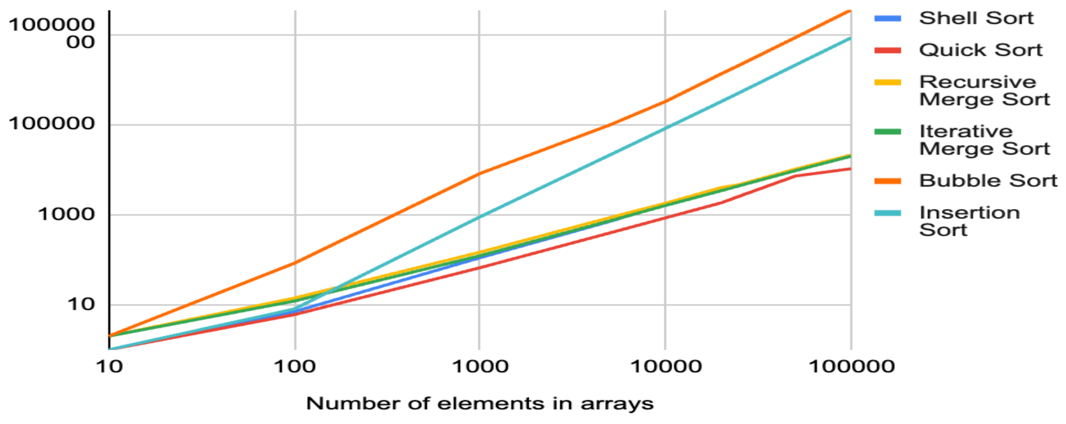
Program 4 | **Performance Report of Different Sorts** |

This report illustrates the performance of different sorting algorithms that includes– bubble sort, insertion sort, quick sort, shell sort, merge(recursion) sort, and iterative merge sort. This report compares the performance of these sorts by checking the time in microseconds it takes to sort the number of elements in arrays of a vector. Below is the table of the data calculated through this running this program:

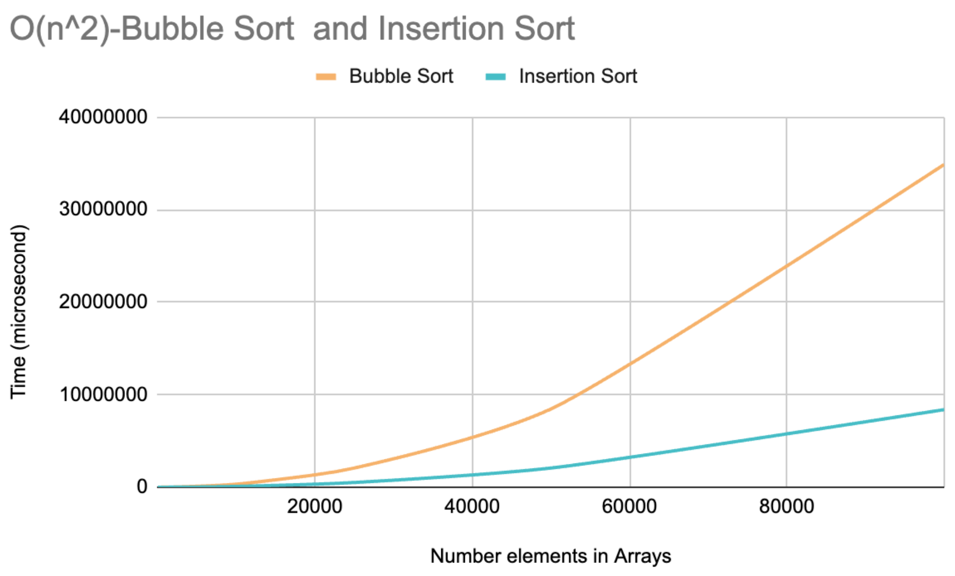
# of elements in Arrays	**Shell Sort**	**Quick Sort**	**Recursive Merge Sort**	**Iterative Merge Sort**	**Bubble Sort**	**Insertion Sort**
	Time in microseconds					
10	1	1	2	2	2	1
100	7	6	14	12	83	8
1000	110	66	146	122	8140	888
5000	702	390	853	723	97061	20908
10000	1673	845	1785	1590	323643	81927
20000	3559	1828	3991	3392	1334906	323634
25000	4645	2558	4631	4358	2071509	507498
50000	9834	7150	10212	9409	8456221	2071240
100000	21074	10406	20972	19733	34895891	8384979
The data above shows the numbers representing the time in microseconds it took for each sorting method to sort the elements in the arrays.						

In this above table, it can be seen that bubble and insertion sorts did not perform very well for larger element data as they took extremely longer to sort elements as compared to other sorts. However, quick sort seems to be the most efficient in higher element data along with iterative merge, shell and then recursive merge sort. Below is the overall graph of the above sorts from the above calculated time in microsecond:



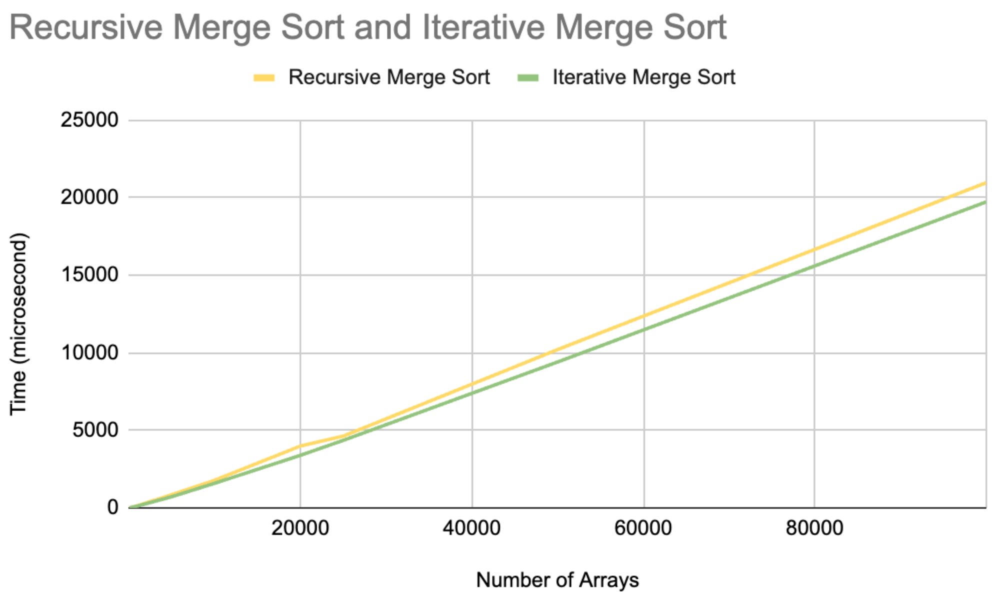
Bubble and Insertion-O(n^2):

The big O notation of bubble and insertion sort is the same which is O(n^2) but insertion outperforms and is very efficient in comparison to bubble sort. The below graph shows the comparison of performance of each sort in respect to time in microseconds:



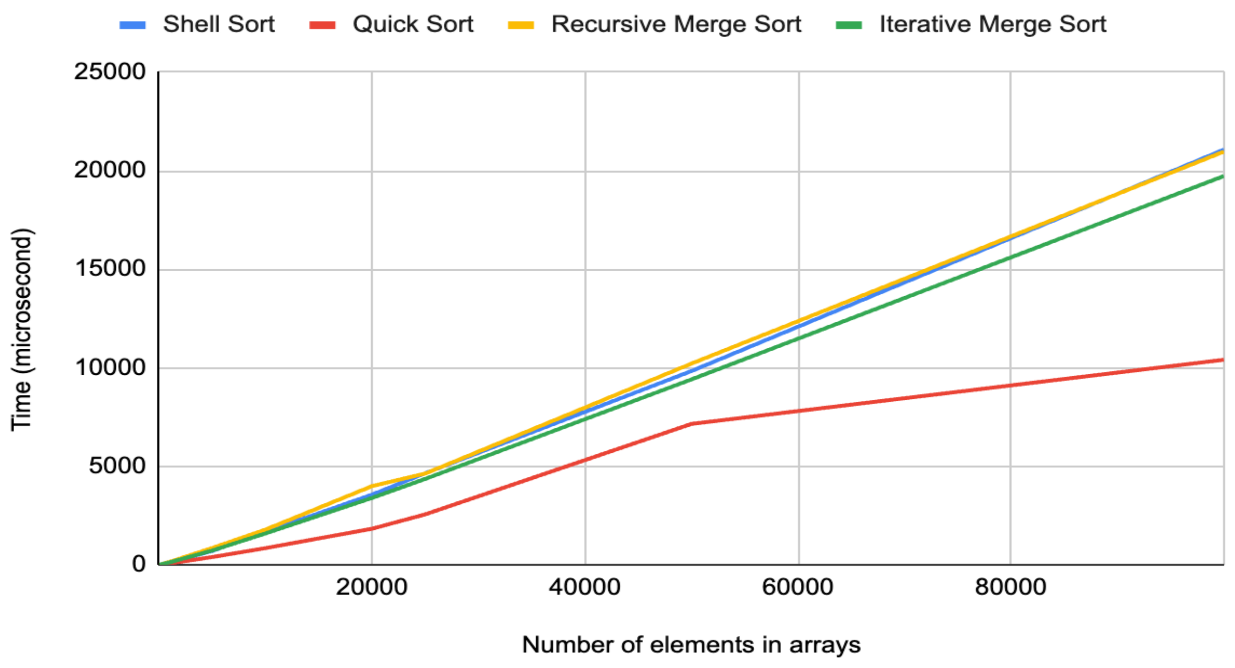
Recursive and Iterative Merge- O(nLog(n)):

The iterative and recursive merge sort have the same big O notation which is O(nLog(n)) but iterative outperforms recursive as it sorts the elements faster. The below graph shows the comparison of performance of each sort in respect to time in microseconds:



O(nLog(n)) Big O notation:

Shell Sort, Quick Sort, Recursive, and Iterative Merge Sort have a Big O(n log n) notation for their best cases, and they exhibit minor time differences in sorting the elements of the list. Quick sort appears to outperform other sorting algorithms. Following Quick Sort, Iterative Merge Sort performs better, followed by Shell Sort and, at least, Recursive Merge Sort. The graph below illustrates the performance comparison of each sorting algorithm in terms of time in microseconds:



In conclusion, the most efficient sorting algorithm based on the time calculated in microseconds is Quick Sort and the least efficient turns around to be the Bubble sort and insertion after it. All others have very close time differences. Shell and Recursive merge sort share close numbers in regard to time complexity.