“十二五”普通高等教育本科国家级规划教材

国家精品课程教材

计算机算法设计与分析习题解答

（第5版）

◎ 王晓东　编著

中国工信出版集团

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

“十二五”普通高等教育本科国家级规划教材
国家精品课程教材

计算机算法设计与分析习题解答

（第5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

本书是与“十二五”普通高等教育本科国家级规划教材《计算机算法设计与分析（第5版）》配套的辅助教材和国家精品课程教材，分别对主教材中的算法分析题和算法实现题给出了解答或解题思路提示。为了提高学生灵活运用算法设计策略解决实际问题的能力，本书还将主教材中的许多习题改造成算法实现题，要求学生设计出求解算法并上机实现。本书教学资料包含各章算法实现题、测试数据和答案，可在华信教育资源网免费注册下载。

本书内容丰富，理论联系实际，可作为高等学校计算机科学与技术、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学习计算机算法设计的辅助教材，也是工程技术人员和自学者的参考书。

图书在版编目（CIP）数据

计算机算法设计与分析习题解答/王晓东编著. —5版. —北京：电子工业出版社，2018.10

ISBN 978-7-121-34438-1

Ⅰ. ① 计… Ⅱ. ① 王… Ⅲ. ① 电子计算机－算法设计－高等学校－题解 ② 电子计算机－算法分析－高等学校－题解 Ⅳ. ① TP301.6-44

中国版本图书馆CIP数据核字（2018）第120711号

策划编辑：章海涛
责任编辑：章海涛
印　　刷：三河市良远印务有限公司
装　　订：三河市良远印务有限公司
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：22.75　　字数：580千字
版　　次：2005年8月第1版
　　2018年10月第5版
印　　次：2018年10月第1次印刷
定　　价：56.00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zlts@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：192910558（QQ群）。

前　言

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。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，在内容选材、深度把握、系统性和可用性方面进行了精心设计，力图适合高校本科生教学对学时数和知识结构的要求。

本书是“十二五”普通高等教育本科国家级规划教材《计算机算法设计与分析（第5版）》（ISBN 978-7-121-34439-8）配套的辅助教材，对《计算机算法设计与分析（第5版）》一书中的全部习题做了详尽的解答，旨在让使用该书的教师更容易教，学生更容易学。为了便于对照阅读，本书的章序与《计算机算法设计与分析（第5版）》一书的章序保持一致，且一一对应。

本书的内容是对《计算机算法设计与分析（第5版）》的较深入的扩展，许多教材中无法讲述的较深入的主题通过习题的形式展现出来。为了加强学生灵活运用算法设计策略解决实际问题的能力，本书将主教材中的许多习题改造成算法实现题，要求学生不仅设计出解决具体问题的算法，而且能上机实现。作者的教学实践反映出这类算法实现题的教学效果非常好。作者还结合国家精品课程建设，建立了“算法设计与分析”教学网站。国家精品资源共享课地址：http://www.icourses.cn/sCourse/course_2535.html。欢迎广大读者访问作者的教学网站并提出宝贵意见。

在本书编写过程中，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与工作环境。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！

作　者

目　录

第7章　概率算法

算法分析题7

7-1　模拟正态分布随机变量。

在实际应用中，常需模拟服从正态分布的随机变量，其密度函数为

$$\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{(x-a)^2}{2\sigma^2}}$$

式中，a 为均值，σ 为标准差。

如果 s 和 t 是(−1, 1)中均匀分布的随机变量，且 $s^2+t^2<1$，令

$$p=s^2+t^2$$
$$q=\sqrt{(-2\ln p)/p}$$
$$u=sq$$
$$v=tq$$

则 u 和 v 是服从标准正态分布（a=0，σ=1）的两个互相独立的随机变量。

（1）利用上述事实，设计一个模拟标准正态分布随机变量的算法。

（2）将上述算法扩展到一般的正态分布。

分析与解答：

（1）模拟标准正态分布随机变量的算法如下。

```
double RandomNumber::Norm() {
   double  s, t, p, q;
   while(true) {
      s = 2*fRandom()-1;
      t = 2*fRandom()-1;
      p = s*s+t*t;
      if(p < 1)
         break;
   }
   q = sqrt((-2*log(p))/p);
   return s*q;
}
```

（2）扩展到一般的正态分布的算法如下。

```
double RandomNumber::Norm(double a, double b) {
   double  x = Norm();
   return a+b*x;
}
```

7-2　随机抽样算法。

设有一个文件含有 n 个记录。

（1）试设计一个算法随机抽取该文件中 m 个记录。

（2）如果事先不知道文件中记录的个数，应如何随机抽取其中的 m 个记录？

分析与解答：

（1）以概率 m/n 抽取记录。该方法的标准差是 $\sqrt{m(1-m/n)}$，有时可能不满足要求。

假设在前 t 次考察的记录中，已抽取了 k 个记录，接下来第 $t+1$ 次考察取得第 $k+1$ 个记录的概率为

$$\binom{n-t-1}{n-k-1}\Big/\binom{n-t}{n-k}=\frac{n-k}{n-t}$$

按此概率抽取记录是无偏的。

抽样算法如下。

```
void samp(int n, int m, int s[]) {
   RandomNumber rnd;
   int  x = 0, y = 0, k;
   while(y < m) {
      double  u = rnd.fRandom();
      k=u*n;
      if(u*(n-x) < (n-y)) {
         s[k]++;
         y++;
      }
      x++;
   }
}
```

（2）如果事先不知道文件中记录的个数，通常可以先做一次扫描，确定记录的个数后再抽样。另一个较好的方法是在扫描时预先随机抽取 $p>m$ 个记录，然后对抽取出的 p 个记录做 2 次抽样，从中随机抽取 m 个记录。

