34. Find First and Last Position of Element in Sorted Array

Medium 4754 183 Add to List Share

Given an array of integers `nums` sorted in ascending order, find the starting and ending position of a given `target` value.

If `target` is not found in the array, return `[-1, -1]`.

Follow up: Could you write an algorithm with `O(log n)` runtime complexity?

Example 1:

```
Input: nums = [5,7,7,8,8,10], target = 8
Output: [3,4]
```

Example 2:

```
Input: nums = [5,7,7,8,8,10], target = 6
Output: [-1,-1]
```

Example 3:

```
Input: nums = [], target = 0
Output: [-1,-1]
```

Constraints:

- `0 <= nums.length <= 10^5`
- `-10^9 <= nums[i] <= 10^9`
- `nums` is a non-decreasing array.
- `-10^9 <= target <= 10^9`

```
int left_bound(int[] nums, int target) {
    if (nums.length == 0) return -1;
    int left = 0;
    int right = nums.length;

    while (left < right) {
        int mid = (left + right) / 2
        if (nums[mid] == target) {
            right = mid;
        } else if (nums[mid] < target) {
            left = mid + 1;
        } else if (nums[mid] > target) {
            right = mid;
        }
    }
    if (left == nums.length) return -1;
    return nums[left] == target ? left : -1;
}
```

nus.length
[left, right)
[left, right)
[left, mid) [mid+1, right)
left ∈ [0, nums.length]
left == right

```
int right_bound(int[] nums, int target) {
    if (nums.length == 0) return -1;
    int left = 0;
    int right = nums.length;

    while (left < right) {
        int mid = (left + right) / 2
        if (nums[mid] == target) {
            left = mid + 1; => mid = left - 1
        } else if (nums[mid] < target) {
            left = mid + 1;
        } else if (nums[mid] > target) {
            right = mid;
        }
    }
    if (left == 0) return -1;
    return nums[left-1] == target ? (left-1) : -1;
}
```

left ∈ [0, nums.length]

```cpp
class Solution {
public:
    vector<int> searchRange(vector<int>& nums, int target) {

        vector<int> res;

        if (nums.size() == 0) {
            res.push_back(-1);
            res.push_back(-1);
            return res;
        }

        res.push_back(low_bound(nums, target));
        res.push_back(high_bound(nums, target));

        return res;

    }

private:
    int low_bound(vector<int> nums, int target) {
        int left = 0;
        int right = nums.size();

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

            if (nums[mid] < target) {
                left = mid + 1;
            } else if (nums[mid] > target) {
                right = mid;
            } else if (nums[mid] == target) {
                right = mid;
            }
        }

        if (left == nums.size() || nums[left] != target) {
            return -1;
        }

        return left;
    }

    int high_bound(vector<int> nums, int target) {
        int left = 0;
        int right = nums.size();

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

            if (nums[mid] < target) {
                left = mid + 1;
            } else if (nums[mid] > target) {
                right = mid;
            } else if (nums[mid] == target) {
                left = mid + 1;
            }
        }

        left--;

        if (left < 0 || nums[left] != target) {
            return -1;
        }

        return left;
    }
};
```