

Problem 2814: Minimum Time Takes to Reach Destination Without Drowning

Problem Information

Difficulty: Hard

Acceptance Rate: 53.80%

Paid Only: Yes

Tags: Array, Breadth-First Search, Matrix

Problem Description

You are given an $n \times m$ 0-indexed grid of string 'land'. Right now, you are standing at the cell that contains "S", and you want to get to the cell containing "D". There are three other types of cells in this land:

* ".": These cells are empty. * "X": These cells are stone. * "*": These cells are flooded.

At each second, you can move to a cell that shares a side with your current cell (if it exists). Also, at each second, every **empty cell** that shares a side with a flooded cell becomes flooded as well. There are two problems ahead of your journey:

* You can't step on stone cells. * You can't step on flooded cells since you will drown (also, you can't step on a cell that will be flooded at the same time as you step on it).

Return the minimum time it takes you to reach the destination in seconds, or -1 if it is impossible.

Note that the destination will never be flooded.

Example 1:

Input: land = `[["D",".","*"],[".",".","."],[".","S","."]]` **Output:** 3 **Explanation:** The picture below shows the simulation of the land second by second. The blue cells are flooded, and the gray cells are stone. Picture (0) shows the initial state and picture (3) shows the final state when we reach destination. As you see, it takes us 3 second to reach destination and the answer would be 3. It can be shown that 3 is the minimum time needed to reach from S to D.

Example 2:

Input: land = `[["D","X",""],[".", ".", "."],[".", ".", "S"]]` **Output:** -1 **Explanation:** The picture below shows the simulation of the land second by second. The blue cells are flooded, and the gray cells are stone. Picture (0) shows the initial state. As you see, no matter which paths we choose, we will drown at the 3rd second. Also the minimum path takes us 4 seconds to reach from S to D. So the answer would be -1.

Example 3:

Input: land = `[["D", ".", ".", ".", ".", ".", "."],[".", "X", ".", "X", ".", ".", "."],[".", ".", ".", ".", "S", "."]]` **Output:** 6 **Explanation:** It can be shown that we can reach destination in 6 seconds. Also it can be shown that 6 is the minimum seconds one need to reach from S to D.

Constraints:

* $2 \leq n, m \leq 100$ * `land` consists only of `"S"`, `"D"`, `"."`, `"X"` and `"#"`. * **Exactly** one of the cells is equal to `"S"`. * **Exactly** one of the cells is equal to `"D"`.

Code Snippets

C++:

```
class Solution {
public:
    int minimumSeconds(vector<vector<string>>& land) {

    }
};
```

Java:

```
class Solution {
    public int minimumSeconds(List<List<String>> land) {
```

```
}  
}
```

Python3:

```
class Solution:  
    def minimumSeconds(self, land: List[List[str]]) -> int:
```