# 题目

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]);

}

类似题目：LeetCode 27（二者不太一样）

# 分析

## 方法一：双指针

思路：

代码：

class Solution {

public:

int removeDuplicates(vector<int>& nums) {

if (nums.empty()) return 0; // 处理空数组的情况

int i = 0; // 定义慢指针 i

for (int j = 1; j < nums.size(); ++j) { // 定义快指针 j，从第二个元素开始遍历

if (nums[i] != nums[j]) { // 如果快慢指针所指元素不相同

nums[++i] = nums[j]; // 将快指针指向的元素赋值给慢指针的下一个位置，并移动慢指针

}

}

return i + 1; // 返回不重复元素的个数（加上第一个元素nums[0]）

}

};

典型错误：

if (nums[i] != nums[j]) { // 如果快慢指针所指元素不相同

nums[i] = nums[j]; // 将快指针指向的元素赋值给慢指针的下一个位置，并移动慢指针

i++;

}

在这段代码中，i 表示慢指针，j 表示快指针，用于遍历数组 nums。在每次比较 nums[i] 和 nums[j] 时，如果它们不相等，你正确地将 nums[j] 的值赋给 nums[i] 的下一个位置，并将 i 向后移动一位。但是，你应该将 i 初始化为 1，而不是 0，因为在比较 nums[i] 和 nums[j] 之前，你不应该覆盖 nums[0] 的值，因为它是第一个元素，不需要删除。因此，应该将 i 初始化为 1，并在每次赋值后将 i 递增，以确保 nums[0:i] 中的元素是唯一的。

或：

你可以使用双指针的方法来解决这个问题。一个指针用于遍历数组，另一个指针用于指示下一个不重复元素应该存放的位置。具体步骤如下：

1、初始化一个变量k为1，表示唯一元素的个数。

2、从第二个元素开始遍历数组，如果当前元素与前一个元素不相等，则将当前元素复制到nums[k]的位置，并将k自增 1。

3、最后返回k，即为唯一元素的个数。

这个算法的时间复杂度是O(n)，其中n是数组的长度。

下面是实现这个算法的 C++ 代码：

class Solution {

public:

int removeDuplicates(vector<int>& nums) {

if (nums.empty()) return 0; // 处理空数组的情况

int k = 1; // 初始唯一元素个数为 1

for (int i = 1; i < nums.size(); ++i) {

if (nums[i] != nums[i - 1]) {

nums[k++] = nums[i];

}

}

return k;

}

};

或：

1、使用快慢双指针法分别定义指针i与指针j，开始两指针都指向num[0],

2、快指针i向前移动指向的数组与慢指针j所知数组比较num[j]!=num[i]

3、若相同则快指针向前移动一格，慢指针向前移动一格；若不同，将快指针i指向数组赋值给慢指针j，快慢指针分别向前移动一格；重复操作直到i指针指到末尾。

4、若数组长度为0则直接return 0；

5、最后，返回慢指针所指数组长度加1，return j+1。

class Solution {

public:

int removeDuplicates(vector<int>& nums) {

int i,j;

int length=nums.size();

if(length==0)//判断数组长度 若为0直接返回

return 0;

for (i=0;i<nums.size();i++)//采用for循环使指针i不停向前移动

{

if(nums[i]!=nums[j])

{

if(i-j>1)

//判断快慢指针：若相邻数组且元素值不同时，虽然也可进行赋值操作，

//但可以省略该赋值操作，避免无意义操作提高运行效率

{

nums[j+1]=nums[i];

}

j++;

}

}

return j+1;

}

};