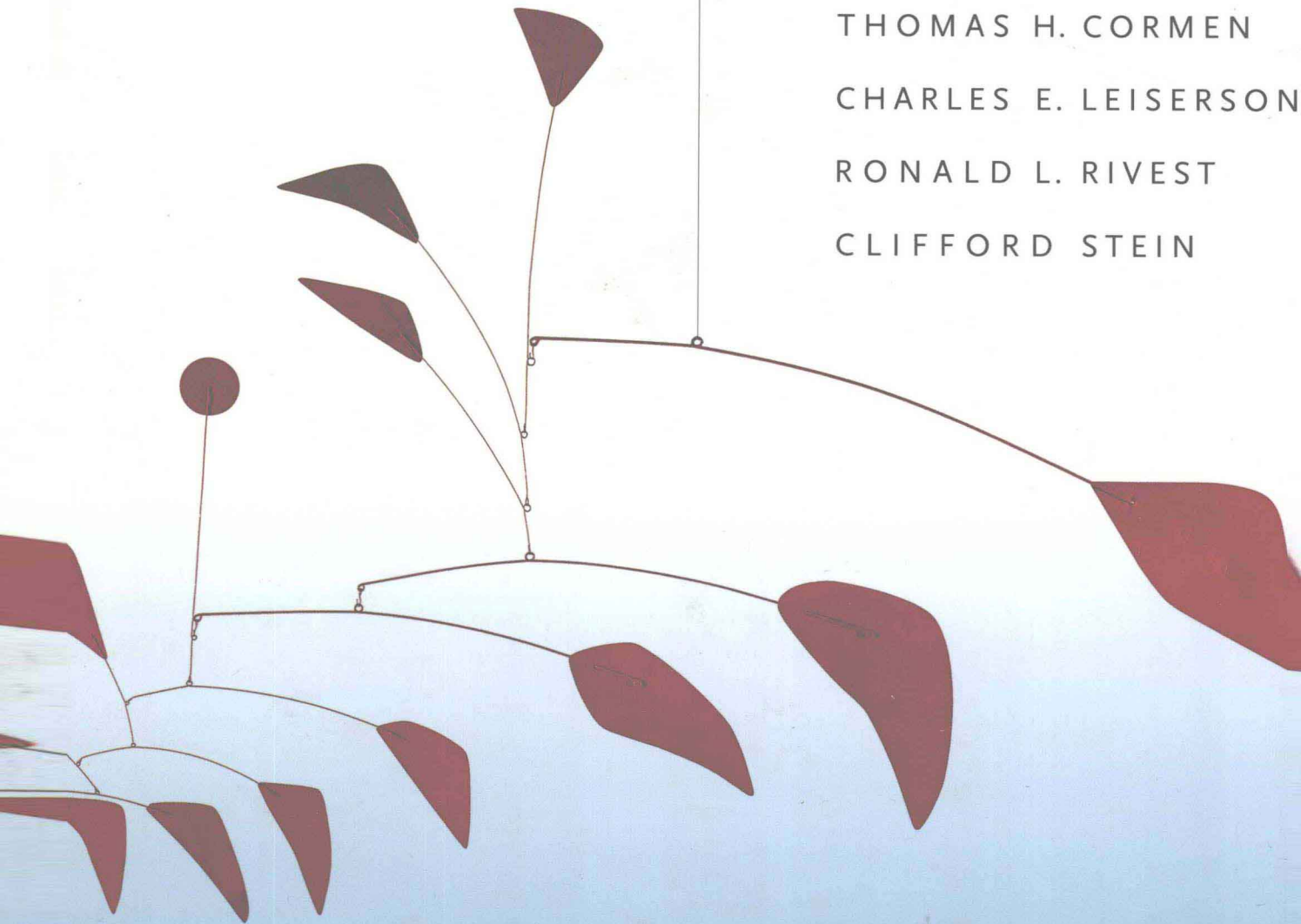
THOMAS H. CORMEN
CHARLES E. LEISERSON
RONALD L. RIVEST
CLIFFORD STEIN



INTRODUCTION TO ALGORITHMS

THIRD EDITION

Thomas H. Cormen
Charles E. Leiserson
Ronald L. Rivest
Clifford Stein

Introduction to Algorithms

Third Edition

The MIT Press
Cambridge, Massachusetts London, England

This book was set in Times Roman and Mathtime Pro 2 by the authors.

Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Introduction to algorithms / Thomas H. Cormen . . . [et al.].—3rd ed.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-262-03384-8 (hardcover : alk. paper)—ISBN 978-0-262-53305-8 (pbk. : alk. paper)
1. Computer programming. 2. Computer algorithms. I. Cormen, Thomas H.

