Basic Searching Algorithms

Linear Search

Def :-

" Linear search, also known as Sequential Search, is a straight forward searching algorithm that checks each element in a list one by one until the desired element is found or the list ends. It is simple & works well for small or unsorted lists."

Advantages :-

- Simplest technique to implement.
- elements in the list can be in any order.

Disadvantages :-

- This method is inefficient when large numbers of elements are present in list because time taken for searching is more.

Step - by - Step Explanation :-

1. Start from the 1st element :-

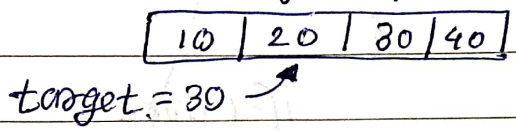
Compare the target element with the 1st element in the array.

0	1	2	3
10	20	30	40

target = 30.

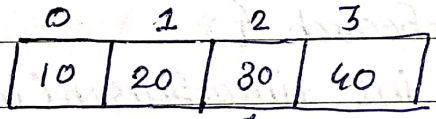
2. Move to the next element :-

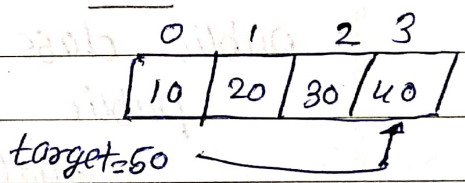
If the 1st element does not match, move to the next element & repeats the comparison.



3. Repeat unit found or end of array :-

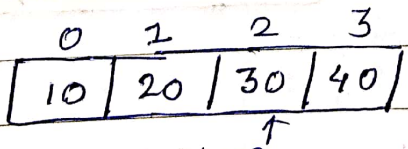
Continue this process until the target element is found or you reach the end of the array.

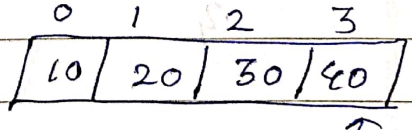




4) Return the result :-

If the element is found, return its index. If not, return -1 or similar indicator that the element is not in the array.





Linear Search code

```
public class LinearSearch {
    public static int linearSearch(int[] arr, int target)
        for(int i=0; i<arr.length; i++){
            if(arr[i] == target) {
                return i;
            }
        }
        return -1;
    }

    public static void main(String[] args){
        int[] arr = {10, 20, 30, 40, 50};
        int target = 30;

        int result = linearSearch(arr, target);

        if (result != -1){
            Sout("Element " + target + "Found at index:"
                + result);
        } else {
            Sout("Element " + target + " not Found");
        }
    }
}
```

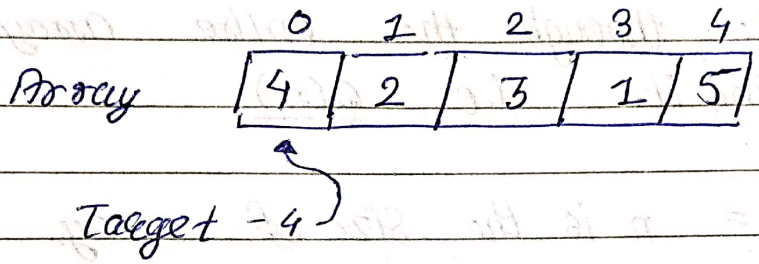
Element Found return index

Element not Found, return -1

Time complexity :-

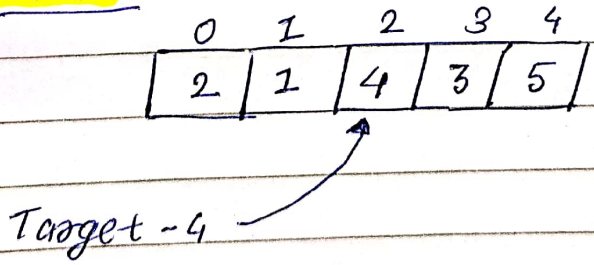
Best case :- O(1)
Average case :- O(n/2)
Worst case :- O(n)

* Best case :-



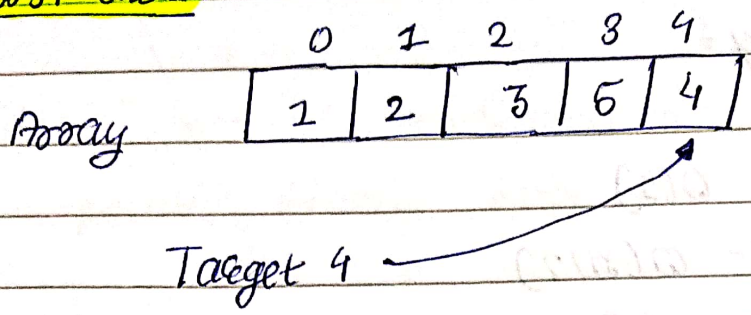
In this case, the target element (4) is at the beginning of the array, so we find it immediately. i.e O(1).

* Average case :-



In this case, the target element is (4) is in the middle of the array. We need to iterate through half of the array (2 elements) to find it. i.e O(n/2).

Woost Case :-



In this case, target element (4) is at very end of the array. We need to iterate through the entire array (5 elements) to find it. i.e O(n)

n = n is the size of array.

& also if element is not present in array. Time complexity for WOEST case is O(n).