第九届全国青少年信息学奥林匹克联赛（N0IP2003）提高组复赛解题报告

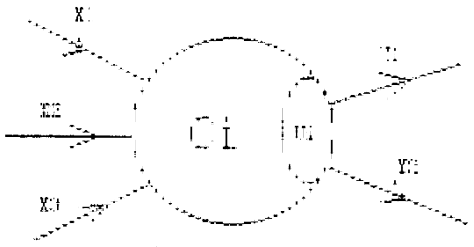
题一　神经网络

【问题背景】

人工神经网络（Artificial Neural Network）是一种新兴的具有自我学习能力的计算系统，在模式识别、函数逼近及贷款风险评估等诸多领域有广泛的应用。对神经网络的研究一直是 当今的热门方向，兰兰同学在自学了一本神经网络的入门书籍后，提出了一个简化模型，他希望你能帮助他用程序检验这个神经网络模型的实用性。

【问题描述】

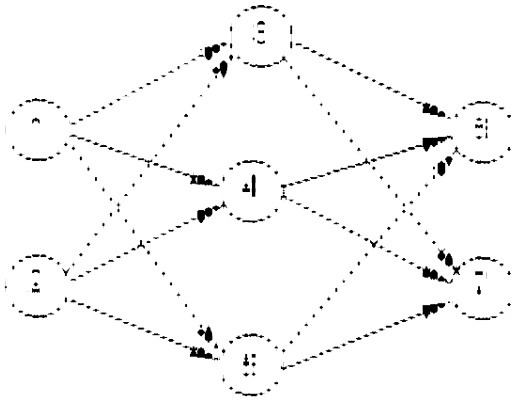
在兰兰的模型中，神经网络就是一张有向图，图中的节点称为神经元，而且两个神经元之间至多有一条边相连，下图是一个神经元的例子：



神经元〔编号为 1）

图中，X1—X3 是信息输入渠道，Y1－Y2 是信息输出渠道，C1 表示神经元目前的状态，Ui 是阈值，可视为神经元的一个内在参数。

神经元按一定的顺序排列，构成整个神经网络。在兰兰的模型之中，神经网络中的神经无分为几层；称为输入层、输出层，和若干个中间层。每层神经元只向下一层的神经元输出信息，只从上一层神经元接受信息。下图是一个简单的三层神经网络的例子。



兰兰规定，C_i服从公式：（其中 n 是网络中所有神经元的数目）

$$C_i = \sum_{(j,i)\in E} W_{ji} C_j - U_i$$

公式中的 Wji（可能为负值）表示连接 j 号神经元和 i 号神经元的边的权值。当 Ci 大于 0 时，该神经元处于兴奋状态，否则就处于平静状态。当神经元处于兴奋状态时，下一秒它会向其他神经元传送信号，信号的强度为 Ci。

如此．在输入层神经元被激发之后，整个网络系统就在信息传输的推动下进行运作。现在，给定一个神经网络，及当前输入层神经元的状态（Ci），要求你的程序运算出最后网 络输出层的状态。

【输入格式】

输入文件第一行是两个整数 n（1≤n≤200）和 p。接下来 n 行，每行两个整数，第 i＋1 行是神经元 i 最初状态和其阈值（Ui），非输入层的神经元开始时状态必然为 0。再下面 P 行，每行由两个整数 i，j 及一个整数 Wij，表示连接神经元 i、j

的边权值为 Wij。

【输出格式】

输出文件包含若干行，每行有两个整数，分别对应一个神经元的编号，及其最后的状态，两个整数间以空格分隔。仅输出最后状态非零的输出层神经元状态，并且按照编号由小到大顺序输出！

若输出层的神经元最后状态均为 0，则输出 NULL。

【输入样例】

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

【输出样例】

```
3 1
4 1
5 1
```

[分析]本题是比较简单的，但要注意神经元的层数，只输出最大层（输出层）的状态非零的神经元的状态，在中间层中只有神经元处于兴奋状态时（Ci>0）才会向下一层传送信号。

