $c_1 \neq a_1 b_1$. Thus, $c_k = a_k b_k$ does not follow from (5.11). What we can observe is that if we multiply $A(x)$ by $B(x)$, we obtain a term $c_k x^k$ by combining terms $a_j x^j$ from $A(x)$ with terms $b_{k-j} x^{k-j}$ from $B(x)$. Thus, for all k,

$$c_k = a_0 b_k + a_1 b_{k-1} + \cdots + a_{k-1} b_1 + a_k b_0. \tag{5.12}$$

This is easy to check in the case where both $A(x)$ and $B(x)$ are $1 + x$. Note that if $k = 0$, (5.12) says that $c_0 = a_0 b_0$. Note also that (5.12) for all k implies (5.11). If (5.12) holds for all k, we say the sequence (c_k) is the *convolution* of the two sequences (a_k) and (b_k), and we write $(c_k) = (a_k) * (b_k)$. Our results are summarized as follows.

Theorem 5.1 Suppose that $A(x), B(x)$, and $C(x)$ are, respectively, the ordinary generating functions for the sequences $(a_k), (b_k)$, and (c_k). Then
(a) $C(x) = A(x) + B(x)$ if and only if $(c_k) = (a_k) + (b_k)$.
(b) $C(x) = A(x)B(x)$ if and only if $(c_k) = (a_k) * (b_k)$.

Example 5.11 Suppose that $b_k = 1$, for all k. Then (5.12) becomes

$$c_k = a_0 + a_1 + \cdots + a_k.$$

The generating function $B(x)$ is given by $B(x) = 1/(1-x) = (1-x)^{-1}$. Hence, by Theorem 5.1,

$$C(x) = A(x)(1-x)^{-1}.$$

This is the generating function for the sum of the first k terms of a series. For instance, suppose that (a_k) is the sequence $(0, 1, 1, 0, 0, \ldots)$. Then $A(x) = x + x^2$ and

$$\begin{aligned} C(x) &= (x + x^2)[1 + x + x^2 + \cdots] \\ &= x + 2x^2 + 2x^3 + 2x^4 + \cdots. \end{aligned}$$

We conclude that

$$(x + x^2)(1 - x)^{-1}$$

is the ordinary generating function for the sequence (c_k) given by $(0, 1, 2, 2, \ldots)$. This can be checked by noting that

$$a_0 = 0, \; a_0 + a_1 = 1, \; a_0 + a_1 + a_2 = 2, \; a_0 + a_1 + a_2 + a_3 = 2, \; \ldots. \qquad \blacksquare$$

Example 5.12 If $A(x)$ is the generating function for the sequence (a_k), then $A^2(x)$ is the generating function for the sequence (c_k) where

$$c_k = a_0 a_k + a_1 a_{k-1} + \cdots + a_{k-1} a_1 + a_k a_0.$$

This result will also be useful in the enumeration of chemical isomers by counting trees in Section 6.4. In particular, if $a_k = 1$ for all k, then $A(x) = (1 - x)^{-1}$. It follows that

$$C(x) = A^2(x) = (1 - x)^{-2}$$

is the generating function for (c_k) where $c_k = k + 1$. We have obtained this result before by differentiating $(1 - x)^{-1}$. $\blacksquare$

Example 5.13 Suppose that

$$G(x) = \frac{1 + x + x^2 + x^3}{1 - x}$$

is the ordinary generating function for a sequence (a_k). Can we find a_k? We can write

$$G(x) = (1 + x + x^2 + x^3)(1 - x)^{-1}.$$

Now $1 + x + x^2 + x^3$ is the ordinary generating function for the sequence

$$(b_k) = (1, 1, 1, 1, 0, 0, \ldots)$$

and $(1 - x)^{-1}$ is the ordinary generating function for the sequence

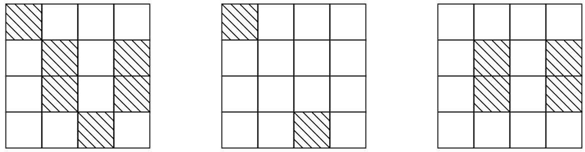
$$(c_k) = (1, 1, 1, \ldots).$$

Thus, $G(x)$ is the ordinary generating function for the convolution of these two sequences, that is,

$$a_k = b_0 c_k + b_1 c_{k-1} + \cdots + b_k c_0 = b_0 + b_1 + \cdots + b_k.$$

It is easy to show from this that

$$(a_k) = (1, 2, 3, 4, 4, \ldots). \qquad \blacksquare$$



B $\qquad$ B_I $\qquad$ B_J

Figure 5.6: B_I and B_J decompose B.

Example 5.14 A Reduction for Rook Polynomials In computing rook polynomials, it is frequently useful to reduce a complicated computation to a number of smaller ones, a trick we have previously encountered in connection with chromatic polynomials in Section 3.4. Exercise 17 of Section 5.1 shows one such reduction. Here we present another. Suppose that I is a set of darkened squares in a board B and B_I is the board obtained from B by lightening the darkened squares not in I. Suppose that the darkened squares of B are partitioned into two sets I and J so that no square in I lies in the same row or column as any square of J. In this case, we say that B_I and B_J *decompose* B. Figure 5.6 illustrates this situation.

If B_I and B_J decompose B, then since all acceptable squares fall in B_I or B_J, and no rook of I can take a rook of J, or vice versa, to place k nontaking rooks on B, we place p nontaking rooks on B_I, and then $k-p$ nontaking rooks on B_J, for some p. Thus,

$$\begin{aligned} r_k(B) &= r_0(B_I)r_k(B_J) + r_1(B_I)r_{k-1}(B_J) + \\ &\quad \cdots + r_p(B_I)r_{k-p}(B_J) + \cdots + r_k(B_I)r_0(B_J). \end{aligned}$$

This implies that the sequence $(r_k(B))$ is simply the convolution of the two sequences $(r_k(B_I))$ and $(r_k(B_J))$. Hence,

$$R(x,B) = R(x,B_I)R(x,B_J).$$

■

EXERCISES FOR SECTION 5.2

1. In each of the following, the function is the ordinary generating function for a sequence (a_k). Find this sequence.

   (a) $x\ln(1+x)$ (b) $\dfrac{1}{x}\sin x$ (c) $x^4\ln(1+x)$

   (d) $\dfrac{1}{x^4}\sin x$ (e) $\dfrac{5}{1-x}+x^3+3x+4$ (f) $\dfrac{x}{1-7x}+\dfrac{4}{1-x}$

   (g) $\dfrac{1}{1-x^2}+6x+5$ (h) $\dfrac{1}{2}\left(e^x-e^{-x}\right)$

2. For each of the following functions $A(x)$, suppose that $B(x) = xA'(x)$. Find the sequence for which $B(x)$ is the ordinary generating function.

(a) $\dfrac{1}{1-x}$ (b) e^{3x} (c) $\cos x$ (d) $\ln(1+x)$

3. In each of the following, find a formula for the convolution of the two sequences.

   (a) $(1, 1, 1, \ldots)$ and $(1, 1, 1, \ldots)$
   (b) $(1, 1, 1, \ldots)$ and $(0, 1, 2, 3, \ldots)$
   (c) $(1, 1, 0, 0, \ldots)$ and $(0, 1, 2, 3, \ldots)$
   (d) $(1, 2, 4, 0, 0, \ldots)$ and $(1, 2, 3, 4, 0, \ldots)$
   (e) $(1, 0, 1, 0, 0, 0, \ldots)$ and $(2, 4, 6, 8, \ldots)$
   (f) $(0, 0, 0, 1, 0, 0, \ldots)$ and $(8, 9, 10, 11, \ldots)$

4. In each of the following, the function is the ordinary generating function for a sequence (a_k). Find this sequence.

   (a) $\left(\dfrac{5}{1-x}\right)\left(\dfrac{3}{1-x}\right)$ (b) $\dfrac{1}{1-x}\ln(1+2x)$ (c) $\dfrac{x^3+x^5}{1-x}$

   (d) $\dfrac{x^2-3x}{1-x}+x$ (e) $(1+x)^q$, where q is a positive integer (f) $xe^{3x}+(1+x)^2$

5. If $G(x) = [1/(1-x)]^2$ is the ordinary generating function for the sequence (a_k), find a_4.

6. If $A(x) = (1-5x^2)(1+2x+3x^2+4x^3+\cdots)$ is the ordinary generating function for the sequence (a_k), find a_{11}.

7. Suppose that $A(x)$ is the ordinary generating function for the sequence (1, 3, 9, 27, 81, ...) and $B(x)$ is the ordinary generating function for the sequence (b_k). Find (b_k) if $B(x)$ equals

   (a) $A(x)+x$ (b) $A(x)+\dfrac{1}{1-x}$ (c) $2A(x)$

8. In each of the following cases, suppose that $B(x)$ is the ordinary generating function for (b_k) and $A(x)$ is the ordinary generating function for (a_k). Find an expression for $B(x)$ in terms of $A(x)$.

   (a) $b_k = \begin{cases} a_k & \text{if } k \neq 3 \\ 11 & \text{if } k = 3 \end{cases}$ (b) $b_k = \begin{cases} a_k & \text{if } k \neq 0, 4 \\ 2 & \text{if } k = 0 \\ 1 & \text{if } k = 4 \end{cases}$

9. Suppose that

$$a_k = \begin{cases} b_k & \text{if } k \neq 0, 2 \\ 4 & \text{if } k = 0 \\ 1 & \text{if } k = 2. \end{cases}$$

   Find an expression for $A(x)$, the ordinary generating function for the sequence (a_k), in terms of $B(x)$, the ordinary generating function for the sequence (b_k), if $b_0 = 2$ and $b_2 = 0$.

10. Find a simple, closed-form expression for the ordinary generating function of the following sequences (a_k).

(a) $a_k = k + 3$ (b) $a_k = 8k$ (c) $a_k = 3k + 4$

11. Make use of derivatives to find the ordinary generating function for the following sequences (b_k).

(a) $b_k = k^2$ (b) $b_k = k(k+1)$ (c) $b_k = (k+1)\frac{1}{k!}$

12. Suppose that $A(x)$ is the ordinary generating function for the sequence (a_k) and $b_k = a_{k+1}$. Find the ordinary generating function for the sequence (b_k).

13. Suppose that
$$a_k = \begin{cases} \displaystyle\sum_{i=0}^{k-2} b_i b_{k-2-i} & \text{if } k \geq 2 \\ 0 & \text{if } k = 0 \text{ or } k = 1. \end{cases}$$
Suppose that $A(x)$ is the ordinary generating function for (a_k) and $B(x)$ is the ordinary generating function for (b_k). Find an expression for $A(x)$ in terms of $B(x)$.

14. Suppose that $A(x)$ is the ordinary generating function for the sequence (a_k) and the sequence (b_k) is defined by taking
$$b_k = \begin{cases} 0 & \text{if } k < i \\ a_{k-i} & \text{if } k \geq i. \end{cases}$$
Find the ordinary generating function for the sequence (b_k) in terms of $A(x)$.

15. Find an ordinary generating function for the sequence whose kth term is
$$a_k = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{k!}.$$

16. (a) Use the reduction result of Example 5.14 to compute the rook polynomial of the board (d) of Figure 5.4.
    (b) Generalize to an $n \times n$ board with all squares on the diagonal darkened.

17. Use the result of Exercise 17, Section 5.1, to find the rook polynomials of B_I and B_J of Figure 5.6. Then use the reduction result of Example 5.14 to compute the rook polynomial of the board B of Figure 5.6.

5.3 APPLICATIONS TO COUNTING

5.3.1 Sampling Problems

Generating functions will help us in counting. To illustrate how this will work, we first consider sampling problems where the objects being sampled are of different types and objects of the same type are indistinguishable. In the language of Section 2.9, we consider sampling without replacement. For instance, suppose that there are three objects, a, b, and c, and each one can be chosen or not. How many selections are possible? Let a_k be the number of ways to select k objects. Let $G(x)$ be the generating function $\sum a_k x^k$. Now it is easy to see that $a_k = \binom{3}{k}$, and hence

$$G(x) = \binom{3}{0}x^0 + \binom{3}{1}x^1 + \binom{3}{2}x^2 + \binom{3}{3}x^3. \tag{5.13}$$

Let us calculate $G(x)$ another way. We can either pick no a's *or* one a, *and* no b's *or* one b, *and* no c's *or* one c. Let us consider the schematic product

$$[(ax)^0 + (ax)^1][(bx)^0 + (bx)^1][(cx)^0 + (cx)^1], \tag{5.14}$$

where addition and multiplication correspond to the words "or" and "and," respectively, which are italicized in the preceding sentence. (Recall the sum rule and the product rule of Chapter 2.) The expression (5.14) becomes

$$(1 + ax)(1 + bx)(1 + cx),$$

which equals

$$1 + (a + b + c)x + (ab + ac + bc)x^2 + abcx^3. \tag{5.15}$$

Notice that the coefficient of x lists the ways to get one object: It is a, or b, or c. The coefficient of x^2 lists the ways to get two objects: It is a and b, or a and c, or b and c. The coefficient of x^3 lists the ways to get three objects, and the coefficient of x^0 (namely, 1) lists the number of ways to get no objects. If we set $a = b = c = 1$, the coefficient of x^k will count the number of ways to get k objects, that is, a_k. Hence, setting $a = b = c = 1$ in (5.15) gives rise to the generating function

$$G(x) = 1 + 3x + 3x^2 + x^3,$$

which is what we calculated in (5.13).

The same technique works in problems where it is not immediately clear what the coefficients in $G(x)$ are. Then we can calculate $G(x)$ by means of this technique and calculate the appropriate coefficients from $G(x)$.

Example 5.15 Suppose that we have three types of objects, a's, b's, and c's. Suppose that we can pick either 0, 1, or 2 a's, then 0 or 1 b, and finally 0 or 1 c. How many ways are there to pick k objects? The answer is not $\binom{4}{k}$. For example, 2 a's and 1 b is not considered the same as 1 a, 1 b, and 1 c. However, picking the first a and also b is considered the same as picking the second a and also b: The a's are indistinguishable. We want the number of distinguishable ways to pick k objects. Suppose that b_k is the desired number of ways. We shall try to calculate the ordinary generating function $G(x) = \sum b_k x^k$. The correct expression to consider here is

$$[(ax)^0 + (ax)^1 + (ax)^2][(bx)^0 + (bx)^1][(cx)^0 + (cx)^1], \tag{5.16}$$

since we can pick either 0, or 1, or 2 a's, and 0 or 1 b, and 0 or 1 c. The expression (5.16) reduces to

$$(1 + ax + a^2x^2)(1 + bx)(1 + cx),$$

which equals

$$1 + (a + b + c)x + (ab + bc + ac + a^2)x^2 + (abc + a^2b + a^2c)x^3 + a^2bcx^4. \tag{5.17}$$

As in Example 5.14, the coefficient of x^3 gives the ways of obtaining three objects: a, b, and c; or 2 a's and b; or 2 a's and c. The same thing holds for the other coefficients. Again, taking $a = b = c = 1$ in (5.17) gives the generating function

$$G(x) = 1 + 3x + 4x^2 + 3x^3 + x^4.$$

The coefficient of x^k is b_k. For example, $b_2 = 4$. (The reader should check why this is so.) ■

In general, suppose that we have p types of objects, with n_1 indistinguishable objects of type 1, n_2 of type 2, ..., n_p of type p. Let c_k be the number of distinguishable ways of picking k objects if we can pick any number of objects of each type. The ordinary generating function is given by $G(x) = \sum c_k x^k$. To calculate this, we consider the product

$$\left[(a_1x)^0 + (a_1x)^1 + \cdots + (a_1x)^{n_1}\right] \times \left[(a_2x)^0 + (a_2x)^1 + \cdots + (a_2x)^{n_2}\right] \times \cdots \\ \times \left[(a_px)^0 + (a_px)^1 + \cdots + (a_px)^{n_p}\right].$$

Setting $a_1 = a_2 = \cdots = a_p = 1$, we obtain

$$G(x) = (1 + x + x^2 + \cdots + x^{n_1})(1 + x + x^2 + \cdots + x^{n_2}) \cdots (1 + x + x^2 + \cdots + x^{n_p}).$$

The number c_k is given by the coefficient of x^k in $G(x)$. Thus, we have the following theorem.

Theorem 5.2 Suppose that we have p types of objects, with n_i indistinguishable objects of type $i, i = 1, 2, \ldots, p$. The number of distinguishable ways of picking k objects if we can pick any number of objects of each type is given by the coefficient of x^k in the ordinary generating function

$$G(x) = (1 + x + x^2 + \cdots + x^{n_1})(1 + x + x^2 + \cdots + x^{n_2}) \cdots (1 + x + x^2 + \cdots + x^{n_p}).$$

Example 5.16 Indistinguishable Men and Women Suppose that we have m (indistinguishable) men and n (indistinguishable) women. If we can choose any number of men and any number of women, Theorem 5.2 implies that the number of ways we can choose k people is given by the coefficient of x^k in

$$G(x) = (1 + x + \cdots + x^m)(1 + x + \cdots + x^n). \tag{5.18}$$

Note: This coefficient is not

$$\binom{m+n}{k}$$

because, for example, having 3 men and $k - 3$ women is the same no matter which men and women you pick. Now $1 + x + \cdots + x^m$ is the generating function for the sequence

$$(a_k) = (1, 1, \ldots, 1, 0, 0, \ldots)$$

and $1 + x + \cdots + x^n$ is the generating function for the sequence

$$(b_k) = (1, 1, \ldots, 1, 0, 0, \ldots),$$

where a_k is 1 for $k = 0, 1, \ldots, m$ and b_k is 1 for $k = 0, 1, \ldots, n$. It follows from (5.18) that $G(x)$ is the generating function for the convolution of the two sequences (a_k) and (b_k). We leave it to the reader (Exercise 11) to compute this convolution in general. For example, if $m = 8$ and $n = 7$, the number of ways we can choose $k = 9$ people is given by

$$a_0b_9 + a_1b_8 + \cdots + a_9b_0 = 0 + 0 + 1 + \cdots + 1 + 1 + 0 = 7.$$

As a check on the answer, we note that the seven ways are the following: 2 men and 7 women, 3 men and 6 women, ..., 8 men and 1 woman. ■

Example 5.17 A Sampling Survey In doing a sampling survey, suppose that we have divided the possible men to be interviewed into various categories, such as teachers, doctors, lawyers, and so on, and similarly for the women. Suppose that in our group we have two men from each category and one woman from each category, and suppose that there are q categories. How many distinguishable ways are there of picking a sample of k people? We now want to distinguish people of the same gender if and only if they belong to different categories. The generating function for the number of ways to choose k people is given by

$$\begin{aligned} G(x) &= \underbrace{(1 + x + x^2)(1 + x + x^2) \cdots (1 + x + x^2)}_{q \text{ terms}} \underbrace{(1 + x)(1 + x) \cdots (1 + x)}_{q \text{ terms}} \\ &= (1 + x + x^2)^q (1 + x)^q. \end{aligned}$$

The individual terms of $G(x)$ come from the fact that in each of the q categories we choose either 0, 1, or 2 men to be sampled, thus giving a $(1 + x + x^2)$ term for each category; and in addition, we choose either 0 or 1 woman, so we have a $(1 + x)$ term for each of the q categories. The number of ways to select k people is the coefficient of x^k in $G(x)$. For instance, if $q = 2$, then

$$G(x) = x^6 + 4x^5 + 8x^4 + 10x^3 + 8x^2 + 4x + 1.$$

For example, there are 10 ways to pick 3 people. The reader might wish to identify those 10 ways. In general, if there are m categories of men and n categories of women, and there are p_i men in category $i, i = 1, 2, \ldots, m$, and q_j women in category $j, j = 1, 2, \ldots, n$, the reader might wish to express the ordinary generating function for the number of ways to select k people. ■

Example 5.18 Another Survey Suppose that our survey team is considering three households. The first household has two people living in it (let us call them "a's"). The second household has one person living in it (let us call that person a "b"). The third household has one person living in it (let us call that person a "c").

In how many ways can k people be selected if we either choose none of the members of a given household or all of the members of that household? Let us again think of the product and sum rule: We can either pick 0 or 2 a's, and either 0 or 1 b, and either 0 or 1 c. Hence, we consider the expression

$$[(ax)^0 + (ax)^2][(bx)^0 + (bx)^1][(cx)^0 + (cx)^1].$$

This becomes

$$(1 + a^2x^2)(1 + bx)(1 + cx) = 1 + (b + c)x + (bc + a^2)x^2 + (a^2b + a^2c)x^3 + a^2bcx^4.$$

The ways of choosing two people are given by the coefficient of x^2: We can choose b and c or we can choose two a's. Setting $a = b = c = 1$, we obtain the generating function

$$G(x) = 1 + 2x + 2x^2 + 2x^3 + x^4.$$

The number of ways of choosing k people is given by the coefficient of x^k. ■

Example 5.19 Suppose that we have p different kinds of objects, each in (for all practical purposes) infinite supply. How many ways are there of picking a sample of k objects? The answer is given by the coefficient of x^k in the ordinary generating function

$$\begin{aligned} G(x) &= \underbrace{(1 + x + x^2 + \cdots)(1 + x + x^2 + \cdots)\cdots(1 + x + x^2 + \cdots)}_{p \text{ terms}} \qquad (5.19) \\ &= (1 + x + x^2 + \cdots)^p \\ &= (1 - x)^{-p} \qquad [\text{by } (5.2)]. \end{aligned}$$

(An alternative approach, leading ultimately to the same result, is to apply Theorem 5.2 with each $n_i = k$.) We shall want to develop ways of finding the coefficient of x^k given an expression for $G(x)$ such as (5.19). To do so, we introduce the Binomial Theorem in the next section. ■

Example 5.20 Integer Solutions of Equations How many integer solutions are there to the equation

$$b_1 + b_2 + b_3 = 14,$$

if $b_i \geq 0$ for $i = 1, 2, 3$? Since each b_i can take on the value 0 or 1 or 2 or $\ldots$, the answer is given by the coefficient of x^{14} in the ordinary generating function

$$\begin{aligned} G(x) &= \left(1 + x + x^2 + \cdots\right)\left(1 + x + x^2 + \cdots\right)\left(1 + x + x^2 + \cdots\right) \\ &= \left(1 + x + x^2 + \cdots\right)^3 \\ &= (1 - x)^{-3}, \end{aligned}$$

by Example 5.19. ■

Example 5.21 Coding Theory (Example 2.27 Revisited) In Example 2.27 we considered the problem of checking codewords after transmission. Here we consider the common case of codewords as bit strings. Suppose that we check for three kinds of errors: addition of a digit (0 or 1), deletion of a digit (0 or 1), and reversal of a digit (0 to 1 or 1 to 0). In how many ways can we find 30 errors? To answer this question, we can assume that two errors of the same type are indistinguishable. Moreover, for all practical purposes, each type of error is in infinite supply. Thus, we seek the coefficient of x^{30} in the ordinary generating function

$$G(x) = (1 + x + x^2 + \cdots)(1 + x + x^2 + \cdots)(1 + x + x^2 + \cdots) = (1 - x)^{-3}.$$

This is a special case of (5.19).

Suppose next that we do not distinguish among the three types of errors, but we do keep a record of whether an error occurred in the first codeword sent, the second codeword sent, and so on. In how many different ways can we find 30 errors if there are 100 codewords? This is the question we addressed in Example 2.27. To answer this question, we note that there are 100 types of errors, one type for each codeword. We can either assume that the number of possible errors per codeword is, for all practical purposes infinite, or that it is bounded by 30, or that it is bounded by the number of digits per codeword. In the former case, we consider the ordinary generating function

$$G(x) = (1 + x + x^2 + \cdots)^{100} = (1 - x)^{-100}$$

and look for the coefficient of x^{30}. In the latter two cases, we simply end the terms in $G(x)$ at an appropriate power. The coefficient of x^{30} will, of course, be the same regardless, provided that at least 30 digits appear per codeword. ■

Example 5.22 Sicherman Dice Suppose that a standard pair of dice are rolled. What are the probabilities for the various outcomes for this roll? Since each die can take on the value 1, 2, 3, 4, 5, or 6, we consider the ordinary generating function

$$\begin{aligned} p(x) &= \left(x + x^2 + x^3 + x^4 + x^5 + x^6\right)\left(x + x^2 + x^3 + x^4 + x^5 + x^6\right) \\ &= \left(x^2 + 2x^3 + 3x^4 + 4x^5 + 5x^6 + 6x^7 + 5x^8 + 4x^9 + 3x^{10} + 2x^{11} + x^{12}\right). \end{aligned}$$

Note that the coefficient of x^j is the number of ways to obtain a roll of j. Also, the sum of the coefficients, 36, gives the total number of different rolls possible with two dice. So, for example, the probability of rolling a 5 is $4/36$.

What about other dice? Does there exist a different pair of six-sided dice which yield the same outcome probabilities as standard dice? We will only consider dice with positive, integer labels. Let $a_1, a_2, a_3, a_4, a_5, a_6$ be the values on one die and $b_1, b_2, b_3, b_4, b_5, b_6$ be the values on the other die. If another pair exists, it must be the case that

$$\begin{aligned} &\left(x + x^2 + x^3 + x^4 + x^5 + x^6\right)\left(x + x^2 + x^3 + x^4 + x^5 + x^6\right) = \\ &\quad \left(x^{a_1} + x^{a_2} + x^{a_3} + x^{a_4} + x^{a_5} + x^{a_6}\right)\left(x^{b_1} + x^{b_2} + x^{b_3} + x^{b_4} + x^{b_5} + x^{b_6}\right). \quad (5.20) \end{aligned}$$

The left-hand side of (5.20), $p(x)$, factors into

$$x^2(1+x)^2(1+x+x^2)^2(1-x+x^2)^2.$$

Since polynomials with integer coefficients always factor uniquely (a standard fact from any abstract algebra course), the right-hand side of (5.20) must also factor this way. Therefore,

$$f(x) = (x^{a_1}+x^{a_2}+x^{a_3}+x^{a_4}+x^{a_5}+x^{a_6}) = x^{c_1}(1+x)^{c_2}(1+x+x^2)^{c_3}(1-x+x^2)^{c_4}$$

and

$$g(x) = \left(x^{b_1}+x^{b_2}+x^{b_3}+x^{b_4}+x^{b_5}+x^{b_6}\right) = x^{d_1}(1+x)^{d_2}(1+x+x^2)^{d_3}(1-x+x^2)^{d_4},$$

where $c_i + d_i = 2$, for $i = 1, 2, 3, 4$.

To find c_i and d_i (and hence a_i and b_i), first consider

$$f(0) = (0^{a_1}+0^{a_2}+0^{a_3}+0^{a_4}+0^{a_5}+0^{a_6}).$$

This equals 0, as does $g(0)$, so the x term must be present in the factorizations of both $f(x)$ and $g(x)$. Therefore, c_1 and d_1 both must equal 1. Next, consider $f(1) = (1^{a_1}+1^{a_2}+1^{a_3}+1^{a_4}+1^{a_5}+1^{a_6}) = 6$ or $1^{c_1}2^{c_2}3^{c_3}1^{c_4}$. So, $c_2 = c_3 = 1$. A similar analysis and result, $d_2 = d_3 = 1$, is also true for $g(x)$. Finally, we consider c_4 and d_4. Either $c_4 = d_4 = 1$, which yields a standard pair of dice, or $c_4 = 0, d_4 = 2$ (or vice versa). In the latter case, after multiplying polynomials, we see that

$$f(x) = x+x^2+x^2+x^3+x^3+x^4, \quad g(x) = x+x^3+x^4+x^5+x^6+x^8$$

and

$$f(x)g(x) = \left(x^2+2x^3+3x^4+4x^5+5x^6+6x^7+5x^8+4x^9+3x^{10}+2x^{11}+x^{12}\right).$$

This pair of dice, one labeled 1, 2, 2, 3, 3, 4 and the other labeled 1, 3, 4, 5, 6, 8, are called *Sicherman dice*.[4] Sicherman dice are the *only* alternative pair of dice (with positive integer labels) that yield the same outcome probabilities as standard dice! ∎

Example 5.23 Partitions of Integers Recall from Section 2.10.5 that a *partition* of a positive integer k is a collection of positive integers that sum to k. For instance, the integer 4 has the partitions $\{1,1,1,1\}$, $\{1,1,2\}$, $\{2,2\}$, $\{1,3\}$, and $\{4\}$. Suppose that $p(k)$ is the number of partitions of the integer k. Thus, $p(4) = 5$. Exercises 12–16 investigate partitions of integers, using the techniques of this section. The idea of doing so goes back to Euler in 1748. For a detailed discussion of partitions, see most number theory books: for instance, Bressoud and Wagon [2000] or Hardy and Wright [1980]. See also Berge [1971], Cohen [1978], or Tomescu [1985]. ∎

[4] George Sicherman first considered and solved this alternative dice problem whose outcome probabilities are the same as standard dice. See Gardner [1978] for a discussion of this problem.

Example 5.24 A Crucial Observation Underlying the Binary Arithmetic of Computing Machines: Partitions into Distinct Integers Let $p^*(k)$ be the number of ways to partition the integer k into distinct integers. Thus, $p^*(7) = 5$, using the partitions $\{7\}, \{1,6\}, \{2,5\}, \{3,4\}$, and $\{1,2,4\}$. A crucial observation underlying the binary arithmetic that pervades computing machines is that every integer k can be partitioned into distinct integers that are powers of 2. For example, $7 = 2^0 + 2^1 + 2^2$ and $19 = 2^0 + 2^1 + 2^4$. We ask the reader to prove this in Exercise 16(c), using methods of this section. ■

5.3.2 A Comment on Occupancy Problems

In Section 2.10 we considered occupancy problems, problems of distributing balls to cells. The second part of Example 5.21 involves an occupancy problem: We have 30 balls (errors) and 100 cells (codewords). This is a special case of the occupancy problem where we have k indistinguishable balls and we wish to distribute them among p distinguishable cells. If we put no restriction on the number of balls in each cell, it is easy to generalize the reasoning in Example 5.21 and show that the number of ways to distribute the k balls into the p cells is given by the coefficient of x^k in the ordinary generating function (5.19). By way of contrast, if we allow no more than n_i balls in the ith cell, we see easily that the number of ways is given by the coefficient of x^k in the ordinary generating function of Theorem 5.2.

EXERCISES FOR SECTION 5.3

1. In each of the following, set up the appropriate generating function. Do not calculate an answer but indicate what you are looking for: for example, the coefficient of x^{10}.

   (a) A survey team wants to select at most 3 male students from Michigan, at most 3 female students from Brown, at most 2 male students from Stanford, and at most 2 female students from Rice. In how many ways can 5 students be chosen to interview if only Michigan and Stanford males and Brown and Rice females can be chosen, and 2 students of the same gender from the same school are indistinguishable?

   (b) In how many ways can 5 letters be picked from the letters a, b, c, d if b, c, and d can be picked at most once and a, if picked, must be picked 4 times?

   (c) In making up an exam, an instructor wants to use at least 3 easy problems, at least 3 problems of medium difficulty, and at least 2 hard problems. She has limited the choice to 7 easy problems, 6 problems of medium difficulty, and 4 hard problems. In how many ways can she pick 11 problems? (The order of the exam problems will be decided upon later.) (*Note:* Do not distinguish 2 easy problems from each other, or 2 problems of medium difficulty, or 2 hard problems.)

   (d) In how many ways can 8 binary digits be picked if each must be picked an even number of times?

(e) How many ways are there to choose 12 voters from a group of 6 Republicans, 6 Democrats, and 7 Independents, if we want at least 4 Independents and any two voters of the same political persuasion are indistinguishable?

(f) A Geiger counter records the impact of five different kinds of radioactive particles over a period of 5 minutes. How many ways are there to obtain a count of 20?

(g) In checking the work of a communication device, we look for 4 types of transmission errors. In how many ways can we find 40 errors?

(h) In part (g), suppose that we do not distinguish the types of errors, but we do keep a record of the day on which an error occurred. In how many different ways can we find 40 errors in 100 days?

(i) How many ways are there to distribute 12 indistinguishable balls into 8 distinguishable cells?

(j) Repeat part (i) if no cell can be empty.

(k) If 14 standard dice are rolled, how many ways are there for the total to equal 30?

(l) A survey team divides the possible people to interview into 6 groups depending on age, and independently into 5 groups depending on geographic location. In how many ways can 10 people be chosen to interview, if 2 people are distinguished only if they belong to different age groups, live in different geographic locations, or are of opposite gender?

(m) Find the number of ways to make change for a dollar using coins (pennies, nickels, dimes, and/or quarters).

(n) Find the number of solutions to the equation

$$x_1 + x_2 + x_3 = 14$$

in which each x_i is a nonnegative integer and $x_i \leq 7$.

(o) Find the number of solutions to the equation

$$x_1 + x_2 + x_3 = 20$$

in which $x_i \geq 0$ for all i, and x_1 odd, $2 \leq x_2 \leq 5$, and x_3 prime.

2. Suppose that we wish to build up an RNA chain which has length 5 and uses U, C, and A each at most once and G arbitrarily often. How many ways are there to choose the bases (not order them)? Answer this question by setting up a generating function and computing an appropriate coefficient.

3. A customer wants to buy six pieces of fruit, including at most two apples, at most two oranges, at most two pears, and at least one but at most two peaches. How many ways are there to buy six pieces of fruit if any two pieces of fruit of the same type, for example, any two peaches, are indistinguishable?

4. Suppose that there are p kinds of objects, with n_i indistinguishable objects of the ith kind. Suppose that we can pick all or none of each kind. Set up a generating function for computing the number of ways to choose k objects.

5. Consider the voting situation of Exercise 7, Section 2.15.

(a) If all representatives of a province vote alike, set up a generating function for calculating the number of ways to get k votes. (Getting a vote from a representative of province A is considered different from getting a vote from a representative of province B, and so on.)

(b) Repeat part (a) if representatives of a province do not necessarily vote alike.

6. Consider the following basic groups of foods from the Food Guide Pyramid: breads, fruits, vegetables, meat, milk, and fats. A dietician wants to choose a daily menu in a cafeteria by choosing 10 items (the limit of the serving space), with at least one item from each category.

   (a) How many ways are there of choosing the basic menu if items in the same group are treated as indistinguishable and it is assumed that there are, for all practical purposes, an arbitrarily large number of different foods in each group? Answer this by setting up a generating function. Do not do the computation.

   (b) How would the problem be treated if items in a group were treated as distinguishable and there are, say, 30 items in each group?

7. Suppose that there are p kinds of objects, each in infinite supply. Let a_k be the number of distinguishable ways of choosing k objects if only an even number (including 0) of each kind of object can be taken. Set up a generating function for a_k.

8. In a presidential primary involving all of the New England states on the same day, a presidential candidate would receive a number of electoral votes from each state proportional to the number of voters who voted for that candidate in each state. For example, a candidate who won 50 percent of the vote in Maine and no votes in the other states would receive two electoral votes (or two state votes) in the convention, since Maine has four electoral votes. Votes would be translated into integers by rounding—there are no fractional votes. Set up a generating function for computing the number of ways that a candidate could receive 25 electoral votes. (The electoral votes of the states in the region as of the 2000 census are as follows: Connecticut, 7; Maine, 4; Massachusetts, 12; New Hampshire, 4; Rhode Island, 4; Vermont, 3.)

9. In Example 5.17 we showed that if $q = 2$, there are 10 ways to pick 3 people. What are they?

10. (Gallian [2002]) Suppose that you have an 18-sided die which is labeled 1, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 8.

    (a) Find the labels of a 2-sided die so that this die, when rolled with the 18-sided die, has the same outcome probabilities as a standard pair of 6-sided dice.

    (b) Extending part (a), find the labels of a 4-sided die so that this die, when rolled with the 18-sided die, has the same outcome probabilities as a standard pair of 6-sided dice.

11. Find a general formula for the convolution of the two sequences (a_k) and (b_k) found in Example 5.16.

12. The next five exercises investigate partitions of integers, as defined in Examples 5.23 and 5.24. Let $p(k)$ be the number of partitions of integer k and let

$$G(x) = \frac{1}{(1-x)(1-x^2)(1-x^3)(1-x^4)\cdots}.$$

Show that $G(x)$ is the ordinary generating function for the sequence $(p(k))$.

13. Recall that $p^*(k)$ is the number of ways to partition the integer k into distinct integers.
    (a) Find $p^*(8)$. (b) Find $p^*(11)$.
    (c) Find an ordinary generating function for $(p^*(k))$.
14. Let $p_o(k)$ be the number of ways to partition integer k into not necessarily distinct odd integers.
    (a) Find $p_o(7)$. (b) Find $p_o(8)$. (c) Find $p_o(11)$.
    (d) Find a generating function for $(p_o(k))$.
15. Let $p^*(k)$ and $p_o(k)$ be defined as in Exercises 13 and 14, respectively. Show that $p^*(k) = p_o(k)$.
16. (a) Show that for $|x| < 1$,
    $$(1-x)(1+x)(1+x^2)(1+x^4)(1+x^8)\cdots(1+x^{2^k})\cdots = 1.$$
    (b) Deduce that for $|x| < 1$,
    $$1 + x + x^2 + x^3 + \cdots = (1+x)(1+x^2)(1+x^4)(1+x^8)\cdots(1+x^{2^k})\cdots.$$
    (c) Conclude that any integer can be written uniquely in binary form, that is, as a sum $a_0 2^0 + a_1 2^1 + a_2 2^2 + \cdots$, where each a_i is 0 or 1. (This conclusion is a crucial one underlying the binary arithmetic that pervades computing machines.)
17. Find the number of integer solutions to the following equations/inequalities:
    (a) $b_1 + b_2 + b_3 + b_4 + b_5 + b_6 = 15$, with $0 \leq b_i \leq 3$ for all i.
    (b) $b_1 + b_2 + b_3 = 15$, with $0 \leq b_i$ for all i and b_1 odd, b_2 even, and b_3 prime.
    (c) $b_1 + b_2 + b_3 + b_4 = 20$, with $2 \leq b_1 \leq 4$ and $4 \leq b_i \leq 7$ for $2 \leq i \leq 4$.
    (d) $b_1 + b_2 + b_3 + b_4 \leq 10$, with $0 \leq b_i$ for all i. (*Hint*: Include a "slack" variable b_5 to create an equality.)
18. Solve the question of Example 5.20 using the method of occupancy problems from Section 2.10.
19. Using the techniques from Example 5.22, find the number of pairs of 4-sided dice which have the same outcome probabilities as a "standard" pair of 4-sided dice. ("Standard" 4-sided dice are labeled 1, 2, 3, 4.)

5.4 THE BINOMIAL THEOREM

In order to expand out the generating function of Equation (5.19), it will be useful to find the Maclaurin series for the function $f(x) = (1+x)^u$, where u is an arbitrary real number, positive or negative, and not necessarily an integer. We have

$$\begin{aligned} f'(x) &= u(1+x)^{u-1} \\ f''(x) &= u(u-1)(1+x)^{u-2} \\ &\vdots \\ f^{(r)}(x) &= u(u-1)\cdots(u-r+1)(1+x)^{u-r}. \end{aligned}$$

Thus, by Equation (5.1), we have the following theorem.

Theorem 5.3 (Binomial Theorem)

$$(1+x)^u = 1 + ux + \frac{u(u-1)}{2!}x^2 + \cdots + \frac{u(u-1)\cdots(u-r+1)}{r!}x^r + \cdots. \tag{5.21}$$

One can prove that the expansion (5.21) holds for $|x| < 1$. This expansion can be written succinctly by introducing the *generalized binomial coefficient*

$$\binom{u}{r} = \begin{cases} \dfrac{u(u-1)\cdots(u-r+1)}{r!} & \text{if } r > 0 \\ 1 & \text{if } r = 0, \end{cases}$$

which is defined for any real number u and nonnegative integer r. Then (5.21) can be rewritten as

$$(1+x)^u = \sum_{r=0}^{\infty} \binom{u}{r} x^r. \tag{5.22}$$

If u is a positive integer n, then $\binom{u}{r}$ is 0 for $r > u$, and 5.22 reduces to the binomial expansion (Theorem 2.7).

Example 5.25 Computing Square Roots Before returning to generating functions, let us give one quick application of the binomial theorem to the computation of square roots. We have

$$(1+x)^{1/2} = 1 + \frac{1}{2}x + \frac{\frac{1}{2}(\frac{1}{2}-1)}{2!}x^2 + \frac{\frac{1}{2}(\frac{1}{2}-1)(\frac{1}{2}-2)}{3!}x^3 + \cdots, \tag{5.23}$$

if $|x| < 1$. Let us use this result to compute $\sqrt{30}$. Note that $|29| \geq 1$, so we cannot use (5.23) directly. However,

$$\sqrt{30} = \sqrt{25+5} = 5\sqrt{1+.2},$$

so we can apply (5.23) with $x = .2$. This gives us

$$\sqrt{30} = 5\left[1 + \frac{1}{2}(.2) - \frac{1}{8}(.2)^2 + \frac{1}{16}(.2)^3 - \cdots\right] \approx 5.4775.$$

■

Returning to Example 5.19, let us apply the Binomial Theorem to find the coefficient of x^k in the expansion of

$$G(x) = (1 + x + x^2 + \cdots)^p.$$

Note that provided that $|x| < 1$,

$$G(x) = \left(\frac{1}{1-x}\right)^p,$$

using the identity (5.2). $G(x)$ can be rewritten as

$$G(x) = (1-x)^{-p}.$$

We can now apply the binomial theorem with $-x$ in place of x and with $u = -p$. Then we have

$$G(x) = \sum_{r=0}^{\infty} \binom{-p}{r} (-x)^r.$$

For $k > 0$, the coefficient of x^k is

$$\binom{-p}{k}(-1)^k = \frac{(-p)(-p-1)\cdots(-p-k+1)}{k!}(-1)^k,$$

which equals

$$\begin{aligned} \frac{p(p+1)\cdots(p+k-1)}{k!} &= \frac{(p+k-1)(p+k-2)\cdots p}{k!} \\ &= \frac{(p+k-1)!}{k!(p-1)!} \\ &= \binom{p+k-1}{k}. \end{aligned}$$

Since $\binom{p+0-1}{0} = 1$, $\binom{p+k-1}{k}$ gives the coefficient of x^0 also. Hence, we have proved the following theorem.

Theorem 5.4 If there are p types of objects, the number of distinguishable ways to choose k objects if we are allowed unlimited repetition of each type is given by

$$\binom{-p}{k}(-1)^k = \binom{p+k-1}{k}.$$

Corollary 5.4.1 Suppose that p is a fixed positive integer. Then the ordinary generating function for the sequence (c_k), where

$$c_k = \binom{p+k-1}{k},$$

is given by

$$C(x) = \left(\frac{1}{1-x}\right)^p = (1-x)^{-p}.$$

Proof. This is a corollary of the proof. Q.E.D.

Note that the case $p = 2$ is covered in Example 5.12. In this case,

$$c_k = \binom{2+k-1}{k} = \binom{k+1}{k} = k+1.$$

Note that in terms of occupancy problems (Section 5.3.2), Theorem 5.4 says that the number of ways to place k indistinguishable balls into p distinguishable cells, with no restriction on the number of balls in a cell, is given by

$$\binom{p+k-1}{k}.$$

We have already seen this result in Theorem 2.4.

Example 5.26 The Pizza Problem (Example 2.20 Revisited) In a restaurant that serves pizza, suppose that there are nine types of toppings (see Example 2.20). If a pizza can have at most one kind of topping, how many ways are there to sell 100 pizzas? It is reasonable to assume that each topping is, for all practical purposes, in infinite supply. Now we have $p = 10$ types of toppings, including the topping "nothing but cheese." Then by Theorem 5.4, the number of distinguishable ways to pick $k = 100$ pizzas is given by $\binom{109}{100}$. ■

Example 5.27 Rating Computer Systems Alternative computer systems are rated on different benchmarks, with an integer score of 1 to 6 possible on each benchmark. In how many ways can the total of the scores on three benchmarks add up to 12? To answer this question, let us think of the score on each benchmark as being chosen as 1, 2, 3, 4, 5, or 6 points. Hence, the generating function to consider is

$$G(x) = (x + x^2 + \cdots + x^6)^3.$$

Note that we start with x rather than 1 (or x^0) because there must be at least one point chosen. We take the third power because there are three benchmarks and we want the coefficient of x^{12}. How can this be found? The answer uses the identity

$$1 + x + x^2 + \cdots + x^s = \frac{1 - x^{s+1}}{1 - x}. \tag{5.24}$$

Then we note that

$$\begin{aligned} G(x) &= [x(1 + x + x^2 + \cdots + x^5)]^3 \\ &= x^3 \left[\frac{1 - x^6}{1 - x}\right]^3 \\ &= x^3(1 - x^6)^3(1 - x)^{-3}. \end{aligned} \tag{5.25}$$

We already know that $C(x) = (1 - x)^{-p}$ is the generating function for the sequence (c_k) where

$$c_k = \binom{p + k - 1}{k}.$$

Here

$$c_k = \binom{3 + k - 1}{k}.$$

The expression $B(x) = x^3(1 - x^6)^3$ may be expanded out using the binomial *expansion*, giving us

$$\begin{aligned} B(x) &= x^3[1 - 3x^6 + 3x^{12} - x^{18}] \\ &= x^3 - 3x^9 + 3x^{15} - x^{21}. \end{aligned}$$

Hence, $B(x)$ is the generating function for the sequence

$$(b_k) = (0, 0, 0, 1, 0, 0, 0, 0, 0, -3, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, -1, 0, 0, \ldots).$$

It follows from (5.25) that $G(x)$ is the generating function for the convolution (a_k) of the sequences (b_k) and (c_k). We wish to find the coefficient a_{12} of x^{12}. This is obtained as

$$\begin{aligned} a_{12} &= b_0c_{12} + b_1c_{11} + b_2c_{10} + \cdots + b_{12}c_0 \\ &= b_3c_9 + b_9c_3 \\ &= 1 \cdot \binom{3+9-1}{9} - 3 \cdot \binom{3+3-1}{3} \\ &= \binom{11}{9} - 3\binom{5}{3} \\ &= 25. \end{aligned}$$

■

Example 5.28 List *T*-Colorings Chromatic polynomials from Section 3.4 are used to find the number of distinct colorings that are possible for a given graph using a fixed number of colors. Here we use generating functions to find the number of distinct list T-colorings (see Examples 3.20 and 3.22) of an unlabeled graph G. A *list T-coloring* of G is a T-coloring in which the color assigned to a vertex x belongs to the list $L(x)$ associated with x. We will assume that $T = \{0, 1, 2, \ldots, r\}$, $G = K_n$, and each vertex submits the list $\{1, 2, \ldots, l\}$ as possible colors.

As an example, suppose that $r = 1$, $n = 5$, and $l = 16$. Consider a list T-coloring of K_5 using the colors 1, 3, 5, 8, 10. Each pair of colors differ by at least two in absolute value and $1 \leq 1 < 3 < 5 < 8 < 10 \leq 16$. This set of inequalities gives rise to a set of differences, namely, $1-1$, $3-1$, $5-3$, $8-5$, $10-8$, $16-10$ or 0, 2, 2, 3, 2, 6, and these differences sum to 15. Note that except for the first and last differences, none of the differences could equal 0 or 1 since $T = \{0, 1\}$ and the only difference of 0 or 1 could occur as the first difference (which it did with a 0 in this case) or the last difference.

Next, consider a different sequence of nonnegative integers that sum to 15, say, 3, 3, 2, 2, 3, 2. These could be considered the set of differences from the set of inequalities $1 \leq 4 < 7 < 9 < 11 < 14 \leq 16$ and thus a list T-coloring of K_5 using the colors 4, 7, 9, 11, 14. Note that all absolute differences among consecutive integers in this increasing list of five integers are greater than 1.

Thus, there is easily seen to be a one-to-one correspondence between the list T-colorings to be counted and integer solutions to

$$b_1 + b_2 + b_3 + b_4 + b_5 + b_6 = 15,$$

where $b_1, b_6 \geq 0$ and $b_2, b_3, b_4, b_5 \geq 2$. Therefore, using the ideas of Example 5.20, we see that the number of list T-colorings in this example is the coefficient of x^{15}

in

$$\begin{aligned} f(x) &= \left(1+x+x^2+\cdots\right)^2\left(x^2+x^3+\cdots\right)^4 \\ &= x^8\left(1-x\right)^{-6} \end{aligned}$$

or the coefficient of x^7 in $(1-x)^{-6}$. This coefficient of x^7 equals

$$\binom{-6}{7}(-1)^7 = \binom{6+7-1}{7} = \binom{12}{7} = 792.$$

■

Example 5.29 Number of Weak Orders We can use generating functions to count the number of distinct weak orders (X, R) on an n-element set X. Recall that weak orders have levels and the elements on each level are equivalent. That is, if $x, y \in X$, then xRy for each x at a higher level than y and xRy and yRx if x, y are at the same level. Consider a weak order with w levels. There must be at least one element on each level. Therefore, the number of weak orders on n elements with w levels is the coefficient of x^n in the generating function

$$\begin{aligned} G(x) &= \underbrace{(x+x^2+\cdots)(x+x^2+\cdots)\cdots(x+x^2+\cdots)}_{w \text{ terms}} \\ &= (x+x^2+\cdots)^w \\ &= x^w(1+x+x^2+\cdots)^w \\ &= x^w(1-x)^{-w}. \end{aligned}$$

This is then the coefficient of x^{n-w} in $(1-x)^{-w}$, which is

$$\binom{-w}{n-w}(-1)^{n-w} = \binom{w+(n-w)-1}{n-w} = \binom{n-1}{n-w}.$$

Since the number of levels w can range from 1 to n, the total number of distinct weak orders on n elements equals

$$\binom{n-1}{n-1} + \binom{n-1}{n-2} + \cdots + \binom{n-1}{n-n} = \sum_{w=1}^{n} \binom{n-1}{n-w}.$$

By Theorem 2.8 [and from Pascal's triangle (Section 2.7)] this sum is given by 2^{n-1}.

■

EXERCISES FOR SECTION 5.4

1. Use the binomial theorem to find the coefficient of x^3 in the expansion of:
   (a) $\sqrt[4]{1+x}$ (b) $(1+x)^{-3}$
   (c) $(1-x)^{-4}$ (d) $(1+5x)^{3/4}$
2. Find the coefficient of x^7 in the expansion of:
   (a) $(1-x)^{-4}x^3$ (b) $(1-x)^{-2}x^8$
   (c) $x^2(1+x^2)^3(1-x)^{-3}$ (d) $(1+x)^{1/2}x^5$

3. If $(1+x)^{1/3}$ is the ordinary generating function for the sequence (a_k), find a_2.
4. Do the calculation to solve Exercise 6(a) of Section 5.3.
5. Use Theorem 5.4 to compute the number of ways to pick six letters if a and b are the only letters available. Check your answer by writing out all the ways.
6. How many ways are there to choose 11 personal computers if 6 different manufacturers' models are available?
7. How many ways are there to choose 50 shares of stock if 100 shares each of four different companies are available?
8. A fruit fly is classified as either dominant, hybrid, or recessive for eye color. Ten fruit flies are to be chosen for an experiment. In how many different ways can the genotypes (classifications) dominant, hybrid, and recessive be chosen if you are interested only in the number of dominants, number of hybrids, and number of recessives?
9. Five different banks offer certificates of deposit (CDs) that can only be purchased in multiples of \$1,000. If an investor has \$10,000, in how many different ways can she invest in CDs?
10. Suppose that there are six different kinds of fruit available, each in (theoretically) infinite supply. How many different fruit baskets of 10 pieces of fruit are there?
11. A person drinks one can of beer an evening, choosing one of six different brands. How many different ways are there in which to drink beer over a period of a week if as many cans of a given brand are available as necessary, any two such cans are interchangeable, and we do not distinguish between drinking brand x on Monday and drinking brand x on Tuesday?
12. In studying defective products in a factory, we classify defects found by the day of the week in which they are found. In how many different ways can we classify 10 defective products?
13. Suppose that there are p different kinds of objects, each in infinite supply. Let a_k be the number of distinguishable ways to pick k of the objects if we must pick at least one of each kind.

    (a) Set up a generating function for a_k.

    (b) The sequence (a_k) is the convolution of the sequence (b_k) whose generating function is x^p and a sequence (c_k). Find c_k.

    (c) Find a_k for all k.

14. In Exercise 7 of Section 5.3, solve for a_k. (*Hint:* Set $y = x^2$ in the generating function.)
15. Suppose that $B(x)$ is the ordinary generating function for the sequence (b_k). Let

$$Sb_k = b_0 + b_1 + \cdots + b_k$$

and

$$S^2(b_k) = S(Sb_k) = \sum_{j=0}^{k}(b_0 + b_1 + \cdots + b_j).$$

In general, let $a_k = S^p(b_k) = S(S^{p-1}(b_k))$. Then we can show that

$$a_k = b_k + pb_{k-1} + \cdots + \binom{p+j-1}{j} b_{k-j} + \cdots + \binom{p+k-1}{k} b_0.$$

(a) Verify this for $p = 2$.

(b) If $A(x)$ is the ordinary generating function for (a_k), find an expression for $A(x)$ in terms of $B(x)$.

16. Consider the list T-coloring problem of Example 5.28.

(a) Solve this problem when:

i. $r = 1$, $n = 5$, and $l = 22$

ii. $r = 1$, $n = 4$, and $l = 16$

iii. $r = 2$, $n = 5$, and $l = 16$

(b) Find the general solution for any positive integers r, n, and l.

17. Consider the weak order counting problem of Example 5.29. How many distinct weak orders on 10 elements have:

(a) 4 elements at the highest level?

(b) at most 4 elements at the highest level?

(c) an even number of elements at every level?

18. Let p_n^r be the number of partitions of the integer n into *exactly* r parts where order counts. For example, there are 10 partitions of 6 into exactly 4 parts where order matters, namely,

$$\begin{array}{ccccc} \{3,1,1,1\}, & \{1,3,1,1\}, & \{1,1,3,1\}, & \{1,1,1,3\}, & \{2,2,1,1\}, \\ \{2,1,2,1\}, & \{2,1,1,2\}, & \{1,2,2,1\}, & \{1,2,1,2\}, & \{1,1,2,2\}. \end{array}$$

(a) Set up an ordinary generating function for p_n^r.

(b) Solve for p_n^r.

19. A polynomial in the three variables u, v, w is called *homogeneous* if the total degree of each term $\alpha u^i v^j w^k$ is the same, that is, if $i + j + k$ is constant. For instance,

$$3v^4 + 2uv^2w + 4vw^3$$

is homogeneous with each term having total degree 4. What is the largest number of terms possible in a polynomial of three variables that is homogeneous of total degree n?

20. (a) Show that p_n^r as defined in Exercise 18 is the maximum number of terms in a homogeneous polynomial in r variables and having total degree n in which each term has each variable with degree at least 1.

(b) Use this result and the result of Exercise 18 to answer the question in Exercise 19.

21. Three people each roll a die once. In how many ways can the score add up to 9?

5.5 EXPONENTIAL GENERATING FUNCTIONS AND GENERATING FUNCTIONS FOR PERMUTATIONS

5.5.1 Definition of Exponential Generating Function

So far we have used ordinary generating functions to count the number of combinations of objects—we use the word *combination* because order does not matter. Let us now try to do something similar if order does matter and we are counting permutations. Recall that $P(n, k)$ is the number of k-permutations of an n-set. The ordinary generating function for $P(n, k)$ with n fixed is given by

$$\begin{aligned} G(x) &= P(n,0)x^0 + P(n,1)x^1 + P(n,2)x^2 + \cdots + P(n,n)x^n \\ &= \sum_{k=0}^{n} \frac{n!}{(n-k)!} x^k. \end{aligned}$$

Unfortunately, there is no good way to simplify this expression. Had we been dealing with combinations, and the number of ways $C(n, k)$ of choosing k elements out of an n-set, we would have been able to simplify, for we would have had the expression

$$C(n,0)x^0 + C(n,1)x^1 + C(n,2)x^2 + \cdots + C(n,n)x^n, \tag{5.26}$$

which by the binomial expansion simplifies to $(1 + x)^n$. By Theorem 2.1,

$$P(n,r) = C(n,r)P(r,r) = C(n,r)r!.$$

Hence, the equivalence of (5.26) to $(1 + x)^n$ can be rewritten as

$$P(n,0)\frac{x^0}{0!} + P(n,1)\frac{x^1}{1!} + P(n,2)\frac{x^2}{2!} + \cdots + P(n,n)\frac{x^n}{n!} = (1+x)^n. \tag{5.27}$$

The number $P(n, k)$ is the coefficient of $x^k/k!$ in the expansion of $(1 + x)^n$.

This suggests the following idea. If (a_k) is any sequence, the *exponential generating function* for the sequence is the function

$$\begin{aligned} H(x) &= a_0\frac{x^0}{0!} + a_1\frac{x^1}{1!} + a_2\frac{x^2}{2!} + \cdots + a_k\frac{x^k}{k!} + \cdots \\ &= \sum_k a_k \frac{x^k}{k!}. \end{aligned}$$

As with the ordinary generating function, we think of x as being chosen so that the sum converges.[5]

[5] As mentioned in the footnote on page 286, we can make this precise using the notion of formal power series.

To give an example, if $a_k = 1$, for $k = 0, 1, \ldots$, then, using Equation (5.3), we see that the exponential generating function is

$$\begin{aligned} H(x) &= 1 \cdot \frac{x^0}{0!} + 1 \cdot \frac{x^1}{1!} + 1 \cdot \frac{x^2}{2!} + \cdots \\ &= 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \cdots \\ &= e^x. \end{aligned}$$

To give another example, if $a_k = P(n, k)$, we have shown in (5.27) that the exponential generating function is $(1+x)^n$. To give still one more example, suppose that α is any real number and (a_k) is the sequence $(1, \alpha, \alpha^2, \alpha^3, \ldots)$. Then the exponential generating function for (a_k) is

$$\begin{aligned} H(x) &= \sum_{k=0}^{\infty} \alpha^k \frac{x^k}{k!} \\ &= \sum_{k=0}^{\infty} \frac{(\alpha x)^k}{k!} \\ &= e^{\alpha x}. \end{aligned}$$

Just as with ordinary generating functions, we will want to go back and forth between sequences and exponential generating functions.

Example 5.30 Eulerian Graphs A connected graph will be called *eulerian* if every vertex has even degree. Eulerian graphs will be very important in a variety of applications discussed in Chapter 11. Harary and Palmer [1973] and Read [1962] show that if u_n is the number of labeled, connected eulerian graphs of n vertices, the exponential generating function $U(x)$ for the sequence (u_n) is given by

$$U(x) = x + \frac{x^3}{3!} + \frac{3x^4}{4!} + \frac{38x^5}{5!} + \cdots.$$

Thus, there is one labeled, connected eulerian graph of three vertices and there are three of four vertices. These are shown in Figure 5.7. ∎

5.5.2 Applications to Counting Permutations

Example 5.31 A code can use three different letters, a, b, or c. A sequence of five or fewer letters gives a codeword. The codeword can use at most one b, at most one c, and at most three a's. How many possible codewords are there of length k, with $k \leq 5$? Note that order matters in a codeword. For example, codewords aab and aba are different, whereas previously, when considering subsets, $\{a, a, b\}$ and $\{a, b, a\}$ were the same. So we are interested in counting permutations rather than

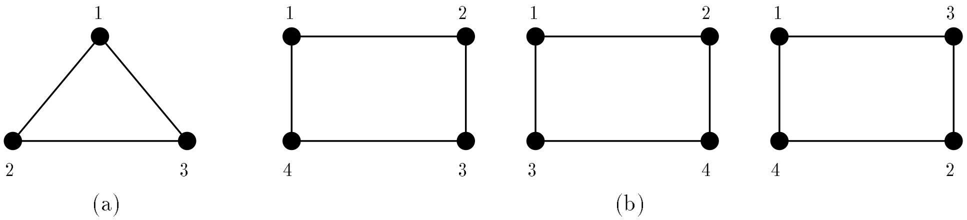


Figure 5.7: The labeled, connected eulerian graphs of (a) three vertices and (b) four vertices.

combinations. However, taking a hint from our previous experience, let us begin by counting combinations, the number of ways of getting k letters if it is possible to pick at most one b, at most one c, and at most three a's. The ordinary generating function is calculated by taking

$$(1 + ax + a^2x^2 + a^3x^3)(1 + bx)(1 + cx),$$

which equals

$$1 + (a + b + c)x + (bc + a^2 + ab + ac)x^2 + (a^3 + abc + a^2b + a^2c)x^3 +$$
$$(a^2bc + a^3b + a^3c)x^4 + a^3bcx^5.$$

The coefficient of x^k gives the ways of obtaining k letters. For example, three letters can be obtained as follows: 3 a's, a and b and c, 2 a's and b, or 2 a's and c. If we make a choice of a and b and c, there are 3! corresponding permutations:

$$abc, \quad acb, \quad bac, \quad bca, \quad cab, \quad cba.$$

For the 3 a's choice, there is only one corresponding permutation: aaa. For the 2 a's and b choice, there are 3 permutations:

$$aab, \quad aba, \quad baa.$$

From our general formula of Theorem 2.6 we see why this is true: The number of distinguishable permutations of 3 objects with 2 of one type and 1 of another is given by

$$\frac{3!}{2!1!}.$$

In general, if we have n_1 a's, n_2 b's, and n_3 c's, the number of corresponding permutations is

$$\frac{n!}{n_1!n_2!n_3!}.$$

In particular, in our schematic, the proper information for the ways to obtain codewords if three letters are chosen is given by

$$\frac{3!}{3!}a^3 + \frac{3!}{1!1!1!}abc + \frac{3!}{2!1!}a^2b + \frac{3!}{2!1!}a^2c. \tag{5.28}$$

Setting $a = b = c = 1$ would yield the proper count of number of such codewords of three letters. We can obtain (5.28) and the other appropriate coefficients by the trick of using

$$\frac{(ax)^p}{p!} = \frac{a^p}{p!}x^p$$

instead of $a^p x^p$ to derive our schematic generating function. In our example, we have

$$\left(1 + \frac{a}{1!}x + \frac{a^2}{2!}x^2 + \frac{a^3}{3!}x^3\right)\left(1 + \frac{b}{1!}x\right)\left(1 + \frac{c}{1!}x\right),$$

which equals

$$1 + \left(\frac{a}{1!} + \frac{b}{1!} + \frac{c}{1!}\right)x + \left(\frac{bc}{1!1!} + \frac{a^2}{2!} + \frac{ab}{1!1!} + \frac{ac}{1!1!}\right)x^2 +$$
$$\left(\frac{a^3}{3!} + \frac{abc}{1!1!1!} + \frac{a^2b}{2!1!} + \frac{a^2c}{2!1!}\right)x^3 + \left(\frac{a^2bc}{2!1!1!} + \frac{a^3b}{3!1!} + \frac{a^3c}{3!1!}\right)x^4 + \frac{a^3bc}{3!1!1!}x^5. \quad (5.29)$$

This is still not a satisfactory schematic—compare the coefficients of x^3 to the expression in (5.28). However, the schematic works if we consider this as an exponential generating function, and choose the coefficient of $x^k/k!$. For the expression (5.29) is equal to

$$1 + 1!\left(\frac{a}{1!} + \frac{b}{1!} + \frac{c}{1!}\right)\frac{x}{1!} + 2!\left(\frac{bc}{1!1!} + \frac{a^2}{2!} + \frac{ab}{1!1!} + \frac{ac}{1!1!}\right)\frac{x^2}{2!} +$$
$$3!\left(\frac{a^3}{3!} + \frac{abc}{1!1!1!} + \frac{a^2b}{2!1!} + \frac{a^2c}{2!1!}\right)\frac{x^3}{3!} + \cdots. \quad (5.30)$$

Setting $a = b = c = 1$ and taking the coefficient of $x^k/k!$ gives the appropriate number of codewords (permutations). For example, the number of length 3 is

$$3!\left(\frac{1}{3!} + 1 + \frac{1}{2!} + \frac{1}{2!}\right) = 13.$$

The corresponding codewords are the six we have listed with a, b, and c, the three with 2 a's and 1 b, the three with 2 a's and 1 c, and the one with 3 a's. ■

The analysis in Example 5.31 generalizes as follows.

Theorem 5.5 Suppose that we have p types of objects, with n_i indistinguishable objects of type $i, i = 1, 2, \ldots, p$. The number of distinguishable permutations of length k with up to n_i objects of type i is the coefficient of $x^k/k!$ in the exponential generating function

$$\left(1 + x + \frac{x^2}{2!} + \cdots + \frac{x^{n_1}}{n_1!}\right)\left(1 + x + \frac{x^2}{2!} + \cdots + \frac{x^{n_2}}{n_2!}\right)\cdots\left(1 + x + \frac{x^2}{2!} + \cdots + \frac{x^{n_p}}{n_p!}\right).$$

Example 5.32 RNA Chains To give an application of this result, let us consider the number of 2-link RNA chains if we have available up to 3 A's, up to 3 G's, up to 2 C's, and up to 1 U. Since order matters, we seek an exponential generating function. This is given by

$$\left(1+x+\frac{x^2}{2!}+\frac{x^3}{3!}\right)^2\left(1+x+\frac{x^2}{2!}\right)(1+x),$$

which turns out to equal

$$1+4x+\frac{15}{2}x^2+\frac{53}{6}x^3+\cdots.$$

Here, the coefficient of x^2 is $15/2$, so the coefficient of $x^2/2!$ is $2!(15/2)=15$. Thus, there are 15 such chains. They are AA, AG, AC, AU, GA, GG, GC, GU, CA, CG, CC, CU, UA, UG, and UC, that is, all but UU. Similarly, the number of 3-link RNA chains made up from these available bases is the coefficient of $x^3/3!$, or $3!(53/6)=53$. The reader can readily check this result. ■

Example 5.33 RNA Chains Continued To continue with Example 5.32, suppose that we wish to find the number of RNA chains of length k if we assume an arbitrarily large supply of each base. The exponential generating function is given by

$$\begin{aligned}\left(1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots\right)^4 &= (e^x)^4\\ &= e^{4x}\\ &= \sum_{k=0}^{\infty}\frac{(4x)^k}{k!}\\ &= \sum_{k=0}^{\infty}4^k\frac{x^k}{k!}.\end{aligned}$$

Thus, the number in question is given by 4^k. This agrees with what we already concluded in Chapter 2, by a simple use of the product rule.

Let us make one modification here, namely, to count the number of RNA chains of length k if the number of U links is even. The exponential generating function is given by

$$H(x)=\left(1+\frac{x^2}{2!}+\frac{x^4}{4!}+\cdots\right)\left(1+x+\frac{x^2}{2!}+\cdots\right)^3.$$

Now the second term in $H(x)$ is given by $(e^x)^3=e^{3x}$. It is also not hard to show that the first term is given by

$$\frac{1}{2}(e^x+e^{-x}).$$

Thus,

$$\begin{aligned} H(x) &= \frac{1}{2}(e^x + e^{-x})(e^{3x}) \\ &= \frac{1}{2}(e^{4x} + e^{2x}) \\ &= \frac{1}{2}\left[\sum_{k=0}^{\infty} 4^k \frac{x^k}{k!} + \sum_{k=0}^{\infty} 2^k \frac{x^k}{k!}\right] \\ &= \sum_{k=0}^{\infty} \left(\frac{4^k + 2^k}{2}\right) \frac{x^k}{k!}. \end{aligned}$$

We conclude that the number of RNA chains in question is

$$\frac{4^k + 2^k}{2}.$$

To check this, note for example that if $k = 2$, this number is 10. The 10 chains are UU, GG, GA, GC, AG, AA, AC, CG, CA, and CC. ■

5.5.3 Distributions of Distinguishable Balls into Indistinguishable Cells[6]

Recall from Section 2.10.4 that the Stirling number of the second kind, $S(n, k)$, is defined to be the number of distributions of n distinguishable balls into k indistinguishable cells with no cell empty. Here we shall show that

$$S(n, k) = \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k - i)^n. \tag{5.31}$$

Let us first consider the problem of finding the number $T(n, k)$ of ways to put n distinguishable balls into k distinguishable cells labeled $1, 2, \ldots, k$, with no cell empty. Note that

$$T(n, k) = k!S(n, k), \tag{5.32}$$

since we obtain a distribution of n distinguishable balls into k distinguishable cells with no cell empty by finding a distribution of n distinguishable balls into k indistinguishable cells with no cell empty and then labeling (ordering) the cells. Next we compute $T(n, k)$. Suppose that ball i goes into cell $C(i)$. We can encode the distribution of balls into distinguishable cells by giving a sequence $C(1)C(2)\cdots C(n)$. This is an n-permutation from the k-set $\{1, 2, \ldots, k\}$ with each label j in the k-set used at least once. Thus, $T(n, k)$ is the number of such permutations, and for fixed k, the exponential generating function for $T(n, k)$ is therefore given by

$$H(x) = \left(x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots\right)^k = (e^x - 1)^k.$$

[6] This subsection may be omitted.

$T(n,k)$ is given by the coefficient of $x^n/n!$ in the expansion of $H(x)$. By the binomial expansion (Theorem 2.7),

$$H(x) = \sum_{i=0}^{k} \binom{k}{i} (-1)^i e^{(k-i)x}.$$

Substituting $(k-i)x$ for x in the power series (5.3) for e^x, we obtain

$$\begin{aligned} H(x) &= \sum_{i=0}^{k} \binom{k}{i} (-1)^i \sum_{n=0}^{\infty} \frac{1}{n!}(k-i)^n x^n \\ &= \sum_{n=0}^{\infty} \frac{x^n}{n!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n. \end{aligned}$$

Finding the coefficient of $x^n/n!$ in the expansion of $H(x)$, we have

$$T(n,k) = \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n. \tag{5.33}$$

Now Equations (5.32) and (5.33) give us Equation (5.31).

EXERCISES FOR SECTION 5.5

1. For each of the following sequences (a_k), find a simple, closed-form expression for the exponential generating function.
   (a) $(5, 5, 5, \ldots)$ (b) $a_k = 3^k$ (c) $(1, 0, 0, 1, 1, \ldots)$
   (d) $(0, 0, 1, 1, \ldots)$ (e) $(1, 0, 1, 0, 1, \ldots)$ (f) $(2, 1, 2, 1, 2, 1, \ldots)$
2. For each of the following functions, find a sequence for which the function is the exponential generating function.
   (a) $4 + 4x + 4x^2 + 4x^3 + \cdots$ (b) $\dfrac{3}{1-x}$ (c) $x^2 + 5e^x$
   (d) $x^2 + 4x^3 + x^5$ (e) e^{6x} (f) $5e^x$
   (g) $e^{2x} + e^{5x}$ (h) $(1+x^2)^n$ (i) $\dfrac{1}{1-6x}$
3. A graph is said to be *even* if every vertex has even degree. If e_k is the number of labeled, even graphs of k vertices, Harary and Palmer [1973] show that the exponential generating function $E(x)$ for the sequence (e_k) is given by
   $$E(x) = x + \frac{x^2}{2!} + \frac{2x^3}{3!} + \frac{8x^4}{4!} + \cdots.$$
   Verify the coefficients of $x^3/3!$ and $x^4/4!$. (Note that e_k can be derived from Exercise 11, Section 11.3, and the results of Section 3.1.3.)
4. In Example 5.32, check by enumeration that there are 53 3-link RNA chains made up from the available bases.

5. Find the number of 3-link RNA chains if the available bases are 2 A's, 3 G's, 3C's, and 1 U. Check your answer by enumeration.

6. In each of the following, set up the appropriate generating function, but do not calculate an answer. Indicate what you are looking for, for example, the coefficient of x^8.

   (a) How many codewords of three letters can be built from the letters a, b, c, and d if b and d can only be picked once?

   (b) A codeword consists of at least one of each of the digits 0, 1, 2, 3, and 4, and has length 6. How many such codewords are there?

   (c) How many 11-digit numbers consist of at most four 0's, at most three 1's, and at most four 2's?

   (d) In how many ways can $3n$ letters be selected from $2n$ A's, $2n$ B's, and $2n$ C's?

   (e) If n is a fixed even number, find the number of n-digit words generated from the alphabet $\{0, 1, 2, 3\}$ in each of which the number of 0's and the number of 1's is even and the number of 2's is odd.

   (f) In how many ways can a total of 100 be obtained if 50 dice are rolled?

   (g) Ten municipal bonds are each to be rated as A, AA, or AAA. In how many different ways can the ratings be assigned?

   (h) In planning a schedule for the next 20 days at a job, in how many ways can one schedule the 20 days using at most 5 vacation days, at most 5 personal days, and at most 15 working days?

   (i) Suppose that with a type A coin, you get 1 point if the coin turns up heads and 2 points if it turns up tails. With a type B coin, you get 2 points for a head and 3 points for a tail. In how many ways can you get 12 points if you toss 3 type A coins and 5 type B coins?

   (j) In how many ways can 200 identical terminals be divided among four computer rooms so that each room will have 20 or 40 or 60 or 80 or 100 terminals?

   (k) One way for a ship to communicate with another visually is to hang a sequence of colored flags from a flagpole. The meaning of a signal depends on the order of the flags from top to bottom. If there are available 5 red flags, 4 green ones, 4 yellow ones, and 1 blue one, how many different signals are possible if 12 flags are to be used?

   (l) In part (k), how many different signals are possible if at least 12 flags are to be used?

7. (a) Find the number of RNA chains of length k if the number of A's is odd.

   (b) Illustrate for $k = 2$.

8. Find the number of RNA chains of length 2 with an even number of U's or an odd number of A's.

9. Suppose that there are p different kinds of objects, each in infinite supply. Let a_k be the number of permutations of k objects chosen from these objects. Find a_k explicitly by using exponential generating functions.

10. In how many ways can 60 identical terminals be divided among two computer rooms so that each room will have 20 or 40 terminals?

11. If order matters, find an exponential generating function for the number of partitions of integer k (Example 5.23 and Exercise 18, Section 5.4).
12. Find a simple, closed-form expression for the exponential generating function if we have p types of objects, each in infinite supply, and we wish to choose k objects, at least one of each kind, and order matters.
13. Find a simple, closed-form expression for the exponential generating function if we have p types of objects, each in infinite supply, and we wish to choose k objects, with an even number (including 0) of each kind, and order matters.
14. Find the number of codewords of length k from an alphabet $\{a, b, c, d, e\}$ if b occurs an odd number of times.
15. Find the number of codewords of length 3 from an alphabet $\{1, 2, 3, 4, 5, 6\}$ if 1, 3, 4, and 6 occur an even number of times.
16. Compute $S(4, 2)$ and $T(4, 2)$ from Equations (5.31) and (5.33), respectively, and check your answers by listing all the appropriate distributions.
17. Exercises 17–20 investigate combinations of exponential generating functions. Suppose that $A(x)$ and $B(x)$ are the exponential generating functions for the sequences (a_k) and (b_k), respectively. Find an expression for the kth term c_k of the sequence (c_k) whose exponential generating function is $C(x) = A(x) + B(x)$.
18. Repeat Exercise 17 for $C(x) = A(x)B(x)$.
19. Find a_3 if the exponential generating function for (a_k) is:

    (a) $e^x(1 + x)^6$ (b) $\dfrac{e^{3x}}{1 - x}$ (c) $\dfrac{x^2}{(1 - x)^2}$

20. Suppose that $a_{n+1} = (n + 1)b_n$, with $a_0 = b_0 = 1$. If $A(x)$ is the exponential generating function for the sequence (a_n) and $B(x)$ is the exponential generating function for the sequence (b_n), derive a relation between $A(x)$ and $B(x)$.

5.6 PROBABILITY GENERATING FUNCTIONS[7]

The simple idea of a generating function has interesting uses in the study of probability. In fact, the first complete treatment of generating functions was by Laplace in his *Théorie Analytique des Probabilités* (Paris, 1812), and much of the motivation for the development of generating functions came from probability. Suppose that after an experiment is performed, it is known that one and only one of a (finite or countably infinite) set of possible events will occur. Let p_k be the probability that the kth event occurs, $k = 0, 1, 2, \ldots$. (Of course, this notation does not work if there is a continuum of possible events.) The ordinary generating function

$$G(x) = \sum p_k x^k \tag{5.34}$$

[7] This section may be omitted without loss of continuity. Although it is essentially self-contained, the reader with some prior exposure to probability theory, at least at the level of a "finite math" book such as Goodman and Ratti [1992] or Kemeny, Snell, and Thompson [1974], will get more out of this.

is called the *probability generating function.* [Note that (5.34) converges at least for $|x| \leq 1$, since $p_0 + p_1 + \cdots + p_k + \cdots = 1$.] We shall see that probability generating functions are extremely useful in evaluating experiments, in particular in analyzing roughly what we "expect" the outcomes to be.

Example 5.34 Coin Tossing Suppose that the experiment is tossing a fair coin. Then the events are heads (H) and tails (T), with p_0, probability of H, equal to 1/2, and p_1, probability of T, equal to 1/2. Hence, the probability generating function is

$$G(x) = \frac{1}{2} + \frac{1}{2}x. \qquad \blacksquare$$

Example 5.35 Bernoulli Trials In Bernoulli trials there are n independent repeated trials of an experiment, with each trial leading to a success with probability p and a failure with probability $q = 1 - p$. The experiment could be a test to see if a product is defective or nondefective, a test for the presence or absence of a disease, or a decision about whether to accept or reject a candidate for a job. If S stands for success and F for failure, a typical outcome in $n = 5$ trials is a sequence like $SSFSF$ or $SSFFF$. The probability that in n trials there will be k successes is given by

$$b(k, n, p) = C(n, k)p^k q^{n-k},$$

as is shown in any standard book on probability theory (such as Feller [1968], Parzen [1992], or Ross [1997]), or on finite mathematics (such as Goodman and Ratti [1992] or Kemeny, Snell, and Thompson [1974]). The probability generating function for the number of successes in n trials is given by

$$\begin{aligned} G(x) &= \sum_{k=0}^{n} b(k, n, p)x^k \\ &= \sum_{k=0}^{n} C(n, k)p^k q^{n-k} x^k. \end{aligned}$$

By the binomial expansion (Theorem 2.7), we have

$$G(x) = (px + q)^n. \qquad \blacksquare$$

Let us note some simple results about probability generating functions.

Theorem 5.6 If G is a probability generating function, then

$$G(1) = 1.$$

Proof. Since the outcomes are mutually exclusive and exhaustive by assumption, we have

$$p_0 + p_1 + \cdots + p_k + \cdots = 1. \qquad \text{Q.E.D.}$$

Corollary 5.6.1

$$\sum_{k=0}^{n} C(n,k)p^k q^{n-k} = 1.$$

Proof. In Bernoulli trials (Example 5.35), set $G(1) = 1$. Q.E.D.

Corollary 5.6.1 may also be proved directly from the binomial expansion, noting that

$$(p+q)^n = 1^n.$$

Suppose that in an experiment, if the kth event occurs, we get k dollars (or k units of some reward). Then the expression $E = \sum kp_k$ is called the *expected value* or the *expectation*. It is what we expect to "win" on the average if the experiment is repeated many times, and we expect 0 dollars a fraction p_0 of the time, 1 dollar a fraction p_1 of the time, and so on. For a more detailed discussion of expected value, see any elementary book on probability theory or on finite mathematics. Note that the expected value is defined only if the sum $\sum kp_k$ converges. If the sum does converge, we say that the expected value *exists*. We can have the same expected value in an experiment that always gives 1 dollar and in an experiment that gives 0 dollars with probability $\frac{1}{2}$ and 2 dollars with probability $\frac{1}{2}$. However, there is more variation in outcomes in the second experiment. Probability theorists have introduced the concept of variance to measure this variation. Specifically, the *variance* V is defined to be

$$V = \sum_k k^2 p_k - \left(\sum_k kp_k\right)^2. \tag{5.35}$$

See a probability book such as Feller [1968], Parzen [1992], or Ross [1997] for a careful explanation of this concept. Variance is defined only if the sums in (5.35) converge. In case they do converge, we say that the variance *exists*. In the first experiment mentioned above,

$$V = \left[1^2(1)\right] - [1(1)]^2 = 0.$$

In the second experiment mentioned above,

$$V = \left[0^2\left(\frac{1}{2}\right) + 2^2\left(\frac{1}{2}\right)\right] - \left[0\left(\frac{1}{2}\right) + 2\left(\frac{1}{2}\right)\right]^2 = 1.$$

Hence, the variance is higher in the second experiment. We shall see how the probability generating function allows us to calculate expected value and variance.

Differentiating (5.34) with respect to x leads to the equation

$$G'(x) = \sum kp_k x^{k-1}.$$

Hence, if $G'(x)$ converges for $x = 1$, that is, if $\sum kp_k$ converges, then

$$G'(1) = \sum kp_k. \tag{5.36}$$

If the kth event gives value k dollars or units, the expression on the right-hand side of (5.36) is the expected value.

Theorem 5.7 Suppose that $G(x)$ is the probability generating function and the kth event gives value k. If the expected value exists, $G'(1)$ is the expected value.

Let us apply Theorem 5.7 to the case of Bernoulli trials. We have

$$\begin{aligned} G(x) &= (px+q)^n \\ G'(x) &= n(px+q)^{n-1}p \\ G'(1) &= np(p+q)^{n-1} \\ &= np(1)^{n-1} \\ &= np. \end{aligned}$$

Thus, the expected number of successes in n trials is np. The reader who recalls the "standard" derivation of this fact should be pleased at how simple this derivation is. To illustrate the result, we note that in $n = 100$ tosses of a fair coin, the probability of a head (success) is $p = .5$ and the expected number of heads is $np = 50$.

Example 5.36 Chip Manufacturing A company manufacturing computer chips estimates that one chip in every 10,000 manufactured is defective. If an order comes in for 100,000 chips, what is the expected number of defective chips in that order? Assuming that defects appear independently, we have an example of Bernoulli trials and we see that the expected number is $(100,000)\dfrac{1}{10,000} = 10$. ■

Example 5.37 Packet Transmission In data transmission, the probability that a transmitted "packet" is lost is 1 in 1000. What is the expected number of packets transmitted before one is lost? Assuming that packet loss is independent from packet to packet, we have Bernoulli trials. We can try to calculate the probability that the *first success* occurs on trial k and then compute the expected value of the first success. We ask the reader to investigate this in Exercise 7. ■

The next theorem is concerned with variance. Its proof is left to the reader (Exercise 9).

Theorem 5.8 Suppose that $G(x)$ is the probability generating function and the kth event has value k. If the variance V exists, V is given by $V = G''(1) + G'(1) - [G'(1)]^2$.

Applying Theorem 5.8 to Bernoulli trials, we have

$$G''(x) = n(n-1)p^2(px+q)^{n-2}.$$

Also,

$$\begin{aligned} G'(1) &= np \\ G''(1) &= n(n-1)p^2. \end{aligned}$$

Hence,

$$\begin{aligned} V &= G''(1) + G'(1) - [G'(1)]^2 \\ &= n(n-1)p^2 + np - n^2p^2 \\ &= np - np^2 \\ &= np(1-p) \\ &= npq. \end{aligned}$$

This gives npq for the variance, a well-known formula.

EXERCISES FOR SECTION 5.6

1. In each of the following situations, find a simple, closed-form expression for the probability generating function, and use this to compute expected value and variance.
   (a) $p_0 = p_1 = p_2 = \frac{1}{3}, p_k = 0$ otherwise.
   (b) $p_4 = \frac{2}{5}, p_7 = \frac{3}{5}, p_k = 0$ otherwise.
   (c) $p_1 = \frac{1}{4}, p_2 = \frac{1}{4}, p_4 = \frac{1}{2}, p_k = 0$ otherwise.
2. A company manufacturing small engines estimates that two engines in every 100,000 manufactured are defective. If an order comes in for 100 engines, what is the expected number of defective engines in the order?
3. A certain disease is thought to be noncontagious. A researcher estimates the disease to be found in 1 in every 50 people. What is the expected number of people the researcher has to examine before finding a person with the disease? What is the variance?
4. For fixed positive number λ, the *Poisson distribution* with parameter λ has
$$p_k = \frac{e^{-\lambda}\lambda^k}{k!}, \qquad k = 0, 1, 2, \ldots .$$
   (a) Find a simple, closed-form expression for the probability generating function.
   (b) Use the methods of generating functions to find the expected value and the variance.
5. (Daniel [1995]) Gibbons, Clark, and Fawcett [1990] studied the monthly distribution of adolescent suicides in Cook County, Illinois, between 1977 and 1987. They found that it closely followed a Poisson distribution with parameter $\lambda = 2.75$.
   (a) Find the probability that if a month is selected at random, it will have four adolescent suicides.

(b) Find the expected number of suicides per month.

(c) Find the variance.

6. (Daniel [1995]) Suppose that a large number of samples are taken from a pond and the average number of aquatic organisms of a given kind found in a sample is 2. Assuming that the number of organisms follows a Poisson distribution, find the probability that the next sample drawn will have three or four such organisms.
7. In Bernoulli trials, suppose that we compute the probability that the first success occurs on trial k. The probability is given by $p_k = 0$, $k = 0$ (assuming that we start with trial 1), and $p_k = (1-p)^{k-1}p$, $k > 0$. The probabilities p_k define the *geometric distribution*. Repeat Exercise 4 for this distribution and apply the results to the question in Example 5.37.
8. Fix a positive integer m. In Bernoulli trials, the probability that the mth success takes place on trial $k+m$ is given by

$$p_k = \binom{k+m-1}{k} q^k p^m.$$

The probabilities p_k define the *negative binomial distribution*.

(a) Show that the probability generating function $G(x)$ for the negative binomial distribution p_k is given by

$$G(x) = \frac{p^m}{(1-qx)^m}.$$

(b) Compute expected value and variance.

9. Prove Theorem 5.8.

5.7 THE COLEMAN AND BANZHAF POWER INDICES[8]

In Section 2.15 we introduced the notion of a simple game and the Shapley-Shubik power index. Here, we shall define two alternative power indices and discuss how to use generating functions to calculate them. We defined the *value* $v(S)$ of a coalition S to be 1 if S is winning and 0 if S is losing. Coleman [1971] defines the power of player i as

$$P_i^C = \frac{\sum_S [v(S) - v(S - \{i\})]}{\sum_S v(S)}. \tag{5.37}$$

In calculating this measure, one takes the sums over all coalitions S. The term

$$v\,(S) - v\,(S - \{i\})$$

is 1 if removal of i changes S from winning to losing, and it is 0 otherwise. (It cannot be -1, since we assumed that a winning coalition can never be contained in a losing one.) Thus, P_i^C is the number of winning coalitions from which removal of i leads to

[8] This section may be omitted without loss of continuity.

a losing coalition divided by the number of winning coalitions, or the proportion of winning coalitions in which i's defection is critical. This index avoids the seemingly extraneous notion of order that underlies the computation of the Shapley-Shubik index.

It is interesting to note that the Shapley-Shubik index p_i^S can be calculated by a formula similar to (5.37). For Shapley [1953] proved that

$$p_i^S = \sum_S \{\gamma(s)[v(S) - v(S - \{i\})] : S \text{ such that } i \in S\}, \tag{5.38}$$

where

$$s = |S| \quad \text{and} \quad \gamma(s) = \frac{(s-1)!(n-s)!}{n!}.$$

(See Exercise 15, Section 2.15.)

A variant of the Coleman power index is the Banzhaf index (Banzhaf [1965]), defined as

$$P_i^B = \frac{\sum_S [v(S) - v(S - \{i\})]}{\sum_{j=1}^n \sum_S [v(S) - v(S - \{j\})]}. \tag{5.39}$$

This index has the same numerator as Coleman's, while the denominator sums the numerators for all players j. Thus, P_i^B is the number of critical defections of player i divided by the total number of critical defections of all players, or player i's proportion of all critical defections.[9]

To give an example, let us consider the game $[51; 49, 48, 3]$. Here, the winning coalitions are $\{1,2,3\}, \{1,2\}, \{1,3\}, \{2,3\}$. Player 1's defection is critical to $\{1,2\}$ and $\{1,3\}$, so we have the Coleman index

$$P_1^C = \tfrac{2}{4} = \tfrac{1}{2}.$$

Similarly, each player's defection is critical to two coalitions, so

$$\begin{aligned} P_2^C &= \tfrac{2}{4} = \tfrac{1}{2} \\ P_3^C &= \tfrac{2}{4} = \tfrac{1}{2}. \end{aligned}$$

Note that in the Coleman index, the powers P_i^C may not add up to 1. It is the relative values that count. The Banzhaf index is given by

$$\begin{aligned} P_1^B &= \tfrac{2}{6} = \tfrac{1}{3} \\ P_2^B &= \tfrac{2}{6} = \tfrac{1}{3} \\ P_3^B &= \tfrac{2}{6} = \tfrac{1}{3}. \end{aligned}$$

[9] For a unifying framework for the Shapley-Shubik, Banzhaf, and Coleman indices, see Straffin [1980]. For a survey of the literature of the Shapley-Shubik index, see Shapley [1981]. For one on the Banzhaf and Coleman indices, see Dubey and Shapley [1979]. For applications of all three indices, see Brams, Lucas, and Straffin [1983], Brams, Schotter, and Schwödianer [1979], Lucas [1981, 1983], and Johnston [1995]. See also Section 2.15.5.

These two indices agree with the Shapley-Shubik index in saying that all three players have equal power. It is not hard to give examples where these indices may differ from that of Shapley-Shubik (see Exercise 2). (When a number of ways to measure something have been introduced, and they can differ, how do we choose among them? One approach is to lay down conditions or axioms that a reasonable measure should satisfy. We can then test different measures to see if they satisfy the axioms. One set of axioms, which is satisfied only by the Shapley-Shubik index, is due to Shapley [1953]; see Owen [1995], Dubey [1975], Myerson [1997], Roberts [1976], or Shapley [1981]. Another set of axioms, which is satisfied only by the Banzhaf index, is due to Dubey and Shapley [1979]; also see Owen [1978a,b] and Straffin [1980]. Felsenthal and Machover [1995] survey and expand on the axiomatic approaches to power indices. The axiomatic approach is probably the most reasonable procedure to use in convincing legislators or judges to use one measure over another, for legislators can then decide whether they like certain general conditions, rather than argue about a procedure. Incidentally, it is the Banzhaf index that has found use in the courts, in one-person, one-vote cases; see Lucas [1983].)

Generating functions can be used to calculate the numerator of P_i^C and P_i^B in case we have a weighted majority game $[q; v_1, v_2, \ldots, v_n]$. (Exercise 3 asks the reader to describe how to find the denominator of the former. The denominator of the latter is trivial to compute if all the numerators are known.) Suppose that player i has v_i votes. His defection will be critical if it comes from a coalition with q votes, or $q+1$ votes, or $\ldots$, or $q+v_i-1$ votes. His defection in these cases will lead to a coalition with $q-v_i$ votes, or $q-v_i+1$ votes, or $\ldots$, or $q-1$ votes. Suppose that $a_k^{(i)}$ is the number of coalitions with exactly k votes and not containing player i. Then the number of coalitions in which player i's defection is critical is given by

$$a_{q-v_i}^{(i)} + a_{q-v_i+1}^{(i)} + \cdots + a_{q-1}^{(i)} = \sum_{k=q-v_i}^{q-1} a_k^{(i)}. \tag{5.40}$$

This expression can be substituted for

$$\sum_S [v(S) - v(S-\{i\})]$$

in the computation of the Coleman or Banzhaf indices, provided that we can calculate the numbers $a_k^{(i)}$. Brams and Affuso [1976] point out that the numbers $a_k^{(i)}$ can be found using ordinary generating functions. To form a coalition, player j contributes either 0 votes or v_j votes. Hence, the ordinary generating function for the $a_k^{(i)}$ is given by

$$\begin{aligned} G^{(i)}(x) &= (1+x^{v_1})(1+x^{v_2})\cdots(1+x^{v_{i-1}})(1+x^{v_{i+1}})\cdots(1+x^{v_n}) \\ &= \Pi_{j\neq i}(1+x^{v_j}). \end{aligned}$$

The number $a_k^{(i)}$ is given by the coefficient of x^k.

Let us consider the weighted majority game $[4; 1, 2, 4]$ as an example. We have

$$\begin{aligned} G^{(1)}(x) &= (1+x^2)(1+x^4) &= 1+x^2+x^4+x^6 \\ G^{(2)}(x) &= (1+x)(1+x^4) &= 1+x+x^4+x^5 \\ G^{(3)}(x) &= (1+x)(1+x^2) &= 1+x+x^2+x^3. \end{aligned}$$

Thus, for example, $a_4^{(i)}$ is the coefficient of x^4 in $G^{(1)}(x)$, i.e., it is 1. There is one coalition not containing player 1 that has exactly four votes: namely, the coalition consisting of the third player alone. Using (5.40), we obtain

$$\begin{aligned} \textstyle\sum_S[v(S) - v(S - \{1\})] &= a_{4-1}^{(1)} = a_3^{(1)} = 0 \\ \textstyle\sum_S[v(S) - v(S - \{2\})] &= a_{4-2}^{(2)} + a_{4-2+1}^{(2)} = a_2^{(2)} + a_3^{(2)} = 0 \\ \textstyle\sum_S[v(S) - v(S - \{3\})] &= a_{4-4}^{(3)} + a_{4-4+1}^{(3)} + a_{4-4+2}^{(3)} + a_{4-4+3}^{(3)} \\ &= a_0^{(3)} + a_1^{(3)} + a_2^{(3)} + a_3^{(3)} \\ &= 4. \end{aligned}$$

This immediately gives us

$$\begin{aligned} P_1^B &= \tfrac{0}{4} = 0 \\ P_2^B &= \tfrac{0}{4} = 0 \\ P_3^B &= \tfrac{4}{4} = 1. \end{aligned}$$

According to the Banzhaf index, player 3 has all the power. This makes sense: No coalition can be winning without him. The Coleman index and Shapley-Shubik index give rise to the same values. Computation is left to the reader.

EXERCISES FOR SECTION 5.7

1. Calculate the Banzhaf and Coleman power indices for each of the following games, using generating functions to calculate the numerators. Check your answer using the definitions of these indices.
   (a) [51; 51, 48, 1] (b) [51; 49, 47, 4] (c) [51; 40, 30, 20, 10]
   (d) [20; 1, 10, 10, 10] (e) [102; 80, 40, 80, 20]
   (f) The Australian government "game" (Section 2.15.1): [5; 1, 1, 1, 1, 1, 1, 3]
   (g) The Board of Supervisors, Nassau County, NY, 1964: [59; 31, 31, 21, 28, 2, 2]
2. Give an example of a game where:
   (a) The Banzhaf and Coleman power indices differ
   (b) The Banzhaf and Shapley-Shubik power indices differ
   (c) The Coleman and Shapley-Shubik power indices differ
   (d) All three of these indices differ
3. Describe how to find $\sum_s v(S)$ by generating functions.

4. Use the formula of Equation (5.38) to calculate the Shapley-Shubik power index of each of the weighted majority games in Exercises 1(a)–(e).
5. (a) Explain how you could use generating functions to compute the Shapley-Shubik power index.
   (b) Apply your results to the games in Exercise 1.

REFERENCES FOR CHAPTER 5

Banzhaf, J. F., III, "Weighted Voting Doesn't Work: A Mathematical Analysis," *Rutgers Law Rev., 19* (1965), 317–343.

Berge, C., *Principles of Combinatorics*, Academic Press, New York, 1971.

Brams, S. J., and Affuso, P. J., "Power and Size: A Near Paradox," *Theory and Decision, 1* (1976), 68–94.

Brams, S. J., Lucas, W. F., and Straffin, P. D. (eds.), *Political and Related Models*, Modules in Applied Mathematics, Vol. 2, Springer-Verlag, New York, 1983.

Brams, S. J., Schotter, A., and Schwödianer, G. (eds.), *Applied Game Theory*, IHS-Studies No. 1, Physica-Verlag, Würzburg, 1979.

Bressoud, D., and Wagon, S., *A Course in Computational Number Theory*, Key College Publishing, Emeryville, CA, 2000.

Cohen, D. I. A., *Basic Techniques of Combinatorial Theory*, Wiley, New York, 1978.

Coleman, J. S., "Control of Collectivities and the Power of a Collectivity to Act," in B. Lieberman (ed.), *Social Choice*, Gordon and Breach, New York, 1971, 269–300.

Daniel, W. W., *Biostatistics: A Foundation for Analysis in the Health Sciences*, 6th ed., Wiley, New York, 1995.

Dubey, P., "On the Uniqueness of the Shapley Value," *Int. J. Game Theory, 4* (1975), 131–140.

Dubey, P., and Shapley, L. S., "Mathematical Properties of the Banzhaf Power Index," *Math. Oper. Res., 4* (1979), 99–131.

Feller, W., *An Introduction to Probability Theory and Its Applications*, 3rd ed., Wiley, New York, 1968.

Felsenthal, D. S., and Machover, M., "Postulates and Paradoxes of Relative Voting Power – A Critical Re-Appraisal," *Theory and Decision, 38* (1995), 195–229.

Gallian, J. A., *Contemporary Abstract Algebra*, 5th ed., Houghton Mifflin, Boston, 2002.

Gardner, M., "Mathematical Games," *Scientific American, 238* (1978), 19–32.

Gibbons, R. D., Clark, D. C., and Fawcett, J., "A Statistical Method for Evaluating Suicide Clusters and Implementing Cluster Surveillance," *Am. J. Epidemiol., 132* (1990), S183–S191.

Goodman, A. W., and Ratti, J. S., *Finite Mathematics with Applications*, 4th ed., Prentice Hall, Upper Saddle River, NJ, 1992.

Harary, F., and Palmer, E. M., *Graphical Enumeration*, Academic Press, New York, 1973.

Hardy, G. H., and Wright, E. M., *An Introduction to the Theory of Numbers*, 5th ed., Oxford University Press, New York, 1980.

Johnston, R. J., "The Conflict over Qualified Majority Voting in the European Union Council of Ministers: An Analysis of the UK Negotiating Stance Using Power Indices," *British J. Pol. Science, 25* (1995), 245–288.

KEMENY, J. G., SNELL, J. L., and THOMPSON, G. L., *Introduction to Finite Mathematics*, Prentice Hall, Englewood Cliffs, NJ, 1974.

LANDO, S. K., *Lectures on Generating Functions*, Student Mathematical Library, Vol. 23, American Mathematical Society, Providence, RI, 2003.

LUCAS, W. F., "Applications of Cooperative Games to Equitable Allocation," in W. F. Lucas (ed.), *Game Theory and Its Applications*, Proceedings of Symposia in Applied Mathematics, Vol. 24, American Mathematical Society, Providence, RI, 1981, 19–36.

LUCAS, W. F., " Measuring Power in Weighted Voting Systems," in S. J. Brams, W. F. Lucas, and P. D. Straffin (eds.), *Political and Related Models*, Modules in Applied Mathematics, Vol. 2, Springer-Verlag, New York, 1983, 183–238.

MACMAHON, P. A., *Combinatory Analysis*, Vols. 1 and 2, The University Press, Cambridge, 1915. (Reprinted in one volume by Chelsea, New York, 1960.)

MYERSON, R. B., *Game Theory*, Harvard University Press, Cambridge, MA, 1997.

NIVEN, I., "Formal Power Series," *Amer. Math. Monthly, 76* (1969), 871–889.

OWEN, G., "Characterization of the Banzhaf-Coleman Index," *SIAM J. Applied Math., 35* (1978), 315–327. (a)

OWEN, G., " A Note on the Banzhaf-Coleman Axioms," in P. Ordeshook (ed.), *Game Theory and Political Science*, New York University Press, New York, 1978, 451–461. (b)

OWEN, G., *Game Theory*, 3rd ed., Academic Press, San Diego, CA, 1995.

PARZEN, E., *Modern Probability Theory and Its Applications*, Wiley, New York, 1992.

READ, R. C., "Euler Graphs on Labelled Nodes," *Canad. J. Math., 14* (1962), 482–486.

RIORDAN, J., "Generating Functions," in E. F. Beckenbach (ed.), *Applied Combinatorial Mathematics*, Wiley, New York, 1964, 67–95.

RIORDAN, J., *An Introduction to Combinatorial Analysis*, Princeton University Press, Princeton, NJ, 1980.

ROBERTS, F. S., *Discrete Mathematical Models, with Applications to Social, Biological, and Environmental Problems*, Prentice Hall, Englewood Cliffs, NJ, 1976.

ROSS, S. M., *A First Course in Probability*, 5th ed., Prentice Hall, Upper Saddle River, NJ, 1997.

SHAPLEY, L. S., "A Value for n-Person Games," in H. W. Kuhn and A. W. Tucker (eds.), *Contributions to the Theory of Games*, Vol. 2, Annals of Mathematics Studies No. 28, Princeton University Press, Princeton, NJ, 1953, 307–317.

SHAPLEY, L. S., "Measurement of Power in Political Systems," in W. F. Lucas (ed.), *Game Theory and Its Applications*, Proceedings of Symposia in Applied Mathematics, Vol. 24, American Mathematical Society, Providence, RI, 1981, 69–81.

SRIVASTAVA, H. M., and MANOCHA, H. L., *A Treatise on Generating Functions*, Halsted Press, New York, 1984.

STRAFFIN, P. D., JR., *Topics in the Theory of Voting*, Birkhäuser-Boston, Cambridge, MA, 1980.

TOMESCU, I., *Problems in Combinatorics and Graph Theory*, Wiley, Chichester, UK, 1985.

WILF, H. S., *generatingfunctionology*, A K Peters, Ltd., Wellesley, MA, 2006.

Chapter 6

Recurrence Relations[1]

6.1 SOME EXAMPLES

At the beginning of Section 5.1, we saw that we frequently want to count a quantity a_k that depends on an input or a parameter k. We then studied the sequence of unknown values, $a_0, a_1, a_2, \ldots, a_k, \ldots$. We shall see how to reduce computation of the kth or the $(k+1)$st member of such a sequence to earlier members of the sequence. In this way we can reduce a bigger problem to a smaller one or to one solved earlier. In Section 3.4 and in Example 5.14, we did much the same thing by giving reduction theorems, which reduced a complicated computation to simpler ones or ones made earlier. Having seen how to reduce computation of later terms of a sequence to earlier terms, we discuss several methods for finding general formulas for the kth term of an unknown sequence. The ideas and methods we present will have a wide variety of applications.

6.1.1 Some Simple Recurrences

Example 6.1 The Grains of Wheat According to Gamow [1954], the following is the story of King Shirham of India. The King wanted to reward his Grand Vizier, Sissa Ben Dahir, for inventing the game of chess. The Vizier made a modest request: Give me one grain of wheat for the first square on a chess board, two grains for the second square, four for the third square, eight for the fourth square, and so on until all the squares are covered. The King was delighted at the modesty of his Vizier's request, and granted it immediately. Did the King do a very wise thing? To answer this question, let s_k be the number of grains of wheat required for the first k squares and t_k be the number of grains for the kth square. We have

$$t_{k+1} = 2t_k. \tag{6.1}$$

[1]If Chapter 5 has been omitted, Sections 6.3 and 6.4 should be omitted. Chapters 5 and 6 are the only chapters that assume calculus except for the sake of "mathematical maturity." The only calculus used in Chapter 6 except in Sections 6.3 and 6.4 is elementary knowledge about infinite sequences; even here, the concept of limit is used in only a few applications, and these can be omitted.

Equation (6.1) is an example of a *recurrence relation*, a formula reducing later values of a sequence of numbers to earlier ones. Let us see how we can use the recurrence formula to get a general expression for t_k. We know that $t_1 = 1$. This is given to us, and is called an *initial condition*. We know that

$$\begin{aligned} t_2 &= 2t_1 \\ t_3 &= 2t_2 = 2^2 t_1 \\ t_4 &= 2t_3 = 2^2 t_2 = 2^3 t_1, \end{aligned}$$

and in general

$$t_k = 2t_{k-1} = \cdots = 2^{k-1} t_1.$$

Using the initial condition, we have

$$t_k = 2^{k-1} \tag{6.2}$$

for all k. We have solved the recurrence (6.1) by *iteration* or repeated use of the recurrence. Note that a recurrence like (6.1) will in general have many *solutions*, that is, sequences which satisfy it. However, once sufficiently many initial conditions are specified, there will be a unique solution. Here the sequence $1, 2, 4, 8, \ldots$ is the unique solution given the initial condition. However, if the initial condition is disregarded, any multiple of this sequence is a solution, as, for instance, $3, 6, 12, 24, \ldots$ or $5, 10, 20, 40, \ldots$.

We are really interested in s_k. We have

$$s_{k+1} = s_k + t_{k+1}, \tag{6.3}$$

another form of recurrence formula that relates later values of s to earlier values of s and to values of t already calculated. We can reduce (6.3) to a recurrence for s_k alone by using (6.2). This gives us

$$s_{k+1} = s_k + 2^k. \tag{6.4}$$

Let us again use iteration to solve the recurrence relation (6.4) for s_k for all k. We have

$$\begin{aligned} s_2 &= s_1 + 2 \\ s_3 &= s_2 + 2^2 = s_1 + 2 + 2^2, \end{aligned}$$

and in general

$$s_k = s_{k-1} + 2^{k-1} = \cdots = s_1 + 2 + 2^2 + \cdots + 2^{k-1}.$$

Since we have the initial condition $s_1 = 1$, we obtain

$$s_k = 1 + 2 + 2^2 + \cdots + 2^{k-1}.$$

This expression can be simplified if we use the following well-known identity, which we have already encountered in Chapter 5:

$$1 + x + x^2 + \cdots + x^p = \frac{1 - x^{p+1}}{1 - x}.$$

Using this identity with $x = 2$ and $p = k - 1$, we have

$$s_k = \frac{1 - 2^k}{1 - 2} = 2^k - 1.$$

Now there are 64 squares on a chess board. Hence, the number of grains of wheat the Vizier asked for is given by $2^{64} - 1$, which is

$$18{,}446{,}744{,}073{,}709{,}551{,}615,$$

a very large number indeed![2] ■

Example 6.2 Computational Complexity One major use of recurrences in computer science is in the computation of the complexity $f(n)$ of an algorithm with input of size n (see Section 2.4). Often, computation of the complexity $f(n+1)$ is reduced to knowledge of the complexities $f(n)$, $f(n-1)$, and so on. As a trivial example, let us consider the following algorithm for summing the first n entries of a sequence or an array A.

Algorithm 6.1: Summing the First n Entries of a Sequence or an Array

Input: A sequence or an array A and a number n.
Output: The sum $A(1) + A(2) + \cdots + A(n)$.

Step 1. Set $i = 1$.

Step 2. Set $T = A(1)$.

Step 3. If $i = n$, stop and output T. Otherwise, set $i = i + 1$ and go to step 4.

Step 4. Set $T = T + A(i)$ and return to step 3.

If $f(n)$ is the number of additions performed in summing the first n entries of A, we have the recurrence

$$f(n) = f(n-1) + 1. \tag{6.5}$$

Also, we have the initial condition $f(1) = 0$. Thus, by iteration, we have

$$f(n) = f(n-1) + 1 = f(n-2) + 1 + 1 = \cdots = f(1) + n - 1 = n - 1.$$ ■

Example 6.3 Simple and Compound Interest Suppose that a sum of money S_0 is deposited in a bank at *interest rate* r per interest period (say, per year), that is, at $100r$ percent. If the interest is *simple*, after every interest period a fraction r

[2] All of the sequences from this chapter can be found at the *On-Line Encyclopedia of Integer Sequences* (Sloane [2003]). This is a database of over 90,000 sequences. The entry for each sequence gives the beginning terms of the sequence, its name or description, references, formulas, and so on.

of the initial deposit S_0 is credited to the account. If S_k is the amount on deposit after k periods, we have the recurrence

$$S_{k+1} = S_k + rS_0. \tag{6.6}$$

By iteration, we find that

$$S_k = S_{k-1} + rS_0 = S_{k-2} + rS_0 + rS_0 = \cdots = S_0 + krS_0,$$

so

$$S_k = S_0(1 + kr).$$

If interest is *compounded* each period, we receive as interest after each period a fraction r of the amount on deposit at the beginning of the period; that is, we have the recurrence

$$S_{k+1} = S_k + rS_k,$$

or

$$S_{k+1} = (1 + r)S_k. \tag{6.7}$$

We find by iteration that

$$S_k = (1 + r)^k S_0.$$

■

Example 6.4 Legitimate Codewords Codewords from the alphabet $\{0, 1, 2, 3\}$ are to be recognized as *legitimate* if and only if they have an even number of 0's. How many legitimate codewords of length k are there? Let a_k be the answer. We derive a recurrence for a_k. (Note that a_k could be computed using the method of generating functions of Chapter 5.) Observe that $4^k - a_k$ is the number of illegitimate k-digit codewords, that is, the k-digit words with an odd number of 0's. Consider a legitimate $(k + 1)$-digit codeword. It starts with 1, 2, or 3, or it starts with 0. In the former case, the last k digits form a legitimate codeword of length k, and in the latter case they form an illegitimate codeword of length k. Thus, by the product and sum rules of Chapter 2,

$$a_{k+1} = 3a_k + 1(4^k - a_k),$$

that is,

$$a_{k+1} = 2a_k + 4^k. \tag{6.8}$$

We have the initial condition $a_1 = 3$. One way to solve the recurrence (6.8) is by the method of iteration. This is analogous to the solution of recurrence (6.4) and is left to the reader. An alternative method is described in Section 6.3. For now, we compute some values of a_k. Note that since $a_1 = 3$, the recurrence gives us

$$a_2 = 2a_1 + 4^1 = 2(3) + 4 = 10$$

and

$$a_3 = 2a_2 + 4^2 = 2(10) + 16 = 36.$$

The reader might wish to check these numbers by writing out the legitimate codewords of lengths 2 and 3. Note how early values of a_k are used to derive later values. We do not need an explicit solution to use a recurrence to calculate unknown numbers. ■

Example 6.5 Duration of Messages Imagine that we transmit messages over a channel using only two signals, a and b. A codeword is any sequence from the alphabet $\{a, b\}$. Now suppose that signal a takes 1 unit of time to transmit and signal b takes 2 units of time to transmit. Let N_t be the number of possible codewords that can be transmitted in exactly t units of time. What is N_t? To answer this question, consider a codeword transmittable in t units of time. It begins either in a or b. If it begins in a, the remainder is any codeword that can be transmitted in $t-1$ units of time. If it begins in b, the remainder is any codeword that can be transmitted in $t-2$ units of time. Thus, by the sum rule, for $t \geq 2$,

$$N_t = N_{t-1} + N_{t-2}. \tag{6.9}$$

This is our first example of a recurrence where a given value depends on more than one previous value. For this recurrence, since the tth term depends on two previous values, we need two initial values, N_1 and N_2. Clearly, $N_1 = 1$ and $N_2 = 2$, the latter since aa and b are the only two sequences that can be transmitted in 2 units of time. We shall solve the recurrence (6.9) in Section 6.2 after we develop some general tools for solving recurrences. Shannon [1956] defines the *capacity* C of the transmission channel as

$$C = \lim_{t \to \infty} \frac{\log_2 N_t}{t}.$$

This is a measure of the capacity of the channel to transmit information. We return to this in Section 6.2.2. ■

Example 6.6 Transposition Average of Permutations Given a permutation π of $\{1, 2, \ldots, n\}$, Jerrum's formula (2.17) calculates the number of transpositions needed to transform the identity permutation into π. If π is chosen at random, what is the expected number of needed transpositions? Or, put another way, what is the average number of transpositions needed to transform the identity permutation into a permutation π of $\{1, 2, \ldots, n\}$?

Suppose that each of the permutations of $\{1, 2, \ldots, n\}$ are equally likely. Let b_n equal the average number of transpositions needed to transform the identity permutation into a permutation of $\{1, 2, \ldots, n\}$. For example, if $n = 2$, there are two permutations of $\{1, 2\}$, namely, 12 and 21. No transpositions are needed to transform the identity into 12, and one transposition is needed to transform it into 21. The average number of transpositions needed is $b_2 = (0 + 1)/2 = 1/2$. To find b_{n+1}, consider any permutation $\pi = \pi_1\pi_2 \cdots \pi_n$ of $\{1, 2, \ldots, n\}$ and let π^i for $i = 1, 2, \ldots, n+1$ be $\pi_1\pi_2 \cdots \pi_{i-1}(n+1)\pi_i \cdots \pi_n$. [If $i = n+1$, π^i is $\pi_1\pi_2 \cdots \pi_n(n+1)$.] Let $a(\pi)$ be the number of transpositions needed to transform the identity into π and $a(\pi^i)$ be the number needed to transform the identity into

π^i. Note that in π^i, $n+1$ is greater than any of the numbers to its right. Therefore, by Jerrum's formula (2.17), $a(\pi^i) = a(\pi) + [n - (i-1)]$, so

$$\begin{aligned}\sum_{i=1}^{n+1} a(\pi^i) &= [(a(\pi)+n) + (a(\pi)+(n-1)) + \cdots + (a(\pi)+0)] \\ &= [(n+1)a(\pi) + n + (n-1) + \cdots + 0] \\ &= [(n+1)a(\pi) + n(n+1)/2],\end{aligned}$$

by the standard formula for the sum of an arithmetic progression. Then

$$\begin{aligned}b_{n+1} &= \frac{\sum_\pi \sum_{i=1}^{n+1} a(\pi^i)}{(n+1)!} \\ &= \frac{\sum_\pi [(n+1)a(\pi) + n(n+1)/2]}{(n+1)!} \\ &= \frac{\sum_\pi a(\pi)}{n!} + \frac{n!\,(n(n+1)/2)}{(n+1)!} \\ &= b_n + \frac{n}{2}.\end{aligned}$$

To solve this recurrence relation, we note that $b_1 = 0$, $b_2 = \frac{1}{2}$, and

$$\begin{aligned}b_3 &= b_2 + \tfrac{2}{2} = \tfrac{1}{2} + \tfrac{2}{2}, \\ b_4 &= b_3 + \tfrac{3}{2} = \tfrac{1}{2} + \tfrac{2}{2} + \tfrac{3}{2}, \\ b_5 &= b_4 + \tfrac{4}{2} = \tfrac{1}{2} + \tfrac{2}{2} + \tfrac{3}{2} + \tfrac{4}{2}.\end{aligned}$$

In general,

$$b_{n+1} = b_n + \frac{n}{2} = \cdots = \frac{1}{2} + \frac{2}{2} + \frac{3}{2} + \cdots + \frac{n}{2} = \frac{1}{2}(1+2+\cdots+n) = \frac{n(n+1)}{4},$$

again using the standard formula for the sum of an arithmetic progression. ■

Example 6.7 Regions in the Plane A line separates the plane into two regions (see Figure 6.1). Two intersecting lines separate the plane into four regions (again see Figure 6.1). Suppose that we have n lines in "general position"; that is, no two are parallel and no three lines intersect in the same point. Into how many regions do these lines divide the plane? To answer this question, let $f(n)$ be the appropriate number of regions. We have already seen that $f(1) = 2$ and $f(2) = 4$. Figure 6.1 also shows that $f(3) = 7$. To determine $f(n)$, we shall derive a recurrence relation.

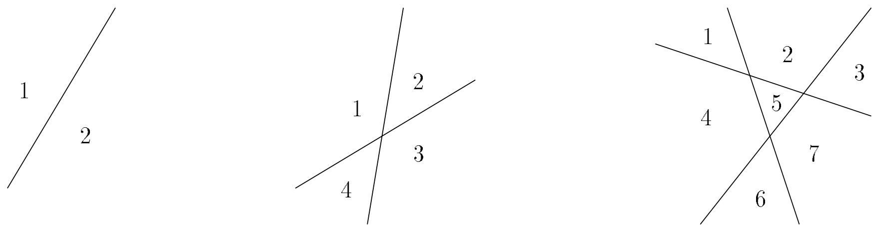


Figure 6.1: Lines dividing the plane into regions.

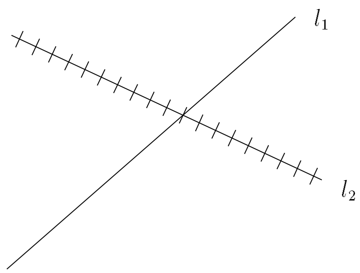


Figure 6.2: Line l_1 divides line l_2 into two segments.

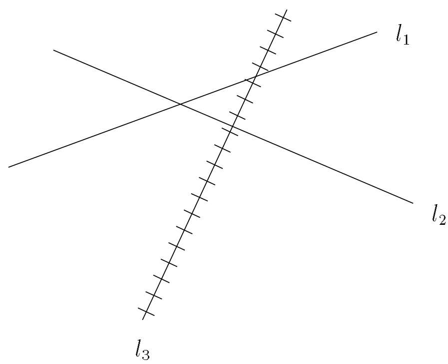


Figure 6.3: Line l_3 is divided by lines l_1 and l_2 into three segments.

Consider a line l_1 as shown in Figure 6.2. Draw a second line l_2. Line l_1 divides line l_2 into two segments, and each segment divides an existing region into two regions. Hence,

$$f(2) = f(1) + 2.$$

Similarly, if we add a third line l_3, this line is divided by l_1 and l_2 into three segments, with each segment splitting an existing region into two parts (see Figure 6.3). Hence,

$$f(3) = f(2) + 3.$$

In general, suppose that we add a line l_{n+1} to already existing lines $l_1, l_2, \ldots, l_n$. The existing lines split l_{n+1} into $n + 1$ segments, each of which splits an existing region into two parts (Figure 6.4). Hence, we have

$$f(n + 1) = f(n) + (n + 1). \tag{6.10}$$

Equation (6.10) gives a recurrence relation that we shall use to solve for $f(n)$. The initial condition is the value of $f(1)$, which is 2. To solve the recurrence relation

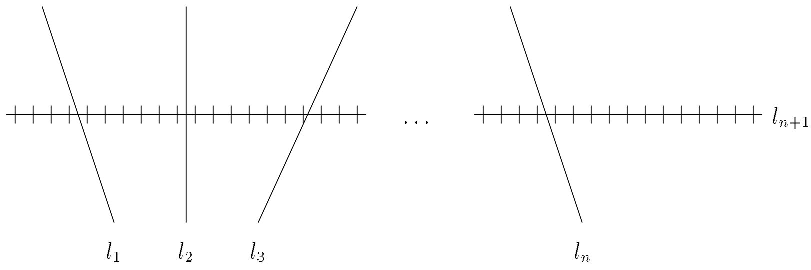


Figure 6.4: Line l_{n+1} is split by lines $l_1, l_2, \ldots, l_n$ into $n+1$ segments.

(6.10), we note that

$$\begin{aligned}
&f(2) = f(1) + 2, \\
&f(3) = f(2) + 3 = f(1) + 2 + 3, \\
&f(4) = f(3) + 4 = f(2) + 3 + 4 = f(1) + 2 + 3 + 4,
\end{aligned}$$

and in general,

$$f(n) = f(n-1) + n = \cdots = f(1) + 2 + 3 + \cdots + n.$$

Since $f(1) = 2$, we have

$$\begin{aligned}
f(n) &= 2 + 2 + 3 + 4 + \cdots + n \\
&= 1 + (1 + 2 + 3 + 4 + \cdots + n) \\
&= 1 + \frac{n(n+1)}{2},
\end{aligned}$$

again using the standard formula for the sum of an arithmetic progression. For example, we have

$$f(4) = 1 + \frac{4 \cdot 5}{2} = 11.$$

■

6.1.2 Fibonacci Numbers and Their Applications

In the year 1202, Leonardo of Pisa, better known as Fibonacci, posed the following problem in his book *Liber Abaci*. Suppose that we study the prolific breeding of rabbits. We start with one pair of adult rabbits (of opposite gender). Assume that each pair of adult rabbits produce one pair of young (of opposite gender) each month. A newborn pair of rabbits become adults in two months, at which time they also produce their first pair of young. Assume that rabbits never die. Let F_k count the number of rabbit pairs present at the beginning of the kth month. Table 6.1 lists for each of several values of k the number of adult pairs, the number of one-month-old young pairs, the number of newborn young pairs, and the total number of rabbit pairs. For example, at the beginning of the second month, there is one newborn rabbit pair. At the beginning of the third month, the newborns from

Table 6.1: Rabbit Breeding

Month k	Number of adult pairs at beginning of month k	Number of one-month-old pairs at beginning of month k	Number of newborn pairs at beginning of month k	Total number of pairs at beginning of month $k = F_k$
1	1	0	0	1
2	1	0	1	2
3	1	1	1	3
4	2	1	2	5
5	3	2	3	8
6	5	3	5	13

the preceding month are one month old, and there is again a newborn pair. At the beginning of the fourth month, the one-month-olds have become adults and given birth to newborns, so there are now two adult pairs and two newborn pairs. The one newborn pair from month 3 has become a one-month-old pair. And so on.

Let us derive a recurrence relation for F_k. Note that the number of rabbit pairs at the beginning of the kth month is given by the number of rabbit pairs at the beginning of the $(k-1)$st month plus the number of newborn pairs at the beginning of the kth month. But the latter number is the same as the number of adult pairs at the beginning of the kth month, which is the same as the number of all rabbit pairs at the beginning of the $(k-2)$th month. (It takes exactly 2 months to become an adult.) Hence, we have for $k \geq 3$,

$$F_k = F_{k-1} + F_{k-2}. \tag{6.11}$$

Note that if we define F_0 to be 1, then (6.11) holds for $k \geq 2$. Observe the similarity of recurrences (6.11) and (6.9). We return to this point in Section 6.2.2. Let us compute several values of F_k using the recurrence (6.11). We already know that

$$F_0 = F_1 = 1.$$

Hence,

$$\begin{aligned}
F_2 &= F_1 + F_0 = 2,\\
F_3 &= F_2 + F_1 = 3,\\
F_4 &= F_3 + F_2 = 5,\\
F_5 &= F_4 + F_3 = 8,\\
F_6 &= F_5 + F_4 = 13,\\
F_7 &= F_6 + F_5 = 21,\\
F_8 &= F_7 + F_6 = 34.
\end{aligned}$$

In Section 6.2.2 we shall use the recurrence (6.11) to obtain an explicit formula for the number F_k. The sequence of numbers $F_0, F_1, F_2, \ldots$ is called the *Fibonacci* *sequence* and the numbers F_k the *Fibonacci numbers.* These numbers have remarkable properties and arise in a great variety of places. We shall describe some of their properties and applications here.

The *growth rate* at time k of the sequence (F_k) is defined to be

$$G_k = \frac{F_k}{F_{k-1}}.$$

Then we have

$$G_1 = \frac{1}{1} = 1,\ G_2 = \frac{2}{1} = 2,\ G_3 = \frac{3}{2} = 1.5,\ G_4 = \frac{5}{3} = 1.67,\ G_5 = \frac{8}{5} = 1.60,$$

$$G_6 = \frac{13}{8} = 1.625,\ G_7 = \frac{21}{13} = 1.615,\ G_8 = \frac{34}{21} = 1.619,\ \ldots.$$

The numbers G_k seem to be converging to a limit between 1.60 and 1.62. In fact, this limit turns out to be exactly

$$\tau = \frac{1}{2}(1+\sqrt{5}) = 1.618034\ldots.$$

The number τ will arise in the development of a general formula for F_k. [This number is called the *golden ratio* or the *divine proportion.* It is the number with the property that if one divides the line AB at C so that $\tau = AB/AC$, then

$$\frac{AB}{AC} = \frac{AC}{CB}.$$

The rectangle with sides in the ratio $\tau : 1$ is called the *golden rectangle.* A fifteenth-century artist Piero della Francesca wrote a whole book (*De Divina Proportione*) about the applications of τ and the golden rectangle in art, in particular in the work of Leonardo da Vinci. Although much has been made of the golden ratio in the arts, architecture, and aesthetics, it has been argued that many of the golden ratio assertions are either "false or seriously misleading" (Markowsky [1992]).]

Fibonacci numbers have important applications in numerical analysis, in particular in the search for the maximum value of a function $f(x)$ in an interval (a, b). A *Fibonacci search* for the maximum value, performed on a computer, makes use of the Fibonacci numbers to determine where to evaluate the function in getting better and better estimates of the location of the maximum value. When f is concave, this is known to be the best possible search procedure in the sense of minimizing the number of function evaluations for finding the maximum to a desired degree of accuracy. See Adby and Dempster [1974], Hollingdale [1978], or Kiefer [1953].

It is intriguing that Fibonacci numbers appear very frequently in nature. The field of botany that studies the arrangements of leaves around stems, the scales on cones, and so on, is called *phyllotaxis.* Usually, leaves appearing on a given stem or branch point out in different directions. The second leaf is rotated from the first by a certain angle, the third leaf from the second by the same angle, and so on, until some leaf points in the same direction as the first. For example, if the angle of

Table 6.2: Values of n and m for Various Plants

Plant	Angle of rotation	n	m
Elm	180°	2	1
Alder, birch	120°	3	1
Rose	144°	5	2
Cabbage	135°	8	3

Source: Schips [1922]; Batschelet [1971].

rotation is 30°, then the twelfth leaf is the first one pointing in the same direction as the first, since $12 \times 30° = 360°$. If the angle is 144°, then the fifth leaf is the first one pointing in the same direction as the first, for $5 \times 144° = 720°$. Two complete 360° returns have been made before a leaf faces in the same direction as the first. In general, let n count the number of leaves before returning to the same direction as the first, and let m count the number of complete 360° turns that have been made before this leaf is encountered. Table 6.2 shows the values of n and m for various plants. It is a remarkable empirical fact of biology that most frequently both n and m take as values the Fibonacci numbers. There is no good theoretical explanation for this fact.

Coxeter [1969] points out that the Fibonacci numbers also arise in the study of scales on a fir cone, whorls on a pineapple, and so on. These whorls (cells) are arranged in fairly visible diagonal rows. The whorls can be assigned integers in such a way that each diagonal row of whorls forms an arithmetic sequence with common difference (difference between successive numbers) a Fibonacci number. This is shown in Figure 6.5. Note, for example, the diagonal

$$9, 22, 35, 48, 61, 74,$$

which has common difference the Fibonacci number 13. Similarly, the diagonal

$$11, 19, 27, 35, 43, 51, 59$$

has common difference 8. Similar remarks hold for the florets of sunflowers, the scales of fir cones, and so on (see Adler [1977] and Fowler, Prusinkiewicz,and Battjes [1992]). Again, no explanation for why Fibonacci numbers arise in this aspect of phyllotaxis is known.

Example 6.8 "Hot Hand" (Example 2.32 Revisited) Recall the basketball phenomenon known as "hot hand." A hot hand assumes that once a player makes a shot, he or she has a higher-than-average chance of making the next shot. In Example 2.32 we considered the probability of all the made shots occurring consecutively. Here, we will consider the number of ways of making *no* consecutive shots.

If a player shoots n shots, there are 2^n possible outcomes; each shot can be either made or missed.[3] Let B_n denote the subset of those orderings of X's and O's that

[3] It does not follow that all 2^n orderings are equally likely. Made shots and missed shots are rarely equally likely.

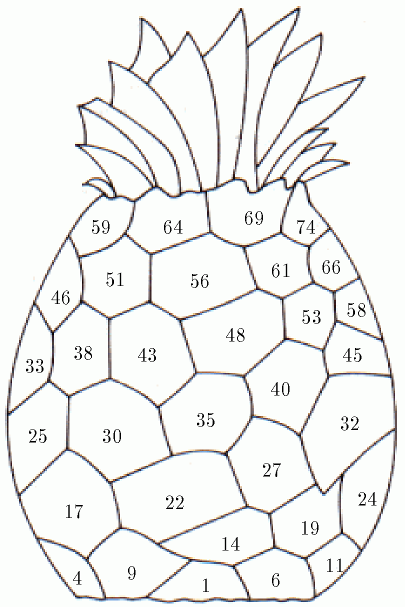


Figure 6.5: Fibonacci numbers and pineapples. (From Coxeter [1969]. Reprinted with permission of John Wiley & Sons, Inc.)

contain no consecutive made shots, i.e., no two consecutive X's. To find $b_n = |B_n|$, we refer to made shot i as c_i and let $C_n = \{c_1, c_2, \ldots, c_n\}$. Put another way, our question asks: How many subsets of C_n contain no c_i, c_{i+1}, $i = 1, 2, \ldots, n-1$? Let B be one such subset. Either B contains c_n or it does not. If it does, then $B - \{c_n\}$ is a subset of C_{n-2} since c_{n-1} is certainly not in B. If B does not contain c_n, then $B - \{c_n\} = B$ is a subset of C_{n-1}. Therefore, for $n \geq 2$,

$$b_n = b_{n-1} + b_{n-2}. \tag{6.12}$$

For $n = 1$, $b_1 = 2$; either the one shot is made or missed. If $n = 2$, then either no shot is made, the first shot is made and the second missed, or vice versa. Thus, $b_3 = 3$. Using these initial conditions, we see that the solution of recurrence (6.12) is closely related to the Fibonacci numbers. In particular,

$$b_n = F_{n+1}, \ n \geq 1. \qquad \blacksquare$$

Thus, if 10 shots are attempted (as in Example 2.32), only $b_{10} = F_{11} = 144$ out of $2^{10} = 1024$ outcomes contain no consecutive made shots.

6.1.3 Derangements

Example 6.9 The Hatcheck Problem Imagine that n gentlemen attend a party and check their hats. The checker has a little too much to drink, and returns the hats at random. What is the probability that no gentleman receives his own hat? How does this probability depend on the number of gentlemen? We shall answer these questions by studying the notion of a derangement. See Takács [1980] for the origin, the history, and alternative formulations of this problem. ■

Let n objects be labeled $1, 2, \ldots, n$. An arrangement or permutation in which object i is not placed in the ith place for any i is called a *derangement*. For example, if n is 3, then 231 is a derangement, but 213 is not since 3 is in the third place. Let D_n be the number of derangements of n objects.

Derangements arise in a card game (*rencontres*) where a deck of cards is laid out in a row on the table and a second deck is laid out randomly, one card on top of each of the cards of the first deck. The number of matching cards determines a score. In 1708, the Frenchman P. R. Montmort posed the problem of calculating the probability that no matches will take place and called it "le problème des rencontres," rencontre meaning "match" in French. This is, of course, the problem of calculating D_{52}. The problem of computing the number of matching cards will be taken up in Section 7.2. There we also discuss applications to the analysis of guessing abilities in psychic experiments. The first deck of cards corresponds to an unknown order of things selected or sampled, and the second to the order predicted by a psychic. In testing claims of psychic powers, one would like to compute the probability of getting matches right. The probability of getting at least one match right is 1 minus the probability of getting no matches right, that is, 1 minus the probability of getting a derangement.

Clearly, $D_1 = 0$: There is no arrangement of one element in which the element does not appear in its proper place. $D_2 = 1$: The only derangement is 21. We shall derive a recurrence relation for D_n. Suppose that there are $n+1$ elements, $1, 2, \ldots, n+1$. A derangement of these $n+1$ elements involves a choice of the first element and then an ordering of the remaining n. The first element can be any of n different elements: $2, 3, \ldots, n+1$. Suppose that k is put first. Then either 1 appears in the kth spot or it does not. If 1 appears in the kth spot, we have left the elements

$$2, 3, \ldots, k-1, k+1, \ldots, n+1,$$

and we wish to order them so that none appears in its proper location [see Figure 6.6(a)]. There are D_{n-1} ways to do this, since there are $n-1$ elements. Suppose next that 1 does not appear in the kth spot. We can think of first putting 1 into the kth spot [as shown in Figure 6.6(b)] and then deranging the elements in the second through $(n+1)$st spots. There are D_n such derangements. In sum, we have n choices for the element k which appears in the first spot. For each of these, we either choose an arrangement with 1 in the kth spot—which we can do in D_{n-1} ways—or we choose an arrangement with 1 not in the kth spot—which we can do in D_n ways. It follows by the product and sum rules that

$$D_{n+1} = n(D_{n-1} + D_n). \tag{6.13}$$

Equation (6.13) is a recurrence relation that makes sense for $n \geq 2$. If we add the

$$(a)\quad \begin{array}{cccc} k & & 1 & \\ \cdot & \cdots & \cdot & \cdots \\ \text{1st} & & k\text{th} & \end{array}$$

$$(b)\quad \begin{array}{cccccccccc} k & 2 & 3 & & k-1 & 1 & k+1 & & n+1 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot & \cdot & \cdots & \cdot \\ \text{1st} & \text{2nd} & \text{3rd} & & (k-1)\text{st} & k\text{th} & (k+1)\text{st} & & (n+1)\text{st} \end{array}$$

Figure 6.6: Derangements with k in the first spot and 1 in the kth spot and other elements (a) in arbitrary order and (b) in the proper spots.

initial conditions $D_1 = 0, D_2 = 1$, it turns out that

$$D_n = n!\left[1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!}\right] \tag{6.14}$$

or (for $n \geq 2$),

$$D_n = n!\left[\frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!}\right]. \tag{6.15}$$

We shall see in Section 6.3.2 how to derive these formulas.

Let us now apply these formulas to the hatcheck problem of Example 6.9. Let p_n be the probability that no gentleman gets his hat back if there are n gentlemen. Then (for $n \geq 2$) we have

$$\begin{aligned} p_n &= \frac{\text{number of arrangements with no one receiving his own hat}}{\text{number of arrangements}} \\ &= \frac{D_n}{n!} \\ &= \left[\frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!}\right]. \end{aligned}$$

Table 6.3 shows values of p_n for several n. Note that p_n seems to be converging rapidly. In fact, we can calculate exactly what p_n converges to. Recall that

$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots$$

[see (5.3)]. Hence,

$$e^{-1} = \frac{1}{2!} - \frac{1}{3!} + \cdots,$$

so p_n converges to $e^{-1} = 0.367879441\ldots$. The convergence is so rapid that p_7 and p_8 already differ only in the fifth decimal place. The probability that no gentleman receives his hat back very rapidly becomes essentially independent of the number of gentlemen.

Table 6.3: Values of p_n

n	2	3	4	5	6	7	8
p_n	.500000	.333333	.375000	.366667	.368056	.367858	.367883

Example 6.10 Latin Rectangles In Chapter 1 we talked about Latin squares and their applications to experimental design, and in Exercise 15, Section 5.1, we introduced Latin rectangles and the idea of building up Latin squares one row at a time. Let $L(r,n)$ be the number of $r \times n$ Latin rectangles with entries $1, 2, \ldots, n$. (Recall that such a rectangle is an $r \times n$ array with entries $1, 2, \ldots, n$ so that no two entries in any row or column are the same.) Let $K(r,n)$ be the number of $r \times n$ Latin rectangles with entries $1, 2, \ldots, n$ and first row in the *standard position* $123 \cdots n$. Then

$$L(r,n) = n!K(r,n), \tag{6.16}$$

for one may obtain any Latin rectangle by finding one with a standard first row and then permuting the first row and performing the same permutation on the elements in all remaining rows.

We would like to calculate $L(r,n)$ or $K(r,n)$ for every r and n. By virtue of (6.16), these problems are equivalent. In Example 1.1, we asked for $L(5,5)$ (which is 161,280; see Ryser [1963]). $K(2,n)$ is easy to calculate. It is simply D_n, the number of derangements of n elements. For we obtain the second row of a Latin rectangle with the first row in the standard position by deranging the elements of the first row.

Two formulas for $L(r,n)$ based on a function of certain matrices are given by Fu [1992]. (Shao and Wei [1992] gave an explicit formula for $L(n,n)$ based on an idea of MacMahon [1898].) ■

Example 6.11 DNA Sequence Alignment As noted in Section 2.17, mutations are a key process by which evolution takes place. Given DNA sequences from two different species, we sometimes try to see how close they are, and in particular look for patterns that appear in both. We often do this by aligning the two sequences, one under the other, so that a subsequence of each is (almost) the same. For instance, consider the sequences AATAATGAC and GAGTAATCGGAT. (Note that these have different lengths.) One alignment is the following:

$$\begin{array}{cccccccccccc} & A & A & T & A & A & T & G & A & C & & \\ G & A & G & T & A & A & T & C & G & G & A & T \end{array} \tag{6.17}$$

Here, we note a common subsequence TAAT. (In practical alignment applications, we often allow insertion and deletion of letters, but we will disregard that here.) Searching for good sequence alignments, ones where there are long common subsequences or patterns, has led to very important biological insights. For example, it was discovered that the sequence for platelet derived factor, which causes growth in the body, is 87 percent identical to the sequence for *v-sis*, a cancer-causing gene. This led to the discovery that *v-sis* works by stimulating growth. Indeed, as Gusfield [1997] points out, in DNA sequences (and, more generally, in other biomolecular sequences such as RNA or amino acid sequences), "high sequence similarity usually implies significant functional or structural similarity."

One way to measure how good an alignment of two sequences is is to count the number of positions where they don't match. This is called the *mismatch distance*. For instance, in alignment (6.17), the mismatch distance is 4 if we disregard places where the top sequence has no entries and 7 if we count places where one sequence has no entries. For more on sequence alignment in molecular biology, see Apostolico and Giancarlo [1999], Gusfield [1997], Myers [1995], Setubal and Meidanis [1997], or Waterman [1995].

Here we consider a simplified version of the sequence alignment problem. Suppose that we are given a DNA sequence of length n and it evolves by a random "permutation" of the elements into another such sequence. What is the probability that the mismatch distance between the two sequences is n? If the first sequence is AGCT, this is the probability that the new sequence is a derangement of the first, i.e., $D_4/4! = 3/8$. However, if the original sequence is AGCC, in which there are repeated entries, the problem is different. First, there are now only $4!/2! = 12$ possible sequences with these bases, not 4!. (Why?) Second, the only permutations of AGCC in which no element is in the same position as in AGCC are CCAG and CCGA. Thus, the desired probability is $\dfrac{2}{4!/2!} = \dfrac{1}{6}$, not $\dfrac{D_4}{4!} = \dfrac{3}{8}$. ■

6.1.4 Recurrences Involving More than One Sequence

Generalizing Example 6.4, let us define a codeword from the alphabet $\{0, 1, 2, 3\}$ to be *legitimate* if and only if it has an even number of 0's *and* an even number of 3's. Let a_k be the number of legitimate codewords of length k. How do we find a_k? To answer this question, it turns out to be useful, in a manner analogous to the situation of Example 6.4, to consider other possibilities for a word of length k. Let b_k be the number of k-digit words from the alphabet $\{0, 1, 2, 3\}$ with an even number of 0's and an odd number of 3's, c_k the number with an odd number of 0's and an even number of 3's, and d_k the number with an odd number of 0's and an odd number of 3's. Note that

$$d_k = 4^k - a_k - b_k - c_k. \tag{6.18}$$

Observe that we get a legitimate codeword of length $k+1$ by preceding a legitimate codeword of length k by a 1 or a 2, by preceding a word of length k with an even number of 0's and an odd number of 3's with a 3, or by preceding a word of length k with an odd number of 0's and an even number of 3's with a 0. Hence,

$$a_{k+1} = 2a_k + b_k + c_k. \tag{6.19}$$

Similarly,

$$b_{k+1} = a_k + 2b_k + d_k, \tag{6.20}$$

or, using (6.18),

$$b_{k+1} = b_k - c_k + 4^k. \tag{6.21}$$

Finally,

$$c_{k+1} = a_k + 2c_k + d_k, \tag{6.22}$$

or

$$c_{k+1} = c_k - b_k + 4^k. \tag{6.23}$$

Equations (6.19), (6.21), and (6.23) can be used together to compute any desired value a_k. We start with the initial conditions $a_1 = 2, b_1 = 1, c_1 = 1$. From (6.19), (6.21), (6.23), we compute

$$\begin{aligned} a_2 &= 2 \cdot 2 + 1 + 1 = 6, \\ b_2 &= 1 - 1 + 4^1 = 4, \\ c_2 &= 1 - 1 + 4^1 = 4. \end{aligned}$$

These results are easy to check by listing the corresponding sequences. For instance, the six sequences from $\{0, 1, 2, 3\}$ of length 2 and having an even number of 0's and 3's are $00, 33, 11, 12, 21, 22$. Similarly, one obtains

$$\begin{aligned} a_3 &= 2 \cdot 6 + 4 + 4 = 20, \\ b_3 &= 4 - 4 + 4^2 = 16, \\ c_3 &= 4 - 4 + 4^2 = 16. \end{aligned}$$

Notice that we have not found a single recurrence relation. However, we have found three relations that may be used simultaneously to compute the desired numbers.

EXERCISES FOR SECTION 6.1

1. Suppose that $a_n = 4a_{n-1} + 3, n \geq 1$, and $a_1 = 5$. Derive a_5 and a_6.
2. In Example 6.3 with $r = .2$, $S_0 = \$5000$, and simple interest, use the recurrence successively to compute S_1, S_2, S_3, S_4, S_5, and S_6, and check your answer by using the equation $S_k = S_0(1 + kr)$.
3. Repeat Exercise 2 for compound interest and use the equation $S_k = (1 + r)^k S_0$ to check your answer.
4. In Example 6.4:
   (a) Derive a_4.
   (b) Derive a_5.
   (c) Verify that $a_2 = 10$ by listing all legitimate codewords of length 2.
   (d) Repeat part (c) for $a_3 = 36$.
5. Check the formula in Example 6.6 for b_3 by explicitly listing all of the permutations of $\{1, 2, 3\}$ and the number of permutations needed to transform the identity permutation into each.
6. In Example 6.7, verify that $f(4) = 11$ by drawing four lines in the plane and labeling the regions.

7. Find a solution to the recurrence (6.5) under the initial condition $f(1) = 15$.
8. Find two different solutions to the recurrence (6.6).
9. In Example 6.8, calculate b_4 and b_5, and verify your answers by enumerating the ways to shoot 4 and 5 shots with none made consecutively.
10. Find a solution to the recurrence (6.13) different from the sequence defined by (6.14).
11. Find all derangements of $\{1, 2, 3, 4\}$.
12. In the example of Section 6.1.4, use the recurrences (6.19), (6.21), and (6.23) to compute a_4, b_4, and c_4.
13. On the first day, n jobs are to be assigned to n workers. On the second day, the jobs are again to be assigned, but no worker is to get the same job that he or she had on the first day. In how many ways can the jobs be assigned for the two days?
14. In a computer system overhaul, a bank employee mistakenly deleted records of seven "pin numbers" belonging to seven accounts. After recreating the records, he assigned those pins to the accounts at random. In how many ways could he do this so that:

    (a) At least one pin gets properly assigned?

    (b) All seven pins get properly assigned?

15. A lab director can run two different kinds of experiments, the expensive one (E) costing \$8000 and the inexpensive one (I) costing \$4000. If, for example, she has a budget of \$12,000, she can perform experiments in the following sequences: III, IE, or EI. Let $F(n)$ be the number of sequences of experiments she can run spending exactly \$$n$. Thus, $F(12,000) = 3$.

    (a) Find a recurrence for $F(n)$.

    (b) Suppose that there are p different kinds of experiments, $E_1, E_2, \ldots, E_p$, with E_i costing d_i dollars to run. Find a recurrence for $F(n)$ in this case.

16. Suppose that we have 10¢ stamps, 18¢ stamps, and 28¢ stamps, each in unlimited supply. Let $f(n)$ be the number of ways of obtaining n cents of postage if the order in which we put on stamps counts. For example, $f(10) = 1$ and $f(20) = 1$ (two 10¢ stamps), while $f(28) = 3$ (one 28¢ stamp, or a 10¢ stamp followed by an 18¢ stamp, or an 18¢ stamp followed by a 10¢ stamp).

    (a) If $n > 29$, derive a recurrence for $f(n)$.

    (b) *Use the recurrence* of part (a) to find the number of ways of obtaining 66¢ of postage.

    (c) Check your answer to part (b) by writing down all the ways.

17. An industrial plant has two sections. Suppose that in one week, nine workers are assigned to nine different jobs in the first section and another nine workers are assigned to nine different jobs in the second section. In the next week, the supervisor would like to reassign jobs so that no worker gets back his or her old job. (No two jobs in the plant are considered the same.)

    (a) In how many ways can this be done if each worker stays in the same section of the plant?

(b) In how many ways can this be done if each worker is shifted to a section of the plant different from the one in which he or she previously worked?

18. In predicting future sales of a product, one (probably incorrect) assumption is to say that the amount sold next year will be the average of the amount sold this year and last year. Suppose that a_n is the amount sold in year n.

    (a) Find a recurrence for a_n. (b) Solve the recurrence if $a_0 = a_1 = 1$.

19. Find the number of derangements of $\{1, 2, 3, 4, 5, 6, 7, 8\}$ in which the first four elements are mapped into:

    (a) 1, 2, 3, 4 in some order (b) 5, 6, 7, 8 in some order

20. Suppose that $f(p)$ is the number of comparisons required to sort p items using the algorithm bubble sort (Section 3.6.4). Find a recurrence for $f(p)$ and solve.

21. A codeword from the alphabet $\{0, 1, 2\}$ is considered legitimate if and only if no two 0's appear consecutively. Find a recurrence for the number b_n of legitimate codewords of length n.

22. A codeword from the alphabet $\{0, 1, 2\}$ is considered legitimate if and only if there is an even number of 0's and an odd number of 1's. Find simultaneous recurrences from which it is possible to compute the number of legitimate codewords of length n.

23. (a) How many permutations of the integers $1, 2, \ldots, 9$ put each even integer into its natural position and no odd integer into its natural position?

    (b) How many permutations of the integers $1, 2, \ldots, 9$ have exactly four numbers in their natural position?

24. In a singles tennis tournament, $2n$ players are paired off in n matches, and $f(n)$ is the number of different ways in which this pairing can be done. Determine a recurrence for $f(n)$.

25. Suppose that a random permutation of the following DNA sequences occurs. What is the probability that the mismatch distance between the original and permuted sequences is n?

    (a) AGT (b) AACC (c) ACTCGC
    (d) ACTGGGG (e) CTTAAA (f) CTAGG

26. Suppose that

$$a_n = \begin{cases} a_{n-1} & n \text{ even} \\ 2a_{n-2} + a_{n-4} + \cdots + a_3 + a_1 + 1 & n \text{ odd.} \end{cases}$$

If $a_1 = 1$ and $a_3 = 3$, prove by induction that $a_{2n} = a_{2n-1} = F_{2n-1}$, i.e., $a_n =$ the Fibonacci number $F_{2\lfloor \frac{n+1}{2} \rfloor - 1}$.

27. (a) Suppose that chairs are arranged in a circle. Let L_n count the number of subsets of n chairs which don't contain consecutive chairs. Show that

$$L_{n+1} = F_n + F_{n+2}.$$

(The numbers L_n are called *Lucas numbers.*[4])

[4] Édouard Lucas (1842–1891), was a French mathematician who attached Fibonacci's name to the sequence solution to Fibonacci's rabbit problem posed in *Liber Abaci.*

(b) Determine two initial conditions for L_n, namely, L_1 and L_2.

(c) Prove that $L_n = L_{n-1} + L_{n-2}$.

28. (a) Prove that the number of ways to write n as the sum of 1's and 2's is equal to the Fibonacci number F_n.

(b) How many different ways can you put \$1.00 into a vending machine using only nickels and dimes?

29. In our notation, F_{n-1} is the nth Fibonacci number since we start with $F_0 = 1$.

(a) Prove that every third Fibonacci number is divisible by $F_2 = 2$.

(b) Prove that every fourth Fibonacci number is divisible by $F_3 = 3$.

(c) Prove that every fifth Fibonacci number is divisible by $F_4 = 5$.

(d) Prove that every nth Fibonacci number is divisible by F_{n-1}.

30. Consider the Fibonacci numbers F_n, $n = 0, 1, \ldots$.

(a) Prove that $F_{n-1}F_n - F_{n-2}F_{n+1} = (-1)^n$.

(b) Prove that $F_{n+1}F_{n-1} - F_n^2 = (-1)^n$.

(c) Prove that $F_n = 2 + \sum_{k=1}^{n-2} F_k$.

(d) Prove that $F_{2n+1} = 1 + \sum_{k=1}^{n} F_{2k}$.

(e) Prove that $F_{n+1} = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n-k}{k}$.

(f) Prove that $F_{n+m} = F_m F_{n+1} + F_{m-1} F_n$.

(g) Prove that $F_{2n+1} = F_n^2 + F_{n+1}^2$.

31. If F_n is the nth Fibonacci number, find a simple, closed-form expression for

$$F_1 + F_2 + \cdots + F_n$$

which involves F_p for only one p.

32. Let $S(n,t)$ denote the number of ways to partition an n-element set into t nonempty, unordered subsets.[5] Then $S(n,t)$ satisfies

$$S(n,t) = tS(n-1,t) + S(n-1,t-1), \tag{6.24}$$

for $t = 1, 2, \ldots, n-1$. Equation (6.24) is an example of a recurrence involving two indices. We could use it to solve for any $S(n,t)$. For instance, suppose that we start with the observation that $S(n,1) = 1$, all n, and $S(n,n) = 1$, all n. Then we can compute $S(n,t)$ for all remaining n and $t \le n$. For $S(3,2)$ can be computed from $S(2,2)$ and $S(2,1)$; $S(4,2)$ from $S(3,2)$ and $S(3,1)$; $S(4,3)$ from $S(3,3)$ and $S(3,2)$; and so on.

[5] $S(n,t)$ is called a Stirling number of the second kind and was discussed in Sections 2.10.4 and 5.5.3.

(a) Compute $S(5,3)$ by using (6.24). (b) Compute $S(6,3)$.

(c) Show that (6.24) holds.

33. In Exercise 24 of the Additional Exercises for Chapter 2, consider a grid with m north-south streets and n east-west streets. Let $s(m,n)$ be the number of different routes from point A to point B if A and B are located as in Figure 2.7. Find a recurrence for $s(m+1,n+1)$.

34. Determine a recurrence relation for $f(n)$, the number of regions into which the plane is divided by n circles each pair of which intersect in exactly two points and no three of which meet in a single point.

35. (a) Suppose that $f(n+1) = f(n)f(n-1)$, all $n \geq 1$, and $f(0) = f(1) = 2$. Find $f(n)$.

    (b) Repeat part (a) if $f(0) = f(1) = 1$.

36. In Example 6.6 we calculated b_n, the average number of transpositions needed to transform the identity permutation into a randomly chosen permutation of $\{1, 2, \ldots, n\}$. This was under the assumption that each permutation of $\{1, 2, \ldots, n\}$ is equally likely to be chosen. Calculate b_n if the identity permutation is twice as likely to be chosen over any other permutation and all other permutations are equally likely to be chosen.

37. (Liu [1968]) A *pattern* in a bit string consists of a number of consecutive digits, for example, 011. A pattern is said to *occur* at the kth digit of a bit string if when scanning the string from left to right, the full pattern appears after the kth digit has been scanned. Once a pattern occurs, that is, is observed, scanning begins again. For example, in the bit string 110101010101, the pattern 010 occurs at the fifth and ninth digits, but not at the seventh digit. Let b_n denote the number of n-digit bit strings with the pattern 010 occurring at the nth digit. Find a recurrence for b_n. (*Hint:* Consider the number of bit strings of length n ending in 010, and divide these into those where the 010 pattern occurs at the nth digit and those where it does not.)

38. Suppose that B is an $n \times n$ board and $r_n(B)$ is the coefficient of x^n in the rook polynomial $R(x,B)$. Use recurrence relations to compute $r_n(B)$ if

    (a) B has all squares darkened;

    (b) B has only the main diagonal lightened.

39. In the example of Section 6.1.4, find a single recurrence for a_k in terms of earlier values of a_p only.

40. Suppose that C_k is the number of connected, labeled graphs of k vertices. Harary and Palmer [1973] derive the recurrence

$$C_k = 2^{\binom{k}{2}} - \frac{1}{k}\sum_{p=1}^{k-1}\binom{k}{p}C_p p 2^{\binom{k-p}{2}}.$$

    Using the fact that $C_1 = 1$, compute C_2, C_3, and C_4, and check your answers by drawing the graphs.

41. A sequence of p 0's, q 1's, and r 2's is considered "good" if there are no consecutive digits in the sequence which are the same. Let $N(p,q,r)$ be the number of such "good" sequences.

(a) Calculate $N(p, q, 0)$.

(b) How many distinct sequences of p 0's, q 1's, and r 2's are possible if no restrictions are imposed?

(c) Find a recurrence relation for $N(p, q, r)$.

6.2 THE METHOD OF CHARACTERISTIC ROOTS

6.2.1 The Case of Distinct Roots

So far we have derived a number of interesting recurrence relations. Several of these we were able to solve by iterating back to the original values or initial conditions. Indeed, we could do something similar even with some of the more difficult recurrences we have encountered. There are no general methods for solving all recurrences. However, there are methods that work for a very broad class of recurrences. In this section we investigate one such method, the method of characteristic roots. In Section 6.3 we show how to make use of the notion of generating function, developed in Chapter 5, for solving recurrences. The methods for solving recurrences were developed originally in the theory of difference equations. Careful treatments of difference equations can be found in Elaydi [1999], Goldberg [1958], or Kelley and Peterson [2001].

Consider the recurrence

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_p a_{n-p}, \tag{6.25}$$

$n \geq p$, where $c_1, c_2, \ldots, c_p$ are constants and $c_p \neq 0$. Such a recurrence is called *linear* because all terms a_k occur to the first power and it is called *homogeneous*[6] because there is no term on the right-hand side that does not involve some a_k, $n-p \leq k \leq n-1$. Since the coefficients c_i are constants, the recurrence (6.25) is called a *linear homogeneous recurrence relation with constant coefficients.* The recurrences (6.1), (6.7), (6.9), and (6.11) are examples of such recurrences. The recurrences (6.4), (6.5), (6.6), (6.8), and (6.10) are not homogeneous and the recurrence (6.13) does not have constant coefficients. All of the recurrences we have encountered so far are linear.

We shall present a technique for solving linear homogeneous recurrence relations with constant coefficients; it is very similar to that used to solve linear differential equations with constant coefficients, as the reader who is familiar with the latter technique will see.

A recurrence (6.25) has a unique solution once we specify the values of the first p terms, $a_0, a_1, \ldots, a_{p-1}$; these values form the *initial conditions.* From a_0, a_1, $\ldots, a_{p-1}$, we can use the recurrence to find a_p. Then from $a_1, a_2, \ldots, a_p$, we can use the recurrence to find a_{p+1}, and so on.

In general, a recurrence (6.25) has many solutions, if the initial conditions are disregarded. Some of these solutions will be sequences of the form

$$\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^n, \ldots, \tag{6.26}$$

[6] We have previously defined homogeneous but in the context of polynomials.

where α is a number. We begin by finding values of α for which (6.26) is a solution to the recurrence (6.25). In (6.25), let us substitute x^k for a_k and solve for x. Making this substitution, we get the equation

$$x^n - c_1 x^{n-1} - c_2 x^{n-2} - \cdots - c_p x^{n-p} = 0. \tag{6.27}$$

Dividing both sides of (6.27) by x^{n-p}, we obtain

$$x^p - c_1 x^{p-1} - c_2 x^{p-2} - \cdots - c_p = 0. \tag{6.28}$$

Equation (6.28) is called the *characteristic equation* of the recurrence (6.25). It is a polynomial in x of power p, so has p roots $\alpha_1, \alpha_2, \ldots, \alpha_p$. Some of these may be repeated roots and some may be complex numbers. These roots are called the *characteristic roots* of the recurrence (6.25). For instance, consider the recurrence

$$a_n = 5a_{n-1} - 6a_{n-2}, \tag{6.29}$$

with initial conditions $a_0 = 1, a_1 = 1$. Then $p = 2, c_1 = 5, c_2 = -6$, and the characteristic equation is given by

$$x^2 - 5x + 6 = 0.$$

This has roots $x = 2$ and $x = 3$, so $\alpha_1 = 2$ and $\alpha_2 = 3$ are the characteristic roots.

If α is a characteristic root of the recurrence (6.25), and if we take $a_n = \alpha^n$, it follows that the sequence (a_n) satisfies the recurrence. Thus, corresponding to each characteristic root, we have a solution to the recurrence. In (6.29), $a_n = 2^n$ and $a_n = 3^n$ give solutions. However, neither satisfies both initial conditions $a_0 = 1, a_1 = 1$.

The next important observation to be made is that if the sequences (a'_n) and (a''_n) both satisfy the recurrence (6.25) and if λ_1 and λ_2 are constants, then the sequence (a'''_n), where $a'''_n = \lambda_1 a'_n + \lambda_2 a''_n$, also is a solution to (6.25). In other words, a weighted sum of solutions is a solution. To see this, note that

$$a'_n = c_1 a'_{n-1} + c_2 a'_{n-2} + \cdots + c_p a'_{n-p} \tag{6.30}$$

and

$$a''_n = c_1 a''_{n-1} + c_2 a''_{n-2} + \cdots + c_p a''_{n-p}. \tag{6.31}$$

Multiplying (6.30) by λ_1 and (6.31) by λ_2 and adding gives us

$$\begin{aligned} a'''_n &= \lambda_1 a'_n + \lambda_2 a''_n \\ &= \lambda_1(c_1 a'_{n-1} + c_2 a'_{n-2} + \cdots + c_p a'_{n-p}) + \lambda_2(c_1 a''_{n-1} + c_2 a''_{n-2} + \cdots + c_p a''_{n-p}) \\ &= c_1(\lambda_1 a'_{n-1} + \lambda_2 a''_{n-1}) + c_2(\lambda_1 a'_{n-2} + \lambda_2 a''_{n-2}) + \cdots + c_p(\lambda_1 a'_{n-p} + \lambda_2 a''_{n-p}) \\ &= c_1 a'''_{n-1} + c_2 a'''_{n-2} + \cdots + c_p a'''_{n-p}. \end{aligned}$$

Thus, a'''_n does satisfy (6.25). For example, if we define $a_n = 3 \cdot 2^n + 8 \cdot 3^n$, it follows that a_n satisfies (6.29).

In general, suppose that $\alpha_1, \alpha_2, \ldots, \alpha_p$ are the characteristic roots of recurrence (6.25). Then our reasoning shows that if $\lambda_1, \lambda_2, \ldots, \lambda_p$ are constants, and if

$$a_n = \lambda_1 \alpha_1^n + \lambda_2 \alpha_2^n + \cdots + \lambda_p \alpha_p^n,$$

then a_n satisfies (6.25). It turns out that every solution of (6.25) can be expressed in this form, *provided that the roots* $\alpha_1, \alpha_2, \ldots, \alpha_p$ are distinct. For a proof of this fact, see the end of this subsection.

Theorem 6.1 Suppose that a linear homogeneous recurrence (6.25) with constant coefficients has characteristic roots $\alpha_1, \alpha_2, \ldots, \alpha_p$. Then if $\lambda_1, \lambda_2, \ldots, \lambda_p$ are constants, every expression of the form

$$a_n = \lambda_1 \alpha_1^n + \lambda_2 \alpha_2^n + \cdots + \lambda_p \alpha_p^n \tag{6.32}$$

is a solution to the recurrence. Moreover, if the characteristic roots are distinct, every solution to the recurrence has the form (6.32) for some constants $\lambda_1, \lambda_2, \ldots, \lambda_p$.

We call the expression in (6.32) the *general solution* of the recurrence (6.25).

It follows from Theorem 6.1 that to find the unique solution of a recurrence (6.25) subject to initial conditions $a_0, a_1, \ldots, a_{p-1}$, if the characteristic roots are distinct, we simply need to find values for the constants $\lambda_1, \lambda_2, \ldots, \lambda_p$ in the general solution so that the initial conditions are satisfied. Let us see how to find these λ_i. In (6.29), every solution has the form

$$a_n = \lambda_1 2^n + \lambda_2 3^n.$$

Now we have

$$a_0 = \lambda_1 2^0 + \lambda_2 3^0, a_1 = \lambda_1 2^1 + \lambda_2 3^1.$$

So, from $a_0 = 1$ and $a_1 = 1$, we get the system of equations

$$\begin{aligned} \lambda_1 + \lambda_2 &= 1 \\ 2\lambda_1 + 3\lambda_2 &= 1. \end{aligned}$$

This system has the unique solution $\lambda_1 = 2, \lambda_2 = -1$. Hence, since $\alpha_1 \neq \alpha_2$, the unique solution to (6.29) with the given initial conditions is $a_n = 2 \cdot 2^n - 3^n$.

The general procedure works just as in this example. If we define a_n by (6.32), we use the initial values of $a_0, a_1, \ldots, a_{p-1}$ to set up a system of p simultaneous equations in the p unknowns $\lambda_1, \lambda_2, \ldots, \lambda_p$. One can show that *if* $\alpha_1, \alpha_2, \ldots, \alpha_p$ are distinct, this system always has a unique solution. The proof of this is the essence of the rest of the proof of Theorem 6.1, which we shall now present.

We close this subsection by sketching a proof of the statement in Theorem 6.1 that if the characteristic roots $\alpha_1, \alpha_2, \ldots, \alpha_p$ are distinct, every solution of a recurrence (6.25) has the form (6.32) for some constants $\lambda_1, \lambda_2, \ldots, \lambda_p$.[7] Suppose that

$$b_n = \lambda_1 \alpha_1^n + \lambda_2 \alpha_2^n + \cdots + \lambda_p \alpha_p^n$$

[7] The proof may be omitted.

is a solution to (6.25). Using the initial conditions $b_0 = a_0, b_1 = a_1, \ldots, b_{p-1} = a_{p-1}$, we find that

$$\left.\begin{array}{rcl} \lambda_1 + \lambda_2 + \cdots + \lambda_p & = & a_0 \\ \lambda_1\alpha_1 + \lambda_2\alpha_2 + \cdots + \lambda_p\alpha_p & = & a_1 \\ \lambda_1\alpha_1^2 + \lambda_2\alpha_2^2 + \cdots + \lambda_p\alpha_p^2 & = & a_2 \\ \vdots & & \\ \lambda_1\alpha_1^{p-1} + \lambda_2\alpha_2^{p-1} + \cdots + \lambda_p\alpha_p^{p-1} & = & a_{p-1}. \end{array}\right\} \tag{6.33}$$

Equations (6.33) are a system of p linear equations in the p unknowns $\lambda_1, \lambda_2, \ldots, \lambda_p$. Consider now the matrix of coefficients of the system (6.33):

$$\begin{bmatrix} 1 & 1 & & 1 \\ \alpha_1 & \alpha_2 & & \alpha_p \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_p^2 \\ \vdots & \vdots & & \vdots \\ \alpha_1^{p-1} & \alpha_2^{p-1} & & \alpha_p^{p-1} \end{bmatrix}.$$

The determinant of this matrix is the famous *Vandermonde determinant.* One can show that the Vandermonde determinant is given by the product

$$\prod_{1 \le i < j \le p} (\alpha_j - \alpha_i),$$

the product of all terms $\alpha_j - \alpha_i$ with $1 \le i < j \le p$. Since $\alpha_1, \alpha_2, \ldots, \alpha_p$ are distinct, the determinant is not zero. Thus, there is a unique solution $\lambda_1, \lambda_2, \ldots, \lambda_p$ of the system (6.33). Hence, we see that a recurrence (6.25) with initial conditions $a_0, a_1, \ldots, a_{p-1}$ has a solution of the form (6.32). Now the recurrence with these initial conditions has just one solution, so this must be it. That completes the proof.

6.2.2 Computation of the kth Fibonacci Number

Let us illustrate the method with another example, the recurrence (6.11) for the Fibonacci numbers, which we repeat here:

$$F_k = F_{k-1} + F_{k-2}. \tag{6.34}$$

Here $p = 2$ and $c_1 = c_2 = 1$. The characteristic equation is given by $x^2 - x - 1 = 0$. By the quadratic formula, the roots of this equation, the characteristic roots, are given by $\alpha_1 = (1 + \sqrt{5})/2$ and $\alpha_2 = (1 - \sqrt{5})/2$. Because $\alpha_1 \ne \alpha_2$, the general solution is

$$\lambda_1 \left(\frac{1+\sqrt{5}}{2}\right)^k + \lambda_2 \left(\frac{1-\sqrt{5}}{2}\right)^k.$$

The initial conditions $F_0 = F_1 = 1$ give us the two equations

$$\lambda_1 + \lambda_2 = 1$$

and

$$\lambda_1\left(\frac{1+\sqrt{5}}{2}\right)+\lambda_2\left(\frac{1-\sqrt{5}}{2}\right)=1.$$

Solving these simultaneous equations for λ_1 and λ_2 gives us

$$\lambda_1=\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right),\quad \lambda_2=-\frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right).$$

Hence, under the given initial conditions, the solution to (6.34), that is, the kth Fibonacci number, is given by

$$F_k=\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)\left(\frac{1+\sqrt{5}}{2}\right)^k-\frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)\left(\frac{1-\sqrt{5}}{2}\right)^k,$$

or

$$F_k=\frac{\left(\frac{1+\sqrt{5}}{2}\right)^{k+1}-\left(\frac{1-\sqrt{5}}{2}\right)^{k+1}}{\sqrt{5}}, \tag{6.35}$$

or

$$F_k=\frac{\tau^{k+1}-(1-\tau)^{k+1}}{\sqrt{5}},$$

where τ is the golden ratio of Section 6.1.2. In Exercise 8 of Section 6.3, this result is derived using generating functions.

Example 6.12 Duration of Messages (Example 6.5 Revisited) We note next that the recurrences (6.9) and (6.34) are the same. Moreover, the initial conditions are the same. For $F_1=1$ and $F_2=2$, while $N_1=1$ and $N_2=2$. Also, for the same reason that we took F_0 to be 0, namely, to maintain the recurrence even for $k=2$, we take N_0 to be 0. Now as we observed earlier, if we are given a recurrence (6.25) that has distinct characteristic roots and initial conditions $a_0, a_1, \ldots, a_{p-1}$, the solution is determined uniquely. Hence, it follows that N_t must equal F_t for all $t\geq 0$, so we may use (6.35) to compute N_k. It is not too hard to show from this result that the Shannon capacity defined in Example 6.5 is given by

$$C=\log_2\left(\frac{1+\sqrt{5}}{2}\right).$$

■

6.2.3 The Case of Multiple Roots

Consider the recurrence

$$a_n=6a_{n-1}-9a_{n-2}, \tag{6.36}$$

with $a_0=1, a_1=2$. Its characteristic equation is $x^2-6x+9=0$, or $(x-3)^2=0$. The two characteristic roots are 3 and 3, that is, 3 is a multiple root. Hence, the second part of Theorem 6.1 does not apply. Whereas it is still true that 3^n is a solution of (6.36), and it is also true that $\lambda_1 3^n + \lambda_2 3^n$ is always a solution, it is not true that every solution of (6.36) takes the form $\lambda_1 3^n + \lambda_2 3^n$. In particular, there is no such solution satisfying our given initial conditions. For these conditions give us the equations

$$\begin{aligned} \lambda_1 + \lambda_2 &= 1 \\ 3\lambda_1 + 3\lambda_2 &= 2. \end{aligned}$$

There are no λ_1, λ_2 satisfying these two equations.

Suppose that α is a characteristic root of multiplicity u; that is, it appears as a root of the characteristic equation exactly u times. Then it turns out that not only does $a_n = \alpha^n$ satisfy the recurrence (6.25), but so do $a_n = n\alpha^n, a_n = n^2\alpha^n, \ldots,$ and $a_n = n^{u-1}\alpha^n$ (see Exercises 29, 30). In our example, 3 is a characteristic root of multiplicity $u = 2$, and both $a_n = 3^n$ and $a_n = n3^n$ are solutions of (6.25). Moreover, since a weighted sum of solutions is a solution and since both $a_n' = 3^n$ and $a_n'' = n3^n$ are solutions, so is $a_n''' = \lambda_1 3^n + \lambda_2 n 3^n$. Using this expression a_n''' and the initial conditions $a_0 = 1, a_1 = 2$, we get the equations

$$\begin{aligned} \lambda_1 &= 1 \\ 3\lambda_1 + 3\lambda_2 &= 2. \end{aligned}$$

These have the unique solution $\lambda_1 = 1, \lambda_2 = -\frac{1}{3}$. Hence, $a_n = 3^n - \frac{1}{3} \cdot n \cdot 3^n$ is a solution to the recurrence (6.36) with the initial conditions $a_0 = 1, a_1 = 2$. It follows that this must be the unique solution.

This procedure generalizes as follows. Suppose that a recurrence (6.25) has characteristic roots $\alpha_1, \alpha_2, \ldots, \alpha_q$, with α_i having multiplicity u_i. Then

$$\begin{gathered} \alpha_1^n, n\alpha_1^n, n^2\alpha_1^n, \ldots, n^{u_1-1}\alpha_1^n, \alpha_2^n, n\alpha_2^n, n^2\alpha_2^n, \ldots, n^{u_2-1}\alpha_2^n, \ldots, \\ \alpha_q^n, n\alpha_q^n, n^2\alpha_q^n, \ldots, n^{u_q-1}\alpha_q^n \end{gathered}$$

must all be solutions of the recurrence. Let us call these the *basic solutions*. There are p of these basic solutions in all. Let us denote them $b_1, b_2, \ldots, b_p$. Since a weighted sum of solutions is a solution, for any constants $\lambda_1, \lambda_2, \ldots, \lambda_p$,

$$a_n = \lambda_1 b_1 + \lambda_2 b_2 + \cdots + \lambda_p b_p$$

is also a solution of the recurrence. By a method analogous to that used to prove Theorem 6.1, one can show that every solution has this form for some constants $\lambda_1, \lambda_2, \ldots, \lambda_p$.

Theorem 6.2 Suppose that a linear homogeneous recurrence (6.25) with constant coefficients has basic solutions $b_1, b_2, \ldots, b_p$. Then the general solution is given by

$$a_n = \lambda_1 b_1 + \lambda_2 b_2 + \cdots + \lambda_p b_p, \tag{6.37}$$

for some constants $\lambda_1, \lambda_2, \ldots, \lambda_p$.

The unique solution satisfying initial conditions $a_0, a_1, \ldots, a_{p-1}$ can be computed by setting $n = 0, 1, \ldots, p-1$ in (6.37) and getting p simultaneous equations in the p unknowns $\lambda_1, \lambda_2, \ldots, \lambda_p$.

To illustrate, consider the recurrence

$$a_n = 7a_{n-1} - 16a_{n-2} + 12a_{n-3}, \tag{6.38}$$

$a_0 = 1, a_1 = 2, a_2 = 0$. Then the characteristic equation is

$$x^3 - 7x^2 + 16x - 12 = 0,$$

which factors as $(x-2)(x-2)(x-3) = 0$. The characteristic roots are therefore $\alpha_1 = 2$, with multiplicity $u_1 = 2$, and $\alpha_2 = 3$, with multiplicity $u_2 = 1$. Thus, the general solution to (6.38) has the form

$$a_n = \lambda_1\alpha_1^n + \lambda_2 n\alpha_1^n + \lambda_3\alpha_2^n = \lambda_1 2^n + \lambda_2 n2^n + \lambda_3 3^n.$$

Setting $n = 0, 1, 2$, we get

$$\begin{aligned} a_0 &= \lambda_1 + \lambda_3 = 1 \\ a_1 &= 2\lambda_1 + 2\lambda_2 + 3\lambda_3 = 2 \\ a_2 &= 4\lambda_1 + 8\lambda_2 + 9\lambda_3 = 0. \end{aligned}$$

This system has the unique solution $\lambda_1 = 5, \lambda_2 = 2, \lambda_3 = -4$. Hence, the unique solution to (6.38) with the given initial conditions is

$$a_n = 5 \cdot 2^n + 2 \cdot n2^n - 4 \cdot 3^n.$$

EXERCISES FOR SECTION 6.2

1. Which of the following recurrences are linear?
   - (a) $a_n = 5a_{n-1} + 2a_{n-2} + 3$
   - (b) $b_n = 3b_{n-1} + 9b_{n-2} + 18b_{n-3} + 32b_{n-4}$
   - (c) $c_n = 21c_{n-2} + 4c_{n-5}$
   - (d) $d_n = 16d_{n-1} - 12d_{n-2}$
   - (e) $e_n = 24e_{n-1} + 22e_{n-2}^2$
   - (f) $f_n = nf_{n-1} + f_{n-2}$
   - (g) $g_n = n^2 g_{n-2}$
   - (h) $h_n = 8h_{n-3} + 81$
   - (i) $i_n = 5i_{n-1} + 3^n i_{n-2}$
2. Which of the recurrences in Exercise 1 are homogeneous?
3. Which of the recurrences in Exercise 1 have constant coefficients?
4. Use (6.35) to derive the values of F_k for $k = 2, 3, 4, 5, 6, 7$.
5. Find the characteristic equation of each of the following recurrences.
   - (a) $a_n = -2a_{n-1} - a_{n-2}$
   - (b) $b_k = -7b_{k-1} + 18b_{k-2}$
   - (c) $c_n = 3c_{n-1} + 18c_{n-2} - 7c_{n-3}$
   - (d) $d_n = 81d_{n-4} + 4d_{n-5}$
   - (e) $e_k = 4e_{k-2}$
   - (f) $f_{n+1} = 2f_n + 3f_{n-1}$
   - (g) $g_n = 18g_{n-7}$
   - (h) $h_n = 9h_{n-2}$
   - (i) $i_n = i_{n-1} + 4i_{n-2} - 4i_{n-3}$
   - (j) $j_n = 2j_{n-1} + 9j_{n-2} - 18j_{n-3}$
   - (k) $k_n = 11k_{n-1} + 22k_{n-2} + 11k_{n-3} - 33k_{n-8}$

6. In Exercise 5, find the characteristic roots of the recurrences of parts (a), (b), (e), (f), (h), (i), and (j). [*Hint:* 2 is a root in parts (i) and (j).]
7. (a) Show that the recurrence $a_n = 5a_{n-1}$ can have many solutions.
   (b) Show that this recurrence has a unique solution if we know that $a_0 = 20$.
8. Solve the following recurrences using the method of characteristic roots.

   (a) $a_n = 6a_{n-1}, a_0 = 5$ (b) $t_{k+1} = 2t_k, t_1 = 1$ (this is Example 6.1)

9. Consider the recurrence $a_n = 15a_{n-1} - 44a_{n-2}$. Show that each of the following sequences is a solution.

   (a) (4^n) (b) $(3 \cdot 11^n)$ (c) $(4^n - 11^n)$ (d) (4^{n+1})

10. In Exercise 9, which of the following sequences is a solution?
    (a) (-4^n) (b) $(4^n + 1)$ (c) $(3 \cdot 4^n + 12 \cdot 11^n)$
    (d) $(n4^n)$ (e) $(4^n 11^n)$ (f) $((-4)^n)$
11. Consider the recurrence $b_k = 7b_{k-1} - 10b_{k-2}$. Which of the following sequences is a solution?
    (a) (2^k) (b) $(5^k - 2^k)$ (c) $(2^k + 7)$
    (d) $(2^k - 5^k)$ (e) $(2^k + 5^{k+1})$
12. Use the method of characteristic roots to solve the following recurrences in Exercise 5 under the following initial conditions.

    (a) That of part (a) if $a_0 = 2, a_1 = 2$
    (b) That of part (b) if $b_0 = 0, b_1 = 8$
    (c) That of part (e) if $e_0 = -1, e_1 = 1$
    (d) That of part (f) if $f_0 = f_1 = 2$
    (e) That of part (h) if $h_0 = 4, h_1 = 2$
    (f) That of part (i) if $i_0 = 0, i_1 = 1, i_2 = 2$
    (g) That of part (j) if $j_0 = 2, j_1 = 1, j_2 = 0$

13. Suppose that in Example 6.5, a requires 2 units of time to transmit and b requires 3 units of time. Solve for N_t.
14. Solve for a_n in the product sales problem of Exercise 18, Section 6.1 if $a_0 = 0, a_1 = 1$.
15. Using (6.35), verify the results in Exercise 29, Section 6.1.
16. Using (6.35), verify the results in Exercise 30, Section 6.1.
17. Consider the recurrence $a_n = -a_{n-2}$.

    (a) Show that i and $-i$ are the characteristic roots. (*Note*: $i = \sqrt{-1}$.)
    (b) Is the sequence (i^n) a solution?
    (c) What about the sequence $(2i^n + (-i)^n)$?
    (d) Find the unique solution if $a_0 = 0, a_1 = 1$.

18. Consider the recurrence $a_n = -9a_{n-2}$.

(a) Find a general solution.

(b) Find the unique solution if $a_0 = 0, a_1 = 5$.

19. Consider the recurrence $F_n = 4F_{n-1} - 4F_{n-2}$. Show that there is a multiple characteristic root α and that for the initial conditions $F_0 = 1, F_1 = 3$, there are no constants λ_1 and λ_2 so that $F_n = \lambda_1\alpha^n + \lambda_2\alpha^n$ for all n.

20. Suppose that (a_n') and (a_n'') are two solutions of a recurrence (6.25) and that $a_n''' = a_n' - a_n''$. Is (a_n''') necessarily a solution to (6.25)? Why?

