POJ2367

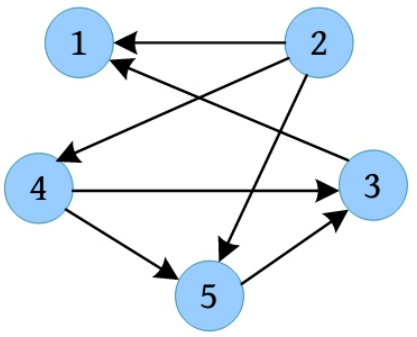
题目描述（POJ2367）：火星人的血缘关系制度令人困惑。在火星行星理事会中，令人困惑的家谱系统导致了一些尴尬：为了在所有讨论中不冒犯任何人，老火星人先发言，而不是年轻人或最年轻的无子女人员。但是，维护这个命令不是一项微不足道的任务，火星人并不总是知道其父母和祖父母是谁，如果一个孙子先发言而不是其年轻的曾祖父先发言，则会出现错误。编写程序，保证理事会的每个成员都早于其每个后代发言。

输入：第 1 行包含整数 N（1≤N≤100），表示火星行星理事会的成员数。成员编号为 1～N。接下来的 N 行，第 i 行包含第 i 个成员的孩子名单。孩子的名单可能是空的，名单以 0 结尾。

输出：单行输出一系列发言者的编号，用空格分隔。如果有几个序列满足条件，则输出任意一个，至少存在一个这样的序列。

输入样例	输出样例
5 0 4 5 1 0 1 0 5 3 0 3 0	2 4 5 3 1

题解：根据输入样例，构建的图形结构如下图所示，其拓扑序列为 2 4 5 3 1。本题属于简单的拓扑排序问题，输出拓扑序列即可。



算法代码：

```
void TopoSort(){ //拓扑排序
   int cnt=0;
   for(int i=1;i<=n;i++)
      if(indegree[i]==0)
         s.push(i);
   while(!s.empty()){
      int u=s.top();
      s.pop();
      topo[++cnt]=u;
      for(int j=1;j<=n;j++)
         if(map[u][j])
            if(--indegree[j]==0)
               s.push(j);
   }
}
int main(){
   cin>>n;
   memset(map,0,sizeof(map));
   memset(indegree,0,sizeof(indegree));
   for(int i=1;i<=n;i++){
      int v;
      while(cin>>v&&v){
         map[i][v]=1;
         indegree[v]++;
      }
   }
   TopoSort();
   for(int i=1;i<n;i++)
      cout<<topo[i]<<" ";
   cout<<topo[n]<<endl;
   return 0;
}
```

POJ1094

题目描述（POJ1094）：不同值的升序排序序列是使用某种形式的小于运算符从小到大排序的元素序列。例如，排序后的序列 ABCD 表示 A<B、B<C 和 C<D。给定一组 A<B 形式的关系，要求确定是否指定已排序的订单。

输入：输入包含多个测试用例。每个测试用例的第 1 行都包含两个正整数 n（2≤n≤26）和 m。n 表示要排序的对象数量，排序的对象是大写字母的前 n 个字符。m 表示将给出的 A<B 形式的关系的数量。接下来的 m 行，每行都包含一种由 3 个字符组成的关系：第 1 个大写字母、字符“<”和第 2 个大写字母。n=m=0 的值表示输入结束。

输出：对于每个问题实例，其输出都由一行组成，该行应该是以下三种之一。

- 在 x 种关系之后确定的排序顺序：yyy...y。
- 无法确定排序顺序。
- 在 x 种关系后发现不一致。

其中，x 是在确定排序序列或找到不一致时处理的关系数，以先到者为准，yyy...y 是已排序的升序序列。

输入样例	输出样例
4 6 A<B A<C B<C C<D B<D A<B 3 2 A<B B<A 26 1 A<Z 0 0	Sorted sequence determined after 4 relations: ABCD. Inconsistency found after 2 relations. Sorted sequence cannot be determined.

