AI Generated Sorting Algorithms Analysis:

BubbleSortAI, SelectionSortAI, and InsertionSortAI

Luis Gabriel Caceres Duran

Data structures & algorithms

Game Development, NBCC

October 3, 2024

Abstract

This report presents the performance analysis of three sorting algorithms developed by AI: BubbleSortAI, SelectionSortAI, and InsertionSortAI. These algorithms are based on traditional sorting methods, but they were generated by AI without further optimizations. The performance was measured for both unsorted and pre-sorted arrays. All the time measurements were taken in a consistent environment with identical conditions for each run. To ensure accuracy, each algorithm was executed three times under the same conditions, and the time displayed in the graphs represents the average of these three runs. This report explores the performance differences between the algorithms for various array sizes ranging from 1,000 to 1,000,000 elements. The tests were conducted in a controlled environment where no changes to hardware or system resources were made between test runs. This ensures that any differences in performance are due to the algorithms themselves.

Keywords: sorting algorithms, time complexity, ai generated

AI Generated Sorting Algorithms Analysis

Key findings

InsertionSortAI outperforms both BubbleSortAI and SelectionSortAI consistently across all dataset sizes. It not only handles large unsorted arrays more efficiently but also excels with already sorted data, likely due to optimizations that reduce its best-case time complexity to O(n). BubbleSortAI shows reasonable performance improvements over SelectionSortAI, especially when dealing with sorted arrays. While it generally operates with a $O(n^2)$ time complexity for unsorted data, optimizations such as early termination when the array is detected as sorted significantly enhance its efficiency in best-case scenarios. SelectionSortAI exhibits the least favorable performance, maintaining a consistent $O(n^2)$ time complexity regardless of whether the input data is sorted or unsorted. This lack of adaptability makes it the slowest among the three, particularly unsuitable for large-scale sorting tasks.

Time Complexity Insights

All three algorithms demonstrate quadratic time complexity ($O(n^2)$) for unsorted inputs, which hampers their scalability with increasing data sizes. However, InsertionSortAI and BubbleSortAI incorporate optimizations that allow them to perform significantly better on already sorted arrays, effectively reducing their best-case time complexity to O(n). In contrast, SelectionSortAI does not benefit from data order, maintaining the same performance irrespective of the initial array state.

Conclusion

InsertionSortAI is the most efficient, especially excelling with sorted data due to possible optimizations. BubbleSortAI follows, offering improved performance over SelectionSortAI when handling sorted arrays. SelectionSortAI remains the least efficient, with no performance gains from sorted inputs.While all three algorithms are limited by their quadratic time complexities, InsertionSortAI provides the best balance of performance and adaptability, making it the preferable choice within the scope of AI-generated sorting solutions.

Figure 1

Sorted Arrays Performance Comparison: BubbleSortAI, InsertionAI, SelectionSortAI

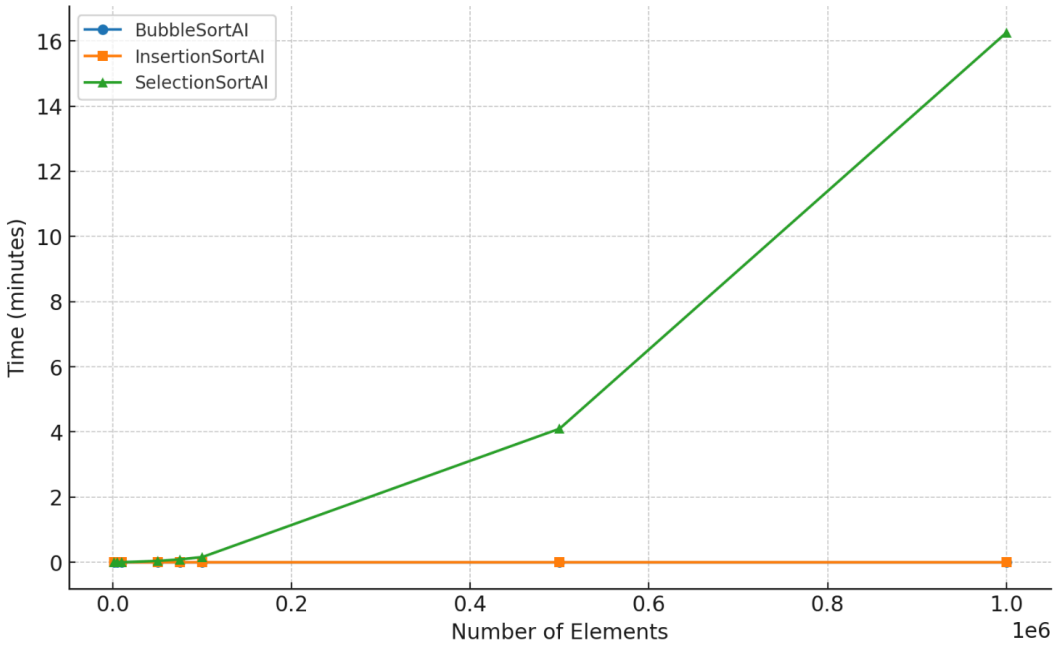

Figure 2

Unsorted Arrays Performance Comparison: BubbleSortAI, InsertionAI, SelectionSortAI