# LeetCode 34

## Description

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

Your algorithm's runtime complexity must be in the order of O(log n).

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

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]

## Thought

The problem can turn into 2 BS problems:

1. find the index of the first target
2. find the index of the last target

## Solution

```
public class Solution {
public int[] searchRange(int[] nums, int target) {
    int[] result = new int[2];
    result[0] = findFirst(nums, target);
    result[1] = findLast(nums, target);
    return result;
}

private int findFirst(int[] nums, int target){
    int idx = -1;
    int start = 0;
    int end = nums.length - 1;
    while(start <= end){
        int mid = (start + end) / 2;
```

```
            if(nums[mid] >= target){
                end = mid - 1;
            }else{
                start = mid + 1;
            }
            if(nums[mid] == target) idx = mid;
        }
        return idx;
    }

    private int findLast(int[] nums, int target){
        int idx = -1;
        int start = 0;
        int end = nums.length - 1;
        while(start <= end){
            int mid = (start + end) / 2;
            if(nums[mid] <= target){
                start = mid + 1;
            }else{
                end = mid - 1;
            }
            if(nums[mid] == target) idx = mid;
        }
        return idx;
    }
    }
```

## Takeaways

- Try to turn problem into a Bineary search method