Run time analysis of Sorting Algorithms based on different input types

CS506 Lab

Objective: To implement and analyze the time complexity of different comparison based sorting algorithms.

Report By:

Kumar Mangalam
2022AIM1002

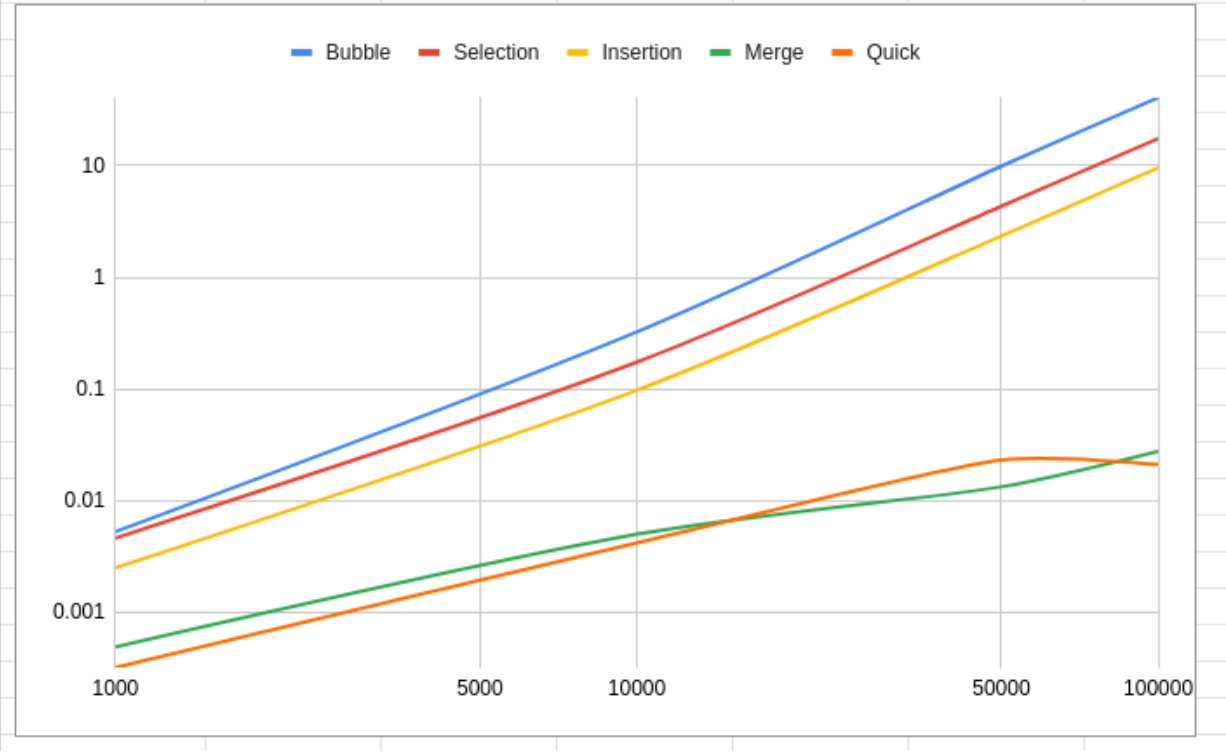
Instructor:

Dr. Puneet Goyal

TIME COMPARISON FOR DIFFERENT SORTING ALGORITHMS (Based on Input Type)

1. **Random Order** (Sort -Optimized Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort) **(k=0)**

Number of Inputs	Bubble	Selection	Insertion	Merge	Quick
1000	0.005322	0.004677	0.002527	0.000496	0.000325
10000	0.326071	0.17451	0.097948	0.005051	0.004226
50000	9.902558	4.340001	2.348005	0.013393	0.023278
100000	40.547145	17.442686	9.572667	0.027953	0.021237



