Santhosh Krishna Maddi (002127881)

Program Structures & Algorithms Fall 2021

Assignment No 2

Task:

- To implement three methods of the Timer class and check the implementation by running the unit tests in BenchmarkTest and TimerTest
- Implement InsertionSort using a *helper.swapStableConditional* method, and check the implementation by running unit tests in InsertionSortTest.
- Implement a driver program to run the following benchmarks: measure the running times of this sort using the following four different initial array ordering situations: i) Random ii) Ordered iii) Reverse Ordered iv) Partially Ordered Also use the doubling method for choosing ‘n’ and test for at least five values of ‘n

Findings:

Part 3 is executed for four different types of initial inputs. The four different initial input types are as follows: Ordered array, partially ordered array, randomly elements array, reverse ordered array.

For each type of array, the experiment is calculated for 5 different sizes of an array using the doubling method as shown below.

N	Ordered	Partial Ordered	Random	Reverse Ordered
1000	0.2	2.2	2.0	4.8
2000	0.2	2.2	6.6	17.4
4000	0.2	9.6	26.0	48.6
8000	0.2	23.6	48.6	108.8
16000	0.2	74.8	275.6	410.0

Output from terminal

For an ordered array with different sizes, the time taken for insertion sort is as follows:

- For N =1000, time in ms: 0.2
- For N =2000, time in ms: 0.2
- For N =4000, time in ms: 0.2
- For N =8000, time in ms: 0.2
- For N =16000, time in ms: 0.2

For a partially ordered array with different sizes, the time taken for insertion sort is as follows:

- For N =1000, time in ms: 2.2
- For N =2000, time in ms: 2.2
- For N =4000, time in ms: 9.6
- For N =8000, time in ms: 23.6
- For N =16000, time in ms: 74.8

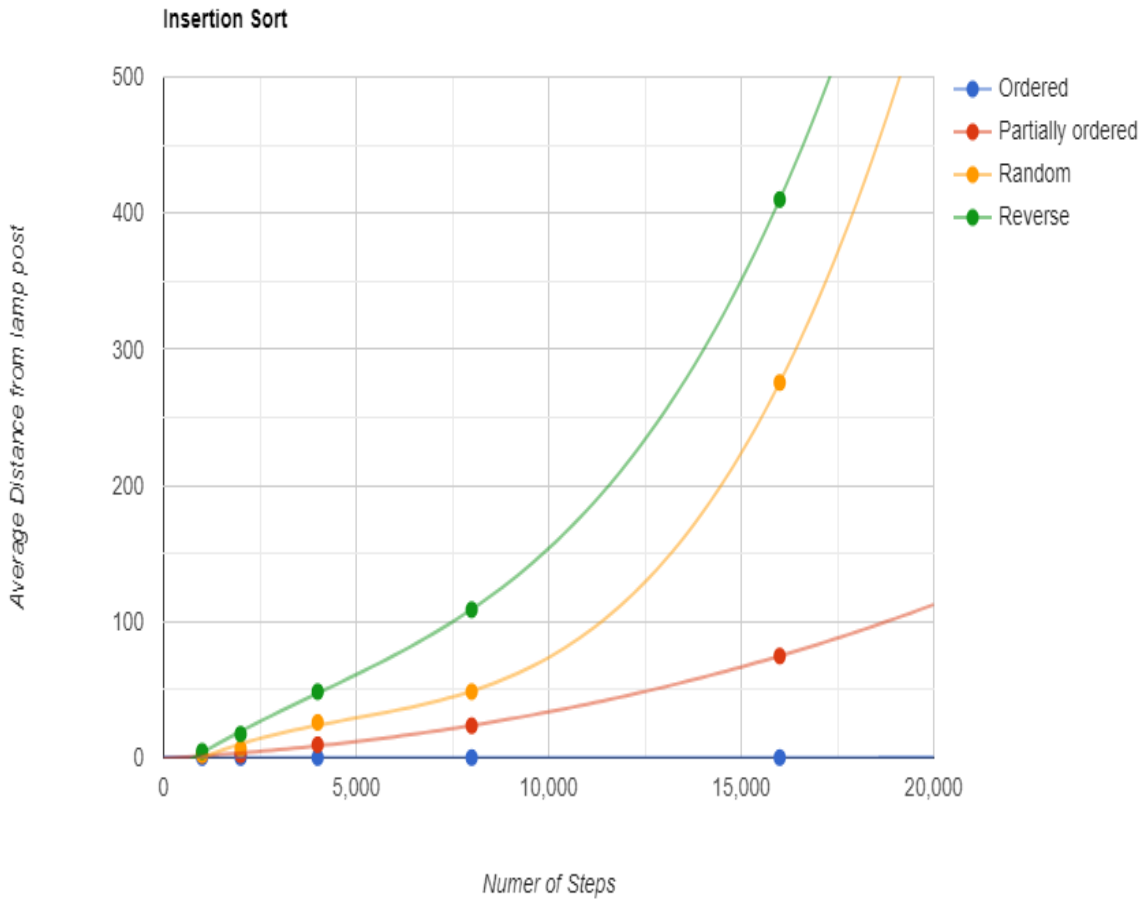
For a random elements array with different sizes, the time taken for insertion sort is as follows:

