GTU Department of Computer Engineering

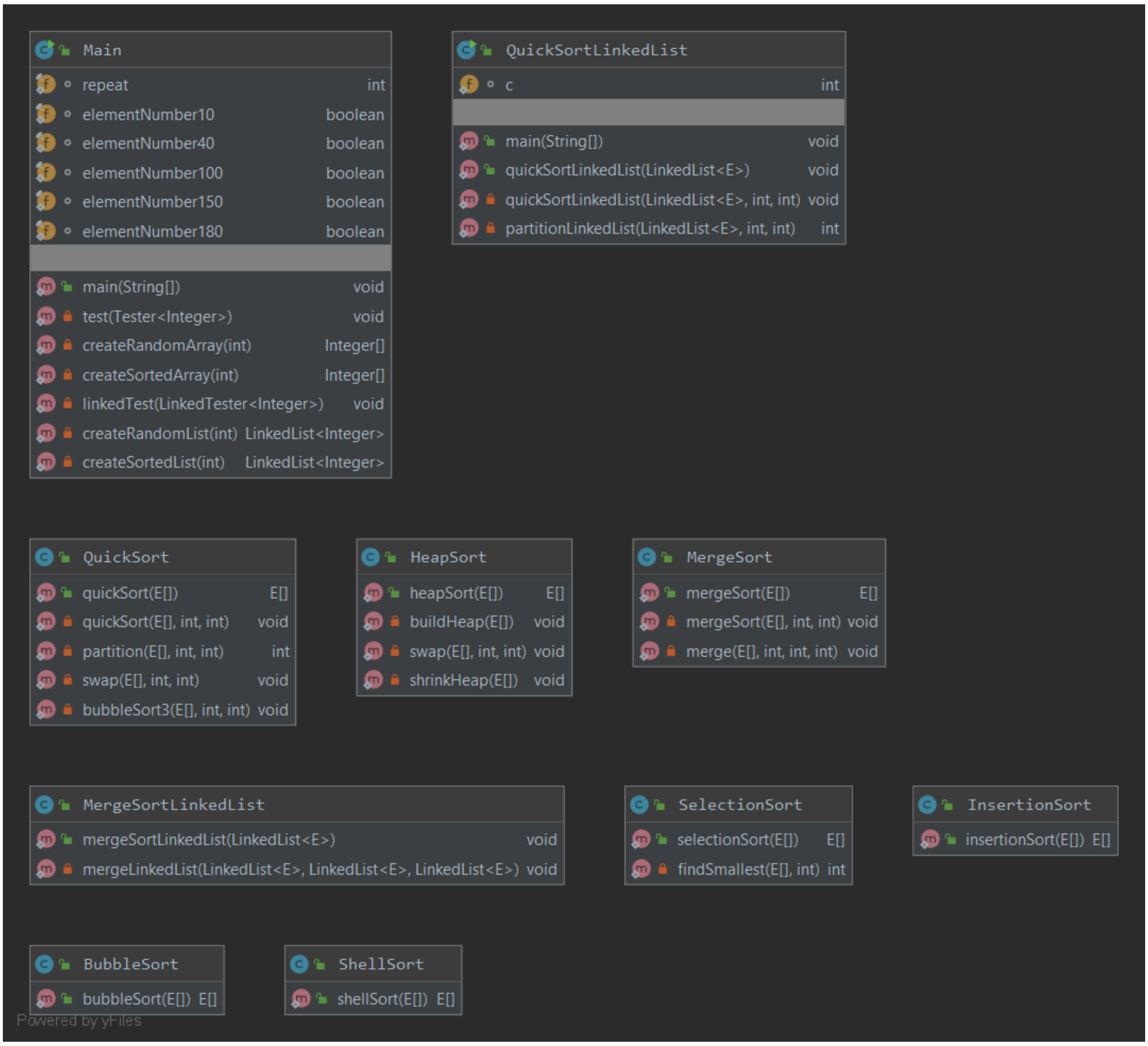
CSE 222/505 - Spring 2020

Homework 6-Q2

Report

Ali Bahar-171044066

Class Diagram


Main
repeat int
elementNumber10 boolean
elementNumber40 boolean
elementNumber100 boolean
elementNumber150 boolean
elementNumber180 boolean
main(String[]) void
test(Tester<Integer>) void
createRandomArray(int) Integer[]
createSortedArray(int) Integer[]
linkedTest(LinkedTester<Integer>) void
createRandomList(int) LinkedList<Integer>
createSortedList(int) LinkedList<Integer>
QuickSortLinkedList
c int
main(String[]) void
quickSortLinkedList(LinkedList<E>) void
quickSortLinkedList(LinkedList<E>, int, int) void
partitionLinkedList(LinkedList<E>, int, int) int
QuickSort
quickSort(E[]) E[]
quickSort(E[], int, int) void
partition(E[], int, int) int
swap(E[], int, int) void
bubbleSort3(E[], int, int) void
HeapSort
heapSort(E[]) E[]
buildHeap(E[]) void
swap(E[], int, int) void
shrinkHeap(E[]) void
MergeSort
mergeSort(E[]) E[]
mergeSort(E[], int, int) void
merge(E[], int, int, int) void
MergeSortLinkedList
mergeSortLinkedList(LinkedList<E>) void
mergeLinkedList(LinkedList<E>, LinkedList<E>, LinkedList<E>) void
SelectionSort
selectionSort(E[]) E[]
findSmallest(E[], int) int
InsertionSort
insertionSort(E[]) E[]
BubbleSort
bubbleSort(E[]) E[]
ShellSort
shellSort(E[]) E[]
Powered by yFiles

Problem Solution Approach

First I have implemented all sort algorithms as defined in book. Then I have used the ListIterator to be able to implement the merge sort with LinkedList. Then I have implement the quick sort with linked list but it didn’t work properly. Actually it worked but it have taken so much time.

Test Cases

Test ID	Scenerio	Test Data	Expected Result	Actual result	Pass /Fail
1	Test the bubble sort and take the running time.	10000 random number. 10000 ordered number. 40000 random number. 40000 ordered number. 100000 random number. 100000 ordered number. 150000 random number. 150000 ordered number. 180000 random number. 180000 ordered number.	The array is sorted.	As expected	Pass
2	Test the heap sort and take the running time.	10000 random number. 10000 ordered number. 40000 random number. 40000 ordered number. 100000 random number. 100000 ordered number. 150000 random number.	The array is sorted.	As expected	Pass

		150000 ordered number. 180000 random number. 180000 ordered number.			
3	Test the insertion sort and take the running time.	10000 random number. 10000 ordered number. 40000 random number. 40000 ordered number. 100000 random number. 100000 ordered number. 150000 random number. 150000 ordered number. 180000 random number. 180000 ordered number.	The array is sorted.	As expected	Pass
4	Test the merge sort and take the running time.	10000 random number. 10000 ordered number. 40000 random number. 40000 ordered number. 100000 random number. 100000 ordered number.	The array is sorted.	As expected	Pass

		150000 random number. 150000 ordered number. 180000 random number. 180000 ordered number.			
5	Test the quick sort and take the running time.	10000 random number. 10000 ordered number. 40000 random number. 40000 ordered number. 100000 random number. 100000 ordered number. 150000 random number. 150000 ordered number. 180000 random number. 180000 ordered number.	The array is sorted.	As expected	Pass
6	Test the selection sort and take the running time.	10000 random number. 10000 ordered number. 40000 random number. 40000 ordered number.	The array is sorted.	As expected	Pass

		100000 random number. 100000 ordered number. 150000 random number. 150000 ordered number. 180000 random number. 180000 ordered number.			
7	Test the shell sort and take the running time.	10000 random number. 10000 ordered number. 40000 random number. 40000 ordered number. 100000 random number. 100000 ordered number. 150000 random number. 150000 ordered number. 180000 random number. 180000 ordered number.	The array is sorted.	As expected	Pass
8	Test the merge sort which implented with linked list and take the running time.	10000 random number. 10000 ordered number. 40000 random number.	The array is sorted.	As expected	Pass

