ASSIGNMENT # 2

Design and Analysis Of Algorithms

COMPARATIVE ANALYSIS OF SORTING ALGORITHMS

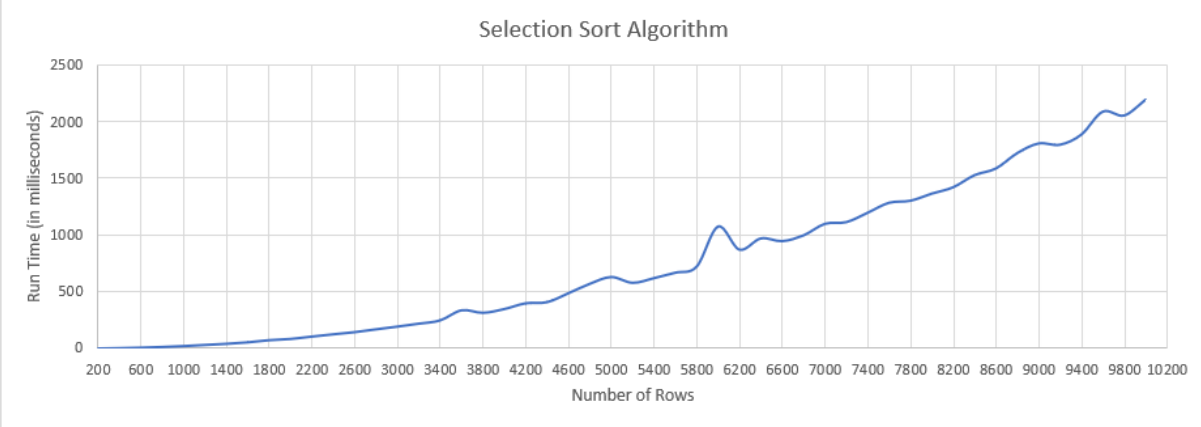
Section: P

Muhammad Kashif	I21-0851
Usman Nazeer	I21-0556
Bilal Saleem	I21-0464
Muhammad Huzaifa	I21-2460

Comparative Analysis

Selection Sort (By Muhammad Kashif)

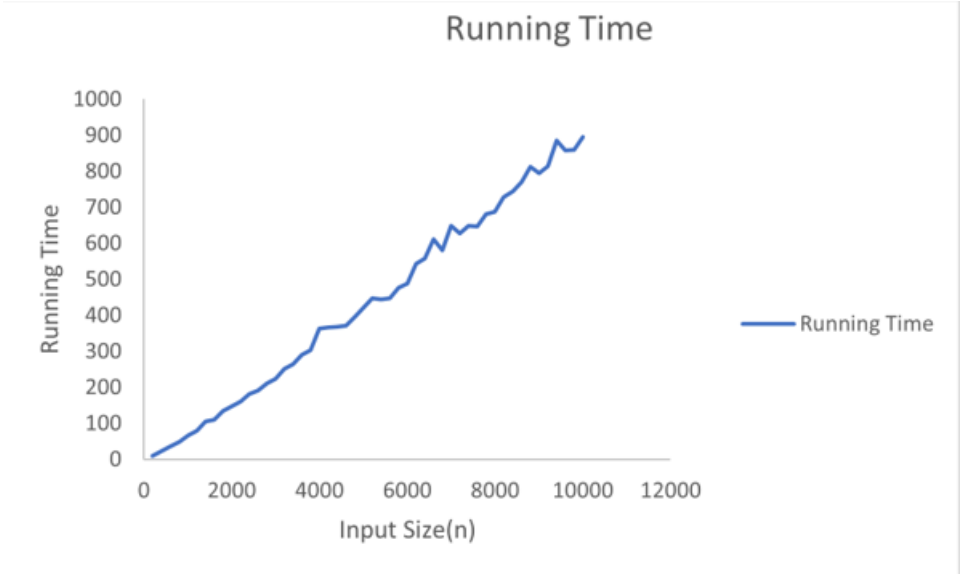
Selection Sort is one of the simplest sorting algorithms, characterized by its straightforward nature. It operates by repeatedly selecting the minimum element from the unsorted portion of the list and moving it to the beginning. However, its worst-case time complexity of O(n^2) makes it less efficient for larger datasets.