21. Suppose that (a_n'), (a_n''), and (a_n''') are three solutions to a recurrence (6.25) and that we have $b_n = \lambda_1 a_n' + \lambda_2 a_n'' + \lambda_3 a_n'''$. Is (b_n) necessarily a solution to (6.25)? Why?

22. Consider the recurrence

$$a_n = 9a_{n-1} - 27a_{n-2} + 27a_{n-3}.$$

Show that each of the following sequences is a solution.

(a) (3^n) (b) $(n3^n)$ (c) $(n^2 3^n)$
(d) $(3 \cdot 3^n)$ (e) $(3^n + n3^n)$ (f) $(4 \cdot 3^n + 8 \cdot n3^n - n^2 3^n)$

23. Consider the recurrence

$$a_n = 3a_{n-1} - 3a_{n-2} + a_{n-3}.$$

Show that each of the following sequences is a solution:

(a) $(1, 1, 1, \ldots)$ (b) $(0, 1, 2, 3, \ldots)$ (c) $(0, 1, 4, 9, \ldots)$

24. Consider the recurrence

$$b_n = 9b_{n-1} - 24b_{n-2} + 20b_{n-3}.$$

Show that each of the sequences (2^n), $(n2^n)$, and (5^n) is a solution.

25. Consider the recurrence

$$c_k = 13c_{k-1} - 60c_{k-2} + 112c_{k-3} - 64c_{k-4}.$$

Show that each of the following sequences is a solution.

(a) $(2, 2, \ldots)$ (b) $(3 \cdot 4^k)$ (c) $(4^k + k4^k + k^2 4^k)$
(d) $(4^k + 1)$ (e) $(k4^k - 11)$

26. Find the unique solution to:

(a) The recurrence of Exercise 22 if $a_0 = 0, a_1 = 1, a_2 = 1$

(b) The recurrence of Exercise 23 if $a_0 = 1, a_1 = 1, a_2 = 2$

(c) The recurrence of Exercise 24 if $b_0 = 1, b_1 = 2, b_2 = 0$

(d) The recurrence of Exercise 25 if $c_0 = c_1 = 0, c_2 = 10, c_3 = 0$

27. Solve the following recurrence relations under the given initial conditions.

(a) $a_n = 10a_{n-1} - 25a_{n-2}, a_0 = 1, a_1 = 2$

(b) $b_k = 14b_{k-1} - 49b_{k-2}, b_0 = 0, b_1 = 10$

(c) $c_n = 9c_{n-1} - 15c_{n-2} + 7c_{n-3}, c_0 = 0, c_1 = 1, c_2 = 2$ (*Hint:* $x = 1$ is a characteristic root.)

(d) $d_n = 13d_{n-1} - 40d_{n-2} + 36d_{n-3}, d_0 = d_1 = 1, d_2 = 0$ (*Hint:* $x = 2$ is a characteristic root.)

(e) $e_k = 10e_{k-1} - 37e_{k-2} + 60e_{k-3} - 36e_{k-4}, e_0 = e_1 = e_2 = 0, e_3 = 5$ (*Hint:* $x = 2$ and $x = 3$ are characteristic roots.)

28. Solve the following recurrence under the given initial condition.

$$a_n = -2a_{n-2} - a_{n-4}, \quad a_0 = 0, \quad a_1 = 1, \quad a_2 = 2, \quad a_3 = 3.$$

(*Hint:* $x = i$ and $x = -i$ are characteristic roots.)

29. Suppose that α is a characteristic root of the recurrence (6.25) and α has multiplicity 2. Show that (α^n) and $(n\alpha^n)$ are solutions to (6.25). [*Hint:* If $C(x) = 0$ is the characteristic equation, then $C(x) = (x - \alpha)^2 D(x)$ for some polynomial $D(x)$. If $C_n(x) = x^{n-p}C(x)$, show that α is a root of the derivative $C_n'(x)$. Substituting α for x in the equation $xC_n'(x) = 0$ shows that $(n\alpha^n)$ is a solution to (6.25).]

30. (a) Suppose that α is a characteristic root of the recurrence (6.25) and α has multiplicity 3. Show that $(\alpha^n), (n\alpha^n)$, and $(n^2\alpha^n)$ are solutions to (6.25). [*Hint:* Generalize the argument in Exercise 29 by noting that $C(x) = (x - \alpha)^3 D(x)$. Consider $C_n(x) = x^{n-p}C(x), A_n(x) = xC_n'(x)$, and $B_n(x) = xA_n'(x)$. Show that $(n\alpha^n)$ is a solution by considering $A_n(x) = 0$, and $(n^2\alpha^n)$ is a solution by considering $B_n(x) = 0$.]

(b) Generalize to the case where α is a characteristic root of multiplicity u.

6.3 SOLVING RECURRENCES USING GENERATING FUNCTIONS

6.3.1 The Method

Another method for solving recurrences uses the notion of generating function developed in Chapter 5. Suppose that $G(x)$ is the ordinary generating function for the sequence (a_k), that is, the function

$$G(x) = \sum_{k=0}^{\infty} a_k x^k.$$

We shall try to find (a_k) by finding its generating function. In particular, if we have a recurrence for a_k, the trick will be to multiply both sides of the recurrence by x^k and then take the sum, giving us an expression that can be used to derive $G(x)$.

Example 6.13 The Grains of Wheat (Example 6.1 Revisited) Let us illustrate the method with the recurrence relation of Example 6.1,

$$t_{k+1} = 2t_k. \tag{6.39}$$

The initial condition was $t_1 = 1$. In this case, t_0 is not defined. However, it will usually be convenient to think of our sequences as beginning with the zeroth term. Hence, we will try to define the early terms from the recurrence if they are not known or given. In particular, by (6.39), it is consistent to take

$$t_0 = \frac{1}{2}t_1 = \frac{1}{2}.$$

The ordinary generating function for (t_k) is

$$G(x) = \sum_{k=0}^{\infty} t_k x^k.$$

To derive $G(x)$, we start by multiplying both sides of the recurrence (6.39) by x_k, obtaining

$$t_{k+1}x^k = 2t_k x^k.$$

Then, we take sums:[8]

$$\sum_{k=0}^{\infty} t_{k+1}x^k = 2\sum_{k=0}^{\infty} t_k x^k. \tag{6.40}$$

The right-hand side of (6.40) is $2G(x)$. What is the left-hand side? We shall try to reduce that to an expression involving $G(x)$. Note that

$$\begin{aligned}
\textstyle\sum_{k=0}^{\infty} t_{k+1}x^k &= t_1 + t_2 x + t_3 x^2 + \cdots \\
&= \frac{1}{x}\left[t_1 x + t_2 x^2 + t_3 x^3 + \cdots\right] \\
&= \frac{1}{x}\left[t_0 + t_1 x + t_2 x^2 + t_3 x^3 + \cdots\right] - \frac{1}{x}t_0 \\
&= \frac{1}{x}\left[G(x) - t_0\right].
\end{aligned}$$

Hence,

$$\sum_{k=0}^{\infty} t_{k+1}x_k = \frac{G(x) - t_0}{x}. \tag{6.41}$$

Equations (6.40) and (6.41) now give us the equation

$$\frac{G(x) - t_0}{x} = 2G(x).$$

This equation, a *functional equation* for $G(x)$, can be solved for $G(x)$. A little bit of algebraic manipulation gives us

$$G(x) = \frac{t_0}{1 - 2x}.$$

[8] The sums may be taken over all values of k for which the recurrence applies. In some cases, it will be better or more appropriate to take the sum from $k = 1$ or from $k = 2$, and so on.

Since we have computed $t_0 = \frac{1}{2}$, we have

$$G(x) = \frac{1}{2}(1-2x)^{-1}.$$

Knowing $G(x)$, we can compute the desired value of t_k from it. The number t_k is given by the coefficient of x^k if we expand out $G(x)$. How can we expand $G(x)$? There are two methods. The easiest is to use the identity

$$1 + y + y^2 + \cdots + y^n + \cdots = \frac{1}{1-y}, \tag{6.42}$$

$|y| < 1$ [see Equation (5.2)]. Doing so gives us the result

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

or

$$G(x) = \frac{1}{2} + x + 2x^2 + \cdots + 2^{n-1}x^n + \cdots.$$

In other words,

$$t_k = 2^{k-1},$$

which agrees with our earlier computation. An alternative way of expanding $G(x)$ is to use the Binomial Theorem (Theorem 5.3). We leave it to the reader to try this. ■

Example 6.14 Legitimate Codewords (Example 6.4 Revisited) Let us now illustrate the method with the following recurrence of Example 6.4:

$$a_{k+1} = 2a_k + 4^k. \tag{6.43}$$

We use the ordinary generating function

$$G(x) = \sum_{k=0}^{\infty} a_k x^k.$$

We know that $a_1 = 3$. From the recurrence, we can derive a_0 even though a_0 is not defined. We obtain

$$\begin{aligned} a_1 &= 2a_0 + 4^0, \\ 3 &= 2a_0 + 1, \\ a_0 &= 1. \end{aligned}$$

We now multiply both sides of the recurrence (6.43) by x^k and sum, obtaining

$$\sum_{k=0}^{\infty} a_{k+1}x^k = 2\sum_{k=0}^{\infty} a_k x^k + \sum_{k=0}^{\infty} 4^k x^k. \tag{6.44}$$

The left-hand side of (6.44) is given by

$$\frac{1}{x}\sum_{k=0}^{\infty} a_{k+1}x^{k+1} = \frac{1}{x}\left[G(x) - a_0\right] = \frac{1}{x}\left[G(x) - 1\right].$$

Hence, we obtain

$$\frac{1}{x}\left[G(x) - 1\right] = 2G(x) + \sum_{k=0}^{\infty}(4x)^k. \tag{6.45}$$

From the identity (6.42), we can rewrite this as

$$\frac{1}{x}\left[G(x) - 1\right] = 2G(x) + \frac{1}{1-4x}.$$

From this functional equation, it is simply a matter of algebraic manipulation to solve for $G(x)$. We obtain

$$\begin{aligned}
G(x) - 1 &= 2xG(x) + \frac{x}{1-4x}, \\
G(x)(1-2x) &= 1 + \frac{x}{1-4x}, \\
G(x) &= \frac{1}{1-2x}\left(1 + \frac{x}{1-4x}\right), \\
G(x) &= \frac{1}{1-2x} + \frac{x}{(1-2x)(1-4x)}.
\end{aligned} \tag{6.46}$$

This gives us the generating function for (a_k). How do we find a_k? It is easy enough to expand out the first term on the right-hand side of (6.46). The second term we expand by the method of partial fractions.[9] Namely, the second term on the right-hand side can be expressed as

$$\frac{a}{1-2x} + \frac{b}{1-4x}$$

for appropriate a and b. We compute that

$$a = -\frac{1}{2}, \quad b = \frac{1}{2}.$$

Thus,

$$G(x) = \frac{1}{1-2x} + \frac{-\frac{1}{2}}{1-2x} + \frac{\frac{1}{2}}{1-4x} = \frac{\frac{1}{2}}{1-2x} + \frac{\frac{1}{2}}{1-4x}. \tag{6.47}$$

To expand (6.47), we again use the identity (6.42), obtaining

$$G(x) = \frac{1}{2}\sum_{k=0}^{\infty}(2x)^k + \frac{1}{2}\sum_{k=0}^{\infty}(4x)^k.$$

[9] See most calculus texts for a discussion of this method.

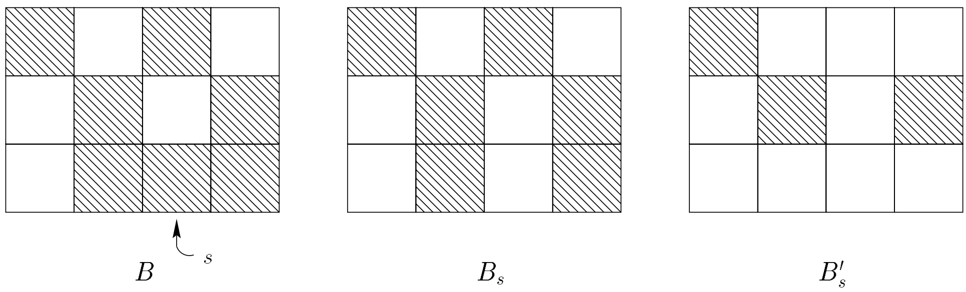


Figure 6.7: A board B, a square s [the $(3,3)$ square], and the boards B_s and B_s' obtained from board B and square s.

Thus, the coefficient of x^k is given by

$$a_k = \frac{1}{2}(2)^k + \frac{1}{2}(4)^k.$$

In particular, we can check our computation in Section 6.1. We have

$$\begin{aligned} a_2 &= \tfrac{1}{2}(2)^2 + \tfrac{1}{2}(4)^2 = 10, \\ a_3 &= \tfrac{1}{2}(2)^3 + \tfrac{1}{2}(4)^3 = 36. \end{aligned}$$

The reader might wish to check our results in still another way, namely by computing an exponential generating function for a_k directly by the methods of Section 5.5. ■

Example 6.15 Rook Polynomials[10] In Examples 5.10 and 5.14, we introduced rook polynomials and stated a result that would reduce computation of a rook polynomial of a board B to computation of the rook polynomials of "simpler" boards. Here we state another such result. Suppose that s is any darkened square of the board B. Let B_s be obtained from B by forbidding s (lightening s) and let B_s' be obtained from B by forbidding all squares in the same row or column as s. Figure 6.7 shows a board B, a square s, and the boards B_s and B_s'.

Note that to place $k \geq 1$ rooks on B, we either use square s or we do not. If we do not use square s, we have to place k rooks on the squares of B_s. If we use square s, we have $k-1$ rooks still to place, and we may use any darkened square of B except those in the same row or column as s; that is, we may use any darkened square of B_s'. Thus, by the sum rule of Chapter 2,

$$r_k(B) = r_k(B_s) + r_{k-1}(B_s') \tag{6.48}$$

for $k \geq 1$. If we multiply both sides of (6.48) by x^k and sum over all $k \geq 1$, we find that

$$\sum_{k=1}^{\infty} r_k(B)x^k = \sum_{k=1}^{\infty} r_k(B_s)x^k + \sum_{k=1}^{\infty} r_{k-1}(B_s')x^k. \tag{6.49}$$

[10] This example may be omitted if the reader has skipped Chapter 5.

The term on the left-hand side of (6.49) is just

$$R(x, B) - r_0(B) = R(x, B) - 1,$$

since $r_0(B) = 1$ for all boards B. The first term on the right-hand side is

$$R(x, B_s) - r_0(B_s) = R(x, B_s) - 1.$$

The second term on the right-hand side is equal to

$$x \sum_{k=1}^{\infty} r_{k-1}(B_s')x^{k-1} = x \sum_{k=0}^{\infty} r_k(B_s')x^k = xR(x, B_s').$$

Thus, (6.49) gives us

$$R(x, B) - 1 = R(x, B_s) - 1 + xR(x, B_s'),$$

or

$$R(x, B) = R(x, B_s) + xR(x, B_s'). \tag{6.50}$$

Application of this result to the board B of Figure 6.7 is left as an exercise (Exercise 27). ■

The method used in the preceding three examples can be applied to solve a variety of recurrences. It will always work on any recurrence like (6.25) which is linear and homogeneous with constant coefficients.[11] The result will give a generating function $G(x)$ of the form

$$\frac{p(x)}{q(x)},$$

where $p(x)$ is a polynomial of degree less than p and $q(x)$ is a polynomial of degree p and constant term equal to 1. [The polynomials $p(x)$ and $q(x)$ can be expressed in terms of the coefficients $c_1, c_2, \ldots, c_p$ and initial conditions $a_0, a_1, \ldots, a_{p-1}$ of the recurrence (6.25). See Brualdi [1999] or Exercise 25 for details.] If all the roots of $q(x)$ are real numbers, one can then use the method of partial fractions to express $p(x)/q(x)$ as a sum of terms of the form

$$\frac{\alpha}{(1 - \beta x)^t},$$

where t is a positive integer and α and β are real numbers. In turn, the terms

$$\frac{\alpha}{(1 - \beta x)^t}$$

can be expanded out using the Binomial Theorem, giving us

$$\frac{\alpha}{(1 - \beta x)^t} = \alpha \sum_{k=0}^{\infty} \binom{t+k-1}{k} \beta^k x^k. \tag{6.51}$$

[11] The rest of this subsection may be omitted on first reading.

This also follows directly by using βx in place of x in Corollary 5.4.1.

If $q(x)$ has complex roots, the method of partial fractions can be used to express $p(x)/q(x)$ as a sum of terms of the form

$$\frac{a}{(x-b)^t} \quad \text{or} \quad \frac{ax+b}{(x^2+cx+d)^t},$$

where t is a positive integer and a, b, c, and d are real numbers. The former terms can be changed into terms of the form

$$\frac{\alpha}{(1-\beta x)^t}.$$

The latter terms can be manipulated by completing the square in the denominator and then using the expansion for

$$\frac{1}{(1+y^2)^t} = \frac{1}{[1-(-y^2)]^t}.$$

We omit the details.

6.3.2 Derangements

Let us next use the techniques of this section to derive the formula for the number of derangements D_n of n elements. We have the recurrence

$$D_{n+1} = n(D_{n-1} + D_n), \tag{6.52}$$

$n \geq 2$. We know that $D_2 = 1$ and $D_1 = 0$. Hence, using the recurrence (6.52), we derive $D_0 = 1$. With $D_0 = 1$, (6.52) holds for $n \geq 1$. The recurrence (6.52) is inconvenient because it expresses D_{n+1} in terms of both D_n and D_{n-1}. Some algebraic manipulation reduces (6.52) to the recurrence

$$D_{n+1} = (n+1)D_n + (-1)^{n+1}, \tag{6.53}$$

$n \geq 0$. For a detailed verification of this fact, see the end of this subsection.

Let us try to calculate the ordinary generating function

$$G(x) = \sum_{n=0}^{\infty} D_n x^n.$$

We multiply (6.53) by x^n and sum, obtaining

$$\sum_{n=0}^{\infty} D_{n+1}x^n = \sum_{n=0}^{\infty}(n+1)D_n x^n + \sum_{n=0}^{\infty}(-1)^{n+1}x^n. \tag{6.54}$$

The left-hand side of (6.54) is

$$\frac{1}{x}\left[G(x) - D_0\right] = \frac{1}{x}G(x) - \frac{1}{x}.$$

The second term on the right-hand side is

$$-\sum_{n=0}^{\infty}(-1)^n x^n = -\sum_{n=0}^{\infty}(-x)^n = -\frac{1}{1+x},$$

using the identity (6.42). Finally, the first term on the right-hand side can be rewritten as

$$\sum_{n=0}^{\infty} nD_n x^n + \sum_{n=0}^{\infty} D_n x^n = x\sum_{n=0}^{\infty} nD_n x^{n-1} + \sum_{n=0}^{\infty} D_n x^n = xG'(x) + G(x).$$

Thus, (6.54) becomes

$$\frac{1}{x}G(x) - \frac{1}{x} = xG'(x) + G(x) - \frac{1}{1+x},$$

or

$$G'(x) + (\frac{1}{x} - \frac{1}{x^2})G(x) = \frac{1}{x+x^2} - \frac{1}{x^2}. \tag{6.55}$$

Equation (6.55) is a linear first-order differential equation. Unfortunately, it is not easy to solve.

It turns out that the recurrence (6.53) is fairly easy to solve if we use instead of the ordinary generating function the exponential generating function

$$H(x) = \sum_{n=0}^{\infty} D_n \frac{x^n}{n!}.$$

To find $H(x)$, we multiply (6.53) by $x^{n+1}/(n+1)!$ and sum, obtaining

$$\sum_{n=0}^{\infty} D_{n+1}\frac{x^{n+1}}{(n+1)!} = \sum_{n=0}^{\infty}(n+1)D_n\frac{x^{n+1}}{(n+1)!} + \sum_{n=0}^{\infty}(-1)^{n+1}\frac{x^{n+1}}{(n+1)!}. \tag{6.56}$$

The left-hand side of (6.56) is

$$H(x) - D_0 = H(x) - 1.$$

The first term on the right-hand side is

$$\sum_{n=0}^{\infty} D_n\frac{x^{n+1}}{n!} = x\sum_{n=0}^{\infty} D_n\frac{x^n}{n!} = xH(x).$$

The second term on the right-hand side is

$$-\frac{x}{1!} + \frac{x^2}{2!} - \frac{x^3}{3!} + \cdots,$$

which is $e^{-x} - 1$ [see (5.3)]. Equation (6.56) now becomes

$$H(x) - 1 = xH(x) + e^{-x} - 1.$$

Hence,

$$H(x) = \frac{e^{-x}}{1-x}.$$

We may expand this out to obtain D_n, which is the coefficient of $x_n/n!$. Writing $H(x)$ as $e^{-x}(1-x)^{-1}$, we have

$$H(x) = \left[1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \cdots\right]\left[1 + x + x^2 + \cdots\right]. \tag{6.57}$$

It is easy to see directly that

$$H(x) = \sum_{n=0}^{\infty} x^n \left[1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!}\right],$$

so that the coefficient of $x^n/n!$ becomes

$$D_n = n!\left[1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!}\right]. \tag{6.58}$$

Equation (6.58) agrees with our earlier formula (6.14).

Another way to derive D_n from $H(x)$ is to observe that $H(x)$ is the ordinary generating function of the sequence (c_n) which is the convolution of the sequences $((-1)^n/n!)$ and $(1, 1, 1, \ldots)$. Hence, $H(x)$ is the exponential generating function of $(n!c_n)$. Still a third way to derive D_n from $H(x)$ is explained in Exercise 20.

We close this subsection by deriving the recurrence (6.53). Note that, by (6.52),

$$\begin{aligned} D_{n+1} - (n+1)D_n &= D_{n+1} - nD_n - D_n \\ &= nD_{n-1} - D_n \\ &= -\left[D_n - nD_{n-1}\right]. \end{aligned}$$

Thus, we conclude that for all $j \geq 1$ and $k \geq 1$,

$$(-1)^j\left[D_j - jD_{j-1}\right] = (-1)^k\left[D_k - kD_{k-1}\right].$$

Now

$$(-1)^2\left[D_2 - 2D_1\right] = 1\left[1 - 0\right] = 1.$$

Thus, we see that for $n \geq 0$,

$$(-1)^{n+1}\left[D_{n+1} - (n+1)D_n\right] = (-1)^2\left[D_2 - 2D_1\right] = 1,$$

from which (6.53) follows for $n \geq 0$.

6.3.3 Simultaneous Equations for Generating Functions

In Section 6.1.4 we considered a situation where instead of one sequence, we had to use three sequences to find a satisfactory system of recurrences (6.19), (6.21), and (6.23). The method of generating functions can be applied to solve a system of recurrences. To illustrate, let us first choose a_0, b_0, and c_0 so that (6.19), (6.21), and (6.23) hold. Using $a_1 = 2, b_1 = 1, c_1 = 1$, we find from (6.19), (6.21), and (6.23) that

$$\begin{aligned} 2 &= 2a_0 + b_0 + c_0 \\ 1 &= b_0 - c_0 + 1 \\ 1 &= c_0 - b_0 + 1 \end{aligned}$$

One solution to this system is to take $a_0 = 1, b_0 = c_0 = 0$. With these values, we can assume that (6.19), (6.21), and (6.23) hold for $k \geq 0$.

We now multiply both sides of each of our equations by x^k and sum from $k = 0$ to ∞. We get

$$\begin{aligned} \sum_{k=0}^{\infty} a_{k+1} x^k &= 2\sum_{k=0}^{\infty} a_k x^k + \sum_{k=0}^{\infty} b_k x^k + \sum_{k=0}^{\infty} c_k x^k, \\ \sum_{k=0}^{\infty} b_{k+1} x^k &= \sum_{k=0}^{\infty} b_k x^k - \sum_{k=0}^{\infty} c_k x^k + \sum_{k=0}^{\infty} 4^k x^k, \\ \sum_{k=0}^{\infty} c_{k+1} x^k &= \sum_{k=0}^{\infty} c_k x^k - \sum_{k=0}^{\infty} b_k x^k + \sum_{k=0}^{\infty} 4^k x^k. \end{aligned}$$

If

$$A(x) = \textstyle\sum_{k=0}^{\infty} a_k x^k, \quad B(x) = \sum_{k=0}^{\infty} b_k x^k, \quad \text{and} \quad C(x) = \sum_{k=0}^{\infty} c_k x^k$$

are the ordinary generating functions for the sequences $(a_k), (b_k)$, and (c_k), respectively, we find that

$$\begin{aligned} \frac{1}{x}[A(x) - a_0] &= 2A(x) + B(x) + C(x), \\ \frac{1}{x}[B(x) - b_0] &= B(x) - C(x) + \frac{1}{1-4x}, \\ \frac{1}{x}[C(x) - c_0] &= C(x) - B(x) + \frac{1}{1-4x}. \end{aligned}$$

Using $a_0 = 1, b_0 = c_0 = 0$, we see from these three equations that

$$A(x) = \frac{1}{1-2x}\left[xB(x) + xC(x) + 1\right], \tag{6.59}$$

$$B(x) = \frac{1}{1-x}\left[-xC(x) + \frac{x}{1-4x}\right], \tag{6.60}$$

$$C(x) = \frac{1}{1-x}\left[-xB(x) + \frac{x}{1-4x}\right]. \tag{6.61}$$

It is easy to see from (6.60) and (6.61) that

$$B(x) = C(x) = \frac{x}{1-4x}. \tag{6.62}$$

It then follows from (6.59) and (6.62) that

$$A(x) = \frac{2x^2 - 4x + 1}{(1-2x)(1-4x)}. \tag{6.63}$$

By using (6.42), we see that (6.62) implies that

$$B(x) = C(x) = \sum_{k=0}^{\infty} 4^k x^{k+1}.$$

Thus, $b_k = c_k = 4^{k-1}$ for $k > 0$, $b_k = c_k = 0$ for $k = 0$. The right-hand side of (6.63) can be expanded out using the method of partial fractions, and we obtain

$$A(x) = \frac{1-3x}{1-4x} + \frac{x}{1-2x}.$$

This can be rewritten as

$$\begin{aligned} A(x) &= 1 + \frac{x}{1-4x} + \frac{x}{1-2x} \\ &= 1 + \sum_{k=0}^{\infty} 4^k x^{k+1} + \sum_{k=0}^{\infty} 2^k x^{k+1}. \end{aligned}$$

Thus, $a_k = 4^{k-1} + 2^{k-1}$ for $k > 0$, and $a_0 = 1$. The results can readily be checked. In particular, we have $a_2 = 4 + 2 = 6$, which agrees with the result obtained in Section 6.1.4.

EXERCISES FOR SECTION 6.3

Note to the reader: In each of these exercises, if the denominator of the generating function turns out to have complex roots, it is acceptable to give the generating function as the answer.

1. Use generating functions to solve the following recurrences.

   (a) (6.5) in Example 6.2 under the initial condition $f(1) = 0$

   (b) (6.6) in Example 6.3 (c) (6.7) in Example 6.3

2. Use generating functions to solve the following recurrences under the given initial conditions.

   (a) $a_{k+1} = a_k + 3, a_0 = 1$ (b) $a_{k+1} = 3a_k + 2, a_1 = 1$

   (c) $a_{k+2} = a_{k+1} - 2a_k, a_0 = 0, a_1 = 1$

3. Use generating functions to solve each of the recurrences in Exercise 12, Section 6.2.

4. In each of the following cases, suppose that $G(x)$ is the ordinary generating function for a sequence (a_k). Find a_k.

   (a) $G(x) = \dfrac{1}{(1-x)(1-3x)}$ (b) $G(x) = \dfrac{2x+1}{(1-3x)(1-2x)}$

   (c) $G(x) = \dfrac{2x^2}{(1-3x)(1-5x)(1-7x)}$ (d) $G(x) = \dfrac{1}{4x^2-5x+1}$

   (e) $G(x) = \dfrac{x}{x^2-5x+6}$ (f) $G(x) = \dfrac{1}{8x^3-6x^2+x}$

5. Use the results of Section 6.3.3 to verify the values we obtained in Section 6.1.4 for:

   (a) a_3 (b) b_3 (c) c_3

6. Consider the product sales problem of Exercise 18, Section 6.1. Suppose that $A(x) = \sum_{n=0}^{\infty} a_n x^n$ is the ordinary generating function for (a_n).

   (a) If $a_0 = 0$ and $a_1 = 1$, find $A(x)$.

   (b) In general, find $A(x)$ in terms of a_0 and a_1.

   (c) If $a_0 = a_1$, use your answer to part (b) to show that a_n is constant. (This is obvious from the recurrence.)

7. In Exercise 21 of Section 6.1, find b_n.

8. This exercise asks the reader to derive the formula for the kth Fibonacci number by the method of generating functions.

   (a) It is useful to define F_{-1}. Use the recurrence (6.34) and the values of F_1 and F_0 to derive an appropriate value for F_{-1}.

   (b) If $G(x) = \sum_{k=0}^{\infty} F_k x^k$ is the ordinary generating function for F_k, derive a functional equation for $G(x)$ by multiplying the recurrence (6.34) by x^k and summing from $k = 0$ to ∞.

   (c) Show that
   $$G(x) = \frac{1}{1 - x - x^2}.$$

   (d) Find the roots of $1-x-x^2$ and use them to write $1-x-x^2$ as $(1-\lambda x)(1-\mu x)$.

   (e) Use partial fractions to write
   $$\frac{1}{(1-\lambda x)(1-\mu x)} \text{ as } \frac{A}{1-\lambda x} + \frac{B}{1-\mu x}.$$

   (f) From the result in part (e), derive a formula for F_k.

9. Suppose that G_n satisfies the equation
   $$G_{n+1} = G_n + G_{n-1},$$
   $n \geq 1$. Suppose that $G_0 = 3$ and $G_1 = 4$. Use generating functions to find a formula for G_n.

10. Use generating functions to solve the recurrence (6.10) in Example 6.7.

11. Solve the following recurrence:
    $$a_{k+1} = 2a_k + k + 5, \quad k \geq 0, \quad a_0 = 0.$$

12. Suppose that $a_{n+1} = (n+1)b_n$, for $n \geq 0$, and $a_0 = 0$. Find a relation involving $A(x), B'(x)$, and $B(x)$, if $A(x)$ and $B(x)$ are the ordinary generating functions for (a_n) and (b_n), respectively.

13. Suppose that
    $$y_{k+1} = Ay_k + B,$$
    for $k \geq 0$, where A and B are real numbers, $A \neq 1$. Find a formula for y_k in terms of y_0 using the method of generating functions.

14. Suppose that
$$y_{k+2} - y_{k+1} + 2y_k = 4^k,$$
$k \geq 0$, and that $y_0 = 2$, $y_1 = 1$. Find y_k using the method of generating functions.

15. Suppose that Y_t is national income at time t. Following Samuelson [1939], Goldberg [1958] derives the recurrence relation
$$Y_t = \alpha(1+\beta)Y_{t-1} - \alpha\beta Y_{t-2} + 1,$$
$t \geq 2$, for α and β positive constants. Assuming that $Y_0 = 2, Y_1 = 3, \alpha = \frac{1}{2}$, and $\beta = 1$, find a generating function for the sequence (Y_t).

16. Repeat Exercise 15 for $\alpha = 2$ and $\beta = 4$.

17. (Goldberg [1958]) In his work on inventory cycles, Metzler [1941] studies the total income i_t produced in the tth time period by an entrepreneur who is producing goods for sales and for inventory. Metzler derives the recurrence
$$i_{t+2} - 2\beta i_{t+1} + \beta i_t = v_0,$$
$t \geq 0$, where β is a constant such that $0 < \beta < 1$ and v_0 is a positive constant. Assuming that $i_0 = i_1 = 0$, find a generating function for the sequence (i_t).

18. If
$$C_{n+1} = 2nC_n + 2C_n + 2,$$
$n \geq 0$, and $C_0 = 1$, find C_n.

19. Solve the recurrence derived in Exercise 37, Section 6.1.

20. Derive D_n from Equation (6.57) for $H(x)$ by observing that $H(x)$ is the product of the exponential generating functions for the sequences (a_k) and (b_k), where $a_k = (-1)^k$ and $b_k = k!$. Use your results from Exercise 18, Section 5.5.

21. Derive a formula for D_n as follows.

    (a) Let
    $$C_n = \frac{D_n}{n!} - \frac{D_{n-1}}{(n-1)!}.$$
    Find a recurrence for C_{n+1} in terms of C_n.

    (b) Solve the recurrence for C_n by iteration.

    (c) Use the formula for C_n to solve for D_n.

22. Solve the recurrences of Exercise 22, Section 6.1, by the method of Section 6.3.3.

23. Solve simultaneously the recurrences
$$\begin{aligned} a_{n+1} &= a_n + b_n + c_n, \quad n \geq 1, \\ b_{n+1} &= 4^n - c_n, \quad n \geq 1, \\ c_{n+1} &= 4^n - b_n, \quad n \geq 1, \end{aligned}$$
subject to the initial conditions $a_1 = b_1 = c_1 = 1$.

24. (Anderson [1974]) Suppose that (a_n) satisfies
$$na_n = 2(a_{n-1} + a_{n-2}),$$
$n \geq 2$, and $a_0 = e, a_1 = 2e$. Let $A(x)$ be the ordinary generating function for (a_n).

(a) Show that $A'(x) = 2(1 + x)A(x)$.

(b) Find $A(x)$. [*Hint:* Recall the equation $f'(x) = f(x)$.]

25. Consider a linear homogeneous recurrence relation (6.25) with constant coefficients. This exercise explores the relationship between the solution using characteristic roots and the solution using generating functions.

(a) Show that the ordinary generating function $G(x)$ for the sequence (a_n) is given by $G(x) = p(x)/q(x)$, where

$$q(x) = 1 - c_1 x - c_2 x^2 - \cdots - c_p x^p$$

and

$$\begin{aligned} p(x) &= a_0 + (a_1 - c_1 a_0)x + (a_2 - c_1 a_1 - c_2 a_0)x^2 + \cdots \\ &\quad + (a_{p-1} - c_1 a_{p-2} - \cdots - c_{p-1} a_0)x^{p-1}. \end{aligned}$$

(b) Show that if $\alpha_1, \alpha_2, \ldots, \alpha_p$ are the characteristic roots, then

$$q(x) = (1 - \alpha_1 x)(1 - \alpha_2 x) \cdots (1 - \alpha_p x)$$

and the roots of $q(x)$ are $1/\alpha_1, 1/\alpha_2, \ldots, 1/\alpha_p$.

(c) Illustrate these results by using the method of generating functions to solve the recurrence (6.29), and compare to the results in Section 6.2.1.

(d) Illustrate these results by using the method of characteristic roots to solve the recurrence (6.1), and compare to the results in Section 6.3.1.

26. Compute $R(x, B_J)$ for board B_J of Figure 5.6 by using Equation (6.50).

27. Compute the rook polynomial of board B in Figure 6.7 by using the results of Exercise 17 of Section 5.1, Example 5.14, and Equation (6.50).

6.4 SOME RECURRENCES INVOLVING CONVOLUTIONS[12]

6.4.1 The Number of Simple, Ordered, Rooted Trees

In Section 3.5.6 we noted that Cayley reduced the problem of counting the saturated hydrocarbons to the problem of counting trees. Here, we discuss a related problem, the problem of counting the number of *simple, ordered, rooted trees*, or *SOR trees* for short. These are (unlabeled) rooted trees[13] which are simple in the sense that each vertex has zero, one, or two children. Also, they are ordered so that the children of each vertex are labeled left (L) or right (R). We distinguish two SOR trees if they are not isomorphic, or if they have different roots, or if they have the same root and are isomorphic, but there is a disagreement on left or right children. For instance, the two SOR trees of Figure 6.8 are considered different even though they are isomorphic and have the same root.

[12] The four subsections of this section are relatively independent and can, in principle, be read in any order. From a purely pedagogical viewpoint, if there is not enough time for all four subsections, one of Sections 6.4.1, 6.4.2, 6.4.3 should be read—6.4.1 would be best—and then Section 6.4.4 should be read.

[13] For the definition of rooted tree, see Section 3.6.1.

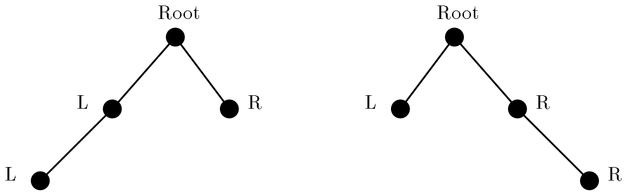


Figure 6.8: Two distinct SOR trees.

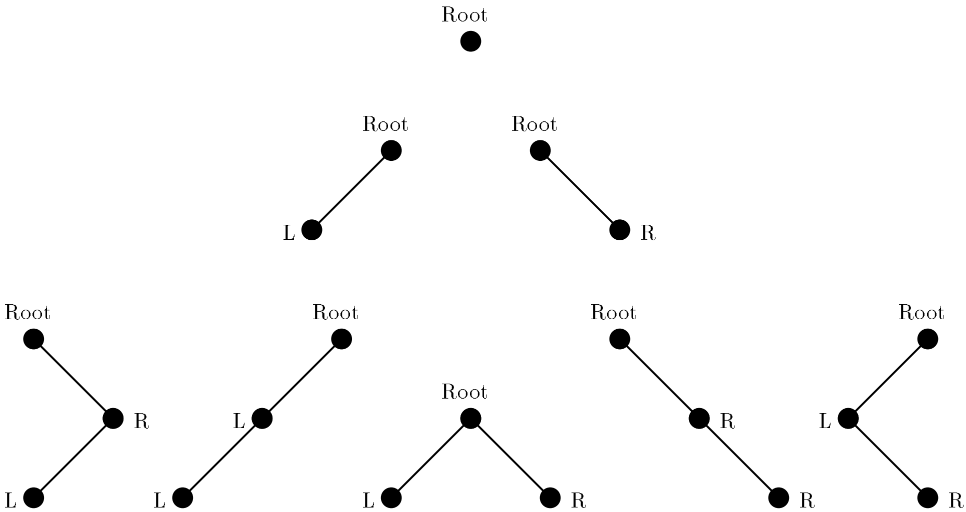


Figure 6.9: The distinct SOR trees of one, two, and three vertices.

We shall let u_n be the number of distinct SOR trees of n vertices. Then Figure 6.9 shows that $u_1 = 1, u_2 = 2$, and $u_3 = 5$. It is convenient to count the tree with no vertices as an SOR tree. Thus, we have $u_0 = 1$.

Suppose that T is an SOR tree of $n+1$ vertices, $n \geq 0$. Then the root has at most two children. If vertices a and b are the left and right children of the root in an SOR tree, then a and b themselves form the roots of SOR trees T_L and T_R, respectively. (If a or b does not exist, the corresponding SOR tree is the tree with no vertices.) In particular, if T_R has r vertices, T_L has $n - r$ vertices. Thus, we have the following recurrence:

$$u_{n+1} = u_0 u_n + u_1 u_{n-1} + u_2 u_{n-2} + \cdots + u_n u_0, \tag{6.64}$$

$n \geq 0$. Equation (6.64) gives us a way of computing u_{n+1} knowing all previous values u_i, $i \leq n$.

Note that the right-hand side of (6.64) comes from a convolution. In particular, if the sequence (v_n) is defined to be the sequence $(u_n) * (u_n)$, then

$$u_{n+1} = v_n. \tag{6.65}$$

Let $U(x) = \sum_{n=0}^{\infty} u_n x^n$ and $V(x) = \sum_{n=0}^{\infty} v_n x^n$ be the ordinary generating functions for the sequences (u_n) and (v_n), respectively. Then by (6.65),

$$\sum_{n=0}^{\infty} u_{n+1} x^n = \sum_{n=0}^{\infty} v_n x^n.$$

We conclude that

$$\frac{1}{x}\left[U(x) - u_0\right] = V(x),$$

so

$$\frac{1}{x}\left[U(x) - 1\right] = V(x). \tag{6.66}$$

But $V(x) = U(x)U(x) = U^2(x)$, so (6.66) gives us

$$xU^2(x) - U(x) + 1 = 0. \tag{6.67}$$

Equation (6.67) is a functional equation for $U(x)$. We can solve this functional equation by treating the unknown $U(x)$ as a variable y. Then, assuming that $x \neq 0$, we apply the quadratic formula to the equation

$$xy^2 - y + 1 = 0$$

and solve for y to obtain

$$y = U(x) = \frac{1 \pm \sqrt{1-4x}}{2x}. \tag{6.68}$$

We can now solve for u_n by expanding out. In particular, we note that $\sqrt{1-4x}$ can be expanded out using the binomial theorem (Theorem 5.3), giving us

$$\begin{aligned}(1-4x)^{1/2} \;&=\; 1 + \frac{1}{2}(-4x) + \frac{\frac{1}{2}\left(-\frac{1}{2}\right)}{2!}(-4x)^2 + \frac{\frac{1}{2}\left(-\frac{1}{2}\right)\left(-\frac{3}{2}\right)}{3!}(-4x)^3 + \\ &\cdots + \binom{\frac{1}{2}}{r}(-4x)^r + \cdots.\end{aligned}$$

For $n \geq 1$, the coefficient of x^n here can be written as

$$\begin{aligned}
\binom{\frac{1}{2}}{n}(-4)^n &= \frac{\left(\frac{1}{2}\right)\left(-\frac{1}{2}\right)\left(-\frac{3}{2}\right)\cdots\left(\frac{1}{2}-n+1\right)}{n!}(-4)^n \\
&= \frac{\left(\frac{1}{2}\right)\left(-\frac{1}{2}\right)\left(-\frac{3}{2}\right)\cdots\left[-\frac{2n-3}{2}\right](-4)^n}{n!} \\
&= \frac{\left(\frac{1}{2}\right)(-1)^{n-1}\left(\frac{1}{2}\right)\left(\frac{3}{2}\right)\cdots\left[\frac{2n-3}{2}\right](-1)^n\,4^n}{n!} \\
&= \frac{\left(-\frac{1}{2^n}\right)[1\cdot 3\cdot 5\cdot \ldots \cdot (2n-3)]\,4^n}{n!} \\
&= \frac{-2^n\,[1\cdot 3\cdot 5\cdot \ldots \cdot (2n-3)]}{n!} \\
&= \frac{-2}{n}\frac{2^{n-1}}{(n-1)!}[1\cdot 3\cdot 5\cdot \ldots \cdot (2n-3)] \\
&= \frac{-2}{n}\frac{2^{n-1}\,[1\cdot 3\cdot 5\cdot \ldots \cdot (2n-3)]\,(n-1)!}{(n-1)!(n-1)!} \\
&= \frac{-2}{n}\frac{[1\cdot 3\cdot 5\cdot \ldots \cdot (2n-3)][2\cdot 4\cdot 6\cdot \ldots \cdot (2n-2)]}{(n-1)!(n-1)!} \\
&= \frac{-2}{n}\frac{(2n-2)!}{(n-1)!(n-1)!} \\
&= \frac{-2}{n}\binom{2n-2}{n-1}.
\end{aligned}$$

Thus,

$$(1-4x)^{1/2} = 1 - \sum_{n=1}^{\infty}\frac{2}{n}\binom{2n-2}{n-1}x^n. \tag{6.69}$$

Now (6.68) has two signs, that is, two possible solutions. If we take the solution of (6.68) with the $-$ sign, we have

$$U(x) = \frac{1}{2x}\left[1-\sqrt{1-4x}\right],$$

so

$$\begin{aligned}
U(x) &= \frac{1}{2x}\left[\sum_{n=1}^{\infty}\frac{2}{n}\binom{2n-2}{n-1}x^n\right], \\
&= \sum_{n=1}^{\infty}\frac{1}{n}\binom{2n-2}{n-1}x^{n-1},
\end{aligned}$$

$$= \sum_{n=0}^{\infty} \frac{1}{n+1} \binom{2n}{n} x^n, \tag{6.70}$$

by replacing n with $n+1$. We conclude from (6.70) that

$$u_n = \frac{1}{n+1} \binom{2n}{n}. \tag{6.71}$$

If we take the solution of (6.68) with the + sign, we find similarly that

$$U(x) = \frac{1}{x} - \sum_{n=1}^{\infty} \frac{1}{n} \binom{2n-2}{n-1} x^{n-1}, \tag{6.72}$$

so for $n \geq 1$,

$$u_n = -\frac{1}{n+1} \binom{2n}{n}. \tag{6.73}$$

Now the coefficients u_n must be nonnegative (why?), so (6.71) must be the solution, not (6.73). We also see that we must take the $-$ sign in (6.68) to get $U(x)$. We can see this directly from (6.72) also, since if $U(x)$ were given by (6.72), $U(x)$ would have a term $1/x$, yet $U(x) = \sum_{n=0}^{\infty} u_n x^n$.

The numbers u_n defined by (6.71) are called the *Catalan numbers*, after Eugene Charles Catalan. For instance, we find that

$$u_0 = \frac{1}{1}\binom{0}{0} = 1, \qquad u_1 = \frac{1}{2}\binom{2}{1} = 1, \qquad u_2 = \frac{1}{3}\binom{4}{2} = 2,$$

$$u_3 = \frac{1}{4}\binom{6}{3} = 5, \qquad u_4 = \frac{1}{5}\binom{8}{4} = 14.$$

The first four results agree with our earlier computations and the fifth can readily be verified. We shall see that the Catalan numbers are very common in combinatorics. See Eggleton and Guy [1988] for an extensive list of different contexts where the Catalan numbers appear.

6.4.2 The Ways to Multiply a Sequence of Numbers in a Computer

Suppose that we are given a sequence of n numbers, $x_1, x_2, \ldots, x_n$, and we wish to find their product. There are various ways in which we can find the product. For instance, suppose that $n = 4$. We can first multiply x_1 and x_2, then this product by x_3, and then this product by x_4. Alternatively, we can begin by multiplying x_1 and x_2, then multiply x_3 and x_4, and, finally, multiply the two products. We can distinguish these two and other approaches by inserting parentheses[14] as appropriate

[14] The parentheses do not distinguish between first performing x_1x_2, then x_3x_4, and multiplying the product, and first performing x_3x_4, then x_1x_2, and multiplying the product. We are only concerned with what products will have to be calculated.

in the string $x_1x_2\cdots x_n$. Thus, the first method corresponds to

$$(((x_1x_2)x_3)x_4)$$

and the second to

$$((x_1x_2)(x_3x_4)).$$

Let us assume that we must perform multiplications in the order given. For example, we do not allow multiplying x_1 by x_3 directly, and so on. Suppose that we are given a sequence of n numbers. How many different ways are there to instruct a computer to find the product? Suppose that P_n represents the number of ways in question. It is easy to see that finding the product corresponds to inserting $n-1$ left and $n-1$ right parentheses into the sequence $x_1x_2\cdots x_n$ in such a way that

1. one never has parentheses around a single term [i.e., (x_i) is not allowed], and
2. as we go from left to right, the number of right parentheses never exceeds the number of left parentheses.

Note that $P_1 = 1$, for there is only one way to insert 0 left and right parentheses. Also, $P_2 = 1, P_3 = 2$, and $P_4 = 5$. Table 6.4 demonstrates the parenthesizations corresponding to these numbers. It is easy to find a recurrence for P_n. Suppose that $n \geq 2$. Consider the last multiplication performed. This involves the product of two subproducts, $x_1\cdots x_r$ and $x_{r+1}\cdots x_n$. That is, we have for $1 < r < n-1$, the multiplication

$$((x_1\cdots x_r)(x_{r+1}\cdots x_n)).$$

If $r = 1$ or $n-1$, we have

$$(x_1(x_2\cdots x_n)) \quad \text{or} \quad ((x_1\cdots x_{n-1})x_n).$$

In either case, there are P_r ways to find the first subproduct and P_{n-r} ways to find the second subproduct, so we obtain the recurrence

$$P_n = \sum_{r=1}^{n-1} P_r P_{n-r}, \tag{6.74}$$

$n \geq 2$. Now if we let $P_0 = 0$, (6.74) becomes

$$P_n = \sum_{r=0}^{n} P_r P_{n-r}, \tag{6.75}$$

$n \geq 2$. Now let $P(x) = \sum_{n=0}^{\infty} P_n x^n$ be the ordinary generating function for the sequence (P_n). Equation (6.75) suggests that (P_n) is related to the convolution $(P_n) * (P_n)$. However, since (6.75) holds only for $n \geq 2$, we cannot conclude that $P(x) = P^2(x)$. To get around this difficulty, we define the sequence (Q_n) to be the sequence $(P_n) * (P_n)$. Then note that

$$Q_n = \begin{cases} 0 = P_0P_0 & \text{if } n = 0 \\ 0 = P_0P_1 + P_1P_0 & \text{if } n = 1 \\ P_n & \text{if } n \geq 2 \end{cases} \tag{6.76}$$

Table 6.4: The Ways of Performing a Multiplication of Two, Three, or Four Numbers

P_2	P_3	P_4
(x_1x_2)	$((x_1x_2)x_3)$	$(((x_1x_2)x_3)x_4)$
	$(x_1(x_2x_3))$	$(x_1(x_2(x_3x_4)))$
		$((x_1(x_2x_3))x_4)$
		$(x_1((x_2x_3)x_4))$
		$((x_1x_2)(x_3x_4))$

and the ordinary generating function $Q(x) = \sum_{n=0}^{\infty} Q_n x^n$ satisfies

$$Q(x) = P^2(x).$$

Moreover, by (6.76),

$$Q(x) = P(x) - x,$$

since $P_n = Q_n$ for $n \neq 1$, and $P_1 = 1, Q_1 = O$. Thus, we know that

$$P(x) - x = P^2(x).$$

This is a functional equation for P(x). We solve it by rewriting it as a quadratic in the unknown $y = P(x)$ and using the quadratic formula, obtaining

$$P^2(x) - P(x) + x = 0,$$

$$P(x) = \frac{1 \pm \sqrt{1-4x}}{2}. \tag{6.77}$$

To find P_n, we could expand out $P(x)$ using the binomial theorem. Alternatively, we recognize that $P(x)$ is $xU(x)$ for $U(x)$ of (6.68). Thus, $P_n = u_{n-1}$ for $n \geq 1$. We have defined $P_0 = 0$. By formula (6.71), we find that for $n \geq 1$,

$$P_n = \frac{1}{n}\binom{2n-2}{n-1}. \tag{6.78}$$

The Catalan numbers have shown up again.

The close relation between the numbers u_n and P_n suggests that we might be able to find a direct relationship between SOR trees and order of multiplication. Following Even [1973], we shall describe such a relationship. Let us consider just a sequence of n left and n right parentheses. Such a sequence is called *well-formed* if condition 2 above holds. Let K_n be the number of such sequences. Then clearly $P_n = K_{n-1}$. Given an SOR tree of n vertices, associate with each vertex of degree 1 the sequence of parentheses ( ). Associate with every other vertex the following sequence: (, followed by the sequence associated with its left child (if there is one),

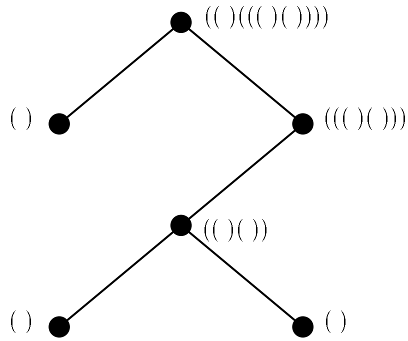


Figure 6.10: Next to each vertex is the corresponding well-formed sequence of parentheses.

followed by the sequence associated with its right child (if there is one), followed by). This associates a unique well-formed sequence of parentheses with each SOR tree, the sequence assigned to its root. Figure 6.10 illustrates the procedure. Conversely, given a well-formed sequence of n left and n right parentheses, one can show that it comes from an SOR tree of n vertices. This is left as an exercise (Exercise 17). Thus, $K_n = u_n$, and we again have the conclusion $P_n = u_{n-1}$.

6.4.3 Secondary Structure in RNA

In Sections 2.11 and 2.12 we studied the linear chain of bases in an RNA molecule. This chain is sometimes said to define the *primary structure* of RNA. When RNA has only the bonds between neighbors in the chain, it is said to be a *random coil.* Now RNA does not remain a random coil. It folds back on itself and forms new bonds referred to as *Watson-Crick bonds*, creating helical regions. In such Watson-Crick bonding of an RNA chain $s = s_1s_2\cdots s_n$, each base can be bonded to at most one other nonneighboring base and if s_i and s_j are bonded, and $i < k < j$, then s_k can only be bonded with bases between s_{i+1} and s_{j-1}; that is, there is no crossover.[15] The new bonds define the *secondary structure* of the original RNA chain. Figure 6.11 shows one possible secondary structure for the RNA chain

AACGGGCGGGACCCUUCAACCCUU.

Watson-Crick bonds usually form between A and U bases or between G and C bases, but we shall, following Howell, Smith, and Waterman [1980], find it convenient to allow all possible bonds in our discussion. In studying RNA chains, Howell, Smith, and Waterman [1980] use recurrences to compute the number R_n of possible secondary structures for an RNA chain of length n. We briefly discuss their approach.[16] For more on secondary structure, see Clote and Backofen [2000], Setubal and Meidanis [1997], and Waterman [1995].

[15] For a related bonding problem, see Nussinov, *et al.* [1978].

[16] For related work, see Stein and Waterman [1978] and Waterman [1978].

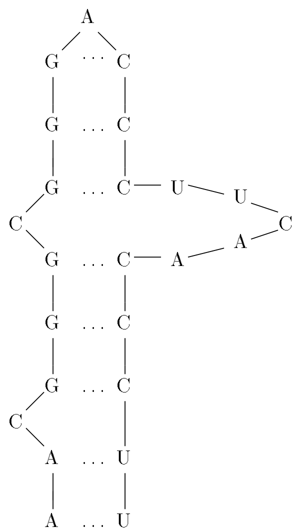


Figure 6.11: A secondary structure for the RNA chain AACGGGCGGGA-CCCUUCAACCCUU. The Watson-Crick bonds are dotted. (Reproduced from Howell, Smith, and Waterman [1980]. Copyright ©1980 Society for Industrial and Applied Mathematics. Reprinted with permission. All rights reserved.)

Note that $R_1 = R_2 = 1$, for there can be no Watson-Crick bonds. Also, $R_0 = 1$ by convention. Consider an RNA chain $s_1 s_2 \cdots s_{n+1}$ of length $n+1 \geq 3$. Now either s_{n+1} is not Watson-Crick bonded, or it is bonded with s_j, $1 \leq j \leq n-1$. Thus, for $n \geq 2$,

$$R_{n+1} = R_n + \sum_{j=1}^{n-1} R_{j-1} R_{n-j}, \tag{6.79}$$

since in the first case, the chain $s_1 s_2 \cdots s_n$ is free to form any secondary structure, and in the second case, the subchains $s_1 s_2 \cdots s_{j-1}$ and $s_{j+1} s_{j+2} \cdots s_n$ are free to form any secondary structure. This is a recurrence for R_n. Let $R(x)$ be the ordinary generating function for R_n, that is,

$$R(x) = \sum_{n=0}^{\infty} R_n x^n.$$

To find $R(x)$, we note that the second term on the right-hand side of (6.79) almost arises from a convolution. If we define $(T_n) = (R_n) * (R_n)$, and we use the fact that $R_0 = 1$, we have for $n \geq 2$,

$$R_{n+1} = R_n + \sum_{j=1}^{n} R_{j-1} R_{n-j} - R_{n-1} R_0,$$

and hence for $n \geq 2$,

$$R_{n+1} = R_n - R_{n-1} + T_{n-1}. \tag{6.80}$$

It is easy to see that (6.80) still holds for $n = 1$. Furthermore, if we define $R_{-1} = T_{-1} = 0$, (6.80) holds for all $n \geq 0$. Now let $T(x) = \sum_{n=0}^{\infty} T_n x^n$ be the ordinary generating function for (T_n). By (6.80),

$$\sum_{n=0}^{\infty} R_{n+1}x^n = \sum_{n=0}^{\infty} R_n x^n - \sum_{n=0}^{\infty} R_{n-1}x^n + \sum_{n=0}^{\infty} T_{n-1}x^n.$$

Hence,

$$\frac{1}{x}\left[R(x) - R_0\right] = R(x) - x\left[\frac{R_{-1}}{x} + R(x)\right] + x\left[\frac{T_{-1}}{x} + T(x)\right],$$

so

$$\frac{1}{x}[R(x) - 1] = R(x) - xR(x) + xT(x).$$

Since (T_n) is a convolution of (R_n) with itself,

$$T(x) = R^2(x).$$

Thus, we have

$$\frac{1}{x}R(x) - \frac{1}{x} = R(x) - xR(x) + xR^2(x),$$

or

$$x^2R^2(x) + (-x^2 + x - 1)R(x) + 1 = 0.$$

We find, for $x \neq 0$,

$$R(x) = \frac{x^2 - x + 1 \pm \sqrt{(-x^2 + x - 1)^2 - 4x^2}}{2x^2},$$

or

$$R(x) = \frac{1}{2x^2}\left[x^2 - x + 1 \pm \sqrt{1 - (2x + x^2 + 2x^3 - x^4)}\right]. \tag{6.81}$$

The square root in (6.81) can be expanded using the binomial theorem. Note that we can easily determine whether the + or − sign is to be used in (6.81) by noting that if the + sign is used, there is a term $1/2x^2$. Thus, the − sign must be right. We can also see this by considering what happens as x approaches 0. Now $R(x)$ should approach $R(0) = R_0 = 1$. If we use the + sign in (6.81), $R(x)$ approaches ∞ as x approaches 0. Thus, the − sign must be right.

6.4.4 Organic Compounds Built Up from Benzene Rings

Harary and Read [1970] and Anderson [1974] point out that certain organic compounds built up from benzene rings can be represented by a configuration of hexagons, as for example in Figure 6.12. Counting hexagonal configurations of various kinds is a central topic in mathematical chemistry. (For a survey of this topic, see

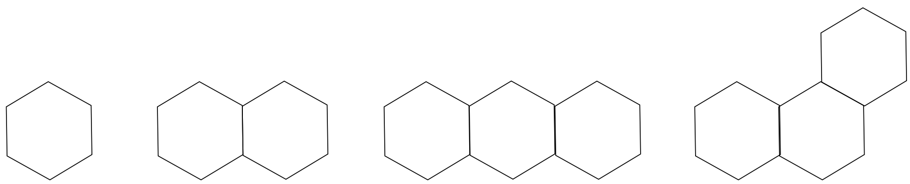


Figure 6.12: Some organic compounds built up from benzene rings.

Cyvin, Brunvoll, and Cyvin [1992].) We illustrate the idea with a counting method using generating functions.

We shall be interested in *polyhexes*, configurations of hexagons in which all hexagons are congruent, regular hexagons, any two such hexagons are either disjoint or share a common edge, and the configuration is connected if we think of it as a graph. We add the restriction that no three hexagons can meet at a point. Then we get *catacondensed polyhexes*.

Following Harary and Read [1970], we consider only catacondensed polyhexes generated by starting with a catacondensed polyhex and adding one hexagon at one of the edges along the "perimeter" of the polyhex. In particular, we make the simplifying assumption that all polyhexes are generated starting from a base hexagon with a base edge as shown in Figure 6.13. Onto this base, one can attach a hexagon only at the sides labeled 1, 2, and 3. Thus, by our previous assumptions, one cannot attach a hexagon at both edge 1 and 2 or at both edge 2 and 3. Let us call a catacondensed polyhex constructed this way with n hexagons in all a *polyhexal configuration*, or an *n-polyhex* for short. In general, an n-polyhex (a polyhex of n hexagons) is obtained from an $(n-1)$-polyhex by attaching a new hexagon to one of the edges of the $(n-1)$-polyhex. This nth hexagon and the $(n-1)$-polyhex can only have a single edge in common. Let h_k denote the number of possible k-polyhexes. We wish to compute h_k. Rather than derive a recurrence for h_k directly, we introduce two other sequences and use these to compute h_k. Let s_k denote the number of such configurations where only one hexagon is joined to the base, and d_k denote the number where exactly two hexagons are joined to the base. Obviously,

$$h_k = s_k + d_k, \tag{6.82}$$

$k \geq 2$. However, (6.82) fails for $k = 1$, since $h_1 = 1$ and $s_1 = d_1 = 0$. If we have a $(k+1)$-polyhex and only one hexagon is joined to the base, there are three possible edges on which to join it, so

$$s_{k+1} = 3h_k, \tag{6.83}$$

$k \geq 1$. If we have a $(k+1)$-polyhex and two hexagons are joined to the base, they must use edges 1 and 3 of Figure 6.13, since three hexagons may not meet in a point. Thus, a r-polyhex is joined to edge 1 and a $k-r$-polyhex is joined to edge 3, with $1 \leq r \leq k-1$. We conclude that

$$d_{k+1} = h_1 h_{k-1} + h_2 h_{k-2} + \cdots + h_{k-1} h_1 \tag{6.84}$$

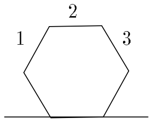


Figure 6.13: A base hexagon.

for $k \geq 2$. Note that the three recurrences (6.82), (6.83), and (6.84) can be used together to compute the desired numbers h_k iteratively. Knowing $h_1, h_2, \ldots, h_k$, we use (6.83) and (6.84) to compute s_{k+1} and d_{k+1}, respectively, and then obtain h_{k+1} from (6.82). This situation is analogous to the situation in Section 6.1.4, where we first encountered a system of recurrences.

Consider now the ordinary generating functions

$$H(x) = \sum_{k=0}^{\infty} h_k x^k, \quad S(x) = \sum_{k=0}^{\infty} s_k x^k, \quad D(x) = \sum_{k=0}^{\infty} d_k x^k. \tag{6.85}$$

We shall compute $H(x)$. The technique for finding it will be somewhat different from that in the previous subsections. It will make use of the results of Section 5.2 on operating on generating functions. Note that h_k, s_k, and d_k are all defined only from $k = 1$ to ∞. To use the generating functions of (6.85), it is convenient to define $h_0 = s_0 = d_0 = 0$ and so to be able to take the sums from 0 to ∞. If we take $h_0 = s_0 = d_0 = 0$, then (6.82) holds for $k = 0$ as well as $k \geq 2$. Also, since $s_1 = 0$ and $h_0 = 0$, (6.83) now holds for all $k \geq 0$. Since $h_0 = 0$, we can add $h_0 h_k + h_k h_0$ to the right-hand side of (6.84), obtaining

$$d_{k+1} = h_0 h_k + h_1 h_{k-1} + h_2 h_{k-2} + \cdots + h_k h_0, \tag{6.86}$$

for all $k \geq 2$. But it is easy to see that (6.86) holds for all $k \geq 0$, since $d_1 = 0 = h_0 h_0$ and $d_2 = 0 = h_0 h_1 + h_1 h_0$.

Using (6.82), we are tempted to conclude, by the methods of Section 5.2, that

$$H(x) = S(x) + D(x).$$

However, this is not true since (6.82) is false for $k = 1$. If we define

$$g_k = \begin{cases} h_k & \text{if } k \neq 1 \\ 0 & \text{if } k = 1, \end{cases}$$

then

$$g_k = s_k + d_k$$

holds for all $k \geq 0$. Moreover, if

$$G(x) = \sum_{k=0}^{\infty} g_k x^k,$$

then

$$G(x) = S(x) + D(x).$$

Finally,

$$H(x) = G(x) + x \tag{6.87}$$

since the sequence (h_k) is the sum of the sequence (g_k) and the sequence (0, 1, 0, 0, ...). Thus,

$$H(x) = S(x) + D(x) + x. \tag{6.88}$$

Next, we have observed that (6.83) holds for $k \geq 0$. Hence,

$$\sum_{k=0}^{\infty} s_{k+1} x^k \;=\; 3 \sum_{k=0}^{\infty} h_k x^k,$$

or

$$\frac{1}{x}\left[S(x) - s_0\right] = 3H(x),$$

and since $s_0 = 0$,

$$\frac{1}{x} S(x) = 3H(x).$$

Thus,

$$S(x) = 3xH(x). \tag{6.89}$$

Finally, let us simplify $D(x)$. Letting $e_k = d_{k+1}, k \geq 0$, we see from (6.86) that (e_k) is the convolution of the sequence (h_k) with itself. Letting

$$E(x) = \sum_{k=0}^{\infty} e_k x^k,$$

we have

$$E(x) = H^2(x). \tag{6.90}$$

Then

$$\begin{aligned} D(x) &= \sum_{k=0}^{\infty} d_k x^k &&= \sum_{k=1}^{\infty} d_k x^k \\ &= \sum_{k=1}^{\infty} e_{k-1} x^k &&= x \sum_{k=1}^{\infty} e_{k-1} x^{k-1} \\ &= x \sum_{k=0}^{\infty} e_k x^k &&= xE(x). \end{aligned}$$

Thus, by (6.90),

$$D(x) = xH^2(x). \tag{6.91}$$

Using (6.89) and (6.91) in (6.88) gives

$$H(x) = 3xH(x) + xH^2(x) + x,$$

or

$$xH^2(x) + (3x - 1)H(x) + x = 0. \tag{6.92}$$

Equation (6.92) is a quadratic equation for the unknown function $H(x)$. We solve it by the quadratic formula, obtaining

$$H(x) = \frac{1}{2x}\left[1 - 3x \pm \sqrt{(3x-1)^2 - 4x^2}\right]$$

or

$$H(x) = \frac{1}{2x}\left[1 - 3x \pm \sqrt{1 - (6x - 5x^2)}\right], \tag{6.93}$$

$x \neq 0$. $H(x)$ can be expanded out using the binomial theorem, and the proper sign, $+$ or $-$, can be chosen once the expansion has been obtained.

The method we have described for computing h_k is, unfortunately, flawed. It is possible to use the method of building up to a k-polyhex that we have described and end up with a configuration which has three hexagons meeting at a point, i.e., that violates the catacondensed condition. (Verification of this fact is left to the reader as an exercise; see Exercise 23.) A second complication of the method we have described is that it can give rise to configurations that circle around and return to themselves in a *ring* that encloses somewhere inside it a hexagon not part of the configuration. One can also end up with hexagons that overlap other than along edges. (Verification of these two additional complications are also left to the reader as exercises; see Exercise 23.) Thus, while the counting method we have described is clever, it overestimates the number of k-polyhexes and it counts configurations such as rings that are not satisfactory. Harary and Read knew about the problems of the type we have described. They rationalized the violation of the no three hexagons meeting at a point property and the property that hexagons overlap only along common edges by thinking of the configuration as broken up into layers, with the system at some point passing from one layer to another. Thus, once a configuration of hexagons circles back on itself to create a ring or a situation of three hexagons meeting at a point, we think of it as taking off in another dimension and lying on top of a previous part of the configuration and overlapping it. This is now a standard idea in the literature (see Cyvin, Brunvoll, and Cyvin [1992]).

The difficulties we have observed illustrate the fact that the use of generating functions or any other method to get an exact count of hexagonal configurations has not met with total success. For the most part, computer-generated methods have replaced generating functions for counting various kinds of configurations. Knop, *et al.* [1983] were the first to publish results of this type for k-polyhexes, $k \leq 10$. Further advances continue to be made; see, for example, Tošić, *et al.* [1995].

EXERCISES FOR SECTION 6.4

1. Check that the Catalan number $u_4 = 14$ does indeed count the number of SOR trees of four vertices.
2. Compute the Catalan numbers u_5 and u_6.

3. Use (6.78) to compute P_5 and check that it does indeed count the number of ways to multiply a sequence of 5 numbers.

4. Compute R_3, R_4, and R_5 from the recurrence (6.79) and the initial conditions, and check by drawing the appropriate secondary structures.

5. Compute h_2, h_3, and h_4 from the recurrences (6.82)–(6.84) and the initial conditions, and check your answers by drawing the appropriate polyhexes.

6. When generating n-polyhexes, if we remove the assumption that the nth hexagon and $(n-1)$-polyhex can only have a single edge in common, then rings can form and internal "holes" can be formed in the polyhex. If there is exactly one hole and it has the size of one hexagon, the polyhex is called a *circulene*.

   (a) What is the smallest number of hexagons needed to form a circulene?

   (b) What is the smallest number of hexagons needed to form a polyhex with a hole at least the size of two hexagons? (Such a polyhex is called a *coronoid*.)

   (c) What is the smallest number of hexagons needed to form a coronoid with two holes each with size at least two hexagons?

7. A *rooted tree* is called *ordered* if a fixed ordering from left to right is assigned to all the children of a given vertex. Put another way, the k children of a given vertex are labeled with the integers $1, 2, \ldots, k$. Two rooted, ordered trees are considered different if they are not isomorphic or if they have a different root or if they are isomorphic and have the same root, but the order of children of two associated vertices differs. Suppose that r_n is the number of rooted, ordered trees of n vertices and $r_n(k)$ is the number of rooted, ordered trees of n vertices where the root has degree k.

   (a) Find an expression for r_n in terms of the $r_n(k)$.

   (b) Find an expression for $r_n(2)$ in terms of the other r_n's.

8. Suppose that

$$A(x) = \textstyle\sum_{n=0}^{\infty} a_n x^n, \quad B(x) = \sum_{n=0}^{\infty} b_n x^n, \quad \text{and} \quad C(x) = \sum_{n=0}^{\infty} c_n x^n$$

   are the ordinary generating functions for the sequences $(a_n), (b_n)$, and (c_n), respectively. Suppose that $a_0 = b_0 = c_0 = 0, a_1 = c_1 = 0, b_1 = 1$, and $a_2 = b_2 = 0, c_2 = 1$.

   (a) Suppose that $c_n = a_n + b_n, n \geq 3$. Translate this into a statement in terms of generating functions.

   (b) Suppose that $a_{n+1} = 4c_n, n \geq 0$. Translate this into a statement in terms of generating functions.

   (c) Suppose that $b_{n+1} = c_1 c_{n-1} + c_2 c_{n-2} + \cdots + c_{n-1} c_1, n \geq 2$. Translate this into a statement using generating functions.

   (d) Use your answers to parts (a)–(c) to derive an equation involving only $C(x)$.

9. Suppose that $A(x), B(x), C(x), a_0, b_0, c_0, a_1, b_1, c_1, a_2$, and b_2 are as in Exercise 8, and $c_2 = 4$.

   (a) Suppose that $c_n = a_n + 2b_n + 2, n \geq 3$. Translate this into a statement using generating functions.

(b) Suppose that $a_{n+1} = 3c_n, n \geq 0$. Translate this into a statement using generating functions.

(c) Suppose that $b_{n+1} = c_1c_{n-1} + c_2c_{n-2} + \cdots + c_{n-1}c_1, n \geq 2$. Translate this into a statement using generating functions.

(d) Use your answers to parts (a)–(c) to derive an equation involving only $C(x)$.

10. If $H(x)$ is given by (6.93), how can you tell whether to use the + sign or the − sign in computing $H(x)$?

11. (a) Use the formula for $H(x)$ [Equation (6.93)] to compute the number h_1.
    (b) Repeat for h_2. (c) Repeat for h_3.

12. Prove that the Catalan numbers $u_n = \dfrac{1}{n+1}\dbinom{2n}{n}$, $n = 0, 1, 2, \ldots$, are integers by finding two binomial coefficients whose difference is u_n. $\left[\textit{Hint:} \text{ Consider } \dbinom{2n}{n}.\right]$

13. (Waterman [1978]) This exercise will find a lower bound for R_n, the number of possible secondary structures for an RNA chain of length n.

    (a) Show from Equation (6.79) and the initial conditions that for $n \geq 2$,
    $$R_{n+1} = R_n + R_{n-1} + \sum_{k=1}^{n-2} R_k R_{n-k-1}. \tag{6.94}$$

    (b) By using Equation (6.94) and the equation obtained by replacing $n+1$ by n in (6.94), show from the initial conditions that for $n \geq 2$,
    $$R_{n+1} = R_n + R_{n-1} + R_{n-2} + \sum_{k=1}^{n-3} R_k R_{n-k-1}. \tag{6.95}$$

    (c) Since $R_{p+1} \geq R_p$, conclude that for $n \geq 2$, $R_{n+1} \geq 2R_n$, and so for $n \geq 2$, $R_n \geq 2^{n-2}$.

14. (Riordan [1975]) Suppose that $2n$ points are arranged on the circumference of a circle. Pair up these points and join corresponding points by chords of the circle. Show that the number C_n of ways of doing this pairing so that none of the chords cross is given by a Catalan number. (Maurer [1992] and Nussinov, *et al.* [1978] relate intersection patterns of these chords to biochemical problems, and Ko [1979], Read [1979], and Riordan [1975] study the number of ways to obtain exactly k overlaps of the chords.)

15. (Even [1973]) A *Last In-First Out* (LIFO) *stack* is a memory device that is like the stack of trays in a cafeteria: The last tray put on top of the stack is the first one that can be removed. A sequence of items labeled $1, 2, \ldots, n$ is waiting to be put into an empty LIFO stack. It must be put into the stack in the order given. However, at any time, we may remove an item. Removed items are never returned to the stack or the sequence awaiting storage. At the end, we remove all items from the stack and achieve a permutation of the labels $1, 2, \ldots, n$. For instance, if $n = 3$, we can first put in 1, then put in 2, then remove 2, then remove 1, then put in 3, and, finally, remove 3, obtaining the permutation $2, 1, 3$. Let q_n be the number of permutations attainable.

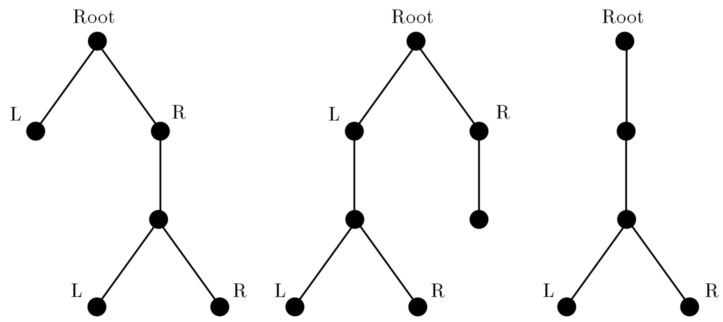


Figure 6.14: Several SPR trees.

(a) Find q_1, q_2, q_3, and q_4.

(b) Find q_n by obtaining a recurrence and solving.

16. In Section 6.4.4, suppose that we take $h_0 = -\frac{3}{2}$ instead of $h_0 = 0$.[17]

(a) Show that
$$h_{k+1} = h_0 h_k + h_1 h_{k-1} + \cdots + h_k h_0$$
holds for all $k \geq 2$.

(b) Define (c_k) to be the sequence $(h_k) * (h_k)$ and let $w_k = h_{k+1}$. Let $C(x)$ be the ordinary generating function for (c_k) and $W(x)$ be the ordinary generating function for (w_k). Relate $W(x)$ and $C(x)$ to $H(x)$ and derive a functional equation for $H(x)$.

(c) Solve for $H(x)$.

(d) Why is the answer in part (c) different from the formula for $H(x)$ given in (6.93)? What is the relation of the new $H(x)$ to the old $H(x)$?

17. Show that each well-formed sequence of n left and n right parentheses comes from some SOR tree by the method described in Section 6.4.2.

18. Let v_n count the number of ways n votes can come in for each of two candidates A and B in an election, where A never trails B. Find v_n by exhibiting a direct relationship between these orders of votes and the orders of multiplication of n numbers.

19. (Anderson [1974]) A *simple, partly ordered, rooted tree* (SPR *tree*) is a simple rooted tree in which the labels L and R are placed on the children of a vertex only if there are two children. Figure 6.14 shows several SPR trees. Let u_n count the number of SPR trees of n vertices, let a_n count the number of SPR trees of n vertices in which the root has one child, and let b_n count the number of SPR trees of n vertices in which the root has two children. Assume that $a_0 = b_0 = u_0 = 0$. Let $U(x), A(x)$, and $B(x)$ be the ordinary generating functions for (u_n), (a_n), and (b_n), respectively.

(a) Compute a_1, b_1, u_1, a_2, b_2, and u_2.

[17] This idea is due to Martin Farber [personal communication].

(b) Derive a relation that gives u_n in terms of a_n and b_n and holds for all $n \neq 1$.

(c) Derive a relation that gives a_{n+1} in terms of u_n and holds for all $n \geq 0$.

(d) Derive a relation that gives b_{n+1} in terms of $u_1, u_2, \ldots, u_n$, and holds for all $n \geq 2$.

(e) Derive a relation that gives b_{n+1} in terms of $u_0, u_1, u_2, \ldots, u_n$, and holds for all $n \geq 0$.

(f) Find u_3, u_4, a_3, a_4, b_3, and b_4 from the answers to parts (b), (c), and (e), and check by drawing SPR trees.

(g) Translate your answer to part (b) into a statement in terms of generating functions.

(h) Do the same for part (c).

(i) Do the same for part (e).

(j) Show that

$$U(x) = \frac{1}{2x}\left[1 - x \pm \sqrt{(x-1)^2 - 4x^2}\right].$$

20. (Liu [1968]) Suppose that $A(x)$ is the ordinary generating function for the sequence (a_n) and $B(x)$ is the ordinary generating function for the sequence (b_n), and that

$$b_n = a_{n-1}b_0 + a_{n-2}b_1 + \cdots + a_0 b_{n-1}, \quad n \geq 1.$$

Find a relation involving $A(x)$ and $B(x)$.

21. Generalize the result in Exercise 20 to the case

$$b_n = a_{n-r}b_0 + a_{n-r-1}b_1 + \cdots + a_0 b_{n-r}$$

for $n \geq k$, where $k \geq r$.

22. (Liu [1968]) Recall the definition of pattern in a bit string introduced in Exercise 37, Section 6.1. Let a_n be the number of n-digit bit strings that have the pattern 010 occurring for the first time at the nth digit.

(a) Show that

$$2^{n-3} = a_n + a_{n-2} + a_{n-3}2^0 + a_{n-4}2^1 + \cdots + a_3 2^{n-6},$$

$n \geq 6$.

(b) Let $b_0 = 1, b_1 = 0, b_2 = 1, b_3 = 2^0, b_4 = 2^1, b_5 = 2^2, \ldots$, and let $a_0 = a_1 = a_2 = 0$. Show that

$$2^{n-3} = a_n b_0 + a_{n-1}b_1 + a_{n-2}b_2 + \cdots + a_0 b_n,$$

$n \geq 3$.

(c) Letting $A(x)$ and $B(x)$ be the ordinary generating functions for the sequences (a_n) and (b_n), respectively, translate the equation obtained in part (b) into a statement involving $A(x)$ and $B(x)$. (See Exercise 21.)

(d) Solve for $A(x)$.

23. Show that it is possible, using the method of building up to a k-polyhex that we have described in Section 6.4.4, to end up with a configuration that:

(a) Has three hexagons meeting at a point, i.e., that violates the catacondensed condition

(b) Circles around and returns to itself in a ring that encloses somewhere inside it a hexagon not part of the configuration

(c) Has hexagons that overlap other than along edges

REFERENCES FOR CHAPTER 6

Adby, P. R., and Dempster, M. A. H., *Introduction to Optimization Methods*, Chapman & Hall, London, 1974.

Adler, I., "The Consequence of Constant Pressure in Phyllotaxis," *J. Theor. Biol., 65* (1977), 29–77.

Anderson, I., *A First Course in Combinatorial Mathematics*, Clarendon Press, Oxford, 1974.

Apostolico, A., and Giancarlo, R., "Sequence Alignment in Molecular Biology," in M. Farach-Colton, F. S. Roberts, M. Vingron, and M. S. Waterman (eds.), *Mathematical Support for Molecular Biology*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 47, American Mathematical Society, Providence, RI, 1999, 85–115.

Batschelet, E., *Introduction to Mathematics for Life Scientists*, Springer-Verlag, New York, 1971.

Brualdi, R. A., *Introductory Combinatorics*, 3rd ed., Prentice Hall, Upper Saddle River, NJ, 1999.

Clote, P., and Backofen, R., *Computational Molecular Biology: An Introduction*, Wiley, New York, 2000.

Coxeter, H. S. M., *Introduction to Geometry*, Wiley, New York, 1969.

Cyvin, B. N., Brunvoll, J., and Cyvin, S. J., "Enumeration of Benzenoid Systems and Other Polyhexes," in I. Gutman (ed.), *Advances in the Theory of Benzenoid Hydrocarbons II*, Springer-Verlag, Berlin, 1992, 65–180.

Eggleton, R. B., and Guy, R. K., "Catalan Strikes Again! How Likely Is a Function to Be Convex?," *Math. Mag., 61* (1988), 211–219.

Elaydi, S. N., *An Introduction to Difference Equations*, Springer-Verlag, New York, 1999.

Even, S., *Algorithmic Combinatorics*, Macmillan, New York, 1973.

Fowler, D. R., Prusinkiewicz, P., and Battjes, J., "A Collision-Based Model of Spiral Phyllotaxis," *Computer Graphics, 26* (1992), 361–368.

Fu, Z. L., "The Number of Latin Rectangles," *Math. Practice Theory, 2* (1992), 40–41.

Gamow, G., *One, Two, Three . . . Infinity*, Mentor Books, New American Library, New York, 1954.

Goldberg, S., *Introduction to Difference Equations*, Wiley, New York, 1958.

Gusfield, D., *Algorithms on Strings, Trees and Sequences; Computer Science and Computational Biology*, Cambridge University Press, New York, 1997.

Harary, F., and Palmer, E. M., *Graphical Enumeration*, Academic Press, New York, 1973.

Harary, F., and Read, R. C., "The Enumeration of Tree-like Polyhexes," *Proc. Edinb. Math. Soc., 17* (1970), 1–14.

Hollingdale, S. H., "Methods of Operational Analysis," in J. Lighthill (ed.), *Newer Uses of Mathematics*, Penguin Books, Hammondsworth, Middlesex, England, 1978, 176–280.

Howell, J. A., Smith, T. F., and Waterman, M. S., "Computation of Generating Functions for Biological Molecules," *SIAM J. Appl. Math., 39* (1980), 119–133.

Kelley, W. G., and Peterson, A. C., *Difference Equations: An Introduction with Applications*, Harcourt/Academic Press, San Diego, CA, 2001.

Kiefer, J., "Sequential Minimax Search for a Maximum," *Proc. Amer. Math. Soc., 4* (1953), 502–506.

Knop, J. V., Szymanski, K., Jeričević, O., and Trinajstić, N., "Computer Enumeration and Generation of Benzenoid Hydrocarbons and Identification of Bay Regions," *J. Comput. Chem., 4* (1983), 23–32.

Ko, C. S., "Broadcasting, Graph Homomorphisms, and Chord Intersections," Ph.D. thesis, Department of Mathematics, Rutgers University, New Brunswick, NJ, 1979.

Liu, C. L., *Introduction to Combinatorial Mathematics*, McGraw-Hill, New York, 1968.

MacMahon, P. A., "A New Method and Combinatory Analysis with Application to Latin Squares and Associated Questions," *Trans. Cambridge Philos. Soc., 16* (1898), 262–290.

Markowsky, G., "Misconceptions about the Golden Ratio," *The College Mathematics Journal, 23* (1992), 2–19.

Maurer, S. B., "A Minimum Cycle Problem in Bacterial DNA Research," RUTCOR (Rutgers Center for Operations Research) Research Report 27–92, 1992.

Metzler, L., "The Nature and Stability of Inventory Cycles," *Rev. Econ. Statist., 23* (1941), 113–129.

Myers, E. W., "Seeing Conserved Signals: Using Algorithms to Detect Similarities Between Biosequences," in E. S. Lander and M. S. Waterman (eds.), *Calculating the Secrets of Life*, National Academy Press, Washington, DC 1995, 56–89.

Nussinov, R. P., Pieczenik, G., Griggs, J. R., and Kleitman, D. J., "Algorithms for Loop Matchings," *SIAM J. Appl. Math., 35* (1978), 68–82.

Read, R. C., "The Chord Intersection Problem," *Ann. N.Y. Acad. Sci., 319* (1979), 444–454.

Riordan, J., "The Distribution of Crossings of Chords Joining Pairs of $2n$ Points on a Circle," *Math. Comp., 29* (1975), 215–222.

Ryser, H. J., *Combinatorial Mathematics*, Carus Mathematical Monographs No. 14, Mathematical Association of America, Washington, DC, 1963.

Samuelson, P. A., "Interactions between the Multiplier Analysis and the Principle of Acceleration," *Rev. Econ. Statist., 21* (1939), 75–78. (Reprinted in *Readings in Business Cycle Theory*, Blakiston, Philadelphia, 1944.)

Schips, M., *Mathematik und Biologie*, Teubner, Leipzig, 1922.

Setubal, J. C., and Meidanis, J., *Introduction to Computational Molecular Biology*, PWS Publishers, Boston, 1997.

Shannon, C. E., "The Zero-Error Capacity of a Noisy Channel," *IRE Trans. Inf. Theory, IT–2* (1956), 8–19.

Shao, J. Y., and Wei, W. D., "A Formula for the Number of Latin Squares," *Discrete Math., 110* (1992), 293–296.

Sloane, N. J. A. (ed.), *The On-Line Encyclopedia of Integer Sequences*, published electronically at http://www.research.att.com/~njas/sequences/ (2003).

Stein, P. R., and Waterman, M. S., "On Some New Sequences Generalizing the Catalan and Motzkin Numbers," *Discrete Math., 26* (1978), 261–272.

Takács, L., "The Problem of Coincidences," *Arch. Hist. Exact Sci., 21* (1980), 229–244.

Tošić, R., Masulovic, D., Stojmenovic, I., Brunvoll, J., Cyvin, B. N., and Cyvin, S. J., "Enumeration of Polyhex Hydrocarbons to $h = 17$," *J. Chem. Inf. Comput. Sci., 35* (1995), 181–187.

Waterman, M. S., "Secondary Structure of Single-Stranded Nucleic Acids," *Studies on Foundations and Combinatorics, Advances in Mathematics Supplementary Studies*, Vol. 1, Academic Press, New York, 1978, 167–212.

Waterman, M. S., *Introduction to Computational Biology; Maps, Sequences and Genomes*, CRC Press, Boca Raton, FL, 1995.

Chapter 7

The Principle of Inclusion and Exclusion

7.1 THE PRINCIPLE AND SOME OF ITS APPLICATIONS

7.1.1 Some Simple Examples

In this chapter we introduce still another basic counting tool, known as the principle of inclusion and exclusion. We introduce it with the following example.

Example 7.1 Job Applicants Suppose that in a group of 18 job applicants, 10 have computer programming expertise, 5 have statistical expertise, and 2 have both programming and statistical expertise. How many of the group have neither expertise? To answer this question we draw a Venn diagram such as that shown in Figure 7.1.[1] There are 18 people altogether. To find out how many people have neither expertise, we want to subtract from 18 the number having programming expertise (10) and the number having statistical expertise (5). However, we may have counted several people twice. In particular, all people who have both kinds of expertise (the number of people in the intersection of the two sets programming expertise and statistical expertise in Figure 7.1) have been counted twice. There are 2 such people. Thus, we have to add these 2 back in to obtain the right count. Altogether, we conclude that

$$18 - 10 - 5 + 2 = 5$$

is the number of people who have neither expertise. We shall generalize the reasoning we have just gone through. ■

[1]The reader who is unfamiliar with Venn diagrams should consult any finite mathematics text, for example, Goldstein, Schneider, and Siegel [2001] or Mizrahi and Sullivan [1999].

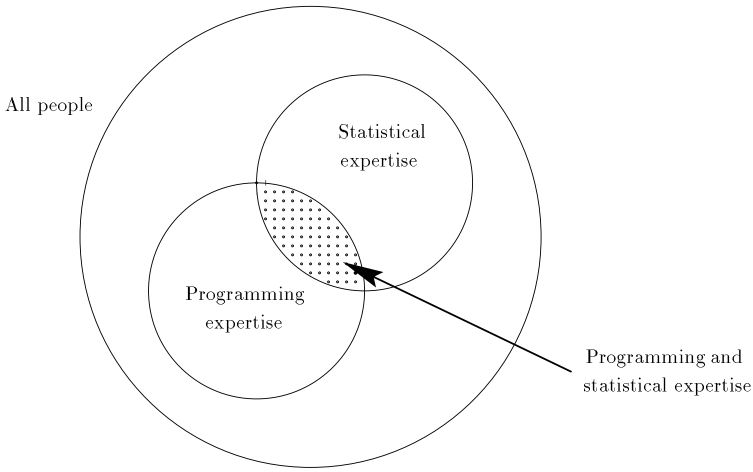


Figure 7.1: A Venn diagram for Example 7.1.

Suppose that we have a set A of N objects. Let $a_1, a_2, \ldots, a_r$ be a set of properties the objects may have, and let $N(a_i)$ be the number of objects having property a_i. An object may have several (or none) of the properties in question. Let $N(a_i')$ count the number of objects not having the property a_i. Hence, we have

$$N = N(a_i) + N(a_i').$$

Since an object can have more than one property, it is useful to count the number of objects having both properties a_i and a_j. This will be denoted by $N(a_i a_j)$. The number of objects having neither of the properties a_i and a_j will be denoted $N(a_i' a_j')$ and the number of objects having property a_j but not property a_i will be denoted $N(a_i' a_j)$. We shall also use the following notation, which has the obvious interpretation:

$$\begin{gathered} N(a_i a_j \cdots a_k), \\ N(a_i' a_j' \cdots a_k'), \\ N(a_i' a_j \cdots a_k), \\ \text{etc.} \end{gathered}$$

In Example 7.1 the objects in A are the 18 people, so $N = 18$. Let property a_1 be having programming expertise and property a_2 be having statistical expertise. Then

$$N(a_1) = 10, \quad N(a_2) = 5, \quad N(a_1 a_2) = 2.$$

By computation, we determined that

$$N(a_1' a_2') = 5.$$

It is also possible to compute $N(a_1' a_2)$ and $N(a_1 a_2')$. The former is the number of people who have statistical expertise but do not have programming expertise, and this is given by $5 - 2 = 3$.

Our computation in Example 7.1 used the following formula for $N(a_1'a_2')$:

$$N(a_1'a_2') = N - N(a_1) - N(a_2) + N(a_1a_2). \tag{7.1}$$

Note that certain objects are *included* too often, so some of these have to be *excluded*. Equation (7.1) is a special case of a principle known as the *principle of inclusion and exclusion*. The process of including and excluding objects corresponds to the addition and subtraction, respectively, in Equation (7.1). Let us develop a similar principle for $N(a_1'a_2'a_3')$, the number of objects having neither property a_1 nor property a_2 nor property a_3. The principle is illustrated in the Venn diagram of Figure 7.2. We first include all the objects in A (all N of them). Then we exclude those having property a_1, those having property a_2, and those having property a_3. Since some objects have more than one of these properties, we need to add back in those objects which have been excluded more than once. We add back in (include) those objects having two of the properties, the objects corresponding to areas in Figure 7.2 that are colored in. Then we have added several objects back in too often, namely those which have all three of the properties, the objects in the one area of Figure 7.2 that has crosshatching. These objects must now be excluded. The result of this reasoning, which we shall formalize below, is the following formula:

$$\begin{aligned} N(a_1'a_2'a_3') \;=\; & N - N(a_1) - N(a_2) - N(a_3) + N(a_1a_2) + \\ & N(a_1a_3) + N(a_2a_3) - N(a_1a_2a_3). \end{aligned} \tag{7.2}$$

In general, the formula for the number of objects not having any of r properties is obtained by generalizing Equation (7.2). The general formula is called the *principle of inclusion and exclusion*. In the form we present it, the principle was discovered by Sylvester in the mid-nineteenth century. In another form, it was discovered by De Moivre [1718] some years earlier. The principle is given in the following theorem:

Theorem 7.1 (Principle of Inclusion and Exclusion) If N is the number of objects in a set A, the number of objects in A having none of the properties $a_1, a_2, \ldots, a_r$ is given by

$$\begin{aligned} N(a_1'a_2'\cdots a_r') \;=\; & N - \sum_i N(a_i) + \sum_{i \neq j} N(a_ia_j) - \\ & \sum_{\substack{i,j,k \\ \text{different}}} N(a_ia_ja_k) \pm \cdots + (-1)^r N(a_1a_2\cdots a_r). \end{aligned} \tag{7.3}$$

In (7.3), the first sum is over all i from $\{1, 2, \ldots, r\}$. The second sum is over all unordered pairs $\{i, j\}$, with i and j from $\{1, 2, \ldots, r\}$ and $i \neq j$. The third sum is over all unordered triples $\{i, j, k\}$, with i, j, and k from $\{1, 2, \ldots, r\}$ and i, j, k distinct. The general term is $(-1)^t$ times a sum of terms of the form $N(a_{i_1}a_{i_2}\cdots a_{i_t})$, where the sum is over all unordered t-tuples $\{i_1, i_2, \ldots, i_t\}$ from $\{1, 2, \ldots, r\}$, with $i_1, i_2, \ldots, i_t$ distinct. In the remainder of this chapter, we present applications and variants of the principle of inclusion and exclusion. We present a proof of Theorem 7.1 in Section 7.1.2.

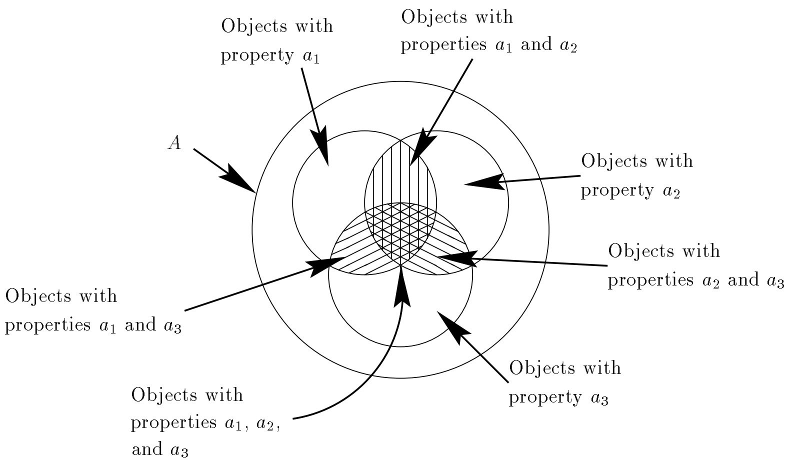


Figure 7.2: A Venn diagram illustrating the principle of inclusion and exclusion.

Example 7.2 Emissions Testing Fifty cars are tested for pollutant emissions of nitrogen oxides (NO_x), hydrocarbons (HC), and carbon monoxide (CO). One of the cars exceeds the environmental standards for all three pollutants. Three cars exceed them for NO_x and HC, two for NO_x and CO, one for HC and CO, six for NO_x, four for HC, and three for CO. How many cars meet the environmental standards for all three pollutants? We let A be the set of cars, let a_1 be the property of exceeding the standards for NO_x and let a_2 and a_3 be the same property for HC and CO, respectively. We would like to calculate $N(a_1'a_2'a_3')$. We are given the following information:

$$\begin{array}{rclrclrclrcl} N &=& 50, & N(a_1a_2a_3) &=& 1, & N(a_1a_2) &=& 3, & N(a_1a_3) &=& 2, \\ N(a_2a_3) &=& 1, & N(a_1) &=& 6, & N(a_2) &=& 4, & N(a_3) &=& 3. \end{array}$$

Thus, by the principle of inclusion and exclusion,

$$N(a_1'a_2'a_3') = 50 - 6 - 4 - 3 + 3 + 2 + 1 - 1 = 42. \qquad \blacksquare$$

7.1.2 Proof of Theorem 6.1[2]

The idea of the proof of Theorem 7.1 is a very simple one. The left-hand side of (7.3) counts the number of objects in A having none of the properties. We shall simply show that every object having none of the properties is counted exactly one time in the right-hand side of (7.3) and every object having at least one property is counted exactly zero times (in a net sense). Suppose that an object has none of the properties in question. Then it is counted once in computing N, but not in $\sum N(a_i)$, $\sum N(a_ia_j)$, and so on. Hence, it is counted exactly once in the right-hand

[2] This subsection may be omitted. However, the reader is urged to read it.

side of (7.3). Suppose that an object has exactly p of the properties $a_1, a_2, \ldots, a_r$, $p > 0$. Now the object is counted $1 = \binom{p}{0}$ times in computing N, the number of objects in A. It is counted once in each expression $N(a_i)$ for a property a_i it has, so exactly $p = \binom{p}{1}$ times in $\sum N(a_i)$. In how many terms $N(a_i a_j)$ is the object counted? The answer is it is the number of pairs of properties a_i and a_j which the object has, and this number is given by the number of ways to choose two properties from p properties, that is, by $\binom{p}{2}$. Thus, the object is counted exactly $\binom{p}{2}$ times in $\sum N(a_i a_j)$. Similarly, in $\sum N(a_i a_j a_k)$ it is counted exactly $\binom{p}{3}$ times, and so on. All together, the number of times the object is counted in the right-hand side of (7.3) is given by

$$\binom{p}{0} - \binom{p}{1} + \binom{p}{2} - \binom{p}{3} \pm \cdots + (-1)^r \binom{p}{r}. \tag{7.4}$$

Since $p \leq r$ and since by convention $\binom{p}{k} = 0$ if $p < k$, (7.4) becomes

$$\binom{p}{0} - \binom{p}{1} + \binom{p}{2} - \binom{p}{3} \pm \cdots + (-1)^p \binom{p}{p}. \tag{7.5}$$

Since $p > 0$, Theorem 2.9 implies that the expression (7.5) is 0, so the object contributes a net count of 0 to the right-hand side of (7.3). This completes the proof.

7.1.3 Prime Numbers, Cryptography, and Sieves

One of the earliest problems about numbers to interest mathematicians was the problem of identifying all *prime numbers*, integers greater than 1 whose only positive divisors are 1 and themselves. Recently, Agrawal, Kayal, and Saxena [2002] presented a deterministic polynomial-time algorithm that determines whether a given integer $n > 1$ is prime or *composite* (nonprime). Previously, only nonpolynomial time algorithms, polynomial-time algorithms assuming well-known conjectures, and probabilistic algorithms were available to test whether a number is prime or composite. For example, if not identified as composite by a probabilistic test, a number is highly likely to be prime, i.e., an *industrial-grade prime.* This designation is an important one since prime numbers are needed in many practical applications and industrial-grade primes have been used. For more on these methods of primality testing, see Adleman and Huang [1992], Adleman, Pomerance, and Rumely [1983], Goldwasser and Kilian [1986], Miller [1976], Rabin [1980], or Solovay and Strassen [1977].

One critical use is in *cryptography*, the field of the mathematical sciences devoted to concealing messages. Nowadays, cryptography arises in problems such as keeping email secure, protecting the privacy of medical records, protecting the integrity of electronic financial transactions, and protecting copyrights in a digital world. For example, it is well known that every positive integer greater than 1 can be uniquely factored as a product of primes. This is a key to the most commonly used "public-key" algorithm in cryptography, the *RSA Algorithm* due to Rivest, Shamir, and Adleman [1978]. Factoring integers has long been considered a hard computational problem. It is widely believed that integers cannot be factored in time polynomial in the number of bits or digits that represent the number. So widely is this believed that the security of cryptography methods such as RSA rely on the difficulty of factoring. One of the cornerstones of basing a cryptographic method on the hardness (i.e., nonpolynomial time) of a problem is that the definition of what it is possible to compute in polynomial time is independent of the model of computing used. Quantum computing, using devices based on quantum mechanics, is one such model. Quantum computers do not operate like conventional ones, but make use of the quantum states of atoms, which offers a computing capacity far in excess of current parallel supercomputers. However, only prototypes of these computers currently exist. Shor [1997] proved the remarkable result that a number can be factored in polynomial time in the quantum computing model. He obtained a similar result for the discrete log problem which is the basis of another cryptography method, Diffie-Hellman (see Diffie and Hellman [1976]). Thus, for problems of very practical interest, the quantum model has significantly faster algorithms than are known for the traditional computing models. These results not only demonstrated that quantum computing could give new power to computing, but also cast doubt on the safety of cryptography based on the hardness of factoring or discrete log. (For more on cryptography, see Anshel, Anshel, and Goldfeld [1999], Koblitz [1994], Rhee [1993], Schneier [1995], Seberry and Pieprzyk [1989], or Stallings [1999].) We return to cryptography in Section 9.2.5 and in particular to the RSA cryptosystem in Section 9.3.2.

An old problem in mathematics is to identify all prime numbers. The Greek Erastothenes is credited with inventing the following procedure for identifying all prime numbers between 1 and N. First write out all the numbers between 1 and N. Cross out 1. Then cross out those numbers divisible by and larger than 2. Search for the first number larger than 2 not yet crossed out—it is 3—and cross out all those numbers divisible by and larger than 3. Then search for the first number larger than 3 not yet crossed out—it is 5—and cross out all numbers divisible by and larger than this number; and so on. The prime numbers are the ones remaining when the procedure ends. The following shows the steps in this procedure if $N = 25$:

1. Cross out 1 and cross out numbers divisible by and larger than 2:

$$\cancel{1}\ 2\ 3\ \cancel{4}\ 5\ \cancel{6}\ 7\ \cancel{8}\ 9\ \cancel{10}\ 11\ \cancel{12}\ 13\ \cancel{14}\ 15\ \cancel{16}\ 17\ \cancel{18}\ 19\ \cancel{20}\ 21\ \cancel{22}\ 23\ \cancel{24}\ 25$$

2. Cross out numbers divisible by and larger than 3 among those not yet crossed out:

$$\cancel{1}\ 2\ 3\ \cancel{4}\ 5\ \cancel{6}\ 7\ \cancel{8}\ \cancel{9}\ \cancel{10}\ 11\ \cancel{12}\ 13\ \cancel{14}\ \cancel{15}\ \cancel{16}\ 17\ \cancel{18}\ 19\ \cancel{20}\ \cancel{21}\ \cancel{22}\ 23\ \cancel{24}\ 25$$

3. Cross out numbers divisible by and larger than 5 among those not yet crossed out:

$$\cancel{1}\ 2\ 3\ \cancel{4}\ 5\ \cancel{6}\ 7\ \cancel{8}\ \cancel{9}\ \cancel{10}\ 11\ \cancel{12}\ 13\ \cancel{14}\ \cancel{15}\ \cancel{16}\ 17\ \cancel{18}\ 19\ \cancel{20}\ \cancel{21}\ \cancel{22}\ 23\ \cancel{24}\ \cancel{25}$$

Dividing by 7, 11, and so on, does not remove any more numbers, so the numbers remaining are the prime numbers between 1 and 25. This procedure used to be carried out on a wax tablet (Vilenkin [1971]), with numbers punched out rather than crossed out. The result was something like a sieve, and hence the procedure came to be known as the *Sieve of Erastothenes*. A basic question that arises is: How many primes are there between 1 and N? The answer is closely related to the answer to the following question: How many numbers between 1 and N (other than 1) are not divisible by 2, 3, 5, ...? In the next example, we see how to answer questions of this type.

Example 7.3 How many integers between 1 and 1000 are:

(a) Not divisible by 2?
(b) Not divisible by either 2 or 5?
(c) Not divisible by 2, 5, or 11?

To answer these questions, let us consider the set of 1000 integers between 1 and 1000 and let a_1 be the property of being divisible by 2, a_2 the property of being divisible by 5, and a_3 the property of being divisible by 11. We would like the following information:

$$\text{(a) } N(a_1'); \quad \text{(b) } N(a_1'a_2'); \quad \text{(c) } N(a_1'a_2'a_3').$$

We are given $N = 1000$. Also,

$$N(a_1) = 500,$$

since every other integer is divisible by 2. Hence,

$$N(a_1') = N - N(a_1) = 500,$$

which gives us the answer to (a). Next,

$$N(a_2) = \frac{1}{5}(1000) = 200.$$

Also, every tenth integer is divisible by 2 and by 5, so

$$N(a_1a_2) = \frac{1}{10}(1000) = 100.$$

Hence, by the principle of inclusion and exclusion,

$$N(a_1'a_2') = 1000 - 500 - 200 + 100 = 400,$$

which answers (b). Finally, we have

$$N(a_3) = \frac{1}{11}(1000) = 90.9.$$

Of course, since $N(a_3)$ is an integer, this means that

$$N(a_3) = 90.$$

In short,

$$N(a_3) = \lfloor 90.9 \rfloor = 90.$$

Also, every 22nd integer is divisible by 2 and by 11, so

$$N(a_1a_3) = \left\lfloor \frac{1}{22}(1000) \right\rfloor = \lfloor 45.5 \rfloor = 45.$$

Similarly,

$$N(a_2a_3) = \left\lfloor \frac{1}{55}(1000) \right\rfloor = \lfloor 18.2 \rfloor = 18.$$

Finally, every 110th integer is divisible by 2, 5, and 11, so

$$N(a_1a_2a_3) = \left\lfloor \frac{1}{110}(1000) \right\rfloor = \lfloor 9.1 \rfloor = 9.$$

Thus,

$$N(a_1'a_2'a_3') = 1000 - (500 + 200 + 90) + (100 + 45 + 18) - 9 = 364. \qquad \blacksquare$$

Example 7.4 The Number of Integers Relatively Prime to a Given Integer Two integers are *relatively prime* if they have no common divisor greater than 1. Two integers that are not relatively prime must have a common divisor which is a prime. (Why?) Hence, $45 = 3^2 \cdot 5$ and $56 = 2^3 \cdot 7$ are relatively prime. How many integers from 1 to 1000 are relatively prime to 1000? Since $1000 = 2^3 \cdot 5^3$, the only primes dividing 1000 are 2 and 5. So, we want to find the number of integers between 1 and 1000 that are not divisible by 2 or 5. In Example 7.3 we found this to be 400. $\blacksquare$

The method in the previous example generalizes. *Euler's ϕ function* $\phi(n)$ is defined to be the number of integers from 1 to n that are relatively prime to n.

Theorem 7.2 If n is an integer whose unique prime factorization is

$$n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r},$$

then

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_r}\right).$$

Proof. An integer from 1 to n is not relatively prime to n if and only if it is divisible by p_1 or p_2 or ... or p_r. Let $A = \{1, 2, \ldots, n\}$ and a_i be the property of being divisible by p_i. Then $N(a_{i_1} a_{i_2} \cdots a_{i_m})$ is obtained by counting elements of A that are multiples of $p_{i_1} p_{i_2} \cdots p_{i_m}$, i.e.,

$$N(a_{i_1} a_{i_2} \cdots a_{i_m}) = \left\lfloor \frac{n}{p_{i_1} p_{i_2} \cdots p_{i_m}} \right\rfloor = \frac{n}{p_{i_1} p_{i_2} \cdots p_{i_m}}$$

since n is divisible by $p_{i_1} p_{i_2} \cdots p_{i_m}$. It follows by the principle of inclusion and exclusion that

$$\phi(n) = N(a_1' a_2' \cdots a_r') = n - n \sum_i \frac{1}{p_i} + n \sum_{i \neq j} \frac{1}{p_i p_j} \mp \cdots + (-1)^r n \frac{1}{p_1 p_2 \cdots p_r}.$$

The result follows from the observation that

$$(1 + x_1)(1 + x_2) \cdots (1 + x_r) = \sum_{I \subseteq \{1,2,\ldots,r\}} \left(\prod_{i \in I} x_i \right).$$

Q.E.D.

For more on Euler's ϕ function and related results in number theory, see books such as Hardy and Wright [1980]; see also Exercise 32.

Example 7.5 The Woman and the Egg, Fibonacci, and Sieves An old medieval puzzle (Reingold, Nievergelt, and Deo [1977]) goes as follows. An old woman is on her way to the market to sell eggs when she is knocked down by a horseman. Since all the eggs were broken, the horseman offers to pay the damages. The old woman does not remember how many eggs she had. She does remember that when she took them 2 at a time, 1 was left over, and this was also true when she took them 3 or 4 at a time. However, none were left over when she took them 5 at a time. Is there a way to determine the number of eggs the woman had, if it is reasonable to assume that she had at most 25 eggs? A natural way to proceed is to mimic the Sieve of Erastothenes. The Sieve is based on the problem of finding all numbers between 1 and n that are in all of the following sets:

$$\{2k+1 : k \geq 1\}, \{3k+1, 3k+2 : k \geq 1\}, \{5k+1, 5k+2, 5k+3, 5k+4 : k \geq 1\},$$
$$\{7k+1, 7k+2, 7k+3, 7k+4, 7k+5, 7k+6 : k \geq 1\}, \ldots,$$
$$\{pk+1, pk+2, \ldots, pk+p-1 : k \geq 1\},$$

where p is the largest prime less than or equal to n. The old woman's problem can similarly be formulated as the problem of finding all integers between 1 and n in all of the following sets:

$$\{2k+1 : k \geq 1\}, \{3k+1 : k \geq 1\}, \{4k+1 : k \geq 1\}, \{5k : k \geq 1\}.$$

A sieve can be used to solve this problem for $n = 25$ by listing all integers between 1 and 25 and crossing out all numbers of the form $2k$, all numbers of the form $3k$ or $3k+2$, all numbers of the form $4k$ or $4k+2$ or $4k+3$ (the former two types of numbers would already have been crossed out), and, finally, all numbers of the form $5k + 1$, $5k + 2$, $5k + 3$, or $5k + 4$. How many eggs did the woman have? Reingold, Nievergelt, and Deo [1977] show how similar sieve methods can be used to solve more complex problems, such as quickly testing the first 1 million Fibonacci numbers to see which are squares. (The only ones are 1 and 144.) ■

7.1.4 The Probabilistic Case

Suppose that an integer between 1 and 1000 is selected at random. What is the probability that it is not divisible by 2, 5, or 11? The answer is simple. Consider an experiment in which the outcome is one of the integers between 1 and 1000, and the outcomes are equally likely. The number of outcomes signaling the event "is not divisible by 2, 5, or 11" is 364, by the computation in the preceding section. Hence, the probability in question is $364/1000 = .364$.

More generally, suppose we consider an experiment that produces an outcome in a finite set S, the sample space. Let us consider the events $E_1, E_2, \ldots, E_r$. What is the probability that none of these events occur? To answer this, we shall assume, as we have throughout this book, that all outcomes in the sample space S are equally likely. (However, it can be shown that this assumption is not needed to obtain the main result of this subsection.) Let the set A be the set S, and let a_i be the property that an outcome signals event E_i. Let $p_{ijk\ldots}$ be the probability that events E_i and E_j and E_k and ... occur. We conclude from Theorem 7.1 that the probability p that none of the events $E_1, E_2, \ldots, E_r$ occur is given by

$$\begin{aligned} p &= \frac{N(a_1'a_2'\cdots a_r')}{n(S)} \\ &= 1 - \frac{\sum N(a_i)}{n(S)} + \frac{\sum N(a_i a_j)}{n(S)} - \frac{\sum N(a_i a_j a_k)}{n(S)} \pm \cdots + (-1)^r \frac{N(a_1 a_2 \cdots a_r)}{n(S)}, \end{aligned}$$

where $n(S)$ is the number of outcomes in S. Thus,

$$p = 1 - \sum p_i + \sum p_{ij} - \sum p_{ijk} \pm \cdots + (-1)^r p_{12\cdots r}. \tag{7.6}$$

Example 7.6 Antipodal Points Both Covered by Water[3] There is an intriguing problem found in most topology books that asks the reader to prove that on any great circle around the Earth, there exist antipodal points that have the same temperature. A similar type of problem can be asked of a combinatorialist. It is known that ocean covers more than half of the Earth's surface. Show that there are two antipodal points on the Earth that are both covered by water. Let X denote a random point on the Earth. For concreteness, we consider only points of integer latitude and longitude, so the set of all such points is finite. We also let $-X$ denote the point antipodal to X. Consider the following events:

$$\begin{aligned} E_1 &= \text{point } X \text{ is covered by water}, \\ E_2 &= \text{point } -X \text{ is covered by water}. \end{aligned}$$

[3] This example is from Shen [1998].

By (7.6), the probability p that neither event E_1 nor E_2 occurs equals $1-(p_1+p_2)+p_{12}$, where p_i is the probability that event E_i occurs and p_{12} is the probability that both E_1 and E_2 occur. Since p_1 and p_2 are each greater than $1/2$, p_{12} must be positive for p to be between 0 and 1. Thus, there must exist a point X with both properties, i.e., so that X and $-X$ are both covered by water. ■

7.1.5 The Occupancy Problem with Distinguishable Balls and Cells

In Section 2.10, we considered the occupancy problem of placing n distinguishable balls into c distinguishable cells. Let us now ask: What is the probability that no cell will be empty? Let S be the set of distributions of balls to cells, and let E_i be the event that the ith cell is empty. Define A and a_i as above. Then $N(S)=c^n$, $N(a_i)=(c-1)^n$, $N(a_ia_j)=(c-2)^n$, $N(a_ia_ja_k)=(c-3)^n, \ldots$. Moreover, there are $\binom{c}{1}$ ways to choose property a_i, $\binom{c}{2}$ ways to choose properties a_i and a_j, and so on. Hence, the number of distributions of n balls into c cells with no empty cell is given by

$$c^n-\binom{c}{1}(c-1)^n+\binom{c}{2}(c-2)^n-\binom{c}{3}(c-3)^n \pm \cdots + (-1)^c\binom{c}{c}(c-c)^n,$$

which equals

$$\sum_{t=0}^{c}(-1)^t\binom{c}{t}(c-t)^n. \tag{7.7}$$

Then the probability that no cell is empty is given by

$$\frac{c^n-\binom{c}{1}(c-1)^n+\binom{c}{2}(c-2)^n-\binom{c}{3}(c-3)^n+\cdots+(-1)^c\binom{c}{c}(c-c)^n}{c^n},$$

which equals

$$\sum_{t=0}^{c}(-1)^t\binom{c}{t}\left(1-\frac{t}{c}\right)^n, \tag{7.8}$$

since

$$\frac{(c-t)^n}{c^n}=\left(1-\frac{t}{c}\right)^n.$$

Example 7.7 Fast-Food Prizes Suppose that a fast-food outlet gives away three different toys in children's meal packs, one to a package. If we buy six children's meals and each toy is equally likely to be in any one meal pack, what is the probability of getting all three different toys? We imagine placing $n=6$ balls or toys into $c=3$ cells or types of toys. The number of ways the toys can be placed into types so that no cell (or type) is empty is given by Equation (7.7) as

$$3^6-\binom{3}{1}\cdot 2^6+\binom{3}{2}\cdot 1^6-\binom{3}{3}\cdot 0^6=540.$$

The probability of this happening is $540/3^6 = .741$. This can also be computed directly by Equation (7.8) as

$$1 - \binom{3}{1} \cdot \left(\frac{2}{3}\right)^6 + \binom{3}{2} \cdot \left(\frac{1}{3}\right)^6 - \binom{3}{3} \cdot 0^6 = .741.$$

■

7.1.6 Chromatic Polynomials

In Section 3.4 we introduced the idea of a chromatic polynomial of a graph. The principle of inclusion and exclusion can be used to calculate chromatic polynomials. It is interesting to note how the same counting problem can be solved in more than one way. On several occasions in this chapter we shall be able to apply the principle of inclusion and exclusion to count a quantity that we have previously counted in a different way. Consider, for example, the graph of Figure 7.3. We consider all possible colorings of the vertices of G in x or fewer colors. We shall even allow colorings where two vertices joined by an edge get the same color, but we shall call such colorings *improper*, and all others *proper*. Let us consider the set of all colorings, proper or improper, of the graph G in x or fewer colors. There are x^4 such colorings since each of the 4 vertices can be colored by any of the x colors. We shall introduce one property a_i for each edge of the graph G. Thus,

a_1 is the property that a and b get the same color,
a_2 is the property that b and c get the same color,
a_3 is the property that c and d get the same color,
a_4 is the property that d and a get the same color.

To calculate $P(G, x)$, the number of (proper) colorings of G with x or fewer colors, we have to calculate $N(a_1'a_2'a_3'a_4')$. We have $N(a_1) = x^3$, since there are x choices for the color for a and b, then x choices for the color for c, and x for the color for d (recall that improper colorings are allowed). Similarly,

$$N(a_2) = N(a_3) = N(a_4) = x^3.$$

Next,

$$N(a_1a_2) = x^2.$$

For a and b must receive the same color, and also b and c. Hence, there are x choices for the one color that a, b, and c receive, and then x choices for the color for d. Similar reasoning shows that

$$N(a_1a_3) = N(a_1a_4) = N(a_2a_3) = N(a_2a_4) = N(a_3a_4) = x^2.$$

Similarly,

$$N(a_1a_2a_3) = N(a_1a_2a_4) = N(a_1a_3a_4) = N(a_2a_3a_4) = x,$$

since in all these cases all the vertices must receive the same color. This reasoning also leads us to conclude

$$N(a_1a_2a_3a_4) = x.$$

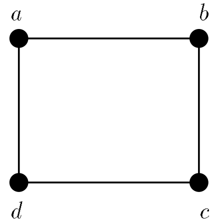


Figure 7.3: A graph.

Hence, by the principle of inclusion and exclusion,

$$\begin{aligned} P(G,x) &= N(a_1'a_2'a_3'a_4') \\ &= x^4 - 4x^3 + 6x^2 - 4x + x \\ &= x^4 - 4x^3 + 6x^2 - 3x. \end{aligned}$$

This computation can be checked by the methods of Chapter 3.

Let us generalize this example as follows. Suppose that G is any graph and we wish to compute $P(G,x)$. Consider the set A of all colorings, proper or improper, of the vertices of G in x or fewer colors. For each edge i, let a_i be the property that the end vertices of edge i get the same color. Suppose that $|V(G)| = n$ and $|E(G)| = r$. Then $N = |A| = x^n$ and

$$P(G,x) = N(a_1'a_2'\cdots a_r').$$

Thus,

$$P(G,x) = x^n - \sum N(a_i) + \sum N(a_ia_j) \mp \cdots + (-1)^e \sum N(a_{i_1}a_{i_2}\cdots a_{i_e}) + \cdots. \tag{7.9}$$

Let us consider the term $N(a_{i_1}a_{i_2}\cdots a_{i_e})$. Suppose that H is the subgraph of G consisting of all the vertices of G and having edges i_1, i_2, $\ldots$, i_e. A subgraph H containing all the vertices of G was called a *spanning subgraph* in Chapter 3. Note that a coloring (proper or improper) of G satisfying properties a_{i_1}, a_{i_2}, $\ldots$, a_{i_e} is equivalent to a coloring (proper or improper) of H satisfying properties a_{i_1}, a_{i_2}, $\ldots$, a_{i_e}. Now in such a coloring of H, any connected component of H must have all of its vertices colored the same. A color for a component of H can be chosen at random. Thus, the number of colorings of vertices of G in x or fewer colors and satisfying properties a_{i_1}, a_{i_2}, $\ldots$, a_{i_e} is given by $x^{c(H)}$, where $c(H)$ is the number of connected components of H.

Each spanning subgraph H of e edges and c components corresponds to some set of properties a_{i_1}, a_{i_2}, $\ldots$, a_{i_e} and will contribute a term $(-1)^e x^c$ to the right-hand side of (7.9). Thus, we have the following theorem.

Theorem 7.3[4] If G is a graph and $h(e,c)$ is the number of spanning subgraphs

[4] This theorem was discovered by Birkhoff [1912] (for graphs arising from maps) and first worked out by inclusion and exclusion by Whitney [1932].

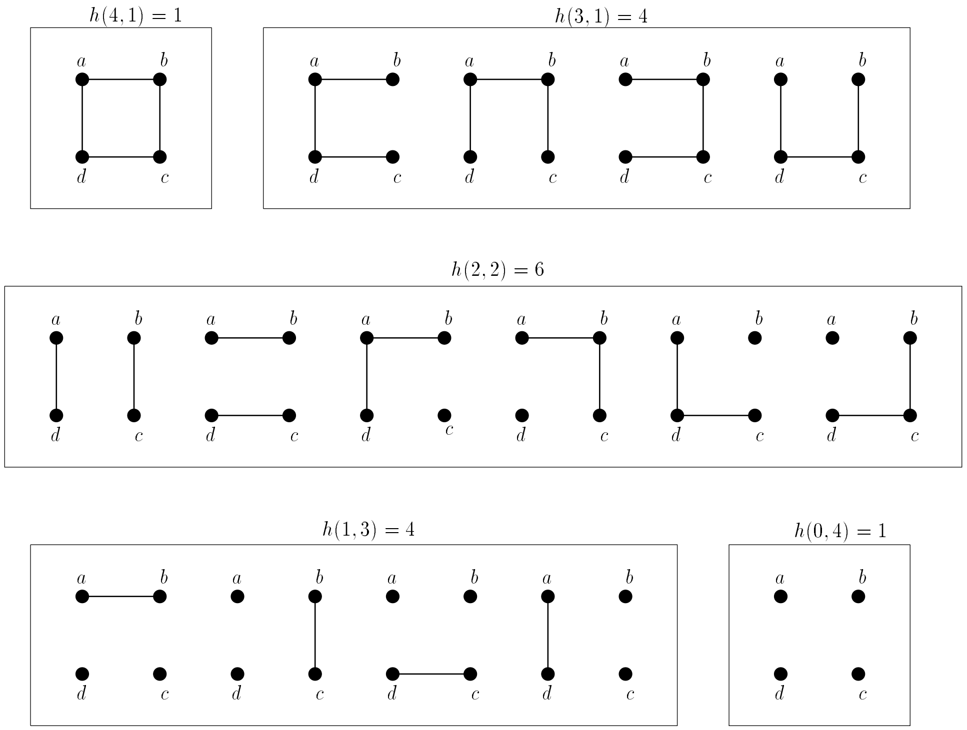


Figure 7.4: The spanning subgraphs of the graph of Figure 7.3.

of e edges and c components, then

$$P(G,x) = \sum_{e,c} (-1)^e h(e,c) x^c.$$

In this theorem, note that $h(0,c)$ is 1 if $c = n$ and 0 otherwise. Note that the theorem gives a quick proof that $P(G,x)$ is a polynomial. In our example of Figure 7.3, we have the following results, which are illustrated in Figure 7.4:

$$h(4,1) = 1, \qquad h(3,1) = 4, \qquad h(2,2) = 6, \qquad h(1,3) = 4, \qquad h(0,4) = 1,$$

and otherwise, $h(e,c) = 0$. Thus, we have

$$\begin{aligned} P(G,x) &= (-1)^4 h(4,1)x + (-1)^3 h(3,1)x + (-1)^2 h(2,2)x^2 + (-1)^1 h(1,3)x^3 \\ &\quad + (-1)^0 h(0,4)x^4 \\ &= x - 4x + 6x^2 - 4x^3 + x^4 \\ &= x^4 - 4x^3 + 6x^2 - 3x, \end{aligned}$$

which agrees with our computation above.

7.1.7 Derangements

The reader will recall from Section 6.1.3 that a derangement is a permutation in which no object is put into its proper position. We shall show how to calculate the number of derangements D_n of a set of n objects by use of the principle of inclusion and exclusion. Consider the set A of all permutations of the n objects. Let a_i be the property that object i is placed in the ith position. Thus,

$$D_n = N(a_1' a_2' \cdots a_n').$$

We have

$$N = n!,$$

the number of permutations. Also,

$$N(a_i) = (n-1)!,$$

for a permutation in which object i returns to its original position is equivalent to a permutation of the remaining objects. Similarly,

$$N(a_i a_j) = (n-2)!$$

and

$$N(a_{i_1} a_{i_2} \cdots a_{i_t}) = (n-t)!.$$

Hence,

$$\sum N(a_{i_1} a_{i_2} \cdots a_{i_t}) = \binom{n}{t}(n-t)!,$$

since there are $\binom{n}{t}$ choices for the properties $a_{i_1}, a_{i_2}, \ldots, a_{i_t}$. It follows by the principle of inclusion and exclusion that

$$\begin{aligned} D_n &= N - \sum N(a_i) + \sum N(a_i a_j) - \sum N(a_i a_j a_k) \pm \cdots + (-1)^n N(a_1 a_2 \cdots a_n) \\ &= n! - \binom{n}{1}(n-1)! + \binom{n}{2}(n-2)! - \binom{n}{3}(n-3)! \pm \cdots + (-1)^n \binom{n}{n}(n-n)!. \end{aligned}$$

Simplifying, we have

$$\begin{aligned} D_n &= n! - \frac{n!}{1!(n-1)!}(n-1)! + \frac{n!}{2!(n-2)!}(n-2)! - \frac{n!}{3!(n-3)!}(n-3)! \pm \cdots \\ &\quad + (-1)^n \frac{n!}{n!(n-n)!}(n-n)! \\ &= n!\left[1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} \pm \cdots + (-1)^n \frac{1}{n!}\right], \end{aligned}$$

as we have seen previously.

7.1.8 Counting Combinations

In Section 5.3 we studied a variety of counting problems which we solved by means of generating functions. Here we note that the principle of inclusion and exclusion can also be applied to such problems. We illustrate the method by means of an example. Suppose that we are doing a survey and we have three teachers, four plumbers, and six autoworkers whom we are considering interviewing. Suppose that we consider two workers with the same job to be indistinguishable, and we seek the number of ways to choose 11 workers to interview. By the methods of Chapter 5, this can be computed by finding the coefficient of x^{11} in the generating function

$$(1+x+x^2+x^3)(1+x+x^2+x^3+x^4)(1+x+x^2+x^3+x^4+x^5+x^6). \quad (7.10)$$

However, we shall compute it in a different way.

Consider the case where there are infinitely many teachers, plumbers, and autoworkers available, and consider the set A consisting of all the ways of choosing 11 workers to interview. For a particular element of the set A, say it satisfies property a_1 if it uses at least four teachers, property a_2 if it uses at least five plumbers, and property a_3 if it uses at least seven autoworkers. We seek to count all elements of the set A satisfying none of the properties a_i; thus we seek $N(a_1'a_2'a_3')$. To compute this, note that by Theorem 5.4,

$$N = |A| = \binom{3+11-1}{11} = \binom{13}{11} = 78.$$

What is $N(a_1)$? Note that a choice satisfies a_1 if and only if it has at least four teachers. Such a choice is equivalent to choosing seven (arbitrary) workers when there are infinitely many workers of each kind, so can be done, by Theorem 5.4, in

$$\binom{3+7-1}{7} = \binom{9}{7} = 36$$

ways. Thus,

$$N(a_1) = 36.$$

Similarly, a choice satisfying a_2 is equivalent to a choice of six workers when there are infinitely many of each kind, so

$$N(a_2) = \binom{3+6-1}{6} = \binom{8}{6} = 28.$$

Finally,

$$N(a_3) = \binom{3+4-1}{4} = \binom{6}{4} = 15.$$

Next, a choice satisfying both a_1 and a_2 has at least four teachers and at least five plumbers, so is equivalent to a choice of two workers when each is in infinite supply. Thus,

$$N(a_1a_2) = \binom{3+2-1}{2} = \binom{4}{2} = 6.$$

Similarly, a choice satisfying a_1 and a_3 is equivalent to a choice of no workers when each is in infinite supply; as there is exactly one way to choose no workers,

$$N(a_1a_3) = 1.$$

Also, there is no way to choose 11 workers, at least 5 of whom are plumbers and at least 7 of whom are autoworkers, so

$$N(a_2a_3) = 0.$$

Similarly,

$$N(a_1a_2a_3) = 0.$$

Thus, by the principle of inclusion and exclusion, the desired number of choices is

$$78 - (36 + 28 + 15) + (6 + 1 + 0) - 0 = 6.$$

It is easy to check this result by computing the coefficient of x^{11} in the generating function (7.10).

7.1.9 Rook Polynomials[5]

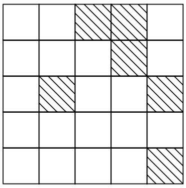
In Examples 5.10, 5.14, and 6.15, we studied the notion of rook polynomial of a board B consisting of acceptable (darkened) or unacceptable squares. If as in Figures 5.1 and 5.2, the board in B has a predominance of darkened squares, it is useful to consider the *complementary board* B' of B, the board obtained from B by interchanging acceptable and forbidden squares. Suppose that B is an $n \times m$ board and we are interested in $r_n(B)$, the number of ways to place n nontaking rooks on acceptable squares of the board B. We shall show that we can obtain $r_n(B)$ from $R(x, B')$, the rook polynomial of the complementary board, rather than from $R(x, B)$. The former rook polynomial, based on a board with fewer darkened squares, will be easier to compute. We may assume that $n \le m$. For if $n > m$, then $r_n(B) = 0$. Note that we shall not be computing $r_j(B)$ for $j < n$, only for the special case $j = n$. In Exercise 37 of Section 7.2 the reader is asked to generalize the results to arbitrary $r_j(B)$, $j \le n$.

Let us say that an *assignment* of n nontaking rooks to an $n \times m$ board B means that each rook is placed in a square, acceptable or not, with no two rooks in the same row and no two rooks in the same column. There are

$$P(m, n) = m(m-1)\cdots(m-n+1)$$

possible assignments. For we choose one of m positions in the first row, then one of $m-1$ positions in the second row, ..., and, finally, one of $m-n+1$ positions in the nth row. Let A be the set of all possible assignments for board B, and let a_i be the property that an assignment has a rook in a forbidden square in the ith column. Then $r_n(B)$ is given by the number of assignments having none of the

[5]This subsection may be omitted.



B'

Figure 7.5: The complementary board B' of the board B of Figure 5.1.

properties $a_1, a_2, \ldots, a_m$. We compute this number using the principle of inclusion and exclusion.

Given t, we shall see how to compute

$$\sum N(a_{i_1} a_{i_2} \cdots a_{i_t}),$$

where t must, of course, be at most m. Note that this sum is 0 if $t > n$ because there could be no assignment with rooks in t different columns, let alone in a forbidden square in t different columns. If $t \leq n$, consider the complementary board B'. An assignment of n nontaking rooks to B with a rook in forbidden position in each of t columns of B corresponds to an assignment of t nontaking rooks to acceptable squares of the board B'—this can be done in $r_t(B')$ ways—and then an arbitrary placement of the remaining $n-t$ rooks in any of the remaining $m-t$ columns—this can be done in $P(m-t, n-t)$ ways. Thus, for $t \leq n$,

$$\sum N(a_{i_1} a_{i_2} \cdots a_{i_t}) = P(m-t, n-t) r_t(B').$$

By the principle of inclusion and exclusion, we conclude that

$$\begin{aligned} r_n(B) = P(m, n) - P(m-1, n-1) r_1(B') \pm \cdots + \\ (-1)^t P(m-t, n-t) r_t(B') \pm \cdots + (-1)^n P(m-n, 0) r_n(B'). \end{aligned} \tag{7.11}$$

Let us apply this result to the 5×5 board B of Figure 5.1 and compute $r_5(B)$. The complementary board B' is shown in Figure 7.5. By the reduction results of Exercise 17, Section 5.1, and Example 5.14, one can show that

$$R(x, B') = 1 + 6x + 11x^2 + 6x^3 + x^4. \tag{7.12}$$

Using (7.12), noting that $P(a, a) = a!$, and applying (7.11), we have

$$r_5(B) = 5! - 4!(6) + 3!(11) - 2!(6) + 1!(1) - 0!(0) = 31.$$

EXERCISES FOR SECTION 7.1

1. Three premium cable television channels, A, B, and C, are available in a city. The following results were obtained in a survey of the households of the city: 20 percent

subscribed to A, 16 percent to B, 14 percent to C, 8 percent to both A and B, 5 percent to both A and C, 4 percent to both B and C, and 2 percent to all three. What percentage of the households subscribed to none of the channels?

2. In an experiment, there are two kinds of treatments, the controls and the noncontrols. There are 3 controls and 80 experimental units or blocks. Each control is used in 25 blocks, each pair of controls is used in the same block 11 times, and all three controls are used in the same block together 12 times. In how many blocks are none of the controls used? (We shall study similar conditions on experimental designs in detail in Chapter 9.)

3. A cigarette company surveys 200,000 people. Of these, 130,000 are males, according to the company's report. Also, 90,000 are smokers and 10,000 of those surveyed have cancer. However, of those surveyed, there are 7000 males with cancer, 8000 smokers with cancer, and 5000 male smokers. Finally, there are 1000 male smokers with cancer. How many female nonsmokers without cancer are there? Is there something wrong with the cigarette company's report?

4. Eight hundred people were tested for immunity to the diseases tuberculosis, rubella, and smallpox. Of the 800 people, 350 were found to have immunity to tuberculosis, 450 to rubella, 450 to smallpox, 150 to tuberculosis and rubella, 200 to rubella and smallpox, 250 to tuberculosis and smallpox, and 100 to tuberculosis, rubella, but not smallpox. How many people were found to have immunity to none of the diseases?

5. (a) Suppose that among 1000 households surveyed, 30 have neither an exercise bicycle nor a treadmill, 50 have only an exercise bicycle, and 60 have only a treadmill. How many households have both?

   (b) How many arrangements of the digits $0, 1, 2, \ldots, 9$ are there in which the first digit is greater than 2 and the last digit is less than 7?

   (c) How many DNA sequences of length 10 are there with at least one of each base A, G, C, T?

6. Find an expression for the number of objects in a set A which have at least one of the properties $a_1, a_2, \ldots, a_r$.

7. One hundred twenty water samples were tested for traces of three different types of chemicals: mercury, arsenic, and lead. Of the 120 samples, 17 were found to have mercury, 15 to have arsenic, 14 to have lead, 10 to have mercury and arsenic, 7 to have arsenic and lead, 15 to have mercury and lead, and 5 to have mercury, arsenic, but no lead. How many samples had a trace of at least one of the three chemicals?

8. Of 100 cars tested at an inspection station, 9 had defective headlights; 8 defective brakes; 7 defective horns; 2 defective windshield wipers; 4 defective headlights and brakes; 3 defective headlights and horns; 2 defective headlights and windshield wipers; 3 defective brakes and horns; none defective brakes and windshield wipers; 1 defective horn and windshield wipers; 1 defective headlights, brakes and horn; 1 defective headlights, horn, and windshield wipers; and none had any other combination of defects. Find the number of cars that had at least one of the defects in question.

9. A total of 100 students at a college were interviewed. Of these, 38 were taking a French course; 45 were taking a physics course; 28 a mathematics course; 25 a history course; 22 were taking French and physics; 23 French and mathematics; 10 physics and mathematics; 1 French and history; 21 physics and history; 14 mathematics

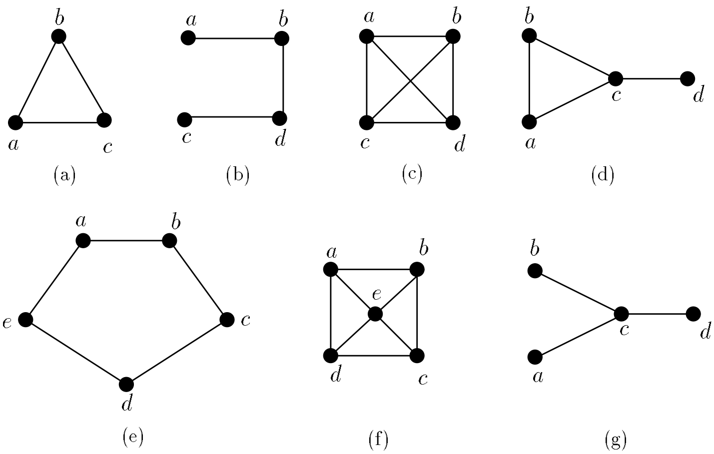


Figure 7.6: Graphs for exercises of Section 7.1.

and history; 11 French, physics, and mathematics; 8 French, physics, and history; 6 French, mathematics, and history; 6 physics, mathematics, and history; and 5 were taking courses in all four subjects. How many students were taking at least one course in the subjects in question?

10. A troubleshooter has pinpointed three 3 files as the source of potential problems on a computer. He has used each file in a test 12 times, each pair of files in the same test together 6 times, and all 3 files in the same test together 4 times. In 8 tests, none of the files were used. How many tests were performed altogether?

11. How many integers between 1 and 10,000 inclusive are divisible by none of 5, 7, and 11?

12. How many integers between 1 and 600 inclusive are divisible by none of 2, 3, and 5?

13. How many integers between 1 and 600 inclusive are divisible by none of 2, 3, 5, and 7?

14. Nine accidents occur during a week. Write an expression for computing the probability that there is at least one accident each day.

15. A total of six misprints occur on five pages of a book. What is the probability that each of these pages has at least one misprint?

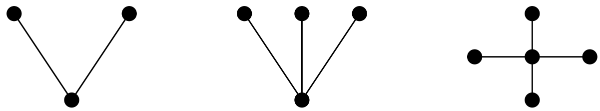
16. Twenty light particles hit a section of the retina that has nine cells. What is the probability that at least one cell is not hit by a light particle?

17. How many permutations of $\{1, 2, 3, 4, 5, 6\}$ have the property that $i+1$ never immediately follows i?

18. Use the principle of inclusion and exclusion (not Theorem 7.3) to find the chromatic polynomial of each of the graphs of Figure 7.6.

19. Use Theorem 7.3 to find the chromatic polynomial of each of the graphs of Figure 7.6.

20. The *star* $S(1, n)$ is the graph consisting of one central vertex and n neighboring vertices, with no other edges. Figure 7.7 shows some stars. Find the chromatic polynomial of $S(1, n)$ using the methods of Section 7.1.6.



$S(1,2)$ $S(1,3)$ $S(1,4)$

Figure 7.7: Some stars.

21. Use the principle of inclusion and exclusion to count the number of ways to choose:

    (a) 8 elements from a set of 4 a's, 4 b's, and 5 c's

    (b) 9 elements from a set of 3 a's, 4 b's, and 5 c's

    (c) 12 elements from a set of 6 a's, 6 b's, and 4 c's

22. Verify Equation (7.12).

23. Use Equation (7.11) to compute $r_n(B)$ if B is the $n \times n$ board with all squares acceptable.

24. Use Equation (7.11) and earlier reduction results to compute $r_5(B)$ for board B of Figure 5.2.

25. Suppose that $r = 3$ and that an object has exactly two properties. How many times is the object counted in computing:

    (a) $\sum N(a_i)$? (b) $\sum N(a_i a_j)$?

26. Suppose that $r = 8$ and that an object has exactly three properties. How many times is the object counted in computing:

    (a) $\sum N(a_i)$? (b) $\sum N(a_i a_j)$? (c) $\sum N(a_i a_j a_k)$? (d) $\sum N(a_i a_j a_k a_l)$?

27. Suppose that d distinguishable CDs are placed in n distinguishable CD players. More than one CD can go in a player. We distinguish placing CD_5 in player 1 from placing CD_5 in player 2, and also distinguish placing CD_5 in player 1 from placing CD_6 in player 1, and so on. Suppose that the CDs are distributed so that no players are empty. In how many ways can this be done?

28. Use inclusion and exclusion to find the number of solutions to the equation

$$x_1 + x_2 + x_3 = 15$$

    in which each x_i is a nonnegative integer and $x_i \leq 7$.

29. Use inclusion and exclusion to find the number of solutions to the equation

$$x_1 + x_2 + x_3 + x_4 = 18$$

    in which each x_i is a positive integer and $x_i \leq 8$.

30. Find the number of n-digit codewords from the alphabet $\{0, 1, 2, \ldots, 9\}$ in which the digits 1, 2, and 3 each appear at least once.

31. (a) Find the number of onto functions from a set with five elements to a set with three elements.

(b) If m and n are positive integers, find a formula for the number of onto functions from a set with m elements to a set with n elements.

32. Recall that every positive integer n can be written in a unique way as the product of powers of primes,

$$n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r},$$

where $p_1, p_2, \ldots, p_r$ are distinct primes and $e_i \geq 1$, all i. The *Moebius function* $\mu(n)$ is defined by

$$\mu(n) = \begin{cases} 1 & \text{if } n = 1 \\ 0 & \text{if } e_i > 1, \text{ any } i \\ (-1)^r & \text{if } e_1, e_2, \ldots, e_r \text{ all equal } 1. \end{cases}$$

Thus, $\mu(100) = 0$ since 2^2 is a factor of 100, and $\mu(30) = \mu(2 \cdot 3 \cdot 5) = (-1)^3 = -1$.

(a) Show from the principle of inclusion and exclusion that

$$\sum_{d|n} \mu(d) = \begin{cases} 1 & \text{if } n = 1 \\ 0 & \text{if } n > 1, \end{cases} \tag{7.13}$$

where the sum in (7.13) is taken over all integers d that divide n. For example,

$$\begin{aligned} \sum_{d|12} \mu(d) &= \mu(1) + \mu(2) + \mu(3) + \mu(4) + \mu(6) + \mu(12) \\ &= 1 + (-1) + (-1) + 0 + (-1)^2 + 0 \\ &= 0. \end{aligned}$$

(b) Suppose that f and g are functions such that

$$f(n) = \sum_{d|n} g(d).$$

Show from the result in part (a) that

$$g(n) = \sum_{d|n} f\left(\frac{n}{d}\right) \mu(d). \tag{7.14}$$

Equation (7.14) is called the *Moebius inversion formula.* For generalizations of this formula of significance in combinatorics, see Rota [1964] (see also Berge [1971], Hall [1986], and Liu [1972]).

(c) Show that if $\phi(n)$ is the Euler ϕ function, then

$$n = \sum_{d|n} \phi(d).$$

(d) Conclude that

$$\phi(n) = \sum_{d|n} \mu(d) \frac{n}{d}.$$

(e) Show that

$$\phi(p^c) = p^c \left(1 - \frac{1}{p}\right).$$

33. (Cohen [1978]) Each of n gentlemen checks both a hat and an umbrella. The hats are returned at random and then the umbrellas are returned at random independently. What is the probability that no man gets back both his hat and his umbrella?

34. Exercises 34 and 35 consider permutations with restrictions on certain patterns. To say that the *pattern* uv *does not appear* in a permutation $j_1 j_2 j_3 \cdots j_n$ of $\{1, 2, \ldots, n\}$ means that $j_i j_{i+1}$ is never uv. Similarly, to say that the *pattern* uvw *does not appear* means that $j_i j_{i+1} j_{i+2}$ is never uvw. Let b_n be the number of permutations of the set $\{1, 2, \ldots, n\}$ in which the patterns $12, 23, \ldots, (n-1)n$ do not appear. Find b_n.

35. Find the number of permutations of $\{1, 2, 3, 4, 5, 6\}$ in which neither the pattern 125 nor the pattern 34 appears (see Exercise 34).

36. Find the number of ways in which the letters a, a, b, b, c, c, c, d, d can be arranged so that two letters of the same kind never appear consecutively.

37. How many codewords of length 9 from the alphabet $\{0, 1, 2\}$ have three of each digit, but no three consecutive digits the same?

38. How many RNA chains have two A's, two U's, two C's, and two G's, and have no repeated base?

39. In our study of partitions of an integer (Exercises 12–16, Section 5.3), let $p^*(k)$ be the number of partitions of k with distinct integers and $p_0(k)$ be the number of partitions of k with odd integers.

    (a) Define a set A and properties a_i and b_i for elements of A so that
    $$p_0(k) = N(a_1' a_2' \cdots)$$
    and
    $$p^*(k) = N(b_1' b_2' \cdots).$$

    (b) Show that A, a_i, and b_i in part (a) can be chosen so that
    $$N(a_{i_1} a_{i_2} \cdots a_{i_k}) = N(b_{i_1} b_{i_2} \cdots b_{i_k})$$
    for all k. Conclude that $p_0(k) = p^*(k)$. (This result was derived by generating functions in Exercise 15, Section 5.3.)

40. (Shen [1998]) A sphere is colored in two colors: 10 percent of its surface is white, the remaining part is black. Prove that there is a cube inscribed in the sphere such that all its 8 vertices are black. (*Hint:* Consider Example 7.6.)

7.2 THE NUMBER OF OBJECTS HAVING EXACTLY m PROPERTIES

7.2.1 The Main Result and Its Applications

Let us return to the general situation of a set of N objects, each of which may or may not have each of r different properties, a_1, a_2, $\ldots$, a_r. There are situations where we want to know how many objects have exactly m of these properties. Let

e_m be the number of objects having exactly m properties, $m \le r$. To express a formula for e_m, suppose that for $t \ge 1$, we let

$$s_t = \sum N(a_{i_1} a_{i_2} \cdots a_{i_t}),$$

where the sum is taken over all choices of t distinct properties $a_{i_1}, a_{i_2}, \ldots, a_{i_t}$. Then we have the following theorem.

Theorem 7.4 The number of objects having exactly m properties if there are r properties and $m \le r$ is given by

$$\begin{aligned} e_m = s_m - \binom{m+1}{1} s_{m+1} + \binom{m+2}{2} s_{m+2} - \binom{m+3}{3} s_{m+3} \pm \cdots \\ + (-1)^p \binom{m+p}{p} s_{m+p} \pm \cdots + (-1)^{r-m} \binom{m+r-m}{r-m} s_r. \end{aligned} \tag{7.15}$$

The reader should note that if s_0 is taken to be N, Theorem 7.4 yields the principle of inclusion and exclusion as a special case when $m = 0$.

We prove this theorem in Section 7.2.2. Here, let us apply it to several examples. In particular, let us return to Example 7.2. How many cars exceed the environmental standards on exactly one pollutant? We seek e_1. To compute e_1, we note that by the computation in the example,

$$\begin{aligned} s_1 &= 6 + 4 + 3 = 13, \\ s_2 &= 3 + 2 + 1 = 6, \\ s_3 &= 1. \end{aligned}$$

Thus, by Theorem 7.4,

$$\begin{aligned} e_1 &= s_1 - \binom{2}{1} s_2 + \binom{3}{2} s_3 \\ &= 13 - 2(6) + 3(1) \\ &= 4. \end{aligned}$$

Example 7.8 The Hatcheck Problem (Example 6.9 Revisited) In Example 6.9, we considered a situation in which the hats of n gentlemen are returned at random. In this situation, let us compute the number of ways in which exactly one gentleman gets his hat back. Let us consider the set A of possible ways of returning hats to gentlemen if they are returned at random—these correspond to permutations—and let a_i be the property that the ith gentleman gets his own hat back. In Section 7.1.7, in dealing with derangements, we calculated

$$\begin{aligned} N(a_i) &= (n-1)!, \quad \text{all } i, \\ N(a_i a_j) &= (n-2)!, \quad \text{all } i \ne j, \end{aligned}$$

and in general

$$N(a_{i_1} a_{i_2} \cdots a_{i_t}) = (n-t)!.$$

Hence,

$$s_t = \binom{n}{t}(n-t)!$$

since there are $\binom{n}{t}$ ways to pick the t properties $a_{i_1}, a_{i_2}, \ldots, a_{i_t}$. Then by Theorem 7.4, with $r = n$ and $m = 1$, we have

$$\begin{aligned}
e_1 &= s_1 - \binom{2}{1}s_2 + \binom{3}{2}s_3 - \binom{4}{3}s_4 \pm \cdots + (-1)^{n-1}\binom{n}{n-1}s_n \\
&= \binom{n}{1}(n-1)! - \binom{2}{1}\binom{n}{2}(n-2)! + \binom{3}{2}\binom{n}{3}(n-3)! - \binom{4}{3}\binom{n}{4}(n-4)! \\
&\quad \pm \cdots + (-1)^{n-1}\binom{n}{n-1}\binom{n}{n}(n-n)! \\
&= \frac{n!}{1!(n-1)!}(n-1)! - \frac{2!}{1!1!}\frac{n!}{2!(n-2)!}(n-2)! + \frac{3!}{2!1!}\frac{n!}{3!(n-3)!}(n-3)! \\
&\quad - \frac{4!}{3!1!}\frac{n!}{4!(n-4)!}(n-4)! \pm \cdots + (-1)^{n-1}\frac{n!}{(n-1)!1!} \\
&= n! - \frac{n!}{1!} + \frac{n!}{2!} - \frac{n!}{3!} \pm \cdots + (-1)^{n-1}\frac{n!}{(n-1)!} \\
&= n!\left[1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} \pm \cdots + (-1)^{n-1}\frac{1}{(n-1)!}\right] \\
&= nD_{n-1}.
\end{aligned}$$

This result is clear, for we pick one gentleman to get his hat back—this can be done in n ways—and then choose a derangement of the rest of the gentlemen—this can be done in D_{n-1} ways.

We conclude that the probability that exactly one gentleman gets his hat back is

$$\frac{nD_{n-1}}{n!} = \frac{D_{n-1}}{(n-1)!},$$

which approaches $1/e$ as n approaches infinity, since $D_n/n!$ does. Thus, in the long run, the probability that exactly one gentleman gets his hat back is the same as the probability that no gentlemen get their hats back. ∎

Example 7.9 Testing Psychic Powers In some psychic experiments, we present a sequence of n elements in an order unknown to a person who claims to have psychic powers. That person predicts the order in advance. We count the number of correct elements, that is, the number of elements whose place in the sequence is predicted exactly right. Suppose that in a sequence of 10 elements, a person gets five right. Would we take this as evidence of psychic powers? To answer the question, we ask whether the observed number of successes is very unlikely if the person is only guessing. In particular, we ask what is the probability of guessing at least five elements correctly. (We are really interested in how likely it is the person did at least as well as he did.) The number of ways to guess exactly m elements correctly in a sequence of n elements can be computed from Theorem 7.4. We let A be the set of all permutations of the set $\{1, 2, \ldots, n\}$ and a_i be the property that i is in the ith position. Then $N(a_i)$, $N(a_i a_j)$, and so on, are exactly as in our analysis of the hatcheck problem, and so is s_t for every t. Thus, one can show that the probability of guessing exactly m positions correctly is given by

$$P_m^n = \frac{1}{m!}\left[1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} \pm \cdots + (-1)^{(n-m)}\frac{1}{(n-m)!}\right]. \tag{7.16}$$

The detailed verification of (7.16) is left as an exercise (Exercise 26). The probability of guessing at least five positions right out of a sequence of 10 is given by

$$\begin{aligned} P_5^{10}+P_6^{10}+P_7^{10}+P_8^{10}+P_9^{10}+P_{10}^{10} &= .00306+.00052+.00007+.00001+.00000+.00000 \\ &= .00366. \end{aligned}$$

(Notice that the next-to-last .00000 here is in fact exactly 0, while the last is actually 1/10!.) We conclude that the probability of achieving this much success by guessing is *very* small. We would have some evidence to conclude that the person *does* seem to have psychic powers. For further references on tests of psychic powers and other applications of the notion of derangement of interest to psychologists, see Barton [1958], Utts [1991], and Vernon [1936]. ■

Example 7.10 RNA Chains Let us find the number of RNA chains of length n with exactly two U's. We can calculate this number directly. For to get an RNA chain of length n with exactly two U's, we choose two positions out of n for the U's and then have three choices of base for each of the remaining $n-2$ positions. This gives us

$$\binom{n}{2}3^{n-2}$$

chains. It is interesting to see how we can obtain this number from Theorem 7.4. Let A be the set of all n-digit sequences from the alphabet U, A, C, G, and let a_i be the property that there is a U in the ith position. Then we seek e_2. Note that

$$N(a_{i_1}a_{i_2}\cdots a_{i_t}) = 4^{n-t},$$

for we have four choices for the ith element in the chain if $i \neq i_1, i_2, \ldots, i_t$. Hence,

$$s_t = \binom{n}{t}4^{n-t},$$

since there are t properties to choose from n properties. We conclude by Theorem 7.4 that

$$\begin{aligned} e_2 &= s_2 - \binom{3}{1}s_3 + \binom{4}{2}s_4 \mp \cdots + (-1)^p\binom{p+2}{p}s_{p+2} \pm \cdots + (-1)^{n-2}\binom{n}{n-2}s_n \\ &= \binom{n}{2}4^{n-2} - \binom{3}{1}\binom{n}{3}4^{n-3} + \binom{4}{2}\binom{n}{4}4^{n-4} \mp \cdots \\ &\quad + (-1)^p\binom{p+2}{p}\binom{n}{p+2}4^{n-p-2} \pm \cdots + (-1)^{n-2}\binom{n}{n-2}\binom{n}{n}4^{n-n}. \end{aligned}$$

To evaluate this expression for e_2, note that

$$\begin{aligned}(-1)^p\binom{p+2}{p}\binom{n}{p+2}4^{n-p-2} &= (-1)^p\frac{(p+2)!}{p!2!}\frac{n!}{(p+2)!(n-p-2)!}4^{n-p-2}\\ &= (-1)^p\frac{n(n-1)}{2}\frac{(n-2)!}{p!(n-p-2)!}4^{n-p-2}\\ &= \binom{n}{2}(-1)^p\binom{n-2}{p}4^{n-p-2}.\end{aligned}$$

Thus

$$e_2 = \binom{n}{2}\sum_{p=0}^{n-2}\binom{n-2}{p}(-1)^p 4^{n-p-2}.$$

By the binomial expansion (Theorem 2.7),

$$e_2 = \binom{n}{2}(4-1)^{n-2} = \binom{n}{2}3^{n-2}.$$

This result agrees with our initial computation. In this case, use of Theorem 7.4 was considerably more difficult! ■

Example 7.11 Legitimate Codewords (Example 6.4 Revisited) In Example 6.4 we defined a codeword from the alphabet $\{0, 1, 2, 3\}$ as *legitimate* if it had an even number of 0's and we let a_k be the number of legitimate codewords of length k. In Section 6.3.1 we used generating functions to show that

$$a_k = \frac{1}{2}(2)^k + \frac{1}{2}(4)^k.$$

Here, we shall derive the same result using Theorem 7.4. Let A be the set of all sequences of length k from $\{0, 1, 2, 3\}$, and let a_i be the property that the ith digit is 0, $i = 1, 2, \ldots, k = r$. We seek the number of elements of A having an even number of these properties; that is, we seek $e_0 + e_2 + e_4 + \cdots$. To compute this sum, note that

$$s_t = \binom{k}{t}4^{k-t}$$

for

$$N(a_{i_1}a_{i_2}\cdots a_{i_t}) = 4^{k-t}.$$

From this and Theorem 7.4, one can show by algebraic manipulation that

$$e_0 + e_2 + e_4 + \cdots = \frac{1}{2}(2)^k + \frac{1}{2}(4)^k. \tag{7.17}$$

An easier way to show (7.17) is to use the following theorem, whose proof comes in Section 7.2.2. ■

Theorem 7.5 If there are r properties, the number of objects having an even number of the properties is given by

$$e_0 + e_2 + e_4 + \cdots = \frac{1}{2}\left[s_0 + \sum_{t=0}^{r}(-2)^t s_t\right]$$

and the number of objects having an odd number of the properties is given by

$$e_1 + e_3 + e_5 + \cdots = \frac{1}{2}\left[s_0 - \sum_{t=0}^{r}(-2)^t s_t\right].$$

Applying Theorem 7.5 to Example 7.11, and recalling that s_0 is taken to be N, we find using the binomial expansion (Theorem 2.7) that

$$\begin{aligned} e_0 + e_2 + e_4 + \cdots &= \frac{1}{2}\left[4^k + \sum_{t=0}^{k}(-2)^t \binom{k}{t} 4^{k-t}\right] \\ &= \frac{1}{2}\left[4^k + (-2+4)^k\right] \\ &= \frac{1}{2}\left[4^k + 2^k\right], \end{aligned}$$

which agrees with (7.17).

Example 7.12 Cosmic Rays and Occupancy Problems Suppose that we have a Geiger counter with c cells which is exposed to a shower of cosmic rays, getting hit by n rays. What is the probability that exactly q counters will go off? To answer this question, we can follow the analysis in Section 7.1.4, where we introduced a sample space S and events E_i. Here, S consists of all distributions of n rays to c cells, and E_i is the event that counter i is not hit. We want the probability that exactly $m = c - q$ counters are not hit, that is, that exactly m of the events in question occur. We can introduce a set A and properties a_i exactly as in Section 7.1.4, and observe that among events $E_1, E_2, \ldots, E_r$, the probability that exactly m of them will occur can be computed from Theorem 7.4 by using $e_m/N(S)$. In our example, we can compute e_m by thinking of this as an occupancy problem and using the computations for $N(a_i)$, $N(a_i a_j)$, and so on, from Section 7.1.5. Then we find that

$$s_t = \sum N(a_{i_1} a_{i_2} \cdots a_{i_t}) = \binom{c}{t}(c-t)^n.$$

Thus, one can show from Theorem 7.4 that the probability that exactly m of the events $E_1, E_2, \ldots, E_r$ will occur is given by

$$\binom{c}{m}\sum_{p=0}^{c-m}(-1)^p \binom{c-m}{p}\left(1 - \frac{m+p}{c}\right)^n. \tag{7.18}$$

A detailed verification is left to the reader (Exercise 28). The result in (7.18) can also be derived directly from the case $m = 0$ (see Exercise 29). ■

For a variety of other applications of Theorem 7.4, see Feller [1968], Irwin [1955], or Parzen [1992].

7.2.2 Proofs of Theorems 7.4 and 7.5[6]

We close this section by presenting proofs of Theorems 7.4 and 7.5.

Proof of Theorem 7.4. The proof is similar to the proof of Theorem 7.1. As a preliminary, we note that

$$\begin{aligned}\binom{m+j}{m+p}\binom{m+p}{p} &= \frac{(m+j)!}{(m+p)!(j-p)!}\frac{(m+p)!}{p!m!} \\ &= \frac{(m+j)!}{m!p!(j-p)!} \\ &= \frac{(m+j)!}{m!j!}\frac{j!}{p!(j-p)!} \\ &= \binom{m+j}{m}\binom{j}{p}.\end{aligned}$$

Thus,

$$\binom{m+j}{m+p}\binom{m+p}{p} = \binom{m+j}{m}\binom{j}{p}. \tag{7.19}$$

Let us now consider Equation (7.15). If an object has fewer than m properties a_i, it is not counted in calculating e_m and it is not counted in any of the terms in the right-hand side of (7.15). Suppose that an object has exactly m of the properties. It is counted exactly once in calculating e_m, and counted exactly once in calculating the right-hand side of (7.15), namely in calculating s_m. Finally, suppose that an object has more than m properties, say $m+j$ properties. It is not counted in calculating e_m. We shall argue that the number of times it is counted in the right-hand side of (7.15) is 0. The object is counted $\binom{m+j}{m}$ times in calculating s_m: It is counted once for every m properties we can choose out of the $m+j$ properties the object has. It is counted $\binom{m+j}{m+1}$ times in calculating s_{m+1}. In general, it is counted $\binom{m+j}{m+p}$ times in calculating s_{m+p} for $p \leq j$. It is not counted otherwise. Hence, the total number of times the object is counted in the right-hand side of (7.15) is calculated by multiplying $\binom{m+j}{m+p}$ by $(-1)^p\binom{m+p}{p}$, the coefficient of s_{m+p}, and adding these terms for $p=0$ up to j. We obtain

$$\begin{aligned}&\binom{m+j}{m} - \binom{m+j}{m+1}\binom{m+1}{1} + \binom{m+j}{m+2}\binom{m+2}{2} \mp \cdots \\ &+ (-1)^p\binom{m+j}{m+p}\binom{m+p}{p} \pm \cdots + (-1)^j\binom{m+j}{m+j}\binom{m+j}{j}.\end{aligned} \tag{7.20}$$

Now by (7.19), (7.20) becomes

$$\binom{m+j}{m} - \binom{m+j}{m}\binom{j}{1} + \binom{m+j}{m}\binom{j}{2} \mp \cdots + (-1)^j\binom{m+j}{m}\binom{j}{j},$$

[6] This subsection may be omitted.

which equals

$$\binom{m+j}{m}\left[\binom{j}{0}-\binom{j}{1}+\binom{j}{2}\mp\cdots+(-1)^j\binom{j}{j}\right]. \tag{7.21}$$

By Theorem 2.9, the bracketed material in (7.21) equals 0 [it arises by expanding $(1-1)^j$ using the binomial expansion], so (7.21) is 0. This completes the proof of Theorem 7.4. Q.E.D.

Proof of Theorem 7.5. Let $E(x)=\sum e_m x^m$ be the ordinary generating function for the sequence $e_0, e_1, e_2, \ldots$. By Theorem 7.4,

$$\begin{aligned}
E(x) = {} & [s_0 - s_1 + s_2 - \cdots + (-1)^r s_r] \\
& + \left[s_1 - \binom{2}{1}s_2 + \binom{3}{2}s_3 - \cdots + (-1)^{r-1}\binom{r}{r-1}s_r\right]x \\
& + \left[s_2 - \binom{3}{1}s_3 + \binom{4}{2}s_4 - \cdots + (-1)^{r-2}\binom{r}{r-2}s_r\right]x^2 \\
& + \cdots \\
& + \left[s_m - \binom{m+1}{1}s_{m+1} + \binom{m+2}{2}s_{m+2} - \cdots + (-1)^{r-m}\binom{m+r-m}{r-m}s_r\right]x^m \\
& + \cdots \\
& + s_r x^r \\
= {} & s_0 \\
& + s_1[x-1] \\
& + s_2\left[x^2 - \binom{2}{1}x + 1\right] \\
& + s_3\left[x^3 - \binom{3}{1}x^2 + \binom{3}{2}x - 1\right] \\
& + \cdots \\
& + s_m\left[x^m - \binom{m}{1}x^{m-1} + \binom{m}{2}x^{m-2} + \cdots + (-1)^{m-1}\binom{m}{m-1}x + (-1)^m\right] \\
& + \cdots \\
& + s_r\left[x^r - \binom{r}{1}x^{r-1} + \binom{r}{2}x^{r-2} + \cdots + (-1)^{r-1}\binom{r}{r-1}x + (-1)^r\right].
\end{aligned}$$

Thus,

$$E(x) = \sum_{m=0}^{r} s_m (x-1)^m. \tag{7.22}$$

The first part of the theorem follows by noting that

$$e_0 + e_2 + e_4 + \cdots = \frac{1}{2}\left[E(1) + E(-1)\right]$$

and taking $x = 1$ and $x = -1$ in (7.22). The second part of the theorem follows by noting that

$$e_1 + e_3 + e_5 + \cdots = \frac{1}{2}\left[E(1) - E(-1)\right].$$

Q.E.D.

EXERCISES FOR SECTION 7.2

1. In Exercise 1, Section 7.1, what percentage of the households subscribe to exactly one of the channels?
2. In Exercise 2, Section 7.1, how many blocks use exactly two controls?
3. In Exercise 4, Section 7.1, how many people were immune to exactly one of the diseases?
4. In Exercise 8, Section 7.1, find the number of cars having exactly two of the defects in question.
5. In Exercise 9, Section 7.1, find the number of students taking exactly three of the subjects in question.
6. How many words of length 6 have an even number of vowels?
7. A variant of Montmort's "probleme des rencontres" discussed in Section 6.1.3 is the following. A deck of n cards is laid out in a row on the table. Cards of a second deck with n cards are placed one by one at random on top of the first set of cards. You get m points if there are m matches between the first and second decks.
   (a) How many ways are there to get 2 points if $n = 4$?
   (b) What is the probability of getting 7 points if $n = 9$?
8. The names on the files of 10 different job candidates appearing for an interview were unfortunately lost, and a new receptionist placed the names on the files at random. In how many ways could this be done so that exactly 3 candidates' files were labeled properly?
9. In the hatcheck problem, use our formula for e_1 to determine the probability that exactly one gentleman gets his hat back if there are 3 gentlemen.
10. In the hatcheck problem, if there are 4 gentlemen, compute the number of ways that exactly 2 of them will get their hats back.
11. Compute e_m for the hatcheck problem for arbitrary m.
12. (a) If four fair coins are tossed, use Theorem 7.4 to compute the probability that there will be exactly 2 heads.

(b) Check your answer by computing it directly.

13. Use Theorem 7.4 to compute the number of ways to get exactly m heads if a coin is tossed n times.

14. (a) Use Theorem 7.4 to find the number of permutations of $\{1, 2, 3, 4, 5, 6, 7, 8\}$ in which exactly 4 integers are in their natural positions.

    (b) Check your answer by computing it directly.

15. (a) Use Theorem 7.4 to compute the number of legitimate codewords of length 7 from the alphabet $\{0, 1, 2\}$ if a codeword is legitimate if and only if it has exactly three 1's.

    (b) Check your answer by an alternative computation.

16. (a) Use Theorem 7.4 to compute the number of legitimate codewords of length n from the alphabet $\{0, 1, 2\}$ if a codeword is legitimate if and only if it has exactly five 1's.

    (b) Check your answer by an alternative computation.

17. (a) Suppose that n children are born to a family. Use Theorem 7.4 to compute the number of ways the family can have exactly 2 boys.

    (b) Check your answer by the methods of Chapter 2.

18. (a) A psychic predicts a sequence of 4 elements, getting 2 right. What is the probability of getting at least this many right?

    (b) What is the probability of getting 3 or more right?

    (c) What is the probability of getting exactly 3 right? (Is there a problem with your answer? Explain.)

19. In a wine-tasting experiment, a taster is told that there will be 5 different wines given to him. After each, he guesses which of the 5 it was, making sure never to repeat a guess. He gets 3 right. What is his probability of getting at least 3 right if he is guessing randomly?

20. Write an expression for the probability that in a sequence of 7 random digits chosen from 0, 1, 2, ..., 9, exactly 2 of the digits will not appear.

21. In a genetics experiment, each mouse in a litter of n mice is classified as belonging to one of M genotypes. What is the probability that exactly g genotypes will be represented among the n mice?

22. Use Theorem 7.5 to find the number of families of 10 children that have an even number of boys. Check your answer by direct computation.

23. Use Theorem 7.5 to find the number of 8-digit sequences from the alphabet $\{0, 1, 2\}$ that have an odd number of 1's. Check your answer by direct computation.

24. Find the number of RNA chains of length 8 that have no U's and an even number of G's.

25. Give an alternative proof of Theorem 7.4 by using mathematical induction.

26. Use Theorem 7.4 to verify Equation (7.16).

27. In Exercise 27, Section 7.1, show that the number of ways to place the CDs so that exactly m players are empty is given by

$$\binom{n}{m} \sum_{i=0}^{n-m} (-1)^i \binom{n-m}{i} (n-m-i)^d.$$

28. Use Theorem 7.4 to verify (7.18).

29. Suppose that $P_m(c, n)$ is the probability that exactly m cells will be empty if n distinguishable balls are distributed into c distinguishable cells.

    (a) Show that
    $$P_m(c, n) = \binom{c}{m}\left(1 - \frac{m}{c}\right)^n P_0(c - m, n).$$

    (b) Derive (7.18) from the equation for $P_0(c - m, n)$.

30. Let e_m^* be the number of elements of the set A having at least m of the properties $a_1, a_2, \ldots, a_r$. Show that
$$e_m^* = s_m - \binom{m}{m-1}s_{m+1} + \binom{m+1}{m-1}s_{m+2} \mp \cdots + (-1)^p\binom{m+p-1}{m-1}s_{m+p} \pm \cdots$$
$$+ (-1)^{r-m}\binom{m+r-m-1}{m-1}s_r.$$

31. Suppose that $E_1, E_2, \ldots, E_r$ are events, that $p_{i_1 i_2 \cdots i_t}$ is the probability that events $E_{i_1}, E_{i_2}, \ldots, E_{i_t}$ all occur, and that $S_t = \sum p_{i_1 i_2 \cdots i_t}$, where the sum in question is taken over all t-element subsets $\{i_1, i_2, \ldots, i_t\}$ of $\{1, 2, \ldots, r\}$. In terms of the S_t, derive expressions for

    (a) The probability that exactly m of the events occur;

    (b) The probability that at least m of the events occur.

32. In Exercise 8, Section 7.1, how many cars have at least 2 of the defects in question?

33. In Exercise 9, Section 7.1, how many students are taking at least 1 of the subjects in question?

34. Compute the number of RNA chains of length 10 with at least 2 U's.

35. Compute the number of legitimate codewords of length 7 from the alphabet $\{0, 1, 2\}$, where a codeword is legitimate if and only if it has at least three 1's.

36. Compute the number of permutations of $\{1, 2, 3, 4, 5\}$ in which at least three integers are in their natural position.

37. Suppose that B' is the complement of the $n \times m$ board B, $n \leq m$. If $j \leq n$, find a formula for $r_j(B)$ in terms of the numbers $r_k(B')$ which generalizes the result of Equation (7.11).

38. Use the result of Exercise 37 to show that
$$R(x, B) = x^n R\left(\frac{1}{x}, B'\right).$$

39. (a) If $E(x)$ is the ordinary generating function for the sequence $e_0, e_1, e_2, \ldots$ and the e_i are defined as in Example 7.11, what is $E(1)$?

    (b) Find a formula for $E(1)$ that holds in general.

REFERENCES FOR CHAPTER 7

Adleman, L. M., and Huang, M.-D., *Primality Testing and Two Dimensional Abelian Varieties over Finite Fields*, Lecture Notes in Mathematics, 1512, Springer-Verlag, Berlin, 1992.

Adleman, L. M., Pomerance, C., and Rumely, R. S., "On Distinguishing Prime Numbers from Composite Numbers, *Ann. Math., 117*, (1983), 173–206.

Agrawal, M., Kayal, N., and Saxena, N., "PRIMES Is in P," Preprint, Aug. 6, 2002. http://www.cse.iitk.ac.in/primality.pdf.

Anshel, I., Anshel, M., and Goldfeld, D., "An Algebraic Method for Public-Key Cryptography," *Math. Res. Lett., 6* (1999), 287–291.

Barton, D. E., "The Matching Distributions: Poisson Limiting Forms and Derived Methods of Approximation," *J. Roy. Statist. Soc., 20* (1958), 73–92.

Berge, C., *Principles of Combinatorics*, Academic Press, New York, 1971.

Birkhoff, G. D., "A Determinant Formula for the Number of Ways of Coloring a Map," *Ann. Math., 14* (1912), 42–46.

Cohen, D. I. A., *Basic Techniques of Combinatorial Theory*, Wiley, New York, 1978.

De Moivre, A., *The Doctrine of Chances*, private printing, London, 1718.

Diffie, W., and Hellman, M. E., "New Directions in Cryptography," *IEEE Trans. Info. Theory, 22* (1976), 644–654.

Feller, W., *An Introduction to Probability Theory and Its Applications*, 3rd ed., Wiley, New York, 1968.

Goldstein, L. J., Schneider, D. I., and Siegel, M. J., *Finite Mathematics and Its Applications*, 7th ed., Prentice Hall, Upper Saddle River, NJ, 2001.

Goldwasser, S., and Kilian, J., "Almost All Primes Can Be Quickly Certified," Proceedings of Annual ACM Symposium on Theory of Computing, 1986, 316–329.

Hall, M., *Combinatorial Theory*, 2nd ed., Wiley, New York, 1986.

Hardy, G. H., and Wright, E. M., *Introduction to the Theory of Numbers*, 5th ed., Oxford University Press, New York, 1980.

Irwin, J. O., "A Unified Derivation of Some Well-Known Frequency Distributions of Interest in Biometry and Statistics," *J. Roy. Statist. Soc., Ser. A, 118* (1955), 389–404.

Koblitz, N., *A Course in Number Theory and Cryptography*, 2nd ed., Springer-Verlag, New York, 1994.

Liu, C. L., *Topics in Combinatorial Mathematics*, Mathematical Association of America, Washington, DC, 1972.

Miller, G. L., "Riemann's Hypothesis and Tests for Primality," *J. Comput. Sys. Sci., 13* (1976), 300–317.

Mizrahi, A., and Sullivan, M., *Finite Mathematics; An Applied Approach*, Wiley, New York, 1999.

Parzen, E., *Modern Probability Theory and Its Applications*, Wiley, New York, 1992.

Rabin, M. O., "Probabilitic Algorithm for Testing Primality," *J. Number Theory, 12* (1980), 128–138.

Reingold, E. M., Nievergelt, J., and Deo, N., *Combinatorial Algorithms: Theory and Practice*, Prentice Hall, Englewood Cliffs, NJ, 1977.

Rhee, M. Y., *Cryptography and Secure Communications*, McGraw-Hill, New York, 1993.

Rivest, R. L., Shamir, A., and Adleman, L. M., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM, 21* (1978), 120–126. (See also U.S. Patent 4,405,829, 1983.)

ROTA, G. C., "On the Foundations of Combinatorial Theory. I. Theory of Möbius Functions," *Z. Wahrscheinlichkeitsth. Verw. Geb., 2* (1964), 340–368.

SCHNEIER, B., *Applied Cryptography: Protocols, Algorithms and Source Code in C*, Wiley, New York, 1995.

SEBERRY, J., and PIEPRZYK, J., *Cryptography: An Introduction to Computer Security*, Prentice Hall, Englewood Cliffs, NJ, 1989.

SHEN, A., "Probabilistic Proofs," *The Mathematical Intelligencer, 20* (1998), 29–31.

SHOR, P. W., "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM J. Computing, 26* (1997), 1484–1509.

SOLOVAY, R., and STRASSEN, V., "A Fast Monte-Carlo Test for Primality," *SIAM J. Computing, 6* (1977), 84–85.

STALLINGS, W., *Cryptography and Network Security: Principles and Practice*, 2nd ed., Prentice Hall, Upper Saddle River, NJ, 1999.

UTTS, J. M., "Replication and Meta-Analysis in Parapsychology," *Statistical Science, 6* (1991), 363–403.

VERNON, P. E., "The Matching Method Applied to Investigations of Personality," *Psychol. Bull., 33* (1936), 149–177.

VILENKIN, N. YA., *Combinatorics*, Academic Press, New York, 1971. (Translated from the Russian by A. Shenitzer and S. Shenitzer.)

WHITNEY, H., "A Logical Expansion in Mathematics," *Bull. Amer. Math. Soc., 38* (1932), 572–579.

Chapter 8

The Pólya Theory of Counting[1]

8.1 EQUIVALENCE RELATIONS

8.1.1 Distinct Configurations and Databases

In this book we have paid a great deal of attention to counting different kinds of configurations. Increasingly, in many fields of science and many areas of scientific application, configurations of various kinds are stored in massive databases. The configurations might be very complex. In medical decisionmaking, we maintain large databases of medical images. In molecular biology, we maintain huge databases of protein structures. In environmental modeling, we keep massive databases of environmental features. Telecommunications and credit card companies maintain gigantic databases of calling and consumption patterns to help discover fraud. The configurations stored in these massive databases are often complex geometrical objects, or objects with a variety of dimensions or properties. The sheer size of the databases encountered makes searching, retrieval, and even just organization a daunting problem. It is sometimes useful to count the number of configurations of a certain kind to help estimate the length of a search in a database. One of the problems encountered is to decide whether or not two configurations are the same. In this chapter we develop techniques for counting the number of distinct configurations of a certain kind. These techniques, of course, make heavy use of the ideas involved in determining precisely whether or not two configurations are the same. Hence, we begin the chapter by studying what it means to say that two things are the same. In the examples in the chapter, it will be much easier to make precise the notion of "sameness" than it is in the examples just described. We will consider notions of sameness for organic molecules, colored (graph-theoretical) trees, switching functions, weak orders, and so on. Because the methods of combinatorics

[1]This chapter should be omitted in an elementary course. In many places, it requires comfort with algebra.

are increasingly important in newer, less precisely defined situations, one can hope that methods such as those described in this chapter will be applicable.

8.1.2 Definition of Equivalence Relations

Suppose that V is a set and S is a set of ordered pairs of elements of V. In Section 4.1.1 we called S a (binary) *relation* on V. For instance, if $V = \{1,2,3\}$ and $S = \{(1,2),(2,3)\}$, then S is a relation on V. We write aSb if the ordered pair (a,b) is in S. Thus, in our example, $1S2$ but not $2S1$ and not $1S3$.

Suppose that V is a set of configurations and that for a, b in V we write aSb to mean that a and b are the same. Then the relation S should have the following properties (previously defined in Section 4.1.2):

Reflexivity: For all a in V, aSa. (Any configuration is the same as itself.)
Symmetry: For all a, b in V, if aSb, then bSa. (If a is the same as b, then b is the same as a.)
Transitivity: For all a, b, c in V, if aSb and bSc, then aSc. (If a is the same as b and b is the same as c, then a is the same as c.)

If S satisfies these three properties, it is called an *equivalence relation.*

We now give several other examples of equivalence relations. Let V be the set of people in New Jersey, and let aSb mean that a and b have the same height. Then S defines an equivalence relation. Let V be the set of all people in the United States, and let aSb mean that a and b have the same birthdate. Then S is an equivalence relation. Let V be the set of all people in the United States and let aSb mean that a is the father of b. Then S does not define an equivalence relation: It is not reflexive, not symmetric, and not transitive. Let V be the set of real numbers and let aSb mean that $a \leq b$. Then S is not an equivalence relation: It is reflexive and transitive but not symmetric.

Let us give some more complicated examples that illustrate the basic problems we discuss in this chapter.

