Project Title: Merge Sort Algorithm Implementation

Overview
This project demonstrates the implementation of the merge sort algorithm in Java. It reads a sequence of strings from standard input, sorts them in ascending order, and then outputs the sorted list. The implementation follows a top-down, recursive approach, making it a classic example of a divide-and-conquer algorithm.

Algorithm Description in Plain English

1. **Divide the Problem:**
   The algorithm begins by splitting the input list into two halves. This process is applied recursively until each sublist contains only one element. Since a single element is inherently sorted, these sublists serve as the base case for the recursion.

2. **Conquer by Recursion:**
   Once the list is divided into the smallest possible sublists, the algorithm starts the merging process. It takes two sorted sublists and combines them into one larger sorted list. This step involves comparing the smallest elements from each sublist and selecting the smaller one to add to the merged list.

3. **Merge the Sublists:**
   During the merge step, an auxiliary array is used to help combine the two sorted sublists. The algorithm iterates over the elements in both sublists, comparing them, and placing them in order into the auxiliary array. After the merging is complete, the auxiliary array replaces the corresponding section of the original array.

4. **Recursive Combination:**
   The merge operation is applied recursively, merging increasingly larger sorted sublists until the entire array is recombined into one sorted sequence. This method ensures that the final output is completely sorted in ascending order.

Key Characteristics

- **Efficiency:**
  The algorithm has a time complexity of O(n log n), making it highly efficient for sorting large datasets.

- **Stability:**
  Merge sort is stable, meaning that the relative order of equal elements remains unchanged after sorting.

- **Space Usage:**
  The algorithm uses extra memory proportional to the size of the input array to facilitate the merging process. This additional space is used to temporarily store elements during the merge phase.

- **Divide and Conquer:**
  By breaking down the sorting problem into smaller, manageable pieces and then combining the results, merge sort exemplifies the divide-and-conquer strategy.