■ Second Edition ■

FOUNDATIONS OF DISCRETE MATHEMATICS

Albert D. Polimeni ■ H. Joseph Straight

Second Edition

FOUNDATIONS OF DISCRETE MATHEMATICS

Albert D. Polimeni
H. Joseph Straight
State University of New York — Fredonia

Brooks/Cole Publishing Company
Pacific Grove, California

Consulting Editor: Robert L. Wisner

Brooks/Cole Publishing Company
A Division of Wadsworth, Inc.

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging in Publication Data
Polimeni, Albert D., [date]
Foundations of discrete mathematics.

1. Mathematics. I. Straight, H. Joseph
II. Title.
QA39.2.P67 1989 510 89-22147
ISBN 0-534-12402-X

Sponsoring Editor: Sue Ewing/Jeremy Hayhurst
Editorial Assistant: Nancy Champlin
Production Editor: Linda Loba
Manuscript Editor: Carol Reitz
Interior and Cover Design: Flora Pomeroy
Art Coordinator: Sue C. Howard/Cloyce Wall
Typesetting: Science Typographers, Inc.
Cover Printing: The Lehigh Press Co.
Printing and Binding: Arcata Graphics—Fairfield

Second Edition

FOUNDATIONS OF DISCRETE MATHEMATICS

Preface

This book is intended as an introduction to discrete mathematics for sophomore-level mathematics and computer science majors. It is meant to be a first encounter with abstract mathematics and, at the same time, an introduction to those topics in mathematics basic to the study of computer science. It can be used for a course that is devoted exclusively to mathematics; indeed, the text material and exercises on computer science can easily be omitted. To facilitate this, the exercises pertaining to computer science concepts have been placed at the end of the exercise sets. If the book is used for such a course, then we recommend a minimum prerequisite of one year of college-level mathematics, including at least one semester of calculus. If one prefers to make use of programming skills and other topics from computer science, then a programming course in a high-level language, such as Pascal, should also be required. Examples in this book that involve programming are most often presented in pseudocode or Pascal.

We use the text for a one-semester (four-credit) course in mathematics and feel that it is important to establish a solid mathematical foundation from which one can smoothly proceed to more advanced courses in mathematics and computer science. For mathematics majors, we find that having had a properly paced treatment of the rudiments of naive set theory, residue class arithmetic, elementary properties of divisibility, mathematical induction, relations, functions, elementary counting techniques, and binary operations provides a much needed edge for courses in linear algebra, abstract algebra, combinatorics, geometry, and analysis. Many of the same ideas are relevant in computer science, for courses in data structures, database systems, algorithms, the theory of computation, and artificial intelligence. Hence, aside from any specific algorithms or applications, it is our hope that a computer science major who completes our course will have developed a good understanding of both the language of mathematics and the fundamental ideas promoted in the course. Such students will have many opportunities, in subsequent course work, to analyze and code specific algorithms.

In a one-semester, four credit course, we find that we can reasonably cover Chapters 1–3, with the exception of the material on program verification (in 1.6) and proofs of program correctness (in 3.5 and 3.6). In Chapters 4 and 5, we generally omit Sections 4.5 and 5.5. Finally, in Chapters 6 and 8, we cover Sections 6.1–6.4, 6.6, 8.1, and as much of 8.6–8.8 as time permits.

We now feel that this text contains enough material for a one-year sequence. Here we would naturally include inclusion-exclusion, recursive algorithms, a good selection of topics from the chapter on graph theory, and further ideas from algebraic structures. Some possible outlines for such a course are presented in the following table:

Chapter	Sections
1	all
2	all
3	all
4	all
5	5.1–5.4
6	6.1–6.7
7	7.1–7.5, 7.9 or 7.1, 7.2, 7.6–7.9
8	8.1, 8.2, 8.5–8.8 or 8.1–8.6

SECOND-EDITION CHANGES

The decision to revise the first edition was based primarily on comments and suggestions solicited from users and selected reviewers. We raised specific questions regarding several aspects of the text and, in most cases, responses were consistent with our own views. We also made use of the report of the *Committee on Discrete Mathematics in the First Two Years*, published by the Mathematical Association of America. The major issues dealt with expanding the coverage on combinatorics, restructuring the chapters on algebraic structures and graph theory, and adding, throughout the text, more exercises of a routine nature. Although most of our efforts were devoted to these areas, other changes have also been effected.