题解：在本题中，一边进行输入，一边进行判断，分为有序、无序（不一致）、无法确定三种情况，可以利用拓扑排序进行判断。

1. 算法设计

（1）如果入度为 0 的节点个数为 0，则说明有环；如果拓扑序列节点数小于 n，则也说明有环。此情况即无序。

（2）如果入度为 0 的节点个数大于 1，则无法确定，因为拓扑序列不唯一。

（3）否则是拓扑有序的，输出拓扑序列。特别注意：①得到判断结果后不能 break，需要继续输入，否则下一个测试用例会读入本次输入的剩余数据；②在数据输入完毕后才能判断是不是无法确定。

2. 算法实现

```
int TopoSort(int n){ //拓扑排序
    flag=1;
    for(int i=1;i<=n;i++)
        temp[i]=indegree[i];//一边进行输入，一边进行拓扑排序，所有入度数组都不能改变
    int m=0,cnt=0;
    for(int i=1;i<=n;i++)//查找入度为 0 的节点个数，若大于 1，则无法确定是否为拓扑排序序列
        if(temp[i]==0){
            s.push(i);
            cnt++;
        }
    if(cnt==0) return 0; //有环
    if(cnt>1) flag=-1; //不确定
```

```
    while(!s.empty()){
        cnt=0;
        int i=s.top();
        s.pop();
        topo[m++]=i;
        for(int j=1;j<=n;j++)
            if(map[i][j]){
                temp[j]--;
                if(!temp[j]){
                    s.push(j);
                    cnt++;
                }
            }
        if(cnt>1) flag=-1;//不确定
    }
    if(m<n)//有环
        return 0;
    return flag;
}

int main(){
    int m,n;
    bool sign;//在sign=1时，已得出结果
    string str;
    while(cin>>n>>m){
        if(m==0&&n==0) break;
        memset(map,0,sizeof(map));
        memset(indegree,0,sizeof(indegree));
        sign=0;
        for(int i=1;i<=m;i++){
            cin>>str;
            if(sign) continue; //一旦得出结果，则对后续的输入不做处理
            int x=str[0]-'A'+1;
            int y=str[2]-'A'+1;
            map[x][y]=1;
            indegree[y]++;
            int s=TopoSort(n);
            if(s==0){ //有环
                printf("Inconsistency found after %d relations.\n",i);
                sign=1;
            }else if(s==1){ //有序
                printf("Sorted sequence determined after %d relations: ",i);
                for(int j=0;j<n;j++)
                    cout<<char(topo[j]+'A'-1);
                printf(".\n");
                sign=1;
            }
        }
        if(!sign) //不确定
            printf("Sorted sequence cannot be determined.\n");
    }
    return 0;
}
```

POJ3687

题目描述（POJ3687）：有 N 个不同重量的球，重量为 1～N 个单位。对球从 1 到 N 进行标记，使得：①没有两个球具有相同的标签；②标签满足几个约束，例如“标签为 a 的球比标签为 b 的球轻”。

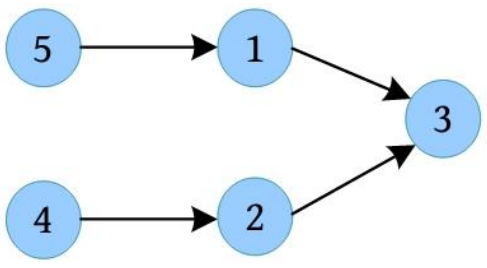
输入：第 1 行包含测试用例的数量。每个测试用例的第 1 行都包含两个整数 N（1≤N≤200）和 M（0≤M≤40000），分别表示球的数量和约束的数量。后面的 M 行，每行都包含两个整数 a 和 b，表示标签为 a 的球比标签为 b 的球轻（1≤a,b≤N）。在每个测试用例前都有一个空行。

输出：对于每个测试用例，都单行输出标签 1～N 的球的重量。如果存在多种解决方案，则输出标签为 1 的球的最小重量，然后输出标签为 2 的球的最小重量，以此类推……如果不存在解，则输出-1。

输入样例	输出样例
5 4 0 4 1 1 1 4 2 1 2 2 1 4 1 2 1 4 1 3 2	1 2 3 4 -1 -1 2 1 3 4 1 3 2 4

