154. 寻找旋转排序数组中的最小值 II

ITCharge 大约 2 分钟

- 标签：数组、二分查找
- 难度：困难

题目链接

- 154. 寻找旋转排序数组中的最小值 II - 力扣

题目大意

描述：给定一个数组 $nums$，$nums$ 是有升序数组经过 $1 \sim n$ 次「旋转」得到的。但是旋转次数未知。数组中可能存在重复元素。

要求：找出数组中的最小元素。

说明：

- 旋转：将数组整体右移 1 位。数组 $[a[0], a[1], a[2], ..., a[n-1]]$ 旋转一次的结果为数组 $[a[n-1], a[0], a[1], a[2], ..., a[n-2]]$。
- $n == nums.length$。
- $1 \le n \le 5000$。
- $-5000 \le nums[i] \le 5000$
- $nums$ 原来是一个升序排序的数组，并进行了 $1 \sim n$ 次旋转。

示例：

- 示例 1：

```py
输入：nums = [1,3,5]
输出：1
```

- 示例 2：

```py
输入：nums = [2,2,2,0,1]
```

输出：0

解题思路

思路 1：二分查找

数组经过「旋转」之后，会有两种情况，第一种就是原先的升序序列，另一种是两段升序的序列。

第一种的最小值在最左边。

```text
          *
        *
      *
    *
  *
*
```

第二种最小值在第二段升序序列的第一个元素。

```text
    *
  *
*
          *
        *
      *
```

最直接的办法就是遍历一遍，找到最小值。但是还可以有更好的方法。考虑用二分查找来降低算法的时间复杂度。

创建两个指针 $left$、$right$，分别指向数组首尾。然后计算出两个指针中间值 mid。将 mid 与右边界进行比较。

1. 如果 $nums[mid] > nums[right]$，则最小值不可能在 mid 左侧，一定在 mid 右侧，则将 $left$ 移动到 $mid + 1$ 位置，继续查找右侧区间。
2. 如果 $nums[mid] < nums[right]$，则最小值一定在 mid 左侧，令右边界 $right$ 为 mid，继续查找左侧区间。
3. 如果 $nums[mid] == nums[right]$，无法判断在 mid 的哪一侧，可以采用 `right = right - 1` 逐步缩小区域。

思路 1：代码

```py
class Solution:
    def findMin(self, nums: List[int]) -> int:
        left = 0
        right = len(nums) - 1
        while left < right:
            mid = left + (right - left) // 2
            if nums[mid] > nums[right]:
                left = mid + 1
            elif nums[mid] < nums[right]:
                right = mid
            else:
                right = right - 1
        return nums[left]
```

思路 1：复杂度分析

- **时间复杂度**：$O(\log n)$。
- **空间复杂度**：$O(1)$。