Stockton University

DATA STRUCTURES AND ALGORITHMS I: SORTERS PROJECT

Dante Anzalone

Stockton University

CSCI 3103: Data Structures and Algorithms I

Byron Hoy

December 12th, 2023

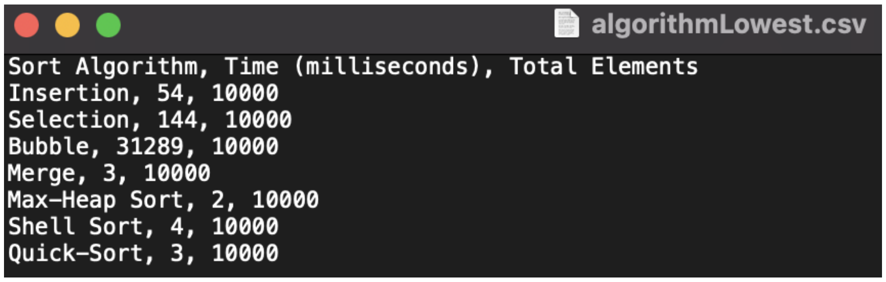
Contents

Algorithms - Sorters Results....................3
Algorithms - CSV Files....................4
Figure 1.0 - Algorithm CSV Data At 10,000 Elements....................4
Figure 1.1 - Algorithm CSV Data At 15,000 Elements....................4
Figure 1.2 - Algorithm CSV Data At 20,000 Elements....................4
Figure 1.3 - Algorithm CSV Data At 25,000 Elements....................4
Figure 1.0 - Algorithm CSV Data At 200,000 Elements....................5
Algorithms - Graphs....................5
Figure 2.0 - Sorting Algorithms At 10,000 Elements....................5
Figure 2.1 - Sorting Algorithms At 10,000 Elements: Including Bubble Sorting....................6
Figure 2.2 - Sorting Algorithms At 15,000 Elements....................7
Figure 2.3 - Sorting Algorithms At 20,000 Elements....................8
Figure 2.4 - Sorting Algorithms At 25,000 Elements....................9
Figure 2.5 - Sorting Algorithms At 200,000 Elements....................10
Figure 2.5 - Bubble Sorting Line Graph....................11
Work Cited....................12

