# 题目

一个机器人位于一个m×n网格的左上角 （起始点在下图中标记为“Start”）。

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（在下图中标记为“Finish”）。

问总共有多少条不同的路径？

示例1：



输入：m = 3, n = 7

输出：28

示例 2：

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

示例 3：

输入：m = 7, n = 3

输出：28

示例 4：

输入：m = 3, n = 3

输出：6

提示：

1 <= m, n <= 100

题目数据保证答案小于等于 2 \* 109

# 分析

## 方法一：动态规划

**思路：**

我们用f(i,j)表示从左上角走到(i,j) 的路径数量，其中i和j的范围分别是[0,m)和[0,n)。

由于我们每一步只能从向下或者向右移动一步，因此要想走到(i,j)，如果向下走一步，那么会从(i−1,j)走过来；如果向右走一步，那么会从(i,j−1)走过来。因此我们可以写出动态规划转移方程：

f(i,j)=f(i−1,j)+f(i,j−1)

需要注意的是，如果i=0，那么f(i−1,j)并不是一个满足要求的状态，我们需要忽略这一项；同理，如果j=0，那么f(i,j−1)并不是一个满足要求的状态，我们需要忽略这一项。

初始条件为f(0,0)=1，即从左上角走到左上角有一种方法。

最终的答案即为f(m−1,n−1)。

**细节：**

为了方便代码编写，我们可以将所有的f(0,j)以及f(i,0)都设置为边界条件，它们的值均为1。

**代码：**

class Solution {

public:

int uniquePaths(int m, int n) {

vector<vector<int>> f(m, vector<int>(n));

for (int i = 0; i < m; ++i) {

f[i][0] = 1;

}

for (int j = 0; j < n; ++j) {

f[0][j] = 1;

}

for (int i = 1; i < m; ++i) {

for (int j = 1; j < n; ++j) {

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

}

}

return f[m - 1][n - 1];

}

};

复杂度分析：

时间复杂度：O(mn)。

空间复杂度：O(mn)，即为存储所有状态需要的空间。注意到f(i,j) 仅与第i行和第i−1行的状态有关，因此我们可以使用滚动数组代替代码中的二维数组，使空间复杂度降低为O(n)。此外，由于我们交换行列的值并不会对答案产生影响，因此我们总可以通过交换m和n使得m≤n，这样空间复杂度降低至O(min(m,n))。