Project 1 Report

Samuel Chandler

CE 4348

4/3/2024

Files and Purpose:

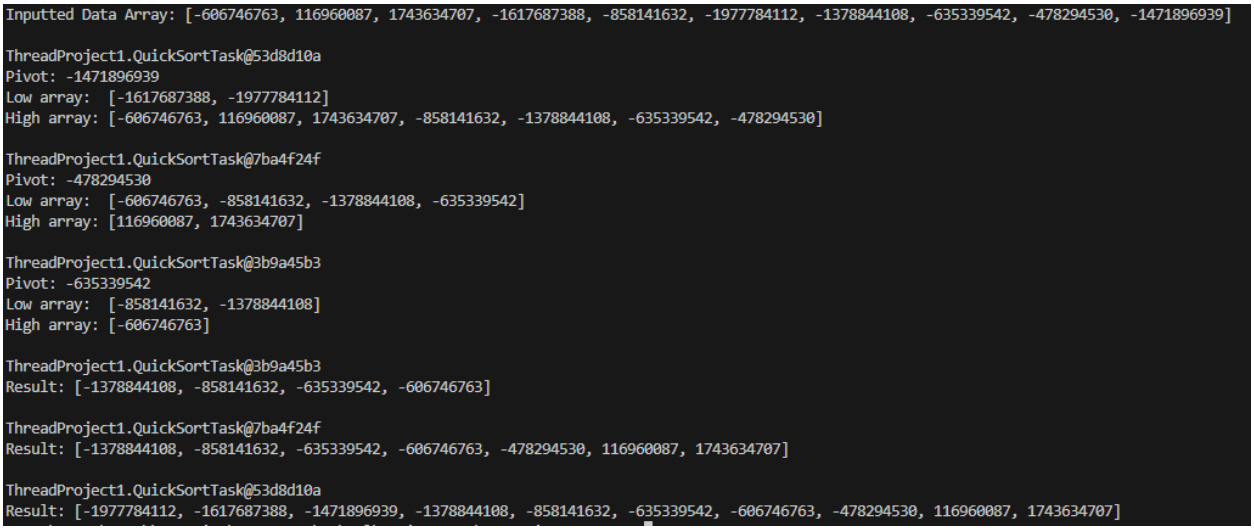
Main: used to read the text file into an array and call the other two classes to start the sorting process for each, as well as write the results for their respective text files.

Quick Sort Task: Extending the Recursive action class, this file contains the quick sort task class, which on compute will sort the array using the quick sort method, splitting the array into a lower and higher array based on the pivot point (last element in the array) and forking with the higher and lower arrays to repeat this process. When the array has only two elements, the thread will check if they need to be swapped, perform the necessary action, and terminate. This will also happen if one or fewer elements in the array are passed to it. From here, all arrays are appended together using an array list and stored to be printed using the Write to File method.

Merge Sort Task: This file contains the merge sort task class, which extends the recursive action class. On compute, the array is sorted using the merge sort method. The function compute forks the left and right portions of the array until there is one or less, at which point it returns and merges those portions of the array using the merge method.

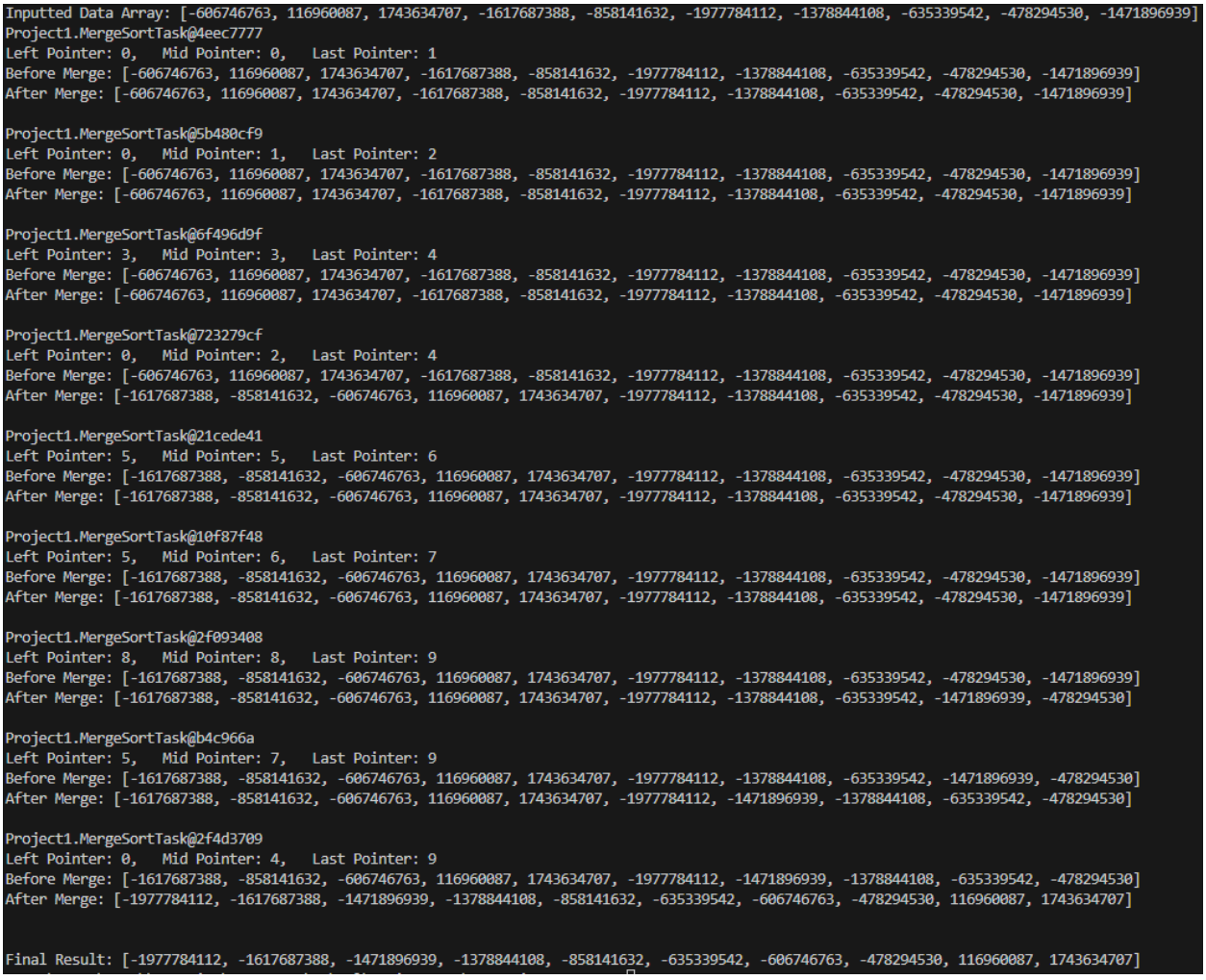
Quick Sort Results:

For the sake of readability, an array of 10 was created instead of 20,000;



The results shown above are the steps taken to split the array at the pivot and recursively on different threads perform the same action until the threshold at 2, where the Array is merged back together to form a fully sorted array, as we see in the result.

Merge Sort Results:



The image above shows the different threads and the values of the left, right, and middle pointers. And the state of the array before and after each merge.

As we can see, the results are the same for Quick sort and merge sort implementations.