普通高等教育“十一五”国家级规划教材

高等学校规划教材

计算机算法设计与分析（第3版）

王晓东 编著

计 算 机 学 科 教 学 计 划

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

策划编辑：童占梅　责任编辑：童占梅　封面美编：闫欢玲

- 计算机导论（第2版）（王玉龙 编著）
- 电路与电子学（第3版）（王文辉 等编著）
- 电路与电子学习题解答与实验指导（李景宏 等编著）
- 数字逻辑与数字系统（第3版）（王永军 等编著）
- 数字逻辑与数字系统习题解答与实验指导（赵丽红 等编著）
- 计算机组成原理与汇编语言程序设计（第2版）（徐 洁 俸远祯 编著）
- 计算机系统结构（第3版）（徐炜民 等编著）
- 操作系统（第2版）（罗 宇 等编著）
- 计算机算法设计与分析（第3版）（王晓东 编著）
- 算法设计与实验题解（王晓东 编著）
- 程序设计语言与编译（第2版）（龚天富 编）
- 软件工程（第2版）（杨文龙 古天龙 编著）
- 数据库系统原理（第2版）（李建中 王 珊 编著）
- 数据结构（C语言版）（王晓东 编著）
- 数据通信与计算机网络（第3版）（杨心强 等编）
- 计算机图形学基础（第2版）（陆 枫 等编著）
- 人工智能原理及其应用（第2版）（王万森 编著）
- Petri网原理与应用（袁崇义 著）
- 信息安全技术导论（陈克非 黄 征 编著）
- 计算机组成原理（罗克露 等编著）
- 计算机组成原理学习指导与习题解答（罗克露 等编）
- Linux教程（第2版）（孟庆昌　牛欣源 编著）

近期推出的部分
高等学校计算机专业规划教材

本套教材在国家规划教材的基础上，按照“计算机学科教学计划”进行全面更新，以适应高校计算机专业课程与教学改革的需要，并特别注意教材的可读性和可用性，为任课教师提供各种教学服务（包括教学电子课件、教学指导材料、习题解答和实验指导等）。

请关注前言，或随时登录电子工业出版社华信教育资源网站 http://www.hxedu.com.cn 或 http://www.huaxin.edu.cn 了解每本书或系列教材的详细信息。

ISBN 978-7-121-04278-2
9 787121 042782 >
定价：29.50 元

普通高等教育“十一五”国家级规划教材

高等学校规划教材

计算机算法设计与分析

（第3版）

王晓东　编著

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

内 容 简 介

本书为普通高等教育“十一五”国家级规划教材，是计算机专业核心课程“算法设计与分析”教材。全书以算法设计策略为知识单元，系统介绍计算机算法的设计方法与分析技巧。主要内容包括：算法概述、递归与分治策略、动态规划、贪心算法、回溯法、分支限界法、随机化算法、线性规划与网络流、NP 完全性理论与近似算法等。书中既涉及经典与实用算法及实例分析，又包括算法热点领域追踪。

为突出教材的可读性和可用性，章首增加了学习要点提示；章末配有难易适度的习题，分为算法分析题和算法实现题两部分；配套出版了《算法设计与实验题解》；并免费提供电子课件和教学网站服务。

本书适合作为大学计算机科学与技术、软件工程、信息安全、信息与计算科学等专业本科生和研究生教材，也适合广大工程技术人员学习参考。

图书在版编目(CIP)数据

计算机算法设计与分析/王晓东编著．—3 版．—北京：电子工业出版社，2007.5
高等学校规划教材
ISBN 978-7-121-04278-2

Ⅰ．计… Ⅱ．王… 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(2007)第 059536 号

策划编辑：童占梅
责任编辑：童占梅
印　　刷：北京季蜂印刷有限公司
装　　订：三河市鹏成印业有限公司
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：787×1 092　1/16　印张：24.5 字数：622 千字
印　　次：2007 年 5 月第 1 次印刷
印　　数：6000 册　　定价：29.50 元

