计算机科学教材系列 —— 新世纪 新版教材

数据结构与算法分析

(Java 版)

A Practical Introduction to Data Structures and Algorithm Analysis Java Edition

[美] Clifford A. Shaffer 著

张 铭 刘晓丹 译

PH PTR

電子工業出版社
Publishing House of Electronics Industry
URL:http://www.phei.com.cn

国外计算机科学教材系列

数据结构与算法分析
(Java 版)

A Practical Introduction to Data Structures and Algorithm Analysis
Java Edition

[美] Clifford A. Shaffer 著

张 铭 刘晓丹 译

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 提 要

作为《数据结构与算法分析(C++版)》的姊妹篇,本书采用了当前十分流行且适合于 Internet 环境的面向对象程序设计语言 Java 作为算法描述语言。本书利用 Java 的接口(Interface)来定义抽象数据类型,这比使用 C++的类更自然。本书把数据结构原理和算法分析技术有机地结合在一起,系统地介绍了各种类型的数据结构和排序、检索的各种方法。作者非常注意对每一种数据结构的不同存储方法及有关算法进行分析比较。本书还引入了一些比较高级的数据结构与先进的算法分析技术,并介绍了可计算性理论的一般知识。

本书概念清楚,逻辑性强,内容新颖,可作为大专院校计算机软件专业与计算机应用专业学生的教材和参考书,也可供计算机工程技术人员参考。

图书在版编目(CIP)数据

数据结构与算法分析(Java 版)/(美)沙佛(Shaffer. C. A.)著;张铭译.—北京:电子工业出版社,2001.1
国外计算机科学教材系列
ISBN 7-5053-6497-9

Ⅰ.数… Ⅱ.①沙…②张… Ⅲ.①数据结构-教材②算法分析-教材 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字(2001)第 03483 号

丛 书 名:国外计算机科学教材系列
书　　名:**数据结构与算法分析(Java 版)**
原 书 名:A Practical Introduction to Data Structures and Algorithm Analysis Java Edition
著　　者:[美] Clifford A. Shaffer
译　　者:张　铭　刘晓丹
责任编辑:吴　源
排版制作:电子工业出版社计算机排版室监制
印 刷 者:北京东光印刷厂
出版发行:电子工业出版社　URL:http://www.phei.com.cn
　　北京市海淀区万寿路 173 信箱　邮编:100036
经　　销:各地新华书店
开　　本:787×1092　1/16　印张:21　字数:537 千字
版　　次:2001 年 2 月第 1 版　2001 年 2 月第 1 次印刷
书　　号:ISBN 7-5053-6497-9 / TP·3566
印　　数:10100 册　　定价:35.00 元
版权贸易合同登记号　图字:01-2000-3484

译 者 序

数据结构以及算法分析是计算机专业十分重要的基础课，计算机科学各领域及各种应用软件都要使用相关的数据结构和算法。

当面临一个新的设计问题时，设计者需要选择适当的数据结构并设计出满足一定时间和空间限制的有效算法。本书作者把数据结构和算法分析有机地结合在一本教材中，有助于读者根据问题的性质选择合理的数据结构，并对时间空间复杂性进行必要的控制。

本书采用当前十分流行的 Java 作为算法描述语言，Java 的接口（Interface）可以使抽象数据类型（ADT）的概念得到更自然的体现。而且随着网络技术的迅速发展，Java 这种平台无关且天生适合于因特网的面向对象程序设计语言得到了非常广泛的应用。广大 Java 程序员十分需要直接用 Java 语言描述经典数据结构和算法的技术书籍。

本书包括四大部分内容，第一部分是准备工作，介绍了一些基本概念和术语以及基本的数学知识。

第二部分介绍了最基本的数据结构，依次为线性表（包括栈和队列）、二叉树、树和图。每种数据结构都从其数学特性入手，先介绍抽象数据类型，然后再讨论不同的存储方法，并且研究不同存储方法的可能算法。

