HZ BOOKS
华章教育

高等院校精品课程系列教材·国家级

数据结构与算法设计

王晓东◎著

精品课程

and Algorithm Design

机械工业出版社
China Machine Press

高等院校精品课程系列教材·国家级

数据结构与算法设计

王晓东◎著

精品课程

ata Structures
and Algorithm Design

机械工业出版社
China Machine Press

本书以基本数据结构为知识单元，系统地介绍数据结构知识与应用、计算机算法的设计与分析方法。全书共分13章，第1章介绍数据结构、抽象数据类型和算法的基本概念；第2～4章以抽象数据类型为主线索，围绕常用的基本数据结构，分别介绍基于序列的常用数据结构表、栈、队列；第5章介绍递归以及递归在数据结构和算法设计中的应用；第6章介绍实际应用中常用的排序与选择算法；第7～12章讨论反映层次关系的数据结构树、表示集合的抽象数据类型、符号表以及散列表等实践中常用实现符号表的方法、字典及其实现方法、优先队列及其实现方法、并查集及其实现方法；第13章介绍非线性结构图及图的算法。

本书内容丰富，观点新颖，理论联系实际，不仅可用做高等院校计算机科学与工程专业学生学习数据结构与算法的教材，而且也适合广大工程技术人员参考。

封底无防伪标均为盗版
版权所有，侵权必究
本书法律顾问　北京市展达律师事务所

图书在版编目(CIP)数据

数据结构与算法设计／王晓东著．—北京：机械工业出版社，2012.7
（高等院校精品课程系列教材·国家级）

ISBN 978-7-111-37924-9

Ⅰ.数…　Ⅱ.王…　Ⅲ.①数据结构—高等学校—教材　②电子计算机—算法设计—高等学校—教材　Ⅳ.①TP311.12　②TP301.6

中国版本图书馆CIP数据核字(2012)第059682号

机械工业出版社(北京市西城区百万庄大街22号　邮政编码　100037)
责任编辑：曾　珊
中国电影出版社印刷厂印刷
2012年7月第1版第1次印刷
185mm×260mm·16印张
标准书号：ISBN 978-7-111-37924-9
定价：29.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：(010)88378991；88361066
购书热线：(010)68326294；88379649；68995259
投稿热线：(010)88379604
读者信箱：hzjsj@hzbook.com

出版者的话

机械工业出版社华章公司秉承“全球采集内容，服务中国教育”的理念，经过十余年的不懈努力，引进、翻译、出版了大量在计算机科学界、电子科学界享有盛名的专家名著与名校教材，其中包括 Knuth、Aho、Jim Gray、Ullman、Baker、Guru 等大师名家的一批经典作品，这些作品对国内计算机教育事业的发展起到了一定的推动作用。今天，全国高等学校精品课程建设工作的蓬勃开展为我们更好地服务于计算机教育、电子信息科学教育提供了良好的契机，我们将以严谨的治学态度及全面服务的专业出版精神，在国内广大院校教师们的支持与帮助下，陆续推出具有国内一流教学水平的“高等院校精品课程系列教材”。

精品课程是具有一流教师队伍、一流教学内容、一流教学方法、一流教材、一流教学管理等特点的示范性课程，是教育部实施的“高等学校教学质量与教学改革工程”的重要组成部分，是教育部深化教学改革，以教育信息化带动教育现代化的一项重要举措。自 2003 年精品课程建设项目持续推进以来，国内高校中的优秀教师纷纷在总结本校富有历史传统而又特色突出的课程教学方法与经验成绩的基础上，充分运用现代网络传播技术将优质的教学资源上网共享，使国内其他高校在实施同类课程教学的过程中能够借鉴、使用这些优质的教学资源，在更大范围内提高高等学校的教学和人才培养质量，提升我国高等教育的综合实力和国际竞争能力。经过几年的共同努力，已经建立起了较为齐全的各门类及各专业的校、省、国家三级精品课程体系，期间先后有总计 750 门课程通过了专家评审，获得了“国家精品课程”称号。