QA76.6.I5858 2009
005.1—dc22

2009008593

10 9 8 7 6 5 4 3 2

Preface

Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing.

This book provides a comprehensive introduction to the modern study of computer algorithms. It presents many algorithms and covers them in considerable depth, yet makes their design and analysis accessible to all levels of readers. We have tried to keep explanations elementary without sacrificing depth of coverage or mathematical rigor.

Each chapter presents an algorithm, a design technique, an application area, or a related topic. Algorithms are described in English and in a pseudocode designed to be readable by anyone who has done a little programming. The book contains 244 figures—many with multiple parts—illustrating how the algorithms work. Since we emphasize *efficiency* as a design criterion, we include careful analyses of the running times of all our algorithms.

The text is intended primarily for use in undergraduate or graduate courses in algorithms or data structures. Because it discusses engineering issues in algorithm design, as well as mathematical aspects, it is equally well suited for self-study by technical professionals.

In this, the third edition, we have once again updated the entire book. The changes cover a broad spectrum, including new chapters, revised pseudocode, and a more active writing style.

To the teacher

We have designed this book to be both versatile and complete. You should find it useful for a variety of courses, from an undergraduate course in data structures up through a graduate course in algorithms. Because we have provided considerably more material than can fit in a typical one-term course, you can consider this book to be a "buffet" or "smorgasbord" from which you can pick and choose the material that best supports the course you wish to teach.

You should find it easy to organize your course around just the chapters you need. We have made chapters relatively self-contained, so that you need not worry about an unexpected and unnecessary dependence of one chapter on another. Each chapter presents the easier material first and the more difficult material later, with section boundaries marking natural stopping points. In an undergraduate course, you might use only the earlier sections from a chapter; in a graduate course, you might cover the entire chapter.

We have included 957 exercises and 158 problems. Each section ends with exercises, and each chapter ends with problems. The exercises are generally short questions that test basic mastery of the material. Some are simple self-check thought exercises, whereas others are more substantial and are suitable as assigned homework. The problems are more elaborate case studies that often introduce new material; they often consist of several questions that lead the student through the steps required to arrive at a solution.

Departing from our practice in previous editions of this book, we have made publicly available solutions to some, but by no means all, of the problems and exercises. Our Web site, http://mitpress.mit.edu/algorithms/, links to these solutions. You will want to check this site to make sure that it does not contain the solution to an exercise or problem that you plan to assign. We expect the set of solutions that we post to grow slowly over time, so you will need to check it each time you teach the course.

We have starred ($\star$) the sections and exercises that are more suitable for graduate students than for undergraduates. A starred section is not necessarily more difficult than an unstarred one, but it may require an understanding of more advanced mathematics. Likewise, starred exercises may require an advanced background or more than average creativity.

To the student

We hope that this textbook provides you with an enjoyable introduction to the field of algorithms. We have attempted to make every algorithm accessible and interesting. To help you when you encounter unfamiliar or difficult algorithms, we describe each one in a step-by-step manner. We also provide careful explanations of the mathematics needed to understand the analysis of the algorithms. If you already have some familiarity with a topic, you will find the chapters organized so that you can skim introductory sections and proceed quickly to the more advanced material.

This is a large book, and your class will probably cover only a portion of its material. We have tried, however, to make this a book that will be useful to you now as a course textbook and also later in your career as a mathematical desk reference or an engineering handbook.

What are the prerequisites for reading this book?

- You should have some programming experience. In particular, you should understand recursive procedures and simple data structures such as arrays and linked lists.
- You should have some facility with mathematical proofs, and especially proofs by mathematical induction. A few portions of the book rely on some knowledge of elementary calculus. Beyond that, Parts I and VIII of this book teach you all the mathematical techniques you will need.

We have heard, loud and clear, the call to supply solutions to problems and exercises. Our Web site, http://mitpress.mit.edu/algorithms/, links to solutions for a few of the problems and exercises. Feel free to check your solutions against ours. We ask, however, that you do not send your solutions to us.

To the professional

The wide range of topics in this book makes it an excellent handbook on algorithms. Because each chapter is relatively self-contained, you can focus in on the topics that most interest you.

Most of the algorithms we discuss have great practical utility. We therefore address implementation concerns and other engineering issues. We often provide practical alternatives to the few algorithms that are primarily of theoretical interest.

