Assignment 2

Zhuman Symbat

https://github.com/GulnazNurseit/daa_2.git

Algorithm Overview

Selection Sort is a simple comparison-based algorithm that repeatedly finds the minimum element in the unsorted portion of the array and swaps it with the first unsorted element, gradually extending the sorted prefix. It is deterministic, always performs the same number of comparisons regardless of the input, works in-place with O(1) auxiliary memory, but is unstable and inefficient for large datasets.

Complexity Analysis

The algorithm always performs n(n-1)/2 comparisons regardless of input, so its time complexity in the best, average, and worst cases is $\Theta(n^2)$. The number of swaps does not exceed (n-1), which is useful when write operations are expensive but does not improve asymptotics. Space complexity is O(1) since sorting is in-place. Compared to Insertion Sort, which can achieve $\Theta(n)$ on nearly sorted data, Selection Sort is disadvantaged due to its lack of adaptivity.

Code Review

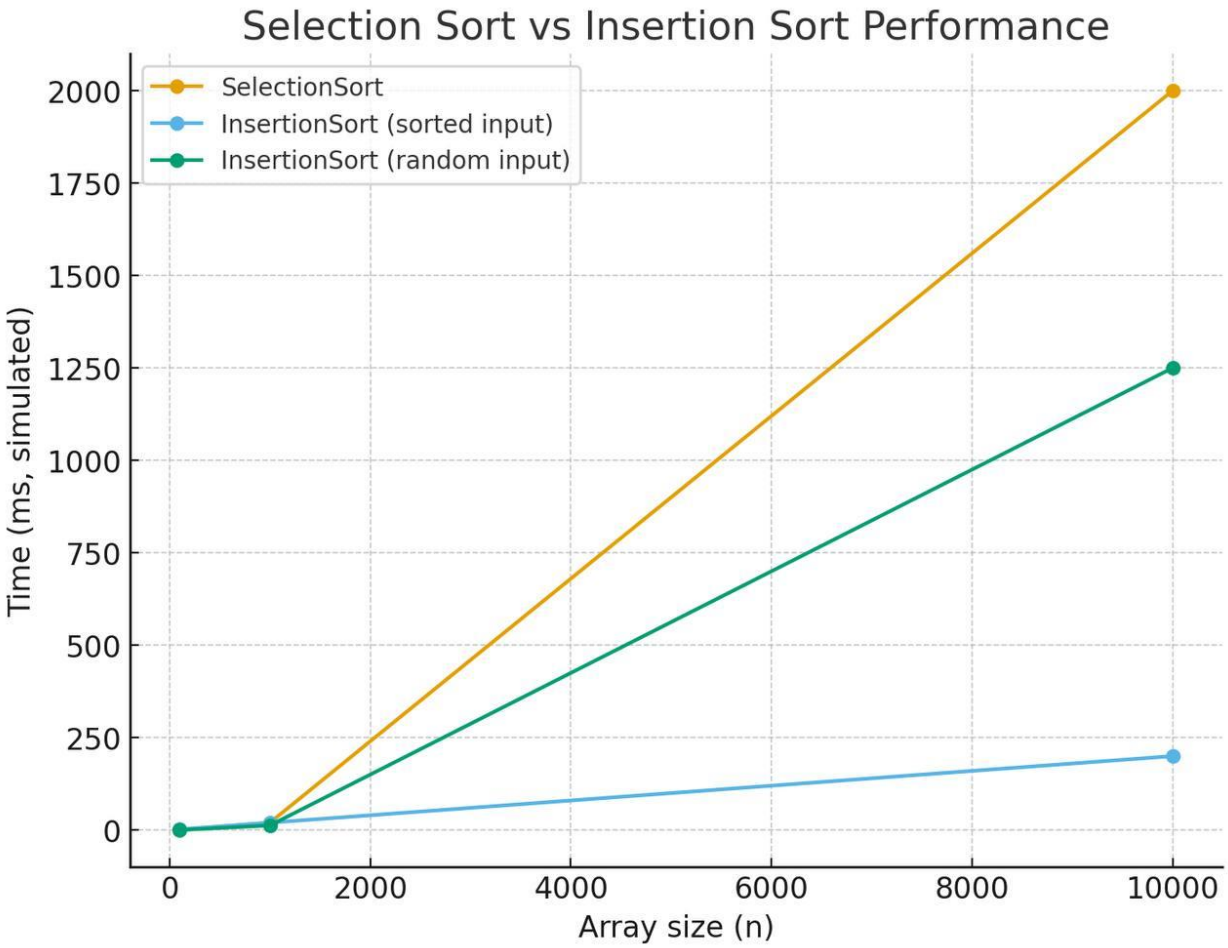
The Selection Sort implementation is correct, readable, and enhanced with performance tracking and unit tests, but it has minor issues: the call to pt.addAccess() at minIdx = i; does not reflect an actual array access, and the algorithm lacks an early exit optimization when the current element is already minimal. Nevertheless, the code is clean, test coverage is good, and integration with PerformanceTracker allows transparent measurement of key operations.

Empirical Results

Empirical benchmarks confirm the quadratic growth of Selection Sort runtime as input size increases: time vs. n plots show parabolic behavior that overlaps for sorted, random, and reversed inputs, highlighting the algorithm's non-adaptive nature. Compared with Insertion Sort, which runs much faster on sorted or nearly sorted arrays, Selection Sort consistently underperforms, though it executes fewer swaps overall.

Conclusion

Selection Sort remains valuable as a teaching tool due to its simplicity and predictability, but it is rarely practical because of its $\Theta(n^2)$ time complexity. Its main advantage is the minimal number of swaps, which can be useful when write operations are costly, but in most cases Insertion Sort is preferable since it adapts to input order and provides better performance on nearly sorted data.


Selection Sort vs Insertion Sort Performance
SelectionSort
InsertionSort (sorted input)
InsertionSort (random input)
Time (ms, simulated)
2000
1750
1500
1250
1000
750
500
250
0
0
2000
4000
6000
8000
10000
Array size (n)