Project 1: Comparison-based Sorting Algorithms

ITCS 6114 – Algorithms and Data Structures

By Ria Banerjee & TJ Bah

Implement the following Sorting Algorithms:

Algorithm	Time Complexity	File Name
Insertion Sort	$O(n^2)$	insertionSort.zip
Merge Sort	O(n log n)	mergeSort.zip
Heapsort	O(n log n)	heapSort.zip
In-Place Quicksort	O(n log n)	inPlaceQuickSort.zip
Modified Quicksort	O(n log n)	modifiedQuickSort.zip

Notes and Observations:

Insertion Sort:

Generally has the worst time complexity but not in all cases. In the special case that the input array is already sorted, the insertion sort is the top performer. This may have implications on nearly sorted arrays as well.

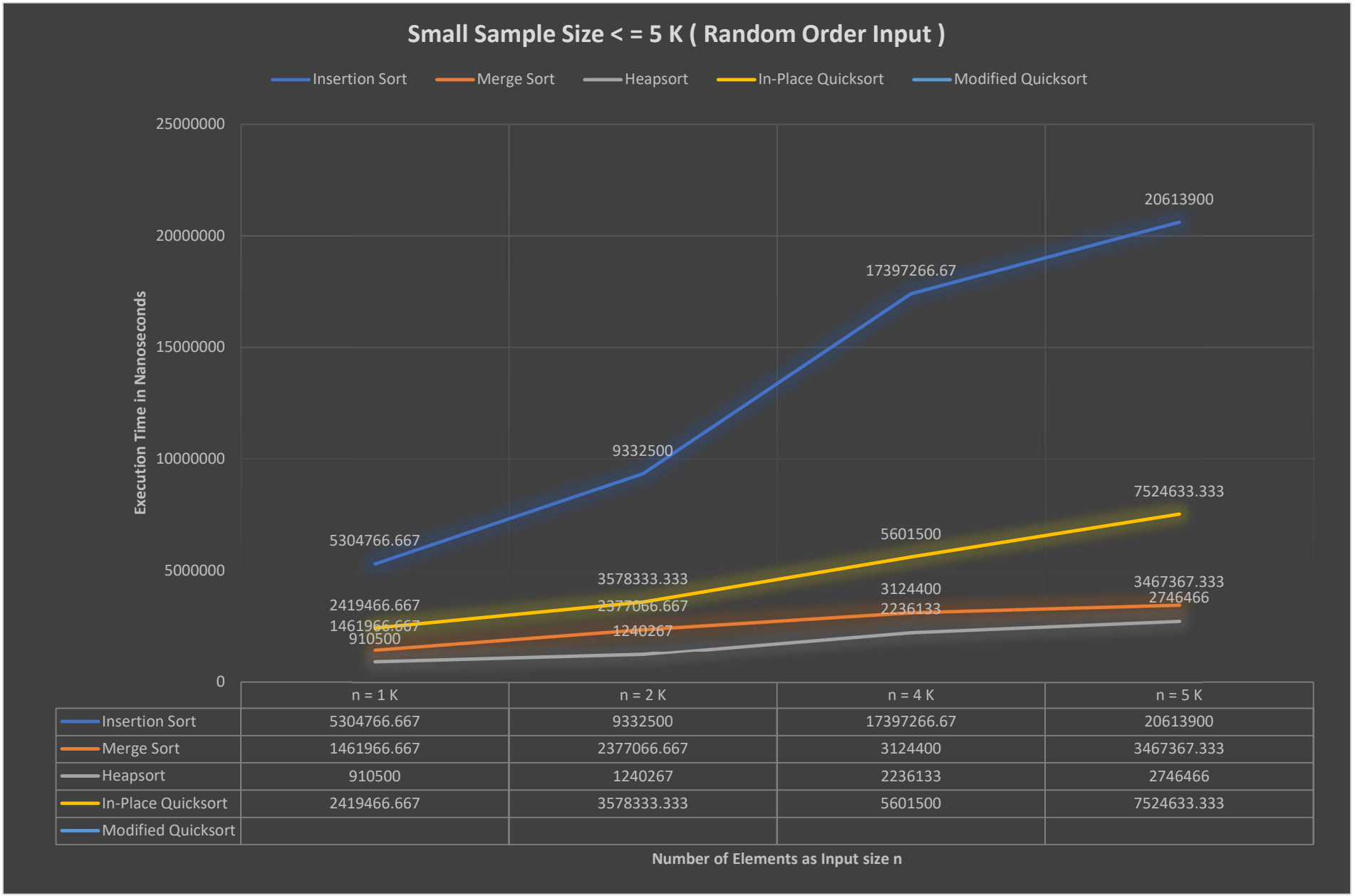
Merge Sort:

Heapsort:

In-Place Quicksort:

Modified Quicksort:

Small Sample Size Execution Results Graphs (<= 5 K):

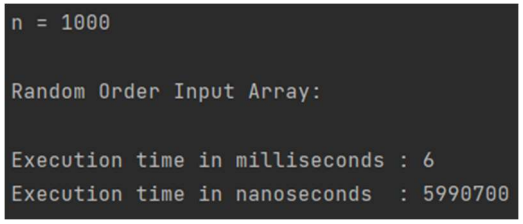


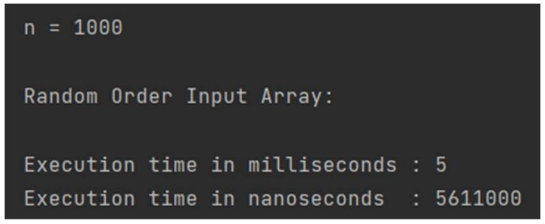
Small size Random Order Input Graph Analysis:

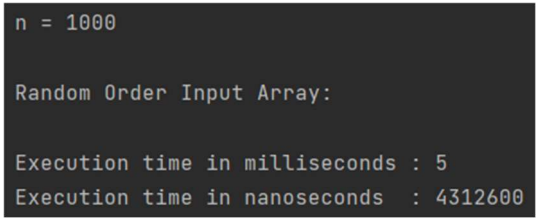
For input sizes less than or equal to 5000, that are in random order our average worst performer is insertion sort that runs in O(n^2) time. The next worst to best performers are in-place quicksort, merge sort and heap sort. These three sorting algorithms run in O(n log n) time. Our results seem to reflect this notion.

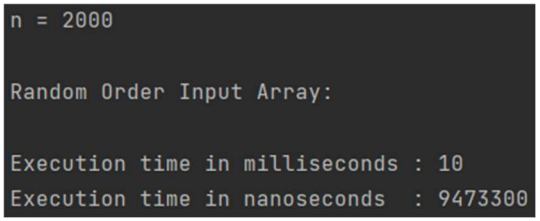
Insertion Sort:

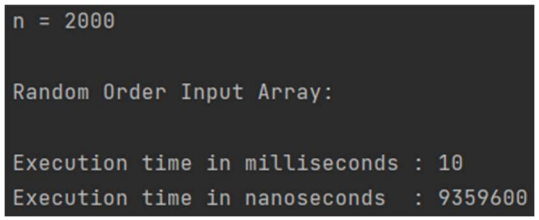
Input Size n	Run # 1	Run # 2	Run #3	Average
1 K	5990700	5611000	4312600	5304766.667
2 K	9473300	9359600	9164600	9332500
4 K	17934000	14865800	19392000	17397266.67
5 K	17435000	18201900	26204800	20613900

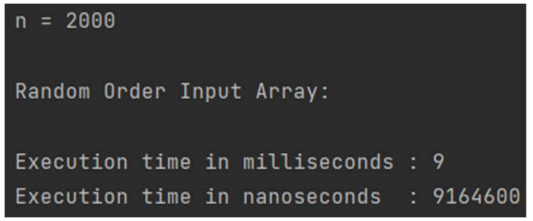


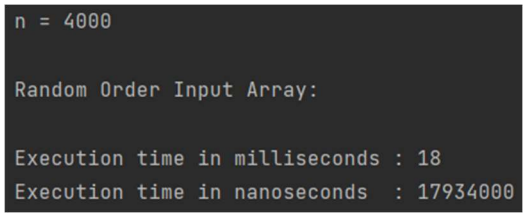


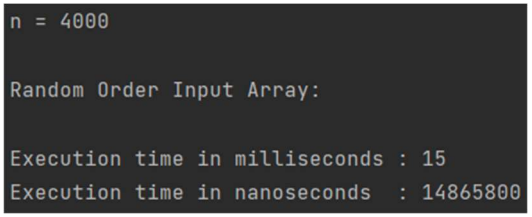


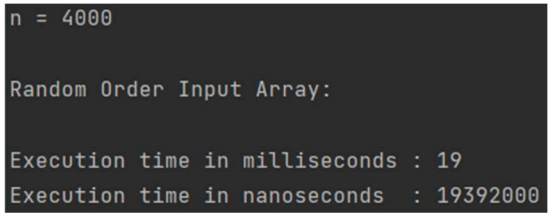












```
n = 5000

Random Order Input Array:

Execution time in milliseconds : 18
Execution time in nanoseconds  : 17435000
```

```
n = 5000

Random Order Input Array:

Execution time in milliseconds : 18
Execution time in nanoseconds  : 18201900
```

```
n = 5000

Random Order Input Array:

Execution time in milliseconds : 26
Execution time in nanoseconds  : 26204800
```

Merge Sort:

Input Size n	Run # 1	Run # 2	Run #3	Average
1 K	1715800	1617000	1053100	1461966.667
2 K	2481700	2069600	2579900	2377066.667
4 K	3234600	2759600	3379000	3124400
5 K	3444001	3125200	3832901	3467367.333

```
n = 1000

Random Order Input Array:

Execution time in milliseconds : 2
Execution time in nanoseconds  : 1715800
```

```
n = 1000

Random Order Input Array:

Execution time in milliseconds : 1
Execution time in nanoseconds  : 1617000
```

```
n = 1000

Random Order Input Array:

Execution time in milliseconds : 1
Execution time in nanoseconds  : 1053100
```

```
n = 2000

Random Order Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 2481700
```

```
n = 2000

Random Order Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 2069600
```

```
n = 2000

Random Order Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 2579900
```

```
n = 4000

Random Order Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 3234600
```

```
n = 4000

Random Order Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 2759600
```

```
n = 4000

Random Order Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 3379000
```

```
n = 5000

Random Order Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 3444001
```

```
n = 5000

Random Order Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 3125200
```

```
n = 5000

Random Order Input Array:

Execution time in milliseconds : 4
Execution time in nanoseconds  : 3832901
```

In-Place Quick Sort:

Input Size n	Run # 1	Run # 2	Run #3	Average
1 K	2064900	2914800	2278700	2419466.667
z K	3836700	3926000	2972300	3578333.333
4 K	5667300	5690300	5446900	5601500
5 K	7141100	8121000	7311800	7524633.333

```
n = 1000

Random Order Input Array:

Execution time in milliseconds : 2
Execution time in nanoseconds  : 2064900
```

```
n = 1000

Random Order Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 2914800
```

```
n = 1000

Random Order Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 2278700
```

```
n = 2000

Random Order Input Array:

Execution time in milliseconds : 4
Execution time in nanoseconds  : 3836700
```

```
n = 2000

Random Order Input Array:

Execution time in milliseconds : 4
Execution time in nanoseconds  : 3926000
```

```
n = 2000

Random Order Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 2972300
```

```
n = 4000

Random Order Input Array:

Execution time in milliseconds : 5
Execution time in nanoseconds  : 5667300
```

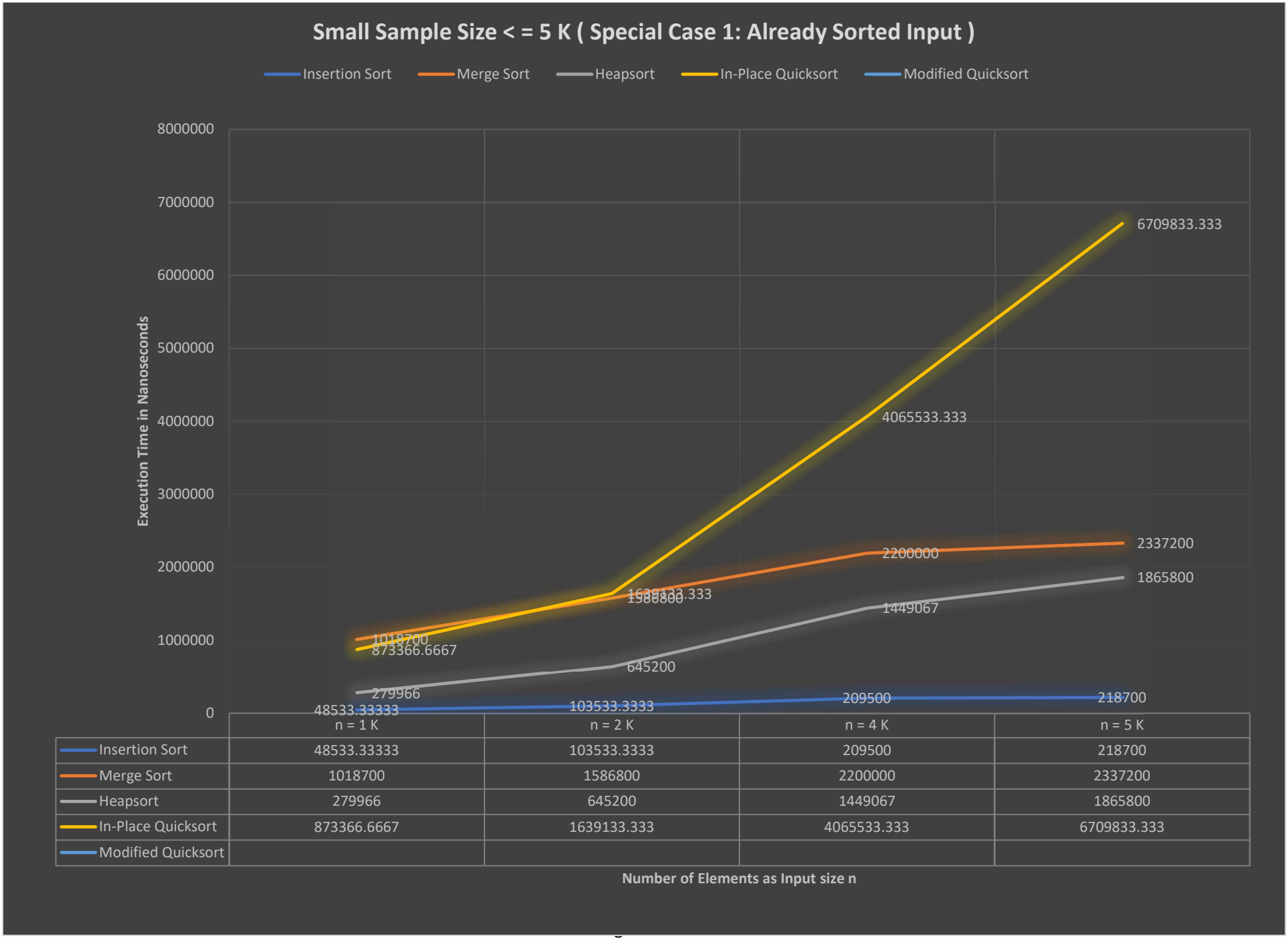
```
n = 4000

Random Order Input Array:

Execution time in milliseconds : 6
Execution time in nanoseconds  : 5690300
```

```
n = 4000

Random Order Input Array:

Execution time in milliseconds : 5
Execution time in nanoseconds  : 5446900
```

```
n = 5000

Random Order Input Array:

Execution time in milliseconds : 7
Execution time in nanoseconds  : 7141100
```

```
n = 5000

Random Order Input Array:

Execution time in milliseconds : 8
Execution time in nanoseconds  : 8121000
```

```
n = 5000

Random Order Input Array:

Execution time in milliseconds : 7
Execution time in nanoseconds  : 7311800
```



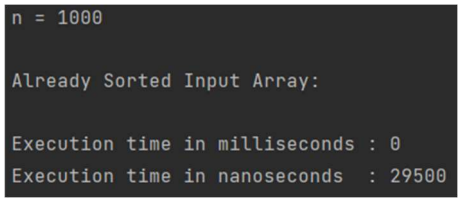
	n = 1 K	n = 2 K	n = 4 K	n = 5 K
Insertion Sort	48533.33333	103533.3333	209500	218700
Merge Sort	1018700	1586800	2200000	2337200
Heapsort	279966	645200	1449067	1865800
In-Place Quicksort	873366.6667	1639133.333	4065533.333	6709833.333
Modified Quicksort				

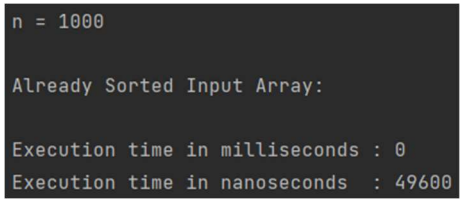
Small size Already Sorted Input Graph Analysis:

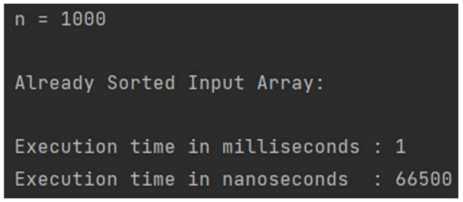
In the special case that the input array is already sorted, our worst performer is the in-place quicksort, and our best performer is insertion sort. One interesting observation is that the in-place quicksort outperforms the merge sort initially when n is less than or equal to 1000. However, for n sizes greater than or equal to 2000 the execution time for in place quick sort grows dramatically. Heap and merge sort follow a similar growth rate with the edge going to heap sort.

