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

Medium Accepted: 1.8M Submissions: 4.1M Acceptance Rate: 43.2%

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

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

You must 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 <= 105
* -109 <= nums[i] <= 109
* nums is a non-decreasing array.
* -109 <= target <= 109

**Code: -**

class Solution {

public:

    int lower(vector<int>& ns, int tg)

    {

        int l = 0, h = ns.size() -1, m = 0;

        int ans = -1;

        while (l <= h) {

            m = l + (h-l)/2;

            if (ns[m] >= tg) {

                h = m-1;

                ns[m] == tg ? ans = m : ans = ans;

            } else {

                l = m+1;

            }

        }

        return ans;

    }

    int upper(vector<int>& ns, int low, int tg)

    {

        int l = low, h = ns.size() -1, m = 0;

        int ans = -1;

        while (l <= h) {

            m = l + (h-l)/2;

            if (ns[m] <= tg) {

                l = m+1;

                ns[m] == tg ? ans = m : ans = ans;

            } else {

                h = m-1;

            }

        }

        return ans;

    }

    vector<int> searchRange(vector<int>& ns, int tg) {

        int l = lower(ns, tg), u = -1;

        if (l >= 0) u = upper(ns, l, tg);

        return {l, u};

    }

};

**T.C: - O(log2N)**

**S.C: - O(1)**