画解剑指 Offer - LeetBook - 力扣（LeetCode）全球极客挚爱的技术成长平台

剑指 Offer 11. 旋转数组的最小数字 - 解决方案

题目描述

把一个数组最开始的若干个元素搬到数组的末尾，我们称之为数组的旋转。输入一个递增排序的数组的一个旋转，输出旋转数组的最小元素。例如，数组 `[3,4,5,1,2]` 为 `[1,2,3,4,5]` 的一个旋转，该数组的最小值为1。

示例 1：

```
输入：[3,4,5,1,2]
输出：1
```

示例 2：

```
输入：[2,2,2,0,1]
输出：0
```

注意：本题与主站 154 题 相同

解题方案

思路

- 标签：二分查找
- 整体思路：首先数组是一个有序数组的旋转，从这个条件可以看出，数组是有大小规律的，可以使用二分查找利用存在的规律快速找出结果
- 时间复杂度：O(logn)O(logn)，空间复杂度：O(1)O(1)

算法流程

1. 初始化下标 `left` 和 `right`
2. 每次计算中间下标 `mid = (right + left) / 2` ，这里的除法是取整运算，不能出现小数
3. 当 `numbers[mid] < numbers[right]` 时，说明最小值在 `[left, mid]` 区间中，则令 `right = mid` ，用于下一轮计算
4. 当 `numbers[mid] > numbers[right]` 时，说明最小值在 `[mid, right]` 区间中，则令 `left = mid + 1` ，用于下一轮计算
5. 当 `numbers[mid] == numbers[right]` 时，无法判断最小值在哪个区间之中，此时让 `right--` ，缩小区间范围，在下一轮进行判断

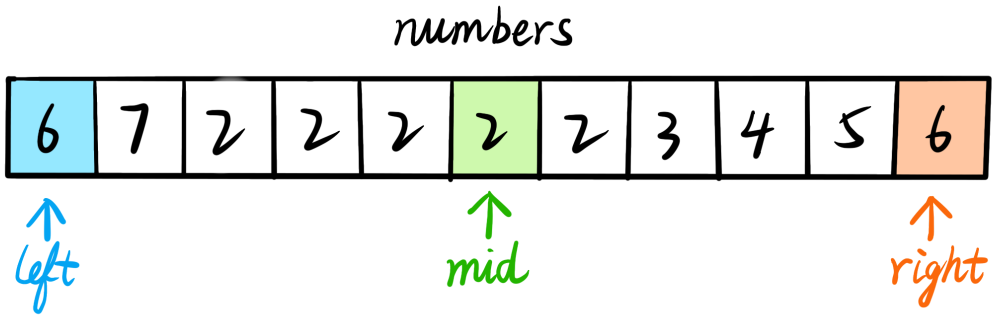
为什么是 `right--` 缩小范围，而不是 `left++` ？

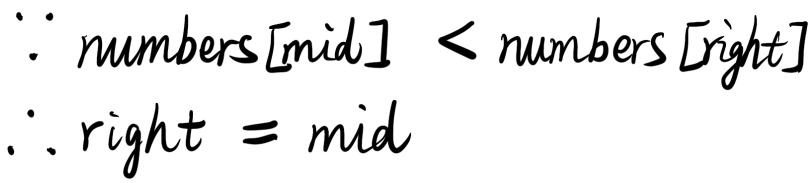
- 因为数组是升序的，所以最小值一定靠近左侧，而不是右侧
- 比如，当存在 `[1,2,2,2,2]` 这种情况时，`left = 0` ，`right = 4` ，`mid = 2` ，数值满足 `numbers[mid] == numbers[right]` 这个条件，如果 `left++` ，则找不到最小值

代码

```
class Solution {
    public int minArray(int[] numbers) {
        int left = 0;
        int right = numbers.length - 1;
        while (left < right) {
            int mid = (right + left) / 2;
            if (numbers[mid] < numbers[right]) {
                right = mid;
            } else if (numbers[mid] > numbers[right]) {
                left = mid + 1;
            } else {
                right --;
            }
        }
        return numbers[left];
    }
}
```

