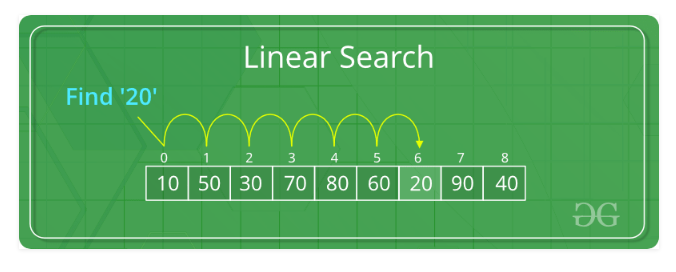
Searching

1. Linear Search
2. Start from the leftmost element of arr[] and one by one compare x with each element of arr[]
3. If x matches with an element, return the index.
4. If x doesn’t match with any of elements, return -1.



def search(arr, n, x):

    for i in range (0, n):

        if (arr[i] == x):

            return i

    return -1

# Driver Code

arr = [ 2, 3, 4, 10, 40 ]

x = 10;

n = len(arr);

result = search(arr, n, x)

if(result == -1):

    print("Element is not present in array")

else:

    print("Element is present at index", result)

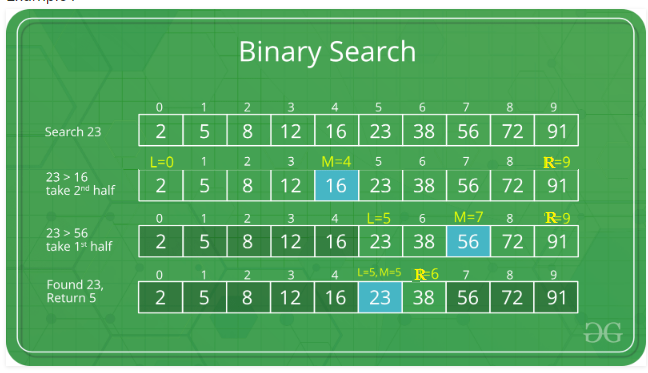
Output : Element is present at index 3

**Time Complexity:**  
The time complexity of Binary Search can be written as

T(n) = T(n)

The time complexity of above algorithm is O(n).

1. Binary Search
2. Search a sorted array by repeatedly dividing the search interval in half.
3. Begin with an interval covering the whole array.
4. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half.
5. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.



We basically ignore half of the elements just after one comparison.

1. Compare x with the middle element.
2. If x matches with middle element, we return the mid index.
3. Else If x is greater than the mid element, then x can only lie in right half subarray after the mid element. So we recur for right half.
4. Else (x is smaller) recur for the left half.

def binarySearch (arr, l, r, x):

    # Check base case

    if r >= l:

        mid = l + (r - l) // 2

        # If element is present at the middle itself

        if arr[mid] == x:

            return mid

        # If element is smaller than mid, then it can only be present in left subarray

        elif arr[mid] > x:

            return binarySearch(arr, l, mid-1, x)

        # Else the element can only be present in right subarray

        else:

            return binarySearch(arr, mid + 1, r, x)

    else:

        # Element is not present in the array

        return -1

# Driver Code

arr = [ 2, 3, 4, 10, 40 ]

x = 10

# Function call

result = binarySearch(arr, 0, len(arr)-1, x)

if result != -1:

    print ("Element is present at index % d" % result)

else:

    print ("Element is not present in array")

**Time Complexity:**  
The time complexity of Binary Search can be written as

T(n) = T(n/2) + c

The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Log n).