普通高等教育“十一五”国家级规划教材

21世纪大学本科
计算机专业系列教材

算法设计与分析习题解答与学习指导（第2版）

屈婉玲 刘 田
张立昂 王捍贫 编著

- 根据教育部“高等学校计算机科学与技术专业规范”组织编写
- 与美国 ACM 和 IEEE CS *Computing Curricula* 最新进展同步

清华大学出版社

普通高等教育“十一五”国家级规划教材

21世纪大学本科计算机专业系列教材

算法设计与分析习题解答与学习指导

（第2版）

屈婉玲 刘田 张立昂 王捍贫 编著

清华大学出版社

北京

内 容 简 介

本教材为普通高等教育“十一五”国家级规划教材《算法设计与分析(第 2 版)》(主教材)的辅助教材. 主教材的主要内容包括基础知识、分治策略、动态规划、贪心法、回溯与分支限界、线性规划、网络流算法、算法分析与问题的计算复杂度、NP 完全性、近似算法、随机算法、处理难解问题的策略等. 本书对主教材所阐述的算法设计技术和分析方法进行了总结,并对其中 200 多道习题给出了详尽的解答和分析.

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的辅助教学用书,也可以作为从事实际问题求解的算法设计与分析工作的参考书.

图书在版编目(CIP)数据

算法设计与分析习题解答与学习指导/屈婉玲等编著. --2 版. --北京:清华大学出版社,2016
21 世纪大学本科计算机专业系列教材
ISBN 978-7-302-42955-5

Ⅰ. ①算… Ⅱ. ①屈… Ⅲ. ①电子计算机－算法设计－高等学校－教学参考资料 ②电子计算机－算法分析－高等学校－教学参考资料 Ⅳ. ①TP301.6

中国版本图书馆 CIP 数据核字(2016)第 024865 号

责任编辑: 张瑞庆
封面设计: 何凤霞
责任校对: 焦丽丽
责任印制: 刘海龙

出版发行: 清华大学出版社
网　　址: http://www.tup.com.cn, http://www.wqbook.com
地　　址: 北京清华大学学研大厦 A 座　　**邮　　编:** 100084
社 总 机: 010-62770175　　**邮　　购:** 010-62786544
投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn
课件下载: http://www.tup.com.cn,010-62795954
印 刷 者: 北京富博印刷有限公司
装 订 者: 北京市密云县京文制本装订厂
经　　销: 全国新华书店
开　　本: 185mm×260mm　　**印　　张:** 12　　**字　　数:** 286 千字
版　　次: 2014 年 8 月第 1 版　2016 年 3 月第 2 版　　**印　　次:** 2016 年 3 月第1次印刷
印　　数: 3501～5500
定　　价: 29.00 元

产品编号: 068216-01

21世纪大学本科计算机专业系列教材编委会

第2版前言

FOREWORD

本书是《算法设计与分析(第2版)》的辅助教材.作为普通高等教育"十一五"国家级规划教材,《算法设计与分析(第1版)》出版已经近5年了.在这5年的时间里,大数据、云计算、互联网+等新领域、新问题、新应用层出不穷,许多问题求解都离不开问题的建模和算法的设计与分析.这次关于主教材的修订保持了原书的基本结构、主要内容与写作特色.考虑到线性规划与网络流问题在实践中的广泛应用,增加了这两章内容,并在第9章增添了整数线性规划的NP完全性证明.此外,补充了部分习题,并对第1版书中的某些疏漏之处进行了更新.

与主教材配套,本书也进行了同步更新,增加了第6章线性规划、第7章网络流算法,并对补充习题给出了解答.

本书第1～4章由屈婉玲编写,第5、8章由王捍贫编写,第6～7、9～10章由张立昂编写,第11～12章由刘田编写.

欢迎广大读者批评指正!

