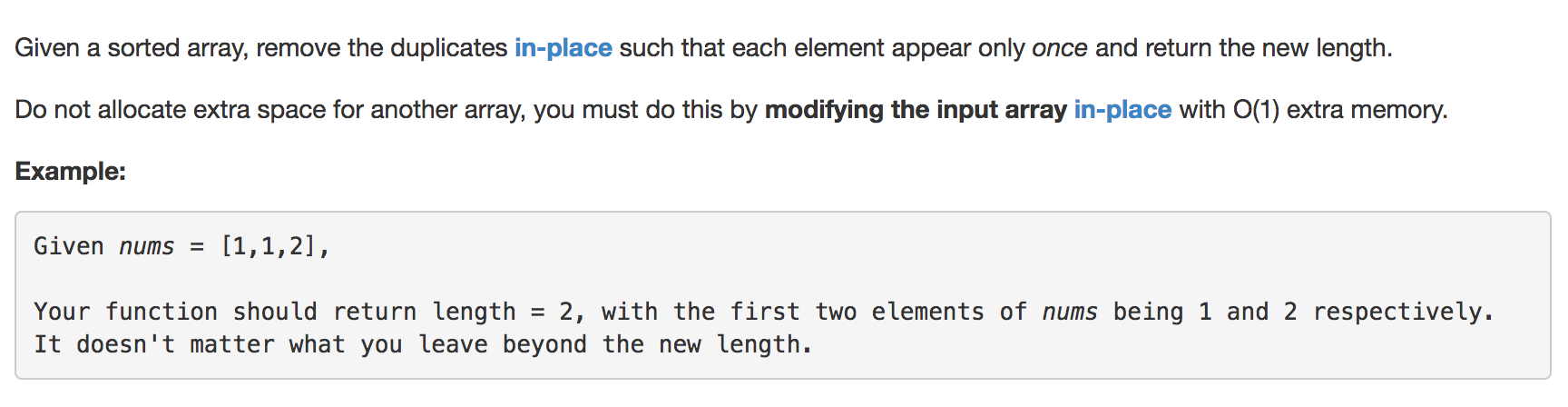
1. 数组
2. Remove Duplicates from sorted array



* 方法1（时间复杂度太大）
* 把数组第1位设为key，从第2位开始和key比较，遇到相同的就remove那个相同的 ————一轮下来跟key相同的都被删掉
* 直到比较到第一个和key不同的，改变key的值为这个不同的值newkey，从newkey的下一位开始和新的newkey比较
* 重复上两步操作至key为最后一个元素
* 注：传来的数组转化为ArrayList，然后使用ArrayList的remove方法进行操作

注：时间复杂度O

* 方法2指导思路（in-place）

设一个变量，只占O(1) extra mermory，用这个变量去记录有多少个不重复的元素，将遇到的不重复的元素都按顺序挪到数组最前边，然后根据记录变量来得到结果数组和结果长度

int index = 0;

for(int i = 0; i < nums.size(); i++){

if(nums[i] != nums[index]){

nums[++index] = nums[i];

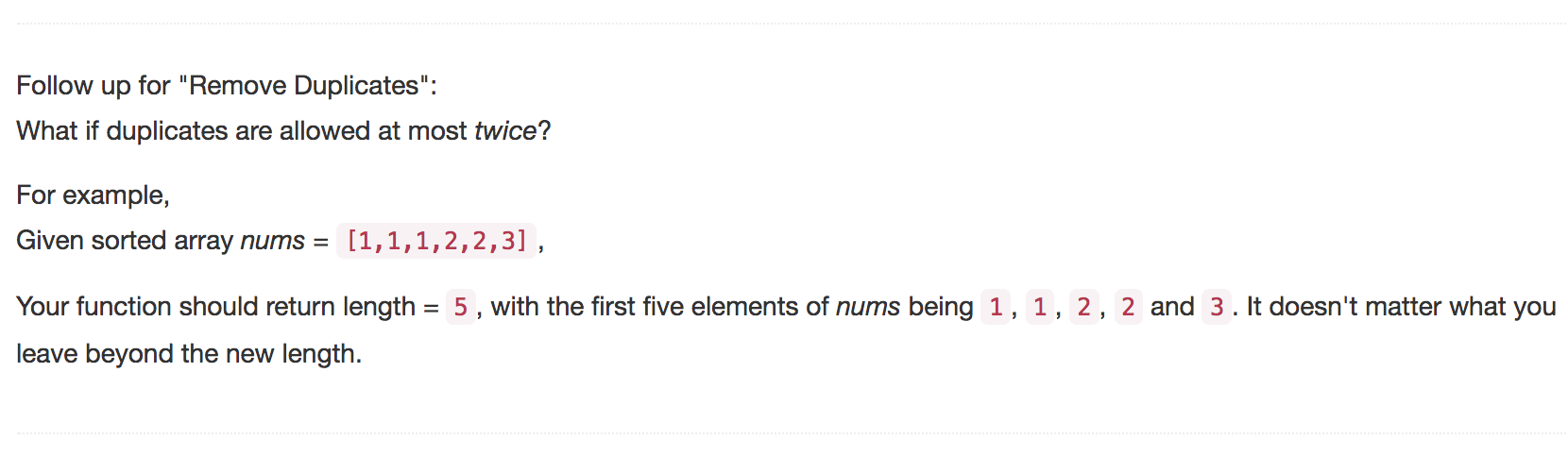
}

}

return index+1;//index代表数组下标，从0开始，+1后代表长度

注：时间复杂度O(n)，空间复杂度O(1)