这些各个层次的“精品课程”建设过程都比较充分地体现了教育部所要求的七个重点，即：具有科学的建设规划，配备高水平的教学队伍，不断进行教学内容和课程体系的改革，使用先进的教学方法和手段，注重建设系列化的优秀教材，高度重视理论与实践两个环节，切实激励各方人员共同参与。也正因为这样的多方面积极参与，使得我国的高等教育在近年来由精英教育转向大众教育的跨越式发展中取得了教学质量上的突破与飞跃。精品课教材作为精品课程的要件之一，比以往教材更加具有实践检验性，教学辅助资源经过不断地更新与补充更加丰富，是精品课教学团队智慧的共同体现。

“师者，所以传道授业解惑也”。教材是体现教学内容和教学要求的知识载体，是教师进行教学活动的基本工具，是提高教学质量的重要保证。精品课程教学团队中优秀的教师们集多年治学经验撰写出版相关教材，也是精品课程建设的一个重要方面。华章作为专业的出版团队，长久以来以“传承专业知识精华，服务中国教育事业”为使命，遵循“分享、专业、创新”的价值观，实践着“国际视野、专业出版、教育为本、科学管理”的出版方针，愿与高等院校的教师共同携手，为中国的高等教育事业走向国际化而努力。

为更好地服务于精品课程配套教材的出版，华章不仅密切关注高校的优秀课程建设，而且还将利用自身的优势帮助教师完善课程设置、提供教辅资料、准备晋级申报、推广教学经验。具体详情可访问专门网站 http://www.hzbook.com/jpkc.aspx，并可在线填写出版申请，欢迎您对我们的工作给予帮助和指导。

投稿专线：010-88379604
投稿 E-mail：hzjsj@ hzbook.com

华章科技图书出版中心

试读结束：需要全本请在线购买： www.ertongbook.com

前言

为了适应培养计算机各类人才的需要，结合我国高等学校教育工作的现状，立足培养学生跟上国际计算机科学技术的发展水平，更新教学内容和教学方法，本书以基本数据结构和算法设计策略为知识单元，系统地介绍数据结构知识与应用、计算机算法的设计与分析方法，以期为计算机学科的学生提供广泛、坚实的数据结构与算法设计基础知识。

全书共分13章，第1章介绍数据结构、抽象数据类型和算法的基本概念，并对算法的计算复杂性和算法的描述作了简要的阐述。然后以抽象数据类型为主线索，围绕设计算法常用的基本数据结构和基本设计策略组织了第2~13章的内容。

第2~4章依次介绍基于序列的抽象数据类型——表、栈和队列。

第5章介绍递归的概念，以及递归在数据结构和算法设计中的广泛应用，并详细阐述分治法、动态规划和回溯法这3个实践中常用的使用递归技术的算法设计策略。

第6章介绍在实际应用中常用的排序与选择算法。

第7章讨论反映层次关系的抽象数据类型——树。

第8章讨论表示集合的抽象数据类型。

第9章讨论符号表以及散列表等实践中常用实现符号表的方法。

第10章的主题是以有序集为基础的字典及其实现方法，讨论实现字典的二叉搜索树以及AVL树等高效算法。

第11章讨论以集合为基础的优先队列及其实现方法。

第12章讨论以不相交的集合为基础的并查集及其实现方法。

第13章介绍非线性结构图及图的算法。

为了加深对知识的理解，各章均配有难易适当的习题，以适应不同程度读者练习的需要；同时在每章的最后配有相应的算法实验。

由于作者的知识和写作水平有限，书稿虽几经修改，仍难免有缺点和错误，热忱欢迎同行专家和读者批评指正，以使本书不断改进、日臻完善。

教学方案设计

教学思想概述

社会不同层次的人才所需要的知识结构和能力不尽相同，因此任何一个学科必须根据毕业生未来在社会中的地位和作用，确定学生的培养目标。学校的一切教学活动都应该围绕着确定的培养目标进行，教学计划是学科教育的纲领性文件，更应该围绕培养目标来安排。而一门课程的教学，应该围绕着培养总目标进行。同一门课，对于不同类型的学生，虽然基本内容一样，但应该有不同的重点，不同的组织方法，甚至应该有不同的讲授与学习风格。在课程教学中，教材固然起着非常重要的作用，但它只能提供对某门课程基本内容的描述，更重要的是教师对教学内容的组织。教师对课程教育思想的理解与贯彻，会强烈地影响课程的教学效果。对同一门课程，不同学校的教学会产生不同的效果，不同的教师也会讲出不同的内容，对基础性比较强的课程来说更是如此。

