Assignment 6 - Sorting

Zachary Jagoda

December 2017

1 Overview

Within this final project, students worked to implement Quick Sort, Insertion Sort, and a sorting algorithm of their choice. In this instance, I chose to include Gnome Sort as my third sorting method. This LaTeX Report will discuss the results of running the algorithms on the same data values and how each varies in time complexities.

2 Results

All three sorting algorithms performed very differently than each other, but the results were expected. The times that each method output were dependent on the size of the data provided to the program; the smaller the data the faster each would essentially perform. For this assignment, a .txt file containing 40,000 values was sorted to demonstrate the actual difference between Quick Sort, Insert Sort and Gnome Sort.

If one wants to sort a large amount of data in the quickest amount of time, Quick Sort is the best option. With the best results clocking in at 15.625ms, Quick Sort seems to be the better method to use. It's ability to 'quickly' sort data is due to its Divide and Conquer algorithm, where the data is partitioned based on pivot points.

If one is dealing with data that is partially presorted, Insert (Insertion) Sort is a better option than Quick Sort. However in the case of the testing data that was not presorted, the algorithm ran for 1203.12ms. This is due to the fact that this method takes into account all of the data before moving a value, causing the performance to decrease.

If one does not care about the time in which it takes to sort through massive amounts of data, than Gnome Sort is the better option. Running at 3109.38ms, the Gnome Sort algorithm works by comparing two numbers at a time, and if they aren't in order it backtracks until the values are correct. This is an example of an inefficient sorting method, and can be inferred by it's adopted name 'Stupid Sort'