Algorithms - Sorters Results

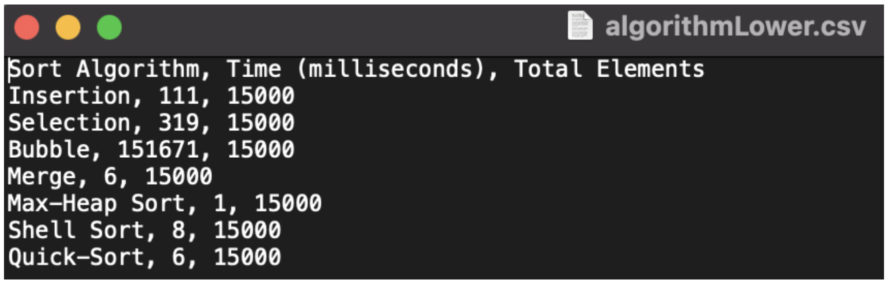
The sorters package contains two classes, Algorithm.java and SortingAlgorithms.java, which test several sorting algorithms, including bubble, selection, insertion, shell, merge, quick, and heap sorts. When the SortingAlgorithms class performs the sorting, the Algorithm class will test the sorts with between ten to two hundred thousand points, which the limitation is the computer itself and whether it has enough memory to perform these long calculations. Retrieving the data via a CSV file called algorithm.csv that shows the label of the sorter, the time it took, and how many points it included. The other figures show the run times, in milliseconds, for how long it took a sorter to perform given a specific amount of points. Referencing outside resources, whether in-class algorithms or references via the Internet, is mentioned in the work cited. Figure 2.0 shows the graph of the time, in milliseconds, it took for each sorter to perform at 10,000 elements. Selection took the longest time, insertion coming after, and the other sorters close in range and while sorting quicker. Figure 2.1 includes bubble sort, separated from Figure 2.0 because of the dramatic increase in the time it took to perform a sort on 10,000 elements. Figure 1.0 shows the times for each sorter in a CSV file. It took merge, heap, shell, and quick sorts below 5 seconds to perform, with insert sort taking 54 milliseconds, selection sort taking 144 milliseconds, and bubble sort taking 31,289 milliseconds. When increasing the elements, the time it takes becomes crucial for some sorters, as these times will increase exponentially. Figures 1.2 and 1.3 show bubble sorting roughly doubling from 203,424 milliseconds to 397,541 milliseconds. Having to put bubble sort in a graph in Figure 2.5 shows the exponential increase, revealing that sorting massive data, even small data, takes the longest and is the weakest sort out of all seven sorts. Bubble sort's average and worst comparison time is O(n2). Insertion and selection sort have the same average and worse comparison times with bubble sort. However, it takes a big data set to start noticing a waiting in compile time for the insertion and selection sorts. Selection's best comparison time is O(n2), so insertion sort is better when it comes to its best performance, especially with a lower amount of data to sort at O(n), and even though bubble sort has the same best case, it still is the worst and weak as a sorter. Note, when doing these sorting calculations, max-heap sorting was intended for a binary tree but modified to accept an array of integers to sort. However, at 200,000 elements, it only took heap sort 13 milliseconds to sort all of it, and testing 25,000 elements took heap sort 12 milliseconds. The best, average, and worst comparison time is O(n log n) for heapsort, as is merge sort and quicksort, except for its worst case being O(n2). Quicksort is efficient with large data sets, finishing faster than every sorter except for heap sort. However, heap sort is for binary trees, and quicksort finished faster than heapsort in Figures 1.1 and 1.2. Merge and shell sort seemed relatively close together, with merge finishing before shell, as expected, except for Figure 1.3. Shell sort has a best comparison time of O(n76), average of O(n54), and worst of O(n2). Based on their big O times, Heap and Merge are efficient, and then comes quicksort, shell sort, insertion sort, bubble sort, and selection sort. According to the calculations performed in the Algorithm programs, the order would be a heap, quick, merge and shell, insertion, selection, and bubble sort.

Algorithms - CSV Files



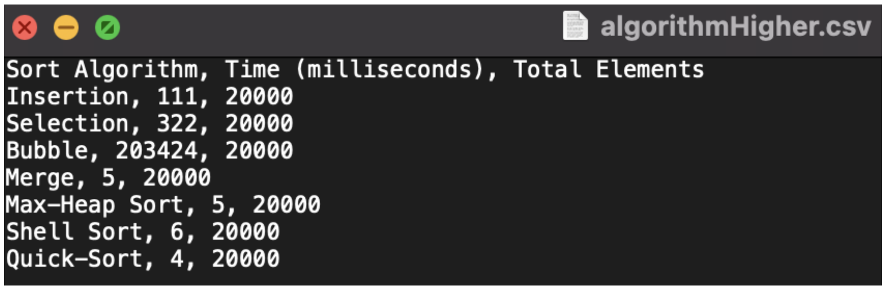
```
Sort Algorithm, Time (milliseconds), Total Elements
Insertion, 54, 10000
Selection, 144, 10000
Bubble, 31289, 10000
Merge, 3, 10000
Max-Heap Sort, 2, 10000
Shell Sort, 4, 10000
Quick-Sort, 3, 10000
```

Figure 1.0 - Algorithm CSV Data At 10,000 Elements



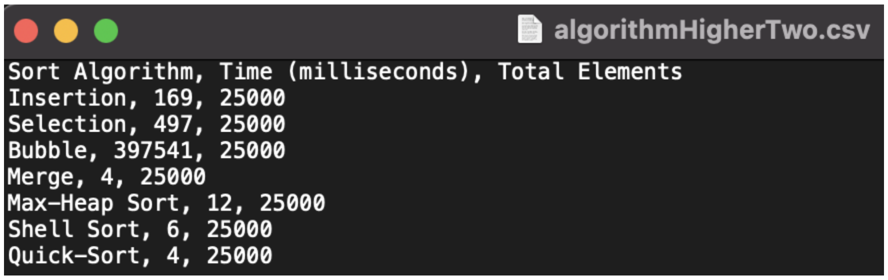
```
Sort Algorithm, Time (milliseconds), Total Elements
Insertion, 111, 15000
Selection, 319, 15000
Bubble, 151671, 15000
Merge, 6, 15000
Max-Heap Sort, 1, 15000
Shell Sort, 8, 15000
Quick-Sort, 6, 15000
```

