IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING
(An Autonomous Institute affiliated to JNTUH, Hyderabad)
Dundigal, Hyderabad - 500 043

LABORATORY WORK BOOK

Name of the Student RAGHERLA SANTHOSH

Roll Number									
2	3	9	5	1	A	1	2	C	3

Class IT-B Semester 03

Course Code ACSD11 Course Name DS laboratory

Name of the Course Faculty Ms. K. Laxminarayanamma Faculty ID IARE10033

Exercise Number 02 Week Number 02 Date 03/09/2024

S. No.	Exercise Number	EXERCISE NAME	MARKS AWARDED					
			Aim/ Preparation	Algorithm / Procedure Performance in the Lab	Source Code Calculations and Graphs	Program Execution Results and Error Analysis	Viva - Voce	Total
			4	4	4	4	4	20
1	2.1	Linear Search						
2	2.2	Binary Search						
3	2.3	Uniform Binary Search						
4	2.4	Interpolation Search						
5	2.5	Fibonacci Search						
6								
7								
8								
9								
10								
11								
12			4	4	4	4	4	20

Santhosh

Signature of the Student

Signature of the Faculty

2. Searching.

2.1 Linear / Sequential Search :-

AIM :- Linear Search is defined as the Searching Algorithm where the list or data Set is traversed from one end to find the desired value. Given an array arr[] of n elements, Write a recursive function to Search a given element x in arr[].

PROGRAM :-

```
import java.util.Scanner;
class LinearSearch
{
  Public Static ~~void~~ ~~main~~ int linearSearch (int[] arr,
                                                    int Key)
  {
    for (int i=0; i< arr.length; i++)
    {
      if (arr[i] == Key)
      {
            return i;
          }
      }
      return -1;
}
Public static void main(String[] args)
{
  Scanner sc = new Scanner(System.in);
  System.out.print("Enter the size of array:");
  int size = sc.nextInt();
  int[] arr = new int[size];
  System.out.println("Enter the elements of array:");
  for (int i=0; i< size; i++)
  {
    arr[i] = sc.nextInt();
  }
  System.out.print("Enter the Key to Search:");
  int Key = sc.nextInt();
  int result = linearSearch(arr, Key);
```

```
if (result == -1)
{
  System.out.println ("Key not found in array");
}
else
{
  System.out.println ("Key found at index" +
                      result + " in the array");
}
Sc.close();
}
}
```

RESULT :-

INPUT : arr[] = {10, 20, 80, 30, 60, 50, 110, 100,
130, 170}

x = 110

OUTPUT : 6

Element x is present at Index 6.

2.2 Binary Search :-

AIM :- Binary Search is defined as 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).

PROGRAM :-

```
import java.util.Scanner;
import java.util.Arrays;
class Binary Search
{
  Public Static int binarySearch(int[] array, int Key)
  {
    int left = 0;
    int right = array.length -1;
    while (left <= right)
    {
      int  mid  =  left  + (right - left) / 2 ;
      if (array [mid]  == Key )
          return  mid ;
      if (array [mid]  < Key )
          left  = mid  + 1 ;
      else
          right  =  mid  - 1 ;
   }
   return  -1 ;
}
 Public  Static  void  main (String[] args )
 {
     Scanner   sc  =  new  Scanner (System.in);
     System. out. print (" Enter the Size of array :");
 int  Size  =  sc. nextInt ( );
 int []  arr  =  new  int [Size] ;
     System. out. println (" Enter the elements of array:");
 for ( int  i=0;  i< Size;  i++)
```

```
{
  arr[i] = sc.nextInt();
}
Arrays.sort (arr);
System.out.println(" After sorting : ");
  for (int i=0; i< size; i++)
  {
    System.out.println(arr[i]);
  }
  System.out.print (" Enter the Key to Search :");
  int Key = sc.nextInt();
  int result = binarySearch (arr, Key);
  if (result != -1)
    System.out.println (" Elements found at index " +
                              result);
 else
    System.out.println (" Element not found in the
  }                                array ");
}
```

RESULT : -

INPUT : arr = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]

OUTPUT : target = 23

Element 23 is present at Index 5.

