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

```
k←0
while n≥1 do
   for j←1 to n do
      k←k+1
   n←n/2
return k
```

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

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

1.16 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			

1.16

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

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 $f(n)=\Theta(g(n))$.

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

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

1.18

$$n!,\quad 2^{2n},\quad n2^n,\quad (\log n)^{\log n}=n^{\log\log n},\quad n^3,\quad n\log n=\Theta(\log(n!)),$$

$$n=\Theta(\log 10^n),\quad 2^{\log\sqrt{n}},\quad 2^{\sqrt{2\log n}},\quad \log n=\Theta\left(\sum_{k=1}^{n}\frac{1}{k}\right),\quad \log\log n$$

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.

1.21 应该选算法 A. 3 个算法最坏情况下时间复杂度的递推方程及解分别为:

算法 A:$T_A(n)=5T_A(n/2)+O(n)$, $T_A(n)=\Theta(n^{\log 5})$

算法 B:$T_B(n)=2T_B(n-1)+O(1)$, $T_B(n)=\Theta(2^n)$

算法 C:$T_C(n)=9T_C(n/3)+O(n^3)$, $T_C(n)=\Theta(n^3)$

时间复杂度最低的是算法 A.

2.2 设 A 是 n 个非 0 实数构成的数组,设计一个算法重新排列数组中的数,使得负数都排在正数前面. 要求算法使用 $O(n)$的时间和 $O(1)$的空间.

2.2 类似快速排序的划分过程. 从后向前把每个数与 0 比较,找到第一个负数$A[p]$. 从前向后把每个数与 0 比较,找到第一个正数 $A[q]$,如果 $p>q$,则将 $A[p]$与 $A[q]$交换. 交换后如果 $p-q=1$,算法停止;否则继续这个过程,或者找到下一对可以交换的数,或者扫描到 $p\leqslant q$ 停止.

2.3 双 Hanoi 塔问题是 Hanoi 塔问题的一种推广,与 Hanoi 塔的不同点在于:$2n$ 个圆盘,分成大小不同的 n 对,每对圆盘完全相同. 初始,这些圆盘按照从大到小的次序从下到上放在 A 柱上,最终要把它们全部移到 C 柱,移动的规则与 Hanoi 塔相同.

(1) 设计一个移动的算法并给出伪码描述.

(2) 计算你的算法所需要的移动次数.

2.3 (1) 算法设计思想：分治策略. 先递归地将上面的 $2(n-1)$ 个盘子从 A 柱移到 B 柱；用 2 次移动将最大的 2 个盘子从 A 柱移到 C 柱；递归地将 B 柱的 $2(n-1)$ 个盘子从 B 柱移到 C 柱.

伪码描述是：

```
BiHanoi(A,C,n)                    //从 A 到 C 移动 2n 只盘子
1. if n=1 then BiMove(A,C)        //从 A 到 C 移动 2 只盘子
2. else BiHanoi(A,B,n-1)
3.      BiMove(A,C)
4.      BiHanoi(B,C,n-1)
```

(2) 设 $2n$ 个圆盘的移动次数是 $T(n)$，则第 2 行和第 4 行的递归调用的子问题规模是 $n-1$，第 3 行是 2 次移动，于是有

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

解得 $T(n) = 2^{n+1} - 2$.

2.4 给定含有 n 个不同的数的数组 $L = <x_1, x_2, \cdots, x_n>$. 如果 L 中存在 x_i，使得 $x_1 < x_2 < \cdots < x_{i-1} < x_i > x_{i+1} > \cdots > x_n$，则称 L 是单峰的，并称 x_i 是 L 的"峰顶". 假设 L 是单峰的，设计一个算法找到 L 的峰顶.

2.4 因为 L 中存在峰顶元素，因此 $|L| \geqslant 3$. 使用二分查找算法. 如元素数等于 3，则 $L[2]$ 是峰顶元素. 当元素数 n 大于 3 时，令 $k = n/2$，比较 $L[k]$ 与它左边和右边相邻的项. 如果 $L[k] > L[k-1]$ 且 $L[k] > L[k+1]$，则 $L[k]$ 为峰顶元素；否则，如果 $L[k-1] > L[k] > L[k+1]$，则继续搜索 $L[1..k-1]$ 的范围；如果 $L[k-1] < L[k] < L[k+1]$，则继续搜索 $L[k+1..n]$ 的范围. 每比较 2 次，搜索范围减半，直到元素数小于等于 3 停止递归调用. 时间复杂度函数为

$$\begin{cases} T(n) = T(n/2) + 2 \\ T(1) = c, \ c \text{ 为某个常数} \end{cases}$$

根据主定理，$T(n) = O(\log n)$.

2.20 有 n 个人，其中某些人是诚实的，其他人可能会说谎. 现在需要进行一项调查，该调查由一系列测试构成. 每次测试如下进行：选 2 个人，然后提问：对方是否诚实？每个人的回答只能是"是"或者"否". 假定在这些人中，所有诚实的人回答都是正确的，而其他人的回答则不能肯定是否正确. 如果诚实的人数 $> n/2$，试设计一个调查算法，以最小的测试次数从其中找出一个诚实的人.

