“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

非外借

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。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

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4章 贪心算法

算法分析题4

4-1 程序最优存储问题。

假定要把长为 $l_1, l_2, \cdots, l_n$ 的 n 个程序放在磁带 T_1 和 T_2 上，并且希望按照使最大检索时间取最小值的方式存放，即如果存放在 T_1 和 T_2 上的程序集合分别是 A 和 B，则希望所选择的 A 和 B 使得 $\max\left\{\sum_{i\in A} l_i, \sum_{i\in B} l_i\right\}$ 取最小值。

贪心算法：开始将 A 和 B 都初始化为空，然后一次考虑一个程序，如果 $\min\left\{\sum_{i\in A} l_i, \sum_{i\in B} l_i\right\}$，则将当前正在考虑的那个程序分配给 A，否则分配给 B。证明无论是按 $l_1 \leqslant l_2 \leqslant \cdots \leqslant l_n$ 还是按 $l_1 \geqslant l_2 \geqslant \cdots \geqslant l_n$ 的次序来考虑程序的，这种方法都不能产生最优解。应当采用什么策略？写出一个完整的算法并证明其正确性。

分析与解答：设变量 $x_i=1$ 表示将 l_i 存放在 T_1 上，且 T_1 的检索时间较短，则

$$\sum_{i=1}^{n} l_i x_i - \sum_{i=1}^{n} l_i(1-x_i) \leqslant 0, 2\sum_{i=1}^{n} l_i x_i \leqslant \sum_{i=1}^{n} l_i, \sum_{i=1}^{n} l_i x_i \leqslant \frac{1}{2}\sum_{i=1}^{n} l_i$$

T_1 的检索时间应取最大值，因此问题归结为

$$\max \sum_{i=1}^{n} l_i x_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n} l_i x_i \leqslant \frac{1}{2}\sum_{i=1}^{n} l_i$$

这与主教材中第5章的装载问题等价，是一个特殊的0-1背包问题。

4-2 最优装载问题的贪心算法。

将最优装载问题的贪心算法推广到2艘船的情形，贪心算法仍能产生最优解吗？

分析与解答：贪心算法不能产生最优解，见主教材第5章的装载问题。

4-3 Fibonacci序列的哈夫曼编码。

字符a～h出现的频率恰好是前8个Fibonacci数，它们的哈夫曼编码是什么？将结果推广到 n 个字符的频率恰好是前 n 个Fibonacci数的情形。

分析与解答：频率恰好是前8个Fibonacci数的哈夫曼编码树如图4-1所示。

在一般情况下，n 个字符的频率恰好是前 n 个Fibonacci数，则相应的哈夫曼编码树深度为 $n-1$，第1个字符的编码长度为 $n-1$。自底向上第 i 个圆结点中的数为 $\sum_{k=0}^{i} f_k$。用数学归纳法容易证明 $\sum_{k=0}^{i} f_k < f_{i+2}$。该性质保证了频率恰好是前 n 个Fibonacci数的哈夫曼编码树具

有所述形状和性质。

4-4　最优前缀码的编码序列。

设 $C=\{0, 1, \cdots, n-1\}$ 是 n 个字符的集合。证明关于 C 的任何最优前缀码可以表示为长度为 $2n-1+n\lceil \log n \rceil$ 位的编码序列（提示：用 $2n-1$ 位描述树结构）。

分析与解答：任何最优前缀码所相应的编码二叉树是一棵完全二叉树，有 n 个叶结点和 $n-1$ 个内结点。用 1 位表示 1 个结点的类型，1 表示内结点，0 表示叶结点，共需 $2n-1$ 位。对编码树的前序遍历可以唯一表示该编码树结构。

