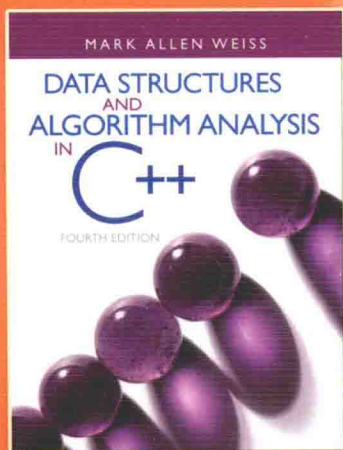


# 数据结构与算法分析

## ——C++语言描述（第四版）

Data Structures and Algorithm Analysis in C++  
Fourth Edition



[美] Mark Allen Weiss 著

冯舜玺 译

国外计算机科学教材系列

# 数据结构与算法分析

——C++语言描述

(第四版)

Data Structures and Algorithm Analysis in C++

Fourth Edition

[美] Mark Allen Weiss 著

冯舜玺 译

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

## 内 容 简 介

本书是数据结构和算法分析的经典教材,书中使用主程序设计语言 C++ 的新标准 C++ 11 作为具体的实现语言。内容包括表、栈、队列、树、散列表、优先队列、排序、不相交集算法、图论算法、算法分析、算法设计、摊还分析、查找树算法、后缀数组、后缀树、k-d 树和配对堆等。本书把算法分析与 C++ 程序的开发有机地结合起来,深入分析每种算法,内容全面、缜密严格,并细致讲解精心构造程序的方法。

本书概念清楚,逻辑性强,内容新颖,适合作为大专院校计算机软件与计算机应用等相关专业的教材或参考书,也适合计算机工程技术人员参考。

Authorized translation from the English language edition, entitled *Data Structures and Algorithm Analysis in C++, Fourth Edition*, 9780132847377 by Mark Allen Weiss, published by Pearson Education, Inc., publishing as Addison-Wesley, Copyright©2014 Pearson Education Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD., and PUBLISHING HOUSE OF ELECTRONICS INDUSTRY Copyright ©2016.

本书中文简体字版专有出版权由 Pearson Education (培生教育出版集团)授予电子工业出版社。未经出版者预先书面许可,不得以任何方式复制或抄袭本书的任何部分。

本书贴有 Pearson Education (培生教育出版集团)激光防伪标签,无标签者不得销售。

版权贸易合同登记号 图字:01-2013-7213

### 图书在版编目(CIP)数据

数据结构与算法分析: C++语言描述: 第四版/(美)M.A.韦斯(Mark Allen Weiss)著;冯舜玺译。

北京:电子工业出版社,2016.8

书名原文: *Data Structures and Algorithm Analysis in C++, Fourth Edition*

国外计算机科学教材系列

ISBN 978-7-121-29057-2

I. ①数… II. ①M… ②冯… III. ①数据结构—高等学校—教材 ②算法分析—高等学校—教材 ③C 语言—程序设计—高等学校—教材 IV. ①TP311.12 ②TP312

中国版本图书馆 CIP 数据核字(2016)第 131907 号

策划编辑:冯小贝

责任编辑:周宏敏

印刷:三河市华成印务有限公司

装订:三河市华成印务有限公司

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开本:787×1092 1/16 印张:31.75 字数:833 千字

版次:2016 年 8 月第 1 版(原著第 4 版)

印次:2016 年 8 月第 1 次印刷

定 价:89.00 元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010)88254888,88258888。

质量投诉请发邮件至 zltts@phei.com.cn,盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式: fengxiaobei@phei.com.cn。

# 前 言

## 目的/目标

本书是《数据结构与算法分析——C++语言描述》的第四版，论述组织大量数据的方法——数据结构，以及算法运行时间的估计——算法分析。随着计算机的速度越来越快，对于能够处理大量输入数据的程序需求变得日益急迫。可是，由于在输入量很大时程序的低效率显得非常突出，因此又要求对效率问题给予更加严密的关注。通过在实际编程之前对算法进行分析，学生们可以确定一个特定的解决方案是否可行。例如，在本书中学生可查阅一些特定的问题并看到精心的实现怎样能够把处理大量数据的时间限制从若干个世纪减至不到一秒。因此，若无运行时间的阐释，就不会有算法和数据结构的提出。在某些情况下，对于影响实现的运行时间的一些微小细节都需要认真的探究。

一旦解法被确定，接着就要编写程序。随着计算机功能的日益强大，它们必须解决的问题就变得更加庞大和复杂，这就要求开发更加复杂的程序。本书的目标是同时教授学生良好的程序设计技巧和算法分析能力，使他们开发的程序能够具有最高的效率。

本书适合于高等数据结构课程或是算法分析第一年的研究生课程。学生应该具有中等程度的程序设计知识，包括指针、递归以及面向对象程序设计这样一些内容，此外还要具有一些离散数学的基础。

