Brief Analysis regarding n^2 sorting algorithms & Priority Queues

by Shaanan E. Curtis

June 2018

It seems it took the most amount of time performing the selection sort. Because this is true, it is safe to say that the program is functioning properly. This is due to the fact that it was originally an unsorted list which, in worst case, would require O(n) time to sort each element. As for the insertion sort, it makes sense that it was second best in performance. This list is sorted by insertion, which takes less time as it does not have to sort through the entire list per element. Instead, because it is by insertion, it only takes O(n) time to perform the last insertion. However the complexity may increase linearly as it continues to insert more elements. Therefore, the bubble sort was the most efficient. This was due to the fact that it did not use more than one data structure to perform the sorting operation. It takes extra time to swap between 2 different data structures in general. Because only an array was used, less time was required.

Data collected from each run...

Size 10000

Insertion Avg = 625.64ms

Selection Avg = 2426.74ms

Bubble Avg = 309.41ms

Size 20000

Insertion Avg = 2547.16ms

Selection Avg = 9519.55ms

Bubble Avg = 1268.17ms

Size 40000

Insertion Avg = 10494.17ms

Selection Avg = 37764.30ms

Bubble Avg = 5283.05ms

Size 80000

Insertion Avg = 57405.93ms

Selection Avg = 171822.67ms

Bubble Avg = 21759.87ms

Size 160000

Insertion Avg = 393671ms

Selection Avg = 982856ms

Bubble Avg = 88254ms

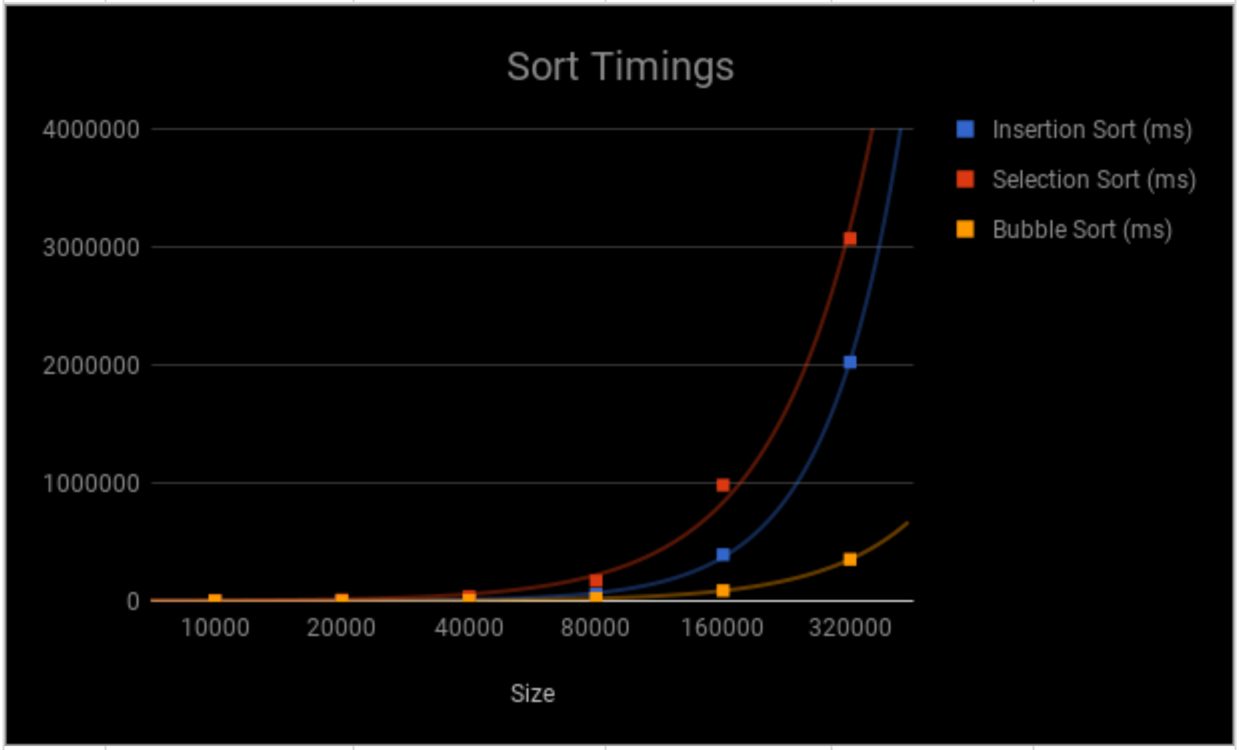
Size 320000

Insertion Avg = 2.02432e+06ms

Selection Avg = 3.06888e+06ms

Bubble Avg = 352575ms

Size	Insertion Sort (ms)	Selection Sort (ms)	Bubble Sort (ms)		
10000	625.64	2426.74	309.41		
20000	2547.16	9519.55	1268.17		
40000	10494.17	37764.3	5283.05		
80000	57405.93	171822.67	21759.87		
160000	393671	982856	88254		
320000	2.02E+06	3.07E+06	352575		


Sort Timings
Size
Insertion Sort (ms)
Selection Sort (ms)
Bubble Sort (ms)