Gebze Technical University

CSE222 – Data Structures and Algorithms

HW6 – Part2

Subject : Sorting Analyzing

Nevzat Seferoglu

171044024

*Problem Description :

Implement Quick Sort and Merge Sort algorithms. Suppose the data is stored in a linked list. You should use LinkedList class in Java.

In the book, there are methods for 7 algorithms which are

- Selection Sort
- Bubble Sort
- Insertion Sort
- Shell Sort
- Merge Sort
- Heap Sort
- Quick Sort

Compare your implementation with the methods in the book. To do this, you must create 100 (20 of each) random and 5 (1 of each) sorted arrays/linked lists with a length of ten thousand (10000), forty thousand (40000), one hundred thousand (100000), one hundred and fifty thousand (150000) and one hundred and eighty thousand (180000). Run all 9 methods for each, save run times and expected run times in excel, and draw graphs for each method. Run times and expected run times should be drawn on the same graphic.

*Solution Approach :

I implemented the all method that are given , measure the running time and expected time of all of them. There are also some properties that can be useful for analyzing each sorting algorithm.

LinkedList merge sort successfully implemented and analyzed but , LinkedList Quick sort could not be analyzed because it takes to much time even for 10K list despite the implementation.

*Note : I think , the reason why the quick sort does not work is because there are too many get() operation of the list and it takes linear time.

Test Cases :

Test ID	Scenario	Test Data	Expected Result	Actual Result	Pass	Fail
T01	Bubble Sort	10.000 40.000 100.000 150.000 180.000 random array. 1 sorted array for each size.	Sort and measure running time.	Sort and measure running time.	**Pass**	
T02	Heap Sort	10.000 40.000 100.000 150.000 180.000 random array. 1 sorted array for each size.	Sort and measure running time.	Sort and measure running time.	**Pass**	
T03	Insertion Sort	10.000 40.000 100.000 150.000 180.000 random array. 1 sorted array for each size.	Sort and measure running time.	Sort and measure running time.	**Pass**	

T04	Merge Sort	10.000 40.000 100.000 150.000 180.000 random array. 1 sorted array for each size.	Sort and measure running time.	Sort and measure running time.	**Pass**	
T05	Quick Sort	10.000 40.000 100.000 150.000 180.000 random array. 1 sorted array for each size.	Sort and measure running time.	Sort and measure running time.	**Pass**	
T06	Selection Sort	10.000 40.000 100.000 150.000 180.000 random array. 1 sorted array for each size.	Sort and measure running time.	Sort and measure running time.	**Pass**	

T07	Shell Sort	10.000 40.000 100.000 150.000 180.000 random array. 1 sorted array for each size.	Sort and measure running time.	Sort and measure running time.	**Pass**	
T08	LinkedList Merge Sort	10.000 40.000 100.000 150.000 180.000 random array. 1 sorted array for each size.	Sort and measure running time.	Sort and measure running time.	**Pass**	
T09	LinkedList Quick Sort	10.000 40.000 100.000 150.000 180.000 random array.	Sort and measure running time.	Sort and measure running time.		**Fail**

