Simple Programs:

Searching

Searching is a process of locating a particular element present in the given set of element. That element may be a record, a table, or a file.

The following are the simple searching techniques.

They are.

1. linear search
2. binary search.

Linear Search:

→ It is an sequential search.

→ The item to be searched is compared with all array elements sequentially.

eg:

0	1	2	3	4	5	6	7
15	5	20	32	2	42	67	17

n=8. data=42.

2 cases:

1. Element is present → Print the location
2. Not present → Not Print

```
for(i=0; i<n; i++)
{ if a[i] == data)
    {
      printf("element found at index %d", i);
      break;
    }
}
if(i==n)
{ printf("element not found");
}
```

Steps:

1. Let 'a' be an array of n items
2. x be the data to be searched.
3. Compare x with a[0], if equals print "Search success".
4. else, compare x with a[1], if equal print "Search success" and so on.
5. This process is repeated till the last element in the list.
6. If no array values matches print "search field"

Eg. Pgm:-

```
#include <stdio.h>
Void main()
{ int a[100], item, i, n;
  printf("enter the no. of elements : \n");
  Scanf("%d", &n);
  Printf("enter the elements :");
```

```
for (i=0; i<n; i++)
scanf("%d", &a[i]);
printf("Enter the item to search: \n");
scanf("%d", &item);
for (i=0; i<n; i++)
{ if(a[i]==item
  { printf("%d is present at location %d",
                item, i+1);
    break;
  }
}
if (i==n)
{ printf("%d is not present in array. \n", item
}
```

O/p:

enter the no. of elements:

5

enter the elements:

5
6
4
2
9

enter the item to search:

4

4 is present at location 3.

Binary Search

→ Binary Search is a algorithm used to find the 'position' of a target value within a sorted array.

→ It works by repeatedly dividing the search interval in half until the target value is found or the interval is empty.

→ The search interval is halved by comparing the target element with the middle value of the search space.

Steps

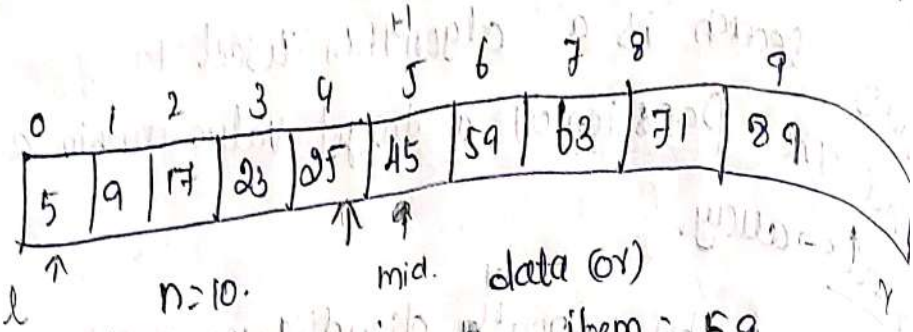
1. Let a be an array of sorted data.
2. Let x be the given data for searching
3. Find the mid position of the given array as

   mid = (lb + ub) / 2;

   where, lb → lower bound
   ub → upper bound of the array
4. Compare x with mid position value.
   if it equals searching over return mid.
5. If x is less than mid value, repeat steps (iii) and (iv) for the sub array 1 to mid -1.
6. If x is greater than mid value, repeat steps (iii) & (iv) for the sub array mid+1 to n.

7. If no value matches return ... failed

eg:



n=10. data (or) item = 59

l	r	mid
0	9	4
5	9	7
5	6	5
6	6	6

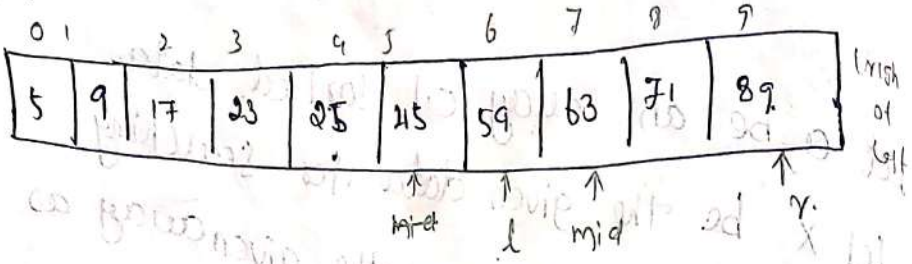
Logic:

case1: data == a[mid]

