# Analysis Time Complexity of Insertion Sort and Selection Sort

## Objective:

This report is based on the analysis the time complexity of two different algorithm named ‘Insertion sort and Selection sort’. Though these two algorithms are used for making a list sorted but according to their runtime and case analysis these two algorithms take different time to sort a list which depends on the size of the list. So, the main objective is to identify the time taken by two different algorithm and measure their complexity according to the runtime.

## Machine Configuration:

| | |
|---|---|
| Windows edition | : Windows 10 Pro |
| Processor | : Inter® Core™ i5-7200U CPU @ 2.50GHz |
| Installed memory (RAM): | 8.00 GB (7.88 GB usable) |
| System Type | : 64-bit operation System, x64-based processor |

## Insertion Sort:

Theoretically, insertion sort runs in O(n) in best case and runs in O(n^2) in both average and worst case.

Insertion sort make a list sorted by scanning the list and swapping the element if they are not sorted. These two operations contribute in runtime of the algorithm. In best case this algorithm runs in O(n) as it does not perform any operation in inner loop like swap and every time not checking the whole list from sorted portion. But in worst and average case, it takes much time relative of O(n^2) according to Big-Oh notation.

**Time Complexity: O(n^2) for worst and average case and O(n) for best case.**

**Space Complexity: O (1**)

## Selection Sort:

Selection sort make a sorted list by finding the minimum element in the unsorted portion of the list and putting it in the beginning of the list.

So, when the list is already sorted the algorithm doesn't do any swap but it makes iteration with the whole list every time. So, runtime for every case is same and don't depend on best, average or worst case.

**Time Complexity: O(n^2) as there are two nested loops.**

**Space Complexity: O (1)**

## Data Table for Insertion and Selection Sort:

### Insertion Sort (time taken in second):

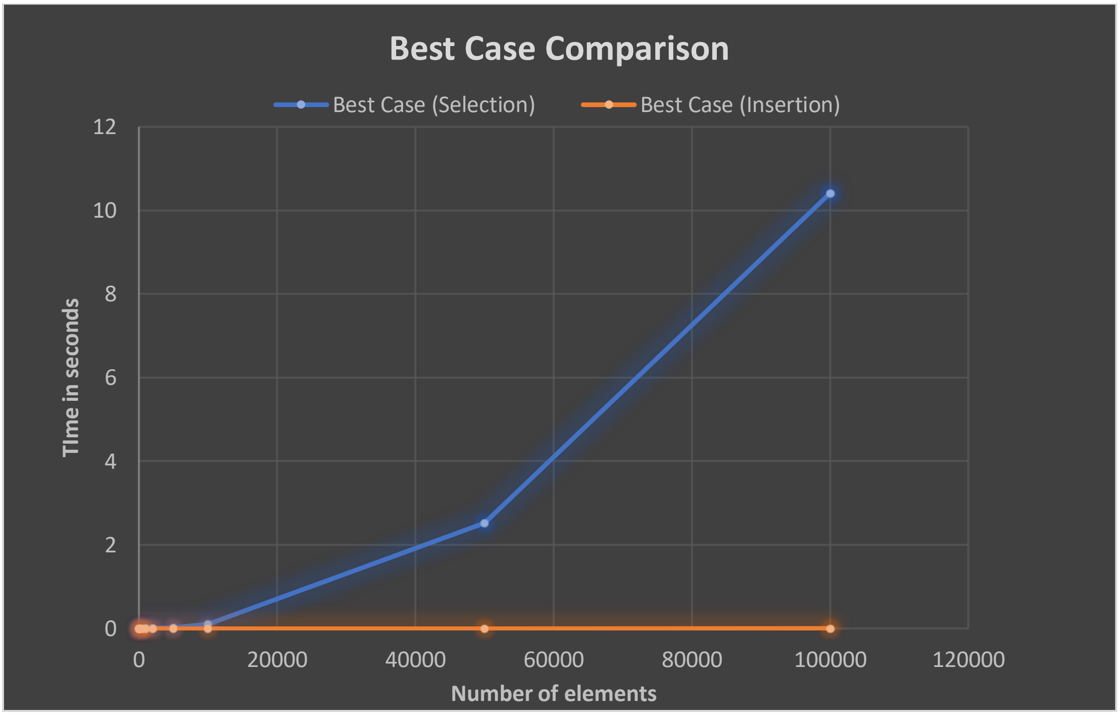
| Size | Average | Best | Worst |
|---|---|---|---|
| 10 | 0 | 0 | 0 |
| 100 | 7.89E-06 | 0 | 2.2E-05 |
| 200 | 2.22E-05 | 0.99E-06 | 3.38E-05 |
| 500 | 1.93E-04 | 1.2E-06 | 2.21E-04 |
| 1000 | 0.0006999 | 2.92E-05 | 0.0007221 |
| 2000 | 0.00245 | 5.87E-05 | 0.00248 |
| 5000 | 0.012165 | 6.9E-05 | 0.012110 |
| 10000 | 0.062219 | 2.86E-04 | 0.0701 |

