CSCI 430: Program 3

John E. Buckley III

December 3, 2017

Implementation:

Counting-Sort is different in implementation compared to the previous algorithms observed earlier this semester. This is because Counting sort uses an array for temporary working storage. This is a good thing and a bad thing. This is good because this makes the running time of Counting-Sort $\Theta(n)$. Counting-Sort assumes that each input element is within the range of 0 to k, and as long as k is $O(n)$ then the running time will be $\Theta(n)$. This is bad because Counting-Sort is only efficient if the range of input data is not significantly greater than the number of objects to be sorted. This is also bad because Counting-Sort takes up a lot of storage.

Testing Expectations:

I expect Counting-Sort to be the most effective algorithm to be tested yet because the running time is $\Theta(n)$.I also expect the graphs to look different then ones previously created, I expect 3 horizontal lines to overlap each other.

Testing Observations and Analysis:

As expected, Counting-Sort out preformed all previous sorting algorithms. The graph came out very linear, give or take a hundredth of a second, however not all 3 lines overlapped each other. Reversed ordered data, and ordered data overlapped each other however random data happened to take just slightly longer, but nonetheless still linear.

Take-Away:

A perfect world will never exist. When you finally come across a sorting algorithm that runs in $\Theta(n)$ time, and you think it can't get any better, you finally realize that not all is right in the world because that algorithm takes up a massive amount of storage compared to other algorithms. The choice at this point is, do I trade run time for less storage or do I trade more storage for faster run time.