Pranoti Dhabu (002124373)

Program Structures & Algorithms

Fall 2021

Assignment No. 2

Tasks performed

i. Implemented below methods in Timer.java

- repeat(int n, Supplier<T> supplier, Function<T, U> function, UnaryOperator<T> preFunction, Consumer<U> postFunction)
- getClock()
- toMillisecs(long ticks)

ii. Ran unit tests and verified successful execution for

- TimerTest.java
- BenchmarkTest.java

iii. Implemented sort(X[] xs, int from, int to) method in InsertionSort.java

iv. Ran unit tests and verified successful execution for

- InsertionSortTest
- InsertionSortOptTest
- InsertionSortMSDTest

v. Implemented main method to run benchmark insertion sort for various types of ordered arrays

- Randomly ordered elements
- Ordered elements
- Reverse-ordered elements
- Partially-ordered elements

vi. Ran the Java code for various number of steps and analyzed output. Analyzed range, min value, max value and average.

vii. Plotted a graph to understand the and analyze results.

⊙ Relationship Conclusion:

A standard plot graph i.e. N vs T(N) was plotted based on the evidences, below are the observations and conclusions:

1. Worst case scenario corresponds to reverse-ordered and randomly ordered array elements which is equivalent to $O(N^2)$. This is a quadratic growth graph.

2. The average case scenario which is a partially ordered array elements insertion sort graph. The plotted graph is similar to linearithmic graph. Thus the complexity is O(N Log(N)).

3. The best case scenario is when no swaps are required – that is already ordered array. The graph is a constant which is O(N).

- **Evidence to support the conclusion:**

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

   a. **Randomly ordered elements**

```
/Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Contents/Home/bin/java ...
Benchmarking insertion sort
2021-09-26 19:57:38 INFO  Benchmark_Timer - Begin run: Insert sort Random for 1000 Integers with 100 runs
2021-09-26 19:57:38 INFO  BenchmarkDriver - 0.800 ms
2021-09-26 19:57:38 INFO  Benchmark_Timer - Begin run: Insert sort Random for 2000 Integers with 100 runs
2021-09-26 19:57:38 INFO  BenchmarkDriver - 2.980 ms
2021-09-26 19:57:38 INFO  Benchmark_Timer - Begin run: Insert sort Random for 4000 Integers with 100 runs
2021-09-26 19:57:40 INFO  BenchmarkDriver - 12.240 ms
2021-09-26 19:57:40 INFO  Benchmark_Timer - Begin run: Insert sort Random for 8000 Integers with 100 runs
2021-09-26 19:57:45 INFO  BenchmarkDriver - 47.600 ms
2021-09-26 19:57:45 INFO  Benchmark_Timer - Begin run: Insert sort Random for 16000 Integers with 100 runs
2021-09-26 19:58:11 INFO  BenchmarkDriver - 240.340 ms
```

   b. **Ordered elements**

```
2021-09-26 19:58:11 INFO  Benchmark_Timer - Begin run: Insert sort Ordered for 1000 Integers with 100 runs
2021-09-26 19:58:11 INFO  BenchmarkDriver - 0.030 ms
2021-09-26 19:58:11 INFO  Benchmark_Timer - Begin run: Insert sort Ordered for 2000 Integers with 100 runs
2021-09-26 19:58:11 INFO  BenchmarkDriver - 0.020 ms
2021-09-26 19:58:11 INFO  Benchmark_Timer - Begin run: Insert sort Ordered for 4000 Integers with 100 runs
2021-09-26 19:58:11 INFO  BenchmarkDriver - 0.030 ms
2021-09-26 19:58:11 INFO  Benchmark_Timer - Begin run: Insert sort Ordered for 8000 Integers with 100 runs
2021-09-26 19:58:11 INFO  BenchmarkDriver - 0.060 ms
2021-09-26 19:58:11 INFO  Benchmark_Timer - Begin run: Insert sort Ordered for 16000 Integers with 100 runs
2021-09-26 19:58:11 INFO  BenchmarkDriver - 0.100 ms
```

