Coins Change & Unique Paths 升级版

笔记本： DP Note
创建时间： 10/13/2019 11:10 AM **更新时间：** 10/21/2019 7:19 PM
作者： tanziqi1756@outlook.com

Leetcode 62 Unique Path II 机器人走路 但是有障碍物

63. Unique Paths II

难度 中等 172 收藏 分享 切换为中文 关注

题目描述 评论 (344) 题解(61) New 提交记录

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 empty space is marked as `1` and `0` respectively in the grid.

Note: *m* and *n* will be at most 100.

Example 1:

```
Input:
[
  [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
```

63. Unique Paths II

难度 中等 172 收藏 分享 切换为中文

题目描述 评论 (344) 题解(61) New 提交

执行结果： 通过 显示详情 >

执行用时： 1 ms , 在所有 java 提交中击败了 97.77% 的用户

内存消耗： 38.8 MB , 在所有 java 提交中击败了 5.08% 的用户

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

        int x = obstacleGrid.length;
        if( x == 0 ) {
            return 0;
        }
        int y = obstacleGrid[0].length;
        if( y == 0 ) {
            return 0;
        }
        if (obstacleGrid[0][0] == 1) {
            return 0;
        }
        // initialize 1st column
        for( int i = 0, flag = 0; i < x; i++ ) {
            if( obstacleGrid[i][0] == 1 ) {
                obstacleGrid[i][0] = -1;
                flag = 1;
            }
            else {
                obstacleGrid[i][0] = 1;
            }
            if( flag == 1 ) {
                obstacleGrid[i][0] = -1;
            }
        }
        // initialize 1st row
        for( int j = 1, flag = 0; j < y; j++ ) {

            if( obstacleGrid[0][j] == 1 ) {
                obstacleGrid[0][j] = -1;
                flag = 1;
            }
            else {
                obstacleGrid[0][j] = 1;
            }
            if( flag == 1 ) {
                obstacleGrid[0][j] = -1;
            }
        }
```

```
        // dp
        for( int i = 1; i < x; i++ ) {
            for( int j = 1; j < y; j++ ) {
                if( obstacleGrid[i][j] == 1 ) {
                    obstacleGrid[i][j] = -1;
                    continue;
                }
                if( obstacleGrid[i-1][j] == -1 && obstacleGrid[i][j-1] == -1 ) {
                    obstacleGrid[i][j] = -1;
                }
                else if( obstacleGrid[i-1][j] == -1 ) {
                    obstacleGrid[i][j] = obstacleGrid[i][j-1];
                }
                else if( obstacleGrid[i][j-1] == -1 ) {
                    obstacleGrid[i][j] = obstacleGrid[i-1][j];
                }
                else {
                    obstacleGrid[i][j] = obstacleGrid[i-1][j] + obstacleGrid[i][j-1];
                }

            }
        }

        return obstacleGrid[x-1][y-1] >= 0 ? obstacleGrid[x-1][y-1] : 0;

    }
}
```