Programming Project Report

PROJECT IMPLEMENTATION

1. Pre-Requisites:

IDE: SpringToolSuite4 must be downloaded and installed.

JDK 1.8v

2. Build & Run the Project on Local

1. Open the project From
   File >> import >> maven >> existing maven project >> select the root directory as project folder and click finish.
2. Select src >> main >> java >> cloud >> uta >> sxg6582 >> daa_prj >> DAAPrjApplication.java
3. Right Click and select Run As >> SpringBootApp
4. Then open browser and open the following link:
   https://localhost:8080/home

You can generate the array of required size, perform all the sorting operation and see the runtime of each sorting function.

DATA STRUCTURES, DATA TYPES, COMPONENTS USED:

- **DataStructures & Datatypes Used:**

1. Array: For sorting of each function array is used as input to the sorting function; operations are performed on the same array and then return back by the selected sorting function.
2. String: User enters the input which is of type string: the input string is parsed and int array is passed to main sorting class.
3. Int: this data is used inside sorting functions for storing temporary values and to perform loop operation inside sorting function.
4. Long: this data type is used to get execution time in nanoseconds

- **Component Usage:**

1. Every sorting algorithm is created as function and placed inside.
src >> main >> java >> cloud >> uta >> sxg6582 >> daa_prj >> sorting.java

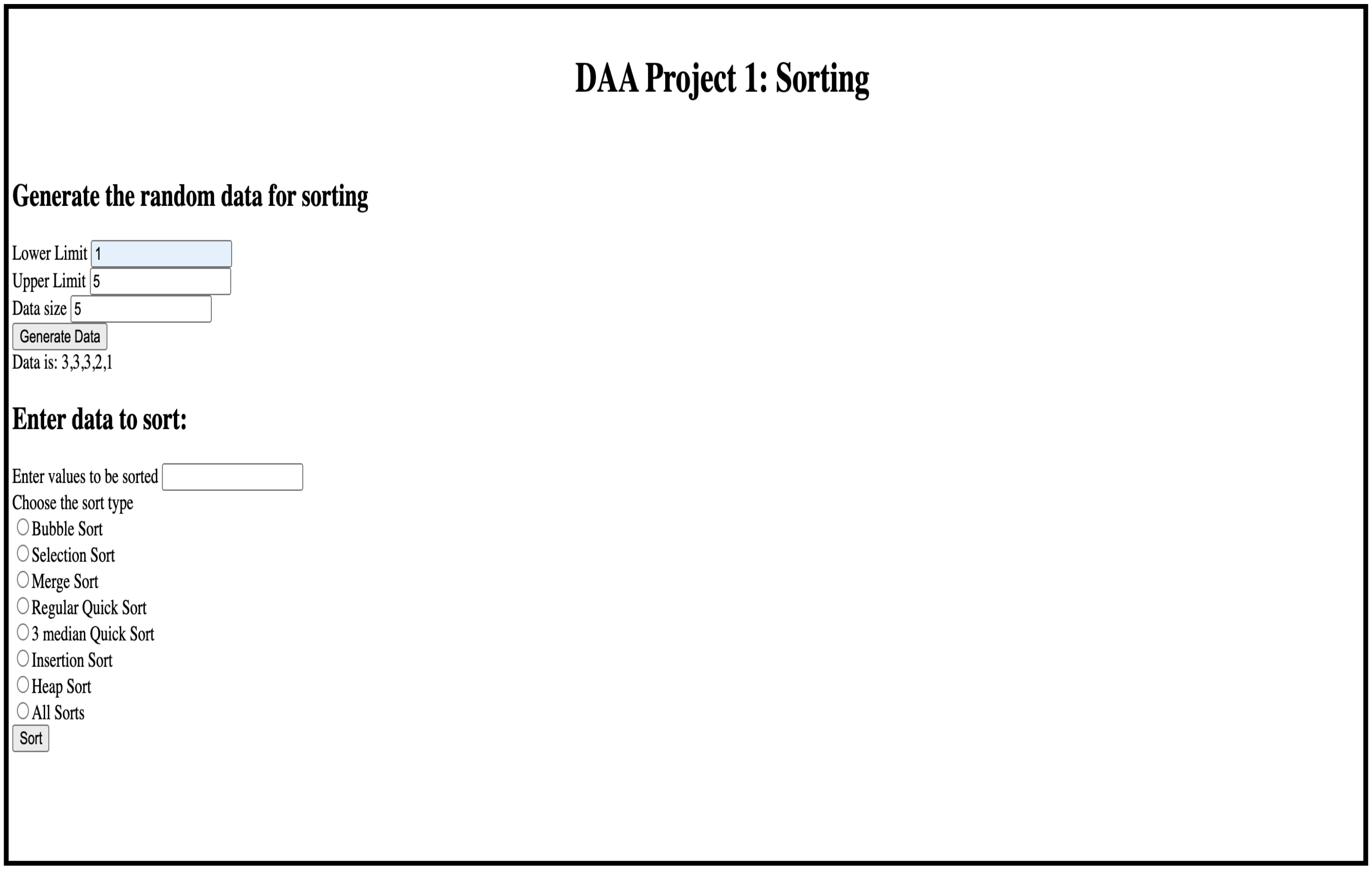
2. Project entry point:
src >> main >> java >> cloud >> uta >> sxg6582 >> daa_prj >> DAAPrjApplication.java

3. From entry point demoController.java class is called which gets input from the user and which then calls the sorting.java for sorting the data as per the user requirement

4. Website is created to run on local for ease of use in selecting the algorithm and check the execution time.

WEBSITE GUI USAGE INSTRUCTION:

1. Enter enter the input array of your size
2. Or generate an array using the Random number generator option by providing following input: upper-limit, lower-limit and number of array element required
   => This will display an array of required size which can copied and used as an input by user for sorting.
   Note: Hard limit to Input size has been set to 1000
3. Select one of the sorting option from the list and then click on sort button to get the result.