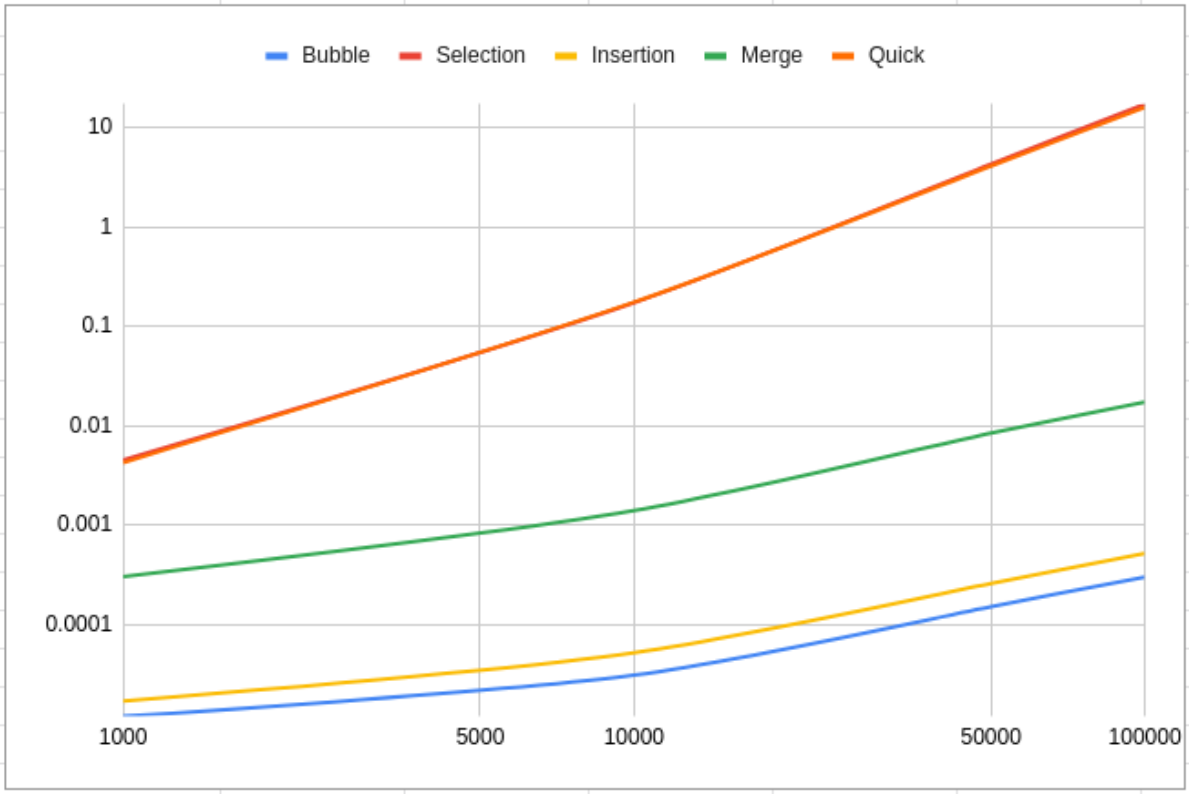
When the input array is in random order, it can be concluded that

Optimized Bubble < *Selection* < *Insertion* < *Merge* < *Quick*

would be the order of sorting algorithms in terms of efficiency. Though we can also observe merge sort giving faster results for a certain number of inputs but with the increase of number of inputs quick sort gives faster results.

2. **Sorted Order** (Sort -Optimized Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort) **(k=1)**

