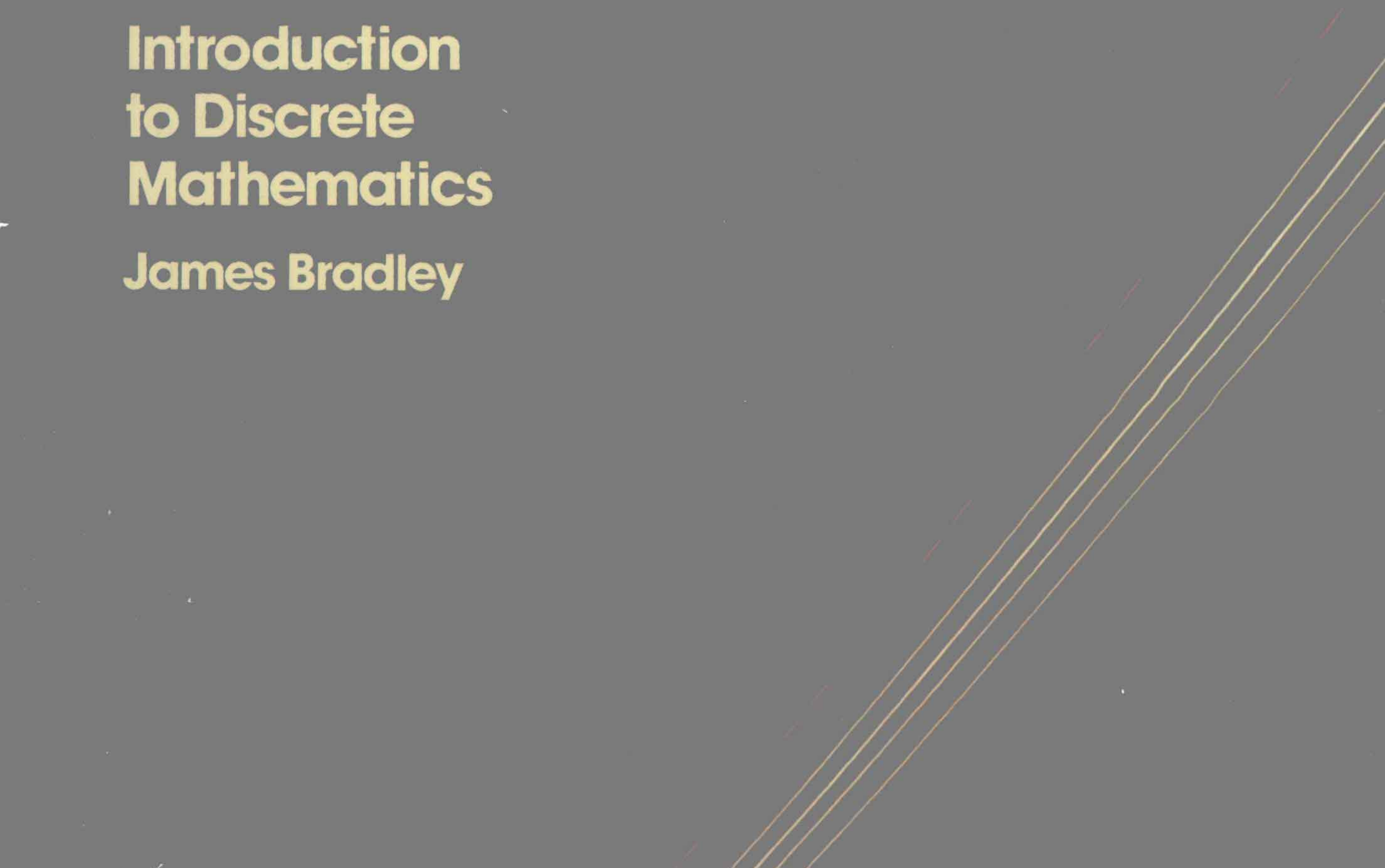
Introduction to Discrete Mathematics

James Bradley



Introduction to Discrete Mathematics

James Bradley
Calvin College

Addison-Wesley Publishing Company
Reading, Massachusetts · Menlo Park, California · New York
Don Mills, Ontario · Wokingham, England · Amsterdam · Bonn
Sydney · Singapore · Tokyo · Madrid · Bogotá · Santiago · San Juan

Sponsoring Editor: Thomas N. Taylor
Production Supervisor: Marion E. Howe
Art Consultant: Loretta Bailey
Copy Editor: Helen Greenberg
Illustrator: KATERPRINT Company, Ltd.
Manufacturing Supervisor: Roy Logan
Cover Design: Marshall Henrichs
Text Designer: Marie McAdam

Library of Congress Cataloging-in-Publication Data

Bradley, James, 1943–
Introduction to discrete mathematics.

