# Introduction to Searching Algorithms

# 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. The same is the life of a computer, there is so much data stored in it, that whenever a user asks for some data, the computer has to search its memory to look for the data and make it available to the user.

# What if you have to write a program to search a given number in an array? How will you do it?

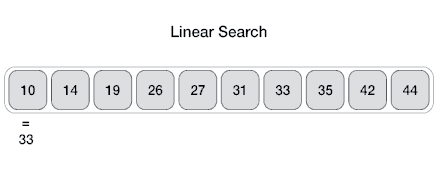
# Well, to search an element in a given array, there are two popular algorithms available:

# Linear Search

# 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 returns -1. Linear Search is applied on unsorted or unordered lists when there are fewer elements in a list.



## Implementing Linear Search

Following are the steps of implementation that we will be following:

1. Traverse the array using a for loop.
2. In every iteration, compare the target value with the current value of the array.
   * If the values match, return the current index of the array.
   * If the values do not match, move on to the next array element.
3. If no match is found, return -1.

We know you like Linear search because it is so damn simple to implement, but it is not used practically because binary search is a lot faster than linear search.

## Binary Search

# Binary Search is used with a sorted array or list.

## Implementing Binary Search Algorithm

Following are the steps of implementation that we will be following:

1. Start with the middle element:
   * If the target value is equal to the middle element of the array, then return the index of the middle element.
   * If not, then compare the middle element with the target value,
     + If the target value is greater than the number in the middle index, then pick the elements to the right of the middle index, and start with Step 1.
     + If the target value is less than the number in the middle index, then pick the elements to the left of the middle index, and start with Step 1.
2. When a match is found, return the index of the element matched.
3. If no match is found, then return -1

# Binary Search is useful when there is a large number of elements in an array and they are sorted. The binary search follows the divide and conquer approach.

Binary search is a fast search algorithm with run-time complexity of Ο(log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form.

### Complexity of algorithm

| **Complexity** | **Best Case** | **Average Case** | **Worst Case** |
| --- | --- | --- | --- |
| Time | O(1) | O(n) | O(n) |
| Space |  |  | O(1) |

## Binary search

## Complexity

| **SN** | **Performance** | **Complexity** |
| --- | --- | --- |
| 1 | Worst case | O(log n) |
| 2 | Best case | O(1) |
| 3 | Average Case | O(log n) |
| 4 | Worst-case space complexity | O(1) |