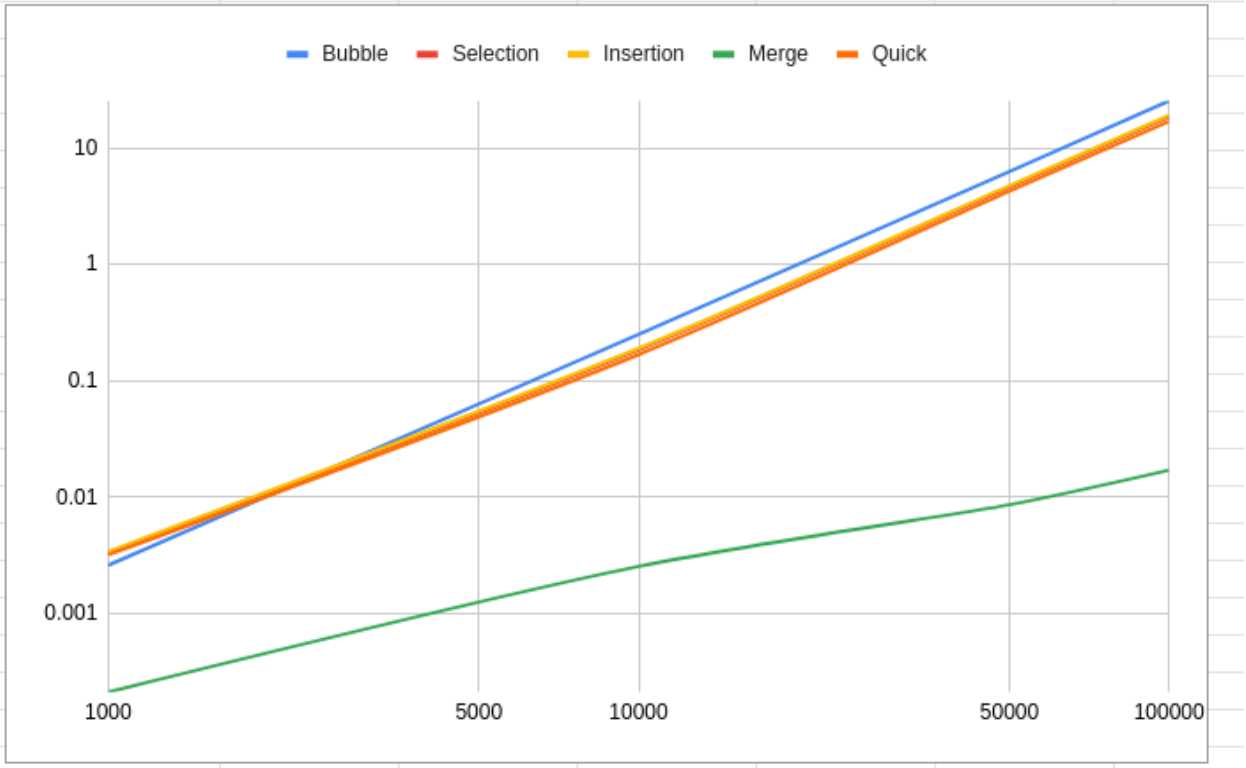
Number of Inputs	Bubble	Selection	Insertion	Merge	Quick
1000	0.000012	0.004534	0.000017	0.000304	0.00428
10000	0.000031	0.173799	0.000052	0.001403	0.176085
50000	0.000152	4.337526	0.000259	0.0085	4.130454
100000	0.000301	17.346065	0.000521	0.017344	16.141901



When the input array is in sorted order, it can be concluded that *Selection* < *Quick* < *Merge* < *Insertion* < *Optimized Bubble* would be the order of sorting algorithms in terms of efficiency. Here we can observe that that quick sort and selection sort performs the worst from starting.

3. **Reversely Sorted Order** (Sort -Optimized Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort) **(k=2)**

Number of Inputs	Bubble	Selection	Insertion	Merge	Quick
1000	0.002599	0.003289	0.003404	0.000212	0.003196
10000	0.251291	0.184995	0.190259	0.002547	0.168387
50000	6.307851	4.630527	4.74532	0.008634	4.284117
100000	25.51639	18.590785	19.088451	0.01713	16.987642



