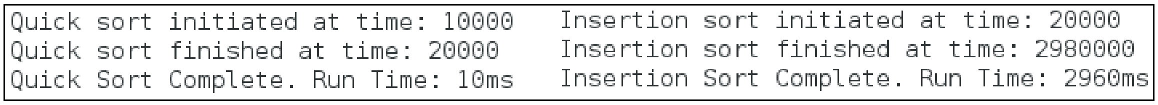
Data Structures Final Project Analysis

Tyler Andrews
December 16, 2016

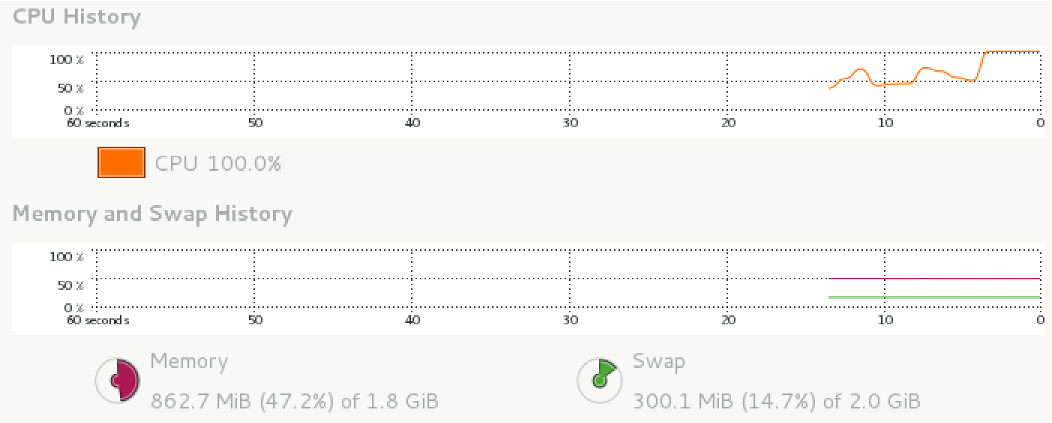
All in all, this assignment was very eye opening. I was extremely surprised by the differences in performance between the sorting algorithms we tried, and specifically how much more efficient quicksort was than the others. On average, it was a few hundred times faster for large inputs. It managed to sort 50,000 numbers in under 10ms, which was much faster than I would have ever guessed. Here you can see my program's output for an input file containing 50,000 randomized doubles:



```
Quick sort initiated at time: 10000    Insertion sort initiated at time: 20000
Quick sort finished at time: 20000     Insertion sort finished at time: 2980000
Quick Sort Complete. Run Time: 10ms    Insertion Sort Complete. Run Time: 2960ms
```

In this case, quicksort was nearly 300 times faster than insertion sort, and (not shown) 600 times faster than gnome sort. This is a good representation of logarithmic vs. quadratic performance. On paper, O(n^2) may not look much larger than O(nlogn), but this exercise certainly shows how far apart they really are in terms of speed and efficiency.

As far as performance, the brute force algorithms had a very clear hindrance on my VM. Both insertion sort and gnome sort spiked CPU usage for my VM to 100%. Here you can see the absurd spike in resources when insertion sort is initated. The program also appears to use about 800MB of RAM, which seems very high.



🟧 CPU 100.0%

Memory
862.7 MiB (47.2%) of 1.8 GiB

Swap
300.1 MiB (14.7%) of 2.0 GiB

I think this analysis would have benefited from a visual component other than seeing the run times themselves. Seeing the data being sorted would be a great way to show the speed of the algorithms (like the YouTube videos we watched in class) in a more direct way. Overall, I was extremely impressed by the efficiency of quicksort, and I think this assignment was a very good indicator of the value of recursion.