

1. Explain linear search and Binary Search.

Linear Search Algorithm

=====

Linear search, also called as sequential search, is a very simple method used for searching an array for a particular value. It works by comparing the value to be searched with every element of the array one by one in a sequence until a match is found.

LINEAR SEARCH(A, N, VAL)

```
Step 1: [INITIALIZE] SET POS = -1
Step 2: [INITIALIZE] SET I = 1
Step 3: Repeat Step 4 while I<=N
Step 4: IF A[I] = VAL
        SET POS = I
        PRINT POS
        Go to Step 6
    [END OF IF]
    SET I = I + 1
    [END OF LOOP]
Step 5: IF POS = -1
    PRINT VALUE IS NOT PRESENT IN THE ARRAY
    [END OF IF]
Step 6: EXIT
```

Binary search algorithm

=====

Binary search algorithm is a search algorithm that finds the position of searched value within the array. In the binary search algorithm, the element in the middle of the array is checked each time for the searched element to be found. If the middle element is not equal to the searched element, the search is repeated in the other half of the searched element. In this way, the search space is halved at each step.

BINARY_SEARCH(A, lower_bound, upper_bound, VAL)

```
Step 1: [INITIALIZE] SET BEG = lower_bound
        END = upper_bound, POS = - 1
Step 2: Repeat Steps 3 and 4 while BEG <= END
Step 3: SET MID = (BEG + END)/2
Step 4: IF A[MID] = VAL          SET POS = MID
PRINT POS
Go to Step 6      ELSE IF A[MID] > VAL
        SET END = MID - 1
        ELSE
        SET BEG = MID + 1
        [END OF IF]
        [END OF LOOP]
Step 5: IF POS = -1
        PRINT "VALUE IS NOT PRESENT IN THE ARRAY"
        [END OF IF]
Step 6: EXIT
```

2. Explain their complexity.

Linear Search

Best case-

In the best possible case,

The element being searched may be found at the first position.
In this case, the search terminates in success with just one comparison.
Thus in best case, linear search algorithm takes $O(1)$ operations.

Worst Case-

In the worst possible case,

The element being searched may be present at the last position or not present in the array at all.
In the former case, the search terminates in success with n comparisons.
In the later case, the search terminates in failure with n comparisons.
Thus in worst case, linear search algorithm takes $O(n)$ operations.

Binary search

In each iteration or in each recursive call, the search gets reduced to half of the array.

So for n elements in the array, there are $\log_2 n$ iterations or recursive calls.

Thus, we have-

Time Complexity of Binary Search Algorithm as $O(\log_2 n)$.

Here, n is the number of elements in the sorted linear array.

3. Implement Linear search.

```
//Linear Search

#include <iostream>
using namespace std;
void display_array(int array[],int length){
    for(int i=0;i<length;i++){
        cout<<array[i];
        (i==length-1)? cout<<".":cout<<",";
    }
}

int find_number(int array[],int length,int search_number){
    int found=-1;
    for(int i=0;i<length;i++){
        if(array[i]==search_number){
            found=i;
            break;
        }
    }
    return found;
}

int main(){
    int array[10]={12,34,21,39,65,48,36,11,58,23};
    int number,found;
    cout<<"\n Linear Search";
    cout<<"\n -----";
    cout<<"\n Elements in a Array ";
    display_array(array,10);
    cout<<"\n Enter Number to be Search : ";
    cin>>number;
    found = find_number(array,10,number);
    if(found>-1){
        cout<<"\n Number Found at index of "<<found<<endl;
    }else{
        cout<<"\n "<<number<<" is not Found in Array "<<endl;
    }
    cout << "Program developed by Umang 20U03031"<<endl;
}
```

```
d/DAA_ASS_2_Linear.cpp
1 //Linear Search
2 #include <iostream>
3 using namespace std;
4 void display_array(int array[],int length){
5     for(int i=0;i<length;i++){
6         cout<<array[i];
7         (i==length-1)? cout<<" ":cout<<" ";
8     }
9 }
10 int find_number(int array[],int length,int search_number){
11     int found=-1;
12     for(int i=0;i<length;i++){
13         if(array[i]==search_number){
14             found=i;
15             break;
16         }
17     }
18     return found;
19 }
20
21 int main(){
22     int array[10]={12,34,21,39,65,48,36,11,58,23};
23     int number,found;
24     cout<<"\n Linear Search";
25     cout<<"\n -----";
26     cout<<"\n Elements in a Array ";
27     display_array(array,10);
28     cout<<"\n Enter Number to be Search : ";
29     cin>>number;
30     found = find_number(array,10,number);
31     if(found!=-1){
32         cout<<"\n Number Found at index of "<<found<<endl;
33     }else{
34         cout<<"\n "<<number<<" is not Found in Array "<<endl;
35     }
36     cout << "Program developed by Umang 20U03031"<<endl;
37 }
```

state: charging
time to full: 34.8 minutes
percentage: 73%

```
~ 🐦 cd Library/ccode/daa
daa 🐦
daa 🐦 clang++ DAA_ASS_2_Linear.cpp -o linear
daa 🐦 ./linear
```

