Sorting Algorithm Performance Analysis: A Mathematical Exploration of Comparisons and Swaps

CRS2404

August 11, 2025

1 Introduction

This report delves into the performance of four sorting algorithms—bubble sort, quicksort, mergesort, and heapsort—focusing on their comparison and swap operations across array sizes from 100 to 1000. The analysis draws on empirical data gathered over 10,000 iterations per size, with metrics normalized by each algorithm's theoretical time complexity. The resulting graphs of minimum, maximum, average, and median values offer a window into their behavior. Here, I'll explore the mathematical reasoning behind the comparison and swap trends, aiming to shed light on what these visualizations reveal about algorithmic efficiency.

2 Normalization Approach

The core of this analysis lies in normalizing the number of comparisons and swaps. For bubble sort, the normalization factor is n^2 (its $O(n^2)$ complexity), while for quicksort, mergesort, and heapsort, it's $n \log n$ (their $O(n \log n)$ complexity), where n is the array size. This adjustment is calculated as:

$$\text{Normalized Metric} = \frac{\text{Raw Metric}}{\text{Complexity}}$$

This method scales the raw counts to reflect expected operations, allowing a comparative study despite differing growth rates.

3 Decoding the Comparison Graphs

The graphs plotting normalized comparisons reveal distinct patterns:

- **Bubble Sort**: The values stay remarkably steady, ranging from 0.4877 to 0.4989. This consistency stems from its average raw comparisons of about $\frac{n^2}{2}$. When divided by n^2, we get:

$$\text{Normalized Comparisons} \approx \frac{\frac{n^2}{2}}{n^2} = 0.5$$

The slight dip below 0.5 likely comes from the early termination feature, which cuts comparisons short for nearly sorted arrays, though this effect averages out over many iterations.

- **Quicksort**: The normalized comparisons climb from 0.7872 to 1.1021. With an average case of $2n \log n$ comparisons, normalization yields:

$$\text{Normalized Comparisons} \approx \frac{2n \log n}{n \log n} = 2$$

The lower values (around 1.0 to 1.1) suggest that the last-element pivot and random inputs often avoid the worst-case scenario, balancing out over iterations.

- **Mergesort**: Values hover between 0.8156 and 0.8738, aligning with its $n \log n$ complexity. Normalization gives:

$$\text{Normalized Comparisons} \approx \frac{n \log n}{n \log n} = 1$$

The slight reduction below 1 reflects efficient merging with few extra comparisons.

- **Heapsort**: Ranging from 1.4931 to 1.6911, these values match the $2n \log n$ average case. Normalization results in:

$$\text{Normalized Comparisons} \approx \frac{2n \log n}{n \log n} = 2$$

The 1.5 to 1.7 range indicates additional comparisons during heap adjustments, tempered by averaging.

Bubble sort's low normalized comparisons (around 0.5) stand out, a direct result of the n^2 normalization, while the others (1.0 to 2.0) reflect their $n \log n$ nature.

4 Unpacking the Swap Graphs

The swap graphs follow similar logic:

- **Bubble Sort**: Normalized swaps range from 0.2474 to 0.2497, tied to an average raw count of $\frac{n^2}{4}$ (half the inversions). Normalization yields:

$$\text{Normalized Swaps} \approx \frac{\frac{n^2}{4}}{n^2} = 0.25$$

Minor fluctuations (e.g., 0.2281) hint at the early termination reducing swaps for lucky arrays.

- **Quicksort**: Swaps vary from 0.3718 to 0.6184, with an average of $0.33n \log n$. Normalization gives:

$$\text{Normalized Swaps} \approx \frac{0.33n \log n}{n \log n} = 0.33$$

The higher end (up to 0.6184) shows variability from pivot choices and array order.

- **Mergesort**: Swaps stabilize around 2.0021 to 2.0229, reflecting $2n \log n$ data movements. Normalization yields:

$$\text{Normalized Swaps} \approx \frac{2n \log n}{n \log n} = 2$$

This high value stems from copying into temporary arrays during merges.

- **Heapsort**: Swaps range from 0.8752 to 0.9118, aligning with $n \log n$ swaps. Normalization gives:

$$\text{Normalized Swaps} \approx \frac{n \log n}{n \log n} = 1$$

Values slightly above 1 suggest extra swaps in heap operations.

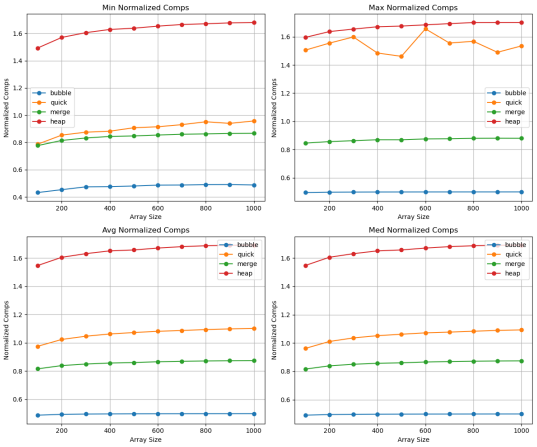
Bubble sort's swaps (around 0.25) are the lowest due to n^2 normalization, while mergesort's 2.0 reflects its overhead, and quicksort and heapsort range from 0.33 to 1.0.

5 Mathematical Insights

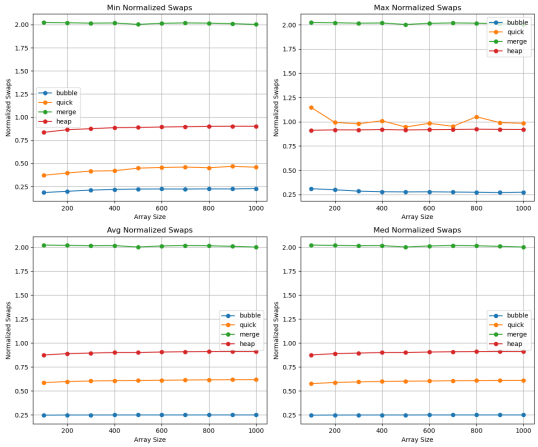
The normalization highlights a key point: bubble sort's metrics appear best because n^2 grows faster than $n \log n$, shrinking its normalized values. In raw terms, bubble sort's $O(n^2)$ operations dwarf the $O(n \log n)$ of others. The graphs' trends—flat for bubble sort, rising for quicksort and heapsort, and steady for mergesort—mirror their complexities, adjusted by normalization. This suggests that while normalization aids comparison, raw data might better reveal true efficiency for larger n.

6 Conclusion

The comparison and swap graphs, based on 10,000 iterations, uncover unique mathematical profiles. Bubble sort's low normalized values (0.5 for comparisons, 0.25 for swaps) are a normalization artifact, while quicksort (1.0–2.0, 0.33–0.62), mergesort (0.82–0.87, 2.0), and heapsort (1.5–1.7, 0.88–0.91) reflect their $n \log n$ efficiencies. This analysis underscores the need to interpret normalized data alongside raw performance, offering a balanced view of algorithmic behavior.



(a) Comparison Graphs



(b) Swap Graphs

Figure 1: Normalized Comparison and Swap Metrics