Joe Parisia
CSC382

**Sorting Algorithms**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | 10 | 100 | 1000 | 10000 |
| Bubble | 0 | 0 | 0.009 | 1.048 |
| Selection | 0 | 0 | 0.004 | 0.144 |
| Insertion | 0 | 0 | 0.001 | 0.068 |

Introduction:

Sorting algorithms are very straight-forward in how they perform. It takes the data / elements from either a linked list or an array and, you guessed it, sorts them. The biggest difference between my selected algorithms is how it goes about sorting them.

Bubble Sort – swaps the adjacent element if it is in the wrong order.

Selection Sort – repeatedly finds the minimum element and puts it in the beginning

Insertion Sort – one at a time transfers the elements into the correct position

Programmer’s Guide:

For my program I implemented a simple menu system to choose between the algorithms you wish to use. Once selected it runs the desired algorithm 4 times. After the data set is run with sizes of 10, 100, 1000, 10000 it records how long each one took to run using the <ctime> library.

Both bubble sort and selection sort use a swap function that exchanges two values depending on their particular trigger parameters.

Analysis:

The worst case scenario of bubble and selection sort algorithms is On^2 where insertion sort is n^2.

Each of these datasets can be use without any real issues with datasets under 1000 elements. Once you reach 10000 you start getting into measurable differences between them.

For this test Insertion Sorting was the overall winner out-performing the other two during every test.

Under which dataset sizes does each algorithm perform best? Why or why not? Does any algorithm perform best under all cases?