## 处理方法

虽然本书的内容大部分都与语言无关，但是，程序设计还是需要使用某种特定的语言。正如书名指出的，我们为本书选择了C++。

C++已经成为主流系统编程语言。除修复C语言的许多语法漏洞之外，C++还提供一些直接结构(类和模板)来实现作为抽象数据类型的泛型数据结构。

编写本书最困难的部分是决定C++所占的篇幅。使用太多C++的特性将使本书难以理解，使用太少又会使读者得不到比支持类的C语言教材更多的收获。

我们所采取的做法是以一种基于对象的处理方式展示书中的内容。这样，本书几乎不存在对继承的使用。书中以类模板描述泛型数据结构。一般我们避免深奥的C++特性，但是使用了vector类和string类，如今它们已是C++标准的一部分。本书以前各版通过把类模板接口从其实现中分离出来已经做到了对类模板的实现。虽然这是有争议的首选方式，但它揭示出编译器使读者难以具体使用代码的一些问题。因此，这一版中联机代码把类模板作为一个独立的单元来介绍，无需接口与实现的分离。第1章提供了用于全书的C++特性的综述，并描述了我们对类模板的处理方式。附录A阐述如何能够重写类模板以使用分离式编译。

以C++和Java二者描述的数据结构的完全版可在互联网上获得。我们使用类似的编码约定以使得这两种语言间平行的结构表现得更加明显。

## 第四版最重要的变化概述

本书第四版吸收了大量对错误的修正，书中许多部分经过修订以增加表述的清晰度。此外：

- 第 4 章包含 AVL 树删除算法的实现，它是一个常常被读者提出的论题。
- 第 5 章已被广泛修订和扩展，现已包含两个更新的算法：杜鹃散列和跳房子散列。此外，还添加了论述通用散列的新的一节。对 C++ 中引入的 `unordered_set` 和 `unordered_map` 两个类模板的简要讨论也是新加的内容。
- 第 6 章基本没有变化，不过，二叉堆的实现用到了 C++11 版引入的移动操作 (move operation)。
- 第 7 章增加了基数排序的内容，并添加了新的一节，论述下界的证明。排序的程序用到 C++11 版中引入的移动操作。
- 第 8 章使用由 Seidel 和 Sharir 所做的新的 union/find 分析，并证明了  $O(M\alpha(M, N))$  的界以代替前面各版较弱的  $O(M\log^*N)$  界。
- 第 12 章添加了论述后缀树和后缀数组的材料，包括 Karkkainen 和 Sanders 的后缀数组线性时间构建算法 (及其实现)。删除了讨论确定性跳跃表和 AA 树的两节。
- 全书所出现的代码均用 C++11 做了更新。值得注意的是，这意味着 C++11 新特性的使用，包括 auto 关键字、范围 for 循环、移动构造和赋值，以及统一初始化 (uniform initialization)。

## 内容提要

第 1 章包含离散数学和递归的一些复习材料。我相信熟练处置递归唯一的办法是反复不断地研读一些好的用法。因此，除第 5 章外，递归遍及本书每一章的例子之中。另外，第 1 章还介绍了一些材料，作为对基本 C++ 的回顾，包括在 C++ 类设计中对模板和一些重要结构的讨论。

第 2 章处理算法分析。这一章阐述渐近分析和它的主要弱点。这里提供了许多例子，包括对对数运行时间的深入解释。通过直观地把一些简单递归程序转变成迭代程序而对它们进行分析。介绍了更复杂的分治程序，不过有些分析 (求解递推关系) 将推迟到第 7 章再详细阐述。

第 3 章包括表、栈和队列。这一章包括对 STL 中 `vector` 类和 `list` 类的讨论，其中涉及到迭代器的一些材料，并提供对 STL 中 `vector` 类和 `list` 类的重要子集的实现。

第 4 章讨论树，重点在查找树，包括外部查找树 (B 树)。UNIX 文件系统和表达式树是作为例子来使用的。本章还介绍了 AVL 树和伸展树。关于查找树实现细节的更详细的处理放在第 12 章介绍。树的另外一些内容，如文件压缩和博弈树，推迟至第 10 章讨论。外部媒体上的数据结构作为几章中的最后论题来考虑。STL 中 `set` 类和 `map` 类的讨论也在本章进行，其中包括一个重要的例子，该例使用 3 个分离的映射高效地解决一个问题。

第 5 章讨论散列表，包括诸如分离链接法以及线性探测法和平方探测法这样一些经典算法，此外还有几个新算法，即杜鹃散列和跳房子散列。通用散列也在这里讨论，而对可扩散列的讨论则放在本章末尾进行。

第 6 章是关于优先队列的。二叉堆也安排在这里，还有些额外的材料论述优先队列某些理论上有趣的实现。斐波那契堆在第 11 章讨论，配对堆在第 12 章讨论。

第 7 章是排序。它是关于编程细节和分析的非常特殊的一章。所有重要的通用排序算法均被讨论并进行了比较。详细分析了 4 种算法：插入排序，希尔排序，堆排序，以及快速排序。本版新增加了基数排序和一些与选择相关问题的下界证明。外部排序的讨论安排在本章的末尾进行。

第 8 章讨论不相交集算法并证明其运行时间。这是短小且特殊的一章，如果不讨论 Kruskal 算法则该章可以跳过。

第 9 章讲授图论算法。图论算法的趣味性不仅因为它们在实践中经常发生，而且还因为它们运行时间强烈地依赖于数据结构的恰当使用。实际上，所有标准算法都是和相应的数据结构、伪代码以及运行时间的分析一起介绍的。为把这些问题放在一个适当的上下文环境下，本章对复杂性理论(包括 NP 完全性和不可判定性)进行了简要的讨论。

