Question 34: Find First and Last Position of Element in Sorted Array

Given an array of integers `nums` sorted in non-decreasing order, find the starting and ending position of a given target value. If the target is not found in the array, return `[-1, -1]`. The algorithm must have a runtime complexity of O(log n).

Examples:

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` <= 105

- -109 <= `nums[i]` <= 109

- `nums` is a non-decreasing array.

- -109 <= `target` <= 109

Solution Approach:

The problem can be solved efficiently using a binary search algorithm. We will implement two binary search functions: one to find the leftmost occurrence of the target and another to find the rightmost occurrence of the target. These functions will narrow down the search space to find the first and last positions of the target value respectively.

Solution Steps:

1. Define two helper functions: `find_left(nums, target)` and `find_right(nums, target)` to find the leftmost and rightmost occurrence of the target using binary search.
2. Initialize `left_idx` using `find_left(nums, target)` and `right_idx` using `find_right(nums, target)`.
3. If `left_idx` is less than or equal to `right_idx`, return `[left_idx, right_idx]`, otherwise return `[-1, -1]`.

Pseudocode:

```
def find_left(nums, target):
    # Binary search to find leftmost occurrence of target
    # Return index or insertion point
def find_right(nums, target):
    # Binary search to find rightmost occurrence of target
    # Return index or insertion point
def searchRange(nums, target):
    left_idx = find_left(nums, target)
    right_idx = find_right(nums, target)
    if left_idx <= right_idx:
        return [left_idx, right_idx]
    else:
        return [-1, -1]
```

Example Usage:

```
nums = [5, 7, 7, 8, 8, 10]
target = 8
solution = Solution()
print(solution.searchRange(nums, target))  # Output: [3, 4]
```