第一章作业答案

作业题目 1.2 节 P14 第 5 题；1.4 节 P30 第 6 题

1 （P14-5） 写出将十进制正整数转换成二进制证书的标准算法，a 用文字描述，b 用伪代码描述

算法思路：任何一个正整数都可以用 2 的幂次方表示，例如，137=2^7+2^3+2^0，所以将正整数每次%2 就能得到当前位置上的 2 进制数。
输入：一个正整数 n
输出：正整数 n 相应的二进制数
第一步：用 n 除以 2，余数赋给 Ki(i=0,1,2...)，商赋给 n 第二步：如果 n=0，则到第三步，否则重复第一步 第三步：将 Ki 按照 i 从高到低的顺序输出

伪代码实现：

```
Algorithm DectoBin（n）
Begin
        i :=n;
        j :=0;
        while   (i≠  0)   do
                  i := i/2;
                   B[j]:=i%2;
                   j :=j+1；
        end;
        while (j > 0) do
                  j :=j-1;
                  Print B[j];
         end;
end
```

2（P30-6）证明下列关于 n 个顶点二叉树高度的不等式 $\lfloor \log_2 n \rfloor \leqslant h \leqslant n-1$

不等式右边的证明：

法一：

当二叉树的每层只有一个节点的时达到最大高度 n-1，所以二叉树的高度 h ≤ n-1，故证；

法二：

归纳法

n = 1 时，h = 0，满足 h≤n-1；

假设 n=k 时，h≤k-1，记为 h(k)≤k-1；

对于 n=k+1，若 n=k 的二叉树为满二叉树，则 h(k+1)=h(k)+1，所以 h(k+1)≤k；若 n=k 的二叉树不为满二叉树，则 h(k+1)=h(k)，h(k+1)≤k-1＜k，此时满足。

综上，h≤n-1，二叉树除叶子节点外其他所有节点只有一个孩子节点时，取等号。

不等式左边的证明：

方法一

当n=1时，只有一个结点的二叉树的高度为0，成立。

令有x个结点的二叉树的高度为h(x)

假设当n=k，k>=2时，结论也成立，即有k个结点的非空二叉树的高度

$h(k) \geqslant \lfloor lgk \rfloor = \lceil lg(k+1) \rceil - 1 \geqslant lg(k+1) - 1$

将n个结点的非空二叉树分成左子树k个结点和右子树n-k-1个结点，于是结点数为n的树高度

$h(n) = max(h(k), h(n-k-1)) + 1$

由假设可知：$h(k) \geqslant \lfloor lgk \rfloor, h(n-k-1) \geqslant \lfloor lg(n-k-1) \rfloor$

当$k \geqslant n-k-1$时，即$k \geqslant (n-1)/2$时，

$h(n) = h(k) + 1 \geqslant \lfloor lg((n-1)/2) \rfloor + 1 = \lceil (lg((n-1)/2) + 1) \rceil - 1 + 1$

$\geqslant (lg(n+1)/2) = lg(n+1) - 1$

故证。

（注意：$\lceil \log_2(n+1) \rceil = \lfloor \log_2 n \rfloor + 1$）

方法二：

满二叉树：高度为 h，由 $2^{h+1}-1$ 个节点构成的二叉树称为满二叉树。

证明：对任意一个高度为 h 的二叉树，节点数最多为高度为 h 的满二叉树的节点数，而高度为 h 的满二叉树的节点数为 $2^{h+1}-1$，故，对任意高度为 h 的二叉树，其节点数一定小与等于 $2^{h+1}-1$， 即 $n \leq 2^{h+1}-1$。

$\Rightarrow \lfloor \log_2 n \rfloor \leq \lfloor \log_2(2^{h+1}-1) \rfloor$

而 $\lfloor \log_2(2^{h+1}-1) \rfloor = h$

所以 $\lfloor \log_2 n \rfloor \leq h$

第二章作业

作业题目（2.2 节 P47 第 5 题，第 6（a）题，2.3 节 P52 第 4 题，2.4 节 P60 第 3 题和第 8 题）

2.2 节 P47 第 5 题，第 6（a）题

5. 根据下列函数的增长次数按照从低到高的顺序对它们进行排序。

$(n-2)!$，$5\lg(n+100)^{10}$，2^{2n}，$0.001n^4+3n^3+1$，$\ln^2 n$，$\sqrt[3]{n}$，3^n

6. **a.** 证明当 $a_k>0$ 时，任何多项式 $p(n)=a_k n^k+a_{k-1}n^{k-1}+\cdots+a_0$ 属于集合 $\Theta(n^k)$。

5、答案

$(n-2)!\in\Theta(\ n!\ )$，$5\lg(n+100)^{10}=50\lg(n+100)\in\Theta(\log n)$，$2^{2n}=(2^2)^n\in\Theta(4^n)$，$0.001n^4+3n^3+1\in\Theta(n^4)$，$\ln^2 n\in\Theta(\log^2 n)$，$\sqrt[3]{n}\in\Theta(n^{\frac{1}{3}})$，$3^n\in\Theta(3^n)$.

按增长次数从低到高顺序排列如下：

$$5\lg(n+100)^{10},\quad \ln^2 n,\quad \sqrt[3]{n},\quad 0.001n^4+3n^3+1,\quad 3^n,\quad 2^{2n},\quad (n-2)!$$

6、 答案

6. a. $\lim\limits_{n\to\infty}\frac{p(n)}{n^k}=\lim\limits_{n\to\infty}\frac{a_k n^k+a_{k-1}n^{k-1}+\ldots+a_0}{n^k}=\lim\limits_{n\to\infty}\left(a_k+\frac{a_{k-1}}{n}+\ldots+\frac{a_0}{n^k}\right)$

$=a_k>0.$

因此，$p(n)\in\Theta(n^k)$。

2.3 节 P52 第 4 题

4. 考虑下面的算法。

算法 Mystery(n)

```
//输入：非负整数 n
S ← 0
for i ← 1 to n do
    S ← S + i * i
return S
```

a. 该算法求的是什么？

b. 它的基本操作是什么？

c. 该基本操作执行了多少次？

4、答案

a. 计算 $S(n)=\sum_{i=1}^{n} i^2.$

b. 乘法（或者，如果假定乘法和加法的时间相同，则两者中的任何一个）。

c. n 次

2.4 节 P60 第 3 题和第 8 题

3. 考虑下列递归算法，该算法用来计算前 n 个立方的和：$S(n)=1^3+2^3+\cdots+n^3$。

算法 $S(n)$
//输入：正整数 n
//输出：前 n 个立方的和
if $n=1$ **return** 1
else return $S(n-1)+n*n*n$

a. 建立该算法的基本操作执行次数的递推关系并求解。

b. 如果将这个算法和直截了当的非递归算法比较，你做何评价？

3、答案

a. 我们可以通过反向替换来解决它：

$$\begin{aligned} M(n) &= M(n-1)+2 \\ &= [M(n-2)+2]+2 = M(n-2)+2+2 \\ &= [M(n-3)+2]+2+2 = M(n-3)+2+2+2 \\ &= \ldots \\ &= M(n-i)+2i \\ &= \ldots \\ &= M(1)+2(n-1) = 2(n-1). \end{aligned}$$

b. 非递归的伪码：

算法 NonrecS(n)
// 输入：一个正整数 n
// 输出： 前 n 个立方的和

$S \leftarrow 1$
for $i \leftarrow 2$ **to** n **do**
    $S \leftarrow S + i * i * i$
return S

这种算法的乘法次数将是：

$$\sum_{i=2}^{n} 2 = 2\sum_{i=2}^{n} 1 = 2(n-1).$$

这与递归版本中的基本操作次数相同，但非递归版本不携带与递归堆栈相关的时间和空间开销。

8. **a.** 请基于公式 $2^n=2^{n-1}+2^{n-1}$ 设计一个递归算法。当 n 是任意非负整数时，该算法能够计算 2^n 的值。

b. 建立该算法所做的加法运算次数的递推关系并求解。

c. 为该算法构造一棵递归调用树，然后计算它所做的递归调用的次数。

d. 对于该问题的求解来说，这是一个好算法吗？

8、 答案

a.