Includes index.
1. Mathematics—1961– . 2. Electronic data processing—Mathematics. I. Title.
QA39.2.B725 1987 004′.01′51 87-27067
ISBN 0-201-10628-0

To my children:
Jeanette and Peter

Preface

In recent years there has been a growing recognition of the potential value of a freshman–sophomore level course in discrete mathematics that is not calculus based. The primary impetus for such a course has come from computer science where faculty have recognized the existence of a broad collection of discrete mathematics concepts underlying even the beginning courses in the computer science curriculum. But faculty in other disciplines also have seen the value of such a course, notably faculty in mathematics, the social sciences, engineering, and the natural sciences. This book is written as a text for such a course.

Goals

Shortly before I started writing this book, a group of leading mathematicians held a conference to discuss the future of college mathematics, especially the role of discrete mathematics in the first two years of the curriculum. Many expressed goals they felt should characterize discrete mathematics courses at that level. Here is a list gleaned from their proceedings:

Students should

- develop in mathematical maturity—modeling and reasoning skills plus the ability to estimate, generalize, simplify and detect sloppy reasoning;
- master the basic concepts, results, methods, vocabulary, and notation associated with contemporary discrete mathematics;
- become acquainted with historically and culturally significant problems in the field such as the traveling salesman problem, the knapsack problem, and the Hamilton cycle problem;
- appreciate the need for proof and be able both to read and to do elementary proofs;
- appreciate the place of creativity in mathematics and have some experience applying their own creativity to discrete mathematics;
- be provided with a mathematics corequisite for the introductory computer science courses and a prerequisite for subsequent courses in data

structures, theory of computation, algorithms, and advanced discrete mathematics;

- be able to apply standard algorithms to common discrete structures and modify these algorithms when necessary;
- be able to analyze the efficiency of such algorithms and compare the quality of algorithms in terms of elegance and efficiency;
- understand the notion of discrete mathematical model and be able to use basic discrete mathematics tools to model appropriate situations;
- enjoy the subject.

Besides accomplishing the above goals, a text in discrete mathematics is expected to be at an intellectual level comparable to a Calculus text and have a good measure of unity.

Features

I believe it is possible to write a discrete mathematics text that can help students significantly in accomplishing the above goals and at the same time have unity and be at a level comparable to Calculus. I have used the following features in trying to do this:

- An algorithmic approach. Over forty algorithms are included in the text. These are written in a Pascal-like pseudo-code with frequent comments to make them as readable as possible. Also the complexity of almost every algorithm presented is computed and its correctness verified. The concepts of complexity and verification of algorithms are of critical importance in computer science, and they also have significant mathematical content. The approach is based on the conviction that a discrete mathematics course is the proper place to give these topics the careful mathematical treatment they need, and thus they are treated here with some care. This has the fringe benefit of strengthening in students' minds the linkage between mathematics and computer science.
- Content has been selected for consistency with recommendations of MAA panels, articles published in the *Communications of the ACM,* and other studies of appropriate topic selections for a freshman–sophomore level discrete mathematics course.
- Motivational discussions, applications, and examples are included throughout to make the book as readable and accessible as possible for freshmen and sophomores who have not had calculus.
- The text has been class tested over a period of three years and much of its development is based on that experience. Students have found it highly readable and feel that they are able to learn from it.
- The book begins with a discussion of the traveling salesman problem and the problem is used throughout to illustrate concepts. This is a very important research problem in contemporary mathematics and computer science and yet the statement of the problem can be easily

grasped by novices. Thus students are brought immediately into contact with the research frontiers in the two disciplines. But also, almost every discrete mathematics topic discussed in this text is applicable to the study of the traveling salesman problem. Thus it serves as a unifying theme for the material.