7-3 随机产生 m 个整数。

试设计一个算法，随机地产生范围在 $1\sim n$ 的 m 个随机整数，且要求这 m 个随机整数互不相同。

分析与解答：与算法分析题 7-2 类似，解此问题的 Floyd 算法如下。

```
void Floyd(int n, int m, int s[]) {
   RandomNumber  rnd;
   int  y = 0;
   while(y < m) {
      for(int j=n-m+1; j <= n; j++) {
         double  u = rnd.fRandom();
         int  k = 1+j*u;
         if(s[k] == 0) {
            s[k]++;
            y++;
         }
         else if(s[j]==0) {
```

```
            s[j]++;
            y++;
        }
    }
  }
}
```

7-4 集合大小的概率算法。

设 X 是含有 n 个元素的集合，从 X 中均匀地选取元素。设第 k 次选取时首次出现重复。

（1）试证明当 n 充分大时，k 的期望值为 $\beta\sqrt{n}$。其中，$\beta\sqrt{\pi/2}=1.253$。

（2）由此设计一个计算给定集合 X 中元素个数的概率算法。

分析与解答：

（1）从含有 n 个元素的集合 X 中均匀地选取元素，第 k 次选取时首次出现重复的概率

$$\frac{\binom{n}{k-1}(k-1)!(k-1)}{n^k}$$

k 的期望值为

$$E(k)=\sum_{k=1}^{n}\frac{\binom{n}{k-1}(k-1)!(k-1)k}{n^k}$$

用 Stirling 公式

$$n!=\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\frac{1}{12n}+\theta\left(\frac{1}{n^2}\right)\right)$$

代入计算可得

$$E(k)=\sqrt{\frac{\pi n}{2}-\frac{1}{3}+\theta\left(\frac{1}{\sqrt{n}}\right)}$$

（2）由（1）的结果可设计计算给定集合中元素个数的概率算法如下。

```
int count(SET X) {
  int  k = 0;
  SET  S;
  a = uniform(X);
  while(true) {
    S.insert(a);
    k++;
    a = uniform(X);
    if(S.find(a) != S.end())
      break;
  }
  return 2*k*k/pi;
}
```

7-5 生日问题。

试设计一个随机化算法计算 $365!/340!365^{25}$，并精确到 4 位有效数字。

分析与解答：该问题中的数是概率论中著名的生日问题的解答。在 k 个人中，至少 2 人有相同生日的概率为 $1-365!/[(365-k)!\times 365^k]$。

一般情况下，$n!/[(n-k)!n^k]$可以用 Stirling 公式化简后做近似计算。

由 Stirling 公式

$$n!=\sqrt{2\pi n}\left(\frac{n}{\mathrm{e}}\right)^n\left[1+\theta\left(\frac{1}{n}\right)\right]$$

$$\ln(x)=x-x^2/2+\theta(x^3)$$

$$n!/\left[(n-k)!n^k\right]=\left[1+\theta\left(\frac{1}{n}\right)\right]\left(\frac{\sqrt{2\pi n}}{\sqrt{2\pi(n-k)}}\right)\frac{\left(\frac{n}{\mathrm{e}}\right)^n}{\left(\frac{n-k}{\mathrm{e}}\right)^{n-k}n^k}$$

$$=\left[1+\theta\left(\frac{1}{n}\right)\right]\left(\frac{n}{n-k}\right)^{n-k}\mathrm{e}^{-k}$$

$$=\left[1+\theta\left(\frac{1}{n}\right)\right]\mathrm{e}^{(n-k)\ln\left(1+\frac{k}{n-k}-k\right)}$$

$$=\left[1+\theta\left(\frac{1}{n}\right)\right]\mathrm{e}^{(n-k)\left[1+\frac{k}{n-k}-\frac{k^2}{2(n-k)^2}+\theta\left(\frac{1}{(n-k)^3}\right)\right]-k}$$

$$=\left[1+\theta\left(\frac{1}{n}\right)\right]\mathrm{e}^{-k^2/(2n)+\theta\left(\frac{1}{n^2}\right)}$$

由此可得

$$n!/\left[(n-k)!n^k\right]\approx\mathrm{e}^{-k^2/(2n)}$$

用更精确些的估计式

$$n!/\sqrt{2\pi n}\left(\frac{n}{\mathrm{e}}\right)^n\left[1+\frac{1}{12n}+\theta\left(\frac{1}{n^2}\right)\right]$$

$$\ln(x)=x-x^2/2+x^3/3-\theta(x^4)$$

可得

$$n!/[(n-k)!n^k]=\mathrm{e}^{-k(k-1)/(2n-k^3)/(6n^2)\pm O(\max(k^2/n^2,\,k^4/n^3))}$$

由此可得

$$365!/(340!\times365^{25})=0.4311$$

$$1-365!/(340!\times365^{25})=0.5689$$

7-6 易验证问题的拉斯维加斯算法。

一个问题是易验证的是指对该问题的给定实例的每个解，都可以有效地验证其正确性。例如，求一个整数的非平凡因子问题是易验证的，而求一个整数的最小非平凡因子就不是易验证的。在一般情况下，易验证问题未必是易解的。

（1）给定一个解易验证问题 P 的蒙特卡罗方法，设计一个相应的解问题 P 的拉斯维加斯算法。

（2）给定一个解易验证问题 P 的拉斯维加斯算法，设计一个相应的解问题 P 的蒙特卡罗算法。

分析与解答：

（1）设给定的解易验证问题 P 的蒙特卡罗算法为 Monte，验证其解的正确性的算法为 charact，则相应的解问题 P 的拉斯维加斯算法 Las 如下。

```
void Las(ST x) {
   bool  r = Monte(x);
   while(!charact(x, r))
      r = Monte(x);
}
```

（2）设给定的解易验证问题 P 的拉斯维加斯算法为 Las。在超过时间界限时终止算法 Las。相应的解问题 P 的蒙特卡罗算法 Monte 如下。

```
bool Monte(ST x) {
   Las(x);
   if(timeused > maxt)
      return false;
   else
      return true;
}
```