DAA Project 1: Sorting

Generate the random data for sorting

Lower Limit 1
Upper Limit 5
Data size 5
Generate Data
Data is: 3,3,3,2,1

Enter data to sort:

Enter values to be sorted
Choose the sort type
○ Bubble Sort
○ Selection Sort
○ Merge Sort
○ Regular Quick Sort
○ 3 median Quick Sort
○ Insertion Sort
○ Heap Sort
○ All Sorts
Sort

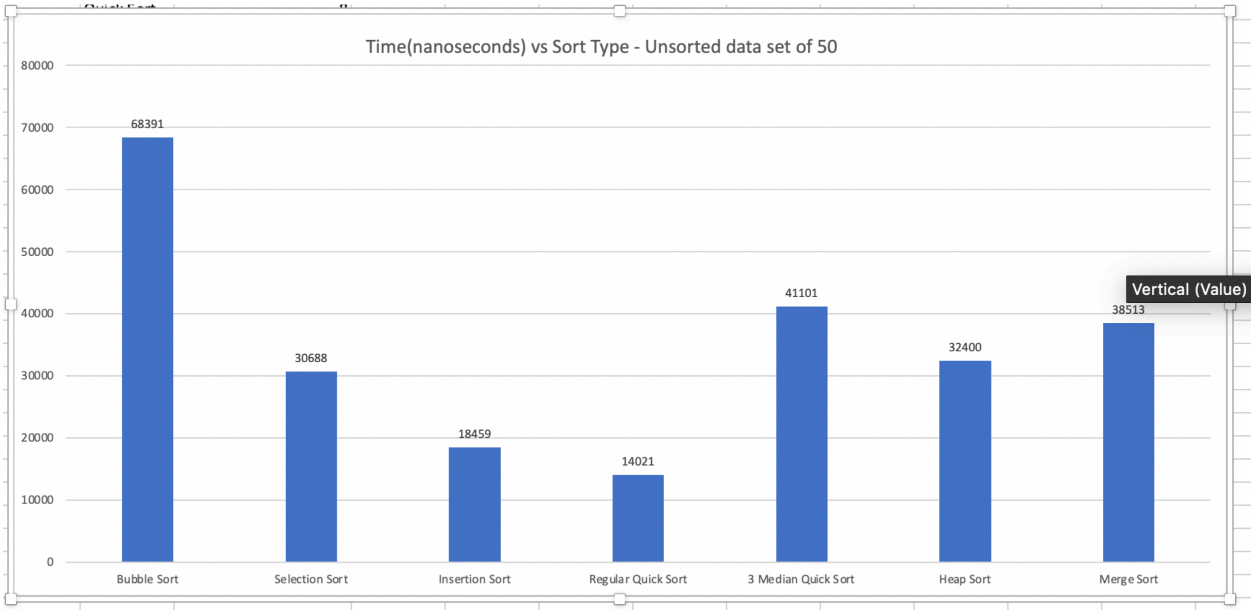
ALGORITHM ANALYSIS:

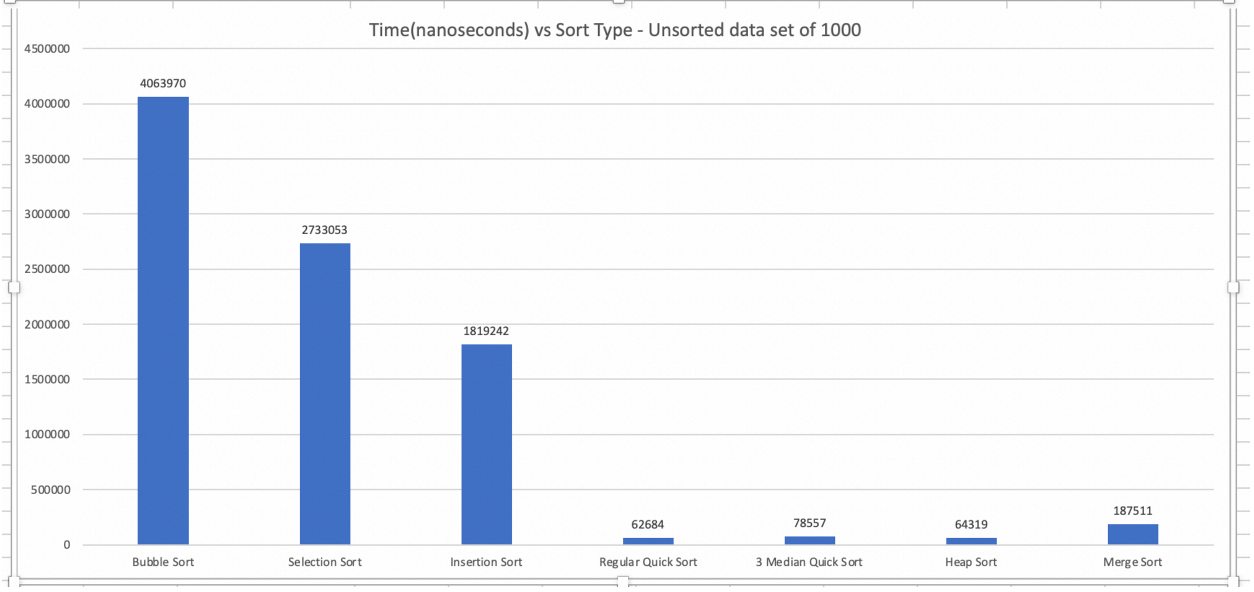
The Time vs Sort Type Analysis:

input size of 50 and 1000 is used to created randomly generated inputs as an array and used to perform all sorting operation with different scenarios:

1. With Unsorted Data:

After Providing the already sorted array as input to sorting algorithm:
the dataset of smaller size(50) : quick sort performs the best and followed by insertion sort
But as the dataset size is increased to 1000: regular quick sort takes least execution time followed by 3 median quick sort, heap sort and merge sort algorithm. All of this above mentioned sort algorithms takes far less time to execute compared to other algorithms' execution time. Bubble sort takes maximum execution time in both the scenarios.

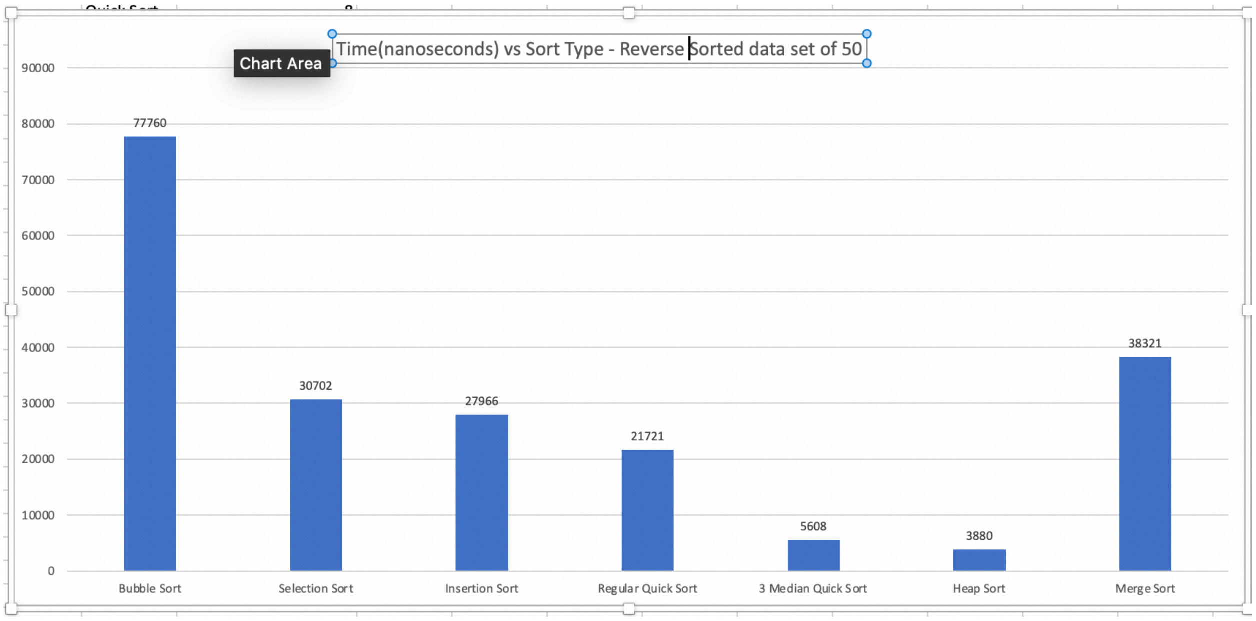


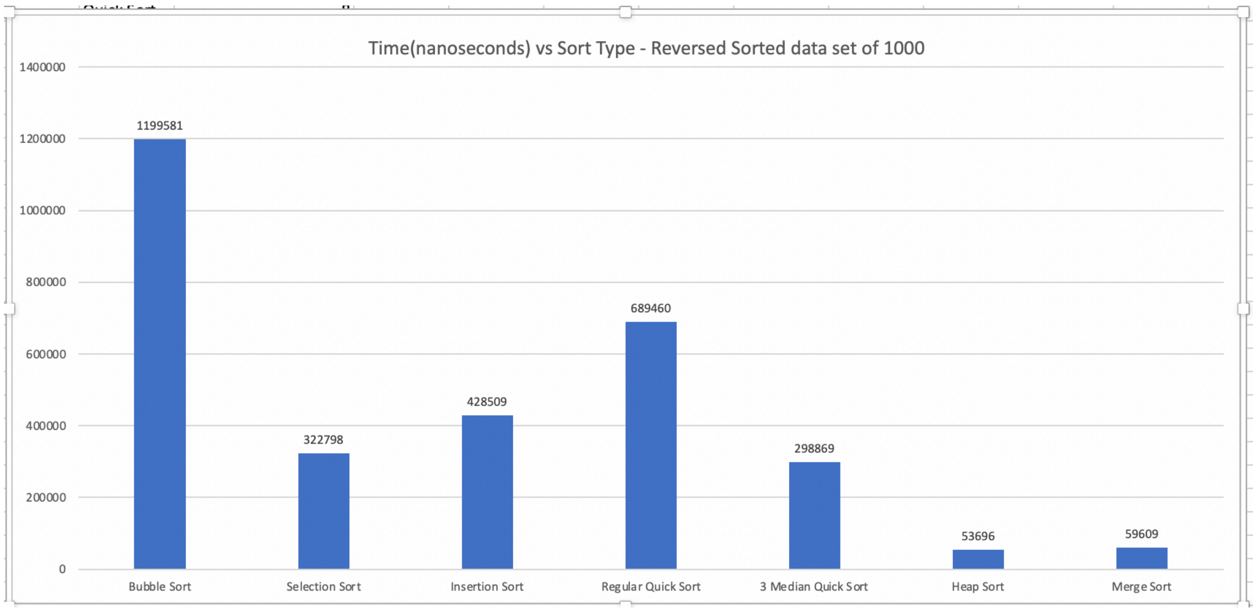


2. With Reverse Sorted Data:

After Providing the reverse sorted array as input to sorting algorithm:
Heap sort takes least execution time to sort for both smaller size and larger size array data.

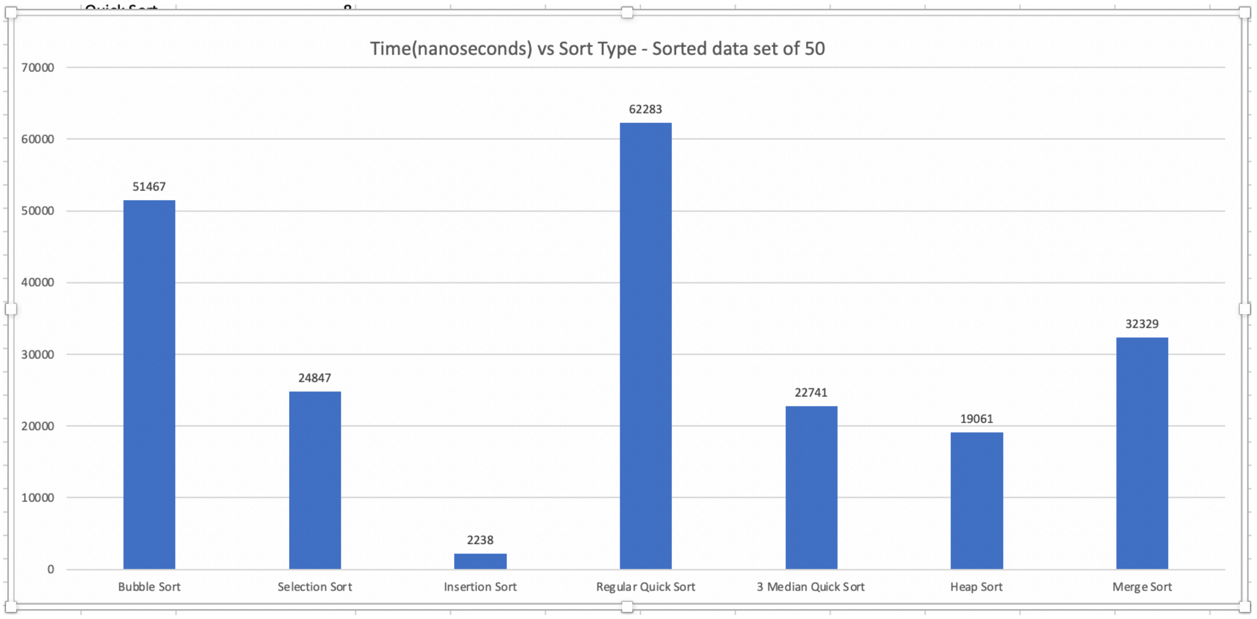
Merge sort execution time improves significantly with increase in data size compared to other sorting algorithm.
Bubble sort takes largest execution time to sort the data.

