5.Sort a given set of n integer elements using Merge 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.Scanner;

**import** java.util.Random;

**class** MergeSort

{

**public** **static** **void** main(String[] args)

{

Scanner scan=**new** Scanner(System.*in*);

Random ran=**new** Random();

**long** start,stop;

System.*out*.println("Enter no of elements");

**int** n=scan.nextInt();

**int**[] a=**new** **int**[n];

System.*out*.println(" Enter the choice 1: Best Case 2: Average Case 3: Worst Case");

**int** ch=scan.nextInt();

**switch**(ch)

{

**case** 1: System.*out*.println(" Best Case");

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

a[i]=i;

**break**;

**case** 2: System.*out*.println(" Average Case");

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

a[i]=ran.nextInt(n);

**break**;

**case** 3: System.*out*.println(" Worst Case");

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

a[i]=n-i;

**break**;

}// end switch

//recording the start time

start=System.*nanoTime*();

//function call

*Mergesort*(a,0,n-1);

// recording the end time

stop=System.*nanoTime*();

*display*(a);

System.*out*.println("\nTime taken to sort " +a.length+ " elements =" +(stop-start));

}// end main

**private** **static** **void** display(**int**[] a)

{

// **TODO** Auto-generated method stub

System.*out*.println("the sorted array is");

**for**(**int** i=0;i<a.length;i++)

System.*out*.println(a[i]);

}//end dispaly

//function o divide the array

**public** **static** **void** Mergesort(**int**[] a, **int** low, **int** high)

{

**int** mid;

**if**(low<high)// array contains more than one element

{

mid=(low+high)/2;// dividing the array in to two sub arrays

*Mergesort*(a, low, mid);// sorting sub arrays

*Mergesort*(a, mid+1, high);

*Merge*(a,low,mid,high);// combining or merging the sorted arrays

}

}// end Mergesort

//function to merge two sorted arrays

**public** **static** **void** Merge(**int**[] arr,**int** low,**int** mid,**int** high)

{

**int** k,h=low,i=low,j=mid+1;

**int**[] b=**new** **int**[arr.length];

**while**(h<=mid && j<=high)

{

**if**(arr[h]<=arr[j])

{

b[i]=arr[h];

h++;

}

**else**

{

b[i]=arr[j];

j++;

}

i++;

}// end while

**if**(h>mid) // for remaining elements in upper half

{

**for**(k=j;k<=high;k++)

{

b[i]=arr[k];

i++;

}

}

**else** // for remaining elements in lower half

{

**for**(k=h;k<=mid;k++)

{

b[i]=arr[k];

i++;

}

}

//copy the contents from auxiliary array i.e. from b to arr

**for**(k=low;k<=high;k++)

arr[k]=b[k];

}// end merge

}// end MergeSort class