数据结构课程设计



班级： 1618403

学号： 161840227

姓名： 韦 鑫

指导教师：孙 涵

目录

1.采用的数据结构 ………………………………………… 3

2.算法设计思想 …………………………………………… 3

3.关键代码 ………………………………………………… 3

4.测试数据和结果 ………………………………………… 9

5.算法的时间复杂度及其改进方法 ……………………… 9

6.结束语 …………………………………………………… 9

1. 采用的数据结构

使用了栈对坐标点信息存储

//坐标点结构体

struct pos

{

int x, y; //坐标

int dir; //方向

int step; //步数

};

二、算法设计思想

①非递归DFS寻找从迷宫起点到终点的最优路径：

a.若在当前位置的下一个位置可达，则将下一个位置入栈

若下个位置到达终点且当所走步数更小，即有更优的路线，则将栈中的坐标点记录到path栈中，并将栈顶坐标弹出，修改栈顶坐标朝向后继续循环。

b.若当前位置的下一个位置不可达，则将当前位置的坐标朝向修改加一，即转向，继续循环

三、关键代码

const int maxn = 10000;

const int inf = 1 << 30 - 1;

int maze[maxn][maxn] = {0};//存储迷宫地图，1表示可走，0表示障碍物

int n, m;//迷宫地图行、列值

//坐标点结构体

struct pos

{

int x, y; //坐标

int dir; //方向

int step; //步数

};

pos S, T;//起点、终点

//方向数组

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

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

//是否走过该位置

bool vis[maxn][maxn] = { 0 };

//读取迷宫地图

void read()

{

FILE \*fp;

if ((fp = fopen("maze.txt", "r")) == NULL)

{

printf("读取迷宫地图失败！\n");

exit(-1);

}

char line[maxn];

int row = 1;

fscanf(fp, "%s", line);

m = strlen(line);//迷宫列数

while(!feof(fp))

{

int col = 1;

for (int i = 0; i < strlen(line); i++)

{

if (line[i] == '.')maze[row][col] = 1;

col++;

}

fscanf(fp, "%s", line);

row++;

}

n = row - 1;//迷宫行数

fclose(fp);

}

//判断坐标是否出界、存在障碍物

bool judge(int x, int y)

{

if (x < 1 || x > n || y < 1 || y > m)return false;

if (vis[x][y])return false;

if (!maze[x][y])return false;

return true;

}

//非递归DFS寻最短路径

void dfs()

{

pos path[maxn]; //存放最优路径

pos stack[maxn];//定义栈

int top = -1,ttop = -1, ptop = -1;;

int step = inf; //从起点到终点最少需要走的步数

stack[++top] = S;

vis[S.x][S.y] = 1;

while (top >= 0)

{

pos t = stack[top];

pos next;

next.dir = 0;

next.step = t.step + 1;

next.x = t.x + dx[t.dir];

next.y = t.y + dy[t.dir];

if (judge(next.x, next.y))//若下个位置可达，则入栈

{

stack[++top] = next;

if (next.x == T.x && next.y == T.y)//若到达终点

{

if (next.step < step)//若有更优的路径，则存入path栈

{

step = next.step;

ttop = top;

ptop = -1;

while (ttop >= 0)

{

path[++ptop] = stack[ttop--];

}

}

top--;//出栈

t = stack[top];//更新栈顶方向

t.dir++;

stack[top] = t;

continue;

}

else

{

vis[next.x][next.y] = 1;

continue;//若未到达终点，继续取栈顶位置获取下一个位置

}

}

else//若下个位置不可达，则换个方向

{

t.dir++;

stack[top] = t;

while (t.dir >= 4)

{

top--;

vis[t.x][t.y] = 0;

if (top < 0) break;

else//更新栈顶位置方向

{

t = stack[top];

t.dir++;

stack[top] = t;

}

}

}

}

//输出路径

printf("从 起点( %d,%d ) 最少需要走 %d 步可达 终点( %d,%d ) \n", S.x, S.y, step, T.x, T.y);

printf("路径为：\n");

while (ptop >= 0)

{

if (ptop != 0) printf("( %d,%d ) - > ", path[ptop]);

else printf("( %d,%d )\n", path[ptop]);

ptop--;

