0198. 打家劫舍

ITCharge 大约 2 分钟

- 标签：数组、动态规划
- 难度：中等

题目链接

- 0198. 打家劫舍 - 力扣

题目大意

描述：给定一个数组 $nums$，$nums[i]$ 代表第 i 间房屋存放的金额。相邻的房屋装有防盗系统，假如相邻的两间房屋同时被偷，系统就会报警。

要求：假如你是一名专业的小偷，计算在不触动警报装置的情况下，一夜之内能够偷窃到的最高金额。

说明：

- $1 \le nums.length \le 100$。
- $0 \le nums[i] \le 400$。

示例：

- 示例 1：

```py
输入：[1,2,3,1]
输出：4
解释：偷窃 1 号房屋 (金额 = 1) ，然后偷窃 3 号房屋 (金额 = 3)。
     偷窃到的最高金额 = 1 + 3 = 4。
```

- 示例 2：

```py
输入：[2,7,9,3,1]
输出：12
解释：偷窃 1 号房屋 (金额 = 2), 偷窃 3 号房屋 (金额 = 9)，接着偷窃 5 号房屋 (金额 = 1)。
     偷窃到的最高金额 = 2 + 9 + 1 = 12。
```

解题思路

思路 1：动态规划

1. 划分阶段

按照房屋序号进行阶段划分。

2. 定义状态

定义状态 $dp[i]$ 表示为：前 i 间房屋所能偷窃到的最高金额。

3. 状态转移方程

i 间房屋的最后一个房子是 $nums[i - 1]$。

如果房屋数大于等于 2 间，则偷窃第 $i - 1$ 间房屋的时候，就有两种状态：

1. 偷窃第 $i - 1$ 间房屋，那么第 $i - 2$ 间房屋就不能偷窃了，偷窃的最高金额为：前 $i - 2$ 间房屋的最高总金额 + 第 $i - 1$ 间房屋的金额，即 $dp[i] = dp[i - 2] + nums[i - 1]$；
2. 不偷窃第 $i - 1$ 间房屋，那么第 $i - 2$ 间房屋可以偷窃，偷窃的最高金额为：前 $i - 1$ 间房屋的最高总金额，即 $dp[i] = dp[i - 1]$。

然后这两种状态取最大值即可，即状态转移方程为：

$$dp[i] = \begin{cases} nums[0] & i = 1 \\ max(dp[i - 2] + nums[i - 1], dp[i - 1]) & i \ge 2 \end{cases}$$

4. 初始条件

- 前 0 间房屋所能偷窃到的最高金额为 0，即 $dp[0] = 0$。
- 前 1 间房屋所能偷窃到的最高金额为 $nums[0]$，即：$dp[1] = nums[0]$。

5. 最终结果

根据我们之前定义的状态，$dp[i]$ 表示为：前 i 间房屋所能偷窃到的最高金额。则最终结果为 $dp[size]$，$size$ 为总的房屋数。

思路 1：代码

```py
class Solution:
    def rob(self, nums: List[int]) -> int:
        size = len(nums)
        if size == 0:
            return 0

        dp = [0 for _ in range(size + 1)]
        dp[0] = 0
        dp[1] = nums[0]

        for i in range(2, size + 1):
            dp[i] = max(dp[i - 2    nums[i - 1], dp[i - 1])

        return dp[size]
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$。一重循环遍历的时间复杂度为 $O(n)$。
- **空间复杂度**：$O(n)$。用到了一维数组保存状态，所以总体空间复杂度为 $O(n)$。