PROGRAM 5

**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 non 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.

**package labprograms;**

**import java.util.\*;**

**import java.io.\*;**

**public class p5 {**

**static int[] a;**

**static int size;**

**static boolean flag=true;**

**void getrn(int a[])throws IOException**

**{**

**Random random=new Random();**

**int n,count=0;**

**PrintWriter out=new PrintWriter(new File("Random.txt"));**

**while(count<size)**

**{**

**n=random.nextInt(size)+1;**

**a[count]=n;**

**out.print(n);**

**out.print("\t");**

**count++;**

**}**

**out.close();**

**System.out.println("The total numbers generated : "+count);**

**}**

**void mergesort(int a[],int low,int high)**

**{**

**int mid;**

**if(low<high)**

**{**

**mid=(low+high)/2;**

**mergesort(a,low,mid);**

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

**merge(a,low,mid,high);**

**}**

**}**

**void merge(int a[],int low,int mid,int high) {**

**int i1,i2,j,k;**

**int[] b=new int[size];**

**i1=j=low;**

**i2=mid+1;**

**if(flag)**

**{**

**while((i1<=mid)&&(i2<=high))**

**{**

**if(a[i1]<=a[i2])**

**{**

**b[j]=a[i1];**

**i1++;**

**}**

**else**

**{**

**b[j]=a[i2];**

**i2++;**

**}**

**j++;**

**}**

**if(i1>mid)**

**for(k=i2;k<=high;j++,k++)**

**b[j]=a[k];**

**else**

**for(k=i1;k<=mid;j++,k++)**

**b[j]=a[k];**

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

**a[k]=b[k];**

**}**

**else**

**{**

**while((i1<=mid)&&(i2<=high))**

**{**

**if(a[i1]>=a[i2])**

**{**

**b[j]=a[i1];**

**i1++;**

**}**

**else**

**{**

**b[j]=a[i2];**

**i2++;**

**}**

**j++;**

**}**

**if(i1>mid)**

**for(k=i2;k<=high;j++,k++)**

**b[j]=a[k];**

**else**

**for(k=i1;k<=mid;j++,k++)**

**b[j]=a[k];**

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

**a[k]=b[k];**

**}**

**}**

**public static void main(String[] args)throws IOException {**

**long st,et;**

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

**System.out.print("Enter the size of array(>5000) : ");**

**size=read.nextInt();**

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

**p5 obj=new p5();**

**obj.getrn(a);**

**st=System.nanoTime();**

**obj.mergesort(a,0,size-1);**

**et=System.nanoTime()-st;**

**PrintWriter outA=new PrintWriter(new File("Ascending.txt"));**

**for(int i:a)**

**{**

**outA.print(i);**

**outA.print("\t");**

**}**

**outA.close();**

**System.out.println("The Time Complexity for Average Case is : "+(et/1000000000.0)+" secs");**

**st=System.nanoTime();**

**obj.mergesort(a, 0, size-1);;**

**et=System.nanoTime()-st;**

**System.out.println("The Time Complexity for Best Case is : "+(et/1000000000.0)+" secs");**

**flag=false;**

**st=System.nanoTime();**

**obj.mergesort(a, 0, size-1);**

**et=System.nanoTime()-st;**

**PrintWriter outD=new PrintWriter(new File("Descending.txt"));**

**for(int i:a)**

**{**

**outD.print(i);**

**outD.print("\t");**

**}**

**outD.close();**

**System.out.println("The Time Complexity for Worst Case is : "+(et/1000000000.0)+" secs");**

**read.close();**

**}**

**}**