2.20 采用芯片测试算法 Test. 将 n 个人分成 $n/2$ 组，每组 2 个人，如果 2 个人的回答相同，则他们或都是诚实的，或都说了谎. 如果回答不同，则至少 1 个人说谎. 回答相同的组中任意保留 1 人进入下一轮，回答不同的 2 人都淘汰. 当有轮空的人时需要单独处理. 根据芯片测试算法的分析，经过一轮测试后，人数至少减半，且进入下一轮的诚实的人数多于可能说谎的人数. 该算法的时间复杂度与 Test 算法一样，是 $O(n)$.

2.28 在 Internet 上的搜索引擎经常需要对信息进行比较，比如可以通过某个人对一些事物的排名来估计他(或她)对各种不同信息的兴趣，从而实现个性化的服务．对于不同的排名结果可以用逆序来评价它们之间的差异．考虑 $1,2,\cdots,n$ 的排列 $i_1 i_2 \cdots i_n$，如果其中存在 i_j, i_k，使得 $j<k$ 但是 $i_j>i_k$，那么就称(i_j, i_k)是这个排列的一个逆序．一个排列含有逆序的个数称为这个排列的逆序数．例如排列 2 6 3 4 5 1 含有 8 个逆序(2,1)，(6,3)，(6,4)，(6,5)，(6,1)，(3,1)，(4,1)，(5,1)，它的逆序数就是 8．显然，由 $1,2,\cdots,n$ 构成的所有 $n!$ 个排列中，最小的逆序数是 0，对应的排列就是 $12\cdots n$；最大的逆序数是 $n(n-1)/2$，对应的排列就是 $n(n-1)\cdots 21$．逆序数越大的排列与原始排列的差异度就越大．

利用二分归并排序算法设计一个计数给定排列逆序的分治算法，并对算法进行时间复杂度的分析．

2.28 算法的主要思想是：在二分归并排序算法中附加计数逆序的工作．在递归调用算法分别对子数组 L_1 与 L_2 排序时，分别计数每个子数组内部的逆序；在归并排好序的子数组 L_1 与 L_2 的过程中，附带计数 L_1 的元素与 L_2 的元素之间产生的逆序．假设 L_1 是前半个数组，L_2 是后半个数组．如果 L_1 的最小元素 x 大于 L_2 的最小元素 y，那么算法将从 L_2 中取走 y．这时 L_1 中的每个元素都和 y 构成逆序，所增加的逆序数恰好等于此刻 L_1 中的元素总数．相反，如果 L_1 的最小元素 x 小于 L_2 的最小元素 y，那么算法将从 L_1 中取走 x．这时 L_2 中的每个元素都不会和 x 构成逆序，因此不必改变逆序总数．在算法运行中，每次把这样增加的逆序数加到逆序总数上．

初始逆序数 $N=0$，进入递归算法，算法的主要步骤是：

1. 将数组 L 从中间划分成前后两个子数组 L_1 和 L_2
2. 递归处理 L_1
3. 递归处理 L_2
4. 在归并 L_1 与 L_2 时计数 L_1 与 L_2 的元素产生的逆序数 m，并将 m 加到 N 上

该算法的第 2 步和第 3 步是递归调用，每个子问题的规模都是原问题规模的 1/2，第 4 步每次比较至少拿走 1 个元素，因此元素之间的比较次数与二分归并排序算法的比较次数一样，是 $n-1$ 次．如果以比较运算为基本运算，对于输入规模为 n 的数组所做的比较次数为 $T(n)$，那么有

$$\begin{cases} T(n) = 2T(n/2) + n - 1 \\ T(1) = 0 \end{cases}$$

这就是二分归并排序算法的递推方程，因此得到

$$T(n) = n\log n - n + 1$$

3.1 用动态规划算法求解下面的组合优化问题，

$$\max \quad g_1(x_1)+g_2(x_2)+g_3(x_3)$$
$$x_1^2+x_2^2+x_3^2 \leqslant 10$$
$$x_1, x_2, x_3 \text{ 为非负整数}$$

其中函数 $g_1(x)$，$g_2(x)$，$g_3(x)$的值给在表 3.1 中.

表 3.1　函数值

x	$g_1(x)$	$g_2(x)$	$g_3(x)$	x	$g_1(x)$	$g_2(x)$	$g_3(x)$
0	2	5	8	2	7	16	17
1	4	10	12	3	11	20	22

3.1 设 $F_k(y)$表示对 $x_1, x_2, \cdots, x_k$ 赋值，且 $x_1^2+x_2^2+\cdots+x_k^2 \leqslant y$ 时所得到的目标函数的最大值，递推关系和初值是

$$F_k(y)=\max_{0 \leqslant x_k \leqslant \sqrt{y}}\{F_{k-1}(y-x_k^2)+g_k(x_k)\}$$

$$F_1(y)=g_1(\sqrt{y})$$

针对给定实例的计算过程如下.