第 10 章通过考查一些常见问题的求解技巧来讨论算法设计。这一章通过大量实例而得到强化。这里及后面各章使用的伪代码使得学生对一个算法实例的理解不至于被实现的细节所困扰。

第 11 章处理摊还分析。对来自第 4 章和第 6 章的 3 种数据结构以及本章介绍的斐波那契堆进行了分析。

第 12 章讨论查找树算法、后缀树和后缀数组、 $k$ -d 树、配对堆。不同于其他各章，本章为查找树和配对堆提供了完全和审慎的实现。材料的安排使得教师可以把一些内容整合到其他各章的讨论中。例如，第 12 章中的自顶向下红黑树可以和 AVL 树(第 4 章的)一起讨论。

第 1 章~第 9 章为大多数一学期的数据结构课程提供了足够的材料。如果时间允许，那么第 10 章也可以包括进来。研究生的算法分析课程可以使用第 7 章~第 11 章的内容。在第 11 章所分析的高级数据结构可以容易地在前面各章中查到。第 9 章中对 NP 完全性的讨论太过简单，以至于不足以用于这样的一门算法分析课程。读者将会发现，参阅一些论述 NP 完全性的著述对深化本书内容大有裨益。

## 练习

在每章末尾提供的练习与正文中讲授内容的顺序相匹配。最后的一些练习是把一章作为一个整体来处理而不是针对特定的某一节来考虑的。难做的练习标以一个星号，更难的练习标注两个星号。

## 参考文献

参考文献列于每章的最后。一般来说，这些参考文献或者是历史性质的，代表着书中材料的原始来源，或者阐述对正文中给出结果的扩展和改进。有些文献提供一些练习的解法。

## 补充材料

所有读者均可从网站<http://cssupport.pearsoncmg.com/>上获取下列补充资料：

- 例子程序的源代码<sup>①</sup>
- 勘误表

<sup>①</sup> 也可登录华信教育资源网([www.hxedu.con.cn](http://www.hxedu.con.cn))免费注册下载。



此外, 下列材料只提供给 Pearson Instructor Resource Center ([www.pearsonhighered.com/irc](http://www.pearsonhighered.com/irc)) 上有资格的教师。如欲获取这些资料, 可访问 IRC 或 Pearson Education 销售代表。<sup>①</sup>

- 书中挑选的一些练习的解答
- 本书中的一些图示
- 勘误表

## 致谢

在该丛书几部著作的准备过程中作者得到了许许多多朋友的帮助, 有些人在本书的其他版本中列出。谢谢所有诸位。

如同往常一样, Pearson 专家们的努力使得本书的写作过程更加轻松。我愿意借此机会感谢我的编辑 Tracy Johnson, 以及制作编辑 Marilyn Lloyd。贤妻 Jill 因其所做的每一件工作应该得到我特别的感谢。

最后, 我还要感谢广大的读者, 他们发送 E-mail 信息并指出较早各版的错误和矛盾之处。我的网站 [www.cis.fiu.edu/~weiss](http://www.cis.fiu.edu/~weiss) 也将包含更新后 (C++ 和 Java) 的源代码, 一个勘误表, 以及到提交问题报告的一个链接。

*M. A. W.*  
*Miami, Florida*

---

<sup>①</sup> 教辅申请方式请参见书末的“教学支持说明”。

## 教学支持说明

McGraw-Hill Education, 麦格劳-希尔教育出版公司, 美国著名图书出版与教育服务机构, 以出版经典、高质量的理工科、经济管理、计算机、生命科学以及人文社科类高校教材享誉全球, 更以丰富的网络化、数字化教学辅助资源深受高校教师的欢迎。

为了更好地服务于中国教育发展, 提升教学质量, 2003年麦格劳-希尔教师服务中心在北京成立。在您确认将本书作为指定教材后, 请您填好以下表格并经系主任签字盖章后寄回, 麦格劳-希尔教师服务中心将免费向您提供相应教学课件或网络化课程管理资源。如果您需要订购或参阅本书的英文原版, 我们也会竭诚为您服务。

### 证 明

兹证明\_\_\_\_\_大学\_\_\_\_\_系/院\_\_\_\_\_专业  
\_\_\_\_\_学年(学期)开设的\_\_\_\_\_课程, 共\_\_\_\_\_学时, 现采用电子工业出版社出版的英文原版/简体中文版\_\_\_\_\_ (书名/作者)\_\_\_\_\_作为主要教材。任课教师为\_\_\_\_\_, 学生\_\_\_\_\_个班共\_\_\_\_\_人。

任课教师需要与本书配套的教师指导手册和习题解答。

电 话: \_\_\_\_\_

传 真: \_\_\_\_\_

E-mail: \_\_\_\_\_

联系地址: \_\_\_\_\_

邮 编: \_\_\_\_\_

建议和要求:

系/院主任: \_\_\_\_\_ (签字)

(系/院办公室章)