[PASCAL 源程序]

```
program NOIP2003_1_Network;
  const
    maxn=200;maxp=200;
  var
    i,j,n,p,maxlayer:integer;
    w:array[0..maxp]of longint;{存放边的权值}
    start,terminal:array[0..maxp]of byte;{存放边的起点与终点}
    u,c:array[0..maxn]of longint;{存放神经元的阀值与状态值}
    layer:array[0..maxn]of byte;{存放神经元的层数}
    f1,f2:text;fn1,fn2,fileNo:string;
    flag:boolean;
  begin
    write('Input fileNo:');
    readln(fileNo);
    fn1:='network.in'+fileNo;
    fn2:='network.ou'+fileNo;
    assign(f1,fn1);reset(f1);
    assign(f2,fn2);rewrite(f2);
    readln(f1,n,p);
    for i:=1 to n do readln(f1,c[i],u[i]);
    fillchar(layer,sizeof(layer),0);
    for i:=1 to p do begin
      readln(f1,start[i],terminal[i],w[i]);{读入边的起点，终点，权}
      layer[terminal[i]]:=layer[start[i]]+1;{计算终点的层数（比起点大 1）}
    end;
    close(f1);
    maxlayer:=layer[terminal[p]];{求最大层数，即输出层的层数}
    for i:=1 to n do{计算非输入层的节点 i 的状态值}
      if layer[i]>0 then begin
        for j:=1 to p do
          if (terminal[j]=i) and (c[start[j]]>0){与目标节点 i 有边相连的节点 j 且其状态处于兴奋时（Cj>0）才向节点 I 传送信号}
            then c[i]:=c[i]+w[j]*c[start[j]];
        c[i]:=c[i]-u[i];
      end;
      {输出结果}
      flag:=true;
    for i:=1 to n do
      if (layer[i]=maxlayer) and (c[i]>0) then begin
        writeln(f2,i,' ',c[i]);
        flag:=false;
      end;
    if flag then writeln(f2,'NULL');
    close(f2);
  end.
```

[点评]基本题。题目阅读起来有些费神，题中边的权值 W,

神经元的状态值 C,阀值 U，均未明确指出其取值范围，反　　映出命题不严谨。

题二　侦探推理

【问题描述】

明明同学最近迷上了侦探漫画《柯南》并沉醉于推理游戏之中，于是他召集了一群同学玩推理游戏。游戏的内容是这样的，明明的同学们先商量好由其中的一个人充当罪犯（在明明不知情的情况下），明明的任务就是找出这个罪犯。接着，明明逐个询问每一个同学，被询问者可能会说：

证词内容	证词含义
I am guilty.	我是罪犯
I am not guilty.	我不是罪犯
XXX is guilty.	XXX 是罪犯（XXX 表示某个同学的名字）
XXX is not guilty.	XXX 不是罪犯
Today is XXX.	今天是 XXX（XXX 表示星期几，是 Monday Tuesday Wednesday Thursday Friday Saturday Sunday 其中之一）

证词中出现的其他话，都不列入逻辑推理的内容。

明明所知道的是，他的同学中有 N 个人始终说假话，其余的人始终说真。

现在，明明需要你帮助他从他同学的话中推断出谁是真正的凶手，请记住，凶手只有一个！

【输入格式】

输入由若干行组成，第一行有二个整数，M（1≤M≤20）、 N（1≤N≤M）和 P（1≤P≤100）； M 是参加游戏的明明的同学数，N 是其中始终说谎的人数，P 是证言的总数。接下来 M 行，每行是明明的一个同学的名字（英文字母组成，没有主格，全部大写）。往后有 P 行，每行开始是某个同学的名字，紧跟着一个冒号和一个空格，后面是一句证词，符合前表中所列格式。证词每行不会超过 250 个字符。

输入中不会出现连续的两个空格，而且每行开头和结尾也没有空格。

【输出格式】

如果你的程序能确定谁是罪犯，则输出他的名字；如果程序判断出不止一个人可能是罪犯，则输出 Cannot Determine；如果程序判断出没有人可能成为罪犯，则输出 Impossible。

【输入样例】

3 1 5
MIKE
CHARLES
KATE
MIKE:I am guilty.
MIKE:Today is Sunday.
CHARLES:MIKE is guilty．
KATE:I am guilty.
KATE:How are you??

