UNIVERSITY OF MISSOURI-COLUMBIA
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

CS 8050– DESIGN AND ANALYSIS OF ALGORITHMS II
Fall 2025

ASSIGNMENT 1: Sorting

Due Date: Friday September 19th , 2025, at 5:00 pm
(100 points)

1 Goals

- Write Java code to sort a sequence of elements into ascending order by using the following sorting algorithms: Bubble, Selection, Insertion, Shell, Merge, Quick and Radix Sorts.
- Assess the efficiency of various sorting algorithms through theoretical and experimental analysis.
- Develop a written comparative report analyzing algorithm performance and trade-offs.

2 Description and Requirements

This assignment has two distinct components:

2.1 Visualization Component (GUI)

Graphical demonstrations of various sorting algorithms are instructive, as they provide insight into how an algorithm behaves. Consider a collection of vertical lines of varying lengths, such as the ones in Figure 1a. Create a sorting demonstration that sorts the lines by length, as shown in Figure 1b.

You should draw the configuration of lines after every swap or move that a given sorting algorithm makes. If you delay execution very briefly after each redraw, the result will be an animation of the sort.

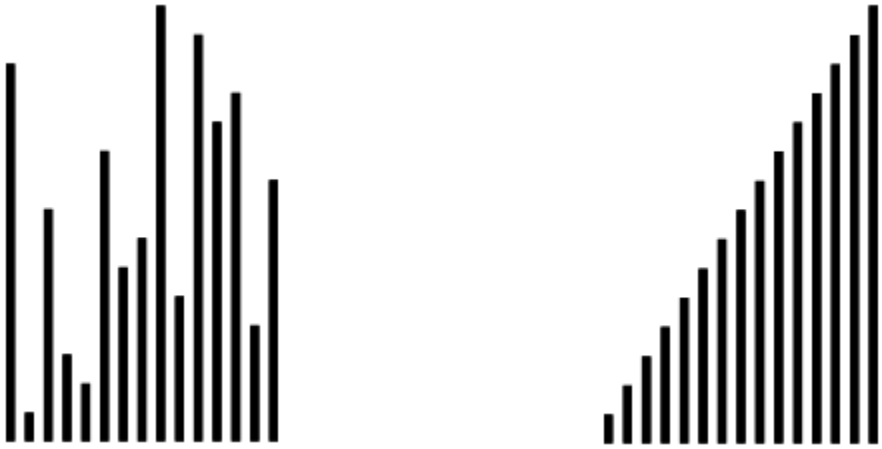


Figure 1: An animated sorting demonstration that sorts vertical lines
(a) before its execution (b) after its execution.

1. You need to read and exercise some of the examples provided in Unit-2 in the course lecture notes before starting to implement the requirements of this assignment.
2. You begin by drawing 256 lines, each one pixel wide but of different lengths and perhaps different colors arranged from shortest to longest so that they appear as a triangle.
3. The user then should exercise an option (one button click) to scramble the lines. At anyone clicks you should generate new random scrambled lines and then disable the action of this button until the all sorting methods are applied.
4. At a user signal, (button click, or radio button selection) your sorting algorithm should sort the lines.
5. When one Sort method is finished, the user should have an option to reset the sorted lines back as they were before applying this sorting method.
6. When all sorting methods have finished, the scrambling button can then be enabled and used to generate other random scrambled lines.
7. This means that each sort **should start** with the same scrambled lines so the user can compare methods.
8. You **must** use the provided two Java classes: SortGUI.java and.SortShow.java. You need to start by completing the code parts commented as //You need to complete this part.
9. Consult your TA for any point, specifically to your code from a Java point of view.

2.2 Experimental Component (Non-GUI)

- In addition to the GUI, implement an **experimental evaluation mode** that tests the same algorithms on large input arrays of size **1,000; 10,000; 100,000 elements**.
- This mode does not require animation. Instead, measure and record **execution times** for each algorithm.
- Results will be analyzed and presented in the sorting report.

Sorting Report Requirement

You must prepare a **written report (5–7 pages, IEEE format)** that includes:

- **Theoretical Analysis**:
  Discuss time and space complexity of each algorithm, highlighting algorithmic strategies.
- **Experimental Evaluation**:
  Run algorithms on 1,000; 10,000; and 100,000 elements in the non-GUI mode. Present execution times in tables and graphs. Compare observed results with theoretical expectations.
- **Comparative Discussion**:
  Compare algorithms in terms of efficiency, scalability, and application suitability. Address trade-offs such as simplicity vs. performance.
- **Critical Reflection**:
  Reflect on the impact of algorithm choice in real-world applications (databases, embedded systems, distributed systems). Discuss how hardware factors (CPU cache, memory hierarchy, parallelism) influence performance.

