Introduction to Searching :-

Searching is the process of finding a given value position in a list of values. It decides whether a search key is present in the data or not.
Searching is an operation or a technique that help find the place of a given element or value in the list. Any search is said to be successful or unsuccessful depending upon whether the element is found or not.

Standard Searching technique are-
1). Linear search or sequential search
2). Binary search.

1) Linear Search :-

This is a simplest method for searching. In this method we search element in linear way. This method can be performed on a sorted or an unsorted list.
In a linear data structure we search element from beginning and scan the element one by one until the end of array or linked list.
If search is successful then it will return the location of element, otherwise it will return the failure notification.
Suppose we use array as a data structure then the searching algorithm for array is -

Algorithm to search an element from array.

1. Start
2. array list [ ] = {- - - -}, i=0, f=0, item;
3. while (i < arraylist size)
4. if (arraylist[i] == item)
5. f=1 and break;
6. end if
7. i = i+1
8. End while.
9. if f == 1
   printf(" search succes at position = %d", i);
10. else
    printf(" search not succes)
11. Stop.

Ex. WAP to search an item from array

Soln-

```
void main()
{ int arrlist[10], i, f=0, item;
printf(" enter Array elements ");
for(i=0; i<10; i++)
  scanf(" %d", &arrlist[i]);
```

```
printf("\nEnter the value of item for search");
scanf("%d", &item);
while (i < 10)
{
    if (arlist[i] == item)
    {
        f=1;
        break;
    }
    i=i+1;
}
if (f==1)
printf("Search is successful at position
        =%d", i+1);
else
printf("Search is unsuccessful");
getch();
}
```

Similarly we can search item from linked list, stack and queue -

Features of Linear Search Algorithm.

(1) It is used for unsorted or, sorted and unordered small list of elements.

(2) It has a time complexity of O(n), which means the time is linearly dependent on the number of elements.

(3) It has a very simple implementation.

Binary Search

The Sequential search situation will be in worst case if the element is at the end of the list.
For eliminating this problem we have an efficient searching technique called binary search.
The Condition for binary search is that all the data should be in sorted array. We Compare the element with the middle element of the array. If it is less than the middle element then we search it in the left portion of the array and if it is greater than the middle element then search will be in the right portion of the array. Now we will take that portion only for search and Compare with middle element of that portion.
This process will be in iteration until we find the element or middle element has no left or right portion to search.

In binary search we will take 3 variables start, End and middle, which will keep track of the status of start End and middle value of the portion of the array, in which we will search the element. The value of middle will be
middle = (start + End)/2

Ex1. We take a sorted array of 10 elements and we want search 49 from this array -

arr

0	1	2	3	4	5	6	7	8	9
10	18	19	20	25	30	49	57	64	72

Sol^n Step 1 or Iteration 1 ->

Start = 0 End = 9, middle = (0+9)/2 = 4

arr

0	1	2	3	4	5	6	7	8	9
10	18	19	20	25	30	49	57	64	72

Now we will Compare middle element which is 25 with 49.

Since 49 > 25

Hence we will search the element in the right portion of the array.

Now Start = middle + 1 = 5, End = 9 (same)

Step 2 or Iteration 2 ->

Start = 5, End = 9, middle = (5+9)/2 = 7

arr

0	1	2	3	4	5	6	7	8	9
10	18	19	20	25	30	49	57	64	72

Now Compare 49 with 57.

Since 49 < 57

Hence we will search the element in the left portion of the array.

So End = middle = 7

Start = 5 (Same as earlier)

Step 3 or Iteration 3 :

Start = 5 , End = 7, middle = (5+7)/2 = 6

arr	0	1	2	3	4	5	6	7	8	9
	10	18	19	20	25	30	49	57	64	72

Now we will Compare middle element which is 49 with 49.

49 = 49 Hence element is found and Searching is successful.

Now we can say 49 found at array index 6 or at position 7.

Program for Binary Search.

```
Void main()
{
   int arr[20], start, end, middle, n, i, item;
   printf(" How many elements you want to enter
                in the array: ");
   Scanf("%d", &n);
printf("\nEnter Array elements \n");
   for(i=0; i<n; i++)
   {
      Scanf("%d", &arr[i]);
   }
   printf("\n Enter the element to be searched");
   Scanf("%d", &item);
```

```
start = 0; end = n-1; middle = (start+end)/2;
while( item != arr[middle] && start<=end)
{
    if (item > arr[middle])
        start = middle+1;
    else
        end = middle-1;
    middle = (start+end)/2;
}

if (item == arr[middle])
    printf(" %d is found at position %d \n",
                item, middle+1);

if (start > end)
    printf(" %d is not found in array\n", item);

getch();
}
```

Algorithm of Binary Search.

1. Start.
2. Take an array and required variables like int arr[20], start, end, middle n, i, item.
3. enter the value of n and enter n elements in the array after that enter value of item also.
4. Assign start = 0, end = n-1 and Calculate middle = (start+end)/2

```
(5) while(item != arr[middle] && start<=end)
(6)   if(item > arr[middle])
        start = middle+1;
(7)   else
        end = middle-1
(8)   middle = (start+end)/2
(9)   end while
(10)  if (item == arr[middle])
        printf(" search successful ");
(11)  if (start > end)
      printf(" search unsuccessful ");
(12)  Stop.
```

Time Space trade off.

A space-time or time space trade off is a way of solving a problem or calculation in less time by using more storage space or by solving a problem in very little space by spending a long time.

If your problem is taking a long time but not much memory then a space time trade off allow you that you can use more space to solve your problem quickly. or if it could be solved very quickly but require more memory than you have then you can try to spend more time to solve the problem in the limited memory.

Basic Ex1. More time less space.

```
Void main()
{
    int a, b;
    printf("Enter value of a");
    Scanf("%d", &a);
    printf("Enter value of b");
    Scanf("%d", &b);
    b = a+b;
    printf("sum=%d", b);
    getch();
}
```

Ex2. more space less time

```
Void main()
{
    int a, b, c;
    printf("Enter value of a and b");
    Scanf("%d %d", &a, &b);
    printf("sum=%d", c=a+b);
    getch();
}
```

Operations on data Structure

① **Traversing a data structure** → Traversing a data structure means "visiting" or "touching" the elements of the data structure and doing something with the data like searching an element or delete the data or insert the ~~dd~~ data etc.

Ex Suppose we have an array which store the RollNo of students

arr

0	1	2	3	4	5	6	7
15	20	25	26	27	28	29	30

if we want to search RollNo 28 then we have to traverse the element 15 to 27 then we get the element 28.

② **Insertion** → Insertion can be defined as the process of adding the elements of the data structure at any location.

③ **Deletion** → The process of removing an element from the data structure is called deletion.

④ **Searching** → The process of finding the location of an element within a data structure is called searching.

⑤ **Sorting** → The process of arranging the data structure in a specific order is known as sorting.

(6) Merging) When two list A and list B of size M and N respectively (of similar type of element), joined to produce the third list C of size M+N. Then this process is called merging.