【输出样例】

MIKE

[分析]基本思路有两种：一是可以穷举 M 个人中有 N 个人始终说假话的所有组合，据此出发，判断每句证词的真伪，再推断谁是罪犯，但这样做运算量大；二是可以穷举 M 个人中的任意一个是罪犯，由此再来判断每句证词的真伪，推断谁说真话谁说假话，这样做运算量小得多。这里采用后者。有几点必须注意：一，不能说找到某人是罪犯或可能是罪犯就完事了，还必须确保是否还有别人是罪犯或可能是罪犯；二，如何处理关于星期的证词呢？还是可以采用穷举；三，某人的证词可能前后矛盾，在给某人标记他是始终说真话还是始终说假话时，一定要考察他此前的诚信记录；因此，对某人的诚信记录要有 4 种不同的区分，可用 0 表示此人尚无有效记录（未说过真话也未说过假话），用 1 表示此人说真话，用 2 表示此人说假话，用 3 表示此人说的话与他前面的证词冲突；四，如何判断最初穷举时设定的前提（某人是罪犯）是否是本题的一个解呢？如果有人的诚信记录为 3，则肯定不是本题的解；但是也不能强求诚信记录为 2 的人的总数一定要等于 n，而是只要诚信记录为 2 的人不超过 n 且诚信记录为 1 的人不超过 m-n 即可，因为诚信记录为 0 的人可能说真话也可能说假话，他们只是没有说话，或只说了废话。五，由于证词要反复用于判断，可以先剔除其中的无效证词；为处理方便，将有效证词