Linear Search

Elements in a Array 12,34,21,39,65,48,36,11,58,23.
Enter Number to be Search : 34

Number Found at index of 1
Program developed by Umang 20U03031

```
daa 🐦 ~/wai.sh
Name : Umang
Sch-No : 20U03031
```

daa 🐦

NORMAL daa/DAA_ASS_2_Linear.cpp cpp utf-8[unix] 2% ln:1/38=61

4. Implement Binary search with recursion.

//Recursive Binary Search

```
#include<iostream>
using namespace std;
void display_array(int array[],int length){
    for(int i=0;i<length;i++){
        cout<<array[i];
        (i==length-1)? cout<<".":cout<<",";
    }
}
//binary_search(array,length,search_number,lower,upper)
int binary_search(int array[],int length,int search_number,int lower,int upper){
    if(lower <= upper){
        int center = (lower+upper)/2;
        if(search_number == array[center]){
            return center;
        }else if(search_number < array[center]){
            return binary_search(array,length,search_number,lower,center-1);
        }else{
            return binary_search(array,length,search_number,center+1,upper);
        }
    }else{
        return -1;
    }
}
void sorting(int array[],int length){
    for(int i = length - 1; i >= 0; i--){
        for(int j = 1; j <= i; j++){
            if(array[j-1] > array[j]){
                //Swap
                int temp_A = array[j-1];
                array[j-1] = array[j];
                array[j] = temp_A;
            }
        }
    }
}
int main(){
    int array[10]={12,34,21,39,65,48,36,11,58,23};
    int number,index;
    int length=10;
    cout<<"\n Recursive Binary Search";
    cout<<"\n -----";
    cout<<"\n Un-Sorted Elements in a Array ";
    display_array(array,length);
    //Sorting
    cout<<"\n\n Recursive Binary Search is Works With Sorted Array.. ";
    sorting(array,length);
    cout<<"\n\n After Sorting Sorted Elements in a Array ";
    display_array(array,length);
    cout<<"\n Enter Number to be Search : ";
    cin>>number;
    int lower=0; // Left of Array Segment
```

```

int upper=length-1; // Right of Array Segment
index = binary_search(array,length,number,lower,upper);
if(index>-1){
    cout<<"\n Number Found at index of "<<index<<endl;
}else{
    cout<<"\n "<<number<<" is not Found in Array \n";
}
cout << "Program developed by Umang 20U03031"<<endl;
}

```

The screenshot shows a C++ IDE with the source code on the left and the program's output on the right. The source code implements a recursive binary search and a sorting function. The output shows the program's execution, including the array elements, the search process, and the final result.

```

d/DAA_ASS_2_Rec_Binary.cpp
1 //Recursive Binary Search
2 #include<iostream>
3 using namespace std;
4 void display_array(int array[],int length){
5     for(int i=0;i<length;i++){
6         cout<<array[i];
7         (i==length-1)? cout<<" ":cout<<" ";
8     }
9 }
10 //binary_search(array, length, search_number, lower, upper)
11 int binary_search(int array[],int length,int search_number,int lower,int upper){
12     if(lower <= upper){
13         int center = (lower+upper)/2;
14         if(search_number == array[center]){
15             return center;
16         }else if(search_number < array[center]){
17             return binary_search(array,length,search_number,lower,center-1);
18         }else{
19             return binary_search(array,length,search_number,center+1,upper);
20         }
21     }else{
22         return -1;
23     }
24 }
25 void sorting(int array[],int length){
26     for(int i = length - 1; i >= 0; i--){
27         for(int j = 1; j <= i; j++){
28             if(array[j-1] > array[j]){
29                 //Swap
30                 int temp_A = array[j-1];
31                 array[j-1] = array[j];
32                 array[j] = temp_A;
33             }
34         }
35     }
36 }
37 int main(){
38     int array[10]={12,34,21,39,65,48,36,11,58,23};
39     int number,index;
40     int length=10;
41     cout<<"\n Recursive Binary Search";
42     cout<<"\n -----";
43     cout<<"\n Un-Sorted Elements in a Array ";
44     display_array(array,length);
45     //Sorting
46     cout<<"\n\n Recursive Binary Search is Works With Sorted Array.. ";
47     sorting(array,length);
48     cout<<"\n\n After Sorting Sorted Elements in a Array ";
49     display_array(array,length);
50 }