\_\_\_\_\_年\_\_\_\_\_月\_\_\_\_\_日

### 请与我们联系

Publishing House of Electronics Industry

电子工业出版社: [www.phei.com.cn](http://www.phei.com.cn)

[www.hxedu.com.cn](http://www.hxedu.com.cn)

联系电话: 010-88254555

传 真: 010-88254560

E-mail: [Te\\_service@phei.com.cn](mailto:Te_service@phei.com.cn)

麦格劳-希尔教育出版公司教师服务中心

北京市海淀区 清华科技园 创业大厦 907 室

北京 100084

传真: 8610-62790292

教师服务热线: 800-810-1936

教师服务信箱: [instructor\\_cn@mcgraw-hill.com](mailto:instructor_cn@mcgraw-hill.com)

网址: <http://www.mcgraw-hill.com.cn>



## 反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：(010) 88254396；(010) 88258888

传 真：(010) 88254397

E-mail: [dbqq@phei.com.cn](mailto:dbqq@phei.com.cn)

通信地址：北京市海淀区万寿路 173 信箱

电子工业出版社总编办公室

邮 编：100036

# 目 录

|   |    |                          |    |
|---|----|--------------------------|----|
| 第 1 章 程序设计：综述   | 1  | 1.7.1 数据成员、构造函数和基本访问函数   | 38 |
| 1.1 本书讨论的内容   | 1  | 1.7.2 operator[ ]        | 38 |
| 1.2 数学知识复习  | 2  | 1.7.3 五大函数               | 39 |
| 1.2.1 指数(exponent)  | 2  | 小结                       | 39 |
| 1.2.2 对数(logarithm)   | 2  | 练习                       | 39 |
| 1.2.3 级数(series)  | 3  | 参考文献                     | 41 |
| 1.2.4 模运算(modular arithmetic)                               | 4  | 第 2 章 算法分析               | 42 |
| 1.2.5 证明方法  | 5  | 2.1 数学基础                 | 42 |
| 1.3 递归简论  | 7  | 2.2 模型                   | 44 |
| 1.4 C++类  | 10 | 2.3 要分析的问题               | 44 |
| 1.4.1 基本的 class 语法  | 10 | 2.4 运行时间计算               | 47 |
| 1.4.2 构造函数的附加语法和访问函数  | 11 | 2.4.1 一个简单的例子            | 47 |
| 1.4.3 接口与实现的分离  | 13 | 2.4.2 一般法则               | 47 |
| 1.4.4 vector 类和 string 类                                    | 16 | 2.4.3 最大子序列和问题的求解        | 49 |
| 1.5 C++细节   | 17 | 2.4.4 运行时间中的对数           | 54 |
| 1.5.1 指针(pointer)   | 18 | 2.4.5 最坏情形分析的局限性         | 57 |
| 1.5.2 左值、右值和引用  | 19 | 小结                       | 58 |
| 1.5.3 参数传递  | 21 | 练习                       | 58 |
| 1.5.4 返回值传递   | 23 | 参考文献                     | 63 |
| 1.5.5 std::swap 和 std::move                                 | 25 | 第 3 章 表、栈和队列             | 64 |
| 1.5.6 五大函数：析构函数，拷贝构造函数，移动构造函数，拷贝赋值 operator=，移动赋值 operator= | 26 | 3.1 抽象数据类型(ADT)          | 64 |
| 1.5.7 C 风格数组和字符串  | 30 | 3.2 表 ADT                | 64 |
| 1.6 模板  | 31 | 3.2.1 表的简单数组实现           | 65 |
| 1.6.1 函数模板  | 31 | 3.2.2 简单链表               | 65 |
| 1.6.2 类模板   | 32 | 3.3 STL 中的 vector 和 list | 67 |
| 1.6.3 Object、Comparable 和一个例子                               | 33 | 3.3.1 迭代器                | 68 |
| 1.6.4 函数对象  | 34 | 3.3.2 例子：对表使用 erase      | 69 |
| 1.6.5 类模板的分离式编译   | 37 | 3.3.3 const_iterators    | 70 |
| 1.7 使用矩阵  | 37 | 3.4 vector 的实现           | 72 |
|   |    | 3.5 list 的实现             | 76 |
|   |    | 3.6 栈 ADT                | 86 |