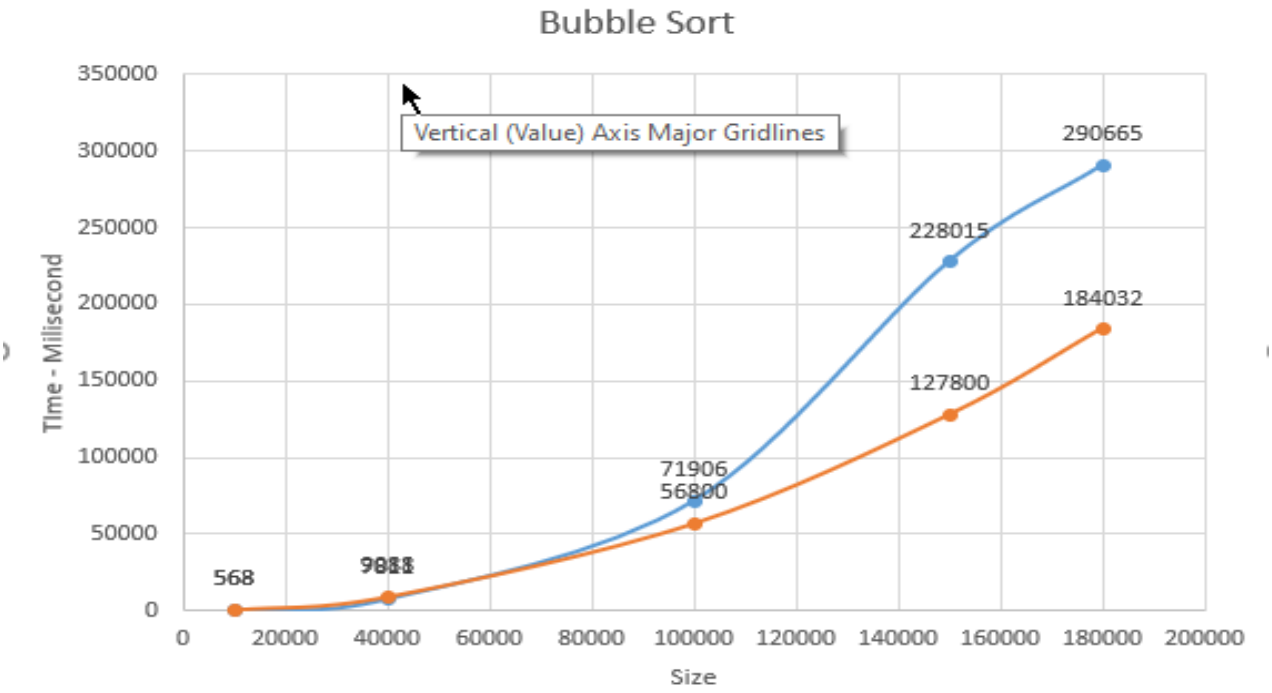
		1 sorted array for each size.				

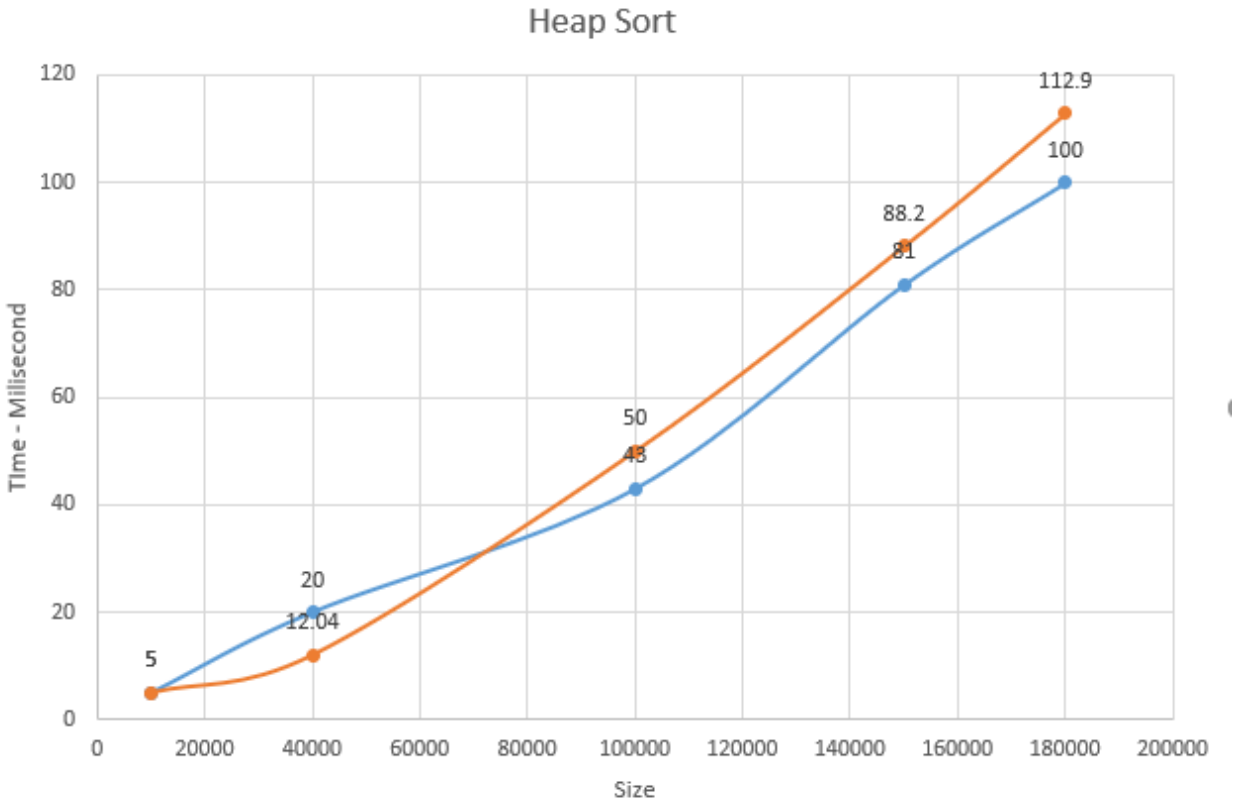
***Note** : For linked list Quick Sort , there is no result because program does not give an output.

