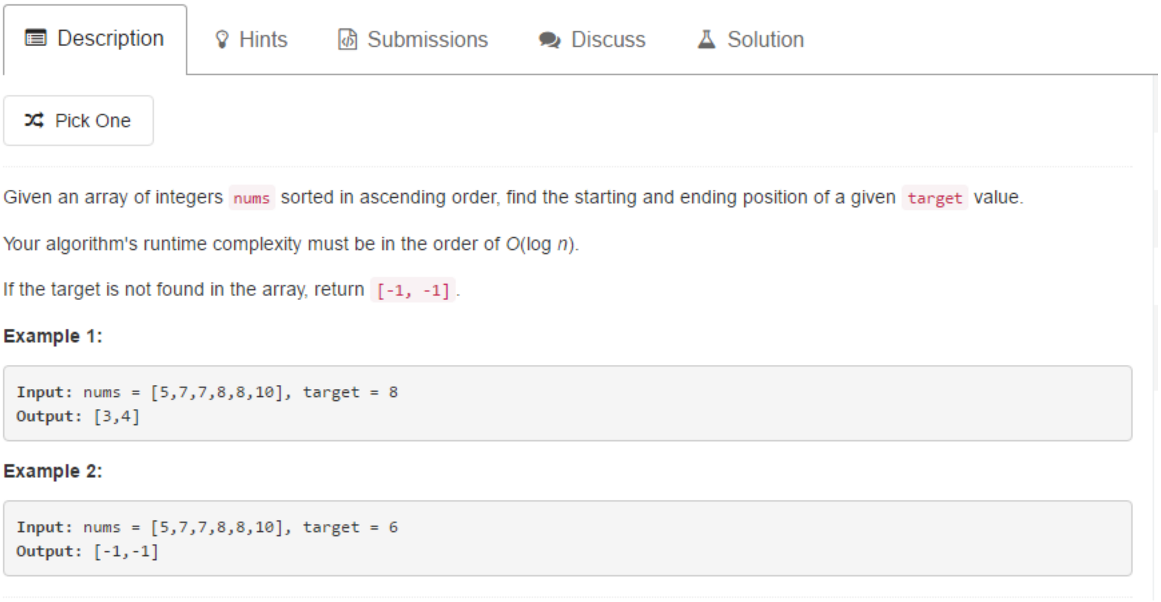
34. Find First and Last Position of Element in Sorted Array



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]
```

```java
public class L34 {

    /*
     * 题意：给定一个按照升序排列的数组，要求找出目标数字所在下标的范围。并且时间复杂度在log(n)
     * 解决：既然数组是已经排序好了的，那么我们只要可以找到其中的一个目标数字，从这个数字向两边
     * 扩散，就可以找到所有我们想要找到的目标数字了，因此可以用二分法查找到其中一个目标数字。
     */
    public int[] searchRange(int[] nums, int target) {
        if(nums.length == 0){
            return new int[] {-1, -1};
        }
        int left = 0;
        int right = nums.length - 1;
        int mid = (left + right) / 2;
        while(left <= right){
            if(nums[mid] == target) break;
            else if(nums[mid] > target) right = mid - 1;
            else left = mid + 1;
            mid = (left + right) / 2;
        }//find one position of the target digit
        if(nums[mid] != target) return new int[] {-1, -1};
        int st = mid;
        int ed = mid;
        while(st >= 0 || ed < nums.length){
            boolean isMatch = false;
            if(st > 0 && nums[st-1] == target) {st--; isMatch = true;}
            if(ed < nums.length - 1 && nums[ed+1] == target) {ed++; isMatch = true;}
            if(!isMatch) break;
        }
        return new int[] {st, ed};
    }
}
```