DISCRETE MATHEMATICS FOR COMPUTER SCIENTISTS

Clifford Stein
Columbia University

Robert L. Drysdale
Dartmouth College

Kenneth Bogart

Addison-Wesley

Boston Columbus Indianapolis New York San Francisco Upper Saddle River
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto
Delhi Mexico City Sao Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Editor in Chief:	Michael Hirsch
Editorial Assistant:	Stephanie Sellinger
Director of Marketing:	Margaret Whaples
Marketing Coordinator:	Kathryn Ferranti
Managing Editor:	Jeffrey Holcomb
Production Project Manager:	Heather McNally
Senior Manufacturing Buyer:	Carol Melville
Media Manufacturing Buyer:	Ginny Michaud
Art Director:	Linda Knowles
Cover Designer:	Elena Sidorova
Cover Art:	Veer
Media Project Manager:	Katelyn Boller
Full-Service Project Management:	Bruce Hobart, Laserwords
Composition:	Laserwords

Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear on appropriate page within text.

Library of Congress Cataloging-in-Publication Data available upon request

10 9 8 7 6 5 4 3 2 1—EB—14 13 12 11 10

Addison-Wesley
is an imprint of

PEARSON

www.pearsonhighered.com

ISBN-13: 978-0-13-212271-9

ISBN-10: 0-13-212271-5

This book is dedicated to our friend and co-author, Ken Bogart, whose untimely death on March 30, 2005 prevented him from seeing the original book in published form. Ken was the driving force behind the creation of the book. We miss him and we wish that we had been able to collaborate with him on this version.

Brief Contents

List of Theorems, Lemmas, and Corollaries xix

Preface xxi

CHAPTER 1 Counting 1

CHAPTER 2 Cryptography and Number Theory 59

CHAPTER 3 Reflections on Logic and Proof 117

CHAPTER 4 Induction, Recursion, and Recurrences 161

CHAPTER 5 Probability 249

CHAPTER 6 Graphs 359

APPENDIX A Derivation of the More General Master Theorem 449

APPENDIX B Answers and Hints to Selected Problems 461

Bibliography 477

Index 479

Contents

List of Theorems, Lemmas, and Corollaries xix

Preface xxi

CHAPTER 1 Counting 1

1.1 Basic Counting 1

The Sum Principle 1

Abstraction 3

Summing Consecutive Integers 3

The Product Principle 4

Two-Element Subsets 6

Important Concepts, Formulas, and Theorems 7

Problems 8

1.2 Counting Lists, Permutations, and Subsets 10

Using the Sum and Product Principles 10

Lists and Functions 12

The Bijection Principle 14

k-Element Permutations of a Set 15

Counting Subsets of a Set 16

Important Concepts, Formulas, and Theorems 18

Problems 20

1.3 Binomial Coefficients 22

Pascal's Triangle 22

A Proof Using the Sum Principle 24

The Binomial Theorem 26

Labeling and Trinomial Coefficients 28

Important Concepts, Formulas, and Theorems 29

Problems 30

1.4 Relations **32**
What Is a Relation? 32
Functions as Relations 33
Properties of Relations 33
Equivalence Relations 36
Partial and Total Orders 39
Important Concepts, Formulas, and Theorems 41
Problems 42

1.5 Using Equivalence Relations in Counting **43**
The Symmetry Principle 43
Equivalence Relations 45
The Quotient Principle 46
Equivalence Class Counting 46
Multisets 48
The Bookcase Arrangement Problem 50
The Number of k-Element Multisets of an n-Element Set 51
Using the Quotient Principle to Explain a Quotient 52
Important Concepts, Formulas, and Theorems 53
Problems 54

CHAPTER 2 Cryptography and Number Theory **59**

2.1 Cryptography and Modular Arithmetic **59**
Introduction to Cryptography 59
Private-Key Cryptography 60
Public-Key Cryptosystems 63
Arithmetic Modulo n 65
Cryptography Using Addition mod n 68
Cryptography Using Multiplication mod n 69
Important Concepts, Formulas, and Theorems 71
Problems 72