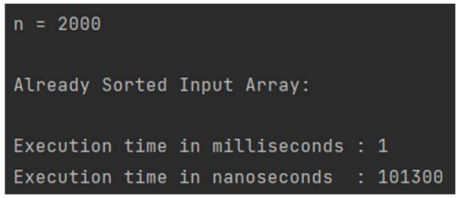
Insertion Sort (Special Case 1: Already Sorted Input):

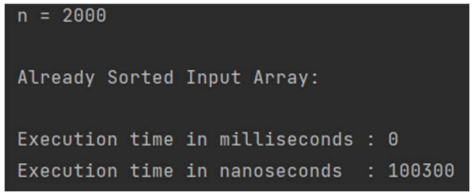
Input Size n	Run # 1	Run # 2	Run #3	Average
1 K	29500	49600	66500	48533.33333
2 K	101300	100300	109000	103533.3333
4 K	179900	228300	220300	209500
5 K	154000	293300	208800	218700

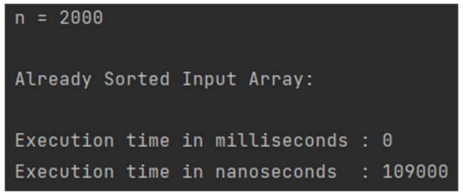


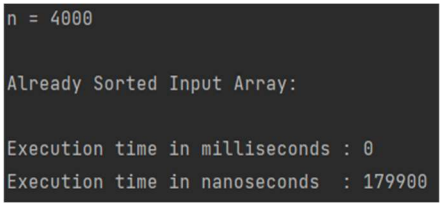


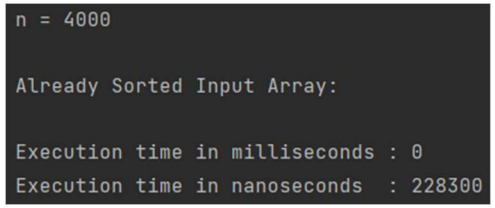


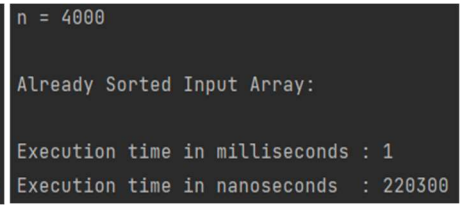


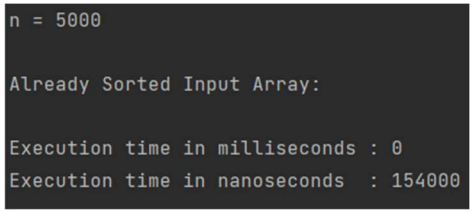


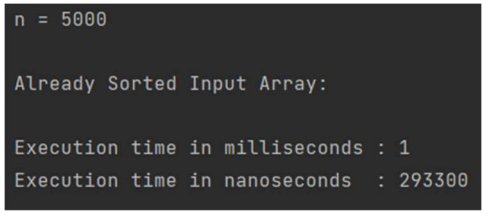


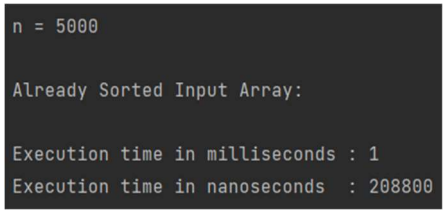






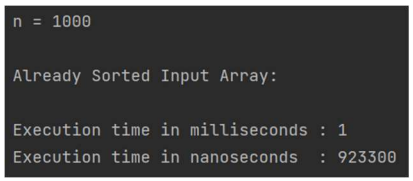


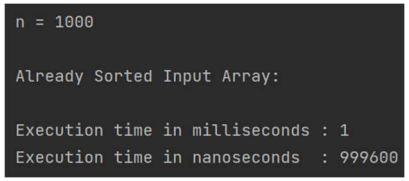


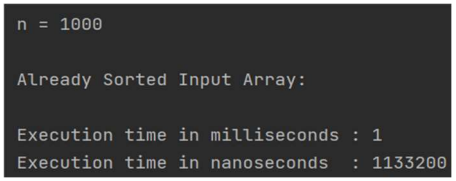


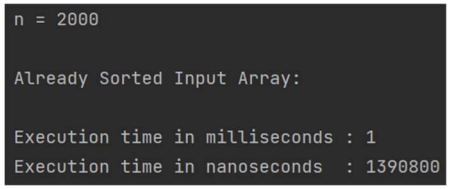
Merge Sort (Special Case 1: Already Sorted Input):

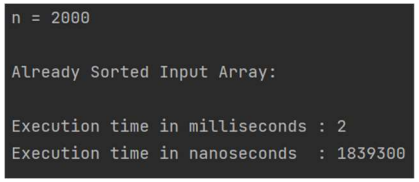
Input Size n	Run # 1	Run # 2	Run #3	Average
1 K	923300	999600	1133200	1018700
2 K	1390800	1839300	1530300	1586800
4 K	2451500	1912100	2236400	2200000
5 K	2354800	2106700	2550100	2337200

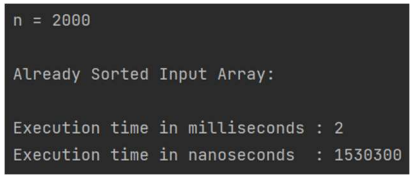


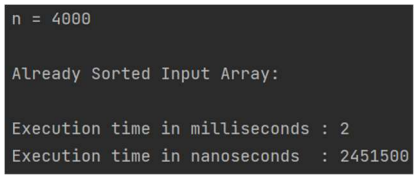


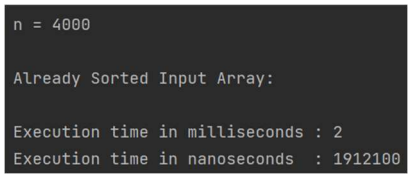


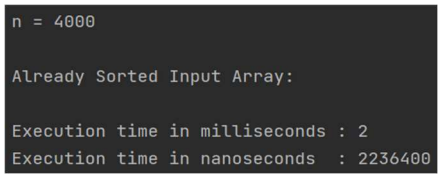


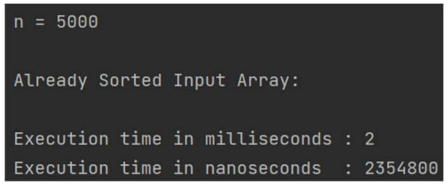


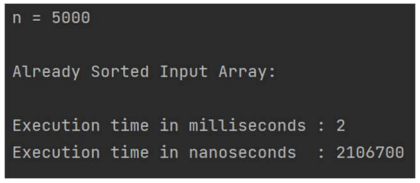


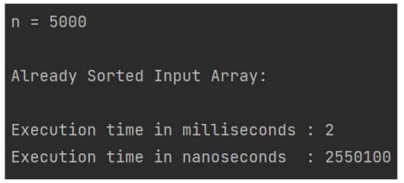






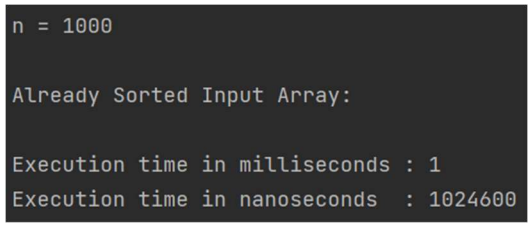


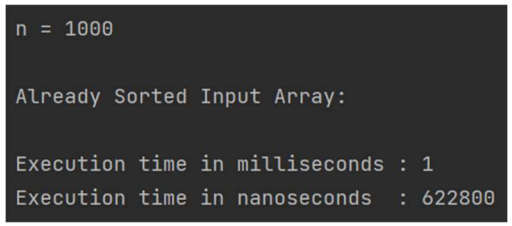


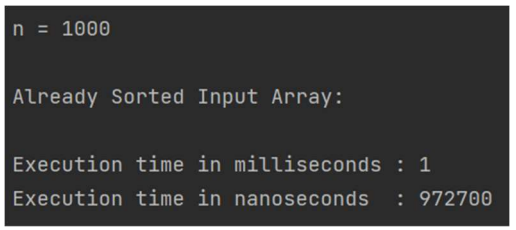


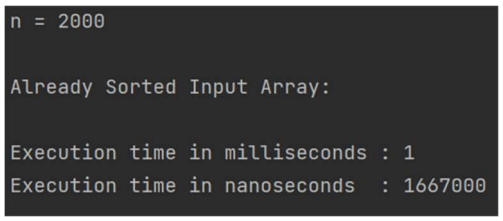
In-Place Quick Sort (Special Case 1: Already Sorted Input):

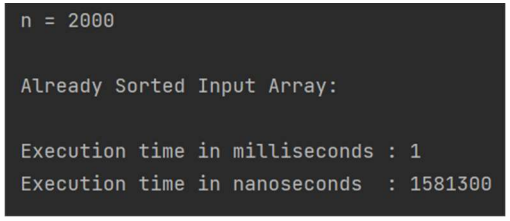
Input Size n	Run # 1	Run # 2	Run #3	Average
1 K	1024600	622800	972700	873366.6667
2 K	1667000	1581300	1669100	1639133.333
4 K	4423500	3949000	3824100	4065533.333
5 K	5166200	7626100	7337200	6709833.333

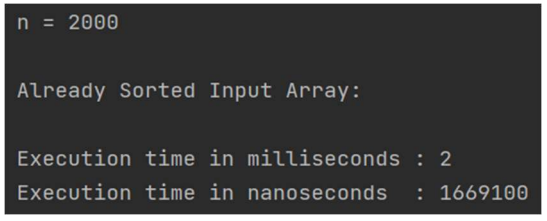












```
n = 4000

Already Sorted Input Array:

Execution time in milliseconds : 4
Execution time in nanoseconds  : 4423500
```

```
n = 4000

Already Sorted Input Array:

Execution time in milliseconds : 4
Execution time in nanoseconds  : 3949000
```

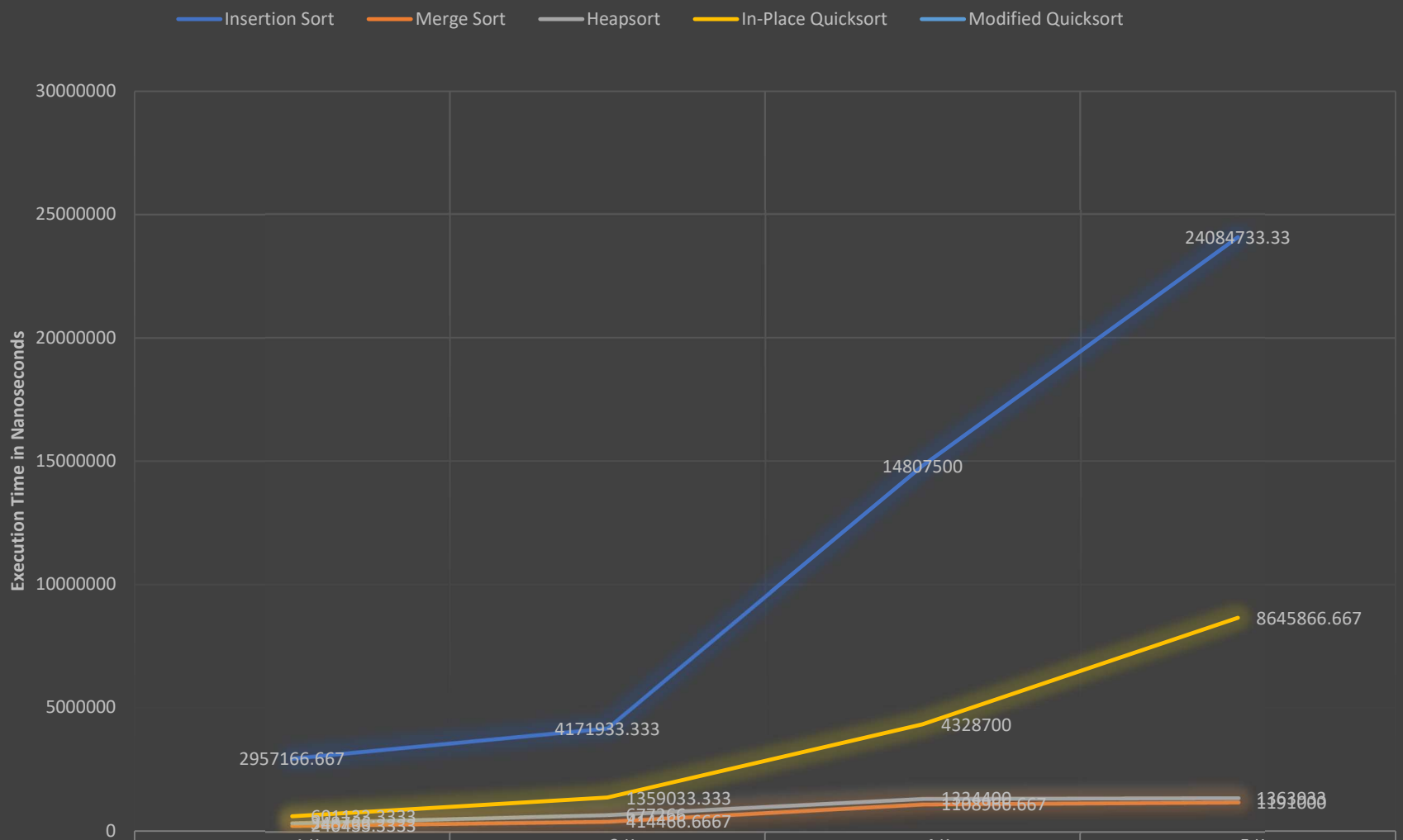
```
n = 4000

Already Sorted Input Array:

Execution time in milliseconds : 4
Execution time in nanoseconds  : 3824100
```

```
n = 5000

Already Sorted Input Array:

Execution time in milliseconds : 5
Execution time in nanoseconds  : 5166200
```

```
n = 5000

Already Sorted Input Array:

Execution time in milliseconds : 8
Execution time in nanoseconds  : 7626100
```

```
n = 5000

Already Sorted Input Array:

Execution time in milliseconds : 7
Execution time in nanoseconds  : 7337200
```

Small Sample Size < = 5 K (Special Case 2: Reversely Sorted Input)



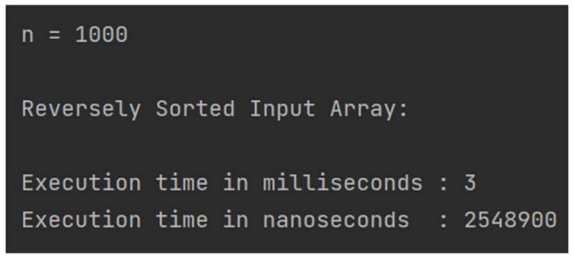
	n = 1 K	n = 2 K	n = 4 K	n = 5 K
Insertion Sort	2957166.667	4171933.333	14807500	24084733.33
Merge Sort	240433.3333	414466.6667	1108966.667	1191000
Heapsort	348766	677266	1334400	1363933
In-Place Quicksort	601133.3333	1359033.333	4328700	8645866.667
Modified Quicksort				

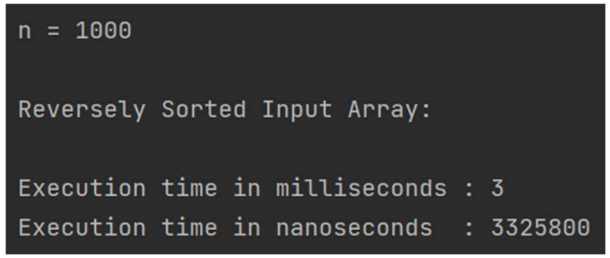
Small size Reversely Sorted Input Graph Analysis:

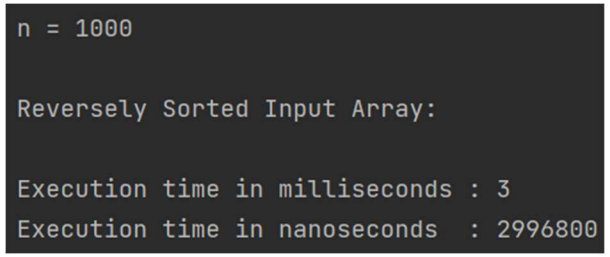
Our worst performer in the special case that the input is reversely sorted is insertion sort. This is the worst case for the algorithm. Similarly one of the scenarios that cause the in-place quicksort to perform at its worst-case running time is when the input array is reversely sorted. This produces an O(n^2) runtime that is on par with the insertion sort in this case. Our best performers in this case are the merge and heap sort. These two algorithms both have a runtime of O(n log n).

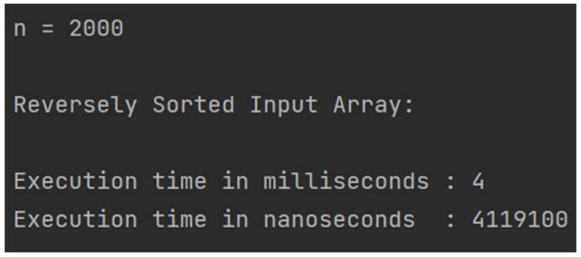
Insertion Sort (Special Case 2: Reversely Sorted Input):

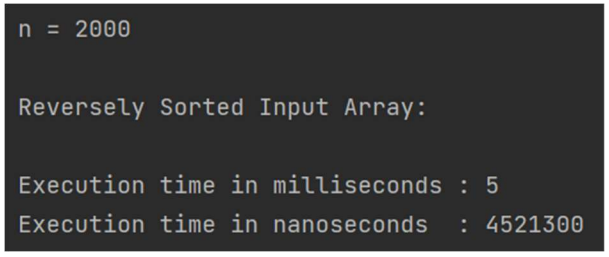
Input Size n	Run # 1	Run # 2	Run #3	Average
1 K	2548900	3325800	2996800	2957166.667
2 K	4119100	4521300	3875400	4171933.333
4 K	16944100	11688400	15790000	14807500
5 K	28739500	16734900	26779800	24084733.33

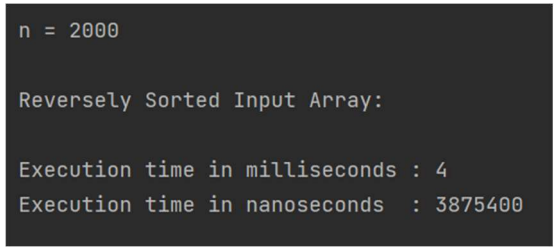












```
n = 4000

Reversely Sorted Input Array:

Execution time in milliseconds : 17
Execution time in nanoseconds  : 16944100
```

```
n = 4000

Reversely Sorted Input Array:

Execution time in milliseconds : 11
Execution time in nanoseconds  : 11688400
```

```
n = 4000

Reversely Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 15790000
```

```
n = 5000

Reversely Sorted Input Array:

Execution time in milliseconds : 21
Execution time in nanoseconds  : 20739500
```

```
n = 5000

Reversely Sorted Input Array:

Execution time in milliseconds : 17
Execution time in nanoseconds  : 16734900
```

```
n = 5000

Reversely Sorted Input Array:

Execution time in milliseconds : 26
Execution time in nanoseconds  : 26779800
```

Merge Sort (Special Case 2: Reversely Sorted Input):

Input Size n	Run # 1	Run # 2	Run #3	Average
1 K	219800	255400	246100	240433.3333
2 K	454700	329100	459600	414466.6667
4 K	1140700	1164400	1021800	1108966.667
5 K	1337300	918100	1317600	1191000

```
n = 1000

Reversely Sorted Input Array:

Execution time in milliseconds : 1
Execution time in nanoseconds  : 219800
```

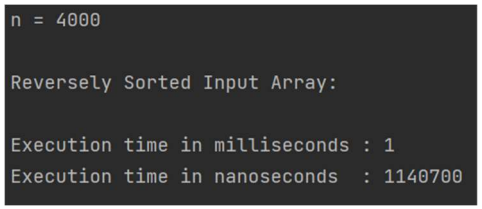
```
n = 1000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 255400
```

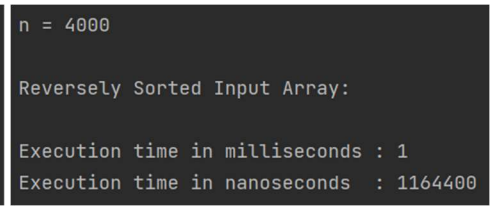
```
n = 1000

Reversely Sorted Input Array:

Execution time in milliseconds : 1
Execution time in nanoseconds  : 246100
```

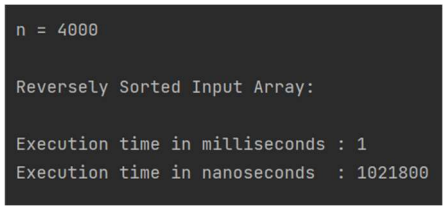
```
n = 2000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 454700
```

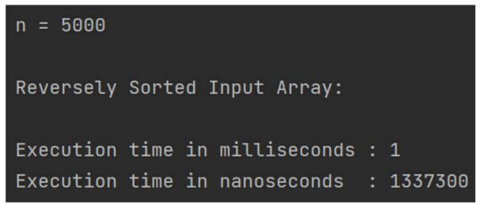
```
n = 2000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 329100
```

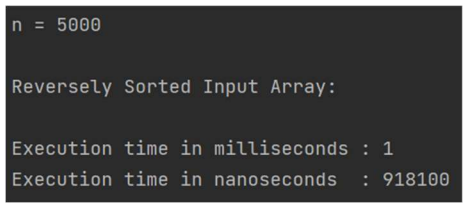
```
n = 2000

Reversely Sorted Input Array:

Execution time in milliseconds : 1
Execution time in nanoseconds  : 459600
```

