Sorting Algorithm Experience

By Venezia Quezada

In this assignment, we were supposed to implement QuickSort, Merge Sort, Selection Sort, Insertion Sort, and Bubble Sort. I was interested in being able to see the differences in an example rather than just knowing the run times.I tested them three times, one time with ten numbers, then 100, and finally, 1000 to see how they reacted with different size data sets.In the first test with ten numbers, they all performed pretty closely. But the two who ran the fastest by .001 milliseconds. Because the data set was so small, I believe that all of them were able to premium pretty quickly. The second data set with 100 numbers quick sort perfumed the fastest at 0.009 compared to the slowest bubble at 0.045 milliseconds. This was interesting to see because I was able to see which larger data sets affected sorting algorithms and which ones stayed consistently quick. Finally, for the third test, the most extensive data set with 1000 numbers. In this test again, the quicksort was the fastest, followed by merge sort. But something new was bubble wasn't the slowest was selection sorting. There are all different types of trade-offs with sorting algorithms because some perform better with different size data sets, and I think that's the essential part to remember. If you have a data set that will forever be small, whatever sorting algorithm can do the job. But for large data sets, it's better to use the ones that are the best for that. For data sets that are continually changing in this case, I would pick a quick sort based on my test trials because it performed the fasting in all three tests. I think my programming language choice affected how the sorting algorithms work because of how it needs to run and compile. Some shortcomings of the empirical analysis are that its expensive. It would be

best if you had the right hardware or platform to run it because it could take up a lot of time.