**学霸的迷宫**

问题描述

　　学霸抢走了大家的作业，班长为了帮同学们找回作业，决定去找学霸决斗。但学霸为了不要别人打扰，住在一个城堡里，城堡外面是一个二维的格子迷宫，要进城堡必须得先通过迷宫。因为班长还有妹子要陪，磨刀不误砍柴功，他为了节约时间，从线人那里搞到了迷宫的地图，准备提前计算最短的路线。可是他现在正向妹子解释这件事情，于是就委托你帮他找一条最短的路线。

输入格式

　　第一行两个整数n， m，为迷宫的长宽。  
　　接下来n行，每行m个数，数之间没有间隔，为0或1中的一个。0表示这个格子可以通过，1表示不可以。假设你现在已经在迷宫坐标(1,1)的地方，即左上角，迷宫的出口在(n,m)。每次移动时只能向上下左右4个方向移动到另外一个可以通过的格子里，每次移动算一步。数据保证(1,1)，(n,m)可以通过。

输出格式

　　第一行一个数为需要的最少步数K。  
　　第二行K个字符，每个字符∈{U,D,L,R},分别表示上下左右。如果有多条长度相同的最短路径，选择在此表示方法下字典序最小的一个。

样例输入

Input Sample 1:  
3 3  
001  
100  
110  
  
Input Sample 2:  
3 3  
000  
000  
000

样例输出

Output Sample 1:  
4  
RDRD  
  
Output Sample 2:  
4  
DDRR

数据规模和约定

　　有20%的数据满足：1<=n,m<=10  
　　有50%的数据满足：1<=n,m<=50  
　　有100%的数据满足：1<=n,m<=500。

C++代码：

#include <stdio.h>

#include <iostream>

#include <string.h>

#include <math.h>

#include <ctype.h>

#include <algorithm>

#include <map>

#include <math.h>

#include <stack>

#include <queue>

#define Max 510

#define inf 100000000

using namespace std;

int vis[Max][Max],r,l;

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

char str[5] = "UDLR",m[Max][Max],step[Max];

struct Node{

int x;

int y;

int s;

char sp;

}path[Max][Max];

int BFS(){

Node now;

memset(vis, 0, sizeof(vis));

now.x = now.y =0;

path[now.x][now.y].s = 0;

path[now.x][now.y].sp = 0;

queue<Node>q;

q.push(now);

while (q.size()>0) {

now = q.front();

if (now.x == r-1 && now.y == l-1) {

break;

}

q.pop();

Node next = now;

//printf("now:x %d y %d step %d straight %c\n",now.x,now.y,path[now.x][now.y].s,path[now.x][now.y].sp);

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

next.x = now.x+Mov[i][0];

next.y = now.y+Mov[i][1];

if (!vis[next.x][next.y]&&(next.x>=0&&next.x<r&&next.y>=0&&next.y<l)&& m[next.x][next.y] != '1' ) {

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

path[next.x][next.y] = now;

path[next.x][next.y].sp = str[i];

path[next.x][next.y].s = path[now.x][now.y].s+1;

q.push(next);

// printf("next:x %d y %d step %d straight %c\n",next.x,next.y,path[next.x][next.y].s,path[next.x][next.y].sp);

}

}

}

//printf("%d\n",path[r-1][l-1].s);

return path[r-1][l-1].s;

}

int main(){

while (scanf("%d%d",&r,&l)!=EOF) {

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

scanf("%s",m[i]);

}

printf("%d\n",BFS());

int len = 0;

int nx = r-1,ny = l-1;

int nextx = nx,nexty = ny;

while (1) {

nx = nextx;

ny = nexty;

if (nx + ny == 0) {

break;

}

step[len++] = path[nx][ny].sp;

nextx = path[nx][ny].x;

nexty = path[nx][ny].y;

}

for (int i=len-1; i>=0; i--) {

printf("%c",step[i]);

}

printf("\n");

}

return 0;

}

C代码：

#include <stdio.h>

#include <string.h>

struct node{

int x;

int y;

int s;

int d;

int p;

};

int main(){

char map[501][501], path[2500] = "";

int book[501][501] = {0};

int n, m, head = 1, tail = 1, flag = 0;

struct node que[2501];

int next[4][3] = {

{-1, 0, 'U'},//上U

{1, 0, 'D'},//下D

{0, -1, 'L'},//左L

{0, 1, 'R'}//右R

};

int i, j,tx, ty;

scanf("%d %d", &n, &m);

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

scanf("%s", map[i]);

}

que[tail].x = 1;

que[tail].y = 1;

que[tail].s = 0;

que[tail].p = 0;

tail++;

book[1][1] = 1;

while(head < tail){

if(flag){

break;

}

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

tx = que[head].x + next[i][0];

ty = que[head].y + next[i][1];

if(tx < 1 || ty < 1 || tx > n || ty > m){

continue;

}

if(book[tx][ty] == 0 && map[tx - 1][ty - 1] == '0'){

book[tx][ty] = 1;

que[tail].x = tx;

que[tail].y = ty;

que[tail].s = que[head].s + 1;

que[tail].d = next[i][2];

que[tail].p = head;

tail++;

}

if(tx == n && ty == m){

flag = 1;

break;

}

}

head++;

}

i = tail - 1;

j = 0;

while(que[i].p != 0){

path[j++] = que[i].d;

i = que[i].p;

}

printf("%d\n", que[tail - 1].s);

for(j--; j >= 0; j--){

printf("%c", path[j]);

}

return 0;

}

JAVA代码：

import java.util.ArrayList;

import java.util.List;

import java.util.Queue;

import java.util.Scanner;

import java.util.concurrent.LinkedBlockingQueue;

public class Main {

static class Point{

int x;

int y;

Point(int x,int y){

this.x = x;

this.y = y;

}

}

public static String mazeBFS(byte[][] map, int n, int m) {

Point start = new Point(0,0);

Queue<Point> q = new LinkedBlockingQueue<Point>();

q.add(start);

map[0][0] = 2;

List<Integer> list = new ArrayList<Integer>();

StringBuilder sb = new StringBuilder();

StringBuilder result = new StringBuilder();

int index = -2;

int[] x = new int[]{1,0,-1,0};

int[] y = new int[]{0,1,0,-1};

char[] dir = new char[]{'D','R','U','L'};

int tempx,tempy;

while(!q.isEmpty()){

if(index==n\*m-2)

break;

Point data = q.remove();

index++;

if(data.x==n-1 && data.y==m-1)

break;

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

tempx = data.x+x[i];

tempy = data.y+y[i];

if(tempx>=0 && tempx<n && tempy>=0 && tempy<m && map[tempx][tempy]==0){

Point temp = new Point(tempx, tempy);

q.add(temp);

map[tempx][tempy] = 2;

list.add(index);

sb.append(dir[i]);

}

}

}

while(index!=-1){

result.append(sb.charAt(index));

index=list.get(index);

}

return result.reverse().toString();

}

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int n = in.nextInt();

int m = in.nextInt();

byte[][] map = new byte[n][m];

String s;

in.nextLine();

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

s = in.nextLine();

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

map[i][j] = (byte) (s.charAt(j)-'0');

}

}

String result = mazeBFS(map, n, m);

System.out.println(result.length());

System.out.println(result);

}

}