- For N =1000, time in ms: 2.0
- For N =2000, time in ms: 6.6
- For N =4000, time in ms: 26.0
- For N =8000, time in ms: 48.6
- For N =16000, time in ms: 275.6

For a reverse ordered array with different sizes, the time taken for insertion sort is as follows:

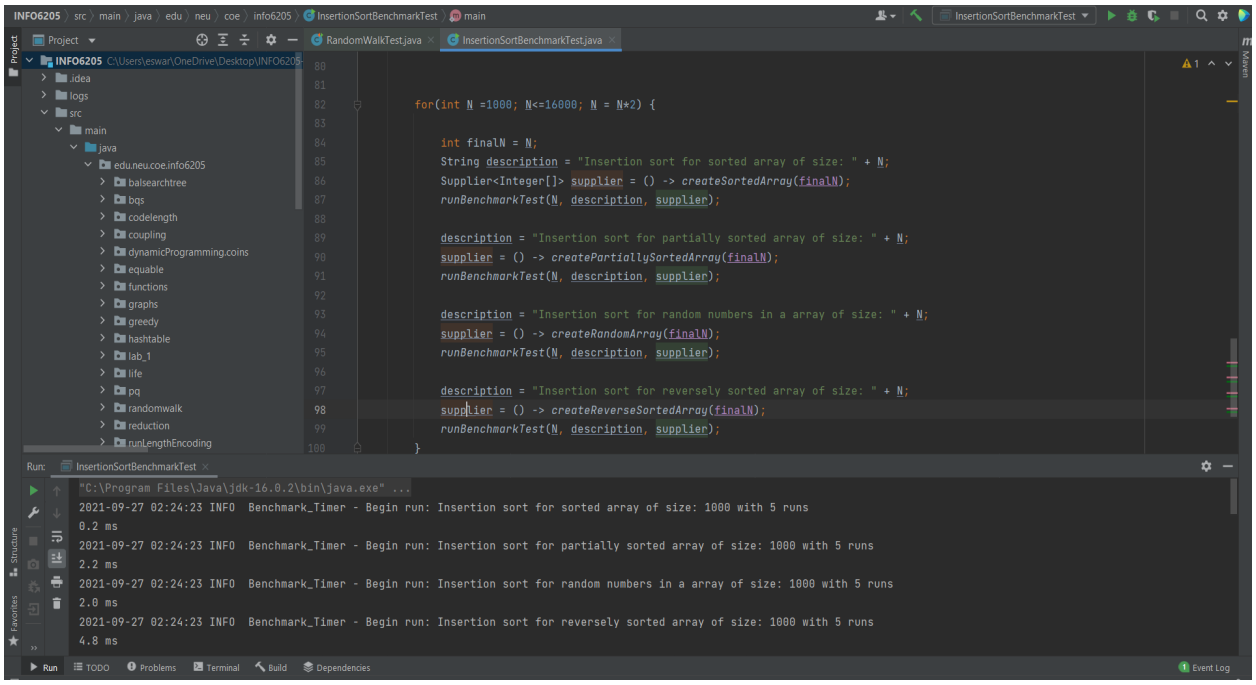
- For N =1000, time in ms: 4.8
- For N =2000, time in ms: 17.4
- For N =4000, time in ms: 48.6
- For N =8000, time in ms: 108.8
- For N =16000, time in ms: 410.0

