# **实验二 实验报告**

**一、实验题目：**

迷宫求解

**二、实验目的：**

在随机生成的迷宫中寻找通路，以此熟悉栈的使用

**三、实验要求**

**1、**随机生成迷宫，以左上角为起点，右下角为终点。

**2、**规定迷宫大小为N\*N,无其他输入。

**3、**输出：若迷宫存在的通路，则输出，若不存在，在输出“无解”。

**四、实验内容和实验步骤**

**1.需求分析：**

**输入形式：**无输入 ，约定起点终点并由程序随机生成迷宫

**输出形式：**可视化输出迷宫路径

**程序所能实现的功能：**在随机生成的迷宫中中寻找通路并输出

**2.** **概要设计：**

**（1）**建立结点类Step，包含每走一步需储存的信息：结点所在坐标，已走过的步数；

**（2）**建立迷宫类Maze，维护迷宫数组的信息与输出；

**（3）**建立算法类Path，实现路径搜索。

算法描述：

A. 首先设置一个最大步数i，当本次搜索步数超过i时，便停止本次搜索，回到初始结点将上限i++。这种迭代加深的搜索方法能保证在深度优先搜索的同时找到最优解。

**B**.将初始结点压栈并开始以下循环

取出栈顶元素，若它不是目标结点且步数未达上限i，则将它扩展为可走的若干子节点并全部压栈。循环执行C。

**C**.若找到目标节点则将结果返回。

**D**.若所有分支搜索结束步数都达到上限i，则将上限i++，返回B从初始结点重新搜索。

#include<cstdlib>  
#include<cstring>  
#include<iostream>  
using namespace std;  
char map[9][9];//输入的迷宫矩阵  
int dir[4][2] = {{1, 0}, {0, 1}, {-1, 0}, {0, -1}};//4个方向  
int OK = 0;  
int N, M, T, si, sj, di, dj;  
int dfs(int si, int sj, int cnt)  
{  
 if (si <= 0 || sj <= 0 || si > N || sj > M)//超出边界就说明这条路已经死了，则返回  
 {  
 return 0;  
 }  
 if (si == di && sj == dj && cnt == T)//找到终点就返回，把标志位置为1  
 {  
 OK = 1;  
 }  
 if (OK)  
 {  
 return 1;  
 }  
 int temp = T - cnt - abs(di - si) - abs(dj - sj);//这里就是剪枝，开始没写这里，Time Limit Exceeded了十几次...下面细谈  
 if (temp < 0 || temp & 1)  
 {  
 return 0;  
 }  
 for (int i = 0; i < 4; ++i)//对走到的每个结点都进行四个方向的探索  
 {  
 if (map[si+dir[i][0]][sj+dir[i][1]] != 'X')  
 {  
 map[si+dir[i][0]][sj+dir[i][1]] = 'X';//走过的路不能走，就先置为墙  
 dfs(si+dir[i][0], sj+dir[i][1], cnt + 1);  
 map[si+dir[i][0]][sj+dir[i][1]] = '.';//探索下一条路时，这个结点要恢复成可以走的状态  
 }  
 }  
 return 0;  
}  
int main()  
{  
 while(cin >> N >> M >> T)  
 {  
 int wall = 0;  
 OK = 0;  
 if (N == 0 && M == 0 && T == 0)  
 {  
 break;  
 }  
 for (int i = 1; i <= N; ++i)  
 {  
 for (int j = 1; j <= M; ++j)  
 {  
 cin >> map[i][j];//这里开始还写的scanf("%c", &map[i][j]);蠢的不谈了... 不过可以这样在for(i)的循环里面写 scanf("%s", &map[i]);  
 if (map[i][j] == 'S')  
 {  
 si = i;  
 sj = j;  
 }else if (map[i][j] == 'D')  
 {  
 di = i;  
 dj = j;  
 }else if (map[i][j] == 'X')  
 {  
 wall++;  
 }  
 }  
 }  
 if (N \* M - wall <= T)//一个小剪枝  
 {  
 cout << "NO" << endl;  
 continue;  
 }  
 map[si][sj] = 'X';  
 dfs(si, sj, 0);  
 if (OK)  
 {  
 cout << "YES" << endl;  
 }else  
 {  
 cout << "NO" << endl;  
 }  
 }  
 return 0;  
}