



Template #2:

C++

Java

Python

Copy

```
1 int binarySearch(int[] nums, int target){
2     if(nums == null || nums.length == 0)
3         return -1;
4
5     int left = 0, right = nums.length;
6     while(left < right){
7         // Prevent (left + right) overflow
8         int mid = left + (right - left) / 2;
9         if(nums[mid] == target){ return mid; }
10        else if(nums[mid] < target) { left = mid + 1; }
11        else { right = mid; }
12    }
13
14    // Post-processing:
15    // End Condition: left == right
16    if(left != nums.length && nums[left] == target) return left;
17    return -1;
18 }
```

Template #2 is an advanced form of Binary Search. It is used to search for an element or condition which requires *accessing the current index and its immediate right neighbor's index* in the array.

Key Attributes:

- An advanced way to implement Binary Search.
- Search Condition needs to access the element's immediate right neighbor
- Use the element's right neighbor to determine if the condition is met and decide whether to go left or right
- Guarantees Search Space is at least 2 in size at each step
- Post-processing required. Loop/Recursion ends when you have 1 element left. Need to assess if the remaining element meets the condition.

Distinguishing Syntax:

- Initial Condition: left = 0, right = length
- Termination: left == right
- Searching Left: right = mid
- Searching Right: left = mid+1