0063. 不同路径 II

ITCharge 大约 2 分钟

- 标签：数组、动态规划、矩阵
- 难度：中等

题目链接

- 0063. 不同路径 II - 力扣

题目大意

描述：一个机器人位于一个 $m \times n$ 网格的左上角。机器人每次只能向下或者向右移动一步。机器人试图达到网格的右下角。但是网格中有障碍物，不能通过。

现在给定一个二维数组表示网格，1 代表障碍物，0 表示空位。

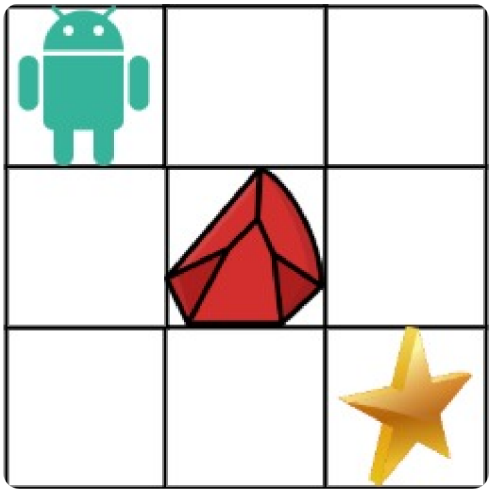
要求：计算出从左上角到右下角会有多少条不同的路径。

说明：

- $m == obstacleGrid.length$。
- $n == obstacleGrid[i].length$。
- $1 \le m, n \le 100$。
- $obstacleGrid[i][j]$ 为 0 或 1。

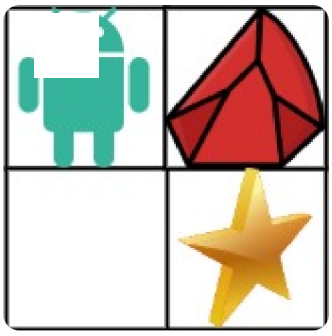
示例：

- 示例 1：



```py
输入：obstacleGrid = [[0,0,0],[0,1,0],[0,0,0]]
输出：2
解释：3x3 网格的正中间有一个障碍物。
从左上角到右下角一共有 2 条不同的路径：
1. 向右 -> 向右 -> 向下 -> 向下
2. 向下 -> 向下 -> 向右 -> 向右
```

- 示例 2：



```py
输入：obstacleGrid = [[0,1],[0,0]]
输出：1
```

解题思路

思路 1：动态规划

1. 划分阶段

按照路径的结尾位置（行位置、列位置组成的二维坐标）进行阶段划分。

2. 定义状态

定义状态 $dp[i][j]$ 表示为：从 $(0, 0)$ 到 (i, j) 的不同路径数。

3. 状态转移方程

因为我们每次只能向右、或者向下移动一步，因此想要走到 (i, j)，只能从 $(i - 1, j)$ 向下走一步走过来；或者从 $(i, j - 1)$ 向右走一步走过来。则状态转移方程为：$dp[i][j] = dp[i - 1][j] + dp[i][j - 1]$，其中 $obstacleGrid[i][j] == 0$。

4. 初始条件

- 对于第一行、第一列，因为只能超一个方向走，所以 $dp[i][0] = 1$，$dp[0][j] = 1$。如果在第一行、第一列遇到障碍，则终止赋值，跳出循环。

5. 最终结果

根据我们之前定义的状态，$dp[i][j]$ 表示为：从 $(0, 0)$ 到 (i, j) 的不同路径数。所以最终结果为 $dp[m - 1][n - 1]$。

思路 1：代码

```py
class Solution:
    def uniquePathsWithObstacles(self, obstacleGrid: List[List[int]]) -> int:
        m = len(obstacleGrid)
        n = len(obstacleGrid[0])
        dp = [[0 for _ in range(n)] for _ in range(m)]

        for i in range(m):
            if obstacleGrid[i][0] == 1:
                break
            dp[i][0] = 1

        for j in range(n):
            if obstacleGrid[0][j] == 1:
                break
            dp[0][j] = 1
```

```
        for i in range(1, m):
            for j in range(1, n):
                if obstacleGrid[i][j] == 1:
                    continue
                dp[i][j] = dp[i - 1][j] + dp[i][j - 1]
        return dp[m - 1][n - 1]
```

思路 1：复杂度分析

- **时间复杂度**：$O(m \times n)$。
- **空间复杂度**：$O(m \times n)$。