505. The Maze II

Medium 938 43 Add to List Share

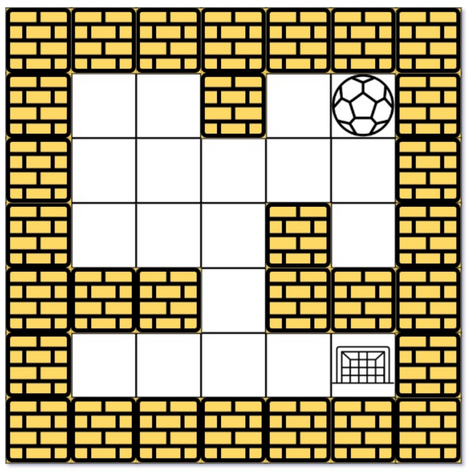
There is a ball in a `maze` with empty spaces (represented as `0`) and walls (represented as `1`). The ball can go through the empty spaces by rolling **up, down, left or right**, but it won't stop rolling until hitting a wall. When the ball stops, it could choose the next direction.

Given the `m x n maze`, the ball's `start` position and the `destination`, where `start = [start_row, start_col]` and `destination = [destination_row, destination_col]`, return *the shortest* ***distance*** *for the ball to stop at the destination*. If the ball cannot stop at `destination`, return `-1`.

The **distance** is the number of **empty spaces** traveled by the ball from the start position (excluded) to the destination (included).

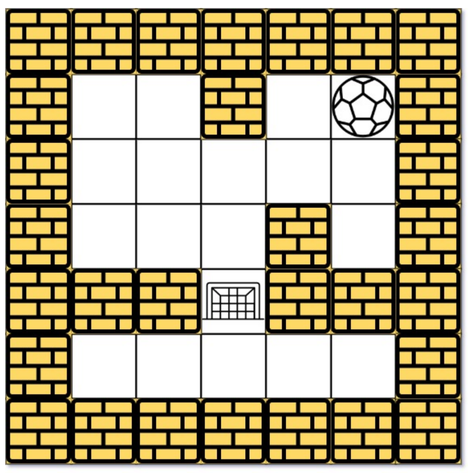
You may assume that **the borders of the maze are all walls** (see examples).

Example 1:



```
Input: maze = [[0,0,1,0,0],[0,0,0,0,0],[0,0,0,1,0],[1,1,0,1,1],[0,0,0,0,0]], start = [0,4], destination = [4,4]
Output: 12
Explanation: One possible way is : left -> down -> left -> down -> right -> down -> right.
The length of the path is 1 + 1 + 3 + 1 + 2 + 2 + 2 = 12.
```

Example 2:



```
Input: maze = [[0,0,1,0,0],[0,0,0,0,0],[0,0,0,1,0],[1,1,0,1,1],[0,0,0,0,0]], start = [0,4], destination = [3,2]
Output: -1
Explanation: There is no way for the ball to stop at the destination. Notice that you can pass through the destination but you cannot stop there.
```

Example 3:

```
Input: maze = [[0,0,0,0,0],[1,1,0,0,1],[0,0,0,0,0],[0,1,0,0,1],[0,1,0,0,0]], start = [4,3], destination = [0,1]
Output: -1
```

Constraints:

- `m == maze.length`
- `n == maze[i].length`
- `1 <= m, n <= 100`
- `maze[i][j]` is `0` or `1`.
- `start.length == 2`
- `destination.length == 2`
- `0 <= start_row, destination_row <= m`
- `0 <= start_col, destination_col <= n`
- Both the ball and the destination exist in an empty space, and they will not be in the same position initially.
- The maze contains **at least 2 empty spaces**.

Accepted 72,592 | Submissions 143,871

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Amazon 2 Google 2

Related Topics

Depth-First Search Breadth-First Search Graph Heap (Priority Queue) Shortest Path

Similar Questions

The Maze	Medium
The Maze III	Hard

Java Autocomplete

```java
public class Solution {
    public int shortestDistance(int[][] maze, int[] start, int[] dest) {
        int[][] distance = new int[maze.length][maze[0].length];
        for (int[] row: distance)
            Arrays.fill(row, Integer.MAX_VALUE);
        distance[start[0]][start[1]] = 0;
        dfs(maze, start, distance);
        return distance[dest[0]][dest[1]] == Integer.MAX_VALUE ? -1 : distance[dest[0]][dest[1]];
    }

    public void dfs(int[][] maze, int[] start, int[][] distance) {
        int[][] dirs={{0,1}, {0,-1}, {-1,0}, {1,0}};
        for (int[] dir: dirs) {
            int x = start[0] + dir[0];
            int y = start[1] + dir[1];
            int count = 0;
            while (x >= 0 && y >= 0 && x < maze.length && y < maze[0].length && maze[x][y] == 0) {
                x += dir[0];
                y += dir[1];
                count++;
            }
            if (distance[start[0]][start[1]] + count < distance[x - dir[0]][y - dir[1]]) {
                distance[x - dir[0]][y - dir[1]] = distance[start[0]][start[1]] + count;
                dfs(maze, new int[]{x - dir[0],y - dir[1]}, distance);
            }
        }
    }
}
```