Linear Search Algorithm

In this article, we will discuss the Linear Search Algorithm. Searching is the process of finding some particular element in the list. If the element is present in the list, then the process is called successful, and the process returns the location of that element; otherwise, the search is called unsuccessful.

Two popular search methods are **Linear Search** and **Binary Search**. So, here we will discuss the popular searching technique, i.e., **Linear Search Algorithm**.

Linear search is also called as **sequential search algorithm.** It is the simplest searching algorithm. In Linear search, we simply traverse the list completely and match each element of the list with the item whose location is to be found. If the match is found, then the location of the item is returned; otherwise, the algorithm returns NULL.

It is widely used to search an element from the unordered list, i.e., the list in which items are not sorted. The worst-case time complexity of linear search is **O(n).**

The steps used in the implementation of Linear Search are listed as follows -

- First, we have to traverse the array elements using a **for** loop.
- In each iteration of **for loop,** compare the search element with the current array element, and -
  - If the element matches, then return the index of the corresponding array element.
  - If the element does not match, then move to the next element.
- If there is no match or the search element is not present in the given array, return **-1.**

Now, let's see the algorithm of linear search.

Algorithm

1. Linear_Search(a, n, val) // 'a' is the given array, 'n' is the size of given array, 'val' is the value to search

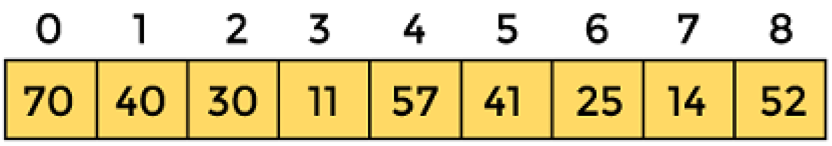
```
2.  Step 1: set pos = -1
3.  Step 2: set i = 1
4.  Step 3: repeat step 4 while i <= n
5.  Step 4: if a[i] == val
6.  set pos = i
7.  print pos
8.  go to step 6
9.  [end of if]
10. set i = i + 1
11. [end of loop]
12. Step 5: if pos = -1
13. print "value is not present in the array "
14. [end of if]
15. Step 6: exit
```

Working of Linear search

Now, let's see the working of the linear search Algorithm.

To understand the working of linear search algorithm, let's take an unsorted array. It will be easy to understand the working of linear search with an example.

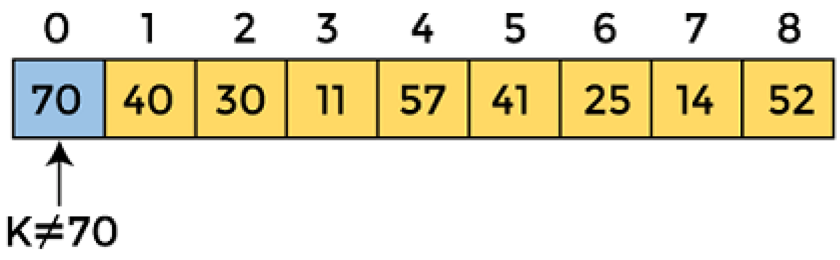
Let the elements of array are -



0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	52

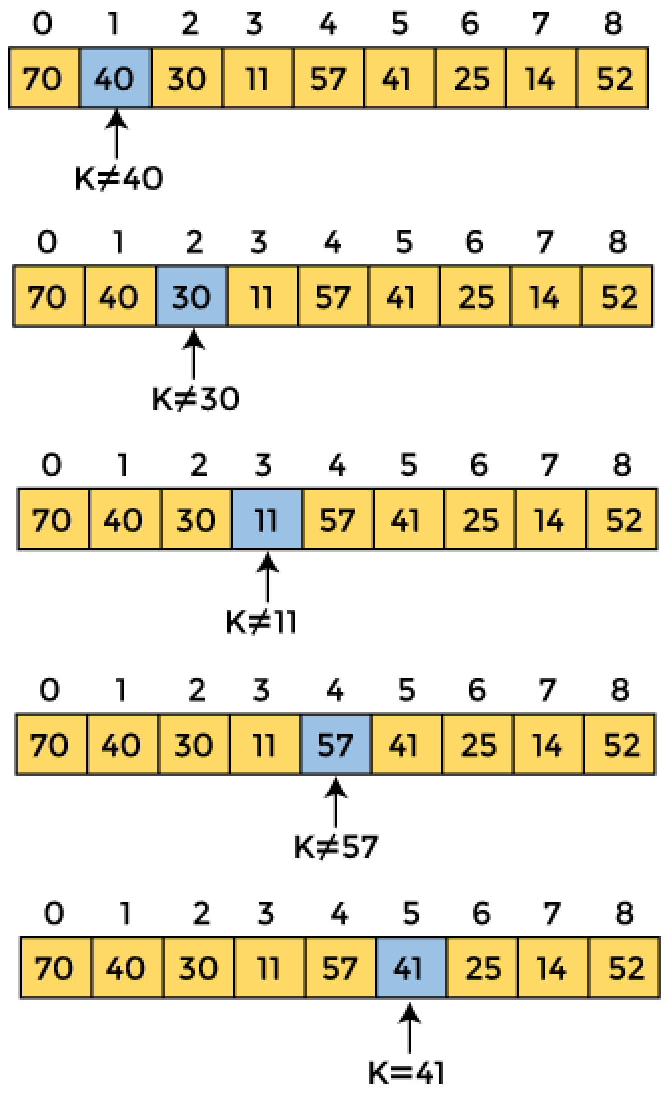
Let the element to be searched is **K = 41**

