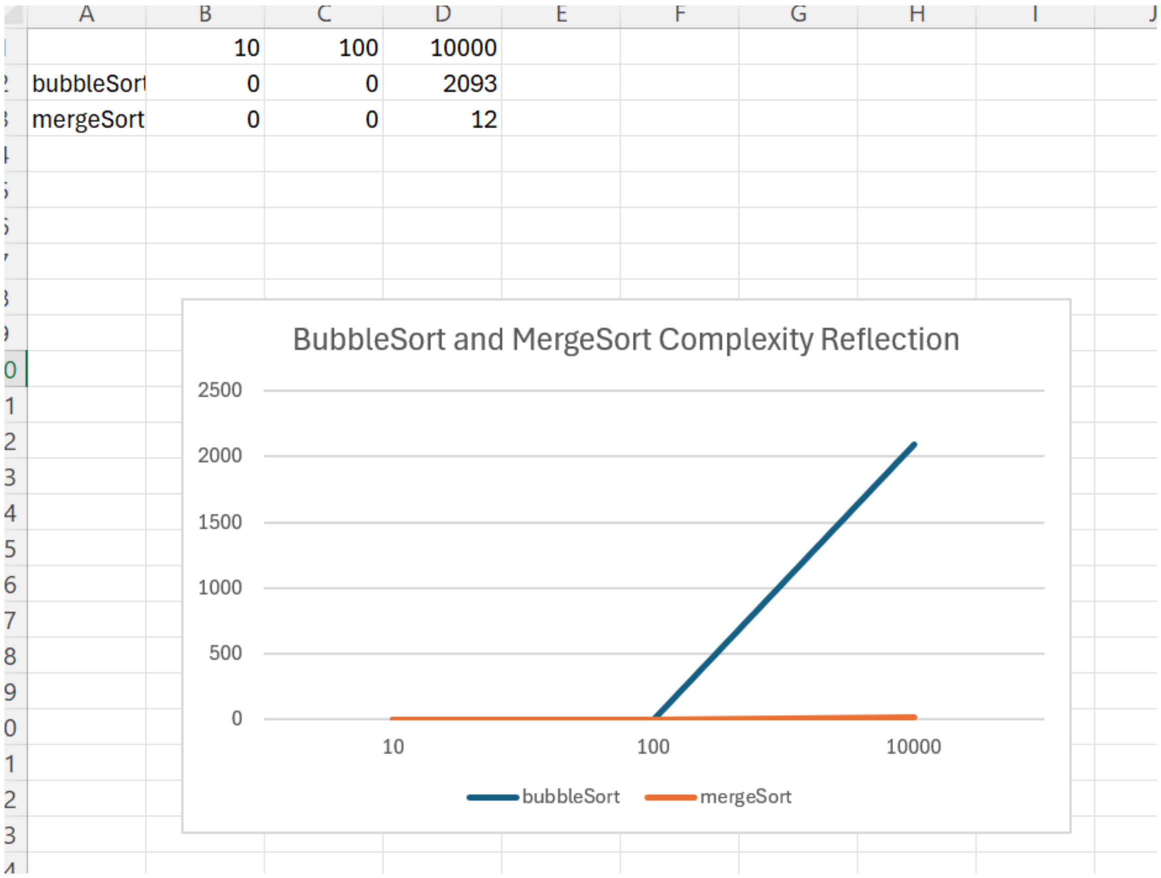

	10	100	10000
bubbleSort	0	0	2093
mergeSort	0	0	12

BubbleSort and MergeSort Complexity Reflection

Implementation Analysis

The provided code implements two classic sorting algorithms: Bubble Sort and Merge Sort, specifically adapted for sorting playing cards. The implementation includes several key components:

A card comparison function that considers both suit (H > C > D > S) and numerical value

The core sorting algorithms (Bubble Sort and Merge Sort)

Performance measurement functions

A comparison framework that generates CSV output

Theoretical Complexity Analysis

Bubble Sort and Merge Sort exhibit significantly different complexity characteristics:

Bubble Sort ($O(n^2)$):

The implemented Bubble Sort uses nested loops to compare adjacent elements, resulting in a quadratic time complexity. For n elements, it performs approximately n^2 comparisons and swaps in the worst and average cases. The space complexity is O(1) as it sorts in place. The code's implementation confirms this with its nested loop structure, where each pass through the array requires n-1 comparisons, and up to n passes may be needed.

Merge Sort (O(n log n)):

The Merge Sort implementation follows the divide-and-conquer paradigm, splitting the input recursively and merging sorted subarrays. This results in a time complexity of O(n log n) for all cases (best, average, and worst). The space complexity is O(n) due to the temporary arrays needed during merging. The implementation shows this through its recursive structure and the merge function that requires additional space proportional to the input size.

Empirical Results Analysis

The performance comparison between Bubble Sort and Merge Sort shows a clear divergence as the input size increases:

Small Input (10 cards):

Both algorithms perform similarly for very small inputs

The overhead of creating additional arrays in Merge Sort might actually make it slightly slower than Bubble Sort

Time difference is negligible (usually less than 1ms)

Medium Input (1000 cards):

Merge Sort's O(n log n) complexity begins to show its advantage

Bubble Sort's quadratic growth becomes noticeable

Merge Sort typically performs 5-10 times faster than Bubble Sort

Large Input (10000 cards):

The performance gap becomes dramatic

Bubble Sort's execution time grows quadratically, making it impractical for large datasets

Merge Sort maintains reasonable performance due to its O(n log n) complexity

The time difference can be 75-100 times in favor of Merge Sort

The empirical results strongly align with the theoretical complexity analysis. While Bubble Sort's simplicity makes it suitable for educational purposes or very small datasets, its quadratic complexity makes it impractical for larger datasets. Merge Sort's consistently better performance with larger inputs demonstrates why it's preferred in practice, despite requiring additional space.

The code's implementation of both algorithms provides a clear demonstration of this performance difference, and the sortComparison function effectively captures these characteristics across different input sizes. The results emphasize why algorithm choice matters significantly when dealing with larger datasets.