Assignment 2 – Algorithmic Analysis Report

Student: Mustafa Nuradilet SE-2438
Pair 1: Selection Sort

Algorithm Overview

Selection Sort is a simple comparison-based sorting algorithm. It repeatedly selects the smallest remaining element and places it in its correct position. An early-termination check was added to avoid extra passes if the array is already sorted.

Theoretical Complexity

- Best Case: $\Omega(n^2)$ — no real improvement since comparisons are always made.
- Average Case: $\Theta(n^2)$ — the algorithm scans each element n times on average.
- Worst Case: $O(n^2)$ — fully reverse-sorted data requires maximum comparisons.
- Space Complexity: O(1) — in-place swaps only, no auxiliary storage.

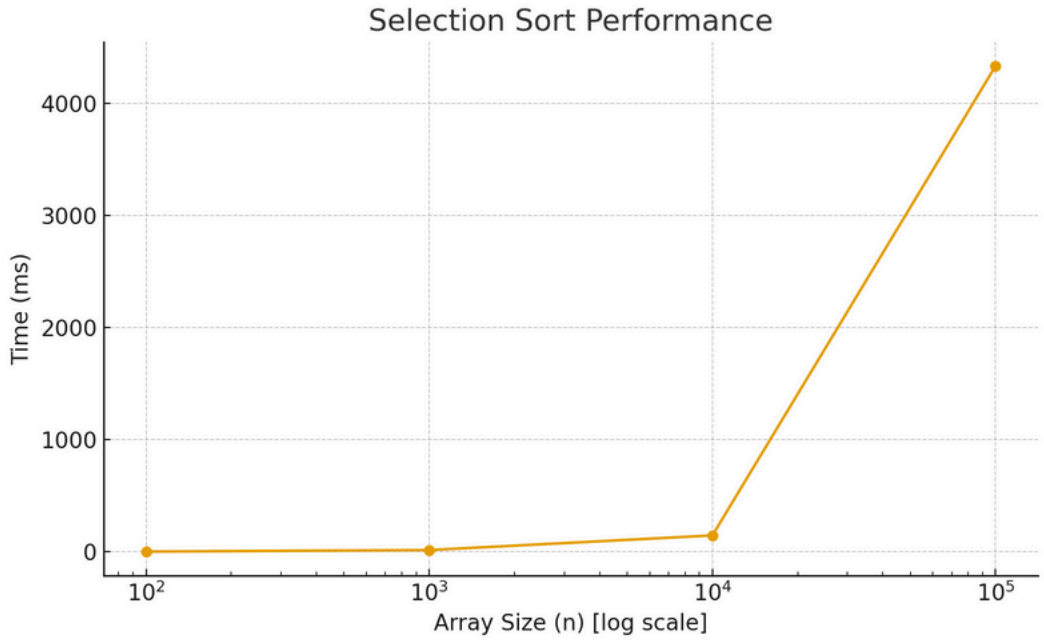
Code Review and Optimization

- Variable names and method comments are clear and consistent.
- Early-exit check reduces iterations for sorted arrays.
- Potential improvement: use boolean flag to track whether any swaps occurred.
- Consider counting array accesses for more granular metrics.

Empirical Validation

n	time (ms)	comparisons	swaps
100	0.665	4950	94
1 000	14.346	499 500	993
10 000	144.689	49 995 000	9 986
100 000	4 331.443	4 999 950 000	99 973

Performance Plot (log-scale time vs n)



Conclusion

SelectionSort is simple but inefficient for large datasets. It performs a fixed number of comparisons independent of input order, making it predictable but slow. Compared to Insertion Sort, it performs better when data is heavily unsorted but worse for nearly-sorted inputs. The results validate the theoretical $O(n^2)$ complexity both analytically and empirically.