Undergraduate Topics in Computer Science

Series editor

Ian Mackie, Department of Informatics, University of Sussex, Palaiseau, France

Advisory editors

Samson Abramsky, Department of Computer Science, University of Oxford, Oxford, UK
Chris Hankin, Department of Computer Science, Imperial College London, London, UK
Dexter C. Kozen, Cornell University, Ithaca, USA
Andrew Pitts, Department of Computer Science, University of Cambridge, Cambridge, UK
Hanne Riis Nielson, Department of Informatics, Technical University of Denmark, Kongens Lyngby, Denmark
Steven S. Skiena, Department of Computer Science, Stony Brook University, Stony Brook, USA
Iain Stewart, Department of Computer Science, University of Durham, Durham, UK
Mike Hinchey, Lero, Tierney Building, University of Limerick, Limerick, Ireland

Undergraduate Topics in Computer Science (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems. Many include fully worked solutions.

More information about this series at http://www.springer.com/series/7592

Tom Jenkyns · Ben Stephenson

Fundamentals of Discrete Math for Computer Science

A Problem-Solving Primer

Second Edition

Springer

Tom Jenkyns
Brock University
St. Catharines, ON
Canada

Ben Stephenson
University of Calgary
Calgary, AB
Canada

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-319-70150-9 ISBN 978-3-319-70151-6 (eBook)
https://doi.org/10.1007/978-3-319-70151-6

Library of Congress Control Number: 2017959909

1st edition: © Springer-Verlag London 2013
2nd edition: © Springer International Publishing AG, part of Springer Nature 2018
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

While this is a new edition of Fundamentals of Discrete Math for Computer Science, the goal of the book remains the same: To present discrete mathematics to computer science students in a form that is accessible to them, and in a way that will improve their programming competence. This edition improves upon its predecessor by introducing a new chapter on directed graphs, introducing a new section on drawing and coloring graphs, adding more than 100 new exercises, and providing solutions to selected exercises. We have also made numerous minor modifications in the second edition that make the text easier to read, improve clarity, or correct errata.

Most chapters begin with an example that sets the stage for its content. Chapter 1 begins by setting the stage for the whole book: How do you design algorithms to solve computing problems? How do you know your algorithm will work correctly for every suitable input? How long will your algorithm take to generate its output?

In our view, teaching is much more than presenting content, and we have written this text as *a design for the experience of students with our subject.* A students' first experience with a new subject can have a long-lasting impact on their perception it. We want that experience to be positive. Our text empowers students to think critically, to be effective problem solvers, to integrate theory and practice, and to recognize the importance of abstraction. It engages them with memorable, motivating examples. It challenges them with many new ideas, methods, and rigorous thinking, and we hope it entertains them like no other textbook with "Math" in its title.

This book introduces much of the "culture" of Computer Science and the common knowledge shared by all computer scientists (beyond programming). Much of it is devoted to the solutions to fundamental problems that all computer scientists have studied: how to search a list for a particular target; how to sort a list into a natural order; how to generate all objects, subsets, or sequences of some kind in such an order; how to traverse all of the nodes in a graph or digraph; and especially, how to compare the efficiency of algorithms and prove their correctness. Our constant theme is the relevance of the mathematics we present to Computer Science.

Perhaps the most distinguishing feature of this text is its informal and interactive nature. Detailed walkthroughs of algorithms appear from beginning to end. We motivate the material by inserting provocative questions and commentary into the prose, and we simulate a conversation with our reader, more like our lectures and less like other pedantic and ponderous mathematics texts. We employ the symbol "//" to denote "*comments*", to indicate *asides* (especially expanding and explaining mathematical arguments), as *signals* of what comes next, and to *prompt* questions we want to raise in the reader's mind. We've kept the text (words, sentences, and paragraphs) short and to the point, and we've used font changes, bold, italic and underlining to **capture**, *focus* and <u>recapture</u> our reader's attention.

But this book is a *mathematics text*. We expand on students' intuition with precise mathematical language and ideas. Although detailed proofs are delayed until Chap. 3, the fundamental nature of proof in mathematics is explained, maintained, and applied repeatedly to proving the correctness of algorithms. One of the purposes of the text is to provide a toolbox of useful algorithms solving standard problems in Computer Science. The other purpose is to provide a catalogue of useful concepts without which the underlying theory of Computer Science cannot be understood.