7-7 用数组模拟有序链表。

用数组模拟有序链表的数据结构，设计支持下列运算的舍伍德型算法，并分析各种运算所需的计算时间：

（1）Predeceessor 找出一给定元素 x 在有序集 S 中的前驱元素；

（2）Successor 找出一给定元素 x 在有序集 S 中的后继元素；

（3）Min 找出有序集 S 中的最小元素；

（4）Max 找出有序集 S 中的最大元素。

分析与解答：对主教材中的有序链表类 OrderList 做简单修改。

7-8 $O(n^{3/2})$舍伍德型排序算法。

采用数组模拟有序链表的数据结构，设计一个舍伍德型排序算法，使算法最坏情况下的平均计算时间为 $O(n^{3/2})$。

分析与解答：用主教材中的有序链表类 OrderList，对有序链表做 n 次插入运算。第 i 次插入运算平均需要 $O(\sqrt{i})$ 计算时间。因此，相应的排序算法在最坏情况下的平均计算时间为 $O(n^{3/2})$。

7-9 n 后问题解的存在性。

如果对于某个 n 值，n 后问题无解，则算法将陷入死循环。

（1）证明或否定下述论断：对于 $n\geq 4$，n 后问题有解。

（2）是否存在正数 δ，使得对所有 $n\geq 4$ 算法成功的概率至少是 δ？

分析与解答：用 x_{ij} 表示在棋盘格子(i,j)处放置皇后的状态。当 $x_{ij}=1$ 时，表示在棋盘格子(i,j)处放置了一个皇后，否则没有放置皇后。n 后问题可以表示为如下的 0-1 线性规划问题。

$$\max \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} \leqslant 1 \qquad 1\leqslant i\leqslant n$$

$$\sum_{i=1}^{n} x_{ij} \leqslant 1 \qquad 1 \leqslant i \leqslant n$$

$$\sum_{i+j=k} x_{ij} \qquad 2 \leqslant k \leqslant 2n$$

$$\sum_{i-j=k} x_{ij} \qquad 1-n \leqslant k \leqslant n-1$$

$$x_{ij} \in \{0,1\} \qquad 1 \leqslant i, j \leqslant n$$

易知

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} \leqslant n$$

下面对 $n \geqslant 4$ 构造上述线性规划问题的一个解，使 $\max \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} = n$，从而证明，对于 $n \geqslant 4$，n 后问题有解，分为以下 3 种情况讨论。

① 当 $n \geqslant 4$ 为偶数且 $(n-2)\%6>0$ 时，对于 $1 \leqslant j \leqslant n/2$，取 $x(j, 2j)=1$，$x(n/2+j, 2j-1)=1$。

② 当 $n \geqslant 4$ 为偶数且 $n\%6>0$ 时，对于 $1 \leqslant j \leqslant n/2$，取 $x(j, k(j))=1$，$x(n+1-j, n-k(j))=1$。其中，$k(j)=(n/2+2(j-1)-1)\%n$。

③ 当 $n>4$ 为奇数时，取 $x(n, n)=1$，转换为 $(n-1)\times(n-1)$ 棋盘的问题，用（1）或（2）的方法构造其余值。

按照上述方法构造 n 后问题解的算法如下。

```
void construct(int k) {
   if(k < 4)
      return;
   if(odd(k))
      x[k] = k--;
   if((k-2)%6 > 0)
      build1(k);
   else
      build2(k);
}
void build1(int k) {
   k /= 2;
   for(int i=1; i <= k; i++) {
      x[i]=2*i;
      x[k+i]=2*i-1;
   }
}
void build2(int k) {
   for(int i=1; i <= k/2; i++) {
      x[i] = 1+(2*(i-1)+k/2-1)%k;
      x[k+1-i] = k(2*(i-1)+k/2-1)%k;
   }
}
```

容易验证，按照上述方法构造出的解是相应于 n 后问题的 0-1 线性规划问题的一个解。由此可见，对于 $n \geqslant 4$，n 后问题有解。

7-10　整数因子分解算法。

假设已有算法 Prime(n)可用于测试整数 n 是否为一素数，算法 Split(n)可以实现对合数 n 的因子分割。利用这两个算法，设计一个对给定整数 n 进行因子分解的算法。

分析与解答：因子分解算法如下。

```
void fact(int n) {
   if(prime(n)) {
      output(n);
      return;
   }
   int  i = split(n);
   if(i>1)
      fact(i);
   if(n>i)
      fact(n/i);
}
```

7-11　非蒙特卡罗算法的例子。

（1）试证明下面的算法 Primality 能以 80%以上的正确率判定给定的整数 n 是否为素数。另一方面，举出整数 n 的一个例子，表明算法对此整数 n 总是给出错误的解答，进而说明该算法不是一个蒙特卡罗算法。

```
bool Primality(int n) {
   if(gcd(n, 30030) == 1)
      return true;
   else
      return false;
}
```

（2）试找出上述算法 Primality 中可用于替换整数 30030 的另一个整数（可使用大整数），使得用此整数代替 30030 后，算法的正确率提高到 85%以上。

分析与解答：

（1）30030 是前 6 个素数的乘积，30030=2×3×5×7×11×13。因此，当合数含有素因子 2、3、5、7、11、13 时，算法 Primality 给出的解答是正确的。而当合数不含素因子 2、3、5、7、11、13 时，算法 Primality 给出的解答是错误的。例如，当 n=323=17×19 时，算法 Primality 给出的解答总是 true，但总是错误的。可见，算法 Primality 不是一个蒙特卡罗算法。

一般情况下，全体整数集合中含有素因子 p 的整数的比例为 $1/p$。设前 m 个素数为 p_1, p_2, …, p_m，由容斥原理知，全体整数集合中至少含有素因子 $p_1, p_2, \cdots, p_m$ 之一的整数的比例为

$$\sum_{i=1}^{m}\frac{1}{p_i}-\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\frac{1}{p_ip_j}+\cdots(-1)^{m-1}\frac{1}{p_1p_2\cdots p_m}=1-\prod_{i=1}^{m}\left(1-\frac{1}{p_i}\right)$$

