

Data Structure Notes

—/—/—

Searching :-

- Searching is the process of finding for something.
- Any search is said to be successful or unsuccessful depending upon whether the element that is being searched is found or not.
- It decides whether a search key is present in the data or not.
- It is the algorithmic process of finding a particular item in a collection of items.

Types of Searching :-

- ci) Linear Search
- cii) Binary Search

(1) Linear Search :-

- Linear search is also called as sequential search.
- Linear search starts at the beginning of the list and check every element of the list.

Example with Explanation:-

When I=1

Index	1	2	3	4	5	6	7	I	A[i]	A[i]=item	Result	LOC
Value	9	7	11	6	3	-10	15	1	9	9=3	FALSE	NULL

When I=2

Index	1	2	3	4	5	6	7	I	A[i]	A[i]=item	Result	LOC
Value	9	7	11	6	3	-10	15	2	7	7=3	FALSE	NULL

When I=3

Index	1	2	3	4	5	6	7	I	A[i]	A[i]=item	Result	LOC
Value	9	7	11	6	3	-10	15	3	11	11=3	FALSE	NULL

When I=4

Index	1	2	3	4	5	6	7	I	A[i]	A[i]=item	Result	LOC
Value	9	7	11	6	3	-10	15	4	6	6=3	FALSE	NULL

When I=5

Index	1	2	3	4	5	6	7	I	A[i]	A[i]=item	Result	LOC
Value	9	7	11	6	3	-10	15	5	3	3=3	TRUE	5

//_

program to search an element using
Linear Search

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a[5] = {10, 20, 30, 40, 50};
    int i = 0, item;
    printf("Enter searching item: ");
    scanf("%d", &item);

    while (i < 5)
    {
        if (a[i] == item)
        {
            printf("item found at location = %d", i);
            break;
        }
        i++;
    }
    if (i == 5)
        printf("item not found");
    getch();
}
```

Output:-
Enter searching item: 30
item found at location = 2

Linear Search Complexity :-

1. Time Complexity

Best Case = $O(1)$

Average Case = $O(n)$

Worst Case = $O(n)$

2. Space Complexity

Space Complexity $O(1)$

Binary Search:-

o In Linear Search algorithms have the problem to walk over the entire list. This problem can be eliminated by using some other searching technique that is Binary Search.

o Binary Search method can be used only for sorted lists

→ In this method the, the value of the element in the middle of the list is compared with the value of the element to be searched.

→
$$\text{Middle} = \text{int}((\text{LB} + \text{UB}) / 2)$$

(a) If it is a desired element, then search is successful.

(b) If it is greater than desired data, then search only the first half of the array (left side of the middle element).

(c) If it is less than the desired data, then search only the second half of the array (Right side of the middle element).

(Repeat the same step until an element is found)

//_

Algorithm (A, LB, UB, ITEM)

↑ ↑ ↑
Lower Upper data to be
Bound Bound searched
(BEGIN) (END)

Step 1:- $LB = 0$, $UB := n$ and $MID = \text{int}((LB + UB) / 2)$

Step 2 - Repeat Steps 3 and 4 while $(LB \leq UB)$ and $A[MID] \neq ITEM$.

Step 3:- If $ITEM < A[MID]$ then

(a) $UB = mid - 1$

Else

(b) $LB = mid + 1$

[End if]

Step 4:- $MID = \text{int}((LB + UB) / 2)$

Step 5:- If $A[MID] = ITEM$ then

(a) Display "the data found"

else

(b) Display "the data is not found"

Step 6:- Exit.

Example:-

0	1	2	3	4	5	6	7	8
10	12	24	29	39	40	51	56	69

element to search = 56

$mid = (LB + UB) / 2$
 $= (0 + 8) / 2 = 4$ mid of the array

0	1	2	3	4	5	6	7	8
10	12	24	29	39	40	51	56	69

A[mid] = 39

$A[mid] < item$
 $39 < 56$
 $LB = mid + 1 = 5$
 $UB = 8$

Now $mid = 5 + 8$
 $= 13 / 2 = 6$

0	1	2	3	4	5	6	7	8
10	12	24	29	39	40	51	56	69

A[mid] = 51

$A[mid] < item$
 $51 < 56$
 $LB = mid + 1 = 7$
 $UB = 8$

Now $mid = 7 + 8$
 $= 15 / 2 = 7$

0	1	2	3	4	5	6	7	8
10	12	24	29	39	40	51	56	69

↑

$$A[\text{mid}] = 56$$

$$A[\text{mid}] = \text{item}$$

$$\therefore 56 = 56$$

So, location = mid

(Element found at 7th location of the array)

* Complexity of Binary Search :-

1. Time Complexity :-

Best Case	$O(1)$
Average Case	$O(\log n)$
Worst Case	$O(\log n)$

2. Space Complexity

Space Complexity $O(1)$

//_

Program to search an element using [Binary Search]

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a[5] = {10, 20, 30, 40, 50};
    int lr = 0, up = 4, f = 0, mid, item;
    clrscr();
    printf("Enter searching item");
    scanf("%d", &item);
    while (lr <= up)
    {
        mid = (lr + up) / 2;
        if (a[mid] == item)
        {
            f = 1;
            break;
        }
        if (a[mid] < item)
        {
            lr = mid + 1;
        }
        else
        {
            up = mid - 1;
        }
    }
}
```



```
if (f == 1)
printf(" item found with location = %d", mid);
else
printf(" Not found");

getch();
```

Output:-

Enter Searching item: 4: 0
Item found with location = 3