# 华南农业大学期末考试试卷（A 卷）

**2010 学年第一学期**　　**考试科目：** 算法分析与设计

**考试类型：（闭卷）考试**　　**考试时间：** **120** **分钟**

学号＿＿＿＿＿＿＿＿姓名＿＿＿＿＿＿年级专业＿＿＿＿＿＿＿＿＿＿＿＿

装

| 题号 | 一 | 二 | 三 | 四 | 总分 |
|---|---|---|---|---|---|
| 得分 | | | | | |
| 评阅人 | | | | | |

**注意：所有答案请写在答卷上，写在试卷上不得分。**

订

**一、选择题**（本大题共 10 小题，每小题 2 分，共 20 分）

| 得分 | |
|---|---|

线

1、以下有关NP完全性理论的相关描述，正确的是（　　）。

（A）P 问题都是 NP 问题

（B）NP 问题指的是不能够在多项式时间内求解的问题

（C）P = NP

（D）0-1 背包问题属于 P 问题

2、
```
void hanoi(int n, int a, int b, int c)
  {
      if (n > 0)
      {
          hanoi(n-1, a, c, b);
          move(a,b);
          hanoi(n-1, c, b, a);
      }
  }
```

上述算法的时间复杂度为（　　）。

（A）O （$2^n$） （B）O （nlog n） （C）Θ （n!） （D）Θ（$n^n$）

3、在对问题的解空间树进行搜索的方法中，一个活结点最多有一次机会成为活结点的是（　　）。

（A）回溯法

（B）分支限界法

（C）回溯法和分支限界法

（D）回溯法求解子集树问题

4、下列算法中通常以自底向上的方式求解最优解的是（　　）。

（A）备忘录法

（B）动态规划法

（C）贪心法

（D）回溯法

5、蒙特卡罗算法是（　　）的一种。

（A）概率算法　（B）分支界限算法　（C）贪心算法　（D）回溯算法

6、有 n 个独立的作业 $\{1, 2, .., n\}$，由 m 台相同的机器进行加工处理。作业 i 所需的处理时间为 ti。现约定，任何作业可以在任何一台机器上加工处理，但未完工前不允许中断处理。任何作业不能拆分成更小的作业。多机调度问题要求给出一种作业调度方案，使所给的 n 个作业在尽可能短的时间内由 m 台机器加工处理完成（$n > m$）。对于多级调度问题，使用以下哪种贪心策略比较合适（　　）。

（A） 作业从小到大依次分配给空闲的机器

（B） 作业从大到小依次分配给空闲的机器

（C） 每个机器分配一样的作业数

（D） 使用以上几种贪心策略都能找到最优解，所以都合适

7、考虑背包问题：n=6，物品重量W=(1,5,2,3,6,1），价值 P=(15,59,21,30,60,5)，背包载重量C＝10。能放进背包的物品价值最大为（　）。

（A）101　（B）110　（C）115　（D）120

8、分派问题一般陈述如下：给n 个人分派n 件工作，把工作j 分派给第i 个人的成本为cost（i，j），1≤i，j≤n，要求在给每个人分派一件工作的情况下使得总成本最小。此问题的解可表示成n 元组（X1，…，Xn），其中Xi 是给第i 个人分配的工作号，且Xi≠Xj（i≠j）。此解空间的状态空间树被称为（　）。

（A）排列树

（B）子集树

（C）宽度优先生成树

（D）深度优先生成树

9、当输入规模为n时，算法增长率最快的是（　　）。

（A）n!　　（B）$10\log_2 n$　　（C）$2^n$　（D）$3n^5$

10、有 9 个村庄，其坐标位置如下表所示：

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| x（i） | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| y（i） | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

现在要盖一所邮局为这 9 个村庄服务，请问邮局应该盖在（　　）才能使到邮局到这 9 个村庄的总距离和最短。

（A）（4.5，0）

（B）（4.5，4.5）

（C）（5，5）

（D）（5，0）

## 二、应用题（本大题共 5 小题，每小题 6 分，共 30 分）

| 得分 | |
|---|---|

1、对于以下程序段，分析其最坏时间复杂度。

```
int Binary(int n)
{
    int count = 1;
    while (n>1)
    {
        count = count +1;
        n = n/2;
    }
    return count;
}
```