题解：本题不是输出小球的标签，而是按标签输出小球的重量，而且标签小的球的重量尽可能小。例如，输入以下数据，构建的图形结构如下图所示。

```
5 4 //节点数、边数
5 1 //标签为 5 的球比标签为 1 的球轻
4 2
1 3
2 3
```

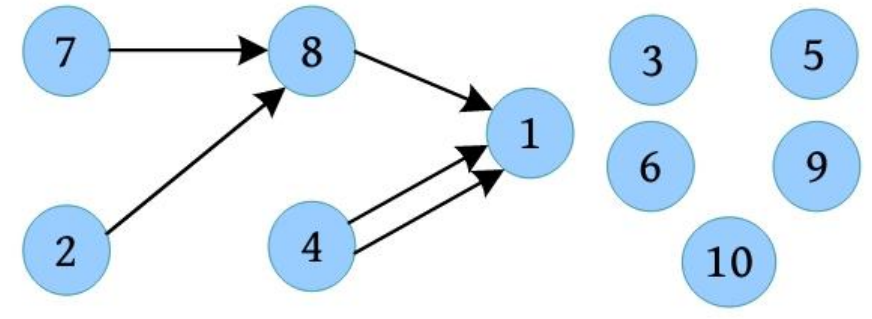


分析：根据重量关系，节点 3 是最重的，因此令重量 weight[3]=5；节点 1 和节点 2 比节点 3 轻，因为每个球的重量都不同，按照标签小的球重量小的原则，先给标签大的球分配重量，先处理节点 2，因此 weight[2]=4；节点 4 比节点 2 轻，weight[4]=3；节点 1 比节点 3 轻，weight[1]=2；节点 5 比节点 1 轻，weight[5]=1。

按照标签 1～5 输出其重量：2 4 5 3 1。

例如，输入以下数据，构建的图形结构如下图所示。

```
10 5 //节点数、边数
4 1  //标签为 4 的球比标签为 1 的球轻
8 1
7 8
4 1
2 8
```



分析：按照标签小的球重量小的原则，先给标签大的球分配重量：weight[10]=10；weight[9]=9；weight[6]=8；weight[5]=7；weight[3]=6；weight[1]=5。节点 8 和节点 4 比节点 1 轻，按照标签小的球重量小的原则，先给标签大的球分配重量，先处理节点 8，因此 weight[8]=4；节点 7 和节点 2 比节点 8 轻，先处理节点 7，weight[7]=3；现在只剩下节点 4 和节点 2，weight[4]=2；weight[2]=1。按照标签 1～10 输出其重量：5 1 6 2 7 8 3 4 9 10。

注意：本题有重复边，需要去重，否则会有环，最后输出-1。

1. 算法设计

可以采用下面**两种方法解决**。

（1）**建立正向图**。i=n...1，j=n...1，检查第 1 个出度为 0 的点 t，分配重量 w[t]=i，将弧尾节点的出度减 1，继续下一个循环。若没有出度为 0 的节点，则说明有环，退出。

（2）**建立原图的逆向图**。i=n...1，j=n...1，检查第 1 个入度为 0 的节点 t，分配重量 w[t]=i，将其邻接点的入度减 1，继续下一个循环。若没有入度为 0 的节点，则说明有环，退出。

2. 算法实现

```
void TopoSort(){ //拓扑排序（逆向图）
   flag=0;
   for(int i=n;i>0;i--){
      int t=-1;
      for(int j=n;j>0;j--)
         if(!in[j]){
            t=j;
            break;
         }
      if(t==-1){//有环
         flag=1;
         return;
      }
      in[t]=-1;
      w[t]=i;
      for(int j=1;j<=n;j++)
         if(map[t][j])
            in[j]--;
   }
}
int main(){
   cin>>T;
   while(T--){
      memset(map,0,sizeof(map));
      memset(in,0,sizeof(in));
      cin>>n>>m;
      for(int i=1;i<=m;i++){
         cin>>u>>v;
         if(!map[v][u]){//建立逆向图，检查重复的边
            map[v][u]=1;
            in[u]++;
         }
      }
      TopoSort();
      if(flag){
         cout<<-1<<endl;
         continue;
      }
      for(int i=1;i<n;i++)
         cout<<w[i]<<" ";
      cout<<w[n]<<endl;
   }
   return 0;
}
```

