AtCoder Beginner Contest 088

Problem D – Grid Repainting

Prof. Edson Alves

Faculdade UnB Gama

We have an $H \times W$ grid whose squares are painted black or white. The square at the i-th row from the top and the j-th column from the left is denoted as (i, j).

Snuke would like to play the following game on this grid. At the beginning of the game, there is a character called Kenus at square $(1, 1)$. The player repeatedly moves Kenus up, down, left or right by one square. The game is completed when Kenus reaches square (H, W) passing only white squares.

Before Snuke starts the game, he can change the color of some of the white squares to black. However, he cannot change the color of square $(1, 1)$ and (H, W). Also, changes of color must all be carried out before the beginning of the game.

Nos temos uma malha $H \times W$ cujos quadrados são pintados de preto ou branco. O quadrado na i-ésima linha a partir do topo e na j-ésima coluna a contar da esquerda é representada pelo par (i, j).

Snuke gostaria de brincar do seguinte jogo nesta malha. No início do jogo, há um personagem chamada Kenus no quadrado $(1, 1)$. O jogador move Kenus um quadrado para cima, baixo, esquerda ou direita. O jogo termina quando Kenus atinge o quadrado (H, W) passando apenas por quadrados brancos.

Antes que Snuke inicie o jogo, ele pode mudar a cor de alguns quadrados brancos para pretos. Contudo, ele não pode mudar a cor dos quadrados $(1, 1)$ e (H, W). As mudanças de cores deve ser todas feitas antes do início do jogo.

When the game is completed, Snuke's score will be the number of times he changed the color of a square before the beginning of the game. Find the maximum possible score that Snuke can achieve, or print -1 if the game cannot be completed, that is, Kenus can never reach square (H, W) regardless of how Snuke changes the color of the squares.

The color of the squares are given to you as characters $s_{i,j}$. If square (i, j) is initially painted by white, $s_{i,j}$ is .; if square (i, j) is initially painted by black, $s_{i,j}$ is #.

Constraints

- H *is an integer between* 2 *and* 50 *(inclusive).*
- W *is an integer between* 2 *and* 50 *(inclusive).*
- $s_{i,j}$ *is* . *or* # $(1 \leq i \leq H, 1 \leq j \leq W)$.
- $s_{1,1}$ *and* $s_{H,W}$ *are* ..

Quando o jogo termina, a pontuação de Snuke será o número de trocas de cores antes do início do jogo. Determine a maior pontuação possível que Snuke pode alcançar, ou imprima -1 se o jogo não pode ser finalizado, isto é, Kenus não pode atingir o quadrado (H, W) independentemente de como Snuke altere as cores dos quadrados.

As cores dos quadrados são dadas na forma de caracteres $s_{i,j}$. Se o quadrado (i, j) é pintado branco inicialmente, $s_{i,j}$ será `.`; se (i, j) é pintado preto, $s_{i,j}$ será #.

Restrições

- H é um inteiro entre 2 e 50 (inclusive).
- W é um inteiro entre 2 e 50 (inclusive).
- $s_{i,j}$ é igual a `.` ou # $(1 \leq i \leq H, 1 \leq j \leq W)$.
- $s_{1,1}$ e $s_{H,W}$ são iguais a `.`.

Input

Input is given from Standard Input in the following format:

$$
\begin{array}{l}
H\ W \\
s_{1,1}s_{1,2}s_{1,3}\cdots s_{1,W} \\
s_{2,1}s_{2,2}s_{2,3}\cdots s_{2,W} \\
\ \vdots \qquad\quad \vdots \\
s_{H,1}s_{H,2}s_{H,3}\cdots s_{H,W}
\end{array}
$$

Output

Print the maximum possible score that Snuke can achieve, or print -1 if the game cannot be completed.

Entrada

A entrada é dada na Entrada Padrão no seguinte formato:

$$
\begin{array}{l}
H\ W \\
s_{1,1} s_{1,2} s_{1,3} \cdots s_{1,W} \\
s_{2,1} s_{2,2} s_{2,3} \cdots s_{2,W} \\
\quad \vdots \qquad\qquad \vdots \\
s_{H,1} s_{H,2} s_{H,3} \cdots s_{H,W}
\end{array}
$$

Output

Imprima a maior pontuação possível que Snuke pode conseguir, ou imprima -1 se o jogo não pode ser concluído.

