高等学校教材

计算机算法设计与分析

（第2版）

苏德富　钟　诚　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

高等学校教材

计算机算法设计与分析

第 2 版

苏德富　钟　诚　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内容简介

算法设计与分析是计算机科学与技术的主要研究领域之一。本课程是计算机科学与技术、软件工程、网络工程、信息安全和其他相关专业高年级本科生、研究生的一门重要专业基础课程。

它的主要目标是讲授设计和分析各种算法的基本原理、方法和技术，讲授在计算机应用中经常遇到的诸如排序、查找、选择、串匹配、矩阵运算、大整数相乘、快速傅里叶变换、数据加密、网络路由、生物信息处理、数据库操作等重要的实际问题的解法。

本书的第1版曾获广西高校优秀教材一等奖，第2版列入广西精品教材建设基金项目。全书共15章，取材先进、内容实用、重点突出、少而精、例题丰富、难易适当，便于自学。全书以非数值算法为主，兼顾数值算法；串行算法和并行算法并重；附录中介绍并行MULTIPASCAL系统的使用方法，并给出一个并行程序实例。

本书可供计算机科学与技术、软件工程、网络工程、信息安全、管理信息系统、系统工程、应用数学和计算数学等专业本科生、研究生作为教材使用，也可供从事计算机科学与技术研究、计算机软件开发的工程技术人员参考。

图书在版编目(CIP)数据

计算机算法设计与分析/苏德富，钟诚编著. —2版. —北京：电子工业出版社，2005.7
高等学校教材
ISBN 7-121-01309-6

Ⅰ.计… Ⅱ.①苏…②钟… Ⅲ.①电子计算机—算法设计—高等学校—教材②电子计算机—算法分析—高等学校—教材 Ⅳ.TP301.6

中国版本图书馆CIP数据核字(2005)第051916号

责任编辑：赵家鹏
印　　刷：北京大中印刷厂
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
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：16.5　字数：419.2千字
印　　次：2005年7月第1次印刷
印　　数：5000册　　定价：22.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：(010)68279077。质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

前　言

算法设计与分析是计算机科学与技术的一个主要研究领域。本课程是计算机科学与技术、软件工程、网络工程、信息安全、管理信息系统、系统工程、应用数学和计算数学等专业高年级本科生、研究生的一门重要专业基础课程。它的主要目标是讲授在计算机应用中经常遇到重要的实际问题的解法，讲授设计和分析各种算法的基本原理、方法和技术，以培养读者在选择或设计一个算法时，思考下列问题：这个算法是否有效？这个算法有多好？是否还有更好的算法？用什么方法和技巧去获得更好的算法？从而使得所设计算法的时空复杂度最优，进而为编写高效的程序、开发优秀软件奠定基础。

考虑到与离散数学、程序设计、计算方法、数据结构等前驱课程的联系与衔接，本书兼顾数值和非数值算法，以非数值算法为主；并力争做到取材先进、内容实用、重点突出、少而精、难易适当、便于自学；并注意收录一些典型问题的最新研究成果。期望读者通过本课程的学习，在教师的指导下能较快地进入某个研究领域，接受基础研究和应用基础研究的初步训练，培养独立开展科研工作的能力和创新意识。

目前计算机正朝着微型化、并行化、网络化、智能化和多媒体方向发展。并行分布计算技术正发挥着传统的串行计算技术所不能比拟的越来越重要的作用。因此，除了介绍串行算法之外，本书特别用较大的篇幅介绍并行算法设计技术及其分析方法。

从方便读者理解的目的出发，书中的算法用类 Pascal 语言，或者用类 C 语言，也可以用接近自然语言的方式描述。读者可根据实际情况，使用 Pascal 或者 C 语言对书中的算法适当加以修改即可上机实现；对于并行算法部分，可采用支持多线程并发程序设计的 Java 语言或者采用并行 MULTIPASCAL 系统等编程上机实现。

本书的第 1 版曾获广西高校优秀教材一等奖，第 2 版列入广西精品教材建设基金项目。第 2 版的编写较详尽地讲述了算法设计方法，主要介绍基本思想和基本技巧，这些方法和技巧将应用于后面各章的算法设计中。同时，对于近二十年来算法研究领域发展迅速的随机算法和近似算法给予足够重视，并增加了近年来应用广泛的网络路由算法和新兴交叉学科的生物信息处理算法。