Now, start from the first element and compare **K** with each element of the array.



0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	52

The value of **K,** i.e., **41,** is not matched with the first element of the array. So, move to the next element. And follow the same process until the respective element is found.



Now, the element to be searched is found. So algorithm will return the index of the element matched.

Linear Search complexity

Now, let's see the time complexity of linear search in the best case, average case, and worst case. We will also see the space complexity of linear search.

1. Time Complexity

Case	Time Complexity
Best Case	O(1)

Average Case	O(n)
Worst Case	O(n)

- **Best Case Complexity -** In Linear search, best case occurs when the element we are finding is at the first position of the array. The best-case time complexity of linear search is **O(1).**
- **Average Case Complexity -** The average case time complexity of linear search is **O(n).**
- **Worst Case Complexity -** In Linear search, the worst case occurs when the element we are looking is present at the end of the array. The worst-case in linear search could be when the target element is not present in the given array, and we have to traverse the entire array. The worst-case time complexity of linear search is **O(n).**

The time complexity of linear search is **O(n)** because every element in the array is compared only once.

2. Space Complexity

Space Complexity	O

- The space complexity of linear search is O(1).

Implementation of Linear Search

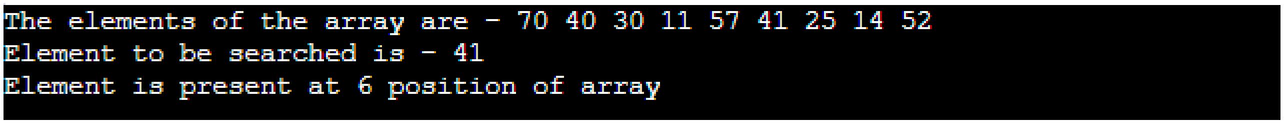
Now, let's see the programs of linear search in different programming languages.

Program: Write a program to implement linear search in C language.

```
#include <stdio.h>
int linearSearch(int a[], int n, int val) {
  // Going through array sequencially
  for (int i = 0; i < n; i++)
    {
       if (a[i] == val)
```

```
        return i+1;
    }
  return -1;
}
int main() {
  int a[] = {70, 40, 30, 11, 57, 41, 25, 14, 52}; // given array
  int val = 41; // value to be searched
  int n = sizeof(a) / sizeof(a[0]); // size of array
  int res = linearSearch(a, n, val); // Store result
  printf("The elements of the array are - ");
  for (int i = 0; i < n; i++)
  printf("%d ", a[i]);
  printf("\nElement to be searched is - %d", val);
  if (res == -1)
  printf("\nElement is not present in the array");
  else
  printf("\nElement is present at %d position of array", res);
  return 0;
}
```

Output

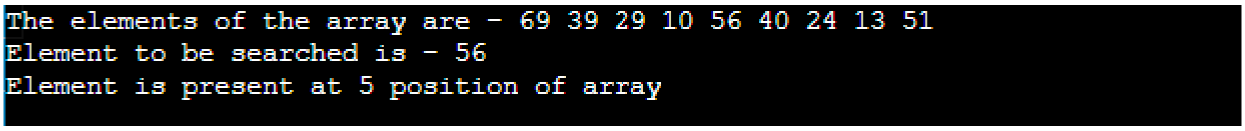


```
The elements of the array are - 70 40 30 11 57 41 25 14 52
Element to be searched is - 41
Element is present at 6 position of array
```