凡购买电子工业出版社的图书有缺损问题，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。

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。

服务热线：(010)88258888。

前　言

计算机的普及极大地改变了人们的生活。目前，各行业、各领域都广泛采用了计算机信息技术，并由此产生出开发各种应用软件的需求。为了以最小的成本、最快的速度、最好的质量开发出适合各种应用需求的软件，必须遵循软件工程的原则。设计一个高效的程序不仅需要编程小技巧，更需要合理的数据组织和清晰高效的算法，这正是计算机科学领域数据结构与算法设计所研究的主要内容。

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。“计算机算法设计与分析”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。通过对计算机算法系统的学习与研究，理解掌握算法设计的主要方法，培养对算法的计算复杂性正确分析的能力，为独立设计算法和对算法进行复杂性分析奠定坚实的理论基础，对每一位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都是非常重要和必不可少的。

为了适应21世纪我国培养计算机各类人才的需要，本课程结合我国高等学校教育工作的现状，追踪国际计算机科学技术的发展水平，更新了教学内容和教学方法，以算法设计策略为知识单元，系统地介绍计算机算法的设计方法与分析技巧，以期为计算机专业的学生提供一个广泛扎实的计算机算法知识基础。本课程的教学改革实践取得了丰硕的成果，**该课程已被评为省精品课程**。

本书修正了第2版中已发现的一些错误，并将各章的**习题分为算法分析题和算法实现题两部分**，增加了算法实践性内容，以期加强教学实践环节。

全书共分9章。

第1章介绍算法的基本概念，并对算法的计算复杂性和算法的描述做了简要阐述。然后围绕算法设计常用的基本设计策略组织了第2～9章的内容。

第2章介绍递归与分治策略，它是设计有效算法最常用的策略，也是必须掌握的方法。

第3章是动态规划算法，以具体实例详述动态规划算法的设计思想、适用性及算法的设计要点。

第4章介绍贪心算法，它也是一种重要的算法设计策略，它与动态规划算法的设计思想有一定的联系，但其效率更高。按贪心算法设计出的许多算法能导致最优解。其中有许多典型问题和典型算法可供学习和使用。

第5章和第6章分别介绍回溯法和分支限界法。这两章所介绍的算法适合处理难解问题。其解题思想各具特色，值得学习和掌握。

第7章介绍随机化算法，对许多难解问题提供了高效的解决途径，是有很高实用价值的算法设计策略。

第8章介绍实用性很强的线性规划与网络流算法。许多实际应用问题可以转化为线性规划和网络流问题，并可用第8章中的算法有效求解。

第9章首先介绍计算模型、确定性和非确定性图灵机，然后进一步深入介绍NP完全性理论和解NP难问题的近似算法，这是当前计算机算法领域的热点研究课题，具有很高的实用价值。

在本书各章的论述中，首先介绍一种算法设计策略的基本思想，然后从解决计算机科学和

应用中的实际问题入手，由简到繁地描述几个经典的精巧算法。同时对每个算法所需的时间和空间进行分析，使读者既能学到一些常用的精巧算法，又能通过对算法设计策略的反复应用，牢固掌握这些算法设计的基本策略，以期收到融会贯通之效。在为各种算法设计策略选择用于展示其设计思想与技巧的具体应用问题时，本书有意重复选择某些经典问题，使读者能深刻地体会到一个问题可以用多种设计策略求解。同时通过对解同一问题的不同算法的比较，使读者更容易体会到每一种具体算法的设计要点。随着本书内容的逐步展开，读者也将进一步感受到综合应用多种设计策略可以更有效地解决问题。