3 Hand In

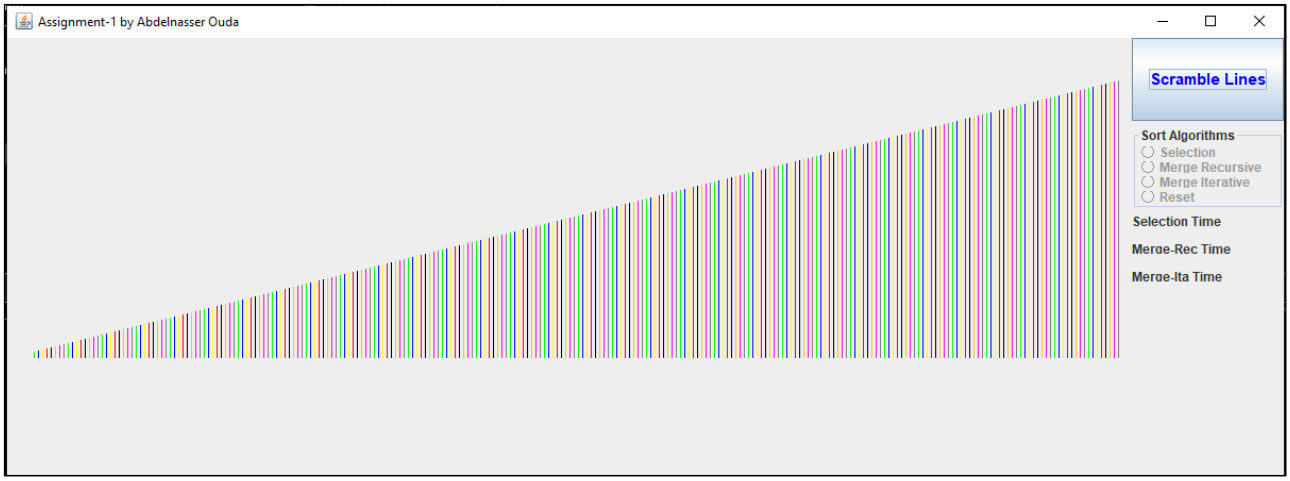
You need to use IntelliJ IDEA to create a Java Project and name it as "group#_cs8050_assignment#" (e.g., if group # is Group1 and you are submitting assignment 1 hence the IntelliJ project name will be "group1_cs8050_assignment1"). Then create a Java package in your IntelliJ Java project and name it the same name as the project name. Under this package you will create the Java classes and interfaces. You should write all your code and documentation under this package. You need to ZIP the IntelliJ project (folder) and submit it electronically through Canvas. Your ZIP file of your IntelliJ project folder must include the GUI visualization mode (256 lines), as well, include the non-GUI experimental mode (large arrays).

