Differences Between Five Different Sorting Algorithms

Sebastian Brumm

December 8, 2019

1 The Algorithms

The algorithms that I used in this assignment are bubble sort, insertion sort, selection sort, merge sort, and quick sort. Bubble sort is supposed to be the slowest among them with a run-time of O(n^2) in every scenario but being very simple to implement. Insertion sort and selection sort have the same run-time of O(n^2) in the average case but can run better if the collection meets certain criteria. It is the same situation with merge and quick sor, except that in they average case they only have a run-time of O(nlogn).

1.1 Time differences

When I ran each of the algorithms, I progressively used more elements until I got to 100,000, at which point I started getting segmentation faults. My computer is very fast so even at that point the longest algorithm only took about 20 seconds, that being bubble sort. Both merge sort and quick sort happened almost instantaneously. I expected about this difference in run-time but I didn't expect my computer to run through all of them so fast.

1.2 Tradeoffs

Each algorithm has different tradeoffs, mainly pertaining to their run-time and complexity. Bubble sort is the slowest but least complex while merge sort is the fastest but most complex. Merge sort also uses a lot of memory, which is where I started getting segmentation faults when using a large amount of numbers.

1.3 Programming Language

I used C++ for the implementation of these algorithms. This is better than using something like Java or Python because you can dynamically allocate the memory in C++ to have the program run faster and more efficient.

1.4 Shortcomings Of Analysis

Since we are just running the algorithms and timing them, we can't be super efficient in terms of time. Using mathematical analysis is way more efficient, even if it is not always as accurate and does not account for all scenarios.