**Analysis and Design of Algorithms**

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

**Lab - 2**  Date : 29-09-2021

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

**AIM:** Implement merge sort and quick sort using iterative and recursive methods. 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.

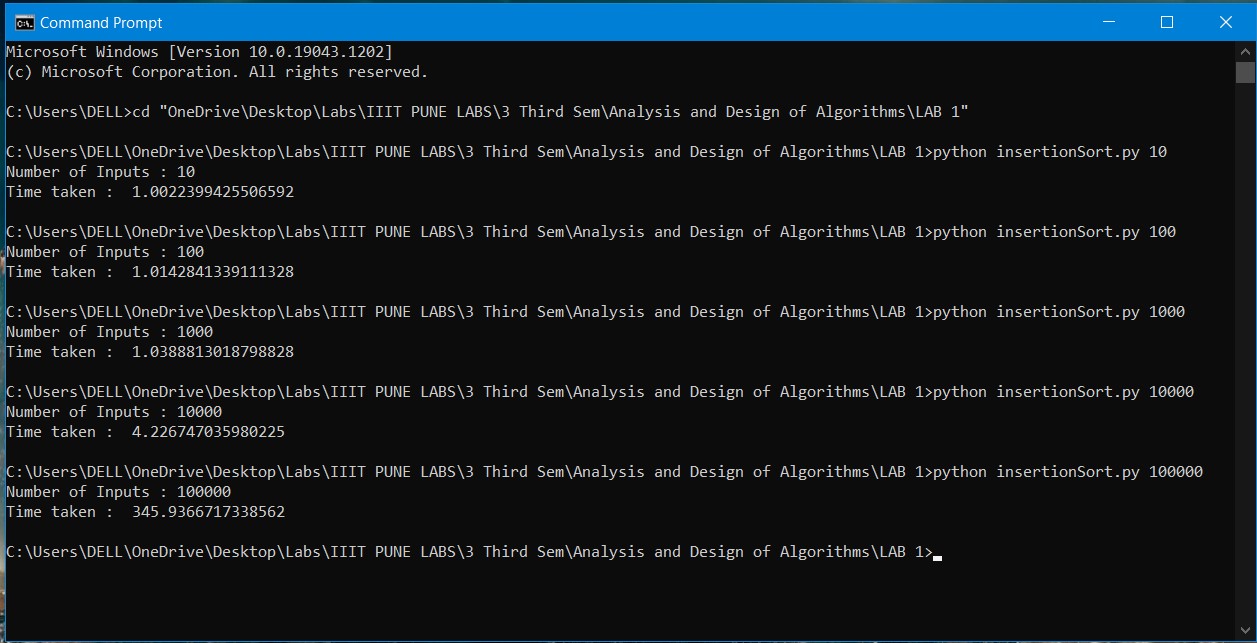
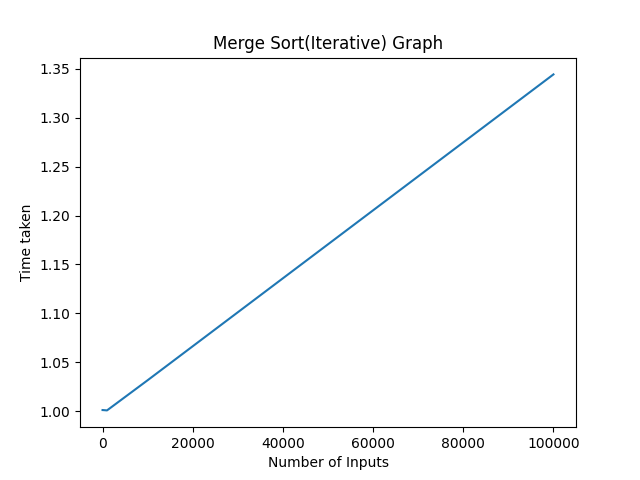
b. Compute the time taken by each case (for particular number of inputs).

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

d. Compute and compare the memory taken by recursive and iteration implementation of the two sorting algorithms.

