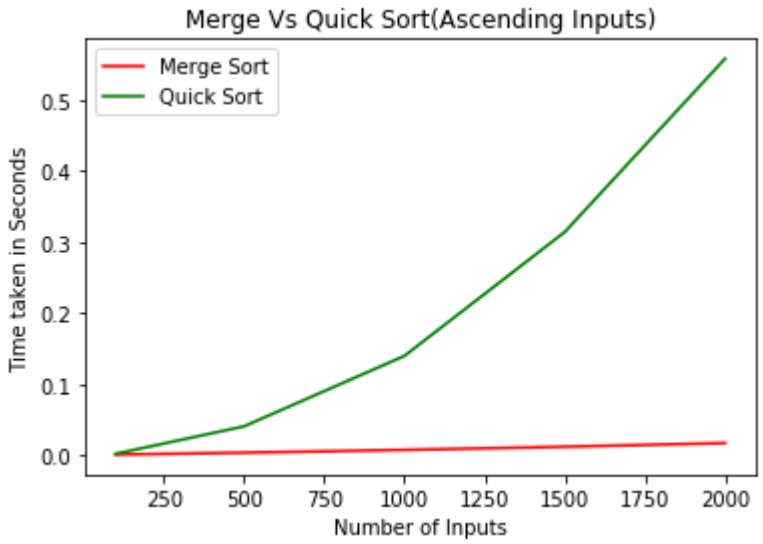
CSE-5311-001 Programming Assignment 1

Avinash Shanker

UTA ID: 1001668570

Time Taken By the algorithm to sort 100, 500, 1000, 1500 & 2000 integers in both random and ascending sorted format

1. **Merge Vs Quick Sort (Ascending Sorted) Graph 1**

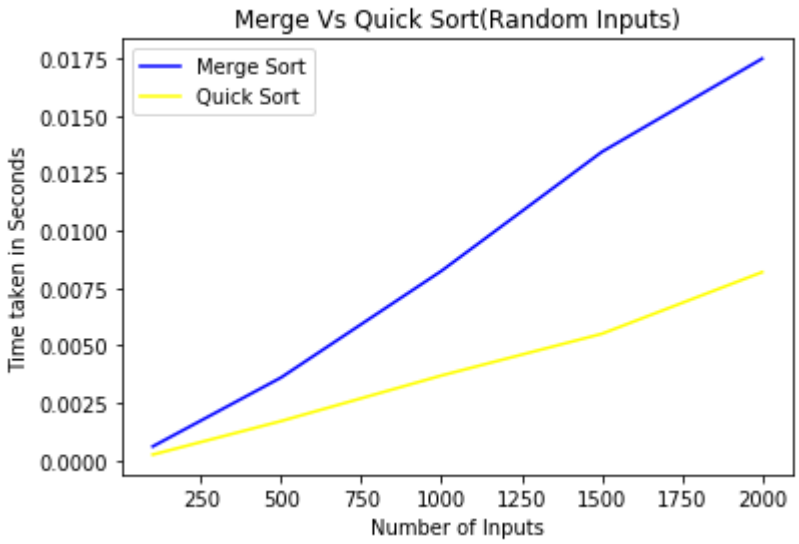


1. Merge sort performs better in this case
2. Quick sort takes much more time than merge sort
3. Since last element is selected as Pivot, Quick sort has its worst case in the scenario
4. Quick Sort portrays **$O(n^2)$** complexity in its worst case as depicted in the graph
5. Merge sort is showing much faster computation
6. Merge sort displays O**(n log n)** complexity while verifying the theoretical formula as it a very slowly increasing linear graph for input ranges within 2000(red line) as below tabular

Input Info for Merge Sort and Time

No of Inputs	Time taken for Random Input	Time taken for Ascending Sorted input
100	0.000603067	0.000552161
500	0.003601979	0.003304346
1000	0.008241775	0.007237622
1500	0.013421827	0.011665995
2000	0.017480725	0.016839479

2. **Merge Vs Quick Sort (Random List) Graph 2**



1. Quick sort performs much better on an average case of unsorted list
2. We have set the last element as pivot for the quick sort
3. Merge sort is very little unaffected for sorted or random list inputs
4. **Quick sort portrays O(n log n) complexity in this scenario**
5. Merge sort portrays close to O(n log n) for both sorted and unsorted list
6. Concluded that Quick sort has better performance random inputs as compared to merge sort for finite range of inputs
7. Below inputs in tabular show that quick sort for unsorted array is close to O(nlogn)

Input Info for Quick Sort and Time

No of Inputs	Time taken for Random Input	Time taken for Ascending Sorted input
100	0.000246317	0.001674547
500	0.001706157	0.040375092
1000	0.003689011	0.139581827
1500	0.00550478	0.314178062
2000	0.008189638	0.558393404

Anomalies, Experimental and Theoretical Results:

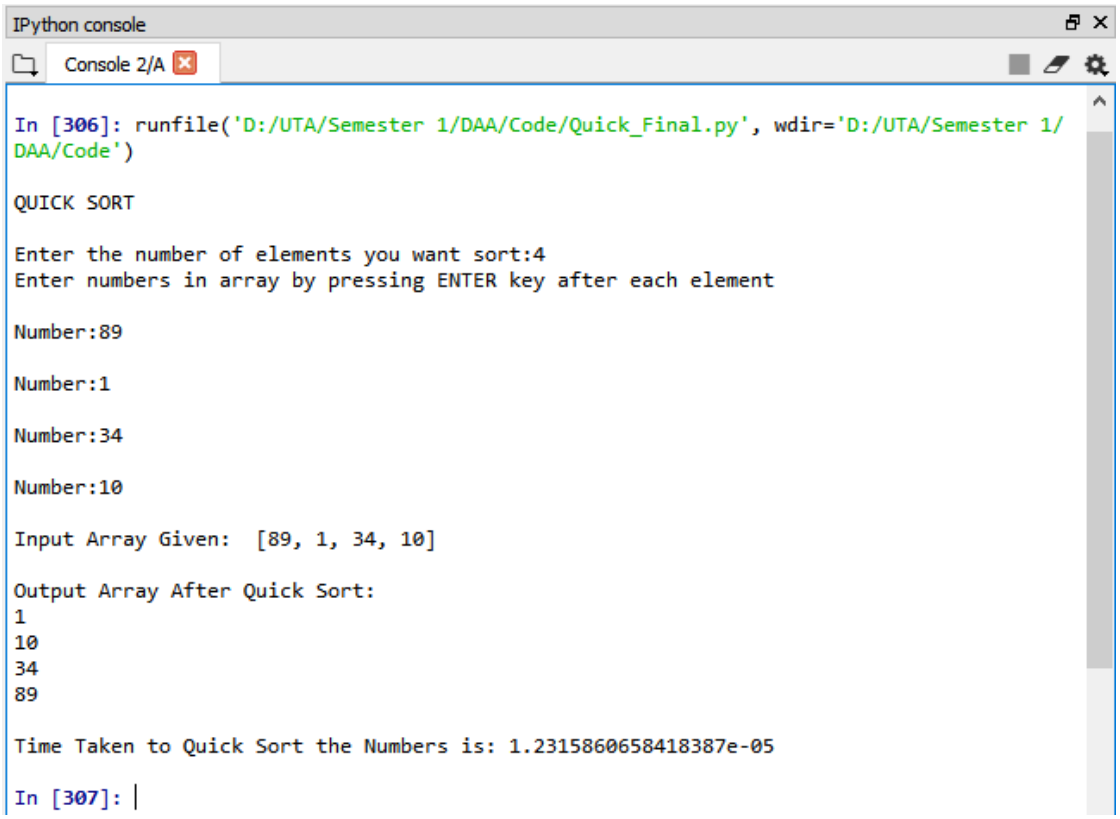
- It can be concluded that merge sort is a stable sort when compared to quick sort, Merge works better for any type of data set. Both sorted and unsorted
- Quick sort has worst case if the **pivot is selected poorly**, example right pivot with ascending sorted array is the worst case for quick sort showing $O(n^2)$ complexity. Sorting time increased considerably
- Theoretically and experimentally verified that worst case of Quick Sort is $O(n^2)$ and Merge Sort has a complexity of O(n log n)

- **Anomaly** can be seen in Graph2 for smaller random inputs, **Quicksort performance is best** as compared to merge sort for unsorted array or randomised array

Readme to Use the code:

Outputs Captured for Ease

Quick Sort output show how to use



Merge Sort Output: