# 北京大学信息科学技术学院 2005-2006 学年

# 第一学期本科生期末考试试卷

考试科目：数据结构与算法实习　考试时间：2006 年 1 月 2 日

_____专业_____级_____班　考场 一教 101　室

姓名_____　学号_____　主讲教师 张铭

| 题号 | 一 | 二 | 三 | 四 | 五 | 总分 |
|---|---|---|---|---|---|---|
| 得分 | | | | | | |

我保证严格遵循考场纪律，诚实地用自己所掌握的知识和能力独立答卷，不抄袭或者协助抄袭。

**诚实考试宣言保证人：**_____

不在“诚实考试宣言”之后签名的试卷，计零分或上报学校处理！

**第四、五大题要写出算法思想，注释，以及算法时间空间效率分析。**

## 一、数学建模（20 分），不用写算法

生产计划问题：某工厂生产甲、乙两种产品，甲产品每生产一件需消耗黄铜 2kg、3 个工作日、两个外协件，每件可获利润 60 元；乙产品每生产一件需消耗黄铜 4kg、1 个工作日、不需要外协件，每件可获利润 30 元，该厂每月可供生产用的黄铜 320kg，总工作日 180 个，外协件 100 个。问怎样安排生产才能使工厂的利润最高？

## 二、算法填空（20 分）

在分析算法的计算复杂性时，通常都将加法和乘法运算当作基本运算来处理，即将执行一次加法或乘法运算所需的计算时间当作一个仅取决于计算机硬件处理速度的常数。这个假定仅当整数比较小，计算机硬件能直接表示和处理时才是合理的。若要精确地表示大整数并在计算结果中要求精确地得到所有位数上的数字，就需要编写算法完成大整数的算术运算。

（1） 下面是两个 n 位大整数（用自定义的 LongInt 表示）的乘法运算算法。请填充算法空白处的语句，使其成为完整的算法，可能需要填写 0 至多条语句。

（2） 请分析本算法的时间代价。

//X 和 Y 为小于 $2^n$ 的二进制整数，返回结果为 X 和 Y 的乘积 XY

```
LongInt MULT(X，Y，n) {
  LongInt S = SIGN(X)*SIGN(Y);          // S 为 X 和 Y 的符号乘积
  LongInt X = ABS(X);
  LongInt Y = ABS(Y);                   // X 和 Y 分别取绝对值
  if (n==1) {
    if (X==1) && (Y==1)
          // ?1
    else return(0)
   }
  else {
    LongInt A = X 的左边 n/2 位;
    LongInt B = X 的右边 n/2 位;
    LongInt C = Y 的左边 n/2 位;
    LongInt D = Y 的右边 n/2 位;
    LongInt ml = MULT(A,C,n/2);
    LongInt m2 = MULT(A-B,D-C,n/2);
    LongInt m3 = MULT(B,D,n/2);
     // ?2
    return(S);
  }
}
```

## 三、算法辨析（25 分）

二叉搜索堆（Binary Search Heap, 简称 BSH 或者 Treap，也称“树堆”）的内部结点由一个标签 label 和权值 priority 组成，并且同时满足二叉搜索树和堆的性质，即(1) 每个结点的左子树中所有结点的 label 小于该结点的 label，而右子树中所有结点的 label 大于该结点的 label；(2) 每个结点的权值都小于其父结点的权值。

往一个二叉检索堆插入新结点的方法是：(1)如果新插入的结点的权大于根结点的权，就将新结点作为新树的根，并将原树作为新树根结点的左子树。(2)否则，就从树的最右下的结点（即中序周游下的最后一个结点，也是上一次插入的结点）往上搜第一个比新结点权大的结点 x。将 x 的右子树设为新结点的左子树，将新结点作为 x 的右子女。

请判断下述二叉搜索堆的构建算法是否正确。

（1） 如果正确，请通过演示该算法运行步骤，给出由 7 个标签/权值对{ a/3，b/6，c/4，d/7，e/2，f/5，g/1 }一步步构建成的二叉搜索堆的状态变化过程。

（2） 如果不正确，请在原题中指出错误之处，并给出改正后的结果。请不要撇开原算法，自己重写一套。然后给出由 7 个标签权值对{ a/3，b/6，c/4，d/7，e/2，f/5，g/1 }一步步构建成的二叉搜索堆的状态变化过程。

```
struct Node   {                  //二叉搜索堆结点定义
        string label;            //记录该结点的标号
        int priority;            //该结点的权
        Node *left;              //指向该结点的左子女
```