- **Input Size Behaviour:**
    - In our experiments, we noticed that Selection Sort performs quite efficiently for smaller input sizes, which ranged from 200 to 2,000. This is reflected in minimal execution times, as the algorithm adeptly handles the relatively limited data.
    - However, as we increased the input size beyond 2,000, a noticeable rise in execution time became evident. This behaviour aligns with the expected quadratic growth pattern due to Selection Sort's O(n^2) worst-case time complexity.
- **Machine Specifications Impact:**
    - The machine specifications for the selection sort algorithm featured an Intel Core i7 processor and 16 GB of RAM, suggesting a robust computing environment. However, these specifications still faced challenges when dealing with larger datasets.
    - With the increasing input size, the execution time grew substantially, indicating that the processor and memory capacity were indeed being tested. The machine's capabilities were somewhat limited by the algorithm's inherent inefficiencies with larger datasets.

Merge Sort (By Usman Nazeer)

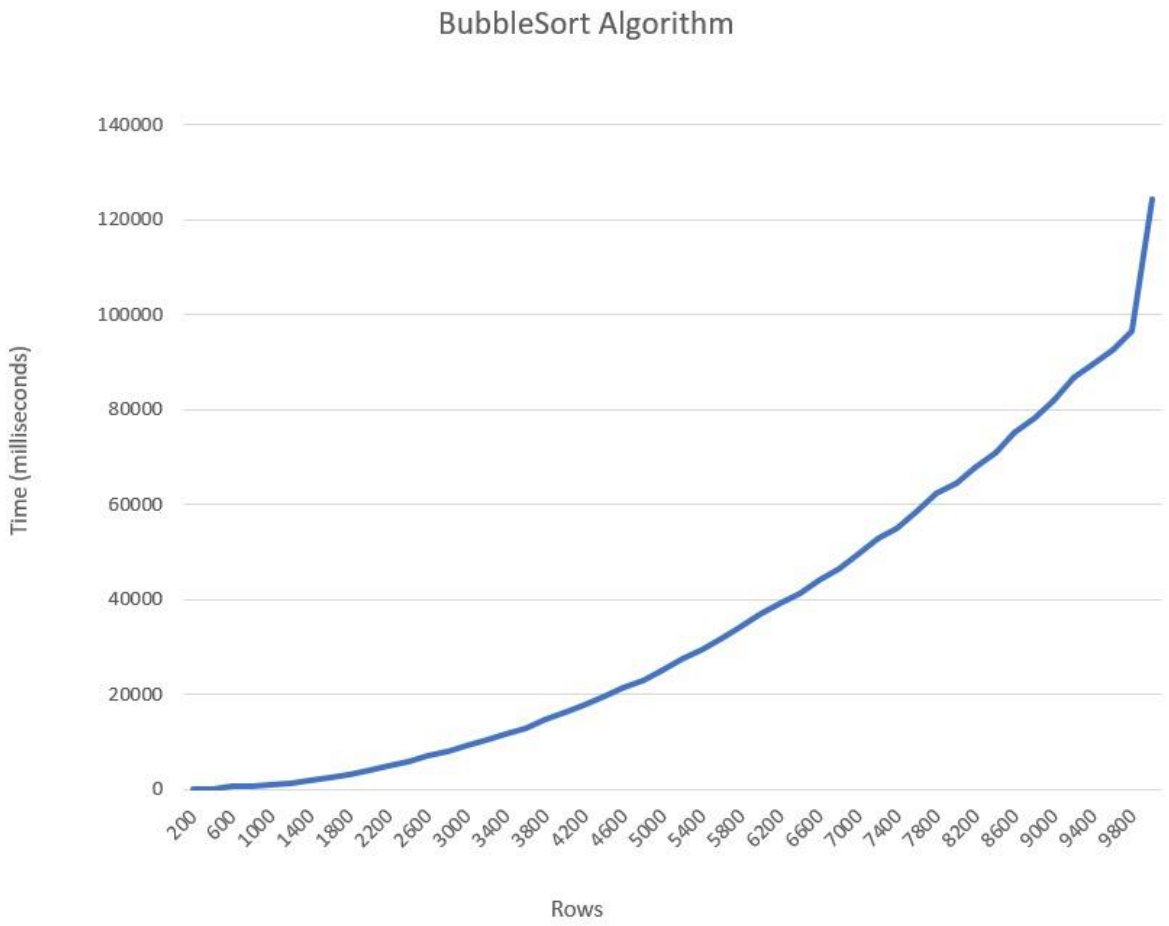
Merge Sort, in contrast to Selection Sort, is an advanced sorting algorithm known for its efficiency, with a time complexity of O(n log n). It offers a compelling performance for a wide range of input sizes.