例如，当 $n=4$ 时，如图 4-2 所示编码树结构可以唯一表示为 1101000。

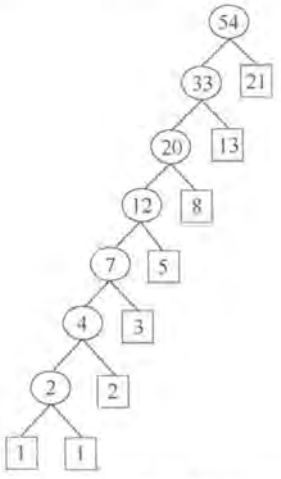

图 4-1　哈夫曼编码树

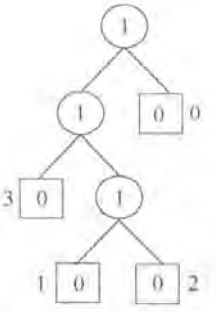

图 4-2　编码树结构

在每个叶结点后，即每个 0 后面紧跟 $\lceil \log n \rceil$ 位表示该叶结点处的数字，即可完整表示整棵编码树。例如，图 4-2 中的编码树可表示为 110111001010000。由此可知，在一般情况下，最优前缀码可以表示长度为 $2n-1+n\lceil \log n \rceil$ 位的编码序列。

算法实现题 4

4-1　会场安排问题。

问题描述：假设要在足够多的会场里安排一批活动，并希望使用尽可能少的会场。设计一个有效的贪心算法进行安排。（这个问题实际上是著名的图着色问题。若将每个活动作为图的一个顶点，不相容活动间用边相连。使相邻顶点着有不同颜色的最小着色数，相当于要找的最小会场数。）

算法设计：对于给定的 k 个待安排的活动，计算使用最少会场的时间表。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 1 个正整数 k，表示有 k 个待安排的活动。接下来的 k 行中，每行有 2 个正整数，分别表示 k 个待安排的活动的开始时间和结束时间。时间以 0 点开始的分钟计。

结果输出：将计算的最少会场数输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt

5	3
1 23	
12 28	
25 35	
27 80	
36 50	

分析与解答：具体算法描述如下。

```
int greedy(vector<point> x) {
   int  sum=0, curr=0, n=x.size();
   sort(x.begin (),x.end ());
   for(int i=0; i < n; i++) {
      if(x[i].leftend())
         curr++;
      else
         curr--;
      if((i == n-1 || x[i] < x[i+1]) && curr > sum)              // 处理x[i]=x[i+1]的情况
         sum=curr;
   }
   return sum;
}
```

4-2 最优合并问题。

问题描述：给定 k 个排好序的序列 $s_1, s_2, \cdots, s_k$，用 2 路合并算法将这 k 个序列合并成一个序列。假设采用的 2 路合并算法合并 2 个长度分别为 m 和 n 的序列需要 $m+n-1$ 次比较。试设计一个算法确定合并这个序列的最优合并顺序，使所需的总比较次数最少。

为了进行比较，还需要确定合并这个序列的最差合并顺序，使所需的总比较次数最多。

算法设计：对于给定的 k 个待合并序列，计算最多比较次数和最少比较次数合并方案。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 1 个正整数 k，表示有 k 个待合并序列。接下来的 1 行中，有 k 个正整数，表示 k 个待合并序列的长度。

结果输出：将计算的最多比较次数和最少比较次数输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
4	78 52
5 12 11 2	

分析与解答：见主教材中的 Huffman 算法。

4-3 磁带最优存储问题。

问题描述：设有 n 个程序{1, 2, …, n}要存放在长度为 L 的磁带上。程序 i 存放在磁带上的长度是 l_i（$1 \leqslant i \leqslant n$）。这 n 个程序的读取概率分别是 $p_1, p_2, \cdots, p_n$，且 $\sum_{i=1}^{n} p_i$。如果将这 n 个程序按 $i_1, i_2, \cdots, i_n$ 的次序存放，则读取程序 i_r 所需的时间 $t_r = c\sum_{k=1}^{r} p_{ik} l_{ik}$。这 n 个程序的平均读取时间为 $\sum_{r=1}^{n} t_r$。

磁带最优存储问题要求确定这 n 个程序在磁带上的一个存储次序，使平均读取时间达到

最小。试设计一个解此问题的算法，并分析算法的正确性和计算复杂性。

算法设计：对于给定的 n 个程序存放在磁带上的长度和读取概率，计算 n 个程序的最优存储方案。

数据输入：由文件 input.txt 给出输入数据。第 1 行是正整数 n，表示文件个数。接下来的 n 行中，每行有 2 个正整数 a 和 b，分别表示程序存放在磁带上的长度和读取概率。实际上第 k 个程序的读取概率为 $a_k \Big/ \sum_{i=1}^{n} a_i$。对所有输入均假定 c=1。