作　者

2015年11月于北京大学

第1版前言

FOREWORD

作为问题求解和程序设计的重要基础，算法设计与分析在计算机科学与技术专业的课程体系中是一门重要的必修课. 通过该课程的学习，不但为学习其他专业课程奠定了扎实的基础，而且对培养学生分析与解决问题的能力及计算思维有着不可替代的作用. *ACM IEEE Computing Curricula* 2004 与我国教育部计算机科学与技术专业教学指导委员会提出的《计算机科学与技术专业规范 2005》都把该课程列入本专业的核心课程之一.

本书是国家高等教育“十一五”规划教材《算法设计与分析》(清华大学出版社出版，屈婉玲等编著)的辅助教材. 主教材包括算法设计、算法分析、计算复杂性理论等重要内容. 结合各种典型应用，主教材首先深入分析了各种算法设计技术的适用范围、设计步骤、正确性证明与复杂度的分析方法、改进算法的途径、局限性等，为从事实际问题求解的算法设计与分析工作在理论上提供清晰的、整体的思路和方法，并在此基础上介绍了问题难度的分析方法和计算复杂性理论的基本框架和一些重要的结果.

算法具有广泛的应用背景，习题量大，方法灵活. 针对给定算法问题，在建模、设计技术选择、效率分析、改进途径等方面，初学者往往不知道如何着手. 本书在多年算法教学的基础上精选了 100 多道典型的习题，给出了详尽的解答和分析，以期对初学者有所帮助.

与主教材配套，本书也分为 10 章. 第 1 章是基础知识；第 2～5 章分别阐述分治策略、动态规划、贪心法、回溯与分支限界等算法设计技术；第 6 章介绍算法分析和问题的计算复杂度；第 7 章是 NP 完全性理论；第 8 章是近似算法；第 9 章是随机算法；第 10 章介绍处理难解问题的策略. 每章首先对所涉及的重要知识点和方法进行总结，然后给出习题和解答.

本书前 4 章由屈婉玲编写，第 5～6 章由王捍贫编写，第 7～8 章由张立昂编写，第 9～10 章由刘田编写.

为了提高本书的质量，欢迎广大读者的批评和指正！

作　者

2014 年 3 月于北京大学

目　录

CONTENTS

第 1 章 基础知识

1.1 内 容 提 要

1. 基本概念

算法 有限条指令的序列，确定了解决某个问题的运算或操作顺序.

算法 A 解问题 P 把问题 P 的任何实例作为算法 A 的输入，A 能够在有限步停机，并输出该实例的正确的解.

算法的时间复杂度

最坏情况下的时间复杂度：算法求解输入规模为 n 的实例所需的最多基本运算次数，通常记作 $W(n)$.

平均情况下的时间复杂度：针对输入规模为 n 的实例的概率分布，算法求解这些实例所需的基本运算次数的概率平均，通常记作 $A(n)$.

函数的渐近的界 设 f 和 g 是定义域为自然数集 **N** 上的函数.

(1) 若存在正数 c 和 n_0，使得对一切 $n\geqslant n_0$，有 $0\leqslant f(n)\leqslant cg(n)$ 成立，则称 $f(n)$ 的**渐近的上界**是 $g(n)$，记作 $f(n)=O(g(n))$.

(2) 若存在正数 c 和 n_0，使得对一切 $n\geqslant n_0$，有 $0\leqslant cg(n)\leqslant f(n)$ 成立，则称 $f(n)$ 的**渐近的下界**是 $g(n)$，记作 $f(n)=\Omega(g(n))$.

(3) 若对于任意正数 c 都存在 n_0，使得当 $n\geqslant n_0$ 时有 $0\leqslant f(n)<cg(n)$ 成立，则记作 $f(n)=o(g(n))$.

(4) 若对于任意正数 c 都存在 n_0，使得当 $n\geqslant n_0$ 时有 $0\leqslant cg(n)<f(n)$ 成立，则记作 $f(n)=\omega(g(n))$.

(5) 若 $f(n)=O(g(n))$ 且 $f(n)=\Omega(g(n))$，则记作 $f(n)=\Theta(g(n))$.

2. 某些重要的结果

定理 1.1 设 f 和 g 是定义域为自然数集合的函数.

(1) 如果 $\lim\limits_{n\to\infty}\dfrac{f(n)}{g(n)}$ 存在，并且等于某个常数 $c>0$，那么 $f(n)=\Theta(g(n))$.

