

# Sai Saketh Kosuri (002199401)

## Program Structures & Algorithms Fall 202

### Assignment No. 02

#### Task:

Implement 3 methods in Timer class and to run the unit tests in TimerTest and BenchmarkTest

Implement insertion sort and verify the implementation by running unit tests in InsertionSortTest.

Implement a main method to run the following benchmarks:

- Random array generation
- Sorted array generation
- Reverse Ordered array generation
- Partially Ordered array generation

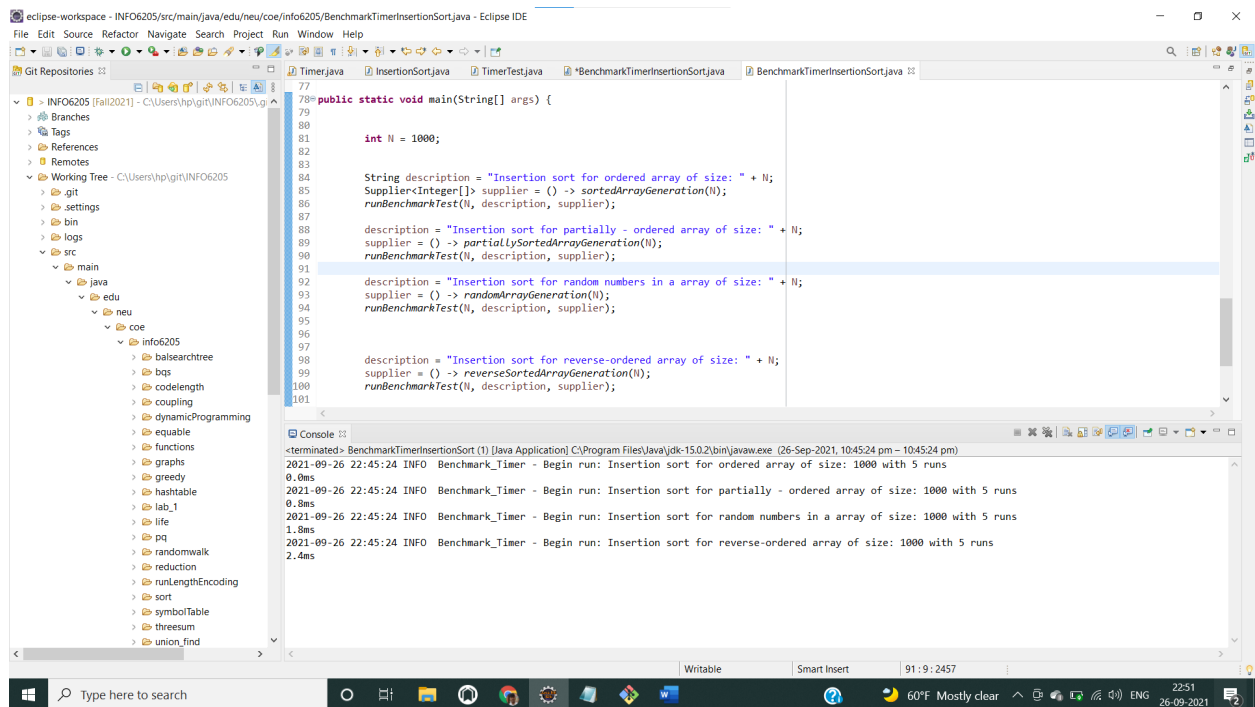
Also use the doubling method for choosing 'n' and test for at least five values of 'n'

#### Relationship Conclusion for Task 3:

From the observations of the graph i.e. relationship between N and T for different ordered arrays, following conclusions are drawn:

1. reverse ordered array elements took maximum amount of time for sorting as it is worst case. Hence the time complexity is  $O(N^2)$  which is a quadratic.
2. Best case scenario corresponds to the ordered array elements with complexity of  $O(N)$ .
3. Partially ordered array elements corresponds to the time complexity is  $O(N \log(N))$  which is a linearithmica.

