Chaman Betrabet (002784662)

Program Structures & Algorithms

Spring 2023

Assignment No. 3

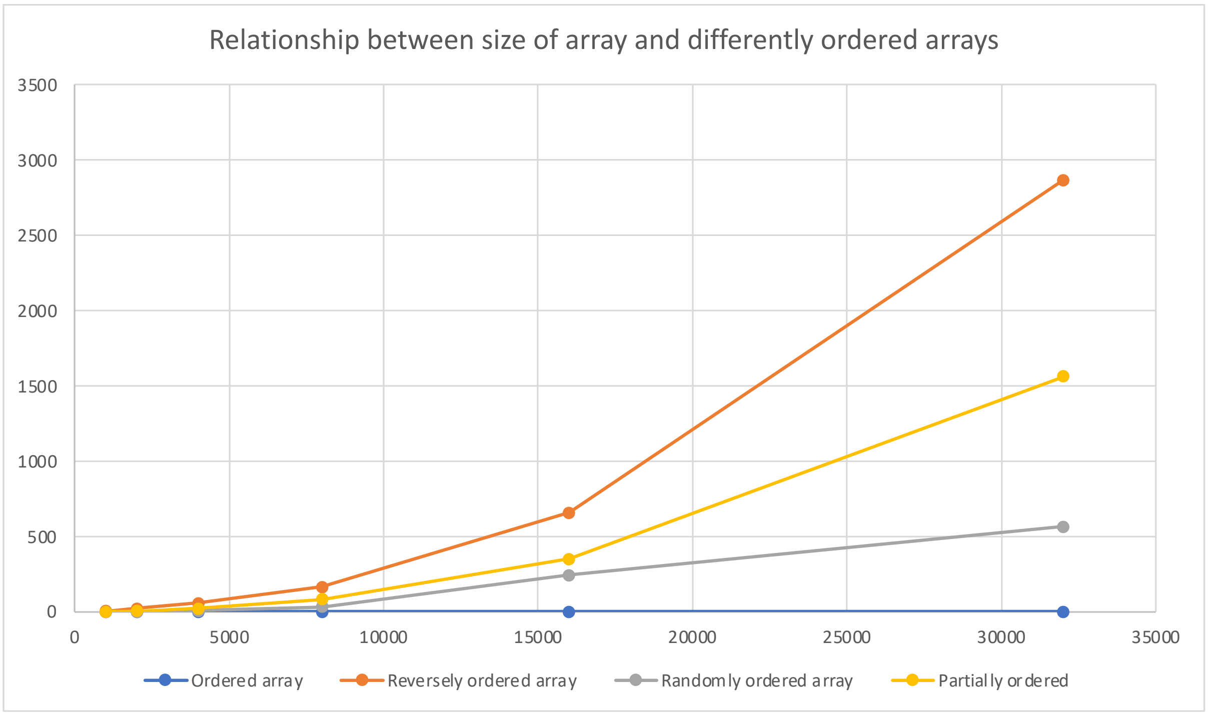
TASK:

- (Part 1) You are to implement three (3) methods (*repeat*, *getClock*, and *toMillisecs*) of a class called *Timer*. Please see the skeleton class that I created in the repository. *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.* If you have the *instrument = true* setting in *test/resources/config.ini*, then you will need to use the *helper* methods for comparing and swapping (so that they properly count the number of swaps/compares). The easiest is to use the *helper.swapStableConditional* method, continuing if it returns true, otherwise breaking the loop. Alternatively, if you are not using instrumenting, then you can write (or copy) your own compare/swap code. Either way, you must run the unit tests in *InsertionSortTest*.
- (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. I suggest that your arrays to be sorted are of type *Integer*. Use the doubling method for choosing *n* and test for at least five values of *n.* Draw any conclusions from your observations regarding the order of growth.