2、请分析使用分治法求解最接近点对问题其最坏时间复杂度如何为O（nlogn）。

3、有n个集装箱要装上一艘载重量为c的轮船。其中集装箱i的重量为Wi，集装箱不可分割。最优装载问题要求在装载体积不受限制的情况下，将尽可能多的集装箱装上轮船。请写出使用贪心算法求解最优装载问题所用的贪心策略。

4、请列举3个会影响回溯搜索算法时间效率的因素。

装

5、使用动态规划算法求解矩阵连乘问题，令m[i][j]为计算矩阵A[i:j]所需的最少乘法次数，请写出m[i][j]的递归式子，并说明计算过程中求解所有m[i][j]的先后次序。

说明：A[i:j]表示从第i个矩阵开始连续到第j个矩阵结束。

订

1.5CM

**三、程序填空题**（本大题共 10 空格，每空 2 分，共 20 分。注意：每个空格只填一个语句）

| 得分 | |
|---|---|

线

1、使用回溯法求解 n 皇后问题，x[t]用于存储第 t 个皇后放置的列号，可直接使用求绝对值函数 int abs(int x)。

```
bool Place(int k)
{
   for (int j=1;j<k;j++)
      if (_____1_____) return false;
   return true;
}
void Backtrack(int t)
{
   if (t>n) sum++;
      else
         for (int i=1;i<=n;i++) {
            x[t]=i;
            if ( Place(t) == true )_____2_____
         }
 }
```

2、计算最长公共子序列长度的动态规划算法LCSLength以序列X = {x1,x2,…,xm}和Y = {y1,y2,…,yn}作为输入，计算得到数组c，其中c[i][j]存储序列Xi = {x1,x2,…,xi}和序列Yj = {y1,y2,…,yj}的最长公共子序列长度。

```
void LCSLength(int m, int n, char *x, char *y, int **c)
{
    int i,j;
    for(i = 1; i <= m; i++)c[i][0] = 0;
    for(i = 1; i <= n; i++)_____3_____
    for(i=1; i <= m; i++)
        for(j = 1; j <= n; j++)
        {
            if(x[i] == y[j])_____4_____
            else if(_____5_____)c[i][j] = c[i-1][j];
            else _____6_____
        }
}
```

3、使用分治法求解棋盘覆盖问题，用一个二维整型数组Board表示棋盘。Board[0][0]是棋盘的左上角方格。全局整型变量tile用来表示L型骨牌的编号，其初始值为0。算法的输入参数是：

tr：棋盘左上角方格的行号；

tc：棋盘左上角方格的列号；

dr：特殊方格所在的行号；

dc：特殊方格所在的列号；

size：size = $2^k$，棋盘规格为$2^k*2^k$。

```
void ChessBoard(int tr, int tc, int dr, int dc, int size)
{
    if(size == 1)return;
    int t = tile++;
    int s = _____7_____
```

```
        if(dr < tr + s && dc < tc + s)
            ChessBoard(tr,tc,dr,dc,s);
        else
        {
            Board[tr + s -1][tc + s] = t;
            ChessBoard(tr , tc + s, tr + s - 1, tc + s ,s);
        }
        if(dr >= tr + s && dc < tc + s)
            ChessBoard(tr + s,tc,dr,dc,s);
        else
        {
            Board[tr + s][tc + s - 1] = t;
            ChessBoard(tr + s , tc, tr + s , tc + s -1,s);
        }
        if(dr >= tr + s && dc >= tc + s)
            ChessBoard(tr + s,tc + s,dr,dc,s);
        else
        {
            Board[tr + s][tc + s] = t;
            ChessBoard(tr + s, tc + s, tr + s, tc + s,s);
        }
    }
```

