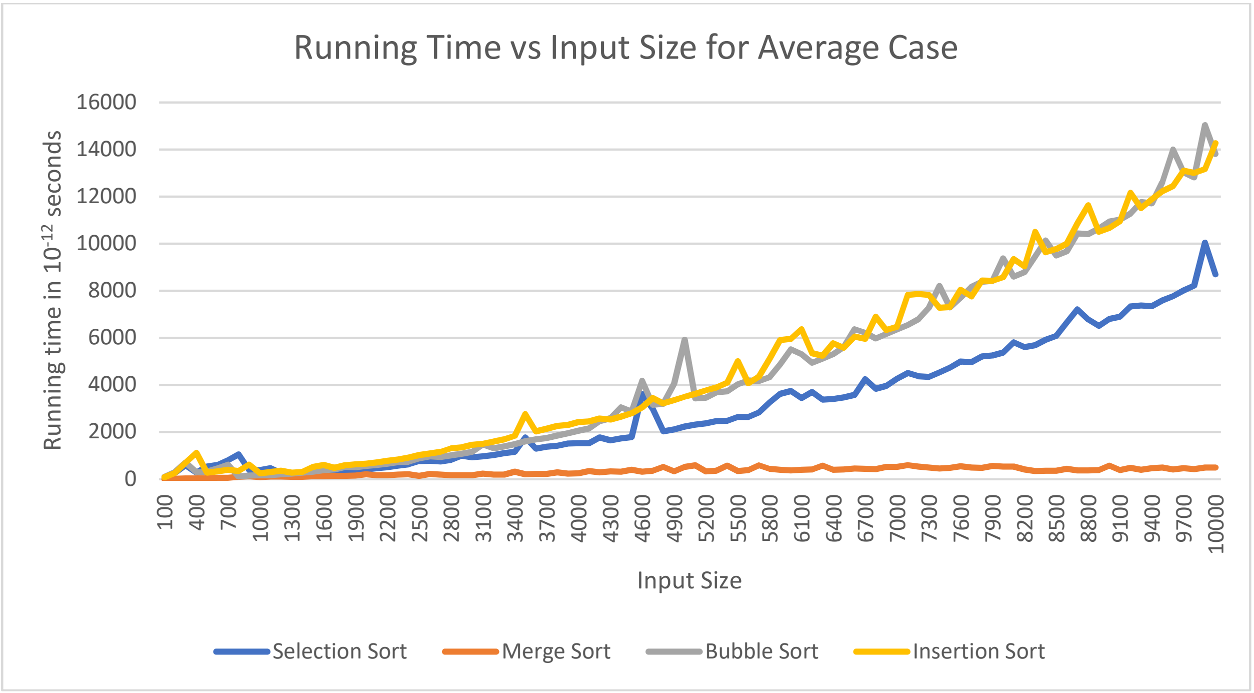
Assignment: Analysis of Sort Algorithms

Below are the formulas for time complexities of all algorithms.

Algorithm	Time Complexity		
	Best	Average	Worst
Selection Sort	Ω(n^2)	θ(n^2)	O(n^2)
Bubble Sort	Ω(n^2)	θ(n^2)	O(n^2)
Bubble Sort (optimized algorithm)	Ω(n)	θ(n^2)	O(n^2)
Insertion Sort	Ω(n)	θ(n^2)	O(n^2)
Merge Sort	Ω(n log(n))	θ(n log(n))	O(n log(n))

