1252：走迷宫

题目描述

一个迷宫由R行C列格子组成，有的格子里有障碍物，不能走；有的格子是空地，可以走。

给定一个迷宫，求从左上角走到右下角最少需要走多少步(数据保证一定能走到)。只能在水平方向或垂直方向走，不能斜着走。

输入格式

第一行是两个整数，R和C，代表迷宫的长和宽。（ 1<= R，C <= 40)

接下来是R行，每行C个字符，代表整个迷宫。

空地格子用‘.’表示，有障碍物的格子用‘#’表示。

迷宫左上角和右下角都是‘.’。

输出格式

输出从左上角走到右下角至少要经过多少步（即至少要经过多少个空地格子）。计算步数要包括起点和终点。

输入样例

```
5 5
..###
#....
#.#.#
#.#.#
#.#..
```

输出样例

```
9
```

解析

最短路径，广搜模板。

编码

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

using namespace std;
```

```
int const MaxNum = 41;

//节点结构体
struct Node {
    int x;   //坐标
   int y;
    int step;  //移动步数
   //当前节点的基本信息
   Node(int nx, int ny, int stepNum) {
        x = nx;
        y = ny;
        step = stepNum;
    }

    Node() = default;
};

int R, C;  //R行 C列
char Maps[MaxNum][MaxNum]; //地图信息;
bool Vis[MaxNum][MaxNum];  //判断是否走过
int gapX[4] = {0, 0, -1, 1};  //上下左右偏移的x
int gapY[4] = {-1, 1, 0, 0};  //上下左右偏移的y


//检测是否可以通过
bool CanPass(int newX, int newY) {
    //保证没有越界
   if (newX >= 0 && newY >= 0 && newX < R && newY < C) {
        //没有被访问过
       if (!Vis[newX][newY]) {
            //可以行走
           if (Maps[newX][newY] == '.') {
                return true;
            }
        }
    }
    return false;
}

Node NodeQueue[1000];


//广度搜索
void Bfs() {
    Node start = Node(0, 0, 1);
    Node end = Node(R - 1, C - 1, 1);
    //初始化
   memset(Vis, false, sizeof(Vis));
```

```
    memset(NodeQueue, 0, sizeof(NodeQueue));
    //设置头尾
    int head = 0;
    int tail = 0;
    //将首点加入队列
    NodeQueue[head] = start;
    //设置该节点已经被访问
    Vis[start.x][start.y] = true;
    //标记当前节点已经走过
    while (head <= tail) {
        //取出一个节点
        Node room = NodeQueue[head];
        head++;
        //向四个方向移动;
        for (int k = 0; k < 4; ++k) {
            int newX = room.x + gapX[k];
            int newY = room.y + gapY[k];
            //找到目标;
            if (newX == end.x && newY == end.y) {
                //输出一共走的步数记录最后一个点
                cout << room.step + 1 << endl;
                return;
            }
            //下个节点可以被访问
            if (CanPass(newX, newY)) {
                //设置该节点已经被访问
                Vis[newX][newY] = true;
                tail++;
                //将数据加入新的队列
                NodeQueue[tail] = Node(newX, newY, room.step + 1);
            }
        }
    }
    //没有找到路线
    cout << -1 << endl;
}

//读取输入的数据
void ReadInfo() {
    cin >> R >> C;//R行 C列
    //读入字符串信息;
    for (int i = 0; i < R; ++i) {
        cin >> Maps[i];
    }
}


int main() {
    ReadInfo();
```

```
    Bfs();
    return 0;
}
```