When the input array is in reversely sorted order, it can be concluded that

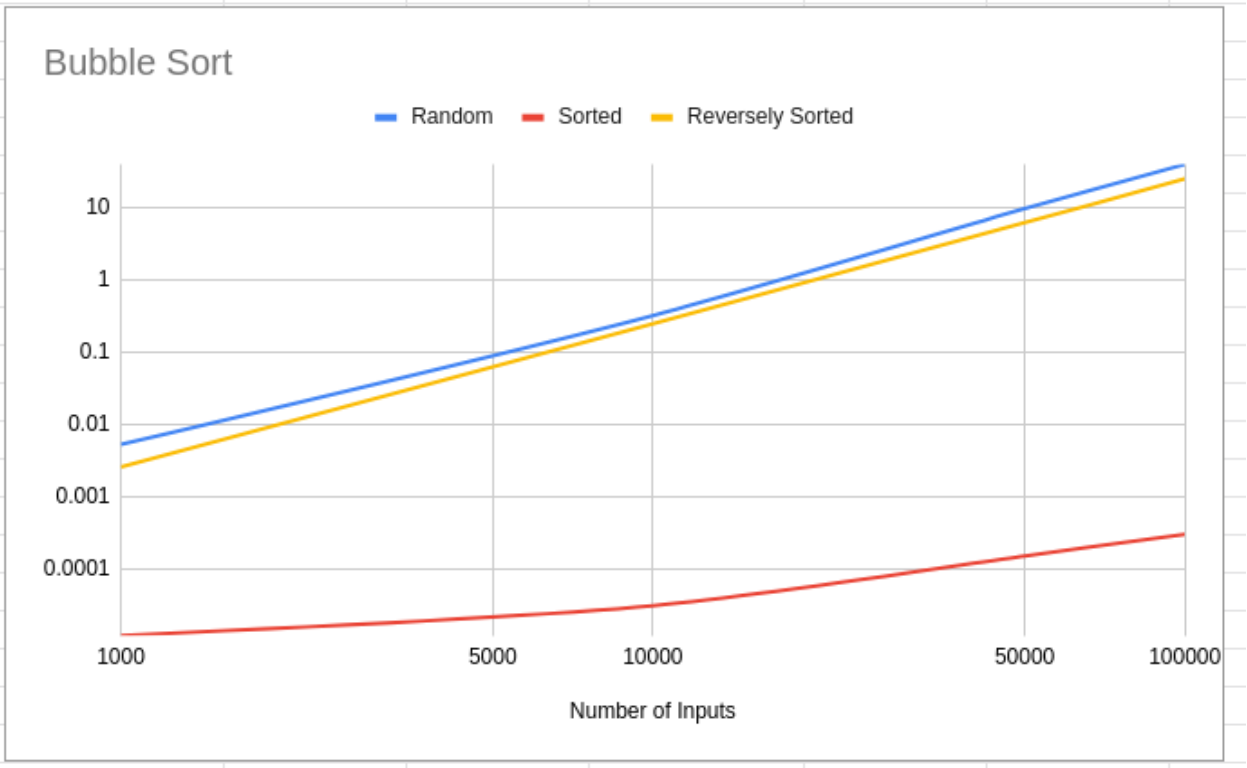
Optimized Bubble < *Insertion* < *Selection* < *Quick* < *Merge*

would be the order of sorting algorithms in terms of efficiency. Here we can observe one more thing that apart from merge sort, all the other sorting algorithms have this as their worst case.

TIME COMPARISON FOR DIFFERENT SORTING ALGORITHMS

1. **Optimized Bubble Sort** (Input type - Random order, Sorted order, Reversely sorted order)

Number of Inputs	Random	Sorted	Reversely Sorted
1000	0.005322	0.000012	0.002599
10000	0.326071	0.000031	0.251291
50000	9.902558	0.000152	6.307851
100000	40.547145	0.000301	25.51639



In the Optimized Bubble sort we can observe that, when the input array is in random or reversely sorted order the time taken to sort the array shoots compared to the input array in sorted order.