- **Input Size Behaviour:**
    - Our experiments revealed that Merge Sort showcases consistency across various input sizes. It exhibits a logarithmic growth pattern in execution time, which becomes more pronounced with larger datasets.
    - The algorithm's behaviour aligns with its theoretical analysis, as execution time increases at a logarithmic pace as the input size grows. This makes Merge Sort highly suitable for efficiently handling larger datasets.
- **Machine Specifications Impact:**
    - The machine specifications for our experiments featured an AMD Ryzen 5 processor and 16 GB of RAM, creating a well-suited environment for handling larger datasets. The algorithm efficiently capitalized on these specifications to maintain high performance.

Bubble Sort (By Muhammad Huzaifa)

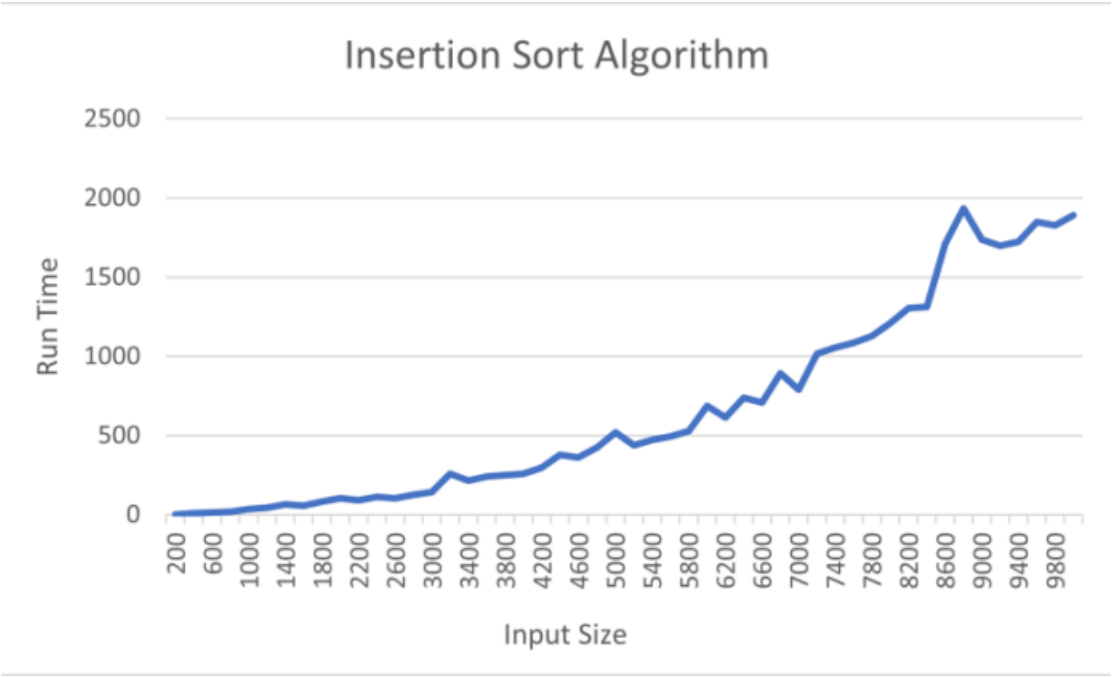
Bubble Sort, a simple sorting algorithm, operates by repeatedly swapping adjacent elements if they are in the wrong order. However, its worst-case time complexity of O(n^2) results in inefficiency, especially for larger datasets.