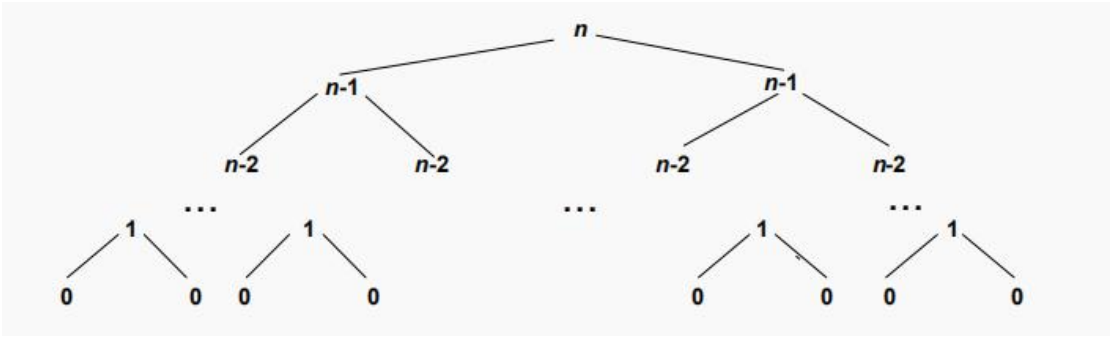
算法 Power(n)
// 通过公式 $2^n=2^{n-1}+2^{n-1}$ 递归计算 2^n

// 输入： 非负整数 n
// 输出：2^n 的值
if n = 0 return 1
else return Power(n-1)+Power(n-1)

b. $A(n) = 2A(n-1) + 1, \ A(0) = 0.$

$$
\begin{aligned}
A(n) &= 2A(n-1)+1 \\
&= 2[2A(n-2)+1]+1 = 2^2A(n-2)+2+1 \\
&= 2^2[2A(n-3)+1]+2+1 = 2^3A(n-3)+2^2+2+1 \\
&= \ldots \\
&= 2^iA(n-i)+2^{i-1}+2^{i-2}+\cdots+1 \\
&= \ldots \\
&= 2^nA(0)+2^{n-1}+2^{n-2}+\cdots+1 = 2^{n-1}+2^{n-2}+\cdots+1 = 2^n-1.
\end{aligned}
$$

c. 该算法的递归调用树如下：



d. 这是一个非常糟糕的算法，因为它远不如简单地将累加器乘以 2n 倍，更不用说我们后来学到的更有效的算法了。

第四章作业

减治作业题目（4.1 节 P105，附加 1，7，9；4.3 节 P114，2；4.4 节 P120，1，3，9，10，11；4.5 节 P128，2，7，13）

习题 4.1 P105

附加 1 实现折半插入排序，分析算法的时间复杂度。

思路：在将一个新元素插入已排好序的数组的过程中，寻找插入点时，将待插入区域的首元素设置为 a[low]，末元素设置为 a[high]，则每轮比较时将待插入元素与参考元素 a[m]，其中 m=(low+high)/2 相比较,如果比参考元素小，则选择 a[low]到 a[m-1]为新的插入区域(即 high=m-1)，否则选择 a[m+1]到 a[high]为新的插入区域（即 low=m+1），如此直至 low<=high 不成立，即将此位置之后所有元素后移一位，并将新元素插入 a[high+1]。

伪代码实现

```
算法 BinInsertSort（A[1..n]）
 {
    For i=2 to n do
    {
        low=1;
        high=i;
        A[0]=A[i]
        While(low < high)
            {
               m=(low+high)/2;
               If a[m]>a[i]
                 high=m-1;
               else
                 low=m+1;
            }//while
      for j = i - 1 down to high + 1
             A[j + 1] = A[j]; //  记录后移

       a[high + 1] = arr[0]; //  插入

    }//For

}//算法
```

时间复杂度分析：

折半插入排序算法是一种稳定的排序算法，比直接插入算法明显减少了关键字之间比较的次数，因此速度比直接插入排序算法快，但记录移动的次数没有变，所以折半插入排序算法的时间复杂度仍然为 O(n^2)，与直接插入排序算法相同。附加空间 O(1)。

7. 应用插入排序将序列 E, X, A, M, P, L, E 按照字母顺序排序。

排序结果：

E
EX
AEX
AEMX
AEMPX
AELMPX
AEELMPX

9. 能不能实现一个对链表排序的插入排序算法？它是不是和数组版本都一样有着$O(n^2)$效率呢？

链表的插入排序时间复杂度是$O(n^2)$比数组版少了移动元素，但是仍需要元素比较；链表的插入排序没法选用折半查找，只能用顺序查找，所以对于元素比较次数无法优化
伪代码

习题 4.3 P114

2. 使用下面的方法生成{1, 2, 3, 4}的全部排列：

a. 从底向上的最小变化算法。

b. Johnson-Trotter 算法。

c. 字典序算法。

答案：

a 减一策略生成排列

(i)	1
(ii)	12 21
(iii)	123, 132, 312 213, 231, 321
(iv)	1234, 1243, 1423, 4123;　1324, 1342, 1432, 4132;　3124, 3142, 3412, 4312 2134, 2143, 2413, 4213;　2314, 2341, 2431, 4231;　3214, 3241, 3421, 4321

B Johnson-Trotter 算法

1234, 1243, 1423, 4123;　4132, 1432, 1342, 1324;　3124, 3142, 3412, 4312;
4321, 3421, 3241, 3214;　2314, 2341, 2431, 4231;　4213, 2413, 2143, 2413;

C 字典排序法

1234, 1243, 1324, 1342, 1423, 1432;　2134, 2143, 2314, 2341, 2413, 2431;
3124, 3142, 3214, 3241, 3412, 3421;　4123, 4132, 4213, 4231, 4312, 4321;

P120

习题 4.4

1. **切割木棍** 一根 n 英寸长的木棍需要切割成 n 段 1 英寸长的小段。描述以最小切割次数完成该任务的算法。如果一次能切多根木棍，再给出最小切割次数的公式。

答案：最小切割次数$log_2 n$

3. **a.** 在下面的数组中查找一个键时，折半查找最多需要进行多少次键值比较？

3	14	27	31	39	42	55	70	74	81	85	93	98

b. 请列出所有这样的键，对于它们，折半查找在查找该数组时，需要进行最多次的键值比较。

c. 在对该数组折半查找成功的前提下，求键值比较的平均次数(假设查找每一个键的概率都是相同的)。

d. 在对该数组折半查找失败的前提下，求键值比较的平均次数(假设查找键位于该数组构成的 14 个区间内的概率都是相同的)。

3 答案：

a 键值比较$log_2 13$，最多 4 次

b 比较次数最多的元素是 14，31，42，74，85，98

c 查找成功的平均查找次数（4*6+3*4+2*2+1*1）/13=41/13=3.15

d 查找失败的平均查找次数 4

9. 一个数组 A[0.. $n-2$]包含 $n-1$ 个从 1 到 n 的整数(因此在这个范围内缺少一个整数)，元素升序排列。尽你所能设计一个求缺失整数的最有效算法，并说明它的时间效率。

```
算法 BinFinMissing（A[0..n-2]）
 {
        low=0;
        high=n-2;
        while(low<high)
        {
            mid=(low+high)/2;
            if(A[mid]>mid+1)
                high=mid-1;
            else
               {
                Low=mid+1;
               }//else

        }
        if(left==n-2)//需要注意一个情况，就是前 n-1 个数都在数组中出现，那缺失的那个数就是 n
            return n;
        else return low+1;

}//
```

时间复杂度分析 logn

10. a. 为假币问题的三分算法写一段伪代码。请确保该算法会正确处理所有的 n 值，而不仅仅是那些 3 的倍数。

b. 为假币问题的三分算法的称重次数建立一个递推关系，并在 $n = 3^k$ 的情况下对它求解。

c. 当 n 的值非常大时，该算法要比把硬币分成两堆的算法快多少倍？这个答案应该与 n 无关。

伪代码：
（1）n%3==0，两堆 n/3 上称称重，一堆 n/3 剩余，如果称平衡，则剩余堆含有假金币，继续分三堆称重；否则轻的堆含有假金币，继续分三堆称重
（2）n%3==1，两堆 n/3 上称称重，一堆 n/3+1 剩余，如果称平衡，则剩余堆含有假金币，继续分三堆称重；否则轻的堆含有假金币，继续分三堆称重
（3）n%3==2，两堆 n/3+1 上称称重，一堆 n/3 剩余，如果称平衡，则剩余堆含有假金币，继续分三堆称重；否则轻的堆含有假金币，继续分三堆称重

11. a. 应用俄式乘法来计算 26×47。

b. 从时间效率的角度看，我们用俄式乘法算法计算 $n \times m$ 和 $m \times n$ 有区别吗？

答案
11a　26*47=13*94=6*188+94=3*376+94=1*752+94+376=1222
11b 有区别，第一个乘数小的时候，迭代次数较少，否则反之

习题 4.5 P128

2. 应用快速选择算法求数列 9, 12, 5, 17, 20, 30, 8 的中位数。

答案：7 个数中位数的存储位置是 4
（i）9 为支点，快排结果：{5，8}，9，{17，20，30，12}，9 的位置是 3，3<4，进入 9 后面大数分区{17，20，30，12}查找
（ii）17 为指点快排结果：{12}，17，{30，20}，17 的位置是 5，5>4，进入 17 前面小数分区{12}查找
（iii）{12}，12 存储为位置 4=4，所以 12 是中位数

13. 假设需要在一个 $n \times n$ 矩阵中搜索一个给定数字，该矩阵每行每列都按升序排列。你能为这个问题设计一个 O(n)算法吗？([Laa10])

设搜索码值为 k
找出中间位置$n^2/2$ 所对应元素与 k 比较大小
（1-1）相等，则找到
（1-2）不相等，依次取出中间位置数所在列的所有元素与 k 比较大小，从而确定 k 所在具体行号，转入相应行依次比较查找，直到找到或者失败。

7. **a.** 描述一个在二叉查找树中寻找最大键的算法。你会把你的算法归类为减可变规模算法吗？

b. 你的算法在最坏情况下的效率类型是哪一种？

答案 7a 每次查找会进入相应分支，缩小一定规模。但是查找树不一定均衡，所以缩小规模不确定，属于减可变规模
7b 最坏情况每次缩小规模减少 1 个码值，退化成顺序查找，时间复杂度 O（n）

第五章作业

分治作业题目（附加题 1-4，5.1 节 P135，第 2 题，第 9 题，5.2 节 P141 第 11 题，5.3 节 P142-143 第 1 题和第 2 题）

预备知识：T(n)=aT(n/b)+f(n)递推式的解法
直接使用公式

$$if \quad f(n) \in \Theta(n^d) \quad d \geq 0$$

$$T(n) \in \begin{cases} \Theta(n^d) & a < b^d \\ \Theta(n^d \log n) & a = b^d \\ \Theta(n^{\log_b a}) & a > b^d \end{cases}$$

1. 对 f(n)=n，并且 a=b 的情况， 证明上述定理。

解： 由 $f(n)=n, f(n) \in \Theta(n^d)$，得 $d=1$

又 $a=b$，即要求我们证明 $T(n) \in \Theta(n \log n)$。不失一般性， 假定 n 为 a 的整幂，即存在正整数 k，使得 $n=a^k$ 。同时假定 $T(1)$ 为常数，当 $n>1$ 时， 则：

$$T(n)=aT(n/a)+n$$

$$=a(aT(n/a^2)+n/a)+n=a^2T(n/a^2)+2n$$
$$=a^2(aT(n/a^3)+n/a^2)+2n=a^3T(n/a^3)+3n$$
$$=.....$$
$$=a^kT(n/a^k)+kn$$

因为 $a^k=n$，所以 $k=\log_a n$，于是 $T(n)=nT(1)+n\log_a n$， 从而 $T(n) \in \Theta(n \log n)$。

2. 设计求解 n 个数字之和的分治算法，给出伪代码描述并分析算法的计算复杂度（每次分为 2 个子问题）。

解：

算法思想：将 n 个待相加的元素分割成 2 个大致相等的子集合 A、B；对每一个子集合分别递归求和；再将每个子集的和相加。当 n 等于 1 时递归终止，此时所求之和为该元素自身。

算法的形式化描述如下：

Algorithm Sum (A[0 ... n-1])

//输入：n 个数字

//输出：n 个数字的和

if n>1 **then**

sum := **Sum**(A[0...$\lfloor n/2 \rfloor$-1]) + **Sum**(A[$\lfloor n/2 \rfloor$...n-1]);

return sum

else return A[0]

时间复杂度分析：设 $T(n)$ 为算法计算 n 个数字之和所花费的基本运算次数。此时的基本运算为加法。显然，当 n=1 时，$T(n)$=0； n>1 时，不失一般性，假设 $n=2^k$，其中 k 为某个正整数， 则

$$
\begin{aligned}
T(n) &= 2T(n/2)+1 \\
&= 2*[2*T(n/2^2)+1]+1 \\
&= 2^2*T(n/2^2)+2+1 \\
&= 2^2*[2*T(n/2^3)+1]+2+1 \\
&= 2^3*\ T(n/2^3)+4+2+1 \\
&= \cdots \\
&= 2^k*T(1)+...+4+2+1 \\
&= 2^k-1 \\
&= n-1
\end{aligned}
$$

因此算法的计算复杂度为$\Theta(n)$。

注： 算法也可按如下形式描述

Algorithm Sum (arr, i, j)

//arr 为待求和的数组，i 指向第一个元素，j 指向最后一个元素

Begin

m:=$\lfloor (i + j)/2 \rfloor$

if i=j **then return** arr[i]

else return Sum(arr, i, m) + **Sum**(arr, m+1, j)

end

3 假设 R 与 S 分别为具有 r 个与 s 个元素的有序表，其中 s ≤ r/log r．设计一个最坏情况下 O(r)时间的算法， 将 R 与 S 合并成一个有序表.

