CodeExercise：

简介：

记录从大一暑假开始的刷题之旅。

PS：这个代码库保存了我从大一入学所刷的所有算法题，只是从2022年6月21日开始整理。

日常刷题：

2022.6.21——数组

周二——雷阵雨转阴——35℃/22℃

今天开始刷LeetCode中的《初涉算法》篇，先从数组开始：

26. 删除有序数组中的重复项

类型： 数组 双指针

```
class Solution {
public:
    int removeDuplicates(vector<int>& nums) {
        int fast = 0,slow = 0;
```

```
        while(fast<nums.size()) {
            if(nums[fast++]!=nums[slow]) {
                nums[++slow] = nums[fast-1];
            }
        }
        return slow+1;
    }
};
```

时间复杂度： O(N) 空间复杂度： O(1)

使用了**fast**和**slow**两个快慢指针进行判断，每次循环快指针都会向前走一位，通过快指针指向的位置和慢指针比较，若和慢指针所指向的数不同则慢指针走一位并将快指针所对应的值赋给慢指针指向的位置。

最后慢指针所指向的位置则为数组长度-1，所以返回**slow+1**即可。

详情参考：双指针技巧秒杀七道数组题目 :: labuladong的算法小抄

由上一题可延伸：

83. 删除排序链表中的重复元素

类型： 链表

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        if(head==nullptr) return nullptr;
        ListNode *slow = head,*fast = head;
        while(fast->next!=nullptr) {
            fast = fast->next;
            if(fast->val!=slow->val) {
                slow = slow->next;
                slow->val = fast->val;
            }
        }
        slow->next = nullptr;
        return head;
    }
};
```

时间复杂度： O(N) 空间复杂度： O(1)

同样使用快慢指针，详情见上文。

27. 移除元素

类型： 数组 双指针

```
class Solution {
public:
    int removeElement(vector<int>& nums, int val) {
        int slow = 0,fast = 0;
        while(fast!=nums.size()) {
            if(nums[fast++]!=val) {
                nums[slow++] = nums[fast-1];
            }
        }
        return slow;
    }
};
```

时间复杂度： O(N) 空间复杂度： O(1)

同样利用双指针，将重复的元素覆盖，最后返回数组长度为slow的值。

283. 移动零

类型： 数组 双指针

```
class Solution {
public:
    void moveZeroes(vector<int>& nums) {
        int slow = 0,fast = 0;
        while(fast!=nums.size()) {
            if(nums[fast++]!=0) {
                nums[slow++] = nums[fast-1];
            }
        }
        while(slow!=nums.size()) {
            nums[slow++] = 0;
        }
    }
};
```

时间复杂度： O(N) 空间复杂度： O(1)

利用上一题的做法最后在将**slow**指针往后的多余数替换为0。

补充内容：