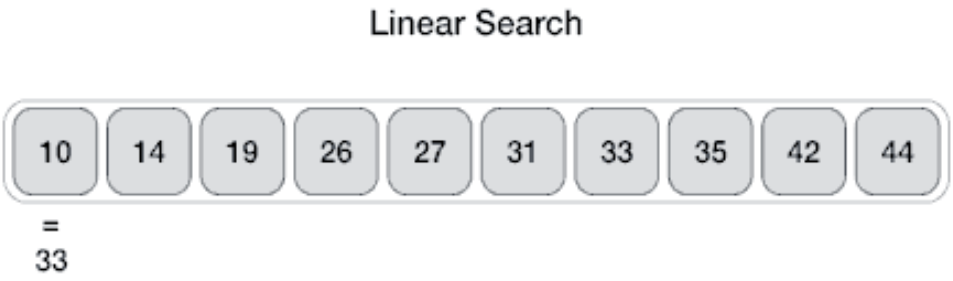
Linear Search Algorithm

Linear search is a type of sequential searching algorithm. In this method, every element within the input array is traversed and compared with the key element to be found. If a match is found in the array the search is said to be successful; if there is no match found the search is said to be unsuccessful and gives the worst-case time complexity.

For instance, in the given animated diagram, we are searching for an element 33. Therefore, the linear search method searches for it sequentially from the very first element until it finds a match. This returns a successful search.



In the same diagram, if we have to search for an element 46, then it returns an unsuccessful search since 46 is not present in the input.

Linear Search Algorithm

The algorithm for linear search is relatively simple. The procedure starts at the very first index of the input array to be searched.

Step 1 – Start from the 0th index of the input array, compare the key value with the value present in the 0th index.

Step 2 – If the value matches with the key, return the position at which the value was found.

Step 3 − If the value does not match with the key, compare the next element in the array.

Step 4 − Repeat Step 3 until there is a match found. Return the position at which the match was found.

Step 5 − If it is an unsuccessful search, print that the element is not present in the array and exit the program.

Pseudocode

```
procedure linear_search (list, value)
   for each item in the list
      if match item == value
         return the item's location
      end if
   end for
end procedure
```

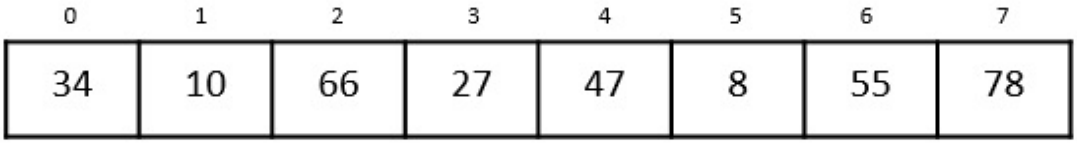
Analysis

Linear search traverses through every element sequentially therefore, the best case is when the element is found in the very first iteration. The best-case time complexity would be **O(1)**.

However, the worst case of the linear search method would be an unsuccessful search that does not find the key value in the array, it performs n iterations. Therefore, the worst-case time complexity of the linear search algorithm would be **O(n)**.

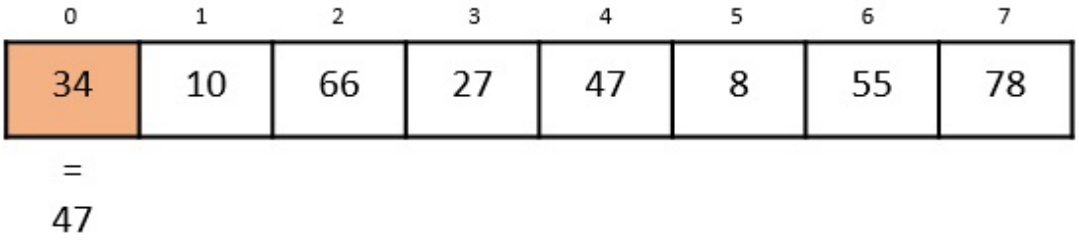
Example

Let us look at the step-by-step searching of the key element (say 47) in an array using the linear search method.



Step 1

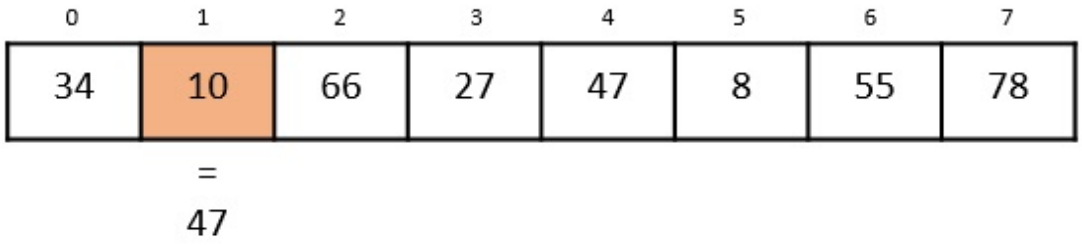
The linear search starts from the 0^{th} index. Compare the key element with the value in the 0^{th} index, 34.



However, 47 ≠ 34. So it moves to the next element.

Step 2

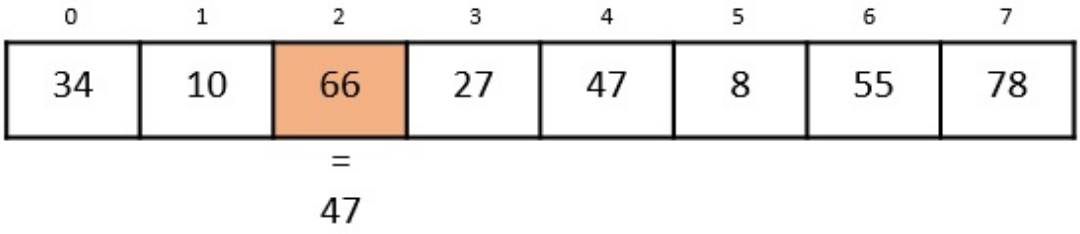
Now, the key is compared with value in the 1st index of the array.



Still, 47 ≠ 10, making the algorithm move for another iteration.

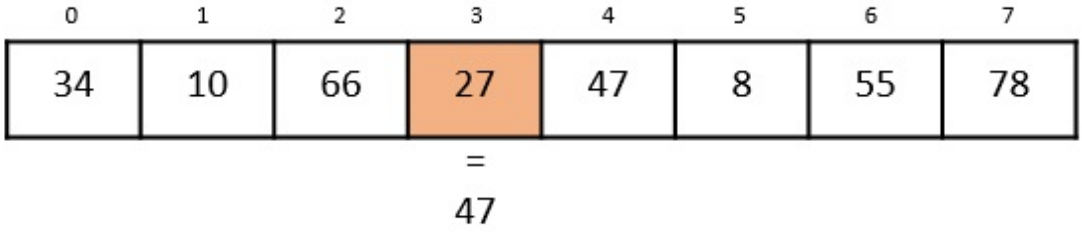
Step 3

The next element 66 is compared with 47. They are both not a match so the algorithm compares the further elements.



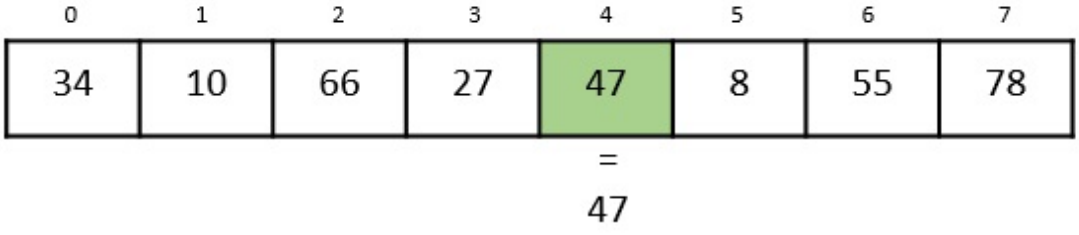
Step 4

Now the element in 3rd index, 27, is compared with the key value, 47. They are not equal so the algorithm is pushed forward to check the next element.



Step 5

Comparing the element in the 4^{th} index of the array, 47, to the key 47. It is figured that both the elements match. Now, the position in which 47 is present, i.e., 4 is returned.



The output achieved is “Element found at 4th index”.

Implementation

In this tutorial, the Linear Search program can be seen implemented in four programming languages. The function compares the elements of input with the key value and returns the position of the key in the array or an unsuccessful search prompt if the key is not present in the array.

C | C++ | Java | Python

```c
#include <stdio.h>
void linear_search(int a[], int n, int key){
   int i, count = 0;
   for(i = 0; i < n; i++) {
      if(a[i] == key) { // compares each element of the array
         printf("The element is found at %d position\n", i+1);
         count = count + 1;
      }
   }
   if(count == 0) // for unsuccessful search
      printf("The element is not present in the array\n");
}
int main(){
   int i, n, key;
   n = 6;
   int a[10] = {12, 44, 32, 18, 4, 10};
   key = 18;
   linear_search(a, n, key);
   key = 23;
   linear_search(a, n, key);
   return 0;
}
```

Output

```
The element is found at 4 position
The element is not present in the array
```