分为两部分：不含星期的和含星期的。

[PASCAL 源程序]

```
program NOIP2003_2_Logic;
  const
    maxm=20;
                                        dow:array[1..7]of
string=('Sunday.','Monday.','Tuesday.','Wednesday.',
    'Thursday.','Friday.','Saturday.');
  var
     i,j,k,weekday,m,n,p,p1,p2,p3,index,resolution,total1,tot
al2:byte;
    name:array[1..maxm]of string;{存放人名}
     witness10,witness20:array[1..100]of byte;{存放说话人
的序号,分为两类,前者存放不含星期的证词的说话人的序号,
后者存放只含星期的证词的说话人的序号}
     witness1,witness2:array[1..100]of string;{存放证词,分
为两类,第一类是不含星期的证词,第二类是只含星期的证
词}
    name0,temp,temp0,temp1,temp2:string;
    truth,truth0:array[1..maxm]of byte; {存放诚信记录}
    f1,f2:text;fn1,fn2,fileNo:string;
    flag:boolean;
  begin
    write('Input fileNo:');readln(fileNo);
    fn1:='logic.in'+fileNo;fn2:='logic.ou'+fileNo;
    assign(f1,fn1);reset(f1);assign(f2,fn2);rewrite(f2);
    readln(f1,m,n,p);
    for i:=1 to m do readln(f1,name[i]);
    total1:=0;total2:=0;
    for i:=1 to p do begin{对证词预处理}
      readln(f1,temp);
      index:=pos(': ',temp);
      temp1:=copy(temp,1,index-1);{取得说话人姓名}
       temp2:=copy(temp,index+2,length(temp)-index-1);{取
得证词}
          if (temp2='I am guilty.') or (temp2='I am not
guilty.') then
            for j:=1 to m do
          if name[j]=temp1 then begin
            inc(total1);{total1 表示第一类证词的总数}
            witness10[total1]:=j;{记下第一类第 total1 条证词的
说话人的序号}
            witness1[total1]:=temp2;{记下第一类第 total1 条证
词}
            break;
          end;
          if (pos(' is guilty.',temp2)>0) or
           (pos(' is not guilty.',temp2)>0) then begin
         temp0:=copy(temp2,1,pos(' is ',temp2)-1);{取得证词
的叙述对象(主语)}
        flag:=false;
        for k:=1 to m do
          if temp0=name[k] then begin
            flag:=true;
            break;
          end;
        if flag then{如果证词的叙述对象(主语)确实存在}
          for j:=1 to m do
             if name[j]=temp1 then begin{记入到第一类证词
中}
              inc(total1);
              witness10[total1]:=j;
              witness1[total1]:=temp2;
              break;
            end;
      end;
      flag:=false;
      for j:=1 to 7 do
        if temp2='Today is '+ dow[j] then begin
          flag:=true;
          break;
        end;
      if flag then{如果证词是关于星期的判断}
        for j:=1 to m do
          if name[j]=temp1 then begin{记入到第二类证词中}
            inc(total2);{total2 表示第二类证词的总数}
            witness20[total2]:=j;{记下第二类第 total2 条证词的
说话人的序号}
            witness2[total2]:=temp2;{记下第二类第 total2 条证
词}
            break;
          end;
    end;
    close(f1);
    resolution:=0;{resolution 表示解的个数 }
    for i:=1 to m do begin{穷举,第 i 个人为罪犯}
          if resolution>1 then break;{如果解的个数多于 1 个,
则跳出循环}
          fillchar(truth,sizeof(truth),0);{诚信记录赋初值为 0,
表示此人尚无有效证词}
```

```
    for j:=1 to total1 do begin{逐条处理第一类证词}
      if witness1[j]='I am guilty.' then begin{如果证词为 I am guilty.}
        if i=witness10[j] then{如果说话人就是罪犯,则本证词为真}
          case truth[i] of
            0:truth[i]:=1;{如果此人的诚信记录为 0,则此人说真话(记为 1)}
            2:truth[i]:=3;{如果此人的诚信记录为 2(即以前说假话),则此人自相矛盾(记为 3)}
          end
        else{如果说话人不是罪犯,则本证词为假}
          case truth[witness10[j]] of
            0:truth[witness10[j]]:=2;{如果此人的诚信记录为 0,则此人说假话(记为 2)}
            1:truth[witness10[j]]:=3;{如果此人的诚信记录为 1(即以前说真话),则此人自相矛盾(记为 3)}
          end;
      end;
      if witness1[j]='I am not guilty.' then begin{如果证词为 I am not guilty.}
        if i=witness10[j] then{如果说话人是罪犯,则本证词为假}
          case truth[i] of
            0:truth[i]:=2;
            1:truth[i]:=3;
          end
        else{如果说话人不是罪犯,则本证词为真}
          case truth[witness10[j]] of
            0:truth[witness10[j]]:=1;
            2:truth[witness10[j]]:=3;
          end;
      end;
      if (pos(' is guilty.',witness1[j])>0) then begin{如果证词含有 is guilty. }
                          temp:=copy(witness1[j],1,pos(' is guilty.',witness1[j])-1);{取得证词的主语}
        if name[i]=temp then{如果证词的主语是罪犯,则本证词为真}
          case truth[witness10[j]] of
            0:truth[witness10[j]]:=1;
            2:truth[witness10[j]]:=3;
          end
        else{如果证词的主语不是罪犯,则本证词为假}
          case truth[witness10[j]] of
            0:truth[witness10[j]]:=2;
            1:truth[witness10[j]]:=3;
          end;
      end;
      if (pos(' is not guilty.',witness1[j])>0) then begin{如果证词含有 is not guilty. }
                          temp:=copy(witness1[j],1,pos(' is not guilty.',witness1[j])-1);{取得证词的主语}
        if name[i]=temp then{如果证词的主语是罪犯,则本证词为假}
          case truth[witness10[j]] of
            0:truth[witness10[j]]:=2;
            1:truth[witness10[j]]:=3;
          end
        else{如果证词的主语不是罪犯,则本证词为真}
          case truth[witness10[j]] of
            0:truth[witness10[j]]:=1;
            2:truth[witness10[j]]:=3;
          end;
      end;
    end;{第一类证词全部处理完毕}
    if total2>0 then begin{如果有第二类证词存在,处理第二类证词}
      for k:=1 to m do truth0[k]:=truth[k];{记下第一类证词全部处理完毕后的诚信记录}
      for weekday:=1 to 7 do begin{穷举,今天是星期日,星期一直到星期六}
        for k:=1 to m do truth[k]:=truth0[k];{诚信记录还原为第一类证词全部处理完毕后的诚信记录}
        for j:=1 to total2 do{逐条处理第二类证词}
          if pos(dow[weekday],witness2[j])>0 then{如果证词中含有当前穷举的星期值,则本证词为真}
            case truth[witness20[j]] of
              0:truth[witness20[j]]:=1;
              2:truth[witness20[j]]:=3;
            end
          else{如果证词中不含有当前穷举的星期值,则本证词为假}
            case truth[witness20[j]] of
              0:truth[witness20[j]]:=2;
              1:truth[witness20[j]]:=3;
            end;
        p1:=0;p2:=0;p3:=0;
        for k:=1 to m do if truth[k]=1 then inc(p1);{p1 表示始终说真话的人的总数}
        for k:=1 to m do if truth[k]=2 then inc(p2);{p2 表示始终说假话的人的总数}
```

```
        for k:=1 to m do if truth[k]=3 then inc(p3);{p3 表示说过自相矛盾的话的人的总数}
            if (p1<=m-n) and (p2<=n) and (p3=0) then begin{如果说真话的人的总数小于等于 m-n 且说假话的人的总数小于等于 n 且没有人说过自相矛盾的话,那么当前罪犯 i 就是本题的一个解}
          name0:=name[i];{记下罪犯的姓名}
          inc(resolution);{解的个数增 1}
          break;{退出星期的穷举}
        end;
      end;{星期的穷举完毕}
    end;{第二类证词处理完毕}
    p1:=0;p2:=0;p3:=0;
    for k:=1 to m do if truth[k]=1 then inc(p1);
    for k:=1 to m do if truth[k]=2 then inc(p2);
    for k:=1 to m do if truth[k]=3 then inc(p3);
        if (p1<=m-n) and (p2<=n) and (p3=0) and (name0<>name[i]) then begin{为避免重复计解,此处多加了一个条件: name0<>name[i]}
          name0:=name[i];
          inc(resolution);
        end;
    end;
    if resolution=1 then writeln(f2,name0);{如果只有一个解,则输出罪犯姓名}
     if resolution=0 then writeln(f2,'Impossible');{如果无解,则输出 Impossible}
     if resolution>1 then writeln(f2,'Cannot Determine');{如果有多个可能解,则输出 Cannot Determine }
    close(f2);
  end.
```

