250714删除有序数组中的重复项 II（题解）

> 题目链接:
> https://leetcode.cn/problems/remove-duplicates-from-sorted-array-ii/description/

给你一个有序数组 `nums` ，请你原地删除重复出现的元素，使得出现次数超过两次的元素只出现两次，返回删除后数组的新长度。
不要使用额外的数组空间，你必须在 原地修改输入数组并在使用 `O(1)` 额外空间的条件下完成。

说明:
为什么返回数值是整数，但输出的答案是数组呢?
请注意，输入数组是以「引用」方式传递的，这意味着在函数里修改输入数组对于调用者是可见的。
你可以想象内部操作如下:

```
//nums 是以“引用”方式传递的。也就是说，不对实参做任何拷贝
int len = removeDuplicates(nums);
// 在函数里修改输入数组对于调用者是可见的。
// 根据你的函数返回的长度, 它会打印出数组中 **该长度范围内** 的所有元素。
for (int i = 0; i < len; i++) {
    print(nums[i]);
}
```

示例 1:
输入: nums = [1,1,1,2,2,3]
输出: 5, nums = [1,1,2,2,3]
解释: 函数应返回新长度 length = 5, 并且原数组的前五个元素被修改为 1, 1, 2, 2, 3。 不需要考虑数组中超出新长度后面的元素。

解法一：快慢指针

```cpp
class Solution {
public:
    int removeDuplicates(vector<int>& nums) {
        if(nums.size()<=2)  return nums.size();
        int s=2,f=2;
        while(f<nums.size()){
            if(nums[s-2]!=nums[f])  nums[s++]=nums[f];
            f++;
        }
        return s;
    }
};
```

使用**双指针**，`s` 是慢指针，`f` 是快指针，保证最多保留两个重复元素。

```cpp
if(nums.size()<=2)  return nums.size();
```

- 若数组长度不超过 2，最多只可能出现两个重复元素，因此直接返回即可。

```
int s = 2, f = 2;
```

- 指针初始化：前两个元素可以保留，所以 `s` 从 2 开始写入位置，`f` 从 2 开始检查。

```
while(f < nums.size()){
    if(nums[s-2] != nums[f])
        nums[s++] = nums[f];
    f++;
}
```

```
if(nums[s-2] != nums[f])
```

- 当前要写入的元素 `nums[f]` 和写入位置前两个元素中的最前一个 `nums[s-2]` 比较。
- 如果 **不等于**，说明没有出现超过 2 次，可以写入。

演示：

输入：`nums = [1,1,1,2,2,3]`

```
初始：
s = 2, f = 2
nums[0] = 1
nums[1] = 1

f = 2 → nums[0] == nums[2] → 跳过
f = 3 → nums[1] != nums[3] → nums[2] = 2, s = 3
f = 4 → nums[2] != nums[4] → nums[3] = 2, s = 4
f = 5 → nums[3] != nums[5] → nums[4] = 3, s = 5
```

返回结果：`s = 5`，数组前 5 项为 `[1, 1, 2, 2, 3]`