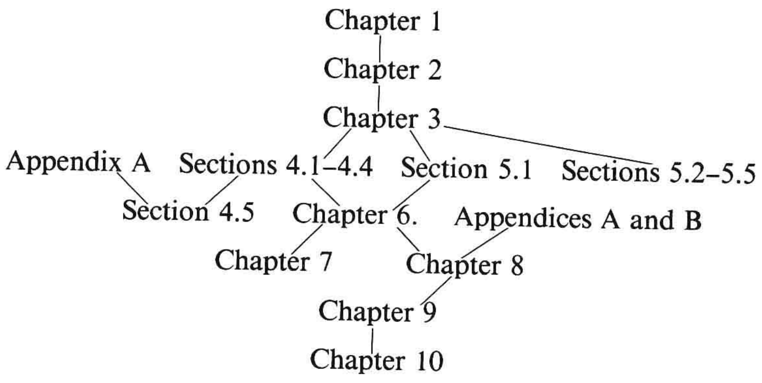
- The history of many topics is explained and historically important problems are presented and discussed at a number of points.
- The proofs of theorems are included throughout with only a few exceptions. These are written to be accessible to freshmen and sophomores, but without compromising mathematical rigor.
- Many exercises are included with each section and these are graded in difficulty from routine to quite challenging. Many exercises requiring proofs are included. In Section 3.6, where rules of inference are presented, a number of exercises are included that are especially designed to introduce students to the concept of proof.
- The notion of mathematical model is presented in the first chapter. Models are discussed throughout the book and a number of exercises requiring students to form elementary models themselves are included.
- A number of features are included to help integrate the text; the most prominent of these are algorithms, discrete models, and the traveling salesman problem. Recursion is also a very important integrating theme in the second half of the book.
- The text includes many examples and applications. The most frequent ones are drawn from computer science; however, many are drawn from the social and natural sciences as well.
- Each section concludes with a list of terminology introduced in the section and a summary of its main points. Students have frequently commented that they found the terminology list and summaries helpful.
- A glossary is included. There is quite a bit of new terminology introduced in a discrete mathematics course, and the glossary provides an easy way for students to look up terms whose meaning they may have forgotten.

Organization

Chapter 1 is primarily motivational but it serves two other valuable purposes—it introduces the notion of computational complexity in an informal way and it introduces the concept of the mathematical model. Thus the unifying themes of algorithms and discrete models are introduced right at the beginning. Chapter 2 is background material in sets, functions, and finite series. Much of this material will be review for many students, so it should be covered as quickly as possible; it is included for completeness. Chapter 3 deals with logic; the material on propositions and predicates in Sections 1 through 4 is also likely to be review for many students and should be covered quickly. Sections 5 through 8 contain extremely important material, dealing with quantifiers, rules of inference, mathematical induction, and combinational circuits. Chapter 4 is on algorithms. It begins with a discussion of the algorithmic language used here and then proceeds to a thorough, mathematically grounded discussion of O-notation, an important topic that is often given superficial treatment. Both the analysis and verification of algorithms and the notions of efficient and inefficient algorithms are introduced here. Chapter 5 deals with basic concepts in number theory. Chapter 6 deals with recursion. I have found that recursion is best explained against a background of mathematical induction. Also students are generally more familiar with equations than algorithms, so recurrence equations are done first, then recursive algorithms. This allows the use of recurrence equations in the analysis of the complexity of recursive algorithms, and induction in their verification. Chapter 7 addresses combinatorics and discrete probability. Chapter 8 deals with relations and introduces digraphs. Rather than introducing relations with functions in Chapter 2, I find it more effective to save relations until students have more examples to build their understanding on. Also, digraphs provide a nice tool to illustrate relations and this way I can introduce digraphs and graphs in subsequent chapters. Chapter 9 deals with graphs and Chapter 10 with trees. Several appendixes are also included; the first two include essential material on arrays and matrices. These topics are included in appendixes to give instructors more flexibility as to when to present them or, with students who have had them before, to skip them. In any case, students will need the material in Appendix A before Section 4.5, if that section is done, and Appendixes A and B before Chapter 8. The other appendixes provide additional depth on some topics that I wanted to include but felt were inappropriate for the body of the text.

Dependency Chart

The following chart illustrates the interdependency of sections:



Sections 3.8, 4.5, 6.4, 6.6, 8.3, 8.4, and 9.4 and portions of some other sections could be omitted without loss of continuity.