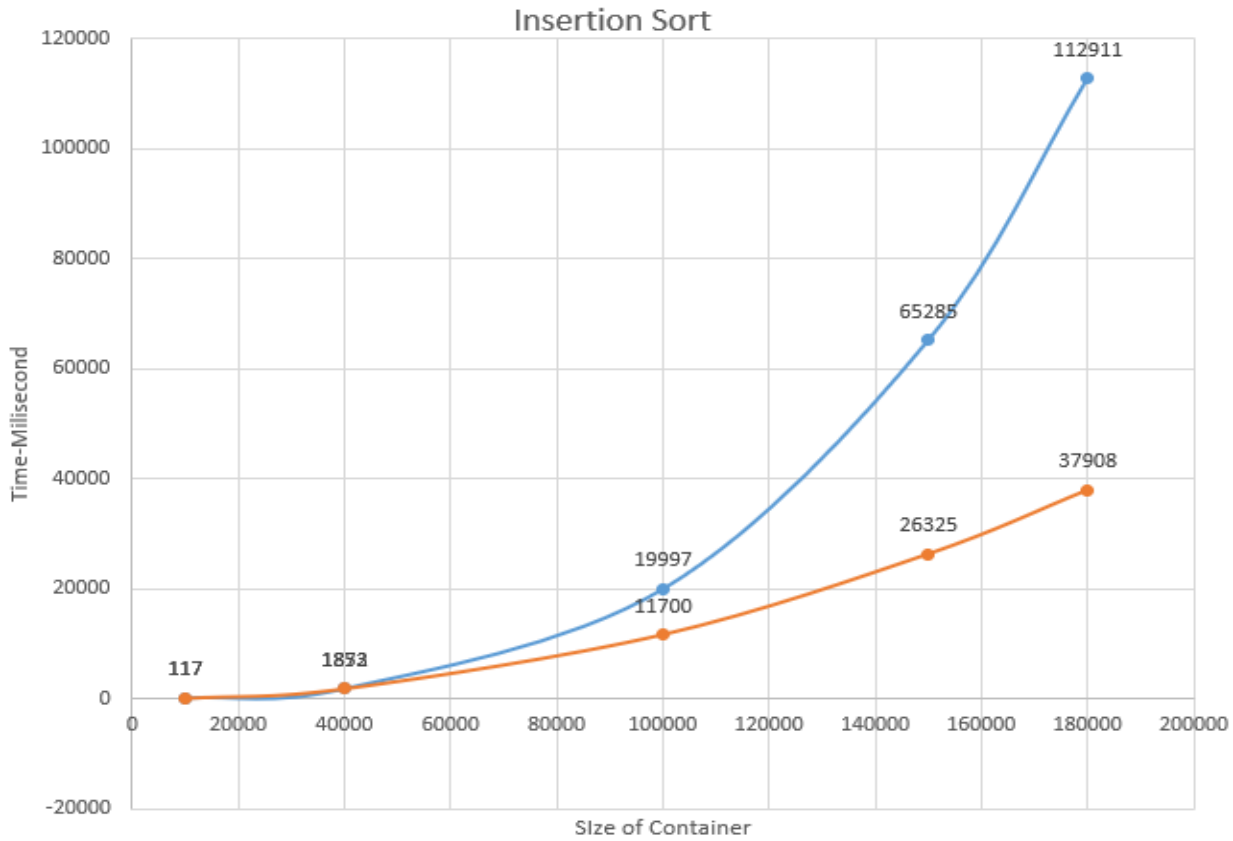
***Note :** Program should not be run for all sorting algorithm at the same time , it will take **too too too** much time to get an output.

