

Venkata Srinivas Kompally (NUID : 002137855)

Program Structures & Algorithms

Fall 2021

Assignment No. 2

⊙ **Task:**

Part 1) Implement three methods of class called Timer. Timer is invoked from a class called Benchmark_Timer which implements the Benchmark interface.

Part 2) Implement InsertionSort (in the InsertionSort class) by simply looking up the insertion code used by Arrays.sort.

Part 3) Implement a main program (or you could do it via your own unit tests) to actually run the following benchmarks: measure the running times of this sort, using four different initial array ordering situations: random, ordered, partially-ordered and reverse-ordered.

⊙ **Relationship Conclusion:**

The Order of Growth for Randomly Ordered Array of Size N is $\approx N^{1.36}$

The Order of Growth for Ordered Array of Size N is $\approx N^{0.80}$

The Order of Growth for Reverse Ordered Array of Size N is $\approx N^{1.53}$

The Order of Growth for Partially Ordered Array of Size N is $\approx N^{1.09}$

The Order of growth, based on running time of insertion sort is:

Ordered < Partially Ordered < Randomly Ordered < Reverse Ordered

⊙ **Evidence to support the conclusion:**

1. Output (Snapshot of Code output in the terminal)

The screenshot shows an IDE with the following components:

- Top Pane (Source Code):** Displays `Benchmark_Timer.java`. The code includes a `for` loop that generates a random array of 200 integers and then calls `Benchmark_Timer` with a description of the sort and array type. The `Benchmark_Timer` class is also visible in the background.
- Bottom Pane (Output):** Shows the execution results. It indicates that the program is running Insertion Sort on a Randomly Ordered Array with 50 runs. The output lists the execution time for different array sizes (200, 400, 800, 1600, 3200) and compares the results for randomly ordered and ordered arrays. The ordered array consistently shows much lower execution times.

Output Details:

- Randomly Ordered Array (50 runs):**
 - N: 200 Time: 0.478606
 - N: 400 Time: 0.5790836326530613
 - N: 800 Time: 1.1879468775510205
 - N: 1600 Time: 4.5778774897959185
 - N: 3200 Time: 20.891424612244897
- Ordered Array (50 runs):**
 - N: 200 Time: 0.007818387755102041
 - N: 400 Time: 0.010361244897959183
 - N: 800 Time: 0.021828714285714285
 - N: 1600 Time: 0.041308
 - N: 3200 Time: 0.07266730612244898

```

for(int num = 200; num < 4000; num += 200) {
    Benchmark_Timer<Integer[]> benchmark_Timer = new Benchmark_Timer<>("Insertion Sort - Ordered Array", false, null,
        Supplier<Integer[]> supplier = () -> {

