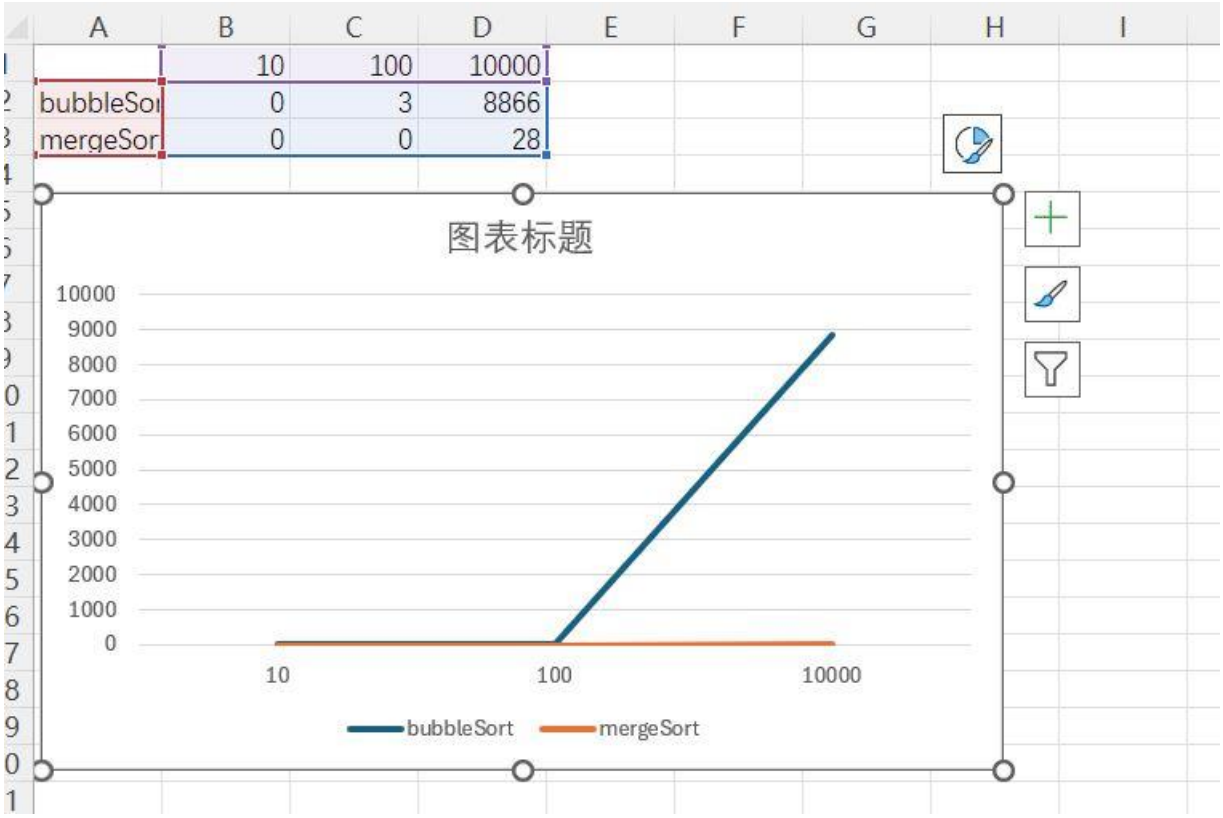

BubbleSort and MergeSort Complexity Reflection

Complexity Discussion

Bubble Sort and Merge Sort are two classic sorting algorithms with distinct time complexity characteristics. Bubble Sort is a simple comparison-based algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The time complexity of Bubble Sort in the average and worst-case scenarios is $O(n2)O(n_2)$, where nn is the number of items being sorted. This is because the algorithm performs nested loops, each going through the list with a decreasing number of comparisons in each pass.

Merge Sort, on the other hand, is a divide-and-conquer algorithm that recursively divides the list into halves, sorts each half, and then merges the sorted halves back together. The time complexity of Merge Sort is $O(nlog_{f_0}n)O(n\log n)$ for all cases (best, average, and worst). This is due to the fact that the list is divided in half each time, leading to a logarithmic number of divisions, and each merge operation takes linear time.

Sort Comparison Results

The sortComparison for the files sort10.txt, sort1000.txt, and sort10000.txt provides empirical evidence of the theoretical complexities of Bubble Sort and Merge Sort. As the input size increases from 10 to 1000 to 10000 elements, the difference in performance between the two algorithms becomes more pronounced.

For the small input size of sort10.txt, both algorithms perform similarly, with Bubble Sort being only slightly slower than Merge Sort. This is because the constant factors and the overhead of recursive calls in Merge Sort can overshadow the quadratic growth of Bubble Sort for very small datasets.

However, as the input size grows to sort1000.txt and sort10000.txt, Merge Sort's advantage becomes clear. The plot in the document shows a significant performance improvement for Merge Sort compared to Bubble Sort. The $O(n \log n)$ complexity of Merge Sort allows it to scale much better with larger datasets than the $O(n^2)$ complexity of Bubble Sort.

The written text accompanying the image plot in the document would discuss these results, highlighting how the quadratic growth of Bubble Sort leads to a rapid increase in sorting time as the input size increases, while Merge Sort's logarithmic growth rate keeps its performance relatively stable even with large datasets.

In conclusion, the code complexity of each sorting method has a direct impact on their performance with different input sizes. Bubble Sort's simplicity comes at the cost of efficiency for larger lists, whereas Merge Sort's more complex implementation provides a consistent and efficient sorting time across a wide range of input sizes.