**Activity 4**

**Name:Nishanthini V**

**Batch No:2022-9615**

**Enroll number: EBEON0323765163**

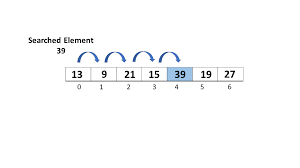
**SEARCHING ALGORITHM**

A searching algorithm is a procedure for finding a specific value, or a set of values, within a given data structure. The most common data structures used in searching algorithms are arrays, linked lists, trees, and graphs.

There are two main types of searching algorithms

* Sequential Search
* Binary Search

**Sequential Search**: This algorithm, also known as linear search, works by iterating through each element in a given data structure, such as an array or a list, and comparing it to the target value until a match is found or the end of the data structure is reached. Sequential search is simple and straightforward but can be slow for large data sets.



**Step 1**: First, read the search element (Target element) in the array.

**Step 2**: In the second step compare the search element with the first

element in the array.

**Step 3**: If both are matched, display “Target element is found” and

terminate the Linear Search function.

**Step 4**: If both are not matched, compare the search element with the next

element in the array.

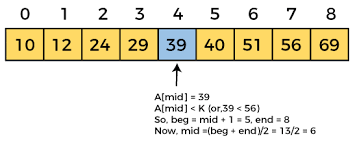
**Step 5**: In this step, repeat steps 3 and 4 until the search (Target) element

is compared with the last element of the array.

**Step 6**: If the last element in the list does not match, the Linear Search

Function will be terminated, and the message “Element is not found” will be displayed.

**Binary Search**: This algorithm works by dividing a sorted data structure, such as an array or a tree, into halves and repeatedly comparing the target value to the middle element of each half to determine which half to continue the search in. Binary search is much faster than sequential search for large data sets, but requires the data to be sorted beforehand.



**Step 1**: Sort the array in ascending order.

**Step 2**: Set the low index to the first element of the array and the high

index to the last element.

**Step 3**: Set the middle index to the average of the low and high indices.

**Step** **4**: If the element at the middle index is the target element, return the

middle index.

**Step 5**: If the target element is less than the element at the middle index,

set the high index to the middle index – 1.

**Step 6**: If the target element is greater than the element at the middle

index, set the low index to the middle index + 1.

**Step 7**: Repeat steps 3-6 until the element is found or it is clear that the

element is not present in the array.