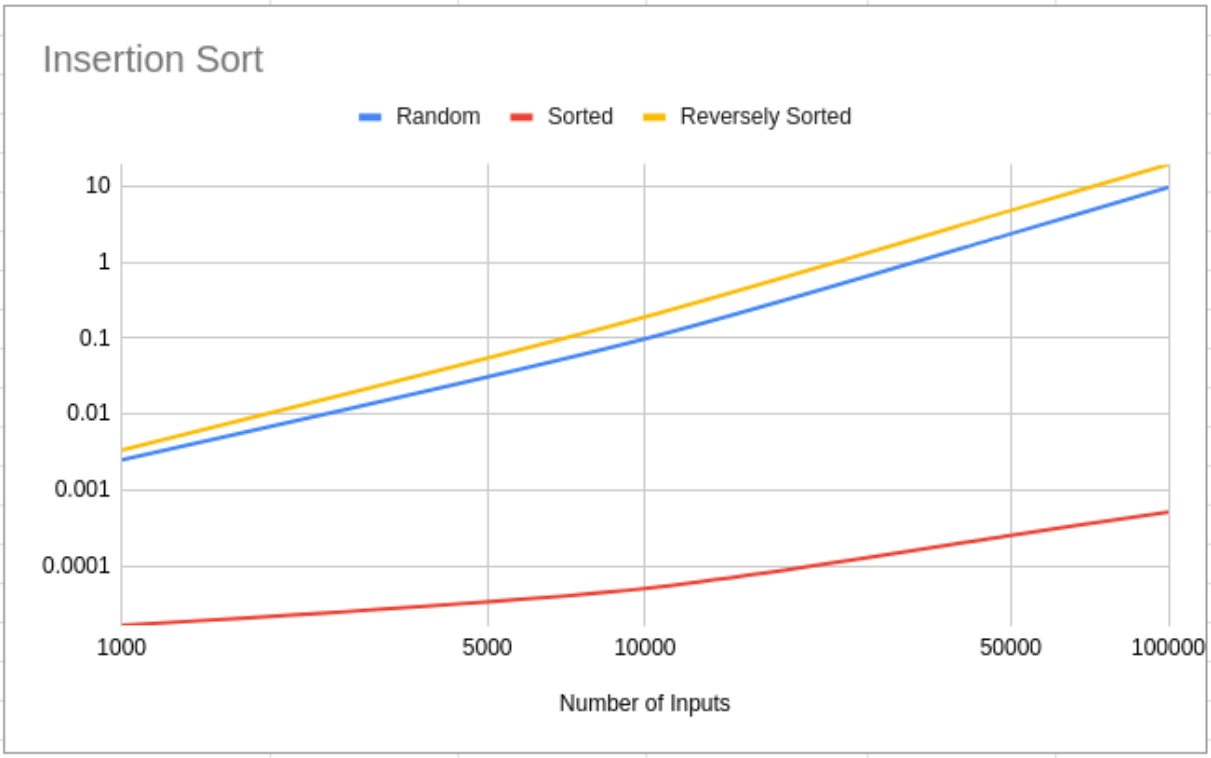
2. **Selection Sort** (Input type - Random order, Sorted order, Reversely sorted order)

Number of Inputs	Random	Sorted	Reversely Sorted
1000	0.004677	0.004534	0.003289
10000	0.17451	0.173799	0.184995
50000	4.340001	4.337526	4.630527
100000	17.442686	17.346065	18.590785

Compared to Optimized Bubble sort though it takes less time to sort the array when the input array is in random or reversely sorted order for a very large number of inputs. But the Optimized Bubble sort is far more efficient when the input array is in sorted order. Selection sort is almost equally efficient for all given kinds of input array.

