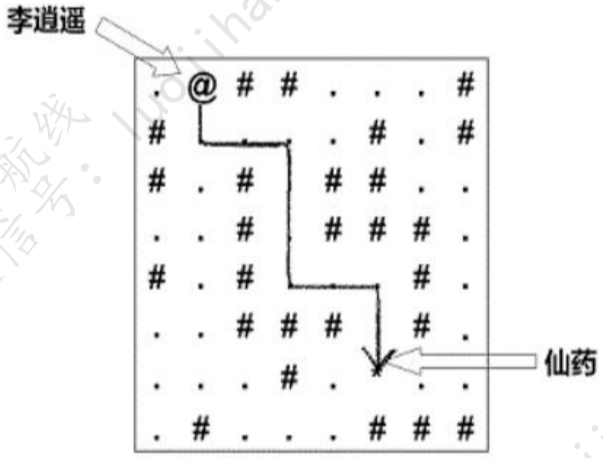
1251：仙岛求药

题目描述

少年李逍遥的婶婶病了，王小虎介绍他去一趟仙灵岛，向仙女姐姐要仙丹救婶婶。叛逆但孝顺的李逍遥闯进了仙灵岛，克服了千险万难来到岛的中心，发现仙药摆在了迷阵的深处。迷阵由M×N个方格组成，有的方格内有可以瞬秒李逍遥的怪物，而有的方格内则是安全。现在李逍遥想尽快找到仙药，显然他应避开有怪物的方格，并经过最少的方格，而且那里会有神秘人物等待着他。现在要求你来帮助他实现这个目标。

下图 显示了一个迷阵的样例及李逍遥找到仙药的路线。



输入格式

输入有多组测试数据. 每组测试数据以两个非零整数 M 和 N 开始，两者均不大于20。M 表示迷阵行数，N 表示迷阵列数。接下来有 M 行，每行包含N个字符,不同字符分别代表不同含义：

1）‘@’：少年李逍遥所在的位置；

2）‘.’：可以安全通行的方格；

3）‘#’：有怪物的方格；

4）‘*’：仙药所在位置。

当在一行中读入的是两个零时，表示输入结束。

输出格式

对于每组测试数据，分别输出一行，该行包含李逍遥找到仙药需要穿过的最少的方格数目(计数包括初始位置的方块)。如果他不可能找到仙药，则输出 -1。

输入样例

```
8 8
.@##...#
#....#.#
#.#.##..
..#.###.
#.#...#.
..###.#.
...#.*..
.#...###
6 5
.*.#.
.#...
..##.
.....
.#...
....@
9 6

.#..#.
.#.*.#
.####.
..#...
..#...
..#...
..#...
#.@.##
.#..#.
0 0
```

输出样例

```
10
8
-1
```

解析

求最短路径，直接上广搜大法。

编码

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

using namespace std;

int const MaxNum = 21;

//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 m, n;  //纵向宽度和横向长度
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 < m && newY < n) {
        //没有被访问过
        if (!Vis[newX][newY]) {
            //可以行走
            if (Maps[newX][newY] == '.') {
                return true;
            }
        }
    }
    return false;
}

queue<Node> NodeQueue; //待查询队列
//广度搜索
void Bfs(Node start, Node end) {
    while (!NodeQueue.empty()) {
        NodeQueue.pop();
    }

    memset(Vis, false, sizeof(Vis));
    //将首点加入队列
    NodeQueue.push(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 (!NodeQueue.empty()) {
        //取出一个节点
        Node room = NodeQueue.front();
        NodeQueue.pop();
        //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，不需要记录最后一个点
                cout << room.step << endl;
                return;
            }
            //将数据加入新的队列
            if (CanPass(newX, newY)) {
                //设置该节点已经被访问
                Vis[newX][newY] = true;
                NodeQueue.push(Node(newX, newY, room.step + 1));
            }
        }
    }
    //没有找到路线
    cout << -1 << endl;
}

//读取输入的数据
void ReadInfo() {
    Node start = Node(-1, -1, 1);
    Node end = Node(-1, -1, 1);
    //读入字符串信息;
    for (int i = 0; i < m; ++i) {
        cin >> Maps[i];
    }
    //遍历纵向
    for (int i = 0; i < m; ++i) {
        //遍历横向
        for (int j = 0; j < n; ++j) {
            if (Maps[i][j] == '@') {
                start = Node(i, j, 1);
            }
            if (Maps[i][j] == '*') {
                end = Node(i, j, 1);
            }
        }
    }
    Bfs(start, end);
}
```

```
int main() {
    while (cin >> m >> n) {
        //读取到终止信息
       if (m + n == 0)
            return 0;
        ReadInfo();
    }
    return 0;
}
```