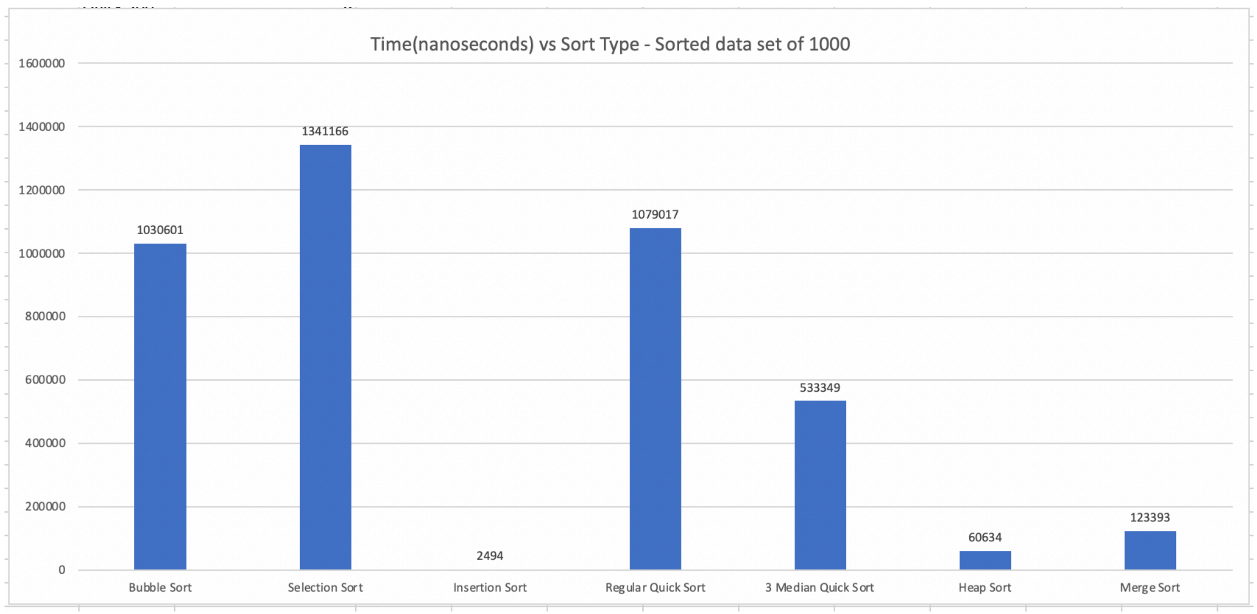




3. With Sorted Data

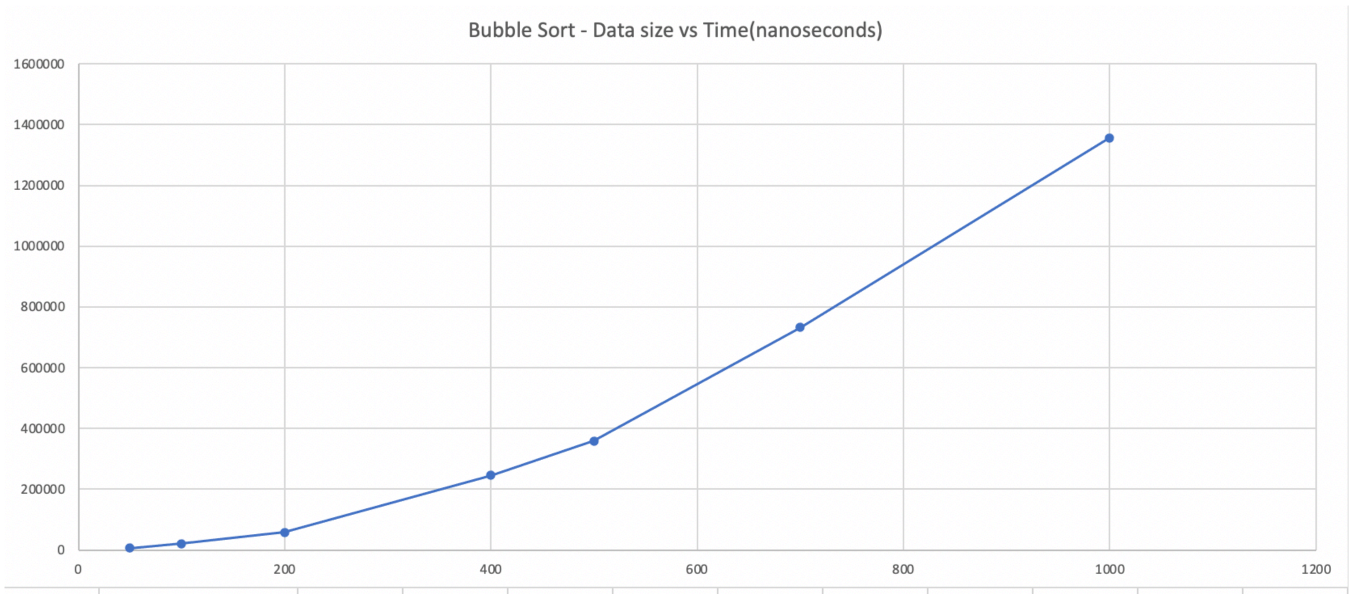
After Providing the already sorted array as input to sorting algorithm:
Regular Quick Sort with last element as pivot takes more time to sort Data compared to other sorting algorithms followed by bubble sort and insertion sort takes least amount of time to execute



