Algorithms: A Top-Down Approach

Rodney R. Howell
Dept. of Computing and Information Sciences
Kansas State University
Manhattan, KS
USA

Ninth Draft
This textbook was prepared with \LaTeX. Figures were prepared using \texttt{Xfig} and \texttt{gnuplot}.

Copyright ©2009 by Rodney R. Howell.

All rights reserved. No portion of this text may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author. Permission is hereby granted to students of Kansas State University to print, photocopy, or store electronically this draft of this text for their own academic use.
Preface

This book gives an introduction to the design and analysis of algorithms. It is intended to introduce the reader to the theories of algorithm correctness and performance analysis, as well as to give a broad overview of algorithm design techniques. As the proper organization of data is essential to the efficiency of many algorithms, a significant portion of the text is devoted to design and analysis of data structures.

This book is motivated in part by the author’s belief that people do not fully understand an algorithm until they are able to prove its correctness. For this reason, all of Chapter 2 and much of Chapter 4 are devoted to techniques for proving correctness. Outside of these two chapters, however, very few correctness proofs are given, as they would needlessly clutter the presentation with tedious details. Instead, all algorithms are presented in such a way as to facilitate a correctness proof. The particular approach used is top-down. This approach fits well with mathematical techniques of induction, proving loop invariants, and applying previously-shown theorems. By having an understanding of the mechanics of proving correctness, readers are equipped to gain a thorough understanding of an algorithm presented in a top-down way.

Correct use of notation, particularly asymptotic notation, is emphasized throughout. Although asymptotic notation is routinely abused in the literature, the author’s experience is that students find such abuse confusing and misleading. As others have argued, nothing is gained by this abuse, so it is avoided in this book.

Algorithms are presented in a pseudo language defined with enough rigor to facilitate both performance analyses and correctness proofs. The essence of an algorithm is not tied to a particular programming language; hence, we have chosen a computational model and corresponding notation that we feel are appropriate for communicating algorithms to humans.

This text contains sufficient material for both an advanced undergraduate course and an introductory graduate-level course on the design and
Prerequisite Material

This book is intended to be reasonably self-contained. However, a certain degree of maturity is assumed regarding the audience. Readers are expected to have enough experience in writing programs so as to be able to understand algorithms presented in a pseudo language. Experience with basic data structures, including stacks, queues, lists, and trees, will be helpful, as will experience in manipulating finite and infinite sums, solving recurrences, and performing combinatorial analyses. Though calculus and number theory are used occasionally, background in these fields of study is not assumed.

Organization

The outline of this book is as follows:

I. Fundamentals — the foundational material upon which the remainder of the text is based.

1. Introduction — introduction to the ideas of specification, correctness, and analysis of algorithms, as well as to the top-down approach.

2. Proving Algorithm Correctness — introduction to techniques for proving algorithm correctness.


II. Data Structures — data structures commonly used with algorithms, including algorithms presented later in this text.

4. Basic Techniques — techniques for reasoning about and analyzing data structures. A more detailed discussion of the computational model and notation is presented. Amortized analysis is introduced.

5. Priority Queues — binary heaps, leftist heaps, skew heaps, and randomized heaps. Randomized algorithms and expected-case analysis are introduced.

6. Storage/Retrieval I: Ordered Keys — ordered arrays, binary search trees, AVL trees, splay trees, and skip lists.
7. Storage/Retrieval II: Unordered Keys — virtually initialized arrays and hashing, including universal hashing.

8. Disjoint Sets — structures for supporting MERGE and FIND operations on disjoint sets.


III. Algorithm Design Techniques — an in-depth look at standard algorithmic design techniques.

10. Divide and Conquer — algorithms for sorting, selecting the kth smallest, multiplying polynomials, and multiplying/dividing integers.

11. Optimization I: Greedy Algorithms — algorithms for finding a minimum-cost spanning tree, finding shortest paths, scheduling, and generating Huffman codes.


IV. Common Reduction Targets — algorithms for problems to which many other problems can be reduced.

13. Depth-First Search — algorithms for topological sort, articulation points, and strongly connected components.


15. * The Fast Fourier Transform — a study of the Fast Fourier Transform algorithm and its application to problems such as integer multiplication.

V. Intractable Problems — dealing with problems for which we cannot find efficient solutions.

16. $\mathcal{NP}$-Completeness — an introduction to the theory of $\mathcal{NP}$-completeness, including proofs of $\mathcal{NP}$-completeness.

17. Approximation Algorithms — an introduction to design and analysis of approximation algorithms for $\mathcal{NP}$-complete problems.
Sections which may be omitted due to more mathematical content have titles prefixed with a “∗”.

An undergraduate course would consist of all of Part I (except Section 3.11) and most of Parts II and III. Parts IV and V are targeted more for graduate students, though selected topics could be used in an undergraduate course. A graduate-level course should also include all of Part I (though it could probably be covered rather quickly) and perhaps selected topics from Part II. The core of this course would come from Part III, supplemented by some or all of Parts IV and V.

**Exercises**

Each chapter includes a section of exercises intended to reinforce the chapter material. Some of the exercises are more challenging, and are therefore marked with “∗” or “∗∗” to indicate their level of difficulty. Exercises marked “∗” will be challenging for most undergraduates, and exercises marked “∗∗” will be challenging for most graduate students.

**Web Site**

This textbook, as well as various supplemental materials, may be accessed online at

```text
www.cis.ksu.edu/~rhowell/algorithms-text/
```

The textbook itself requires a password if accessed from outside the `ksu.edu` domain. All other materials are publicly available.

**Feedback**

Your feedback is most welcome. Please send corrections, questions, comments, and/or suggestions to the author at

```text
rhowell@ksu.edu
```