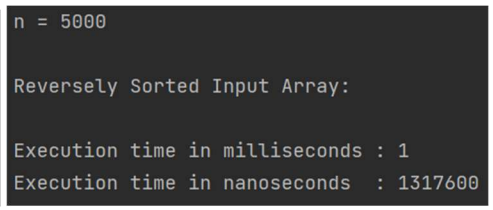






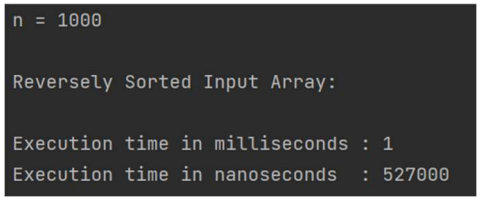


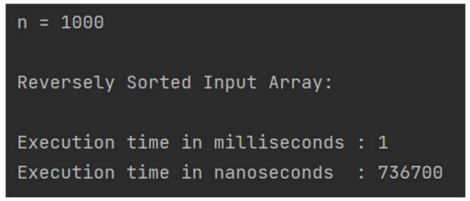


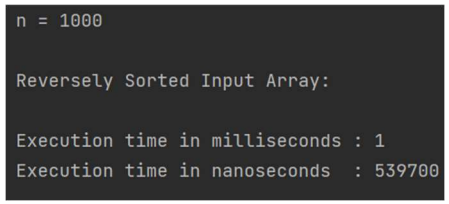


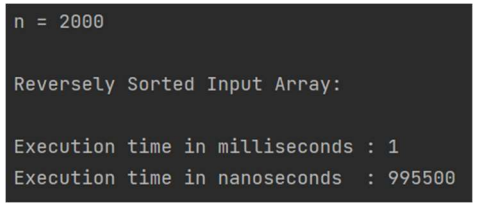
In-Place Quick Sort (Special Case 2: Reversely Sorted Input):

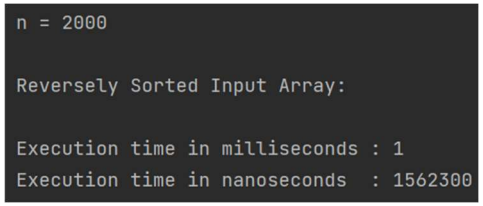
Input Size n	Run # 1	Run # 2	Run #3	Average
1 K	527000	736700	539700	601133.3333
2 K	995500	1562300	1519300	1359033.333
4 K	4517600	4706200	3762300	4328700
5 K	8268900	7417000	10251700	8645866.667

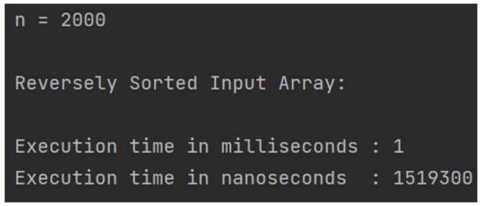












```
n = 4000

Reversely Sorted Input Array:

Execution time in milliseconds : 4
Execution time in nanoseconds  : 4517600
```

```
n = 4000

Reversely Sorted Input Array:

Execution time in milliseconds : 5
Execution time in nanoseconds  : 4706200
```

```
n = 4000

Reversely Sorted Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 3762300
```

```
n = 5000

Reversely Sorted Input Array:

Execution time in milliseconds : 8
Execution time in nanoseconds  : 8268900
```

```
n = 5000

Reversely Sorted Input Array:

Execution time in milliseconds : 7
Execution time in nanoseconds  : 7417000
```

```
n = 5000

Reversely Sorted Input Array:

Execution time in milliseconds : 10
Execution time in nanoseconds  : 10251700
```

Heap sort:

Table: Input size, and time in nanoseconds

Input Size	Run #1	Run #2	Run #3	Average
1K	900300	887600	943600	910500
2K	1123700	1178701	1418401	1240267
4K	1878300	2501500	2328600	2236133
5K	3170700	2477000	2591700	2746466

```
n = 1000

Random Order Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 900300
```

```
n = 1000

Random Order Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 887600
```

```
n = 1000

Random Order Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 943600
```

```
n = 2000

Random Order Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 1123700
```

```
n = 2000

Random Order Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 1178701
```

```
n = 2000

Random Order Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 1418401
```

```
n = 4000

Random Order Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 1878300
```

```
n = 4000

Random Order Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 2501500
```

```
n = 4000

Random Order Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 2328600
```

```
n = 5000

Random Order Input Array:

Execution time in milliseconds : 15
Execution time in nanoseconds  : 3170700
```

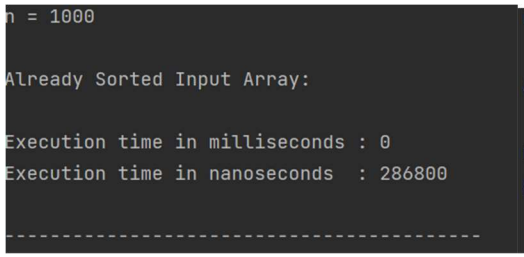
```
n = 5000

Random Order Input Array:

Execution time in milliseconds : 15
Execution time in nanoseconds  : 2477000
```

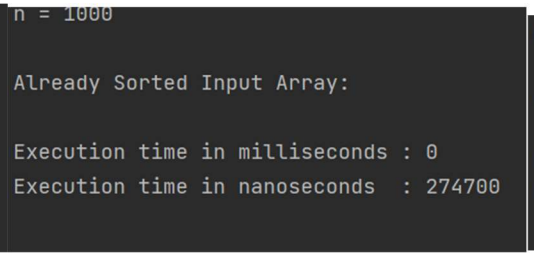
```
n = 5000

Random Order Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 2591700
```

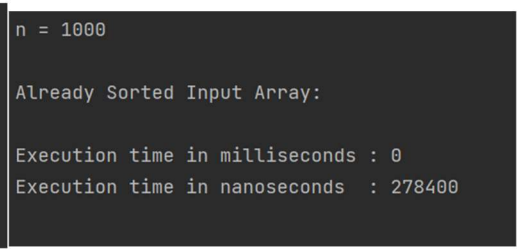
Heap Sort Special Case 1 for Small Input (Sorted Array):

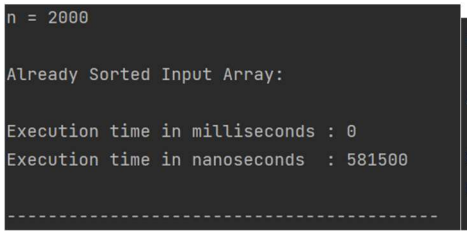
Table: Input size, and time in nanoseconds

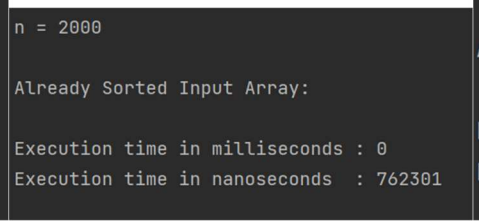
Input size	Run #1	Run#2	Run#3	Average
1K	286800	274700	278400	279966
2K	581500	762301	591801	645200
4K	1667300	1215400	1464501	1449067
5K	2397600	1617300	1582500	1865800

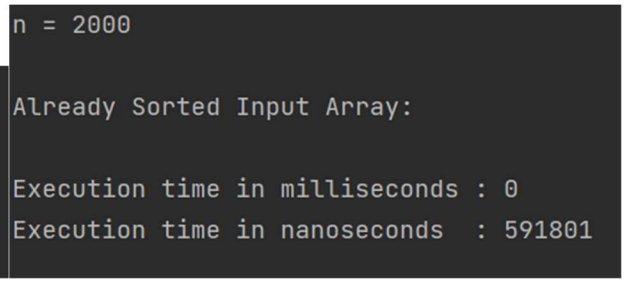


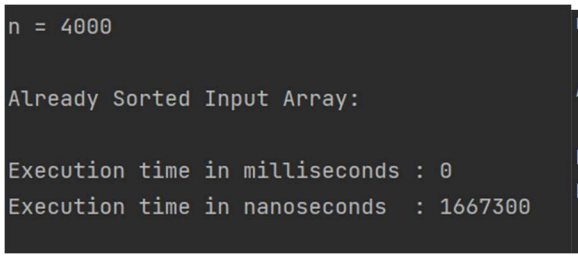


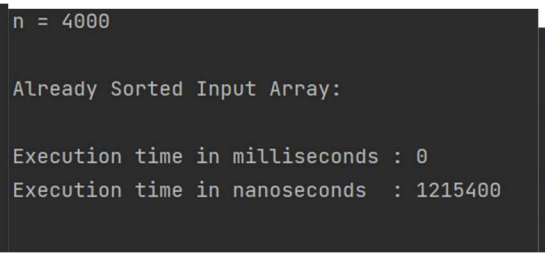


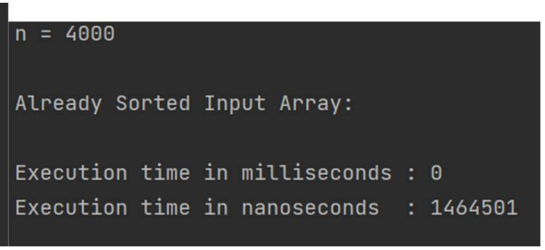


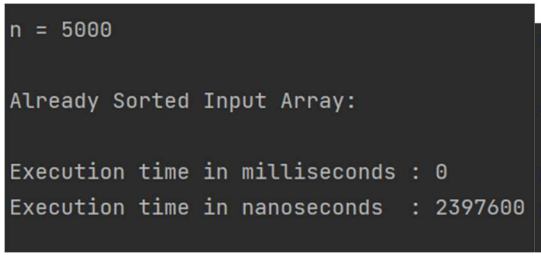


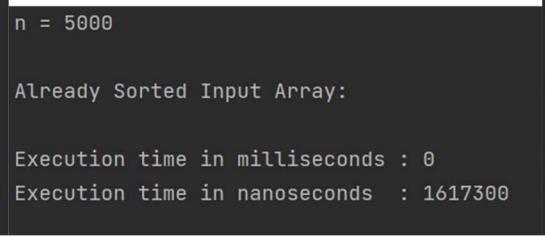


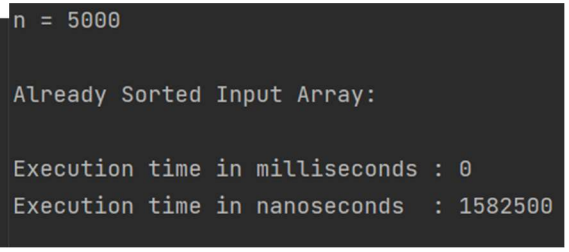












Heapsort Special Case #2 (Reverse sorted array)

Input Size	Run #1	Run#2	Run#3	Average
1K	391801	397900	256599	348766
2K	585900	567399	878499	677266
4K	1380899	1191001	1431300	1334400
5K	1422500	1100300	1569000	1363933

```
n = 1000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 391801

----------------------------------------
```

```
n = 1000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 397900

----------------------------------------
```

```
n = 1000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 256599
```

```
n = 2000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 585900

----------------------------------------
```

```
n = 2000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 567399
```

```
n = 2000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 878499
```

```
n = 4000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 1380899
```

```
n = 4000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 1191001
```

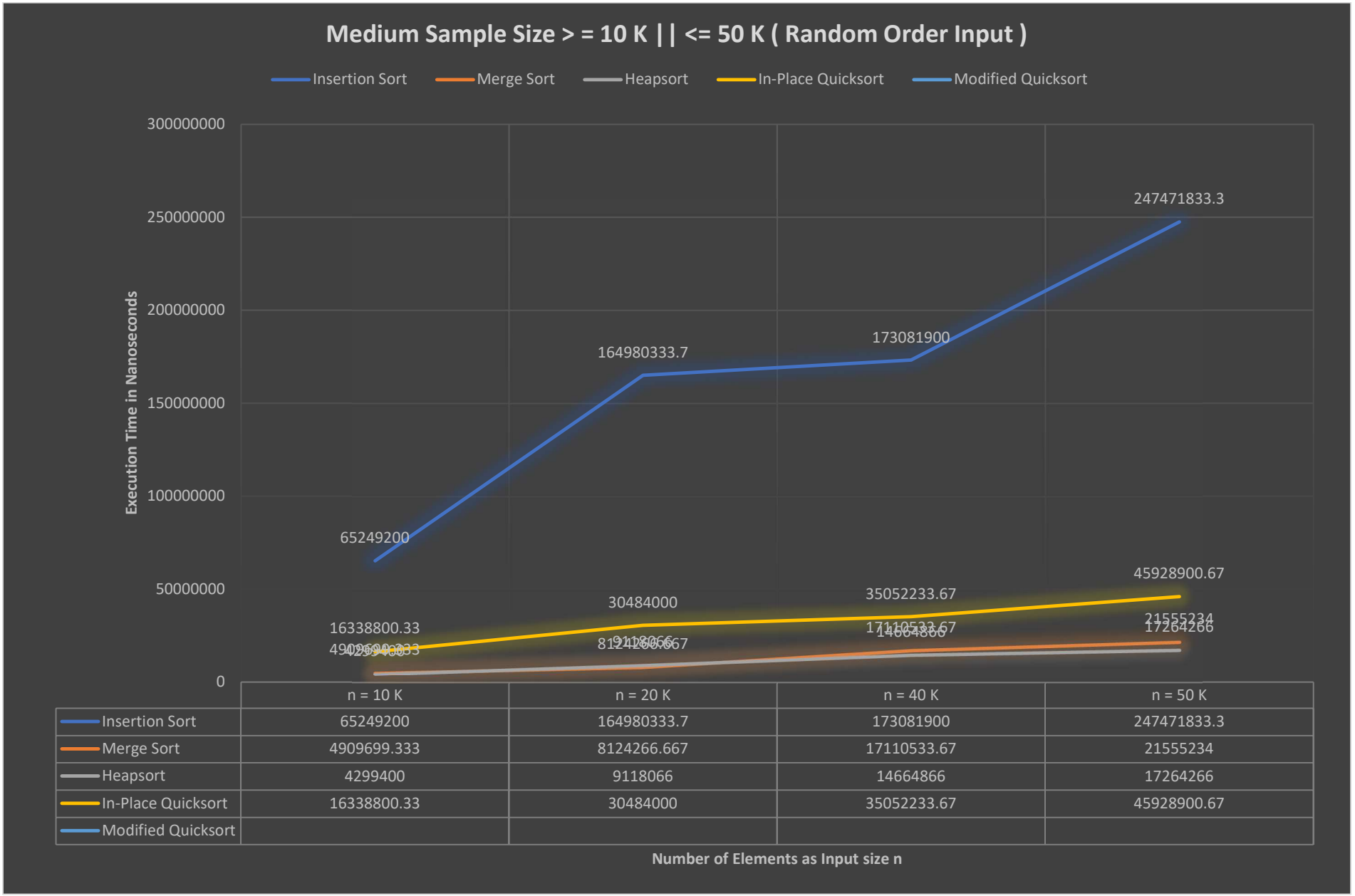
```
n = 4000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 1431300
```

```
n = 5000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 1422500
```

```
n = 5000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 1100300
```

```
n = 5000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 1569000
```

Medium Sample Size Execution Results Graphs (> = 10 K || <= 50 K):

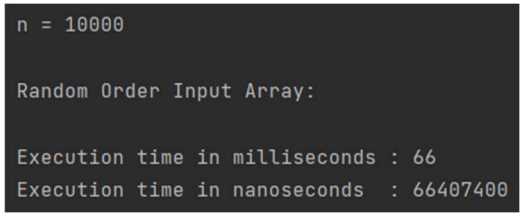


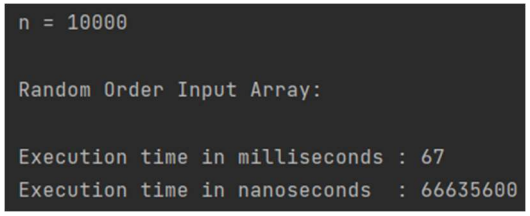
Medium size Random Order Input Graph Analysis:

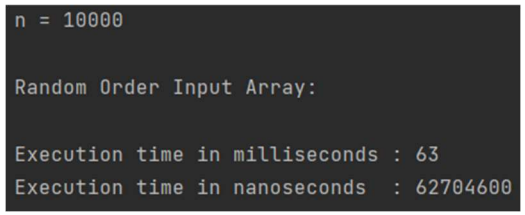
For medium size random order input we have a similar trend with small size random order inputs. Insertion sort has a worst and average running time of O(n^2).