全书共分 15 章。第 1 章介绍算法分析的基本概念和基本理论，详细分析搜索有序表二分查找算法的平均复杂性和最坏情形复杂性；第 2 章介绍设计算法的基本技术和分析算法复杂性的基本方法；第 3 章讨论若干经典数值计算问题，包括大整数相乘、矩阵乘积和快速傅里叶变换算法，以及数据加密算法、数字签名和数据压缩技术；第 4 章主要介绍基于映射(散列)的排序算法和 Quick 排序的随机算法；第 5 章介绍选择算法；第 6 章讲授字符串精确匹配和近似匹配技术，重点剖析一些典型的、能启迪人们思维的算法设计思想；第 7 章讨论应用十分广泛的计算机网络路由算法；第 8 章讲解 NP 完全问题和近似算法；第 9 章的内容则是近年来新兴起的生物信息处理算法；第 10 章介绍并行计算基础知识和并行处理技术的应用；第 11 章从数据求和问题入手，讨论同步和异步并行求和算法

的设计与分析方法；第 12 章阐述并行排序技术，其中展示了许多著名的串行排序算法如何转换成并行算法，这对读者开发新的并行算法会有帮助；第 13 章介绍查找和近似串匹配问题的并行化算法，以拓宽读者的视野；第 14 章讨论一些重要数值计算问题的并行算法；第 15 章重点介绍数据库操作的并行选择算法、并行投影算法、并行集合操作算法和并行连接算法。最后，在附录中简介并行 MULTIPASCAL 系统及其使用方法，同时给出一个基于散列技术的(m,n)选择并行算法和相应的并行程序实例。

本书编者自 20 世纪 80 年代中后期以来，一直从事算法设计与分析、并行分布计算等领域的学习、教学和研究工作。本书是在第 1 版的基础上，参考国内外有关论著编写而成的。此次再版得到中国科学院院士、第一届全国高等学校国家级教学名师奖获得者、国家（合肥）高性能计算中心主任、中国科学技术大学计算机科学与技术系博士生导师陈国良教授的热情关心和指导，得到电子工业出版社和责任编辑赵家鹏老师的大力帮助，我们深表感谢。在编写过程中引用有关参考文献的成果，在此也一并致谢。

国家教育部高等学校计算机专业教学指导委员会和中国计算机学会教育专业委员会制定的《计算机学科教学计划 2003》强调要加强计算机科学与技术、软件工程、网络工程、信息安全专业学生的算法设计与分析能力的培养。我们希望本书的出版有助于推动我国《计算机算法设计与分析》课程教学的普及和发展。

由于编者学识有限，加之编写时间较紧，书中如有不妥之处，敬请专家和读者批评指正，以臻完善。

编著者

2005.3

目　录

第1章 引 论

众所周知，计算机要真正能够充分发挥作用离不开计算机软件。软件由计算机程序、文档和有关的控制数据组成。而计算机程序的核心则是计算机算法。可见，如果没有不断发明新的更有效的算法，就不可能开发出新的更先进的软件。著名的方正排版软件系统每推出新的功能更强大的版本都是在发明新的更有效的算法基础上进行的。

算法是计算学科的核心概念，也被誉为计算学科的灵魂。算法设计的优劣决定着软件系统的性能，对算法进行研究能使人们深刻地理解问题的本质以及可能的求解技术。

有关算法的定义很多，其内涵基本上是一致的，其中最为著名的是计算机科学家克努特在其经典巨著——《计算机程序设计艺术》(The Art of Computer Programming)第1卷中对算法的定义和特性所做的有关描述。

1. 算法的非形式化定义

一个算法，就是一个有穷规则的集合，其中之规则规定了一个解决某一特定类型问题的运算序列。

(1) 有穷性：一个算法在执行有穷步之后必须结束。也就是说，一个算法，它所包含的计算步骤是有限的。

(2) 确定性：算法的每一个步骤必须确切地定义，即算法中所有有待执行的动作必须严格而不含混地进行规定，不能有歧义性。

(3) 输入：算法有零个或多个输入，即在算法开始之前，对算法最初给出的量。

(4) 输出：算法有一个或多个输出，即与输入有某个特定关系的量，简单地说就是执行算法的最终结果。

(5) 能行性：算法中有待执行的运算和操作必须是相当基本的，能够精确地进行运算和操作，算法执行者甚至不需要掌握算法的含义即可根据该算法的每一步骤的要求进行操作，并最终得出正确的结果。