$k=1$：

$F_1(0)=2$　　$x_1=0$

$F_1(1)=F_1(2)=F_1(3)=g_1(1)=4$　　$x_1=1$

$F_1(4)=F_1(5)=F_1(6)=F_1(7)=F_1(8)=g_1(2)=7$　　$x_1=2$

$F_1(9)=F_1(10)=g_1(3)=11$　　$x_1=3$

$k=2$：

$F_2(0)=\max\{F_1(0)+g_2(0)\}=7$　　$x_2=0$

$F_2(1)=\max\{F_1(1)+g_2(0), F_1(0)+g_2(1)\}=12$　　$x_2=1$

$F_2(2)=\max\{F_1(2)+g_2(0), F_1(1)+g_2(1)\}=14$　　$x_2=1$

$F_2(3)=\max\{F_1(3)+g_2(0), F_1(2)+g_2(1)\}=14$　　$x_2=1$

$F_2(4)=\max\{F_1(4)+g_2(0), F_1(3)+g_2(1), F_1(0)+g_2(2)\}=18$　　$x_2=2$

$F_2(5)=\max\{F_1(5)+g_2(0), F_1(4)+g_2(1), F_1(1)+g_2(2)\}=20$　　$x_2=2$

$F_2(6)=\max\{F_1(6)+g_2(0), F_1(5)+g_2(1), F_1(2)+g_2(2)\}=20$　　$x_2=2$

$F_2(7)=\max\{F_1(7)+g_2(0), F_1(6)+g_2(1), F_1(3)+g_2(2)\}=20$　　$x_2=2$

$F_2(8)=\max\{F_1(8)+g_2(0), F_1(7)+g_2(1), F_1(4)+g_2(2)\}=23$　　$x_2=2$

$F_2(9)=\max\{F_1(9)+g_2(0), F_1(8)+g_2(1), F_1(5)+g_2(2), F_1(0)+g_2(3)\}=23$　　$x_2=2$

$F_2(10)=\max\{F_1(10)+g_2(0), F_1(9)+g_2(1), F_1(6)+g_2(2), F_1(1)+g_2(3)\}=24$　　$x_2=3$

$k=3$：

$F_3(0)=\max\{F_2(0)+g_3(0)\}=15$ $\qquad x_3=0$

$F_3(1)=\max\{F_2(1)+g_3(0),F_2(0)+g_3(1)\}=20$ $\qquad x_3=0$

$F_3(2)=\max\{F_2(2)+g_3(0),F_2(1)+g_3(1)\}=24$ $\qquad x_3=1$

$F_3(3)=\max\{F_2(3)+g_3(0),F_2(2)+g_3(1)\}=26$ $\qquad x_3=1$

$F_3(4)=\max\{F_2(4)+g_3(0),F_2(3)+g_3(1),F_2(0)+g_3(2)\}=26$ $\qquad x_3=1$

$F_3(5)=\max\{F_2(5)+g_3(0),F_2(4)+g_3(1),F_2(1)+g_3(2)\}=30$ $\qquad x_3=1$

$F_3(6)=\max\{F_2(6)+g_3(0),F_2(5)+g_3(1),F_2(2)+g_3(2)\}=32$ $\qquad x_3=1$

$F_3(7)=\max\{F_2(7)+g_3(0),F_2(6)+g_3(1),F_2(3)+g_3(2)\}=32$ $\qquad x_3=1$

$F_3(8)=\max\{F_2(8)+g_3(0),F_2(7)+g_3(1),F_2(4)+g_3(2)\}=35$ $\qquad x_3=2$

$F_3(9)=\max\{F_2(9)+g_3(0),F_2(8)+g_3(1),F_2(5)+g_3(2),F_2(0)+g_3(3)\}=37$ $\qquad x_3=2$

$F_3(10)=\max\{F_2(10)+g_3(0),F_2(9)+g_3(1),F_2(6)+g_3(2),F_2(1)+g_3(3)\}=37$ $\qquad x_3=2$

关于中间结果的备忘录如表 3.2 所示.

表 3.2 备忘录

y	$k=1$		$k=2$		$k=3$	
	$F_1(y)$	x_1	$F_2(y)$	x_2	$F_3(y)$	x_3
0	2	0	7	0	15	0
1	4	1	12	1	20	0
2	4	1	14	1	24	1
3	4	1	14	1	26	1
4	7	2	18	2	26	1
5	7	2	20	2	30	1
6	7	2	20	2	32	1
7	7	2	20	2	32	1
8	7	2	23	2	35	2
9	11	3	23	2	37	2
10	11	3	24	3	37	2

从而得到，$F_3(10)=37$，此刻 $x_3=2$，于是

$$x_1^2+x_2^2\leqslant 10-2^2=6$$

再查 $F_2(6)=20$，此刻 $x_2=2$，于是

$$x_1^2\leqslant 6-2^2=2$$

再查 $F_1(2)=4$ 得 $x_1=1$. 问题的解是：在 $x_1=1,x_2=2,x_3=2$ 时得到 $g_1(x_1)+g_2(x_2)+g_3(x_3)$的最大值 37.