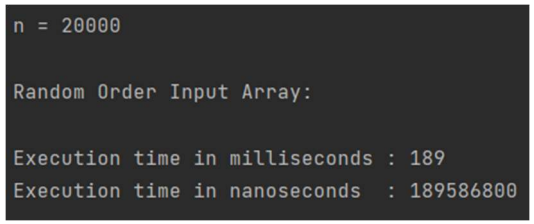
Insertion Sort (Random Order Input):

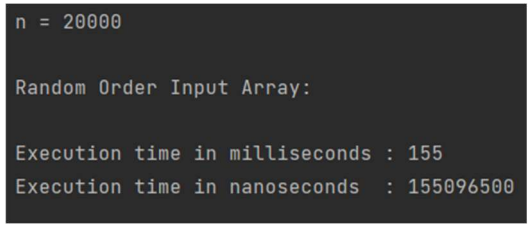
Input Size n	Run # 1	Run # 2	Run #3	Average
10 K	66407400	66635600	62704600	65249200
20 K	189586800	155096500	150257701	164980333.7
40 K	166306500	180626400	172312800	173081900
50 K	236952201	252255800	253207499	247471833.3

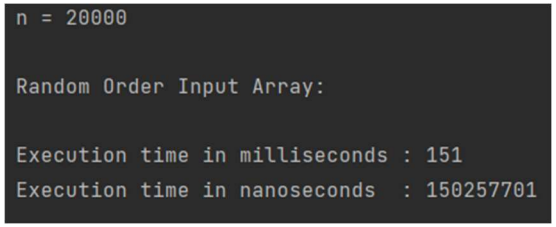


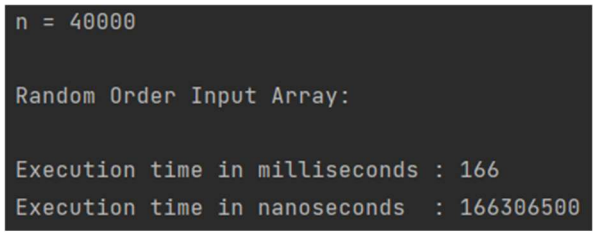


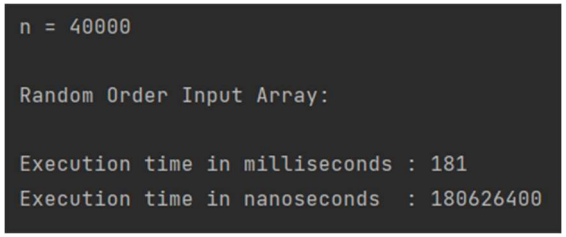


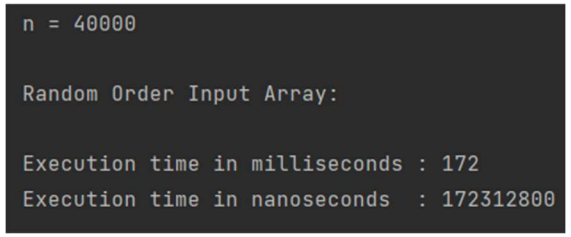












```
n = 50000

Random Order Input Array:

Execution time in milliseconds : 237
Execution time in nanoseconds  : 236952201
```

```
n = 50000

Random Order Input Array:

Execution time in milliseconds : 252
Execution time in nanoseconds  : 252255800
```

```
n = 50000

Random Order Input Array:

Execution time in milliseconds : 253
Execution time in nanoseconds  : 253207499
```

Merge Sort (Random Order Input):

Input Size n	Run # 1	Run # 2	Run #3	Average
10 K	4578399	5327000	4823699	4909699.333
20 K	7730800	9174800	7467200	8124266.667
40 K	15798801	17065500	18467300	17110533.67
50 K	22190601	19802401	22672700	21555234

```
n = 10000

Random Order Input Array:

Execution time in milliseconds : 5
Execution time in nanoseconds  : 4578399
```

```
n = 10000

Random Order Input Array:

Execution time in milliseconds : 5
Execution time in nanoseconds  : 5327000
```

```
n = 10000

Random Order Input Array:

Execution time in milliseconds : 5
Execution time in nanoseconds  : 4823699
```

```
n = 20000

Random Order Input Array:

Execution time in milliseconds : 8
Execution time in nanoseconds  : 7730800
```

```
n = 20000

Random Order Input Array:

Execution time in milliseconds : 9
Execution time in nanoseconds  : 9174800
```

```
n = 20000

Random Order Input Array:

Execution time in milliseconds : 7
Execution time in nanoseconds  : 7467200
```

```
n = 40000

Random Order Input Array:

Execution time in milliseconds : 15
Execution time in nanoseconds  : 15798801
```

```
n = 40000

Random Order Input Array:

Execution time in milliseconds : 17
Execution time in nanoseconds  : 17065500
```

```
n = 40000

Random Order Input Array:

Execution time in milliseconds : 18
Execution time in nanoseconds  : 18467300
```

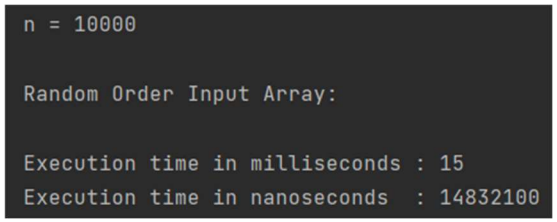
```
n = 50000

Random Order Input Array:

Execution time in milliseconds : 23
Execution time in nanoseconds  : 22190601
```

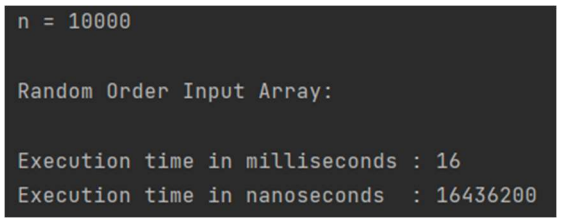
```
n = 50000

Random Order Input Array:

Execution time in milliseconds : 20
Execution time in nanoseconds  : 19802401
```

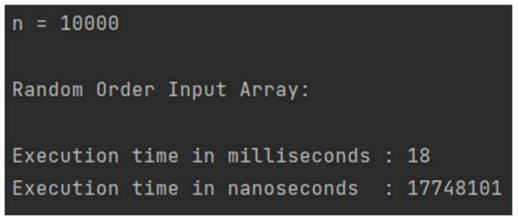
```
n = 50000

Random Order Input Array:

Execution time in milliseconds : 23
Execution time in nanoseconds  : 22672700
```

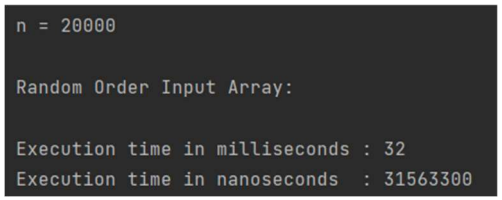
In Place Quick Sort (Random Order Input):

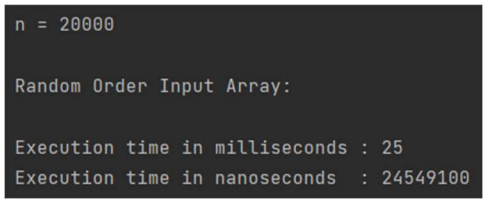
Input Size n	Run # 1	Run # 2	Run #3	Average
10 K	14832100	16436200	17748101	16338800.33
20 K	31563300	24549100	35339600	30484000
40 K	37954800	34556901	32645000	35052233.67
50 K	51342801	43340801	43103100	45928900.67

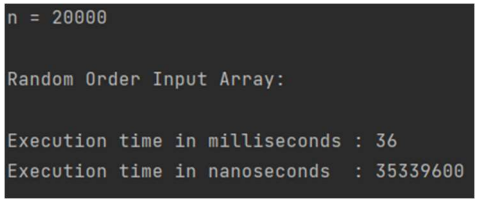


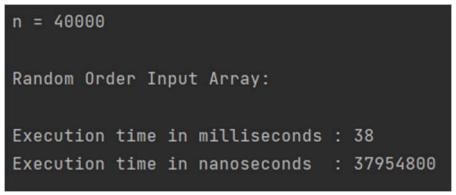


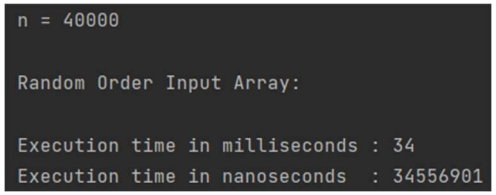


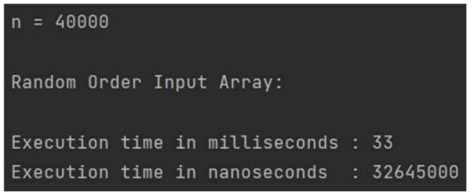












```
n = 50000

Random Order Input Array:

Execution time in milliseconds : 51
Execution time in nanoseconds  : 51342801
```

```
n = 50000

Random Order Input Array:

Execution time in milliseconds : 44
Execution time in nanoseconds  : 43340801
```

```
n = 50000

Random Order Input Array:

Execution time in milliseconds : 43
Execution time in nanoseconds  : 43103100
```

Heap Sort:

Input Size	Run #1	Run #2	Run #3	Average
10K	4029200	4091600	4777401	4299400
20K	8848400	9660800	8845000	9118066
40K	16673299	13945401	13375900	14664866
50L	18475900	16046799	17270100	17264266

```
n = 10000

Random Order Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 4029200
```

```
n = 10000

Random Order Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 4091600
```

```
n = 10000

Random Order Input Array:

Execution time in milliseconds : 15
Execution time in nanoseconds  : 4777401
```

```
n = 20000

Random Order Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 8848400
```

```
n = 20000

Random Order Input Array:

Execution time in milliseconds : 10
Execution time in nanoseconds  : 9660800
```

```
n = 20000

Random Order Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 8845000
```

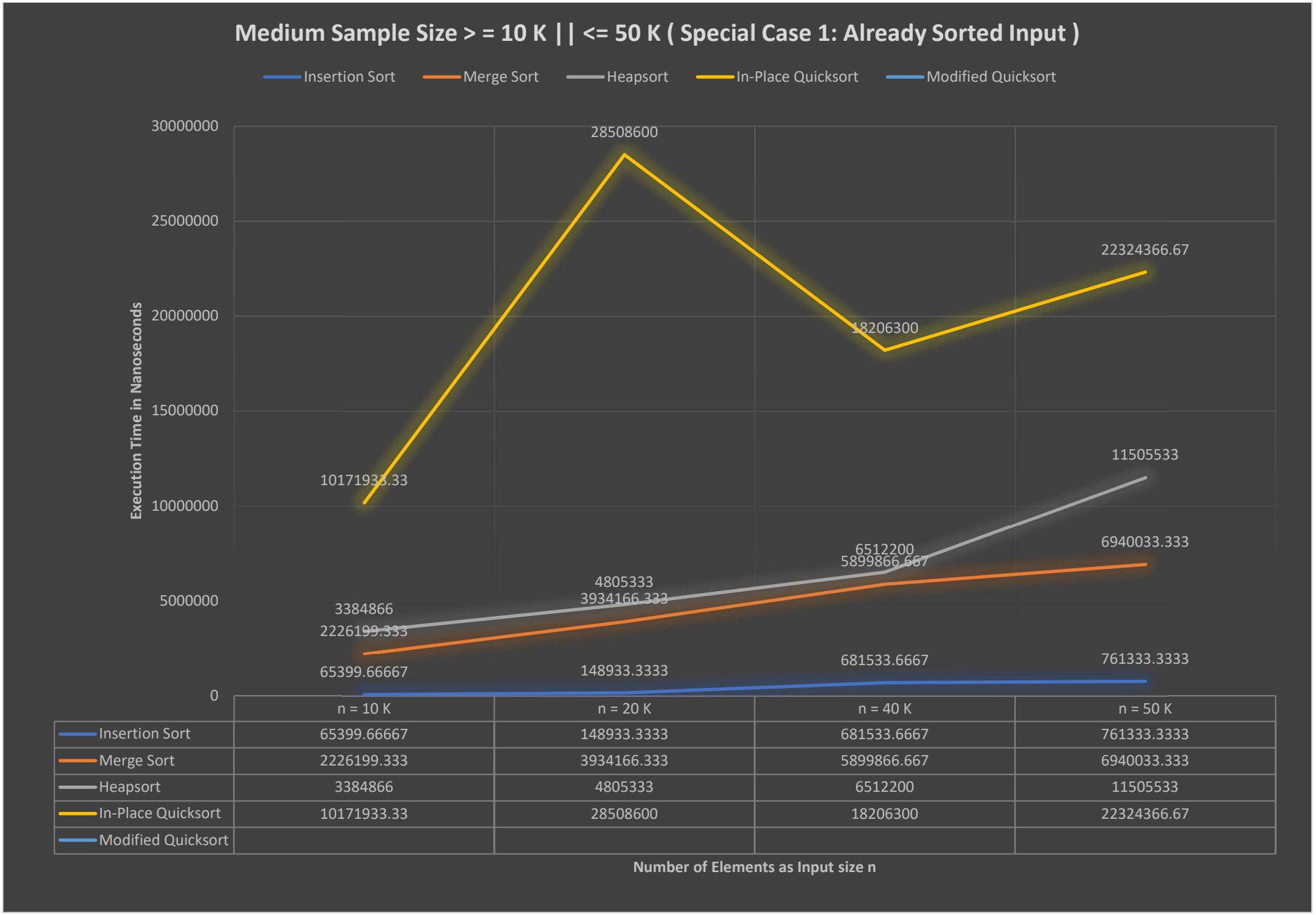
```
n = 40000

Random Order Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 16673299
```

```
n = 40000

Random Order Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 13945401
```

```
n = 40000

Random Order Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 13375900
```

```
n = 50000

Random Order Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 18475900
```

```
n = 50000

Random Order Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 17270100
```

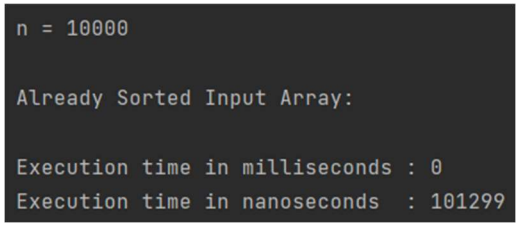

Medium Sample Size > = 10 K || <= 50 K (Special Case 1: Already Sorted Input)
Insertion Sort
Merge Sort
Heapsort
In-Place Quicksort
Modified Quicksort
30000000
25000000
20000000
15000000
10000000
5000000
0
Execution Time in Nanoseconds
28508600
22324366.67
18206300
10171933.33
11505533
6512200
5899866.667
6940033.333
4805333
3934166.333
3384866
2226199.333
65399.66667
148933.3333
681533.6667
761333.3333
n = 10 K
n = 20 K
n = 40 K
n = 50 K
Insertion Sort 65399.66667 148933.3333 681533.6667 761333.3333
Merge Sort 2226199.333 3934166.333 5899866.667 6940033.333
Heapsort 3384866 4805333 6512200 11505533
In-Place Quicksort 10171933.33 28508600 18206300 22324366.67
Modified Quicksort
Number of Elements as Input size n

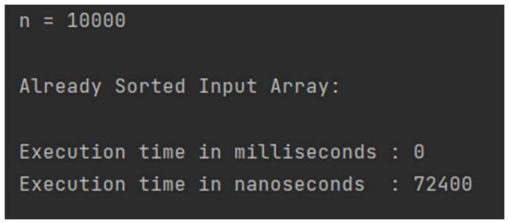
Medium size Already Sorted Input Graph Analysis:

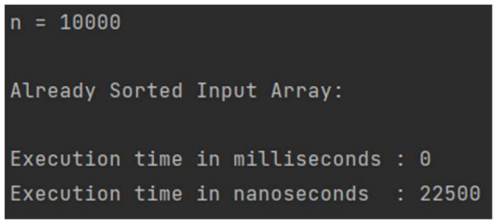
For medium size already sorted input the in-place quicksort is our worst performer. It runs at its worst-case runtime O(n^2) when the input array is already sorted. This graph is similar to the smaller input size. As the input grows however, heapsort seams to do worse. Insertion sort runs best in this case.

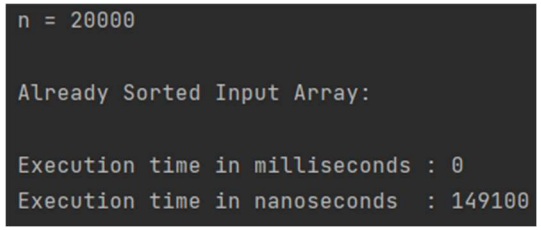
Insertion Sort (Already Sorted Input):

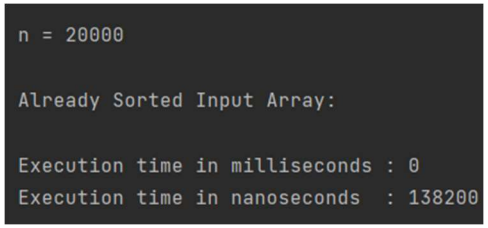
Input Size n	Run # 1	Run # 2	Run #3	Average
10 K	101299	72400	22500	65399.66667
20 K	149100	138200	159500	148933.3333
40 K	779800	694401	570400	681533.6667
50 K	989200	719200	575600	761333.3333

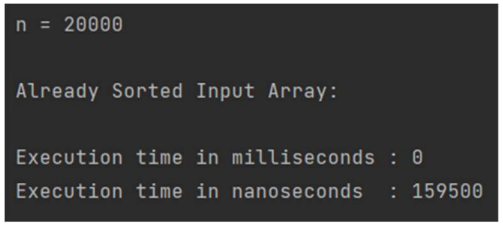


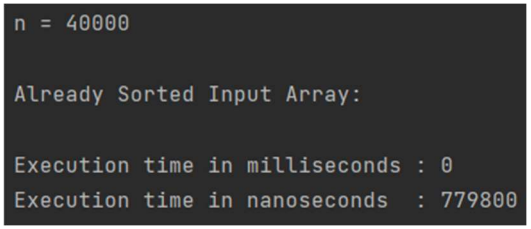


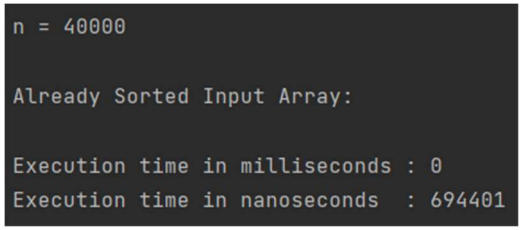


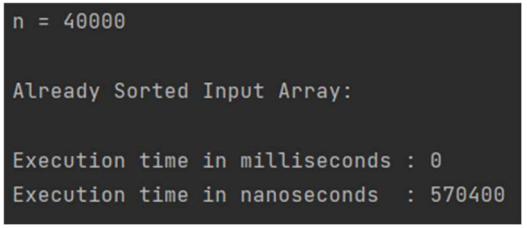












```
n = 50000

Already Sorted Input Array:

Execution time in milliseconds : 1
Execution time in nanoseconds  : 989200
```

```
n = 50000

Already Sorted Input Array:

Execution time in milliseconds : 1
Execution time in nanoseconds  : 719200
```

```
n = 50000

Already Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 575600
```

Merge Sort (Already Sorted Input):

Input Size n	Run # 1	Run # 2	Run #3	Average
10 K	2027299	2899999	1751300	2226199.333
20 K	3494399	4707500	3600600	3934166.333
40 K	5464599	6144101	6090900	5899866.667
50 K	6907700	6239800	7672600	6940033.333

```
n = 10000

Already Sorted Input Array:

Execution time in milliseconds : 2
Execution time in nanoseconds  : 2027299
```

```
n = 10000

Already Sorted Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 2899999
```

```
n = 10000

Already Sorted Input Array:

Execution time in milliseconds : 2
Execution time in nanoseconds  : 1751300
```

```
n = 20000

Already Sorted Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 3494399
```

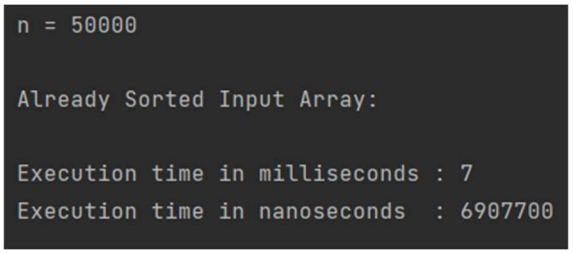
```
n = 20000

Already Sorted Input Array:

Execution time in milliseconds : 5
Execution time in nanoseconds  : 4707500
```

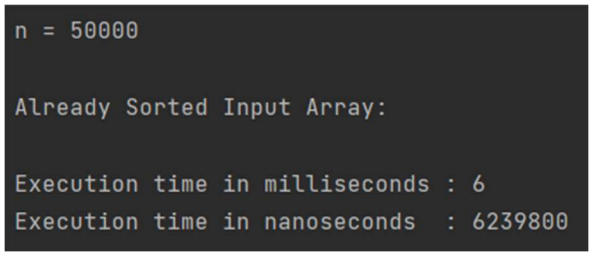
```
n = 20000

Already Sorted Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 3600600
```

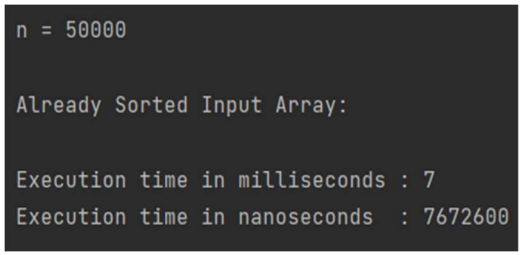
```
n = 40000

Already Sorted Input Array:

Execution time in milliseconds : 6
Execution time in nanoseconds  : 5464599
```

```
n = 40000

Already Sorted Input Array:

Execution time in milliseconds : 6
Execution time in nanoseconds  : 6144101
```

```
n = 40000

Already Sorted Input Array:

Execution time in milliseconds : 6
Execution time in nanoseconds  : 6090900
```

