算法设计与分析

- LECTURE 6

Outline

- 从蛮力到分治
  - 最大的k个数
  - 数组中重复的数字
  - 数组中出现次数超过一半的数字
  - 搜索旋转排序数组
  - 寻找旋转排序数组中的最小值

最大的k个数

- 给定有 n 个数的集合，现要求找出其中的前k大的k个数。

- 请设计多种基于比较的算法，使其最坏情况时间复杂度分别满足下面的要求：

- O(nlogn)
- O(n + klogn)
- O(n + k^2)
- O(n + klogk)

回顾

- O(nlogn): 排序
- O(n)：选择 select(k)、partition、堆构建
- O(logn): 折半、

数组中重复的数字

题目描述　评论 (1.0k)　题解 (2.9k)　提交记录

剑指 Offer 03. 数组中重复的数字

难度 简单　594　收藏　分享　切换为英文　接收动态　反馈

找出数组中重复的数字。

在一个长度为 n 的数组 nums 里的所有数字都在 0～n-1 的范围内。数组中某些数字是重复的，但不知道有几个数字重复了，也不知道每个数字重复了几次。请找出数组中任意一个重复的数字。

示例 1：

输入：
`[2, 3, 1, 0, 2, 5, 3]`
输出： `2 或 3`

数组中重复的数字

- 方法一： 排序

- 把输入的数组排序，然后从排序的数组中找到重复的数字只需要从头到尾扫描排序后的数组即可。

- 排序一个长度为n的数组需要O(nlogn)的时间复杂度

数组中重复的数字

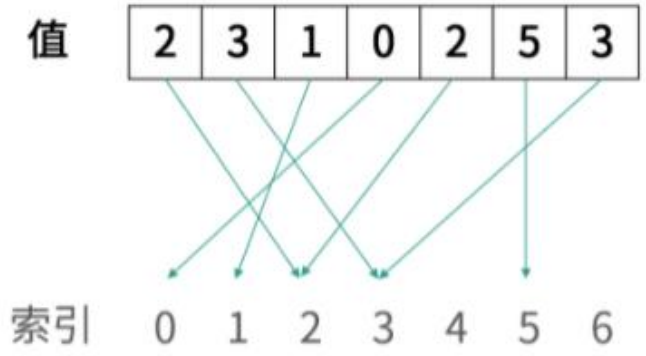
- 方法二：哈希表 / Set

- 利用数据结构特点，容易想到使用哈希表（Set）记录数组的各个数字，当查找到重复数字则直接返回。
- 算法流程：初始化： 新建 HashSet，记为 dic；遍历数组 nums 中的每个数字 num：当 num在 dic 中，说明重复，直接返回 num；将 num 添加至 dic 中 。

- 复杂度分析：
  - 时间复杂度 O(N)： 遍历数组使用 O(N)，HashSet 添加与查找元素皆为 O(1) 。
  - 空间复杂度 O(N)： HashSet 占用 O(N) 大小的额外空间。

数组中重复的数字

- 要求：时间复杂度`O(N)`，空间复杂度`O(1)`
- 方法三：原地交换：
  - 复杂度要求表明**不能使用排序**，也不能使用 **map/set**
  - 注意到 n 个数字的范围为 `0` 到 `n-1`，考虑类似**选择排序**的思路，通过一次遍历将每个数交换到排序后的位置，如果该位置已经存在相同的数字，那么该数就是重复的。



∵ 在一个长度为 n 的数组 nums 里的所有数字都在 0～n-1 的范围内

∴ 数组元素的索引和值是一对多的关系，因此可建立索引和值的映射

数组中重复的数字

算法流程：

1. 遍历数组 $nums$ ，设索引初始值为 $i = 0$ ：
    1. **若** $nums[i] = i$ **：** 说明此数字已在对应索引位置，无需交换，因此跳过；
    2. **若** $nums[nums[i]] = nums[i]$ **：** 代表索引 $nums[i]$ 处和索引 i 处的元素值都为 $nums[i]$ ，即找到一组重复值，返回此值 $nums[i]$ ；
    3. **否则：** 交换索引为 i 和 $nums[i]$ 的元素值，将此数字交换至对应索引位置。
2. 若遍历完毕尚未返回，则返回 -1 。

复杂度分析：

- **时间复杂度** $O(N)$ **：** 遍历数组使用 $O(N)$ ，每轮遍历的判断和交换操作使用 $O(1)$ 。
- **空间复杂度** $O(1)$ **：** 使用常数复杂度的额外空间。

数组中重复的数字

- 示例

```
Input:
{2, 3, 1, 0, 2, 5}

Output:
2
```

```
position-0 : (2,3,1,0,2,5) // 2 <-> 1
             (1,3,2,0,2,5) // 1 <-> 3
             (3,1,2,0,2,5) // 3 <-> 0
             (0,1,2,3,2,5) // already in position
position-1 : (0,1,2,3,2,5) // already in position
position-2 : (0,1,2,3,2,5) // already in position
position-3 : (0,1,2,3,2,5) // already in position
position-4 : (0,1,2,3,2,5) // nums[i] == nums[nums[i]], exit
```

数组中重复的数字

	时间	空间
排序sort	$O(n \log n)$	$O(1)$
哈希表	$O(n)$	$O(n)$
原地交换	$O(n)$	$O(1)$

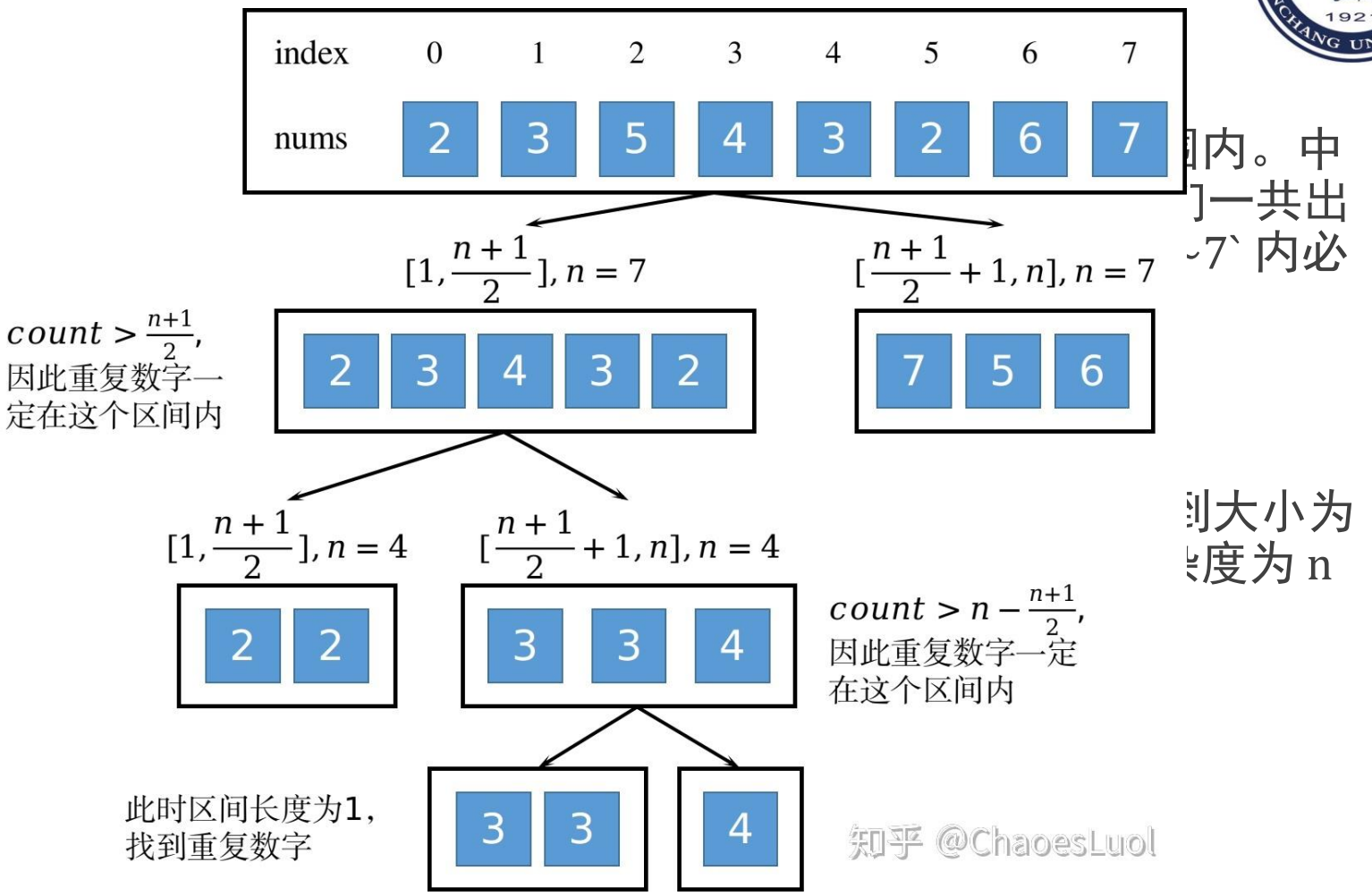
不修改数组找出重复的数字

- 在一个长度为n+1的数组里的所有数字都在1到n的范围内，所以数组中至少有一个数字是重复的。
- 请找出数组中任意一个重复的数字，但不能修改输入的数组。
- 例如，如果输入长度为8的数组{2, 3, 5, 4, 3, 2, 6, 7}，那么对应的输出是重复的数字2或者3。

- 要求：时间复杂度`O(NlogN)`，空间复杂度`O(1)`