Figure 1.1 - Algorithm CSV Data At 15,000 Elements



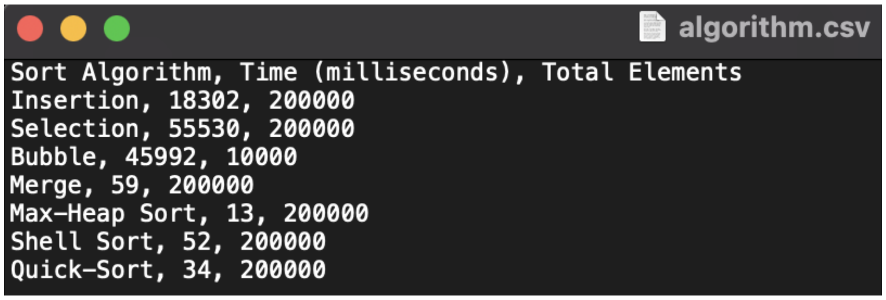
```
Sort Algorithm, Time (milliseconds), Total Elements
Insertion, 111, 20000
Selection, 322, 20000
Bubble, 203424, 20000
Merge, 5, 20000
Max-Heap Sort, 5, 20000
Shell Sort, 6, 20000
Quick-Sort, 4, 20000
```

Figure 1.2 - Algorithm CSV Data At 20,000 Elements



```
Sort Algorithm, Time (milliseconds), Total Elements
Insertion, 169, 25000
Selection, 497, 25000
Bubble, 397541, 25000
Merge, 4, 25000
Max-Heap Sort, 12, 25000
Shell Sort, 6, 25000
Quick-Sort, 4, 25000
```

Figure 1.3 - Algorithm CSV Data At 25,000 Elements



```
Sort Algorithm, Time (milliseconds), Total Elements
Insertion, 18302, 200000
Selection, 55530, 200000
Bubble, 45992, 10000
Merge, 59, 200000
Max-Heap Sort, 13, 200000
Shell Sort, 52, 200000
Quick-Sort, 34, 200000
```

Figure 1.0 - Algorithm CSV Data At 200,000 Elements

Bubble Sorting is at 10,000 elements since the other sorting algorithms, specifically Insertion and Selection sorting, took longer than before. Before termination, it took over two hours to compile with all sorters at 200,000 elements.