Program: Write a program to implement linear search in C++.

```
#include <iostream>
using namespace std;
int linearSearch(int a[], int n, int val) {
  // Going through array linearly
  for (int i = 0; i < n; i++)
    {
        if (a[i] == val)
        return i+1;
    }
  return -1;
}
int main() {
  int a[] = {69, 39, 29, 10, 56, 40, 24, 13, 51}; // given array
  int val = 56; // value to be searched
  int n = sizeof(a) / sizeof(a[0]); // size of array
  int res = linearSearch(a, n, val); // Store result
  cout<<"The elements of the array are - ";
  for (int i = 0; i < n; i++)
  cout<<a[i]<<" ";
  cout<<"\nElement to be searched is - "<<val;
  if (res == -1)
  cout<<"\nElement is not present in the array";
  else
  cout<<"\nElement is present at "<<res<<" position of array";
  return 0;
}
```

Output

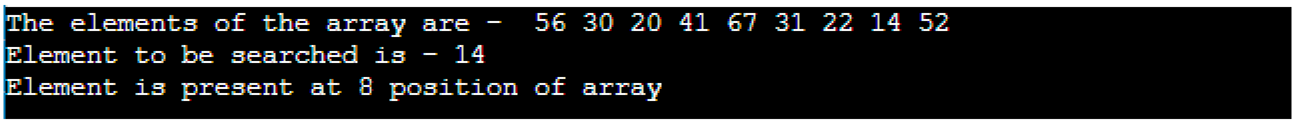


```
The elements of the array are - 69 39 29 10 56 40 24 13 51
Element to be searched is - 56
Element is present at 5 position of array
```

Program: Write a program to implement linear search in C#.

```
using System;
class LinearSearch {
static int linearSearch(int[] a, int n, int val) {
  // Going through array sequencially
  for (int i = 0; i < n; i++)
    {
        if (a[i] == val)
        return i+1;
    }
  return -1;
}
static void Main() {
  int[] a = {56, 30, 20, 41, 67, 31, 22, 14, 52}; // given array
  int val = 14; // value to be searched
  int n = a.Length; // size of array
  int res = linearSearch(a, n, val); // Store result
  Console.Write("The elements of the array are - ");
  for (int i = 0; i < n; i++)
  Console.Write(" " + a[i]);
  Console.WriteLine();
  Console.WriteLine("Element to be searched is - " + val);
  if (res == -1)
  Console.WriteLine("Element is not present in the array");
  else
  Console.Write("Element is present at " + res +" position of array");
}
}
```

Output



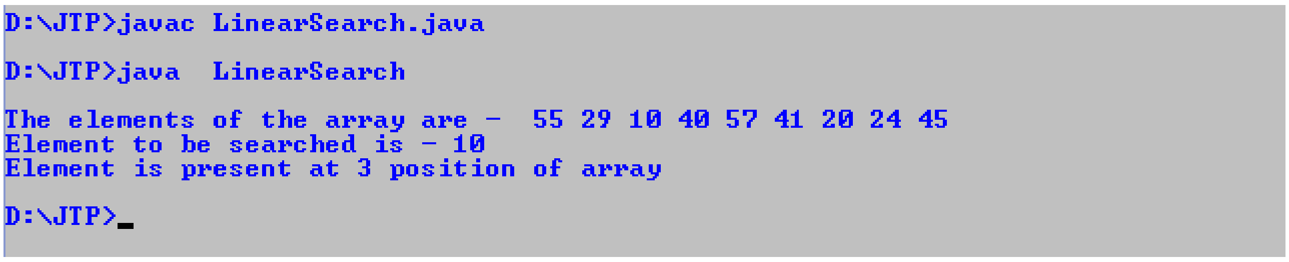
```
The elements of the array are -  56 30 20 41 67 31 22 14 52
Element to be searched is - 14
Element is present at 8 position of array
```

Program: Write a program to implement linear search in Java.

```
class LinearSearch {
static int linearSearch(int a[], int n, int val) {
  // Going through array sequencially
  for (int i = 0; i < n; i++)
    {
        if (a[i] == val)
        return i+1;
    }
```

```
 return -1;
}
public static void main(String args[]) {
 int a[] = {55, 29, 10, 40, 57, 41, 20, 24, 45}; // given array
 int val = 10; // value to be searched
 int n = a.length; // size of array
 int res = linearSearch(a, n, val); // Store result
 System.out.println();
 System.out.print("The elements of the array are - ");
 for (int i = 0; i < n; i++)
 System.out.print(" " + a[i]);
 System.out.println();
 System.out.println("Element to be searched is - " + val);
 if (res == -1)
 System.out.println("Element is not present in the array");
 else
 System.out.println("Element is present at " + res +" position of array");
}
}
```