## Output for Benchmark Timer Insertion sort:



The screenshot shows the Eclipse IDE with the file `BenchmarkTimerInsertionSort.java` open. The code defines a `main` method that benchmarks insertion sort for different input sizes and types. The console output shows the execution results for an input size of 1000.

```
77 public static void main(String[] args) {
78
79     int N = 1000;
80
81
82
83
84     String description = "Insertion sort for ordered array of size: " + N;
85     Supplier<Integer[]> supplier = () -> sortedArrayGeneration(N);
86     runBenchmarkTest(N, description, supplier);
87
88     description = "Insertion sort for partially - ordered array of size: " + N;
89     supplier = () -> partiallySortedArrayGeneration(N);
90     runBenchmarkTest(N, description, supplier);
91
92     description = "Insertion sort for random numbers in a array of size: " + N;
93     supplier = () -> randomArrayGeneration(N);
94     runBenchmarkTest(N, description, supplier);
95
96
97     description = "Insertion sort for reverse-ordered array of size: " + N;
98     supplier = () -> reverseSortedArrayGeneration(N);
99     runBenchmarkTest(N, description, supplier);
100
101 }
```

Console Output:

```
<terminated> BenchmarkTimerInsertionSort (1) [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe (26-Sep-2021, 10:45:24 pm - 10:45:24 pm)
2021-09-26 22:45:24 INFO Benchmark_Timer - Begin run: Insertion sort for ordered array of size: 1000 with 5 runs
0.0ms
2021-09-26 22:45:24 INFO Benchmark_Timer - Begin run: Insertion sort for partially - ordered array of size: 1000 with 5 runs
0.8ms
2021-09-26 22:45:24 INFO Benchmark_Timer - Begin run: Insertion sort for random numbers in a array of size: 1000 with 5 runs
0.8ms
2021-09-26 22:45:24 INFO Benchmark_Timer - Begin run: Insertion sort for reverse-ordered array of size: 1000 with 5 runs
1.6ms
2.4ms
```

### For 1000 Array input size:

Benchmark\_Timer - Begin run: Insertion sort for ordered array of size: 1000 with 5 runs

0.0ms

Benchmark\_Timer - Begin run: Insertion sort for partially - ordered array of size: 1000 with 5 runs

0.8ms

Benchmark\_Timer - Begin run: Insertion sort for random numbers in a array of size: 1000 with 5 runs

0.8ms

Benchmark\_Timer - Begin run: Insertion sort for reverse-ordered array of size: 1000 with 5 runs

1.6ms

### For 5000 array input size:

Benchmark\_Timer - Begin run: Insertion sort for ordered array of size: 5000 with 5 runs

0.2ms

Benchmark\_Timer - Begin run: Insertion sort for partially - ordered array of size: 5000 with 5 runs

5.2ms

Benchmark\_Timer - Begin run: Insertion sort for random numbers in a array of size: 5000 with 5 runs

19.8ms

Benchmark\_Timer - Begin run: Insertion sort for reverse-ordered array of size: 5000 with 5 runs

35.8ms

### For 10000 array input size:

Benchmark\_Timer - Begin run: Insertion sort for ordered array of size: 10000 with 5 runs

0.8ms

Benchmark\_Timer - Begin run: Insertion sort for partially - ordered array of size: 10000 with 5 runs

22.6ms

Benchmark\_Timer - Begin run: Insertion sort for random numbers in a array of size: 10000 with 5 runs

81.6ms

Benchmark\_Timer - Begin run: Insertion sort for reverse-ordered array of size: 10000 with 5 runs

148.6ms

### For 20000 array input size:

2021-09-26 23:49:01 INFO Benchmark\_Timer - Begin run: Insertion sort for ordered array of size: 5000 with 5 runs

0.2ms

2021-09-26 23:49:01 INFO Benchmark\_Timer - Begin run: Insertion sort for partially - ordered array of size: 5000 with 5 runs