Algorithms - Graphs

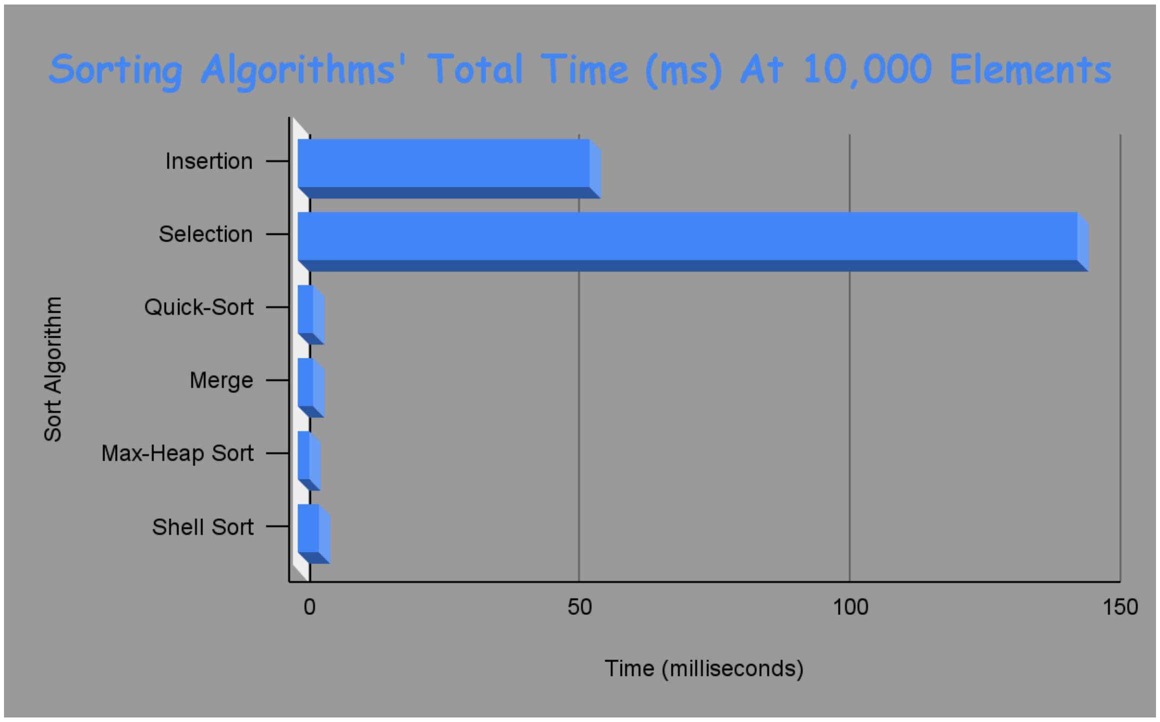


Figure 2.0 - Sorting Algorithms At 10,000 Elements

Selection sort took the longest to compile, with insertion sort reaching over 50 milliseconds. Quick, merge, heap, and shell sorts are within range of each other due to the lower amount of elements and these four sort's efficiency.