本书采用**面向对象的C++语言作为算法描述手段**，在保持C++优点的同时，尽量使算法描述简明、清晰。

为便于学习，我们在章首增加了学习要点提示，在章末配有难易适度的习题，并分为算法分析题和算法实现题两部分，以强化实践环节。为便于教学，本教材**配套出版了《算法设计与实验题解》**，并**免费提供电子课件**，请任课教师登录电子工业出版社华信教育资源网 http://www.huaxin.edu.cn 下载。

作者还结合精品课程建设，建立了**教学网站 http://algorithim.fzu.edu.cn**，欢迎广大读者访问教学网站，并提出宝贵意见，作者 E-mail：wangxd@fzu.edu.cn。

在本书编写过程中，得到了全国高等学校计算机专业教学指导委员会的关心和支持。福州大学“211工程”计算机与信息工程重点学科实验室为本书的写作提供了优良的设备和工作环境。田俊教授认真审阅了全书，对本书的内容及各章节的编排提出了许多建设性意见。电子工业出版社负责本书编辑出版工作的全体同仁为本书的出版付出了大量辛勤的劳动，他们认真细致、一丝不苟的工作精神保证了本书的出版质量。在此，谨向每一位曾经关心和支持本书编写工作的各方面人士表示衷心的谢意！

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，使本书在使用中不断得到改进，日臻完善。

作　者

目　录

第1章 算法概述

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
- 掌握用C++语言描述算法的方法

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。更严格地讲，算法是由若干条指令组成的有穷序列，且满足下述4条性质。

(1) 输入：有零个或多个由外部提供的量作为算法的输入。

(2) 输出：算法产生至少一个量作为输出。

(3) 确定性：组成算法的每条指令是清晰的，无歧义的。

(4) 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性质(4)。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。

描述算法可以有多种方式，如自然语言方式、表格方式等。在本书中，采用C++语言描述算法。C++语言的优点是类型丰富、语句精炼，具有面向过程和面向对象的双重特点。用C++来描述算法可使整个算法结构紧凑，可读性强。在本书中，有时为了更好地阐明算法的思路，还采用C++与自然语言相结合的方式来描述算法。

1.2 算法复杂性分析

算法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，所需资源越多，该算法的复杂性越高；反之，所需资源越少，该算法的复杂性越低。对计算机资源，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，是选用算法时遵循的一个重要准则。因此，算法的复杂性分析对算法的设计或选用有着重要的指导意义和实用

价值。

更确切地说，算法的复杂性是算法运行所需要的计算机资源的量，需要时间资源的量称为时间复杂性，需要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，并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要解的问题的规模、算法的输入和算法本身的函数。如果分别用 N，I 和 A 表示算法要解的问题的规模、算法的输入和算法本身，而且用 C 表示复杂性，那么，应该有 $C = F(N,I,A)$，其中 $F(N,I,A)$ 是一个由 N，I 和 A 确定的三元函数。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，应该有 $T = T(N,I,A)$ 和 $S = S(N,I,A)$。通常，A 隐含在复杂性函数名当中，因而将 T 和 S 分别简写为 $T = T(N,I)$ 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，I 和 A，如何导出 $T(N,I)$ 和 $S(N,I)$ 的数学表达式，来给出计算 $T(N,I)$ 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，它们分别记为 $O_1,O_2,\cdots,O_k$。又设每执行一次这些元运算所需要时间分别为 $t_1,t_2,\cdots,t_k$。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 $e_i = e_i(N,I)$。因此有

$$T(N,I) = \sum_{i=1}^{k} t_i e_i(N,I)$$

式中，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 都去统计 $e_i(N,I)$，$i = 1,2,\cdots,k$。因此 $T(N,I)$ 的表达式还要进一步简化，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，$i = 1,2,\cdots,k$，评价其时间复杂性。

