Lab: Sorting Algorithms

Academic Integrity Notice

You are required to use only the algorithm implementations taught in class. Do not use AI-generated code, online code generators, or copied solutions. Submissions that do not reflect your own understanding and class instruction may receive little or no credit.
This assignment is designed to help you demonstrate your knowledge of sorting algorithms through implementation, experimentation, and analysis.

Learning Objectives

- By completing this assignment, students will be able to:
- Understand the fundamental principles of common sorting algorithms
- Correctly implement insertion sort, selection sort, merge sort, and quick sort
- Analyze time complexity and performance differences
- Collect and interpret experimental data
- Visualize algorithm performance using charts
- Compare theoretical expectations with empirical results
- Communicate technical findings in a clear written report

In this assignment, we are going to analyze the processing times of a number of sorting algorithms that we have learned in the class.

We would like to determine how the sorting algorithm would behave when they are tasked to sort an already sorted list of elements. In addition, we would like to determine the worst case performance of each of the sorting algorithms.

Please study the following problem description and prepare your solution for the question.

All sorting algorithms may accept an array variable or an array variable (both are acceptable). In turn, the algorithm must sort the elements of the array variable in ascending sorting order. For example, in case we have the following

```
int[] array {10, 50, 16, 16, 2, 5};
insertionSort(array);
// now, the contents of the array variable would be: 2, 5, 10, 16, 16, 50.
```

The algorithms that your assignment should implement and test are the following:

a) Insertion sort
b) Selection sort
c) Merge sort
d) Quick sort

Inside the algorithms, create custom variables to count the number of steps that are needed to sort a given list of items.

Deliverable 1: Implement the four sorting algorithms and demonstrate that one sample input/output for each algorithm (attach four screenshots in the report).

Deliverable 2:

For each of the four algorithms, use different input sizes as specified below and note down the average number of steps of your algorithm for each input size. Use the following table and report those numbers from your simulation.

In addition, separately, we need to provide sorted numbers to our algorithm in the simulation in order to observe how the algorithm would perform (best case or worst case scenario). Note, if we are to provide to the simulation a sorted list of numbers (ascending or descending) then do the following:

a) before each simulation starts, generate random number of the given input size
b) sort the elements by using the sort method. In order to sort the elements in descending order, you can specify a third parameter to your sort method (third parameter can be a functor). Or, you can create a descending sort method by modifying any existing sorting method so that it would sort the elements in descending order.

Simulation size: average of 1,000 steps

Input size: N random values (ranging between 0 and N), where N can be 1K, 5K, 10K, 15K, 20K

Input size	elements sorted in ascending order	elements sorted in descending order	random values	average steps
1K				
5K				
10K				
15K				
20K				

Deliverable 3: By using the tabular values,. Compare the graph of the five algorithms and comment on the following:

a) Plot a graph showing the performance of each algorithm when the elements are already sorted. Determine which sorting algorithm's performance is the worst? What is the reason of this performance?
b) Plot a graph showing the performance of each algorithm when the elements are sorted in descending order. Determine which sorting algorithm's performance is the worst? What is the reason of this performance?
c) Considering all the tabular data that you have gathered for the four algorithms, which sorting algorithm's performance on average is better than the others? Why?
d) Considering all the tabular data that you have gathered for the four algorithms, which sorting algorithm's performance on average is the worse than the others? Why?

Use a report to include all the deliverables and export the document as a pdf file. Your submission would be the pdf file containing the deliverables and the five program files (one program file for each of the four sorting programs, and the simulation program).

Submit the solution on the BrightSpace website by 11:59pm, Friday, 20 February 2026. Let me know if you have any questions. Thank you.

Rubrics of the Assignment is the following:

Assignment rubrics | Total points: 50	Points
Deliverable 1: summit the four program files (each program file must have a main method to run the sorting method) + attach four screenshots depicting the sample run of the program	15 points -5x: the sorting algorithm was not implemented correctly and one execution run of the method has not been shown in the main function -5x: Screenshot depicting the sample run of the algorithm was not attached
Deliverable 2: add the simulation table, graph in the report	15 points -3x: the simulation uses random numbers to sort the elements by using specified sorting methods one at a time -3x: the simulation further uses sorted list of numbers to sort the elements by using specified sorting methods -5x: graph and tabular data is incorrect due to faulty simulation design
Deliverable 3: based on the data gathered from the simulation, answer the four questions	20 points -5x: simulation data has been correctly analyzed to answer the provided questions -2x: the conclusion provided in the answer of the question is not supported by the data or the data is incorrect.
Miscellaneous	-10x points: The programs use algorithms that have not been taught in the class -15: Javadoc comments were not used throughout the programs -10x: programs have complier errors -30: Uses algorithms or techniques that have not been taught in the class