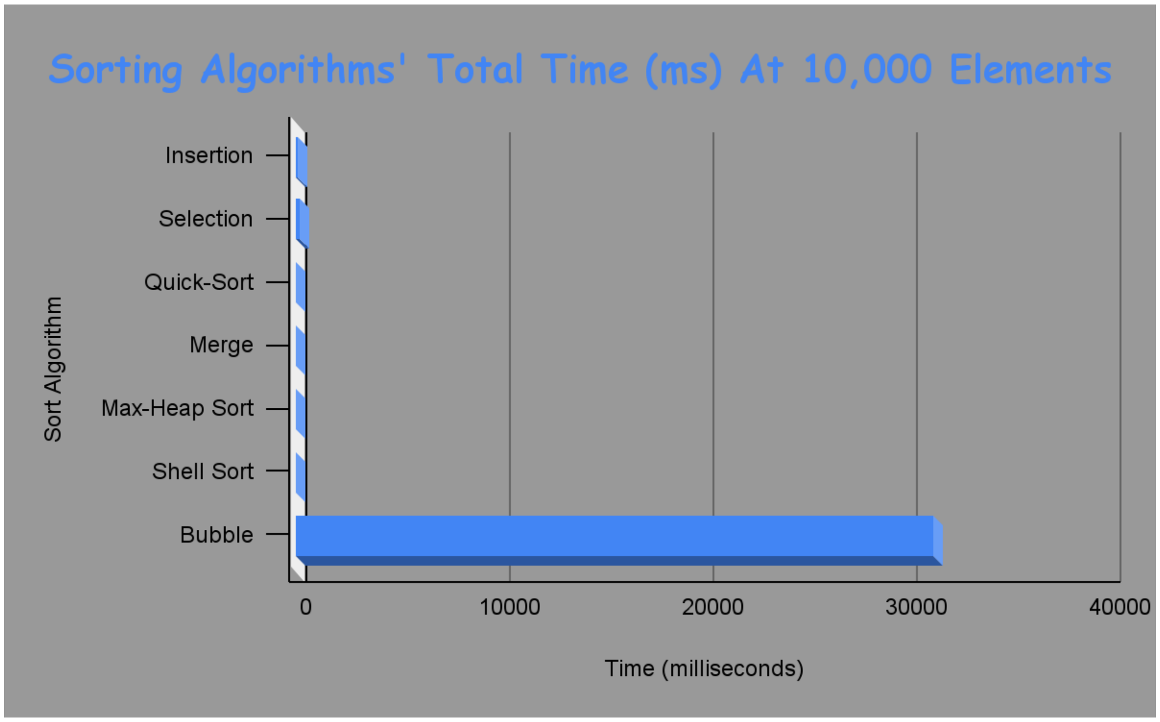


Figure 2.1 - Sorting Algorithms At 10,000 Elements: Including Bubble Sorting

Added Bubble sort to the graph, and as demonstrated via Figure 2.1’s graph, the time it took to compile exceeded 30,000 milliseconds, making it difficult to see Figure 2.0’s difference between the other remaining sorters. Bubble sort will be kept out of future graphs until Figure 2.5’s collection of Bubble sort results.

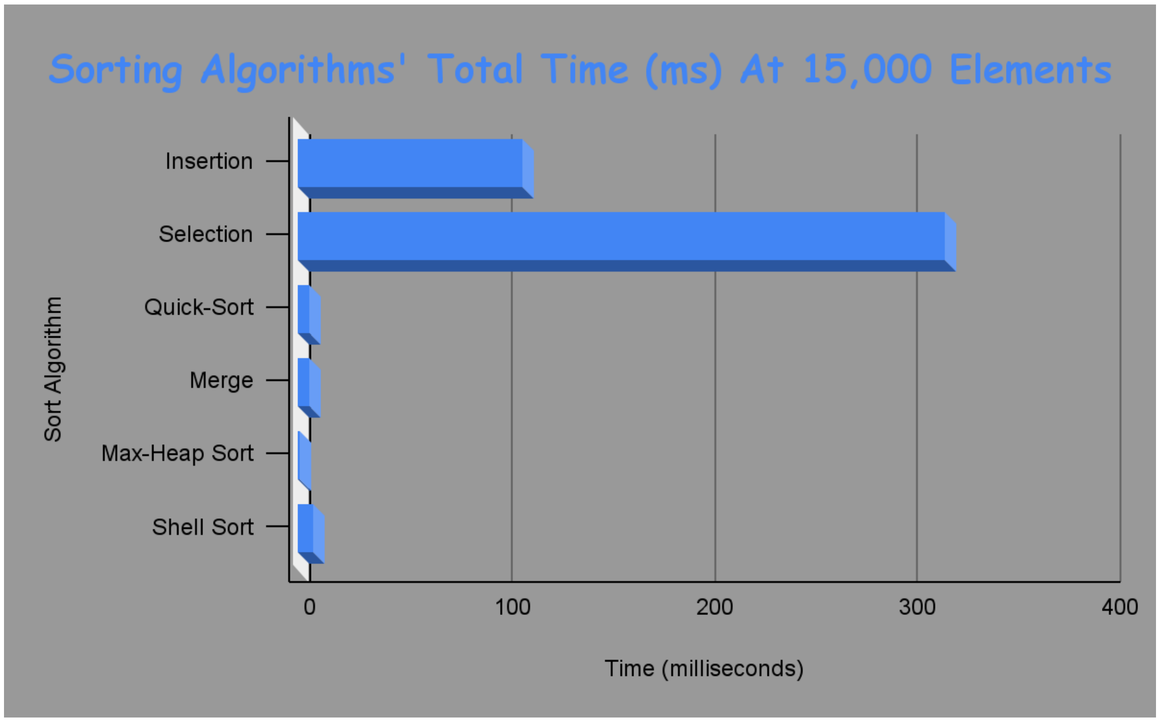


Figure 2.2 - Sorting Algorithms At 15,000 Elements

Insertion sort doubled in time with an added 5,000 elements and did Selection sort.