Output



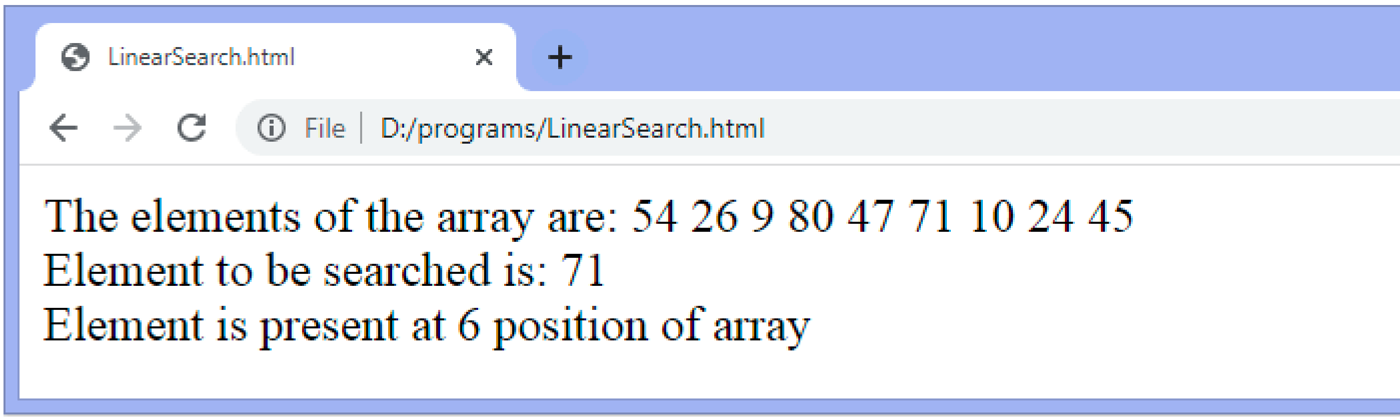
```
D:\JTP>javac LinearSearch.java

D:\JTP>java  LinearSearch

The elements of the array are -  55 29 10 40 57 41 20 24 45
Element to be searched is - 10
Element is present at 3 position of array

D:\JTP>_
```

Program: Write a program to implement linear search in JavaScript.

```
<html>
<head>
</head>
<body>
<script>
```

```
    var a = [54, 26, 9, 80, 47, 71, 10, 24, 45]; // given array
    var val = 71; // value to be searched
    var n = a.length; // size of array
    function linearSearch(a, n, val) {
    // Going through array sequencially
    for (var i = 0; i < n; i++)
    {
        if (a[i] == val)
        return i+1;
    }
    return -1
    }
   var res = linearSearch(a, n, val); // Store result
   document.write("The elements of the array are: ");
   for (i = 0; i < n; i++)
   document.write(" " + a[i]);
   document.write("<br>" + "Element to be searched is: " + val);
   if (res == -1)
   document.write("<br>" + "Element is not present in the array");
   else
   document.write("<br>" + "Element is present at " + res +" position of array");
    </script>
    </body>
</html>
```

Output



```
The elements of the array are: 54 26 9 80 47 71 10 24 45
Element to be searched is: 71
Element is present at 6 position of array
```

Program: Write a program to implement linear search in PHP.

```php
<?php
    $a = array(45, 24, 8, 80, 62, 71, 10, 23, 43); // given array
    $val = 62; // value to be searched
    $n = sizeof($a); //size of array
    function linearSearch($a, $n, $val) {
    // Going through array sequencially
    for ($i = 0; $i < $n; $i++)
    {
        if ($a[$i] == $val)
        return $i+1;
    }
    return -1;
    }
   $res = linearSearch($a, $n, $val); // Store result
   echo "The elements of the array are: ";
   for ($i = 0; $i < $n; $i++)
   echo " " , $a[$i];
   echo "<br>" , "Element to be searched is: " , $val;
   if ($res == -1)
   echo "<br>" , "Element is not present in the array";
   else
   echo "<br>" , "Element is present at " , $res , " position of array";
?>
```