[点评]基本题，比较复杂，重点考查参赛者的字符串运算和逻辑运算，逻辑推理能力。难点主要在于如何处理关于星期的证词，以及证词之间是否存在矛盾。

题三　加分二叉树

【问题描述】

设一个 n 个节点的二叉树 tree 的中序遍历为（l,2,3,…,n），其中数字 1,2,3,…,n 为节点编号。每个节点都有一个分数（均为正整数），记第 i 个节点的分数为 di，tree 及它的每个子树都有一个加分，任一棵子树 subtree（也包含 tree 本身）的加分计算方法如下：

subtree 的左子树的加分× subtree 的右子树的加分＋subtree 的根的分数

若某个子树为空，规定其加分为 1，叶子的加分就是叶节点本身的分数。不考虑它的空子树。

试求一棵符合中序遍历为（1,2,3,…,n）且加分最高的二叉树 tree。要求输出；

（1）tree 的最高加分

（2）tree 的前序遍历

【输入格式】

第 1 行：一个整数 n（n＜30），为节点个数。

第 2 行：n 个用空格隔开的整数，为每个节点的分数（分数＜100）。

【输出格式】

第 1 行：一个整数，为最高加分（结果不会超过 4,000,000,000）。

第 2 行：n 个用空格隔开的整数，为该树的前序遍历。

【输入样例】

5

5 7 1 2 10

【输出样例】

145

3 1 2 4 5

[分析]很显然，本题适合用动态规划来解。如果用数组 value[i,j]表示从节点 i 到节点 j 所组成的二叉树的最大加分，则动态方程可以表示如下：

value[i,j]=max{value[i,i]+value[i+1,j],value[i+1,i+1]+value[i,i]*value[i+2,j], value[i+2,i+2]+value[i,i+1]*value[i+3,j],…,value[j-1,j-1]+value[i,j-2]*value[j,j], value[j,j]+value[i,j-1]}

