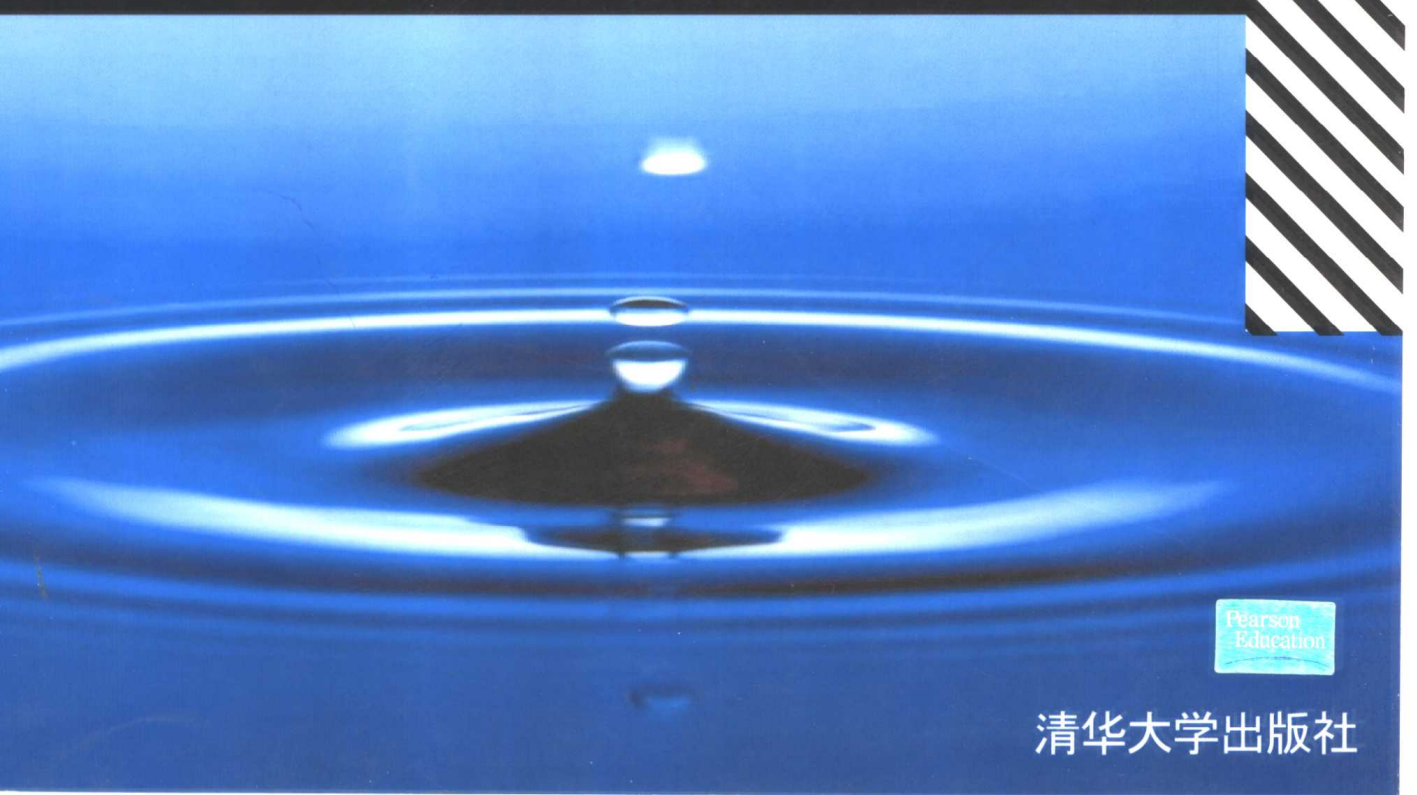
大学计算机教育国外著名教材、教参系列 (影印版)

DATA STRUCTURES & ALGORITHM ANALYSIS IN C++

(Second Edition)

MARK ALLEN WEISS

数据结构与算法分析 C++描述 (第2版)



Pearson Education

清华大学出版社

Data Structures & Algorithm Analysis in C++

Second Edition

数据结构与算法分析 C++描述

第2版

Mark Allen Weiss

Florida International University

清华大学出版社

（京）新登字 158 号

Data Sturctures & Alogorithm Analysis in C++ 2nd ed.
Mark Allen Weiss

北京市版权局著作权合同登记号：图字：01-2002-1661

书　名：Data Structures & Algorithm Analysis in C++ (2nd ed.)
作　者：Mark Allen Weiss
出版者：清华大学出版社（北京清华大学学研大厦，邮编 100084）
http:// www.tup.tsinghua.edu.cn
印刷者：北京牛山世兴印刷厂
发行者：新华书店总店北京发行所
开　本：787×960 1/16　印张：38
版　次：2002 年 9 月第 1 版　2002 年 9 月第 1 次印刷
书　号：ISBN 7-302-05702-8/TP・3362
印　数：0001～4000
定　价：54.00 元