|                           |     |                         |     |
|---------------------------|-----|-------------------------|-----|
| 3.6.1 栈模型                 | 86  | 练习                      | 147 |
| 3.6.2 栈的实现                | 86  | 参考文献                    | 153 |
| 3.6.3 应用                  | 87  | <b>第 5 章 散列</b>         | 155 |
| 3.7 队列 ADT                | 93  | 5.1 一般想法                | 155 |
| 3.7.1 队列模型                | 93  | 5.2 散列函数                | 155 |
| 3.7.2 队列的数组实现             | 93  | 5.3 分离链接法               | 157 |
| 3.7.3 队列的应用               | 95  | 5.4 不用链表的散列表            | 161 |
| 小结                        | 96  | 5.4.1 线性探测法             | 161 |
| 练习                        | 96  | 5.4.2 平方探测法             | 163 |
| <b>第 4 章 树</b>            | 100 | 5.4.3 双散列               | 166 |
| 4.1 预备知识                  | 100 | 5.5 再散列                 | 167 |
| 4.1.1 树的实现                | 101 | 5.6 标准库中的散列表            | 169 |
| 4.1.2 树的遍历及应用             | 102 | 5.7 以最坏情形 $O(1)$ 访问的散列表 | 170 |
| 4.2 二叉树                   | 105 | 5.7.1 完美散列              | 170 |
| 4.2.1 实现                  | 105 | 5.7.2 杜鹃散列              | 172 |
| 4.2.2 一个例子——表达式树          | 105 | 5.7.3 跳房子散列             | 181 |
| 4.3 查找树 ADT——二叉查找树        | 108 | 5.8 通用散列                | 184 |
| 4.3.1 contains            | 110 | 5.9 可扩展散列               | 186 |
| 4.3.2 findMin 和 findMax   | 111 | 小结                      | 188 |
| 4.3.3 insert              | 112 | 练习                      | 189 |
| 4.3.4 remove              | 113 | 参考文献                    | 193 |
| 4.3.5 析构函数和拷贝构造函数         | 115 | <b>第 6 章 优先队列(堆)</b>    | 196 |
| 4.3.6 平均情况分析              | 115 | 6.1 模型                  | 196 |
| 4.4 AVL 树                 | 118 | 6.2 一些简单的实现             | 197 |
| 4.4.1 单旋转                 | 119 | 6.3 二叉堆                 | 197 |
| 4.4.2 双旋转                 | 121 | 6.3.1 结构性性质             | 197 |
| 4.5 伸展树                   | 128 | 6.3.2 堆序性质              | 198 |
| 4.5.1 一个简单的想法(不能直接<br>使用) | 128 | 6.3.3 基本的堆操作            | 199 |
| 4.5.2 展开                  | 130 | 6.3.4 其他的堆操作            | 203 |
| 4.6 树的遍历                  | 134 | 6.4 优先队列的应用             | 206 |
| 4.7 B 树                   | 135 | 6.4.1 选择问题              | 206 |
| 4.8 标准库中的容器 set 和 map     | 140 | 6.4.2 事件模拟              | 207 |
| 4.8.1 集合容器 set            | 140 | 6.5 d 堆                 | 208 |
| 4.8.2 映射容器 map            | 141 | 6.6 左式堆                 | 209 |
| 4.8.3 set 和 map 的实现       | 142 | 6.6.1 左式堆的性质            | 209 |
| 4.8.4 使用多个 map 的示例        | 142 | 6.6.2 左式堆操作             | 210 |
| 小结                        | 147 | 6.7 斜堆                  | 215 |
|                           |     | 6.8 二项队列                | 216 |

|              |                                  |            |              |                         |            |
|--------------|----------------------------------|------------|--------------|-------------------------|------------|
| 6.8.1        | 二项队列构建                           | 216        | 7.12.3       | 简单算法                    | 269        |
| 6.8.2        | 二项队列操作                           | 217        | 7.12.4       | 多路合并                    | 270        |
| 6.8.3        | 二项队列的实现                          | 219        | 7.12.5       | 多相合并                    | 271        |
| 6.9          | 标准库中的优先队列                        | 224        | 7.12.6       | 替换选择                    | 272        |
|              | 小结                               | 225        |              | 小结                      | 273        |
|              | 练习                               | 225        |              | 练习题                     | 273        |
|              | 参考文献                             | 229        |              | 参考文献                    | 278        |
| <b>第 7 章</b> | <b>排序</b>                        | <b>232</b> | <b>第 8 章</b> | <b>不相交集类</b>            | <b>281</b> |
| 7.1          | 预备知识                             | 232        | 8.1          | 等价关系                    | 281        |
| 7.2          | 插入排序                             | 233        | 8.2          | 动态等价性问题                 | 281        |
| 7.2.1        | 算法                               | 233        | 8.3          | 基本数据结构                  | 283        |
| 7.2.2        | 插入排序的 STL 实现                     | 233        | 8.4          | 灵巧求并算法                  | 286        |
| 7.2.3        | 插入排序的分析                          | 235        | 8.5          | 路径压缩                    | 288        |
| 7.3          | 一些简单排序算法的下界                      | 235        | 8.6          | 按秩求并和路径压缩的最坏情形          | 289        |
| 7.4          | 希尔排序                             | 236        | 8.6.1        | 缓慢增长的函数                 | 289        |
| 7.4.1        | 希尔排序的最坏情形分析                      | 237        | 8.6.2        | 通过递归分解进行的分析             | 290        |
| 7.5          | 堆排序                              | 239        | 8.6.3        | 一个 $O(M \log^* N)$ 界    | 295        |
| 7.5.1        | 堆排序的分析                           | 241        | 8.6.4        | 一个 $O(M\alpha(M, N))$ 界 | 296        |
| 7.6          | 归并排序                             | 242        | 8.7          | 一个应用                    | 297        |
| 7.6.1        | 归并排序的分析                          | 245        |              | 小结                      | 299        |
| 7.7          | 快速排序                             | 247        |              | 练习                      | 299        |
| 7.7.1        | 选取枢纽元                            | 249        |              | 参考文献                    | 301        |
| 7.7.2        | 分割策略                             | 250        | <b>第 9 章</b> | <b>图论算法</b>             | <b>303</b> |
| 7.7.3        | 小数组                              | 252        | 9.1          | 若干定义                    | 303        |
| 7.7.4        | 实际的快速排序例程                        | 252        | 9.1.1        | 图的表示                    | 304        |
| 7.7.5        | 快速排序的分析                          | 254        | 9.2          | 拓扑排序                    | 305        |
| 7.7.6        | 选择问题的线性期望时间<br>算法                | 256        | 9.3          | 最短路径算法                  | 308        |
| 7.8          | 排序算法的一般下界                        | 258        | 9.3.1        | 无权最短路径                  | 309        |
| 7.8.1        | 决策树                              | 258        | 9.3.2        | Dijkstra 算法             | 312        |
| 7.9          | 选择问题的决策树下界                       | 260        | 9.3.3        | 具有负边值的图                 | 317        |
| 7.10         | 对手下界 (adversary lower<br>bounds) | 262        | 9.3.4        | 无圈图                     | 318        |
| 7.11         | 线性时间排序: 桶式排序和<br>基数排序            | 265        | 9.3.5        | 所有顶点对间的最短路径             | 320        |
| 7.12         | 外部排序                             | 269        | 9.3.6        | 最短路径的例                  | 320        |
| 7.12.1       | 为什么需要一些新的算法                      | 269        | 9.4          | 网络流问题                   | 322        |
| 7.12.2       | 外部排序模型                           | 269        | 9.4.1        | 一个简单的最大流算法              | 323        |
|              |                                  |            | 9.5          | 最小生成树                   | 326        |
|              |                                  |            | 9.5.1        | Prim 算法                 | 327        |

