本日学习内容

1. 完成博客两篇（二分查找 + 打家劫舍 总结）
2. 6道dp算法 和 几道二分查找
3. git三剑客一集

本日分享内容

题目1.

你是一个专业的小偷，计划偷窃沿街的房屋。每间房内都藏有一定的现金，影响你偷窃的唯一制约因素就是相邻的房屋装有相互连通的防盗系统，如果两间相邻的房屋在同一晚上被小偷闯入，系统会自动报警。

给定一个代表每个房屋存放金额的非负整数数组，计算你 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。

示例 1：

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：

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 。
提示：

1 <= nums.length <= 100
0 <= nums[i] <= 400

思路

首先考虑最简单的情况。如果只有一间房屋，则偷窃该房屋，可以偷窃到最高总金额。如果只有两间房屋，则由于两间房屋相邻，不能同时偷窃，只能偷窃其中的一间房屋，因此选择其中金额较高的房屋进行偷窃，可以偷窃到最高总金额。

如果房屋数量大于两间，应该如何计算能够偷窃到的最高总金额呢？对于第 k (k>2) 间房屋，有两个选项：

偷窃第 k 间房屋，那么就不能偷窃第 k−1 间房屋，偷窃总金额为前 k−2 间房屋的最高总金额与第 k 间房屋的金额之和。

不偷窃第 k 间房屋，偷窃总金额为前 k−1 间房屋的最高总金额。

在两个选项中选择偷窃总金额较大的选项，该选项对应的偷窃总金额即为前 k 间房屋能偷窃到的最高总金额。

用 dp[i] 表示前 i 间房屋能偷窃到的最高总金额，那么就有如下的状态转移方程：

$dp[i]=max(dp[i-2]+nums[i],dp[i-1])$

边界条件为：

$dp[0]=nums[0]$
$dp[1]=max(nums[0],nums[1])$

只有一间房屋，则偷窃该房屋
只有两间房屋，选择其中金额较高的房屋进行偷窃

最终的答案即为 dp[n−1]，其中 n 是数组的长度。

代码

```
public:
    int rob(vector<int>& nums) {
        int n = nums.size();

        if (nums.empty()) {
            return 0;
        }
        if (n == 1) {
            return nums[0];
        }

        vector<int> dp(n, 0);
        dp[0] = nums[0];
        dp[1] = max(nums[0], nums[1]);
        for (int i = 2; i < n; i++) {
            dp[i] = max(dp[i - 2] + nums[i], dp[i - 1]);
        }

        return dp[n - 1];
    }
};
```

题目2.

给你一个整数数组 `nums` ，你可以对它进行一些操作。

每次操作中，选择任意一个 `nums[i]` ，删除它并获得 `nums[i]` 的点数。之后，你必须删除 **所有** 等于 `nums[i] - 1` 和 `nums[i] + 1` 的元素。

开始你拥有 `0` 个点数。返回你能通过这些操作获得的最大点数。

示例 1：

```
输入：nums = [3,4,2]
输出：6
解释：
删除 4 获得 4 个点数，因此 3 也被删除。
之后，删除 2 获得 2 个点数。总共获得 6 个点数。
```

示例 2：

```
输入：nums = [2,2,3,3,3,4]
输出：9
解释：
删除 3 获得 3 个点数，接着要删除两个 2 和 4 。
之后，再次删除 3 获得 3 个点数，再次删除 3 获得 3 个点数。
总共获得 9 个点数。
```

思路

这道题与打家劫舍极为相似，只需遍历nums，用一个数组points[i] 表示每个i * 次数

代码

```cpp
class Solution {
public:
    int deleteAndEarn(vector<int>& nums) {
        if (nums.empty()) {
            return 0;
        }
        int maxVal = *max_element(nums.begin(), nums.end());
        vector<int> points(maxVal + 1, 0);
        vector<int> dp(maxVal + 1, 0);
        for (int num : nums) {
            points[num] += num;
        }
        dp[1] = points[1];
        for (int i = 2; i <= maxVal; i++) {
            dp[i] = max(dp[i - 1], dp[i - 2] + points[i]);
        }
        return dp[maxVal];
    }
};
```

本日遇到的问题

1. c++语法有些遗忘，有的函数返回迭代器却忘记加 *
2. 有时候不太能变通，容易生搬硬套之前的代码

明日学习内容

1. 开始完全背包的学习和练习（灵神题单 + 随想录）
2. 着手完善管理系统