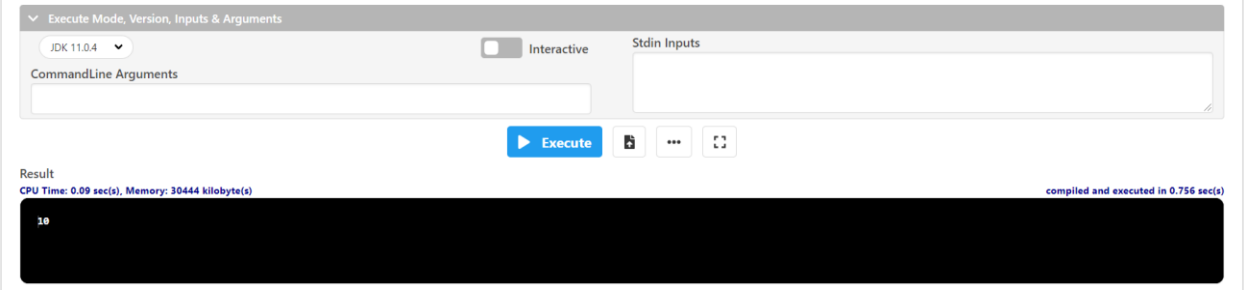
HARSHITHA
19BCE7582

```
import java.io.*;

public class maxVoteNaive {

static void findMajority(int arr[], int n)

{

int maxCount = 0;

int index = -1;

for (int i = 0; i < n; i++) {

int count = 0;

for (int j = 0; j < n; j++) {

if (arr[i] == arr[j])

count++;

}

if (count > maxCount) {

maxCount = count;

index = i;

}

}

if (maxCount > n / 2)

System.out.println(arr[index]);

else

System.out.println("No Majority Element");
```

```
}

public static void main(String[] args)

{

int arr[] = { 10, 10, 21, 11, 33, 10, 10 };

int n = arr.length;

findMajority(arr, n);

}

}
```



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))