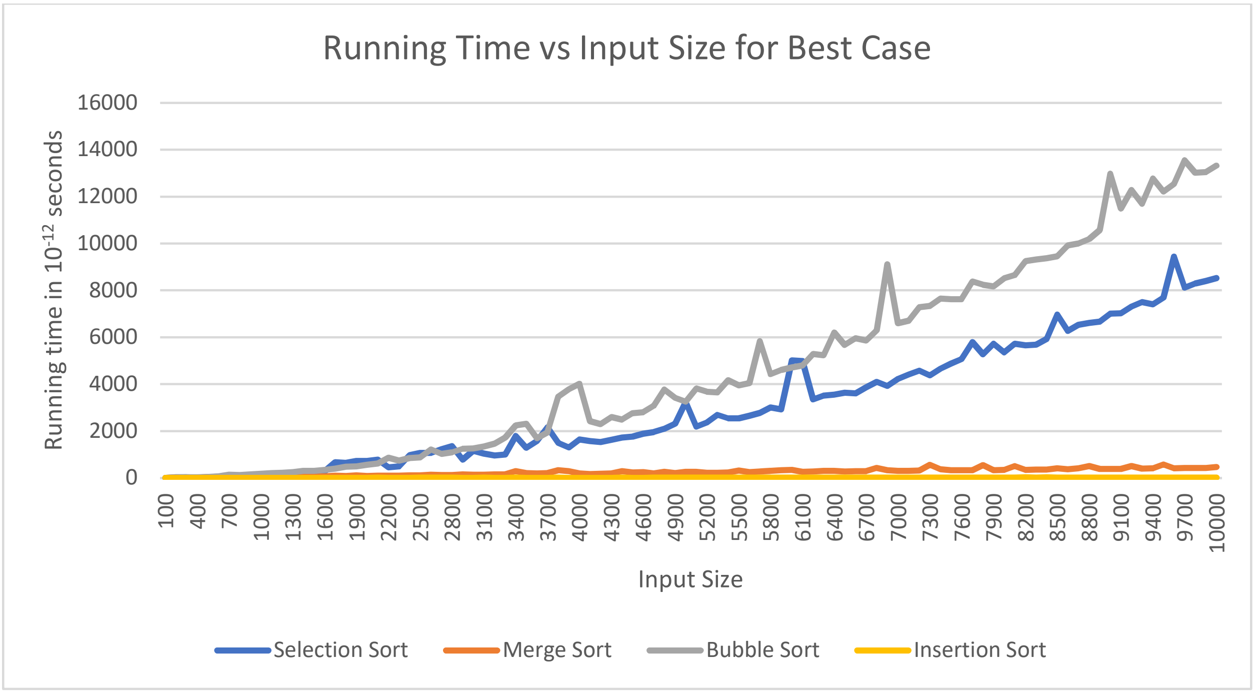
The analysis of the graphs is as follows:

Below graph depicts the time complexities of selection sort, merge sort, insertion sort and bubble sort for the **average case**.



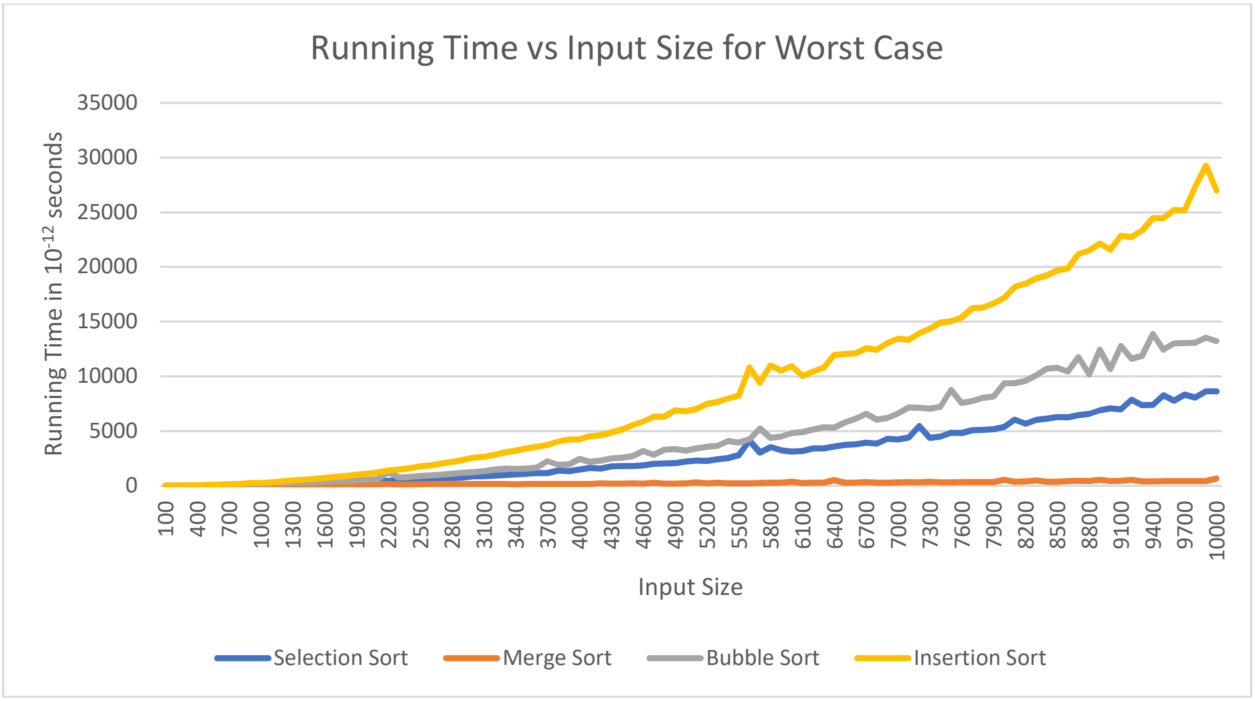
1. From the above graph, it can be noticed that merge sort has the lowest time complexity when compared to the other three sorting algorithms, that is, selection sort, bubble sort and insertion sort. Therefore, Merge sort is the best way of sorting when we have data like average case.
2. Also, it can be noticed that when the input size is small, the time taken to execute by the algorithms is almost the same.
3. As soon as the input size increases, the time taken by sorting algorithms to execute varies.

Below graph depicts the time complexities of selection sort, merge sort, insertion sort and bubble sort for the **best case**.



1. I have written bubble sort code in a non-optimized manner. So, time complexity is not reduced for best case as compared to average case. If we optimize the code for bubble sort, then for the best case its time complexity would have been somewhat like that of insertion sort, that is, the least.
2. From the above graph, it can be noticed that bubble and selection sort have the highest time complexity when compared to the other two sorting algorithms, that is, insertion sort and merge sort.
3. Merge sort has higher time complexity than that of insertion sort. So, insertion sort is the most efficient in best case.
4. Also, it can be noticed that when the input size is small, the time taken to execute by the algorithms is almost the same.
5. As soon as the input size increases, the time taken by sorting algorithms to execute varies.

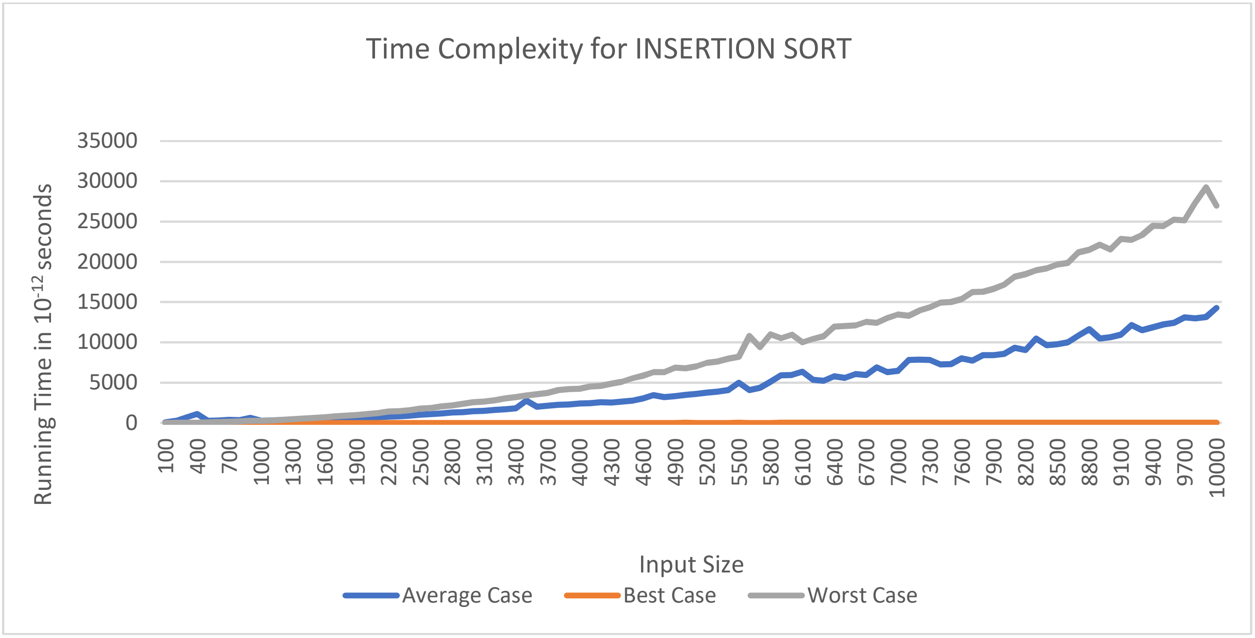
Below graph depicts the time complexities of selection sort, merge sort, insertion sort and bubble sort for the **worst case**.