2. 算法的形式化定义

算法是一个四元组，即(Q, I, Ω, F)。

(1) Q是一个包含子集I和Ω的集合，它表示计算的状态；

(2) I表示计算的输入集合；

(3) Ω表示计算的输出集合；

(4) F表示计算的规则，它是一个由Q到它自身的函数，且具有自反性，即对于任何一个元素$q \in Q$，有$F(q)=q$。

通俗地说，一步一步解问题的过程称为算法，研究计算复杂性的一个典型模式是设计和提出解问题的算法所需的基本运算次数和证明其可能达到的界限。

通常所说的算法是在计算机上执行计算过程的抽象描述，与计算机程序是有区别的。程序是在指定的计算机上执行算法，而算法是抽象的，它凌驾于一切具体执行的计算机之

上。我们不打算提出任何高级语言程序，但尽可能用大家熟悉的语法和句法对算法给出形式的描述。

求解同一个问题 P 通常有若干种算法，但是，一定要针对问题，选择求解问题的算法。

(1) 什么是解问题 P 的“好”算法？为回答这个问题，需要比较不同算法的相对有效性。

(2) 什么是解问题 P 所需要的基本运算的最小次数？

(3) 在指定的适当时间内是否有解该问题的算法？

总之，解一个问题，往往有若干种不同的算法。这些算法决定着根据该算法编写的程序性能的好坏。那么，在保证算法正确性的前提下，如何确定算法的优劣就是一个值得研究的课题。

在算法的分析中，一般应考虑以下 3 个问题。

(1) 算法的时间复杂度；

(2) 算法的空间复杂度；

(3) 算法是否便于阅读、修改和测试。

上述问题一直是计算科学研究的核心问题之一，不少计算机科学家为此耗尽了毕生的精力，取得大量极富创造性的成果，并获得了计算机科学领域的最高荣誉——计算图灵奖(Turing Awards)。

一年一度的计算图灵奖是国际计算机界公认的、权威的、影响重大和深远的一项荣誉，被誉为计算机科学领域的“诺贝尔奖”。设立图灵奖的一个主要目的是纪念计算机科学之父——图灵(Turing)。国际性的图灵奖由 ACM(美国计算机学会)主持并于 1966 年开始颁奖。从 1966 年至 2001 年所颁发的图灵奖获得者中，有 14 人由于在算法与数据结构、计算复杂性理论、程序设计以及相关领域做出杰出贡献而荣获图灵奖。他们是：

美国斯坦福大学计算机科学系终身教授 D. E. Knuth 因撰写多卷巨著“The Art of Computer Programming”而于 1974 年成为最年轻的图灵奖获得者。

以色列特拉维夫大学 M. O. Rabin 教授和英国的 D. S. Scott 教授因发表著名的计算复杂性方面的论文“Finite automata and Their Decision problem”荣获 1976 年度图灵奖。

美国的 R. W. Floyd 教授由于在算法分析和程序设计正确性证明领域做出开创性的工作而荣获 1978 年度的图灵奖。

1980 年度的图灵奖则授予在程序设计和算法方面做出突出贡献的、发明著名 QUICKSORT 算法的英国的 C. A. R. Hoare 教授。

研究计算复杂性理论和 NP 完全性问题专家、加拿大的 S. A. Cook 教授的论文“The complexity of Theorem Proving Procedures”也荣获 1982 年度图灵奖。

提出著名公式“算法＋数据结构＝程序”的瑞士的 N. Wirth 教授则成为 1984 年图灵奖得主。

对组合优化算法、网络流有效算法、NP 完全性和随机算法研究很有造诣的美国加州大学伯克利分校著名教授 R. M. Karp 的论文“Reducibility among combinatorial problems”荣获 1985 年度图灵奖。

