DAA LAB 2

AIM: In this lab we would be implementation two sorting algorithm, quick sort and merge sort. The implementation would be carried out using iterative and recursive methods. Additionally, we would be also computing the memory utilization for each program

1.Quick Sort

QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of Quicksort that pick pivot in different ways.

1. Always pick first element as pivot.
2. Always pick last element as pivot.
3. Pick a random element as pivot.
4. Pick median as pivot.

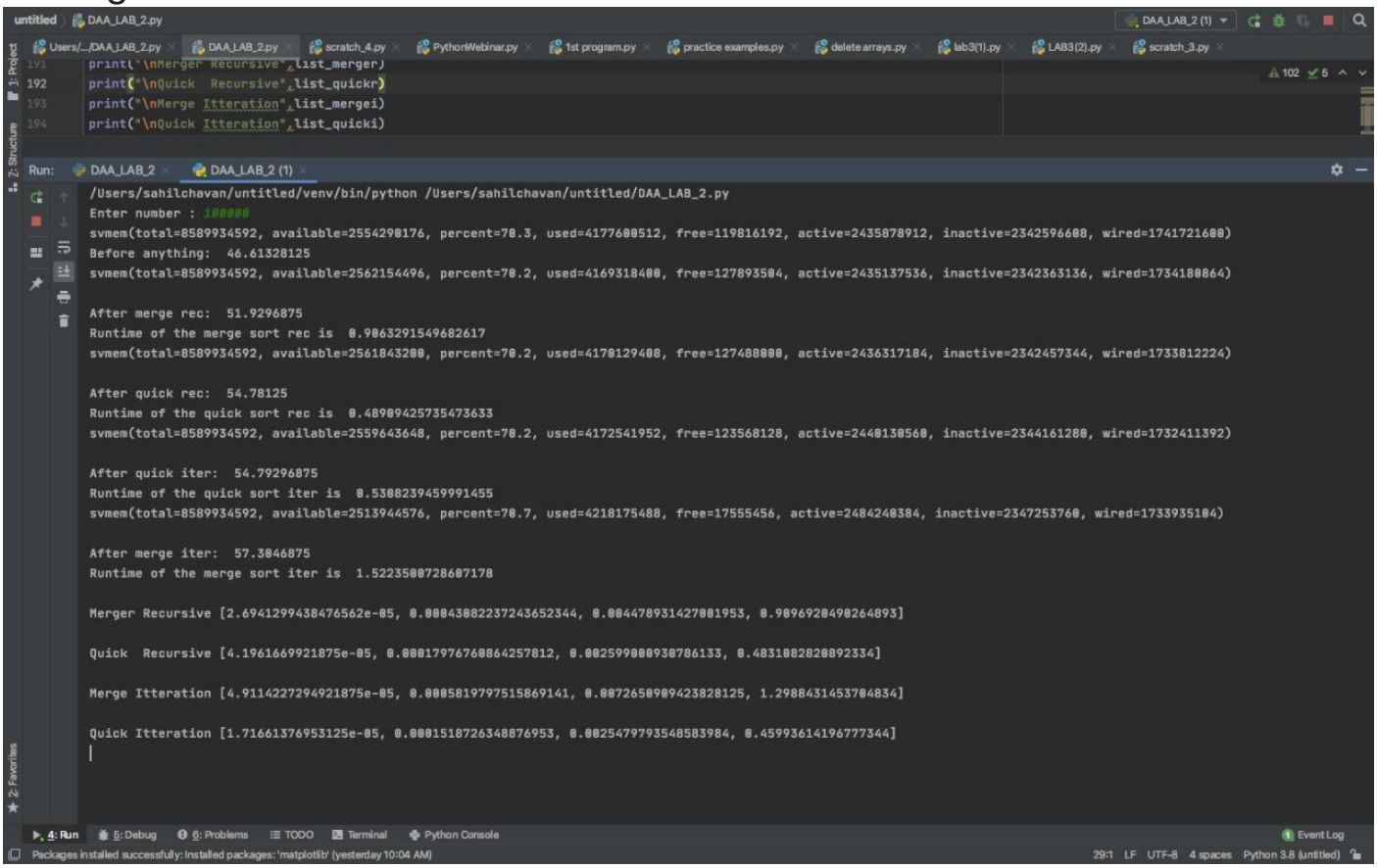
Worst complexity: n^2
Avg complexity: n*log(n) **Best complexity:** n*log(n)

2.Merge Sort

Merge Sort is a Divide and Conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. **The merge () function** is used for merging two halves. The merge (arr, l, r) is key process that assumes that arr[l] and arr [r] are sorted and merges the two sorted sub-arrays into one.

Worst complexity: n*log(n)
Avg complexity: n*log(n)
Best complexity: n*log(n)

Taking n : 100000



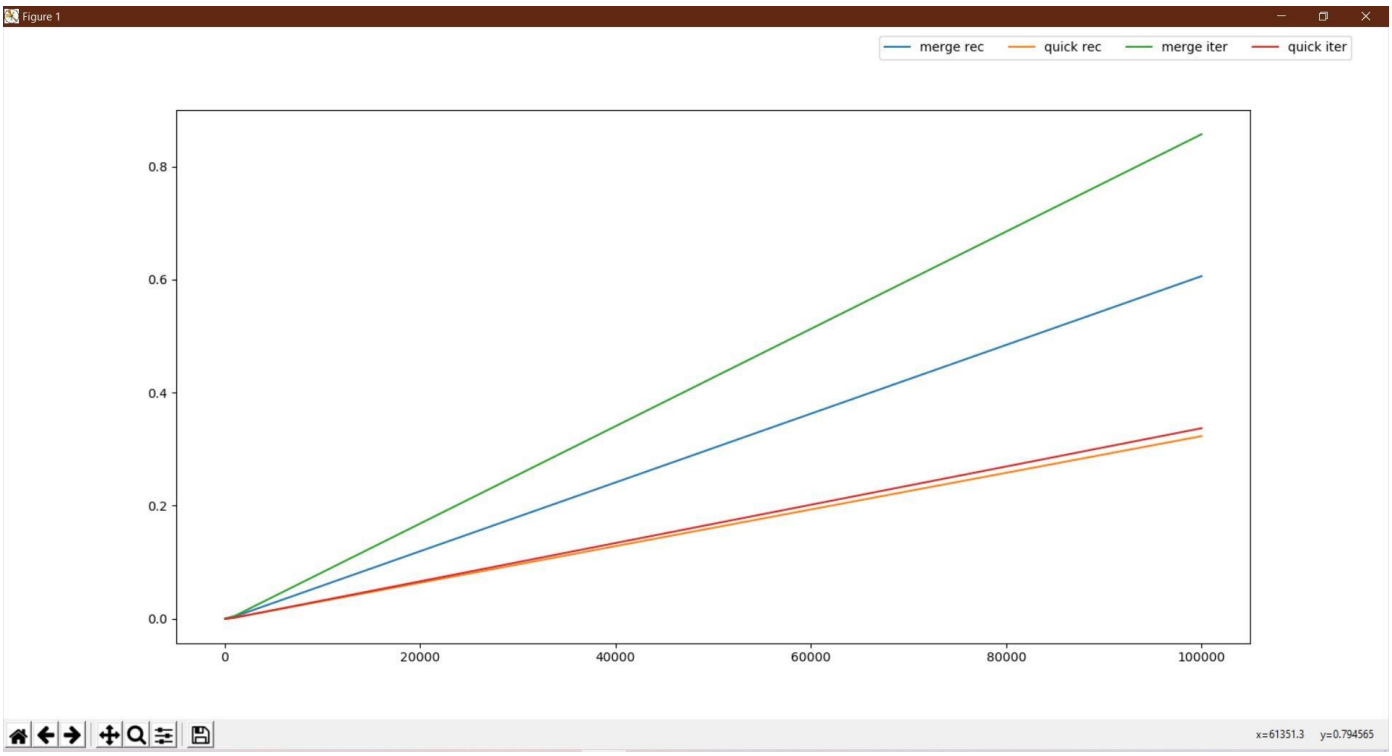
1) While checking the performance for randomly generated array I observed that out of all sorting algorithms if runtime was considered Iterative Merge Sort was the worst followed by Recursive Merge Sort. While both Quick Sorts were very close in runtime Recursive was clearly faster for large input sizes. By Space complexity however iterative merge sort and recursive quick sort took the least space while the other two were equally worse.

2) The Table below gives time(s) taken by each sort for values of n:

Sort n	10	100	1000	10000	100000
Merge (Rec)	0.0	0.0	0.00399	0.6010	0.90632
Merge (Itter)	0.0	0.0	0.0050	0.7820	1.5223
Quick (Rec)	0.0	0.0	0.00200	0.3609	0.48909

Quick (Itter)	0.0	0.0	0.0020	0.3410	0.5308

3)The Graph for the above table is given below:



4) Computing the Memory taken for n=10000

	Total Memory used(MB)	Actual Memory used(MB)
Initial Before Sort	73.98	0
Merge Sort (Recursive)	73.98	73.98
Quick Sort (Recursive)	73.96	0.002
Quick Sort (Iterative)	73.97	0.001
Merge Sort (Iterative)	73.98	0.001

Name :- Sahil Dinesh Chavan
PRN No. :- 201900802042