Graph

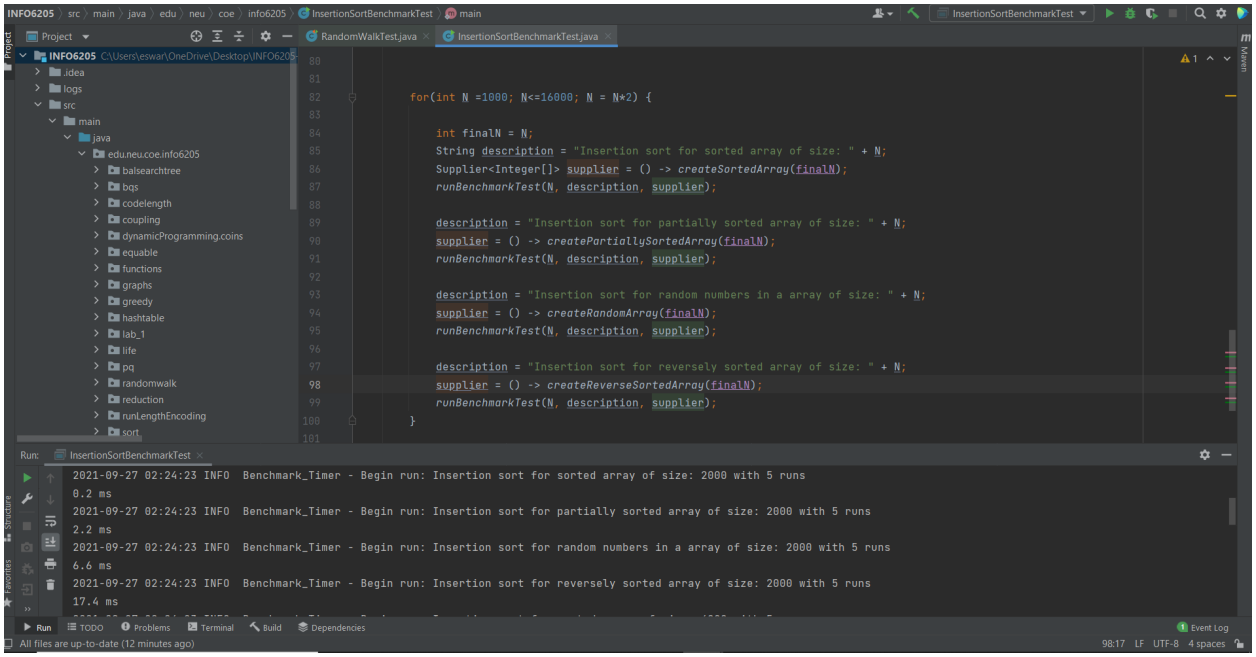

Insertion Sort
Average Distance from lamp post
500
400
300
200
100
0
0
5,000
10,000
15,000
20,000
Numer of Steps
Ordered
Partially ordered
Random
Reverse

Output (Snapshot of Code output in the terminal)

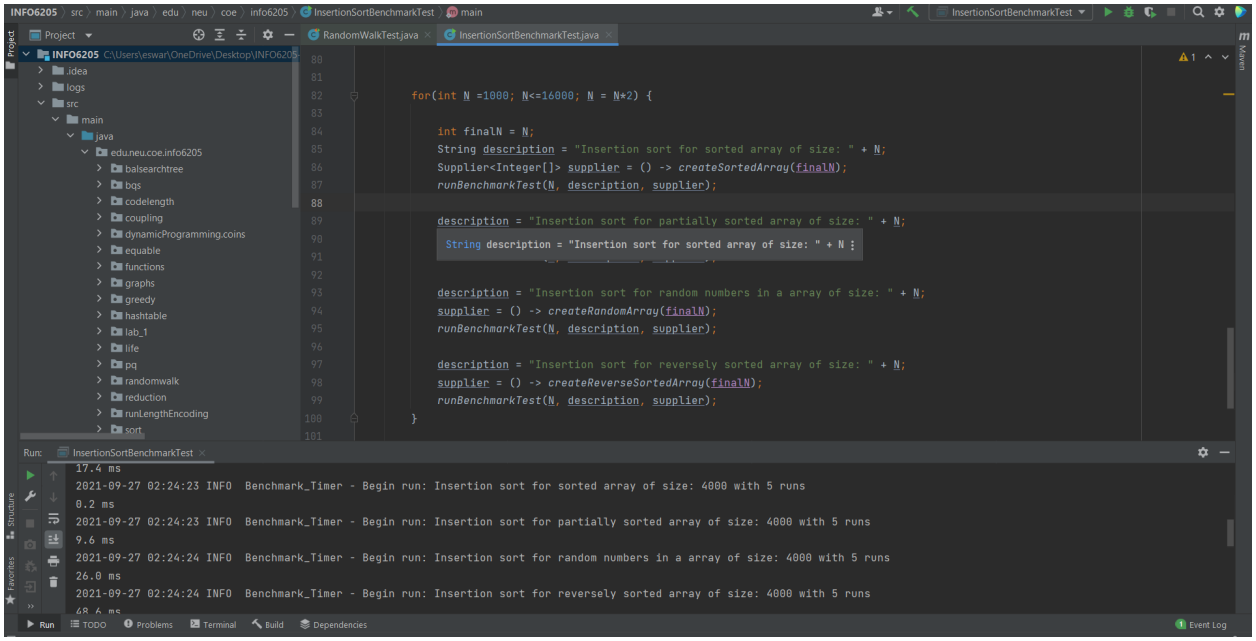
For N = 1000, the output for four types of input arrays :



