704. Binary Search

Easy 1071 55 Add to List Share

Given a **sorted** (in ascending order) integer array `nums` of `n` elements and a `target` value, write a function to search `target` in `nums`. If `target` exists, then return its index, otherwise return `-1`.

Example 1:

```
Input: nums = [-1,0,3,5,9,12], target = 9
Output: 4
Explanation: 9 exists in nums and its index is 4
```

Example 2:

```
Input: nums = [-1,0,3,5,9,12], target = 2
Output: -1
Explanation: 2 does not exist in nums so return -1
```

Note:

1. You may assume that all elements in `nums` are unique.
2. `n` will be in the range `[1, 10000]`.
3. The value of each element in `nums` will be in the range `[-9999, 9999]`.

```
int binarySearch(int[] nums, int target) {
    int left = 0, right = ...;

    while(...) {
        int mid = left + (right - left) / 2;
        if (nums[mid] == target) {
            ...
        } else if (nums[mid] < target) {
            left = ...;
        } else if (nums[mid] > target) {
            right = ...;
        }
    }
    return ...;
}
```

Find a number:

```
int binarySearch(int[] nums, int target) {
    int left = 0;
    int right = ① nums.length - 1;

    while(② left <= right) {
        int mid = left + (right - left) / 2;
        if (nums[mid] == target) {
            return mid;
        } else if (nums[mid] < target) {
            left = ③ mid + 1;
        } else if (nums[mid] > target) {
            right = ③ mid - 1;
        }
    }
    return -1;
}
```

① nums.length - 1 <=> [left, right]
nums.length <=> [left, right)

② while(left <= right)
<=> [right + 1, right]
while(left < right)
<=> [right, right]

③ mid has searched already

```cpp
class Solution {
public:
    int search(vector<int>& nums, int target) {
        int left = 0, right = nums.size() - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) {
                return mid;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else if (nums[mid] > target) {
                right = mid - 1;
            }
        }

        return -1;
    }
};
```