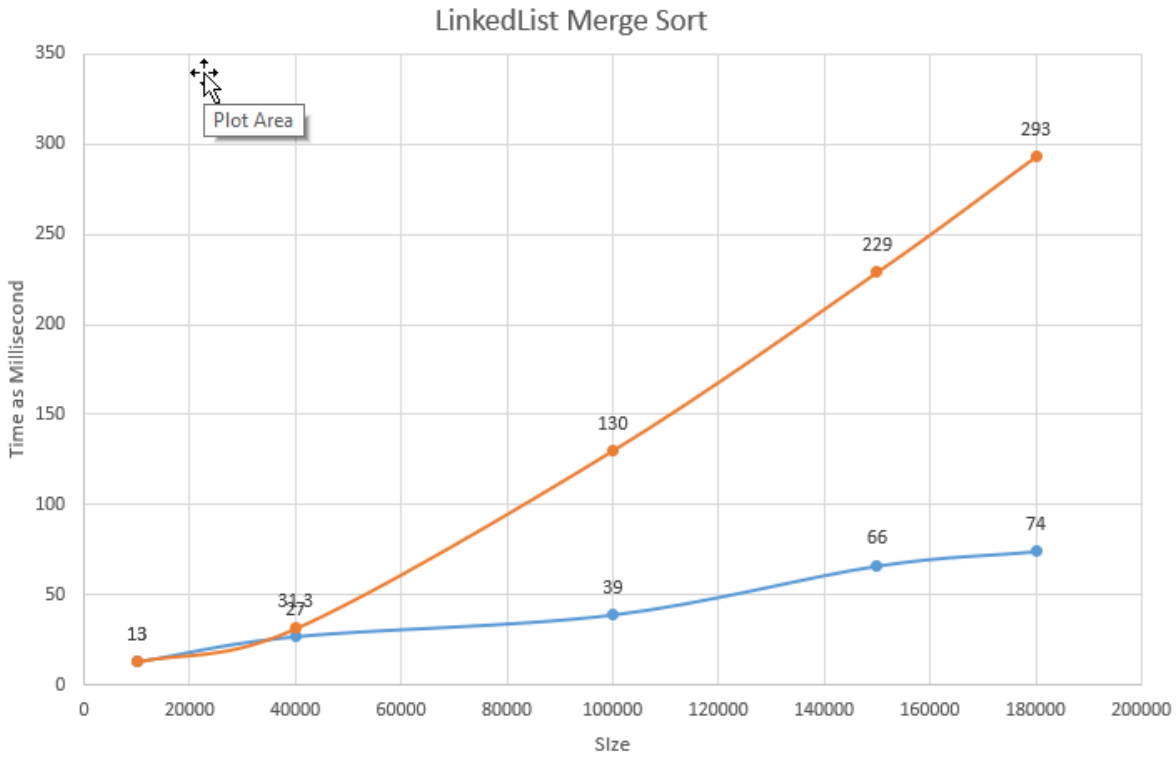
Blue Line : Representing actual running time.

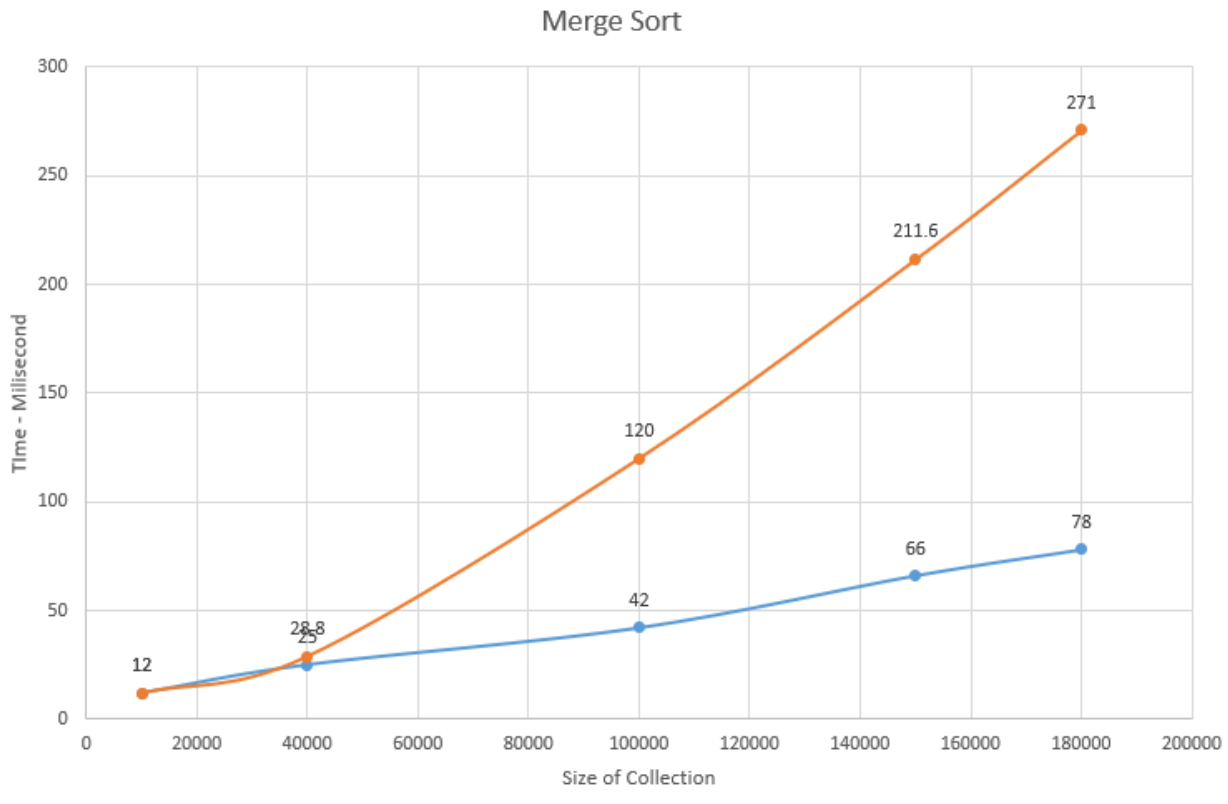
Orange Line : Representing expected running time.

