Cross-Review Summary Report

Assignment 2: Algorithmic Analysis and Peer Code Review

Pair 1: Basic Quadratic Sorts

Group: SE-2438

Student A: Agabekuly Asylbek – Selection Sort

Student B: Almukhamedov Temirlan – Insertion Sort

1. Algorithm Overview

This report presents a cross-analysis between two basic quadratic sorting algorithms: Selection Sort implemented by Agabekuly Asylbek and Insertion Sort implemented by Almukhamedov Temirlan. Both algorithms are comparison-based and operate in-place, requiring no additional data structures. While they share $O(n^2)$ complexity on average, their internal behavior differs significantly.

Selection Sort repeatedly selects the smallest (and optionally the largest) element from the unsorted portion and places it at its correct position. It performs a fixed number of comparisons but minimizes swaps. Insertion Sort, on the other hand, builds the sorted array one element at a time and is adaptive — it performs fewer operations if the data is already or nearly sorted.

2. Complexity Analysis

Algorithm	Best Case	Average Case	Worst Case
Insertion Sort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$
Selection Sort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$

Both algorithms exhibit quadratic time complexity on average and in the worst case. However, Insertion Sort adapts to partially sorted input, achieving $\Theta(n)$ performance in the best case, while Selection Sort performs a constant number of comparisons regardless of data order. Both algorithms require $O(1)$ auxiliary memory, as they modify the array in-place.

3. Code Quality and Design

Both implementations follow clean coding standards. Agabekuly Asylbek's Selection Sort uses bidirectional selection and early termination optimizations, while Temirlan's Insertion Sort uses an early-exit check for already sorted elements. Both implementations make use of a PerformanceTracker class to measure execution time, comparisons, and swaps. Comments and structure make the code easy to understand and extend.

4. Empirical Benchmark Comparison

Benchmark results were collected for input sizes n = 100, 1,000, 10,000, and 100,000.

Input Size (n)	Insertion Sort (ms)	Selection Sort (ms)	Observations
100	0.52	0.84	Both very fast for small input
1,000	18.9	20.9	Insertion slightly faster
10,000	121.4	126.7	Similar quadratic behavior
100,000	7050	7417.9	Both show $O(n^2)$ growth

Empirical data confirms theoretical expectations: both algorithms scale quadratically. Insertion Sort performs slightly faster due to fewer swaps and adaptive behavior. Selection Sort performs more consistently across all input orders.

5. Graphical and Empirical Discussion

When plotted, both algorithms produce parabolic curves, confirming $O(n^2)$ time complexity. Insertion Sort's curve is lower on nearly sorted data, while Selection Sort maintains a stable slope because its number of comparisons does not depend on input order. Both algorithms are inefficient for large input sizes but perfectly demonstrate basic sorting principles.

6. Conclusion

Both students successfully implemented and analyzed their respective algorithms. Insertion Sort, implemented by Almukhamedov Temirlan, showed slightly better empirical performance due to its adaptive nature. Selection Sort, implemented by Agabekuly Asylbek, provided predictable results and fewer swaps. Both implementations include metrics tracking, theoretical and empirical validation, and adhere to clean code practices. The project demonstrates a strong understanding of algorithmic analysis and performance evaluation principles.