2.2 Inverses and Greatest Common Divisors 75
Solutions to Equations and Inverses mod *n* 75
Inverses mod *n* 76
Converting Modular Equations to Normal Equations 79
Greatest Common Divisors 80
Euclid's Division Theorem 81
Euclid's GCD Algorithm 84
Extended GCD Algorithm 85
Computing Inverses 88
Important Concepts, Formulas, and Theorems 89
Problems 90
2.3 The RSA Cryptosystem 93
Exponentiation mod *n* 93
The Rules of Exponents 93
Fermat's Little Theorem 96
The RSA Cryptosystem 97
The Chinese Remainder Theorem 101
Important Concepts, Formulas, and Theorems 102
Problems 104
2.4 Details of the RSA Cryptosystem 106
Practical Aspects of Exponentiation mod *n* 106
How Long Does It Take to Use the RSA Algorithm? 109
How Hard Is Factoring? 110
Finding Large Primes 110
Important Concepts, Formulas, and Theorems 113
Problems 114

CHAPTER 3 Reflections on Logic and Proof 117
3.1 Equivalence and Implication 117
Equivalence of Statements 117
Truth Tables 120
DeMorgan's Laws 123

Implication 125
If and Only If 126
Important Concepts, Formulas, and Theorems 129
Problems 131

3.2 Variables and Quantifiers 133
Variables and Universes 133
Quantifiers 134
Standard Notation for Quantification 136
Statements about Variables 138
Rewriting Statements to Encompass Larger Universes 138
Proving Quantified Statements True or False 139
Negation of Quantified Statements 140
Implicit Quantification 143
Proof of Quantified Statements 144
Important Concepts, Formulas, and Theorems 145
Problems 147

3.3 Inference 149
Direct Inference (Modus Ponens) and Proofs 149
Rules of Inference for Direct Proofs 151
Contrapositive Rule of Inference 153
Proof by Contradiction 155
Important Concepts, Formulas, and Theorems 158
Problems 159

CHAPTER 4 Induction, Recursion, and Recurrences 161

4.1 Mathematical Induction 161
Smallest Counterexamples 161
The Principle of Mathematical Induction 165
Strong Induction 169
Induction in General 171
A Recursive View of Induction 173

Structural Induction 176
Important Concepts, Formulas, and Theorems 178
Problems 180

4.2 Recursion, Recurrences, and Induction 183
Recursion 183
Examples of First-Order Linear Recurrences 185
Iterating a Recurrence 187
Geometric Series 188
First-Order Linear Recurrences 191
Important Concepts, Formulas, and Theorems 195
Problems 197

4.3 Growth Rates of Solutions to Recurrences 198
Divide and Conquer Algorithms 198
Recursion Trees 201
Three Different Behaviors 209
Important Concepts, Formulas, and Theorems 210
Problems 212

4.4 The Master Theorem 214
Master Theorem 214
Solving More General Kinds of Recurrences 217
Extending the Master Theorem 218
Important Concepts, Formulas, and Theorems 220
Problems 221

4.5 More General Kinds of Recurrences 222
Recurrence Inequalities 222
The Master Theorem for Inequalities 223
A Wrinkle with Induction 225
Further Wrinkles in Induction Proofs 227
Dealing with Functions Other Than n^c 230
Important Concepts, Formulas, and Theorems 232
Problems 233

4.6 Recurrences and Selection **235**

The Idea of Selection 235
A Recursive Selection Algorithm 236
Selection without Knowing the Median in Advance 237
An Algorithm to Find an Element in the Middle Half 239
An Analysis of the Revised Selection Algorithm 242
Uneven Divisions 244
Important Concepts, Formulas, and Theorems 246
Problems 247

CHAPTER 5 Probability **249**

5.1 Introduction to Probability **249**

Why Study Probability? 249
Some Examples of Probability Computations 252
Complementary Probabilities 253
Probability and Hashing 254
The Uniform Probability Distribution 256
Important Concepts, Formulas, and Theorems 259
Problems 260