3.2 设 $A=<x_1,x_2,\cdots,x_n>$是 n 个不等的整数构成的序列，A 的一个单调递增子序列是序列$<x_{i_1},x_{i_2},\cdots,x_{i_k}>$，使得 $i_1<i_2<\cdots<i_k$，且 $x_{i_1}<x_{i_2}<\cdots<x_{i_k}$. 子序列$<x_{i_1},x_{i_2},\cdots,x_{i_k}>$的长度是含有的整数个数 k. 例如 $A=<1,5,3,8,10,6,4,9>$，它的长为 4 的递增子序列是：$<1,5,8,10>$，$<1,5,8,9>$，…. 设计一个算法求 A 的一个最长的单调递增子序列，分析算法的时间复杂度. 设算法的输入实例是 $A=<2,8,4,-4,5,9,11>$，给出算法的计算过程和最后的解.

3.2 使用动态规划设计技术. 对于 $i=1,2,\cdots,n$,考虑以 x_i 作为最后项的最长递增子序列的长度 $C[i]$. 如果在 x_i 项前面存在 $x_j<x_i$,那么 $C[i]=\max\{C[j]\}+1$;否则 $C[i]=1$. 因此

$$C[i]=\begin{cases}\max\{C[j]\}+1 & \exists j(1\leqslant j<i,x_j<x_i)\\ 1 & \forall j(1\leqslant j<i,x_j>x_i)\end{cases},\ i>1$$

$$C[1]=1$$

在计算 $C[i]$时,用 $k[i]$记录 $C[i]$取得最大值时的 j 的值;如果不存在这样的 j,令 $k[i]=0$. 这个记录用于追踪解. 所求的最长递增子序列的长度是

$$C=\max\{C[i]\mid i=1,2,\cdots,n\}$$

对每个 i,需要检索比 i 小的所有的 j,需要 $O(n)$时间,i 的取值有 n 种,于是算法时间复杂度是 $W(n)=O(n^2)$.

对于给定的实例 $A=<2,8,4,-4,5,9,11>$,具体的计算过程如下:

$C[1]=1$

$C[2]=\max\{C[1]+1\}=2$ $\qquad k[2]=1$

$C[3]=\max\{C[1]+1\}=2$ $\qquad k[3]=1$

$C[4]=1$ $\qquad k[4]=0$

$C[5]=\max\{C[1]+1,C[3]+1,C[4]+1\}=3$ $\qquad k[5]=3$

$C[6]=\max\{C[1]+1,C[2]+1,C[3]+1,C[4]+1,C[5]+1\}=4$ $\qquad k[6]=5$

$C[7]=\max\{C[1]+1,C[2]+1,C[3]+1,C[4]+1,C[5]+1,C[6]+1\}=5$ $\qquad k[7]=6$

在 $C[1],C[2],\cdots,C[7]$中,$C[7]=5$ 是最大值,这意味着 A 的第 7 项 $x_7=11$ 是最长递增子序列的最后项,长度是 5. 子序列的构造从后向前进行,开始是 11,追踪过程是:

$k[7]=6\Rightarrow x_6=9,\quad k[6]=5\Rightarrow x_5=5,\quad k[5]=3\Rightarrow x_3=4,\quad k[3]=1\Rightarrow x_1=2$

于是得到解是:$<2,4,5,9,11>$,长度是 5. 本题可以有多个解.

3.16 设 P 是一台 Internet 上的 Web 服务器. $T=\{1,2,\cdots,n\}$是 n 个下载请求的集合,$\forall i\in T,a_i\in\mathbf{Z}^+$ 表示下载请求 i 所申请的带宽. 已知服务器的最大带宽是正整数 K. 我们的目标是使带宽得到最大限度的利用,即确定 T 的一个子集 S,使得 $\sum_{i\in S}a_i\leqslant K$,且 $K-\sum_{i\in S}a_i$ 的值达到最小. 设计一个算法求解服务器下载问题,用文字说明算法的主要设计思想和步骤,给出最坏情况下的时间复杂度.

3.16 设 $x_1, x_2, \cdots, x_n$ 是 0 或 1，$x_i = 1$ 当且仅当下载请求 i 被批准．那么

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

$$\sum_{i=1}^{n} a_i x_i \leqslant K, \quad x_i = 0, 1$$

类似于 0-1 背包问题，令 $F_i(y)$ 表示考虑前 i 个申请，带宽限制为 y 时的最大带宽使用量，则有如下递推式：

$$F_i(y) = \max\{F_{i-1}(y), F_{i-1}(y - a_i) + a_i\}, \quad i > 1,\ 0 < y \leqslant K$$

$$F_1(y) = \begin{cases} a_1, & y \geqslant a_1 \\ 0, & y < a_1 \end{cases}$$

$$F_i(0) = 0$$

$$F_i(y) = -\infty, \quad y < 0$$

与前面背包问题类似可设立标记函数，自底向上顺序处理 $i = 1, 2, \cdots, n$ 的情况．对于给定的 i，令 $y = 1, 2, \cdots, K$，依次确定各个子问题的 $F_i(y)$ 值，最后由标记函数得到 x_i 的值．算法时间复杂度为 $O(nK)$．