装

订

线

装

订

线

**四、算法设计题**（本大题共 2 小题，每小题 15 分，共 30 分。请先说明算法设计思路，然后用C语言实现）

| 得分 | |
|---|---|

1、图的m着色问题

给定有n个顶点的无向连通图G和m种不同的颜色。用这些颜色为图G的各顶点着色，每个顶点着一种颜色。是否有一种着色法使G中每条边的2个顶点着不同颜色。这个问题是图的m可着色判定问题。

输入：

第一行：图的顶点数n和颜色数m

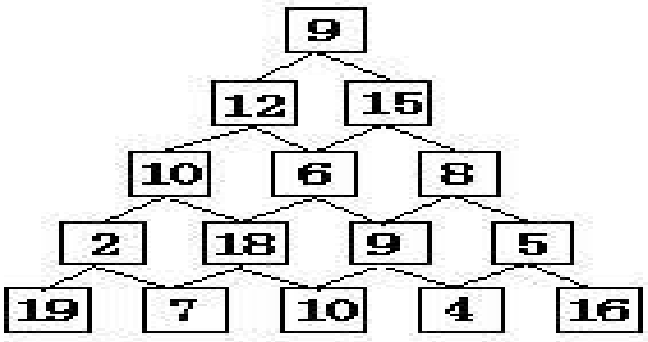
接下来输入n行，每行有n个数组成，每个数可以是0或1，其中第i行第j列的数表示顶点i与顶点j是否有边相连，是1表示有边相连，是0表示无

输出：

能用m种不同颜色对图根据规则进行着色则输出Y，否则输出N。

2、数塔问题

有形如下图所示的数塔，从顶部出发，在每一结点可以选择向左走或是向右走， 一直走到底层，要求找出一条路径，使路径上的结点总和最大。




输入：第一行数塔的层数n

接下来输入n行，分别为数塔的n层，其中第i行输入i个数

输出：

路径的最大和。

# 华南农业大学期末考试答案（A 卷）

**2010 学年第一学期**　　**考试科目：** 算法分析与设计

**考试类型：（闭卷）考试**　　**考试时间：** **120** **分钟**

学号＿＿＿＿＿＿姓名＿＿＿＿＿年级专业＿＿＿＿＿＿＿＿＿＿

装

| 题号 | 一 | 二 | 三 | 四 | 总分 |
|---|---|---|---|---|---|
| 得分 | | | | | |
| 评阅人 | | | | | |

**注意：所有答案请写在答卷上，写在试卷上不得分。**

订

**一、选择题**（本大题共 10 小题，每小题 2 分，共 20 分）

| 得分 | |
|---|---|

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | A | 2 | A | 3 | B | 4 | B | 5 | A |
| 6 | B | 7 | C | 8 | A | 9 | A | 10 | C |

线

二、**应用题**（本大题共 5 小题，每小题 6 分，共 30 分）

| 得分 | |
|---|---|

1、由于n每次缩小为原来的1/2，因此其最坏时间复杂度为O（logn）。

2、T（n）＝ 2T（n/2） + O(n)

T（1）＝ O（1）

有T（n）属于O（nlogn）

其中2T（n/2）表示求n个点的最接近点对问题可以把点平均分成两部分，每一部分的点数目为n/2，对每部分递归地求最接近点。除此之外，还要计算一个点在左半部分，一个点在右半部分的情况，此时对于左边的每个点，在右边最多只需考虑6个点，因此可以在O（n）时间内完成此类情况。

3、贪心策略为：重量轻的集装箱先装上轮船，依次循环直到不能再把剩余的集装箱装上去为止。

4、

(1)产生x[k]的时间；

(2)满足显约束的x[k]值的个数；

(3)计算约束函数constraint的时间；

(4)计算上界函数bound的时间；

(5)满足约束函数和上界函数约束的所有x[k]的个数。

5、

$$m[i,j]=\begin{cases}0 & i=j\\ \min\limits_{i\le k<j}\{m[i,k]+m[k+1,j]+p_{i-1}p_kp_j\} & i<j\end{cases}$$

计算次序为：先算m[1][1],m[2][2]….m[n][n]1个矩阵连乘的情况，再算两个矩阵连乘的情况，依次类推，最后算n个矩阵连乘。

装

1.5CM

订

**三、程序填空题**（本大题共 10 空格，每空 2 分，共 20 分。注意：每个空格只填一个语句）

| 得分 | |
|---|---|

1、 (abs(k-j)==abs(x[j]-x[k]))||(x[j]==x[k])

