Author: Xiaoqiang Yan

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

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]
```

Solution

According to the problem, we know time complexity should be `O(log n)`, which is the time complexity of Binary Search. So let’s try to use BS to solve this problem.

This problem should return two index, we can find the first index, then find the second index.

Finding the first index, we first find the middle element `mid`, if

`(mid_1<target)`,that means (0~mid) elements are all smaller than target, so we should find target in `(mid_1+1, tail)`. Then we need to find the first index in the latter half of the array. if `mid_2 = target`, the first index will before the mid_2 or exact mid_2, so we should find target in `(mid_1+1, mid_2)`.
Using same idea we can find the last index of target.
But there's a case that maybe we don't have the target in array, so we need to add a corner case to check this condition.

Code

```
vector<int> searchRange(vector<int> &nums, int target){
    if(nums.size()==0 || nums[0] > target) return {-1,-1};
    int pos_1 = BS(nums, 0, nums.size(), target, 0);
    int pos_2 = BS(nums, 0, nums.size(), target, 1);
    return {pos_1, pos_2};
}
int BS(vector<int> &nums, int head, int tail, int target, boo
l forl)
{
    if(forl == 1){
        if(head < tail){
            int mid = head + (tail-head)/2;
            if(nums[mid]<target)
                return BS(nums, mid+1, tail, target, forl);
            else
                return BS(nums, head, mid, target, forl);
          }
        else if(nums[head]==target)
```

```
                return head;
            else
                return -1;
        }
        else{
            if(head< tail){
                int mid = head+ (tail-head+1)/2;
                if(nums[mid]<=target)
                    return BS(nums, mid, tail, target, forl);
                else
                    return BS(nums, head, mid-1, target, forl);
            }
            else if(nums[head]==target)
                return head;
            else
                return -1;
        }
    }
```

Note

We need to learn how to update head, and tail in different cases.

Summary

Binary Search/Recursion.