Acknowledgments

Above all else, I want to express my thanks to God both for the gift of the ability to write this text in the first place and for the stamina to complete it. I also want to thank my wife, Hope, and my family for their patient endurance of evenings and Saturdays I spent in front of a keyboard rather than with them. My colleagues, both at Nazareth College of Rochester, NY, and at Calvin College, deserve a great deal of credit for their encouragement and help, notably Janet Elmore (who suggested the problem that introduces Chapter 6), Judith Rose, Herbert Elliott, Mary Harrigan, Richard DelVecchio, Joseph Kelly, John Edelman, and many others. I thank my students who not only patiently endured drafts and revisions but also offered me the benefit of their considerable wisdom as to what was presented effectively and what was not. Of particular note is John Freckleton, who strongly encouraged me to write. But I would also like to thank the following students from Nazareth College: Joe Arieno, Pat Assel, Richard Barth, Gina Cecala, Kristine Clauss, Sylvia Cooney, Brenda Dupee, Tim Freed, Lawana Jones, Wendy Marsden, Sabrina May, David Munson, Judy Olivieri, James Palamar, Kathy Pinckey, Jim Porter, Trisha Post, Dina Rice, Phyllis Roberts, Lori Schmidt, Teresa Snyder, and Keith Turner; and from Calvin College: Bruce Abernathy, Judy Arnett, John Brewer, Derek Brouwer, Bennet Bush, Rick Conklin, Joel DeBruin, Harmen DeJong, Deb DeRose, Alan DeVries, Dave Dorner, Dave Dreyer, Dan Fletcher, Kevin Hoag, Carl Hordyk, Steve Klaasen, Steve Kroese, Rich Manni, Richard McClain, Nancy Morrow, Paul Mulder, Jong Myung, Patrick Nagle, Joel Oakes, Stephern Pase, Priya Ramchandran, Debbie Smith, David Stevens, Rick Stiles, Sean Stroub, Alice VandeHeide, Jackie VandenBurg, Kevin VanderMeulen, Mark VanGorp, Ron Vanlwaarden, John Verbrugge, Ken VerHulst, Jonathan Youngsma, Brent Zomerlei, and Steve Zuidema.

In conclusion, I would like to thank the following reviewers for their valuable help and suggestions:

Robert Earles, St. Cloud University
Michael D. Grady, Loyola Marymount College
Denny Gulick, University of Maryland
Georgiana Klein, Grand Valley State College
Joseph B. Klerlein, Western Carolina University
Thomas Koshy, Framingham State University
Richard S. Palais, Brandeis University
Dana Richards, University of Virigina
Diane M. Spresser, James Madison University
Michael Stecher, Texas A&M University
Don Thompson, Pepperdine University
Keith Yale, University of Montana

Grand Rapids, Michigan J.B.

Contents

Discrete Models

1

Let us start with a problem. Suppose you are a traveling salesman[†] and there are five cities, including your own, that you would like to visit. Suppose also that the cities are called Amherst, Big Forks, Cattaragus, Devon, and Eden (*A, B, C, D,* and *E* for short). Figure 1.1 is a map of these cities. Starting from *A,* you would like to decide in what order to visit the cities so that the total distance (or time) you must travel to visit each city and return to *A* is minimal. What order should you follow? Before reading further, take a few minutes to see if you can figure out the answer.

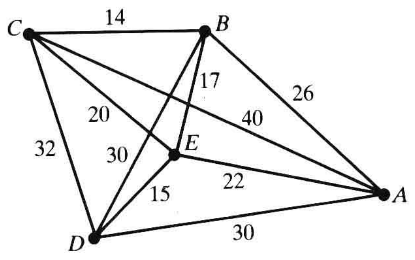


Figure 1.1 Map of five cities.

This problem and others like it have come to be called the "traveling salesman problem" and are among the most widely studied problems in contemporary mathematics and computer science. There are several solution techniques, but none that is altogether satisfactory.

The Brute Force Algorithm

The most obvious way to solve the problem is by a **brute force** technique:

1. List all of the possible routes.
2. Compute the length of each.
3. Select the shortest route.

Table 1.1 is a list of each of the routes starting at *A* and the distance associated with each. From this list, it is obvious that the best routes are *ABCEDA* and *ADECBA*. In fact, both routes are the same, merely traveled in opposite directions. So here is a method that works. But it is not a very good method because the process of listing all possible routes and computing the length of each is so time-consuming. With five cities there are 24 (i.e., $4 \cdot 3 \cdot 2 \cdot 1$) routes, starting at *A*. With six cities, there are 120 ($5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$), with seven cities 720 ($6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$), etc.

