

0153. 寻找旋转排序数组中的最小值

👤 ITCharge ⌚ 大约 2 分钟

- 标签：数组、二分查找
- 难度：中等

题目链接

- [0153. 寻找旋转排序数组中的最小值 - 力扣](#)

题目大意

描述： 给定一个数组 $nums$ ， $nums$ 是有升序数组经过「旋转」得到的。但是旋转次数未知。数组中不存在重复元素。

要求： 找出数组中的最小元素。

说明：

- 旋转操作：将数组整体右移若干位置。
- $n == nums.length$ 。
- $1 \leq n \leq 5000$ 。
- $-5000 \leq nums[i] \leq 5000$ 。
- $nums$ 中的所有整数互不相同。
- $nums$ 原来是一个升序排序的数组，并进行了 1 至 n 次旋转。

示例：

- 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。

py

- 示例 2:

py

输入: `nums = [4,5,6,7,0,1,2]`

输出: `0`

解释: 原数组为 `[0,1,2,4,5,6,7]` , 旋转 `4` 次得到输入数组。

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。第二种最小值在第二段升序序列的第一个元素。

py



py



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。让后计算出两个指针中间值 *mid*。将 *mid* 与两个指针做比较。

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 $nums[mid] \leq nums[right]$, 则最小值一定在 *mid* 左侧, 或者 *mid* 位置, 将 *right* 移动到 *mid* 位置上, 继续查找左侧区间。

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
            else:
                right = mid
        return nums[left]
```

思路 1：复杂度分析

- **时间复杂度：** $O(\log n)$ 。二分查找算法的时间复杂度为 $O(\log n)$ 。
- **空间复杂度：** $O(1)$ 。只用到了常数空间存放若干变量。