算法思想如下：

将 $b_1, b_2, ..., b_s$ 依次插入到有序表 R 中。对于元素 b_j，在 R 上使用二分查找方法，找到 b_j 所要插入的位置。假设 b_j 插入后的状态为

$$a_1, a_2, ..., a_k, b_j, a_{k+1}, a_{k+2}, ..., a_r,$$

由于 $b_j \le b_{j+1}$，则将 b_{j+1} 插入 R 时的工作可在有序子序列 $a_{k+1}, a_{k+2}, ..., a_r$ 上进行，从而对 S 中的每个元素在 R 上实施二分查找插入方法的序列长度始终不超过 r。根据二分查找算法的复杂度为 $O(\log r)$，得知归并 R 与 S 的算法复杂度为 $O(s \cdot \log r)$。由于 $s \le r / \log r$，得出算法复杂度为 $O(r)$。

算法形式化描述如下：

令 **Binsearch**(b_i , *Temp*)为一个函数过程，能够在有序表 *Temp* 上使用二分查找方法返回 b_i 应该被插入的位置 k，算法可形式化描述为：

Algorithm Merger (*R*, *S*)
Begin
1. *Temp* := *R*;
2. **for** *i* := 1 **to** *s* **do**
   2.1. *k* := **Binsearch**(b_i , *Temp*);
   2.2. 将 b_i 插入到 *R* 中，并更新 *R*；
   2.3. *Temp* := ($a_{k+1}, a_{k+2}, ..., a_r$);
   end of for;
3. **return**(*R*);

end

复杂度分析：算法的基本操作为元素之间的比较。由于 *Temp* 的长度始终不超过 r，因此步骤 2.1 总能在 $O(\log r)$ 时间内完成。由于 $s \le r / \log r$，从而步骤 2 的开销为 $s \cdot \log r \le (r / \log r) \cdot \log r \le r$，所以整个算法的复杂度为 $O(r)$。

4 给定长度为 n 的表，表中的所有元素已构成 k 个有序段， 设计一个计算复杂度为 O(n log k)的算法完成对表元素的排序。给出算法的形式化描述与复杂度分析。

解法一：

解:假定 k 个有序段依次为 L_1，L_2，...，L_k，其中每个 L_i 的长度为 l_i，$1 \le i \le k$，则 $l_1+l_2+...+l_k = n$。

算法思想为：将这 k 个有序段进行两两合并，生成$\lceil k/2 \rceil$个有序段；并连续使用这种有序段归并方法，将$\lceil k/2 \rceil$个有序段归并为$\lceil k/2^2 \rceil$个有序段，直到将最后所剩的两个有序段归并成最终的一个有序段。

令 $A=\{L_1, L_2, ..., L_k\}$，merge（B，C，A）为将集合 B 与 C 合并成有序集合 A 的过程，其复杂度为 O（$|B|+|C|$），即以 B 与 C 的所有元素个数为上界。

算法的形式化描述如下：

```
Algorithm Sort（A）
begin
    If k = 1 then return（A）
    else begin
    1. m := ⌈k / 2⌉;
    2. B := {L1, L2, ..., Lm};
    3. C := {Lm+1, Lm+2, ..., Lk};
    4. Sort(B);
    5. Sort(C);
    6. Merge(B, C, A);
      end;
end.
```

算法复杂度分析：

由于有序段个数为 k，从而递归的深度为 O（$\log k$）。而在每个递归层算法总要完成当前层所有子序列的两两归并，共计花费 O（n）次比较。所以算法的复杂度为 O（$n \log k$）。

解法二：

以 k 个有序段的首位元素建立堆（建立最小堆），在此堆上连续地进行删除操作以达到归并的目的。当堆顶元素被删除后，其所在的有序段增补一个次小元素入堆并完成堆化。

```
//Input：  k 个链表，记为 A[k]
//Output：  有序序列 B[n]
Begin
    For i:=1 to k do
        B[i]:= A[i].value
    Createheap(B,index) //index 记录最小堆中元素对应于 A 的索引
    While i != n do
    Begin

        If(A[index[0]].next != 0) then
            B[0] := A[index[0]].next.value
        Else:
            B[0] := B[k]
            Swap(index[0],index[k])
            k:=k-1
        k:=k-1
        AdjustHeap(B,0,k)
    end
end
```

时间复杂度分析：

假设 k 个有序段均为非降序排列的。

（1）取 k 个元素建立最小堆，这 k 个元素分别是 k 个有序段的第一个元素。建立最小堆的时间复杂度为 $O(k)$。

（2）此时，该最小堆的堆顶元素就是 k 个有序段中最小的那个元素，将它取出返回，时间复杂度 $O(1)$。

（3）若堆顶元素所在有序段链表不为空，则取下一个元素放到堆顶位置，进行堆调整，堆调整时间复杂度 $O(\log k)$。若为空，此子链表已经被合并完毕，则删除最小堆的堆顶元素，此时最小堆的结点数减小了 1 ，删除指定元素的时间复杂度 $O(\log k)$。

（4）重复步骤（2）、（3）$n-k$ 次。总的时间复杂度是 $O(k)+O(n\log k)$即 $O(n\log k)$。

习题 5.1 pp.135

2. a. 为一个分治算法编写伪代码，该算法同时求出一个 n 元素数组的最大元素和最小元素的值。

b. 假设 $n = 2^k$，为该算法的键值比较次数建立递推关系式并求解。

c. 请将该算法与同样问题的蛮力算法做一个比较。

a. 解：

算法思想：将数组 A 划分为大小大致相等的两个子数组；递归地对这两个子数组求最大元素和最小元素；将两个子数组的最大元素进行比较，返回数组 A 的最大元素，将其最小元素进行比较，返回数组 A 的最小元素；当 n 不超过 2 时，递归终止，此时最多通过一次比较就可得到最大元素与最小元素。

假设 $\max\{a, b\}$, $\min\{a, b\}$ 分别为通过一次比较产生出两者之间的最大者与最小者的标准过程。算法的伪代码如下：

Algorithm Max-Min (A, M, N)

//输入：n 个元素的数组 A

//输出：数组 A 的最大元素 M 和最小元素 N

begin

if $n = 1$ **then** $M := A[0]$ and $N := A[0]$

else if $n = 2$ **then** M=**max**$\{A[0], A[1]\}$ and $N :=$ **min**$\{A[0], A[1]\}$

else begin

$B := A[0..\lfloor n/2 \rfloor]$, $C := A[\lfloor n/2 \rfloor + 1..n-1]$;

Max-Min(B, M_0, N_0);

Max-Min(C, M_1, N_1);

$M :=$ **max**$\{M_0, M_1\}$;

$N :=$ **min**$\{N_0, N_1\}$;

end;

end.

b. 设 $T(n)$ 为算法求 n 个元素数组的最大元素和最小元素的基本运算次数，此时的基本运算为比较。当 n=2 时，$T(2)$=1，n=1 时，$T(1)$=0，则 $n \geq 2$ 时有：

$T(n) = 2T(n/2)+2$

$= 2*(2T(n/4)+2)+2$

$= \ldots \ldots$

$= 2^{k-1}T(n/(2^{k-1}))+2^{k-1}+ \ldots +2$

$=3*2^{k-1}-2$

依据假设 $n=2^k$，得出 $T(n)=3/2 \cdot n-2$。因此算法的时间复杂度为 $O(n)$。

c. 当 n 大于 1 时，用蛮力算法找到最大元素并取出需要 n-1 次比较，接着在剩下的 n-1 个元素中找出最小元素需要 n-2 次比较，总的比较次数为 $2n$-3 次，因此分治法在计算效率上比蛮力法要快。

习题 5.1 pp. 135

9. $A[0..n-1]$是一个 n 个不同实数构成的数组。如果 $i<j$，但是 $A[i]>A[j]$，则这 对元素（$A[i]$，$A[j]$）被称为一个倒置（inversion）。设计一个 $O(n\log n)$算法来计算数组中的倒置数量。