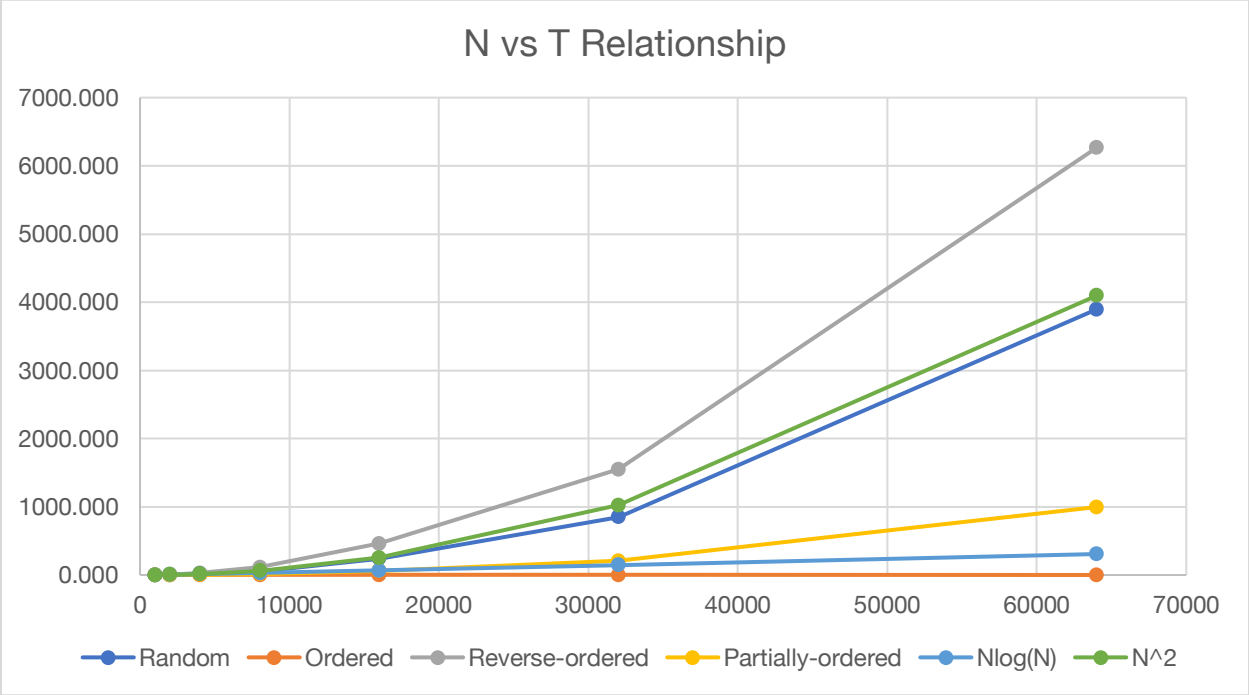
   c. **Reverse ordered elements**

```
2021-09-26 19:58:11 INFO  BenchmarkDriver - 0.100 ms
2021-09-26 19:58:11 INFO  Benchmark_Timer - Begin run: Insert sort Reverse Ordered for 1000 Integers with 100 runs
2021-09-26 19:58:11 INFO  BenchmarkDriver - 1.830 ms
2021-09-26 19:58:11 INFO  Benchmark_Timer - Begin run: Insert sort Reverse Ordered for 2000 Integers with 100 runs
2021-09-26 19:58:12 INFO  BenchmarkDriver - 7.190 ms
2021-09-26 19:58:12 INFO  Benchmark_Timer - Begin run: Insert sort Reverse Ordered for 4000 Integers with 100 runs
2021-09-26 19:58:15 INFO  BenchmarkDriver - 29.070 ms
2021-09-26 19:58:15 INFO  Benchmark_Timer - Begin run: Insert sort Reverse Ordered for 8000 Integers with 100 runs
2021-09-26 19:58:28 INFO  BenchmarkDriver - 115.140 ms
2021-09-26 19:58:28 INFO  Benchmark_Timer - Begin run: Insert sort Reverse Ordered for 16000 Integers with 100 runs
2021-09-26 19:59:19 INFO  BenchmarkDriver - 462.610 ms
```

   d. **Partially ordered elements**

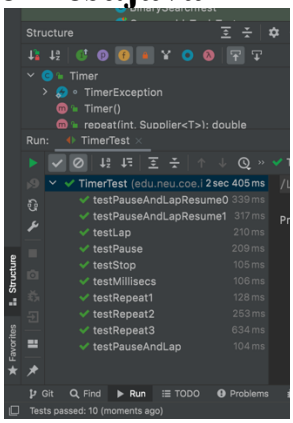
```
2021-09-26 19:59:19 INFO  Benchmark_Timer - Begin run: Insert sort Partially Ordered for 1000 Integers with 100 runs
2021-09-26 19:59:19 INFO  BenchmarkDriver - 0.290 ms
2021-09-26 19:59:19 INFO  Benchmark_Timer - Begin run: Insert sort Partially Ordered for 2000 Integers with 100 runs
2021-09-26 19:59:19 INFO  BenchmarkDriver - 0.990 ms
2021-09-26 19:59:19 INFO  Benchmark_Timer - Begin run: Insert sort Partially Ordered for 4000 Integers with 100 runs
2021-09-26 19:59:19 INFO  BenchmarkDriver - 3.830 ms
2021-09-26 19:59:19 INFO  Benchmark_Timer - Begin run: Insert sort Partially Ordered for 8000 Integers with 100 runs
2021-09-26 19:59:21 INFO  BenchmarkDriver - 15.250 ms
2021-09-26 19:59:21 INFO  Benchmark_Timer - Begin run: Insert sort Partially Ordered for 16000 Integers with 100 runs
2021-09-26 19:59:28 INFO  BenchmarkDriver - 64.100 ms

Process finished with exit code 0
```

2. **Graphical Representation(Observations from experiments should be tabulated and analyzed by plotting graphs(usually in excel) to arrive on the relationship conclusion)**

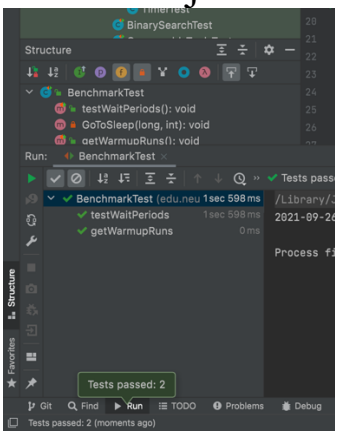


- **Unit tests result:(Snapshot of successful unit test run)**

TimerTest.java



BenchmarkTest.java



InsertionSortTest.java