Michael T. Goodrich Roberto Tamassia

Algorithm Design and Applications

WILEY

Algorithm Design and Applications

Michael T. Goodrich
Department of Information and Computer Science
University of California, Irvine

Roberto Tamassia
Department of Computer Science
Brown University

WILEY

VICE PRESIDENT & PUBLISHER	Don Fowley
EXECUTIVE EDITOR	Beth Lang Golub
EDITORIAL ASSISTANT	Jayne Ziemba
ASSISTANT MARKETING MANAGER	Debbie Martin
ASSOCIATE PRODUCTION MANAGER	Joyce Poh
PRODUCTION EDITOR	Wanqian Ye
COVER CREDIT	© Ansel Adams/U.S. National Archives and Records Administration

This book was set by the authors and printed and bound by Courier Kendallville.

This book is printed on acid free paper. ∞

Founded in 1807, John Wiley & Sons, Inc. has been a valued source of knowledge and understanding for more than 200 years, helping people around the world meet their needs and fulfill their aspirations. Our company is built on a foundation of principles that include responsibility to the communities we serve and where we live and work. In 2008, we launched a Corporate Citizenship Initiative, a global effort to address the environmental, social, economic, and ethical challenges we face in our business. Among the issues we are addressing are carbon impact, paper specifications and procurement, ethical conduct within our business and among our vendors, and community and charitable support. For more information, please visit our website: www.wiley.com/go/citizenship.

Library of Congress Cataloging in Publication Data:

Goodrich, Michael T., author.
Algorithm design and applications / Michael T. Goodrich, Department of Information and Computer Science, University of California, Irvine, Roberto Tamassia, Department of Computer Science, Brown University.
pages cm
Includes bibliographical references and index.
ISBN 978-1-118-33591-8 (hardback)
1. Computer algorithms. 2. Data structures (Computer science) I. Tamassia, Roberto, 1960- author. II. Title.
QA76.9.A43G668 2014
005.7'3--dc23

2014021534

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To Karen, Paul, Anna, and Jack
– *Michael T. Goodrich*

To Isabel
– *Roberto Tamassia*

Contents

Part V: Computational Intractability

Part VI: Additional Topics

Preface

This book is designed to provide a comprehensive introduction to the design and analysis of computer algorithms and data structures. We have made each chapter to be relatively independent of other chapters so as to provide instructors and readers greater flexibility with respect to which chapters to explore. Moreover, the extensive collection of topics we include provides coverage of both classic and emerging algorithmic methods, including the following:

- ***Mathematics for asymptotic analysis***, including amortization and randomization
- ***General algorithm design techniques***, including the greedy method, divide-and-conquer, and dynamic programming
- ***Data structures***, including lists, trees, heaps, search trees, B-trees, hash tables, skip lists, union-find structures, and multidimensional trees
- ***Algorithmic frameworks***, including ***NP***-completeness, approximation algorithms, and external-memory algorithms
- ***Fundamental algorithms***, including sorting, graph algorithms, computational geometry, numerical algorithms, cryptography, Fast Fourier Transform (FFT), and linear programming.

Application-Motivated Approach

This is an exciting time for computer science. Computers have moved beyond their early uses as computational engines to now be used as information processors, with applications to every other discipline. Moreover, the expansion of the Internet has brought about new paradigms and modalities for computer applications to society and commerce. For instance, computers can be used to store and retrieve large amounts of data, and they are used in many other application areas, such as sports, video games, biology, medicine, social networking, engineering, and science. Thus, we feel that algorithms should be taught to emphasize not only their mathematical analysis but also their practical applications.

To fulfill this need, we have written each chapter to begin with a brief discussion of an application that motivates the topic of that chapter. In some cases, this application comes from a real-world use of the topic discussed in the chapter, and in other cases it is a contrived application that highlights how the topic of the chapter could be used in practice. Our intent in providing this motivation is to give readers a conceptual context and practical justification to accompany their reading of each chapter. In addition to this application-based motivation we include also detailed pseudocode descriptions and complete mathematical analysis. Indeed, we feel that mathematical rigor should not simply be for its own sake, but also for its pragmatic implications.

For the Instructor

This book is structured to allow an instructor a great deal of freedom in how to organize and present material. The dependence between chapters is relatively minimal, which allows the instructor to cover topics in her preferred sequence. Moreover, each chapter is designed so that it can be covered in 1–3 lectures, depending on the depth of coverage.