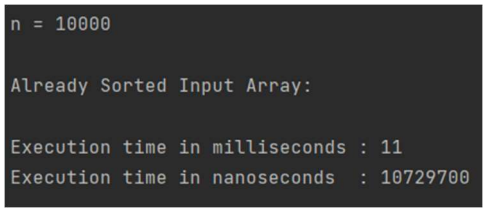


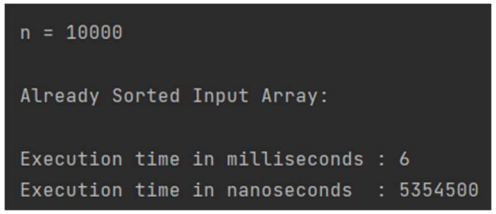


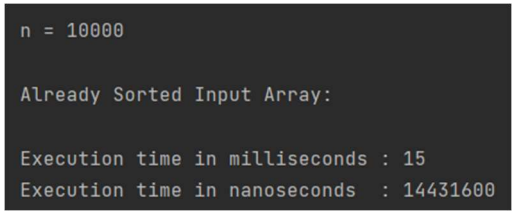


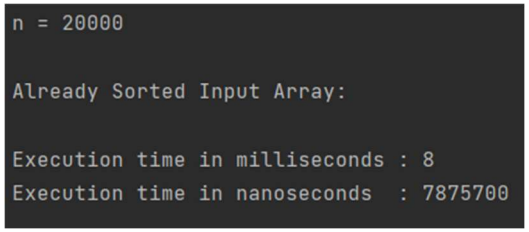
In Place Quick Sort (Already Sorted Input):

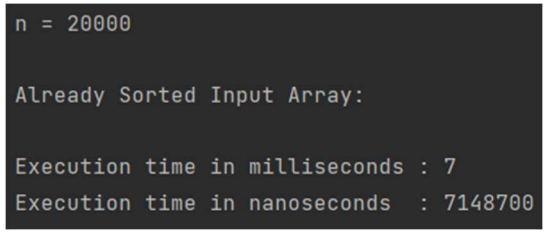
Input Size n	Run # 1	Run # 2	Run #3	Average
10 K	10729700	5354500	14431600	10171933.33
20 K	7875700	71448700	6201400	28508600
40 K	17416600	20701000	16501300	18206300
50 K	23109800	23693400	20169900	22324366.67

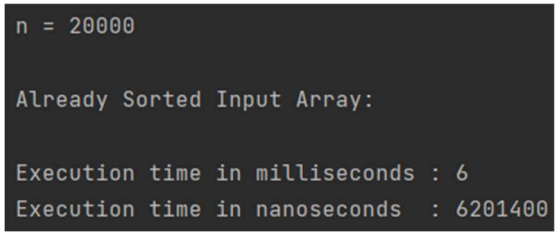


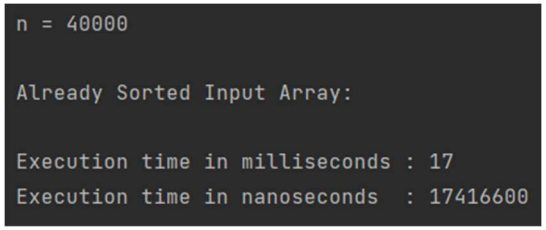


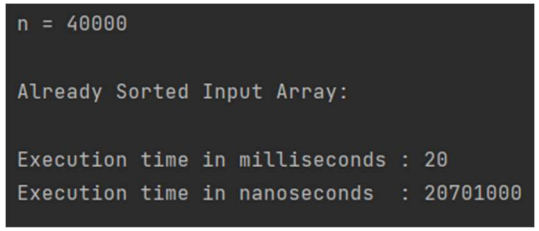


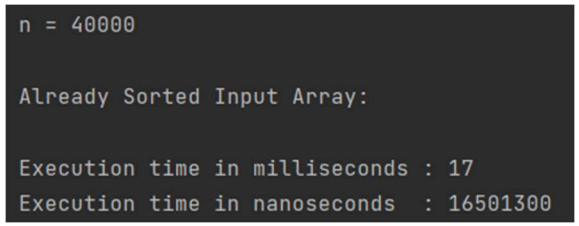












```
n = 50000

Already Sorted Input Array:

Execution time in milliseconds : 23
Execution time in nanoseconds  : 23109800
```

```
n = 50000

Already Sorted Input Array:

Execution time in milliseconds : 24
Execution time in nanoseconds  : 23693400
```

```
n = 50000

Already Sorted Input Array:

Execution time in milliseconds : 20
Execution time in nanoseconds  : 20169900
```

Heap Sort Special Case #1 (Already Sorted Array):

Input Size	Run #1	Run #2	Run #3	Average
10K	4091000	3138800	2924799	3384866
20K	4427001	5389100	4599899	4805333
40K	5969000	6925600	6642000	6512200
50K	9294800	14312000	10909800	11505533

```
n = 10000

Already Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 4091000
```

```
n = 10000

Already Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 3138800
```

```
n = 10000

Already Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 2924799
```

```
n = 20000

Already Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 4427001
```

```
n = 20000

Already Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 5389100
```

```
n = 20000

Already Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 4599899
```

```
n = 40000

Already Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 5969000
```

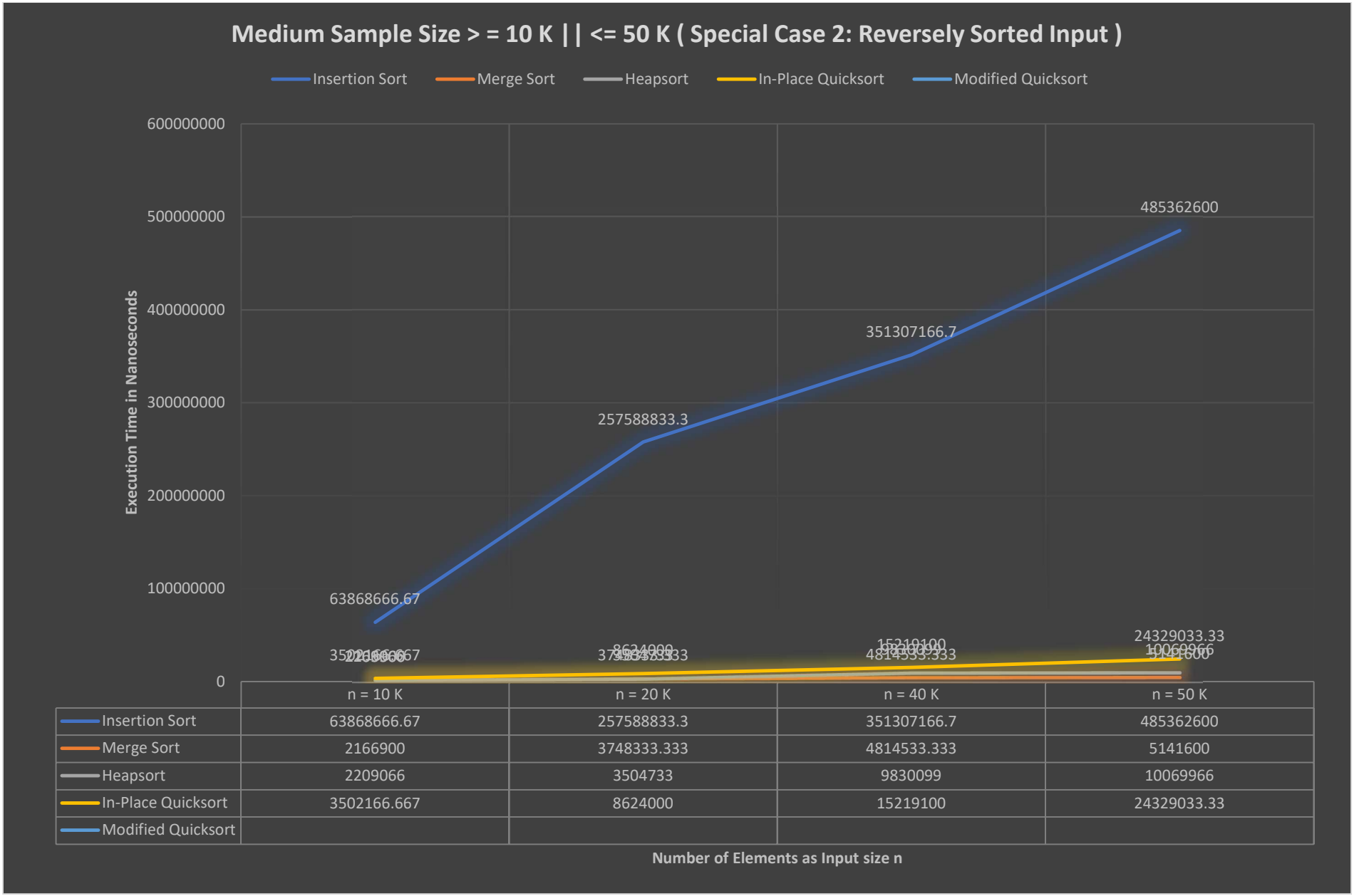
```
n = 40000

Already Sorted Input Array:

Execution time in milliseconds : 15
Execution time in nanoseconds  : 6925600
```

```
n = 40000

Already Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 6642000
```

```
n = 50000

Already Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 9294800
```

```
n = 50000

Already Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 14312000
```

```
n = 50000

Already Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 10909800
```



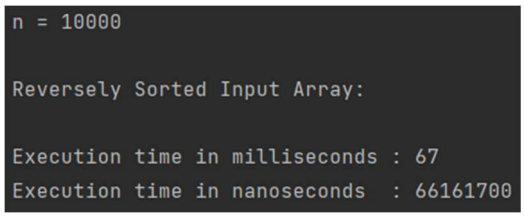
	n = 10 K	n = 20 K	n = 40 K	n = 50 K
Insertion Sort	63868666.67	257588833.3	351307166.7	485362600
Merge Sort	2166900	3748333.333	4814533.333	5141600
Heapsort	2209066	3504733	9830099	10069966
In-Place Quicksort	3502166.667	8624000	15219100	24329033.33
Modified Quicksort				

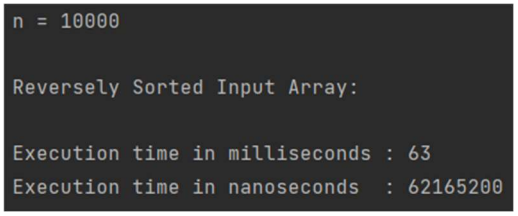
Medium size Reversely Sorted Input Graph Analysis:

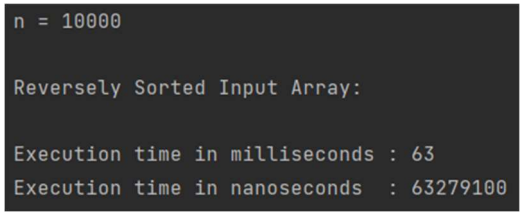
In the medium size special case of reversely sorted input, the insertion sort continues to significantly do worse. In-place quicksort still struggles but when compared to insertion sort almost looks as good as merge and heap sort.

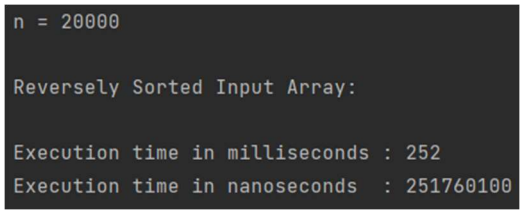
Insertion Sort (Reversely Sorted Input):

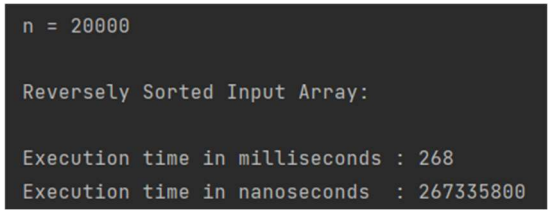
Input Size n	Run # 1	Run # 2	Run #3	Average
10 K	66161700	62165200	63279100	63868666.67
20 K	251760100	267335800	253670600	257588833.3
40 K	330769800	407943200	315208500	351307166.7
50 K	474811200	482682800	498593800	485362600

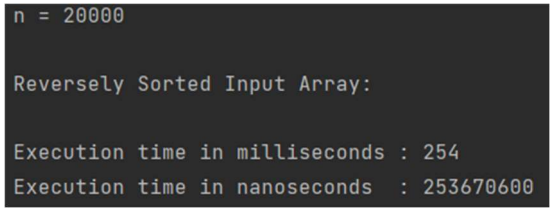


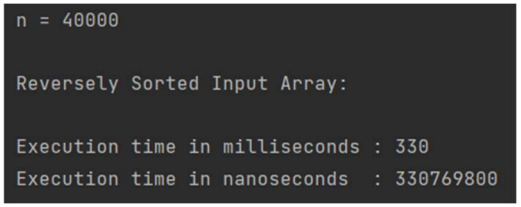


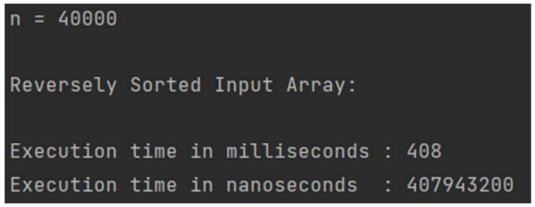


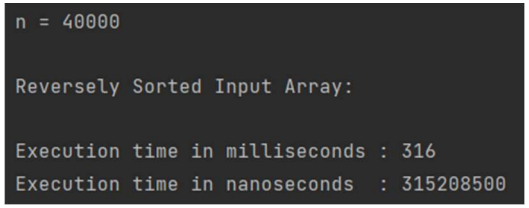


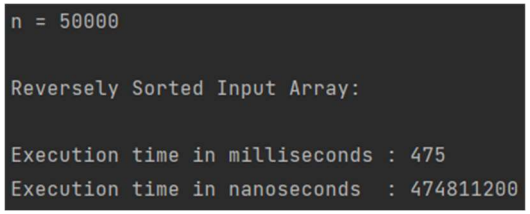


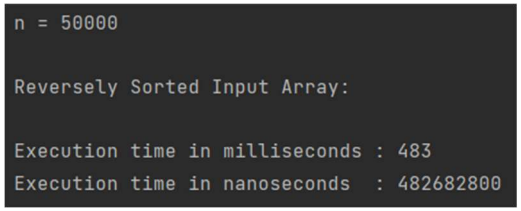


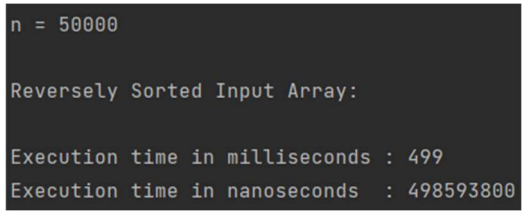






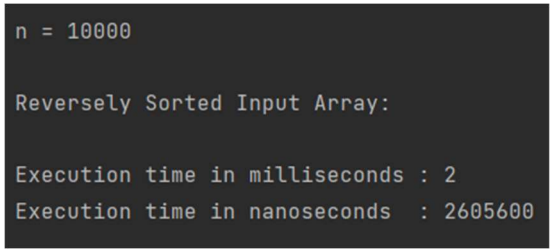


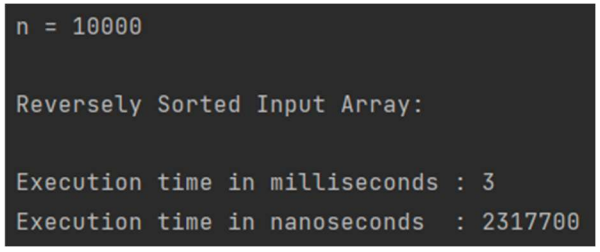


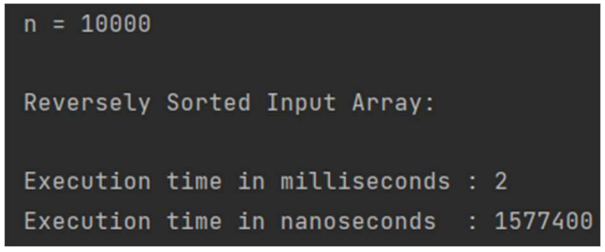


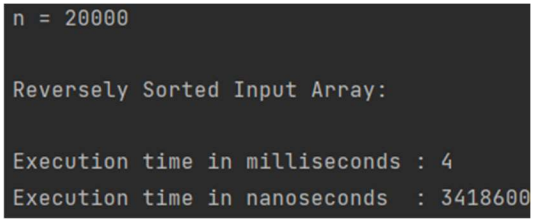
Merge Sort (Reversely Sorted Input):

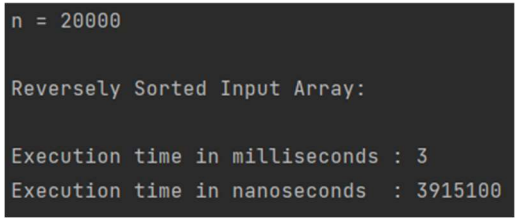
Input Size n	Run # 1	Run # 2	Run #3	Average
10 K	2605600	2317700	1577400	2166900
20 K	3418600	3915100	3911300	3748333.333
40 K	4331200	6136200	3976200	4814533.333
50 K	5174700	5262900	4987200	5141600

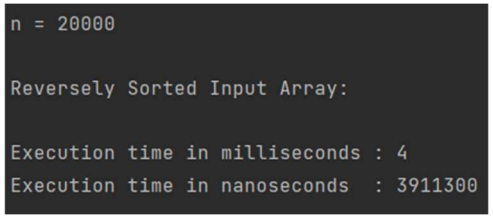












```
n = 40000

Reversely Sorted Input Array:

Execution time in milliseconds : 4
Execution time in nanoseconds  : 4331200
```

```
n = 40000

Reversely Sorted Input Array:

Execution time in milliseconds : 6
Execution time in nanoseconds  : 6136200
```

```
n = 40000

Reversely Sorted Input Array:

Execution time in milliseconds : 4
Execution time in nanoseconds  : 3976200
```

```
n = 50000

Reversely Sorted Input Array:

Execution time in milliseconds : 5
Execution time in nanoseconds  : 5174700
```

```
n = 50000

Reversely Sorted Input Array:

Execution time in milliseconds : 5
Execution time in nanoseconds  : 5262900
```

```
n = 50000

Reversely Sorted Input Array:

Execution time in milliseconds : 5
Execution time in nanoseconds  : 4987200
```

In Place Quick Sort (Reversely Sorted Input):

Input Size n	Run # 1	Run # 2	Run #3	Average
10 K	2613700	2691300	5201500	3502166.667
20 K	8031200	9000200	8840600	8624000
40 K	14736200	14506600	16414500	15219100
50 K	20821900	22757500	29407700	24329033.33

```
n = 10000

Reversely Sorted Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 2613700
```

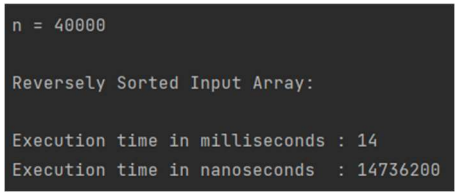
```
n = 10000

Reversely Sorted Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 2691300
```

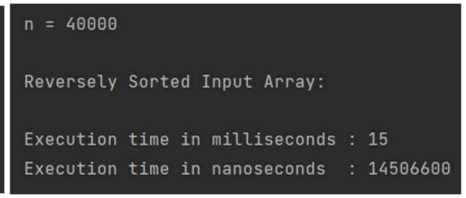
```
n = 10000

Reversely Sorted Input Array:

Execution time in milliseconds : 6
Execution time in nanoseconds  : 5201500
```

