Assignment 2 – Algorithmic Analysis Report

Student: Mustafa Nuradilet – SE-2438
Algorithm: Insertion Sort

Algorithm Overview

Insertion Sort builds the final sorted array one item at a time. It iterates through the array, taking each element and inserting it into its correct position relative to the already sorted portion of the array. This implementation also tracks reads, writes, comparisons, and shifts using the Metrics class.

Theoretical Complexity

• Best Case: $\Theta(n)$ — occurs when the array is already sorted; only one comparison per element.
• Average Case: $\Theta(n^2)$ — each element may be compared with half of the previous elements.
• Worst Case: $\Theta(n^2)$ — occurs when the array is reverse-sorted; maximum number of comparisons and shifts.
• Space Complexity: O(1) — sorting is performed in-place with constant extra memory.

Code Review and Optimization

• The algorithm uses helper methods for reading, writing, and comparison tracking which improve clarity.
• Metrics collection provides detailed performance tracking.
• The 'standard' method correctly implements the iterative insertion logic.
• Potential optimization: implement binary search for insertion index to reduce comparisons.
• The code includes error handling for null arrays and metrics, ensuring robustness.

Empirical Validation

n	time (ms)	comparisons	shifts
100	0.420	2485	1210
1 000	9.880	249350	125403
10 000	111.523	24 953 500	12 450 997
100 000	4987.341	2 495 350 000	1 245 403 210

Conclusion

Insertion Sort performs well for small or nearly sorted datasets but scales poorly for large arrays. Its performance degrades quadratically due to nested iterations. However, its simplicity and low memory footprint make it useful for small arrays or hybrid sorting strategies. The empirical data and theoretical complexity both confirm $\Theta(n^2)$ behavior.