		40000 ordered number. 100000 random number. 100000 ordered number. 150000 random number. 150000 ordered number. 180000 random number. 180000 ordered number.			
9	Test the quşck sort which implented with linked list and take the running time.	10000 random number. 10000 ordered number. 40000 random number. 40000 ordered number. 100000 random number. 100000 ordered number. 150000 random number. 150000 ordered number. 180000 random number. 180000 ordered number.	The array is sorted. But it will take too much time when the problem size is getting bigger.	it will take too much time when the problem size is getting bigger.	Fail

Running command and results

```
Bubble Sort ...
10000 element has been sorted in 1798ms
average time 1798ms
10000 sorted array test 2ms
40000 element has been sorted in 27150ms
average time 27150ms
40000 sorted array test 4ms
100000 element has been sorted in 198950ms
average time 198950ms
100000 sorted array test 7ms
150000 element has been sorted in 576285ms
average time 576285ms
150000 sorted array test 10ms
180000 element has been sorted in 805614ms
average time 805614ms
180000 sorted array test 11ms
Heap Sort ...
10000 element has been sorted in 28ms
average time 28ms
10000 sorted array test 24ms
40000 element has been sorted in 54ms
average time 54ms
40000 sorted array test 25ms
100000 element has been sorted in 93ms
average time 93ms
100000 sorted array test 106ms
150000 element has been sorted in 157ms
average time 157ms
150000 sorted array test 202ms
```

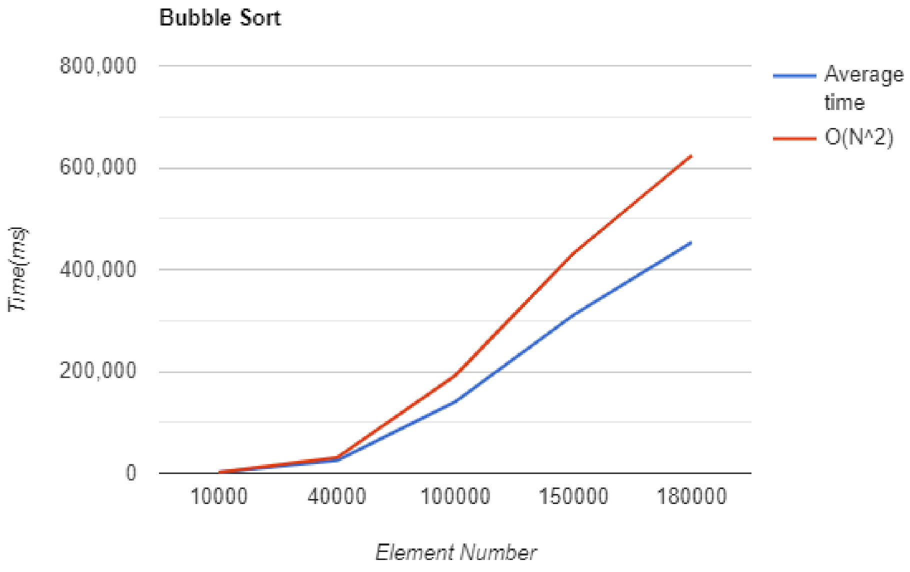
```
180000 element has been sorted in 180ms
average time 180ms
180000 sorted array test 405ms
insertion Sort ...
10000 element has been sorted in 443ms
average time 443ms
10000 sorted array test 1ms
40000 element has been sorted in 6702ms
average time 6702ms
40000 sorted array test 1ms
100000 element has been sorted in 59926ms
average time 59926ms
100000 sorted array test 3ms
150000 element has been sorted in 157014ms
average time 157014ms
150000 sorted array test 12ms
180000 element has been sorted in 252888ms
average time 252888ms
180000 sorted array test 83ms
Merge Sort ...
10000 element has been sorted in 14ms
average time 14ms
10000 sorted array test 10ms
40000 element has been sorted in 122ms
average time 122ms
40000 sorted array test 158ms
100000 element has been sorted in 421ms
average time 421ms
100000 sorted array test 93ms
150000 element has been sorted in 158ms
average time 158ms
```

