Finding the First Occurrence/Last Occurrence of an Element in a Sorted Array

---

Problem Statement

Given a sorted array arr of size n find the index of the first occurrence of a target element x. If the element does not exist in the array, return -1.

Key Insight

- In a standard binary search, we stop as soon as we find the target.
- For finding the first occurrence, we:
    - Continue searching in the left half even after finding the target to ensure it is the first occurrence.

Algorithm

1. Initialize two pointers: low = 0 and high = n - 1.
2. Perform binary search:
    - Compute mid = low + (high - low) / 2.
    - If arr[mid] == x:
        - Record the index mid as a potential answer.
        - Continue searching in the left half by setting high = mid - 1.
    - If arr[mid] < x, search in the right half (low = mid + 1).
    - If arr[mid] > x, search in the left half (high = mid - 1).
3. If no occurrence is found, return -1.

```
input
7
10 20 30 40 50 50 60
50
Output
4
```

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

int findFirstOccurrence(const vector<int>& arr, int target) {
   int low = 0, high = arr.size() - 1;
   int result = -1;

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

      if (arr[mid] == target) {
         result = mid;  // Update result
         high = mid - 1; // Search left half for earlier occurrences
      } else if (arr[mid] < target) {
         low = mid + 1;
      } else {
         high = mid - 1;
      }
   }

   return result;
}

int main() {
   int n, target;
   cin >> n;

   vector<int> arr(n);
   for (int i = 0; i < n; i++) {
      cin >> arr[i];
   }

   cin >> target;

   int result = findFirstOccurrence(arr, target);

   if (result != -1) {
      cout result << endl;
   } else {
      cout << "Element not found in the array." << endl;
   }

   return 0;
}
```

Java

```
import java.util.Scanner;

public class FirstOccurrence {

    public static int findFirstOccurrence(int[] arr, int target) {
        int low = 0, high = arr.length - 1;
        int result = -1;

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

            if (arr[mid] == target) {
                result = mid; // Update result
                high = mid - 1; // Search left half for earlier occurrences
            } else if (arr[mid] < target) {
                low = mid + 1;
            } else {
```

```
            high = mid - 1;
         }
      }
      return result;
   }

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

      int n = sc.nextInt();
      int[] arr = new int[n];
      for (int i = 0; i < n; i++) {
         arr[i] = sc.nextInt();
      }

      int target = sc.nextInt();

      int result = findFirstOccurrence(arr, target);

      if (result != -1) {
         System.out.println(result);
      } else {
         System.out.println("Element not found in the array.");
      }
      sc.close();
   }
}
```

Finding last occurrence

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

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

// Function to find the last occurrence of the target
int findLastOccurrence(const vector<int>& arr, int target) {
   int low = 0, high = arr.size() - 1;
   int result = -1; // To store the last occurrence index

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

      if (arr[mid] == target) {
            result = mid;    // Update the result and move right to find the last occurrence
            low = mid + 1;
        } else if (arr[mid] < target) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return result;
}

int main() {
    int n, target;
    cin >> n;

    vector<int> arr(n);
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    cin >> target;

    int result = findLastOccurrence(arr, target);

    if (result != -1) {
        cout << "Element found at index " << result << endl;
    } else {
        cout << "Element not found in the array." << endl;
    }

    return 0;
}
```

java

```
import java.util.Scanner;

class LastOccurrenceBinarySearch {

    // Function to find the last occurrence of the target
    public static int findLastOccurrence(int[] arr, int target) {
        int low = 0, high = arr.length - 1;
        int result = -1; // To store the last occurrence index

        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == target) {
                result = mid;    // Update the result and move right to find the last occurrence
                low = mid + 1;
            } else if (arr[mid] < target) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return result;
    }

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

        int n = sc.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        int target = sc.nextInt();

        int result = findLastOccurrence(arr, target);

        if (result != -1) {
            System.out.println("Element found at index " + result);
        } else {
            System.out.println("Element not found in the array.");
        }

        sc.close();
    }
}
```

Comparison with First Occurrence

Aspect	First Occurrence	Last Occurrence
Search Direction	Move left on match (high = mid - 1)	Move right on match (low = mid + 1)
Condition to Return	First index where arr[mid] == x	Last index where arr[mid] == x
Commonality	Both use binary search logic.	Both use binary search logic.