对于本题容易计算出，全体整数集合中至少含有前 6 个素因子之一的整数的比例为

$$1-\left(1-\frac{1}{2}\right)\times\left(1-\frac{1}{3}\right)\times\left(1-\frac{1}{5}\right)\times\left(1-\frac{1}{7}\right)\times\left(1-\frac{1}{11}\right)\times\left(1-\frac{1}{13}\right)=0.8082$$

由此可见，算法 Primality 能以 80.82%以上的正确率判定给定的一个整数 n 是否为素数。

（2）要使算法的正确率提高到 85%以上，必须用前 m 个素数的乘积 $p_1p_2\cdots p_m$ 替代算法

中的常数 30030，使

$$1-\prod_{i=1}^{m}\left(1-\frac{1}{p_i}\right)>0.85$$

经计算得知：

当 m=11 时，$1-\prod_{i=1}^{11}\left(1-\frac{1}{p_i}\right)=0.8471$

当 m=12 时，$1-\prod_{i=1}^{12}\left(1-\frac{1}{p_i}\right)=0.8513$

可见，应该用 $p_1p_2\cdots p_{12}$=7420738134810 替代算法中的常数 30030 才能使算法的正确率提高到 85%以上。

7-12 重复 3 次的蒙特卡罗算法。

设 mc(x)是一致的 75%正确的蒙特卡罗算法，考虑下面的算法：

```
mc3(x) {
   int  t, u, v;
   t = mc(x);
   u = mc(x);
   v = mc(x);
   if ((t == u) || (t == v))
      return t;
   return v;
}
```

（1）试证明上述算法 mc3(x)是一致的 27/32 正确的算法，因此是 84%正确的。

（2）试证明如果 mc(x)不是一致的，则 mc3(x)的正确率有可能低于 71%。

分析与解答：

（1）重复 3 次的蒙特卡罗算法各次正确的分布有 8 种不同情况：

000，001，010，011，100，101，110，111

其中，011、101、110、111 这 4 种情况返回正确解。因此返回正确解的概率为

$$\frac{1}{4}\times\frac{3}{4}\times\frac{3}{4}+\frac{3}{4}\times\frac{1}{4}\times\frac{3}{4}+\frac{3}{4}\times\frac{3}{4}\times\frac{1}{4}+\frac{3}{4}\times\frac{3}{4}\times\frac{3}{4}=\frac{27}{32}$$

（2）如果 mc(x)不是一致的，则 110 不能保证返回正确解，因此返回正确解的概率可能低到

$$\frac{1}{4}\times\frac{3}{4}\times\frac{3}{4}+\frac{3}{4}\times\frac{1}{4}\times\frac{3}{4}+\frac{3}{4}\times\frac{3}{4}\times\frac{3}{4}=\frac{45}{64}=0.7031$$

所以，mc3(x)的正确率有可能低于 71%。

7-13 集合随机元素算法。

设 I={1, 2, …, n}，$S\subseteq I$ 是 I 的一个子集。mc(x)是一个偏假 p 正确蒙特卡罗算法。该算法用于判定所给的整数 $1\leqslant x\leqslant n$ 是否为集合 S 中的整数，即 $x\in S$。设 q=1−p。由偏假算法的定义可知，对任意 $x\in S$ 有 Prob{mc(x)=true}=1。当 $x\in S$ 时，Prob{mc(x)=true}$\leqslant q$。考虑下面的产生 S 中随机元素的算法 GenRand 如下：

```
bool RepeatMC(int x, int k) {
```

```
    int  i = 0;
    bool  ans = true;
    while(ans && (i<k)) {
       i++;
       ans = mc(x);
    }
    return ans;
}

int GenRand(int n, int k) {
    RandomNumber  rnd;
    int  x = rnd.Random(n)+1;
    while(!RepeatMC(x, k))
       x = rnd.Random(n)+1;
    return x;
}
```

假设由语句“x=rnd.Random(n)+1;”产生的整数 $x \in S$ 的概率为 r，证明算法 GenRand 返回的整数不在 S 中的概率最多为

$$\frac{1}{1+\frac{r}{1-r}q^{-k}}$$

分析与解答：算法 GenRand 返回的整数 x 是第 1 次由语句 x=rnd.Random(n)+1 产生的整数，且 x 不在 S 中的概率为 $(1-r)q^k$。

算法 GenRand 返回的整数 x 是由语句“x=rnd.Random(n)+1;”第 2 次产生的整数，且 x 不在 S 中的概率为

$$(1-r)(1-q^k)(1-r)q^k=(1-r)^2(1-q^k)q^k$$

……

由此可知，算法 GenRand 返回的整数不在 S 中的概率为

$$\begin{aligned}
&(1-r)q^k+(1-r)^2(1-q^k)q^k+(1-r)^3(1-q^k)^2q^k+\cdots \\
&=(1-r)q^k\sum_{i=0}^{\infty}(1-r)(1-q^k)^i \\
&=\frac{(1-r)q^k}{1-(1-r)(1-q^k)} \\
&=\frac{1}{\frac{1}{(1-r)q^k}-\left(\frac{1}{q^k}-1\right)} \\
&=\frac{1}{1+\left(\frac{1}{1-r}-1\right)q^{-k}} \\
&=\frac{1}{1+\frac{r}{1-r}q^{-k}}
\end{aligned}$$

7-14 由蒙特卡罗算法构造拉斯维加斯算法。

设算法 A 和 B 是解同一判定问题的两个有效的蒙特卡罗算法。算法 A 是 p 正确偏真算法，算法 B 是 q 正确偏假算法。试利用这两个算法设计一个解同一问题的拉斯维加斯算法，并使所得到的算法对任何实例的成功率尽可能高。

分析与解答：

```
bool Las(ST x) {
  while(true) {
    if(A(x))
      return true;
    if(!B(x))
      return false;
  }
}
```

7-15 产生素数算法。

考虑下面的无限循环算法：

```
void PrintPrimes(void) {
  cout << '2' << endl;
  cout << '3' << endl;
  int  n = 5;
  while(true) {
    int  m = floor(log(double(n)));
    if(PrimeMC(n, m))
      cout << n << endl;
    n = n+2;
  }
}
```

每个素数都会被上述算法输出。但是除了所有素数，算法可能偶尔错误地输出某些合数。说明上述情况不太可能发生。或更精确，证明上述算法错误地输出一个合数的概率小于 1%。

分析与解答：

$m=\log n$。

PrimeMC(n, m)发生错误的概率小于 $\left(\frac{1}{4}\right)^m=\frac{1}{2^{2\log n}}=\frac{1}{n^2}$。

7-16 矩阵方程问题。

给定三个 $n\times n$ 矩阵 $\boldsymbol{A}$、$\boldsymbol{B}$ 和 $\boldsymbol{C}$，下面的偏假 1/2 正确的蒙特卡罗算法用于判定 $\boldsymbol{AB}=\boldsymbol{C}$。

```
bool Product(int **A, int **B, int **C, int n) {          // 判定 AB=C 的蒙特卡罗算法
  RandomNumber rnd;
  int  *x = new int [n+1];
  int  *y = new int [n+1];
  int  *z = new int [n+1];
  for (int i=1; i <= n; i++) {
    x[i] = rnd.Random(2);
    if(x[i] == 0)
      x[i] = -1;
  }
```

```
    Mult(B, x, y, n);
    Mult(A, y, z,n);
    Mult(C, x, y,  n);
    for (int i=1; i<=  n; i++)
      if(y[i] != z[i])
        return false;
    return true;
}
```

算法所需的计算时间为 $O(n^2)$。显然当 $AB=C$ 时，算法 Product(A, B, C, n)返回 true。试证明当 $AB\neq C$ 时，算法返回值为 false 的概率至少为 1/2（提示：考虑矩阵 $AB-C$ 并证明当 $AB\neq C$ 时，将该矩阵各行相加或相减最终得到的行向量至少有一半是非零向量）。

分析与解答：

设

$$X=\{x\in R^n \mid x_i=\pm 1,\ 1\leqslant i\leqslant n\}$$

$$Y=\{x\in X \mid (AB-C)x\neq 0\}$$

$$Z=\{x\in X \mid (AB-C)x=0\}$$

当 $AB\neq C$ 时，设 $AB-C$ 的第 i 列非 0，即$(AB-C)e_i\neq 0$。

对于任意 $z\in Z$，取 $y\in R^n$ 满足：

$$y_j=z_j\ (1\leqslant j\leqslant n,\ j\neq i)$$

$$y_i=-z_i$$

则易知 $y=z-2z_ie_i$。由此可知

$$(AB-C)y=(AB-C)(z-2z_ie_i)=-2(AB-C)e_i\neq 0$$

可见，如此构造出的 $y\in Y$。

由不同的 z 构造出的 y 也不同。因此，$|Z|\leqslant|Y|$。

由此可知，当 $AB\neq C$ 时，算法 Product(A, B, C, n)返回值为 false 的概率至少为 1/2。

算法实现题 7

7-1　模平方根问题。

问题描述：设 p 是奇素数，$1\leqslant x\leqslant p-1$，如果存在一个整数 y（$1\leqslant y\leqslant p-1$），使得 $x\equiv y^2 \pmod p$，则称 y 是 x 的模 p 平方根。例如，63 是 55 的模 103 平方根。试设计一个求整数 x 的模 p 平方根的拉斯维加斯算法。算法的计算时间应为 $\log p$ 的多项式。

算法设计：设计一个拉斯维加斯算法，对于给定的奇素数 p 和整数 x，计算 x 的模 p 平方根。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 2 个正整数 p 和 x。

结果输出：将计算的 x 的模 p 平方根输出到文件 output.txt。当不存在 x 的模 p 平方根时，输出 0。

输入文件示例	输出文件示例
input.txt	output.txt
103 55	63

分析与解答：求整数 x 的模 p 平方根的 Tonelli 算法是一个拉斯维加斯算法，描述如下。

```
Tonelli(x, p) {
   (1) 随机选取 g;
   (2) 设 p-1=2st, t 为奇数;
   (3) e=0;
   (4) for(i=2; i <= s; i++)
         if((xg-e)(p-1)/2i != 1)
            e = e+2i;
   (5) h=xge;
   (6) b=ge/2h(t+1)/2;
   (7) return b;
}
```

算法实现如下。

```
uint64 sqrt(uint64 a, uint64 q) {
   RandomNumber rnd;
   uint64  b=1, d, g, x, y, h, k, s=0, t=q-1;
   if ((q%2)==0)
      return 0;
   while(t%2 == 0) {
      t /= 2;
      s++;
   }
   g = rnd.Random(q-1)+1;
   exeuclid(g, q, d, x, y);
   if(x < 0)
      x += q;
   a %= q;
   for(uint64 i=2, j=2, e=0; i <= s; i++, j*=2)
      if(del(a, q, e, x, j) != 1)
         e += j;
   for(i=1,h=a; i <= e; i++) {
      h*=x;
      h%=q;
   }
   for(i=1, e/=2; i <= e; i++) {
      b*=g;
      b%=q;
   }
   for(i=1, k=(t+1)/2; i <= k; i++) {
      b*=h;
      b%=q;
   }
   return b;
}
```

其中，uint64 是 64 位整型数，uint32 是 32 位整型数。

```
#define  uint64    __int64
```

```
#define  uint32    unsigned int
```

Exeuclid 是扩展的 euclid 算法，对于整数 a 和 b，计算出 d、x 和 y 使

$$d=\gcd(a, b)=ax+by$$

```
void exeuclid(uint64 a,uint64 b,uint64 &d, uint64 &x, uint64 &y) {
   uint64  d1, x1, y1;
   if(b == 0) {
      d = a;
      x = 1;
      y = 0;
      return;
   }
   exeuclid(b, a%b, d1, x1, y1);
   d = d1;
   x = y1;
   y = x1a/b*y1;
}
```

