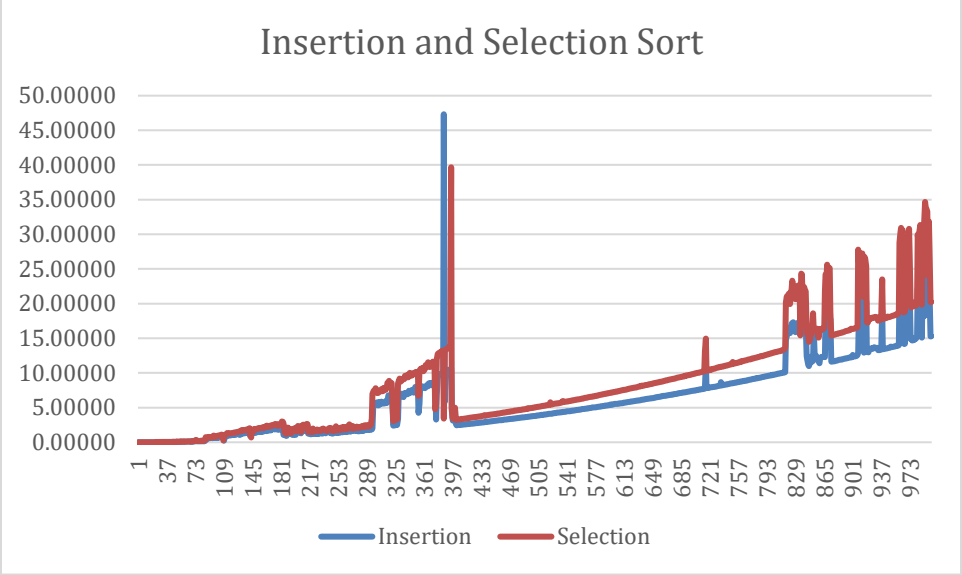
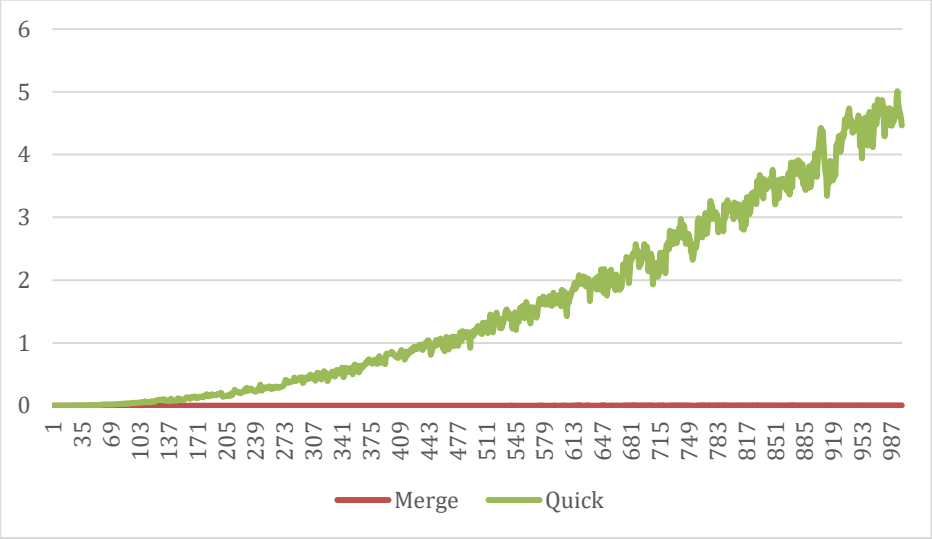
NAME:	Prithvi Singh
UID:	2022301014
SUBJECT	DAA
AIM:	Graphs of all Sorting Algorithms.
Graphs and Observation:	 • **<u>Insertion Sort:</u>** Here as we can see in the graph for insertion sort that as the values change time required for sorting changes too. This can be understood by seeing throttling at different stages in the execution. Sometimes also due to different values in each block the time to sort those blocks also differs. We can also accumulate from the graph that insertion sort for the most amount of running time takes less time than compared to selection sort. Despite of throttling we can observe that graph moves almost evenly for most amount of running time. Also, the running time differs from system to system as each system has different characteristics and specifications.

- **Selection Sort:**

Sometimes also due to different values in each block the time to sort those blocks also differs. We can also acknowledge from the graph that selection sort for the most amount of running time takes more time than compared to insertion sort. Despite of throttling we can observe that graph moves almost evenly for most amount of running time and also almost linear to insertion sort at the end stages of the execution.