3. **Insertion Sort** (Input type - Random order, Sorted order, Reversely sorted order)

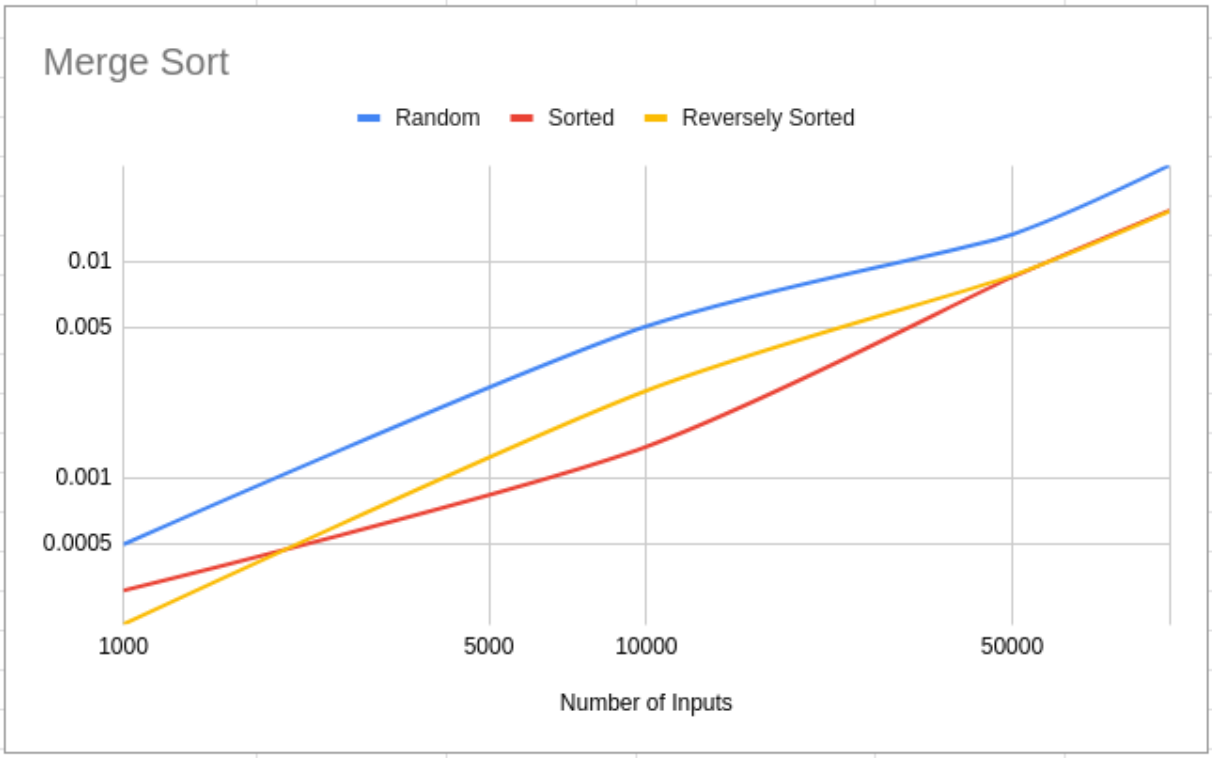
Number of Inputs	Random	Sorted	Reversely Sorted
1000	0.002527	0.000017	0.003404
10000	0.097948	0.000052	0.190259
50000	2.348005	0.000259	4.74532
100000	9.572667	0.000521	19.088451



Compared to Selection sort and Optimized Bubble sort it can be said Insertion sort is more efficient than both of them in general as when comparing for input array in sorted order insertion sort is slightly less efficient than Optimized Bubble sort, for reversely sorted order it takes slightly more time than selection sort but when the input array is in random order then it's much more efficient than both Optimized Bubble and selection sort.

4. **Merge Sort** (Input type - Random order, Sorted order, Reversely sorted order)