为了对计算机学科的教育进行较全面的引导，美国 IEEE-CS/ACM 在 1991 年发布了《Computing Curricula 1991》(计算学科教程 1991，简称 CC91)。10 年之后(2001 年)又发布了《Computing Curricula CS2001》(计算学科教程—CS2001，简称 CC2001)。根据学科的发展及社会对计算机人才的要求，中国计算机学会教育专业委员会、全国高校计算机教育研究会和清华大学出版社联合组织了计算机学科教学计划的研究。2002 年 8 月，在第 5 届全国高校计算机系主任论坛上正式发布了《中国计算机科学与技术学科教程 2002》(简称 CCC2002)。“数据结构与算法”教学设计就是根据 CC2001—CS 和 CCC2002 的基本要求，按照总学时 76、每学时 50 分钟的要求制定的。实际上，各高校可以根据学生的具体情况作适当调整，建议最少 48 学时，最多 80 学时。

课程教学体系

1. 课程描述

(1) 课程名称

课程名称：数据结构与算法。

课程英文名称为：Data Structures and Algorithms。

(2) 课程性质

“数据结构与算法”是计算机学科的技术基础和主干必修课。

(3) 基础知识要求

课程“数据结构与算法”不需要特别的基础知识，第1章中对有关的内容有一定的介绍。但为了更好地理解教材的内容，应该先修数学分析(或者高等数学)、离散数学和C语言程序设计。

(4) 学时安排

总学时76，课堂授课60学时，习题课8学时，上机实践8学时。其中，课堂授课部分第1～13章的学时安排分别为2、4、4、4、6*、6、4、4、4*、6*、6*、4*、6学时。(学时数有*号者可以根据具体情况进行删减。)

2. 教学目标定位

(1) 算法设计和实现能力的培养

计算机科学与技术专业的人员应该具有4种基本的专业能力：计算思维能力，算法设计与分析能力，程序设计和实现能力，计算机软硬件系统的认知、分析、设计与应用能力。本课程着重培养学生的算法设计与分析能力、程序设计和实现能力。

(2) 主要特点

本课程的主要特点是抽象和形式化，既有严格的理论证明，又具有很强的构造性和应用性，包括一些基本数据结构、数据结构的建立和性质等。课程以抽象数据类型为主线索，围绕设计算法常用的基本数据结构和基本设计策略组织教学内容。它不仅是计算机科学教育后续课程的理论基础，而且还广泛地用于新兴的技术和研究领域。软件工程专业的学生更要注重抽象以及抽象描述下的构造思想和方法。

(3) 教学定位

一些著名的计算机科学家在有关计算机科学教育的论述中认为，计算机科学是一种创造性思维活动，其教育必须面向设计。“数据结构与算法”正是一门面向设计且处于计算机学科核心地位的教育课程。通过对数据结构与算法的系统学习与研究，理解和掌握算法设计的主要方法，培养对算法的计算复杂性进行正确分析的能力，可以为独立地设计算法和对给定算法进行复杂性分析奠定坚实的理论基础，这对于从事计算机系统结构、系统软件和应用软件研究与开发的科技工作者来说是非常重要和必不可少的。

3. 知识点与教学要求

见各章章首的“学习要点”及章尾的“本章小结”。

4. 习题与实验

(1) 指导思想

课后作业需要学生综合运用教师在课堂上讲述的方法(包括思维方法)独立思考求解

问题，以深化对课堂讲述内容的理解。教师在布置作业后，要留给学生足够多的独立思考、独立求解的时间。一定要鼓励学生自己去想问题，亲身体验解题过程。本课程的习题一般都有一定难度，因此，授课教师应该在适当的时候安排习题课，选择典型的习题，讲解典型的解题思路和方法，注意重点讲算法思路和典型算法的应用。