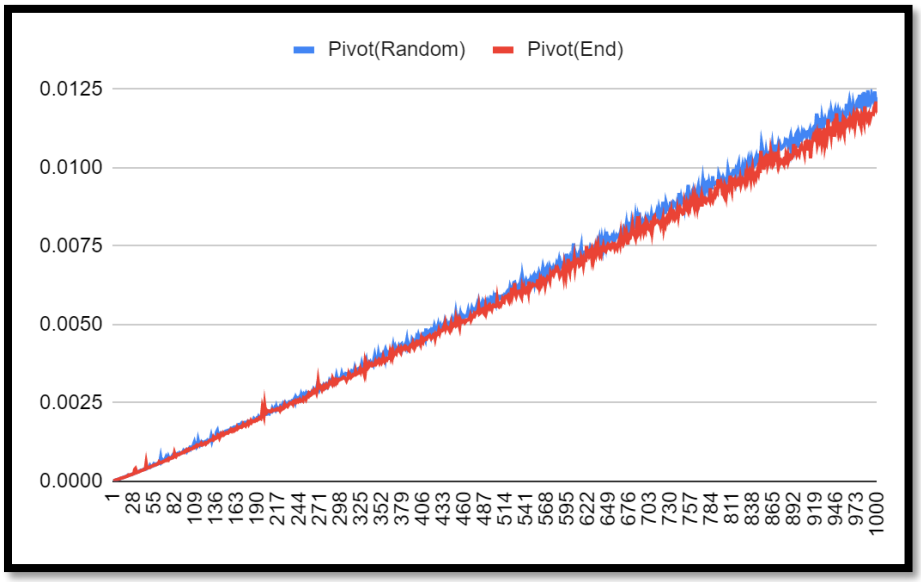
- **Merge Sort:**

Time complexity of merge sort as represented in red we can see that according to values there isn't any drastic changes throughout execution of the program. Majorly the whole sorting algorithm is executed within 1 second. It takes less time as compared to sorting the same values using quick sort algorithm.

- **Quick Sort:**

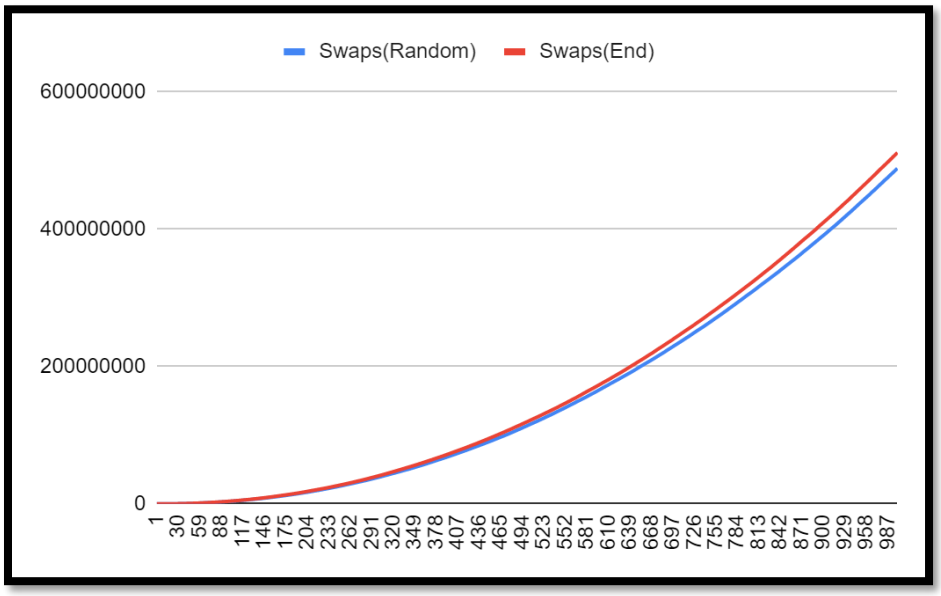
Time complexity of quick sort as represented in red we can see that according to values there are comparatively drastic changes as the execution of program is reaching its end. We can accumulate that quick sort takes more time than merge sort for same values.

- **Now considering Different Pivot Positions:**



Here we can understand that even after considering different pivot positions the time complexity of quick sort is almost same. We can clearly accumulate that quick sort when pivot is at random position takes more time than compared to when pivot is at end position at the end of execution. Although both executions are completed within 0.1 seconds.

- **Count of swaps considering different pivot positions:**



	Here we can understand that Number of swaps required for quick sort when pivot is at random position is less than for quick sort when pivot is at end position. The average number of swaps are almost more than 500000000. The number of swaps is constantly increasing throughout the whole execution.
Conclusion:	Thus, we have provided observations for different sorting algorithms.