0062. 不同路径

ITCharge 大约 3 分钟

- 标签：数学、动态规划、组合数学
- 难度：中等

题目链接

- 0062. 不同路径 - 力扣

题目大意

描述：给定两个整数 m 和 n，代表大小为 $m \times n$ 的棋盘， 一个机器人位于棋盘左上角的位置，机器人每次只能向右、或者向下移动一步。

要求：计算出机器人从棋盘左上角到达棋盘右下角一共有多少条不同的路径。

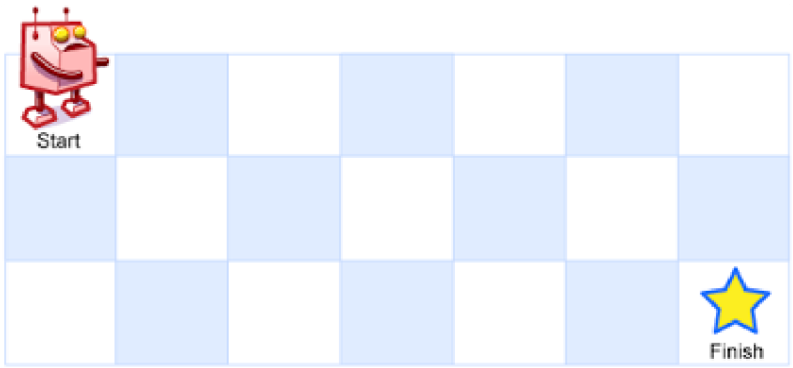
说明：

- $1 \leq m, n \leq 100$。
- 题目数据保证答案小于等于 2×10^9。

示例：

- 示例 1：

```py
输入：m = 3, n = 7
输出：28
```



- 示例 2：

```py
输入：m = 3, n = 2
输出：3
解释：
从左上角开始，总共有 3 条路径可以到达右下角。
1. 向右 -> 向下 -> 向下
2. 向下 -> 向下 -> 向右
3. 向下 -> 向右 -> 向下
```

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 $dp[i][j]$ 为：从左上角到达位置 (i, j) 的路径数量。

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。所以可以写出状态转移方程为：$dp[i][j] = dp[i - 1][j] + dp[i][j - 1]$，此时 $i > 0, j > 0$。

4. 初始条件

- 从左上角走到 $(0, 0)$ 只有一种方法，即 $dp[0][0] = 1$。
- 第一行元素只有一条路径（即只能通过前一个元素向右走得到），所以 $dp[0][j] = 1$。
- 同理，第一列元素只有一条路径（即只能通过前一个元素向下走得到），所以 $dp[i][0] = 1$。

5. 最终结果

根据状态定义，最终结果为 $dp[m-1][n-1]$，即从左上角到达右下角 $(m-1,n-1)$ 位置的路径数量为 $dp[m-1][n-1]$。

思路 1：动态规划代码

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

        for j in range(n):
            dp[0][j] = 1
        for i in range(m):
            dp[i][0] = 1

        for i in range(1, m):
            for j in range(1, n):
                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**：$O(m \times n)$。初始条件赋值的时间复杂度为 $O(m+n)$，两重循环遍历的时间复杂度为 $O(m \times n)$，所以总体时间复杂度为 $O(m \times n)$。
- **空间复杂度**：$O(m \times n)$。用到了二维数组保存状态，所以总体空间复杂度为 $O(m \times n)$。因为 $dp[i][j]$ 的状态只依赖于上方值 $dp[i-1][j]$ 和左侧值 $dp[i][j-1]$，而我们在进行遍历时的顺序刚好是从上至下、从左到右。所以我们可以使用长度为 n 的一维数组来保存状态，从而将空间复杂度优化到 $O(n)$。