```

Output:

```

Name : Umang
Sch-No : 20U03031
dAA clang++ DAA_ASS_2_Rec_Binary.cpp -o recBin
dAA ./recBin

Recursive Binary Search
-----
Un-Sorted Elements in a Array 12,34,21,39,65,48,36,11,58,23.

Recursive Binary Search is Works With Sorted Array..

After Sorting Sorted Elements in a Array 11,12,21,23,34,36,39,48,58,65.
Enter Number to be Search : 65

Number Found at index of 9
Program developed by Umang 20U03031
dAA ~/wai.sh
Name : Umang
Sch-No : 20U03031
dAA 

```

4. Implement Binary search without recursion.

```
//Iterative Binary Search
#include<iostream>
using namespace std;
void display_array(int array[],int length){
    for(int i=0;i<length;i++){
        cout<<array[i];
        (i==length-1)? cout<<".":cout<<" ";
    }
}
//binary_search(array,length,search_number,lower,upper)

int binarySearch(int arr[], int l, int r, int x)
{ while (l <= r)
    { int mid = l + (r - l) / 2;
      if (arr[mid] == x) {
          return mid;
      }
      if (arr[mid] < x) {
          l = mid + 1;
      } else {
          r = mid - 1;
      }
    }
    return -1;
}

void sorting(int array[],int length){
    for(int i = length - 1; i >= 0; i--){
        for(int j = 1; j <= i; j++){
            if(array[j-1] > array[j]){
                //Swap
                int temp_A = array[j-1];
                array[j-1] = array[j];
                array[j] = temp_A;
            }
        }
    }
}

int main(){

    int array[10]={12,34,21,39,65,48,36,11,58,23};
    int number,index;
    int length=10;
    cout<<"\n Recursive Binary Search";
    cout<<"\n -----";
    cout<<"\n Un-Sorted Elements in a Array ";
    display_array(array,length);
    //Sorting
    cout<<"\n\n Recursive Binary Search is Works With Sorted Array.. ";
    sorting(array,length);
    cout<<"\n\n After Sorting Sorted Elements in a Array ";
    display_array(array,length);
    cout<<"\n Enter Number to be Search : ";
    cin>>number;
    int lower=0; // Left of Array Segment
```

```

int upper=length-1; // Right of Array Segment

index = binarySearch(array,lower,upper,number);
if(index>-1){
    cout<<"\n Number Found at index of "<<index<<endl;
}else{
    cout<<"\n "<<number<<" is not Found in Array \n";
}
cout << "Program developed by Umang 20U03031"<<endl;
}

```

```

d/DAA_ASS_2_Itt_Binary.cpp
1 //Recursive Binary Search
2 #include<iostream>
3 using namespace std;
4 void display_array(int array[],int length){
5     for(int i=0;i<length;i++){
6         cout<<array[i];
7         (i==length-1)? cout<<" ":cout<<" ";
8     }
9 } //binary_search(array, length, search_number, lower, upper)
10
11 int binarySearch(int arr[], int l, int r, int x)
12 { while (l <= r)
13     { int mid = l + (r - l) / 2;
14       if (arr[mid] == x) {
15           return mid;
16       }
17       if (arr[mid] < x) {
18           l = mid + 1;
19       } else {
20           r = mid - 1;
21       }
22     }
23     return -1;
24 }
25 void sorting(int array[],int length){
26     for(int i = length - 1; i >= 0; i--){
27         for(int j = 1; j <= i; j++){
28             if(array[j-1] > array[j]){
29                 //Swap
30                 int temp_A = array[j-1];
31                 array[j-1] = array[j];
32                 array[j] = temp_A;
33             }
34         }
35     }
36 }
37 int main(){
38
39     int array[10]={12,34,21,39,65,48,36,11,58,23};
40     int number,index;
41     int length=10;
42     cout<<"\n Recursive Binary Search";
43     cout<<"\n -----";
44     cout<<"\n Un-Sorted Elements in a Array ";
45     display_array(array,length);
46     //Sorting
47     cout<<"\n\n Recursive Binary Search is Works With Sorted Array.. ";
48     sorting(array,length);
49     cout<<"\n\n After Sorting Sorted Elements in a Array ";
50
51 NORMAL daa/DAA_ASS_2_Itt_Binary.cpp      cpp utf-8[unix] 1% In:1/64-361

```

state: charging
time to full: 44.8 minutes
percentage: 74%
daa clang++ DAA_ASS_2_Itt_Binary.cpp -o ittBin
daa ./ittBin

Recursive Binary Search

Un-Sorted Elements in a Array 12,34,21,39,65,48,36,11,58,23.
Recursive Binary Search is Works With Sorted Array..
After Sorting Sorted Elements in a Array 11,12,21,23,34,36,39,48,58,65.
Enter Number to be Search : 21
Number Found at index of 2
Program developed by Umang 20U03031
daa ~/wai.sh
Name : Umang
Sch-No : 20U03031