作为最常用的算法，排序和检索历来是数据结构讨论的重点问题，这在第三部分作了详尽的讨论。排序算法最能体现算法分析的魅力，它的算法速度要求非常高。第 8 章证明了所有基于比较的排序算法的时间代价是 $\Theta(n\log n)$，这也是排序问题的时间代价。检索则考虑怎样提高检索速度，这往往是与存储方法有关的。书中介绍了几种高效的数据结构，例如自组织线性表、散列表、B 树和 B^+ 树等，都具有极好的检索性能。

第四部分介绍了数据结构的应用与一些高级主题。例如跳跃表、广义表和稀疏矩阵等更复杂的线性表结构，Trie 结构、伸展树等复杂树结构，k-d 树、PR 四分树等空间数据结构。另外还简单介绍了求和、递归关系分析和均摊分析等高级算法分析技术。这些技术对于提高程序员的算法分析能力具有重要的作用。

附录部分介绍了对于理解本书例子必要的一些 Java，从而使得不懂 Java 语法的 C/C++ 和 Pascal 程序员能够更好地理解本书。作者提供的参考书目也颇有价值。

本书的前言及第 1 章至第 8 章由张铭翻译，第 9 章至第 15 章由刘晓丹翻译，肖毅、柴栗、肖之屏、刘堃等人参与了本书的核对工作，译者在此表示感谢。由于水平有限，难免有不妥之处，欢迎批评指正。

前　言

我们研究数据结构的目的是为了学会编写效率更高的程序。现在的计算机速度一年比一年快，为什么还需要高效率的程序？这是由于人类解决问题的雄心与能力是同步增长的。现代计算技术在计算能力和存储容量上的革命，仅仅提供了计算更复杂问题的有效工具，而程序的高效性要求永远也不会过时。

程序高效性的要求不会，也不应该与合理的设计和简明清晰的编码相矛盾。高效程序的设计基于良好的信息组织和优秀的算法，而不是基于"编程小技巧"。程序员如果没有掌握设计简明清晰程序的基本原理，就不可能编写有效的程序。反过来说，简洁的程序需要合理的数据组织和清晰的算法。大多数计算机科学系的课程设置都意识到要培养良好的程序设计技能，首先应该强调基本的软件工程原理。因此，一旦程序员学会了设计和实现简明清晰程序的原理，下一步就应该学习有效的数据组织和算法，以提高程序的效率。

途径：本书描述了许多表示数据的技术。这些技术包括以下原则：

1. 每一种数据结构和每一个算法都有其时间、空间的开销和效率。当面临一个新的设计问题时，设计者要透彻掌握怎样权衡时间、空间开销和算法有效性的方法，以适应问题的需要。这就需要懂得算法分析的原理，而且还需要了解所使用的物理介质的特性（例如，数据存储在磁盘上与存储在主存上，就有不同的考虑）。

2. 与开销和效率有关的是时空权衡。例如，人们通常增加空间开销来减少运行时间，或者相反。程序员所面对的时空权衡问题普遍存在于软件设计和实现的各个阶段，因此这个概念必须牢记在心。

3. 程序员应该充分了解一些现成的方法，以免作不必要的重复开发工作。因此，学生需要了解经常使用的数据结构和相关算法。

4. 数据结构服从于应用需求。学生必须把分析应用需求放在第一位，然后再寻找一个与实际应用相匹配的数据结构。要做到这一点，需要应用上述三条原则。

组织：数据结构和算法设计的书籍往往囿于下面这两种情形之一：一种是教材，一种是百科全书。有的书籍试图融合这两种编排，但通常是二者都没有组织好。本书是作为教材来编写的。我相信了解选择或设计解决问题的高效数据结构的基本原理是十分重要的，这比死记硬背书本内容重要得多。因此，我在本书中涵盖了大多数但不是所有的标准数据结构。为了阐述一些重要原理，也包括了某些并非广泛使用的数据结构。另外，还介绍了一些相对较新但即将得到广泛应用的数据结构。

本书可以作为本科生一个学期的教学内容，也可以作为专业技术人员的自学教材。读者应该具有编程经验，最好学过相当于两个学期的结构化程序设计语言 Pascal 或 C 语言。在第 2 章中，复习了一些数学预备知识，早已熟悉数学归纳法和递归的读者会容易掌握一些。

