Week 9 Homework 1 : sort in linear time

Que 14) Compare Histogram Sort and Counting Sort by Execution Time and Time Complexity Analysis

Ans):

Midterm.txt

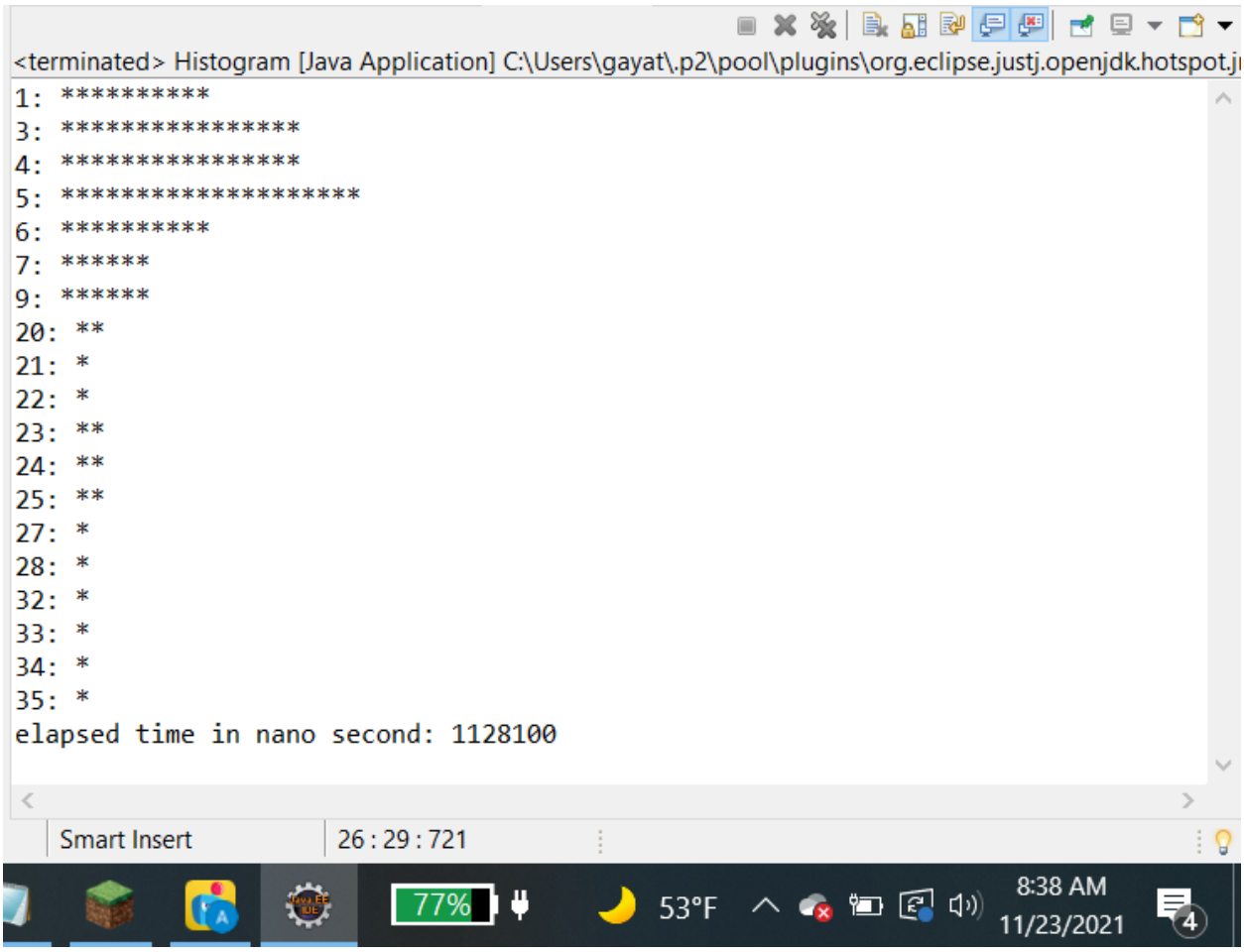
```
1  3  5  7  9  3  4  4  5  6
1  3  5  7  9  3  4  4  5  6
1  3  5  7  9  3  4  4  5  6
1  3  5 20 25 24 33  5  6  4
1  3  5 22 35 24 32  5  6  4
1  3  5 20 28 34 23  5  6  4
1  3  5 21 25 27 23  5  6  4
1  3  5  7  9  3  4  4  5  6
1  3  5  7  9  3  4  4  5  6
1  3  5  7  9  3  4  4  5  6
```

Histogram.java

```
package wk3;


import java.io.*;
import java.util.*;

public class Histogram {
    public static void main(String[] args)
            throws FileNotFoundException
   {
        // counters of test scores 0 - 100
    int[] counts = new int[101];

    Scanner input = new Scanner(new File("midterm.txt"));
    // read file into counts array
    while (input.hasNextInt()) {
        int score = input.nextInt();
        // if score is 87, then counts[87]++
        counts[score]++;
```

```
    }
    // Using nanoTime() is more precise
    long startTime = System.nanoTime();
    // print star histogram
    for (int i = 0; i < counts.length; i++) {
        if (counts[i] > 0) {
            // counts[i] stores how many students
            // scored i on the test, so print a
            // star (counts[i]) times
            System.out.print(i + ": ");
            for (int j = 0; j < counts[i]; j++) {
                System.out.print("*");
                }
                System.out.println();

            }
        }
    long stopTime = System.nanoTime();
    long elapsedTime= stopTime - startTime;
    System.out.println("elapsed time in nano second: "+ elapsedTime);
    }
} //Big(O)= n*log log n = n
```