(2) 如果 $\lim\limits_{n\to\infty}\dfrac{f(n)}{g(n)}=0$，那么 $f(n)=o(g(n))$.

(3) 如果 $\lim\limits_{n\to\infty}\dfrac{f(n)}{g(n)}=+\infty$，那么 $f(n)=\omega(g(n))$.

定理 1.2 设 f、g 和 h 是定义域为自然数集合的函数：

(1) 如果 $f=O(g)$ 且 $g=O(h)$，那么 $f=O(h)$.

(2) 如果 $f=\Omega(g)$且 $g=\Omega(h)$,那么 $f=\Omega(h)$.

(3) 如果 $f=\Theta(g)$且 $g=\Theta(h)$,那么 $f=\Theta(h)$.

定理 1.3 假设 f 和 g 是定义域为自然数集合的函数,若对某个其他的函数 h,有 $f=O(h)$和 $g=O(h)$,那么 $f+g=O(h)$.

关于几类函数的阶的结果

多项式函数: $f(n)=a_0+a_1n+a_2n^2+\cdots+a_dn^d$,有 $f(n)=\Theta(n^d)$.

对数函数: 对每个 $b>1$ 和每个 $a>0$,有 $\log_bn=o(n^a)$;对于不同的 a 与 b, $a,b>1$, $\log_an=\Theta(\log_bn)$.

指数函数: 对每个 $r>1$ 和每个 $d>0$,r^n 满足 $n^d=o(r^n)$.

阶乘函数: 斯特灵(Stirling)公式 $n!=\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\Theta\left(\frac{1}{n}\right)\right)$.

求和方法 利用公式或者以积分作为近似结果,其中常用的公式是:

$$\sum_{k=1}^{n}a_k=\frac{n(a_1+a_n)}{2}$$

$$\sum_{k=0}^{n}aq^k=\frac{a(1-q^{n+1})}{1-q}$$

$$\sum_{k=0}^{n}x^k=\frac{1-x^{n+1}}{1-x}$$

$$\sum_{k=1}^{n}\frac{1}{k}=\ln n+O(1)$$

递推方程求解方法 公式法、迭代归纳法、尝试法、递归树、主定理.

主定理(Master Theorem) 设 $a\geqslant1,b>1$ 为常数,$f(n)$为函数,$T(n)$为非负整数,且

$$T(n)=aT(n/b)+f(n)$$

则有以下结果:

(1) 若 $f(n)=O(n^{\log_ba-\varepsilon})$,$\varepsilon>0$,那么 $T(n)=\Theta(n^{\log_ba})$.

(2) 若 $f(n)=\Theta(n^{\log_ba})$,那么 $T(n)=\Theta(n^{\log_ba}\log n)$.

(3) 若 $f(n)=\Omega(n^{\log_ba+\varepsilon})$,$\varepsilon>0$,且对于某个常数 $c<1$ 和所有充分大的 n,有 $af(n/b)\leqslant cf(n)$,那么 $T(n)=\Theta(f(n))$.

3. 算法

顺序搜索算法 Search(L,x),在 L 中查找 x. 基本运算为 x 与 L 中元素的比较,算法最坏情况下的时间为 $W(n)=O(n)$.

插入排序算法 InserSort(A,n),对 n 个元素的数组 A 排序,基本运算为 A 中元素的比较,算法最坏情况下的时间为 $W(n)=O(n^2)$.

二分归并排序算法 MergeSort(A,p,r) 对数组 $A[p..r]$排序,基本运算为 A 中元素的比较,最坏情况下的时间为 $W(n)=O(n\log n)$.

Hanoi 塔的递归算法 Hanoi(A,C,n),把 n 个圆盘从 A 柱移到 C 柱,基本运算为 1 个圆盘的 1 次移动,时间复杂度为 2^n-1.

1.2 习 题

1.1 设 A 是 n 个不等的数的数组，$n>2$. 以比较作为基本运算，试给出一个 $O(1)$ 时间的算法，找出 A 中一个既不是最大也不是最小的数. 写出算法的伪码，说明该算法在最坏情况下执行的比较次数.

