<u>Linear Search:</u>

```
package linearsearch;
public class Linearsearch {
   public static int linearsearch(int arr[],int key){
      int n=arr.length;
      for(int i=0;i<n;i++){
         if(arr[i]==key);
         return i;
      }
      return -1;
   }
   public static void main(String[] args) {
      int arr[]={10,20,30,40,50,60};
      int key=40;
      int result=linearsearch(arr,key);
      if(result==-1)
         System.out.println("Element Not Found");
      else
         System.out.println("Element Found at Index:"+result);
  }
}
```

<u>Time complexity:</u>

Best Case: Best case means the array is already sorted. So loop run for one time. There for time complexity is O (1).

Worst Case: In worst is in last index. So whole loop need to run. Therefor time complexity is O(n).

Average Case:

$\frac{1+2+3+\cdots+n}{n} = \frac{(n(n-1))/2}{n}$

$= \frac{n-1}{2}$

By avoiding all constant, time complexity is O(n);

<u>Binary Search:</u>

```
package binareysearch;

public class BinareySearch {

 int binearySearch(int arr[],int l, int h,int key){

      while(l<=h){

          int mid=(l+h)/2;

          if(arr[mid]==key)

            return mid;

          if(arr[mid]>key)

             l=mid-1;

          else

             l=mid;

         }

       return -1;

   }

   public static void main(String[] args) {

        BinareySearch obj = new BinareySearch();

       int arr[]={10,20,30,40,50,60};

       int n=arr.length;

       int key=60;

       int result=obj.binearySearch(arr,0,n,key);

       if(result==-1)

          System.out.println("Element not Found");
```

```
    else
        System.out.println("Element Found at " + "index " + result)
  }
  }
```

Time complexity:

Time Complexity:

In this algorithm total time is divided by 2 recursively.
Lets, total time is n.

T(n)= $\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+\cdots+1$

T(n)=$\frac{n}{2^1}+\frac{n}{2^2}+\frac{n}{2^3}+\cdots+1$

So, $\frac{n}{2^x}=1$

x=n log n

Best Case: O(1) [when array is already sorted]

Average Case: $\frac{\text{nlog n}}{\text{n}}=\log \text{n}$

Worst Case: O(1)

Practice Problem

01. Write a program that find the last occurrence of a given number in a sorted array.

```
package lastoccurence;

import java.util.Scanner;

public class LastOccurence {

public static void main(String[] args) {

    int i;

    int Ar[]={1,2,3,3,3,4,4,5};

    int n=Ar.length;

    Scanner obj=new Scanner(System.in);

    System.out.print("Key=");

    int key=obj.nextInt();
```

```
            System.out.println("");
            for(i=0;i<n;i++)
               if(Ar[i]>key){
                   System.out.println("Last Occurrence="+Ar[i]);
               break;
               }
                else
                System.out.println("Item not found");
                }
        }
```

2. Suppose you go to the supermarket, you have to press the code number to find a product, you can find the product position by pressing the product code. Write a search program (You have the product Code).

```
package positiion;
import java.util.Scanner;
 public class Positiion {
public static void main(String[] args) {
            // TODO code application logic here
            int i;
            int Ar[]={202,104,304,223,553,123};
            int n=Ar.length;
            Scanner obj=new Scanner(System.in);
            System.out.print("Search=");
            int Search=obj.nextInt();
            System.out.println("");
            for(i=0;i<n;i++){
               if(Ar[i]==Search){
                   System.out.println("Position="+i);
```

```
                }
              }
                else
                System.out.println("Product not found");
            }
          }
```

3. Suppose you and your friend are talking about Searching Algorithm, now one algorithm takes more time to solve one problem and another takes less time. So which algorithm is better? Choose the right algorithm & Write the program [Time Complexity O (n), O (log n)].

```
package binarysearch;
import java.util.Scanner;
public class Binarysearch {
  int binarysearch(int arr[],int l,int h,int key){
    while(l<=h){
      int mid=(l+h)/2;
      if(arr[mid]==key)
        return mid;
      if(arr[mid]>key)
        l=mid-1;
      else
        l=mid;
     }
    return -1;

  }
```

```
    public static void main(String[] args) {
        int i,n,l=0;
        Scanner obj=new Scanner(System.in);
        System.out.print("Enter the Size of Array:");
        n=obj.nextInt();
        System.out.println("");
        int arr[]=new int[n];
        System.out.print("Enter the Element:");
        for(i=0;i<n;i++){
            arr[i]=obj.nextInt();
        }
        System.out.println("");
        System.out.print("Enter the Key:");
       int key=obj.nextInt();
        int result=obj.binarysearch(arr,l,n-1,key);
        if(result==-1)
            System.out.println("Not Found");
        else
            System.out.println("Found at "+result+" position");


    }


}
```