(2) 大作业和实验

本课程是计算机科学与技术学科的专业基础理论课程，教学难度较大。有的习题也有较大难度和解题复杂度，可以将这类习题中各种算法的实现作为大作业和实验来安排。

5. 考试与成绩评定

为了保证教学质量，建议采取如下考试和评定成绩的方式。

(1) 考题设计

考试题应注重考查学生在学习过程中对基本概念、基本算法、基本技术的掌握，尤其要考核学生在期终总结复习阶段对整个知识系统的全面掌握和灵活运用的情况。

考试题可分为以下 3 种类型。

- 概念型

重点考查学生对基本概念掌握的程度，引导学生重视对基本概念的深入理解，同时还要注意对一些基本术语和符号的理解和掌握，这些都是今后进一步学习和工作必备的工具。只会死背定义者，难以准确回答这类问题。

这种类型考题的基本形式有以下几种：正确性判断、类属判定、改正错误、填空。

- 设计型

重点考查学生灵活运用课堂讲授的基本算法和基本数据结构来解决应用问题的能力以及对问题进行形式化描述和处理的能力，要求学生能深刻理解和综合应用所学的知识、算法和思维方法。这是计算机学科高级人才的基本功之一。

- 证明型

严密的思维和严格的证明是计算机学科高级人才的另一个基本功。证明型考题重点考查学生运用所学的定理、重要引理来证明有关结论的能力。学生在今后的学习中，设计出解决具体问题的算法，并证明算法的正确性是非常有意义的。

(2) 成绩评定

- 考试资格

授课教师要认真足量批改作业，并对每次作业进行认真登记，作为认定学生取得考试资格的依据，作为其参加课程考试、获得课程成绩的必要条件。

- 平时作业成绩

将平时作业成绩按期末成绩总分的 20 分计。按时完成平时作业者获得基础分 12 分，然后按较好、良好、优秀分别加 3、6、8 分。每缺 1 次作业扣除 1 分，未完成作业达 5 次及 5 次以上者，不允许参加期末考试。

- 期末考试

期末考试采用笔试，有条件的高校也可采用机试。

目 录

试读结束：需要全本请在线购买： www.ertongbook.com

第1章

引　论

学习要点

- 理解算法的概念。
- 理解什么是程序，程序与算法的区别和内在联系。
- 能够列举求解问题的基本步骤。
- 掌握算法在最坏情况、最好情况和平均情况下的计算复杂性概念。
- 掌握算法复杂性的渐近性态的数学表述。
- 了解表达算法的抽象机制。
- 熟悉数据类型和数据结构的概念。
- 熟悉抽象数据类型的基本概念。
- 理解数据结构、数据类型和抽象数据类型三者的区别和联系。
- 掌握用 C 语言描述数据结构与算法的方法。

1.1　算法及其复杂性的概念

1.1.1　算法与程序

对于计算机科学来说，算法(Algorithm)的概念是至关重要的。例如，在一个大型软件系统的开发中，设计出有效的算法将起决定性的作用。通俗地讲，算法是指解决问题的一种方法或一个过程。严格地讲，算法是由若干条指令组成的有穷序列，且满足下述几条性质：

- 输入：有零个或多个由外部提供的量作为算法的输入。
- 输出：算法产生至少一个量作为输出。
- 确定性：组成算法的每条指令是清晰的、无歧义的。
- 有限性：算法中每条指令的执行次数是有限的，执行每条指令的时间也是有限的。

程序(Program)与算法不同。程序是算法用某种程序设计语言的具体实现。程序可以不满足算法的有限性。例如操作系统，它是一个在无限循环中执行的程序，因而不是一个算法。然而可把操作系统的各种任务看成是一些单独的问题，每一个问题由操作系统中的一个