CHAPTER REVISIONS

Chapters 1 through 4 show no significant change. In Chapter 1, Logic, for example, we introduce the notion of a set, along with the standard notations for well-known sets, and define the set relations. This makes it much easier to converse throughout the chapter and also allows one to deal more effectively with restricted quantifiers. In Chapter 2, Set Theory, the student is given the opportunity to use the ideas of proof that he or she has learned in Chapter 1. Since this is the first encounter with problems that use the wording "Prove that...," we have deliberately tried to be careful and somewhat formal in our approach to proofs. This idea is relaxed as one progresses in the text. In Chapter 3, Number Theory and Mathematical Induction, we have made a special effort to include a good number of elementary and instructive exercises. In this chapter students are given the opportunity to develop number theoretic algorithms and, if desired, to implement them in pseudocode or Pascal. In Chapter 4, Relations, we make use of matrix properties of relations. An appendix on matrices has been added at the end of the text.

In Chapter 5, Functions, we have added sections on the big-O notation and set equivalence. The big-O notation was written into the text to furnish the computer science major with an elementary treatment of how one estimates the running time of an algorithm. Set equivalence is more mathematical and is intended to extend the idea of cardinal equivalence to infinite sets.

Chapter 6, Combinatorics, is a new chapter and includes a selection of standard topics. Section 6.3 on permutations and combinations appeared in the second chapter of the first edition and the material on recursion and iteration was formerly in Chapter 5. Among other things, we have added material on the pigeonhole principle, the binomial theorem, and the principle of inclusion-exclusion. Also, the treatment of recurrence relations and recursive algorithms has been expanded.

Chapter 7, Graph Theory, has been changed significantly in that there is generally more emphasis on algorithms and directed graphs. For instance, Warshall's algorithm for determining the reachability matrix of a digraph, as well as the topological sort and depth-first-search algorithms, are included. We have also added sections on eulerian graphs and hamiltonian graphs, and a section on finite state automata. One can provide a basic introduction to graphs by covering Sections 7.1, 7.2 (Eulerian graphs), and 7.6 (Trees). A unit on directed graphs can be given by starting with Section 7.3 (an introduction), followed by 7.4 (Subdigraphs, Reachability, and Distance) and 7.5 (Acyclic Digraphs and Rooted Trees). Sections 7.7 (Hamiltonian Graphs), 7.8 (Vertex Coloring and Planar Graphs), and 7.9 (Finite-State Automata) are more advanced in nature.

In Chapter 8, Algebraic Structures, the discussion of groups has been softened and spread over three sections, as compared to one compressed section in the first edition. Some rather fundamental ideas, which were formerly in the exercise sets, have now become text material. A treatment of normal forms for Boolean switching functions has been inserted after the section on Boolean algebras, and Section 8.8 deals with switching circuits and logic gates. The layout of the chapter allows one to easily pick a small selection of topics to cover. For instance, in our case we try to cover the section on binary operations and the sections on Boolean algebras.

MATHEMATICAL STYLE

Just as with the first edition, we have made a serious effort to maintain a level of exposition that is suitable for a sophomore-level student in mathematics or computer science. At the same time, we have made a special attempt to respect rigor in the mathematics. It is essential at this level that the student learn to read and do mathematical proofs. This is initially dealt with in Chapter 1, where the formal language of mathematics and various proof techniques are covered. In the remaining chapters, theorems are clearly displayed and stated. If the proof of a theorem is given and the method used is other than direct, then the particular proof technique to be used is specified. In all cases we strived to make the presentation both precise and readable.

EXAMPLES

Users and reviewers of the first edition considered the examples to be a strong feature of the text. In this edition we have included most of the examples given in the first edition, with some corrections and improvements; in addition, we have expanded the list of examples in areas where there was a clear need. Our belief has been that there should be examples to illustrate each new concept and to show how theorems and algorithms are applied. Just as in the first edition, we continue to phrase most examples in problem form and encourage students to participate in the development of a solution, to attempt to solve these problems independently and check their work with the given solution.

EXERCISES AND CHAPTER PROBLEMS

Although most users and reviewers felt that there is a good selection of exercises and chapter problems in the first edition, many of them stated that there was a strong need for more routine and elementary exercises that illustrate concepts. This was especially noted in Chapter 3 on number theory and induction. We made this a priority for the current edition and have added many new exercises that we hope will allow the student to move progressively to the more difficult problems in the text. As with the first edition, there are exercises at the end of each section and problems at the end of each chapter; the exercises focus on the material covered in a section, while chapter problems are intended not only for review but to deepen the student's exposure to key ideas. Some problems integrate concepts from several previous chapters or invite the student to explore some aspect of a topic not covered in detail in the text. Exercises and problems that are especially challenging are prefixed with a star (★).

