

Name: Bishal Sarangkoti
CE: III
Roll: 45

- <https://github.com/sarangbishal/COMP-314-Algorithms-and-Complexity-Labworks/tree/master/Lab-2%20Sorting>

Q1.

- [insertion_sort.py](#)
- [merge_sort.py](#)

Q2.

- [tester.py](#)
- [stress_test.py](#)

Q3.
[plotter.py](#)

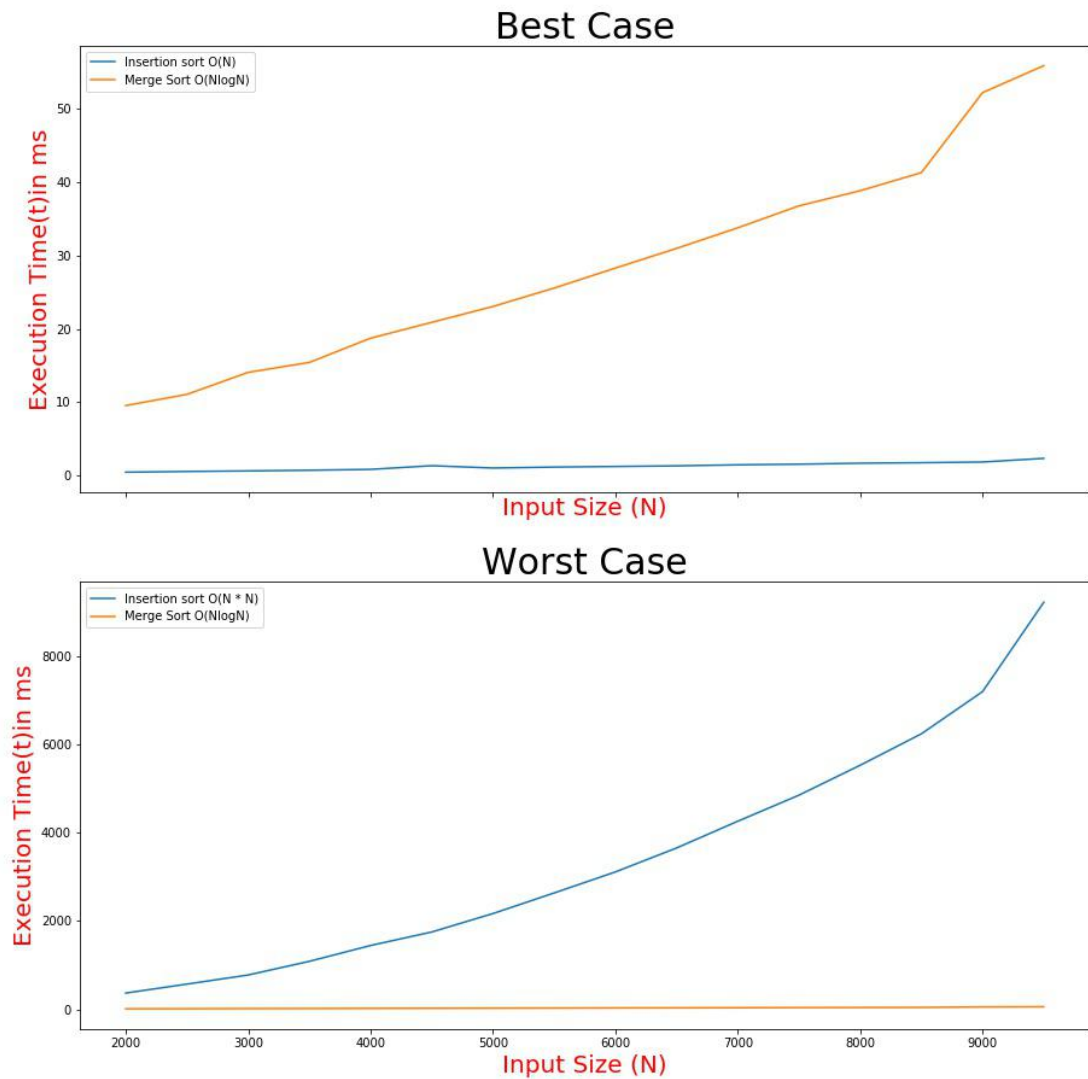


Figure 1 Input Size vs Execution Time Graphs

Q4.
From the graphs of input size vs execution time, we have following observations:

A. Insertion Sort:

a) Best Case:

From graph(a) of figure 1, we can see that the execution time for insertion sort grows less rapidly compared to that of merge sort.

The best case for insertion sort is when the input array is already sorted. For a sorted array of size N , the bestcase time complexity for insertion sort is order of linear to input size i.e $O(N)$.

b) Worst Case:

From graph(b) of figure1, we can see that the execution time for insertion sort grows rapidly compared to merge sort.

*The worst case for insertion sort is when the input array is sorted in reverse order. For a reversed sorted array of size N , the worstcase complexity for insertion sort is order of quadratic to input size i.e $O(N * N)$.*

B. Merge Sort:

a) Best Case:

From graph(a) of figure1, we can see that the execution time for merge sort grows more rapidly compared to insertion sort. ***It has time complexity of $O(N\log N)$ for bestcase as well as worst case.***

b) Worst Case:

From graph(b) of figure1, we can see that the execution time for merge sort grows less rapidly compared to insertion sort. ***In worst case also, it has time complexity of $O(N\log N)$.***