Tushar Ganta

PROGRAM 4 REPORT

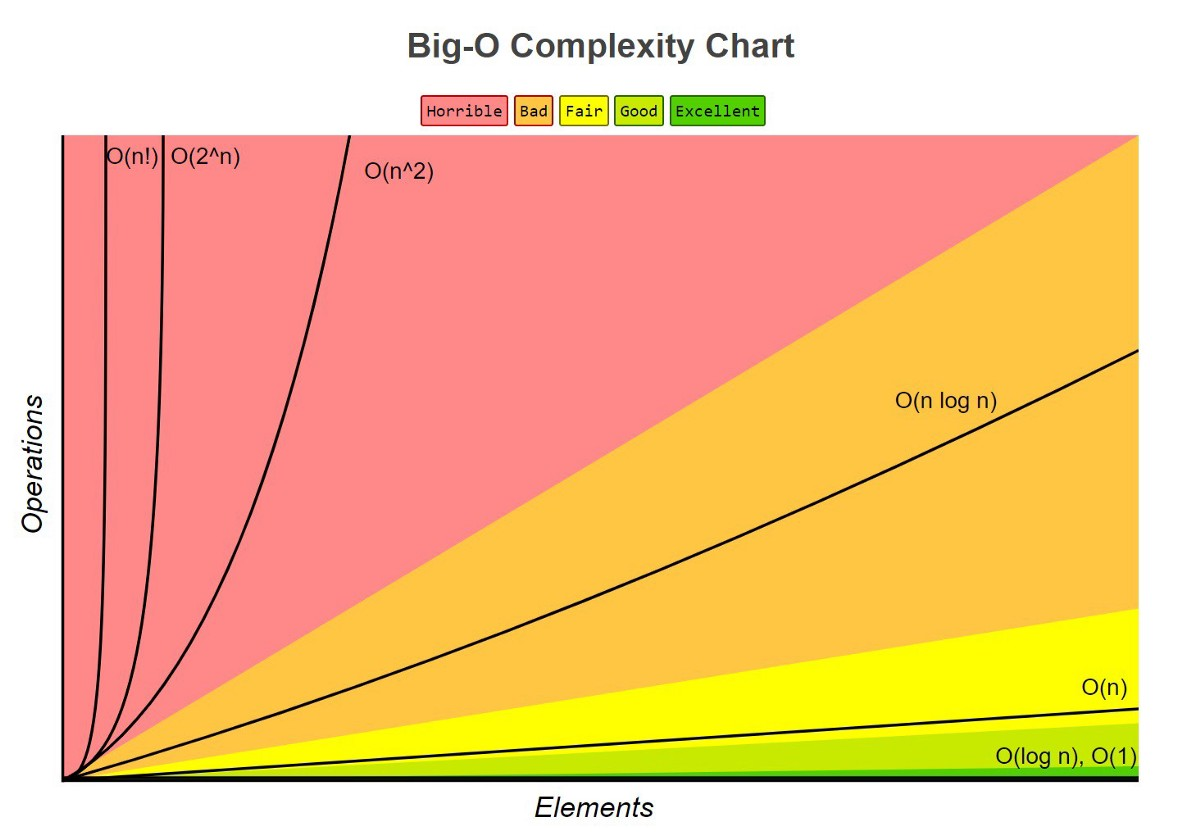
November 22, 2022

**WHAT ARE SORTING ALGORITHMS?**

Sorting Algorithms are a program which is used to rearrange data or a list of elements based on how we are comparing the data. In this report, we will be talking about 6 different sorts and their Big O Notation. Big O Notation is used to show the space and time complexity of certain algorithms. This also indicates the efficiency of the algorithm itself. The 6 Sorts we will be talking about today are; BubbleSort, InsertionSort, MergeSort, QuickSort, ShellSort, and the Iterative Merge Sort.