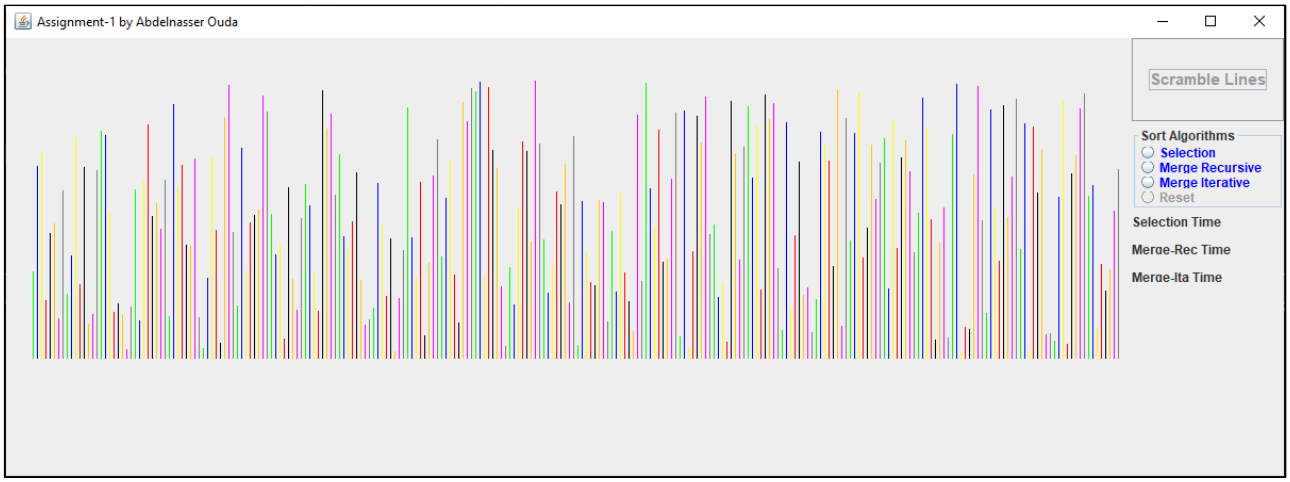
4 Grading

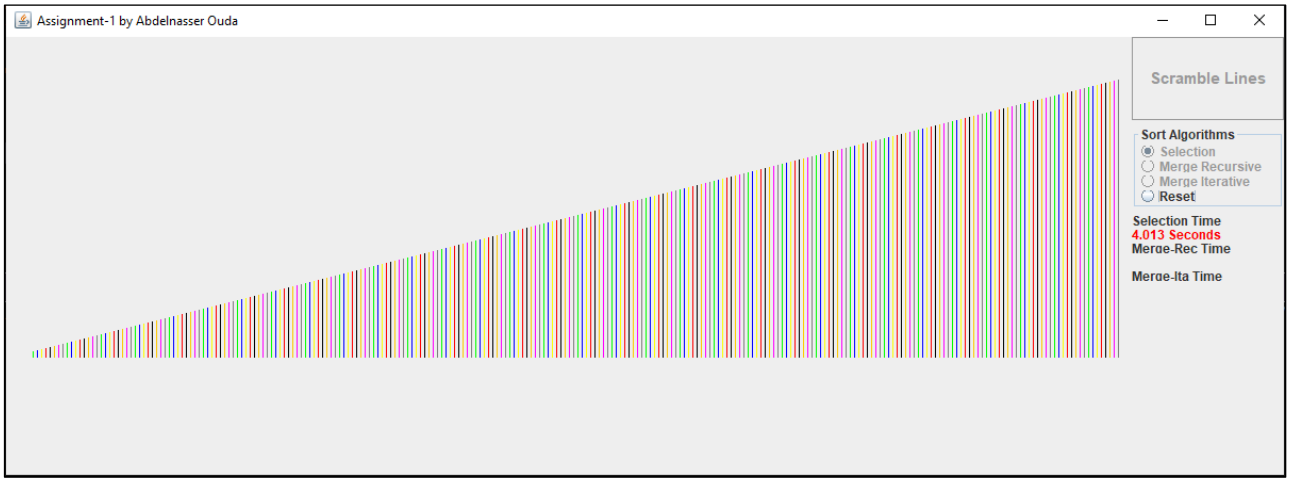
Description	Points
Code documentation	10
GUI + Scrambling method	15
Bubble Sort	5
Selection Sort	5
Insertion Sort	5
Shell Sort	5
Merge Sort Recursive	5
Quick Sort Recursive	5
Radix Sort	5
Report – Theoretical Analysis	10
Report – Experimental Evaluation (large arrays)	10
Report – Comparative Discussion	10
Report – Critical Reflection	5
Report – Presentation/Formatting	5
Total	100

