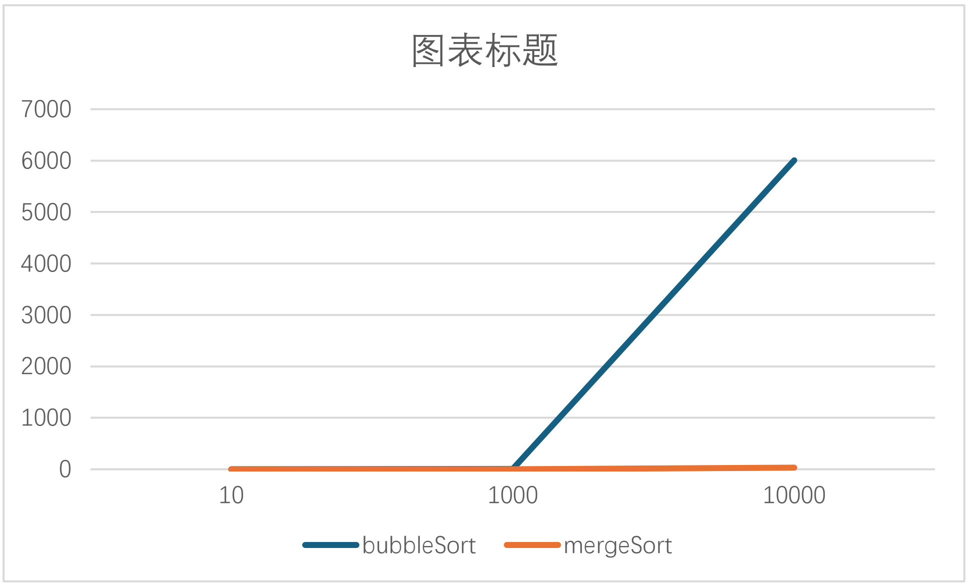
BubleSort and MergeSort complexity reflection



In this section, we discuss the time complexity of the BubbleSort and MergeSort algorithms, supported by experimental results from sorting files of varying sizes: sort10.txt, sort1000.txt, and sort10000.txt. We compare the observed behavior of each algorithm, analyze the results, and relate them to the theoretical time complexities of the respective sorting methods.

BubbleSort Complexity

BubbleSort is a simple comparison-based sorting algorithm with a worst-case time complexity of O(n2)O(n^2)O(n2), This quadratic growth occurs because BubbleSort repeatedly compares adjacent elements and swaps them if they are in the wrong order, requiring multiple passes through the data. As the input size increases, the number of comparisons grows significantly, which is evident in the experimental results for larger input files (sort1000.txt and sort10000.txt). For small input sizes, such as sort10.txt, BubbleSort performs relatively efficiently, but its performance rapidly degrades as the number of elements increases. The plot of runtime versus input size shows a sharp upward curve for BubbleSort, illustrating its inefficiency for large datasets.

MergeSort Complexity

In contrast, MergeSort is a more efficient sorting algorithm with a time complexity of O(nlog n)O(n \log n)O(nlogn) This complexity arises because MergeSort divides the input array into smaller subarrays, sorts them recursively, and then merges the sorted subarrays back together. The divide-and-conquer approach of MergeSort ensures that even for large input sizes, the growth in runtime is much slower compared to BubbleSort. The experimental data shows that MergeSort performs much better for larger inputs, maintaining a relatively linear increase in execution time as the input size increases. The plot for MergeSort clearly shows a more gradual and predictable growth in runtime, consistent with its O(nlog n)O(n \log n)O(nlog n) complexity.

Comparison of Performance

The comparison of sorting times for the three input files (sort10.txt, sort1000.txt, sort10000.txt) illustrates the stark difference in efficiency between BubbleSort and MergeSort. For smaller datasets, the difference in performance is less noticeable, as both algorithms execute fairly quickly. However, as the size of the input increases, MergeSort becomes significantly faster than BubbleSort. This is especially evident in the sort10000.txt file, where BubbleSort's time complexity dominates, and MergeSort remains efficient. These results highlight the importance of choosing the appropriate sorting algorithm based on the expected size of the dataset.