CSCI 430: Program2: MaxSubArray

John E. Buckley

October 11, 2017

Implementation:
Implementing selection-sort and bubble-sort were similar to the previous assignment, as in, I could recycle code. However each program was different enough. Bubble-sort simply compares adjacent items and exchanges those that are out of order, this takes many passes through the list. Selection-sort improves on bubble-sort by making only one exchange for every pass through the list, which still takes many passes, but is slightly faster from my experience.

Testing Expectations:
I expected selection-sort to be worse off than bubble-sort since selection-sort only makes 1 change for every iteration through the list. I also expected that insertion-sort, bubble-sort, and selection-sort to have similar looking graphs since they are all $O(n^2)$ for average-case run times. I expected merge-sort to still be the best because out of the four, it has the best running-times.

Testing Observations and Analysis
Selection sort for reversed order data ended up being linear which I found odd as selection-sort has a best-case time of $O(n^2)$, this can be due to an error in my implementation of it. Bubble-sort went exactly as I expected because bubble-sorts best-case is $O(n)$, which is if the data is already ordered, and this shows on the graph that it ran twice as fast for when the data was already ordered. Lastly, Merge-sort still shows to be the most effective overall.

Take-Away:
There are many different algorithms and variations of algorithms for sorting data, and it pays in the long-run to sit down and spend the time calculating run-times to see which will be best to suit your needs. Don't pick the algorithm that seems the easiest to implement, as it usually is the one that takes longer to run, at least for the four algorithms we used already.