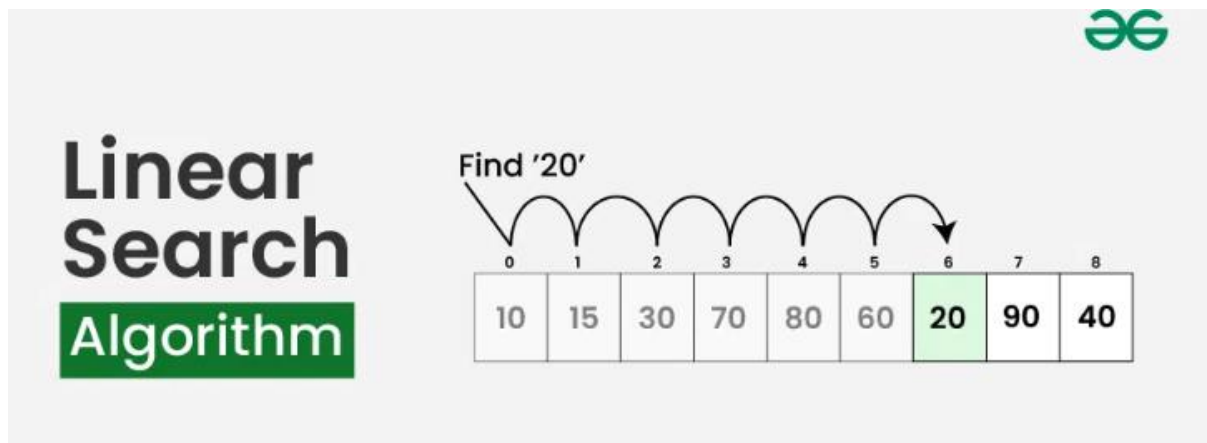


 Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the Searching Algorithms and understanding the time and space complexities	
Experiment No: 2	Date:	Enrolment No: 92200133023

Theory: (1)

Linear Search:

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.




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?

 Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the Searching Algorithms and understanding the time and space complexities	
Experiment No: 2	Date:	Enrolment No: 92200133023

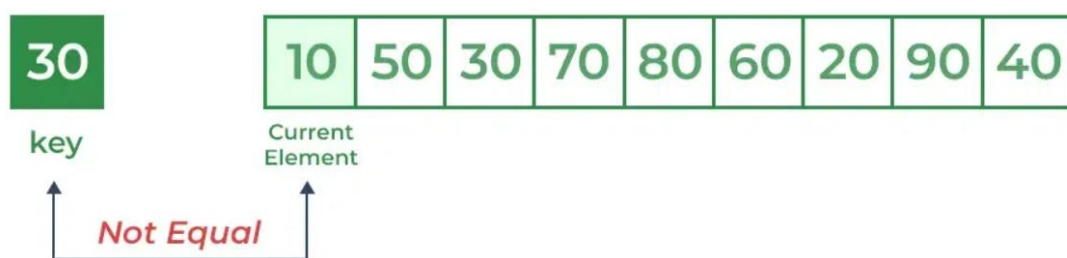
In Linear Search Algorithm,

- 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[] = {10, 50, 30, 70, 80, 20, 90, 40}` 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.

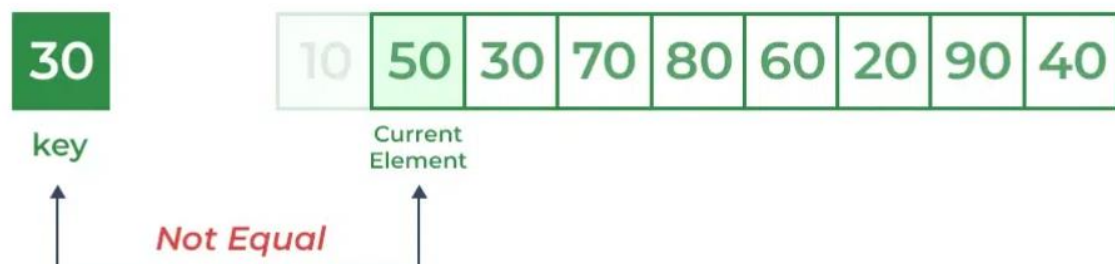


Linear Search Algorithm

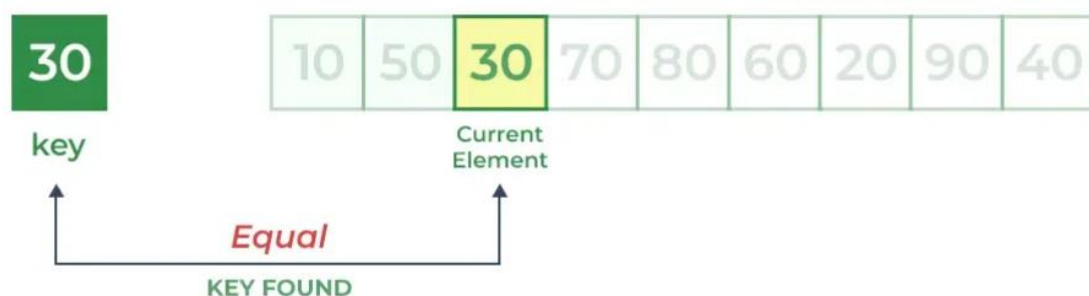


- Comparing key with next element `arr[1]`. Since not equal, the iterator moves to the next element as a potential match.

 Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the Searching Algorithms and understanding the time and space complexities	
Experiment No: 2	Date:	Enrolment No: 92200133023




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.

 Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the Searching Algorithms and understanding the time and space complexities	
Experiment No: 2	Date:	Enrolment No: 92200133023

Code:

```
def search(arr, N, x):

    for i in range(0, N):
        if (arr[i] == x):
            return i
    return -1

if __name__ == "__main__":
    arr = [2, 3, 4, 10, 40]
    x = 40
    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 4
```

Space complexity: _____

Justification: _____

 Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the Searching Algorithms and understanding the time and space complexities	
Experiment No: 2	Date:	Enrolment No: 92200133023

Time complexity:

Best case time complexity: _____

Justification: _____

Worst case time complexity: _____

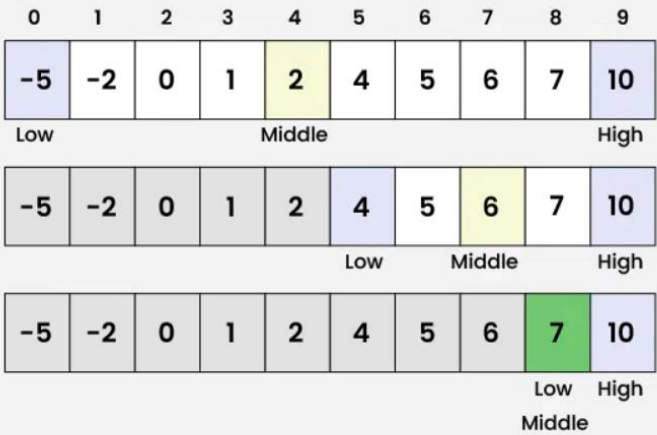
Justification: _____


Theory: (2)


Binary Search:

Binary Search Algorithm is a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to $O(\log N)$.

Binary Search Algorithm





 Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the Searching Algorithms and understanding the time and space complexities	
Experiment No: 2	Date:	Enrolment No: 92200133023

Conditions to apply Binary Search Algorithm in a Data Structure:

To apply Binary Search algorithm:

The data structure must be sorted.

Access to any element of the data structure should take constant time.

Binary Search Algorithm:


Below is the step-by-step algorithm for Binary Search:


- Divide the search space into two halves by finding the middle index “mid”.
- Compare the middle element of the search space with the key.
- If the key is found at middle element, the process is terminated.
- If the key is not found at middle element, choose which half will be used as the next search space.
- If the key is smaller than the middle element, then the left side is used for next search.
- If the key is larger than the middle element, then the right side is used for next search.
- This process is continued until the key is found or the total search space is exhausted.

How does Binary Search Algorithm work?

To understand the working of binary search, consider the following illustration:

Consider an array `arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}`, and the target = 23.

 Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the Searching Algorithms and understanding the time and space complexities	
Experiment No: 2	Date:	Enrolment No: 92200133023




Initially

Find Key = 23 using Binary Search

arr[] =

0	1	2	3	4	5	6	7	8	9
2	5	8	12	16	23	38	56	72	91




Step 1

Key (i.e., 23) is greater than current mid element (i.e., 16). The search space moves to the right.

Key
23

0	1	2	3	4	5	6	7	8	9
2	5	8	12	16	23	38	56	72	91

Low = 0
Mid = 4
High = 9

 Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the Searching Algorithms and understanding the time and space complexities	
Experiment No: 2	Date:	Enrolment No: 92200133023

Step 2

Key is less than the current mid 56.
The search space moves to the left.

Key
23

arr[] =

0	1	2	3	4	5	6	7	8	9
2	5	8	12	16	23	38	56	72	91

Low = 5 Mid = 7 High = 9

Step 3

If the key matches the value of the mid element,
the element is found and stop search.

Key
23

arr[] =

0	1	2	3	4	5	6	7	8	9
2	5	8	12	16	23	38	56	72	91

Low = 5 High = 9
Mid = 5

Algorithm:

 Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the Searching Algorithms and understanding the time and space complexities	
Experiment No: 2	Date:	Enrolment No: 92200133023

Programming Language:

Code:

```
def binarysearch(arr,key,l,r):
    if(l<=r):


        mid=int((l+r)/2)
        if(arr[mid]==key):
            return mid
        elif(arr[mid]<key):
            return (binarysearch(arr,key,mid+1,r))
        else:
            return (binarysearch(arr,key,l,mid-1))
    else:
        return -1

arr = [1,5,7,8,22,33,37]
key = 22
l=0
r=len(arr)-1
index = binarysearch(arr,key,l,r)

if index != -1:
    print("Element is present at index", index)
else:
    print("Element is not present in array")
```

Output:

```
Element is present at index 4
```

 Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the Searching Algorithms and understanding the time and space complexities	
Experiment No: 2	Date:	Enrolment No: 92200133023

Space complexity: _____

Justification: _____

Time complexity:

Best case time complexity: _____

Justification: _____

Worst case time complexity: _____

Justification: _____