**SORTING ALGORITHMS AND THEIR RUNTIMES**

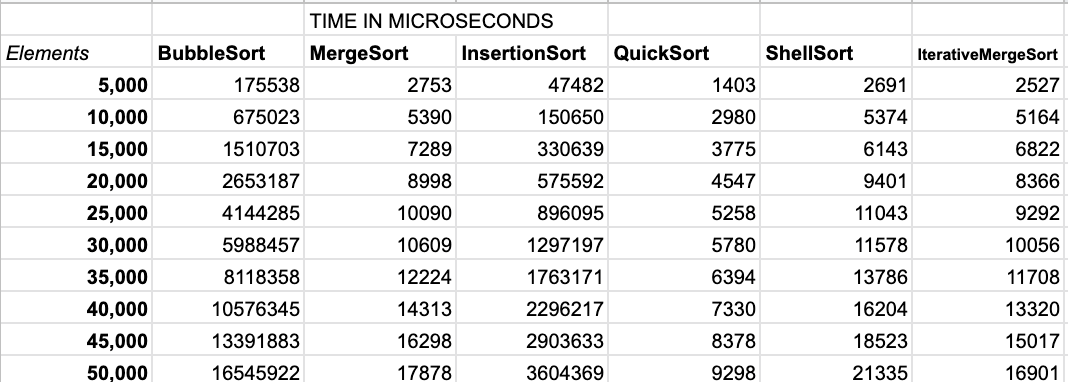
|  |  |  |
| --- | --- | --- |
| Sorting Algorithm | Worst Case | Average Case |
| Bubble Sort | O(n^2) | O(n^2) |
| Insertion Sort | O(n^2) | O(n^2) |
| Shell Sort | O(n^2) | O(n^3/2) |
| Merge Sort | O(n log n) | O(n log n) |
| Iterative Merge Sort | O(n log n) | O(n log n) |
| Quick Sort | O(n log n) | O(n log n) |