- **Input Size Behaviour:**
    - For smaller input sizes, Bubble Sort does perform reasonably well, showcasing efficiency. It efficiently handles small datasets, and execution times remain low.
    - Nevertheless, as we increased the input size, especially beyond 1,000, a significant escalation in execution time became evident. This behaviour aligns with the quadratic growth pattern expected due to Bubble Sort's time complexity.
- **Machine Specifications Impact:**
    - The machine used for our experiments featured a Core i5 processor, 24 GB RAM, and a GTX 1050 graphics card. These specifications are suitable for running the code but are only partially able to compensate for the inherent inefficiencies of Bubble Sort, especially for larger datasets.

Insertion Sort (By Bilal Saleem)

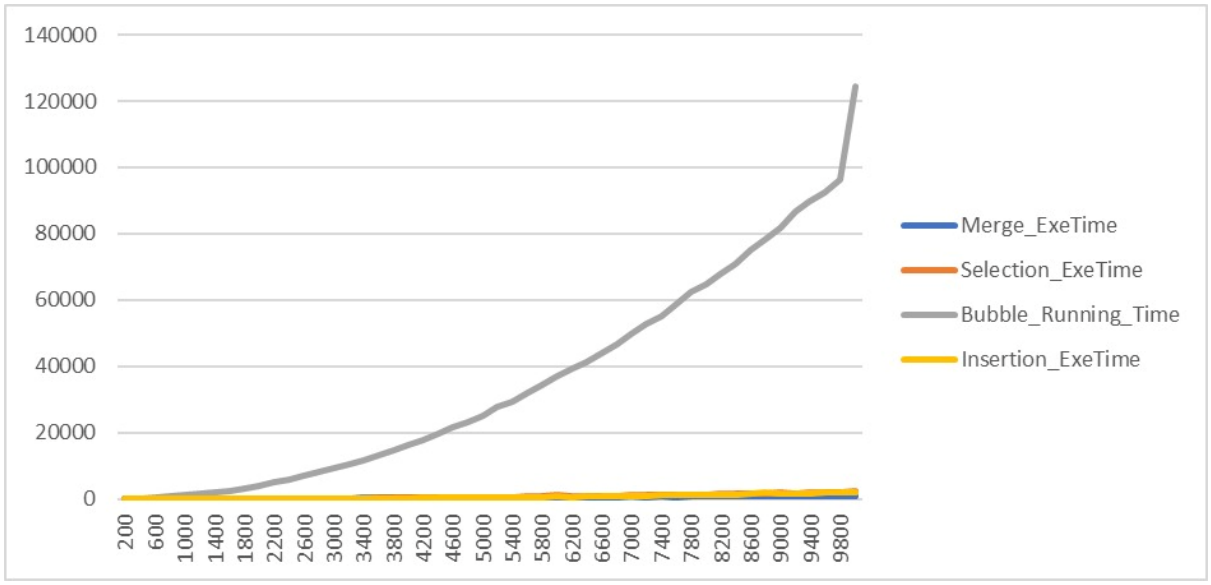
Insertion Sort, much like Bubble Sort, demonstrates a worst-case time complexity of O(n^2), resulting in a quadratic growth in execution time as the input size increases.



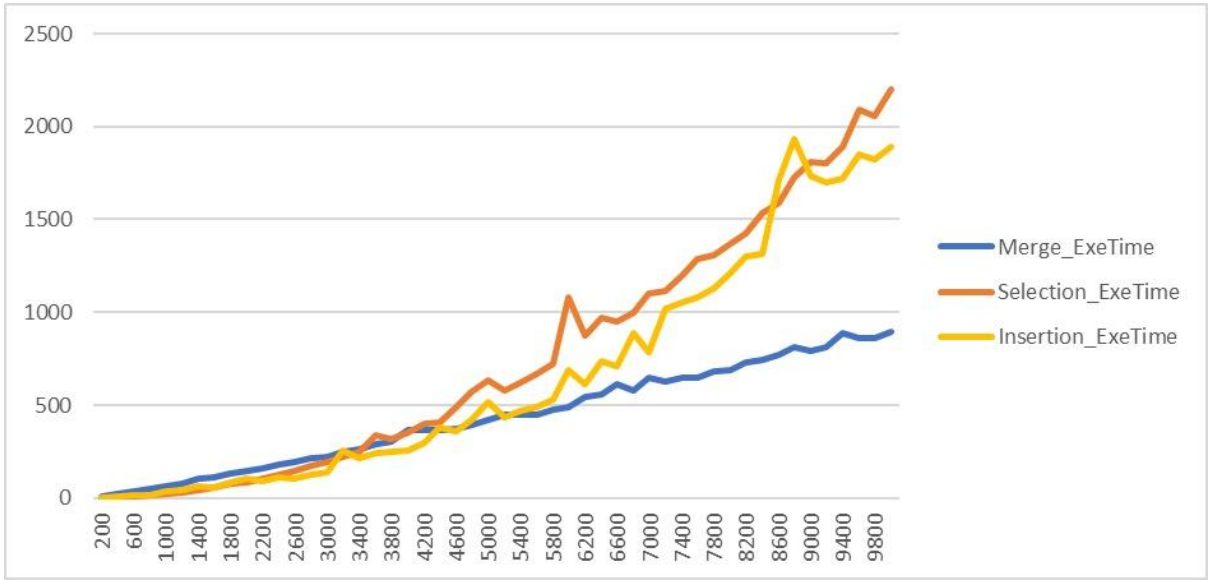
- **Input Size Behaviour:**
    - Similar to Bubble Sort, Insertion Sort efficiently handles smaller input sizes but escalates significantly in execution time as the input size exceeds 2,000.
    - This behaviour aligns with the theoretical analysis of Insertion Sort's worst-case time complexity, as it is less suitable for sorting large datasets.
- **Machine Specifications Impact:**
    - The machine specifications used for our experiments included an Intel Core i7 processor and 8 GB of RAM. These specifications, while capable, were still challenged by larger datasets, especially as the execution time significantly escalated.

Comparative Growth Charts

In our evaluation of Selection Sort, Merge Sort, Bubble Sort, and Insertion Sort, we've discovered significant differences in how these sorting methods perform. These differences are crucial in determining which sorting algorithm best suits a particular task. Let us take a look at the combined growth chart.