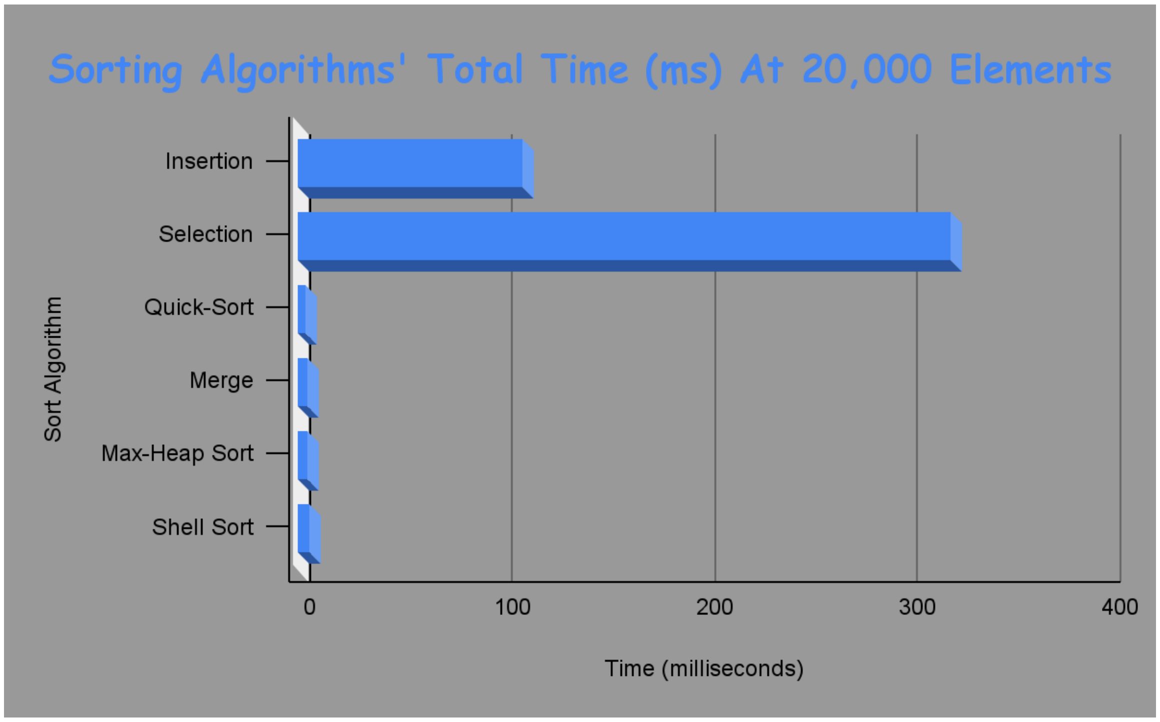


Figure 2.3 - Sorting Algorithms At 20,000 Elements

Not much change from Figure 2.2, the sorters are similar compared to their last compiling times.