3.17 有正实数构成的数字三角形排列形式如图 3.2 所示．第一行的数为 a_{11}；第二行的数从左到右依次为 a_{21}, a_{22}；以此类推，第 n 行的数为 $a_{n1}, a_{n2}, \cdots, a_{nn}$．从 a_{11} 开始，每一行的数 a_{ij} 只有两条边可以分别通向下一行的两个数 $a_{(i+1)j}$ 和 $a_{(i+1)(j+1)}$．请设计一个算法，计算出从 a_{11} 通到 $a_{n1}, a_{n2}, \cdots, a_{nn}$ 中某个数的一条路径，并且使得该路径上的数之和达到最小．

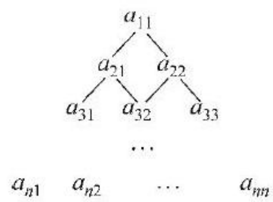

图 3.2　数字三角形

3.17 令 $F[i,j]$ 表示 a_{11} 到 a_{ij} 的路径上的数的最小和，则

$$F[i,j] = \min\{F[i-1,j], F[i-1,j-1]\} + a_{ij}, \quad i = 2,3,\cdots,n,\ j = 2,3,\cdots,i-1$$

$$F[i,1] = F[i-1,1] + a_{i1}, \quad i = 2,3,\cdots,n$$

$$F[i,i] = F[i-1,i-1] + a_{ii}, \quad i = 2,3,\cdots,n$$

$$F[1,1] = a_{11}$$

问题的最优路径上的和是

$$\min\{F[n,j] \mid j = 1,2,\cdots,n\}$$

可以定义标记函数 $k[i,j]$，记录得到最优 $F[i,j]$ 时的路径选择，即

$$k[i,j] = \begin{cases} j & 若\ F[i-1,j] < F[i-1,j-1] \\ j-1 & 否则 \end{cases}$$

算法的时间复杂度为 $O(n^2)$．

4.12 设字符集 S,其中 8 个字符 A,B,C,D,E,F,G,H 的频率是 $f_1,f_2,\cdots,f_8$,且 $100\times f_i$ 是第 i 个 Fibonacci 数的值,$i=1,2,\cdots,8$.

(1) 给出这 8 个字符的 Huffman 树和编码.

(2) 如果有 n 个字符,其频率恰好对应前 n 个 Fibonacci 数,那么 Huffman 树是什么结构,证明你的结论.

4.12 (1) Huffman 树如图 4.2 所示. 编码为

H：0， G：10， F：110， E：1110， D：11110，

C：111110， B：1111110， A：1111111

(2) 先证明以下命题.

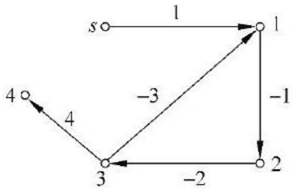

图 4.1 反例

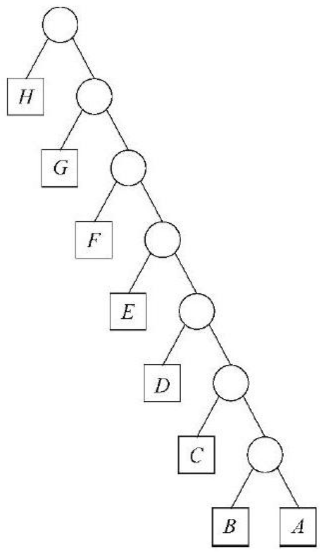

图 4.2 Huffman 树

命题 4.10 设 $f_1,f_2,\cdots$ 为 Fibonacci 数列,则

$$\sum_{i=1}^{k} f_i \leqslant f_{k+2}$$

证 对 k 归纳.

当 $k=1$ 时,$f_1<f_3$ 显然为真.

假设 $k=n$ 时命题成立,则 $k=n+1$ 时,有

$$\sum_{i=1}^{n+1} f_i = \sum_{i=1}^{n} f_i + f_{n+1} \leqslant f_{n+2} + f_{n+1} = f_{n+3}$$

所以命题对于 $k=n+1$ 成立.

综上所述,$\sum_{i=1}^{k} f_i \leqslant f_{k+2}$ 对任意正整数 k 成立.

根据命题 4.10,前 k 个字符合并后子树的根的权值不大于第 $k+2$ 个 Fibonacci 数. 根据 Huffman 算法,它将继续参加与第 $k+1$ 个字符的合并. 因此 n 个字符的 Huffman 编码按照频率从小到大依次是:

11…1(含 $n-1$ 个 1), 11…10(含 $n-2$ 个 1), 11…10(含 $n-3$ 个 1),…, 10, 0

即第 $i(i>1)$ 个字母的编码为 11…10(含 $n-i$ 个 1).

