0035. 搜索插入位置

ITCharge 大约 2 分钟

- 标签：数组、二分查找
- 难度：简单

题目链接

- 0035. 搜索插入位置 - 力扣

题目大意

描述：给定一个排好序的数组 $nums$，以及一个目标值 $target$。

要求：在数组中找到目标值，并返回下标。如果找不到，则返回目标值按顺序插入数组的位置。

说明：

- $1 \le nums.length \le 10^4$。
- $-10^4 \le nums[i] \le 10^4$。
- $nums$ 为无重复元素的升序排列数组。
- $-10^4 \le target \le 10^4$。

示例：

- 示例 1：

```py
输入：nums = [1,3,5,6], target = 5
输出：2
```

解题思路

思路 1：二分查找

设定左右节点为数组两端，即 `left = 0 ， right = len(nums) - 1` ，代表待查找区间为 $[left, right]$（左闭右闭）。

取两个节点中心位置 mid，先比较中心位置值 $nums[mid]$ 与目标值 $target$ 的大小。

- 如果 $target == nums[mid]$，则当前中心位置为待插入数组的位置。
- 如果 $target > nums[mid]$，则将左节点设置为 $mid + 1$，然后继续在右区间 $[mid + 1, right]$ 搜索。
- 如果 $target < nums[mid]$，则将右节点设置为 $mid - 1$，然后继续在左区间 $[left, mid - 1]$ 搜索。

直到查找到目标值返回待插入数组的位置，或者等到 $left > right$ 时停止查找，此时 $left$ 所在位置就是待插入数组的位置。

思路 1：二分查找代码

```py
class Solution:
    def searchInsert(self, nums: List[int], target: int) -> int:
        size = len(nums)
        left, right = 0, size - 1

        while left <= right:
            mid = left + (right - left) // 2
            if nums[mid] == target:
                return mid
            elif nums[mid] < target:
                left = mid + 1
            else:
                right = mid - 1

        return 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。