Example 8.1 Coloring a 2×2 Array Let us consider a 2×2 array in which each block is occupied or not. We color the block black if it is occupied and color it white or leave it uncolored otherwise. Figure 8.1 shows several such colorings. Let V be the collection of all such colorings. Let us suppose that we allow rotation[2] of the array by $0°, 90°, 180°$, or $270°$. We consider colored arrays a and b the same, and write aSb, if b can be obtained from a by one of the rotations in question. Then S defines an equivalence relation. To illustrate, note that Figure 8.1 shows some pairs of arrays that are considered in the relation S. To see why S is an equivalence relation, note that it is reflexive because a can be obtained from a by a $0°$ rotation. It is symmetric because if b can be obtained from a by a rotation of x degrees, then a can be obtained from b by a rotation of $360 - x$ degrees. Finally, it is transitive

[2] All rotations in this chapter are counterclockwise unless noted otherwise.

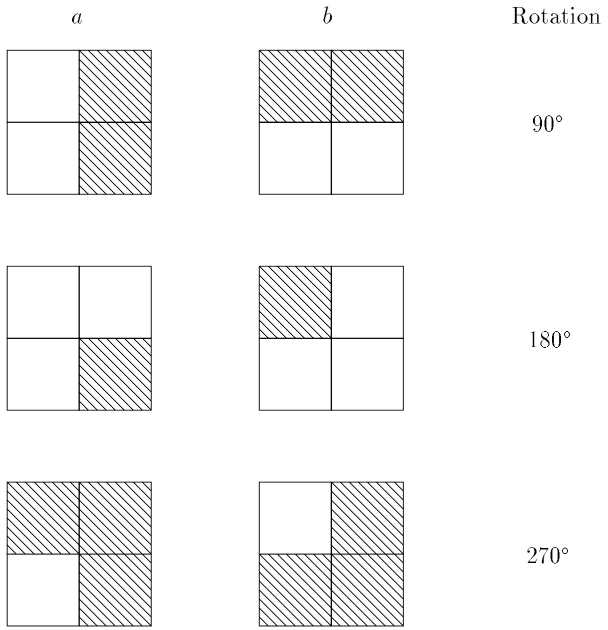


Figure 8.1: Colorings of a 2×2 array. In each case aSb for b is obtained from a by a rotation as indicated.

because if b can be obtained from a by a rotation and c from b by a rotation, then c can be obtained from a by following the first rotation by the second. (It is assumed that a rotation by $360 + x$ degrees is equivalent to a rotation by x degrees.) ■

Example 8.2 Necklaces Suppose that an open necklace consists of a string of k beads, each being either blue or red. Thus, a typical necklace of three beads can be represented by a string such as bbr or brb. A necklace is not considered to have a designated front end, so two such necklaces x and y are considered the same, and we write xSy, if x equals y or if y can be obtained from x by reversing. Thus, bbr is the same as rbb. S defines an equivalence relation. The verification is left to the reader (Exercise 4). ■

Example 8.3 Switching Functions (Example 2.4 Revisited) Recall from Example 2.4 that a switching function of n variables is a function that assigns to every bit string of length n a number 0 or 1. These functions arise in computer engineering. Recall from our discussion in Example 2.4 that certain switching functions are considered equivalent or the same. To make this precise, suppose that T and U are the two switching functions defined by Table 8.1. It is easy to see that $T(x_1x_2) = U(x_2x_1)$ for all bit strings x_1x_2. Thus, T can be obtained from U simply by reordering the input, interchanging the two positions. In this sense, T

Table 8.1: Two Switching Functions, T and U

Bit string x	$T(x)$	$U(x)$
00	1	1
01	0	1
10	1	0
11	1	1

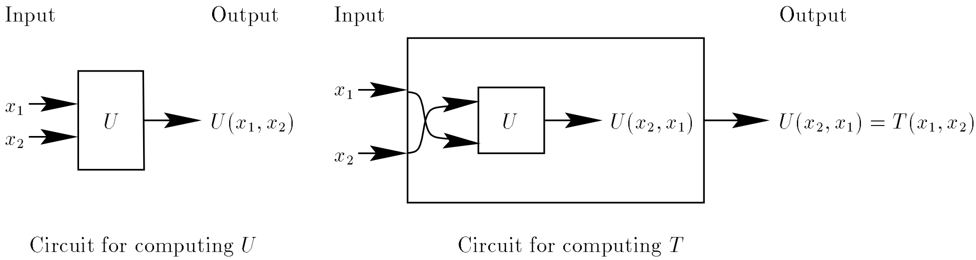


Figure 8.2: A circuit for computing T can be obtained from a circuit for computing U.

and U can be considered equivalent. Indeed, for all practical purposes, they are. For suppose that we can design an electronic circuit which computes U. Then we can design one to compute T, as shown in Figure 8.2, where the circuit computing U is shown as a black box. In general, we consider two switching functions, T and U of two variables the same, and write TSU, if either $T = U$ or $T(x_1x_2) = U(x_2x_1)$ for all bit strings x_1x_2. Then S is an equivalence relation. We leave the proof to the reader (Exercise 5). In what follows, we generalize this concept of equivalence to switching functions of more than two variables. In Section 2.1 we noted that there were many switching functions, even of four variables. Hence, it is impractical to compile a manual listing, for each switching function of n variables, the best corresponding electronic circuit. However, it is not necessary to include every switching function in such a manual, but only enough switching functions so that every switching function of n variables is equivalent to one of the included ones. Counting the number of switching functions required was an historically important problem in computer science (see Section 2.1) and we shall show how to make this computation. ∎

Example 8.4 Coloring Trees[3] Let T be a fixed tree, for instance, the binary tree of seven vertices shown in Figure 8.3. Color each vertex of T black or white, and do not distinguish left from right. Let V be the collection of all colorings of T. Let aSb mean that a and b are considered the same, that is, if b can be obtained

[3] This example is from Reingold, Nievergelt, and Deo [1977].

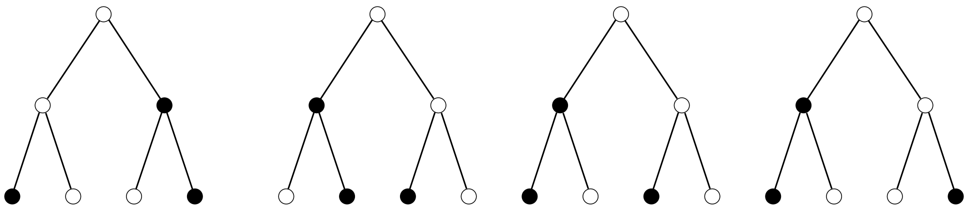


Figure 8.3: Four equivalent colorings of the binary tree of seven vertices.

from a by interchanging left and right subtrees. We shall define this more precisely below. However, since we do not distinguish left from right, it should be clear that all colorings in Figure 8.3 are considered the same. It follows from our general results below that S defines an equivalence relation. ■

Example 8.5 Organic Molecules[4] One of the historically important motivations for the theory developed in this chapter was the desire to count distinct organic molecules in chemistry. Consider the set V of molecules of the form shown in Figure 8.4, where C is a carbon atom and each X can be either CH_3 (methyl), C_2H_5 (ethyl), H (hydrogen), or Cl (chlorine). A typical such molecule is CH_2Cl_2, which has two hydrogen atoms and two chlorine atoms. We can model such a molecule using a regular tetrahedron, a figure consisting of four equilateral triangles that meet at six edges and four corners, as in Figure 8.5. The carbon atom is thought of as being at the center of this tetrahedron and the four components labeled X are at the corners labeled a, b, c, and d. Two such molecules x and y are considered the same, and we write xSy, if y can be obtained from x by one of the following 12 symmetries of the tetrahedron: no change; a rotation by 120° or 240° around a line connecting a vertex and the center of its opposite face (there are eight of these rotations); or a rotation by 180° around a line connecting the midpoints of opposite edges (there are three of these rotations). Figures 8.6 and 8.7 illustrate the second and third kinds of symmetries. ■

Example 8.6 Number of Weak Orders (Example 5.29 Revisited) Recall from Section 4.2 that Figure 4.14 shows a typical weak order R on a set A. Each element has a horizontal level, all elements a and b at the same horizontal level satisfy aRb and bRa, and otherwise, aRb iff a is at a higher level than b. We shall consider two weak orders on a set A to be the same if they have the same number of levels and the same number of elements at corresponding levels. For example, the first two weak orders shown in Figure 8.8 are considered the same. The first and third weak orders are, in fact, identical as weak orders since they have the same set of ordered pairs, $\{(a,c)$, (a,d), (a,e), (a,f), (a,g), (a,h), (b,c), (b,d), (b,e), (b,f), (b,g), (b,h), (c,d), (c,e), (c,f), (c,g), (c,h), (d,g), (d,h), (e,g), (e,h), (f,g),

[4] This example is from Liu [1968]. For more extensive treatment of chemical compounds from the point of view of this chapter, see Pólya and Read [1987].

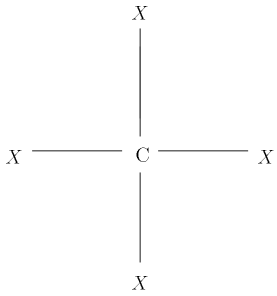


Figure 8.4: An organic molecule.

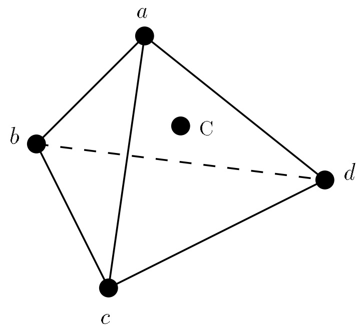


Figure 8.5: A regular tetrahedron.

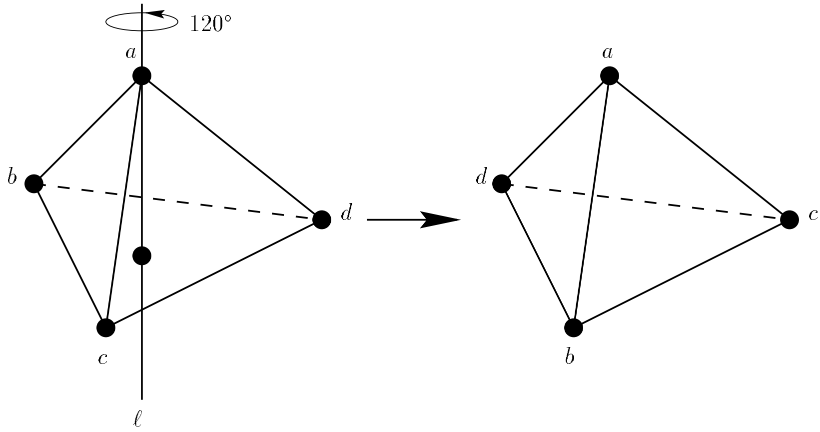


Figure 8.6: Rotation by $120°$ around a line ℓ connecting vertex a and the center of the opposite face.

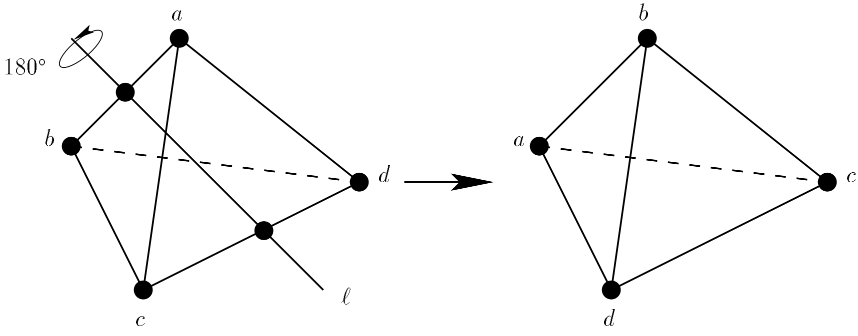


Figure 8.7: Rotation by $180°$ around a line ℓ connecting the midpoints of opposite edges.

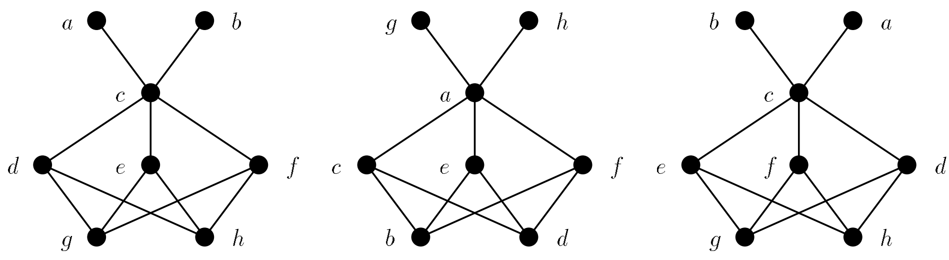


Figure 8.8: Three weak orders considered the same.

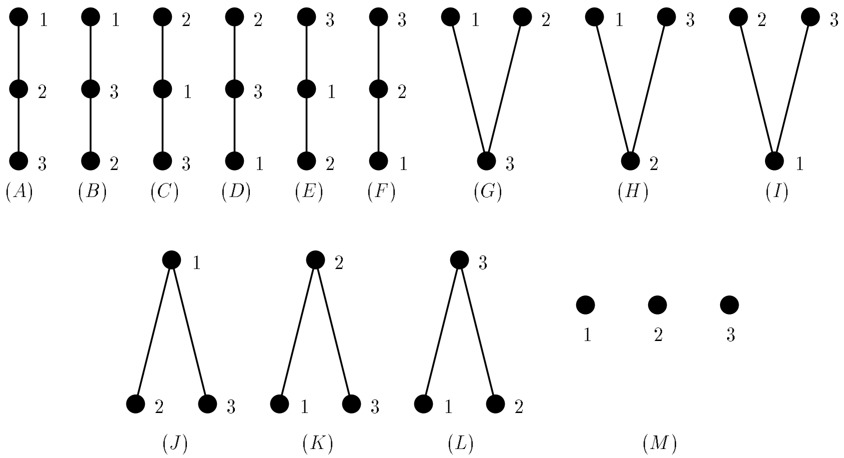


Figure 8.9: All weak orders on $\{1, 2, 3\}$.

$(f, h)\}$. It is easy to see that if aSb means that a and b are the same, S defines an equivalence relation among weak orders. Figure 8.9 shows all possible weak orders on $\{1, 2, 3\}$. Note that, for example, ASD, GSI, JSL. ■

8.1.3 Equivalence Classes

An equivalence relation S on V divides the elements of V into classes called *equivalence classes*. Specifically, if a is any element of V, the *equivalence class containing* a, $C(a)$, consists of all elements b such that aSb, i.e., $C(a) = \{b \in V : aSb\}$. By reflexivity, aSa, so every element of V is in some equivalence class; in particular, $a \in C(a)$. Moreover, for all a, b in V, either $C(a) = C(b)$ or $C(a)$ and $C(b)$ are disjoint. For suppose that x is in both $C(a)$ and $C(b)$. Then aSx and bSx. By symmetry, aSx and xSb. Transitivity now implies that aSb. This shows that $C(a) = C(b)$. For if y is in $C(b)$, then bSy. Now aSb and bSy imply aSy, so y is in $C(a)$. Thus, $C(b) \subseteqq C(a)$. Similarly, we can show that $C(a) \subseteqq C(b)$. Thus, $C(a) = C(b)$. If we now think of $C(a)$ and $C(b)$ as being the same if they have the

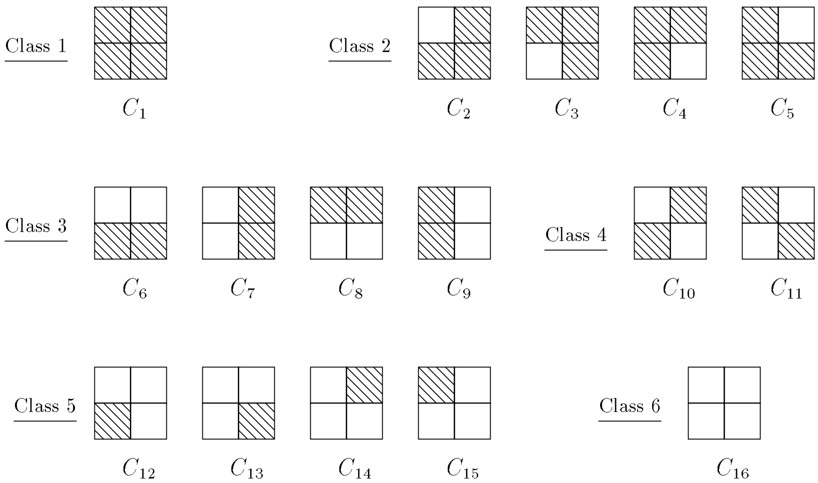


Figure 8.10: Equivalence classes of black-white colorings of the 2×2 array.

same members, we have the following theorem.

Theorem 8.1 If S is an equivalence relation, every element is in one and only one equivalence class.

To illustrate this result, note that in Example 8.2, if the necklaces have length 2, there are four kinds of necklaces, bb, br, rb, and rr. The second and third are equivalent. Thus, for instance, $C(bb) = \{bb\}$ and $C(br) = \{br, rb\}$. There are three distinct equivalence classes, $\{bb\}$, $\{br, rb\}$, and $\{rr\}$.

In Example 8.1, there are six equivalence classes. These are shown in Figure 8.10.

EXERCISES FOR SECTION 8.1

1. In each of the following cases, is S an equivalence relation on V? If not, determine which of the properties of an equivalence relation hold.

   (a) V = real numbers, aSb iff $a = b$.

   (b) V = real numbers, aSb iff $a \neq b$.

   (c) V = real numbers, aSb iff a divides evenly into b.

   (d) V = all subsets of $\{1, 2, \ldots, n\}$, aSb iff a and b have the same number of elements.

   (e) V as in part (d), aSb iff a and b overlap.

   (f) V = all people in the world, aSb iff a is a sibling of b.

   (g) V = all people in the United States, aSb iff a and b have the same blood type.

   (h) $V = \{1, 2, 3, 4\}$, $S = \{(1, 1), (2, 2), (3, 4), (4, 3), (1, 3), (3, 1)\}$.

(i) $V = \{w, x, y, z\}$, $S = \{(x, x),\ (y, y),\ (z, z),\ (w, w),\ (x, z),\ (z, x),\ (x, w),\ (w, x),\ (z, w),\ (w, z)\}$.

(j) V = all residents of California, aSb iff a and b live within 10 miles of each other.

2. Suppose that V is the set of bit strings of length 4, and aSb holds if and only if a and b have the same number of 1's. Is (V, S) an equivalence relation?

3. For each of the following equivalence relations, identify all equivalence classes.

   (a) $V = \{a, b, c, d\}, S = \{(a, a), (b, b), (c, c), (d, d), (a, b), (b, a), (c, d), (d, c)\}$.

   (b) $V = \{u, v, w\}$, $S = \{(u, u),\ (v, v),\ (w, w),\ (u, v),\ (v, u),\ (v, w),\ (w, v),\ (u, w),\ (w, u)\}$.

   (c) $V = \{1, 2, 3, 4\}$, $S = \{(1, 1),\ (2, 2),\ (3, 3),\ (4, 4),\ (1, 2),\ (2, 1),\ (2, 3),\ (3, 2),\ (1, 3),\ (3, 1)\}$.

   (d) V = the set of all positive integers, aSb iff $|a - b|$ is an even number.

   (e) $V = \{1, 2, 3, \ldots, 15\}$, aSb iff $a \equiv b(\text{mod } 3)$.

4. Show that S of Example 8.2 is an equivalence relation.

5. Show that S of Example 8.3 is an equivalence relation among switching functions of two variables.

6. In Example 8.2, identify the equivalence classes of necklaces of length 3.

7. In Example 8.2, identify the equivalence classes of necklaces of length 2 if each bead can be one of three colors: blue, red, or purple.

8. In Example 8.1, suppose that we can use any of three colors: black (b), white (w), or red (r). Describe all equivalence classes of colorings.

9. In Example 8.1, suppose that we allow not only rotations but also reflections in either a vertical, a horizontal, or a diagonal line. (The latter would switch the colors assigned to two diagonally opposite cells.) Identify all equivalence classes of colorings. (Only two colors are used, black and white.)

10. In Example 8.6, identify the equivalence classes of the weak orders of Figure 8.9.

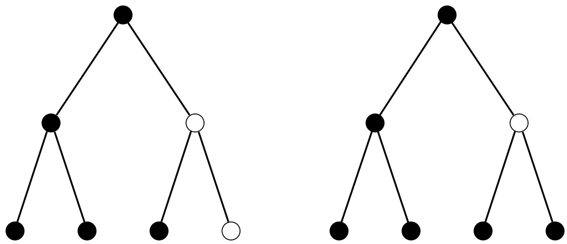
11. In Example 8.6, describe the equivalence classes of weak orders on $\{1, 2, 3, 4\}$.

12. In Example 8.3, identify all equivalence classes of switching functions of two variables.

13. For each tree of Figure 8.11, draw all trees that are equivalent to it in the sense of Example 8.4.

14. The *complement* x' of a bit string x is obtained from x by interchanging all 0's and 1's. For instance, if $x = 00110$, then $x' = 11001$. Suppose that we consider two switching functions T and U of n variables the same if $T = U$ or $T(x) = U(x')$ for every bit string x. Describe all equivalence classes of switching functions under this sameness relation if $n = 3$.

15. Suppose that V is the set of all colorings of the binary tree of Figure 8.12 in which each vertex gets one of the colors black or white. Find all equivalence classes of colorings if two colorings are considered the same if one can be obtained from the other by interchanging the colors of the vertices labeled 1 and 2.



(a) (b)

Figure 8.11: Trees for Exercise 13, Section 8.1.

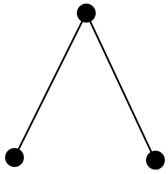


Figure 8.12: Tree for Exercise 15, Section 8.1.

16. In Example 8.5, suppose that a molecule has three hydrogen atoms and one nonhydrogen. How many other molecules with three hydrogens and one nonhydrogen are considered the same as this one?

17. Consider a square and let V be the set of all colorings of its vertices using colors red and blue. For colorings f and g, let fSg hold if g can be obtained from f by rotating the square by $0°$, $90°$, $180°$, or $270°$. Show that (V, S) is an equivalence relation and find all equivalence classes.

18. Generalizing Exercise 17, let V be the set of all colorings of the vertices of a regular p-gon using colors in $\{1, 2, \ldots, n\}$, and let fSg hold for colorings f and g if g can be obtained from f by rotating the p-gon through one of the angles $k(360/p)$ for $k = 0, 1, \ldots, p-1$. Count the number of equivalence classes if:

    (a) $p = 5$, $n = 2$ (b) $p = 6$, $n = 3$ (c) $p = 12$, $n = 2$

19. Repeat Exercise 17 with fSg holding if g can obtained from f by rotating the square by $0°$, $90°$, $180°$, or $270°$ or by reflecting about a line joining opposite corners of the square or by reflecting about a line joining midpoints of opposite sides of the square.

20. Consider the set $A = \{1, 2, \ldots, n\}$.

    (a) How many binary relations are possible on A?

    (b) How many reflexive relations are possible on A?

    (c) How many symmetric relations are possible on A?

    (d) How many transitive relations are possible on A?

(e) How many equivalence relations are possible on A when $n = 4$?

21. Suppose that V is the set of unlabeled graphs of n vertices and that aSb iff a and b are isomorphic.

    (a) Show that S is an equivalence relation on V.

    (b) Find one unlabeled graph from each equivalence class if $n = 3$.

22. Let E_n equal the number of equivalence relations on a set $A = \{1, 2, \ldots, n\}$. Show that E_n satisfies the following recurrence:

$$E_n = \sum_{i=0}^{n-1} \binom{n-1}{i} E_i, \quad n \geq 1.$$

8.2 PERMUTATION GROUPS

8.2.1 Definition of a Permutation Group

In studying examples such as Examples 8.1–8.5, we make heavy use of the notion of a permutation. Recall that a permutation of a set $A = \{1, 2, ..., n\}$ is an ordering of the elements of A. The permutation that sends 1 to a_1, 2 to a_2, and so on, can be written as

$$\begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ a_1 & a_2 & a_3 & \cdots & a_n \end{pmatrix},$$

or as $a_1 a_2 \cdots a_n$ for short. Thus, the permutation 132 stands for

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}.$$

Similarly, the permutation

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}$$

is written as 3142.

A permutation of A can also be thought of as a function from A onto itself. This function must be one-to-one. Thus, the permutation

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}$$

can be thought of as the function $\pi : \{1, 2, 3\} \to \{1, 2, 3\}$ defined by $\pi(1) = 1$, $\pi(2) = 3$, $\pi(3) = 2$. Similarly, if A is any finite set, any one-to-one function from A into A can be thought of as a permutation of A; we simply identify elements of A with the integers $1, 2, \ldots, n$. For instance, suppose that $A = \{a, b, c, d\}$ and $f(a) = b$, $f(b) = c$, $f(c) = d$, $f(d) = a$. If $a = 1, b = 2, c = 3, d = 4$, f can be thought of as the permutation

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}.$$

Suppose that π_1 and π_2 are permutations of the set A. We can define the *product* or *composition*, $\pi_1 \circ \pi_2$, of the permutations π_1 and π_2 as the permutation that first permutes by the permutation π_2 and then permutes the resulting arrangement by the permutation π_1. For instance, if

$$\pi_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix} \quad \text{and} \quad \pi_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix},$$

then

$$\pi_1 \circ \pi_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 3 & 1 \end{pmatrix}.$$

For 1 is sent to 2 by π_2, which is sent to 2 by π_1, so the composition sends 1 to 2. That is, $\pi_1 \circ \pi_2(1) = \pi_1(\pi_2(1)) = \pi_1(2) = 2$. Similarly, 2 is sent to 1 by π_2 and 1 to 4 by π_1, so the composition sends 2 to 4; and so on.

Let X be the collection of all permutations of the set A. Note that this collection of permutations satisfies the following conditions:

Condition **G1** (*Closure*). If $\pi_1 \in X$ and $\pi_2 \in X$, then $\pi_1 \circ \pi_2 \in X$.
Condition **G2** (*Associativity*). If $\pi_1, \pi_2, \pi_3 \in X$, then

$$\pi_1 \circ (\pi_2 \circ \pi_3) = (\pi_1 \circ \pi_2) \circ \pi_3.$$

Condition **G3** (*Identity*). There is an element $I \in X$, called the *identity*, so that for each $\pi \in X$,

$$I \circ \pi = \pi \circ I = \pi.$$

Condition **G4** (*Inverse*). For each $\pi \in X$, there is a $\pi^{-1} \in X$, called the *inverse* of π, so that

$$\pi \circ \pi^{-1} = \pi^{-1} \circ \pi = I.$$

To verify these conditions, note, for example, that **G3** follows by taking I to be the permutation

$$\begin{pmatrix} 1 & 2 & \cdots & n \\ 1 & 2 & \cdots & n \end{pmatrix}.$$

Also, **G4** holds if we take π^{-1} to be the permutation that reverses what π does. For example, if

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 3 & 1 \end{pmatrix}$$

then

$$\pi^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 3 & 2 \end{pmatrix}.$$

G1 has been tacitly assumed above. Its verification and that of **G2** are straightforward.

If X is any set and $\circ$ defines a product[5] on elements of X, the pair $G = (X, \circ)$ is called a *group* if the four properties G1, G2, G3, G4 hold. Let us give some examples of groups. If X is the positive real numbers and $a \circ b$ means $a \times b$, the pair $(X, \circ)$ is a group. Axiom **G1** holds because $a \times b$ is always a positive real number if a and b are positive reals. Axiom **G2** holds because $a \times (b \times c) = (a \times b) \times c$. Axiom **G3** holds because we take I to be 1. Axiom **G4** holds because we take a^{-1} to be $1/a$.

Another example of a group is $(X, \circ)$, where X is all the real numbers and $a \circ b$ is defined to be $a + b$. The identity element for Axiom **G3** is the number 0 and the inverse of element a is $-a$. Note that the real numbers where $a \circ b$ is defined to be $a \times b$ do not define a group. For the only possible identity is 1. But then the number 0 does not have an inverse: There is no number 0^{-1} so that $0 \times 0^{-1} = 1$.

We shall be interested in groups of permutations, or *permutation groups.* We have observed that the collection of *all* permutations of $A = \{1, 2, \ldots, n\}$ defines a group. This permutation group is called the *symmetric group.* Another example of a permutation group consists of the following three permutations of the set $\{1, 2, 3\}$:

$$\pi_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}, \quad \pi_2 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}, \quad \pi_3 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}. \tag{8.1}$$

It is left to the reader (Exercise 4) to verify that the group axioms are satisfied here.

Often, the symmetries of physical objects or configurations define groups, and hence the theory of groups is very important in modern physics. To give an example, consider the symmetries of the 2×2 array studied in Example 8.1: namely, the rotations by $0°, 90°, 180°$, and $270°$. These symmetries define a group if we take $a \circ b$ to mean first perform symmetry b and then perform symmetry a. For instance, if a is rotation by $90°$ and b is rotation by $180°$, then $a \circ b$ is rotation by $270°$.

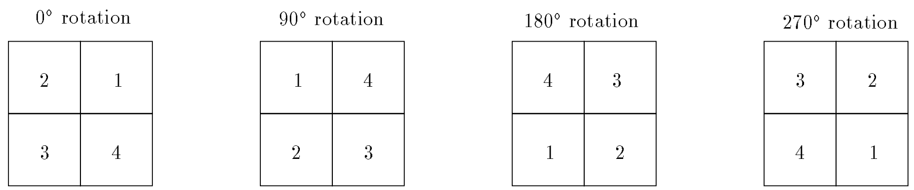
This group of symmetries can be thought of as a permutation group, each symmetry permuting $\{1, 2, 3, 4\}$. To see why, let us label the four cells in the 2×2 array as in the first part of Figure 8.13. Then Figure 8.13 shows the resulting labeling from the different symmetries. We can think of this labeling as corresponding to a permutation that takes the label i into the label j. For example, we can think of the $90°$ rotation as the permutation that takes 1 into 4, 2 into 1, 3 into 2, and 4 into 3, that is, the permutation

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{pmatrix}.$$

The permutations corresponding to the other rotations are also shown in Figure 8.13.

Suppose that A is any finite set and f is any one-to-one function from A into A. Then as we have observed before, f can be thought of as a permutation of A. If X is a collection of such functions and $\circ$ is the composition of functions and $G = (X, \circ)$

[5]Technically, a product $\circ$ is a function that assigns to each pair of elements a and b of X, another element (of X) denoted $a \circ b$. (Note that we can either define $a \circ b$ to always be an element of X, or make explicit condition **G1**.)



$$\pi_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix} \quad \pi_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{pmatrix} \quad \pi_3 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix} \quad \pi_4 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$$

Figure 8.13: The permutations corresponding to the rotations of the 2×2 array.

is a group, we can think of G as a permutation group. For instance, suppose that $A = \{a, b, c\}$ and f, g, and h are defined as follows:

$$\begin{array}{lll} f(a) = a, & f(b) = b, & f(c) = c; \\ g(a) = b, & g(b) = c, & g(c) = a; \\ h(a) = c, & h(b) = a, & h(c) = b. \end{array}$$

Then f, g, and h are one-to-one functions. It is easy to show that if $X = \{f, g, h\}$, then $(X, \circ)$ is a group. It is a permutation group. Indeed, if we take $a = 1$, $b = 2$, and $c = 3$, then f, g, and h are the permutations π_1, π_2, and π_3 of (8.1), so this is exactly the permutation group that we encountered earlier using different notation.

8.2.2 The Equivalence Relation Induced by a Permutation Group

Suppose that $G = (X, \circ)$ is a permutation group on a set A. We will sometimes use $\pi \in G$ to mean $\pi \in X$. We can define a sameness relation S on A by saying that

$$aSb \text{ iff there is a permutation } \pi \text{ in } G \text{ such that } \pi(a) = b; \tag{8.2}$$

that is, π takes a into b. For instance, if $A = \{1, 2, 3\}$ and G consists of the three permutations of (8.1), then $1S2$ because $\pi_2(1) = 2$ and $3S2$ because $\pi_3(3) = 2$. It is easy to see that for this S, aSb for all a, b.

Theorem 8.2 If G is a permutation group on a set A, then S as defined in (8.2) defines an equivalence relation on A.

Proof. We have to show that S satisfies reflexivity, symmetry, and transitivity. Since the identity permutation I is in G, $I(a) = a$ for all $a \in A$, so aSa holds for all a. Thus, reflexivity holds. If aSb, there is π in G so that $\pi(a) = b$. Now π^{-1} is in G and $\pi^{-1}(b) = a$. We conclude that bSa. Thus, symmetry holds. Finally, suppose that aSb and bSc. Then there are π_1 and π_2 in G so that $\pi_1(b) = c$ and $\pi_2(a) = b$. Then $\pi_1 \circ \pi_2(a) = c$, so aSc follows. Q.E.D.

The relation S will be called the *equivalence relation induced by the permutation group G*.

Let us give several more examples. If G is the group of permutations of $\{1,2,3,4\}$ shown in Figure 8.13, that is, the group of rotations of the 2×2 array, then aSb for all a, b in $\{1,2,3,4\}$. Thus, S has one equivalence class, $\{1,2,3,4\}$. Next, suppose that $A=\{1,2,3\}$ and G consists of the permutations

$$\pi_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} \quad \text{and} \quad \pi_2 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}.$$

Then G is a group (Exercise 3). Moreover, the equivalence classes under S are $\{1,3\}$ and $\{2\}$. Recall that $C(a)$, the equivalence class containing a, consists of all b in A such that aSb, equivalently all b in A such that $\pi(a)=b$ for some π in G. Thus,

$$C(a) = \{\pi(a) : \pi \in G\}.$$

In the special case of a permutation group, $C(a)$ is sometimes called the *orbit* of a. In the example we have just given,

$$C(1) = \{\pi_1(1), \pi_2(1)\} = \{1,3\}$$

is the orbit of 1.

In counting the number of distinct configurations, we shall be interested in counting the number of (distinct) equivalence classes under a sameness relation. One way to count is simply to compute all the equivalence classes and enumerate them. But this is often impractical. In the next section we present a method for counting the number of equivalence classes without enumeration.

8.2.3 Automorphisms of Graphs

Let H be a fixed, unlabeled graph.[6] An *automorphism* of H is a permutation π of the vertices of H so that if $\{x,y\}\in E(H)$, then $\{\pi(x),\pi(y)\}\in E(H)$. To use the terminology of Section 3.1.3, an automorphism is an isomorphism from a graph into itself. Consider, for example, the graph of Figure 8.14. We can define an automorphism by labeling the vertices as $1,2,3,4,5$ as shown and taking $\pi(1)=4, \pi(2)=5, \pi(3)=1, \pi(4)=2, \pi(5)=3$. This is the same as a rotation by 144°. A second automorphism is obtained by taking $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 5 & 4 & 3 & 2 \end{pmatrix}$. This is the same as a reflection through the line joining vertex 1 to the midpoint of edge $\{3,4\}$.

As a second example, consider the graph known as $K_{1,3}$ and shown in Figure 8.15. A vertex labeling using the integers 1, 2, 3, 4 is shown. One example of an automorphism of $K_{1,3}$ is obtained by rotating the labeled figure by 120° clockwise. This corresponds to the permutation $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix}$. Reflection through the edge $\{1,2\}$ produces the automorphism $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 4 & 3 \end{pmatrix}$. There are six automorphisms

[6] In this chapter, we reserve G for a group and use H for a graph.

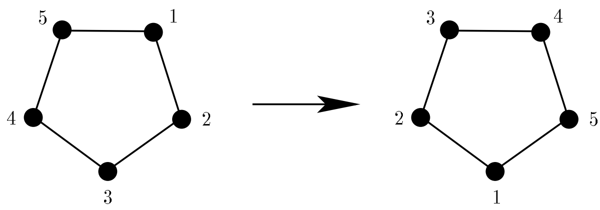


Figure 8.14: An automorphism is given by $\pi(1) = 4, \pi(2) = 5, \pi(3) = 1, \pi(4) = 2, \pi(5) = 3$.

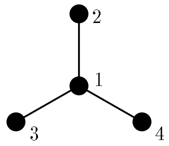


Figure 8.15: The graph $K_{1,3}$.

in all: the identity, the ones obtained by rotations of 120° and 240°, and the ones obtained by reflections through the three edges. How do we know that there are no others? Clearly, every automorphism must take 1 into 1. Thus, we need to look for permutations of $\{1, 2, 3, 4\}$ that take 1 into 1, and there are $3! = 6$ of them. The sameness relation of (8.2) gives two equivalence classes, $\{1\}$ and $\{2, 3, 4\}$.

Theorem 8.3 The set of all automorphisms of a graph is a permutation group.

Proof. Left to the reader (Exercise 26). Q.E.D.

We shall use the notation $\mathrm{Aut}(H)$ for the automorphism group of H. Thus, for instance, $\mathrm{Aut}(K_n)$ is the symmetric group on $\{1, 2, \ldots, n\}$, since all permutations of the vertices of K_n define automorphisms. For more on automorphism groups of graphs, see, for example, Cameron [1983] and Gross and Yellen [1999].

EXERCISES FOR SECTION 8.2

1. Write each of the following permutations in the form

$$\begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ a_1 & a_2 & a_3 & \cdots & a_n \end{pmatrix}.$$

(a) 7263541 (b) 52413 (c) 67583412

2. Find $\pi_1 \circ \pi_2$ if π_1 and π_2 are as follows:

(a) $\pi_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}$, $\pi_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 1 & 4 \end{pmatrix}$

(b) $\pi_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}$, $\pi_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{pmatrix}$

(c) $\pi_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 5 & 4 \end{pmatrix}$, $\pi_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 5 & 1 & 3 \end{pmatrix}$

(d) $\pi_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 1 & 4 & 5 & 6 & 2 \end{pmatrix}$, $\pi_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 1 & 3 & 6 & 5 \end{pmatrix}$

3. Suppose that $A = \{1, 2, 3\}$ and X is the set of permutations

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}.$$

If $\circ$ is composition, show that $(X, \circ)$ is a group.

4. Suppose that $A = \{1, 2, 3\}$ and X is the set of the three permutations given in Equation (8.1). Show that X defines a group under composition.

5. For each of the following X and $\circ$, check which of the four axioms for a group hold.

(a) X = the permutations $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$ and $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 3 & 2 & 1 \end{pmatrix}$ and $\circ$ = composition

(b) X = the permutations $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$ and $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 2 & 4 & 5 \end{pmatrix}$ and $\circ$ = composition

(c) $X = \{0, 1\}$ and $\circ$ is defined by the following rules: $0 \circ 0 = 0, 0 \circ 1 = 1, 1 \circ 0 = 1, 1 \circ 1 = 1$

(d) X = rational numbers, $\circ$ = addition

(e) X = rational numbers, $\circ$ = multiplication

(f) X = negative real numbers, $\circ$ = addition

(g) X = all 2×2 matrices of real numbers, $\circ$ = matrix multiplication

6. Show that the set of functions $\{f, g\}$ is a group of permutations on $A = \{x, y, u, v\}$ if $f(x) = v, f(y) = u, f(u) = y, f(v) = x$, and $g(x) = x, g(y) = y, g(u) = u, g(v) = v$.

7. Is the conclusion of Exercise 6 still true if f is redefined by $f(x) = y, f(y) = x, f(u) = v, f(v) = u$?

8. Suppose that $A = \{1, 2, 3, 4, 5, 6\}$ and G is the following group of permutations:

$$\left\{ \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 \end{pmatrix} \right\}.$$

If S is the equivalence relation induced by G:

(a) Is $1S2$? (b) Is $3S5$? (c) Is $5S6$?

9. In Exercise 8, find the orbits $C(1)$ and $C(4)$.

10. If A and G are as follows, find the equivalence classes under the equivalence relation S induced by G.

(a)
$$\begin{aligned} A &= \{1, 2, 3, 4, 5\}, \\ G &= \left\{ \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 3 & 2 & 1 \end{pmatrix} \right\} \end{aligned}$$

(b) $A = \{1,2,3,4,5,6\}$,
$G = \left\{\begin{pmatrix} 1&2&3&4&5&6\\ 1&2&3&4&5&6\end{pmatrix}, \begin{pmatrix} 1&2&3&4&5&6\\ 2&1&3&4&5&6\end{pmatrix}, \begin{pmatrix} 1&2&3&4&5&6\\ 1&2&4&3&5&6\end{pmatrix},\right.$
$\begin{pmatrix} 1&2&3&4&5&6\\ 1&2&3&4&6&5\end{pmatrix}, \begin{pmatrix} 1&2&3&4&5&6\\ 2&1&4&3&5&6\end{pmatrix}, \begin{pmatrix} 1&2&3&4&5&6\\ 2&1&3&4&6&5\end{pmatrix},$
$\left.\begin{pmatrix} 1&2&3&4&5&6\\ 1&2&4&3&6&5\end{pmatrix}, \begin{pmatrix} 1&2&3&4&5&6\\ 2&1&4&3&6&5\end{pmatrix}\right\}$

(c) $A = \{1,2,3,4,5\}$,
$G = \left\{\begin{pmatrix} 1&2&3&4&5\\ 1&2&3&4&5\end{pmatrix}, \begin{pmatrix} 1&2&3&4&5\\ 1&3&2&4&5\end{pmatrix}, \begin{pmatrix} 1&2&3&4&5\\ 5&2&3&4&1\end{pmatrix},\right.$
$\left.\begin{pmatrix} 1&2&3&4&5\\ 5&3&2&4&1\end{pmatrix}\right\}$

11. Consider the collection of all symmetries of the 2×2 array described in Exercise 9, Section 8.1. If π_1 and π_2 are the following symmetries, find $\pi_1 \circ \pi_2$.

 (a) π_1 = rotation by $90°$, π_2 = reflection in a horizontal line
 (b) π_1 = reflection in a vertical line, π_2 = rotation by $180°$
 (c) π_1 = rotation by $270°$, π_2 = reflection in a vertical line
 (d) π_1 = rotation by $180°$, π_2 = reflection in the diagonal going from lower left to upper right

12. Continuing with Exercise 11, describe the following symmetries as permutations of $\{1,2,3,4\}$:

 (a) Reflection in a horizontal line
 (b) Reflection in a vertical line
 (c) Reflection in a diagonal going from lower left to upper right
 (d) Reflection in a diagonal going from upper left to lower right

13. Continuing with Exercise 11, is the collection of all the symmetries (rotations and reflections) a group?
14. In Example 8.1, suppose that we can use any of c colors in any of the squares.

 (a) How many distinct colorings are possible with only rotations allowed?
 (b) How many distinct colorings are possible with rotations and reflections, in either a vertical, horizontal, or diagonal line, allowed?

15. Find the permutation corresponding to each automorphism of graph $K_{1,3}$ that is not described in the text.
16. In the situation of Exercise 17, Section 8.1, find $\pi_1 \circ \pi_2$ if π_1 and π_2 are rotations by $180°$ and $270°$, respectively.
17. In the situation of Exercise 19, Section 8.1, find $\pi_1 \circ \pi_2$ if π_1 and π_2 are:

 (a) Rotation by $180°$ and reflection about a horizontal line joining midpoints of opposite sides, respectively
 (b) Reflection about a vertical line joining midpoints of opposite sides and rotation by $90°$, respectively
 (c) Reflection about a positively sloped diagonal line (/) joining opposite corners and reflection about a horizontal line joining midpoints of opposite sides, respectively

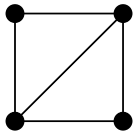


Figure 8.16: $K_4 - K_2$.

18. In the situation of Exercise 19, Section 8.1, is the collection of all symmetries described a group?
19. If π_1 and π_2 are permutations, $\pi_1 \circ \pi_2$ may not equal $\pi_2 \circ \pi_1$. (Thus, we say that the product of permutations is not necessarily commutative.)

    (a) Demonstrate this with $\pi_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 2 & 3 \end{pmatrix}$ and $\pi_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix}$.

    (b) Find two symmetries π_1 and π_2 of the 2×2 array (π_1 and π_2 can be rotations or reflections) such that $\pi_1 \circ \pi_2 \neq \pi_2 \circ \pi_1$.

20. Show that for all prime numbers p, the set of integers $\{1, 2, \ldots, p-1\}$ with $\circ$ equal to multiplication modulo p forms a group.
21. In Exercise 20, do we still get a group if p is not a prime? Why?
22. Suppose that G is a permutation group. Fix a permutation σ in G. If π_1 and π_2 are in G, we say that $\pi_1 S \pi_2$ if $\pi_1 = \sigma^{-1} \circ (\pi_2 \circ \sigma)$. Show that S is an equivalence relation.
23. (a) Find $\text{Aut}(L_4)$, where L_4 is the chain of four vertices.

    (b) Find $\text{Aut}(Z_4)$, where Z_4 is the circuit of four vertices.

    (c) Find $\text{Aut}(K_4 - K_2)$, where $K_4 - K_2$ is the graph shown in Figure 8.16.
24. Describe $\text{Aut}(Z_n)$ and find the number of automorphisms of Z_n.
25. The graph $K_{m,n}$ has m vertices in one class, n vertices in a second class, and edges between all pairs of vertices in different classes. $K_{1,3}$ of Figure 8.15 is a special case.

    (a) If $m \neq n$, describe $\text{Aut}(K_{m,n})$ and find the number of automorphisms of $K_{m,n}$.

    (b) Repeat part (a) for $m = n$.

26. Prove Theorem 8.3.

8.3 BURNSIDE'S LEMMA

8.3.1 Statement of Burnside's Lemma

In this section we present a method for counting the number of (distinct) equivalence classes under the equivalence relation induced by a permutation group. Suppose that G is a group of permutations of a set A. An element a in A is said to be *invariant* (or *fixed*) under a permutation π of G if $\pi(a) = a$. Let $\text{Inv}(\pi)$ be the number of elements of A that are invariant under π.

Theorem 8.4 (Burnside's Lemma[7]) Let G be a group of permutations of a set A and let S be the equivalence relation on A induced by G. Then the number of equivalence classes in S is given by

$$\frac{1}{|G|}\sum_{\pi\in G}\text{Inv}(\pi).$$

To illustrate this theorem, let us first consider the set $A = \{1,2,3\}$ and the group G of permutations of A defined by Equation (8.1). Then $\text{Inv}(\pi_1) = 3$ since 1, 2, and 3 are invariant under π_1, and $\text{Inv}(\pi_2) = \text{Inv}\ (\pi_3) = 0$, since no element is invariant under either π_2 or π_3. Hence, the number of equivalence classes under the induced equivalence relation S is given by $\frac{1}{3}(3+0+0) = 1$. This is correct, since aSb holds for all a, $b \in A$. There is just one equivalence class, $\{1,2,3\}$.

To give a second example, suppose that $A = \{1,2,3,4\}$ and G consists of the following permutations:

$$\begin{aligned}
\pi_1 &= \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}, \qquad & \pi_2 &= \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix}, \\
\pi_3 &= \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 4 & 3 \end{pmatrix}, \qquad & \pi_4 &= \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}.
\end{aligned} \tag{8.3}$$

It is easy to check that G is a group. Now $\text{Inv}(\pi_1) = 4$, $\text{Inv}\ (\pi_2) = 2$, $\text{Inv}(\pi_3) = 2$, $\text{Inv}(\pi_4) = 0$, and the number of equivalence classes under the induced equivalence relation S is $\frac{1}{4}(4+2+2+0) = 2$. This is correct since the two equivalence classes are $\{1, 2\}$ and $\{3, 4\}$ (Exercise 1).

As a third example, consider the set A of all weak orders on $\{1,2,3\}$, as shown in Figure 8.9. Every permutation π of $\{1,2,3\}$ induces a permutation π^* of A by replacing each element i in a weak order by $\pi(i)$. For instance, if $\pi = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$, then $\pi^*(J) = K$ for weak orders J and K shown in Figure 8.9. The set of all π^* defines a permutation group G of A. There are six permutations in G, one corresponding to each of $\pi_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$, $\pi_2 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}$, $\pi_3 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$, $\pi_4 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$, $\pi_5 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$, $\pi_6 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$. If we use the sameness relation defined in Example 8.6, then $\text{Inv}(\pi_1^*) = 13$ since π_1^* leaves invariant all 13 weak orders of Figure 8.9. $\text{Inv}(\pi_2^*) = 3$ since π_2^* leaves invariant weak orders I, J, and M. Similarly, $\text{Inv}(\pi_3^*) = \text{Inv}(\pi_6^*) = 3$. Finally, $\text{Inv}(\pi_4^*) = \text{Inv}(\pi_5^*) = 1$ since π_4^* and π_5^* leave only M invariant. By Burnside's Lemma, the number of equivalence classes of weak orders is given by

$$\frac{1}{6}\,[13+3+3+1+1+3] = 4.$$

[7] This version of the lemma is a simple consequence of the crucial lemma given by Burnside [1911] and is usually called Burnside's Lemma.

The equivalence classes are given by $\{A, B, C, D, E, F\}$, $\{G, H, I\}$, $\{J, K, L\}$, and $\{M\}$.

As a fourth example, consider the automorphism group $G = \text{Aut}(K_{1,3})$. From our discussion in Section 8.2.3, $|G| = 6$. $\text{Inv}(\pi) = 4$ for the identity π, $\text{Inv}(\pi) = 1$ for the rotations by 120° and 240°, and $\text{Inv}(\pi) = 2$ for the three reflections through an edge. Thus,

$$\frac{1}{|G|}\sum_{\pi \in G} \text{Inv}(\pi) = \frac{1}{6}[4+1+1+2+2+2] = 2,$$

which agrees with our conclusion that $\{1\}$ and $\{2, 3, 4\}$ are the equivalence classes under S.

In Section 8.4 we shall see how to apply Burnside's Lemma to examples such as Examples 8.1–8.5.

8.3.2 Proof of Burnside's Lemma[8]

We now present a proof of Burnside's Lemma. Suppose that G is a group of permutations on a set A. For each $a \in A$, let $\text{St}(a)$, the *stabilizer* of a, be the set of all permutations in G under which a is invariant, i.e., $\text{St}(a) = \{\pi \in G : \pi(a) = a\}$. Let $C(a)$ be the orbit of a, the equivalence class containing a under the induced equivalence relation S, that is, the set of all b such that $\pi(a) = b$ for some π in G. To illustrate, suppose that $A = \{1, 2, 3\}$ and G is defined by Equation (8.1). Then $C(2) = \{\pi_1(2), \pi_2(2), \pi_3(2)\} = \{1, 2, 3\}$. Also, $\text{St}(2) = \{\pi_1\}$.

Lemma 8.1 Suppose that G is a group of permutations on a set A and a is in A. Then

$$|\text{St}(a)| \cdot |C(a)| = |G|.$$

Proof. Suppose that $C(a) = \{b_1, b_2, \ldots, b_r\}$. Then there is a permutation π_1 that sends a to b_1. (There may be other permutations that send a to b_1, but we pick one such.) There is also a permutation π_2 that sends a to b_2, a permutation π_3 that sends a to b_3, and so on. Let $P = \{\pi_1, \pi_2, \ldots, \pi_r\}$. Note that $|P| = |C(a)|$. We shall show that every permutation π in G can be written in exactly one way as the product of a permutation in P and a permutation in $\text{St}(a)$. It then follows by the product rule that $|G| = |P| \cdot |\text{St}(a)| = |C(a)| \cdot |\text{St}(a)|$.

Given π in G, note that $\pi(a) = b_k$, some k. Thus, $\pi(a) = \pi_k(a)$, so $\pi_k^{-1} \circ \pi$ leaves a invariant. Thus, $\pi_k^{-1} \circ \pi$ is in $\text{St}(a)$. But

$$\pi_k \circ (\pi_k^{-1} \circ \pi) = (\pi_k \circ \pi_k^{-1}) \circ \pi = I \circ \pi = \pi,$$

so π is the product of a permutation in P and a permutation in $\text{St}(a)$.

Next, suppose that π can be written in two ways as a product of a permutation in P and a permutation in $\text{St}(a)$; that is, suppose that $\pi = \pi_k \circ \gamma = \pi_l \circ \delta$, where γ, δ are in $\text{St}(a)$. Now $(\pi_k \circ \gamma)(a) = b_k$ and $(\pi_l \circ \delta)(a) = b_l$. Since $\pi_k \circ \gamma = \pi_l \circ \delta$, b_k must equal b_l, so $k = l$. Thus, $\pi_k \circ \gamma = \pi_k \circ \delta$, and by multiplying by π_k^{-1}, we conclude that $\gamma = \delta$. Q.E.D.

[8] This subsection may be omitted.

To illustrate this lemma, let $A = \{1, 2, 3\}$ and let G be defined by (8.1). By our computation above, $C(2) = \{1, 2, 3\}$ and $\mathrm{St}(2) = \{\pi_1\}$ Thus,

$$|G| = 3 = (1) \cdot (3) = |\mathrm{St}(2)| \cdot |C(2)|.$$

To complete the proof of Burnside's Lemma, we show that if $A = \{1, 2, \ldots, n\}$ and $G = \{\pi_1, \pi_2, \ldots \pi_m\}$, then

$$\mathrm{Inv}(\pi_1) + \mathrm{Inv}(\pi_2) + \cdots + \mathrm{Inv}(\pi_m) = |\mathrm{St}(1)| + |\mathrm{St}(2)| + \cdots + |\mathrm{St}(n)|.$$

This is true because both sides of this equation count the number of ordered pairs (a, π) such that $\pi(a) = a$. It then follows by Lemma 8.1 that

$$\frac{1}{|G|}\left[\mathrm{Inv}(\pi_1) + \mathrm{Inv}(\pi_2) + \cdots + \mathrm{Inv}(\pi_m)\right] = \frac{1}{|C(1)|} + \frac{1}{|C(2)|} + \cdots + \frac{1}{|C(n)|}. \quad (8.4)$$

Note that x is always in $C(x)$, since $I(x) = x$. Thus, by Theorem 8.1, $C(x) = C(y)$ iff x is in $C(y)$. Hence, if $C(x) = \{b_1, b_2, \ldots, b_k\}$, there are exactly k equivalence classes $C(b_1), C(b_2), \ldots, C(b_k)$ that equal $C(x)$. It follows that we may split the equivalence classes up into groups such as $\{C(b_1), C(b_2), \ldots, C(b_k)\}$, each group being a list of identical equivalence classes. Note that $|C(b_i)| = k$. Thus,

$$\frac{1}{|C(b_1)|} + \frac{1}{|C(b_2)|} + \cdots + \frac{1}{|C(b_k)|} = \frac{1}{k} + \frac{1}{k} + \cdots + \frac{1}{k} = 1.$$

It follows that the sum on the right-hand side of (8.4) will count the number of distinct equivalence classes, so this number is also given by the left-hand side of (8.4). Burnside's Lemma follows.

To illustrate the proof, suppose that $A = \{1, 2, 3, 4\}$ and G is given by the four permutations of (8.3). Then $C(1) = \{1, 2\}, C(2) = \{1, 2\}, C(3) = \{3, 4\}$, and $C(4) = \{3, 4\}$. Thus,

$$\frac{1}{|C(1)|} + \frac{1}{|C(2)|} = 1,$$
$$\frac{1}{|C(3)|} + \frac{1}{|C(4)|} = 1,$$

and

$$\frac{1}{|C(1)|} + \frac{1}{|C(2)|} + \frac{1}{|C(3)|} + \frac{1}{|C(4)|} = 2,$$

the number of equivalence classes.

EXERCISES FOR SECTION 8.3

1. Verify that the four permutations of (8.3) define a group and that the equivalence classes under the induced equivalence relation are $\{1, 2\}$ and $\{3, 4\}$.
2. For each automorphism a in $\mathrm{Aut}(K_{1,3})$, calculate $\mathrm{St}(a)$, $C(a)$, and verify Lemma 8.1.

3. Verify the proof of Burnside's Lemma for $\text{Aut}(K_{1,3})$ by calculating
$$\frac{1}{|C(1)|}+\frac{1}{|C(2)|}+\frac{1}{|C(3)|}+\frac{1}{|C(4)|}.$$
4. In Exercise 8, Section 8.2, use Burnside's Lemma to find the number of equivalence classes under S.
5. In each case of Exercise 10, Section 8.2, use Burnside's Lemma to find the number of equivalence classes under S and check by computing the equivalence classes.
6. For each case of Exercise 10, Section 8.2, let $a = 1$.

   (a) Find $\text{St}(a)$. (b) Find $C(a)$. (c) Verify Lemma 8.1.
7. Repeat Exercise 6 with $a = 3$.
8. For each case of Exercise 10, Section 8.2, check that
$$\frac{1}{|C(1)|}+\frac{1}{|C(2)|}+\cdots$$
gives the number of equivalence classes under S.
9. For every automorphism a of the graph of Figure 8.14, calculate $\text{St}(a)$ and $C(a)$ and verify Lemma 8.1.
10. For the graph H of Figure 8.14, verify the proof of Burnside's Lemma for $\text{Aut}(H)$ by calculating
$$\frac{1}{|C(1)|}+\frac{1}{|C(2)|}+\frac{1}{|C(3)|}+\frac{1}{|C(4)|}+\frac{1}{|C(5)|}$$
and comparing to the number of equivalence classes under S.
11. Use Burnside's Lemma to calculate the number of equivalence classes of weak orders on $\{1, 2, 3, 4\}$ if sameness is defined as in Example 8.6.
12. Use Burnside's Lemma to compute the number of distinct ways to seat 5 negotiators in fixed chairs around a circular table if rotating seat assignments around the circle is not considered to change the seating arrangement.
13. Suppose that
$$A = \{1, 2, 3, 4\},\quad G = \left\{\begin{pmatrix} 1 & 2 & 3 & 4\\ 1 & 2 & 3 & 4\end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4\\ 2 & 1 & 3 & 4\end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4\\ 1 & 2 & 4 & 3\end{pmatrix}\right\}.$$
Is $|\text{St}(1)| \cdot |C(1)| = |G|$? Explain what happened.
14. Suppose that we label the n vertices of a graph H with the labels $1, 2, \ldots, n$. Any labeling of H can be thought of as a permutation of $\{1, 2, \ldots, n\}$, if we start with a fixed labeling. For instance, if we start with the original labeling shown in Figure 8.14, the new labeling shown in Figure 8.17 corresponds to the permutation
$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5\\ 2 & 1 & 5 & 3 & 4\end{pmatrix}.$$
Every automorphism π of H can be thought of as taking any labeling σ of H into another labeling: We simply use the labeling $\pi \circ \sigma$. Thus, the number of distinct labelings of H corresponds to the number of equivalence classes in the equivalence relation induced on the set of permutations of $\{1, 2, \ldots, n\}$ by the automorphism group $\text{Aut}(H)$.

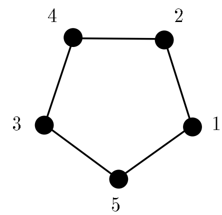


Figure 8.17: A new labeling of the graph of Figure 8.14.

(a) Show that the number of distinct labelings is given by $n!/|\mathrm{Aut}(H)|$.

(b) If L_4 is the chain of four vertices, find the number of distinct labelings from the result in part (a) and check by enumerating the labelings.

(c) Repeat for Z_4, the circuit of length 4.

(d) Repeat for $K_{1,3}$, the graph of Figure 8.15.

15. Using the methods of this section, do Exercise 25 from the Additional Exercises for Chapter 2.

8.4 DISTINCT COLORINGS

8.4.1 Definition of a Coloring

Suppose that D is a collection of objects. A *coloring of* D assigns a color to each object in D. In this sense, if D is the vertex set of a graph, a coloring simply assigns a color to each vertex, independent of the rule used in Chapter 3 that if x and y are joined by an edge, they must get different colors. A coloring can be thought of as a function $f : D \to R$, where R is the set of colors. If D has n elements and R has m elements, there are m^n colorings of D.

In Example 8.1, the set D is the set of four boxes in the 2×2 array, and the set R is the set $\{\text{black, white}\}$. In Example 8.2, the set D can be thought of as the integers $1, 2, \ldots, k$, representing the k spaces for beads, and R is the set $\{b, r\}$. In Example 8.3, D is the set of bit strings of length n, and R is the set $\{0, 1\}$. In Example 8.4, D is the set of vertices of the tree of seven vertices, and $R = \{\text{black, white}\}$. Finally, in Example 8.5, the set D consists of the vertices a, b, c, d of the regular tetrahedron, and the set R is the set $\{\mathrm{CH_3}, \mathrm{C_2H_5}, \mathrm{H}, \mathrm{Cl}\}$.

Every graph G on the vertex set $V = \{1, 2, \ldots, p\}$ can be thought of as a coloring. Take D to be the set of all 2-element subsets of V, R to be $\{0, 1\}$, and let $f(\{i, j\})$ be 1 if $\{i, j\} \in E(G)$ and 0 otherwise. Exercises 21 and 22 exploit this idea to compute the number of distinct (nonisomorphic) graphs of p vertices.

In all of our examples, we also allow certain permutations of the elements of D. These permutations define a group G. In particular, in Example 8.1 we allow the four rotations given in Figure 8.13, which define a group G of permutations. In

Example 8.2, in the case of two beads, the permutations are

$$\pi_1 = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix} \quad \text{and} \quad \pi_2 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}.$$

Note that π_1 and π_2 define a group—this is the group G. More generally, if there are k beads, the group G is the group of the two permutations

$$\begin{pmatrix} 1 & 2 & \cdots & k \\ 1 & 2 & \cdots & k \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 2 & \cdots & k \\ k & k-1 & \cdots & 1 \end{pmatrix}.$$

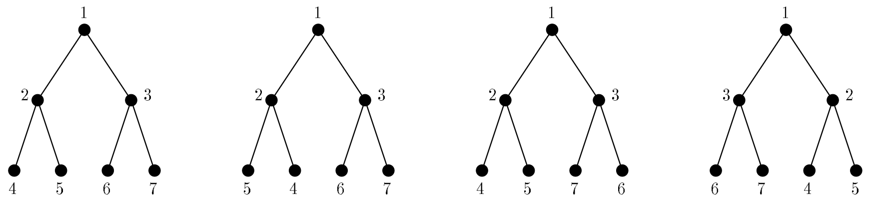
In Example 8.5, the permutations in the group G correspond to the symmetries of the regular tetrahedron that were described in Section 8.1. We return to this example in Section 8.5, where we describe a simple way to represent these permutations.

What is the group in Example 8.4? Suppose that we start with the first labeled tree shown in Figure 8.18, that with π_1 under it. Then any other labeling of this tree corresponds to a permutation of $\{1, 2, \ldots, 7\}$. Not every permutation of $\{1, 2, \ldots, 7\}$ corresponds to a labeling which is considered equivalent in the sense that left and right have been interchanged. Figure 8.18 shows all labeled trees obtained from the first one by interchanging left and right. For instance, the labeled tree with π_2 under it is obtained by interchanging vertices 4 and 5, and the labeled tree with π_3 under it is obtained by interchanging vertices 6 and 7. The labeled tree with π_4 under it is obtained by interchanging the subtree T_1 generated by vertices $2, 4, 5$ with the subtree T_2 generated by vertices $3, 6, 7$. The labeled tree with π_5 under it is obtained by interchanging both 4 and 5 and 6 and 7. The labeled tree with π_6 under it is obtained by first interchanging subtrees T_1 and T_2 and then interchanging vertices 6 and 7; and so on. These eight trees correspond to the legitimate permutations of the elements of D, the members of the group G. The permutation corresponding to each labeled tree is also shown in Figure 8.18. Note that

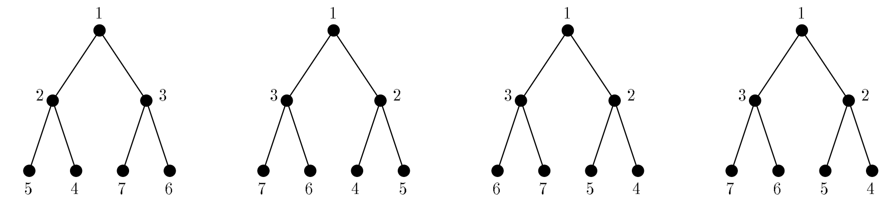
$$\pi_4 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 3 & 2 & 6 & 7 & 4 & 5 \end{pmatrix},$$

because vertex 1 stays unchanged, vertex 2 is changed to vertex 3 and vertex 3 to vertex 2, and so on. Verification that G is a group is left to the reader (Exercise 19). It is not hard to show that the permutations we have described are exactly the automorphisms of the first tree of Figure 8.18.

We shall discuss Example 8.3, the switching functions, shortly. In Examples 8.1–8.5, we are interested in determining whether or not two colorings are distinct and in counting the number of equivalence classes of colorings. However, the equivalence relation of two colorings being the same is not the same as the equivalence relation S induced by the permutation group G. For S is a relation on the set D, not on the set of colorings of D. Thus, direct use of Burnside's Lemma would not help us to count the number of equivalence classes of colorings. In Section 8.4.2 we discuss how to define the appropriate equivalence relation.



$\pi_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{pmatrix}$ $\pi_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 5 & 4 & 6 & 7 \end{pmatrix}$ $\pi_3 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 7 & 6 \end{pmatrix}$ $\pi_4 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 3 & 2 & 6 & 7 & 4 & 5 \end{pmatrix}$



$\pi_5 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 5 & 4 & 7 & 6 \end{pmatrix}$ $\pi_6 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 3 & 2 & 7 & 6 & 4 & 5 \end{pmatrix}$ $\pi_7 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 3 & 2 & 6 & 7 & 5 & 4 \end{pmatrix}$ $\pi_8 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 3 & 2 & 7 & 6 & 5 & 4 \end{pmatrix}$

Figure 8.18: A labeled tree and the eight labeled subtrees obtained from it by interchanging left and right.

8.4.2 Equivalent Colorings

Suppose that $C(D,R)$ is the set of all colorings of D using colors in R and that G is a group of permutations of the set D and π is in G. Corresponding to π is a permutation π^* of $C(D,R)$. π^* takes each coloring in $C(D,R)$ into another coloring. If f is a coloring, the new coloring $\pi^* f$ is defined by taking $(\pi^* f)(a)$ to be $f(\pi(a))$. That is, $\pi^* f$ assigns to a the same color that f assigns to $\pi(a)$. In Example 8.1, π^* takes a given coloring C_i of the 2×2 array into another one. For instance, if π_2 is the $90°$ rotation of the array, as shown in Figure 8.13, let us compute π_2^*. If $C_1, C_2, \ldots$ are as in Figure 8.10, first note that $C_1(x) =$ black, all x, so $(\pi_2^* C_1)(a) = C_1(\pi_2(a)) =$ black. That means that $(\pi_2^* C_1)(a) =$ black, all a, so $\pi_2^* C_1$ is the coloring C_1. Next, since $4 = \pi_2(1), (\pi_2^* C_2)(1) = C_2(\pi_2(1)) = C_2(4) =$ black. Also, $(\pi_2^* C_2)(2) = C_2(\pi_2(2)) = C_2(1) =$ black, $(\pi_2^* C_2)(3) = C_2(\pi_2(3)) = C_2(2) =$ white, and $(\pi_2^* C_2)(4) = C_2(\pi_2(4)) = C_2(3) =$ black. Thus, $\pi_2^* C_2$ is the same as the coloring C_3. Similarly, $\pi_2^* C_3 = C_4, \pi_2^* C_4 = C_5$, and so on. In sum, the permutation π_2^* is given by

$$\pi_2^* = \begin{pmatrix} C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 & C_9 & C_{10} & C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_1 & C_3 & C_4 & C_5 & C_2 & C_7 & C_8 & C_9 & C_6 & C_{11} & C_{10} & C_{13} & C_{14} & C_{15} & C_{13} & C_{16} \end{pmatrix}. \quad (8.5)$$

Similarly, if π_1 is the $0°$ rotation, π_3 is the $180°$ rotation, and π_4 is the $270°$ rotation, then

$$\pi_1^* = \begin{pmatrix} C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 & C_9 & C_{10} & C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 & C_9 & C_{10} & C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \end{pmatrix}, \quad (8.6)$$

$$\pi_3^* = \begin{pmatrix} C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 & C_9 & C_{10} & C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_1 & C_4 & C_5 & C_2 & C_3 & C_8 & C_9 & C_6 & C_7 & C_{10} & C_{11} & C_{14} & C_{15} & C_{12} & C_{13} & C_{16} \end{pmatrix}, \quad (8.7)$$

and

$$\pi_4^* = \begin{pmatrix} C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 & C_9 & C_{10} & C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_1 & C_5 & C_2 & C_3 & C_4 & C_9 & C_6 & C_7 & C_8 & C_{11} & C_{10} & C_{15} & C_{12} & C_{13} & C_{14} & C_{16} \end{pmatrix}. \quad (8.8)$$

Thus, the group G of permutations of D corresponds to a group G^* of permutations of $C(D,R)$; $G^* = \{\pi^* : \pi \in G\}$. (Why is G^* a group?) Note that G and G^* have the same number of elements. Moreover, if S^* is the equivalence relation induced by G^*, then S^* is the sameness relation in which we are interested. Under S^*, two colorings f and g are considered equivalent if for some permutation π of D, $g = \pi^* f$, that is, for all a in D, $g(a) = f(\pi(a))$.

In Example 8.2, suppose that

$$\pi = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}.$$

Let $f(1) = r, f(2) = b, g(1) = b, g(2) = r$. Then $g(a) = f(\pi(a))$ for all a, so fS^*g. This just says that the two colorings rb and br are equivalent. Henceforth, to distinguish equivalence classes under S from those under S^*, we shall refer to the latter equivalence classes as *patterns*. We are interested in computing the number of distinct patterns. This can be done by applying Burnside's Lemma to G^*.

In Example 8.1, $\text{Inv}(\pi_1^*) = 16$, $\text{Inv}(\pi_2^*) = 2$, $\text{Inv}(\pi_3^*) = 4$, and $\text{Inv}(\pi_4^*) = 2$, since π_1^* leaves all 16 colorings C_i invariant, π_2^* and π_4^* leave only C_1 and C_{16} invariant, and π_3^* leaves C_1, C_{10}, C_{11}, and C_{16} invariant. Thus, the number of equivalence classes under S^* is given by $\frac{1}{4}(16+2+4+2) = 6$, which agrees with Figure 8.10. [We could have computed $\text{Inv}(\pi_i^*)$ directly without first computing π_i^*. For instance, π_3^* leaves invariant only those colorings that agree in boxes 1 and 3 and agree in boxes 2 and 4. Since in such a coloring there are 2 choices for the color for boxes 1 and 3 and 2 choices for the color for boxes 2 and 4, there are $2^2 = 4$ choices for the coloring. Similarly, π_2^* leaves invariant only those colorings that agree in all four boxes, since each box must get the same color as the one $90°$ away in a clockwise direction. Thus, there are only 2 such colorings.]

In the case of the necklaces (Example 8.2), if

$$\pi_1 = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix} \quad \text{and} \quad \pi_2 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix},$$

then

$$\pi_1^* = \begin{pmatrix} bb & br & rb & rr \\ bb & br & rb & rr \end{pmatrix} \quad \text{and} \quad \pi_2^* = \begin{pmatrix} bb & br & rb & rr \\ bb & rb & br & rr \end{pmatrix}. \quad (8.9)$$

Note that $\text{Inv}(\pi_1^*) = 4$ and $\text{Inv}(\pi_2^*) = 2$, so that the number of patterns (equivalence classes) under S^* is given by $\frac{1}{2}(4+2) = 3$. This agrees with our earlier observation that the patterns are $\{bb\}, \{br, rb\}$, and $\{rr\}$.

In the tree colorings (Example 8.4), note that there are 2^7 tree colorings in all: Each vertex of the tree can get one of the two colors. Suppose that π_i^* is the permutation of tree colorings that corresponds to the permutation π_i of labelings shown in Figure 8.18. It is impractical to write out π_i^*. However, note that $\pi_1^* = I^*$ leaves invariant all 2^7 tree colorings, so $\text{Inv}(\pi_1^*) = 2^7 = 128$. Also, permutation π_2 interchanges vertices 4 and 5. Thus, π_2^* leaves invariant exactly those colorings that color vertices 4 and 5 the same, that is, 2^6 colorings. Thus, $\text{Inv}\ (\pi_2^*) = 2^6 = 64$. Similarly, $\text{Inv}(\pi_3^*) = 2^6 = 64$, $\text{Inv}(\pi_4^*) = 2^4 = 16$, $\text{Inv}(\pi_5^*) = 2^5 = 32$, $\text{Inv}(\pi_6^*) = 2^3 = 8$, $\text{Inv}(\pi_7^*) = 2^3 = 8$, and $\text{Inv}(\pi_8^*) = 2^4 = 16$. Thus, the number of patterns or the number of distinct colorings is

$$\frac{1}{8}(128 + 64 + 64 + 16 + 32 + 8 + 8 + 16) = 42.$$

8.4.3 Graph Colorings Equivalent under Automorphisms

Suppose that we wish to color the vertices of graph $K_{1,3}$ of Figure 8.15 using the colors green (G), black (B), or white (W), with no requirement that two vertices joined by an edge get different colors. We shall consider two such colorings equivalent if one can be obtained from the other by an automorphism. Any two colorings with vertex 1 of Figure 8.15 getting color G and the others all getting different colors are equivalent, since either a rotation by 120° or 240° or a reflection about the edge $\{1, 2\}$, $\{1, 3\}$, or $\{1, 4\}$ can be used to map one such coloring into another. These colorings are shown in Figure 8.19. The rotations also show that all colorings with vertex 1 getting G, two of the other vertices getting B, and the last getting W are equivalent. These colorings are shown in Figure 8.20. Similarly, for any choice of two distinct colors from {G, B, W}, $X \neq Y$, neither equal to G, there is a pattern (set of equivalent colorings) with vertex 1 getting G, two other vertices getting the first chosen color X from $\{X, Y\}$, and the last vertex getting the other chosen color Y from $\{X, Y\}$. There are $3 \times 2 = 6$ ways to choose the two distinct colors X and Y, so 6 patterns (equivalence classes) of this kind. There are also 3 patterns (equivalence classes) consisting of one coloring each, with vertex 1 getting color G and the other vertices all getting the same color (possibly, G). Hence, there are 10 equivalence classes of colorings in all in which vertex 1 gets color G. Repeating for the other two choices for color of vertex 1, we find 30 patterns (equivalence classes) under automorphism in all.

We can also obtain this result from Burnside's Lemma. All in all, there are 3^4 colorings of the vertices of $K_{1,3}$. Suppose that π_i^* is the permutation of graph colorings that corresponds to the automorphism π_i of $K_{1,3}$. The identity automorphism π_1 leaves all such colorings invariant, so $|\text{Inv}\,(\pi_1^*)| = 3^4$. The reflection through edge $\{1, 2\}$, π_2, leaves invariant those colorings in which vertices 3 and 4 get the same color. There are 3^3 such colorings, so $|\text{Inv}\,(\pi_2^*)| = 3^3$. Similarly, $|\text{Inv}\,(\pi_3^*)| = |\text{Inv}\,(\pi_4^*)| = 3^3$ if π_3 and π_4 are the reflections through the edges $\{1, 3\}$ and $\{1, 4\}$, respectively. A rotation through 120°, π_5, leaves invariant those colorings in which vertices 2, 3, and 4 get the same color. Thus, $|\text{Inv}\,(\pi_5^*)| = 3^2$. Similarly,

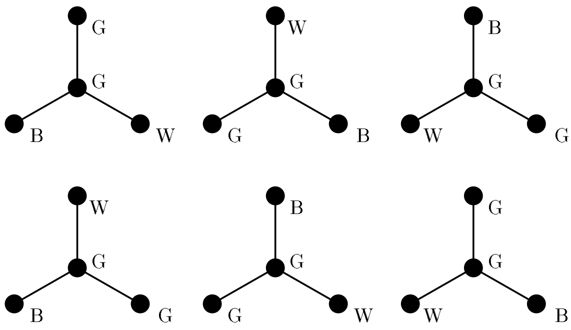


Figure 8.19: Pattern (equivalence class) of colorings of $K_{1,3}$ with vertex 1 of Figure 8.15 getting color G and vertices 2, 3, 4 of Figure 8.15 getting distinct colors from {G, B, W}.

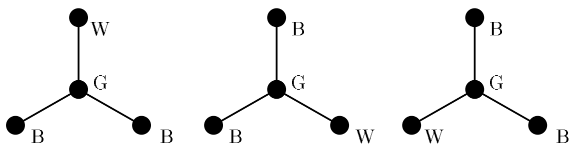


Figure 8.20: Pattern (equivalence class) of colorings of $K_{1,3}$ with vertex 1 of Figure 8.15 getting color G and vertices 2, 3, 4 of Figure 8.15 having two colored B and one colored W.

$|\text{Inv}\,(\pi_6^*)| = 3^2$ for π_6, the rotation through 240°. There are 6 permutations π_i^* in all, so Burnside's Lemma shows that the number of patterns (equivalence classes) of colorings under automorphism is

$$\frac{1}{6}\left[3^4 + 3^3 + 3^3 + 3^3 + 3^2 + 3^2\right] = \frac{1}{6}[180] = 30.$$

8.4.4 The Case of Switching Functions[9]

Let us now apply the theory we have been developing to the case of switching functions, Example 8.3. If there are two variables, we considered two such functions T and U the same if $T = U$ or $T(x_1x_2) = U(x_2x_1)$. This idea generalizes as follows: Two switching functions T and U of n variables are considered the same if there is a permutation π of $\{1, 2, \ldots, n\}$ so that

$$T(x_1x_2\cdots x_n) = U(x_{\pi(1)}x_{\pi(2)}\cdots x_{\pi(n)}). \tag{8.10}$$

In the case $n = 2$, the two possible π are

$$\begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}.$$

[9]This subsection may be omitted.

Table 8.2: Switching Functions T and U Satisfying (8.10) for π as in (8.11)

Bit string x	$T(x)$	$U(x)$
000	1	1
001	0	0
010	1	0
011	1	0
100	0	1
101	0	1
110	1	1
111	0	0

If $n = 3$, an example of two switching functions satisfying (8.10) with

$$\pi = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} \tag{8.11}$$

is given in Table 8.2. That (8.10) should correspond to sameness or equivalence makes sense. For if (8.10) holds, a circuit design for T can be obtained from one for U, in a manner analogous to Figure 8.2. Alternative sameness relations also make sense for computer engineering. We explore them in the exercises.

How does this sameness relation fit into the formal structure we have developed? D here is the set B_n of bit strings of length n. Let π be any permutation of $\{1, 2, \ldots, n\}$, and let S_n be the group of all permutations of $\{1, 2, \ldots, n\}$. Then a bit string $x_1 x_2 \cdots x_n$ can be looked at as a coloring of $\{1, 2, \ldots, n\}$ using the colors 0 and 1. The corresponding group of permutations of colorings is S_n^*. This is the group G of our theory. The group G^* is the group $(S_n^*)^*$. $G^* = (S_n^*)^*$ consists of permutations $(\pi^*)^*$ for all π in S_n. How does $(\pi^*)^*$ work? First, note that $\pi^*(x_1 x_2 \cdots x_n) = x_{\pi(1)} x_{\pi(2)} \cdots x_{\pi(n)}$. Note that S_n^* is a group of permutations of the collection of bit strings. $(S_n^*)^*$ consists of permutations of colorings of bit strings. But a coloring of bit strings using colors $0, 1$ is a switching function. Note that by definition, if U is a switching function,

$$[(\pi^*)^* U]\,(x_1 x_2 \cdots x_n) = U\,[\pi^*(x_1 x_2 \cdots x_n)] = U\left(x_{\pi(1)} x_{\pi(2)} \cdots x_{\pi(n)}\right).$$

Thus, if $T = (\pi^*)^* U$, (8.10) follows.

Let $n = 2$. Then $S_n = \{\pi_1, \pi_2\}$, where

$$\pi_1 = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix} \quad \text{and} \quad \pi_2 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}.$$

The permutations in $G = S_n^*$ are

$$\pi_1^* = \begin{pmatrix} 00 & 01 & 10 & 11 \\ 00 & 01 & 10 & 11 \end{pmatrix} \quad \text{and} \quad \pi_2^* = \begin{pmatrix} 00 & 01 & 10 & 11 \\ 00 & 10 & 01 & 11 \end{pmatrix}. \tag{8.12}$$

Table 8.3: The 16 Switching Functions of Two Variables

Bit string	$T(x)$															
x	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}
00	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

There are $2^{2^2} = 16$ switching functions of two variables. These are shown as $T_1, T_2, \ldots, T_{16}$ of Table 8.3. Then the permutations in $G^* = (S_n^*)^*$ are

$$(\pi_1^*)^* = \begin{pmatrix} T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 & T_9 & T_{10} & T_{11} & T_{12} & T_{13} & T_{14} & T_{15} & T_{16} \\ T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 & T_9 & T_{10} & T_{11} & T_{12} & T_{13} & T_{14} & T_{15} & T_{16} \end{pmatrix} \quad (8.13)$$

and

$$(\pi_2^*)^* = \begin{pmatrix} T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 & T_9 & T_{10} & T_{11} & T_{12} & T_{13} & T_{14} & T_{15} & T_{16} \\ T_1 & T_2 & T_5 & T_6 & T_3 & T_4 & T_7 & T_8 & T_9 & T_{10} & T_{13} & T_{14} & T_{11} & T_{12} & T_{15} & T_{16} \end{pmatrix}. \quad (8.14)$$

Note that $(\pi_2^*)^*U$ is the function T which does on 01 what U does on 10, and on 10 what U does on 01, and otherwise agrees with U. We have $\mathrm{Inv}((\pi_1^*)^*) = 16$ and $\mathrm{Inv}((\pi_2^*)^*) = 8$, and the number of equivalence classes or patterns of switching functions of two variables is given by $\frac{1}{2}(16+8) = 12$. The number of patterns of switching functions of three variables can similarly be shown to be 80, the number of patterns of switching functions of four variables can be shown to be 3,984, and the number of patterns of switching functions of five variables can be shown to be 37,333,248 (see Harrison [1965] or Prather [1976]). By allowing other symmetries (such as interchange of 0 and 1 in the domain or range) of a switching function—see Exercises 23 and 24—we can further reduce the number of equivalence classes. In fact, the number can be reduced to 222 if $n = 4$ (Harrison [1965], Stone [1973]). This gives a small enough number so that for $n = 4$ it is reasonable to prepare a catalog of optimal circuit for realizing switching functions which contains a representative of each equivalence class.

EXERCISES FOR SECTION 8.4

1. Suppose that $D = \{a, b, c\}$ and $R = \{1, 2\}$. Find all colorings in $C(D, R)$.
2. How many colorings (not necessarily distinct) are there for the vertices of a cube if the set of allowable colors is {red, green, blue}?
3. How many allowable colorings (not necessarily distinct) are there for the vertices of a regular tetrahedron if six colors are available?
4. In Example 8.1, check (8.5), (8.7), and (8.8) by computing:

   (a) $(\pi_2^* C_4)(3)$ (b) $(\pi_3^* C_5)(2)$ (c) $(\pi_4^* C_{11})(4)$

5. In Example 8.2, check (8.9) by computing:

   (a) $(\pi_2^* br)(1)$ (b) $(\pi_2^* br)(2)$ (c) $(\pi_2^* rr)(1)$

6. Suppose that $D = \{1,2,3,4\}, R = \{1,2\}$, and G consists of the permutations in Equation (8.3).

   (a) Suppose that f and g are the following colorings. $f(a) = 1$, all a, and that $g(1) = g(2) = 2$, $g(3) = g(4) = 1$. Is fS^*g?

   (b) Suppose that $f(1) = f(3) = 2, f(2) = f(4) = 1$ and $g(1) = g(2) = 2$, $g(3) = g(4) = 1$. Is fS^*g?

   (c) Find π_2^*. (d) Find π_3^*. (e) Find π_4^*.

   (f) Find $\mathrm{Inv}(\pi_2^*)$. (g) Find $\mathrm{Inv}(\pi_3^*)$. (h) Find $\mathrm{Inv}(\pi_4^*)$.

   (i) Find S. (j) Find S^*.

7. Repeat Exercise 6 [except parts (e) and (h)] if G consists of

$$\pi_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}, \quad \pi_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \end{pmatrix}, \quad \pi_3 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix}.$$

8. In Example 8.2, suppose that $k = 2$ and that three colors are available: red (r), blue (b), and purple (p). Find π_2^* if

$$\pi_2 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}.$$

9. In Example 8.2, suppose that $k = 3$ and that two colors are available: red (r) and blue (b).

   (a) Find G^*.

   (b) Find the number of distinct necklaces using Burnside's Lemma.

   (c) Check your answer by enumerating the distinct necklaces.

10. In Exercise 15, Section 8.1, find:

    (a) D (b) R (c) G

    (d) G^* (e) The number of distinct colorings

11. If graph $K_{1,3}$ of Figure 8.15 is colored using colors from {G, B, W}, find the following patterns:

    (a) That containing the coloring with vertex 1 colored W and vertices 2, 3, 4 colored G, B, B, respectively

    (b) That containing the coloring with all vertices colored W

12. If graph $K_{1,3}$ is colored using colors from {G, B, W, P}, find the number of patterns:

    (a) By describing them (b) Using Burnside's Lemma

13. If the graph Z_4 is colored using colors from {G, B, W}, describe all patterns and count them using Burnside's Lemma.

14. If the graph $K_4 - K_2$ of Figure 8.16 is colored using colors from {B, W}, describe all patterns and count them using Burnside's Lemma.

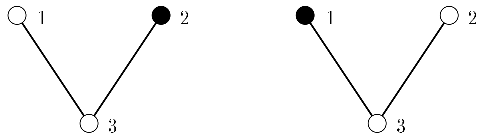


Figure 8.21: Colored weak orders.

15. In Example 8.4, verify that:

    (a) $\text{Inv}(\pi_3^*) = 2^6$ (b) $\text{Inv}(\pi_4^*) = 2^4$

    (c) $\text{Inv}(\pi_5^*) = 2^5$ (d) $\text{Inv}(\pi_6^*) = 2^3$

16. In Example 8.3, verify (8.14) by computing:

    (a) $(\pi_2^*)^* T_3$ (b) $(\pi_2^*)^* T_{12}$ (c) $(\pi_2^*)^* T_{15}$

17. In the situation of Exercise 9, Section 8.1:

    (a) Find G^*.

    (b) Use Burnside's Lemma to compute the number of distinct colorings.

    (c) Check your answer by comparing the enumeration of equivalence classes you gave as your answer in Section 8.1.

18. Find the number of distinct ways to 2-color a 4×4 array that can rotate by $0°$ or $180°$.

19. Show that the eight permutations in Figure 8.18 define a group.

20. Suppose that we consider a weak order on $\{1, 2, 3\}$ and color each element of $\{1, 2, 3\}$ dark or light. Then we distinguish weak order A from weak order B in Figure 8.21. Count the number of distinct colored weak orders if, independent of coloring, we use the notion of sameness of Example 8.6.

21. Suppose that $V = \{1, 2, \ldots, p\}$. Recall that there is a one-to-one correspondence between graphs (V, E) on the vertex set V and functions that assign 0 or 1 to each 2-element subset of V. The idea is that

    $$f(\{i, j\}) = 1 \text{ iff } \{i, j\} \in E.$$

    The function f is a coloring of the set D of all 2-element subsets of V, using the colors 0 and 1.

    (a) If $p = 3$, find all such functions f and their corresponding graphs.

    (b) If H and H' are two graphs on V and f and f' are their corresponding functions, show that H and H' are isomorphic iff there is a permutation π on D so that for all $\{i, j\}$ in D, $f(\{i, j\}) = f'(\pi(\{i, j\}))$, that is, so that f and f' are equivalent.

    (c) Let G be the group of all permutations π of D. If $p = 3$, write down all the elements of G and compute $\text{Inv}(\pi_i^*)$ for all π_i in G.

    (d) Use Burnside's Lemma to determine the number of distinct (nonisomorphic) graphs of three vertices, and verify your result by identifying the classes of equivalent (isomorphic) graphs.

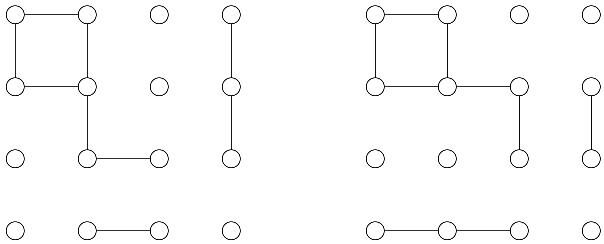


Figure 8.22: Sample interconnection patterns for chips.

22. Repeat parts (a), (c), and (d) of Exercise 21 for $p = 4$.
23. Suppose that D is the collection of all bit strings of length 3, and let G be the group that consists of the identity permutation and the permutation that complements a string by interchanging 0 and 1 (see Exercise 14, Section 8.1). Find the number of distinct switching functions (number of distinct colorings using the colors 0 and 1), that is, the number of equivalence classes in the equivalence relation induced by G^*. Note that in this case we have no S_n.
24. Suppose that two switching functions are considered equivalent if one can be obtained from the other by permuting or complementing the variables as in Exercise 23 or both. Find the number of distinct switching functions of two variables.
25. (Reingold, Nievergelt, and Deo [1977]) A manufacturer of integrated circuits makes chips that have 16 elements arranged in a 4×4 array. These elements are interconnected between some adjacent horizontal or vertical elements. Figure 8.22 shows some sample interconnection patterns. A photomask of the interconnection pattern is used to deposit interconnections on a chip. Two patterns are considered the same if the same photomask could be used for each. For instance, by flipping the photomask over on a diagonal, it can be used for both the interconnection patterns shown in Figure 8.22. Thus, they are considered the same. How many photomasks are required in order to lay out all possible interconnection patterns? Formulate this problem as a coloring problem by defining an appropriate D, R, and G. However, do not attempt to compute G^* or to solve the problem completely with the tools developed so far.

8.5 THE CYCLE INDEX

8.5.1 Permutations as Products of Cycles

It gets rather messy to apply Burnside's Lemma to many counting problems. For instance, it gets rather long and complicated to compute the permutations in the group G^*. We shall develop alternative procedures, procedures that will also allow us to get more information than provided by Burnside's Lemma.

The permutation

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix}$$

cycles the numbers around, sending 1 to $2, 2$ to $3, 3$ to $4, 4$ to 5, and 5 to 1. It can be abbreviated by writing simply (12345). More generally, $(a_1 a_2 \cdots a_{m-1} a_m)$ will represent the permutation that takes a_1 to a_2, a_2 to $a_3, \ldots, a_{m-1}$ to a_m, and a_m to a_1. This is called a *cyclic permutation.* For instance,

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$$

is a cyclic permutation (132). Now consider the permutation

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 1 & 6 & 3 & 2 & 4 \end{pmatrix}.$$

This consists of two cycles, (152) and (364). We say that

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 1 & 6 & 3 & 2 & 4 \end{pmatrix}$$

is the *product* of these two cycles, and write it as (152)(364). It is the product of these cycles in the same sense as taking the product of two permutations, if we think of a cycle like (152) as leaving 3, 4, and 6 fixed. In the same way,

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 1 & 3 & 5 & 6 & 4 \end{pmatrix}$$

is the *product* of three cycles, (12)(3)(456), where (3) means that 3 is mapped into itself. In this product, the three cycles are *disjoint* in the sense that no two of them involve the same element.

We now show that every permutation of $\{1, 2, \ldots, n\}$ can be written as the product of *disjoint* cycles, with each element i of $1, 2, \ldots, n$ appearing in some cycle. To see why, let us take the permutation

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 5 & 4 & 8 & 6 & 7 & 2 & 1 & 3 \end{pmatrix}.$$

Take the element 1. It goes to 5. In turn, 5 goes to 7, and 7 to 1, so we have a cycle (157). Take the first element not in this cycle, 2. It goes to 4, which goes to 6, which goes to 2, so we have a cycle (246). Take the first element not in either of these cycles. It is 3. Now 3 goes to 8, which goes to 3, so we have a cycle (38). Thus, the original permutation is the product of the three disjoint cycles (157)(246)(38). Similar reasoning applies to any permutation.

Could a permutation π be written in two different ways as the product of disjoint cycles (with every element in some cycle)? The answer is yes if we consider $(a_1 a_2 \cdots a_m)$ and $(a_i a_{i+1} \cdots a_m a_1 a_2 \cdots a_{i-1})$ as different. However, we consider them the same, since they correspond to the same permutation. Thus, (123) and (231) and (312) are all the same. Moreover, the order in which we write the cycles does not matter in the product of disjoint cycles. For example, we consider (12)(345) and (345)(12) to be the same. Suppose that

$$(xyz \cdots) \cdots (abc \cdots) = \pi = (uvw \cdots) \cdots (\alpha\beta\gamma \cdots).$$

If these two ways of writing the permutation are different, there must be a number k such that the cycle containing k on the left is different from the cycle containing k on the right. We can write these two cycles with k first. Then whatever π takes k into must be next in each cycle, and whatever π takes this into must be third in each cycle, and so on. Thus, the two cycles must be the same. To summarize, we have the following result.

Theorem 8.5 Every permutation of $\{1, 2, \ldots, n\}$ can be written in exactly one way as the product of disjoint cycles with every element of $\{1, 2, \ldots, n\}$ appearing in some cycle.

We shall call the unique way of writing a permutation described in Theorem 8.5 the *cycle decomposition* of the permutation.

8.5.2 A Special Case of Pólya's Theorem

We are now ready to present another result about counting equivalence classes, which is a special case of the main theorem we are aiming for. Suppose that $\text{cyc}(\pi)$ counts the number of cycles in the unique cycle decomposition of the permutation π. For instance, if $\pi = (12)(3)(456)$, then $\text{cyc}(\pi) = 3$.

Theorem 8.6 (A Special Case of Pólya's Theorem) Suppose that G is a group of permutations of the set D and $C(D, R)$ is the set of colorings of elements of D using colors in R, a set of m elements. Then the number of distinct colorings in $C(D, R)$ (the number of equivalence classes or patterns in the equivalence relation S^* induced by G^*) is given by

$$\frac{1}{|G|}\left[m^{\text{cyc}(\pi_1)} + m^{\text{cyc}(\pi_2)} + \cdots + m^{\text{cyc}(\pi_k)}\right],$$

where $G = \{\pi_1, \pi_2, \ldots, \pi_k\}$.

Note that this theorem allows us to compute the number of distinct colorings without first computing G^*. We prove the theorem in Section 8.5.6.

To illustrate this theorem, let us reconsider the case of the 2×2 arrays, Example 8.1. There are four permutations in G, the four rotations $\pi_1, \pi_2, \pi_3, \pi_4$ shown in Figure 8.13. These have the following cycle decompositions: $\pi_1 = (1)(2)(3)(4), \pi_2 = (1432), \pi_3 = (13)(24), \pi_4 = (1234)$. Thus, $\text{cyc}(\pi_1) = 4$, $\text{cyc}(\pi_2) = 1$, $\text{cyc}(\pi_3) = 2$, and $\text{cyc}(\pi_4) = 1$. The number of distinct colorings (number of patterns) is given by $\frac{1}{4}(2^4 + 2^1 + 2^2 + 2^1) = 6$, which agrees with our earlier computation.

In Example 8.2, with necklaces of k beads, we can write the two permutations in G as

$$\pi_1 = (1)(2)\cdots(k)$$

and as

$$\pi_2 = (1\ k)\,(2\ k-1)\,(3\ k-2)\cdots\left(\frac{k}{2}\ \frac{k}{2}+1\right)$$

if k is even and as

$$\pi_2 = (1\ k)\,(2\ k-1)\,(3\ k-2)\cdots\left(\frac{k-1}{2}\ \frac{k+3}{2}\right)\left(\frac{k+1}{2}\right)$$

if k is odd. For instance, if k is 4, $\pi_2 = (14)(23)$. If k is 5, $\pi_2 = (15)(24)(3)$. It follows that $\text{cyc}(\pi_1) = k$ and $\text{cyc}(\pi_2) = k/2$ if k is even and $(k+1)/2$ if k is odd. Hence, the number of distinct necklaces is $\frac{1}{2}(2^k + 2^{k/2})$ if k is even and $\frac{1}{2}(2^k + 2^{(k+1)/2})$ if k is odd. For instance, the case $k = 2$ gives us 3, which agrees with our earlier computation. The case $k = 3$ gives us 6, which is left to the reader to check (Exercise 8). If there are three different colors of beads and k is even, we would have $\frac{1}{2}(3^k + 3^{k/2})$ distinct necklaces. For instance, for $k = 2$, we would have 6 distinct necklaces. If the colors of beads are r, b, and p, the 6 equivalence classes are $\{rb, br\}, \{rp, pr\}, \{bp, pb\}, \{rr\}, \{bb\}, \{pp\}$.

In Example 8.4, the tree colorings, G is given by the eight permutations π_1, π_2, ..., π_8 of Figure 8.18. We have

$$\begin{array}{rclrcl}
\pi_1 &=& (1)(2)(3)(4)(5)(6)(7), & \pi_2 &=& (1)(2)(3)(45)(6)(7), \\
\pi_3 &=& (1)(2)(3)(4)(5)(67), & \pi_4 &=& (1)(23)(46)(57), \\
\pi_5 &=& (1)(2)(3)(45)(67), & \pi_6 &=& (1)(23)(4756), \\
\pi_7 &=& (1)(23)(4657), & \pi_8 &=& (1)(23)(47)(56).
\end{array}$$

Then the number of distinct tree colorings is given by

$$\frac{1}{8}(2^7 + 2^6 + 2^6 + 2^4 + 2^5 + 2^3 + 2^3 + 2^4) = 42,$$

which is what we computed earlier.

8.5.3 Graph Colorings Equivalent under Automorphisms Revisited

Let us return to the colorings of the vertices of graph $K_{1,3}$ as discussed in Section 8.4.3. It is easy to see that

$$\pi_1^* = (1)(2)(3)(4), \quad \pi_2^* = (1)(2)(3\ 4), \quad \pi_3^* = (1)(3)(2\ 4),$$
$$\pi_4^* = (1)(4)(2\ 3), \quad \pi_5^* = (1)(3\ 4\ 2), \quad \pi_6^* = (1)(4\ 2\ 3).$$

Thus, if we color using colors G, B, W, the number of distinct colorings is given by

$$\frac{1}{6}\left[3^{\text{cyc}(\pi_1^*)} + 3^{\text{cyc}(\pi_2^*)} + 3^{\text{cyc}(\pi_3^*)} + 3^{\text{cyc}(\pi_4^*)} + 3^{\text{cyc}(\pi_5^*)} + 3^{\text{cyc}(\pi_6^*)}\right]$$

$$= \frac{1}{6}\left[3^4 + 3^3 + 3^3 + 3^3 + 3^2 + 3^2\right] = 30.$$

This agrees with our earlier computation.

8.5.4 The Case of Switching Functions[10]

In Example 8.3, the switching functions, we have to consider the elements of $G = H^*$. In case we consider switching functions of two variables, we have $G = \{\pi_1^*, \pi_2^*\}$, where

$$\pi_1 = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix} \quad \text{and} \quad \pi_2 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}.$$

By our computation in Section 8.4.4, π_1^* and π_2^* are given by (8.12). Now we can think of π_1^* and π_2^* as acting on $\{1,2,3,4\}$, in which case $\pi_1^* = (1)(2)(3)(4)$ and $\pi_2^* = (1)(23)(4)$, so $\text{cyc}(\pi_1^*) = 4$, $\text{cyc}(\pi_2^*) = 3$, and the number of distinct switching functions is $\frac{1}{2}(2^4 + 2^3) = 12$, which agrees with our earlier computation.

8.5.5 The Cycle Index of a Permutation Group

It will be convenient to summarize the cycle structure of the permutations in a permutation group in a manner analogous to generating functions. Suppose that π is a permutation with b_1 cycles of length 1, b_2 cycles of length 2, ... in its unique cycle decomposition. Then if $x_1, x_2, \ldots$ are placeholders and k is at least the length of the longest cycle in the cycle decomposition of π, we can encode π by using the expression $x_1^{b_1} x_2^{b_2} \cdots x_k^{b_k}$. Moreover, we can encode an entire permutation group G by taking the sum of these expressions for members of G divided by the number of permutations of G. That is, if k is the length of the longest cycle in the cycle decomposition[11] of any π of G, we write

$$P_G(x_1, x_2, \ldots, x_k) = \frac{1}{|G|} \sum_{\pi \in G} x_1^{b_1} x_2^{b_2} \cdots x_k^{b_k}$$

and call $P_G(x_1, x_2, \ldots, x_k)$ the *cycle index* of G. For instance, consider Example 8.4. Then, to use the notation of Figure 8.18, $\pi_6 = (1)(23)(4756)$, and its corresponding code is $x_1 x_2 x_4$. Also, $\pi_4 = (1)(23)(46)(57)$, and it is encoded as $x_1 x_2^3$. By a similar analysis, the cycle index for the group of permutations is

$$\begin{aligned} P_G(x_1, x_2, \ldots, x_8) = {} & \frac{1}{8} \left[x_1^7 + x_1^5 x_2 + x_1^5 x_2 + x_1 x_2^3 + \right. \\ & \left. x_1^3 x_2^2 + x_1 x_2 x_4 + x_1 x_2 x_4 + x_1 x_2^3 \right]. \end{aligned} \tag{8.15}$$

Note that if π is a permutation with corresponding code $x_1^{b_1} x_2^{b_2} \cdots x_k^{b_k}$, then $\text{cyc}(\pi) = b_1 + b_2 + \cdots + b_k$ and $x_1^{b_1} x_2^{b_2} \cdots x_k^{b_k}$ is $m^{\text{cyc}(\pi)}$ if all x_i are taken to be m. Hence, Theorem 8.6 can be restated as follows:

Corollary 8.6.1 Suppose that G is a group of permutations of the set D and that $C(D, R)$ is the set of colorings of elements of D using colors in R, a set

[10] This subsection may be omitted.

[11] We can also take $k = |D|$ and note that the length of the longest cycle in the cycle decomposition of any π of G is at most k.

of m elements. Then the number of distinct colorings in $C(D,R)$ is given by $P_G(m,m,\ldots,m)$.

It is this version of the result that we generalize in Section 8.6.

To make use of this result here, we return to Example 8.4. Here $m=2$. We let $x_1=x_2=\cdots=x_8=2$ in (8.15), obtaining

$$P_G(2,2,\ldots,2)=\frac{1}{8}(2^7+2^6+2^6+2^4+2^5+2^3+2^3+2^4)=42,$$

which agrees with our earlier result about the number of distinct colorings.

Let us now consider Example 8.5. In Section 8.1 we identified 12 different symmetries of the tetrahedron. These can be thought of as permutations of the letters a,b,c,d of Figure 8.5. The identity symmetry is the permutation $(a)(b)(c)(d)$, which has cycle structure code x_1^4. The $120°$ rotation about the line joining vertex a to the middle of the face determined by b, c, and d corresponds to the permutation $(a)(bdc)$, which can be encoded as x_1x_3 (see Figure 8.6). All eight $120°$ and $240°$ rotational symmetries have similar structure and coding. Finally, the rotation by $180°$ about the line connecting the midpoints of edges ab and cd corresponds to the permutation $(ab)(cd)$, which has the coding x_2^2. The other two $180°$ rotations have similar coding. Thus, the cycle index is given by

$$P_G(x_1,x_2,x_3)=\frac{1}{12}(x_1^4+8x_1x_3+3x_2^2).$$

We seek a coloring of the set $D=\{a,b,c,d\}$ using the four colors contained in the set $R=\{\mathrm{CH_3},\mathrm{C_2H_5},\mathrm{H},\mathrm{Cl}\}$. Hence, $m=4$, and the number of distinct colorings (number of distinct molecules) is given by

$$P_G(m,m,m)=\frac{1}{12}\left[4^4+8(4)(4)+3(4)^2\right]=36.$$

8.5.6 Proof of Theorem 8.6[12]

We shall apply Burnside's Lemma (Theorem 8.4) to G^* in order to prove Theorem 8.6. Since $|G|=|G^*|$, it suffices to show that $m^{\mathrm{cyc}(\pi)}=\mathrm{Inv}(\pi^*)$. Let π be in G. We try to compute $\mathrm{Inv}(\pi^*)$. Note that an element of $C(D,R)$ is left invariant by π^* iff in the corresponding permutation π of D, all the elements of D in each cycle of π receive the same color. For instance, suppose that $\pi=(12)(345)(67)(8)$. Let f be the coloring such that $f(1)=f(2)=$ black, $f(3)=f(4)=f(5)=$ white, $f(6)=f(7)=$ red, and $f(8)=$ blue. Then clearly π^*f is the same coloring.

In sum, to find a coloring that is left invariant by π^*, we compute the cycle decomposition of π and color each element in a cycle with the same color. Now π has $\mathrm{cyc}(\pi)$ different cycles in its cycle decomposition, and we have m choices for the common color of each cycle. Hence, there are $m^{\mathrm{cyc}(\pi)}$ different colorings left invariant by π^*. In short, $\mathrm{Inv}(\pi^*)=m^{\mathrm{cyc}(\pi)}$.

[12] This subsection may be omitted.

EXERCISES FOR SECTION 8.5

1. Find the cycle decomposition of each permutation of Exercise 1, Section 8.2.
2. Find the cycle decomposition for all permutations π_i arising from the situation of Exercise 9, Section 8.1.
3. Compute $\mathrm{cyc}(\pi)$ for every permutation in parts (a), (b), and (c) of Exercise 10, Section 8.2.
4. For every permutation of parts (a), (b), (c) of Exercise 10, Section 8.2, encode the permutation as $x_1^{b_1} x_2^{b_2} \cdots x_k^{b_k}$.
5. For each group of permutations in Exercise 10, Section 8.2, compute the cycle index.
6. Given D, R, and G as in Exercise 6, Section 8.4:

   (a) Find the number of distinct colorings by Theorem 8.6.
   (b) Repeat using Corollary 8.6.1.

7. Repeat Exercise 6 for D, R, and G as in Exercise 7, Section 8.4.
8. In Example 8.2, check that if $k = 3$, there are six distinct necklaces.
9. In Example 8.2, use Theorem 8.6 to find the number of distinct necklaces if:

   (a) The number of colors is 2 and k is 4
   (b) The number of colors is 2 and k is 5
   (c) The number of colors is 3 and k is 3
   (d) The number of colors is 3 and k is 4

10. Repeat Exercise 9 using Corollary 8.6.1.
11. (a) In Exercise 9, Section 8.1, use Theorem 8.6 to find the number of equivalence classes.
    (b) Repeat using Corollary 8.6.1.
12. (a) In Exercise 15, Section 8.1, use Theorem 8.6 to find the number of distinct colorings.
    (b) Repeat using Corollary 8.6.1.
13. (a) In Exercise 8, Section 8.1, use Theorem 8.6 to find the number of distinct colorings.
    (b) Repeat using Corollary 8.6.1.
14. In Example 8.1, for each $\pi_i, i = 1, 2, 3, 4$, verify that $\mathrm{Inv}(\pi_i^*) = m^{\mathrm{cyc}(\pi_i)}$.
15. Continuing with Exercise 23 of Section 8.2, if equivalence of two graph colorings is defined as in Section 8.4.3, find the number of distinct colorings of:

    (a) Graph L_4 with 2 colors
    (b) Graph L_4 with 3 colors
    (c) Graph Z_4 with 2 colors
    (d) Graph Z_4 with 3 colors
    (e) Graph $K_4 - K_2$ with 2 colors
    (f) Graph $K_4 - K_2$ with 3 colors

16. Repeat Exercise 20, Section 8.4, using the methods of this section.
17. (a) Use Theorem 8.6 to find the number of nonisomorphic graphs of $p = 3$ vertices. (See Exercise 21, Section 8.4.)
    (b) Repeat using Corollary 8.6.1.
18. Repeat Exercise 17 for $p = 4$.

19. (a) In Exercise 23, Section 8.4, use Theorem 8.6 to find the number of distinct switching functions.
    (b) Repeat using Corollary 8.6.1.
20. Repeat Exercise 19 for Exercise 24, Section 8.4.
21. If the definition of sameness of Exercise 24, Section 8.4, is adopted, find the number of distinct switching functions of three variables given that the cycle index for the appropriate group of permutations is
$$\frac{1}{12}\left[x_1^8+4x_2^4+2x_1^2x_3^2+2x_2x_6+3x_1^4x_2^2\right].$$
22. Consider a cube in 3-space. There are eight vertices. The following symmetries correspond to permutations of these vertices. Encode each of these symmetries in the form $x_1^{b_1}x_2^{b_2}\cdots x_k^{b_k}$ and compute the cycle index of the group G of all the permutations corresponding to these symmetries.
    (a) The identity symmetry
    (b) Rotations by 180° around lines connecting the centers of opposite faces (there are three)
    (c) Rotations by 90° or 270° around lines connecting the centers of opposite faces (there are six)
    (d) Rotations by 180° around lines connecting the midpoints of opposite edges (there are six)
    (e) Rotations by 120° around lines connecting opposite vertices (there are eight)
23. In Exercise 22, find the number of distinct ways of coloring the vertices of the cube with two colors, red and blue.
24. Complete the solution of Exercise 25, Section 8.4.
25. A *transposition* is a cycle (ij). Show that every permutation is the product of transpositions. (*Hint:* It suffices to show that every cycle is the product of transpositions.)
26. Continuing with Exercise 25, write (123456) as a product of transpositions.
27. Write each permutation of Exercise 1, Section 8.2, as the product of transpositions.
28. Show that a permutation can be written as a product of transpositions in more than one way.
29. Although a permutation can be written in more than one way as a product of transpositions, it turns out that every way of writing the permutation as such a product either includes an even number of transpositions or an odd number. (For a proof, see Exercise 31.) A permutation, therefore, can be called *even* if every way of writing it as a product of transpositions uses an even number of transpositions, and *odd* otherwise.
    (a) Identify all even permutations of $\{1,2,3\}$.
    (b) Show that the collection of even permutations of $\{1,2,\ldots,n\}$ forms a group.
    (c) Does the collection of odd permutations of $\{1,2,\ldots,n\}$ form a group?
30. The number of permutations of $\{1,2,\ldots,n\}$ with code $x_1^{b_1}x_2^{b_2}\cdots x_k^{b_k}$ is given by the formula
$$\frac{n!}{b_1!b_2!\cdots b_n!1^{b_1}2^{b_2}\cdots n^{b_n}}.$$
This is called *Cauchy's formula.*

(a) Verify this formula for $n = 5, b_1 = 3, b_2 = 1, b_3 = b_4 = b_5 = 0$.

(b) Verify this formula for $n = 3$ and all possible codes.

31. Suppose that

$$D_n = (2-1)(3-2)(3-1)(4-3)(4-2)(4-1)\cdots(n-1). \tag{8.16}$$

If π is a permutation of $\{1, 2, \ldots, n\}$, define πD_n from D_n by replacing the term $(i-j)$ in (8.16) by the term $(\pi(i) - \pi(j))$.

(a) Find D_5.

(b) Find πD_5 if

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 5 & 2 & 1 \end{pmatrix}.$$

(c) Show that if π is a transposition, $\pi D_n = -D_n$.

(d) Conclude from part (c) that if π is the product of an even number of transpositions, $\pi D_n = D_n$, and if π is the product of an odd number of transpositions, $\pi D_n = -D_n$.

(e) Conclude that a permutation cannot both be written as the product of an even number of transpositions and the product of an odd number of transpositions.

8.6 PÓLYA'S THEOREM

8.6.1 The Inventory of Colorings

We may be interested in counting not just the number of distinct colorings, but the number of distinct colorings of a certain kind. For instance, in Example 8.1, we might be interested in counting the number of distinct 2-colorings of the 2×2 array in which exactly two black colors are used; in Example 8.5, we might be interested in counting the number of distinct molecules with at least one hydrogen atom; and so on. We shall now present a general result for answering questions of this type.

Let $D = \{a_1, a_2, \ldots, a_n\}$ be the set of objects to be colored and $R = \{r_1, r_2, \ldots, r_m\}$ be the set of colors. We shall distinguish colorings by assigning a *weight* $w(r)$ to each color r. This weight can be either a symbol or a number.

If we have assigned weights to the colors, we can assign a *weight to a coloring*. It is defined to be the product of the weights of the colors assigned to the elements of D. To illustrate this, suppose that $R = \{x, y, z\}$ and $w(x) = 1, w(y) = 5, w(z) = 7$. Suppose that the objects being colored are the seven vertices of the first binary tree of Figure 8.18, and we color vertices $1, 2, 4, 6$ with color x, vertices $3, 7$ with color y, and vertex 5 with color z. Then the weight of this coloring is $w(x)^4 w(y)^2 w(z) = (1)^4(5)^2 7 = 175$. If $w(x) = r, w(y) = g$, and $w(z) = b$, the weight of the coloring is $r^4 g^2 b$. We shall see below how the weight of a coloring encodes the coloring in a very useful way.

Suppose now that K is a set of colorings. The sum of the weights of colorings in K is called the *inventory* of K. For instance, suppose that $D = \{a, b, c, d\}, R =$

$\{x, y, z\}$, and $w(x) = r, w(y) = g, w(z) = b$. Let colorings f_1, f_2, and f_3 in $C(D, R)$ be defined as follows:

$$\begin{array}{llll} f_1(a) = x, & f_1(b) = y, & f_1(c) = y, & f_1(d) = z, \\ f_2(a) = z, & f_2(b) = z, & f_2(c) = x, & f_2(d) = z, \\ f_3(a) = x, & f_3(b) = z, & f_3(c) = y, & f_3(d) = x. \end{array}$$

Let $W(f_i)$ be the weight of coloring f_i. Then $W(f_1) = w(x)w(y)w(y)w(z) = rg^2b$, and, similarly, $W(f_2) = rb^3$ and $W(f_3) = r^2gb$. The inventory of the set $K = \{f_1, f_2, f_3\}$ is given by $rg^2b + rb^3 + r^2gb$. If all the weights of colors are different symbols, the weight of a coloring represents the distribution of colors used. For instance, $W(f_1) = rg^2b$ shows that f_1 used color x once, color y twice, and color z once. The inventory of a set of colorings summarizes the distribution of colors in the different colorings in the set. This is like a generating function.

Now suppose that G is a group of permutations of the set D and that f and g are two equivalent colorings in $C(D, R)$. Then as observed in Section 8.4.2, there is a π in G so that for all a in D, $g(a) = f(\pi(a))$ If $D = \{a_1, a_2, \ldots, a_n\}$, then

$$W(f) = w[f(a_1)]w[f(a_2)] \cdots w[f(a_n)] \tag{8.17}$$

and

$$W(g) = w[g(a_1)]w[g(a_2)] \cdots w[g(a_n)]. \tag{8.18}$$

Since π is a permutation, the set $\{a_1, a_2, \ldots, a_n\}$ has exactly the same elements as the set $\{\pi(a_1), \pi(a_2), \ldots, \pi(a_n)\}$. Thus, (8.17) implies that

$$W(f) = w[f(\pi(a_1))]w[(\pi(a_2))] \cdots w[f(\pi(a_n))]. \tag{8.19}$$

But since $g(a) = f(\pi(a))$, (8.18) and (8.19) imply that $W(f) = W(g)$. Thus, we have shown the following.

Theorem 8.7 If colorings f and g are equivalent, they have the same weight.

As a result of this theorem, we can speak of the *weight of an equivalence class of colorings* or, what is the same, the *weight of a pattern.* This is the weight of any coloring in this class. We shall also be able to speak of the *inventory of a set of patterns* or of a set of equivalence classes, the *pattern inventory*, as the sum of the weights of the patterns in the set. For instance, let us consider Example 8.1, the colorings of the 2×2 arrays. There are six patterns of colorings, as shown in Figure 8.10. Let the color black have weight b and the color white have weight w. Then the coloring of class 1 of Figure 8.10 has weight b^4, all the colorings of class 2 have weight b^3w, all of class 3 have weight b^2w^2, all of class 4 have weight b^2w^2, all of class 5 have weight bw^3, and the coloring of class 6 has weight w^4. Note that two different equivalence classes can have the same weight. The pattern inventory is given by

$$b^4 + b^3w + 2b^2w^2 + bw^3 + w^4. \tag{8.20}$$

We find that there is one equivalence class using four black colors, one using three black and one white, two using two blacks and two whites, and so on. This information can be read directly from the pattern inventory. If we simply wanted to find the number of patterns, we would proceed as we did with generating functions in Chapter 5, and take all the weights to be 1. Here, setting $b = w = 1$ in (8.20) gives us 6, the number of patterns. If we wanted to find the number of patterns using no black, we would set $w(\text{black}) = 0$ and $w(\text{white}) = 1$, or, equivalently, let $b = 0$ and $w = 1$ in (8.20). The result is 1. There is only one pattern with no black, that corresponding to the term w^4 in the pattern inventory. We shall now seek a method for computing the pattern inventory without knowing the equivalence classes.

8.6.2 Computing the Pattern Inventory

Theorem 8.8 (Pólya's Theorem[13]) Suppose that G is a group of permutations on a set D and $C(D,R)$ is the collection of all colorings of D using colors in R. If w is a weight assignment on R, the pattern inventory of colorings in $C(D,R)$ is given by

$$P_G\left(\sum_{r\in R} w(r), \sum_{r\in R}[w(r)]^2, \sum_{r\in R}[w(r)]^3, \ldots, \sum_{r\in R}[w(r)]^k\right),$$

where $P_G(x_1, x_2, x_3, \ldots, x_k)$ is the cycle index.

Note that Corollary 8.6.1 is a special case of this theorem in which $w(r) = 1$ for all r in R. To illustrate the theorem, let us return to Example 8.1, the 2×2 arrays, one more time. Note that G consists of the permutations $\pi_1 = (1)(2)(3)(4), \pi_2 = (1432), \pi_3 = (13)(24)$, and $\pi_4 = (1234)$. Thus, the cycle index is given by

$$P_G(x_1, x_2, x_3, x_4) = \frac{1}{4}(x_1^4 + 2x_4 + x_2^2).$$

Now let us assign a weight b to a black coloring and a weight w to a white coloring. Then $R = \{\text{black, white}\}$ and

$$\begin{array}{ll} \displaystyle\sum_{r\in R} w(r) = b + w, & \displaystyle\sum_{r\in R}[w(r)]^2 = b^2 + w^2, \\ \displaystyle\sum_{r\in R}[w(r)]^3 = b^3 + w^3, & \displaystyle\sum_{r\in R}[w(r)]^4 = b^4 + w^4. \end{array}$$

By Pólya's Theorem, the pattern inventory is given by taking $P_G(x_1, x_2, x_3, x_4)$ and substituting $\sum_{r\in R} w(r)$ for x_1, $\sum_{r\in R}[w(r)]^2$ for x_2, and so on. Thus, the pattern inventory is

$$\frac{1}{4}[(b+w)^4 + 2(b^4 + w^4) + (b^2 + w^2)^2]. \tag{8.21}$$

[13] Pólya's fundamental theorem was first presented in his classic paper (Pólya [1937]). The result was anticipated by Redfield [1927], but few people understood Redfield's results and Pólya was unaware of them. A generalization of Pólya's Theorem can be found in de Bruijn [1959]–for an exposition of this, see, for example, Liu [1968]. A well-known exposition of Pólya theory is in a paper by de Bruijn [1964]. For other expositions, see Bogart [1999], Brualdi [1999], or Tucker [1995].

b	b
r	w

b	b
w	r

b	r
w	b

Figure 8.23: Examples of the three different patterns of colorings of the 2×2 array using two blacks, one white, and one red.

By using the binomial expansion (Theorem 2.7), we can expand out (8.21) and obtain (8.20), which was our previous description of the pattern inventory.

Suppose that we allow three colors in coloring the 2×2 array: black, white, and red. If we let $w(\text{red}) = r$, we find that the pattern inventory is

$$\frac{1}{4}[(b+w+r)^4 + 2(b^4+w^4+r^4) + (b^2+w^2+r^2)^2] = b^4 + w^4 + r^4 + b^3w + w^3b$$
$$+b^3r + r^3b + w^3r + r^3w + 2b^2w^2 + 2b^2r^2 + 2w^2r^2 + 3b^2wr + 3w^2br + 3r^2wb.$$

We see, for instance, that there are three patterns with two blacks, one white, and one red. One example of each of these patterns is shown in Figure 8.23. Any other pattern using two black, one white, and one red can be obtained from one of these by rotation. The number of patterns is obtained by substituting $b = w = r = 1$ into the pattern inventory. Notice how once having computed the cycle index, we can apply it easily to do a great many different counting procedures without having to repeat computation of the index.

Let us next consider Example 8.5, the organic molecules. We have already noted that

$$P_G(x_1, x_2, x_3) = \frac{1}{12}(x_1^4 + 8x_1x_3 + 3x_2^2).$$

Suppose that we want to find the number of distinct molecules (patterns) containing at least one chlorine atom. It is a little easier to compute first the number of patterns having no chlorine atoms. This can be obtained by assigning the weight of 1 to each color CH_3, C_2H_5, and H, and the weight of 0 to the color Cl. Then for all $k \geq 1$,

$$\sum_{r \in R}[w(r)]^k = [w(\mathrm{CH_3})]^k + [w(\mathrm{C_2H_5})]^k + [w(\mathrm{H})]^k + [w(\mathrm{Cl})]^k = 1+1+1+0 = 3.$$

It follows that the pattern inventory is given by $\frac{1}{12}(3^4 + 8(3)(3) + 3(3)^2) = 15$. Since we have previously calculated that there are 36 patterns in all, the number with at least one chlorine atom is $36 - 15 = 21$.

Continuing with this example, suppose that we assign a weight of 1 to each color except Cl and a weight of c to Cl. Then the pattern inventory is given by

$$\frac{1}{12}[(c+3)^4 + 8(c+3)(c^3+3) + 3(c^3+3)^2] = c^4 + 3c^3 + 6c^2 + 11c + 15.$$

We conclude that there is one pattern consisting of four chlorine atoms, while three patterns consist of three chlorine atoms, six of two chlorine atoms, 11 of one chlorine atom, and 15 of no chlorine atoms.

Next, let us return to the graph colorings of $K_{1,3}$ using colors in the set {G, B, W} as discussed in Sections 8.4.4 and 8.5.4. How many distinct colorings use no G's? The longest cycle in a permutation π_i^* has length 3, and from the results of Section 8.5.4, we see that

$$P_G(x_1, x_2, x_3) = \frac{1}{6}(x_1^4 + 3x_1^2 x_2 + 2x_1 x_3).$$

Letting $w(\mathrm{G}) = g$, $w(\mathrm{B}) = b$, $w(\mathrm{W}) = w$, we get

$$\begin{aligned} P_G(g, b, w) &= \frac{1}{6}\Big[(g+b+w)^4 + 3\,(g+b+w)^2\left(g^2+b^2+w^2\right) \\ &\quad + 2\,(g+b+w)\left(g^3+b^3+w^3\right)\Big]. \end{aligned} \tag{8.22}$$

We can answer our question by setting $g = 0$, $b = w = 1$ in (8.22), getting

$$\frac{1}{6}\,[16 + 24 + 8] = 8.$$

Thus, there are 8 distinct colorings of $K_{1,3}$ using no G. The reader should check this. What if we want to know how many colorings have exactly two W's? We can set $g = b = 1$ in (8.22) and calculate the pattern inventory

$$\frac{1}{6}\left[(2+w)^4 + 3\,(2+w)^2\left(2+w^2\right) + 2\,(2+w)\left(2+w^3\right)\right].$$

Simplifying, we get

$$8 + 10w + 7w^2 + 4w^3 + w^4. \tag{8.23}$$

The number of distinct colorings with two W's is given by the coefficient of w^2, i.e., 7. The reader should check this.

8.6.3 The Case of Switching Functions[14]

Next let us turn to Example 8.3, the switching functions, and take $n = 2$. Then G consists of the permutations π_1^* and π_2^* given by (8.12). As before, it is natural to think of π_1^* as (1)(2)(3)(4) and π_2^* as (1)(23)(4). Hence,

$$P_G(x_1, x_2) = \frac{1}{2}(x_1^4 + x_1^2 x_2).$$

Setting $w(0) = a$ and $w(1) = b$, we find that $\sum_{r \in R}[w(r)]^k = a^k + b^k$. Thus, the pattern inventory is given by

$$\frac{1}{2}[(a+b)^4 + (a+b)^2(a^2+b^2)] = a^4 + 3a^3b + 4a^2b^2 + 3ab^3 + b^4.$$

The term $3a^3b$ indicates that there are three patterns of switching functions which assign three 0's and one 1. The reader might wish to identify these patterns.

[14] This subsection may be omitted.

8.6.4 Proof of Pólya's Theorem[15]

We now present a proof of Pólya's Theorem. We proceed by a series of lemmas. Throughout, let us assume that $R = \{1, 2, \ldots, m\}$.

Lemma 8.2 Suppose that D is divided up into disjoint sets $D_1, D_2, \ldots, D_p$. Let C be the subset of $C(D, R)$ that consists of all colorings f with the property that if a and b are both in D_i, some i, then $f(a) = f(b)$. Then the inventory of the set C is given by

$$\left[w(1)^{|D_1|} + w(2)^{|D_1|} + \cdots + w(m)^{|D_1|}\right] \times \left[w(1)^{|D_2|} + w(2)^{|D_2|} + \cdots \right.$$
$$\left. + \, w(m)^{|D_2|}\right] \times \cdots \times \left[w(1)^{|D_p|} + w(2)^{|D_p|} + \cdots + w(m)^{|D_p|}\right]. \tag{8.24}$$

Proof. Multiplying out (8.24), we get terms such as

$$w(i_1)^{|D_1|} w(i_2)^{|D_2|} \cdots w(i_p)^{|D_p|}.$$

This is the weight of the coloring that gives color i_1 to all elements of D_1, color i_2 to all elements of D_2, and so on. Thus, (8.24) gives the sum of the weights of colorings that color all of D_i the same color. Q.E.D.

Lemma 8.3 Suppose that $G^* = \{\pi_1^*, \pi_2^*, \ldots\}$ is a group of permutations of $C(D, R)$. For each π^* in G^*, let $\bar{w}(\pi^*)$ be the sum of the weights of all colorings f in $C(D, R)$ left invariant by π^*. Suppose that $C_1, C_2, \ldots$ are the equivalence classes of colorings and $w(C_i)$ is the common weight of all f in C_i. Then

$$w(C_1) + w(C_2) + \cdots = \frac{1}{|G^*|}[\bar{w}(\pi_1^*) + \bar{w}(\pi_2^*) + \cdots]. \tag{8.25}$$

Note that if all weights are 1, Lemma 8.3 reduces to Burnside's Lemma.

Proof of Lemma 8.3. The sum on the right-hand side of (8.25) adds up for each π^* the weights of all colorings f left fixed by π^*. Thus, $w(f)$ is added in here exactly the number of times it is left invariant by some π^*. This is, to use the terminology of Section 8.3.2, the number of elements in the stabilizer of f, $\mathrm{St}(f)$. By Lemma 8.1 of Section 8.3.2, $|\mathrm{St}(f)| = |G^*|/|C(f)|$, where $C(f)$ is the equivalence class containing f. Therefore, if $C(D, R) = \{f_1, f_2, \ldots\}$, the right-hand side of (8.25) is given by

$$\frac{1}{|G^*|}[w(f_1) \cdot |\mathrm{St}(f_1)| + w(f_2) \cdot |\mathrm{St}(f_2)| + \cdots] =$$
$$\frac{1}{|G^*|}\left[w(f_1)\frac{|G^*|}{|C(f_1)|} + w(f_2)\frac{|G^*|}{|C(f_2)|} + \cdots\right],$$

which equals

$$\frac{w(f_1)}{|C(f_1)|} + \frac{w(f_2)}{|C(f_2)|} + \cdots. \tag{8.26}$$

[15] This subsection may be omitted.

If we add up the terms $w(f_i)/|C(f_i)|$ for f_i in equivalence class C_j, we get $w(C_j)$, since each $w(f_i) = w(C_j)$ and since $|C(f_i)| = |C_j|$. Thus, (8.26) equals $w(C_1) + w(C_2) + \cdots$. Q.E.D.

We are now ready to complete the proof of Pólya's Theorem. In (8.25) of Lemma 8.3, the left-hand side is the pattern inventory. Recall that $\bar{w}(\pi^*)$ is the sum of the weights of the colorings f left invariant by π^*. Suppose that the permutation π has cycles $D_1, D_2, \ldots, D_p$ in its cycle decomposition. Note that a coloring f is left invariant by π^* iff $f(a) = f(b)$ whenever a and b are in the same D_i. Thus, by Lemma 8.2, (8.24) gives the inventory or the sum of the weights of the set of colorings left invariant by π^*, i.e., (8.24) gives $\bar{w}(\pi^*)$. Each term in (8.24) is of the form

$$[w(1)]^j + [w(2)]^j + \cdots + [w(m)]^j = \sum_{r \in R} [w(r)]^j, \tag{8.27}$$

where $j = |D_i|$. Thus, a term (8.27) occurs in (8.24) as many times as $|D_i|$ equals j, that is, as many times as π has a cycle of length j. We denoted this as b_j in Section 8.5.5 when we defined the cycle index. Hence, $\bar{w}(\pi^*)$ or (8.24) can be rewritten as

$$\left[\sum_{r \in R} [w(r)]^1\right]^{b_1} \left[\sum_{r \in R} [w(r)]^2\right]^{b_2} \cdots.$$

Therefore, the right-hand side of (8.25) becomes

$$P_G\left(\sum_{r \in R} [w(r)]^1, \sum_{r \in R} [w(r)]^2, \ldots\right).$$

This proves Pólya's Theorem.

EXERCISES FOR SECTION 8.6

1. Find the weight of each coloring in column a in Figure 8.1 if $w(\text{black}) = 3$ and $w(\text{white}) = 4$.
2. If $w(1) = x$ and $w(2) = y$, find the weights of colorings f and g in parts (a) and (b) of Exercise 6, Section 8.4.
3. Suppose that K consists of the colorings C_2, C_8, C_{10}, and C_{14} of Figure 8.10. If $w(\text{black}) = b$ and $w(\text{white}) = w$, find the inventory of the collection K.
4. Suppose that K consists of the switching functions T_2, T_3, T_8, T_{10}, and T_{15} of Table 8.3. Find the inventory of K if $w(0) = a$ and $w(1) = b$.
5. In the situation of Exercise 9, Section 8.1, find the pattern inventory if $w(\text{black}) = b$ and $w(\text{white}) = w$.
6. In the situation of Exercise 7, Section 8.4, find the pattern inventory if $w(1) = \alpha$ and $w(2) = \beta$.
7. Use Pólya's Theorem to compute the number of distinct four-bead necklaces, where each bead has one of three colors.

8. In Example 8.4, suppose that we have four possible colors for the vertices. Use Pólya's Theorem to find the number of distinct colorings of the tree.

9. In Example 8.5, find the number of distinct molecules with no CH_3's.

10. How many four-bead necklaces are there in which each bead is one of the colors b, r, or p, and there is at least one p?

11. Consider colorings of $K_{1,3}$ with colors G, B, W.

    (a) Check that there are exactly 8 distinct colorings using no G, by showing the colorings.

    (b) Check that there are exactly 7 distinct colorings with exactly two W's, by showing the colorings.

    (c) Find the number of distinct colorings with exactly one W.

12. Find the number of distinct colorings of the following graphs, which were introduced in Section 8.2, Exercise 23, using the colors G, B, W and exactly one G.

    (a) L_4 (b) Z_4 (c) $K_4 - K_2$

13. Find the number of distinct colorings of the graphs in Exercise 12 using the colors G, B, W and exactly three B's.

14. Find the number of distinct switching functions of two variables that have at least one 1 in the range, that is, which assign 1 to at least one bit string.

15. In Exercise 23, Section 8.4, find the number of distinct switching functions of three variables that have at least one 1 in the range.

16. In Example 8.5, find the number of distinct molecules that have at least one Cl atom and at least one H atom.

17. Use Pólya's Theorem to compute the number of nonisomorphic graphs with:

    (a) Three vertices (b) Three vertices and two edges

    (c) Three vertices and at least one edge (d) Four vertices

    (e) Four vertices and three edges (f) Four vertices and at least two edges

    (See Exercises 21 and 22, Section 8.4.) For further applications of Pólya's Theorem to graph theory, see Chartrand and Lesniak [1996], Gross and Yellen [1999], Harary [1969], or Harary and Palmer [1973].

18. The vertices of a cube are to be colored and five colors are available: red, blue, green, yellow, and purple. Count the number of distinct colorings in which at least one green and one purple are used (see Exercises 22 and 23, Section 8.5).

19. Let $D = \{a, b, c, d\}$, $R = \{0, 1\}$, let G consist of the permutations $(1)(2)(3)(4)$, $(12)(34)$, $(13)(24)$, $(14)(23)$, and take $w(0) = 1, w(1) = x$.

    (a) Find $C(D, R)$. (b) Find G^*.

    (c) Find all equivalence classes of colorings under G^*.

    (d) Find the weights of all equivalence classes under G^*.

    (e) Let e_i be the number of colors of weight x^i and let $e(x)$ be the ordinary generating function of the e_i; that is, $e(x) = \sum_{i=0}^{\infty} e_i x^i$. Compute $e(x)$.

(f) Let E_j be the number of patterns of weight x^j and let $E(x)$ be the ordinary generating function of the E_j; that is, $E(x) = \sum_{j=0}^{\infty} E_j x^j$. Compute $E(x)$.

(g) Show that for $e(x)$ and $E(x)$ as computed in parts (e) and (f),

$$E(x) = P_G[e(x), e(x^2), e(x^3), \ldots]. \tag{8.28}$$

20. Repeat Exercise 19 for $D = \{a, b, c\}$, $R = \{0, 1\}$, G the set of permutations (1)(2)(3) and (12)(3), and $w(0) = x^2$, $w(1) = x^7$.

21. Generalizing the results of Exercises 19 and 20, suppose that for every r, $w(r) = x^p$ for some nonnegative integer p. Let $e(x)$ and $E(x)$ be defined as in Exercise 19. Show that, in general, (8.28) holds.

REFERENCES FOR CHAPTER 8

Bogart, K. P., *Introductory Combinatorics*, 3rd ed., Academic Press, San Diego, 1999.

Brualdi, R. A., *Introductory Combinatorics*, 3rd ed., Prentice Hall, Upper Saddle River, NJ, 1999.

Burnside, W., *Theory of Groups of Finite Order*, 2nd ed., Cambridge University Press, Cambridge, 1911. (Reprinted by Dover, New York, 1955.)

Cameron, P. J., "Automorphism Groups of Graphs," in L. W. Beineke and R. J. Wilson (eds.), *Selected Topics in Graph Theory, 2*, Academic Press, London, 1983, 89–127.

Chartrand, G., and Lesniak, L., *Graphs and Digraphs*, 3rd ed., CRC Press, Boca Raton, 1996.

de Bruijn, N. G., "Generalization of Pólya's Fundamental Theorem in Enumerative Combinatorial Analysis," *Ned. Akad. Wet., Proc. Ser. A 62, Indag. Math., 21* (1959), 59–79.

de Bruijn, N. G., "Pólya's Theory of Counting," in E. F. Beckenbach (ed.), *Applied Combinatorial Mathematics*, Wiley, New York, 1964, 144–184.

Gross, J., and Yellen, J., *Graph Theory and Its Applications*, CRC Press, Boca Raton, FL, 1999.

Harary, F., *Graph Theory*, Addison-Wesley, Reading, MA, 1969.

Harary, F., and Palmer, E. M., *Graphical Enumeration*, Academic Press, NY, 1973.

Harrison, M. A., *Introduction to Switching and Automata Theory*, McGraw-Hill, New York, 1965.

Liu, C. L., *Introduction to Combinatorial Mathematics*, McGraw-Hill, New York, 1968.

Pólya, G., "Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen," *Acta Math., 68* (1937), 145–254.

Pólya, G., and Read, R. C., *Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds*, Springer-Verlag, New York, 1987.

Prather, R. E., *Discrete Mathematical Structures for Computer Science*, Houghton Mifflin, Boston, 1976.

Redfield, J. H., "The Theory of Group-Reduced Distributions," *Amer. J. Math., 49* (1927), 433–455.

Reingold, E. M., Nievergelt, J., and Deo, N., *Combinatorial Algorithms: Theory and Practice*, Prentice Hall, Englewood Cliffs, NJ, 1977.

Stone, H. S., *Discrete Mathematical Structures and Their Applications*, Science Research Associates, Chicago, 1973.

Tucker, A. C., *Applied Combinatorics*, 3rd ed., Wiley, New York, 1995.

PART III. The Existence Problem

Chapter 9

Combinatorial Designs

9.1 BLOCK DESIGNS

In the history of attempts to perform scientifically sound experiments, combinatorics has played an important role. We have already encountered problems of experimental design in Section 1.1, where we discussed the design of an experiment to study the effects of different drugs, and used this design problem to introduce the notion of Latin squares. In this chapter we study the combinatorial questions that arose originally from issues in experimental design, and discuss the role of combinatorial analysis in the theory of experimental design. In Chapter 10 we apply combinatorial designs to the theory of error-correcting codes. Other applications, some of which we will touch upon in this chapter, include topics in cryptography, design of computer and communication networks, software testing, storage in disk arrays, signal processing, sports scheduling, "group testing" for defective items, designing chips for DNA probes, and clone screening in molecular biology. Some of these applications are described in Stinson [2003] and Colbourn, Dinitz, and Stinson [1999]. For general references on combinatorial designs, see Anderson [1990], Beth, Jungnickel, and Lenz [1999], Colbourn and Dinitz [1996], Dinitz and Stinson [1992], Hughes and Piper [1988], Lindner and Rodger [1997], Street and Street [1987], and Wallis [1988].

The theory of design of experiments came into being largely through the work of R. A. Fisher, F. Yates, and others, motivated by questions of design of careful field experiments in agriculture. Although the applicability of this theory is now very widespread, much of the terminology still bears the stamp of its origin.

We shall be concerned with experiments aimed at comparing effects of different *treatments* or *varieties*, e.g., different types of fertilizers, different doses of a drug, or different brands of shoes or tires. Each treatment is applied to a number of *experimental units* or *plots*. In agriculture, the experimental unit may be an area

Table 9.1: An Experimental Design for Testing Tread Wear[a]

		Car			
		A	B	C	D
Wheel position	Left front	1	2	3	4
	Right front	1	2	3	4
	Left rear	1	2	3	4
	Right rear	1	2	3	4

[a]The i, j entry is the brand of tire used in position i on car j.

in which a crop is grown. However, the experimental unit may be a human subject on a given day, a piece of animal tissue, or the site on an animal or plant where an injection or chemical treatment is applied, or it may in other cases be a machine used in a certain location for some purpose.

Certain experimental units are grouped together in *blocks*. These are usually chosen because they have some inherent features in common: for example, because they are all on the same human subject or all in the same horizontal row in a field or all on the skin of the same animal, or all on the same machine.

To be concrete, let us consider the problem of comparing the tread wear of four different brands of tires.[1] The treatments we are comparing are the different brands of tires. Clearly, individual tires of a given brand may differ. Hence, we certainly want to try out more than one tire of each brand. A particular tire is an experimental unit. Now suppose that the tires are to be tested under real driving conditions. Then we naturally group four tires or experimental units together, since a car used to test the tires takes four of them. The test cars define the blocks.

It is natural to try to let each brand of tire or treatment be used as often as any other. Suppose that each is used r times. Then we need $4r$ experimental units in all, since there are four treatments or tire brands. Since the experimental units are split into blocks of size 4, $4r$ must be divisible by 4. In this case, r could be any positive integer. If there were five brands of tires, we would need $5r$ experimental units in all, and then r could only be chosen to be an integer so that $5r$ is divisible by 4.

If we take r to be 4, we could have a very simple experimental design. Find four cars, say A, B, C, D, and place four tires of brand 1 on car A, four tires of brand 2 on car B, four tires of brand 3 on car C, and four tires of brand 4 on car D. This design is summarized in Table 9.1. This is clearly an unsatisfactory experimental design. Different cars (and different drivers) may lead to different amounts of tire wear, and the attempt to distinguish brands of tires as to wear will be confused by extraneous factors.

Much of the theory of experimental design has been directed at eliminating the

[1]Our treatment follows Hicks [1973].

Table 9.2: A Randomized Design for Testing Tread Wear[a]

		Car A	B	C	D
Wheel position	Left front	3	4	2	2
	Right front	1	1	4	4
	Left rear	3	4	1	3
	Right rear	2	3	2	1

[a]The i, j entry is the brand of tire used in position i on car j.

Table 9.3: A Complete Block Design for Testing Tread Wear[a]

		Car A	B	C	D
Wheel position	Left front	1	1	3	4
	Right front	2	3	4	2
	Left rear	3	2	1	1
	Right rear	4	4	2	3

[a]The i, j entry is the brand of tire used in position i on car j.

biasing or confusing effect caused by variations in particular experimental units. One often tries to eliminate the effect by randomizing and by assigning treatments to experimental units in a random way. For instance, we could start with four tires of each brand, and assign tires to each car completely at random. This might lead to a design such as the one shown in Table 9.2. Unfortunately, as the table shows, we could end up with a tire brand such as 4 never being used on a particular car such as A, or one brand such as 3 used several times in a particular car such as A. The results might still be biased by car effects. We can avoid this situation if we require that each treatment or brand be used in each block or car, and then make the assignment of tires to wheels of the car randomly. A major question in the theory of experimental design is what we have called in Chapter 1 the existence question. Here, we ask this question as follows: Does there exist a design in which there are four brands and four cars, each brand is used four times, and it is used at least once, equivalently exactly once, in each car? The answer is yes. Table 9.3 gives such a design.

The design in Table 9.3 still has some defects. The position of a tire on a car can affect its tread life. For instance, rear tires get different wear than front tires, and even the side of a car a tire is on could affect its tread life. If we wish also to eliminate the biasing effect of wheel position, we could require that each brand

Table 9.4: A Latin Square Design for Testing Tread Wear[a]

		Car A	Car B	Car C	Car D
Wheel position	Left front	1	2	3	4
	Right front	2	3	4	1
	Left rear	3	4	1	2
	Right rear	4	1	2	3

[a]The i, j entry is the brand of tire used in position i on car j.

or treatment be used exactly once on each car and also exactly once in each of the possible positions. Then we ask for an assignment of the numbers $1, 2, 3, 4$ in a 4×4 array with each number appearing exactly once in each row and in each column. That is, we ask for a *Latin square* (see Section 1.1). Table 9.4 shows such a design. Among all possible 4×4 Latin square designs, we might still want to pick the particular one to use randomly.

In some experiments, it may not be possible to apply all treatments to every block. For instance, if there were five brands of tires, we could use only four of them in each block. How would we design an experiment now? If each brand of tire is used r times, we have $5r$ tires in all to distribute into groups of four, so as we observed above, $5r$ must be divisible by 4. For example, r must be $4, 8, 12$, and so on. Note that we could not do the experiment with six cars; that is, there does not exist an experimental design using five brands and six cars, with each brand used the same number of times, and four (different) brands assigned to each car. For there are 24 tire locations in all, and $5r = 24$ is impossible. Suppose that we take $r = 4$. Then there are $5r = 20$ tire locations in all. If s is the number of cars, $4s$ should be 20, so s should be 5. One possible design is given in Table 9.5. Here there are four different brands of tires on each car, each brand is used exactly once in each position, and each brand is used the same number of times, 5. There are various additional requirements that we can place on such a design. We discuss some of them below.

Let us now introduce some general terminology. Suppose that P is a set of experimental units or plots, and V is a set of treatments or varieties. Certain subsets of P will be called *blocks*. Given P and V, a *block design* is defined by giving the collection of blocks and assigning to each experimental unit in P a treatment in V. Thus, corresponding to each block is a set (possibly with repetitions) of treatments. Speaking abstractly, we shall be able to disregard the experimental units and think of a block design as simply consisting of a set V of treatments and a collection of subsets of V (possibly with repetitions) called blocks. Thus, the block design corresponding to Table 9.2 has $V = \{1, 2, 3, 4\}$ and has the following blocks:

$$\{3, 1, 3, 2\}, \quad \{4, 1, 4, 3\}, \quad \{2, 4, 1, 2\}, \quad \{2, 4, 3, 1\}.$$

Table 9.5: An Incomplete Block Design for Testing Tread Wear[a]

		Car				
		A	B	C	D	E
	Left front	1	2	3	4	5
Wheel	Right front	2	3	4	5	1
position	Left rear	3	4	5	1	2
	Right rear	4	5	1	2	3

[a]The i, j entry is the brand of tire used in position i on car j.

If order counts, as in the case of Latin squares, we can think of the blocks as sequences rather than subsets. A block design is called *complete* if each block is all of V, and *incomplete* otherwise. Tables 9.3 and 9.4 define complete block designs, and Table 9.5 defines an incomplete block design. A block design is called *randomized* if elements within each block are ordered by some random device, such as a random number table or a computer program designed to pick out random permutations.

We study two types of block designs in this chapter, the complete designs that come from Latin squares and families of Latin squares, and the incomplete designs that are called balanced. We also relate experimental design to the study of the finite geometries known as finite projective planes. In Chapter 10, we apply our results about experimental design to the design of error-correcting codes.

EXERCISES FOR SECTION 9.1

1. Find a Latin square design for the tread wear experiment different from that given in Table 9.4.
2. Suppose that we wish to test the effects of six different allergy medicines. Each subject gets one medicine each day for a week.

   (a) What are the varieties, experimental units, and blocks?

   (b) What can you say about the number of subjects needed if an experiment gives each medicine to the same number of subjects?

   (c) Is there a Latin square design for this experiment?

   (d) Is there a design for this experiment in which each medicine is used the same number of times in a week? If so, what are the blocks?

   (e) Is there a design for this experiment in which each medicine is used on the same number of subjects?

   (f) Is there a design in which each subject gets each medicine the same number of times?

3. (a) Give (as subsets) the blocks of a design in which the varieties are $\{1, 2, 3, 4, 5, 6, 7\}$ and the blocks are 3-element subsets.

   (b) Repeat part (a) so that each variety appears in exactly 3 blocks.

4. Give (as subsets) the blocks of a design in which the varieties are $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, the blocks are 3-element subsets, and each variety appears in exactly 4 blocks.

5. (a) Does there exist a design in which there are 8 varieties, blocks of size 4, and each variety appears in only 1 block? If so, give (as subsets) the blocks of the design. Otherwise, explain why a design doesn't exist.

   (b) Repeat part (a) with 12 varieties, blocks of size 5, and each variety appears in only 1 block.

   (c) Repeat part (a) with 7 varieties, blocks of size 3, and each variety appears in exactly 3 blocks.

   (d) Repeat part (a) with 10 varieties, blocks of size 4, and each variety appears in exactly 2 blocks.

6. (a) For a design in which the varieties are $\{1,2,3,4,5,6,7,8\}$ and the blocks are all of the 3-element subsets, in how many different blocks does each variety appear?

   (b) In general, in how many blocks does each variety appear if there are v varieties and the blocks are all of the k-element subsets?

7. Suppose that we have a block design in which each of the v varieties appears in r blocks and the blocks are k-element subsets. Consider an associated block design in which the blocks are the complements of the original blocks. Describe this new block design in terms of the number of varieties and blocks, the size of each block, and the number of blocks in which each variety appears.

9.2 LATIN SQUARES

9.2.1 Some Examples

A Latin square design is an appropriate experimental design if there are two factors, e.g., subject and day, wheel position and car, or *row* and *column*, and we want to control for both factors. In agricultural experiments, the rows and columns are literally rows and columns in a rectangular field. Latin squares were introduced by Fisher [1926] to deal with such experiments. Suppose, for example, that there are k different row effects and k different column effects, and we wish to test k different treatments. We wish to arrange things so that each treatment appears once and only once in a given row and in a given column, for example, in a given position and on a given car. Clearly, there is such an arrangement or $k \times k$ Latin square for every k. Table 9.6 shows a $k \times k$ Latin square. Thus, for Latin squares, the existence problem is very simple. The existence problem will not be so simple for the other designs we consider in this chapter.

We now turn to a series of examples of the use of Latin square designs.

Table 9.6: A $k \times k$ Latin Square

1	2	3	$\cdots$	$k-1$	k
2	3	4	$\cdots$	k	1
3	4	5	$\cdots$	1	2
			$\vdots$		
$k-1$	k	1	$\cdots$	$k-3$	$k-2$
k	1	2	$\cdots$	$k-2$	$k-1$

Example 9.1 Prosthodontics Cox [1958][2] discusses an experiment in prosthodontics that compares seven treatments, which are commercial dentures of different materials and set at different angles. It is desirable to eliminate as much as possible of the variation due to differences between patients. Hence, each patient wears dentures of one type for a month, then dentures of another type for another month, and so on. After seven months, each patient has worn each type of denture, that is, has been subjected to each treatment.

In this experiment, it seems likely that the results in later months will be different from those in earlier months, and hence it is sensible to arrange that each treatment be used equally often in each time position. Thus, there are two types of variation: between-patient and between-time variation. The desire to balance out both types suggests the use of a Latin square. The rows correspond to the months and the columns to the patients. Each patient defines a block, and the experimental unit is the jth patient in the ith month. ■

Example 9.2 Cardiac Drugs Chen, Bliss, and Robbins [1942] tested the effects of 12 different cardiac drugs on cats. The experiment required an observer to measure carefully the effect over a period of time, so a given observer could observe only four different cats in a day. The experimenters desired to eliminate the effects of the day on which an observation was made, the observer who made the observation, and the time of day (early AM, late AM, early PM, late PM) the observation was made. Thus, there were three factors, which is inappropriate for a Latin square design. However, a Latin square design could be carried out by taking as one factor the day on which the observation was made and as a second factor the observer and the time of day of the observation. A 12×12 Latin square experiment was performed, carried out over 12 days, with each of three observers observing four cats per day, two in the morning and two in the afternoon. The design used had 12 rows, coded by observer and time of observation, and 12 columns, coded by date. The i, j entry was the drug used on date j at the time of day and by the observer encoded by i. The dates defined the blocks. ■

[2] Examples 9.2, 9.3, 9.5, 9.13, and 9.14 below are also discussed by Cox [1958]. These and other examples can also be found in Box, Hunter, and Hunter [1978], Cochran and Cox [1957], Finney [1960], or Hicks [1973].

Table 9.7: Two Latin Square Designs for the Two Parts of the Week in the Market Research Experiment[a]

First Part of the Week

		Time			
		Mon.	Tues.	Wed.	Thurs.
Store	A	2	1	4	3
	B	3	2	1	4
	C	4	3	2	1
	D	1	4	3	2

Second Part of the Week

	Time			
	Fri. AM	Fri. PM	Sat. AM	Sat. PM
A	2	3	1	4
B	1	4	2	3
C	3	2	4	1
D	4	1	3	2

[a]The i, j entry gives the treatment used in store i in period j.

Example 9.3 Market Research Brunk and Federer [1953] discuss some investigations in market research. One of these studied the effect on the sale of apples of varying practices of pricing, displaying, and packaging. In each experiment of a series, four merchandising practices (treatments), $1, 2, 3$, and 4, were compared and four supermarkets took part. It was clearly desirable that each treatment should be used in each store, so it was sensible to arrange for the experiment to continue for a multiple of four time periods. The experimenters wanted to eliminate systematic differences between stores and between periods. Since there were two types of variations, a Latin square design, in particular a 4×4 Latin square, was appropriate. In fact, however, the week was divided into two parts, Monday through Thursday, and Friday and Saturday, and one 4×4 Latin square was built up for each part of the week. This was a good idea because the grocery order per customer was larger over the weekend and it was quite possible that the treatment differences would not be the same in the two parts of the experiment. For an experiment lasting one week and comparing four treatments, the design of Table 9.7 was used. ■

Example 9.4 Spinning Synthetic Yarn Box, Hunter, and Hunter [1978] discuss an experiment dealing with the breaking strength of synthetic yarn and how this is affected by changes in draw ratio, the tension applied to yarn as it is spun. The three treatments tested were (1) the usual draw ratio, (2) a 5 percent increase in draw ratio, and (3) a 10 percent increase in draw ratio. One spinning machine was used, with three different spinnerets supplying yarn to three different bobbins under different draw ratios. When all the bobbins were completely wound with yarn, they were each replaced with an empty bobbin spool and the experiment was continued. The experimenter wished to control for two factors: the effect of the three different spinnerets and the effect of the time (order) in which the spinnerets were used. This called for a 3×3 Latin square design, with columns labeled I, II, III corresponding to order of production of the yarn, and rows labeled A, B, C corresponding to which spinneret was used. The i, j entry was the treatment or draw ratio $(1, 2$, or $3)$ used in producing yarn from the ith spinneret in the jth production

Table 9.8: Latin Square Designs for the Synthetic Yarn Experiment[a]

Spinneret	First replication			Second replication			Third replication			Fourth replication		
Order of production	I	II	III	I	II	III	I	II	III	I	II	III
A	1	2	3	2	1	3	3	1	2	1	2	3
B	2	3	1	3	2	1	1	2	3	2	3	1
C	3	1	2	1	3	2	2	3	1	3	1	2

[a]The i, j entry is the draw ratio used with the ith spinneret in the jth production run.

run. When small Latin squares are used, it is often desirable to replicate them, so in fact the experiment was replicated four times, using different 3×3 Latin square designs. Table 9.8 shows the designs. ■

9.2.2 Orthogonal Latin Squares

Let us return to the example of the differing effects on tire wear of four tire brands, which we discussed in Section 9.1. Let us imagine that we are also interested in the effect of brake linings on tire wear. Suppose for simplicity that we also have four different brands of brake linings. Thus, we would like to arrange, in addition to having each brand of tire tested exactly once on each car and exactly once in each tire position, that each tire brand be tested exactly once in combination with each brand of brake lining. We can accomplish this by building a 4×4 array, with rows corresponding to wheel position and columns to cars, and placing in each box *both* a tire brand and a brake lining brand to be used in the corresponding position and on the corresponding car. If a_{ij} is the tire brand in entry i, j of the array and b_{ij} is the brake lining brand in this entry, we require that every possible ordered pair (a, b) of tire brands a and brake lining brands b appear if we list all ordered pairs (a_{ij}, b_{ij}). Equivalently, since there are $4 \times 4 = 16$ possible ordered pairs (a, b) and exactly 16 spots in the array, we require that all the pairs (a_{ij}, b_{ij}) be different. Can we accomplish this? We certainly can. If the brake lining brands are denoted $1, 2,$ 3,4, simply test brake lining brand i on every wheel of the ith car. Combining this design with the tire brand design of Table 9.4 gives us the array of ordered pairs of Table 9.9. All the ordered pairs in this table are different.

Unfortunately, the array of Table 9.9 is not a very satisfactory design if we consider just brake linings. For we only use brake linings of brand 1 on car A, of brand 2 on car B, and so on. It would be good to have the brake linings tested by a Latin square design, not just the tires. Thus, we would like to find two Latin square

Table 9.9: Design for Testing the Combined Effects of Tire Brand and Brake Lining Brand on Tread Wear[a]

		Car			
		A	B	C	D
Wheel position	Left front	(1, 1)	(2, 2)	(3, 3)	(4, 4)
	Right front	(2, 1)	(3, 2)	(4, 3)	(1, 4)
	Left rear	(3, 1)	(4, 2)	(1, 3)	(2, 4)
	Right rear	(4, 1)	(1, 2)	(2, 3)	(3, 4)

[a]The i, j entry is an ordered pair, giving first the tire brand used in position i on car j, and then the brake lining brand used there.

experiments, $A = (a_{ij})$ and $B = (b_{ij})$, one for tire brands and one for brake lining brands, both using the same row and column effects. Moreover, we want the ordered pairs (a_{ij}, b_{ij}) all to be different. Can this be done? In our case, it can. Table 9.10 shows a pair of Latin square designs and the corresponding array of ordered pairs, which is easily seen to have each ordered pair (a, b), with $1 \leq a \leq 4$ and $1 \leq b \leq 4$, appearing exactly once. Equivalently, the ordered pairs are all different. We shall say that two distinct $n \times n$ Latin squares $A = (a_{ij})$ and $B = (b_{ij})$ are *orthogonal* if the n^2 ordered pairs (a_{ij}, b_{ij}) are all distinct. Thus, the two 4×4 Latin squares of Table 9.10 are orthogonal. However, the two Latin squares of Table 9.7 are not, as the ordered pair $(2, 4)$ appears twice, once in the $2, 2$ position and once in the $3, 3$ position. More generally, if $A^{(1)}, A^{(2)}, \ldots, A^{(r)}$ are distinct $n \times n$ Latin squares, they are said to form an *orthogonal family* if every pair of them is orthogonal.

The main question we address in this section is the fundamental existence question: If we want to design an experiment using a pair of $n \times n$ orthogonal Latin squares, can we always be sure that such a pair exists? More generally, we shall ask: When does an orthogonal family of r different $n \times n$ Latin squares exist?

Before addressing these questions, we give several examples of the use of orthogonal Latin square designs.

Example 9.5 Fuel Economy Davies [1945] used a pair of orthogonal Latin squares in the comparison of fuel economy in miles per gallon achieved with different grades of gasoline. Seven grades of gasoline were tested. One car was used throughout. Each test involved driving the test car over a fixed route of 20 miles, including various gradients. To remove possible biases connected with the driver, seven drivers were used; and to remove possible effects connected with the traffic conditions, the experiment was run on different days and at seven different times of the day. Thus, in addition to the seven treatments under comparison, there are three classifications of the experimental units: by drivers, by days, and by times of the day. A double classification of the experimental units suggests the use of a Latin square, a triple classification a pair of orthogonal Latin squares. The latter allows for an experiment in which each grade of gasoline is used once on each day, once by

Table 9.10: Two Orthogonal Latin Square Designs for Testing the Combined Effects of Tire Brand and Brake Lining Brand on Tread Wear[a]

		Car A	Car B	Car C	Car D
Wheel position	Left front	1	2	3	4
	Right front	2	1	4	3
	Left rear	3	4	1	2
	Right rear	4	3	2	1

	Car A	Car B	Car C	Car D
Left front	4	1	2	3
Right front	3	2	1	4
Left rear	1	4	3	2
Right rear	2	3	4	1

		Car A	Car B	Car C	Car D
Wheel position	Left front	(1, 4)	(2, 1)	(3, 2)	(4, 3)
	Right front	(2, 3)	(1, 2)	(4, 1)	(3, 4)
	Left rear	(3, 1)	(4, 4)	(1, 3)	(2, 2)
	Right rear	(4, 2)	(3, 3)	(2, 4)	(1, 1)

Combined Design

[a]The combined array lists in the i, j entry the ordered pair consisting of the tire brand and then the brake lining brand used in the two Latin squares in tire position i on car j.

each driver, and once at each time of day, ensuring a balanced comparison. The design assigns to each day (row) and each time of day (column) one grade of gasoline (in the first Latin square) and one driver (in the second square). (In our tire wear example of Section 9.1, we could not control for the driver in the same way, that is, it would not make sense to use a pair of orthogonal Latin square experiments, the first indicating tire brand at position i on car j and the second indicating driver at position i on car j. For the same driver must be assigned to all positions of a given car!) ■

Example 9.6 Testing Cloth for Wear Box, Hunter, and Hunter [1978] describe an experiment involving a Martindale wear tester, a machine used to test the wearing quality of materials such as cloth. In one run of a Martindale wear tester of the type considered, four pieces of cloth could be rubbed simultaneously, each against a sheet of emery paper, and then the weight loss could be measured. There were four different specimen holders, labeled A, B, C, D, and each could be used in one of four positions, P_1, P_2, P_3, P_4, on the machine. In a particular experiment, four types of cloth or treatments, labeled $1, 2, 3, 4$, were compared. The experimenters wanted to control for the effects of the four different specimen holders, the four positions of the machine, which run the cloth was tested in, and which sheet of emery paper the cloth was rubbed against. A quadruple classification of experimental units suggests an orthogonal family of three 4×4 Latin squares. It was decided to use four sheets of emery paper, labeled $\alpha, \beta, \gamma, \delta$, to cut each into four quarters, and to use each quarter in one experimental unit. There were four runs

Table 9.11: An Orthogonal Family of Three Latin Squares for Testing Cloth for Wear[a]

Position (Run)	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1	1	3	4	2	A	D	B	C	α	β	γ	δ
P_2	2	4	3	1	B	C	A	D	β	α	δ	γ
P_3	3	1	2	4	C	B	D	A	γ	δ	α	β
P_4	4	2	1	3	D	A	C	B	δ	γ	β	α
	Latin Square Design for Treatments				Latin Square Design for Holders				Latin Square Design for Emery Sheets			

[a]The i, j entry shows the treatment (cloth type), holder, and emery paper sheet, respectively, used in run R_j in position P_i.

in all, R_1, R_2, R_3, R_4, each testing four specimens of cloth with different holders in varying positions and with different quarter-pieces of emery paper. Table 9.11 shows the three Latin square designs used. The reader can check that these are pairwise orthogonal. (In fact, the experiment was replicated, using four more runs and four more sheets of emery paper, again in a design involving an orthogonal family of three 4×4 Latin squares.) ■

9.2.3 Existence Results for Orthogonal Families

Let the *order* of an $n \times n$ Latin square be n. In what follows we usually assume that the entries in a Latin square of order n are the integers $1, 2, \ldots, n$. We now discuss the question: Does there exist an orthogonal family of r Latin squares of order n? We shall assume that $n > 1$, for there is only one 1×1 Latin square. There does not exist a pair of orthogonal 2×2 Latin squares. For the only Latin squares of order 2 are shown in Table 9.12. They are not orthogonal since the pair $(1, 2)$ appears twice. We have seen in Table 9.10 that there is a pair of orthogonal Latin squares of order 4, and in Table 9.11 that there is an orthogonal family of three Latin squares of order 4. It is easy enough to give a pair of orthogonal Latin squares of order 3. (Try it.)

The first theorem gives necessary conditions for the existence of an orthogonal family of r Latin squares of order n.

Theorem 9.1 If there is an orthogonal family of r Latin squares of order n, then $r \leq n - 1$.

Proof. Suppose that $A^{(1)}, A^{(2)}, \ldots, A^{(r)}$ form an orthogonal family of $n \times n$ Latin squares. Let $a_{ij}^{(p)}$ be the i, j entry of $A^{(p)}$. Relabel the entries in the first square so that 1 comes in the 1, 1 spot, that is, so that $a_{11}^{(1)} = 1$. Do this as follows. If $a_{11}^{(1)}$

Table 9.12: The Two Latin Squares of Order 2

1	2		2	1
2	1		1	2

was k, switch 1 with k and k with 1 throughout $A^{(1)}$. This does not change $A^{(1)}$ from being a Latin square and it does not change the orthogonality, for if the pair

$$\left(a_{ij}^{(1)}, a_{ij}^{(p)}\right)$$

was (k, l), it is now $(1, l)$, and if it was $(1, l)$, it is now (k, l).

By the same reasoning, without affecting the fact that we have an orthogonal family of $n \times n$ Latin squares, we can arrange matters so that each $1, 1$ entry in each square is 1, and more generally so that

$$\begin{aligned} a_{11}^{(1)} &= a_{11}^{(2)} = \cdots = a_{11}^{(r)} = 1, \\ a_{12}^{(1)} &= a_{12}^{(2)} = \cdots = a_{12}^{(r)} = 2, \\ a_{13}^{(1)} &= a_{13}^{(2)} = \cdots = a_{13}^{(r)} = 3, \\ &\vdots \\ a_{1n}^{(1)} &= a_{1n}^{(2)} = \cdots = a_{1n}^{(r)} = n. \end{aligned}$$

That is, we can arrange matters so that each $A^{(p)}$ has the same first row:

$$1 \quad 2 \quad 3 \quad \cdots \quad n.$$

Let us consider the $2, 1$ entry in each square. Since $A^{(p)}$ is a Latin square, and since $a_{11}^{(p)}$ is 1 and 1 can appear only once in a column, $a_{21}^{(p)}$ must be different from 1. Moreover, by orthogonality,

$$a_{21}^{(p)} \neq a_{21}^{(q)}$$

if $p \neq q$. For otherwise,

$$\left(a_{21}^{(p)}, a_{21}^{(q)}\right) = (i, i)$$

for some i, so

$$\left(a_{21}^{(p)}, a_{21}^{(q)}\right) = \left(a_{1i}^{(p)}, a_{1i}^{(q)}\right),$$

which violates orthogonality. Thus, the numbers

$$a_{21}^{(1)}, a_{21}^{(2)}, \ldots, a_{21}^{(r)}$$

are all different and all different from 1. It follows that there are at most $n - 1$ of these numbers, and $r \leq n - 1$. (Formally, this reasoning uses the pigeonhole principle of Section 2.19.) Q.E.D.

Table 9.13: The Procedure for Changing an Orthogonal Family of Latin Squares into One Where Each Square Has First Row $123\cdots n$

$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
4 3 2 1	2 1 4 3	3 4 1 2
3 4 1 2	4 3 2 1	2 1 4 3
2 1 4 3	3 4 1 2	4 3 2 1
1 2 3 4	1 2 3 4	1 2 3 4
↓ Interchange 1 with 4 ↓	↓ Interchange 1 with 2 ↓	↓ Interchange 1 with 3 ↓
1 3 2 4	1 2 4 3	1 4 3 2
3 1 4 2	4 3 1 2	2 3 4 1
2 4 1 3	3 4 2 1	4 1 2 3
4 2 3 1	2 1 3 4	3 2 1 4
↓ Interchange 2 with 3 ↓	↓ Interchange 3 with 4 ↓	↓ Interchange 2 with 4 ↓
1 2 3 4	1 2 3 4	1 2 3 4
2 1 4 3	3 4 1 2	4 3 2 1
3 4 1 2	4 3 2 1	2 1 4 3
4 3 2 1	2 1 4 3	3 4 1 2

We illustrate the proof of this theorem by starting with an orthogonal family of three Latin squares of order 4 as shown in Table 9.13. The procedure to arrange that all the first rows are 1234 is illustrated in the table. Note that the $2, 1$ entries in the three Latin squares in the last row of Table 9.13 are $2, 3$, and 4, respectively.

Theorem 9.1 says that we can never find an orthogonal family of $n \times n$ Latin squares consisting of more than $n-1$ squares. Let us say that an orthogonal family of Latin squares of order n is *complete* if it has $n-1$ Latin squares in it. Thus, the three Latin squares of order 4 shown in Table 9.13 form a complete orthogonal family. It will be convenient to think of a single 2×2 Latin square as constituting an orthogonal family.

Theorem 9.2 gives sufficient conditions for the existence of a complete orthogonal family of Latin squares.

Theorem 9.2 If $n > 1$ and $n = p^k$, where p is a prime number[3] and k is a positive integer, then there is a complete orthogonal family of Latin squares of order n.

We omit a proof of Theorem 9.2 at this point. We prove it in Sections 9.3.4 and 9.3.5 by describing a constructive procedure for finding a complete orthogonal family of Latin squares of order n if n is a power of a prime. In particular, Theorem 9.2

[3] Recall that a *prime number* n is an integer greater than 1 whose only divisors are 1 and n. See Section 7.1.3.

says that there exists a pair of orthogonal 3×3 Latin squares, since $3 = 3^1$. It also says that there exist three pairwise orthogonal 4×4 Latin squares, since $4 = 2^2$. (We have already seen three such squares in Tables 9.11 and 9.13.) There also exists a family of four pairwise orthogonal 5×5 Latin squares, since $5 = 5^1$. Since 6 is not a power of a prime, Theorem 9.2 does not tell us whether or not there is a set of five pairwise orthogonal 6×6 Latin squares, or indeed whether there is even a pair of such squares. We shall show below that there is no complete orthogonal family of Latin squares of order n, indeed, not even a pair of such squares, if $n = 6$. Thus, for $n \leq 9$, there is a complete orthogonal family of Latin squares of order n if and only if $n \neq 6$. Lam, Thiel, and Swiercz [1989], by a massive computer search, found that there is not a complete orthogonal family of Latin squares of order 10. There is one if $n = 11$. This leaves $n = 12$ as the smallest number for which we don't know if there is a complete orthogonal family. As of this writing, the best we know is that there can be 5 pairwise orthogonal Latin squares of order 12. (See Johnson, Dulmage, and Mendelsohn [1961].)

According to the Fundamental Theorem of Algebra, any integer $n > 1$ can be written *uniquely* as the product of (integer) powers of primes:

$$p_1^{t_1} p_2^{t_2} \cdots p_s^{t_s} \tag{9.1}$$

This product is called the *prime power decomposition.* For example,

$$\begin{aligned} 6 &= 2^1 3^1, \\ 12 &= 3 \times 4 = 3^1 2^2, \\ 80 &= 16 \times 5 = 2^4 5^1, \\ 60 &= 4 \times 15 = 4 \times 3 \times 5 = 2^2 3^1 5^1. \end{aligned}$$

Theorem 9.3 Suppose that $n = p_1^{t_1} p_2^{t_2} \cdots p_s^{t_s}$ is the prime power decomposition of n, $n > 1$, and r is the smallest of the quantities

$$(p_1^{t_1} - 1), (p_2^{t_2} - 1), \ldots, (p_s^{t_s} - 1).$$

Then there is an orthogonal family of r Latin squares of order n.

We prove this theorem below. To illustrate it, we recall that $12 = 2^2 3^1$. Then

$$2^2 - 1 = 3, \quad 3^1 - 1 = 2,$$

so $r = 2$. It follows that there exist two orthogonal Latin squares of order 12. This does not say that there is not a larger orthogonal family of 12×12's. Note that Theorem 9.2 does not apply, since 12 is not a power of a prime.

Let us try to apply Theorem 9.3 to $n = 6$. We have $6 = 2^1 3^1$. Since

$$2^1 - 1 = 1, \quad 3^1 - 1 = 2,$$

$r = 1$, and we do not even know from Theorem 9.3 if there exists a pair of orthogonal 6×6 Latin squares. The famous mathematician Euler conjectured in 1782 that there was no such pair. For more than 100 years, the conjecture could neither be proved nor disproved. Around 1900, Tarry looked systematically at all possible pairs of 6×6 Latin squares. (There are 812,851,200; but by making the first row 123456, it is only necessary to consider 9408 pairs.) He succeeded in proving that Euler was right. Thus, there does not exist a pair of orthogonal Latin squares of order 6 (see Tarry [1900, 1901]; see Stinson [1984] for a modern proof that doesn't just consider all cases).

The following is a corollary of Theorem 9.3.

Corollary 9.3.1 Suppose that $n > 1$ and either 2 does not divide n or the prime power decomposition of n is

$$n = 2^{t_1} p_2^{t_2} p_3^{t_3} \cdots,$$

with $t_1 > 1$. Then there is a pair of orthogonal Latin squares.

Proof. If $t_1 > 1$,

$$2^{t_1} - 1 \geq 3.$$

Each other p_i is greater than 2, so

$$p_i^{t_i} - 1 \geq 2.$$

It follows that $r \geq 2$. Q.E.D.

Corollary 9.3.1 leaves open the question of the existence of pairs of orthogonal Latin squares of order $n = 2k$ where 2 does not divide k. Euler, also in 1782, conjectured that there does not exist a pair of orthogonal Latin squares of order n for all such n. He was right for $n = 2$ and $n = 6$. However, contrary to his usual performance, he was wrong otherwise. It was not until 1960 that he was proved wrong.

Theorem 9.4 (Bose, Shrikhande, and Parker [1960]) If $n > 6$, $n = 2k$, and 2 does not divide k, then there is a pair of orthogonal Latin squares of order n.

We can now summarize what we know about the existence of pairs of orthogonal Latin squares.

Theorem 9.5 There is a pair of orthogonal Latin squares of order n for all $n > 1$ except $n = 2$ or 6.

Thus, the existence of pairs of orthogonal Latin squares is completely settled. This is not the case for larger families of orthogonal Latin squares. For $n = 2, 3, \ldots, 9$, the size of the largest orthogonal family of $n \times n$ Latin squares is known. For by Theorems 9.2 and 9.5, it is $n - 1$ for $n = 3, 4, 5, 7, 8, 9$, and it is 1 for $n = 2, 6$. However, for $n = 10$, it is not even known if there is a family of three pairwise orthogonal $n \times n$ Latin squares.

Example 9.7 The Problem of the 36 Officers Euler encountered the notion of orthogonal Latin squares not in connection with experimental design, but in connection with the following problem. There are 36 officers, six officers of six different ranks in each of six regiments. Find an arrangement of the 36 officers in a 6×6 square formation such that each row and each column contains one and only one officer of each rank and one and only one officer from each regiment and there is only one officer from each regiment of each rank. Can this be done? The officers must be arranged so that their ranks form a Latin square and also so that their regiments form a Latin square. Moreover, the pairs of rank and regiment appear once and only once, so the two squares must be orthogonal. We now know that this cannot be done. ■

9.2.4 Proof of Theorem 9.3[4]

To prove Theorem 9.3, we first prove the following result.

Theorem 9.6 (MacNeish [1922]) Suppose that there is an orthogonal family of r Latin squares of order m and another orthogonal family of r Latin squares of order n. Then there is an orthogonal family of r Latin squares of order mn.

Proof. Let $A^{(1)}, A^{(2)}, \ldots, A^{(r)}$ be pairwise orthogonal Latin squares of order m and $B^{(1)}, B^{(2)}, \ldots, B^{(r)}$ be pairwise orthogonal Latin squares of order n. For $e = 1, 2, \ldots, r$, let $(a_{ij}^{(e)}, B^{(e)})$ represent the $n \times n$ matrix whose u, v entry is the ordered pair $(a_{ij}^{(e)}, b_{uv}^{(e)})$. For instance, suppose that $A^{(1)}, A^{(2)}, B^{(1)}$, and $B^{(2)}$ are as in Table 9.14. Then $(a_{12}^{(1)}, B^{(1)})$ and $(a_{32}^{(2)}, B^{(2)})$ are as shown in the table. Let $C^{(e)}$ be the matrix that can be represented schematically as follows:

$$C^{(e)} = \begin{bmatrix} (a_{11}^{(e)}, B^{(e)}) & (a_{12}^{(e)}, B^{(e)}) & \cdots & (a_{1m}^{(e)}, B^{(e)}) \\ (a_{21}^{(e)}, B^{(e)}) & (a_{22}^{(e)}, B^{(e)}) & \cdots & (a_{2m}^{(e)}, B^{(e)}) \\ & & \cdots & \\ (a_{m1}^{(e)}, B^{(e)}) & (a_{m2}^{(e)}, B^{(e)}) & \cdots & (a_{mm}^{(e)}, B^{(e)}) \end{bmatrix}.$$

Then $C^{(e)}$ is an $mn \times mn$ matrix. We shall show that $C^{(1)}, C^{(2)}, \ldots, C^{(r)}$ is an orthogonal family of Latin squares of order mn.

To see that $C^{(e)}$ is a Latin square, note first that in a given row, two entries in different columns are given by $(a_{ij}^{(e)}, b_{uv}^{(e)})$ and $(a_{ik}^{(e)}, b_{uw}^{(e)})$; so they are distinct since $A^{(e)}$ and $B^{(e)}$ are Latin squares. In a given column, two entries in different rows are given by $(a_{ij}^{(e)}, b_{uv}^{(e)})$ and $(a_{kj}^{(e)}, b_{wv}^{(e)})$; so again they are distinct because $A^{(e)}$ and $B^{(e)}$ are Latin squares.

To see that $C^{(e)}$ and $C^{(f)}$ are orthogonal, suppose that

$$\langle (a_{ij}^{(e)}, b_{uv}^{(e)}), (a_{ij}^{(f)}, b_{uv}^{(f)}) \rangle = \langle (a_{pq}^{(e)}, b_{st}^{(e)}), (a_{pq}^{(f)}, b_{st}^{(f)}) \rangle.$$

[4]This subsection may be omitted.

Table 9.14: Orthogonal Latin Squares $A^{(1)}$, $A^{(2)}$, $B^{(1)}$, $B^{(2)}$; Matrices $\left(a_{12}^{(1)}, B^{(1)}\right)$, $\left(a_{32}^{(2)}, B^{(2)}\right)$

$$A^{(1)} = \begin{array}{|ccc|} \hline 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \\ \hline \end{array}\,, \qquad A^{(2)} = \begin{array}{|ccc|} \hline 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \\ \hline \end{array}$$

$$B^{(1)} = \begin{array}{|cccc|} \hline 4 & 3 & 2 & 1 \\ 3 & 4 & 1 & 2 \\ 2 & 1 & 4 & 3 \\ 1 & 2 & 3 & 4 \\ \hline \end{array}\,, \qquad B^{(2)} = \begin{array}{|cccc|} \hline 2 & 1 & 4 & 3 \\ 4 & 3 & 2 & 1 \\ 3 & 4 & 1 & 2 \\ 1 & 2 & 3 & 4 \\ \hline \end{array}$$

$$\left(a_{12}^{(1)}, B^{(1)}\right) = \begin{bmatrix} (2,4) & (2,3) & (2,2) & (2,1) \\ (2,3) & (2,4) & (2,1) & (2,2) \\ (2,2) & (2,1) & (2,4) & (2,3) \\ (2,1) & (2,2) & (2,3) & (2,4) \end{bmatrix}, \quad \left(a_{32}^{(2)}, B^{(2)}\right) = \begin{bmatrix} (3,2) & (3,1) & (3,4) & (3,3) \\ (3,4) & (3,3) & (3,2) & (3,1) \\ (3,3) & (3,4) & (3,1) & (3,2) \\ (3,1) & (3,2) & (3,3) & (3,4) \end{bmatrix}$$

Then it follows that

$$(a_{ij}^{(e)}, a_{ij}^{(f)}) = (a_{pq}^{(e)}, a_{pq}^{(f)}),$$

so by orthogonality of $A^{(e)}$ and $A^{(f)}$, $i = p$ and $j = q$. Similarly, orthogonality of $B^{(e)}$ and $B^{(f)}$ implies that $u = s$ and $v = t$. Q.E.D.

Proof of Theorem 9.3. By Theorem 9.2, for $i = 1, 2, \ldots, s$, there is an orthogonal family of $p_i^{t_i} - 1$ Latin squares of order $p_i^{t_i}$. Thus, for $i = 1, 2, \ldots, s$, there is an orthogonal family of r Latin squares of order $p_i^{t_i}$. The result follows from Theorem 9.6 by mathematical induction on s. Q.E.D.

9.2.5 Orthogonal Arrays with Applications to Cryptography[5]

Suppose that V is a set of n elements and $k \geq 2$ is an integer. An *orthogonal array* $\mathrm{OA}(k, n)$ is an $n^2 \times k$ matrix A whose entries are elements of V and so that within any two columns of A, every ordered pair (a, b) with $a, b \in V$ occurs in exactly one row of A. Table 9.15 shows an $\mathrm{OA}(4, 3)$ with $V = \{1, 2, 3\}$ and an $\mathrm{OA}(5, 4)$ with $V = \{1, 2, 3, 4\}$.

There is a very simple relationship between orthogonal arrays and families of orthogonal Latin squares.

Theorem 9.7 There is an $\mathrm{OA}(k, n)$ if and only if there is a family of $k - 2$ orthogonal Latin squares of order n.

[5] This subsection depends heavily on Stinson [2003].

Table 9.15: An OA(4, 3) and an OA(5, 4)

$$\begin{bmatrix} 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 \\ 2 & 3 & 3 & 3 \\ 1 & 2 & 1 & 3 \\ 1 & 1 & 3 & 2 \\ 1 & 3 & 2 & 1 \\ 3 & 2 & 3 & 1 \\ 3 & 1 & 2 & 3 \\ 3 & 3 & 1 & 2 \end{bmatrix}. \quad \begin{bmatrix} 1 & 4 & 4 & 4 & 4 \\ 2 & 4 & 3 & 2 & 1 \\ 2 & 3 & 4 & 1 & 2 \\ 3 & 3 & 1 & 2 & 4 \\ 4 & 4 & 1 & 3 & 2 \\ 4 & 3 & 2 & 4 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 2 & 3 & 4 \\ 1 & 3 & 3 & 3 & 3 \\ 4 & 1 & 4 & 2 & 3 \\ 3 & 4 & 2 & 1 & 3 \\ 4 & 2 & 3 & 1 & 4 \\ 1 & 2 & 2 & 2 & 2 \\ 3 & 1 & 3 & 4 & 2 \\ 2 & 2 & 1 & 4 & 3 \\ 3 & 2 & 4 & 3 & 1 \end{bmatrix}.$$

Proof. Suppose, without loss of generality, that entries in the orthogonal arrays and Latin squares come from the set $V = \{1, 2, \ldots, n\}$. Let $A^{(1)}, A^{(2)}, \ldots, A^{(k-2)}$ denote the $k-2$ orthogonal Latin squares of order n. Form the matrix A whose rows are the n^2 sequences

$$i, j, a_{ij}^{(1)}, a_{ij}^{(2)}, \ldots, a_{ij}^{(k-2)},$$

where $i,\ j \in \{1, 2, \ldots, n\}$ and $a_{ij}^{(l)}$ or $A^{(l)}(i, j)$ is the i, j entry of the matrix $A^{(l)}$. To see that A is an OA(k, n), consider columns u and v of A. If $u = 1$ and $v = 2$, obviously every ordered pair (i, j) from $V \times V$ occurs in these two columns, in the row corresponding to (i, j). If $u = 1$ or 2 and $v \geq 3$, every ordered pair (i, j) appears since every column of $A^{(v)}$ is a permutation of $\{1, 2, \ldots, n\}$. Finally, if $u,\ v \geq 3$, every ordered pair (i, j) appears since $A^{(u)}$ and $A^{(v)}$ are orthogonal.

Conversely, suppose that A is an OA(k, n) on $V = \{1, 2, \ldots, n\}$. We define $A^{(u)}$, $u \in \{1, 2, \ldots, k-2\}$, as follows. Given i and j, there is a unique r so that $i = A(r, 1)$ and $j = A(r, 2)$. Then let

$$a_{ij}^{(u)} = A(r, u + 2).$$

It is not hard to show that $A^{(1)}, A^{(2)}, \ldots, A^{(k-2)}$ defined this way are orthogonal Latin squares of order n. Proof is left as an exercise (Exercise 21). Q.E.D.

Example 9.8 Authentication Codes in Cryptography In cryptography, we are concerned with checking the authenticity of messages. We use codes to help us encrypt those messages so they will be hard to modify, steal, or otherwise falsify. For more on codes, see Chapter 10. Suppose that A sends a message to B. In the theory of cryptography, we usually refer to A as Alice and B as Bob. Alice sends a message by email or fax or from her cellular telephone, all insecure channels. Bob wants to be sure that the message was really sent by Alice and, also, that no one altered the message that Alice sent. The message could be an order to purchase something, for example, so this problem is a central one in electronic commerce.

We shall consider the possibility that an outsider O, to be called Oscar, interferes with the message sent from Alice to Bob. Let us suppose that Oscar can simply send a message to Bob impersonating Alice, or that Oscar can modify a message sent by Alice. Bob and Alice protect against Oscar's "attacks" by sending, along with a message, an authentication code. Let M be the set of possible messages, C be a set of "authenticators," and K be a set of "keys." Alice and Bob agree on a key beforehand, when they meet in person or over a secure channel. Let us suppose that they choose the key from K at random. Associated with each key $k \in K$ is an *authentication rule* r_k that assigns an authenticator $r_k(m) \in C$ to each message $m \in M$. If Alice wants to send message m to Bob, she sends the message (m, c), where $c = r_k(m)$. When Bob receives the message (m, c), he checks that, in fact, c is $r_k(m)$. If not, Bob has reason to believe that Oscar did something and he doubts the message. Of course, it is possible that Oscar might have guessed $r_k(m)$ correctly, and thus this process will not detect all attacks by Oscar. However, Alice and Bob will be happy if the probability that this will happen is small and independent of the actual message that is sent.

Orthogonal arrays can be used to construct authentication rules. Suppose that $M = \{1, 2, \ldots, p\}$, $C = \{1, 2, \ldots, n\}$, and $K = \{1, 2, \ldots, n^2\}$. Let matrix A be an $\mathrm{OA}(p, n)$, with rows indexed by elements of K and columns by elements of M. Define $r_k(m) = a_{km}$.

What is the probability, if Oscar sends a message (m, c) to Bob, that in fact $c = r_k(m)$? We call this the *impersonation probability*. We can assume that Oscar knows the matrix A, but Oscar doesn't know which $k \in K$ is being used. Given m and c, there are n possible rows i of matrix A such that $a_{im} = c$, and there are n^2 rows of A. Hence, if $c = r_k(m)$, the probability that Oscar will choose a row i so that $r_i(m)$ is also equal to c is $n/n^2 = 1/n$. This code thus gives Oscar only a one in n chance of impersonating Alice.

What if Oscar simply replaces a message (m, c) that Alice sends by another message (m', c')? The *deception probability* is the probability that $c' = r_k(m')$. In other words, the deception probability is the probability that Bob will think that the message he received is authentic, thus falling for Oscar's deception. Oscar saw that Alice sent message (m, c), so he knows that $r_k(m) = c$, but he doesn't know k. He has to hope that $r_k(m') = c'$. For any two columns m, m' of A, the ordered pair (c, c') appears in those two columns in exactly one row of A. There are n rows of A in which c appears in column m. Thus, if Oscar picks one of those rows at random, the chances of his picking a row in which c' appears in column m are $1/n$. Therefore, the deception probability is $1/n$.

The authentication code Alice and Bob will choose will depend on how small they want the impersonation and deception probabilities to be. For further information about authentication codes, see Colbourn and Dinitz [1996]. ■

Example 9.9 Threshold Schemes and Secret Sharing There are situations where a decision or action is sufficiently sensitive that it must require concurrence by more than one member of a group. This is the case, for example, with the secret codes for setting off a nuclear attack, or in banks when more than one person is needed to identify the secret combination needed to open the vault.

Suppose that I is a set of p people, κ is a secret key to initiating an action (such as opening the vault or unleashing the attack), $q \geq 2$ is a fixed integer, and we want to make sure that any q of the people in the group can together determine κ, while no subgroup of fewer than q people can do so. A method for accomplishing this with high probability is called a (q,p)-*threshold scheme*. Fix a set K of keys and identify a leader not in I. The leader gives each person partial information about κ, chosen from a set P of partial information. The leader has to do this so that partial information available to any q people is enough to figure out κ, whereas that available to smaller subgroups is not. We consider the case $q = 2$.

Suppose that $K = P = \{1, 2, \ldots, n\}$, and let A be an orthogonal array OA$(p+1, n)$. Associate the first p columns of A with the participants and the last column with the keys. All people in the group are given A. Given $\kappa \in K$, let $R_\kappa = \{i : a_{i,p+1} = \kappa\}$ be the set of rows whose last entry is κ. The leader chooses a row $i \in R_\kappa$ at random and gives the partial information a_{iu} to the uth person.

Can person u and person v determine the key κ? Suppose that u gets partial information p_u and v gets partial information p_v. Since there is a unique row i with $a_{ru} = p_u$, $a_{rv} = p_v$, u and v can determine r and, therefore, can find $a_{r,p+1}$, which is the key κ they need.

Can any one person u determine κ based solely on his or her partial information p_u? For any possible value κ' of the key, there is a unique row i for which $a_{iu} = p_u$, $a_{i,p+1} = \kappa'$. Person u has no way of knowing (without sharing information with someone else) which of the n possible rows i is correct (i.e., was chosen by the leader). Thus, the probability of u correctly guessing the key based solely on his partial information is $1/n$. Therefore, we have found a $(2,p)$-threshold scheme that works with high probability.

For further information about threshold schemes and secret sharing, see Colbourn and Dinitz [1996]. ■

EXERCISES FOR SECTION 9.2

1. For each pair of Latin squares in Table 9.16, determine if it is orthogonal.
2. Check that the three Latin squares of Table 9.11 form an orthogonal family.
3. For each family of Latin squares of Table 9.17, determine if it is orthogonal.
4. Suppose that A is an $n \times n$ Latin square. For each of the following operations, determine if it results in a new Latin square.

   (a) Interchange the entries 2 and 4 whenever they occur.

   (b) Replace each row by going from last to first.

Table 9.16: Pairs of Latin Squares for Exercises of Section 9.2

1	2	3
2	3	1
3	2	1

1	2	3
3	1	2
2	3	1

(a)

1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

1	2	3	4
3	4	1	2
2	3	4	1
4	1	2	3

(b)

1	2	3	4	5
2	3	4	5	1
3	4	5	1	2
4	5	1	2	3
5	1	2	3	4

5	1	2	3	4
4	5	1	2	3
3	4	5	1	2
2	3	4	5	1
1	2	3	4	5

(c)

1	2	3	4	5	6
6	1	2	3	4	5
2	3	4	5	6	1
5	6	1	2	3	4
3	4	5	6	1	2
4	5	6	1	2	3

1	2	3	4	5	6
2	3	4	5	6	1
5	6	1	2	3	4
3	4	5	6	1	2
4	5	6	1	2	3
6	1	2	3	4	5

(d)

Table 9.17: Families of Latin Squares for Exercises of Section 9.2

1	3	2	4
3	1	4	2
2	4	1	3
4	2	3	1

3	1	4	2
2	4	1	3
1	3	2	4
4	2	3	1

2	4	1	3
1	3	4	2
3	1	4	2
4	2	3	1

(a)

1	2	3	4	5
2	3	4	5	2
3	4	5	1	2
4	5	1	2	3
5	1	2	3	4

1	2	3	4	5
3	4	5	1	2
5	1	2	3	4
2	3	4	5	1
4	5	1	2	3

1	2	3	4	5
5	1	2	3	4
4	5	1	2	3
3	4	5	1	2
2	3	4	5	1

(b)

Table 9.18: An Orthogonal Family of Latin Squares

$$\begin{array}{|ccccc|}\hline 5&1&2&3&4\\4&5&1&2&3\\3&4&5&1&2\\2&3&4&5&1\\1&2&3&4&5\\\hline\end{array}\quad\begin{array}{|ccccc|}\hline 4&5&1&2&3\\2&3&4&5&1\\5&1&2&3&4\\3&4&5&1&2\\1&2&3&4&5\\\hline\end{array}\quad\begin{array}{|ccccc|}\hline 3&4&5&1&2\\5&1&2&3&4\\2&3&4&5&1\\4&5&1&2&3\\1&2&3&4&5\\\hline\end{array}\quad\begin{array}{|ccccc|}\hline 2&3&4&5&1\\3&4&5&1&2\\4&5&1&2&3\\5&1&2&3&4\\1&2&3&4&5\\\hline\end{array}$$

(c) Replace A by its transpose.

5. For which of the following numbers n can you be sure that there is an orthogonal family of 3 Latin squares of order n? Why?

   (a) $n = 12$ (b) $n = 17$ (c) $n = 21$ (d) $n = 24$
   (e) $n = 33$ (f) $n = 36$ (g) $n = 55$ (h) $n = 75$
   (i) $n = 161$ (j) $n = 220$ (k) $n = 369$ (l) $n = 539$

6. If $n = 539$, show that there is a set of 10 pairwise orthogonal Latin squares of order n.

7. If $n = 130$, does there exist a pair of orthogonal Latin squares of order n? Why?

8. Does there exist a pair of orthogonal Latin squares of order 12? Why?

9. If n is divisible by 4, can you be sure (with the theorems we have stated) whether or not there is a set of three pairwise orthogonal Latin squares of order n? Give a reason for your answer.

10. For the orthogonal family of Latin squares of order 5 in Table 9.18, use the procedure in the proof of Theorem 9.1 to rearrange elements so that the first row of each Latin square is 12345.

11. Suppose that two orthogonal 8×8 Latin squares both have 87654321 as the first row.

    (a) Is it possible for them to have the same $2, 4$ entry?

    (b) What does your answer tell you about the number of possible 8×8 pairwise orthogonal Latin squares each of which has 87654321 as the first row?

12. Suppose that two orthogonal 7×7 Latin squares both have 1234567 as the last row.

    (a) Is it possible for them to have the same $1, 3$ entry?

    (b) What does your answer tell you about the number of possible 7×7 pairwise orthogonal Latin squares each of which has 1234567 as the last row?

13. Suppose that two orthogonal 4×4 Latin squares both have 1234 as the main diagonal.

    (a) Is it possible for them to have the same $2, 3$ entry?

    (b) What does your answer tell you about the number of possible 4×4 pairwise orthogonal Latin squares each of which has 1234 as the main diagonal?

14. Use the Latin squares of Table 9.14 to find a pair of orthogonal Latin squares of order 12.
15. Find a pair of orthogonal Latin squares of order 9.
16. If there exists a pair of orthogonal Latin squares of order n, and A is a Latin square of order n, A is not necessarily a member of an orthogonal pair of Latin squares. Give an example to illustrate this.
17. Given the OA$(4,3)$ of Table 9.15, find two orthogonal Latin squares from it.
18. Given the OA$(5,4)$ of Table 9.15, find three pairwise orthogonal Latin squares from it.
19. Given the two orthogonal Latin squares of Table 9.10, find a corresponding orthogonal array OA$(4,4)$.
20. Given the orthogonal family of three Latin squares of Table 9.13, find a corresponding orthogonal array OA$(5,4)$.
21. Complete the proof of Theorem 9.7.
22. For which of the following values of p and n does there exist an OA(p,n)?

    (a) $p=3$, $n=81$ (b) $p=4$, $n=6$ (c) $p=4$, $n=63$

23. (Stinson [2003])

    (a) If Alice and Bob deal with a set of 200 possible messages and want to be sure to limit the probability of impersonation by Oscar to less than $1/1000$, explain how an orthogonal array OA$(200,1009)$ would help them.

    (b) Show that there is such an orthogonal array.

    (c) Explain why a number smaller than 1009 in OA$(200,1009)$ was not used.

24. In Example 9.8, assume that Oscar has information that limits the possible keys to a fixed subset of size s. How does this change the probability of impersonation?
25. (Stinson [2003])

    (a) If a group of 10 people wants to build a $(2,10)$-threshold scheme and to be sure that the probability that any one person can guess the secret key is less than $1/100$, explain how an orthogonal array OA$(11,101)$ can accomplish this.

    (b) Show that there is such an orthogonal array.

26. (Stinson [1990, 2003]) In Example 9.8, suppose that the method of authentication rules is used, but do not assume that the rules are determined from an orthogonal array. Show that the probability of impersonation is always at least $1/|C|$ and equals $1/|C|$ if and only if $|\{k : r_k(m)=c\}| = |K|/|C|$, for all $m \in M$, $c \in C$.
27. (Stinson [1990, 2003]) In the situation of Exercise 26, suppose that the probability of impersonation is $1/|C|$. Show that the probability of deception is at least $1/|C|$ and it is equal to $1/|C|$ if and only if $|\{k : r_k(m)=c\} \cap \{k : r_k(m')=c'\}| = |K|/|C|$, for all $m, m' \in M$, $c, c' \in C$.
28. (Stinson [1990, 2003]) In the situation of Exercise 26, show that if $|M| = p$, $|C| = n$, and if the probability of impersonation and the probability of deception are both $1/n$, then $|K| \geq n^2$, with equality if and only if the authentication rules define the rows of an orthogonal array OA(p,n). (This shows that the orthogonal arrays give authentication rules that minimize the number of keys required.)

9.3 FINITE FIELDS AND COMPLETE ORTHOGONAL FAMILIES OF LATIN SQUARES[6]

In this section we aim to present a constructive proof of Theorem 9.2, namely, that if $n > 1$ and $n = p^k$ for p prime, there is a complete orthogonal family of Latin squares of order n. We begin with some mathematical preliminaries.

9.3.1 Modular Arithmetic

Arithmetics with only finitely many numbers underlie the construction of combinatorial designs. They are also vitally important in computing, where there are practical bounds on the size of the sets of integers that can be considered. In this subsection we introduce a simple example of an arithmetic with only finitely many elements, modular arithmetic. In Section 9.3.3 we introduce a general notion of an arithmetic with only finitely many elements: namely, a finite field. Then we use finite fields to construct complete orthogonal families of Latin squares. Modular arithmetic and finite fields underlie the operation of the shift registers that operate in a computer to take a bit string and produce another one. For a discussion of this application, see, for example, Fisher [1977].

Let us consider the remainders left when integers are divided by the number 3. We find that

$$\begin{array}{rclrclrcl} 0 & = & 0\cdot 3+0, & 1 & = & 0\cdot 3+1, & 2 & = & 0\cdot 3+2,\\ 3 & = & 1\cdot 3+0, & 4 & = & 1\cdot 3+1, & 5 & = & 1\cdot 3+2,\\ 6 & = & 2\cdot 3+0, & 7 & = & 2\cdot 3+1, & 8 & = & 2\cdot 3+2,\\ 9 & = & 3\cdot 3+0, & 10 & = & 3\cdot 3+1, & 11 & = & 3\cdot 3+2. \end{array}$$

The remainder is always one of the three integers $0, 1,$ or 2. We say that two integers a and b are *congruent modulo* 3, and write $a \equiv b \pmod 3$, if they leave the same remainder on division by 3. For instance, $2 \equiv 5 \pmod 3$ and $1 \equiv 7 \pmod 3$. In general, if $a, b,$ and n are integers, we say that a is *congruent to b modulo n*, and write $a \equiv b \pmod n$ if a and b leave the same remainder on division by n. For instance, $29 \equiv 17 \pmod 4$, since $29 = 7 \cdot 4 + 1$ and $17 = 4 \cdot 4 + 1$. Congruence modulo 12 is used every day when we look at a clock. The hands of the clock indicate the hour modulo 12. Similarly, the mileage indicator in a car gives the mileage the car has traveled modulo 1,000,000 (depending on its make and model).

For all its beauty and functionality, a major software snafu was "created" due to modular arithmetic. The *Year 2000 problem* (Y2K) arose due to the internal workings of some computer software. Many software developers used a two-digit indicator when recording a year; the 1900s was being assumed. For example, using 2/17/59, the "59" refers to 1959. Thus, the two digits were modulo 100. The Y2K problem came to light when it was realized that in the year 2000, this same software would not be able to distinguish between the years 1900 and 2000 when using a 00 designation.

[6] This section may be omitted without loss of continuity. As an alternative, the reader might wish to read all but Section 9.3.5.

Table 9.19: The Operations $+$ and $\times$ of Addition and Multiplication Modulo 2 on Z_2

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

Table 9.20: The Operations $+$ and $\times$ of Addition and Multiplication Modulo 3 on Z_3

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\times$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Let us now fix a number n and consider the set of integers $Z_n = \{0, 1, 2, \ldots, n-1\}$. Every integer is congruent modulo n to one of the integers in the set Z_n. If we add two integers in the set Z_n, their sum is not necessarily in the set. However, their sum is congruent to an element in Z_n. In performing *modular addition*, if a and b are in Z_n, we *define* $a+b$ to be that number in Z_n which is congruent to the ordinary sum of a and b, modulo n. For instance, suppose that $n = 3$. Then $2+2$ is 4 in ordinary arithmetic, which is congruent to 1, modulo 3. Hence, in addition modulo 3, $2+2$ is 1. Similarly, $1+2$ is 0 and $1+1$ is 2. In addition modulo 4, $3+3$ is 2 and $2+2$ is 0. Modular multiplication works similarly. If a, b are in Z_n, we define $a \times b$ to be that integer in Z_n congruent to the ordinary product of a and b, modulo n. For instance, in multiplication modulo 3, 2×2 is 1, since the ordinary product of 2 and 2 is 4, which is congruent to 1 modulo 3. Similarly, in multiplication modulo 4, 3×3 is 1 and 2×2 is 0.

We can summarize the *operations* $+$ and $\times$ on Z_n by giving addition and multiplication tables. Tables 9.19 and 9.20 give these tables for the cases $n = 2$ and $n = 3$, respectively. In these tables, the elements of Z_n are listed in the same order along both rows and columns and the entry in the row corresponding to a and the column corresponding to b is $a + b$ or $a \times b$, depending on the table.

9.3.2 Modular Arithmetic and the RSA Cryptosystem[7]

Sending sensitive information (for example, credit card numbers) over the Internet has created much interest in cryptographic codes, codes whose goal is to conceal information. (Once information is concealed, it may be encoded to ensure correct transmission and decoding. Coding theory, in particular error-correcting and

[7] This section follows Hill [1991]. It may be omitted.

error-detecting codes, is the subject of Chapter 10.) We shall describe a "public-key cryptosystem" developed by Rivest, Shamir, and Adleman [1978], now called RSA, and based on the work of Diffie and Hellman [1976]. RSA was discussed in Section 7.1.3.

Suppose that you need to send your credit card number to a retail store over the Internet. How can you change this number (encrypt it) so that others don't have access to it if the transmission is read by someone other than at the intended store? The answer is based on prime numbers and modular arithmetic.

First, a theorem about prime numbers is needed.

Theorem 9.8 (Fermat's Little Theorem) If p is a prime, x a positive integer, and $x \not\equiv 0 \pmod{p}$, then

$$x^{p-1} \equiv 1 \pmod{p}.$$

Proof.[8] Consider the set $A = \{1x, 2x, 3x, \ldots, (p-1)x\}$. The elements of A are all distinct modulo p. (To see this, suppose not, i.e., that $ix \equiv jx \pmod{p}$ for $1 \leq i \neq j \leq p-1$. Since $x \not\equiv 0 \pmod{p}$, x^{-1} exists. Then multiplying both sides by x^{-1} results in $i \equiv j \pmod{p}$, which implies that $i = j$ since $1 \leq i, j < p$; a contradiction.) Also, since $x \not\equiv 0 \pmod{p}$, each element of A, ix, is $\not\equiv 0 \pmod{p}$ since $1 \leq i \leq p-1$. Thus, $\{1x \pmod{p}, 2x \pmod{p}, \ldots, (p-1)x \pmod{p}\} = \{1, 2, \ldots, p-1\}$. Hence,

$$(1x)(2x)\cdots((p-1)x) \equiv (1)(2)\cdots(p-1) \pmod{p}$$

or

$$(p-1)!x^{p-1} \equiv (p-1)! \pmod{p}.$$

It is clear that $(p-1)! \not\equiv 0 \pmod{p}$, so $[(p-1)!]^{-1}$ exists. Multiplying both sides by $[(p-1)!]^{-1}$ finishes the proof. Q.E.D.

To use RSA, the store chooses two distinct and extremely large prime numbers p and q. It then computes $r = pq$ and finds two positive integers s and t such that

$$st \equiv 1 \pmod{(p-1)(q-1)},$$

i.e., so that $st = j(p-1)(q-1) + 1$ for some integer j. The store then makes publicly known r and s (hence the term *public-key*). As an example, suppose the store chooses the primes $p = 37$ and $q = 23$. Thus, $r = 37 \cdot 23 = 851$, and the store can choose $s = 5$ and $t = 317$ since

$$5 \cdot 317 = 1585 = 2(36)(22) + 1 = 2(37-1)(23-1) + 1 \equiv 1 \,[\mathrm{mod}\ (37-1)(23-1)].$$

(Other choices for s and t are possible. However, it is wise to avoid either of them being 1.) It is important to note that the store does not make public p, q, or t, only r and s.

[8] The proof may be omitted. It uses well-known properties of modular arithmetic that are not developed in detail here. See Exercise 9 for an alternative proof.

To encrypt your credit card number n before sending it over the Internet, you calculate and send

$$m = n^s \pmod{r}. \tag{9.2}$$

The two questions that arise are: (1) If someone steals this number m, how could they find your credit card number n? (2) When the store receives this number m, how do they calculate (*decrypt*) your credit card number n? Both questions are important.

The store computes $m^t \pmod{r}$ and, using Fermat's Little Theorem (Theorem 9.8), finds your credit card number n. Note that the store essentially only needs t for the decryption! For $m^t \equiv n^{st} \pmod{r}$ by Equation (9.2) and $n^{st} = n^{j(p-1)(q-1)+1}$. Next, if $n \not\equiv 0 \pmod{p}$, then $n^{j(p-1)(q-1)+1} = \left(n^{p-1}\right)^{j(q-1)} n \equiv n \pmod{p}$ by Fermat's Little Theorem (Theorem 9.8). And if $n \equiv 0 \pmod{p}$, then certainly $n^{j(p-1)(q-1)+1} \equiv n \pmod{p}$. In either case, p must be a factor of $n^{j(p-1)(q-1)+1} - n$. Similarly, q must be a factor of $n^{j(p-1)(q-1)+1} - n$. Since p and q are distinct primes and $r = pq$,

$$n^{j(p-1)(q-1)+1} \equiv n \pmod{r} = n,$$

which is your credit card number, as long as your credit card number is not larger than r. This will not be the case since the security of this system is based on using a very large r.

Actually, the security of the entire system is based on the unavailability of t. Since r and s are publicly known, why can't t be deduced from them by someone who wants to steal your credit card number? The answer lies in the fact that this would require knowledge of the original primes p and q, i.e., the ability to factor $r = pq$. However, prime factorization is a difficult problem. In fact, in the notation of Section 2.18, it is in the class of NP-complete problems. For our example, $r = 851$ would not take long to factor. But if the primes p and q are each 100-digit numbers then factoring r, at about 200 digits, would not be feasible using the present day (i.e., 2003) best-known algorithms and fastest computers (Stinson [2003]). So, as long as no deterministic algorithm, whose complexity is polynomial, is found for prime factorization (and credit card numbers don't exceed 200 digits), RSA cryptosystems should be secure.[9] For more on cryptographic codes and RSA, see Garrett [2001], Joye and Quisquater [1998], Kaliski [1997], Koblitz [1994], Menezes, van Oorschot, and Vanstone [1997], Salomaa [1996], and Sloane [1981].

9.3.3 The Finite Fields GF(p^k)

We turn next to a generalization of the modular arithmetic introduced in Section 9.3.1. Suppose that X is a set. A *binary operation* $\circ$ on X is a function that

[9] In Section 7.1.3 we noted that recent work has shown that the problem of testing whether or not an integer is prime can be done efficiently (i.e., in polynomial time). Although this result does not say anything about the possibility of factoring a number into primes efficiently, it certainly raises the issue as to whether or not this could be done. This is a crucial question since hardness of factoring is so critical in cryptography.

assigns to each ordered pair of elements of X another element of X, usually denoted $a \circ b$. For example, if X is the set of integers, then $+$ and $\times$ define binary operations on X. If X is a finite set, we can define a binary operation $\circ$ by giving a table such as those in Tables 9.19 or 9.20.

A *field* $\mathcal{F}$ is a triple $(F, +, x)$, where F is a set and $+$ and $\times$ are two binary operations on F (not necessarily the usual operations of $+$ and $\times$), with certain conditions holding. These are the following:[10]

Condition **F1** (*Closure*).[11] For all a, b in F, $a + b$ is in F and $a \times b$ is in F.

Condition **F2** (*Associativity*). For all a, b, c in F,

$$\begin{aligned} a + (b + c) &= (a + b) + c, \\ a \times (b \times c) &= (a \times b) \times c. \end{aligned}$$

Condition **F3** (*Commutativity*). For all a, b in F,

$$\begin{aligned} a + b &= b + a, \\ a \times b &= b \times a. \end{aligned}$$

Condition **F4** (*Identity*).

(a) There is an element in F, which is denoted 0 and called the *additive identity*, so that for all a in F,

$$a + 0 = a.$$

(b) There is an element in F different from 0, which is denoted 1 and called the *multiplicative identity*, so that for all a in F,

$$a \times 1 = a.$$

Condition **F5** (*Inverse*).

(a) For all a in F, there is an element b in F, called an *additive inverse* of a, so that

$$a + b = 0.$$

(b) For all a in F with $a \neq 0$, there is an element b in F, called a *multiplicative inverse* of a, so that

$$a \times b = 1.$$

[10] Our treatment of fields is necessarily brief. The reader without a background in this subject might consult such elementary treatments as Dornhoff and Hohn [1978], Durbin [1999], Fisher [1977], or Gilbert and Gilbert [1999].

[11] Condition **F1** is actually implicit in our definition of operation.

Condition **F6** (*Distributivity*). For all a, b, c in F,

$$a \times (b + c) = (a \times b) + (a \times c).$$

[Conditions **F1, F2, F4,** and **F5** say that $(F, +)$ is a group in the sense of Chapter 8. Also, if $F' = F$ less the element 0, these conditions also say that $(F', \times)$ is a group, since we can show that $a \times b$ is never 0 if $a \neq 0$ and $b \neq 0$ and we can show that the multiplicative inverse of a is never 0.] Note that it is possible to prove from conditions **F1**–**F6** that the additive and multiplicative inverses of an element a are, respectively, unique; they are denoted, respectively, as $-a$ and a^{-1}.

The following are examples of fields.

1. $(Re, +, \times)$, where Re is the set of real numbers and $+$ and $\times$ are the usual addition and multiplication operations.

2. $(Q, +, \times)$, where Q is the set of rational numbers and $+$ and $\times$ are the usual addition and multiplication operations.

3. $(C, +, \times)$, where C is the set of complex numbers and $+$ and $\times$ are the usual addition and multiplication operations.

However, $(Z, +, \times)$, where Z is the set of integers and $+$ and $\times$ are the usual addition and multiplication operations, is not a field. Conditions **F1**–**F4** and **F6** hold. However, part (b) of condition **F5** fails. There is no b in Z so that $2 \times b = 1$.

We now consider some examples of *finite fields*, fields where F is a finite set. Consider $(Z_2, +, \times)$, where $+$ and $\times$ are modulo 2. Then this is a field, as is easy to check. Note that 0 and 1 are, respectively, the additive and multiplicative identities. The additive inverse of 1 is 1, since $1 + 1 = 0$, and the multiplicative inverse of 1 is 1, since $1 \times 1 = 1$.

Similarly, $(Z_3, +, \times)$, where $+$ and $\times$ are modulo 3, is a field. The additive and multiplicative inverses are again 0 and 1, respectively. Note that the additive inverse of 2 is 1, since $2 + 1 = 0$. The multiplicative inverse of 2 is 2, since $2 \times 2 = 1$.

Is Z_n under modular addition and multiplication always a field? The answer is no. Consider Z_6. We have $3 \times 2 = 0$. If Z_6 under addition and multiplication modulo 6 is a field, let 2^{-1} denote the multiplicative inverse of 2. We have

$$(3 \times 2) \times 2^{-1} = 0 \times 2^{-1} = 0.$$

However,

$$(3 \times 2) \times 2^{-1} = 3 \times (2 \times 2^{-1}) = 3 \times 1 = 3.$$

We conclude that $0 = 3$, a contradiction.

Theorem 9.9 . For $n \geq 2$, Z_n under addition and multiplication modulo n is a field if and only if n is a prime number.

The proof of Theorem 9.9 is left as an exercise (Exercise 16).

We close this subsection by asking: For what values of n does there exist a finite field of n elements? We shall be able to give an explicit answer. Note that by

Table 9.21: Addition and Multiplication Tables for a Field GF(2^2) of Four Elements

$+$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

$\times$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Theorem 9.9, Z_4 does not define a field under modular addition and multiplication. However, it is possible to define addition and multiplication operations on $\{0, 1, 2, 3\}$ which define a field. Such operations are shown in Table 9.21. Verification that these define a field is left to the reader (Exercise 17). (The reader familiar with algebra will be able to derive these tables by letting $2 = w$ and $3 = 1 + w$ and performing addition and multiplication modulo the irreducible polynomial $1 + w + w^2$ over GF[2], the finite field of 2 elements to be defined below.) The arithmetic of binary numbers which is actually used in many large computers is based on Z_{2^n} for some n. (Here, $n = 2$.) For a discussion of this particular arithmetic, see Dornhoff and Hohn [1978], Hennessy and Patterson [1998, Ch. 4], and Patterson and Hennessy [1998, Appendix A].

If $n = 6$, it is not possible to define addition and multiplication on a set of n elements, such as $\{0, 1, 2, 3, 4, 5\}$, so that we get a finite field. The situation is summarized in Theorem 9.10.

Theorem 9.10 If $(F, +, \times)$ is a finite field, there is a prime number p and a positive integer k so that F has p^k elements. Conversely, for all prime numbers p and positive integers k, there is a finite field of p^k elements.

The proof of this theorem can be found in most books on modern algebra: for instance, any of the references in footnote 10 on page 517. It turns out that there is essentially only one field of p^k elements for p a prime and k a positive integer, in the sense that any two of these fields are isomorphic.[12] The unique field of p^k elements will be denoted GF(p^k). (The letters GF stand for Galois field and are in honor of the famous mathematician Evariste Galois, who made fundamental contributions to modern algebra.)

9.3.4 Construction of a Complete Orthogonal Family of $n \times n$ Latin Squares if n Is a Power of a Prime

We now present a construction of a complete orthogonal family of $n \times n$ Latin squares that applies whenever $n = p^k$, for p prime and k a positive integer, and

[12] Two fields $\mathcal{F}$ and $\mathcal{G}$ are isomorphic if there is a one-to-one mapping from $\mathcal{F}$ onto $\mathcal{G}$ that preserves addition and multiplication.

Table 9.22: The Orthogonal Latin Squares $A^{(1)}$ and $A^{(2)}$ Defined from the Finite Field GF(3) by (9.3)

$$A^{(1)} = \begin{array}{|ccc|} \hline 2 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 2 & 0 \\ \hline \end{array} \qquad A^{(2)} = \begin{array}{|ccc|} \hline 0 & 1 & 2 \\ 2 & 0 & 1 \\ 1 & 2 & 0 \\ \hline \end{array}$$

$n > 1$. This will prove Theorem 9.2. Let $b_1, b_2, \ldots, b_n$ be elements of a finite field $\mathrm{GF}(n)$ of $n = p^k$ elements. Let b_1 be the multiplicative identity of this field and b_n be the additive identity. For $e = 1, 2, \ldots, n-1$, define the $n \times n$ array $A^{(e)} = (a_{ij}^{(e)})$ by taking

$$a_{ij}^{(e)} = (b_e \times b_i) + b_j, \tag{9.3}$$

where $+$ and $\times$ are the operations of the field $\mathrm{GF}(n)$. In Section 9.3.5 we show that $A^{(e)}$ is a Latin square and that $A^{(1)}, A^{(2)}, \ldots, A^{(n-1)}$ is an orthogonal family. Thus, if $n > 1$, we get a complete orthogonal family of $n \times n$ Latin squares. For instance, if $n = 3$, we use GF(3), whose addition and multiplication operations are addition and multiplication modulo 3, as defined by Table 9.20. We let $b_1 = 1, b_2 = 2$, and $b_3 = 0$. (Remember that b_1 is to be chosen as the multiplicative identity and b_n as the additive identity.) Then we find that $A^{(1)}$ and $A^{(2)}$ are given by Table 9.22. To see how the $1, 2$ entry of $A^{(2)}$ was computed, for instance, note that

$$(b_2 \times b_1) + b_2 = (2 \times 1) + 2 = 2 + 2 = 1.$$

It is easy to check directly that $A^{(1)}$ and $A^{(2)}$ of Table 9.22 are Latin squares and that they are orthogonal.

To give another example, suppose that $n = 4$. Then we use $\mathrm{GF}(4) = \mathrm{GF}(2^2)$, whose addition and multiplication operations are given in Table 9.21. Taking $b_1 = 1, b_2 = 2, b_3 = 3$, and $b_4 = 0$, and using (9.3), we get the three pairwise orthogonal Latin squares of Table 9.23. To see how these entries are obtained, note, for example, that

$$\begin{aligned} a_{23}^{(3)} &= (b_3 \times b_2) + b_3 \\ &= (3 \times 2) + 3 \\ &= 1 + 3 \\ &= 2, \end{aligned}$$

where we have used the addition and multiplication rules of Table 9.21.

Table 9.23: The Orthogonal Family of 4×4 Latin Squares Obtained from the Finite Field GF(2^2) of Table 9.21

$$A^{(1)} = \begin{array}{|cccc|}\hline 0 & 3 & 2 & 1 \\ 3 & 0 & 1 & 2 \\ 2 & 1 & 0 & 3 \\ 1 & 2 & 3 & 0 \\ \hline\end{array} \quad A^{(2)} = \begin{array}{|cccc|}\hline 3 & 0 & 1 & 2 \\ 2 & 1 & 0 & 3 \\ 0 & 3 & 2 & 1 \\ 1 & 2 & 3 & 0 \\ \hline\end{array} \quad A^{(3)} = \begin{array}{|cccc|}\hline 2 & 1 & 0 & 3 \\ 0 & 3 & 2 & 1 \\ 3 & 0 & 1 & 2 \\ 1 & 2 & 3 & 0 \\ \hline\end{array}$$

9.3.5 Justification of the Construction of a Complete Orthogonal Family if $n = p^k$ [13]

To justify the construction of Section 9.3.4, we first show in general that if $A^{(e)}$ is defined by (9.3), then it is a Latin square. Suppose that $a_{ij}^{(e)} = a_{ik}^{(e)}$. Then

$$(b_e \times b_i) + b_j = (b_e \times b_i) + b_k. \tag{9.4}$$

By adding the additive inverse c of $(b_e \times b_i)$ to both sides of (9.4) and using associativity and commutativity of addition, we find that

$$\begin{aligned} (c + [(b_e \times b_i) + b_j]) &= (c + [(b_e \times b_i) + b_k]), \\ ([c + (b_e \times b_i)] + b_j) &= ([c + (b_e \times b_i)] + b_k), \\ 0 + b_j &= 0 + b_k, \\ b_j &= b_k, \\ j &= k. \end{aligned}$$

Thus, all elements of the same row are different.

Next, suppose that $a_{ji}^{(e)} = a_{ki}^{(e)}$. Then

$$(b_e \times b_j) + b_i = (b_e \times b_k) + b_i. \tag{9.5}$$

By adding the additive inverse of b_i to both sides of (9.5) and using associativity of addition, we obtain

$$b_e \times b_j = b_e \times b_k. \tag{9.6}$$

We now multiply (9.6) by the multiplicative inverse a of b_e, which exists since $b_e \neq 0$, and use commutativity and associativity of the $\times$ operation. We obtain

$$\begin{aligned} a \times (b_e \times b_j) &= a \times (b_e \times b_k), \\ (a \times b_e) \times b_j &= (a \times b_e) \times b_k, \\ 1 \times b_j &= 1 \times b_k, \\ b_j &= b_k, \\ j &= k. \end{aligned}$$

[13] This subsection may be omitted.

Thus, all elements of the same column are different, and we conclude that $A^{(e)}$ is a Latin square.

Finally, we verify orthogonality of $A^{(e)}$ and $A^{(f)}$, for $e \neq f$. Suppose that

$$(a_{ij}^{(e)}, a_{ij}^{(f)}) = (a_{kl}^{(e)}, a_{kl}^{(f)}).$$

Then

$$a_{ij}^{(e)} = a_{kl}^{(e)} \quad \text{and} \quad a_{ij}^{(f)} = a_{kl}^{(f)},$$

so

$$(b_e \times b_i) + b_j = (b_e \times b_k) + b_l \tag{9.7}$$

and

$$(b_f \times b_i) + b_j = (b_f \times b_k) + b_l. \tag{9.8}$$

Using the properties of fields freely, we subtract (9.8) from (9.7); that is, we add the additive inverse of both sides of (9.8) to both sides of (9.7). This yields

$$(b_e \times b_i) - (b_f \times b_i) = (b_e \times b_k) - (b_f \times b_k),$$

where $-$ means add the additive inverse. Thus, again using the properties of fields freely, we find that

$$(b_e - b_f) \times b_i = (b_e - b_f) \times b_k. \tag{9.9}$$

Finally, since $e \neq f$, it follows that $(b_e - b_f) \neq 0$, so $(b_e - b_f)$ has a multiplicative inverse. Multiplying (9.9) by this multiplicative inverse, we derive the equation

$$b_i = b_k,$$

whence

$$i = k.$$

Now (9.7) gives us

$$(b_e \times b_i) + b_j = (b_e \times b_i) + b_l,$$

from which we derive

$$b_j = b_l,$$
$$j = l.$$

Hence, $i = k$ and $j = l$, and we conclude that $A^{(e)}$ and $A^{(f)}$ are orthogonal. This completes the proof that $A^{(1)}, A^{(2)}, \ldots, A^{(n-1)}$ is an orthogonal family of $n \times n$ Latin squares.

EXERCISES FOR SECTION 9.3

1. For each of the following values of a and n, find a number b in $\{0, 1, \ldots, n-1\}$ so that $a \equiv b \pmod{n}$.

   (a) $a = 37, n = 5$ (b) $a = 42, n = 3$ (c) $a = 8, n = 10$

   (d) $a = 11, n = 9$ (e) $a = 625, n = 71$ (f) $a = 1652, n = 7$

2. For each of the following values of a, b, and n, compute $a+b$ and $a \times b$ using addition and multiplication modulo n.

   (a) $a=2, b=4, n=4$ (b) $a=4, b=5, n=12$ (c) $a=5, b=6, n=9$

   (d) $a=4, b=4, n=15$ (e) $a=3, b=11, n=2$ (f) $a=10, b=11, n=12$

3. Verify the following facts about congruence.

   (a) If $a \equiv b \pmod{n}$, then $b \equiv a (\mathrm{mod}\ n)$.

   (b) If $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$.

   (c) If $a \equiv a' \pmod{n}$ and $b \equiv b' \pmod{n}$, then $a+b \equiv a'+b' \pmod{n}$.

   (d) If $a \equiv a' \pmod{n}$ and $b \equiv b' \pmod{n}$, then $a \times b \equiv a' \times b' \pmod{n}$.

4. Suppose that $c_1 c_2 \cdots c_n$ is any permutation of $0, 1, 2, \ldots, n-1$. Build a matrix A as follows. The first row of A is the permutation $c_1 c_2 \cdots c_n$. Each successive row of A is obtained from the preceding row by adding 1 to each element and using addition modulo n.

   (a) Build A for the permutation 32401.

   (b) Show that A is always a Latin square.

5. (Williams [1949]) Let $n = 2m$ and let

$$0 \quad 1 \quad 2m-1 \quad 2 \quad 2m-2 \quad 3 \quad 2m-3 \quad \cdots \quad m+1 \quad m$$

   be a permutation of $\{0, 1, 2, ..., n-1\}$. Let A be the Latin square constructed from that permutation by the method of Exercise 4.

   (a) Build A for $m = 2$.

   (b) Show that for every value of $m \geq 1$, A is *horizontally complete* in the sense that whenever $1 \leq \alpha \leq n$ and $1 \leq \beta \leq n$ and $\alpha \neq \beta$, there is a row of A in which α is followed immediately by β. (Such Latin squares are important in agricultural experiments where we wish to minimize interaction of treatments applied to adjacent plots.)

   (c) For arbitrary $m \geq 1$, is A vertically complete (in the obvious sense)?

6. As opposed to a four-digit representation for the year, some computer software represents a date by the number of seconds that have elapsed since January 1, 1970. The seconds are stored in binary using 32 bits of storage. Eventually, the number of seconds since January 1, 1970 will exceed the 32 bits of storage and result in an incorrect year representation. This has been called the *Unix Time Problem.*

   (a) This representation uses congruence modulo n for what value of n?

   (b) For what date will the Unix Time Problem first occur?

7. Recall the values from Section 9.3.2: $p = 37$, $q = 23$, $r = 851$, $s = 5$, and $t = 317$.

   (a) If the number $m = 852$ is received by the store, find the credit card number of the customer.

   (b) Find m for the credit card number 123-45-6789.

(c) Find another possible s and t pair. That is, find positive integers s and t so that $st = j(37-1)(23-1)+1$ for some positive integer j. (Please avoid the trivial case of s or t being 1.)

8. From Section 9.3.2, suppose that the store chose the prime numbers $p = 37$ and $q = 41$. Then $r = 1517$.

   (a) Find positive integers s and t so that $st = j(37-1)(41-1)+1$ for some positive integer j. (Please avoid the trivial case of s or t being 1.)

   (b) Find m for the credit card number 123-45-6789.

   (c) Find the credit card number if the store receives the number 1163.

9. This exercise provides an alternative proof of Fermat's Little Theorem (Theorem 9.8).

   (a) Show that if p is a prime number, p divides the binomial coefficient $\binom{p}{i}$ for $1 \leq i \leq p-1$.

   (b) Argue by induction on positive integers x that $x^p \equiv x \pmod{p}$ by using the binomial theorem to expand $(x+1)^p$.

10. Write down the addition and multiplication tables for the following fields.

    (a) GF(5) (b) GF(7) (c) GF(9)

11. (a) Write down the tables for the binary operations of addition and multiplication modulo 4 on the set Z_4.

    (b) Find an element in Z_4 that does not have a multiplicative inverse.

12. Repeat Exercise 11 for addition and multiplication modulo 10 on the set Z_{10}.

13. Find the additive and multiplicative inverse of 8 in each of the following fields.

    (a) GF(11) (b) GF(13) (c) GF(17)

14. Repeat Exercise 13 for 6 in place of 8.

15. Which of the following triples $(F, +, \times)$ define fields?

    (a) F is the *positive* reals, $+$ and $\times$ are ordinary addition and multiplication.

    (b) F is the reals with an additional element ∞. The operations $+$ and $\times$ are the usual addition and multiplication operations on the reals, and in addition we have for all real numbers a,

    $$a + \infty = a \times \infty = \infty + a = \infty \times a = \infty + \infty = \infty \times \infty = \infty$$

    (c) F is Re, $a+b$ is ordinary addition, and $a \times b = 1$ for all a, b in F.

    (d) F is Re, and $a + b = a \times b = 0$ for all a, b in F.

16. Consider Z_n under addition and multiplication modulo n and consider the conditions for a field.

    (a) Show that condition **F1** holds.

    (b) Show that condition **F2** holds.

(c) Show that condition **F3** holds.

(d) Show that condition **F4** holds by showing that 0 and 1 are the additive and multiplicative identities, respectively.

(e) Show that condition **F5**(a) holds by showing that $n - a$ is the additive inverse of a.

(f) Show that condition **F6** holds.

(g) Show that condition **F5**(b) fails if n is not a prime number.

(h) Show that condition **F5**(b) holds if n is a prime number. [*Hint:* Use Fermat's Little Theorem (Theorem 9.8) to conclude that $a^{-1} = a^{n-2}$.]

17. Verify that Table 9.21 defines a field.

18. Use the method of Section 9.3.4 to find a complete orthogonal family of Latin squares of the following orders. [Parts (c) and (d) are for the reader with knowledge of modern algebra.]

(a) 5 (b) 7 (c) 8 (d) 9

19. For a given prime number n and integer k with $3 \le k \le n$, build an $n^2 \times k$ matrix A as follows: Build one row corresponding to each ordered pair (i, j) for $1 \le i, j \le n$. If $1 \le c \le k$, the entry in the row corresponding to (i, j) and column c is $i + jc \pmod{n}$. Show that A is an orthogonal array.

9.4 BALANCED INCOMPLETE BLOCK DESIGNS

9.4.1 (b, v, r, k, λ)-Designs

In Section 9.1 we pointed out that in a block design, it is not always possible to test each treatment in each block. For instance, in testing tire wear, if there are five brands of tires, then, as we observed, only four of these can be tested in any one block. Thus, it is necessary to use an incomplete block design. The basic incomplete block design we shall study is called a balanced incomplete block design. A *balanced block design* consists of a set V of $v \ge 2$ elements called *varieties* or *treatments*, and a collection of $b > 0$ subsets of V, called *blocks*, such that the following conditions are satisfied:

$$\text{each block consists of exactly the same number } k \text{ of varieties, } k > 0; \tag{9.10}$$

$$\text{each variety appears in exactly the same number } r \text{ of blocks, } r > 0; \tag{9.11}$$

$$\begin{gathered}\text{each pair of varieties appears simultaneously in exactly the}\\ \text{same number } \lambda \text{ of blocks, } \lambda > 0.\end{gathered} \tag{9.12}$$

A balanced block design with $k < v$ is called a *balanced incomplete block design* since each block has fewer than the total number of varieties. Such a design is also called a *BIBD*, a (b, v, r, k, λ)-*design*, or a (b, v, r, k, λ)-*configuration*. The basic ideas behind BIBDs were introduced by Yates [1936]. Note that if $k = v$ and no block has repeated varieties, conditions (9.10), (9.11) and (9.12) hold trivially, with $k = v, r = b$, and $\lambda = b$. That is why we will assume that $k < v$ unless indicated otherwise.

Example 9.10 A (7, 7, 3, 3, 1)-Design If $b = 7, v = 7, r = 3, k = 3$, and $\lambda = 1$, there is a (b, v, r, k, λ)-design. It is given by taking the set of varieties V to be $\{1, 2, 3, 4, 5, 6, 7\}$ and using the following blocks:

$$B_1 = \{1, 2, 4\}, \quad B_2 = \{2, 3, 5\}, \quad B_3 = \{3, 4, 6\},$$

$$B_4 = \{4, 5, 7\}, \quad B_5 = \{5, 6, 1\}, \quad B_6 = \{6, 7, 2\}, \quad B_7 = \{7, 1, 3\}.$$

It is easy to see that each block consists of exactly 3 varieties, that each variety appears in exactly 3 blocks, and that each pair of varieties appears simultaneously in exactly 1 block (e.g., 3 and 6 appear together in B_3 and nowhere else). ■

Example 9.11 A (4, 4, 3, 3, 2)-Design If $b = 4, v = 4, r = 3, k = 3$, and $\lambda = 2$, a (b, v, r, k, λ)-design is given by

$$V = \{1, 2, 3, 4\}$$

and the blocks

$$\{1, 2, 3\},\ \{2, 3, 4\},\ \{3, 4, 1\},\ \{4, 1, 2\}.$$ ■

In the tire wear example of Section 9.1, the varieties are tire brands and the blocks are the sets of tire brands corresponding to the tires used in a given test car. The conditions (9.10), (9.11), and (9.12) correspond to the following reasonable requirements:

each car uses the same number k of tire brands;
each brand appears on the same number r of cars;
each pair of brands is tested together on the same car the same number λ of times.

It should be clear from our experience with orthogonal families of Latin squares that (b, v, r, k, λ)-designs may not exist for all combinations of the parameters b, v, r, k, and λ. Indeed, the basic combinatorial question of the subject of balanced incomplete block designs is the existence question: For what values of b, v, r, k, λ does a (b, v, r, k, λ)-design exist? We address this question below. In general, it is an unsolved problem to state complete conditions on the parameters b, v, r, k, λ necessary and sufficient for the existence of a (b, v, r, k, λ)-design. A typical reference book on practical experimental design will list, for reasonable values of the parameters, those (b, v, r, k, λ)-designs that do exist. For now, let us give some examples of the use of (b, v, r, k, λ)-designs. Then we shall study the basic existence question in some detail.

Example 9.12 Testing Cloth for Wear (Example 9.6 Revisited) Suppose that we have a Martindale wear tester as described in Example 9.6, and we wish to

Table 9.24: A Youden Square for the Wear Testing Experiment[a]

		Block (run)						
		B_1	B_2	B_3	B_4	B_5	B_6	B_7
Variety (cloth type)	1		p		q		r	s
	2	p			s	r		q
	3		q	p		s		r
	4	q		r			s	p
	5			s	r	q	p	
	6	r	s			q	p	
	7	s	r	q	p			

[a]The i, j entry gives the position of variety i in block B_j.

use it to compare seven different types of cloth. Since only four pieces of cloth can be tested in one run of the machine, an incomplete block design must be used. The number v of varieties is 7 and the blocks will all be of size $k = 4$. Box, Hunter, and Hunter [1978] describe a (b, v, r, k, λ)-design for this situation in which there are 7 blocks ($b = 7$), each type of cloth is run $r = 4$ times, and each pair of cloth types is used together in a run $\lambda = 2$ times. If the cloth types are labeled $1, 2, 3, 4, 5, 6, 7$, the blocks used can be described as

$$B_1 = \{2, 4, 6, 7\}, \quad B_2 = \{1, 3, 6, 7\}, \quad B_3 = \{3, 4, 5, 7\},$$

$$B_4 = \{1, 2, 5, 7\}, \quad B_5 = \{2, 3, 5, 6\}, \quad B_6 = \{1, 4, 5, 6\}, \quad B_7 = \{1, 2, 3, 4\}.$$

Since there were four positions in which to place the cloth, and since the design could be chosen so that each cloth type was used in 4 runs, it was also possible to arrange to place each type of cloth exactly once in each position. Thus, it was possible to control for differences due to machine position. Such an incomplete block design which is balanced for 2 different sources of block variation is called a *Youden square*, after its inventor W. J. Youden (see Youden [1937]). In this case, the Youden square can be summarized in Table 9.24, where p, q, r, and s represent the 4 positions, and the i, j entry gives the position of variety i in block B_j. ■

Example 9.13 Tuberculosis in Cattle Wadley [1948] used balanced incomplete block designs in work on diagnosing tuberculosis in cattle. The disease can be diagnosed by injecting the skin of a cow with an appropriate allergen and observing the thickening produced. In an experiment to compare allergens, the observation for each allergen was the log concentration required to produce a 3-millimeter thickening. This concentration was being estimated by observing the thickenings at four different concentrations and interpolating. Thus, each test of an allergen required a number of injections of the allergen at different concentrations.

In Wadley's experiment, 16 allergens were under comparison. Thus, $v = 16$. On each cow there were four main regions, and in each region about 16 injections could be made. This suggests using each region as a block, with four allergen preparations in each block, each used four times at different concentrations, making 16 injections in all. Thus, $k = 4$. There is a design with $k = 4, v = 16, b = 20$, and $r = 5$. This information is available from a typical reference book. Since there are 20 blocks and four blocks per cow, this calls for five cows (or, by repeating the experiment, some multiple of five cows; Wadley's experiment used 10 cows). If no suitable design had been available with $k = 4$ and $v = 16$, it would have been natural to have considered whether five preparations could have been included in each region (20 injections per region). ■

Example 9.14 Comparing Dishwashing Detergents In experiments such as that by Pugh [1953] to compare detergents used for domestic dishwashing, the following procedure has been used. To obtain a series of homogeneous experimental units, a pile of plates from one course in a canteen is divided into groups. Each group of plates is then washed with water at a standard temperature and with a controlled amount of one detergent. The experimenter records the (logarithm of the) number of plates washed before the foam is reduced to a thin surface layer. The detergents form the varieties. Each group of plates from the one course forms an experimental unit and the different groups of plates from the same course form a block. The washing for one block is done by one person. Each group of plates within a course is assigned to a variety. The experimental conditions are as constant as possible within one block. Different blocks consist of plates soiled in different ways and washed by different people.

Now the number of plates available in one block is limited. It frequently allows only four tests to be completed; that is, there is a restriction to four experimental units and hence varieties per block. If eight varieties are to be compared, not every variety can be tried out in every block. This calls for an incomplete block design, with $v = 8, k = 4$. There is such a design with $r = 7$ and $b = 14$.

In sum, the experimenter takes a set of dishes from a given course, divides it into four groups, and applies a different detergent to each of the groups. The experiment is repeated 14 times, each time with a collection of four detergents chosen to make up the four varieties in the corresponding block. ■

9.4.2 Necessary Conditions for the Existence of (b, v, r, k, λ)-Designs

Our first theorem states some necessary conditions that the parameters for a balanced incomplete block design must satisfy.

Theorem 9.11 In a (b, v, r, k, λ)-design,

$$bk = vr \tag{9.13}$$

and

$$r(k - 1) = \lambda(v - 1). \tag{9.14}$$

To illustrate this theorem, we note that no $(12, 9, 4, 3, 2)$-design exists, for $bk = 36, vr = 36, r(k-1) = 8$, and $\lambda(v-1) = 16$. Hence, although (9.13) is satisfied, (9.14) is not. If $b = 12, v = 9, r = 4, k = 3$, and $\lambda = 1$, $bk = vr = 36$, and $r(k-1) = \lambda(v-1) = 8$, so (9.13) and (9.14) are satisfied. This says that a $(12, 9, 4, 3, 1)$-design *could* exist; it does not guarantee that such a design *does* exist. [Conditions (9.13) and (9.14) are necessary, but not sufficient.]

Proof of Theorem 9.11. The product bk is the product of the number of blocks (b) by the number of varieties in each block (k), and hence gives the total number of elements that are listed if the blocks are written out as

$$B_1: \quad \ldots \quad B_2: \quad \ldots \quad \cdots \quad B_b: \quad \ldots$$

The product vr is the product of the number of varieties (v) by the number of replications of each variety (r) and hence also gives the number of elements listed above. Hence, $bk = vr$, and (9.13) holds.

The product $r(k-1)$ is the product of the number of blocks in which a variety i appears (r) by the number of other varieties in each block in which i appears, and hence gives the number of pairs $\{i, j\}$ appearing in a common block (counting a pair once for each time it occurs). The product $\lambda(v-1)$ is the product of the number of times each pair $\{i, j\}$ appears in a block (λ) by the number of possible j's $(v-1)$, and hence gives the number of pairs $\{i, j\}$ appearing in a common block (counting a pair once for each time it occurs). Thus, $r(k-1) = \lambda(v-1)$, and (9.14) holds. Q.E.D.

Corollary 9.11.1 Suppose that in an incomplete block design with $v \geq 2$ varieties and b blocks,

1. each block consists of the same number k of varieties, and
2. each pair of varieties appears simultaneously in exactly the same number λ of blocks, $\lambda > 0$.

Then each variety appears in the same number r of blocks, r is given by $\lambda(v-1)/(k-1)$, and the block design is balanced.

Proof. The proof of (9.14) above actually shows this. For suppose that a given variety i appears in r_i blocks. The proof above shows that

$$r_i(k-1) = \lambda(v-1).$$

Note that since $v \geq 2$ and $\lambda > 0, k$ could not be 1. Thus,

$$r_i = \frac{\lambda(v-1)}{k-1}.$$

This is the same number r_i for each i. Q.E.D.

Corollary 9.11.1 shows that the definition of balanced incomplete block design is redundant: One of the conditions in the definition, namely the condition (9.11) that every variety appears in the same number of blocks, follows from the other conditions, (9.10) and (9.12).

Theorem 9.12 (Fisher's Inequality[14]) In a (b, v, r, k, λ)-design, $b \geq v$.

We shall prove this result in Section 9.4.3. To prove it, it will be helpful to introduce a concept that will also be very useful in our study of error-correcting codes in Chapter 10. This is the notion of an *incidence matrix* A of a block design. If the design has varieties $x_1, x_2, \ldots, x_v$, and blocks $B_1, B_2, \ldots, B_b$, then A is a $v \times b$ matrix of 0's and 1's, with the i, j entry of A being 1 if x_i is in B_j and 0 otherwise. (This is the point-set incidence matrix of Section 3.7.) For example, in the (b, v, r, k, λ)-design of Example 9.10, we have the following incidence matrix:

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} B_1 & B_2 & B_3 & B_4 & B_5 & B_6 & B_7 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Designs can be defined by giving $v \times b$ matrices of 0's and 1's. An arbitrary $v \times b$ matrix of 0's and 1's with $v \geq 2$ is the incidence matrix of a (b, v, r, k, λ)-design, $b, v, r, k, \lambda > 0$, if and only if the following conditions hold:

$$\text{each column has the same number of 1's, } k \text{ of them, } k > 0; \tag{9.15}$$

$$\text{each row has the same number of 1's, } r \text{ of them, } r > 0; \tag{9.16}$$

$$\begin{array}{c}\text{each pair of rows has the same number of columns} \\ \text{with a common 1, } \lambda \text{ of them, } \lambda > 0.\end{array} \tag{9.17}$$

We have seen (in Corollary 9.11.1) that (9.15) and (9.17) imply (9.16). A natural analog of (9.17) is the following:

$$\text{each pair of columns has the same number of rows with a common 1.} \tag{9.18}$$

Exercise 48 investigates the relations among conditions (9.15)–(9.18).

9.4.3 Proof of Fisher's Inequality[15]

To prove Fisher's Inequality, we shall first prove a result about incidence matrices of (b, v, r, k, λ)-designs.

[14] This theorem is due to Fisher [1940].
[15] This subsection may be omitted.

Theorem 9.13 If $\mathbf{A}$ is the incidence matrix of a (b, v, r, k, λ)-design, then

$$\mathbf{A}\mathbf{A}^T = (r - \lambda)\mathbf{I} + \lambda\mathbf{J}, \tag{9.19}$$

where $\mathbf{A}^T$ is the transpose of $\mathbf{A}$, $\mathbf{I}$ is a $v \times v$ identity matrix, and $\mathbf{J}$ is the $v \times v$ matrix of all 1's.

Proof. Let b_{ij} be the i, j entry of $\mathbf{A}\mathbf{A}^T$. Then b_{ij} is the *inner product* of the ith row of $\mathbf{A}$ with the jth row of $\mathbf{A}$, that is,

$$b_{ij} = \sum_{k=1}^{b} a_{ik}a_{jk}.$$

If $i = j$, we see that $a_{ik}a_{jk}$ is 1 if the ith variety belongs to the kth block, and it is 0 otherwise. Thus, b_{ii} counts the number of blocks that i belongs to, that is, r. If $i \neq j$, then $a_{ik}a_{jk}$ is 1 if the ith and jth varieties both belong to the kth block, and it is 0 otherwise. Thus, b_{ij} counts the number of blocks that the ith and jth varieties both belong to, that is, λ. Translating these conclusions into matrix language gives us (9.19). Q.E.D.

Proof of Fisher's Inequality (Theorem 9.12). We shall suppose that $b < v$ and reach a contradiction. Let $\mathbf{A}$ be the incidence matrix. Since $b < v$, we can add $v - b$ columns of 0's to $\mathbf{A}$, giving us a square $v \times v$ matrix $\mathbf{B}$. Now $\mathbf{A}\mathbf{A}^T = \mathbf{B}\mathbf{B}^T$, since the inner product of two rows of $\mathbf{A}$ is the same as the inner product of two rows of $\mathbf{B}$. Taking determinants, we conclude that

$$\det(\mathbf{A}\mathbf{A}^T) = \det(\mathbf{B}\mathbf{B}^T) = (\det \mathbf{B})(\det \mathbf{B}^T).$$

But $\det \mathbf{B} = 0$ since $\mathbf{B}$ has a column of 0's. Thus, $\det(\mathbf{A}\mathbf{A}^T) = 0$. Now by Theorem 9.13,

$$\det(\mathbf{A}\mathbf{A}^T) = \det \begin{bmatrix} r & \lambda & \lambda & \lambda & \cdots & \lambda \\ \lambda & r & \lambda & \lambda & \cdots & \lambda \\ \lambda & \lambda & r & \lambda & \cdots & \lambda \\ \lambda & \lambda & \lambda & r & \cdots & \lambda \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \lambda & \lambda & \lambda & \lambda & \cdots & r \end{bmatrix}. \tag{9.20}$$

Subtracting the first column from each of the others in the matrix in the right-hand side of (9.20) does not change the determinant. Hence,

$$\det(\mathbf{A}\mathbf{A}^T) = \det \begin{bmatrix} r & \lambda - r & \lambda - r & \lambda - r & \cdots & \lambda - r \\ \lambda & r - \lambda & 0 & 0 & \cdots & 0 \\ \lambda & 0 & r - \lambda & 0 & \cdots & 0 \\ \lambda & 0 & 0 & r - \lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \lambda & 0 & 0 & 0 & \cdots & 0 \\ \lambda & 0 & 0 & 0 & \cdots & r - \lambda \end{bmatrix}. \tag{9.21}$$

Adding to the first row of the matrix on the right-hand side of (9.21) all the other rows does not change the determinant. Hence,

$$\det(\mathbf{A}\mathbf{A}^T) = \det \begin{bmatrix} r+(v-1)\lambda & 0 & 0 & 0 & \cdots & 0 \\ \lambda & r-\lambda & 0 & 0 & \cdots & 0 \\ \lambda & 0 & r-\lambda & 0 & \cdots & 0 \\ \lambda & 0 & 0 & r-\lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \lambda & 0 & 0 & 0 & \cdots & 0 \\ \lambda & 0 & 0 & 0 & \cdots & r-\lambda \end{bmatrix}. \tag{9.22}$$

Since the matrix in the right-hand side of (9.22) has all 0's above the diagonal, its determinant is the product of the diagonal elements, so

$$\det(\mathbf{A}\mathbf{A}^T) = [r+(v-1)\lambda](r-\lambda)^{v-1}.$$

Since we have concluded that $\det(\mathbf{A}\mathbf{A}^T) = 0$, we have

$$[r+(v-1)\lambda](r-\lambda)^{v-1} = 0. \tag{9.23}$$

But since r, v, and λ are all assumed positive,

$$[r+(v-1)\lambda] > 0.$$

Also, by Equation (9.14) of Theorem 9.11, since $k < v$, it follows that $r > \lambda$. Hence,

$$(r-\lambda)^{v-1} > 0$$

We conclude that the left-hand side of (9.23) is positive, which is a contradiction. Q.E.D.

9.4.4 Resolvable Designs

We say that a (b, v, r, k, λ)-design is *resolvable* if the blocks can be partitioned into groups, called *parallel classes*, so that the blocks in each parallel class in turn partition the set of varieties. For example, Table 9.25 shows a $(12, 9, 4, 3, 1)$-design that consists of four parallel classes. Note that in each parallel class, the three blocks are disjoint and their union is $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Example 9.15 Anonymous Threshold Schemes and Secret Sharing[16] In Example 9.9 we introduced (q, p)-threshold schemes for secret sharing. In an *anonymous (q, p)-threshold scheme*, the p persons receive p distinct partial pieces of information and the secret key can be computed from any q of the partial pieces without knowing which person holds which piece. The threshold schemes we have constructed from orthogonal arrays in Example 9.9 are not anonymous (see Exercise 13). We shall see how resolvable (b, v, r, k, λ)-designs can help us find anonymous (q, p)-threshold schemes, in particular anonymous $(2, p)$-threshold schemes.

[16] This example is due to Stinson [2003].

Table 9.25: A $(12, 9, 4, 3, 1)$-Design with Parallel Classes C_1, C_2, C_3, C_4

$$\begin{array}{l} C_1 : \{1,2,3\}, \{4,5,6\}, \{7,8,9\} \\ C_2 : \{1,4,7\}, \{2,5,8\}, \{3,6,9\} \\ C_3 : \{1,5,9\}, \{2,6,7\}, \{3,4,8\} \\ C_4 : \{1,6,8\}, \{2,4,9\}, \{3,5,7\} \end{array}$$

Suppose that we have a resolvable (b, v, r, k, λ)-design with $\lambda = 1$ and $k = p$ varieties per block. Let C_1, C_2, ..., C_r be the parallel classes. Thus, each parallel class has v/p blocks. Each variety appears once per parallel class, so the number of parallel classes can be computed from Equation (9.14):

$$r = \frac{\lambda(v-1)}{p-1} = \frac{(v-1)}{p-1}.$$

Suppose that the parallel classes are made known to all p persons in the group. Let us take the set K of possible keys to be $\{1, 2, \ldots, r\}$ and the set P of pieces of partial information to be V, the set of varieties. Suppose that the leader wants to share the secret key $\kappa \in K$. Then he chooses a block from parallel class C_κ at random and gives the p pieces of partial information from this block to the p persons, one per participant. Note that any two persons can now identify the secret key κ. For if p_u is the partial information given to person u and p_v is the partial information given to person v, then since $\lambda = 1$, there is a unique block in the design that contains the varieties p_u and p_v. Persons u and v can find that block and know that the secret key is κ if the block containing p_u and p_v is in parallel class C_κ. Note that this is anonymous since it does not matter which person holds which key, just which keys are held by the pair of persons.

What is the probability that any one person u can determine the secret key given his or her partial information p_u? Every parallel class has exactly one block containing p_u. Thus, the probability of correctly guessing which secret key/parallel class the leader had in mind is $1/r$, where r is the number of parallel classes. This is exactly the same as the probability of guessing right given no partial information. If r is large, we have a very secure anonymous $(2, p)$-threshold scheme.

For example, the resolvable $(12, 9, 4, 3, 1)$-design of Table 9.25 can be used to build an anonymous $(2, 3)$-threshold scheme. If the leader wants to share secret key 3, he or she picks a random block in C_3, say $\{3, 4, 8\}$. These pieces of partial information are given to the three persons in the group. Those getting 3 and 8, for example, know that the only block that the leader could have had in mind is $\{3, 4, 8\}$, which is in C_3, and so know that the secret key is 3. ■

9.4.5 Steiner Triple Systems

So far our results have given necessary conditions for the existence of (b, v, r, k, λ)-designs, but have not given us sufficient conditions for their existence, or constructive procedures for finding them. We shall describe several such procedures. We begin by considering special cases of (b, v, r, k, λ)-designs.

In particular, suppose that $k = 2$ and $\lambda = 1$. In this case, each block consists of two varieties and each pair of varieties appears in exactly one block. Equation (9.14) implies that $r = v - 1$, so (9.13) implies that

$$2b = v(v-1)$$

or

$$b = \frac{v(v-1)}{2}.$$

Now

$$\frac{v(v-1)}{2} = \binom{v}{2}$$

is the number of two-element subsets of a set of v elements. Hence, the number of blocks is the number of two-element subsets of the set of varieties. If, for example, $v = 3$, such a design with $V = \{1, 2, 3\}$ has as blocks the subsets

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

In this subsection we concentrate on another special case of (b, v, r, k, λ)-designs, that where $k = 3$ and $\lambda = 1$. Such a design is a set of triples in which each pair of varieties appears in exactly one triple. These designs are called *Steiner triple systems*. Some authors define Steiner triple systems as block designs in which the blocks are triples from a set V and each pair of varieties appears in exactly one triple. This definition allows inclusion of the complete block design where $k = v$. This is the trivial design where $V = \{1, 2, 3\}$ and there is only one block, $\{1, 2, 3\}$. For the purposes of this subsection, we shall include this design as a Steiner triple system. A more interesting example of a Steiner triple system occurs when $v = 7$. Example 9.10, a $(7, 7, 3, 3, 1)$-design, is such an example.

We shall now discuss the existence problem for Steiner triple systems. Note that in a Steiner triple system, (9.14) implies that

$$r(2) = v - 1, \tag{9.24}$$

so

$$r = \frac{v-1}{2}. \tag{9.25}$$

Equation (9.13) now implies that

$$3b = \frac{v(v-1)}{2},$$

so

$$b = \frac{v(v-1)}{6}. \tag{9.26}$$

Equation (9.25) implies that $v - 1$ is even and v is odd. Also, $v \geq 2$ implies that v is at least 3. Equation (9.26) implies that $v(v-1) = 6b$, so $v(v-1)$ is a multiple of

6. These are necessary conditions. Let us begin to tabulate what values of v satisfy the two necessary conditions: v odd and at least 3, $v(v-1)$ a multiple of 6. If $v = 3$, then $v(v-1) = 6$, so there could be a Steiner triple system with $v = 3$; that is, the necessary conditions are satisfied. However, with $v = 5$, $v(v-1) = 20$, which is not divisible by 6, so there is no Steiner triple system with $v = 5$. In general, Steiner triple systems are possible for $v = 3, 7, 9, 13, 15, 19, 21, \ldots$, that is, for $v = 6n+1$ or $6n+3, n \geq 1$, and $v = 3$. In fact, these systems do exist for all of these values of v.

Theorem 9.14 (Kirkman [1847]) There is a Steiner triple system of v varieties if and only if $v = 3$ or $v = 6n+1$ or $v = 6n+3, n \geq 1$.

We have already proved the necessity of the conditions in Theorem 9.14. Rather than prove sufficiency, we shall prove a simpler theorem, which gives us the existence of some of these Steiner triple systems: for instance, those with $3 \cdot 3 = 9$ varieties, $3 \cdot 7 = 21$ varieties, $7 \cdot 7 = 49$ varieties, $9 \cdot 7 = 63$ varieties, and so on. For a number of proofs of sufficiency, see, for instance, Lindner and Rodger [1997].

Theorem 9.15 If there is a Steiner triple system S_1 of v_1 varieties and a Steiner triple system S_2 of v_2 varieties, then there is a Steiner triple system S of v_1v_2 varieties.

Proof.[17] The proof provides a construction for building a Steiner triple system S given Steiner triple systems S_1 and S_2. Suppose that the varieties of S_1 are $a_1, a_2, \ldots, a_{v_1}$ and those of S_2 are $b_1, b_2, \ldots, b_{v_2}$. Let S consist of the v_1v_2 elements $C_{ij}, i = 1, 2, \ldots, v_1, j = 1, 2, \ldots, v_2$. A triple $\{c_{ir}, c_{js}, c_{kt}\}$ is in S if and only if one of the following conditions holds:

(1) $r = s = t$ and $\{a_i, a_j, a_k\} \in S_1$,
(2) $i = j = k$ and $\{b_r, b_s, b_t\} \in S_2$,

or

(3) $\{a_i, a_j, a_k\} \in S_1$ and $\{b_r, b_s, b_t\} \in S_2$.

Then it is easy to prove that S forms a Steiner triple system. Q.E.D.

Let us illustrate the construction in the proof of Theorem 9.15. Suppose that $v_1 = v_2 = 3$, S_1 has the one triple $\{a_1, a_2, a_3\}$ and S_2 has the one triple $\{b_1, b_2, b_3\}$. Then S has the triples shown in Table 9.26 and forms a Steiner triple system of 9 varieties and 12 blocks.

If an experimental design is to be a Steiner triple system on v varieties, the specific choice of design is simple if $v = 3, 7$, or 9, for there is (up to relabeling of varieties) only one Steiner triple system of v varieties in these cases. However, for $v = 13$ there are two essentially different Steiner triple systems, for $v = 15$ there are 80, and for $v = 19$ there are 11,084,874,829 (Kaski and Östergård [2004]). Presumably, one of these will be chosen at random if a Steiner triple system of 13, 15, or 19 varieties is required. In general, when there exist Steiner triple systems for $v > 19$, the number of such distinct Steiner triple systems is unknown.

[17] The proof and the illustration of it may be omitted.

Table 9.26: Construction of a Steiner Triple System S of $v_1v_2 = 9$ Varieties from S_1, S_2 If S_1 Has Only the Triple $\{a_1, a_2, a_3\}$ and S_2 Only the Triple $\{b_1, b_2, b_3\}$

Condition (1) from the proof of Theorem 9.15:	$r = s = t = 1$ $\{c_{11}, c_{21}, c_{31}\}$	$r = s = t = 2$ $\{c_{12}, c_{22}, c_{32}\}$	$r = s = t = 3$ $\{c_{13}, c_{23}, c_{33}\}$
Condition (2) from the proof of Theorem 9.15:	$i = j = k = 1$ $\{c_{11}, c_{12}, c_{13}\}$	$i = j = k = 2$ $\{c_{21}, c_{22}, c_{23}\}$	$i = j = k = 3$ $\{c_{31}, c_{32}, c_{33}\}$
Condition (3) from the proof of Theorem 9.15:	$\{c_{11}, c_{22}, c_{33}\}$ $\{c_{11}, c_{23}, c_{32}\}$	$\{c_{12}, c_{23}, c_{31}\}$ $\{c_{12}, c_{21}, c_{33}\}$	$\{c_{13}, c_{21}, c_{32}\}$ $\{c_{13}, c_{22}, c_{31}\}$

9.4.6 Symmetric Balanced Incomplete Block Designs

A balanced incomplete block design or (b, v, r, k, λ)-design is called *symmetric* if $b = v$ (the number of blocks is the same as the number of varieties) and if $r = k$ (the number of times a variety occurs is the same as the number of varieties in a block). A symmetric BIBD is sometimes called a (v, k, λ)-*design* or a (v, k, λ)-*configuration*. By Equation (9.13) of Theorem 9.11,

$$b = v \quad \text{iff} \quad k = r.$$

Hence, the two conditions in the definition are redundant. Example 9.11 is an example of a symmetric BIBD: We have $b = v = 4$ and $r = k = 3$. So is Example 9.10: We have $b = v = 7$ and $r = k = 3$. The Steiner triple system of Table 9.26 is an example of a BIBD that is not symmetric.

Theorem 9.16 (Bruck-Ryser-Chowla Theorem[18]) The following conditions are necessary for the existence of a (v, k, λ)-design:

1. If v is even, then $k - \lambda$ is the square of an integer.
2. If v is odd, the following equation has a solution in integers x, y, z, not all of which are 0:

$$x^2 = (k - \lambda)y^2 + (-1)^{(v-1)/2}\lambda z^2. \tag{9.27}$$

We omit the proof of Theorem 9.16. For a proof, see Ryser [1963] or Hall [1967]. To illustrate the theorem, suppose that $v = 16, k = 6$, and $\lambda = 2$. Then v is even and $k - \lambda = 4$ is a square, so condition 1 says that a $(16, 6, 2)$-design *could* exist. However, it also implies that a $(22, 7, 2)$-design *could not* exist, since $k - \lambda = 5$ is not a square. Suppose that $v = 111, k = 11$, and $\lambda = 1$. Then v is odd and (9.27) becomes

$$x^2 = 10y^2 - z^2.$$

This has a solution $x = y = 1, z = 3$. Hence, a $(111, 11, 1)$-design *could* exist.

[18] This theorem was proved for $\lambda = 1$ by Bruck and Ryser [1949] and in generality by Chowla and Ryser [1950].

The conditions for existence of a symmetric BIBD given in Theorem 9.16 are not sufficient. Even though we have shown that a $(111, 11, 1)$-design could exist, Lam, Thiel, and Swiercz [1989] proved that one does not. Some specific sufficient conditions are given by the following theorem, whose proof we leave to Section 10.5.2.[19]

Theorem 9.17 For arbitrarily large values of m, and in particular for $m = 2^k, k \geq 1$, there is a $(4m-1, 2m-1, m-1)$-design.

A $(4m-1, 2m-1, m-1)$-design is called a *Hadamard design of dimension* m. The case $m = 2$ gives a $(7, 3, 1)$-design, an example of which is given in Example 9.10. That Hadamard designs of dimension m *could* exist for all $m \geq 2$ follows from Theorem 9.16. For $v = 4m - 1$ is odd and (9.27) becomes

$$x^2 = my^2 - (m-1)z^2,$$

which has the solution $x = y = z = 1$. Hadamard designs will be very important in the theory of error-correcting codes in Section 10.5.

A second theorem giving sufficient conditions for the existence of symmetric BIBDs is the following, which is proved in Section 9.5.2 in our study of projective planes.

Theorem 9.18 If $m \geq 1$ is a power of a prime, there is an $(m^2+m+1, m+1, 1)$-design.

To illustrate this theorem, note that taking $m = 1$ gives us a $(3, 2, 1)$-design. We have seen such a design at the beginning of Section 9.4.5. Taking $m = 2$ gives us a $(7, 3, 1)$-design. We have seen such a design in Example 9.10. Taking $m = 3$ gives us a $(13, 4, 1)$-design, which is something new.

Still a third way to construct symmetric BIBDs is to use difference sets. This method is described in Exercises 31 and 32.

9.4.7 Building New (b, v, r, k, λ)-Designs from Existing Ones

Theorem 9.15 gives us a way of building new (b, v, r, k, λ)-designs from old ones. Here we shall present other such ways. The most trivial way to get one design from another is simply to repeat blocks. If we take p copies of each block in a (b, v, r, k, λ)-design, we get a $(pb, v, pr, k, p\lambda)$-design. For example, from the $(4, 4, 3, 3, 2)$-design of Example 9.11, we get an $(8, 4, 6, 3, 4)$-design by repeating each block twice. To describe more interesting methods of obtaining new designs from old ones, we need one preliminary result.

Theorem 9.19 In a (v, k, λ)-design, any two blocks have exactly λ elements in common.

Proof. Exercise 48. Q.E.D.

[19] That section may be read at this point.

If U and V are sets, $U - V$ will denote the set $U \cap V^c$.

Theorem 9.20 Suppose that $B_1, B_2, \ldots, B_v$ are the blocks of a (v, k, λ)-design with $V = \{x_1, x_2, \ldots, x_v\}$ the set of varieties. Then for any i,

$$B_1 - B_i, B_2 - B_i, \ldots, B_{i-1} - B_i, B_{i+1} - B_i, \ldots, B_v - B_i$$

are the blocks of a $(v - 1, v - k, k, k - \lambda, \lambda)$-design on the set of varieties $V - B_i$.

Proof. There are clearly $v-1$ blocks and $v-k$ varieties. By Theorem 9.19, each block $B_j - B_i$ has $k - \lambda$ elements. Each variety in $V - B_i$ appears in k blocks of the original design and hence in k blocks of the new design. Similarly, each pair of varieties in $V - B_i$ appear in common in λ blocks of the original design and hence in λ blocks of the new design. Q.E.D.

To illustrate this construction, suppose that we start with the $(7, 3, 1)$-design of Example 9.10 and let $B_i = \{3, 4, 6\}$. Then the following blocks form a $(6, 4, 3, 2, 1)$-design on the set of varieties $\{1, 2, 5, 7\}$:

$$\{1, 2\}, \{2, 5\}, \{5, 7\}, \{1, 5\}, \{2, 7\}, \{1, 7\}.$$

Theorem 9.21 Suppose that $B_1, B_2, \ldots, B_v$ are the blocks of a (v, k, λ)-design with $V = \{x_1, x_2, \ldots, x_v\}$ the set of varieties. Then for any i,

$$B_1 \cap B_i, B_2 \cap B_i, \ldots, B_{i-1} \cap B_i, B_{i+1} \cap B_i, \ldots, B_v \cap B_i$$

are the blocks of a $(v - 1, k, k - 1, \lambda, \lambda - 1)$-design on the set of varieties B_i.

Proof. There are clearly $v - 1$ blocks and k varieties. By Theorem 9.19, each block $B_j \cap B_i$ has λ elements. Moreover, a given variety in B_i appears in the original design in blocks

$$B_{j_1}, \quad B_{j_2}, \quad \ldots, \quad B_{j_{k-1}}, \quad B_i.$$

Then it appears in the new design in $k - 1$ blocks,

$$B_{j_1} \cap B_i, \quad B_{j_2} \cap B_i, \quad \ldots, \quad B_{j_{k-1}} \cap B_i.$$

Moreover, any pair of varieties in B_i appear in common in the original design in λ blocks,

$$B_{j_1}, \quad B_{j_2}, \quad \ldots, \quad B_{j_{\lambda-1}}, \quad B_i,$$

and hence appear in common in the new design in $\lambda - 1$ blocks,

$$B_{j_1} \cap B_i, \quad B_{j_2} \cap B_i, \quad \ldots, \quad B_{j_{\lambda-1}} \cap B_i.$$ Q.E.D.

To illustrate this theorem, we note that by Theorem 9.17, there is a $(15, 7, 3)$-design. Hence, Theorem 9.21 implies that there is a $(14, 7, 6, 3, 2)$-design. To exhibit such a design, we note that the blocks in Table 9.27 define a $(15, 7, 3)$-design on $V = \{1, 2, \ldots, 15\}$. We show how to construct this design in Section 10.5.2. Taking $B_i = \{1, 2, 3, 8, 9, 10, 11\}$, we get the $(14, 7, 6, 3, 2)$-design of Table 9.28 on the set of varieties B_i. Note that this design has repeated blocks. If we take only one copy of each of these blocks, we get a $(7, 3, 1)$-design.

Table 9.27: The Blocks of a $(15, 7, 3)$-Design on the Set of Varieties $V = \{1, 2, \ldots, 15\}$

{2, 4, 6, 8, 10, 12, 14},	{1, 4, 5, 8, 9, 12, 13},	{3, 4, 7, 8, 11, 12, 15},
{1, 2, 3, 8, 9, 10, 11},	{2, 5, 7, 8, 10, 13, 15},	{1, 6, 7, 8, 9, 14, 15},
{3, 5, 6, 8, 11, 13, 14},	{1, 2, 3, 4, 5, 6, 7},	{2, 4, 6, 9, 11, 13, 15},
{1, 4, 5, 10, 11, 14, 15},	{3, 4, 7, 9, 10, 13, 14},	{1, 2, 3, 12, 13, 14, 15}
{2, 5, 7, 9, 11, 12, 14}	{1, 6, 7, 10, 11, 12, 13},	{3, 5, 6, 9, 10, 12, 15}

Table 9.28: The Blocks of a $(14, 7, 6, 3, 2)$-Design Obtained from the Design of Table 9.27 by Intersecting Blocks with $B_i = \{1, 2, 3, 8, 9, 10, 11\}$

{2, 8, 10},	{1, 8, 9},	{3, 8, 11},	{2, 8, 10},	{1, 8, 9},	{3, 8, 11},	{1, 2, 3}
{2, 9, 11},	{1, 10, 11},	{3, 9, 10},	{1, 2, 3},	{2, 9, 11},	{1, 10, 11},	{3, 9, 10}

9.4.8 Group Testing and Its Applications

Suppose that a large population U of items is partitioned into two classes, positive and negative. We wish to locate positive items, but to examine each item in U is prohibitively expensive. However, we can group the items into subsets of U and can test if a subset contains at least one positive item. We would like to identify all positive items through a number of group tests. If we have to identify all groups to test and then carry out the group tests without being able to use the results of these tests to select new groups to test, we talk about *nonadaptive group testing*. Otherwise, we talk about *adaptive group testing*. The modern theory of group testing is heavily influenced by combinatorial methods, in particular by the methods of combinatorial designs.

Here are some examples of the uses of group testing.

(1) Defective Products. We want to pick out those items manufactured in a given plant that are defective before shipping them. (Here, defective items are "positive.")

(2) Screening for Diseases. We want to determine which persons in a group have a certain disease. It was this problem (in connection with testing millions of military draftees for syphilis) that gave rise to the study of group testing by Dorfman [1943]. The subject has become very important with the possibility of large-scale HIV screening. (Here, having the disease is "positive.")

(3) Mapping Genomes. We have a long molecular sequence S, e.g., DNA. We form a library of substrings known as *clones*. We test whether or not a particular sequence, known as a *probe*, appears in S by testing to see in which clones it appears. We do this by pooling the clones into groups, since a clone library can have thousands, even hundreds of thousands, of clones.

(4) Satellite Communications. Many ground stations are potential users of a satellite communications link. In scheduling requests for time slots in the satellite link, one doesn't contact all of the ground stations individually, but instead, pools of ground stations are contacted to see if a station in the pool wishes to reserve satellite time.

(5) Scientific or Industrial Experiments. We want to determine which of a large number of possible variables are important. If we can assume that the effect of an important variable is strong enough not to be masked by other variables, we can study them first in groups to identify the important ones.

Information about these and other applications of group testing, the theory of group testing, as well as references to the literature, can be found in the book by Du and Hwang [2000]. See also Colbourn, Dinitz, and Stinson [1999].

In this section we illustrate the connection between nonadaptive group testing and (b,v,r,k,λ)-designs. A *nonadaptive group testing algorithm* or *NAGTA* starts with a population U of u elements. It seeks to identify those items in U that belong to a subset of positive items. The algorithm uses a collection G of g subsets of U called *groups*. We order the subsets in G. The data of our group testing can be reported as a vector v of 0's and 1's whose ith entry is 1 if and only if the ith group tests positive. For each group $X \subseteq U$ in G and each subset $P \subseteq U$, let $f_X(P)$ give the outcome 1 if $X \cap P \neq \emptyset$ and 0 otherwise. The bit string $(f_{X_1}(P), f_{X_2}(P), \ldots, f_{X_g}(P))$ corresponding to all the groups X in G is denoted by $f_G(P)$. Suppose we can find the collection G so that for all subsets $Q \neq P$ of U of size at most t, $f_G(Q) \neq f_G(P)$. If this is the case, then, as long as the number of positive items is at most t, we can tell from our g group tests exactly which items in U are positive by seeing for which subset P the vector $f_G(P)$ matches the observed vector v. To give an example, suppose that $U = \{1,2,3,4\}$ and G is given by the ordered sequence

$$G = (\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}). \tag{9.28}$$

Then

$$f_G(\{2\}) = (1,0,0,1,1,0)$$

and

$$f_G(\{1,2\}) = (1,1,1,1,1,0).$$

In Table 9.29, the vectors $f_G(P)$ are given for all P with $|P| \leq 2$. So $f_G(\{2\}) \neq f_G(\{1,2\})$, and, clearly, $f_G(Q) \neq f_G(P)$ for all $Q \neq P$ with $|P| \leq 2$, $|Q| \leq 2$. Thus, the result of the six group tests in G uniquely identifies the subset P of positive items. Suppose again that $U = \{1,2,3,4\}$. The collection

$$G' = (\{1,2\},\{1,3\},\{1,4\})$$

Table 9.29: $f_G(P)$ for All P with $|P| \leq 2$, $U = \{1,2,3,4\}$, and G from (9.28)

P	$f_G(P)$
$\{1\}$	$(1,1,1,0,0,0)$
$\{2\}$	$(1,0,0,1,1,0)$
$\{3\}$	$(0,1,0,1,0,1)$
$\{4\}$	$(0,0,1,0,1,1)$
$\{1,2\}$	$(1,1,1,1,1,0)$
$\{1,3\}$	$(1,1,1,1,0,1)$
$\{1,4\}$	$(1,1,1,0,1,1)$
$\{2,3\}$	$(1,1,0,1,1,1)$
$\{2,4\}$	$(1,0,1,1,1,1)$
$\{3,4\}$	$(0,1,1,1,1,1)$

has $f_{G'}(\{1\}) = f_{G'}(\{1,2\}) = (1,1,1)$, but it has $f_{G'}(\{i\}) \neq f_{G'}(\{j\})$ for all $i \neq j$. We say that G defines a *successful NAGTA with threshold* t if $f_G(Q) \neq f_G(P)$ for all Q and P with at most t elements (and $P \neq Q$). Thus, G' is a successful NAGTA with threshold 1, but not with threshold 2. In other words, the collection of groups G' can be used to determine the positive items if there is exactly one but not if there are two or more.

We now show how to construct successful NAGTAs with given threshold t if we have a population U of size u.[20] Start with a (b,v,r,k,λ)-design with $b = u$, $k = t+1$, and $\lambda = 1$. Let the b blocks correspond to the elements of the population U. Let $V = \{1,2,\ldots,v\}$ be the set of varieties and define the ith group X_i in G to be the set of blocks containing variety i. Then:

- Each element of U (block of the design) is in exactly k groups.
- Each group has exactly r items from U.
- Each pair of distinct items in U is contained in at most one group (since $\lambda = 1$).

The reader may find it useful to think of the pair (U, G) as defined by the transpose of the incidence matrix of the (b,v,r,k,λ)-design.[21]

We shall prove that G has threshold $t = k-1$. First, to illustrate the construction, suppose that we start with the $(7,7,3,3,1)$-design of Example 9.10. Then the population U is $\{B_1,B_2,B_3,B_4,B_5,B_6,B_7\}$ and the groups are given by

$$X_1 = \{B_1,B_5,B_7\},\ X_2 = \{B_1,B_2,B_6\},\ X_3 = \{B_2,B_3,B_7\},\ X_4 = \{B_1,B_3,B_4\},$$
$$X_5 = \{B_2,B_4,B_5\},\ X_6 = \{B_3,B_5,B_6\},\ X_7 = \{B_4,B_6,B_7\}.$$

[20] This construction follows Stinson [2003].

[21] The design whose incidence matrix is the transpose of the incidence matrix of a known BIBD is called the *dual* of that design.

Now, suppose that we know that $f_G(P) = (1,1,1,1,0,1,0)$. The reader can check that $f_G(\{B_1, B_3\}) = (1,1,1,1,0,1,0)$. Since we have a threshold of $k-1=2$, we know that P must be $\{B_1, B_3\}$.

We conclude by proving that the threshold of G is $k-1$. Suppose that $|P| \leq k-1$, $|Q| \leq k-1$, and $P \neq Q$. We show that $f_{X_j}(P) \neq f_{X_j}(Q)$ for some j. Without loss of generality, there is a block B in Q but not in P. Now B is in exactly k groups $X_{j_1}, X_{j_2}, \ldots, X_{j_k}$. If $X_{j_i} \cap P \neq \emptyset$ for all $i = 1, 2, \ldots, k$, then for every i there is a block $B(i) \in X_{j_i} \cap P$. Now $B(i) \neq B(i')$ if $i \neq i'$ since B and $B(i)$ are in at most one group. But then P has at least k elements, $B(1), B(2), \ldots, B(k)$, contradicting $|P| \leq k-1$. Thus, for some i with $1 \leq i \leq k$, $X_{j_i} \cap P = \emptyset$. Hence, $f_{X_{j_i}}(P) = 0$. However, B is in Q and X_{j_i} so $f_{X_{j_i}}(Q) = 1$.

9.4.9 Steiner Systems and the National Lottery[22]

The National Lottery in the United Kingdom involves buying a ticket with six of the integers from 1 to 49. Twice a week, six "winning" numbers are randomly drawn. A ticket with three or more of the winning numbers wins at least £10. Our question is: What is the fewest number of tickets that must be bought to ensure a winning ticket? To address this question, we consider a variant of block designs called Steiner systems.

An $S(t,k,v)$ *Steiner system* is a set V of v elements and a collection of subsets of V of size k called blocks such that any t elements of V are in exactly one of the blocks. Any Steiner triple system (Section 9.4.5) is an $S(2,3,v)$ Steiner system.

Steiner systems do not exist for many t, k, and v. In fact, we do not know if any exist for $t \geq 6$.

Theorem 9.22 If an $S(t,k,v)$ Steiner system exists, $\binom{k}{t}$ divides $\binom{v}{t}$ and the number of blocks is given by $\binom{v}{t} \Big/ \binom{k}{t}$.

Proof. Consider an $S(t,k,v)$ Steiner system on the set V. There are $\binom{v}{t}$ t-element subsets of V. Each occurs in exactly one block of V. And each block, being a k-element subset, contains $\binom{k}{t}$ t-element subsets of V. Therefore,

$$\binom{v}{t} \Big/ \binom{k}{t} = \text{ the number of blocks of } S(t,k,v).$$

If this is not an integer, $S(t,k,v)$ cannot exist. Q.E.D.

An $S(3,6,49)$ Steiner system would yield a method to ensure that every possible triple appeared on exactly one of our tickets. (Why?) Unfortunately, an $S(3,6,49)$

[22]This subsection is based on Brinkman, Hodgkinson, and Humphreys [2001].

Steiner system doesn't exist, since

$$\binom{49}{3} \Big/ \binom{6}{3} = 18{,}424/20 = 921.2.$$

A method that modifies this idea (with perhaps some triple of numbers appearing in more than one block) would therefore need at least 922 blocks or tickets (921.2 rounded up). We describe a much more efficient method.

What if the lottery only used numbers between 1 and 26 and required that of the six winning numbers drawn, a winning ticket had to have at least three of them? An $S(3, 6, 26)$ Steiner system exists (Chen [1972]). By Theorem 9.22, it contains $\binom{26}{3} \Big/ \binom{6}{3} = 130$ blocks, so there is a method that can be used to guarantee a winning ticket if 130 tickets are purchased.

We now show how to modify this idea to guarantee a winning ticket in the UK National Lottery with numbers 1 to 49 used if one purchases 260 tickets. Note that any six winning numbers must contain either three even numbers or three odd ones. Thus, the idea is to buy enough tickets to be sure that every even triple and every odd triple is represented. If $V = \{2, 4, \ldots, 48\} \cup \{1, 3\}$, then $|V| = 26$. Using an $S(3, 6, 26)$ Steiner system on V, we find 130 tickets that guarantee that every even triple of integers from 2 through 48 is included in one of these tickets. Similarly, using an $S(3, 6, 26)$ Steiner system on $V = \{1, 3, \ldots, 49\} \cup \{2\}$, we find 130 other tickets that guarantee that every odd triple of integers from 1 through 49 is included in one of these tickets. Thus, 260 tickets in all suffice to guarantee that one of them will have three numbers among the six chosen from 1 to 49. For a recent paper that summarizes all so-called "lotto designs," see Li and van Rees [2002].

EXERCISES FOR SECTION 9.4

1. For each of the following block designs, determine if the design is a BIBD, and if so, determine its parameters b, v, r, k, and λ.

   (a) Varieties: $\{1, 2, 3, 4\}$
   Blocks: $\{1, 2\}, \{1, 3\}, \{2, 4\}, \{1, 2, 3\}, \{2, 3, 4\}$

   (b) Varieties: $\{1, 2, 3, 4, 5\}$
   Blocks: $\{1, 2, 3\}, \{2, 3, 4\}, \{3, 4, 5\}, \{1, 4, 5\}, \{1, 2, 5\}$

   (c) Varieties: $\{1, 2, 3, 4, 5\}$
   Blocks: $\{1, 2, 3, 4\}, \{1, 3, 4, 5\}, \{1, 2, 4, 5\}, \{1, 2, 3, 5\}, \{2, 3, 4, 5\}$

   (d) Varieties: $\{1, 2, 3, 4, 5, 6, 7\}$
   Blocks: $\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 5\}, \{1, 4, 6\}, \{1, 5, 7\}, \{1, 6, 7\}, \{3, 5, 6\}, \{2, 3, 7\}, \{2, 4, 5\}, \{2, 5, 6\}, \{2, 6, 7\}, \{3, 4, 6\}, \{3, 4, 7\}, \{4, 5, 7\}$

   (e) Varieties: $\{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p\}$
   Blocks: $\{a, b, c, d, e, f\}, \{a, b, g, h, i, j\}, \{a, c, g, k, l, m\}, \{a, d, h, k, n, o\}, \{a, e, i, l, n, p\}, \{a, f, j, m, o, p\}, \{b, c, g, n, o, p\}, \{b, d, h, l, m, p\}, \{b, e, i, k, m, o\}, \{b, f, j, k, l, n\}, \{c, d, i, j, m, n\}, \{c, e, h, j, k, p\}, \{c, f, h, i, l, o\}, \{d, e, g, j, l, o\}, \{d, f, g, i, k, p\}, \{e, f, g, h, m, n\}$

2. (a) A BIBD has parameters $v = 6, k = 3$, and $\lambda = 10$. Find b and r.
   (b) A BIBD has parameters $v = 13, b = 78$, and $r = 24$. Find k and λ.
   (c) A BIBD has parameters $b = 85, k = 21$, and $\lambda = 5$. Find v and r.
3. Show that there is no (b, v, r, k, λ)-design with the following parameters.

   (a) $b = 6, v = 9, r = 2, k = 3, \lambda = 1$; (b) $b = 22, v = 22, r = 7, k = 7, \lambda = 2$.

4. Show that there is no (b, v, r, k, λ)-design with the following parameters: $b = 4, v = 9, r = 4, k = 9, \lambda = 4$.
5. Could there be a $(12, 6, 8, 7, 1)$-design?
6. Could there be a $(13, 3, 26, 6, 1)$-design?
7. Could there be a $(6, 8, 6, 8, 6)$-design?
8. In Wadley's experiment (Example 9.13), in the case where there are five cows, find λ.
9. For each of the block designs of Exercise 1, find its incidence matrix.
10. For each of the following block designs, compute $\mathbf{AA}^T$ for $\mathbf{A}$ the incidence matrix of the design.

    (a) The design of Example 9.11 (b) The design of Example 9.12
    (c) A $(15, 15, 7, 7, 3)$-design

11. In a Steiner triple system with $v = 9$, find b and r.
12. The following nine blocks form part of a Steiner triple system with nine varieties:

$$\{a, b, c\}, \{d, e, f\}, \{g, h, i\}, \{a, d, g\}, \{c, e, h\}, \{b, f, i\}, \{a, e, i\}, \{c, f, g\}, \{b, d, h\}.$$

    (a) How many missing blocks are there?
    (b) Add additional blocks that will lead to a Steiner triple system.

13. Show why the threshold schemes constructed from an orthogonal array in Example 9.9 are not anonymous.
14. There is a resolvable $(35, 15, 7, 3, 1)$-design. Find the number of parallel classes and the number of blocks per parallel class.
15. Find a resolvable $(12, 9, 4, 3, 1)$-design different from the one in Table 9.25.
16. Two designs on X and X' are called *isomorphic* if there exists a one-to-one function f from X onto X' such that $f(B)$ is a block of X' whenever B is a block of X and $f(B) = \{f(x) | x \in B\}$.

    (a) Show that any two resolvable $(12, 9, 4, 3, 1)$-designs are isomorphic.
    (b) Show that any two Steiner triple systems on seven varieties are isomorphic.

17. In an anonymous threshold scheme based on the design of Table 9.25, the leader gives person u partial information 8 and person v partial information 6. What is the secret key?
18. If a Steiner triple system has 67 varieties, how many blocks does it have?
19. Compute $\mathbf{AA}^T$ for $\mathbf{A}$ the incidence matrix of a Steiner triple system of 13 varieties.

20. Given a design, the incidence matrix of the *complementary design* is obtained by interchanging 0 and 1 in the incidence matrix of the original design. In general, if one starts with a (b, v, r, k, λ)-design, the complementary design is a $(b', v', r', k', \lambda')$-design.
    (a) Find formulas for b', v', r', and k'.
    (b) Show that $\lambda' = b + \lambda - 2r$.
    (c) Find a $(16, 16, 6, 6, 2)$-design.
    (d) Find a $(12, 9, 8, 6, 5)$-design.
21. Suppose that the complementary design (Exercise 20) of a Steiner triple system with 13 varieties is a (b, v, r, k, λ)-design. Find b, v, r, k, and λ.
22. Construct a Steiner triple system of 21 varieties.
23. Complete the proof of Theorem 9.15 by showing that the set S is in fact a Steiner triple system.
24. Four of the blocks of a $(7, 3, 1)$-design are

$$\{1, 2, 3\}, \quad \{1, 5, 6\}, \quad \{2, 5, 7\}, \quad \text{and} \quad \{1, 4, 7\}.$$

    Find the remaining blocks.
25. Show by construction that there is a $(v, v-1, v-2)$-design.
26. Show that each of the following designs exists.
    (a) A $(31, 15, 7)$-design
    (b) A $(63, 31, 15)$-design
    (c) A $(21, 5, 1)$-design
    (d) A $(31, 6, 1)$-design
27. Compute $\mathbf{AA}^T$ for $\mathbf{A}$ the incidence matrix of a $(31, 15, 7)$-design.
28. Show that in a (v, k, λ)-design, any two blocks have exactly λ varieties in common.
29. (a) If $\mathbf{A}$ is the incidence matrix of a (b, v, r, k, λ)-design, show that $\mathbf{A}^T$ is not necessarily the incidence matrix of a (v, k, λ)-design.
    (b) Show that if $\mathbf{A}$ is the incidence matrix of a (v, k, λ)-design, then $\mathbf{A}^T$ is the incidence matrix of a (v, k, λ)-design.
30. Show that there can be no $(43, 43, 7, 7, 1)$-design.
31. Consider Z_v, the set of integers $\{0, 1, 2, \ldots, v-1\}$, with addition modulo v. A subset D of k integers in Z_v is called a (v, k, λ)-*difference set*, or just a *difference set*, if every nonzero integer in Z_v appears the exact same number λ of times if we list the differences among distinct elements x, y of D (using both $x - y$ and $y - x$) modulo v.
    (a) Show that $D = \{0, 1, 3\}$ is a difference set in Z_7.
    (b) Show that $D = \{0, 1, 4\}$ is not a difference set in Z_7.
    (c) Show that $D = \{0, 1, 6, 8, 18\}$ is a difference set in Z_{21}.
    (d) Find an $(11, 5, 2)$-difference set.
    (e) If D is a (v, k, λ)-difference set, how many elements will it have?
    (f) If D is a (v, k, λ)-difference set, find an expression for λ as a function of v and k.
32. Suppose that D is a (v, k, λ)-difference set. If $x \in Z_v$, let

$$D + x = \{y + x : y \in D\},$$

    where addition is modulo v.

(a) Prove the following theorem:

Theorem: If D is a (v, k, λ)-difference set, then $\{D+x : x \in Z_v\}$ is a (v, k, λ)-design.

(b) Illustrate the theorem by constructing a $(7, 3, 1)$-design corresponding to the difference set $D = \{0, 1, 3\}$ in Z_7.

(c) Illustrate the theorem by constructing a $(21, 5, 1)$-design corresponding to the difference set $D = \{0, 1, 6, 8, 18\}$ in Z_{21}.

(d) Illustrate the theorem by constructing an $(11, 5, 2)$-design corresponding to the difference set you found in Exercise 31(d).

33. Show that if $m \geq 1$ is a power of a prime, there is a $(2m^2+2m+2, m^2+m+1, 2m+2, m+1, 2)$-design.

34. Use the Bruck-Ryser-Chowla Theorem to show that a $(20, 5, 1)$-design could exist.

35. Which of the following (v, k, λ)-designs could possibly exist?

(a) (16, 9, 1) (b) (34, 12, 4) (c) (43, 7, 1)

36. Show that a $(46, 46, 10, 10, 2)$-design does not exist.

37. Show by construction that there is a $(14, 8, 7, 4, 3)$-design. (*Hint:* Use Theorem 9.17 and another theorem.)

38. Show by construction that there is a $(30, 16, 15, 8, 7)$-design. (*Hint:* Use Theorem 9.17 and another theorem.)

39. Show that there is a $(30, 15, 14, 7, 6)$-design.

40. Suppose that there is a (v, k, λ)-design.

(a) Show that there is a $(2v, v, 2k, k, 2\lambda)$-design.

(b) Show that for any positive integer p, there is a $(pv, v, pk, k, p\lambda)$-design.

41. Show that there is a $(62, 31, 30, 15, 14)$-design.

42. We wish to test for the presence of HIV in a group of six people whose names are encoded as A, B, C, D, E, F. Let $P \subseteqq U = \{A, B, C, D, E, F\}$. We use the four groups $X_1 = \{A, B, C\}$, $X_2 = \{A, D, E\}$, $X_3 = \{B, D, F\}$, $X_4 = \{C, E, F\}$.

(a) Compute the vector $(f_{X_1}(P), f_{X_2}(P), f_{X_3}(P), f_{X_4}(P))$ for all subsets P of U.

(b) If the vector $(1, 1, 1, 1)$ is obtained as $f_G(P)$ for some P and we know that $|P| \leq 2$, can we determine P?

(c) Show that this collection of groups gives a successful NAGTA with threshold 1.

43. We wish to determine the interest in a new network on the part of 8 cable TV providers, whose names are encoded as A, B, $\ldots$, H. Let $P \subseteqq \{A, B, C, D, E, F, G, H\}$. Consider the collection G of 6 groups $X_1 = \{A, B, C, D\}$, $X_2 = \{E, F, G, H\}$, $X_3 = \{A, C, E, G\}$, $X_4 = \{B, D, F, H\}$, $X_5 = \{A, B, D, G\}$, $X_6 = \{C, E, F, H\}$.

(a) If the vector $(1, 0, 1, 0, 1)$ is obtained as $f_G(P)$ for some P, can we determine P?

(b) Show that this collection G does not give a successful NAGTA with threshold 1.

44. (a) Use the $(12, 9, 4, 3, 1)$-design of Table 9.25 to construct a successful NAGTA G with threshold 3.

(b) Use the notation $B_{i,j}$ for the jth block in parallel class C_i. If

$$f_G(P) = (1,1,0,1,1,0,1,0,1,1,1,1)$$

and we know that $|P| \leq 3$, find P.

45. If $t \geq 2$, a t-(b, v, r, k, λ)-*design* consists of a set V of $v \geq 2$ varieties, and a collection of $b > 0$ subsets of V called blocks, such that (9.10) and (9.11) hold, such that

$$\text{every } t\text{-element subset of } V \text{ is a subset of exactly } \lambda \text{ blocks, } \lambda > 0, \tag{9.29}$$

and such that $k < v$. Obviously, a 2-(b, v, r, k, λ)-design is a (b, v, r, k, λ)-design.

(a) Suppose that $x_{i_1}, x_{i_2}, \ldots, x_{i_t}$, are t distinct varieties of a t-(b, v, r, k, λ)-design. For $1 \leq j \leq t$, let λ_j be the number of blocks containing $x_{i_1}, x_{i_2}, \ldots, x_{i_j}$. Let $\lambda_0 = b$. Show that for $0 \leq j \leq t$,

$$\lambda_j = \frac{\lambda \binom{v-j}{t-j}}{\binom{k-j}{t-j}}, \tag{9.30}$$

and conclude that λ_j is independent of the choice of $x_{i_1}, x_{i_2}, \ldots, x_{i_j}$. Hence, conclude that for all $1 \leq j \leq t$, a t-(b, v, r, k, λ)-design is also a j-(b, v, r, k, λ)-design.

(b) Show that for $t \geq 2$, (9.10) and (9.29) imply (9.11).

(c) Note that if a t-(b, v, r, k, λ)-design exists, the numbers λ_j defined by (9.30) are integers for all j with $0 \leq j \leq t$.

(d) The results of Section 9.5.2 will imply that there is no $(43, 7, 1)$-design. Use this result to prove that even if all λ_j are integers, this is not sufficient for the existence of a t-(b, v, r, k, λ)-design.

46. Suppose that the square matrix $\mathbf{A}$ is the incidence matrix of a BIBD. Show that $\mathbf{A}^{-1}$ exists.

47. If $\mathbf{A}$ is the incidence matrix of a (b, v, r, k, λ)-design, show that $\mathbf{AJ} = r\mathbf{J}$, where $\mathbf{J}$ is a matrix of all 1's.

48. Suppose that $\mathbf{A}$ is a $v \times v$ matrix of 0's and 1's, $v \geq 2$, and that there are $k > 0$ and $\lambda > 0$ with $k > \lambda$ and so that:

(1) Any row of $\mathbf{A}$ contains exactly k 1's.
(2) Any pair of rows of $\mathbf{A}$ have 1's in common in exactly λ columns.

This exercise asks the reader to prove that:

(3) Any column of $\mathbf{A}$ contains exactly k 1's.
(4) Any pair of columns of $\mathbf{A}$ have 1's in common in exactly λ rows.

[In particular, it follows that (3) and (4) hold for incidence matrices of (v,k,λ)-designs, and Theorem 9.19 follows.]

(a) Show that $\mathbf{AJ} = k\mathbf{J}$, where $\mathbf{J}$ is a square matrix of all 1's.

(b) Show that $\mathbf{AA}^T = (k-\lambda)\mathbf{I} + \lambda\mathbf{J}$.

(c) Show that $\mathbf{A}^{-1}$ exists.

(d) Show that $\mathbf{A}^{-1}\mathbf{J} = k^{-1}\mathbf{J}$.

(e) Show that $\mathbf{A}^T\mathbf{A} = (k-\lambda)\mathbf{I} + \lambda k^{-1}\mathbf{JA}$.

(f) Show that if $\mathbf{JA} = k\mathbf{J}$, then (3) and (4) follow.

(g) Show that $\mathbf{JA} = k^{-1}(k-\lambda+\lambda v)\mathbf{J}$.

(h) Show that $k - \lambda + \lambda v = k^2$.

(i) Show that $\mathbf{JA} = k\mathbf{J}$ and hence that (3) and (4) hold.

49. In an experiment, there are two kinds of treatments or varieties, the controls and the noncontrols. There are three controls and 120 blocks. Each control is used in 48 blocks. Each pair of controls is used in the same block 24 times. All three controls are used in the same block together 16 times. In how many blocks are none of the controls used?

50. (Stinson [2003]). Suppose that there is an anonymous $(2,p)$-threshold scheme that allows any two persons to find the key but no single person to find it with probability higher than $1/|K|$.

(a) Show that $|P| \geq (p-1)|K| + 1$. (*Hint:* For $\kappa \in K$, let C_κ be the set of all possible p-element subsets of P that could be distributed, one per person, when the secret key is κ. Argue that these subsets overlap in exactly one element.)

(b) Show that if $|P| = (p-1)|K|+1 = v$, there must exist a resolvable (b,v,r,k,λ)-design with $k = p$. (*Hint:* Use the subsets defined in part (a) as blocks.)

51. Prove that an $S(5,7,18)$ Steiner system doesn't exist.

52. How many blocks are present in an $S(5,6,48)$ Steiner system?

53. Suppose that an $S(t,k,v)$ Steiner system exists on a set V.

(a) Prove that an $S(t-1,k-1,v-1)$ Steiner system also exists. (*Hint:* Fix an element of V and consider only those k-element subsets that contain it.)

(b) Conclude that an $S(t-j,k-j,v-j)$ Steiner system exists for every $j < t$.

54. (Anderson [1990]) Using Exercise 53 and the fact that an $S(5,8,24)$ Steiner system exists on a set V, show that

(a) the number of blocks in an $S(5,8,24)$ Steiner system is 759,

(b) every element of V lies in 253 blocks,

(c) every pair of elements of V lies in 77 blocks,

(d) every triple of elements of V lies in 21 blocks,

(e) every quadruple of elements of V lies in 5 blocks,

(f) every quintuple of elements of V lies in exactly 1 block.

9.5 FINITE PROJECTIVE PLANES

9.5.1 Basic Properties

It is interesting that experimental designs have geometric applications, and conversely that geometry has played an important role in the analysis of experimental designs. Let us consider the design of Example 9.10. This is a Steiner triple system and a symmetric BIBD. It can be represented geometrically by letting the varieties be points and representing a block by a "line" (not necessarily straight) through the points it contains. Figure 9.1 shows this geometric representation. All but one line is straight. This representation is known as a *projective plane*, the *Fano plane*.[23] It has the following properties:

(P_1) Two distinct points lie on one and only one common line.
(P_2) Two distinct lines pass through one and only one common point.

In general, a *projective plane* consists of a set of objects called *points*, a second set of objects called *lines*, and a notion of when a *point lies on a line*, or equivalently, when a *line passes through a point*, so that conditions (P_1) and (P_2) hold. A projective plane is *finite* if the set of points is finite. Projective planes are important not only in combinatorial design but also in art, where they arise in the study of perspective. They are also important in geometry, for they define a geometry where Euclid's parallel postulate is violated: By (P_2), there is no line that passes through a given point and has no points in common with (and hence is "parallel" to) a given line. The development of projective geometry had its roots in the work of Pappas of Alexandria in the fourth century. It led in the 1840s to the algebraic theory of invariance, developed by the famous mathematicians Boole, Cayley, and Sylvester. This in turn led to the tensor calculus, and eventually to ideas of fundamental importance in physics, in particular to the work of Einstein in the theory of gravitation.

The basic existence question that dominates the theory of combinatorial design arises also for projective planes: For what values of n is there a projective plane of n points? If $n = 2$, we can take two points a and b and one line L that passes through the two points. The postulates (P_1) and (P_2) for a projective plane are trivially satisfied. They are also trivially satisfied if there are n points, any n, and just one line, which passes through all n points. Finally, they are trivially satisfied if there are three points, a, b, and c, and three lines, L_1, L_2, and L_3, with a and b lying on L_1, b and c on L_2, and a and c on L_3. To rule out these dull examples, one usually adds one additional postulate:

(P_3) There are four distinct points, no three of which lie on the same line.

A finite projective plane satisfying (P_3) is called *nondegenerate* and we shall assume (without making the assumption explicit every time) that *all finite projective*

[23] Named after a nineteenth-century mathematician, Gino Fano.

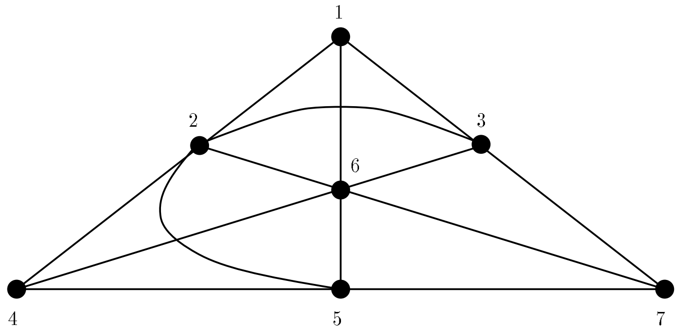


Figure 9.1: The Fano plane.

planes are nondegenerate. Any theorem about these planes will be proved using the postulates (P_1), (P_2), and (P_3).

The smallest possible projective plane would now have at least four points. Is there such a plane with exactly four points? Suppose that a, b, c, and d are four points, and that no three lie on a line. By (P_1), there must be a line L_1 passing through a and b and a line L_2 passing through c and d. Since no three of these points lie on a line, c and d are not on L_1 and a and b are not on L_2. Then if a, b, c, and d are all the points of the projective plane, L_1 and L_2 do not have a common point, which violates (P_2). Thus, there is no projective plane of four points. We shall see below that there is no projective plane of five or six points either. However, the Fano plane of Figure 9.1 is a projective plane of seven points, for (P_3) is easy to verify.

The reader will note that the postulates (P_1) and (P_2) have a certain *duality*: We obtain (P_2) from (P_1) by interchanging the words "point" and "line" and interchanging the expressions "point lying on a line" and "line passing through a point." We obtain (P_1) from (P_2) by the same interchanges. If (P) is any *statement* about finite projective planes, the *dual* of (P) is the statement obtained from (P) by making these interchanges. The dual of postulate (P_3) turns out to be true, and we formulate this result as a theorem.

Theorem 9.23 In a finite projective plane, the following holds:

(P_4) There are four distinct lines, no three of which pass through the same point.

Proof. By (P_3), there are points a, b, c, d no three of which lie on a line. By (P_1), there are lines L_1, through a and b, L_2 through b and c, L_3 through c and d, and L_4 through d and a. Now these four lines are distinct, because c and d are not in L_1, a and d are not in L_2, and so on. Moreover, no three of these lines pass through a common point. We prove this by contradiction. Suppose without loss of generality that L_1, L_2, and L_3 have the point x in common. Then x could not be b, for b is not on L_3. Now L_1 and L_2 have two distinct points in common, b and x. Since $L_1 \neq L_2$, postulate (P_2) is violated, which is a contradiction. Q.E.D.

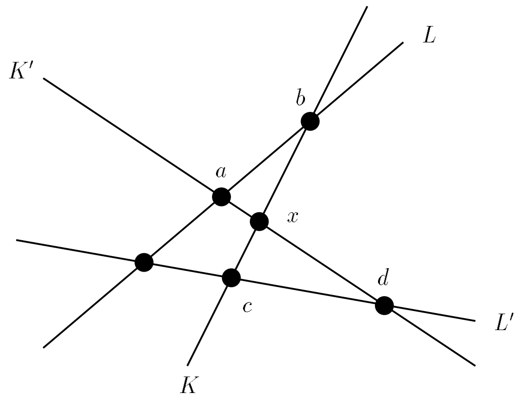


Figure 9.2: The point x is not on either line L or line L'.

Now conditions (P_1) and (P_2) are duals and conditions (P_3) and (P_4) are duals. Any theorem (provable statement) about finite projective planes must be proved from the postulates (P_1), (P_2), and (P_3). Any such theorem will have a *dual theorem*, obtained by interchanging the words "point" and "line" and interchanging the expressions "point lying on a line" with "line passing through a point." A proof of the dual theorem can be obtained from a proof of the theorem by replacing (P_1), (P_2), and (P_3) by their appropriate dual statements, which we know to be true. Thus, we have the following result.

Theorem 9.24 (Duality Principle) For every statement about finite projective planes which is a theorem, the dual statement is also a theorem.

The next basic theorem about finite projective planes is the following.

Theorem 9.25 In a finite projective plane, every point lies on the same number of lines, and every line passes through the same number of points.

To illustrate this theorem, we note that the projective plane of Figure 9.1 has three points on a line and three lines through every point.

Proof of Theorem 9.25. We first show that every line passes through the same number of points. The basic idea of the proof is to set up a one-to-one correspondence between points on two distinct lines, L and L', which shows that the two lines have the same number of points.

We first show that there is a point x not on either L or L'. By postulate (P_3), there are four points a, b, c, and d, no three of which lie on a line. If any one of these is not on either L or L', we can take that as x. If all of these are on L or L', we must have two points (say, a and b) on L and two (say, c and d) on L'. By (P_1), there are lines K through b and c and K' through a and d. By (P_2), the lines K and K' have a point x in common (see Figure 9.2). If x lies on L, then x and b lie

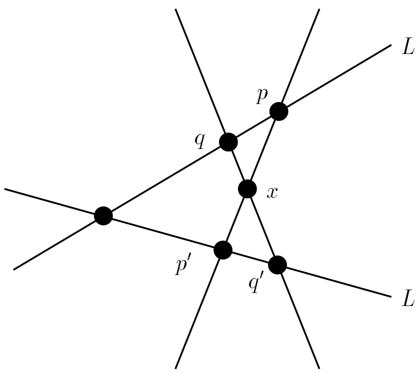


Figure 9.3: The point p' is the projection of p through x onto L'.

on two distinct lines, violating (P_1). If x lies on L', then x and c lie on two distinct lines, again violating (P_1). Thus, x is the desired point.

Now given a point p on line L, the line through p and x [which exists by (P_1)] must meet L' in exactly one point p' [by (P_2)]. We say that p' is the *projection* of p through x onto L' (see Figure 9.3). If q is any other point on L, let q' be its projection through x onto L'. Now if $q \neq p, q'$ must be different from p'. For otherwise, q' and x are on two distinct lines, violating (P_1). We conclude that projection defines a one-to-one correspondence between points of L and points of L'. We know that it is one-to-one. To see that it is a correspondence, note that every point r of L' is obtained from some point of L by this procedure. To see that, simply project back from r through x onto L. Thus, L and L' have the same number of points. This proves that every line passes through the same number of points.

By using the duality principle (Theorem 9.24), we conclude that every point lies on the same number of lines. Q.E.D.

Theorem 9.26 In a finite projective plane, the number of lines through each point is the same as the number of points on each line.

Proof. Pick an arbitrary line L. By (P_3), there is a point x not on line L. By (P_1), for any point y on L, there is one and only one line $L(y)$ passing through x and y. Moreover, any line L' through x cannot be L, and hence by (P_2) it must pass through a point y of L, so L' is $L(y)$. Thus, $L(y)$ defines a one-to-one correspondence between points of L and lines through x. Thus, there are the same number of lines through x as there are points on L. The theorem follows from Theorem 9.25. Q.E.D.

It follows from Theorem 9.26 that the projective plane with $m + 1$ points on each line has $m + 1$ lines through each point.

Corollary 9.26.1 A projective plane with $m+1$ points on each line and $m+1$ lines through each point has $m^2 + m + 1$ points and $m^2 + m + 1$ lines.

Table 9.30: A Projective Plane of Order 3 (Having 13 Points and 13 Lines)

Points:	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
Lines:	{1, 2, 8, 12}, {3, 9, 10, 12}, {4, 5, 11, 12}, {1, 4, 6, 9}, {3, 6, 8, 11}, {2, 5, 6, 10}, {1, 10, 11, 13}, {5, 8, 9, 13}, {2, 3, 4, 13}, {1, 3, 5, 7}, {4, 7, 8, 10}, {2, 7, 9, 11}, {6, 7, 12, 13}.

Proof. Let x be any point. There are $m+1$ lines through x. Each such line has m points other than x. Every point lies on one and only one line through x. Hence, we count all the points by counting the number of points other than x on the lines through x and adding x; we get

$$(m+1)(m)+1=m^2+m+1$$

points. The rest of the corollary follows by a duality argument. Q.E.D.

In our example of Figure 9.1, $m+1=3, m=2$, and $m^2+m+1=7$. We know now that projective planes of n points can only exist for n of the form m^2+m+1. In particular, no such planes can exist for $n=4,5$, or 6. Postulate (P_3) rules out $n=3$ (even though $3=1^2+1+1$). Thus, $n=7$ corresponds to the first possible projective plane. The next possible one is obtained by taking $m=3$, obtaining $n=3^2+3+1=13$. We will see that projective planes exist whenever m is a power of a prime. Thus, the 13-point plane ($m=3$) will exist. (Table 9.30 shows such a plane.) The number m will be called the *order* of the projective plane. Note that the order is different from the number of points.

9.5.2 Projective Planes, Latin Squares, and (v, k, λ)-Designs

In this section we investigate the relations among projective planes and certain kinds of combinatorial designs.

Theorem 9.27 (Bose [1938]) Suppose that $m \geq 2$. A finite projective plane of order m exists if and only if a complete orthogonal family of $m \times m$ Latin squares exists.[24]

The proof of this theorem is constructive; that is, it shows how to go back and forth between finite projective planes and sets of orthogonal Latin squares. We sketch the proof in Exercises 22 and 23.

Corollary 9.27.1 If $m=p^k$ for p prime and k a positive integer, there is a projective plane of order m.

Proof. The result follows by Theorem 9.2. Q.E.D.

[24]Recall that by our convention, a single 2×2 Latin square constitutes an orthogonal family.

It follows from Theorem 9.27 and Corollary 9.27.1 that the first possible order m for which there does not exist a finite projective plane of that order (i.e., of $m^2 + m + 1$ points) is $m = 6$. In fact, for $m = 6$, we have seen in Section 9.2 that there does not exist a set of five orthogonal 6×6 Latin squares (or indeed a pair of such squares), and hence there is no finite projective plane of order 6 (i.e., of $6^2 + 6 + 1 = 43$ points). There are projective planes of orders $7, 8$, and 9, since $7 = 7^1, 8 = 2^3$, and $9 = 3^2$. However, $m = 10$ is not a power of a prime. Lam, Thiel, and Swiercz [1989] used a computer proof to show the nonexistence of a finite projective plane of order 10 (i.e., $10^2 + 10 + 1 = 111$ points). We know that there is a finite projective plane of order 11, but 12 remains an open problem.

The next theorem takes care of some other cases not covered by Corollary 9.27.1. We omit the proof.

Theorem 9.28 (Bruck and Ryser [1949]) Let $m \equiv 1$ or $2 \pmod 4$. Suppose that the largest square dividing into m is d and that $m = m'd$. If m' is divisible by a prime number p which is $\equiv 3 (\text{mod } 4)$, there does not exist a projective plane of order m.

To illustrate this theorem, suppose that $m = 6$. Note that $6 \equiv 2 \pmod 4$. Then $d = 1$ and $m' = m$. Since m' is divisible by 3, there is no projective plane of order 6, as we observed previously. Next, suppose that $m = 54$. Note that $54 \equiv 2 \pmod 4$, that $d = 9$, and that $m' = 6$. Since 3 divides m', there is no projective plane of order 54. It follows by Theorem 9.27 that there is no complete orthogonal family of Latin squares of order 54.

A projective plane gives rise to a (b, v, r, k, λ)-design by taking the varieties as the points and the blocks as the lines. Then $b = v = m^2 + m + 1, k = r = m + 1$, and $\lambda = 1$ since each pair of points lies on one and only one line. Hence, we have a symmetric balanced incomplete block design or a (v, k, λ)-design. Conversely, for every $m \geq 2$, an $(m^2 + m + 1, m + 1, 1)$-design gives rise to a projective plane by taking the varieties as the points and the blocks as the lines. To see why Axiom (P_3) holds, let a and b be any two points. There is a unique block B_1 containing a and b. Since the design is incomplete, there is a point x not in block B_1. Now there is a unique block B_2 containing a and x and there is a unique block B_3 containing b and x. Each block has $m+1$ points. Thus, other than a, b, and x, B_1, B_2, and B_3 each have at most $m - 1$ other points. In short, the number of points in B_1, B_2, or B_3 is at most $3 + 3(m - 1) = 3m$. Since $m \geq 2$, we have $m^2 + m + 1 > 3m$. Thus, there is a point y not in B_1, B_2, or B_3. No three of the points a, b, x, and y lie in a block. Thus, we have the following result.

Theorem 9.29 If $m \geq 2$, a finite projective plane of order m exists if and only if an $(m^2 + m + 1, m + 1, 1)$-design exists.

Corollary 9.29.1 There are $(m^2 + m + 1, m + 1, 1)$-designs whenever m is a power of a prime.

Corollary 9.29.2 Suppose that $m \geq 2$. Then the following are equivalent.

(a) There exists a finite projective plane of order m.
(b) There exists a complete orthogonal family of Latin squares of order m.
(c) There exists an $(m^2 + m + 1, m + 1, 1)$-design.

EXERCISES FOR SECTION 9.5

1. In each of the following, P is a set of points and Q a set of lines. A point x lies on a line L if $x \in L$ and a line L passes through a point x if $x \in L$. Determine which of axioms $(P_1), (P_2)$, and (P_3) hold.

   (a) $P =$ all points in 3-space, $Q =$ all lines in 3-space.

   (b) $P = \{1, 2, 3\}, Q = \{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$.

   (c) $P =$ any set, $Q =$ all ordered pairs from P.

   (d) $P =$ all lines in 3-space, $Q =$ all planes in 3-space.

   (e) $P =$ all lines through the origin in 3-space, $Q =$ all planes through the origin in 3-space.

2. State the dual of each of the following (not necessarily true) statements about finite projective planes.

   (a) There are nine distinct lines, no three of which pass through the same point.

   (b) There is a point that lies on every line.

   (c) There are four distinct lines so that every point lies on one of these lines.

   (d) There are four distinct points, no three of which lie on the same line, so that every line passes through one of these points.

3. If a projective plane has four points on every line, how many points does it have in all?
4. If a projective plane has five lines through every point, how many points does it have in all?
5. If a projective plane has 31 points, how many points lie on each line?
6. If a projective plane has 57 points, how many lines pass through each point?
7. If a projective plane has 73 lines, how many points lie on each line?
8. Is there a projective plane of:

   (a) 25 points? (b) 73 points? (c) 43 lines? (d) 91 lines?

9. Suppose that a projective plane has n points and a (v, k, λ)-design is defined from the plane with the points as the varieties and the lines as the blocks. For each of the following values of n, compute v, k, and λ.

   (a) 31 (b) 91 (c) 133

10. Could there be a finite projective plane of order (not number of points) equal to the following values? Justify your answer.

    (a) 11 (b) 49 (c) 81

11. Show that there is no finite projective plane of order 14.
12. Show that there could be no finite projective plane of order 245.
13. Show that there could be no finite projective plane of order 150.
14. Could there be a finite projective plane of order equal to the following values? Justify your answer.

    (a) 60 (b) 81 (c) 93

15. Show that there is no complete orthogonal family of Latin squares of order 378.
16. Recall the definition of *projection* in the proof of Theorem 9.25. In the Fano plane (Figure 9.1):

    (a) Find the projection of the point 6 on the line $\{3, 4, 6\}$ through the point 5 onto the line $\{1, 3, 7\}$.
    (b) Find the projection of the point 4 on the line $\{1, 2, 4\}$ through the point 3 onto the line $\{2, 6, 7\}$.
    (c) Find the projection of the point 2 on the line $\{1, 2, 4\}$ through the point 3 onto the line $\{4, 5, 7\}$.

17. Recall the definition of *projection* in the proof of Theorem 9.25. In the projective plane of Table 9.30:

    (a) Find the projection of the point 2 on the line $\{2, 3, 4, 13\}$ through the point 8 onto the line $\{1, 4, 6, 9\}$.
    (b) Find the projection of the point 11 on the line $\{2, 7, 9, 11\}$ through the point 1 onto the line $\{2, 5, 6, 10\}$.

18. (Bogart [1983]). Take the points of an $(n^2+n, n^2, n+1, n, 1)$-design as the points and the blocks of this design as the lines. Which of axioms $(P_1), (P_2), (P_3)$ are satisfied?
19. Show that in Exercise 18, the following "parallel postulate" is satisfied: Given a point x and a line L not passing through x, there is one and only one line L' passing through x that has no points in common with L.
20. An *affine plane* is related to a projective plane in that it satisfies axioms (P_1), (P_3), and the "parallel postulate" of Exercise 19. Prove the following about affine planes:

    (a) Every line contains the same number of points.
    (b) Every point is on the same number of lines.
    (c) If every line contains the same number n of points, then every point is on exactly $n + 1$ lines, there are exactly n^2 points, and there are exactly $n^2 + n$ lines.

21. In an affine plane (see Exercise 20), every line contains the same number n of points. We call n the *order* of the affine plane. Prove that if there is a projective plane of order n, there is an affine plane of order n. (*Hint:* Remove a line and all of its points.)

22. The next two exercises sketch a proof of Theorem 9.27. Let P be a finite projective plane of order m.

    (a) Pick a line L from P arbitrarily and call L the *line at infinity*. Let L have points
    $$u, v, w_1, w_2, \ldots, w_{m-1}.$$
    Through each point of L there are $m+1$ lines, hence m lines in addition to L. Let these be listed as follows:
    $$\begin{aligned} \text{lines through } u: &\quad U_1, U_2, \ldots, U_m, \\ \text{lines through } v: &\quad V_1, V_2, \ldots, V_m, \\ \text{lines through } w_j: &\quad W_{j1}, W_{j2}, \ldots, W_{jm}. \end{aligned}$$
    Every point x not on line L is joined by a unique line to each point on L. Suppose that U_h is the line containing x and u, V_i the line containing x and v, and W_{jk_j} the line containing x and w_j. Thus, we can associate with the point x the $(m+1)$-tuple $(h, i, k_1, k_2, \ldots, k_{m-1})$. Show that the correspondence between points x not on L and ordered pairs (h, i) is one-to-one.

    (b) Illustrate this construction with the projective plane of Table 9.30. Write out all of the lines and the associations $x \to (h, i)$ and $x \to (h, i, k_1, k_2)$, assuming that $\{1, 2, 8, 12\}$ is the line at infinity.

    (c) Let $a_{hi}^{(j)} = k_j$ if the point x corresponding to ordered pair (h, i) gives rise to the $(m+1)$-tuple $(h, i, k_1, k_2, \ldots, k_{m-1})$, and let $A^{(j)} = (a_{hi}^{(j)})$, $j = 1, 2, \ldots, m-1$. Find $A^{(1)}$ and $A^{(2)}$ for the projective plane of Table 9.30.

    (d) Show that $A^{(j)}$ is a Latin square.

    (e) Show that $A^{(p)}$ and $A^{(q)}$ are orthogonal if $p \neq q$.

23. Suppose that $A^{(1)}, A^{(2)}, \ldots, A^{(m-1)}$ is a family of pairwise orthogonal Latin squares of order m.

    (a) Consider m^2 "finite" points $(h, i), h = 1, 2, \ldots, m, i = 1, 2, \ldots, m$. Given the point (h, i), associate with it the $(m+1)$-tuple
    $$(h, i, k_1, k_2, \ldots, k_{m-1}),$$
    where k_j is $a_{hi}^{(j)}$ Find these $(m+1)$-tuples given the two orthogonal Latin squares $A^{(1)}$ and $A^{(2)}$ of order 3 shown in Table 9.14. (This will be our running example.)

    (b) Form $m^2 + m = m(m+1)$ lines W_{jk}, $j = -1, 0, 1, 2, \ldots, m-1$, $k = 1, 2, \ldots, m$, by letting W_{jk} be the set of all finite points (h, i) where the $(j+2)$th entry in the $(m+1)$-tuple corresponding to (h, i) is k. (These lines will be extended by one point later.) Identify the lines W_{jk} in our example.

    (c) Note that for fixed j,
    $$W_{j1}, W_{j2}, \ldots, W_{jm} \tag{9.31}$$
    as we have defined them is a collection of m lines, no two of which intersect. We say that two of these are *parallel* in the sense of having no finite points in common. Show that W_{jk} has m finite points.

(d) Show that if $j \neq j'$, then W_{jk} and $W_{j'k'}$ as we have defined them have one and only one common point.

(e) We now have $m+1$ sets of m parallel lines, and any two nonparallels intersect in one point. To each set of parallels (9.31), we now add a distinct "point at infinity," w_j, lying on each line in the set. Let $w_{-1} = u$ and $w_0 = v$. We have added $m+1$ infinite points in all. We then add one more line L, the "line at infinity," defined to be the line consisting of $u, v, w_1, w_2, \ldots, w_{m-1}$. Complete and update the list of lines begun for our example in part (b). Also list all points (finite or infinite) in this example.

(f) Find in general the number of points and lines constructed.

(g) Find the number of points on each line and the number of lines passing through each point.

(h) Verify that postulates $(P_1), (P_2)$, and (P_3) hold with the collection of all points (finite or infinite) and the collection of all lines W_{jk} as augmented plus the line L at infinity. [*Hint:* Verify (P_2) first.]

REFERENCES FOR CHAPTER 9

Anderson, I., *Combinatorial Designs: Construction Methods*, Prentice Hall, Upper Saddle River, NJ, 1990.

Beth, T., Jungnickel, D., and Lenz, H., *Design Theory*, 2nd ed., Cambridge University Press, New York, 1999.

Bogart, K. P., *Introductory Combinatorics*, Pitman, Marshfield, MA, 1983.

Bose, R. C., "On the Application of the Properties of Galois Fields to the Problem of Construction of Hyper-Graeco-Latin Squares," *Sankhyā, 3* (1938), 323–338.

Bose, R. C., Shrikhande, S. S., and Parker, E. T., "Further Results on the Construction of Mutually Orthogonal Latin Squares and the Falsity of Euler's Conjecture," *Canad. J. Math., 12* (1960), 189–203.

Box, G. E. P., Hunter, W. G., and Hunter, J. S., *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*, Wiley, New York, 1978.

Brinkman, J., Hodgkinson, D. E., and Humphreys, J. F., "How to Buy a Winning Ticket on the National Lottery," *Math. Gaz., 85* (2001), 202–207.

Bruck, R. H., and Ryser, H. J., "The Nonexistence of Certain Finite Projective Planes," *Canad. J. Math, 1* (1949), 88–93.

Brunk, M. E., and Federer, W. T., "Experimental Designs and Probability Sampling in Marketing Research," *J. Am. Statist. Assoc., 48* (1953), 440–452.

Chen, K. K., Bliss, C., and Robbins, E. B., "The Digitalis-like Principles of *Calotropis* Compared with Other Cardiac Substances," *J. Pharmacol. Exp. Ther., 74* (1942), 223–234.

Chen, Y., "The Steiner System $S(3, 6, 26)$," *J. Geometry, 2* (1972), 7–28.

Chowla, S., and Ryser, H. J., "Combinatorial Problems," *Canad. J. Math., 2* (1950), 93–99.

Cochran, W. G., and Cox, G. M., *Experimental Designs*, 2nd ed., Wiley, New York, 1957.

Colbourn, C. J., and Dinitz, J. H. (eds.), *The CRC Handbook of Combinatorial Designs*, CRC Press, Boca Raton, FL, 1996.

COLBOURN, C. J., DINITZ, J. H., and STINSON, D. R., "Applications of Combinatorial Designs to Communications, Cryptography, and Networking," in J. D. Lamb and D. A. Preece (eds.), *Surveys in Combinatorics, 1999,* London Mathematical Society Lecture Note Series, 267, Cambridge University Press, Cambridge, 1999, 37–100.

COX, D. R., *Planning of Experiments*, Wiley, New York, 1958.

DAVIES, H. M., "The Application of Variance Analysis to Some Problems of Petroleum Technology," Technical paper, Institute of Petroleum, London, 1945.

DIFFIE, W., and HELLMAN, M. E., "New Directions in Cryptography," *IEEE Trans. Info. Theory, 22* (1976), 644–654.

DINITZ, J. H., and STINSON, D. R. (eds.), *Contemporary Design Theory: A Collection of Surveys*, Wiley, New York, 1992.

DORFMAN, R., "The Detection of Defective Members of a Large Population," *Annals of Math. Stats., 14* (1943), 436–440.

DORNHOFF, L. L., and HOHN, F. E., *Applied Modern Algebra*, Macmillan, New York, 1978.

DU, D.-Z., and HWANG, F. K., *Combinatorial Group Testing and Its Applications*, 2nd ed., World Scientific, Singapore, 2000.

DURBIN, J. R., *Modern Algebra*, 4th ed., University of Chicago Press, Chicago, 1999.

FINNEY, D. J., *An Introduction to the Theory of Experimental Design*, University of Chicago Press, Chicago, 1960.

FISHER, J. L., *Application-Oriented Algebra*, Harper & Row, New York, 1977.

FISHER, R. A., "The Arrangement of Field Experiments," *J. Minist. Agric., 33* (1926), 503–513.

FISHER, R. A., "An Examination of the Different Possible Solutions of a Problem in Incomplete Blocks," *Ann. Eugen., 10* (1940), 52–75.

GARRETT, P., *Making, Breaking Codes: Introduction to Cryptology*, Prentice Hall, Upper Saddle River, NJ, 2001.

GILBERT, J., and GILBERT, L., *Elements of Modern Algebra*, Brooks/Cole, Pacific Grove, CA, 1999.

HALL, M., *Combinatorial Theory*, Blaisdell, Waltham, MA, 1967. (Second printing, Wiley, New York, 1980.)

HENNESSY, J. L., and PATTERSON, D. A., *Computer Architecture: A Quantitative Approach*, 2nd ed., Morgan Kaufmann Publishers, San Francisco, 1998.

HICKS, C. R., *Fundamental Concepts in the Design of Experiments*, Holt, Rinehart and Winston, New York, 1973.

HILL, R., *A First Course in Coding Theory*, Oxford University Press, New York, 1991.

HUGHES, D. R., and PIPER, F. C., *Design Theory*, 2nd ed., Cambridge University Press, New York, 1988.

JOHNSON, D. M., DULMAGE, A. L., and MENDELSOHN, N. S., "Orthomorphisms of Groups and Orthogonal Latin Squares. I," *Canad. J. Math., 13* (1961), 356–372.

JOYE, M., and QUISQUATER, J.-J., "Cryptoanalysis of RSA-Type Cryptosystems: A Visit," in R. N. Wright and P. G. Neumann (eds.), *Network Threats*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 38, American Mathematical Society, Providence, RI, 1998, 21–31.

KALISKI, B. S. (ed.), *Advances in Cryptology – CRYPTO '97. Proceedings of the 17th Annual International Cryptology Conference*, Lecture Notes in Computer Science, 1294, Springer-Verlag, Berlin, 1997.

KASKI, P., and ÖSTERGÅRD, P. R. J., "The Steiner Triple Systems of Order 19," *Math. Comp., 73*, (2004), 2075–2092.

KIRKMAN, T. A., "On a Problem in Combinatorics," *Camb. Dublin Math J., 2* (1847), 191–204.

KOBLITZ, N., *A Course in Number Theory and Cryptography*, 2nd ed., Springer-Verlag, New York, 1994.

LAM, C. W. H., THIEL, L. H., and SWIERCZ, S., "The Nonexistence of Finite Projective Planes of Order 10," *Canad. J. Math., 41* (1989), 1117–1123.

LI, P. C., and VAN REES, G. H. J., "Lotto Design Tables," *J. Combin. Des., 10* (2002), 335–359.

LINDNER, C. C., and RODGER, C. A., *Design Theory*, CRC Press, Boca Raton, FL, 1997.

MACNEISH, H. F., "Euler Squares," *Ann. Math., 23* (1922), 221–227.

MENEZES, A. J., VAN OORSCHOT, P. C., and VANSTONE, S. A., *Handbook of Applied Cryptography*, CRC Press Series on Discrete Mathematics and Its Applications, CRC Press, Boca Raton, FL, 1997.

PATTERSON, D. A., and HENNESSY, J. L., *Computer Organization and Design: The Hardware/Software Interface*, 2nd ed., Morgan Kaufmann Publishers, San Francisco, 1998.

PUGH, C., "The Evaluation of Detergent Performance in Domestic Dishwashing," *Appl. Statist., 2* (1953), 172–179.

RIVEST, R. L., SHAMIR, A., and ADLEMAN, L. M., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM, 21* (1978), 120–126.

RYSER, H. J., *Combinatorial Mathematics*, Carus Mathematical Monographs, No. 14, Mathematical Association of America, Washington, DC, 1963.

SALOMAA, A., *Public-Key Cryptography*, Texts in Theoretical Computer Science—An EATCS Series, Springer-Verlag, Berlin, 1996.

SLOANE, N. J. A., "Error-Correcting Codes and Cryptography," in D. A. Klarner (ed.), *The Mathematical Gardner*, Wadsworth, Belmont, CA, 1981, 346–382.

STINSON, D. R., "A Short Proof of the Nonexistence of a Pair of Orthogonal Latin Squares of Order Six," *J. Combin. Theory (A), 36* (1984), 373–376.

STINSON, D. R., "The Combinatorics of Authentication and Secrecy Codes," *J. Cryptology, 2* (1990), 23–49.

STINSON, D. R., *Combinatorial Designs: Constructions and Analysis*, Springer-Verlag, New York, 2003.

STREET, A. P., and STREET, D. J., *Combinatorics of Experimental Design*, Oxford University Press, New York, 1987.

TARRY, G., "Le Problème de 36 Officieurs," *C. R. Assoc. Fr. Avance. Sci. Nat., 1* (1900), 122–123.

TARRY, G., "Le Problème de 36 Officieurs," *C. R. Assoc. Fr. Avance. Sci. Nat., 2* (1901), 170–203.

WADLEY, F. M., "Experimental Design in the Comparison of Allergens on Cattle," *Biometrics, 4* (1948), 100–108.

WALLIS, W. D., *Combinatorial Designs*, Marcel Dekker, New York, 1988.

WILLIAMS, E. J., "Experimental Designs Balanced for the Estimation of Residual Effects of Treatments," *Aust. J. Sci. Res., A2* (1949), 149–168.

YATES, F., "Incomplete Randomized Blocks," *Ann. Eugen., 7* (1936), 121–140.

YOUDEN, W. J., "Use of Incomplete Block Replications in Estimating Tobacco-Mosaic Virus," *Contrib. Boyce Thompson Inst., 9* (1937), 41–48.

Chapter 10

Coding Theory[1]

10.1 INFORMATION TRANSMISSION

In this chapter we provide a brief overview of coding theory. Our concern will be with two aspects of the use of codes: to ensure the secrecy of transmitted messages and to detect and correct errors in transmission. The methods we discuss have application to communications with computers, with distant space probes, and with missiles in launching pads; to electronic commerce; to optical/magnetic recording; to genetic codes; and so on. Today, the development of coding theory is closely related to the explosion of information technology, with applications to the Internet and the "next generation of network technologies." The rapid development of a myriad of networked devices for computing and telecommunications presents challenging and exciting new issues for coding theory. For many references on the applications of coding theory, and in particular to those having to do with communication with computers, see MacWilliams and Sloane [1983]. For more detailed treatments of coding theory as a whole, in addition to MacWilliams and Sloane [1983], see Berlekamp [1968], Blake and Mullin [1975], Cameron and van Lint [1991], Goldie and Pinch [1991], Hill [1986], Peterson [1961], Peterson and Weldon [1972], Pless [1998], van Lint [1999], or Welsh [1988]. For good short treatments, see Dornhoff and Hohn [1978] or Fisher [1977].

The basic steps in information transmission are modeled in Figure 10.1. We imagine that we start with a "word," an English word or a word already in some code, for example a bit string. In step (a) we encode it, usually into a bit string. We then transmit the encoded word over a transmission channel in step (b). Finally, the received word is decoded in step (c). This model applies to transmission over physical communication paths such as telegraph lines or across space via radio waves. However, as MacWilliams and Sloane [1983] point out, a similar analysis applies, for example, to the situation when data are stored in a computer and later retrieved and to many other data transmission applications in the modern world

[1]Sections 10.1–10.3 form the basis for this chapter. The reader interested in a brief treatment may then go directly to Section 10.5 (which depends on Chapter 9).

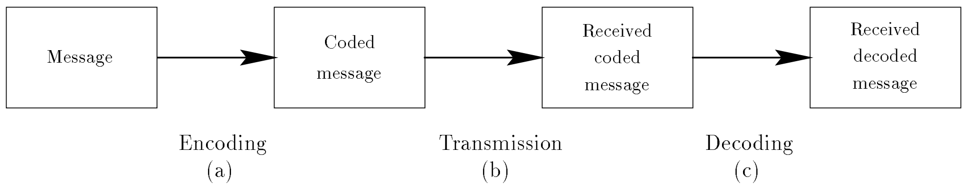


Figure 10.1: The basic steps in information transmission.

of telecommunications. As Fisher [1977] points out, the analysis also applies to communication of visual input into the retina (patterns of photons). The input is encoded into electrical impulses in some of the cells in the retina, and these impulses are transmitted through neurons to a visual area of the brain, where they are "decoded" as a visual pattern. There are many other applications as well.

We shall assume that the only errors which occur are in step (b) and are caused by the presence of noise or by weak signals. In Section 10.2 we discuss steps (a) and (c), the encoding and decoding steps, without paying attention to errors in transmission. In Section 10.3 we begin to see how to deal with such errors. In Section 10.4 we demonstrate how to use the encoding and decoding process to detect and correct errors in transmission. Finally, in Section 10.5 we discuss the use of block designs to obtain error-detecting and error-correcting codes. That section should be omitted by the reader who has skipped Chapter 9.

Much of the emphasis in this chapter is on the existence question: Is there a code of a certain kind? However, we also deal with the optimization question: What is the best or richest or largest code of a certain kind?

10.2 ENCODING AND DECODING

Sometimes, the encoding step (a) of Figure 10.1 must produce a coded message from a message containing "sensitive" information. In this case, we use the terms "encrypting" and "decrypting" to describe steps (a) and (c) of Figure 10.1. However, we shall use the terms "encode" and "decode" throughout. The field called *cryptography* is concerned with such encoding and decoding, and also with deciphering received coded messages if the code is not known. We discuss the encoding and decoding problems of cryptography here but not the deciphering problem.

A message to be encoded involves a sequence of symbols from some *message alphabet* A. The encoded message will be a sequence of symbols from a *code alphabet* B, possibly the same as the message alphabet. A simple encoding rule will encode each symbol a of A as a symbol $E(a)$ of B.

Example 10.1 Caesar Cyphers If A and B are both the 26 uppercase letters of the alphabet, a simple encoding rule $E(a)$ might take $E(a)$ to be the letter following a, with $E(Z) = A$. Thus, the message

$$\text{DEPOSIT SIX MILLION DOLLARS} \tag{10.1}$$

would be encoded as

$$\text{EFQPTJU TJY NJMMJPO EPMMBST.}$$

If A is as above but $B = \{0, 1, 2, \ldots, 25\}$, we could take $E(a) =$ the position of a in the alphabet + 2, where Z is assumed to have position 0. Here $24 + 2$ is interpreted as 0 and $25 + 2$ as 1. (Addition is modulo 26, to use the terminology of Section 9.3.1.) Thus, $E(\text{D}) = 6$. Similar encodings were used by Julius Caesar about 2000 years ago. They are now called *Caesar cyphers.* ■

Any function $E(a)$ from A into B will suffice for encoding provided that we can adequately decode, that is, find a from $E(a)$. To be able to do so unambiguously, we cannot have $E(a) = E(b)$ if $a \neq b$; that is, $E(a)$ must be one-to-one.

It is frequently useful to break a long message up into block of symbols rather than just one symbol, and encode the message in blocks. For instance, if we use blocks of length 2, the message (10.1) becomes

$$\text{DE} \quad \text{PO} \quad \text{SI} \quad \text{TS} \quad \text{IX} \quad \text{MI} \quad \text{LL} \quad \text{IO} \quad \text{ND} \quad \text{OL} \quad \text{LA} \quad \text{RS}. \tag{10.2}$$

We then encode each sequence of two symbols, that is, each block.[2] One way to encode blocks uses matrices, as the next example shows.

Example 10.2 Matrix Codes Let us replace each letter of the alphabet by a number representing its position in the alphabet (with Z having position 0). Then a block corresponds to a vector. For instance, the block DE above corresponds to the vector $(4, 5)$. Suppose that all blocks have length m. Let $\mathbf{M}$ be an $m \times m$ matrix, for instance

$$\mathbf{M} = \begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix}.$$

Then we can encode a block a as a block $a\mathbf{M}$. In our case, we encode a block (i, j) as a block $E(i, j) = (i, j)\mathbf{M}$. Hence, DE or $(4, 5)$ gets encoded as

$$(4, 5) \begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix} = (13, 22).$$

The reader can check that the message (10.1), when broken up as in (10.2), now gets encoded as

$$13, 22, 47, 78, 47, 75, 59, 98, 42, 75, 35, 57, 36, 60, 33, 57, 32, 50, 42, 69, 25, 38, 55, 92.$$

In general, the procedure we have defined is called *matrix encoding.* It is highly efficient. Moreover, decoding is easy. To decode, we break a coded message into blocks b, and find a so that $a\mathbf{M} = b$. We can unambiguously decode provided that $\mathbf{M}$ has an inverse. For then $a = b\mathbf{M}^{-1}$. In our example,

$$\mathbf{M}^{-1} = \begin{pmatrix} 2 & -3 \\ -1 & 2 \end{pmatrix}.$$

[2] To guarantee that a message will always break up into blocks of the same size, we can always lengthen the message by adding a recognizable "closing" string of copies of the same letter, for example, Z or ZZZ.

For instance, if we have the encoded block $(6, 11)$, we find that it came from

$$(6, 11)\mathbf{M}^{-1} = (6, 11)\begin{pmatrix} 2 & -3 \\ -1 & 2 \end{pmatrix} = (1, 4) = \text{AD}.$$

Note that if we have the encoded block $(3, 4)$, decoding gives us

$$(3, 4)\mathbf{M}^{-1} = (2, -1).$$

Since $(2, -1)$ does not correspond to any pair of letters, we can only conclude that an error was made in sending us the message that included the block $(3, 4)$. Much of the emphasis in this chapter is on ways to detect errors, as we have done here. ■

In general, suppose that a message is broken up into blocks of length k. Let A^k consist of all blocks (sequences) of length k from the message alphabet A and B^n consist of all blocks (sequences) of length n from the code alphabet B. Let $\mathcal{A}$ be a subset of A^k called the set of *message blocks*. In most practical applications, $\mathcal{A} = A^k$, and we shall assume this unless stated explicitly otherwise. A *block code* or $k \to n$ *block code* is a one-to-one function $E\colon \mathcal{A} \to B^n$. The set C of all $E(a)$ for a in $\mathcal{A}$ is defined to be the set of *codewords*. Sometimes C alone is called the *code*. In Example 10.2, $k = n = 2$,

$$\begin{aligned} A &= \{\text{A}, \text{B}, \cdots, \text{Z}\}, \\ B &= \{0, 1, \cdots, 25\}, \end{aligned}$$

and $\mathcal{A} = A^k$. The blocks $(13, 22)$ and $(6, 11)$ are codewords. However, the block $(3, 4)$ is not a codeword. In most practical examples, A and B will both be $\{0, 1\}$, messages will be bit strings, and the encoding will take bit strings of length k into bit strings of length n. We shall see in Section 10.4 that taking $n > k$ will help us in error detection and correction.

Example 10.3 A Repetition Code Perhaps the simplest way to encode a message is to repeat the message. This type of encoding results in a *repetition code*. Suppose that we define $E\colon A^k \to A^{pk}$ by

$$E(a_1 a_2 \cdots a_k) = a_1 a_2 \cdots a_k a_1 a_2 \cdots a_k \cdots a_1 a_2 \cdots a_k,$$

where we have p copies of $a_1 a_2 \cdots a_k$. For instance, suppose that $k = 4$ and $p = 3$. Then $E(a_1 a_2 a_3 a_4) = a_1 a_2 a_3 a_4 a_1 a_2 a_3 a_4 a_1 a_2 a_3 a_4$. This is an example of a triple repetition code or $k \to 3k$ block code. In such a code, it is easy to detect errors by comparing the successive elements of A^k in the coded message received. It is even possible to use repetition codes to correct errors. We can simply use the majority rule of decoding. We pick for the ith digit in the message that letter from the message alphabet that appears most often in that place among the p copies. For instance, if $k = 4$, $p = 3$, and we receive the message $axybauybaxvb$, then since x appears a majority of times in the second position and y appears a majority of times in the third position, we "correct" the error and interpret the original message as $axyb$. We shall find more efficient ways to correct and detect errors later in this chapter. ■

We close this section by giving one more example of a block code called a *permutation code*.

Example 10.4 Permutation Codes Suppose that $A = B = \{0, 1\}$ and π is a permutation of $\{1, 2, \ldots, k\}$. Then we can define $E : A^k \to B^k$ by taking $E(a_1 a_2 \cdots a_k) = a_{\pi(1)} a_{\pi(2)} \cdots a_{\pi(k)}$. For instance, suppose that $k = 3$ and $\pi(1) = 2, \pi(2) = 3, \pi(3) = 1$. Then $E(a_1 a_2 a_3) = a_2 a_3 a_1$, so $E(011) = 110, E(101) = 011$, and so on. ■

It is easy to see (Exercises 15 and 16) that every such permutation code and repetition code is a matrix encoding.

EXERCISES FOR SECTION 10.2

1. Suppose that
$$\mathbf{M} = \begin{pmatrix} 1 & 0 \\ 2 & 4 \end{pmatrix}.$$
Encode the following expressions using the matrix code defined by $\mathbf{M}$.

   (a) AX (b) UV (c) BUNNZ HIL

   (d) SELL ALL SHARES OF IBM (e) INVEST TWO MILLION

   (f) ABORT THE MISSIONZ (*Note*: Z has been added at the end because there is an odd number of letters.)

2. Suppose that
$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$
Using $\mathbf{M}$, find the codeword $x_1 x_2 \cdots x_n$ corresponding to each of the following message words $a_1 a_2 \cdots a_k$.

   (a) 11 (b) 10 (c) 01 (d) 00

3. Suppose that
$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$
Repeat Exercise 2 for the following message words.

   (a) 111 (b) 101 (c) 000

4. Suppose that
$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$
Repeat Exercise 2 for the following message words.

   (a) 1111 (b) 1000 (c) 0001

5. Suppose that
$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 1 & 2 & 1 \end{bmatrix}.$$
Encode the following expressions using the matrix code defined by $\mathbf{M}$.
   (a) ABC (b) XAT (c) TTU
   (d) BUY TWENTY SHARES (e) SEND THE MESSAGE AT EIGHT
   (f) OBSERVE THE TRANSFERSZZ. (*Note*: Two Z's have been added to make the number of letters divisible by 3.)

6. Repeat Exercise 1 with
$$\mathbf{M} = \begin{pmatrix} 4 & 6 \\ 1 & 2 \end{pmatrix}.$$

7. Repeat Exercise 5 with
$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 4 \\ 2 & 1 & 0 \end{bmatrix}.$$

8. If $\mathbf{M}$ of Exercise 1 is used in a matrix code, decode the following or show that an error was made; that is, this is not proper code.
   (a) $(5, 8)$ (b) $(8, 12)$ (c) $(1, 8)$ (d) $(7, 12)$
   (e) $16, 48, 44, 60$ (f) $8, 16, 8, 12$ (g) $6, 10, 51, 69, 20, 20$

9. If $\mathbf{M}$ of Exercise 5 is used in a matrix code, decode the following or show that an error was made; that is, this is not proper code.
   (a) $(8, 14, 14)$ (b) $(17, 35, 23)$ (c) $(24, 47, 30)$
   (d) $52, 95, 102, 46, 72, 88$ (e) $48, 91, 70, 15, 27, 23$

10. Repeat Exercise 8 for $\mathbf{M}$ of Exercise 6.

11. Repeat Exercise 9 for $\mathbf{M}$ of Exercise 7.

12. For each of the following permutations of $\{1, 2, \ldots, k\}$, find the values of E indicated in the corresponding permutation code.
   (a) $k = 3, \pi(1) = 3, \pi(2) = 2, \pi(3) = 1$.
      i. $E(110)$ ii. $E(010)$ iii. $E(011)$
   (b) $k = 4, \pi(1) = 3, \pi(2) = 4, \pi(3) = 1, \pi(4) = 2$.
      i. $E(0110)$ ii. $E(1100)$ iii. $E(0111)$
   (c) $k = 5, \pi(1) = 5, \pi(2) = 1, \pi(3) = 4, \pi(4) = 2, \pi(5) = 3$.
      i. $E(00001)$ ii. $E(10101)$ iii. $E(11010)$

13. Find C if $\mathcal{A}$ consists of the blocks $010, 111, 001$, and 110 and $E(abc)$ is $a+b, a+c, b+c$, where addition is modulo 2.

14. Make up an example of a $1 \to 2$ block code.

15. Show that every permutation code is a matrix encoding.

16. Show that every repetition code is a matrix encoding.

10.3 ERROR-CORRECTING CODES

10.3.1 Error Correction and Hamming Distance

In this section we study the use of codes to detect and correct errors in transmission. In particular, we study step (b) of Figure 10.1. We assume that we start with an encoded message, which has been encoded using a block code, with blocks in the code alphabet having length n. For concreteness, we assume that the encoded message is a bit string, so all blocks are bit strings of length n. We speak of *binary codes* or *binary block codes* or *binary n-codes*. In Exercise 7 we modify the assumption and take encoded messages that are strings from an alphabet $\{0, 1, \ldots, q-1\}$. We then speak of *q-ary codes*. The message is sent by blocks. We assume that there are no errors in encoding, so the only blocks that are ever sent are codewords. Let us suppose that the only possible errors that can take place in transmission are interchanges of the digits 0 and 1. Other errors, for example deletion or addition of a digit, will be disregarded. They are easily detected since all blocks have the same length. Following common practice, we shall assume that the probability of changing 1 to 0 is the same as the probability of changing 0 to 1, and that the probability of an error is the same at each digit, independent of any previous errors that may have occurred.[3] In this case, we speak of a *binary symmetric channel*. For the implications of these assumptions, see Section 10.3.3.

Recall that we will be sending only codewords. If we ever receive a block that is not a codeword, we have *detected* an error in transmission. For instance, if the codeword 10010 is sent and the block 10110 is received, and if 10110 is not a codeword, we know there was an error. There are situations where we wish to *correct* the error, that is, guess what codeword was sent. This is especially the case when we cannot ask for retransmission, for instance in transmission of a photograph from a space probe (such as occurred on the Mariner 9 Mars probe, whose code we discuss below), or if the transmission is based on an old magnetic tape. Error-detecting and error-correcting codes have many applications. For instance, simple error-detecting codes called parity check codes (see Example 10.6) were used on the earliest computers: UNIVAC, Whirlwind I, and the IBM 650 (Wakerly [1978]). Indeed, it was the idea that coding methods could correct errors in the early computers used at Bell Laboratories that led Hamming [1950] to develop error-correcting codes. Such codes are now used extensively in designing fault-tolerant computers. (For references on the applications of error-detecting and error-correcting codes to computing, see MacWilliams and Sloane [1983], Pless [1998], Poli and Huguet [1992], Sellers, Hsiao, and Bearnson [1968], or Wakerly [1978], and the survey articles by Avizenius [1976] and Carter and Bouricius [1971].) Such codes were also fundamental to the design of the compact disc, which revolutionized the music industry in the 1980s (as we note in Section 10.5).

There are good ways of designing error-correcting codes. Suppose that we choose the code so that in the set of codewords, no two codewords are too "close" or too

[3] Errors that don't occur randomly sometimes occur in bursts, i.e., several errors close together. See Bossert, *et al.* [1997] for more on burst-error-correcting codes.

similar. For example, suppose that the only codewords are

$$000000, \quad 010101, \quad 101010, \quad \text{and} \quad 111111. \tag{10.3}$$

Then we would be very unlikely to confuse two of these, and a message that is received which is different from one of the codewords could readily be interpreted as the codeword to which it is closest.

Let us make this notion of closest more precise. The *Hamming distance* between two bit strings of the same length is the number of digits on which they differ.[4] Hence, if $d(\cdot, \cdot)$ denotes this distance, we have

$$d(000000, 010101) = 3.$$

Our aim is to find a set of codewords with no two words too close in terms of the Hamming distance.

Suppose that we have found a set of codewords (all of the same length, according to our running assumption). Suppose that the smallest distance between two of these codewords is d. Then we can *detect* all errors of $d-1$ or fewer digits. For if $d-1$ or fewer digits are interchanged, the resulting bit string will not be a codeword and we can recognize or detect that there was an error. For example, if the possible codewords are those of (10.3), then d is 3 and we can detect errors in up to two digits. Suppose we use the strategy that if we receive a word which is not a codeword, we interpret it as the codeword to which it is closest in terms of Hamming distance. (In case of a tie, choose arbitrarily.) This is called the *nearest-neighbor rule.* For example, if the codewords are those of (10.3), and we receive the word 010000, we would interpret it as 000000, since

$$d(010000, 000000) = 1$$

whereas for every other codeword α,

$$d(010000, \alpha) > 1.$$

Using the nearest-neighbor rule, we can *correct* all errors that involve fewer than $d/2$ digits. For if fewer than $d/2$ digits are interchanged, the resulting bit string is closest to the correct codeword (the one transmitted), and so is interpreted as that codeword by the nearest-neighbor rule. Hence, we have an *error-correcting code.* A code that can correct up to t errors is called a *t-error-correcting code.*

We summarize our results as follows.

Theorem 10.1 Suppose that d is the minimum (Hamming) distance between two codewords in the binary code C. Then the code C can detect up to $d-1$ errors and, using the nearest-neighbor rule, can correct up to $\lceil (d/2) - 1 \rceil$ errors.

Our next theorem says that no error-correcting rule can do better than the nearest-neighbor rule.

[4] The Hamming distance was named after R. W. Hamming, who wrote the pioneering paper (Hamming [1950]) on error-detecting and error-correcting codes.

Theorem 10.2 Suppose that d is the minimum (Hamming) distance between two codewords in the binary code C. Then no error-detecting rule can detect more than $d-1$ errors and no error-correcting rule can correct more than $t = \lceil (d/2) - 1 \rceil$ errors.

Proof. Since d is the minimum (Hamming) distance, there are two codewords α and β with $d(\alpha, \beta) = d$. If α is sent and d errors occur, then β could be received. Thus, no error would be detected. Next, note that an error-correcting rule assigns a codeword $R(\lambda)$ to each word λ (each bit string λ of length n). Let γ be a word different from α and β with $d(\alpha, \gamma) \leq t+1$ and $d(\beta, \gamma) \leq t+1$. Then $R(\gamma)$ must be some codeword. This cannot be both α and β. Say that it is not α without loss of generality. Then α could be sent, and with at most $t+1$ errors, γ could be received. This would be "corrected" as $R(\gamma)$, which is not α. Q.E.D.

Since the nearest-neighbor rule (and any other reasonable rule) does not allow us to correct as many errors as we could detect, we sometimes do not attempt to correct errors but only to detect them and ask for a retransmission if errors are detected. This is the case in such critical situations as sending a command to fire a missile or to change course in a space shot.

Suppose that we know how likely errors are to occur. If they are very likely to occur, we would like to have a code with minimum Hamming distance d fairly large. If errors are unlikely, this minimum distance d could be smaller, perhaps as small as 3 if no more than 1 error is likely to occur. In general, we would like to be able to construct sets of codewords of given length n with minimum Hamming distance a given number d. Such a set of codewords in a binary code will be called an (n, d)*-code.*

There always is an (n, d)-code if $d \leq n$. Let the set of codewords be

$$\underbrace{000\cdots0}_{n \text{ 0's}} \quad \text{and} \quad \underbrace{111\cdots1}_{d \text{ 1's}}\underbrace{00\cdots0}_{n-d \text{ 0's}}.$$

The trouble with this code is that there are very few codewords. We could only encode sets $\mathcal{A}$ of two different message blocks. We want ways of constructing richer codes, codes with more possible codewords.

In Section 10.4 we shall see how to use clever encoding methods $E : \mathcal{A} \to B^n$ to find richer (n, d)-codes. In Section 10.5 we shall see how to use block designs, and in particular their incidence matrices, to define very rich (n, d)-codes. In the next subsection, we obtain an upper bound on the size of an (n, d)-code.

In Example 10.3, we introduced the idea of a repetition code.

Example 10.5 A Repetition Code (Example 10.3 Revisited) As described in Example 10.3, a repetition code that repeats a block r times, a $k \to rk$ code, takes $E(a) = aa\cdots a$, where a is repeated r times. Using such a code, we can easily detect errors and correct errors. For example, consider the case where we have a triple repetition code. The set C of codewords consists of blocks of length $3k$ which have a block of length k repeated three times. To detect errors, we compare the ith digits in the three repetitions of the k-block. Thus, suppose that $k = 4$ and we receive 001101110110, which breaks into k-blocks as 0011/0111/0110. Then we know that there has been an error, since, for example, the second digits of the first two k-blocks differ. We can detect up to two transmission errors in this way. However, three errors might go undetected if they were all in the same digit of the original k-block. Of course, the minimum distance between two codewords aaa is $d = 3$, so this observation agrees with the result of Theorem 10.2. We could try to correct the error as follows. Note that the second digit of the k-blocks is 1 twice and 0 once. If we assume that errors are unlikely, we could use the majority rule and assume that the correct digit was 1. Using similar reasoning, we would decode 001101110110 as 0111. This error-correcting procedure can correct up to one error. However, two errors would be "corrected" incorrectly if they were both on the same digit. This observation again agrees with the results of Theorem 10.2, as we have a code with $d = 3$. If we want to correct more errors, we simply use more repetitions. For example, a five-repetitions code $E(a) = aaaaa$ has $d = 5$ and can detect up to four errors and correct up to two. Unfortunately, the repetition method of designing error-correcting codes is expensive in terms of both time and space. ■

Example 10.6 Parity Check Codes A simple way to detect a single error is to add one digit to a block, a digit that will always bring the number of 1's to an even number. The extra digit added is called a *parity check digit*. The corresponding code is a $k \to (k+1)$ code, and it lets $E(0011) = 00110, E(0010) = 00101$, and so on. We can represent E by

$$E(a_1a_2\cdots a_k) = a_1a_2\cdots a_k\sum_{i=1}^{k} a_i,$$

where $\sum_{i=1}^{k} a_i$ is interpreted as the summation modulo 2, as defined in Section 9.3.1. Throughout the rest of this chapter, addition will be interpreted this way. Thus, $1 + 1 = 0$, which is why E(0011) = 00110. In such a *parity check code* E, the minimum distance between two codewords is $d = 2$, so one error can be detected. No errors can be corrected. As we have pointed out before, such parity check codes were used on the earliest computers: UNIVAC, Whirlwind I, and the IBM 650. Large and small computers of the 1960s had memory parity check as an optional feature, and by the 1970s this was standard on a number of machines. ■

10.3.2 The Hamming Bound

Fix a given bit string s in an (n, d)-code C. Let the collection of bit strings of length n which have distance exactly t from string s be denoted $B_t(s)$. Then the number of bit strings in $B_t(s)$ is given by $\binom{n}{t}$, for we choose t positions out of n in which to change digits. Let the collection of bit strings of length n that have distance *at* *most* t from s be denoted by $B'_t(s)$. Thus, the number of bit strings in $B'_t(s)$ is given by

$$b = \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{t}. \tag{10.4}$$

If $t = \lceil (d/2) - 1 \rceil$, then $B'_t(s) \cap B'_t(s^*) = \emptyset$ for every $s \neq s^*$ in C. Thus, with $t = \lceil (d/2) - 1 \rceil$, every bit string of length n is in at most one set $B'_t(s)$. Since there are 2^n bit strings of length n, we have

$$|C|\, b = \sum_{s \in C} |B'_t(s)| = |\cup_{s \in C} B'_t(s)| \leq 2^n.$$

Thus, we get the following theorem.

Theorem 10.3 (Hamming Bound) If C is an (n, d)-code and $t = \lceil (d/2) - 1 \rceil$, then

$$|C| \leq \frac{2^n}{\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{t}}. \tag{10.5}$$

This result is due to Hamming [1950]. It is sometimes called the *sphere packing bound.*

To illustrate this theorem, let us take $n = d = 3$. Then $t = 1$. We find that any (n, d)-code has at most

$$\frac{2^3}{\binom{3}{0} + \binom{3}{1}} = 2$$

codewords. There is a $(3, 3)$-code of two codewords: Use the bit strings 111 and 000. We investigate an alternative upper bound on the size of (n, d)-codes in Section 10.5.3.

10.3.3 The Probability of Error

Recall that we are assuming that we have a binary symmetric channel: The probability of switching 1 to 0 is the same as the probability of switching 0 to 1, and this common probability p is the same at each digit, independent of any previous errors that may have occurred. Then we have the following theorem.

Theorem 10.4 In a binary symmetric channel, the probability that exactly r errors will be made in transmitting a bit string of length n is given by

$$\binom{n}{r} p^r (1 - p)^{n-r}.$$

Proof.[5] The student familiar with probability will recognize that we are in a situation of Bernoulli trials as described in Example 5.35. We seek the probability

[5] The proof should be omitted by the student who is not familiar with probability.

of r successes in n independent, repeated trials, if the probability of success on any one trial is p. The formula given in the theorem is the well-known formula for this probability (see Example 5.35). Q.E.D.

For instance, suppose that $p = .01$. Then in a 4-digit message, the probability of no errors is

$$(1 - .01)^4 = .960596,$$

the probability of one error is

$$\binom{4}{1}(.01)(1 - .01)^3 = .038812,$$

the probability of two errors is

$$\binom{4}{2}(.01)^2(1 - .01)^2 = .000588,$$

and the probability of more than two errors is

$$1 - .960596 - .038812 - .000588 = .000004.$$

It is useful to compare the two codes of Examples 10.5 and 10.6. Suppose that we wish to send a message of 1000 digits, with the probability of error equal to $p = .01$. If there is no encoding, the probability of making no errors is $(1 - .01)^{1000}$, which is approximately .000043. Suppose by way of contrast that we use the $k \to 3k$ triple repetition code of Example 10.5, with $k = 1$. Suppose that we want to send a digit a. We encode it as aaa. The probability of correctly sending the block aaa is $(.99)^3$, or approximately .970299. By Theorem 10.4, the probability of a single error is $\binom{3}{1}(.01)(.99)^2$, or approximately .029403. Thus, since we can correct single errors, the probability of correctly interpreting the message aaa and hence of correctly decoding the single digit a is $.970299 + .029403 = .999702$. There are 1000 digits in the original message, so the probability of decoding the whole message correctly is $(.999702)^{1000}$, or approximately .742268. This is much higher than .000043. Note that the greatly increased likelihood of error-free transmission is bought at a price: We need to send 3000 digits in order to receive 1000.

Let us compare the parity check code. Suppose that we break the 1000-digit message into blocks of length 10; there are 100 such in all. Each such block is encoded as an 11-digit block by adding a parity check digit. The probability of sending the 11-digit block without making any errors is $(.99)^{11}$, or approximately .895338. Also, by Theorem 10.4, the probability of sending the block with exactly one error is $\binom{11}{1}(.01)(.99)^{10}$, or approximately .099482. Now if a single error is made, we can detect it and ask for a retransmission. Thus, it is reasonable to assume that we can eliminate single errors. Hence, the probability of receiving the 11-digit block correctly is

$$.895338 + .099482 = .994820.$$

Now the original 1000-digit message has 100 blocks of 10 digits, so the probability of eventually decoding this entire message correctly is $.994820^{100}$, or approximately .594909. The probability of correct transmission is less than with a triple repetition code, but much greater than without a code. The price is much less than the triple repetition code: We need to send 1100 digits to receive 1000.

10.3.4 Consensus Decoding and Its Connection to Finding Patterns in Molecular Sequences[6]

Consider the situation of transmission over a "noisy" binary symmetric channel where there is a good chance of errors. We might request retransmission a number of times, as in repetition codes, receiving a collection of strings in B^n, say strings $x_1, x_2, \ldots, x_p$. Some of them are in the set C of codewords, others are not. We would like to determine which codeword in C was intended. The problem is to find a word in C that is in some sense a "consensus" of the words actually received. The idea of "majority rule decoding" is a special case of this. This problem of consensus is widely encountered in applications including voting and "group decisionmaking," selecting the closest match from a database of molecular sequences, and "metasearching" on the Internet (comparing the results of several search engines). The mathematics of the consensus problem has been studied widely in the literature of "mathematical social science." For an introduction to this topic, see Bock [1988], Johnson [1998], and Roberts [1976].

One consensus procedure widely in use is the *median procedure*: Find the codeword w in C for which $\sum_{i=1}^{p} d(w, x_i)$ is minimized, where d is the Hamming distance. An alternative, the *mean procedure*, is to find the codeword w in C for which $\sum_{i=1}^{p} d(w, x_i)^2$ is minimized. Consider, for example, the code C consisting of the codewords in (10.3). Suppose that we ask for three transmissions of our message and receive the three words:

$$x_1 = 100000, \quad x_2 = 110000, \quad x_3 = 111000.$$

We calculate

$$\sum_{i=1}^{3} d(000000, x_i) = 1 + 2 + 3 = 6, \qquad \sum_{i=1}^{3} d(010101, x_i) = 4 + 3 + 4 = 11,$$
$$\sum_{i=1}^{3} d(101010, x_i) = 2 + 3 + 2 = 7, \qquad \sum_{i=1}^{3} d(111111, x_i) = 5 + 4 + 3 = 12.$$

Since 000000 has the smallest sum, it is chosen by the median procedure. We call

[6] This subsection may be omitted.

it a *median*. Similarly,

$$\sum_{i=1}^{3} d(000000, x_i)^2 = 1+4+9 = 14, \qquad \sum_{i=1}^{3} d(010101, x_i)^2 = 16+9+16 = 41,$$
$$\sum_{i=1}^{3} d(101010, x_i)^2 = 4+9+4 = 17, \qquad \sum_{i=1}^{3} d(111111, x_i)^2 = 25+16+9 = 50,$$

so the mean procedure also chooses codeword 000000. We call it a *mean*.

The median procedure and mean procedure do not always agree and, moreover, they do not always give a unique decoding—there can be ambiguity. Consider, for example, the code

$$C = \{111111, 001110\}.$$

Suppose that words $x_1 = 001111$, $x_2 = 101011$ are received. Then

$$\sum_{i=1}^{2} d(111111, x_i) = 2+2 = 4, \qquad \sum_{i=1}^{2} d(001110, x_i) = 1+3 = 4,$$
$$\sum_{i=1}^{2} d(111111, x_i)^2 = 4+4 = 8, \qquad \sum_{i=1}^{2} d(001110, x_i)^2 = 1+9 = 10.$$

The median procedure leads to ambiguity, giving both codewords in C as medians. However, the mean procedure leads to the unique solution 111111 as the mean.

In many problems of the social and biological sciences, data are presented as a sequence or "word" from some alphabet Σ.[7] Given a set of sequences, we seek a *pattern* or *feature* that appears widely, and we think of this as a *consensus sequence* or set of sequences. A pattern is often thought of as a consecutive subsequence of short, fixed length. In Example 6.11 we noted that the discovery of such patterns or consensus sequences for molecular sequences has already led to such important discoveries as the fact that the sequence for platelet-derived factor, which causes growth in the body, is 87 percent identical to the sequence for *v-sis*, a cancer-causing gene. This led to the discovery that *v-sis* works by stimulating growth. To measure how closely a pattern fits into a sequence, we have to measure the distance between words of different lengths. If b is longer than a, then $d(a, b)$ could be the smallest number of mismatches in all possible *alignments* of a as a consecutive subsequence of b. We call this the *best-mismatch distance*. If the sequences are bit strings, this is

[7] The following discussion of pattern recognition in molecular sequences follows Mirkin and Roberts [1993]. It is an example from the field known as *bioconsensus*, which involves the use of (mostly) social-science-based methods in biological applications. In typical problems of bioconsensus, several alternatives (such as alternative phylogenetic trees or alternative molecular sequences) are produced using different methods or models, and then one needs to find a consensus solution. Day and McMorris [1992] surveyed the use of consensus methods in molecular biology and Day [2002] and Day and McMorris [1993] gave many references; there are literally hundreds of papers in the field of "alignment and consensus." For example, Kannan [1995] surveyed consensus methods for phylogenetic tree reconstruction. For more information, see Day and McMorris [2003] and Janowitz, *et al.* [2003].

the minimum Hamming distance between a and all consecutive subsequences of b of the same length as a. Consider, for example, the case where $a = 0011$, $b = 111010$. Then the possible alignments are:

$$\begin{array}{lll} 111010 & 1111010 & 111010 \\ 0011 & \;\;0011 & \;\;\;\;0011 \end{array}$$

The best-mismatch distance is 2, which is achieved in the third alignment. An alternative way to measure $d(a, b)$ is to count the smallest number of mismatches between sequences obtained from a and b by inserting gaps in appropriate places (where a mismatch between a letter of Σ and a gap is counted as an ordinary mismatch). We won't use this alternative measure of distance, although it is used widely in molecular biology.

Waterman [1989] and others (Waterman, Arratia, and Galas [1984], Galas, Eggert, and Waterman [1985], and Waterman [1995]) study the situation where Σ is a finite alphabet, M is a fixed finite number (the pattern length), $\chi = \{x_1, x_2, \ldots, x_n\}$ is a set of words (sequences) of length L from Σ, where $L \geq M$, and we seek a set $F(\chi) = F(x_1, x_2, \ldots, x_n)$ of consensus words of length M from Σ. The set of consensus words could be any words in Σ^M or words from a fixed subset of allowable pattern words. For now, we shall disregard the latter possibility. Here is a small piece of data from Waterman [1989], in which he looks at 59 bacterial promoter sequences:

RRNABP1	ACTCCCTATAATGCGCCA
TNAA	GAGTGTAATAATGTAGCC
UVRBP2	TTATCCAGTATAATTTGT
SFC	AAGCGGTGTTATAATGCC.

Notice that if we are looking for patterns of length 4, each sequence has the pattern TAAT. However, suppose that we add another sequence:

M1RNA AACCCTCTATACTGCGCG.

The pattern TAAT does not appear here. However, it almost appears since the word TACT appears, and this has only one mismatch from the pattern TAAT. So, in some sense, the pattern TAAT is a pattern that is a good consensus pattern. We now make this idea more precise.

In practice, the problem is a bit more complicated than we have described it. We have long sequences and we consider "windows" of length L beginning at a fixed position, say the jth. Thus, we consider words of length L in a long sequence, beginning at the jth position. For each possible pattern of length M, we ask how closely it can be matched in each of the sequences in a window of length L starting at the jth position. To formalize this, let Σ be a finite alphabet of size at least 2 and χ be a finite collection of words of length L on Σ. Let $F(\chi)$ be the set of words of length $M \geq 2$ that are our *consensus patterns*. Let $\chi = \{x_1, x_2, \ldots, x_p\}$. One way to define $F(\chi)$ is as follows. Let $d(a, b)$ be the best-mismatch distance. Consider nonnegative parameters λ_d that are monotone decreasing with d, and let

$F(x_1, x_2, \ldots, x_p)$ be all those words w of length M from Σ^M (all M-tuples from Σ) that maximize

$$s_\chi(w) = \sum_{i=1}^{p} \lambda_{d(w,x_i)}.$$

We call such an F a *Waterman consensus.* In particular, Waterman and others use the parameters $\lambda_d = (M-d)/M$.

As an example, we note that a frequently used alphabet is the purine/pyrimidine alphabet {R, Y}, where R = A or G and Y = C or T. For simplicity, it is easier to use the digits 0, 1 rather than the letters R, Y. Thus, let $\Sigma = \{0,1\}$, $M = 2$, and consider $F(x_1, x_2)$, where $x_1 = 111010$, $x_2 = 111111$. The possible pattern words are 00, 01, 10, 11. We have

$$\begin{array}{ll} d(00, x_1) = 1, & d(00, x_2) = 2, \\ d(01, x_1) = 0, & d(01, x_2) = 1, \\ d(10, x_1) = 0, & d(10, x_2) = 1, \\ d(11, x_1) = 0, & d(11, x_2) = 0. \end{array}$$

Thus,

$$\begin{aligned} s_\chi(00) &= \sum_{i=1}^{2} \lambda_{d(00,x_i)} = \lambda_1 + \lambda_2, \\ s_\chi(01) &= \sum_{i=1}^{2} \lambda_{d(01,x_i)} = \lambda_0 + \lambda_1, \\ s_\chi(10) &= \sum_{i=1}^{2} \lambda_{d(10,x_i)} = \lambda_0 + \lambda_1, \\ s_\chi(11) &= \sum_{i=1}^{2} \lambda_{d(11,x_i)} = \lambda_0 + \lambda_0. \end{aligned}$$

As long as $\lambda_0 > \lambda_1 > \lambda_2$, it follows that 11 is the consensus pattern, according to Waterman's consensus.

To give another example, let $\Sigma = \{0,1\}$, $M = 3$, and consider $F(x_1, x_2, x_3)$ where $x_1 = 000000$, $x_2 = 100000$, $x_3 = 111110$. The possible pattern words are 000, 001, 010, 011, 100, 101, 110, 111. We have

$$s_\chi(000) = \lambda_2 + 2\lambda_0, \quad s_\chi(001) = \lambda_2 + 2\lambda_1, \quad s_\chi(100) = 2\lambda_1 + \lambda_0, \text{ etc.}$$

Now, $\lambda_0 > \lambda_1 > \lambda_2$ implies that $s_\chi(000) > s_\chi(001)$. Similarly, one can show that s_χ is maximized by $s_\chi(000)$ or $s_\chi(100)$. Monotonicity doesn't say which of these is greater.

An alternative consensus procedure is to use a variant of the median procedure that gives all words w of length M that minimize

$$\sigma_\chi(w) = \sum_{i=1}^{p} d(w, x_i);$$

or we can use a variant of the mean procedure that gives all words w of length M that minimize

$$\tau_\chi(w) = \sum_{i=1}^{p} d(w, x_i)^2.$$

For instance, suppose that $\Sigma = \{0, 1\}$, $M = 2$, $\chi = \{x_1, x_2, x_3, x_4\}$, $x_1 = 1111$, $x_2 = 0000$, $x_3 = 1000$, and $x_4 = 0001$. Then the possible pattern words are 00, 01, 10, 11. We have

$$\sum_{i=1}^{4} d(00, x_i) = 2, \quad \sum_{i=1}^{4} d(01, x_i) = 3,$$
$$\sum_{i=1}^{4} d(10, x_i) = 3, \quad \sum_{i=1}^{4} d(11, x_i) = 4.$$

Thus, 00 is the median. However,

$$\sum_{i=1}^{4} d(00, x_i)^2 = 4, \quad \sum_{i=1}^{4} d(01, x_i)^2 = 3,$$
$$\sum_{i=1}^{4} d(10, x_i)^2 = 3, \quad \sum_{i=1}^{4} d(11, x_i)^2 = 6,$$

so the mean procedure leads to the two words 01 and 10, neither of which is a median.

Let us now consider the Waterman consensus with the special case of parameter $\lambda_d = (M - d)/M$ that is generally used in practice. Recall that

$$\sigma_\chi(w) = \sum_{i=1}^{p} d(w, x_i) \text{ and } s_\chi(w) = \sum_{i=1}^{p} \lambda_{d(w, x_i)} = p - \frac{1}{M} \sum_{i=1}^{p} d(w, x_i).$$

Thus, for fixed $M \geq 2$, Σ of size at least 2, and any size set χ of x_i's of length $L \geq M$, for all words w, w' of length M:

$$\sigma_\chi(w) \geq \sigma_\chi(w') \leftrightarrow s_\chi(w) \leq s_\chi(w').$$

It follows that for fixed $M \geq 2$, Σ of size at least 2, and any size set χ of x_i's of length $L \geq M$, there is a choice of the parameter λ_d so that the Waterman consensus is the same as the median. (This also holds for $M = 1$ or $|\Sigma| = 1$, but these are uninteresting cases.)

Similarly, one can show (see Exercises 32 and 33) that for fixed $M \geq 2$, Σ of size at least 2, and any size set χ of x_i's of length $L \geq M$, there is a choice of parameter λ_d so that for all words w, w' of length M:

$$\tau_\chi(w) \geq \tau_\chi(w') \leftrightarrow s_\chi(w) \leq s_\chi(w').$$

For this choice of λ_d, a word is a Waterman consensus if and only if it is a mean. It is surprising that the widely used Waterman consensus can actually be the mean or median in disguise.

EXERCISES FOR SECTION 10.3

1. In each of the following cases, find the Hamming distance between the two codewords x and y.

   (a) $x = 1010001, y = 0101010$ (b) $x = 11110011000, y = 11001001001$

   (c) $x = 10011001, y = 10111101$ (d) $x = 111010111010, y = 101110111011$

2. In the parity check code, find $E(\mathbf{a})$ if $\mathbf{a}$ is:

   (a) 111111 (b) 1001011

   (c) 001001001 (d) 01010110111

3. For each of the following codes C, find the number of errors that could be detected and the number that could be corrected using the nearest-neighbor rule.

   (a) $C = \{00000000, 11111110, 10101000, 01010100\}$

   (b) $C = \{000000000, 111111111, 111110000, 000001111\}$

   (c) $C = \{000000000, 111111111, 111100000, 000011111, 101010101, 010101010\}$

4. For each of the following binary codes C and received strings $x_1, x_2, \ldots, x_p$, find all medians and means.

   (a) C: $\{00000000, 11111110, 10101000, 01010100\}$
   x_i's: 00011000, 01000010, 01100110

   (b) C: $\{000000000, 111111111, 111110000, 000001111\}$
   x_i's: 000101000, 010010010, 011010110

   (c) C: $\{000000000, 111111111, 111100000, 000011111, 101010101, 010101010\}$
   x_i's: 100011001, 011000110, 000110101, 110000011, 010001010

5. Find the best-mismatch distance $d(a, b)$ for the following:

   (a) $a = 01$, $b = 1010$ (b) $a = 001$, $b = 101010101010101$

   (c) $a = 00$, $b = 1101011011$ (d) $a = 101$, $b = 100110$

6. Show that there is a set C of codewords of length n and a set of messages of length n so that the median procedure gives a unique solution and the mean procedure does not.

7. A *q-ary block n-code* (*q-ary code*) is a collection of strings of length n chosen from the alphabet $\{0, 1, \ldots, q-1\}$. The *Hamming distance* between two strings from this alphabet is again defined to be the number of digits on which they differ. For instance, if $q = 4$, then $d(0123, 1111) = 3$.

   (a) Find the Hamming distance between the following strings x and y.

      i. $x = 0226215, y = 2026125$

      ii. $x = 000111222333, y = 001110223332$

      iii. $x = 01010101010, y = 01020304050$

   (b) If the minimum distance between two codewords in a q-ary block n-code is d, how many errors can the code detect?

   (c) How many errors can the code correct under the nearest-neighbor rule?

8. Find an upper bound on the number of codewords in a code where each codeword has length 10 and the minimum distance between any two codewords is 7.
9. Repeat Exercise 8 for length 11 and minimum distance 6.
10. Find an upper bound on the number of codewords in a code C whose codewords have length 7 and which corrects up to 2 errors.
11. If codewords are bit strings of length 10, we have a binary symmetric channel, and the probability of error is .1, find the probability that if a codeword is sent there will be:

    (a) No errors
    (b) Exactly one error
    (c) Exactly two errors
    (d) More than two errors
12. Repeat Exercise 11 if the length of codewords is 6 and the probability of error is .001.
13. Suppose we assume that under transmission of a q-ary codeword (Exercise 7), the probability p of error is the same at each digit, independent of any previous errors. Moreover, if there is an error at the ith digit, the digit is equally likely to be changed into any one of the remaining $q-1$ symbols. We then refer to a *q-ary symmetric channel.*

    (a) In a q-ary symmetric channel, what is the probability of exactly t errors if a string of length n is sent?
    (b) If $q = 3$, what is the probability of receiving 22222 if 11111 is sent?
    (c) If $q = 4$, what is the probability of receiving 2013 if 1111 is sent?
14. Suppose that in a binary symmetric channel, codewords have length $n = 3$ and that the probability of error is .1. If we want a code which, with probability $\lambda \geq .95$, will correct a received message if it is in error, what must be the minimum distance between two codewords?
15. Repeat Exercise 14 if:

    (a) $n = 5$ and $\lambda \geq .50$
    (b) $n = 3$ and $\lambda \geq .90$
16. Suppose that in a binary triple repetition code, the value of k is 5 and the set $\mathcal{A} = A^5$.

    (a) How many errors can be detected? (b) How many errors can be corrected?
    (c) Suppose that instead, $k = 6$. How do your answers to parts (a) and (b) change?
17. Exercises 17–22 deal with q-ary codes (Exercise 7). Find a bound on the number of codewords in a q-ary code that is analogous to Theorem 10.3.
18. Suppose that a code is built up of strings of symbols using 0's, 1's, and 2's, and that S is a set of codewords with the following properties:

    (i) Each word in S has length n.
    (ii) Each word in S has a 2 in the first and last places and nowhere else.
    (iii) $d(a, b) = d$ for each pair of words a and b in S, where $d(a, b)$ is the Hamming distance.

    Let T be defined from S by changing 0 to 1 and 1 to 0 but leaving 2 unchanged. For example, the word 2102 becomes the word 2012.

(a) What is the distance between two words of T?

(b) What is the distance between a word of S and a word of T?

(c) Suppose that R consists of the words of S plus the words of T. How many errors can R detect? Correct?

19. Let A be a $q \times q$ Latin square on the set of symbols $\{0, 1, 2, \ldots, q-1\}$. Consider the q-ary code C consisting of all strings of length 3 of the form i, j, a_{ij}.

(a) If 5 is replaced by 0, find C corresponding to the Latin square of Table 1.4.

(b) Given the code C for an arbitrary $q \times q$ Latin square, how many errors can the code detect?

(c) How many errors can it correct?

20. Suppose that we have an orthogonal family of Latin squares of order q, $A^{(1)}$, $A^{(2)}$, $\ldots$, $A^{(r)}$, each using the symbols $0, 1, \ldots, q-1$. Consider the q-ary code S consisting of all strings of length $r+2$ of the form

$$i, j, a_{ij}^{(1)}, a_{ij}^{(2)}, \ldots, a_{ij}^{(r)}.$$

(a) Find S corresponding to the orthogonal family of Table 9.23.

(b) For arbitrary q, how many errors can the code S detect?

(c) How many errors can it correct?

21. Consider codewords of length 4, with each digit chosen from the alphabet {0, 1, 2, 3, 4, 5}. Show that there cannot be a set of 36 codewords each one of which has Hamming distance at least 3 from each other one.

22. Let A be a Latin square of order q on the set of symbols $\{0, 1, \ldots, q-1\}$ which is horizontally complete in the sense of Section 9.3, Exercise 5. Consider the q-ary code T consisting of all strings of length 4 of the form

$$i, j, a_{ij}, a_{i\ j+1}$$

for $i = 0, 1, \ldots, q-1$, $j = 0, 1, \ldots, q-2$.

(a) Find T corresponding to the Latin square constructed in Exercise 5, Section 9.3, in the case $n = 4$.

(b) For arbitrary q, how many errors can the code T detect?

(c) How many errors can it correct?

23. Suppose that we have a message of 10 digits and that $p = .1$ is the probability of an error.

(a) What is the probability of no errors in transmission if we use no code?

(b) What is the probability of correctly transmitting the message if we use the $k \to 3k$ triple repetition code with $k = 1$?

(c) What is this probability if we use the parity check code with blocks of size 2?

24. Suppose that we have a message of 1000 digits and that $p = .01$ is the probability of error. Suppose that we use the $k \to 5k$ five repetitions code with $k = 1$. What is the probability of transmitting the message correctly? (*Hint:* How many errors can be corrected?)
25. Suppose that majority rule decoding is used with a $k \to 3k$ repetition code, $k = 5$.
    (a) What is the "corrected" message if we receive *auvwbcuvzbcuvwd*?
    (b) If a given digit in a message is changed with probability $p = .1$, independent of its position in the message, what is the probability that a message of length $3k$ will be received without any errors?
    (c) Continuing with part (b), what is the probability of decoding a message of length $k = 5$ correctly if majority rule decoding is used?
    (d) Continuing with part (b), what number of repetitions would be needed to achieve a probability of at least .9999 of decoding a message of length $k = 5$ correctly if majority rule decoding is used?
26. An (n, d)-code C is called a *perfect t-error-correcting code* if $t = \lceil (d/2) - 1 \rceil$ and inequality (10.5) is an equality. Show that such a C *never* corrects more than t errors.
27. Show that the $1 \to 2t + 1$ repetition codes (Example 10.5) are perfect t-error-correcting codes.
28. It turns out (Tietäväinen [1973], van Lint [1971]) that there is only one perfect t-error-correcting code other than the repetition codes and the Hamming codes (to be introduced in Section 10.4.3). This is a $12 \to 23$ perfect 3-error correcting code due to Golay [1949]. How many codewords does this code have?
29. Suppose that C is a q-ary block n-code of minimum distance d. Generalize the Hamming bound (Theorem 10.3) to find an upper bound on $|C|$.
30. Let $\Sigma = \{0, 1\}$, $L = 6$, $M = 2$, and $\chi = \{110100, 010101\}$.
    (a) Calculate $s_\chi(w)$ for all possible words w in Σ^M.
    (b) Find the Waterman consensus if $\lambda_d = (M - d)/M$.
    (c) Calculate $\sigma_\chi(w)$ for all w in Σ^M.
    (d) Calculate $\tau_\chi(w)$ for all w in Σ^M.
    (e) Find all medians.
    (f) Find all means.
31. Repeat Exercise 30 if $\Sigma = \{0, 1\}$, $L = 6$, $M = 3$, and $\chi = \{000111, 011001, 100101\}$.
32. (Mirkin and Roberts [1993]) Show that if $\lambda_d = Ad + B$, $A < 0$, the Waterman consensus is the same as that chosen by the median procedure.
33. (Mirkin and Roberts [1993]) Show that if $\lambda_d = Ad^2 + B$, $A < 0$, the Waterman consensus is the same as that chosen by the mean procedure.
34. (Mirkin and Roberts [1993])
    (a) Suppose that $M \geq 2$, $|\Sigma| \geq 2$, $\lambda_1 < \lambda_0$, and $\lambda_d = Ad + B$, $A < 0$. Show that $\sigma_\chi(w) \geq \sigma_\chi(w') \leftrightarrow s_\chi(w) \leq s_\chi(w')$ for all words w, w' from Σ^M and all finite sets χ of *distinct* words from Σ^L, $L \geq M$.

(b) Interpret the conclusion.

35. (Mirkin and Roberts [1993])

(a) Suppose that $M \geq 2$, $|\Sigma| \geq 2$, $\lambda_1 < \lambda_0$, and $\lambda_d = Ad^2 + B$, $A < 0$. Show that $\tau_\chi(w) \geq \tau_\chi(w') \leftrightarrow s_\chi(w) \leq s_\chi(w')$ for all words w, w' from Σ^M and all finite sets χ of *distinct* words from Σ^L, $L \geq M$.

(b) Interpret the conclusion.

10.4 LINEAR CODES[8]

10.4.1 Generator Matrices

In this section we shall see how to use the encoding step to build error-detecting and error-correcting codes. Moreover, the encoding we present will have associated with it very efficient encoding and decoding procedures. The approach we describe was greatly influenced by the work of R. W. Hamming and D. Slepian in the 1950s. See, for example, the papers of Hamming [1950] and Slepian [1956a,b, 1960].

We first generalize Example 10.6. We can consider a $k \to n$ code E as adding $n-k$ *parity check digits* to a block of length k. In general, a *message block* $a_1 a_2 \cdots a_k$ is encoded as $x_1 x_2 \cdots x_n$, where $x_1 = a_1, x_2 = a_2, \ldots, x_k = a_k$ and the parity check digits are determined from the k *message digits* a_1, a_2, ..., a_k. The easiest way to obtain such an encoding is to generalize the matrix encoding method of Example 10.2. We let $\mathbf{M}$ be a $k \times n$ matrix, called the *generator matrix*, and we define $E(\mathbf{a})$ to be $\mathbf{aM}$.

Example 10.7 Suppose that $k = 3$, $n = 6$, and

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Then the message word 110 is encoded as $110\mathbf{M} = 110101$. The fifth digit is 0 because $1 + 1 = 0$: Recall that we are using addition modulo 2. Note that $\mathbf{M}$ begins with a 3×3 identity matrix. Since we want $a_i = x_i$ for $i = 1, 2, \ldots, k$, every generator matrix will begin with the $k \times k$ identity matrix $\mathbf{I}_k$. ■

To give some other examples, the parity check codes of Example 10.6 are defined by taking

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 1 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 1 & \cdots & 0 & 1 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{bmatrix}. \tag{10.6}$$

[8] This section may be omitted if the reader wants to go directly to Section 10.5.

We add a column of all 1's to the $k \times k$ identity matrix $\mathbf{I}_k$. The triple repetition code of Example 10.5 is obtained by taking $\mathbf{M}$ to be three copies of the $k \times k$ identity matrix.

If a code is defined by a generator matrix, decoding is trivial (when the transmission is correct). We simply decode $x_1 x_2 \cdots x_n$ as $x_1 x_2 \cdots x_k$; that is, we drop the $n - k$ parity check digits.

In general, codes definable by such generator matrices are called *linear codes*. This is because if $\mathbf{x} = x_1 x_2 \cdots x_n$ and $\mathbf{y} = y_1 y_2 \cdots y_n$ are codewords, so is the word $\mathbf{x} + \mathbf{y}$ whose ith digit is $x_i + y_i$, where addition is modulo 2. To see this, simply observe that if $\mathbf{aM} = \mathbf{x}$ and $\mathbf{bM} = \mathbf{y}$, then $(\mathbf{a} + \mathbf{b})\mathbf{M} = \mathbf{x} + \mathbf{y}$.[9]

It follows that $\mathbf{0} = 00 \cdots 0$ is a codeword of every (nonempty) linear code. For if $\mathbf{x}$ is any codeword, then since addition is modulo 2, $\mathbf{x} + \mathbf{x} = \mathbf{0}$. Indeed, it turns out that the set of all codewords under the operation $+$ defines a group, to use the language of Section 8.2 (see Exercise 26). Thus, linear codes are sometimes called *binary group codes.*

We next note that in a linear code, the minimum distance between two codewords is easy to find. Let the *Hamming weight* of a bit string $\mathbf{x}$, denoted $wt(\mathbf{x})$, be the number of nonzero digits of $\mathbf{x}$.

Theorem 10.5 In a linear code C, the minimum distance d between two codewords is equal to the minimum Hamming weight w of a codeword other than the codeword $\mathbf{0}$.

Proof. Note that if $d(\mathbf{x}, \mathbf{y})$ is Hamming distance, then since C is linear, $\mathbf{x} + \mathbf{y}$ is in C, so

$$d(\mathbf{x}, \mathbf{y}) = wt(\mathbf{x} + \mathbf{y}).$$

This conclusion uses the fact that addition is modulo 2. Suppose that the minimum distance d is given by $d(\mathbf{x}, \mathbf{y})$ for a particular $\mathbf{x}$, $\mathbf{y}$ in C. Then

$$d = d(\mathbf{x}, \mathbf{y}) = wt(\mathbf{x} + \mathbf{y}) \geq w.$$

Next, suppose that $\mathbf{u} \neq \mathbf{0}$ in C has minimum weight w. Then since $\mathbf{0} = \mathbf{u} + \mathbf{u}$ is in C, we have

$$w = wt(\mathbf{u}) = wt(\mathbf{u} + \mathbf{0}) = d(\mathbf{u}, \mathbf{0}) \geq d. \qquad \text{Q.E.D.}$$

It follows by Theorem 10.5 that in the code of Example 10.6, the minimum distance between two codewords is 2, since the minimum weight of a codeword other than $\mathbf{0}$ is 2. This weight is attained, for example, in the string 11000. The minimum distance is attained, for example, as

$$d(11000, 00000).$$

[9] We use the assumption, which we shall make throughout this section, that $\mathcal{A} = A^k$.

10.4.2 Error Correction Using Linear Codes

If $\mathbf{M}$ is a generator matrix for a linear code, it can be represented schematically as $[\mathbf{I}_k\mathbf{G}]$, where $\mathbf{G}$ is a $k \times (n-k)$ matrix. Let $\mathbf{G}^T$ be the transpose of matrix $\mathbf{G}$ and let $\mathbf{H} = [\mathbf{G}^T\mathbf{I}_{n-k}]$ be an $(n-k) \times n$ matrix, called the *parity check matrix*. The matrix $\mathbf{G}^T$ in Example 10.6 is a row vector of k 1's, and the matrix $\mathbf{H}$ is a row vector of $(k+1)$ 1's. In Example 10.5, with three repetitions and message blocks of length $k = 4$, we have

$$\mathbf{G}^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{10.7}$$

In Example 10.7,

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \tag{10.8}$$

Note that a linear code can be defined by giving either the matrix $\mathbf{G}$ from the generator matrix $\mathbf{M}$ or the parity check matrix $\mathbf{H}$. For we can derive $\mathbf{H}$ from $\mathbf{G}$ and $\mathbf{G}$ from $\mathbf{H}$. We shall see that parity check matrices will provide a way to detect and correct errors. The basic ideas of this approach are due to Slepian [1956a,b, 1960].

First, we note that the parity check matrix can be used to identify codewords.

Theorem 10.6 In a linear code, a block $\mathbf{a} = a_1a_2\cdots a_k$ is encoded as $\mathbf{x} = x_1x_2\cdots x_n$ if and only if $a_i = x_i$ for $i \leq k$ and $\mathbf{H}\mathbf{x}^T = \mathbf{0}$.

Proof.[10] By definition of the encoding, if block $\mathbf{a} = a_1a_2\cdots a_k$ is encoded as $\mathbf{x} = x_1x_2\cdots x_n$, then

$$(a_1a_2\cdots a_k)[\mathbf{I}_k\mathbf{G}] = (x_1x_2\cdots x_n).$$

[10] The proof may be omitted.

Now, clearly, $a_i = x_i, i \leq k$. Also,

$$\begin{aligned}
\mathbf{Hx}^T &= \mathbf{H}(\mathbf{a}[\mathbf{I}_k\mathbf{G}])^T \\
&= \mathbf{H}[\mathbf{I}_k\mathbf{G}]^T\mathbf{a}^T \\
&= [\mathbf{G}^T\mathbf{I}_{n-k}]\begin{bmatrix} \mathbf{I}_k \\ \mathbf{G}^T \end{bmatrix}\mathbf{a}^T \\
&= (\mathbf{G}^T + \mathbf{G}^T)\mathbf{a}^T \\
&= \mathbf{0}\mathbf{a}^T \\
&= \mathbf{0},
\end{aligned}$$

where $\mathbf{G}^T + \mathbf{G}^T = \mathbf{0}$ since addition is modulo 2.

Conversely, suppose that $a_i = x_i, i \leq k$, and $\mathbf{Hx}^T = \mathbf{0}$. Now suppose that $\mathbf{a}$ is encoded as $\mathbf{y} = y_1y_2\cdots y_n$. Then $a_i = y_i, i \leq k$, so $x_i = y_i, i \leq k$. Also, by the first part of the proof, $\mathbf{Hy}^T = \mathbf{0}$. It follows that $x = y$. To see why, note that the equations $\mathbf{Hx}^T = \mathbf{0}$ and $\mathbf{Hy}^T = \mathbf{0}$ each give rise to a system of equations. The jth equation in the first case involves at most the variables $x_1, x_2, \ldots, x_k$ and x_{k+j}. Thus, since $x_1, x_2, \ldots, x_k$ are given by $a_1, a_2, \ldots, a_k$, the jth equation defines x_{k+j} uniquely in terms of the k message digits $a_1, a_2, \ldots, a_k$. The same is true of y_{k+j}. Thus, $x_{k+j} = y_{k+j}$. Q.E.D.

Corollary 10.6.1 A bit string $\mathbf{x} = x_1x_2\cdots x_n$ is a codeword if and only if $\mathbf{Hx}^T = \mathbf{0}$.

Corollary 10.6.2 There is a unique bit string $\mathbf{x}$ so that $x_i = a_i, i \leq k$, and $\mathbf{Hx}^T = \mathbf{0}$. This bit string $\mathbf{x}$ is the encoding of $a_1a_2\cdots a_k$. An expression for the entry x_{k+j} of $\mathbf{x}$ in terms of $a_1, a_2, \ldots, a_k$ is obtained by multiplying the jth row of $\mathbf{H}$ by $\mathbf{x}^T$.

Proof. This is a corollary of the proof. Q.E.D.

To illustrate these results, note that in the parity check codes, $H = (11\cdots 1)$ with $n = k+1$ 1's. Then $\mathbf{Hx}^T = \mathbf{0}$ if and only if

$$x_1 + x_2 + \cdots + x_{k+1} = 0. \tag{10.9}$$

But since addition is modulo 2, (10.9) is exactly equivalent to the condition that

$$x_{k+1} = x_1 + x_2 + \cdots + x_k,$$

which defines codewords. This equation is called the *parity check equation.* Similarly, in the triple repetition codes with $k = 4$, we see from (10.7) that $\mathbf{Hx}^T = \mathbf{0}$ if and only if

$$\begin{array}{llll} x_1 + x_5 = 0, & x_2 + x_6 = 0, & x_3 + x_7 = 0, & x_4 + x_8 = 0, \\ x_1 + x_9 = 0, & x_2 + x_{10} = 0, & x_3 + x_{11} = 0, & x_4 + x_{12} = 0. \end{array} \tag{10.10}$$

Since addition is modulo 2, Equations (10.10) are equivalent to

$$\begin{array}{llll} x_1 = x_5, & x_2 = x_6, & x_3 = x_7, & x_4 = x_8, \\ x_1 = x_9, & x_2 = x_{10}, & x_3 = x_{11}, & x_4 = x_{12}, \end{array}$$

which define a codeword. These equations are called the *parity check equations.* Finally, in Example 10.7, by (10.8), $\mathbf{Hx}^T = \mathbf{0}$ if and only if

$$\begin{aligned} x_1 + x_3 + x_4 &= 0, \\ x_1 + x_2 + x_5 &= 0, \\ x_2 + x_3 + x_6 &= 0. \end{aligned}$$

Using the fact that addition is modulo 2, these equations are equivalent to the following parity check equations, which determine the parity check digits x_{k+j} in terms of the message digits $x_i, i \leq k$:

$$\begin{aligned} x_4 &= x_1 + x_3, \\ x_5 &= x_1 + x_2, \\ x_6 &= x_2 + x_3. \end{aligned}$$

Thus, as we have seen before, 110 is encoded as 110101, since $x_4 = 1 + 0 = 1, x_5 = 1 + 1 = 0$, and $x_6 = 1 + 0 = 1$.

In general, by Corollary 10.6.2, the equations $\mathbf{Hx}^T = \mathbf{0}$ define x_{k+j} in terms of $a_1, a_2, \ldots, a_k$. The equations giving x_{k+j} in these terms are the *parity check equations.*

Theorem 10.6 and Corollary 10.6.1 provide a way to detect errors. We simply note that if $\mathbf{x}$ is received and if $\mathbf{Hx}^T \neq \mathbf{0}$, then an error occurred. The parity check matrix allows us not only to detect errors, but to correct them, as the next theorem shows.

Theorem 10.7 Suppose that the columns of the parity check matrix $\mathbf{H}$ are all nonzero and all distinct. Suppose that a codeword $\mathbf{y}$ is transmitted and a word $\mathbf{x}$ is received. If $\mathbf{x}$ differs from $\mathbf{y}$ only on the ith digit, then $\mathbf{Hx}^T$ is the ith column of $\mathbf{H}$.

Proof. Note that by Corollary 10.6.1, $\mathbf{Hy}^T = \mathbf{0}$. Now $\mathbf{x}$ can be written as $\mathbf{y}+\mathbf{e}$, where $\mathbf{e}$ is an error string, a bit string with 1's in the digits that differ from $\mathbf{y}$. (Recall that addition is modulo 2.) We conclude that

$$\mathbf{Hx}^T = \mathbf{H}(\mathbf{y}+\mathbf{e})^T = \mathbf{H}(\mathbf{y}^T + \mathbf{e}^T) = \mathbf{Hy}^T + \mathbf{He}^T = \mathbf{He}^T.$$

Now if $\mathbf{e}$ is the vector with all 0's except a 1 in the ith digit, then $\mathbf{He}^T$ is the ith column of $\mathbf{H}$. Q.E.D.

To illustrate this result, suppose that we have a triple repetition code and $k = 4$. Then the parity check matrix $\mathbf{H}$ is given by (10.7). Suppose that a codeword $\mathbf{y} = 110111011101$ is sent and a codeword $\mathbf{x} = 110011011101$, which differs on the fourth digit, is received. Then note that

$$\mathbf{H}\mathbf{x}^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

which is the fourth column of $\mathbf{H}$.

In general, error correction using the parity check matrix $\mathbf{H}$ will proceed as follows, assuming that $\mathbf{H}$ has distinct columns and all are nonzero. Having received a block $\mathbf{x}$, compute $\mathbf{H}\mathbf{x}^T$. If this is $\mathbf{0}$, and errors in transmission are unlikely, it is reasonable to assume that $\mathbf{x}$ is correct. If it is not $\mathbf{0}$, but is the ith column of $\mathbf{H}$, and if errors are unlikely, it is reasonable to assume that only one error was made, so the correct word differs from $\mathbf{x}$ on the ith digit. If $\mathbf{H}\mathbf{x}^T$ is not $\mathbf{0}$ and not a column of $\mathbf{H}$, at least two errors occurred in transmission and error correction cannot be carried out this way.

10.4.3 Hamming Codes

Recall that for error correction, we want the $(n-k) \times n$ parity check matrix $\mathbf{H}$ to have columns that are nonzero and distinct. Now if $p = n - k$, the columns of $\mathbf{H}$ are bit strings of length p. There are 2^p such strings, $2^p - 1$ nonzero ones. The *Hamming code* $\mathcal{H}_p$ arises when we take $\mathbf{H}$ to be a matrix whose columns are all of these $2^p - 1$ nonzero bit strings of length p, arranged in any order. Technically, to conform with our definition, the last $n-k$ columns should form the identity matrix. The resulting code is a $k \to n$ code, where $n = 2^p - 1$ and $k = n - p = 2^p - 1 - p$. For instance, if $p = 2$, then $n = 2^2 - 1 = 3$ and a typical $\mathbf{H}$ is given by

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

It is easy to see that $\mathbf{H}$ defines a $1 \to 3$ triple repetition code. If $p = 3$, then $n = 2^3 - 1 = 7$, and a typical $\mathbf{H}$ is given by

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

We get a $4 \to 7$ code. The Hamming codes were introduced by Hamming [1950] and Golay [1949]. [Note that if we change the order of columns in $\mathbf{H}$, we get (essentially) the same code or set of codewords. For any two such codes are seen to be equivalent (in the sense to be defined in Exercise 25) by changing the order of the parity check digits. That is why we speak of *the* Hamming code $\mathcal{H}_p$.]

Theorem 10.8 In the Hamming codes $\mathcal{H}_p, p \geq 2$, the minimum distance d between two codewords is 3.

Proof. Since the columns of the parity check matrix $\mathbf{H}$ are nonzero and distinct, single errors can be corrected (see the discussion following Theorem 10.7). Thus, by Theorem 10.2, $d \geq 3$. Now it is easy to show that for $p \geq 2$, there are always three nonzero bit strings of length p whose sum (under addition modulo 2) is zero (Exercise 23). If these occur as columns u, v, and w of the matrix $\mathbf{H}$, we take $\mathbf{x}$ to be the vector that is 1 in positions u, v, and w, and 0 otherwise. Then $\mathbf{Hx}^T$ is the sum of the uth, vth, and wth columns of $\mathbf{H}$, and so is $\mathbf{0}$. Thus, $\mathbf{x}$ is a codeword of weight 3. By Theorem 10.5, $d \leq 3$. Q.E.D.

It follows from Theorem 10.8 that the Hamming codes *always* detect up to two errors and correct up to one. In Exercise 27 we ask the reader to show that the Hamming codes can *never* correct more than one error.

EXERCISES FOR SECTION 10.4

1. Suppose that
$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$
Using $\mathbf{M}$, find the codeword $x_1x_2 \cdots x_n$ corresponding to each of the following message words $a_1a_2 \cdots a_k$.

   (a) 11 (b) 10 (c) 01 (d) 00

2. Suppose that
$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$
Repeat Exercise 1 for the following message words.

   (a) 111 (b) 101 (c) 000

3. Suppose that
$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$
Repeat Exercise 1 for the following message words.

   (a) 1111 (b) 1000 (c) 0001

4. Find the linear code C generated by the matrices $\mathbf{M}$ of:

   (a) Exercise 1 (b) Exercise 2 (c) Exercise 3

5. Find a generator matrix for the code that translates 0 into 000000 and 1 into 111111.

6. Each of the following codes is linear. For each, find the minimum distance between two codewords.

   (a) 000000, 001001, 010010, 100100, 011011, 101101, 110110, 111111

   (b) 0000, 0011, 0101, 1001, 0110, 1010, 1100, 1111

   (c) 00000, 00011, 00101, 01001, 10001, 00110, 01010, 01100, 10010, 10100, 11000, 01111, 11011, 10111, 11101, 11110

   (d) 11111111, 10101010, 11001100, 10011001, 11110000, 10100101, 11000011, 10010110, 00000000, 01010101, 00110011, 01100110, 00001111, 01011010, 00111100, 01101001

7. Find the parity check matrix $\mathbf{H}$ corresponding to $\mathbf{M}$ of:

   (a) Exercise 1 (b) Exercise 2 (c) Exercise 3

8. Find the generator matrix $\mathbf{M}$ corresponding to the following parity check matrices:

$$\text{(a) } \mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(b) } \mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{(c) } \mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

9. Find the parity check equations in each of the following:

   (a) Exercise 1 (b) Exercise 2 (c) Exercise 3
   (d) Exercise 8(a) (e) Exercise 8(b) (f) Exercise 8(c)

10. Suppose that we use the $k \to 5k$ five repetitions code with $k = 2$. Find the parity check matrix $\mathbf{H}$ and derive the parity check equations.

11. For the matrix $\mathbf{H}$ of Exercise 8(a), suppose that a word $\mathbf{x}$ is received over a noisy channel. Assume that errors are unlikely. For each of the following $\mathbf{x}$, determine if an error was made in transmission.

    (a) $\mathbf{x} = 111000$ (b) $\mathbf{x} = 111100$ (c) $\mathbf{x} = 000100$

12. Repeat Exercise 11 for $\mathbf{H}$ of Exercise 8(b) and

    (a) $\mathbf{x} = 11101$ (b) $\mathbf{x} = 01101$ (c) $\mathbf{x} = 00011$

13. Repeat Exercise 11 for $\mathbf{M}$ of Exercise 1 and

    (a) $\mathbf{x} = 1111$ (b) $\mathbf{x} = 1000$ (c) $\mathbf{x} = 0100$

14. In each part of Exercise 11, if possible, correct an error if there was one.

15. In each part of Exercise 12, if possible, correct an error if there was one.

16. Consider the parity check matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Find all the codewords in the corresponding code. Does the code correct all single errors?

17. In the Hamming code $\mathcal{H}_p$ with $p = 3$, encode the following words.

(a) 1001 (b) 0001 (c) 1110

18. For the Hamming code $\mathcal{H}_p$ with $p = 4$, find a parity check matrix and encode the word 10000000000.

19. Find all codewords in the Hamming code $\mathcal{H}_2$.

20. Find all codewords in the Hamming code $\mathcal{H}_3$.

21. Find a word of weight 3 in the Hamming code $\mathcal{H}_p$, for every $p \geq 2$.

22. Consider a codeword $x_1x_2\cdots x_n$ in the Hamming code $\mathcal{H}_p$, $p \geq 2$. Let $k = n - p$.

(a) Show that if $x_i = 0$, all $i \leq k$, then $x_i = 0$, all i.

(b) Show that it is impossible to have exactly one x_i equal to 1 and all other x_i equal to 0.

23. Show that if $p \geq 2$, there are always three bit strings of length p whose sum (under addition modulo 2) is zero.

24. Let C be a linear code with codewords of length n in which some codewords have odd weight. Form a new code C' by adding 0 at the end of each word of C of even weight and 1 at the end of each word of C of odd weight.

(a) If $\mathbf{M}$ is the parity check matrix for C, show that the parity check matrix for C' is given by

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ & & & 0 \\ & \mathbf{M} & & 0 \\ & & & \vdots \\ & & & 0 \end{bmatrix}.$$

(b) Show that the distance between any two codewords of C' is even.

(c) Show that if the minimum distance d between two codewords of C was odd, the minimum distance between two codewords of C' is $d + 1$.

(d) For all $p \geq 2$, find a linear code with codewords of length $n = 2^p$ and with minimum distance between two codewords equal to 4.

25. Two n-codes C and C' are *equivalent* if they differ only in the order of symbols, that is, if one can be obtained from the other by applying the same permutation to each element. For example, the following C and C' are equivalent:

C	C'
0000	0000
1111	1111
0011	0101
1100	1010

Show that if $\mathbf{M}$ is a generator matrix giving rise to a code C, and $\mathbf{M}'$ is obtained from $\mathbf{M}$ by interchanging rows in the matrix $\mathbf{G}$ corresponding to $\mathbf{M}$, and $\mathbf{M}'$ gives rise to a code C', then C and C' are equivalent.

26. Show that under a linear code, the set of codewords under the operation + defines a group.
27. Show that the Hamming codes $\mathcal{H}_p$, $p \geq 2$, are perfect 1-error-correcting codes. (See Exercise 26, Section 10.3 for the definition of perfect t-error-correcting codes.)
28. We can extend the theory of linear codes from binary codes to q-ary codes (Exercises 7, 13, and 17–22, Section 10.3), where q is a power of a prime. We continue to define a codeword from a message word by adding parity check digits, and specifically use a generator matrix $\mathbf{M}$, but with addition in the finite field $GF(q)$, rather than modulo 2. Note that most of our results hold in this general setting, with the major exceptions being noted in Exercises 29 and 30. Suppose that $q = 5$.

    (a) If $\mathbf{M}$ is as in Exercise 1, find the codeword corresponding to the message words:

    i. 14 ii. 03 iii. 13

    (b) If $\mathbf{M}$ is as in Exercise 2, find the codeword corresponding to the message words:

    i. 124 ii. 102 iii. 432

29. Continuing with Exercise 28

    (a) Note that $d(\mathbf{x}, \mathbf{y})$ is not necessarily $wt(\mathbf{x} + \mathbf{y})$.

    (b) What is the relation between distance and weight?

30. Continuing with Exercise 28, show that Theorem 10.6 may not be true.
31. Continuing with Exercise 28, show that Theorem 10.5 still holds.
32. Suppose that C is a q-ary block n-code of minimum distance d. Generalize the Hamming bound (Theorem 10.3) to find an upper bound on $|C|$.

10.5 THE USE OF BLOCK DESIGNS TO FIND ERROR-CORRECTING CODES[11]

10.5.1 Hadamard Codes

One way to find error-correcting codes is to concentrate first on finding a rich set C of codewords and then perform an encoding of messages into C. This is the opposite of the approach we have taken so far, which first defined the encoding, and defined the set C to be the set of all the words arising from the encoding. Our goal will be to find a set C of codewords that corrects a given number of errors or, equivalently, has a given minimum distance d between codewords, and which has many codewords in it, thus allowing more possible message blocks to be encoded.

Recall that an (n, d)-code has codewords of length n and the minimum distance between two codewords is d. A useful way to build (n, d)-codes C is to use the

[11] This section depends on Section 9.4.

incidence matrix of a (v, k, λ)-design (see Section 9.4.2). Each row of this incidence matrix has $r = k$ 1's and the rest 0's. The rows have length $b = v$. Any two rows have 1's in common in a column exactly λ times. The rows can define codewords for an (n, d)-code, with $n = v$. What is d? Let us consider two rows, the ith and jth. There are λ columns where there is a 1 in both rows. There are k 1's in each row, and hence there are $k - \lambda$ columns where row i has 1 and row j has 0, and there are $k - \lambda$ columns where row i has 0 and row j has 1. All other columns have 0's in both rows. It follows that the two rows differ in $2(k - \lambda)$ places. This is true for every pair of rows, so

$$d = 2(k - \lambda).$$

Theorem 9.17 says that for arbitrarily large m, there are Hadamard designs of dimension m, that is, $(4m - 1, 2m - 1, m - 1)$-designs. It follows that we can find (v, k, λ)-designs with arbitrarily large $k - \lambda$, and hence (n, d)-codes for arbitrarily large d. For given a Hadamard design of dimension m, we have

$$d = 2\,(k - \lambda) = 2\,[(2m - 1) - (m - 1)] = 2m.$$

Hence, if there are Hadamard designs of dimension m for arbitrarily large m, there are error-correcting codes that will detect up to $d-1 = 2m-1$ errors and correct up to $\lceil (d/2) - 1 \rceil = m-1$ errors for arbitrarily large m. These codes are $(4m-1, 2m)$-codes, since each codeword has length $4m - 1$. We call them *Hadamard codes.* We shall first set out to prove the existence of Hadamard designs of arbitrarily large dimension m, that is, to prove Theorem 9.17. Then we shall ask how rich are the codes constructed from the incidence matrices of Hadamard designs; that is, how do these codes compare to the richest possible (n, d)-codes?

10.5.2 Constructing Hadamard Designs[12]

The basic idea in constructing Hadamard designs is that certain kinds of matrices will give rise to the incidence matrices of these designs. An $n \times n$ matrix $\mathbf{H} = (h_{ij})$ is called a *Hadamard matrix of order* n if h_{ij} is $+1$ or -1 for every i and j and if

$$\mathbf{H}\mathbf{H}^T = n\mathbf{I},$$

where $\mathbf{H}^T$ is the transpose of $\mathbf{H}$ and $\mathbf{I}$ is the $n \times n$ identity matrix.[13] The matrix $n\mathbf{I}$ has n's down the diagonal, and 0's otherwise. To give an example, suppose that

$$\mathbf{H} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}. \tag{10.11}$$

Then

$$\mathbf{H}^T = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

[12] This subsection may be omitted.

[13] Hadamard matrices were introduced by Hadamard [1893] and, in an earlier use, by Sylvester [1867]. Plotkin [1960] and Bose and Shrikhande [1959] constructed binary codes from Hadamard matrices.

and

$$\mathbf{H}\mathbf{H}^T = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}.$$

Hence, the matrix $\mathbf{H}$ of (10.11) is a Hadamard matrix of order 2. A Hadamard matrix of order 4 is given by

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix}. \tag{10.12}$$

For it is easy to check that

$$\mathbf{H}\mathbf{H}^T = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}.$$

An equivalent definition of Hadamard matrix is the following. Recall that if

$$(x_1, x_2, \ldots, x_n) \text{ and } (y_1, y_2, \ldots, y_n)$$

are two vectors, their *inner product* is defined to be

$$\sum_i x_i y_i.$$

Then an $n \times n$ matrix of $+1$'s and -1's is a Hadamard matrix if for all i, the inner product of the ith row with itself is n, and for all $i \neq j$, the inner product of the ith row with the jth is 0. This is just a restatement of the definition. To illustrate, let us consider the matrix $\mathbf{H}$ of (10.11). The first row is the vector $(1, 1)$ and the second is the vector $(-1, 1)$. The inner product of the first row with itself is

$$1 \cdot 1 + 1 \cdot 1 = 2.$$

The inner product of the first row with the second is

$$1 \cdot (-1) + 1 \cdot 1 = 0,$$

and so on.

A Hadamard matrix is called *normalized* if the first row and first column consist of just $+1$'s. For example, the matrix

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{10.13}$$

is a normalized Hadamard matrix. So is the matrix

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}. \tag{10.14}$$

Some of the most important properties of Hadamard matrices are summarized in the following theorem.

Theorem 10.9 If $\mathbf{H}$ is a normalized Hadamard matrix of order $n > 2$, then $n = 4m$ for some m. Moreover, each row (column) except the first has exactly $2m$ $+1$'s and $2m$ -1's, and for any two rows (columns) other than the first, there are exactly m columns (rows) in which both rows (columns) have $+1$.

We prove Theorem 10.9 later in this subsection. However, let us see how Theorem 10.9 gives rise to a proof of Theorem 9.17. Given a normalized Hadamard matrix, we can define a (v, k, λ)-design. We do so by deleting the first row and column and, in what is left, replacing every -1 by a 0. As we shall see, this gives an incidence matrix $\mathbf{A}$ of a (v, k, λ)-design. Performing this procedure on the normalized Hadamard matrix of (10.14) first gives

$$\begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & 1 \end{bmatrix}$$

and then the incidence matrix

$$\mathbf{A} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \!\!\begin{array}{c} \begin{array}{ccc} B_1 & B_2 & B_3 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}.$$

This gives rise to the blocks

$$B_1 = \{2\}, \quad B_2 = \{1\}, \quad B_3 = \{3\},$$

and to a design with $v = 3$, $k = 1$, and $\lambda = 0$. (This is, technically, not a design, since we have required that $\lambda > 0$. However, it illustrates the procedure.)

To show that this procedure always gives rise to a (v, k, λ)-design, let us note that by Theorem 10.9, the incidence matrix $\mathbf{A}$ has $4m-1$ rows and $4m-1$ columns, so $b = v = 4m - 1$. Also, one 1 (the first one) has been removed from each row, so each row of $\mathbf{A}$ has $2m-1$ 1's and $r = 2m-1$. By a similar argument, each column of $\mathbf{A}$ has $2m-1$ 1's and $k = 2m-1$. Finally, any two rows had a pair of 1's in the first column in common, so now have one fewer pair, namely $m - 1$, in common. Hence, $\lambda = m-1$. Thus, we have a (v, k, λ)-design with $v = 4m-1, k = 2m-1, \lambda = m-1$, provided that $m \geq 2$.[14]

In fact, the procedure we have described can be reversed. We have the following theorem.

Theorem 10.10 There exists a Hadamard design of dimension m if and only if there exists a Hadamard matrix of order $4m$.

[14] If $m = 1$, λ turns out to be 0, as we have seen.

Proof. It remains to start with a Hadamard design of dimension m and construct a Hadamard matrix of order $4m$ from it. Let $\mathbf{A}$ be the incidence matrix of such a design. Construct $\mathbf{H}$ by changing 0 to -1, adding a row of 1's at the top, and adding a column of 1's at the front. It is easy to verify that $\mathbf{H}$ is a Hadamard matrix. The proof is left as an exercise (Exercise 14). Q.E.D.

Let us next show how to construct normalized Hadamard matrices of order $4m$ for arbitrarily large m. This will prove Theorem 9.17. Suppose that $\mathbf{H}$ is an $n \times n$ Hadamard matrix. Let $\mathbf{K}$ be the matrix

$$\mathbf{K} = \begin{bmatrix} \mathbf{H} & \mathbf{H} \\ \mathbf{H} & -\mathbf{H} \end{bmatrix},$$

where $-\mathbf{H}$ is the matrix obtained from $\mathbf{H}$ by multiplying each entry by -1. For example, if $\mathbf{H}$ is the matrix of (10.13), $\mathbf{K}$ is the matrix

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

$\mathbf{K}$ is always a Hadamard matrix of order $2n$, for each entry is $+1$ or -1. Moreover, it is easy to show that the inner product of a row with itself is always $2n$, and the inner product of two different rows is always 0. Finally, we observe that if $\mathbf{H}$ is normalized, so is $\mathbf{K}$. Thus, there are normalized Hadamard matrices of arbitrarily large orders, and in particular of all orders 2^p for $p \geq 1$. A Hadamard matrix of order $4m$ for $m \geq 2$ gives rise to a $(4m-1, 2m-1, m-1)$-design. Since $4m = 2^p$, $m \geq 2$ is equivalent to $m = 2^k, k \geq 1$, this completes the proof of Theorem 9.17.

It is interesting to go through the construction we have just gone through if the matrix $\mathbf{H}$ is the matrix shown in (10.14). Then the corresponding matrix $\mathbf{K}$ is given by

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}. \tag{10.15}$$

The corresponding incidence matrix for a (v, k, λ)-design is given by deleting the first row and column and changing the remaining -1's to 0's. We get

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}. \tag{10.16}$$

This is the incidence matrix for a symmetric BIBD with $v = 7, k = 3$, and $\lambda = 1$. It defines a design different from that of Example 9.10. Repeating the construction one more time gives a 16×16 Hadamard matrix and hence a $(15, 7, 3)$-design. It is this design whose blocks are shown in Table 9.27. A proof is left to the reader as Exercise 8.

The construction procedure we have outlined certainly leads to normalized Hadamard matrices of order $4m$ for arbitrarily large m, in particular for $m = 1, 2, 4, 8, \ldots$, and so of orders $4, 8, 16, 32, \ldots$. Notice that we do not claim to construct normalized Hadamard matrices of order $4m$ for every m. It is conjectured that for every m, a normalized Hadamard matrix of order $4m$ exists. Whether or not this conjecture is true is another open question of the mathematics of combinatorial design and coding. Note that if there is any Hadamard matrix of order $4m$, there is a normalized one (Exercise 6). (For other methods of constructing Hadamard matrices, see, for example, Hall [1967] or MacWilliams and Sloane [1983]. Craigen and Wallis [1993] is a survey article on Hadamard matrices marking the 100th anniversary of their introduction.)

We conclude this subsection by filling in the one missing step in the proof of Theorem 9.17, namely, by proving Theorem 10.9.[15] We need one preliminary result.

Theorem 10.11 If $\mathbf{H}$ is a Hadamard matrix, so is $\mathbf{H}^T$.

Proof. If $\mathbf{H}\mathbf{H}^T = n\mathbf{I}$, then

$$\frac{\mathbf{H}}{\sqrt{n}} \frac{\mathbf{H}^T}{\sqrt{n}} = \mathbf{I},$$

where $\mathbf{H}/\sqrt{n}$ is obtained from $\mathbf{H}$ by dividing each entry by $\sqrt{n}$, and similarly for $\mathbf{H}^T/\sqrt{n}$. Since $\mathbf{AB} = \mathbf{I}$ implies that $\mathbf{BA} = \mathbf{I}$ for square matrices $\mathbf{A}$ and $\mathbf{B}$, we have

$$\frac{\mathbf{H}^T}{\sqrt{n}} \frac{\mathbf{H}}{\sqrt{n}} = \mathbf{I}$$

or

$$\mathbf{H}^T\mathbf{H} = n\mathbf{I}.$$

Since $(\mathbf{H}^T)^T = \mathbf{H}$, it follows that $\mathbf{H}^T$ is Hadamard. Q.E.D.

Proof of Theorem 10.9. Let $\mathbf{H}$ be a normalized Hadamard matrix of order n. The results for columns follow from the results for rows by applying Theorem 10.11. Hence, we need only prove the row results. Since the first row of $\mathbf{H}$ is

$$1 \quad 1 \quad 1 \quad \cdots \quad 1$$

and since the inner product with any other row is 0, any other row must have an equal number of $+1$'s and -1's. Thus, n is even, and there are $(n/2)$ $+1$'s and $(n/2)$ -1's. Interchange columns so that the second row has $+1$'s coming first and then -1's. Thus, the first two rows look like this:

$$\begin{array}{cccccccc} 1 & 1 & \cdots & 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 & -1 & -1 & \cdots & -1 \end{array}$$

[15] The rest of this subsection may be omitted.

Second Row	1	1	-1	-1
ith row, $i \neq 1, 2$	1	-1	1	-1
	u digits	$\frac{n}{2} - u$ digits	v digits	$\frac{n}{2} - v$ digits

Figure 10.2: Schematic for computing the inner product of the second and ith rows ($i \neq 1, 2$) in a normalized Hadamard matrix.

This interchange does not affect the size of the matrix or the number of $+1$'s in each row or the number of columns where two rows share a $+1$. Consider the ith row, $i \neq 1, 2$. The ith row has u digits which are $+1$ and $n/2 - u$ digits which are -1 in the first half (the first $n/2$ entries), and v digits which are $+1$ and $n/2 - v$ digits which are -1 in the second half. The first row and ith row have inner product 0. Hence, the ith row has $(n/2)$ $+1$'s and

$$u + v = \frac{n}{2}. \tag{10.17}$$

The second and ith rows have inner product 0 if $i \neq 1, 2$. Hence,

$$u - \left(\frac{n}{2} - u\right) - v + \left(\frac{n}{2} - v\right) = 0 \tag{10.18}$$

(see Figure 10.2). It follows that

$$u - v = 0. \tag{10.19}$$

Equations (10.17) and (10.19) give us

$$2u = \frac{n}{2}$$

or

$$n = 4u.$$

Thus, n is a multiple of 4, which proves the first part of Theorem 10.9. Moreover, there are $(n/2 = 2u)$ $+1$'s in each row other than the first, proving another part of Theorem 10.9. Finally, the second and ith rows, $i \neq 1, 2$, have exactly u columns in which $+1$'s appear in common. The same can be shown for rows j and i, $j \neq i$, $j, i \neq 1$, by repeating the argument above, interchanging columns so that the $+1$'s in row j come first and then the -1's. This completes the proof of Theorem 10.9. Q.E.D.

10.5.3 The Richest (n, d)-Codes

We consider next the question: How "rich" are the codes obtained from Hadamard matrices? That is, compared to other possible codes with the same n (length of codewords) and the same d (minimum distance between codewords), does such a code have many or few codewords?

Let $A(n,d)$ be the maximum number of codewords in an (n,d)-code. Equation (10.5) of Theorem 10.3 gives an upper bound for $A(n,d)$. Our next result gives a better bound in most cases.

Theorem 10.12 (Plotkin [1960]) Suppose that an (n,d)-code has N codewords. If $d > n/2$, then

$$N \leq \frac{2d}{2d-n}.$$

Proof.[16] Form the $N \times n$ matrix $\mathbf{M} = (m_{ij})$ whose rows are codewords. Consider

$$S = \sum d(u,v), \tag{10.20}$$

where the sum is taken over all (unordered) pairs of words u, v from the set of codewords. We know that $d(u,v) \geq d$ for all u, v, so

$$S \geq \binom{N}{2} d = \frac{N(N-1)}{2} d. \tag{10.21}$$

Let $t_0^{(i)}$ be the number of times 0 appears in the ith column of $\mathbf{M}$ and $t_1^{(i)}$ be the number of times 1 appears in the ith column. Note that

$$S = \sum_{\{i,k\}} \sum_j |m_{ij} - m_{kj}| = \sum_j \sum_{\{i,k\}} |m_{ij} - m_{kj}|. \tag{10.22}$$

Now

$$\sum_{\{i,k\}} |m_{ij} - m_{kj}|$$

is the number of times that we have a pair of rows i and k with one having a 1 in the jth column and the other a 0 in the jth column, and this is given by $t_0^{(j)} t_1^{(j)}$. Thus, by (10.22),

$$S = \sum_j t_0^{(j)} t_1^{(j)}. \tag{10.23}$$

Now

$$t_1^{(j)} = N - t_0^{(j)},$$

so

$$t_0^{(j)} t_1^{(j)} = t_0^{(j)} [N - t_0^{(j)}].$$

We seek an upper bound on $t_0^{(j)} t_1^{(j)}$. Let $f(x)$ be the function defined by

$$f(x) = x(N-x),$$

[16] The proof may be omitted.

$0 \le x \le N$. Note that $f(x)$ is maximized when $x = N/2$ and the maximum value of $f(x)$ is $(N/2)(N/2) = N^2/4$. Thus,

$$t_0^{(j)} t_1^{(j)} \le \frac{N^2}{4},$$

so, by (10.23),

$$S \le n\frac{N^2}{4}. \tag{10.24}$$

Then, by (10.21) and (10.24),

$$\begin{aligned} \frac{N(N-1)d}{2} &\le \frac{nN^2}{4}, \\ (N-1)d &\le \frac{nN}{2}, \\ N\left(d - \frac{n}{2}\right) &\le d. \end{aligned}$$

Since $d > n/2$, we have

$$N \le \frac{d}{d - n/2} = \frac{2d}{2d - n}.$$

Q.E.D.

Corollary 10.12.1 $A(n, d) \le \dfrac{2d}{2d - n}$, if $d > \dfrac{n}{2}$.

A normalized Hadamard matrix of order $4m$ gives rise to an (n, d)-code with $n = 4m - 1, d = 2m$, and $N = 4m - 1$ codewords. Thus,

$$A(4m - 1, 2m) \ge 4m - 1. \tag{10.25}$$

By Corollary 10.12.1,

$$A(4m - 1, 2m) \le \frac{2(2m)}{2(2m) - (4m - 1)} = \frac{4m}{1} = 4m. \tag{10.26}$$

Thus, the code obtained from a normalized Hadamard matrix is close to the best possible. One gets the best possible code in terms of number of codewords by adding one codeword:

$$1 \quad 1 \quad \cdots \quad 1.$$

Adding this word is equivalent to modifying our earlier construction and not deleting the first row of the normalized Hadamard matrix of order $4m$. The distance of this word to any codeword obtained from the Hadamard matrix is $2m$, because any such codeword has $2m - 1$ 1's and $2m$ 0's. It follows that we have a code with $4m$ words, each of length $4m - 1$, and having minimum distance between two codewords equal to $2m$. This code will be called the $(4m - 1)$-*Hadamard code.* It follows that

$$A(4m - 1, 2m) \ge 4m. \tag{10.27}$$

Equations (10.26) and (10.27) give us the following theorem.

Theorem 10.13 For all m for which there is a (normalized)[17] Hadamard matrix of order $4m$,

$$A(4m-1, 2m) = 4m.$$

The bound is attained by using a $(4m-1)$-Hadamard code.

Let $B(n, d)$ be the maximum number of words of length n each of which has distance exactly d from the other. Clearly, $B(n, d) \leq A(n, d)$.

Corollary 10.13.1 For all m for which there is a (normalized) Hadamard matrix of order $4m$,

$$B(4m-1, 2m) = 4m.$$

Proof. The code we have constructed has all distances equal to $2m$. Q.E.D.

It is interesting to note that sometimes we can get much richer codes by increasing the length of codewords by 1. We shall prove the following theorem.

Theorem 10.14 For all m for which there is a (normalized) Hadamard matrix of order $4m$,

$$A(4m, 2m) = 8m.$$

This theorem shows that in cases where there is a (normalized) Hadamard matrix of order $4m$, adding one digit leads to a doubling in the number of possible codewords.

We first show that $A(4m, 2m) \geq 8m$ by constructing a $(4m, 2m)$-code with $8m$ codewords.

Starting with a $4m \times 4m$ normalized Hadamard matrix $\mathbf{H}$, we have constructed a $(4m-1, 2m-1, m-1)$-design. Let $\mathbf{A}$ be the incidence matrix of this design. Use the rows of $\mathbf{A}$ and the rows of $\mathbf{B}$, the matrix obtained from $\mathbf{A}$ by interchanging 0's and 1's. This gives us $8m-2$ words in all. The distance between two words in $\mathbf{A}$ is $2m$, the distance between two words in $\mathbf{B}$ is $2m$, and the distance between a word in $\mathbf{A}$ and a word in $\mathbf{B}$ is $2m-1$ if one is the ith word in $\mathbf{A}$ and the second is the jth word in $\mathbf{B}$ with $i \neq j$, and it is $4m-1$ if one is the ith word in $\mathbf{A}$ and the second the ith word in $\mathbf{B}$. Now add the digit 1 before words of $\mathbf{A}$ and the digit 0 before words of $\mathbf{B}$, obtaining two sets of words, $\mathbf{A}'$ and $\mathbf{B}'$. The distance between two words of $\mathbf{A}'$ or of $\mathbf{B}'$ is now still $2m$, while the distance between the ith word of $\mathbf{A}'$ and the jth word of $\mathbf{B}'$ is $2m$ if $i \neq j$ and $4m$ if $i = j$. There are still only $8m-2$ words. Add the $4m$-digit words

$$0 \quad 0 \quad \cdots \quad 0$$

and

$$1 \quad 1 \quad \cdots \quad 1.$$

We now have $8m$ words. The distance between these two new words is $4m$. Any word in $\mathbf{A}'$ or $\mathbf{B}'$ has $2m$ 1's and $2m$ 0's [since words of $\mathbf{A}$ had $k = (2m-1)$ 1's]. Thus, the new words have distance $2m$ from any word of $\mathbf{A}'$ or of $\mathbf{B}'$. We now

[17] Recall that if there is any Hadamard matrix of order $4m$, there is a normalized one.

Table 10.1: The $8m = 16$ Codewords of the $4m$-Hadamard Code Derived from the Normalized Hadamard Matrix $\mathbf{H}$ of (10.15)

1	1	1	1	1	1	1	1		0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0		0	1	0	1	0	1	0	1
1	1	0	0	1	1	0	0		0	0	1	1	0	0	1	1
1	0	0	1	1	0	0	1		0	1	1	0	0	1	1	0
1	1	1	1	0	0	0	0		0	0	0	0	1	1	1	1
1	0	1	0	0	1	0	1		0	1	0	1	1	0	1	0
1	1	0	0	0	0	1	1		0	0	1	1	1	1	0	0
1	0	0	1	0	1	1	0		0	1	1	0	1	0	0	1
From $\mathbf{H}$									From $-\mathbf{H}$							

have $8m$ codewords, each of which has length $4m$ and with the minimum distance between two such codewords equal to $2m$. This set of codewords defines the *$4m$-Hadamard code*. It is equivalent to the set of codewords obtained by taking the normalized Hadamard matrix $\mathbf{H}$ of order $4m$, not deleting the first row or column, considering the matrix

$$\begin{bmatrix} \mathbf{H} \\ -\mathbf{H} \end{bmatrix},$$

changing -1 to 0, and using the rows in the resulting matrix as codewords. The $4m$-Hadamard code with $m = 2$ derived from the normalized Hadamard matrix of (10.15) is given in Table 10.1. This completes the proof of the fact that if there is a (normalized) Hadamard matrix of order $4m$, then

$$A(4m, 2m) \geq 8m. \tag{10.28}$$

We now prove that for all $m > 0$,

$$A(4m, 2m) \leq 8m, \tag{10.29}$$

which will prove Theorem 10.14.

Theorem 10.15 If $0 < d < n$, then $A(n, d) < 2A(n-1, d)$.

Proof. Consider an (n, d)-code C of $A(n, d)$ words. Let C' be obtained from C by choosing all codewords beginning with 1 and deleting the first digit of these codewords. Then C' defines an $(n-1, d)$-code, so

$$|C'| \leq A(n-1, d).$$

Similarly, if C'' is obtained by choosing all codewords of C beginning with 0 and deleting the first digit, then

$$|C''| \leq A(n-1, d).$$

Thus,

$$A(n, d) = |C| = |C'| + |C''| \leq 2A(n-1, d). \qquad \text{Q.E.D.}$$

Now (10.29) follows immediately from Theorem 10.15 and (10.26) for we have

$$A(4m, 2m) \leq 2A(4m-1, 2m) \leq 2(4m) = 8m.$$

(Note that Theorem 10.13 could not be used directly to give this result. Why?)

The $4m$-Hadamard codes we have constructed are also called *Reed-Muller codes* (of the first kind), after Reed [1954] and Muller [1954]. The results in Theorems 10.13 and 10.14 are due to Levenshtein [1964], who obtains more general results as well.

In closing this section we note that the $(4m-1)$-Hadamard codes and the $4m$-Hadamard codes are in fact linear codes if 0's and 1's are interchanged and $m = 2^p$ for some p, as in our construction. However, the 24-Hadamard code is an example of one that is not linear. We shall not present a proof of these facts here, but instead refer the reader to the coding theory literature: for example, to MacWilliams and Sloane [1983]. (See also Exercises 25–28.)

10.5.4 Some Applications

Example 10.8 The Mariner 9 Space Probe Error-correcting codes are widely used to send messages back to Earth from deep-space missions such as the 1997 NASA Pathfinder mission to Mars. The Reed-Muller codes were used as early as the 1972 Mariner 9 space probe, which returned photographs of Mars. The specific code used was the $4m$-Hadamard code for $m = 8$, that is, the code based on a 32×32 normalized Hadamard matrix. This code has 64 codewords of 32 bits each. A photograph of Mars was broken into very small dots, and each dot was assigned 1 of 64 levels of grayness, which was encoded in one of the 32-bit codewords. (See Posner [1968], van Lint [1982], or MacWilliams and Sloane [1983] for more details.) ■

Example 10.9 The Role of Error-Correcting Codes in the Development of Compact Discs The compact disc (CD) encodes a signal digitally with high "information density." It is designed to have powerful error-correction properties, which explains the tremendous improvement in sound that CD recordings introduced. Errors can result from damage in use, dust, manufacturing defects, and so on. Such errors lead to "bursts" or consecutive sequences of errors. A coding technique known as CIRC (Cross-Interleaved Reed-Solomon Code) leads to the ability to correct up to 4000 consecutive errors through the use of two interleaved codes. For more details about the use of error-correcting codes in CDs, see Bossert, *et al.* [1997], Immink [1991], Peek [1985], Pinch [2001], or Vries and Odaka [1982]. ■

Example 10.10 "Reading" DNA to Produce Protein Golomb [1962] speculated that error-correcting codes might be at work in genetic coding, specifically in the process by which DNA strands are "read" in order to produce proteins. Error correction would be used whenever a sequence of three bases does not correspond to a code for an amino acid (see Example 2.2). Golomb speculated that a $4m$-Hadamard code is used. The smallest such code that would encode for the 20 different amino acids has 24 codewords and results from a 12×12 normalized Hadamard matrix. (That such a matrix exists does not follow from our theorems.) Codewords have length 12. A DNA chain would be encoded into a string of 0's and 1's in two steps. First, one of the letters A, C, G, and T would be encoded as 00, one as 01, one as 10, and one as 11. This represents a DNA chain of length m as a bit string of length $2m$, the message word. Every message word would be broken into blocks of length $k = 2$ and encoded by a $2 \to 12$ code. Every six bits of the message word (every three letters of the chain) or every 36 bits of the code would correspond to an amino acid. ■

EXERCISES FOR SECTION 10.5

1. A Hadamard design includes the following blocks. Find the missing blocks.

$$\{1, 2, 4\}, \{2, 3, 5\}, \{1, 3, 6\}, \{3, 4, 7\}.$$

2. Suppose that $\mathbf{A}$ is the incidence matrix of a (b, v, r, k, λ)-design and that a code is made up of the rows of this matrix.
   (a) What is the distance between two words in this code?
   (b) How many errors will the code detect?
   (c) How many errors will the code correct?
   (d) Suppose that $\mathbf{B}$ is the incidence matrix of the complementary design (Exercise 20, Section 9.4). What is the distance between the ith row of $\mathbf{A}$ and the jth row of $\mathbf{B}$ if $i \neq j$?
   (e) If rows of $\mathbf{B}$ form a code, how many errors will the code detect?
   (f) How many will it correct?
3. Given a Steiner triple system of nine varieties, build a code by using the rows of the incidence matrix. Find the minimum distance between two codewords.
4. (a) Could there be a 3×3 Hadamard matrix?
   (b) A 6×6?
5. If $\mathbf{H}$ is a Hadamard matrix and some row is multiplied by -1, is the resulting matrix still Hadamard? Why?
6. Show that if there is a Hadamard matrix of order $4m$, there is a normalized Hadamard matrix of order $4m$.
7. Suppose that

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

   Use $\mathbf{H}$ to find an 8×8 Hadamard matrix and a $(7, 3, 1)$-design.
8. Derive the $(15, 7, 3)$-design of Table 9.27 from a 16×16 normalized Hadamard matrix.

9. (a) Find a $(4m-1)$-Hadamard code for $m=4$.
   (b) How many errors can this code detect?
   (c) How many errors can it correct?
10. (a) Find a $4m$-Hadamard code for $m=4$.
    (b) How many errors can this code detect?
    (c) How many errors can it correct?
11. Repeat Exercise 9 for $m=8$.
12. Repeat Exercise 10 for $m=8$.
13. The following is an $(11,5,2)$-design. Use it to find a code of 12 words that can detect up to five errors and a code of 24 words that can detect up to five errors.

$$\begin{array}{llll} \{1,2,3,4,9\}, & \{1,2,5,6,10\}, & \{2,3,6,7,11\}, & \{1,4,6,7,8\}, \\ \{2,5,7,8,9\}, & \{3,6,8,9,10\} & \{1,7,9,10,11\}, & \{2,4,8,10,11\}, \\ \{4,5,6,9,11\}, & \{3,4,5,7,10\}, & \{1,3,5,8,11\}. & \end{array}$$

14. Complete the proof that the matrix $\mathbf{H}$ constructed in the proof of Theorem 10.10 defines a Hadamard matrix.
15. Illustrate the construction in the proof of Theorem 10.10 by starting with the incidence matrix of a $(7,3,1)$-design and constructing the corresponding Hadamard matrix.
16. Suppose that $\mathbf{H}$ is the incidence matrix of a $(4m-1, 2m-1, m-1)$-design and $\mathbf{K}$ is the incidence matrix of the complementary design (Exercise 20, Section 9.4).
    (a) What is the distance between the ith row of $\mathbf{H}$ and the jth row of $\mathbf{H}$?
    (b) What is the distance between the ith row of $\mathbf{H}$ and the jth row of $\mathbf{K}$?
17. Suppose that S is a set of binary codewords of length n with Hamming distance $d(a,b)$ equal to d for each pair of words a and b in S. Let T be defined from S by taking complements of words in S (interchanging 0 and 1).
    (a) What is the distance between two words of T?
    (b) What is the distance between a word of S and a word of T?
    (c) If n is 12 and d is 5, how many errors will the code T detect? How many will it correct?
    (d) Suppose that a code is defined from the words in S and the words in T. If n is 12 and d is 5, how many errors will this code detect? How many will it correct?
18. Suppose that the $m \times n$ matrix $\mathbf{M}$ is the incidence matrix of a block design, and that the ith row and the jth row have 1's in common in u columns.
    (a) What is the inner product of these two rows?
    (b) What is the inner product in the complementary design (Exercise 20, Section 9.4)?
19. Find a 2×4 matrix of 1's and -1's such that the inner product of rows i and j is 0 if $i \neq j$ and 4 if $i = j$.

20. For what values of k does there exist a $2 \times k$ matrix of 1's and -1's such that the inner product of rows i and j is 0 if $i \neq j$ and k if $i = j$?
21. Show that $A(n, n-1) = 2$ for n sufficiently large.
22. (a) Show that if $n > m$, then $A(n,d) \geq A(m,d)$.
    (b) If $n > m$, is it necessarily the case that $A(n,d) > A(m,d)$? Why?
23. (a) Find an upper bound for $A(4m-2, 2m)$.
    (b) Find $A(4m-2, 2m)$ exactly for arbitrarily large values of m.
24. Using the results of Exercise 17, show that
$$A(2d, d) \geq 2B(2d, d).$$
25. Show that the $(4m-1)$-Hadamard code as we have defined it is not linear. (*Hint:* Consider sums of codewords.)
26. If $m = 2^p$ and if 0's and 1's are interchanged in the $(4m-1)$-Hadamard code as we have defined it, the code becomes linear. Show this for the case $m = 1$ by finding a generator matrix.
27. Continuing with Exercise 26, show linearity for the case $m = 2$ as follows:
    (a) Using the generator matrix obtained by taking
$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix},$$
    generate the corresponding code.
    (b) Show that this code is equivalent (Exercise 25, Section 10.4) to the $(4m-1)$-Hadamard code with $m = 2$ and 0's and 1's interchanged.
28. If $m \neq 2^p$, show that even if 0's and 1's are interchanged in the $(4m-1)$-Hadamard code, the code is not linear. (*Hint:* How many codewords does a linear code have?)
29. Recall that the *Hamming weight* of a bit string $\mathbf{x}$, denoted $wt(\mathbf{x})$, is the number of nonzero digits of $\mathbf{x}$. Let $A(n,d,w)$ be the maximum number of codewords in an (n,d)-code in which each word has the same Hamming weight w. Show that:
    (a) $A(n, 2d-1, w) = A(n, 2d, w)$ (b) $A(n, 2d, w) = A(n, 2d, n-w)$
    (c) $A(n, 2d, w) = 1$ if $w < d$ (d) $A(n, 2d, d) = \lfloor n/d \rfloor$

REFERENCES FOR CHAPTER 10

Avizenius, A., "Fault-Tolerant Systems," *IEEE Trans. Comput., C-25* (1976), 1304–1312.

Berlekamp, E. R., *Algebraic Coding Theory*, McGraw-Hill, New York, 1968.

Blake, I. F., and Mullin, R. C., *The Mathematical Theory of Coding*, Academic Press, New York, 1975. (Abridged as *An Introduction to Algebraic and Combinatorial Coding Theory*, Academic Press, New York, 1976.)

Bock, H. H. (ed.), *Classification and Related Methods of Data Analysis*, North-Holland, Amsterdam, 1988.

Bose, R. C., and Shrikhande, S. S., "A Note on a Result in the Theory of Code Construction," *Inf. Control, 2* (1959), 183–194.

Bossert, M., Braĭtbakh, M., Zyablov, V. V., and Sidorenko, V. R., "Codes That Correct Multiple Burst Errors or Erasures (Russian)" *Problemy Peredachi Informatsii, 33* (1997), 15–25; translation in *Problems Inform. Transmission, 33* (1997), 297–306.

Cameron, P. J., and van Lint, J. H., *Designs, Graphs, Codes, and Their Links*, London Mathematical Society Student Texts 22, Cambridge University Press, Cambridge, 1991.

Carter, W. C., and Bouricius, W. G., "A Survey of Fault-Tolerant Architecture and Its Evaluation," *Computer, 4* (1971), 9–16.

Craigen, R., and Wallis, W. D., "Hadamard Matrices: 1893 – 1993," *Congr. Numer., 97* (1993), 99–129.

Day, W. H. E., "The Sequence Analysis and Comparison Bibliography," at http://www.classification-society.org/sequence.html, Oct. 12, 2002.

Day, W. H. E., and McMorris, F. R., "Critical Comparison of Consensus Methods for Molecular Sequences," *Nucleic Acids Research, 20* (1992), 1093–1099.

Day, W. H. E., and McMorris, F. R., "Discovering Consensus Molecular Sequences," in O. Opitz, B. Lausen, and R. Klar (eds.), *Information and Classification: Proceedings of the 16th Annual Conference of the Gesellsshaft fuer Klassifikation e.V.,* Springer-Verlag, Heidelberg, 1993, 393–402.

Day, W. H. E., and McMorris, F. R., *Axiomatic Consensus Theory in Group Choice and Biomathematics*, SIAM Publications, Philadelphia, 2003.

Dornhoff, L. L., and Hohn, F. E., *Applied Modern Algebra*, Macmillan, New York, 1978.

Fisher, J. L., *Application-Oriented Algebra: An Introduction to Discrete Mathematics*, Harper & Row, New York, 1977.

Galas, D. J., Eggert, M., and Waterman, M. S., "Rigorous Pattern-Recognition Methods for DNA Sequences. Analysis of Promoter Sequences from *Escherichia Coli*," *J. Molecular Biol., 186* (1985), 117–128.

Golay, M. J. E., "Notes on Digital Coding," *Proc. IEEE, 37* (1949), 657.

Goldie, C. M., and Pinch, R. G. E., *Communication Theory*, London Mathematical Society Student Texts 20, Cambridge University Press, Cambridge, 1991.

Golomb, S. W., "Efficient Coding for the Deoxyribonucleic Channel," in *Mathematical Problems in the Biological Sciences*, Proceedings of Symposia in Applied Mathematics, Vol. 14, American Mathematical Society, Providence, RI, 1962, 87–100.

Hadamard, J., "Résolution d'une Question Relative aux Déterminants," *Bull. Sci. Math., 17* (1893), 240–248.

Hall, M., *Combinatorial Theory*, Ginn (Blaisdell), Boston, 1967. (Second printing, Wiley, New York, 1980.)

Hamming, R. W., "Error Detecting and Error Correcting Codes," *Bell Syst. Tech. J., 29* (1950), 147–160.

Hill, R., *A First Course in Coding Theory*, Oxford University Press, New York, 1986.

Immink, K. A. S., *Coding Techniques for Digital Recorders*, Prentice Hall International, Hertfordshire, UK, 1991.

Janowitz, M., LaPointe, F.-J., McMorris, F. R., Mirkin, B., and Roberts, F. S., *Bioconsensus*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 61, American Mathematical Society, Providence, RI, 2003.

JOHNSON, P. E., *Social Choice: Theory and Research*, Sage Publications, Thousand Oaks, CA, 1998.

KANNAN, S., "A Survey of Tree Consensus Criteria and Methods," in S. Tavare (ed.), *Proceedings of Phylogeny Workshop*, DIMACS Tech. Rep. 95–48, DIMACS Center, Rutgers University, Piscataway, NJ, 1995.

LEVENSHTEIN, V. I., "The Application of Hadamard Matrices to a Problem in Coding," *Probl. Kibern., 5* (1961), 123–136. [English transl.: *Probl. Cybern., 5* (1964), 166–184.]

MACWILLIAMS, F. J., and SLOANE, N. J. A., *The Theory of Error-Correcting Codes*, Vols. 1 and 2, North-Holland, Amsterdam, 1983.

MIRKIN, B., and ROBERTS, F. S., "Consensus Functions and Patterns in Molecular Sequences," *Bull. Math. Biol., 55* (1993), 695–713.

MULLER, D. E., "Application of Boolean Algebra to Switching Circuit Design and to Error Detection," *IEEE Trans. Comput., 3* (1954), 6–12.

PEEK, J. B. H., "Communications Aspects of the Compact Disc Digital Audio System," *IEEE Commun. Magazine, 23* (1985), 7–15.

PETERSON, W. W., *Error Correcting Codes*, MIT Press, Cambridge, MA, 1961.

PETERSON, W. W., and WELDON, E. J., *Error Correcting Codes*, MIT Press, Cambridge, MA, 1972. (Second edition of Peterson [1961].)

PINCH, R. G. E., "Coding Theory: The First 50 Years," at http://pass.maths.org/issue3/codes/, Feb. 21, 2001.

PLESS, V., *Introduction to the Theory of Error-Correcting Codes*, Wiley, New York, 1998.

PLOTKIN, M., "Binary Codes with Specified Minimum Distances," *IEEE Trans. Inf. Theory, 6* (1960), 445–450.

POLI, A., and HUGUET, L., *Error Correcting Codes: Theory and Applications*, Prentice Hall International, Hemel Hempstead/Masson, Paris, 1992.

POSNER, E. C., "Combinatorial Structures in Planetary Reconnaissance," in H. B. Mann (ed.), *Error Correcting Codes*, Wiley, New York, 1968.

REED, I. S., "A Class of Multiple-Error-Correcting Codes and the Decoding Scheme," *IEEE Trans. Inf. Theory, 4* (1954), 38–49.

ROBERTS, F. S., *Discrete Mathematical Models, with Applications to Social, Biological, and Environmental Problems*, Prentice Hall, Englewood Cliffs, NJ, 1976.

SELLERS, F. F., HSIAO, M. Y., and BEARNSON, L. W., *Error Detecting Logic for Digital Computers*, McGraw-Hill, New York, 1968.

SLEPIAN, D., "A Class of Binary Signaling Alphabets," *Bell Syst. Tech. J., 35* (1956), 203–234. (a)

SLEPIAN, D., "A Note on Two Binary Signaling Alphabets," *IEEE Trans. Inf. Theory, 2* (1956), 84–86. (b)

SLEPIAN, D., "Some Further Theory of Group Codes," *Bell Syst. Tech. J., 39* (1960), 1219–1252.

SYLVESTER, J. J., "Thoughts on Inverse Orthogonal Matrices, Simultaneous Sign Successions, and Tesselated Pavements in Two or More Colors, with Applications to Newton's Rule, Ornamental Tile-Work, and the Theory of Numbers," *Philos. Mag., 34* (1867), 461–475.

TIETÄVÄINEN, A., "On the Nonexistence of Perfect Codes over Finite Fields," *SIAM J. Appl. Math., 24* (1973), 88–96.

VAN LINT, J. H., "Nonexistence Theorems for Perfect Error-Correcting Codes," in *Computers in Algebra and Number Theory*, Vol. 4 (SIAM-AMS Proceedings), 1971.

van Lint, J. H., "Coding, Decoding and Combinatorics," in R. J. Wilson (ed.), *Applications of Combinatorics*, Shiva, Nantwich, UK, 1982, 67–74.

van Lint, J. H., *Introduction to Coding Theory*, 3rd ed., Springer-Verlag, New York, 1999.

Vries, L. B., and Odaka, K., "CIRC – The Error-Correcting Code for the Compact Disc Digital Audio System," *Collected Papers from the AES Premiere Conference, Rye, New York*, Audio Engineering Society, New York, 1982, 178–188.

Wakerly, J., *Error Detecting Codes, Self-Checking Circuits, and Applications*, Elsevier North-Holland, New York, 1978.

Waterman, M. S., "Consensus Patterns in Sequences," in M. S. Waterman (ed.), *Mathematical Methods for DNA Sequences*, CRC Press, Boca Raton, FL, 1989, 93–115.

Waterman, M. S., *Introduction to Computational Biology; Maps, Sequences and Genomes*, CRC Press, Boca Raton, FL, 1995.

Waterman, M. S., Arratia, R., and Galas, D. J., "Pattern Recognition in Several Sequences: Consensus and Alignment," *Bull. Math. Biol., 46* (1984), 515–527.

Welsh, D., *Codes and Cryptography*, Oxford University Press, New York, 1988.

Chapter 11

Existence Problems in Graph Theory

Topics in graph theory underlie much of combinatorics. This chapter begins a sequence of three chapters dealing mostly with graphs and networks. The modern approach to graph theory is heavily influenced by computer science, and it emphasizes algorithms for solving problems. Among other things, we give an introduction to graph algorithms in these chapters. We also talk about applications of graph theory, many of which have already been mentioned in Section 3.1. This chapter emphasizes existence questions in graph theory and Chapter 13 emphasizes optimization questions. Chapter 12 is a transitional chapter, which begins with existence questions and ends with optimization questions. Of course, it is hard to make such a rigid partition. The same techniques and concepts of graph theory that are used on existence problems are usually useful for optimization problems, and vice versa.

In this chapter we examine four basic existence questions and discuss their applications. The existence questions are:

1. Is a given graph G connected; that is, does there exist, for every pair of vertices x and y of G, a chain between x and y?

2. Does a given graph G have a strongly connected orientation?

3. Does a given graph G have an eulerian chain, a chain that uses each edge exactly once?

4. Does a given graph G have a hamiltonian chain, a chain that uses each vertex exactly once?

We present theorems that help us to answer these questions. With practical use in mind, we shall be concerned with describing good procedures or algorithms for answering them. The results will have applications to such subjects as traffic flow, RNA codes, street sweeping, and telecommunications.

11.1 DEPTH-FIRST SEARCH: A TEST FOR CONNECTEDNESS

Suppose that $G = (V, E)$ is a graph. The first question we shall ask is this: Is G connected, that is, does there exist, for every pair of vertices x and y of G, a chain between x and y?

Given a small graph, it is easy to see from the corresponding diagram whether or not it is connected. However, for large graphs, drawing such diagrams is infeasible. Moreover, diagrams are not readily amenable to use as input in a computer. Instead, one has to input a graph into a computer by, for example, listing its edges. (See Section 3.7 for a more complete discussion of ways to input a graph into a computer.) In any case, it is now not so easy to check if a graph is connected. Thus, we would like an algorithm for testing connectedness. In Section 11.1.1 we present such an algorithm. The algorithm will play a crucial role in solving the traffic flow problem we study in Section 11.2 and can also be utilized in finding eulerian chains, the problem we discuss in Section 11.3. This algorithm is also important for solving optimization problems in graph theory, not just existence problems. However, for truly massive graphs, i.e., when the edge set does not fit into a computer's available RAM (Random Access Memory), totally new algorithms are needed. We discuss this point further in Section 11.1.4.

11.1.1 Depth-First Search

Suppose that $G = (V, E)$ is any graph. The method we describe for testing if G is connected is based on the *depth-first search procedure*, a highly efficient procedure which is the basis for a number of important computer algorithms. (See Aho, Hopcroft, and Ullman [1974, Ch. 5], Baase [1992], Cormen, Leiserson, and Rivest [1999], Golumbic [1980], Hopcroft and Tarjan [1973], Reingold, Nievergelt, and Deo [1977], or Tarjan [1972] for a discussion. See Exercise 9 for a related search procedure known as breadth-first search.)

In the depth-first search procedure, we start with a graph G with n vertices and aim to label the vertices with the integers $1, 2, \ldots, n$. We choose an arbitrary vertex and label it 1. Having just labeled a given vertex x with the integer k, we search among all neighbors of x and find an unlabeled neighbor, say y. We give vertex y the next label, $k+1$. We also keep track of the edges used in the labeling procedure by *marking* the edge $\{x, y\}$ traversed from x to y. We then continue the search among neighbors of y. In this way, we progress along chains of G leading away from x. The complication arises if we have just labeled a vertex z which has no unlabeled neighbors. We then go back to the neighbor u of z from which z was labeled—u is called the *parent* of z—and continue the search from u. We can keep track of the parent of a vertex since we have marked the edges traversed from a vertex to the next one labeled. The procedure continues until all the labels $1, 2, \ldots, n$ have been used (equivalently all vertices have been labeled) or it is impossible to continue because we have returned to a labeled vertex with no parent: namely, the vertex labeled 1.

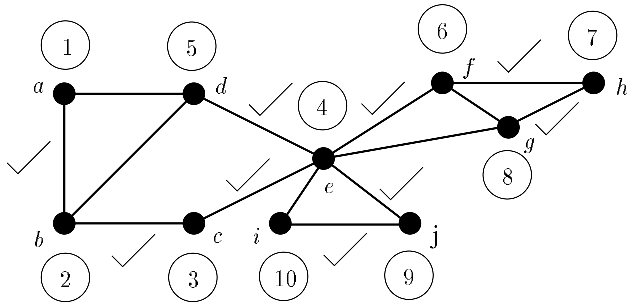


Figure 11.1: A labeling of vertices obtained using the depth-first search procedure. A check mark indicates marked edges.

We illustrate the labeling procedure in the graph of Figure 11.1. Vertex a is chosen first and labeled 1. The labels are shown in the figure by circled numbers next to the vertices. Vertex b is an unlabeled neighbor of vertex a which is chosen next and labeled 2. The edge $\{a, b\}$ is marked (we put a check mark in the figure). Next, an unlabeled neighbor of vertex b is found and labeled 3. This is vertex c. The edge $\{b, c\}$ is marked. An unlabeled neighbor of c, vertex e, is labeled 4, and the edge $\{c, e\}$ is marked. An unlabeled neighbor of e, vertex d, is labeled 5, and the edge $\{e, d\}$ is marked. Now after vertex d is labeled 5, all neighbors of d are labeled. Thus, we go back to e, the parent of d, and seek an unlabeled neighbor. We find one, vertex f, and label it 6. We also mark edge $\{e, f\}$. We now label vertex h with integer 7 and mark edge $\{f, h\}$ and label vertex g with integer 8 and mark edge $\{h, g\}$. Now all neighbors of g are labeled, so we go back to the parent of g, namely h. All neighbors of h are also labeled, so we go back to the parent of h, namely f. Finally, since all neighbors of f are labeled, we go back to the parent of f, namely e. From e, we next label vertex j with integer 9 and mark edge $\{e, j\}$. Then from j, we label vertex i with integer 10 and mark edge $\{j, i\}$. Now all vertices are labeled, so we stop.

It is easy to show that the labeling procedure can be completed if and only if the graph is connected. Thus, the depth-first search procedure provides an algorithm for testing if a graph is connected. Figure 11.2 shows the procedure on a disconnected graph. After having labeled vertex d with label 4, we find that the parent of d, namely c, has no unlabeled neighbors. Moreover, the same is true of the parent of c, namely a. Finally, a has no parent, so we cannot continue the labeling.

It should also be noted that if the labeling procedure is completed and T is the collection of marked edges, then there are $n - 1$ edges in T, for one edge is added to T each time we assign a new label. Moreover, suppose that H is the spanning subgraph of G whose edges are the edges in T. Then H has no circuits, because every edge added to T goes from a vertex labeled i to a vertex labeled j with $i < j$. Hence, H is a subgraph of G with $n = e + 1$ and no circuits. By Theorem 3.20, H is a tree. It is called the *depth-first search spanning tree.* (The reader should observe that the checked edges of Figure 11.1 form a tree.)

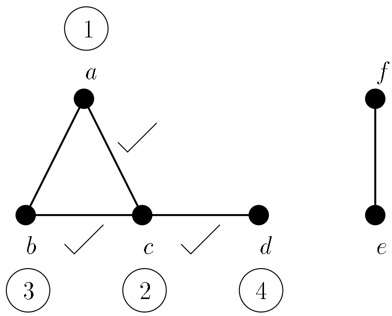


Figure 11.2: Depth-first search labeling and marking on a disconnected graph.

11.1.2 The Computational Complexity of Depth-First Search

The computational complexity of this depth-first search algorithm can be computed as follows. Suppose that the graph has n vertices and e edges. Each step of the procedure involves assigning a label or traversing an edge. The traversal is either forward from a labeled vertex to a neighbor or backward from a vertex to its parent. By marking not only edges that are *used* in a forward direction, but also edges that are *investigated* in a forward direction but lead to already labeled vertices, we can be sure that no edge is used or investigated in a forward direction more than once. Also, it is not hard to see that no edge can be used backward more than once. Thus, the entire procedure investigates at most $2e$ edges. Since at most n labels are assigned, the procedure terminates in at most $n+2e$ steps. [In the notation of Section 2.18, this is an $O(n+e)$ algorithm.] Since $2e$ is at most

$$2\binom{n}{2} = 2\frac{n(n-1)}{2} = n^2 - n,$$

this is a polynomial bound on the complexity in terms of the number of vertices. [It is an $O(n^2)$ algorithm.[1]] The algorithm is very efficient.

It should be noted that a similar directed depth-first search procedure can be defined for digraphs. We simply label as before, but only go from a given vertex to vertices reachable from it by arcs, not edges. This method can be used to provide an efficient test for strong connectedness. For details, the reader is referred to Aho, Hopcroft, and Ullman [1974, Ch. 5], Cormen, Leiserson, and Rivest [1999], Hopcroft and Tarjan [1973], or Tarjan [1972]. (See also Exercise 8.)

11.1.3 A Formal Statement of the Algorithm[2]

For the reader who is familiar with recursive programming, we close this section with a more formal statement of the depth-first search algorithm. We first state a subroutine called DFSEARCH (v, u). In this subroutine, as well as in the main algorithm, k will represent the current value of the label being assigned and T will

[1] Some authors call the algorithm *linear* because its complexity is linear in $n+e$. Others call it *quadratic* because its complexity is quadratic in n.

[2] This subsection may be omitted.

be the set of marked edges. (We disregard the separate marking of edges that are investigated but lead to already labeled vertices.) The control is the vertex whose neighbors are currently being searched. Some vertices will already bear labels. The vertex v is the one that is just to be labeled and u is the parent of v.

Algorithm 11.1: DFSEARCH (v, u)

Input: A graph G with some vertices labeled and some edges in T, a vertex v that is just to be labeled, a vertex u that is the parent of v, and a current label k to be assigned.

Output: Some additional label assignments and a (possibly) new set T and a new current label k.

Step 1. Set the control equal to v, mark v labeled, assign v the label k, and replace k by $k + 1$.

Step 2. For each neighbor w of v, if w is unlabeled, add the edge $\{v, w\}$ to T, mark the edge $\{v, w\}$, call v the parent of w, and perform the algorithm DFSEARCH (w, v).

Step 3. If v has no more unlabeled neighbors and all vertices have been labeled, stop and output the labeling. If some vertex is still unlabeled, set the control equal to u and stop.

We can now summarize the entire depth-first search algorithm as follows.

Algorithm 11.2: Depth-First Search

Input: A graph G of n vertices.

Output: A labeling of the vertices of G using the integers $1, 2, \ldots, n$ and an assignment of $n - 1$ edges of G to a set T, or the message that the procedure cannot be completed.

Step 1. Set $T = \emptyset$ and $k = 1$ and let no vertex be labeled.

Step 2. Pick any vertex v, introduce a (dummy) vertex α, call α the parent of v, and perform DFSEARCH (v, α).

Step 3. If the control is ever set equal to α, output the message that the procedure cannot be completed.

Algorithm 11.2 terminates either because all vertices have been labeled or because the labeling procedure has returned to the vertex labeled 1, all neighbors of that vertex have labels, and there are some unlabeled vertices. In this case, it is easy to see that the graph is disconnected.

11.1.4 Testing for Connectedness of Truly Massive Graphs[3]

Graphs arising from modern applications involving telecommunications traffic and web data may be so massive that their edge sets do not fit into main memory

[3] This subsection may be omitted.

Table 11.1: Adjacency Structure for a Graph

Vertex x	a	b	c	d	e	f	g
Vertices adjacent to x	e, d	c, g	b, g	a, e	a, d, f	e	b, c

on modern computers. For example, massive telephone calling graphs arise when telephone numbers are vertices and a call between telephone numbers is an edge. (More generally, we view the calls as directed and we look at multidigraphs.) Such graphs can have several billion edges. When the edge set does not fit into RAM (Random Access Memory), many of the classical graph theory algorithms break down. (Even a computer with 6 gigabytes of main memory cannot hold the full telephone calling graphs that arise in practice.) This calls for the development of "external memory algorithms." Examples of such algorithms for connectedness are those in Abello, Buchsbaum, and Westbrook [2002]. Other work on massive telephone-calling graphs and related graphs of connectivity over the Internet emphasizes the evolution of such graphs over time and the changing connectivity of the graphs. For work on this theme, see, for example, Aiello, Chung, and Lu [2000]. For information on external memory algorithms, see Abello and Vitter [1999]. In particular, see the paper by Abello, Pardalos, and Resende [1999].

EXERCISES FOR SECTION 11.1

1. For each graph of Figure 11.3, perform a depth-first search beginning with the vertex labeled a. Indicate all vertex labels and all marked edges.
2. Given the adjacency structure of Table 11.1 for a graph G of vertices a, b, c, d, e, f, g, use depth-first search to determine if G is connected.
3. From each of the following adjacency matrices for a graph G, use depth-first search to determine if G is connected.

$$\text{(a)}\ \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad \text{(b)}\ \mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4. For each graph of Figure 11.3, find a depth-first search spanning tree.
5. Is every spanning tree attainable as a depth-first search spanning tree? Why?
6. Suppose that the depth-first search algorithm is modified so that if further labeling is impossible, but there is still an unlabeled vertex, some unlabeled vertex is chosen, given a new label, and the process starts again. What can you say about the subgraph determined by the marked edges (the edges in T)?

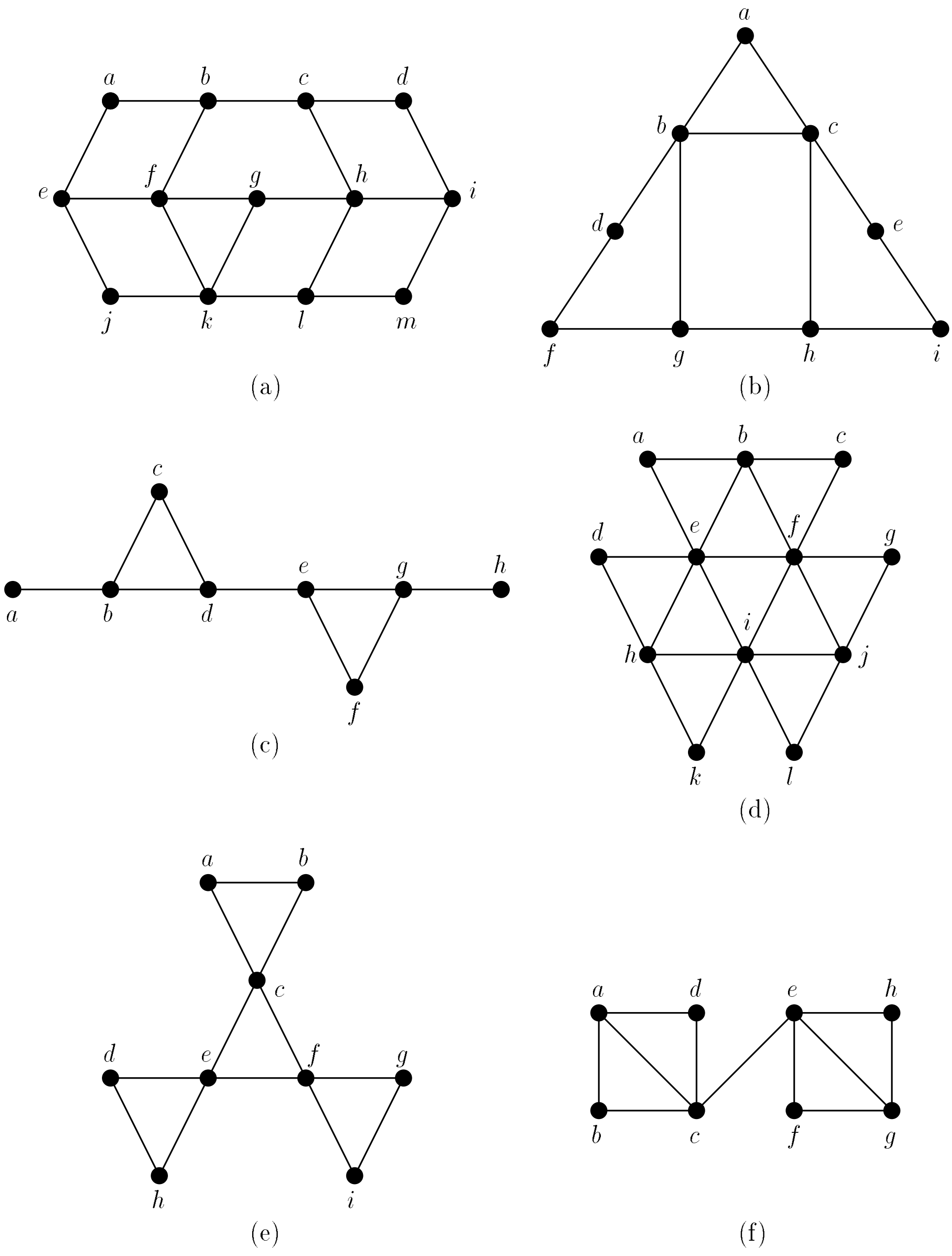


Figure 11.3: Graphs for exercises of Section 11.1.

7. How can the depth-first search algorithm as described in the text be modified to count the number of connected components of a graph G?

8. (a) Is the following test for strong connectedness of a digraph D correct? Pick any vertex of the digraph to receive label 1 and perform a directed depth-first search until no more vertices can be labeled. Then D is strongly connected if and only if all vertices have received a label.

   (b) If this is not a correct test, how can it be modified?

9. In the algorithm known as *breadth-first search*, we start with an arbitrarily chosen vertex x and place it at the head of an ordered list or *queue*. At each stage of the algorithm, we pick the vertex y at the head of the queue, delete y from the queue, and add to the tail of the queue, one at a time, all vertices adjacent to y which have never been on the queue. We continue until the queue is empty. If not all vertices have been on the queue, we start with another arbitrarily chosen vertex which has never been on the queue. Breadth-first search fans out in all directions from a starting vertex, whereas depth-first search follows one chain at a time from this starting vertex to an end. We have already employed breadth-first search in Algorithm 3.1 of Section 3.3.4. For a detailed description of breadth-first search, see, for example, Baase [1992], Even [1979], or Golumbic [1980]. Perform a breadth-first search on each graph of Figure 11.3, beginning with the vertex a. Indicate the order of vertices encountered by labeling each with an integer from 1 to n in the order they reach the head of the queue. (*Note*: In many applications, breadth-first search is inefficient or unwieldy compared to depth-first search. This is true, for example, in algorithms for testing if a given graph is planar. However, in many network optimization problems, such as most of those studied in Chapter 13, breadth-first search is the underlying procedure.)

10. Can breadth-first search (Exercise 9) be used to test a graph for connectedness? If so, how?

11. What is the computational complexity of breadth-first search?

12. Consider a telephone calling graph over the span of six weeks for the phone numbers in a given area code.

   (a) What is the size of the vertex set for this graph assuming that all possible phone numbers are available?

   (b) What is the size of the edge set if each phone number initiates an average of 12 calls per day?

11.2 THE ONE-WAY STREET PROBLEM

11.2.1 Robbins' Theorem

In this section we discuss an application of graph theory to a traffic flow problem. Imagine that a city has a number of locations, some of which are joined by two-way streets. The number of cars on the road has markedly increased, resulting in traffic jams and increased air pollution, and it has been suggested that the city should make all its streets one way. This would, presumably, cut down on traffic congestion. The

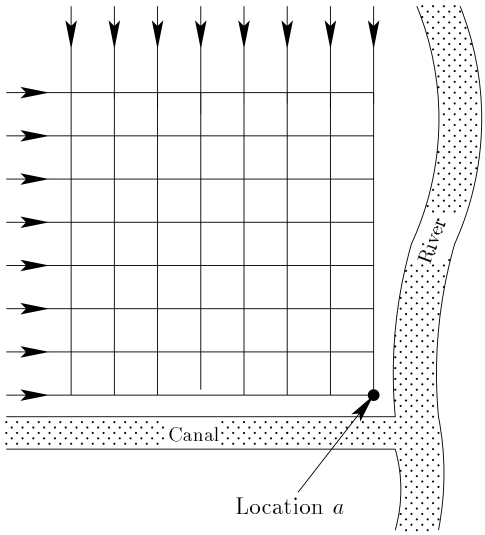
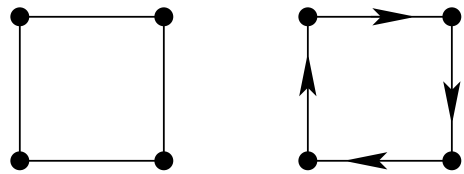


Figure 11.4: A one-way street assignment for the streets of a city which leaves travelers stuck at location a.

question is: Can this always be done? If not, when? The answer is: Of course, it can always be done. Just put a one-way sign on each street! However, it is quite possible that we will get into trouble if we make the assignment arbitrarily, for example by ending up with some places that we can get into and never leave. (See, for example, Figure 11.4, which shows an assignment of directions to the streets of a city that is satisfactory only for someone who happens to own a parking lot at location a.) We would like to make every street one-way in such a manner that for every pair of locations x and y, it is possible (legally) to reach x from y and reach y from x. Can this always be done?

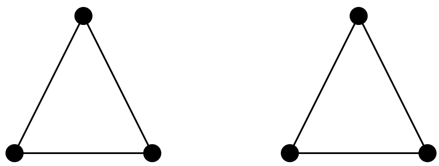
To solve this problem, we assume for simplicity that all streets are two-way at the beginning. See Boesch and Tindell [1980] or Roberts [1978] for a treatment of the problem where some streets are one-way before we start (see also Exercise 13). We represent the transportation network of a city by a graph G. Let the locations in question be the vertices of G and draw an edge between two locations x and y if and only if x and y are joined by a two-way street. A simple example of such a graph is shown in Figure 11.5(a). In terms of the graph, our problem can now be restated as follows: Is it possible to put a direction or arrow (a one-way sign) on each edge of the graph G in such a way that by following arrows in the resulting figure, which is a digraph, one can always get from any point x to any other point y? If it is not always possible, when is it possible? Formally, we define an *orientation* of a graph G as an assignment of a direction to each edge of G. We seek an orientation of G which, to use the terminology of Section 3.2, is strongly connected.

In the graph of Figure 11.5(a), we can certainly assign a direction to each edge to obtain a strongly connected digraph. We have done this in Figure 11.5(b). However,



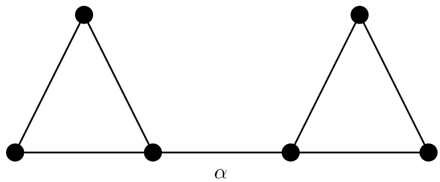
(a) (b)

Figure 11.5: (a) A two-way street graph for a city and (b) a strongly connected orientation.



I II

Figure 11.6: Graph representing a disconnected city.



I II

Figure 11.7: α is a bridge.

it is not always possible to obtain a strongly connected orientation. For example, if our graph has two components, as in the graph of Figure 11.6, there is no way of assigning directions to edges which will make it possible to go from vertices in section I of the graph to vertices in section II. To have a strongly connected orientation, our graph must certainly be connected. However, even a connected graph may not have a strongly connected orientation. Consider the graph of Figure 11.7. It is connected. We must put a direction on edge α. If α is directed from section I to section II, then from no vertex in II can we reach any vertex in I. If α is directed from II to I, then from no vertex in I can we reach any vertex in II. What is so special about the edge α? The answer is that it is the only edge joining two separate pieces of the graph. Put more formally, removal of α (but not the two vertices it joins) results in a disconnected graph. Such an edge in a connected graph is called a *bridge*. Figure 11.8 gives another example of a bridge α. It is clear that for a graph G to have a strongly connected orientation, G must not only be connected, but it must also have no bridges.

In Figure 11.8, suppose that we add another bridge β joining the two separate components which are joined by the bridge α, obtaining the graph of Figure 11.9. Does this graph have a strongly connected orientation? The answer is yes. A satisfactory assignment is shown in Figure 11.10. Doesn't this violate the observation we have just made, namely that if a graph G has a strongly connected orientation, it can have no bridges? The answer here is no. For we were too quick to call β a bridge. In the sense of graph theory, neither β nor α is a bridge in the graph of Figure 11.9. A graph-theoretical bridge has the nasty habit that if we have a bridge

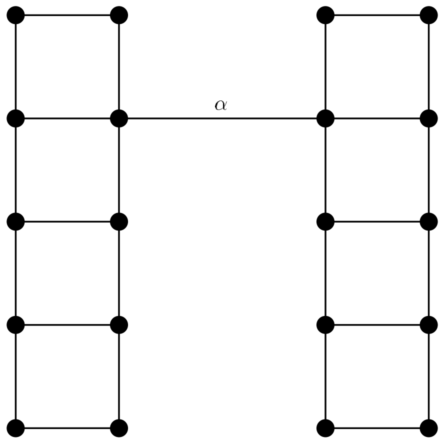


Figure 11.8: α is again a bridge.

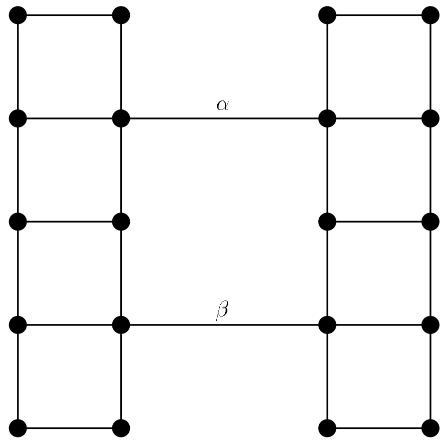


Figure 11.9: Edge α is no longer a bridge.

and build another bridge, then neither is a bridge! In applying combinatorics, if we use suggestive terminology such as the term *bridge*, we have to be careful to be consistent in our usage.

Suppose now that G has the following properties: It is a connected graph and has no bridges. Are these properties sufficient to guarantee that G has a strongly connected orientation? The answer turns out to be yes, as we summarize in the following theorem.

Theorem 11.1 (Robbins [1939].) A graph G has a strongly connected orientation if and only if G is connected and has no bridges.

We omit the proof of Theorem 11.1. For two different proofs, see Roberts [1976, 1978] and Boesch and Tindell [1980]. For a sketch of one of the proofs, see Exercise 13.

11.2.2 A Depth-First Search Algorithm

Robbins' Theorem in some sense completely solves the problem we have stated. However, it is not, by itself, a very useful result. For the theorem states that there is a one-way street assignment for a connected, bridgeless graph, but it does not tell us how to find such an assignment. In this section we present an efficient algorithm for finding such an assignment if one exists.

Suppose that $G = (V, E)$ is any graph. We shall describe a procedure for orienting G which can be completed if and only if G is connected. If G has no bridges, the resulting orientation is strongly connected. For a proof of the second assertion, see Roberts [1976]. The procedure begins by using the depth-first search procedure described in Section 11.1.1 to label the vertices and mark the edges. If the labeling procedure cannot be completed, G is disconnected and hence has no

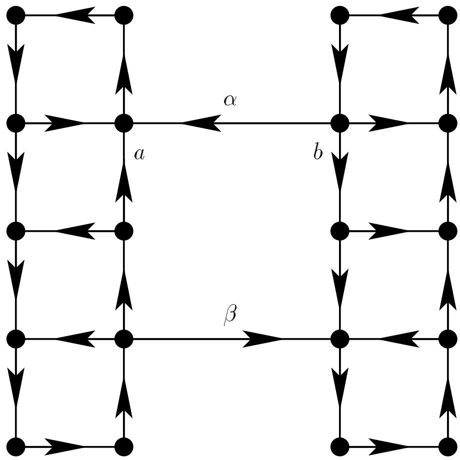


Figure 11.10: A satisfactory one-way assignment for the graph of Figure 11.9.

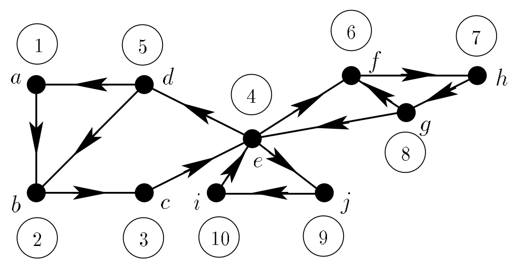


Figure 11.11: The strongly connected orientation for the graph G of Figure 11.1 obtained from the labeling and marking in that figure.

strongly connected orientation. The next step is to use the labeling and the set of marked edges T to define an orientation. Suppose that an edge is labeled with the labels i and j. If the edge is in T, that is, if it is marked, we orient from lower number to higher number, that is, from i to j if i is less than j and from j to i otherwise. If the edge is not in T, that is, if it is unmarked, we orient from higher number to lower number. We illustrate this procedure using the labeling and marking of Figure 11.1. The corresponding orientation is shown in Figure 11.11. Note that we orient from a to b because edge $\{a, b\}$ is in T and a gets the label 1, which is less than the label 2 given to b. However, we orient from d to b because edge $\{d, b\}$ is not in T and the label of d, that is, 5, is more than the label of b, that is, 2.

This algorithm always leads to an orientation of G if the labeling procedure can be completed, that is, if G is connected. However, it is guaranteed to lead to a strongly connected orientation only if G has no bridges. If we assume that the decision of how to orient an edge, given a labeling and a record of edges in T, takes about the same amount of time as each step in the depth-first search procedure which is used to label and record edges in T, then the computational complexity

$g(n, e)$ of this algorithm for an n vertex, e edge graph is at most the computational complexity $f(n, e)$ of the depth-first search procedure plus the number of edges in the graph G. In Section 11.1.2 we showed that $f(n, e)$ is at most $n + 2e$. Thus, $g(n, e)$ is at most $n + 3e$. [It is again an $O(n + e)$ algorithm.] In terms of n, the complexity is at most $\frac{3}{2}n^2 - \frac{3}{2}n + n$, since $e \leq \binom{n}{2}$. Thus, we have a polynomially bounded algorithm. [It is $O(n^2)$.] For other algorithms, see Boesch and Tindell [1980] and Chung, Garey, and Tarjan [1985]. For information on the use of parallel algorithms for strongly connected orientations, see Atallah [1984] and Vishkin [1985] and see a summary in Karp and Ramachandran [1990].

11.2.3 Efficient One-Way Street Assignments

Not every one-way street assignment for a city's streets is very efficient. Consider, for example, the one-way street assignment shown in the graph of Figure 11.10. This one-way street assignment is obtainable by the algorithm we have described (Exercise 4). If a person at location a wanted to reach location b, he or she used to be able to do it very quickly, but now has to travel a long way around. In short, this assignment meets the criterion we have set up, of being able to go from any point to any other point, but it does not give a very efficient solution to the traffic flow problem. In general, the most efficient one-way street assignment is one in which, "on the whole," distances traveled are not too great.

There are many possible notions of efficiency. We shall make some precise. These are all based on notions of distance. If G is a graph, recall that the distance $d(u, v)$ between vertices u and v is the length (number of edges) of the shortest chain between u and v in G. If D is a digraph, the *distance* $\bar{d}(u, v)$ from u to v is the length of the shortest path from u to v in D. One way to define efficiency, the only way we shall explore, is to say that an orientation is most efficient if it minimizes some "objective function" defined in terms of distances. Examples of such objective functions are given in Table 11.2. The problems of minimizing some of these objective functions are equivalent. For instance, minimizing (2) is equivalent to minimizing (3).

It turns out that we are in a "bad news-worse news" situation. The bad news is that no good algorithms are known for finding the most efficient strongly connected orientation according to any of the criteria in Table 11.2. The worse news is that some of these optimization problems are really hard. For example, Chvátal and Thomassen [1978] show that finding a strongly connected orientation that minimizes (1) is an NP-hard problem, to use the terminology of Section 2.18.

It should be remarked that efficiency is not always the goal of a one-way street assignment. In the National Park System throughout the United States, traffic congestion has become a serious problem. A solution being implemented by the U.S. National Park Service is to try to discourage people from driving during their visits to national parks. This can be done by designing very inefficient one-way street assignments, which make it hard to get from one place to another by car, and by encouraging people to use alternatives, for example bicycles or buses. Figure 11.12

Table 11.2: Some Objective Functions for Strongly Connected Orientations

Objective Function	Description
(1) $\max_{u,v} \bar{d}(u,v)$	Maximum distance traveled
(2) $\sum_{u,v} \bar{d}(u,v)$	Sum of distances traveled
(3) $\frac{1}{n^2-n} \sum_{u,v} \bar{d}(u,v)$	Average distance traveled
(4) $\max_{u,v} \left[\bar{d}(u,v) - d(u,v)\right]$	Maximum change in distance
(5) $\sum_{u,v} \left[\bar{d}(u,v) - d(u,v)\right]$	Sum of changes in distance
(6) $\frac{1}{n^2-n} \sum_{u,v} \left[\bar{d}(u,v) - d(u,v)\right]$	Average change in distance
(7) $\sum_{u} \max_{x} \bar{d}(u,x)$	Sum of maximum distance to travel
(8) $\sum_{u} \left[\max_{x} \bar{d}(u,x) - \max_{x} d(u,x)\right]$	Sum of changes in maximum distance to travel

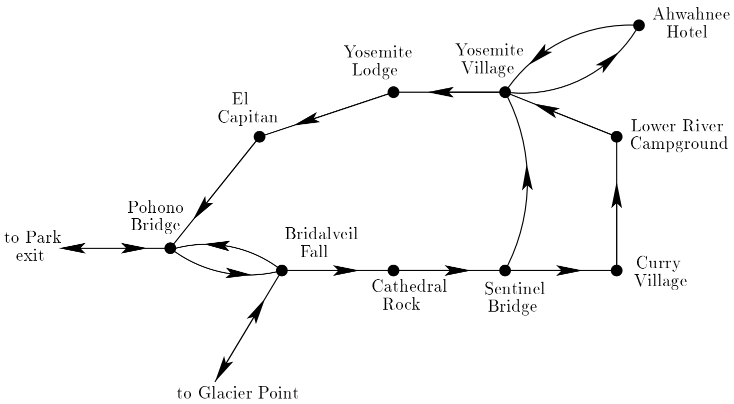


Figure 11.12: One-way street assignment for Yosemite Valley automobile traffic, summer of 1973. (*Note*: Buses may go both ways between Yosemite Lodge and Yosemite Village and between Yosemite Village and Sentinel Bridge.)

shows approximately the one-way street assignment instituted in the Yosemite Valley section of Yosemite National Park during the summer of 1973. (In addition to making roads one-way, the Park Service closed off others to cars entirely. Note that there is a two-way street; the idea of making *every* street one-way is obviously too simple. Also, in this situation, there is no satisfactory assignment in which every street is one-way. Why?) This is a highly inefficient system. For example, to get from Yosemite Lodge to Yosemite Village, a distance of less than 1 mile by road, it is necessary to drive all the way around via Pohono Bridge, a distance of over 8 miles. However, the Park's buses are allowed to go directly from Yosemite Lodge to Yosemite Village and many people are riding them!

From a mathematical point of view, it would be nice to find algorithms for finding an inefficient (the most inefficient) one-way street assignment, just as to find an efficient (the most efficient) such assignment. Unfortunately, no algorithms for inefficient assignments are known either.

11.2.4 Efficient One-Way Street Assignments for Grids

If a problem proves to be hard, one can explore simpler versions of the problem, solve it in special cases, and so on. This suggests the question: Are there good solutions for the problem of finding the most efficient strongly connected orientation if the graph is known to have a special structure? The structures of interest should, of course, have something to do with the application of interest. The most natural structures to explore are the grid graphs which correspond to cities with north-south streets and east-west avenues. We explore these in what follows. To fix some notation, we let G_{n_1,n_2} be the graph consisting of $n_1 + 1$ east-west avenues and $n_2 + 1$ north-south streets. Thus, for example, $G_{3,5}$ is the 4×6 grid. (The slightly unwieldy notation is borrowed from the literature of the subject, where it is useful in some of the proofs.)

One can always find an optimal strongly connected orientation by brute force: Try all orientations. However, brute force algorithms can be very impractical. Consider the brute force algorithm of trying all orientations of the grid graph $G_{n-1,n-1}$ consisting of n east-west avenues and n north-south streets. Each such orientation would be tested for strong connectedness, and if it is strongly connected, the objective function being minimized would be calculated. How many orientations of this grid graph are there?

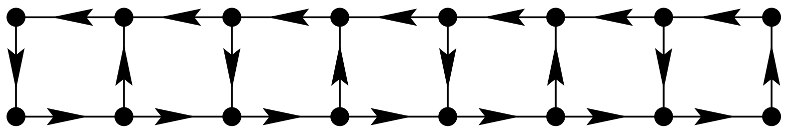
To count the number of orientations of our grid graph $G_{n-1,n-1}$, one observes that there are $n-1$ edges in each east-west avenue and $n-1$ edges in each north-south street. Thus, the number of edges e is given by

$$e = 2n(n-1) = 2n^2 - 2n$$

(see Exercise 15). Having computed the number of edges, we note that the number of orientations of $G_{n-1,n-1}$ is given by the product rule. Since each edge has two possible orientations, the total number of orientations is

$$\overbrace{2 \times 2 \times \cdots \times 2}^{e \text{ times}} = 2^e = 2^{2n^2-2n}.$$

D_1



D_2

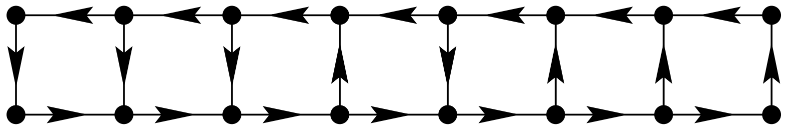


Figure 11.13: The alternating orientation D_1 and an optimal solution D_2 to the problem of minimizing measures (1) and (8) for the grid $G_{1,7}$.

Even if we could check 1 billion orientations per second, then when n is only 7, a simple calculation shows that it would take more than 500,000,000 years to check all of these orientations. The brute force method of trying all orientations is clearly impractical. This conclusion does not change even with an order-of-magnitude change in the speed of computation. (As an aside remark, we conclude that one should always count the number of steps an algorithm will take before implementing it.)

For each of the measures or objective functions defined in Table 11.2, it is a very concrete question to find the optimal orientation of the grid G_{n_1,n_2} in the sense of minimizing that measure. It is surprising that it took such a long time for it to be answered, and then only for some of the measures. The first efforts to find optimal strongly connected orientations for $G_{1,n}$, the grid of 2 east-west avenues and $n+1$ north-south streets, were carried out in 1973. Yet it was not until quite a bit later that a series of papers by Roberts and Xu [1988, 1989, 1992, 1994] and Han [1989] provided some solutions to these problems, at least for measures (1) and (8) of Table 11.2.

It is useful to describe here the optimal solutions for the case of $G_{1,n}$ with n odd. One can show that in this case, any solution that minimizes either measure (1) or (8) will have a cycle around the outside of the grid (Exercise 18). Thus, what is left is to decide on the orientations of interior north-south streets. One's intuition is to alternate these interior streets. It turns out that this alternating orientation is not optimal. Rather, it is optimal to orient the two middle streets opposite to their closest (east or west) boundaries, and to orient all other interior streets to agree with their closest boundaries (see Figure 11.13). Roberts and Xu [1989] prove that if $n \geq 5$, this orientation minimizes measures (1) and (8) among all strongly connected orientations of $G_{1,n}$, n odd. They also prove that in the case of measure (8), this orientation is the unique optimal orientation, up to the choice of direction to orient the cycle around the boundary.

It is of interest to compare the optimal solution to the alternating orientation, i.e., the orientation that alternates north-south streets. For instance, when measure (1) is used and $n = 5$, the optimal solution gets a score (measure) of 7 and the alternating orientation a score of 9. When $n = 7$, the scores go up to 9 and 11, respectively. When measure (8) is used, the difference is much greater. When $n = 5$, the optimal solution gets a score of 10 and the alternating orientation a score of 26; when $n = 7$, the scores are 12 and 32, respectively. (These scores are all easy to verify.)

Similar results are described in Roberts and Xu [1988, 1989, 1992, 1994] and Han [1989] for measures (1) and (8) for other grids. Some of this work is described in the exercises. However, finding optimal solutions for grids under other measures remain open problems. We are quickly led to the frontiers of mathematical research.

It is interesting to study the natural "New York City" solution which alternates the orientation of east-west avenues and of north-south streets. Although not optimal for measures (1) and (8), the New York City solution is better for some grids under some other measures than solutions that are optimal under measures (1) and (8). For example, this is true for the grid $G_{5,7}$ under measure (3). Thus, there is much more research to be done here.

It is also interesting to study optimal strongly connected orientations for other graphs. We consider annular cities, cities with circular beltways and spokes heading out from the city center, in the next section.

There are layers of complication that our simplified mathematical models have omitted. For instance, our models do not consider volume of traffic on different streets or different distances or times needed to traverse different streets. To describe another complication, some of the optimal solutions described by Roberts and Xu [1994] contain arcs (x, y) and (z, y). Imagine the problems at an intersection like y. If we add penalties for such intersections, we get a different optimization problem.

11.2.5 Annular Cities and Communications in Interconnection Networks

The problem of finding the most efficient one-way street assignment has also been studied for annular cities, cities with circular beltways and spokes heading out from the city center. In particular, we shall discuss the graph $AN(c, s)$, where there are c concentric circles around a center and s straight-line spokes crossing all circles. Figure 11.14 shows $AN(3, 8)$. We describe some results about $AN(c, s)$ in the exercises (Exercises 23 and 24). Here, we mention one result. Bermond, *et al.* [2000] study orientations that minimize measure (8) of Table 11.2 for annular cities. In particular, they show that if $s = 4k$ for some k and $c > k + 2$, there is a strongly connected orientation for which $\bar{D} = \max_{u,v} \bar{d}(u, v) = \max_{u,v} d(u, v) = D$, i.e., the maximum distance traveled is no more than the maximum distance traveled before the orientation.

The problem of identifying those graphs that have a strongly connected orientation for which $\bar{D} = D$ or $\bar{D}$ is close to D also arises in communications in interconnection networks. In "total exchange," initially each processor in such a network has an item of information that must be distributed to every other processor. This arises in a variety of parallel processing applications. The process is accomplished by a sequence of synchronous calls between processors. During each

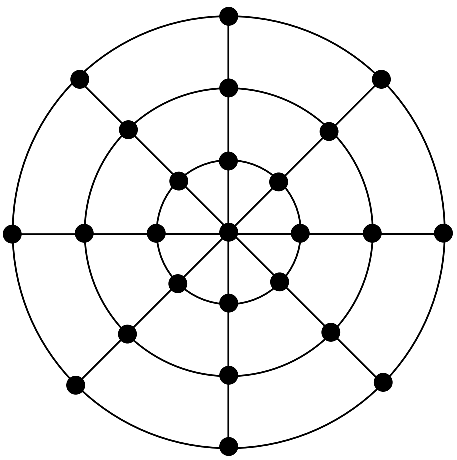


Figure 11.14: $AN(3,8)$

call, a processor can communicate with all its neighbors. In *full-duplex* Δ-*port*, communication can go both ways between neighbors. Thus, the minimum number of steps needed for all processors to get all the information is $\max_{u,v} d(u,v)$. In *half-duplex* Δ-*port*, simultaneous exchange on a given link is not authorized. Then, a given protocol corresponds to an orientation of edges. At each step, all the information known by a processor goes to all the processors to which there is an arrow in the orientation. Thus, $\bar{D}$ is an upper bound on the minimum number of steps needed for all the processors to get all the information. Since D is a lower bound, any network G for which there is a strongly connected orientation and which has $\bar{D}$ close to D has a protocol that has close to optimal performance.

The measure (1) or $\bar{D}$ has been studied for other graphs as well. For instance, Gutin [1994], Koh and Tan [1996a,b], and Plesnik [1986] found optimal orientations for the so-called complete multipartite graphs and McCanna [1988] for the n-dimensional cube, an important architecture in computer networks.

EXERCISES FOR SECTION 11.2

1. In each graph of Figure 11.3, find all bridges.
2. Apply the procedure described in the text for finding one-way street assignments to the graphs of Figure 11.3. In each case, check if the assignment obtained is strongly connected.
3. Repeat Exercise 2 for the graphs of Figure 3.23.
4. Show that the one-way street assignment shown in Figure 11.10 is attainable using the algorithm described in the text.
5. Find conditions for a graph to have a weakly connected orientation (Exercise 10, Section 3.2).

6. If a graph has a weakly connected orientation, how many weakly connected orientations does it have?
7. Give an example of a graph that has a weakly connected orientation but not a unilaterally connected orientation (Exercise 9, Section 3.2).
8. Let G be a connected graph. A *cut vertex* of G is a vertex with the following property: When you remove it and all edges to which it belongs, the result is a disconnected graph. In each graph of Figure 11.3, find all cut vertices.
9. Can a graph with a cut vertex (Exercise 8) have a strongly connected orientation?
10. Prove or disprove: A connected graph with no cut vertices (Exercise 8) has a strongly connected orientation.
11. Prove that an edge $\{u, v\}$ in a connected graph G is a bridge if and only if every chain from u to v in G includes edge $\{u, v\}$.
12. If u and v are two vertices in a digraph D, let $d_D(u, v) = \bar{d}(u, v)$ be the distance from u to v, i.e., the length of the shortest path from u to v in D. Distance is undefined if there is no path from u to v. Similarly, if u and v are two vertices in a graph G, the distance $d_G(u, v)$ from u to v is the length of the shortest chain from u to v in G. Again, distance is undefined if there is no chain.
    (a) Show that $d_D(u, v)$ may be different from $d_D(v, u)$.
    (b) Show that $d_G(u, v)$ always equals $d_G(v, u)$.
    (c) Show that if v is reachable from u and w from v, then $d_D(u, w) \leq d_D(u, v) + d_D(v, w)$.
13. In this exercise we consider the case of cities in which some streets are one-way to begin with. Our approach is based on Boesch and Tindell [1980]. A *mixed graph* consists of a set of vertices, some of which are joined by one-way arcs and some of which are joined by undirected edges. (The arcs correspond to one-way streets, the edges to two-way streets.) A mixed graph G can be translated into a digraph $D(G)$ by letting each edge be replaced by two arcs, one in each direction. G will be called *strongly connected* if $D(G)$ is strongly connected, and *connected* if $D(G)$ is weakly connected. An edge α in a connected mixed graph is called a *bridge* if removal of α but not its end vertices results in a mixed graph that is not connected.
    (a) Suppose that G is a strongly connected mixed graph and that $\{u, v\}$ is an edge of G. Let D' be the digraph obtained from $D(G)$ by omitting arcs (u, v) and (v, u) but not vertices u and v. Let A be the set of all vertices reachable from u by a path in D', less the vertex u. Let B be defined similarly from v. Suppose that u is not in B and v is not in A. Prove that the edge $\{u, v\}$ must be a bridge of G.
    (b) Use the result of part (a) to prove the following theorem of Boesch and Tindell [1980]: If G is a strongly connected mixed graph and $\{u, v\}$ is an edge of G that is not a bridge, there is an orientation of $\{u, v\}$ so that the resulting mixed graph is still strongly connected.
    (c) Prove from part (b) that every connected graph without bridges has a strongly connected orientation.
    (d) Translate your proof of part (c) into an algorithm for finding a strongly connected orientation of a connected, bridgeless graph.

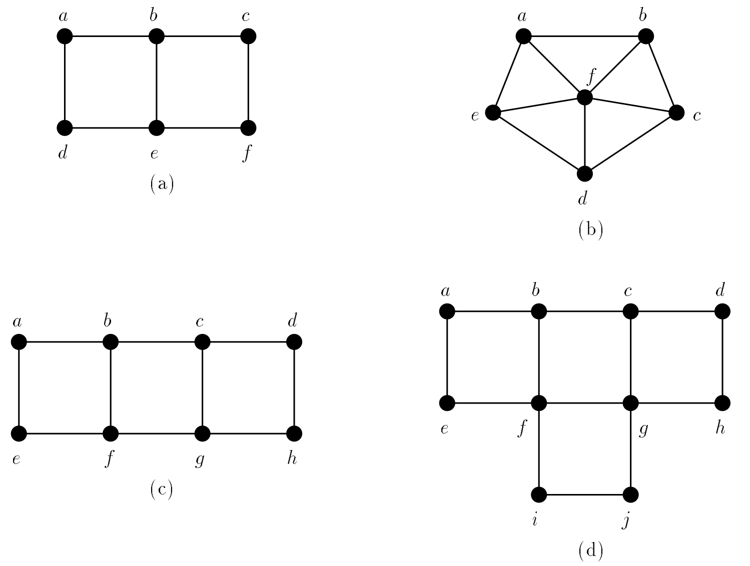


Figure 11.15: Graphs for exercises of Section 11.2.

(e) Illustrate your algorithm in part (d) on the graphs of Figure 11.3.

(f) What is the computational complexity of your algorithm in part (d)? How does this compare to the algorithm described in the text?

14. Find the number of orientations (not necessarily strongly connected) of

(a) $G_{2,n}$ (b) $G_{5,11}$ (c) $AN(4,6)$

15. Use Theorem 3.1 to calculate the number of edges in the grid graph $G_{n-1,n-1}$.

16. For each of measures (1)–(8) of Table 11.2, find a most efficient strongly connected orientation for each graph of Figure 11.15.

17. Calculate the measures (1)–(8) of Table 11.2 for the orientations of Figure 11.16. These are, respectively, the alternating orientation and optimal solution analogues to D_1 and D_2 of Figure 11.13 for $G_{1,7}$.

18. Show that in the case of $G_{1,n}$, n odd, any solution that minimizes either measure (1) or (8) of Table 11.2 will have a cycle around the outside of the grid.

19. When n is even, let orientation O_1 of $G_{1,n}$ be obtained as follows. Orient the outside of the grid with a counterclockwise cycle. Let $k+1$ be $(n/2)+1$. Orient the $(k-1)$st north-south street up, the $(k+2)$nd north-south street down, and all others to agree with the 1st or $(n+1)$st north-south street, whichever is closer. Roberts and Xu [1989] show that O_1 is optimal for measures (1) and (8) of Table 11.2 for $n \geq 10$.

(a) Draw O_1 for $G_{1,10}$.

(b) Calculate measures (1) and (8) of Table 11.2 for O_1.

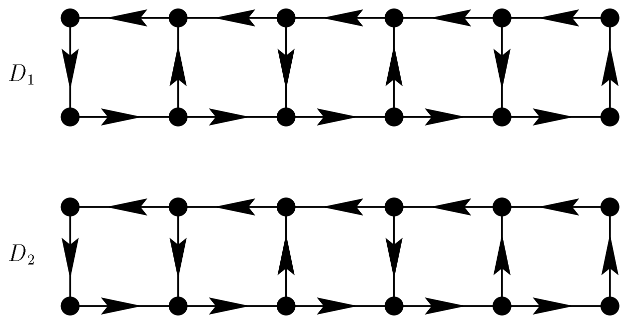


Figure 11.16: Orientations for grid $G_{1,5}$.

(c) Why is the orientation with alternating north-south streets, together with the outside counterclockwise cycle, not an optimal solution for these measures for $G_{1,n}$, n even?

20. When n is odd, let orientation O_2 of $G_{2,n}$ be obtained as follows. Orient the top and bottom east-west avenues from right to left and the middle east-west avenue from left to right. Orient the north-south streets joining the top and middle east-west avenues as follows: Let $k = (n/2) - 1$. Street 1 is down, street $n+1$ is up, the "middle streets" k and $k+1$ disagree with (have opposite orientations of) the closer of streets 1 and $n+1$, while all others have the same orientation as the closer of streets 1 and $n+1$. Orient the north-south streets joining the middle and bottom east-west avenues as follows: Street 1 is up, street $n+1$ is down, streets $k-1$ and $k+1$ disagree with the closer of streets 1 and $n+1$, while all others have the same orientation as the closer of streets 1 and $n+1$. Roberts and Xu [1989] show that O_2 is an optimal orientation under measures (1) and (8) of Table 11.2 for $n \geq 7$.

    (a) Draw O_2 for $G_{2,7}$.
    (b) Draw O_2 for $G_{2,9}$.
    (c) Calculate measures (1) and (8) of Table 11.2 for $G_{2,7}$.
    (d) Calculate measures (1) and (8) of Table 11.2 for $G_{2,9}$.
    (e) Suppose that you alternate east-west avenues, with top and bottom going right to left and middle going left to right; and you alternate north-south streets, with street 1 going down all the way, street 2 going up all the way, and so on. Why is this not a satisfactory orientation?
    (f) Modify the "alternating" orientation of part (e) by reversing the bottom half of north-south street 1 and north-south street $n+1$. Compare measures (1) and (8) of Table 11.2 for $G_{2,7}$ under this modified alternating orientation and O_2.
    (g) In O_2, do we ever find orientations with arcs (x, y) and (z, y)?
    (h) Calculate measures (2)–(7) of Table 11.2 for O_2 for $G_{2,7}$ and for the modified alternating orientation of part (f). Which is better?

21. Let $\bar{d}(u) = \max_x \bar{d}(u, x)$, $d(u) = \max_x d(u, x)$, and $m(u) = \bar{d}(u) - d(u)$. Then measure (8) of Table 11.2 is $\sum_u m(u)$. A *corner point* in G_{n_1,n_2} is one of the points

at the intersection of the bounding east-west avenues and north-south streets. A *limiting path* from u to v in an orientation O of graph G is a path of length $\bar{d}(u,v)$ equal to $d(u,v)$.

(a) If w is a corner point farthest from u in G_{n_1,n_2} and there is no limiting path in O from u to w, show that $m(u) \geq 2$.

(b) Illustrate the result of part (a) with orientation O_2 of Exercise 20.

(c) Repeat part (b) with the orientation of part (f) of Exercise 20.

22. Suppose that $n_1 = 2k_1 + 1$ is odd, $i = k_1$ or $k_1 + 1$. Suppose that w is a corner point (Exercise 21) in G_{n_1,n_2} farthest from the point u on the ith east-west avenue and jth north-south street and w' is the opposite corner point on the same north-south street as w.

(a) If there is no limiting path from u to w', show that $m(u) \geq 1$ (see Exercise 21).

(b) Illustrate the result of part (a) with orientation O_2 of Exercise 20.

(c) Repeat part (b) with the orientation of part (f) of Exercise 20.

23. (Bermond, *et al.* [2000]) Let $D = \max_{u,v} d(u,v)$ and consider the graph $AN(c,s)$.

(a) Show that if $c \leq \left\lfloor \frac{s}{4} \right\rfloor$, then $D = 2c$.

(b) Show that if $c \geq \left\lfloor \frac{s}{4} \right\rfloor$, then $D = c + \left\lfloor \frac{s}{4} \right\rfloor$.

24. (Bermond, *et al.* [2000]) Let O_3 be the following orientation of $AN(c,s)$. Alternate orientations of the spokes with the first spoke oriented in from outer circle to center. If s is odd, spokes 1 and s will have the same orientation. Orient circles as follows: The first circle (starting from the center) is oriented counterclockwise, the next circle clockwise, the next counterclockwise, and so on, through to the next-to-last circle. On the outer circle, orient arcs going from spoke 2 to spoke 1 and spoke 2 to spoke 3, spoke 4 to spoke 3 and spoke 4 to spoke 5, and so on. If s is odd, the part of the outer circle from spoke 2 to spoke 1 to spoke s forms the only path of length 2.

(a) If $\bar{D} = \max_{u,v} \bar{d}(u,v)$ for O_3 and $D = \max_{u,v} d(u,v)$ for $AN(c,s)$, show that $\bar{D} \leq D + 2$ if $c \leq k$, where $s = 4k + r$, $r < 4$.

(b) By the same reasoning with O_3, show that $\bar{D} \leq D + 3$ if s is odd and $c = k + 1$.

25. One of the major concerns with transportation networks, communication networks, telephone networks, electrical networks, and so on, is to construct them so that they are not very vulnerable to disruption. Motivated by the notion of bridge, we shall be interested in Exercises 25–33 in the disruption of connectedness that arises from the removal of arcs or vertices. We say that a digraph D is in *connectedness category 3* if it is strongly connected, in *category 2* if it is unilaterally connected but not strongly connected, in *category 1* if it is weakly connected but not unilaterally connected, and in *category 0* if it is not weakly connected. Find the connectedness category of every digraph of Figure 3.7.

26. If digraph D is in connectedness category i (Exercise 25), we say that arc (u,v) is an *(i,j)-arc* if removal of (u,v) (but not vertices u and v) results in a digraph of category j. For each pair (i,j), $i = 0,1,2,3$, and $j = O,1,2,3$, either give an example of a digraph with an (i,j)-arc or prove there is no such digraph.

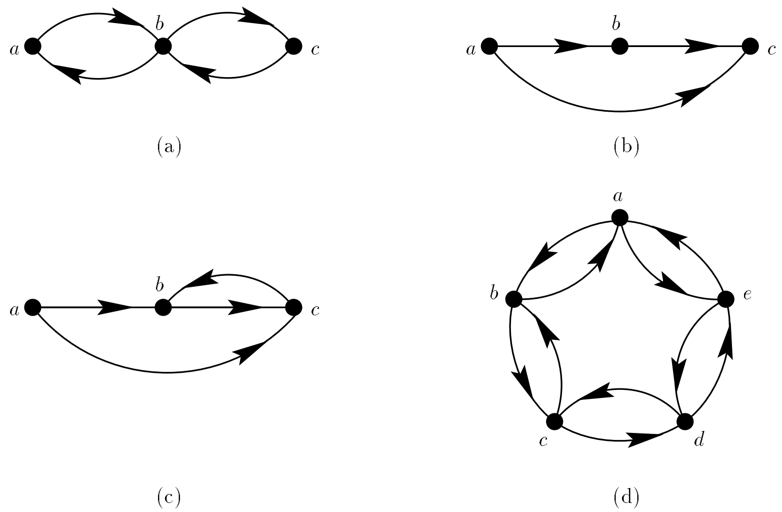


Figure 11.17: Digraphs for exercises of Section 11.2.

27. If digraph D is in connectedness category i, we say that vertex u is an (i, j)-*vertex* if the subgraph generated by vertices of D other than u is in category j. For each pair $(i, j), i = 0, 1, 2, 3$ and $j = 0, 1, 2, 3$, either give an example of a digraph with an (i, j)-vertex or prove that there is no such digraph.

28. Let us define the *arc vulnerability* of a digraph in connectedness categories $1, 2$, or 3 as the minimum number of arcs whose removal results in a digraph of lower connectedness category. For each digraph of Figure 11.17, determine its arc vulnerability.

29. Give an example of a digraph D with arc vulnerability equal to 4.

30. For every k, give an example of a digraph D with arc vulnerability equal to k.

31. Give an example of a digraph with n vertices and arc vulnerability equal to $n - 1$. Could there be a strongly connected digraph with n vertices and arc vulnerability equal to n?

32. (a) If D is strongly connected, what is the relation between the arc vulnerability of D and the minimum *indegree* of a vertex, the minimum number of incoming arcs of a vertex?

    (b) Show that the arc vulnerability of a strongly connected digraph is at most a/n, where a is the number of arcs and n the number of vertices.

33. (Whitney [1932]) If there is a set of vertices in a digraph D whose removal results in a digraph in a lower connectedness category than D, we define the *vertex vulnerability* of D to be the size of the smallest such set of vertices. Otherwise, vertex vulnerability is undefined.

    (a) Show that if D is weakly connected and there is at least one pair $u \neq v$ such that neither (u, v) nor (v, u) is an arc of D, then the vertex vulnerability of D is less than or equal to the minimum total degree of any vertex of D. [The *total degree* of a vertex x is the sum of the *indegree* (the number of incoming arcs) of x and the *outdegree* (the number of outgoing arcs) of x.]

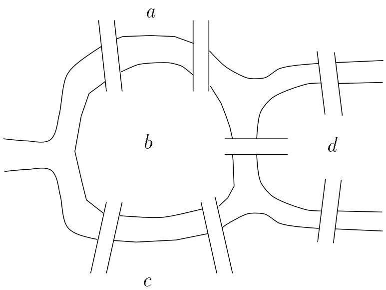


Figure 11.18: The Königsberg bridges.

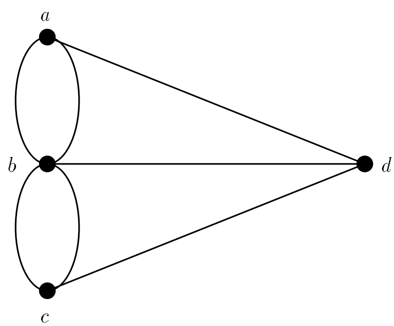


Figure 11.19: Multigraph obtained from Figure 11.18.

(b) Show that the vertex vulnerability and the minimum total degree can be different.

(c) Why do we need to assume that there is at least one pair $u \neq v$ such that neither (u, v) nor (v, u) is an arc of D?

11.3 EULERIAN CHAINS AND PATHS

11.3.1 The Königsberg Bridge Problem

Graph theory was invented by the famous mathematician Leonhard Euler [1736] in the process of settling the famous Königsberg bridge problem.[4] Euler's techniques have found modern applications in the study of street sweeping, mechanical plotting by computer, RNA chains, coding, telecommunications, and other subjects, and we survey some of these applications in Section 11.4. Here, we present the Königsberg bridge problem and then present general techniques arising from its solution.

The city of Königsberg had seven bridges linking islands in the River Pregel to the banks and to each other, as shown in Figure 11.18. The residents wanted to know if it was possible to take a walk that starts at some point, crosses each bridge exactly once, and returns to the starting point. Euler translated this into a graph theory problem by letting the various land areas be vertices and joining two vertices by one edge for each bridge joining them. The resulting object, shown in Figure 11.19, is called a *multigraph*. We shall use the terms *multigraph* and *multidigraph* when more than one edge or arc is allowed between two vertices x and y or from vertex x to vertex y. The concepts of connectedness, such as chain, circuit, path, cycle, component, and so on, are defined just as for graphs and digraphs. (Note, however, that in a multigraph, we can have a circuit from a to b to a if there are two edges between a and b.) Loops will be allowed here.

We say that a *chain* in a multigraph G or *path* in a multidigraph D is *eulerian* if it uses every edge of G or arc of D once and only once. Euler observed that the

[4] See "The Königsberg Bridges," *Scientific American, 189* (1953), 66–70.

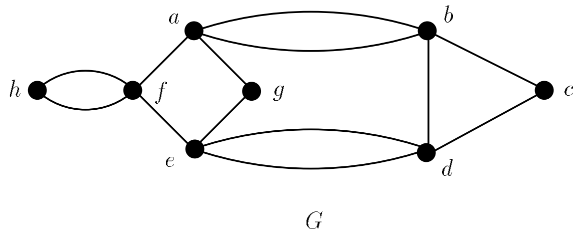


Figure 11.20: Multigraph with an eulerian closed chain.

citizens of Königsberg were seeking an *eulerian closed chain* in the multigraph of Figure 11.19. He asked the following general question: When does a multigraph G have an eulerian closed chain? We begin by attempting to answer this question. Clearly, if G has an eulerian closed chain, G must be *connected up to isolated*[5] *vertices*, that is, at most one component of G has an edge. Moreover, an eulerian closed chain must leave each vertex x as often as it enters x, and hence each vertex x must have even degree, where the degree of x, the reader will recall, is the number of neighbors of x (except that for loops from x to x we add 2 to the degree of x). Euler discovered the following result.

Theorem 11.2 (Euler) A multigraph G has an eulerian closed chain if and only if G is connected up to isolated vertices and every vertex of G has even degree.

Before proving the sufficiency of the condition stated in Theorem 11.2, we observe that the multigraph of Figure 11.19 does not have an eulerian closed chain. For vertex a has odd degree. Thus, the citizens of Königsberg could not complete their walk. To further illustrate the theorem, we note that the multigraph G of Figure 11.20 is connected and every vertex has even degree. An eulerian closed chain is given by $a, b, c, d, e, f, h, f, a, g, e, d, b, a$. If every vertex of a multigraph G has even degree, we say that G is *even*.

11.3.2 An Algorithm for Finding an Eulerian Closed Chain

A good way to prove sufficiency in Theorem 11.2 is to describe a procedure for finding an eulerian closed chain in an even multigraph. Start with any vertex x that has a neighbor and choose any edge joining x, say $\{x, y\}$. Next, choose any edge $\{y, z\}$ joining y, making sure not to choose an edge used previously. Continue at any vertex by choosing a previously unused edge which joins that vertex. Now each time we pass through a vertex, we use up two adjoining edges. Thus, the number of unused edges joining any vertex other than x remains even at all times. It follows that any time we enter such a vertex other than x, we can leave it. Since we started at x, the number of unused edges joining x remains odd at all times. Continue the procedure until it is impossible to continue. Now since every vertex except x can be left whenever it is entered, the only way the procedure can end is

[5] A vertex is called *isolated* if it has no neighbors.

back at x. Let us illustrate it on the multigraph G of Figure 11.20. Suppose that x is c. We can use edge $\{c,b\}$ to get to b, then edge $\{b,a\}$ to get to a, then edge $\{a,g\}$ to get to g, then edge $\{g,e\}$ to get to e, then edge $\{e,d\}$ to get to d, and then edge $\{d,c\}$ to return to c. At this point, we can go no further. Note that we have found a closed chain c,b,a,g,e,d,c starting and ending at c which uses each edge of G at most once. However, the chain does not use up all the edges of G. Let us call the procedure so far "finding a closed chain from x to x," or CL CHAIN (x,x) for short.

The algorithm can now continue. Note that at this stage, we have a chain C from x to x. Moreover, every vertex has an even number of unused joining edges. Since the graph is connected up to isolated vertices, if there is any unused edge, there must be at least one unused edge joining a vertex u on the chain C. In the multigraph of unused edges, we apply the procedure CL CHAIN (u,u) to find a closed chain D from u to u which uses each previously unused edge at most once. We can now modify our original closed chain C from x to x by inserting the "detour" D when we first hit u. We get a new closed chain C' from x to x which uses each edge of G at most once. If there are still unused edges, we repeat the procedure. We must eventually use up all the edges of the original multigraph. Continuing with our example, note that we have so far obtained the closed chain $C = c,b,a,g,e,d,c$. One unused edge joining this chain is the edge $\{a,f\}$. Since a is on C, we look for a closed chain of unused edges from a to a. Such a chain is found by our earlier procedure CL CHAIN (a,a). One example is $D = a,f,e,d,b,a$. Note that we used the second edges $\{e,d\}$ and $\{b,a\}$ here. The first ones have already been used. We insert the detour D into C, obtaining the closed chain $C' = c,b,a,f,e,d,b,a,g,e,d,c$. Since there are still unused edges, we repeat the process again. We find a vertex f on C' that joins an unused edge, and we apply CL CHAIN (f,f) to find a closed chain f,h,f from f to f. We insert this detour into C' to find $C'' = c,b,a,f,h,f,e,d,b,a,g,e,d,c$, which uses each edge once and only once.

The algorithm we have described can now be formalized. We have a subroutine called CL CHAIN (x,x) which is described as follows.

Algorithm 11.3: CL CHAIN (x,x)

Input: A multigraph G, a set U of unused edges of G, with each vertex appearing in an even number of edges in U, and a designated vertex x that appears in some edge of U.

Output: A closed chain from x to x that uses each edge of U at most once.

Step 1. Set $v = x$ and output x. (Here, v is the last vertex visited.)

Step 2. If there is an edge $\{v,y\}$ in U, set $v = y$, output y, remove $\{v,y\}$ from U, and repeat this step.

Step 3. If there is no edge $\{v,y\}$ in U, stop. The outputs in order give us the desired chain from x to x.

Using Algorithm 11.3, we can now summarize the entire procedure.

Algorithm 11.4: Finding an Eulerian Closed Chain

Input: An even multigraph G that is connected up to isolated vertices.
Output: An eulerian closed chain.

Step 1. Find any vertex x that has a neighbor. (If there is none, every vertex is isolated and any vertex x alone defines an eulerian closed chain. Output this and stop.) Let $U = E(G)$.

Step 2. Apply CL CHAIN (x, x) to obtain a chain C.

Step 3. Remove all edges in C from U. (Technically, this is already done in step 2.)

Step 4. If $U = \emptyset$, stop and output C. If $U \neq \emptyset$, find a vertex u on C that has a joining edge in U, and go to step 5.

Step 5. Apply CL CHAIN (u, u) to obtain a chain D.

Step 6. Redefine C by inserting the detour D at the first point u is visited, remove all edges of D from U, and go to step 4.

For a more formal statement of this algorithm, see Even [1979]. Even shows that the algorithm can be completed in the worst case in a number of steps that is a constant k times the number of edges e. Hence, if G is a graph (i.e., if there are no multiple edges), the computational complexity of the algorithm we have described is

$$ke \leq k\frac{n(n-1)}{2} = \frac{k}{2}n^2 - \frac{k}{2}n,$$

a polynomial bound in terms of n. [In the terminology of Section 2.18, this is an $O(e)$ or, for graphs, an $O(n^2)$ algorithm.] Since it is clear that the algorithm works, Theorem 11.2 is proved.

An alternative algorithm for finding an eulerian closed chain would use the depth-first search procedure of Section 11.1. We leave the description of such an algorithm to the reader (Exercise 18).

11.3.3 Further Results about Eulerian Chains and Paths

The next theorem tells when a multigraph has an eulerian chain (not necessarily closed). We leave the proof to the reader.

Theorem 11.3 (Euler) A multigraph G has an eulerian chain if and only if G is connected up to isolated vertices and the number of vertices of odd degree is either 0 or 2.

According to this theorem, the multigraph of Figure 11.19 does not have an eulerian chain. However, the multigraph of Figure 11.21 does, since there are exactly two vertices of odd degree.

We now state analogous results for multidigraphs, leaving the proofs to the reader. In these theorems, the *indegree* (*outdegree*) of a vertex is the number of

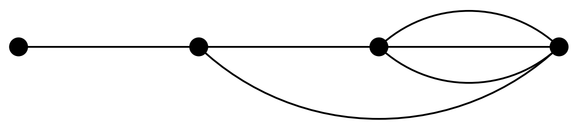
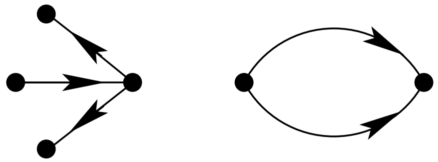


Figure 11.21: Multigraph with an eulerian chain.



(a) (b)

Figure 11.22: Two multidigraphs without eulerian paths.

incoming (outgoing) arcs. A digraph (multidigraph) is called *weakly connected* if when all directions on arcs are disregarded, the resulting graph (multigraph) is connected.

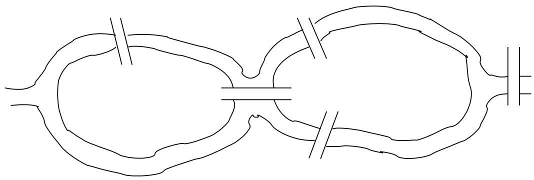
Theorem 11.4 (Good [1946]) A multidigraph D has an eulerian closed path if and only if D is weakly connected up to isolated vertices[6] and for every vertex, indegree equals outdegree.

Theorem 11.5 (Good [1946]) A multidigraph D has an eulerian path if and only if D is weakly connected up to isolated vertices and for all vertices with the possible exception of two, indegree equals outdegree, and for at most two vertices, indegree and outdegree differ by one.

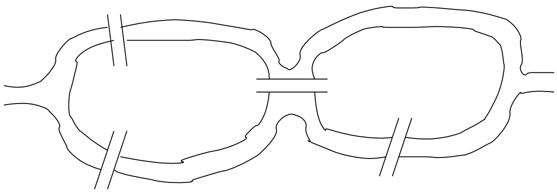
Theorem 11.5 is illustrated by the two multidigraphs of Figure 11.22. Neither has an eulerian path. In the first example, there are four vertices where indegree is different from outdegree. In the second example, there are just two such vertices, but for each the indegree and outdegree differ by more than 1. Note that the hypotheses of Theorem 11.5 imply that if indegree and outdegree differ for any vertex, then exactly one vertex has an excess of one indegree and exactly one vertex has an excess of one outdegree (see Exercise 9). These vertices turn out to correspond to the first and last vertices of the eulerian path.

Note that Theorems 11.2–11.5 hold if there are loops. Note also that in a multigraph, a loop adds 2 to the degree of a vertex, while in a multidigraph, it adds 1 to indegree and 1 to outdegree. Thus, loops do not affect the existence of eulerian (closed) chains or paths.

[6] That is, when directions on arcs are disregarded, the underlying multigraph is connected up to isolated vertices.



(a)



(b)

Figure 11.23: Rivers and bridges for Exercise 1, Section 11.3.

EXERCISES FOR SECTION 11.3

1. For each river with bridges shown in Figure 11.23, build a corresponding multigraph and determine if it is possible to take a walk that starts at a given location, crosses each bridge once and only once, and returns to the starting location.
2. Which of the multigraphs of Figure 11.24 have an eulerian closed chain? For those that do, find one.
3. Of the multigraphs of Figure 11.24 that do not have an eulerian closed chain, which have an eulerian chain?
4. Which of the multidigraphs of Figure 11.25 have an eulerian closed path? For those that do, find one.
5. Of the multidigraphs of Figure 11.25 that do not have an eulerian closed path, which have an eulerian path?
6. For each multigraph G of Figure 11.26, apply the subroutine CL CHAIN (a, a) to the vertex a with $U = E(G)$.
7. For each multigraph of Figure 11.26, apply Algorithm 11.4 to find an eulerian closed chain.
8. How would you modify Algorithm 11.4 to find an eulerian chain from x to y?
9. Show that in a multidigraph with an eulerian path but no eulerian closed path, exactly one vertex has an excess of one indegree and exactly one vertex has an excess of one outdegree.
10. (a) Can the drawing of Figure 11.27(a) be made without taking your pencil off the paper or retracing?

    (b) What about Figure 11.27(b)? (c) What about Figure 11.27(c)?

    (d) What about Figure 11.27(d)?

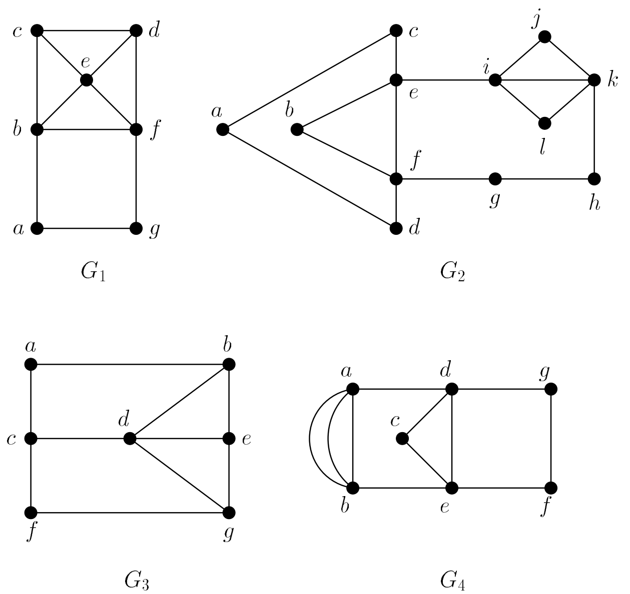


Figure 11.24: Multigraphs for exercises of Section 11.3.

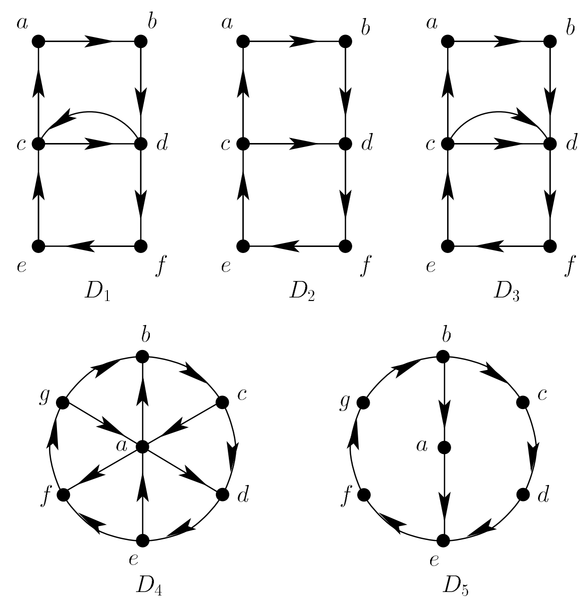


Figure 11.25: Multidigraphs for exercises of Section 11.3.

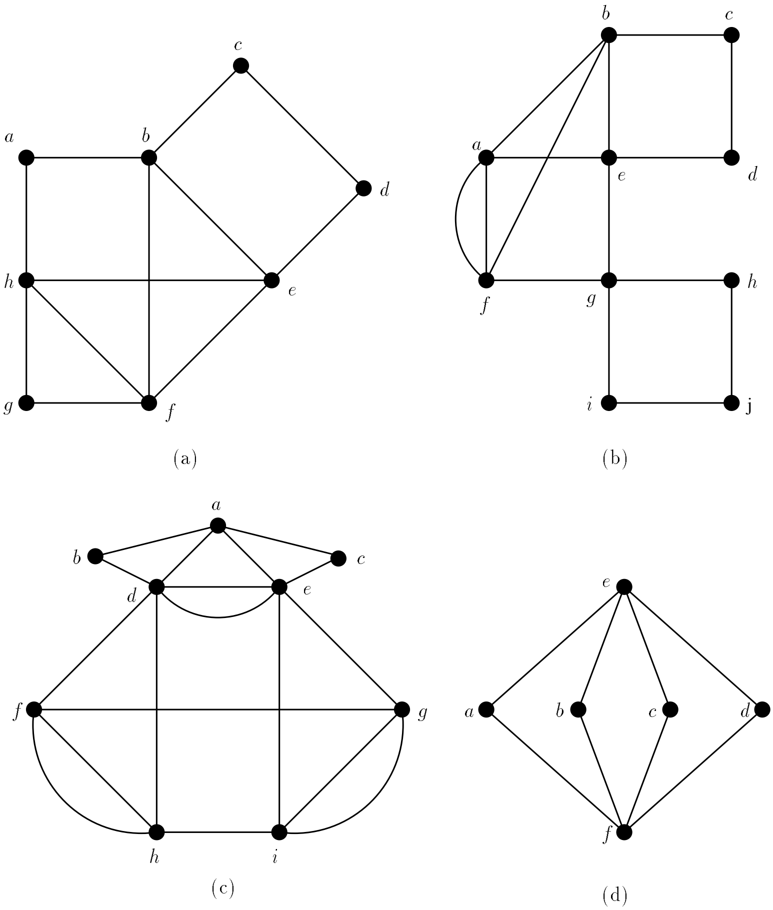


Figure 11.26: Multigraphs for exercises of Section 11.3.

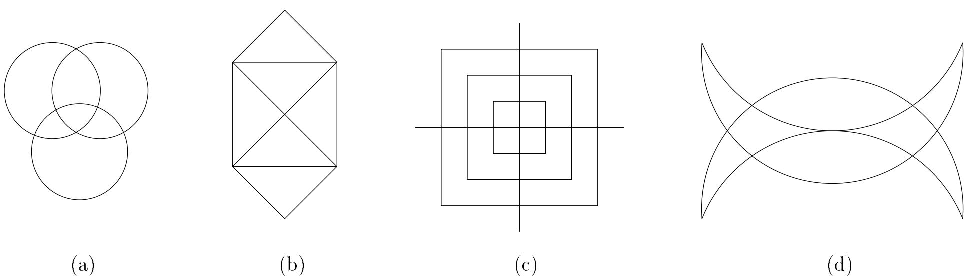


Figure 11.27: Drawings for Exercise 10, Section 11.3.

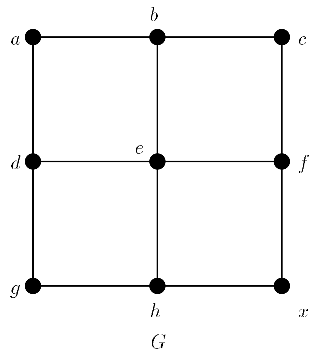
11. (Harary and Palmer [1973]) Show that the number of labeled even graphs of n vertices equals the number of labeled graphs of $n-1$ vertices. (*Hint:* To a labeled graph G of $n-1$ vertices, add a vertex labeled n and join it to the vertices of G of odd degree.)
12. (a) For each multidigraph of Figure 11.25 that has an eulerian closed path, find the number of such paths starting and ending at a.
    (b) For each multidigraph of Figure 11.25 that doesn't have an eulerian closed path but has an eulerian path, find the number of such paths.
13. Find the number of eulerian closed chains starting and ending at a for the graph of Figure 11.26(d).
14. Show that every digraph without isolated vertices and with an eulerian closed path is strongly connected.
15. Show that every digraph without isolated vertices and with an eulerian path is unilaterally connected.
16. Does every strongly connected digraph have an eulerian closed path? Why?
17. Does every unilaterally connected digraph have an eulerian path? Why?
18. Describe an algorithm for finding an eulerian closed chain that uses the depth-first search procedure.
19. Prove the necessity part of Theorem 11.3.
20. Prove the sufficiency part of Theorem 11.3.
21. Prove Theorem 11.4.
22. Prove Theorem 11.5.

11.4 APPLICATIONS OF EULERIAN CHAINS AND PATHS

In this section we present various applications of the ideas presented in Section 11.3. The subsections of this section are independent, except that Sections 11.4.2 and 11.4.3 depend on Section 11.4.1. Other than this, the subsections may be used in any order. The reader without the time to cover all of these subsections might sample Sections 11.4.1, 11.4.2, and 11.4.4 or 11.4.6.

11.4.1 The "Chinese Postman" Problem

A mail carrier starting out from a post office must deliver letters to each block in a territory and return to the post office. What is the least amount of walking the mail carrier can do? This problem was originally studied by Kwan [1962], and has traditionally been called the *"Chinese Postman" Problem*, or just the *Postman Problem.* A similar problem is faced by many delivery people, by farmers who have fields to seed, by street repair crews, and so on.



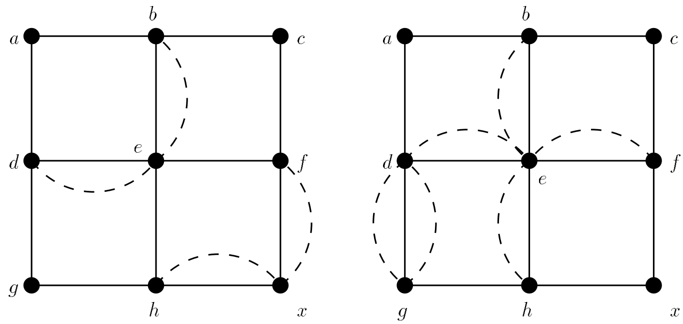


Figure 11.28: Graph representing a mail carrier's territory and two multigraphs corresponding to mail carriers' routes.

We can represent the mail carrier's problem by building a graph G with each vertex representing a street corner and each edge a street.[7] Assuming for simplicity that the post office is near a street corner, the mail carrier then seeks a closed chain in this graph, which starts and ends at the vertex x corresponding to the corner where the post office is located, and which uses each edge of the graph *at least once*.

If the graph G has an eulerian closed chain, we can pick up this chain at x and follow it back to x. No chain can give a shorter route. If there is no such eulerian closed chain, we can formulate the problem as follows. Any mail carrier's route will use all of the edges of G once and possibly some of them more than once. Suppose that we modify G by replacing each edge by as many copies of it as are in the mail carrier's route, or equivalently, by adding to G enough copies of each edge to exactly achieve the mail carrier's route. Then in the resulting multigraph, the mail carrier's route corresponds to an eulerian closed chain from x to x. For instance, consider the graph G of Figure 11.28, which represents a four-square-block area in a city. There is no eulerian closed chain in G, because for instance vertex d has degree 3. A possible mail carrier's route would be the closed chain $x, h, g, d, e, f, c, b, a, d, e, b, e, h, x, f, x$.

[7]This already simplifies the problem. A street between x and y with houses on both sides should really be represented by two edges between x and y. A mail carrier can walk up one side of the street first, and later walk up (or down) the other side.

This corresponds to the first multigraph shown in Figure 11.28. (Added edges are dashed.) An alternative route would be $x, h, e, h, g, d, e, d, g, d, a, b, e, b, c, f, e, f, x$. This corresponds to the second multigraph shown in Figure 11.28. The first of these routes is the shorter; equivalently, it requires the addition of fewer copies of edges of G. (The second route can be shortened by omitting the d, g, d part.)

The problem of trying to find the shortest mail carrier's route from x to x in G is equivalent to the problem of determining the smallest number of copies of edges of G to add to G to obtain a multigraph that has an eulerian closed chain, that is, one in which all vertices have even degree. A general method for solving this combinatorial optimization problem is to translate it into the maximum-weight matching problem of the type we discuss in Chapter 12. We discuss this translation specifically in Section 12.7.1. In our example it is easy to see that since there are four vertices of odd degree, and no two are adjacent, at least four edges must be added. Hence, the first multigraph of Figure 11.28 corresponds to a shortest mail carrier's route.

In Section 11.4.3 we generalize this problem. For a good general discussion of the "Chinese Postman" Problem (and eulerian "tours"), see Lawler [1976], Minieka [1978], and Johnson [2000].

11.4.2 Computer Graph Plotting

Reingold and Tarjan [1981] point out that the "Chinese Postman" Problem of Section 11.4.1 arises in mechanical plotting by computer[8] of a graph with prespecified vertex locations. Applications of mechanical graph plotting described by Reingold and Tarjan include shock-wave propagation problems, where meshes of thousands of vertices must be plotted; map drawing; electrical networks; and activity charts (see Reingold and Tarjan's paper for references).