出 版 说 明

进入 21 世纪，世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的争夺。谁拥有大量高素质的人才，谁就能在竞争中取得优势。高等教育，作为培养高素质人才的事业，必然受到高度重视。目前我国高等教育的教材更新较慢，为了加快教材的更新频率，教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从 1996 年开始，与国外著名出版公司合作，影印出版了“大学计算机教育丛书（影印版）”等一系列引进图书，受到了国内读者的欢迎和支持。跨入 21 世纪，我们本着为我国高等教育教材建设服务的初衷，在已有的基础上，进一步扩大选题内容，改变图书开本尺寸，一如既往地请有关专家挑选适用于我国高校本科及研究生计算机教育的国外经典教材或著名教材以及教学参考书，组成本套“大学计算机教育国外著名教材、教参系列（影印版）”，以飨读者。深切期盼读者及时将使用本系列教材、教参的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材，以利我们把“大学计算机教育国外著名教材、教参系列（影印版）”做得更好，更适合高校师生的需要。

计算机引进版图书编辑室

2002.3

Preface

Purpose/Goals

The second edition of *Data Structures and Algorithms Analysis in* C++ describes *data structures,* methods of organizing large amounts of data, and *algorithm analysis,* the estimation of the running time of algorithms. As computers become faster and faster, the need for programs that can handle large amounts of input becomes more acute. Paradoxically, this requires more careful attention to efficiency, since inefficiencies in programs become most obvious when input sizes are large. By analyzing an algorithm before it is actually coded, students can decide if a particular solution will be feasible. For example, in this text students look at specific problems and see how careful implementations can reduce the time constraint for large amounts of data from 16 years to less than a second. Therefore, no algorithm or data structure is presented without an explanation of its running time. In some cases, minute details that affect the running time of the implementation are explored.

Once a solution method is determined, a program must still be written. As computers have become more powerful, the problems they must solve have become larger and more complex, requiring development of more intricate programs. The goal of this text is to teach students good programming and algorithm analysis skills simultaneously so that they can develop such programs with the maximum amount of efficiency.

This book is suitable for either an advanced data structures (CS7) course or a first-year graduate course in algorithm analysis. Students should have some knowledge of intermediate programming, including such topics as pointers, recursion, and object-based programming, and some background in discrete math.

Approach

Although the material in this text is largely language independent, programming requires the use of a specific language. As the title implies, we have chosen C++ for this book.

C++ has emerged as the leading systems programming language. In addition to fixing many of the syntactic flaws of C, C++ provides direct constructs (the *class* and *template*) to implement generic data structures as abstract data types.

The most difficult part of writing the book was deciding on the amount of C++ to include. Use too many features of C++, and one gets an incomprehensible text; use too few and you have little more than a C text that supports classes.

The approach we take is to present the material in an *object-based approach*. As such, unlike the first edition, there is no use of inheritance in the text. We use class templates to describe generic data structures. We generally avoid esoteric C++ features, and use the `vector` and `string` classes that are now part of the C++ standard. Using these first-class versions, instead of the second-class counterparts that were used in the first edition, simplifies much of the code. Because not all compilers are current, we provide a `vector` and `string` class in Appendix B; this is the class that is actually used in the online code. Chapter 1 provides a review of the C++ features that are used throughout the text.

Complete versions of the data structures, in both C++ and Java, are available on the Internet. We use similar coding conventions to make the parallels between the two languages more evident. The code has been tested on UNIX systems using g++ (2.7.2 and 2.8.1) and SunPro 4.0 and on Windows95 systems using Visual C++ 5.0 and 6.0, Borland C++ 5.0, and Codewarrior Pro Release 2.

Overview

Chapter 1 contains review material on discrete math and recursion. I believe the only way to be comfortable with recursion is to see good uses over and over. Therefore, recursion is prevalent in this text, with examples in every chapter except Chapter 5. Chapter 1 also includes material that serves as a review of basic C++. Included is a discussion of templates and important constructs in C++ class design.

Chapter 2 deals with algorithm analysis. This chapter explains asymptotic analysis and its major weaknesses. Many examples are provided, including an in-depth explanation of logarithmic running time. Simple recursive programs are analyzed by intuitively converting them into iterative programs. More complicated divide-and-conquer programs are introduced, but some of the analysis (solving recurrence relations) is implicitly delayed until Chapter 7, where it is performed in detail.

Chapter 3 covers lists, stacks, and queues. The emphasis here is on coding these data structures using ADTs, fast implementation of these data structures, and an exposition of some of their uses. There are almost no complete programs, but the exercises contain plenty of ideas for programming assignments.

