Binary Search and Its Variants (Iterative & Recursive)

Algorithm	Best Case	Average Case	Worst Case
Binary Search (Iterative & Recursive)	O(1)	O(log n)	O(log n)
First Occurrence (Iterative & Recursive)	O(1)	O(log n)	O(log n)
Last Occurrence (Iterative & Recursive)	O(1)	O(log n)	O(log n)

Python Code for Searching Algorithms (Iterative & Recursive)

```python
# Binary Search (Iterative)
def binary_search(nums, target):
    left, right = 0, len(nums) - 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 -1
    # Time Complexity: O(log n)

# Binary Search (Recursive)
def binary_recursive(nums, target, left, right):
    if left > right:
        return -1

    mid = left + (right - left) // 2
    if nums[mid] == target:
        return mid
    elif nums[mid] > target:
        return binary_recursive(nums, target, left, mid - 1)
    else:
        return binary_recursive(nums, target, mid + 1, right)
    # Time Complexity: O(log n)

# First Occurrence (Iterative)
def first_occurrence(nums, target):
    left, right = 0, len(nums) - 1
    sol = -1
    while left <= right:
        mid = left + (right - left) // 2
        if nums[mid] == target:
            sol = mid
            right = mid - 1  # Keep searching left
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return sol
    # Time Complexity: O(log n)

# First Occurrence (Recursive)
def first_occurrence_recursive(nums, target, left, right, sol=-1):
    if left > right:
        return sol

    mid = left + (right - left) // 2
    if nums[mid] == target:
        return first_occurrence_recursive(nums, target, left, mid - 1, mid)
    elif nums[mid] > target:
        return first_occurrence_recursive(nums, target, left, mid - 1, sol)
    else:
        return first_occurrence_recursive(nums, target, mid + 1, right, sol)
    # Time Complexity: O(log n)

# Last Occurrence (Iterative)
def last_occurrence(nums, target):
    left, right = 0, len(nums) - 1
    sol = -1
    while left <= right:
        mid = left + (right - left) // 2
        if nums[mid] == target:
            sol = mid
            left = mid + 1  # Keep searching right
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return sol
    # Time Complexity: O(log n)

# Last Occurrence (Recursive)
def last_occurrence_recursive(nums, target, left, right, sol=-1):
    if left > right:
        return sol

    mid = left + (right - left) // 2
    if nums[mid] == target:
        return last_occurrence_recursive(nums, target, mid + 1, right, mid)
    elif nums[mid] > target:
        return last_occurrence_recursive(nums, target, left, mid - 1, sol)
    else:
        return last_occurrence_recursive(nums, target, mid + 1, right, sol)
    # Time Complexity: O(log n)
```