St. Catharines, Canada Tom Jenkyns
Calgary, Canada Ben Stephenson

Acknowledgements

We wish to acknowledge the support and encouragement of our friends and families throughout the long gestation of this book; especially Eleanor, Glenys, Janice, Flora and Jonathan. We are also grateful to the students and teachers that made this book possible, including the students at Brock University who indulged experimentation in both the form and content of this project over two decades of “class-testing”, and every student and teacher that adopted, recommended, read or worked their way through the first edition. We also want to thank everyone who provided helpful comments and suggestions based on their experience using the first edition, and particularly acknowledge the comments submitted by Dominic Magnarelli and Carlo Tomasi. Finally, we are indebted to Eric R. Muller who initiated the prototype course at Brock University.

Contents

1 Algorithms, Numbers, and Machines 1
- 1.1 What Is an Algorithm? 4
- 1.2 Integer Algorithms and Complexity 8
  - 1.2.1 Prime Testing 9
  - 1.2.2 Real Numbers 11
  - 1.2.3 More Prime Testing 12
  - 1.2.4 Prime Factorization 14
  - 1.2.5 Logarithms 16
  - 1.2.6 Greatest Common Divisor 18
- 1.3 Machine Representation of Numbers 21
  - 1.3.1 Approximation Errors 23
  - 1.3.2 Base 2, 8, and 16 26
- 1.4 Numerical Solutions 35
  - 1.4.1 Newton's Method for Square Roots 35
  - 1.4.2 The Bisection Algorithm 37
- Exercises 41

2 Sets, Sequences, and Counting 45
- 2.1 Naïve Set Theory 45
  - 2.1.1 The Diabolical Librarian 48
  - 2.1.2 Operations on Sets and Cardinality 49
  - 2.1.3 The Pigeonhole Principle 51
- 2.2 Sequences 53
  - 2.2.1 The Characteristic Sequence of a Subset 55
- 2.3 Counting 56
  - 2.3.1 Number of *k*-Sequences on an *n*-Set 57
  - 2.3.2 Number of Subsets of an *n*-Set 58
  - 2.3.3 Number of *k*-Permutations on an *n*-Set 58
  - 2.3.4 *n*-Factorial 59
  - 2.3.5 Number of *k*-Subsets of an *n*-Set 60
  - 2.3.6 Pascal's Triangle 63
  - 2.3.7 Counting Algorithmically (Without a Formula) 66

2.4 Infinite Sequences and Complexity Functions . . . 69
2.4.1 The Towers of Hanoi . . . 72
2.4.2 Bad Complexity Functions . . . 75
Exercises . . . 76

3 Boolean Expressions, Logic, and Proof . . . 81
3.1 The Greedy Algorithm and Three Cookie Problems . . . 81
3.1.1 The Greedy Algorithm . . . 82
3.2 Boolean Expressions and Truth Tables . . . 86
3.2.1 The Negation Operator . . . 86
3.2.2 The Conjunction Operator . . . 86
3.2.3 The Disjunction Operator . . . 87
3.2.4 The Conditional Operator . . . 89
3.2.5 The Biconditional Operator . . . 91
3.3 Predicates and Quantifiers . . . 92
3.4 Valid Arguments . . . 93
3.5 Examples of Proofs . . . 97
3.5.1 Direct Proof . . . 100
3.5.2 Indirect Proof . . . 101
3.5.3 Cantor's Diagonalization Process . . . 104
3.6 Mathematical Induction . . . 106
3.6.1 Strong Induction . . . 117
3.7 Proofs Promised in Chap. 1 . . . 119
3.7.1 Russian Peasant Multiplication Is Correct . . . 119
3.7.2 Resolving the Cake Cutting Conundrum . . . 121
3.7.3 Casting Out Nines . . . 123
3.7.4 Euclid's Algorithm for GCD Is Correct . . . 125
3.8 The Proof Promised in Chap. 2 . . . 128
Exercises . . . 130

4 Searching and Sorting . . . 137
4.1 Searching . . . 137
4.1.1 Searching an Arbitrary List . . . 137
4.1.2 Searching a Sorted List . . . 139
4.2 Branching Diagrams . . . 145
4.2.1 A Second Version of Binary Search . . . 145
4.3 Sorting . . . 153
4.3.1 Selection Sorts . . . 153
4.3.2 Exchange Sorts . . . 156
4.4 Binary Trees with (at Least) $n!$ Leaves . . . 163
4.5 Partition Sorts . . . 171
4.6 Comparison of Sorting Algorithms . . . 184
4.6.1 Timings and Operation Counts . . . 184
Exercises . . . 185

