拉勾教育
—互联网人实战大学—

删除排序数组的重复项

中等/双指针

学习目标

拉勾教育
—互联网人实战大学—

了解算法题的解题思路

双指针的特点

双指针的应用

题目描述

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。

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。

给定数组 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。

题目描述

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]);
}
```

一. Comprehend 理解题意

1. 题目主干要求

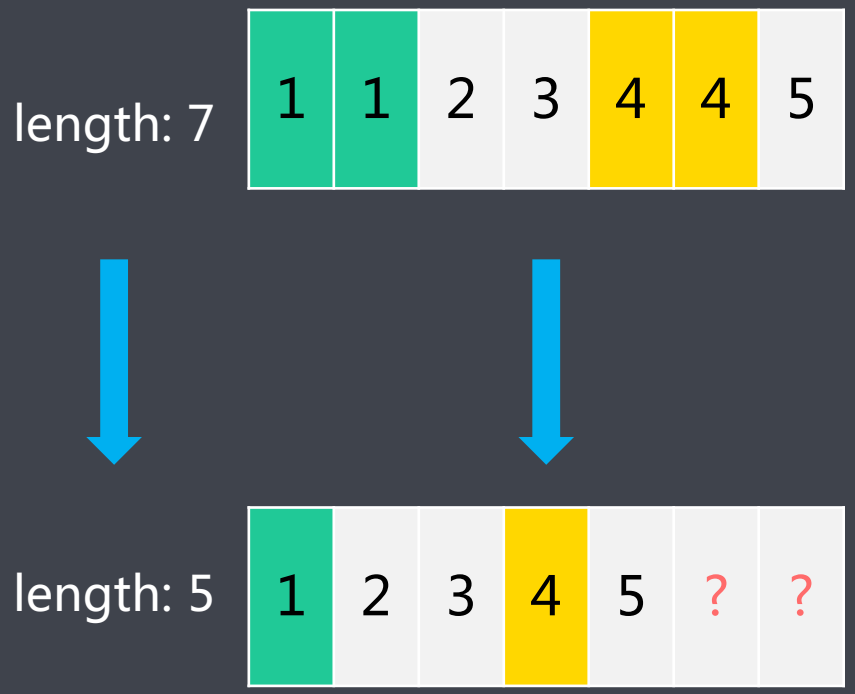
   删除重复出现的元素，使每个元素只出现一次

   返回移除后数组的新长度

2. 附加限制要求

   原地删除重复出现的元素

   不使用额外数组空间，额外空间复杂度O(1)



一. Comprehend 理解题意

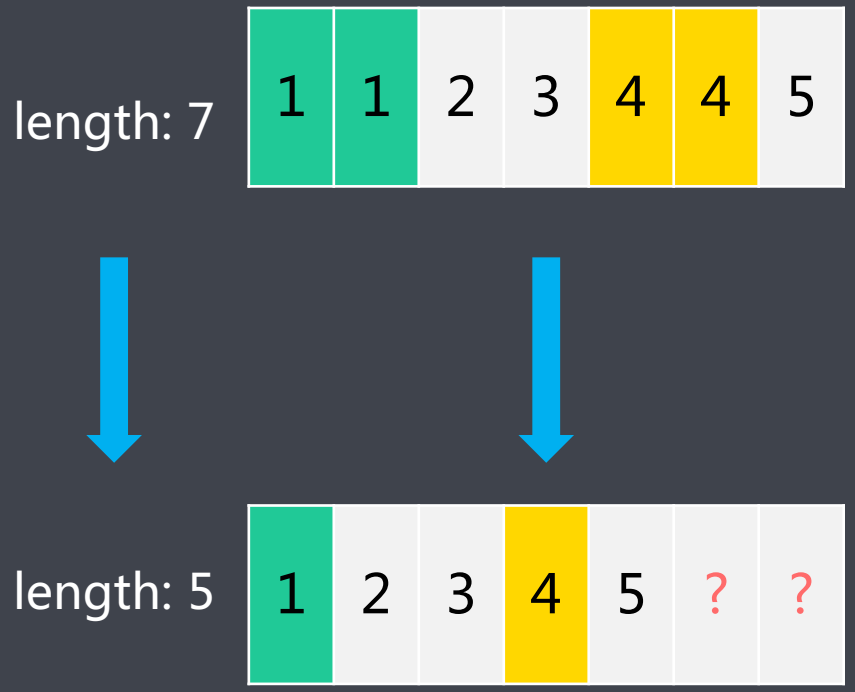
拉勾教育
—互联网人实战大学—

3. 宽松条件

给定一个排序数组

重复元素都是相邻的

不需要考虑数组中超出新长度后的元素



4. 其它细节

给定数组可能为空的情况

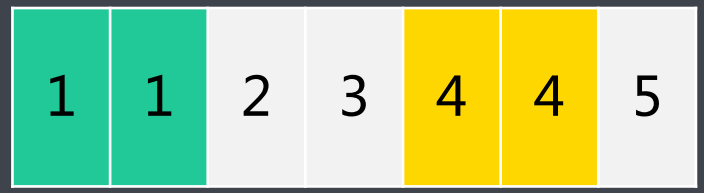
数组下标问题，在编写代码时要注意边界问题

元素重复次数没有限定，可能是0到n-1次（n为数组长度）

二. Choose 数据结构及算法思维选择

数据结构选择

- 题目限定了额外O(1)的空间
- 给定的数组



算法思维选择

- 遍历
- 比较

三. Code 基本解法及编码实现

解法一：暴力解法思路分析

1. 遍历数组
2. 依次比较相邻的元素（i和i+1）

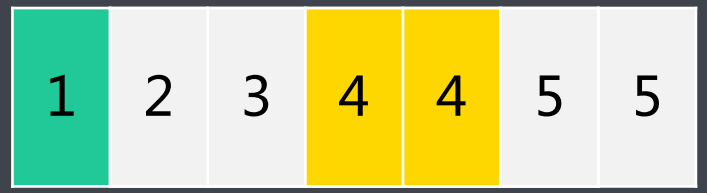
   不同：向后遍历，i++；

   相同：后面所有元素前移一位；
3. 每遇到重复元素，数组长度缩减1

三. Code 基本解法及编码实现

解法一: 暴力解法边界和细节问题

边界问题



- 数组索引越界
- 循环退出条件：遍历到length – 1时

细节问题

每次处理重复数据后，需要 i 与 i+1 再比较一次，避免掉落元素

相邻元素不相等时，索引才能继续指向下个元素

三. Code 基本解法及编码实现

拉勾教育
—互联网人实战大学—

