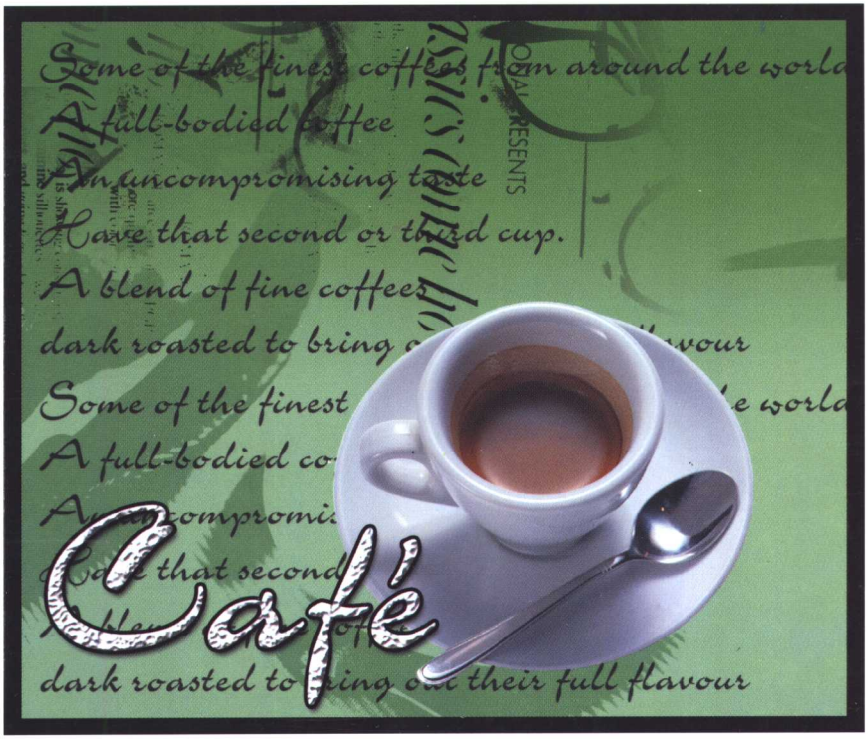
Java程序员书库

Java数据结构与算法分析（影印版）

Data Structures & Algorithm Analysis in Java™



(美) Mark Allen Weiss 编著

Pearson Education

科学出版社
www.sciencep.com

Java 程序员书库

Java 数据结构与算法分析

（影印版）

Data Structures & Algorithm Analysis In Java

First Edition

（美） Mark Allen Weiss 编著

科学出版社

北京

图字：01-2003-7653 号

内 容 简 介

本书介绍了常见的数据结构，如链表、堆栈、队列、树、哈希表等，并对查找、排序等进行了算法分析，还给出了相应的 Java 实现。

本书逻辑结构严谨，主次分明，可用做计算机教材或程序员参考用书。

图书在版编目(CIP)数据

Java 数据结构与算法分析=Data Structures and Algorithm in Java/（美）Mark Allen Weiss(马克·艾伦·威丝)编著.—影印本. —北京：科学出版社，2004

ISBN 7-03-012498-7

Ⅰ.J… Ⅱ.M… Ⅲ.①Java 语言—程序设计—英文②数据机构—英文③算法分析—英文 Ⅳ.①TP312 ②TP311.12

中国版本图书馆 CIP 数据核字（2003）第 103029 号

策划编辑：李佩乾 / 责任编辑：王日臣

责任印制：吕春珉 / 封面制作：东方人华平面设计室

科学出版社 出版

北京东黄城根北街16号

邮政编码:100717

http://www.sciencep.com

双青印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2004 年 1 月第 一 版 开本：787×960 1/16

2004 年 1 月第 一次印刷 印张：35

印数：1—3 000 字数：800 000

定价：56.00 元

（如有印装质量问题，我社负责调换〈环伟〉）

影印前言

程序设计在于处理复杂性：问题的复杂性和所用的程序设计工具的复杂性。Java 的魅力在于其本身的低复杂性，同时又能很好地处理高度复杂的问题。Java 程序的开发周期只有类似的 C++程序的一半甚至更少，而且 Java 可以方便地处理复杂软件问题：多线程、分布式、跨平台、安全性等。Java 从诞生到现在，已经广泛应用于几乎所有类型软件系统的构建。尤其在基于 Web 的系统开发中，Java 技术具有独特的优势。熟悉 Java 历史的人都知道，Java 的前身——编程语言 Oak 就是致力于电子产品互连的语言，Internet 的发展导致了 Oak 的重生和 Java 的广为流行。现在，J2EE 技术已经成为企业级 Web 应用系统的标准平台。

好的程序设计人员不仅仅要掌握优秀的编程工具，更需要掌握优秀的编程思想。随着面向对象编程技术数十年的发展，人们开始提炼和总结面向对象编程中行之有效的、具有一定普遍意义的方法，即面向对象的设计模式。以 Gamma、Helm、Johnson 和 Vlissides 合著的经典书籍《设计模式》为开端，面向对象设计模式的研究和应用成为面向对象程序设计的重要内容。所有结构良好的面向对象软件系统都包含了大量的设计模式，能否熟练应用设计模式已经成为衡量程序员水平的至关重要的标准。

