**Analysis and Design of Algorithms**

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

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**AIM:** Analysis of sorting algorithms bubble Sort, insertion sort, selection sort, merge sort.

**Experiment:** Evaluating the performance of each sorting algorithm by varying the number of inputs, also computing time taken by particular cases, and comparing their graphs with theoretical time complexity.

**Problem 1**: Check the performance of program by varying the number of elements.

**Output:**

* Time performance: For a randomly generated list in python execution time to perform sorting is in most cases are as follow:

Bubble sort > Insertion sort > Selection sort > Merge sort

* Space performance: Bubble sort, Insertion sort, and selection sort have the same space performance though the number of comparisons in each sort is different. While merge sort takes more space as compared to the other three sorts.

**Problem 2:** Compute the time taken by each case (for particular number of inputs) n = 10,

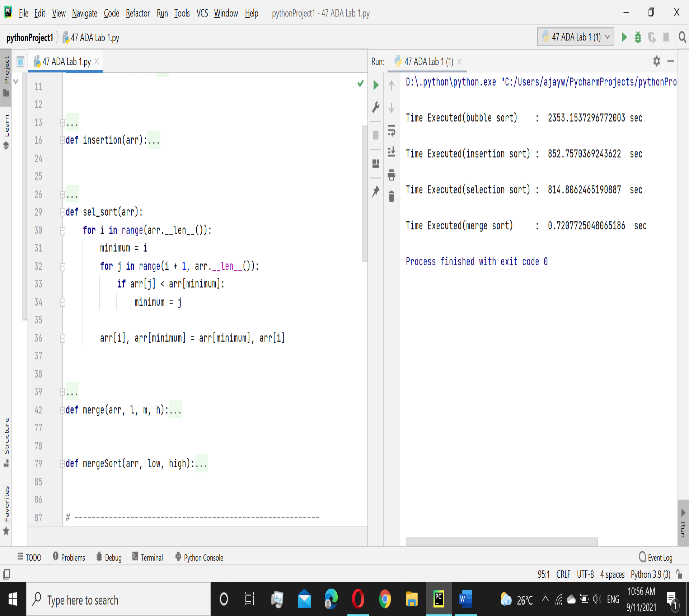
100, 1000, 10000, 1000000.

**Output:**

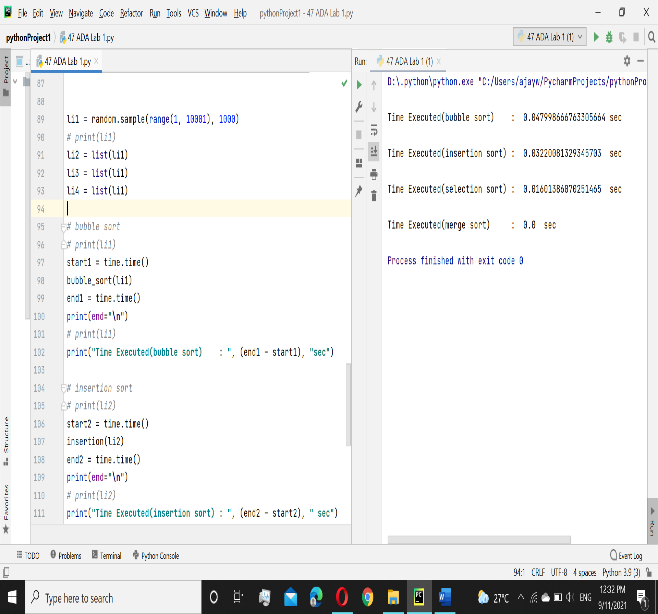
Belowtable represent the time taken for a particular case run in python -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Bubble sort | Insertion sort | Selection sort | Merge sort |
| 10 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1000 | 0.0479 | 0.0322 | 0.0160 | 0.00 |
| 5000 | 1.3055 | 0.5640 | 0.4999 | 0.0156 |
| 10000 | 12.489 | 9.0011 | 7.7221 | 0.1255 |
| 30000 | 171.50 | 82.829 | 71.342 | 0.4230 |
| 50000 | 520.81 | 227.65 | 210.35 | 0.7233 |
| 100000 | 2353.1 | 852.75 | 814.80 | 0.7207 |

Time taken for n = 100000:



Time taken for n = 1000:



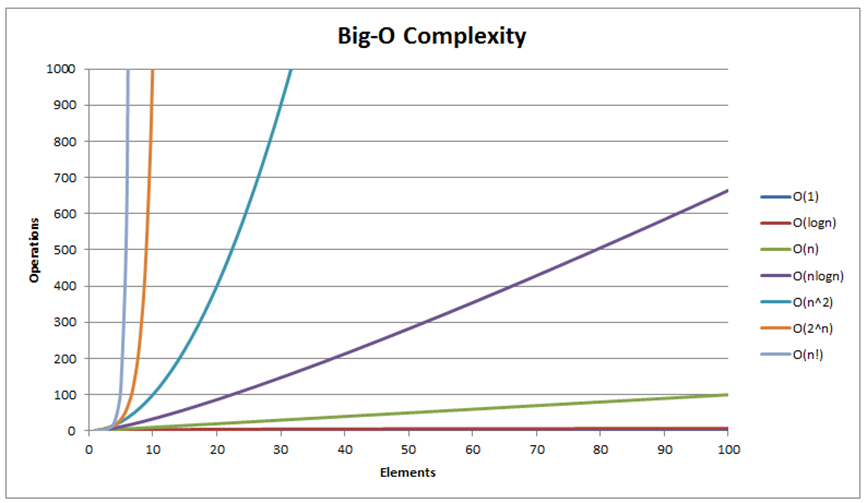
**Problem 3:** Plot a graph with number of inputs Vs time taken in seconds.

**Output:**

**Problem 4:** Compare the graphical plots for each sorting algorithms with its theoretical time

complexity.

**Output:**



Comparing with the theoretical time

complexity bubble, insertion, selection

sort have the complexity of (n^2) as in

graphical representation’s curve is higher to

the theoretical.

**Problem 5:** Compute the time taken for sorted array (worst case) and compare with

different number of elements.

**Output:**

Worst case involves when an array is sorted or in reverse ordered. Execution time for the sorted array is lesser as compared to reverse order. Insertion sort is best when an array is sorted.

**Result:**

* Time Complexity:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Bubble Sort | Insertion Sort | Selection sort | Merge sort |
| Worst-case | O(n^2) | O(n^2) | O(n^2) | O(nlog(n)) |
| Average-case | θ(n^2) | θ(n^2) | θ(n^2) | θ(nlog(n))) |
| Best-case | Ω(n) | Ω(n) | Ω(n^2) | Ω(nlog(n))) |

* Space Complexity:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Bubble sort | Insertion sort | Selection sort | Merge sort |
| Space complexity | O(1) | O(1) | O(1) | O(n) |

* Bubble Sort is the slowest the worst performer of all the algorithms. While it is useful as an introduction to sorting and algorithms, it's not fit for practical use. While merge sort works effectively in all conditions and is the faster performer. As Insertion Sort performs much fewer comparisons than Selection Sort, the implementations are usually quicker but, in these runs, Selection Sort is slightly faster.

**Conclusion:**

* When input order is not known merge sort is preferable as it has complexity nlogn and it is more stable as compared to bubble, insertion, and selection.
* When an array is sorted insertion sort is more effective.
* Merge Sort is efficient in terms of time and Insertion Sort is efficient in terms of space.