DESIGN AND ANALYSIS OF ALGORITHM

K S PRABHATH

19BCE7564

MaxVotes.jav

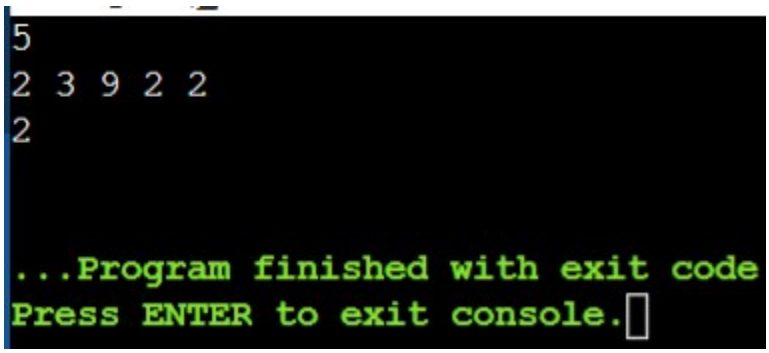
a:

CODE:

```
import java.util.*;
public class Main {
public static int countInRange(int[] nums, int num,
int lo, int hi) {
int count = 0;
for (int i = lo; i <= hi; i++) {
if (nums[i] == num) {
count++;
}
}
return count;
}
public static int majorityElementRec(int[] nums,
int lo, int hi) {
if (lo == hi) {
return nums[lo];
}
int mid = (hi-lo)/2 + lo;
int left = majorityElementRec(nums, lo,
mid);
int right = majorityElementRec(nums,
mid+1, hi);}
```

```
public static int majorityElement(int[]
nums) {
return majorityElementRec(nums, 0,
nums.length-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 res = majorityElement(arr);
System.out.println(res);
}
}
```

OUTPUT:



Linear Search:

CODE:

```
public class LinearSearch {
```

```
public static void searchIter(int arr[],
int target) {
for(int i=0; i<arr.length;i++) {
if(arr[i]==target) {
System.out.println(target+" is present in given array -
Iterative");
return;
}
}
System.out.println(target+" is not present in given array
- Iterative");
}
public static void searchRec(int arr[], int
target, int idx) {
if(arr[idx]==target) {
System.out.println(target+" is present in given array -
Recursive");
}
else if(idx<arr.length-
1) {
searchRec(arr,target,i
dx+1);
}
else
{
System.out.println(target+" is not present in given array
- Recursive");
}
}
public static void main(String
args[]) {
int arr[] = {12,33,
45,65,77,98};
int target  = 98;
searchIter(arr, target);
```

```
searchRec(arr, target, 0);
target = 48;
searchIter(arr, target);
searchRec(arr, target, 0);
}
}
```

OUTPUT:

```
<terminated> LinearSearch [Java Application] C:\Program Fil
98 is present in given array - Iterative
98 is present in given array - Recursive
48 is not present in given array - Iterative
48 is not present in given array - Recursive
```

Binary Search:

CODE:

```
public class BinarySearch {
public static void searchIter(int arr[], int target) {
int mid, left = 0, right = arr.length - 1;
while(left<=right){
mid = left + (right-left)/2;
if (arr[mid]==target) {
System.out.println(target+" is present in given array - Iterative");
return;
}
else if(target < arr[mid]) {
right = mid -1;
}
else {
left = mid +1;
}
}
System.out.println(target+" is not present in given array - Iterative");
}
public static void searchRec(int arr[], int left, int right, int target) {
if(left>right) {
System.out.println(target+" is not present in given array - Recursive");
return;
}
int mid = left + (right-left)/2;
if(arr[mid]==target) {
System.out.println(target+" is present in given array - Recursive");
return;
}
else if(arr[mid]>target) {
searchRec(arr,left, mid - 1,target);
}
else {
```

```
searchRec(arr,mid+1,right,target);}
public static void main(String args[]) {
int arr[] = {12,33, 45,65,77,98};
int target  = 98;
searchIter(arr, target);
searchRec(arr,0,arr.length-1,target);
target = 48;
searchIter(arr, target);
searchRec(arr, 0, arr.length-1,target);
}
}
```

OUTPUT:

```
<terminated> BinarySearch [Java Application] C:\Program Fil
98 is present in given array - Iterative
98 is present in given array - Recursive
48 is not present in given array - Iterative
48 is not present in given array - Recursive
```

ANALYSIS:

```
Array.sort(a)   runs for O(nlog(n))
while(i<n-1)
{
if(A[i]==A[i+1])
{
count++;
}
else
{
if(count>=(n/2)) {
flag=1;
break;
}
else
count=0;
}
i++;
}
```

At worst case the while loop runs for n-1 times

At best case it will run for n/2 times

Therefore, the total time complexity is given as nlogn + n = n(1+logn) which gives O(nlog(n))