**DAA-Lab**

**LAB - 3**

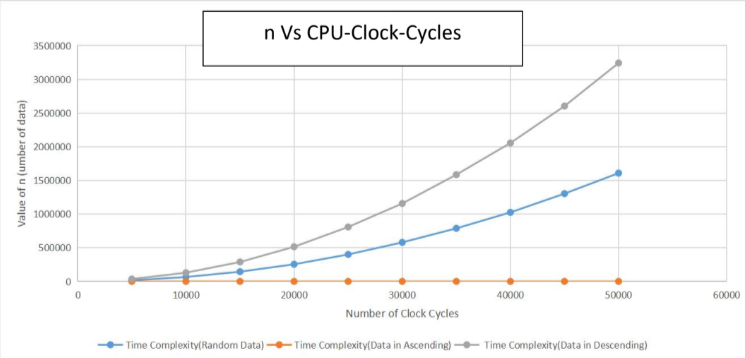
LQ1. Aim of the experiment : Rewrite the program no-HQ2 (Insertion Sort) with the following details. I. Compare the best case, worst case and average case time complexity with the same data except time complexity will count the CPU clock cycle time. II. Plot a graph showing the above comparison (n, the input data Vs. CPU times for best, worst & average case) III. Compare manually program no-2.2 graph vs program no-3.1 graph and draw your inference. Sample Input/Output Same menu of 2.2 will be displayed. If option 8 will be pressed, then the output will be displayed as follows :

**INPUT:**

Enter your choice : 1

**OUTPUT:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sln0** | **Value of N** | **Tc of Random Data** | **TC in Ascending Data** | **TC in Descending Data** |
| **1** | **5000** | **16107** | **18** | **32649** |
| **2** | **10000** | **64401** | **36** | **129228** |
| **3** | **15000** | **144820** | **53** | **288926** |
| **4** | **20000** | **258228** | **71** | **515440** |
| **5** | **25000** | **402970** | **89** | **808491** |
| **6** | **30000** | **580602** | **106** | **1156705** |
| **7** | **35000** | **791042** | **123** | **1574296** |
| **8** | **40000** | **1029574** | **140** | **2056922** |
| **9** | **45000** | **1305766** | **158** | **2600633** |
| **10** | **50000** | **1606417** | **178** | **3215868** |



LQ2. Aim of the experiment: Consider an n × n matrix A = (aij), each of whose elements a is a nonnegative real number, and suppose that each row and column of A sums to an integer value. We wish to replace each element aj with either L aij] or [ aj without disturbing the row and column sums. Write a program by defining an user defined function that is used to produce the rounded matrix as described in the above example. Find out the time complexity of your algorithm/function.

Enter the value of n: 4

Enter element at row 1, column 1: 10.9

Enter element at row 1, column 2: 2.5

Enter element at row 1, column 3: 1.3

Enter element at row 1, column 4: 9.3

Enter element at row 2, column 1: 3.8

Enter element at row 2, column 2: 9.2

Enter element at row 2, column 3: 2.2

Enter element at row 2, column 4: 11.8

Enter element at row 3, column 1: 7.9

Enter element at row 3, column 2: 5.2

Enter element at row 3, column 3: 7.3

Enter element at row 3, column 4: 0.6

Enter element at row 4, column 1: 3.4

Enter element at row 4, column 2: 13.1

Enter element at row 4, column 3: 1.2

Enter element at row 4, column 4: 6.3

|

Rounded matrix:

24.00 | 11.00 3.00 1.00 9.00

27.00 | 4.00 9.00 2.00 12.00

21.00 | 8.00 5.00 7.00 1.00

24.00 | 3.00 13.00 2.00 6.00

**- - - -**

26.00 30.00 12.00 28.00

LQ3: Aim of the experiment: Write a menu driven program as given below, to sort an array of n integers in ascending order by insertion sort algorithm and determine the time required (in terms of step/frequency count) to sort the elements. Repeat the experiment for different values of n and different nature of data (i.e.apply insertion sort algorithm on the data of array that are already sorted, reversely sprted and random data). Finally plot a graph of the time taken versus n for each type of data. The elements can be read from a file or can be generated using the random number generator.

**INPUT:**

INSERTION SORT MENU

0. Quit

1. Generate n Random numbers and create an Array

2. Display the Array

3. Sort the Array in Ascending Order using Insertion Sort

4. Sort the Array in Descending Order

5. Time Complexity to sort ascending of Random Data

6. Time Complexity to sort ascending of Sorted Data

7. Time Complexity to sort ascending of Reversely Sorted Data

8. Time Complexities in tabular form

Enter your choice: 8

**OUTPUT:**

