### 判断是否有环 从文件读取，图用邻接矩阵储存

#### 代码：

#include<bits/stdc++.h>

using namespace std;

const int MAX\_NUM=105;

class openFile { //打开文件的class

private:

public:

fstream io;

openFile(string filename) {//需要指定文件名

io.open(filename.c\_str(),ios::in|ios::app);//以读取和添加的方式打开

}

~openFile() {

io.close();

}

};

class Bin\_fun { //将读取的字符串形式转化为两个元素字符串

public:

string x,y,s;//s为原子符串，x为第一元素，y为第二元素

Bin\_fun(string newS) { //new 为a={(1,2),(2,3)...}

s=newS;

int pos=-1;

for(int i=0; i<s.size(); i++)

if(s[i]>='0'&&s[i]<='9') {

pos=i;

break;

}

for(int i=pos; i<s.size(); i+=6) {

x+=s[i];

y+=s[i+2];

}

//cout<<x<<endl;

}

};

struct matrix {//邻接矩阵储存

string node;//节点信息

int arcs[MAX\_NUM][MAX\_NUM];//矩阵

int node\_num, edge\_num;//节点数,边数

} Graph;

void read\_graph() {

openFile No\_vector("No\_vector\_graph.txt");

string s;

No\_vector.io>>s;

for(int i=0; i<s.size(); i++)

if(s[i]>='0'&&s[i]<='9')//节点为数字

Graph.node+=s[i];

string s1;

No\_vector.io>>s1;

Bin\_fun E(s1.c\_str());//转换

memset(Graph.arcs,0,sizeof(Graph.arcs));//将矩阵初始化 为0

for(int i=0; i<E.x.size(); i++) {//cout<<"ok"<<endl;

Graph.arcs[E.x[i]-1-'0'][E.y[i]-1-'0']++;//矩阵从零开始图节点从1开始

Graph.arcs[E.y[i]-1-'0'][E.x[i]-1-'0']++;//因为是无向图所以反向也有要记录

}

Graph.node\_num=Graph.node.size();

}

//从某点出判断无向图是否有回路（图用邻接矩阵表示）不考虑孤立点调用一次就够了

bool isClose(int begin, int temp[][MAX\_NUM]) {

bool flag[MAX\_NUM];

int i, j;

stack <int>stk;

queue <int> vexQueue;

for (i = 0; i<Graph.node\_num; i++)

flag[i] = false;

//从a或b点出发，广度遍历所有点

//如果该点和之前已访问过的点（除了它父亲）有边，

//说明有回路

vexQueue.push(begin);

flag[begin] = true;

int father;

//

for(int i=0;i<Graph.node.size();i++)

for(int j=0;j<Graph.node.size();j++)

if(temp[i][j]>1/\*无向图中两节点的回路\*/||temp[i][j]!=0&&i==j/\*自身的回路\*/)//

return true;

//

while (vexQueue.empty() == false) {

father = vexQueue.front();

vexQueue.pop();

for (int i = 0; i < Graph.node\_num; ++i) {

if (temp[father][i] != 0 && flag[i] == false) {

vexQueue.push(i);

flag[i] = true;

//看看当前点和其他已访问点是否有边

for (int j = 0; j < Graph.node\_num; ++j) {

if (temp[i][j] != 0 && j != father && flag[j] == true) {

return true;//如果有边，说明有回路

}

}

}

}

}

return false;

}

void print\_graph(){

for(int i=0;i<Graph.node\_num;i++){

for(int j=0;j<Graph.node.size();j++)

cout<<Graph.arcs[i][j]<<" ";

cout<<endl;

}

}

int main() {

read\_graph();

print\_graph();

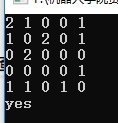
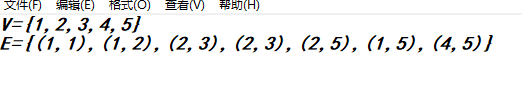
if(isClose(0,Graph.arcs))

cout<<"yes"<<endl;

else cout<<"no" <<endl;

}

#### 运行图：



### 计算前驱后续邻域

#### 代码：

#include<bits/stdc++.h>

using namespace std;

const int MAX\_NUM=105;

class openFile { //打开文件的class

private:

public:

fstream io;

openFile(string filename) {//需要指定文件名

io.open(filename.c\_str(),ios::in|ios::app);//以读取和添加的方式打开

}

~openFile() {

io.close();

}

};

class Bin\_fun { //将读取的字符串形式转化为两个元素字符串

public:

string x,y,s;//s为原子符串，x为第一元素，y为第二元素

Bin\_fun(string newS) { //new 为a={(1,2),(2,3)...}

s=newS;

int pos=-1;

for(int i=0; i<s.size(); i++)

if(s[i]>='0'&&s[i]<='9') {

pos=i;

break;

}

for(int i=pos; i<s.size(); i+=6) {

x+=s[i];

y+=s[i+2];

}

//cout<<x<<endl;

//cout<<y<<endl;

}

};

string get\_D\_(int n){//T+ D(d)后续

openFile link("link.txt");

string s;

link.io>>s;

string s1;

link.io>>s1;

Bin\_fun E(s1.c\_str());//转换

string ans("{") ,ansTemp;//ansTemp用于返回有价值的量 ans用于记录字符结果带 {}

for(int i=0;i<E.x.size();i++)

if(E.x[i]-'0'==n&&ans.find(E.y[i])==string::npos&&n!=E.y[i]-'0'){//cout<<"ok"<<endl;

ans+=E.y[i];

ans+=',';

ansTemp+=E.y[i];

}

ans.replace(ans.size()-1,1,"}");//将多余的，号替换

cout<<"T+=" <<ans<<endl;

link.io<<"T+=" <<ans<<endl;

return ansTemp;

}

string get\_D(int n){//T- D(d)前驱

openFile link("link.txt");

string s;

link.io>>s;

string s1;

link.io>>s1;

Bin\_fun E(s1.c\_str());//转换

string ans("{") ,ansTemp;//ansTemp用于返回有价值的量 ans用于记录字符结果带 {}

for(int i=0;i<E.y.size();i++)

if(E.y[i]-'0'==n&&ans.find(E.x[i])==string::npos&&n!=E.x[i]-'0'){

ans+=E.x[i];

ans+=',';

ansTemp+=E.x[i];

}

ans.replace(ans.size()-1,1,"}");//将多余的，号替换

cout<<"T-=" <<ans<<endl;

link.io<<"T-=" <<ans<<endl;

return ansTemp;

}

string ND\_d(int n){//邻域

openFile link("link.txt");

string ans("{"),ansTemp;

string s1=get\_D(n),s2=get\_D\_(n);

for(int i=0;i<s1.size();i++){

if(ans.find(s1[i])==string::npos)

ans+=s1[i];

ans+=',';

ansTemp+=s1[i];

}

for(int i=0;i<s2.size();i++){

if(ans.find(s2[i])==string::npos)

ans+=s2[i];

ans+=',';

ansTemp+=s2[i];

}

ans.replace(ans.size()-2,2,"}");//将多余的，号替换

cout<<"ND\_d="<<ans<<endl;

link.io<<"ND\_d="<<ans<<endl;

return ans;

}

void ND\_d\_(int n){//闭邻域

openFile link("link.txt");

string ans=ND\_d(n);

ans.insert(ans.size()-1,1,',');

ans.insert(ans.size()-1,1,n+'0');

cout<<"ND\_d="<<ans<<endl;

link.io<<"ND\_d="<<ans<<endl;

}

int main() {

ND\_d\_(4);

}

#### 运行图

