CSE331 Assignment Manual #1

Sorting Algorithm Analysis and Implementation

1 Introduction

- This document provides comprehensive guidelines regarding the task of coursework: **CSE331: Introduction to Algorithms**.
- The assignment constitutes **10%** of the overall grade. Students who demonstrate outstanding performance can be awarded additional bonus marks.
- The goal of this task is the **analysis and implementation of sorting algorithms**.
- Students are required to implement and analyse 6 conventional comparison-based sorting algorithms, as well as 6 relatively new sorting algorithms, in a comparative study.
- A thorough analysis of the **time complexity** of each algorithm is mandatory. Additionally, students must conduct experiments on various input sequences to compare the **execution time** and **sorting accuracy** of the implemented algorithms.
- The use of **Large Language Models (LLMs)** is encouraged for brainstorming and code refactoring at a conceptual level. However, the direct use of AI-generated solutions(code) is strictly prohibited.

2 Submission Deadline

The deadline for the coursework submission is **17:59 on 15 April 2025**. Submissions must be made in **PDF format** via **Blackboard**.

3 Coursework Structure

3.1 Core Requirements

- **Conventional Comparison-Based Sorting Algorithms (6 algorithms):**

- Implement and analyse the following algorithms: **Merge Sort** [6], **Heap Sort** [3], **Bubble Sort** [4], **Insertion Sort** [6], **Selection Sort** [6], and **Quick Sort** [5].
- Students are required to implement these algorithms from scratch, ensuring a clear understanding and explanation of their design principles.
  Note: Standard library sorting functions may only be consulted for reference purposes and must not be utilised in the implementation.
- A comprehensive analysis of the **design rationale** and **time complexity** of each algorithm is essential. ↳ why this design

- **Contemporary Sorting Algorithms (6 algorithms):**
  - Implement and analyse the following algorithms: **Library Sort** [1], **Tim Sort**[1], **Cocktail Shaker Sort** [6], **Comb Sort** [2], **Tournament Sort** [8], and **Introsort** [7].
  - Provide a pseudo code of each contemporary sorting algorithm.
  - Provide a comparative discussion regarding the distinct characteristics, advantages, and disadvantages of each algorithm.

- **Generation and Evaluation of Input Data:**
  - Generate a diverse set of input sequences to evaluate and compare the performance of each algorithm.
  - Assess the algorithms based on multiple criteria, including **execution time**, **memory consumption**, and **stability**.
  - Input sizes must be varied from a ($1K$-$1M$) and each test must be executed a minimum of *10* times to calculate and record the mean execution time.

3.2 Report Writing Guidelines

- **Language**: The report must be composed in **English**.
- **Structure**: The report must adhere to the following structure:
  1. Problem Statement
  2. Basic Sorting Algorithms
  3. Advanced Sorting Algorithms
  4. Experimental Results and Analysis
- **Page Limit**: A maximum of **eight pages**, inclusive of tables and figures, excluding references. Appendices are not accepted.
- **Formatting Requirements**: A template will be provided. Modification of font size or margins is strictly prohibited.

[1] `https://mail.python.org/pipermail/python-dev/2002-July/026837.html`

- **Code and Data Publication**: Students are required to publish their source code and datasets on a public repository (URL of the repository(e.g., GitHub) must be included in the submitted report.)

3.3 Evaluation Criteria

- Understanding of the Problem and Algorithm Overview
- Theoretical Analysis
- Experimental Design and Implementation
- Experimental Results and Interpretation
- Report Structure and Presentation

4 Experimental Details

Students are required to generate diverse input sequences to conduct performance evaluations. The following data categories must be included.

- Sorted Data (ascending and descending order)
- Random Data (completely randomly-ordered sequences)
- Reverse Sorted
- Partially Sorted Data (sequences with partially ordered elements)

For each dataset, students must compare the execution time across all implemented sorting algorithms and evaluate the sorting accuracy.

A References

- Overleaf Tutorial: [2]
- Example GitHub Repo: [3]
- Academic Writing Guide: [4]

¡- style of reference

[2] https://www.youtube.com/playlist?list=PLHXZ9OQGMqxcWWkx2DMnQmj5os2X5ZR73
[3] https://github.com/jyyulab/MICA
[4] https://www.youtube.com/watch?v=-m3U-JrbBBg&list=PLzZ7PPT4KK5qNzQoszfF-BDIj5LOJatu1

References

[1] M. A. Bender, M. Farach-Colton, and M. A. Mosteiro. Insertion sort is o (n log n). *Theory of Computing systems*, 39:391–397, 2006.

[2] W. Dobosiewicz. An efficient variation of bubble sort. 1980.

[3] G. E. Forsythe. Algorithms. *Communications of the ACM*, 7(6):347–349, 1964.

[4] E. H. Friend. Sorting on electronic computer systems. *Journal of the ACM (JACM)*, 3(3): 134–168, 1956.

[5] C. A. R. Hoare. Algorithm 64: quicksort. *Communications of the ACM*, 4(7):321, 1961.

[6] D. E. Knuth. *The Art of Computer Programming: Sorting and Searching, volume 3*. Addison-Wesley Professional, 1998.

[7] D. R. Musser. Introspective sorting and selection algorithms. *Software: Practice and Experience*, 27(8):983–993, 1997.

[8] A. Stepanov and A. Kershenbaum. Using tournament trees to sort. *Polytechnical Institute of New York, CATT Technical Report*, pages 86–13, 1986.