Experiment No. 4

Sort a given set of ***n*** integer elements using **Quick Sort** method and compute its time complexity. Run the program for varied values of $n > 5000$ and record the time taken to sort. Plot a graph of the time taken versus ***n*** on graph sheet. The elements can be read from a file or can be generated using the random number generator. Demonstrate using Java how the divide -and- conquer method works along with its time complexity analysis: worst case, average case and best case.

```
import java.util.*;
public class quicksort {
 int partition(int a[],int low,int high)
 {
        int p,i,j,temp;
        i= low+1;
        j= high;
        p=a[low];
        while(low<high)  // This loop is required   as we have to repeat swapping
       a[i] and a[j]  till cross over happens
        {
               while(a[i]<=p &&i<high) // (i<high) required to manage array index
otherwise exception will occur for i= n
                      i++;
               while(a[j]>p)
                      j--;
 if(i<j)
 {
        temp= a[i];
        a[i]=a[j];
        a[j]=temp;

 }
 else
 {
        temp= a[low];
 a[low]=a[j];
 a[j]=temp;
 return j; // when cross over happens

 }

 }
 return j;  // Function must return value
 }
 void sort(int a[],int low,int high)
 {
        if(low<high)
        {
               int k= partition(a,low,high);
               sort(a,low,k-1);
               sort(a,k+1,high);
        }
 }

       public static void main(String[] args) {
```

```
		Scanner sc= new Scanner(System.in);
		System.out.println("enter the value n");
		int n= sc.nextInt();
		int a[]= new int[n];
		Random rand = new Random();
		for(int i=0;i<n;i++)
			a[i]= rand.nextInt(10000);// range of values
		System.out.println("before sorting");
		for(int i=0;i<n;i++)
		System.out.println(a[i]);
		long startTime=System.nanoTime();
		quicksort q= new quicksort();
		q.sort(a, 0, n-1);
		long stopTime=System.nanoTime();
		long elapseTime=(stopTime-startTime);
		System.out.println("Time taken to sort array is:"+elapseTime+"nano
	seconds");
		System.out.println("after sorting");
		for(int i=0;i<n;i++)
		System.out.println(a[i]);

				}

}
```