算法的计算时间为 $O(\log b)$。

del()函数用于计算$(ag^{-e})^{(q-1)/2^j}$。

```
uint64 del(uint64 a, uint64 q, uint64 e, uint64 x, uint64 j) {
   uint64  i, y=1, k=(q-1)/j/2;
   for(i=1; i <= e; i++) {
      a*=x;
      a%=q;
   }
   for(i=1; i <= k; i++) {
      y*=a;
      y%=q;
   }
   return y;
}
```

算法返回正确解的概率为 1/2，所需计算时间为 $O(\log^4 q)$。

可以通过多次调用，提高算法返回正确解的概率。

```
uint64 sqrtLV(uint64 a, uint64 q) {
   RandomNumber rnd;
   uint64  k = rnd.Random(100)+1;
   for(uint64 i=1; i <= k; i++) {
      uint64  r = sqrt(a, q);
      if(r*r%q == a)
         return r;
   }
   return 0;
}
```

7-2　素数测试问题。

问题描述：试设计一个素数测试的偏真蒙特卡罗算法，对于测试的整数 n，所述算法是

一个关于 $\log n$ 的多项式时间算法。结合教材中素数测试的偏假蒙特卡罗算法，设计一个素数测试的拉斯维加斯算法（见算法分析题 7-14）。

算法设计：设计一个拉斯维加斯算法，对于给定的正整数，判定其是否为素数。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 1 个正整数 p。

结果输出：将计算结果输出到文件 output.txt。若正整数 p 是素数，则输出“YES”，否则输出“NO”。

输入文件示例	输出文件示例
input.txt	output.txt
103	YES

分析与解答：用 Goldwasser-Kilian 方法或 Adleman-Huang 方法。见如下参考文献：

[1] Shafi Goldwasser, Joe Kilian. Almost all primes can be quickly certified. Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing (11 West 42nd St., New York), Association for Computing Machinery, May 1986, pp. 316-329.

[2] Leonard M.Adleman, Ming-Deh A.Huang, Primality Testing and Abelian Varieties Over Finite Fields. Lecture Notes in Mathematics, vol. 1512, Springer-Verlag, 1992.

7-3 集合相等问题。

问题描述：给定两个集合 S 和 T，试设计一个判定 S 和 T 是否相等的蒙特卡罗算法。

算法设计：设计一个拉斯维加斯算法，对于给定的集合 S 和 T，判定其是否相等。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 1 个正整数 n，表示集合的大小。接下来的 2 行，每行有 n 个正整数，分别表示集合 S 和 T 中的元素。

结果输出：将计算结果输出到文件 output.txt。若集合 S 和 T 相等则输出“YES”，否则输出“NO”。

输入文件示例	输出文件示例
input.txt	output.txt
3	YES
2 3 7	
7 2 3	

分析与解答：类似主教材中的主元素问题。

7-4 逆矩阵问题。

问题描述：给定两个 $n \times n$ 矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$，试设计一个判定 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 是否互逆的蒙特卡罗算法（算法的计算时间应为 $O(n^2)$）。

算法设计：设计一个蒙特卡罗算法，对于给定的矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$，判定其是否互逆。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 1 个正整数 n，表示矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 为 $n \times n$ 矩阵。接下来的 $2n$ 行，每行有 n 个实数，分别表示矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 中的元素。

结果输出：将计算结果输出到文件 output.txt。若矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 互逆，则输出“YES”，否则输出“NO”。

输入文件示例	输出文件示例
input.txt	output.txt
3	YES
1 2 3	
2 2 3	

3 3 3
-1 1 0
1 -2 1
0 1 -0.666667

分析与解答：

```
bool verse(double **A, double **B, int n) {                    // 判定AB=I的蒙特卡罗算法
   RandomNumber rnd;
   double *x=new double [n+1];
   double *y=new double [n+1];
   double *z=new double [n+1];
   for(int i=1; i <= n; i++) {
      x[i] = rnd.Random(2);
      if(x[i] == 0.0)
         x[i] = -1.0;
   }
   Mult(B, x, y, n);
   Mult(A, y, z, n);
   for(i=1; i <= n; i++)
      if (fabs(x[i]-z[i]) > 1e-4)
         return false;
   return true;
}
```

7-5 多项式乘积问题。

问题描述：给定阶数分别为 n、m 和 $n+m$ 的多项式 $p(x)$、q(x)和 $r(x)$。试设计一个判定 $p(x)q(x)=r(x)$的偏假 1/2 正确的蒙特卡罗算法，并要求算法的计算时间为 $O(n+m)$。

算法设计：设计一个蒙特卡罗算法，对于给定多项式 $p(x)$、$q(x)$和 $r(x)$，判定 $p(x)q(x)=r(x)$是否成立。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 3 个正整数 n、m、l，分别表示多项式 $p(x)$、$q(x)$和 $r(x)$的阶数。接下来的 3 行，每行分别有 n、m、l 个实数，分别表示多项式 $p(x)$、$q(x)$和 $r(x)$的系数。

结果输出：将计算结果输出到文件 output.txt。若 $p(x)q(x)=r(x)$成立，则输出“YES”，否则输出“NO”。

输入文件示例	输出文件示例
input.txt	output.txt
2 1 3	YES
1 2 3	
2 2	
2 6 10 6	

分析与解答：多项式的阶为 $k=\max\{mn, l\}$。随机选取 $k+1$ 个实数，测试等式是否成立。

7-6 皇后控制问题。

问题描述：在 $n\times n$ 个方格组成的棋盘上的任一方格中放置一个皇后，该皇后可以控制其所在的行、列及对角线上的所有方格。对于给定的自然数 n，在 $n\times n$ 个方格组成的棋盘上最少要放置多少个皇后才能控制棋盘上的所有方格，且放置的皇后互不攻击？

算法设计：设计一个拉斯维加斯算法，对于给定的自然数 n（$1 \leqslant n \leqslant 100$）计算在 $n \times n$ 个方格组成的棋盘上最少要放置多少个皇后才能控制棋盘上的所有方格，且放置的皇后互不攻击。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 1 个正整数 n。

