Insight on Searching on Boundaries in Binary Search

When solving problems, Binary Search can be extended to find **boundary elements**, like the **lower bound** and **upper bound**, in sorted arrays. These concepts are particularly useful for solving placement-oriented problems.

---

1. Lower Bound

The **lower bound** of a target element in a sorted array is the smallest index at which the element is **greater than or equal to** the target.

Key Points:

- If the target exists in the array, it points to the first occurrence of the target.
- If the target does not exist, it points to the position where the target could be inserted to maintain the sorted order.
- The array must be sorted.

Implementation Logic:

- Use Binary Search to narrow the search space.
- Check if the mid-value is **greater than or equal to** the target:
    - If true, update the result and move high to search for earlier indices.
    - Otherwise, move low to search later indices.

Lower Bound In c++

Sample Input

```
7
10 20 30 40 50 50 60
50
```

Sample Output

```
4
```

Sample Input

```
7
10 20 30 40 50 50 60
31
```

Sample Output

```
3
```

Sample Input

```
7
10 20 30 40 50 50 60
39
```

Sample Output

```
3
```

```
#include <iostream>
#include <vector>
using namespace std;

// Function to find the lower bound of the target
int lowerBound(const vector<int>& arr, int target) {
   int low = 0, high = arr.size() - 1;
```

```
    int result = arr.size(); // Default to the end of the array

    while (low <= high) {
        int mid = low + (high - low) / 2;

        if (arr[mid] >= target) {
            result = mid; // Update result and move left
            high = mid - 1;
        } else {
            low = mid + 1;
        }
    }
    return result;
}

int main() {
    int n, target;

    // Input the size of the array
    cin >> n;

    vector<int> arr(n);

    // Input the sorted array elements
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    // Input the target value
    cin >> target;

    // Call the lowerBound function
    int result = lowerBound(arr, target);

    // Print the result
    if (result < n) {
        cout << "Lower Bound index: " << result << endl;
    } else {
        cout << "No element >= " << target << " found in the array." << endl;
    }
    return 0;
}
```

In Java

```
import java.util.Scanner;

class Main {

    // Function to find the lower bound of the target
    public static int lowerBound(int[] arr, int target) {
        int low = 0, high = arr.length - 1;
        int result = arr.length; // Default to the end of the array

        while (low <= high) {
            int mid = low + (high - low) / 2;

            if (arr[mid] >= target) {
                result = mid; // Update result and move left
                high = mid - 1;
            } else {
                low = mid + 1;
            }
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Input the size of the array
        int n = sc.nextInt();

        int[] arr = new int[n];

        // Input the sorted array elements
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        // Input the target value
        int target = sc.nextInt();

        // Call the lowerBound function
        int result = lowerBound(arr, target);

        // Print the result
        if (result < n) {
            System.out.println("Lower Bound index: " + result);
        } else {
            System.out.println("No element >= " + target + " found in the array.");
        }
        sc.close();
    }
}
```

The upperBound function calculates the index of the first element that is strictly greater than the target.

C++ Implementation
Input

```
5
10 20 30 40 50
9
```

Output

```
0
```

Input

```
5
10 20 30 40 50
51
```

Output

```
No element > 51 found in the array.
```

Input:

```
7
1 2 4 4 4 5 7
4
```

Output:

```
Upper Bound index: 5
```

Input:

```
5
1 3 5 7 9
6
```

Output:

```
Upper Bound index: 3
```

```cpp
#include <iostream>
#include <vector>
using namespace std;

// Function to find the upper bound of the target
int upperBound(const vector<int>& arr, int target) {
    int low = 0, high = arr.size() - 1;
    int result = arr.size(); // Default to the end of the array

    while (low <= high) {
        int mid = low + (high - low) / 2;

        if (arr[mid] > target) {
            result = mid; // Update result and move left
            high = mid - 1;
        } else {
            low = mid + 1;
        }
    }
    return result;
}

int main() {
    int n, target;

    // Input the size of the array
    cin >> n;
    vector<int> arr(n);

    // Input the sorted array elements
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    // Input the target value
    cin >> target;

    // Call the upperBound function
    int result = upperBound(arr, target);

    // Print the result
    if (result < n) {
        cout << "Upper Bound index: " << result << endl;
    } else {
        cout << "No element > " << target << " found in the array." << endl;
    }

    return 0;
}
```

Java Implementation

```
javaimport java.util.Scanner;

public class UpperBound {

    // Function to find the upper bound of the target
    public static int upperBound(int[] arr, int target) {
        int low = 0, high = arr.length - 1;
        int result = arr.length; // Default to the end of the array

        while (low <= high) {
            int mid = low + (high - low) / 2;

            if (arr[mid] > target) {
                result = mid; // Update result and move left
                high = mid - 1;
            } else {
                low = mid + 1;
            }
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Input the size of the array
        int n = sc.nextInt();
        int[] arr = new int[n];

        // Input the sorted array elements
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        // Input the target value
        int target = sc.nextInt();

        // Call the upperBound function
        int result = upperBound(arr, target);

        // Print the result
        if (result < n) {
            System.out.println("Upper Bound index: " + result);
        } else {
            System.out.println("No element > " + target + " found in the array.");
        }

        sc.close();
    }
}
```