|        |             |     |                            |     |
|--------|-------------|-----|----------------------------|-----|
| 9.5.2  | Kruskal 算法  | 329 | 小结                         | 405 |
| 9.6    | 深度优先搜索的应用   | 330 | 练习                         | 406 |
| 9.6.1  | 无向图         | 331 | 参考文献                       | 413 |
| 9.6.2  | 双连通性        | 332 | 第 11 章 摊还分析                | 418 |
| 9.6.3  | 欧拉回路        | 335 | 11.1 一个无关的智力问题             | 418 |
| 9.6.4  | 有向图         | 338 | 11.2 二项队列                  | 419 |
| 9.6.5  | 查找强分支       | 339 | 11.3 斜堆                    | 423 |
| 9.7    | NP 完全性介绍    | 340 | 11.4 斐波那契堆                 | 425 |
| 9.7.1  | 难与易         | 341 | 11.4.1 切除左式堆中的节点           | 425 |
| 9.7.2  | NP 类        | 341 | 11.4.2 二项队列的懒惰合并           | 427 |
| 9.7.3  | NP 完全问题     | 342 | 11.4.3 斐波那契堆操作             | 429 |
| 小结     |             | 344 | 11.4.4 时间界的证明              | 430 |
| 练习     |             | 344 | 11.5 伸展树                   | 432 |
| 参考文献   |             | 350 | 小结                         | 436 |
| 第 10 章 | 算法设计技巧      | 353 | 练习                         | 436 |
| 10.1   | 贪婪算法        | 353 | 参考文献                       | 437 |
| 10.1.1 | 一个简单的调度问题   | 354 | 第 12 章 高级数据结构及其实现          | 439 |
| 10.1.2 | 哈夫曼编码       | 355 | 12.1 自顶向下伸展树               | 439 |
| 10.1.3 | 近似装箱问题      | 359 | 12.2 红黑树                   | 445 |
| 10.2   | 分治算法        | 366 | 12.2.1 自底向上的插入             | 446 |
| 10.2.1 | 分治算法的运行时间   | 367 | 12.2.2 自顶向下红黑树             | 447 |
| 10.2.2 | 最近点问题       | 369 | 12.2.3 自顶向下删除              | 452 |
| 10.2.3 | 选择问题        | 371 | 12.3 treap 树               | 453 |
| 10.2.4 | 一些算术问题的理论改进 | 374 | 12.4 后缀数组和后缀树              | 456 |
| 10.3   | 动态规划        | 377 | 12.4.1 后缀数组                | 456 |
| 10.3.1 | 用表代替递归      | 377 | 12.4.2 后缀树                 | 458 |
| 10.3.2 | 矩阵乘法的顺序安排   | 379 | 12.4.3 后缀数组和后缀树的线性<br>时间构建 | 461 |
| 10.3.3 | 最优二叉查找树     | 382 | 12.5 $k$ -d 树              | 471 |
| 10.3.4 | 所有点对最短路径    | 384 | 12.6 配对堆                   | 474 |
| 10.4   | 随机化算法       | 386 | 小结                         | 479 |
| 10.4.1 | 随机数发生器      | 387 | 练习                         | 479 |
| 10.4.2 | 跳跃表         | 392 | 参考文献                       | 483 |
| 10.4.3 | 素性测试        | 393 | 附录 A 类模板的分离式编译             | 486 |
| 10.5   | 回溯算法        | 396 | 索引                         | 489 |
| 10.5.1 | 收费公路重建问题    | 396 |                            |     |
| 10.5.2 | 博弈          | 400 |                            |     |

