**Analysis and Design of Algorithms**

**Semester III**, Year **2021-22**

**Lab - 1**  Date : 08-09-2021

Name: E. Sai Manoj MIS. No: 112015044 Branch: CSE

**AIM:** Implement insertion sort, selection sort bubble sort and merge sort. The number of inputs elements has to be passed from command line arguments. The elements has to be generated randomly within the code.

Compute:

a. Check the performance of program by varying the number of elements.

b. Compute the time taken by each case (for particular number of inputs) n = 10, 100, 1000, 10000, 100000.

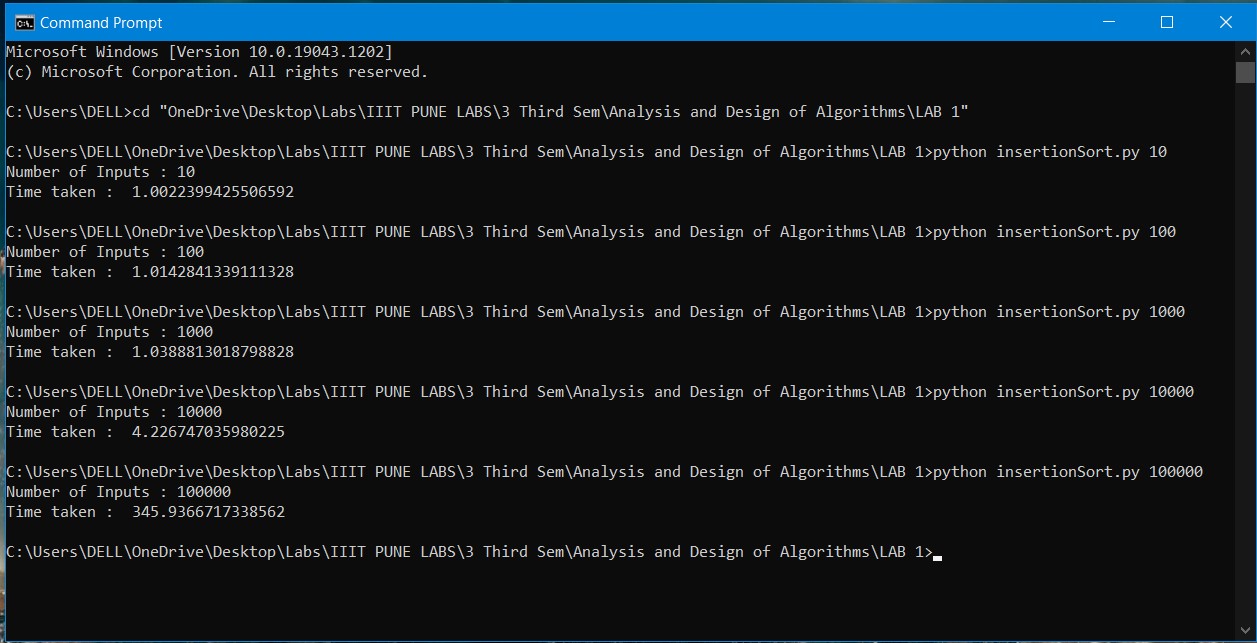
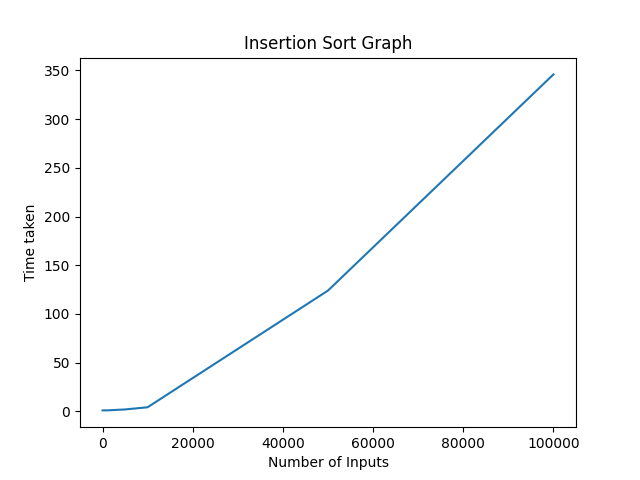
c. Plot a graph with number of inputs Vs time taken in seconds.

d. Compare the graphical plots for each sorting algorithms with its theoretical time complexity.

e. Also compute the time taken for sorted array (worst case) and compare with different number of elements.

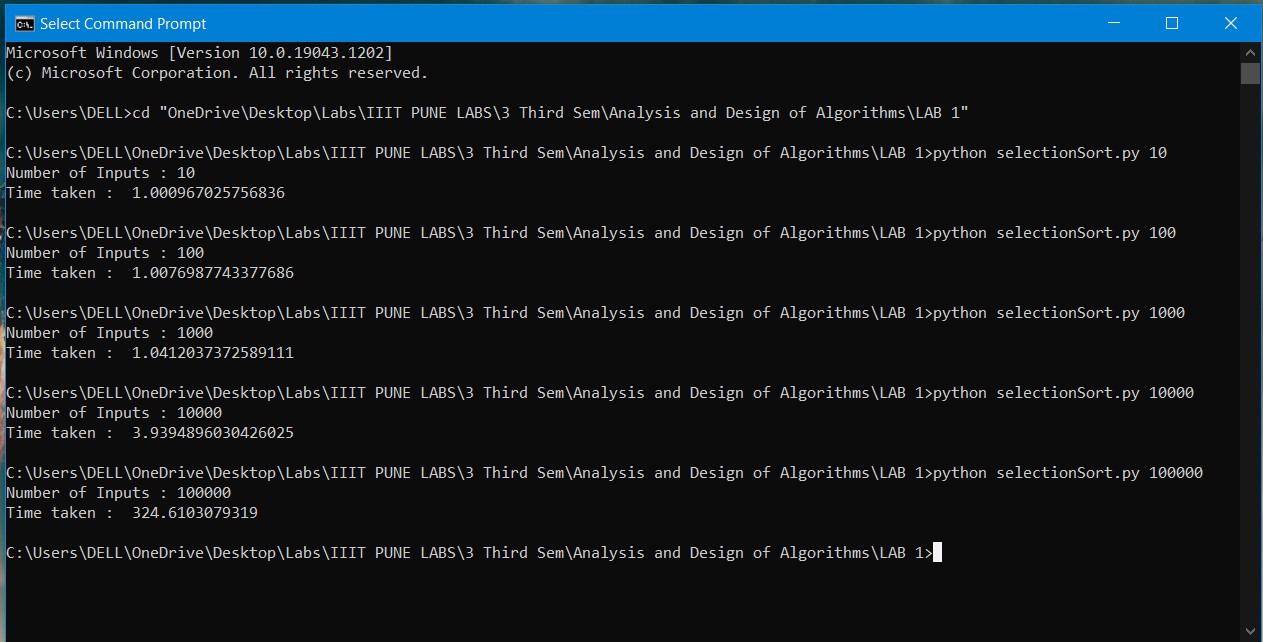
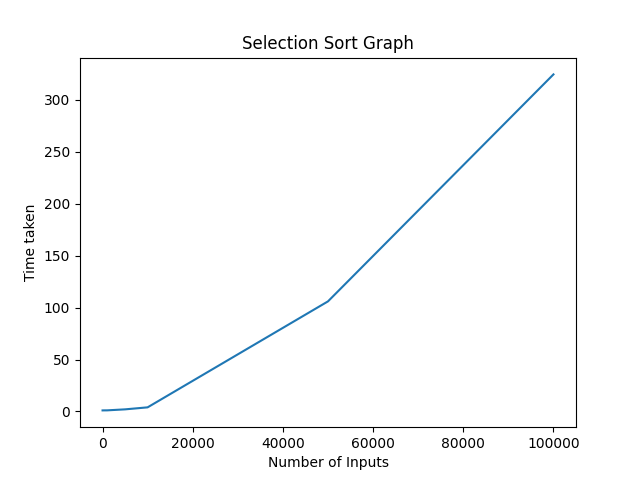
**Insertion Sort:**

Output: Graph:

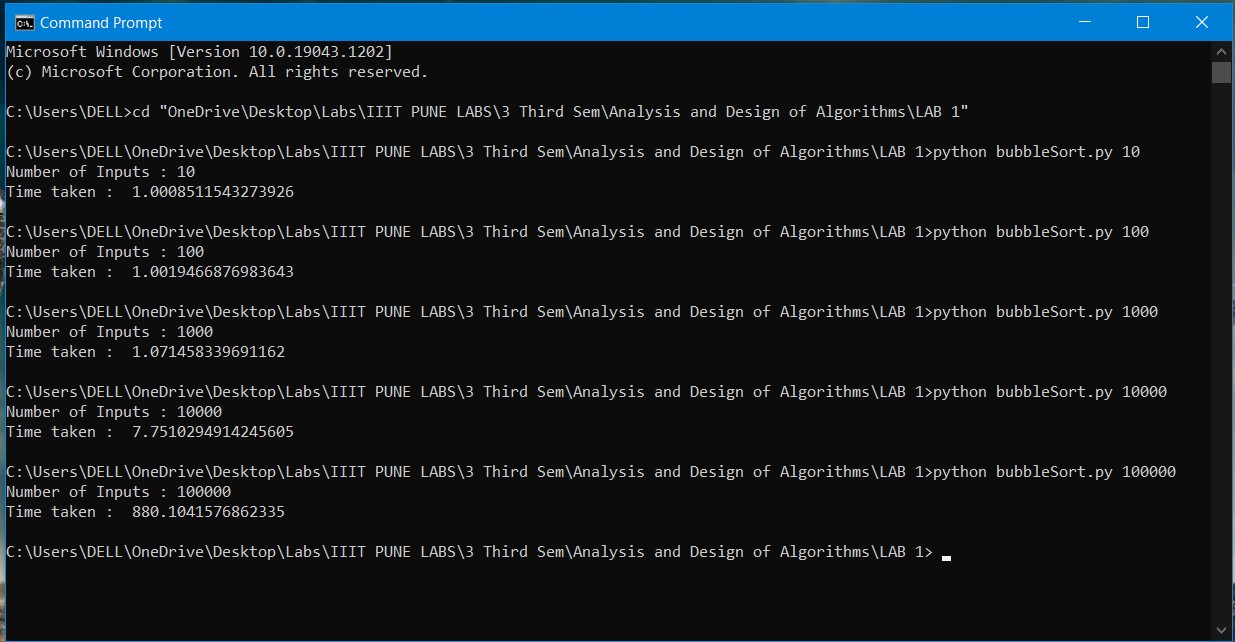
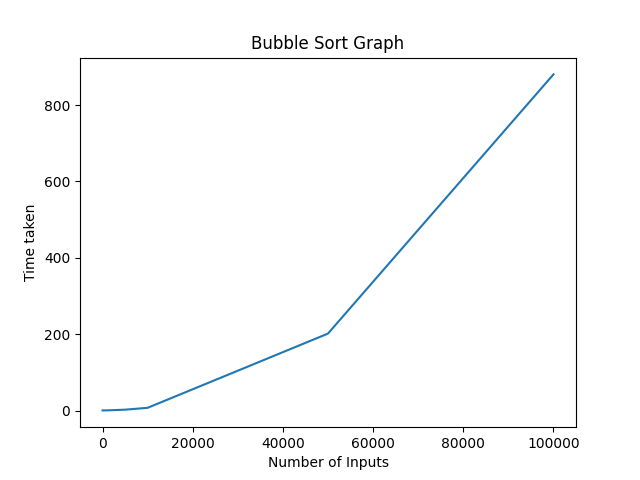
**Selection Sort:**

Output: Graph:

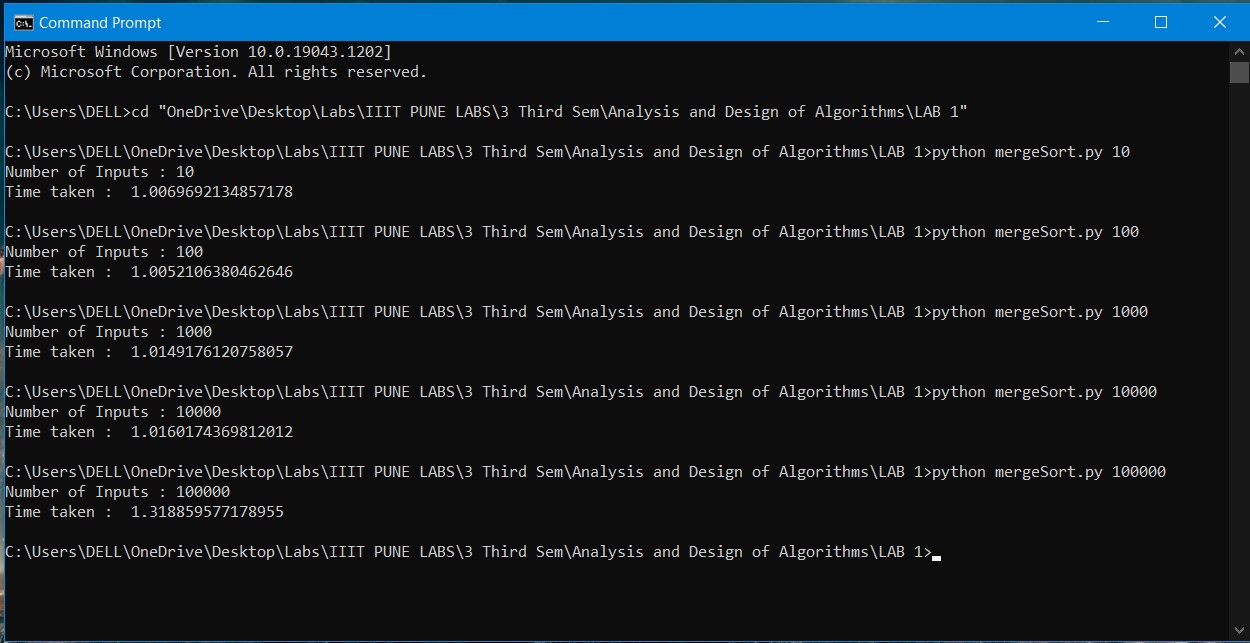
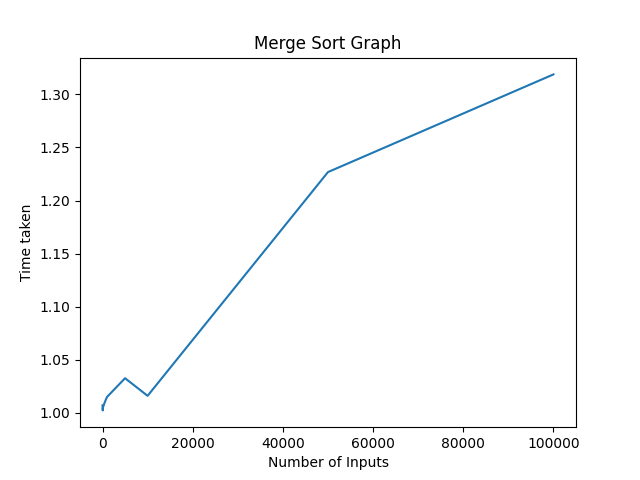
**Bubble Sort:**

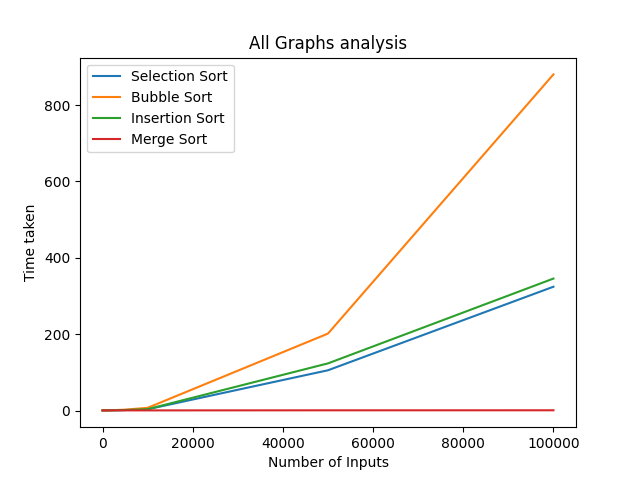
Output: Graph:

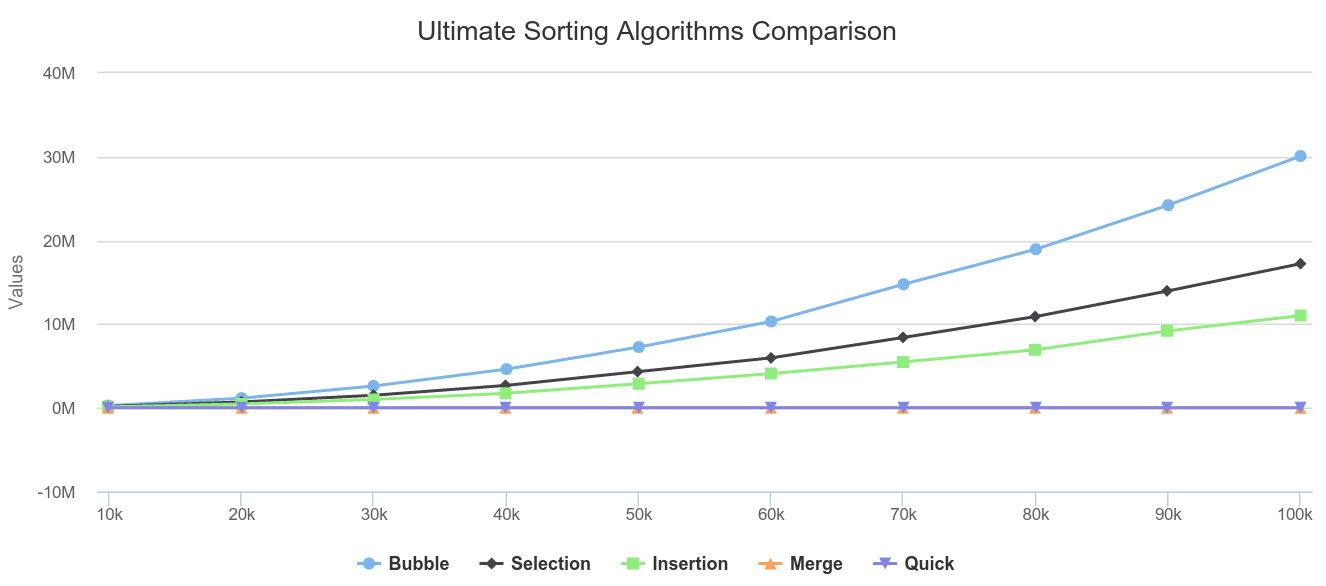
 

**Merge Sort:**

Output: Graph:

**Comparison Graph:**

 \* Graph taken from internet for reference and comparisonThe graph which I obtained is the same as the theoretical time complexities graph, bubble sort with highest growth and merge sort with least. Selection sort and Insertion sort were almost similar and would be precise if more number of inputs were taken also the graphs would have become curved to represent precise growth of algorithms.

**Result:**

Here are the results obtained showing the time taken for different sorting algorithms to run different number of inputs.

\* All times obtained were added with one second using time.sleep(1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number of Inputs  (To right) | 10 | 100 | 1000 | 10000 | 100000 |
| Insertion Sort | 1.0022399425506592 | 1.0142841339111328 | 1.0388813018798828 | 4.226747035980225 | 345.9366717338562 |
| Selection Sort | 1.000967025756836 | 1.0076987743377686 | 1.0412037372589111 | 3.9394896030426025 | 324.6103079319 |
| Bubble Sort | 1.0008511543273926 | 1.0019466876983643 | 1.071458339691162 | 7.7510294914245605 | 880.1041576862335 |
| Merge Sort | 1.0069692134857178 | 1.0052106380462646 | 1.0149176120758057 | 1.0160174369812012 | 1.318859577178955 |

Worst case time complexities**:** \* All times obtained were added with one second using time.sleep(1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number of Inputs  (To right) | 10 | 100 | 1000 | 10000 | 100000 |
| Insertion Sort | 1.0095903873443604 | 1.015418062210083 | 1.1003468036651611 | 8.794252872467041 | 759.9043262004852 |
| Selection Sort | 1.0060904026031494 | 1.0086463012695312 | 1.0807583332061768 | 4.893258333206177 | 429.09446573257446 |
| Bubble Sort | 1.0025291442871094 | 1.0156090259552002 | 1.1648099422454834 | 12.424667358398438 | 1159.5737493038177 |
| Merge Sort | 1.0050386447906494 | 1.0106956958770752 | 1.015178804397583 | 1.040414571762085 | 1.3664815425872803 |

**Conclusion:**

After analysing the results obtained, I observed that in case of small number of inputs the time taken by the sorting algorithms were as Bubble sort < Selection sort < Insertion sort < Merge Sort whereas when it comes to large number of inputs there are noteworthy changes the time taken were as Merge sort << Selection sort < Insertion sort < Bubble sort. In case of small number of inputs there is little difference between time taken by sorting algorithms. But coming to large number of inputs merge sort gave the input in very much less time compared to remaining sorting algorithms. This comparison shows how efficiency of an algorithm plays an important role in designing of algorithms.

\* All graphs were plotted taking some extra inputs for better precision.