北京大学信息科学技术学院 2007-2008 学年第一学期

《数据结构与算法实习》期末考试参考答案——张铭编写

一、填空（45 分），空缺的可能是 0 至多条语句，也可能是表达式

1. （**10** 分，各 **5** 分）

/* 填空 1 */ ch != '/' && ch != '>';

/* 填空 2 */ BuildTree(fin, *(parent.children()));

2. （**4** 分）

/* 填空 3 */ assert(Function(A) == 10);

/* 填空 4 */ assert(Function(B) == 180);

3. /* 填空 5 */ （**14** 分，前 **4** 个各 **2** 分，后两个各 **3** 分）

```
X[i] = 1;
cw += W[i];
cp += P[i];                      // P 是存储各物品价值的数组
backtrack(i+1, X, W, P, M);      // 前进，搜索
cw -= W[i];                      // 回溯, 消除标记
cp -= P[i];
```

/* 填空 6 */ （**6** 分）

```
b += P[i] * cr / W[i];
```

4. /* 填空 7 */ （**4** 分）

```
return(S);
```

/* 填空 8 */ （**7** 分）

```
S:=S*(m1*2n+(m1+m2+m3)*2n/2+m3);
```

用移位的方法也可以：符号位 * (m1 左移 n 位+ m3 + (m1+m2+m3)左移 n/2 位)

二、 辨析题（35 分）

1. （10 分）请分析“一、4 大整数乘法”算法的时间代价。

设 X 和 Y 都是 n 位的二进制整数，现在要计算它们的乘积 XY。我们可以用小学所学的方法来设计一个计算乘积 XY 的算法，但是这样做计算步骤太多，显得效率较低。如果将每 2 个 1 位数的乘法或加法看作一步运算，那么这种方法要作 $O(n^2)$步运算才能求出乘积 XY。下面我们用分治法来设计一个更有效的大整数乘积算法。

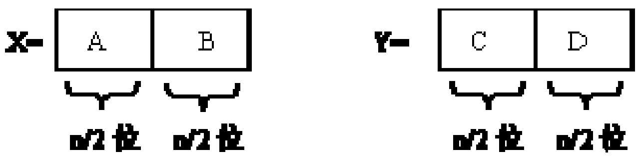


图 大整数 X 和 Y 的分段

我们将 n 位的二进制整数 X 和 Y 各分为 2 段，每段的长为 n/2 位(为简单起见，假设 n 是 2 的幂)，如图所示。

由此，$X=A2^{n/2}+B$ ，$Y=C2^{n/2}+D$。这样，X 和 Y 的乘积为：

$$XY=(A2^{n/2}+B)(C2^{n/2}+D)=AC2^{n}+(AD+CB)2^{n/2}+BD \qquad (1)$$

如果按式(1)计算 XY，则我们必须进行 4 次 n/2 位整数的乘法(AC，AD，BC 和 BD)，以及 3 次不超过 n 位的整数加法(分别对应于式(1)中的加号)，此外还要做 2 次移位(分别对应于式(1)中乘 2^n 和乘 $2^{n/2}$)。所有这些加法和移位共用 $O(n)$步运算。设 T(n)是 2 个 n 位整数相乘所需的运算总数，则由式(1)，我们有:

$$\begin{cases} T(1)=1 \\ T(n)=4T(n/2)+O(n) \end{cases} \qquad (2)$$

由此可得 $T(n)=O(n^2)$。因此，用(1)式来计算 X 和 Y 的乘积并不比小学生的方法更有效。要想改进算法的计算复杂性，必须减少乘法次数。为此我们把 XY 写成另一种形式:

$XY=AC2^n+[(A-B)(D-C)+AC+BD]2^{n/2}+BD$ (3)

虽然，式(3)看起来比式(1)复杂些，但它仅需做 3 次 n/2 位整数的乘法(AC，BD 和(A-B)(D-C))，6 次加、减法和 2 次移位。由此可得:

$$\begin{cases} T(1)=1 \\ T(n)=3T(n/2)+cn \end{cases} \qquad (4)$$

用解递归方程的套用公式法马上可得其解为 $T(n)=O(n^{\log 3})=O(n^{1.59})$。

2. （6 分）某厂生产 A，B，C 三种产品，每件产品消耗的原料和设备台时如下表所示：

产品	*A*	*B*	*C*	资源数量
原料单耗 机时单耗	2 2.5	3 3	5 6	2000 2600
利润	10	14	20	

另外，要求三种产品总产量不低于 65 件，*A* 的产量不高于 *B* 的产量。

试制定使总利润最大的代数模型（不需要画图，更不需要求解具体的最优值）。

用 x_1, x_2, x_3 分别表示 A、B、C 三种产品生产的件数，该厂追求的目标是获取最高利润，用数学表达式表示为：

$$\max f = 10x_1 + 14x_2 + 20x_3$$

由于生产甲、乙产品的件数要受到生产能力的约束，即

原料约束：$2x_1 + 3x_2 + 5x_3 \le 2000$,

机时约束：$2.5x_1 + 3x_2 + 5x_3 \le 2600$,

总量约束：$x_1 + x_2 + x_3 \le 65$,

AB 产量约束：$x_1 \le x_2$

非负约束： $x_1, x_2, x_3 \ge 0$.

