Assignment 1 – Megan Bender

Bubble Sort

1,000,000 Random: 48 mins and 52.450 secs

1,000,000 Sorted: 24 mins and 48.972

Algorithm Description: This algorithm is probably the most intuitive, since we are just moving the value, from one index, to the next until we put the largest value at the top and repeating this until sorted. So again, intuitively it's very simplistic, but with that is lacking the factor of optimization, especially with large sets of data. In turn causing unnecessarily long runtimes, as compared to any other sorting algorithm we have discussed.

Algorithm Runtime Classification:

Best Case = $O(n^2)$

Worst Case = $O(n^2)$

Average Case = $O(n^2)$

Selection Sort

1,000,000 Random: 21 mins and 2.669 secs

1,000,000 Sorted: 21 mins and 12.365 secs

Algorithm Description: This sorting algorithm is running quicker than the bubble sort due to the implementation of the swap code. In bubble sort, we constantly “bubble up” the value all the way through the array, take for instance the array 9, 4, 5, 3, 1, we would need to swap 9 four times. However, in this sorting algorithm, selection sort, we are just saving the index of the best value and making the swap at the end after many comparisons, thus in my example the swap would happen once for the first loop through. The reason this still takes longer to run is due to the nested for loops, causing us to have a runtime of n^2.

Algorithm Runtime Classification:

Best Case = $O(n^2)$

Worst Case = $O(n^2)$

Average Case = $O(n^2)$

Insertion Sort

1,000,000 Random: 14 mins and 49.608 secs

1,000,000 Sorted: 39.129 secs

Algorithm Description: This algorithm is much faster than the two previous algorithms since there are no nested for loops. Since we do not have the nested for loops, in the best case, we go from an n^2 to an n runtime, which is significantly faster. However, in random list this algorithm still runs at an n^2 pace.

Algorithm Runtime Classification:

Best Case = O(n)

Worst Case = $O(n^2)$

Average Case = $O(n^2)$

Shell Sort

1,000,000 Random: 1.352 secs

1,000,000 Sorted: 0.151 secs

Algorithm Description: This algorithm was able to run much faster than that of all the other sorting algorithms. When specifically looking at bubble and selection, shell sort runs much faster since it does not have nested for loops, thus making the runtime faster. As for insertion sort, shell sort takes a very similar concept and breaks it up, so instead of just running through each value and comparing them to each other, we can quickly jump values closer to their sorted positions. Thus, reducing the amount of time, we need to make comparisons.

Algorithm Runtime Classification:

Best Case = O(n)

Worst Case = O(nlogn)

Average Case = O(nlogn)