5.2 Unions and Intersections **262**

The Probability of a Union of Events 262
Principle of Inclusion and Exclusion for Probability 265
The Principle of Inclusion and Exclusion for Counting 271
Important Concepts, Formulas, and Theorems 273
Problems 274

5.3 Conditional Probability and Independence **276**

Conditional Probability 276
Bayes' Theorem 280
Independence 280
Independent Trials Processes 282
Tree Diagrams 284
Primality Testing 288

Important Concepts, Formulas, and Theorems 289
Problems 290

5.4 Random Variables 292
What Are Random Variables? 292
Binomial Probabilities 293
A Taste of Generating Functions 295
Expected Value 296
Expected Values of Sums and Numerical Multiples 299
Indicator Random Variables 302
The Number of Trials until the First Success 304
Important Concepts, Formulas, and Theorems 306
Problems 307

5.5 Probability Calculations in Hashing 310
Expected Number of Items per Location 310
Expected Number of Empty Locations 311
Expected Number of Collisions 312
Expected Maximum Number of Elements in a Location of a Hash Table 315
Important Concepts, Formulas, and Theorems 320
Problems 321

5.6 Conditional Expectations, Recurrences, and Algorithms 325
When Running Times Depend on More than Size of Inputs 325
Conditional Expected Values 327
Randomized Algorithms 329
Selection Revisited 331
QuickSort 333
A More Careful Analysis of RandomSelect 336
Important Concepts, Formulas, and Theorems 339
Problems 340

5.7 Probability Distributions and Variance **343**
Distributions of Random Variables 343
Variance 346
Important Concepts, Formulas, and Theorems 354
Problems 355

CHAPTER 6 Graphs **359**
6.1 Graphs **359**
The Degree of a Vertex 363
Connectivity 365
Cycles 367
Trees 368
Other Properties of Trees 368
Important Concepts, Formulas, and Theorems 371
Problems 373
6.2 Spanning Trees and Rooted Trees **375**
Spanning Trees 375
Breadth-First Search 377
Rooted Trees 382
Important Concepts, Formulas, and Theorems 386
Problems 387
6.3 Eulerian and Hamiltonian Graphs **389**
Eulerian Tours and Trails 389
Finding Eulerian Tours 394
Hamiltonian Paths and Cycles 395
NP-Complete Problems 401
Proving That Problems Are NP-Complete 403
Important Concepts, Formulas, and Theorems 406
Problems 407
6.4 Matching Theory **410**
The Idea of a Matching 410
Making Matchings Bigger 414

Matching in Bipartite Graphs 417
Searching for Augmenting Paths in Bipartite Graphs 417
The Augmentation-Cover Algorithm 420
Efficient Algorithms 426
Important Concepts, Formulas, and Theorems 427
Problems 428

6.5 Coloring and Planarity 430
The Idea of Coloring 430
Interval Graphs 433
Planarity 435
The Faces of a Planar Drawing 437
The Five-Color Theorem 441
Important Concepts, Formulas, and Theorems 444
Problems 445

APPENDIX A Derivation of the More General Master Theorem 449
More General Recurrences 449
Recurrences for General *n* 451
Removing Floors and Ceilings 452
Floors and Ceilings in the Stronger Version of the Master Theorem 453
Proofs of Theorems 453
Important Concepts, Formulas, and Theorems 457
Problems 458

APPENDIX B Answers and Hints to Selected Problems 461

Bibliography 477

Index 479

List of Theorems, Lemmas, and Corollaries

Chapter 1

Theorem 1.1 16
Theorem 1.2 18
Theorem 1.3 25
Theorem 1.4 26
Theorem 1.5 38
Theorem 1.6 39
Theorem 1.7 46
Theorem 1.8 52