ANSWERS TO EXERCISES AND PROBLEMS

Answers to selected exercises and problems are given in Appendix B, with answers to most of the remaining exercises and problems included in the Instructor's Manual. The rational used in selecting exercises and problems for which to provide answers was twofold: We tried to achieve a balance between the needs of the students, who want to check whether they have done problems correctly, and those of the instructor, who may wish to assign some problems (for which no solution is given) to be collected and graded. With the current emphasis on problem solving and writing in mathematics, we felt that these latter problems should be of a type requiring a more intricate analysis or the writing of a proof.

We would like to thank the following reviewers for their helpful comments and suggestions: Professor Jonathan Barron, Eastern College, St. Davids, Pennsylvania; Professor David Berman, University of New Orleans; Professor Joseph Buoni, Youngstown State University, Youngstown, Ohio; Professor Elwyn Davis, Pittsburgh State University, Pittsburgh, Kansas; Professor Joseph Gallian, University of Minnesota, Duluth; Professor Charles Parry, Virginia Polytechnic Institute and State University; Professor George Peck, Arizona State University; Professor Wayne Powell, Oklahoma State University; Professor Tina Straley, Kennesaw State College, Marietta, Georgia; and Professor Thomas Upson, Rochester Institute of Technology, Rochester, New York.

We wish to express our sincere thanks to our colleagues Nancy Boynton, Fred Byham, and Richard Dowds who used the text and made many valuable suggestions for this revision. In addition, we would like to express our sincere appreciation to our editors, Jeremy Hayhurst and Sue Ewing, for their valuable assistance and patience throughout the preparation of this text.

Albert D. Polimeni
H. Joseph Straight

For the Student

What is discrete mathematics? When addressing this question, one usually draws a contrast between discrete mathematics and *continuous mathematics*. We shall, by illustration and intuition, attempt to convey some feeling for both discrete and continuous mathematics. For instance, the following are examples of problems from discrete mathematics:

1. In how many ways can 12 different problems be distributed among 20 students if no student gets more than one problem and each problem is assigned to at most one student.
2. How many social security numbers have no repeated digits?
3. Given n points in the plane, no three collinear, how many triangles do they determine?
4. Find all integers n such that $n^2 + 1$ is a multiple of 11.
5. Prove that $1^2 + 2^2 + 3^2 + \cdots + n^2 = n(n+1)(2n+1)/6$ for all positive integers n.
6. (Travelling Salesman Problem) A salesman must go on a sales trip, beginning at his hometown H, visiting each of the cities A, B, C, D, E, F, and G. Under what conditions is it possible for him to do so in such a way that he visits each of the cities A, B, C, E, F, and G exactly once and then returns home.
7. If possible, design an algorithm (step-by-step process) that provides a solution to the Travelling Salesman Problem.

Notice that each problem deals with a finite collection of objects, a finite process, the positive integers $1, 2, 3, \ldots$, or all the integers $\ldots -3, -2, -1, 0, 1, 2, 3, \ldots$. Any problem that involves a finite collection of objects or a finite process is a problem in discrete mathematics. More generally, any problem involving any collection of integers is a problem from discrete mathematics. Formally, one should think of discrete mathematics as the mathematics of *discrete sets*. If we think of numbers as points of a coordinate line, then one can think of a discrete set of numbers as a set S of points on the line having the following property

Each point x of S is contained in some interval containing no other points of S.

For instance, the set of integers is a discrete set but the set of rational numbers is not. An interval can be thought of as a "continuous set" of points of the line, and any problem or process that deals with continuous sets can be said to fall in the area of "continuous mathematics." For example, the problems

1. Find the absolute maximum and minimum values of the function

$$f(x) = x^3 - x \text{ on the given closed interval } [-2, 2].$$

2. Evaluate the definite integral

$$\int_{-i}^{3} x^4 + x^3 + 3x + 2\,dx$$

belong to the realm of continuous mathematics. However, the student may recall that there are methods for approximating the value of a definite integral by a Riemann sum (Simpson's Rule provides one such method). A Riemann sum is a finite sum of numbers and, hence, falls within the scope of discrete mathematics. One can use the power of a computer to rapidly compute the value of such a sum and thus obtain a good approximation to the exact value of the definite integral. In many and varied cases one can use discrete mathematics to approximate solutions to problems from continuous mathematics. In this text you will encounter ideas from areas of mathematics that are basic to both discrete and continuous mathematics; however, in the larger share of cases, we will deal with problems and processes involving discrete sets.