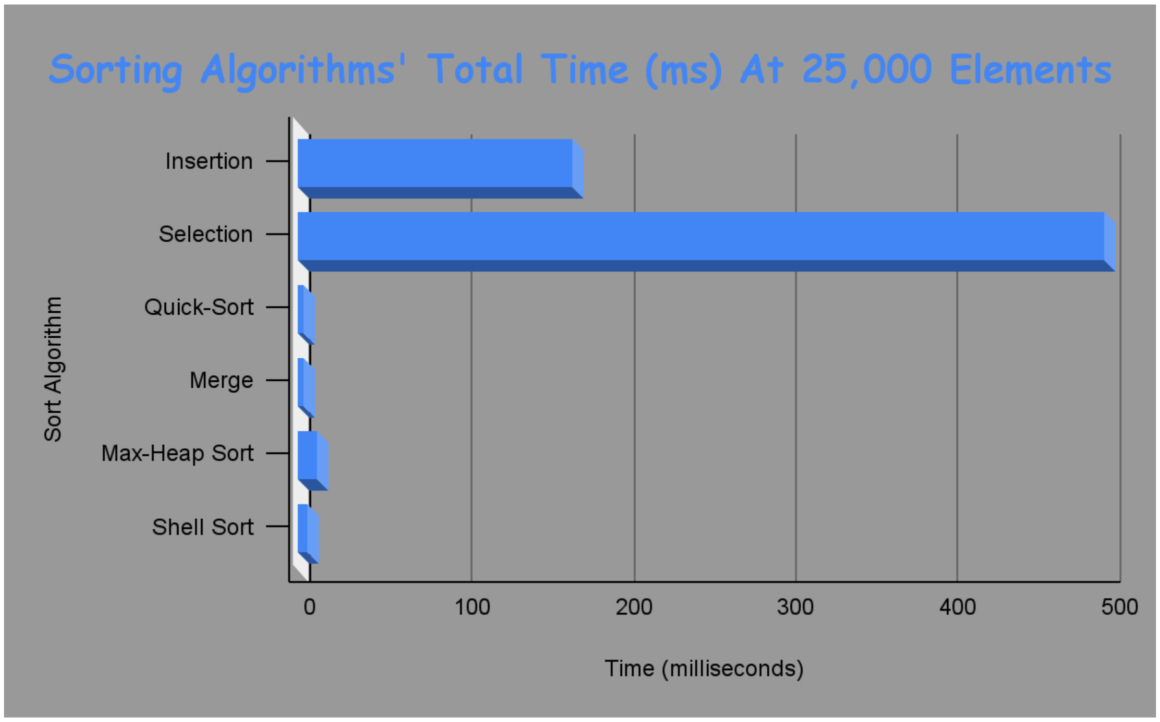


Figure 2.4 - Sorting Algorithms At 25,000 Elements

Insertion and Selection sorters have increased again, perhaps these sorters notice more change every 15,000 elements added.

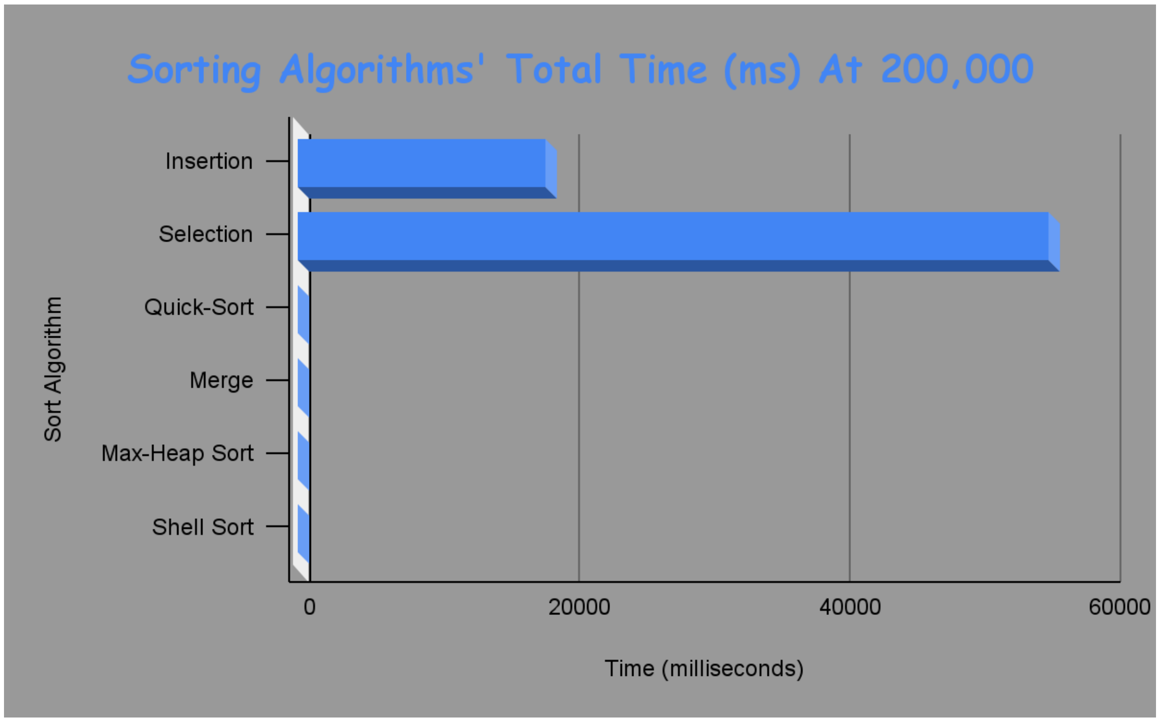


Figure 2.5 - Sorting Algorithms At 200,000 Elements

At 200,000 elements, Insertion and Selection sorts are taking 20,000 and about 55,000 milliseconds, respectively.