For N = 2000, the output for four types of input arrays :



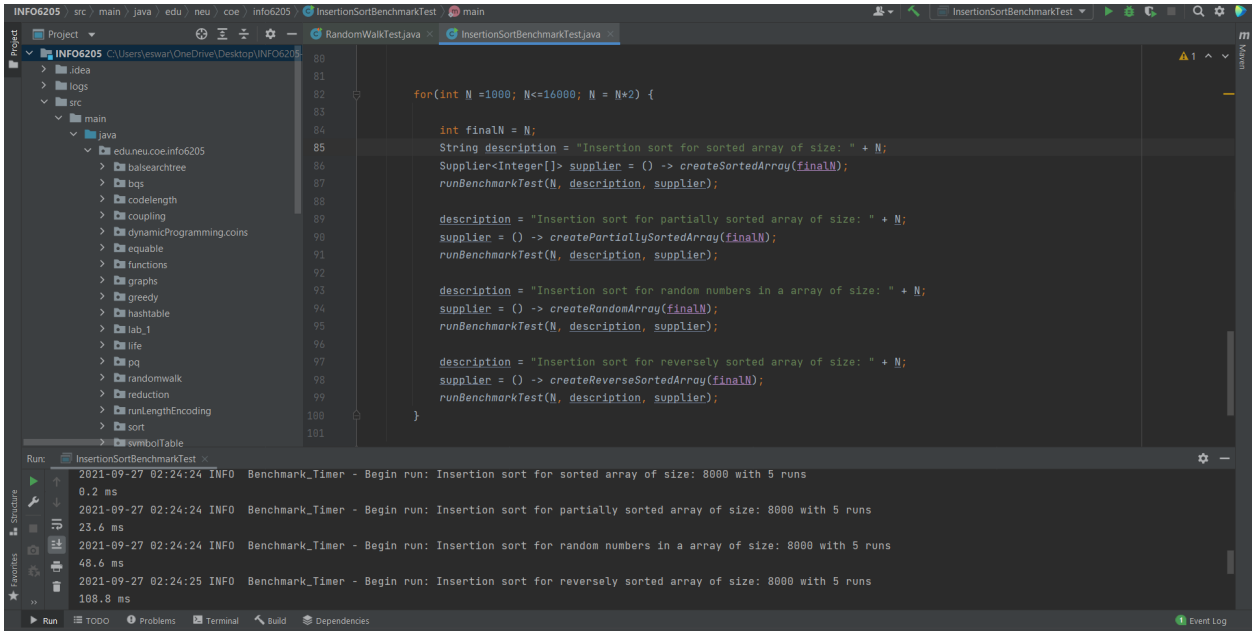
For N = 4000, the output for four types of input arrays :



```
for(int N =1000; N<=16000; N = N*2) {

    int finalN = N;
    String description = "Insertion sort for sorted array of size: " + N;
    Supplier<Integer[]> supplier = () -> createSortedArray(finalN);
    runBenchmarkTest(N, description, supplier);

    description = "Insertion sort for partially sorted array of size: " + N;
    String description = "Insertion sort for sorted array of size: " + N

    description = "Insertion sort for random numbers in a array of size: " + N;
    supplier = () -> createRandomArray(finalN);
    runBenchmarkTest(N, description, supplier);

    description = "Insertion sort for reversely sorted array of size: " + N;
    supplier = () -> createReverseSortedArray(finalN);
    runBenchmarkTest(N, description, supplier);
}
```

```
17.4 ms
2021-09-27 02:24:23 INFO  Benchmark_Timer - Begin run: Insertion sort for sorted array of size: 4000 with 5 runs
0.2 ms
2021-09-27 02:24:23 INFO  Benchmark_Timer - Begin run: Insertion sort for partially sorted array of size: 4000 with 5 runs
9.6 ms
2021-09-27 02:24:24 INFO  Benchmark_Timer - Begin run: Insertion sort for random numbers in a array of size: 4000 with 5 runs
26.0 ms
2021-09-27 02:24:24 INFO  Benchmark_Timer - Begin run: Insertion sort for reversely sorted array of size: 4000 with 5 runs
```

