A Binary Search Template II

Template #2:

C++ | Java | Python — Copy

```cpp
int binarySearch(vector<int>& nums, int target){
  if(nums.size() == 0)
    return -1;

  int left = 0, right = nums.size();
  while(left < right){
    // Prevent (left + right) overflow
    int mid = left + (right - left) / 2;
    if(nums[mid] == target){ return mid; }
    else if(nums[mid] < target) { left = mid + 1; }
    else { right = mid; }
  }

  // Post-processing:
  // End Condition: left == right
  if(left != nums.size() && nums[left] == target) return left;
  return -1;
}
```

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 element's immediate right neighbor
- Use element's right neighbor to determine if condition is met and decide whether to go left or right
- Gurantees 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`