# 题目

三步问题。有个小孩正在上楼梯，楼梯有n阶台阶，小孩一次可以上1阶、2阶或3阶。实现一种方法，计算小孩有多少种上楼梯的方式。结果可能很大，你需要对结果模1000000007。

**示例1:**

输入：n = 3

输出：4

说明: 有四种走法

**示例2:**

输入：n = 5

输出：13

**提示:**

n范围在[1, 1000000]之间

# 分析

## 方法一：动态规划

**思路：**

三步问题，采用动态规划（思路和上楼梯问题一致），n层楼梯的走法可以分为三种情况：

1. 剩余一层楼梯要走然后一步走一层；
2. 剩余两层楼梯要走，然后一步走两层；
3. 剩余三层楼梯要走，然后一步走三层

dp[n] = dp[n-1] + dp[n-2] + dp[n-3]

**代码：**

class Solution {

public:

int waysToStep(int n) {

if(n == 1) return 1;

if(n == 2) return 2;

if(n == 3) return 4;

vector<long> dp(n+1);

dp[1] = 1;

dp[2] = 2;

dp[3] = 4;

for(int i =4;i <= n;i ++){

/\*

取模，对两个较大的数之和取模再对整体取模，防止越界

假如对三个dp[i-n]都 % 1000000007，那么也是会出现越界情况（导致溢出变为负数的问题）

因为如果本来三个dp[i-n]都接近1000000007那么取模后仍然不变，但三个相加则溢出

但对两个较大的dp[i-n]:dp[i-2],dp[i-3]之和mod 1000000007，那么这两个较大的数相加大于 1000000007但又不溢出

取模后变成一个很小的数，与dp[i-1]相加也不溢出

所以取模操作也需要仔细分析

\*/

dp[i] = (dp[i - 1] + dp[i - 2] + dp[i - 3])%1000000007;

}

return (int)dp[n];

}

};

或：

class Solution {

public:

    int waysToStep(int n) {

        if(n == 1) return 1;

        if(n == 2) return 2;

        if(n == 3) return 4;

        vector<long> dp(n+1);

        dp[1] = 1;

        dp[2] = 2;

        dp[3] = 4;

        for(int i=4;i<=n;i++)

        {

            dp[i] = (dp[i-1] + dp[i-2] + dp[i-3]);

//每一个值都要取余，而不是最后的值

        }

        return (int)dp[n] % 1000000007;

    }

};