Nicholas LaBruna

Rene German

CPSC 350

18 May 2018

Sorting Assignment

It was quite shocking to see BubbleSort take more than thirty times the amount of time it took QuickSort to sort a list of 50,000 numbers. Both SelectionSort and CocktailSort ran much faster than BubbleSort, but QuickSort still took a fraction of the time of either of those. This was interesting because using such a large list of numbers really showed how big the discrepancy between log(n) and n^2 runtime is as numbers continue increasing. The tradeoff for using something like QuickSort instead of one of the easier options is that it is much harder to code and understand. Also, it's runtime can approach n^2 if the list it is sorting has already been sorted, making InsertionSort the more viable option in that case because it can run at a near O(n) runtime if data is given already partially sorted. MergeSort will consistently be at a log(n) runtime, but the tradeoff for it's speed is it's inefficient ram usage. C++ affected the results in that it is generally faster than other interpreted languages because it makes machine code specifically made for the computer architecture of one's computer. This forced us to use massive amounts of numbers in order to be able to tell how long each algorithm ran. Empirical analysis is not always accurate because so many different variables go into creating and running a program, and for sorting it is even more variable with the organization of data given playing a huge part in how the program ends up running.