# DISCRETE MATHEMATICS WITH ALGORITHMS

MICHAEL O. ALBERTSON
JOAN P. HUTCHINSON

# *DISCRETE MATHEMATICS WITH ALGORITHMS*

**Michael O. Albertson**

**Joan P. Hutchinson**

*Smith College*

WILEY

JOHN WILEY & SONS
*New York Chichester Brisbane Toronto Singapore*

***Library of Congress Cataloging-in-Publication Data:***

Albertson, Michael O.
Discrete mathematics with algorithms.
Includes indexes.
1. Mathematics—1961- . 2. Electronic data processing—Mathematics. I. Hutchinson, Joan P. II. Title.
QA39.2.A43 1988 510 88-235
ISBN 9971-51-308-0

Printed and bound at Replika Press Pvt Ltd, 100% EOU, Delhi-110 040, India

10 9 8 7 6 5

# *DISCRETE MATHEMATICS WITH ALGORITHMS*

Dedicated to

*Margot, Matt, Nick, and Beth,*
*and to*
*Neal H. McCoy*

# PREFACE

Discrete mathematics is playing an increasingly central role in the development of computer science and in both pure and applied mathematics. Consequently, there is pressure to teach courses in discrete mathematics earlier in the college curriculum. This makes sense. The material is accessible to students at this level, probably more accessible than calculus. Furthermore, a background in discrete mathematics is necessary for study in mathematics and computer science and is helpful throughout a broad spectrum of the sciences.

Historically, Finite Mathematics as a freshman course has had two fatal flaws. First, it did not lead anywhere in the curriculum. Second, the material was not a coherent entity. Within the mathematics curriculum, Discrete Mathematics can be a prerequisite for Linear Algebra and Number Theory and should be a prerequisite for Combinatorics and Graph Theory, Linear Programming, and Probability. Within the computer science curriculum it would be desirable to study Discrete Mathematics either before, or concurrently with, Data Structures. Both the Foundations of Computer Science and the Design and Analysis of Algorithms should have Discrete Mathematics as a prerequisite. Thus the influence of computer science has firmly placed Discrete Mathematics in the mainstream of both mathematics and computer science.

To cure the second flaw, we adopt algorithmic reasoning as our unifying theme. We are problem solvers by nature and want efficient algorithmic solutions in preference to existential results. Our paradigm begins with a specific mathematical problem that is transparent and easy to solve in small instances. The naive algorithm that works in small cases may require unimaginably large amounts of computation when the problem size is increased (sometimes only modestly). Then the search is on for an efficient algorithm. For each major problem considered, we construct an algorithm whose performance is markedly better than the naive algorithm. Discrete mathematics provides the tools to understand these algorithms, prove their correctness, and analyze their efficiency.

This book has no calculus prerequisite and no calculus is used. However, four years of high school mathematics are required for technique and sophistication. Some exposure to computers is advisable, but no programming experience is needed. The algorithms are presented in English in a format compatible with the Pascal programming language. Most could easily be turned into Pascal programs. If further technical details are needed to implement the algorithm as a program, these are included in the exercises. We have found it instructive to have the students observe computer implementations of many of the algorithms, especially to see the differences in efficiencies of different algorithms.

The text is meant to be *read* **by the student**. We mean the whole text including examples, questions, and proofs. Mathematics is not a spectator sport! Acquiring an understanding of the definitions, theorems, examples, and algorithms requires participation. Designated questions are placed throughout the text. These involve checking examples, exploring newly introduced theory, and working through algorithms by hand. The questions should be required work. Specifically, every question should be attempted before the material following it is read. We believe that it is crucial for the student to work out the solutions to these questions in order to become involved with the material as it is presented. To facilitate this end, solutions to all questions are supplied at the end of the book. Detailed solutions are given for the more difficult questions. With these, students can check their work and improve their understanding. After completing a section, the student should attempt a substantial number of the exercises at the end of the section. (In later references an exercise labeled $x.y$ is the $y$th exercise in Section x. If a chapter reference is necessary, it is indicated.) The exercises are not necessarily listed in the order of increasing difficulty; however, at the end of each chapter the Supplementary Exercises contain challenging problems on material from the entire chapter.

*Discrete Mathematics with Algorithms* is intended as a textbook for a one-semester course at the freshman-sophomore level. All the material cannot be covered in one semester so that choices must be made. In our 13-week semester we typically teach most of Chapters 1–5, several sections of Chapter 8, and an introduction to Chapter 6. A one-quarter course might cover Chapters 1–3 and one additional chapter. A two-quarter course could cover the entire text. Chapters 1, 2, and the beginning of 3 are necessary for the rest of the book.