1.2 考虑下述选择排序算法：

算法 ModSelectSort

输入：n 个不等的整数的数组 $A[1..n]$

输出：按递增次序排序的 A

```
1.  for i←1 to n−1 do
2.      for j←i+1 to n do
3.          if A[j]<A[i] then A[i]↔A[j]
```

问：

(1) 最坏情况下该算法做多少次比较运算？

(2) 最坏情况下该算法做多少次交换运算？这种情况在什么输入条件下发生？

1.3 给定正整数的数组 $A[1..n]$，测试 A 的每个元素 $A[i]$ 的奇偶性. 如果 $A[i]$ 是奇数，则将它 2 倍后输出；否则直接输出 $A[i]$.

(1) 以乘法作为基本运算，使用大 O 记号，还是使用大 Θ 记号，哪个记号能够正确表达这个算法对于规模为 n 的输入所做的基本运算次数？为什么？

(2) 如果以元素的测试作为基本运算，重复问题(1).

1.4 计算下述算法所执行的加法次数.

算法 1

输入：$n=2^t$，t 为正整数

输出：k

```
1.  k←0
2.  while n≥1 do
3.      for j←1 to n do
4.          k←k+1
5.      n←n/2
6.  return k
```

1.5 计数算法 C 所执行的加法次数.

算法 C

输入：n 为正整数

输出：k

```
1.  k←0
2.  for i←1 to n do
3.          m←⌊n/i⌋
4.      for j←1 to m do
5.              k←k+1
```

```
6.  return k
```

1.6 阅读关于下述算法 A 的伪码,说明该算法求解的是什么问题,并计算该算法所做的乘法运算(*)和加法运算次数.

```
算法 A
输入:数组 P[0..n],实数 x
输出:y
1.  y←P[0]; power←1
2.  for i←1 to n do
3.      power←power*x
4.      y←y+P[i]*power
5.  return y
```

1.7 下述 Find-Second-Min 算法是找第二小算法. 输入是 n 个不等的数构成的数组 S,输出是第二小的数 *SecondMin*.

(1) 在最坏情况下,该算法做多少次比较?

(2) 若所有输入是等概率分布的,平均情况下该算法做多少次比较?

```
算法 Find-Second-Min(S,n)
1.  if S[1]<S[2]
2.  then min←S[1]; SecondMin←S[2]
3.  else min←S[2]; SecondMin←S[1]
4.  for i←3 to n do
5.      if S[i]<SecondMin
6.      then if S[i]<min
7.              then SecondMin←min; min←S[i]
8.              else SecondMin←S[i]
```

1.8 已知 L 是含有 n 个元素并且排好序的数组,x 在 L 中. 如果 x 出现在 L 中第 i 个 ($i=2,3,\cdots,n$)位置的概率是在前一个位置概率的一半,当 n 充分大时,估计下述查找算法平均情况下的时间复杂度 $A(n)$. 只需给出近似值.

```
算法:顺序查找
1.  j←1
2.  while j≤n and x>L[j] do
3.      j←j+1
4.  if x<L[j] or j>n
5.  then j←0
```

1.9 设 A 为 n 个不等的数的数组. 给定 x,若 x 在 A 中,输出 x 的下标 k;若 x 不在 A 中,输出 0. BinarySearch 和 LinearSearch 分别表示二分和顺序搜索,设计下述算法:

```
算法 Search(A,x)
1. if n 为奇数 then k←BinarySearch(A,x)
2. else k←LinearSearch(A,x)
```

以比较作基本运算,用大 O 记号表示算法在最坏情况下的时间复杂度 $W(n)$;能否使用大 Θ 记号表示 $W(n)$?为什么?

1.10 考虑下述素数测试算法：

算法 PrimalityTest(n)

输入：n，n 为大于 2 的奇整数

输出：true 或者 false

1. $s \leftarrow \lfloor \sqrt{n} \rfloor$
2. for $j \leftarrow 2$ to s
3. if j 整除 n
4. then return false
5. return true

(1) 假设计算$\lfloor \sqrt{n} \rfloor$可以在$O(1)$时间完成，估计该算法在最坏情况下的时间复杂度.