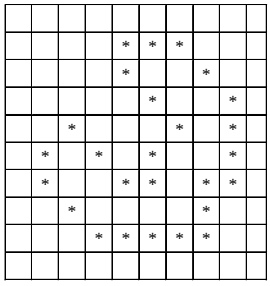
```
        Node *right;            //指向该结点的右子女
        Node * parent;          //指向该结点的父结点
};
// 构建 n 个结点的 BSH。list 为存放 n 个结点的数组，
// 且已经按 label 从小到大排完序。函数最后返回 BSH 的根结点指针 root
Node * buildTreap( Node *list, int n)  {
        Node * root = &list[0]; //list 第一个结点地址赋给根结点
        Node * pNew;            //pNew 指向每次要插入的新结点
        Node * p;               //p 指向树在中序周游下的最后一个结点
        p = root; //刚开始时，p 和 root 指向同一个结点，即 list[0]
        for(int i = 1; i <= n; i ++)  {
                pNew = & list[i];           //指向要插入的结点
                if( root ->priority < pNew->priority )//新插入结点比根结点权值大
                {
                        pNew->right = root;
                        root->parent = pNew;
                        root = pNew;
                }
                else  {
                  //从树的最右下角开始往上寻找第一个比 pNew 权大的结点
                        while( p->priority > pNew->priority )
                                p = p->parent;
                        pNew->left = p->right;//p 的右子树成为新结点的左子树
                        if(p->right != NULL )
                                p->right->parent = pNew;
                        p->left = pNew;
                        pNew->parent = p;
                }
                p = pNew;
        }
        return root; //返回根结点指针
}
```

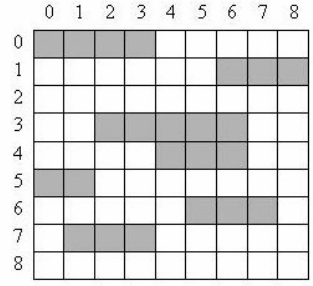
## 四、算法设计（40 分），只要求写出完整函数段

1. （20 分）有若干个大小有限制的背包，这些包的总容量相同。给定若干物品的质量总和，问这些包能否将这些物品全部装走？怎么装走？例如，三个背包容量都为 6。7 个物品质量分别为 {5，3，3，2，2，2，1}，解答为{(5+1)，(3+3)，(2+2+2)}。请编写算法，给出具体的解决方案。
2. （20 分）计算围棋的“目”。围棋中最后判断胜负时，要比较“目”的大小。如果一方完全围住了一些棋盘，那么按照日式计算法，这些棋盘中的空格算作他所占有的“目”。假设下图中模拟的棋盘，由“*”所代表的棋子连成线就围成了一个闭合的空间（与边界围的不算）。例如下图“*”围住了 15 个空格。因此“目”为 15。请编写算法计算其中“*”组成的闭合曲线所围住的空格数数，棋盘中数据通过数组传入。

## 五、程序设计（45 分），要求写出完整的可执行程序

1. （20 分）在穿墙魔术魔术中，魔术师要在一个事先设定好的场景下穿越好几堵墙。魔术师的能力被设定成至多穿越 k 堵墙。在右下图场景中（这个图是从上往下看得到的结果），墙被布置在一个网格区域上，并且每个网格单元至多只能属于一堵墙。在场景中，墙的厚度都是 1，但是宽度不确定。观众可以要求魔术师需要穿过网格中的任意一列。如果在这一列上他面对墙的数目多于 k，他的表演就会失败，否则表演可以成功。比如，在下图的场景当中，如果魔术师的 k = 3，那么除了第 6 列以外，其他列他都可以穿过。给定场景和魔术师的能力值 k，你需要计算的是，要至少去掉多少堵墙，魔术师才能成功穿越任何列。






【输入数据】

输入数据的第一行是两个整数 n（1 <= n <= 100，表示墙的数目）和 k（0 <= k <= 100，表示魔术师的能力值）。在第一行的后面是 n 行，每行上有两个坐标对(x, y)（x, y 是整数而且满足 0 <= x, y <= 100），每个坐标对表示墙的一个端点。在这里我们认为网格左上单元的坐标是(0, 0)，而且所有的墙都平行于 x 方向。

下面第二个输入样例对应的就是上图中的场景。

【输出数据】

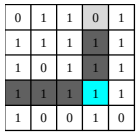
在输出一个整数，表示要至少去掉多少堵墙，魔术师才能成功穿越任何列。

【样例输入】

```
7 3
0 0 3 0
6 1 8 1
2 3 6 3
4 4 6 4
0 5 1 5
5 6 7 6
1 7 3 7
```

【样例输出】

```
1
```




2. （25 分）一个 n*n 的矩阵 A，其元素只能取值 0 或 1。设计一个程序求出最大的 m*m 全 1 子矩阵（子矩阵中元素全部为 1），m<=n。例如，右上图最大的全 1 子矩阵为 3*3 矩阵。