34. Find First and Last Position of Element in Sorted Array
We will use binary search manually.

```
/**
 * @param {number[]} nums
 * @param {number} target
 * @return {number[]}
 */
var searchRange = function(nums, target) {
    let left = 0, right = nums.length - 1, middle;
    const result = [];
    while (left < right) {
      middle = Math.floor((left + right) / 2)
      if (nums[middle] < target) {
        left = middle + 1
      } else {
        right = middle
      }
    }

    if (nums[left] != target)
      return [-1, -1]
    else
      result.push(left)
    right = nums.length - 1
    while (left < right) {
      middle = Math.floor((left + right) / 2) + 1
      if (nums[middle] > target) {
        right = middle - 1
      } else {
        left = middle
      }
    }
    result.push(right)
  return result
};
```

35. Search Insert Position

```
var searchInsert = function(nums, target) {
    let left = 0, right = nums.length - 1, middle;
    if (nums[nums.length - 1] < target) {
      return nums.length;
    }
```

```
    while (left <= right) {
      middle = Math.floor((left + right) / 2)
      if (nums[middle] < target) {
        left = middle + 1
      } else {
        right = middle - 1
      }
    }
    return left
};
```

704. Binary Search
We will use binary search and check whether it's out of the bound. TC is O(logn).

```
from bisect import *
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        idx = bisect_left(nums, target)
        return idx if idx < len(nums) and nums[idx] == target else -1
```

981. Time Based Key-Value Store

```
from collections import defaultdict
from bisect import *
class TimeMap:

    def __init__(self):
        """
        Initialize your data structure here.
        """
        self.time = defaultdict(list)
        self.value = defaultdict(list)

    def set(self, key: str, value: str, timestamp: int) -> None:
        self.time[key].append(timestamp)
        self.value[key].append(value)

    def get(self, key: str, timestamp: int) -> str:
        if key not in self.time:
          return ''
        idx = bisect_right(self.time[key], timestamp)
        if idx == 0:
          return ""
        return self.value[key][idx - 1]
```

```
# Your TimeMap object will be instantiated and called as such:
# obj = TimeMap()
# obj.set(key,value,timestamp)
# param_2 = obj.get(key,timestamp)
```

33. Search in Rotated Sorted Array

We will confirm nums[middle] and target are in the same section and then execute binary search.

```
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        left, right = 0, len(nums) - 1
        while left <= right:
          middle = (left + right) // 2
          if nums[middle] == target:
            return middle

          if (nums[middle] < nums[0] and target < nums[0]) or (nums[middle] >= nums[0] and target >= nums[0]):
            if nums[middle] < target:
              left = middle + 1
            else:
              right = middle - 1
          else:
            if nums[middle] < nums[0] and target >= nums[0]:
              right = middle - 1
            else:
              left = middle + 1
        return -1
```