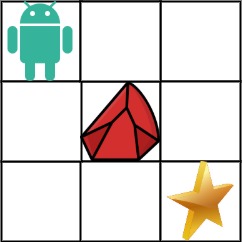
A robot is located at the top-left corner of a m x n grid (marked 'Start' in the diagram below).

The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corner of the grid (marked 'Finish' in the diagram below).

Now consider if some obstacles are added to the grids. How many unique paths would there be?

An obstacle and space is marked as 1 and 0 respectively in the grid.

**Example 1:**



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

**Output:** 2

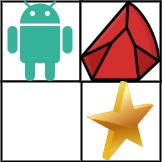
**Explanation:** There is one obstacle in the middle of the 3x3 grid above.

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

**Solution:**

class Solution {

public int uniquePathsWithObstacles(int[][] dp) {

// If the starting cell has an obstacle, then simply return as there would be

// no paths to the destination.

if (dp[0][0] == 1) {

return 0;

}

// Number of ways of reaching the starting cell = 1.

dp[0][0] = 1;

// Filling the values for the first column

for(int i=1; i<dp.length; i++){

dp[i][0] = (dp[i][0] == 0 && dp[i - 1][0] == 1) ? 1 : 0;

}

// Filling the values for the first row

for(int i=1;i<dp[0].length;i++){

dp[0][i] = (dp[0][i] == 0 && dp[0][i - 1] == 1) ? 1 : 0;

}

// Starting from cell(1,1) fill up the values

// No. of ways of reaching cell[i][j] = cell[i - 1][j] + cell[i][j - 1]

// i.e. From above and left.

for(int i=1; i<dp.length; i++){

for(int j=1;j<dp[i].length;j++){

if (dp[i][j] == 0) {

dp[i][j] = dp[i - 1][j] + dp[i][j - 1];

} else {

dp[i][j] = 0;

}

}

}

// Return value stored in rightmost bottommost cell. That is the destination.

return dp[dp.length-1][dp[0].length-1];

}

}