本书只考虑三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

$$T_{\max}(N) = \max_{I \in D_N} T(N,I) = \max_{I \in D_N} \sum_{i=1}^{k} t_i e_i(N,I) = \sum_{i=1}^{k} t_i e_i(N,I^*) = T(N,I^*)$$

$$T_{\min}(N) = \min_{I \in D_N} T(N,I) = \min_{I \in D_N} \sum_{i=1}^{k} t_i e_i(N,I) = \sum_{i=1}^{k} t_i e_i(N,\widetilde{I}) = T(N,\widetilde{I})$$

$$T_{\text{avg}}(N) = \sum_{I \in D_N} P(I) T(N,I) = \sum_{I \in D_N} P(I) \sum_{i=1}^{k} t_i e_i(N,I)$$

式中，D_N 是规模为 N 的合法输入的集合；I^* 是 D_N 中使 $T(N,I^*)$ 达到 $T_{\max}(N)$ 的合法输入；$\widetilde{I}$ 是 D_N 中使 $T(N,\widetilde{I})$ 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。

以上三种情况下的时间复杂性从某一个角度反映算法的效率，各有其局限性，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。

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。在此，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，$T(N)$ 也将单调增加且趋于 ∞。对于 $T(N)$，如果存在 $\widetilde{T}(N)$，使得当 $N\to\infty$ 时有 $(T(N)-\widetilde{T}(N))/T(N)\to 0$，那么，就说 $\widetilde{T}(N)$ 是 $T(N)$ 当 $N\to\infty$ 时的渐近性态，或称 $\widetilde{T}(N)$ 为算法 A 当 $N\to\infty$ 的渐近复杂性，而与 $T(N)$ 相区别。因为在数学上，$\widetilde{T}(N)$ 是 $T(N)$ 当 $N\to\infty$ 时的渐近表达式。直观上，$\widetilde{T}(N)$ 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。比如当 $T(N)=3N^2+4N\log N+7$ 时，$\widetilde{T}(N)$ 的一个答案是 $3N^2$，因为这时有

$$(T(N)-\widetilde{T}(N))/T(N)=\frac{4N\log N+7}{3N^2+4N\log N+7}\to 0(\text{当 } N\to\infty \text{ 时})$$

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 $\widetilde{T}(N)$，我们有理由用 $\widetilde{T}(N)$ 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 $\widetilde{T}(N)$ 明显地比 $T(N)$ 简单，这种替代明显地是对复杂性分析的一种简化。进一步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 $\widetilde{T}(N)$ 的阶就够了，不必关心包含在 $\widetilde{T}(N)$ 中的常数因子。所以，常常又对 $\widetilde{T}(N)$ 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。

上面给出了简化算法复杂性分析的方法和步骤，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。与此简化的复杂性分析相配套，需要引入以下渐近意义下的记号 O,Ω,θ 和 o。

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 $3N\leqslant 4N$，所以有 $3N=O(N)$；

(2) 因为当 $N\geqslant 1$ 时有 $N+1024\leqslant 1025N$，所以有 $N+1024=O(N)$；

(3) 因为当 $N\geqslant 10$ 时有 $2N^2+11N-10\leqslant 3N^2$，所以有 $2N^2+11N-10=O(N^2)$；

(4) 因为对所有 $N\geqslant 1$ 有 $N^2\leqslant N^3$，所以有 $N^2=O(N^3)$；

(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 $N\geqslant N_0$ 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) $f = O(f)$。

规则(1)的**证明**：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，留给读者作为练习。

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。

关于符号 Ω，文献里有两种不同的定义。本书只采用其中的一种，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。Ω 的这个定义的优点是与 O 的定义对称，缺点是当 $f(N)$ 对自然数的不同无穷子集有不同的表达式，且有不同的阶时，不能很好地刻画出 $f(N)$ 的下界。比如当

$$f(N) = \begin{cases} 100 & N \text{ 为正偶数} \\ 6N^2 & N \text{ 为正奇数} \end{cases}$$

时，按上述定义，只能得到 $f(N) = \Omega(1)$，这是一个平凡的下界，对算法分析没有什么价值。然而，考虑到上述定义有与符号 O 定义的对称性，又考虑到本书介绍的算法都没出现上例中那种情况，所以本书还是选用它。

同样要指出，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。再则，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的，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。

定义 $f(N) = \theta(g,(N))$ 当且仅当 $f(N) = O(g(N))$ 且 $f(N) = \Omega(g(N))$。这时，我们说

$f(N)$ 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：$4N\log N+7=o(3N^2+4N\log N+7)$。

习题 1

1. 算法分析题

算法分析题1-1 求下列函数的渐近表达式：

$3n^2+10n$；$n^2/10+2^n$；$21+1/n$；$\log n^3$；$10\log 3^n$

算法分析题1-2 试论 $O(1)$ 和 $O(2)$ 的区别。

算法分析题1-3 按照渐近阶从低到高的顺序排列以下表达式：

$$4n^2,\log n,3^n,20n,2,n^{2/3}$$

又 $n!$ 应该排在哪一位？

算法分析题1-4 (1) 假设某算法在输入规模为 n 时的计算时间为 $T(n)=3\times 2^n$。在某台计算机上实现并完成该算法的时间为 t 秒。现有另一台计算机，其运行速度为第一台的 64 倍，那么在这台新机器上用同一算法在 t 秒内能解输入规模为多大的问题？

(2) 若上述算法的计算时间改进为 $T(n)=n^2$，其余条件不变，则在新机器上用 t 秒时间能解输入规模为多大的问题？

(3) 若上述算法的计算时间进一步改进为 $T(n)=8$，其余条件不变，那么在新机器上用 t 秒时间能解输入规模为多大的问题？

算法分析题1-5 硬件厂商 XYZ 公司宣称他们最新研制的微处理器运行速度为其竞争对手 ABC 公司同类产品的 100 倍。对于计算复杂性分别为 n,n^2,n^3 和 $n!$ 的各算法，若用 ABC 公司的计算机在 1 小时内能解输入规模为 n 的问题，那么用 XYZ 公司的计算机在 1 小时内分别能解输入规模为多大的问题？

算法分析题1-6 对于下列各组函数 $f(n)$ 和 $g(n)$，确定 $f(n)=O(g(n))$ 或 $f(n)=\Omega(g(n))$ 或 $f(n)=\theta(g(n))$，并简述理由。

(1) $f(n)=\log n^2$；$g(n)=\log n+5$

(2) $f(n)=\log n^2$；$g(n)=\sqrt{n}$

(3) $f(n)=n$；$g(n)=\log^2 n$

(4) $f(n)=n\log n+n$；$g(n)=\log n$

(5) $f(n)=10$；$g(n)=\log 10$

(6) $f(n)=\log^2 n$；$g(n)=\log n$

(7) $f(n)=2^n$；$g(n)=100n^2$

(8) $f(n)=2^n$；$g(n)=3^n$

算法分析题1-7 证明 $n!=o(n^n)$。

算法分析题1-8 下面的算法段用于确定 n 的初始值。试分析该算法段所需计算时间的上界和下界。

```
while(n > 1)
  if(odd(n))
    n = 3 * n + 1;
  else
    n = n/2;
```

算法分析题1-9　证明：如果一个算法在平均情况下的计算时间复杂性为 $\theta(f(n))$，则该算法在最坏情况下所需的计算时间为 $\Omega(f(n))$。

2. 算法实现题

算法实现题 1-1　统计数字问题

★ 问题描述：一本书的页码从自然数 1 开始顺序编码直到自然数 n。书的页码按照通常的习惯编排，每个页码都不含多余的前导数字 0。例如，第 6 页用数字 6 表示，而不是 06 或 006 等。数字计数问题要求对给定书的总页码 n，计算出书的全部页码中分别用到多少次数字 0，1，2，…，9。

★ 算法设计：给定表示书的总页码的十进制整数 n（$1 \leqslant n \leqslant 10^9$），计算书的全部页码中分别用到多少次数字 0，1，2，…，9。