解：算法思想

对数组 A 实施合并排序，在排序的过程中统计元素倒置的个数。递归步骤为：

1. 将 A 划分为两个大小基本相等的子数组 B 与 C；
2. 分别递归地对 B 与 C 进行排序并计算各自元素的倒置数；
3. 对有序数组 B 与 C 进行归并，并统计两个有序数组归并时存在的元素倒置数；

当数组 A 的大小不超过 2 时，算法递归终止、直接排序并统计倒置个数。

伪代码描述如下：

```
Algroithm Inversion (A[0..n-1], a)
    //输入数组 A[0..n-1]
    //输出 n 个数的倒置数量 a
begin
    if n=1 then return (0)
          else begin
```

```
        B := A[0..⌊n/2⌋], C := A[⌊n/2⌋+1 .. n-1]   // 将 A 划分成子数组 B 与 C
        Inversion (B, b);   //对 B 进行排序并计算倒置数量 b
        Inversion (C, c);   //对 C 进行排序并计算倒置数量 c
        Merge (B, C, A, d); //将有序数组 B 与 C 合并成 A 并计算倒置数 d;
        a := b + c + d;
      end;
end.
```

其中 Merge(*B, C, A, d*)过程中倒置数的计算依据如下：*B* 与 *C* 中各自当前最小的元素进行比较，较小者移入 *A* 中，并对倒置数进行累加。 若较小者来自数组 *B*，则不产生倒置；若较小者来自数组 *C*，则将产生倒置，并且与此较小者相关的倒置数量为 *B* 中的还没有移入 *A* 的剩余元素个数。伪代码描述如下：

```
Procedure Merge (B[0..p-1], C[0..q-1], A[0..p+q-1], d)
// 将有序数组 B 与 C 合并成有序数组 A，并计算元素的倒置数 d.
begin
   i: =0, j:=0, k:=0; d:=0;
   while i<p and j<q do
        if B[i] ≤ C[j] then A[k] :=B[i], i:=i+1
                       else A[k] := C[j], j :=j+1, d := d + (p – i);
        k := k+1;
    end of while
    if i=p then // 数组 B 中的元素已经处理完毕，无需倒置追加
            copy C[j..q-1] to A[k..p+q-1]
        else // 数组 C 中的元素已经处理完毕，无需倒置追加
            copy B[i..p-1] to A[0..p+q-1];
end.
```

复杂度分析：

由于算法是在合并排序的过程中进行倒置数的计算，从而算法的计算复杂度与合并排序复杂度相同，即为 $O(n \log n)$。

习题 5.2 pp. 141

11. 螺钉和螺姆问题 假设我们有 *n* 个直径各不相同的螺钉以及 *n* 个相应的螺母。我们一次只能比较一对螺钉和螺母，来判断螺母是大于螺钉、小于螺钉还是正好适合螺钉。然而，我们不能拿两个螺母做比较，也不能拿两个螺钉做比较。我们的问题是找到每一对匹配的螺钉和螺母。为该问题设计一个算法，它的平均效率必须属于集合 (*n* log *n*) 。

算法思想：假设螺母和螺钉分别存放在数组 *Nut* 和 *Bolt* 中。运用快速排序的思想，在 *Nut* 中随机选择一个螺母 *u*，通过螺母与螺钉匹配的检测，对 *Bolt* 中的螺钉进行划分，生成螺钉的子集合 *L**、{*u**} 与 *R**，其中 *L**中的螺钉小于螺母 *u*，*R**中的螺钉大于螺母 *u*，而 *u**为与 *u* 成功匹配的螺钉。然后再用螺钉 *u** 对 *Nut* 中的螺母进行类同的划分，生成螺母的子集合 *L*、{*u*} 与 *R*。这样在整个 *Nut* 和 *Bolt* 上进行匹配的问题就可继续在 *L* 与 *L**、 *R* 与 *R** 上递归进行。当子集合中只剩下一个元素时递归终止，此时的螺母和螺钉一定互相匹配。

令 Partition(*Nut*，*u**) 与 Partition(*Bolt*，*u*)为上述的划分过程，并分别返回划分后的 *u* 与 *u**，以及所在的位置 *s*，算法的形式化描述如下：

```
Algorithm Pairing (Nut[0..n-1], Bolt[0..n-1])
//输入：n 个直径各不相同的螺母和 n 个相应的螺钉
//输出：每一对匹配的螺钉和螺母
Begin
    If n = 1 then return (Nut[0], Bolt[0]);
    else begin
            在 Nut 中随机选择一个螺母 u；
            (s, u*) := Partition(Bolt[0..n-1], u);
            (s, u) := Partition(Nut[0..n-1], u*);
            Pairing(Nut[0.. s-1], Bolt[0..s-1]);
            Pairing(Nut[s+1..n-1], Bolt[s+1..n-1]);
        end;
end.
```

时间复杂度分析：算法的基本操作为元素之间的比较。算法完全在快速排序的框架下进行，在递归前仅仅增加了一次划分，而每次划分的代价为 *O(n)*，在量级上与快速排序中的划分过程的代价相同，依据快速排序算法的平均复杂度为 *O*(*nlogn*) 的结果，得知上述算法的平均复杂度依然是 *O*(*nlogn*)。

5.3 节 P143-144 第 2 题和第 3 题

1. 设计一个分治算法来计算二叉树的层数（具体来说，对于空树和单节点树，该算法应该返回 0 和 1）。这个算法的效率类型是怎样的？

解：

算法思想：对于给定的二叉树 T，如果二叉树为空，则返回 0；若不为空，则递归地计算其左子树的层数和右子树的层数，取其中最大者加 1，即为此二叉树的层数。

伪代码实现：

Algorithm BinaryTree(T)

//输入：给定的二叉树 T

//输出：T 的层数

Begin

if $T = \emptyset$ **then return** 0;

else return max{**BinaryTree**(T_{left}), **BinaryTree**(T_{right})}+1;

end.

时间复杂度分析：求二叉树的高度需要遍历数的所有节点，所以该算法的时间复杂度为 $O(n)$。

习题 5.3　pp. 143

2. 下列算法试图计算一颗二叉树的叶子数。

算法 LeafCounter(T)

//递归计算二叉树的叶子数

//输入：一颗二叉树 T

//输出：T 的叶子树

if $T = \emptyset$ **return** 0

else return LeafCounter(T_{left})+**LeafCounter**(T_{rigth})

该算法正确吗？如果正确，请证明；否则，请改正。（page143）

解：当树为单节点树时，由题目所给的算法计算得该叶子节点数为 0，所以该算法不正确。对算法的改正如下：

算法 LeafCounter (T)

//输入：一颗二叉树 T

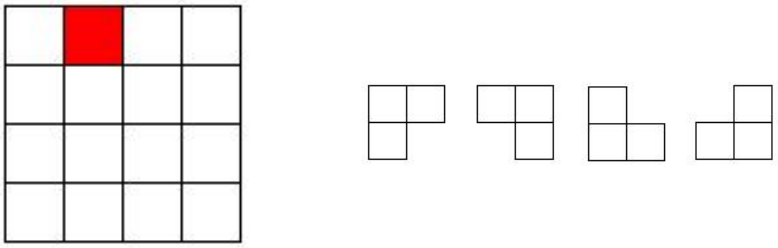
//输出：T 的叶子树

if $T = \emptyset$ **return** 0;

else if $T = 1$ **then return** 1;

else return LeafCounter(T_{left}) + **LeafCounter**(T_{rigth})

