POJ2312：

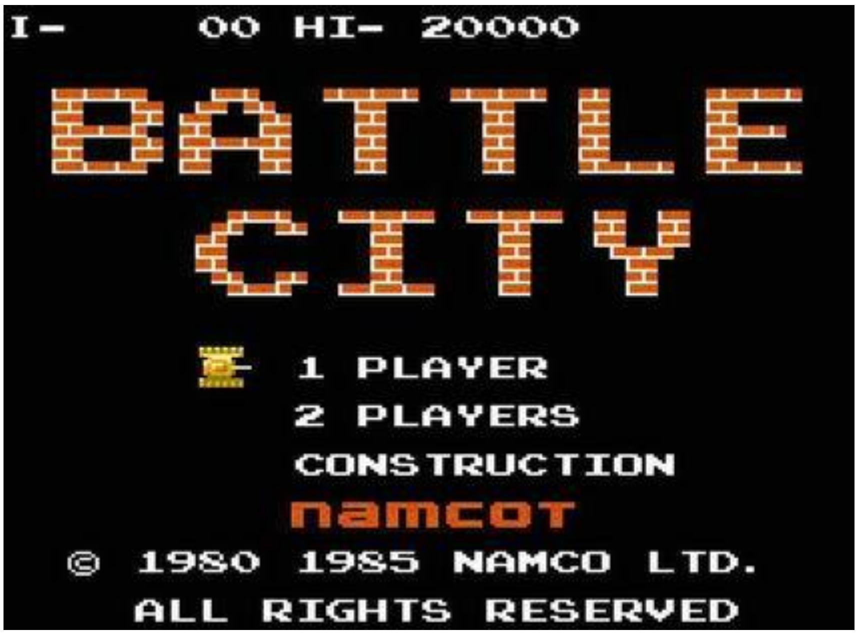
Battle City

Time Limit: 1000MS **Memory Limit:** 65536K

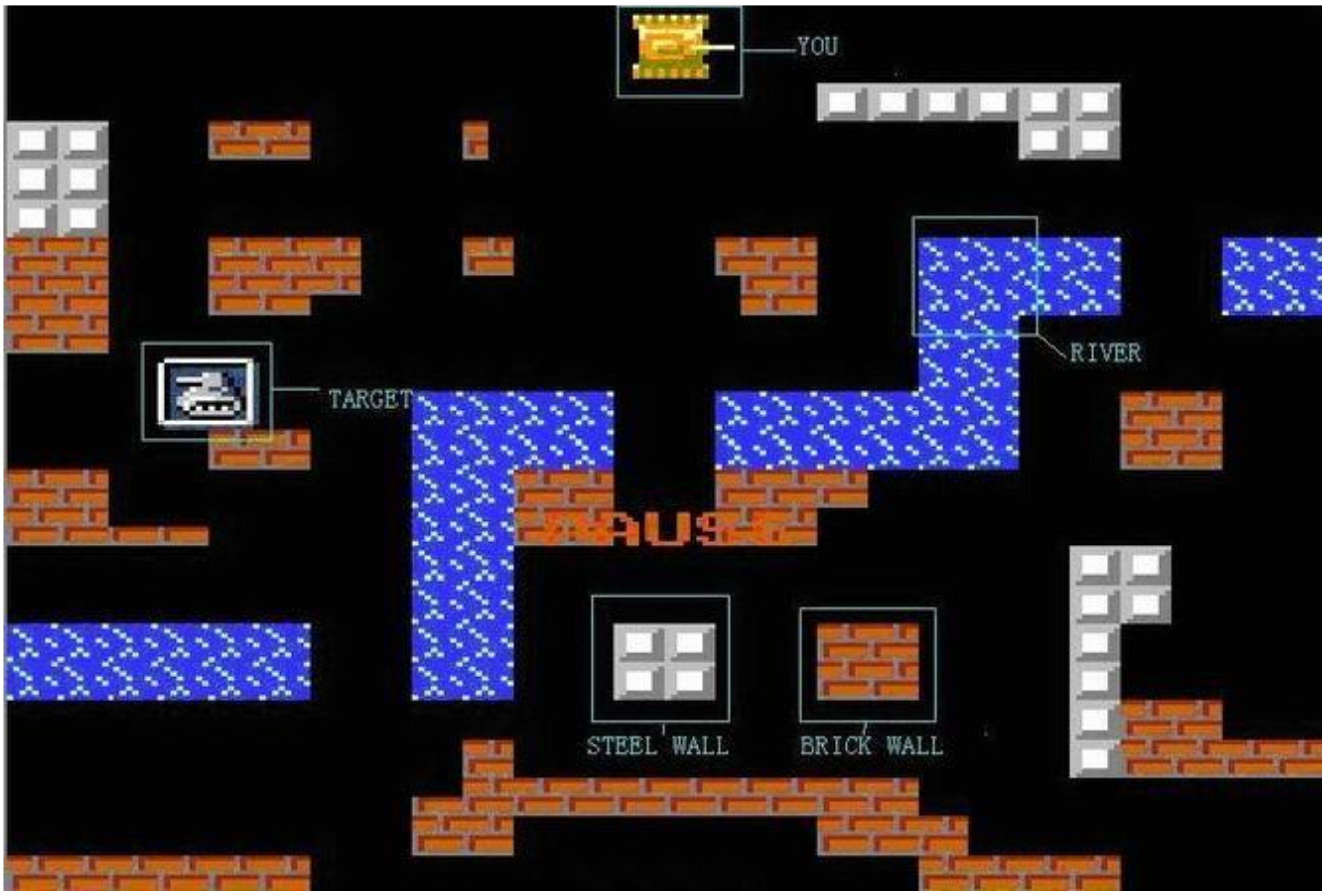
Total Submissions: 7270 **Accepted:** 2454

