本日学习内容

1. 完成股票买卖问题全部，总结为博客发布
2. 完成学生管理系统的功能

本日分享内容

题目一：2958. 最多 K 个重复元素的最长子数组

给你一个整数数组 `nums` 和一个整数 `k` 。

一个元素 `x` 在数组中的 **频率** 指的是它在数组中的出现次数。

如果一个数组中所有元素的频率都 **小于等于** `k` ，那么我们称这个数组是 **好** 数组。

请你返回 `nums` 中 **最长好** 子数组的长度。

子数组 指的是一个数组中一段连续非空的元素序列。

示例 1：

```
输入：nums = [1,2,3,1,2,3,1,2], k = 2
输出：6
解释：最长好子数组是 [1,2,3,1,2,3] ，值 1 ，2 和 3 在子数组中的频率都没有超过 k = 2 。[2,3,1,2,3,1] 和 [3,1,2,3,1,2] 也是好子数组。
最长好子数组的长度为 6 。
```

示例 2：

```
输入：nums = [1,2,1,2,1,2,1,2], k = 1
输出：2
解释：最长好子数组是 [1,2] ，值 1 和 2 在子数组中的频率都没有超过 k = 1 。[2,1] 也是好子数组。
最长好子数组的长度为 2 。
```

示例 3：

```
输入：nums = [5,5,5,5,5,5,5], k = 4
输出：4
解释：最长好子数组是 [5,5,5,5] ，值 5 在子数组中的频率没有超过 k = 4 。
最长好子数组的长度为 4 。
```

提示：

- `1 <= nums.length <= 105`
- `1 <= nums[i] <= 109`

- `1 <= k <= nums.length`

思路

我们可以使用滑动窗口（Sliding Window）的方法来解决这个问题。具体步骤如下：

1. **初始化指针和哈希表**：使用两个指针 `left` 和 `right` 来表示窗口的左右边界。`freq` 是一个哈希表，用来记录当前窗口中每个元素的出现次数。
2. **扩展窗口**：移动 `right` 指针，扩展窗口的右边界，同时更新 `freq` 中对应元素的出现次数。
3. **收缩窗口**：当 `freq` 中某个元素的出现次数超过 `k` 时，移动 `left` 指针，减少 `freq` 中对应元素的出现次数，直到所有元素的出现次数都不超过 `k`。
4. **更新结果**：在每次移动 `right` 指针后，计算当前窗口的长度（`right - left + 1`），并更新最大长度。

代码

```cpp
class Solution {
public:
    int maxSubarrayLength(vector<int>& nums, int k) {
        unordered_map<int, int> freq;
        int left = 0;
        int max_len = 0;

        for (int right = 0; right < nums.size(); ++right) {
            freq[nums[right]]++;

            while (freq[nums[right]] > k) {
                freq[nums[left]]--;
                left++;
            }

            max_len = max(max_len, right - left + 1);
        }

        return max_len;
    }
};
```

题目二：2024. 考试的最大困扰度

一位老师正在出一场由 `n` 道判断题构成的考试，每道题的答案为 true （用 `'T'` 表示）或者 false （用 `'F'` 表示）。老师想增加学生对自己做出答案的不确定性，方法是 **最大化** 有 **连续相同** 结果的题数。（也就是连续出现 true 或者连续出现 false）。

给你一个字符串 `answerKey` ，其中 `answerKey[i]` 是第 `i` 个问题的正确结果。除此以外，还给你一个整数 `k` ，表示你能进行以下操作的最多次数：

- 每次操作中，将问题的正确答案改为 `'T'` 或者 `'F'` （也就是将 `answerKey[i]` 改为 `'T'` 或者 `'F'` ）。

请你返回在不超过 `k` 次操作的情况下，**最大** 连续 `'T'` 或者 `'F'` 的数目。

示例 1：

```
输入：answerKey = "TTFF", k = 2
输出：4
解释：我们可以将两个 'F' 都变为 'T' ，得到 answerKey = "TTTT" 。
总共有四个连续的 'T' 。
```

示例 2：

```
输入：answerKey = "TFFT", k = 1
输出：3
解释：我们可以将最前面的 'T' 换成 'F' ，得到 answerKey = "FFFT" 。
或者，我们可以将第二个 'T' 换成 'F' ，得到 answerKey = "TFFF" 。
两种情况下，都有三个连续的 'F' 。
```

示例 3：

```
输入：answerKey = "TTFTTFTT", k = 1
输出：5
解释：我们可以将第一个 'F' 换成 'T' ，得到 answerKey = "TTTTTFTT" 。
或者我们可以将第二个 'F' 换成 'T' ，得到 answerKey = "TTFTTTTT" 。
两种情况下，都有五个连续的 'T' 。
```

提示：

- `n == answerKey.length`
- `1 <= n <= 5 * 104`
- `answerKey[i]` 要么是 `'T'` ，要么是 `'F'`
- `1 <= k <= n`

思路

1. 使用两个指针 `left` 和 `right` 来表示窗口的左右边界。`count_T` 和 `count_F` 分别记录当前窗口中 `'T'` 和 `'F'` 的数量。
2. 移动 `right` 指针，扩展窗口的右边界，同时更新 `count_T` 或 `count_F` 。
3. 当窗口中较少的字符的数量超过 `k` 时（即 `min(count_T, count_F) > k`），移动 `left` 指针，减少对应字符的数量，直到满足条件。

4. 在每次移动 `right` 指针后，计算当前窗口的长度（`right - left + 1`），并更新最大长度。

代码

```
public:
    int maxConsecutiveAnswers(string answerKey, int maxOperations) {
        int maxLength = 0;
        int countT = 0, countF = 0;
        int left = 0;

        for (int right = 0; right < answerKey.size(); ++right) {
            if (answerKey[right] == 'T') {
                ++countT;
            } else {
                ++countF;
            }

            while (min(countT, countF) > maxOperations) {
                if (answerKey[left] == 'T') {
                    --countT;
                } else {
                    --countF;
                }
                ++left;
            }

            maxLength = max(maxLength, right - left + 1);
        }

        return maxLength;
    }
};
```

本日遇到的问题

1. 管理系统功能有bug
2. 滑动窗口边界判断有时不清楚

明日学习内容

1. 开始复习算法：dp，kmp字符串
2. 开始写一个新博客（暂定关于时空复杂度）