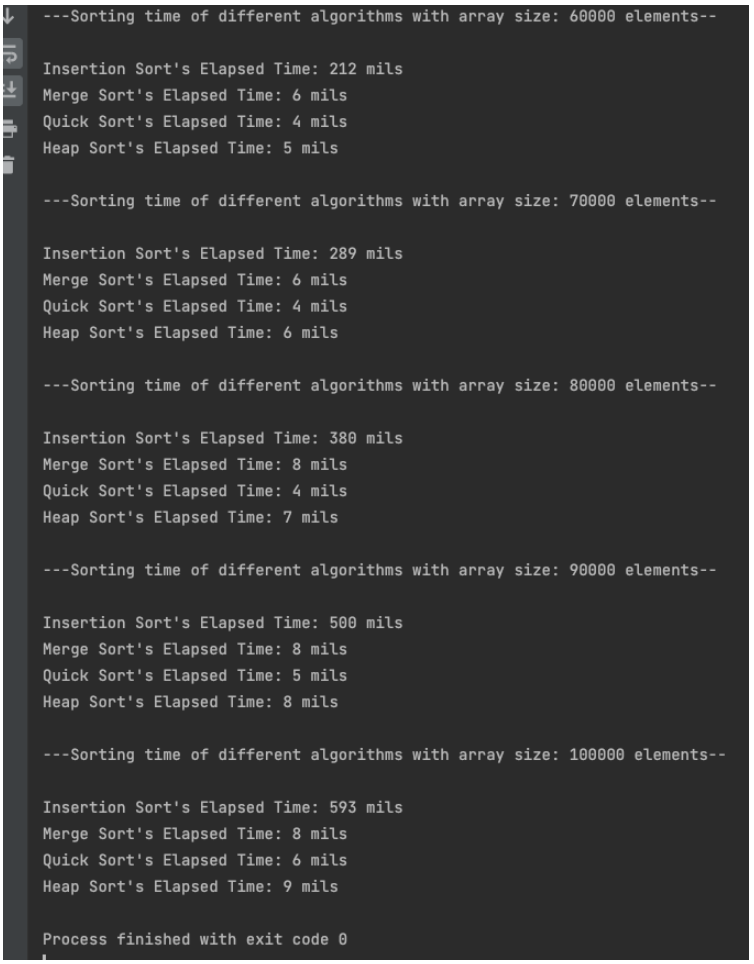
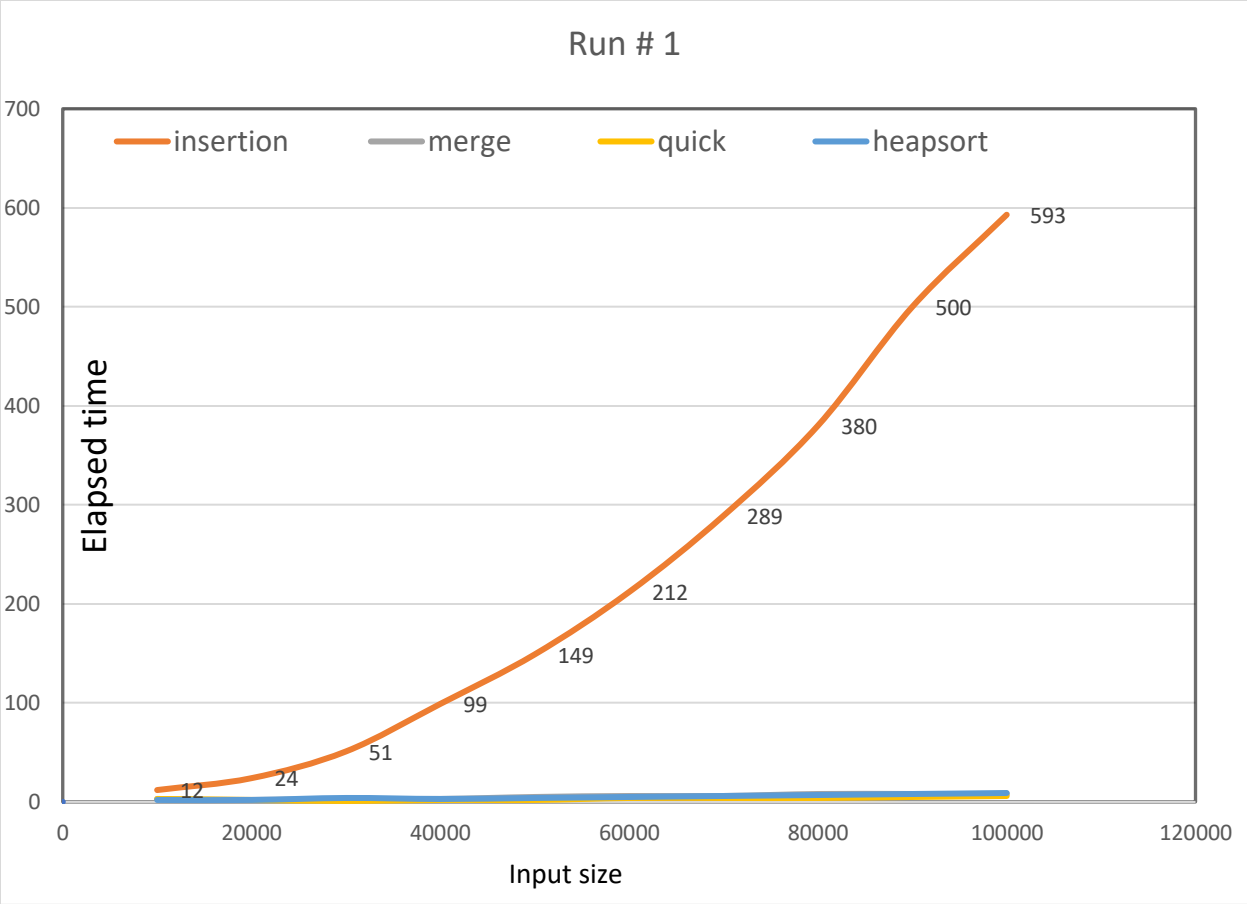
Run first time:

