1. 题目
2. 题干

一个机器人位于一个 m x 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？

1. 示例

示例 1:

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 -> 向右 -> 向下

2. 向右 -> 向下 -> 向右

3. 向下 -> 向右 -> 向右

示例 2:

输入: m = 7, n = 3

输出: 28

1. 题解
2. 思路

这是一个组合问题，从某个点走到某个点有多少种路径。首先，规则是只允许往右和往下走，每次只能走一格，因此，我们可以得知每个点只能通过它上面的点和左边的点到达。这样就可以构造状态转移方程了。设置一个二维数组dp[][]来存储从起始点(0,0)走到(i,j)的路径数量，那么状态转移方程为： dp[i][j] = dp[i-1][j] +dp[i][j-1]。由于(i,j-1)和(i-1,j)能够到达(i,j)，因此他们两个的dp数组的值加起来就是dp[i][j]的值。

确定边界条件：当i=0或者j=0时会从状态转移方程中越界，而当i=0是处在最上面一行之中，而最上面一行的点都只有一条路径可以到达，因为从起始点（0,0）只能往右才能到达最上面一行的每个点，因此dp[0][j] = 1。同理，当j=0时也只有一条路径可以到达，因此dp[i][0] = 1;

1. 代码实现

JAVA：

class Solution {

    public int uniquePaths(int m, int n) {

        int[][] dp = new int[m][n];

        for(int i = 0; i <m; ++i){

            dp[i][0] = 1;

        }

        for(int i = 0; i < n; ++i){

            dp[0][i] = 1;

        }

        for(int i = 1; i < m; ++i){

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

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

            }

        }

        return dp[m-1][n-1];

    }

}