

Problem 63: Unique Paths II

Problem Information

Difficulty: Medium

Acceptance Rate: 43.81%

Paid Only: No

Tags: Array, Dynamic Programming, Matrix

Problem Description

You are given an $m \times n$ integer array `grid`. There is a robot initially located at the **top-left corner** (i.e., `grid[0][0]`). The robot tries to move to the **bottom-right corner** (i.e., `grid[m - 1][n - 1]`). The robot can only move either down or right at any point in time.

An obstacle and space are marked as `1` or `0` respectively in `grid`. A path that the robot takes cannot include **any** square that is an obstacle.

Return `_` the number of possible unique paths that the robot can take to reach the bottom-right corner.

The testcases are generated so that the answer will be less than or equal to $2 * 10^9$.

Example 1:



Input: `obstacleGrid = [[0,0,0],[0,1,0],[0,0,0]]` **Output:** `2` **Explanation:** There is one obstacle in the middle of the 3x3 grid above. There are two ways to reach the bottom-right corner: 1. Right -> Right -> Down -> Down 2. Down -> Down -> Right -> Right

Example 2:



Input: `obstacleGrid = [[0,1],[0,0]]` **Output:** `1`

****Constraints:****

* `m == obstacleGrid.length` * `n == obstacleGrid[i].length` * `1 <= m, n <= 100` *
`obstacleGrid[i][j]` is `0` or `1`.

Code Snippets

C++:

```
class Solution {
public:
    int uniquePathsWithObstacles(vector<vector<int>>& obstacleGrid) {

    }
};
```

Java:

```
class Solution {
    public int uniquePathsWithObstacles(int[][] obstacleGrid) {

    }
}
```

Python3:

```
class Solution:
    def uniquePathsWithObstacles(self, obstacleGrid: List[List[int]]) -> int:
```