1256：献给阿尔吉侬的花束

题目描述

阿尔吉侬是一只聪明又慵懒的小白鼠，它最擅长的就是走各种各样的迷宫。今天它要挑战一个非常大的迷宫，研究员们为了鼓励阿尔吉侬尽快到达终点，就在终点放了一块阿尔吉侬最喜欢的奶酪。现在研究员们想知道，如果阿尔吉侬足够聪明，它最少需要多少时间就能吃到奶酪。

迷宫用一个R×C的字符矩阵来表示。字符S表示阿尔吉侬所在的位置，字符E表示奶酪所在的位置，字符#表示墙壁，字符.表示可以通行。阿尔吉侬在1个单位时间内可以从当前的位置走到它上下左右四个方向上的任意一个位置，但不能走出地图边界。

输入格式

第一行是一个正整数T（1 ≤ T ≤ 10），表示一共有T组数据。

每一组数据的第一行包含了两个用空格分开的正整数R和C（2 ≤ R, C ≤ 200），表示地图是一个R×C的矩阵。

接下来的R行描述了地图的具体内容，每一行包含了C个字符。字符含义如题目描述中所述。保证有且仅有一个S和E。

输出格式

```
3
3 4
.S..
###.
..E.
3 4
.S..
.E..
....
3 4
.S..
####
..E.
```

输入样例

```
5
1
oop!
```

输出样例

2

解析

通过描述可知，本题求的是最短路径，直接套用广搜模板。

编码

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

using namespace std;

int const MaxNum = 201;

//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;  //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
Node NodeQueue[MaxNum * MaxNum];


//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;
}


//广度搜索
void Bfs(Node start, Node end) {
    //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，不需要记录最后一个点
             cout << room.step << 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);
            }
        }
    }
    cout << "oop!" << 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 < R; ++i) {
        cin >> Maps[i];
    }
    //遍历纵向
    for (int i = 0; i < R; ++i) {
        //遍历横向
        for (int j = 0; j < C; ++j) {
            if (Maps[i][j] == 'S') {
                start = Node(i, j, 1);
            }
            if (Maps[i][j] == 'E') {
                end = Node(i, j, 1);
            }
        }
    }
    Bfs(start, end);
}

int main() {
    int n;
    cin >> n;
    for (int i = 0; i < n; ++i) {
        cin >> R >> C;
        ReadInfo();
    }

    return 0;
}
```