Report on Sorting Programming Project

I used the numbers 20,000, 10,000, and 2,000 as test values for number of numbers to be generated.

The functional difference in the sorting was, of course, that the entire rest of the array is filled with INT\_MAX and therefore already pre-sorted, so every element past UserNumOfNumbers is INT\_MAX.

This provides an interesting piece of information, namely, how relatively efficient is the algorithm with partially sorted data sets?

The data is tabulated in the accompanying txt file, sorted first by algorithm, then by test number, then by piece of information (swaps, loops, time taken). The data is not plotted on a line graph because I didn’t have enough time, however the length of the numbers of loops and swaps (in digits) serves as a *very* rough bar graph.

The Theoretical Big-O performance of the algorithms are as follows:

Bubble Sort: Average of approx O(N^2)

Selection Sort: O(N^2)

Shell Sort: O(N log N)^2

Quick sort: O(N log N)

There are also worst and best cases, however I will forego speaking about those for brevity.

Methods which performed a lot of loops were Selection and Bubble sorts, and Bubble sort was unmatched in swap counts.

*Bubble sort is really bad on large, random arrays, although it is reasonably efficient on partially sorted arrays, such as with small UserNumOfNumbers. By reasonable, I mean within a couple orders of magnitude of the shell and quick sorts.*

**Extraneous features~**

I have added multithreading capabilities to the program to allow a small animation to play while the sorts are occurring. To enable multithreading, uncomment the line that says #define USE\_THREADING and compile with -lpthread.

I could have used multithreading to do multiple sorts at once, which might be utilitarian, however it is also harder to do and not as much fun as rendering the mandelbrot set.

The program is capable of rendering a spinning mandelbrot set (iterations = 128) while you wait for your data to be sorted.

This is accomplished through the miracles of floating point arithmetic.