Joseph Sneifer

12/19/20

CPSC 350 – 03

Assignment 6 Reflection

The time taken for all the sorting algorithms surprised me in some cases and did not surprise me in others. What did not surprise me was that the time take for a small set of data was essentially the same for all the sorting algorithms. To test my program, I used a data set of twenty to check that the data was being correctly sorted and the times were tracking correctly. Since I formatted the doubles to five decimal places, the algorithms almost always finished running in 0.00001 seconds. Occasionally merge sort would take 0.00002 seconds, which I believe is due to its complexity and use of recursion. The lack of difference in runtimes for small sets of data did not surprise me much since we have discussed this in class.

What did surprise me, however, was the range of completion times when sorting large sets of data. When I randomly generated around 15,000 doubles to be sorted, some of the algorithms took much longer than the initial tests with the smaller set of data. All the algorithms increased their runtimes fairly significantly, and some more than others. With 10,000 doubles to sort, the longest runtime was the bubble sort at almost one second: 0.94919 seconds. The shortest runtime was quick sort at 0.00283 seconds.

There are tradeoffs when using these algorithms for larger sets of data. One algorithm may be faster; however, it may use up more storage on your computer. For example, I was unable to run quick sort with a set of data larger than around 17,000 even though it most likely would have been the fastest at sorting the set of doubles. We can look at the runtimes in the program and compare them to one another, but it does not tell us easily how much the time will increase depending on how much we increase the data set size. That is why mathematical analysis is more useful for comparing algorithms.