Andriana Agrusa

CPSC 350-02

Sorting Report

This program ran 5 sorting algorithms--Quick sort, Merge sort, Selection sort, Insertion sort, and Bubble sort--on 100 randomly-generated double numbers from a text file. The difference in runtimes were not as drastic as I was expecting, however, if this program were to be run on hundreds of thousands of numbers, I think there would be a more noticeable difference.

When choosing which sorting algorithm to use we must be mindful of certain circumstances like disk space and the number of items to be sorted. Quick sort is best when we need to sort a large number of data in the quickest time possible. The results of this program show that Quick sort indeed yielded the shortest duration time.

Insertion sort is most efficient when the data is already partially sorted. The numbers in this data set were randomly shuffled which is why Insertion sort had a longer duration time than Quick Sort or Merge sort. Bubble sort is a slower algorithm in general which we can conclude from the results as well, and it would've run faster if we used a very small data set.

I think that using a compiled language like C++ allowed these algorithms to run a bit faster than an interpreted language. If we tested these sorting methods in Python, for instance, the results would yield similar patterns but overall have longer duration times.

This empirical analysis is restricted to the number of data tested and the algorithms written in C++. Had this program been written in another language, or tested with hundreds of different combinations of larger data sets, the results might have appeared differently.