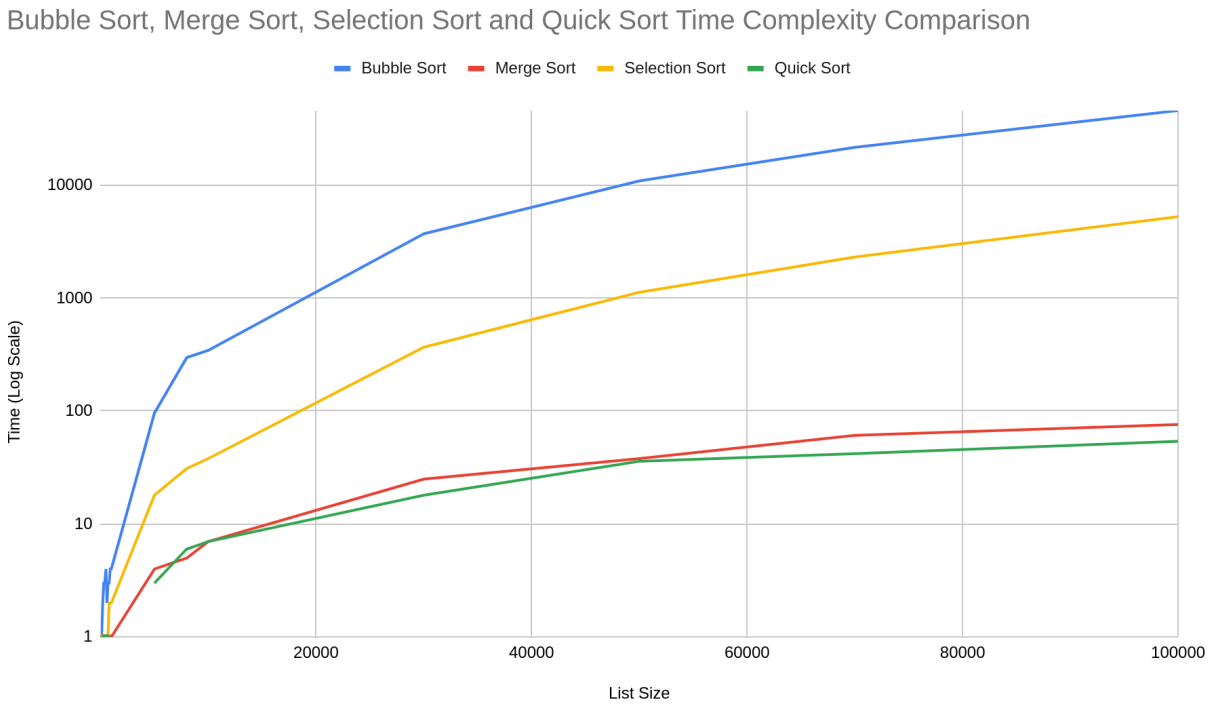
Benchmarking Functions

The 4 sorting algorithms I chose was Bubble Sort, Merge Sort, Selection Sort, and Quick Sort. Out of the 4, Merge & Quick sort were the “Divide & Conquer” style algorithms that I implemented recursively.
ear
Below is the benchmarking that I conducted on the following algorithms.



I tested list sizes with randomly generated numbers from 17 list sizes from 10 to 100,000 with the time being in Milliseconds. I made a function called measureTimeMillisPair to calculate the runtime of each algorithm on the different list sizes.

My findings were that Merge sort & Quick sort, the two “Divide & Conquer” Recursive algorithms, are almost the same in terms of speed, though Quick sort was noticeable faster on average which is consistent with what is to be expected since in practice quickSort is one of the fastest algorithms.

Research Paper

The paper I read was:https://www.nature.com/articles/s41586-023-06004-9

This paper introduces AlphaDev, an AI system that uses Deep Reinforcement Learning and stochastic search optimization algorithms to generate improved sorting algorithms. Instead of trying to refine already existing sorting algorithms to improve time complexity mathematically, AlphaDev prioritizes optimization of the CPU's performance. AlphaDev not only utilizes AI but it specifically focuses on optimization on the CPU level by working on low level programming, specifically in assembly & C++. This allows AlphaDev to take into consideration how memory and registers are impacting during the sorting algorithm performance.

The paper goes into detail as to how they trained AlphaDev, specifically through a game called AssemblyGame which would reward low latency and correct sorting. After training the AI system, it will generate assembly code that is significantly better than if it was written by a human or following a typical sorting algorithm implementation. Again, AlphaDev does not prioritize generating a new and never seen sorting algorithm. Rather, it is able to implement the most resource-optimized assembly code to sort, and may potentially generate a new sorting algorithm. One example of this is that AlphaDev created a new algorithm were based on the size of the list/sequence, it would call different algorithms. This makes sense since depending on the list size, different algorithms can become faster or slower. For example, even in our benchmark testing, we can see for small values of list sizes, merge sort was preferable to quicksort.

Master Theorem