更有意义的是美国教授 J. E. Hpocrft 和他的学生 R. E. Tarjan 博士密切合作，发明了一种非常重要的数据结构及其算法从而荣获 1986 年度图灵奖。

在计算复杂性领域做出卓越贡献的美国的 J. Hartmanis 和 R. E. Steams 教授荣幸地于 1993 年走上图灵奖领奖台。

两年之后的 1995 年，图灵奖又是颁发给在计算复杂性、密码学算法和程序验证技术取得重要成就的美国教授 M. Blum。

尤其令人感到特别高兴的是，2001 年图灵奖颁发给美籍华人著名计算机科学家 A. C. C. YAO(姚奇智)教授，以表彰他在算法设计与分析、计算复杂性、量子计算领域所做出的突出贡献。

由此可见研究算法对推动计算机科学与技术的研究和发展具有十分重要的作用。

当 Karp 站在图灵奖的领奖台上时，思绪万千，他回顾自己的经历时说："我进入计算机领域甚为偶然。1955 年从哈佛大学毕业并取得数学学位后，我面临的问题是决定下一步干什么。为谋生而工作，对我没有什么吸引力，因此鲜明地选择了研究生院。尽管课程缺乏深度和一致性，但学习空气特别浓厚，我们知道我们正在目睹着一门以计算机为中心的科学诞生。我发现我在算法的结构中找到了美和雅致，……，简言之，多少出于偶然，我摸索着进入了一个我极其喜欢的领域。"(参见 Karp："组合论、复杂性和随机性"，CACM，Vol. 29，NO. 2，1986)

另一位图灵奖获得者 Hopcrft 在谈到美国的现状时说："教育机构和研究室对计算机科学家的需求增长快于这一领域为培养所需合格人材而构造必要的基础结构的速度"。并呼吁美国政府："今天全球都在力争取得技术和经济的领导地位。计算技术在这一斗争中将起关键性的作用。除非我们提出一种全国性的政策来支持计算机科学，否则由于我们无所作为，就会使其他国家在计算机技术方面建立我们不可能赶上的领先地位"(参见：CACM，Vol. 30，NO. 3，1987)。事实上，海湾战争的胜利是高技术的胜利，而高性能计算技术则从中发挥了非常重要的作用，这充分说明了 Hopcrft 的预言是正确的。

那么，面对这一世界性挑战，有幸进入计算机科学领域的青年学者应该怎么办？

1.1 算法分析

在 20 世纪 60 年代，对于算法研究的现状是不能令人满意的。那时，对算法的有效性没有一个令人满意的客观标准，一个研究者可能会在杂志上发表一个算法及对于少数例题的执行时间；而几年后，第二名研究者又会给出一个改进的算法以及对同样例题的执行时间，新的算法肯定更快，因为在相间的几年里，计算机性能和程序设计语言都得到改善。算法在不同的计算机上运行并且编程语言也不同，使得对这种比较的可比性不能令人满意。要弄清楚计算机性能提高与实现编程技术对算法执行时间的影响是很困难的，再者，有可能第二个研究者无意中将其算法搞得对这些例题特别有效。可以预见，如果对于另外的例题再运行这两个算法，第一个算法也可能更快。因此，对算法的分析必须脱离具体的计算机结构和程序设计语言。

比较两个算法的好坏，看其所需的运算时间和存储空间，时间的长短是由算法所需的

运算次数决定的。任何一个算法都可能有几种运算，因此，必须抓住其中影响算法运行时间最大的运算作为基本运算。如在一个字表中寻找 Z 的问题，把 Z 和表中元素的比较作为基本运算。两个实数矩阵相乘的问题中，则把两个实数相乘作为基本运算。

但是同一个问题对不同的输入，基本运算的次数亦可能不同。因此，引进问题大小(即规模，Size)的概念。例如，在一个姓名表中寻找给定的 Z，问题的大小可用表中姓名的数目表示。对于两矩阵相乘问题，其大小可用矩阵的阶表示。而对于遍历一棵二叉树的问题，其大小是用树中的节点数来表示，等等。这样，一个算法的基本运算次数就可能用问题的大小 n 的函数 $f(n)$来表示。$f(n)$是 $N\to N$ 的一个函数(N 为自然数集合)。