}

}

//以0，1的形式输出地图。0表示存在障碍物

void print\_maze()

{

//横坐标轴

printf(" ");

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

{

printf("%-3d ", i);

}

printf("\n");

printf(" ");

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

{

printf("——");

}

printf("\n");

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

{

printf("%3d|", i);

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

{

printf("%3d ", maze[i][j]);

}

printf("\n");

}

printf("\n");

}

int main()

{

read();

print\_maze();

while (1)

{

int sx, sy;//起点坐标

int tx, ty;//终点坐标

printf("迷宫地图行数 1 < x < %d，列数 1 < y < %d\n", n, m);

printf("请选择没有障碍物的坐标点作为 起点 和 终点\n");

printf("起点坐标(x,y)：");

scanf("%d %d", &sx, &sy);

printf("终点坐标(x,y)：");

scanf("%d %d", &tx, &ty);

if (!judge(sx, sy) && !judge(tx, ty))

{

printf("起点、终点输入有误，请重新输入！\n\n");

continue;

}

else if (!judge(sx, sy) && judge(tx, ty))

{

printf("起点输入有误，请重新输入！\n\n");

continue;

}

else if (judge(sx, sy) && !judge(tx, ty))

{

printf("终点输入有误，请重新输入！\n\n");

continue;

}

else

{

S.x = sx;

S.y = sy;

S.dir = 0;

S.step = 0;

T.x = tx;

T.y = ty;

T.dir = 0;

T.step = 0;

break;

}

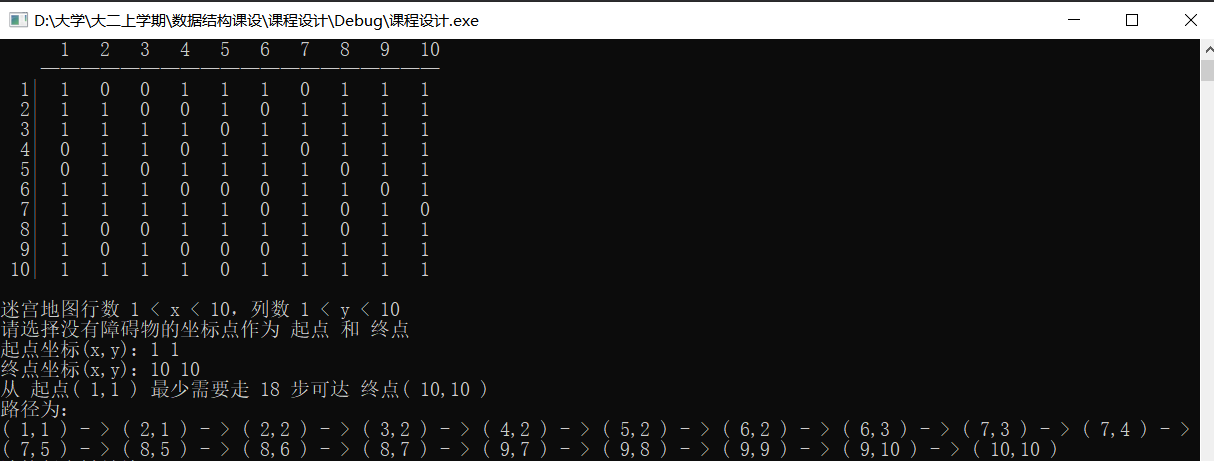
}

dfs();

system("pause");

return 0;

}

四、测试数据和结果

五、算法的时间复杂度即改进方法

时间复杂度为O(n\*m)，n、m为迷宫的长、宽

六、结束语

代码共227行

在做该实验前，解决迷宫问题时我都会使用DFS递归算法或者BFS对该问题进行求解，原因是递归算法和BFS算法书写简单，清晰易懂，缺点就是递归层数过大时会有系统栈溢出的风险；在解决本实验时，我第一次尝试着使用非递归DFS算法、以栈模拟递归过程求解。原以为会被该题的非递归算法绕脑的我通过先在草稿纸上书写思路和伪代码，再将其以代码形式实现，以这样一个过程求解该题让我的思维更加清晰。完成这道题后，让我对非递归问题也没有原先那样恐惧了。