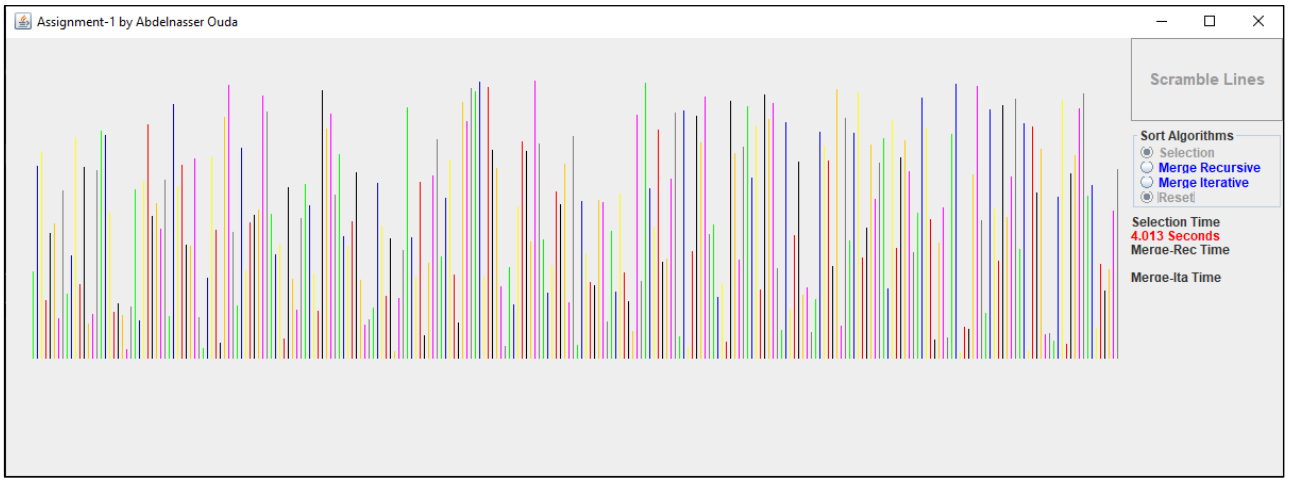
5 Assignment Sample Snapshots

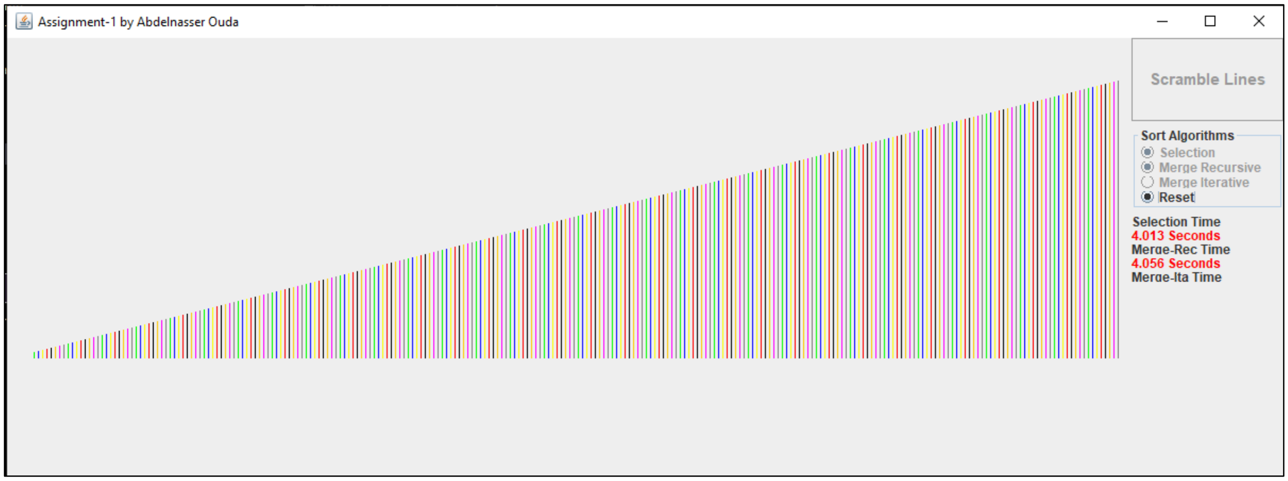
Please note the diferences between each of the following snapshots and implement similarly. These snapshots are taken by the order of the program execution.