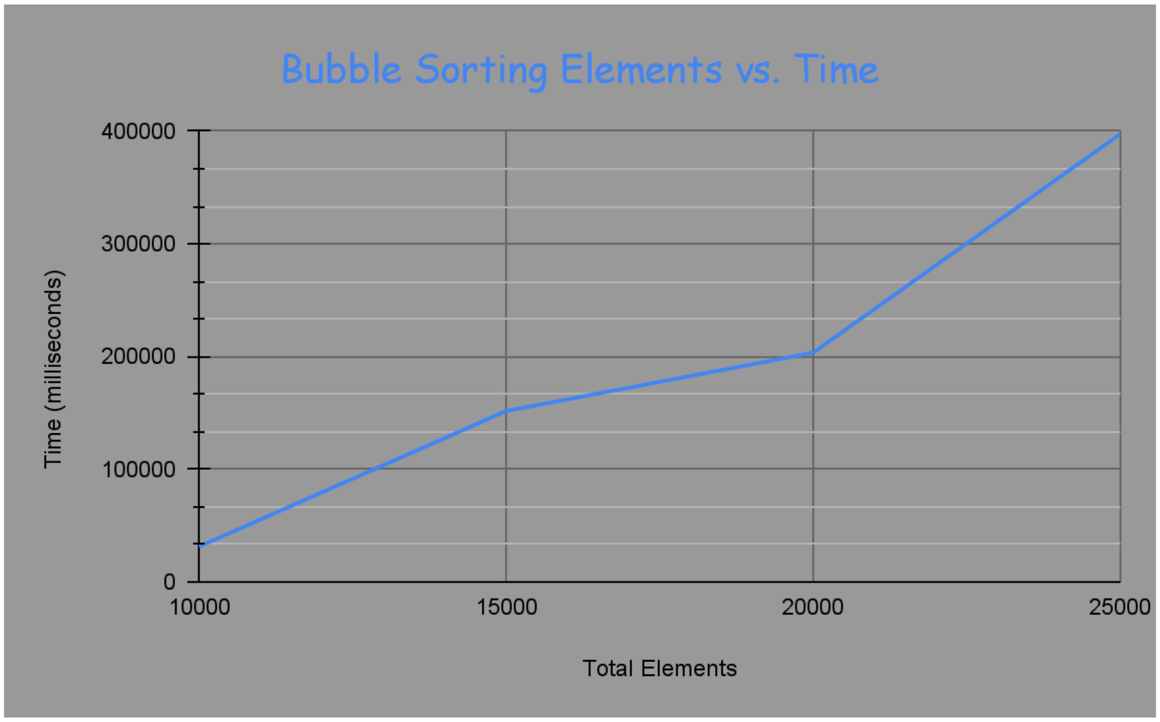


Figure 2.5 - Bubble Sorting Line Graph

Bubble sorting takes the longest time compiling than any other sorter, and after 20,000 elements, the time sky rockets. It becomes impractical to use.

Work Cited

baeldung. "Quicksort Algorithm Implementation in Java | Baeldung." *Www.baeldung.com*, 2 Oct. 2018, www.baeldung.com/java-quicksort.

"Data Structure and Algorithms - Shell Sort - Tutorialspoint." *Www.tutorialspoint.com*, www.tutorialspoint.com/data_structures_algorithms/shell_sort_algorithm.htm.

"Data Structures: Heaps." *Www.youtube.com*, www.youtube.com/watch?v=t0Cq6tVNRBA&t=295s&ab_channel=HackerRank.

"How to Build a Heap from an Array." *Educative: Interactive Courses for Software Developers*, www.educative.io/answers/how-to-build-a-heap-from-an-array.