If you wish to implement any of the algorithms, you should find the translation of our pseudocode into your favorite programming language to be a fairly straightforward task. We have designed the pseudocode to present each algorithm clearly and succinctly. Consequently, we do not address error-handling and other software-engineering issues that require specific assumptions about your programming environment. We attempt to present each algorithm simply and directly without allowing the idiosyncrasies of a particular programming language to obscure its essence.

We understand that if you are using this book outside of a course, then you might be unable to check your solutions to problems and exercises against solutions provided by an instructor. Our Web site, http://mitpress.mit.edu/algorithms/, links to solutions for some of the problems and exercises so that you can check your work. Please do not send your solutions to us.

To our colleagues

We have supplied an extensive bibliography and pointers to the current literature. Each chapter ends with a set of chapter notes that give historical details and references. The chapter notes do not provide a complete reference to the whole field

of algorithms, however. Though it may be hard to believe for a book of this size, space constraints prevented us from including many interesting algorithms.

Despite myriad requests from students for solutions to problems and exercises, we have chosen as a matter of policy not to supply references for problems and exercises, to remove the temptation for students to look up a solution rather than to find it themselves.

Changes for the third edition

What has changed between the second and third editions of this book? The magnitude of the changes is on a par with the changes between the first and second editions. As we said about the second-edition changes, depending on how you look at it, the book changed either not much or quite a bit.

A quick look at the table of contents shows that most of the second-edition chapters and sections appear in the third edition. We removed two chapters and one section, but we have added three new chapters and two new sections apart from these new chapters.

We kept the hybrid organization from the first two editions. Rather than organizing chapters by only problem domains or according only to techniques, this book has elements of both. It contains technique-based chapters on divide-and-conquer, dynamic programming, greedy algorithms, amortized analysis, NP-Completeness, and approximation algorithms. But it also has entire parts on sorting, on data structures for dynamic sets, and on algorithms for graph problems. We find that although you need to know how to apply techniques for designing and analyzing algorithms, problems seldom announce to you which techniques are most amenable to solving them.

Here is a summary of the most significant changes for the third edition:

- We added new chapters on van Emde Boas trees and multithreaded algorithms, and we have broken out material on matrix basics into its own appendix chapter.
- We revised the chapter on recurrences to more broadly cover the divide-and-conquer technique, and its first two sections apply divide-and-conquer to solve two problems. The second section of this chapter presents Strassen's algorithm for matrix multiplication, which we have moved from the chapter on matrix operations.
- We removed two chapters that were rarely taught: binomial heaps and sorting networks. One key idea in the sorting networks chapter, the 0-1 principle, appears in this edition within Problem 8-7 as the 0-1 sorting lemma for compare-exchange algorithms. The treatment of Fibonacci heaps no longer relies on binomial heaps as a precursor.

- We revised our treatment of dynamic programming and greedy algorithms. Dynamic programming now leads off with a more interesting problem, rod cutting, than the assembly-line scheduling problem from the second edition. Furthermore, we emphasize memoization a bit more than we did in the second edition, and we introduce the notion of the subproblem graph as a way to understand the running time of a dynamic-programming algorithm. In our opening example of greedy algorithms, the activity-selection problem, we get to the greedy algorithm more directly than we did in the second edition.
- The way we delete a node from binary search trees (which includes red-black trees) now guarantees that the node requested for deletion is the node that is actually deleted. In the first two editions, in certain cases, some other node would be deleted, with its contents moving into the node passed to the deletion procedure. With our new way to delete nodes, if other components of a program maintain pointers to nodes in the tree, they will not mistakenly end up with stale pointers to nodes that have been deleted.
- The material on flow networks now bases flows entirely on edges. This approach is more intuitive than the net flow used in the first two editions.
- With the material on matrix basics and Strassen's algorithm moved to other chapters, the chapter on matrix operations is smaller than in the second edition.
- We have modified our treatment of the Knuth-Morris-Pratt string-matching algorithm.
- We corrected several errors. Most of these errors were posted on our Web site of second-edition errata, but a few were not.
- Based on many requests, we changed the syntax (as it were) of our pseudocode. We now use "$=$" to indicate assignment and "==" to test for equality, just as C, C++, Java, and Python do. Likewise, we have eliminated the keywords **do** and **then** and adopted "**//**" as our comment-to-end-of-line symbol. We also now use dot-notation to indicate object attributes. Our pseudocode remains procedural, rather than object-oriented. In other words, rather than running methods on objects, we simply call procedures, passing objects as parameters.
- We added 100 new exercises and 28 new problems. We also updated many bibliography entries and added several new ones.
- Finally, we went through the entire book and rewrote sentences, paragraphs, and sections to make the writing clearer and more active.