Much time is wasted in mechanical graph plotting when the plotter pen moves with the pen off the paper. Thus, we seek to minimize the number of moves from vertex to vertex with the pen off the paper. This is exactly the problem of minimizing the number of edges to be added to the graph being plotted so as to obtain a multigraph with an eulerian closed chain.

11.4.3 Street Sweeping

A large area for applications of combinatorial techniques is the area of urban services. Cities spend billions of dollars a year providing such services. Combinatorics has been applied to problems involving the location and staffing of fire and police stations, design of rapid transit systems, assignments of shifts for municipal workers, routing of street-sweeping and snow-removal vehicles, and so on. See Beltrami [1977], Dror [2000], and Helly [1975] for a variety of examples. We discussed in Example 3.14 the problem of routing garbage trucks to pick up garbage. Here

[8] Although still used for CAD applications, pen plotters are quickly becoming obsolete, due to the advent of ink jet and laser printers.

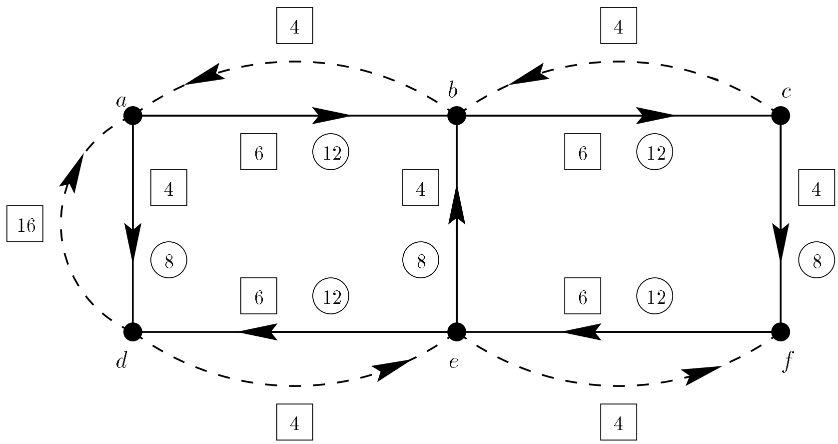


Figure 11.29: Curb multidigraph, with solid arcs defining the sweep subgraph, deadheading times in squares, and sweep times in circles.

we discuss the problem of determining optimal routes for street-sweeping or snow-removal equipment. We follow Tucker and Bodin [1983] and Roberts [1978]; see also Liebling [1970].

Consider the street-sweeping problem for concreteness. Let the street corners in the neighborhood to be swept be the vertices of a multidigraph. Include an arc from x to y if there is a curb that can be traveled from x to y. In general, a one-way street will give rise to two arcs from x to y, which is why we get a multidigraph. Call this multidigraph the *curb multidigraph*.

During a given period of time, certain curbs are to be swept. The corresponding arcs define a subgraph of the curb multidigraph called the *sweep subgraph*. (By means of parking regulations, curbs to be swept in a large city such as New York are kept free of cars during the period in question.)

Now any arc in the sweep subgraph has associated with it a number indicating the length of time required to sweep the corresponding curb. Also, any arc in the curb multidigraph has associated with it a number indicating the length of time required to follow the corresponding curb without sweeping it. This is called the *deadheading time*.

Figure 11.29 shows a curb multidigraph. Some arcs are solid—these define the sweep subgraph. Each arc has a number in a square; this is the deadheading time. The solid arcs have in addition a number in a circle; this is the sweep time.

We would like to find a way to start from a particular location (the garage), sweep all the curbs in the sweep subgraph, return to the start, and use as little time as possible. This is a generalization of the "Chinese Postman" Problem studied in Section 11.4.1, and our approach to it is similar. We seek a closed path in the curb multidigraph which includes all arcs of the sweep subgraph. The time associated with any acceptable path is the sum of the sweeping times for arcs swept plus the sum of the deadheading times for arcs in the path that are not swept. Note that

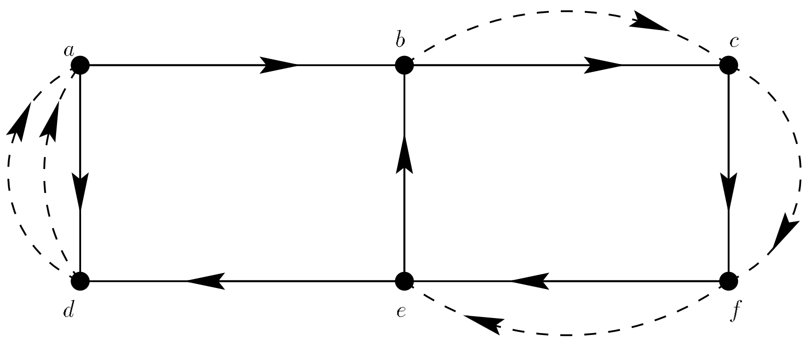


Figure 11.30: Multidigraph with an eulerian closed path (corresponding to (11.1)) obtained from the sweep subgraph of Figure 11.29 by adding the dashed arcs.

an arc can appear in the path several times. Even if it is a solid arc, we count the sweeping time only once. It is swept only the first time it is traversed. For each other appearance of the arc, we count the deadheading time.

If the sweep subgraph has an eulerian closed path, then, as in the "Chinese Postman" Problem, this path must be the minimal time street-sweeping route. In our example, Theorem 11.4 implies that there could be no such path. For there are a number of vertices in the sweep subgraph for which indegree is different from outdegree. Thus, a solution to the street-sweeping problem will be a closed path P of the curb multidigraph using some arcs not in the sweep subgraph. Suppose that we add to the sweep subgraph all the arcs used in P for deadheading, either because they are not in the sweep subgraph or because they have previously been swept in P. Add these arcs as many times as they are used. Adding these arcs gives rise to a multidigraph in which the path P is an eulerian closed path. For instance, one closed path in the curb multidigraph of Figure 11.29 that uses all arcs of the sweep subgraph is given by

$$f, e, b, c, f, e, d, a, d, a, b, c, f \tag{11.1}$$

(assume that the garage is at f). Path (11.1) corresponds to adding to the sweep subgraph the dashed arcs shown in Figure 11.30.

In general, we want the sum of the deadheading times on the added arcs to be minimized,[9] and we want to add deadheading arcs so that in the resulting multidigraph, there is an eulerian closed path; that is, every vertex has indegree equal to outdegree. Tucker and Bodin [1983] show how to solve the resulting combinatorial optimization problem by setting it up as a transportation problem of the type discussed in Section 13.4.1. The approach is similar to the approach used in the "Chinese Postman" Problem in Section 12.7.1. In our example, an optimal route

[9] We are omitting sources of delay other than deadheading, for example, delays associated with turns. These delays can be introduced by defining a system of penalties associated with different kinds of turns (see Tucker and Bodin [1983]).

is the path $f, e, d, e, b, a, b, a, d, e, b, c, f$. The total time required for this route is 72 for sweeping plus 20 for deadheading. The total time required for the route (11.1) is 72 for sweeping plus 48 for deadheading, which is considerably worse.

11.4.4 Finding Unknown RNA/DNA Chains

In this section we consider the problem of finding an unknown RNA or DNA string from a set of fragments (substrings) of the unknown string.

Example 11.1 Complete Digest by Enzymes In Section 2.12 we discussed the problem of finding an RNA chain given knowledge of its fragments after applying two enzymes, one of which breaks up the chain after each G link and the other of which breaks up the chain after each U or C link. Here, we show how to use the notion of eulerian closed path to find all RNA chains with given G and U, C fragments.

If there is only one G fragment or only one U, C fragment, the chain is determined. Hence, we shall assume that there are at least two G fragments and at least two U, C fragments.

We shall illustrate the procedure with the following G and U, C fragments:

G fragments: CCG, G, UCACG, AAAG, AA
U, C fragments: C, C, GGU, C, AC, GAAAGAA.

We first break down each fragment after each G, U, or C: for instance, breaking fragment GAAAGAA into G · AAAG · AA, GGU into G · G · U, and UCACG into U · C · AC · G. Each piece is called an *extended base*, and all extended bases in a fragment except the first and last are called *interior extended bases*. Using this further breakup of the fragments, we first observe how to find the beginning and end of the desired RNA chain. We make two lists, one giving all interior extended bases of all fragments from both digests, and one giving all fragments consisting of one extended base. In our example, we obtain the following lists:

Interior extended bases: C, C, AC, G, AAAG
Fragments consisting of one extended base: G, AAAG, AA, C, C, C, AC.

It is not hard to show that every entry on the first list is on the second list (Exercise 17). Moreover, the first and last extended bases in the entire chain are on the second list but not the first (Exercise 17). Since we are assuming that there are at least two G fragments and at least two U, C fragments, it is not hard to show that there will always be exactly two entries on the second list which are not on the first list (Exercise 17). One of these will be the first extended base of the RNA chain and one will be the last. In our example, these are AA and C. How do we tell which is last? We do it by observing that one of these entries will be from an *abnormal fragment*, that is, it will be the last extended base of a G fragment not ending in G or a U, C fragment not ending in U or C. In our example, AA is the last extended base of two abnormal fragments AA and GAAAGAA. This implies that the chain we are seeking begins in C and ends in AA.

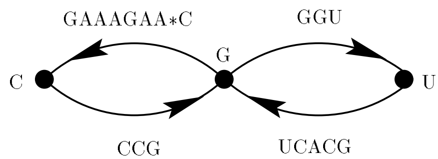


Figure 11.31: Multidigraph for reconstruction of RNA chains from complete enzyme digest.

To find all possible chains, we build a multidigraph as follows. First identify all normal fragments with more than one extended base. From each such fragment, use the first and last extended bases as vertices and draw an arc from the first to the last, labeling it with the corresponding fragment. (We shall add one more arc shortly.) Figure 11.31 illustrates this construction. For example, we have included an arc from U to G, labeled with the name of the corresponding fragment UCACG. Similarly, we add arcs from C to G and G to U. There might be several arcs from a given extended base to another, if there are several normal fragments with the same first and last extended base. Finally, we add one additional arc to our multidigraph. This is obtained by identifying the longest abnormal fragment—here this is GAAAGAA—and drawing an arc from the first (and perhaps only) extended base in this abnormal fragment to the first extended base in the chain. Here, we add an arc from G to C. We label this arc differently, by marking it $X * Y$, where X is the longest abnormal fragment, $*$ is a symbol marking this as a special arc, and Y is the first extended base in the chain. Hence, in our example, we label the arc from G to C by GAAAGAA$*$C. Every possible RNA chain with the given G and U, C fragments can now be identified from the multidigraph we have built. It turns out that each such chain corresponds to an eulerian closed path which ends with the special arc $X * Y$ (Exercise 20). In our example, the only such eulerian closed path goes from C to G to U to G to C. By using the corresponding arc labeling, we obtain the chain

CCGGUCACGAAAGAA.

It is easy to check that this chain has the desired G and U, C fragments. For more details on this graph-theoretic approach, see Hutchinson [1969]. ■

Example 11.2 Sequencing by Hybridization[10] A problem that arises in "sequencing" of DNA is the problem of determining a DNA string S from the list of all k-length substrings that appear in S. Known as the *sequencing by hybridization* or *SBH problem*, its solution again uses the notion of eulerian path (see Pevzner [1989]).

Consider the four bases A, C, G, T found in DNA. Let L be the list of all k-length substrings of the string S. [The data are obtained from a *DNA array* (*DNA chip*) with synthetic fragments ("probes") corresponding to all 4^k DNA sequences of length k. If a solution containing a fluorescently-labeled DNA fragment F of

[10] Our discussion follows Gusfield [1997] and Pevzner [2000].

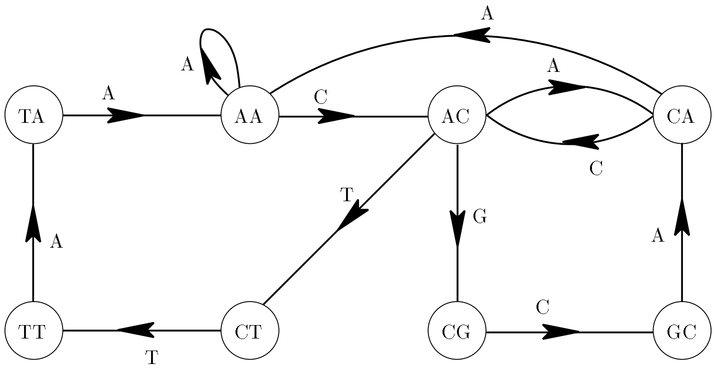


Figure 11.32: Digraph $D(L)$, where L is given by (11.2).

an unknown sequence is applied to the array, it hybridizes with those probes corresponding to substrings of length k in F, corresponding in the sense that A and T are reversed and G and C are reversed.] For example, if $S =$ ACACGCAACTTAAA and $k = 3$, then

$$L = \{\text{ACA}, \text{CAC}, \text{ACG}, \text{CGC}, \text{GCA}, \text{CAA}, \text{AAC}, \text{ACT}, \text{CTT}, \text{TTA}, \text{TAA}, \text{AAA}\}. \tag{11.2}$$

(We shall assume that no k-length substring appears more than once in S, as in this example. In practice, a value of $k \approx 10$ suffices for this purpose.) We build the digraph $D(L)$ as follows. Let each possible $(k-1)$-length DNA string be a vertex of $D(L)$. Thus, $|V(D(L))| = 4^{k-1}$. For each string $l = b_1 b_2 \cdots b_k$ in L, there is an associated arc a_l in $D(L)$. Arc a_l goes from the vertex representing the first $k-1$ bases of l to the vertex representing the last $k-1$ bases of l. So, string l corresponds to arc

$$a_l = (b_1 b_2 \cdots b_{k-1}, b_2 b_3 \cdots b_k)\,.$$

We also label arc a_l with the rightmost base b_k of l. Finally, we remove all isolated vertices of $D(L)$ (see Figure 11.32).

If P is a path in $D(L)$, there is an associated string S_P constructed from P in the following way: String S_P starts with the label of the first vertex in P and continues with the labels on the arcs of P. In Figure 11.32, for example, the path $P =$ AC, CG, GC, CA, AC, CA, AA, AA, AC, CT, TT, TA, AA yields the string $S_P =$ ACGCACAAACTTAA. The path P is said to *specify* the string S_P. The digraphs $D(L)$ are actually subgraphs of the de Bruijn diagrams presented in Section 11.4.6.

Given a set L of k-length substrings, a string S is called *compatible* with L if S contains every substring in L and S contains no other substrings of length k. The goal, then, is to find those strings (and hopefully, only one) compatible with a dataset L of k-length DNA substrings. The proof of the next theorem is left for Exercise 16.

Theorem 11.6 A string S is compatible with L iff S is specified by an eulerian path in $D(L)$.

It should be clear from the construction above that there is a one-to-one correspondence between strings that are compatible with L and eulerian paths in $D(L)$. So, assuming that each substring in L occurs exactly once in a DNA string S, L uniquely determines S iff $D(L)$ has a unique eulerian path. (van Aardenne-Ehrenfest and de Bruijn [1951] give a formula for the number of eulerian paths in a digraph D.)

Finishing with our example, notice that each vertex in the digraph $D(L)$ of Figure 11.32 has indegree equal to outdegree except for two vertices, AC and AA. For these vertices, outdegree(AC) $= 1 +$ indegree(AC), while indegree(AA) $= 1 +$ outdegree(AA). Therefore, $D(L)$ has an eulerian path.

Although the goal of DNA sequence reconstruction from k-length substrings seems to be attainable in theory, sequencing by hybridization is not as simple as we have described. It is common that errors are made in producing the substrings in L from an unknown string S. Also, our assumption of nonrepeated substrings in L is not very reasonable, due to the nature of DNA strings. For more on sequencing by hybridization, see Drmanac, *et al.* [1989], Gusfield [1997], Pevzner [2000], or Pevzner and Lipshutz [1994]. ∎

11.4.5 A Coding Application

Hutchinson and Wilf [1975] study codes on an alphabet of n letters, under the following assumption: All the information in a codeword is carried in the number of letters of each type and in the frequency of ordered pairs of letters, that is, the frequency with which one letter follows a second. For instance, Hutchinson and Wilf treat a DNA or RNA molecule as a word, with bases (not extended bases) as the letters, and make the assumption above about the genetic code. Specifically, Hutchinson and Wilf ask the following question: Given nonnegative integers $v_i, v_{ij}, i, j = 1, 2, \ldots, n$, is there a word from an alphabet of n letters so that the ith letter occurs exactly v_i times and so that the ith letter is followed by the jth exactly v_{ij} times? If so, what are all such words? We shall present Hutchinson and Wilf's solution, which uses the notion of eulerian path.

To give an example, suppose that $v_1 = 2, v_2 = v_3 = 1$ and v_{ij} is given by the following matrix:

$$(v_{ij}) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}. \tag{11.3}$$

Then one word that has the prescribed pattern is $ABCA$, if A corresponds to the first letter, B to the second, and C to the third. To give a second example, suppose that $v_1 = 2, v_2 = 4, v_3 = 3$, and

$$(v_{ij}) = \begin{pmatrix} 0 & 0 & 2 \\ 2 & 1 & 1 \\ 0 & 2 & 0 \end{pmatrix}. \tag{11.4}$$

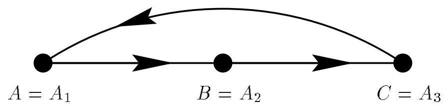


Figure 11.33: Multidigraph corresponding to (11.3).

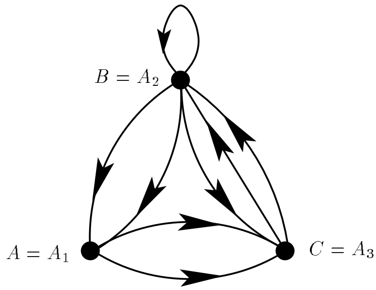


Figure 11.34: Multidigraph corresponding to (11.4).

One word that has the prescribed pattern is $BBCBACBAC$.

To analyze our problem, let us draw a multidigraph D with vertices the n letters $A_1, A_2, \ldots, A_n$, and with v_{ij} arcs from A_i to A_j. Loops are allowed. The multidigraphs corresponding to the matrices of (11.3) and (11.4) are shown in Figures 11.33 and 11.34, respectively. Let us suppose that $w = A_{i_1}, A_{i_2}, \ldots, A_{i_q}$ is a solution word. Then it is clear that w corresponds in the multidigraph D to an eulerian path that begins at A_{i_1} and ends at A_{i_q}. It is easy to see this for the two solution words we have given for our two examples. In what follows, we use the observation that a solution word corresponds to an eulerian path to learn more about solution words. The reader who wishes to may skip the rest of this subsection.

Since a solution word corresponds to an eulerian path, it follows that if there is a solution word, then D must be weakly connected up to isolated vertices. We consider first the case where $i_1 \neq i_q$. For every $i \neq i_1, i_q$, we have indegree at A_i equal to outdegree. For $i = i_1$, we have outdegree one higher than indegree, and for $i = i_q$ we have indegree one higher than outdegree. Thus, we have

$$\sum_{k=1}^{n} v_{ik} = \begin{cases} \sum_{k=1}^{n} v_{ki}, & i \neq i_1, i_q \\ \sum_{k=1}^{n} v_{ki} + 1, & i = i_1 \\ \sum_{k=1}^{n} v_{ki} - 1, & i = i_q. \end{cases} \tag{11.5}$$

This condition says that in the matrix (v_{ij}), the row sums equal the corresponding column sums, except in two places where they are off by one in the manner indicated. We also have a consistency condition, which relates the v_i to the v_{ij}:

$$v_i = \begin{cases} \sum_{k=1}^{n} v_{ki}, & i \neq i_1 \\ \sum_{k=1}^{n} v_{ki} + 1, & i = i_1. \end{cases} \tag{11.6}$$

It is easy to see, using Theorem 11.5, that if conditions (11.5) and (11.6) hold for some i_1 and $i_q, i_1 \neq i_q$, and if D is weakly connected up to isolated vertices, then there is a solution word, and every solution word corresponds to an eulerian path that begins in A_{i_1} and ends in A_{i_q}. In our second example, conditions (11.5) and (11.6) hold with $i_1 = 2, i_q = 3$. There are a number of eulerian paths from

$B = A_{i_1}$, to $C = A_{i_q}$, each giving rise to a solution word. A second example is $BACBBCBAC$.

What if the solution word begins and ends in the same letter, that is, if $i_1 = i_q$? Then there is an eulerian closed path, and we have

$$\sum_{k=1}^{n} v_{ik} = \sum_{k=1}^{n} v_{ki}, \quad i = 1, 2, \ldots, n. \tag{11.7}$$

Also, given i_1, (11.6) holds (for all i). Condition (11.7) says that in (v_{ij}), every row sum equals its corresponding column sum. Conversely, if (11.7) holds, and for i_1 (11.6) holds (for all i), and if D is weakly connected up to isolated vertices, then there is a solution and every solution word corresponds to an eulerian closed path in the multidigraph D, beginning and ending at A_{i_1}. This is the situation in our first example with $i_1 = 1$.

In sum, if there is a solution word, D is weakly connected up to isolated vertices and (11.6) holds for some i_1. Moreover, either (11.7) holds, or for i_1 and some i_q, $i_1 \neq i_q$, (11.5) holds. Conversely, suppose that D is weakly connected up to isolated vertices. If (11.6) holds for some i_1 and (11.5) holds for i_1 and some $i_q, i_1 \neq i_q$, there is a solution and all solution words correspond to eulerian paths beginning at A_{i_1} and ending at A_{i_q}. If (11.6) holds for some i_1 and (11.7) holds, there is a solution and all solution words correspond to eulerian closed paths beginning and ending at A_{i_1}.

11.4.6 De Bruijn Sequences and Telecommunications

In this subsection we consider another coding problem and its application in telecommunications. Let

$$\Sigma = \{0, 1, \ldots, p-1\}$$

be an alphabet of p letters and consider all words of length n from Σ. A (p, n) *de Bruijn sequence* (named after N. G. de Bruijn) is a sequence

$$a_0 a_1 \cdots a_{L-1} \tag{11.8}$$

with each a_i in Σ such that every word w of length n from Σ is realized as

$$a_i a_{i+1} \cdots a_{i+n-1} \tag{11.9}$$

for exactly one i, where addition in the subscripts of (11.9) is modulo L.[11] For instance,

$$01110100$$

is a $(2, 3)$ de Bruijn sequence over the alphabet $\Sigma = \{0, 1\}$ if $n = 3$. For, starting with the beginning, the three-letter words $a_i a_{i+1} a_{i+2}$ obtained are, respectively,

$$011, \quad 111, \quad 110, \quad 101, \quad 010, \quad 100, \quad 000, \quad 001.$$

[11]For a discussion of modular arithmetic, see Section 9.3.1.

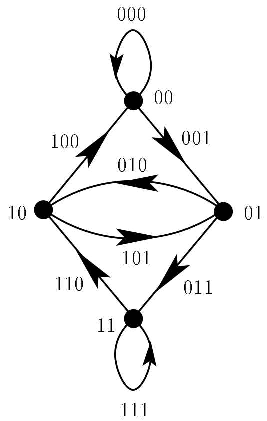


Figure 11.35: The digraph $D_{2,3}$.

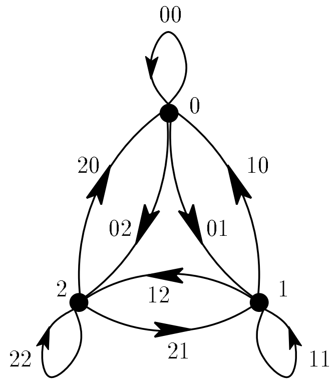


Figure 11.36: The digraph $D_{3,2}$.

The latter two are the words $a_6a_7a_0$ and $a_7a_0a_1$. De Bruijn sequences are of great significance in coding theory. They are implemented by shift registers and are sometimes called *shift register sequences.* For a detailed treatment of this topic, see Golomb [1982].

Clearly, if there is a (p, n) de Bruijn sequence (11.8), L must be p^n, where $p = |\Sigma|$. We shall show that for every positive p and n, there is a (p, n) de Bruijn sequence (11.8). Given p and n, build a digraph $D_{p,n}$, called a *de Bruijn diagram,* as follows. Let $V(D_{p,n})$ consist of all words of length $n-1$ from alphabet Σ, and include an arc from word $b_1b_2\cdots b_{n-1}$ to every word of the form $b_2b_3\cdots b_n$. Label such an arc with the word $b_1b_2\cdots b_n$. Figures 11.35 and 11.36 show the digraph $D_{p,n}$ for cases $p = 2, n = 3$ and $p = 3, n = 2$. Suppose that the de Bruijn diagram $D_{p,n}$ has an eulerian closed path. We then use successively the first letter from each arc in this path to obtain a de Bruijn sequence (11.8), where $L = p^n$. To see that this is a de Bruijn sequence, note that each word w of length n from Σ is realized. For if $w = b_1b_2\cdots b_n$, we know that the eulerian path covers the arc from $b_1b_2\cdots b_{n-1}$ to $b_2b_3\cdots b_n$. Thus, the eulerian path must go from $b_1b_2\cdots b_{n-1}$ to $b_2b_3\cdots b_n$. From there it must go to $b_3b_4\cdots$ to $b_4b_5\cdots$ to $b_n\cdots$. Thus, the first letters of the arcs in this path are $b_1b_2\cdots b_n$. It is easy to see, using Theorem 11.4, that the de Bruijn diagram of Figure 11.35 has an eulerian closed path, for this digraph is weakly connected and every vertex has indegree equal to outdegree. The de Bruijn sequence 01110100 corresponds to the eulerian closed path that goes from 01 to 11 to 11 to 10 to 01 to 10 to 00 to 00 to 01.

We shall prove the following theorem, which implies that for every pair of positive integers p and n, there is a (p, n) de Bruijn sequence. This theorem was discovered for the case $p = 2$ by de Bruijn [1946], and it was discovered for arbitrary p by Good [1946].

Theorem 11.7 For all positive integers p and n, the de Bruijn diagram $D_{p,n}$ has an eulerian closed path.

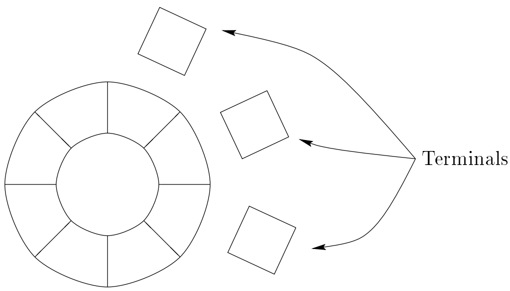


Figure 11.37: Rotating drum with eight sectors and three adjacent terminals.

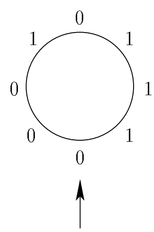


Figure 11.38: Arrangement of 0's and 1's that solves the rotating drum problem.

Proof. We first show that $D_{p,n}$ is weakly connected, indeed, strongly connected. Let $b_1b_2\cdots b_{n-1}$ and $c_1c_2\cdots c_{n-1}$ be any two vertices of $D_{p,n}$ Since we have a path

$$b_1b_2\cdots b_{n-1},\quad b_2b_3\cdots b_{n-1}c_1,\quad b_3b_4\cdots b_{n-1}c_1c_2,\quad \cdots,\quad c_1c_2\cdots c_{n-1},$$

$D_{p,n}$ is strongly connected. Next, note that every vertex has indegree and outdegree equal to p. The result follows by Theorem 11.4. Q.E.D.

We close this section by applying our results to a problem in telecommunications. We follow Liu [1968]. A rotating drum has eight different sectors. The question is: Can we tell the position of the drum without looking at it? One approach is by putting conducting material in some of the sectors and nonconducting material in other sectors. Place three terminals adjacent to the drum so that in any position of the drum, the terminals adjoin three consecutive sectors, as shown in Figure 11.37. A terminal will be activated if it adjoins a sector with conducting material. If we are clever, the pattern of conducting and nonconducting material will be so chosen that the pattern of activated and nonactivated terminals will tell us the position of the drum.

We can reformulate this as follows. Let each sector receive a 1 or a 0. We wish to arrange eight 0's and 1's around a circle so that every sequence of three consecutive digits is different. This is accomplished by finding a $(2,3)$ de Bruijn sequence, in particular the sequence 01110100. If we arrange these digits around a circle as shown in Figure 11.38, the following sequences of consecutive digits occur going counterclockwise beginning from the arrow: $011, 111, 110, 101, 010, 100, 000, 001$. These are all distinct, as desired. Thus, each position of the drum can be encoded uniquely.

Related problems concerned with teleprinting and cryptography are described in Exercises 22 and 23.

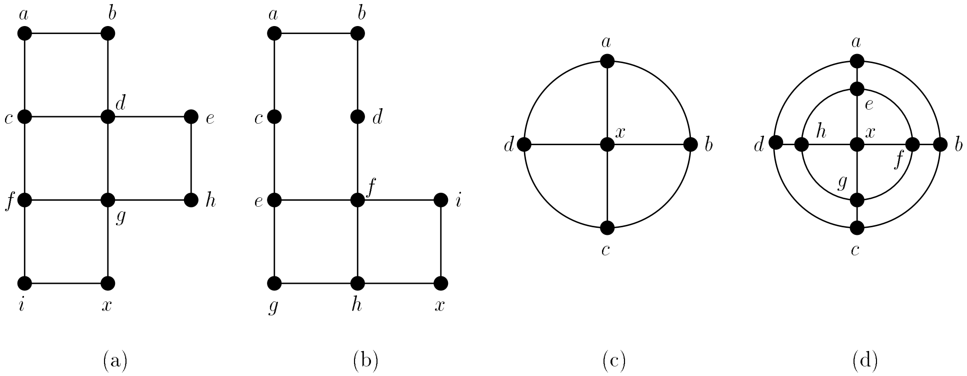


Figure 11.39: Graphs for Exercise 3 of Section 11.4.

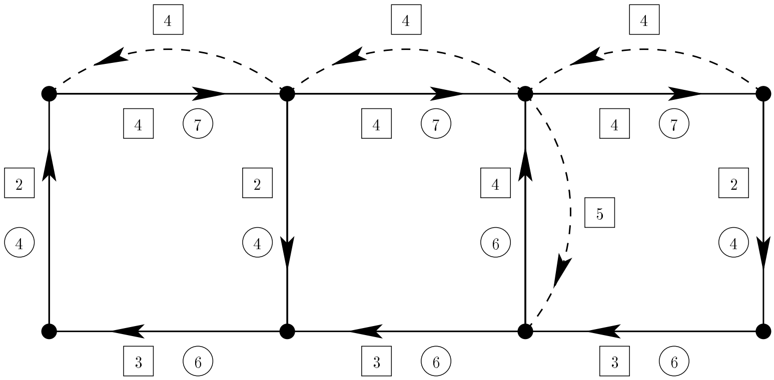


Figure 11.40: Curb multidigraph with sweep subgraph and sweeping and deadheading times shown as in Figure 11.29.

EXERCISES FOR SECTION 11.4

1. In graph G of Figure 11.28, find another way to add four copies of edges of G to obtain a multigraph with an eulerian closed chain.
2. Enumerate all eulerian paths for digraph $D(L)$ in Figure 11.32.
3. In each graph of Figure 11.39, find a shortest mail carrier's route from x to x by finding the smallest number of copies of edges of G that give rise to a multigraph with an eulerian closed chain.
4. Given the sweep subgraph and curb multidigraph of Figure 11.40, determine a set of deadheading arcs to add to the sweep subgraph to obtain a multidigraph with an eulerian closed path. Determine the total time required to traverse this path, including sweeping and deadheading.
5. Draw the de Bruijn diagram $D_{p,n}$ and find a (p, n) de Bruijn sequence for:

   (a) $p = 2, n = 4$ (b) $p = 3, n = 3$

6. For the G and U, C fragments given in the following exercises of Section 2.12, find an RNA chain using the method of Section 11.4.4.

   (a) Exercise 1 (b) Exercise 3 (c) Exercise 5

7. (a) Draw digraph $D(L)$ of Example 11.2 for the list $L = \{$CCGA, CGCG, CGTG, CGAA, GCCG, GCGT, GTGC, GAAC, TGCC, AACC, ACCC$\}$.

   (b) Find the number of eulerian paths in $D(L)$.

   (c) Find the compatible string(s) of L.

8. (a) Draw digraph $D(L)$ of Example 11.2 for the list $L = \{$AA, AC, AG, AT, CA, CT, GA, GT, TA, TC, TG, TT$\}$.

   (b) Find the number of eulerian paths in $D(L)$.

   (c) Is $S =$ ACTTCATGAGTAA compatible with L?

   (d) Is $S =$ AATGAGTACTTCA compatible with L?

   (e) Is $S =$ CTAGTACATGATA compatible with L?

9. In each of the following cases, determine if there is a codeword with the ith letter occurring exactly v_i times and the ith followed by the jth exactly v_{ij} times, and find such a word if there is one.

   (a) $v_1 = 3, v_2 = 4, v_3 = 5$,
$$(v_{ij}) = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 4 \end{pmatrix}.$$

   (b) $v_1 = 3, v_2 = 4, v_3 = 5$,
$$(v_{ij}) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 0 & 1 & 3 \end{pmatrix}.$$

   (c) $v_1 = 3, v_2 = 4, v_3 = 5$,
$$(v_{ij}) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{pmatrix}.$$

   (d) $v_1 = 3, v_2 = 3, v_3 = 3, v_4 = 3$,
$$(v_{ij}) = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

   (e) $v_1 = 3, v_2 = 3, v_3 = 2, v_4 = 3$ and (v_{ij}) as in part (d).

10. In the "Chinese Postman" Problem, show that if x and y are any two vertices of G, the shortest mail carrier's route from x to x has the same length as the shortest mail carrier's route from y to y.

11. Notice that in the second multigraph of Figure 11.28, the dashed edges can be divided into two chains, each joining two vertices of odd degree in G. These are the chains d, e, b and h, x, f.

(a) Can two such chains be found in the third multigraph of Figure 11.28?

(b) In general, if H is a multigraph obtained from G by using edges in a shortest mail carrier's route, show that the dashed edges in H can be grouped into several (not necessarily two) chains, each joining two vertices of odd degree in G. Moreover, every vertex of odd degree in G is an end vertex of exactly one of these chains. (This result will be important in discussing the last part of the solution to the "Chinese Postman" Problem in Section 12.7.1.)

12. In a shortest mail carrier's route, can any edge be used more than twice? Why?

13. Suppose that we have an optimal solution to the street-sweeping problem, that is, a closed path in the curb multidigraph which includes all arcs of the sweep subgraph and uses as little total time as possible. Is it possible that some arc is traversed more than twice?

14. Given a weakly connected digraph D, does there always exist a path that uses each arc of D at least once?

15. Prove that a digraph D is strongly connected if and only if D has a circuit that uses each arc of D at least once.

16. Prove Theorem 11.6.

17. Show the following about the list of interior extended bases of the G and U, C fragments and the list of fragments consisting of one extended base.

(a) Every entry on the first list is on the second list.

(b) The first and last extended bases in the entire RNA chain are on the second list but not on the first.

(c) If there are at least two G fragments and at least two U, C fragments, there will always be exactly two entries on the second list that are not on the first.

18. Under what circumstances does an RNA chain have two abnormal fragments?

19. Find the length of the smallest possible ambiguous RNA chain, that is, one whose G and U, C fragments are the same as that of another RNA chain.

20. This exercise sketches a proof[12] of the claim S: Suppose that we are given G and U, C fragments from an RNA chain with at least two G fragments and at least two U, C fragments. Then every RNA chain with given G and U, C fragments corresponds to an eulerian closed path that ends with the arc $X * Y$ in the multidigraph D constructed in Section 11.4.4. Let us say that an extended base has *type* G if it ends in G and *type* U, C if it ends in U or C. Note that any RNA chain can be written as

$$A_0 A_1 \cdots A_k A_{k+1} B,$$

where each A_i consists of extended bases of one type, the A_i alternate in type, and B is the last extended base if the chain ends in A and is the empty chain otherwise. For $i = 0, 1, \ldots, k$, let $\bar{A}_i$ be $A_i a_{i+1}$, where a_j is the first extended base in A_j. Let $\bar{A}_{k+1}$, be $A_{k+1}B$. Finally, say a fragment of one extended base is *trivial*. Show the following.

[12]The authors thank Michael Vivino for the idea behind this proof.

(a) $\bar{A}_0, \bar{A}_1, \ldots, \bar{A}_k$ are all nontrivial normal fragments and there are no other nontrivial normal fragments.

(b) $\bar{A}_{k+1}$ is the longest abnormal fragment.

(c) $a_0, a_1, \ldots, a_{k+1}$ are the vertices of the multidigraph D, $\bar{A}_i$ is the label on the arc from a_i to a_{i+1}, and $\bar{A}_{k+1} * a_0$ is the label on the arc from a_{k+1} to a_0.

(d) The arcs labeled $\bar{A}_0, \bar{A}_1, \ldots, \bar{A}_k, \bar{A}_{k+1} * a_0$ in this order define an eulerian closed path.

(e) Any other RNA chain that produces the same set of nontrivial normal fragments and the same longest abnormal fragment produces the same multidigraph D.

(f) Any RNA chain built from D produces the same set of nontrivial normal fragments and the same longest abnormal fragment.

(g) The nontrivial normal fragments and the longest abnormal fragment uniquely determine all of the fragments.

(h) Statement S holds.

21. In the problem discussed in Section 11.4.5, if there is a solution word, every eulerian path determines exactly one solution word. However, how many different eulerian paths are determined by each solution word?

22. The following problem arises in cryptography: Find a word from a given m-letter alphabet in which each arrangement of r letters appears exactly once. Find the solution to this problem if $r = 3$ and $m = 4$.

23. An important problem in communications known as the teleprinter's problem is the following: How long is the longest circular sequence of 0's and 1's such that no sequence of r consecutive bits appears more than once in the sequence? (The sequence of r bits is considered to appear if it starts near the end and finishes at the beginning.) Solve the teleprinter's problem. (It was first solved by Good [1946].)

11.5 HAMILTONIAN CHAINS AND PATHS

11.5.1 Definitions

Analogous to an eulerian chain or path in a graph or digraph is a *hamiltonian chain* or *path*, a chain or path that uses each vertex once and only once. The notion of hamiltonian chain goes back to Sir William Rowan Hamilton, who in 1857 described a puzzle that led to this concept (see below). In this section we discuss the question of existence of hamiltonian chains and paths. We discuss some applications here and others in Section 11.6.

Note that a hamiltonian chain or path is automatically simple, and hence by our conventions cannot be closed. However, we shall call a *circuit* or a *cycle* $u_1, u_2, \ldots, u_t, u_1$ *hamiltonian* if $u_1, u_2, \ldots, u_t$ is, respectively, a hamiltonian chain or path.

Although the notions of hamiltonian chain, path, and so on, are analogous to the comparable eulerian notions, it is very hard to tell if a graph or digraph has

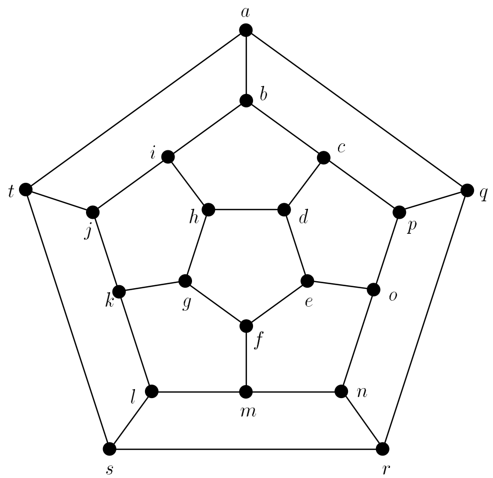


Figure 11.41: A graph representing the vertices and edges of a dodecahedron.

such an object. Indeed, it is an NP-complete problem to determine if a graph or digraph has a hamiltonian chain (path, circuit, cycle). Some results are known about existence of these objects, and we present some here. First we mention some applications of hamiltonian chains, paths, circuits, or cycles.

Example 11.3 Following the Edges of a Dodecahedron We begin with an example. Hamilton's puzzle was to determine a way to follow the edges of a dodecahedron that visited each corner exactly once and returned to the start. The vertices and edges of a dodecahedron can be drawn as a graph as in Figure 11.41. The problem becomes: Find a hamiltonian circuit in this graph. There is one: $a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, a$. ∎

Example 11.4 The Traveling Salesman Problem (Example 2.10 Revisited) The notion of hamiltonian cycle arises in the traveling salesman problem, which we discussed in Section 2.4. A salesman wishes to visit n different cities, each exactly once, and return to his starting point, in such a way as to minimize cost. (In Section 2.4 we mentioned other applications of the traveling salesman problem, for example to finding optimal routes for a bank courier or a robot in an automated warehouse.) Suppose that we let the cities be the vertices of a complete symmetric digraph, a digraph with all pairs of vertices joined by two arcs, and suppose that we put a weight c_{ij} on the arc (i, j) if c_{ij} is the cost of traveling from city i to city j. Since the complete symmetric digraph has a hamiltonian cycle, the existence question is not of interest. Rather, we seek a hamiltonian cycle in this digraph that has a minimum sum of weights. As we have pointed out previously, this traveling salesman problem is hard; that is, it is NP-complete. ∎

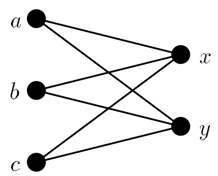


Figure 11.42: A graph without a hamiltonian circuit.

Example 11.5 Scheduling Industrial Processes A manufacturing plant has a single processing facility. A number of items are to be processed there. If item j is processed immediately after item i, there is a cost c_{ij} of resetting the processing facility from its configuration for processing item i to its configuration for processing item j. If no resetting is necessary, the cost is, of course, zero. Assuming that the costs of processing items are independent of the order in which items are processed, the problem is to choose the order so as to minimize the sum of the resetting costs c_{ij}. This problem arises in scheduling computer runs, as we observed in Example 2.17. It also arises in the chemical and pharmaceutical industries, where the processing facility might be a reaction vessel and resetting means cleaning. Costs are, of course, minimized if we can find a production schedule in which no resetting is necessary. To see if such a production schedule exists, we build a digraph D whose vertices are the items to be processed and which has an arc from i to j if j can follow i without resetting. If there is a hamiltonian path in D, such a production schedule exists. For more information on this problem, and for a treatment of the case where there is no hamiltonian path, see Christofides [1975]. See also the related discussion in Section 11.6.3. ■

11.5.2 Sufficient Conditions for the Existence of a Hamiltonian Circuit in a Graph

Not every graph has a hamiltonian circuit. Consider the graph of Figure 11.42. Note that every edge joins one of the vertices in $A = \{a, b, c\}$ to one of the vertices in $B = \{x, y\}$. Thus, a hamiltonian circuit would have to pass alternately from one of the vertices in A to one of the vertices in B. This could happen only if $|A| = |B|$.

In this section we present sufficient conditions for the existence of a hamiltonian circuit.

Theorem 11.8 (Ore [1960]) Suppose that G is a graph with $n \geq 3$ vertices and whenever vertices $x \neq y$ are not joined by an edge, the degree of x plus the degree of y is at least n. Then G has a hamiltonian circuit.

Proof.[13] Suppose that G has no hamiltonian circuit. We shall show that for some nonadjacent x, y in $V(G)$,

$$\deg_G(x) + \deg_G(y) < n, \tag{11.10}$$

[13] The proof may be omitted.

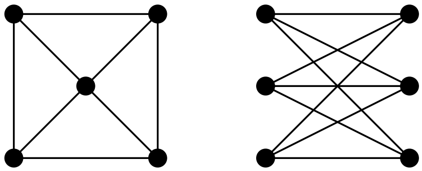

(a) (b)

Figure 11.43: Two graphs with hamiltonian circuits.

where $\deg_G(a)$ means degree of a in G. If we add edges to G, we eventually obtain a complete graph, which has a hamiltonian circuit. Thus, in the process of adding edges, we must eventually hit a graph H with the property that H has no hamiltonian circuit, but adding any edge to H gives us a graph with a hamiltonian circuit. We shall show that in H, there are nonadjacent x and y so that

$$\deg_H(x) + \deg_H(y) < n. \tag{11.11}$$

But $\deg_G(a) < \deg_H(a)$ for all a, so (11.11) implies (11.10).

Pick any nonadjacent x and y in H. Then H plus the edge $\{x, y\}$ has a hamiltonian circuit. Since H does not, this circuit must use the edge $\{x, y\}$. Hence, it can be written as

$$x, y, a_1, a_2, \ldots, a_{n-2}, x.$$

Now $V(H) = \{x, y, a_1, a_2, \ldots, a_{n-2}\}$. Moreover, we note that for $i > 1$,

$$\{y, a_i\} \in E(H) \Rightarrow \{x, a_{i-1}\} \notin E(H). \tag{11.12}$$

For if not, then

$$y, a_i, a_{i+1}, \ldots, a_{n-2}, x, a_{i-1}, a_{i-2}, \ldots, a_1, y$$

is a hamiltonian circuit in H, which is a contradiction. Now (11.12) and $\{x, y\} \notin E(H)$ imply (11.11). Q.E.D.

To illustrate Ore's Theorem, note that the graph (a) of Figure 11.43 has a hamiltonian circuit, for any two vertices have a sum of degrees at least 5. Note that Ore's Theorem does not give necessary conditions. For consider the circuit of length 5, Z_5. There is a hamiltonian circuit, but any two vertices have a sum of degrees equal to 4, which is less than n.

The next result, originally proved independently, follows immediately from Theorem 11.8.

Corollary 11.8.1 (Dirac [1952]) Suppose that G is a graph with $n \geq 3$ vertices and each vertex has degree at least $n/2$. Then G has a hamiltonian circuit.

To illustrate Corollary 11.8.1, note that the graph (b) of Figure 11.43 has a hamiltonian circuit because every vertex has degree 3 and there are six vertices.

Corollary 11.8.2 (Bondy and Chvátal [1976]) Suppose that G is a graph with $n \geq 3$ vertices and that x and y are nonadjacent vertices in G so that

$$\deg(x) + \deg(y) \geq n.$$

Then G has a hamiltonian circuit if and only if G plus the edge $\{x, y\}$ has a hamiltonian circuit.

Proof. If G has a hamiltonian circuit, then certainly G plus edge $\{x, y\}$ does. The converse follows from the proof of Theorem 11.8. Q.E.D.

Let us see what happens if we try to repeat the construction in Corollary 11.8.2. That is, we start with a graph $G = G_1$. We find a pair of nonadjacent vertices x_1 and y_1 in G_1 so that in G_1, the degrees of x_1 and y_1 sum to at least n. We let G_2 be G_1 plus edge $\{x_1, y_1\}$. We now find a pair of nonadjacent vertices x_2 and y_2 in G_2 so that in G_2, the degrees of x_2 and y_2 sum to at least n. We let G_3 be G_2 plus edge $\{x_2, y_2\}$. We continue this procedure until we obtain a graph $H = G_i$ with no nonadjacent x_i and y_i whose degrees in G_i sum to at least n. It is not hard to show (Exercise 17) that no matter in what order we perform this construction, we always obtain the same graph H. We call this graph H the *closure* of G, and denote it by $c(G)$. We illustrate the construction of $c(G)$ in Figure 11.44. The next result follows from Corollary 11.8.2.

Corollary 11.8.3 (Bondy and Chvátal [1976]) G has a hamiltonian circuit if and only if $c(G)$ has a hamiltonian circuit.

Note that in Figure 11.44, $c(G)$ is a complete graph in parts (a) and (c) but not in part (b). Parts (a) and (c) illustrate the following theorem. [Note that part (c) could not be handled by Ore's Theorem.]

Theorem 11.9 (Bondy and Chvátal [1976]) Suppose that G is a graph with at least three vertices. If $c(G)$ is complete, G has a hamiltonian circuit.

Proof. A complete graph with at least three vertices has a hamiltonian circuit. Q.E.D.

Note that Ore's Theorem is a corollary of Theorem 11.9.

We have given conditions sufficient for the existence of a hamiltonian circuit but not for a hamiltonian chain. For some conditions sufficient for the latter, see, for example, the text and exercises of Chartrand and Lesniak [1996].

11.5.3 Sufficient Conditions for the Existence of a Hamiltonian Cycle in a Digraph

Ore's and Dirac's results (Theorem 11.8 and Corollary 11.8.1) have the following analogues for digraphs. Here, $\text{od}(u)$ is the outdegree of u and $\text{id}(u)$ is the indegree of u.

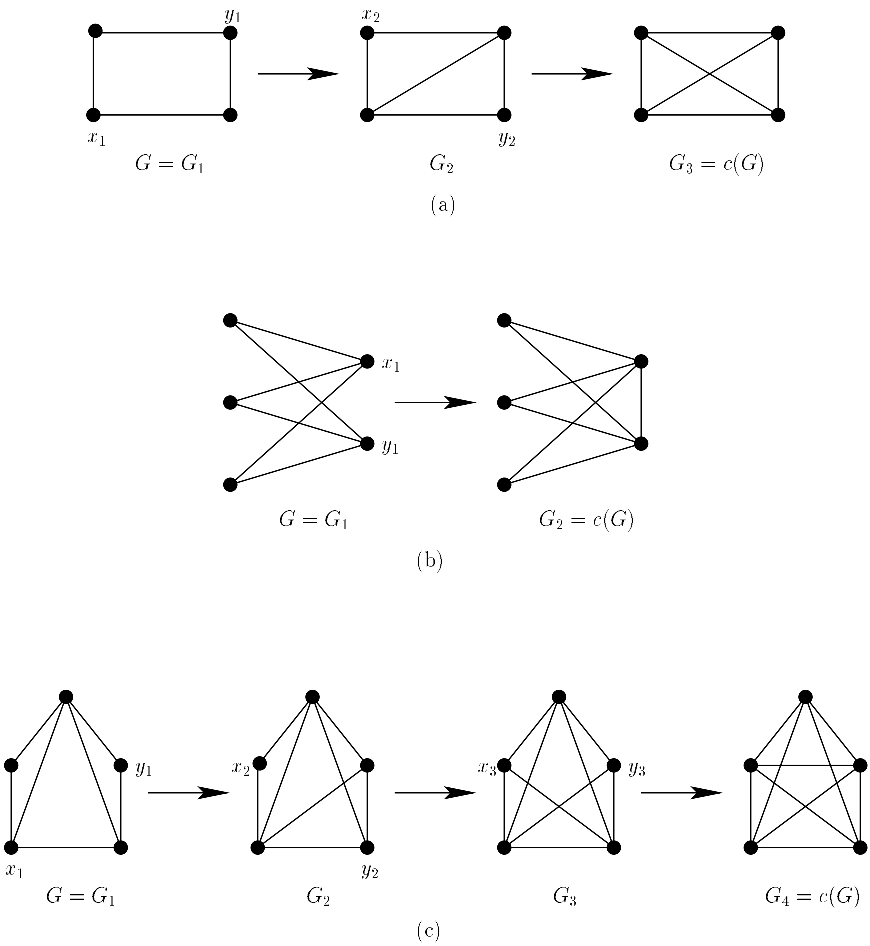
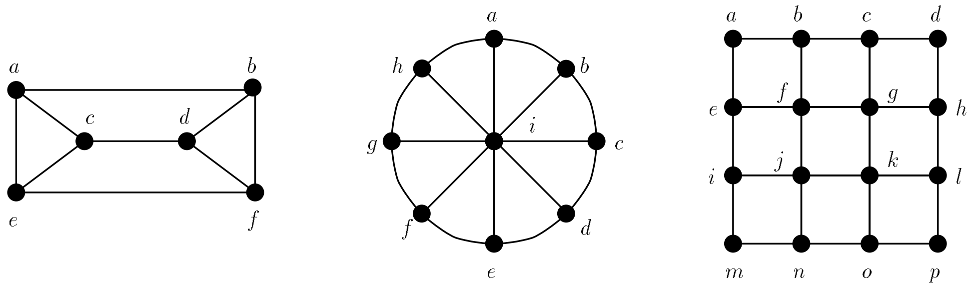


Figure 11.44: Three constructions of the closure.



(a) (b) (c)

Figure 11.45: Graphs for exercises of Section 11.5.

Theorem 11.10 (Woodall [1972]) Suppose that D is a digraph with $n \geq 3$ vertices and whenever $x \neq y$ and there is no arc from x to y, then

$$\mathrm{od}(x) + \mathrm{id}(y) \geq n. \tag{11.13}$$

Then D has a hamiltonian cycle.

Theorem 11.11 (Ghouila-Houri [1960]) Suppose that D is a strongly connected digraph with n vertices and for every vertex x,

$$\mathrm{od}(x) + \mathrm{id}(x) \geq n. \tag{11.14}$$

Then D has a hamiltonian cycle.

Corollary 11.11.1 Suppose that D is a digraph with n vertices and for every vertex x,

$$\mathrm{od}(x) \geq \frac{n}{2} \quad \text{and} \quad \mathrm{id}(x) \geq \frac{n}{2}. \tag{11.15}$$

Then D has a hamiltonian cycle.

Proof. One shows that (11.15) implies that D is strongly connected. The proof is left to the reader (Exercise 18). Q.E.D.

We have given conditions for the existence of a hamiltonian cycle, but not for a hamiltonian path. For a summary of conditions sufficient for the latter, see, for example, Chartrand and Lesniak [1996].

EXERCISES FOR SECTION 11.5

1. In each graph of Figure 11.45, find a hamiltonian circuit.
2. In each graph of Figure 11.46, find a hamiltonian chain.

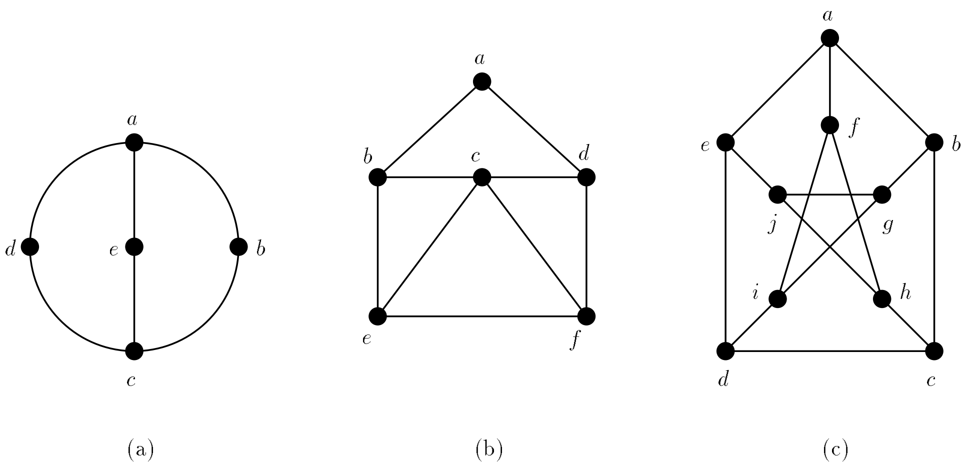


Figure 11.46: Graphs for exercises of Section 11.5.

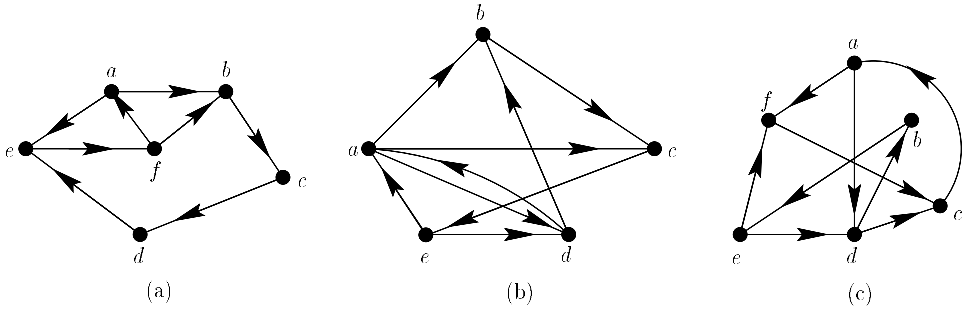


Figure 11.47: Digraphs for exercises of Section 11.5.

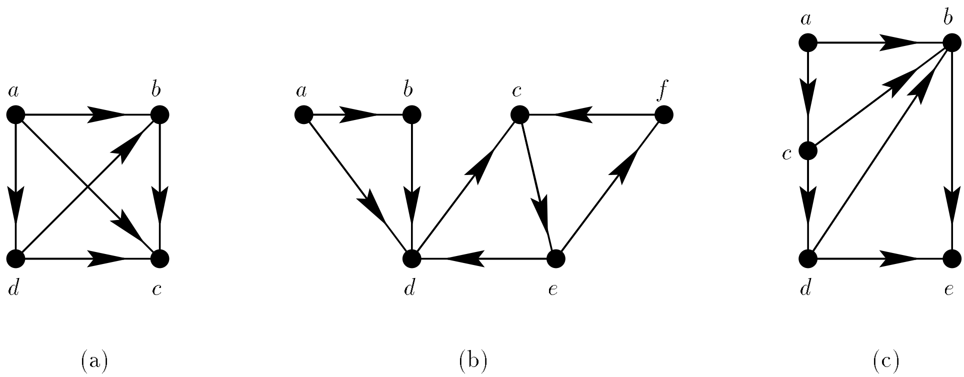


Figure 11.48: Digraphs for exercises of Section 11.5.

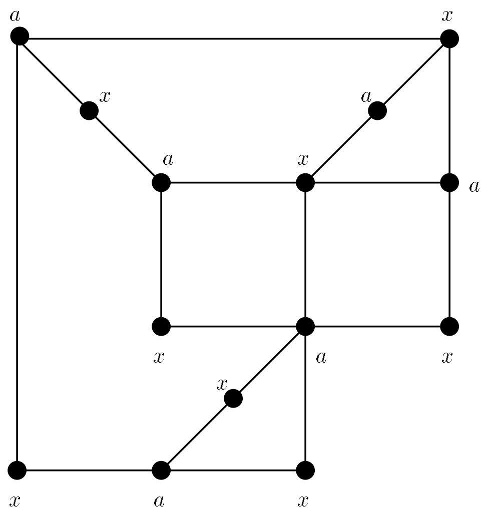


Figure 11.49: A graph with no hamiltonian circuit.

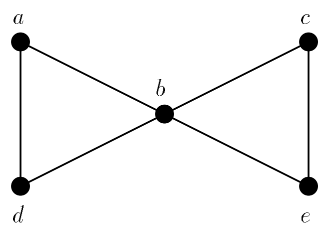


Figure 11.50: Graph for Exercise 6, Section 11.5.

3. In each digraph of Figure 11.47, find a hamiltonian cycle.
4. In each digraph of Figure 11.48, find a hamiltonian path.
5. Show that the graph of Figure 11.49 can have no hamiltonian circuit. (*Hint:* The a's and x's should tell you something.)
6. Can the graph of Figure 11.50 have a hamiltonian circuit? Why?
7. Give examples of graphs that:

   (a) Have both eulerian and hamiltonian circuits

   (b) Have a hamiltonian but not an eulerian circuit

   (c) Have an eulerian but not a hamiltonian circuit

   (d) Have neither an eulerian nor a hamiltonian circuit

8. Give an example of:

   (a) A graph with no hamiltonian chain

   (b) A digraph with no hamiltonian path

9. Suppose that G is a graph in which there are 10 vertices, a, b, c, d, e and x, y, u, v, w, and each of the first five vertices is joined to each of the last five.

   (a) Does Ore's Theorem (Theorem 11.8) imply that G has a hamiltonian circuit? Why?

   (b) Does Dirac's Theorem (Corollary 11.8.1) imply that G has a hamiltonian circuit? Why?

   (c) Does G have a hamiltonian circuit?

10. (a) Find the closure $c(G)$ of each graph of Figures 11.45 and 11.46.
    (b) For which graphs in Figures 11.45 and 11.46 does Theorem 11.9 imply that there is a hamiltonian circuit?
11. Give an example of a graph whose closure does not have a hamiltonian circuit.
12. Show that if G has a hamiltonian circuit, $c(G)$ is not necessarily complete.
13. For which digraphs of Figures 11.47 and 11.48 does Theorem 11.10 imply that there is a hamiltonian cycle?
14. For which digraphs of Figures 11.47 and 11.48 does Theorem 11.11 imply that there is a hamiltonian cycle?
15. A graph is *regular of degree* k if every vertex has the same degree, k. Show that G has a hamiltonian circuit if
    (a) G has 11 vertices and is regular of degree 6;
    (b) G has 13 vertices and is regular of degree 6.
16. Suppose that $n \geq 4$ and that of n people, any two of them together know all the remaining $n-2$. Show that the people can be seated around a circular table so that everyone is seated between two friends.
17. Suppose that by successively adding edges joining nonadjacent vertices whose degrees sum to at least n, we eventually obtain a graph H and then are unable to continue. Suppose that by doing the construction in a different order, we eventually obtain a graph K and then are unable to continue. Show that $H = K$. (*Hint:* If not, find the first edge added to G in the first construction which is not an edge of K.)
18. Show that Equation (11.15) implies that D is strongly connected.
19. Suppose that the hypotheses of Theorem 11.10 hold with n in Equation (11.13) replaced by $n-1$. Use Theorem 11.10 to show that D has a hamiltonian path.
20. Show that Corollary 11.8.1 is false if "each vertex has degree at least $n/2$" is replaced by "each vertex has degree at least $(n-1)/2$."
21. Suppose that the resetting costs as in Example 11.5 are given in the following matrix C, where c_{ij} is the cost of resetting the production facility from its configuration for processing item i to its configuration for processing item j.

$$C = \begin{pmatrix} 0 & 0 & 0 & 6 & 5 & 5 & 3 & 8 & 4 & 9 \\ 6 & 0 & 1 & 7 & 3 & 9 & 8 & 0 & 6 & 8 \\ 11 & 4 & 0 & 8 & 0 & 7 & 10 & 5 & 3 & 5 \\ 8 & 8 & 0 & 0 & 11 & 8 & 7 & 3 & 9 & 8 \\ 4 & 4 & 7 & 4 & 0 & 0 & 0 & 11 & 10 & 8 \\ 6 & 5 & 6 & 2 & 0 & 0 & 8 & 8 & 2 & 5 \\ 6 & 7 & 6 & 5 & 3 & 11 & 0 & 9 & 0 & 15 \\ 4 & 3 & 9 & 8 & 8 & 7 & 0 & 0 & 3 & 5 \\ 0 & 3 & 4 & 11 & 10 & 6 & 8 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 7 & 6 & 5 & 3 & 1 & 0 \end{pmatrix}.$$

Is there a production schedule in which no resetting costs are incurred?

22. (Pósa [1962]). Use Theorem 11.9 to show that if G is a graph of at least three vertices and if for every integer j with $1 \leq j < n/2$, the number of vertices of degree not exceeding j is less than j, then G has a hamiltonian circuit.

23. Prove or disprove the following variant on Ore's Theorem (Theorem 11.8). Suppose that G is a graph with $n \geq 3$ vertices and

$$\frac{\sum (\deg_G(x) + \deg_G(y))}{q} \geq n,$$

where the sum is taken over all nonadjacent pairs x and y and q equals the number of such x, y pairs; then G is hamiltonian.

11.6 APPLICATIONS OF HAMILTONIAN CHAINS AND PATHS

In this section we present several applications of the ideas of hamiltonian chains and paths. Sections 11.6.2 and 11.6.3 depend in small ways on Section 11.6.1. Otherwise, these sections are independent and can be read in any order. In particular, a quick reading could include just a glance at Section 11.6.1, followed by Section 11.6.3, or it could consist of just Section 11.6.4. Section 11.6.5 should be read as a natural follow-up to Section 11.4.4.

11.6.1 Tournaments

Let (V, A) be a digraph and assume that for all $u \neq v$ in V, (u, v) is in A or (v, u) is in A, but not both. Such a digraph is called a *tournament*. Tournaments arise in many different situations. There are the obvious ones, the round-robin competitions in tennis, basketball, and so on.[14] In a round-robin competition, every pair of players (pair of teams) competes, and one and only one member of each pair beats the other. (We assume that no ties are allowed.) Tournaments also arise from *pair comparison experiments*, where a subject or a decisionmaker is presented with each pair of alternatives from a set and is asked to say which of the two he or she prefers, which is more important, which is more qualified, and so on. (The reader might wish to consider the discussion of preference in Chapter 4, in Example 4.1 and elsewhere, for comparison.) Tournaments also arise in nature, where certain individuals in a given species develop dominance over others of the same species. In such situations, for every pair of animals of the species, one and only one is dominant over the other. The dominance relation defines what is called a *pecking order* among the individuals concerned.

Sometimes we want to *rank* the players of a tournament. This might be true if we are giving out second prize, third prize, and so on. It might also be true, for example, if the "players" are alternative candidates for a job and the tournament represents preferences among candidates. Then our first-choice job candidate might

[14] The reader should distinguish a round-robin competition from the elimination-type competition, which is more common.

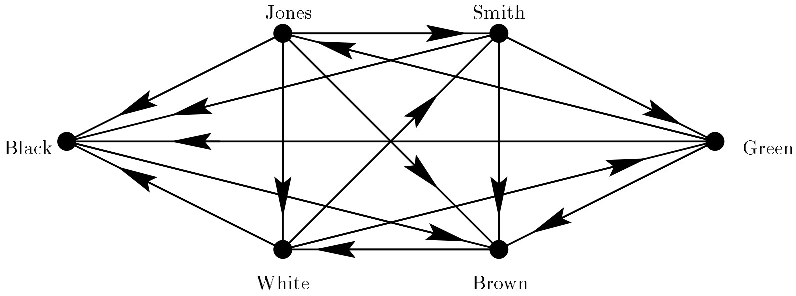


Figure 11.51: A round-robin Ping-Pong tournament.

not accept and we might want to have a second choice, third choice, and so on, chosen ahead of time. One way to find a ranking of the players is to observe that every tournament has a hamiltonian path. This result, which we shall prove shortly, implies that we can label the n players as $u_1, u_2, \ldots, u_n$ in such a way that u_1 beats u_2, u_2 beats u_3, ..., and u_{n-1} beats u_n. Such a labeling gives us a ranking of the players: u_1 is ranked first, u_2 second, and so on. To illustrate, consider the tournament of Figure 11.51. Here, a hamiltonian path and hence a ranking is given by: Jones, Smith, Green, Brown, White, Black. Another way to find a ranking of the players is to use the score sequence defined in Exercise 20 (see also Exercises 23 and 28).

Theorem 11.12 (Rédei [1934]) Every tournament (V, A) has a hamiltonian path.

Proof.[15] We proceed by induction on the number n of vertices. The result is trivial if $n = 2$. Assuming the result for tournaments of n vertices, let us consider a tournament (V, A) with $n + 1$ vertices. Let u be an arbitrary vertex and consider the subgraph generated by $V - \{u\}$. It is easy to verify that this subgraph is still a tournament. Hence, by inductive assumption, it has a hamiltonian path $u_1, u_2, \ldots, u_n$. If there is an arc from u to u_1, then $u, u_1, u_2, \ldots, u_n$ is a hamiltonian path of (V, A). If there is no arc from u to u_1, then, since (V, A) is a tournament, there is an arc from u_1 to u. Let i be the largest integer such that there is an arc from u_i to u. If i is n, there is an arc from u_n to u, and we conclude that $u_1, u_2, \ldots, u_n, u$ is a hamiltonian path of (V, A). If $i < n$, there is an arc from u_i to u, and moreover, by definition of i, there is no arc from u_{i+1} to u. Since (V, A) is a tournament, there is an arc from u to u_{i+1}. Thus, $u_1, u_2, \ldots, u_i, u, u_{i+1}, \ldots, u_n$ is a hamiltonian path of (V, A). Q.E.D.

If $u_1, u_2, \ldots, u_n$ is a hamiltonian path in a tournament (V, A), then as we have already observed, we can use it to define a ranking of the players. Unfortunately, there can be other hamiltonian paths in the tournament. In our example, Smith, Brown, White, Green, Jones, Black is another. In this situation, then, there are

[15] The proof may be omitted.

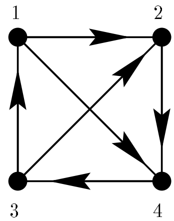


Figure 11.52: For each vertex a and rank r, there is some hamiltonian path for which a has rank r.

many possible rankings of the players. The situation can get even worse. In the tournament of Figure 11.52, in fact, for each vertex a and each possible rank r, there is some hamiltonian path for which a has rank r.

In general, given a set of possible rankings, we might want to try to choose a "consensus" ranking. The problem of finding a consensus among alternative possible rankings is a rather difficult one, that deserves a longer discussion than we can give it here (see Roberts [1976, Ch. 7]).

We next ask whether there are any circumstances where the problem of having many different rankings does not occur. That is, are there circumstances when a tournament has a unique hamiltonian path? The answer is given by saying that a tournament is *transitive* if whenever (u, v) is an arc and (v, w) is an arc, then (u, w) is an arc.[16] The tournament of Figure 11.51 is not transitive, since, for example, Jones beats Smith and Smith beats Green, but Jones does not beat Green.

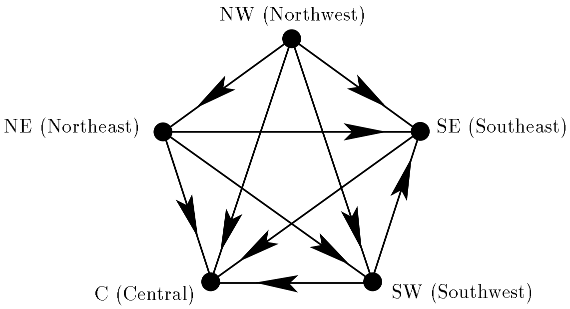
Theorem 11.13 A tournament has a unique hamiltonian path if and only if the tournament is transitive.

Proof. See Exercise 27. Q.E.D.

At this point, let us consider applications of Theorem 11.13 to a decisionmaker's preferences among alternatives, or to his or her ratings of relative importance of alternatives, and so on. The results are illustrated by pair comparison data for preference: for example, that shown in Figure 11.53. Here, transitivity holds, so there is a unique hamiltonian path. The path is Northwest, Northeast, Southwest, Southeast, Central.

In studying preference, it is often reasonable to assume (or demand) that the decisionmaker's preferences define a transitive tournament, that is, that if he or she prefers u to v and v to w, then he or she prefers u to w. This turns out to be equivalent to assuming that the decisionmaker can uniquely rank the alternatives among which he or she is expressing preferences. The reader might wish to compare the discussion of preference in Example 4.1 and of various order relations in Section 4.2. In the language of that section, a tournament is a strict linear order (see Exercise 6).

[16]Transitivity played an important role in Chapter 4. Sometimes, transitivity is defined differently for digraphs, requiring that the existence of arcs (u, v) and (v, w) implies the existence of arc (u, w) only if $u \neq w$. The condition $u \neq w$ holds automatically for tournaments if (u, v) and (v, w) are arcs. Why?



$$\begin{array}{c} \\ \text{NW} \\ \text{SW} \\ \text{C} \\ \text{SE} \\ \text{NE} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{NW} & \text{SW} & \text{C} & \text{SE} & \text{NE} \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

Figure 11.53: Results of a pair comparison experiment for preference among geographical areas. The x, y entry in the matrix is 1 iff x is preferred to y. In the corresponding digraph, an arc from x to y indicates that x is preferred to y.

Performing a pair comparison experiment to elicit preferences can be a tedious job. If there are n alternatives, $\binom{n}{2} = n(n-1)/2$ comparisons are required, and this can get to be a large number even for moderate n. However, if we believe that a subject is transitive, we know that a pair comparison experiment amounts to a transitive tournament and hence to a unique ranking of the alternatives. Thus, we are trying to order a set of n items. How many comparisons are really required to recover this ranking and hence the original tournament? This is the problem of sorting that we discussed in Section 3.6. There, we pointed out that good sorting algorithms can take on the order of $n \log_2 n$ steps, which is a smaller number than $n(n-1)/2$.

11.6.2 Topological Sorting

It turns out that a tournament is transitive if and only if it is *acyclic*, that is, has no cycles (Exercise 26). Finding the unique hamiltonian path in a transitive or acyclic tournament is a special case of the following more general problem. Suppose that we are given a digraph D of n vertices. Label the vertices of D with the integers $1, 2, \ldots, n$ so that every arc of D leads from a vertex with a smaller label to a vertex with a larger label. Such a labeling is called a *topological order* for D, and the process of finding such an order is called *topological sorting.*

Theorem 11.14 A digraph D has a topological order if and only if D is acyclic.

Proof. Clearly, if there is a topological order $1, 2, \ldots, n$, then D could not have a cycle. Conversely, suppose that D is acyclic. By Exercise 31, there is a vertex x_1 with no incoming arcs. Label x_1 with the label 1, and delete it from D. Now the resulting digraph is still acyclic and hence has a vertex x_2 with no incoming arcs. Label x_2 with label 2; and so on. This clearly gives rise to a topological order.

Q.E.D.

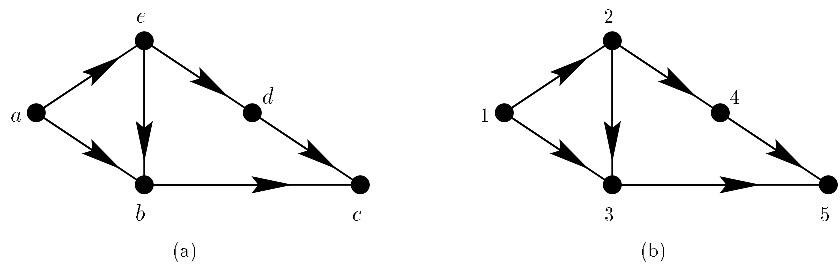


Figure 11.54: Part (b) shows a topological order for the digraph of part (a).

We illustrate the construction in the proof of Theorem 11.14 by finding a topological order in the digraph D shown in Figure 11.54(a). We first choose vertex a. After eliminating a, we choose vertex e. After eliminating e, we have a choice of either b or d; say, for concreteness, that we choose b. Then after eliminating b, we choose d. Finally, we are left with c. This gives the labeling shown in Figure 11.54(b). One of the problems with the procedure described is that at each stage, we must search through the entire remaining digraph for the next vertex to choose. For a description of a more efficient algorithm, based on the depth-first search procedure described in Section 11.1, see Cormen, Leiserson, and Rivest [1999], Golumbic [1980], or Reingold, Nievergelt, and Deo [1977].

Topological sorting has a variety of applications. For instance, it arises in the analysis of *activity networks* in project planning (Deo [1974]). A large project is divided into individual tasks called *activities*. Some activities must be completed before others can be started—for example, an item must be sanded before it can be painted. Build a digraph D as follows. The vertices are the activities, and there is an arc from activity x to activity y if x must precede y. We seek to find an order in which to perform the activities so that if there is an arc from x to y in D, then x comes before y. This requires a topological sorting of D. It can be done if and only if D has no cycles. A similar situation arises, for example, if we wish to arrange words in a glossary and guarantee that no word is used before it is defined. This, too, requires a topological sorting.

11.6.3 Scheduling Problems in Operations Research[17]

Many scheduling problems in operations research, for instance those involving activity networks discussed in Section 11.6.2, involve finding an order in which to perform a certain number of operations. We often seek an optimal order. Sometimes such problems can be solved by finding hamiltonian paths. We have already seen an example of this in Example 11.5. Here we present another example. Related problems are discussed in Section 13.2.3.

Suppose that a printer has n different books to publish. He has two machines, a printing machine and a binding machine. A book must be printed before it can be

[17] This subsection is based on Berge [1962], Johnson [1954], and Liu [1972].

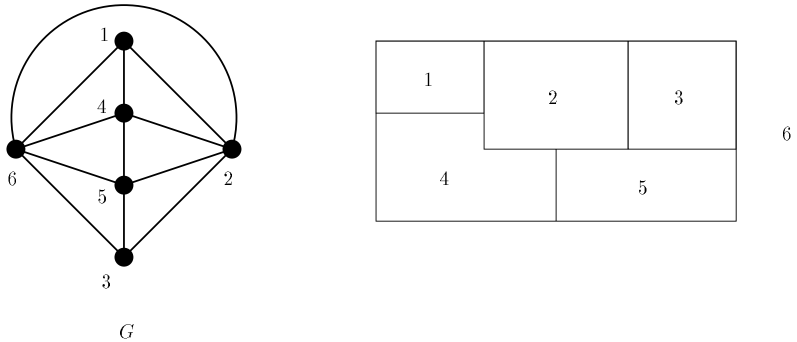


Figure 11.55: A relationship graph G and a corresponding layout plan.

bound. The binding machine operator earns much more money than the printing machine operator, and must be paid from the time the binding machine is first started until all the books have been bound. In what order should the books be printed so that the total pay of the binding machine operator is minimized?

Let p_k be the time required to print the kth book and b_k the time required to bind the kth book. Let us make the special assumption that for all i and j, either $p_i \leq b_j$ or $p_j \leq b_i$. Note that it is now possible to find an ordering of books so that if books are printed and bound in that order, the binding machine will be kept busy without idle time after the first book is printed. This clearly will minimize the pay of the binding machine operator. To find the desired ordering, draw a digraph with an arc from i to j if and only if $b_i \geq p_j$. Then this digraph contains a tournament and so, by Theorem 11.12, has a hamiltonian path. This path provides the desired ordering. More general treatment of this problem, without the special assumption, can be found in Johnson [1954] (see also Exercise 38).

11.6.4 Facilities Design[18]

Graph theory is finding a variety of applications in the design of such physical facilities as manufacturing plants, hospitals, schools, golf courses, and so on. In such design problems, we have a number of areas that need to be located and it is desired that certain of them be next to each other. We draw a graph G, the *relationship graph*, whose vertices are the areas in question and that has an edge between two areas if they should be next to each other. If G is planar, it comes from a map (see Example 3.19), and the corresponding map represents a layout plan where two areas (countries) joined by an edge in G in fact share a common wall (boundary). Figure 11.55 shows a planar graph and the corresponding facilities layout.

If the relationship graph G is not planar, we seek to eliminate some requirements, i.e., eliminate some edges of G, to find a spanning subgraph G' of G that is planar.

[18] This subsection is based on Chachra, Ghare, and Moore [1979].

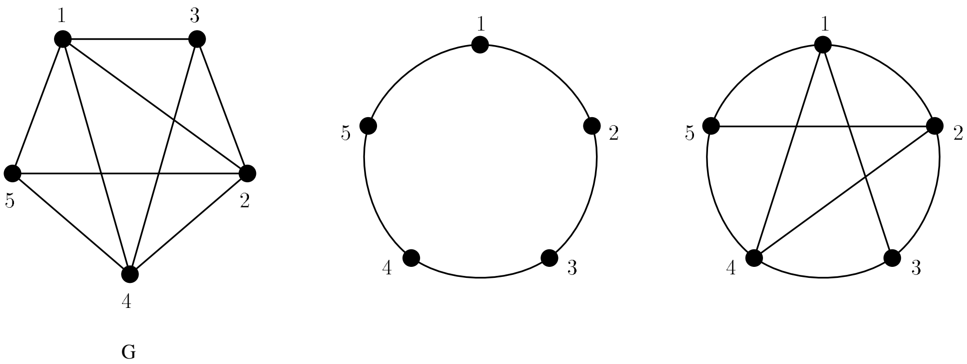


Figure 11.56: A relationship graph G, the vertices of a hamiltonian circuit arranged around a circle, and the remaining edges of G represented as chords of the circle.

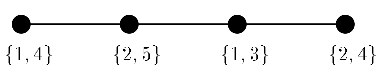


Figure 11.57: Graph H obtained from Figure 11.56.

Then G' can be used to build the layout plan. One method for finding G' that is used in facilities planning is the following (due to Demourcron, Malgrance, and Pertuiset [1964]). Determine if G has a hamiltonian circuit. If so, find such a circuit $C = u_1, u_2, \ldots, u_n$. Locate the vertices u_i around a circle in this order. For instance, consider graph G of Figure 11.56. Then $C = 1, 2, 3, 4, 5$ is a hamiltonian circuit and it is represented around a circle in Figure 11.56. Build a new graph H as follows. The vertices of H are the edges in G which are not edges of C. Two edges of H are adjacent if and only if when the corresponding chords are drawn in the circle determined by C, these chords intersect. See Figure 11.56 for the chords in our example, and see Figure 11.57 for the graph H. Suppose that H is 2-colorable, say using the colors red and blue. Then G is planar. To see this, simply note that all red chords can be drawn in the interior of the circle determined by C and all blue chords in the exterior, with no edges crossing. For instance, in graph H of Figure 11.57, we can color vertices $\{1, 3\}$ and $\{1, 4\}$ red and vertices $\{2, 4\}$ and $\{2, 5\}$ blue. Then we obtain the planar drawing of G shown in Figure 11.58. If H is not 2-colorable, we seek a maximal subgraph K of H that is 2-colorable. The edges of K can be added to C to get a planar graph G' which is a spanning subgraph of G. G' is used for finding a layout plan. For instance, suppose that G is obtained from G of Figure 11.56 by adding edge $\{3, 5\}$. Then, using the same hamiltonian circuit C, we see that H is as shown in Figure 11.59. This H is not 2-colorable. We have to eliminate some vertex to get K, for instance vertex $\{3, 5\}$. The resulting graph G' is the graph G of Figure 11.56, which is planar.

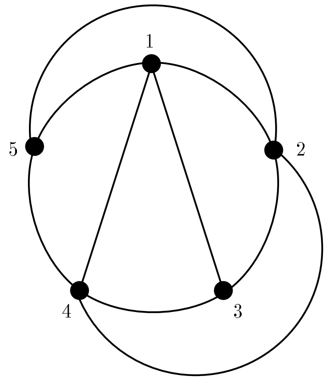


Figure 11.58: Planar drawing of graph G of Figure 11.56.

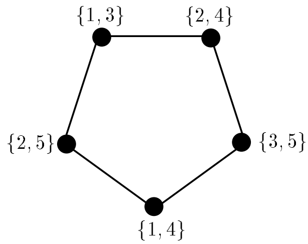


Figure 11.59: Graph H obtained if edge $\{3,5\}$ is added to G of Figure 11.56.

11.6.5 Sequencing by Hybridization[19]

In Example 11.2 we described the problem of determining a DNA string S from the list L of all k-length substrings, the problem of sequencing by hybridization. An alternative approach to this problem is to build a digraph $E(L)$ whose vertices are strings in L and which has an arc from $a_1a_2\cdots a_k$ in L to $b_1b_2\cdots b_k$ in L if $a_2 = b_1$, $a_3 = b_2$, $\ldots$, $a_k = b_{k-1}$. The digraph $E(L)$ corresponding to L of (11.2) is shown in Figure 11.60. Assuming, as in Example 11.2, that no k-length substring appears more than once in L, it is easy to see that there is a one-to-one correspondence between paths that visit each vertex of $E(L)$ exactly once and DNA strings whose k-length substrings correspond to strings in L. We use the first letter of each vertex in such a path and end with the last $k-1$ letters of the last vertex in the path. Thus, the DNA strings arising from L correspond to hamiltonian paths in $E(L)$. In Figure 11.60, the hamiltonian path ACA, CAC, ACG, CGC, GCA, CAA, AAC, ACT, CTT, TTA, TAA, AAA corresponds to the DNA string ACACGCAACTTAAA. However, there are other hamiltonian paths. For instance, the path ACG, CGC, GCA, CAA, AAA, AAC, ACA, CAC, ACT, CTT, TTA, TAA. Since the problem of finding hamiltonian paths is NP-complete, this method of finding an unknown DNA fragment is not as efficient as the eulerian path method described in Example 11.2.

EXERCISES FOR SECTION 11.6

1. In each of the following situations, determine if the digraph corresponding to the binary relation is a tournament.

[19] This subsection follows Pevzner [2000].

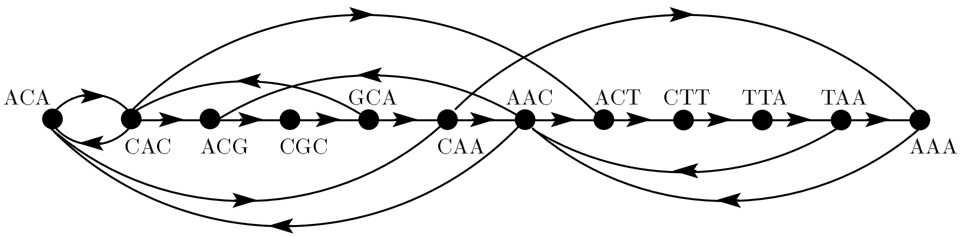


Figure 11.60: Digraph $E(L)$ obtained from list L of (11.2).

Table 11.3: Results of a Pair Comparison Experiment for Preference Among Cities as a Place to Live (Entry i, j is 1 iff i is Preferred to j)

	New York	Boston	San Francisco	Los Angeles	Houston
New York	0	0	0	0	1
Boston	1	0	0	1	1
San Francisco	1	1	0	1	1
Los Angeles	1	0	0	0	0
Houston	0	0	0	1	0

(a) The binary relation (X, R), where $X = \{1, 2, 3, 4\}$, R is defined by Equation (4.1).

(b) The binary relation (X, S), where $X = \{1, 2, 3, 4\}$, S is defined by Equation (4.2).

2. For each binary relation of Exercise 1, Section 4.1, determine if the corresponding digraph is a tournament.
3. For each binary relation of Exercise 1, determine if the digraph corresponding to the binary relation has a topological order and find such an order if there is one.
4. For each binary relation of Exercise 1, Section 4.1, determine if the digraph corresponding to the binary relation has a topological order.
5. For each property of a binary relation defined in Table 4.2, determine if it holds for a tournament (i) always, (ii) sometimes, or (iii) never.
6. Show that a transitive tournament is a strict linear order as defined in Section 4.2.
7. In each tournament of Figure 11.61, find all hamiltonian paths.
8. For the tournament defined by the preference data of Table 11.3, find all hamiltonian paths.
9. Which of the tournaments with four or fewer vertices is transitive?
10. Draw a digraph of a transitive tournament on five players.
11. Find a topological order for each of the digraphs of Figure 11.62.
12. In the book printing problem of Section 11.6.3, suppose that $b_1 = 3, b_2 = 5, b_3 = 8, p_1 = 6, p_2 = 2, p_3 = 9$. Find an optimal order in which to produce books.
13. Repeat Exercise 12 if $b_1 = 10, b_2 = 7, b_3 = 5, p_1 = 11, p_2 = 4, p_3 = 8$.

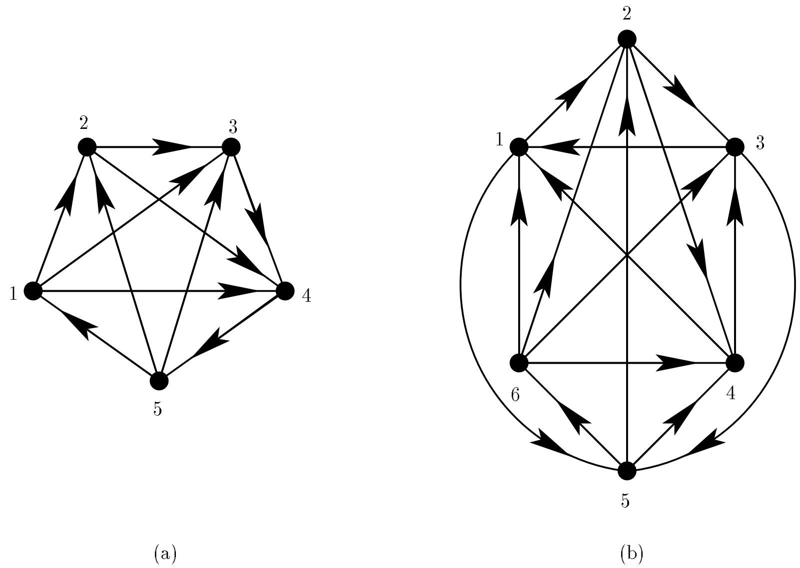


Figure 11.61: Tournaments for exercises of Section 11.6.

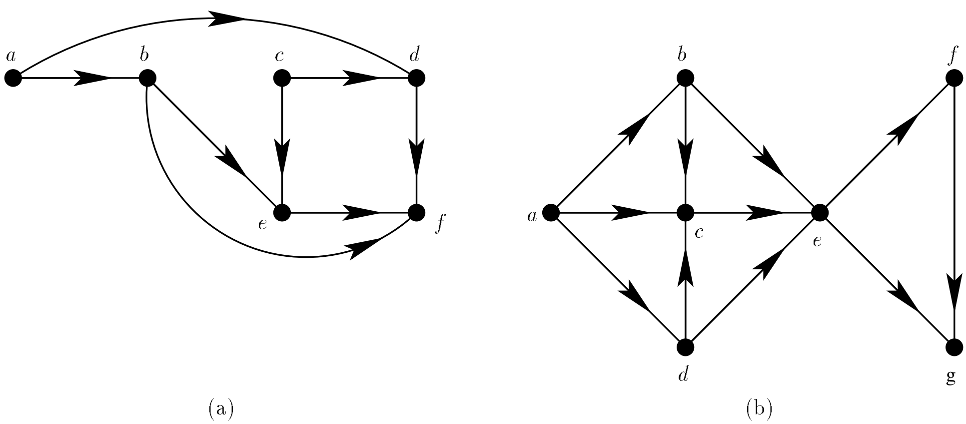


Figure 11.62: Digraphs for Exercise 11 of Section 11.6.

14. Design a layout plan for the planar graphs (a) and (b) of Figure 11.46.
15. If G is graph (a) of Figure 11.45, find a hamiltonian circuit C and construct H as in Section 11.6.4. Use H to determine a planar diagram for G, if one exists. Otherwise, find a planar G'.
16. Repeat Exercise 15 if G is graph (b) of Figure 11.45.
17. Repeat Exercise 15 if G is graph (b) of Figure 11.43.
18. (a) Draw digraph $E(L)$ of Section 11.6.5 for the list $L = \{$CCGA, CGCG, CGTG, CGAA, GCCG, GCGT, GTGC, GAAC, TGCC, AACC, ACCC$\}$.
    (b) Find the number of hamiltonian paths in $E(L)$.
    (c) Find the compatible DNA string(s) of L.
19. (a) Draw digraph $E(L)$ of Section 11.6.5 for the list $L = \{$AA, AC, AG, AT, CA, CT, GA, GT, TA, TC, TG, TT$\}$.
    (b) Is $S =$ ACTTCATGAGTAA compatible with L?
    (c) Is $S =$ AATGAGTACTTCA compatible with L?
    (d) Is $S =$ CTAGTACATGATA compatible with L?
20. In a tournament, the *score* of vertex u is the outdegree of u. (It is the number of players u beats.) If the vertices are labeled $1, 2, \ldots, n$, with $s(1) \leq s(2) \leq \cdots \leq s(n)$, the sequence $(s(1), s(2), \ldots, s(n))$ is called the *score sequence* of the tournament. Exercises 20–24 and 28 investigate the score sequence. Find the score sequence of the tournaments of:

    (a) Figure 11.51 (b) Figure 11.53

21. Show that if $(s(1), s(2), \ldots, s(n))$ is the score sequence of a tournament, then $\sum_{i=1}^{n} s(i) = \binom{n}{2}$.
22. Could each of the following be the score sequence of a tournament? Why?
    (a) $(1, 1, 2, 3)$ (b) $(0, 0, 0, 2, 7)$ (c) $(0, 1, 1, 4, 4)$
    (d) $(0, 0, 3, 3)$ (e) $(1, 2, 2, 3, 3, 4)$
23. Show that in a tournament, the ranking of players using the score sequence can be different from the ranking of players using a hamiltonian path.
24. Draw the digraph of a tournament with score sequence:

    (a) $(0, 1, 2, 3)$ (b) $(2, 2, 2, 2, 2)$ (c) $(1, 1, 1, 4, 4, 4)$

25. Show that a tournament is transitive if and only if it has no cycles of length 3.
26. Show that a tournament is transitive if and only if it is acyclic.
27. Prove Theorem 11.13 as follows.

    (a) Show that if D is transitive, it has a unique hamiltonian path, by observing that if there are two such paths, there must be u and v with u following v in one, but v following u in the other.
    (b) Prove the converse by observing that in the unique hamiltonian path u_1, u_2, $\ldots$, u_n, we have $(u_i, u_j) \in A$ iff $i < j$.

28. (a) Use the result of Exercise 27(b) to find the score sequence of a transitive tournament of n vertices.
    (b) Show that for a transitive tournament, the ranking obtained by the score sequence is the same as the ranking obtained by the unique hamiltonian path.
29. Suppose that the vertices of a tournament can be listed as $u_1, u_2, \ldots, u_n$ so that the score $s(u_i) = n - i$. Does it follow that the tournament has a hamiltonian path? (Give proof or counterexample.)
30. Use the result of Exercise 28(a) to determine if the preference data of Table 11.3 are transitive.
31. Show that every acyclic digraph has a vertex with no incoming arcs.
32. In a tournament, a triple $\{x, y, z\}$ of vertices is called *transitive* if the subgraph generated by x, y, and z is transitive, equivalently if one of these three vertices beats the other two. This one vertex is called the *transmitter* of the transitive triple.
    (a) How would you find the number of transitive triples of which a given vertex u is the transmitter?
    (b) Show that in a tournament, if $s(x)$ is the score of x, there are $\sum_x \binom{s(x)}{2}$ transitive triples.
    (c) Show that every tournament of at least four vertices has to have a transitive triple.
33. If you have not already done so, do Exercise 40, Section 3.6.
34. (Harary, Norman, and Cartwright [1965]) If D has a level assignment (Exercise 40, Section 3.6) and r is the length of the longest simple path of D, show that $r + 1$ is the smallest number of distinct levels (values L_i) in a level assignment for D.
35. If you have not already done so, do Exercise 41, Section 3.6.
36. In a transitive tournament, if vertex u has maximum score, u beats every other player.
    (a) Show that in an arbitrary tournament, if u has maximum score, then for every other player v, either u beats v or u beats a player who beats v. (This result was discovered by Landau [1955] during a study of pecking orders among chickens. For generalizations of this result, see Maurer [1980].)
    (b) Show that the necessary condition of part (a) for u to be a winner is not sufficient.
37. (Camion [1959] and Foulkes [1960]) Prove that every strongly connected tournament has a hamiltonian cycle. (*Hint:* Show that there is a cycle of length k for $k = 3, 4, \ldots, n$, where n is the number of vertices.)
38. An alternative criterion for optimality in the book production problem of Section 11.6.3 is to finish printing and binding all the books in as short a time as possible. Show that even under the special assumption of Section 11.6.3, there can be an optimal solution that does not correspond to a hamiltonian path in the digraph constructed in that section.

REFERENCES FOR CHAPTER 11

Abello, J., Buchsbaum, A., and Westbrook, J., "A Functional Approach to Extremal Graph Algorithms," *Algorithmica, 32* (2002), 437–458.

Abello, J., Pardalos, P. M., and Resende, M. G. C., "On Maximum Clique Problems in Very Large Graphs," in J. Abello and J. Vitter (eds.), *External Memory Algorithms*, DIMACS Series, Vol. 50, American Mathematical Society, Providence, RI, 1999, 119–130.

Abello, J., and Vitter, J. (eds.), *External Memory Algorithms*, DIMACS Series, Vol. 50, American Mathematical Society, Providence, RI, 1999.

Aho, A. V., Hopcroft, J. E., and Ullman, J. D., *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

Aiello, W., Chung, F. R. K., and Lu, L., "A Random Graph Model for Massive Graphs," *Proc. 32nd Annual ACM Symposium on Theory of Computing*, (2000), 171–180.

Atallah, M. J., "Parallel Strong Orientation of an Undirected Graph," *Inform. Process. Lett., 18* (1984), 37–39.

Baase, S., *Computer Algorithms*, 2nd ed., Addison-Wesley Longman, Reading, MA, 1992.

Beltrami, E. J., *Models for Public Systems Analysis*, Academic Press, New York, 1977.

Berge, C., *The Theory of Graphs and Its Applications*, Wiley, New York, 1962.

Bermond, J.-C., Bond, J., Martin, C., Pekeč, A., and Roberts, F. S., "Optimal Orientations of Annular Networks," *J. Interconn. Networks, 1* (2000), 21–46.

Boesch, F., and Tindell, R., "Robbins' Theorem for Mixed Graphs," *Amer. Math. Monthly, 87* (1980), 716–719.

Bondy, J. A., and Chvátal, V., "A Method in Graph Theory," *Discrete Math., 15* (1976), 111–136.

Camion, P., "Chemins et Circuits Hamiltoniens des Graphes Complets," *C. R. Acad. Sci. Paris, 249* (1959), 2151–2152.

Chachra, V., Ghare, P. M., and Moore, J. M., *Applications of Graph Theory Algorithms*, Elsevier North Holland, New York, 1979.

Chartrand, G., and Lesniak, L., *Graphs and Digraphs*, 3rd ed., CRC Press, Boca Raton, 1996.

Christofides, N., *Graph Theory: An Algorithmic Approach*, Academic Press, New York, 1975.

Chung, F. R. K., Garey, M. R., and Tarjan, R. E., "Strongly Connected Orientations of Mixed Multigraphs," *Networks, 15* (1985), 477–484.

Chvátal, V., and Thomassen, C., "Distances in Orientations of Graphs," *J. Comb. Theory B, 24* (1978), 61–75.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L., *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1999.

de Bruijn, N. G., "A Combinatorial Problem," *Nedl. Akad. Wet., Proc., 49* (1946), 758–764; *Indag. Math., 8* (1946), 461–467.

Demourcron, G., Malgrance, V., and Pertuiset, R., "Graphes Planaires: Reconnaissance et Construction de Representations Planaires Topologiques," *Recherche Operationelle, 30* (1964), 33.

Deo, N., *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall, Englewood Cliffs, NJ, 1974.

Dirac, G. A., "Some Theorems on Abstract Graphs," *Proc. Lond. Math. Soc., 2* (1952),

69–81.

Drmanac, R., Labat, I., Brukner, I., and Crkvenjakov, R., "Sequencing of Megabase Plus DNA by Hybridization: Theory of the Method," *Genomics, 4* (1989), 114–128.

Dror, M., *Arc Routing: Theory, Solutions and Applications*, Kluwer, Boston, 2000.

Euler, L., "Solutio Problematis ad Geometriam Situs Pertinentis," *Comment. Acad. Sci. I. Petropolitanae, 8* (1736), 128–140. [Reprinted in *Opera Omnia*, Series 1–7 (1766), 1–10.]

Even, S., *Graph Algorithms*, Computer Science Press, Potomac, MD, 1979.

Foulkes, J. D., "Directed Graphs and Assembly Schedules," *Proc. Symp. Appl. Math, Amer. Math. Soc., 10* (1960), 281–289.

Ghouila-Houri, A., "Une Condition Suffisante D'existence d'un Circuit Hamiltonien," *C. R. Acad. Sci. Paris, 156* (1960), 495–497.

Golomb, S. W., *Shift Register Sequences*, Aegean Park Press, Laguna Hills, CA, 1982.

Golumbic, M. C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

Good, I. J., "Normal Recurring Decimals," *J. Lond. Math. Soc., 21* (1946), 167–169.

Gusfield, D., *Algorithms on Strings, Trees and Sequences; Computer Science and Computational Biology*, Cambridge University Press, New York, 1997.

Gutin, Z. G., "Minimizing and Maximizing the Diameter in Orientations of Graphs," *Graphs Combin., 10* (1994), 225–230.

Han, X., "On the Optimal Strongly Connected Orientations of City Street Graphs: Over Twenty East-West Avenues or North-South Streets," mimeographed, Shanghai Computer Software Laboratory, Shanghai, China, 1989.

Harary, F., Norman, R. Z., and Cartwright, D., *Structural Models: An Introduction to the Theory of Directed Graphs*, Wiley, New York, 1965.

Harary, F., and Palmer, E. M., *Graphical Enumeration*, Academic Press, New York, 1973.

Helly, W., *Urban Systems Models*, Academic Press, New York, 1975.

Hopcroft, J. E., and Tarjan, R. E., "Algorithm 447: Efficient Algorithms for Graph Manipulation," *Commun. ACM, 16* (1973), 372–378.

Hutchinson, G., "Evaluation of Polymer Sequence Fragment Data Using Graph Theory," *Bull. Math. Biophys., 31* (1969), 541–562.

Hutchinson, J. P., and Wilf, H. S., "On Eulerian Circuits and Words with Prescribed Adjacency Patterns," *J. Comb. Theory A, 18* (1975), 80–87.

Johnson, E. L., "Chinese Postman and Euler Tour Problems in Bi-Directed Graphs," in M. Dror (ed.), *Arc Routing: Theory, Solutions and Applications*, Kluwer, Boston, 2000, 171–196.

Johnson, S. M., "Optimal Two- and Three-Stage Production Schedules with Setup Times Included," *Naval Res. Logist. Quart., 1* (1954), 61–68.

Karp, R. M., and Ramachandran, V., "Parallel Algorithms for Shared-Memory Machines," in J. Van Leeuwen (ed.), *Handbook of Theoretical Computer Science*, Vol. A, Elsevier, Amsterdam, 1990, 869–941.

Koh, K. M., and Tan, B. P., "The Diameter of an Orientation of a Complete Multipartite Graph," *Discr. Math., 149* (1996), 331–356. (a) [See also addendum *Discr. Math., 173* (1997), 297–298.]

Koh, K. M., and Tan, B. P., "The Minimum Diameter of Orientations of a Complete Multipartite Graph," *Graphs Combin., 12* (1996), 333–339. (b)

Kwan, M. K., "Graphic Programming Using Odd or Even Points," *Chin. Math., 1*

(1962), 273–277.

LANDAU, H. G., "On Dominance Relations and the Structure of Animal Societies III. The Condition for a Score Sequence," *Bull. Math. Biophys., 15* (1955), 143–148.

LAWLER, E. L., *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

LIEBLING, T. M., *Graphentheorie in Planungs- und Tourenproblemen*, Lecture Notes in Operations Research and Mathematical Systems No. 21, Springer-Verlag, New York, 1970.

LIU, C. L., *Introduction to Combinatorial Mathematics*, McGraw-Hill, New York, 1968.

LIU, C. L., *Topics in Combinatorial Mathematics*, Mathematical Association of America, Washington, DC, 1972.

MAURER, S. B., "The King Chicken Theorems," *Math. Magazine, 53* (1980), 67–80.

MCCANNA, J. E., "Orientations of the n-Cube with Minimum Diameter," *Disc. Math., 68* (1988), 309–310.

MINIEKA, E., *Optimization Algorithms for Networks and Graphs*, Dekker, New York, 1978.

ORE, O., "Note on Hamilton Circuits," *Amer. Math. Monthly, 67* (1960), 55.

PEVZNER, P. A., "1-Tuple DNA Sequencing: Computer Analysis," *J. Biomol. Structure Dynamics, 7* (1989), 63–73.

PEVZNER, P. A., *Computational Molecular Biology*, MIT Press, Cambridge, MA, 2000.

PEVZNER, P. A., and LIPSHUTZ, R., "Towards DNA Sequencing Chips," in *Proc. 19th Symp. Math. Found. Comp. Sci.*, LNCS 684, Springer, New York, 1994, 143–158.

PLESNIK, J., "Remarks on Diameters of Orientations of Graphs," *Math. Univ. Comeniance, 46/47* (1986), 225–236.

PÓSA, L., "A Theorem Concerning Hamiltonian Lines," *Magyar Tud. Akad. Mat. Kutató Int. Közl., 7* (1962), 225–226.

RÉDEI, L., "Ein kombinatorischer Satz," *Acta Litt. Sci. (Sect. Sci. Math.), Szeged, 7* (1934), 39–43.

REINGOLD, E. M., NIEVERGELT, J., and DEO, N., *Combinatorial Algorithms: Theory and Practice*, Prentice Hall, Englewood Cliffs, NJ, 1977.

REINGOLD, E. M., and TARJAN, R. E., "On a Greedy Heuristic for Complete Matching," *SIAM J. Comput., 10* (1981), 676–681.

ROBBINS, H. E., "A Theorem on Graphs, with an Application to a Problem of Traffic Control," *Amer. Math. Monthly, 46* (1939), 281–283.

ROBERTS, F. S., *Discrete Mathematical Models, with Applications to Social, Biological, and Environmental Problems*, Prentice Hall, Englewood Cliffs, NJ, 1976.

ROBERTS, F. S., *Graph Theory and Its Applications to Problems of Society*, NSF-CBMS Monograph No. 29, SIAM, Philadelphia, 1978.

ROBERTS, F. S., and XU, Y., "On the Optimal Strongly Connected Orientations of City Street Graphs. I: Large Grids," *SIAM J. Discr. Math., 1* (1988), 199–222.

ROBERTS, F. S., and XU, Y., "On the Optimal Strongly Connected Orientations of City Street Graphs. II: Two East-West Avenues or North-South Streets," *Networks, 19* (1989), 221–233.

ROBERTS, F. S., and XU, Y., "On the Optimal Strongly Connected Orientations of City Street Graphs. III: Three East-West Avenues or North-South Streets," *Networks, 22* (1992), 109–143.

ROBERTS, F. S., and XU, Y., "On the Optimal Strongly Connected Orientations of City Street Graphs. IV: Four East-West Avenues or North-South Streets," *Discr. Appl. Math., 49* (1994), 331–356.

Tarjan, R. E., "Depth-First Search and Linear Graph Algorithms," *SIAM J. Comput., 1* (1972), 146–160.

Tucker, A. C., and Bodin, L. D., "A Model for Municipal Street-Sweeping Operations," in W. F. Lucas, F. S. Roberts, and R. M. Thrall (eds.), *Discrete and System Models*, Vol. 3 of *Modules in Applied Mathematics*, Springer-Verlag, New York, 1983, 76–111.

van Aardenne-Ehrenfest, T., and de Bruijn, N. G., "Circuits and Tress in Oriented Linear Graphs," *Simon Stevin, 28* (1951), 203–217.

Vishkin, U., "On Efficient Parallel Strong Orientation," *Inform. Process. Lett., 20* (1985), 235–240.

Whitney, H., "Congruent Graphs and the Connectivity of Graphs," *Amer. J. Math., 54* (1932), 150–168.

Woodall, D. R., "Sufficient Conditions for Circuits in Graphs," *Proc. Lond. Math. Soc., 24* (1972), 739–755.

PART IV. COMBINATORIAL OPTIMIZATION

Chapter 12

Matching and Covering

12.1 SOME MATCHING PROBLEMS

In this chapter we study a variety of problems that fall into two general categories called matching problems and covering problems. We look at these problems in two ways, first as existence problems and then as optimization problems. Thus, this chapter will serve as a transition from our emphasis on the second basic problem of combinatorics, the existence problem, to the third basic problem, the optimization problem. We begin with a variety of examples.

Example 12.1 Job Assignments (Example 5.10 Revisited) In Example 5.10 we discussed a job assignment problem. In general, one can formulate this problem as follows. There are n workers and m jobs. Each worker is suited for some of the jobs. Assign each worker to one job, making sure that it is a job for which he or she is suited, and making sure that no two workers get the same job. In Example 5.10 we were concerned with counting the number of ways to make such an assignment. We used rook polynomials to do the counting. Here we ask an existence question: Is there *any* assignment that assigns each worker to one job to which he or she is suited, making sure that no job has two workers? Later, we ask an optimization question: What is the best assignment?

It will be convenient to formulate the existence question graph-theoretically. Build a graph G as follows. There are $m+n$ vertices in G, one for each worker and one for each job. Join each worker by an edge to each job for which he or she is suited. There are no other edges. Figure 12.1 shows the resulting graph G for one specific job assignment problem. Graph G is a *bipartite graph* (X, Y, E), a graph whose vertex set is divided into two sets X and Y, and with an edge set E so that all edges in E are between sets X and Y (see Section 3.3.4).

A job assignment of the kind we are seeking can be represented by replacing an edge from worker x to job y by a wiggly edge if x is assigned job y. Figure 12.2

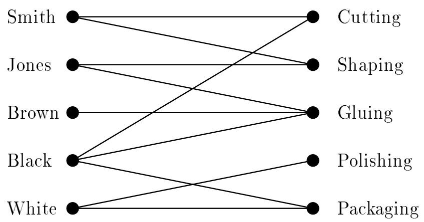


Figure 12.1: An edge from worker x to job y indicates that x is suitable for y.

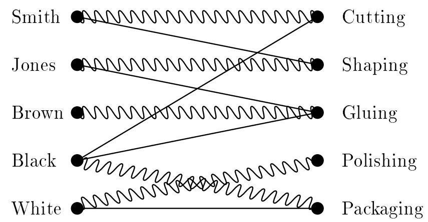


Figure 12.2: A job assignment for the graph of Figure 12.1 shown by wiggly edges.

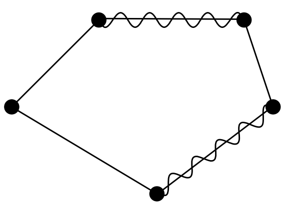


Figure 12.3: A matching M in a nonbipartite graph.

shows one such assignment. Note that in this assignment, each vertex is on at most one wiggly edge. This corresponds to each worker being assigned to at most one job and each job being assigned to at most one worker. A set M of edges in a graph G is called a *matching* if each vertex in G is on at most one edge of M. Thus, we seek a matching. In a matching M, a vertex is said to be *saturated* (M*-saturated*) if it is on some edge of M. We seek a matching that saturates every vertex corresponding to a worker. We first study matchings for bipartite graphs and then study them for arbitrary graphs. Figure 12.3 shows a matching in a nonbipartite graph. ∎

Example 12.2 Storing Computer Programs (Example 5.10 Revisited) In Example 5.10 we also discussed a problem of assigning storage locations to computer programs. Formulating this in general terms, we can think of n programs and m storage locations making up the vertices of a graph. We put an edge between program x and location y if y has sufficient storage capacity for x. We seek an assignment of programs to storage locations such that each storage location gets at most one program and each program is assigned to exactly one location, a location that has sufficient storage capacity for the program. Thus, we seek a matching in the corresponding bipartite graph that saturates all vertices corresponding to programs. In contrast to the situation in Example 5.10, where we were interested in counting the number of such assignments, here we are interested in the question of whether or not there is such an assignment. ∎

Example 12.3 Smallpox Vaccinations In preparing a plan for a possible outbreak of smallpox, a local health authority seeks to assign each person living in a city to a vaccination facility. They would like to find an assignment so that each person gets assigned to a facility no more than 10 miles from his or her home. We can think of a bipartite graph with vertex set X being the residents and vertex set Y being the facilities. An edge joins a resident to any facility within 10 miles of his or her home. Note that this is *not* a matching problem, since we allow more than one person to be assigned to each facility. However, we can reformulate it as a matching problem if we realize that each facility has a maximum capacity for vaccinations and replace Y by a set consisting of available appointment times in all facilities. Then we include an edge from a resident to all the appointment times in all facilities within 10 miles of his or her home and we seek a matching that saturates every X vertex. ■

Example 12.4 Pilots for RAF Planes (Berge [1973], Minieka [1978]) During the Battle of Britain in World War II, the Royal Air Force (RAF) had many pilots from foreign countries. The RAF had to assign two pilots to each plane, and always wanted to assign two pilots to the same plane whose language and training were compatible. The RAF's problem can be translated as follows. Given a collection of pilots available for use on a mission, use these as the vertices of a graph, and join two vertices with an edge if and only if the two pilots can fly together. Then the RAF wanted a matching in this graph. (It is not necessarily bipartite.) Moreover, they were interested in flying as large a number of planes as possible. Thus, they were interested in a *maximum-cardinality matching*, a matching of the graph with the largest possible number of edges. In Sections 12.4 and 12.5 we will be interested in solving the combinatorial optimization problem that asks for a maximum-cardinality matching. ■

Example 12.5 Real Estate Transactions (Minieka [1978]) A real estate agent at a given time has a collection X of potential buyers and a collection Y of houses for sale. Let r_{xy} be the revenue to the agent if buyer x purchases house y. From a purely monetary standpoint, the agent wants to match up buyers to houses so as to maximize the sum of the corresponding r_{xy}. We can represent the problem of the agent by letting the vertices X and Y define a *complete bipartite graph* $G = (X, Y, E)$; that is, we take all possible edges between X and Y. We then seek a matching M in G that maximizes the sum of the weights. More generally, we can consider the following *maximum-weight matching problem*. Suppose that G is an arbitrary graph (not necessarily bipartite) with a weight (real number) r_{xy} on each edge $\{x, y\}$. If M is a matching of G, we define

$$r(M) = \sum \{r_{xy} : \{x, y\} \in M\}.$$

We seek a *maximum-weight matching* of G, a matching M such that for all the matchings M' of G, $r(M) \geq r(M')$. ■

Example 12.6 The Optimal Assignment Problem Let us return to the job assignment problem of Example 12.1, but add the simplifying assumption that every worker is suited for every job. Then the "is suited for" graph G is a complete bipartite graph. Let us also assume that worker x is given a rating r_{xy} for his or her potential performance (output) on job y. We seek an assignment of workers to jobs that assigns every worker a job, no more than one per job, and maximizes the sum of the ratings. The problem of finding such an assignment is called the *optimal assignment problem.* If there are at least as many jobs as workers, the problem can be solved. It calls for a matching of the complete bipartite graph G that saturates the set of workers and has at least as large a sum of weights as any other matching that saturates the set of workers. But it is clear that every maximum-weight matching saturates the set of workers. Hence, the optimal assignment problem reduces to the maximum-weight matching problem for a complete bipartite graph. We return to the optimal assignment problem in Section 12.7, where we present an algorithm for solving it. (The algorithm can be understood from the material of Section 12.4, specifically Corollary 12.5.1.) ■

Example 12.7 Smallpox Vaccinations (Example 12.3 Revisited) In Example 12.3 we tried to assign people to vaccination facilities within 10 miles of their homes. Now, suppose that we wish to assign people to the closest possible vaccination facilities. We can let r_{xy} represent the distance from person x to the facility whose appointment time is represented by y. Then we seek an assignment that minimizes the sum of the r_{xy} over all x, y, where x is assigned to y. This is the minimum variant of the optimal assignment problem of Example 12.6. An alternative optimal assignment problem arises if people have preferences for different appointment times in different facilities. Let s_{xy} represent the rating by person x of appointment time y in its corresponding facility, independent of whether y is within 10 miles of x. Then we might seek an assignment of people to facilities so that the sum of the ratings is maximized, exactly as in the job assignment problem (Example 12.1). ■

Example 12.8 Pairing Speakers in Sound Systems (Ahuja, Magnanti, and Orlin [1993], Mason and Philpott [1988]) A manufacturer of sound systems seeks to pair up speakers before selling them as a set. How the two speakers will perform as a set depends on frequency response. The response f_{xy} between speakers x and y is measured (by comparing responses at a variety of frequencies), with low f_{xy} being better than high f_{xy}. The manufacturer might only be willing to pair up speakers if the measure f_{xy} is sufficiently small, e.g., less than some value T. Then, starting with a collection of speakers, the manufacturer might wish to create as many pairs of speakers as possible. This is a *maximum-cardinality matching problem.* The speakers are the vertices and an edge between x and y means that f_{xy} is less than T. A more sophisticated goal is to create pairs with $f_{xy} < T$ so that the sum of all of the f_{xy} over such pairs is as small as possible. This is the minimum-weight matching problem. In contrast to that of Example 12.5 (which has maximum instead of minimum), note that here the problem is on an arbitrary graph, not necessarily a bipartite graph. ■

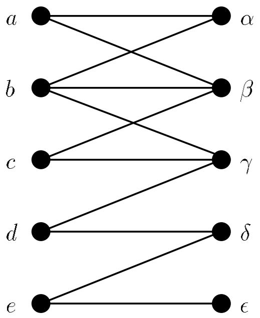


Figure 12.4: Graph for Exercise 1, Section 12.1.

Example 12.9 Oil Drilling (Ahuja, Magnanti, and Orlin [1993], Devine [1973]) A number of potential drilling sites have been identified by an oil company. It might be cheaper to drill one well for two sites rather than to drill separate wells. The company estimates the cost of drilling at each site, as well as the cost of drilling one well for each pair of sites. The problem is to identify which sites will involve a single well and which will be paired up so as to minimize total drilling costs. If we disregard the single-well possibility, we have a minimum-weight matching problem. If we allow an edge from a well to itself, we can also formulate the entire problem as a minimum-weight matching problem. Note that the graph in question is not bipartite. ■

There are many other applications of matching. Examples described by Ahuja, Magnanti, and Orlin [1993] include determining chemical bonds, rewiring special electrical typewriters in manufacturing, locating objects in space, matching moving objects, optimal depletion of inventory, and scheduling on parallel machines. We explore some of these applications in the exercises.

In Section 12.7 we shall see how the maximum-weight matching problem arises in connection with the "Chinese Postman" Problem of Section 11.4.1 and the related computer graph plotting problem of Section 11.4.2. For a discussion of the general maximum-weight matching problem as well as others described in the chapter, see, for example, Ahuja, Magnanti, and Orlin [1993], Christofides [1975], Cook, *et al.* [1998], Grötschel, Lovász, and Schrijver [1993], Lawler [1976], Lovász and Plummer [1986], Minieka [1978], or Papadimitriou and Steiglitz [1982].

EXERCISES FOR SECTION 12.1

1. In the graph of Figure 12.4:

   (a) Find a matching that has four edges.

   (b) Find a matching that saturates vertex b.

   (c) Find a matching that is not maximum but is maximal.

2. Repeat Exercise 1 for the graph of Figure 12.5.

3. In each weighted graph of Figure 12.6, find a maximum-weight matching.

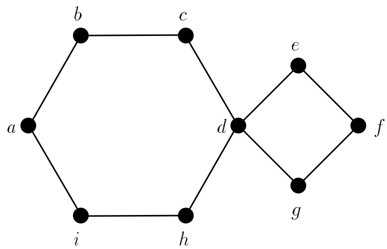


Figure 12.5: Graph for Exercise 2, Section 12.1.

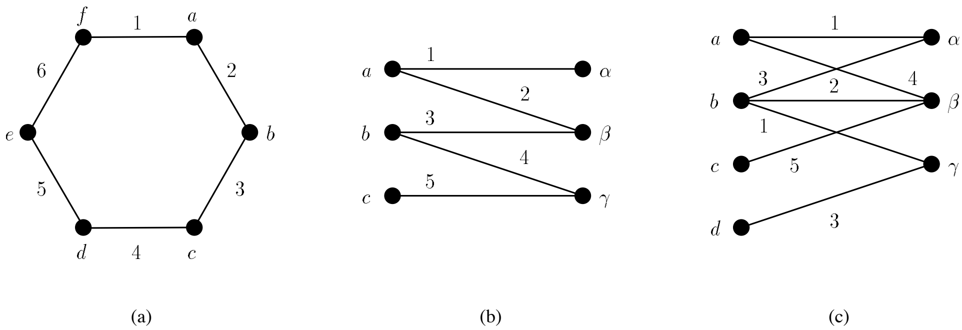


Figure 12.6: Weighted graphs for Exercise 3, Section 12.1.

4. A company has five positions to fill: computer technician (c), receptionist (r), accountant (a), word processor operator (w), and data analyst (d). There are seven applicants for jobs. The first applicant is qualified for the positions c, r, a; the second for the positions w, d; the third for the positions r, w, d; the fourth for the positions c, a; the fifth for the positions c, r, a, w; the sixth for the position c only; and the seventh for all the positions. Can all the positions be filled? Formulate this problem in the language of this section. What are we looking for? You are not asked to solve the problem at this point.

5. A company assigns its workers to temporary jobs. The i, j entry of a matrix gives the distance from worker i's home to the jth job. This matrix is given information. If every worker is suited for every job, find an assignment of workers to jobs that minimizes the sum of the distances the workers have to travel. Formulate this problem in the language of this section. Do not solve. What are we looking for?

6. There are six students looking for rooms in a dormitory: Ingram, Knight, Marks, Odell, Quincy, and Spencer. Ingram likes Knight, Marks, Odell, and Spencer; Knight likes Ingram, Odell, Quincy, and Spencer; Marks likes Ingram, Quincy, and Spencer; Odell likes Ingram, Knight, and Quincy; Quincy likes Knight, Marks, and Odell; and Spencer likes Ingram, Knight, and Marks. Note that a likes b iff b likes a. We wish to assign roommates, two to a room, such that each person only gets a roommate whom he likes. Can we assign each person to a room? If not, what is the largest number of people we can assign to rooms? Formulate this problem in the language of this section. Do not solve. What are we looking for?

7. (Ahuja, Magnanti, and Orlin [1993], Brogan [1989]) Two infrared sensors are used to identify objects in space. Each is used to provide a line of sight to the object, and

the two lines help us determine the location of the object. Suppose that we are given p lines from the first sensor and p lines from the second, but don't know which line corresponds to which object. A line from one sensor might intersect more than one line from the second sensor. Also, lines from the two sensors corresponding to the same object might not intersect due to measurement error. We wish to match up lines from the two sensors. Formulate this as a matching problem: Define the graph, define any appropriate measure r_{xy}, and comment on the type of matching problem involved: M-saturating, maximum-cardinality matching on a bipartite graph, and so on.

8. (Ahuja, Magnanti, and Orlin [1993], Brogan [1989], Kolitz [1991]) In missile defense and other applications, we wish to estimate speed and direction of objects in space. We can take "snapshots" of a variety of objects at two different times. If we can match up the first and second snapshots corresponding to an object, we can estimate its speed (from the time between snapshots) and direction of movement. Let r_{xy} denote the distance between snapshots. If we assume that snapshots are taken very quickly, we can try to match up objects with small r_{xy}. Formulate this problem as a matching problem, as in Exercise 7.

9. (Ahuja, Magnanti, and Orlin [1993], Derman and Klein [1959]) Items in our inventory (books, chemicals, etc.) may either gain or lose value over time. Suppose that at the beginning, we know the age of every item in our inventory. Suppose that we have a set of times when we must remove an item from inventory to use in some process, such as a manufacturing process. When should we remove a given item? Suppose that we can measure the value of an item of age a using a utility function $u(a)$. Then we can measure the utility u_{ij} of removing at time t_j an item of age a_i at the beginning of the process. Formulate the problem of finding a plan to remove items from inventory as a matching problem, as in Exercise 7.

10. Show that to describe a general solution to the maximum-weight matching problem for a weighted graph G, we may assume that:

    (a) G has an even number of vertices. (b) G is a complete graph.

11. Show that to describe a general solution to the maximum-weight matching problem for a bipartite graph G, we may assume that the graph is complete bipartite and both classes have the same number of vertices.

12. Show that the maximum-weight matching for a weighted bipartite graph does not necessarily saturate the first class of vertices, even if that class has no more than half the vertices.

13. Draw a bipartite graph $G = (X, Y, E)$ with $|X| = |Y| = 3$ such that G has a unique X-saturating matching and $|E|$ is as large as possible. Explain why a larger G does not exist.

14. (Ahuja, Magnanti, and Orlin [1993]) A ski rental shop wishes to assign n pairs of skis to n skiers. The skis come in lengths $l_1 \leq l_2 \leq \cdots \leq l_n$ and the skiers have heights $h_1 \leq h_2 \leq \cdots \leq h_n$. Ideally, a skier of height h will receive a ski of length αh, where α is a fixed constant. However, given the limited supply of skis, the shop can only try to minimize the sum of the (absolute) differences between αh_i and the length of the ski assigned to skier i. Show that if ski pairs and skiers are labeled as above, the assignment of the ith pair of skis to the ith skier is optimal.

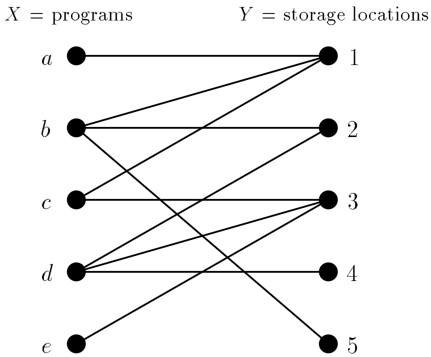


Figure 12.7: Bipartite graph for computer program storage. The vertices in X are the programs, the vertices in Y are the storage locations, and there is an edge between x in X and y in Y if and only if y has sufficient capacity to store x.

12.2 SOME EXISTENCE RESULTS: BIPARTITE MATCHING AND SYSTEMS OF DISTINCT REPRESENTATIVES

12.2.1 Bipartite Matching

In many of the examples of the preceding section, the graph constructed was a bipartite graph $G = (X, Y, E)$. Here we shall ask the question: Given a bipartite graph $G = (X, Y, E)$, under what conditions does there exist a matching that saturates X?

Consider the computer program storage assignment graph of Figure 12.7. Note that the three programs a, c, e have together only two possible locations they could be stored in, locations 1 and 3. Thus, there is no storage assignment that assigns each program to a location of sufficient storage capacity. There is no X-saturating matching. To generalize this example, it is clear that for there to exist an X-saturating matching, if S is any set of vertices in X and $N(S)$ is the *open neighborhood* of S, the set of all vertices y joined to some x in S by an edge, then $N(S)$ must have at least as many elements as S. In our example, S was $\{a, c, e\}$ and $N(S)$ was $\{1, 3\}$. What is startling is that this obvious necessary condition is sufficient as well.

Theorem 12.1 (Philip Hall's Theorem) (Hall [1935]) Let $G = (X, Y, E)$ be a bipartite graph. Then there exists an X-saturating matching if and only if for all subsets S of X, $|N(S)| \geq |S|$.

We prove Theorem 12.1 in Section 12.4.

To see whether the bipartite graph of Figure 12.1 has an X-saturating matching, where $X = \{\text{Smith, Jones, Brown, Black, White}\}$, we have to compute $N(S)$ for all subsets S of X. There are $2^{|S|} = 2^5 = 32$ such subsets. To make the computation, we note for instance that

$$N(\{\text{Smith, Jones, Black}\}) = \{\text{Cutting, Shaping, Gluing, Packaging}\},$$

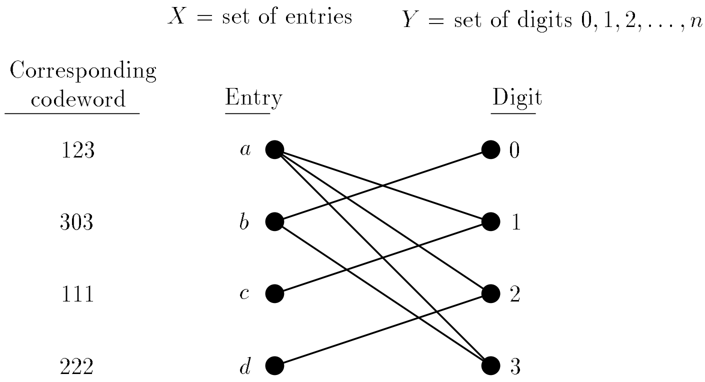


Figure 12.8: Bipartite graph for the coding problem with $n = 3$.

Table 12.1: Subsets S of X and Corresponding Neighborhoods $N(S)$, for Graph of Figure 12.8

S	$\emptyset$	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{a,b\}$
$N(S)$	$\emptyset$	$\{1,2,3\}$	$\{0,3\}$	$\{1\}$	$\{2\}$	$\{0,1,2,3\}$
S	$\{a,c\}$	$\{a,d\}$	$\{b,c\}$	$\{b,d\}$	$\{c,d\}$	$\{a,b,c\}$
$N(S)$	$\{1,2,3\}$	$\{1,2,3\}$	$\{0,1,3\}$	$\{0,2,3\}$	$\{1,2\}$	$\{0,1,2,3\}$
S	$\{a,b,d\}$	$\{a,c,d\}$	$\{b,c,d\}$	$\{a,b,c,d\}$		
$N(S)$	$\{0,1,2,3\}$	$\{1,2,3\}$	$\{0,1,2,3\}$	$\{0,1,2,3\}$		

so $N(S)$ has four elements, whereas S has three. A similar computation for all 32 cases shows that there is an X-saturating matching. (In this case, finding one directly is faster.)

Example 12.10 A Coding Problem Each entry in a small file has a three-digit codeword associated with it, using the digits $0, 1, 2, \ldots, n$. Can we associate with each entry in the file just one of the digits of its codeword so that the entry can be recovered uniquely from this single digit? We can formulate this problem graph-theoretically as follows. The vertices of a bipartite graph $G = (X, Y, E)$ consist of X, the set of entries, and Y, the set of digits $0, 1, 2, \ldots, n$. There is an edge from entry x to digit y if y is used in the codeword for x. Figure 12.8 gives an example. We seek an X-saturating matching in the bipartite graph G. In our example, Table 12.1 lists all subsets S of X and the corresponding neighborhood $N(S)$. Note that in each case, $|N(S)| \geq |S|$. Thus, an X-saturating matching exists. This does not help us to find one. However, it is easy to see that the edges $\{a,3\}, \{b,0\}, \{c,1\}, \{d,2\}$ form such a matching. ■

A graph G is called *regular* if every vertex has the same degree, that is, the same number of neighbors.

Corollary 12.1.1 Suppose that $G = (X, Y, E)$ is a regular bipartite graph with at least one edge. Then G has an X-saturating matching.

Proof. Let S be a subset of X. Let E_1 be the collection of edges leading from vertices of S, and E_2 be the collection of edges leading from vertices of $N(S)$. Since every edge of E_1 leads from a vertex of S to a vertex of $N(S)$, we have $E_1 \subseteqq E_2$. Thus, $|E_1| \leq |E_2|$. Moreover, since every vertex has the same number of neighbors, say k, and since $S \subseteqq X$, it follows that $|E_1| = k|S|$ and $|E_2| = k|N(S)|$. Thus, $k|S| \leq k|N(S)|$. Since G has an edge, k is positive, so we conclude that $|S| \leq |N(S)|$. Q.E.D.

As an application of Corollary 12.1.1, suppose in Example 12.10 that every codeword uses exactly k of the digits $0, 1, 2, \ldots, n$ and every digit among $0, 1, 2, \ldots, n$ appears in exactly k codewords. Then there is an association of one digit with each file entry that allows a unique decoding.

Note that the X-saturating matching of a regular bipartite graph $G = (X, Y, E)$ is also Y-saturating, for every edge of G touches both a vertex of X and a vertex of Y. Thus, if k is the number of neighbors of each vertex, $k|X| = |E|$ and $k|Y| = |E|$. Thus, $k|X| = k|Y|$ and $|X| = |Y|$. Thus, every X-saturating matching must also be Y-saturating. A matching that saturates every vertex of a graph is called *perfect*. Perfect matchings in job assignments mean that every worker gets a job and every job gets a worker. The question of when in general there exists a perfect matching is discussed further in Section 12.3 (see also Exercises 18, 21, 24, and 26).

12.2.2 Systems of Distinct Representatives

Suppose that $\mathcal{F} = \{S_1, S_2, \ldots, S_p\}$ is a family of sets, not necessarily distinct. Let $T = (a_1, a_2, \ldots, a_p)$ be a p-tuple with $a_1 \in S_1, a_2 \in S_2, \ldots, a_p \in S_p$. Then T is called a *system of representatives* for $\mathcal{F}$. If, in addition, all the a_i are distinct, T is called a *system of distinct representatives* (SDR) for $\mathcal{F}$. For instance, suppose that

$$\mathcal{F} = \{S_1, S_2, S_3, S_4, S_5\},$$

where

$$\begin{aligned} S_1 = \{a, b, c\}, S_2 = \{b, c, d\}, S_3 = \{c, d, e\} \\ S_4 = \{d, e\}, S_5 = \{e, a, b\}. \end{aligned} \tag{12.1}$$

Then $T = (a, b, c, d, e)$ is an SDR for $\mathcal{F}$. Next, suppose that $\mathcal{F} = \{S_1, S_2, S_3, S_4, S_5, S_6\}$, where

$$\begin{aligned} S_1 = \{a, b\}, S_2 = \{b, c\}, S_3 = \{a, b, c\}, \\ S_4 = \{b, c, d\}, S_5 = \{a, c\}, S_6 = \{c, d\}. \end{aligned} \tag{12.2}$$

Then $T = (a, b, c, d, a, d)$ is a system of representatives for $\mathcal{F}$. However, there is no SDR, as we show below.

Example 12.11 List Colorings (Example 3.22 Revisited) In Example 3.22 we introduced the notion of list coloring a graph G that has a list assignment with a list of acceptable colors, $L(x)$, for each vertex x. A list coloring is an ordinary graph coloring with the color assigned to vertex x chosen from $L(x)$. If G is a complete graph, a list coloring corresponds to an SDR for the lists $L(x)$. ■

Example 12.12 Hospital Internships In a given year, suppose that p medical school graduates have applied for internships in hospitals. For the ith medical school graduate, let S_i be the set of all hospitals that find i acceptable. Then a system of representatives for the family $\mathcal{F} = \{S_i : i = 1, 2, \ldots, p\}$ would assign each potential intern to a hospital that is willing to take him or her. An SDR would make sure that, in addition, no hospital gets more than one intern. In practice, SDRs could be used in the following way. Modify S_i to include i. Find an SDR. This assigns each i to a hospital or to himself or herself; the latter is interpreted to mean that on the first round, i is not assigned to a hospital. (Exercise 7 asks the reader to show that an SDR exists.) After the initial assignment based on an SDR, the hospitals would be asked to modify their list of acceptable applicants among those not yet assigned. (Hopefully, at least some are assigned in the first round.) Then a new SDR would be found; and so on. A similar procedure could be used to place applicants for admission to graduate school. More complicated procedures actually in use in the National Resident Matching Program since 1952 make use of hospitals' and also applicants' ratings or rankings of the alternatives. We return to this idea in Section 12.7. ■

The problem of finding an SDR can be formulated graph-theoretically as follows. Suppose that X is the collection of sets S_i in $\mathcal{F}$ and Y is the collection of points in $\cup S_i$. Let $G = (X, Y, E)$, be a bipartite graph, with an edge from x in X to y in Y iff y is in x. Then an SDR is simply an X-saturating matching in G.

For instance, consider the family of sets in (12.2). The corresponding bipartite graph is shown in Figure 12.9. Note that if $S = \{S_1, S_2, S_3, S_5\}$, then $N(S) = \{a, b, c\}$. Thus, $|S| > |N(S)|$, so by Philip Hall's Theorem, there is no X-saturating matching and hence no SDR. Using the language of SDRs, we may now reformulate Philip Hall's Theorem as follows.

Corollary 12.1.2 The family $\mathcal{F} = \{S_1, S_2, \ldots, S_p\}$ possesses an SDR iff for all $k = 1, 2, \ldots, n$, any k S_i's together contain at least k elements of $\cup S_i$.

Example 12.13 Extending Latin Rectangles A *Latin rectangle* is an $r \times s$ array using the elements $1, 2, \ldots, n$, such that each row and column uses each element from $1, 2, \ldots, n$ at most once. A Latin rectangle is called *complete* if $n = s$. One way to build up an $n \times n$ Latin square is to try building it up one row at a time, that is, by creating a complete $r \times n$ Latin rectangle and then extending it. For instance, consider the 2×6 Latin rectangle of Figure 12.10. Can we extend this to a 6×6 Latin square? More particularly, can we add a third row to obtain a 3×6 Latin rectangle? One approach to this question is to use rook polynomials (see Exercise 15, Section 5.1). Another approach is to ask what numbers could be

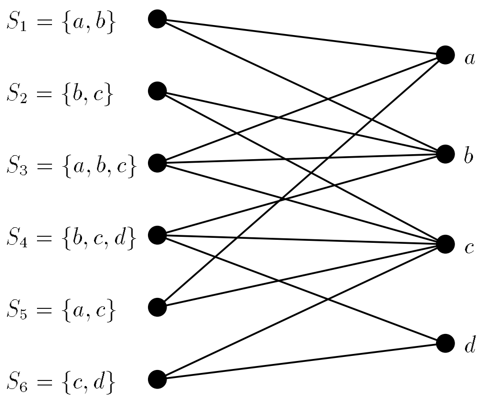


Figure 12.9: Bipartite graph representing the family of sets of (12.2).

1	2	3	4	5	6
4	3	6	5	1	2

Figure 12.10: A 2×6 Latin rectangle.

1	2	3	4	5	6
4	3	6	5	1	2
2	1	4	3	6	5

Figure 12.11: A 3×6 Latin rectangle obtained by extending the Latin rectangle of Figure 12.10.

placed in the new row in the ith column. Let S_i be the set of numbers not yet occurring in the ith column. In our example,

$$S_1 = \{2, 3, 5, 6\}, \quad S_2 = \{1, 4, 5, 6\}, \quad S_3 = \{1, 2, 4, 5\},$$
$$S_4 = \{1, 2, 3, 6\}, \quad S_5 = \{2, 3, 4, 6\}, \quad S_6 = \{1, 3, 4, 5\}.$$

We want to pick one element from each S_i and we want these elements to be distinct. Thus, we want an SDR. One SDR is (2, 1, 4, 3, 6, 5). Thus, we can use this as a new third row, obtaining the Latin rectangle of Figure 12.11. This idea generalizes. In general, given an $r \times n$ complete Latin rectangle, let S_i be the set of numbers not yet occurring in the ith column. To add an $(r + 1)$st row, we need an SDR for the family of S_i. We shall show that we can always find one. ■

Theorem 12.2 If $r < n$, then every $r \times n$ complete Latin rectangle L can be extended to an $(r + 1) \times n$ complete Latin rectangle.

Proof.[1] If S_i is defined as in Example 12.13, we shall show that the family of S_i possesses an SDR. Pick k of the sets in this family, $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$. By Corollary 12.1.2, it suffices to show that $A = S_{i_1} \cup S_{i_2} \cup \cdots \cup S_{i_k}$ has at least k elements. Each S_{i_j} has $n - r$ elements. Thus, A has $k(n - r)$ elements, including repetitions. Since the $r \times n$ Latin rectangle L is complete, each number in $1, 2, \ldots, n$ appears

[1] The proof may be omitted.

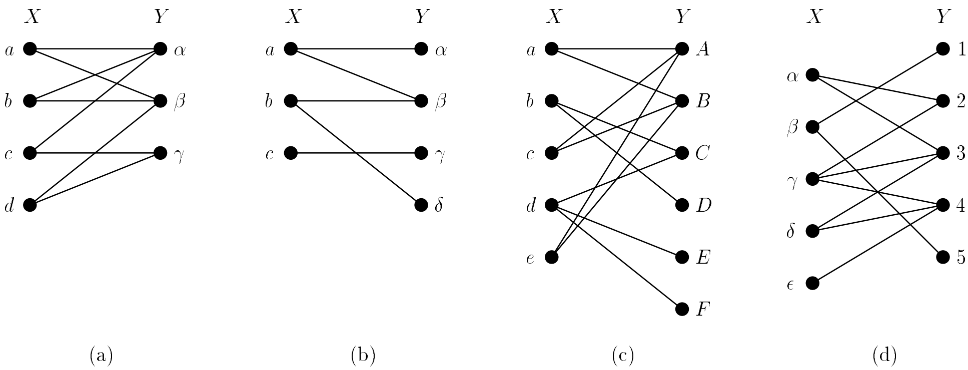


Figure 12.12: Bipartite graphs for exercises of Section 12.2.

exactly once in each row, so each number in $1, 2, \ldots, n$ appears exactly r times in L. Thus, each number in $1, 2, \ldots, n$ is in exactly $n - r$ of the sets $S_1, S_2, \ldots, S_n$. Thus, each number in A appears in at most $n - r$ of the sets $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$. Now if we make a list of elements of A, including repetitions, we have $k(n - r)$ elements. Each number in A appears in this list at most $n - r$ times. By the pigeonhole principle (Theorem 2.15), there must be at least k distinct numbers in A. Q.E.D.

EXERCISES FOR SECTION 12.2

1. Find $N(S)$ if G and S are as follows:

   (a) $G =$ graph of Figure 12.1 and $S = \{$Smith, White$\}$

   (b) $G =$ graph of Figure 12.8 and $S = \{a, c, d\}$

   (c) $G =$ graph of Figure 12.9 and $S = \{S_1, S_3, S_4\}$

2. Find $N(S)$ if G and S are as follows:

   (a) $G =$ graph of Figure 12.1 and $S = \{$Cutting, Shaping$\}$

   (b) $G =$ graph of Figure 12.1 and $S = \{$Smith, Black, Polishing$\}$

   (c) $G =$ graph of Figure 12.5 and $S = \{a, b, e\}$

3. For each bipartite graph $G = (X, Y, E)$ of Figure 12.12, determine if G has an X-saturating matching.

4. (a) Find a bipartite graph corresponding to the family of sets of (12.1).

   (b) Use Philip Hall's Theorem to show that there is an X-saturating matching in this graph.

5. For each of the following families of sets, find a system of representatives.

   (a) $S_1 = \{a, b, f\}$, $S_2 = \{a\}$, $S_3 = \{a, b, d, f\}$, $S_4 = \{a, b\}$, $S_5 = \{b, f\}$, $S_6 = \{d, e, f\}$, $S_7 = \{a, f\}$

(b) $S_1 = \{a, c\}$, $S_2 = \{a, c\}$, $S_3 = \{a, c\}$, $S_4 = \{a, b, c, d, e\}$

(c) $S_1 = \{a_1, a_2, a_3\}$, $S_2 = \{a_3, a_4\}$, $S_3 = \{a_1, a_2, a_3\}$, $S_4 = \{a_2, a_4\}$

(d) $S_1 = \{b_3, b_5\}$, $S_2 = \{b_1, b_3, b_4\}$, $S_3 = \{b_3, b_4, b_5\}$, $S_4 = \{b_3, b_4, b_5\}$

(e) $S_1 = \{x, y\}$, $S_2 = \{x\}$, $S_3 = \{u, v, w\}$, $S_4 = \{x, y, z\}$, $S_5 = \{y, z\}$

(f) $S_1 = \{a, b, c\}$, $S_2 = \{b, c\}$, $S_3 = \{c, e, f\}$, $S_4 = \{a, b\}$, $S_5 = \{a, c\}$, $S_6 = \{d, e, f\}$

6. For each of the families of sets in Exercise 5, determine if there is a system of distinct representatives and if so, find one.

7. In Example 12.12, show that if i is in S_i, an SDR exists.

8. Find the number of SDRs of each of the following families of sets.

   (a) $S_1 = \{1, 2\}, S_2 = \{2, 3\}, S_3 = \{3, 4\}, S_4 = \{4, 5\}, S_5 = \{5, 1\}, S_6 = \{6, 1\}$

   (b) $S_1 = \{1, 2\}, S_2 = \{2, 3\}, S_3 = \{3, 4\}, \ldots, S_n = \{n, 1\}$

   (c) $S_1 = \{1, 2\}, S_2 = \{3, 4\}, S_3 = \{5, 6\}, S_4 = \{7, 8\}, S_5 = \{9, 10\}$

   (d) $S_1 = \{1, 2\}, S_2 = \{3, 4\}, S_3 = \{5, 6\}, \ldots, S_n = \{2n - 1, 2n\}$

9. There are six committees of a state legislature: Finance, Environment, Health, Transportation, Education, and Housing. In Chapter 1 we discussed a meeting assignment problem for these committees. Here, we ask a different question. Suppose that there are 12 legislators who need to be assigned to committees, each to one committee. The following matrix has its i, j entry equal to 1 iff the ith legislator would like to serve on the jth committee.

	Finance	Environment	Health	Transportation	Education	Housing
Allen	1	1	1	0	0	0
Barnes	1	1	0	1	1	0
Cash	1	1	1	0	0	0
Dunn	1	0	0	1	1	1
Ecker	0	1	1	0	0	0
Frank	1	1	0	0	0	0
Graham	1	1	1	0	0	0
Hall	1	0	0	0	0	0
Inman	1	1	1	0	0	0
Johnson	1	1	0	0	0	0

   Suppose that we want to choose exactly one new member for each committee, choosing only a legislator who would like to serve. Can we do so? (Not every legislator needs to be assigned to a committee, and no legislator can be assigned to more than one committee.)

10. Consider the complete graph on the vertices $\{a, b, c, d, e, f\}$ with lists $L(a) = \{2, 3, 4\}$, $L(b) = \{3, 4\}$, $L(c) = \{1, 4, 6\}$, $L(d) = \{2, 3\}$, $L(e) = \{2, 4\}$, $L(f) = \{1, 5, 6\}$. Is there a list coloring?

11. Suppose that $G = (X, Y, E)$ is a complete bipartite graph with a list $L(a)$ assigned to each vertex a in $X \cup Y$. Give proof or counterexample: G has an L-list coloring iff the set of lists $L(a)$ has an SDR.

12. An *edge coloring* of graph G is an assignment of a color to each edge of G so that if two edges have a vertex in common, they get different colors.

(a) Show that an edge coloring with k colors decomposes a graph into k edge-disjoint matchings.

(b) Show that if a bipartite graph G is regular with each vertex of degree k, there is an edge coloring with k colors.

(c) Show that if G is a bipartite graph and $\Delta(G)$ is the maximum degree of a vertex in G, then G has an edge coloring using $\Delta(G)$ colors. (*Hint:* Make G regular by adding vertices and edges.)

13. If q is a positive integer and $S_1, S_2, \ldots, S_p$ are sets of integers, we say that an SDR $(a_1, a_2, \ldots, a_p)$ is a system of *q-distant representatives* if $|a_i - a_j| \geq q$ for all $i \neq j$. This concept, that arises in channel assignment, is due to Fiala, Kratochvíl, and Proskurowski [2001]. Suppose that G is a complete graph and that a list $L(a)$ of integers is assigned to each vertex a of G. Show that G has a T-coloring (Example 3.20) for a certain set T iff the lists $L(a)$ have a system of q-distant representatives.

14. Suppose that we are given a collection of codewords and we would like to store the collection on a computer as succinctly as possible, namely by picking one letter from each word to store.

    (a) Can we do this for the following set of codewords in such a way that the codewords can be uniquely decoded? Words: $abcd, cde, a, b, ce$.

    (b) If so, in how many ways can it be done?

15. Solve Exercise 4, Section 12.1.

16. Consider the following arrays. In each case, can we add one more column to the array, using the numbers $1, 2, 3, 4, 5, 6, 7$, so that the entries in this new column are all different and so that the entries in each row of the new array are all different?

(a)

123
265
371
436
542

(b)

67345
12734
73456
34567

17. Give proof or counterexample: Every regular bipartite graph with at least one edge has a perfect matching.

18. (a) What circuits Z_n have perfect matchings?

    (b) What chains L_n of n vertices have perfect matchings?

    (c) A wheel graph W_n is obtained from Z_n by adding a vertex adjacent to all vertices of Z_n. What wheel graphs W_n have perfect matchings?

    (d) Which trees have perfect matchings?

19. There are n men and n women, and each man likes exactly p women and each woman likes exactly p men. Assuming that a likes b iff b likes a, show that the $2n$ people can be paired off into couples so that each man is paired with a woman he likes and who likes him.

20. Is the result of Exercise 19 still true if each man likes at least p women and each woman likes at most p men? Why?

21. (Tucker [1980]) There are n men and n women enrolled in a computer dating service. Suppose that the service has made nm pairings, so that each man dates m different women and each woman dates m different men, and suppose that $m < n$.

    (a) Suppose that we want to schedule all the dates over a period of m nights. Show that it is possible to do so. That is, show that the pairings can be divided into m perfect matchings.

    (b) Show that we can make the division in part (a) one step at a time; that is, show that no matter how the first k perfect matchings are chosen, we can always find a $(k+1)$st.

22. Let $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ be a family of sets that has an SDR.

    (a) If x is in at least one of the S_i, show that there is an SDR that chooses x as the representative of at least one of the S_i.

    (b) For each S_i that contains x, is there necessarily an SDR in which x is picked as the representative of S_i? Why?

23. Let $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$, where $S_i = \{1, 2, \ldots, n\} - \{i\}$.

    (a) Show that $\mathcal{F}$ has an SDR.

    (b) Show that the number of different SDRs is the number of derangements of n elements, D_n.

24. Is it possible for a tree to have more than one perfect matching? Why?

25. Consider a bipartite graph $G = (X, Y, E)$. Suppose that $m \geq 1$ is the maximum degree of a vertex of G and X_1 is the subset of X consisting of all vertices of degree m. Assume that X_1 is nonempty. Give a proof or a counterexample of the following assertion: There is a matching of G in which all vertices of X_1 are saturated.

26. (a) If H is a graph, let $o(H)$ count the number of components of H with an odd number of vertices. If S is a set of vertices in graph G, $G - S$ is the subgraph of G generated by vertices not in S. Show that if G has a perfect matching, $o(G - S) \leq |S|$ for all $S \subseteqq V(G)$. (Tutte [1947] proves this result and also its converse.)

    (b) (Peterson [1891]) Generalizing the ideas of Section 11.2, let an edge $\{a, b\}$ in a graph G be called a *bridge* if removal of the edge, but not its end vertices a and b, increases the number of components. Suppose that G is a graph in which every vertex has degree 3 and suppose that G does not have any bridges. Assuming the converse in part (a), show that G has a perfect matching.

27. Suppose that $A_1, A_2, \ldots, A_m$ and $B_1, B_2, \ldots, B_m$ are two partitions of the same set S with the same number of A_i and B_j and with all $A_i \neq \emptyset$ and all $B_j \neq \emptyset$. (A *partition* of S is a division of all elements of S into disjoint subsets.) Let E be an m-element subset of S such that $A_i \cap E \neq \emptyset$ for all i and $B_j \cap E \neq \emptyset$ for all j. Then it is clear that $|A_i \cap E| = |B_j \cap E| = 1$ for all i, j. The set E is called a *system of common representatives* (SCR) for the partitions A_i and B_j.

    (a) Find an SCR for the following partitions of $S = \{1, 2, 3, 4, 5, 6\}$: $A_1 = \{1, 2, 3\}, A_2 = \{4\}, A_3 = \{5, 6\}$; $B_1 = \{1, 3\}, B_2 = \{2, 6\}, B_3 = \{4, 5\}$.

(b) Show that an SCR exists for the partitions A_i and B_j if and only if there is a suitable renumbering of the components of the partitions A_i and B_j such that for all $i, A_i \cap B_i \neq \emptyset$.

(c) Show that the partitions A_i and B_j have an SCR if and only if for all $k = 1, 2, \ldots, m$ and all $i_1, i_2, \ldots, i_k$ and $j_1, j_2, \ldots, j_{k-1}$ from $1, 2, \ldots, m$, the set $A_{i_1} \cup A_{i_2} \cup \cdots \cup A_{i_k}$ is not a subset of the set $B_{j_1} \cup B_{j_2} \cup \cdots \cup B_{j_{k-1}}$. (*Hint:* Form a bipartite graph and construct a matching.)

12.3 THE EXISTENCE OF PERFECT MATCHINGS FOR ARBITRARY GRAPHS[2]

Example 12.14 Pilots for RAF Planes (Example 12.4 Revisited) Let us consider whether or not, given a group of pilots, it is possible to assign two compatible pilots to each airplane, in such a way that every pilot is assigned to a plane. This is the case if and only if the graph constructed in Example 12.4 has a perfect matching, a matching in which every vertex is saturated. In this section we investigate one condition that guarantees the existence of a perfect matching. ■

Theorem 12.3 If a graph G has $2n$ vertices and each vertex has degree $\geq n$, the graph has a perfect matching.

We show why this theorem is true by trying to build up a matching M one step at a time. Here is an algorithm for doing this.

Algorithm 12.1: Finding a Perfect Matching

Input: A graph G with $2n$ vertices and each vertex having degree $\geq n$.
Output: A perfect matching of G.

Step 1. Set $M = \emptyset$.

Step 2. Find any pair of unsaturated vertices a and b that are joined by an edge of G, and place edge $\{a, b\}$ in M.

Step 3. If there is a pair of unsaturated vertices of G joined by an edge of G, return to Step 2. Otherwise, go to Step 4.

Step 4. If M has n edges, stop and output M. If M does not have n edges, go to Step 5.

Step 5. Find a pair of unsaturated vertices a and b in G. These will not be joined by an edge of G. (Why?) Find an edge $\{u, v\}$ in M such that $\{a, u\}$ and $\{b, v\}$ are edges of G. Remove edge $\{u, v\}$ from M and place edges $\{a, u\}$ and $\{b, v\}$ in M. Return to Step 4.

[2] This section may be omitted without loss of continuity.

Table 12.2: The Edges in M at Each Stage of Algorithm 12.1 Applied to the Graph of Figure 12.13

Stage	1	2	3	4
Edges in M	None	$\{1,6\}$	$\{1,6\},\{3,5\}$	$\{1,6\},\{2,3\},\{4,5\}$

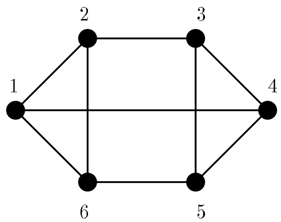


Figure 12.13: Graph used to illustrate Algorithm 12.1.

We illustrate this procedure with the graph of Figure 12.13. We show below why the procedure works. Note that in this graph, there are $2n = 2(3) = 6$ vertices, and each has degree $\geq n = 3$. Start in Step 2 by placing edge $\{1, 6\}$ in M. Table 12.2 shows the successive edges of M. Go to Step 3. Next, since there is a pair of unsaturated vertices joined by an edge of G, go to Step 2. Find such a pair, say 3, 5, and place edge $\{3, 5\}$ in M. Now there is no pair of unsaturated vertices joined by an edge of G, and since M has fewer than n edges, we proceed from Step 3 to Step 4 to Step 5. Pick vertices 2 and 4, that are unsaturated in G and not joined by an edge of G. Note that 3 and 5 are matched in M, and there are edges $\{2, 3\}$ and $\{4, 5\}$ in G. Thus, remove $\{3, 5\}$ from M, add $\{2, 3\}$ and $\{4, 5\}$, and return to Step 4. Since we have $n = 3$ elements in M, we stop, having obtained a perfect matching.

To see why this procedure works, note that at each stage we have a matching. Moreover, if the algorithm stops, we always end up with n edges in M, and since there are $2n$ vertices in G, we must have a perfect matching. The crucial question is this: How do we know that Step 5 works? In particular, how do we know that there always exist u and v as called for in Step 5? To see why,[3] suppose that a and b are unsaturated in M and not joined by an edge of G. There are at present $r < n$ edges in M. If Step 5 could not be carried out, then for every edge $\{u, v\}$ in M, at most two of the edges $\{a, u\}$, $\{a, v\}$, $\{b, u\}$, $\{b, v\}$ are in G. The number of edges from a or b to matched edges $\{u, v\}$ in G is thus at most $2r$. Every edge from a or from b goes to some edge in M, for otherwise a or b is joined by an edge of G to an unsaturated vertex of G, and we would not have gotten to Step 5 in the algorithm. Thus, degree of a + degree of $b \leq 2r$. Since $r < n$, degree of a + degree of $b < 2n$. But by hypothesis, degree of a + degree of $b \geq n + n = 2n$, which is a contradiction. We conclude that Step 5 can always be carried out.

Let us examine the computational complexity of Algorithm 12.1. To get a crude upper bound, suppose that the ith iteration of the algorithm is the procedure

[3] This argument may be omitted.

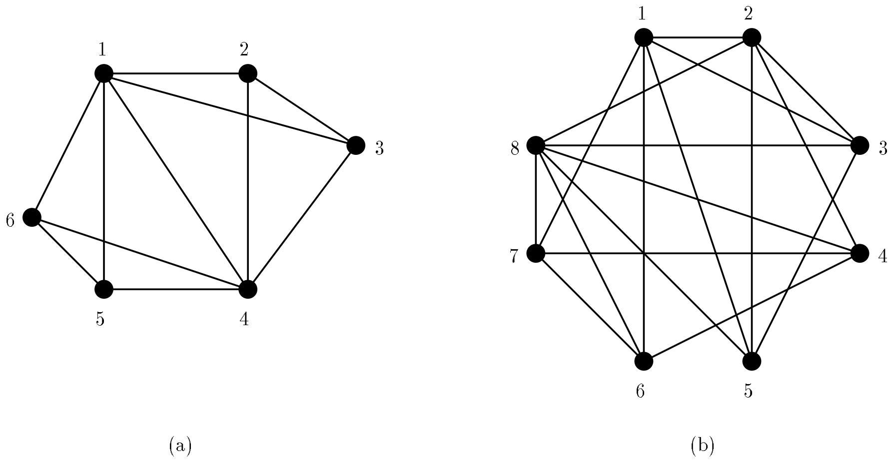


Figure 12.14: Graphs for exercises of Sections 12.3 and 12.4.

between the time the matching M has $i-1$ edges and the time it has i edges. Thus, there are n iterations in all. We can keep a record of the unsaturated vertices. Thus, it takes at most e steps to find an edge joining a pair of unsaturated vertices, where $e = |E(G)|$. If there is no such edge, we pick an arbitrary pair of unsaturated vertices a and b. For each edge $\{u, v\}$ in M, we ask if $\{a, u\}$ and $\{b, v\}$ are edges of G or if $\{a, v\}$ and $\{b, u\}$ are edges of G. This requires four questions for each of at most e edges, so at most $4e$ steps. Thus, we have at most $e + 4e = 5e$ steps in each iteration, and at most $n(5e)$ steps in all. Since $e \leq \binom{n}{2} \leq n^2$, we have at most $5n^3$ steps, a polynomial bound. In the terminology of Section 2.18, we have an $O(n^3)$ algorithm.

EXERCISES FOR SECTION 12.3

1. Suppose that K_{2p} is the complete graph of $2p$ vertices, $p \geq 2$, and G is obtained from K_{2p} by removing an edge. Does G have a perfect matching?

2. The *complete p-partite graph* $K(n_1, n_2, \ldots, n_p)$ has p classes of vertices, n_i vertices in the ith class, and all vertices in class i joined to all vertices in class j, for $i \neq j$, with no other vertices joined. Which of the following graphs have perfect matchings?

   (a) $K(2, 2)$ (b) $K(3, 3)$ (c) $K(1, 5)$ (d) $K(2, 3)$
   (e) $K(2, 2, 2)$ (f) $K(3, 3, 3)$ (g) $K(4, 4, 4)$ (h) $K(2, 2, 2, 2)$

3. In Exercise 6, Section 12.1, can all students be assigned roommates?

4. For each graph of Figure 12.14, use Algorithm 12.1 to find a perfect matching.

5. Suppose that G has $2n$ vertices and for all $a \neq b$ such that $\{a, b\}$ is not an edge of G, degree of a + degree of $b \geq 2n$. Show that G has a perfect matching.

12.4 MAXIMUM MATCHINGS AND MINIMUM COVERINGS

12.4.1 Vertex Coverings

A *vertex covering* in a graph, or a *covering* (or *cover*) for short, is any set of vertices such that each edge of the graph has at least one of its end vertices in the set. For example, in the graph of Figure 12.13, the set of vertices $\{1,2,3,4,5\}$ forms a covering. So does the set of vertices $\{1,3,5,6\}$. In this section we study *minimum-cardinality coverings* (*minimum coverings* for short). These are coverings that use as few vertices as possible. We shall relate minimum coverings to *maximum-cardinality matchings* (*maximum matchings* for short). Such matchings arose in Example 12.4.

Example 12.15 Police Surveillance A police officer standing at a street intersection in a city can watch one block in any direction. If we attempt to locate a criminal, we want to be sure that all the blocks in a neighborhood are being watched simultaneously. Assuming that police officers stand only at street intersections, what is the smallest number of police officers we need, and where do we put them? The answer is obtained by letting the street intersections in the neighborhood be vertices of a graph G and joining two vertices by an edge if and only if they are joined by a one-block street. Then we seek a minimum covering of the graph G. ■

Example 12.16 Smallpox Vaccinations (Example 12.3 Revisited) In Example 12.3 we investigated assignments of people to vaccination facilities not more than 10 miles from their home. There, we assumed that vaccination facilities were already established. But what if we want to locate as small a number of facilities as we can so that every person is within 10 miles of one of the facilities. We could let each square block of the city be a vertex and join two vertices with an edge if they are within 10 miles of each other. Assuming that each square block is inhabited, we seek a minimum-size set of vertices (square blocks) so that every edge has one of its end vertices in that set. Such a set is a minimum covering of the graph. ■

Next we investigate the relation between matchings and coverings. Let G be a graph, let M be a matching of G, and let K be a covering. Then $|M| \leq |K|$, for given any edge $\alpha = \{x, y\}$ of M, either x or y is in K. Let $f(\alpha)$ be whichever of x and y is in K, picking either if both are in K. Note that since M is a matching, if α and β are two edges of M, $f(\alpha)$ must be different from $f(\beta)$. Why? Thus, we assign each edge of M to a different element of K. Hence, M must have no more elements than K.

It follows that if M^* is a maximum matching and K^* is a minimum covering, $|M^*| \leq |K^*|$. Note that $|M^*|$ can be strictly less than $|K^*|$. In the graph of Figure 12.13, a minimum covering K^* consists of vertices $1, 3, 5, 6$, so $|K^*| = 4$. However, a maximum matching consists of three edges. For instance, $M^* = \{\{1,2\},\{3,4\},\{5,6\}\}$ is a maximum matching here. Hence, $|M^*| = 3 < |K^*|$. A crucial result is the following.

Theorem 12.4 If M is a matching and K is a covering and $|M| = |K|$, K is a minimum covering and M is a maximum matching.

Proof. If M^* is any maximum matching and K^* is any minimum covering, we have

$$|M| \leq |M^*| \leq |K^*| \leq |K|. \tag{12.3}$$

Since $|M| = |K|$, all terms in (12.3) must be equal, and in particular $|M| = |M^*|$ and $|K| = |K^*|$ hold. Q.E.D.

For example, consider the graph of Figure 12.9. Here there is a matching M consisting of the four edges $\{S_1, a\}, \{S_3, b\}, \{S_4, c\}$, and $\{S_6, d\}$, and a covering K consisting of the four vertices a, b, c, d. Thus, by Theorem 12.4, M is a maximum matching and K is a minimum covering. This example illustrates the following result.

Theorem 12.5 (König [1931]) Suppose that $G = (X, Y, E)$ is a bipartite graph. Then the number of edges in a maximum matching equals the number of vertices in a minimum covering.

This theorem is proved in Section 13.3.8.

As a corollary of this theorem, we can now derive Philip Hall's Theorem, Theorem 12.1.

Proof of Philip Hall's Theorem (Theorem 12.1).[4] It remains to show that if $|N(S)| \geq |S|$ for all $S \subseteqq X$, there is an X-saturating matching. By Theorem 12.5, since G is bipartite, an X-saturating matching exists if and only if each cover K has $|K| \geq |X|$. Let K be a cover and let $X - S$ be the set of X-vertices in K. Then vertices in S are not in K, so all vertices of $N(S)$ must be in K. It follows that

$$|K| \geq |X - S| + |N(S)| = |X| - |S| + |N(S)| \geq |X|.$$

Hence, $|K| \geq |X|$ for all K, so an X-saturating matching exists. Q.E.D.

Another corollary of Theorem 12.5 will be important in Section 12.7 in an algorithm for the optimal assignment problem (Example 12.6). To state this corollary, let $\mathbf{A}$ be a matrix of 0's and 1's. A *line* of this matrix is either a row or a column. A set of 0's in this matrix is called *independent* if no two 0's lie in the same line. For instance, suppose that

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}. \tag{12.4}$$

Then the $(2,1)$ and $(1,3)$ entries form an independent set of 0's of $\mathbf{A}$, as do the $(2,1), (4,2)$, and $(3,3)$ entries.

[4] This proof may be omitted.

Corollary 12.5.1 (König-Egerváry Theorem[5]) If **A** is a matrix of 0's and 1's, a maximum independent set of 0's has the same number of elements as a minimum set of lines covering all the 0's of **A**.

Proof. Build a bipartite graph $G = (X, Y, E)$ by letting X be the rows of **A**, Y the columns of **A**, and joining rows i and j by an edge if and only if the i, j entry of **A** is 0. Then a maximum independent set of 0's of **A** corresponds to a maximum matching of G, and a minimum set of lines covering all the 0's of **A** corresponds to a minimum covering of G. Q.E.D.

To illustrate this result, for **A** of (12.4), a maximum independent set of 0's is the set of three 0's at the $(2, 1), (4, 2)$, and $(3, 3)$ entries. A minimum set of lines covering all the 0's of **A** consists of the second row and the second and third columns.

12.4.2 Edge Coverings

A collection F of edges in a graph is called an *edge covering* if every vertex of the graph is on one of the edges in F. An edge covering is called *minimum* if no other edge covering has fewer edges. Minimum edge coverings have applications to switching functions in computer engineering, to crystal physics, and to other fields (see Deo [1974, pp. 184ff.]). We investigate edge coverings and their relations to matchings and vertex coverings in the exercises.

EXERCISES FOR SECTION 12.4

1. In graph (a) of Figure 12.14, find:

   (a) A covering of five vertices (b) A minimum covering

2. In the graph of Figure 12.1, find:

   (a) A covering of six vertices (b) A minimum covering

3. In an attempt to improve the security of its computer operations, a company has put in 20 special passwords. Each password is known by exactly two people in the company. Find the smallest set of people who together know all the passwords. Formulate this problem using the terminology of this section.
4. Suppose that **A** is a matrix of 0's and 1's. Suppose that we find k lines covering all 0's of **A** and we find k independent 0's. What conclusion can we draw?
5. Is the König-Egerváry Theorem still true if the matrix is allowed to have any integers as entries? Why?
6. In each of the following graphs, find an edge covering that is not minimum and an edge covering that is minimum.

   (a) The graph of Figure 12.13 (b) Graph (a) of Figure 12.14
   (c) Graph (b) of Figure 12.14 (d) Graph (b) of Figure 12.12

[5] This theorem is based on work of König [1931] and Egerváry [1931].

7. (a) Can a graph with isolated vertices have an edge covering?
   (b) What is the smallest conceivable number of edges in an edge covering of a graph of n vertices?
8. A patrol car is parked in the middle of a block and can observe the two street intersections at the end of the block. How would you go about finding the smallest number of patrol cars required to keep all street corners in a neighborhood under surveillance? (See Gondran and Minoux [1984].)
9. (Minieka [1978]) A committee is to be chosen with at least one member from each of the 50 states and at least one member from each of the 65 major ethnic groups in the United States. What is the smallest committee that can be constructed from a group of volunteers if the requirements are to be met? To answer this question, let the vertices of a graph be the 50 states and the 65 ethnic groups, and let a volunteer correspond to an edge joining his or her state and ethnic group. What are we looking for to answer the question?
10. Illustrate Theorem 12.5 on the grid graphs G_{n_1,n_2} defined in Section 11.2.4 by calculating maximum matchings and minimum coverings in:
    (a) $G_{1,2}$ (b) $G_{1,m}$ (c) $G_{2,2}$ (d) $G_{n,n}$
11. (a) Show that Theorem 12.5 fails for at least some annular grids $AN(c, s)$ defined in Section 11.2.5.
    (b) Does it hold for any annular grids?
12. (a) Can a minimum edge covering of a graph contain a circuit? Why?
    (b) Show that every minimum edge covering of a graph of n vertices has at most $n - 1$ edges.
13. (Gallai [1959]) Suppose that G is a graph without isolated vertices, K^* is a minimum (vertex) covering of G, and I^* is a maximum independent set of vertices. Show that $|K^*| + |I^*| = |V|$.
14. (Norman and Rabin [1959], Gallai [1959]) Suppose that G is a graph without isolated vertices, M^* is a maximum matching, and F^* is a minimum edge covering. Show that $|M^*| + |F^*| = |V|$.
15. Suppose that I is an independent set of vertices in a graph G without isolated vertices and F is an edge covering of G. Show that $|I| \leq |F|$.
16. Suppose that we find an independent set I of vertices in a graph G without isolated vertices and an edge covering F in G such that $|I| = |F|$. What conclusion can we draw? Why?
17. Suppose that $G = (X, Y, E)$ is a bipartite graph without isolated vertices. Let I^* be an independent set with a maximum number of vertices (a *maximum independent set*) and F^* be a minimum edge covering of G. Show from Exercises 13 and 14 that $|I^*| = |F^*|$.
18. (Minieka [1978]) Consider the following algorithms on a graph with no isolated vertices. *Algorithm 1* starts with a matching M and selects any unsaturated vertex x. It adds to M any edge incident to x and repeats the procedure until every vertex is saturated. The resulting set of edges is called C'. *Algorithm 2* starts with an edge covering C and selects any vertex x covered by more than one edge in C. It removes from C an edge that covers x and repeats the procedure until no vertex is covered by more than one edge. The resulting set of edges is called M'.

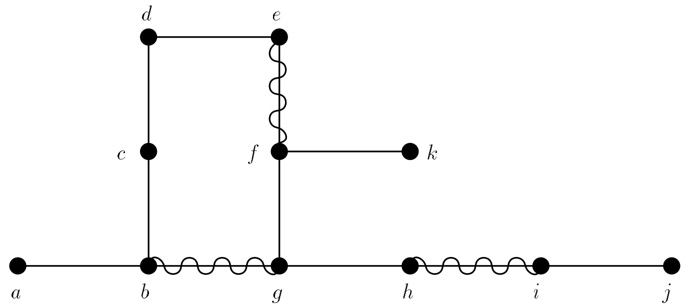


Figure 12.15: A matching M is shown by the wiggly edges.

(a) Show that C' as produced by Algorithm 1 is an edge covering.

(b) Show that M' as produced by Algorithm 2 is a matching.

(c) Show that if M is a maximum matching, then C' is a minimum edge covering.

(d) Show that if C is a minimum edge covering, then M' is a maximum matching.

19. Derive Philip Hall's Theorem (Theorem 12.1) from Theorem 12.4.

12.5 FINDING A MAXIMUM MATCHING

12.5.1 M-Augmenting Chains

In this section we present a procedure for finding a maximum matching in a graph G. Suppose that M is some matching of G. An *M-alternating chain* in G is a simple chain

$$u_1, e_1, u_2, e_2, \ldots, u_t, e_t, u_{t+1} \tag{12.5}$$

such that e_1 is not in M, e_2 is in M, e_3 is not in M, e_4 is in M, and so on. For instance, consider the matching shown in Figure 12.15 by wiggly edges. Then the chain $a, \{a,b\}, b, \{b,g\}, g, \{g,f\}, f, \{f,e\}, e$ is M-alternating. If an M-alternating chain (12.5) joins two vertices u_1 and u_{t+1} that are not M-saturated (i.e., not on any edge of M), we call the chain an *M-augmenting chain*. Our example is not M-augmenting, since e is M-saturated. However, the chain

$$a, \{a,b\}, b, \{b,g\}, g, \{g,f\}, f, \{f,e\}, e, \{e,d\}, d \tag{12.6}$$

is M-augmenting. Let us now find a new matching M' by deleting from M all edges of M used in the M-augmenting chain (12.6) and adding all edges of (12.6) not in M. Then M' is shown in Figure 12.16. It is indeed a matching. Moreover, M' has one more edge than M. This procedure always gives us a larger matching.

Theorem 12.6 Suppose that M is a matching and C is an M-augmenting chain of M. Let M' be M less edges of C in M plus edges of C not in M. Then M' is a matching and $|M'| > |M|$.

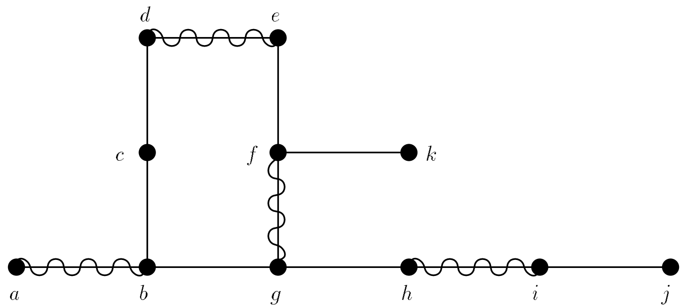


Figure 12.16: The matching M' obtained from the matching M of Figure 12.15 by using the M-augmenting chain (12.6).

Proof. To see that $|M'| > |M|$, note that in an M-augmenting chain (12.5), e_1 and e_t cannot be in M, so t is odd. Hence, we add the edges $e_1, e_3, \ldots, e_t$ and delete the edges $e_2, e_4, \ldots, e_{t-1}$, and we add one more edge than we take away. To see that M' is a matching, note that if j is odd, edge e_j is added and edge e_{j+1} is subtracted. Thus, if j is odd and $j \neq 1, t+1$, u_j was previously only on e_{j-1} and is now only on e_j. If j is even, u_j was previously only on e_j and is now only on e_{j-1}. Also, u_1 was previously unmatched and is now only on e_1, and u_{t+1} was previously unmatched and is now only on e_t. Q.E.D.

As a corollary of Theorem 12.6, we observe that if M has an M-augmenting chain, M could not be a maximum matching. In fact, the converse is also true.

Theorem 12.7 (Berge [1957], Norman and Rabin [1959]) A matching M of G is maximum if and only if G contains no M-augmenting chain.

To apply this theorem, note that the matching M' in Figure 12.16 is not maximum, since there is an M'-augmenting chain

$$j, \{j,i\}, i, \{i,h\}, h, \{h,g\}, g, \{g,f\}, f, \{f,k\}, k. \tag{12.7}$$

If we modify M' by deleting edges $\{i,h\}$ and $\{g,f\}$, the M' edges of the chain (12.7), and adding edges $\{j,i\}, \{h,g\}$, and $\{f,k\}$, the non-M' edges of (12.7), we obtain the matching M'' shown in Figure 12.17. There is no M''-augmenting chain, since there is only one unsaturated vertex. Thus, M'' is maximum.

12.5.2 Proof of Theorem 12.7[6]

To prove Theorem 12.7, it remains to show that if there is a matching M' such that $|M'| > |M|$, there is an M-augmenting chain. Let H be the subgraph of G consisting of all vertices of G and all edges of G that are in M or in M' but not in both M and M'. Note that in H, there are more M' edges than M edges. Moreover, each

[6]This subsection may be omitted.

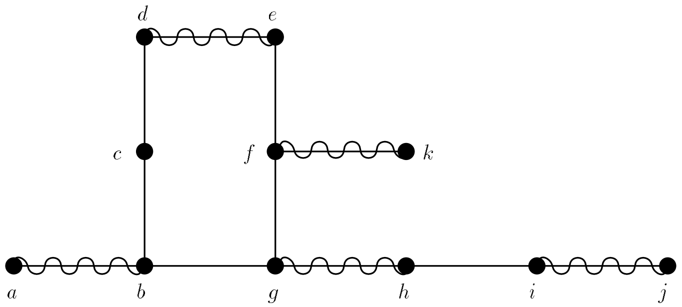


Figure 12.17: The matching M'' obtained from the matching M' of Figure 12.16 by using the M'-augmenting chain (12.7).

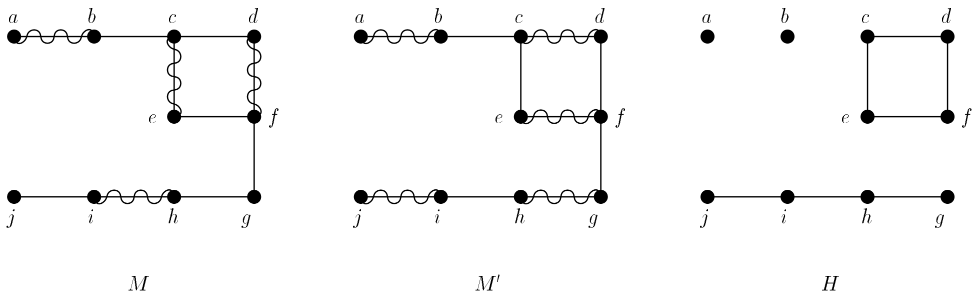


Figure 12.18: The graph H is obtained from the matchings M and M'.

vertex of H has at most two neighbors in H, for it can have at most one neighbor in M and at most one neighbor in M'. Using the latter observation, one can show (see Exercise 10) that each connected component of H is either a circuit Z_n or a simple chain L_n of n vertices. Moreover, the edges in Z_n or L_n must alternate between M and M' because M and M' are matchings. It follows that each Z_n has an equal number of edges from M and from M'. Thus, since H has more edges of M' than of M, some component of the form L_n has this property. This L_n must be a chain of the form (12.5) with $e_1, e_3, \ldots, e_t$ in M' and not in M, and $e_2, e_4, \ldots, e_{t-1}$ in M and not in M'. Moreover, u_1 and u_{t+1} cannot be M-saturated, for otherwise L_n would not be a component of H. Hence, the chain L_n is an M-augmenting chain. This completes the proof.

To illustrate this proof, consider the matchings M and M' of Figure 12.18. Then the graph H is also shown in that figure. The chain

$$g, \{g, h\}, h, \{h, i\}, i, \{i, j\}, j$$

is a simple chain of two M' edges and one M edge.