5 Graphs and Trees 191
5.1 Introduction 191
5.1.1 Degrees 196
5.1.2 Eulerian Graphs 196
5.1.3 Hamiltonian Graphs 197
5.2 Paths, Circuits, and Polygons 198
5.2.1 Subgraphs Determined by Paths 200
5.3 Trees 202
5.3.1 Traversals 203
5.4 Edge-Weighted Graphs 217
5.4.1 Shortest Paths 221
5.5 Drawing and Coloring 222
5.5.1 Bipartite Graphs 223
5.5.2 Planar Graphs 225
5.5.3 Some History of the Four Color Theorem 232
Exercises 233

6 Directed Graphs 241
6.1 Introducing Directions 241
6.2 Strong Connectivity 243
6.3 Topological Sorting 247
6.4 Shortest Paths in Digraphs (Acyclic or not) 258
6.4.1 Distance Function 259
6.4.2 Dijkstra's Algorithm 259
6.4.3 Floyd-Warshall Algorithm 267
6.5 The Maximum Flow Problem 272
6.6 Matchings in Bipartite Graphs 284
Exercises 291

7 Relations: Especially on (Integer) Sequences 299
7.1 Relations and Representations 299
7.1.1 Matrix Representation 300
7.1.2 Directed Graph Representation 301
7.1.3 Properties of Relations 301
7.2 Equivalence Relations 302
7.2.1 Matrix and Digraph of an Equivalence Relation 303
7.3 Order Relations 306
7.3.1 Matrix and Digraph of a Partial Order 307
7.3.2 Minimal and Maximal Elements 308
7.4 Relations on Finite Sequences 311
7.4.1 Domination 311
7.4.2 Lexicographic Order 313

7.5 Relations on Infinite Sequences 315
7.5.1 Asymptotic Dominance and Big-Oh Notation 316
7.5.2 Asymptotic Equivalence and Big-Theta Notation 322
7.5.3 Asymptotic Ranking 325
7.5.4 Strong Asymptotic Dominance and Little-Oh Notation. 326
Exercises. 328

8 Sequences and Series. 333
8.1 Examples Defined by Recurrence Equations. 334
8.2 Solving First-Order Linear Recurrence Equations. 340
8.3 The Fibonacci Sequence. 346
8.3.1 Algorithms for the Fibonacci Sequence. 348
8.3.2 The Golden Ratio 350
8.3.3 The Fibonacci Sequence and the Golden Ratio 351
8.3.4 The Order of the Fibonacci Sequence 354
8.3.5 The Complexity of Euclid's Algorithm for GCD. 355
8.4 Solving Second-Order Linear Recurrence Equations. 358
8.5 Infinite Series 366
8.5.1 Zeno's Paradoxes 366
8.5.2 Formal Definitions of Convergence of Sequences and Series 367
Exercises. 373

9 Generating Sequences and Subsets. 379
9.1 Generating Sequences in Lexicographic-Order 381
9.2 Generating All *k*-Sequences on $\{1..n\}$ 384
9.2.1 Average-Case Complexity. 384
9.3 Generating Subsets of $\{1..n\}$ as Increasing Sequences 388
9.4 Generating Permutations in Lexicographic-Order 398
9.4.1 Generating All *k*-Permutations of $\{1..n\}$ in Lex-Order 407
Exercises. 412

10 Discrete Probability and Average-Case Complexity 421
10.1 Probabilistic Models. 421
10.1.1 Sample Spaces. 422
10.1.2 Probability Functions. 422
10.1.3 The Special Case of Equally Likely Outcomes 424
10.2 Conditional Probability. 427
10.2.1 Combinations of Events. 428
10.2.2 Conditional Probability 428
10.2.3 Independent Events 430
10.2.4 Mutually Exclusive Events 430

10.3 Random Variables and Expected Values. 435
10.3.1 Expected Frequency 436
10.3.2 Expected Values 437
10.3.3 Probability Distributions 438
10.4 Standard Distributions and Their Expected Values 439
10.4.1 The Uniform Distribution 439
10.4.2 The Binomial Distribution. 443
10.4.3 The Geometric Distribution 444
10.5 Conditional Expected Values 447
10.5.1 Conditional Expectation. 452
10.6 Average-Case Complexity 454
10.6.1 Applying Expectation to Linear Search 454
10.6.2 Applying Expectation to QuickSort 455
Exercises 460

11 Turing Machines 467
11.1 What Is an Algorithm? 468
11.1.1 The Church-Turing Thesis. 475
11.1.2 Universal Turing Machine: As a Computational Model 476
11.1.3 The Halting Problem 476
Exercises 479

Appendix A: Solutions to Selected Exercises. 483

Index 509