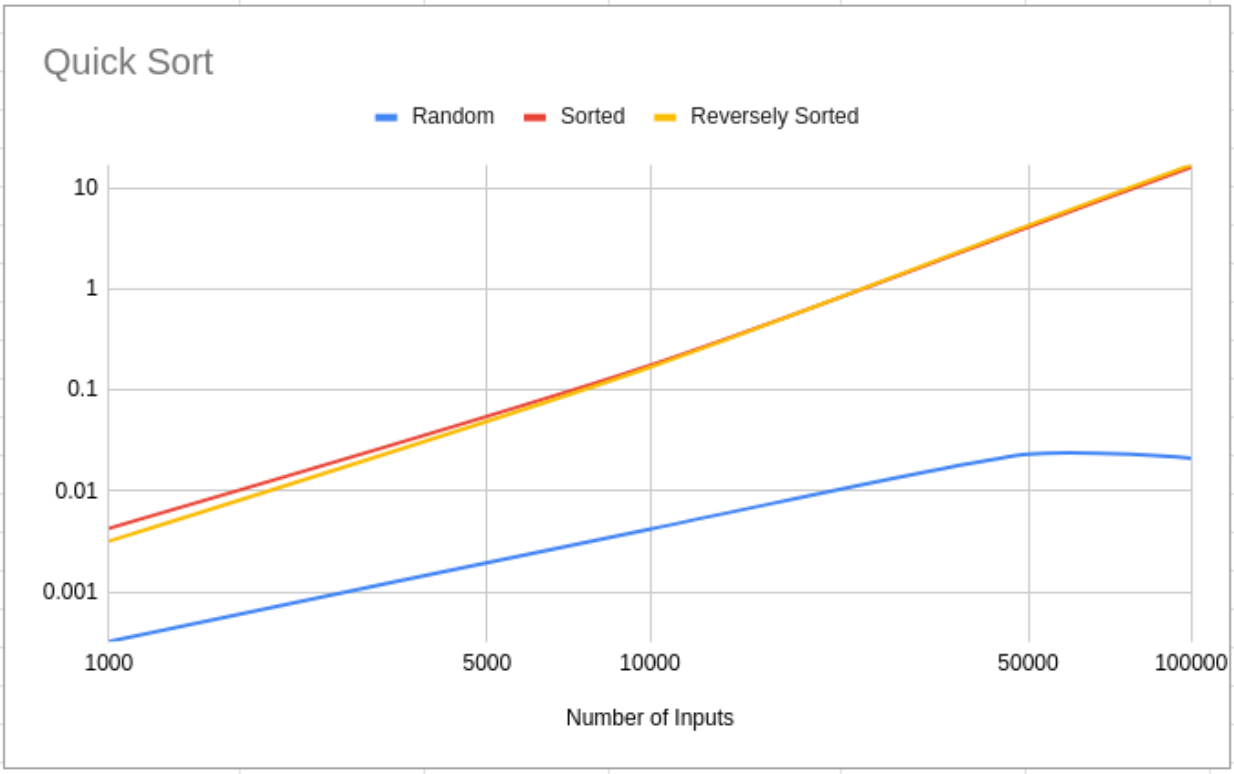
Number of Inputs	Random	Sorted	Reversely Sorted
1000	0.000496	0.000304	0.000212
10000	0.005051	0.001403	0.002547
50000	0.013393	0.0085	0.008634
100000	0.027953	0.017344	0.01713



Compared to previous sorting algorithms, merge sort is much more efficient but on certain conditions like when the input array is in sorted order then the Optimized Bubble sort and insertion sort is much more efficient for a greater number of inputs.

5. **Quick Sort** (Input type - Random order, Sorted order, Reversely sorted order)

Number of Inputs	Random	Sorted	Reversely Sorted
1000	0.000325	0.00428	0.003196
10000	0.004226	0.176085	0.168387
50000	0.023278	4.130454	4.284117
100000	0.021237	16.141901	16.987642



It can be said that quick sort is the fastest sorting algorithm when the input array is in random order but when the input array is in sorted or reversely sorted order (worst case scenario) then quick sort efficiency decreases.

Citations:

1. Used Geeksforgeeks/ Abdul Bari youtube lectures to understand the algorithms.
2. Sort Graph contains the output data generated by me to create graphs and the input data used to create the graph.

Things to be considered:

1. In the graphs Y-axis denotes time taken to sort the array and X-axis denotes number of elements in the input array. Log scale was used during plotting.