Chapter 2

Theorem 2.1 61
Theorem 2.4 67
Theorem 2.7 78
Theorem 2.9 80
Theorem 2.12 82
Theorem 2.14 88
Theorem 2.15 88
Theorem 2.21 96
Theorem 2.23 101
Theorem 2.24 101

Lemma 2.2 65
Lemma 2.3 66
Lemma 2.5 75
Lemma 2.8 79
Lemma 2.11 81
Lemma 2.13 83
Lemma 2.19 93
Lemma 2.20 96
Lemma 2.25 111

Corollary 2.6 77
Corollary 2.10 80
Corollary 2.16 88
Corollary 2.17 88
Corollary 2.18 89
Corollary 2.22 97

Chapter 3

Theorem 3.2 139
Theorem 3.3 141

Lemma 3.1 123
Lemma 3.5 154

Corollary 3.4 141

Chapter 4

Theorem 4.1 188
Theorem 4.4 191
Theorem 4.5 192
Theorem 4.6 195
Theorem 4.9 215
Theorem 4.10 219
Theorem 4.11 220
Theorem 4.12 224
Theorem 4.15 244

Lemma 4.3 190
Lemma 4.7 209
Lemma 4.14 242

Corollary 4.2 189
Corollary 4.8 210
Corollary 4.13 224

Chapter 5

Theorem 5.1 253
Theorem 5.2 256
Theorem 5.3 266

Theorem 5.4 272
Theorem 5.5 280
Theorem 5.7 282
Theorem 5.8 294
Theorem 5.10 300
Theorem 5.11 300
Theorem 5.12 301
Theorem 5.13 305
Theorem 5.14 311
Theorem 5.15 311
Theorem 5.16 312
Theorem 5.17 314
Theorem 5.22 319
Theorem 5.23 329
Theorem 5.24 333
Theorem 5.25 336
Theorem 5.29 350
Theorem 5.30 353

Lemma 5.9 298
Lemma 5.18 315
Lemma 5.19 316
Lemma 5.20 317
Lemma 5.21 318
Lemma 5.26 346
Lemma 5.28 349

Corollary 5.6 281
Corollary 5.27 346

Chapter 6

Theorem 6.2 363
Theorem 6.3 369
Theorem 6.5 371
Theorem 6.7 375
Theorem 6.9 381
Theorem 6.10 392
Theorem 6.11 393
Theorem 6.12 398
Theorem 6.13 400
Theorem 6.18 417
Theorem 6.20 424
Theorem 6.22 425
Theorem 6.24 434
Theorem 6.26 439
Theorem 6.29 441

Lemma 6.1 363
Lemma 6.4 370
Lemma 6.8 380
Lemma 6.14 413
Lemma 6.15 413
Lemma 6.16 415
Lemma 6.23 432

Corollary 6.6 371
Corollary 6.17 416
Corollary 6.19 417
Corollary 6.21 425
Corollary 6.25 435
Corollary 6.27 440
Corollary 6.28 440

Appendix A

Theorem A.1 451
Theorem A.2 452
Theorem A.3 453
Theorem A.4 454
Theorem A.5 455

Preface

OUR MOTIVATION AND VISION

Many colleges and universities offer a course in discrete mathematics. Students taking these courses are from many disciplines, one of the largest being computer science. As a part of the Mathematics Across the Curriculum project at Dartmouth, supported by the National Science Foundation,[1] we proposed to create a discrete mathematics course that directly addresses the needs of computer science students. In analyzing what topics in discrete mathematics we want our computer science students to know and why we want them to know these topics, we made two observations.

First, there are a few topics we consider important to computer science that are not always covered thoroughly, if at all, in traditional discrete mathematics courses. Among these are recursion trees and the Master Theorem for solving recurrence relations, the probability theory needed to compute average run times and to analyze randomized algorithms, and an emphasis on strong and structural induction.

Second, for each topic in discrete mathematics that we consider important for computer science students, there is a motivating topic in computer science that can be understood at the level of a first or second course in computer science. We feel this makes it possible to answer the age-old question students ask in applied mathematics courses: "Why do we have to learn this?" We therefore chose to write a textbook with computer science students in mind, with the objective of providing the necessary mathematical foundations for a computer science major, motivated by computer science problems that students can understand early in their study of computer science.

