**实验三** DFS（深度优先搜索算法）

171491328 罗晓璐

一、内容介绍：

深度优先搜索算法：一种用于遍历或搜索树或图的算法。沿着树或图的深度遍历树的节点，尽可能深的搜索树的分支。当节点v的所在边都已被探寻过或在搜索时节点不满足条件，搜索将回到发现节点v的那条边的起始节点。整个过程反复进行直到所有节点都被访问为止。属于盲目搜索。

**基本思想：**

（1）访问顶点v；

（2）从v的未被访问的邻接点中选取一个顶点w，从w出发进行深度优先遍历；

（3）重复上述两步，直至图中所有和v有路径相通的顶点都被访问到。

**算法复杂度：**

若有v个顶点、E条边，则

用邻接表储存图，有O(V+E)

用邻接矩阵储存图，有O(V^2)

深度搜索：时间复杂度大，空间复杂度小。

二、实验过程：

1、算法伪代码：

bool visited[MAXNODE]; //顶点的访问标识数组

void DFSInit(Graph G){

for(i=0; i<G.VertexNum; i++){

visited[i] = false;

}

}

void DFS(Graph G,int v){ //v:顶点数组中的序号

Visit[v]; visited[v]=true;

w = FirstAdj(G,v); //返回：v的第一个邻接点，0表示无邻接点

while(w!=0){

if(!visited[w]{

DFS(G,w); //参数传递w->v

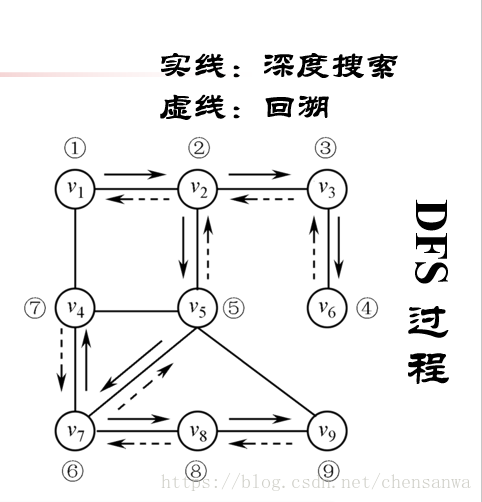
}

w = NextAdj(G,v,w); //返回：v的在邻接点w后的邻接点，0表示不存在

}

}

2、图解



 存储结构不确定，遍历结果不唯一

 其中一种DFS序列：DFS(G,v1) = (v1,v2,v3,v6,v5,v7,v4,v8,v9)

3、从一个题来诠释深度优先搜索算法

输入数据（图）：

0 0 1 1 0

1 0 0 1 1

1 1 0 0 0

1 0 0 1 1

0 0 1 0 0

‘0’代表能够走的，‘1’代表不能走的，可以理解为墙，同时只能在所给的地图上走，也就是不能跳出去再调回来。从起点（0，0）开始走，输出能到达的区域。

算法：

#include <iostream>

#include <cstring>

#define N 10

using namespace std;

int dir[4][2] = {0,1,0,-1,1,0,-1,0};

int vst[N][N]; //标记数组

int map[N][N]; //给定图

bool checkEdge(int x,int y);

void dfs(int x,int y);

int main(){

int n;

cin>>n;

memset(vst,0,sizeof(vst)); //初始化

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

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

for(int j=0;j<n;j++){

cin>>map[i][j]; //输入图

}

}

dfs(0,0); //进行DFS

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

for(int j=0;j<n;j++){

// cout<<vst[i][j]<<" ";

if(vst[i][j] == 6){ //找寻到的结果，

cout<<"i:"<<i<<" j:"<<j<<endl;

}

}

cout<<endl;

}

return 0;

}

bool checkEdge(int x,int y){ //检验当前点是否可走

if(vst[x][y]==0 && map[x][y]==0 && x>=0 && x<55 && y>=0 && y<5){

return true;

}

return false;

}

void dfs(int x,int y){

vst[x][y] = 6;

for(int i=0;i<4;i++){

if(checkEdge(x+dir[i][0],y+dir[i][1])){

dfs(x+dir[i][0],y+dir[i][1]);

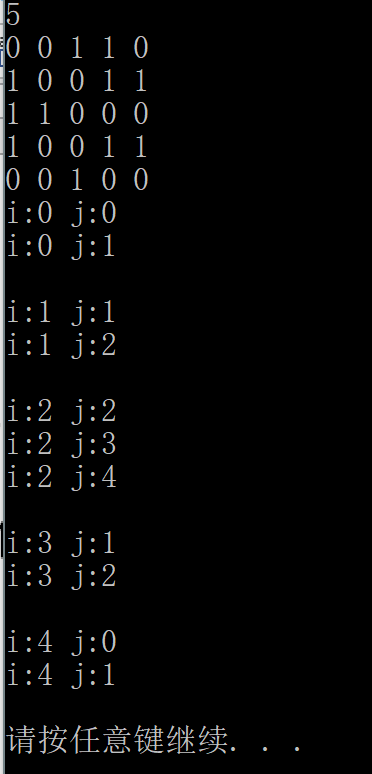
}

}

return;

}

结果：



三、总结

DFS其实就是找准一条路径，一直走下去（深度优先），直到满足条件或者走投无路、走不下去。如果没有找到相应的结果状态，那么就回退一步（回溯），再进行下一步找路径，再撞再回溯，一直到找到目标状态或者所有情况都遍历完，程序结束。