2.3 Uniform Binary Search :-

AIM :- It is an optimization of Binary Search Algorithm when many Searches are made on Same array or many arrays of Same Size. In Normal Binary Search, We do arithmetic operations to find the mid points. Here we ~~Compute~~ Precompute mid points & fills them in lookup table. The array look-up generally works faster than arithmetic done (addition & Shift) to find the mid-point. It is used to modify the index of the Pointer in the array which makes the Search Faster.

PROGRAM :-

```
import java . util . Scanner ;
class  UniformBinarySearch
{
  private Static int[][] createLookupTable (int length)
  {
    int[][] lookupTable = new int [length][length] ;
    for (int i=0 ; i< length ; i++)
    {
      for (int j=i ; j< length ; j++)
      {
        lookupTable [i][j] = i+ (j-i)/2 ;
      }
  }
  Public Static int uniformBinarySearch (int [] arr, intx)
  {
    int length = arr. length ;
    int[][] lookupTable = createLookupTable (length) ;
    int left = 0 ;
    int right = length - 1 ;
    while ( left <= right)
  {
  int mid = lookupTable [left] [right] ;
  if (arr [mid] ) = = x )
  {
    return mid ;
  }
  else if (arr [mid] < x )
  {
    left = mid + 1 ;
  }
  else
  {
    right = mid - 1 ;
  }
}
 return -1 ;
}

Public Static void main ( String [] args)
{
  Scanner scanner = new Scanner ( System.in ) ;
```

```
System.out.println(" Enter the number of elements in
                the array : ");
int n = scanner.nextInt();
int[] arr = new int[n];
System.out.println(" Enter the elements of the array
                (sorted in ascending order) : ");
for (int i=0; i<n; i++)
{
  arr[i] = scanner.nextInt();
}
System.out.print(" Enter the value to search for:");
int x = scanner.nextInt();
int result = uniformBinarySearch(arr, x);
if (result != -1)
{
  System.out.println(" Position of " + x + " in array ="
                + result);
}
else
```

```
        {
        System.out.println(" Element" + x + " not found in
                                  the array");
        }
       Scanner.close();
      }
}
```

RESULT :-

INPUT : array = {1, 3, 5, 6, 7, 8, 9} v = 3

OUTPUT : position of 3 in array = 2.

2.4 Interpolation Search :-

AIM :- Interpolation Search go to different locations according to Search-Key. If the value of the Search-Key is close to the last element, Interpolation Search is likely to Start Search toward the end Side. Interpolation Search is more efficient than binary Search when the elements in the list are uniformly distributed. Interpolation Search can take longer to implement than binary search, as it requires the use of additional calculations to estimate the position of the target element.

PROGRAM :-

```
import java.util.Scanner;
class InterpolationSearch
{
  Public Static int interpolationSearch(int[] arr, int x)
  {
      int low = 0;
      int high = arr.length - 1;
      While (low <= high && x >= arr[low] &&
                x <= arr[high])
      {
        int pos = low + ((x - arr[low]) * (high - low) /
                          (arr[high] - arr[low]));
      if (arr[pos] == x)
      {
        return pos ;
      }
      if (arr [pos] < x )
      {
        low = pos + 1 ;
      }
      else
      {
        high = pas - 1 ;
      }
   }
    return -1 ;
 }
 Public Static void main (String [] args)
 {
   Scanner scanner = new Scanner ( System.in);
   System.out.print (" Enter the number of elements
                        in the array: ");
   int n = scanner.nextInt ( );
```

```
int[] arr = new int[n];
System.out.println("Enter the elements of array
                    (sorted in ascending order):");
for (int i=0; i<n; i++)
{
   arr[i] = scanner.nextInt();
}
System.out.print("Enter the value to search for:");
int x = scanner.nextInt();
int result = interpolationSearch(arr, x);
if (result != -1)
{
  System.out.println("Position of" + x + " in
                     array = " + result);
}
else
{
  System.out.println("Element" + x + " not
                     found in the array");
}
scanner.close();
}
}
```

RESULT :-

INPUT : arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]

OUTPUT : target = 5.