结果输出：将计算的最小平均读取时间输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
5	85.6193
71 872	
46 452	
9 265	
73 120	
35 85	

分析与解答：贪心策略，即最短平均读取时间程序优先。

4-4 磁盘文件最优存储问题。

问题描述：设磁盘上有 n 个文件 $f_1, f_2, \cdots, f_n$，每个文件占用磁盘上的 1 个磁道。这 n 个文件的检索概率分别是 $p_1, p_2, \cdots, p_n$ 且 $\sum_{i=1}^{n} p_i = 1$。磁头从当前磁道移到被检信息磁道所需的时间可用这两个磁道之间的径向距离来度量。如果文件 f_i 存放在第 i（$1 \leqslant i \leqslant n$）道上，则检索这 n 个文件的期望时间是 $\sum_{1 \leqslant i \leqslant j \leqslant n} p_i p_j d(i,j)$。式中，$d(i,j)$ 是第 i 道与第 j 道之间的径向距离 $|i-j|$。

磁盘文件的最优存储问题要求确定这 n 个文件在磁盘上的存储位置，使期望检索时间达到最小。试设计一个解此问题的算法，并分析算法的正确性与计算复杂性。

算法设计：对于给定的文件检索概率，计算磁盘文件的最优存储方案。

数据输入：由文件 input.txt 给出输入数据。第 1 行是正整数 n，表示文件个数。第 2 行有 n 个正整数 a_i，表示文件的检索概率。实际上第 k 个文件的检索概率应为 $a_k \Big/ \sum_{i=1}^{n} a_i$。

结果输出：将计算的最小期望检索时间输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
5	0.547396
33 55 22 11 9	

分析与解答：将 n 个文件按其概率排序。设排序后有 $p_1 \geqslant p_2 \geqslant \cdots \geqslant p_n$。

贪心策略：f_1 占中心磁道，f_2 和 f_3 分居 f_1 的两侧，f_4 在 f_2 的左侧，f_5 在 f_3 的右侧，……

具体算法实现如下。

```
double greedy(vector<int> p) {
  int  n = p.size ();
  vector<int> x(n, 0);
  sort(p.begin(), p.end());
```

```
    int  k = (n-1)/2;
    x[k] = p[n-1];
    for(int i=k+1; i < n; i++)
       x[i] = p[n-2*(i-k)];
    for(i=k-1; i >= 0; i--)
       x[i] = p[n-2*(k-i)-1];
    double  m = 0, t = 0;
    for (i=0; i < n; i++) {
       m += p[i];
       for(int j=i+1; j<n; j++)
          t += x[i]*x[j]*(j-i);
    }
    t = t/m/m;
    return t;
}
```

4-5　程序存储问题。

问题描述：设有 n 个程序$\{1, 2, \cdots, n\}$要存放在长度为 L 的磁带上。程序 i 存放在磁带上的长度是 l_i（$1 \leqslant i \leqslant n$）。程序存储问题要求确定这 n 个程序在磁带上的一个存储方案，使得能够在磁带上存储尽可能多的程序。

算法设计：对于给定的 n 个程序存放在磁带上的长度，计算磁带上最多可以存储的程序数。

数据输入：由文件 input.txt 给出输入数据。第 1 行是 2 个正整数，分别表示文件个数 n 和磁带的长度 L。接下来的 1 行中，有 n 个正整数，表示程序存放在磁带上的长度。

结果输出：将计算的最多可以存储的程序数输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
6　50	5
2 3 13 8 80 20	

分析与解答：贪心策略，即最短程序优先。

```
int greedy(vector<int> x,int m) {
   int  i=0, sum=0, n=x.size();
   sort(x.begin(), x.end());
   while(i < n){
      sum += x[i];
      if(sum <= m)
         i++;
      else
         return i;
   }
   return n;
}
```

4-6　最优服务次序问题。

问题描述：设有 n 个顾客同时等待一项服务，顾客 i 需要的服务时间为 t_i（$1 \leqslant i \leqslant n$）。应如何安排 n 个顾客的服务次序才能使平均等待时间达到最小？平均等待时间是 n 个顾客等待服务时间的总和除以 n。

算法设计：对于给定的 n 个顾客需要的服务时间，计算最优服务次序。

数据输入：由文件 input.txt 给出输入数据。第 1 行是正整数 n，表示有 n 个顾客。接下来的 1 行中，有 n 个正整数，表示 n 个顾客需要的服务时间。

结果输出：将计算的最小平均等待时间输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
10	532.00
56 12 1 99 1000 234 33 55 99 812	

分析与解答：贪心策略，即最短服务时间优先。

```
double greedy(vector<int> x) {
   int  n = x.size();
   sort(x.begin(), x.end());
   for(int i=1; i < n ;i++)
     x[i] += x[i-1];
   double  t = 0;
   for(i=0; i < n; i++)
     t += x[i];
   t /= n;
   return t;
}
```

4-7　多处最优服务次序问题。

问题描述：设有 n 个顾客同时等待一项服务。顾客 i 需要的服务时间为 t_i（$1 \leqslant i \leqslant n$），共有 s 处可以提供此项服务。应如何安排 n 个顾客的服务次序，才能使平均等待时间达到最小？平均等待时间是 n 个顾客等待服务时间的总和除以 n。

算法设计：对于给定的 n 个顾客需要的服务时间和 s 的值，计算最优服务次序。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 2 个正整数 n 和 s，表示有 n 个顾客且有 s 处可以提供顾客需要的服务。接下来的 1 行中有 n 个正整数，表示 n 个顾客需要的服务时间。

结果输出：将计算的最小平均等待时间输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
10　2	336
56 12 1 99 1000 234 33 55 99 812	

分析与解答：贪心策略，即最短服务时间优先。

```
double greedy(vector<int> x, int s) {
   vector<int> st(s+1, 0);
   vector<int> su(s+1, 0);
   int  n = x.size();
   sort(x.begin(), x.end());
   int  i = 0, j = 0;
   while(i < n) {
     st[j] += x[i];
     su[j] += st[j];
     i++;
```

```
        j++;
        if(j == s)
          j = 0;
    }
    double  t = 0;
    for(i=0; i<  s; i++)
      t += su[i];
    t /= n;
    return t;
}
```

4-8　d 森林问题。

问题描述：设 T 是一棵带权树，树的每条边带一个正权，S 是 T 的顶点集，T/S 是从树 T 中将 S 中顶点删去后得到的森林。如果 T/S 中所有树的从根到叶的路长都不超过 d，则称 T/S 是一个 d 森林。

① 设计一个算法求 T 的最小顶点集 S，使 T/S 是 d 森林（提示：从叶向根移动）。

② 分析算法的正确性和计算复杂性。

③ 设 T 中有 n 个顶点，则算法的计算时间复杂性应为 $O(n)$。

算法设计：对于给定的带权树，计算最小分离集 S。

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，表示给定的带权树有 n 个顶点，编号为 1, 2, …, n。编号为 1 的顶点是树根。接下来的 n 行中，第 i+1 行描述与 i 个顶点相关联的边的信息。每行的第 1 个正整数 k 表示与该顶点相关联的边数。其后 $2k$ 个数中，每 2 个数表示 1 条边。第 1 个数是与该顶点相关联的另一个顶点的编号，第 2 个数是边权值。k=0，表示相应的结点是叶结点。文件的最后一行是正整数 d，表示森林中所有树的从根到叶的路长都不超过 d。

结果输出：将计算的最小分离集 S 的顶点数输出到文件 output.txt。如果无法得到所要求的 d 森林则输出“No Solution!”。

输入文件示例	输出文件示例
input.txt	output.txt
4	1
2 2 3 3 1	
1 4 2	
0	
0	
4	

分析与解答：父亲数组 parent 表示树，leaf 存储叶结点编号，readin()函数读入初始数据。

```
void readin() {
   fin >> n;
   for(int i=1; i <= n; i++) {
     fin >> deg[i];
     for(int j=0; j < deg[i]; j++) {
       fin >> p >> len;
       parent[p] = i;
       parlen[p] = len;
     }
```

```
    if(deg[i] == 0)
      leaf[++leaf[0]] = i;
  }
  fin >> p;
}
```

从叶结点向根结点移动，从根结点到叶结点的路长超过 d 时，将该子树分离。

```
int count() {
  for (int i=1, total=0; i <= leaf[0]; i++) {
    if(leaf[i] != 1) {                                    // 非根结点
      int  plen = parlen[leaf[i]], par = parent[leaf[i]];
      if(cut[par] < 1 && dist[leaf[i]]+plen > p) {
        total++;
        cut[par] = 1;
        par = parent[par];
      }
      else if(cut[par] < 1 && dist[par] < dist[leaf[i]]+plen)
        dist[par] = dist[leaf[i]] + plen;
      if(--deg[par] == 0)
        leaf[++leaf[0]] = par;
    }
    return total;
}
```

4-9 虚拟汽车加油问题。

问题描述：一辆虚拟汽车加满油后可行驶 n km。旅途中有若干加油站。设计一个有效算法，指出应在哪些加油站停靠加油，使沿途加油次数最少。并证明算法能产生一个最优解。

算法设计：对于给定的 n 和 k 个加油站位置，计算最少加油次数。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 2 个正整数 n 和 k，表示汽车加满油后可行驶 n km，且旅途中有 k 个加油站。接下来的 1 行中有 k+1 个整数，表示第 k 个加油站与第 k–1 个加油站之间的距离。第 0 个加油站表示出发地，汽车已加满油。第 k+1 个加油站表示目的地。

结果输出：将计算的最少加油次数输出到文件 output.txt。如果无法到达目的地，则输出“No Solution”。

输入文件示例	输出文件示例
input.txt	output.txt
7　7	4
1 2 3 4 5 1 6 6	

分析与解答：贪心策略：最远加油站优先。

```
int greedy(vector<int> x, int n) {
  int  sum=0, k=x.size();
  for(int j=0; j < k; j++) {
    if(x[j] > n) {
      cout << "No Solution!" << endl;
      return-1;
    }
    for(int i=0, s=0; i < k; i++) {
```

```
            s += x[i];
            if(s > n) {
               sum++;
               s = x[i];
            }
         }
      }
      return sum;
}
```

4-10 区间覆盖问题。

问题描述：设 $x_1, x_2, \cdots, x_n$ 是实直线上的 n 个点。用固定长度的闭区间覆盖这 n 个点，至少需要多少个这样的固定长度闭区间？设计解此问题的有效算法，并证明算法的正确性。

算法设计：对于给定的实直线上的 n 个点和闭区间的长度 k，计算覆盖点集的最少区间数。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 2 个正整数 n 和 k，表示有 n 个点，且固定长度闭区间的长度为 k。接下来的 1 行中有 n 个整数，表示 n 个点在实直线上的坐标（可能相同）。

结果输出：将计算的最少区间数输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
7 3 1 2 3 4 5 −2 6	3

分析与解答：贪心策略，即每次覆盖尽可能多的点。

```
int greedy(vector<int> x, int k) {
   int  sum = 1, n = x.size();
   sort(x.begin(), x.end());
   for(int i=1, temp=x[0]; i < n; i++) {
      if(x[i] temp > k) {
         sum++;
         temp = x[i];
      }
   }
   return sum;
}
```

4-11 删数问题。

问题描述：给定 n 位正整数 a，去掉其中任意 $k \leq n$ 个数字后，剩下的数字按原次序排列组成一个新的正整数。对于给定的 n 位正整数 a 和正整数 k，设计一个算法找出剩下数字组成的新数最小的删数方案。

算法设计：对于给定的正整数 a，计算删去 k 个数字后得到的最小数。

数据输入：由文件 input.txt 提供输入数据。文件的第 1 行是 1 个正整数 a。第 2 行是正整数 k。

结果输出：将计算的最小数输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt

178543 13
4

分析与解答：贪心策略，即最近下降点优先。

```
void delek() {
   int  m = a.size();
   if(k >= m) {
      a.erase();
      return;
   }
   while (k>0){
      for(int i=0; (i<a.size()-1) && (a[i] <= a[i+1]); i++);
         a.erase(i, 1);
         k--;
   }
   while(a.size()>  1 && a[0] == '0')
      a.erase(0, 1);
}
```