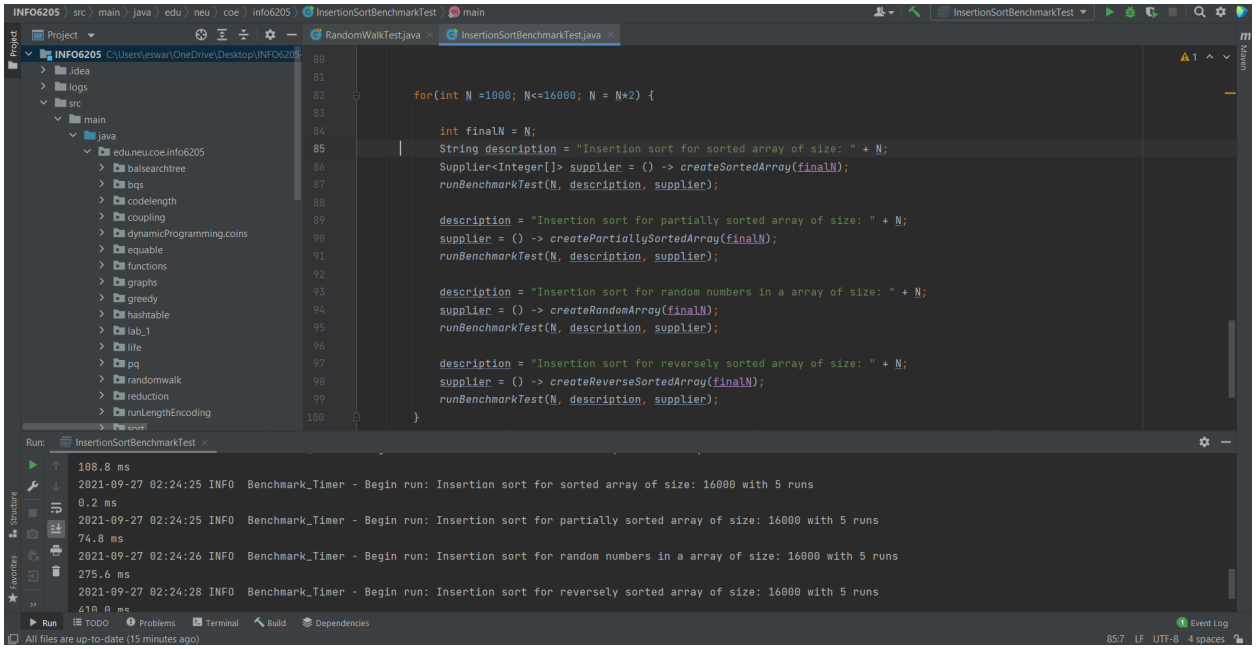
For N =8000, the output for four types of input arrays :



```
for(int N =1000; N<=16000; N = N*2) {

    int finalN = N;
    String description = "Insertion sort for sorted array of size: " + N;
    Supplier<Integer[]> supplier = () -> createSortedArray(finalN);
    runBenchmarkTest(N, description, supplier);

    description = "Insertion sort for partially sorted array of size: " + N;
    supplier = () -> createPartiallySortedArray(finalN);
    runBenchmarkTest(N, description, supplier);

    description = "Insertion sort for random numbers in a array of size: " + N;
    supplier = () -> createRandomArray(finalN);
    runBenchmarkTest(N, description, supplier);

    description = "Insertion sort for reversely sorted array of size: " + N;
    supplier = () -> createReverseSortedArray(finalN);
    runBenchmarkTest(N, description, supplier);
}
```

```
2021-09-27 02:24:24 INFO  Benchmark_Timer - Begin run: Insertion sort for sorted array of size: 8000 with 5 runs
0.2 ms
2021-09-27 02:24:24 INFO  Benchmark_Timer - Begin run: Insertion sort for partially sorted array of size: 8000 with 5 runs
23.6 ms
2021-09-27 02:24:24 INFO  Benchmark_Timer - Begin run: Insertion sort for random numbers in a array of size: 8000 with 5 runs
48.6 ms
2021-09-27 02:24:25 INFO  Benchmark_Timer - Begin run: Insertion sort for reversely sorted array of size: 8000 with 5 runs
108.8 ms
```