Exemplo de entrada e saída

Exemplo de entrada e saída

3 3

Exemplo de entrada e saída

3 3

H

Exemplo de entrada e saída

$$\begin{tikzcd} 3 & 3 \arrow[r] & W \\ H \arrow[u] & & \end{tikzcd}$$

Exemplo de entrada e saída

3 3

Exemplo de entrada e saída

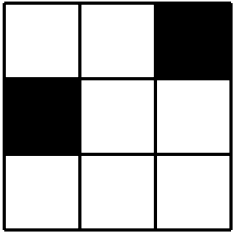
```
3 3
..#
#..
...
```

Exemplo de entrada e saída

```
3 3
..#
#..  ⟶ malha
...
```

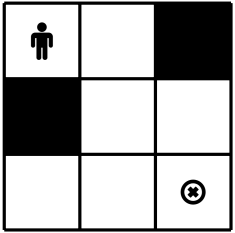
Exemplo de entrada e saída

```
3 3
..#
#..
...
```



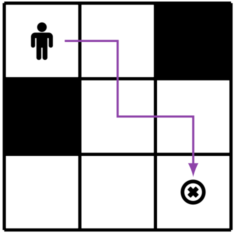
Exemplo de entrada e saída

```
3 3
..#
#..
...
```



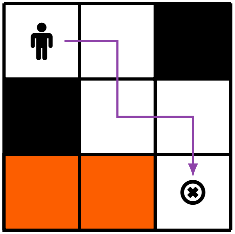
Exemplo de entrada e saída

```
3 3
..#
#..
...
```



Exemplo de entrada e saída

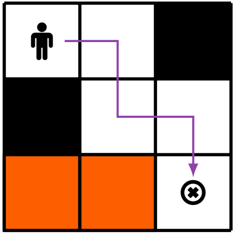
```
3 3
..#
#..
...
```



Exemplo de entrada e saída

```
3 3
..#
#..
...
```

2



Exemplo de entrada e saída

Exemplo de entrada e saída

```
10 37
```

Exemplo de entrada e saída

```
10 37
.....................................
...#...####...####..###...###...###..
..#.#..#...#.##....#...#.#...#.#...#.
..#.#..#...#.#.....#...#.#...#.#...#.
.#...#.#..##.#.....#...#.#.###.#.###.
.#####.####..#.....#...#..##....##...
.#...#.#...#.#.....#...#.#...#.#...#.
.#...#.#...#.##....#...#.#...#.#...#.
.#...#.####...####..###...###...###..
.....................................
```

Exemplo de entrada e saída

```
10 37
.....................................
####################################.
####################################.
####################################.
####################################.
####################################.
####################################.
####################################.
####################################.
####################################.
```

Exemplo de entrada e saída

```
10 37
.....................................
####################################.
####################################.
####################################.
####################################.
####################################.
####################################.
####################################.
####################################.
####################################.
```

→ **209**

Solução

Solução

- $\star$ **Para que exista solução, é preciso que exista ao menos um caminho de $(1,1)$ a (H,W)**

Solução

- $\star$ **Para que exista solução, é preciso que exista ao menos um caminho de** $(1, 1)$ **a** (H, W)

- $\star$ **Uma vez identificado um caminho, todos os quadrados brancos que não fizerem parte do caminho podem ser pintados de preto**

Solução

$\star$ **Para que exista solução, é preciso que exista ao menos um caminho de $(1, 1)$ a (H, W)**

$\star$ **Uma vez identificado um caminho, todos os quadrados brancos que não fizerem parte do caminho podem ser pintados de preto**

$\star$ **Deste modo, para maximizar o número de quadrados a serem pintados é preciso escolher um caminho mínimo**

Solução

- ⋆ **Para que exista solução, é preciso que exista ao menos um caminho de $(1,1)$ a (H,W)**

- ⋆ **Uma vez identificado um caminho, todos os quadrados brancos que não fizerem parte do caminho podem ser pintados de preto**

- ⋆ **Deste modo, para maximizar o número de quadrados a serem pintados é preciso escolher um caminho mínimo**

- ⋆ **Como o custo para ser mover entre quadrados vizinhos é constante, este caminho pode ser obtido por meio de uma BFS**

```
int solve(const vector<string>& S, int H, int W)
{
    memset(dist, -1, sizeof dist);
    int whites = 0;

    for (int i = 0; i < H; ++i)
        for (int j = 0; j < W; ++j)
            whites += (S[i][j] == '.' ? 1 : 0);

    queue<ii> q;
    q.push(ii(0, 0));

    dist[0][0] = 1;

    while (not q.empty())
    {
        const vector<ii> dirs { ii(1, 0), ii(0, 1), ii(-1, 0), ii(0, -1) };
        auto [x, y] = q.front();
        q.pop();
```

```
        if (x == H - 1 and y == W - 1)
            break;

        for (auto [dx, dy] : dirs)
        {
            int u = x + dx, v = y + dy;

            if (u < 0 or u >= H or v < 0 or v >= W
                or dist[u][v] > -1 or S[u][v] == '#')
                    continue;

            dist[u][v] = dist[x][y] + 1;
            q.push(ii(u, v));
        }
    }

    return dist[H - 1][W - 1] == -1 ? -1 : whites - dist[H - 1][W - 1];
}
```