Susannah Bennett

Dr. Pohly

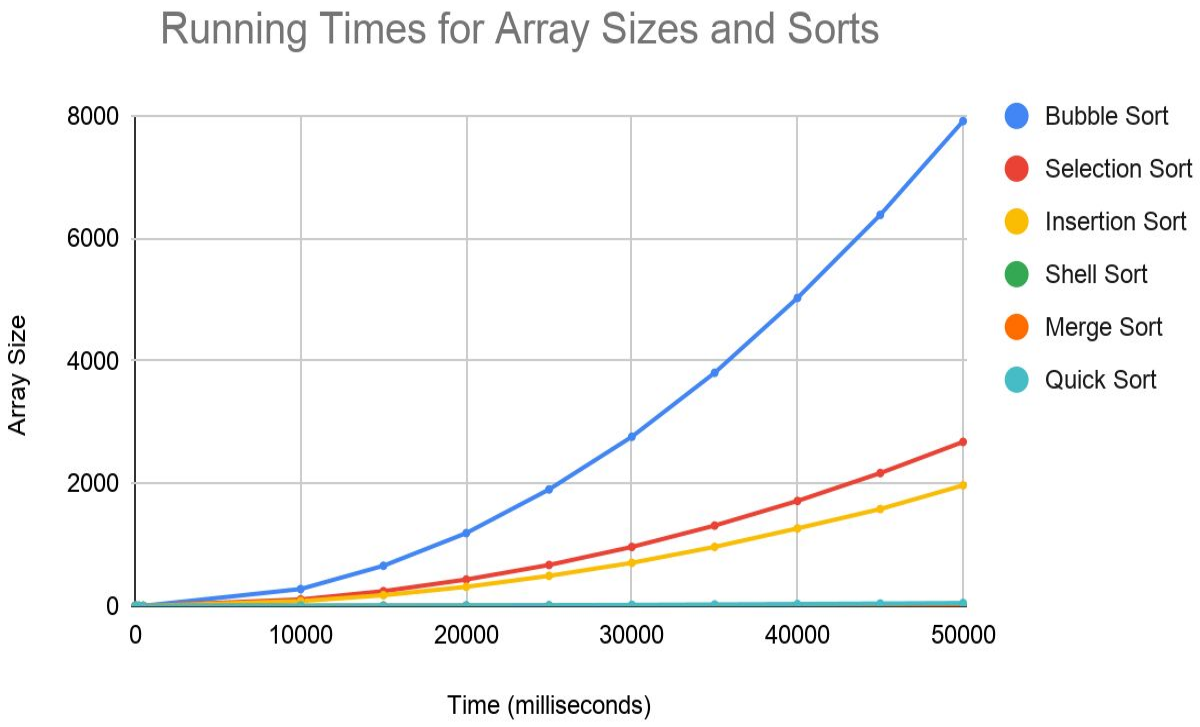
CSCI 245

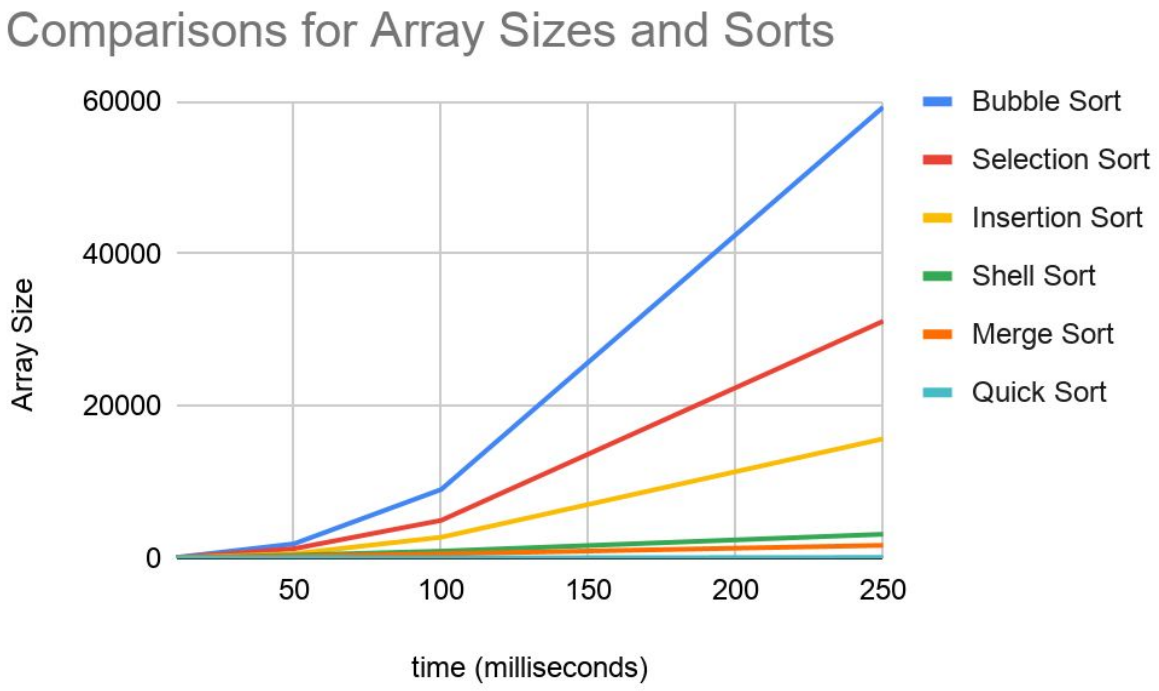
3 February 2020

In my experiment, I examined how running times and comparisons inform the user of the efficiency of a program. Similarly to what we tested in lab, I printed out in terminal the number of comparisons and, under that, the running times in milliseconds per implementation of each sorting method. Generally, the comparisons were helpful to gauge roughly the patterns in efficiency of the programs, but for some of the methods, the comparisons were completely unhelpful and inaccurate. For example, the bubble sort had the most comparisons of the different algorithms and the longest running time, which was a helpful correlation to note. Additionally, however, the quick sort had the few comparisons by far but did not have the shortest running time. In most cases, the comparisons were largely unhelpful in assessing the running times, although there were a few cases of it correctly proving how efficient one algorithm was over another.

The rates of the comparisons and running times were difficult to compare because of the difference in testing array sizes. To better compare the two, I had the running times also start with the array size of 10, even though the running times were very small numbers with such small array. From what I could assess, the comparisons and running times didn't increase at equivalent rates but did maintain somewhat similar trends of increase and efficiency. At most, the comparisons and running times are loosely related, both depicting trends with running times being more accurate concerning efficiency than comparisons. For bubble, selection, and insertion

sorts, there were parallels between the number of comparisons and running times, but for the other algorithms, the numbers differed greatly, with the comparisons inaccurately showing the efficiency (considering the smaller numbers are most efficient).


Running Times for Array Sizes and Sorts
8000
6000
4000
2000
0
Array Size
0
10000
20000
30000
40000
50000
Time (milliseconds)
Bubble Sort
Selection Sort
Insertion Sort
Shell Sort
Merge Sort
Quick Sort


Comparisons for Array Sizes and Sorts
60000
40000
20000
0
Array Size
50
100
150
200
250
time (milliseconds)
Bubble Sort
Selection Sort
Insertion Sort
Shell Sort
Merge Sort
Quick Sort