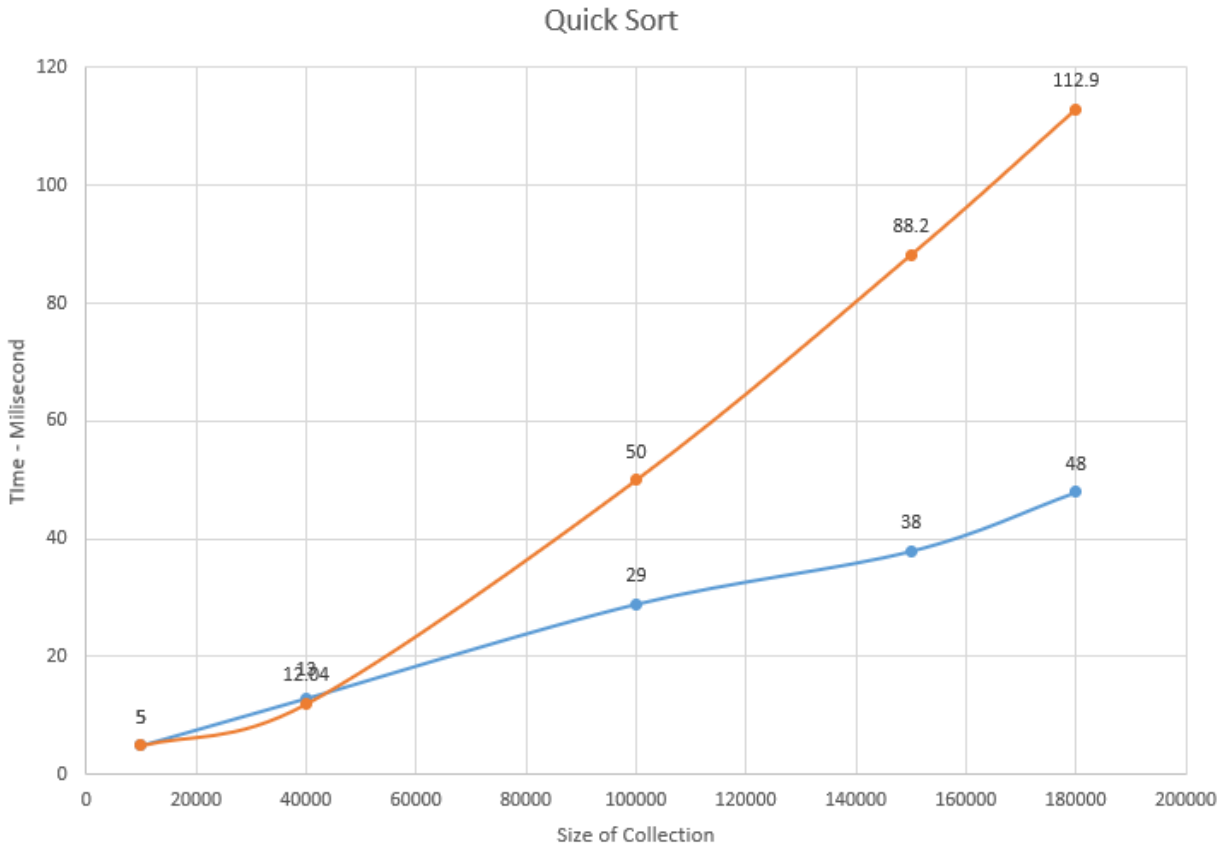


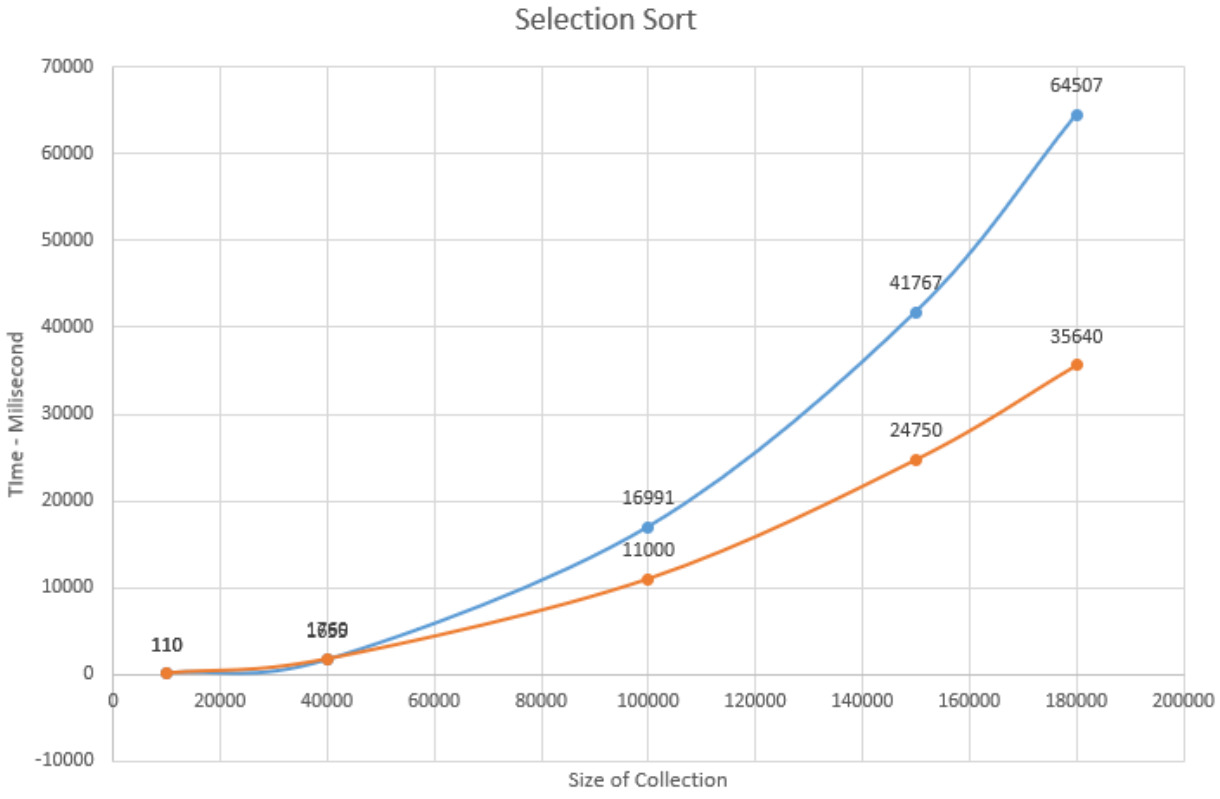

Heap Sort
Time - Milisecond
Size
120
100
80
60
40
20
0
0
20000
40000
60000
80000
100000
120000
140000
160000
180000
200000
112.9
100
88.2
81
50
43
20
12.04
5

