CS 6301.011

Savan Amitbhai Visalpara (sxv180069) and Srikumar Ramaswamy (sxr170016)

Experiments:

This experiment aims to test four different implementation of merge sort as explained in the class. The program was ran on different and large values of n, the length an array. As required, we ran this script for 50 times to take the average time.

Take 2: avoid creating array B in merge()
Take 3: avoid unnecessary copying between A and B in merge()
Take 4: use insertionSort() if n is below certain threshold
Take 6: use iteration and insertionSort() together for optimal performance

Below table outlines our results:

Take	Length of an Array			
	16000000	32000000	64000000	128000000
2	Time: 2567 msec. Memory: 125 MB / 256 MB.	Time: 5598 msec. Memory: 247 MB / 414 MB.	Time: 11378 msec. Memory: 491 MB / 820 MB.	Time: 52776 msec. Memory: 979 MB / 1634 MB.
3	Time: 2470 msec. Memory: 125 MB / 256 MB.	Time: 5034 msec. Memory: 247 MB / 256 MB.	Time: 10616 msec. Memory: 491 MB / 820 MB.	Time: 92995 msec. Memory: 979 MB / 1634 MB.
4	Time: 2386 msec. Memory: 125 MB / 256 MB.	Time: 5142 msec. Memory: 247 MB / 414 MB.	Time: 10639 msec. Memory: 491 MB / 820 MB.	Time: 22144 msec. Memory: 979 MB / 1634 MB.
6	Time: 2210 msec. Memory: 125 MB / 256 MB.	Time: 4273 msec. Memory: 247 MB / 415 MB.	Time: 9043 msec. Memory: 491 MB / 822 MB.	Time: 19866 msec. Memory: 979 MB / 1635 MB.

Conclusion:

Based on the data above, we reached to the conclusion that the take 6 (with iteration and insertion sort) works the best, as expected.