CSCI 430: Program 3

John E. Buckley III

November 10, 2017

Implementation:

Heapsort is similar to selectionsort, as in they both divide the input into a sorted and unsorted region and shrinks the unsorted region by taking the largest element and puting that into the sorted region, however, heapsort is an improvement because it uses the heap data structure to do this rather than a linear search to find the maximum. Quicksort is similar to mergesort, as they are both divide and conquer, however quicksort picks an element called the pivot and puts everything less than the pivot on one side and all the larger elements on the other, which is called partitioning, and then recursively does this until the array is sorted. Randomized quicksort does this same thing as well, however, it takes away the worst case scenario of quicksort by assuring all data to be sorted is random.

Testing Expectations:

I expect heapsort and randomized quicksort to have similar looking graphs as they both have worst case and average case times of $nlogn$. Quicksort may have an average case time of $nlogn$ as well, however it has a worst case time of n^2, so I expect it to have the steepest curve in its graph.

Testing Observations and Analysis:

I'm shocked to see that heapsort did so much better than the other two, especially with randomized quicksort as they had the same running times. However it may only vary in fractions of seconds, I assume that randomized took slightly longer because it may be making more calls from start to finish as its using partitions rather than a binary heap. Quicksort nonetheless performed the worst which was to be expected as it had the worst running time of the three.

Take-Away:

Although it may have "quick" in the name doesn't mean that it will be the quickest. Divide and Conquer still shows to be the more efficient way to go about with sorting as seen between quicksort and randomized quicksort, and lastly, don't be afraid to think a bit outside the box and instead of making a new algorithm to sort data, use what you already have, for example, heapsort.