ARTS 008

作者：lucianolixin
原文链接：https://ld246.com/article/1571539392689
来源网站：链滴

ARTS 是由左耳朵耗子陈皓在极客时间专栏《左耳听风》中发起的一个每周学习打卡计划。

Algorithm：至少做一个 LeetCode 的算法题。主要为了编程训练和学习。

Review ： 阅读并点评至少一篇英文技术文章。主要为了学习英文，如果你英文不行，很难成为技术手。

Tip：学习至少一个技术技巧。主要是为了总结和归纳你日常工作中所遇到的知识点。

Share：分享一篇有观点和思考的技术文章。主要为了输出你的影响力，能够输出你的价值观。

Algorithm

LeetCode26. 删除排序数组中的重复项

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。

暴力法

- 思路

1. 比较当前元素和前一个元素是否相等，如果相等则后面向前都挪动一个坐标。

- 1层循环：得到当前需要比较元素

- 2层循环：在当前下标位置删除元素的个数

- 3层循环：真正移动当前以及后续元素

2. 此方法，思路简单，但是编写程序还是不容易的。并且时间复杂度比较高,三层循环 O（n 重复元个数(最新数组长度-当前下标)）

● 需要注意的点

● 应该用当前值和前一个元素进行比较，这样不用考虑数组越界

● 第二层循环的终止条件有两个，1）前后元素不相等 2）数组的长度已经减小到了i（第一层循环遍的位置

```
 func RemoveDuplicates1(nums []int) int {
        l := len(nums)
    for i := 1; i < l; i++ {
        for {
            if nums[i] != nums[i-1] || l <= i {
                break
            }
            for j := i; j < l; j++ {
                nums[j-1] = nums[j]
            }
            l--
        }
    }
    return l
}
```

双指针法

● 思路

1. 指针1维护需要比较的元素

2. 指针2维护已经比较的元素

3. 最后返回指针2

● 分析

● 思路1的重心是操作未比较的元素

● 思路2的重心是维护比较的元素

● 对于此题，显然已经完成比较的元素更可控，不需要移动元素，减少复杂度。

```
func RemoveDuplicates2(nums []int) int {
        p2 := 0
    for p1 := 1; p1 < len(nums); p1++ {
        if nums[p1] != nums[p2] {
            p2++
            nums[p2] = nums[p1]
        }
```

```
    }
    return p2 + 1
}
```

Review

简单Review了java 12的数组，连表，队列，优先队列的实现

Tip

解题思路

1. 暴力法

- 任何一道题，理解题目后，应该直接思考其暴力法，给出时空复杂度。由于暴力法不考虑任何复比较情况，时间复杂度会较高
- 分析出重复比较的部分。分析出重复的部分，才能一步步优化

2. 空间换时间

- 缓存重复比较的结果
- 使用Hash缓存之前浏览过的节点
- 对于在原数组上操作的情况，可以开辟新数组进行操作，规避操作可能会覆盖元素的问题。
- 使用合适的数据结构，可以大大简化问题
  - 栈：解决最近相似性问题，在做题过程中发现有些问题可以被抽象成相似性问题
  - HashMap：可以缓存比较结果，不必重复操作比较。

3. 双指针

- 可以有效的减少比较次数而不会漏掉任何一种情况，双指针是颇有技巧性的
- 左右指针往中间夹逼
- 从节点i往两边分散
- 快慢指针

4. 寻找重复子问题

- 将一个大问题，分解为数个小问题。逐步解决

5. 设置哨兵简化边界条件

在做题过程中，都应该按照上面的思路去思考一遍。

Share

github fork pull request 工作模式

1. fork 仓库到自己的远程仓库

2. 配置ssh身份验证

3. git clone 代码到本地

4. 设置上游upstream源仓库，git add remote upstream 地址

5. commit 、push到远程分支

6. 提交pull request到源仓库，等待代码审核

注意点:

● .gitignore只能忽略那些原来没有被track的文件，如果某些文件已经被纳入了版本管理中，则修改.gitignore是无效的。

解决方法就是先把本地缓存删除（改变成未track状态），然后再提交:

```
git rm -r --cached 文件名

git add 文件名

git commit -m```'update .gitignore'
```

● 前一个pull request没有merge 后面的没办法创建

● 要经常同步源仓库，必须设置upstream，用git fetch upstream