CSC301 Programming 1: Sorting Stuff

October 11, 2023

Havin Lim

Raw Data

Array Size	Insertion Sort	Quick Sort	Merge Sort
10	2	2	2
100	2	2	2
150	2	2	2
175	2	2	2
250	3	2	2
500	7	2	3
1000	8	2	2
10000	205	21	7

Explanations for each row

Array Size = The total number of elements in the array (before and after sorting)

Insertion Sort = The milliseconds it took for insertion sort to sort the following array size (calculated with System.currentTimeMillis)

Quick Sort = The milliseconds it took for quick sort to sort the following array size (calculated with System.currentTimeMillis)

Merge Sort = The milliseconds it took for merge sort to sort the following array size (calculated with System.currentTimeMillis)

Explanations of the data

Each array size, 10 through 10,000, was run 10 times on each sorting algorithm and the milliseconds on the data was the mean of the 10 results.

In my opinion, the base case for quick sort and merge sort to utilize insertion sort would be when the runtime of insertion sort is faster compared to the other two sorts. By trying different sizes of arrays, 10 to 10,000 elements, the results have shown that when there are less than 175 elements in an array, insertion sort runs faster than quick or merge sort. The number of elements may vary from computer to computer, and I thought it was reasonable to choose 175 elements since that was the data result from my experiment.