The easiest algorithms to implement were the naïve algorithms (bubble sort, linear search). While simpler to conceptualize and translate into code, the naïve algorithms are not as efficient as the more robust algorithms (binary search, merge sort).

The differences were as follows:

A screenshot of a computer

AI-generated content may be incorrect.

Generally, the efficient algorithms were faster than the simple sorts with the exception of sorting at 1000 elements. Since the current implementation of merge sort being used here creates 2 new arrays per level of recursion, it has a massive overhead cost compared to bubble sort, and is therefore slower when sorting arrays around 1000 elements long.

When selecting which algorithm to use in my capstone, I would consider the size and uniformity of the data I’m working with and choose to implement the most efficient algorithm for the task. If I choose to keep my current version of merge sort, this would mean relying on bubble sort for smaller arrays and merge sort for larger arrays. Like how C# implements an introspective sorting layer in its sort() methods, I am considering implementing a layer that analyzes the size and uniformity of an array before passing it off to the most efficient method.