# 746-使用最小花费爬楼梯

# 题述

#### 746. 使用最小花费爬楼梯

难度 简单 凸 870 ☆ □ 🛕 🗅

给你一个整数数组 cost , 其中 cost[i] 是从楼梯第 i 个台阶向上爬需要支付的费用。一旦你支付此费用,即可选择向上爬一个或者两个台阶。

你可以选择从下标为 0 或下标为 1 的台阶开始爬楼梯。

请你计算并返回达到楼梯顶部的最低花费。

#### 示例 1:

输入: cost = [10,15,20]

输出: 15

解释: 你将从下标为 1 的台阶开始。

- 支付 15 ,向上爬两个台阶,到达楼梯顶部。

总花费为 15 。

#### 示例 2:

输入: cost = [1,100,1,1,1,100,1,1,100,1]

输出:6

解释: 你将从下标为 0 的台阶开始。

- 支付 1 , 向上爬两个台阶, 到达下标为 2 的台阶。
- 支付 1 , 向上爬两个台阶, 到达下标为 4 的台阶。
- 支付 1 , 向上爬两个台阶, 到达下标为 6 的台阶。
- 支付 1 ,向上爬一个台阶,到达下标为 7 的台阶。
- 支付 1 , 向上爬两个台阶, 到达下标为 9 的台阶。
- 支付 1 , 向上爬一个台阶, 到达楼梯顶部。

总花费为 6。

## 思路

# 动态规划

这道题是爬楼梯的升级版本,需要考虑花费问题

注意题目描述: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

#### 1、确定dp数组以及下标的含义

dp[i]的定义: 到达第i个台阶所花费的最少体力为dp[i]。(注意这里认为是第一步一定是要花费)

#### 2、确定递推公式

可以有两个途径得到dp[i],一个是dp[i-1] 一个是dp[i-2]。

至于选择哪一个,我们应该选择较小的那一个,所以是取min

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

加cost[i]的原因--当你爬上一个阶梯你都要花费对应的体力值

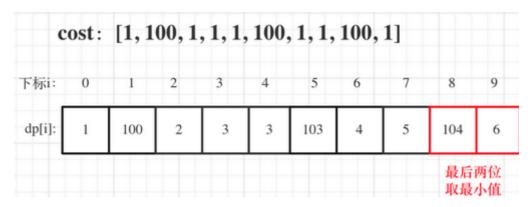
#### 3、dp数组如何初始化

- dp[0] = cost[0]
- dp[1] = cost[1]

#### 4、确定遍历顺序

因为是模拟台阶,而且dp[i]又dp[i-1]dp[i-2]推出,所以是从前到后遍历cost数组就可以了。

### 5、举例推导dp数组



# 题解

# 动态规划

```
class Solution:
def minCostClimbingStairs(self, cost: List[int]) -> int:
    # 爬楼梯进阶版
    # 带有花费的爬楼梯
    # 递推公式 dp[i] = min(dp[i - 1], dp[i - 2]) + cost[i];
    dp = [0] * len(cost)
    dp[0] = cost[0]
    dp[1] = cost[1]
    for i in range(2,len(cost)):
        dp[i] = min(dp[i-1],dp[i-2]) + cost[i]
    return min(dp[len(cost)-1],dp[len(cost)-2])
```

# 思考