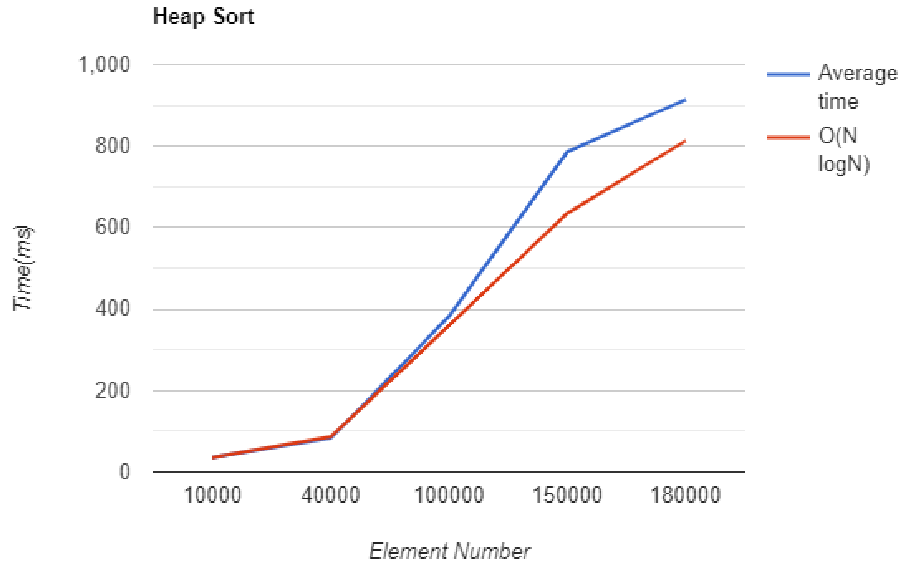
```
150000 sorted array test 149ms
180000 element has been sorted in 300ms
average time 300ms
180000 sorted array test 176ms
Quick Sort ...
10000 element has been sorted in 15ms
average time 15ms
10000 sorted array test 4ms
40000 element has been sorted in 47ms
average time 47ms
40000 sorted array test 19ms
100000 element has been sorted in 151ms
average time 151ms
100000 sorted array test 63ms
150000 element has been sorted in 254ms
average time 254ms
150000 sorted array test 217ms
180000 element has been sorted in 177ms
average time 177ms
180000 sorted array test 40ms
selection Sort ...
10000 element has been sorted in 282ms
average time 282ms
10000 sorted array test 255ms
40000 element has been sorted in 4400ms
average time 4400ms
40000 sorted array test 3898ms
100000 element has been sorted in 27323ms
average time 27323ms
100000 sorted array test 16542ms
150000 element has been sorted in 82507ms
```

