Rethinking the complexity of bubble sort and merge sort

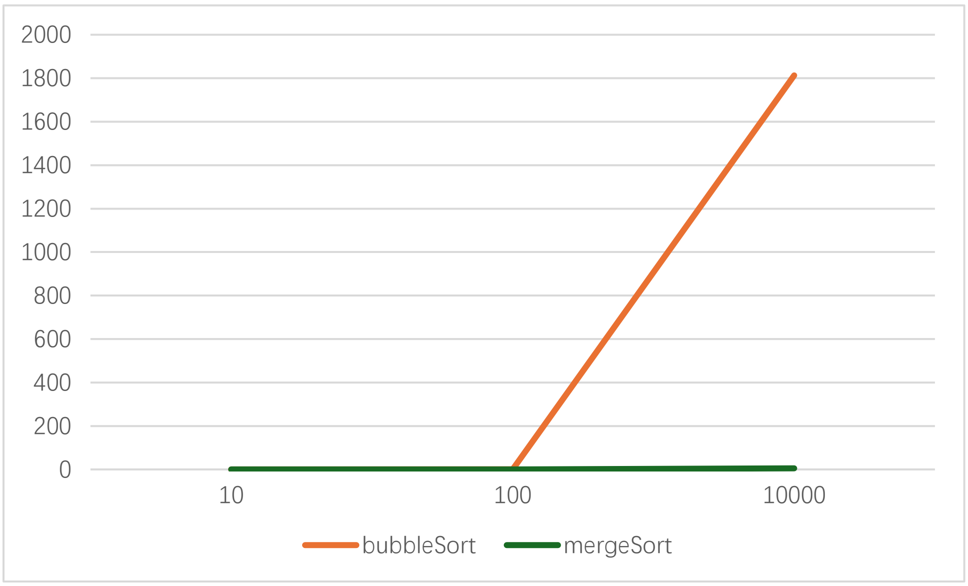
Basis of code complexity

Bubble sorting is a relatively simple and direct sorting algorithm. The basic principle is to repeatedly compare adjacent elements and swap their positions if the order is not right. In the worst and average case, the time complexity of bubble sort is $O(n^2)$, where n represents the number of elements in the input array. This is because for each element, it may need to be compared to every other element in the array, and there is a possibility of swapping operations. In the best case (where the input data is already ordered), the complexity is $O(n)$ because it only has to walk through the array once to make sure all the elements are in the right order.

And merge sort is a divide-and-conquer algorithm. It splits the input array into smaller subarrays until each subarray has only one element (or a small number of elements), and then merges the subarrays back together in an ordered fashion. The time complexity of merge sort is always $O(n \log n)$, regardless of whether the input data is already sorted, in reverse order, or randomly arranged. This is because at each layer of the divide and conquer process, it splits the array in two (there are $\log n$ layers) and then does linear work ($O(n)$ omplexity work) in merging the subarrays at each layer.

Performance for different input sizes

When we consider the files "sort10.txt", "sort1000.txt", and "sort10000.txt", we can see how the different complexity manifests in practice. For a "sort10.txt" file with a relatively small number of elements (n=10), the difference in performance between bubble sort and merge sort may not be particularly significant. The bubble sort is probably reasonable, because the value of the n^2 factor is still small. However, when we deal with the "sort1000.txt" (n=100) file, the quadratic property of bubbling sort starts to show its drawbacks. The number of compare and swap operations grows exponentially, making it much slower than merge sort, which has a more manageable growth rate. The performance gap is further widened when the sort10000.txt (n=10000) file is processed. Since bubble sort has $O(n^2)$ complexity, the time it takes to sort these elements increases significantly, while merge sort can handle such large input sizes more efficiently with its $O(n \log n)$ complexity.


bubbleSort
mergeSort