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

Medium 17.3K 422

Companies

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]
```

if (T= a[i] && sp==-1) sp = ep = i ;

if (T==a[i] && a[i+1] != T) ep = i

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`

Accepted 1.6M | Submissions 3.7M | Acceptance Rate 42.1%

Approach 1:

As the array is sorted peform linear scan & mark the starting and ending positions of target in the array.

```
for(i: 0 to n-1)
{
    if(a[i]==Target && sp==-1)    sp=ep=i

    if(a[i]==Target && (i==n-1 || a[i]!=Target))
        ep=i
}
```

O(n)

Approach 2:

Using binary search

```
class Solution {
public:
    vector<int> searchRange(vector<int>& nums, int target) {
        int start=binarysearch(nums,target);
        int end=binarysearch(nums,target+1);
        if(start<nums.size()&&nums[start]==target) return {start,end-1};
        else return {-1,-1};
    }

    int binarysearch(vector<int>& nums,int target){
        int l=0,r=nums.size()-1;
        while(l<=r){
            int mid=(l+r)/2;
            if(nums[mid]<target) l=mid+1;
            else r=mid-1;
        }
        return l;
    }
};
```

we modified binary search such that it returns the index of first occurence of an element.

O(log n)