Searching Algorithm

What is Searching?

Searching is the fundamental process of locating a specific element or item within a collection of data. This collection of data can take various forms, such as arrays, lists, trees, or other structured representations. The primary objective of searching is to determine whether the desired element exists within the data, and if so, to identify its precise location or retrieve it. It plays an important role in various computational tasks and real-world applications, including information retrieval, data analysis, decision-making processes, and more.

What is Searching Algorithm?

Searching algorithms are essential tools in computer science used to locate specific items within a collection of data. These algorithms are designed to efficiently navigate through data structures to find the desired information, making them fundamental in various applications such as databases, web search engines, and more.

Searching terminologies:

Target Element:
In searching, there is always a specific target element or item that you want to find within the data collection. This target could be a value, a record, a key, or any other data entity of interest.

Search Space:
The search space refers to the entire collection of data within which you are looking for the target element. Depending on the data structure used, the search space may vary in size and organization.

Complexity:
Searching can have different levels of complexity depending on the data structure and the algorithm used. The complexity is often measured in terms of time and space requirements.

Deterministic vs. Non-deterministic:

Some searching algorithms, like binary search, are deterministic, meaning they follow a clear, systematic approach. Others, such as linear search, are non-deterministic, as they may need to examine the entire search space in the worst case.

Importance of Searching in DSA:

- **Efficiency:** Efficient searching algorithms improve program performance.
- **Data Retrieval:** Quickly find and retrieve specific data from large datasets.
- **Database Systems:** Enables fast querying of databases.
- **Problem Solving:** Used in a wide range of problem-solving tasks

Applications of Searching:

Searching algorithms have numerous applications across various fields. Here are some common applications:

- **Information Retrieval:** Search engines like Google, Bing, and Yahoo use sophisticated searching algorithms to retrieve relevant information from vast amounts of data on the web.
- **Database Systems:** Searching is fundamental in database systems for retrieving specific data records based on user queries, improving efficiency in data retrieval.
- **E-commerce:** Searching is crucial in e-commerce platforms for users to find products quickly based on their preferences, specifications, or keywords.
- **Networking:** In networking, searching algorithms are used for routing packets efficiently through networks, finding optimal paths, and managing network resources.
- **Artificial Intelligence:** Searching algorithms play a vital role in AI applications, such as problem-solving, game playing (e.g., chess), and decision- making processes
- **Pattern Recognition:** Searching algorithms are used in pattern matching tasks, such as image recognition, speech recognition, and handwriting recognition.

Linear Search Algorithm

The linear search algorithm is defined as a sequential search algorithm that starts at one end and goes through each element of a list until the desired element is found; otherwise, the search continues till the end of the dataset. In this article, we will learn about the basics of the linear search algorithm, its applications, advantages, disadvantages, and more to provide a deep understanding of linear search.

What is Linear Search Algorithm?

Linear search is a method for searching for an element in a collection of elements. In linear search, each element of the collection is visited one by one in a sequential fashion to find the desired element. Linear search is also known as **sequential search.**

Algorithm for Linear Search Algorithm:

The algorithm for linear search can be broken down into the following steps:

- **Start:** Begin at the first element of the collection of elements.
- **Compare:** Compare the current element with the desired element.
- **Found:** If the current element is equal to the desired element, return true or index to the current element.
- **Move:** Otherwise, move to the next element in the collection.

- **Repeat:** Repeat steps 2-4 until we have reached the end of collection.
- **Not found:** If the end of the collection is reached without finding the desired element, return that the desired element is not in the array.

How Does Linear Search Algorithm Work?

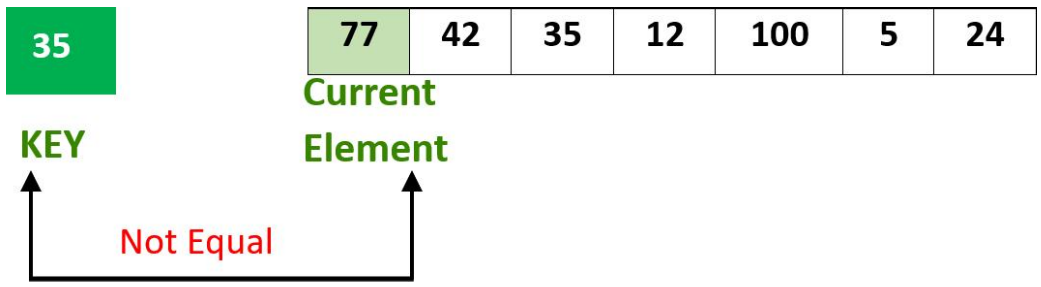
In Linear Search Algorithm,

- Every element is considered as a potential match for the key and checked for the same.
- If any element is found equal to the key, the search is successful and the index of that element is returned.
- If no element is found equal to the key, the search yields "No match found".

