Performance Analysis of the Insertion Sort Algorithm

Talgat Sailaubekov

October 5, 2025

1. Objective

The goal of this work is to analyze the performance of the **Insertion Sort** algorithm based on the execution time for various input sizes and data orderings. The algorithm was tested on four types of data:

- *Random* — completely random order of elements;
- *Sorted* — elements already sorted in ascending order;
- *Reversed* — elements sorted in descending order;
- *Nearly-sorted* — elements mostly sorted with a few random swaps.

2. Methodology

Experiments were conducted on arrays with sizes of $n = 100$, 1000, and 5000. Execution time was measured using the `PerformanceTracker` class, and the results were saved to the CSV file `insertion_results.csv`. The data was then used to generate a performance plot.

3. Results and Analysis

The figure below illustrates the relationship between input size (n) and execution time for different input types.

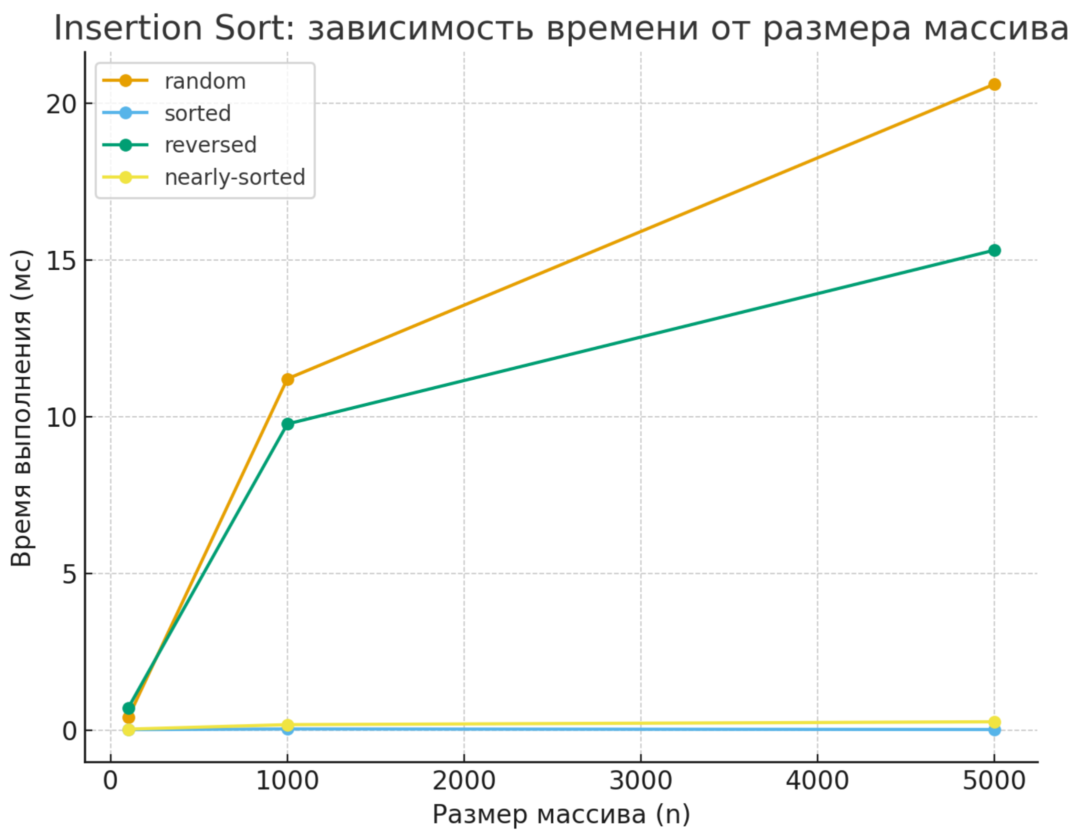

Figure 1: Execution time vs. array size for different input data types

As expected:

- For **sorted** data, the algorithm performs almost linearly $O(n)$;
- For **random** and **reversed** data, the time complexity is quadratic $O(n^2)$;
- For **nearly-sorted** data, performance is between the two extremes.

4. Conclusion

The experiment confirms that **Insertion Sort** performs efficiently for small or nearly sorted datasets, but is inefficient for large, unordered data due to its quadratic complexity. This analysis demonstrates the expected time complexity and behavior of the algorithm.