Example Courses

This book has several possible uses as a textbook. It can be used, for instance, for a core Algorithms course, which is classically known as CS7. Alternatively, it could be used for an upper-division/graduate data structures course, an upper-division/graduate algorithms course, or a two-course sequence on these topics. To highlight these alternatives, we give an example syllabus for each of these possible courses below.

Example syllabus for a core Algorithms (CS7) course:

1. Algorithm Analysis
(Skip, skim, or review Chapters 2–4 on fundamental data structures)[1]
5. Priority Queues and Heaps
6. Hash Tables
7. Union-Find Structures
8. Merge-Sort and Quick-Sort
9. Fast Sorting and Selection (if time permits)
10. The Greedy Method
11. Divide-and-Conquer
12. Dynamic Programming
13. Graphs and Traversals
14. Shortest Paths
15. Minimum Spanning Trees
16. Network Flow and Matching (if time permits)
17. ***NP***-Completeness
18. Approximation Algorithms

Optional choices from Chapters 19–26, as time permits

The optional choices from Chapters 19–26 that could be covered at the end of the course include randomized algorithms, computational geometry, string algorithms, cryptography, Fast Fourier Transform (FFT), and linear programming.

[1] These topics, and possibly even the topics of Chapters 5 and 6, are typically covered to at least a basic level in a Data Structures course that could be a prerequisite to this course.

Example syllabus for an upper-division/graduate Data Structures course:

1. Algorithm Analysis
2. Basic Data Structures
3. Binary Search Trees
4. Balanced Binary Search Trees
5. Priority Queues and Heaps
6. Hash Tables
7. Union-Find Structures
8. Merge-Sort and Quick-Sort
13. Graphs and Traversals
14. Shortest Paths
15. Minimum Spanning Trees
20. B-Trees and External-Memory
21. Multi-Dimensional Searching

Example syllabus for an upper-division/graduate Algorithms course:

(Skip, skim, or review Chapters 1–8)
9. Fast Sorting and Selection
10. The Greedy Method
11. Divide-and-Conquer
12. Dynamic Programming
16. Network Flow and Matching
17. ***NP***-Completeness
18. Approximation Algorithms
19. Randomized Algorithms
22. Computational Geometry
23. String Algorithms
24. Cryptography
25. The Fast Fourier Transform (FFT)
26. Linear Programming

This course could be taught either as a stand-alone course or in conjunction with an upper-division Data Structures course, such as that given above.

Of course, other options are also possible. Let us not belabor this point, however, leaving such creative arrangements to instructors.

Three Kinds of Exercises

This book contains many exercises—over 800—which are divided between the following three categories:

- ***reinforcement*** exercises, which test comprehension of chapter topics
- ***creativity*** exercises, which test creative utilization of techniques from the chapter
- ***application*** exercises, which test uses of the topics of the chapter for real-world or contrived applications

The exercises are distributed so that roughly 35% are reinforcement exercises, 40% are creativity exercises, and 25% are application exercises.

Web Added-Value Education

This book comes accompanied by an extensive website:

http://www.wiley.com/college/goodrich/

This site includes an extensive collection of educational aids that augment the topics of this book. Specifically for ***students*** we include the following:

- Presentation handouts in PDF format for most topics in this book
- Hints on selected exercises.

The hints should be of particular interest for creativity and application problems that may be quite challenging for some students.

For ***instructors*** using this book, there is a dedicated portion of the site just for them, which includes the following additional ***teaching aids***:

- Solutions to selected exercises in this book
- Editable presentations in PowerPoint format for most topics in this book.

Prerequisites

We have written this book assuming that the reader comes to it with certain knowledge. In particular, we assume that the reader has a basic understanding of elementary data structures, such as arrays and linked lists, and is at least vaguely familiar with a high-level programming language, such as C, C++, Java, or Python. Thus, all algorithms are described in a high-level "pseudocode," which avoids some details, such as error condition testing, but is suitable for a knowledgeable reader to convert algorithm descriptions into working code.

In terms of mathematical background, we assume the reader is familiar with exponents, logarithms, summations, limits, and elementary probability. Even so, we review many of these concepts in Chapter 1, and we give a summary of other useful mathematical facts, including elementary probability, in Appendix A.

About the Authors

Professors Goodrich and Tamassia are well-recognized researchers in algorithms and data structures, having published many papers in this field, with applications to computer security, cryptography, Internet computing, information visualization, and geometric computing. They have served as principal investigators in several joint projects sponsored by the National Science Foundation, the Army Research Office, and the Defense Advanced Research Projects Agency. They are also active in educational technology research.