不修改数组找出重复的数字

- 二分查找
  - 以长度为 8 的数组 `{ 间的数字 `4` 将 `1~7` 现了 5 次，说明 `1~4 有重复的数字。
- 复杂度分析:
  - 用二分的方法，每次 1；每次缩小范围之后
  - 时间复杂度为 O(nlo
  - 不需要额外的辅助空



数组中出现次数超过一半的数字

剑指 Offer 39. 数组中出现次数超过一半的数字

难度 简单 216 收藏 分享 切换为英文 接收动态 反馈

数组中有一个数字出现的次数超过数组长度的一半，请找出这个数字。

你可以假设数组是非空的，并且给定的数组总是存在多数元素。

示例 1:

```
输入：[1, 2, 3, 2, 2, 2, 5, 4, 2]
输出：2
```

数组中出现次数超过一半的数字

	时间	空间
排序sort	$O(n\log n)$	$O(1)$
哈希表	$O(n)$	$O(n)$
分治	$O(n\log n)$	$O(1)$
投票法	O(n)	O(1)

搜索旋转排序数组

33. 搜索旋转排序数组

难度 中等 1655 收藏 分享 切换为英文 接收动态 反馈

整数数组 `nums` 按升序排列，数组中的值 **互不相同** 。

在传递给函数之前，`nums` 在预先未知的某个下标 `k`（`0 <= k < nums.length`）上进行了 **旋转**，使数组变为 `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]`（下标 **从 0 开始** 计数）。例如，`[0,1,2,4,5,6,7]` 在下标 `3` 处经旋转后可能变为 `[4,5,6,7,0,1,2]` 。

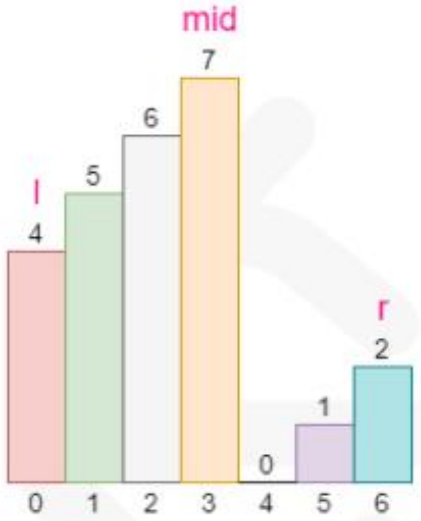
给你 **旋转后** 的数组 `nums` 和一个整数 `target` ，如果 `nums` 中存在这个目标值 `target` ，则返回它的下标，否则返回 `-1` 。

示例 1:

```
输入：nums = [4,5,6,7,0,1,2], target = 0
输出：4
```

搜索旋转排序数组

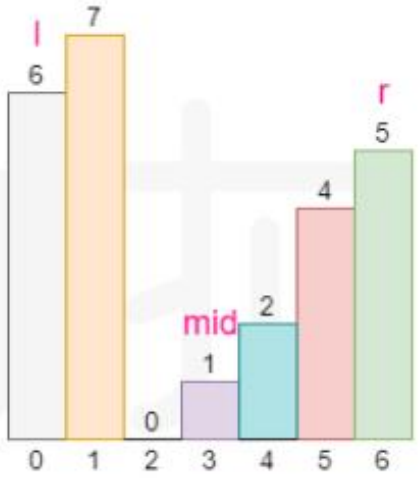
- 二分查找



[l, mid] 是有序数组

如果 target = 5，在 [l, mid - 1] 中寻找

如果 target = 2，在 [mid + 1, r] 中寻找



[mid + 1, r] 是有序数组

如果 target = 6，在 [l, mid - 1] 中寻找

如果 target = 4，在 [mid + 1, r] 中寻找

搜索旋转排序数组II

- 思考：如果nums 可能包含重复元素。

81. 搜索旋转排序数组 II

难度 中等　504　收藏　分享　切换为英文　接收动态　反馈

已知存在一个按非降序排列的整数数组 `nums` ，数组中的值不必互不相同。

在传递给函数之前， `nums` 在预先未知的某个下标 `k` （`0 <= k < nums.length`）上进行了 **旋转** ，使数组变为 `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` （下标 **从 0 开始** 计数）。例如， `[0,1,2,4,4,4,5,6,6,7]` 在下标 `5` 处经旋转后可能变为 `[4,5,6,6,7,0,1,2,4,4]` 。

给你 **旋转后** 的数组 `nums` 和一个整数 `target` ，请你编写一个函数来判断给定的目标值是否存在于数组中。如果 `nums` 中存在这个目标值 `target` ，则返回 `true` ，否则返回 `false` 。

示例 1：

```
输入：nums = [2,5,6,0,0,1,2], target = 0
输出：true
```

寻找旋转排序数组中的最小值

153. 寻找旋转排序数组中的最小值

难度 中等　601　收藏　分享　切换为英文　接收动态　反馈

已知一个长度为 `n` 的数组，预先按照升序排列，经由 `1` 到 `n` 次 **旋转** 后，得到输入数组。例如，原数组 `nums = [0,1,2,4,5,6,7]` 在变化后可能得到：

- 若旋转 `4` 次，则可以得到 `[4,5,6,7,0,1,2]`
- 若旋转 `7` 次，则可以得到 `[0,1,2,4,5,6,7]`

注意，数组 `[a[0], a[1], a[2], ..., a[n-1]]` **旋转一次** 的结果为数组 `[a[n-1], a[0], a[1], a[2], ..., a[n-2]]` 。

