**用dfs判断一个有向图是否有环**

解决这个问题的算法的思路是对一个节点u进行dfs，判断是否能从u回到自己这个节点，即是否存在从u到u的回路。

我们可以用一个color数组代表每个结点的状态，-1代表还没被访问，0代表正在被访问，1代表访问结束

如果一个状态为“0”的结点，与他相连的结点状态也为“0”的话就代表有环,这个可以用dfs实现

#include <iostream>

#include <vector>

#include <string.h>

using namespace std;

/\*

color代表每个结点的状态，-1代表还没被访问，0代表正在被访问，1代表访问结束

如果一个状态为“0”的结点，与他相连的结点状态也为0的话就代表有环,这个可以用dfs实现

\*/

vector<int> vec[1200];

int color[1200];

bool flag;//标记是否有环

void dfs(int x){

if(flag){//如果有环就返回，否者继续收索

return;

}

color[x] = 0;//x正在被反问，状态为0

for(int i = 0 ; i < vec[x].size() ; i++){

if(color[vec[x][i]] == -1){//与x相连的结点状态也为-1，代表还未被访问，则继续搜索

dfs(vec[x][i]);

} else if(color[vec[x][i]] == 0){//与x相连的结点状态也为0，代表有环，返回

flag = true;

return;

}

}

color[x] = 1;//对x的访问结束

}

int main(){

memset(color, -1, sizeof(color));

int n , m;

cin >> n >> m;//n个结点，m条边

for(int i = 0 ; i < m ; i++){//建立邻接表

int u,v;

cin >> u >> v;

vec[u].push\_back(v);

}

flag = false;

dfs(1);

if(flag){

cout<<" have "<<endl;

} else {

cout << " not have" <<endl;

}

return 0;

}

/\*

int n,m;

for(i -> m)

vec[u].p(v)

dfs(x)

if(flag) return

color x <- 0;

for i -> size

if color u = -1

dfs u

else if color u = 0

flag = true

return

color x = 1

\*/