CS 3303-01 Data Structures

Instructor: Professor Vandelun Amokaha

Name: Anonymos

Assignment Unit 6

**Overview of the Assignment**

In this assignment, we are required to sort an array containing 21 random integers using the Merge Sort algorithm. During the sorting process, we must track the number of exchanges made and display the result. Additionally, we will perform an asymptotic analysis of the algorithm and compare it with the Insertion Sort.

**Explanation of the Algorithm**

Explanation of the Merge Sort Algorithm

Merge Sort is a sorting algorithm based on the "Divide and Conquer" method, where the list is divided into smaller parts, recursively sorted, and then merged to form a sorted list. The algorithm works through the following steps:

1. Divide the list in half recursively until each part has one element.

2. Merge the divided parts while maintaining order.

3. Finally, a fully sorted list is obtained.

Comparison with Insertion Sort

Insertion Sort is a "brute force" algorithm that sequentially traverses the list, inserting each element into the correct position. Although Insertion Sort is relatively simple to implement and effective for small datasets, the number of exchanges becomes large, making it inefficient for large datasets.

On the other hand, Merge Sort, which uses the Divide and Conquer method, results in fewer exchanges compared to Insertion Sort and is particularly more efficient for large datasets.

**Implementation Details**

In the implementation of Merge Sort, the provided list `12, 9, 4, 99, 120, 1, 3, 10, 23, 45, 75, 69, 31, 88, 101, 14, 29, 91, 2, 0, 77` was used and sorted. During the sorting process, the number of exchanges was tracked, and this final number is compared with the 114 exchanges recorded for Insertion Sort.

Below is the implemented Merge Sort code and the sorting result:





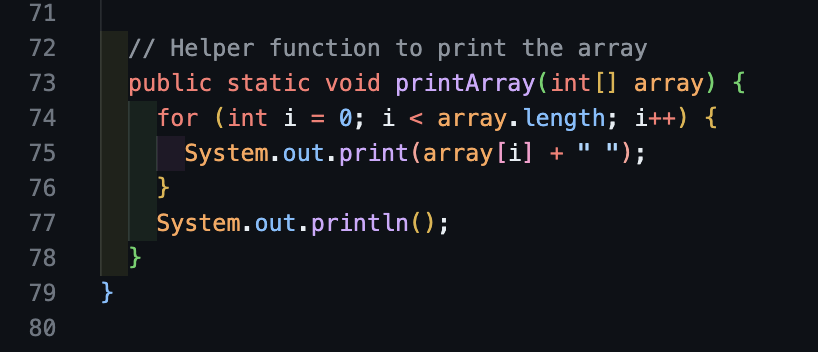


Figure 1: MergeSort.java

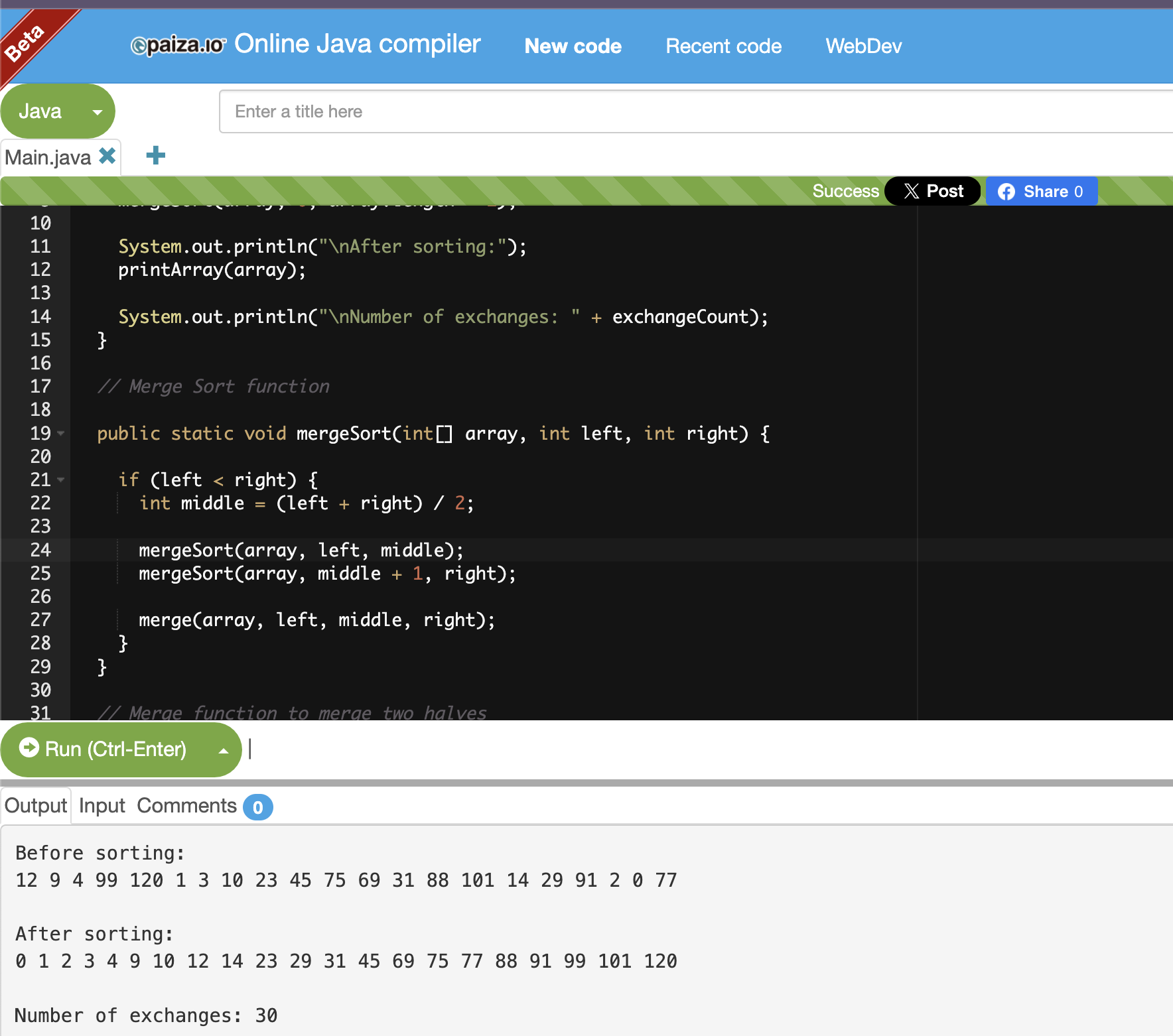


Figure 2: Result from online Java compiler execution

**Asymptotic Analysis**

The time complexity of Merge Sort, due to its Divide and Conquer nature, is in the worst, best, and average cases (Shaffer, 2011). This complexity arises from the time taken to recursively divide the list and the time taken to merge it.

In contrast, the time complexity of Insertion Sort is in the worst case and) in the best case, which is effective for nearly sorted data but performs poorly on random datasets (Shaffer, 2011).

**Analysis of the Results**

The result of executing Merge Sort showed that the number of exchanges was 30, which is significantly lower compared to the 114 exchanges in Insertion Sort. This indicates that Merge Sort is more efficient than Insertion Sort, especially in large datasets, as it reduces the number of exchanges required.

As expected, the performance of Merge Sort demonstrated its efficiency. As confirmed in the asymptotic analysis, Insertion Sort has a complexity of , which requires many exchanges for random datasets. In contrast, Merge Sort has a complexity of , allowing it to complete the sorting process with fewer exchanges.

**Conclusion**

In this assignment, we implemented Merge Sort and compared its efficiency with Insertion Sort. Merge Sort proved to be more efficient than Insertion Sort, particularly for large datasets, where its performance is significantly better. By tracking the number of exchanges, we confirmed that Merge Sort required fewer exchanges than Insertion Sort. In the future, comparing other sorting algorithms will be important to ensure the most suitable algorithm is selected based on the use case.

Word Count: 560

References

1. Shaffer, C. (2011). *A Practical Introduction to Data Structures and Algorithm Analysis.* Blacksburg: Virginia. Tech.
2. paiza. (n.d.). *Online Java compiler and coding environment.* paiza.io. Retrieved October 1, 2024, from https://paiza.io/