(2) 能否使用Θ符号表示这个算法在最坏情况下的时间复杂度？为什么？

1.11 证明定理 1.3：假设 f 和 g 是定义域为自然数集合的函数，若对某个其他函数 h 有 $f=O(h)$和 $g=O(h)$成立，那么 $f+g=O(h)$.

1.12 证明定理 1.4：对每个 $b>1$ 和每个 $a>0$，有 $\log_b n=o(n^a)$.

1.13 证明定理 1.5：对每个 $r>1$ 和每个 $d>0$，有 $n^d=o(r^n)$.

1.14 设 x 为实数，n、a 和 b 为整数，证明下述性质：

(1) $x-1<\lfloor x \rfloor \leqslant x \leqslant \lceil x \rceil < x+1$

(2) $\lfloor x+n \rfloor = \lfloor x \rfloor + n$，$\lceil x+n \rceil = \lceil x \rceil + n$

(3) $\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor = n$

(4) $\left\lceil \frac{\left\lceil \frac{n}{a} \right\rceil}{b} \right\rceil = \left\lceil \frac{n}{ab} \right\rceil$，$\left\lfloor \frac{\left\lfloor \frac{n}{a} \right\rfloor}{b} \right\rfloor = \left\lfloor \frac{n}{ab} \right\rfloor$

1.15 考虑下面每对函数 $f(n)$和 $g(n)$，如果它们的阶相等，则使用 Θ 记号；否则使用 O 记号表示它们的关系.

(1) $f(n)=(n^2-n)/2$，$g(n)=6n$

(2) $f(n)=n+2\sqrt{n}$，$g(n)=n^2$

(3) $f(n)=n+n\log n$，$g(n)=n\sqrt{n}$

(4) $f(n)=2\log^2 n$，$g(n)=\log n+1$

(5) $f(n)=\log(n!)$，$g(n)=n^{1.05}$

1.16 在表 1.1 中填入 true 或 false.

表 1.1 函数 f 与 g

	$f(n)$	$g(n)$	$f(n)=O(g(n))$	$f(n)=\Omega(g(n))$	$f(n)=\Theta(g(n))$
1	$2n^3+3n$	$100n^2+2n+100$			
2	$50n+\log n$	$10n+\log\log n$			
3	$50n\log n$	$10n\log\log n$			
4	$\log n$	$\log^2 n$			
5	$n!$	5^n			

1.17 对于下面每个函数 $f(n)$,用 Θ 符号表示成 $f(n)=\Theta(g(n))$ 的形式,其中 $g(n)$ 要尽可能简洁. 比如 $f(n)=n^2+2n+3$ 可以写成 $f(n)=\Theta(n^2)$. 然后,按照阶递增的顺序将这些函数进行排列.

$$(n-2)!,\quad 5\log(n+100)^{10},\quad 2^{2n},\quad 0.001n^4+3n^3+1$$

$$(\ln n)^2,\quad \sqrt[3]{n}+\log n,\quad 3^n,\quad \log(n!),\quad \log(n^{n+1}),\quad 1+\frac{1}{2}+\cdots+\frac{1}{n}$$

1.18 对以下函数,按照它们的阶从高到低排列;如果 $f(n)$ 与 $g(n)$ 的阶相等,表示为 $f(n)=\Theta(g(n))$.

$$2^{\sqrt{2\log n}},\quad n\log n,\quad \sum_{k=1}^{n}\frac{1}{k},\quad n2^n,\quad (\log n)^{\log n},\quad 2^{2n},\quad 2^{\log\sqrt{n}}$$

$$n^3,\quad \log(n!),\quad \log n,\quad \log\log n,\quad n^{\log\log n},\quad n!,\quad n,\quad \log 10^n$$

1.19 求解以下递推方程:

(1) $\begin{cases}T(n)=T(n-1)+n^2\\T(1)=1\end{cases}$

(2) $\begin{cases}T(n)=9T(n/3)+n\\T(1)=1\end{cases}$

(3) $\begin{cases}T(n)=T\left(\frac{n}{2}\right)+T\left(\frac{n}{4}\right)+cn,\ c\text{ 为常数}\\T(1)=1\end{cases}$

(4) $\begin{cases}T(n)=T(n-1)+\log 3^n\\T(1)=1\end{cases}$

(5) $\begin{cases}T(n)=5T(n/2)+(n\log n)^2\\T(1)=1\end{cases}$

(6) $\begin{cases}T(n)=2T\left(\frac{n}{2}\right)+n^2\log n\\T(1)=1\end{cases}$

(7) $\begin{cases}T(n)=T(n-1)+\frac{1}{n}\\T(1)=1\end{cases}$

(8) $T(n)=T(n-1)+\log n$,估计 $T(n)$ 的阶

1.20 设递推方程 $T(n)=7T(n/2)+n^2$ 给出了算法 A 在最坏情况下的时间复杂度函数,算法 B 在最坏情况下的时间复杂度函数 $W(n)$ 满足递推方程 $W(n)=aW(n/4)+n^2$. 试确定最大的正整数 a 使得 $W(n)$ 的阶低于 $T(n)$ 的阶.

1.21 设原问题的规模是 n,从下述3个算法中选择一个最坏情况下时间复杂度最低的算法,简要说明你的理由.

算法 A:将原问题划分规模减半的5个子问题,递归求解每个子问题,然后在线性时间将子问题的解合并得到原问题的解.

算法 B:先递归求解2个规模为 $n-1$ 的子问题,然后在常量时间内将子问题的解合并.

算法 C:将原问题划分规模为 $n/3$ 的9个子问题,递归求解每个子问题,然后在 $O(n^3)$ 时间将子问题的解合并得到原问题的解.

1.3 习题解答与分析

1.1 算法

输入：数组 A

输出：A 中既不是最大也不是最小的一个数

1. 任选 3 个数 a_1, a_2, a_3
2. if $a_1 < a_2$
3. then if $a_1 > a_3$
4. then return a_1
5. else return $\min\{a_2, a_3\}$
6. else if $a_2 > a_3$ then return a_2
7. else return $\min\{a_1, a_3\}$

算法至多比较 3 次.

1.2 (1) 最坏情况下的比较次数是：

$$W(n) = \sum_{i=1}^{n-1}(n-i) = 1+2+\cdots+n-1 = n(n-1)/2$$

(2) 当输入的 n 个数彼此不等且按递降次序排列时，比较次数是 $n(n-1)/2$. 每次比较后都发生交换，因此最坏情况下的交换次数也是 $n(n-1)/2$.

1.3 (1) 对于任意大的 n，若 n 个数都是奇数，则算法做 n 次乘法；若 n 个数都是偶数，则算法做 0 次乘法. 因此乘法次数可以表示为 $O(n)$，但是不能表示成 $\Theta(n)$.

(2) 对于 A 中的每个数，不管是奇数还是偶数，算法都做 1 次测试，测试次数可以表示为 $O(n)$，也可以表示为 $\Theta(n)$.

1.4 第一次 for 循环执行 n 次加法，第 2 次 for 循环执行 $n/2$ 次加法……直到最后执行 1 次加法，加法总次数为

$$T(n) = n + \frac{n}{2} + \frac{n}{2^2} + \frac{n}{2^3} + \cdots + 2 + 1 = 2n - 1$$

1.5 该算法的 for 循环执行 n 次，总计加法次数为

$$W(n) = n + \lfloor n/2 \rfloor + \lfloor n/3 \rfloor + \cdots + \lfloor n/n \rfloor$$

由于

$$\sum_{i=1}^{n}\left(\frac{n}{i}-1\right) \leqslant \sum_{i=1}^{n}\left\lfloor\frac{n}{i}\right\rfloor \leqslant \sum_{i=1}^{n}\frac{n}{i}$$

于是

$$W(n) = \Theta(n\log n)$$

1.6 给定实数 x，算法 A 是对多项式 $P(x) = p_0 + p_1 x + \cdots + p_n x^n$ 求值，其中 $p_i = P[i], i = 0, 1, \cdots, n$. 算法 A 执行 $2n$ 次乘法，n 次加法.

