Current Skill Simple (linear) search

Simple linear search

Not even a single day pass, when we do not have to search for something in our day to day life, car keys, books, pen, mobile charger and what not. Same is the life of a computer, there is so much data stored in it that whenever a user asks for some data, computer has to search its memory to look for the data and make it available to the user.

Searching Algorithms are designed to check for an element or retrieve an element from any data structure where it is stored. Based on the type of search operation, these algorithms are generally classified into two categories:

1. Sequential Search: In this, the list or array is traversed sequentially and every element is checked. For example: Linear Search.
2. Interval Search: These algorithms are specifically designed for searching in sorted data structures. These types of searching algorithms are much more efficient than Linear Search as they repeatedly target the center of the search structure and divide the search space in half. For Example: Binary Search.

Linear search

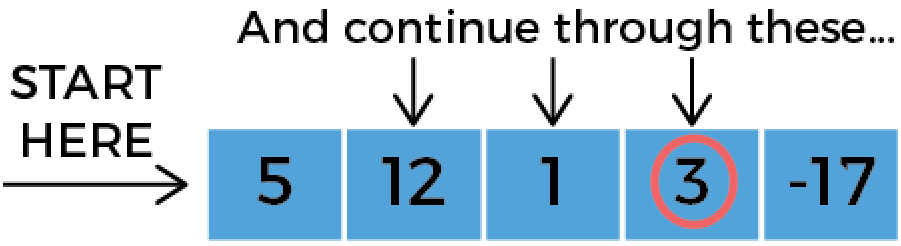
Linear search is a very basic and simple search algorithm. In linear search, we search an element or value in a given array by traversing the array from the starting, till the desired element or value is found.

It compares the element to be searched with all the elements present in the array and when the element is matched successfully, it returns the index of the element in the array, else it return -1.

Linear Search is applied on unsorted or unordered lists, when there are fewer elements in a list.

Features of Linear Search Algorithm

1. It is used for unsorted and unordered small list of elements.
2. It has a time complexity of O(n), which means the time is linearly dependent on the number of elements, which is not bad, but not that good either.
3. It has a very simple implementation.



Linear search

The time complexity of the linear search algorithm is O(n).

This is its implementation:

```
FUNCTION linear_search(tab : ARRAY_OF INTEGER, elt : INTEGER) : INTEGER
VAR
  j : INTEGER;
BEGIN
   j := 0;
   WHILE (j< tab.length) DO
      IF (tab[j] = elt) THEN
          RETURN j; // element is found let's break the loop and return the index
      END_IF
      j := j+1; // update the index
   END_WHILE
   // we reached the end of array without finding the element
   RETURN -1 ;// means that we did not find the element
END
```

‹ Previous

next ›