# 第 1 章 程序设计：综述

我们在这一章里阐述本书的目的和目标，并简要复习离散数学以及程序设计的一些概念。我们将要

- 看到程序对于合理的大量输入的运行性能与其在适量输入下运行性能的同等重要性。
- 概括本书其余部分所需要的基本的数学基础。
- 简要复习递归。
- 概括用于本书的 C++ 语言的某些重要特点。

## 1.1 本书讨论的内容

设有一组  $N$  个数而要确定其中第  $k$  个最大者，我们称之为选择问题(selection problem)。大多数学习过一两门程序设计课程的学生编写一个解决这种问题的程序不会有什么困难。“直观的”解决方法是相当多的。

该问题的一种解法就是将这  $N$  个数读进一个数组中，再通过某种简单的算法，比如冒泡排序法(bubble sort)，以递减顺序将数组排序，然后返回位置  $k$  上的元素。

稍微好一点的算法可以先把前  $k$  个元素读入数组并(以递减的顺序)对其排序。然后，将剩下的元素再逐个读入。当新元素被读到时，如果它小于数组中的第  $k$  个元素则忽略之，否则就将其放到数组中的正确位置上，同时将原数组中的最后一个元素挤出数组。当算法终止时，位于第  $k$  个位置上的元素作为答案返回。

这两种算法编码都很简单，建议读者试一试。此时我们自然要问：哪个算法更好？而更重要的是，两个算法都足够好吗？使用 3000 万个元素的随机文件和  $k=15\,000\,000$  进行模拟指出，两个算法在合理的时间量内均不能结束计算；每种算法都需要计算机处理若干天才能算完(不过最终还是给出了正确的答案)。在第 7 章中将讨论另一种算法，该算法将在 1 秒种左右给出问题的解。因此，虽然我们提出的两个算法都能算出结果，但是它们不能看作是好的算法，因为对于第三种算法能够在合理的时间内处理的输入数据量而言，这两种算法是完全不切实际的。

第二个问题是解决一个流行的字谜(word puzzle)游戏。输入由一些字母构成的二维数组和一个单词表组成。目标是要找出字谜中的单词，这些单词可能是水平、垂直或在对角线上以任何方向放置的。作为例子，图 1.1 所示的字谜由单词 this、two、fat 和 that 组成。单词 this 从第一行第一列的位置即(1, 1)处开始并延伸至(1, 4)；单词 two 从(1, 1)到(3, 1)；fat 从(4,1)到(2, 3)；而 that 则从(4, 4)到(1, 1)。

|   | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | t | h | i | s |
| 2 | w | a | t | s |
| 3 | o | a | h | g |
| 4 | f | g | d | t |

图 1.1 字谜示例

现在至少也有两种直观的算法来求解这个问题。对单词表中的每个单词，我们检查每一个有序三元组(行，列，方向)，验证是否有单词存在。这需要大量嵌套的 for 循环，但它基本上是直观的算法。

也可以这样,对于每一个尚未越出迷板边缘的有序四元组(行,列,方向,字符数),我们可以检测是否所指的单词在单词表中。这也导致使用大量嵌套的 for 循环。如果任意单词中的最大字符数已知,那么该算法有可能节省一些时间。

上述两种方法相对来说都不难编码,并可求解通常发表于杂志上的许多现实的字谜游戏。这些字谜通常有 16 行、16 列以及 40 个左右的单词。然而,假设我们把字谜变成只给出迷板(puzzle board)而单词表基本上是一本英语词典,则上面提出的两种解法均需要相当可观的时间来解决这个字谜问题,因而这两种方法都是不可接受的。不过,这样的问题仍有可能快速解决,甚至单词表可以很大。

一个重要的观念是,在许多问题中,只写出一个工作的程序并不够。如果这个程序在大数据集上运行,那么运行时间就成了重要的问题。我们将在本书看到对于大量的输入如何估计程序的运行时间,尤其重要的是,如何在尚未具体编码的情况下比较两个程序的运行时间。我们还将看到显著改进程序速度以及确定程序瓶颈的方法,这些方法将使我们能够找到那些需要集中精力去努力优化的代码段。

## 1.2 数学知识复习

这一节列出一些需要读者记忆或是能够推导出的基本公式,并复习基本的证明方法。

### 1.2.1 指数(exponent)

$$X^A X^B = X^{A+B}$$

$$\frac{X^A}{X^B} = X^{A-B}$$

$$(X^A)^B = X^{AB}$$

$$X^N + X^N = 2X^N \neq X^{2N}$$

$$2^N + 2^N = 2^{N+1}$$

### 1.2.2 对数(logarithm)

在计算机科学中,除非有特别的声明,所有的对数都是以 2 为底的。

#### 定义 1.1

$X^A = B$ , 当且仅当  $\log_X B = A$ 。

由该定义可以得到几个方便的等式。

#### 定理 1.1

$$\log_A B = \frac{\log_C B}{\log_C A}; \quad A, B, C > 0, A \neq 1$$

证明:

令  $X = \log_C B$ ,  $Y = \log_C A$ , 以及  $Z = \log_A B$ 。此时由对数的定义,  $C^X = B$ ,  $C^Y = A$ , 以及  $A^Z = B$ 。联合这三个等式则得到  $B = C^X = (C^Y)^Z$ 。因此,  $X = YZ$ , 这意味着  $Z = X/Y$ , 定理得证。  $\square$



