Program Structures and Algorithms

Spring 2023 (SEC – 01)

NAME: Aravind Dasarathy

NUID: 002130918

Task:

The task is split into three parts:

a. To implement three methods – repeat(), getClock(), and toMilliSecs() in Timer class.
b. To implement InsertionSort in the InsertionSort class.
c. To perform and record observations for insertion sort with different types of input arrays – sorted, reverse sorted, partially sorted, and random.

Unit Tests:

All test cases are passed with respect to the Benchmark project.

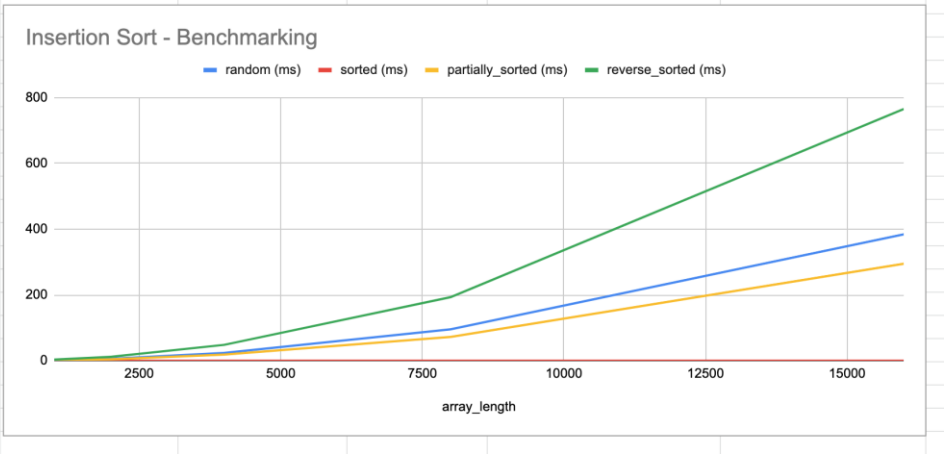
1. Timer class:

2. InsertionSort class:

Timing and Graphical Observation:

I've created a class named 'InsertionSortBenchmark.java' to perform the timing observations.
(./src/main/java/edu/neu/coe/info6205/sort/elementary/InsertionSortBenchmark.java)

array_length	random (ms)	sorted (ms)	partially_sorted (ms)	reverse_sorted (ms)
1000	2.16	0.25	1.71	3.38
2000	5.89	0.18	4.41	12.1
4000	23.49	0.17	18.74	48.26
8000	95.55	0.22	72.39	193.18
16000	384.4	0.32	294.91	765.55



Logical Conclusion:

From the timing observations, for the same array length, the order of sorting is given by:

1. Input array being ordered.
2. Input array being partially ordered.
3. Input array being reversed (descending order)

Where 1 being the fastest sorting and 3 being the slowest sorting.

- When the input is ordered, we can say that it is the best case for Insertion Sort. Because the insertion sort will be a one-pass algorithm and no swaps will be happening. Hence, Insertion sort in an ordered array will be O(n) where 'n' is the length of the input array.
- When the input array is fully reversed or is arranged in descending order, we insert each element at the beginning of the array (for each level of iteration). Hence, this is approximately equal to O(N^2).
- The performance in the case of a partially sorted array is between 1st and the 2nd point.
- For a randomly organized array, it cannot be listed as the order is indeterministic.