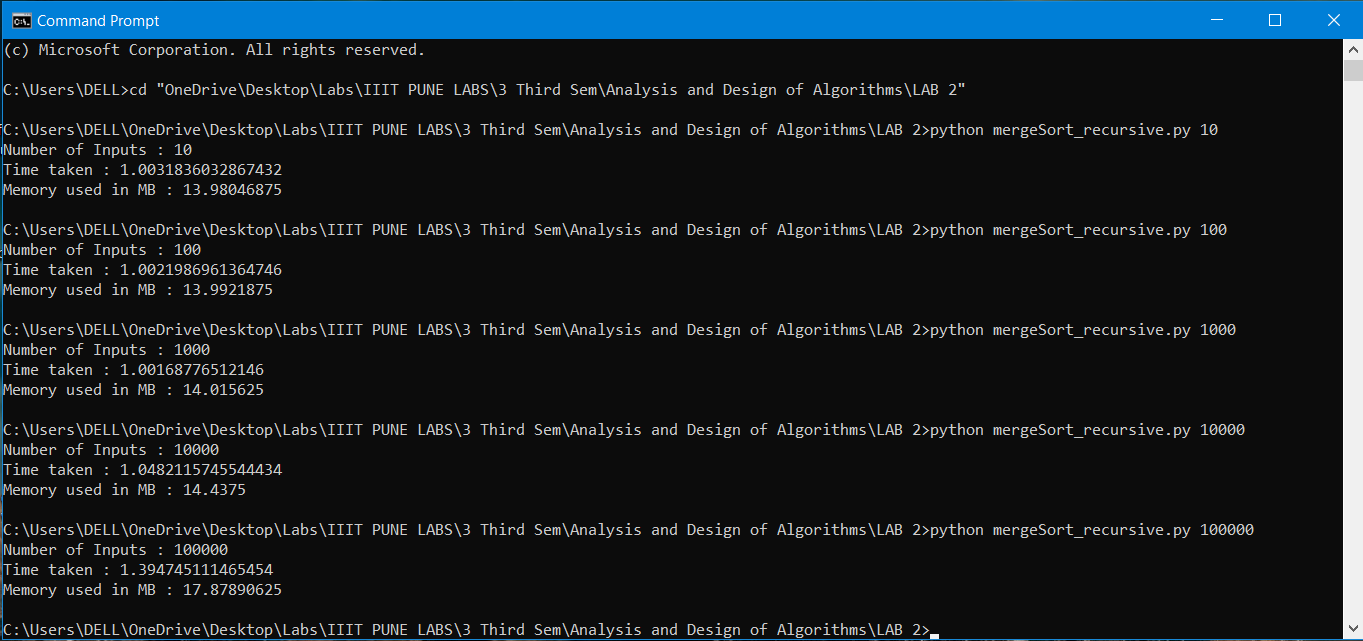
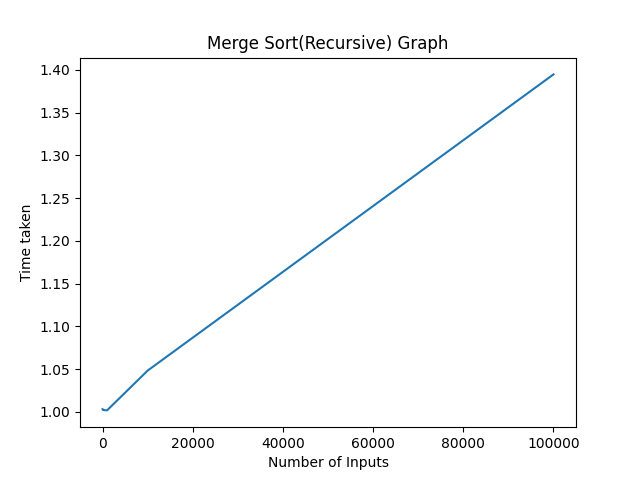
**Merge Sort (Iterative):**

Output: Graph:

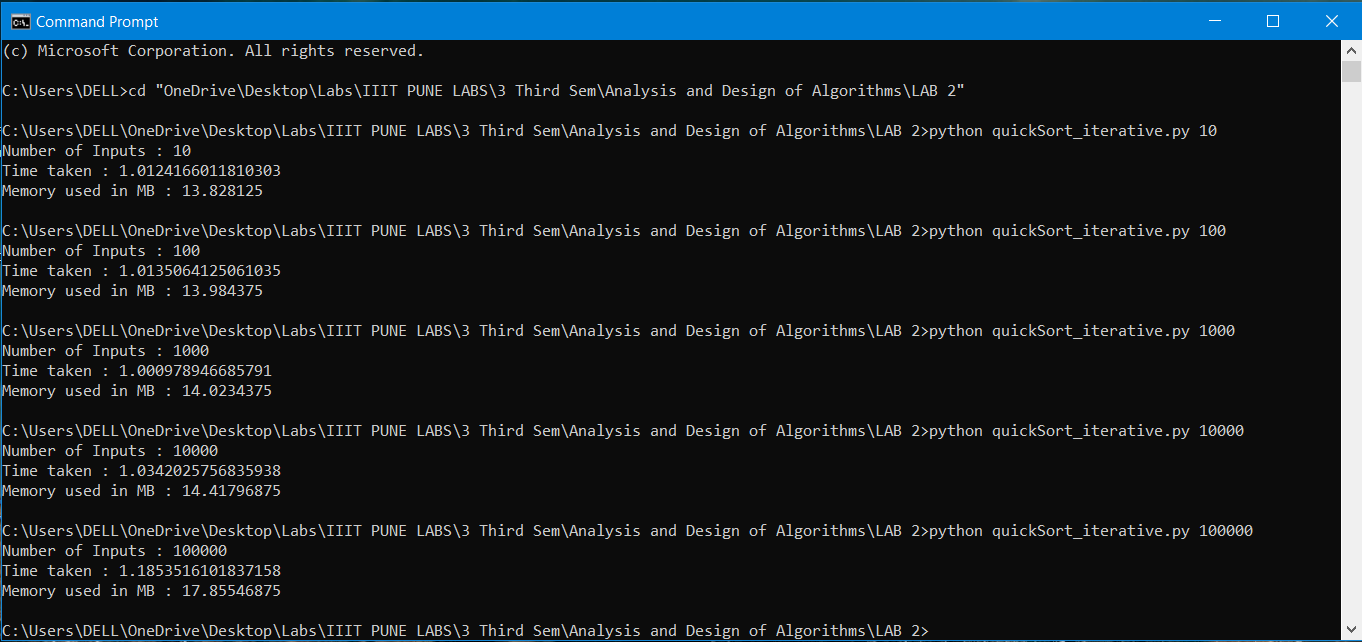
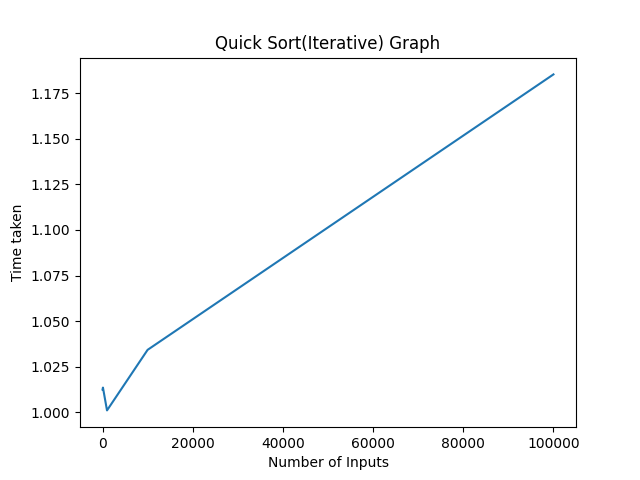
**Merge Sort (Recursive):**

Output: Graph:

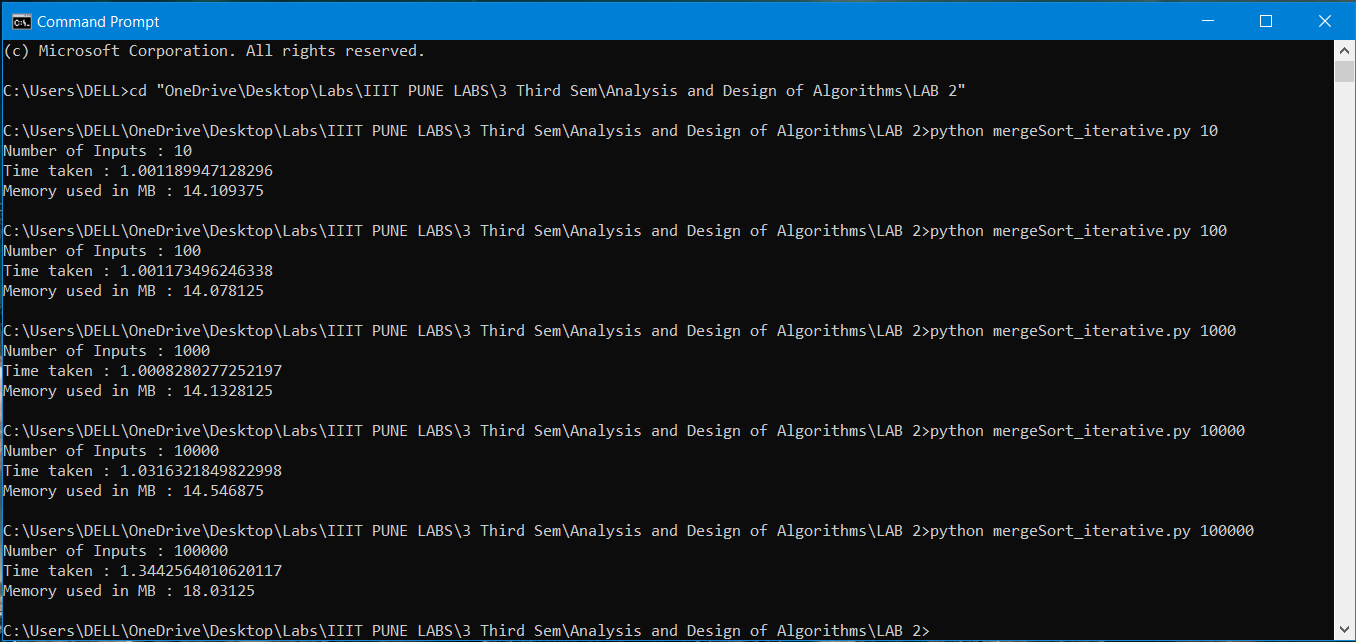
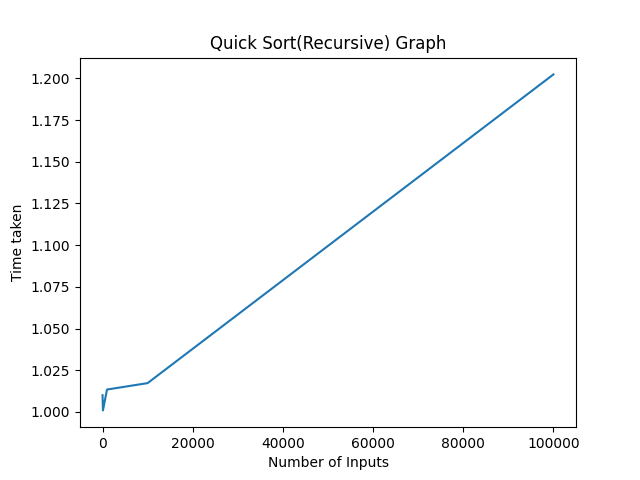
**Quick Sort (Iterative):**

