本日学习内容

1. 修复管理系统bug，新增了管理员注册验证功能，上传到git仓库
2. 每日一题

本日分享内容

题目一：2364. 统计坏数对的数目

- 给你一个下标从 **0** 开始的整数数组 `nums` 。如果 `i < j` 且 `j - i != nums[j] - nums[i]` ，那么我们称 `(i, j)` 是一个 **坏**数对** 。

  请你返回 `nums` 中 **坏数对** 的总数目。

  示例 1：

  ```
  输入：nums = [4,1,3,3]
  输出：5
  解释：数对 (0, 1) 是坏数对，因为 1 - 0 != 1 - 4 。
  数对 (0, 2) 是坏数对，因为 2 - 0 != 3 - 4, 2 != -1 。
  数对 (0, 3) 是坏数对，因为 3 - 0 != 3 - 4, 3 != -1 。
  数对 (1, 2) 是坏数对，因为 2 - 1 != 3 - 1, 1 != 2 。
  数对 (2, 3) 是坏数对，因为 3 - 2 != 3 - 3, 1 != 0 。
  总共有 5 个坏数对，所以我们返回 5 。
  ```

  示例 2：

  ```
  输入：nums = [1,2,3,4,5]
  输出：0
  解释：没有坏数对。
  ```

  提示：

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

思路

我们想找到好数对只需满足nums[i] - i == nums[j] - j 的 i < j 数对有欧少个，然后总的数对数量是

总数对 = n * (n - 1) / 2
坏数对 = 总数对 - 好数对

代码

```cpp
class Solution {
public:
    long long countBadPairs(vector<int>& nums) {
        unordered_map<int, long long> mp;
        long long good = 0, n = nums.size();

        for (int i = 0; i < n; ++i) {
            int key = nums[i] - i;
            good += mp[key]; // 当前数对可以和之前相同 key 的配对构成好数对
            mp[key]++;
        }

        long long total = n * (n - 1) / 2;
        return total - good; // 坏数对 = 总数对 - 好数对
    }
};
```

题目二：1004. 最大连续1的个数 III

- 给定一个二进制数组 `nums` 和一个整数 `k`，假设最多可以翻转 `k` 个 `0` ，则返回执行操作后 *数组中连续 `1` 的最大个数* 。

示例 1：

```
输入：nums = [1,1,1,0,0,0,1,1,1,1,0], K = 2
输出：6
解释：[1,1,1,0,0,1,1,1,1,1,1]
粗体数字从 0 翻转到 1，最长的子数组长度为 6。
```

示例 2：

```
输入：nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], K = 3
输出：10
解释：[0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,1]
粗体数字从 0 翻转到 1，最长的子数组长度为 10。
```

提示：

- `1 <= nums.length <= 105`

- nums[i]`不是`0`就是`1`
- `0 <= k <= nums.length`

思路

统计窗口内 0 的个数 $cnt0$，则问题转换成在 $cnt0 \le k$ 的前提下，窗口大小的最大值。left初始值为0， for循环遍历right。如果窗口内有k个0 left向右移动，知道0的个数小于k为止

代码

```cpp
int longestOnes(vector<int>& nums, int k) {
    int left = 0;
    int zeroCount = 0;
    int maxLength = 0;

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

        while (zeroCount > k) {
            if (nums[left] == 0) {
                --zeroCount;
            }
            ++left;
        }

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

    return maxLength;
}
```

本日遇到的问题

1. 部分dp问题有些遗忘

明日学习内容

1. 开始复习算法：dp，kmp字符串