12.5.3 An Algorithm for Finding a Maximum Matching

We next describe an algorithm for finding a maximum matching. This algorithm originates in the work of L. R. Ford and D. R. Fulkerson (see Ford and Fulkerson [1962]). The algorithm has two basic subroutines. Subroutine 1 searches for an M-augmenting chain starting with an unsaturated vertex x, and subroutine 2 builds a larger matching M' if subroutine 1 finds an M-augmenting chain. The algorithm chooses an unsaturated vertex x and applies subroutine 1. If an M-augmenting chain starting with x is found, subroutine 2 is called. If no such chain is found, another unsaturated vertex y is used in subroutine 1. The procedure is repeated until either an M-augmenting chain is found or no unsaturated vertices remain. In the latter case, we conclude that there is no M-augmenting chain, so we have a maximum matching.

Subroutine 2 works exactly as described in Theorem 12.6. We now present subroutine 1. It is easiest to describe this subroutine for the case of a bipartite graph G. If G is not bipartite, the procedure is more complicated. It was Edmonds [1965a] who first observed that something much more subtle was needed here (see Exercise 16). For more details on bipartite and nonbipartite matching, see, for example, Ahuja, Magnanti, and Orlin [1993], Cook, *et al.* [1998], Lawler [1976], Lovász and Plummer [1986], Minieka [1978], or Papadimitriou and Steiglitz [1982].

Subroutine 1 begins with a matching M and a vertex x unsaturated in M and builds in stages a tree T called an *alternating tree*. The idea is that x is in T and all simple chains in T beginning at x are M-alternating chains. For instance, in Figure 12.15, if $x = a$, one such alternating tree consists of the edges $\{a,b\}, \{b,g\}, \{g,h\}$, and $\{g,f\}$. The vertices in the alternating tree T are called outer or inner. Vertex y is called *outer* if the unique simple chain[7] between x and y ends in an edge of M, and *inner* otherwise. The vertex x is called *outer*. If an alternating tree T with an unsaturated vertex $y \neq x$ is found, the unique simple chain between x and y is an M-augmenting chain. Vertices and edges are added to T until either such a chain is found or no more vertices can be added to T. In the latter case, there is no M-augmenting chain. We now present the subroutine in detail.

Algorithm 12.2: Subroutine 1: Searching for an M-Augmenting Chain Beginning with Vertex x

Input: A bipartite graph G, a matching M of G, and a vertex x unsaturated in M.

Output: An M-augmenting chain beginning at x or the message that no such chain exists.

Step 1. Set $T = \emptyset$ and $T' = \emptyset$. (T is the set of edges in the tree and T' the set of edges definitely not in the tree.) Set $O = \{x\}$ and $I = \emptyset$. (O is the set of outer vertices and I is the set of inner vertices.)

[7] We are using the result of Theorem 3.18—that in a tree, there is a unique simple chain between any pair of vertices.

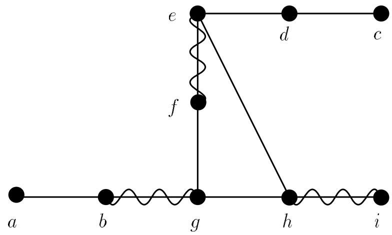


Figure 12.19: A matching M is shown by wiggly edges.

Step 2.

Step 2.1. Among the edges not in T or T', if there is no edge between an outer vertex (a vertex of O) and any other vertex, go to Step 3. Otherwise, let $\{u, v\}$ be such an edge with $u \in O$.

Step 2.2. If vertex v is an inner vertex (is in I), place edge $\{u, v\}$ in T' and repeat Step 2.1. If vertex v is neither inner nor outer, place edge $\{u, v\}$ in T, and go to Step 2.3. (Since G is bipartite, v cannot be an outer vertex, for otherwise one can show that G has an odd circuit, which is impossible for bipartite graphs by Theorem 3.4; see Exercise 11.)

Step 2.3. If v is unsaturated, stop. The unique chain in T from x to v forms an M-augmenting chain from x to v. If v is saturated, there is a unique edge $\{v, w\}$ in M. Then place $\{v, w\}$ in T, v in I, and w in O. Return to Step 2.1.

Step 3. We get to this step only when no further assignment of edges to T or T' is possible. Stop and give the message that there is no M-augmenting chain beginning with x.

Note that the algorithm stops in two ways, having found an M-augmenting chain or having found an alternating tree T where it is impossible to add edges to either T or T'. In the former case, the procedure goes to subroutine 2. In the latter case, one can show that there is no M-augmenting chain starting from the vertex x. We repeat this subroutine for another unsaturated vertex x. If in repeated applications of subroutine 1, we fail to find an M-augmenting chain beginning from an unsaturated vertex x, we conclude that the matching is maximum.

We illustrate the algorithm on the matching M of Figure 12.19. Pick unsaturated vertex x to be a, and call a an outer vertex. This is Step 1. Go to Step 2.1 and select the edge $\{a, b\}$ that is neither in T nor in T' and joins an outer vertex. Since b is not inner and not outer, in Step 2.2 we place this edge in T and go to Step 2.3. Since b is saturated, we consider the unique edge $\{b, g\}$ of M. We place this edge in T, call b inner and g outer, and return to Step 2.1. (See Table 12.3 for a summary. For the purposes of this summary, an iteration is considered a return to Step 2.1.)

In Step 2.1 we consider edges not in T and not in T' and joining an outer vertex. There are two such edges, $\{g, h\}$ and $\{g, f\}$. Let us suppose that we choose edge $\{g, f\}$. Since f is neither inner nor outer, in Step 2.2 we place $\{g, f\}$ in T. Now f

Table 12.3: Steps in Algorithm 12.2 Applied to M of Figure 12.19 and Starting with Vertex $x = a$

Iteration	T	T'	I (inner vertices)	O (outer vertices)
1	$\emptyset$	$\emptyset$	$\emptyset$	a
2	$\{a, b\}$	$\emptyset$	$\emptyset$	a
	$\{a, b\}, \{b, g\}$	$\emptyset$	b	a, g
3	$\{a, b\}, \{b, g\}, \{g, f\}$	$\emptyset$	b	a, g
	$\{a, b\}, \{b, g\}, \{g, f\}, \{f, e\}$	$\emptyset$	b, f	a, g, e
4	$\{a, b\}, \{b, g\}, \{g, f\}, \{f, e\}, \{g, h\}$	$\emptyset$	b, f	a, g, e
	$\{a, b\}, \{b, g\}, \{g, f\}, \{f, e\}, \{g, h\}, \{h, i\}$	$\emptyset$	b, f, h	a, g, e, i
5	$\{a, b\}, \{b, g\}, \{g, f\}, \{f, e\}, \{g, h\}, \{h, i\}$	$\{e.h\}$	b, f, h	a, g, e, i
6	$\{a, b\}, \{b, g\}, \{g, f\}, \{f, e\}, \{g, h\}, \{h, i\}, \{e, d\}$	$\{e, h\}$	b, f, h	a, g, e, i

is saturated, so we consider the unique edge $\{f, e\}$ of M. We place this edge in T, call f inner and e outer, and return to Step 2.1.

Next, we again consider edges not in T and not in T' and joining an outer vertex. The possible edges are $\{g, h\}, \{e, h\}$, and $\{e, d\}$. Suppose that we choose $\{g, h\}$. Then in Step 2.2, since h is neither inner nor outer, we place edge $\{g, h\}$ in T. Then in Step 2.3, since h is saturated, we place edge $\{h, i\}$ in T and call h inner and i outer.

We again consider edges not in T and not in T' and joining an outer vertex. The possible edges are $\{e, h\}$ and $\{e, d\}$. Suppose that we pick the former. Then vertex h is inner, so in Step 2.2 we place edge $\{e, h\}$ in T', and repeat Step 2.1.

In Step 2.1 we now choose edge $\{e, d\}$ and in Step 2.2 we add edge $\{e, d\}$ to T. Then in Step 2.3, since d is unsaturated, we stop and find the unique chain in T from x to v, that is, from a to d. This is the chain a, $\{a, b\}$, b, $\{b, g\}$, g, $\{g, f\}$, f, $\{f, e\}$, e, $\{e, d\}$, d. It is an M-augmenting chain.

In closing, we note that it can be shown that the algorithm described can be modified to take on the order of $[\min\{|X|, |Y|\}] \cdot |E|$ steps, given a bipartite graph $G = (X, Y, E)$. Thus, it is a polynomial algorithm in the number of vertices $n = |V|$. For $|E| \leq \binom{n}{2} \leq n^2$ and min $\{|X|, |Y|\} \leq n$. Thus, the algorithm concludes in a number of steps that is on the order of n^3. In the notation of Section 2.18, we say it is an $O(n^3)$ algorithm. A related algorithm for arbitrary G also takes on the order of n^3 steps. As of this writing, the fastest known algorithms for finding a maximum matching take on the order of $|E||V|^{1/2}$ steps. In terms of n, these algorithms take on the order of $n^{5/2}$ steps. They are due to Hopcroft and Karp [1973] for the bipartite case and to Micali and Vazirani [1980] for the general case. These algorithms relate matching to network flows. (We discuss this relation in Section 13.3.8.) For a more detailed discussion of the complexity for matching algorithms, see Ahuja, Magnanti, and Orlin [1993], Cook, *et al.* [1998], Lawler

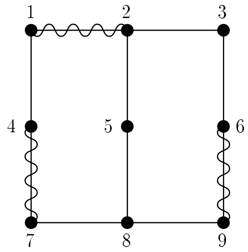


Figure 12.20: Matching for exercises of Section 12.5.

[1976], or Papadimitriou and Steiglitz [1982].

EXERCISES FOR SECTION 12.5

1. In the matching of Figure 12.20:
   (a) Find an M-alternating chain that is not M-augmenting.
   (b) Find an M-augmenting chain if one exists.
   (c) Use the chain in part (b), if it exists, to find a larger matching.
2. Repeat Exercise 1 for the matching of Figure 12.21.
3. Repeat Exercise 1 for the matching of Figure 12.22.
4. Repeat Exercise 1 for the matching of Figure 12.23.
5. Repeat Exercise 1 for the matching of Figure 12.24.
6. Apply subroutine 1 to:
   (a) The matching M of Figure 12.20 and the vertex 5
   (b) The matching M of Figure 12.20 and the vertex 3
   (c) The matching M of Figure 12.21 and the vertex 12
7. Apply subroutine 1 to the matching M of Figure 12.24 by starting with vertex 1 and, in successive iterations, choosing $v = 2, v = 4, v = 6, v = 4, v = 8$.
8. Show that if subroutine 1 is applied to the matching of Figure 12.23, starting with the vertex 3, it is possible for the edge $\{u, v\}$ chosen at Step 2.1 to join two outer vertices. Why can this happen?
9. Find an alternating tree starting at vertex x in the following situations.
   (a) $x = 6$ and the matching of Figure 12.21
   (b) $x = 8$ and the matching of Figure 12.20
10. Show that in the proof of Theorem 12.7, each component of H is either Z_n or L_n.
11. Prove that in Step 2.2 of Algorithm 12.2, v cannot be an outer vertex.
12. When applying subroutine 2, show that once a vertex is saturated, it stays saturated.
13. Prove that Algorithm 12.2 works.

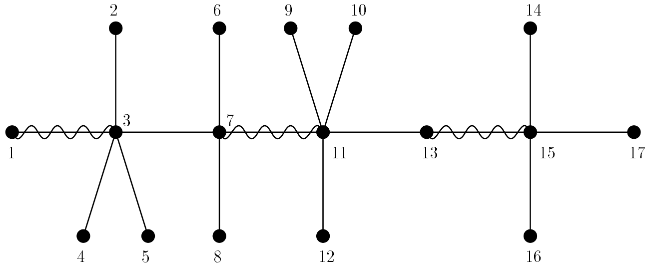


Figure 12.21: Matching for exercises of Section 12.5.

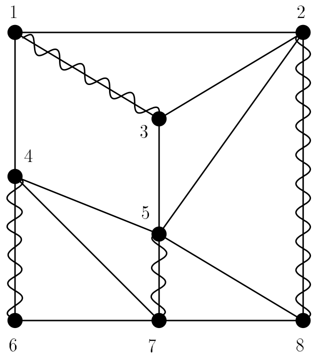


Figure 12.22: Matching for exercises of Section 12.5.

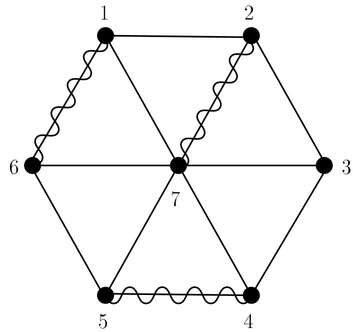


Figure 12.23: Matching for exercises of Section 12.5.

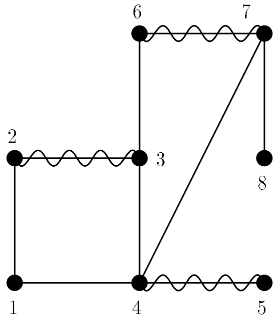


Figure 12.24: Matching for exercises of Section 12.5.

14. If a vertex x is unsaturated in a matching M and there is no M-augmenting chain starting at x, show that there is a maximum-cardinality matching in which x is unsaturated.
15. A subset S of vertices is called *matchable* if every vertex in S is matched in some matching. Show that if S is matchable, every vertex in S is matched in some maximum-cardinality matching.
16. Suppose that M is a matching in a graph G. A *blossom* relative to M is an odd-length circuit B of $2k+1$ vertices having k edges in M. Show that if there are no blossoms relative to M, subroutine 1 finds an M-augmenting chain beginning at x if there is one. (Thus, the algorithm we have described for finding a maximum matching must be modified only if blossoms are found. The modification due to Edmonds [1965a] works by searching for blossoms, shrinking them to single vertices, and searching for M-augmenting chains in the resulting graph.)

12.6 MATCHING AS MANY ELEMENTS OF X AS POSSIBLE

Suppose that $G = (X, Y, E)$ is a bipartite graph. If there is no X-saturating matching in G, we may at least wish to find a matching that matches as large a number of elements of X as possible. Let $m(G)$ be the largest number of elements of X that can be matched in a matching of G. We show how to compute $m(G)$. First, we mention a couple of applications of this idea.

Example 12.17 Telephone Switching Networks At a telephone switching station, phone calls are routed through incoming lines to outgoing trunk lines. A *switching network* connects each incoming line to some of the outgoing lines. When a call comes in on an incoming line, it is routed through the switching network to an outgoing line. The switching network can be represented by a bipartite graph $G = (X, Y, E)$. The vertices of X are the incoming lines, the vertices of Y are the outgoing lines, and there is an edge between an incoming line and an outgoing line if and only if the former is connected to the latter. Suppose that a number of incoming calls come in at once. We want to be able to send all of them out at the same time if possible. If S is the set of incoming lines on which there are calls, we want an assignment of each line of S to an edge of G leading to a different outgoing trunk line. That is, we want an S-saturating matching of G. If this is impossible to find, we want to match a large number of lines in S. In general, we want to design our switching network so that if calls come in on all possible incoming lines, we can match a large number of them with outgoing lines. That is, we want to design a switching network so that it has a maximum matching that has a large number of edges. Put another way, we want to find a bipartite graph $G = (X, Y, E)$ with given X and Y so that $m(G)$ is large enough to be reasonable. ■

Example 12.18 Smallpox Vaccinations (Example 12.3 Revisited) In planning ahead for a public health emergency such as a smallpox outbreak, we want to

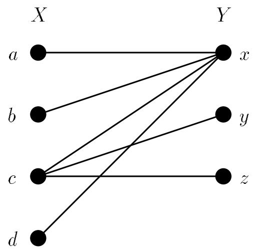


Figure 12.25: A graph G with $\delta(G) = 2$.

be sure that we have enough facilities for vaccinating people. However, the vaccination facilities might not have enough capacity to vaccinate everyone, at least not in a short time. Suppose that each facility can handle at most a certain number of people a day. We want to arrange assignments of people to facilities so that we can vaccinate as many people as possible in the first day. In other words, we want to make the assignment of people to facilities, defining a graph G, so that $m(G)$ is as large as possible. ■

If $S \subseteqq X$, let the *deficiency* of S, $\delta(S)$, be defined as $|S| - |N(S)|$, and let the *deficiency* of G, $\delta(G)$, be $\max_{S \subseteqq X} \delta(S)$. Note that $\delta(\emptyset) = 0$, so by Philip Hall's Theorem, G has an X-saturating matching if and only if $\delta(G) = 0$.

Theorem 12.8 (König [1931]) If $G = (X, Y, E)$ is a bipartite graph, $m(G) = |X| - \delta(G)$.

To illustrate this theorem, consider the graph of Figure 12.25. Note that $\delta(\{a, b, d\}) = 2$ and this is maximum, so $\delta(G) = 2$. Thus, $m(G) = |X| - \delta(G) = 4 - 2 = 2$. The largest subset of X that can be matched has two elements. An example of such a set is $\{a, c\}$.

A proof of Theorem 12.8 is sketched in Exercises 6 and 7.

EXERCISES FOR SECTION 12.6

1. Compute $\delta(G)$ and $m(G)$ for each graph of Figure 12.12.
2. Consider a switching station with 9 incoming lines and 6 outgoing trunk lines. Suppose that by engineering considerations, each incoming line is to be connected in the switching network to exactly two outgoing lines, and each outgoing line can be connected to at most 3 incoming lines.
   (a) Consider any set S of p incoming lines. Show that in the switching network, the sum of the degrees of the vertices in S is at most the sum of the degrees of the vertices in $N(S)$.
   (b) Using the result in part (a), show that for any set S of p incoming lines, $N(S)$ has at least $\frac{2}{3}p$ elements.
   (c) Conclude that for any set S of p incoming lines, $\delta(S) \leq \frac{1}{3}p$.
   (d) Conclude that no matter how the switching network is designed, there is always a matching that matches at least 6 incoming calls.

3. Suppose that the situation of Exercise 2 is modified so that there are 12 incoming lines and 10 outgoing lines. If each incoming line is to be connected to exactly 3 outgoing lines and each outgoing line to at most 4 incoming lines, show that no matter how the switching network is designed, there is always a matching that matches at least 9 incoming calls.

4. Suppose that the situation of Exercise 2 is modified so that there are two kinds of incoming lines, 4 of type I and 4 of type II, and there are 8 outgoing lines, all of the same type. Assume that each type I incoming line is connected to exactly 3 outgoing trunk lines and each type II incoming line to exactly 6 outgoing trunk lines, and assume that each outgoing line is connected to at most 4 incoming lines. Show that no matter how the switching network is designed, there is always a matching that matches at least 7 incoming calls. Do so by considering a set S of p incoming lines of type I and q incoming lines of type II. Show that
$$\delta(S) \leq (p+q) - \frac{3p+6q}{4}.$$
Conclude that since $p \leq 4$ and $q \geq 0$, $\delta(G) \leq 1$.

5. Suppose that in a small town there are four smallpox vaccination facilities and 56 people. Additionally, each person will be vaccinated at one of two facilities of their choosing, but each facility has a daily capacity of 14 people. Define a bipartite graph G so that $m(G)$ corresponds to the maximum number of people that can be vaccinated in the town in a single day.

6. As a preliminary to the proof of Theorem 12.8, prove the following: A bipartite graph $G = (X, Y, E)$ has a matching of k elements if and only if for all subsets $S \subsetneqq X$, $|N(S)| \geq |S| + k - |X| = k - |X - S|$. (*Hint:* Modify G by adding $|X| - k$ vertices to Y and joining each new vertex of Y to each vertex of X.)

7. Use the result of Exercise 6 to prove Theorem 12.8.

8. This exercise presents an alternative proof of the König-Egerváry Theorem (Corollary 12.5.1). Let $\mathbf{A}$ be a matrix of 0's and 1's and build a bipartite graph $G = (X, Y, E)$ from $\mathbf{A}$ as in the proof of the König-Egerváry Theorem. Let S be a subset of X, the set of rows, such that $\delta(S) = \delta(G)$.

   (a) Show that the rows corresponding to vertices of $X - S$ together with the columns corresponding to vertices of $N(S)$ contain all the 0's in $\mathbf{A}$.

   (b) Show that the total number of rows and columns in part (a) is $|X| - \delta(G)$.

   (c) Conclude that the minimum number of lines of $\mathbf{A}$ covering all 0's is at most $|X| - \delta(G)$.

   (d) Show that the maximum number of independent 0's in G is $|X| - \delta(G)$.

   (e) Prove the König-Egerváry Theorem.

12.7 MAXIMUM-WEIGHT MATCHING

In Examples 12.5–12.9 of Section 12.1, we introduced the problem of finding a matching of maximum (or minimum) weight in a graph where we have weights on its edges. Here we discuss the problem in more detail, including an algorithm for solving it presented in the language of one of these problems.

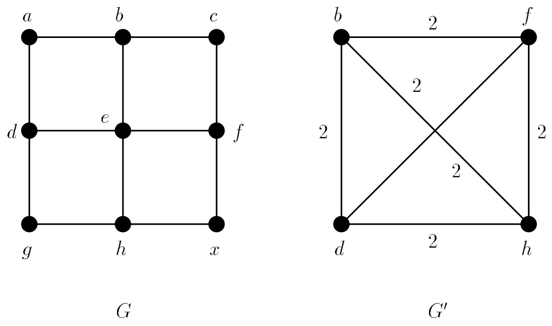


Figure 12.26: The graph G of Figure 11.28 and the corresponding graph G'.

12.7.1 The "Chinese Postman" Problem Revisited

In Section 11.4.1 we discussed the problem of a mail carrier who wishes to find the smallest number of blocks to walk, yet cover all the blocks on an assigned route. We formulated the problem as follows. Given a graph G, we study *feasible multigraphs* H, multigraphs that have an eulerian closed chain and are obtained from G by adding copies of edges of G. We seek an *optimal multigraph*, a feasible multigraph with a minimum number of edges. In Section 11.4.2 we observed that the same problem arises in connection with computer graph plotting. In Exercise 11, Section 11.4, the reader was asked to observe that in any optimal multigraph H, the newly added edges could be divided up into chains joining vertices of odd degree in G, with any such vertex an end vertex of exactly one such chain. Now suppose that for every pair of vertices u and v of odd degree in G, we find between them a shortest chain in G. [We can use an algorithm like Dijkstra's Algorithm (Section 13.2.2) to do this.] Let us build a graph G' by taking as vertices all odd-degree vertices in G, joining each pair of vertices by an edge, and putting on this edge a weight equal to the length of the shortest chain between u and v. To illustrate, suppose that G is the graph of Figure 11.28, that is repeated in Figure 12.26. Then there are four vertices of odd degree in G: b, f, h, and d. Between each pair of these, a shortest chain clearly has length 2. Hence, the graph G' is as shown in the second part of Figure 12.26. Now any optimal multigraph H obtained from G corresponds to a collection of chains joining pairs of odd-degree vertices in G, with each such vertex an end vertex of exactly one such chain. Such a collection of chains defines a perfect matching in the graph G'. Moreover, an optimum H corresponds to a collection of such chains, the sum of whose lengths is as small as possible. This collection in turn defines a perfect matching in G' of minimum weight. If G' is changed by taking the negative of each weight, we seek a maximum-weight matching in G'. (Such a matching will be perfect. Why?) This is a problem that we discussed briefly in Examples 12.5–12.9. In our example, a minimum-weight perfect matching in G' consists of any two edges that do not touch, such as, $\{b, d\}$ and $\{f, h\}$. Then the shortest chains corresponding to the chosen edges will define the optimal multigraph H. Here a shortest chain between b and d is b, e, d, and one between f and h is

f, x, h. If we add to G the edges on these chains, we get the first optimal multigraph H of Figure 11.28. This is now known to be the smallest number of edges that could be added to give an H that has an eulerian closed chain. The eulerian closed chain in H will give us an optimal mail carrier's route in G.

The approach we have described is due to Edmonds (see Edmonds [1965b] and Edmonds and Johnson [1973]). It should be pointed out that the procedure described is an efficient one. For an efficient shortest path algorithm (Dijkstra's Algorithm of Section 13.2.2) can be completed in a number of steps on the order of n^2 and the minimum- (maximum-) weight matching can be found in a number of steps on the order of n^3 (Lawler [1976]).

12.7.2 An Algorithm for the Optimal Assignment Problem (Maximum-Weight Matching)[8]

In Example 12.6 we discussed the following job assignment problem. There are n workers and m jobs, every worker is suited for every job, and worker i's potential performance on job j is given a rating r_{ij}. We wish to assign workers to jobs so as to maximize the sum of the performance ratings. This is an example of a maximum-weight matching problem. Let us assume that $m = n$ and let x_{ij} be a variable that is 1 if worker i is assigned to job j and 0 otherwise. Then, since $m = n$, we want to maximize

$$\sum_{\substack{i=1\\j=1}}^{\substack{m\\n}} r_{ij}x_{ij}.$$

We require that no two workers get the same job, that is, that

$$\sum_{i=1}^{n} x_{ij} \leq 1. \tag{12.8}$$

We also require that every worker get a job, that is, that

$$\sum_{j=1}^{n} x_{ij} \geq 1. \tag{12.9}$$

Then since the number of workers equals the number of jobs, (12.8) and (12.9) give us the constraints

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{and} \quad \sum_{j=1}^{n} x_{ij} = 1. \tag{12.10}$$

We now present an algorithm for solving the optimal assignment problem. Suppose that we are given an $n \times n$ matrix (c_{ij}) and we wish to find numbers $x_{ij} = 0$ or 1 such that we minimize $\sum_{i,j} c_{ij}x_{ij}$ and such that (12.10) holds. Note that if a

[8] This subsection may be deferred until Section 13.4.

constant p_k is subtracted from all entries in the kth row of (c_{ij}), giving rise to a matrix (c'_{ij}), then

$$\sum_{i,j} c'_{ij}x_{ij} = \sum_{i,j} c_{ij}x_{ij} - p_k \sum_j x_{kj} = \sum_{i,j} c_{ij}x_{ij} - p_k,$$

by (12.10). Thus, the assignment x_{ij} that minimizes $\sum_{i,j} c_{ij}x_{ij}$ will also minimize $\sum_{i,j} c'_{ij}x_{ij}$. The same is true if a constant q_l is subtracted from each entry in the lth column of (c_{ij}). Indeed, the same is true if we subtract constants p_i from the ith row of (c_{ij}) for all i and q_j from the jth column of (c_{ij}) for all j.

Using this observation, we let p_i be the minimum element in the ith row of (c_{ij}). For each i, we subtract p_i from each element in the ith row of (c_{ij}), obtaining a matrix (c'_{ij}). We now let q_j be the minimum element in the jth column of (c'_{ij}), and for each j, subtract q_j from each element in the jth column of (c'_{ij}), obtaining a matrix $(\bar{c}_{ij})$. This is called the *reduced matrix*. By what we have observed, we may as well solve the optimal assignment problem using the reduced matrix.

To illustrate this procedure, suppose that $n = 4$ and

$$(c_{ij}) = \begin{bmatrix} 12 & 14 & 15 & 14 \\ 9 & 6 & 11 & 8 \\ 10 & 9 & 16 & 14 \\ 12 & 13 & 13 & 10 \end{bmatrix}. \tag{12.11}$$

Then $p_1 = 12, p_2 = 6, p_3 = 9, p_4 = 10$, and

$$(c'_{ij}) = \begin{bmatrix} 0 & 2 & 3 & 2 \\ 3 & 0 & 5 & 2 \\ 1 & 0 & 7 & 5 \\ 2 & 3 & 3 & 0 \end{bmatrix}.$$

Now $q_1 = 0, q_2 = 0, q_3 = 3$, and $q_4 = 0$, so

$$(\bar{c}_{ij}) = \begin{bmatrix} 0 & 2 & 0 & 2 \\ 3 & 0 & 2 & 2 \\ 1 & 0 & 4 & 5 \\ 2 & 3 & 0 & 0 \end{bmatrix}. \tag{12.12}$$

Note that the reduced matrix $(\bar{c}_{ij})$ always has nonnegative entries. (Why?) A job assignment x_{ij} giving each worker one job and each job one worker corresponds to a choice of n entries of this matrix, one in each row and one in each column. We have $x_{ij} = 1$ if and only if the i, j entry is picked. Now suppose that we can find a choice of n entries, one in each row and one in each column, so that all the entries are 0. Let us take $x_{ij} = 1$ for precisely these n i, j entries and $x_{ij} = 0$ otherwise. Then the corresponding $\sum \bar{c}_{ij}x_{ij}$ will be 0, because when x_{ij} is 1, $\bar{c}_{ij} = 0$. For instance, if

$$(\bar{c}_{ij}) = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 5 & 4 \\ 0 & 3 & 0 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 2 & 0 & 2 \\ 3 & 0 & 2 & 2 \\ 1 & 0 & 4 & 5 \\ 2 & 3 & 0 & 0 \end{bmatrix}$$

Figure 12.27: A minimum covering of 0's by lines.

we can pick the $1, 2$, the $2, 1$, and the $3, 3$ entries. If we take $x_{12} = 1, x_{21} = 1, x_{33} = 1$, and $x_{ij} = 0$ otherwise, then $\sum \bar{c}_{ij}x_{ij} = 0$. This is clearly a minimum, since $\sum \bar{c}_{ij}x_{ij} \geq 0$ because $\bar{c}_{ij} \geq 0$.

Recall from Section 12.4.1 that an *independent set of* 0's in a matrix is a collection of 0's no two of which are in the some row and no two of which are in the same column. What we have just observed is that if we can find an independent set of n 0's in $(\bar{c}_{ij})$, we can find an optimal job assignment by taking the corresponding x_{ij} to be 1. Since there can be no independent set of more than n 0's in $(\bar{c}_{ij})$, we look for a maximum independent set of 0's. How do we find such a set? Let us change all positive entries in $(\bar{c}_{ij})$ to 1. Then since $\bar{c}_{ij}$ is nonnegative, the resulting matrix is a matrix of 0's and 1's. Recall from Section 12.4.1 that a *line* of a matrix is either a row or a column. Then by the König-Egerváry Theorem (Corollary 12.5.1), the maximum number of independent 0's in $(\bar{c}_{ij})$, equivalently in its modified matrix of 0's and 1's, is equal to the minimum number of lines that cover all the 0's. Thus, we can check to see if there is a set of n independent 0's by checking to see if there is a set of n lines that cover all the 0's. Alternatively, from the proof of the König-Egerváry Theorem, we recall that a maximum independent set of 0's in the matrix $(\bar{c}_{ij})$, equivalently in the modified matrix of 0's and 1's, corresponds to a maximum matching in the bipartite graph $G = (X, Y, E)$, where $X = Y = \{1, 2, \ldots, n\}$ and where there is an edge between i in X and j in Y iff $\bar{c}_{ij} = 0$. Thus, we can apply the maximum matching algorithm of Section 12.5.3 to see if there is a set of n independent 0's.

In our example of Equation (12.11), we have derived $(\bar{c}_{ij})$ in Equation (12.12). Note that the minimum number of lines covering all the 0's in $(\bar{c}_{ij})$ is 3: Use the first and fourth rows and the second column (see Figure 12.27). Thus, the maximum independent set of 0's has three elements, not enough to give us a job assignment, as we need four independent 0's.

The algorithm for solving the optimal assignment problem proceeds by successively modifying the reduced matrix $(\bar{c}_{ij})$ so that eventually we obtain one where we can find n independent 0's. The modification step is to use the minimum covering of 0's by lines, such as shown in Figure 12.27, and find the smallest uncovered element. Then subtract this from each uncovered element and add it to each twice-covered element and find a new matrix in which to search for independent 0's. In our example, the smallest uncovered element is 1, the $3, 1$ element. We subtract 1 from the 2, 1, the 2, 3, the 2, 4, the 3, 1, the 3, 3, and the 3, 4 entries of $(\bar{c}_{ij})$, and add it to the 1, 2 and the 4, 2 entries, obtaining the new reduced matrix

$$\begin{bmatrix} 0 & 3 & 0 & 2 \\ 2 & 0 & 1 & 1 \\ 0 & 0 & 3 & 4 \\ 2 & 4 & 0 & 0 \end{bmatrix}. \tag{12.13}$$

It is not hard to show that the new reduced matrix has been obtained from the preceding one by adding or subtracting a constant from different rows or columns (Exercise 8). Thus, solving the optimal assignment problem with this matrix is the same as solving it with the previous reduced matrix. Also, the new reduced matrix has all entries nonnegative. Thus, once again with this new matrix, we can seek an independent set of n 0's. We seek this by finding a minimum covering by lines. In (12.13), a minimum covering uses four lines, and hence there must be an independent set of four 0's. One such set consists of the 1, 3 entry, the 2, 2 entry, the 3, 1 entry, and the 4, 4 entry. Hence, we can find an optimal assignment by letting $x_{13} = 1, x_{22} = 1, x_{31} = 1, x_{44} = 1$, and all other $x_{ij} = 0$. If we had not found a set of n independent 0's, we would have repeated the modification of the reduced matrix.

The algorithm we have described, called the *Hungarian Algorithm*, is due to Kuhn [1955]. We summarize it as follows.

Algorithm 12.3: The Hungarian Algorithm for the Optimal Assignment Problem

Input: An $n \times n$ matrix (c_{ij}).
Output: An optimal assignment x_{ij}.

Step 1. (*Initialization*)

Step 1.1. For each i, let p_i be the minimum element in the ith row of (c_{ij}).

Step 1.2. Let (c'_{ij}) be computed from (c_{ij}) by subtracting p_i from each element of the ith row, for all i.

Step 1.3. For each j, let q_j be the minimum element in the jth column of (c'_{ij}).

Step 1.4. Let $(\bar{c}_{ij})$ be computed from (c'_{ij}) by subtracting q_j from each element of the jth column, for all j.

Step 2.

Step 2.1. Find a minimum collection of lines covering the 0's of $(\bar{c}_{ij})$.

Step 2.2. If this collection of lines has fewer than n elements, go to Step 3. Otherwise, go to Step 4.

Step 3. (*Modification of Reduced Matrix*)

Step 3.1. Using the covering obtained in Step 2.1, let p be the smallest uncovered element in $(\bar{c}_{ij})$.

Step 3.2. Change the reduced matrix $(\bar{c}_{ij})$ by subtracting p from each uncovered element and adding p to each twice-covered element. Return to Step 2.

Step 4.

Step 4.1. Find a set of n independent 0's in $(\bar{c}_{ij})$.

Step 4.2. Let x_{ij} be 1 if the i, j entry of $(\bar{c}_{ij})$ is one of the independent 0's, and let x_{ij} be 0 otherwise. Output this solution x_{ij} and stop.

Theorem 12.9 If all of the c_{ij} are integers, the Hungarian Algorithm gives an optimal assignment.

Proof.[9] We have already observed that if the algorithm gives an assignment x_{ij}, this must be optimal. But how do we know that the algorithm will ever give an assignment? The reason it does is because the reduced matrix always has nonnegative integer entries and because at each modification, the sum of the entries in this matrix decreases by an integer at least 1, as we show below. Thus, in a finite number of steps, if we have not yet reached an optimal solution, all the entries of the reduced matrix will be 0. In this case, there is, of course, a collection of n independent 0's, so we find an optimal solution.

We now show that if $(\bar{d}_{ij})$ is the modified reduced matrix obtained in Step 3 from the reduced matrix $(\bar{c}_{ij})$, then

$$\sum_{i,j}(\bar{c}_{ij}) - \sum_{i,j}(\bar{d}_{ij}) = \text{an integer} \geq 1.$$

Recall that we have a covering of $(\bar{c}_{ij})$ by $k < n$ lines. Let S_r be the set of uncovered rows of $(\bar{c}_{ij})$, S_c be the set of uncovered columns, $\bar{S}_r$ be the set of covered rows, and $\bar{S}_c$ the set of covered columns. Let $\alpha = |S_r|$ and $\beta = |S_c|$. Then $k = (n-\alpha)+(n-\beta)$. Also, recall that p is the smallest uncovered entry of $(\bar{c}_{ij})$. Then, remembering how we get $(\bar{d}_{ij})$ from $(\bar{c}_{ij})$, we have

$$\begin{aligned}
\textstyle\sum_{i,j}\bar{c}_{ij} - \sum_{i,j}\bar{d}_{ij} &= \textstyle\sum_{\substack{i\in S_r\\ j\in S_c}}[\bar{c}_{ij}-\bar{d}_{ij}] + \sum_{\substack{i\in S_r\\ j\in \bar{S}_c}}[\bar{c}_{ij}-\bar{d}_{ij}]\\
&\quad \textstyle+ \sum_{\substack{i\in \bar{S}_r\\ j\in S_c}}[\bar{c}_{ij}-\bar{d}_{ij}] + \sum_{\substack{i\in \bar{S}_r\\ j\in \bar{S}_c}}[\bar{c}_{ij}-\bar{d}_{ij}]\\
&= \textstyle\sum_{\substack{i\in S_r\\ j\in S_c}} p + \sum_{\substack{i\in S_r\\ j\in \bar{S}_c}} 0 + \sum_{\substack{i\in \bar{S}_r\\ j\in S_c}} 0 + \sum_{\substack{i\in \bar{S}_r\\ j\in \bar{S}_c}} (-p)\\
&= \alpha\beta p - (n-\alpha)(n-\beta)p\\
&= n(\alpha+\beta-n)p.
\end{aligned}$$

But $\alpha + \beta$ is the number of uncovered rows and columns, so

$$\alpha + \beta - n = (2n-k) - n = n - k > 0,$$

since $k = (n-\alpha) + (n-\beta)$ and $k < n$. Thus,

$$\sum_{i,j}\bar{c}_{ij} - \sum_{i,j}\bar{d}_{ij} = n(\alpha+\beta-n)p = \text{an integer} \geq 1,$$

[9] The proof may be omitted.

since $n, \alpha + \beta - n$, and p are all positive integers. Q.E.D.

We close by observing that if there are n workers and n jobs, the Hungarian Algorithm can be implemented in $O(n^3)$ time, to use the terminology of Section 2.18. For a discussion of this point, see Papadimitriou and Steiglitz [1982].

EXERCISES FOR SECTION 12.7

1. For each graph G of Figure 11.39, find the corresponding graph G' as described in the solution to the "Chinese Postman" Problem in Section 12.7.1. Find a minimum-weight perfect matching in G' and translate this into a solution to the "Chinese Postman" Problem.
2. Each of the following matrices give the cost c_{ij} of using worker i on job j. Use the Hungarian Algorithm to find a minimum-cost job assignment.

$$\text{(a)}\quad \begin{array}{c} \\ 1\\2\\3\\4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 8 & 3 & 2 & 4 \\ 10 & 9 & 3 & 6 \\ 2 & 1 & 1 & 5 \\ 3 & 8 & 2 & 1 \end{pmatrix} \end{array} \qquad \text{(b)}\quad \begin{array}{c} \\ 1\\2\\3\\4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 17 & 5 & 8 & 11 \\ 3 & 9 & 2 & 10 \\ 4 & 2 & 8 & 6 \\ 7 & 6 & 4 & 5 \end{pmatrix} \end{array} \qquad \text{(c)}\quad \begin{array}{c} \\ 1\\2\\3\\4\\5 \end{array}\!\!\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 8 & 7 & 5 & 11 & 4 \\ 9 & 7 & 6 & 11 & 3 \\ 12 & 9 & 4 & 8 & 2 \\ 1 & 2 & 3 & 5 & 6 \\ 11 & 4 & 2 & 8 & 2 \end{pmatrix} \end{array}$$

3. The following matrix gives the rating r_{ij} of worker i on job j. Use the Hungarian Algorithm to find an optimum (i.e., highest rated) job assignment.

$$\begin{array}{c} \\ 1\\2\\3\\4\\5 \end{array}\!\!\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 8 & 7 & 5 & 9 & 6 \\ 11 & 9 & 7 & 4 & 8 \\ 12 & 6 & 7 & 5 & 10 \\ 9 & 7 & 6 & 9 & 6 \\ 3 & 9 & 8 & 9 & 8 \end{pmatrix} \end{array}$$

4. A company has purchased five new machines of different types. In its factory, there are five locations where the machines can be located. For a given machine, its location in a particular spot would have an effect on the ease of handling materials. For instance, if the machine is near the work center that produces materials for it, this would be efficient. For machine i at location j, the hourly cost of handling materials to be brought to the machine can be estimated. The following matrix gives this information.

$$\text{Machine}\ \begin{array}{c} \\ \\ 1\\2\\3\\4\\5 \end{array}\!\!\begin{array}{c} \text{Location} \\ \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 4 & 6 & 8 & 5 & 7 \\ 2 & 4 & 6 & 9 & 5 \\ 1 & 7 & 6 & 8 & 3 \\ 4 & 6 & 7 & 5 & 7 \\ 10 & 4 & 5 & 4 & 5 \end{pmatrix} \end{array}$$

How would we assign machines to locations so as to minimize the resulting materials handling costs?

5. The following matrix gives the frequency response f_{ij} when speakers i and j are paired (see Example 12.8).

   (a) Use the Hungarian Algorithm to find a minimum-frequency response pairing.

$$\begin{array}{c c} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} 0 & 4 & 8 & 8 & 7 & 8 \\ 4 & 0 & 6 & 8 & 1 & 6 \\ 8 & 6 & 0 & 5 & 7 & 4 \\ 8 & 8 & 5 & 0 & 4 & 2 \\ 7 & 1 & 7 & 4 & 0 & 7 \\ 8 & 6 & 4 & 2 & 7 & 0 \end{pmatrix} \end{array}$$

   (b) If all entries in a single row were the same (except for the diagonal entry, which must be 0), how would this affect your use of the Hungarian Algorithm?

6. Recall the real estate transactions problem of Example 12.5 (Section 12.1). Suppose that an agent has three listed houses, H_1, H_2, and H_3, and four buyers, B_1, B_2, B_3, and B_4. She stands to receive 20, 18, and 8 thousand dollars if B_1 buys H_1, H_2, or H_3, respectively. She also stands to receive 20, 12, and 16 if B_2 buys a house; 18, 20, and 20 if B_3 buys a house; and 16, 18, and 16 if B_4 buys a house. Is there a unique solution to her problem of maximizing profits? (*Hint:* Introduce a "dummy" fourth house.)

7. (a) Suppose that certain people have to be assigned to certain facilities in the smallpox vaccination problem of Example 12.7 (Section 12.1). Explain how you would accommodate this requirement. In particular, what changes would you make to the matrix input for the Hungarian Algorithm?

   (b) Now suppose that certain people cannot be assigned to certain facilities in the smallpox vaccination problem of Example 12.7 (Section 12.1). Explain how you would accommodate this requirement. In particular, what changes would you make to the matrix input for the Hungarian Algorithm?

8. Show that in the Hungarian Algorithm, the new reduced matrix is obtained from the old by adding or subtracting a constant from different rows or columns.

12.8 STABLE MATCHINGS

In Example 12.12 we introduced the problem of how to assign medical school graduates to internships in hospitals. In 1952, an algorithm called the National Intern Matching Program was developed to make such an assignment. The idea was that interns would submit a ranking of the hospitals to which they had applied and the hospitals would similarly rank their applicants. Essentially unchanged, the algorithm now called the National Resident Matching Program is still in use today. It is aimed at producing a "stable" matching.

The idea of combining preferences with matchings to form a stable matching was formally introduced by Gale and Shapley [1962]. We discussed the idea briefly in Section 4.2.5. This work has led to numerous theoretical extensions as well as further practical applications. See Gusfield and Irving [1989] and Roth and Sotomayor

Table 12.4: Preference Orderings for a Size 4 Stable Marriage Problem

Men's Preferences				Women's Preferences			
m_1	m_2	m_3	m_4	w_1	w_2	w_3	w_4
w_1	w_2	w_3	w_4	m_4	m_3	m_2	m_1
w_2	w_1	w_4	w_3	m_3	m_4	m_1	m_2
w_3	w_4	w_1	w_2	m_2	m_1	m_4	m_3
w_4	w_3	w_2	w_1	m_1	m_2	m_3	m_4

[1990] for extended discussions. (For another approach to stability, Balinski and Ratier [1997] redeveloped the idea of stable matchings in the language of directed graphs.)

One particular type of stable matching introduced by Gale and Shapley [1962] solves a stable marriage problem. This *stable marriage problem of size* n involves a set of n men and a set of n women. Each person supplies a (strict) preference ordering of all members of the opposite sex. If monogamous marriages are set up for each person, this set of marriages will be called a matching, i.e., a one-to-one correspondence between the men and women. A matching is called *unstable* if some man and woman are each willing, based on their preference ordering, to leave their respective partners for each other. Such a man–woman pair will be called a *blocking pair*. If no blocking pair exists, the matching is called *stable*.

In this section we consider the existence problem, the counting problem, and the optimization problem as they relate to the stable marriage problem. The following example highlights these points.

Example 12.19 Stable Marriages[10] Suppose that four men and four women are to be married to each other. Their preference orderings are given in Table 12.4. We denote the marriage between m_i and w_j by $m_i - w_j$. Note that $M = \{m_1 - w_4, m_2 - w_3, m_3 - w_1, m_4 - w_2\}$ is a stable set of marriages. To see why, note that w_4 and w_3 married their first choice; so neither would be willing to leave their partner, m_1 and m_2, respectively. Also, m_3 and m_4 are doing their best with the remaining women, that is, w_1 and w_2, respectively.

While M is stable, there are actually nine other stable matchings associated with these people and their preferences. This is out of a total of $4! = 24$ matchings. (Why?) $M_u = \{m_1 - w_1, m_2 - w_3, m_3 - w_4, m_4 - w_2\}$ is one example of an unstable matching. To see why, note that m_2, although married to w_3, actually prefers w_4 (as well as w_1 and w_2). And w_4, married to m_3, actually prefers m_2 (and m_1). Since m_2 and w_4 prefer each other to their respective partners, they form a blocking pair and hence M_u is unstable.

Two other obvious stable matchings are $M_m = \{m_1 - w_1, m_2 - w_2, m_3 - w_3, m_4 - w_4\}$ and $M_w = \{m_1 - w_4, m_2 - w_3, m_3 - w_2, m_4 - w_1\}$. These are both stable since each man in M_m and each woman in M_w got their first choice. Also, because of

[10] This example is from Gusfield and Irving [1989].

this fact, note that any man would prefer M_m and any woman would prefer M_w to any of the other stable matchings. ■

12.8.1 Gale-Shapley Algorithm

Does a stable set of marriages exist for every stable marriage problem? This existence question was settled positively by Gale and Shapley [1962]. We describe the Gale-Shapley (GS) Algorithm for finding a stable matching. The GS Algorithm allows for interim "engagements" until its final stage, when all engagements are finalized in marriage. A stage is a two-step process where all free (unengaged) men propose and then all women with proposals become engaged.

The GS Algorithm starts with each man proposing to the number 1 woman on his preference list. Each woman then becomes engaged to the proposer who is highest on her preference list. Stage 2 has all remaining free men proposing to their number 2 woman and then each woman becoming engaged to the best of either her new proposal(s) or her fiancé, if one exists. (Note that at any stage, some women may not be making any decision, as they may not have been proposed to in that stage.)

Each subsequent stage proceeds in the same manner. All free men propose to the highest woman on their preference list to whom they haven't yet proposed, and each woman compares her new proposal(s) and fiancé for the best choice and enters into her new engagement. The GS Algorithm terminates when all women (men) are engaged.

Is the matching produced by the GS Algorithm stable? Suppose that m_i and w_j are not married but m_i prefers w_j to his wife w_k. It must be the case that m_i proposed to w_j before w_k. However, when w_j received m_i's proposal, either she rejected it in favor of someone whom she preferred or became engaged to m_i and later rejected him in favor of a better proposal. In either case, w_j prefers her husband to m_i and the matching produced by the GS Algorithm must be stable.

Note that the GS Algorithm could also be carried out with the women doing the proposing, also producing a stable matching. This stable matching could be different from that obtained using the "man proposing" version.

To compute the complexity of the GS Algorithm, we note that at each step, a woman who receives a proposal either (i) receives a proposal for the first time or (ii) rejects a new proposal or breaks an engagement. She does (i) exactly once and (ii) at most $n-1$ times. Thus, the algorithm has $O(n)$ steps for each woman and therefore $O(n^2)$ steps in all.

One final note about the GS Algorithm. We described a stage by saying that all free men propose to the next-highest woman on their preference list; however, no order in which the men should do so is given. Gusfield and Irving [1989] have shown that a particular ordering is inconsequential to the outcome of the GS Algorithm.

Theorem 12.10 (Gusfield and Irving [1989]) All possible executions of the GS Algorithm yield the same stable matching.

To see the GS Algorithm in action, consider the preference lists for the four

Table 12.5: Preference Orderings for a Size 4 Stable Marriage Problem

Men's Preferences				Women's Preferences			
m_1	m_2	m_3	m_4	w_1	w_2	w_3	w_4
w_1	w_2	w_3	w_2	m_4	m_3	m_2	m_1
w_4	w_1	w_4	w_3	m_3	m_4	m_1	m_2
w_3	w_4	w_1	w_4	m_2	m_1	m_4	m_3
w_2	w_3	w_2	w_1	m_1	m_2	m_3	m_4

Table 12.6: Preference Orderings for a Size 2 Stable Marriage Problem

Men's Preferences		Women's Preferences	
m_1	m_2	w_1	w_2
w_1	w_2	m_2	m_1
w_2	w_1	m_1	m_2

men and four women in Table 12.5. The GS Algorithm starts with m_1, m_2, m_3, m_4 proposing to w_1, w_2, w_3, w_2, respectively. The engagements resulting from these proposals are $m_1 - w_1, m_3 - w_3$, and $m_4 - w_2$. (Note that w_2 chose m_4 since he was higher on her list than m_2.) The second stage has m_2 proposing to his second choice, w_1. She accepts his proposal, switching engagements from m_1, since m_2 is higher on her list than m_1. The third stage has m_1 proposing to his second choice, w_4. She accepts his proposal and they become engaged. At this point, all women (and men) are engaged, so the GS Algorithm stops with all engagements becoming (stable) marriages.

Although the GS Algorithm assures the existence of at least one stable matching, there are instances of stable marriage problems with exactly one stable matching. That is, the GS Algorithm produces the only stable matching (see Exercise 6). In the next section we consider the maximum number of stable matchings that are possible over all stable marriage problems of size n.

12.8.2 Numbers of Stable Matchings

In any stable marriage problem of size 2, there are a total of $2! = 2$ matchings. By the GS Algorithm, there is always at least one stable matching. The next example shows that all (both) of the matchings could be stable. Consider the stable marriage problem of size 2 with the preference lists of Table 12.6. It is easy to check that $M_m = \{m_1 - w_1, m_2 - w_2\}$ and $M_w = \{m_1 - w_2, m_2 - w_1\}$ are both stable. Note that M_m is the output of the GS Algorithm if the men do the proposing, while M_w is the output if the women do the proposing.

In Example 12.19 we said that there were 10 stable matchings for this problem out of a total of 24 matchings. For small examples it is not too difficult to enumerate all the matchings and to check individually if each one is stable. (Exercise 7 asks the reader to devise an algorithm to test a matching for stability.) However, for larger problems, enumeration is not possible since stable marriage problems of size n have $n!$ matchings. The next theorem gives a lower bound for the number of sets of stable marriages for problems of size n. (Finding the largest possible number of sets of stable marriages for problems of size n is an open problem; see Knuth [1997].)

Theorem 12.11 (Irving and Leather [1986]) For each $n \geq 1$, n a power of 2, there is a stable marriage problem of size n with at least 2^{n-1} stable matchings.

Proof. We will prove this by induction on n. For $n = 1 = 2^0$, $2^{n-1} = 2^0 = 1$, and by the GS Algorithm there must be (at least) one stable matching in any stable marriage problem, including size 1. For $n = 2 = 2^1$, the problem described in Table 12.6 gives a stable marriage problem of size 2 with $2 = 2^{2-1}$ stable matchings. This instance of a stable marriage problem will be important for this proof. Assume that the theorem is true for $n = 2^k$. That is, assume that there is a stable marriage problem of size $n = 2^k$ with at least $2^{n-1} = 2^{2^k-1}$ stable matchings. Suppose that the men and women in this case are labeled $b_1, b_2, \ldots, b_{2^k}$ and $g_1, g_2, \ldots, g_{2^k}$, respectively.

Recall that m_1, m_2 and w_1, w_2 are the men and women, respectively, from the stable marriage problem of size 2 in Table 12.6. To finish this proof, we need to create a new stable marriage problem that will have size 2^{k+1} and at least $2^{2^{k+1}-1}$ stable matchings.

Consider the stable matching problem of size 2^{k+1} with

$$\text{men} : (b_1, m_1), \ldots, (b_{2^k}, m_1), (b_1, m_2), \ldots, (b_{2^k}, m_2)$$

and

$$\text{women} : (g_1, w_1), \ldots, (g_{2^k}, w_1), (g_1, w_2), \ldots, (g_{2^k}, w_2).$$

The preference listings for each man will be as follows:

(b_i, m_j) prefers (g_k, w_l) to $(g_{k'}, w_{l'})$ if m_j prefers w_l to $w_{l'}$, or if $l = l'$ and b_i prefers g_k to $g_{k'}$.

The preference listings for the women are defined similarly. It now suffices to show that there are at least $2^{2^{k+1}-1}$ stable matchings for this problem.

Let $M_1, M_2, \ldots, M_n$ be any sequence of (not necessarily distinct) stable matchings from the stable marriage problem of Table 12.6; note that there are two choices for each M_i. Let M be any one of the (at least) 2^{2^k-1} stable matchings from the size $n = 2^k$ problem from above. Thus, the total number of possible choices available for these $n+1$ stable matchings, $M_1, M_2, \ldots, M_n, M$, is at least $2 \cdot 2 \cdot 2 \cdot \cdots \cdot 2 \cdot 2^{2^k-1}$, which equals $2^n \cdot 2^{2^k-1} = 2^{2^k} \cdot 2^{2^k-1} = 2^{2^{k+1}-1}$. We next show that a distinct stable matching in our new stable marriage problem of size 2^{k+1} can be defined from any one of these combinations. This would mean that we have found at least $2^{2^{k+1}-1}$

stable matchings for a stable marriage problem of size 2^{k+1} and the proof would be complete.

From the sequence $M_1, M_2, \ldots, M_n, M$ of stable matchings, we define a marriage as follows:

$$(b_i, m_j) \text{ marries } (b_i\text{'s partner in } M, m_j\text{'s partner in } M_i). \tag{12.14}$$

The proof is finished by showing that these marriages form a matching, each one is different, and that the matching is stable. These three facts are left to the exercises. Q.E.D.

Among other things, Theorem 12.11 shows that the number of stable matchings for stable marriage problems grows exponentially in the size of the problem.[11] Although finding all of the stable matchings for a given stable marriage problem is difficult, both algorithmically and numerically, we can still comment on the structure of the stable matchings. We do this next.

12.8.3 Structure of Stable Matchings

As in Section 4.2.5, we can define a *dominance* relation for stable matchings as follows: A person x prefers one stable matching, M_i, over another, M_j, if x prefers his/her partner in M_i to his/her partner in M_j. A *man-oriented dominance relation* says that M_i dominates M_j if every man either prefers M_i to M_j or is indifferent between them. (Recall that you are indifferent between alternatives A and B if you neither prefer A to B nor B to A.) A *woman-oriented dominance relation* is defined similarly.

As we stated in Section 4.2.5, stable matchings form a partial order (actually, they form a lattice) under either of the man-oriented or woman-oriented dominance relations. (The proof of this fact is part of Exercise 9.) Consider all of the stable matchings associated with the stable marriage problem of Table 12.4. There are 10 of them:

$$\begin{aligned}
M_0 &= \{m_1 - w_4, m_2 - w_3, m_3 - w_2, m_4 - w_1\} \\
M_1 &= \{m_1 - w_3, m_2 - w_4, m_3 - w_2, m_4 - w_1\} \\
M_2 &= \{m_1 - w_4, m_2 - w_3, m_3 - w_1, m_4 - w_2\} \\
M_3 &= \{m_1 - w_3, m_2 - w_4, m_3 - w_1, m_4 - w_2\} \\
M_4 &= \{m_1 - w_3, m_2 - w_1, m_3 - w_4, m_4 - w_2\} \\
M_5 &= \{m_1 - w_2, m_2 - w_4, m_3 - w_1, m_4 - w_3\} \\
M_6 &= \{m_1 - w_2, m_2 - w_1, m_3 - w_4, m_4 - w_3\} \\
M_7 &= \{m_1 - w_1, m_2 - w_2, m_3 - w_4, m_4 - w_3\} \\
M_8 &= \{m_1 - w_2, m_2 - w_1, m_3 - w_3, m_4 - w_4\} \\
M_9 &= \{m_1 - w_1, m_2 - w_2, m_3 - w_3, m_4 - w_4\}.
\end{aligned} \tag{12.15}$$

[11] Irving and Leather [1986] found a recurrence relation that improves greatly on Theorem 12.11.

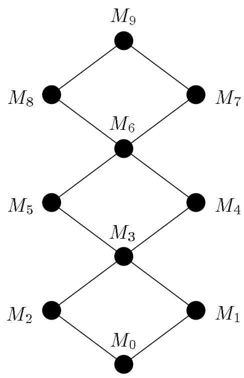


Figure 12.28: A lattice of all stable matchings (12.15) for the preferences in Table 12.4 under the man-oriented dominance relation.

The lattice diagram for this example, under the man-oriented dominance relation, is shown in Figure 12.28. One property of lattices is that they have a maximum and a minimum element. The maximum stable matching and minimum stable matching in the man-oriented lattice are called *man-optimal* and *man-pessimal*, respectively. Because of how man-oriented dominance is defined, of all the women to whom each man is married in any of the stable matchings, he ranks highest the one that he is married to in the maximum element of the man-oriented lattice. Notice, however, that w_1 is married to either m_1, m_2, m_3, or m_4 when considering all of the stable matchings in the lattice. Of the four men, she least prefers m_1 and he is the one she is married to in the maximum element of the man-oriented lattice. This is the case for all of the women; of all the men each woman is married to in any of the stable matchings, she least prefers the one that she is married to in the maximum element of the man-oriented lattice. This is not by coincidence.

Theorem 12.12 (McVitie and Wilson [1971]) In the man-optimal stable matching, each woman has her least preferred partner of all her stable marriage partners.

Proof. The proof will be by contradiction. Let M_m be the man-optimal stable matching. Suppose that woman w prefers m_1 to m_2 but she is married to m_1 in M_m and m_2 in another stable matching M. Since the marriage $w - m_1$ occurs in M_m, it must be the case that m_1 prefers w to any other woman in any of the other stable matchings, including M. A contradiction is reached by noting that m_1 and w must be a blocking pair in M. Q.E.D.

Corollary 12.12.1 The maximum element in the man-oriented lattice is the minimum element in the woman-oriented lattice, i.e., the man-optimal stable matching is the woman-pessimal stable matching.

Similar results hold for the woman-oriented dominance relation and lattice.

Finally, it is interesting to see which element in the man-oriented lattice is the stable matching that results from the GS Algorithm.

Table 12.7: Preference Lists for a 5-Resident, 2-Hospital Stable Matching Problem

Residents' Preferences					Hospitals' Preferences	
r_1	r_2	r_3	r_4	r_5	H_a	H_b
H_a	H_b	H_a	H_a	H_a	r_2	r_1
H_b	H_a	H_b	H_b	H_b	r_3	r_3
					r_4	r_5
					r_5	r_2
					r_1	r_4

Theorem 12.13 In the GS Algorithm's stable matching, each man has the best partner that he can have in any stable matching.

It should not be surprising that the GS Algorithm produces the maximum (man-optimal) element in the man-oriented lattice. For it is the men proposing and doing so from their top choice on down. Theorem 12.13 is proved in Exercise 10.

12.8.4 Stable Marriage Extensions

The stable marriage problem that was introduced in this section is fairly restrictive. Namely, the sets of men and women must be of equal size; each person's preference list contains every member of the other gender; there are no ties; marriages must be monogamous and heterosexual; and the list goes on. We discuss a number of variants of the stable marriage problem in this section as well as in the exercises.

Example 12.20 National Resident Matching Program When students graduate from medical school, they apply to hospitals to do their internships. Each student (resident) ranks the list of hospitals and the hospitals rank the list of residents. The National Resident Matching Program (NRMP) was devised to find a stable matching of the hospitals and residents. (See Roth [1982] for an extensive treatment of the NRMP.)

The NRMP is different from the stable marriage problem in a number of respects. First, the number of residents and hospitals are not necessarily the same. Also, each hospital typically needs to be matched with more than one resident. Note that these two conditions are not unrelated.

The NRMP Algorithm, that is a slight modification of the GS Algorithm, will be used to find a stable matching in this situation. Suppose that hospital H_i has a quota of q_i residents. As opposed to engagements in the GS Algorithm, the NRMP Algorithm uses waiting lists to commit temporarily to a resident. When the algorithm terminates, the waiting lists become internships.

As with the GS Algorithm, each stage of the NRMP Algorithm will be a two-step process where all free (not on waiting list) residents apply to the highest hospital

Table 12.8: The NRMP Algorithm as Applied to the Preference Lists of Table 12.7

Stage 1		Stage 2		Stage 3		Stage 4	
Apps.	Wait Lists: H_a H_b	Apps.	Wait Lists: H_a H_b	Apps.	Wait Lists: H_a H_b	Apps.	Wait Lists: H_a H_b
$r_1 - H_a$	r_3 r_2	$r_1 - H_b$	r_3 r_1	$r_1 -$	r_2 r_1	$r_1 -$	r_2 r_1
$r_2 - H_b$	r_4	$r_2 -$	r_4 r_5	$r_2 - H_a$	r_3 r_5	$r_2 -$	r_3 r_5
$r_3 - H_a$		$r_3 -$		$r_3 -$		$r_3 -$	
$r_4 - H_a$		$r_4 -$		$r_4 -$		$r_4 - H_b$	
$r_5 - H_a$		$r_5 - H_b$		$r_5 -$		$r_5 -$	

on their preference list to which they haven't yet applied. Then each hospital compares its newest applicant(s) to those already on its waiting list and chooses, up to its quota and based on its preference list, a revised waiting list. The algorithm terminates when each applicant is either on a waiting list or has been rejected by every hospital on its preference list. (A proof of stability, similar to the one showing that the GS Algorithm produces a stable matching, is left to the exercises.)

For example, suppose that two hospitals, H_a and H_b, each has a quota of two residents to accept into their internship programs and there are five residents, $r_1, r_2, \ldots, r_5$. The residents and hospitals have the preference lists shown in Table 12.7. (Note that one resident will not be matched, which was not possible in the stable marriage problem with equal-sized lists of men and women.) The stages of the NRMP Algorithm are shown in Table 12.8. Since hospital H_a was matched with its two highest preferred residents, it could not be part of a blocking pair. Therefore, in this case, the results of the NRMP Algorithm produced a stable matching.

Besides the differences mentioned above, the NRMP Algorithm also includes the possibility that not every hospital is on everyone's preference list. With a slight modification, it is also possible to allow each hospital to supply a preference list of only its applicants. ■

Both the stable marriage problem and internship matching have in common the fact that the (stable) matching is taking place between two distinct groups of objects: men and women, residents and hospitals. When colleges, for example, are assigning roommates, they are working with a single group of objects (men or women). The question of finding stable matchings for roommates is due to Gale and Shapley [1962].

Example 12.21 Stable Roommates Consider the case that four people need to share two rooms—two to a room. Each person has a strict preference ranking of the other three people. Room assignments must be made so that there is stability in the assignment. That is, based on their preference lists, two people would not do better by leaving their respective roommates to room together. Suppose that their preference lists are as follows:

p_1	p_2	p_3	p_4
p_4	p_1	p_2	p_2
p_2	p_3	p_4	p_1
p_3	p_4	p_1	p_3

If the pairings were $R = \{p_1 - p_2, p_3 - p_4\}$, then p_1 and p_4 would each be willing to leave their respective roommates to room with each other; p_1 is higher on p_4's list than p_3 and p_4 is higher on p_1's list than p_2. Thus, R is not a stable matching. However, $S = \{p_1 - p_4, p_2 - p_3\}$ is easily seen to be stable. ■

The stable roommates problem was one of 12 open problems stated by Donald Knuth in 1976; see Knuth [1997]. He asked: Each person from a set of $2n$ persons rates the other $2n - 1$ people according to a certain order of preference. Find an efficient algorithm permitting us to obtain, if possible, a stable set of n couples. An answer to Knuth's question was given by Irving [1985], who found a polynomial-time algorithm that produced a stable matching, if one existed. Unlike the stable marriage problem, stability is not always possible in the stable roommates problem. Exercise 11 presents such an example. Other references to the stable roommates problem include Subramanian [1994] and Feder [1992].

Stability has found its way into many diverse areas. Besides internships and roommates, some connections exist between stable matchings and auctions, computer science (e.g., hashing and data structures), mathematics (e.g., coloring and shortest path problems), and coupon collecting, among others. (See Galvin [1995], Knuth [1974], Knuth [1997], Roth and Sotomayor [1990], Spira [1973], and Ullman [1972].)

EXERCISES FOR SECTION 12.8

1. How many different stable marriage problems of size n are possible? Explain.
2. In a stable marriage problem of size n, how many possible matchings exist? Explain.
3. Find all of the stable matchings for the stable marriage problem of size 4 given in Table 12.5.
4. Show that $M = \{m_1 - w_3, m_2 - w_2, m_3 - w_1, m_4 - w_4\}$ is not a stable matching for the stable marriage problem of size 4 given in Table 12.4.
5. (a) Apply the GS Algorithm to the stable marriage problem of Table 12.9.
   (b) Apply a "woman proposing" GS Algorithm to the stable marriage problem of Table 12.9.
   (c) Explain the significance of the results of parts (a) and (b).
6. Produce a stable marriage problem of size 3 that has exactly one stable matching.
7. Write an algorithm for testing stability of a matching for a stable marriage problem of size n.

Table 12.9: A Stable Marriage Problem of Size 4

Men's Preferences				Women's Preferences			
m_1	m_2	m_3	m_4	w_1	w_2	w_3	w_4
w_1	w_2	w_3	w_4	m_4	m_3	m_2	m_1
w_2	w_1	w_4	w_3	m_3	m_4	m_1	m_2
w_3	w_4	w_1	w_2	m_2	m_1	m_4	m_3
w_4	w_3	w_2	w_1	m_1	m_2	m_3	m_4

Table 12.10: Roommate Preference Lists.

p_1	p_2	p_3	p_4
p_2	p_3	p_1	p_1
p_3	p_1	p_2	p_2
p_4	p_4	p_4	p_3

8. This exercise finishes the proof of Theorem 12.11.

   (a) Show that the marriages defined in (12.14) form a matching.

   (b) Show that the matchings are all different.

   (c) Show that the matching resulting from the marriages defined in (12.14) is stable.

9. For any given stable marriage problem, prove that the stable matchings form a lattice under either the man-oriented or woman-oriented dominance relation.

10. This exercise produces a proof of Theorem 12.13 by contradiction. Let M_{GS} and M be two stable matchings in the man-oriented lattice of a stable marriage problem of size n. Suppose that man m is married to w_i and w_j in M_{GS} and M, respectively, and that m prefers w_j to w_i.

    (a) Explain why w_j must have broken her engagement to m, in favor of, say, m', in the GS Algorithm.

    (b) Suppose, without loss of generality, that w_j's choice of m' was the first such broken engagement to a stable partner in some stable matching during the GS Algorithm. Of all of m''s partners in all the stable matchings, show that m' prefers w_j.

    (c) Explain why m' and w_j are a blocking pair in the stable matching M. (This would then be the contradiction.)

11. Consider the roommates problem of Table 12.10.

    (a) Show that there does not exist a stable matching.

    (b) If person p_4 changed his or her preference list, could a stable matching be found?

REFERENCES FOR CHAPTER 12

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B., *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.

Balinski, M., and Ratier, G., "Of Stable Marriages and Graphs, and Strategy and Polytopes," *SIAM Rev., 39* (1997), 575–604.

Berge, C., "Two Theorems in Graph Theory," *Proc. Natl. Acad. Sci. USA, 43* (1957), 842–844.

Berge, C., *Graphs and Hypergraphs*, American Elsevier, New York, 1973.

Brogan, W. L., "Algorithm for Ranked Assignments with Application to Multiobject Tracking," *J. Guidance,* (1989), 357–364.

Christofides, N., *Graph Theory: An Algorithmic Approach*, Academic Press, New York, 1975.

Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., and Schrijver, A., *Combinatorial Optimization*, Wiley, New York, 1998.

Deo, N., *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall, Englewood Cliffs, NJ, 1974.

Derman, C., and Klein, M., "A Note on the Optimal Depletion of Inventory," *Management Sci., 5* (1959), 210–214.

Devine, M. V., "A Model for Minimizing the Cost of Drilling Dual Completion Oil Wells," *Management Sci., 20* (1973), 532–535.

Edmonds, J., "Paths, Trees and Flowers," *Canad. J. Math., 17* (1965), 449–467. (a)

Edmonds, J., "The Chinese Postman Problem," *Oper. Res., 13, S1* (1965), 373. (b)

Edmonds, J., and Johnson, E. L., "Matching, Euler Tours and the Chinese Postman," *Math. Program., 5* (1973), 88–124.

Egerváry, E., "Matrixok Kombinatóricus Tuladjdonságairól," *Mat. Fiz. Lapok, 38* (1931), 16–28. ("On Combinatorial Properties of Matrices," translated by H. W. Kuhn, Office of Naval Research Logistics Project Report, Department of Mathematics, Princeton University, Princeton, NJ, 1953.)

Feder, T., "A New Fixed Point Approach for Stable Networks and Stable Marriages," *J. Comput. System Sci., 45* (1992), 233–284.

Fiala, J., Kratochvíl, J., and Proskurowski, A., "Distance Constrained Labeling of Precolored Trees," in A. Restivo, S. Ronchi Della Rocca, and L. Roversi (eds.), *Theoretical Computer Science: Proceedings of the 7th Italian Conference, ICTCS 2001, Torino, Italy, October 2001*, Lecture Notes in Computer Science, 2202, Springer-Verlag, Berlin, 2001, 285–292.

Ford, L. R., and Fulkerson, D. R., *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

Gale, D., and Shapley, L. S., "College Admissions and the Stability of Marriage," *Amer. Math. Monthly, 69* (1962), 9–15.

Gallai, T., "Über Extreme Punkt- und Kantenmengen," *Ann. Univ. Sci. Budap., Eötvös Sect. Math., 2* (1959), 133–138.

Galvin, F., "The List Chromatic Index of a Bipartite Multigraph," *J. Combin. Theory, Ser. B, 63* (1995), 153–158.

Gondran, M., and Minoux, M., *Graphs and Algorithms*, Wiley, New York, 1984.

Grötschel, M., Lovász, L., and Schrijver, A., *Geometric Algorithms and Combinatorial Optimization*, 2nd ed., Springer-Verlag, Berlin, 1993.

Gusfield, D., and Irving, R. W., *The Stable Marriage Problem: Structure and Algorithms*, Foundations of Computing Series, MIT Press, Cambridge, MA, 1989.

HALL, P., "On Representatives of Subsets," *J. Lond. Math. Soc., 10* (1935), 26–30.

HOPCROFT, J. E., and KARP, R. M., "A $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs," *SIAM J. Comput., 2* (1973), 225–231.

IRVING, R. W., "An Efficient Algorithm for the Stable Roommates Problem," *J. Alg., 6* (1985), 577–595.

IRVING, R. W., and LEATHER, P., "The Complexity of Counting Stable Marriages," *SIAM J. Comput., 15* (1986), 655–667.

KNUTH, D. E., "Structured Programming with **go to** Statements," *Comput. Surveys, 6* (1974), 261–301.

KNUTH, D. E., *Stable Marriage and Its Relation to Other Combinatorial Problems*, CRM Proceedings & Lecture Notes, 10, American Mathematical Society, Providence, RI, 1997. (Translated by M. Goldstein with revisions by D. Knuth from *Mariages Stables et Leurs Relations Avec d'Autres Problèmes Combinatoires*, Collection de la Chaire Aisenstadt, Les Presses de l'Université de Montréal, Montreal, Canada, 1976.)

KOLITZ, S., personal communication to R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, 1991.

KÖNIG, D., "Graphen und Matrizen," *Mat. Fiz. Lapok, 38* (1931), 116–119.

KUHN, H. W., "The Hungarian Method for the Assignment Problem," *Naval Res. Logist. Quart., 2* (1955), 83–97.

LAWLER, E. L., *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

LOVÁSZ, L., and PLUMMER, M. D., *Matching Theory*, North-Holland, Amsterdam, 1986.

MASON, A. J., and PHILPOTT, A. B., "Pairing Stereo Speakers Using Matching Algorithms," *Asia-Pacific J. Oper. Res., 5* (1988), 101–116.

MCVITIE, D. G., and WILSON, L. B., "The Stable Marriage Problem," *Comm. ACM, 14* (1971), 486–490.

MICALI, S., and VAZIRANI, V. V., "An $O(\sqrt{|V|} \cdot |E|)$ Algorithm for Finding Maximum Matching in General Graphs," *Proceedings of the Twenty-First Annual Symposium on the Foundations of Computer Science*, IEEE, Long Beach, CA, 1980, 17–27.

MINIEKA, E., *Optimization Algorithms for Networks and Graphs*, Dekker, New York, 1978.

NORMAN, R. Z., and RABIN, M. O., "An Algorithm for a Minimum Cover of a Graph," *Proc. Amer. Math. Soc., 10* (1959), 315–319.

PAPADIMITRIOU, C. H., and STEIGLITZ, K., *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, NJ, 1982.

PETERSON, J., "Die Theorie der Regulären Graphen," *Acta Math., 15* (1891), 193–220.

ROTH, A. E., "The Economics of Matching: Stability and Incentives," *Math. Oper. Res., 7* (1982), 617–628.

ROTH, A. E., and SOTOMAYOR, M. A. O., *Two-Sided Matching*, Cambridge University Press, Cambridge, MA, 1990.

SPIRA, P. M., "A New Algorithm for Finding All Shortest Paths in a Graph of Positive Arcs in Average Time $O(n^2 \log^2 n)$," *SIAM J. Comput., 2* (1973), 28–32.

SUBRAMANIAN, A., "A New Approach to Stable Matching Problems," *SIAM J. Comput., 23* (1994), 671–700.

TUCKER, A. C., *Applied Combinatorics*, Wiley, New York, 1980.

TUTTE, W. T., "The Factorization of Linear Graphs," *J. Lond. Math. Soc., 22* (1947), 107–111.

ULLMAN, J. D., "A Note on the Efficiency of Hashing Functions," *J. Assoc. Comput. Mach., 19* (1972), 569–575.

Chapter 13

Optimization Problems for Graphs and Networks

In this chapter we present procedures for solving a number of combinatorial optimization problems. We have encountered such problems throughout the book and discussed some in detail in Sections 12.4–12.6. The problems discussed will all be translated into problems involving graphs or digraphs. More generally, they will be translated into problems involving graphs or digraphs in which each edge or arc has one or more *nonnegative* real numbers assigned to it. Such a graph or digraph is called a *network* or a *directed network*, respectively.

It should be mentioned that we have chosen problems to discuss for which good solutions (good algorithms) are known. Not all combinatorial optimization problems have good solutions. The traveling salesman problem, discussed in Sections 2.4 and 11.5, is a case in point. In such a case, as we pointed out in Section 2.18, we seek good algorithms that solve the problem in special cases or that give approximate solutions. Discussion of these approaches is beyond the scope of this book.

13.1 MINIMUM SPANNING TREES

13.1.1 Kruskal's Algorithm

In Examples 3.28–3.32 we described the problem of finding a minimum spanning tree of a graph G with weights, that is, a network. This is a spanning subgraph of G that forms a tree and that has the property that no other spanning tree has a smaller sum of weights on its edges. Here we discuss a simple algorithm for finding a minimum spanning tree of a network G. The algorithm will either find a minimum spanning tree or conclude that the network does not have a spanning tree. By Theorem 3.19, the latter implies that the network is disconnected. The problem of finding a minimum spanning tree has a wide variety of applications: for example, in the planning of large-scale transportation, communication, or distribution networks;

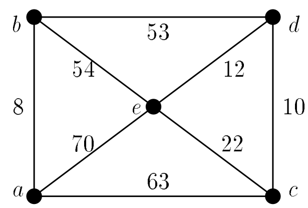


Figure 13.1: A network of road distances between villages.

in reducing data storage; and in data mining. We mentioned some of these in Section 3.5.4. See Ahuja, Magnanti, and Orlin [1993] or Graham and Hell [1985] for a survey and references on many applications.

According to Graham and Hell [1985], the earliest algorithms for finding a minimum spanning tree apparently go back to Borůvka [1926a,b][1], who was interested in the most economical layout for a power-line network. Earlier work on this problem is due to the anthropologist Czekanowski [1909, 1911, 1928] in his work on classification schemes.

Let us recall the problem by continuing with Example 3.29. Suppose that there are five villages in a remote region and that the road distances between them are as shown in the network of Figure 13.1. We wish to put in telephone lines along the existing roads in such a way that every pair of villages is linked by telephone service and we minimize the total number of miles of telephone line. The problem amounts to finding a minimum spanning tree of the network of Figure 13.1.

In finding a spanning tree of such a network that minimizes the sum of the weights of its edges, we will build up the tree one edge at a time, and we will be *greedy*; that is, we will add edges of smallest weight first. It turns out that being greedy works. Specifically, let us order the edges of the network G in order of increasing weight. In our example, we order them as follows: $\{a,b\}$, $\{c,d\}$, $\{d,e\}$, $\{c,e\}$, $\{b,d\}$, $\{b,e\}$, $\{a,c\}$, $\{a,e\}$. In case of ties, the ordering is arbitrary. We examine the edges in this order, deciding for each edge whether or not to include it in the spanning tree T. It is included as long as it does not form a circuit with some already included edges. This algorithm is called Kruskal's Algorithm (it is due to Kruskal [1956]) and can be summarized as follows.

Algorithm 13.1: Kruskal's Minimum Spanning Tree Algorithm

Input: A network G on $n > 1$ vertices.
Output: A set of edges defining a minimum spanning tree of G or the message that G is disconnected.

Step 1. Arrange the edges of G by increasing order of weight. (Order arbitrarily in case of ties.) Set $T = \emptyset$. (T is the set of edges of the minimum spanning tree.)

Step 2. If every edge of G has been examined, stop and output the message that G is disconnected. Otherwise, examine the first unexamined edge in the ordered

[1] Nešetřil, Milková, and Nešetřilová [2001] is an English translation of Borůvka's works in Czech on minimum spanning trees, with commentaries and historical development.

list and include it in the set T if and only if it does not form a circuit with some edges already in T. If the edge is added to T, go to Step 3. Otherwise, repeat Step 2.

Step 3. If T has $n-1$ edges, stop and output T. Otherwise, go to Step 2.

Theorem 13.1 If G is a connected network on n vertices, Kruskal's Algorithm will terminate with a minimum spanning tree T of $n-1$ edges. If G is a disconnected network, the algorithm will terminate with the message that G is disconnected after having examined all the edges and not yet having put $n-1$ edges into T.

Before presenting a proof of Theorem 13.1, we illustrate the algorithm on our example of Figure 13.1. We start by including edge $\{a, b\}$ in T. Now the number of vertices of G is $n = 5$, and T does not yet have $n - 1 = 4$ edges, so we return to Step 2. In Step 2 we note that not every edge of G has been examined. We examine edge $\{c, d\}$, adding it to T because it does not form a circuit with edges in T. Now T has two edges, so we return to Step 2. Since the first unexamined edge $\{d, e\}$ does not form a circuit with existing edges in T, we add this edge to T. Now T has three edges, so we return to Step 2. The next edge examined, $\{c, e\}$, does form a circuit with edges $\{c, d\}$ and $\{d, e\}$ of T, so we do not add it to T. We repeat Step 2 for the next edge, $\{b, d\}$. This is added to T. Since T now has four edges in it, we stop and conclude that these edges define a minimum spanning tree of G.

Let us note that Kruskal's Algorithm can be applied to a graph G without weights if we simply list the edges in arbitrary order. Then the algorithm will produce a spanning tree of G or it will conclude that G is disconnected.

Let us consider the computational complexity of Kruskal's Algorithm. Disregarding the step of putting the edges in order, the algorithm takes at most e steps, one for each edge. The reader should not confuse "step" here with "step" in Algorithm 13.1. Some edges require both Steps 2 and 3 of the algorithm. So, in some sense, up to $2e$ steps are required. However, $O(e) = O(2e)$ (see Section 2.18.1), so e steps and $2e$ steps are considered "the same."

How many steps does it take to order the edges in terms of increasing weight? This is simply a sorting problem, of the kind discussed in Section 3.6.4. In that section we observed that sorting a set of e items can be performed in $ke \log_2 e$ steps, where k is a constant. Thus, the entire Kruskal's Algorithm can be performed in at most $e + ke \log_2 e \leq e + ke^2$ steps. Note that $e + ke^2 \leq n^2 + kn^4$, since $e \leq \binom{n}{2} \leq n^2$. Thus, the entire Kruskal's Algorithm can be performed in a number of steps that is a polynomial in either e or n. (In the terminology of Section 2.18, our crude discussion shows that we have an $O(e \log_2 e)$ algorithm. The best implementation of Kruskal's Algorithm, by Tarjan [1984a], finds a minimum spanning tree in $O(e\alpha(n, e))$ steps for a function $\alpha(n, e)$ that grows so slowly that for all practical purposes it can be considered a constant less than 6. See Ahuja, Magnanti, and Orlin [1993], Cheriton and Tarjan [1976], Graham and Hell [1985], or Tarjan [1983, 1984b].)

Example 13.1 Clustering and Data Mining (Ahuja, Magnanti, and Orlin [1993], Gower and Ross [1969], Zahn [1971]) In many practical problems of detection, decisionmaking, or pattern recognition, especially in "data mining" of today's massive datasets in a wide variety of fields, we often seek to partition the data into natural groups or *clusters*. Clustering problems arise in applications involving molecular databases, astrophysical studies, geographic information systems, software measurement, worldwide ecological monitoring, medicine, and analysis of telecommunications and credit card data for fraud. (See Arratia and Lander [1990], Baritchi, Cook, and Holder [2000], Church [1997], Godehart [1990], Karger [1997], or Neil [1992].) The idea is that elements in the same cluster should be closely related, whereas elements in different clusters should not be as closely related, although how to define this precisely is subject to a variety of attempts to make it precise. (See, for example, Guénoche, Hansen, and Jaumard [1991], Hansen, Frank, and Jaumard [1989], Hansen and Jaumard [1987], Hansen, Jaumard, and Mladenovic [1998], or Mirkin [1996].) In many applications, the objects to be clustered are represented as points in two-dimensional Euclidean space (more generally, k-dimensional space). Here, Kruskal's Algorithm is widely used for clustering. At each iteration, the algorithm produces a set of edges that partition the graph into a set of trees. Each tree can be considered a cluster. The algorithm produces n partitions if there are n vertices, starting with the partition where each cluster has one vertex and ending with the partition where there is only one cluster consisting of all the points. Which of these partitions is "best" or "most useful" depends on defining an "objective function" precisely. ■

13.1.2 Proof of Theorem 13.1[2]

To prove Theorem 13.1, note that at each stage of the algorithm the edges in T define a graph that has no circuits. Suppose that the algorithm terminates with T having $n-1$ edges. Then, by Theorem 3.20, T is a tree, and hence a spanning tree of G. Thus, G is connected. Hence, if G is not connected, the algorithm will terminate without finding a set T with $n-1$ edges.

If G is connected, we shall show that the algorithm gives rise to a minimum spanning tree. First, we show that the algorithm does give rise to a spanning tree. Then we show that this spanning tree is minimum. The first observation follows trivially from Theorem 3.16 if the algorithm gives us a T of $n-1$ edges. Suppose that the algorithm terminates without giving us a T of $n-1$ edges. At the time of termination, let us consider the edges in T. These edges define a spanning subgraph of G, that we also refer to as T. This subgraph is not connected, for any connected spanning subgraph of G, having a spanning tree in it, must have at least $n-1$ edges. Hence, the subgraph T has at least two components. But since G is connected, there is in G an edge $\{x, y\}$ joining vertices in different components of T. Now $\{x, y\}$ cannot form a circuit with edges of T. Hence, when the algorithm came to examine this edge, it should have included it in T. Thus, this situation arises only if the algorithm was applied incorrectly.

[2] This subsection may be omitted.

We now know that the algorithm gives us a spanning tree T. Let S be a minimum spanning tree of G. If $S = T$, we are done. Suppose that $S \neq T$. Since $S \neq T$ and since both have the same number of edges $(n-1)$, there must be an edge in T that is not in S. In the order of edges by increasing weight, find the first edge $e_1 = \{x, y\}$ that is in T but not in S.

Since S is a spanning tree of G, there is a simple chain $C(x, y)$ from x to y in S. Now adding edge e_1 to S gives us a circuit. Thus, since T has no circuits, there is an edge e_2 on this circuit and hence on the chain $C(x, y)$ that is not in T. Let S' be the set of edges obtained from S by removing e_2 and adding e_1. Then S' defines a graph that is connected (why?) and has n vertices and $n-1$ edges, so by Theorem 3.16, S' is a spanning tree. We consider two cases.

Case 1: Edge e_2 precedes edge e_1 in the order of edges by increasing weights. In this case, e_2 was not put in T, so it must form a circuit D with edges of T that were examined before e_2. But e_1 is the first edge of T not in S, so all edges in D must be in S since they are in T. Thus, e_2 could not have been put in S, which is a contradiction. Case 1 is impossible.

Case 2: Edge e_1 precedes edge e_2. In this case, e_2 has weight at least as high as the weight of e_1. Thus, S' has its sum of weights at most the sum of weights of S, so S' is a minimum spanning tree since S is. Moreover, S' has one more edge in common with T than does S.

If $T \neq S'$, we repeat the argument for T and S', obtaining a minimum spanning tree S'' with one more edge in common with T than S'. Eventually, we find a minimum spanning tree that is the same as T. This completes the proof.

13.1.3 Prim's Algorithm

There is an alternative algorithm for finding a minimum spanning tree that is also a greedy algorithm. We present this algorithm here.

Algorithm 13.2: Prim's Minimum Spanning Tree Algorithm[3]

Input: A network G on $n > 1$ vertices.
Output: A minimum spanning tree of G or the message that G is disconnected.

Step 1. Set $T = \emptyset$. Pick an arbitrary vertex of G and put it in the tree T.

Step 2. Add to T that edge joining a vertex of T to a vertex not in T that has the smallest weight among all such edges. Pick arbitrarily in case of ties. If it is impossible to add any edge to T, stop and output the message that G is disconnected.

Step 3. If T has $n-1$ edges, stop and output T. Otherwise, repeat Step 2.

To illustrate this algorithm, let us again consider the graph of Figure 13.1 and start with vertex a. Then we add edge $\{a, b\}$ to T because it has the smallest weight

[3]This algorithm was discovered by Prim [1957] and is usually attributed to him. In fact, it seems to have been discovered previously by Jarník [1930] (Graham and Hell [1985]).

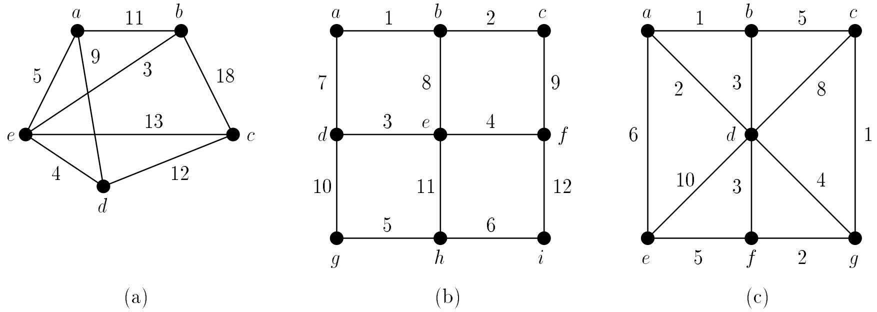


Figure 13.2: Networks for exercises of Section 13.1.

of the edges joining a. Next, we examine edges joining vertices a or b to vertices not in T: namely, c, d, or e. Edge $\{b, d\}$ has the smallest weight of all these edges; we add it to T. Next, we examine edges joining vertices a, b, or d to vertices not in T, namely c or e. Edge $\{d, c\}$ has the smallest weight of all of these edges. We add this edge to T. Finally, we examine edges joining a, b, c, or d to the remaining vertex not in T, namely e. Edge $\{d, e\}$ has the smallest weight of all these edges, so we add it to T. Now T has four edges, and we terminate. Note that we found the same T by using Kruskal's Algorithm.

Theorem 13.2 If G is a connected network on n vertices, Prim's Algorithm will terminate with a minimum spanning tree T of $n - 1$ edges. If G is a disconnected network, the algorithm will terminate with the message that G is disconnected because it is impossible to add another edge to T.

The proof is left as an exercise (Exercise 14).

As of this time, the best implementations of Prim's Algorithm run in $O(e + n \log_2 n)$ steps. (See Ahuja, Magnanti, and Orlin [1993], Graham and Hell [1985], Johnson [1975], or Kershenbaum and Van Slyke [1972]). See Gabow, *et al.* [1986] for a variant of Prim, that is currently the fastest algorithm for finding a minimum spanning tree. Another, early, unpublished minimum spanning tree algorithm is due to Sollin (see Ahuja, Magnanti, and Orlin [1993]). Yao [1975] improved on Sollin's Algorithm to develop an $O(e \log\log n)$ minimum spanning tree algorithm. See Exercise 16 for another minimum spanning tree algorithm that originates from work of Borůvka [1926a,b] and that applies if all weights are distinct.

EXERCISES FOR SECTION 13.1

1. For each network of Figure 13.2, find a minimum spanning tree using Kruskal's Algorithm.
2. Repeat Exercise 1 with Prim's Algorithm. Start with vertex a.
3. Apply Kruskal's Algorithm to each network of Figure 13.3.
4. Apply Prim's Algorithm to each network of Figure 13.3. Start with vertex a.

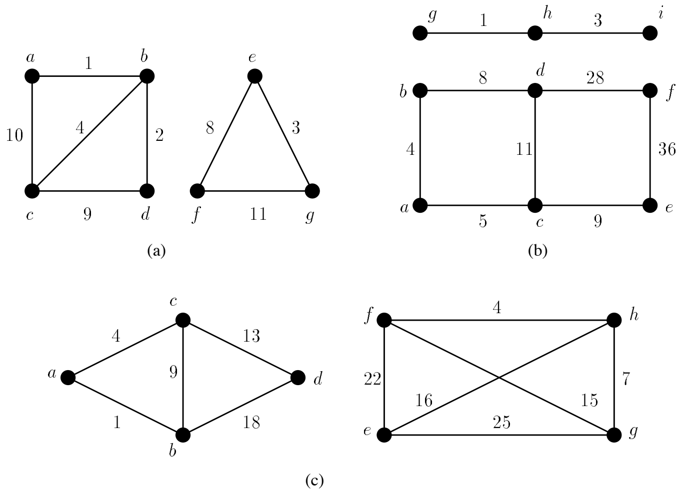


Figure 13.3: Networks for exercises of Section 13.1.

Table 13.1: Distance Between Vats

Vat	1	2	3	4	5	6	7	8
1	0	3.9	6.3	2.7	2.1	5.4	6.0	4.5
2	3.9	0	2.7	5.4	3.6	7.8	6.9	3.3
3	6.3	2.7	0	7.8	5.1	7.5	5.7	3.0
4	2.7	5.4	7.8	0	2.1	4.8	4.5	2.7
5	2.1	3.6	5.1	2.1	0	2.7	3.3	2.4
6	5.4	7.8	7.5	4.8	2.7	0	1.8	3.0
7	6.0	6.9	5.7	4.5	3.3	1.8	0	1.5
8	4.5	3.3	3.0	2.7	2.4	3.0	1.5	0

5. A chemical company has eight storage vats and wants to develop a system of pipelines that makes it possible to move chemicals from any vat to any other vat. The distance in feet between each pair of vats is given in Table 13.1. Determine between which pairs of vats to build pipelines so that chemicals can be moved from any vat to any other vat and so that the total length of pipe used is minimized.

6. A digital computer has a variety of components to be connected by high-frequency circuitry (wires) (see Example 3.30). The distance in millimeters between each pair of components is given in Table 13.2. Determine which pairs of components to connect so that the collection of components is connected and the total length of wire between components is minimized (to reduce capacitance and delay line effects).

7. For the set of points in Figure 13.4, use the method of Example 13.1 to find all partitions into 8 clusters, into 7 clusters, ..., into 1 cluster based on actual distance between the points.

Table 13.2: Distance Between Components

	1	2	3	4	5	6
1	0	6.7	5.2	2.8	5.6	3.6
2	6.7	0	5.7	7.3	5.1	3.2
3	5.2	5.7	0	3.4	8.5	4.0
4	2.8	7.3	3.4	0	8.0	4.4
5	5.6	5.1	8.5	8.0	0	4.6
6	3.6	3.2	4.0	4.4	4.6	0

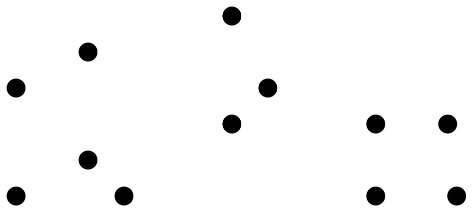


Figure 13.4: A set of points.

8. Using straight-line distances between cities, find the minimum spanning tree that connects Washington, DC and each (continental) U.S. state capital. (These distances can be found easily on the Internet.)
9. In each network of Figure 13.2, find a *maximum spanning tree*, a spanning tree so that no other spanning tree has a larger sum of weights.
10. Modify Kruskal's Algorithm so that it finds a maximum spanning tree.
11. If each edge of a network has a different weight, can there be more than one minimum spanning tree? Why?
12. (a) In the network of Figure 13.1, find a minimum spanning tree if it is required that the tree include edge $\{a, e\}$. (Such a problem might arise if we are required to include a particular telephone line or if one already exists.)

    (b) Repeat part (a) for edge $\{c, e\}$.
13. How would you modify Kruskal's Algorithm to deal with a situation such as that in Exercise 12, where one or more edges are specified as having to belong to the spanning tree?
14. Prove that Prim's Algorithm works.
15. Show by a crude argument that the computational complexity of Prim's Algorithm is bounded by a polynomial in e or in n.
16. Another algorithm for finding a minimum spanning tree T in an n vertex network G, provided that all weights are distinct, works as follows.

    Step 1. Set $T = \emptyset$.

    Step 2. Let G' be the spanning subgraph of G consisting of edges in T.

    Step 3. For each connected component K of G', find the minimum-weight edge of G joining a vertex of K to a vertex of some other component of G'. If there is no such edge, stop and output the message that G is disconnected. Otherwise, add all the new edges to T.

Step 4. If T has $n-1$ edges, stop and output T. Otherwise, repeat Step 2.

This algorithm has its origin in the work of Borůvka [1926a,b].

(a) Apply this algorithm to each network of Figure 13.2.

(b) Prove that the algorithm works provided that all weights are distinct.

17. Show that a minimum spanning tree T is unique if and only if any edge $\{x, y\}$ not in T has larger weight than any edge on the circuit created by adding edge $\{x, y\}$ to T.

18. Let G be a connected graph. A *simple cut set* in G is a set B of edges whose removal disconnects G but such that no proper subset of B has this property. Let F be a set of edges of G. Show the following.

    (a) If F is a simple cut set, F satisfies the following property C: For every spanning tree H of G, some edge of F is in H.

    (b) If F satisfies property C but no proper subset of F does, then F is a simple cut set.

19. (Ahuja, Magnanti, and Orlin [1993]) How would you find a spanning tree T that minimizes

$$\left[\sum_{\{i,j\}\in T} (w_{ij})^2\right]^{1/2},$$

where w_{ij} is the weight on edge $\{i, j\}$?

20. (Ahuja, Magnanti, and Orlin [1993]) Two spanning trees T and T' are *adjacent* if they have all but one edge in common. Show that for any two spanning trees T' and T'', we can find a sequence of spanning trees

$$T' = T_1,\ T_2,\ T_3,\ \ldots,\ T_k = T'',$$

with T_i adjacent to T_{i+1} for $i = 1, 2, \ldots, k-1$.

21. (Ahuja, Magnanti, and Orlin [1993]) Suppose in network G that every edge is colored either red or blue.

    (a) Show how to find a spanning tree with the maximum number of red edges.

    (b) Suppose that some spanning tree has k' red edges and another has $k'' > k'$ red edges. Show that for k with $k'' > k > k'$, there is a spanning tree with k red edges.

13.2 THE SHORTEST ROUTE PROBLEM

13.2.1 The Problem

In this section we consider the problem of finding the shortest route between two vertices in a (directed) network. We begin with some examples that illustrate the problem.

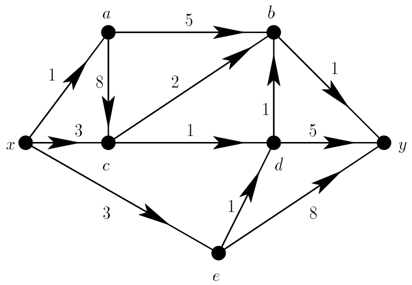


Figure 13.5: A shortest path from x to y is x, c, d, b, y.

Example 13.2 Interstate Highways Suppose that you wish to drive from New York to Los Angeles using only interstate highways. What is the shortest route to take? This problem can be translated into a network problem as follows. Let the vertices of a graph be junctions of interstate highways and join two such vertices by an edge if there is a single interstate highway joining them and uninterrupted by junctions. Put a real number on the edge $\{x, y\}$ representing the number of miles by interstate highway between vertices x and y. In the resulting network, let the *length* of a chain be defined to be the sum of the numbers (weights) on its edges, and take the *distance* $d(x, y)$ between vertices x and y to be the length of the shortest chain between x and y. (If there is no chain between x and y, distance is undefined.) We seek that chain between New York and Los Angeles that is of minimum length, that is, of length equal to d(New York, Los Angeles). ■

Example 13.3 Planning an Airplane Trip Suppose that you wish to fly from New York to Bangkok. What is the route that will have you spending as little time in the air as possible? To answer this question, let the vertices of a digraph be cities in the world air transportation network, and draw an arc from x to y if there is a direct flight from city x to city y. Put a real number on arc (x, y) representing the flying time. Then define the *length* of a path from x to y in this directed network as the sum of the numbers (weights) on the arcs, and define the *distance* $d(x, y)$ from x to y as the length of the shortest path from x to y. (Distance is again undefined if there is no path from x to y.) We seek the path of shortest length from New York to Bangkok, that is, the path with length equal to d(New York, Bangkok). ■

In general, we deal with a network or a directed network, and we seek the shortest chain or path from vertex x to vertex y. We concentrate on directed networks. In Section 13.2.2 we present an algorithm for finding the shortest path from x to y in a directed network. First, let us illustrate the basic ideas. In the directed network of Figure 13.5, the path x, a, b, y has length $1 + 5 + 1 = 7$. The path x, c, d, b, y is shorter; its length is $3 + 1 + 1 + 1 = 6$. Indeed, this is a shortest path from x to y. Hence, $d(x, y) = 6$. Note that we say that x, c, d, b, y is *a* shortest path from x to y. There can be more than one path from x to y of length equal to $d(x, y)$. Here, x, e, d, b, y is another such path. Note also that $d(y, x)$ is undefined; there is no path from y to x.

The problem of finding a shortest path from vertex x to vertex y in a (directed) network is a very common combinatorial problem. According to Goldman [1981], it is perhaps the most widely encountered combinatorial problem in government. Goldman estimated that the shortest path algorithm developed by just one government agency, the Urban Mass Transit Agency in the U.S. Department of Transportation, was regularly applied *billions* of times a year.

Surprisingly, a wide variety of problems can be restated as shortest route problems. We give a few examples.

Example 13.4 The TEX Document Processing System (Ahuja, Magnanti, and Orlin [1993]) A widely used document processing system in the mathematical sciences, TEX, decomposes paragraphs into lines that are both left- and right-justified and so that the appearance is "optimized." (Note that the last line of a paragraph need only be left-justified.) To see how this is done, let the paragraph have words 1, 2, ..., n, in that order. Let $c_{i,j}$ denote the "unattractiveness" of a line that begins with word i and ends with word $j-1$ ($c_{i,j} \geq 0$). For example, $c_{i,j}$ could measure the absolute difference between the paragraph's formatted width and the total length of the words $i, i+1, \ldots, j-1$. The problem is to decompose a paragraph into lines so that the total "cost" of the paragraph is minimized. To see how this is a shortest route problem, let the vertices of a directed network be the words plus a "dummy" vertex (word) denoted $n+1$. Include arcs from i to all $j > i$. The cost on arc (i, j) is $c_{i,j}$. If $1 = i_1, i_2, \ldots, i_k = n+1$ is a shortest path from word 1 to word $n+1$, this means that the most attractive appearance will be obtained by using lines starting at words i_p, $p = 1, 2, \ldots, k-1$, and ending at words $i_{p+1} - 1$, for $p = 1, 2, \ldots, k-1$. (Note that vertex $n+1$ must be included so that the definition of how each line ends can include word n.)[4] ■

Example 13.5 An Inventory of Structural Steel Beams (Ahuja, Magnanti, and Orlin [1993], Frank [1965]) A construction company needs structural steel beams of various lengths. To save storage space and the cost of maintaining a large inventory, the company will cut longer beams into shorter lengths rather than keep beams of all desired lengths. What lengths of beams to keep in inventory will depend on the demand for beams of different lengths. Moreover, the cutting operation will waste some steel. How can the company determine what length beams to keep in inventory so as to minimize the total cost of setting up the inventory and of discarding usable steel lost in cutting? We can formulate this as a shortest route problem by defining a directed network with vertices $0, 1, \ldots, n$ corresponding to the different beam lengths that might be needed. We assume that a beam of length L_i is shorter than a beam of length L_{i+1} and a beam of length L_0 has length 0. We include an arc from vertex i to each vertex $j > i$. We will interpret the arc (i, j) as corresponding to the strategy of keeping beams of lengths L_j in inventory and using them to meet the demand for beams of length L_{i+1}, L_{i+2}, ..., L_j. A path from vertex 0 to vertex n then corresponds to a set of beam lengths

[4] Allowing hyphenated words to come at the end of a line is a complication that we have disregarded. In Exercise 27 we ask the reader to address this complication.

to keep in inventory. Suppose that D_i represents the demand for steel beams of length L_i. The cost $C_{i,j}$ on the arc (i, j) is given by

$$C_{i,j} = K_j + C_j \sum_{k=i+1}^{j} D_k, \tag{13.1}$$

where K_j is the cost of setting up the inventory facility to handle beams of length L_j, C_j is the cost of a beam of length L_j, and the second term corresponds to the cost of using beams of length L_j to meet the demand for all beams of lengths L_{i+1} to L_j. A shortest path from vertex 0 to vertex n gives us the assortment of beams to keep in inventory if we want to minimize total cost. Note that it takes into account the (presumed) higher cost of longer beams and thus the waste from cutting them. What oversimplifications does this analysis make? [See Exercise 20.] ■

There are numerous other examples of shortest route problems. We give several others in Section 13.2.3 and many more are discussed or cited in Ahuja, Magnanti, and Orlin [1993].

13.2.2 Dijkstra's Algorithm

The shortest path algorithm we present is due to Dijkstra [1959]. We present it for a directed network D. Let $w(u, v)$ be the weight on arc (u, v). We recall our standing assumption that weights on arcs in networks are nonnegative. We will need this assumption. It will also be convenient to let $w(u, v)$ be ∞ if there is no arc from u to v. The basic idea is that we find at the kth iteration the k vertices $u_1, u_2, \ldots, u_k$ that are the k closest vertices to x in the network, that is, the ones that have (up to ties) the k smallest distances $d(x, u_1), d(x, u_2), \ldots, d(x, u_k)$. We also find for each u_j a shortest path from x to u_j. Having solved this problem for k, we solve it for $k + 1$.

The general idea in going from the kth iteration to the $(k+1)$st iteration is that for each vertex v not among the k closest vertices, we calculate $d(x, u_j) + w(u_j, v)$ for all j, and find the $(k+1)$st closest vertex as a v for which this sum is minimized. The rationale for this is that if $x, a_1, a_2, \ldots, a_p, v$ is a shortest path from x to v, then $x, a_1, a_2, \ldots, a_p$ must be a shortest path from x to a_p. If not, we could find a shorter path from x to v by using such a shorter path from x to a_p. Moreover, if weights are all positive, if v is the $(k + 1)$st closest vertex to x, and $x, a_1, a_2, \ldots, a_p, v$ is a shortest path from x to v, then a_p must be among the k closest vertices. Even if weights can be zero, the $(k + 1)$st vertex v can be chosen so that there is a shortest path $x, a_1, a_2, \ldots, a_p, v$ from x to v with a_p among the k closest vertices.

At each stage of Dijkstra's Algorithm, we keep a list of vertices included in the first k iterations—this defines a class W—and we keep a list of arcs used in the shortest paths from x to u_j—this defines a class B—and we keep a record of $d(x, u_j)$ for all j. We stop once y is added to W. We then use B to construct the path. We are now ready to present the algorithm more formally.

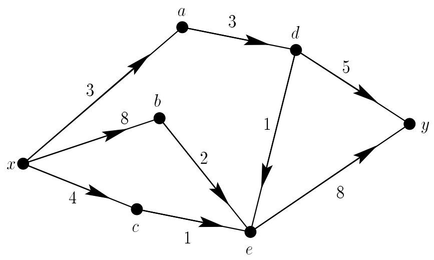


Figure 13.6: A directed network.

Algorithm 13.3: Dijkstra's Shortest Path Algorithm[5]

Input: A directed network D and vertices x and y from D.
Output: A shortest path from x to y or the message that there is no path from x to y.

Step 1. Initially, place vertex x in the class W, let $B = \emptyset$, and let $d(x, x) = 0$.

Step 2.

Step 2.1. For each vertex u in W and each vertex v not in W, let $\alpha(u, v) = d(x, u) + w(u, v)$. Find u in W and v not in W such that $\alpha(u, v)$ is minimized. (Choose arbitrarily in case of ties.)

Step 2.2. If the minimum in Step 2.1 is ∞, stop and give the message that there is no path from x to y.

Step 2.3. If the minimum in Step 2.1 is not ∞, place v in W (v is the next vertex chosen), place arc (u, v) in B, and set $d(x, v) = \alpha(u, v)$.

Step 3. If y is not yet in W, return to Step 2. If y is in W, stop. A shortest path from x to y can be found by using the unique path of arcs of B that goes from x to y. This can be found by working backward from y.

Let us illustrate this algorithm on the directed network of Figure 13.6. The successive steps in the algorithm are shown in Table 13.3. [The table does not show the $\alpha(u, v)$ that are infinite.] In iteration 2, for instance, arc (x, a) has the smallest α value of those computed, so a is added to W and (x, a) to B and $d(x, a)$ is taken to be $\alpha(x, a)$; in iteration 4, $\alpha(c, e)$ is minimum, so e is added to W and (c, e) to B, and $d(x, e)$ is taken to be $\alpha(c, e)$; and so on. At the seventh iteration, vertex y has been added to W. We now work backward from y, using arcs in B. We find that we got to y from d, to d from a, and to a from x. Thus, we find the path $x, (x, a), a, (a, d), d, (d, y), y$, that is a shortest path from x to y.

Again we comment on the computational complexity of the algorithm. The algorithm takes at most n iterations, where n is the number of vertices of the directed network. For each iteration adds a vertex. Each iteration involves additions $d(x, u) + w(u, v)$, one for each pair of vertices u in W and v not in W, that is, at

[5]At this point, we present the basic idea of the algorithm. Later, we describe how to improve it.

Table 13.3: Dijkstra's Algorithm Applied to the Directed Network of Figure 13.6

Iteration	Finite numbers $\alpha(u,v)$ computed	Added to W	Added to B	New $d(x,v)$
1	None	x	—	$d(x,x) = 0$
2	$\alpha(x,a) = 0+3 = 3$	a	(x,a)	$d(x,a) = 3$
	$\alpha(x,b) = 0+8 = 8$			
	$\alpha(x,c) = 0+4 = 4$			
3	$\alpha(x,b) = 0+8 = 8$	c	(x,c)	$d(x,c) = 4$
	$\alpha(x,c) = 0+4 = 4$			
	$\alpha(a,d) = 3+3 = 6$			
4	$\alpha(x,b) = 0+8 = 8$	e	(c,e)	$d(x,e) = 5$
	$\alpha(a,d) = 3+3 = 6$			
	$\alpha(c,e) = 4+1 = 5$			
5	$\alpha(x,b) = 0+8 = 8$	d	(a,d)	$d(x,d) = 6$
	$\alpha(a,d) = 3+3 = 6$			
	$\alpha(e,y) = 5+8 = 13$			
6	$\alpha(x,b) = 0+8 = 8$	b	(x,b)	$d(x,b) = 8$
	$\alpha(e,y) = 5+8 = 13$			
	$\alpha(d,y) = 6+5 = 11$			
7	$\alpha(e,y) = 5+8 = 13$	y	(d,y)	$d(x,y) = 11$
	$\alpha(d,y) = 6+5 = 11$			

most n^2 additions in all. Also, finding the smallest number among the at most n^2 numbers $d(x,u)+w(u,v)$ can be accomplished in at most n^2 comparisons. Hence, at each iteration the algorithm takes at most $2n^2$ steps. Altogether, the algorithm takes at most $2n^3$ steps, a polynomial bound. In the terminology of Section 2.18, we have an $O(n^3)$ algorithm.

Actually, by several simple changes in the procedure, the algorithm can be improved to be an $O(n^2)$ algorithm.[6] Suppose that we let u_k be the vertex that is the kth closest. We take $u_1 = x$. Let $\alpha_1(v) = \infty$ for all $v \neq x, \alpha_1(x) = 0$, and define

$$\alpha_{k+1}(v) = \min\{\alpha_k(v), \alpha_k(u_k) + w(u_k, v)\}. \tag{13.2}$$

Then it is easy to show (Exercise 25) that

$$\alpha_{k+1}(v) = \min\{d(x,u_j) + w(u_j, v) : j = 1, 2, \ldots, k\}. \tag{13.3}$$

At each iteration, we compute the n numbers $\alpha_{k+1}(v)$. For each v we do this by first doing one addition and then finding one minimum of two numbers. Then we find the minimum of the set of numbers $\alpha_{k+1}(v)$, that can be done in at most n

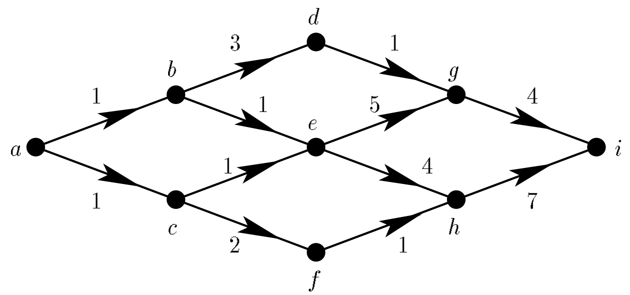
[6] The rest of this subsection may be omitted.

steps. The vertex v that gives us this minimum is u_{k+1}. (Choose arbitrarily in case of ties.) The total number of steps at the iteration that adds u_{k+1} is now at most $n+2$, so the total number of steps in the entire algorithm is at most n^2+2n. If we just use the labels $\alpha_k(v)$, we will have to be more careful to compute the set B. Specifically, at each stage that $\alpha_{k+1}(v)$ decreases, we will have to keep track of the vertex u_j such that $\alpha_{k+1}(v)$ is redefined to be $d(x, u_j) + w(u_j, v)$. At the point when u_{k+1} is taken to be v, the corresponding (u_j, v) will be added to B. Details of the improved version of Dijkstra's Algorithm are left to the reader. For a discussion of various implementations of and improvements to Dijkstra's Algorithm under different assumptions, see Ahuja, Magnanti, and Orlin [1993].

13.2.3 Applications to Scheduling Problems

Although the shortest route problem has been formulated in the language of distances, the weights do not have to represent distances. Many applications of the shortest route algorithm apply to situations where the arcs correspond to activities of some kind and the weight on an arc corresponds to the cost of the activity. The problem involves finding a sequence of activities that begins at a starting point, accomplishes a desired objective, and minimizes the total cost. Alternatively, the weight is the time involved to carry out the activity and the problem seeks a sequence of activities that accomplishes the desired objective in a minimum amount of time. We encountered similar problems in Section 11.6.3. In these situations the network is sometimes called a PERT (Project Evaluation and Review Technique) network or a CPM (Critical Path Method) network. We illustrate these ideas with the following examples.

Example 13.6 A Manufacturing Process A manufacturing process starts with a piece of raw wood. The wood must be cut into shape, stripped, have holes punched, and be painted. The cutting must precede the hole punching and the stripping must precede the painting. Suppose that cutting takes 1 unit of time; stripping takes 1 unit of time; painting takes 2 units of time for uncut wood and 4 units of time for cut wood; and punching holes takes 3 units of time for unstripped wood, 5 units for stripped but unpainted wood, and 7 units for painted wood. The problem is to find the sequence of activities that will allow completion of the process in as short a period of time as possible. We can let the vertices of a directed network D represent stages in the manufacturing process, for example, raw wood; cut wood; cut and holes punched; cut, stripped, and holes punched; and so on. We take an arc from stage i to stage j if a single activity can take the process from stage i to stage j. Then we put a weight on arc (i, j) corresponding to the amount of time required to go from i to j. The directed network in question is shown in Figure 13.7. There are, for example, arcs from b to d and e, because cut wood can next either have holes punched or be stripped. The arc (b, d) has weight 3 because it corresponds to punching holes in unstripped wood. We seek a shortest path in this directed network from the raw wood vertex (a) to the stripped, cut, holes punched, and painted vertex (i). ■



Key:

a. raw wood	d. cut and holes punched	g. cut, holes punched, and stripped
b. cut	e. cut and stripped	h. cut, stripped, and painted
c. stripped	f. stripped and painted	i. stripped, cut, holes punched, and painted

Figure 13.7: Directed network for a manufacturing process. The vertices correspond to stages in the process, arcs to activities taking the process from stage to stage, and weights to times required for the activities.

Example 13.7 Inspection for Defective Products on a Production Line (Ahuja, Magnanti, and Orlin [1993], White [1969]) A production line has n stages and a batch of items is run through the production line at the same time. At each stage, defects might be introduced. Inspection at each stage is costly, but only inspecting at the end of stage n means that a lot of production time might be spent on continuing to produce items that already have defects. What is the optimal plan for inspecting for defective items? Let us assume that inspections are "perfect" in the sense of uncovering all defects and that defective items are unrepairable and are immediately discarded. Suppose that B is the size of the batches of items sent through the production line, a_i is the probability of producing a defect at stage i, p_i is the cost of manufacturing one item in stage i, $f_{i,j}$ is the fixed cost of inspecting the batch of items after stage j given that the last inspection was after stage i, and $g_{i,j}$ is the cost of inspecting one item after stage j given that the last inspection was after stage i. (Note that the costs of inspecting depend on when the last inspection was made since defects could have been introduced at all intervening stages.) We can calculate the expected number of nondefective items at the end of stage i, B_i, as

$$B_i = B \prod_{k=1}^{i} (1 - a_k).$$

To formulate our question as a shortest route problem, let the vertices of a directed network be the stages 1, 2, ..., n with an extra vertex 0 corresponding to the beginning, and let the arcs go from stage i to each stage $j > i$. A path corresponds to the plan of inspecting its vertices (except vertex 0). The cost we associate with arc (i, j) is given by

$$c_{i,j} = f_{i,j} + B_i g_{i,j} + B_i \sum_{k=i+1}^{j} p_k. \tag{13.4}$$

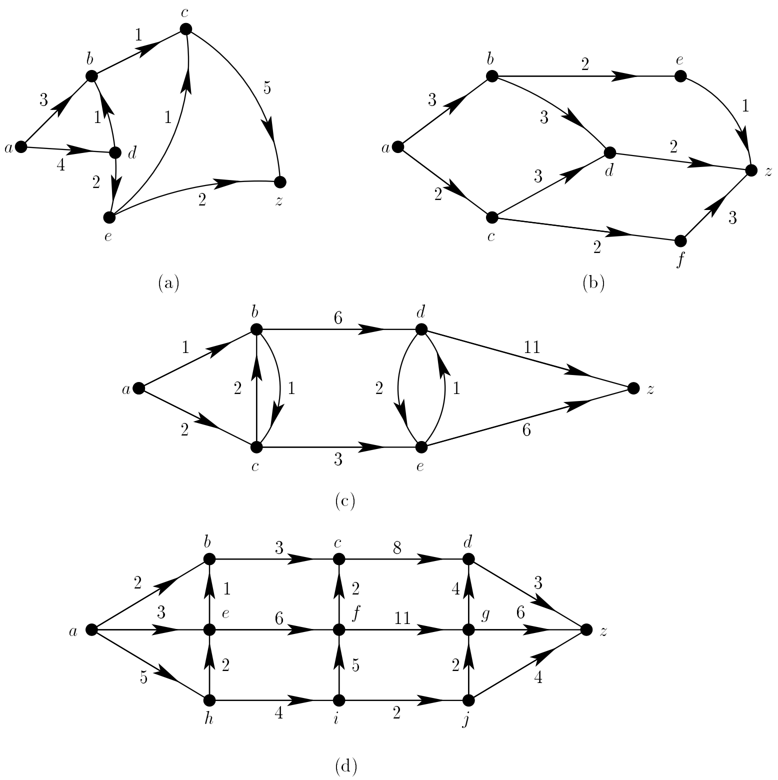


Figure 13.8: Directed networks for exercises of Sections 13.2 and 13.3.

This represents the total cost in stages $i+1, i+2, \ldots, j$ if we inspect in stage i and next in stage j. The first two terms are the fixed and variable costs of inspection after stage j, and the third term is the production cost in the stages $i+1$ through j. A shortest path from vertex 0 to vertex n gives us a least expensive inspection schedule. ∎

EXERCISES FOR SECTION 13.2

1. Show that in a digraph, a shortest path from x to y must be a simple path.
2. Show that in a graph, a shortest chain from x to y must be a simple chain.
3. In each directed network of Figure 13.8, use Dijkstra's Algorithm (as described in Algorithm 13.3) to find a shortest path from a to z.
4. Find the most efficient manufacturing process in the problem of Example 13.6.
5. A product must be ground, polished, weighed, and inspected. The grinding must precede the polishing and the weighing, and the polishing must precede the inspection. Grinding takes 7 units of time, polishing takes 10 units of time, weighing takes

1 unit of time for an unpolished product and 3 units of time for a polished one, and inspection takes 2 units of time for an unweighed product and 3 units of time for a weighed one. What is the fastest production schedule?

6. A company wants to invest in a fleet of automobiles and is trying to decide on the best strategy for how long to keep a car. After 5 years it will sell all remaining cars and let an outside firm provide transportation. In planning over the next 5 years, the company estimates that a car bought at the beginning of year i and sold at the beginning of year j will have a net cost (purchase price minus trade-in allowance, plus running and maintenance costs) of a_{ij}. The numbers a_{ij} in thousands of dollars are given by the following matrix:

$$(a_{ij}) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 4 & 6 & 9 & 12 & 20 \\ & 5 & 7 & 11 & 16 \\ & & 6 & 8 & 13 \\ & & & 8 & 11 \\ & & & & 10 \end{pmatrix} \end{array}.$$

To determine the cheapest strategy for when to buy and sell cars, we let the vertices of a directed network be the numbers $1, 2, 3, 4, 5, 6$, include all arcs (i, j) for $i < j$, and let weight $w(i, j)$ be a_{ij}. The arc (i, j) has the interpretation of buying a car at the beginning of year i and selling it at the beginning of year j. Find the cheapest strategy.

7. In Example 13.5, suppose that 8 beam lengths are available. Each length corresponds to a vertex in a directed network N. The following chart contains the demand (D_j), inventory facility cost (K_j), and beam cost (C_j) for each of the beam lengths:

j	0	1	2	3	4	5	6	7
L_j	0	2	4	8	16	32	64	128
D_j	0	4	6	16	2	8	5	4
K_j	0	1	2	3	4	5	6	7
C_j	0	10	19	27	34	40	45	49

The costs $C_{i,j}$ for most of the arcs in N are given by the following matrix:

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} & 4 & 10 & 26 & X & 36 & 41 & 45 \\ & & 6 & 22 & 24 & 32 & 37 & 41 \\ & & & 16 & 18 & Y & 31 & 35 \\ & & & & 2 & 10 & 15 & Z \\ & & & & & 8 & 13 & 17 \\ & & & & & & 5 & 9 \\ & & & & & & & 4 \\ & & & & & & & \end{pmatrix} \end{array}.$$

(a) Using (13.1), compute the missing arc costs (X, Y, Z) in the matrix.

(b) Find a shortest path from vertex 0 to vertex 7 in the directed network N. Which beams should be kept in inventory?

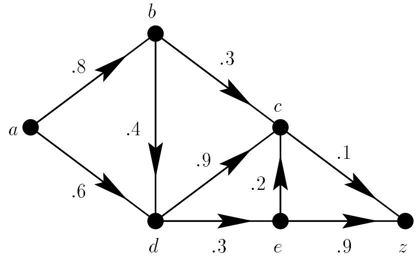


Figure 13.9: Communication network.

8. Figure 13.9 shows a communication network [arc (i, j) corresponds to a link over which i can communicate directly with j]. Suppose that the weight p_{ij} on the arc (i, j) is the probability that the link from i to j is operative. Assuming that defects in links occur independent of each other, the probability that all the links in a path are operative is the product of the link probabilities. Find the most reliable path from a to z. (*Hint:* Consider $-\log p_{ij}$.)

9. Suppose that you plan to retire in 20 years and want to invest \$100,000 toward retirement and you know that at year i, there will be available investment alternatives $A(i, 1)$, $A(i, 2)$, $\ldots$, $A(i, k_i)$, with investment $A(i, j)$ giving $r(i, j)$ rate of return and reaching maturity in $y(i, j)$ years. (This is, of course, very oversimplified.) Formulate the problem of finding an optimal investment strategy as a shortest route problem.

10. It is not efficient to consider all possible paths from i to j in searching for a shortest path. To illustrate the point, suppose that D has n vertices and an arc from every vertex to every other vertex. If x and y are any two vertices of D, find the number of paths from x to y.

11. If the cost of each arc increases by u units, does the cost of the shortest route from a to z increase by a multiple of u? Why?

12. In Example 13.4, suppose that we want to create a paragraph with 6 words. The following matrix shows the costs, $c_{i,j}$, for each i and j, $i < j \le 7$. ("Word" 7 corresponds to the dummy vertex.)

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
\begin{pmatrix}
 & 8 & 4 & 0 & 5 & 10 & 15 \\
 & & 7 & 3 & 2 & 7 & 12 \\
 & & & 7 & 3 & 3 & 8 \\
 & & & & 10 & 6 & 9 \\
 & & & & & 2 & 4 \\
 & & & & & & 0 \\
 & & & & & &
\end{pmatrix}
\end{array}$$

(a) Build the network N whose vertices are the words (including "word" 7), whose arcs are (i, j), with $j > i$, and where each arc (i, j) has weight $c_{i,j}$.

(b) Find the shortest path in N from word 1 to word 7.

(c) Describe how the 6-word paragraph will be formatted.

13. In Example 13.7, suppose that there are 6 stages in a production line. The total cost of inspecting at stage i and next in stage j is given by Equation (13.4) using the following fixed, variable, and production cost equations:

$$\begin{aligned} f_{i,j} &= (j-i) \\ g_{i,j} &= (j-i)\frac{i}{j} \\ (p_k) &= (5,8,4,7,2,6) \\ B_i &= .8(1.1)^{-i} \end{aligned}$$

Find a least expensive production schedule.

14. (Bondy and Murty [1976]) A wolf, a goat, and a cabbage are on one bank of a river and a boatman will take them across, but can only take one at a time. The wolf and the goat cannot be left on one bank of the river together, nor can the goat and the cabbage. How can the boatman get them across the river in the shortest amount of time?

15. (Bondy and Murty [1976]) A man has a full 8-gallon jug of wine and empty jugs that hold 5 and 3 gallons, respectively. What is the fewest number of steps required for the man to divide the wine into two equal amounts?

16. (Ahuja, Magnanti, and Orlin [1993]) You own a summer rental property and at the beginning of the summer season, you receive a number of requests for the property. Each request gives you the days the rental would start and finish. Your rates for rentals of x days are given by a function $f(x)$. How do you decide which requests to accept in order to maximize your income for the summer? Formulate the problem as a shortest route problem.

17. (Ahuja, Magnanti, and Orlin [1993], Ravindran [1971]) A library wishes to minimize the cost of shelving for books in a special collection. A certain number of books in the collection have height H_i for $i = 1, 2, \ldots, n$, with $H_i < H_{i+1}$. We want to shelve books of the same height together and shelve books in increasing order of height. Assume that the books have known thickness. This then determines the total length L_i of shelving needed for books of height H_i. There is a fixed cost F_i of ordering shelves of height H_i that is independent of the length of shelving ordered, and there is a cost C_i per inch of length of shelving of height H_i. Thus, if we order shelving of height H_i of total length x_i, the cost of that shelving is $F_i + C_i x_i$. Note that since there is a fixed cost per order, we might not order shelving of each height since we can always shelve books of height H_i in shelving of height H_j for $j > i$. How much shelving of each length should we order? Formulate this as a shortest path problem.

18. Recall the problem of planning an airplane trip (Example 13.3). Besides the time spent in the air flying, consider the case where there is a "cost" associated with each city and it represents the average layover time at that city.

    (a) Formulate this "new" problem as a standard shortest path problem with only arc weights.

    (b) Solve this problem for the directed network of Figure 13.5 if the costs (average layover times) for each city are given by

City	a	b	c	d	e
Cost	1	2	5	4	3

.

19. If all arcs in a network have different weights, does the network necessarily have a unique shortest path? Give a proof or counterexample.

20. (Ahuja, Magnanti, and Orlin [1993]) Our formulation of the structural beam inventory problem in Example 13.5 has a variety of oversimplifications.

    (a) What does it assume happens with a beam of length 5 if it is cut to make a beam of length 2?

    (b) What other oversimplifications are there in this example?

    (c) If we can cut a single beam into multiple beams of the same length (e.g., a beam of length 14 into three beams of length 4, with some wastage), how might we modify the analysis?

    (d) How might we modify the analysis if we can not only cut a beam into multiple beams of the same length but can also sell the leftover for scrap value, measured say at $\$d$ per unit length?

21. Let D be a digraph. We define distance $\bar{d}(x, y)$ to be the length of the shortest path from x to y in the directed network obtained from D by putting a weight of 1 on each arc. Discuss how to use the powers of the adjacency matrix of D (Section 3.7) to calculate $\bar{d}(x, y)$.

22. A shortest path from a to z in a network is not necessarily unique, nor does it have the smallest number of arcs in a path from a to z. However, among all shortest paths from a to z, does Dijkstra's Algorithm produce one with the smallest number of arcs? Why?

23. In Dijkstra's Algorithm, show that if direction of arcs is disregarded, the arcs of the set B define a tree rooted at vertex x.

24. Describe a shortest path algorithm for finding, in a directed network D, the shortest paths from a given vertex x to each other vertex.

25. Show that α_{k+1} defined by (13.2) satisfies (13.3).

26. (a) Write a careful description of the $O(n^2)$ version of Dijkstra's Algorithm using the labels $\alpha_k(v)$.

    (b) Apply the algorithm to each directed network of Figure 13.8 to find a shortest path from a to z.

27. In Example 13.4 and Exercise 12, we did not consider the possibility of hyphenating a word to better optimize a paragraph's appearance. How could you modify the directed network formulation of the document processing system in Example 13.4 to accommodate hyphenation?

13.3 NETWORK FLOWS

13.3.1 The Maximum-Flow Problem

Suppose that D is a directed network and let $c_{ij} = w(i, j)$ be the nonnegative weight on the arc (i, j). In this section we call c_{ij} the *capacity* of the arc (i, j) and interpret it as the maximum amount of some commodity that can "flow" through the arc per unit time in a steady-state situation. The commodity can be finished products, messages, people, oil, trucks, letters, electricity, and so on.

Flows are permitted only in the direction of the arc, that is, from i to j. We fix a *source* vertex s and a *sink* vertex t, and think of a flow starting at s and ending at t. (In all of our examples, s will have no incoming arcs and t no outgoing arcs. But it is not necessary to assume these properties.) Let x_{ij} be the flow through arc (i,j). Then we require that

$$0 \leq x_{ij} \leq c_{ij}. \tag{13.5}$$

This says that flow is nonnegative and cannot exceed capacity. We also have a *conservation law*, which says that for all vertices $i \neq s, t$, what goes in must go out; that is,

$$\sum_j x_{ij} = \sum_j x_{ji}, \;\; i \neq s, t. \tag{13.6}$$

A set of numbers $x = \{x_{ij}\}$ satisfying (13.5) and (13.6) is called an (s,t)-*feasible flow*, or an (s,t)-*flow*, or just a *flow*. For instance, consider the directed network of Figure 13.10. In part (a) of the figure, the capacities c_{ij} are shown on each arc with the numbers in squares. In part (b), a flow is shown with the numbers in circles. Note that (13.5) and (13.6) hold. For instance, $c_{24} = 2$ and $x_{24} = 0$, so $0 \leq x_{24} \leq c_{24}$. Also, $\sum_j x_{3j} = x_{35}+x_{36}+x_{37} = 2+1+0 = 3$, and $\sum_j x_{j3} = x_{23} = 3$; and so on.

Suppose that x defines a flow. Let

$$v_t = \sum_j x_{jt} - \sum_j x_{tj}$$

and

$$v_s = \sum_j x_{js} - \sum_j x_{sj}.$$

Note that if we sum the terms

$$\sum_j x_{ji} - \sum_j x_{ij}$$

over all i, Equation (13.6) tells us that all of these terms are 0 except for the cases $i = s$ and $i = t$. Thus,

$$\sum_i \left[\sum_j x_{ji} - \sum_j x_{ij} \right] = v_t + v_s. \tag{13.7}$$

However, the left-hand side of (13.7) is the same as

$$\alpha = \sum_{i,j} x_{ji} - \sum_{i,j} x_{ij}. \tag{13.8}$$

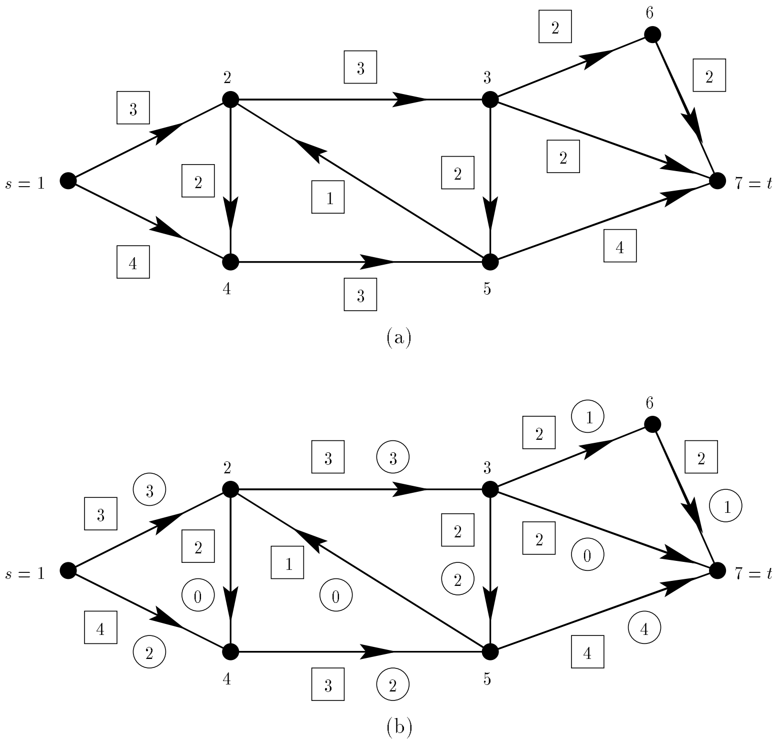


Figure 13.10: Capacities are shown in squares, flows in circles.

Since both sums in (13.8) simply sum all flows over all arcs, they are equal and hence $\alpha = 0$. We conclude that $v_s = -v_t$. Thus, there is a number v such that

$$\sum_j x_{ji} - \sum_j x_{ij} = \begin{cases} -v & \text{if } i = s \\ v & \text{if } i = t \\ 0 & \text{if } i \neq s, t. \end{cases} \tag{13.9}$$

The number v in our example of Figure 13.10(b) is 5. It is called the *value* of the flow. The value represents the total amount of the commodity that can be sent through the network in a given period of time if this flow is used. The problem we will consider is this: Find a flow that has maximum value, a *maximum flow.*

The classic reference on the theory of flows in networks is the book by Ford and Fulkerson [1962]. Other comprehensive references are Ahuja, Magnanti, and Orlin [1993], Berge and Ghouila-Houri [1965], Cook, *et al.* [1998], Frank and Frisch [1971], Hu [1969], Iri [1969], Lawler [1976], Minieka [1978], and Papadimitriou and Steiglitz [1982].

Although our presentation is for directed networks, everything in this section will apply to undirected networks. Simply replace each (undirected) edge $\{i, j\}$ with capacity c_{ij} by two arcs (i, j) and (j, i) and let each of these arcs have the same capacity as edge $\{i, j\}$.

A large number of combinatorial optimization problems can be formulated as network flow problems. The next example illustrates this point. For many practical applications of network flows, the reader should consult such books and survey articles as Ahuja, Magnanti, and Orlin [1993], Aronson [1989], Bazaraa, Jarvis, and Sherali [1990], Glover and Klingman [1977], and Gondran and Minoux [1984].

Example 13.8 College Admissions The admissions officer at a university has a list of applicants. Each applicant has provided a list of potential majors that she or he might consider. The job of the admissions officer is to admit a first round of applicants so that every major represented among the applicants' interests has exactly one student admitted and so that the number of students admitted from state i is no more than q_i. Can this be done? Build a directed network whose vertices are the applicants, the majors, and the states, plus a source vertex s and a sink vertex t. Include arcs (s, m) for every major m, (m, a) whenever applicant a is interested in major m, (a, i) if applicant a lives in state i, and (i, t) for every state i. Let arc (i, t) have capacity q_i and all other arcs have capacity 1. We seek a maximum flow from s to t in this network. If the maximum flow value equals the number of majors represented among the applicants' interests, it is possible to solve the problem. If not, it cannot. The proof is left to the reader (Exercise 31). ■

13.3.2 Cuts

Let S and T be two sets that partition the vertex set of the digraph D, that is, $V(D) = S \cup T$ and $S \cap T = \emptyset$. We refer to the partition (S, T) as a *cut.* Equivalently, we think of the cut as the set C of all arcs that go from vertices in S to vertices in T. The set C is called a cut because after the arcs of C are removed, there is no path from any vertex of S to any vertex of T. If x is any vertex of S and y any vertex of T, C is called an (x, y)-*cut.* For instance, in Figure 13.10, $S = \{1, 2\}$ and $T = \{3, 4, 5, 6, 7\}$ is a cut, and this is equivalent to the set of arcs $\{(1, 4), (2, 3), (2, 4)\}$. [We do not include arc $(5, 2)$ as it goes from a vertex in T to a vertex in S.] Notice that if there are more than two vertices, there are always at least two (x, y)-cuts: $S = \{x\}, T = V(D) - \{x\}$, and $S = V(D) - \{y\}, T = \{y\}$.

In a directed network, if (S, T) is a cut, we can define its *capacity* as $c(S, T) = \sum_{i \in S} \sum_{j \in T} c_{ij}$. In our example above, $c(S, T) = c_{14} + c_{23} + c_{24} = 9$. Notice that in our example, the value of the flow is 5 and the capacity of this cut is 9, which is greater. The next result shows that this is no accident.

Theorem 13.3 In a directed network, the value of any (s, t)-flow is $\leq$ the capacity of any (s, t)-cut.

Proof. Let x be an (s, t)-flow and (S, T) be an (s, t)-cut. Note that $\sum_j x_{ij} - \sum_j x_{ji}$ is 0 if $i \in S$ and $i \neq s$, and is v if $i = s$. Thus,

$$v = \sum_j x_{sj} - \sum_j x_{js},$$

$$
\begin{aligned}
&= \sum_{i\in S}\left[\sum_{j} x_{ij} - \sum_{j} x_{ji}\right],\\
&= \sum_{i\in S}\sum_{j\in S}[x_{ij}-x_{ji}] + \sum_{i\in S}\sum_{j\in T}[x_{ij}-x_{ji}],\\
&= \sum_{i\in S}\sum_{j\in S} x_{ij} - \sum_{i\in S}\sum_{j\in S} x_{ji} + \sum_{i\in S}\sum_{j\in T}[x_{ij}-x_{ji}], \qquad (13.10)\\
&= \sum_{i\in S}\sum_{j\in T}(x_{ij}-x_{ji}), \qquad (13.11)
\end{aligned}
$$

because the first two terms of (13.10) are the same. Thus, by (13.11), the value of any flow is the net flow through any cut. Since $x_{ij} \leq c_{ij}$ and $x_{ij} \geq 0$, we have

$$x_{ij} - x_{ji} \leq x_{ij} \leq c_{ij},$$

so (13.11) implies that

$$v \leq \sum_{i\in S}\sum_{j\in T} c_{ij} = c(S,T).$$

Q.E.D.

Corollary 13.3.1 In a directed network, if (S,T) is an (s,t)-cut and x is an (s,t)-flow, then

$$v = \sum_{i\in S}\sum_{j\in T}[x_{ij}-x_{ji}].$$

Proof. This is a corollary of the proof. Q.E.D.

Figure 13.11 shows another (s,t)-flow in the network of Figure 13.10(a). This flow has value 6. Notice that if $S = \{1,2,4\}$ and $T = \{3,5,6,7\}$, then $c(S,T) = c_{23} + c_{45} = 6$. Now there can be no (s,t)-flow with value more than the capacity of this cut, that is, 6. Hence, the flow shown is a maximum flow. Similarly, this cut must be an (s,t)-cut of minimum capacity, a *minimum cut.* Indeed, the same thing must be true any time we find a flow and a cut where the value of the flow is the same as the capacity of the cut.

Theorem 13.4 If x is an (s,t)-flow with value v and (S,T) is an (s,t)-cut with capacity $c(S,T)$, and if $v = c(S,T)$, then x is a maximum flow and (S,T) is a minimum cut.

We have used reasoning similar to this in Theorem 12.4, where we argued that since the number of edges in any matching of a graph is less than or equal to the number of vertices in any covering, if we ever find a matching and a covering of the same size, the matching must be maximum and the covering must be minimum.

As with flows, we have emphasized cuts in directed networks in this section. However, we can speak of cuts in undirected networks N. As for directed networks, a *cut* in N is a partition of the vertices into two sets S and T or, equivalently, the

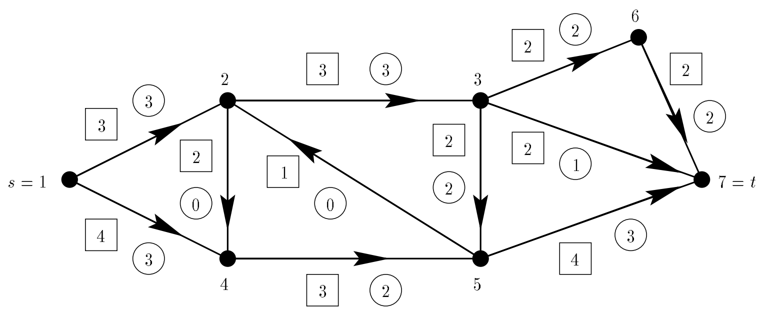


Figure 13.11: Another flow for the directed network of Figure 13.10(a).

set of all edges going from a vertex in S to a vertex in T. If $s \in S$ and $t \in T$, then (S, T) is called an (s,t)-*cut*.

Finding the minimum cut in a network is often of interest in its own right, as the next example shows.

Example 13.9 Distributed Computing on a Two-Processor Computer (Ahuja, Magnanti, and Orlin [1993], Stone [1977]) In the simplest version of a distributed computing system, there are two processors in the system and we wish to assign different modules (subroutines) of a program to the processors in such a way that minimizes the costs of interprocessor communication and computation. The cost of executing module i on processor 1 is a_i and on processor 2 is b_i. There is a relatively high interprocessor communication cost $c_{i,j}$ if modules i and j are assigned to different processors and these modules interact. We seek to allocate modules of the program to the two processors so as to minimize the total cost, which is the sum of the processing cost and the interprocessor communication cost. Build an undirected network N with a source s representing processor 1 and sink t representing processor 2 and a vertex for every module of the program. Add edges $\{s,i\}$ for every module i with capacity b_i, and edges $\{i,t\}$ for every module i with capacity a_i. Finally, if modules i and j interact during program execution, add an edge $\{i,j\}$ with capacity $c_{i,j}$. Let $E(N)$ be the set of edges in N. We note that there is a one-to-one correspondence between assignments of modules to processors and (s,t)-cuts and that the capacity of a cut is the same as the cost of the corresponding assignment. To see this, suppose that A_1 consists of all the modules assigned to processor 1 and similarly A_2. The cost of this assignment is

$$\sum_{i \in A_1} a_i + \sum_{i \in A_2} b_i + \sum_{\substack{\{i,j\} \in E(N), \\ i \in A_1,\, j \in A_2}} c_{i,j}.$$

The (s,t)-cut corresponding to this assignment consists of the sets $\{s\} \cup A_1$, $\{t\} \cup A_2$. The cut thus contains edges $\{i,t\}$ with capacity a_i for $i \in A_1$, $\{s,i\}$ with capacity b_i for $i \in A_2$, and $\{i,j\}$ with capacity $c_{i,j}$ for i, j interacting modules with i in A_1 and j in A_2. This is exactly the cost of the assignment. Hence, the minimum-cost assignment we are seeking corresponds to the minimum (s,t)-cut in the network N. (We note that this example oversimplifies the practical problem. In fact, the processing cost and the interprocessor communication cost are measured in different units and a major challenge in practice is to "scale" these costs so that they can be compared.[7]) ■

13.3.3 A Faulty Max-Flow Algorithm

Our goal is to describe an algorithm for finding the maximum flow. First we present an intuitive, although faulty, technique. If P is a simple path from s to t, we call it an *(s,t)-path* and we let the corresponding *unit flow* x^P be given by

$$x_{ij}^P = \begin{cases} 1 & \text{if arc } (i,j) \text{ is in } P \\ 0 & \text{otherwise.} \end{cases}$$

The idea is to add unit flows successively.

Let us say that an arc (i,j) is *unsaturated* by a flow x if $x_{ij} < c_{ij}$, and let us define the *slack* by $s_{ij} = c_{ij} - x_{ij}$. The basic point is that if θ is the minimum slack among arcs of P, we can add the unit flow x^P a total of θ times and obtain a new flow with value increased by θ. Here is the algorithm.

Algorithm 13.4: Max-Flow Algorithm: First Attempt

Input: A directed network with a source s and a sink t.
Output: A supposedly maximum (s,t)-flow x.

Step 1. Set $x_{ij} = 0$, all i, j.

Step 2.

Step 2.1. Find an (s,t)-path P with all arcs unsaturated. If none exists, go to Step 3.

Step 2.2. Compute the slack of each arc of P.

Step 2.3. Compute θ, the minimum slack of arcs of P.

Step 2.4. Redefine x by adding θ to the flow on arc (i,j) if (i,j) is in P. Return to Step 2.1.

Step 3. Stop with the flow x.

Let us apply this algorithm to the directed network of Figure 13.10(a). Figure 13.12 shows the successive flows defined by the following iterations. In the first iteration, we take $x_{ij} = 0$, all i, j. We then note that $P = 1,4,5,7$ is an (s,t)-path with all arcs unsaturated, and the corresponding θ is 3, for $s_{14} = c_{14} - x_{14} = 4 - 0 = 4$, $s_{45} = c_{45} - x_{45} = 3 - 0 = 3$, and $s_{57} = c_{57} - x_{57} = 4 - 0 = 4$. Hence, we increase x_{14}, x_{45}, and x_{57} by $\theta = 3$, obtaining the second flow in Figure 13.12.

[7] We thank David Roberts for this observation.

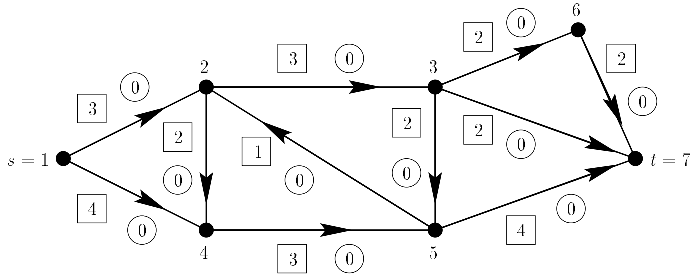
In this flow, $P = 1, 2, 3, 7$ is an (s, t)-path with all unsaturated arcs. Its θ is 2, for $s_{12} = 3, s_{23} = 3$, and $s_{37} = 2$. Thus, we increase x_{12}, x_{23}, and x_{37} by $\theta = 2$, obtaining the third flow in Figure 13.12. In this flow, $P = 1, 2, 3, 6, 7$ has all arcs unsaturated and $\theta = 1$, since $s_{12} = c_{12} - x_{12} = 3 - 2 = 1, s_{23} = 1, s_{36} = 2$, and $s_{67} = 2$. Thus, we increase x_{12}, x_{23}, x_{36}, and x_{67} by $\theta = 1$, obtaining the fourth flow in Figure 13.12. Since in this flow there are no more (s, t)-paths with all arcs unsaturated, we stop. Note that we have obtained a flow of value equal to 6, which we know to be a maximum.

Unfortunately, this algorithm does not necessarily lead to a maximum flow. Let us consider the same directed network. Figure 13.13 shows the successive steps in another use of this algorithm using different (s, t)-paths. Notice that after obtaining the fourth flow, we can find no (s, t)-path with all arcs unsaturated. Thus, the algorithm stops. However, the flow obtained has value only 5, which is not maximum. What went wrong?

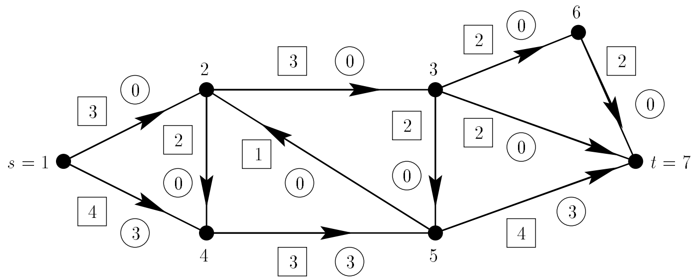
Consider the minimum cut $S = \{1, 2, 4\}, T = \{3, 5, 6, 7\}$. One problem is that one of the unit flow paths used, $1, 4, 5, 2, 3, 7$, crosses this cut backward, that is, from T to S; equivalently, it crosses it forward from S to T twice. Thus, the 1 unit of flow on P uses up 2 units of capacity—too much capacity is used. We would be better off if we got rid of some of the flow going backward and used more in a forward direction. This suggests a way to improve on our algorithm.

13.3.4 Augmenting Chains

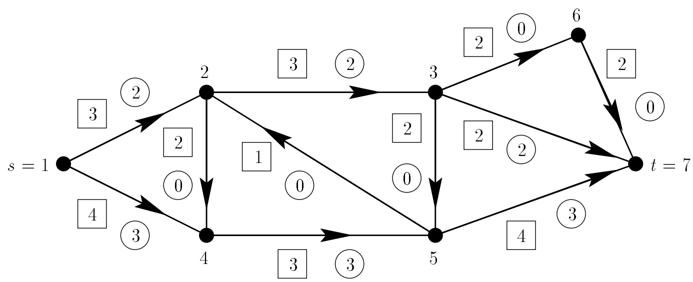
Consider the graph (multigraph) G obtained from a directed network by disregarding directions on arcs. Let C be a chain from s to t in this graph. An arc (i, j) of D is said to be a *forward arc* of C if it is followed from i to j, and a *backward arc* otherwise. For instance, in the directed network D of Figure 13.10, one chain is $1, 4, 5, 3, 6, 7$. Here, arc $(1, 4)$ is a forward arc, but arc $(3, 5)$ is backward. If x is a flow, C is said to be a *flow-augmenting chain*, or just an *augmenting chain, relative to x* if $x_{ij} < c_{ij}$ for each forward arc and $x_{ij} > 0$ for each backward arc. The unit flow paths with no saturated arcs, which we discussed in Section 13.3.3, all correspond to flow-augmenting chains with no backward arcs. The chain $1, 4, 5, 3, 6, 7$ in Figure 13.10(b) is a flow-augmenting chain relative to the flow shown in the figure. For the forward arcs $(1, 4), (4, 5), (3, 6)$, and $(6, 7)$ are all under capacity and the backward arc $(3, 5)$ has positive flow. We can improve on the value of the flow by decreasing the backward flow and increasing the forward flow. Specifically, we record x_{ij} for all backward arcs on the chain and compute and record the slack $s_{ij} = c_{ij} - x_{ij}$ for all forward arcs. If λ is the minimum of these recorded numbers, λ is called the *capacity* of the augmenting chain. We then increase each x_{ij} on a forward arc by λ and decrease each x_{ij} on a backward arc by λ. By choice of λ, each new x_{ij} is still nonnegative and is no higher than the capacity c_{ij}. Moreover, the conservation law (13.6) still holds. Next, we observe that the value of the flow increases by λ, for the chain starts with an edge from s and ends with an edge to t. If the edge from s is forward, the flow out of s is increased; if it is backward, the flow into s is decreased. In any case, the value or net flow out of s is increased. A



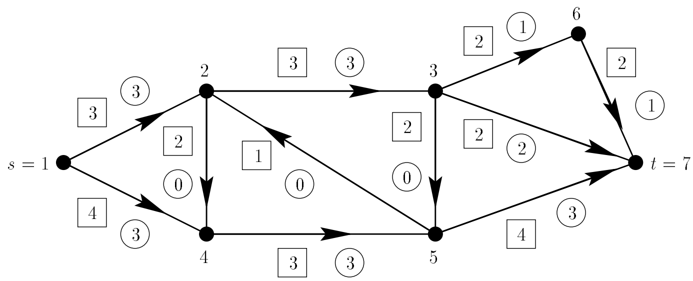
I



II (from $P = 1, 4, 5, 7$ and $\theta = 3$)

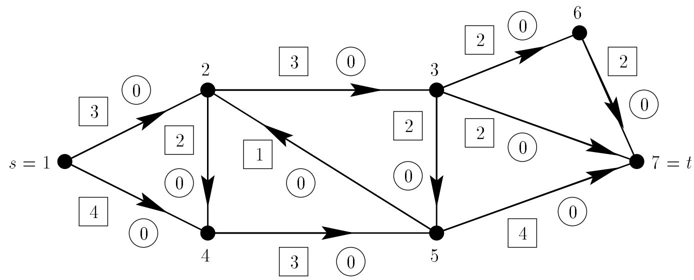


III (from $P = 1, 2, 3, 7$ and $\theta = 2$)

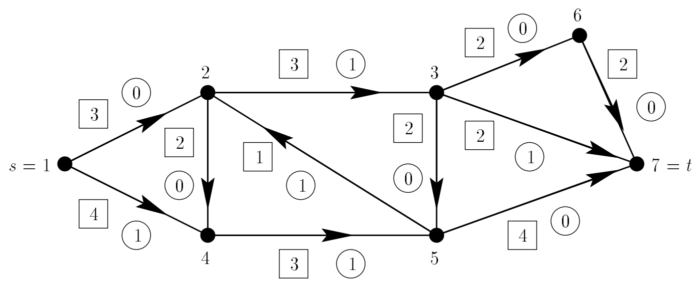


IV (from $P = 1, 2, 3, 6, 7$ and $\theta = 1$)

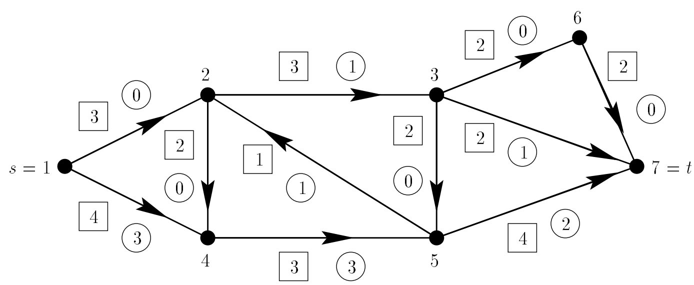
Figure 13.12: Applying Algorithm 13.4 to the directed network of Figure 13.10(a).



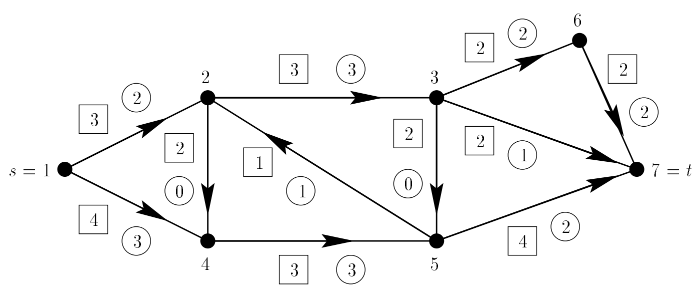
I



II (from $P = 1, 4, 5, 2, 3, 7$ and $\theta = 1$)



III (from $P = 1, 4, 5, 7$ and $\theta = 2$)



IV = (from P = 1, 2, 3, 6, 7 and T = 2)

Figure 13.13: Applying Algorithm 13.4 to the directed network of Figure 13.10(a) using a different choice of (s, t)-paths.

similar argument holds for the edge to t. [These arguments need to be expanded in case the flow-augmenting chain is not a simple chain. The expansion is left to the reader (Exercise 35).] However, we will only need simple flow-augmenting chains (Exercise 36). To illustrate with our example of Figure 13.10(b), note that $s_{14} = 2, s_{45} = 1, s_{36} = 1, s_{67} = 1$, and $x_{35} = 2$. Hence, the minimum of these numbers, λ, is 1. We increase x_{14}, x_{45}, x_{36}, and x_{67} by 1, and decrease x_{35} by 1, obtaining a flow of value 6, one more than the value of the flow shown.

We are now ready to state the main results about maximum flows, which will allow us to present our main algorithm. We have already shown that if a flow admits an augmenting chain, the value of the flow can be increased, so the flow is not maximum. The first result says that if the flow is not maximum, we can find an augmenting chain. (Thus, flow-augmenting chains are analogous to M-augmenting chains for matchings, and the next theorem is analogous to Theorem 12.7. We expand on the relation between network flows and matching in Section 13.3.8.)

Theorem 13.5 An (s,t)-flow is maximum if and only if it admits no augmenting chain from s to t.

Proof.[8] It remains to suppose that x is a flow with no augmenting chain and to show that x is a maximum. Let S be the set of vertices j such that there is an augmenting chain from s to j and let T be all other vertices. Note that s is in S because s alone defines an augmenting chain from s to s. Moreover, t is in T, since there is no augmenting chain. By definition of augmenting chain and by definition of S and T, we have for all i in S and all j in T, $x_{ij} = c_{ij}$ and $x_{ji} = 0$. For there is an augmenting chain from s to i, since i is in S. If $x_{ij} < c_{ij}$ or $x_{ji} > 0$, we can add edge $\{i,j\}$ to this chain to find an augmenting chain from s to j, contradicting j in T. Thus, for all i in S and j in T, $x_{ij} - x_{ji} = c_{ij}$.

By Corollary 13.3.1,

$$v = \sum_{i \in S} \sum_{j \in T} [x_{ij} - x_{ji}] = \sum_{i \in S} \sum_{j \in T} c_{ij} = c(S,T).$$

Hence, we have found a cut (S,T) with the same capacity as the value of the flow x. By Theorem 13.4, the flow is a maximum. Q.E.D.

A cut (S,T) is called *saturated* relative to a flow x if $x_{ij} = c_{ij}$ for all $i \in S, j \in T$, and $x_{ji} = 0$ for all $i \in S, j \in T$. The following is a corollary of the proof of Theorem 13.5.

Corollary 13.5.1 If (S,T) is a saturated (s,t)-cut relative to flow x, then x is a maximum (s,t)-flow.

The next result is a very famous theorem due to Elias, Feinstein, and Shannon [1956] and Ford and Fulkerson [1956].

[8] The proof may be omitted.

Theorem 13.6 (The Max-Flow Min-Cut Theorem) In a directed network, the maximum value of an (s,t)-flow equals the minimum capacity of an (s,t)-cut.

Proof.[9] It follows from Theorem 13.3 that the maximum value of an (s,t)-flow is at most the minimum capacity of an (s,t)-cut. To show equality, we suppose that x is a maximum flow, with value v. Then it can have no flow-augmenting chain, and so, by the proof of Theorem 13.5, we can find an (s,t)-cut (S,T) so that $v = c(S,T)$. It follows by Theorem 13.4 that x is a maximum flow and (S,T) is a minimum cut. Q.E.D.

Remark. Our proof of the Max-Flow Min-Cut Theorem uses the tacit assumption that there exists a maximum flow. This is easy to prove if all capacities are rational numbers. For then the Maximum-Flow Algorithm, described in Section 13.3.5, finds a maximum flow. If some capacities are not rational, a maximum flow still exists (even though the Maximum-Flow Algorithm, as we describe it, does not necessarily find a maximum flow). See Lawler [1976] or Papadimitriou and Steiglitz [1982] for a proof.

13.3.5 The Max-Flow Algorithm

We can now formulate the Max-Flow Algorithm.

Algorithm 13.5: The Max-Flow Algorithm

Input: A directed network with a source s and a sink t.
Output: A maximum (s,t)-flow x.

Step 1. Set $x_{ij} = 0$ for all i, j.

Step 2.

- ***Step 2.1.*** Find a flow-augmenting chain C from s to t. If none exists, go to Step 3.
- ***Step 2.2.*** Compute and record the slack of each forward arc of C and record the flow of each backward arc of C.
- ***Step 2.3.*** Compute λ, the minimum of the numbers recorded in Step 2.2.
- ***Step 2.4.*** Redefine x by adding λ to the flow on all forward arcs of C and subtracting λ from the flow on all backward arcs of C. Return to Step 2.1.

Step 3. Stop with the flow x.

Suppose that we apply this algorithm to the directed network of Figure 13.10(a). Since every (s,t)-path is an augmenting chain with no backward arcs, we can get to the fourth flow of Figure 13.13. We then identify the flow-augmenting chain

[9] The proof may be omitted.

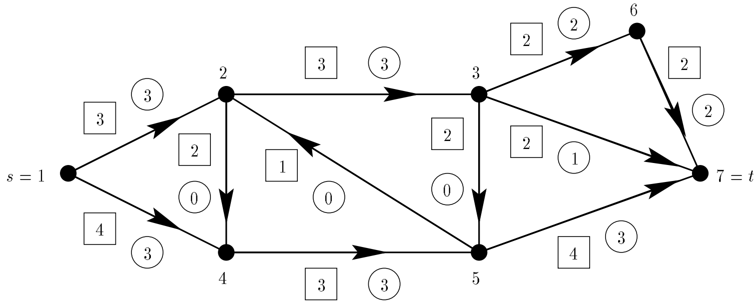


Figure 13.14: The flow obtained from the fourth flow of Figure 13.13 by using the flow-augmenting chain $C = 1, 2, 5, 7$.

$C = 1, 2, 5, 7$. In Step 2.2 we compute $s_{12} = 1, s_{57} = 2, x_{52} = 1$. Then the minimum, λ, of these numbers is $\lambda = 1$. We increase x_{12} and x_{57} by $\lambda = 1$ and decrease x_{52} by $\lambda = 1$, obtaining the flow of Figure 13.14. In this flow, there is no augmenting chain. We conclude that it is maximum. This agrees with our earlier conclusion, since the value of this flow is 6.

Theorem 13.7 If all capacities in the directed network are rational numbers, the Max-Flow Algorithm will attain a maximum flow.

Proof.[10] Clearly, if all capacities are integers, then at each iteration the number λ is an integer, and the value of the flow increases by λ. Since the value is at most the capacity of the cut defined by $S = \{s\}, T = V(D) - \{s\}$, the value cannot keep increasing by an integer amount more than a finite number of times. Hence, there will come a time when there is no augmenting chain from s to t. By Theorem 13.5, the corresponding flow will be maximum. The case of rational capacities can be handled by finding a common denominator δ for all the capacities and considering the directed network obtained from the original one by multiplying all capacities by δ. Q.E.D.

In general, if some capacities are not rational numbers, the algorithm does not necessarily converge in a finite number of steps.[11] Moreover, it can converge to a flow that is not a maximum, as Ford and Fulkerson [1962] showed. Edmonds and Karp [1972] have shown that if each flow augmentation is made along an augmenting chain with a minimum number of edges, the algorithm terminates in a finite number of steps and a maximum flow is attained. See Ahuja, Magnanti, and Orlin [1993], Lawler [1976], or Papadimitriou and Steiglitz [1982] for details. Ahuja, Magnanti, and Orlin [1989, 1991] give surveys of various improvements in network flow algorithms.

[10] The proof may be omitted.

[11] This point is of little practical significance, since computers work with rational numbers.

13.3.6 A Labeling Procedure for Finding Augmenting Chains[12]

The Max-Flow Algorithm as we have described it does not discuss how to find an augmenting chain in Step 2.1. In this subsection we describe a labeling procedure for finding such a chain. (The procedure is due to Ford and Fulkerson [1957].) In this procedure, at each step, a vertex is *scanned* and its neighbors are given *labels*. Vertex j gets label (i^+) or (i^-). The label is determined by finding an augmenting chain from s to j which ends with the edge $\{i, j\}$. The $+$ indicates that (i, j) is a forward arc in this augmenting chain, the $-$ that it is a backward arc. Eventually, if vertex t is labeled, we have an augmenting chain from s to t. If the procedure concludes without labeling vertex t, we will see that no augmenting chain exists, and we conclude that the flow is maximum. The procedure is described in detail as follows.

Algorithm 13.6: Subroutine: Labeling Algorithm for Finding an Augmenting Chain

Input: A directed network with a source s, a sink t, and an (s, t)-flow x.
Output: An augmenting chain or the message that x is a maximum flow.

Step 1. Give vertex s the label $(-)$.

Step 2. Let F be the set of arcs (i, j) such that $s_{ij} > 0$. Let B be the set of arcs (i, j) such that $x_{ij} > 0$. (Note that arcs in F can be used as forward arcs and arcs in B as backward arcs in an augmenting chain.)

Step 3. (*Labeling and Scanning*)

Step 3.1. If all labeled vertices have been scanned, go to Step 5.

Step 3.2. If not, find a labeled but unscanned vertex i and scan it as follows. For each arc (i, j), if $(i, j) \in F$ and j is unlabeled, give j the label (i^+). For each arc (j, i), if $(j, i) \in B$ and j is unlabeled, give j the label (i^-). Do not label any other neighbors of i. Vertex i has now been scanned.

Step 3.3. If vertex t has been labeled, go to Step 4. Otherwise, go to Step 3.1.

Step 4. Starting at vertex t, use the index labels to construct an augmenting chain. The label on vertex t indicates the next-to-last vertex in this chain, the label on that vertex indicates its predecessor in the chain, and so on. Stop and output this chain.

Step 5. Stop and output the message that the flow x is a maximum.

Let us illustrate this algorithm with the flow of Figure 13.10(b). We begin by labeling vertex s by $(-)$. Then we find that F consists of the arcs $(1, 4)$, $(2, 4)$, $(3, 6)$, $(3, 7)$, $(4, 5)$, $(5, 2)$, and $(6, 7)$, and B consists of all arcs except $(2, 4), (5, 2)$,

[12] This subsection may be omitted if time is short.

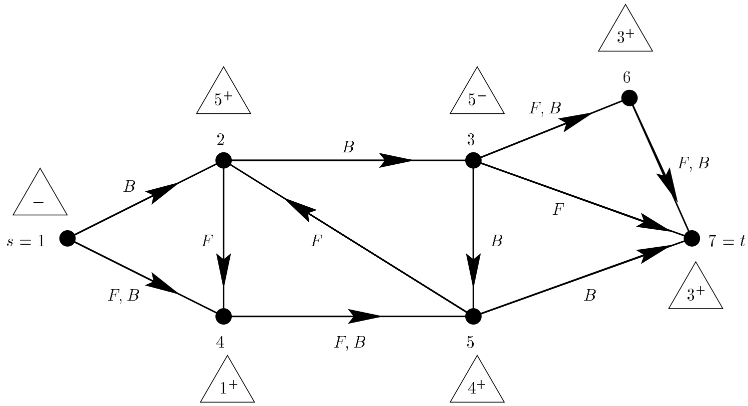


Figure 13.15: The labeling algorithm applied to the flow of Figure 13.10(b). The label is shown in a triangle next to a vertex. The arcs in F and in B are labeled F and B, respectively.

and $(3, 7)$. We now go through the labeling and scanning procedure (Step 3). Figure 13.15 shows the labels. Note that vertex $s = 1$ is labeled but unscanned. The only arc $(1, x)$ in F is $(1, 4)$. We therefore label vertex 4 with (1^+). There are no arcs $(x, 1)$ in B. Thus, vertex 1 has been scanned. Since vertex $7 = t$ has not yet been labeled, we find another labeled, but unscanned vertex, namely 4. We now consider the arcs $(4, x)$ in F, namely the arc $(4, 5)$. Thus, vertex 5 gets the label (4^+). We also note that no arc $(x, 4)$ is in B. Thus, vertex 4 has been scanned. Note that t has not yet been labeled, so we find another labeled but unscanned vertex, namely 5. The arc $(5, 2)$ is in F. Hence, we label vertex 2 with (5^+). The arc $(3, 5)$ is in B, so we label vertex 3 with the label (5^-). Then vertex 5 has been scanned. Now t has not yet been labeled. We find a labeled but unscanned vertex. We have a choice of vertices 2 and 3. Suppose that we pick 3. Scanning 3 leads to the label (3^+) on vertices 6 and 7. Now $t = 7$ has been labeled. (Coincidentally, all vertices have been labeled.) We go to Step 4 and read backward to find a flow-augmenting chain. In particular, the label (3^+) on vertex 7 sends us back to vertex 3. The label (5^-) here sends us back to vertex 5, the label (4^+) to vertex 4, and the label (1^+) to vertex 1. Thus, we have the flow-augmenting chain $1, 4, 5, 3, 7$. In this example we did not need to differentiate a $+$ label from a $-$ label in order to find a flow-augmenting chain. However, had there been two arcs between 3 and 5, $(3, 5)$ and $(5, 3)$, the label 5^- on vertex 3 would have told us to use the backward arc $(3, 5)$. The $+$ and $-$ labels will be useful in modifying the labeling algorithm to compute the number λ needed for the Max-Flow Algorithm.

Theorem 13.8 The labeling algorithm finds an augmenting chain if the flow x is not a maximum and ends by concluding that x is maximum otherwise.

Proof.[13] It is clear that if the algorithm produces a chain from s to t, the chain is

[13] The proof may be omitted.

augmenting. We show that if t has not yet been labeled and there is no labeled but unscanned vertex, then the flow is a maximum. Let S consist of all labeled vertices and T of all unlabeled vertices. Then (S, T) is an (s, t)-cut. Moreover, every arc from i in S to j in T is saturated and every arc from j in T to i in S has flow 0, for otherwise we could have labeled a vertex of T in scanning the vertices of S. We conclude that (S, T) is a saturated cut. By Corollary 13.5.1, we conclude that x is a maximum flow. Q.E.D.

In closing this subsection, we note that the labeling algorithm can be modified so that at the end, it is easy to compute the number λ needed for the Max-Flow Algorithm. At each step where we assign a label to a vertex j, we have just found an augmenting chain from s to j. We then let $\lambda(j)$ be the minimum of the numbers s_{uv} for (u, v) a forward arc of this chain and x_{uv} for (u, v) a backward arc of the chain. We let $\lambda(s)$ be $+\infty$. If we label j by scanning from i, we can compute $\lambda(j)$ from $\lambda(i)$. In particular, if j gets labeled (i^+), $\lambda(j) = \min\{\lambda(i), s_{ij}\}$, and if j gets labeled (i^-), then $\lambda(j) = \min\{\lambda(i), x_{ji}\}$. Finally, λ is $\lambda(t)$ (see Exercise 40). To illustrate, in our example, we would compute the following $\lambda(j)$ in the following order: $\lambda(s) = +\infty, \lambda(4) = \min\{\lambda(s), s_{14}\} = 2, \lambda(5) = \min\{\lambda(4), s_{45}\} = 1, \lambda(2) = \min\{\lambda(5), s_{52}\} = 1, \lambda(3) = \min\{\lambda(5), x_{35}\} = 1, \lambda(6) = \min\{\lambda(3), s_{36}\} = 1, \lambda(7) = \min\{\lambda(3), s_{37}\} = 1$, and conclude that $\lambda = \lambda(7) = 1$.

13.3.7 Complexity of the Max-Flow Algorithm

Let us make a comment on the computational complexity of the Max-Flow Algorithm. Let a be the number of arcs in the directed network and v be the maximum flow. The labeling algorithm described in Section 13.3.6 requires at most $2a$ arc inspections in each use to find an augmenting chain—we look at each arc at most once as a forward arc and at most once as a backward arc. If all capacities are integers and we start the Max-Flow Algorithm with the flow $x_{ij} = 0$ for all i, j, then each flow-augmenting chain increases the value by at least 1, so the number of iterations involving a search for an augmenting chain is at most v. Hence, the number of steps is at most $2av$.[14] One problem with this computation is that we want to determine complexity solely as a function of the size of the input, not in terms of the solution v. Note that our computation implies that the algorithm might take a long time. Indeed, it can. Consider the directed network of Figure 13.16. Starting with the 0 flow, we could conceivably choose first the augmenting chain $1, 2, 3, 4$, then the augmenting chain $1, 3, 2, 4$, then $1, 2, 3, 4$, then $1, 3, 2, 4$, and so on. This would require 2 billion iterations before converging! However, if we happen to choose first the augmenting chain $1, 2, 4$, and then the augmenting chain $1, 3, 4$, then

[14] This disregards the simple computations of computing slacks and modifying the flow. These are each applied to each arc at most twice in each iteration, so add a constant times av to the number of steps. In each iteration, we also have to compute λ, which is the minimum of a set of no more than a numbers. The total number of computations in computing all the λ's is thus again at most av. Finally, we have to construct the augmenting chain from the labels, which again takes at most a steps in each iteration, or at most av steps in all. In sum, if k is a constant, the total number of steps is thus at most kav, which is $O(av)$, to use the notation of Section 2.18.

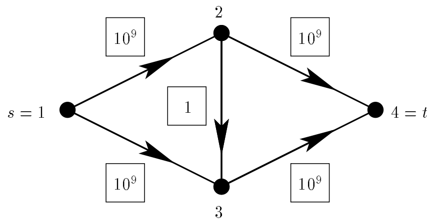


Figure 13.16: A poor choice of augmenting chains leads to 2 billion iterations in the Max-Flow Algorithm.

we finish in two iterations! To avoid these sorts of problems, we change Step 3.2 of the labeling algorithm so that vertices are scanned in the same order in which they receive labels. Edmonds and Karp [1972] have shown that in this case, a maximum flow is obtained in at most $an/2$ applications of the labeling algorithm, where n is the number of vertices. Thus, since each use of the labeling algorithm requires at most $2a$ steps, the total number of steps[15] is at most

$$(2a)(an/2) = a^2 n \leq [n(n-1)]^2 n.$$

The Edmonds and Karp algorithm has since been improved. See Ahuja, Magnanti, and Orlin [1993] and Papadimitriou and Steiglitz [1982] for references on improved algorithms.

13.3.8 Matching Revisited[16]

In this subsection we investigate the relation between network flows and the matchings we studied in Chapter 12. In particular, we show how to prove Theorem 12.5, the result that in a bipartite graph, the number of edges in a maximum matching equals the number of vertices in a minimum covering.

Suppose that $G = (X, Y, E)$ is a bipartite graph. We can make it into a directed network D, called the *associated network*, by adding a source s, a sink t, arcs from s to all vertices of X and from all vertices of Y to t and by directing all edges between X and Y from X to Y. We put capacity 1 on all new arcs, and capacity ∞ (or a very large number) on all arcs from X to Y. Then it is easy to see that any integer flow (a flow all of whose x_{ij} values are integers) from s to t in D corresponds to a matching M in G: We include an edge $\{i, j\}$ in M if and only if there is positive flow along the arc (i, j), $i \in X$, $j \in Y$. Conversely, any matching M in G defines an integer flow x in D: For $\{i, j\}$ in E with i in X, take x_{ij} to be 1 if $\{i, j\}$ is in M, and 0 otherwise; take x_{sk} to be 1 if and only if k is saturated in M, and x_{lt} to be 1 if and only if l is saturated in M. Moreover, the number of edges in the matching M equals the value of the flow x. Figure 13.17 illustrates this construction by showing a bipartite graph G and the associated network D. The flow shown in D corresponds to the matching shown by wiggly edges in G. We summarize these results in the next theorem.

[15] As per the previous footnote (14), a more accurate estimate is at most $ka(an/2)$ steps.
[16] This subsection may be omitted.

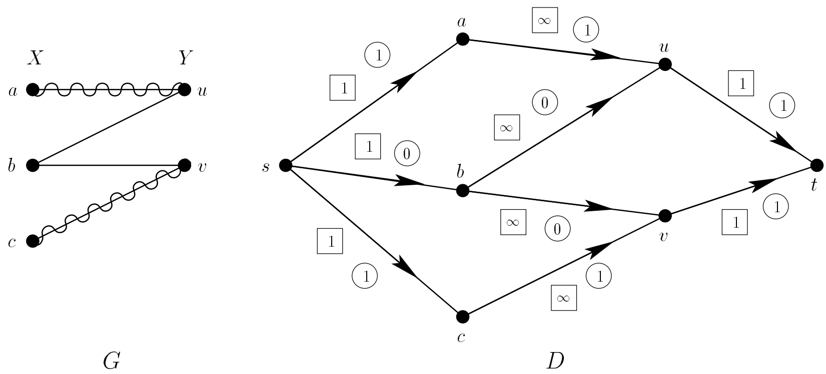


Figure 13.17: A bipartite graph G and its associated network D. The flow in D shown by encircled numbers corresponds to the matching in G shown by wiggly edges. Capacities are shown in squares.

Theorem 13.9 Suppose that $G = (X, Y, E)$ is a bipartite graph and D is the associated network. Then there is a one-to-one correspondence between integer (s,t)-flows in D and matchings in G. Moreover, the value of an integer flow is the same as the number of edges in the corresponding matching.

Since matchings in a bipartite graph $G = (X, Y, E)$ correspond to flows in the associated network D, we might ask what coverings in the graph correspond to in the network. The answer is that coverings correspond to (s,t)-cuts of finite capacity. To make this precise, suppose that $A \subseteqq X$ and $B \subseteqq Y$. Any covering K in G can be written in the form $A \cup B$ for such A and B. Any (s,t)-cut (S,T) of D can be written in the following form:

$$S = \{s\} \cup (X - A) \cup B, \quad T = \{t\} \cup A \cup (Y - B);$$

for s is in S and t is in T. We simply define A to be all vertices in $T \cap X$ and B to be all vertices in $S \cap Y$ (see Figure 13.18). We now have the following theorem.

Theorem 13.10 Suppose that $G = (X, Y, E)$ is a bipartite graph and D is the associated network. Suppose that $A \subseteqq X$ and $B \subseteqq Y$. Let $K = A \cup B$ and let

$$S = \{s\} \cup (X - A) \cup B, \quad T = \{t\} \cup A \cup (Y - B). \tag{13.12}$$

Then K is a covering of G if and only if (S,T) is an (s,t)-cut of D of finite capacity. Moreover, if (S,T) is an (s,t)-cut of finite capacity, $|A \cup B|$ is the capacity of the cut (S,T).