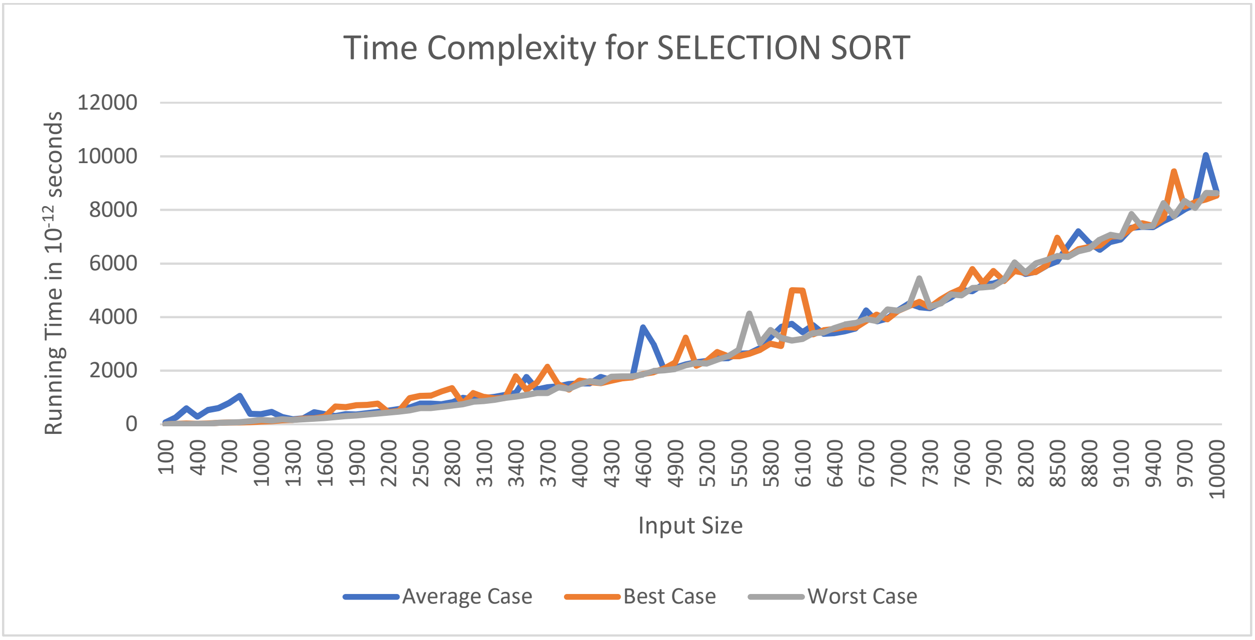
1. From the above graph, it can be noticed that merge sort has the lowest time complexity when compared to the other three sorting algorithms, that is, selection sort, bubble sort and insertion sort. Therefore, Merge sort is the best way of sorting when we have data like worst case.
2. Also, it can be noticed that when the input size is small, the time taken to execute by the algorithms is almost the same.
3. As soon as the input size increases, the time taken by sorting algorithms to execute varies.

Below are the graphs of time complexity of different sorting algorithms, explaining their best, average and worst cases.

