Christopher Shenton

27 October 2023

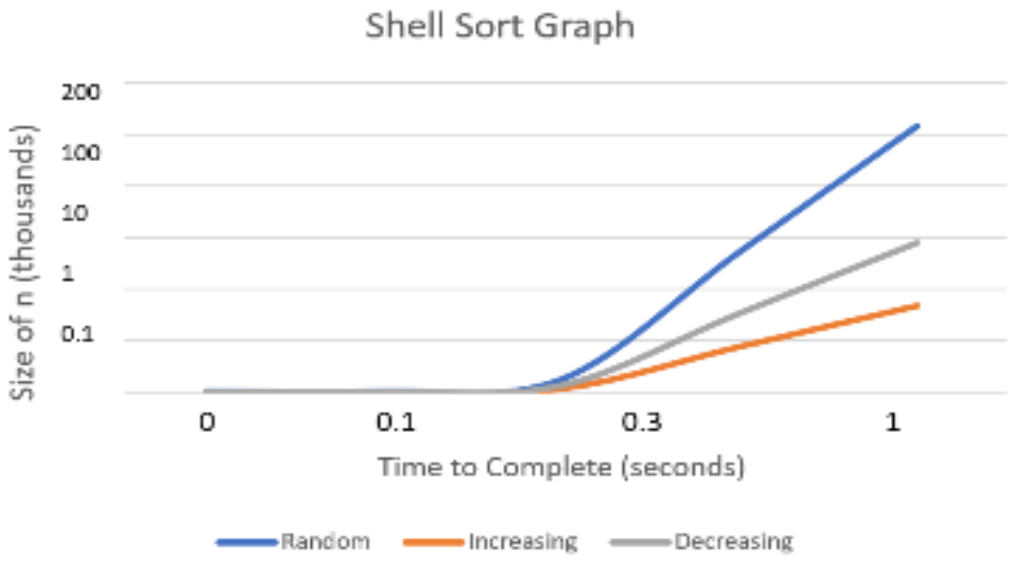
CSC 2400

Assignment 2

Below are four tables, one for each assigned sorting algorithm, containing the size of the arrays sorted, as well as the time to complete and factor increase in time for random, increasing, and decreasing arrays. Along with each table is a respective graph showing the increase in completion time for each type of array being sorted. The final page contains my analysis of each sorting function.