题目还要求输出最大加分树的前序遍历序列，因此必须在计算过程中记下从节点 i 到节点 j 所组成的最大加分二叉树的根节点，用数组 root[i,j]表示

[PASCAL 源程序]

```
{$N+}
program NOIP2003_3_Tree;
  const
    maxn=30;
  var
    i,j,n,d:byte;
    a:array[1..maxn]of byte;
    value:array[1..maxn,1..maxn]of comp;
    root:array[1..maxn,1..maxn]of byte;
    s,temp:comp;
    f1,f2:text;fn1,fn2,fileNo:string;
  procedure preorder(p1,p2:byte);{按前序遍历输出最大加分二叉树}
    begin
      if p2>=p1 then begin
        write(f2,root[p1,p2],' ');
        preorder(p1,root[p1,p2]-1);
        preorder(root[p1,p2]+1,p2);
      end;
    end;
  begin
    write('Input fileNo:');readln(fileNo);
    fn1:='tree.in'+fileNo;fn2:='tree.ou'+fileNo;
    assign(f1,fn1);reset(f1);
    assign(f2,fn2);rewrite(f2);
    readln(f1,n);
    for i:=1 to n do read(f1,a[i]);
    close(f1);
    fillchar(value,sizeof(value),0);
    for i:=1 to n do begin
      value[i,i]:=a[i];{计算单个节点构成的二叉树的加分}
      root[i,i]:=i;{记录单个节点构成的二叉树的根节点}
    end;
    for i:=1 to n-1 do begin
      value[i,i+1]:=a[i]+a[i+1];{计算相邻两个节点构成的二叉树的最大加分}
```

```
      root[i,i+1]:=i;{记录相邻两个节点构成的二叉树的根节点；需要说明的是，两个节点构成的二叉树，其根节点可以是其中的任何一个；这里选编号小的为根节点，则编号大的为其右子树；若选编号大的为根节点，则编号小的为其左子树；因此，最后输出的前序遍历结果会有部分不同但同样是正确的。如果最大加分二叉树的所有节点的度数都是 0 或 2，则最后输出的前序遍历结果是唯一的。}
    end;
    for d:=2 to n-1 do begin{依次计算间距为 d 的两个节点构成的二叉树的最大加分}
      for i:=1 to n-d do begin
        s:=value[i,i]+value[i+1,i+d];{计算以 i 为根节点，以 i+1 至 i+d 间所有节点为右子树的二叉树的最大加分}
        root[i,i+d]:=i; {记录根节点 i}
        for j:=1 to d do begin
          temp:=value[i+j,i+j]+value[i,i+j-1]*value[i+j+1,i+d];{计算以 i+j 为根节点，以 i 至 i+j-1 间所有节点为左子树，以 i+j+1 至 i+d 间所有节点为右子树的二叉树的最大加分}
          if temp>s then begin{如果此值为最大}
            s:=temp;root[i,i+d]:=i+j;{记下新的最大值和新的根节点}
          end;
        end;
        temp:=value[i,i+d-1]+value[i+d,i+d];{计算以 i+d 为根节点，以 i 至 i+d-1 间所有节点为左子树的二叉树的最大加分}
        if temp>s then begin
          s:=temp;root[i,i+d]:=i+d+1;
        end;
        value[i,i+d]:=s;
      end;
    end;
    writeln(f2,value[1,n]:0:0);{输出最大加分}
    preorder(1,n);{输出最大加分二叉树的前序遍历序列}
    close(f2);
  end.
```

[点评]基本题。考查了二叉树的遍历和动态规划算法。难点在于要记录当前最大加分二叉树的根节点。疑点是最大加分二叉树的前序遍历序列可能不唯一。

题四　传染病控制

【问题背景】

近来，一种新的传染病肆虐全球。蓬莱国也发现了零星感染者，为防止该病在蓬莱国大范围流行，该国政府决定不惜一切代价控制传染病的蔓延。不幸的是，由于人们尚未完全认识这种传染病，难以准确判别病毒携带者，更没有研制出疫苗以保护易感人群。于是，蓬莱国的疾病控制中心决定采取切断传播途径的方法控制疾病传播。经过 WHO（世界卫生组织）以及全球各国科研部门的努力，这种新兴传染病的传播途径和控制方法已经研究消楚，剩下的任务就是由你协助蓬莱国疾控中心制定一个有效的控制办法。

