BubbleSort and MergeSort Complexity Reflection

Sorting algorithms are fundamental in computer science, and their performance varies significantly depending on their time complexity and the size of the input data. In this analysis, we compare the performance of BubbleSort and MergeSort using three datasets: `sort10.txt`, `sort1000.txt`, and `sort10000.txt`. The results are visualized in the plot below, which shows the runtime (in milliseconds) of each algorithm for the given input sizes.

Complexity Analysis

BubbleSort is a simple sorting algorithm with a worst-case and average-case time complexity of $O(n^2)$. This quadratic growth makes it inefficient for large datasets, as the number of comparisons and swaps increases dramatically with input size. For small datasets, such as `sort10.txt`, BubbleSort performs well, completing in 0ms. However, as the input size increases to `sort1000.txt` and `sort10000.txt`, the runtime grows significantly, reaching 6907ms for the largest dataset.

In contrast, MergeSort is a divide-and-conquer algorithm with a worst-case and average-case time complexity of $O(n \log n)$. This logarithmic growth makes it much more efficient for larger datasets. For `sort10.txt` and `sort1000.txt`, MergeSort completes in 0ms, and for `sort10000.txt`, it takes only 29ms, a stark contrast to BubbleSort's performance.

Results Discussion

The results clearly demonstrate the impact of algorithmic complexity on performance. For small datasets, the difference between BubbleSort and MergeSort is negligible, as the overhead of MergeSort's recursive calls is not significant. However, as the input size grows, the quadratic complexity of BubbleSort leads to a dramatic increase in runtime, while MergeSort's logarithmic complexity ensures it remains efficient.

This comparison highlights the importance of selecting the appropriate sorting algorithm based on the size of the input data. While BubbleSort may be suitable for small datasets due to its simplicity, MergeSort is the clear choice for larger datasets where performance is critical.

BubbleSort and MergeSort Complexity Reflection