1 棋盘覆盖：在一个 $2^k \times 2^k$ 个方格组成的棋盘中，恰有一个方格与其它方格不同，称该方格为一特殊方格，且称该棋盘为一特殊棋盘。在棋盘覆盖问题中，要用图示的 4 种不同形态的 L 型骨牌覆盖给定的特殊棋盘上除特殊方格以外的所有方格，且任何 2 个 L 型骨牌不得重叠覆盖。



2 输油管道问题输油管道问题 某石油公司计划建造一条由东向西的主输油管道。该管道要穿过一个有 n 口油井的油田。从每口油井 都要有一条输油管道沿最短路经(或南或北)与主 管道相连。如果给定 n 口油井的位置,即它们的 x 坐 标（东西向）和 y 坐标（南北向）,应如何确定主 管道的最优位置,即使各油井到主管道之间的输油 管道长度总和最小的位置?证明可在线性时间内确 定主管道的最优位置。

3 二维最近点对问题:

给定平面 S 上 n 个点,找其中的一对点,使得在 n(n-1)/2 个点对中, 该点对的距离最小。

给定平面上 N 个点的坐标，找出距离最近的两个点（如图 2-8 所示）。

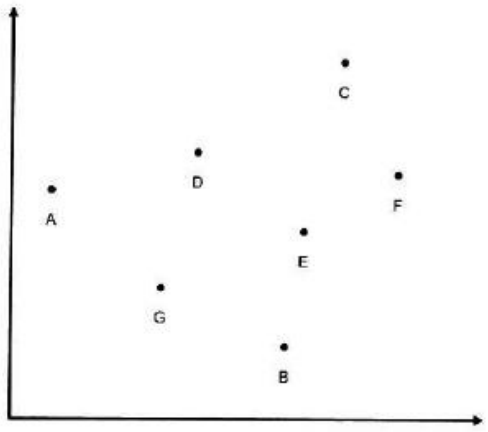


图 2-8　七个平面上的点

4 循环赛日程表

设计一个满足以下要求的比赛日程表：

(1)每个选手必须与其他 n-1 个选手各赛一次；

(2)每个选手一天只能赛一次；

(3)循环赛一共进行 n-1 天。

5 请用大整数乘法计算 1234*5678，并分析时间复杂度

解：

1234*5678

=（12*100+34）*（56*100+78）

=12*56*10000+（（12+34）*（56+78）-12*56-34*78）*100+34*78

=12*56*10000+（46*134-12*56-34*78）*100+34*78

12*56

=（1*10+2）*（5*10+6）

=1*5*100+（（1+2）*（5+6）-1*5-2*6）*10+2*6

=500+（33-5-12）*10+12

=500+160+12

=672

46*134

=（4*10+6）*（13*10+4）

=4*13*100+（（4+6）*（13+4）-4*13-6*4）*10+6*4

=5200+（170-52-24）*10+24

=5200+940+24

=6164

34*78

=（3*10+4）*（7*10+8）

=3*7*100+（（3+4）*（7+8）-3*7-4*8）*10+4*8

=2100+（105-21-32）*10+32

=2100+520+32

=2652

1234*5678

= 672*10000+（6164-672-2652）*100+2652

= 672*10000+2840*100+2652=7006652

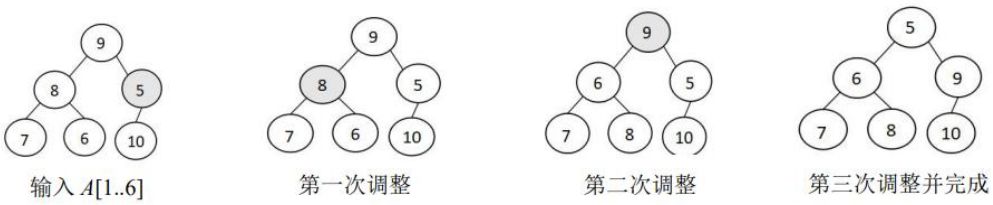
6 应用 Strassen 算法计算，当 n=2，停止递归，也就说 2*2 的矩阵用蛮力算法计算。思考当 n 不是 2^k 时，或者$A_{m*n} * B_{n*k}$矩阵乘法时,如何应用 Strassen 算法。

$$\begin{bmatrix} 1 & 0 & 2 & 1 \\ 4 & 1 & 1 & 0 \\ 0 & 1 & 3 & 0 \\ 5 & 0 & 2 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 4 \\ 2 & 0 & 1 & 1 \\ 1 & 3 & 5 & 0 \end{bmatrix}$$

第六章作业

变治作业题目（附加题 1-2，6.5 节 P186 第 4 题，附加 4）

附加 1，假定集合{9, 8, 5, 7, 6, 10}的元素依次存放在数组 A[1..6]中，试用图示给出线性建堆的执行过程，其中堆中任一结点均小于其子节点。



附加 2. 利用堆排序的算法思想，设计一个算法从 n 个元素的集合中选出第 k 个最小元素。

a) 给出算法思想的自然语言描述；

b) 已知线性建堆过程与堆顶元素的删除过程， 给出算法的形式化描述；

c) 分析算法的计算复杂度，并找出算法复杂度为线性时变量 k 应满足的条件。

解法一：

a) 1）利用堆排序的思想，先将 n 个元素构建成小顶堆，此时整个序列的最小元素就是堆顶元素；2）删除堆顶元素，并将剩余 n-1 个元素重新构造成小顶堆；3）重复第 2 步 k-1 次，此时的堆顶元素即为第 k 个小元素。

b) 算法的伪代码描述：

```
Algorithm FindKthNum(A[1..n-1])
//输入：n 个元素的数组 A
//输出：第 k 个最小元素
Begin
   BuildMinHeap(A)   //对数组 A 中的 n 个元素建立小顶堆
   for i :=1 to k-1 do:
       Delete(A, 0);      //删除堆顶元素
       DownAdjust(A, 0);  //从堆顶元素开始向下调整堆
   return A[0];
end
```

c) 线性建堆时间复杂度为 $O(n)$，一次删除堆顶元素并调整堆的时间复杂度为 $O(\log n)$，总时间复杂度为 $O(n+k*\log n)$，当总时间复杂度为线性时，k 应满足 $k*\log n \le n$，即 $k \le n/\log n$。

解法二：

a) 算法基本思想:我们可以利用大顶堆的特性，维护一个含 k 个元素的大顶堆，堆中的 k 个节点代表当前最小的 k 个元素，而堆顶显然是当前 k 个元素中的最大值。本题算法具体步骤如下：

1）从 n 个元素的集合中任意取出 k 个元素建立大顶堆；

2）将堆顶元素和剩余 n-k 个比较，如果比堆顶元素小则交换当前堆顶元素，调整堆；

3）直到剩余的 n-k 个元素比较完，则堆顶元素就是第 k 个最小元素；

b) 算法的伪代码描述：

```
Algorithm FindKthNum(A[1..n-1])
//输入：n 个元素的数组 A
//输出：第 k 个最小元素
Begin
  B := A[n-k+1..n-1]; //选取数组 A 的最后 k 个元素建立大顶堆
  BuildMaxHeap(B);
  for i ≤ n-k do:
      if A[i] ≤ B[0] then B[0] = A[i];
      DownAdjust(B, 0)//从堆顶元素开始向下调整堆
  return B[0];
end
```

c) 算法的时间复杂度分析：建堆的时间复杂度是 $O(k)$; 遍历剩余的数组元素的时间复杂度是 $O(n-k)$;每次调整堆的时间复杂度是 $O(\log k)$; 所以总的时间是 $O(k+(n-k)\log k)$。由于 $k \le n$, 从而当 $(n-k)\log k \le n$ 时，算法的计算复杂度为线性，即 $O(n)$。 由于$(n-k)\log k \le (n-k)\log n$, 因此当 $(n-k)\log n \le n$,