5.2ms

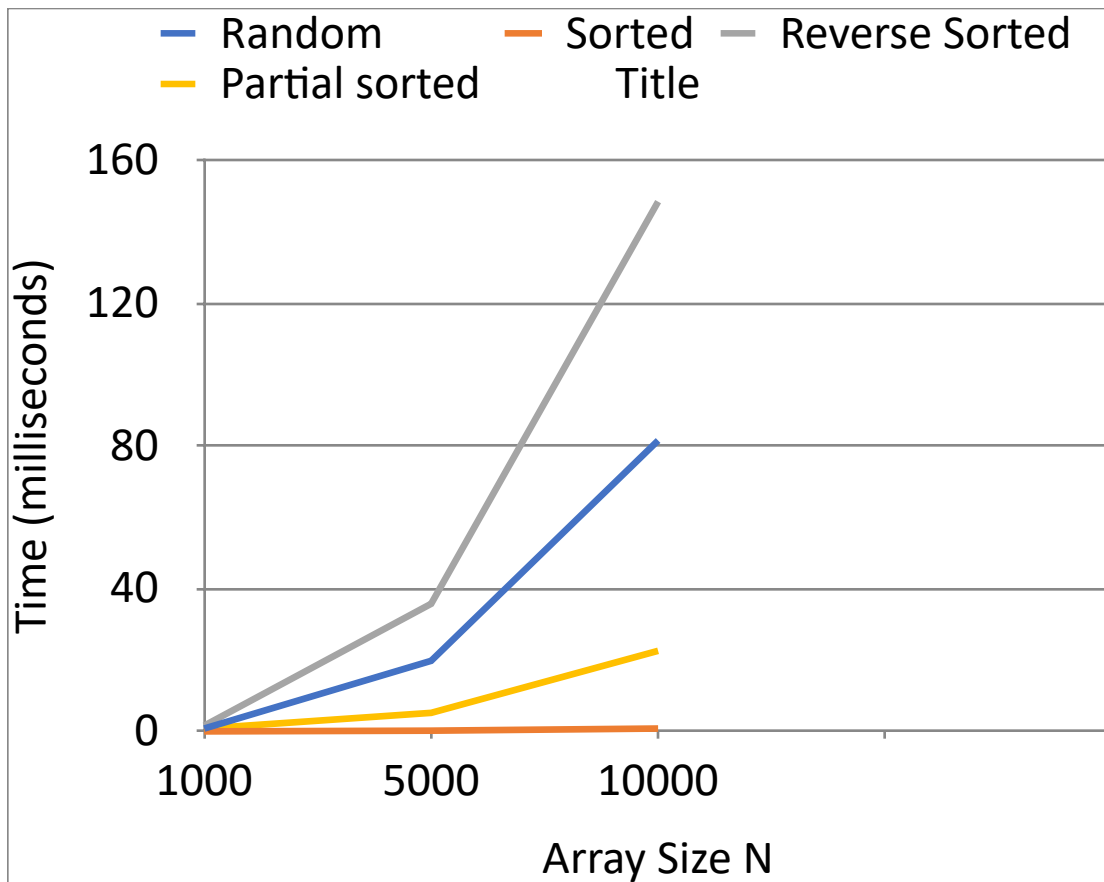
2021-09-26 23:49:01 INFO Benchmark\_Timer - Begin run: Insertion sort for random numbers in a array of size: 5000 with 5 runs

19.8ms

2021-09-26 23:49:01 INFO Benchmark\_Timer - Begin run: Insertion sort for reverse-ordered array of size: 5000 with 5 runs

35.8ms

N	Random	Sorted	Reverse Sorted	Partial Sorted
1000	0.8	0	1.6	0.8
5000	19.8	0.2	35.8	5.2
10000	81.6	0.8	148.6	22.6
20000	19.8	0.9	35.8	5.2



Array Size (N) vs Time in Milliseconds (T)

## Output for Timer Test java:

The screenshot displays the Eclipse IDE with the source code for `TimerTest.java` and the test results in the right-hand pane.

```

1 package edu.neu.coe.info6205.util;
2
3 import org.junit.Before;
4
5 public class TimerTest {
6
7     @Before
8     public void setup() {
9         pre = 0;
10        run = 0;
11        post = 0;
12    }
13
14    @Test
15    public void testStop() {
16        final Timer timer = new Timer();
17        GoToSleep(TENTH, 0);
18        final double time = timer.stop();
19        assertEquals(TENTH_DOUBLE, time, 10);
20        assertEquals(1, run);
21        assertEquals(1, new PrivateMethodTester(timer).invokePrivate("getLaps"));
22    }
23
24    @Test
25    public void testPauseAndLap() {
26        final Timer timer = new Timer();
27        final PrivateMethodTester privateMethodTester = new PrivateMethodTester(timer);
28        GoToSleep(TENTH, 0);
29        timer.pauseAndLap();
30        final Long ticks = (Long) privateMethodTester.invokePrivate("getTicks");
31        assertEquals(TENTH_DOUBLE, ticks / 1e6, 12);
32        assertEquals((Boolean) privateMethodTester.invokePrivate("isRunning"));
33        assertEquals(1, privateMethodTester.invokePrivate("getLaps"));
34    }
35
36    @Test
37    public void testPauseAndLapResume() {
38        final Timer timer = new Timer();
39        final PrivateMethodTester privateMethodTester = new PrivateMethodTester(timer);
40        GoToSleep(TENTH, 0);
41        timer.pauseAndLap();
42        timer.resume();
43        assertEquals((Boolean) privateMethodTester.invokePrivate("isRunning"));
44    }
45
46 }

