6月13日日报

本日学习内容

1. 完成ZARA仿写。上传到github仓库

今日算法题

题目1：33. 搜索旋转排序数组

33. 搜索旋转排序数组

已解答

中等 相关标签 相关企业

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` 。

你必须设计一个时间复杂度为 `O(log n)` 的算法解决此问题。

示例 1：

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

示例 2：

> **输入：** nums = [4,5,6,7,0,1,2], target = 3
> **输出：** -1

示例 3：

> **输入：** nums = [1], target = 0
> **输出：** -1

提示：

- `1 <= nums.length <= 5000`
- -10^4 <= nums[i] <= 10^4
- `nums` 中的每个值都 **独一无二**
- 题目数据保证 `nums` 在预先未知的某个下标上进行了旋转
- -10^4 <= target <= 10^4

```
class Solution {
public:
    int search(vector<int>& nums, int target) {
        int left = 0, right = nums.size() - 1;
```

```
        while (left <= right) {
            int mid = left + (right - left) / 2;

            if (nums[mid] == target) {
                return mid;
            }

            // 左半部分有序
            if (nums[left] <= nums[mid]) {
                if (nums[left] <= target && target < nums[mid]) {
                    right = mid - 1;
                } else {
                    left = mid + 1;
                }
            }
            // 右半部分有序
            else {
                if (nums[mid] < target && target <= nums[right]) {
                    left = mid + 1;
                } else {
                    right = mid - 1;
                }
            }
        }

        return -1;
    }
};
```

本日遇到的问题

1. github上传时分支遇到问题，在查询后解决

明日学习计划

1. 看学长博客，准备开始着手网易云的学习
2. 开始复习期末