1.7 (1) 行 4 的 for 循环执行 $n-2$ 次，每次至多做 2 次比较(行 5 和行 6). 行 1 做 1 次比较，算法比较次数至多为

$$W(n) = 2(n-2)+1 = 2n-3$$

(2) 对于 i,在 $S[1..i]$的数之间,如果 $S[i]$是最小的 2 个数,则需要比较 2 次;如果是较大的 $i-2$ 个数,则需要比较 1 次. 由于等概率,处于较小的 2 个数的概率是 $2/i$,处于较大的 $i-2$ 个数的概率是 $(i-2)/i$,因此有

$$\begin{aligned}A(n)&=1+\sum_{i=3}^{n}\left(2\cdot\frac{2}{i}+1\cdot\frac{i-2}{i}\right)\\&=n-1+2\sum_{i=3}^{n}\frac{1}{i}\\&\approx n-1+2(\ln n-3/2)\\&=n-4+2\ln n\end{aligned}$$

1.8 设 x 恰好在第一位的概率为 p,在第二位的概率为 $p/2$……那么 x 恰好在第 i 位的概率为 $p/2^{i-1}$,于是

$$p+\frac{p}{2}+\frac{p}{2^2}+\cdots+\frac{p}{2^{n-1}}=1\Rightarrow p=\frac{2^{n-1}}{2^n-1}\approx\frac{1}{2}$$

平均比较次数是

$$\begin{aligned}A(n)&=p\cdot 1+\frac{p}{2}\cdot 2+\frac{p}{2^2}\cdot 3+\cdots+\frac{p}{2^{n-1}}\cdot n\\&=p\left(1+\frac{2}{2}+\frac{3}{2^2}+\cdots+\frac{n}{2^{n-1}}\right)\approx\frac{1}{2}\cdot 4=2\end{aligned}$$

1.9 n 是奇数时使用二分搜索,时间复杂度为 $O(\log n)$;n 是偶数时使用顺序搜索,时间复杂度为 $O(n)$. 因此以比较作基本运算有 $W(n)=O(n)$. 不能写 $W(n)=\Theta(n)$,因为 n 为奇数时,$\log n$ 不能写成 $\Omega(n)$.

1.10 (1) 以行 3 的除法作为基本运算,行 2 的循环最多执行 $s-1$ 次,于是

$$W(n)=O(s)=O(\sqrt{n})$$

(2) 不能使用 Θ 符号表示算法最坏情况下的时间复杂度,因为对于大素数 n,算法确实需要做$\lfloor\sqrt{n}\rfloor$次除法. 但是如果 $n=3k$,k 为任意大的奇整数,那么算法只需 2 次除法;因此不存在常数 c 和 n_0,使得当 $n\geqslant n_0$ 时都有 $c\lfloor\sqrt{n}\rfloor\leqslant W(n)$ 成立,即$\sqrt{n}$不是 $W(n)$的渐近的下界.

1.11 **证** 根据 $f=O(h)$和 $g=O(h)$,存在 $c_1>0$ 和正整数 n_1 使得当 $n\geqslant n_1$ 时,有 $f(n)\leqslant c_1h(n)$. 同理,存在 $c_2>0$ 和正整数 n_2,使得当 $n\geqslant n_2$ 时,有 $f(n)\leqslant c_2h(n)$. 取 $c=\max\{2c_1,2c_2\}$,$n_0=\max\{n_1,n_2\}$,那么当 $n\geqslant n_0$ 时,有

$$f(n)+g(n)\leqslant c_1h(n)+c_2h(n)\leqslant ch(n)$$

其中 c 为常数,因此 $f+g=O(h)$.

1.12 **证** 因为

$$\lim_{n\to+\infty}\frac{\log_b n}{n^a}=\lim_{n\to+\infty}\frac{\frac{1}{n\ln b}}{\alpha n^{\alpha-1}}=\lim_{n\to+\infty}\frac{1}{\alpha\ln b}\frac{1}{n^a}=0$$

根据定理 1.1 命题得证.