对于算法的优劣，通常以平均性态和最坏情形两种结果来衡量。

1. 算法的平均复杂性(Average behavior)

设 D_n 是对于所考虑问题来说大小为 n 的输入的集合，并设 I 是 D_n 的一个元素，$p(I)$是 I 出现的概率，$t(I)$是算法在输入 I 时所执行的基本运算次数。那么，算法的平均复杂性定义为：

$$A(n)=\sum_{I\in D_n}p(I)t(I)$$

2. 最坏情形复杂性(Worst-case complexity)

$$W(n)=\max_{I\in D_n}t(I)$$

例 1.1 考虑顺序查找算法：设 L 是一个有 n 个数据的线性表 L(数组)。对于某个指定的 z，如果 z 在表 L 中，则查找出它在表 L 的下标；如果 z 不在表 L 中，则返回结果零。

算法 1.1.1 Sequential Search(L, n, z)

```
Begin
  j=1;
  while j<=n and L[j]≠z do
    j=j+1;
  if j>=n then
    j=0;
  writeln ('j=', j);
End
```

上述顺序查找算法的基本运算为将 z 和表 L 中数据项做比较操作。现在分析算法的平均复杂性。

设 I_i 表示 z 出现在表中第 i 个位置的情况。$t(I_i)$表示输入 I_i 的比较次数。所以

$$t(I_i)=i,\ 1\leqslant i\leqslant n,\quad t(I_{n+1})=n+1$$

设 q 表示 z 出现在表 L 中的概率，并假定它出现在表 L 中任何位置的可能性(概率)是一样的。那么，$p(I_i)=q/n$ 且 $p(I_{n+1})=1-q$，从而

$$A(n)=\sum_{i=1}^{n+1}p(I_i)t(I_i)=\sum_{i=1}^{n}\left(\frac{q}{n}\times i\right)+(1-q)(n+1)$$

$$= \frac{q}{n}\sum_{i=1}^{n} i + (1-q)(n+1)$$

$$= \frac{q}{n} \times n(n+1)/2 + (1-q)(n+1)$$

$$= q(n+1)/2 + (1-q)(n+1)$$

如果 z 肯定出现在表 L 中，即 $q=1$；则

$$A(n)=(n+1)/2$$

这说明平均要查找半个线性表。

如果 z 只有一半的可能出现在线性表 L 中，即 $q=1/2$，则

$$A(n)=(n+1)/4+(n+1)/2 \approx 3/4 \times n$$

这说明平均需要查找线性表中 3/4 的元素。

而算法在最坏情况下，所需的时间是：

$$W(n)=\max\{t(I_i): 1 \leqslant i \leqslant n+1\}=n+1$$

即 z 不出现在表 L 中或出现在表 L 的最后一个位置时，都要查找整个线性表 L。

例 1.2 已知矩阵 $A=(a_{ij})_{n\times n}$，$B=(b_{ij})_{n\times n}$，求乘积 $C=A\times B$。

计算乘积的方法是：$C_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{kj}$，$1 \leqslant i, j \leqslant n$

算法 1.1.2 Matrix-Multiplication (A, B, C, n)

```
Begin
  for i=1 to n do
    for j=1 to n do
      begin
        C[i, j]=0;
        for k=1 to n do
            C[i, j]=C[i, j]+a[i, k] * b[k, j];
      end;
End
```

矩阵相乘算法的基本操作是矩阵元素相乘操作。

算法复杂性分析：

对于矩阵 C 中的每一个元素都需要做 n 次乘法操作，而 C 共有 n^2 项，因此

$$A(n) = W(n) = n^3$$

但是，仅仅通过算法平均和最坏情形的分析还不能完全说明一个算法的有效性。我们需要一种渐近表示法。

1.2 算法的渐近性态分析

假设正在考虑两种算法 A 与 B 来求解某一给定问题。而且假定已经对每一个算法运行时间都做过详尽的分析，并将其分别设为 $T_A(n)$ 与 $T_B(n)$，其中，n 表示所求问题的规模(Size)。然后比较函数 $T_A(n)$ 与 $T_B(n)$ 的大小，以确定哪一种算法最优。