4.19 有 n 个文件存在磁带上，每个文件占用连续的空间. 已知第 i 个文件需要的存储空间为 s_i，被检索的概率是 $f_i, i=1,2,\cdots,n$，且 $f_1+f_2+\cdots+f_n=1$. 检索每个文件需要从磁带的开始位置进行操作，比如文件 i 需要空间 $s_i=310$，存储在磁带的 121～430 单元，那么检索该文件需要的时间为 430. 问如何排列 n 个文件使得平均检索时间最少？设计算法求解这个问题，说明算法的设计思想，证明算法的正确性，给出算法最坏情况下的时间复杂度.

4.19 采用贪心法.

贪心策略：按照单位存储空间被检索的概率 f_i/s_i 从大到小排序文件使得

$$\frac{f_1}{s_1} \geqslant \frac{f_2}{s_2} \geqslant \cdots \geqslant \frac{f_n}{s_n}$$

然后按照文件排序的顺序依次存入磁带.

命题 4.18 按上述次序存储磁盘的平均检索时间达到最小.

证 设某个最优解 OPT(I)的磁盘文件顺序是 $i_1, i_2, \cdots, i_n$，则平均检索时间是

$$t(\mathrm{OPT}(I)) = f_{i_1}s_{i_1} + f_{i_2}(s_{i_1}+s_{i_2}) + f_{i_3}(s_{i_1}+s_{i_2}+s_{i_3}) + \cdots + f_{i_n}(s_{i_1}+\cdots+s_{i_n})$$

如果 OPT(I)与算法的解不等，那么它们的区别在于 OPT(I)中存在逆序. 若在 OPT(I)中存在逆序，一定存在相邻的逆序，比如第 i 个文件和第 j 个文件构成相邻的逆序. 交换文件 i 和 j 得到解 $P(I)$，那么

$$\begin{aligned} t(\mathrm{OPT}(I)) - t(P(I)) &= [f_i s_i + f_j(s_i+s_j)] - [f_j s_j + f_i(s_j+s_i)] \\ &= f_j s_i - f_i s_j \geqslant 0 \quad \left(\frac{f_i}{s_i} \leqslant \frac{f_j}{s_j}\right) \end{aligned}$$

从而证明了从 OPT(I)出发，交换具有相邻逆序的元素后仍旧得到最优解. 每进行 1 次交换，消除 1 个逆序. 至多经过 $n(n-1)/2$ 次交换，就消除了所有的逆序，从而得到算法的解，因此算法的解也是最优解.

算法最坏情况下时间复杂度是 $O(n\log n)$.

5.6 子集和问题. 设 n 个不同的正数构成集合 S，求出使得和为某数 M 的 S 的所有子集.

5.6 设 $S=\{a_1,a_2,\cdots,a_n\}$. 求 S 满足条件 $\sum_{a_i \in A} a_i = M$ 的所有的子集 A. 用回溯算法. 解向量为 $<x_1,x_2,\cdots,x_n>$，$x_i=0,1$. 其中 $x_i=1$ 当且仅当 $a_i \in A$. 搜索空间为子集树. 部分向量 $<x_1,x_2,\cdots,x_k>$ 表示已经考虑了对 $a_1,a_2,\cdots,a_k$ 的选择. 结点分支的约束条件为

$$B(i) = \sum_{i=1}^{k} a_i x_i < M \quad \text{且} \quad a_{k+1} \in S - \{a_1,a_2,\cdots,a_k\}$$

最坏情况下算法的时间复杂度为 $O(2^n)$.

5.7 分派问题. 给 n 个人分配 n 件工作，给第 i 个人分配第 j 件工作的成本是 $C(i,j)$. 试求成本最小的工作分配方案.

5.7 设 n 个人的集合是$\{1,2,\cdots,n\}$，n 项工作的集合是$\{1,2,\cdots,n\}$，每个人恰好 1 项工作.

$$\text{把工作 } j \text{ 分配给 } i \Leftrightarrow x_i = j, \quad i,j = 1,2,\cdots,n$$

解向量是 $X=\langle x_1,x_2,\cdots,x_n\rangle$，分配成本是 $C(X)=\sum_{i=1}^{n} C(i,x_i)$．搜索空间是排列树. 部分向量$\langle x_1,x_2,\cdots,x_k\rangle$表示已经考虑了对人 $1,2,\cdots,k$ 的工作分配. 结点分支的约束条件为：

$$x_{k+1} \in \{1,2,\cdots,n\} - \{x_1,x_2,\cdots,x_k\}$$

可以设立代价函数：

$$\begin{aligned} & F(x_1,x_2,\cdots,x_k) \\ = & \sum_{i=1}^{k} C(i,x_i) + \sum_{i=k+1}^{n} \min\{C(i,t) \mid t \in \{1,2,\cdots,n\} - \{x_1,x_2,\cdots,x_k\}\} \end{aligned}$$

界 B 是已得到的最好可行解的分配成本. 如果代价函数大于界，则回溯.

算法最坏情况下的时间复杂度是 $O(nn!)$.

6.4 用图解法解下列线性规划.