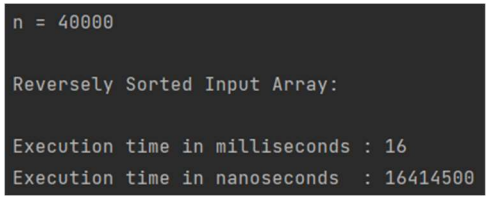
```
n = 20000

Reversely Sorted Input Array:

Execution time in milliseconds : 8
Execution time in nanoseconds  : 8031200
```

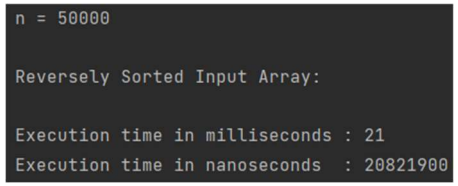
```
n = 20000

Reversely Sorted Input Array:

Execution time in milliseconds : 9
Execution time in nanoseconds  : 9000200
```

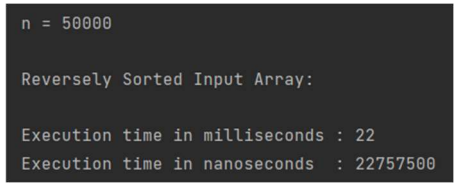
```
n = 20000

Reversely Sorted Input Array:

Execution time in milliseconds : 9
Execution time in nanoseconds  : 8840600
```

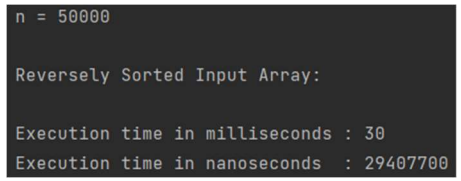






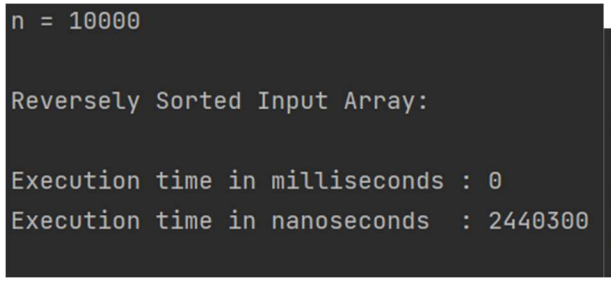


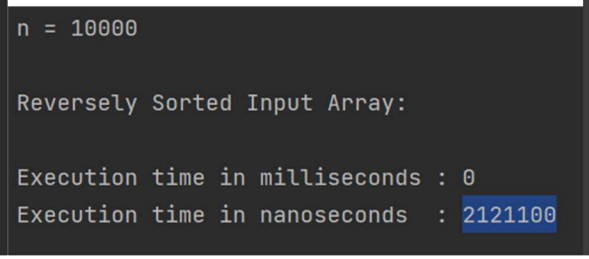


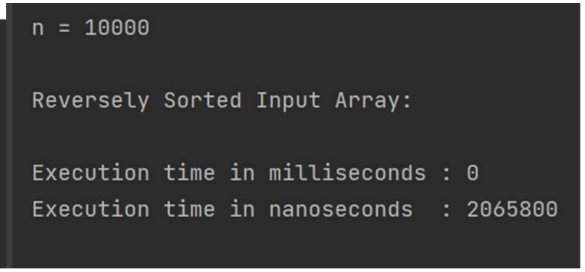


Heap Sort Special Case #2 (Reverse sorted array):

Input Size	Run #1	Run #2	Run #2	Average
10K	2440300	2121100	2065800	2209066
20K	3126900	4074400	3312899	3504733
40K	11313100	8887699	9289500	9830099
50K	10903600	9524300	9782000	10069966







```
n = 20000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 3126900
```

```
n = 20000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 4074400
```

```
n = 20000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 3312899
```

```
n = 40000

Reversely Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 11313100
```

```
n = 40000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 8887699
```

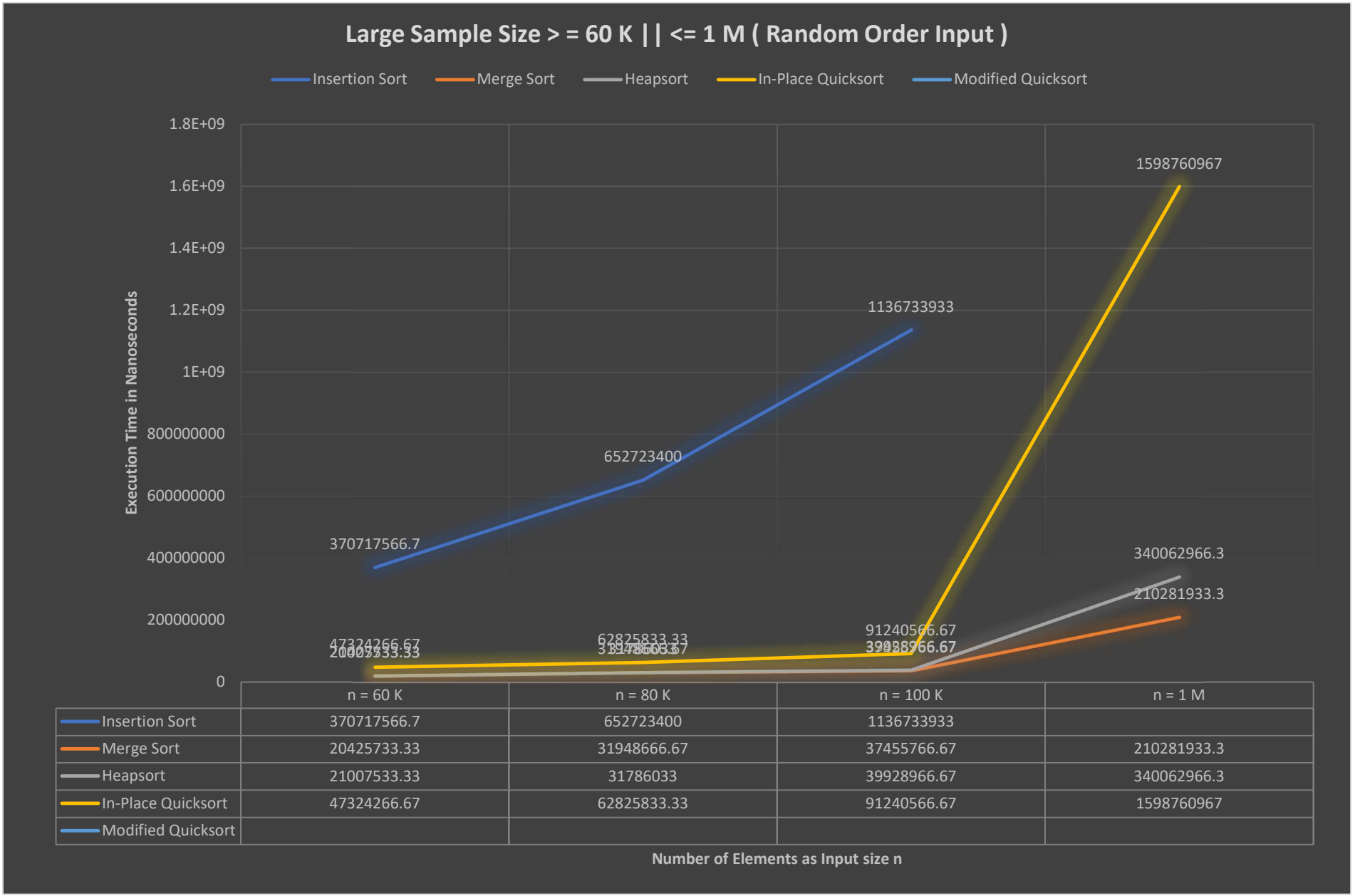
```
n = 40000

Reversely Sorted Input Array:

Execution time in milliseconds : 0
Execution time in nanoseconds  : 9289500
```

```
n = 50000

Reversely Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 10903600
```

```
n = 50000

Reversely Sorted Input Array:

Execution time in milliseconds : 24
Execution time in nanoseconds  : 9524300
```

```
n = 50000

Reversely Sorted Input Array:

Execution time in milliseconds : 15
Execution time in nanoseconds  : 9782000
```

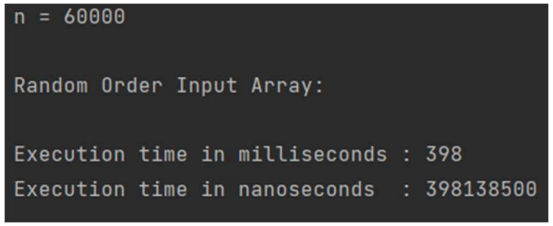
Large Sample Size Execution Results Graphs (> = 60 K || <= 1 M):

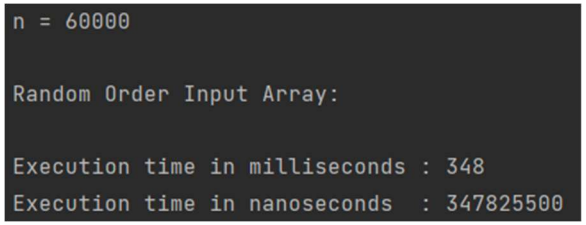


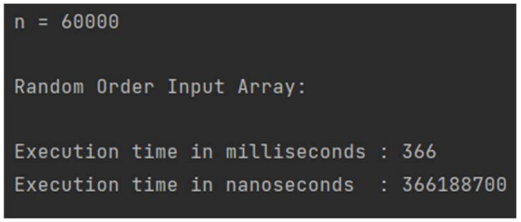
Large size Random Order Input Graph Analysis:

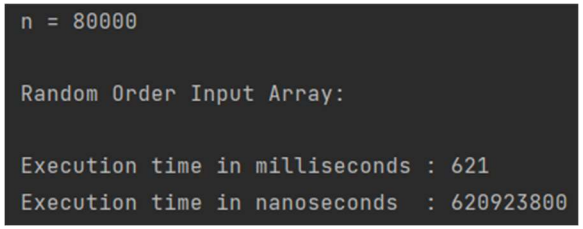
Insertion Sort (Random Order Input):

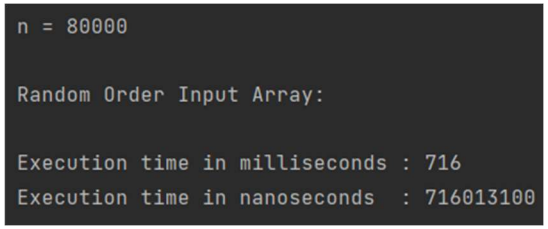
Input Size n	Run # 1	Run # 2	Run #3	Average
60 K	398138500	347825500	366188700	370717566.7
80 K	620923800	716013100	621233300	652723400
100 K	940499300	1221745100	1247957400	1136733933
1 M				

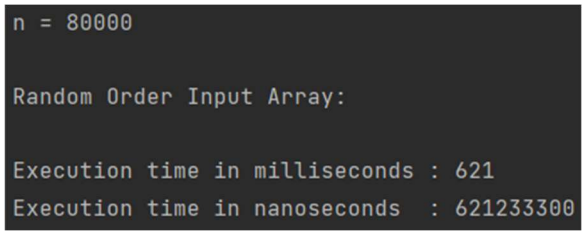


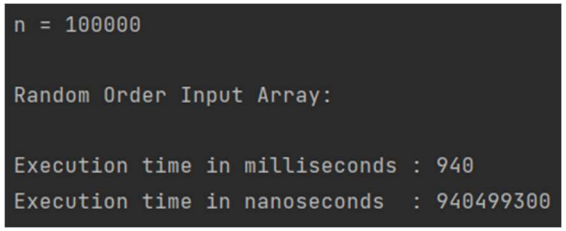


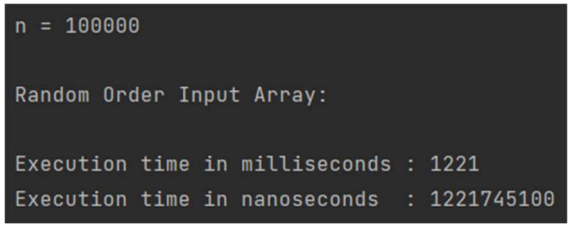


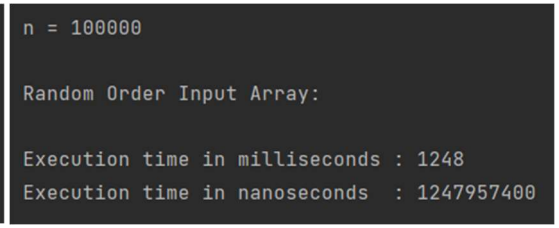






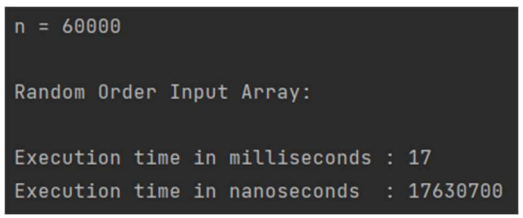


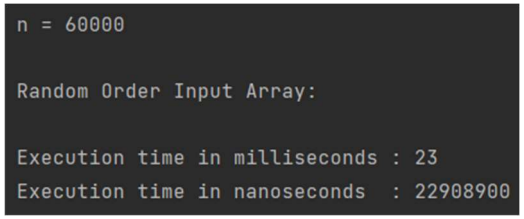


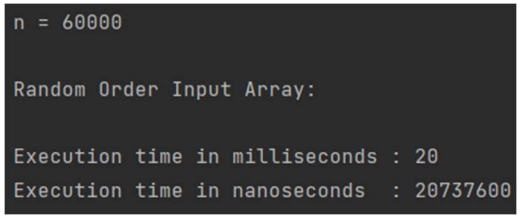


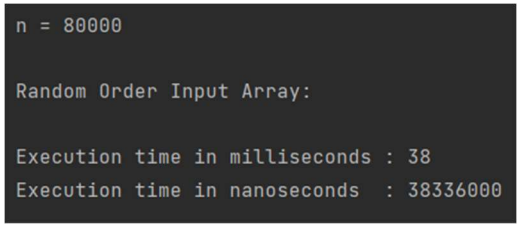
Merge Sort (Random Order Input):

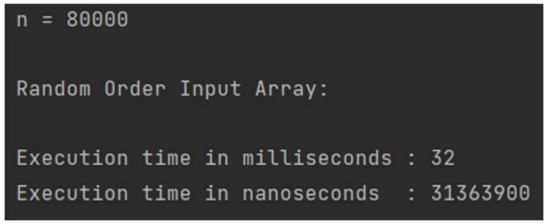
Input Size n	Run # 1	Run # 2	Run #3	Average
60 K	17630700	22908900	20737600	20425733.33
80 K	38336000	31363900	26146100	31948666.67
100 K	30154100	40828100	41385100	37455766.67
1 M	191173900	242738000	196933900	210281933.3

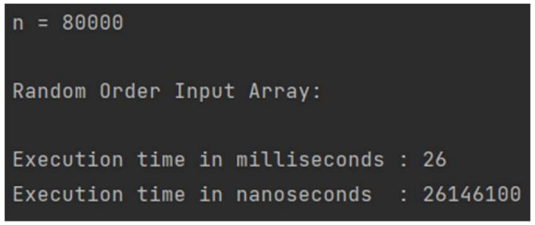


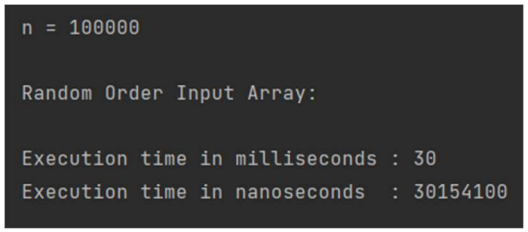


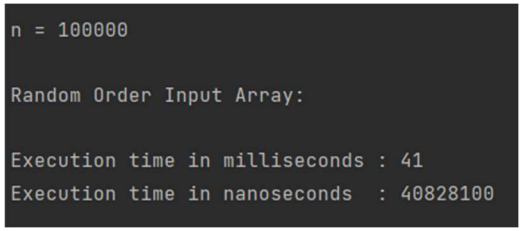


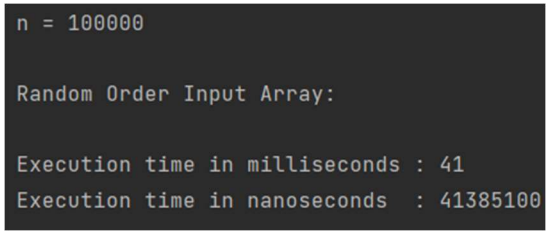


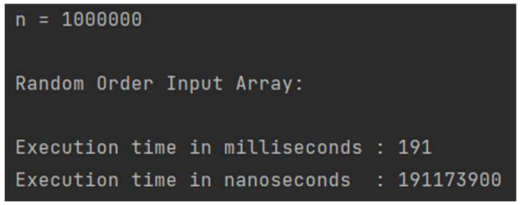


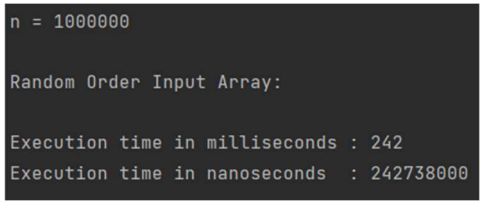


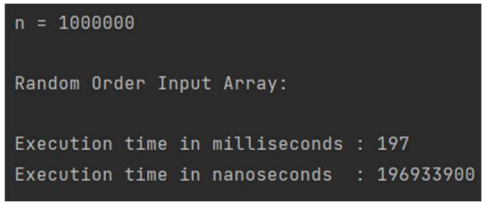






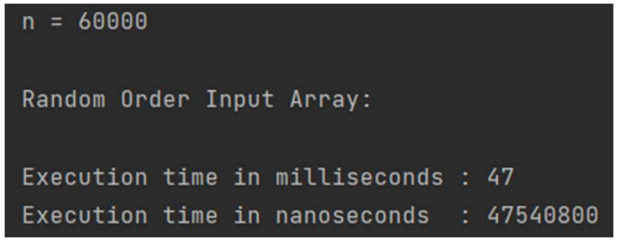


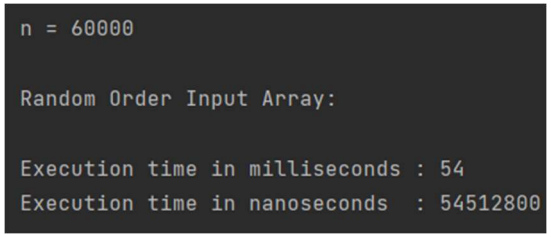


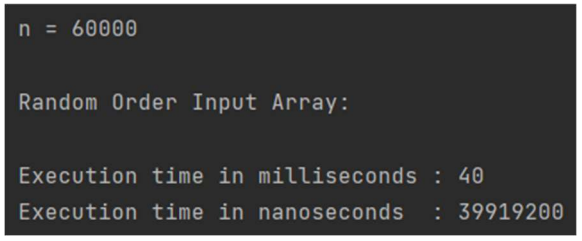


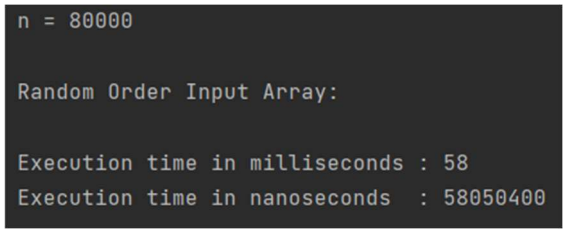
In place Quick Sort (Random Order Input):