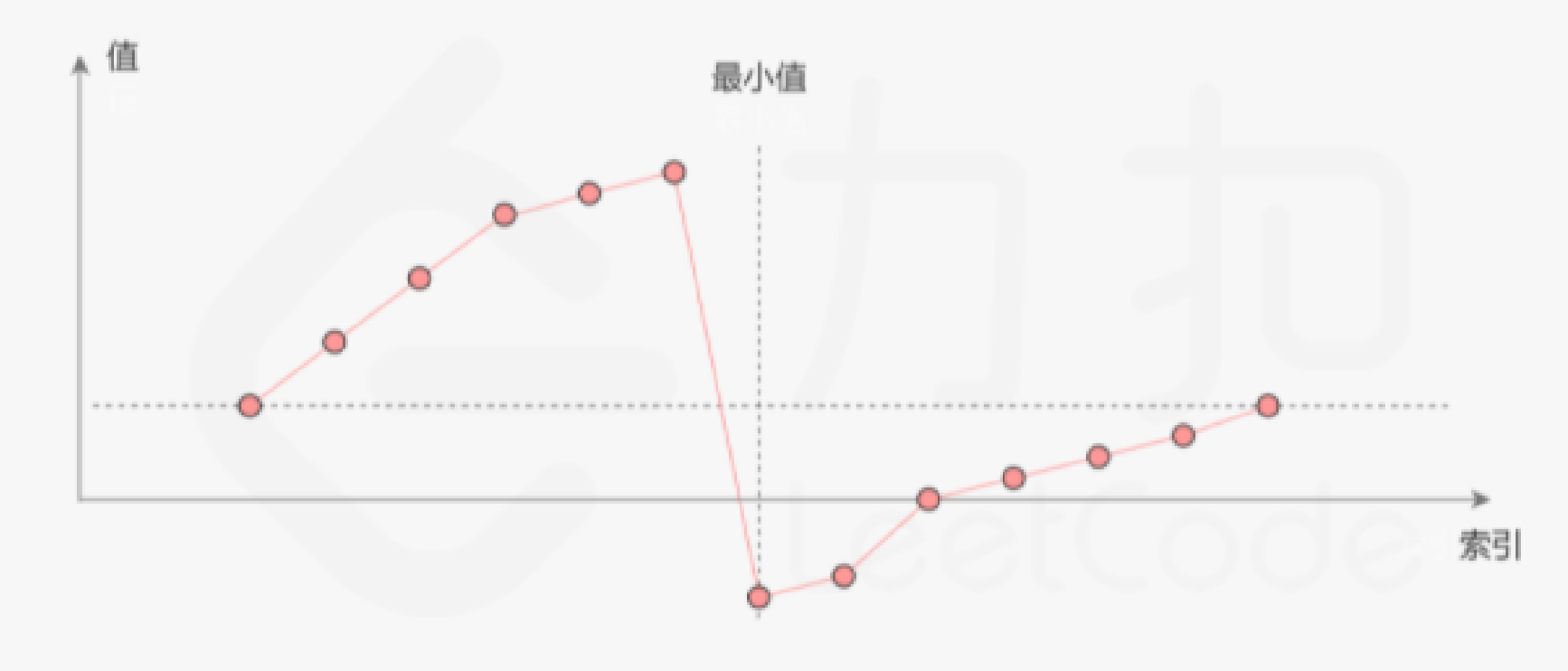
给你一个元素值 **互不相同** 的数组 `nums` ，它原来是一个升序排列的数组，并按上述情形进行了多次旋转。请你找出并返回数组中的 **最小元素** 。

示例 1：

输入： `nums = [3,4,5,1,2]`
输出： `1`
解释： 原数组为 `[1,2,3,4,5]` ，旋转 `3` 次得到输入数组。

寻找旋转排序数组中的最小值

- 二分查找



寻找旋转排序数组中的最小值II

- 思考：如果nums 可能包含重复元素。

154. 寻找旋转排序数组中的最小值 II

难度 困难　416　收藏　分享　切换为英文　接收动态　反馈

已知一个长度为 `n` 的数组，预先按照升序排列，经由 `1` 到 `n` 次 **旋转** 后，得到输入数组。例如，原数组 `nums = [0,1,4,4,5,6,7]` 在变化后可能得到：

- 若旋转 `4` 次，则可以得到 `[4,5,6,7,0,1,4]`
- 若旋转 `7` 次，则可以得到 `[0,1,4,4,5,6,7]`

注意，数组 `[a[0], a[1], a[2], ..., a[n-1]]` **旋转一次** 的结果为数组 `[a[n-1], a[0], a[1], a[2], ..., a[n-2]]` 。

给你一个可能存在 **重复** 元素值的数组 `nums` ，它原来是一个升序排列的数组，并按上述情形进行了多次旋转。请你找出并返回数组中的 **最小元素** 。

示例 1：

```
输入：nums = [1,3,5]
输出：1
```

Thanks