Zoj2412Farm IrrigationDFS

**题意**：有11种正方形，每种正方形里面对应一种形状的水管，不同的的正方形一用A到K表示，给一个矩阵，问至少需要多少个水源可以使矩形中所有的地方都可以被灌溉，如果两个相邻的正方形的水管正好对口，那么这两个正方形可以共用一个水源。

**分析**：

首先，处理11种不同的水管，分4个方向，1表示有接口，0表示没有接口。用一个二维数组存所有种类的水管。

然后，把输入的字符转化成数字，可以再输入的时候边输入边处理，用字符减去字符“A”就可以了，对应的上面给11中不同水管中的一种。

最后，就是如何深搜了。这个要考虑清楚。用一个flag二维数组表示是否访问过这个方格。深搜的时候，如果访问到它时，先判断它是否被访问过，然后立刻标记为已访问。然后就是判断当前方格的四个方向是不是有接口，如果某个方向有接口的话，就判断这个方向上的下一个方格中，和当前方格相邻的边是不是有接口，如果有接口，则继续深搜这个相邻的点。这里有个处理：（k+2）%4，意味着，比如：当前方格如果右边有接口，则判断右边的方格的左边的边是不是有接口，其它情况一样。然后就是主函数里面的dfs外面的for循环，思想和以前做过的zoj 1709是一样的，就是搜到某个点的时候，用一个mrk标记一下，把和它有关系的点都搜完，如果有符合条件的，mrk会改变，然后就cnt++用来计数，这种题目是相似的，都是要求符合条件的点相邻。

有一个细节，就是矩阵的数组还是从1开始吧，数组稍微开大一点儿，这样在深搜下一个方向的时候，就不会因为数组越界出错了。这个问题以前没有注意到。

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <cstring>//memset()

#include <cstdio>

using namespace std;

int grid[11][4] = {//0~10分别表示A~K方格，0~4表示东南西北

{ 0, 0, 1, 1 },//西北有水管接口

{ 1, 0, 0, 1 },

{ 0, 1, 1, 0 },

{ 1, 1, 0, 0 },

{ 0, 1, 0, 1 },

{ 1, 0, 1, 0 },

{ 1, 0, 1, 1 },

{ 0, 1, 1, 1 },

{ 1, 1, 1, 0 },

{ 1, 1, 0, 1 },

{ 1, 1, 1, 1 } };

int dir[4][2] = {//东南西北方向

{ 0, 1 }, //东：同行，右

{ 1, 0 },//南：下行

{ 0, -1 },

{ -1, 0 } };

int flag[55][55];//是否访问过这个方格

int s[55][55];

bool mrk = false;//全局标记

int n, m;

void dfs(int i, int j){//当前格子s[i][j]

if (i <= 0 || i > m || j <= 0 || j > n) return;

if (flag[i][j]) return;//已经访问过

mrk = true;//表示flag[i][j]有标记过

flag[i][j] = 1;//标记为已访问

//判断当前方格的四个方向是不是有接口，如果某个方向有接口的话，

//就判断这个方向上的下一个方格中，和当前方格相邻的边是不是有接口，

//如果有接口，则继续深搜这个相邻的点

for (int k = 0; k < 4; ++k){//[(k + 2) % 4]表示k的相对方向，东西

if (grid[s[i][j]][k] && //当前格子s[i][j]的k方向

grid[s[i + dir[k][0]][j + dir[k][1]]][(k + 2) % 4]){

dfs(i + dir[k][0], j + dir[k][1]);

}//s[i + dir[k][0]][j + dir[k][1]]表示当前格子s[i][j]的同k方向下个格子

}

return;

}

int main(void){

while (~scanf("%d%d", &m, &n)){

if (m + n < 0) break;

char ch; getchar();//吸收上行后的换行

int cnt = 0;

for (int i = 1; i <= m; ++i){//实际农场1~m行

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

scanf("%c", &ch);

//实际的农场格子，转为数值，在grid中作行下标，表示某格

s[i][j] = ch - 'A';

}

getchar();

}

memset(flag, 0, sizeof(flag));

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

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

mrk = false;

if (!flag[i][j]) dfs(i, j);

if (mrk) cnt++;

}

}

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

}

return 0;

}

dfs(i,j)搜格子，若没访问过，作标记flag[i][j]，同时mrk设为true，表示flag[i][j]有标记过，意味着新的连续块开始。

Main

Mrk=false

Dfs(i,j)

判断Mrk

…

Dfs(i,j)

返回

设置全局Mrk

Dfs(i,j)

全局Mrk

Dfs(i,j)

全局Mrk

Dfs(i,j)

全局Mrk