Assignment 2 — Report (Partner Algorithm: Selection Sort)

Author: Myssyrgaliyev Mirzhan

Course/Group: SE-2414

Date:01.10.2025

# 1. Goal

Analyze the performance of my partner’s Selection Sort implementation using instrumentation for comparisons, swaps, and array accesses across a range of input cases.

# 2. Algorithm Description (Selection Sort + early-exit)

Selection Sort repeatedly selects the minimum element from the unsorted suffix and swaps it with the first element of that suffix. In the provided code, there is an additional early-exit optimization: if during an outer pass no swap is performed (the position i already holds the minimum), the algorithm breaks out early. This can reduce work on already-sorted or nearly-sorted arrays.

# 3. Theoretical Complexity

- Classical Selection Sort:  
 • Comparisons: Θ(n²) in all cases (the inner scan always runs).  
 • Swaps: O(n) (at most one per outer pass).  
 • Extra space: O(1).  
- With the provided early-exit (break when no swap in a pass):  
 • Best case can approach O(n) comparisons (e.g., array already sorted → only the first full scan then break).  
 • Worst/average remain Θ(n²) comparisons.

# 4. Test Harness & Correctness

Correctness was validated using the included tests:  
 - Empty array  
 - Single element  
 - Array with duplicates  
 - Already sorted array  
 - Reverse-sorted array  
 - Large random input (size 100000)  
All tests passed.

# 5. Instrumented Results

Metrics captured by the partner’s Selection Sort (from program output):

|  |  |  |  |
| --- | --- | --- | --- |
| Case | Comparisons | Swaps | Array accesses |
| Empty array | 0 | 0 | 0 |
| Single element | 0 | 0 | 0 |
| Array with duplicates | 9 | 2 | 18 |
| Already sorted (small) | 4 | 0 | 8 |
| Reverse-sorted (small) | 9 | 2 | 18 |
| Large random (n=100000) | 659767409 | 90489 | 1319534818 |

# 6. Discussion

• Trivial inputs (empty, single) require zero work, as expected.  
• Small arrays show low counts consistent with a single full scan and, when needed, one swap.  
• The large random case exhibits very high comparisons and accesses, reflecting Θ(n²) behavior of Selection Sort.  
• The early-exit reduced work on the already-sorted small case (no swap and a break after the first outer pass).

# 7. Conclusion

The partner’s Selection Sort passes all tests and, with an early-exit optimization, can be efficient on already-sorted inputs, but still demonstrates quadratic scaling on large random inputs. Metrics align with theoretical expectations.