Web site

You can use our Web site, http://mitpress.mit.edu/algorithms/, to obtain supplementary information and to communicate with us. The Web site links to a list of known errors, solutions to selected exercises and problems, and (of course) a list explaining the corny professor jokes, as well as other content that we might add. The Web site also tells you how to report errors or make suggestions.

How we produced this book

Like the second edition, the third edition was produced in LaTeX 2_ε. We used the Times font with mathematics typeset using the MathTime Pro 2 fonts. We thank Michael Spivak from Publish or Perish, Inc., Lance Carnes from Personal TeX, Inc., and Tim Tregubov from Dartmouth College for technical support. As in the previous two editions, we compiled the index using Windex, a C program that we wrote, and the bibliography was produced with BibTeX. The PDF files for this book were created on a MacBook running OS 10.5.

We drew the illustrations for the third edition using MacDraw Pro, with some of the mathematical expressions in illustrations laid in with the psfrag package for LaTeX 2_ε. Unfortunately, MacDraw Pro is legacy software, having not been marketed for over a decade now. Happily, we still have a couple of Macintoshes that can run the Classic environment under OS 10.4, and hence they can run MacDraw Pro—mostly. Even under the Classic environment, we find MacDraw Pro to be far easier to use than any other drawing software for the types of illustrations that accompany computer-science text, and it produces beautiful output.[1] Who knows how long our pre-Intel Macs will continue to run, so if anyone from Apple is listening: *Please create an OS X-compatible version of MacDraw Pro!*

Acknowledgments for the third edition

We have been working with the MIT Press for over two decades now, and what a terrific relationship it has been! We thank Ellen Faran, Bob Prior, Ada Brunstein, and Mary Reilly for their help and support.

We were geographically distributed while producing the third edition, working in the Dartmouth College Department of Computer Science, the MIT Computer

[1] We investigated several drawing programs that run under Mac OS X, but all had significant shortcomings compared with MacDraw Pro. We briefly attempted to produce the illustrations for this book with a different, well known drawing program. We found that it took at least five times as long to produce each illustration as it took with MacDraw Pro, and the resulting illustrations did not look as good. Hence the decision to revert to MacDraw Pro running on older Macintoshes.

Science and Artificial Intelligence Laboratory, and the Columbia University Department of Industrial Engineering and Operations Research. We thank our respective universities and colleagues for providing such supportive and stimulating environments.

Julie Sussman, P.P.A., once again bailed us out as the technical copyeditor. Time and again, we were amazed at the errors that eluded us, but that Julie caught. She also helped us improve our presentation in several places. If there is a Hall of Fame for technical copyeditors, Julie is a sure-fire, first-ballot inductee. She is nothing short of phenomenal. Thank you, thank you, thank you, Julie! Priya Natarajan also found some errors that we were able to correct before this book went to press. Any errors that remain (and undoubtedly, some do) are the responsibility of the authors (and probably were inserted after Julie read the material).

The treatment for van Emde Boas trees derives from Erik Demaine's notes, which were in turn influenced by Michael Bender. We also incorporated ideas from Javed Aslam, Bradley Kuszmaul, and Hui Zha into this edition.

The chapter on multithreading was based on notes originally written jointly with Harald Prokop. The material was influenced by several others working on the Cilk project at MIT, including Bradley Kuszmaul and Matteo Frigo. The design of the multithreaded pseudocode took its inspiration from the MIT Cilk extensions to C and by Cilk Arts's Cilk++ extensions to C++.

We also thank the many readers of the first and second editions who reported errors or submitted suggestions for how to improve this book. We corrected all the bona fide errors that were reported, and we incorporated as many suggestions as we could. We rejoice that the number of such contributors has grown so great that we must regret that it has become impractical to list them all.

Finally, we thank our wives—Nicole Cormen, Wendy Leiserson, Gail Rivest, and Rebecca Ivry—and our children—Ricky, Will, Debby, and Katie Leiserson; Alex and Christopher Rivest; and Molly, Noah, and Benjamin Stein—for their love and support while we prepared this book. The patience and encouragement of our families made this project possible. We affectionately dedicate this book to them.

THOMAS H. CORMEN — *Lebanon, New Hampshire*
CHARLES E. LEISERSON — *Cambridge, Massachusetts*
RONALD L. RIVEST — *Cambridge, Massachusetts*
CLIFFORD STEIN — *New York, New York*

February 2009

Contents

VIII Appendix: Mathematical Background