Selection Sort (time taken in second):

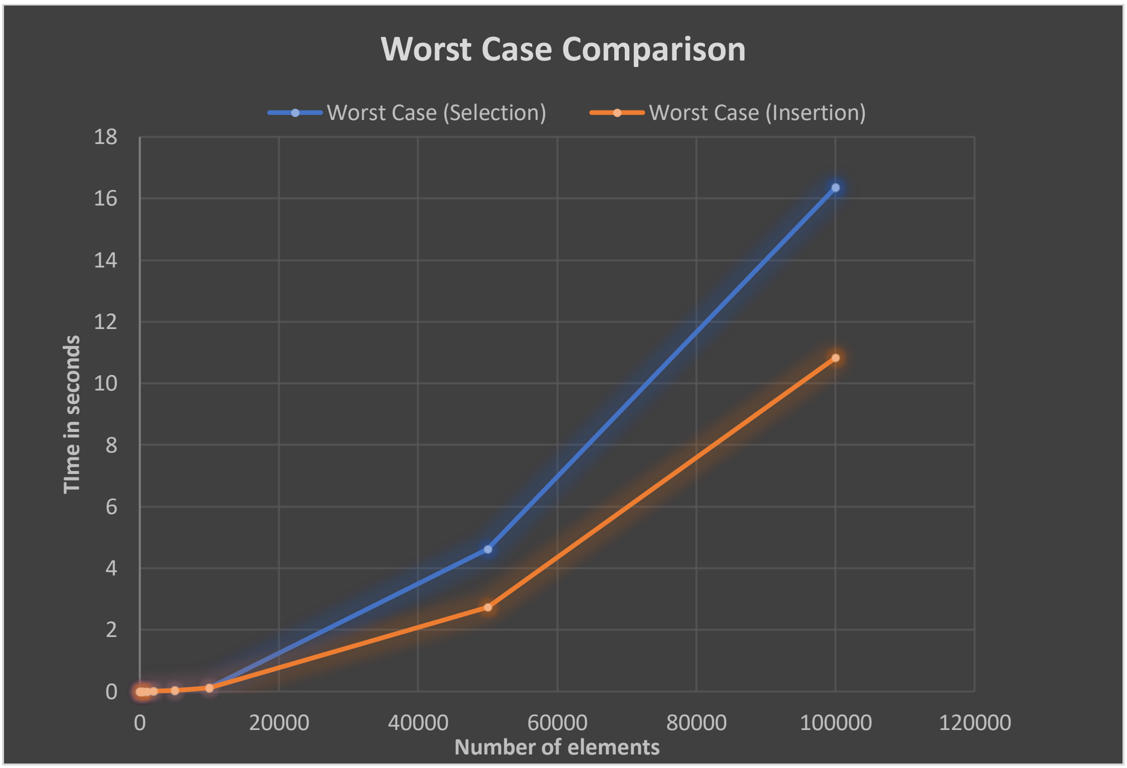
| Size | Average | Best | Worst |
|---|---|---|---|
| 10 | 0 | 0 | 0 |
| 100 | 7.0E-05 | 4.3E-05 | 7.7E-05 |
| 200 | 1.21E-04 | 1.19E-04 | 2.23E-03 |
| 500 | 3.025E-04 | 2.93E-04 | 3.1E-03 |
| 1000 | 0.605E-03 | 0.721E-03 | .0442101 |
| 2000 | 0.013563 | 0.005699 | 0.04507 |
| 5000 | 0.063001 | 0.06421 | 0.02464 |
| 10000 | 0.1210 | 0.1107 | 1.2348 |

## Graphical Representation:

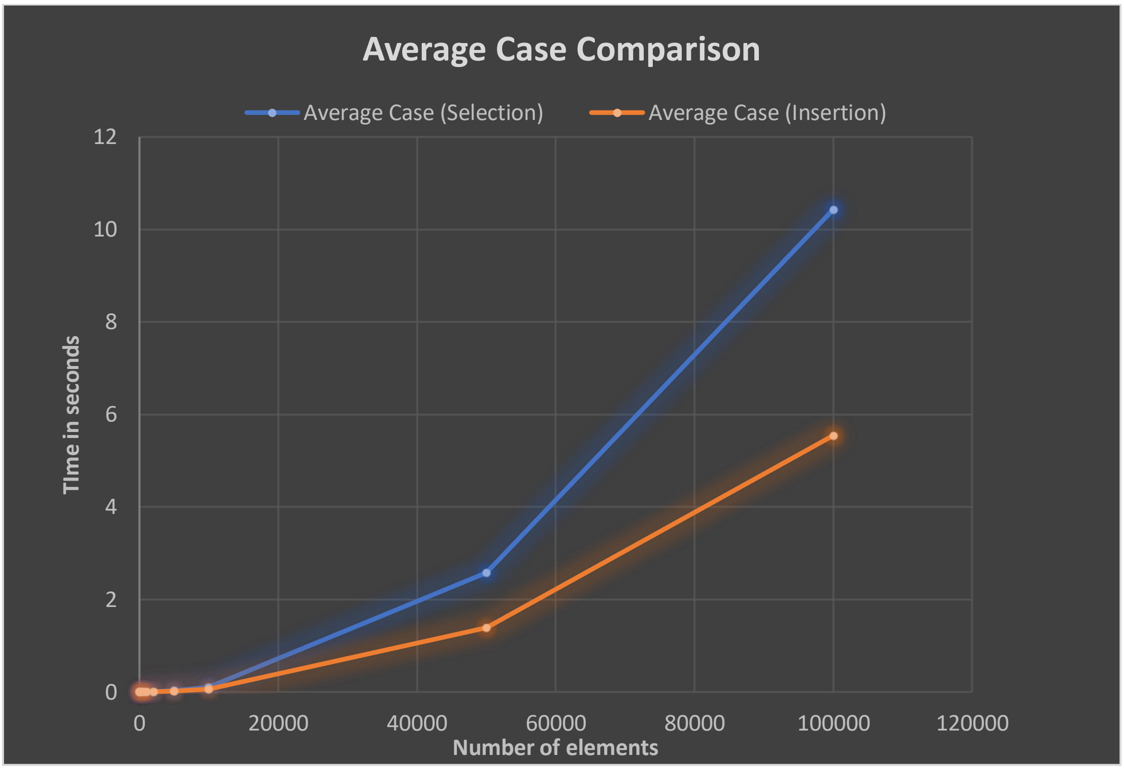
1: Time taken is Best Case:



2.Time taken in worst case:



3. Time taken in Average Case:



## Analysis:

According to the time analysis and graphical representation of the time taken by the two graph, it is clear that both the graph takes O(n^2) runtime in Big-Oh sense, but as we don't know that the list is sorted or not, so insertion sort will be better to use between the two algorithm to sort a list.

So, though both has the same complexity in big-oh sense, but insertion sort is better than selection sort.