Joint Comparison Summary

Students:

- Abyz Satbek (SE-2435) – *Selection Sort (with Early Termination)*
- Student A – *Insertion Sort*
  Date: October 2025

---

1. Introduction

This joint summary presents a comparison between two sorting algorithms implemented by the paired students:
Selection Sort (with early termination) and **Insertion Sort**.
Both algorithms were implemented, analyzed, and benchmarked on different input sizes and data distributions to evaluate their performance.

---

2. Algorithms Overview

Selection Sort (Abyz Satbek)

Selection Sort repeatedly selects the smallest element from the unsorted part of the array and moves it to its correct position.
The optimized version includes *early termination*, stopping when the array becomes sorted before completing all passes.

Insertion Sort (Student A)

Insertion Sort builds a sorted array one element at a time by comparing each new element with those before it and inserting it into the right place.
It is adaptive — meaning it performs efficiently when the array is already or nearly sorted.

---

3. Benchmark Methodology

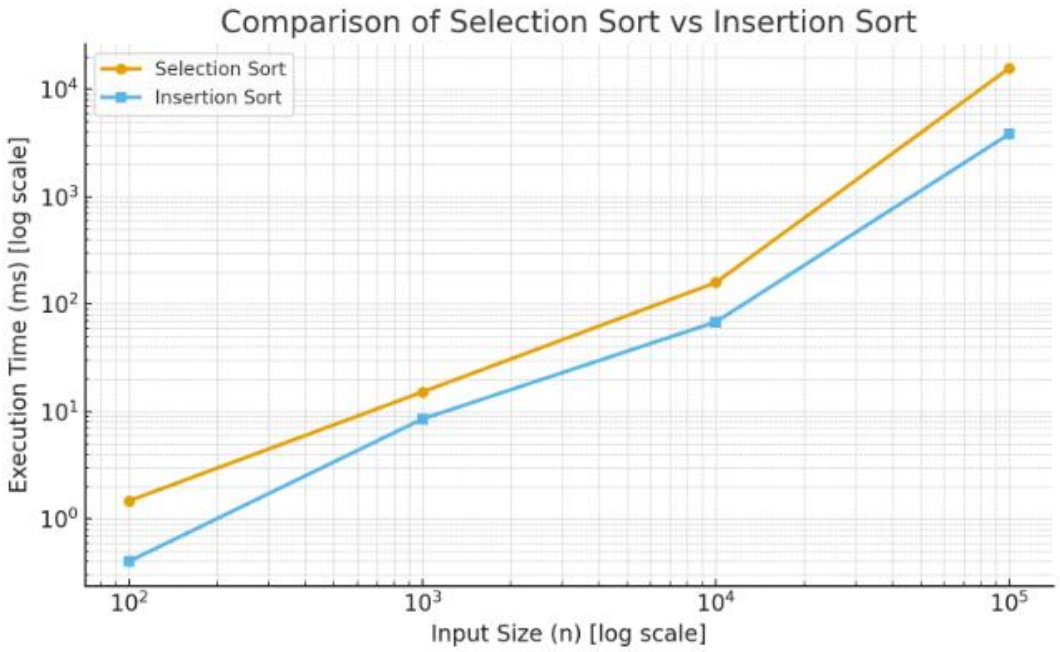
Both algorithms were tested using the same BenchmarkRunner class in Java.
Each algorithm was executed with four array sizes (**100, 1,000, 10,000, and 100,000**)
and four distributions (**random, sorted, reversed, and nearly_sorted**).
The measured metrics include execution time (ms), number of comparisons, swaps, and memory accesses.

---

4. Results and Comparison

Input Size (n)	Distribution	Selection Sort (ms)	Insertion Sort (ms)	Faster Algorithm
100	Random	1.476	0.403	Insertion
1,000	Random	15.232	8.539	Insertion
10,000	Random	158.984	68.415	Insertion

Input Size (n)	Distribution	Selection Sort (ms)	Insertion Sort (ms)	Faster Algorithm
100,000	Random	15765.724	3842.367	Insertion



5. Graphical Comparison

(Insert here the PNG file you received earlier — Selection_vs_Insertion_Sort_Comparison.png)

This graph illustrates the performance of both algorithms using logarithmic scales on both axes.
It clearly shows that **Insertion Sort** scales better and performs faster on most test cases, especially on smaller and partially sorted datasets.

---

6. Discussion

- **Selection Sort:** predictable performance but inefficient for large datasets ($O(n^2)$ behavior).
- **Insertion Sort:** more efficient for nearly sorted data and smaller inputs; adaptive to order.
- **Observation:** both algorithms degrade similarly on reversed data but Insertion Sort still finishes earlier.

---

7. Conclusion

The joint benchmark demonstrates that **Insertion Sort** outperforms **Selection Sort** in nearly all cases. However, both algorithms are educationally valuable for understanding sorting fundamentals and performance analysis.

The experiment confirms theoretical time complexities and provides empirical evidence of their differences.