Michael Goodrich received his Ph.D. in Computer Sciences from Purdue University in 1987. He is a Chancellor's Professor in the Department of Computer Science at University of California, Irvine. Previously, he was a professor at Johns Hopkins University. His research interests include analysis, design, and implementation of algorithms, data security, cloud computing, graph drawing, and computational geometry. He is a Fulbright scholar and a fellow of the American Association for the Advancement of Science (AAAS), Association for Computing Machinery (ACM), and Institute of Electrical and Electronics Engineers (IEEE). He is a recipient of the IEEE Computer Society Technical Achievement Award, the ACM Recognition of Service Award, and the Pond Award for Excellence in Undergraduate Teaching. He serves on the advisory boards of the *International Journal of Computational Geometry & Applications* (IJCGA) and of the *Journal of Graph Algorithms and Applications* (JGAA).

Roberto Tamassia received his Ph.D. in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign in 1988. He is the Plastech Professor of Computer Science in the Department of Computer Science at Brown University. He is also the Director of Brown's Center for Geometric Computing. His research interests include data security, applied cryptography, cloud computing, analysis, design, and implementation of algorithms, graph drawing and computational geometry. He is a fellow of the American Association for the Advancement of Science (AAAS), Association for Computing Machinery (ACM), and Institute for Electrical and Electronic Engineers (IEEE). He is also a recipient of the Technical Achievement Award from the IEEE Computer Society. He co-founded the *Journal of Graph Algorithms and Applications* (JGAA) and the Symposium on Graph Drawing. He serves as co-editor-in-chief of JGAA.

Acknowledgments

There are a number of individuals with whom we have collaborated on research and educational projects about algorithms. Working with them helped us refine the vision and content of this book. Specifically, we thank Jeff Achter, Vesselin Arnaudov, James Baker, Ryan Baker, Benjamin Boer, John Boreiko, Devin Borland, Lubomir Bourdev, Ulrik Brandes, Stina Bridgeman, Bryan Cantrill, Yi-Jen Chiang,

Robert Cohen, David Ellis, David Emory, Jody Fanto, Ben Finkel, Peter Fröhlich, Ashim Garg, David Ginat, Natasha Gelfand, Esha Ghosh, Michael Goldwasser, Mark Handy, Michael Horn, Greg Howard, Benoît Hudson, Jovanna Ignatowicz, James Kelley, Evgenios Kornaropoulos, Giuseppe Liotta, David Mount, Jeremy Mullendore, Olga Ohrimenko, Seth Padowitz, Bernardo Palazzi, Charalampos Papamanthou, James Piechota, Daniel Polivy, Seth Proctor, Susannah Raub, Haru Sakai, John Schultz, Andrew Schwerin, Michael Shapiro, Michael Shim, Michael Shin, Galina Shubina, Amy Simpson, Christian Straub, Ye Sun, Nikos Triandopoulos, Luca Vismara, Danfeng Yao, Jason Ye, and Eric Zamore.

We are grateful to our editor, Beth Golub, for her enthusiastic support of this project. The production team at Wiley has been great. Many thanks go to people who helped us with the book development, including Jayne Ziemba, Jennifer Welter, Debbie Martin, Chris Ruel, Julie Kennedy, Wanqian Ye, Joyce Poh, and Janis Soo.

We are especially grateful to Michael Bannister, Jenny Lam, and Joseph Simons for their contributions to the chapter on linear programming. We would like to thank Siddhartha Sen and Robert Tarjan for an illuminating discussion about balanced search trees.

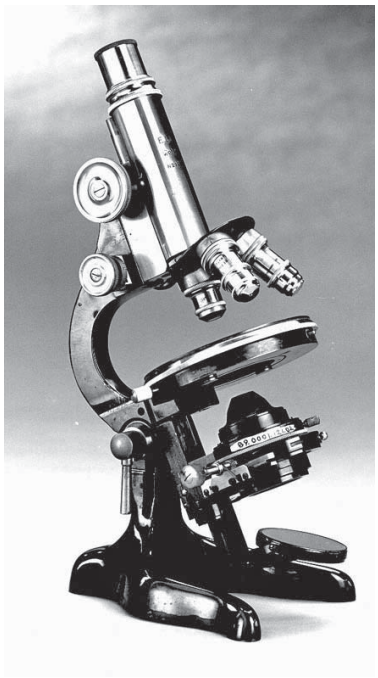
We are truly indebted to the outside reviewers, and especially to Jack Snoeyink, for detailed comments and constructive criticism, which were extremely useful. These other outside reviewers included John Donald, Hui Yang, Nicholas Tran, John Black, My Thai, Dana Randall, Ming-Yang Kao, Qiang Cheng, Ravi Janarden, Fikret Ercal, Jack Snoeyink, S. Muthukrishnan, Elliot Anshelevich, Mukkai Krishnamoorthy, Roxanne Canosa, Michael Cutler, Roger Crawfis, Glencora Borradaile, and Jennifer Welch.