尽管本书应该一个学期完成，但书中超过了一个学期的内容，这可以为教师提供一些选择的余地。二年级学生的基本数据结构和算法分析背景不太多，可以对他们详细讲解第 1 章～第 12 章的内容，再从第 13 章中选择一些专题来讲解，我就是这样来给二年级学生讲课的。背

景知识更丰富的学生,可以先读第 1 章,跳过第 2 章中除参考书目之外的内容,简要地浏览第 3 章和第 4 章(请着重阅读 4.1.3 小节),然后详细阅读其余章节。另外,教师可以根据程序设计实习的需要,选择第 13 章以后的某些专题内容。

第 13 章是针对做较大的程序设计练习而编写的。我建议所有选修数据结构的学生,都应该做一些高级树结构或其他较复杂的动态数据结构的上机实习,比如第 12 章中的跳跃表(Skip List)或稀疏矩阵。所有这些数据结构都不比二叉检索树更难,而且学完第 5 章的学生都有能力来实现它们。

我尽量合理地安排内容顺序。教师可以根据需要自由地重新组织内容。读者掌握了第 1 章至第 6 章后,以下的内容就相对独立了。显然,外排序依赖于内排序和磁盘文件系统。Kruskal 最小支撑树算法使用了 6.2 节关于 UNION/FIND 的算法。10.2 节的自组织线性表提到了 9.3 节讨论的缓冲区置换技术。第 14 章的讨论基于本书的例题。15.3 节依赖于图论知识。一般情况下,大多数主题都只依赖于同一章中讨论过的内容。

关于 Java:本书的示例程序是用 Java 来写的。像其他程序设计语言一样,Java 有利有弊。Java 是一种小型语言,往往只有一种方法来解决某个问题。程序员只要正确使用 Java,就能得到清晰的程序结构。从这个角度来看,它比 C 或 C++ 优越。Java 可以很好地定义和使用大多数传统的数据结构,如线性表和树。另一方面,Java 的文件处理能力很弱,笨拙而低效,而且它的内存控制能力也很弱。例如,12.4 节讨论的存储管理,就很难用 Java 来编写。由于我希望全书能够使用同一种语言,所以像其他程序员一样,我只能接受 Java 的这些弱点,瑕不掩瑜嘛。最重要的是表达算法的思想,并不需要考虑这些思想是否适合某种具体的语言。大多数程序员将使用多种程序设计语言,本书所描述的概念将在不同的环境中被证明是有用的。

我并不想难倒那些对 Java 不熟悉的读者。在保持 Java 优点的同时,我尽量使示例程序简明、清晰。Java 在本书中只作为阐释数据结构的工具。值得庆幸的是,对于 C 或 Pascal 程序员而言,Java 是一种很容易掌握的语言,只需要学习很少的与面向对象程序设计有关的语法即可。特别是我用到了 Java 隐蔽实现细节的特性,例如类(class)、私有成员(private class member)和接口(Interface)。这些特性支持了一个关键的概念:体现于抽象数据类型(abstract data type)中的逻辑设计与体现于数据结构中的物理实现的分离。

我没有在本书中讲授 Java 语言的意图,只是提供了一个附录来解释一些基本的 Java 语义和概念,这对读懂示例程序是必要的。另外还提供了通过匿名 FTP 得到本书中 Java 源代码的途径。

本书很少使用继承(Inheritance)这一面向对象程序设计的关键特征。类的继承是避免重复编码和降低程序错误率的重要工具;但是从教育学的标准观点来看,类的继承在若干类中分散了数据元素的描述,从而使得程序更难理解。因此,我对于一些对象的类定义,比如线性表和树结点的定义,就没有充分继承前面示例编码中的类定义。这并不意味着程序员也应该这样做。避免代码重复和减少错误是很重要的目标,请不要把本书中的示例程序直接拷贝到自己的程序中,而只是把它们看作是对数据结构原理的阐释。