```java
public int removeDuplicates(int[] nums) {
    int length = nums.length; // 数组长度
    // 1.遍历数组
    for (int i = 0; i < length - 1; ) {
        // 2.依次比较相邻的元素
        if (nums[i] != nums[i + 1]) {
            // 2.1.不同：向后遍历
            i++;
        } else {
            // 2.2.相同：后面所有元素前移一位
            for (int j = i + 1; j < length - 1; j++) {
                nums[j] = nums[j + 1];
            }
            // 3.每遇到重复元素，数组长度缩减1
            length--;
        }
    }
    return length;
}
```

时间复杂度：O(n^2)

- 循环遍历进行比较：O(n)
- 遇到重复元素，前移后续元素：O(n)
- 上述操作是嵌套的：O(n)*O(n)=O(n^2)

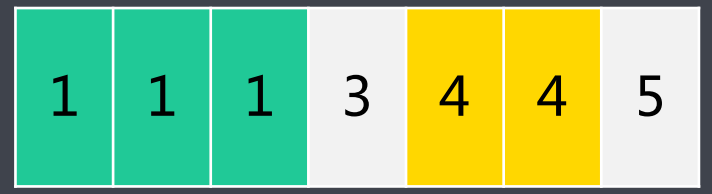
空间复杂度：O(1)

- 需要一个临时变量：O(1)

执行耗时:177 ms,击败了5.06% 的Java用户
内存消耗:40.4 MB,击败了88.26% 的Java用户

四. Consider 思考更优解

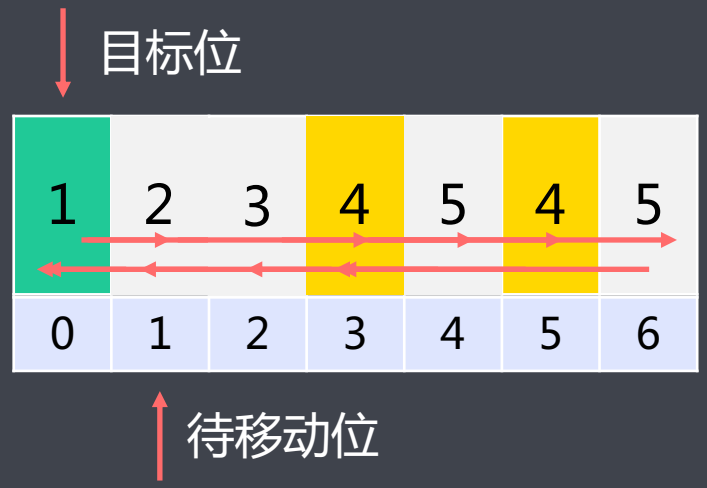
1. 剔除无效代码或优化空间消耗
   - 前移后续所有元素的操作是必须的吗？
2. 寻找更好的算法思维
   - 待移动元素能否一次性移动到目标位？
   - 双指针：分别记录待移动元素和目标位置的索引
   - 参考其它算法



五. Code 最优解思路及编码实现

最优解：双指针解法

1. 定义两个指针（索引）分别指向目标位和待移动元素位
   - 初始目标位为0，待移动元素位为1
2. 比较两个指针对应的数据
   - 比较相等：目标位不变，待移动位加1
   - 比较不等：目标位加1，待移动元素赋值到目标位；待移动位加1



五. Code 最优解思路及编码实现

拉勾教育
—互联网人实战大学—

最优解：双指针解法边界和细节问题

边界问题

- 只需要一层循环，循环退出条件：待移动指针超出数组范围

细节问题

- 题目要求返回数组新长度，即目标位对应的数据是最后一个元素，下标+1
- 尝试移动元素前，若目标位与待移动位相同，跳过赋值操作

五. Code 最优解思路及编码实现

拉勾教育
—互联网人实战大学—

```java
public int removeDuplicates(int[] nums) {
    // 1.定义两个指针（索引）分别指向目标位和待移动元素位
    int target = 0; // 目标位指针
    for (int i = 1; i < nums.length; i++) { // i为待移动位指针
        // 2.比较两个指针对应的数据
        if (nums[target] != nums[i]) {
            // 不相等，则把目标位置后移
            if (++target != i) {
                nums[target] = nums[i];
            }
        }
    }
    return target + 1;
}
```

时间复杂度：O(n)

- 只需要一层循环遍历：O(n)

空间复杂度：O(1)

- 需要有限个常量空间：O(1)

执行耗时:1 ms,击败了97.67% 的Java用户
内存消耗:39.9 MB,击败了99.29% 的Java用户

六. Change 变形延伸

拉勾教育
—互联网人实战大学—

延伸扩展

- 什么情况下使用双指针：两个按照一定规律同步变化的数据；
- 合理的使用双指针能将时间复杂度从O(n^2)降低到O(n)级别

本题来源：

- Leetcode 26 https://leetcode-cn.com/problems/remove-duplicates-from-sorted-array/

课后练习

拉勾教育
—互联网人实战大学—

1. 删除排序链表中的重复元素(Leetcode 83/简单)
2. 删除排序链表中的重复元素 II(Leetcode 82/中等)
3. 删除排序数组中的重复项 II(Leetcode 80/中等)
4. 原地移除全部给定数组中的指定值，返回数组新长度(Leetcode 27/简单)

拉勾教育

—互联网人实战大学—

下载「拉勾教育App」
获取更多内容