In many computer science departments, discrete mathematics is one of the first courses taken by majors. It may even be a prerequisite to the first computer science course. In this case instructors are faced with a dilemma—teach the concepts purely mathematically with little or no visible application to computer science, or teach computer science examples to create a context

[1]Grant Number DUE-9552462

relevant to computer science students. The first leaves students complaining that they are being forced to take too much "irrelevant" mathematics before they can take their first computer science course. The second leaves professors (who are often mathematicians) trying to explain fairly advanced computer science topics such as hashing, binary trees, and recursive programs to students who may never have written a program. Even under the best of circumstances, this approach significantly reduces the depth of the mathematics that can be taught. Our analysis led to a different approach, creating a course that appears slightly later in students' studies. While we do not explicitly assume students have taken calculus, we assume familiarity with and make significant use of summation notation, logarithms, and exponential functions, so that a strong understanding of precalculus material is very helpful.[2] This course is meant to be taken after an introductory computer science course where students have seen recursive programs. Ideally it would be taken concurrently with or after a data structures course, but we explain the data structures that we use as examples. Therefore a data structures course is not a prerequisite for this course.

We feel that there are several advantages to this placement. Particular examples include:

- Students have already had serious experience thinking about problem solving, algorithms, and writing code.
- Students have learned or are ready to learn several important computer science concepts such as hashing, recursion, sorting and searching, and basic data structures.
- Students know enough computer science that they already know the motivating examples or the examples are straightforward enough for them to understand. For example,
  - Hashing can be used to motivate the study of probability.
  - Analysis of recursive programs such as mergesort and quicksort can be used to motivate recurrence relations and their solutions.
  - Analyzing how often we expect to find a new minimum in a procedure that finds the minimum element of a list can be used to motivate studying linearity of expectation and harmonic numbers.

[2]Most of our students have had calculus. In isolated places we make use of elementary derivatives and in optional subsections in probability we use natural logarithm and exponential functions and elementary power series. By ignoring the few proofs or problems using derivatives and the optional subsections in probability, the instructor can avoid calculus.

– Binary trees can be used to teach structural induction and also to motivate the study of trees as examples of graphs.

In our own teaching experiences this class is a prerequisite to an algorithms class and students often take the algorithms course soon after the discrete mathematics course. In doing so, they find themselves immediately using the mathematics that they have just learned.

OUR EDUCATIONAL PHILOSOPHY

This text is driven by activities, presented as exercises. The material is fleshed out through explanations and extensions of the activities. The most effective way for students to use the book is for them to attempt seriously the student activities before they read the explanation that follows. The activities are primarily meant to be done in groups in class; thus for activities done out of class we recommend that students form groups to work together. The class and this text are designed in this way to help students develop their own habits of mathematical thought. Our reading of the research in how undergraduate students learn mathematics leads us to several conclusions.

- Students who actively discover what they are learning (thus engaging in what is often called "active learning") remember concepts far longer than those who don't. They are also more likely to be able to use them outside the context in which they were learned.
- Students are more likely to ask questions until they understand a subject when they are working in a small group of peers rather than in a larger class with an instructor. (However, this isn't always the case. Many students need to feel comfortable within their group before they ask questions that they fear will slow down the others. We try to develop this comfort level in class by allowing students to choose their groups and change from group to group on different days as attendance patterns allow or require.)
- Finally, explaining ideas to someone else helps students organize these ideas in their minds and familiarizes students with the language of mathematics.

There is ample material in the book for a four-semester-hour course. At Dartmouth we use the book for a fast-paced course that meets three days a week for just over nine weeks and covers all but the last few sections of the book and some material marked with an asterisk.