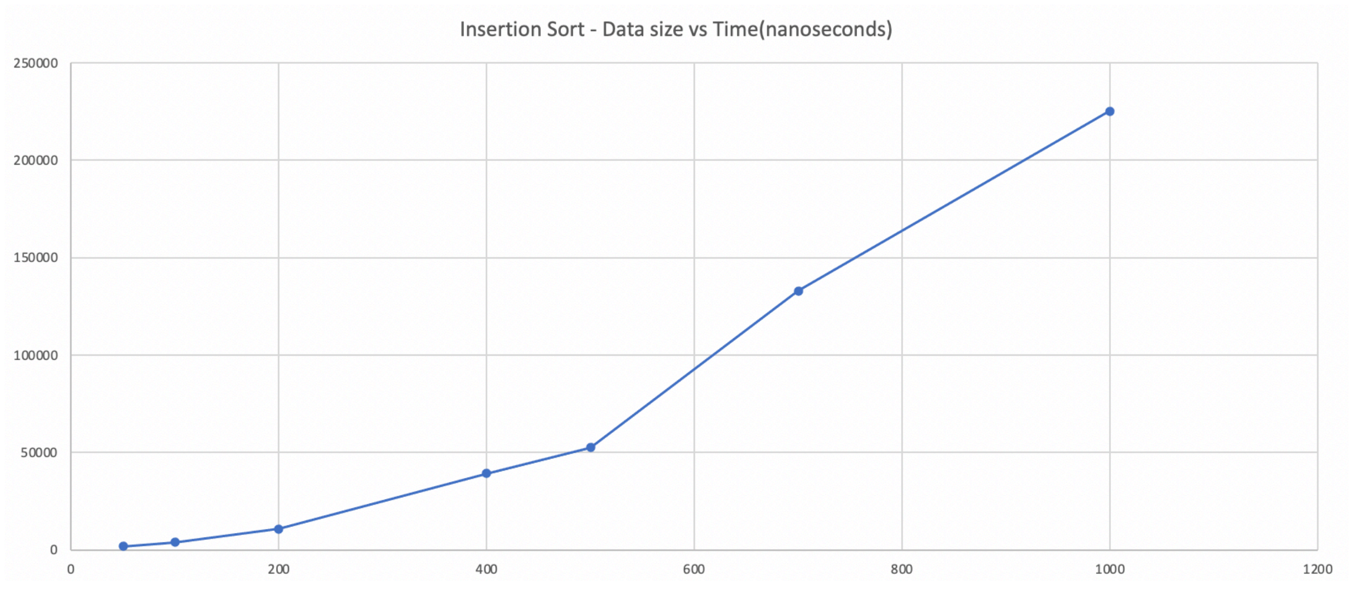


THE SORT-TYPE EXECUTION TIME VS DATA SIZE ANALYSIS:

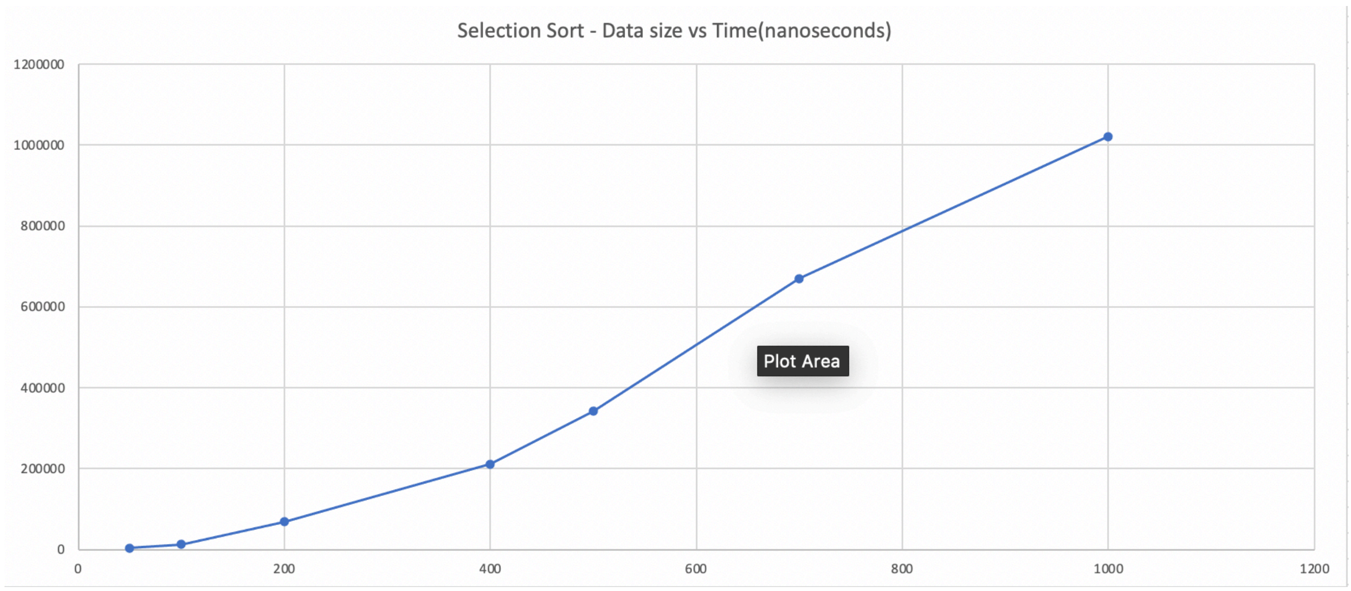
1.Bubble Sort Execution Time Vs DataSize