Chapter 4 covers trees, with an emphasis on search trees, including external search trees (B-trees). The UNIX file system and expression trees are used as examples. AVL trees and splay trees are introduced. More careful treatment of search tree implementation details is found in Chapter 12. Additional coverage of trees, such as file compression and game trees, is deferred until Chapter 10. Data structures for an external medium are considered as the final topic in several chapters.

Chapter 5 is a relatively short chapter concerning hash tables. Some analysis is performed, and extendible hashing is covered at the end of the chapter.

Chapter 6 is about priority queues. Binary heaps are covered, and there is additional material on some of the theoretically interesting implementations of

priority queues. The Fibonacci heap is discussed in Chapter 11, and the pairing heap is discussed in Chapter 12.

Chapter 7 covers sorting. It is very specific with respect to coding details and analysis. All the important general-purpose sorting algorithms are covered and compared. Four algorithms are analyzed in detail: insertion sort, Shellsort, heapsort, and quicksort. External sorting is covered at the end of the chapter.

Chapter 8 discusses the disjoint set algorithm with proof of the running time. This is a short and specific chapter that can be skipped if Kruskal's algorithm is not discussed.

Chapter 9 covers graph algorithms. Algorithms on graphs are interesting, not only because they frequently occur in practice but also because their running time is so heavily dependent on the proper use of data structures. Virtually all of the standard algorithms are presented along with appropriate data structures, pseudocode, and analysis of running time. To place these problems in a proper context, a short discussion on complexity theory (including *NP*-completeness and undecidability) is provided.

Chapter 10 covers algorithm design by examining common problem-solving techniques. This chapter is heavily fortified with examples. Pseudocode is used in these later chapters so that the student's appreciation of an example algorithm is not obscured by implementation details.

Chapter 11 deals with amortized analysis. Three data structures from Chapters 4 and 6 and the Fibonacci heap, introduced in this chapter, are analyzed.

Chapter 12 covers search tree algorithms, the k-d tree, and the pairing heap. This chapter departs from the rest of the text by providing complete and careful implementations for the search trees and pairing heap. The material is structured so that the instructor can integrate sections into discussions from other chapters. For example, the top-down red-black tree in Chapter 12 can be discussed under AVL trees (in Chapter 4). Appendix A discusses the Standard Template Library and illustrates how the concepts described in this text are applied to a high-performance data structures and algorithms library. Appendix B describes an implementation of `vector` and `string`.

Chapters 1–9 provide enough material for most one-semester data structures courses. If time permits, then Chapter 10 can be covered. A graduate course on algorithm analysis could cover Chapters 7–11. The advanced data structures analyzed in Chapter 11 can easily be referred to in the earlier chapters. The discussion of *NP*-completeness in Chapter 9 is far too brief to be used in such a course. Garey and Johnson's book on *NP*-completeness can be used to augment this text.

Exercises

Exercises, provided at the end of each chapter, match the order in which material is presented. The last exercises may address the chapter as a whole rather than a specific section. Difficult exercises are marked with an asterisk, and more challenging exercises have two asterisks.

A solutions manual containing solutions to almost all the exercises is available online to instructors from the Addison Wesley Longman Publishing Company. Instructors should contact their Addison-Wesley local sales representative for information on the manual's availability.

References

References are placed at the end of each chapter. Generally the references either are historical, representing the original source of the material, or they represent extensions and improvements to the results given in the text. Some references represent solutions to exercises.

Code Availability

The example program code in this book is available via anonymous ftp at `ftp.awl.com`. It is also accessible through the World Wide Web; the URL is `http://www.awl.com/cseng/` (follow the links from there). The exact location of this material may change.

Acknowledgments

Many, many people have helped me in the preparation of books in this series. Some are listed in other versions of the book; thanks to all.

As usual, the writing process was made easier by the professionals at Addison Wesley Longman. I'd like to thank my editor, Susan Hartman; associate editor, Katherine Harutunian; and production editor, Pat Unubun. I'd also like to thank Kris Engberg and her staff at Publication Services for their usual fine work putting the final pieces together.

I would like to thank the reviewers, who provided valuable comments, many of which have been incorporated into the text. For this edition, they are Phillip T. Conrad (Temple University), Robin Hill (University of Wyoming), Bob Robinson (University of Georgia), Gurdip Singh (Kansas State University), Bernard M. Waxman (Southern Illinois University at Edwardsville), and William W. White (Southern Illinois University at Edwardsville).

Finally, I'd like to thank the numerous readers who have sent e-mail messages and pointed out errors or inconsistencies in earlier versions. My World Wide Web page `http://www.cs.fiu.edu/~weiss` will contain updated source code (in C++, C, and Java), an errata list, and a link to submit bug reports.

M.A.W.
Miami, Florida

Contents