本书中的示例程序提供了有关数据结构原理的具体描述,而不是一系列具有商业质量的类实现。这些例子中所作的参数检查,比起从事商业软件设计的程序员在编程中所作的要少得多。某些参数检查是以调用类 Assert 中函数的形式包含进来的,这些函数模仿了 C 的标准库函数 assert。方法 Assert.notFalse 的输入是一个布尔表达式,一旦这个表达式的值为假

(false),程序就立即终止。方法 Assert.notNull 的输入是类 Object 的一个引用(reference),如果这个引用的值为空(null),程序就会终止。更准确地说,这些函数产生一种 IllegalArgumentException(非法参数异常)。除非程序员处理这个异常(exception),否则将导致程序终止。当一个函数接收到非法参数时终止程序,这种做法在实际程序中是不合适的,但是对于理解一个数据结构怎样运作是十分有用的。在实际程序应用中,应该使用 Java 的异常处理功能来处理输入数据错误。

在示例程序中,我严格区分了"Java 实现"和"伪码"(pseudocode)。一个标明"Java 实现"的示例程序至少在一个编译器中被真正编译过。伪码的示例通常具有与 Java 接近的语法,但是一般包含一行以上的更高级的描述。当我发现简单的、尽管并不十分精确的描述具有更好的教学效果时,就使用伪码。

几乎每一章都是以"深入学习导读"一节来结束的。它并不是那一章的综合参考索引,而是为了通过这些导读书籍或文章提供给读者更广泛的信息和乐趣。有些情况下我也提供了某个知名计算机科学家的重要背景文章。

习题和项目设计:只靠读书是不能学会灵活使用数据结构的,一定要通过编制实际的程序,比较不同的数据结构技术来观察在一种给定的条件下哪一种结构更有效。另外,学生也需要通过编程序来提高他们的算法分析与设计能力。这里提供了 300 多个习题和项目设计的程序实习题,希望读者能够很好地利用它们。

与作者联系的方法以及相关资料的获取:本书难免有一些错误,有些方面还有待进一步研究。作者非常欢迎读者指正,并提出建设性意见。作者在因特网(Internet)上的电子邮件(E-mail)地址是 shaffer@cs.vt.edu。也可以写信给以下地址:

Cliff Shaffer
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061
USA

与本书有关的一套基于 LATEX 系统的幻灯片材料可以通过 FTP 的匿名帐号(anonymous),从 ftp.prenhall.com 的目录中获得。该目录为:

pub/esm/computer_science.s-041/shaffer/ds/supplements/transparencies

例题的 Java 代码可以从相同 FTP 站点下面的目录获得:

pub/esm/computer_science.s-041/shaffer/ds/code

弗吉尼亚技术学院二年级数据结构课程的 WWW 主页(Home Page)之 URL 为:

http://ei.cs.vt.edu/~cs2604

该网址上同时可以看到一个针对数据结构的可视化和图形化调试工具,称为 SWAN 系统。

本书是用 LATEX 的写作系统编排的,LATEX 是 TEX 系统的一个软件包。参考书目(bibliography)是用 BIBTEX 编排的,词汇表用 makeindex 编写。图形主要是用 Xfig 来制作,但图 3.1 和图 10.5 实验性地使用了 Mathematica。

致谢:本书得到了许多友人的帮助。我想特别感谢其中的几位,他们对本书的出版贡献最大。对于没有被提及的朋友,在此表示歉意。系主任 Jack Carroll 对本书给予了重要的精神上的帮助。弗吉尼亚技术学院在 1994 年秋季的学术休假中使得整个出书的事情成为可能,我是从那时开始着手准备的。Mike Keenan、Lenny Heath 和 Jeff Shaffer 对本书最初版本的内容提供了有价值的意见。尤其是 Lenny Heath 多年来一直与我深入讨论算法设计和分析的有关问题以及怎样把二者讲授给学生的方法。Layne Watson 提供了有关 Mathematica 的帮助,Bo Begole、Philip Isenhour 和 Craig Struble 提供了一些技术上的帮助。Steve Edwards、Mark Abrams 和 Dennis Kafura 回答了一些有关 C++ 和 Java 的问题。

