**CT – Bubble and Merge Sort Analysis**

Dwight Abrahams

CSU Global

CSC506-1: Principles of Software Development

Professor Pubali Banerjee

March 02, 2025

**Bubble and Merge Sort Analysis**

## **Introduction**

Efficient data processing is paramount in healthcare, where timely access to patient records can have direct clinical implications. This analysis explores the algorithmic underpinnings of two sorting methods—Bubble Sort and Merge Sort—evaluating their theoretical time complexities, optimal operating conditions, and practical performance in managing medical records. The discussion further justifies the use of specific data structures and examines external factors influencing algorithmic efficiency and theoretical lower bounds.

## Time Complexity and Performance

Bubble Sort is characterized by iterative pairwise comparisons and exhibits a worst-case complexity of *O(n²)*. Although an optimized version may achieve *O(n)* under nearly sorted conditions, its performance deteriorates rapidly as dataset size increases (Cormen et al., 2009). In contrast, Merge Sort utilizes a divide-and-conquer strategy, consistently operating at *O(n log n)* regardless of initial data ordering. Benchmarking results reveal that Merge Sort consistently outperforms Bubble Sort—with average execution times of approximately 0.000697 seconds versus 0.004567 seconds—thereby demonstrating its scalability and robustness in processing large-scale medical data (Cormen et al., 2009).

## Optimal Conditions and Data Structure Justification

For applications such as daily appointment logs where records may be nearly sorted, Bubble Sort can be effective. However, given the unpredictability and volume of healthcare data—where sudden influxes of new patient records are common—Merge Sort’s stable and predictable performance is preferable. Arrays were chosen as the primary data structure due to their direct memory access and efficient support for indexed operations. This choice complements Merge Sort’s recursive partitioning process despite incurring additional memory overhead during merging (Cormen et al., 2009). Moreover, hardware capabilities and memory availability are critical external factors that enable Merge Sort to perform near the theoretical lower bound of *Ω(n log n)* for comparison-based sorting.

## Development Challenges and Acquired Expertise

The development process presented multiple obstacles, including precise timing of execution, handling irregularities in record data (e.g., duplicates, missing fields), and optimizing recursive function calls to manage memory effectively. Addressing these challenges fostered advanced skills in algorithmic analysis, performance optimization, and rigorous software testing—all critical in developing reliable healthcare information systems. These competencies are instrumental in translating theoretical knowledge into practical solutions that enhance clinical decision-making and operational efficiency.

## Conclusion

In summary, while Bubble Sort may perform adequately on small or nearly sorted datasets, its quadratic time complexity renders it impractical for most healthcare applications. Merge Sort, with its consistent *O(n log n)* behavior, is better suited for the large, dynamic datasets typical in medical records management. This analysis not only reinforces the importance of selecting appropriate algorithms and data structures but also highlights the broader implications of computational efficiency in healthcare.

# References

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

# Appendix A

A screenshot of a computer

Description automatically generated

*Figure 1. Program execution showcasing successful run times for Bubble Sort and Merge Sort.*