THE ROLE OF PROOFS

One of our purposes in writing this book is to give students a background for the kinds of proofs that they will need to understand and write in their computer science courses. Our view is that one learns how to do proofs by hearing, seeing, discussing, and trying to do proofs. In order to discuss proofs, we need to have a common language that classifies the ingredients of a proof and provides us a framework for discussion. For this reason we have included a chapter on logic, designed both to give students this language and to assist them in the process of reflecting on the proofs they have already seen. In order to have something significant to talk about in this chapter, we have introduced it after the students have seen some combinatorial and number theoretic proofs. This way the students have concrete examples of proofs that can be used to illustrate the logical abstractions. We realize that this is not the usual order in a discrete mathematics book. However, we find that dealing with concrete examples of proofs of non-trivial facts gives some grounding to what otherwise can seem to be a long list of formal rules of inference.

We have placed the chapter on logic before the chapter on mathematical induction so that we can use its language in discussing and reflecting on mathematical induction.

MATHEMATICAL INDUCTION

Inductive proofs in computer science frequently use subproblems that are not "one smaller." We therefore emphasize strong induction as well as weak induction. We also introduce structural induction on trees and graphs. We try to use students' experiences with recursion to help them understand induction and develop inductive proofs. In particular, when creating an inductive proof it is usually more profitable to start with a big problem and recursively subdivide it into smaller problems than to start with small problems and try to "build up" to bigger problems.

THE USE OF PSEUDOCODE

We describe algorithms both in prose and by using pseudocode. The pseudocode should be easily readable by any one who has programmed in Java™, C, or C++ and should be understandable to people who have programmed in other languages. We do not strive to give syntactically correct

code in any language, rather we strive for clarity. For example, to say, "Swap the values held in variables x and y," we write, "exchange x and y" rather than writing three lines of code. Similarly, we write, "if points i, j, and k are not collinear," without concern for how a more detailed computation proceeds. Here are some particular conventions we use in the text.

- Blocks of code are denoted by indentation. There is no begin, end or "{" "}" as in many languages.
- For loops are written as "for $i = 1$ to n" to denote that the variable i ranges from 1 to n.
- While loop bodies are repeatedly executed while the boolean expression after the while is true.
- Repeat loops have the form: "repeat . . . until". The code between the repeat and until is executed at least once, and is repeatedly executed until the boolean expression after the until is true.
- If statements have one of the following forms:
  - if (*expression*) *block of code*
  - if (*expression*) *block1 of code* else *block2 of code*

  In the first form, *block of code* is executed if and only if the expression is true. In the second form, *block1* is executed if the expression is true and *block2* is executed if the expression is false.
- Arrays are subscripted using "[ ]."
- Assignment is represented with "=" and comparison for equality with "==".
- Shorthand for incrementing and decrementing x is "x ++" and "x −−."
- The logical operator "not" is indicated by "!," so "!true" is "false," and $!(x < y)$ is true when x is not less than y. Logical "and" is indicated by "&&" and logical "or" by "| |"

WHAT HAS CHANGED IN THIS VERSION OF THE BOOK

A different book with a similar title and the same set of authors was published by a different publisher who has since withdrawn from the college textbook market. Ken Bogart, the lead author on that book, died shortly before it came out. We greatly miss his participation in preparing this revised book.

The most significant changes between the former book and this one are:

- The previous book discussed equivalence relations, but only as partitions of a set. The reflexive, symmetric, and transitive properties were relegated to an appendix, and partial orders and total orders were not discussed. This book introduces relations as a concept that connects functions, equivalence relations, partial orders, and total orders. It shows why reflexivity, symmetry, and transitivity lead to equivalence relations and reflexivity, antisymmetry, and transitivity lead to partial orders.
- This book includes structural induction. Also, the section on the relationship between recursion and induction has been expanded and uses some different examples.
- Some sections in the chapter on recurrence relations have been removed or moved to an appendix. These sections showed that eliminating floors and ceilings in recurrences and extending the domain of the relation to numbers other than the powers of a base do not invalidate the Master Theorem. We decided that they interfered with the flow of the chapter and dealt with picky details that most students at this level do not need to know.
- Bayes' Theorem was added to the section on conditional probability.
- Problems were added to cover the new topics.

