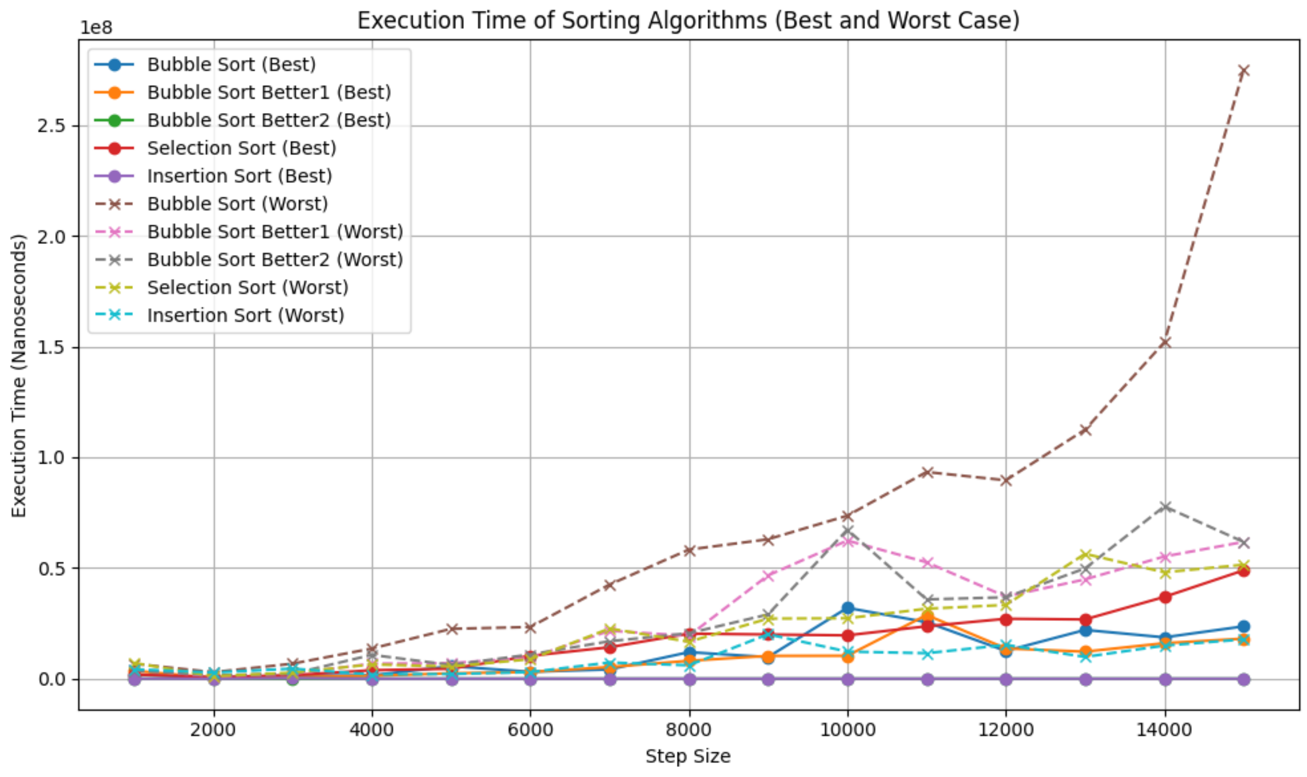
CO322

Lab01

E/20/280

Sorting Algorithms

From the implemented code of the three algorithms, which have same time complexity of O(n^2). Here I have used the data generated from each algorithm and two improved versions of the Bubble sorting algorithms, for increasing number of input sizes with the total execution time, to plot the below graph.



