**难度: 简单**

给定一个排序数组，你需要在 原地 删除重复出现的元素，使得每个元素只出现一次，返回移除后数组的新长度。

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。

?

示例?1:

给定数组 nums = [1,1,2],

函数应该返回新的长度 2, 并且原数组 nums 的前两个元素被修改为 1, 2。

你不需要考虑数组中超出新长度后面的元素。

示例?2:

给定 nums = [0,0,1,1,1,2,2,3,3,4],

函数应该返回新的长度 5, 并且原数组 nums 的前五个元素被修改为 0, 1, 2, 3, 4。

你不需要考虑数组中超出新长度后面的元素。

?

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]);

}

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/remove-duplicates-from-sorted-array

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路(双指针):**

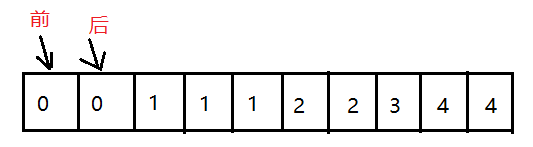
1). 判断数组的长度是否为0,为0直接返回0

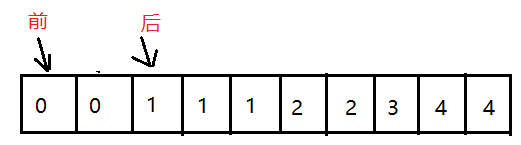
2). 定义一个前指针,一个后指针

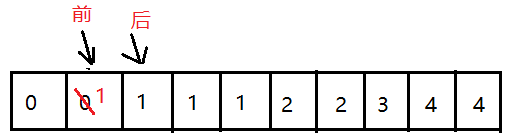
3). 每次用后指针去和前指针比较,是否相等,不想等的话,就前指针加1,然后把不相等的值给到前指针+1的值

4). 持续循环直到全部结束.然后返回前指针下标+1

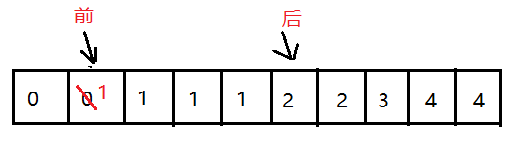
图解: *nums* = **[0,0,1,1,1,2,2,3,3,4]**

 ← 此时后指针和前指针相比 0 和 0 相等,条件不成立,所以后指针+1 ↓

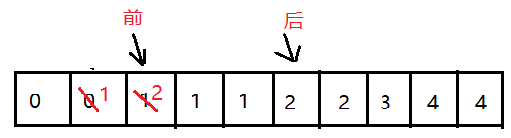


再 1 和 0 比较,它们两个不相等,条件成立,于是前指针1,然后进行替换 ↓

此时再持续循环比较,后指针后面有三个1和和前指针相同,于是后指针的下标一直+1直到 ↓



此时后指针和前指针比较 2 和 1

于是前指针+1,然后替换 →

**大致就是这个步骤了**

|  |
| --- |
| public int removeDuplicates(int[] nums) {  if (nums.length == 0) {  return 0;  }  int i = 0;  for (int j = 1; j < nums.length; j++) {  if (nums[j] != nums[i]) {  i++;  nums[i] = nums[j];  }  }  return i+1;  } |