Output: Graph:

**Quick Sort (Recursive):**

Output: Graph:

**Result:**

Here are the results obtained showing the time taken for merge and quick sort both iterative and recursive way to run different number of inputs.

Time taken for different inputs: \* All times obtained were added with one second using time.sleep(1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number of Inputs  (To right) | 10 | 100 | 1000 | 10000 | 100000 |
| Merge Sort(Iterative) | 1.001189947128296 | 1.001173496246338 | 1.0008280277252197 | 1.0316321849822998 | 1.3442564010620117 |
| Merge Sort (Recursive) | 1.0031836032867432 | 1.0021986961364746 | 1.00168776512146 | 1.0482115745544434 | 1.394745111465454 |
| Quick Sort (Iterative) | 1.0124166011810303 | 1.0135064125061035 | 1.000978946685791 | 1.0342025756835938 | 1.1853516101837158 |
| Quick Sort (Recursive) | 1.0100398063659668 | 1.0009138584136963 | 1.0134236812591553 | 1.0172724723815918 | 1.202369213104248 |

Average of time taken:

Merge Sort(Iterative) : 1.0758160114288

Merge Sort(Recursive) : 1.0900053501129

Quick Sort(Iterative) : 1.049291229248

Quick Sort(Recursive) : 1.0488038063049

Memory Usage**:** \* Memory in MB (Mega Bytes)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number of Inputs  (To right) | 10 | 100 | 1000 | 10000 | 100000 |
| Merge Sort(Iterative) | 14.109375 | 14.078215 | 14.1328125 | 14.546875 | 18.03125 |
| Merge Sort (Recursive) | 13.98046875 | 13.9921875 | 14.015625 | 14.4375 | 17.87890625 |
| Quick Sort (Iterative) | 13.828125 | 13.984375 | 14.0234375 | 14.41796875 | 17.85546875 |
| Quick Sort (Recursive) | 13.984375 | 13.984375 | 13.96484375 | 14.35546875 | 17.7890625 |

Average of memory usage:

Merge Sort(Iterative) : 14.9797055

Merge Sort(Recursive) : 14.8609375

Quick Sort(Iterative) : 14.821875

Quick Sort(Recursive) : 14.815625

**Conclusion:**

After analysing the results obtained, I observed that in case of merge sort and quick sort memory usage during iterative implementation is larger than in recursive implementation by a little fraction. But recursive implementation uses more memory than iterative implementation. This small fraction in the result might have occurred because of random input. But when compared with same input recursive implementation uses more memory compared to iterative implementation.

Comparing the time taken to sort the elements, the iterative implementation is little faster than recursive implementation. In some cases, recursive implementation is faster than iterative implementation. On average, iterative implementation is way more bit faster than recursive implementation.