这样，该生产计划问题就归结为如下数学模型：

$$\max f = 10x_1 + 14x_2 + 20x_3 \quad (1\text{分})$$

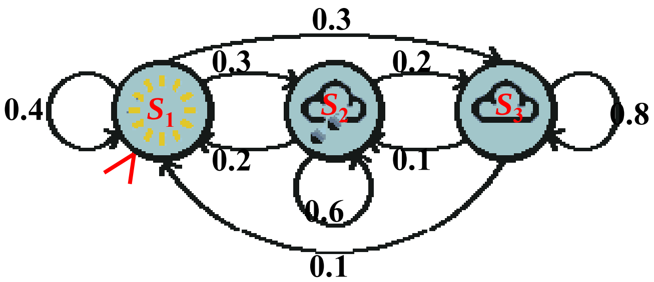
$$\begin{cases} 2x_1 + 3x_2 + 5x_3 \le 2000 \quad (1\text{分}) \\ 2.5x_1 + 3x_2 + 5x_3 \le 2600 \quad (1\text{分}) \\ x_1 + x_2 + x_3 \le 65 \quad (1\text{分}) \\ x_1 \le x_2 \quad (1\text{分}) \end{cases}$$

$$x_1, x_2, x_3 \ge 0 \quad (1\text{分})$$

3. 10 分，**每一个条件 0.5 分**

输入条件	有效等价类	无效等价类
是否三角形的三条边	**（A>0），（1）** **（B>0），（2）** **（C>0），（3）** **（A+B>C），（4）** **（B+C>A），（5）** **（A+C>B），（6）**	**（A≤0），（7）** **（B≤0），（8）** **（C≤0），（9）** **（A+B≤C），（10）** **（B+C≤A），（11）** **（A+C≤B），（12）**
是否等腰三角形	**（A=B），（13）** **（B=C），（14）** **（C=A），（15）**	**（A≠B）and（B≠C）and（C≠A）（16）**
是否等边三角形	**（A=B）and（B=C）and（C=A）（17）**	**（A≠B），（18）** **（B≠C），（19）** **（C≠A），（20）**

4. （9 分）



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

$$A = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

第一天天气“天晴”，接下来 4 天天气为“天晴—下雨—下雨—多云”的概率是多少？

P(O | Moel) = P(S1,S1, S2,S2,S3 | Model)

= P[1]• P[S1|S1] • P[S2|S1] • P[S2|S2] • P[S3|S2]

$= \pi_1 \bullet a_{11} \bullet a_{12} \bullet a_{22} \bullet a_{23}$

= 1 • 0.4 • 0.3 • 0.6• 0.2

$= 1.44 \times 10^{-2}$

第 2 种解答，给 7 分（扣 3 分）

P(O | Moel) = P(S1, S2,S2,S3 | Model)

= P[1]• P[S2|S1] • P[S2|S2] • P[S3|S2]

$= \pi_1 \bullet a_{12} \bullet a_{22} \bullet a_{23}$

= 1 • 0.3 • 0.6• 0.2

$= 3.6 \times 10^{-2}$

三、算法设计题（40 分）

1. （20 分）

判分标准：算法思想 3 分，注释 2 分，完整的程序 12 分，时间代价分析 3 分

题意：给出正整数 n 个数列，每个数列有 m 个数。求出出现次数第二多的数。
保证每个整数在一个数列中至多出现一次。

（1）用贪心法解答。

数据结构定义一个长为 10001 的 int 类型数组和几个变量。

用一个初值全零的数组统计每个号码出现的次数。再遍历一次该数组，找出出现次数第二多的。

算法核心代码如下：

```
#include "iostream"
#include "string.h"
using namespace std;
#define MAX 10001
int p[MAX], n, m, i, j, k, max1, max2;

int main() {
       while (cin>>n>>m, n) {
          memset(p, 0, 4 * MAX);             // 计数记录初始化为空
          for (i = 0; i < n; ++i) {          // 输入
              for (j = 0; j < m; ++j) {
                  cin>>k;
                  ++p[k];
              }
          }
          max1 = max2 = 0; // 最大值用max1记录,次大值用max2记录
          for (i = 1; i <= 10000; ++i) {
              if (p[i] > max1) {
                       max2 = max1;
                       max1 = p[i];
              }
              else  if (p[i] > max2)
                       max2 = p[i];

          }
          for (i = 1; i <= 10000; ++i) // 枚举,如果是次大值,输出序号
              if (p[i] == max2)
                 cout<<i<<" ";
          cout<<endl;
    }
    return 0;
}
```

（2）时间代价

输入并计数，代价为 O(M×N)。寻找出现次数第二多的，为 10000。核心代价为 O(M×N)。

2. 20 分

判分标准：算法思想 4 分，递推方程 6 分，伪码 8 分，注释 2 分

假设 m[i][j]表示在前 j 个村庄建立 i 个邮局的最小距离和，m[p][n]就是最后的结果。

设 w[i][j]为从第 i 到第 j 个村庄间设立一个邮局需要的最小距离，

$m[1][j] = w[1][j]$ $(1 <= j)$

$m[i][j] = \min(m[i-1][k] + w[k+1][j])$

其中 $i-1 <= k <= j-1$

即找出一个划分点 k，第 i 个邮局安排在 k+1 与 j 之间，从第 1 到第 k 个村庄安排 i-1 个邮局，总的距离最少。