学习周报1

数组算法：

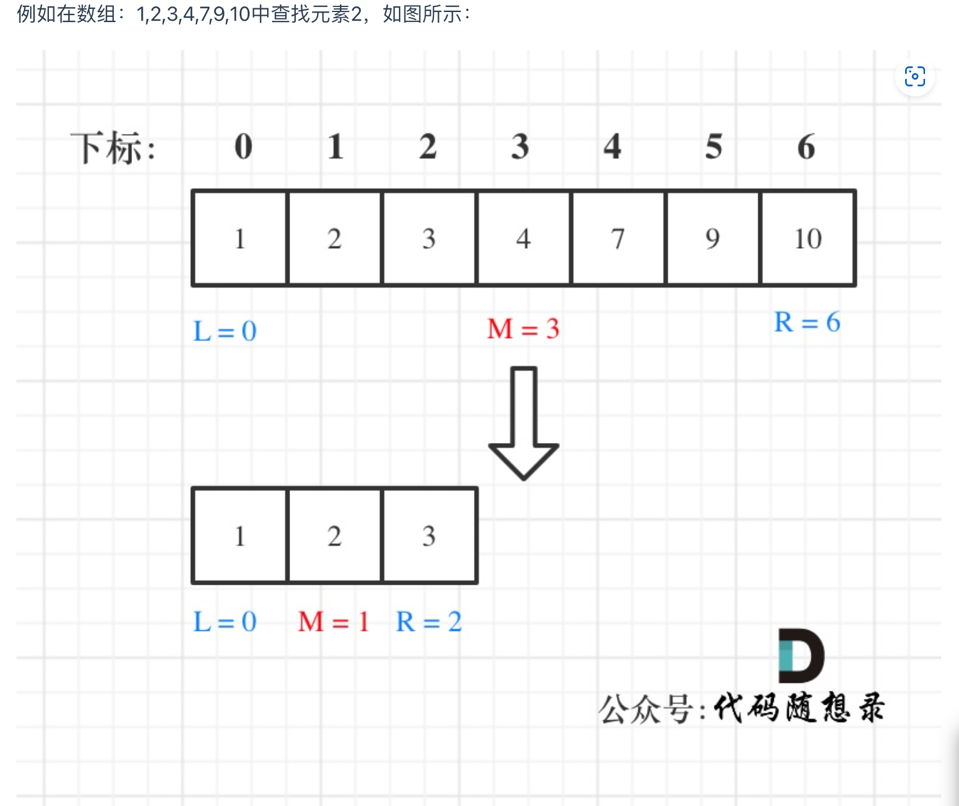
1. 二分查找

问题的关键在于查找的边界，也就是循环的区间该怎么确定。这里一般使用左闭右闭，即类似[3,4]。一般二分法算法的使用前提是数组为有序序列，关于算法的实现借用卡哥的描述：

**第一种写法，我们定义 target 是在一个在左闭右闭的区间里，也就是[left, right] （这个很重要非常重要）。**

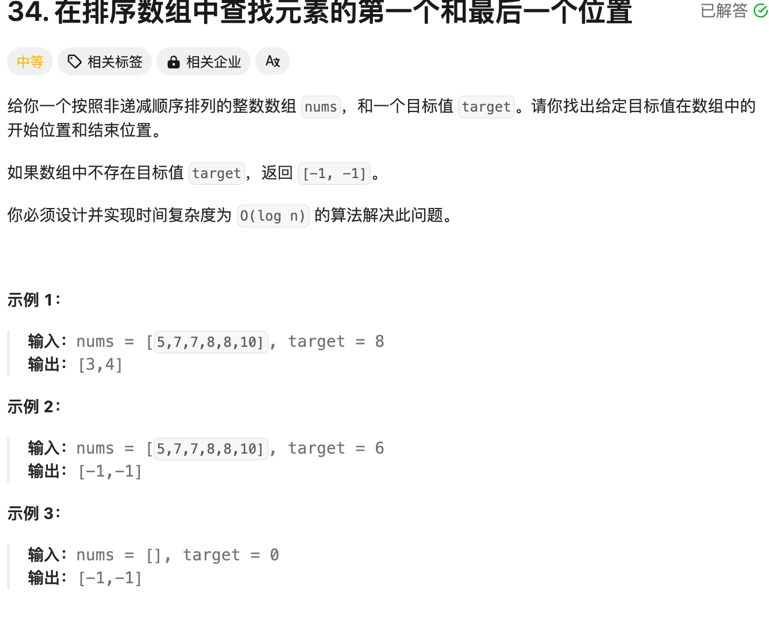
**区间的定义这就决定了二分法的代码应该如何写，因为定义target在[left, right]区间，所以有如下两点：**

