# 题目

假设你正在爬楼梯。需要 n 阶你才能到达楼顶。

每次你可以爬1或2个台阶。你有多少种不同的方法可以爬到楼顶呢？

注意：给定n是一个正整数。

**示例 1：**

输入：2

输出：2

解释：有两种方法可以爬到楼顶。

1. 1阶+ 1阶

2. 2阶

**示例 2：**

输入：3

输出：3

解释：有三种方法可以爬到楼顶。

1. 1阶+ 1阶+ 1阶

2. 1阶+ 2阶

3. 2阶+ 1阶

类似题目：剑指offer 10-II 青蛙跳台阶问题

# 分析

## 方法一：递归

**思路：**

**代码：**

class Solution {

public:

    int climbStairs(int n) {

        if(n<0)

            return 0;

        if(1==n)

            return 1;

        if(2==n)

            return 2;

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

    }

};

## 方法二：动态规划

**思路：**

每一步结果都是前面1步或者2步。

**状态转移方程：**

dp[n]=dp[n−1]+dp[n−2]

**初始值：**

dp[0]=1和dp[1]=1

**代码：**

class Solution {

public:

int climbStairs(int n) {

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

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

{

v[i] = (v[i-1] + v[i-2]);//是否需要考虑溢出问题

}

return v[n];

}

};

时间复杂度：O(n)

或：

// 版本一

class Solution {

public:

int climbStairs(int n) {

if (n <= 1) return n; // 因为下面直接对dp[2]操作了，防止空指针

vector<int> dp(n + 1); //因为dp[0]无意义，所以多申请一个内存 dp[1] = 1;

dp[2] = 2;

for (int i = 3; i <= n; i++) { // 注意i是从3开始的

dp[i] = dp[i - 1] + dp[i - 2];

}

return dp[n];

}

};