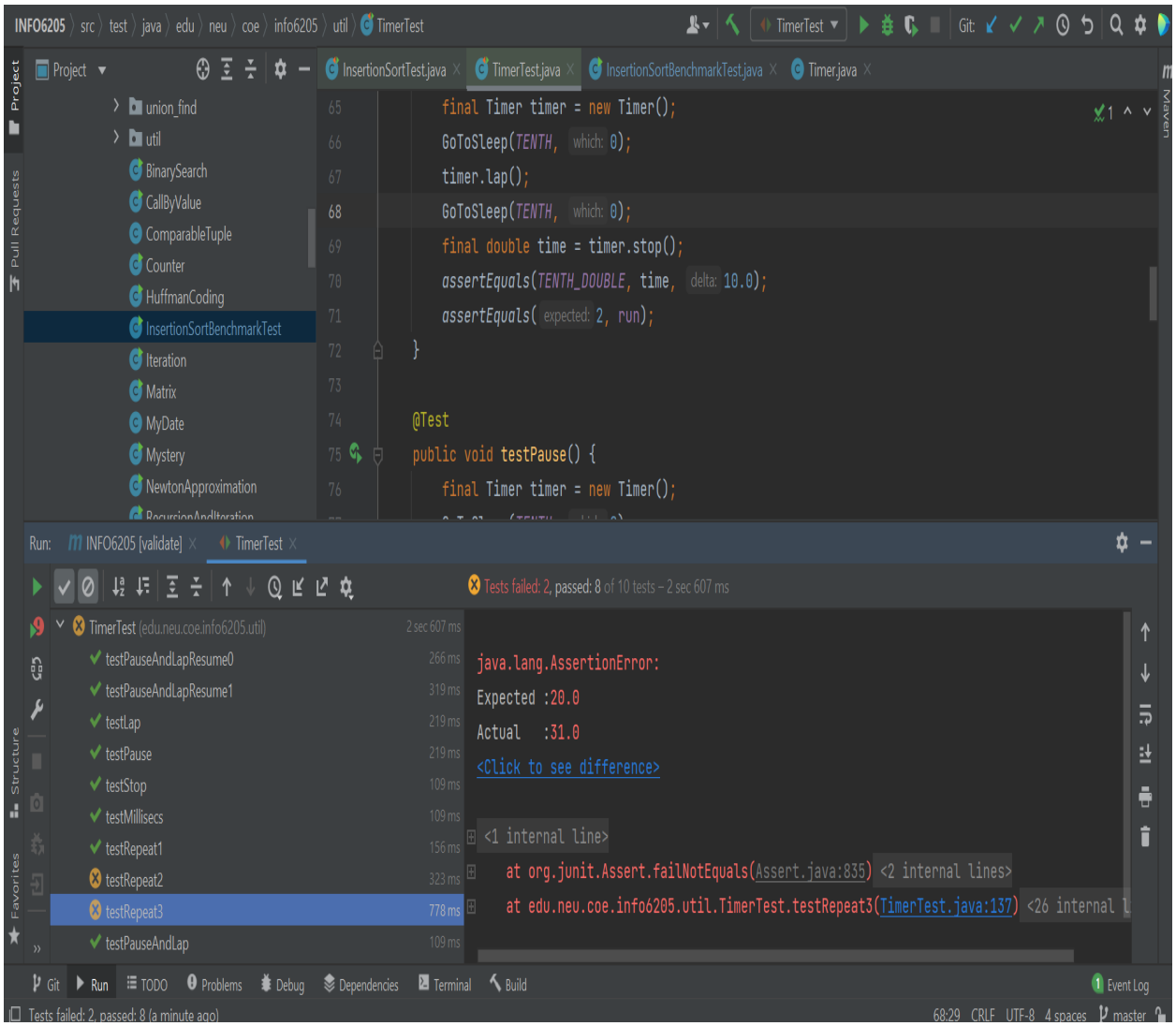
For N = 16000, the output for four types of input arrays :



Test cases:

Part1 Unit test cases :

The screenshot below shows that 8 out of the 10 test cases passed while the other has timing issues where the expected value is very close to the actual value.



Part2 Unit test cases :

The screenshot below shows that 6 out of 6 test cases related to insertion sort are passed