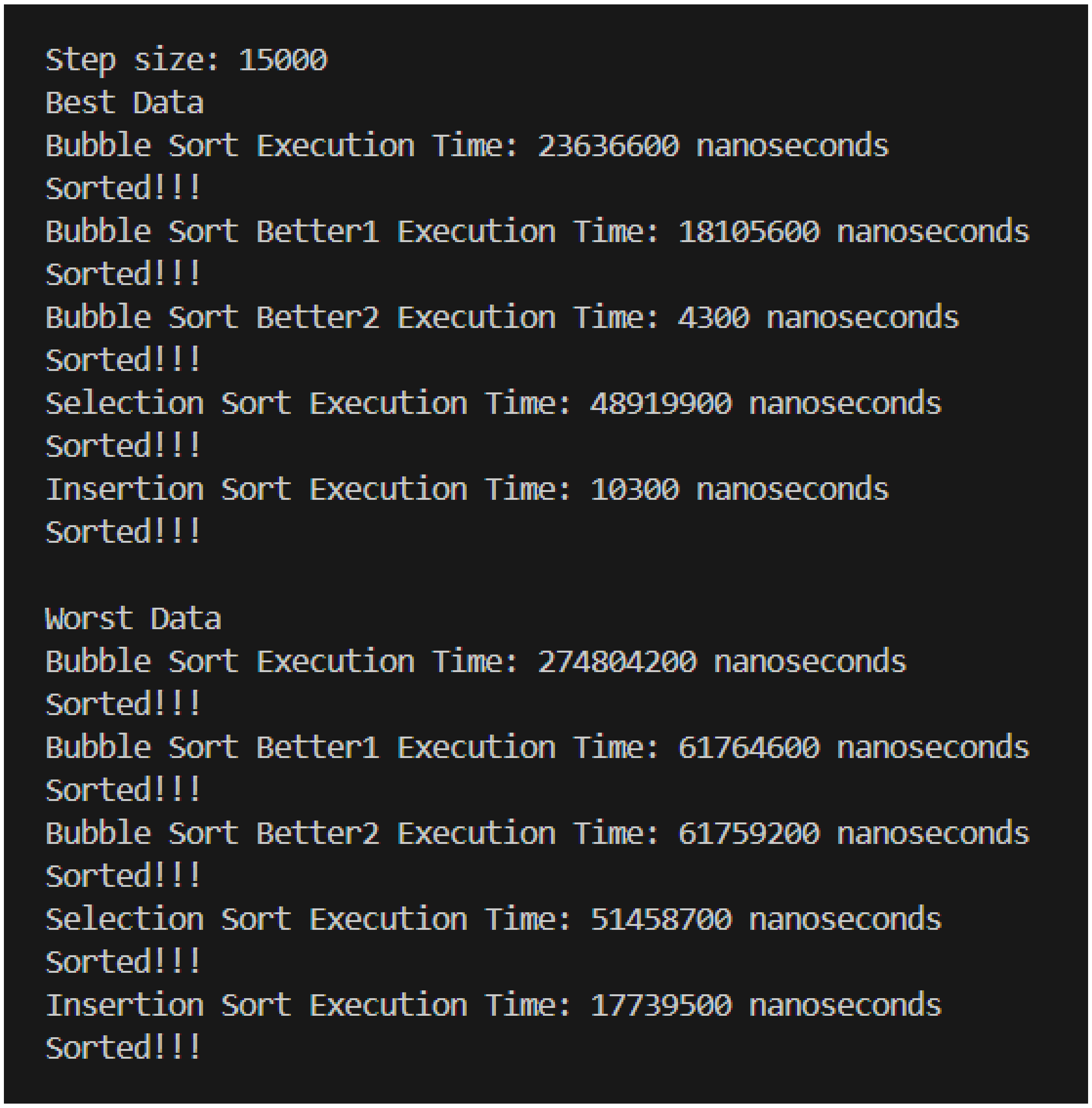
<u>Figure 1</u>

Here we can observe that as the Input size grows, the variation between the methods are very visible.

As visible for all initial implementation, the Insertion sort has performed the best, against worst data and the best data.

Therefore, the overall shape of each of the graphs for initial implementations does match with the theoretical suggestions for the execution time.

to test each implementation, I have used the given function to check of the returned array is in fact sorted or not.

