Emilee Mason

Professor Rene German

CPSC 350 - Data Structures and Algorithms

19 December 2020

Sorting Algorithms Analysis

BubbleSort can quickly sort through between 1-10,000 values, but after that point, it becomes very time consuming. It was able to work through 10,000 values within a second, but took around 52-64 seconds to get through 100,000. It's not worth the wait to see any other values that may take longer than this, cause at this point we know it's doing to increase quadratically, which means it could take BubbleSort around 5-10 minutes to get through 1,000,000 values.

SelectionSort was similar to BubbleSort, where between 1-10,000 values was quick. When sorting through 100,000, it took 10-12 seconds and for 1,000,000 values, it took over 12 minutes to finish. I had to cancel the process because it was taking too long.

InsertionSort either immediately or took around 20 seconds to sort through 10,000 values, and took around 12 seconds to sort through 1,000,000 values, which is much faster than both Selection and Bubble sort. I tried to test 10,000,000 values just to see what would happen, and it took over 10 minutes again.

MergeSort was able to sort through 10,000 values, 100,000, and even 1,000,000 values within 1-2 seconds. At 10,000,000 values, it took 5 seconds, and at 100,000,000, the program actually killed itself within around 5 seconds, most likely from the high volume of doubles. I tried 20,000,000 as well just for fun and it also killed that one within the same amount of time as the 10 million one.

QuickSort, like merge sort, was able to sort through 10,000 - 1,000,000 values within a second or two. It only took 3 seconds to sort through 10,000,000 values as well! When I did 100,000,000 values, it was never killed and actually only took around 26 seconds to sort through all of them!

For some reason, I expected Selection Sort and Bubble Sort to have similar results, but now I realize that the amount of swaps that occur during the sorting can really slow down the entire process. And I also thought Insertion Sort would be able to run a more significant amount faster than Selection Sort, and although it does run quicker for smaller amounts of data, it still gets super slow at the same points as Selection Sort.

Merge Sort and Quick Sort are such a significant amount more efficient than the other three sorting algorithms that it's really mind blowing. I didn't realize that even though they are a little bit hard to implement, these sorting algorithms are really worth the time and effort you put into them. Not that they were super hard to implement, but it definitely took a lot more time to get them set up and working correctly. I also thought it was a little strange that Quick Sort seemed to run a little bit better than the Merge Sort in terms of handling more values, but that might have to do with my machine and the quality of the code I made for the two algorithms.

Although this was a super fascinating and super accurate way of testing how various algorithms work, empirical analysis isn't the most effective and efficient way to do it. It took a couple days to implement all the algorithms and get them all working properly before I could actually start conducting any tests on them. Once they were all working, there's a lot of interesting things you can learn from them, which I can appreciate as a student. However in the business world, more time means money and you can't really afford to be spending hours figuring out why things are working the way that they are and what makes them more efficient or not. You need to be able to analyze situations and get good, working solutions quickly to minimize the costs.

Also, my machine had issues running the Merge Sort algorithm I created, but other machines and compilers might actually show that the algorithm works perfectly fine. The hardware and platforms start to really affect the way the code runs, which shouldn't really be a factor when you're trying to analyze and find the right solutions, especially when working on projects that are not exclusively on a certain OS or specific computer setup. A lot of bigger businesses have more expansive reaches over the globe, and will need to find solutions that will fix the problems for every computer that's having them, not just my specific Dell Laptop.

Overall this was an interesting assignment, and I'm glad I was able to analyze the data in this manner because it really did help me understand some concepts about these sorting algorithms that I couldn't have understood by just reading it on paper. It's beneficial to know how each algorithm works and what makes them more effective as well as what their downfalls are.