即 $k \ge n\left(1-\frac{1}{\log n}\right)$时，算法的计算复杂度为线性。

特别地，当 k=n 或 1 时，算法复杂度为线性。

3. 习题 6.5, pp.186 第四题

a. 应用霍纳法则计算下列多项式：

$$p(x) = 3x^4 - x^3 + 2x + 5 \text{ , } x = -2$$

利用霍纳法则的运算结果，求 $p(x)$除以 x+2 之后的商和余数。

解：

a. $p(x) = 3x^4 - x^3 + 2x + 5 = ((3x - 1)x^2 + 2)x + 5$

系数	3	-1	0	2	5
x = -2	3	-2*3 + (-1) = -7	-2*(-7) + 0 = 14	-2*14 + 2 = -26	-2*(-26) + 5 = 57

所以 $p(x) = 57$。

b. $p(x)$除以 $x+2$ 之后的商为 $3x^3 + (-7)x^2 + 14x + (-26)$，余数为 57。

附加 4 请用多种方法计算3^{499}，并分析算法策略和时间复杂度。

算法 1，减一策略：$3^{499} = 3^{498} * 3 = 3 * 3 * ...*3$，乘法执行次数 498 次，时间复杂度 O（n）

算法 2，减常因策略：$3^{499} = (3^{249})^2 * 3$，乘法执行次数 8+6=14 次，时间复杂度 $O(\log_2 n)$

算法 3，变治策略从左向右计算（霍纳法则变形）

499=（111110011）$_2$=$1 * 2^8 + 1 * 2^7 + 1 * 2^6 + 1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$=（（（（（（（（1*2+1）*2+1）*2+1）*2+1）*2+1）*2+0）*2+0）*2+1）*2+1

	1	1	1	1	1	0	0	1	1
X0=2	1	3	7	15	31	62	124	249	499

共执行乘法 8+6=14 时间复杂度 $O(\log_2 n)$

算法 4，变治策略从右到左计算

	1	1	1	1	1	0	0	1	1
2^i	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	1*2^8+243 =499	1*2^7+115 =243	1*2^6+51 =115	1*2^5+1 9=51	1*2^4+ 3=19	0+3=3	0+3=3	1* 2^1 +1=3	1

$$3^{499} = 3^{111110011} = 3^{1*2^8+1*2^7+1*2^6+1*2^5+1*2^4+0*2^3+0*2^2+1*2^1+0*2^0}$$

$$= 3^{256} * 3^{128} * 3^{64} * 3^{32} * 3^{16} * 3^1 * 3^0$$

$3^0 = 1$

$3^1 = 3$

$3^2 = (3^1)^2$

$3^4 = (3^2)^2$

$3^8 = (3^4)^2$

$3^{16} = (3^8)^2$

$3^{32} = (3^{16})^2$

$3^{64} = (3^{32})^2$

$3^{128} = (3^{64})^2$

$3^{256} = (3^{128})^2$　共执行乘法 8+6=14 次，时间复杂度 $O(\log_2 n)$

第八章动态规划作业

（习题 8.2 P229，1 题和 9 题；习题 8.3 P234，11 题；习题 8.4 P241 习题 1 和 7）

习题 8.2 P229

习题 8.2

1. a. 对于下列背包问题的实例，应用自底向上动态规划算法求解。

物　品	重　量	价值/美元
1	3	25
2	2	20
3	1	15
4	4	40
5	5	50

承重量 $W=6$

b. a 中的实例有多少个不同的最优子集？

c. 一般来说，如何从动态规划算法所生成的表中判断出背包问题的实例是不是具有不止一个最优子集？

d.请设计三种贪心算法，求解上述实例。

答案：

A)动态规划求解实例

件数\重量	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	25	25	25	25
2	0	0	20	25	25	45	45
3	0	15	20	35	40	45	60
4	0	15	20	35	40	55	60
5	0	15	20	35	40	45	65

B) W=6，可以有 3 个子集，子集{物品 3，物品 5}对应最优解 65；子集{物品 1，物品 2，物品 3}对应解 60；子集{物品 2 和物品 4}对应最优解 60；

C)动态规划所生成最优解唯一，但是子集不一定唯一，可能存在多种组合。

D)贪心一，价值优先，贪心选择价值当前最大物品组成子集，{物品 5，物品 3}，对应解为 65

贪心二，容量优先，贪心选择当前容量小的物品组成子集，{物品 3，物品 2，物品 1} ，对应解为 60

贪心三，性价比优先，贪心选择当前容量小的物品组成子集，{物品 3，物品 5} ，对应解为 65；或者{物品 3，物品 2，物品 1}，对应解为 60；或者{物品 3，物品 4}，对应解为 55。

物品	重量	价值	单位重量价值
1	3	25	25/3>8
2	2	20	10

3	1	15	15
4	4	40	10
5	5	50	10

9. 针对下面某一种著名的动态规划方法的应用写一个研究报告。

a. 求两个序列中最长的公共子序列。

b. 最优串编辑。

c. 多边形的最小三角剖分。

王晓东教材公共子串，凸多边形的最优三角剖分，9 题选做题目

习题 8.3P234

11．矩阵连乘　考虑如何使得在计算 n 个矩阵的乘积 $A_1\ A_2\cdots A_n$ 时，总的乘法次数最小，这些矩阵的维度分别为 $d_0\times d_1$，$d_1\times d_2$，…，$d_{n-1}\times d_n$。假设所有两个矩阵的中间乘积都使用蛮力算法(基于定义)计算。

a．给出一个三个矩阵连乘的例子，当分别用 $(A_1A_2)A_3$ 和 $A_1(A_2A_3)$ 计算时，它们的乘法次数至少相差 1 000 倍。

b．有多少种不同的方法来计算 n 个矩阵的连乘乘积？

c．设计一个求 n 个矩阵乘法最优次数的动态规划算法。

王晓东教材矩阵连乘问题，思考最优二叉查找树和凸多边形的最有三角剖分的关系，11 题选做题目

P241

习题 8.4

1. 对由下列邻接矩阵定义的有向图应用 Warshall 算法，求它的传递闭包。

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

解答

（1）W0=

0	1	0	0
0	0	1	0
0	0	0	1
0	0	0	0

a 为桥梁，没有更新，W1=W0

W1=

0	1	0	0
0	0	1	0
0	0	0	1
0	0	0	0

b 为桥梁，从 W1 更新得到 W2，更新如下：

a---b--c

W2=

0	1	1	0
0	0	1	0
0	0	0	1
0	0	0	0

c 为桥梁，更新 W2 得到 W3，更行如下

a---c---d,　　b---c---d

W3=

0	1	1	1
0	0	1	1
0	0	0	1
0	0	0	0

d 为桥梁，没有更新 W4=W3

W4=

0	1	1	1
0	0	1	1
0	0	0	1
0	0	0	0

7. 对于下面具有权重矩阵的有向图，求解完全最短路径问题。

$$\begin{bmatrix} 0 & 2 & \infty & 1 & 8 \\ 6 & 0 & 3 & 2 & \infty \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \infty & \infty & \infty & 0 \end{bmatrix}$$

