# 题目

一只青蛙一次可以跳上1级台阶，也可以跳上2级台阶。求该青蛙跳上一个 n 级的台阶总共有多少种跳法。

答案需要取模 1e9+7（1000000007），如计算初始结果为：1000000008，请返回 1。

**示例 1：**

输入：n = 2

输出：2

**示例 2：**

输入：n = 7

输出：21

提示：

0 <= n <= 100

**注意：**本题与主站 70 题相同（爬楼梯）：

https://leetcode-cn.com/problems/climbing-stairs/

# 分析

## 方法一：递归

**思路：**

此类求多少种可能性 的题目一般都有递推性质 ，即f(n)和f(n−1)…f(1)之间是有联系的。

设跳上n级台阶有f(n)种跳法。在所有跳法中，青蛙的最后一步只有两种情况： 跳上1级或2级台阶。

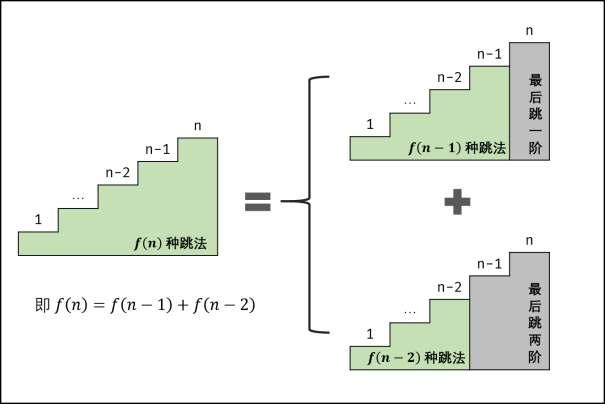
当为1级台阶： 剩n−1个台阶，此情况共有f(n−1)种跳法；

当为2级台阶： 剩n−2个台阶，此情况共有f(n−2)种跳法。

f(n)为以上两种情况之和，即f(n)=f(n-1)+f(n-2)，以上递推性质为斐波那契数列。本题可转化为 求斐波那契数列第n项的值 ，与面试题10- I. 斐波那契数列 等价，唯一的不同在于起始数字不同。

青蛙跳台阶问题：f(0)=1,f(1)=1, f(2)=2；

斐波那契数列问题：f(0)=0, ff(1)=1, f(2)=1。



斐波那契数列的定义是f(n+1)=f(n)+f(n−1)，生成第n项的做法有以下几种：

**递归法：**

原理：把f(n)问题的计算拆分成f(n−1)和f(n−2)两个子问题的计算，并递归，以f(0)和f(1)为终止条件。

缺点：大量重复的递归计算，例如f(n)和f(n−1)两者向下递归都需要计算f(n−2) 的值。

**记忆化递归法：**

原理：在递归法的基础上，新建一个长度为n的数组，用于在递归时存储f(0)至f(n)的数字值，重复遇到某数字时则直接从数组取用，避免了重复的递归计算。

缺点：记忆化存储的数组需要使用O(N)的额外空间。

**动态规划：**

原理：以斐波那契数列性质f(n + 1) = f(n) + f(n - 1)为转移方程。

从计算效率、空间复杂度上看，动态规划是本题的最佳解法。

**代码：**

class Solution {

public:

int numWays(int n) {

if(n == 0 || n == 1)

return 1;

return numWays(n-1) + numWays(n-2);

}

};

## 方法二：动态规划

**思路：**

**代码：**

class Solution {

public:

int numWays(int n) {

vector<int> v(n+1,1);

for(int i=2;i<=n;i++)

{

v[i] = (v[i-1] + v[i-2]) % 1000000007;

}

return v[n];

}

};

或：

class Solution {

public:

    int numWays(int n) {

        if(0 == n)  return 1; //不要忘记n=0的情况

        if(1 == n)  return 1;

        if(2 == n)  return 2;

        vector<int> dp(n+1);

        dp[1] = 1;

        dp[2] = 2;

        for(int i=3;i<=n;i++)

        {

            dp[i] = (dp[i-1] + dp[i-2]) % 1000000007;

        }

        return dp[n];

    }

};