Binary Search Template III

Report Issue (https://github.com/LeetCode-Feedback/LeetCode-Feedback/issues)

Template #3:

C++ Java Python Copy

```java
int binarySearch(int[] nums, int target) {
    if (nums == null || nums.length == 0)
        return -1;

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

    // Post-processing:
    // End Condition: left + 1 == right
```

Template #3 is another unique form of Binary Search. It is used to search for an element or condition which requires *accessing the current index and its immediate left and right neighbor's index* in the array.

Key Attributes:

- An alternative way to implement Binary Search
- Search Condition needs to access element's immediate left and right neighbors
- Use element's neighbors to determine if condition is met and decide whether to go left or right
- Gurantees Search Space is at least 3 in size at each step
- Post-processing required. Loop/Recursion ends when you have 2 elements left. Need to assess if the remaining elements meet the condition.

Distinguishing Syntax:

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