* **while (left <= right) 要使用 <= ，因为left == right是有意义的，所以使用 <=**
* **if (nums[middle] > target) right 要赋值为 middle - 1，因为当前这个nums[middle]一定不是target，那么接下来要查找的左区间结束下标位置就是 middle – 1。**



这种算法的时间复杂度是O（logn）

衍生例题：leetcode34



题目要我们返回开始和结束在数组中的位置，运用二分法来解答，这个问题要分开解答，即分别求出第一个该数（左边第一个）以及第二个该数（右边最后一个）。如果只用最简单的二分法显然是不行的，因为我们还需要判断返回的mid左右侧是否还存在相同的数。

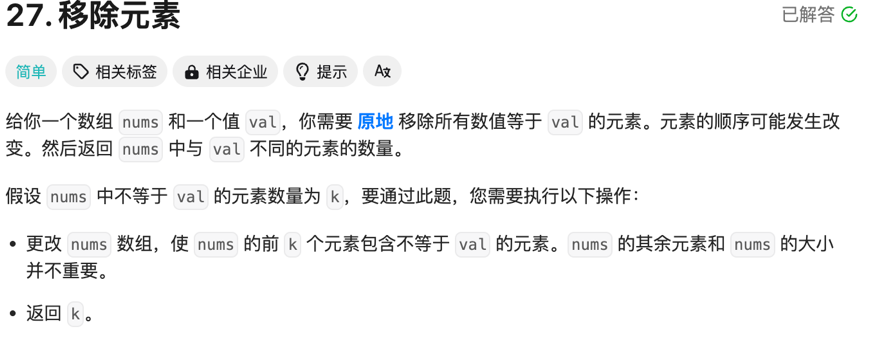
因此我们要加入条件判断，以求leftborder为例：

1. 当数组的mid值等于target，且其左侧与mid不相等，则返回该数
2. 否则，将right改为mid-1再次二分，直到左侧没有想等的值为止。



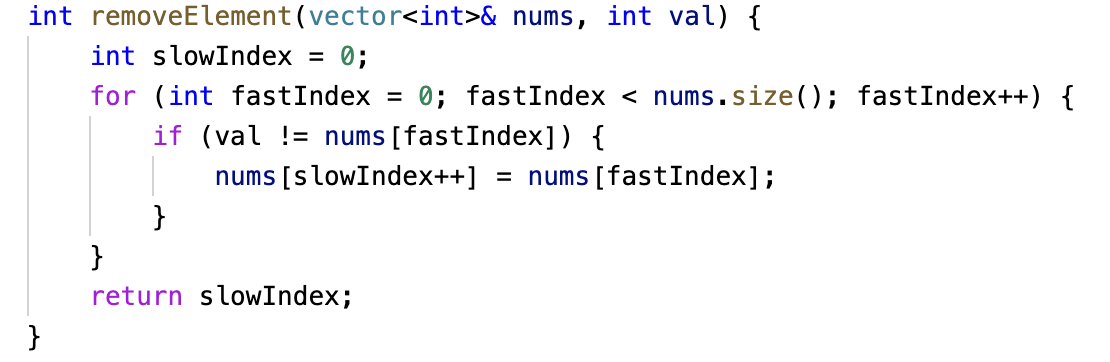
Leetcode27 移除元素：

快慢指针法



详见<https://programmercarl.com/0027.%E7%A7%BB%E9%99%A4%E5%85%83%E7%B4%A0.html#%E6%80%9D%E8%B7%AF>

主要思路就是两个指针一快一慢，快的那个选择出不要的元素并进行交换，而慢的指针选出最后数组需要的元素。注意，这里的删除只需要覆盖即可，所以当fast位于数组最后，直接返回得到的slow前的数组即可。



定义快慢指针

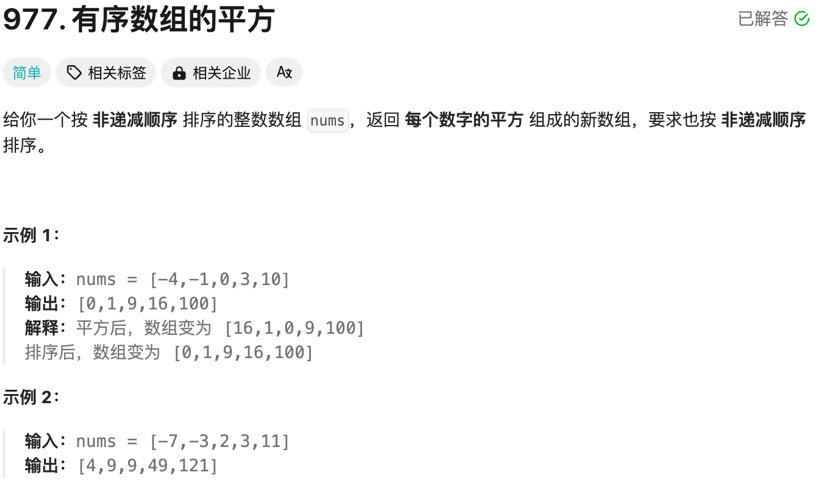
* 快指针：寻找新数组的元素 ，新数组就是不含有目标元素的数组
* 慢指针：指向更新 新数组下标的位置
* 在此题中，快负责找，而找到的直接赋给慢，因此最后慢的所有元素都为有效。