† Although "traveling salesperson" has the advantage of being sex-neutral, we will keep the traditional and less awkward term, "traveling salesman."

TABLE 1.1 Distances (in Miles) for All Routes in Figure 1.1 that Both Originate and Terminate at *A*

ABCDEA	109	*ADBCEA*	116
ABCEDA	105	*ADBECA*	137
ABDECA	131	*ADCBEA*	115
ABDCEA	130	*ADCEBA*	125
ABEDCA	130	*ADEBCA*	116
ABECDA	125	*ADECBA*	105
ACBDEA	121	*AEBCDA*	115
ACBEDA	116	*AEBDCA*	141
ACDBEA	141	*AECBDA*	116
ACDEBA	130	*AECDBA*	130
ACEBDA	137	*AEDBCA*	121
ACEDBA	131	*AEDCBA*	109

Suppose a very fast computer program could list a route and find its length in 10 microseconds (10^{-5} second). With 20 cities this approach would take about 38,573 years ($19 \cdot 18 \cdot 17 \ldots 3 \cdot 2 \cdot 1 \cdot 10^{-5}$ seconds) to find the shortest route. Adding one more city would multiply this time by 20. We describe this result by saying that the time it takes is proportional to $(n - 1)!$ ("$n - 1$ factorial," a condensed way of saying $(n - 1) \cdot (n - 2) \cdot (n - 3) \ldots 3 \cdot 2 \cdot 1$).

The Nearest Neighbor Algorithm

While brute force may be very effective in solving small problems, it is unworkable with larger ones. Two things should be obvious from the list of routes given in Table 1.1. First, every route was listed twice, for instance, *ABCDEA* and *AEDCBA*. So we could conceivably cut the processing time in half. This is helpful; it would reduce our computer program time by 19,287 years. However, it would still take 19,287 years to solve the problem! So we have to find something that will make a bigger improvement. Second, many of the routes are obviously inappropriate; for instance, going from A directly to C without going through E is only 2 miles less than going from A to E and then to C, which seems foolish. One method that holds out the hope of avoiding such foolish routes is this:

1. Start at A.
2. Go to A's nearest neighbor.
3. Move from there to its nearest neighbor that has not yet been visited.

4. Continue this process until all cities have been visited.
5. Return to A

This approach is sometimes called the **nearest neighbor algorithm** and is an example of a type of approach known as a "greedy algorithm"—get the best "deal" you can at each step but never look further ahead than that step. Apply it to our problem before reading further. You should get the sequence AEDBCA, with a total distance of 121 miles.

To determine how long this approach would take, we note that at each city the four other cities need to be checked to see which is closest and whether it has already been visited. So $5 \cdot 4$ checks need to be made. If each check takes 10^{-5} second, the time required is $20 \cdot 10^{-5}$ seconds. For 20 cities this would mean $20 \cdot 19 = 380$ checks. This comes to $380 \cdot 10^{-5}$ seconds, or $3.8 \cdot 10^{-3}$ seconds, far less than 38,573 years. With n cities, then, we say that the time required is proportional to $n \cdot (n - 1)$.

Thus the nearest neighbor algorithm does not give us the shortest route, but 121 miles may not be a bad solution. Also, this algorithm certainly doesn't take very long to perform. When compared to the optimal answer of 105 miles, it may be close enough to decide that finding the best solution isn't worth the extra effort.

Why Do We Need Mathematical Models?

Is there always a solution that allows us to visit each city only once? The answer is no; consider the map in Figure 1.2.

Sometimes there is a solution that visits each city only once and sometimes there is not. Is there, then, any way to tell from the problem itself when there is a solution that passes through each city only once? There are a couple of approaches one could use to answer this question. One is the method of an experimental scientist: Try out many different examples, note which have solutions that pass through each city only once and which do not, and try to observe some patterns common to one group of examples or another. This method is often very helpful and is the one mathema-

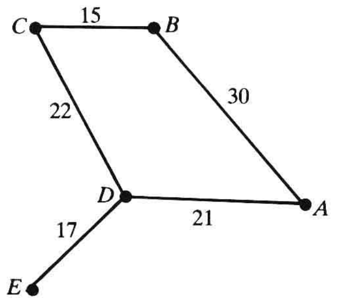


Figure 1.2 Map of five cities where not all are connected.