Proof. Suppose that (S,T) defined by (13.12) is an (s,t)-cut of finite capacity. Since the cut has finite capacity, there can be no arcs from X to Y in the cut, that is, no arcs from $X - A$ to $Y - B$ (see Figure 13.18). Thus, by (13.12), all arcs from X to Y in D go from A or to B, so all edges of G are joined to A or to B, so

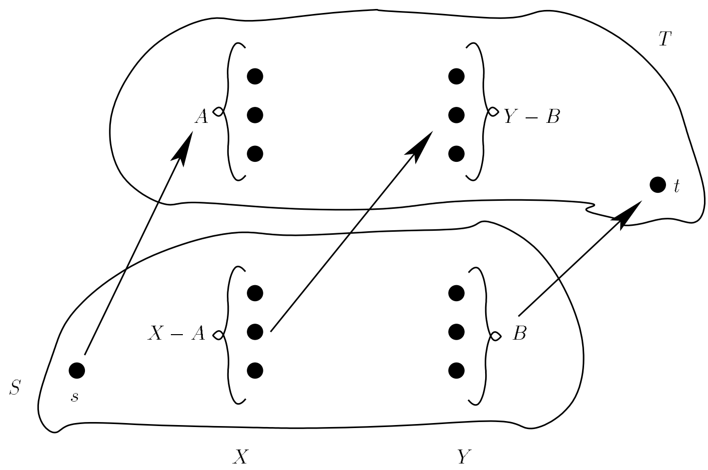


Figure 13.18: The cut (13.12). The only possible arcs in the cut are shown by arrows.

$K = A \cup B$ is a covering. Next, note that $|A \cup B|$ is the capacity of the cut (S, T). For by Figure 13.18, the arcs in the cut are exactly the arcs (s, x) for x in A and (y, t) for y in B. There are $|A \cup B|$ such arcs. Each has unit capacity, so the cut has capacity $|A \cup B|$.

It remains to prove that if $K = A \cup B$ is a covering of G, then (S, T) defined by (13.12) gives an (s, t)-cut and it has finite capacity. This is left to the reader (Exercise 37). Q.E.D.

We can now prove Theorem 12.5.

Proof of Theorem 12.5. [Theorem 12.5: Suppose that $G = (X, Y, E)$ is a bipartite graph. Then the number of edges in a maximum matching equals the number of vertices in a minimum covering.]

Consider the associated network. Since all the capacities are integers, it follows from the proof of Theorem 13.7 that there is a maximum flow in which all x_{ij} are integers. The value of this flow will also have to be an integer, and will, by Theorem 13.9, give the number α of edges in the maximum matching. Moreover, by the Max-Flow Min-Cut Theorem (Theorem 13.6), this number α is also the minimum capacity of an (s, t)-cut (S, T). Suppose that this cut is given by (13.12). Now clearly, an (s, t)-cut of minimum capacity has finite capacity, so by Theorem 13.10, (S, T) corresponds via (13.12) to a covering $K = A \cup B$. The number of vertices in K is equal to $|A \cup B|$, which is equal to the capacity of the cut (S, T), which is equal to α. Then we have a covering of α vertices and a matching of α edges, so by Theorem 12.4, the covering must be a minimum. Hence, the number of edges in a maximum matching equals the number of vertices in a minimum covering. Q.E.D.

13.3.9 Menger's Theorems[17]

The Max-Flow Min-Cut Theorem has useful application in the design of communication networks. In designing such networks, we like to make sure that their "connectivity" remains intact after some connections are destroyed and we are therefore interested in the minimum number of arcs whose removal destroys all paths from a vertex a to a vertex z. One way to preserve connectivity is to make sure that we build in "redundancy" and have a lot of paths between pairs of vertices. Let us say that a pair of paths from a to z is *arc-disjoint* if they do not have any arcs in common. We shall see that there is a close relationship between the minimum number of arcs whose removal destroys all (simple) paths from a to z and the maximum number of arc-disjoint simple paths from a to z. The result is one of a series of remarkable theorems known as Menger's Theorems (after Menger [1927]).

Theorem 13.11 Let N be a connected directed network with source a and sink z in which each arc has capacity 1. Then:

(a) The value of a maximum (a, z)-flow in N is equal to the maximum number M of arc-disjoint simple paths from a to z.

(b) The capacity of a minimum (a, z)-cut in N is equal to the minimum number p of arcs whose removal destroys all simple paths from a to z.

Proof. We give the proof in the case where a has only outgoing arcs and z only incoming arcs. The proof in the general case is left to the reader (Exercise 32).

(a) Let F be the set of all integers k such that if there is an (a, z)-flow of value k in a directed network with capacities all 1, there is a set of k arc-disjoint simple paths from a to z. Clearly, $k = 1$ is in F. We now argue by induction on k that all integers are in F. We assume that k is in F and show that $k + 1$ is in F. Suppose that there is a flow of value $k + 1$. Then there is a simple path P from a to z each arc of which has nonzero flow. Since the capacities are all 1, the flows are 1 on each arc of P. The remaining k units of flow must go through arcs not on P. Thus, deleting the arcs of P leaves a directed network with all capacities 1 and with a flow of value k. By the inductive hypothesis, there is a set of k arc-disjoint simple paths from a to z in this new directed network. These paths together with P form a set of $k+1$ arc-disjoint simple paths from a to z in the original directed network. This shows that if v is the value of a maximum (a, z)-flow, then $v \leq M$.

Suppose that we have M arc-disjoint simple paths from a to z in N. If we define x_{ij} to be 1 on all arcs (i, j) that are in such a path and 0 otherwise, then clearly $x = \{x_{ij}\}$ is a flow with value M and so $v \geq M$.

(b) Let (S, T) be an (a, z)-cut in N. Then if all arcs from S to T are destroyed, z is not reachable from a. It follows that the capacity of this cut is at least p.

Now let A be a set of p arcs whose deletion destroys all simple paths from a to z, and let N' be the directed network obtained from N by removing arcs of A. Let B be the set of all vertices reachable from a in N'. By definition of B, there can

[17] This subsection may be omitted.

be no arc from a vertex of B to a vertex of $B' = V(N) - B$, so all arcs from B to B' in N are in A. It follows that (B, B') is a cut of capacity at most p and thus it must be a minimum cut. Q.E.D.

Theorem 13.12 (Menger [1927]) The maximum number of arc-disjoint simple paths from a vertex a to a vertex z in a digraph D is equal to the minimum number of arcs whose deletion destroys all simple paths from a to z in D.

Proof. Build a directed network with source a and sink z by assigning unit capacity to each arc of D. The result follows from Theorem 13.11 and the Max-Flow Min-Cut Theorem (Theorem 13.6). Q.E.D.

To illustrate this theorem, consider the directed network of Figure 13.10 (disregarding capacities). Then there are two arc-disjoint simple paths from s to t: namely, $s, 2, 3, 6, t$ and $s, 4, 5, t$. There are also two arcs whose deletion destroys all (simple) paths from s to t: arcs $(2, 3)$ and $(4, 5)$. Thus, it follows that the maximum number of arc-disjoint simple paths from s to t is 2 and the minimum number of arcs whose deletion destroys all (simple) paths from s to t is also 2.

We say that two paths from a to z in a digraph are *vertex-disjoint* if they do not have a vertex in common except a and z.

Theorem 13.13 (Menger [1927]) Suppose that there is no arc from vertex a to vertex z in digraph D. Then the maximum number of vertex-disjoint simple paths from a to z in D is equal to the minimum number of vertices whose removal from D destroys all simple paths from a to z.

Proof. Define a new digraph D' by splitting each vertex u other than a and z into two new vertices u_1 and u_2, adding an arc (u_1, u_2), replacing each arc (u, w) with arc (u_2, w_1), replacing each arc (u, z) with (u_2, z), and replacing each arc (a, u) with (a, u_1). Then one can show that the maximum number of arc-disjoint simple paths from a to z in D' is equal to the maximum number of vertex-disjoint simple paths from a to z in D and the minimum number of arcs in D' whose deletion destroys all simple paths from a to z is equal to the minimum number of vertices whose removal destroys all simple paths from a to z in D. Details are left to the exercises [Exercises 33(a) and (b)]. Q.E.D.

To illustrate this theorem, consider again the directed network of Figure 13.10. The two arc-disjoint simple paths from s to t given above are also vertex-disjoint. Two vertices whose removal destroys all simple paths from s to t are vertices 2 and 4.

EXERCISES FOR SECTION 13.3

1. In each directed network of Figure 13.19, a (potential) (s, t)-flow is shown in the circled numbers.
   (a) Which of these flows is feasible? Why?
   (b) If the flow is feasible, find its value.

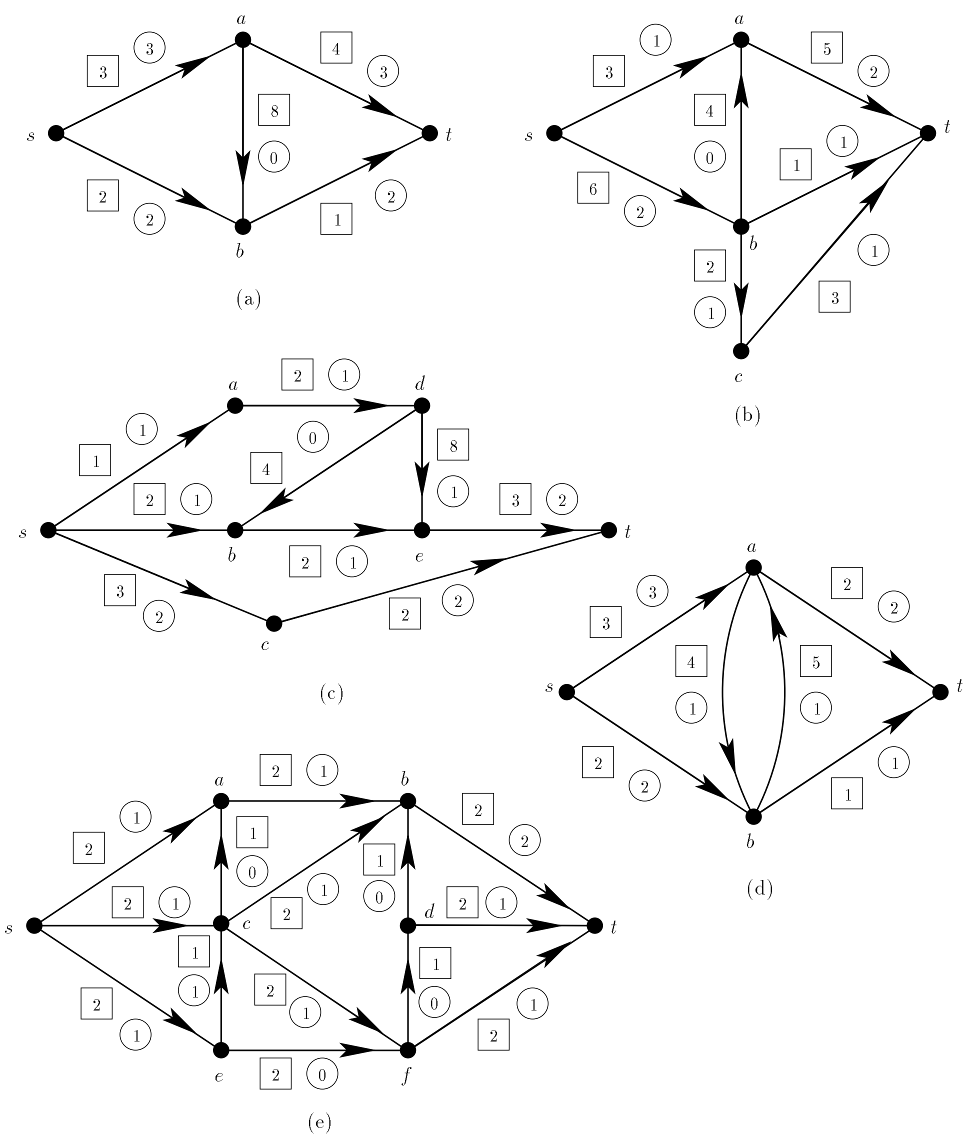


Figure 13.19: (Potential) flows for exercises of Section 13.3.

2. In each directed network D of Figure 13.8, interpret the weights as capacities and find the capacity of the cut (S, T) where $S = \{a, b, e\}$ and $T = V(D) - S$.
3. In each flow x of Figure 13.20, compute the slack on each arc.
4. In the flow III of Figure 13.13, which of the following are flow-augmenting chains?

   (a) $1, 2, 3, 7$ (b) $1, 2, 5, 7$ (c) $1, 4, 5, 7$ (d) $1, 4, 2, 3, 7$

5. In the flow III of Figure 13.12, which of the chains of Exercise 4 are flow-augmenting chains?
6. For each flow of Figure 13.20, either show that it is maximum by finding an (s, t)-cut (S, T) such that $v = c(S, T)$ or show that it is not maximum by finding an augmenting chain.
7. Give an example of a directed network and a flow x for this network which has value 0 but such that x_{ij} is not 0 for all i, j.
8. Illustrate Menger's (first) Theorem, Theorem 13.12, on the directed networks of:

   (a) Figure 13.19 (b) Figure 13.21

9. Illustrate Menger's (second) Theorem, Theorem 13.13, on the directed networks of:

   (a) Figure 13.19 (b) Figure 13.21

10. Let G be an undirected graph.

    (a) State and prove a variant of Menger's (first) Theorem, Theorem 13.12.

    (b) State and prove a variant of Menger's (second) Theorem, Theorem 13.13.

11. Apply Algorithm 13.4 to each directed network of Figure 13.8 if weights are interpreted as capacities and $s = a$, $t = z$.
12. Repeat Exercise 11 using Algorithm 13.5.
13. For each flow of Figure 13.20, apply Algorithm 13.6 to search for an augmenting chain.
14. In each application of Algorithm 13.6 in Exercise 13, also calculate the numbers $\lambda(j)$.
15. In directed network (d) of Figure 13.8, let $s = a$ and $t = z$ and define a flow by letting $x_{ab} = x_{bc} = x_{cd} = x_{dz} = 2$ and $x_{ae} = x_{ef} = x_{fg} = x_{gz} = 3$, and $x_{ah} = x_{hi} = x_{ij} = x_{jz} = 2$, and otherwise taking $x_{ij} = 0$. Apply Algorithm 13.6 to search for a flow-augmenting chain.
16. Let G be a graph and S and T be disjoint subsets of the vertices. Show that the maximum number of vertex-disjoint simple paths with one end in S and one end in T is equal to the minimum number of vertices whose deletion separates S from T in the sense that after deletion, no connected component contains both a vertex of S and a vertex of T.
17. For each bipartite graph G of Figure 12.12:

    (a) Find the associated network D.

    (b) Find an integer (s, t)-flow in D and its corresponding matching in G.

    (c) Find an (s, t)-cut in D and its corresponding covering in G.

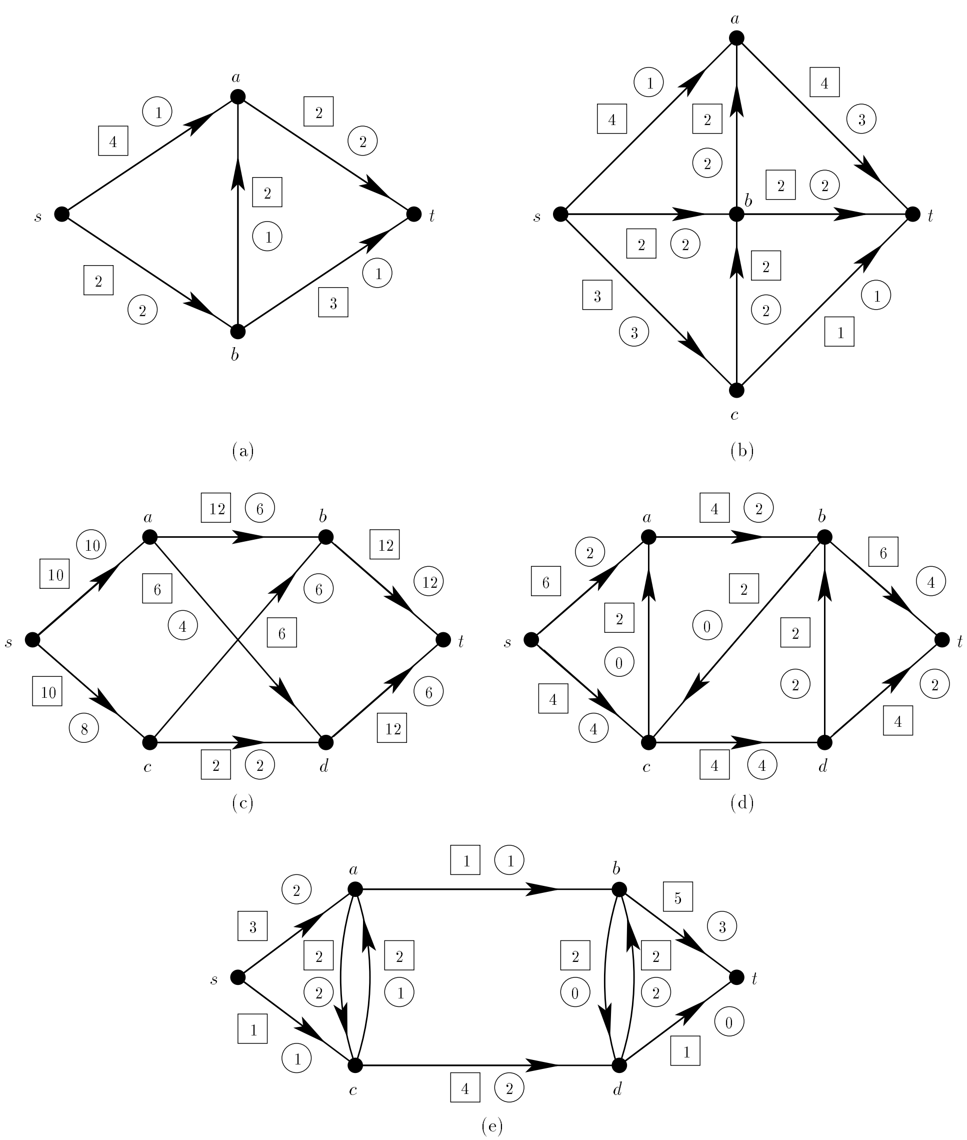


Figure 13.20: Flows for exercises of Sections 13.3 and 13.4.

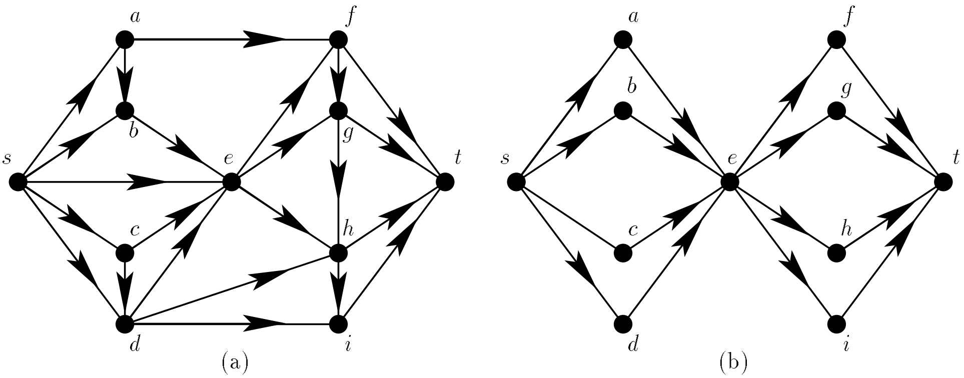


Figure 13.21: Digraphs for exercises of Section 13.3.

18. Consider the college admissions problem of Example 13.8. The following chart contains the list of 8 applicants, including their potential major and their home state.

Applicant	Potential Major				Home State
	Chem.	Bio.	Math.	Geo.	
A	x				MD
B		x			MD
C			x	x	NY
D	x	x			MD
E			x	x	PA
F			x		NJ
G			x		NY
H				x	PA

   (a) Draw the associated directed network.

   (b) If the college wants at most 2 students from each state, can the admissions staff do their job?

   (c) If the college wants at most 1 student from each state, can the admissions staff do their job?

19. Consider the distributed computing on a two-processor computer problem of Example 13.9. Table 13.4 gives the execution costs for each module on the two processors and Table 13.5 gives the interprocessor communication costs.

   (a) Draw the associated network.

   (b) Find the minimum-cost assignment of modules to processors using a minimum-cost cut.

20. Recall that a flow from a source s to a sink t in an undirected network is defined as a flow from s to t in the directed network obtained by replacing each edge $\{i, j\}$ by the two arcs (i, j) and (j, i) and letting each arc have the same capacity as the corresponding edge. Find a maximum flow from s to t in each network of Figure 13.22.

Table 13.4: Execution Costs

i	1	2	3	4	5
a_i	6	5	10	4	8
b_i	3	6	5	10	4

Table 13.5: Interprocessor Communication Costs

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 4 & 1 & 0 & 0 \\ 4 & 0 & 5 & 0 & 0 \\ 1 & 5 & 0 & 6 & 2 \\ 0 & 0 & 6 & 0 & 1 \\ 0 & 0 & 2 & 1 & 0 \end{pmatrix} \end{array}$$

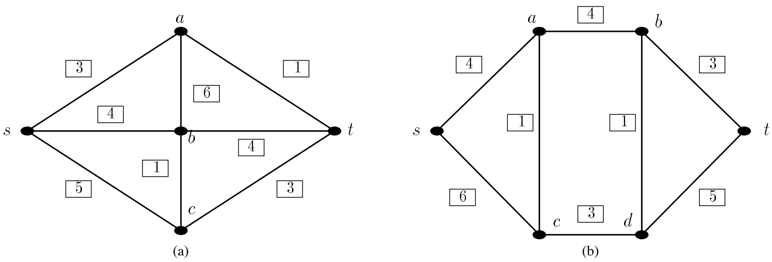


Figure 13.22: Networks for exercises of Section 13.3.

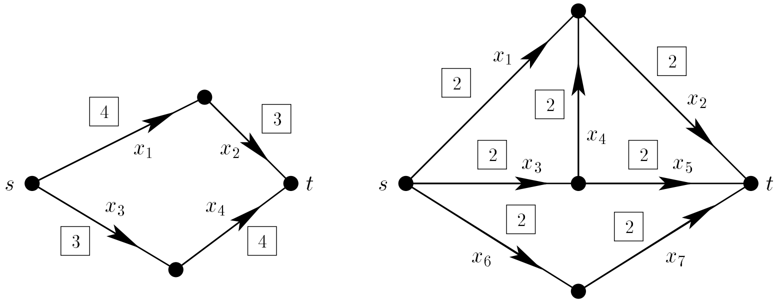
21. A pipeline network sends oil from location A to location B. The oil can go via the northern route or the southern route. Each route has one junction, with a pipeline going from the junction on the southern route to the junction on the northern route. The first leg of the northern route, from location A to the junction, has a capacity of 400 barrels an hour; the second leg, from the junction to location B, has a capacity of 300 barrels an hour. The first leg of the southern route has a 500-barrel/hour capacity, and the second leg has a 300-barrel hour capacity. The pipeline joining the two junctions also has a 300-barrel/hour capacity. What is the largest number of barrels of oil that can be shipped from location A to location B in an hour?

22. In this exercise we build on the notion of reliability of systems discussed in Example 2.21 and in Example 3.10. Suppose that a system is represented by a directed network, with the components corresponding to arcs. Let us say that the system works if and only if in the modified network defined by working components, there is an (s,t)-flow of value at least v. For each directed network of Figure 13.23, compute $F(x_1x_2\cdots x_n)$ as defined in Example 2.21 if v is 3.

23. Suppose that D is a directed network and X is a set of sources and Y is a set of sinks. [We assume that $X, Y \subsetneqq V(D)$ and $X \cap Y = \emptyset$.] An (X,Y)-*flow* is a flow where the conservation conditions (13.6) hold only for vertices that are not sources or sinks. The *value* of the flow is defined to be

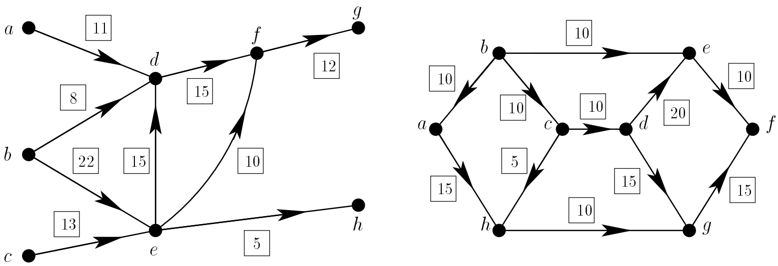
$$\sum_{\substack{i\in X\\ j\notin X}} x_{ij} - \sum_{\substack{i\in X\\ j\notin X}} x_{ji}.$$

We can find a maximum (X,Y)-flow by joining two new vertices, s and t, to D,



(a) (b)

Figure 13.23: Directed networks for exercises of Section 13.3.



$X = \{a, b, c\}$, $Y = \{g, h\}$ $X = \{a, b\}$, $Y = \{f, g, h\}$

(a) (b)

Figure 13.24: Directed networks with a set X of sources and a set Y of sinks.

adding arcs of capacity ∞ from s to all vertices in X and from all vertices in Y to t, and finding a maximum (s, t)-flow in the new network. Find a maximum (X, Y)-flow in each directed network of Figure 13.24.

24. There are three warehouses, w_1, w_2, and w_3, and three retail outlets, r_1, r_2, and r_3. The warehouses have, respectively, 3000, 4000, and 6000 drums of paint, and the retail outlets have demand for, respectively, 2000, 4000, and 3000 drums. Figure 13.25 shows a freight network, with the capacity on arc (i, j) giving the largest number of drums that can be shipped from location i to location j during a given day. Can all the demands be met if only one day is allowed for shipping from warehouses? (A drum can go along as many arcs as necessary in one day.) If not, what is the largest total demand that can be met? Solve this problem by translating it into a multisource, multisink problem and then into an ordinary network flow problem (see Exercise 23). This is an example of a *transshipment problem.*

25. (Ahuja, Magnanti, and Orlin [1993]) A union has a certain number of skilled craftsmen with varied skills and varying levels of seniority. Each person has at least one of a set of designated skills but only one level of seniority. The union wants to organize a governing board subject to certain conditions: One person with each type of skill is a member of the governing board, and the number of people with seniority level k

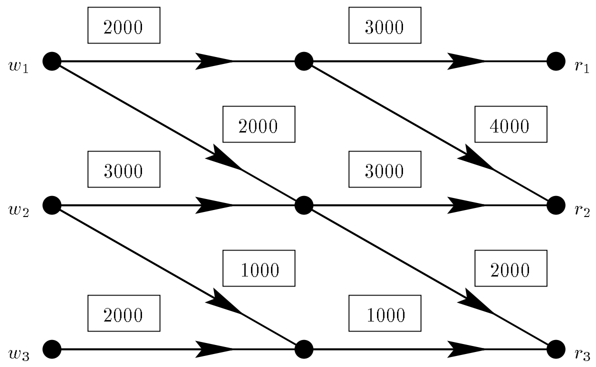


Figure 13.25: Freight network for Exercise 24, Section 13.3.

is at most u_k. Is it possible to find such a governing board? Show that the answer is "yes" if and only if a certain (s, t) directed network has a maximum-flow value equal to the number of designated skills.

26. Several legislative committees with no members in common are gathering to exchange ideas. We would like to assign them to different rooms so that no two people on the same committee go to the same room. Show how to assign committee members to rooms using a maximum-flow formulation. You should use the number of people in each committee and the capacity of the ith meeting room as input to your formulation.

27. In some directed networks, we might have vertex capacities as well as arc capacities. For instance, there might be limited capacity for the number of cars that can pass through a given toll facility. Then we seek a maximum flow satisfying both arc and vertex capacity constraints. Transform this problem into a standard maximum-flow problem with only arc capacity constraints.

28. Show that if x is an (s, t)-flow of value v and (S, T) is an (s, t)-cut of capacity c, then $v = c$ if and only if for each arc (i, j) from T to S, $x_{ij} = 0$, and for each arc (i, j) from S to T, $x_{ij} = c_{ij}$.

29. Suppose that a flow from s to t is decomposed into unit flow paths from s to t, and each of these unit flow paths crosses a given saturated cut exactly once. Show that the flow is maximum.

30. We wish to send messengers from a location s to a location t in a region whose road network is modeled by a digraph. Because some roads may be blocked, we wish each messenger to drive along a route that is totally disjoint from that of all other messengers. How would we find the largest number of messengers who could be sent? (*Hint:* Use unit capacities.)

31. In Example 13.8, prove:

    (a) If a maximum-flow value equals the number of majors represented among the applicants' interests, it solves the "college admissions" problem.

    (b) If not, there is no solution.

32. Complete the proof of Theorem 13.11 in the case that there may be incoming arcs to vertex a or outgoing arcs from vertex z.

33. This exercise establishes the proof of Theorem 13.13. Suppose that there is no arc from vertex a to vertex z in digraph D. Define D' as in the "proof" of the theorem.

    (a) Show that the maximum number of arc-disjoint simple paths from a to z in D' is equal to the maximum number of vertex-disjoint simple paths from a to z in D.

    (b) Show that the minimum number of arcs in D' whose deletion destroys all (simple) paths from a to z is equal to the minimum number of vertices whose removal destroys all simple paths from a to z in D.

34. Let N be an undirected network whose underlying graph is connected. Recall that a *cut* in N is a partition of the vertices into two sets S and T or, equivalently, the set of all edges joining a vertex of S to a vertex of T. If s and t are the source and sink of N, respectively, (S, T) is an *(s,t)-cut* if $s \in S$ and $t \in T$. Let F be a set of edges of N.

    (a) Show that if F is a simple cut set in the sense of Exercise 18, Section 13.1, it is a cut.

    (b) Show that if F is an (s,t)-cut with a minimum capacity, F is a simple cut set.

35. Suppose that C is a flow-augmenting chain and λ is the capacity of C as defined in Section 13.3.4. If we increase each x_{ij} on a forward arc by λ and decrease each x_{ij} on a backward arc by λ, show that even if C is not a simple chain, the value or net flow out of s is increased.

36. Show that in the Max-Flow Algorithm, it suffices to find flow-augmenting chains that are simple chains.

37. Complete the proof of Theorem 13.10.

38. Use the results of Section 13.3.8 to prove the König-Egerváry Theorem (Corollary 12.5.1) without first proving Theorem 12.5.

39. Suppose that $G = (X, Y, E)$ is a bipartite graph and M is a matching of G. Use the results of Section 13.3.8 to prove Theorem 12.7, namely, that M is maximum if and only if G contains no M-augmenting chain.

40. Show that in the labeling algorithm, if $\lambda(j)$ is defined as in the discussion following the proof of Theorem 13.8, then $\lambda = \lambda(t)$.

13.4 MINIMUM-COST FLOW PROBLEMS

13.4.1 Some Examples

An alternative network flow problem, which has many important special cases, is the following. Suppose that in a directed network, in addition to a nonnegative capacity c_{ij} on each arc, we have a nonnegative cost a_{ij} of shipping one unit of flow from i to j. We want to find a flow sending a fixed nonnegative number v of units from source s to sink t, and doing so at minimum cost. That is, we want to find

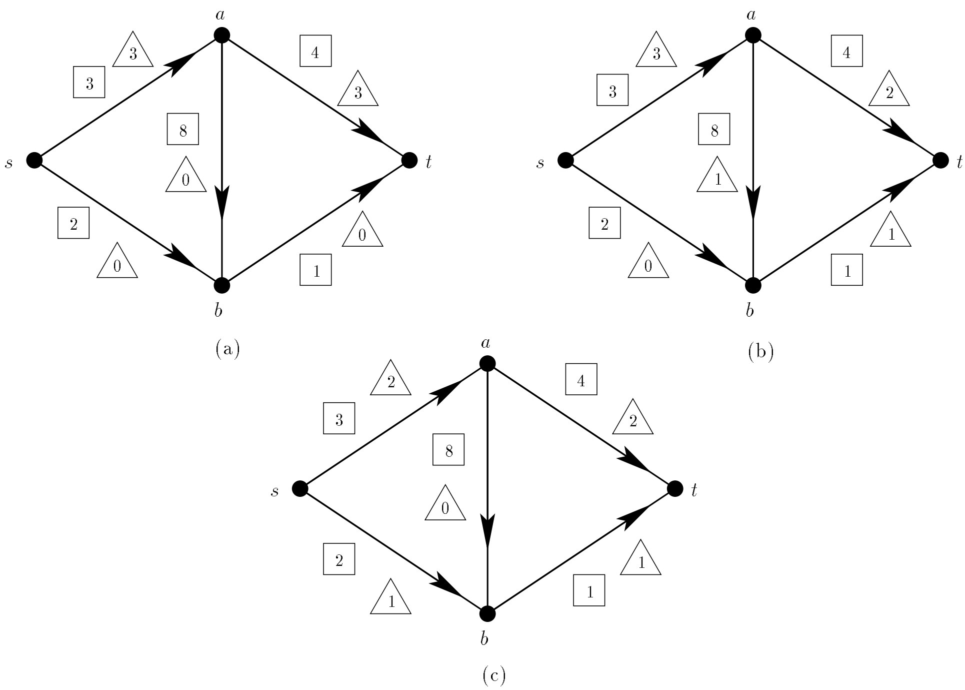


Figure 13.26: The three flows of value 3 for the directed network (a) of Figure 13.19. The capacity is shown in a square, the flow in a triangle.

a flow $x = \{x_{ij}\}$ such that its value is v and such that $\sum a_{ij}x_{ij}$ is minimized. We call this a *minimum-cost flow problem.*

To illustrate, consider the directed network (a) of Figure 13.19. Consider the number in the square as the capacity and the number in the circle as the cost. There are three flows that attain a value of 3; these are shown in Figure 13.26. Of these flows, flow (c) has the minimum cost: namely,

$$2 \cdot 3 + 2 \cdot 3 + 0 \cdot 0 + 1 \cdot 2 + 1 \cdot 2 = 16.$$

Flows (a) and (b) have values 18 and 17, respectively.

Example 13.10 Routing a Traveling Party (Minieka [1978]) A traveling party of 75 people is to go from New York to Honolulu. What is the least-cost way to route the party? The solution is obtained by finding a minimum-cost flow of 75 from New York to Honolulu in a directed network where the vertices are cities, an arc indicates a direct air link, and there is a capacity constraint (unbooked seats) and a cost constraint (air fare) on each arc. (This assumes that air fares are simple sums along links, which usually would not be true.) ■

Example 13.11 The Transportation Problem Imagine that a particular commodity is stored in n warehouses and is to be shipped to m markets. Let a_i be the supply of the commodity at the ith warehouse, let b_j be the demand for the commodity at the jth market, and let a_{ij} be the cost of transporting one unit of the commodity from warehouse i to market j. For simplicity, we assume that $\sum a_i = \sum b_j$, that is, that the total supply equals the total demand. (This assumption can easily be eliminated—see Exercise 10.) The problem is to find a shipping pattern that minimizes the total transportation cost.

This problem can be formulated as follows. Let x_{ij} be the number of units of the commodity shipped from i to j. We seek to minimize

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

subject to the following constraints: For every i,

$$\sum_{j=1}^{m} x_{ij} \leq a_i, \tag{13.13}$$

and for every j,

$$\sum_{i=1}^{n} x_{ij} \geq b_j. \tag{13.14}$$

Constraint (13.13) says that the total amount of commodity shipped from the ith warehouse is at most the amount there, and constraint (13.14) says that the total amount of commodity shipped to the jth market is at least the amount demanded. Note that since $\sum a_i = \sum b_j$, any solution satisfying (13.13) and (13.14) for all i and j will also satisfy

$$\sum_{j=1}^{m} x_{ij} = a_i \tag{13.15}$$

and

$$\sum_{i=1}^{n} x_{ij} = b_j. \tag{13.16}$$

We can look at this transportation problem as a minimum-cost flow problem. Draw a digraph with vertices the n warehouses $w_1, w_2, \ldots, w_n$ and the m markets $k_1, k_2, \ldots, k_m$. Add a source vertex s and a sink vertex t, and include arcs from each warehouse to each market, from the source to all warehouses, and from each market to the sink (see Figure 13.27). On arc (w_i, k_j), place a cost a_{ij} and a capacity $c_{ij} = \infty$ (or a very large number); on arc (s, w_i), place a cost of 0 and a capacity of a_i; and on arc (k_j, t), place a cost of 0 and a capacity of b_j. Because we have constraints (13.13) and (13.14), it is easy to see that we have a minimum-cost flow problem for this directed network: We seek a flow of value equal to $\sum a_i = \sum b_j$, at minimum cost. ∎

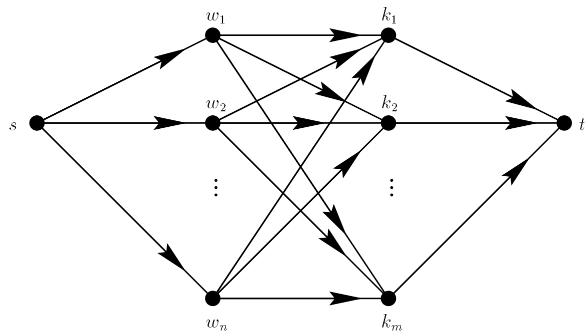


Figure 13.27: A directed network for the transportation problem.

Example 13.12 The Optimal Assignment Problem (Example 12.6 Revisited) In Example 12.6 and Section 12.7.2 we discussed the following job assignment problem. There are n workers and m jobs, every worker is suited for every job, and worker i's potential performance on job j is given a rating r_{ij}. We wish to assign workers to jobs so as to maximize the sum of the performance ratings. Let x_{ij} be a variable that is 1 if worker i is assigned to job j and 0 otherwise. Then we want to maximize

$$\sum_{\substack{i=1\\j=1}}^{\substack{m\\n}} r_{ij}x_{ij}$$

under the constraints

$$\sum_{i=1}^{n} x_{ij} \leq 1 \quad \text{and} \quad \sum_{j=1}^{m} x_{ij} \geq 1. \tag{13.17}$$

(Note that in Section 12.7.2, we took $m = n$. Here we consider the general case.) We may think of this as a transportation problem (Example 13.11) by letting the workers correspond to the warehouses and the jobs to the markets. We then take all a_i and b_j equal to 1 and let the cost be $c_{ij} = -r_{ij}$. We seek to maximize $\sum_{i,j} r_{ij}x_{ij}$ or minimize $\sum_{i,j} c_{ij}x_{ij}$. Hence, this optimal assignment problem is also a minimum-cost flow problem. [One difference is that we have the additional requirement that $x_{ij} = 0$ or 1. However, it is possible to show that if a minimum-cost flow problem has integer capacities, costs, and value v, some optimal solution x is in integers, and standard algorithms produce integer solutions (see below). Then the requirement (13.17) and the added requirement $x_{ij} \geq 0$ are sufficient to give us $x_{ij} = 0$ or 1.] ■

Example 13.13 Building Evacuation (Ahuja, Magnanti, and Orlin [1993], Chalmet, Francis, and Saunders [1982]) The September 11, 2001 World Trade Center collapse has focused attention on methods for evacuation of tall buildings. Design of such buildings must allow for rapid evacuation. Models of building evacuation can help in the building design phase. The following is a simplified

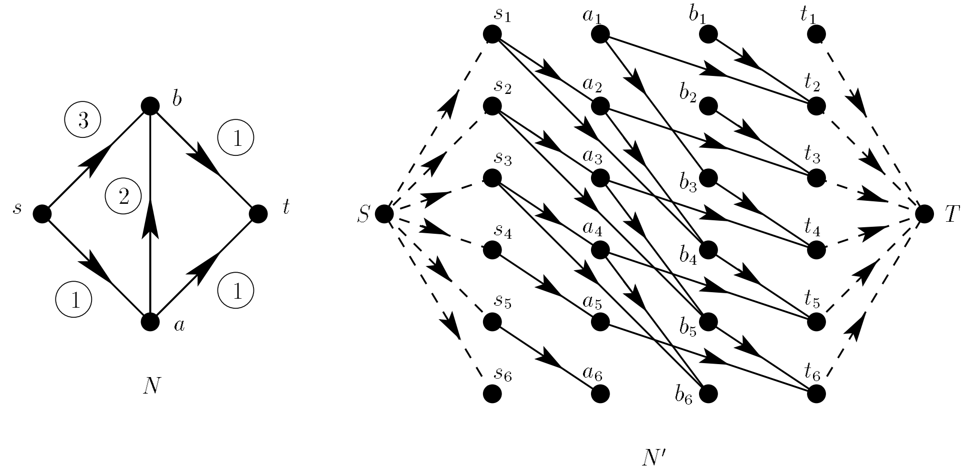


Figure 13.28: Directed networks N and N' for a hypothetical small building with $C = 6$. All capacities are 2 except for dashed arcs, which have infinite capacity (and zero cost). The numbers in the circles in N are times t_{ij}. The cost on arc (i_p, j_q) in N' is the cost on arc (i, j) in N.

model of the problem. There are many locations in a building from which people will have to be evacuated and many exits. For simplicity, we assume that there is only one location from which we need to evacuate people and only one exit, and these will become the source and sink, respectively, of a directed network N. We assume that we have a certain number v of people whom we need to evacuate to the exit. A building has various locations that will become the remaining vertices of the directed network N. We join vertex i to vertex j by an arc if there is a direct passage (stairway, hallway, etc.) from location i to location j and we let the capacity c_{ij} of this arc (i, j) be the number of people who can pass through this passage per unit time. Note that it might take several units of time t_{ij} to pass through a passage (i, j). We assume, again by way of simplifying the problem, that t_{ij} is an integer. We now replace the directed network N by a larger one, N', in which we make C copies of each of the vertices (including the source and sink), where C is chosen large enough to be sure that we can evacuate the building in p units of time. We think of copy i_p of vertex i as representing that vertex at time p. In the larger directed network N', we draw an arc from i_p to j_q if (i, j) is in N and $q - p = t_{ij}$. In other words, the arc (i_p, j_q) represents movement from vertex i to vertex j in the time it takes a person to move along the passage from i to j. Put capacity c_{ij} and "cost" t_{ij} on the new arcs (i_p, j_q). Add a source vertex S and sink vertex T and arcs (S, s_p) for all p and (t_q, T) for all q, with infinite capacities and zero cost. Figure 13.28 shows the directed networks N and N' for a hypothetical small building. A minimum "cost" flow of value v from S to T will give us a plan for evacuating all v persons from the building in a minimum amount of time. ■

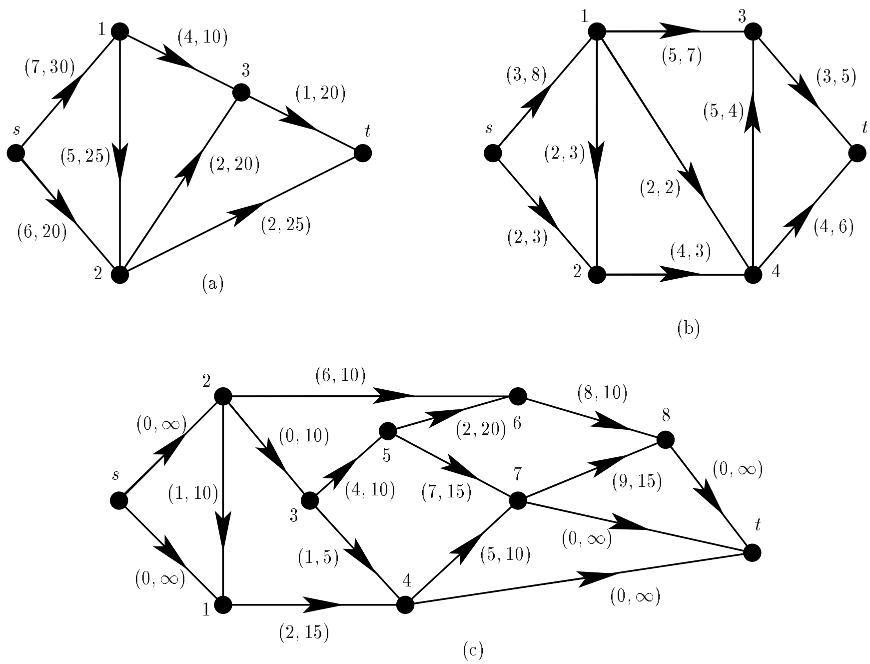


Figure 13.29: Minimum-cost flow problems for Exercise 2.

The Max-Flow Algorithm we have described in Section 13.3.5 is readily modified to give an efficient algorithm for solving the minimum-cost flow problem. (This algorithm will produce an integer solution if capacities, costs, and value v are integers.) For details, see, for example, Ahuja, Magnanti, and Orlin [1993], Lawler [1976], Minieka [1978], or Papadimitriou and Steiglitz [1982]. In special cases such as the transportation problem or the optimal assignment problem, there are more efficient algorithms.

EXERCISES FOR SECTION 13.4

1. In each directed network of Figure 13.20, consider the number in the circle as the cost rather than the flow, and consider the number in the square as the capacity as usual. In each case, find a minimum-cost flow of value 3 from s to t.

2. In each directed network of Figure 13.29, consider the ordered pair (a_{ij}, c_{ij}) on each arc (i, j) to be its cost (a_{ij}) and capacity (c_{ij}).

   (a) Find a minimum-cost flow of value 25 from s to t in directed network (a).

   (b) Find a minimum-cost flow of value 9 from s to t in directed network (b).

   (c) Find a minimum-cost flow of value 30 from s to t in directed network (c).

3. (a) Find a minimum-cost flow of value 10 for the building evacuation problem (Example 13.13) shown in Figure 13.28.

   (b) Explain your answer to part (a) in terms of how you would evacuate 10 people from the building described in Figure 13.28.

   (c) What is the largest number of people that can be evacuated from the building described in Figure 13.28? What is the minimum-cost flow for this value?

4. (Ahuja, Magnanti, and Orlin [1993]) An airline has 6 daily flights from New York to Chicago. They leave every two hours starting at 7 AM. The planes used on the morning flights have a 100-person capacity. After noon, the planes used have a 150-person capacity. The airline can "bump" overbooked passengers onto a later flight. After 5 PM, the airline can put a passenger on another airline's 10 PM flight, on which there are always seats. Bumped passengers are compensated at the rate of \$200 plus \$20 per hour of delay on any delay over 2 hours. Suppose that on a certain day, the airline has bookings of 110, 160, 103, 149, 175, and 140 people on their 6 flights. Formulate the problem of minimizing compensation costs as a minimum-cost flow problem.

5. A caterer knows in advance that for the next n days, a_j napkins will be needed on the jth day, $j = 1, 2, \ldots, n$. For any given day, the caterer can buy new napkins or use laundered napkins. Laundering of napkins can be done either by quick service, which takes q days, or by slow service, which takes r days. The cost of a new napkin is c cents, the cost of laundering a napkin quickly is d cents, and the cost of laundering one slowly is e cents. Starting with no napkins, how would the caterer meet the napkin requirement with minimum cost? Set this up as a minimum-cost flow problem. (*Note:* An analogous problem, which predates this problem historically, involves aircraft maintenance, with either quick or slow overhaul of engines.)

6. (Ahuja, Magnanti, and Orlin [1993], Prager [1957]). An investor considers investments in gold in T future time periods. In each time period, he can buy, sell, or hold gold that he already owns. Suppose that in period i, he can buy at most α_i ounces of gold, hold at most β_i ounces of gold, and must sell at least γ_i ounces of gold (due to prior contracts), and assume that he must hold purchased gold at least until the next time period. The costs involved with investing in gold are per ounce purchase cost p_i and per ounce selling price s_i in period i and per ounce holding cost w_i during period i. How should the investor make purchases, sales, and holds during the T time periods in order to maximize profit? Formulate the problem as a minimum-cost flow problem.

7. In the investment problem of Exercise 6, suppose we assume that the various restrictions and costs are independent of i, but that if you buy gold, you must order it two time periods in advance, and if you decide to hold gold in any time period, you must hold it for three time periods. How would you analyze this investment problem? (*Hint*: Compare with Example 13.13, building evacuation.)

8. (Cook, *et al.* [1998]) Suppose that D is a weakly connected digraph. We can "duplicate" any arc $a = (u, v)$ of D at cost $c(a)$, where by duplicating we mean adding another arc from u to v. We can make as many duplicates of an arc as we like. We want to duplicate as few arcs as possible in order to obtain a multidigraph that has an eulerian closed path. Formulate this as a minimum-cost flow problem.

(A similar problem arises in connection with the "Chinese Postman" Problem of Section 12.7.1.)

9. Suppose that we have two warehouses and two markets. There are 10 spools of wire at the first warehouse and 14 at the second. Moreover, 13 are required at the first market and 11 at the second. If the following matrix gives the transportation costs, find the minimum-cost transportation schedule.

$$\begin{array}{lc} & \text{Factory} \\ & \begin{array}{cc} 1 & 2 \end{array} \\ \text{Warehouse} \begin{array}{c} 1 \\ 2 \end{array} & \begin{array}{|cc|} \hline 100 & 84 \\ 69 & 75 \\ \hline \end{array} \end{array}$$

10. Consider the transportation problem (Example 13.11).

 (a) Show that if $\sum a_i < \sum b_j$, there is no shipping pattern that meets requirements (13.13) and (13.14).

 (b) If $\sum a_i > \sum b_j$, show that we may as well assume that $\sum a_i = \sum b_j$ by creating a new $(m+1)$st market, setting $b_{m+1} = \sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j$, adding arcs from each warehouse to the new market, and letting the capacities of all new arcs be 0.

11. In a directed network with costs, modify the definition of an augmenting chain to allow it to start at a vertex other than s and end at a vertex other than t. Define the cost of an augmenting chain to be the sum of the costs of forward arcs minus the sum of the costs of backward arcs. An *augmenting circuit* is an augmenting chain that forms a circuit.

 (a) In each example for Exercise 1, find a flow of value 3 that *does not* have minimum cost and find a flow-augmenting circuit with negative cost.

 (b) Show that if a flow of value v is of minimum cost, it admits no flow-augmenting circuit with negative cost.

 (c) Show that if a flow of value v admits no flow-augmenting circuit of negative cost, the flow has minimum cost.

12. We can reduce a transportation problem (Example 13.11) to a maximum-weight matching problem (Section 12.1) as follows. Build a bipartite graph $G = (X, Y, E)$ of $2\sum a_i$ vertices, as follows. Let X consist of a_i copies of the ith warehouse, $i = 1, 2, \ldots, n$, and let Y consist of b_j copies of the jth market, $j = 1, 2, \ldots, m$. G has all possible edges between vertices in X and in Y. On the edge joining a copy of the ith warehouse to a copy of the jth market, place a weight equal to $K - a_{ij}$, where K is sufficiently large. Show that an optimal solution to the transportation problem corresponds to a maximum-weight matching in G.

REFERENCES FOR CHAPTER 13

AHUJA, R. K., MAGNANTI, T. L., and ORLIN, J. B., "Network Flows," in G. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd (eds.), *Handbooks in Operations Research and Management Science*, Vol. 1: *Optimization*, North-Holland, Amsterdam, 1989, 211–369.

AHUJA, R. K., MAGNANTI, T. L., and ORLIN, J. B., "Some Recent Advances in Network Flows," *SIAM Review, 33* (1991), 175–219.

AHUJA, R. K., MAGNANTI, T. L., and ORLIN, J. B., *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.

ARONSON, J. E., "A Survey of Dynamic Network Flows," *Ann. Oper. Res., 20* (1989), 1–66.

ARRATIA, R., and LANDER, E. S., "The Distribution of Clusters in Random Graphs," *Adv. Appl. Math., 11* (1990), 36–48.

BARITCHI, A., COOK, D. J., and HOLDER, L. B., "Discovering Structural Patterns in Telecommunications Data," in *Proceedings of the Thirteenth Annual Florida AI Research Symposium*, 2000.

BAZARAA, M. S., JARVIS, J. J., and SHERALI, H. D., *Linear Programming and Network Flows*, 2nd ed., Wiley, New York, 1990.

BERGE, C., and GHOUILA-HOURI, A., *Programming, Games and Transportation Networks*, Wiley, New York, 1965.

BONDY, J. A., and MURTY, U. S. R., *Graph Theory with Applications*, American Elsevier, New York, 1976.

BORŮVKA, O., "O Jistém Problému Minimálním," *Práce Mor. Pŕírodoved. Spol. v Brne (Acta Soc. Sci. Nat. Moravicae), 3* (1926), 37–58. (a)

BORŮVKA, O., "Příspevěk k. Řešení Otázky Ekonomické Stavby Elektrovodních Sítí," *Elektrotech. Obzor, 15* (1926), 153–154. (b)

CHALMET, L. G., FRANCIS, R. L., and SAUNDERS, P. B., "Network Models for Building Evacuation," *Management Sci., 28* (1982), 86–105.

CHERITON, D., and TARJAN, R. E., "Finding Minimum Spanning Trees," *SIAM J. Comput., 5* (1976), 724–742.

CHURCH, K., "Massive Data Sets and Graph Algorithms in Telecommunications Systems," Session on Mathematical, Statistical, and Algorithmic Problems of Very Large Data Sets, American Mathematical Society Meeting, San Diego, CA, January 1997.

COOK, W. J., CUNNINGHAM, W. H., PULLEYBLANK, W. R., and SCHRIJVER, A., *Combinatorial Optimization*, Wiley, New York, 1998.

CZEKANOWSKI, J., "Zur Differentialdiagnose der Neandertalgruppe," *Korrespondenzbl. Dtsch. Ges. Anthrop., Ethn. Urg., 40* (1909), 44–47.

CZEKANOWSKI, J., "Objektive Kriterien in der Ethnologie," *Ebenda, 43* (1911), 71–75.

CZEKANOWSKI, J., "Das Typenfrequenzgesetz," *Anthropol. Anz., 5* (1928), 15–20.

DIJKSTRA, E. W., "A Note on Two Problems in Connexion with Graphs," *Numer. Math., 1* (1959), 269–271.

EDMONDS, J., and KARP, R. M., "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *J. ACM, 19* (1972), 248–264.

ELIAS, P., FEINSTEIN, A., and SHANNON, C. E., "Note on Maximum Flow through a Network," *IRE Trans. Inf. Theory, IT-2* (1956), 117–119.

FORD, L. R., and FULKERSON, D. R., "Maximal Flow through a Network," *Canad. J. Math., 8* (1956), 399–404.

FORD, L. R., and FULKERSON, D. R., "A Simple Algorithm for Finding Maximal Network Flows and an Application to the Hitchcock Problem," *Canad. J. Math., 9* (1957), 210–218.

FORD, L. R., and FULKERSON, D. R., *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

FRANK, C. R., "A Note on the Assortment Problem," *Management Science, 11* (1965),

724–726.

Frank, H., and Frisch, I. T., *Communication, Transportation and Flow Networks*, Addison-Wesley, Reading, MA, 1971.

Gabow, H. N., Galil, Z., Spencer, T., and Tarjan, R. E., "Efficient Algorithms for Finding Minimum Spanning Trees in Undirected and Directed Graphs," *Combinatorics, 6* (1986), 109–122.

Glover, F., and Klingman, D., "Network Applications in Industry and Government," *AIIE Trans., 9* (1977), 363–376.

Godehart, E., *Graphs as Structural Models*, 2nd ed., Friedr. Vieweg & Sohn, Braunschweig, Germany, 1990.

Goldman, A. J., "Discrete Mathematics in Government," lecture presented at SIAM Symposium on Applications of Discrete Mathematics, Troy, NY, June 1981.

Gondran, M., and Minoux, M., *Graphs and Algorithms*, Wiley, New York, 1984.

Gower, J. C., and Ross, G. J. S., "Minimum Spanning Trees and Single Linkage Cluster Cluster Analysis," *Applied Statistics, 18*, (1969), 54–64.

Graham, R. L., and Hell, P., "On the History of the Minimum Spanning Tree Problem," *Annals of the History of Computing, 7* (1985), 43–57.

Guénoche, A., Hansen, P., and Jaumard, B., "Efficient Algorithms for Divisive Hierarchical Clustering with the Diameter Criterion," *J. Classification, 8* (1991), 5–30.

Hansen, P., Frank, O., and Jaumard, B., "Maximum Sum of Splits Clustering," *J. Classification, 6* (1989), 177–193.

Hansen, P., and Jaumard, B., "Minimum Sum of Diameters Clustering," *J. Classification, 4* (1987), 215–226.

Hansen, P., Jaumard, B., and Mladenovic, N., "Minimum Sum of Squares Clustering in a Low Dimensional Space," *J. Classification, 15* (1998), 37–55.

Hu, T. C., *Integer Programming and Network Flows*, Addison-Wesley, Reading, MA, 1969.

Iri, M., *Network Flows, Transportation and Scheduling*, Academic Press, New York, 1969.

Jarník, V., "O Jistém Problému Minimálním," *Práce Mor. Prírodoved Spol. v Brne (Acta Soc. Sci. Nat. Moravicae), 6* (1930), 57–63.

Johnson, D. B., "Priority Queues with Update and Minimum Spanning Trees," *Inf. Process. Lett., 4* (1975), 53–57.

Karger, D., "Information Retrieval: Challenges in Interactive-Time Manipulation of Massive Text Collections," Session on Mathematical, Statistical, and Algorithmic Problems of Very Large Data Sets, American Mathematical Society Meeting, San Diego, CA, January 1997.

Kershenbaum, A., and Van Slyke, R., Computing Minimum Spanning Trees Efficiently, ACM 72, *Proceedings of the Annual ACM Conference*, 1972, 518–527.

Kruskal, J. B., "On the Shortest Spanning Tree of a Graph and the Traveling Salesman Problem," *Proc. Amer. Math. Soc., 7* (1956), 48–50.

Lawler, E. L., *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

Menger, K., "Zur allgemeinen Kurventheorie," *Fund. Math., 10* (1927), 96–115.

Minieka, E., *Optimization Algorithms for Networks and Graphs*, Dekker, New York, 1978.

Mirkin, B., *Mathematical Classification and Clustering*, Kluwer Academic Publishers, Dordrecht-Boston-London, 1996.

Neil, M. D., "Multivariate Assessment of Software Products," *Software Testing, Veri-*

fication & Reliability, 1 (1992), 17–37.

Nešetřil, J., Milková, E., and Nešetřilová, H., "Otakar Borůvka on Minimum Spanning Tree Problem: Translation of Both the 1926 Papers, Comments, History," *Discrete Math., 233* (2001), 3–36.

Papadimitriou, C. H., and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, NJ, 1982.

Prager, W., "On Warehousing Problems," *Oper. Res., 5* (1957), 504–512.

Prim, R. C., "Shortest Connection Networks and Some Generalizations," *Bell Syst. Tech. J., 36* (1957), 1389–1401.

Ravindran, A., "On Compact Book Storage in Libraries," *Opsearch, 8* (1971), 245–252.

Stone, H. S., "Multiprocessor Scheduling with the Aid of Network Flow Algorithms," *IEEE Trans. Software Engrg., 3* (1977), 85–93.

Tarjan, R. E., *Data Structures and Network Algorithms*, SIAM, Philadelphia, 1983.

Tarjan, R. E., "A Simple Version of Karzanov's Blocking Flow Algorithm," *Oper. Res. Lett., 2* (1984), 265–268. (a)

Tarjan, R. E., "Efficient Algorithms for Network Optimization," *Proceedings of the International Congress of Mathematicians*, (Warsaw, 1983) Vols. 1, 2 (1984), 1619–1635. (b)

White, L. S., "Shortest Route Models for the Allocation of Inspection Effort on a Production Line," *Management Sci., 15* (1969), 249–259.

Yao, A., "An $O(|E| \log \log |V|)$ Algorithm for Finding Minimum Spanning Trees," *Information Processing Letters, 4* (1975), 21–23.

Zahn, C. T., "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters," *IEEE Trans. Computing, C-20* (1971), 68–86.

Appendix

Answers to Selected Exercises

Chapter 1

Section 1.1.

No exercises.

Section 1.2.

1. $\begin{array}{|cccc|}\hline 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \\ \hline \end{array}$.

3. Let row 1 be: $1\ 2\ \cdots\ n$. Row 2 is gotten by taking the first element (1) of row 1 and moving it to the end of the row. Row 3 is gotten from row 2 by taking the first element (2) of row 2 and moving it to the end of the row. Continue until you have n rows.

4(b). $\begin{array}{|cccc|}\hline 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ \hline \end{array}$; $\begin{array}{|cccc|}\hline a & b & c & d \\ c & d & a & b \\ d & c & b & a \\ b & a & d & c \\ \hline \end{array}$.

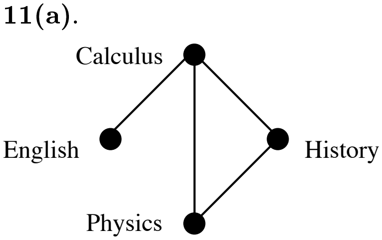
5. 0, 1, 00, 01, 10, 11, 000, 001, 010, 100, 011, 101, 110, 111.

7(a). No - there are only 12 such strings.

8. There are 84 such strings: 4 of length one; 16 of length two; 64 of length three.

9. LLLL, LLLS, LLSL, LSLL, SLLL, LLSS, LSLS, SLLS, LSSL, SLSL, SSLL, LSSS, SLSS, SSLS, SSSL, SSSS.

11(a).



12(a). No assignment exists. Each of Calculus, History, and Physics must get a different exam time since they overlap with one another.

12(d). Time 1: English and Physics; time 2: Calculus; time 3: History; time 4: Economics.

13. If Economics is Wednesday and Transportation is Tuesday then both Housing and Health must be Thursday - this is not possible. Or, if Economics is Wednesday and Transportation is Thursday then both Housing and Health must be Tuesday - again this is not possible.

14(a). If English must be Thur. AM, then Calculus must be Wed. AM. But then History and Physics must be Tues. AM - this is not possible.

Chapter 2

Section 2.1.

1. Yes: $26^3 < 20,000$. **3(a)**. $3^1 + 3^2 + 3^3$. **3(c)**. $2 \cdot 3^3$.
4. $8^3 \times 10^5; 8 \times 2 \times 10 \times 8^3 \times 10^5$. **6**. 2^{mn}. **8**. $10^6 - 9^6$.
10. $2^{2^{2^n}}$. **12**. 2^p; $2^p - 1$. **13(b)**. $2^{m-1}2^{p-m}$. **14(a)**. $3 \cdot 2 \cdot 3$.

Section 2.2.

1. $2^3 + 2^4 + 2^5$. **3**. $5^5 + 4^5$. **5**. $26^4 + 25^5$. **7**. $(7 \cdot 3)^{30}$.

Section 2.3.

1(a). 123, 132, 213, 231, 312, 321. **3**. $(n-2)!$. **4(b)**. $s_6 \approx 710$ vs. $6! = 720$.
6(b). $(n-2)!$. **8(b)**. $n \cdot n \cdot n - 1 \cdot n - 1 \cdot \cdots \cdot 2 \cdot 2 \cdot 1 \cdot 1$.

Section 2.4.

2. $25! \cdot \frac{1}{10^{11}} \cdot \frac{1}{3.15 \times 10^7} \approx 4.9 \times 10^6$ years.

3. There are $n!$ schedules. Each committee in each schedule must be checked to see if it received its first choice - this takes n steps per schedule. The computational complexity is $f(n) = n \cdot n!$.

6. Best order is 2, 3, 1. **7(b)**. $\frac{n+1}{2} \times 3 \times 10^{-9}$. **8(b)**. $\frac{n+1}{2} \times 3 \times 10^{-11}$.

Section 2.5.

1(b). $5 \cdot 4 \cdot 3$. **1(c)**. $8 \cdot 7 \cdot 6 \cdot 5 \cdot 4$. **2(d)**. $1 \cdot 5 \cdot 4$. **3(d)**. $1 \cdot 6 \cdot 5 \cdot 1$. **6**. 40^3.

Section 2.6.

2. 2^{35}. **3**. 2^7. **5**. $2^{10} - 1$. **7(a)**. 2^8. **8(a)**. 2^{2^3}.

Section 2.7.

2. $C(50,7)$. **3(a)**. $\frac{6!}{3!(6-3)!} = 20$.

5. $\frac{5!}{2!3!} = 10$; $\{a,b\}, \{a,c\}, \{a,d\}, \{a,e\}, \{b,c\}, \{b,d\}, \{b,e\}, \{c,d\}, \{c,e\}, \{d,e\}$.

7(b). $C(6,4) = \frac{6!}{4!(6-4)!} = \frac{6!}{4!2!}$ and $C(6,2) = \frac{6!}{2!(6-2)!} = \frac{6!}{2!4!}$.

8. 1 7 21 35 35 21 7 1.

10. $C(7,5) = \frac{7!}{5!2!} = 21$, $C(6,4) = \frac{6!}{4!2!} = 15$, $C(6,5) = \frac{6!}{5!1!} = 6$, and $21 = 15 + 6$.

11(c). $2^8 - 2$. **13(b).** $2^{10} - 2$.

14. $C(21,5)C(5,3)8! + C(21,4)C(5,4)8! + C(21,3)C(5,5)8!$.

16(d). $C(7,2) \cdot C(4,2) - C(6,1) \cdot C(3,1)$. **17(a)(iii).** 2.

17(b). JAVA, JAVA, JAVA, JAVA, JAVA, C++, C++, C++, C++. **17(c)(i).** 9!.

18(a). $[C(3,1)C(27,2) + C(3,2)C(27,1) + C(3,3)C(27,0)] \times [C(12,1)C(138,2) + C(12,2)C(138,1) + C(12,3)C(138,0)]$.

19(a). $\binom{n}{m}\binom{m}{k} = \frac{n!}{m!(n-m)!}\frac{m!}{k!(m-k)!} = \frac{n!}{(n-m)!k!(m-k)!}$;
$\binom{n}{k}\binom{n-k}{m-k} = \frac{n!}{k!(n-k)!}\frac{(n-k)!}{(m-k)!((n-k)-(m-k))!} = \frac{n!}{k!(m-k)!(n-m)!}$.

21. Sum the entries in the row labeled n, i.e., the $(n+1)^{st}$ row since the labels start with 0. For $n = 2$: $1 + 2 + 1 = 4$; for $n = 3$: $1 + 3 + 3 + 1 = 8$; for $n = 4$: $1 + 4 + 6 + 4 + 1 = 16$; in general, 2^n.

24(a). $\left\langle {n \atop r} \right\rangle = \binom{n+r-1}{r}$ and $\left\langle {n \atop r-1} \right\rangle + \left\langle {n-1 \atop r} \right\rangle = \binom{n+r-2}{r-1} + \binom{n+r-2}{r}$ are equal using equation (2.3).

25. $\left\langle {n \atop r} \right\rangle = \binom{n+r-1}{r} = \frac{(n+r-1)!}{r!(n-1)!}$ and $\frac{n}{r}\left\langle {n+1 \atop r-1} \right\rangle = \frac{n}{r}\binom{n+r-1}{r-1} = \frac{n}{r}\frac{(n+r-1)!}{(r-1)!n!} = \frac{(n+r-1)!}{r!(n-1)!}$ and $\frac{n+r-1}{r}\left\langle {n \atop r-1} \right\rangle = \frac{n+r-1}{r}\binom{n+r-2}{r-1} = \frac{n+r-1}{r}\frac{(n+r-2)!}{(r-1)!(n-1)!} = \frac{(n+r-1)!}{r!(n-1)!}$.

26(b). The sequence is strictly increasing when $\binom{n}{i} < \binom{n}{i+1}$ or $\frac{n!}{i!(n-i)!} < \frac{n!}{(i+1)!(n-i-1)!}$ or $i!(n-i)! > (i+1)!(n-i-1)!$ or $n - i > i + 1$ or $i < \frac{n-1}{2}$. Similarly, it is strictly decreasing when $i > \frac{n-1}{2}$.

Section 2.8.

1(c). Yes. **1(e).** No. **2(c).** 1/3. **3(c).** $\frac{3}{4}$. **4.** $\dfrac{C(3,2) \cdot 2 + C(3,3)}{3^3} = \dfrac{7}{27}$.

6. $\frac{16}{4^5}$. **8.** $\frac{4}{52} \cdot \frac{4}{52}$. **10.** $\frac{5}{8}$. **11.** $\dfrac{C(4,3) + C(4,4)}{4^2} = \dfrac{5}{16}$. **13.** $\frac{21}{32}$.

15(a). $C(6,2)/2^6 + C(6,3)/2^6$. **15(e).** $C(5,1)/2^6 + C(5,3)/2^6 + C(5,5)/2^6$.

16(b). E and F disjoint implies $n(E \cup F) = n(E) + n(F)$. So, probability of $E \cup F = \frac{n(E \cup F)}{n(S)} = \frac{n(E)+n(F)}{n(S)}$ and probability of $E +$ probability of $F = \frac{n(E)}{n(S)} + \frac{n(F)}{n(S)}$.

17(a). $2^2/2^4 + 2^2/2^4 - 1/2^4$.

Section 2.9.

1(b). $aa, ab, ac, ba, bb, bc, ca, cb, cc$. **1(d).** $\{a,a\}, \{a,b\}, \{a,c\}, \{b,b\}, \{b,c\}, \{c,c\}$.

2(b). $P^R(3,2) = 3^2$. **2(d).** $C^R(3,2) = C(3+2-1,2) = 6$. **3(a).** $P^R(3,7) = 3^7$.

4. $C^R(4,8) = C(4+8-1,8)$. **6.** $P^R(5,8) - P^R(5,7) = 5^8 - 5^7$.

7(a). $C^R(4,2) = C(4+2-1,2) = 10$.

8(b). $C^R(2,82) = C(2+82-1,82) = 83$. **10.** $\sum_{j=0}^{400} C^R(3,j)$.

Section 2.10.

1(a).

		Distribution							
		1	2	3	4	5	6	7	8
Cell	1	abc		ab	c	ac	b	bc	a
	2		abc	c	ab	b	ac	a	bc

1(d).

		Distribution			
		1	2	3	4
Cell	1	aaa		aa	a
	2		aaa	a	aa

3(b). $2^4 = 16$. **3(f).** $C(4+2-1,2) = 10$.

4(a). $S(3,1) + S(3,2) = 4$.

4(d). Number of partitions of 3 into two or fewer parts = 2. **5(a).** $2!S(3,2) = 6$.

5(d). $C(2,1) = 2$. **6(a).** $S(3,2) = 3$.

6(d). Number of partitions of 3 into exactly two parts = 1.

7(a). $\{1,1,1,1\}, \{1,1,2\}, \{2,2\}, \{4\}, \{1,3\}$. **9(b).** 1. **9(d).** $\binom{n}{2}$.

11. 25 indistinguishable balls (misprints), 75 distinguishable cells (pages), cells can be empty: $C(75+25-1,25)$.

13. 9 distinguishable balls (passengers), 5 indistinguishable cells (floors), cells can be empty: $S(9,1) + S(9,2) + S(9,3) + S(9,4) + S(9,5)$.

15. $C(6+30-1,30)$. **17.** $\dfrac{10!}{2^5 5!} = 945$. **20.** $1/C(11,8)$.

22(a). $S(n,k)$ counts the number of ways to place n distinguishable balls into k indistinguishable cells with no cell empty. Consider the n^{th} ball. Either it is by itself in a cell OR it is with other balls. $S(n-1,k-1)$ counts the number of ways where the n^{th} ball is by itself: To assure it is by itself, place the remaining $n-1$ balls in $k-1$ cells with no cell empty and then put the n^{th} ball in its own k^{th} cell. $kS(n-1,k)$ counts the number of ways where the n^{th} ball is with other balls in some cell: To assure that it is with other balls, place the remaining $n-1$ balls in k cells *with no cell empty* and then put the n^{th} ball in one of the k (nonempty) cells (i.e., there are k choices for the n^{th} ball).

24(a). 16. One ordering of $\{1,1,1,1,1\}$, 4 orderings of $\{1,1,1,2\}$, 3 orderings of $\{1,2,2\}$, 3 orderings of $\{1,1,3\}$, 2 orderings of $\{2,3\}$, 2 orderings of $\{1,4\}$, and 1 ordering of $\{5\}$.

Section 2.11.

1(a). 630. **1(d).** 90/729.

2. $C(n;1,1,1,\ldots,1) = \begin{pmatrix} n \\ 1,1,1,\ldots,1 \end{pmatrix} = \dfrac{n!}{1!1!1!\cdots 1!} = n!$.

6(a). $C(7;4,1,2) = 105$. **9(a).** $C(9;3,2,1,1,1,1) = 30,240$.

9(b). $\dfrac{C(9;3,2,1,1,1,1)}{26^9} = \dfrac{30,240}{26^9} \approx 5.56957 \times 10^{-9}$.

11. 4!. Since there are 5 a's and 4 non-a's, any such permutation must start with an a and alternate a's and non-a's. Thus, we need only count how to order the four non-a's.

13(a). $C(5;3,1,1)$. **16(b)**. $P(4;3,1)$. **16(d)**. $P(4;2,1,1) = 12$.

Section 2.12.

1(a). $3! = 6$. **1(b)**. $\dfrac{5!}{2} = 60$. **1(c)**. CAAGCUGGUC. **3(a)**. $\dfrac{4!}{3!} = 4$. **3(b)**. $4!$.

3(c). GUCGGGUU and GGGUCGUU. **7(a)**. $\dfrac{3!}{2!} = 3$. **7(b)**. $\dfrac{5!}{3!2!} = 10$.

7(c). 00101010, 01001010 and 01010010. **8**. GCGUGU and GUGCGU.

10. Yes: 00101010 and 01001010 have the same breakup.

Section 2.13.

1. $9/C(12,4)$.

3(a). The given *order* has a probability of $\frac{1}{C(13;8,5)}$ of being observed.

3(b). $\frac{C(9;8,1)}{C(13;8,5)}$. Group the 5 sick trees together as one unit S^*. Then the number of orders of 1 S^* and 8 W's is $C(9;8,1)$.

5(a). Group the 6 infested houses together as one unit I^*. Then the number of orders of 1 I^* and 5 noninfested houses is $C(6;5,1)$.

5(b). Only 1: Start with an infested and alternate. **8(a)**. $7/4^4$. **9**. $C(17;13,4)$.

10. $\dbinom{15}{5,4,5,1}$. This is equivalent to RNA chains of length 15 having five A's, four U's, five H's and one G, where H = "CG."

Section 2.14.

1(b). $\binom{3}{0}a^3 + \binom{3}{1}a^2 2b + \binom{3}{2}a(2b)^2 + \binom{3}{3}(2b)^3 = a^3 + 6a^2b + 12ab^2 + 8b^3$.

2(b). $\binom{14}{11}2^3$. **3**. $C(12,9)C(4,0) + C(12,8)C(4,1) + \cdots + C(12,5)C(4,4)$.

6. $\binom{6}{2,2,2}$. **8**. $\displaystyle\sum_{\substack{n_i \geq 0 \\ n_1+n_2+\cdots+n_k=n}} \binom{n}{n_1, n_2, \ldots, n_k} a_1^{n_1} a_2^{n_2} \cdots a_k^{n_k}$. **10**. $\binom{6}{1,2,2,1} \cdot 2$.

13. $\frac{1}{2} \cdot 2^{12}$. **15**. $\frac{1}{2} \cdot 2^n = 2^{n-1}$. **16(b)**. 5^n. **16(e)**. $n2^{n-1}$.

17(a). Start with $(x+2)^n = \displaystyle\sum_{k=0}^{n} \binom{n}{k} 2^{n-k} x^k$, differentiate. and then let $x = 2$.

Section 2.15.

1(a). $\{1,2\}, \{1,3\}, \{1,2,3\}$. **2(a)**. $\{1,2,5\}, \{1,3\}, \{2,3\}, \{1,4\}, \{2,4\}, \{3,4\}, \{4,5\}$.

3(b). $\frac{1}{5}$ for each. **6(a)**. Yes - $[5;5,2,1]$. **11(b)**. nonpermanent: $\frac{9!9!}{3!16!}$.

Section 2.16.

1(a). 2143 precedes 3412. **2(b)**. 0101. **3(c)**. $\{1,3,5,6\}$. **4(c)**. 152634.

11. $\{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,3\}, \{2,4\}, \{2,5\}, \{3,4\}, \{3,5\}, \{4,5\}$.

12. $0111 \cdots 1$. **21(b)**. 4.

Section 2.17.

3. 456123. **5(c)**. $n12\cdots(n-1)$. **5(d)**. 2.

7(b). 5: $15423 \to 14523 \to 14253 \to 14235 \to 12435 \to 12345$.

9(a). 2: Transpose 7 and 46 to get 5123467; then transpose 5 and 1234. **10(a)**. 2.

Section 2.18.

1(b). Yes. **1(d)**. Yes.

4. $f(n) \leq k_1 g(n)$ for some positive constant k_1 and $n \geq r_1$; $g(n) \leq k_2 h(n)$ for some positive constant k_2 and $n \geq r_2$; then $f(n) \leq k_1 k_2 h(n)$ for positive constant $k_1 k_2$ and $n \geq \max\{r_1, r_2\}$.

8(a). Yes. **9(a)**. Yes. **9(f)**. Yes. **9(i)**. Yes.

12. The third is "big oh" of the other two; the first is "big oh" of the second.

Section 2.19.

1(a). 27. **2(a)**. 53. **3**. 6. **4(a)**. 4.

6. Yes; use Corollary 2.15.1 and the fact that the average number of seats per car is $\frac{465}{95} \approx 4.89$.

9(a). 9. **9(b)**. 20.

13(a). Longest increasing is 6, 7 or 5, 7 and longest decreasing is 6, 5, 4, 1.

14. $13, 14, 15, 16, 9, 10, 11, 12, 5, 6, 7, 8, 1, 2, 3, 4$.

15. Let the pigeons be the 81 hours and the five holes be days 1 and 2, 3 and 4, 5 and 6, 7 and 8, and 9 and 10.

24. If there are n people, each has at most $n-1$ acquaintances.

29(a). $X = \{\{a,b\},\{b,c\},\{c,d\}\}, Y = \{\{a,c\},\{b,d\},\{a,d\}\}$. **30(a)**. 2.

Additional Exercises for Chapter 2.

No answers provided.

Chapter 3

Section 3.1.

1(a). $V =$ {Chicago (C), Springfield (S), Albany (A), New York (N), Miami (M)}.

1(b). $A =$
{(C, S), (S, C), (C, N), (N, C), (A, N), (N, A), (C, M), (M, C), (N, M), (M, N)}.

4(b). In $G_3, E = \{\{u,v\},\{v,w\},\{u,w\},\{x,y\},\{x,z\},\{y,z\}\}$.

6. {Chicago, Albany, Miami}.

15. Outline: The sum of the degrees of all the vertices is precisely $2e$, so by the Pigeonhole Principle there is some vertex whose degree is at least $\frac{2e}{n}$.

17(a). Yes. **17(b)**. No. **19**. 15; **20**. 32;

23. No: The first graph has four edges, and the second graph has three edges.

24. Yes. **26**. Yes. **29**. no.

Section 3.2.

1(d). Yes. **5**. D_4: No; D_6: Yes. **6(a)**. No. **6(e)**. Yes. **9(b)**. D_4: Yes; D_8: No. **10(c)**. D_4: Yes; D_8: Yes. **13**. D_8: $\{p, q, r, s, t\}, \{u\}, \{v\}, \{w\}$.

15(a). Let a vertex v be in D. The set consisting of v itself generates a strongly connected subgraph. A maximal strongly connected generated subgraph containing v is therefore a strong component.

15(b). Outline: Suppose a vertex v was in two different strong components, A and B. Then $A \cup B$ is also a strongly connected generated subgraph containing v.

18. *Hint:* Use induction on the number of vertices. **22(b)**. No. **25(a)**. Yes.

28. Outline: Given v, let w be another vertex. There are paths from v to w and w to v, and together they form a closed path containing v. A shortest closed path containing v must be a cycle.

30(a). 9. **30(b)**. $2\binom{n-1}{2} + (n-1) = (n-1)^2$.

Section 3.3.

3(a). (b): Yes. **3(b)**. (b): 3. **5**. No. **8**. 4.

12. For graph (a): (a) $\omega(G) = 4$. (b) $\alpha(G) = 2$. (c) $\chi(G) = 4$ and it's true that $2 \geq \frac{4}{4}$.

14. The largest set can have at most five vertices.

16. Since $\frac{|V(G)|}{\theta(G)}$ is the average size of a clique in S, then there is a clique of size $\frac{|V(G)|}{\omega(G)}$. Since each vertex in a clique must have its own color, we have $\frac{|V(G)|}{\omega(G)} \leq n$.

18. No. **21**. (b): Yes.

26. Outline: If we order the vertices arbitrarily and color each one successively, we will always be guaranteed to have a color available for the vertex currently being colored.

28. Any complete graph is weakly γ-perfect.

32. Outline: The odd-length closed chain can be decomposed into k circuits, where the sum of the lengths of the circuits equals the length of the odd-length closed chain. If all the circuits have even length, then the original chain would have even length also, a contradiction.

33. Outline: In the forward direction: Show that if p is even, then $\chi(\mathcal{Z}_p) = 2$. In the reverse direction: Exhibit a 3-coloring of $\mathcal{Z}_p$.

37. Outline: If $i \neq j$ are different colors in our original coloring, then they will become $i(t+1)$ and $j(t+1)$ in the $T - coloring$. Then, assuming $i > j$ (without loss of generality) we have $i(t+1) - j(t+1) = (i-j)(t+1) > t$.

41(a). 4. **41(e)**. 9.

43. Outline: We must have $\chi_m(G) \geq 2m$ since otherwise any two m-sets will intersect. Given a 2-coloring of G, we produce an m-fold coloring by letting all vertices that were colored 1 have sets $\{1, 2, \ldots, m\}$ and all vertices that were colored 2 have sets $\{m+1, m+2, \ldots, 2m\}$.

45(a). No. **45(b)**. Yes. **49(a)**. Z_3 is not 2-choosable since it is not 2-colorable.

49(b). Z_3 is 3-choosable since we can arbitrarily color the vertices in any order, and we will still be guaranteed to have a color available every time.

51. There are $k^{|V(G)|}$ colorings possible. **54(b)**. Z_n, n odd, $n \geq 3$.

55(a). $\omega(G) = \alpha(G^c)$.

Section 3.4.

1(a). $x(x-1)^3$. **2(a)**. 24. **2(c)**. 48. **4**. $P(L_n, x) = x(x-1)^{n-1}$.

6. $[P(I_2, x) - P(I_1, x)][P(I_1, x) - 2]^2$. **7**. $P(Z_n, x) = P(L_n, x) - P(Z_{n-1}, x)$.

9(a). $5 \cdot 4!$; $\binom{5}{2} \cdot 4!$. **11(a)**. 2. **11(c)**. (a): 0.

13(d). $P(x) \neq x^n$ and the sum of the coefficients is not zero.

15. Sketch of proof: Take any edge and apply the Fundamental Reduction Theorem to obtain $P(G, x) = P(G - e, x) - P(G/e, x)$. By induction, $P(G - e, x)$ has leading term x^n and $P(G/e, x)$ has leading term x^{n-1} so the difference has leading term x^n.

20(b). Yes.

22(a).
$$\begin{aligned}(-1)^n[P(Z_n, x) - (x-1)^n] &= (-1)^n[P(L_n, x) - P(Z_n, x) - (x-1)^n] \\ &= (-1)^n[x(x-1)^{n-1} - (x-1)^n - P(Z_n, x)] \\ &= (-1)^n[(x-1)^{n-1} - P(Z_{n-1}, x)] \\ &= (-1)^{n-1}[P(Z_{n-1}, x) - (x-1)^{n-1}].\end{aligned}$$

22(b). We know that $P(Z_3, x) = x(x-1)(x-2)$ and so

$(-1)^n[x(x-1)(x-2) - (x-1)^3] = x - 1$. Therefore,
$(-1)^n[P(Z_n, x) - (x-1)^n] = x - 1$ or $P(Z_n, x) = (-1)^n(x-1) + (x-1)^n$.

25(c). $(-1)^{n-1}(n-1)!$.

Section 3.5.

4(a). 11. **4(b)**. 9. **6**. G can be the disjoint union of a cycle and a path.

9. $n - k$. **11**. 16. **13**. 16.

15. Sketch: If the new edge e connected u and v, then the circuit we are looking for will involve e and the unique path in the tree between u and v.

16. There are too few edges to have a spanning tree; alternatively, the deleted edge was the only simple chain between its end vertices.

19. 2 if $n \geq 2$, since $2(2-1)^{n-1} > 0$ and $1(1-1)^{n-1} = 0$; 1 if $n = 1$.

21. Sketch: Since x^1 is the highest power of x that divides $x(x-1)^{n-1}$, then G has one component. If G had n edges, then it would have a cycle, and applying the Fundamental Reduction Theorem on an edge on the cycle would yield a contradiction. Similar contradictions are obtained if there were more than n edges, so G must have $n - 1$ edges.

26. *Hint:* The sum of the degrees is $4k + m$ and we have a tree.

29(b). Yes. **31(a)**. 6. **32(b)**. $\binom{8}{2}6! = 20,160$. **35(a)**. Yes.

Section 3.6.

2(a). a: 0; b, c: 1; d, e, f, g: 2; h, i, j, k: 3. **3(a)**. 3. **4(a)**. $\{d, e, h, i\}$.

7(a). The children of vertex 1 are 2 and 3 and the children of vertex 2 are 4 and 5.

11(a). 6. **13**. The least possible height is $\log_3(2n+1)$.

21. $[1 \cdot 2^0 + 2 \cdot 2^1 + 3 \cdot 2^2 + \cdots + (h+1)2^h]/n$, where h is the height.

26. 3 4 1 2, 3 1 4 2, 3 1 2 4, 1 3 2 4, 1 2 3 4, 1 2 3 4. **31**. 10. **43(b)**. 240.

Section 3.7.

1(a). For D_1 : $\begin{array}{c} \\ u \\ v \\ w \\ x \end{array} \begin{array}{c} \begin{array}{cccc} u & v & w & x \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$.

2(h). $\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} \{a,b\} & \{b,c\} & \{a,d\} & \{b,e\} & \{d,e\} \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$. **7**. For G_1 : $\begin{array}{c} \\ u \\ v \\ w \end{array} \begin{array}{c} \begin{array}{ccc} u & v & w \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{array}$.

9. $A(D) + A(D^c)$ has 0's on the diagonal and 1's off the diagonal.

13. For D_2 : $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$. For D_4 : $\begin{array}{c} \\ u \\ v \\ w \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{cccccc} u & v & w & x & y & z \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array}$.

For D_5 : $\begin{array}{c} \\ u \\ v \\ w \\ x \\ y \\ z \\ a \end{array} \begin{array}{c} \begin{array}{ccccccc} u & v & w & x & y & z & a \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$.

15. If D is strongly connected, then any vertex is reachable from any other, meaning that the corresponding matrix entry is 1. This argument works in reverse, too.

17. $\mathbb{A}^k$ corresponds to the number of chains (not necessarily simple) from i to j of length k.

19. For D_1 : $\begin{array}{c|cccc} & u & v & w & x \\ \hline u & 0 & 1 & 1 & 0 \\ v & 0 & 0 & 1 & 0 \\ w & 0 & 0 & 0 & 1 \\ x & 1 & 0 & 0 & 0 \end{array}$.

21(a). There is a path from i to j if and only if there is a path of length at most $n-1$.

24(a). j is in the strong component containing i if and only if $r_{ij} = 1$ and $r_{ji} = 1$.

26. $c(i)$ is the number of vertices v such that i is reachable from v.

29. The following is not a reachability matrix of any graph: $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$.

30. It gives the number of vertices that edges i and j have in common.

33. Yes: Take Z_4 as in Figure 3.22 and append x adjacent to a and b and y adjacent to b and c; repeat with Z_4 and x as above, but take y adjacent to c and d; relabel edges.

Section 3.8.

2. There is clearly no triangle in the graph. To see why there is no independent set of 5 vertices, assume that vertex 1 is in such an independent set. Then we must either choose both vertices 3 and 5, or both 10 and 12. Without loss of generality, we can choose vertices 3 and 5. Then the only two vertices left to choose are 7 and 12, which are adjacent to each other.

4. 7. **5(a)**. $\{a, c, e\}$. **5(f)**. $\{a, b, d\}$.

6(a). G has an independent set of size 3 since there will be a color class of size at least 3.

8(a). Let 4 "red edges from one vertex" be $\{a, b\}$, $\{a, c\}$, $\{a, d\}$, $\{a, e\}$. If any one of the edges $\{b, c\}$, $\{b, d\}$, $\{b, e\}$, $\{c, d\}$, $\{c, e\}$, $\{d, e\}$ is red then there will be 3 vertices all joined by red edges. If they are all blue then vertices b, c, d, e are all joined by blue edges.

8(b). Let v be the vertex with the six blue edges. Consider the six vertices that are the other endpoints of the six blue edges mentioned. Since $R(3, 3) = 6$, then these six vertices have either a red clique (and we're done) or a blue clique, which makes a clique of size 4 when we add v.

9(a). Yes. **11(c)**. 4.

Chapter 4

Section 4.1.

2. Consider a brother and a sister. **4**. Less than.

6(a). It is a complete graph with loops at every vertex.

8(b). Suppose that R^c is not symmetric. Then, for some $a, b \in X$, aR^cb but $\sim bR^ca$. Therefore, R is not symmetric since bRa but $\sim aRb$.

10. The relation $(X, R \cap S)$ is reflexive, symmetric, asymmetric, antisymmetric, and transitive.

14. Reflexive, symmetric, asymmetric, transitive, and negatively transitive.

16. Consider the binary relation (X, R) and suppose that aRa for some $a \in X$. By asymmetry, it must be the case that $\sim aRa$, which is a contradiction.

18(a). If we had a relation that was symmetric and asymmetric, and aRb, then we would have bRa and $\sim bRa$, which is a contradiction.

18(b). The relation $\{(x, x) \mid x \in X\}$ for any set X is both symmetric and antisymmetric.

19(a). $X = \{a, b, c\}$ and $R = \{(a, c)\}$. **20**. (a): Yes; (d): Yes; and (f): No.

22(a). On $\{a, b, c\} : \{(a, a), (b, b), (c, c), (a, b), (b, a), (b, c), (c, b)\}$.

22(b). On $\{a, b, c\} : \{(a, a), (b, b), (c, c), (a, b)\}$.

25. Reflexive and Symmetric hold.

26(e)i. aSb iff $\sim aRb$ & $\sim bRa$ iff $a \not> b$ & $b \not> a$ iff $a = b$. **27**. 2^{n^2}.

Section 4.2.

1(b). Yes. **2(b)**. Yes. **3(b)**. No. **4(b)**. No. **5(b)**. No. **6(b)**. No. **7(b)**. No.

8(b). No. **9(b)**. No. **10(b)**. No. **13(c)**. $K = \{(1, 3), (2, 3), (3, 4)\}$.

15. Sketch: Let xSy and ySz, where S is the lexicographic order. The only nontrivial case to check is when x, y, and z all have the same common prefix, which we will call a. Let $x = ax_1x'$, $y = ay_1y'$, $z = az_1z'$, where x_1, y_1, z_1 are letters and x', y', z' are words, possibly empty. In this case, we know $x_1 < y_1$ and $y_1 < z_1$ in the letter-ordering, so we must have $x_1 < z_1$, so xSz.

17. Sketch: If xb had a cover, then the cover would have to end with the letter z. However, if the cover, call it c, ended in the letter z, then the word cz would be between c and xb in the order.

19(a). $L_{S^{-1}} = [x_n, x_{n-1}, \ldots, x_1]$. **19(b)**. $L_S \cap L_{S^{-1}} = \emptyset$.

21. Transitive and complete, but not asymmetric: $X = \{a\}$ and $R = \{(a, a)\}$. Transitive and asymmetric, but not complete: $X = \{a, b, c\}$ and $R = \{(a, b), (a, c)\}$. Complete and asymmetric, but not transitive: $X = \{a, b, c\}$ and $R = \{(a, b), (b, c), (c, a)\}$.

25(a). Yes.

27. Reflexive: Trivial. Antisymmetric: if aSb and bSa, then the only case we need to check is if aRb and bRa. However, this can't happen since R is asymmetric. Transitive: The only nontrivial case to check is when aRb and bRc. Suppose $\sim aRc$. By asymmetry, $\sim cRb$. Therefore, by negative transitivity, $\sim aRb$, which is a contradiction.

30(a). No.

33(a). w_1 and m_2 are both better off by leaving their assigned partners and marrying each other.

Section 4.3.

1. No. **4(a)**. 3. **4(e)**. 4. **5(c)**. 2.

7. $\{\{1,2,3,4,5,6\},\{1,2,3,4,6,5\},\{2,1,3,4,5,6\},\{2,1,3,4,6,5\}\}$.

9(a). $[\hat{1},x,y,d,z,a,b,c,\hat{0}]$.

12. Strict partial order (c) of Figure 4.23 has dimension 2.

14. Sketch: There are $k!$ ways to build up the set s, one element at a time. Each way to build it up represents a unique partial chain. We can then add on each of the $n-k$ elements left, one at a time, in $(n-k)!$ ways. Again, each way to build it up represents a unique extension of the partial chain to a maximal chain.

17. $[a,x]$, $[b,y]$, $[c,z]$, $[d,w]$. **18**. $\{u\}$, $\{y,w\}$, $\{z,v\}$, $\{x\}$.

20. Let $\prec$ denote the order. Transitivity: Let $[a_1,b_1]\prec[a_2,b_2]$ and $[a_2,b_2]\prec[a_3,b_3]$. Then $b_2<a_1$ and $b_3<a_2$, but since $a_2\le b_2$ then $b_3<a_2\le b_2<a_1$, so we are done. Asymmetry: Let $[a_1,b_1]\prec[a_2,b_2]$. Then $b_2<a_1$ so if it were true that $[a_2,b_2]\prec[a_1,b_1]$ then it would mean that $b_1<a_2$, but then $a_1\le b_1<a_2\le b_2$, which is a contradiction.

21(a). If there was a loop, then it would mean that there was an arc from $[a,b]$ to itself, implying that $b<a$, which is a contradiction. If there is an arc from $[a_1,b_1]$ to $[a_2,b_2]$, and an arc from $[a_3,b_3]$ to $[a_4,b_4]$, then $a_1>b_2$ and $a_3>b_4$. If there is no arc from $[a_1,b_1]$ to $[a_4,b_4]$, then $a_1\le b_4$ so $b_2<a_1\le b_4<a_3$ and so there is an arc from $[a_3,b_3]$ to $[a_2,b_2]$.

Section 4.4.

1(a). No for both strict partial orders. **1(b)**. No for both strict partial orders.

2(b). (a): $\hat{0}$; (b): d; (c): $\hat{0}$. **3(a)**. (a): Not a lattice; (b): Not a lattice.

11.

a	b	c	$a\wedge(b\vee c)$	$(a\wedge b)\vee(a\wedge c)$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
1	0	0	0	0
0	1	1	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

12. (a): No.

16(c).

p	q	q'	$p\wedge q'$	$(p\wedge q')\to q$
F	F	T	F	T
F	T	F	F	T
T	F	T	T	F
T	T	F	F	T

17(b). $p=$ Pete loves Christine; $q=$ Christine loves Pete.

$p\wedge q=$ Pete and Christine love each other.

p	q	$p \wedge q$
F	F	F
F	T	F
T	F	F
T	T	T

18(b). Equivalent.

20(a).

x_1	x_2	$x_1' \wedge x_2$	$x_1 \vee x_2$	$(x_1' \wedge x_2) \vee (x_1 \vee x_2)$
1	1	0	1	1
1	0	0	1	1
0	1	1	1	1
0	0	0	0	0

Chapter 5

Section 5.1.

1(b). $\sum_{k=0}^{\infty} \frac{2^k x^k}{k!}$. **1(e)**. $1 + x + \frac{3x^2}{2!} + \sum_{k=3}^{\infty} \frac{x^k}{k!}$. **1(h)**. $\sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+2}}{(2k+1)!}$.

2(b). $\sum_{k=0}^{\infty} x^{k+2}$. **2(c)**. $\sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)!} x^{8k+4} = x^4 - \frac{x^{12}}{3!} + \frac{x^{20}}{5!} - \cdots$.

2(f). $\sum_{k=0}^{\infty} \frac{(k+1)(k+2)}{2} x^k$. **2(j)**. $\sum_{k=0}^{\infty} (-1)^k \frac{x^{10k+8}}{(2k+1)!}$. **3(a)**. $1 + x + x^2$.

3(d). $-x - x^2 + \frac{1}{1-x}$. **3(g)**. $\frac{1}{1-x} + x^2$. **3(j)**. $-1 - x + e^x$. **3(m)**. $\frac{3\ln(1+x)}{x}$.

3(p). $\frac{2\sin(x)}{x}$. **4(c)**. $(0,0,0,1,1,1,\ldots)$. **4(e)**. $(1,-8,8^2,-8^3,8^4,-8^5,\ldots)$.

4(h). $\left(6, \frac{2^1}{1!}, \frac{2^2}{2!}, \frac{2^3}{3!}, \ldots, \frac{2^k}{k!}, \ldots\right)$. **4(k)**. $(1,0,1,0,1,0,\ldots)$.

4(n). $\left(0,3,0,-\frac{3^3}{3!},0,\frac{3^5}{5!},0,-\frac{3^7}{7!},\ldots\right)$. **4(q)**. $\left(1,0,-\frac{3^2}{2!},0,\frac{3^4}{4!},0,-\frac{3^6}{6!},0,\frac{3^8}{8!},\ldots\right)$.

5(a). 1. **6(a)**. 0. **8(a)**. 7. **11**. 3^n. **13(b)**. $1 + 8x + 12x^2$.

14(b). $1 + 16x + 72x^2 + 96x^3 + 24x^4$.

Section 5.2.

1(a). $\left(0,0,1,-\frac{1}{2},\frac{1}{3},-\frac{1}{4},\ldots\right)$. **1(b)**. $\left(1,0,-\frac{1}{3!},0,\frac{1}{5!},0,\ldots\right)$.

1(e). $(9,8,5,6,5,5,5,\ldots)$. **1(h)**. $\left(0,1,0,\frac{1}{3!},0,\frac{1}{5!},0,\frac{1}{7!},\ldots\right)$. **2(a)**. $(0,1,2,3,\ldots)$.

3(b). $a_k = \sum_{i=0}^{k} i$. **3(e)**. $a_0 = 2; a_k = 4(k-1)$ for $k > 0$. **4(a)**. $a_k = 15(k+1)$.

5. 5. **6**. 38. **7(a)**. $(1,4,9,27,81,\ldots)$. **8(a)**. $A(x) + (11 - a_3)x^3$.

10(a). $\frac{x}{(1-x)^2} + \frac{3}{1-x} = \frac{3-2x}{(1-x)^2}$. **11(b)**. $\frac{2x}{(1-x)^3}$. **13**. $x^2 B(x) B(x)$. **15**. $\frac{1}{1-x} e^x$.

17. $R(x,B) = 1 + 6x + 11x^2 + 8x^3 + 2x^4$.

Section 5.3.

1(a). $(1+x+x^2)^2(1+x+x^2+x^3)^2$, coefficient of x^5.

1(c). $(x^3+x^4+x^5+x^6+x^7)(x^3+x^4+x^5+x^6+x^7)(x^2+x^3+x^4)$, coefficient of x^{11}.

1(e). $(1+x+x^2+x^3+x^4+x^5+x^6)^2(x^4+x^5+x^6+x^7)$, coefficient of x^{12}.

1(g). $(1+x+x^2+\cdots+x^40)^4$, coefficient of x^{40}.

1(i). $(1+x+x^2+x^3+x^4+\cdots+x^{12})^8$, coefficient of x^{12}.

1(k). $(x+x^2+x^3+x^4+x^5+x^6)^14$, coefficient of x^{30}.

1(m). $(1+x+x^2+x^3+\cdots)(1+x^5+x^{10}+x^{15}+\cdots)(1+x^{10}+x^{20}+x^{30}+\cdots)(1+x^{25}+x^{50}+x^{75}+\cdots)$, coefficient of x^{100}.

2. $(1+x)^3(1+x+x^2+\cdots)$, coefficient of x^5. **4**. $(1+n_1)(1+n_2)\cdots(1+n_p)$.

6(a). The generating function is $(x+x^2+\cdots+x^{10})^6$, and we want the coefficient of x^{10}.

8. $(1+x+x^2+x^3+x^4)^3(1+x+x^2+x^3)(1+x+\cdots+x^7)(1+x+\cdots+x^{12})$.

10(a). 1 and 4.

11. If $(c_k)=(a_k)*(b_k)$ with m and n as before, then

$$c_k=\begin{cases} k+1 & \text{if } k<\min(m,n) \\ \min(m,n)+1 & \text{otherwise.}\end{cases}$$

13(c). $(1+x)(1+x^2)(1+x^3)\cdots(1+x^k)$. **14(a)**. $p_o(7)=5$. **14(b)**. $p_o(8)=6$.

17(a). 56. **17(c)**. 34.

Section 5.4.

1(b). -10. **2(a)**. 35. **3**. $-\frac{1}{9}$.

5. There are seven ways: $aaaaaa, baaaaa, bbaaaa, bbbaaa, bbbbaa, bbbbba, bbbbbb$.

6. $\binom{16}{11}$. **8**. There are $\binom{12}{10}=66$ ways. **9**. There are $\binom{14}{10}=1001$ ways. **11**. $\binom{12}{7}$.

13(b). $\binom{p+k-1}{k}$. **18(b)**. $\binom{n-1}{r-1}$. **21**. 37.

Section 5.5.

1(b). e^{3x}. **1(c)**. $e^x-x-\frac{x^2}{2!}$. **2(a)**. $a_k=4k!$. **2(g)**. $a_k=2^k+5^k$.

6(a). $\left(1+x+\frac{x^2}{2!}+\frac{x^3}{3!}\right)^2(1+x)^2$, coefficient of $\frac{x^3}{3!}$.

6(d). $(1+x+x^2+\cdots+x^{2n})^3$, coefficient of x^{3n}.

6(g). The coefficient of x^{10} in $(1+x+x^2+\cdots+x^{10})^3$. Here we are assuming that the municipal bonds are indistinguishable. If we are assuming that they are distinguishable, then we are looking for the coefficient of $\frac{x^{10}}{10!}$ in $\left(1+x+\frac{x^2}{2!}+\cdots+\frac{x^{10}}{10!}\right)^3$.

6(j). The coefficient of $\frac{x^{200}}{200!}$ in $\left(\frac{x^{20}}{20!}+\frac{x^{40}}{40!}+\frac{x^{60}}{60!}+\frac{x^{80}}{80!}+\frac{x^{100}}{100!}\right)^4$.

9. The exponential generating function in this case is e^{px}, so the coefficient of x^k is what we seek, which is p^k.

12. $(e^x-1)^p$. **14**. $\frac{1}{2}[5^k-3^k]$.

16. $S(4,2)=7$ with distributions $(1,234),(2,134),(3,124),(4,123),(12,34),(13,24),(14,23)$. $T(4,2)=14$ with distributions $(1,234),(234,1),(2,134),(134,2),(3,124),(124,3),(4,123),(123,4),(12,34),(34,12),(13,24),(24,13),(14,23),(23,14)$.

19(b). $\left[\frac{3^0}{0!}+\frac{3^1}{1!}+\frac{3^2}{2!}+\frac{3^3}{3!}\right]3!$.

Section 5.6.

1(a). $G(x) = \frac{1}{3} + \frac{1}{3}x + \frac{1}{3}x^2, E = 1, V = \frac{2}{3}$.

3. If p_k is the probability that the k^{th} person studied has the disease and no one else before that person does, then $p_k = \left(\frac{49}{50}\right)^{k-1}(150)$. The generating function in this case is $P(x) = \frac{1}{49}\left[\frac{1}{1-\frac{49}{50}x} - 1\right]$. So the expected value is $P'(1) = 1$ and the variance is $P''(1) + P'(1) - [P'(1)]^2 = 2450$.

6. Since $\lambda = 2$, then from exercise 4 we know that $p_3 = \frac{e^{-2}2^3}{6}$ and $p_4 = \frac{e^{-2}2^4}{24}$ so we are looking for the sum, which is $\frac{2}{e^2}$.

8(b). $E = \frac{qm}{p}, V = \frac{q^2m}{p^2} + \frac{qm}{p}$.

Section 5.7.

1(d). Coleman: 0, $\frac{4}{8}$, $\frac{4}{8}$, $\frac{4}{8}$; Banzhaf: 0, $\frac{4}{12}$, $\frac{4}{12}$, $\frac{4}{12}$. **2(b).** [5; 4, 2, 1, 1].

4(c). $\frac{5}{12}, \frac{3}{12}, \frac{3}{12}, \frac{1}{12}$.

Chapter 6

Section 6.1.

1. $a_5 = 1535$, $a_6 = 6143$,. **3.** $S_4 = 10,368$. **4(b).** 528.

4(c). 00, 11, 12, 13, 21, 22, 23, 31, 32, 33. **7.** $f(n) = n + 14$.

9. $b_4 = F_5 = 8$: XOOX, XOXO, OXOX, XOOO, OXOO, OOXO, OOOX, OOOO.

11. 4123, 4312, 4321, 3142, 3412, 3421, 2143, 2341, 2413.

12. $a_4 = 72$, $b_4 = c_4 = 64$. **14(a).** $7! - D_7$. **14(b).** 1.

16(a). $f(n) = f(n-10) + f(n-18) + f(n-28)$. **16(b).** $f(66) = 13$.

16(c). 28/28/10, 28/10/28, 10/28/28, 18/18/10/10/10, 18/10/18/10/10, 18/10/10/18/10, 18/10/10/10/18, 10/18/18/10/10, 10/18/10/18/10, 10/18/10/10/18, 10/10/18/18/10, 10/10/18/10/18. 10/10/10/18/18.

19(a). $(D_4)^2$. **19(b).** $(4!)^2$. **21.** $b_n = 3b_{n-1} - 2b_{n-3}$. **24.** $f(n+1) = (2n+1)f(n)$.

26. Sketch of induction step:

$$\begin{aligned} a_{2n-1} &= 2a_{2n-3} + a_{2n-5} + \cdots + a_3 + a_1 + 1 \\ &= 2F_{2n-3} + F_{2n-5} + \cdots + F_3 + F_1 + F_0 \\ &= 2F_{2n-3} + F_{2n-5} + \cdots + F_3 + F_2 \\ &= \qquad \vdots \\ &= F_{2n-2} + F_{2n-3} \\ &= F_{2n-1}. \end{aligned}$$

28(a). Sketch: Show that the numbers we seek follow the same recurrence as F_n.

28(b). $F_{20} = 10946$.

29(a). Induction:

$$\begin{aligned} F_{3n+2} &= F_{3n+1} + F_{3n} \\ &= F_{3n} + F_{3n-1} + F_{3n-1} + F_{3n-2} \\ &= F_{3n-1} + F_{3n-2} + F_{3n-1} + F_{3n-2} \\ &= 2(F_{3n-1} + F_{3n-2}). \end{aligned}$$

30(a). Induction:

$$\begin{array}{rcl} F_nF_{n+1} - F_{n-1}F_{n+2} & = & (F_{n-1} + F_{n-2})F_{n+1} - F_{n-1}(F_{n+1} + F_n) \\ & = & F_{n-1}F_{n+1} + F_{n-2}F_{n+1} - F_{n-1}F_{n+1} - F_{n-1}F_n \\ & = & (-1)(F_{n-1}F_n - F_{n-2}F_{n+1} \\ & = & (-1)(-1)^n \\ & = & (-1)^{n+1}. \end{array}$$

31. $F_{n+2} - 1$. **32(a)**. $S(5,3) = 25$. **32(b)**. $S(6,3) = 90$.

32(c). Sketch: Let $\{1, 2, \ldots, n\}$ be our n-element set. The $S(n-1, t-1)$ term is for the case where the element n is in a set by itself, and the $tS(n-1,t)$ term is for the case when n is in a set with other elements.

34. $f(n+1) = f(n) + 2n, n \geq 1, f(1) = 2$.

35(a). $f(n) = 2^{F_n}$, where F_n is the Fibonacci number.

35(b). $f(n) = 1$ for all n. **38(a)**. $n!$. **40**. $C_3 = 4$.

Section 6.2.

1(a). Linear. **1(e)**. Not linear. **2(a)**. Not homogeneous. **2(e)**. Homogeneous.

3(a). Has constant coefficients. **3(f)**. Does not have constant coefficients.

5(a). $x^2 + 2x + 1 = 0$. **5(f)**. $x^2 - 2x - 3 = 0$. **6(a)**. -1, -1. **6(i)**. 1, 2, -2.

8(a). $a_n = 5 \cdot 6^n$. **8(b)**. $t_k = 2^{k-1}$. **10(a)**. A solution. **11(c)**. Not a solution.

12(a). $a_n = 2 \cdot (-1)^n - 4n \cdot (-1)^n$. **12(e)**. $h_n = \frac{7}{3} \cdot 3^n + \frac{5}{3} \cdot (-3)^n$.

14. $a_n = -\frac{2}{3}\left(-\frac{1}{2}\right)^n + \frac{2}{3}$. **17(c)**. Yes. **17(d)**. $a_n = (-\frac{i}{2})(i)^n + (\frac{i}{2})(-i)^n$.

19. The characteristic equation is $x^2 - 4x + 4$, which has 2 as a multiple root. If $F_n = \lambda_1 2^n + \lambda_2 2^n = (\lambda_1 + \lambda_2)2^n = K2^n$, then $1 = K$ but $3 = 2K$, which is a contradiction.

21. Yes, because it is a weighted sum of solutions.

22. 3 is a characteristic root of multiplicity 3.

24. $b_n = n2^n$ is a solution since $9(n-1)2^{n-1} - 24(n-2)2^{n-2} + 20(n-3)b^{n-3}$

$$\begin{array}{rl} = & 36(n-1)2^{n-3} - 48(n-2)2^{n-3} + 20(n-3)2^{n-3} \\ = & [36(n-1) - 48(n-2) + 20(n-3)]2^{n-3} \;\; = \;\; 8n2^{n-3} \;\; = \;\; n2^n. \end{array}$$

26(a). $a_n = \frac{8}{9}n(3^n) + (-\frac{5}{9})n^2(3^n)$. **27(a)**. $a_n = 5^n - \frac{3}{5}n5^n$.

Section 6.3.

2(a). $a_k = 3k + 1$. **3(a)**. $a_n = 2 \cdot (-1)^n - 4n \cdot (-1)^n$. **3(e)**. $h_n = \frac{7}{3} \cdot 3^n + \frac{5}{3} \cdot (-3)^n$.

4(a). $-\frac{1}{2} + \frac{3}{2} \cdot 3^k$. **4(e)**. $\left(\frac{1}{2}\right)^k - \left(\frac{1}{3}\right)^k$. **6(a)**. $A(x) = -\frac{2x}{x^2+x-2}$.

6(b). $A(x) = \frac{a_0(x-2)-2a_1x}{x^2+x-2}$. **9**. $G_n = \left(\frac{3+\sqrt{5}}{2}\right)\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{3-\sqrt{5}}{2}\right)\left(\frac{1-\sqrt{5}}{2}\right)^n$.

11. $A(x) = \frac{x(4x-5)}{(x-1)^2(2x-1)}$. **13**. $Y(x) = -\frac{-B+xy_0-y_0}{(x-1)(Ax-1)}$. **15**. $G(x) = \frac{2(x-2)}{(x-1)(x^2-2x+2)}$.

17. $I(x) = -\frac{v_0x^2}{(x-1)(\beta x^2-2\beta x+1)}$. **19**. $G(x) = \frac{x^3}{(1-2x)(x^2+1)}$. **22**. $b_n = 2b_{n-1} + 3b_{n-3}$.

24(a).

$$\begin{array}{rcl} A'(x) & = & \sum_{k=0}^{\infty} ka_kx^{k-1} \\ & = & \sum_{k=0}^{\infty} 2(a_{k-1} + a_{k-2})x^{k-1} \\ & = & 2\sum_{k=0}^{\infty} a_{k-1}x^{k-1} + 2\sum_{k=0}^{\infty} a_{k-2}x^{k-1} \\ & = & 2A(x) + 2xA(x) = 2(1+x)A(x). \end{array}$$

24(b). $A(x) = e^{x^2+2x+1}$. **27**. $1 + 7x + 13x^2 + 6x^3$.

Section 6.4.

2. $u_5 = 42$. **4**. $R_4 = 4$. **5**. $h_4 = 36$. **6(a)**. 6. **6(b)**. 8.

8(a). $C(x) - x^2 = A(x) + B(x) - x$. **8(d)**. $C(x) - x^2 = 4xC(x) + x[C(x)]^2$.

10. Using the plus sign produces terms that are negative. **12**. $u_n = \binom{2n}{n} - \binom{2n}{n-1}$.

15(a). $q_3 = 5$.

17. Sketch: Given a set of well-formed parentheses, inductively build up the SOR tree from the top down as follows: If one can split up the set into two smaller sets of well-formed parentheses, then those two separate sets make up the left- and right-subtrees. If it can't be split up, then peel off the outer parentheses, and the rest is the left-subtree.

19(i). $B(x) = x[U(x)]^2$. **20**. $A(x)B(x) = \frac{B(x)-b_0}{x}$. **22(c)**. $A(x)B(x) = \frac{x^3}{1-2x}$.

Chapter 7

Section 7.1.

1. 65. **3**. There are $-10,000$ female nonsmokers without cancer, which is absurd.

5(a). 860. **5(b)**. 1,814,400.

5(c). Let S_A be the number of sequences without an A, ditto for the rest. Then $S_A = 3^{10}$, ditto for the rest. So, the number we're looking for is $4^{10} - 4(3^{10}) + 6(2^{10}) - 4(1^{10}) + 0^{10} = 818520$.

8. 15. **9**. $38 + 45 + 28 + 25 - 22 - 23 - 10 - 1 - 21 - 14 + 11 + 8 + 6 + 6 - 5 = 71$.

11. 6233. **12**. 160. **14**. $\left(7^9 - \binom{7}{1}(7-1)^9 + \binom{7}{2}(7-2)^9 \mp \cdots + \binom{7}{6}(7-6)^9 - 0\right)/7^9$.

16. There are 9^{20} total possibilities, when we match up each light particle to whatever it hits. There are 8^{20} ways that the i^{th} particle isn't hit, etc., so the number we are looking for is

$$9^{20} - \binom{9}{1}8^{20} + \ldots = 4358654246117808000$$

and dividing by 9^{20} we get the probability of 0.358511.

18(d). $P(G, x) = x^4 - 4x^3 + 5x^2 - 2x$. **21(a)**. 19. **21(c)**. 15. **23**. $r_n(B) = n!$.

25(a). The object is counted twice. **26(b)**. 3 times. **28**. 28. **29**. 456. **31(a)**. 150.

34. $b_n = n! - \binom{n-1}{1}(n-1)! + \binom{n-1}{2}(n-2)! - \binom{n-1}{3}(n-3)! \pm \cdots + (-1)^{n-1}\binom{n-1}{n-1}1!$.

38. 13824.

Section 7.2.

1. 22%. **4**. 7. **7(a)**. 24. **9**. $\frac{1}{2}$. **11**. $\binom{n}{m}D_{n-m}$. **12(a)**. $\frac{6}{16}$. **14(a)**. 630.

16(a). Number of legitimate codewords $= \binom{n}{5}2^{n-5}$.

16(b). We can specify in $\binom{n}{5}$ ways precisely where the five 1's will go; then we have 2^{n-5} choices for the rest of the word.

18(a). 7/24. **18(b)**. 1/24. **20**. $\sum_{i=2}^{10}(-1)^i\binom{i}{i-2}\binom{10}{i}(10-i)^7$. **22**. 512. **24**. 3281.

32. 9. **36**. 11.

Chapter 8

Section 8.1.

1(c). Reflexivity, transitivity. **1(f)**. Symmetry, transitivity. **1(h)**. Symmetry.

2. Yes. **3(a)**. $\{a, b\}, \{c, d\}$.

5. Sketch: It is reflexive since by definition TST. It is symmetric since $T(x_1x_2) = U(x_2x_1)$ implies that $U(y_1, y_2) = T(y_2, y_1)$. For transitivity, if TSU and USV, then $T(x_1x_2) = U(x_2x_1) = V(x_1x_2)$, so $T = V$.

7. $\{bb\}, \{rr\}, \{pp\}, \{bp, pb\}, \{br, rb\}, \{pr, rp\}$. **9**. See Figure 8.10 in the text.

13(a). There are 3 others. **15**. There are six such colorings. **16**. 3. **18(a)**. 8.

18(c). 352.

20(a). Any subset of ordered pairs of elements of A is a binary relation, so there are $2^{\binom{n}{2}}$ binary relations.

20(b). A reflexive relation must contain all the pairs (i, i), so there are $\binom{n}{2} - n$ pairs left to choose, so there are $2^{\binom{n}{2}-n}$ different reflexive relations.

20(c). There are $2^{\frac{\binom{n}{2}-n}{2}-n}$ different symmetric relations.

Section 8.2.

1(b). $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 2 & 4 & 1 & 3 \end{pmatrix}$. **2(a)**. $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{pmatrix}$.

3. If $e = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$ and $x = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$ then e is the identity, and the other group conditions are readily verified by hand.

5(a). All.

5(c). We have closure by definition. Associativity follows by checking all cases. 0 is the identity element. The element 1 has no inverse, however.

5(f). **G1, G2**. **6**. See exercise 3. **8(a)**. No. **9**. $C(1) = \{1, 7\}$.

10(a). $\{1, 5\}, \{2, 4\}, \{3\}$.

11(a). Reflection in a diagonal from upper right to lower left.

11(c). $\pi_1 \circ \pi_2$ is equal to the reflection along the diagonal that points up and to the right.

12(a). $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix}$. **12(c)**. The symmetry is $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 3 & 1 \end{pmatrix}$.

16. $\pi_1 \circ \pi_2$ is a rotation by 90 degrees.

19(a). $\pi_1 \circ \pi_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{pmatrix}$. $\pi_2 \circ \pi_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}$.

19(b). For one example, π_1 is the rotation by 90 degrees, and π_2 is the reflection across the vertical line.

21. No. For example, associativity may not hold. Let $p = 6$, then $(2 \circ 3) \circ 5 \neq 2 \circ (3 \circ 5)$.

23(a). $\left\{ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix} \right\}$.

25(a). $|\mathrm{Aut}(K_{m,n})| = m!n!$. Sketch: We can permute around any of the vertices in the m-set and the n-set, but we cannot swap between them.

25(b). $|\mathrm{Aut}(K_{n,n})| = 2(n!)(n!)$. Sketch: We can permute around any of the vertices in the two n-sets, and we can also completely swap the positions of the two n-sets, which adds the extra factor of 2.

Section 8.3.

3. If vertex 1 is the vertex of degree 3, then $|C(1)| = 1$ as all automorphisms fix vertex 1. $|C(2)| = |C(3)| = |C(4)|$ since there are automorphisms that send vertex 2 to vertices 3 and 4, vertex 3 to vertices 2 and 4, and vertex 4 to vertices 2 and 3. Hence,

$$\frac{1}{|C(1)|} + \frac{1}{|C(2)|} + \frac{1}{|C(3)|} + \frac{1}{|C(4)|} = 1 + \frac{1}{3} = \frac{1}{3} + \frac{1}{3} = 2.$$

5(a). $\frac{1}{2}(5+1) = 3 : \{1,5\}, \{2,4\}, \{3\}$.

6(a). **(a)** $\mathrm{St}(1) = \left\{ \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix} \right\}$, **(b)** $C(1) = \{1,5\}$.

9. $|C(i)| = 5$ for all i as we can continually apply the permutation $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix}$ to each vertex and we will hit all vertices in the graph. Therefore,

$$\frac{1}{|C(1)|} + \frac{1}{|C(2)|} + \frac{1}{|C(3)|} + \frac{1}{|C(4)|} + \frac{1}{|C(5)|} = \frac{1}{5} + \frac{1}{5} + \frac{1}{5} + \frac{1}{5} + \frac{1}{5} = 1.$$

12. No, G is not a group of permutations. **13(b).** 12.

Section 8.4.

2. 3^8. **4(a).** black. **5(a).** r. **6(a).** No. **6(g).** 8. **7(a).** No. **7(f).** 4. **10(e).** 6.

12(b). There are $\frac{1}{6}\left[4^4 + 4^3 + 4^3 + 4^3 + 4^2 + 4^2\right] = 80$ colorings.

14. There are $\frac{1}{4}\left[2^4 + 2^3 + 2^3 + 2^2\right] = 9$ colorings.

16(a). $[(\pi_2^*)^* T_3](x_1x_2) = T_3(x_2x_1)$ so,

$$\begin{array}{lclcl} [(\pi_2^*)^*T_3](00) & = & T_3(00) & = & 0 \\ [(\pi_2^*)^*T_3](10) & = & T_3(01) & = & 0 \end{array} \qquad \begin{array}{lclcl} [(\pi_2^*)^*T_3](01) & = & T_3(10) & = & 1 \\ [(\pi_2^*)^*T_3](11) & = & T_3(11) & = & 0 \end{array}$$

which is the definition of T_5.

18. There are $\frac{1}{2}\left[2^{16} + 2^1\right] = 32769$ different colorings.

21(c). if π_i is $\begin{pmatrix} \{1,2\} & \{1,3\} & \{2,3\} \\ \{1,2\} & \{2,3\} & \{1,3\} \end{pmatrix}$, then $\mathrm{Inv}(\pi_i^*)$ is 4.

Section 8.5.

1(a). (2)(5)(17)(364). **1(b).** (2)(1534). **3(a).** $\mathrm{cyc}(\pi)$ is 5 and 3, respectively.

4(a). x_1^5 and $x_1x_2^2$, respectively.

5(a). $\frac{1}{2}(x_1^5 + x_1 x_2^2)$. **9(a)**. $\frac{1}{2}\left[2^4 + 2^2\right] = 10$. **9(c)**. $\frac{1}{2}(3^1 + 3^2) = 6$.

11(a). The permutation group is $\{(1)(2)(3)(4), (1342)(14)(23), (1243), (123)(4), (12)(34), (13)(24), (14)(2)(3), (1)(23)(4)\}$ and so the number of colorings is $\frac{1}{8}\left[2^4 + 2^1 + 2^2 + 2^1 + 2^2 + 2^2 + 2^3 + 2^3\right] = 6$.

12(a). $\frac{1}{2}\left[2^3 + 2^2\right] = 6$.

14. $\pi_1 = (1)(2)(3)(4)$ and all colorings are invariant; there are m^4 such colorings. $\pi_2 = (1243)$ and $\pi_4 = (1342)$ and the only invariant colorings are the ones where all blocks are colored the same; there are m^1 such colorings. $\pi_3 = (14)(23)$ and the only invariant colorings are the ones where diagonally-opposite blocks are colored the same; there are m^2 such colorings.

17(a). There are at most $\binom{3}{2} = 3$ edges in the graph, so we must look over all permutations of three elements: $(1)(2)(3), (1)(32), (21)(3), (2)(31), (312), (321)$. Therefore, there are $\frac{1}{6}\left(2^3 + 2^2 + 2^2 + 2^2 + 2^1 + 2^1\right) = 4$ nonisomorphic graphs on 3 vertices.

19(a). For the switching functions on n variables, the total number of potential inputs is 2^n. The underlying group consists of the identity permutation and the permutation that switches each bit, which is a permutation that consists of 2^{n-1} cycles, each of length 2. Therefore, the number of switching functions on n variables is $\frac{1}{2}\left(2^{2^n} + 2^{2^{n-1}}\right)$.

22(b). x_2^4.

25. Sketch: A cycle $(a_1 a_2 \cdots a_k)$ can be decomposed into the following transpositions: $(a_1 a_2), (a_2 a_3), \cdots, (a_{k-1} a_k)$.

26. $(16)(15)(14)(13)(12)$.

28. Hint: Cycles can be written in more than one way, so use this fact as well as exercise 25.

31(a). $D_5 = (2-1)(3-2)(3-1)(4-3)(4-2)(4-1)(5-4)(5-3)(5-2)(5-1)$.

31(b). $\pi D_5 = (3-4)(5-3)(5-4)(2-5)(2-3)(2-4)(1-2)(1-5)(1-3)(1-4)$.

Section 8.6.

1. The second has weight 192. **2(a)**. g of part (a) has weight x^2y^2.

4. $2a^3b + 2ab^3 + a^2b^2$.

6. In cycle form, the permutations are $(1)(2)(3)(4)$, $(123)(4)$, and $(132)(4)$. Therefore, the cycle index is $P_G(x_1, x_2, x_3, x_4) = \frac{1}{3}(x_1^4 + 2x_1x_3)$ and the pattern inventory is

$$P_G(\alpha+\beta, \alpha^2+\beta^2, \alpha^3+\beta^3, \alpha^4+\beta^4) = \alpha^4 + 2\alpha^3\beta + 2\alpha^2\beta^2 + 2\alpha\beta^3 + \beta^4.$$

8. 3280.

10. In cycle form, the permutations are $(1)(2)(3)(4)$ and $(14)(23)$. Therefore, the cycle index is $P_G(x_1, x_2, x_3, x_4) = \frac{1}{2}(x_1^4 + x_2^2)$ and the pattern inventory is $P_G(b+r+p, b^2+r^2+p^2, b^3+r^3+p^3, b^4+r^4+p^4) = b^4 + 2b^3p + 4b^2p^2 + 2bp^3 + p^4 + 2b^3r + 6b^2pr + 6bp^2r + 2p^3r + 4b^2r^2 + 6bpr^2 + 4p^2r^2 + 2br^3 + 2pr^3 + r^4$.

Removing all monomials without p, we are left with
$2b^3p + 4b^2p^2 + 2bp^3 + p^4 + 6b^2pr + 6bp^2r + 2p^3r + 6bpr^2 + 4p^2r^2 + 2pr^3$, and plugging in $b = r = p = 1$ we obtain 35 possible necklaces.

12(b). The automorphism group of Z_4 consists of the permutations $\pi_1 = (1)(2)(3)(4)$, $\pi_2 = (1)(3)(24)$, $\pi_3 = (13)(2)(4)$, and $\pi_4 = (13)(24)$. Therefore, the cycle index is $P_G(x_1, x_2, x_3, x_4) = \frac{1}{4}(x_1^4 + 2x_1^2x_2 + x_2^2)$ and the pattern inventory is $P_G(G+B+W, G^2+B^2+W^2, G^3+B^3+W^3, G^4+B^4+W^4) = B^4 + 2B^3G + 3B^2G^2 + 2BG^3 + G^4 + 2B^3W + 4B^2GW + 4BG^2W + 2G^3W + 3B^2W^2 + 4BGW^2 + 3G^2W^2 + 2BW^3 + 2GW^3 + W^4$. Considering only the terms that have exactly one G, we see that there are 12 distinct colorings.

14. The underlying permutations are $(1)(2)(3)(4)$ and $(1)(4)(23)$, where 1 corresponds to 00, 2 corresponds to 01, 3 corresponds to 10, and 4 corresponds to 11. The cycle index is $P_G(x_1, x_2, x_3, x_4) = \frac{1}{2}(x_1^4 + x_1^2x_2)$ and if we set the weight of 0 to be a and the weight of 1 to be b, then the pattern inventory is $P_G(a+b, a^2+b^2, a^3+b^3, a^4+b^4) = a^4 + 3a^3b + 4a^2b^2 + 3ab^3 + b^4$. The part of this expression that has at least one b is $3a^3b + 4a^2b^2 + 3ab^3 + b^4$, and plugging in $a = b = 1$ yields 11 such switching functions.

16. 11. **19(e)**. $1 + x$.

20(c). if the entries of (x_1, x_2, x_3) give the colors of a, b, c, respectively, then equivalence classes are $\{(0,0,0)\}$, $\{(0,0,1)\}$, $\{(0,1,0),(1,0,0)\}$, $\{(0,1,1),(1,0,1)\}$, $\{(1,1,0)\}$, $\{(1,1,1)\}$.

Chapter 9

Section 9.1.

1.

1	4	3	2
1	4	3	2
1	4	3	2
1	4	3	2

. **2(b)**. It must be a multiple of 6.

3(b). $\{1,2,3\}, \{1,4,5\}, \{1,6,7\}, \{2,4,7\}, \{2,5,6\}, \{3,4,6\}, \{3,5,7\}$.

5(a). Yes: $\{1,2,3,4\}, \{5,6,7,8\}$. **5(b)**. Yes. (Hint: Use repetition in the blocks.).

6(a). 21.

Section 9.2.

1(a). No. **1(c)**. Yes. **3(a)**. No. **4(a)**. Yes. **4(c)**. Yes. **5(a)**. Cannot be sure.

5(c). Cannot be sure. **6**. Use Theorem 9.3 and that $539 = 7^2 \cdot 11^1$. **8**. Yes. **9**. No.

11(a). No. **11(b)**. It is at most 7. **12(a)**. No.

12(b). There cannot be more than six of them.

15. $\begin{pmatrix} 1&2&3&4&5&6&7&8&9\\ 9&1&2&3&4&5&6&7&8\\ 8&9&1&2&3&4&5&6&7\\ 7&8&9&1&2&3&4&5&6\\ 6&7&8&9&1&2&3&4&5\\ 5&6&7&8&9&1&2&3&4\\ 4&5&6&7&8&9&1&2&3\\ 3&4&5&6&7&8&9&1&2\\ 2&3&4&5&6&7&8&9&1 \end{pmatrix}$, $\begin{pmatrix} 2&3&4&5&6&7&8&9&1\\ 3&4&5&6&7&8&9&1&2\\ 4&5&6&7&8&9&1&2&3\\ 5&6&7&8&9&1&2&3&4\\ 6&7&8&9&1&2&3&4&5\\ 7&8&9&1&2&3&4&5&6\\ 8&9&1&2&3&4&5&6&7\\ 9&1&2&3&4&5&6&7&8\\ 1&2&3&4&5&6&7&8&9 \end{pmatrix}$.

16. For $n = 4$ there are orthogonal pairs of Latin squares, but there is no orthogonal pair including this one: $\begin{pmatrix} 1&2&3&4\\ 2&4&1&3\\ 3&1&4&2\\ 4&3&2&1 \end{pmatrix}$.

17. $A^{(1)} = \boxed{\begin{matrix} 3&1&2\\ 1&2&3\\ 2&3&1 \end{matrix}}$, $A^{(2)} = \boxed{\begin{matrix} 2&3&1\\ 1&2&3\\ 3&1&2 \end{matrix}}$.

19. $\mathrm{OA}(2,4)^T = \begin{bmatrix} 1&1&1&1&2&2&2&2&3&3&3&3&4&4&4&4\\ 1&2&3&4&1&2&3&4&1&2&3&4&1&2&3&4\\ 1&2&3&4&2&1&4&3&3&4&1&2&4&3&2&1\\ 4&1&2&3&3&2&1&4&1&4&3&2&2&3&4&1 \end{bmatrix}$.

(The transpose is given here to conserve space.)

22(a). Yes. **22(c)**. Yes.

25(a). They need at least 101 different possible key values, and hence an $OA(11, 101)$ array, or any $OA(11, n)$ with $n > 101$.

25(b). Use Theorems 9.2 (or 9.3) and 9.7.

Section 9.3.

1(a). $b = 2$. **1(b)**. $b = 0$. **1(e)**. $b = 57$. **2(b)**. $a + b = 9, a \times b = 8$.
2(e). $a + b = 0, a \times b = 1$.

4(b). Hint: modulo n, $c_i \neq c_j$ if and only if $c_i + 1 \neq c_j + 1$.

5(c). No. **6(a)**. 2^{32}. **6(b)**. February 7, 2106. **7(b)**. 258. **8(a)**. $s = 11$, $t = 131$.

9(a). The numerator is divisible by p, whereas the denominator is not.

9(b). $(x+1)^p = x^p + \sum_{i=1}^{p-1} \binom{p}{i} x^{p-i} + 1 \equiv x^p + 1 (\mathrm{mod}\ p)$. Since $0^p = 0$, using the previous equation, $x^p \equiv x$ follows for all positive integers x.

10(a).

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

. **11(b)**. 2. **13(a)**. 3; 7.

15(a). Not a field, there is no additive inverse.

15(b). Not a field, ∞ has no multiplicative inverse, even though it is not the additive identity.

18(a). $\begin{pmatrix} 2 & 3 & 4 & 0 & 1 \\ 3 & 4 & 0 & 1 & 2 \\ 4 & 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 0 \end{pmatrix}, \begin{pmatrix} 3 & 4 & 0 & 1 & 2 \\ 0 & 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 0 & 1 \\ 4 & 0 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 & 0 \end{pmatrix}, \begin{pmatrix} 4 & 0 & 1 & 2 & 3 \\ 2 & 3 & 4 & 0 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 3 & 4 & 0 & 1 & 2 \\ 1 & 2 & 3 & 4 & 0 \end{pmatrix},$
$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 4 & 0 & 1 & 2 & 3 \\ 3 & 4 & 0 & 1 & 2 \\ 2 & 3 & 4 & 0 & 1 \\ 1 & 2 & 3 & 4 & 0 \end{pmatrix}.$

Section 9.4.

1(a). Not a BIBD.

1(b). Not a BIBD; some pairs are contained in more blocks than others.

1(e). This is a BIBD with $b = 14, v = 7, r = 6, k = 3, \lambda = 2$.

2(a). $b = 50, r = 25$. **3(a)**. $r(k-1) \neq \lambda(v-1)$. **5**. No. **7**. No: $b \geq v$ fails.

9(a). $\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} \{1,2\} & \{1,3\} & \{2,4\} & \{1,2,3\} & \{2,3,4\} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix} \end{array}$. **10(a)**. $\begin{bmatrix} 3 & 2 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 2 & 3 & 2 \\ 2 & 2 & 2 & 3 \end{bmatrix}.$

12(a). Three blocks are missing. **12(b)**. Add $\{a, f, h\}$, $\{b, e, g\}$, and $\{c, d, i\}$.

15. Interchanging all occurrences of a pair of numbers in each block yields new resolvable designs.

18. 737. **20(a)**. $b' = b, v' = v, r' = b - r, k' = v - k$.

20(d). Take the complementary design of the $(12, 9, 4, 3, 1)$-design of Table 9.2.

21. $b = 26, v = 13, r = 20, k = 10, \lambda = 15$. **24**. $\{2, 4, 6\}, \{3, 4, 5\}, \{3, 6, 7\}$.

26(a). This is a $(4m-1, 2m-1, m-1)$-design, $m = 2^3$.

26(c). Use Theorem 9.18.

30. Use Theorem 9.16. To show that the equation $x^2 + z^2 = 6y^2$ has no nonzero solution, examine both sides of the equation modulo 3.

33. Use Theorem 9.18, and take two copies of each block of an $(m^2 + m + 1, m + 1, 1)$-design.

35(a). No: $k - \lambda$ is not a square. **35(c)**. No. **38**. Use Theorems 9.17 and 9.20.

40. Take an existing design and repeat blocks.

41. Take two copies of each block of a (31, 15, 7)-design. **42(b)**. No.

42(c). Verify that $f_G(P)$ is different for each singleton P. **44(b)**. $P = \{1, 5, 7\}$.

46. In the proof of Fisher's Inequality it is shown that $\det \mathbf{A}A^T \neq 0$. It follows that $\det \mathbf{A} \neq 0$.

52. If such a system exists, it must have 285384 blocks.

Section 9.5.

1(a). (P_1) and (P_3). **1(d)**. (P_3). **1(e)**. All three axioms hold.

2(a). There are 9 distinct points, no 3 of which lie on the same line.

2(b). There is a line that passes through each point. **3**. 13. **4**. 21. **6**. 8. **8(a)**. No.

8(d). Yes. **9(a)**. $v = 31, k = 6, \lambda = 1$. **10(a)**. Yes (Corollary 9.27.1).

11. Use Theorem 9.28. **14(a)**. Yes (but cannot be sure).

15. By Corollary 9.29.2 it is enough to show that there is no finite projective plane of order 378; this follows from Theorem 9.28.

16(a). 1. **17(a)**. 1. **18**. P_1 and P_3 are satisfied, but P_2 is not.

21. Choose an arbitrary line of the projective plane and delete it together with all of its points from the plane. It is immediate that P_1 still holds. To see why P_3 holds, argue that there is a line that avoids all the four points of P_3. (Let a, b, c, d be the four points of P_3, and consider the line passing through the intersection of the lines containing ab and cd, and the intersection of the lines containing ac and bd.) The "parallel postulate" follows from P_2 and P_1.

22(b). If we take $U_3 = \{1, 3, 5, 7\}$, $V_2 = \{2, 3, 4, 13\}$, $W_{11} = \{3, 6, 8, 11\}$, $W_{21} = \{3, 9, 10, 12\}$, then the point 3 is associated with (3, 2) and (3, 2, 1, 1).

22(c). $a_{32}^{(1)} = 1, a_{32}^{(2)} = 1$. **23(a)**. (2, 3) is associated with (2, 3, 1, 2).

23(b). $W_{12} = \{(1, 2), (2, 1), (3, 3)\}$.

23(e). W_{12} is now $\{(1, 2), (2, 1), (3, 3), w_1\}$, the finite points are all (i, j) with $1 \le i, j \le 3$, and the infinite points are u, v, w_1, w_2.

23(f). $m^2 + m + 1$ lines, including the line at infinity.

Chapter 10

Section 10.1.

No exercises.

Section 10.2.

1(a). (49, 96). **1(c)**. 44, 84, 42, 56, 42, 32, 33, 48.

1(e). 37, 56, 32, 20, 59, 80, 66, 92, 41, 52, 33, 48, 30, 36, 43, 56. **2(a)**. 1, 1, 2, 1.

2(c). 0, 1, 1, 0. **3(a)**. 1, 1, 1, 1, 1, 1. **4(a)**. 1, 1, 1, 1, 3, 3, 3. **5(a)**. (8, 14, 14).

5(c). (81, 142, 161).

5(e). (43, 81, 91, 52, 84, 100, 36, 59, 72, 58, 97, 134, 18, 31, 42, 39, 73, 89, 43, 78, 73).

6(a). (28, 54). **6(c)**. 29, 54, 70, 112, 112, 172, 48, 78.

6(e). 50, 82, 93, 142, 96, 154, 103, 166, 73, 116, 48, 78, 57, 90, 74, 118.

7(a). (9, 8, 8). **7(c)**. (82, 81, 80).

7(e). (52, 43, 20, 40, 52, 80, 28, 36, 52, 40, 58, 76, 14, 18, 20, 43, 39, 20, 55, 43, 32).

8(a). AB. **8(c)**. Error. **8(e)**. Error. **9(a)**. ABC. **9(c)**. BAT. **10(a)**. AA.

10(c). Error. **10(e)**. Error. **11(a)**. Error. **11(c)**. Error. **12(a). (i)** 011. **12(b). (i)** 1001. **12(c). (i)** 10000. **13.** 101, 000, 011, 011.

15. Using notation from Example 10.3, let $\mathbf{M}= (I_k \quad I_k \quad \cdots \quad I_k)$ where there are p copies of I_k which is the $k \times k$ identity matrix.

Section 10.3.

1(a). 6. **1(c)**. 2. **2(a)**. $111111\underline{0}$. **2(c)**. $001001001\underline{1}$. **3(a)**. detect 2, correct 1. **4(a)**. median: 00000000, mean: 00000000. **5(a)**. 0. **5(c)**. 1.

6. For example, let $C = \{001111, 110110\}$ and x_i's: $001000, 110110$. **7(a). (i)** 4. **7(b)**. $d-1$. **7(c)**. $\lceil (d/2)-1 \rceil$. **9**. $\frac{2^{11}}{\binom{11}{0}+\binom{11}{1}+\binom{11}{2}} = 2^{11}/67$. **11(a)**. $\binom{10}{0}(.1)^0(.9)^{10}$.

11(d). $1 - \left(\binom{10}{0}(.1)^0(.9)^{10}\right) - \left(\binom{10}{1}(.1)^1(.9)^9\right) - \left(\binom{10}{2}(.1)^2(.9)^8\right)$.

12(a). $\binom{6}{0}(.001)^0(.999)^6$. **12(c)**. $\binom{6}{2}(.001)^2(.999)^4$. **13(a)**. $\binom{n}{t}(p)^t(1-p)^{n-t}$.

14. Probability of 0 errors is .729, of 0 or 1 errors is .972, so $d = 3$.

16(a). Two errors can always be detected, but three may not be.

16(b). One error can be corrected.

17. $|C| \le \frac{q^n}{\binom{n}{0}(q-1)^0+\binom{n}{1}(q-1)^1+\cdots+\binom{n}{t}(q-1)^t}$, where $t = \lceil (d/2)-1 \rceil$.

18(c). One error can be detected, no errors can be corrected. **19(b)**. 1 error.

21. Show that if such a set of codewords existed, we could construct two orthogonal Latin squares of order 6, which is impossible. (See Section 9.2.3).

22(b). 2 errors (use horizontal completeness).

24. The probability that a single digit can be decoded correctly is $q = (0.99)^5 + 5(0.99)^4(0.01)^1 + 10(0.99)^3(0.01)^2$. The probability that the entire message can be decoded correctly is $q^1000 \approx 0.9902$.

28. 4096. **30(a)**. $s_\chi(00) = \lambda_0 + \lambda_1$, $s_\chi(01) = 2\lambda_0$, $s_\chi(10) = 2\lambda_0$, $s_\chi(11) = \lambda_0 + \lambda_1$.

30(c). $\sigma_\chi(00) = \sigma_\chi(11) = 1$, $\sigma_\chi(01) = \sigma_\chi(10) = 0$.

32. Use that $s_\chi(\omega) = A\sigma_\chi(\omega) + pB$, $A < 0$, and pB is a constant.

33. Use that $s_\chi(\omega) = A\tau_\chi(\omega) + pB$, $A < 0$, and pB is a constant.

Section 10.4.

1(a). 1101. **1(c)**. 0110. **2(a)**. 111111. **3(a)**. 1111111.

4(b). The $3 \to 6$ double repetition code. **5**. $\mathrm{M} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$. **6(a)**. 2.

6(c). 2. **7(a)**. $\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$. **7(b)**. $\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$.

8(a). $\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$. **9(a)**. $x_3 = x_1 + x_2, x_4 = x_1$.

9(b). $x_1 = x_4, x_2 = x_5, x_3 = x_6$. **9(e)**. $x_1 = x_4, x_3 = x_5, x_2 = x_4 + x_5$.

11(a). No error. **12(a)**. An error was made in transmission. **14(b)**. 111000.

15(a). The intended codeword was most likely 01101.

16. The codewords are 000000, 001001, 010011, 100111, 011010, 101110, 110100, 111101. It does not correct all single errors (the minimum Hamming distance is only 2).

17(a). 1001100. **21**. 111 if $p = 2$.

23. Let x and y be distinct, nonzero vectors, and consider x, y, and $x + y$.

26. Linear codes are closed with respect to componentwise modulo-two addition by definition. Associativity follows from the associativity of binary addition. The zero vector is the identity element, and each vector is its own inverse.

27. use $|C| = 2^{2^p-1-p}$, $t = 1$. **28(a). (i)** 1401. **28(b). (ii)** 102102.

29(b). $d(\mathbf{x}, \mathbf{y}) = wt(\mathbf{x} - \mathbf{y})$.

31. Let w be the weight of the codeword with smallest weight. Then $\min d(\mathbf{x}, \mathbf{y}) \leq \min d(\mathbf{x}, \mathbf{0}) = \min wt(\mathbf{x}) = w$, so the minimum weight is an upper bound on the minimum distance. On the other hand, $d(\mathbf{x}, \mathbf{y}) \geq wt(x - y)$, which in turn implies $\min d(\mathbf{x}, \mathbf{y}) \geq w$.

Section 10.5.

1. The three missing blocks are $\{1,5,7\}, \{4,5,6\}, \{2,6,7\}$. **2(e)**. $2(r - \lambda) - 1$.

4(a). No. **4(b)**. No. **5**. Yes.

6. Use the statement of Exercise 5, and Theorem 10.11. **9(b)**. 7. **9(c)**. 3.

14. The sketch of the proof: It is trivial that the inner product of any row and itself is $n = 4m$. The inner product of two distinct rows equals the number of columns that have a 1 in both rows + the number of columns that have a -1 in both rows $-$ the number of columns that have a 1 in one row and a -1 in the other. This translates, in the block design, to the number of blocks that contain both varieties + the number of blocks that contain neither $-$ the number of blocks that contain only one of them, which is $(m - 1) + 2m - (3m - 1) = 0$.

16(a). 0 if $i = j$, and $2m$ if $i \neq j$. **16(b)**. $4m$ if $i = j$, and $2m$ if $i \neq j$. **18(a)**. u.

22(a). Adding $n - m$ zeros to the codewords of the (m, d)-code yields an (n, d)-code.

22(b). No. For example, $A(n, n-1) = A(n-1, n-1) = 2$ for n sufficiently large. (See Exercise 21).

25. Any two distinct codewords have inner product zero, so if the code was linear, then for any $\mathbf{x} \neq \mathbf{y} \neq \mathbf{0}$, $0 = \mathbf{x}^T(\mathbf{x} + \mathbf{y}) = \mathbf{x}^T\mathbf{x} + \mathbf{x}^T\mathbf{y} = n$, a contradiction.

26. $\mathbf{M} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$.

27(a). The generated code is the Hamming code,

$$C = \{0000000, 0011101, 0101011, 1000111, 0110110, 1011010, 1101100, 1110001\}.$$

27(b). The 7-Hadamard code, with the 0's and 1's interchanged is formed by the rows of the matrix

$$\begin{pmatrix} 0&0&0&0&0&0&0\\ 1&0&1&0&1&0&1\\ 0&1&1&0&0&1&1\\ 1&1&0&0&1&1&0\\ 0&0&0&1&1&1&1\\ 1&0&1&1&0&1&0\\ 0&1&1&1&1&0&0\\ 1&1&0&1&0&0&1 \end{pmatrix}.$$

29(a). The distance between codewords of equal weight is even.

Chapter 11

Section 11.1.

1(a). One example: Assign labels $a = 1, e = 2, j = 3, k = 4, f = 5, g = 6, h = 7, l = 8, m = 9, i = 10, d = 11, c = 12, b = 13$ and mark the edges $\{a, e\}, \{e, j\}, \{j, k\}, \{k, f\}, \{f, g\}, \{g, h\}, \{h, l\}, \{l, m\}, \{m, i\}, \{i, d\}, \{d, c\}, \{c, b\}$.

2. Not connected. **3(a)**. Connected.

4(a). One example: Use the marked edges in answer to 1(a).

5. No. Hint: Consider the depth-first search spanning trees of a complete graph.

7. Hint: Use the result of Exercise 6. **8(a)**. No.

9(a). The vertices are visited in the order: a, b, e, c, f, j, d, h, g, k, i, l, m.

9(f). The vertices are visited in the order: a, b, c, d, e, f, g, h. **10**. Yes.

Section 11.2.

1(a). None. **1(c)**. ab, de, and gh.

2(a). Orientation based on answer to Exercise 1(a), Section 11.1 orients 1 to 2 to 3 to ... to 13 and all other edges from higher number to lower number.

2(b). The arcs are oriented as ab, bc, ce, ei, ih, hg, gf, fd (marked edges), and ca, hc, bg, db (unmarked edges).

6. 2^e. **7**. $V = \{x, a, b, c\}, E = \{\{x, a\}, \{x.b\}, \{x, c\}\}$. **8(a)**. None. **8(c)**. b, d, e, g.

8(f). c, e.

10. A graph with no cut vertices cannot have a bridge, hence it has a strongly connected orientation by Theorem 11.1.

12. Argue that if $d_D(u, w) > d_D(u, v) + d_D(v, w)$, then $d_D(u, w)$ cannot be the length of a shortest u–w path, because we can go from u to w through v on a shorter route.

14. 2^{126}.

16. For digraph (a) and measure (1): Essentially the only orientations are: (i) which uses arcs $(a, b), (b, c), (c, f), (f, e), (e, d), (d, a)$, and (b, e), or (ii) which uses arcs $(a, b), (b, e), (e, d), (d, a), (e, f), (f, c)$, and (c, b); both are equally efficient. For digraph (a) and measure (3): Orientation (ii) above is best. For digraph (a) and measure (4): Orientation (ii) above is best. For digraph (a) and measure (6): Orientation (ii) above is best.

18. For (1), show that, for example, the alternating orientation D_1 (see Figure 11.13) is better than any orientation that does not have a cycle around the grid.

23(a). It is easy to construct a path of length $2c$ between any two vertices, so $D \leq 2c$, and the shortest path distance between the two endpoints of a spoke is exactly $2c$.

23(b). It is easy to show two vertices whose distance is exactly $c + \lfloor s/4 \rfloor$. To construct a path of length $c + \lfloor s/4 \rfloor$ between any two vertices consider two cases: When the circles of the two vertices are at least or at most $\lfloor s/4 \rfloor$ steps away.

25. D_1: category 3; D_3: category 2; D_8: category 3.

26. $(3, 2)$: Any arc from $V = \{a, b, c, d\}$ and $A = \{(a, b), (b, c), (c, d), (d, a)\}$.

27. $(3, 2)$: Any vertex in the previous example. **28(a)**. 1. **28(c)**. 2.

30. Hint: Duplicate edges.

Section 11.3.

1. No such walk is possible for (a).

2. G_1 : None; $G_2 : a, c, e, b, f, e, i, j, k, l, i, k, h, g, f, d, a$. **3**. G_1: Yes.

4. D_1: $a, b, d, c, d, f, e, c, a$; D_3: None. **5**. $D_2 : c, a, b, d, f, e, c, d$; D_3: None

9. Hint: Consider an eulerian path of the multidigraph D that starts at a and ends at z. Adding the arc to za to D yields a multidigraph with a closed eulerian path.

10(a). Yes. **10(c)**. No. **12(a)**. $D_1 : 2$; D_2: None. **12(b)**. $D_2 : 2$; $D_5 : 2$.

14. Use the definition: For any two vertices v and w it is possible to find a path leading from v to w; just follow the eulerian path.

16. No: Consider D_2 of Figure 11.25. **20**. Use the hint for Exercise 9.

21. Hint: Devise an algorithm similar to Algorithm 11.4 to find an eulerian closed path.

Section 11.4.

1. Add, for example, a copy of ab, ad, hx, and fx. **3(a)**. Add edge $\{c, f\}$.

3(c). Add edges $\{a, d\}$ and $\{b, c\}$. **3(d)**. Add edges $\{a, d\}$ and $\{b, c\}$.

5(b). A possible sequence is 20210100200012011122212110 2.

6(a). CAAGCUGGUC. **9(a)**. Yes: $A_1 A_1 A_2 A_2 A_2 A_3 A_3 A_3 A_3 A_3 A_2 A_1$.

9(b). There are no such codewords. **11(a)**. Yes.

11(b). Use the fact that the additional edges are added so that in the resulting graph every vertex has even degree.

12. No. Hint: If more than two copies of the same edge are needed, then removing two copies also yields a graph whose vertices have even degree.

17(a). Say B is an interior extended base of a U, C (G) fragment; then both B and the preceeding extended base end in G (U, C), so B is on the second list.

17(b). Both parts of the claim are immediate from the definitions. **18**. Ends in A.

20(b). If there is a second abnormal fragment, it is B alone. **21**. $(\prod_i v_i!)(\prod_{i,j} v_{ij}!)$.

23. Hint: Try modifying the algorithm that finds de Bruijn sequences.

Section 11.5.

1(b). $i, a, b, c, d, e, f, g, h, i$. **2(b)**. a, b, c, d, f, e. **3(b)**. a, b, c, e, d, a.

4(b). a, b, d, c, e, f. **6**. No. **7(a)**. Z_4. **7(b)**. K_4. **8(a)**. $K_{1,3}$. **9(a)**. Yes.

10(a). For (a) of Figure 11.45: Complete graph.

10(b). For (a) of Figure 11.45: Yes. **11**. $K_{1,3}$. **13**. For (a) of Figure 11.4: No.

14. For (a) of Figure 11.4: No. **15(a)**. Use Theorem 11.8.

19. Hint: Add an additional vertex and some arcs, so that the problem reduces to finding a hamiltonian cycle using Theorem 11.10.

20. Hint: A graph that satisfies the modified condition may not even be connected.

Section 11.6.

1(b). Yes. **2**. For (b) of Exercise 1, Section 4.1: No.

3. For (a), the labeling is a topological order. **4(a)**. No. **4(b)**. Yes.

7. 12345, 12453, 13452, 14523, 23451, 24513, 34512, 45123, 51234.

8. SF, B, H, LA, NY or SF, B, LA, NY H, or SF, B, NY H, LA.

10. i beats j iff $i < j$. **11(a)**. $a = 1, b = 2, c = 3, d = 4, e = 5, f = 6$. **12**. 3, 1, 2.

15. If C is a, b, f, d, c, e, a, then H has edges $\{b, d\}$ to $\{e, f\}$ and $\{a, c\}$ to $\{e, f\}$ and is 2-colorable.

19(b). Yes. **19(d)**. No. **20(b)**. (0, 1, 2, 3, 4).

21. Each arc is counted precisely once on both sides of the equation. **22(a)**. No.

22(b). No.

23. Consider the tournament on $V(D) = \{1, 2, 3, 4\}$ and $A(D) = \{(1, 2), (2, 4), (3, 1), (3, 2), (4, 1), (4, 3)\}$.

24(b). Direct the edges of a complete pentagon so that they form two directed cycles of length five.

28(a). $(0, 1, 2, \ldots, n - 1)$. **29**. Yes, $u_1, \ldots, u_n$ is a hamiltonian path.

31. Start with any vertex x and find the longest simple path heading into x; this must start at a vertex with no incoming arcs.

32(a). $\binom{s(u)}{2}$.

32(c). Use part (b) and the fact that $s(u) \geq 2$ for some vertex u in every tournament of four or more vertices.

33. A digraph has a level assignment if and only if it has a topological order. Given a level assignment order the vertices by levels, and in each level order the vertices arbitrarily to obtain a topological order. The conversion from topological orders to level assignments is equally simple. Finally, invoke Theorem 11.14.

36(a). If w is not beaten by u and u does not beat any player that beats w, then clearly w beats more players than u: He beats everyone whom u beats, and he also beats u.

Chapter 12

Section 12.1.

1(a,b,c). $\{a, \alpha\}, \{b, \beta\}, \{c, \gamma\}, \{e, \delta\}$. **3(a)**. $\{a, b\}, \{c, d\}, \{e, f\}$.

5. Find a minimum-weight matching.

8. Let $G = (X, Y, E)$ be a bipartite graph in which the edge $\{x, y\}$ ($x \in X$ and $y \in Y$) has weight r_{xy}. Since we want to match up objects with small r_{xy}, we are looking for a minimum-weight matching.

9. Let $G = (X, Y, E)$ be a complete bipartite graph in which each vertex $i \in X$ corresponds to an item of age a_i at the beginning of the process and each vertex $j \in Y$ corresponds to a removal time t_j. Each edge $\{i, j\}$ has weight equal to u_{ij} (the utility of removing the item i at time t_j). The idea is to maximize the utility, so we look for a maximum-weight matching.

11. For $G = (X, Y, E)$, assume that the edges $f \in F = E(K_{n,n})\backslash E$ with $n = \max\{|X|, |Y|\}$ have weight -1 (actually, any negative number will do). The edges in F will never be part of the solution and so the resulting maximum-weight matching in $K_{n,n}$ will be a maximum-weight matching in G.

13. A possible graph $G = (X, Y, E)$ is given by $X = \{a, b, c\}$, $Y = \{\alpha, \beta, \gamma\}$, $E = \{\{a, \alpha\}, \{b, \alpha\}, \{b, \beta\}, \{b, \gamma\}, \{c, \alpha\}, \{c, \gamma\}\}$. The unique X-saturating matching is $\{\{a, \alpha\}, \{b, \beta\}, \{c, \gamma\}\}$ and $|E|$ is as large as possible. The reason is that $|N(u) \cap N(v)| = 1$ for all $u, v \in X$ with $u \neq v$, where $N(u)$ is the set of all vertices adjacent to u. If we include any edge $\{u, v\} \in E(K_{3,3})\backslash E$ in G we will then have $|N(u) \cap N(v)| = 2$, clearly implying the existence of more than one X-saturating matching.

Section 12.2.

1(a). {Cutting, Shaping, Polishing, Packaging}. **2(a)**. {Smith, Jones, Black}.

3. (a): No; (c): No.

4(a). Let $X = \{S_i\}_{i=1}^{5}$, $Y = \cup_{i=1}^{5} S_i$ and $E = \{\{x, y\} : x \in X, y \in Y, y \in x\}$.

4(b). Just observe that for every $S \subseteq X$, $|N(S)| \geq |S|$. **5(a)**. (a, a, a, a, b, d, a).

6(a). No SDR. **6(c)**. (a_1, a_3, a_2, a_4). **6(f)**. No SDR.

7. Suppose there is no SDR. Thus, by Theorem 12.1 (Philip Hall), there is $S \subseteq \{1, \ldots, p\}$ with $|S| > |N(S)| = |\cup_{i \in S} S_i|$ and this clearly contradicts the condition that $i \in S_i$.

8(a). 2. **8(c)**. 2^5. **10**. No, since for $S = \{a, b, d, e\}$ we have $N(S) = \{2, 3, 4\}$.

11. Let $G = (X, Y, E)$ be any complete bipartite graph with $|X| \geq 2$. We may have $L(x) = \{0\}$ and $L(y) = \{1\}$ for every $x \in X$ and $y \in Y$. This is certainly an L-list coloring for G (because G is bipartite) while there is not an SDR to the set of lists $L(a)$, $a \in X \cup Y$.

12(c). Suppose, without loss of generality, that $|X| \geq |Y|$. Add $|X| - |Y|$ artificial vertices to Y and also add a minimal number of edges until the new graph is $\Delta(G)$-regular. Now, apply the previous exercise.

14(a). Yes. **14(b)**. Three: (c, d, a, b, e), (d, c, a, b, e), (d, e, a, b, c).

16(a). Yes: $\begin{pmatrix} 6 \\ 7 \\ 4 \\ 2 \\ 1 \end{pmatrix}$. **17**. Proof by Corollary 12.1.1 and comment right after it.

18(b). n even. **18(c)**. n odd.

19. Just apply Corollary 12.1.1 (and the comment right after it).

21(a). The pairing graph is bipartite and m-regular. By Exercise 12(a), Section 12.2, the schedule is possible.

22(a). Since an SDR exists, either x is used or it is not. If not and S_i contains x, replace S_i's representative in the SDR with x.

23(a). For example, $(2, 3, 4, \ldots, n-1, n, 1)$.

25. The subgraph H induced by $X_1 \cup N(X_1)$ is bipartite and m-regular ($m \geq 1$). Thus, by Corollary 12.1.1, there is a matching of H saturating X_1. This matching can be extended to be a matching in G, showing that the statement is true, i.e, there is a matching of G saturating X_1.

26(a). Suppose there is $S \subseteq V(G)$ such that $o(G - S) > |S|$. Since each odd component of $G - S$ must have at least one of its vertices matched to a vertex in S in a perfect matching, it is clear that no perfect matching of G can exist under such conditions.

27(a). $E = \{1, 4, 6\}$ or $E = \{3, 4, 6\}$.

27(b). Necessity is obvious: Just pick an element for each $A_i \cap B_i$ and put it into E. For sufficiency, let E be an SCR. For i starting from 1, let $e = E \cap A_i$ (there is a unique e), and let j be such that $e \in B_j$ (there is only one j). If $i \neq j$, swap B_i with B_j. Increment i and repeat the process until $i = m$.

Section 12.3.

1. Yes. **2(a)**. Yes. **2(c)**. No. **2(e)**. Yes. **2(g)**. Yes.

3. Yes, since we have $6 = 2 * 3$ students and each of them likes at least 3 other students. We just have to use Algorithm 12.1.

5. Just follow the correctness proof of Algorithm 12.1 and substitute the new hypothesis appropriately.

Section 12.4.

1(a). $\{1, 2, 3, 4, 5\}$. **1(b)**. $\{1, 2, 4, 5\}$.

3. Consider a graph where the vertices represent the people and the edges the passwords, i.e, there is an edge between two people if and only if both know one of the passwords. Finding a minimum covering in such graph gives the smallest set of people who together know all the passwords.

5. The key property here is that the graph constructed from the matrix must be bipartite (since the result is not valid in general). So, for k being one of the integers used in the matrix, as long as we interpret k as 0 and all the other values different from k as 1, we can still draw the same conclusion. Observe, however, that such a scheme is very artificial.

6(a). Minimum $\{2,6\},\{3,5\},\{1,4\}$. **6(c)**. Minimum $\{1,3\},\{2,4\},\{5,8\},\{6,7\}$.

7(a). No. **7(b)**. $\left\lceil \dfrac{n}{2} \right\rceil$.

8. Construct a graph where the vertices are the streets (middle of the blocks) and two vertices are joined by an edge if and only if the streets meet at an intersection. A minimum covering in such graph solves the problem of surveillance.

9. Minimum edge covering.

12(a). No. Let F be a minimum edge covering containing the circuit $C = u_1, e_1, u_2, \ldots, u_t, e_t, u_1$. Besides the edge e_t, the vertices u_1 and u_t are also covered by e_1 and e_{t-1}, respectively. Thus, the set $F - \{e_t\}$ is also an edge covering of smaller cardinality, contradicting the optimality of F.

13. If I is independent, $V - I$ is a vertex cover and if K is a vertex cover, $V - K$ is independent.

15. $|M^*| \leq |K^*|$ implies $|I| \leq |I^*| \leq |F^*| \leq |F|$.

17. By Exercise 13 we have $|K^*| + |I^*| = |V|$, by Exercise 14 we have $|M^*| + |F^*| = |V|$, and by Theorem 12.5 (Kőnig) we have $|M^*| = |K^*|$. Thus, $|I^*| = |F^*|$.

Section 12.5.

1(a). $8, \{8,9\}, 9, \{9,6\}, 6$. **1(b)**. $8, \{8,9\}, 9, \{9,6\}, 6, \{6,3\}, 3$.

1(c). Add edges $\{8,9\}$ and $\{6,3\}$, delete edge $\{9,6\}$. **3(a)**. $4, \{4,1\}, 1, \{1,3\}, 3$.

3(b). The matching presented is already minimum (and perfect).

3(c). Not applicable. **5(a)**. $1, \{1,2\}, 2, \{2,3\}, 3, \{3,6\}, 6, \{6,7\}, 7$.

5(b). $1, \{1,2\}, 2, \{2,3\}, 3, \{3,6\}, 6, \{6,7\}, 7, \{7,8\}, 8$.

5(c). Larger matching: $\{1,2\}, \{3,6\}, \{4,5\}, \{7,8\}$.

6(a). Pick vertex 8, place edge $\{5,8\}$ in T, note 8 is unsaturated, and note $5, \{5,8\}, 8$ is an M-augmenting chain.

8. Starting at vertex 3 (outer), we might choose edge $\{3,2\}$ in Step 2.1. The vertex 2 is neither inner nor outer. Hence, in Step 2.2 we place $\{3,2\}$ in T and since 2 is saturated, we consider the unique edge $\{2,7\}$ of the matching and put it in T'. We call 2 inner and 7 outer. Now $O = \{3,7\}$ and we might choose the edge $\{3,7\}$ in Step 2.1. We observe in Step 2.2 that 7 is already an outer vertex. This can happen because the graph in question is not bipartite.

9(a). Use edges $\{6,7\}, \{7,11\}, \{11,9\}$.

11. Suppose in Step 2.2 of Algorithm 12.2 that v is an outer vertex. Let C_1 and C_2 be the simple chains with origin in x and endpoints in u and v, respectively. Let y be the least common ancestor of u and v according to C_1 and C_2. Clearly y is also an outer vertex. Both the chains from y to u and from y to v have even length. Together with the edge $\{u, v\}$ we thus have an odd circuit, but this is impossible since the graph is bipartite. Hence, v is not an outer vertex.

14. Let x be an unsaturated vertex in a matching M such that there is no M-augmenting chain starting at x. Now suppose that x is saturated in every maximum matching M^*. Consider the graph $G = M \triangle M^*$ whose edges are in M or M^* but not in both. The component that x belongs to is not a Z_n since x is nonsaturated in M. Thus, it must be of the form L_n, which is an M-augmenting chain, a contradiction.

16. With no blossoms relative to M, the vertex v identified in Step 2.2 of Algorithm 12.2 will never be an outer vertex (cf. Exercise 11). Thus, the algorithm runs as if the graph is a bipartite one.

Section 12.6.

1. (a): $\delta(G) = 1, m(G) = 3$; (c): $\delta(G) = 1, m(G) = 4$.

2(a). Clearly $S \subseteq N(N(S))$, implying that $\deg(S) \leq \deg(N(S))$.

2(b). $2p \leq 3|N(S)|$. **2(c)**. $|S| - |N(S)| \leq p - \frac{2}{3}p$.

2(d). $m(G) = |X| - \delta(G) \geq 9 - \frac{1}{3}(9) = 6$.

4. Consider $X = X_I \cup X_{II}$ and let $S \subseteq X$ such that $|S \cap X_I| = p$ and $|S \cap X_{II}| = q$. By the definition of the switching network, we have $3p + 6q \leq 4|N(S)|$, implying that $\delta(S) = |S| - |N(S)| \leq (p + q) - \frac{3p+6q}{4}$. The maximum of $\delta(S)$ equals 1 precisely when $p = 4$ and $q = 0$. Therefore, $\delta(G) \leq 1$ and there is always a matching that matches at least 7 incoming calls.

7. If $\delta(G) = 0$, the result follows by Theorem 12.1 (Hall). Otherwise, if $\delta(G) > 0$ then Exercise 6 states that $m(G) = |X| - \delta(G)$.

Section 12.7.

1(a). $G' = (\{c, f\}, \{\{c, f\}\})$ and weight$(\{c, f\})$=1. The minimum-weight perfect matching is obvious and the multigraph is then $(V(G), E(G) \cup \{\{c, f\}\})$.

1(c). G' is isomorphic to K_4 with vertex set $\{a, b, c, d\}$. We have weight$(\{a, c\})$=weight$(\{d, b\})$=2 and all the other edges have weight 1. A minimum-weight perfect matching is $\{\{a, b\}, \{c, d\}\}$ and the multigraph is then $(V(G), E(G) \cup \{\{a, b\}, \{c, d\}\})$.

2(a). Worker 1 to job 2, 2 to 3, 3 to 1, and 4 to 4.

3. Worker 1 to job 4, 2 to 1, 3 to 5, 4 to 2, and 5 to 3.

5(a). Speaker 1 with speaker 3, 2 with 5, 3 with 1, 4 with 6, 5 with 2, 6 with 4.

6. No, there are three solutions: $B_1 - H_1$, $B_4 - H_2$, $B_3 - H_3$ or $B_1 - H_2$, $B_2 - H_1$, $B_3 - H_3$ or $B_2 - H_1$, $B_4 - H_2$, $B_3 - H_3$.

8. Let $C = (c_{ij})$ be the current matrix and c be the minimum among the uncovered elements of C. For every column where there is an element twice-covered, add c to every element of it. For every row that is not a covering line, subtract c from every element of it. All previously uncovered elements were subtracted correctly and similarly for all twice-covered elements. Observe that the elements at the intersection of the columns and rows used remain the same.

Section 12.8.

1. $(n!)^{2n}$ since there are $n!$ choices for the preference list for each of the n men and n women.

4. m_3 has w_4 higher on his preference list and w_4 has m_3 higher on her's.

5(a). $m_1 - w_1$, $m_2 - w_2$, $m_3 - w_3$, $m_4 - w_4$.

7. One possible algorithm would be the following. For each man m_i, let w_j be the woman matched to him. Now, for each woman w_k ranked better than w_j in m_i's list, test whether m_i is ranked better than the current pair of w_k in her list. In the affirmative case, m_i and w_k form a blocking pair and therefore the matching is unstable, hence stop. In the negative case, proceed until all the men are evaluated. The answer is that the matching is stable.

8(a). Since M is a matching, we already have that if $b_{i_1} - g_{j_1}$ and $b_{i_2} - g_{j_2}$ and $i_1 \neq i_2$, then $j_1 \neq j_2$. Hence, we just have to show that for each $b_i - g_j$, the men (b_i, m_1) and (b_i, m_2) are matched to women (g_j, w_k) and (g_j, w_l), respectively, with $w_k \neq w_l$. This last relation indeed holds since each M_i is also a matching.

10(a). Because m' proposed to w_j some moment and he was ranked best in her preference list.

10(c). By Exercise 10(b), we know that m' prefers w_j to w_k. Also, we have that w_j prefers m' to m. We have $m - w_j$ and $m' - w_k$ in M and hence, both m' and w_j are simultaneously not paired with their best option. Therefore, m' and w_j are a blocking pair in M, a contradiction since we assumed M was a stable matching.

11(b). No.

Chapter 13

Section 13.1.

1(a). Add edges in the order $\{b, e\}, \{d, e\}, \{a, e\}, \{c, d\}$.

2(a). Add edges $\{a, e\}, \{b, e\}, \{d, e\}, \{c, d\}$.

3(b). Terminate with message disconnected; T ends up with 7 edges.

4(b). Add edges $\{a, b\}, \{a, c\}, \{b, d\}, \{c, e\}, \{d, f\}$.

5. Vat pairs: $\{7, 8\}, \{6, 7\}, \{1, 5\}, \{4, 5\}, \{5, 8\}, \{2, 3\}, \{3, 8\}$.

6. Component pairs: $\{1, 4\}, \{2, 6\}, \{3, 4\}, \{1, 6\}, \{5, 6\}$.

9. (a): Edges $\{b, c\}, \{c, e\}, \{c, d\}, \{a, b\}$.

10. In Step 1 of Algorithm 13.1, arrange the edges of G by decreasing order of weight.

12(a). Edges $\{a, e\}, \{a, b\}, \{c, d\}, \{d, e\}$.

13. In Step 1 of Kruskal's Algorithm, set $T = \{$the set of edges specified as having to belong to the spanning tree$\}$.

16(a). For network (c): G' has edges $\{a, b\}, \{c, g\}, \{a, d\}, \{f, g\}, \{d, f\}$; in the next iteration we add $\{e, f\}$ and obtain a minimum spanning tree.

18(a). Select two vertices disconnected by the cut, and examine the path in the spanning tree that connects them.

18(b). F is clearly a cut (otherwise take a spanning tree of the graph obtained after the removal of F) and it is minimal with respect to inclusion.

21(a). Assign weight 0 to the red edges, weight 1 to the blue ones, and find a minimum spanning tree.

21(b). Use the result of exercise 20.

Section 13.2.

3(a). Successively add to $W : a, b, d, c, e, z$, obtaining the path a, d, e, z.

3(c). Successively add to $W : a, b, c, e, d, z$, obtaining the path a, c, e, z.

5. Grind, weigh, polish, inspect; it takes 21 units of time.

6. Buy in year 1, sell in year 3, buy in year 3 and then sell in year 6.

8. The most reliable path is a, d, e, z. **10**. $(n-2)!$. **12(b)**. $1, 4, 6, 7$.

12(c). First line: Words $1, 2, 3$; second line: Words $4, 5$; third (last) line: Word 6.

14. It will take the boatman seven trips across the river. First the boatman takes the goat across the river. He then goes back and takes the wolf across. He drops the wolf off, but at the same time puts the goat back into the boat. He takes the goat back across the river. The boatman drops the goat off, but at the same time puts the cabbage in the boat. He takes the cabbage across and then goes back and gets the goat.

15. 7: Draw a digraph D where $V(D) = \{x = (x_1, x_2, x_3) : x_i = \text{number of gallons in jug } i\}$, and $A(D) = \{(x, y) : y \text{ is attainable from } x \text{ by pouring one jug into another}\}$ and find a shortest path from (8, 0, 0) to (4, 4, 0).

19. No.

21. Let $\bar{d}(i, j)$ be the distance from i to j and let $\mathbf{A}$ = adjacency matrix; then $\bar{d}(i, j)$ is the smallest k such that the i, j entry of $\mathbf{A}^k$ is nonzero.

24. Modify the stopping criterion (Step 3) of Algorithm 13.3 to "W contains every vertex of the graph" instead of "y is in W."

27. Treat syllables as words.

Section 13.3.

1. For network (c): **(a)** feasible, **(b)** value 4. **2**. (a): 8.

3. (a): $s_{sa} = 3, s_{sb} = 0, s_{ba} = 1, s_{at} = 0, s_{bt} = 2$.

3. (c): $s_{sa} = 0, s_{sc} = 2, s_{ab} = 6, s_{ad} = 2, s_{cb} = 0, s_{cd} = 0, s_{bt} = 0, s_{dt} = 6$.

4. (a): Yes; (c): No. **5(b)**. No. **6**. (a): Augmenting chain s, a, b, t.

6. (c): The flow is maximum; its value is 18. An (s, t)-cut of the same capacity is $(\{s, c\}, \{a, b, d, t\})$.

8. (b): On Figure 13.21a, deleting the 4 arcs entering t clearly destroys all $s - t$ paths. On the other hand, it is easy to find 4 arc-disjoint paths.

9. (b): On Figure 13.21a, deleting vertices d, e, and f destroys all $s - t$ paths. On the other hand, it is easy to find 3 vertex-disjoint paths.

17(b). For graph (a), let the flow be 1 on arcs (s, a), (s, b), (s, d), $(a\alpha)$, (b, β), (d, γ), (α, t), (β, t), (γ, t), and 0 otherwise; the matching is $\{a, \alpha\}$, $\{b, \beta\}$, $\{d, \gamma\}$.

17(c). For graph (a), the (s,t) cut $S = \{s, c, d, \alpha, \beta, \gamma\}, T = \{t, a, b\}$ has corresponding covering $\{a, b, \alpha, \beta, \gamma\}$.

19. The optimal assignment is to do everything with processor 2; total cost is 28.

20. (a): $x_{sa} = 3, x_{sb} = 2, x_{at} = 1, x_{ab} = 2, x_{bt} = 4, x_{sc} = 3, x_{ct} = 3, \text{rest} = 0$.

22. (a): $F(0011) = 1, F(0010) = 0$, and so on.

23. (a): $x_{ad} = 11, x_{bd} = 1, x_{df} = 12, x_{ce} = 5, x_{eh} = 5, \text{rest} = 0$.

24. 7000 drums of paint is the maximum that can be shipped in one day.

27. Apply the same transformation as in the proof of Theorem 13.13.

29. The number of paths the flow decomposes into equals the value of the flow, as well as the capacity of the saturated cut each path crosses once.

30. Assign unit capacities to the arcs. The value of the maximum flow in the resulting network is the maximum number of messengers that can be sent.

33. (a): There is a simple one-to-one correspondence between arc-disjoint paths in D' and vertex-disjoint simple paths in D.

33. (b): First argue that in order to destroy all simple paths from a to z in D' (with the removal of the minimum number of arcs), it suffices to use only arcs (u_1, u_2) corresponding to a vertex u of D.

36. Argue by induction on the length of C that if C is flow-augmenting, it contains a simple flow-augmenting chain.

Section 13.4.

1.(a):$x_{sa} = x_{at} = 2, x_{sb} = x_{bt} = 1, x_{ba} = 0, \text{or } x_{sa} = x_{at} = 1, x_{sb} = x_{bt} = 2, x_{ba} = 0.$

2. (b): The minimum-cost flow has cost 92:
$x_{s1} = 6, x_{s2} = 3, x_{12} = 0, x_{13} = 4, x_{14} = 2, x_{24} = 3, x_{43} = 0, x_{3t} = 4, x_{4t} = 5.$

4. Construct a network with a source s, a sink t, and one vertex for each flight. Add an arc (s, u) for each flight u with capacity equal to the tickets sold for that flight. Add an arc (u, t) for each flight u with capacity equal to the capacity of the corresponding aircraft. Let these arcs have 0 cost associated with them. Finally, add an arc (u, v) between flights u and v if u is earlier than v, with infinite capacity, and cost equal to the cost of bumping one passenger. A minimum-cost maximum flow in this network is in natural correspondence with the optimal solution of the problem.

5. Construct a network similar to the one in Exercise 4, in which napkins "flow" along arcs as they are bought (from the source) or being laundered.

9. Ship 10 spools from warehouse 1 to market (factory) 2; 13 spools from warehouse 2 to market 1; 1 spool from warehouse 2 to market 2.

10(a). (13.13) and (13.14) give us
$\sum_{i=1}^n \sum_{j=1}^m x_{ij} \leq \sum_{i=1}^n a_i < \sum_{j=1}^m b_j \leq \sum_{j=1}^m \sum_{i=1}^n x_{ij}.$

11(a). In (a), such a flow has $x_{sa} = 1, x_{at} = 2, x_{sb} = 2, x_{ba} = 1, x_{bt} = 1$ and a negative cost augmenting circuit is t, a, b, t.

11(b). Increasing the flow along a flow-augmenting circuit modifies the flow without changing its value, while reducing its cost, if the circuit has negative cost.

Author Index[1]

[1] The italicized page numbers refer to reference sections that are at the end of each chapter.

Subject Index