结果输出：将计算的最少皇后数及最佳放置方案输出到文件 output.txt。文件的第 1 行是最少皇后数；接下来的 1 行是皇后的最佳放置方案。

输入文件示例	输出文件示例
input.txt	output.txt
8	5
0 3 6 0 0 2 5 8	

分析与解答：与主教材中 n 后问题的拉斯维加斯算法类似。具体算法描述如下。

类 Queen 的私有成员 n 表示皇后个数，数组 x 存储 n 后问题的解。

```
class Queen {
   friend bool nQueen(int);
private:
   bool Place(int k);
   int QLV(int m, int stopVegas);
   bool QueensLV(int stopVegas);
   bool ctrl(int n);
   int placed(int k);
   bool backtrack (int t);
   int  n, *x, *y, **yy;
};
```

Place(k)函数用于测试将皇后 k 置于第 $x[k]$列的合法性。

```
bool Queen::Place(int k) {
   if(x[k] > 0)
      for(int j=1; j <= k-1; j++)
         if(x[j]>0 && (abs(k-j) == abs(x[j]-x[k]) || x[j] == x[k]))
            return false;
   return true;
}
```

ctrl()函数用于测试皇后是否已控制棋盘。

```
bool Queen::ctrl(int nn) {
   int  i, j, t1, t2, cont = 0, xmin = 0;
   for(i=1; i <= nn; i++)
      for(j=1; j <= nn; j++)
         yy[i][j]=0;
   for (i=1; i <=nn; i++) {
      if (x[i]>0) {
         xmin++;
         for (j=1; j <=nn; j++) {
            yy[i][j]=1;
            yy[j][x[i]]=1;
         }
         for(t1=i, t2=x[i]; t1 >= 1 && t2 >= 1; t1--, t2--)
```

```
            yy[t1][t2] = 1;
         for(t1=i, t2=x[i]; t1 <= nn && t2 <= nn; t1++, t2++)
            yy[t1][t2] = 1;
         for(t1=i, t2=x[i]; t1 >= 1 && t2 <= nn; t1--, t2++)
            yy[t1][t2] = 1;
         for(t1=i, t2=x[i]; t1 <= nn && t2 >= 1; t1++, t2--)
            yy[t1][t2] = 1;
      }
   }
   for (i=1; i <= nn; i++)
      for (j=1; j <= nn; j++)
         cont += yy[i][j];
   return (cont == nn*nn);
}
```

QueensLV(stopVegas)函数实现在棋盘上随机放置若干皇后的拉斯维加斯算法，其中1≤stopVegas≤n表示随机放置的皇后数。

```
bool Queen::QueensLV(int stopVegas) {
   int  m = stopVegas, count = 0, cont = 10000;
   while(true){
      int  ret = QLV(m, stopVegas);
      if(ret > 0) {
         m=ret-1;
         count=0;
      }
      else
         count++;
      if(count > cont) {
         while(QLV(m+1, stopVegas) == 0);
         break;
      }
   }
   return true;
}
int Queen::QLV(int m,int stopVegas) {
   randomNumber rnd;
   while(true){
      int  k= 1;
      while(k <= stopVegas) {
         for(int i=0,count=0; i <= n; i++) {
            x[k] = i;
            if(Place(k))
               y[count++] = i;
         }
         x[k++] = y[rnd.random(count)];
      }
      int  pla = placed(stopVegas);
      if(pla <= m)
         return pla;
```

```
    else
      return 0;
  }
}
```

与回溯法相结合的解 n 后控制问题的拉斯维加斯算法描述如下。

```
bool nQueen(int n) {
  Queen  X;
  X.n = n;
  int  *p = new int [n+1];
  int  *q = new int [n+1];
  int  **r;
  Make2DArray(r, n+1, n+1);
  for(int i=0; i<=n; i++)
    p[i] = 0;
  X.x = p;
  X.y = q;
  X.yy = r;
  int  stop = 3;
  if(n > 15)
    stop = n15;
  bool  found = false;
  while(!X.QueensLV(stop))  ;
  if(X.backtrack(stop+1)) {
    cout<<X.placed(n)<<endl;
    for(i=1; i <= n; i++)
      cout<<p[i]<<" ";
    cout << endl;
    found = true;
  }
  delete [] p;
  delete [] q;
  Delete2DArray(r, n+1);
  return found;
}
```

算法的回溯搜索部分与解 n 后问题的回溯法是类似的，只要找到一个解就返回。

```
bool Queen::backtrack(int t) {
  if(t > n) {
    if(ctrl(n))
      return true;
    else
      return false;
  }
  for(int i=0; i <= n; i++) {
    x[t]=i;
    if(Place(t) && backtrack(t+1))
      return true;
  }
```

```
    return false;
}
```

placed()函数计算已放置的皇后数。

```
int Queen::placed(int k) {
  for(int j=1, num=0; j <= k; j++)
    if(x[j] > 0)
      num++;
  return num;
}
```

7-7 3-SAT 问题。

问题描述：SAT 的一个实例是 k 个布尔变量 $x_1, x_2, \cdots, x_k$ 的 m 个布尔表达式 $A_1, A_2, \cdots, A_m$。若存在各布尔变量 x_i（$1 \leqslant i \leqslant k$）的 0，1 赋值，使每个布尔表达式 A_i（$1 \leqslant i \leqslant m$）都取值为 1，则称布尔表达式 $A_1 A_2 \cdots A_m$ 是可满足的。

（1）合取范式的可满足性问题 CNF-SAT

如果一个布尔表达式是一些因子和之积，则称为合取范式（Conjunctive Normal Form，CNF）。这里的因子是变量 x 或 $\bar{x}$。例如，$(x_1 + x_2)(x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3)$ 就是一个合取范式，而 $x_1x_2+x_3$ 就不是合取范式。

