Documentation for Unique Paths II Solution

Problem Description

Given an `m x n` grid where each cell can either be empty (marked as `0`) or contain an obstacle (marked as `1`), there is a robot initially positioned at the top-left corner of the grid. The robot aims to reach the bottom-right corner of the grid by moving only right or down at any point in time. The task is to find the number of unique paths the robot can take to reach the bottom-right corner, avoiding obstacles.

Example

Input

obstacleGrid = [

[0,0,0],

[0,1,0],

[0,0,0]

]

Output = 2

Explanation

In the given grid:

0 0 0

0 1 0

0 0 0

There are two unique paths for the robot to reach the bottom-right corner:

1. Right -> Right -> Down -> Down
2. Down -> Down -> Right -> Right

Constraints

- The grid dimensions `m` and `n` satisfy `1 <= m, n <= 100`.
- Each cell of the grid contains either `0` or `1`.

Approach

The problem can be solved using dynamic programming. We can iterate through each cell of the grid and fill it with the number of unique paths to reach that cell. At each cell, we can calculate the number of unique paths based on the values of the neighboring cells. We need to handle obstacle cells by setting their value to `0` since the robot cannot pass through them.

1. Initialize the starting cell with `1` if it's not an obstacle. If it's an obstacle, return `0` as there's no way to reach the end.
2. Fill the values for the first column and first row of the grid based on the previous cell's value and whether the current cell is an obstacle.
3. Iterate through the rest of the grid, updating each cell's value based on the values of the neighboring cells. If the current cell is an obstacle, set its value to `0`.
4. Finally, return the value in the bottom-right corner of the grid, which represents the number of unique paths.

Complexity Analysis

- **Time Complexity:** (O(m times n)), where (m) and (n) are the dimensions of the grid. We traverse each cell once.
- **Space Complexity:** (O(1)) as we are modifying the given grid in place without using any extra space.