n / *Algorithm*	10000	20000	30000	40000	50000	60000	70000	80000	90000	100000
insertion	12	24	51	99	149	212	289	380	500	593
merge	3	2	3	3	5	6	6	8	8	8
quick	3	2	1	2	2	4	4	4	5	6
heapsort	2	2	4	3	4	5	6	7	8	9



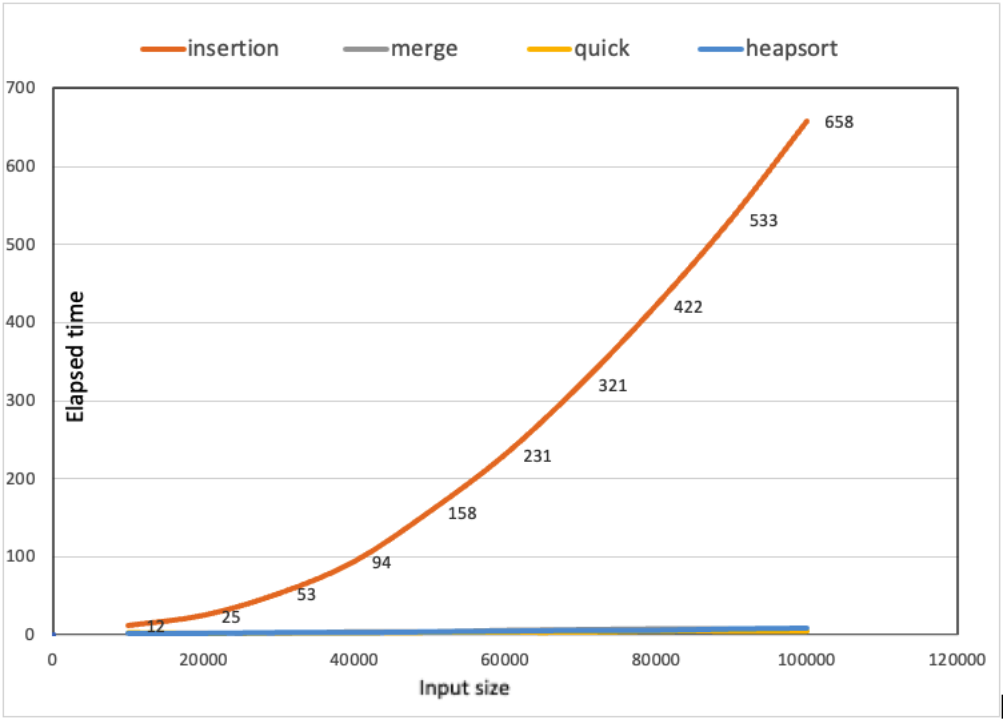

Run # 1
insertion
merge
quick
heapsort
Elapsed time
Input size
700
600
500
400
300
200
100
0
0
20000
40000
60000
80000
100000
120000
12
24
51
99
149
212
289
380
500
593