(1) $\max\ x_1+x_2$

s. t. $x_1\leqslant 5$

$x_2\leqslant 3$

$x_1+3x_2\leqslant 11$

$x_1,x_2\geqslant 0$

(2) $\min\ x_1-x_2$

s. t. $2x_1+3x_2\leqslant 14$

$-x_1+x_2\leqslant 3$

$x_1\leqslant 4$

$x_1,x_2\geqslant 0$

(3) $\min\ 2x_1+x_2$

s. t. $x_1+x_2\geqslant 1$

$x_2\leqslant 2$

$x_1,x_2\geqslant 0$

(4) $\min\ 2x_1-x_2$

s. t. $2x_1+x_2\geqslant 2$

$x_1-x_2\leqslant -3$

$x_1,x_2\geqslant 0$

(5) $\max\ 3x_1-2x_2$

s. t. $x_1-x_2\geqslant -1$

$3x_1+x_2\leqslant 9$

$x_1+2x_2\geqslant 9$

$x_1,x_2\geqslant 0$

6.4 (1) 见图 6.1,最优解为点 A.

$$x_1 = 5, \quad x_2 = 2, \quad z = 7.$$

(2) 见图 6.2,最优解是线段 AB 上的所有点,有无穷多个解.

$$x_1 = 1 - t, \quad x_2 = 3t + 4(1 - t) = 4 - t, \quad z = -3, \quad 0 \leqslant t \leqslant 1$$

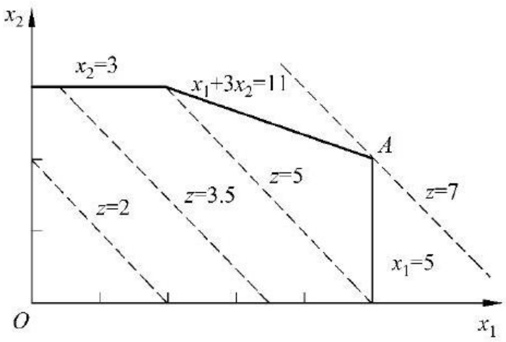

图 6.1

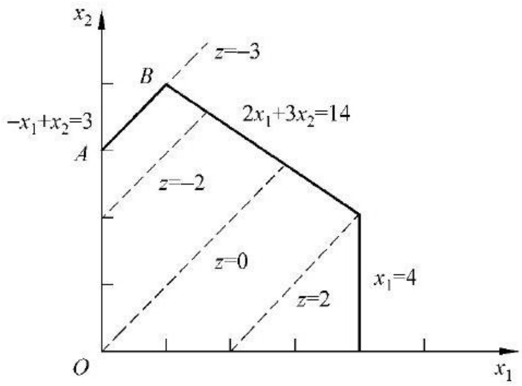

图 6.2

(3) 见图 6.3,最优解是点 A.

$$x_1 = 0, \quad x_2 = 1, \quad z = 1.$$

(4) 见图 6.4,有可行解,目标函数无界,无最优解.

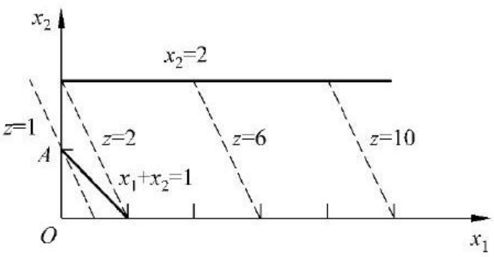

图 6.3

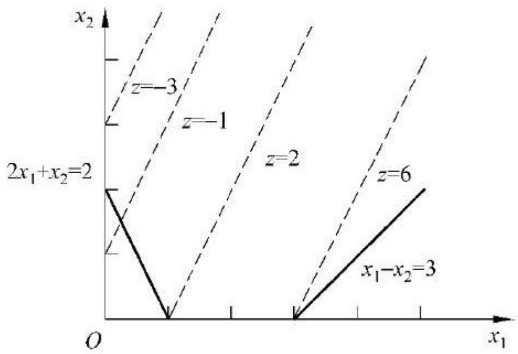

图 6.4

(5) 见图 6.5,无可行解.

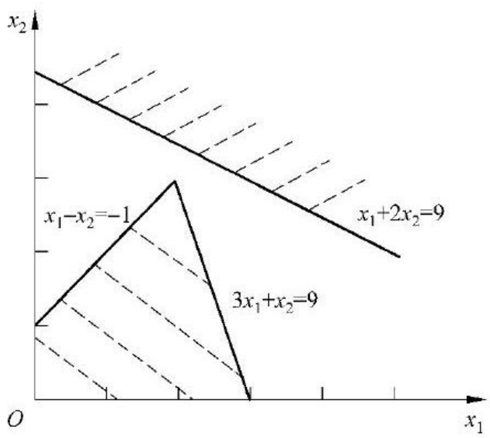

图 6.5

6.5 写出下述线性规划的标准形.

$$
\begin{aligned}
&\max\ 3x_1-2x_2+x_3\\
&\text{s.t.}\ \ x_1+2x_2-x_3\leqslant 1\\
&\quad 4x_1-2x_3\geqslant 5\\
&\quad x_2-5x_3\leqslant -4\\
&\quad x_1-3x_2+2x_3=-10\\
&\quad x_1\geqslant 0,\ x_2\ \text{任意},\ x_3\geqslant 0
\end{aligned}
$$

