<<算法技术手册>>

图书基本信息

书名：<<算法技术手册>>

13位ISBN编号：9787111286745

10位ISBN编号：711128674X

出版时间：2010-3

出版时间：机械工业出版社

作者：George T. Heineman,Gary Pollice,Stanley Selkow

页数：333

译者：杨晨,李明

前言

算法，神秘而晦涩的词汇。
算法，是计算机科学中最重要同时也是最基础的一环。
从开始学习计算机，我们就深知，算法是整个计算机科学的核心。
然而直至我们工作数年后，能够真正学好算法的人，却依旧是风毛麟角。
这并不是计算机教育的错，也不是计算机从业人员的错，更不是算法的错。
长久以来，算法就像古老的咒语，算法背后高深的数学知识更让人望而生畏。
其实，我们始终没有找到一条从理论走向实践的路。
在这里，我们很高兴能向大家介绍本书。
它正是能够带领你学好算法的一本不可多得的好书。
本书的三位作者是伍斯特理工学院的教授，其中GeorgeT．Heineman毕业于达特茅斯学院和哥伦比亚大学，曾经获得过GE、IBM和AT&T的研究奖金，在软件工程方面有独到的研究。
而Gary Pollice曾经供职于Rational Sonware，Sun等多家巨头，有着丰富的工业界经验，知道如何将学术和工业结合起来。
Stanley M．Selkow毕业于卡内基梅隆大学和宾夕法尼亚大学，擅长图论和算法设计。
本书由这三位伍斯特理工学院计算机理论专家合著，向我们展示了工业界和学术界对算法的不同看法以及如何高效地将理论和实践相结合。
本书搭建了一条真正属于开发者的路。
本书的读者主要面向本科生以及程序设计人员，同样也适用于产品和项目管理人员。
由于译者的知识和经验有限，翻译中难免有疏漏或错误，敬请广大读者谅解并批评指正。

<<算法技术手册>>

内容概要

开发健壮的软件需要高效的算法，然后程序员们往往直至问题发生之时，才会去求助于算法。《算法技术手册》讲解了许多现有的算法，可用于解决各种问题。
通过阅读它，可以使您学会如何选择和实现正确的算法，来达成自己的目标。
另外，书中的数学深浅适中，足够使您可以了解并分析算法的性能。
较之理论而言，本书更专注于应用。
《算法技术手册》提供了高效的代码解决方案，使用多种语言进行编写，让您可以轻松地将其应用于特定的工程当中。
通过本书，您可以：　　·解决特定代码的问题，或者提升既有解决方案的性能　　·快速找到与您所解决的问题相关的算法，并决定哪个算法才是最适合的那一个　　·探索使用C、C++、Java以及Ruby实现的算法解决方案以及开发小贴士　　·了解算法预期的性能，以及它达到最高性能时所需要的条件　　·发现不同算法之间相似的设计哲学　　·学习高级数据结构，来提升算法的性能
通过《算法技术手册》，您能学到如何提升算法的性能，这将是您的软件应用程序走向成功的关键。
作者简介：George T.Heineman，Gary Pollice和Stanley Selkow均为 Woree ste r PolYteChniC In stitute（伍斯特理工学院）计算机科学系的教授。
George是《Component—B ased Software Engineering：Putting the Pieces Together》（Addison—Wesley（的合编者，Gary则是《Head First Object-Oriented Analysis and Design》（O'Reilly）的合著者。

作者简介

George T.Heineman，Gary Pollice和Stanley Selkow均为 Woree ster PolYteChniC In stitute（伍斯特理工学院）计算机科学系的教授。
George是《Component—B ased Software Engineering：Putting the Pieces Together》（Addison—Wesley（的合编者，Gary则是《Head First Object-Oriented Analysis and Design》（O'Reilly）的合著者。

<<算法技术手册>>

书籍目录

章节摘录

插图：这个方法的总开销取决于你在每个槽中是元素列表还是nil值（表示没有列表）。
当一个槽只有一个元素时，它可以使用列表来存取。
作为我们的第一个近似的解决方案，我们将会使用这种方法并且在性能成为瓶颈时改进它。
驱动因素选择散列函数是在实现基于散列的查找之前必须做的第一个决定。
散列已经被研究多年，并且有大量的描述高效散列函数的论文，不过用在查找上使用这些函数简直是杀鸡用牛刀。
例如，某些特定的散列函数对于加密非常重要。
对于查找来说，一个散列函数必须要有一个好的分布，并且计算速度非常快。
存储器空间是设计基于散列查找时需要考虑的另外一个因素。
主存储器A必须能够足够大，以存下所有的查找键值，并且要给冲突键值留下足够的空间。
A的大小通常是一个素数。
但是，如果我们不使用开放定址的话（见接下来的“变种”一节），我们能够使用任何数字作为A的大小。
实践中一个好的选择是2k-1，即使这个值并不是素数。
存储在散列表的元素对内存有直接的影响。
考虑图5-3的是如何在链表中存储每个字符串的，所以存储在A中的元素看做链表元素。
堆中包含着指向元素的指针。
每一个链表都有存储的开销，包含着指向链表中的第一个和最后一个元素。
如果你使用Java的LinkedList类，一些很重要的附加字段和类使得实现非常灵活。
程序员可以写一个简单得多的链表类，只提供必需的功能，但是确实给基于散列查找算法带来了额外的开销。
如果你使用Linked List类，假设指针是四个字节，那么A中的每一个非空元素都需要12字节内存。
每个字符串元素不可以直接转换成ListElement，需要12个额外字节。
对于之前的那个有213 557个单词的例子，我们需要5005488字节的额外存储。

媒体关注与评论

“作者完成了一项了不起的工作，将晦涩的学术加以精炼，完美地在理论和实践之间取得了平衡，从而使本书成为了一本不可或缺的指南。
从此，彻底领悟算法就变得十分简单了。
”　　——Matthew Russell. Digital ReasOning Systems高级技术总监，《D0jo权威指南》（O’Reilly）的作者

编辑推荐

《算法技术手册》是由机械工业出版社出版的。

版权说明