

## Experiment No. 1

### Aim:

To write a menu-based program that stores roll numbers of students who attended a training program and allows searching using Linear and Binary Search algorithms.

### Objective:

- To understand and implement linear search on unsorted data.
- To understand and implement binary search on sorted data.
- To practice working with arrays and searching techniques.
- To develop a menu-driven approach for choosing between linear and binary search.

### Problem Statement:

#### Write Menu-Based Program

a) Write a program to store roll numbers of students in an array who attended a training program in random order.

Write a function for searching whether a particular student attended the training program or not, using **Linear Search**.

b) Write a program to store roll numbers of students in an array who attended the training program in sorted order.

Write a function for searching whether a particular student attended the training program or not, using **Binary Search**.

### Theory:

#### 1. Linear Search:

Linear search, also called sequential search, is the simplest searching algorithm. It checks each element in the array one by one, from the beginning to the end, to find the target element.

- **How it works:**  
Start at the first element of the array and compare it with the target value. If it matches, the search ends successfully. If not, move to the next element and repeat until either the element is found or the array ends.
- **Advantages:**
  - Simple to implement.
  - Does not require the data to be sorted.
- **Disadvantages:**
  - Inefficient for large data sets, as it may require checking every element.
  - Time complexity is  $O(n)$ , where  $n$  is the number of elements.

- **Example:**

Suppose we have the array: [34, 21, 56, 12, 78]

And we want to find 56.

Steps:

- Check 34 → Not 56
- Check 21 → Not 56
- Check 56 → Found! Return success.

If we search for 99, it will check all elements and conclude the element is not present.

## 2. Binary Search:

Binary search is a more efficient searching algorithm but works only on **sorted arrays**. It divides the search interval into halves to quickly narrow down the search space.

- **How it works:**

1. Start with the entire sorted array.
2. Find the middle element and compare it with the target.
3. If the middle element is the target, search ends successfully.
4. If the target is smaller than the middle element, discard the right half and continue searching in the left half.
5. If the target is larger, discard the left half and search the right half.
6. Repeat steps 2-5 until the element is found or the search space is empty.

- **Advantages:**

- Much faster than linear search for large, sorted datasets.
- Time complexity is  $O(\log n)$ , which means it reduces search space exponentially.

- **Disadvantages:**

- Requires data to be sorted.
- More complex to implement than linear search.

- **Example:**

Consider the sorted array: [12, 21, 34, 56, 78]

Search for 56:

- Middle element index = 2 (0-based), value = 34
- $56 > 34$ , so search right half: [56, 78]
- Middle element index = 3, value = 56 → Found!

If searching for 25:

- Middle = 34,  $25 < 34$ , search left half [12, 21]
- Middle = 21,  $25 > 21$ , search right half (empty) → Not found.

### **Algorithm:**

#### **Linear Search Algorithm:**

1. Input the number of students.
2. Input roll numbers in random order.
3. Input the roll number to be searched.
4. Traverse the array from the beginning.
5. If a match is found, display “Student attended the training.”
6. If not, display “Student did not attend the training.”

#### **Binary Search Algorithm:**

1. Input the number of students.
2. Input roll numbers in sorted order.
3. Input the roll number to be searched.
4. Set low = 0 and high = n - 1.
5. Repeat while low  $\leq$  high:
  - o Calculate mid = (low + high) / 2.
  - o If element at mid equals the target, return success.
  - o If target < element at mid, set high = mid - 1.
  - o Else, set low = mid + 1.
6. If not found, return failure.

### **Input:**

- o Number of students
- o Roll numbers of students
- o Roll number to be searched
- o User choice for Linear or Binary Search

**Note: Write input you are giving to your program**

### **Output:**

**Note: Write output of your program**

**Conclusion:**

Hence we have implemented basic searching techniques such as linear and binary search. It also emphasized the importance of sorting for binary search and how user-driven menus can make a program more interactive.