本丛书收录了与 Java 程序设计和 Java 设计模式相关的经典书籍，反映了应用 Java 开发软件系统的最佳经验总结和最新动态。

《Java 设计模式》采用方便而简洁的编写风格，以可视化的 Java 程序为例，详细介绍了 Gamma、Helm、Johnson 和 Vlissides 合著的经典书籍《设计模式》中列出的所有 23 种模式。通过本书，Java 程序员可以迅速了解和掌握设计模式的内容，并在实践中应用设计模式来创建复杂而健壮的 Java 程序。

《Java 模式应用》介绍了基于模式的开发技巧，演示了使用 Java 开发各种商务系统中的模式应用。书中首先概述设计模式，然后就四种主要模式——创建模式、行为模式、结构模式和系统模式展开了详细的论述。该书还针对系统构建过程中常用的 J2EE、JSP、EJB 和 API 等技术作了介绍，适合具有一定编程基础的 Java 程序员阅读参考。

《J2EE 核心模式》是 Sun Java Center 的资深设计师的 J2EE 亲身实践经验的总结。该书主要描述 J2EE 关键技术（Java Server Pages，Java Servlet，Enterprise Java Beans，Java Message Services 等）的模式、最佳实践、设计策略和经过验证的解决方案。该书介绍了 J2EE 包括的 15 个模式的分类和大量的策略，不仅具有理论深度，而且非常实用。该书内容适合 J2EE 的爱好者、程序员、设计师、开发者和技术管理者。一句话，该书

适合于设计、构建和开发 J2EE 平台应用的所有人。

XML 是新一代文档的标准，Web 页、数据、源码等等，均可以用 XML 文档表示。越来越多的程序员正在使用 Java 来处理 XML 文档。《用 Java 处理 XML》详细论述了如何使用 Java 来读写 XML 文档。该书是目前最新和最全的 Java 处理 XML 技术的介绍，包含内容超过 1000 页的关于 SAX, DOM, JDOM, JAXP, TrAX, XPath, XSLT, SOAP 等的讲解。该书适合于使用 Java 读写 XML 文档的 Java 程序员。其内容从基本概念到高级应用无所不包，特别适合作为手册随时参考。

在《实时 Java》中，作为 RTSJ 专家组的成员之一，Dibble 从 Java 平台特有的实时问题概述开始，依次讲解了 RTSJ 各项主要特性的使用方法。从广泛的实时原理到详细的编程隐患，该书覆盖了构建有效实时程序所需的一切知识。其主要内容包括：与非实时代码的互操作性、实时开发中的取舍以及 JVM 软件的实时问题；垃圾收集、无堆栈访问、物理内存和“不朽”内存以及无堆栈内存的常数时间分配；优先级调度、期限调度以及速率单调分析；闭包、异步传输控制、异步事件以及计时器。这是一本非常实用的指南，适用于有经验的 Java 平台开发人员。

《Java 数据结构与算法分析》介绍了常见的数据结构，如链表、堆栈、队列、树和哈希表等，并对查找和排序进行了算法分析，给出了相应的 Java 实现。该书逻辑结构严谨，主次分明，可用作程序员参考书。

总之，这套书详细介绍了 Java 应用的许多重要方面：从具有普遍性的 Java 数据结构和算法、Java 设计模式、Java 模式应用、J2EE 核心模式，到日益显著的 Java 特殊应用领域（Java 处理 XML 文档和 Java 实时系统开发）。其内容具有一定的理论深度，更有重要的实际参考价值。

有鉴于此，特向 Java 系统开发和应用领域中不同程度的读者推荐这套书，相信每位有心的读者都能得到物超所值的收获。

清华大学经济管理学院管理科学与工程系　朱涛 博士
讲授课程：Java 程序设计，面向对象的分析设计方法

Preface

Purpose/Goals

This new Java edition 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 object-based programming and recursion, 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 Java for this book.

Java is a relatively new language that is often examined in comparison with C++. Java offers many benefits, and programmers often view Java as a safer, more portable, and easier-to-use language than C++. As such, it makes a fine core language for discussing and implementing fundamental data structures. Other important parts of Java, such as threads and its GUI, although important, are not needed in this text and thus are not discussed.

As with every programming language, Java has some disadvantages. It does not directly support generic programming; a workaround is required that is discussed in Chapter 1. I/O

support using Java is minimal, but there are now many I/O classes available on the Internet that make things simpler. In any event, the examples in the text make minimal use of the Java I/O facilities.

Java's advantage—that it simplifies C++'s (sometimes confusing) syntax—is also a liability. Programmers who eventually code in C++ will find hidden traps in what should be a straightforward conversion of Java code to C++ code. To help alleviate this problem, the accompanying website contains a chapter that describes C++ and illustrates many of C++'s subtleties and nuances that would not be obvious to many.

Complete versions of the data structures, in both Java and C++, are available on the Internet. We use similar coding conventions to make the parallels between the two languages more evident.

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 presents material that serves as a review of basic Java. Included is a discussion of the Java workaround needed for generic algorithms and data structures.

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 along with AVL trees (in Chapter 4).

Appendix A lists some commonly used Java classes. Appendix B discusses the Java Collections package. This standard package (as of Java 1.2) contains implementations of several data structures discussed in the text.

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 to instructors from the Addison Wesley Longman Publishing Company.

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 `aw.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 Chris Brown (University of Rochester), Thang N. Bui (Pennsylvania State University at Harrisburg), G. H. Hedrick (Oklahoma State University), Sampath Kannan (University of Pennsylvania), and Gurdip Singh (Kansas State University).

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 Java, C, and C++), an errata list, and a link to submit bug reports.

M.A.W.

Contents