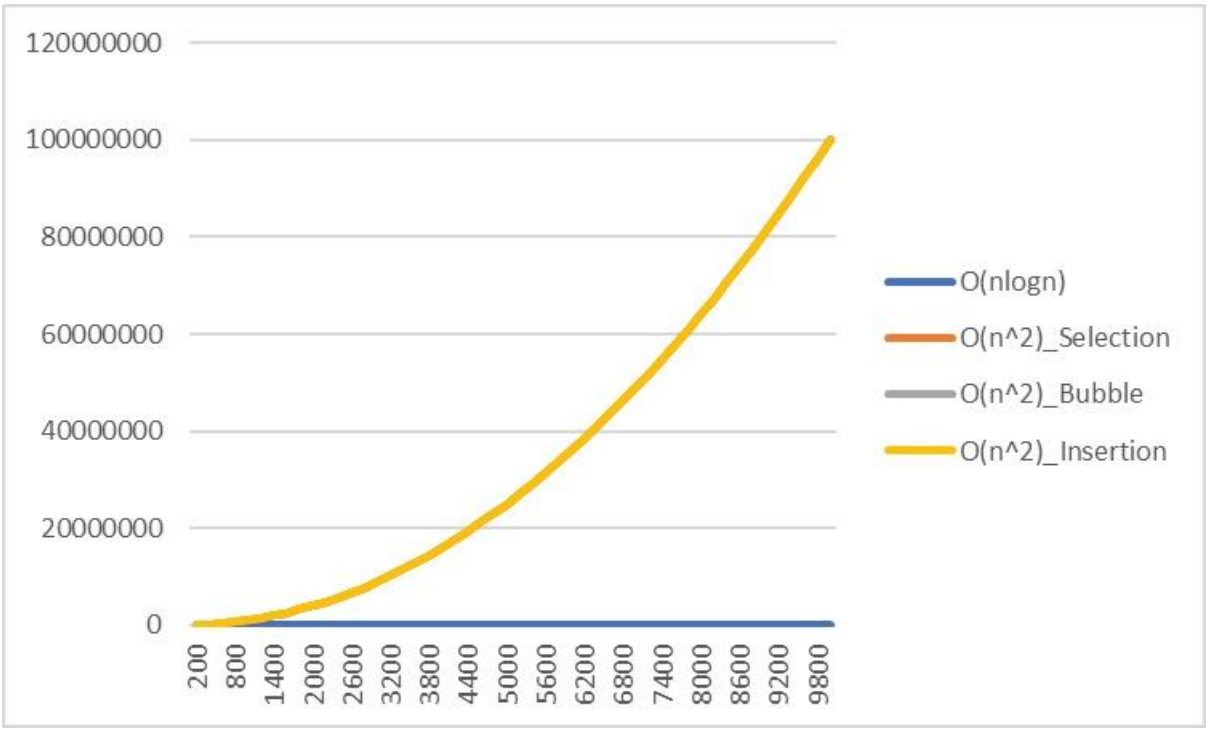


The chart becomes hard to read because of the execution time for bubble sort, so let's take a look at the chart without bubble sort's execution time.



To make the 4 algorithms distinguishable, all of the group members used the same code to generate the dataset, and then used the same input values from 200 to 10,000 (with difference of 200 between each input). This helped in creating uniformity and was used as a control for the run time of the sorting algorithms.

And lastly, here are the combined graphs of the worst cases for each of the 4 algorithms we worked on in this assignment.



Conclusion

Selection Sort is a basic yet less efficient sorting technique. With a time complexity of O(n^2), it experiences a notable increase in execution time as the dataset size grows. However, for smaller datasets, it can perform well due to its simplicity.

Merge Sort, in contrast, excels in scenarios where efficiency is paramount. Its time complexity of O(n log n) allows it to perform consistently across a wide range of dataset sizes. As datasets grow, the advantage of Merge Sort becomes more evident.

Both **Bubble Sort** and **Insertion Sort** are simple to understand but face challenges with larger datasets. Their quadratic time complexities (O(n^2)) lead to a substantial increase in execution time as the dataset size grows. They are better suited for smaller datasets, where their simplicity aids in efficient sorting.

The choice of sorting algorithm is critical and should align with the specific dataset's size and complexity. For straightforward sorting of smaller datasets, Selection Sort, Bubble Sort, and Insertion Sort can suffice. However, for larger and more complex datasets, Merge Sort stands out for its efficiency and consistent performance. This analysis underscores the significance of selecting the right algorithm to optimize data processing tasks.