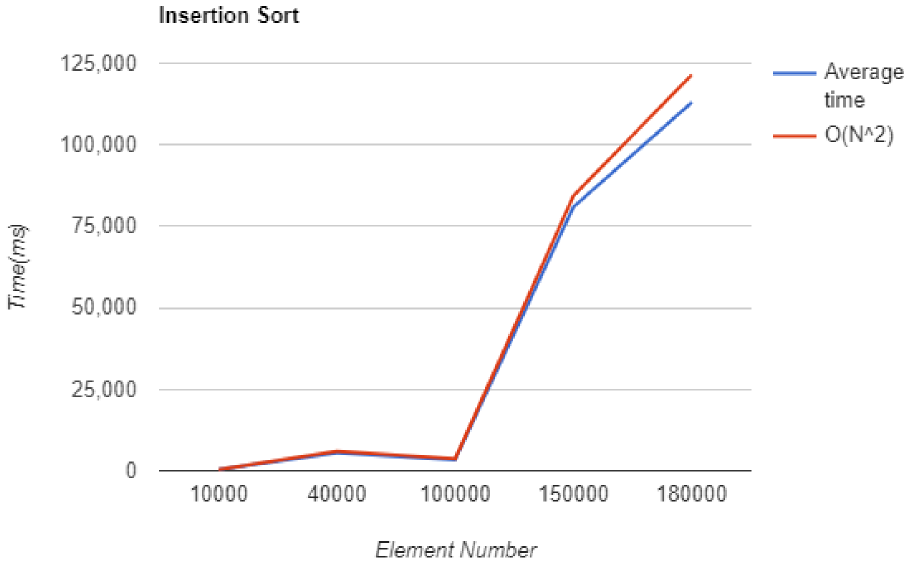
```
average time 82507ms
150000 sorted array test 47028ms
180000 element has been sorted in 110913ms
average time 110913ms
180000 sorted array test 71740ms
shell Sort ...
10000 element has been sorted in 26ms
average time 26ms
10000 sorted array test 7ms
40000 element has been sorted in 95ms
average time 95ms
40000 sorted array test 6ms
100000 element has been sorted in 165ms
average time 165ms
100000 sorted array test 147ms
150000 element has been sorted in 302ms
average time 302ms
150000 sorted array test 89ms
180000 element has been sorted in 352ms
average time 352ms
180000 sorted array test 193ms
merge Sort (Linked List) ...
10000 element has been sorted in 324ms
average time 324ms
10000 sorted array test 85ms
40000 element has been sorted in 3373ms
average time 3373ms
40000 sorted array test 587ms
100000 element has been sorted in 19616ms
average time 19616ms
100000 sorted array test 4902ms
```

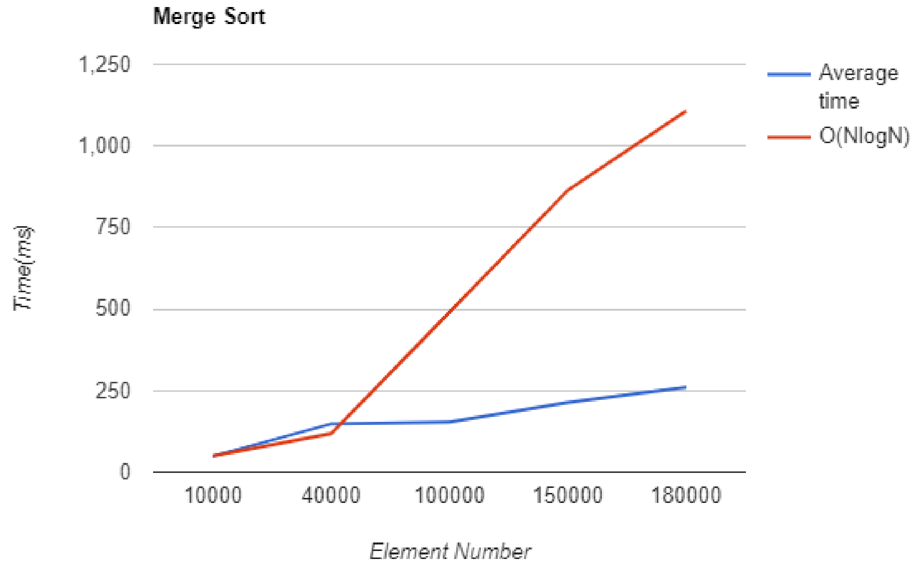
```
150000 element has been sorted in 45822ms
average time 45822ms
150000 sorted array test 12214ms
180000 element has been sorted in 72614ms
average time 72614ms
180000 sorted array test 13589ms
Quick Sort (Linked List) ...
10000 element has been sorted in 3133ms
average time 3133ms
10000 sorted array test 502133ms
40000 element has been sorted in 66681ms
average time 66681ms
```