Second time:

n / *Algorithm*	10000	20000	30000	40000	50000	60000	70000	80000	90000	100000
insertion	12	25	53	94	158	231	321	422	533	658
merge	2	2	2	4	4	6	7	8	8	9
quick	2	1	2	2	3	3	3	5	5	5
heapsort	2	2	3	3	4	5	6	6	8	9

```
---Sorting time of different algorithms with array size: 60000 elements--

Insertion Sort's Elapsed Time: 231 mils
Merge Sort's Elapsed Time: 6 mils
Quick Sort's Elapsed Time: 3 mils
Heap Sort's Elapsed Time: 5 mils

---Sorting time of different algorithms with array size: 70000 elements--

Insertion Sort's Elapsed Time: 321 mils
Merge Sort's Elapsed Time: 7 mils
Quick Sort's Elapsed Time: 3 mils
Heap Sort's Elapsed Time: 6 mils

---Sorting time of different algorithms with array size: 80000 elements--

Insertion Sort's Elapsed Time: 422 mils
Merge Sort's Elapsed Time: 8 mils
Quick Sort's Elapsed Time: 5 mils
Heap Sort's Elapsed Time: 6 mils

---Sorting time of different algorithms with array size: 90000 elements--

Insertion Sort's Elapsed Time: 533 mils
Merge Sort's Elapsed Time: 8 mils
Quick Sort's Elapsed Time: 5 mils
Heap Sort's Elapsed Time: 8 mils

---Sorting time of different algorithms with array size: 100000 elements--

Insertion Sort's Elapsed Time: 658 mils
Merge Sort's Elapsed Time: 9 mils
Quick Sort's Elapsed Time: 5 mils
Heap Sort's Elapsed Time: 9 mils
```

```
---Sorting time of different algorithms with array size: 10000 elements--

Insertion Sort's Elapsed Time: 12 mils
Merge Sort's Elapsed Time: 2 mils
Quick Sort's Elapsed Time: 2 mils
Heap Sort's Elapsed Time: 2 mils

---Sorting time of different algorithms with array size: 20000 elements--

Insertion Sort's Elapsed Time: 25 mils
Merge Sort's Elapsed Time: 2 mils
Quick Sort's Elapsed Time: 1 mils
Heap Sort's Elapsed Time: 2 mils

---Sorting time of different algorithms with array size: 30000 elements--

Insertion Sort's Elapsed Time: 53 mils
Merge Sort's Elapsed Time: 2 mils
Quick Sort's Elapsed Time: 2 mils
Heap Sort's Elapsed Time: 3 mils

---Sorting time of different algorithms with array size: 40000 elements--

Insertion Sort's Elapsed Time: 94 mils
Merge Sort's Elapsed Time: 4 mils
Quick Sort's Elapsed Time: 2 mils
Heap Sort's Elapsed Time: 3 mils

---Sorting time of different algorithms with array size: 50000 elements--

Insertion Sort's Elapsed Time: 158 mils
Merge Sort's Elapsed Time: 4 mils
Quick Sort's Elapsed Time: 3 mils
Heap Sort's Elapsed Time: 4 mils
```



Discussion:

After running the code for 2 time, it is noticeable that insertion sort increases significantly when we increase input size and it takes longest time than merge sort, quick sort and heap sort and those sorting algorithms not quite stable and increase elapsed time not much when we increase the input size. Another factor which contributes to this different is insertion sort time complexity is O(n^2) versus O(nlogn) in the merge sort, quick sort, and heap sort. I also notice that the quick sort is slightly perform better than merge sort and heap sort and it is the fastest elapsed time compared to all sorting algorithms above. Merge sort and heap sort are quite the same.