深度优先遍历

1. 代码实现

#include <iostream>

#include <stack>

using namespace std;

#define MaxNode 20

#define MAX 2000

#define StartNode 1

int map[MaxNode+1][MaxNode+1];

void dfs\_stack(int start, int n){

int visited[MaxNode],s\_top;

for(int i = 0;i <= MaxNode; i++){

visited[i] = 0;

}

visited[start] = 1;

stack <int> s;

cout<<start<<" ";

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

if(map[i][start] == 1 && !visited[i] ){

visited[i] = 1;

s.push(i);

}

}

while(!s.empty()){

s\_top = s.top();

visited[s\_top] = 1;

cout<<s\_top<<" ";

s.pop();

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

if(map[i][s\_top] == 1 && !visited[i] ){

visited[i] = 1;

s.push(i);

}

}

}

}

int main(int argc, const char \* argv[]) {

int num\_edge,num\_node;

int x,y;

cout<<"Input number of nodes and edges >"<<endl;

cin>>num\_node>>num\_edge;

for(int i =0;i<num\_node;i++){

for(int j=0;j<num\_node;j++){

map[i][j] = 0;

}

}

for(int i = 1; i <= num\_edge; i++){

cin>>x>>y;

map[x][y] = map[y][x] = 1;

}

dfs\_stack(StartNode, num\_node);

return 0;

}

运行结果