子程序通过特定的算法来实现。该子程序得到输出结果后便终止。

1.1.2 算法复杂性的概念

一个算法的复杂性的高低体现在运行该算法所需要的计算机资源的多少上，所需要的资源越多，就说明该算法的复杂性越高；反之，所需要的资源越少，就说明该算法的复杂性越低。**计算机的资源，最重要的是时间和空间(即存储器)资源。**因此，算法的复杂性有时间复杂性和空间复杂性之分。不言而喻，对于任意给定的问题，设计出复杂性尽可能低的算法是设计算法时追求的一个重要目标。另一方面，当给定的问题已有多种算法时，选择其中复杂性最低者，是选用算法时应遵循的一个重要准则。因此，算法复杂性分析对算法的设计或选用有重要的指导意义和实用价值。确切地说，算法的复杂性是运行算法所需要的计算机资源的量。需要的时间资源的量称为时间复杂性；需要的空间资源的量称为空间复杂性。这个量应该集中反映算法的效率，而从运行该算法的实际计算机中抽象出来。换句话说，这个量应该是只依赖于算法要解的问题的规模和算法的输入的函数。如果分别用 n 和 I 表示算法要解的问题的规模和算法的输入，用 C 表示复杂性，那么，算法复杂性表示为 $C(n,I)$。如果把时间复杂性和空间复杂性分开，并分别用 T 和 S 来表示，那么有：$T=T(n,I)$ 和 $S=S(n,I)$。由于时间复杂性与空间复杂性概念类同，计量方法相似，且空间复杂性分析相对简单些，所以本书将主要讨论时间复杂性。现在的问题是如何将复杂性函数具体化，即对于给定的 n 和 I，如何导出 $T(n,I)$ 和 $S(n,I)$ 的数学表达式，给出计算 $T(n,I)$ 和 $S(n,I)$ 的法则。下面以 $T(n,I)$ 为例，将复杂性函数具体化。

根据 $T(n,I)$ 的概念，它应该是算法在一台抽象的计算机上运行所需要的时间。设此抽象的计算机所提供的元运算有 k 种，分别记为 O_1，O_2，…，O_k。又设每执行一次这些元运算所需要的时间分别为 t_1，t_2，…，t_k。对于给定的算法 A，设经统计，用到元运算 O_i 的次数为 $e_i(i=1,2,\cdots,k)$。很清楚，对于每一个 i，$1\leqslant i\leqslant k$，e_i 是 n 和 I 的函数，即 $e_i=e_i(n,I)$。那么有 $T(n,I)=\sum_{i=1}^{k}t_ie_i(n,I)$。其中 $t_i(i=1,2,\cdots,k)$ 是与 n 和 I 无关的常数。

显然，不可能对规模为 n 的每一种合法的输入 I 都统计 $e_i(n,I)(i=1,2,\cdots,k)$。因此 $T(n,I)$ 的表达式还得进一步简化，或者说，只能在规模为 n 的某些或某类有代表性的合法输入中统计相应的 $e_i(i=1,2,\cdots,k)$，并评价时间复杂性。

通常考虑3种情况下的时间复杂性，即最坏情况、最好情况和平均情况下的时间复杂性，并分别记为 $T_{\max}(n)$、$T_{\min}(n)$ 和 $T_{\text{avg}}(n)$。在数学上有：

$$T_{\max}(n)=\max_{I\in D_n}T(n,I)=\max_{I\in D_n}\sum_{i=1}^{k}t_ie_i(n,I)=\sum_{i=1}^{k}t_ie_i(n,I^*)=T(n,I^*)$$

$$T_{\min}(n)=\min_{I\in D_n}T(n,I)=\min_{I\in D_n}\sum_{i=1}^{k}t_ie_i(n,I)=\sum_{i=1}^{k}t_ie_i(n,\tilde{I})=T(n,\tilde{I})$$

$$T_{\text{avg}}(n)=\sum_{I\in D_n}P(I)T(n,I)=\sum_{I\in D_n}P(I)\sum_{i=1}^{k}t_ie_i(n,I)$$

其中，D_n 是规模为 n 的合法输入的集合；I^* 是 D_n 中一个使 $T(n,I^*)$ 达到 $T_{\max}(n)$ 的合法输入；$\tilde{I}$ 是 D_n 中一个使 $T(n,\tilde{I})$ 达到 $T_{\min}(n)$ 的合法输入；而 $P(I)$ 是在算法的应用中出现输入 I 的概率。

以上3种情况下的时间复杂性各从某一个角度来反映算法的效率，各有各的局限性，也各有各的用处。实践表明，可操作性最好且最有实际价值的是最坏情况下的时间复杂性。本书对算法时间复杂性的分析主要采用最坏情况下的时间复杂性分析。

