

# Design and Analysis of Algorithm Lab7

Name: V. Prasanna Vyshnavi

Regno: 19BCE7661

## Binary Search:

### Code:

```
import java.util.*;

class BinarySearch{

public static void binarySearch(int arr[], int first, int last, int key){

    int mid = (first + last)/2;

    while( first <= last ){

        if ( arr[mid] < key){

            first = mid + 1;

        }else if ( arr[mid] == key){

            System.out.print(" "+mid);

            break;

        }

    }

    else{

        last = mid - 1;

    }

    mid = (first + last)/2;

}

if ( first > last ){

    System.out.print(" -1");

}
```

```

    }
}
public static void main(String args[]){
    Scanner sc=new Scanner(System.in);

    int n=sc.nextInt();
    int arr[] = new int[n];
    for(int i=0;i<n;i++)
    {
        arr[i]=sc.nextInt();
    }
    int m=sc.nextInt();
    int key[]=new int[m];
    for(int i=0;i<n;i++)
    {
        key[i]=sc.nextInt();
    }
    int last=n-1;
    System.out.println("output :");
    for(int i=0;i<m;i++)
    {
        int k=key[i];
        binarySearch(arr,0,last,k);
    }
}

```

```
}
```

## Output:

```
Administrator: cmd
C:\Users\Personal\Downloads\5th sem>javac BinarySearch.java
C:\Users\Personal\Downloads\5th sem>java BinarySearch
5
1
5
8
12
13
5
8
1
23
1
11
output :
 2 0 -1 0 -1
C:\Users\Personal\Downloads\5th sem>_
```

## Asymptotic Analysis:

Binary Search:-

BinarySearch (A, low, high, Key)

if high < low:

return low-1

mid  $\leftarrow \left[ \text{low} + \frac{\text{high} - \text{low}}{2} \right]$

if Key = A[mid]:

return mid

else if Key < A[mid]:

return BinarySearch(A, low, mid-1, Key)

else

return BinarySearch(A, mid+1, high, Key)

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

↳ one condition only

↓ Satisfied in code

$$= T\left(\frac{n}{4}\right) + 1 + 1$$

$$= T\left(\frac{n}{8}\right) + 1 + 1 + 1$$

$$= T\left(\frac{n}{2^k}\right) + 1 + 1 + \dots + 1$$

$$n = 2^k$$

$$= T(1) + k = 1 + \log_2 n$$

$$T(n) = O(\log_2 n)$$

## Maximum Votes:

### Code:

```
import java.util.*;
```

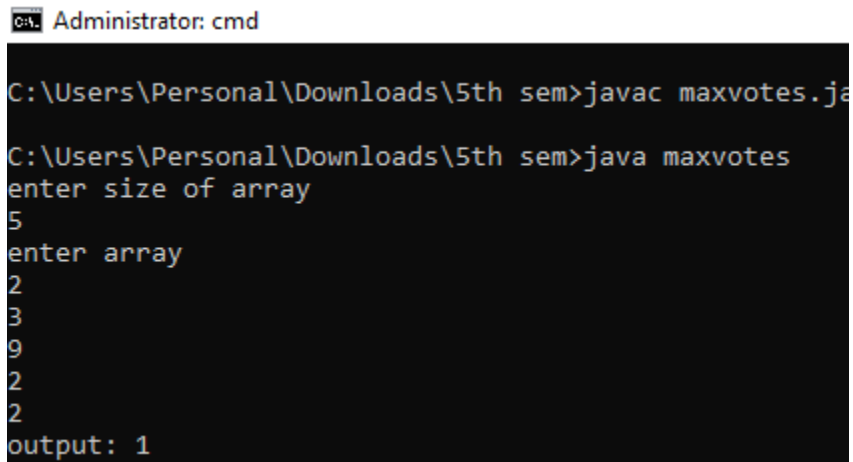
```
public class maxvotes
```

```
{
static int maxvote(int a,int A[])
{
int ca;
for(int i=1;i<=a;i++)
{
ca=A[i];
int count=0;
for(int j=1;j<=a;j++)
{
if (A[j]==ca)
count=count+1;
}
if(count>a/2)
{
return 1;
}
}
return 0;
}

public static void main(String args[])
{
Scanner sc=new Scanner(System.in);
System.out.println("enter size of array");
```

```
int n=sc.nextInt();
int A1[]=new int[10];
System.out.println("enter array");
for(int i=0;i<n;i++)
{
    A1[i]=sc.nextInt();
}
System.out.println("output: "+maxvote(n,A1));
}
}
```

### Output:



```
C:\Users\Personal\Downloads\5th sem>javac maxvotes.java
C:\Users\Personal\Downloads\5th sem>java maxvotes
enter size of array
5
enter array
2
3
9
2
2
output: 1
```

### Analysis:

Max vote Naive:

MaxVotes ( $a_1, a_2, \dots, a_n$ ): for  $i$  from 1 to  $n$ :

currentElement  $\leftarrow a_i$

Count  $\leftarrow 0$

for  $j$  from 1 to  $n$ :

if  $a_j = \text{currentElement}$ :

Count  $\leftarrow \text{Count} + 1$

if Count  $> n/2$ :

return  $a_i$

return "no max vote"

$$T(n) = (n+1)(n+1) + 1 + 1$$

$\downarrow$

there are two for loops

$$= n^2 + 2n + 1 + 2$$

$$= n^2 + 2n + 3$$

$$T(n) = O(n^2)$$