1. Remove Duplicates from sorted array II



参照上题第2种方法，通过将数组中满足条件的元素移动到数组的最前端，组成结果数组，即将nums拆分为两部分：结果数组和待处理数组。使用index来标记数组的最后一位。针对允许每个元素最多出现两次的情况，使用一个新变量count来统计每个元素出现的次数。

判断条件：

* 数组中的第0个元素一定在结果数组中，所以从数组的第1个元素开始和第0个元素比较，index表示当前结果数组的最后一位元素，剩余待处理部分的第1位和结果数组的index位进行比较。有以下两种情况可以将该比较位加入结果数组中：
* ①待处理位和index位相等并且该元素出现的次数小于两次（将比较位加入结果数组，count++）
* ②待处理位和index位不相等（将比较位加入结果数组，count重置）

int index = 0;

int count = 1;

for(int i = 1; i < nums.size(); i++){

if((nums[index] == nums[i]) && count < 2){

nums[++index] = nums[i];

count++;

}else if(nums[index] != nums[i]){

nums[++index] = nums[i];

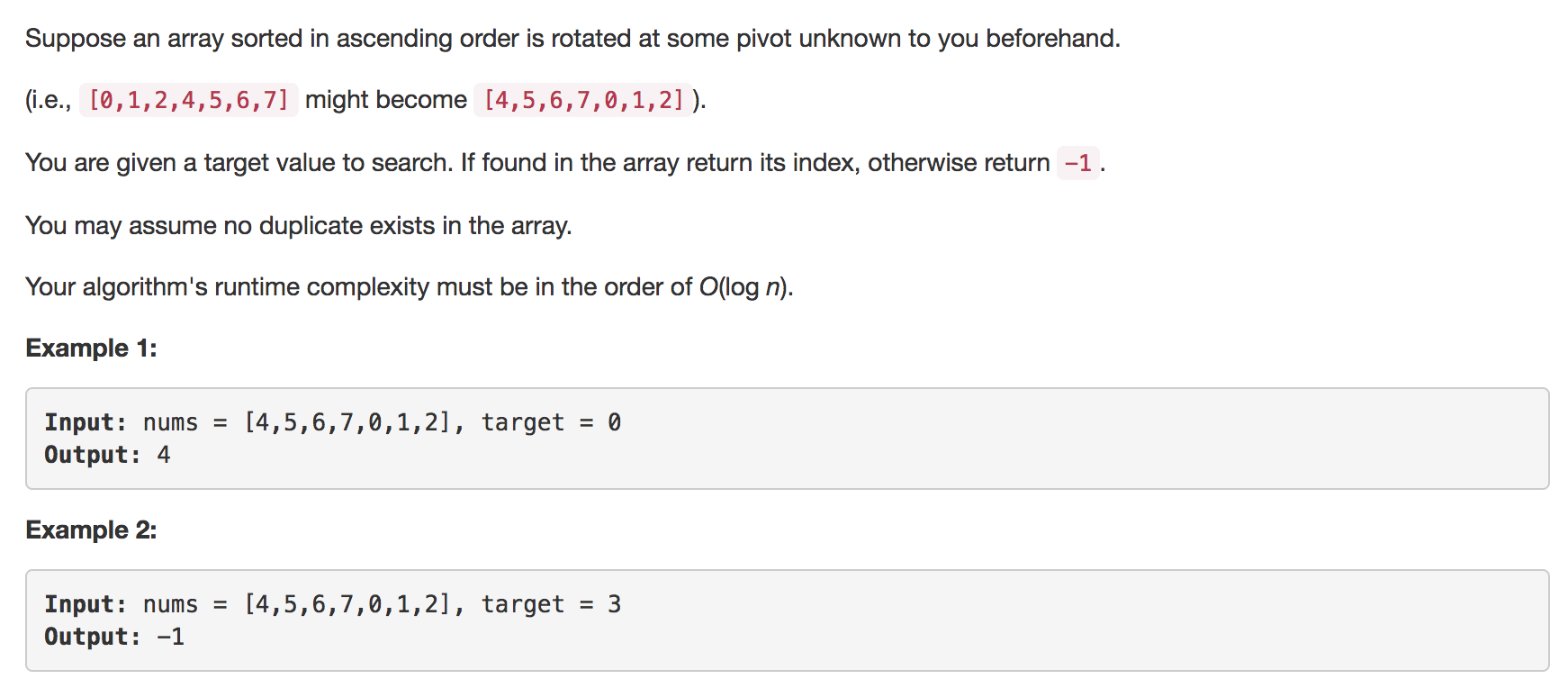
count = 1;

}

}

return 0;

1. Search in Rotated Sorted Array



旋转数组：

0　　1　　2　　 **4　　5　　6　　7**

7　　0　　1　　 **2　　4　　5　　6**

6　　7　　0　　 **1　　2　　4　　5**

5　　6　　7　　 **0　　1　　2　　4**

**4　　5　　6　　7**　　0　　1　　2

**2　　4　　5　　6**　　7　　0　　1

**1　　2　　4　　5**　　6　　7　　0

* 二分搜索法的关键在于获得了中间数后，判断下面要搜索左半段还是右半段。
* 如果中间的数小于最右边的数，则右半段是有序的，若中间数大于最右边数，则左半段是有序的，我们只要在有序的半段里用首尾两个数组来判断目标值是否在这一区域内，这样就可以确定保留哪半边了

while(left <= right){

if(target == nums[mid]){

return mid;

}else if(nums[mid] <nums[right]){

if(target > nums[mid] && target <= nums[right]){

left = mid + 1;

}else{

right = mid – 1;

}

}else{

if(target < nums[mid] && target >= nums[left]){

right = mid – 1;

}else{

left = mid + 1;

}

}

}

1. Search in Rotated Sorted Array II
2. 单链表