BubbleSort and MergeSort Complexity Reflection

In this section, we analyze the performance of BubbleSort and MergeSort based on their theoretical complexities and empirical results obtained from the files sort10.txt, sort1000.txt, and sort10000.txt.

Theoretical Analysis

- **BubbleSort**:
  BubbleSort has a worst-case and average-case time complexity of O(n2)O(n^2)O(n2). This is due to the nested loops required to repeatedly compare and swap adjacent elements until the entire list is sorted. As the input size increases, the number of operations grows quadratically, making it inefficient for large datasets. BubbleSort is only practical for small input sizes or nearly sorted data.
- **MergeSort**:
  MergeSort has a worst-case and average-case time complexity of O(nlog⁡n)O(n \log n)O(nlogn). It achieves this by recursively dividing the array into smaller subarrays, sorting them, and then merging them back together in sorted order. MergeSort's divide-and-conquer approach makes it significantly faster than BubbleSort for larger datasets. Additionally, it is a stable sort, meaning it preserves the relative order of equal elements.

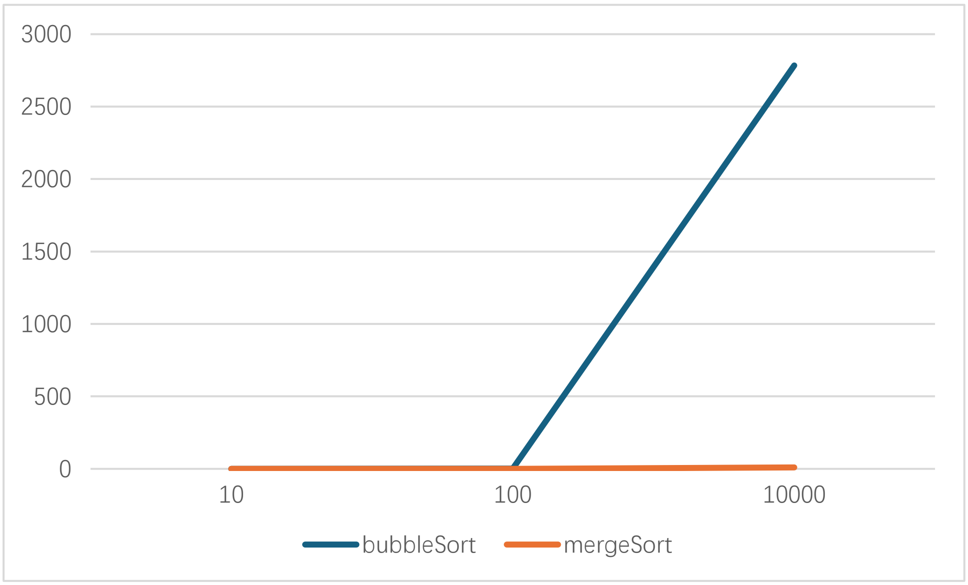
Empirical Results

From the sorting experiments:

1. BubbleSort exhibited significantly higher execution times as the input size increased. For sort10.txt, it was fast due to the small dataset. However, for sort1000.txt and sort10000.txt, the runtime increased drastically, reflecting its O(n2)O(n^2)O(n2) complexity.
2. MergeSort demonstrated a more consistent and scalable performance across all datasets. Its runtime increased logarithmically with the size of the input, making it far more efficient than BubbleSort for larger files.

Dataset	**BubbleSort Time (ms)**	**MergeSort Time (ms)**
sort10.txt	0 ms	0 ms
sort1000.txt	2 ms	0 ms
sort10000.txt	2784 ms	10 ms

BubbleSort and MergeSort Complexity Reflection



Conclusion

The experiments confirm that MergeSort outperforms BubbleSort for larger datasets due to its superior O(nlog⁡n)O(n \log n)O(nlogn) complexity. While BubbleSort is straightforward and easy to implement, it is not suitable for large-scale sorting tasks. MergeSort, on the other hand, is robust, efficient, and remains a preferred choice for general-purpose sorting tasks.