2.5 Fibonacci Search :-

AIM :- Given a Sorted array arr[] of Size n & an element x to be Searched in it. Return index of x if it is present in array else return -1.

PROGRAM :-

```
import java.util.Scanner;
class FibonacciSearch
{
 Private Static int[] fibonacciNumbers (int length)
 {
   int fibMm2 = 0;
   int fibMm1 = 1;
   int fibM = fibMm1 + fibMm2;
   While (fibM < length)
   {
```

```
            fibMm2  =  fibMm1 ;
            fibMm1  =  fibM ;
            fibM    =  fibMm1 + fibMm2 ;
        }
        return new int[] { fibM, fibMm1, fibMm2 };
    }
    Private Static int fibonacciSearch (int[] arr, int x )
    {
        int n = arr. length ;
        int [] fibs = fibonacciNumbers(n) ;
        int fibM = fibs [0] ;
        int fibMm1 = fibs [1];
        int fibMm2 = fibs [2];
        int offset = -1 ;
        while (fibM > 1)
        {
            int i = Math. min (offset + fibMm2 , n-1) ;
            if (arr [i] == x )
            {
                return i ;
            }
    else if (arr[i] < x)
    {
       fibm = fibMm1;
       fibMm1 = fibMm2;
       fibMm2 = fib M - fibMm1;
       Offset = i;
    } else {
       fibM = fibMm2;
       fibMm1 = fibMm1 - fibMm2;
       fibMm2 = fibm - fibMm1;
    }
 }
 if (fibMm1 == 1 && arr [offset + 1] == x)
 {
    return Offset +1;
 }
    return -1;
 }
Public Static void main (String[] args)
{
   Scanner Scanner = new Scanner (System.in);
   System.out.print ("Enter the number of elements
                     in the array:");
```

```
        int n = Scanner.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i=0; i<n ; i++) {
            arr[i] = Scanner.nextInt();
        }
        System.out.print ("Enter the value to search for:");
        int x = Scanner.nextInt();
        int result = fibonacciSearch (arr,x);
        if (result != -1) {
          System.out.println ("Position of"+ x +" in array ='+ result);
        } else {
          System.out.println ("Element" + x+" not found in array");
        }
        Scanner.close();
    }
}
```

RESULT :-

INPUT : arr[] = {2, 3, 4, 10, 40}, x = 10

OUTPUT : 3

Element x is present at index 3.

VIVA VOCE :-

1) What is Binary Search ?

A) Binary Search finds a value in a Sorted list by repeatedly halving the Search range. It's fast, with a time Complexity of O (log n).

2) Write the Difference between Linear & Binary Search ?

A) Linear Search :-

Approach : Checks each element one by one.

Time Complexity : O (n)

List Requirement : Works with both Sorted and unsorted lists.

Binary Search :-

Approach : Divides the Search range in half repeatedly.

Time Complexity : O (log n)

List Requirement : Requires a Sorted list.

3) What is Recursion ?

A) Recursion is when a function calls itself to Solve Smaller instances of the Same Problem. It has a base case to stop the recursion and a recursive case to break down the problem.

4) Write the steps Involved In Linear Search ?

A) (1) Start at the Beginning : Begin with the first element of the list.

(2) Compare : Check if the current element is equal to the target Value.

(3) Match : If the current element matches the target, return its Index.

(4) Move to Next : If the current element does not match, move to the next element.

(5) Repeat : Continue steps 2-4 until the target is found or the end of the list is reached.

(6) END : If the end of the list is reached without finding the target, return an indication that the target is not present.

(e.g. -1 (or) null) .