Assignment 6

Jed Malashock

December 2019

1 Report

Well, this assignment was supposed to take 2 hours, but it ended up taking closer to 6. Truth be told I could have done it in 2 hours, but some of the sort functions behaviors seemed off to me so I played around with the them for a while. To start out with, bubble sort really sucks. On my MacBook Pro, I gave Docker 12GB of RAM and 2 out of my 4 cores, and with those resources it took bubble sort just under 18 minutes to sort 500,000 numbers, selection sort exactly 5 minutes, insert sort just over 3, and merge sort didn't work- I got a segmentation fault. Once I reduced it down to 100,000 numbers, I didn't understand how, but merge sort worked almost instantly- within one second. I didn't believe it, so I checked my code to make sure it wasn't sorting an array that's already been sorted by another algorithm, and then I ran it again a few more times. I still wasn't satisfied so I went to my desktop for more testing, where I gave docker 24GB of RAM and 4 out of my 8 cores. With this I got similar results- for 330,000 numbers bubble took 9 minutes, selection took 2 minutes, insert took 2 minutes, and merge sort took, once again, less than a second. After some testing, I found that merge sort starts getting a segmentation fault somewhere between 330,000 and 350,000. At this point I wondered if quick sort was similar to merge sort in run times, so I Frankenstein'ed that algorithm together using google and the textbook (like I did with merge sort). Quick sort ended up always running less than a second too, but this time clunked out somewhere between 500,000 and 750,000 values. These results were more drastic than I expected. I didn't realize how big of a difference x*logx was on a graph from x until I graphed them out and set x to 500,000 (representing the two big-oh run times)- the former takes about 88,000 (!!!) times longer than the latter. Trade off wise I am inclined to believe the O(n) equations in this case are better for memory since they could run with more values, but for speed and any realistic practicality, the two O(nlogn) equations are light years ahead. The biggest shortcoming of this empirical analysis was that I used a sample size of 2 (since I only screenshotted 2 of the outputs and my tests were too similar to make me want to go back and run more tests).