Output cutout with snipping tool:

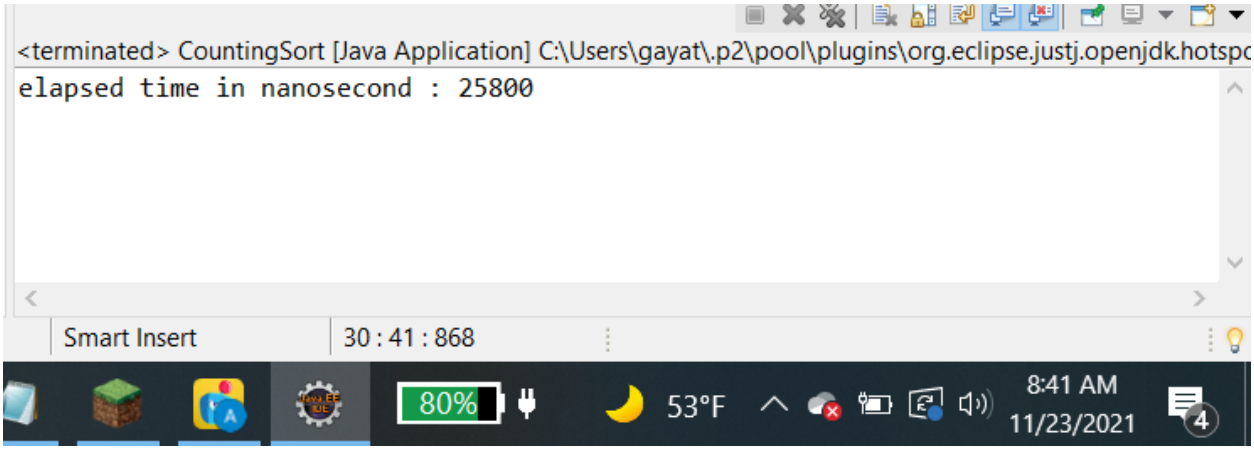


CountingSort.java

```java
package wk3;

//Java implementation of Counting Sort
//Output:
// Sorted character array is eeeefggkkorss
class CountingSort
{
 void sort(char arr[])
 {
     int n = arr.length;

     // The output character array that will have sorted arr
     char output[] = new char[n];

     // Create a count array to store count of inidividul
     // characters and initialize count array as 0
     int count[] = new int[256];
     for (int i=0; i<256; ++i)
         count[i] = 0;

     // store count of each character
     for (int i=0; i<n; ++i) //Big(O) = n
         ++count[arr[i]]; //Big(O) = 1

     // Change count[i] so that count[i] now contains actual
     // position of this character in output array
     for (int i=1; i<=255; ++i) //Big(O) = 1
         count[i] += count[i-1];

     // Build the output character array
     // To make it stable we are operating in reverse order.
     for (int i = n-1; i>=0; i--) //Big(O) = n
     {
         output[count[arr[i]]-1] = arr[i];
         --count[arr[i]];
     }

     // Copy the output array to arr, so that arr now
     // contains sorted characters
     for (int i = 0; i<n; ++i)  //Big(O)  = n
         arr[i] = output[i];
 } //Big(O) = 3n = n

 // Driver method
 public static void main(String args[])
 {
     CountingSort ob = new CountingSort();
     char arr[] = {
              1,  3,  5,  7,  9,  3,  4,  4,  5,  6,
              1,  3,  5,  7,  9,  3,  4,  4,  5,  6,
              1,  3,  5,  7,  9,  3,  4,  4,  5,  6,
              1,  3,  5, 20, 25, 24, 33,  5,  6,  4,
              1,  3,  5, 22, 35, 24, 32,  5,  6,  4,
```

```
                1,  3,  5, 20, 28, 34, 23,  5,  6,  4,
                1,  3,  5, 21, 25, 27, 23,  5,  6,  4,
                1,  3,  5,  7,  9,  3,  4,  4,  5,  6,
                1,  3,  5,  7,  9,  3,  4,  4,  5,  6,
                1,  3,  5,  7,  9,  3,  4,  4,  5,  6
                   };
        long startTime = System.nanoTime();
        ob.sort(arr);
        long stopTime = System.nanoTime();
        //System.out.print("Sorted character array is ");
        //for (int i=0; i<arr.length; ++i)
            //System.out.print(""+arr[i]+ " " + ",");
        //System.out.println();



        long elapsedTime = stopTime - startTime;
        System.out.println("elapsed time in nanosecond : "+elapsedTime);
    }
}
```

Output cutout with snipping tool:



- Step 3: Compare the execution time of Histogram Sort and Counting Sort and provide your observation.

Ans: The execution for Histogram sort in nano second is 1128100 and the execution time in nano second for Counting Sort is 25800. The execution time required for

Histogram sort in nano second is more than the execution time in nano second required for Counting Sort.

- Step 4: Compare Histogram Sort and Counting Sort by Big-O Time Complexity Analysis.

Ans: In Histogram Sort, 2 nested for loops are used for iteration. In one for loop iteration, for n time(n is length of array) the for loop is visited and in second for loop iteration, for log log n number of time loop is visited, and log log n is very small value so you can ignore it.

Big-O Time complexity Analysis for Histogram Sort is n*log log n.

With ignoring the small value, Big-O Time complexity Analysis for Histogram Sort is n.

In Counting Sort, 3 separate for loops are used for iteration. Every single for loops has n iterations. So one for loop has n iterations, second for loop has n iterations and third for loop has n iterations. So final

Big-O Time complexity Analysis for Counting Sort is n+n+n i.e. 3n

With ignoring the constant value, Big-O Time complexity Analysis for Counting Sort is n.