**双指针法（快慢指针法）在数组和链表的操作中是非常常见的，很多考察数组、链表、字符串等操作的面试题，都使用双指针法。事实上，vector的erase操作也是这个逻辑。**

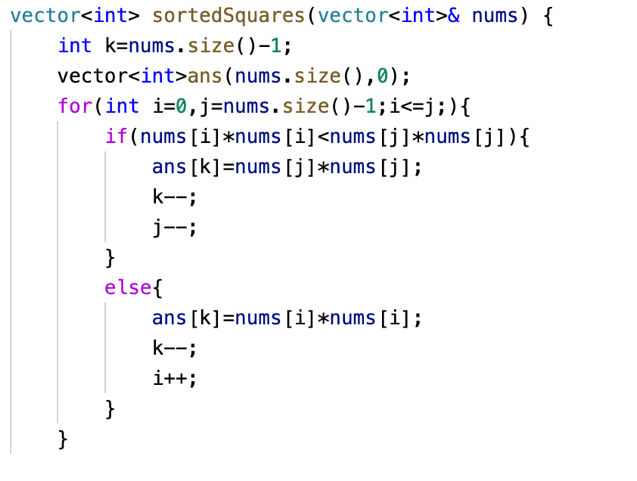
衍生题：删除重复元素

只需要循环内添加条件判断即可，只要后一个元素相等就只令fast++，而slow不变。

Leetcode 977有序数组的平方:



碰到这种需要对比多个值的问题，可以优先考虑多指针。首先由于平方数的最大值只会出现在一头一尾，故将两个指针放在一头一尾。在定义一个新的result数组（大小与原数组一样），并从后往前填充，填充两个指针处的最大值。

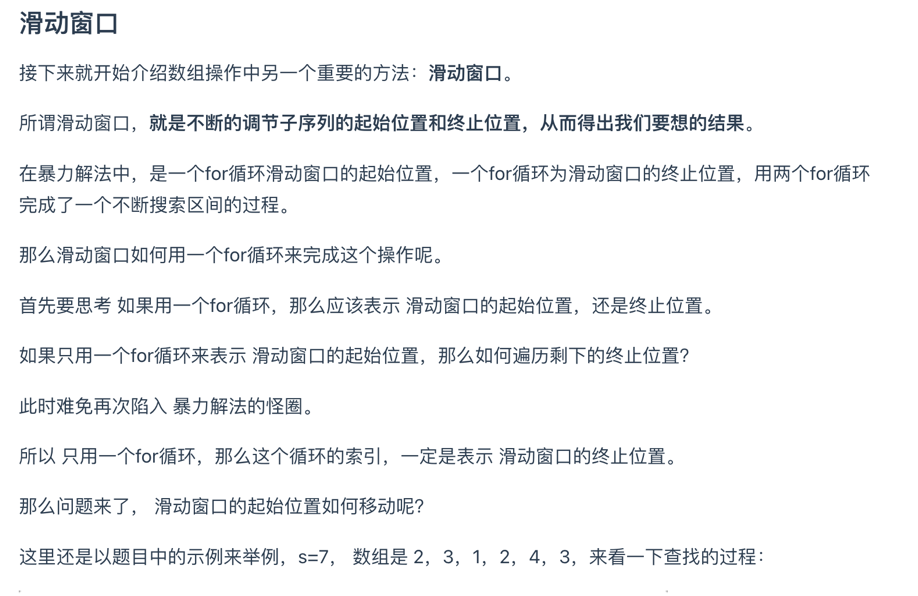


Leetcode 209:长度最小的子数组



使用的是双指针中的滑动窗口。正常的暴力解法下需要n\*n次计算，使用滑动窗口只需要n\*x，x是快指针所在的位置。

这个问题的关键是因为题目的性质，使用双指针进行循环的时候fast是不需要移动的，也就是说框中数组的和不需要从头加，而是可以通过减去或加上单个数来改变。



因此第二层循环的判断条件应该是sum>=target，然后更新子序列的长度最小值。而最核心的在于更新子数组的起始位置，也就是慢指针的位置，直到sum<target时再改变快指针的位置。

