ASSIGNMENT I : Principles of Analysis of Algorithms and Sorting Methods Semester II 2021/2022

**Objectives**

Apply fundamental techniques of algorithms and algorithm design, and their associated data structures in problem solving and programming.

**Specification**

Implement the assignment in **Java**. The program is required to implement **insertion, bubble, quick and radix sort** algorithms on a list of words in **lexicographic order**. Add a counter in appropriate location of your implementation to count the number of times primitive operations. Run the program using experimental English dataset given and try to find and run the worst case, average case, and best case.

* 1)  Implement the insertion sort, bubble sort, and quick sort, and make sure they run as it supposed to be on the English dataset. Explain in the report how the algorithms that you implement sort the words clearly. (+30%)
* 2). Implement the radix sort algorithm **based on the explanation in the lecture** and modify it for a) sorting words instead of numbers b) words of different length. Explain in the report how the algorithm that you implement sort the words clearly. (+30%)
* 3) Use good object-oriented practice (e.g. comment, indentation, function, inheritance …) and appropriate design pattern when implementing the sorting algorithms. (+10%)
* 4)  Add counter to count the number of primitive operations. Plot the best case, average case, and worst-case time complexity of the 4 sort algorithms. Note: can plot using Excel. (+20%)
* 5) Specify the time complexity of the algorithms, justify your answers based on the analysis of the graphs obtained. (+10%)

**This assignment is to be carried out in a group of 4. References taken from any sources must be quoted and declared. No sharing of answers with other groups. Penalty for late submission, no excuse for late submission will be accepted.**

* **Program and report submission deadline**: **Sunday,** **22/5/2022 11.59 pm.**
* **Only one of the group members has to submit the work.**

**Assignment Assessment Rubric**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Excellent (80-100%) | Good (65-79%) | Moderate (40-64%) | Poor (0-39%) | Total |
| Part 1:  Sorting algorithms (30%) | Algorithms are correctly implemented for sorting words and they are explained clearly. | One algorithm is incorrectly implemented for sorting words, or one of the algorithms is fail to be explained clearly. | Two algorithms are incorrectly implemented for sorting words, or the algorithms are fail to be explained clearly. | All algorithms are incorrectly implemented for sorting words, or fail to be explained clearly, or cannot run. |  |
| Part 2: Radix sort (30%) | Radix sort is correctly implemented for sorting words and clearly explained. | Radix sort can only sort words with same length, or some part of the radix sort is not explained clearly. | Some errors in sorting the words, or the radix sort is not explained properly. | Program does not run, or wrong radix sort algorithm variety is implemented, or radix sort is not explained. |  |
| Part 3:  Good OOP | Proper comments, indentation, functions, and inheritance used in all codes. | Two of the algorithms do not properly implements OOP. | Three of the algorithms do not properly implements OOP. | Lack of comments, indentations, functions, and inheritance used in all algorithms. |  |
| Part 4: Counting | All counters are correctly added. Graphs are plotted. | Some counters are incorrectly added/ not added. Graphs are plotted. | Many counters are incorrectly added/ not added. Some graphs are incorrectly plotted. | No counter is added. No graphs. |  |
| Part 5: Analysis | Correctly specify the time complexity of the algorithms and correctly justify by analyzing the graphs obtained. | Correctly specify the time complexity of the algorithms, but some of the graphs obtained in the experiments are not analyzed. | Correctly specify the time complexity of the algorithms but do not analyze the graphs obtained. | No time complexity given. No analysis being carried out. |  |