Problem 1778: Shortest Path in a Hidden Grid

Problem Information

Difficulty: Medium
Acceptance Rate: 44.26%
Paid Only: Yes
Tags: Array, Depth-First Search, Breadth-First Search, Matrix, Interactive

Problem Description

This is an **interactive problem**.

There is a robot in a hidden grid, and you are trying to get it from its starting cell to the target cell in this grid. The grid is of size `m x n`, and each cell in the grid is either empty or blocked. It is **guaranteed** that the starting cell and the target cell are different, and neither of them is blocked.

You want to find the minimum distance to the target cell. However, you **do not know** the grid's dimensions, the starting cell, nor the target cell. You are only allowed to ask queries to the `GridMaster` object.

Thr `GridMaster` class has the following functions:

* `boolean canMove(char direction)` Returns `true` if the robot can move in that direction. Otherwise, it returns `false`. * `void move(char direction)` Moves the robot in that direction. If this move would move the robot to a blocked cell or off the grid, the move will be **ignored** , and the robot will remain in the same position. * `boolean isTarget()` Returns `true` if the robot is currently on the target cell. Otherwise, it returns `false`.

Note that `direction` in the above functions should be a character from `{'U','D','L','R'}`, representing the directions up, down, left, and right, respectively.

Return _the**minimum distance** between the robot's initial starting cell and the target cell. If there is no valid path between the cells, return _`-1`.

Custom testing:

The test input is read as a 2D matrix `grid` of size `m x n` where:

* `grid[i][j] == -1` indicates that the robot is in cell `(i, j)` (the starting cell). * `grid[i][j] == 0` indicates that the cell `(i, j)` is blocked. * `grid[i][j] == 1` indicates that the cell `(i, j)` is empty. * `grid[i][j] == 2` indicates that the cell `(i, j)` is the target cell.

There is exactly one `-1` and `2` in `grid`. Remember that you will **not** have this information in your code.

Example 1:

Input: grid = [[1,2],[-1,0]] **Output:** 2 **Explanation:** One possible interaction is described below: The robot is initially standing on cell (1, 0), denoted by the -1. - master.canMove('U') returns true. - master.canMove('D') returns false. - master.canMove('L') returns false. - master.canMove('R') returns false. - master.move('U') moves the robot to the cell (0, 0). - master.isTarget() returns false. - master.canMove('U') returns false. - master.canMove('D') returns true. - master.canMove('L') returns false. - master.canMove('R') returns true. - master.move('R') moves the robot to the cell (0, 1). - master.isTarget() returns true. We now know that the target is the cell (0, 1), and the shortest path to the target cell is 2.

Example 2:

Input: grid = [[0,0,-1],[1,1,1],[2,0,0]] **Output:** 4 **Explanation:** The minimum distance between the robot and the target cell is 4.

Example 3:

Input: grid = [[-1,0],[0,2]] **Output:** -1 **Explanation:** There is no path from the robot to the target cell.

Constraints:

* `1 <= n, m <= 500` * `m == grid.length` * `n == grid[i].length` * `grid[i][j]` is either `-1`, `0`, `1`, or `2`. * There is **exactly one** `-1` in `grid`. * There is **exactly one** `2` in `grid`.

Code Snippets

C++:

```
/**
* // This is the GridMaster's API interface.
* // You should not implement it, or speculate about its implementation
* class GridMaster {
* public:
* bool canMove(char direction);
* void move(char direction);
* boolean isTarget();
* };
*/

class Solution {
public:
int findShortestPath(GridMaster &master) {

}
};
```

Java:

```
/**
* // This is the GridMaster's API interface.
* // You should not implement it, or speculate about its implementation
* class GridMaster {
* boolean canMove(char direction);
* void move(char direction);
* boolean isTarget();
* }
*/

class Solution {
public int findShortestPath(GridMaster master) {

}
}
```

Python3:

```
# """
# This is GridMaster's API interface.
# You should not implement it, or speculate about its implementation
# """
#class GridMaster(object):
```

```
# def canMove(self, direction: str) -> bool:
#
#
# def move(self, direction: str) -> None:
#
#
# def isTarget(self) -> bool:
#
#

class Solution(object):
def findShortestPath(self, master: 'GridMaster') -> int:
```