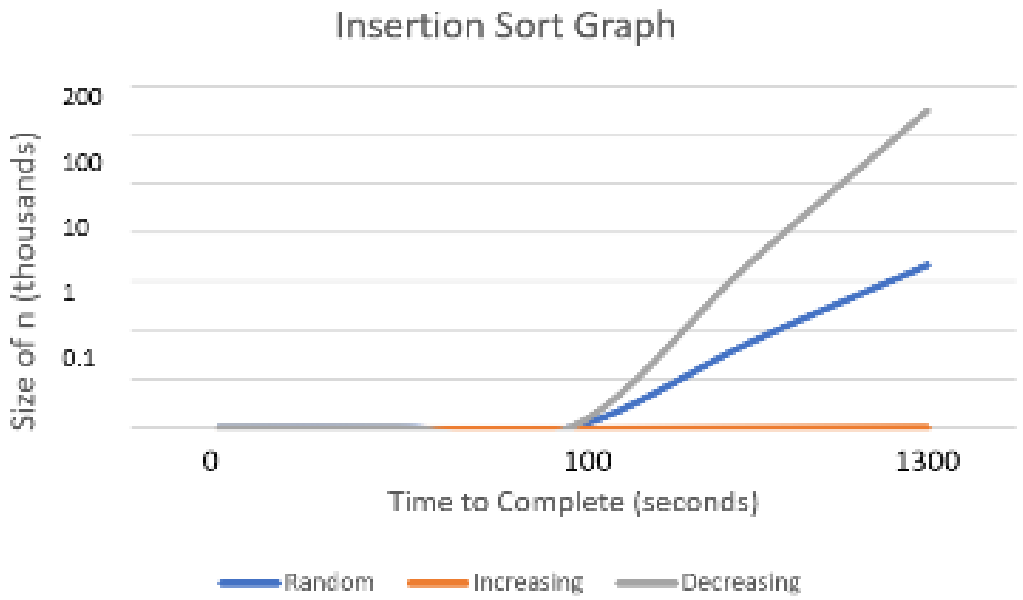
Shell Sort

Size of n	Time to Complete (Random)	Time to Complete (Increasing)	Time to Complete (Decreasing)	Factor Increase in Time (Random)	Factor Increase in Time (Increasing)	Factor Increase in Time (Decreasing)
100	0.000s	0.000s	0.000s	N/A	N/A	N/A
1000	0.002s	0.001s	0.002s	N/A	N/A	N/A
10000	0.037s	0.014s	0.025s	18.5	14	12.5
100000	0.542s	0.176s	0.304s	14.6	12.6	12.2
200000	1.031s	0.334s	0.578s	1.9	1.8	1.9



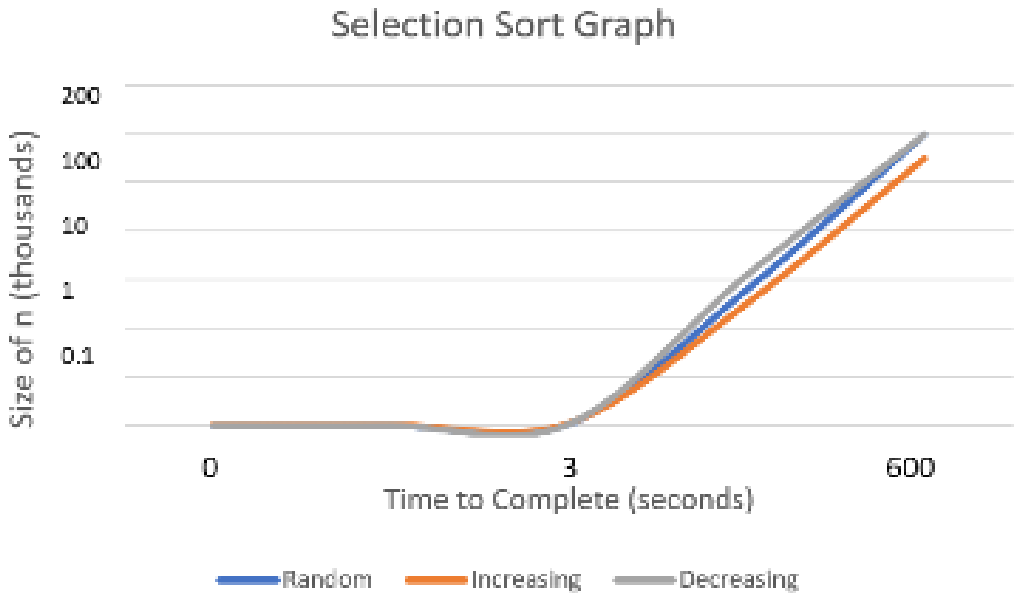
Insertion Sort

Size of n	Time to Complete (Random)	Time to Complete (Increasing)	Time to Complete (Decreasing)	Factor Increase in Time (Random)	Factor Increase in Time (Increasing)	Factor Increase in Time (Decreasing)
100	0.001s	0.000s	0.001s	N/A	N/A	N/A
1000	0.042s	0.000s	0.080s	42	N/A	80
10000	3.717s	0.002s	7.225s	88.3	N/A	90.3
100000	350.830s	0.002s	678.979s	94.5	N/A	93.8
200000	666.577s	0.026s	1290.061s	1.8	13	1.9