6.5 将 max 改写成 min,第 3 和第 4 式两边变号,并令 $x_2=x_{21}-x_{22}$,

$$
\begin{aligned}
&\min -3x_1+2x_{21}-2x_{22}-x_3\\
&\text{s.t.}\ \ x_1+2x_{21}-2x_{22}-x_3\leqslant 1\\
&\quad 4x_1-2x_3\geqslant 5\\
&\quad -x_{21}+x_{22}+5x_3\geqslant 4\\
&\quad -x_1+3x_{21}-3x_{22}-2x_3=10\\
&\quad x_1,x_{21},x_{22},x_3\geqslant 0
\end{aligned}
$$

再引入松弛变量 x_4 和剩余变量 x_5,x_6,得到标准形

$$
\begin{aligned}
&\min -3x_1+2x_{21}-2x_{22}-x_3\\
&\text{s.t.}\ \ x_1+2x_{21}-2x_{22}-x_3+x_4=1\\
&\quad 4x_1-2x_3-x_5=5\\
&\quad -x_{21}+x_{22}+5x_3-x_6=4\\
&\quad -x_1+3x_{21}-3x_{22}-2x_3=10\\
&\quad x_1,x_{21},x_{22},x_3,x_4,x_5,x_6\geqslant 0
\end{aligned}
$$

6.6 设线性规划

$$
\begin{aligned}
&\max\ 2x_1+x_2\\
&\text{s.t.}\ -x_1+2x_2\leqslant 4\\
&\quad x_1\leqslant 5\\
&\quad x_1,x_2\geqslant 0
\end{aligned}
$$

(1) 画出它的可行域,用图解法求最优解.

(2) 写出它的标准形,列出所有的基,指出哪些是可行基. 通过列出所有的可行解及其目标函数值找到最优解. 指出每个可行解对应的可行域的顶点.

6.6 (1) 见图 6.6,最优解是点 B.

$x_1=5,\quad x_2=4.5,\quad z=14.5.$

(2) 标准形为

$$
\begin{aligned}
&\min -2x_1-x_2\\
&\text{s.t. } -x_1+2x_2+x_3 \qquad = 4\\
&\qquad x_1 \qquad\qquad +x_4=5\\
&\qquad x_j\geqslant 0,\quad 1\leqslant j\leqslant 4
\end{aligned}
$$

$$A=\begin{pmatrix}-1 & 2 & 1 & 0\\ 1 & 0 & 0 & 1\end{pmatrix}$$

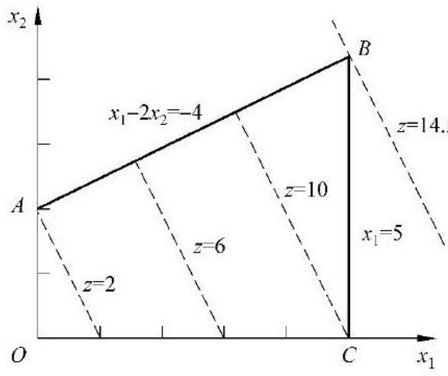

图 6.6

$B_1=(P_1,P_2)=\begin{pmatrix}-1 & 2\\ 1 & 0\end{pmatrix}$, $x_1^{(1)}=5$, $x_2^{(1)}=4.5$, $x_3^{(1)}=0$, $x_4^{(1)}=0$, $z^{(1)}=-14.5$. 对应点 B. B_1是可行基.

$B_2=(P_1,P_3)=\begin{pmatrix}-1 & 1\\ 1 & 0\end{pmatrix}$, $x_1^{(2)}=5$, $x_2^{(2)}=0$, $x_3^{(2)}=9$, $x_4^{(2)}=0$, $z^{(2)}=-10$. 对应点 C. B_2是可行基.

$B_3=(P_1,P_4)=\begin{pmatrix}-1 & 0\\ 1 & 1\end{pmatrix}$, $x_1^{(3)}=-4$, $x_2^{(3)}=0$, $x_3^{(3)}=0$, $x_4^{(3)}=9$. B_3是基,但不是可行基.

$B_4=(P_2,P_3)=\begin{pmatrix}2 & 1\\ 0 & 0\end{pmatrix}$,不是基.

$B_5=(P_2,P_4)=\begin{pmatrix}2 & 0\\ 0 & 1\end{pmatrix}$, $x_1^{(5)}=0$, $x_2^{(5)}=2$, $x_3^{(5)}=0$, $x_4^{(5)}=5$, $z^{(5)}=-2$. 对应点 A, B_5是可行基.

$B_6=(P_3,P_4)=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$, $x_1^{(5)}=0$, $x_2^{(5)}=0$, $x_3^{(5)}=4$, $x_4^{(5)}=5$, $z^{(5)}=0$. 对应点 O,B_6是可行基.

$x^{(1)}=(5,4.5,0,0)$是最优解.