**2、** Backtrack(t+1)

**3、** c[0][i] = 0;

**4、** c[i][j] = c[i-1][j-1] + 1;

线

**5、** c[i-1][j] >= c[i][j-1]

**6、** c[i][j] = c[i][j-1];

**7、** size / 2;

**8、** ChessBoard(tr,tc,dr,dc,s);

**9、** Board[tr + s -1][tc + s - 1] = t;

**10、** ChessBoard(tr , tc, tr + s - 1, tc + s -1,s);

**四、算法设计题**（本大题共 2 小题，每小题 15 分，共 30 分。请先说明算法设计思路，然后用C语言实现）

| 得分 | |
|---|---|

1、

可以通过回溯的方法，不断的为每一个节点着色，在前面 n-1 个节点都合法的着色之后，开始对第 n 个节点进行着色，这时候枚举可用 的 m 个颜色，通过和第 n 个节点相邻的节点的颜色，来判断这个颜色是否合法，如果找到那么一种颜色使得第 n 个节点能够着色，那么说明 m 种颜色的方案是可行 的。返回真即可：

```
1.  //用于判断当前节点上涂上这个颜色可不可行，与其邻接节点的颜色做判断，这里
    用邻接表来存储图的信息
2.  bool isok(int step)
3.  {
4.      vector<int>::iterator iter;
5.      for(iter = input[step].begin(); iter != input[step].end(); iter++)
6.      {
```

```
            if(Color[step] == Color[*iter]) return false;
        }
        return true;
}
//step 表示 0->n 的节点，color_num 是指给 color_num 的颜色的个数可用
//判断如果给 color_num 的颜色的个数是否可行，如果可行返回 true,否则 false
bool DFS(int step, int color_num)
{
    if(step >= n) return true;
    else
    {
        int i;
        for(i = 1; i<= color_num; i++)
        {
            Color[step] = i;
            if(isok(step))
            {
                if(DFS(step + 1, color_num))
                    return true;
            }
            Color[step] = 0;
        }
    }
    return false;
}
```

2、在用动态规划考虑数塔问题时可以自顶向下的分析，自底向上的计算。

从顶点出发时到底向左走还是向右走应取决于是从左走能取到最大值还是从右走能取到最大值， 只要左右两道路径上的最大值求出来了才能作出决策。

同样的道理下一层的走向又要取决于再下一层上的最大值是否已经求出才能决策。

这样一层一层推下去，直到倒数第二层时就非常明了。

所以实际求解时，可从底层开始，层层递进，最后得到最大值。

设：f[i,j]为从第i阶段中的点j至第n行的最大的数字和；

f[n,j]=a[n,j]　1<=j<=n

f[i,j]=max{a[i,j]+f[i+1,j],a[i,j]+f[i+1,j+1]}　1<=j<=i.

f[1,1]即为所求。

```
#include<stdio.h>
#include<algorithm>
```

装

订

线

```
using namespace std;
int main()
{
int c,n ,i,j ;int arr[105][105],f[105][105];
int fi,se;
scanf("%d",&c);
while(c--)
{
scanf("%d",&n);
for(i=1;i<=n;i++)
for(j=1;j<=i;j++)
scanf("%d",&arr[i][j]);
memset(f,0,sizeof(f));//每次新输入一个测试数据先把 F 赋初值为 0
for(i=1;i<=n;i++)//从底至上时最底一行数字不用再加，直接赋值

f[n][i]=arr[n][i];
for(i=n-1;i>=1;i--)
                    for(j=1;j<=i;j++)
{
fi=j;se=j+1;
if(f[i+1][fi]+arr[i][j]>f[i][j])
f[i][j]=f[i+1][fi]+arr[i][j];
if(f[i+1][se]+arr[i][j]>f[i][j])
```

```
f[i][j]=f[i+1][se]+arr[i][j];
}
printf("%d\n",f[1][1]);
}
return  0;
}
```

装

订

线