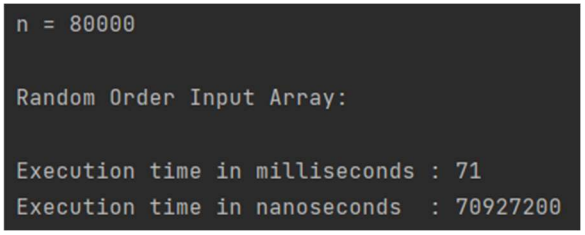
Input Size n	Run # 1	Run # 2	Run #3	Average
60 K	47540800	54512800	39919200	47324266.67
80 K	58050400	70927200	59499900	62825833.33
100 K	69283700	81541700	122896300	91240566.67
1 M	1245538200	1771692400	1779052300	1598760967

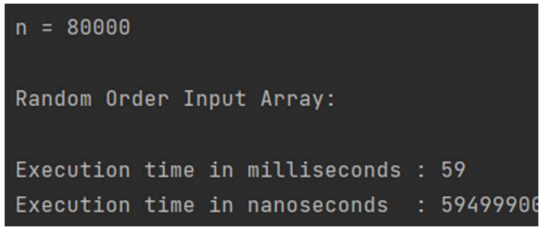


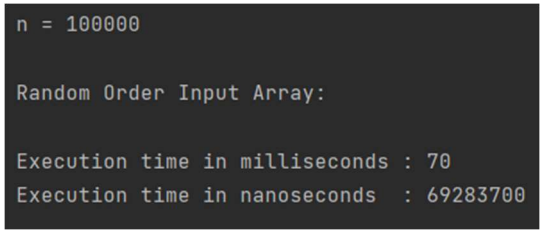


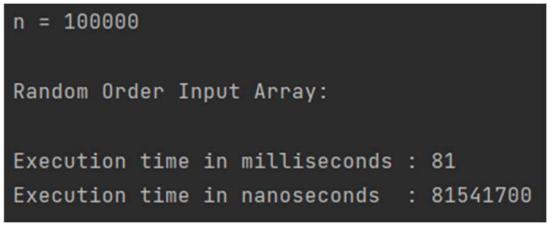


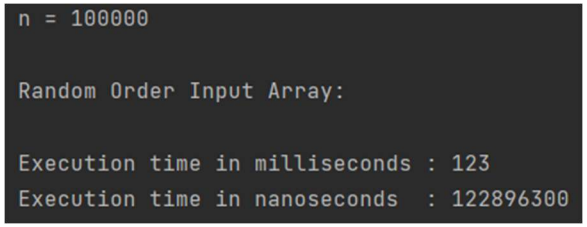


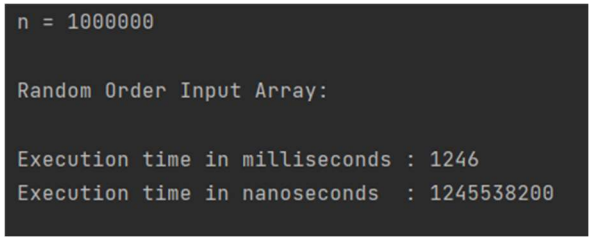


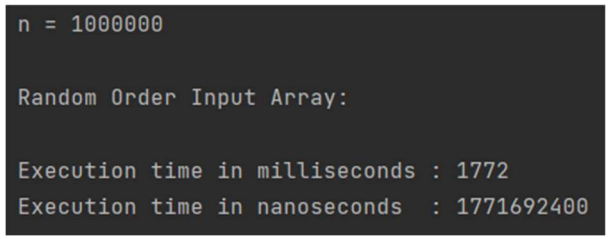


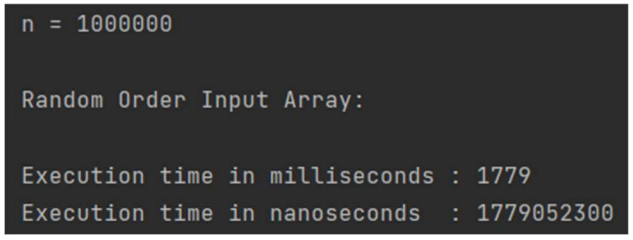






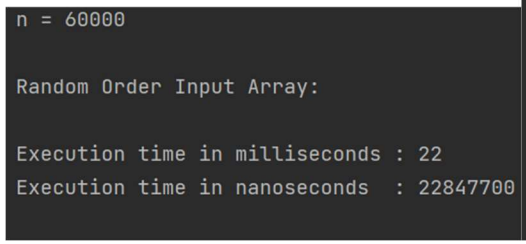


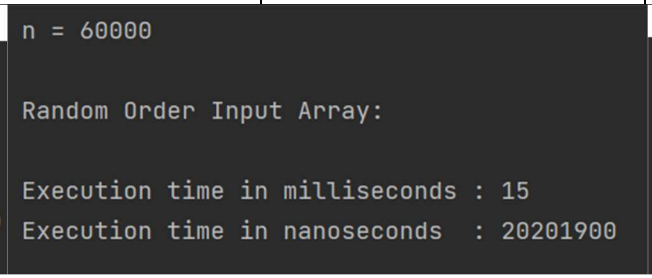


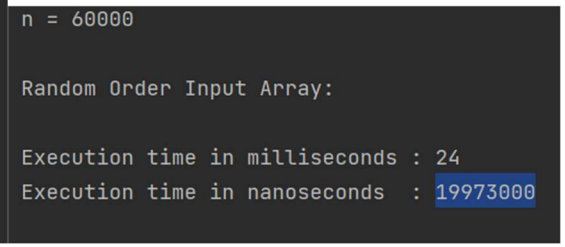


Heap Sort:

Input Size	Run #1	Run #2	Run #3	Average
60K	22847700	20201900	19973000	21007533.33
80K	34017500	27233199	34107400	31786033
100K	47983400	33374800	38428700	39928966.67
1M	328314300	359297600	332576999	340062966.3







```
n = 80000

Random Order Input Array:

Execution time in milliseconds : 31
Execution time in nanoseconds  : 34017500
```

```
n = 80000

Random Order Input Array:

Execution time in milliseconds : 15
Execution time in nanoseconds  : 27233199
```

```
n = 80000

Random Order Input Array:

Execution time in milliseconds : 32
Execution time in nanoseconds  : 34107400
```

```
n = 100000

Random Order Input Array:

Execution time in milliseconds : 57
Execution time in nanoseconds  : 47983400
```

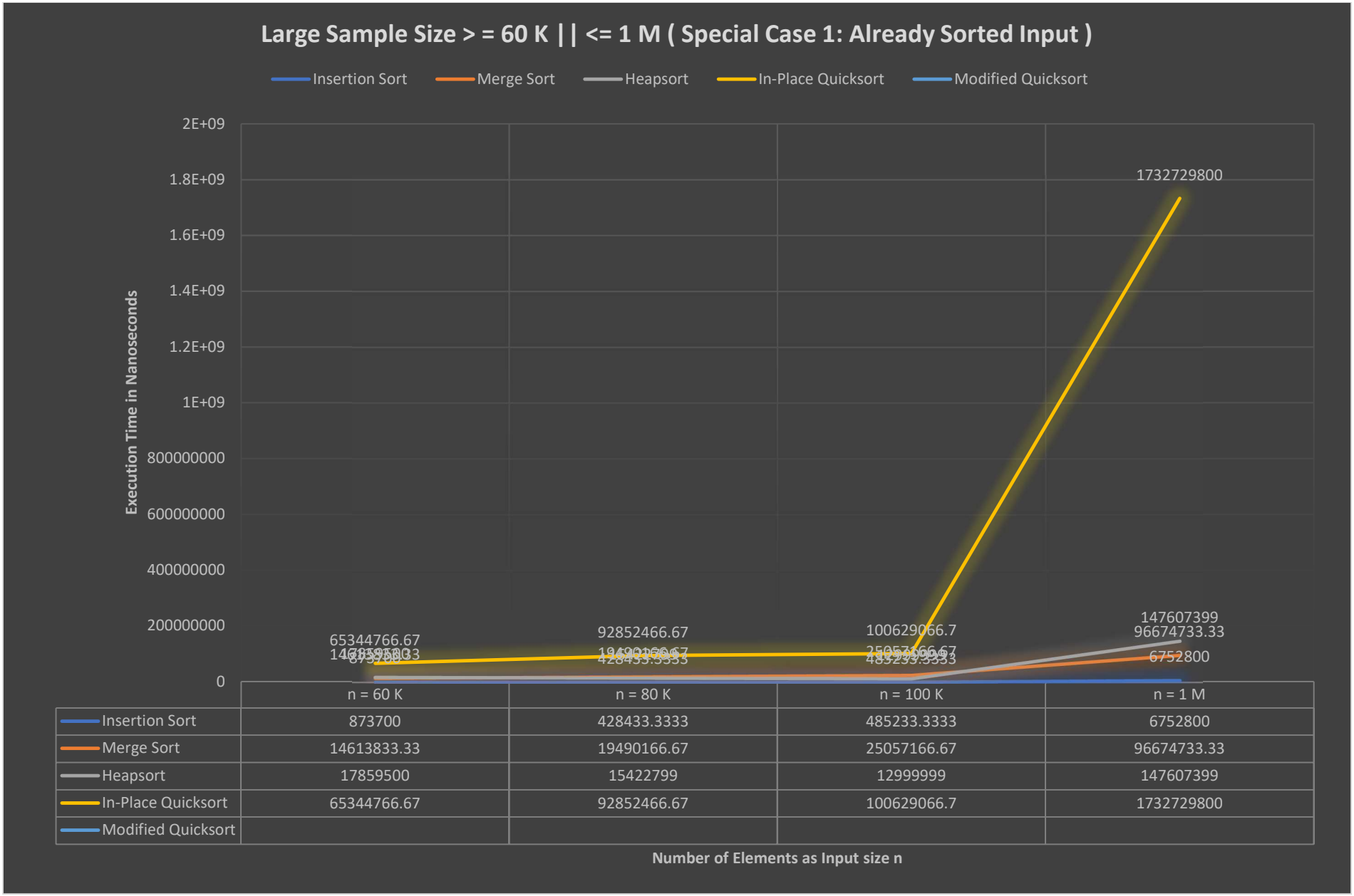
```
n = 100000

Random Order Input Array:

Execution time in milliseconds : 31
Execution time in nanoseconds  : 33374800
```

```
n = 100000

Random Order Input Array:

Execution time in milliseconds : 31
Execution time in nanoseconds  : 38428700
```

```
n = 1000000

Random Order Input Array:

Execution time in milliseconds : 329
Execution time in nanoseconds  : 328314300
```

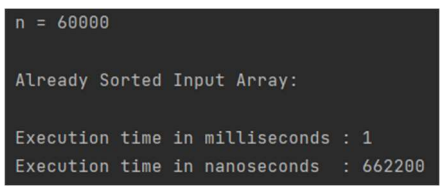
```
n = 1000000

Random Order Input Array:

Execution time in milliseconds : 360
Execution time in nanoseconds  : 359297600
```

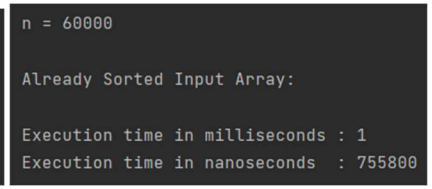
```
n = 1000000

Random Order Input Array:

Execution time in milliseconds : 331
Execution time in nanoseconds  : 332576999
```

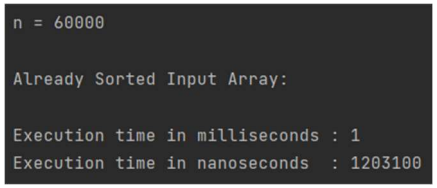

Large Sample Size > = 60 K || <= 1 M (Special Case 1: Already Sorted Input)
Insertion Sort
Merge Sort
Heapsort
In-Place Quicksort
Modified Quicksort
2E+09
1.8E+09
1.6E+09
1.4E+09
1.2E+09
1E+09
800000000
600000000
400000000
200000000
0
Execution Time in Nanoseconds
1732729800
147607399
96674733.33
92852466.67
65344766.67
6752800
100629066.7
n = 60 K
n = 80 K
n = 100 K
n = 1 M
Insertion Sort 873700 428433.3333 485233.3333 6752800
Merge Sort 14613833.33 19490166.67 25057166.67 96674733.33
Heapsort 17859500 15422799 12999999 147607399
In-Place Quicksort 65344766.67 92852466.67 100629066.7 1732729800
Modified Quicksort
Number of Elements as Input size n

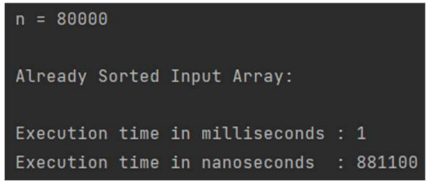
Insertion Sort (Already Sorted Input):

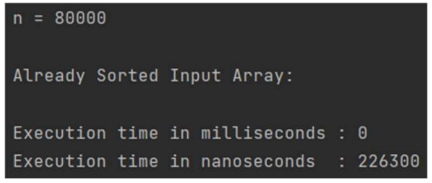
Input Size n	Run # 1	Run # 2	Run #3	Average
60 K	662200	755800	1203100	873700
80 K	881100	226300	177900	428433.3333
100 K	953700	188300	313700	485233.3333
1 M	5486200	8725900	6046300	6752800

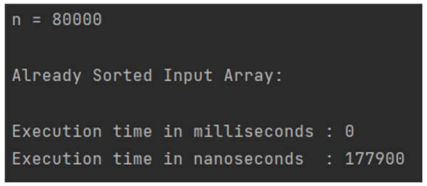


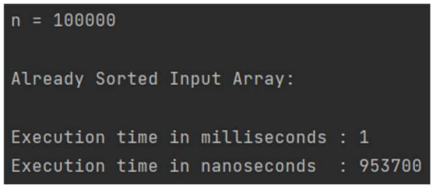


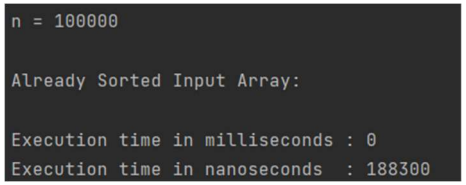


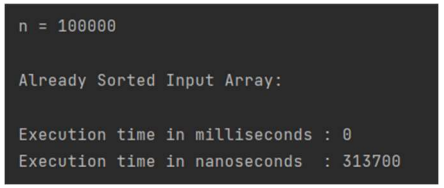


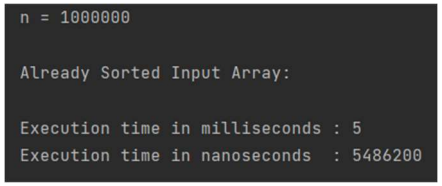


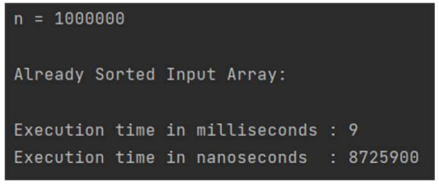


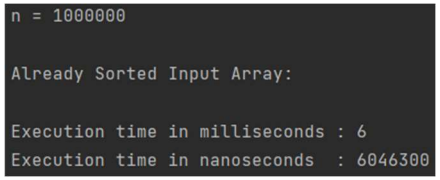






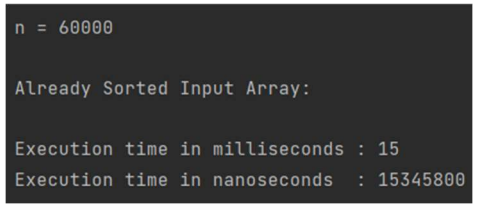


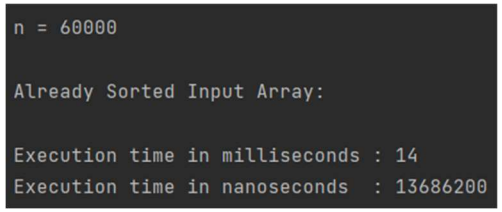


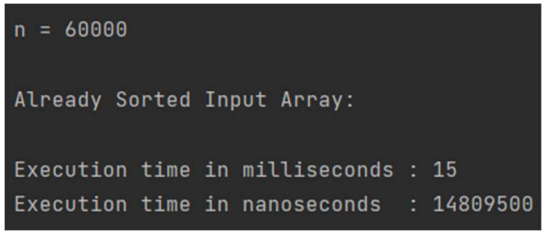


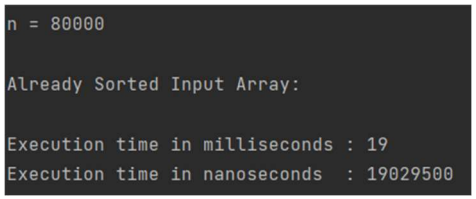
Merge Sort (Already Sorted Input):

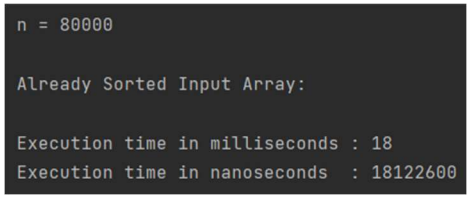
Input Size n	Run # 1	Run # 2	Run #3	Average
60 K	15345800	13686200	14809500	14613833.33
80 K	19029500	18122600	21318400	19490166.67
100 K	21528200	25879900	27763400	25057166.67
1 M	83941800	99615300	106467100	96674733.33

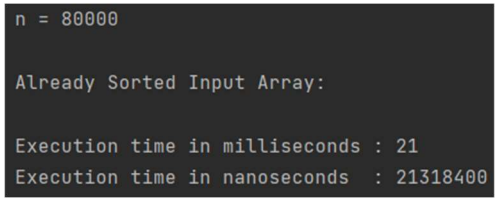


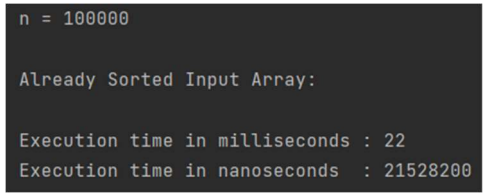


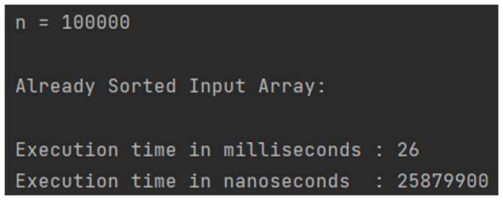


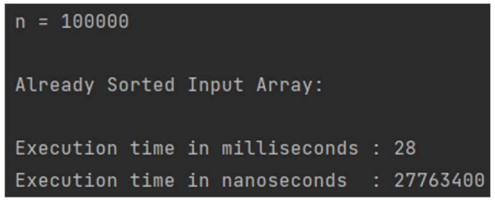


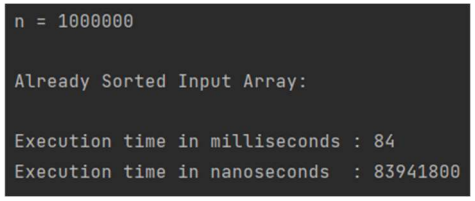


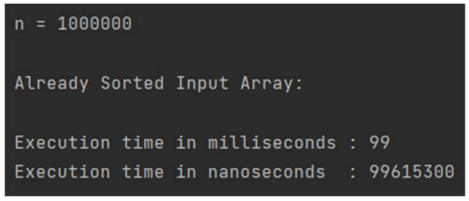


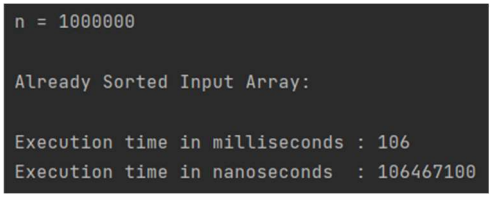












In place Quick Sort (Already Sorted Input):