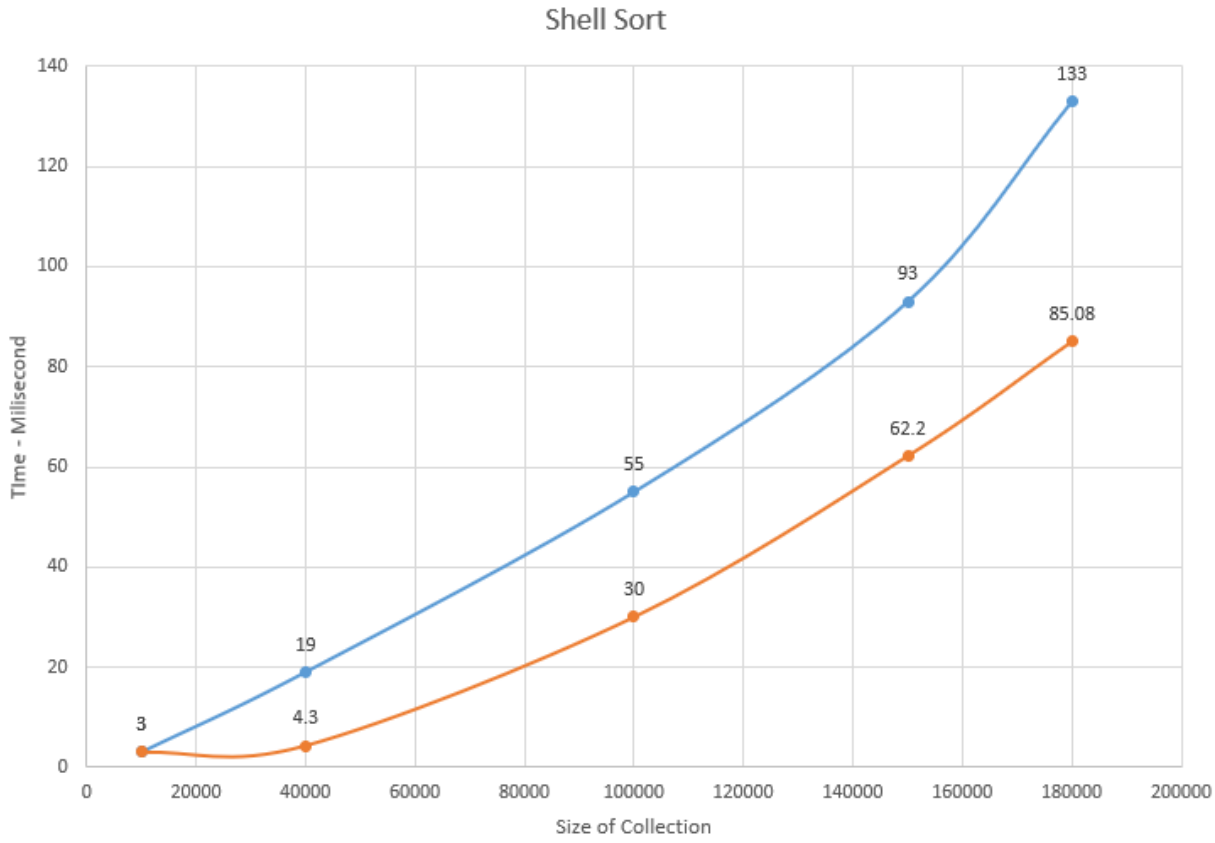

Insertion Sort
Time-Milisecond
SIze of Container
120000
100000
80000
60000
40000
20000
0
-20000
0
20000
40000
60000
80000
100000
120000
140000
160000
180000
200000
112911
65285
37908
26325
19997
11700
117


LinkedList Merge Sort
Plot Area
Time as Millisecond
Size
350
300
250
200
150
100
50
0
0
20000
40000
60000
80000
100000
120000
140000
160000
180000
200000
293
229
130
74
66
39
31.3
27
13


Merge Sort
TIme - Millisecond
Size of Collection
300
250
200
150
100
50
0
0
20000
40000
60000
80000
100000
120000
140000
160000
180000
200000
271
211.6
120
78
66
42
28.8
25
12


Quick Sort
120
100
80
60
40
20
0
TIme - Milisecond
0
20000
40000
60000
80000
100000
120000
140000
160000
180000
200000
Size of Collection
112.9
88.2
50
48
38
29
5


Selection Sort
70000
60000
50000
40000
30000
20000
10000
0
-10000
TIme - Milisecond
0
20000
40000
60000
80000
100000
120000
140000
160000
180000
200000
Size of Collection
64507
41767
35640
24750
16991
11000
110



*Class Diagram :

*Attached to assignment file.

*Running Command and Result :

*Attached to assignment file.