Chapter 1, which introduces set theory, the definition of an algorithm, and the basic properties of functions and Boolean functions, can be covered in two weeks (less if the students' backgrounds warrant). Some material on functions can be deferred until needed later; the Satisfiability Problem can be omitted or deferred. There are three substantive topics in Chapter 2. First, the students meet induction proofs. We recommend ample written work. Submitting and resubmitting induction proofs until they are correct is an effective technique. Second, the big oh concept is not easy to grasp. It is the discrete mathematics analogue of the definition of limit in the calculus. Computer science students tend to see a rather casual development of big oh in their intermediate and advanced courses. It is our experience that the big oh concept becomes easier to understand after the students see it applied. Finally, proofs by contradiction are introduced with a concurrent discussion of logical reasoning. A total of 3–4 weeks should be spent on Chapter 2. The material in Chapter 3 on subsets and binomial coefficients goes more rapidly. Two weeks suffice. It is possible to omit the algorithm JSET, the game of Mastermind, and harder applications of the binomial theorem if time is tight.

With the semester approximately half over it is time to embark on the optional material. Each of the remaining five chapters can be thoroughly covered in two or three weeks. Each contains material that can be omitted. We enjoy the material on Fibonacci numbers and the Euclidean algorithm in Chapter 4. It is very different in content from most of what gets taught in discrete mathematics courses. Students find the application to public key encryption memorable. This application necessitates an introduction to relations, modular arithmetic, and basic results of number theory. Complete induction is first explained in this chapter. Although the Fibonacci numbers reappear for motivation in Chapter 7, no subsequent chapter depends on Chapter 4.

Chapter 5 forms a brief introduction to graph theory. The goal is a minimum-weight spanning tree algorithm, and only those graph theory definitions and results necessary for this end are introduced here. Students frequently implement such an algorithm in a Data Structures course. Here they understand how, why, and how efficiently it works. If time is short, the final section on greedy algorithms can be safely omitted. The material on trees reappears in the middle of Chapter 6. We believe that at least one additional graph theory application from Chapter 8 should be covered.

Chapter 6 covers sorting and searching algorithms. These algorithms are implemented in Data Structures, but the analysis is usually omitted. Proving, for instance, that the binary insertion sort is essentially best possible is important for the computer science student. This material is covered in five sections and forms a complete unit. If time permits, the final sections provide an introduction to recursion and recursive algorithms. Many earlier algorithms are reformulated using this technique and merge sorting is presented as an efficient recursive sorting algorithm.

Many algorithms in Chapter 6 lead naturally into the material of Chapter 7 on recurrence relations. The topics of sequences and recurrence relations in Chapter 7 are classic combinatorics, essential for the analysis of recursive algorithms. The recurrence relations are motivated by topics introduced earlier and are solved first by common-sense approaches, then by the general theory of linear homogeneous recurrence relations with constant coefficients, and finally by divide-and-conquer methods.

Chapter 8 contains an algorithmic development of some of the most important problems in graph theory. Each of the five sections introduces a real-world problem, modeled by graphs. The underlying abstract graph theory is developed, and then algorithms, derived from this theory, are applied to solve the problem at hand. Each section is independent of the others (except that Section 8.5 uses material from part of Section 8.3) although each uses and reinforces material from Chapter 5. In some cases a best-possible solution to the graph theory and real-world problem is produced. In one section an efficient approximation algorithm is used to search for an optimal solution. In the instances of three well-known NP-complete problems, the unsettled state of affairs is discussed, since there are no known polynomial-time solutions and yet no one has proved that there cannot be such a polynomial solution.

There is no consensus among either mathematicians or computer scientists concerning the curriculum of discrete mathematics. We believe the essentials of such a course must include an introduction to proofs, especially by induction, and an introduction to algorithmic problem solving. Otherwise, students are best served by working on a few realistic problems, developed in depth. This book includes material that we have been teaching for four years in a freshman-level Discrete Mathematics course. We have found the material and its presentation effective and relevant for further work in mathematics and computer science. Our colleagues in computer science appreciate the improved understanding their students in Data Structures exhibit and the increased amount of material the foundations course can cover. We have found Discrete Mathematics to be an excellent beginning college mathematics course for well-prepared freshmen and an appropriate transition for all students from the computational approach of low-level mathematics courses to more problem-oriented and abstract intermediate courses.

We especially thank our colleagues David Berman, Jim Henle, Marjorie Senechal, and Patricia Sipe for teaching from preliminary versions of this book and for their helpful suggestions. We also thank the Mathematics 153 students of Smith College for their patience and perseverance as this book was being prepared. We are grateful to Ellen Gethner for her help in preparing solutions to the book's questions. In addition, we have benefited from insightful comments from Larry Carter, Jeanne Ferrante, Andrew Pasquale, Margot Thomas, and Stan Wagon. Finally, we thank Gene Davenport, our editor, Deborah Herbert, Elaine Wetterau, copy editors, and Pam Pelton, production supervisor, at John Wiley & Sons for their help in preparing the manuscript.

