PROGRAM 4

**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 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 p4 {

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 sort(int[] a)

{

quicksort(a,0,size-1);

}

void quicksort(int[] a,int low,int high)

{

int i=low,j=high;

int temp;

int pivot=a[(low+high)/2];

if (flag)

{

while(i<=j)

{

while(a[i]<pivot)

i++;

while(a[j]>pivot)

j--;

if(i<=j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

i++;

j--;

}

}

if(low<j)

quicksort(a,low,j);

if(i<high)

quicksort(a,i,high);

}

else

{

while(i<=j)

{

while(a[i]>pivot)

i++;

while(a[j]<pivot)

j--;

if(i<=j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

i++;

j--;

}

}

if(low<j)

quicksort(a,low,j);

if(i<high)

quicksort(a,i,high);

}

}

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

long st,et;

Scanner sc=new Scanner(System.in);

System.out.print("Enter the number of elements(>5000) : ");

size=sc.nextInt();

a=new int[size];

p4 obj=new p4();

obj.getrn(a);

st=System.nanoTime();

obj.sort(a);

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 Worst Case is : "+(et/1000000000.0)+" secs");

st=System.nanoTime();

obj.sort(a);

et=System.nanoTime()-st;

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

flag=false;

st=System.nanoTime();

obj.sort(a);

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 Average Case is : "+(et/1000000000.0)+" secs");

sc.close();

}

}