1.13 **证** 若 $d\leqslant 1$,显然有

$$\lim_{n\to+\infty}\frac{n^d}{r^n}=\lim_{n\to+\infty}\frac{dn^{d-1}}{r^n\ln r}=\frac{d}{\ln r}\lim_{n\to+\infty}\frac{n^{d-1}}{r^n}=0$$

若 $d>1$,根据洛必达法则,上式中 n^d 将通过逐次求导降低为 n^{d-1},n^{d-2},…,直到 n^{d_0},其中

$d_0<1$. 而分母的指数函数不变. 根据前面的结果,上式极限为 0. 由定理 1.1 命题得证.

1.14 证 (1) 如果 x 是整数 n,根据定义$\lfloor x \rfloor=\lceil x \rceil=n$,从而有

$$x-1<\lfloor x \rfloor=x=\lceil x \rceil<x+1$$

如果 $n<x<n+1$,n 为整数,那么$\lfloor x \rfloor=n$,$\lceil x \rceil=n+1$,从而有

$$x-1<n=\lfloor x \rfloor \quad n<x<n+1=\lceil x \rceil$$
$$\Rightarrow x-1<n=\lfloor x \rfloor<x<\lceil x \rceil=n+1<x+1$$

综合上述命题得证.

(2) 易见,当 x 为整数时命题显然正确,假设 $m<x<m+1$,其中 m 是整数,那么

$$m+n<x+n<m+1+n$$

根据定义有

$$\lfloor x+n \rfloor=m+n=\lfloor x \rfloor+n$$
$$\lceil x+n \rceil=m+1+n=\lceil x \rceil+n$$

(3) 若 $n=2k$,k 为整数,那么

$$\left\lceil \frac{n}{2} \right\rceil+\left\lfloor \frac{n}{2} \right\rfloor=k+k=2k=n$$

若 $n=2k+1$,k 为整数,那么

$$\left\lceil \frac{n}{2} \right\rceil+\left\lfloor \frac{n}{2} \right\rfloor=(k+1)+k=2k+1=n$$

(4) 易见,当 $n=pab$ 时(其中 p 是整数),命题成立. 假设 $n=pab+r$,其中 $0<r<ab$. 那么

$$\left\lceil \frac{\left\lceil \frac{n}{a} \right\rceil}{b} \right\rceil=\left\lceil \frac{\left\lceil \frac{pab+r}{a} \right\rceil}{b} \right\rceil=\left\lceil \frac{pb+\left\lceil \frac{r}{a} \right\rceil}{b} \right\rceil=\left\lceil p+\frac{\left\lceil \frac{r}{a} \right\rceil}{b} \right\rceil=p+\left\lceil \frac{\left\lceil \frac{r}{a} \right\rceil}{b} \right\rceil=p+1$$

上式的最后一步是由于 $0<r<ab$,即 $0<\lceil r/a \rceil \leqslant b$. 同理也可以得到

$$\left\lceil \frac{n}{ab} \right\rceil=\left\lceil \frac{pab+r}{ab} \right\rceil=\left\lceil p+\frac{r}{ab} \right\rceil=p+\left\lceil \frac{r}{ab} \right\rceil=p+1$$

类似地可以证明命题的另一半.

1.15 (1) $g(n)=O(f(n))$

(2) $f(n)=O(g(n))$

(3) $f(n)=O(g(n))$

(4) $g(n)=O(f(n))$

(5) $f(n)=O(g(n))$

1.16

函数	$f(n)$	$g(n)$	$f(n)=O(g(n))$	$f(n)=\Omega(g(n))$	$f(n)=\Theta(g(n))$
1	$2n^3+3n$	$100n^2+2n+100$	false	true	false
2	$50n+\log n$	$10n+\log\log n$	true	true	true
3	$50n\log n$	$10n\log\log n$	false	true	false
4	$\log n$	$\log^2 n$	true	false	false
5	$n!$	5^n	false	true	false