**Technical Specifications for Software: «Binary Search»**

**Done By Rami Alshalabi   
SE (eng) 1-21**

## 1. Problem Specification

**1.1 Significant Time Costs:**  
Performing searches in an array can be time-consuming, especially in large datasets. Traditional linear search methods are inefficient, leading to increased time complexity.

## 2. Purpose and Goals of the System

**2.1 Purpose:**  
This program is designed to find the index of a given element in a sorted array using the efficient binary search algorithm.

**2.2 Goals:**

* Provide an accurate and efficient binary search mechanism.
* Minimize search execution time by leveraging the binary search algorithm.

**2.3 Functional Requirements:**

* **Binary Search:** Implement a binary search to locate an element in a sorted array.
* **Sorting Check:** Validate that the input array is sorted before performing the search.
* **User Input:**
  + Allow users to input an array of numbers.
  + Allow users to specify the value they want to search for.
* **Result Display:** Output the search results to the screen, including the index of the found element or an appropriate message if not found.
* **Input Validation:** Ensure the input array and search value are valid and handle any errors appropriately.

**2.4 Non-Functional Requirements:**

* **Programming Language:** JavaScript
* **Application Type:** Console application
* **Supported Operating Systems:** Windows 10

## 3. Input Constraints

* **Array Size:** Minimum of 1 element, maximum of 10 elements.

## 4. System Requirements

* **Personal Computer:** The program requires a personal computer with sufficient resources to run JavaScript-based applications.
* **Operating System:** Windows OS family, specifically tested on Windows 10.

## 5. Test Cases

| **Test No.** | **Input Array** | **Key** | **Expected Result** | **Actual Result** |
| --- | --- | --- | --- | --- |
| 1 | [25] | 25 | 0 (Element found at index 0) | 0 |
| 2 | [25, 55, 66] | 55 | 1 (Element found at index 1) | 1 |
| 3 | [34, 55, 73] | 0 | Element not found | Element not found |
| 4 | [11, 22, 33, 44] | 44 | 3 (Element found at index 3) | 3 |
| 5 | [55, 33, 44, 22, 11] | 11 | "Array not sorted" | "Array not sorted" |
| 6 | [-2, 0, 2, 4] | 2 | 2 (Element found at index 2) | 2 |
| 7 | null | 1 | "Array not entered" | "Array not entered" |
| 8 | [1, 4, 5, 6, 6, 9, 10] | 6 | 3 (Index of first occurrence) | 3 |
| 9 | [1, 3, 5, 7, 9, 11] | 7 | 3 (Element found at index 3) | 3 |
| 10 | [2, 4, 6, 8, 10, 12] | 4 | 1 (Element found at index 1) | 1 |
| 11 | ['a', 'b', 'c', 'd', 'e'] | 'b' | "Input array contains symbols. Please enter only numbers" | "Input array contains symbols. Please enter only numbers" |

## 6. Implementation Notes

### 6.1 Input Validation

* **Null or Empty Input:** Check if the input array is null or empty. If so, display an error message.
* **Numeric Validation:** Ensure all elements in the input array are numbers.
* **Sorting Check:** Verify that the input array is sorted in ascending order before performing the binary search.

### 6.2 Search Algorithm

* Implement the binary search algorithm, which works by dividing the search space in half at each step:
  + Compare the target value with the middle element.
  + If the target is equal to the middle element, return the index.
  + If the target is less than the middle element, repeat the search in the lower half.
  + If the target is greater, repeat in the upper half.
* Return the index of the found element or an appropriate message ("Element not found") if the target value is not present.

### 6.3 Edge Case Handling

* **Boundary Conditions:** Handle cases where the target value is located at the beginning or end of the array.
* **Duplicate Elements:** If there are duplicate values, return the index of the first occurrence.

### 6.4 Performance Considerations

* **Time Complexity:** The binary search algorithm should have a time complexity of O(log n), where n is the number of elements in the array.

### 6.5 Error Handling

* Provide clear error messages when invalid input is detected:
  + **Unsorted Array:** If the array is not sorted, prompt the user with an appropriate message.
  + **Non-Numeric Input:** If the input array contains non-numeric elements, ask the user to provide an array of numbers only.