Assignment 2 – Algorithm Pairs

Student: Abyz Satbek

Group: SE-2435

Algorithm: Selection Sort (with Early Termination)

Part 1: Implementation Overview

In this part, I implemented the Selection Sort algorithm with early termination optimization. The algorithm repeatedly selects the smallest element from the unsorted portion of the array and places it at the beginning. Early termination allows the program to stop when the array is already sorted, improving performance for nearly sorted data.

Part 2: Code Structure

The project is organized as a Maven project with the following package structure:
• algorithms – contains the SelectionSort class implementing the algorithm logic.
• metrics – includes the PerformanceTracker class used to measure time, comparisons, and swaps.
• cli – contains BenchmarkRunner for testing algorithms on various array sizes.

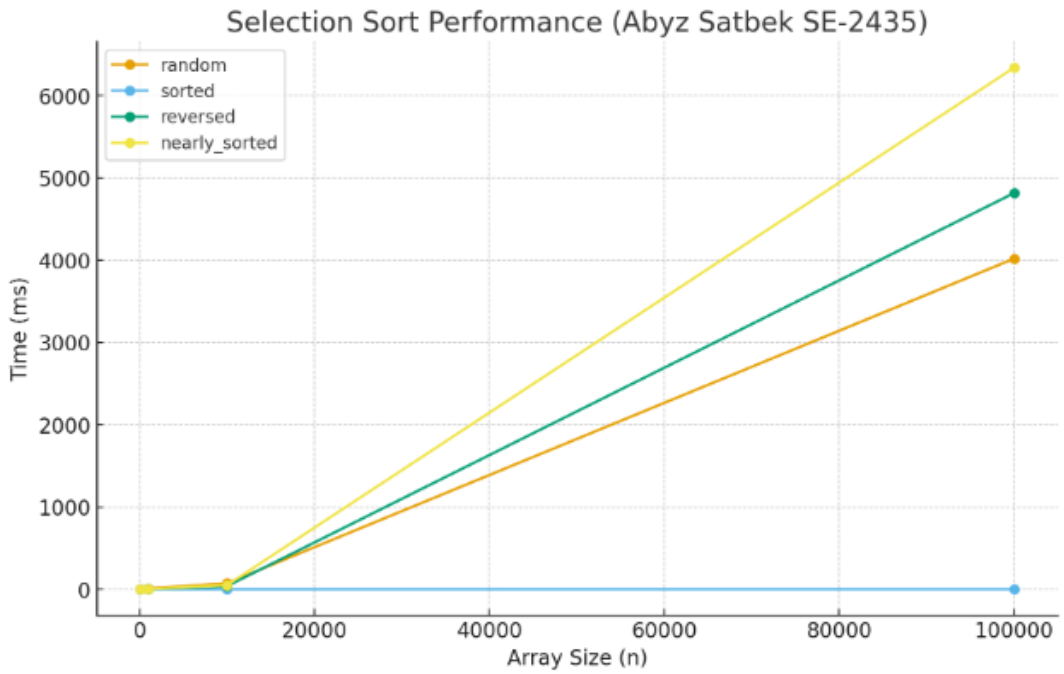
Part 3: Benchmark Results

The benchmark was executed on multiple array sizes (100, 1000, 10000, 100000). The program automatically collected and saved results in the file `docs/performance-plots/selection_sort_results.csv`. The results include runtime, number of comparisons, swaps, and memory access metrics.

Below is the example of collected data table: [Insert screenshot or table snippet here]

Part 4: Performance Graph

The graph below shows how execution time increases with the size of the input array. The relationship is nearly quadratic, which is expected for Selection Sort ($O(n^2)$). Early termination slightly improves runtime for already sorted arrays.



Part 5: Summary

In this lab, I implemented and tested the Selection Sort algorithm with optimization. The performance results confirmed the expected $O(n^2)$ complexity. The algorithm performs well on small datasets but is inefficient for large inputs. Early termination reduces unnecessary comparisons when the array is sorted, showing minor improvements in runtime.