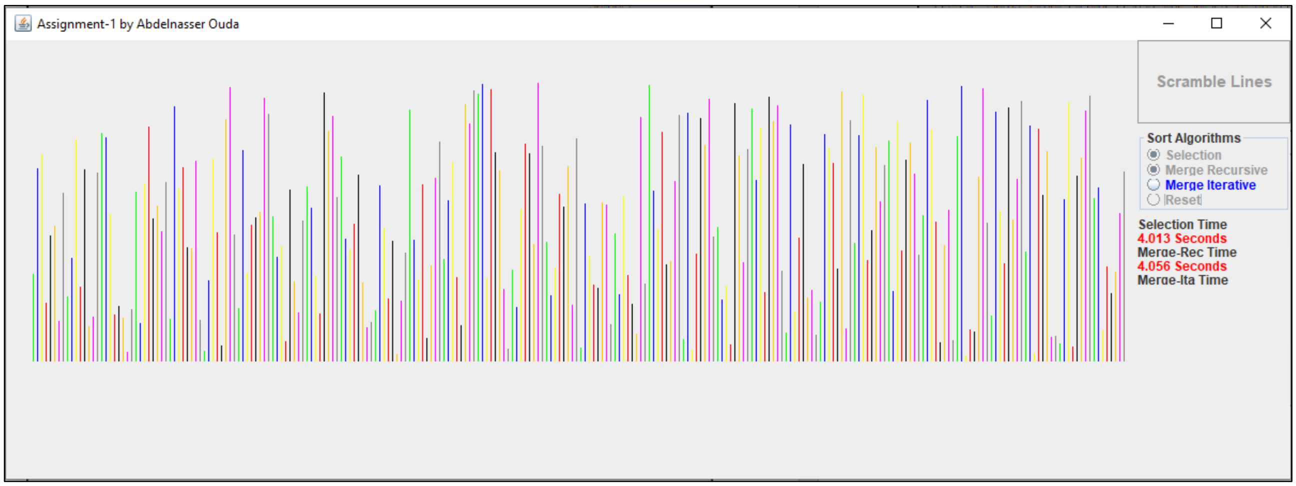


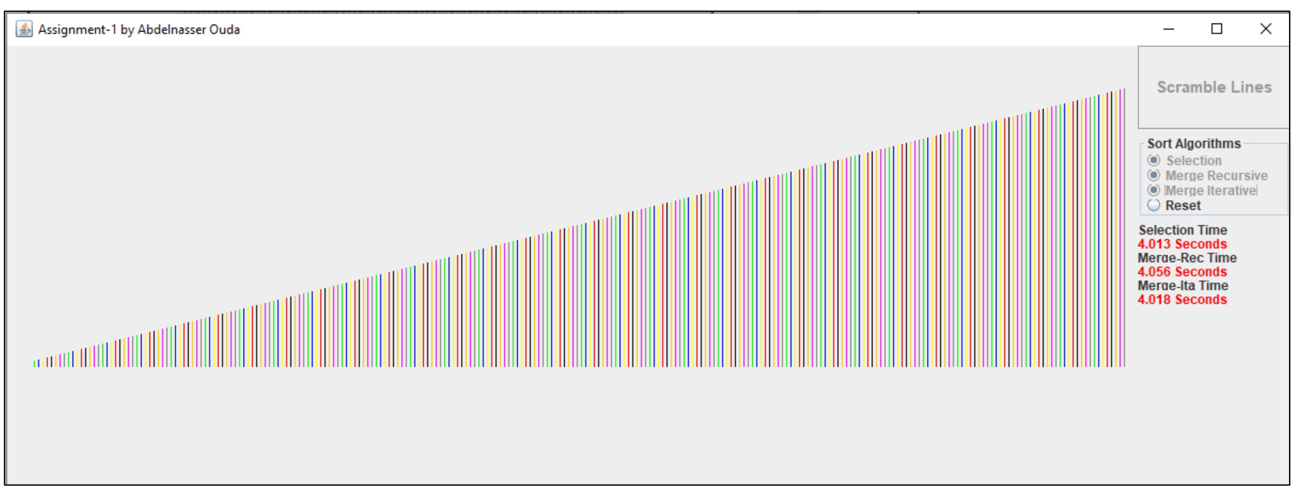

Assignment-1 by Abdelnasser Ouda
Scramble Lines
Sort Algorithms
Selection
Merge Recursive
Merge Iterative
Reset
Selection Time
Merge-Rec Time
Merge-Ita Time


Assignment-1 by Abdelnasser Ouda
Scramble Lines
Sort Algorithms
Selection
Merge Recursive
Merge Iterative
Reset
Selection Time
4.013 Seconds
Merge-Rec Time
Merge-Ita Time


Assignment-1 by Abdelnasser Ouda
Scramble Lines
Sort Algorithms
Selection
Merge Recursive
Merge Iterative
Reset
Selection Time
4.013 Seconds
Merge-Rec Time
Merge-Ita Time


Assignment-1 by Abdelnasser Ouda
Scramble Lines
Sort Algorithms
Selection
Merge Recursive
Merge Iterative
Reset
Selection Time
4.013 Seconds
Merge-Rec Time
4.056 Seconds
Merge-Ita Time


Assignment-1 by Abdelnasser Ouda
Scramble Lines
Sort Algorithms
Selection
Merge Recursive
Merge Iterative
Reset
Selection Time
4.013 Seconds
Merge-Rec Time
4.056 Seconds
Merge-Ita Time


Assignment-1 by Abdelnasser Ouda
Scramble Lines
Sort Algorithms
Selection
Merge Recursive
Merge Iterative
Reset
Selection Time
4.013 Seconds
Merge-Rec Time
4.056 Seconds
Merge-Ita Time
4.018 Seconds