4-12　磁带最大利用率问题。

问题描述：设有 n 个程序$\{1, 2, \cdots, n\}$要存放在长度为 L 的磁带上，程序 i 存放在磁带上的长度是 l_i（$1 \leqslant i \leqslant n$）。程序存储问题要求确定这 n 个程序在磁带上的一个存储方案，使得能够在磁带上存储尽可能多的程序。在保证存储最多程序的前提下，还要求磁带的利用率达到最大。

算法设计：对于给定的 n 个程序存放在磁带上的长度，计算磁带上最多可以存储的程序数和占用磁带的长度。

数据输入：由文件 input.txt 给出输入数据。第 1 行是 2 个正整数，分别表示文件个数 n 和磁带的长度 L。接下来的 1 行中有 n 个正整数，表示程序存放在磁带上的长度。

结果输出：将计算的最多可以存储的程序数和占用磁带的长度及存放在磁带上的每个程序的长度输出到文件 output.txt。第 1 行输出最多可以存储的程序数和占用磁带的长度；第 2 行输出存放在磁带上的每个程序的长度。

输入文件示例	输出文件示例
input.txt	output.txt
9　50	5　49
2 3 13 8 80 20 21 22 23	2 3 13 8 23

分析与解答：贪心策略，即最短程序优先。

求得最多可以存储的程序个数 m 后，再求最大利用率。问题转化为主教材第 5 章中的装载问题，但 m 已知。对主教材第 5 章中的装载问题的解略做如下修改。

```
template<class T>
void Loading<T>::maxLoading(int i) {
   if(i > n) {
      if(xm == m) {
         for(int j=1; j <= n; j++)
            bestx[j] = x[j];
         bestw = cw;
      }
```

```
        return;
    }
    r -= w[i];
    if(cw+w[i] <= c && xm < m) {
        x[i] = 1;
        xm++;
        cw += w[i];
        maxLoading(i+1);
        cw -= w[i];
        xm--;
    }
    if(cw+r > bestw){
        x[i] = 0;
        maxLoading(i+1);
    }
    r += w[i];
}
```

4-13 非单位时间任务安排问题。

问题描述：具有截止时间和误时惩罚的任务安排问题可描述如下。

（1）给定 n 个任务的集合 $S=\{1, 2, \cdots, n\}$；

（2）完成任务 i 需要 t_i（$1 \leqslant i \leqslant n$）时间；

（3）任务 i 的截止时间 d_i（$1 \leqslant i \leqslant n$），即要求任务 i 在时间 d_i 之前结束；

（4）任务 i 的误时惩罚 w_i（$1 \leqslant i \leqslant n$），即任务 i 未在时间 d_i 之前结束，将招致 w_i 的惩罚；若按时完成，则无惩罚。

任务安排问题要求确定 S 的一个时间表（最优时间表）使得总误时惩罚达到最小。

算法设计：对于给定的 n 个任务，计算总误时惩罚最小的最优时间表。

数据输入：由文件 input.txt 给出输入数据。第 1 行是 1 个正整数 n，表示任务数。接下来的 n 行中，每行有 3 个正整数 a、b、c，表示完成相应任务需要时间 a，截止时间为 b，误时惩罚值为 c。

结果输出：将计算的总误时惩罚输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
7	110
1 4 70	6
2 2 60	
1 4 50	
1 3 40	
1 1 30	
1 4 20	
3 6 80	

分析与解答：首先将任务依其截止时间非减序排列。

设对任务 $1, 2, \cdots, i$，截止时间为 d 的最小误时惩罚为 $p(i, d)$，则 $p(i, d)$具有最优子结构性质且满足如下递归式：

$$p(i, d)=\min\{p(i-1, d)+w(i), p(i-1, \min\{d, d_i\}-t_i)\}$$

$$p(1,d)=\begin{cases}20 & t_1 \leqslant d \\ w(1) & t_1 > d\end{cases}$$

据此可设计解此问题的算法如下。

init()函数读入数据并进行初始化处理。

```
void init() {
  cin >> n;
  tsk.resize(n);
  for(int i=0; i < n; i++) {
    tsk[i].resize(3);
    cin >> tsk[i][0] >> tsk[i][1] >> tsk[i][2];
  }
  sort(tsk.begin(), tsk.end(), vless);
  d = tsk[n-1][1];
  f.resize(n, d+1);
  for(i=0; i < n; i++)
    for(int j=0; j <= d; j++)
      f[i][j] = INT_MAX;
}
```