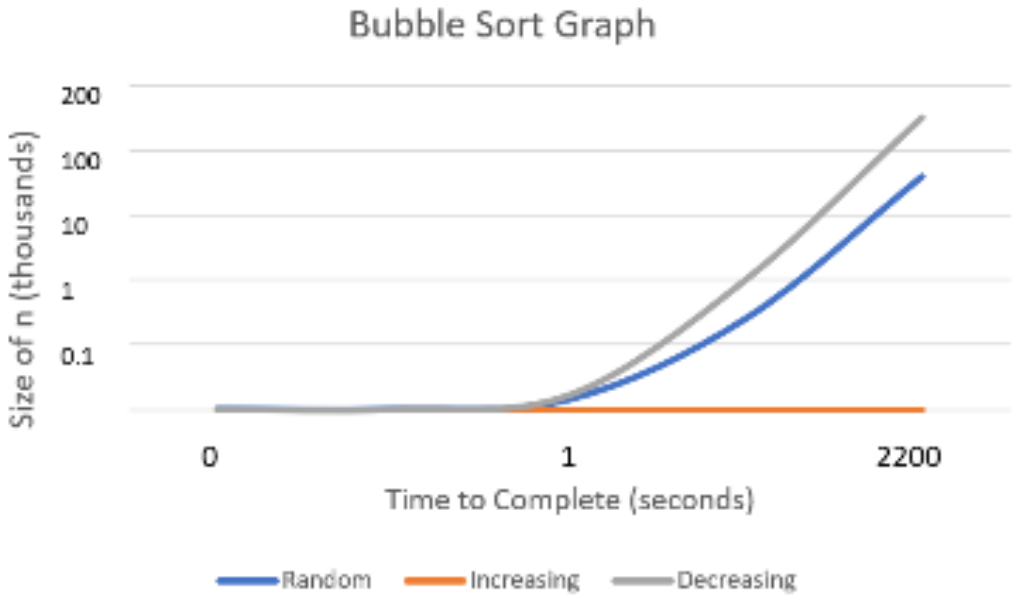
Selection Sort

Size of n	Time to Complete (Random)	Time to Complete (Increasing)	Time to Complete (Decreasing)	Factor Increase in Time (Random)	Factor Increase in Time (Increasing)	Factor Increase in Time (Decreasing)
100	0.000s	0.000s	0.001s	N/A	N/A	N/A
1000	0.036s	0.033s	0.045s	N/A	N/A	N/A
10000	3.358s	3.231s	3.849s	93.3	97.9	85.5
100000	283.753s	272.022s	291.554s	84.6	84.206	75.64
200000	599.130s	576.843s	603.954s	2.1	2.1	2.1



Bubble Sort

Size of n	Time to Complete (Random)	Time to Complete (Increasing)	Time to Complete (Decreasing)	Factor Increase in Time (Random)	Factor Increase in Time (Increasing)	Factor Increase in Time (Decreasing)
100	0.001s	0.000s	0.002s	N/A	N/A	N/A
1000	0.080s	0.000s	0.124s	80	N/A	62
10000	7.753s	0.001s	11.734s	96.9	N/A	94.5
100000	744.337s	0.007s	1007.961s	96	7	85.8
200000	1783.701s	0.013s	2214.125s	2.4	1.9	2.2



Analysis

When looking at the data collected for shell sort, we notice very fast completion times for all categories, however as the array size (n) reaches 100-200 thousand, we notice that the completion times start slowing down slightly. This is consistent with the mathematical complexity of the shell sort algorithm, as it is quite involved and should easily be the fastest of all the $O(N^2)$ time complexity algorithms.

The insertion sort results were much slower than the shell sort, which was to be expected. The time to complete for random and decreasing arrays became exponentially slower as input size increased, to the point where it took several minutes to complete, as compared to the few seconds it took for small input sizes. When sorting increasing arrays, however, it was almost instantaneous, as expected.

Selection sort had similar results to insertion sort, however, its completion time for increasing arrays also grew exponentially. Selection sort was quite slow, taking around ten minutes in the worst-case scenario. This is likely due to selection sort usually doing fewer swaps than insertion sort, making it a less complex sorting method, and one of the slower sorting methods with $O(N^2)$ time complexity.

Bubble sort was by far the slowest method for sorting random and decreasing arrays, which was expected as it is a very inefficient and simple algorithm. Bubble sort is particularly slow at sorting decreasing arrays, which is consistent with the data above. This is because it cannot swap a value towards the end with a value towards the beginning; it can only move a value by swapping it with an adjacent one. When sorting increasing arrays, bubble sort was almost instant, similar to insertion sort.