```

```

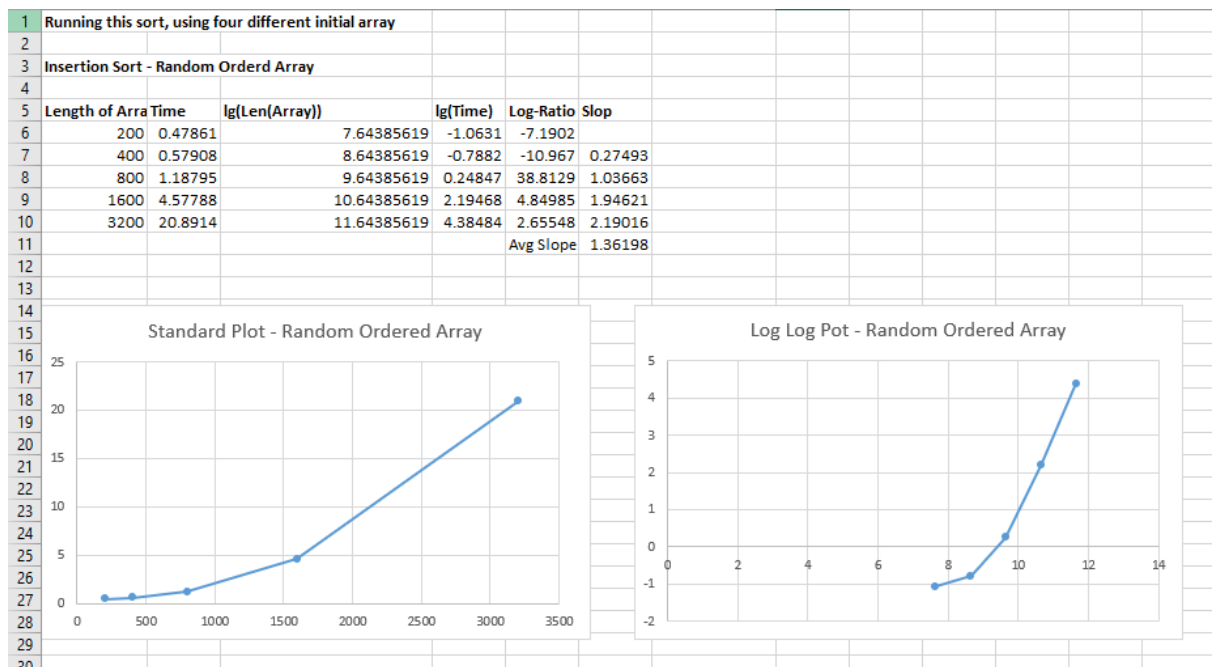
2021-09-26 17:06:47 INFO Benchmark_Timer - Begin run: Insertion Sort - Reverse Ordered Array with 50 runs
N: 200 Time: 0.25098604081632656
2021-09-26 17:06:47 INFO Benchmark_Timer - Begin run: Insertion Sort - Reverse Ordered Array with 50 runs
N: 400 Time: 0.9370674693877551
2021-09-26 17:06:47 INFO Benchmark_Timer - Begin run: Insertion Sort - Reverse Ordered Array with 50 runs
N: 800 Time: 2.524761306122449
2021-09-26 17:06:47 INFO Benchmark_Timer - Begin run: Insertion Sort - Reverse Ordered Array with 50 runs
N: 1600 Time: 10.602795979591837
2021-09-26 17:06:48 INFO Benchmark_Timer - Begin run: Insertion Sort - Reverse Ordered Array with 50 runs
N: 3200 Time: 53.71010002040817

2021-09-26 17:06:51 INFO Benchmark_Timer - Begin run: Insertion Sort - Partially Ordered Array with 50 runs
N: 200 Time: 0.032936816326530616
2021-09-26 17:06:51 INFO Benchmark_Timer - Begin run: Insertion Sort - Partially Ordered Array with 50 runs
N: 400 Time: 0.1248837551020408
2021-09-26 17:06:51 INFO Benchmark_Timer - Begin run: Insertion Sort - Partially Ordered Array with 50 runs
N: 800 Time: 0.49631439775510206
2021-09-26 17:06:51 INFO Benchmark_Timer - Begin run: Insertion Sort - Partially Ordered Array with 50 runs
N: 1600 Time: 1.813942591836735
2021-09-26 17:06:51 INFO Benchmark_Timer - Begin run: Insertion Sort - Partially Ordered Array with 50 runs
N: 3200 Time: 5.61126324489796

```

2. Graphical Representation

Random Ordered Array:



Initial input size is 200 and increased upto 3200. Calculated the running time of the sorting algorithm Random Orderly.

Using Doubling Hypothesis:

For Function $T(N) = aN^b$

N = Input Size

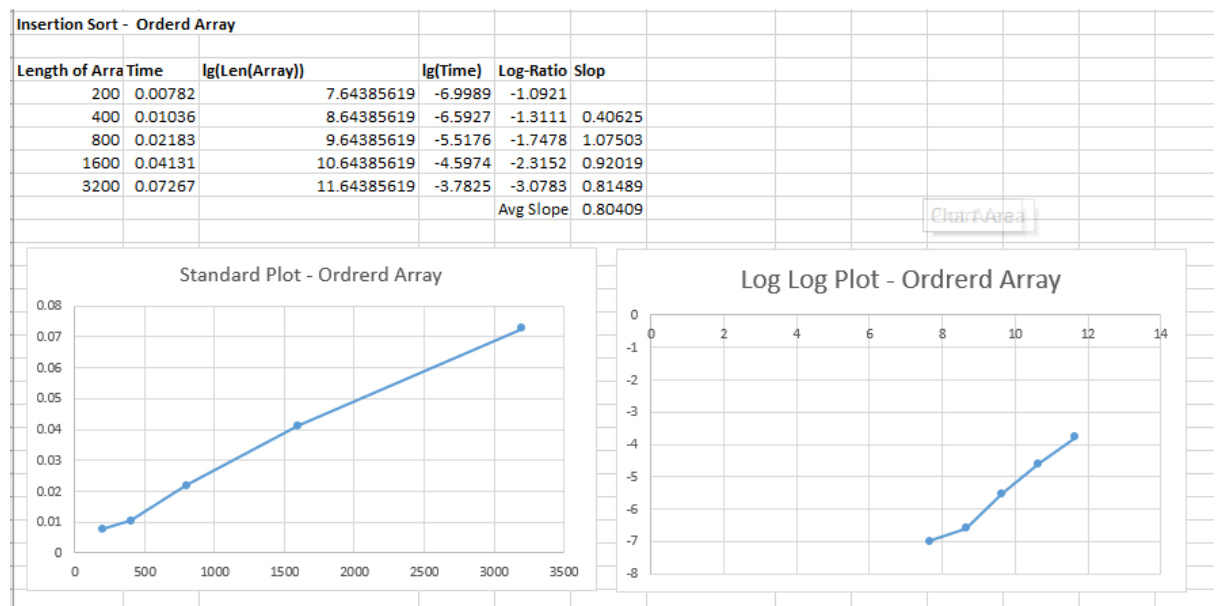
a = Constant

b = slope of the log-log graph

The average slope for Randomly Ordered Array is: 1.36

The Equation of such a line is:

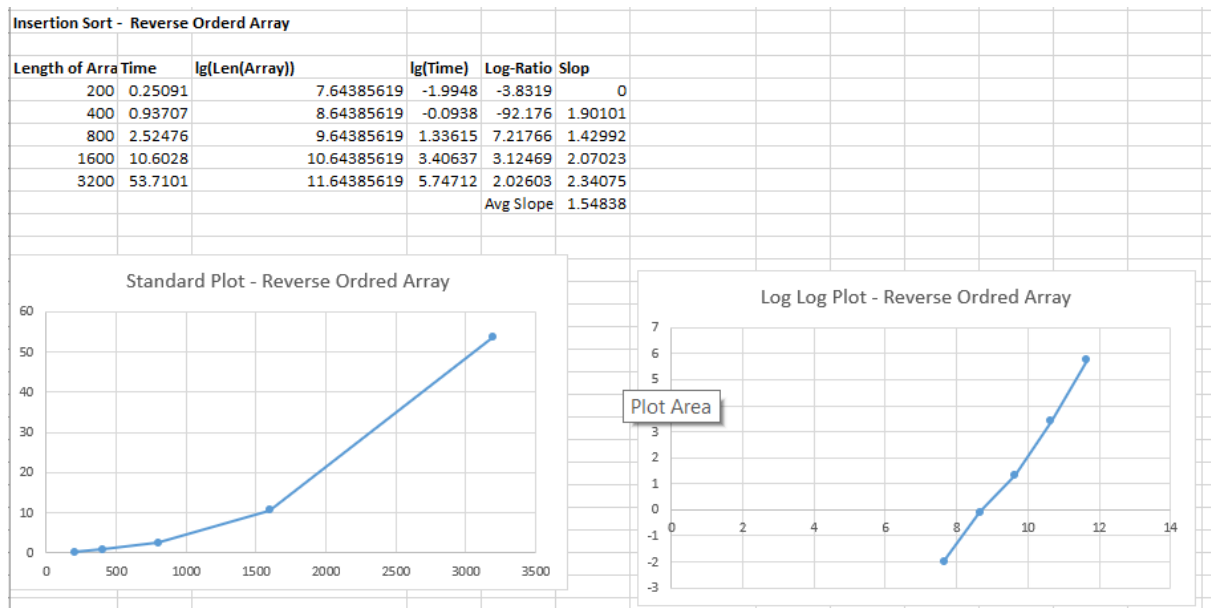
$$T(N) = aN^{1.36}$$



The average slope for Ordered Array is: 0.80

The Equation of such a line is:

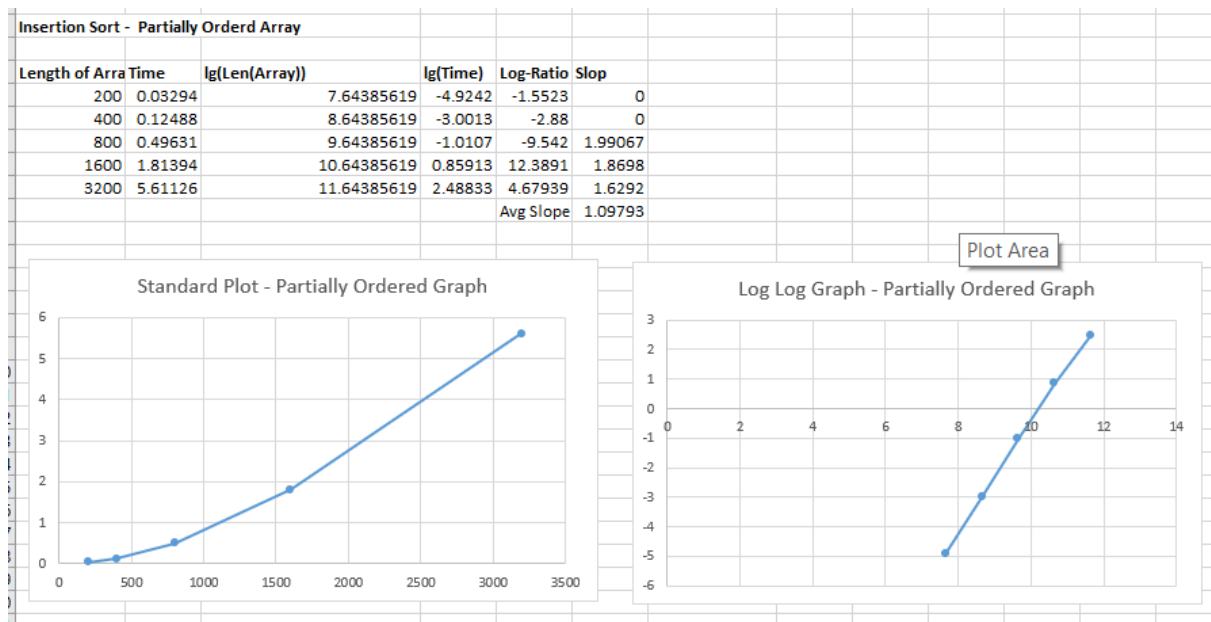
$$T(N) = aN^{0.80}$$



The average slope for Reverse Ordered Array is: 1.54

The Equation of such a line is:

$$I \quad (T(N)) = aN^{1.54}$$



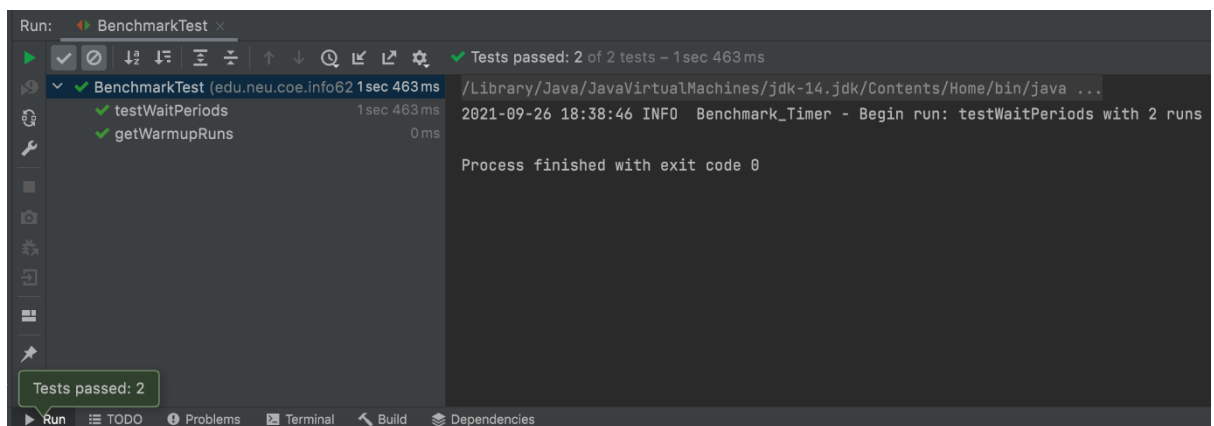
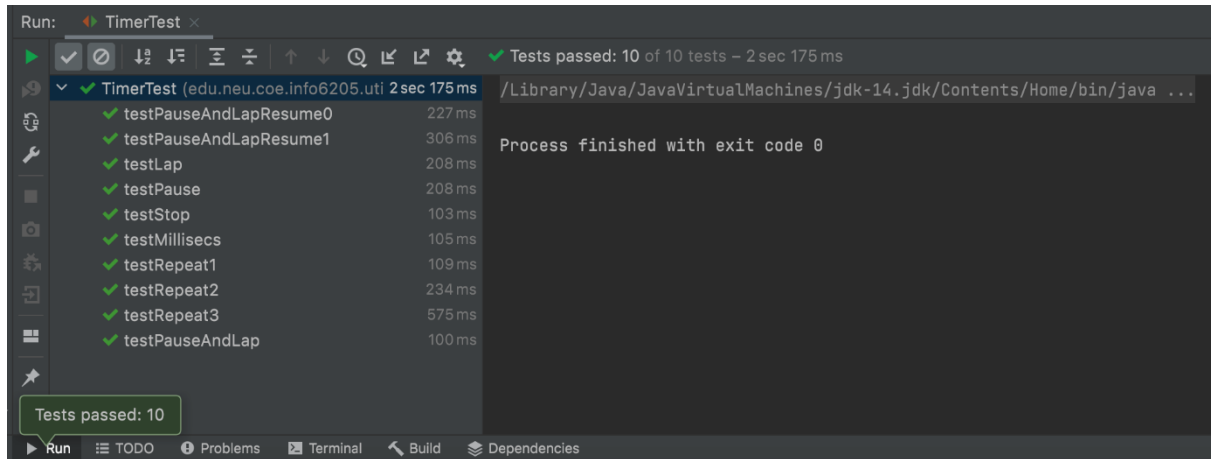
The average slope for Partially Ordered Array is: 1.09

The Equation of such a line is:

$$I \quad (T(N)) = aN^{1.09}$$

Unit tests result:

Part 1 : Timer Test



Part 2: Insertion Sort