POJ1270

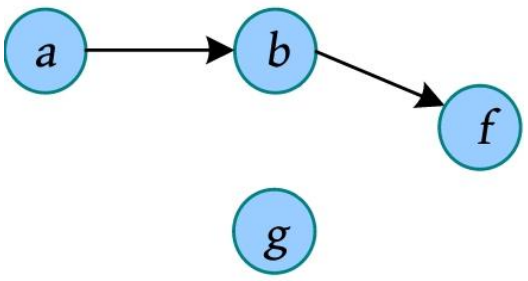
题目描述（POJ1270）：给定 x<y 形式的变量约束列表，编写程序，输出与约束一致的变量的所有顺序。例如，给定约束 x<y 和 x<z，变量 x、y 和 z 的两个排序与这些约束一致：xyz 和 xzy。

输入：输入由一系列约束规范组成。每个约束规范都由两行组成：一行为变量列表，后面一行为约束列表。约束由一对变量给出，其中 xy 表示 x<y。所有变量都是单个小写字母。在约束规范中至少有两个且不超过 20 个变量，至少有一个且不超过 50 个约束，至少有一个且不超过 300 个与约束规范中的约束条件一致的顺序。

输出：对每个约束规范，都以字典顺序单行输出与约束一致的所有排序。不同约束规范的输出以空行分隔。

输入样例	输出样例
a b f g a b b f v w x y z v y x v z v w v	abfg abgf agbf gabf wxzvy wzxvy xwzvy xzwvy zwxvy zxwvy

题解：根据输入样例 1，构建的图形结构如下图所示。



本题需要按照字典序输出所有拓扑序列，因此使用回溯法搜索所有拓扑序列。注意，到达叶子时输出，回溯时需要还原现场。

1. 算法设计

（1）将变量列表的字符转换为数字并统计出现次数，累计变量列表的长度。

（2）将每对约束都转换为数字，用邻接矩阵存储并统计入度。

（3）以回溯法求解所有拓扑序列并输出。

3. 算法实现

```
void dfs(int t){ //以回溯法求解所有拓扑序列
   if(t>=len){//到达叶子节点，输出一个拓扑序列
      for(int i=0;i<len;i++)
         cout<<char(ans[i]+'a');//转换为字符
      cout<<endl;
   }
   for(int i=0;i<26;i++){
      if(!in[i]&&s[i]){//i 的入度为 0 且该字符在变量列表中
         s[i]--;
         for(int j=0;j<26;j++)//将 i 的所有邻接点 j 的入度都减 1
            if(map[i][j])
               in[j]--;
         ans[t]=i;//记录拓扑序列的第 t 个字符为 i
         dfs(t+1);//深度优先搜索 t+1 个字符
         for(int j=0;j<26;j++)//回溯时还原现场，将 i 的所有邻接点 j 的入度都加 1
            if(map[i][j])
               in[j]++;
         s[i]++;//恢复该字符在变量列表中
      }
   }
}
```

```
int main(){
   while(getline(cin,str)){ //读入变量列表
      memset(map,0,sizeof(map));
      memset(in,0,sizeof(in));
      memset(s,0,sizeof(s));
      len=str.length();
      int i,j=0;
      for(i=0;i<len;i++){
         if(str[i]!=' '){
            s[str[i]-'a']++;//转换为数字统计
            j++;
         }
      }
      len=j;//变量列表的长度
      getline(cin,ord);//读入约束列表
      num=ord.length();
      for(i=0;i<num;i+=2){//有空格，一次读两个字符
      for(i=0;i<num;i+=2){//有空格，一次读两个字符
         int u=ord[i]-'a';
         i+=2;
         int v=ord[i]-'a';
         map[u][v]=1;//以邻接矩阵存储
         in[v]++;//入度加 1
      }
      dfs(0);//深度优先搜索（回溯法）
      cout<<endl;
   }
   return 0;
}
```