解答：

D0

0	2	Inf	1	8
6	0	3	2	Inf
Inf	Inf	0	4	Inf
Inf	Inf	2	0	3
3	Inf	Inf	Inf	0

（a 为桥梁更新距离矩阵） 从 D0 得到 D1

b--6--a--8--e: 14

e--3--a--2--b: 5

e--3--a--1--d: 4

D1

0	2	Inf	1	8
6	0	3	2	14
Inf	Inf	0	4	Inf
Inf	Inf	2	0	3
3	5	Inf	4	0

（b 为桥梁更新距离矩阵） 从 D1 得到 D2

a--2--b--3--c: 5

e--5--b--3--c: 8

D2

0	2	5	1	8
6	0	3	2	14
Inf	Inf	0	4	Inf
Inf	Inf	2	0	3
3	5	8	4	0

c 为桥梁更新距离矩阵没有变化 从 D2 得到 D3 ， D3=D2

D3

0	2	5	1	8
6	0	3	2	14
Inf	Inf	0	4	Inf
Inf	Inf	2	0	3
3	5	8	4	0

d 为桥梁更新距离矩阵没有变化 从 D3 得到 D4 ， 更新如下

a--1--d--2--c: 3

a--1--d--3--e: 4

b--2--d--3--e: 5

c--4--d--3--e: 7

e--5--d--2--c: 6

D4

0	2	3	1	4
6	0	3	2	5
Inf	Inf	0	4	7
Inf	Inf	2	0	3
3	5	6	4	0

e 为桥梁更新距离矩阵从 D3 得到 D4 ， 更新如下

c--7--e--3--a: 10

c--7--e--5--b: 12

d--3--e--3--a: 6

d--3--e--5--b: 8

D5

0	2	3	1	4
6	0	3	2	5
10	12	0	4	7
6	8	2	0	3
3	5	6	4	0

选做题目（11，12，11）

11. *Maximum square submatrix* Given an $m \times n$ boolean matrix B, find its largest square submatrix whose elements are all zeros. Design a dynamic programming algorithm and indicate its time efficiency. (The algorithm may be useful for, say, finding the largest free square area on a computer screen or for selecting a construction site.)

11. *Jack Straws* In the game of Jack Straws, a number of plastic or wooden "straws" are dumped on the table and players try to remove them one by one without disturbing the other straws. Here, we are only concerned with whether various pairs of straws are connected by a path of touching straws. Given a list of the endpoints for $n > 1$ straws (as if they were dumped on a large piece of graph paper), determine all the pairs of straws that are connected. Note that touching is connecting, but also that two straws can be connected indirectly via other connected straws. [1994 East-Central Regionals of the ACM International Collegiate Programming Contest]

12. *World Series odds* Consider two teams, A and B, playing a series of games until one of the teams wins n games. Assume that the probability of A winning a game is the same for each game and equal to p, and the probability of A losing a game is $q = 1 - p$. (Hence, there are no ties.) Let $P(i, j)$ be the probability of A winning the series if A needs i more games to win the series and B needs j more games to win the series.

a. Set up a recurrence relation for $P(i, j)$ that can be used by a dynamic programming algorithm.

b. Find the probability of team A winning a seven-game series if the probability of it winning a game is 0.4.

c. Write pseudocode of the dynamic programming algorithm for solving this problem and determine its time and space efficiencies.

第九章贪心算法作业题目

（习题 9.1，pp.250 ，习题 1；习题 9.3（P260）习题 2；习题 9.4，pp.264 ）

习题 9.1，pp.250

1. 为找零问题写一个贪婪算法的伪代码，它以金额 n 和硬币的面额 $d_1>d_2>\cdots>d_m$ 作为输入。该算法的时间效率类型是怎样的？

伪代码:

贪心选择最大面额去分解 n，希望所用找零硬币数量最小，但是贪心算法无法保证得到最优解。

1 选择面额最大的币值 dm 分解 n，直到所剩余金额 n1 小于 dm

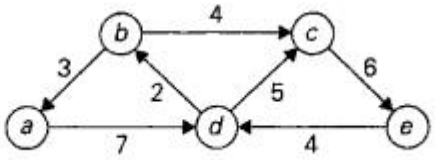
2 选择小于剩余金额 n1 的最大币值，继续分解，直到剩余面额小于 d2 且大于 d1，或者恰好分解完结束

3 剩余面额小于 d2 且大于 d1，用 d1 分解，直到分解完成。

习题 9.3（P260）

2. 解下面这个单起点最短路径问题的实例，以顶点 a 作为起点。

a.



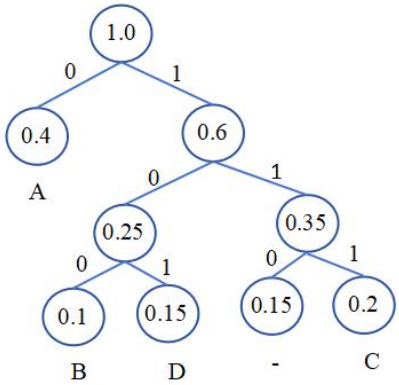
迪杰斯特拉算法运算过程如下

序号	S	u	b	c	d	e
0	{a}		Inf	Inf	7	Inf
1	{a,d}	d	7+2=9	7+5=12	---	inf
2	{a,b,d}	b	---	12	---	inf
3	{a,b,c,d}	c	---	---	---	12+6=18
4	{a,b,c,d,e}	e	---	---	---	---

习题 9.4，pp.264

1. a. 对下面的数据构造一套哈夫曼编码：

字符	A	B	C	D	-
出现概率	0.4	0.1	0.2	0.15	0.15



A: 0　　B:100　　C:111　　D:101　　-:110

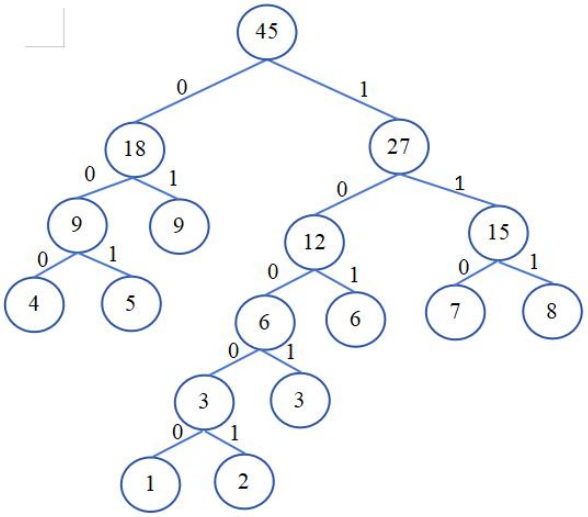
b. 用 a 中的编码对文本 ABACABAD 进行编码。

ABACABAD 编码: 0100011101000101

c, 对于 100010111001010 用 a 中的编码进行解码。

100010111001010 解码：BAD-ADA

10. 猜底牌 设计一种策略，使在下面的游戏中，期望提问的次数达到最小([Gar94])。有一副纸牌，是由 1 张 A，2 张 2，3 张 3，…，9 张 9 组成的，一共包含 45 张牌。有人从这副洗过的牌中抽出一张牌，问一连串可以回答是或否的问题来确定这张牌的点数。



第一个问题，是否是 9，如果是，牌为 9 结束；如果不是进入第二个问题

第二个问题，是否是 45678，如果不是，进入第三个问题；如果是，继续追问比 6 大？如果是，则问是否是 7，是牌为 7 结束；如果不是，牌为 8 结束；否则不

是，则问是否是 6，是牌为 6 结束；如果不是，则问是否是 4，是牌为 5 结束；否则牌为 5 结束；
第三个问题，是否是 3，如果不是进入第 4 个问题，否则牌为 3 结束
第四个问题，是否是 2，如果是，牌为 2 结束，否则牌为 1 结束

备用图