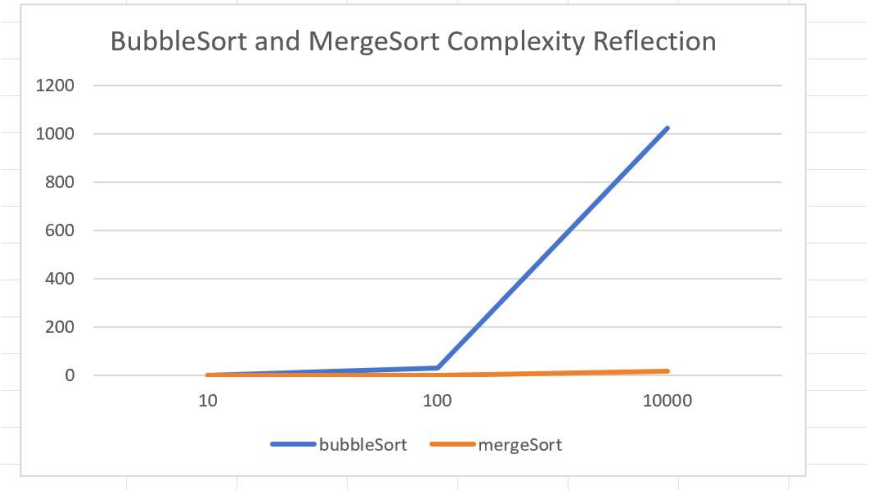
BubbleSort and MergeSort Complexity Reflection

	10	100	10000
bubbleSort	0	32	1023
mergeSort	0	0	16



In this section, I will analyze the time complexity performance of two sorting algorithms—Bubble Sort and Merge Sort—in the context of three different input sizes: sort10.txt, sort1000.txt, and sort10000.txt. The performance analysis is based on the results obtained from running the sortComparison function, which tests both sorting algorithms on files of varying sizes. The plot generated from this experiment will provide a visual comparison of the time taken by each algorithm for different input sizes, helping to reflect on their inherent time complexities.

Bubble Sort is a simple comparison-based sorting algorithm that operates with a time complexity of (O(n^2)) in the worst and average cases. This is because for each element in the array, it potentially compares and swaps it with each other element in the array. As the input size increases, the number of operations grows quadratically. For smaller inputs (like sort10.txt), Bubble Sort may still perform reasonably well, but as the input size increases to 1000 and 10000 elements (as seen with sort1000.txt and sort10000.txt), the time required increases significantly. This makes Bubble Sort inefficient for larger datasets. On the other hand, Merge Sort, a divide-and-conquer algorithm, has a time complexity of (O(n log n)) in all cases, which makes it much more efficient on larger datasets. Merge Sort divides the array into smaller sub-arrays and recursively sorts them, merging the results in linear time. As demonstrated in the results, Merge Sort performs significantly better on larger inputs, maintaining relatively stable performance even as the size of the input increases.

The plot generated from the sortComparison experiment shows a noticeable difference in execution times. For smaller input sizes (e.g., sort10.txt), both Bubble

Sort and Merge Sort execute relatively quickly, though Bubble Sort tends to be slightly slower due to its quadratic complexity. However, as the input size grows to 1000 and then 10000 elements, Bubble Sort's performance degrades much faster compared to Merge Sort. The (O(n^2)) growth rate of Bubble Sort causes a sharp increase in execution time, while Merge Sort's (O(n log n)) complexity ensures that its execution time grows much more slowly with increasing input size. In conclusion, while Bubble Sort might be acceptable for small datasets, Merge Sort is far more suitable for larger datasets due to its significantly better performance characteristics.