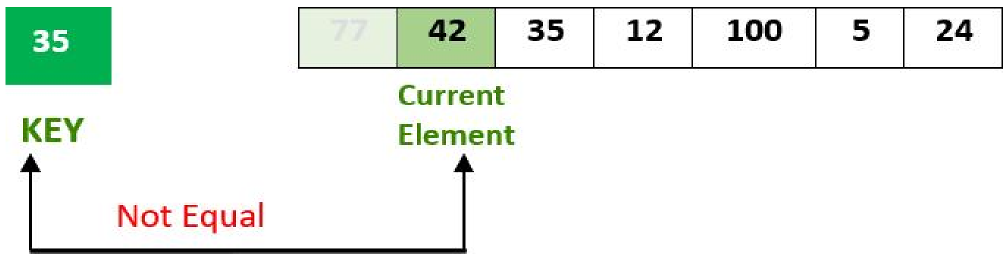
For example: Consider the array **arr[] = {77, 42, 35, 12, 100, 5, 24 }** and **key** = 30

Step 1: *Start from the first element (index 0) and compare **key** with each element (arr[i]).*

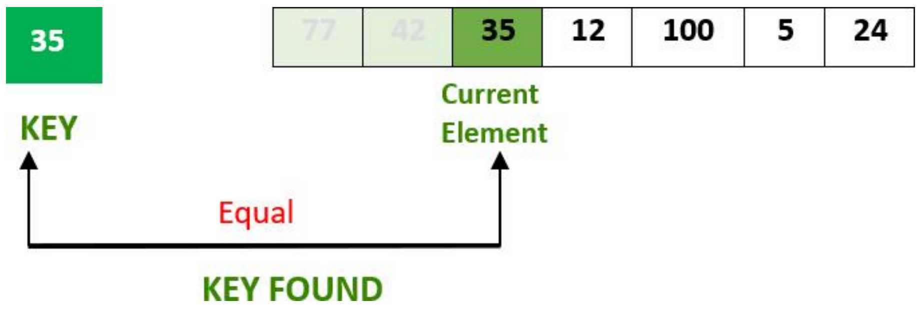
- *Comparing key with first element arr[0]. SInce not equal, the iterator moves to the next element as a potential match.*



- *Comparing key with next element arr[1]. SInce not equal, the iterator moves to the next element as a potential match.*



Step 2: *Now when comparing arr[2] with key, the value matches. So the Linear Search Algorithm will yield a successful message and return the index of the element when key is found (here 2).*



__Implementation of Linear Search Algorithm:__
In Linear Search, we iterate over all the elements of the array and check if it the current element is equal to the target element. If we find any element to be equal to the target element, then return the index of the current element. Otherwise, if no element is equal to the target element, then return -1 as the element is not found.

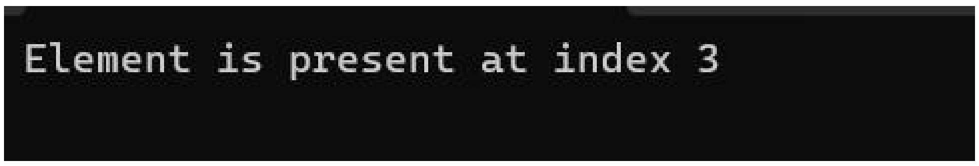
Implementing the linear search algorithm in C#

```
using System;

class LinearSearch {
    public static int search(int[] arr, int N, int x)
    {
        for (int i = 0; i < N; i++) {
            if (arr[i] == x)
                return i;
        }
        return -1;
    }

    // Driver's code
    public static void Main()
    {
        int[] arr = { 2, 3, 4, 10, 40 };
        int x = 10;

        // Function call
        int result = search(arr, arr.Length, x);
        if (result == -1)
            Console.WriteLine(
                "Element is not present in array");
        else
            Console.WriteLine("Element is present at index "
                              + result);
        Console.ReadLine();
    }
}
```



```
Element is present at index 3
```