## 定理 1.2

$$\log AB = \log A + \log B; A, B > 0$$

证明:

令  $X = \log A$ ,  $Y = \log B$ , 以及  $Z = \log AB$ 。此时由于假设默认的底为 2,  $2^X = A$ ,  $2^Y = B$ , 以及  $2^Z = AB$ 。联合最后的三个等式则有  $2^X 2^Y = 2^Z = AB$ 。因此  $X + Y = Z$ , 从而证明了该定理。 □

其他一些有用的公式如下, 它们都能够用类似的方法推导。

$$\log A/B = \log A - \log B$$

$$\log(A^B) = B \log A$$

$\log X < X$  对所有的  $X > 0$  成立。

$$\log 1 = 0, \log 2 = 1, \log 1024 = 10, \log 1048576 = 20.$$

## 1.2.3 级数(series)

最容易记忆的公式是

$$\sum_{i=0}^N 2^i = 2^{N+1} - 1$$

和

$$\sum_{i=0}^N A^i = \frac{A^{N+1} - 1}{A - 1}$$

在第二个公式中, 如果  $0 < A < 1$ , 则

$$\sum_{i=0}^N A^i \leq \frac{1}{1 - A}$$

当  $N$  趋于  $\infty$  时该和趋向于  $1/(1-A)$ 。这些公式是“几何级数(geometric series)”公式。

我们可以用下面的方法推导关于  $\sum_{i=0}^{\infty} A^i$  ( $0 < A < 1$ ) 的公式。令  $S$  是其和, 此时

$$S = 1 + A + A^2 + A^3 + A^4 + A^5 + \dots$$

于是

$$AS = A + A^2 + A^3 + A^4 + A^5 + \dots$$

如果将这两个方程相减(这种运算只允许对收敛级数进行), 等号右边所有的项相消, 只留下 1:

$$S - AS = 1$$

这就是说

$$S = \frac{1}{1 - A}$$

可以用相同的方法计算  $\sum_{i=1}^{\infty} i/2^i$ , 它是一个经常出现的和。我们把它写成

$$S = \frac{1}{2} + \frac{2}{2^2} + \frac{3}{2^3} + \frac{4}{2^4} + \frac{5}{2^5} + \dots$$

用 2 乘两边得到

$$2S = 1 + \frac{2}{2} + \frac{3}{2^2} + \frac{4}{2^3} + \frac{5}{2^4} + \frac{6}{2^5} + \dots$$

将这两个等式相减得到

$$S = 1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^5} + \dots$$

因此,  $S = 2$ 。

分析中另一种常用类型的级数是**算术级数**(arithmetic series)。任何这样的级数都可以从下面的基本公式计算其值:

$$\sum_{i=1}^N i = \frac{N(N+1)}{2} \approx \frac{N^2}{2}$$

例如, 为求出和  $2 + 5 + 8 + \dots + (3k-1)$ , 将其改写为  $3(1+2+3+\dots+k) - (1+1+1+\dots+1)$ , 显然, 它就是  $3k(k+1)/2 - k$ 。另一种记忆的方法则是将第一项与最后一项相加(和为  $3k+1$ ), 第二项与倒数第二项相加(和也是  $3k+1$ ), 等等。由于有  $k/2$  个这样的数对, 因此总和就是  $k(3k+1)/2$ , 这与前面的答案相同。

现在介绍下面两个公式, 不过它们就没有那么常见了。

$$\sum_{i=1}^N i^2 = \frac{N(N+1)(2N+1)}{6} \approx \frac{N^3}{3}$$

$$\sum_{i=1}^N i^k \approx \frac{N^{k+1}}{|k+1|}, \quad k \neq -1$$

当  $k = -1$  时, 后一个公式不成立。此时我们需要下面的公式, 这个公式在计算机科学中的使用要远比在数学其他学科中用得频繁。数  $H_N$  叫作**调和数**(harmonic number), 其和叫作**调和和**(harmonic sum)。下面近似式中的误差趋向于  $\gamma \approx 0.577\ 215\ 66$ , 称为**欧拉常数**(Euler's constant)。

$$H_N = \sum_{i=1}^N \frac{1}{i} \approx \log_e N$$

以下两个公式只不过是一般的代数运算:

$$\sum_{i=1}^N f(N) = Nf(N)$$

$$\sum_{i=n_0}^N f(i) = \sum_{i=1}^N f(i) - \sum_{i=1}^{n_0-1} f(i)$$

### 1.2.4 模运算(modular arithmetic)

如果  $N$  整除  $A-B$ , 那么我们就说  $A$  与  $B$  模  $N$  同余(congruent), 记为  $A \equiv B \pmod{N}$ 。直观地看, 这意味着无论  $A$  还是  $B$  被  $N$  去除, 所得余数都是相同的。于是,  $81 \equiv 61 \equiv 1 \pmod{10}$ 。如同等式的情形一样, 若  $A \equiv B \pmod{N}$ , 则  $A+C \equiv B+C \pmod{N}$ , 以及  $AD \equiv BD \pmod{N}$ 。