双指针：删除排序数组中的重复项

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

暴力解法：重复后元素整体前移

Java代码

```
/**
 * 解法一：暴力解法
 * 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
 *
 * 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
 * @param nums
 * @return
 */
public int removeDuplicates(int[] nums) { // 1,1,2,3,4,4,5
    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;
    }
```

最优解：双指针解法

java代码

```
    /**
     * 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
     *
     * 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
     * @param nums
     * @return
     */
    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待移动位指针
            // 2.比较两个指针对应的数据
            if (nums[target] != nums[i]) {
                if (++target != i) {
                    nums[target] = nums[i];
                }
            }
        }
        return target + 1;
    }
```

C++代码

```cpp
/**
 * 执行用时：12 ms, 在所有 C++ 提交中击败了99.03%的用户
 * 内存消耗：13.8 MB, 在所有 C++ 提交中击败了5.12%的用户
 */
class Solution {
public:
    int removeDuplicates(vector<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
        int size = nums.size();
        for (int i = 1; i < size; i++) { // i待移动位指针
            // 2.比较两个指针对应的数据
            if (nums[target] != nums[i]) {
                if (++target != i) {
                    nums[target] = nums[i];
                }
            }
        }
        return target + 1;
    }
};
```

Python代码

```python
    '''
    执行用时：36 ms, 在所有 Python3 提交中击败了98.10%的用户
    内存消耗：14.4 MB, 在所有 Python3 提交中击败了73.19%的用户
    '''
    def removeDuplicates(self, nums: List[int]) -> int:
        if len(nums) == 0:
            return 0
        # 1.定义两个指针（索引）分别指向目标位和待移动元素位
        target = 0
        for i in range(1, len(nums)):  # i待移动位指针
            # 2.比较两个指针对应的数据
            if nums[target] != nums[i]:
                target += 1
                if target != i:
                    nums[target] = nums[i]
        return target + 1
```

测试用例

输入：给定数组 `nums = [1, 1, 2];`
输出：`2`
解释：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。

输入：给定数组 `nums = [0, 0, 1, 1, 1, 2, 2, 3, 3, 4];`
输出：`5`
解释：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。