746.使用最小花费爬楼梯

题目地址（746.使用最小花费爬楼梯）

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

标签

- 动态规划

难度

- 简单

入选理由

1. 我们讲几道爬楼梯以及爬楼梯的换皮题。 让大家感受一下套路是什么

题目描述

数组的每个下标作为一个阶梯，第 i 个阶梯对应着一个非负数的体力花费值 cost[i]（下标从 0 开始）。

每当你爬上一个阶梯你都要花费对应的体力值，一旦支付了相应的体力值，你就可以选择向上爬一个阶梯或者爬两个阶梯。

请你找出达到楼层顶部的最低花费。在开始时，你可以选择从下标为 0 或 1 的元素作为初始阶梯。

示例 1:

输入: cost = [10, 15, 20]
输出: 15
解释: 最低花费是从 cost[1] 开始，然后走两步即可到阶梯顶，一共花费 15 。
 示例 2:

输入: cost = [1, 100, 1, 1, 1, 100, 1, 1, 100, 1]
输出: 6
解释: 最低花费方式是从 cost[0] 开始，逐个经过那些 1 ，跳过 cost[3] ，一共花费 6 。

提示:

```
cost 的长度范围是 [2, 1000]。
cost[i] 将会是一个整型数据，范围为 [0, 999] 。
```

前置知识

- 动态规划

分析

该题其实就是讲义中爬楼梯的变形题目，核心思路是不变的，只不过所求目标变成了**登完所有台阶所需要的最小花费**

- 定义 dp 数组，dp[i]定义为登完 i 阶台阶所需最小花费（子问题）
- 思考：登完当前第 i 阶台阶所需花费是第 i 阶台阶消耗体力+（dp[i-1] or dp[i - 2]），由于所求为最小，故可得状态转移方程为：

$$dp[i] = \min(dp[i-1], dp[i-2]) + cost[i]$$

- 由于我们需要前两个 dp 数组位置的值，因此我们需要先初始化 dp[0] 和 dp[1] 然后再使用转移方程算出其他 dp。 dp[0] 和 dp[1] 对应第 1 阶和第 2 阶的最小体力。

代码：

代码支持：Java，CPP,Python

Java Code:

```java
class Solution {
    public int minCostClimbingStairs(int[] cost) {

        if (cost == null || cost.length == 0)
            return 0;

        int[] dp = new int[cost.length + 1];
        dp[0] = cost[0];
        dp[1] = cost[1];

        for(int i = 2; i <= cost.length; i++)
            dp[i] = Math.min(dp[i - 1], dp[i - 2]) + (i == cost.length ? 0 : cost[i]);

        return dp[cost.length];
    }
}
```

CPP Code:

```cpp
class Solution {
public:
    int minCostClimbingStairs(vector<int>& cost) {
      int n = cost.size();
      vector<int> dp(n + 1);
      dp[0] = dp[1] = 0;
      for(int i = 2; i <= n; i++){
        dp[i] = min(dp[i - 1] + cost[i - 1], dp[i - 2] + cost[i - 2]);
      }

      return dp[n];
    }
};
```

Python Code:

```python
class Solution:
    def minCostClimbingStairs(self, cost: List[int]) -> int:
        dp = [0] * (len(cost)+1)
        dp[0], dp[1] = cost[0], cost[1]
        for i in range(2, len(cost)+1):
            dp[i] = min(dp[i-1], dp[i-2]) + (cost[i] if i != len(cost) else 0)
        return dp[-1]
```

复杂度分析

设：N阶台阶

- 时间复杂度：$O(N)$

- 空间复杂度：$O(N)$

进阶：尝试将空间复杂度优化到$O(1)$

> 提示：使用滚动数组优化，即用两个变量记录前一个状态和前前一个状态。