假设一个有向图G采用邻接矩阵存储，分别设计实现以下要求的算法：

求出图G中每个顶点的入度

求出图G中每个顶点的出度

求出图G中出度最大的一个顶点，并输出该顶点编号。

计算图G中出度为0的顶点数

判断图G中是否存在边<i,j>

思路：用一个二维数组graph[][]存储这个邻接矩阵，

由于矩阵中只存在0和1，所以点i的入度=i列元素之和；点i的出度=i行元素之和。在求每个点的出度时，设一个max\_outdegree\_point来记录出度最大的点的编号，设一个zero\_outdegree\_count记录出度为0的顶点数。若存在边<i,j>，则graph[i][j]=1,否则为0，据此来判断图G中是否存在边<i,j>。

输入：

第一行输入两个数num，n。num表示图G有num个点，n表示下面要输入n行信息。输入保证i,j<=num。

（n,m<=1000）

接下来的n行，每行输入两个数字i,j，表示图G有一条由点i指向点j的有向边。

接着输入一个数m，m表示要询问m次图G中是否存在边<i,j>。

再接下来的m行，每行输入两个数字i,j，表示询问图G中是否存在边<i,j>。

输入保证i,j<=num。

注：n,m<=1000

输出：

输出图G中每个顶点的入度；

输出图G中每个顶点的出度；

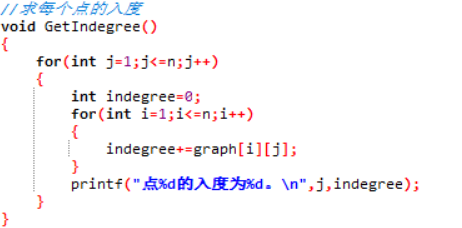
输出图G中出度最大的顶点的编号；

输出图G中出度为0的顶点数；

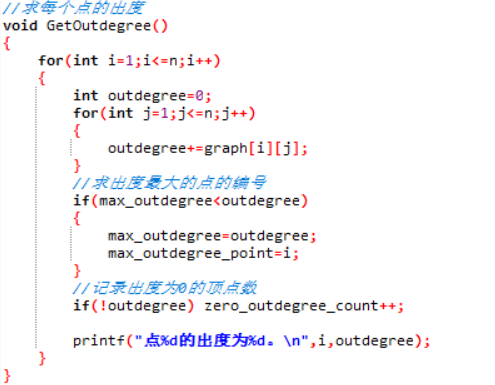
后面对于每次询问，若存在该边，则输出yes；否则输出no。

代码：

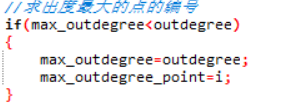
1. 求每个顶点的入度：



1. 求每个顶点的出度：



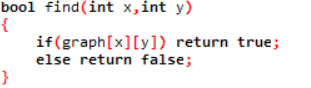
1. 求图G中出度最大的顶点的编号：

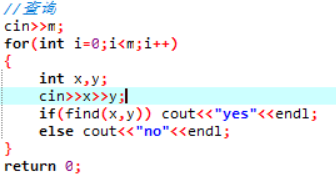


1. 求图G中出度为0的顶点数：



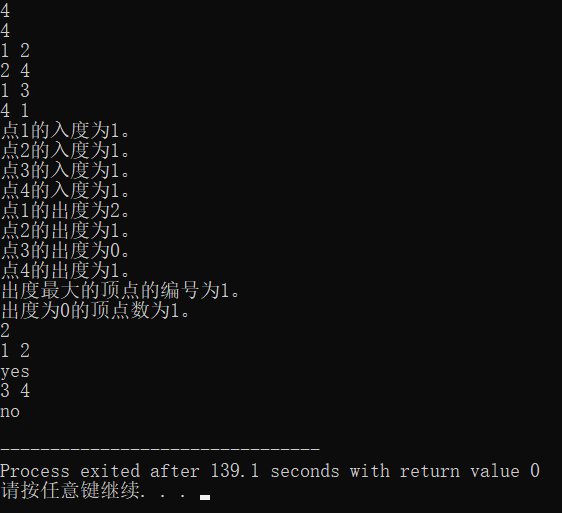
1. 对于每次询问，若存在该边，则输出yes；否则输出no。





附件：

1. 示例图G：
2. 程序截图：



1. 源代码：

#include<bits/stdc++.h>

using namespace std;

#define max 1005

int graph[max][max]={0};//初始化

int num,n,m;

int max\_outdegree\_point,max\_outdegree=0,zero\_outdegree\_count=0;

//求每个点的入度

void GetIndegree()

{

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

{

int indegree=0;

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

{

indegree+=graph[i][j];

}

printf("点%d的入度为%d。\n",j,indegree);

}

}

//求每个点的出度

void GetOutdegree()

{

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

{

int outdegree=0;

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

{

outdegree+=graph[i][j];

}

//求出度最大的点的编号

if(max\_outdegree<outdegree)

{

max\_outdegree=outdegree;

max\_outdegree\_point=i;

}

//记录出度为0的顶点数

if(!outdegree) zero\_outdegree\_count++;

printf("点%d的出度为%d。\n",i,outdegree);

}

}

//询问函数

bool find(int x,int y)

{

if(graph[x][y]) return true;

else return false;

}

int main()

{

cin>>num>>n;

for(int i=0;i<n;i++)

{

int x,y;

cin>>x>>y;

graph[x][y]=1;

}

//邻接矩阵完成建立

GetIndegree();

GetOutdegree();

printf("出度最大的顶点的编号为%d。\n",max\_outdegree\_point);

printf("出度为0的顶点数为%d。\n",zero\_outdegree\_count);

//查询

cin>>m;

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

{

int x,y;

cin>>x>>y;

if(find(x,y)) cout<<"yes"<<endl;

else cout<<"no"<<endl;

}

return 0;

}