例如，对于同一个问题 P，假设一个算法在最坏情形下的时间复杂性为 $W1(n)=3n^2$，而另一个算法的最坏情形时间复杂性为 $W2(n)=25n$。当 $n=8$ 时，$W1(8)=192$，$W2(8)=200$；显然，$W1(8)<W2(8)$，即 $W1(n)$ 比 $W2(n)$ 的运算次数少。但是，当 $n=9$ 时，$W1(9)=243$，$W2(9)=225$ 并且当 $n\geqslant 9$ 时，都有 $W1(n)>W2(n)$ 成立，即第一个算法比第二个算法速度慢。

可以证明只要 $\forall n\geqslant 0$，且 $T_A(n)\leqslant T_B(n)$，那么算法 A 就要优于算法 B，且与问题的规模无关。

通常预先既不知道问题的规模，也不知道在整个问题规模区间上一个函数是否小于或等于另一个函数。此时，我们要在大规模区间上考虑两个函数的渐近性态(asymptotic behavior)。

因此，需要考虑当 n 足够大时对算法进行比较，即讨论算法的渐近性态。

1.2.1 渐近表示法

1892 年，P. Bachmann 发明了一种表示函数渐近特征的方法，称之为大 O 表示法。

定义 1.2.1 (大 O 表示法)设对一切 $n\geqslant 0$ 的整数有一个非负函数 $f(n)$。如果存在一个整数 n_0 和一个正常数 c，且对任何 $n\geqslant n_0$ 都有 $f(n)\leqslant cg(n)$，则称“$f(n)$ 是 $g(n)$ 的大 O 表示”，记为 $f(n)=O(g(n))$。

此时，称 $f(n)$ 的阶低于或等于 $g(n)$ 的阶，记为“$f(n)$ 是 $O(g(n))$”或“$f(n)\in O(g(n))$”或“$f(n)=O(g(n))$”。读做 $f(n)$ 是 $g(n)$ 的大 O。这里，大“O”是 Order 一字的字头，是“阶”或“数量级”的意思。因此，属于同一数量级的算法很多，是一个集合。

例 1.3 设有函数 $f(n)=8n+128$。很显然，对于所有 $n\geqslant 0$ 的整数，$f(n)$ 非负。现在若把 $f(n)$ 表示为 $f(n)=O(n^2)$，根据定义 1.2.1，就需要找到一个整数 n_0 和一个正常数 c，使得 $n\geqslant n_0$ 时，$f(n)\leqslant cn^2$。

特殊的常数 c 只要存在，它的大小究竟是多少无关紧要。例如，假定 $c=1$，那么

$$\begin{aligned} f(n)\leqslant cn^2 &=> 8n+128\leqslant n^2 \\ &=> 0\leqslant n^2-8n-128 \\ &=> (n-16)(n+8) \end{aligned}$$

由于对任意 $n\geqslant 0$，有 $(n+8)>0$，那么推得 $(n-16)\geqslant 0$，即 $n_0=16$。

所以，存在 $c=1$ 和 $n_0=16$，对一切 $n\geqslant n_0$ 的整数有 $f(n)\leqslant cn^2$，故 $f(n)=O(n^2)$。当 $n=16$ 时，对右边所有的整数而言，函数 $f(n)=n^2$ 都大于函数 $f(n)=8n+128$。

当然，c 和 n_0 还存在许多其他的值，使得 $f(n)\leqslant cg(n)$ 成立。例如 $c=2$，$n_0=\lceil 2+4\sqrt{17}\rceil=11$ 时成立；同样 $c=4$，$n_0=\lceil 1+\sqrt{33}\rceil=7$ 时上式也成立。

例 1.4 $f(n)=3^n$ 不是 $O(2^n)$。因为假设存在常数 c 与 n_0，使得当 $n\geqslant n_0$ 时，有 $3^n\leqslant c2^n$，那么 $c\geqslant(3/2)^n$ 对于一切 $n>n_0$ 成立。但是，我们知道随着 n 的增大，$(3/2)^n$ 可以无限地增大，因此只要 n 充分大，就可以使 $(3/2)^n$ 大于任意预先指定的常数 c。这就说明所假设的常数 c 是不存在的。

1.2.2 大 O 表示法中的误区

描述的大 O 法可能会误导初学者。本节介绍由于对大 O 表示法的曲解而引起的两