★ 数据输入：输入数据由文件名为 input. txt 的文本文件提供。每个文件只有 1 行，给出表示书的总页码的整数 n。

★ 结果输出：将计算结果输出到文件 output. txt。输出文件共有 10 行，在第 k 行输出页码中用到数字 k-1 的次数，$k = 1,2,\cdots,10$。

输入文件示例	输出文件示例
input. txt	output. txt
11	1
	4
	1
	1
	1
	1
	1
	1
	1
	1

算法实现题 1-2　字典序问题

★ 问题描述：在数据加密和数据压缩中常需要对特殊的字符串进行编码。给定的字母表 A 由 26 个小写英文字母组成，即 $A = \{a,b,\cdots,z\}$。该字母表产生的升序字符串是指字符串中字母从左到右出现的次序与字母在字母表中出现的次序相同，且每个字符最多出现 1 次。例如，a，b，ab，bc，xyz 等字符串都是升序字符串。现在对字母表 A 产生的所有长度不超过 6 的升序字符串按照字典序排列并编码如下。

1	2	…	26	27	28	…
a	b	…	z	ab	ac	…

对于任意长度不超过 6 的升序字符串，迅速计算出它在上述字典中的编码。

★ 算法设计：对于给定的长度不超过 6 的升序字符串，计算它在上述字典中的编码。

★ 数据输入：输入数据由文件名为 input. txt 的文本文件提供。文件的第 1 行是一个正整数 k，表示接下来共有 k 行。在接下来的 k 行中，每行给出一个字符串。

★ 结果输出：将计算结果输出到文件 output. txt。文件共有 k 行，每行对应于一个字符串的编码。

输入文件示例	输出文件示例
input. txt	output. txt
2	1
a	2
b	

算法实现题 1-3　最多约数问题

★ 问题描述：正整数 x 的约数是能整除 x 的正整数。正整数 x 的约数个数记为 div(x)。例如，1，2，5，10 都是正整数 10 的约数，且 div(10) = 4。设 a 和 b 是 2 个正整数，$a \leqslant b$，找出 a 和 b 之间约数个数最多的数 x。

★ 算法设计：对于给定的 2 个正整数 $a \leqslant b$，计算 a 和 b 之间约数个数最多的数。

★ 数据输入：输入数据由文件名为 input. txt 的文本文件提供。文件的第 1 行有 2 个正整数 a 和 b。

★ 结果输出：若找到的 a 和 b 之间约数个数最多的数是 x，则将 div(x) 输出到文件 output. txt。

输入文件示例	输出文件示例
input. txt	output. txt
1 36	9

算法实现题 1-4　金币阵列问题

★ 问题描述：有 $m \times n(m \leqslant 100, n \leqslant 100)$ 枚金币在桌面上排成一个 m 行 n 列的金币阵列。每一枚金币或正面朝上，或背面朝上。用数字表示金币状态，0 表示金币正面朝上，1 表示金币背面朝上。

金币阵列游戏的规则是：

(1) 每次可将任一行金币翻过来放在原来的位置上；

(2) 每次可任选 2 列，交换这 2 列金币的位置。

★ 算法设计：给定金币阵列的初始状态和目标状态，计算按金币游戏规则，将金币阵列从初始状态变换到目标状态所需的最少变换次数。

★ 数据输入：由文件 input. txt 给出输入数据。文件中有多组数据。文件的第 1 行有 1 个正整数 k，表示有 k 组数据。每组数据的第 1 行有 2 个正整数 m 和 n。以下的 m 行是金币阵列的初始状态，每行有 n 个数字表示该行金币的状态，0 表示正面朝上，1 表示背面朝上。接着的 m 行是金币阵列的目标状态。

★ 结果输出：将计算出的最少变换次数按照输入数据的次序输出到文件 output. txt。相应数据无解时输出 −1。