70-爬楼梯

题述

70. 爬楼梯

难度 简单 2343

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?

示例 1:

输入: n = 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: n = 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

思路

这道题一开始可能会感觉无从下手，比较难，先列几个例子然后推导。

爬到第一层楼梯有一种方法，爬到二层楼梯有两种方法。

那么第一层楼梯再跨两步就到第三层 ， 第二层楼梯再跨一步就到第三层。

所以到第三层楼梯的状态可以由第二层楼梯 和 到第一层楼梯状态推导出来，那么就可以想到动态规划了。

动态规划

1、确定确定dp数组以及下标的含义

- dp[i]： 爬到第i层楼梯，有dp[i]种方法

2、确定递推公式

从dp[i]的定义可以看出，dp[i] 可以有两个方向推出来。

首先是dp[i - 1]，上i-1层楼梯，有dp[i - 1]种方法，那么再一步跳一个台阶不就是dp[i]了么。

还有就是dp[i - 2]，上i-2层楼梯，有dp[i - 2]种方法，那么再一步跳两个台阶不就是dp[i]了么。

那么dp[i]就是 dp[i - 1]与dp[i - 2]之和！

所以dp[i] = dp[i - 1] + dp[i - 2] 。

3、dp数组如何初始化

不用考虑下标为0时的情况 楼层是0，直接站楼顶上了，就是不用方法，dp[0]就应该是0

dp[1] = 1，dp[2] = 2，

4、确定遍历顺序

从递推公式dp[i] = dp[i - 1] + dp[i - 2];中可以看出，遍历顺序一定是从前向后遍历的

5、举例推导dp数组

举例当n为5的时候，dp table（dp数组）应该是这样的

1 2 3 5 8

题解

C++

```
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);
        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];
    }
};
```

Python

```python
class Solution:
    def climbStairs(self, n: int) -> int:
        # dp[i] 为第 i 阶楼梯有多少种方法爬到楼顶
        # 动态规划

        # 初始化dp数组
        dp = [0] * (n+1)
        dp[0] = 1
        dp[1] = 1
        for i in range(2,n+1):
            dp[i] = dp[i-1] + dp[i-2]
        return dp[n]
```

思考

这道题做完之后会觉得和斐波那契数列差不多，可实际做起来却有种无力感，因为这道题需要我们从零开始去逐个分析动态规划的五个步骤，在这个过程中慢慢尝试掌握。