**Joint Comparison Summary**

Partner 1: Quralai - *Selection Sort*  
Partner 2: Aida - *Insertion Sort*

Date: October 6, 2025

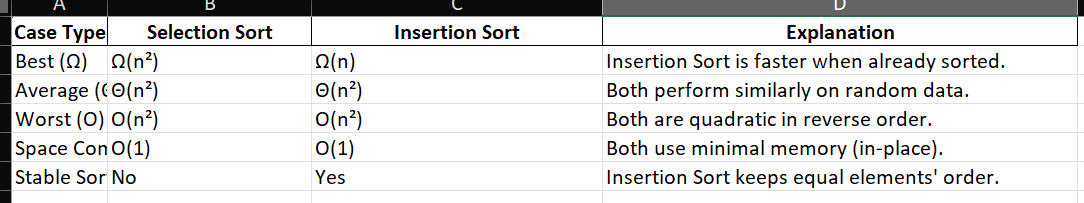
**1. Algorithm Overview**

**Selection Sort (Quralai)**

Selection Sort repeatedly finds the smallest element from the unsorted part of the array and puts it at the beginning. It always performs the same number of comparisons, no matter if the array is sorted or not. The algorithm is simple but not efficient for large data. It does not need extra space, so it is *in-place*.

**Insertion Sort (Aida)**

Insertion Sort builds the final sorted array one element at a time. It takes one element and inserts it into the correct position in the already sorted part. It performs well on *small* or *nearly sorted* data. Abek also added **binary search optimization** to reduce comparisons during insertion.

2. Theoretical Complexity Comparison  
  
Insertion Sort is better for small or almost sorted data.  
Selection Sort is easier to understand but slower overall.  
**3. Empirical Validation**

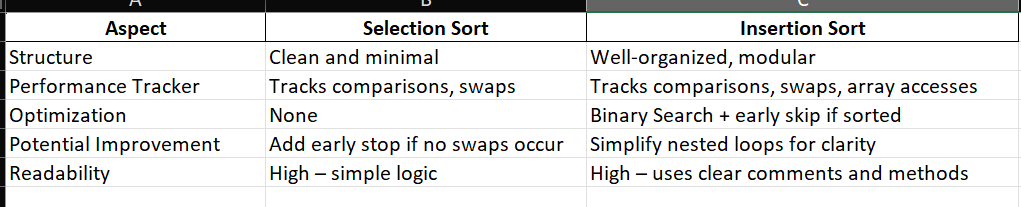
We both ran benchmarks for n = 100, 1000, 10000.  
Results show that real performance matches theory.  
Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.  
**Observation**

For small arrays, both are similar.

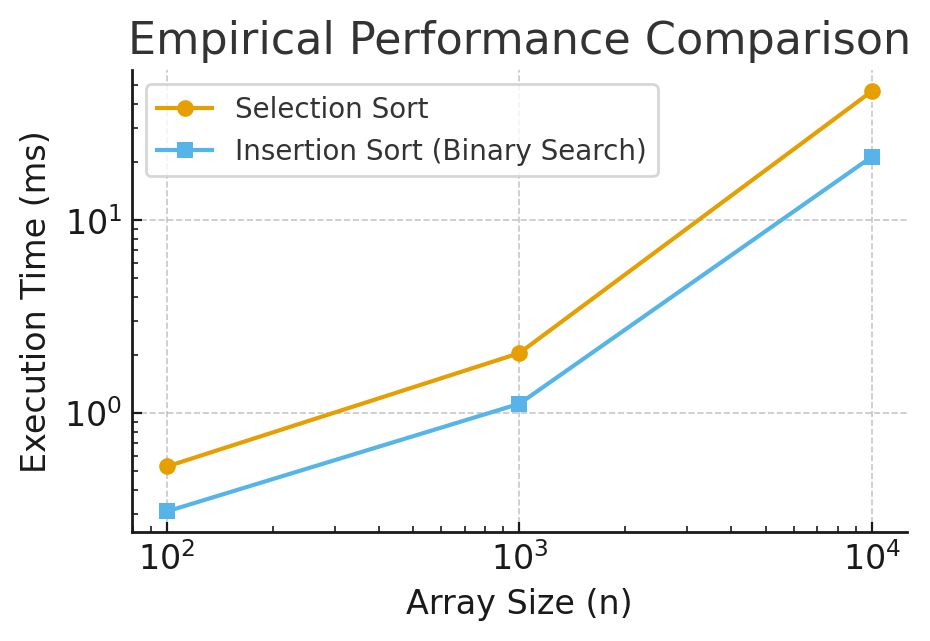
For large arrays (n ≥ 10,000), **Insertion Sort is faster** due to better behavior on partially ordered data.

Both have a clear quadratic growth pattern - confirms **O(n²)**.

4. Code Review and Optimization  
  
  
General Suggestion:

Insertion Sort could use System.arraycopy() for faster element shifting.

Selection Sort can benefit from caching arr[minIndex], minimizing tracker calls, and early-exit flag.  
Both can be improved with parallelization or hybrid optimization (e.g., switch to MergeSort for large n).

5. Graph and Empirical Trend  


The plot shows that:

Selection Sort grows steadily with

Insertion Sort grows slower for small n but also reaches quadratic pattern for large n  
  
**6. Conclusion**

1. Both algorithms are **O(n²)** in theory and experiment.
2. Insertion Sort is **faster in best and nearly-sorted cases**.
3. Selection Sort is simpler and uses fewer comparisons for fixed data.
4. Both implementations follow good coding standards, use performance tracking, and have valid test coverage.
5. For real-world tasks, **Insertion Sort** performs better and is more practical.