There are also a number of smaller changes (e.g., introducing the "multiply by x and subtract" approach to getting a closed form for geometric series).

INSTRUCTORS' SUPPLEMENTS

The following supplemental material is available to qualified instructors only. Please visit the Instructor Resource Center (www.pearsonhighered.com/irc), contact your local Addison-Wesley/Pearson Education sales representative, or send email to computing@aw.com for information about how to access them.

- Instructor's Manual with Solutions
  - Teaching suggestions
  - Solutions to homework problems
  - Exercise handouts for use in class
  - Detailed discussion of how we have students work on exercises in groups in class to stimulate discussion
- PowerPoint Presentations

ACKNOWLEDGMENTS

A number of people contributed to the original version of this book. We would like to thank Eddie Cheng, Oakland University; Alice Dean, Skidmore College; Ruth Hass, Smith College; and Italo Dejter, University of Puerto Rico for their thoughtful review comments on an early version of the manuscript. As the book was being developed, preliminary versions were used to teach discrete mathematics at Dartmouth by the authors and by Neal Young, Prasad Jayanti, Tom Shemanske, Rosa Orellana, April Rasala, Amit Chakrabarti, and Carl Pomerance. Each of them had an impact on the final product, some very substantial, and we thank them for their advice. We offer a special thanks to Carl Pomerance for his thorough and insightful commentary as he taught the course. Qun Li was a graduate assistant to us as we were initially preparing the manuscript, and he had the job of making sure that the problems we created really did have solutions! His work still forms the core of the solutions available to the instructor. The graduate teaching assistants from the Computer Science and Mathematics Departments while we and others taught from the manuscript also gave us valuable insights into what students were and were not learning and further help with solutions to problems. In order of their service, they were S. Agrawal, Elishiva Werner-Reiss, Robert Savell, Virgiliu Pavlu, Libo Song, Geeta Chaudhry, King Tan, Yurong Xu, Gabriella Dumitrascu, Florin Constantin, Alin Popescu, and Wei Zhang. Our students over the years have provided us with valuable feedback. In particular, Eric Robinson carefully read a near-final version, looking explicitly for passages that were hard to understand.

We would also like to thank the people who made this version of the book possible. The following reviewers provided many thoughtful suggestions: Michael Rothstein, Kent State University; Ravi Janardan, University of Minnesota, Twin Cities; Klaus Sutner, Carnegie Mellon University; Doug Baldwin, SUNY Genesco; Stuart Reges, University of Washington; Richard Anderson, University of Washington; Jonathan Goldstine, Penn State University. Sandra Hakanson, a Pearson sales representative, first suggested that the Addison-Wesley division of Pearson might be interested in the book. She put us in touch with Michael Hirsch, Editor-in-Chief, who agreed to publish the book and has shepherded it through the publication process. Many of the improvements that were made were his suggestions. Others at Addison-Wesley who contributed directly to the publication are Stephanie Sellinger (Editorial Assistant), Jeff Holcomb (Managing Editor), Heather McNally (Project Editor), and Elena Sidorova (Cover Designer). Bruce Hobart at

Laserwords was in charge of ushering the manuscript through copyediting, composition, and proofreading.

Each of the authors would like to thank the other two for the time they have taken from other professional activities to work on this project. Because of the time required to meld the points of view of our disciplines, it was only with the support of the National Science Foundation (Grant DUE-9552462) that we were able to undertake this project. We believe that the staff of the Division of Undergraduate Education showed excellent insight into the needs of undergraduates and the difficulties of interdisciplinary curricular development when they conceived their program of Mathematical Sciences and their applications throughout the curriculum. We would like to acknowledge the positive impact this program had on undergraduate mathematics education and on the development of interdisciplinary collaborations in curriculum development.

Cliff Stein
Scot Drysdale

DISCRETE MATHEMATICS FOR COMPUTER SCIENTISTS