Input Size n	Run # 1	Run # 2	Run #3	Average
60 K	70601900	73674900	51757500	65344766.67
80 K	102043300	95630400	80883700	92852466.67
100 K	105932100	102476200	93478900	100629066.7
1 M	1374436900	2374246500	1449506000	1732729800

```
n = 60000

Already Sorted Input Array:

Execution time in milliseconds : 71
Execution time in nanoseconds  : 70601900
```

```
n = 60000

Already Sorted Input Array:

Execution time in milliseconds : 74
Execution time in nanoseconds  : 73674900
```

```
n = 60000

Already Sorted Input Array:

Execution time in milliseconds : 52
Execution time in nanoseconds  : 51757500
```

```
n = 80000

Already Sorted Input Array:

Execution time in milliseconds : 102
Execution time in nanoseconds  : 102043300
```

```
n = 80000

Already Sorted Input Array:

Execution time in milliseconds : 95
Execution time in nanoseconds  : 95630400
```

```
n = 80000

Already Sorted Input Array:

Execution time in milliseconds : 80
Execution time in nanoseconds  : 80883700
```

```
n = 100000

Already Sorted Input Array:

Execution time in milliseconds : 106
Execution time in nanoseconds  : 105932100
```

```
n = 100000

Already Sorted Input Array:

Execution time in milliseconds : 102
Execution time in nanoseconds  : 102476200
```

```
n = 100000

Already Sorted Input Array:

Execution time in milliseconds : 93
Execution time in nanoseconds  : 93478900
```

```
n = 1000000

Already Sorted Input Array:

Execution time in milliseconds : 1375
Execution time in nanoseconds  : 1374436900
```

```
n = 1000000

Already Sorted Input Array:

Execution time in milliseconds : 2374
Execution time in nanoseconds  : 2374246500
```

```
n = 1000000

Already Sorted Input Array:

Execution time in milliseconds : 1449
Execution time in nanoseconds  : 1449506000
```

Heap Sort Special Case #1: Already Sorted array

Input Size	Run #1	Run #2	Run #2	Average
60K	19055601	15929299	18593600	17859500
80K	17305699	14614600	14348099	15422799
100K	11476999	14463400	13059600	12999999
1M	146953899	145098400	150769900	147607399

```
n = 60000

Already Sorted Input Array:

Execution time in milliseconds : 15
Execution time in nanoseconds  : 19055601

----------------------------------------
```

```
n = 60000

Already Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 15929299
```

```
n = 60000

Already Sorted Input Array:

Execution time in milliseconds : 31
Execution time in nanoseconds  : 18593600
```

```
n = 80000

Already Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 17305699
```

```
n = 80000

Already Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 14614600
```

```
n = 80000

Already Sorted Input Array:

Execution time in milliseconds : 8
Execution time in nanoseconds  : 14348099

----------------------------------------
```

```
n = 100000

Already Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 11476999
```

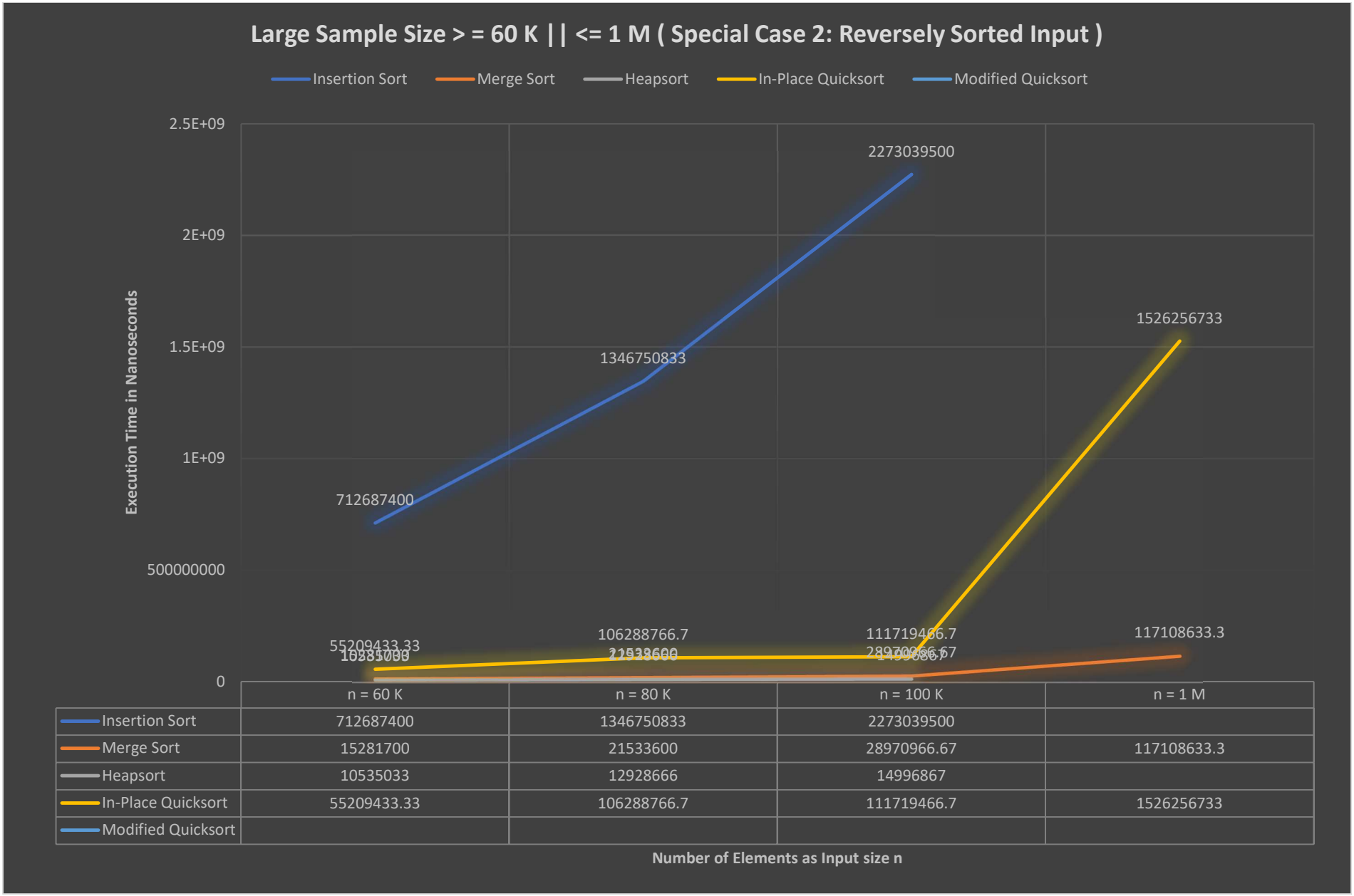
```
n = 100000

Already Sorted Input Array:

Execution time in milliseconds : 15
Execution time in nanoseconds  : 14463400
```

```
n = 100000

Already Sorted Input Array:

Execution time in milliseconds : 15
Execution time in nanoseconds  : 13059600
```

```
n = 1000000

Already Sorted Input Array:

Execution time in milliseconds : 157
Execution time in nanoseconds  : 146953899
```

```
n = 1000000

Already Sorted Input Array:

Execution time in milliseconds : 141
Execution time in nanoseconds  : 145098400
```

```
n = 1000000

Already Sorted Input Array:

Execution time in milliseconds : 157
Execution time in nanoseconds  : 150769900
```


Large Sample Size > = 60 K || <= 1 M (Special Case 2: Reversely Sorted Input)
Insertion Sort
Merge Sort
Heapsort
In-Place Quicksort
Modified Quicksort
2.5E+09
2E+09
1.5E+09
1E+09
500000000
0
Execution Time in Nanoseconds
2273039500
1346750833
712687400
1526256733
106288766.7
111719466.7
117108633.3
55209433.33
n = 60 K
n = 80 K
n = 100 K
n = 1 M
Insertion Sort 712687400 1346750833 2273039500
Merge Sort 15281700 21533600 28970966.67 117108633.3
Heapsort 10535033 12928666 14996867
In-Place Quicksort 55209433.33 106288766.7 111719466.7 1526256733
Modified Quicksort
Number of Elements as Input size n

Insertion Sort (Reversely Sorted Input):

Input Size n	Run # 1	Run # 2	Run #3	Average
60 K	668815200	733397000	735850000	712687400
80 K	1308373400	1492046300	1239832800	1346750833
100 K	2121994600	1933651000	2763472900	2273039500
1 M	203586661900	207244867600	204927491800	205253007100

```
n = 60000

Reversely Sorted Input Array:

Execution time in milliseconds : 669
Execution time in nanoseconds  : 668815200
```

```
n = 60000

Reversely Sorted Input Array:

Execution time in milliseconds : 734
Execution time in nanoseconds  : 733397000
```

```
n = 60000

Reversely Sorted Input Array:

Execution time in milliseconds : 736
Execution time in nanoseconds  : 735850000
```

```
n = 80000

Reversely Sorted Input Array:

Execution time in milliseconds : 1308
Execution time in nanoseconds  : 1308373400
```

```
n = 80000

Reversely Sorted Input Array:

Execution time in milliseconds : 1492
Execution time in nanoseconds  : 1492046300
```

```
n = 80000

Reversely Sorted Input Array:

Execution time in milliseconds : 1239
Execution time in nanoseconds  : 1239832800
```

```
n = 100000

Reversely Sorted Input Array:

Execution time in milliseconds : 2122
Execution time in nanoseconds  : 2121994600
```

```
n = 100000

Reversely Sorted Input Array:

Execution time in milliseconds : 1934
Execution time in nanoseconds  : 1933651000
```

```
n = 100000

Reversely Sorted Input Array:

Execution time in milliseconds : 2763
Execution time in nanoseconds  : 2763472900
```

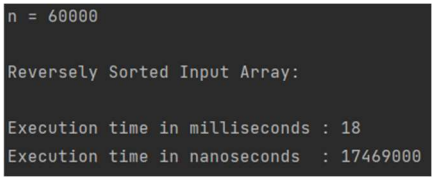
```
n = 1000000

Reversely Sorted Input Array:

Execution time in milliseconds : 203587
Execution time in nanoseconds  : 203586661900
```

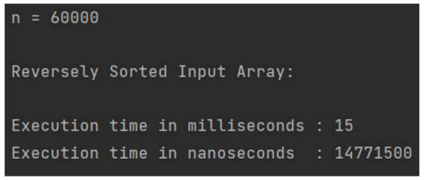
```
n = 1000000

Reversely Sorted Input Array:

Execution time in milliseconds : 207245
Execution time in nanoseconds  : 207244867600
```

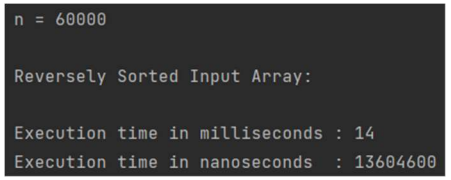
```
n = 1000000

Reversely Sorted Input Array:

Execution time in milliseconds : 204928
Execution time in nanoseconds  : 204927491800
```

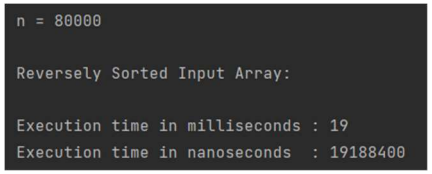
Merge Sort (Reversely Sorted Input):

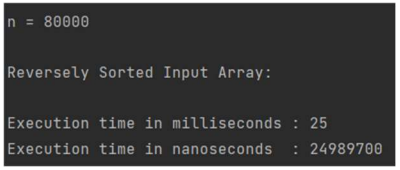
Input Size n	Run # 1	Run # 2	Run #3	Average
60 K	17469000	14771500	13604600	15281700
80 K	19188400	24989700	20422700	21533600
100 K	24658800	37225900	25028200	28970966.67
1 M	132468100	105238400	113619400	117108633.3

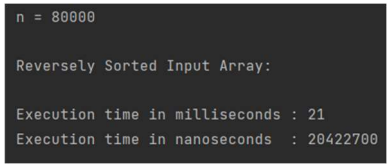


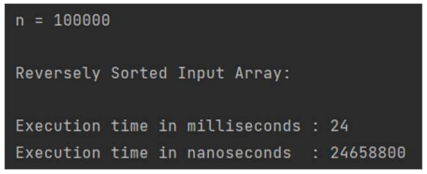


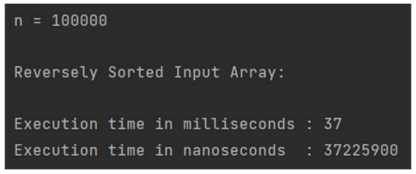


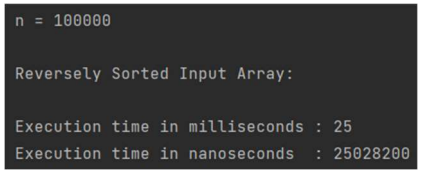


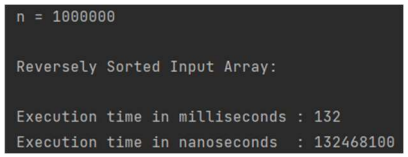


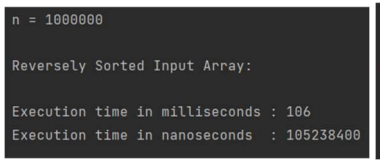


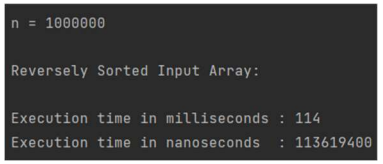






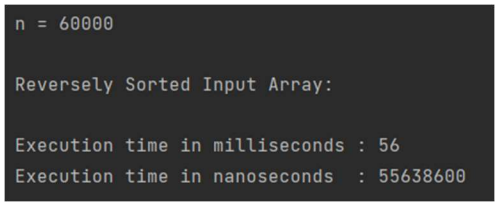


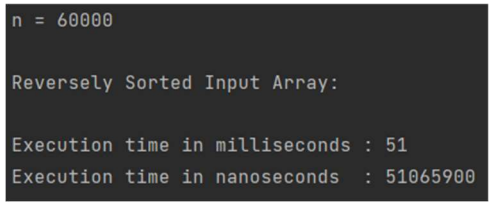


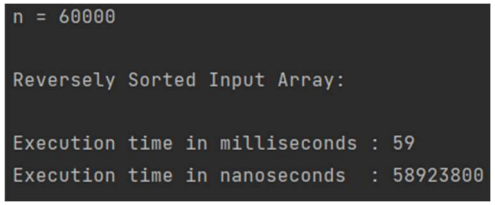


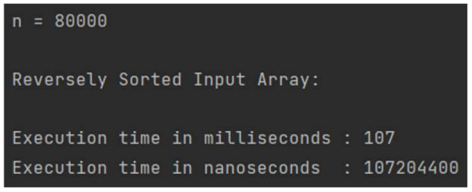
In place Quick Sort (Reversely Sorted Input):

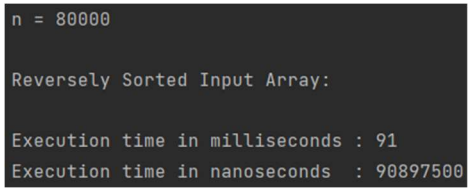
Input Size n	Run # 1	Run # 2	Run #3	Average
60 K	55638600	51065900	58923800	55209433.33
80 K	107204400	90897500	120764400	106288766.7
100 K	103132000	112217400	119809000	111719466.7
1 M	1328513000	1327086900	1923170300	1526256733

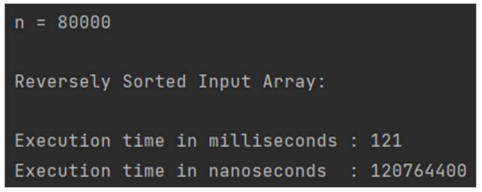


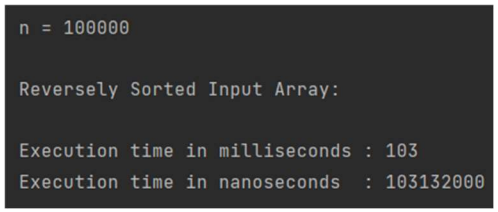


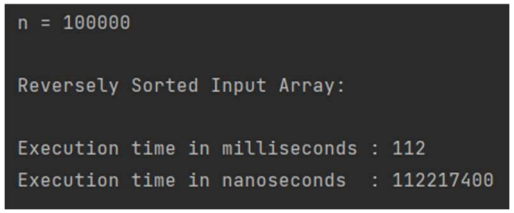


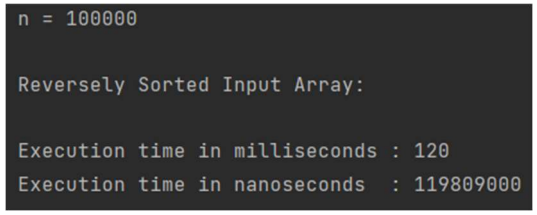


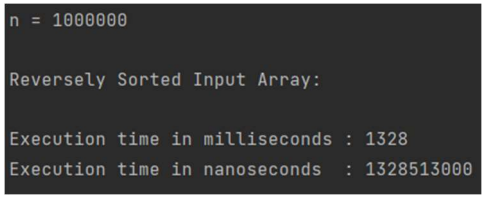


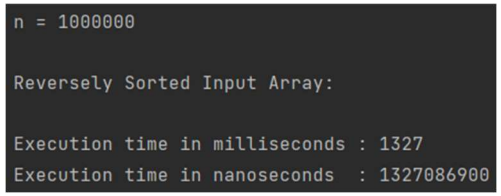


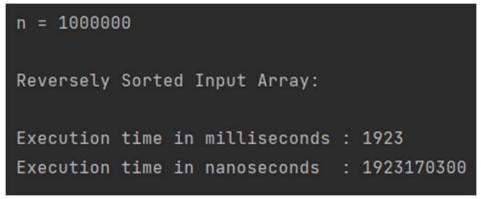












Heap Sort (Reverse Sorted array):

Input Size	Run #1	Run #2	Run #3	Average
60K	9115900	13962200	8526999	10535033
80K	14543701	12818100	11424199	12928666
100K	15422101	14948900	14619600	14996867

```
n = 60000

Reversely Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 9115900
```

```
n = 60000

Reversely Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 13962200
```

```
n = 60000

Reversely Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 8526999
```

```
n = 80000

Reversely Sorted Input Array:

Execution time in milliseconds : 3
Execution time in nanoseconds  : 14543701
```

```
n = 80000

Reversely Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 12818100
```

```
n = 80000

Reversely Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 11424199
```

```
n = 100000

Reversely Sorted Input Array:

Execution time in milliseconds : 15
Execution time in nanoseconds  : 15422101
```

```
n = 100000

Reversely Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 14948900
```

```
n = 100000

Reversely Sorted Input Array:

Execution time in milliseconds : 16
Execution time in nanoseconds  : 14619600
```

Extra Large Sample Size Execution Results Graphs (> = 2 M || <= 50 M):

// in place quicksort

```
n = 50000000

Already Sorted Input Array:

Execution time in milliseconds : 471143
Execution time in nanoseconds  : 471142602300

----------------------------------------

Process finished with exit code 0
```