【问题描述】

研究表明，这种传染病的传播具有两种很特殊的性质；

第一是它的传播途径是树型的，一个人 X 只可能被某个特定的人 Y 感染，只要 Y 不得病，或者是 XY 之间的传播途径被切断，则 X 就不会得病。

第二是，这种疾病的传播有周期性，在一个疾病传播周期之内，传染病将只会感染一代患者，而不会再传播给下一代。

这些性质大大减轻了蓬莱国疾病防控的压力，并且他们已经得到了国内部分易感人群的潜在传播途径图（一棵树）。但是，麻烦还没有结束。由于蓬莱国疾控中心人手不够，同时也缺乏强大的技术，以致他们在一个疾病传播周期内，只能设法切断一条传播途径，而没有被控制的传播途径就会引起更多的易感人群被感染（也就是与当前已经被感染的人有传播途径相连，且连接途径没有被切断的人群）。当不可能有健康人被感染时，疾病就中止传播。所以，蓬莱国疾控中心要制定出一个切断传播途径的顺序，以使尽量少的人被感染。你的程序要针对给定的树，找出合适的切断顺序。

【输入格式】

输入格式的第一行是两个整数 n（1≤n≤300）和 p。接下来 p 行，每一行有两个整数 i 和 j，表示节点 i 和 j 间有边相连（意即，第 i 人和第 j 人之间有传播途径相连）。其中节点 1 是已经被感染的患者。

