# 题目

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。

示例 2：

输入：nums = [0,0,1,1,1,1,2,3,3]

输出：7, nums = [0,0,1,1,2,3,3]

解释：函数应返回新长度 length = 7, 并且原数组的前七个元素被修改为 0, 0, 1, 1, 2, 3, 3。不需要考虑数组中超出新长度后面的元素。

提示：

1 <= nums.length <= 3 \* 104

-104 <= nums[i] <= 104

nums 已按升序排列

类似题目：

26. 删除有序数组中的重复项

# 分析

## 方法一：双指针

class Solution {

public:

int removeDuplicates(vector<int>& nums) {

int n = nums.size();

if (n <= 2) return n;

// 因为这里要求重复的只能出现2次，这里从2开始

int slow = 2, fast = 2;

while (fast < n) {

if (nums[slow - 2] != nums[fast]) {

nums[slow] = nums[fast];

slow++;

}

fast++;

}

return slow;

}

};

复杂度分析

时间复杂度：O(n)，其中n是数组的长度。我们最多遍历该数组一次。

空间复杂度：O(1)。我们只需要常数的空间存储若干变量。