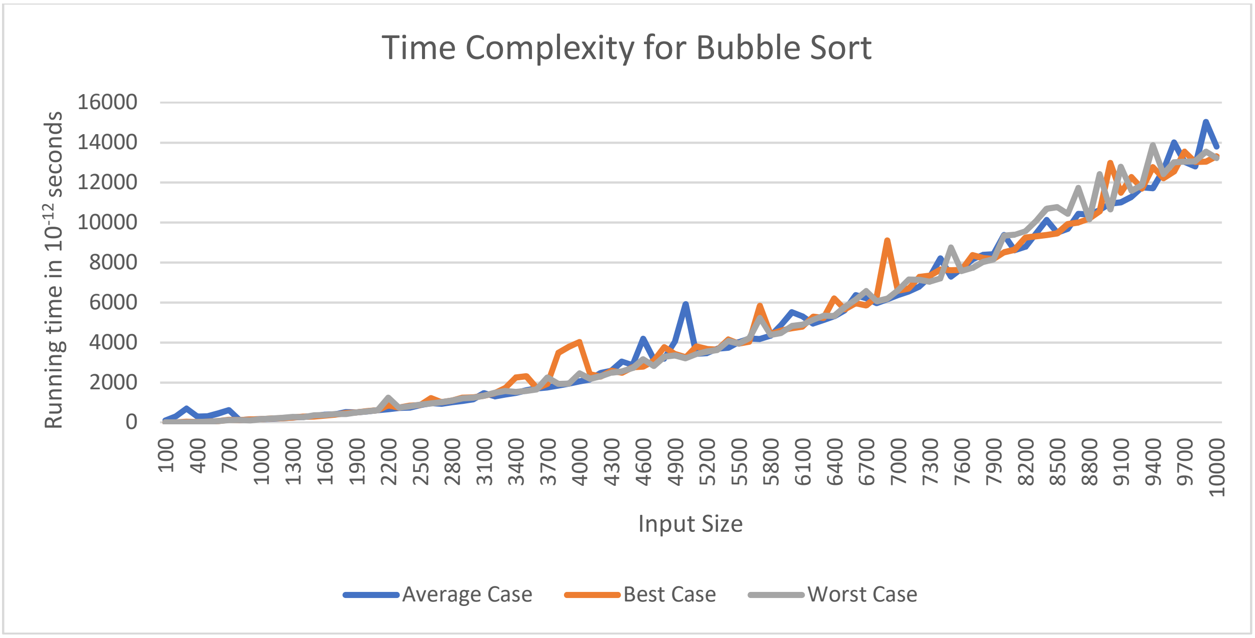
1. INSERTION SORT



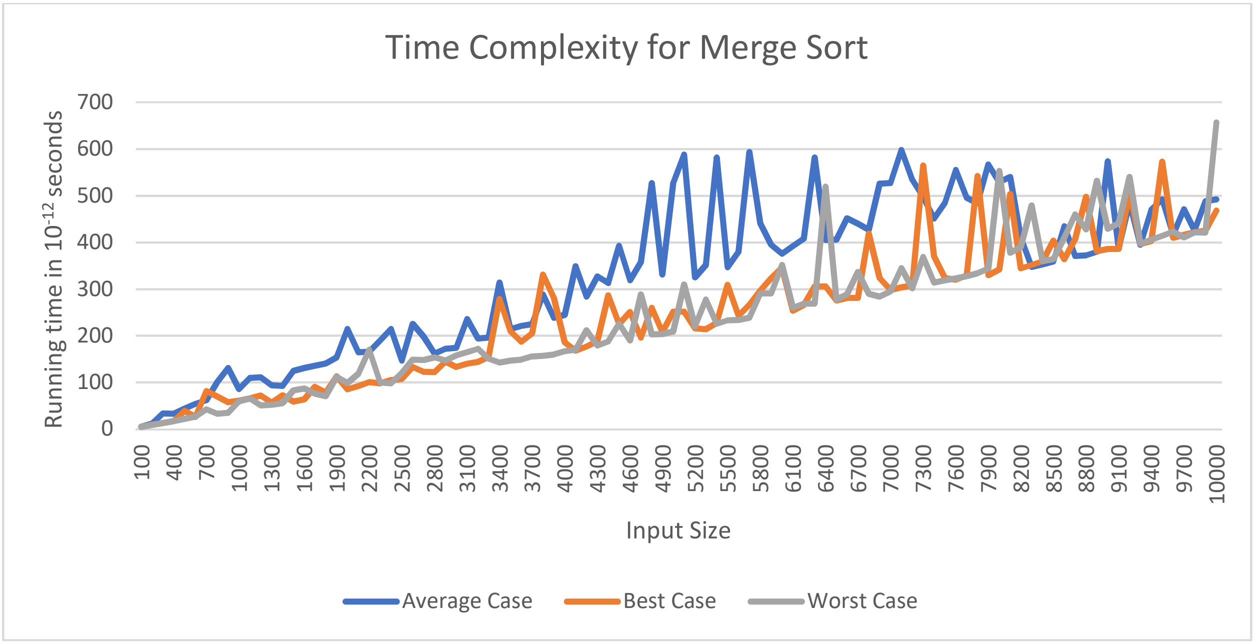
2. SELECTION SORT



3. BUBBLE SORT



4. MERGE SORT



From the graphs depicting time complexities of selection sort, bubble sort, insertion sort and merge sort, the below observations are made:

1. Time complexity of selection sort is similar in all the three cases namely best, average and worst case.

2. Time complexity of merge sort is similar in all the three cases namely best, average and worst case.
3. Time complexity of insertion sort in worst and average case is way more than that of best case.
4. Time complexity of bubble sort is similar in all the three cases namely best, average and worst case. (non-optimized bubble sort)

Therefore, from the above observations, the most optimized algorithm in average case and worst case is **Merge Sort** and the most optimized algorithm in best case is **Insertion Sort**.

--END OF ASSIGNMENT--