对于许多评阅了本书初稿的朋友,本人欠情甚深。这些评阅者是:J. David Bezek (University of Evansville)、Douglas Campbell(Brigham Young University)、Karen Davis (University of Cincinnati)、Vijay Kumar Garg(University of Texas-Austin)、Jim Miller(University of Kansas)、Bruce Maxim(University of Michigan-Dearborn)、Jeff Parker(Agile Networks/Harvard)、Dana Richards(George Mason University)、Jack Tan(University of Houston)和 Lixin Tao(Concordia University)。要不是他们的热心帮助,本书会出现更多技术上的错误,内容也将更肤浅。

没有 Prentice Hall 公司众多朋友的帮助,不可能有本书的出版,因为作者不可能自己印出书来。因此,我要感谢 Laura Steele 和 Alan Apt 这两位编辑。感谢本书 C++ 版的责任编辑 Kathleen Caren 和 Java 版的责任编辑 Ed DeFelipis,他们在本书接近出版的最紧张的日子里,保持各个方面运作良好。感谢 Bill Zobrist 和 Bruce Gregory 使我着手此事。感谢 Prentice Hall 的 Truly Donovan、Linda Behrens 和 Phyllis Bregman 在本书出版过程中提供的帮助。可能还有许多没有被提及的 Prentice Hall 公司的朋友,默默地提供了帮助。

我十分感激 Hanan Samet 传授给我有关数据结构的知识。我从他那里学到了许多原理,当然本书中可能的错误并不是他的责任。感谢我的妻子 Terry 对我的爱和支持。最后,也是最重要的是,要感谢这些年来选修数据结构的学生,是他们使我知道了在数据结构课程中什么是重要的而什么应该忽略,许多深入的问题也是他们提供的。这本书献给他们。

克利福德·沙佛(Clifford A. Shaffer)
福吉尼亚州,布莱克斯堡(Blacksburg, Virginia)

目　录

第一部分　预备知识

第二部分 基本数据结构

第三部分　排序和检索

第四部分　应用与高级话题

第一部分　预备知识

第 1 章　数据结构和算法

第 2 章　数学预备知识

第 3 章　算法分析

第 1 章　数据结构和算法

信息的表示是计算机科学的基础。大多数计算程序的主要目标与其说是完成运算，倒不如说是存储和检索信息。从存储空间和运行时间的现实角度来看，这些程序必须组织信息，以支持高效的信息处理过程。因此，研究数据结构和算法以有效地支持程序的实现就成了计算机科学的核心问题。

1.1　数据结构的原则

1.1.1　学习数据结构的必要性

有人可能认为，随着计算机功能的日益强大，程序的运行效率变得越来越不那么重要了。然而，计算机功能越强大，人们就越想去尝试更复杂的问题。而更复杂的问题需要更大的计算量，这就使得对高效程序的需求更加明显，工作愈复杂就愈偏离人们的日常经验。当今的计算机科学家必须训练出彻底理解隐藏在高效程序设计后面的一般原理的能力，因为他们的日常生活经验并不具备这些能力。

一般说来，一种数据结构就是一类普通数据的表示及其相关操作。从数据表示的观点来看，即使是存储在计算机中的一个整数或者一个浮点数也是一个简单的数据结构。更典型地，一个数据结构被认为是一组数据项的组织或者结构。存储在数组中的一个有序整数表就是这种结构的一个例子。

如果有足够的空间来存储一组数据项，总会有可能在这个数据项集合中查找出指定的数据项、打印数据项或将这些数据项处理成任何期望得到的顺序，或者更改任何特定数据项的值。因此，就有可能对任何数据结构施加所有必要的运算。然而，选择不同的数据结构可能会产生很大的差异：同样一个程序，可能在几秒钟内运行完毕，也可能需要几天时间才能完成运行。

毋庸置疑，人们编写程序是为了解决问题，这个道理本来不用作者再次提出。然而，在选择数据结构解决特定问题时，头脑中有这个不言而喻的道理是具有重要意义的。只有通过预先分析问题来确定必须达到的性能目标，才有希望挑选出正确的数据结构。有相当多的程序设计人员却忽视了这一分析过程，而直接选用某一个他们习惯使用的、但是与问题不相称的数据结构，结果设计出一个低效率的程序。相反，当使用简单的设计能够达到性能目标时，选用复杂的数据表示来改进这个程序也是没有道理的。

