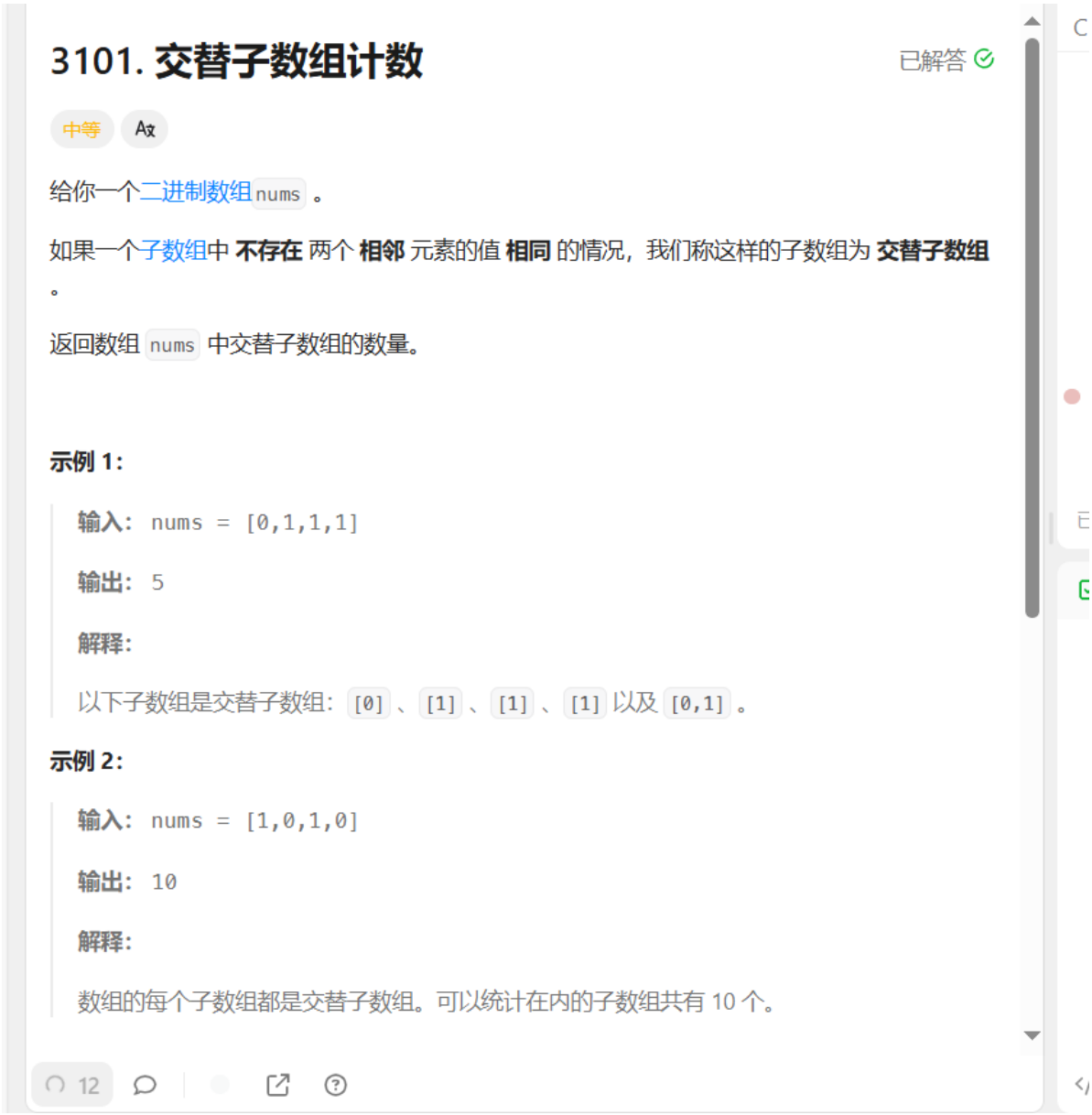
力扣+交替子数组计数+7.6

题目：



3101. 交替子数组计数

已解答

中等

给你一个二进制数组 `nums` 。

如果一个子数组中 **不存在** 两个 **相邻** 元素的值 **相同** 的情况，我们称这样的子数组为 **交替子数组** 。

返回数组 `nums` 中交替子数组的数量。

示例 1：

> **输入：** nums = [0,1,1,1]
>
> **输出：** 5
>
> **解释：**
>
> 以下子数组是交替子数组：`[0]` 、`[1]` 、`[1]` 、`[1]` 以及 `[0,1]` 。

示例 2：

> **输入：** nums = [1,0,1,0]
>
> **输出：** 10
>
> **解释：**
>
> 数组的每个子数组都是交替子数组。可以统计在内的子数组共有 10 个。

12

解题思路一：

双指针计数法，把数据拆分，我们发现每次如果前后数据是相同的情况下，当前数组的解一定是上一个解+1，如果我们每次遇到相同的数字就置为1，不同的数字就++，那么我们发现当前数组的值恰好就是前一个值加上cur，由于只需要最后一个，不必纠结每个值，那么我们使用双指针计数即可，一个pre记录当前指针前一个，一个cur记录当前指针的值，对于解，我们需要初始化解的值为1

```
class Solution {
public:
    long long countAlternatingSubarrays(vector<int>& nums) {
        long long res = 0, cur = 0;
        int pre = -1;
        //每次遇到相同数字重新计数
```

```
        for (int a : nums) {
            cur = (pre != a) ? cur + 1 : 1;
            pre = a;
            res += cur;
        }
        return res;
    }
};
```

解题思路二：

纯数学方法，把数字相同的元素之间设定一堵墙，形如0 1 0 0 1的数组就被墙隔为了0 1 0和01，我们只要分别计算两个数组的子数组个数即可，对于每个不重复的子数组我们发现，其和一定是当前数组的不重复子数组个数，那么当数组长度为n时，无墙的子数组的个数为(n*(n+1))/2

```
class Solution {
public:
    long long countAlternatingSubarrays(vector<int>& nums) {
        long long noWallArea = 1, sumSubSet = 0;
        for (int i = 0; i < nums.size() - 1; i++) {
            if (nums[i] ^ nums[i + 1])
                noWallArea++;
            else {
                sumSubSet += noWallArea * (noWallArea + 1) / 2;
                noWallArea = 1;
            }
        }
        //处理最后一段无墙区域
        sumSubSet += noWallArea * (noWallArea + 1) / 2;
        return sumSubSet;
    }
};
```