Contents

ONE
Logic

1.1 INTRODUCTION TO LOGIC AND SETS

Like other scientists, mathematicians engaged in research are primarily interested in investigating assertions and determining whether a given assertion is true or false. Research problems in mathematics may arise in various ways. For example, it often happens that interesting problems arise as a result of seminar discussions with colleagues, poring over research articles, or attempts to solve other problems. Once an assertion is clearly stated, a mathematician uses formal training and acquired knowledge and experience to try to resolve it. Valuable insight and understanding are gained by building examples that illustrate the ideas involved. Of course, if an example denies the assertion, then one need go no further; the assertion is false. If all the examples considered seem to support the assertion, along with other related information (such as computer-generated data and previously established results), there may be good reason to suspect that the assertion is true. Then, to be called a mathematical fact, or *theorem*, the result must be logically deduced from known facts; that is, it must be proved.

The logic of mathematics furnishes a set of ground rules that govern the development of a mathematical proof. These rules allow us to determine whether a proposed proof is valid. They are one aspect of proof that can be learned easily; the rest is intuition, creativity, imagination, and instinct.

The deductive method is important in computer science as well. Here, there is great interest in the process of designing, coding, and testing programs, particularly large, complicated programs. Ideally, the rules of logic are used in this process, with the aim of ensuring program correctness. Indeed, one often is required to give or read a proof that a given algorithm "works." There is also a relatively new programming methodology, called logic programming, that explicitly embodies the ideas of mathematical logic. A popular language for logic programming is PROLOG. It is widely used to program expert systems—programs that simulate the deductive analysis of the human expert in some narrow domain, such as medical diagnosis.

Emphasis in this chapter is placed on certain common forms of exposition and reasoning, apart from any particular applications. A student who learns these well finds it easier to follow the line of reasoning in advanced mathematics and computer science courses. This allows the student to focus on the content of a course and not be distracted by the logical forms being used.

Hereafter the term *argument* is used in its mathematical sense, as a logical discussion that establishes the validity of some mathematical fact. We ask, then, What is the logic of an argument? Crudely put, the logic of an argument is what is left over when the particular meaning of the argument is ignored. In other words, the logic of an argument is its form or syntax.

A similar distinction is made concerning the statements in a programming language like Pascal. These statements must obey certain syntactic rules, apart from their actual semantic interpretation. For example, consider the assignment statement

```
SUM := SUM+NEXT
```

We can say that this statement is syntactically correct but, when taken out of the context of a particular program, its meaning is not clear.

Perhaps an example will help to clarify the preceding comments.

Example 1.1 Consider the following argument about Randy, who is a student in the discrete mathematics course at a particular college.

> If Randy has the ability and works hard, then Randy is successful in this course. Therefore, if Randy is not successful in this course, then Randy does not have the ability or Randy does not work hard.

This argument is symbolized by letting p, q, and r represent the following statements:

p: Randy has the ability.

q: Randy works hard.

r: Randy is successful in this course.

The logical form of the argument is then represented as follows:

If p and q, then r. Therefore, if not r, then not p or not q.

It is easy to think of other arguments with the same logical or abstract form, such as:

> If the Red Sox hit well and avoid injuries, then they will win the pennant. Therefore, if the Red Sox do not win the pennant, then they did not hit well or they did not avoid injuries. □

Note that the abstracted form of an argument such as the one in the preceding example is not bound to any particular content. Logic provides a means for validating such an abstract form in a way that is independent of the truth or falsity of the constituent statements represented. This is done through the use of "symbolic logic" and the construction of "truth tables." We discuss how this is done later in the chapter.

It is important to understand the meaning of the term *statement* as it is used in symbolic logic. Sentences such as

Everyone in this class gets an A grade.

or

The number 72 is positive.

are "declarative sentences"—each makes an assertion. On the other hand, the question

What time is it?

and the exclamation

Holy cow!

are not declarative sentences. For purposes of mathematical logic, our interest is in declarative sentences that are either true or false; these are called *statements* or *propositions*. If a statement is true, then its truth-value is denoted T, whereas the truth-value of a false statement is denoted F.

For instance, each of the sentences

The quotient obtained when 7 is divided by 3 is an odd integer.

and

The integer 6 is a factor of the product of 117 and 118.

is a statement. In fact, the first is false and the second is true.

Next consider the sentences

The number x is positive.

and

He is a baseball pitcher.

These are not statements because, as they are presented, we cannot determine the truth-value of either one. If we replace x by -3 in the first sentence, then we obtain the (false) statement

The number -3 is positive.