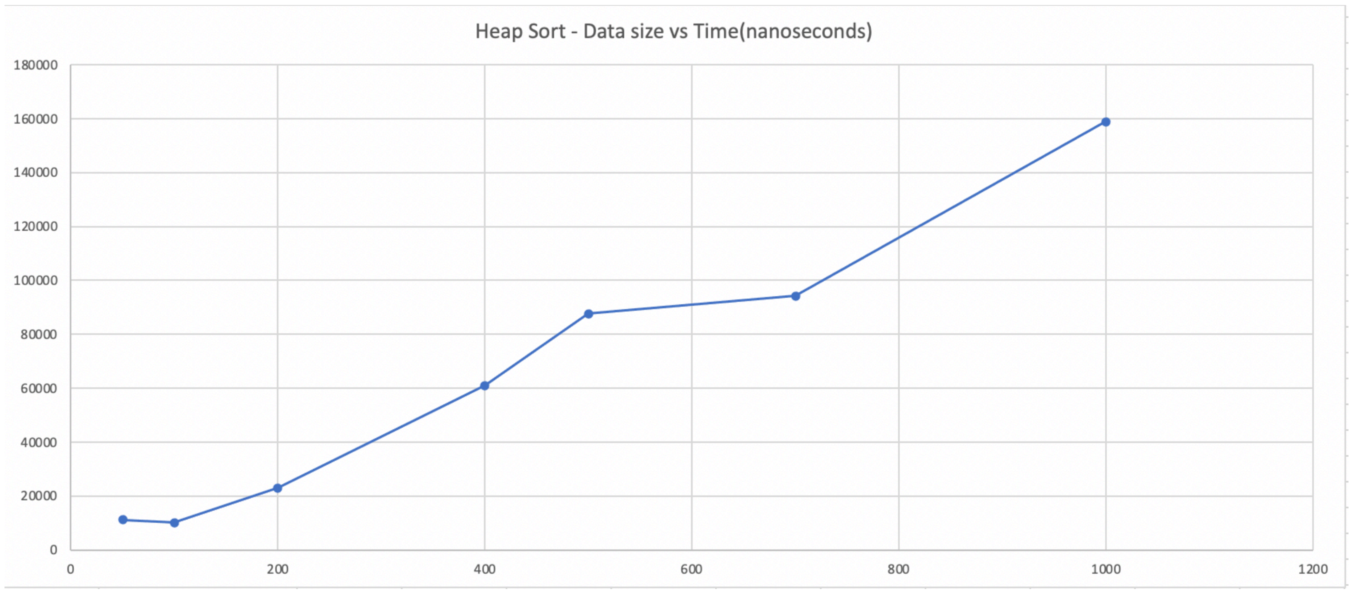
2. Insertion Sort Execution Time Vs DataSize:



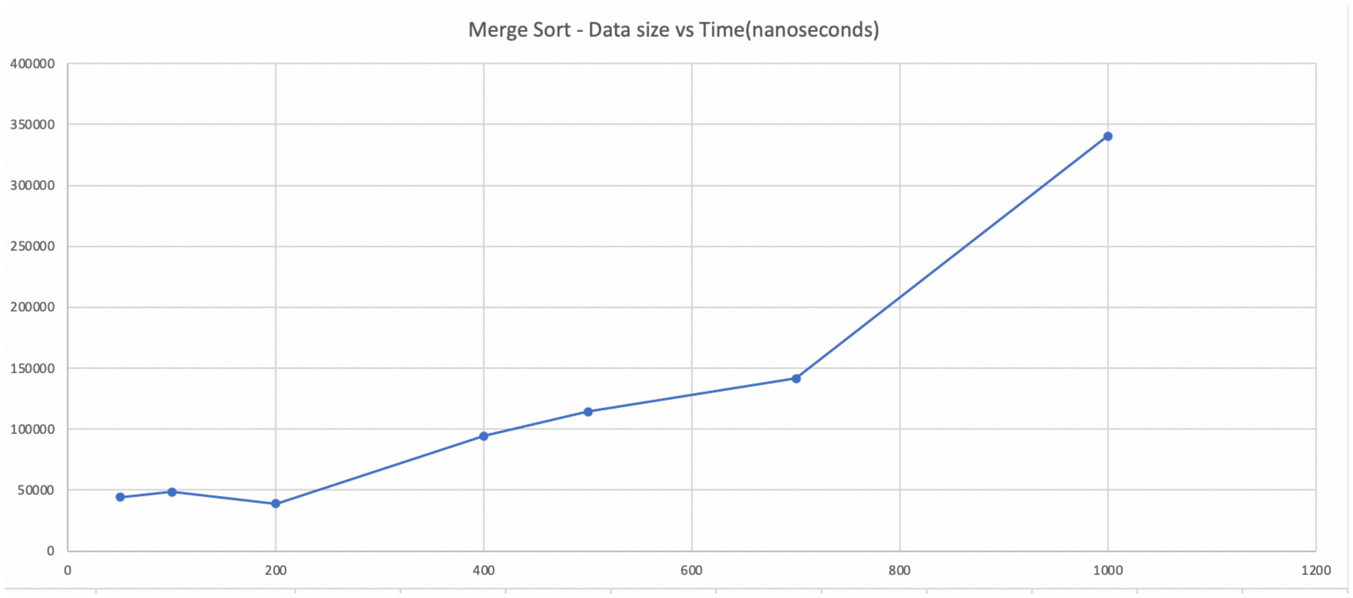
3. Selection Sort Execution Time Vs DataSize:



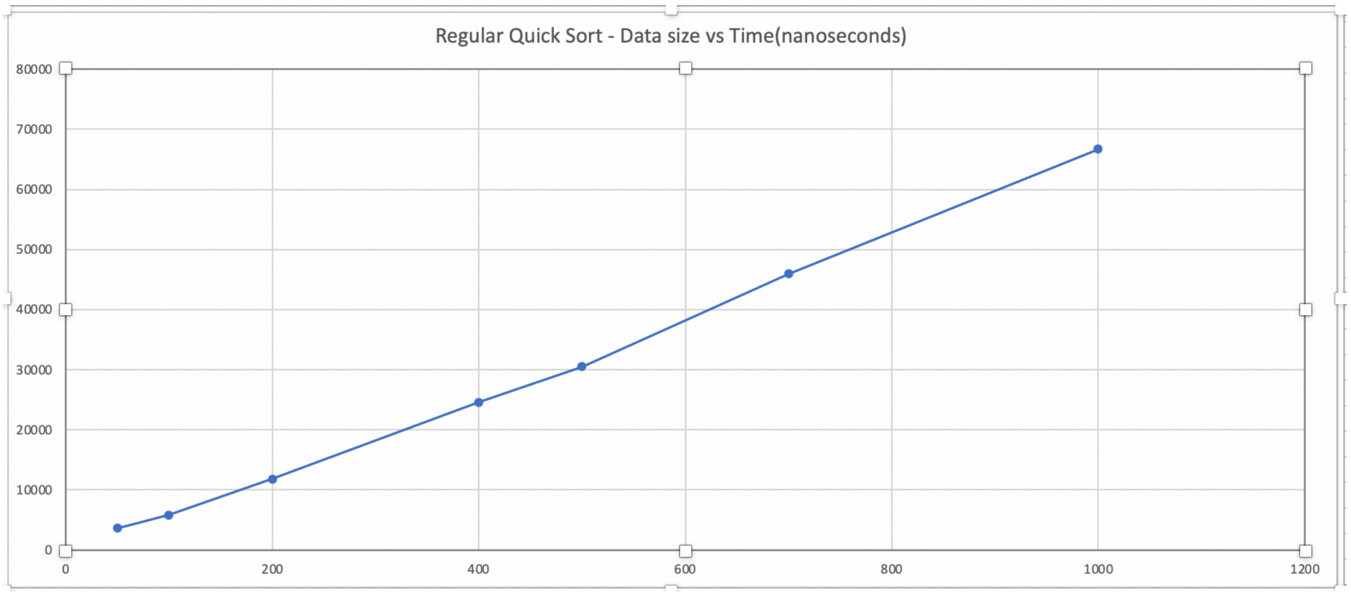
4. Heap Sort Execution Time Vs DataSize



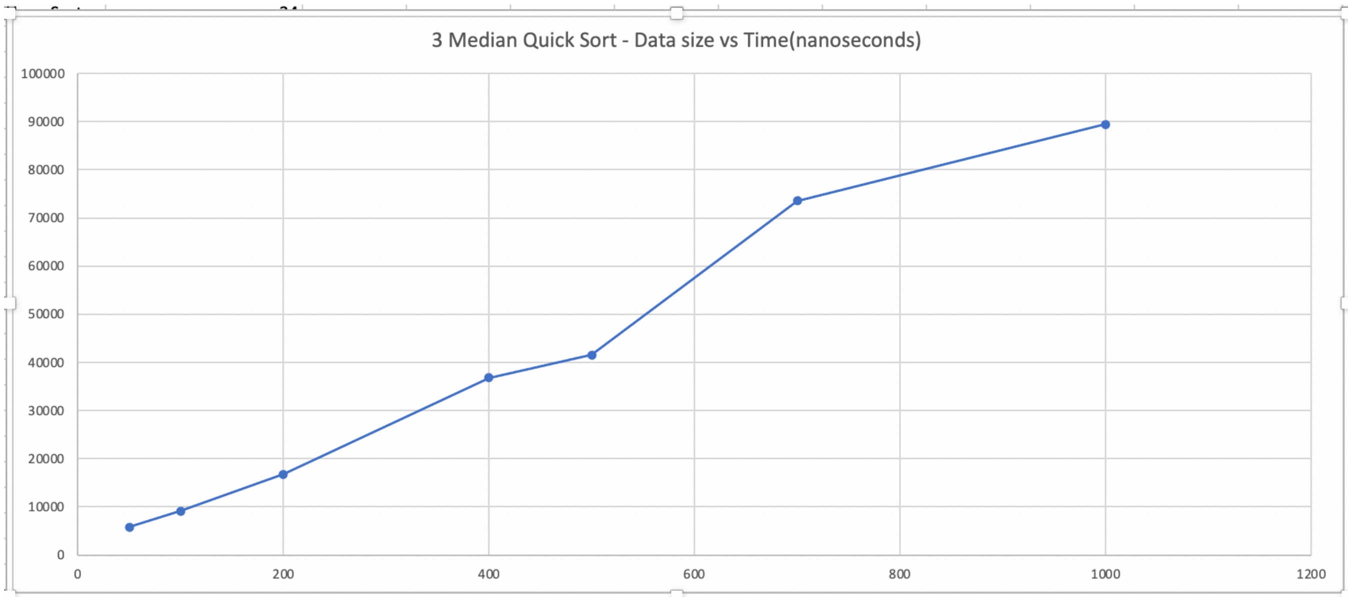
5. Merge Sort Execution Time Vs DataSize



6.1. Quick Sort Execution Time Regular vs DataSize



6.2. Quick Sort 3 Median Execution Time Vs DataSize



CONCLUSION:

From the analysis with Different Data sizes and sorting algorithm, we can conclude that as number of input size builds, time required to sort increases respectively. From over seven calculations, Quick sort provides best execution while bubble sort provides the noticeable higher execution time. Their is exponentially increase in execution time for bubble sort, insertion sort and selection sort while the execution time increments step by step for merge sort, heap sort and quick sort.Insertion Data works best with already sorted data. For smaller reversed data heap sort perform the best and and for larger sized reverse data Heap sort and Merge sort performs best. For unsorted data QuickSort performs the best and for larger size of unsorted data the performance improves for Quick Sort, 3 Median Quick Sort, Merge Sort and Heap Sort.

Time Complexity of Algorithms:

Bubble Sort	O(n^2)
Selection Sort	O(n^2)
Insertion Sort	O(n^2)
Merge Sort	O(n*logn)
Heap Sort	O(n*logn)
Quick Sort	O(n*logn)