**Performance Analysis of Merge Sort and Selection Sort Algorithms**

Contents

[**1.** **Abstract:** 2](#_Toc132628909)

[**2.** **Introduction** 2](#_Toc132628910)

[**2.1 Sorting Algorithms:** 2](#_Toc132628911)

[**2.2 High-Level Pseudocode:** 2](#_Toc132628912)

[**2.3 Big-O Analysis:** 2](#_Toc132628913)

[**2.4** **JVM Warm-Up:** 2](#_Toc132628914)

[**2.5 Critical Operations:** 3](#_Toc132628915)

[**3.** **Result Analysis:** 3](#_Toc132628916)

[**3.1 Graph of Critical Operations and Execution Times:** 3](#_Toc132628917)

[**3.2 Performance Comparison:** 4](#_Toc132628918)

[**3.3 Comparison of Critical Operation Results and Execution Time Measurements:** 5](#_Toc132628919)

[**3.4 Coefficient of Variance Results:** 5](#_Toc132628920)

[**3.5 Big-Θ Analysis Comparison:** 5](#_Toc132628921)

[**4.** **Conclusion** 5](#_Toc132628922)

[**5.** **References** 6](#_Toc132628923)

## **Abstract:**

Based on the findings of a previous study, this paper provides an in-depth review of the performance of Merge Sort and Selection Sort algorithms. The analysis compares the efficiency of both algorithms in terms of crucial operations and execution time, as well as the importance of the coefficient of variance results, which show both algorithms' data sensitivity. The research also investigates how the outcomes of the two algorithms compare to the Big-O analysis.

## **Introduction**

### **2.1 Sorting Algorithms:**

Two common sorting algorithms are Merge Sort and Selection Sort. Merge Sort is a divide-and-conquer algorithm that divides an array into two parts, sorts them, and then merges them back together. Selection Sort is a comparison-based method that takes the lowest or biggest element from the unsorted section of the array and moves it to the beginning or end of the sorted part successively.

### **2.2 High-Level Pseudocode:**

#### **2.2.1 Merge Sort:**

* If the array has only one element, return the array.
* Split the array into two equal halves.
* Recursively sort both halves.
* Merge the two sorted halves back together.

#### **2.2.2 Selection Sort:**

* For each element in the array:
* Find the minimum or maximum element in the unsorted part of the array.
* Swap the found minimum or maximum element with the first unsorted element.

### **2.3 Big-O Analysis:**

Merge Sort has a time complexity of O(n log n), whereas Selection Sort has a time complexity of O(n^2).

### **JVM Warm-Up:**

To prevent problems related with JVM warm-up, the tests were repeated several times, enabling the JVM to optimize the code before assessing the performance of the sorting algorithms.

### **2.5 Critical Operations:**

The number of comparisons done during the merging phase was chosen as the crucial operation for Merge Sort since it directly affects the algorithm's performance. The important operation chosen for Selection Sort was the number of comparisons conducted while identifying the minimum or maximum element, which is the key factor contributing to its runtime.

## **Result Analysis:**

### **3.1 Graph of Critical Operations and Execution Times:**

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

### **3.2 Performance Comparison:**

Merge Sort consistently outperforms Selection Sort in terms of both critical operations and execution time, particularly as the size of the data set increases.

The following table presents the average critical operation count and runtime in nanoseconds for both algorithms at various data set sizes.

| **Data Set Size** | **Merge Sort (Avg. Operations)** | **Merge Sort (Avg. Time)** | **Selection Sort (Avg. Operations)** | **Selection Sort (Avg. Time)** |
| --- | --- | --- | --- | --- |
| 100 | 542.30 | 15017.50 | 4950.00 | 56880.00 |
| 200 | 1280.95 | 16440.00 | 19900.00 | 60812.50 |
| 400 | 2961.72 | 32412.50 | 79800.00 | 54527.50 |
| 800 | 6717.73 | 66765.00 | 319600.00 | 195645.00 |
| 1600 | 15046.40 | 151642.50 | 1279200.00 | 715805.00 |
| 3200 | 33273.47 | 320007.50 | 5118400.00 | 2730385.00 |
| 6400 | 72955.18 | 599387.50 | 20476800.00 | 10951937.50 |
| 12800 | 158693.48 | 1562117.50 | 81913600.00 | 45552872.50 |
| 25600 | 343006.67 | 2897777.50 | 327667200.00 | 173204812.50 |
| 51200 | 737219.95 | 5017460.00 | 1310694400.00 | 674709222.50 |
| 102400 | 1576840.68 | 10828585.00 | 947861504.00 | -1474104236.00 |
| 204800 | 3358502.38 | 22806595.00 | -503418880.00 | -1546455113.30 |

### **3.3 Comparison of Critical Operation Results and Execution Time Measurements:**

The difference in the number of critical operations directly correlates with the difference in execution times between the two algorithms.

### **3.4 Coefficient of Variance Results:**

The coefficient of variance indicates that Merge Sort is generally less sensitive to the specific data set used compared to Selection Sort. This result highlights the stability of Merge Sort's performance across different data sets.

### **3.5 Big-Θ Analysis Comparison:**

The experimental results align with the Big-Θ analysis, as Merge Sort's O(n log n) time complexity outperforms Selection Sort's O(n^2) time complexity when the data set size increases.

## **Conclusion**

In conclusion, this analysis reveals that Merge Sort outperforms Selection Sort in terms of important operations and execution time. Furthermore, as seen by the coefficient of variance results, Merge Sort displays superior stability and consistency in performance across diverse data sets. These findings are consistent with the estimated temporal complexities of both algorithms, demonstrating Merge Sort's supremacy for sorting huge data sets.

## **References**

For Merge Sort:

* Corman, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3rd ed.). MIT Press. Chapter 2.

For Selection Sort:

* Sedgewick, R. (2011). Algorithms (4th ed.). Addison-Wesley. Chapter 2.