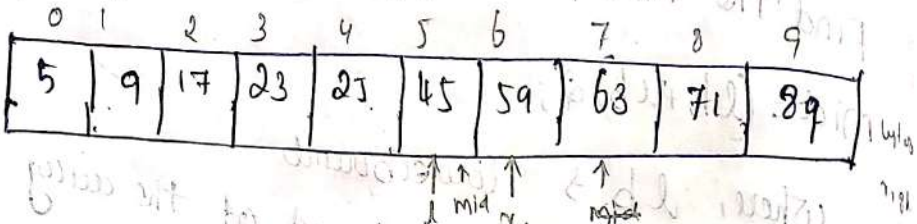
(or) case2: data < a[mid]

(or) case3: data > a[mid]

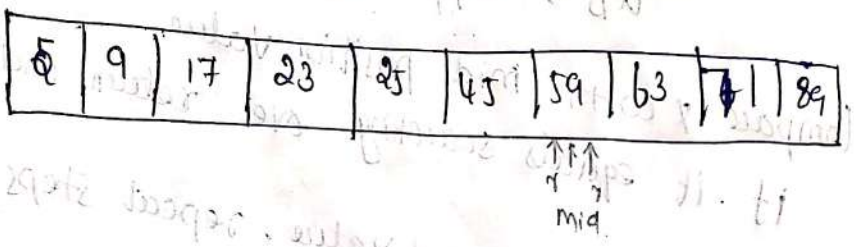
Step:1:



Step 2:

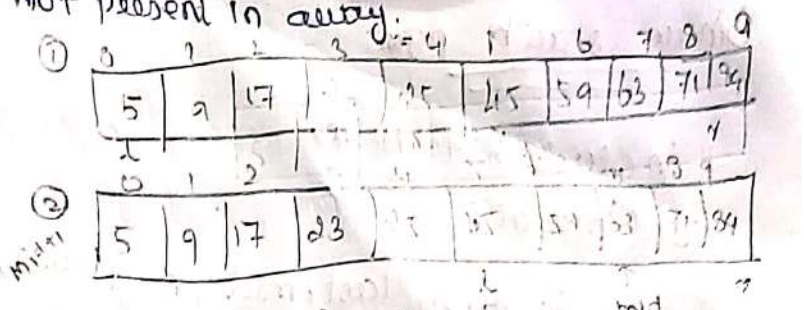


Step3:



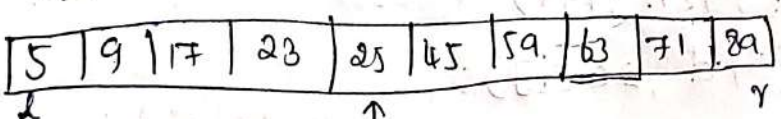
case 2: Data not present in array.

l	r	mid
0	9	4
5	9	7
5	6	5
6	6	6



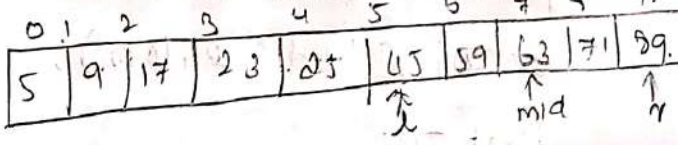
case 3: data = 60.

Stop any



l	r	mid
0	9	4
5	9	7
5	6	5
6	6	6
7	6	→ not present

if (l > r)



for Binarysearch

```
Binary search ( a, n, data)
   { l=0, r=n-1;
while(l<r)
   { mid = (l+r)/2;
   if (data == a[mid])
         return mid;
   else if (data < a[mid])
         r = mid -1;
   else  l = mid+1;
   } return -1;
   }
```

Binary search pgm:

```
#include <stdio.h>
Void main()
{ int i, first, last, mid, n, a[100], item;
 Printf(" Enter the no. of elements: \n");
 Scanf("%d ", &n);
 Printf ("enter the elements: \n");
 for( i=0 ; i<n ; i++)
 Scanf ("%d", &a[i]);
 Printf (" Enter the value to find: \n");
 Scanf ("%d", &item);
 first =0;
 last =n-1;
 mid = (first +last) /2 ;
 while (first <= last)
  {
   if (a[mid] < item)
    first = mid +1;
   else if (a[mid] == item)
   { Printf ("%d found at location %d \n",
                item, mid );
     break ;
   }
   else
     last = mid -1 ;
     mid = (first +last) /2;
   }
```

```
if (first > last)
printf("%d is not present in the list \n", item);
}.
```

o/p:

```
Enter the no. of elements:
7.
Enter the elements:
10
15
87
9
11
43
435
5
Enter the value to find:
11
11 found at location 5.
```

=

Sorting: