**LAB 1**

**AIM:**

Write a program to evaluate the performance of Bubble Sort, Insertion Sort, Selection Sort, Merge Sort and Quick Sort.

**EXPERIMENT:**

A sorting algorithm is an algorithm that arranges the elements of a list in order. The theoretical study of computer-program performance and resource usage is called an algorithm.

**Bubble Sort**: Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order. In each iteration, the largest element bubbles out towards the end.

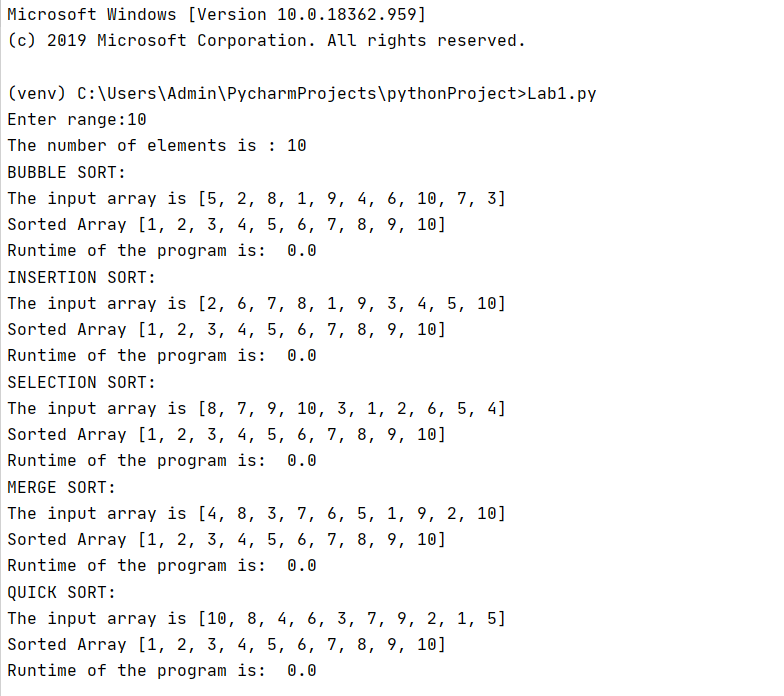
**Insertion Sort**: Insertion sort is an algorithm in which the array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

**Selection Sort**: The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array and in each iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

**Merge Sort**: Merge Sort is a [Divide and Conquer](https://www.geeksforgeeks.org/divide-and-conquer-introduction/) algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves.

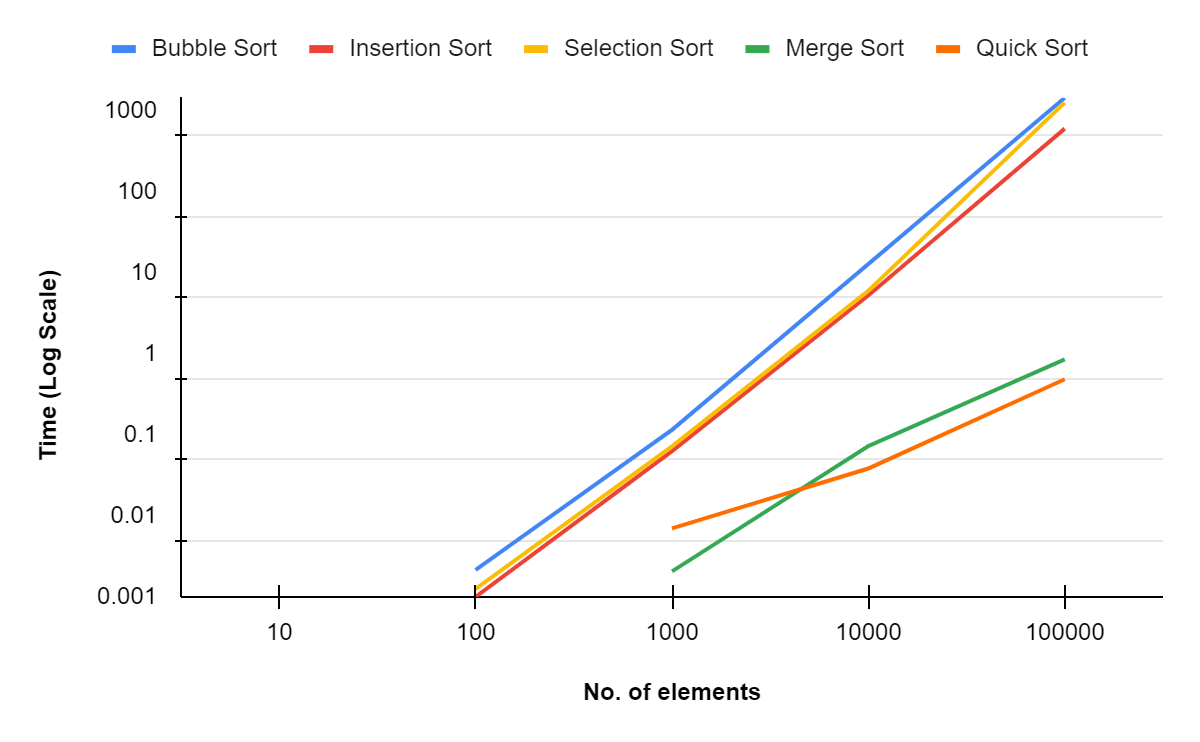
**Quick Sort**: Quick Sort depends on the Divide and Conquer algorithm. It does partition of the array into subarrays around the pivot(x) such that the elements smaller than pivot are placed before x and elements greater than pivot are placed after x.

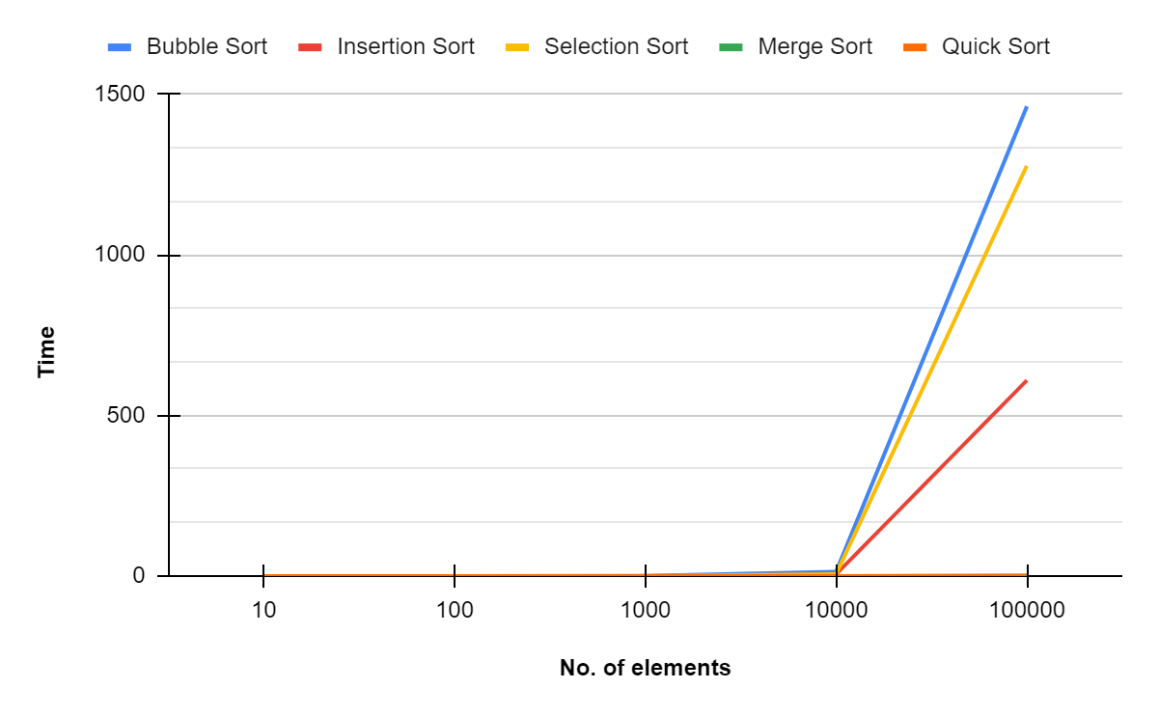
**OUTPUT** (for 10 numbers):



GRAPH:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | Bubble Sort | Insertion Sort | Selection Sort | Merge Sort | Quick Sort |
| 10 | 0 | 0 | 0 | 0 | 0 |
| 100 | 0.002143383 | 0.000996351 | 0.001248837 | 0 | 0 |
| 1000 | 0.115070581 | 0.062875748 | 0.072017908 | 0.002058029 | 0.007017851 |
| 10000 | 12.90969276 | 5.323912382 | 6.071282148 | 0.073041916 | 0.038439989 |
| 100000 | 1462.688337 | 609.278585 | 1277.310182 | 0.856137276 | 0.48716712 |





If the array is already sorted (Worst Case):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | Bubble Sort | Insertion Sort | Selection Sort | Merge Sort |
| 10 | 0 | 0 | 0 | 0 |
| 100 | 0.0009968280792 | 0 | 0.0009911060333 | 0 |
| 1000 | 0.06813883781 | 0.0009977817535 | 0.06687092781 | 0.0009570121765 |
| 10000 | 7.24308157 | 0.002821207047 | 6.208508968 | 0.07608103752 |
| 100000 | 678.3921392 | 0.02589082718 | 1326.423532 | 0.7676782608 |

**CONCLUSION:**

Hence, the performance of Bubble Sort, Insertion Sort, Selection Sort, Merge Sort and Quick Sort has been studied.