Michael O. Albertson
Joan P. Hutchinson

# CONTENTS

# 1

# *SETS AND ALGORITHMS: AN INTRODUCTION*

## 1:1 INTRODUCTION

The four cards labeled *A*, *B*, *C*, and *D* in Figure 1.1 are part of a magic trick played by Player 1 upon Player 2. The trick is played as follows:

| *A* | | *B* | | *C* | | *D* | |
|---|---|---|---|---|---|---|---|
| 8 | 9 | 4 | 5 | 2 | 3 | 1 | 3 |
| 10 | 11 | 6 | 7 | 6 | 7 | 5 | 7 |
| 12 | 13 | 12 | 13 | 10 | 11 | 9 | 11 |
| 14 | 15 | 14 | 15 | 14 | 15 | 13 | 15 |

**Figure 1.1**

| *Player 1* | *Player 2* |
|---|---|
| Pick a whole number between 0 and 15. | |
| Got it? | Yes. |
| Is it on card *A*? | Yes. |
| Is it on card *B*? | No. |
| Is it on card *C*? | No. |
| Is it on card *D*? | Yes. |
| The number you picked is 9. | That's amazing! How did you do that? And so fast! |

Now let's play again only this time you'll be player 1. I have a whole number between 0 and 15. It appears on cards $A$, $C$, and $D$ and does not appear on card $B$. What number am I thinking of?

**Question 1.1.** (Figure it out before you read any further.) If you are at a loss for what to do, ask yourself the following questions. Can it be 0? Can it be 1? . . . Can it be 15?

Now it can't be 0 because 0 doesn't appear on any of the cards and the number I'm thinking of appears on three cards. It can't be 1 because even though 1 does appear on card $D$, it does not appear on cards $A$ or $C$, and the number I'm thinking of appears on both cards $A$ and $C$. If this magic trick is well designed, meaning that it is always possible for player 1 to guess player 2's number correctly, then there must be a unique number that corresponds with any possible sequence of answers provided by player 2. In this case the number I am thinking of is 11. It is easy to check that 11 appears on the cards $A$, $C$, and $D$ but does not appear on the card $B$. It seems less obvious that 11 is the only such number.

**Question 1.2.** What would you need to do to check that this trick will always work?

**Question 1.3.** Design a pair of cards that will serve to distinguish the numbers 0, 1, 2, and 3. Is there more than one way to do this? Why can't two cards distinguish the numbers 0, 1, 2, 3, and 4?

Understanding why two cards can distinguish four numbers and why four cards can distinguish 16 numbers is fundamental to seeing how to design this game as well as how to play it well. Each card that player 1 shows to player 2 elicits one of two responses, either a "yes" or a "no." A game with two cards has four possible responses from player 2. These are "no, no," "no, yes," "yes, no," and "yes, yes." How many responses has a game with four cards? Justice seems to suggest that you respond 16. That is correct. Now let's see why.

*Multiplication Principle.* Suppose that a counting procedure can be divided into two successive stages. If there are $r$ outcomes for the first stage, and if for each of these outcomes for the first stage, there are $s$ outcomes for the second stage (where $r$ and $s$ are positive integers), then the total number of possible outcomes equals the product of $r$ and $s$, $rs$.

**Example 1.1.** At tea one afternoon you are offered your choice of a bagel, a corn muffin, or a croissant with either cream cheese or lightly salted butter. How many different choices do you have? (Reread the multiplication principle.) At the first stage you can choose whether to have a bagel, muffin, or croissant. There are three different outcomes ($r = 3$). At the second stage you can choose cheese or butter. There are two different outcomes ($s = 2$). By the multiplication principle as well as by a direct count you have 6 ($=rs$) choices.

**Example 1.2.** In the magic trick, how many different responses are there to the four cards? (Reread the multiplication principle.) First consider cards $A$ and $B$. As we've already seen, there are four distinct responses to these two cards ($r = 4$). Next look at cards $C$ and $D$. It doesn't matter what the responses to the $A$ and $B$ cards were. There are four distinct responses to these two cards ($s = 4$). Thus there are 16 ($=rs$) distinct responses in all to the four cards. Note that these 16 responses could have been counted in four stages with two responses at each of these stages. The multiplication principle works analogously for any number of stages. (See Exercises 7 and 8.)

**Question 1.4.** How many different seven-digit telephone numbers are there beginning with the digits 584?

Now returning to the magic trick, you see that player 1 could perform the trick by memorizing the 16 different responses that player 2 might give in order to successfully "guess" player 2's number. The possible responses are listed in Table 1.1.

**Table 1.1**

| | *Responses* | | | |
|---|---|---|---|---|
| *Player 2's Number* | *Card A* | *Card B* | *Card C* | *Card D* |
| 0 | no | no | no | no |
| 1 | no | no | no | yes |
| 2 | no | no | yes | no |
| 3 | no | no | yes | yes |
| 4 | no | yes | no | no |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | | |
| 9 | | | | |
| 10 | | | | |
| 11 | | | | |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |

**Question 1.5.** Complete Table 1.1.

## EXERCISES FOR SECTION 1

1. Design a set of three cards that will distinguish the numbers $0, 1, 2, \ldots, 7$. Suppose that we only wished to distinguish the numbers $0, 1, 2, \ldots, 5$. Could your three cards be modified to play this game? Could your three cards be modified to play the game with the numbers $0, 1, 2, \ldots, 8$?
2. Suppose that the local ice cream store offers 12 different flavors of ice cream and 5 different types of topping (chocolate, butterscotch, strawberry, blueberry, and raspberry). How many different dishes of ice cream plus topping are possible? Suppose that you can turn these dishes of ice cream plus topping into special sundaes by adding one kind of nuts (walnuts, almonds, or hazelnuts) and whipped cream if you like. How many different types of special sundaes can you order at this ice cream store?
3. A certain fast food chain offers a one-price meal consisting of a burger, an order of potatoes, a salad, a dessert, and a beverage. There are seven different kinds of burgers, three different kinds of potatoes, five different kinds of salads, and four different kinds of desserts. The restaurant advertises that you can eat one meal here every day for four years without ever having the same meal twice. What can you say about the number of beverage choices that the restaurant offers?
4. Often, when you sign onto a time-sharing computer, you are asked to specify the room you are in and the kind of terminal that you are using. Suppose that there are 13 different room categories and 16 different kinds of terminals. How many different pairs of answers is it possible to give as you sign on?
5. In the context of the preceding problem it is typically the case that not all answers are possible, since there are not 16 different kinds of terminals in every room. If every room contains four different kinds of terminals, how many different answers are possible?
6. Even the idea in the last problem might not be correct, since the kind and number of terminal types may vary from place to place. Suppose that we consider only five rooms and that they contain the following kinds of terminals: Every room contains a Digital VT terminal; Tektronix machines are located in the Social Science Room and in the Science Lab; IBM PCs are found in the Graphics Lab and in the Library Terminal Room; and Apple Macintoshes are available in the Library Terminal room and in the Humanities Computer Room. How many pairs of responses are now possible to send to the computer when you sign on?
7. Here is an extension of the multiplication principle: Suppose that a counting procedure can be divided into four successive stages. If there are $p$ outcomes for the first stage, if for each of these outcomes for the first stage, there are $r$ outcomes for the second stage, if for each pair of these first two outcomes,

there are $s$ outcomes for the third stage, and finally if for each of these first three outcomes, there are $t$ outcomes for the fourth stage (where $p$, $r$, $s$, and $t$ are positive integers), then the total number of possible outcomes equals the product $prst$. Explain why this is valid, using the original form (two-stage) of the multiplication principle.

**8.** State and explain a multiplication principle that is valid for three stages, and then do the same for five stages.

**9.** Suppose that we have a rather primitive computer that can receive only strings of zeros and ones as input. Furthermore, these strings must contain exactly eight digits. How many different input strings are there?

**10.** Suppose that the machine in the preceding problem can receive strings with one to eight digits, and suppose that the machine disregards initial zeros. Thus, for instance, 1001 is the same input as the string of eight digits, 00001001. Now how many different input strings are possible?

**11.** How many different seven-digit phone numbers are there that begin 584 and contain no zero? How many phone numbers are there that begin 584 and contain at least one zero?

**12.** How many different seven-digit phone numbers are there that begin 58_-____ and contain seven different digits? How many of these contain no zero? How many do contain a zero? How many different phone numbers are there that begin with 58_, but contain no two identical consecutive digits?

**13.** Recently, a new telephone area code was introduced for the area of New York that contains Brooklyn and Queens because all seven-digit phone numbers had been used up. Assuming that none of the first three digits in a phone number can be either a 0 or a 1, what can you say about the number of phone lines in this area?

**14.** In the lottery game called Megabucks a player selects six different numbers between 0 and 35. How many different such selections are there? Before answering, specify when two selections are the same and when they are different.

## 1:2 BINARY ARITHMETIC AND THE MAGIC TRICK REVISITED

The magic trick of Section 1 was based on each of four questions receiving either a "yes" or a "no" answer. Thus a seemingly complex task, in this case deciding which number player 2 had chosen, could be broken down into a sequence of smaller tasks associated with each of the cards. This fundamental yes-no, true-false, or on-off dichotomy pervades most of the mathematics associated with computers. It even is fundamental to how computers "think" about numbers. We now model how a computer stores an integer using binary numbers.