（2）k-SAT

如果一个布尔合取范式的每个乘积项最多是 k 个因子的析取式，就称为 k 元合取范式，简记为 k-CNF。k-SAT 问题是判定一个 k-CNF 是否可满足。特别地，当 k=3 时，3-SAT 问题在 NP 完全问题树中具有重要地位。

（3）MAX-SAT

给定 k 个布尔变量 $x_1, x_2, \cdots, x_k$ 的 m 个布尔表达式 $A_1, A_2, \cdots, A_m$，求各布尔变量 x_i（$1 \leqslant i \leqslant k$）的 0、1 赋值，使尽可能多的布尔表达式 A_i 取值为 1。

（4）Weighted-MAX-SAT

给定 k 个布尔变量 $x_1, x_2, \cdots, x_k$ 的 m 个布尔表达式 $A_1, A_2, \cdots, A_m$，每个布尔表达式 A_i 都有权值 w_i，求各布尔变量 x_i（$1 \leqslant i \leqslant k$）的 0、1 赋值，使取值 1 的布尔表达式权值之和达到最大。

算法设计：对于给定的带权 3-CNF，设计一个蒙特卡罗算法，使其权值之和尽可能大。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 2 个正整数 k 和 m，分别表示变量数和布尔表达式数。接下来的 m 行中，每行有 5 个整数 w、i、j、k、0，表示相应表达式的权值为 w，表达式含的变量下标分别为 i、j、k，行末以 0 结尾。下标为负数时，表示相应的变量为取反变量。

结果输出：将计算的最大权值输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
5 3	26
9 3 1 4 0	
9 1 −5 3 0	
8 2 −5 1 0	

分析与解答：与主教材中 n 后问题的拉斯维加斯算法类似。随机产生布尔变量的真值赋值。

7-8 战车问题。

问题描述：在 $n \times n$ 格的棋盘上放置彼此不受攻击的车。按照国际象棋规则，车可以攻击与之处在同一行或同一列上的车。在棋盘上的若干个格中设置了堡垒，战车无法穿越堡垒攻击别的战车。对于给定的设置了堡垒的 $n \times n$ 格棋盘，设法放置尽可能多的彼此不受攻击的车。

算法设计：对于给定的设置了堡垒的 $n \times n$ 格棋盘，设计一个随机化算法，在棋盘上放置尽可能多的彼此不受攻击的车。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 1 个正整数 n。接下来的 n 行中，每行有 1 个由字符“.”和“X”组成的长度为 n 的字符串。

结果输出：将计算的在棋盘上可以放置的彼此不受攻击的战车数输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
4	6
....	
..X.	
.X..	
....	

分析与解答：与主教材中 n 后问题的拉斯维加斯算法类似。在 $n \times n$ 格的棋盘上随机放置彼此不受攻击的车。具体算法实现如下。

init()函数用于初始化。

```
void init(int n) {
   int  i, j, k, x, y;
   state = new int[n*n+2];
   Make2DArray(link, n*n+1, 2*n+1);
   state[0] = -1;
   state[n*n+1] = -1;
   for(int no=1; no <= n*n; no++) {
      i = (no-1)/n;
      j = (no-1)%n;
      link[no][0] = 0;
      state[no] = -1;
      if(row[i][j] == '.') {
         state[no] = 0;
         k = 0;
         y = j;
         while((y<n-1) && (row[i][y+1]=='.')) {
            link[no][0]++;
            y++;
            k++;
            link[no][k] = i*n+y+1;
         }
         y = j;
         while((y>0) && (row[i][y1] == '.')) {
            link[no][0]++;
            y--;
            k++;
```

```
            link[no][k] = i*n+y+1;
        }
        x=i;
        while ((x<n-1) && (row[x+1][j]=='.')) {
            link[no][0]++;
            x++;
            k++;
            link[no][k] = x*n+j+1;
        }
        x=i;
        while ((x>0) && (row[x1][j]=='.')) {
            link[no][0]++;
            x--;
            k++;
            link[no][k] = x*n+j+1;
        }
    }
  }
}
```

实现随机算法的主函数如下。

```
int main() {
   int  n;
   randomNumber  rnd;
   fin >> n;
   for(int i=0; i < n; i++)
      fin >> row[i];
   init(n);
   int  max = 0, rept = 0, put = 0;
   while(rept < 100000) {
      rept++;
      int  count = 0;
      while(true) {
         int  x=rnd.random(n*n)+1, c = x;
         while((x <= n*n) && (state[x] != put))
            x++;
         if(state[x] != put) {
            x = c;
            while((x>0) && (state[x] != put))
               x;
         }
         if (state[x] == put) {
            count++;
            for(i=1; i <= link[x][0]; i++)
               if(state[link[x][i]] == put)
                  state[link[x][i]]++;
            state[x]++;
         }
         else
```

```
            break;
        }
        if(count > max)
            max = count;
        put++;
    }
    cout << max << endl;
    return 0;
}
```

计算机算法设计与分析习题解答（第5版）

本书是与“十二五”普通高等教育本科国家级规划教材《计算机算法设计与分析（第5版）》配套的辅助教材和国家精品课程教材，分别对主教材中的算法分析题和算法实现题给出了解答或解题思路提示。为了提高学生灵活运用算法设计策略解决实际问题的能力，本书还将主教材中的许多习题改造成算法实现题，要求学生设计出求解算法并上机实现。本书教学资料包含各章算法实现题、测试数据和答案，可在华信教育资源网免费注册下载。

本书内容丰富，理论联系实际，可作为高等学校计算机科学与技术、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学习计算机算法设计的辅助教材，也是工程技术人员和自学者的参考书。

提升学生“知识—能力—素质”	体现“基础—技术—应用”内容
把握教学“难度—深度—强度”	提供“教材—教辅—课件”支持

相关图书：《计算机算法设计与分析（第5版）》 ISBN 978-7-121-34439-8

策划编辑：章海涛
责任编辑：章海涛
封面设计：张 昱

ISBN 978-7-121-34438-1

定价：56.00 元