【输出格式】

只有一行，输出总共被感染的人数。

【输入样例】

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

【输出样例】

```
3
```

[分析]本题关键是要找到切断传染途径的算法，这并不难。以测试数据 Epidemic.in5 为例说明如下：

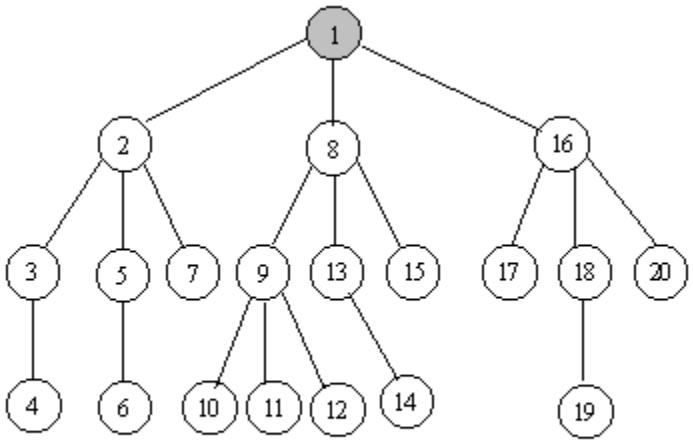


图1

在图 1 中，可以先切断 1 与 2 或者 1 与 8 或者 1 与 16 之间的任何一条途径，以先切断 1 与 2 为例，结果如下：

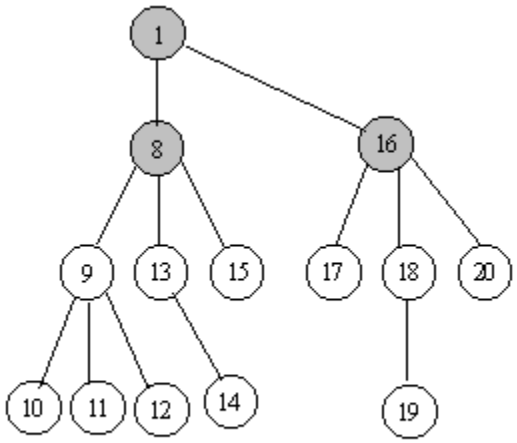


图2

此时，1，8，16 均已被感染，可将它们“合并”为一个新的节点，节点号取作 8，如下图：

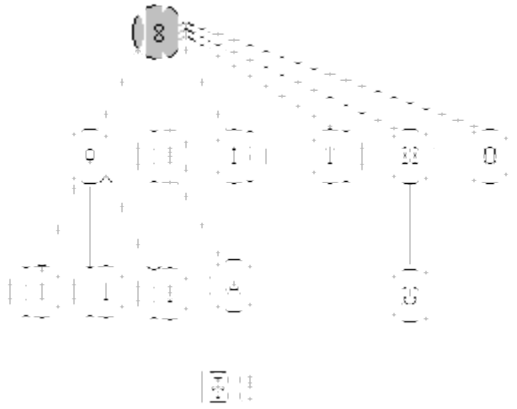


图3

这样，它与图 1 十分相似，因此可用递归算法来做。

[PASCAL 源程序]

```
program NOIP2003_4_Epidemic;
  const
    maxn=300;maxp=300;
  type
    node=array [0..maxp] of integer;{节点数据类型是一维数组，其中下标为 0 的元素中存放该节点的孩子节点个数，
```

```
下标为 1 的元素中存放该节点的第 1 个孩子的节点序号，下标为 i 的元素中存放该节点的第 i 个孩子的节点序号}
  var
    i,n,p,min,max,temp,s,smin:integer;
    a:array[1..maxn] of ^node;{存放 n 个节点的数据，使用常规数组容量是不够的，故采用动态数组}
    f1,f2:text;fn1,fn2,fileNo:string;
  procedure try(i:integer);{求以 i 为根节点的树中被感染人数的最少值}
    var
      root1,root2,j,k,m,temp,s0:integer;b:node;flag:boolean;
    begin
      if a[i]^[0]<=1 then begin{如果根节点 i 的孩子数为 0 或 1，则已完成一轮递归}
        if s<smin then smin:=s;{如果此轮递归得到的感染人数最少，则刷新最少感染人数}
        exit;
      end;
      s0:=s;{记下上一层递归完后的感染人数}
      flag:=true;{逻辑标志}
      for j:=1 to a[i]^[0] do if (a[a[i]^[j]]^[0]>0) then begin{依次切断根节点 i 的第 j 个孩子，这里进行了优化，如果根节点 i 的第 j 个孩子是叶子节点则暂不考虑}
        flag:=false;
        s:=s0+a[i]^[0]-1;{切断 j 的同时，被感染的人数增加了 a[i]^[0]-1 个}
        if j=1 then root1:=2 else root1:=1;{选根节点 i 的第 root1 个孩子为新树的根节点}
        root2:=a[i]^[root1];{求新树的根节点的序号}
        temp:=a[root2]^[0]; {求新树的根节点的孩子数}
        for k:=1 to temp do b[k]:=a[root2]^[k];{记下合并前新树的根节点的孩子情况}
        for k:=1 to a[i]^[0] do{开始合并生成新的树}
          if (k<>j) and (k<>root1) then begin{如果不是刚被切断的子树或被选作新树根节点的子树}
            for m:=1 to a[a[i]^[k]]^[0] do{将它们并入新树}
              a[root2]^[a[root2]^[0]+m]:=a[a[i]^[k]]^[m];
            a[root2]^[0]:=a[root2]^[0]+a[a[i]^[k]]^[0];{新树根节点的孩子数也随着增加}
          end;
        try(root2);{对新树进行递归运算}
        a[root2]^[0]:=temp;{节点 root2 的孩子数还原}
        for m:=1 to temp do a[root2]^[m]:=b[m];{节点 root2 的孩子情况数据还原}
      end;
      if flag then begin{如果根节点 i 的所有孩子都是叶子节点}
        s:=s0+a[i]^[0]-1;{则切断任何一个都是等效的，故感染人数为 s0+a[i]^[0]-1}
        if s<smin then smin:=s;{如果此轮递归得到的感染人数最少，则刷新最少感染人数}
        exit;
      end;
    end;
  begin
    write('Input fileNo:');readln(fileNo);
    fn1:='Epidemic.in'+fileNo;fn2:='Epidemic.ou'+fileNo;
    assign(f1,fn1);reset(f1);assign(f2,fn2);rewrite(f2);
    readln(f1,n,p);
    for i:=1 to n do new(a[i]);{产生动态数组元素}
    for i:=1 to n do a[i]^[0]:=0;{每个节点的孩子数赋初值 0}
    for i:=1 to p do begin{读入节点间的连接情况}
      readln(f1,min,max);
      if min>max then begin
        temp:=min;min:=max;max:=temp{序号较小的节点为根节点，较大的为子节点}
      end;
      inc(a[min]^[0]);{根节点的孩子数增加 1}
      a[min]^[a[min]^[0]]:=max;{max 成为根节点 min 的第 a[min]^[0]个孩子}
    end;
    close(f1);
    s:=1;smin:=300;try(1);
    writeln(f2,smin);close(f2);
  end.
```

[点评]基本题。考查了动态数据结构和递归算法。