RELATIONSHIP CONCLUSION:

We establish the following relationship between the length of the input(n) and the mean time taken for the InsertionSort_Benchmark class by repeatedly running the main method in the InsertionSort_Benchmark class using the doubling method to test for different values of input arrays (from array length = 100 to array length = 51200) for arrays of 4 different types: ordered arrays, partially ordered arrays, randomly ordered arrays, and reversely ordered arrays.

n (Length of array)	Ordered array	Reversely ordered array	Randomly ordered array	Partially ordered
1000	0.2192958	5.2792457	1.7452872	1.4426415
2000	0.2846455	22.4307918	2.172975	5.3770248
4000	0.6043666	58.5188416	8.3745709	21.2085917
8000	0.3072709	166.247925	32.4364499	84.0837915
16000	0.5667793	658.4606709	243.1770333	352.1181876
32000	0.8670455	2866.027117	566.5442333	1563.113029



Few observations we observe:

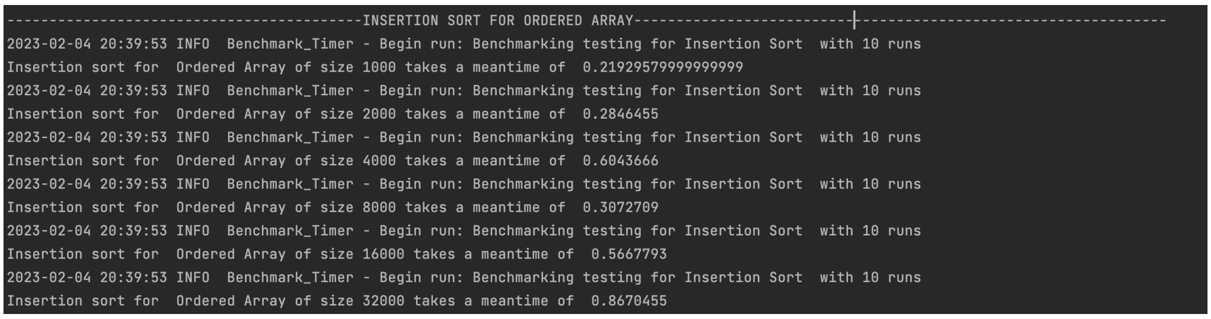
1. Insertion sort compares consecutive elements in the array and swaps them if they are not in order.
2. Insertion sort takes most time for a reversely ordered array.
3. Insertion sort takes least time for a ordered array.
4. Ordered Array< Partially Ordered < Randomly Ordered < Reversely Ordered in order of time it takes to perform insertion sort.

EVIDENCE TO SUPPORT CONCLUSION:

OUTPUT:

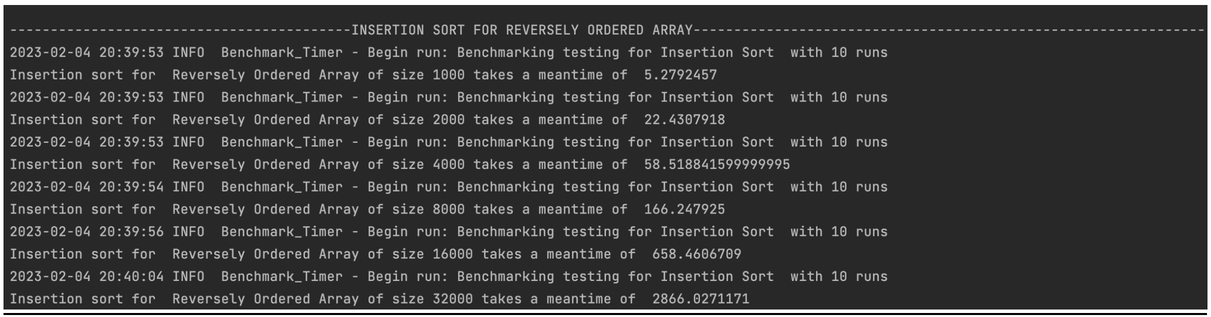
Running the InsertionSort_Benchmark class, we get the following outputs:

INSERTION SORT FOR SORTED ARRAY:



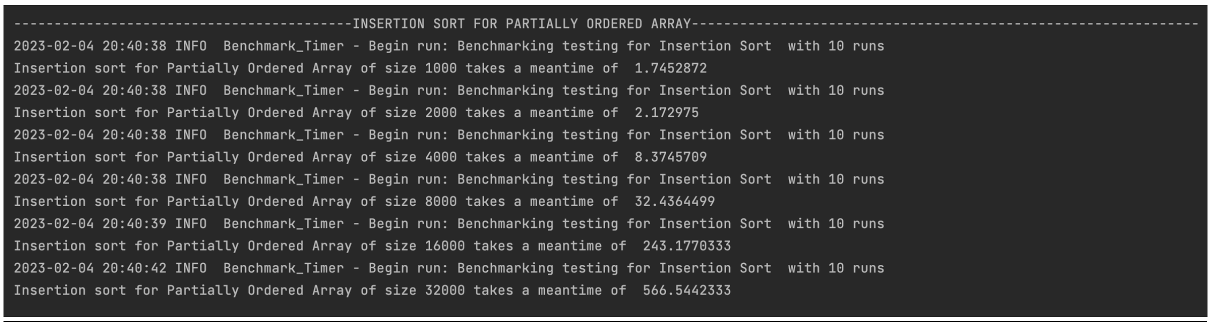
```
-----------------------------------------INSERTION SORT FOR ORDERED ARRAY-------------------------|------------------------------------
2023-02-04 20:39:53 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 1000 takes a meantime of  0.21929579999999999
2023-02-04 20:39:53 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 2000 takes a meantime of  0.2846455
2023-02-04 20:39:53 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 4000 takes a meantime of  0.6043666
2023-02-04 20:39:53 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 8000 takes a meantime of  0.3072709
2023-02-04 20:39:53 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 16000 takes a meantime of  0.5667793
2023-02-04 20:39:53 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 32000 takes a meantime of  0.8670455
```

INSERTION SORT FOR REVERSELY ORDERED ARRAY:



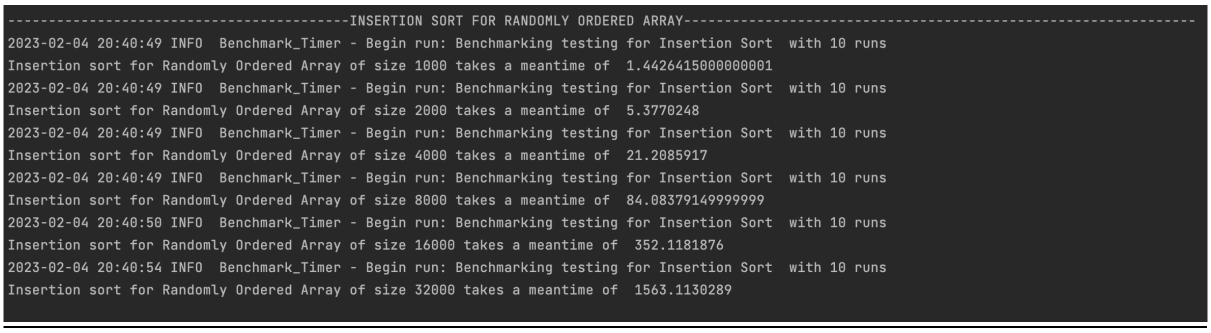
```
-----------------------------------------INSERTION SORT FOR REVERSELY ORDERED ARRAY--------------------------------------------------------------
2023-02-04 20:39:53 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 1000 takes a meantime of  5.2792457
2023-02-04 20:39:53 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 2000 takes a meantime of  22.4307918
2023-02-04 20:39:53 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 4000 takes a meantime of  58.518841599999995
2023-02-04 20:39:54 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 8000 takes a meantime of  166.247925
2023-02-04 20:39:56 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 16000 takes a meantime of  658.4606709
2023-02-04 20:40:04 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 32000 takes a meantime of  2866.0271171
```