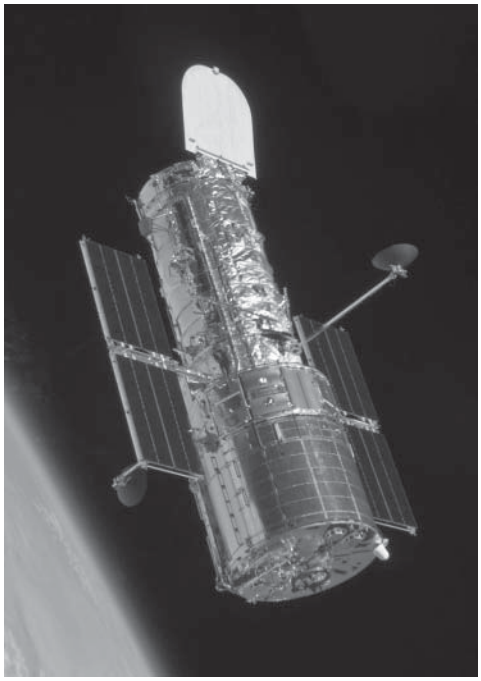
This manuscript was prepared primarily with LaTeX for the text and Microsoft PowerPoint®, Visio®, and Adobe FrameMaker® for the figures.

Finally, we warmly thank Isabel Cruz, Karen Goodrich, Giuseppe Di Battista, Franco Preparata, Ioannis Tollis, and our parents for providing advice, encouragement, and support at various stages of the preparation of this book. We also thank them for reminding us that there are things in life beyond writing books.

Michael T. Goodrich
Roberto Tamassia

Chapter 1 Algorithm Analysis





Microscope: U.S. government image, from the N.I.H. Medical Instrument Gallery, DeWitt Stetten, Jr., Museum of Medical Research. Hubble Space Telescope: U.S. government image, from NASA, STS-125 Crew, May 25, 2009.

Contents

Scientists often have to deal with differences in scale, from the microscopically small to the astronomically large, and they have developed a wide range of tools for dealing with the differences in scale in the objects they study. Similarly, computer scientists must also deal with scale, but they deal with it primarily in terms of data volume rather than physical object size. In the world of information technology, ***scalability*** refers to the ability of a system to gracefully accommodate growing sizes of inputs or amounts of workload. Being able to achieve scalability for a computer system can mean the difference between a technological solution that can succeed in the marketplace or scientific application and one that becomes effectively unusable as data volumes increase. In this book, we are therefore interested in the design of scalable algorithms and data structures.

Simply put, an ***algorithm*** is a step-by-step procedure for performing some task in a finite amount of time, and a ***data structure*** is a systematic way of organizing and accessing data. These concepts are central to computing, and this book is dedicated to the discussion of paradigms and principles for the design and implementation of correct and efficient data structures and algorithms. But to be able to determine the degree to which algorithms and data structures are scalable, we must have precise ways of analyzing them.

The primary analysis tool we use in this book is to characterize the ***running time*** of an algorithm or data structure operation, with ***space usage*** also being of interest. Running time is a natural measure for the purposes of scalability, since time is a precious resource. It is an important consideration in economic and scientific applications, since everyone expects computer applications to run as fast as possible.

We begin this chapter by describing the basic framework needed for analyzing algorithms, which includes the language for describing algorithms, the computational model that language is intended for, and the main factors we count when considering running time. We also include a brief discussion of how recursive algorithms are analyzed. In Section 1.1.5, we present the main notation we use to characterize running times—the so-called "big-Oh" notation. These tools comprise the main theoretical tools for designing and analyzing algorithms.

In Section 1.2, we take a short break from our development of the framework for algorithm analysis to review some important mathematical facts, including discussions of summations, logarithms, proof techniques, and basic probability. Given this background and our notation for algorithm analysis, we present a case study on algorithm analysis in Section 1.3, focusing on a problem often used as a test question during job interviews. We follow this case study in Section 1.4 by presenting an interesting analysis technique, known as amortization, which allows us to account for the group behavior of many individual operations. Finally, we conclude the chapter with some exercises that include several problems inspired by questions commonly asked during job interviews at major software and Internet companies.