画解





花絮

讨论区

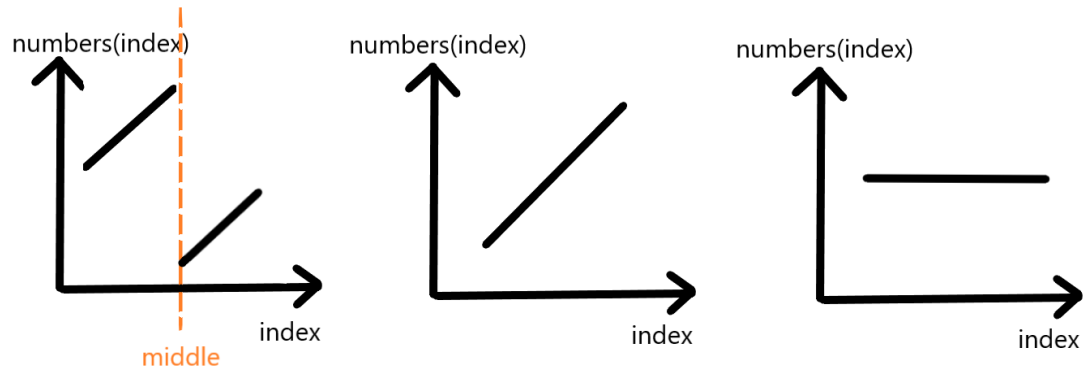
共 27 个讨论

最热

竟成 L1

来自浙江2021-07-09

解题思路中，关于为何numbers[middle]要跟numbers[right]进行比较，可以参考下图。旋转数组无非是如下3种场景，其他场景约等于这3种场景的变种。可见，在图中任何一处画一个纵向虚线作为middle，如果最右侧的点的纵坐标numbers[right]大于numbers[middle]，则最小数一定在left到middle之间（含left点和middle点）。小于和等于的场景可自行结合下图判断。



16

Sabertoothkk L1

来自江苏2021-02-23

这个也过了

```
class Solution {
    public int minArray(int[] numbers) {
        for(int i = 1;i < numbers.length ;i++){
            if(numbers[i] < numbers[i-1]){
                return numbers[i];
            }
        }
        return numbers[0];
    }
}
```

16

哈哈 L1
来自北京2021-10-06

直接从左往右扫描，出现第一个后值小于前值的，不就是吗，时间复杂度是O(N)

3

taste95

来自辽宁2021-11-13

这个也过了 「代码块」

你这时间复杂度是O（n）的啊，二分法O（logn）

2

Bytws L1
来自陕西2022-03-09

```
class Solution {
    public int minArray(int[] numbers) {
        int min=numbers[0];
        int length=numbers.length;
        for(int i=1;i<numbers.length;i++){
            if(min>numbers[i]){
                min=numbers[i];
            }
        }
       return min;
    }
}
```

这也过了，旋转有啥用吗

1

zhiyong.lizy

来自河北2022-03-08

这个解答和题目中描述的旋转有什么关系？旋转这个条件是用来干嘛的？另外给出来的测试case也没有旋转呀？如果单纯的是递增数组，二分法确实是OK的。

1

活力甜橙奶昔

来自陕西2021-04-06

其实最后返回值应该min(numbers[left],numbers[0]),毕竟不是纯递增，例如[0,2,3,7,8,1,2,6],这样按照二分法找出1是最小，但实际上第一个0，才是最小的。

[0,2,3,7,8,1,2,6]不是由递增数组旋转来的。汗

1

改密码必死

来自广东2021-03-28

其实最后返回值应该min(numbers[left],numbers[0]),毕竟不是纯递增，例如[0,2,3,7,8,1,2,6],这样按照二分法找出1是最小，但实际上第一个0，才是最小的。

不看题的吗？

1

MR超级程序员~ L1
来自广东2022-10-27

解题思路中，关于为何numbers[middle]要跟numbers[right]进行比较，可以参考下图。旋转数组无非是如下3种场景，其他场景约等于这3种场景的变种。可见，在图中任何一处画一个纵向虚线作为middle，如果最右侧的点的纵坐标numbers[right]大于numbers[middle]，则最小数一定在l

不用啦，两行代码搞定

```
class Solution {
    public int minArray(int[] numbers) {
        Arrays.sort(numbers);
        return numbers[0];
    }
}
```

来自中国澳门2022-08-15

这个也过了 「代码块」

二分法的时间复杂度低一点，正常遍历也可以，只不过是O(n)