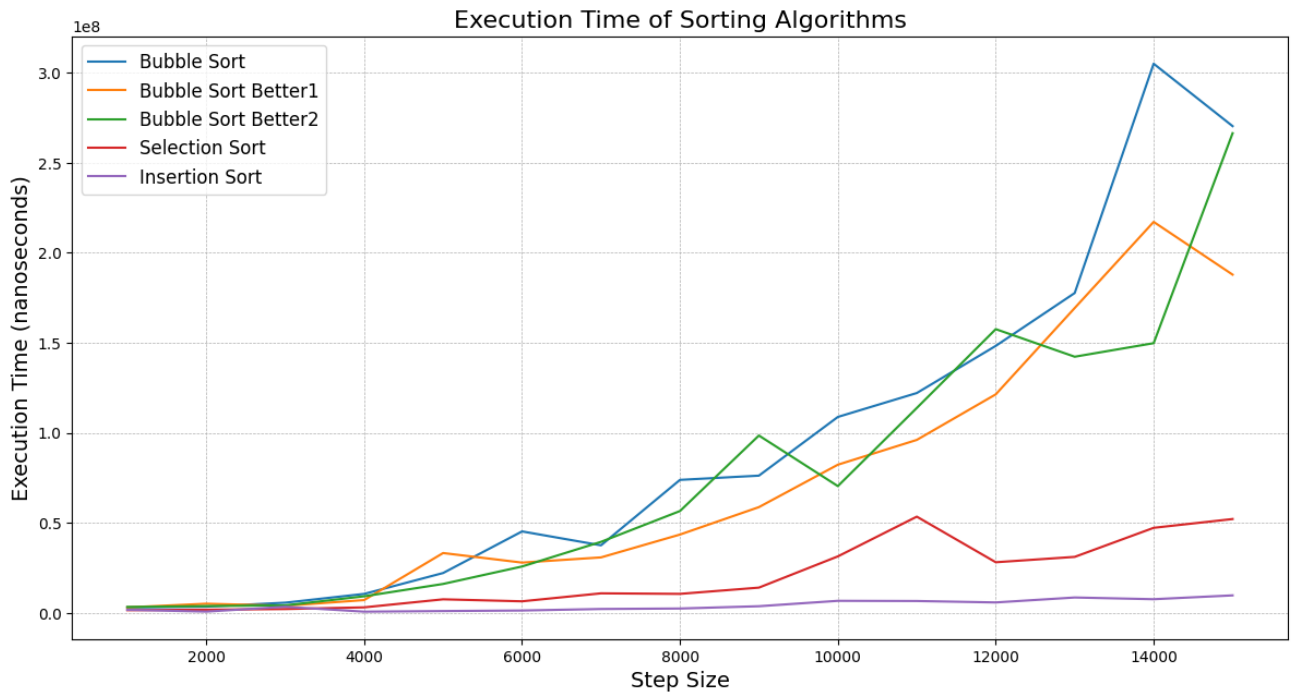


```
Step size: 15000
Best Data
Bubble Sort Execution Time: 23636600 nanoseconds
Sorted!!!
Bubble Sort Better1 Execution Time: 18105600 nanoseconds
Sorted!!!
Bubble Sort Better2 Execution Time: 4300 nanoseconds
Sorted!!!
Selection Sort Execution Time: 48919900 nanoseconds
Sorted!!!
Insertion Sort Execution Time: 10300 nanoseconds
Sorted!!!

Worst Data
Bubble Sort Execution Time: 274804200 nanoseconds
Sorted!!!
Bubble Sort Better1 Execution Time: 61764600 nanoseconds
Sorted!!!
Bubble Sort Better2 Execution Time: 61759200 nanoseconds
Sorted!!!
Selection Sort Execution Time: 51458700 nanoseconds
Sorted!!!
Insertion Sort Execution Time: 17739500 nanoseconds
Sorted!!!
```

Figure 2

However, in order to truly analyze each implementation it is better to use random data(Average), inserted of the Best data or Worst data.



<u>Figure 3</u>

In the figure 1 it seems that the 2nd improved version of bubble sort performed same as the Insertion (only in best case)

How ever here we can clearly see with random data, the improved versions performed slightly better than initial implementation. But not close to selection sort or Insertion sort.

Therefore, we can conclude that the Insertion Sort is the most efficient.