一个有向图能被拓扑排序的充要条件就是它是一个有向无环图

# 1285确定比赛名次

**Problem Description**

有N个比赛队（1<=N<=500），编号依次为1，2，3，。。。。，N进行比赛，比赛结束后，裁判委员会要将所有参赛队伍从前往后依次排名，但现在裁判委员会不能直接获得每个队的比赛成绩，只知道每场比赛的结果，即P1赢P2，用P1，P2表示，排名时P1在P2之前。现在请你编程序确定排名。

**Input**

输入有若干组，每组中的第一行为二个数N（1<=N<=500），M；其中N表示队伍的个数，M表示接着有M行的输入数据。接下来的M行数据中，每行也有两个整数P1，P2表示即P1队赢了P2队。

**Output**

给出一个符合要求的排名。输出时队伍号之间有空格，最后一名后面没有空格。  
其他说明：符合条件的排名可能不是唯一的，此时要求输出时编号小的队伍在前；输入数据保证是正确的，即输入数据确保一定能有一个符合要求的排名。

**Sample Input**

4 3

1 2

2 3

4 3

**Sample Output**

1 2 4 3

//直接法

#include <cstdio>

#include <cstring>

#define MAXN 517

int G[MAXN][MAXN];//路径

int in\_degree[MAXN];//入度

int ans[MAXN];

int n, m, x, y;

int i, j;

void toposort()

{

for(i = 1; i <= n; i++)

{

for(j = 1; j <= n; j++)

{

if(G[i][j])

{

in\_degree[j]++;

}

}

}

for(i = 1; i <= n; i++)//从最小的开始寻找，

{//这样保证了有多个答案时序号小的先输出

int k = 1;

while(in\_degree[k] != 0)//寻找入度为零的点

k++;

ans[i] = k;

in\_degree[k] = -1;

//更新为-1，后边检测不受影响，相当于删除节点

for(int j = 1; j <= n; j++)

{

if(G[k][j])

in\_degree[j]--;//相关联的入度减1

}

}

}

int main()

{

while(~scanf("%d%d",&n,&m))

{

memset(in\_degree,0,sizeof(in\_degree));

memset(ans,0,sizeof(ans));

memset(G,0,sizeof(G));

for(i = 0; i < m; i++)

{

scanf("%d%d",&x,&y);

G[x][y] = 1;

}

toposort();

for(i = 1; i < n; i++)

printf("%d ",ans[i]);

printf("%d\n",ans[n]);

}

return 0;

}

//拓扑排序+优先队列

#include<iostream>

#include<queue>

#include<functional>

#include<cstdio>

#include<cstring>

using namespace std;

bool map[517][517];

int in[517];

priority\_queue<int,vector<int>,greater<int> > q;

//从小到大的优先级队列，可将greater改为less，即为从大到小

void topo(int n)

{

for(int i=1;i<=n;i++)

{

if(in[i]==0)

q.push(i);

}

int c=1;

while(!q.empty())

{

int v=q.top();

q.pop();

if(c!=n)

{

cout<<v<<" ";

c++;

}

else

cout<<v<<endl;

for(int i=1;i<=n;i++)

{

if(!map[v][i])

continue;

in[i]--;

if(!in[i])

q.push(i);

}

}

}

int main()

{

int n,m,i,j;

while(cin>>n>>m)

{

int k=0;

memset(map,0,sizeof map);

memset(in,0,sizeof in);

while(m--)

{

cin>>i>>j;

if(map[i][j])

continue;

map[i][j]=1;

in[j]++;

}

topo(n);

}

}