dyna()函数进行动态规划计算。

```
void dyna() {
  for(int i=0; i <= d; i++) {
    if(tsk[0][0] <= i)
      f[0][i] = 0;
    else
      f[0][i] = tsk[0][2];
    for(i=1; i < n; i++) {
      for(int j=0; j <= d; j++) {
        f[i][j] = f[i1][j] + tsk[i][2];
        int  jj = tsk[i][1]>j ? j : tsk[i][1];
        if(jj >= tsk[i][0] && f[i][j] > f[i-1][jj-tsk[i][0]])
          f[i][j] = f[i-1][jj-tsk[i][0]];
      }
    }
  }
}
```

实现算法的主函数如下。

```
void main(){
  init();
  dyna();
  cout << f[n1][d] << endl;
}
```

算法所需的计算时间为 $O(n\log n+nd)$。其中，$d=\max\limits_{1\leqslant i\leqslant n}\{d_i\}$。

4-14 多元 Huffman 编码问题。

问题描述：在一个操场的四周摆放着 n 堆石子。现要将石子有次序地合并成一堆。规定

每次至少选 2 堆，最多选 k 堆石子合并成新的一堆，合并的费用为新的一堆的石子数。试设计一个算法，计算出将 n 堆石子合并成一堆的最大总费用和最小总费用。

算法设计：对于给定的 n 堆石子，计算合并成一堆的最大总费用和最小总费用。

数据输入：由文件 input.txt 提供输入数据。文件的第 1 行有 2 个正整数 n 和 k，表示有 n 堆石子，每次至少选 2 堆最多选 k 堆石子合并。第 2 行有 n 个数，分别表示每堆石子的个数。

结果输出：将计算的最大总费用和最小总费用输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
7 3	593 199
45 13 12 16 9 5 22	

分析与解答：不妨设 $n \bmod (k-1)=1$，若不满足，可增加若干个 0。

贪心策略：每次选最小的 k 个元素进行合并。与二元 Huffman 算法类似，可证明其满足贪心选择性质。具体算法描述如下。

```
int tuffman(int a[], int n) {
   MinHeap<int> H(1);
   H.Initialize(a, n, n);
   int  i, j, m, x, t, sum;
   m = (k - n%(k-1)) % (k-1);
   for(i=1, t=0; i <= k-m; i++) {
      H.DeleteMin(x);
      t += x;
   }
   um = t;
   n = (n-k+m) / (k-1);
   H.Insert(t);
   for (i=1; i <= n; i++) {
      for(j=1, t=0; j <= k; j++) {
         H.DeleteMin(x);
         t += x;
      }
      sum += t;
      H.Insert(t);
   }
   H.Deactivate();
   return sum;
}
```

算法所需的计算时间为 $O(n\log_k n)$。

4-15 最优分解问题。

问题描述：设 n 是一个正整数。现在要求将 n 分解为若干互不相同的自然数的和，且使这些自然数的乘积最大。

算法设计：对于给定的正整数 n，计算最优分解方案。

数据输入：由文件 input.txt 提供输入数据。文件的第 1 行是正整数 n。

结果输出：将计算的最大乘积输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
10	30

分析与解答：注意到，若 $a+b$=const，则$|a-b|$越小，ab 越大。

贪心策略：将 n 分成从 2 开始的连续自然数的和。如果最后剩下一个数，将此数在后项优先的方式下均匀地分给前面各项。

```
void dicomp() {
   k = 1;
   if(n < 3) {
      a[1] = 0;
      return;
   }
   if(n < 5) {
      a[k] = 1;
      a[++k] = n-1;
      return;
   }
   a[1] = 2;
   n -= 2;
   while(n > a[k]) {
      k++;
      a[k] = a[k-1]+1;
      n -= a[k];
   }
   if(n == a[k]) {
      a[k]++;
      n--;
   }
   for(int i=0; i < n; i++)
      a[k-i]++;
}
```

欢迎登录 免费 获取本书教学资源
http://www.hxedu.com.cn

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