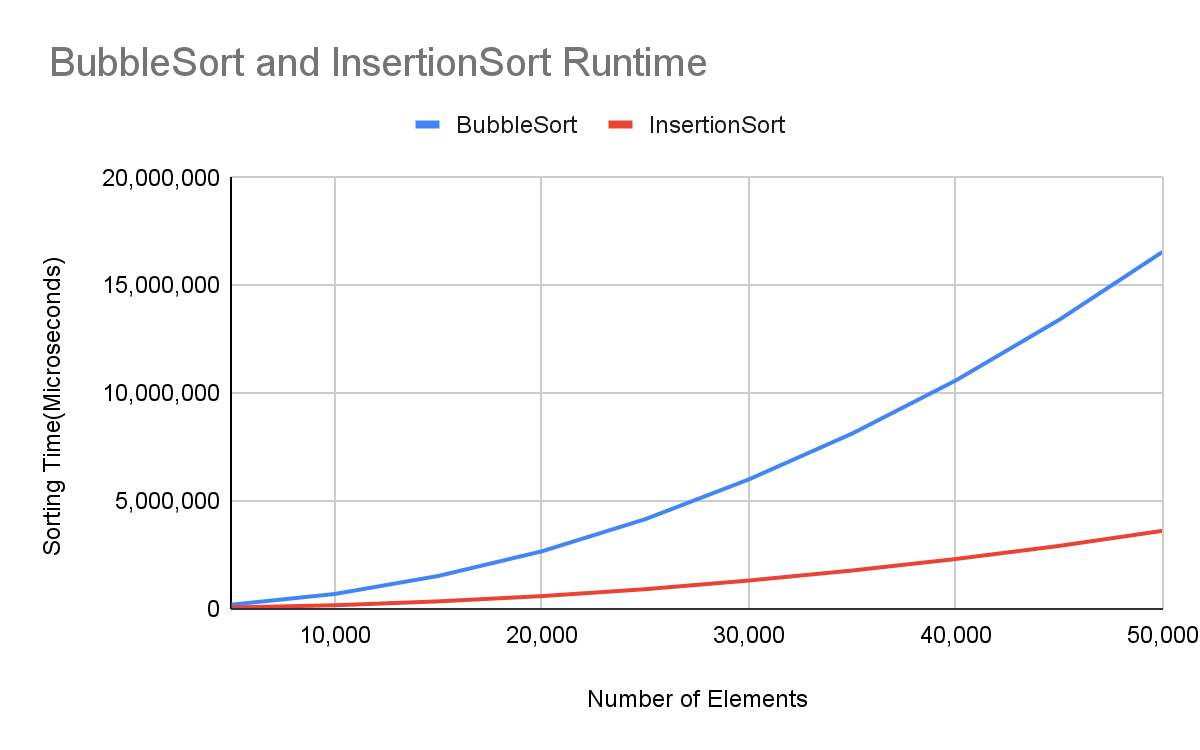
****

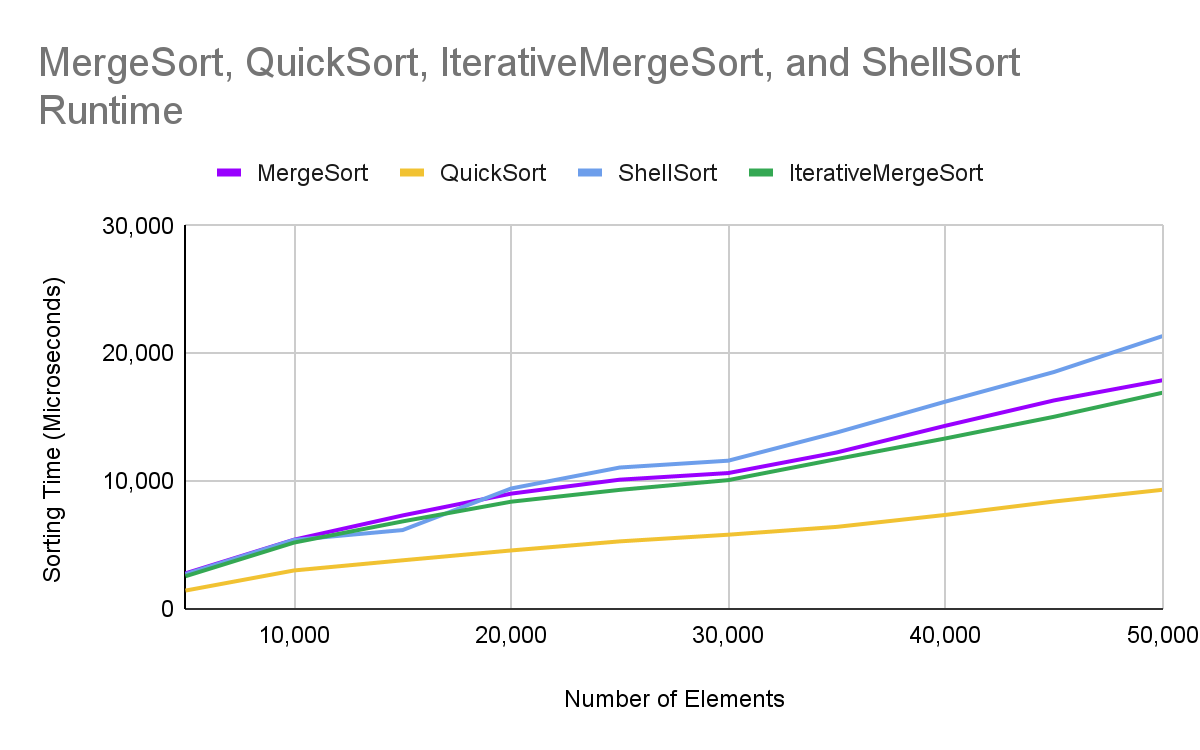
**WHAT DO THESE MEAN?**

* If an algorithm is in **O(1)** time, that means no matter how much data is inputted, the time to run it will stay the exact same. This is known as Constant Time.
* If an algorithm is in **O(N)** time, that means the time to run the code increases linearly with the elements inputted. This is known as Linear Time.
* If an algorithm is in **O(logN)** time, that means in each step of the algorithm, the amount of data processed reduces. This is known as Logarithmic Time.
* If an algorithm is in **O(N^2)** time, that means there are normally nested loops. This is known as Quadratic Time.

We tested the above Sorts with different numbers of elements, and we counted how long it took to sort the given number of elements in Microseconds in the table below



Below are the Graphs of the plotted data above.

Above is the chart of the most time consuming Sorts; BubbleSort and InsertionSort. These are on a chart of their own because the other sorts do not take nearly as much time as these 2, so they barely show up on the graph. These both have the Time Complexity of O(N^2). ShellSort also has O(N^2), but the way it swaps data makes it quicker than BubbleSort and Insertion Sort. The chart below is the rest of the sorts.

In the graph above, we can see that the ShellSort takes the longest amount of time because it is O(N^2). MergeSort and Iterative Merge Sort are almost the same, as the one done recursively takes a little longer to complete, both of which are O(logN). QuickSort is the fastest, hence its name; this is also O(logN).

Overall, with these charts, it is evident as to which sorts are quicker than others. As data gets larger, the sorts with longer runtimes will get even longer, while those are relatively steadily going up will continue to follow that path.