**Project - 1 Report**

**UTA ID:**

1002144180: Pranavakesava Reddy Madduri

1002175121: Lokeshwar Kodipunjula

**Time Complexity of each of the Algorithm:**

Insertion Sort:

In the Best case scenario i.e., when the array is already sorted the time complexity of the algorithm would be Ω (n). The worst case scenario would be when the array is in descending order and to be sorted in the ascending order or vice-versa the time complexity of the algorithm would be O (n2) and for the average case where all permutations are equally likely the time complexity would be Θ (n2).

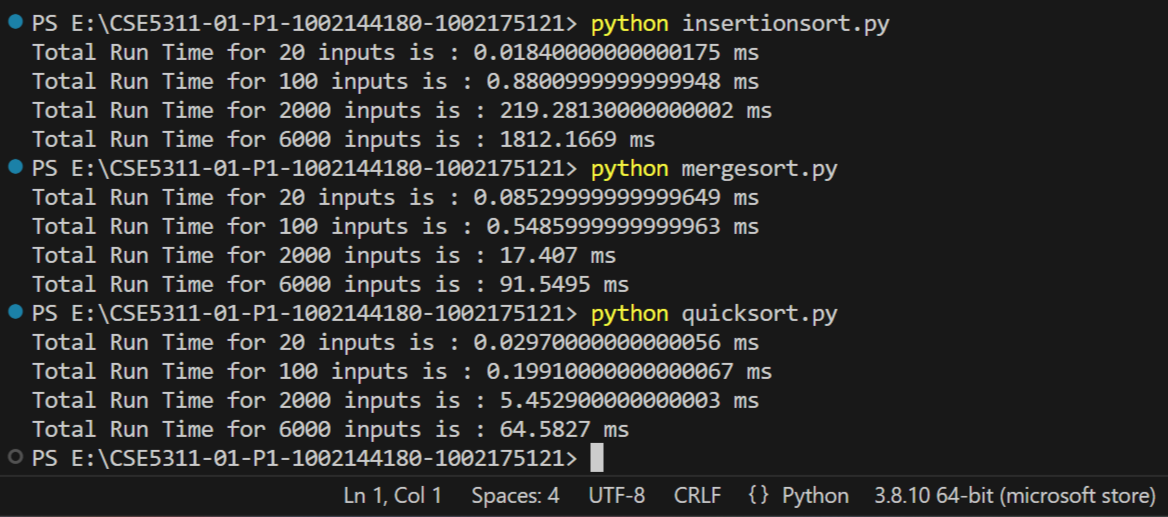
Merge Sort:

In the best, average and worst cases the Merge sort algorithm exhibits a time complexity of O (n log n).

Quick Sort:

The time complexity of the algorithm would generally be O (n log n) but in a case where pivot is the smallest element at every instance, array is to be sorted in ascending order or vice - versa the time complexity of the algorithm would be O (n2) (Worst-case).

**Experimental Results:**



Differences between **Experimental and Theoretical results**:

**Environmental factors on the Algorithm:**

Theoretical: In theory, we assume one processor, Random – access machine model by simplifying assumption to use 1 unit cost to all the operations, data movement and control. This eliminates the dependence on the speed of our computer.

Experimental: In real world scenario, several factors affect the order of growth of an algorithm such as the usage of data structures, compiler and the processor.

**Input Size:**

Theoretical: To find time and space complexity for larger inputs we use a tail behaviour which is asymptotic bounding.

Experimental: Sorting algorithms may behave in a different way with respect to the input size.

**Comparison of results between three sorting algorithms and Time taken to sort the 4 arrays:**

1. The experimental results for Insertion merge and quick sorting algorithms show that with the increase in input size, the total runtime of the algorithm increases.
2. Although merge sort and quick sort exceeds in performance for larger input sizes compared to Insertion sort, the results show that with input size 20 Insertion sort performs better than the merge and quick sort.

This gives the analysis of an algorithm whose running time has a higher order of growth might take less time for small inputs.

1. The quick sort algorithm sorts an array in much lesser run time for the larger input size compared to merge sort and merge sort sorts at faster rate than the Insertion sort

**Anomaly:** In practical for input size greater than 30 merge sorts beats Insertion sort but at times the experimental result show that for an input size of 100, sorting through insertion sort takes lesser run time than merge sort. This result might happen when Insertion sort skips few of the sorted values in randomly generated array.

**References:**

1. Lecture PPTS
2. [https://developer.nvidia.com/blog/insertion-sort-explained-a-data-scientists-algorithm- guide/#:~:text=The%20worst%2Dcase%20(and%20average,O(n)%20time%20complexity](https://developer.nvidia.com/blog/insertion-sort-explained-a-data-scientists-algorithm-%20%20%20guide/#:~:text=The%20worst%2Dcase%20(and%20average,O(n)%20time%20complexity).
3. <https://docs.python.org/3/library/timeit.html>
4. https://www.geeksforgeeks.org/file-handling-python/