P1825 [USACO11OPEN]Corn Maze S

题目描述

去年秋天，奶牛们去参观了一个玉米迷宫，迷宫里有一些传送装置，可以将奶牛从一点到另一点进行瞬间转移。这些装置可以双向使用：一头奶牛可以从这个装置的起点立即到此装置的终点，同时也可以从终点出发，到达这个装置的起点。如果一头奶牛处在这个装置的起点或者终点，这头奶牛就必须使用这个装置。

玉米迷宫的外部完全被玉米田包围，除了唯一的一个出口。

这个迷宫可以表示为N×M的矩阵(2 ≤ N ≤ 300；2 ≤ M ≤ 300)，矩阵中的每个元素都由以下项目中的一项组成：

玉米，这些格子是不可以通过的。

. 草地，可以简单的通过。

W 一个装置的结点，可以将一头奶牛传送到相对应的另一个结点。

= 出口

最优方案为：先向右走到装置的结点，花费一个单位时间，再到装置的另一个结点上，花费0个单位时间，然后再向右走一个，再向上走一个，到达出口处，总共花费了3个单位时间。

奶牛仅可以在相邻两个格子之间移动，要在这两个格子不是由玉米组成的前提下才可以移动。奶牛能在一格草地上可能存在的四个相邻的格子移动。从草地移动到相邻的一个格子需要花费一个单位的时间，从装置的一个结点到另一个结点需要花费0个单位时间。

被填充为玉米的格子用“#”表示，草地用“.”表示，每一对装置的结点由相同的大写字母组成“A-Z”，且没有两个不同装置的结点用同一个字母表示，出口用“=”表示。

Bessie在这个迷宫中迷路了，她知道她在矩阵中的位置，将Bessie所在的那一块草地用“@”表示。求出Bessie需要移动到出口处的最短时间。

例如以下矩阵，N=5，M=6：

```
###=##
#.W.##
#.####
#.@W##
######
```

唯一的一个装置的结点用大写字母W表示。

输入格式

第一行：两个用空格隔开的整数N和M；

第2-N+1行：第i+1行描述了迷宫中的第i行的情况（共有M个字符，每个字符中间没有空格。）

输出格式

一个整数，表示Bessie到达终点所需的最短时间。

输入样例

```
5 6
###=##
#.W.##
#.####
#.@W##
######
```

输出样例

```
3
```

解析

求最小步数，典型的广搜题目。此外，题目中有个很特殊的地方，就是存在传送门，因此在这里我们需要对传送门进行额外的处理。所使用的方式就是传值引用。

传值引用

```
#include<bits/stdc++.h>

using namespace std;

int main() {

    //普通的赋值操作
    int a = 3;
    int b = a;
    b = 5;
    cout << "a:" << a << endl;

    //将a的地址赋值给c，通过操作地址直接修改a的值
    int &c = a;
    c = 5;
    cout << "a:" << a << endl;

    return 0;
}
```

编码

```
#include<bits/stdc++.h>

using namespace std;
const int N = 350;

struct point {
    int x;  //横坐标
   int y;  //纵坐标
   int t;  //移动步数
};

//待搜索列表
queue<point> que;

//原地图信息
char maps[N][N];
//访问列表
bool vis[N][N];
//地图边界
int n, m;
//四方向数组
int dx[4] = {1, 0, -1, 0};
int dy[4] = {0, 1, 0, -1};
//起始坐标
int sx, sy;

//使用取址符，处理传送行为
void goto_another(int &nx, int &ny) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            //字母相同且不是同一个点，就是传送门的另一端
          if (maps[i][j] == maps[nx][ny] && (i != nx || j != ny)) {
                //修改坐标，实现传送
             nx = i;
                ny = j;
                return;
            }
        }
    }
}

void Bfs() {
    while (!que.empty()) {
        point f = que.front();
        que.pop();
        if (maps[f.x][f.y] == '=') {
            cout << f.t;
```

```
            return;
        }
        //特判部分，如果当前点是一个传送门，那么就传送至另一个传送门
        if (maps[f.x][f.y] >= 'A' && maps[f.x][f.y] <= 'Z') {
            goto_another(f.x, f.y);
        }
        for (int i = 0; i <= 3; i++) {
            int nx = f.x + dx[i];
            int ny = f.y + dy[i];
            //有效性检测
          if (nx >= 1 && nx <= n && ny >= 1 && ny <= m
            && maps[nx][ny] != '#' && !vis[nx][ny]) {
                vis[nx][ny] = true;
                que.push((point) {nx, ny, f.t + 1});
            }
        }
    }
}

int main() {
    cin >> n >> m;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            cin >> maps[i][j];
            //获取起点坐标
          if (maps[i][j] == '@') {
                sx = i;
                sy = j;
            }
        }
    }
    //将起点放入待搜索列表
   que.push((point) {sx, sy, 0});
    //执行广搜
   Bfs();
    return 0;
}
```