```

The test results pane shows the following output:

```

Finished after 2.571 seconds
Runs: 10/10 Errors: 0 Failures: 2
Failure Trace
Error: expected <20.0> but was <31.0>
edu.neu.coe.info6205.util.TimerTestRepeat2(TimerTest.java:119)

```

## Output of Insertion Sort test

The screenshot shows the Eclipse IDE with the following components:

- Left Panel (Project Explorer):** Displays the project structure. The 'elementary' package is expanded, showing 'InsertionSortTest.java'.
- Editor:** Displays the source code of 'InsertionSortTest.java'. The code includes imports, a class definition, and a test method 'sort0()' that uses 'Helper' and 'StatPack' to test the insertion sort algorithm.
- Bottom Panel (Console):** Shows the output of the test. It includes debug messages for configuration and the final statistics for the insertion sort test.

```
7 *import edu.neu.coe.info6205.sort.*;
17
18 @SuppressWarnings("ALL")
19 public class InsertionSortTest {
20
21     @Test
22     public void sort0() throws Exception {
23         final List<Integer> list = new ArrayList<>();
24         list.add(1);
25         list.add(2);
26         list.add(3);
27         list.add(4);
28         Integer[] xs = list.toArray(new Integer[0]);
29         final Config config = ConfigTest.setupConfig("true", "0", "1", "", "");
30         Helper<Integer> helper = HelperFactory.create("InsertionSort", list.size(), config);
31         helper.init(list.size());
32         final PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
33         final StatPack statPack = (StatPack) privateMethodTester.invokePrivate("getStatPack");
34         SortWithHelper<Integer> sorter = new InsertionSort<Integer>(helper);
35         sorter.preProcess(xs);
36         Integer[] ys = sorter.sort(xs);
37         assertTrue(helper.sorted(ys));
38         sorter.postProcess(ys);
39         final int compares = (int) statPack.getStatistics(InstrumentedHelper.COMPARES).mean();
40         assertEquals(list.size() - 1, compares);
41     }
42 }
```

```
<terminated> InsertionSortTest [JUnit] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe (26-Sep-2021, 11:34:52 pm ~ 11:34:52 pm)
2021-09-26 23:34:52 DEBUG Config - Config.get(helper, instrument) = true
2021-09-26 23:34:52 DEBUG Config - Config.get(helper, seed) = 0
2021-09-26 23:34:52 DEBUG Config - Config.get(instrumenting, copies) = true
2021-09-26 23:34:52 DEBUG Config - Config.get(instrumenting, swaps) = true
2021-09-26 23:34:52 DEBUG Config - Config.get(instrumenting, compares) = true
2021-09-26 23:34:52 DEBUG Config - Config.get(instrumenting, inversions) = 1
2021-09-26 23:34:52 DEBUG Config - Config.get(instrumenting, fixes) = true
2021-09-26 23:34:52 DEBUG Config - Config.get(instrumenting, hits) = true
2021-09-26 23:34:52 DEBUG Config - Config.get(helper, cutoff) =
Helper for InsertionSort with 4 elements
StatPack {hits: 9,880; copies: 0; inversions: 2,421; swaps: 2,421; fixes: 2,421; compares: 2,519}
StatPack {hits: 19,800; copies: 0; inversions: 4,950; swaps: 4,950; fixes: 4,950; compares: 4,950}
```

Helper for InsertionSort with 4 elements

StatPack {hits: 9,880; copies: 0; inversions: 2,421; swaps: 2,421; fixes: 2,421; compares: 2,519}  
StatPack {hits: 19,800; copies: 0; inversions: 4,950; swaps: 4,950; fixes: 4,950; compares: 4,950}