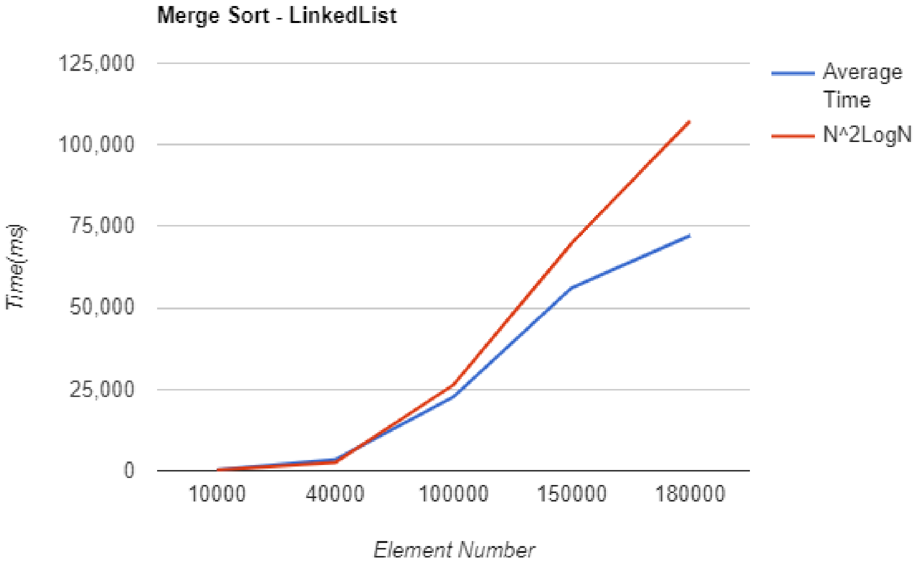
Line-Chart

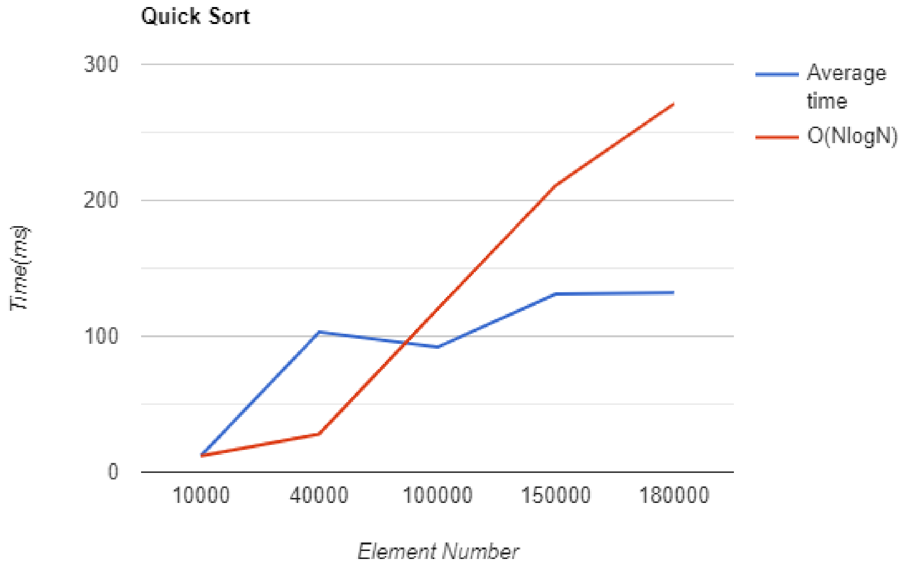


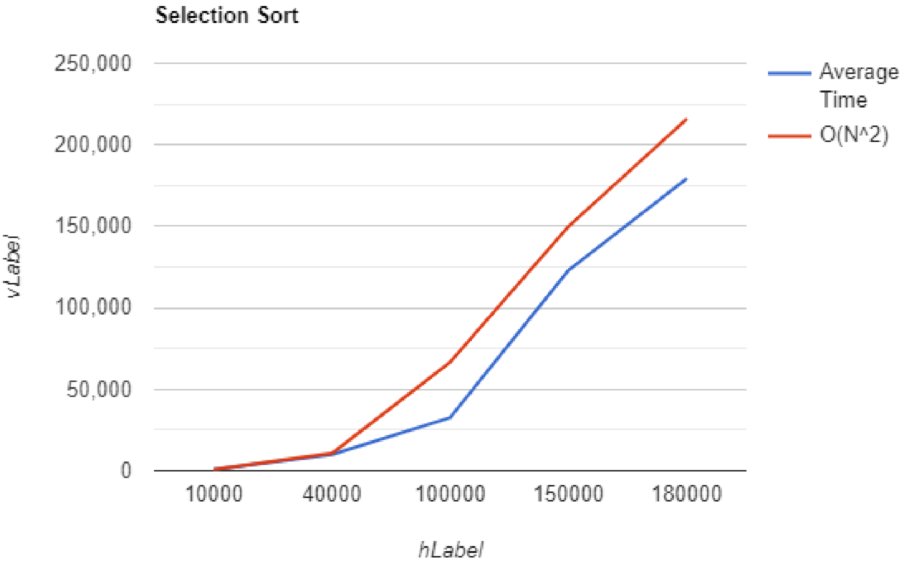

Heap Sort
1,000
800
600
400
200
0
Time(ms)
10000
40000
100000
150000
180000
Element Number
Average time
O(N logN)

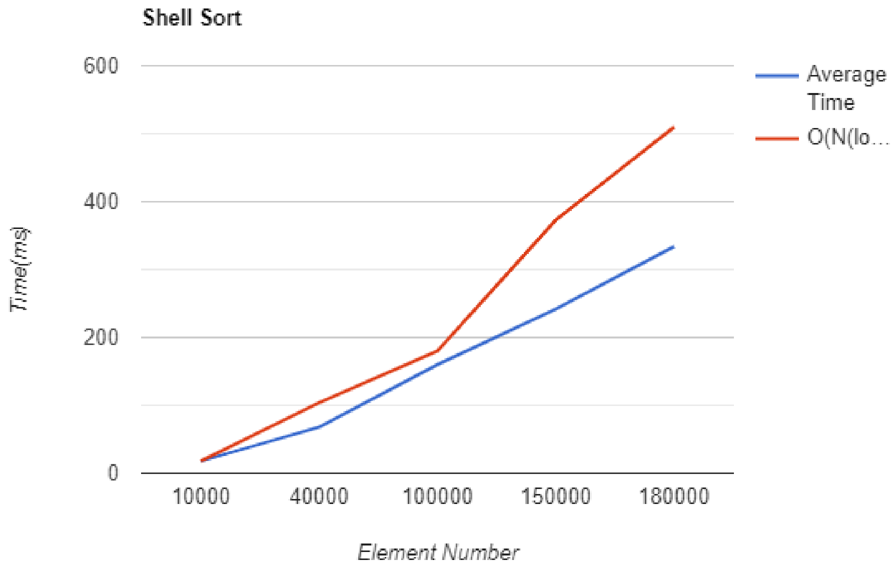

Insertion Sort
125,000
100,000
75,000
50,000
25,000
0
Time(ms)
10000
40000
100000
150000
180000
Element Number
Average time
O(N^2)


Merge Sort
1,250
1,000
750
500
250
0
Time(ms)
10000
40000
100000
150000
180000
Element Number
Average time
O(NlogN)


Merge Sort - LinkedList
125,000
100,000
75,000
50,000
25,000
0
Time(ms)
10000
40000
100000
150000
180000
Element Number
Average Time
N^2LogN


Quick Sort
300
200
100
0
Time(ms)
10000
40000
100000
150000
180000
Element Number
Average time
O(NlogN)


Selection Sort
250,000
200,000
150,000
100,000
50,000
0
vLabel
10000
40000
100000
150000
180000
hLabel
Average Time
O(N^2)



There is no chart of quick sort(with linked-list) because I couldn't take the run time.