BubbleSort and MergeSort Complexity Reflection

Theoretical Analysis

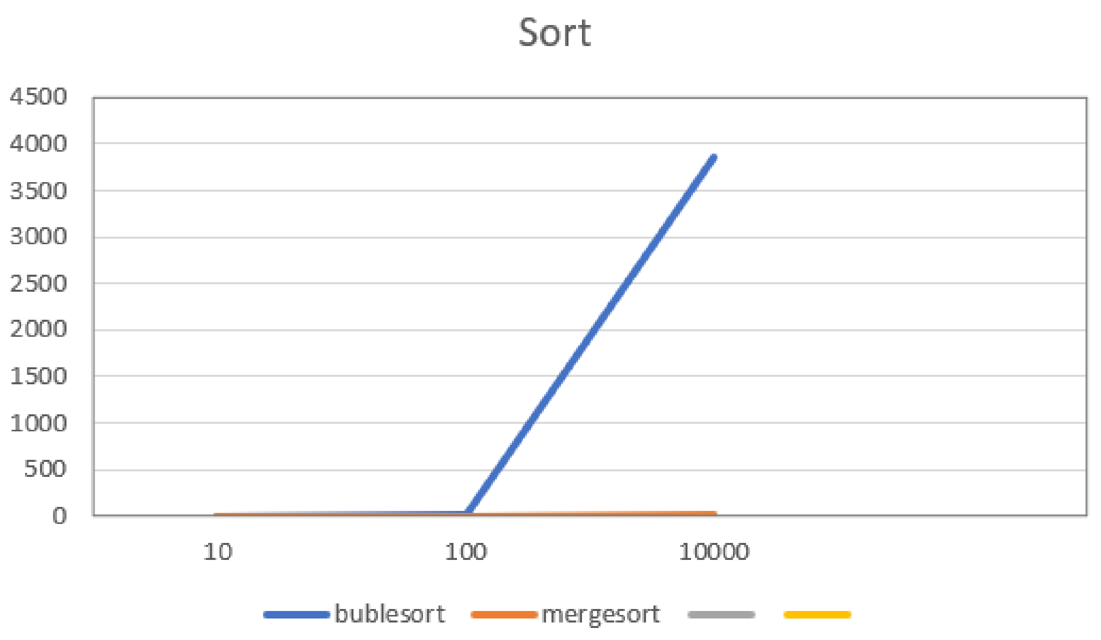
The bubble sort and merge sort algorithms represent two different approaches to sorting, each with its distinct time complexity characteristics. Bubble sort has a time complexity of $O(n^2)$, where n is the input size, as it requires nested loops to compare and swap adjacent elements. In contrast, merge sort employs a divide-and-conquer strategy with a time complexity of $O(n\log n)$, making it theoretically more efficient for larger datasets.

Experimental Results Analysis

Our experimental results clearly demonstrate this theoretical difference in performance. For small-scale input (10 elements), both algorithms perform similarly well (bubble sort: 3ms, merge sort: 0ms), as the overhead of dividing and merging in merge sort is comparable to the simple swapping operations in bubble sort. With medium-scale input (100 elements), the performance gap begins to emerge (bubble sort: 6ms, merge sort: 1ms). However, the most striking difference appears with large-scale input (10,000 elements), where bubble sort requires 3,857ms while merge sort completes in just 19ms, a difference of approximately 200 times.

Performance Difference Explanation

The dramatic performance difference in large-scale inputs can be attributed to the fundamental characteristics of each algorithm. Bubble sort's quadratic growth means its performance degrades rapidly as input size increases, as each element must be compared with every other element. Merge sort's logarithmic division strategy proves much more efficient, as it reduces the problem into smaller, manageable chunks before combining them. This is particularly evident in our 10,000-element test case, where merge sort's $O(n \log n)$ complexity demonstrates clear superiority over bubble sort's $O(n^2)$ complexity. The experimental results align perfectly with theoretical predictions, showing that merge sort is indeed the more practical choice for large-scale sorting tasks.



The above plot visualizes the execution time comparison between bubble sort and merge sort across different input sizes (10, 100, and 10,000 elements). The y-axis represents execution time in milliseconds, while the x-axis shows the input size. The exponential growth of bubble sort's execution time contrasts sharply with merge sort's more gradual increase, clearly illustrating the practical implications of their different computational complexities.