定义 1.1　一个算法如果能在所要求的资源限制(resource constraint)内将问题解决好，则称这个算法是有效率的(efficient)。例如，一个资源限制是：可用来存储数据的全部空间——可以分为内存空间限制和磁盘(外存)空间限制——和允许执行每一个子任务所需要的时间。一个算法如果比其他已知解所需要的资源都少，这个算法也可以称为是有效率的。一个算法的代价(cost)是指这个算法消耗的资源量。一般说来，代价是由一个关键资源例如时间来评估的，这意味着这个算

法满足其他资源限制。

当为解决某一问题而选择数据结构时，应该完成以下几步：

1. 分析问题以确定任何算法均会遇到的资源限制。
2. 确定必须支持的基本运算，并度量每种运算所受的资源限制。基本运算的实例包括向数据结构中插入一个数据项、从数据结构中删除一个数据项和查找指定的数据项。
3. 选择最接近这些开销的数据结构。

根据这三个步骤来选择数据结构，实际上贯彻了一种以数据为中心的设计观点。先定义数据和对数据的操作，然后确定数据的表示方法，最后是数据表示的实现。

某些重要的操作，例如查找、插入和删除数据记录的资源限制通常决定了数据结构的选择过程。对于这些操作相对重要性的争论焦点集中在以下三个问题中，无论什么时候，只要选择数据结构，就应该仔细考虑这三个问题：

- 开始时将所有数据项都插入数据结构，还是与其他操作混合在一起插入？
- 数据项可以删除吗？
- 所有数据项是安排在某一个已经定义好的序列中，还是允许随机进入？

显然，非集中插入、允许删除数据项和支持数据项的随机进入都需要更复杂的表示方法。

1.1.2　代价与效益

每一个数据结构都将代价与效益联系在一起。如果有人说某个算法在所有情况下都比其他算法好，这通常是不正确的。因为本书提到的几乎每一个数据结构和算法，我们都会发现一些例子说明它在什么地方才是最好的选择，其中有些例子是出人意料的。

一个数据结构需要一定的空间来存储它的每一个数据项，一定的时间来执行单个基本操作，一定的程序设计工作。每一个问题都有可利用的空间和时间的限制。问题的每一个解决方案都利用了一定比例的相关基本操作，数据结构的选择过程必须考虑到这一点。只有对问题的特性仔细分析之后，才能得到执行这项任务的最好的数据结构。

例 1.1　设计一个支持银行顾客账户记录的数据库系统。这个数据库必须可以添加或者删除顾客的账户，并且可以让顾客查询账户信息以及存款取款。顾客愿意在开户或者销户时等上几分钟，却不愿意为个别账目业务（如存钱、取钱）等候几秒钟。

银行提供一种自动取款机（ATM）让顾客使用，以查询余额、存款或者取款。比较特殊的是，这些 ATM 业务并不太更改数据库（为简单起见，只是假设钱增加了或减少了，这项业务仅仅改变账户记录里存储的值）。向数据库中添加一个新账户需要花费几分钟的时间（一般的顾客并不经常开户，而且开户时顾客在银行经理办公室等候）。销户没有时间限制，因为从顾客的角度来说，重要的只是所有的钱被取回（跟取款过程相同）；从银行的角度来说，账户记录可以在营业时间之后从数据库中删除。

一个删除效率很低、查找效率极高，并且具有适度插入效率的数据结构，应该符合上述问题所要求的资源限制。通过账号很容易取得账户记录（有时称之为“精确匹配”检索方法）。符合这些要求的数据结构就是在 10.4 节中描述的散列表。散列表具有非常快的精确匹配检索。当修改操作不影响记录长度时，记录可以很快被修改。散列表也支持新记录的高效率插入。散列表还能够支持高效率删除，但是删除得太多会导致其他操作性能的降低。然而，散列表可以定期重组，以将系统还原到最高效率状态。这种重组应该脱机进行，以免影响 ATM 业务。