1.1.3 算法复杂性的渐近性态

随着经济的发展、社会的进步和科学研究的深入，要求用计算机解决的问题越来越复杂，规模越来越大。对求解这类问题的算法进行复杂性分析具有特别重要的意义，因而要特别关注。为此，要引入复杂性渐近性态的概念。设 $T(n)$ 是前面所定义的关于算法 A 的复杂性函数。一般来说，当 n 单调增加且趋于 ∞ 时，$T(n)$ 也将单调增加趋于 ∞。对于 $T(n)$，如果存在 $\tilde{T}(n)$，使得当 $n\to\infty$ 时有 $\dfrac{T(n)-\tilde{T}(n)}{T(n)}\to 0$，那么，就说 $\tilde{T}(n)$ 是 $T(n)$ 当 $n\to\infty$ 时的渐近性态，或称 $\tilde{T}(n)$ 为算法 A 当 $n\to\infty$ 时的渐近复杂性而与 $T(n)$ 相区别。因为在数学上，$\tilde{T}(n)$ 是 $T(n)$ 当 $n\to\infty$ 时的渐近表达式。直观上，$\tilde{T}(n)$ 是 $T(n)$ 中略去低阶项所留下的主项。所以它无疑比 $T(n)$ 来得简单。例如，当 $T(n)=3n^2+4n\log n+7$ 时，$\tilde{T}(n)$ 的一个答案是 $3n^2$，因为这时有

$$\lim_{n\to\infty}\frac{T(n)-\tilde{T}(n)}{T(n)}=\lim_{n\to\infty}\frac{4n\log n+7}{3n^2+4n\log n+7}=0$$

显然，$3n^2$ 比 $3n^2+4n\log n+7$ 简单得多。

由于当 $n\to\infty$ 时 $T(n)$ 渐近于 $\tilde{T}(n)$，有理由用 $\tilde{T}(n)$ 来替代 $T(n)$ 作为算法 A 在 $n\to\infty$ 时的复杂性的度量。而且由于 $\tilde{T}(n)$ 明显地比 $T(n)$ 简单，这种替代是对复杂性分析的一种简化。进一步来说，考虑到分析算法的复杂性的目的在于比较求解同一问题的两个不同算法的效率。而当要比较的两个算法的渐近复杂性的阶不相同时，只要能确定出各自的阶，就可以判定哪一个算法的效率高。换句话说，这时的渐近复杂性分析只要关心 $\tilde{T}(n)$ 的阶就够了，不必关心包含在 $\tilde{T}(n)$ 中的常数因子。所以，常常又对 $\tilde{T}(n)$ 的分析进一步简化，即假设算法中用到的所有不同的元运算各执行一次所需要的时间都是一个单位时间。

综上所述，已经给出了简化算法复杂性分析的方法，即只要考察当问题的规模充分大时，算法复杂性在渐近意义下的阶。本书的算法分析都将这么做。与此简化的复杂性分析相配套，需要引入以下渐近复杂性的记号。

以下设 $f(n)$ 和 $g(n)$ 是定义在正数集上的正函数。

如果存在正的常数 c 和自然数 n_0，使得当 $n\geqslant n_0$ 时，有 $f(n)\leqslant cg(n)$，则称函数 $f(n)$ 当 n 充分大时上有界，且 $g(n)$ 是它的一个上界，记为 $f(n)=O(g(n))$。这时还说 $f(n)$ 的阶不高于 $g(n)$ 的阶。

举几个例子：

① 因为对所有的 $n\geqslant 1$ 有 $3n\leqslant 4n$，从而有 $3n=O(n)$；

② 因为当 $n\geqslant 1$ 时有 $n+1024\leqslant 1025n$，从而有 $n+1024=O(n)$；

③ 因为当 $n\geqslant 10$ 时有 $2n^2+11n-10\leqslant 3n^2$，从而有 $2n^2+11n-10=O(n^2)$；

④ 因为对所有 $n\geqslant 1$ 有 $n^2\leqslant n^3$，从而有 $n^2=O(n^3)$；

⑤ 作为一个反例 $n^3\neq O(n^2)$。因为若不然，则存在正的常数 c 和自然数 n_0，使得当 $n\geqslant n_0$ 时，有 $n^3\leqslant cn^2$，即 $n\leqslant c$。显然，当取 $n=\max\{n_0,\lfloor c\rfloor+1\}$ 时，这个不等式不成立，所以 $n^3\neq O(n^2)$。