Description

Many of us had played the game "Battle city" in our childhood, and some people (like me) even often play it on computer now.



What we are discussing is a simple edition of this game. Given a map that consists of empty spaces, rivers, steel walls and brick walls only. Your task is to get a bonus as soon as possible suppose that no enemies will disturb you (See the following picture).



Your tank can't move through rivers or walls, but it can destroy brick walls by shooting. A brick wall will be turned into empty spaces when you hit it, however, if your shot hit a steel wall, there will be no damage to the wall. In each of your turns, you can choose to move to a neighboring (4 directions, not 8) empty space, or shoot in one of the four directions without a move. The shot will go ahead in that direction, until it go out of the map or hit a wall. If the shot hits a brick wall, the wall will disappear (i.e., in this turn). Well, given the description of a map, the positions of your tank and the target, how many turns will you take at least to arrive there?

Input

The input consists of several test cases. The first line of each test case contains two integers M and N (2 <= M, N <= 300). Each of the following M lines contains N uppercase letters, each of which is one of 'Y' (you), 'T' (target), 'S' (steel wall), 'B' (brick wall), 'R' (river) and 'E' (empty space). Both 'Y' and 'T' appear only once. A test case of M = N = 0 indicates the end of input, and should not be processed.

Output

For each test case, please output the turns you take at least in a separate line. If you can't arrive at the target, output "-1" instead.

Sample Input

```
3 4
YBEB
EERE
SSTE
0 0
```

Sample Output

```
8
```

大意：

给你一个 m 行 n 列的矩阵。

Y 代表起点，T 代表终点。B、E 可以走，S、R 不可以走，B 的时间花费为 2，E 为 1.

求 Y 到 T 的最短时间。

思路：

简单的广搜

Code:

```
#include<iostream>
#include<string>
#include<cstring>
#include<cstdio>
#include<algorithm>
#include<queue>
#include<cmath>
using namespace std;

int MaxRow, MaxCol;
char map[305][305];
bool visit[305][305];
```

```
int dir[4][2] = {{0, 1}, {0, -1}, {1, 0}, {-1, 0}}; //上， 下， 右， 左

typedef struct tank
{
	int x, y, cost;
};

bool operator < (const tank &a, const tank &b) //优先队列， 队头为最优解。即花费最少
{
	return a.cost > b.cost;
}

bool is_error(int x, int y) //是否为边界或障碍
{
	if(x < 0 || y < 0 || x >= MaxRow || y >= MaxCol)
		return 1;
	if(map[x][y] == 'R' || map[x][y] == 'S' || visit[x][y])
		return 1;
	return 0;
}

bool BFS(int a, int b) //广搜， 找到即为最优解
{
	int x, y;
	priority_queue<tank> que; //优先队列
	tank in, out;
	in.x = a;
	in.y = b;
	in.cost = 0;
	que.push(in); //起点
	while(!que.empty())
	{
		out = que.top(); //出队寻找四个方向， 满足条件则入队
		que.pop();
		if(map[out.x][out.y] == 'T') //结束
		{
			printf("%d\n", out.cost);
			return true;
		}
		for(int i = 0; i < 4; ++i) //四个方向
		{
			x = out.x + dir[i][0]; //x，y 用于记录 out.x 的四个方向，不能用 out.x += dir[i][0]。
则每次 out.x 值都改变！ ~
			y = out.y + dir[i][1];
```

```
                        if(is_error(x, y)) //是否为边界或障碍
                                continue;
                        visit[x][y] = 1; //搜过则标记
                                in.x = x;
                                in.y = y;
                                in.cost = out.cost + 1;
                        if(map[x][y] == 'B')
                                in.cost++;
                        que.push(in); //满足条件入队
                }
        }
        return false;
}

int main()
{
        int StartRow, StartCol;
        while(scanf("%d%d", &MaxRow, &MaxCol) && MaxRow && MaxCol)
        {
                memset(visit, 0, sizeof(visit));
                getchar(); //吃掉 scanf 留下的\n
                for(int i = 0; i < MaxRow; ++i)
                        gets(map[i]);
                for(int i = 0; i < MaxRow; ++i)
                        for(int j = 0; j < MaxCol; ++j)
                                if(map[i][j] == 'Y') //起点
                                {
                                        StartRow = i;
                                        StartCol = j;
                                }
                visit[StartRow][StartCol] = 1; //起点标记
                if(!BFS(StartRow, StartCol))
                        printf("-1\n");
        }
        return 0;
}
```