按照符号 O 的定义，容易证明它有如下运算规则：

① $O(f)+O(g)=O(\max(f,g))$；

② $O(f)+O(g)=O(f+g)$；

③ $O(f)O(g)=O(fg)$；

④ 如果 $g(n)=O(f(n))$，则 $O(f)+O(g)=O(f)$；

⑤ $O(cf(n))=O(f(n))$，其中 c 是一个正的常数；

⑥ $f=O(f)$。

对于规则①的证明如下：设 $F(n)=O(f)$，根据符号 O 的定义，存在正常数 c_1 和自然数 n_1，使得对所有的 $n\geqslant n_1$，有 $F(n)\leqslant c_1 f(n)$。类似地，设 $G(n)=O(g)$，则存在正的常数 c_2 和自然数 n_2，使得对所有的 $n\geqslant n_2$，有 $G(n)\leqslant c_2 g(n)$。

令 $c_3=\max\{c_1, c_2\}$，$n_3=\max\{n_1, n_2\}$，$h(n)=\max\{f, g\}$。则对于所有的 $n\geqslant n_3$，有

$$F(n)\leqslant c_1 f(n)\leqslant c_1 h(n)\leqslant c_3 h(n)$$

类似地，有：

$$G(n)\leqslant c_2 f(n)\leqslant c_2 h(n)\leqslant c_3 h(n)$$

因而：

$$\begin{aligned} O(f)+O(g) &= F(n)+G(n)\leqslant c_3 h(n)+c_3 h(n)\\ &= 2c_3 h(n)=O(h)=O(\max(f,g)) \end{aligned}$$

其余规则的证明类似，可作为读者的练习。

应该指出，根据符号 O 的定义，用它评估算法的复杂性，得到的只是当规模充分大时的一个上界。这个上界的阶越低，则评估就越精确，结果就越有价值。

与渐近复杂性有关的另一记号是 Ω，其定义如下：如果存在正的常数 c 和自然数 n_0，使得当 $n\geqslant n_0$ 时，有 $f(n)\geqslant cg(n)$，则称函数 $f(n)$ 当 n 充分大时下有界，且 $g(n)$ 是它的一个下界，记为 $f(n)=\Omega(g(n))$。这时还说 $f(n)$ 的阶不低于 $g(n)$ 的阶。

用 Ω 评估算法的复杂性，得到的只是该复杂性的一个下界。这个下界的阶越高，则评估就越精确，结果就越有价值。这里的 Ω 只对问题的一个算法而言。如果它是对一个问题的所有算法或某类算法而言，即对于一个问题和任意给定的充分大的规模 n，下界在该问题的所有算法或某类算法的复杂性中取，那么它将更有意义。这时得到的相应下界，称为问题的下界或某类算法的下界。它常常与符号 O 配合，以证明某问题的一个特定算法是该问题的最优算法或该问题的某算法类中的最优算法。

明白了符号 O 和 Ω 之后，符号 Θ 也随之明确。定义 $f(n)=\Theta(g(n))$，当且仅当 $f(n)=O(g(n))$，且 $f(n)=\Omega(g(n))$ 时，$f(n)$ 与 $g(n)$ 同阶。

最后，如果对于任意给定的 $\varepsilon>0$，都存在正整数 n_0，使得当 $n\geqslant n_0$ 时，有 $f(n)/g(n)<\varepsilon$，则称函数 $f(n)$ 当 n 充分大时的阶比 $g(n)$ 低，记为 $f(n)=O(g(n))$。

例如：$4n\log n+7=O(3n^2+4n\log n+7)$。

1.2 算法的表达与数据表示

1.2.1 问题求解

用计算机解决一个稍复杂的实际问题，大体都要经历如下的步骤：

试读结束：需要全本请在线购买： www.ertongbook.com