INSERTION SORT FOR PARTIALLY ORDERED ARRAY:



```
-----------------------------------------INSERTION SORT FOR PARTIALLY ORDERED ARRAY--------------------------------------------------------------
2023-02-04 20:40:38 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 1000 takes a meantime of  1.7452872
2023-02-04 20:40:38 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 2000 takes a meantime of  2.172975
2023-02-04 20:40:38 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 4000 takes a meantime of  8.3745709
2023-02-04 20:40:38 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 8000 takes a meantime of  32.4364499
2023-02-04 20:40:39 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 16000 takes a meantime of  243.1770333
2023-02-04 20:40:42 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 32000 takes a meantime of  566.5442333
```

INSERTION SORT FOR RANDOMLY ORDERED ARRAY:

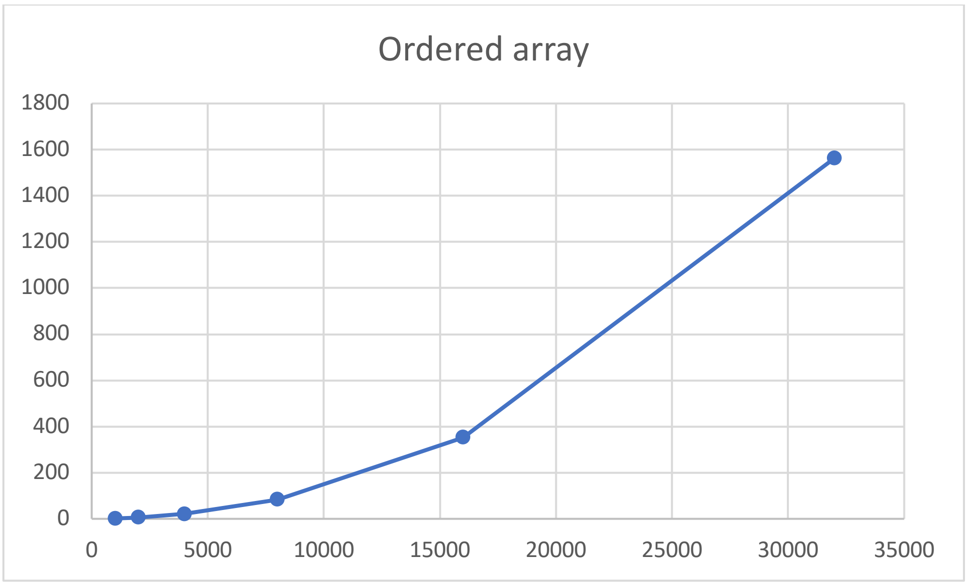


```
----------------------------------------INSERTION SORT FOR RANDOMLY ORDERED ARRAY---------------------------------------------------------------
2023-02-04 20:40:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 1000 takes a meantime of  1.4426415000000001
2023-02-04 20:40:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 2000 takes a meantime of  5.3770248
2023-02-04 20:40:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 4000 takes a meantime of  21.2085917
2023-02-04 20:40:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 8000 takes a meantime of  84.08379149999999
2023-02-04 20:40:50 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 16000 takes a meantime of  352.1181876
2023-02-04 20:40:54 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 32000 takes a meantime of  1563.1130289
```

GRAPHICAL REPRESENTATION:

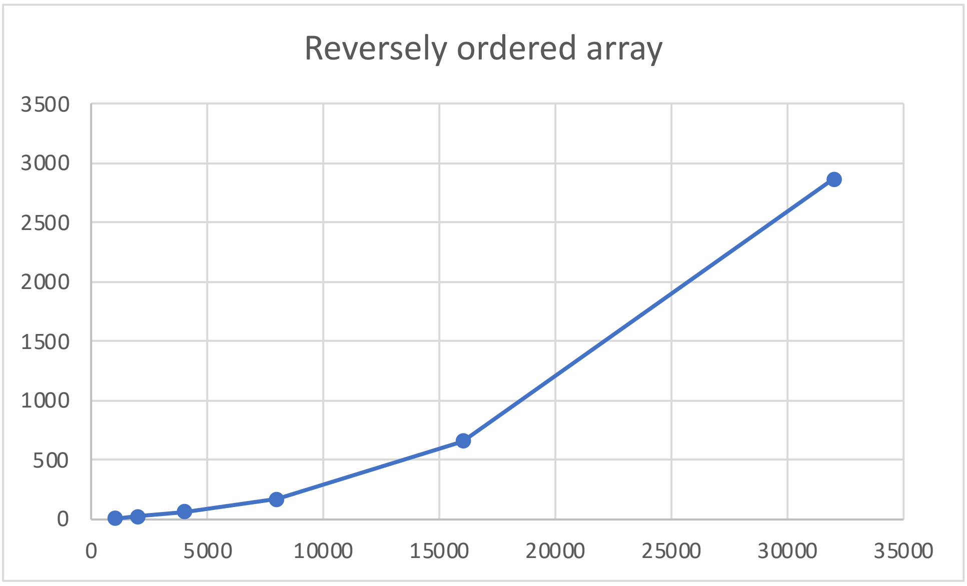
1. Ordered array:

n (Length of array)	Ordered array
1000	0.2192958
2000	0.2846455
4000	0.6043666
8000	0.3072709
16000	0.5667793
32000	0.8670455



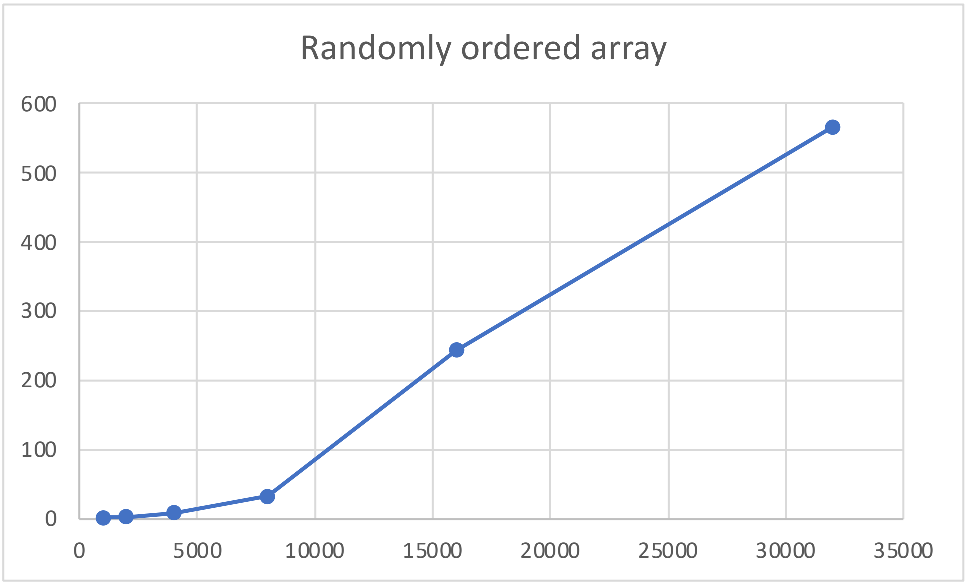
2. Reversely ordered array:

n (Length of array)	Reversely ordered array
1000	5.2792457
2000	22.4307918
4000	58.5188416
8000	166.247925
16000	658.4606709
32000	2866.027117



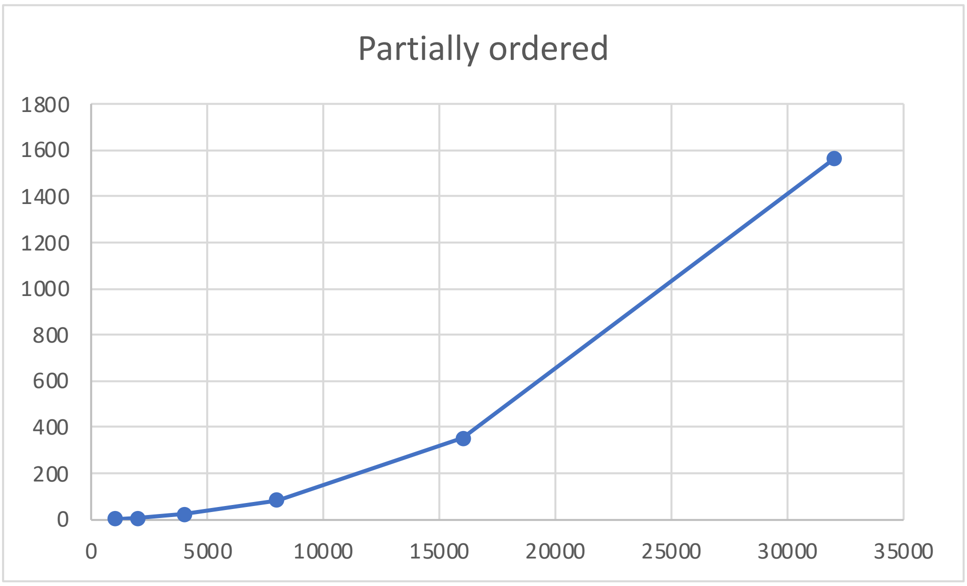
3. Randomly ordered array:

n (Length of array)	Randomly ordered array
1000	1.7452872
2000	2.172975
4000	8.3745709
8000	32.4364499
16000	243.1770333
32000	566.5442333



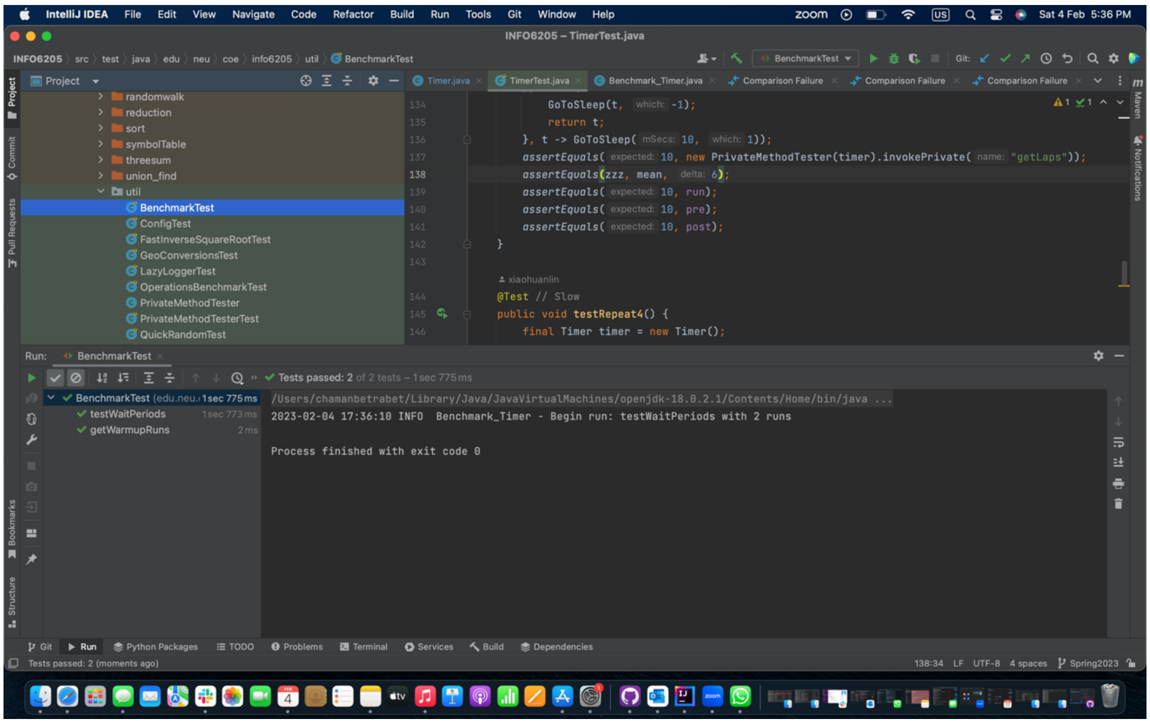
4. Partially ordered array:

n (Length of array)	Partially ordered
1000	1.4426415
2000	5.3770248
4000	21.2085917
8000	84.0837915
16000	352.1181876
32000	1563.113029



UNIT TEST RESULTS:

Benchmark tests:



INFO6205 – TimerTest.java

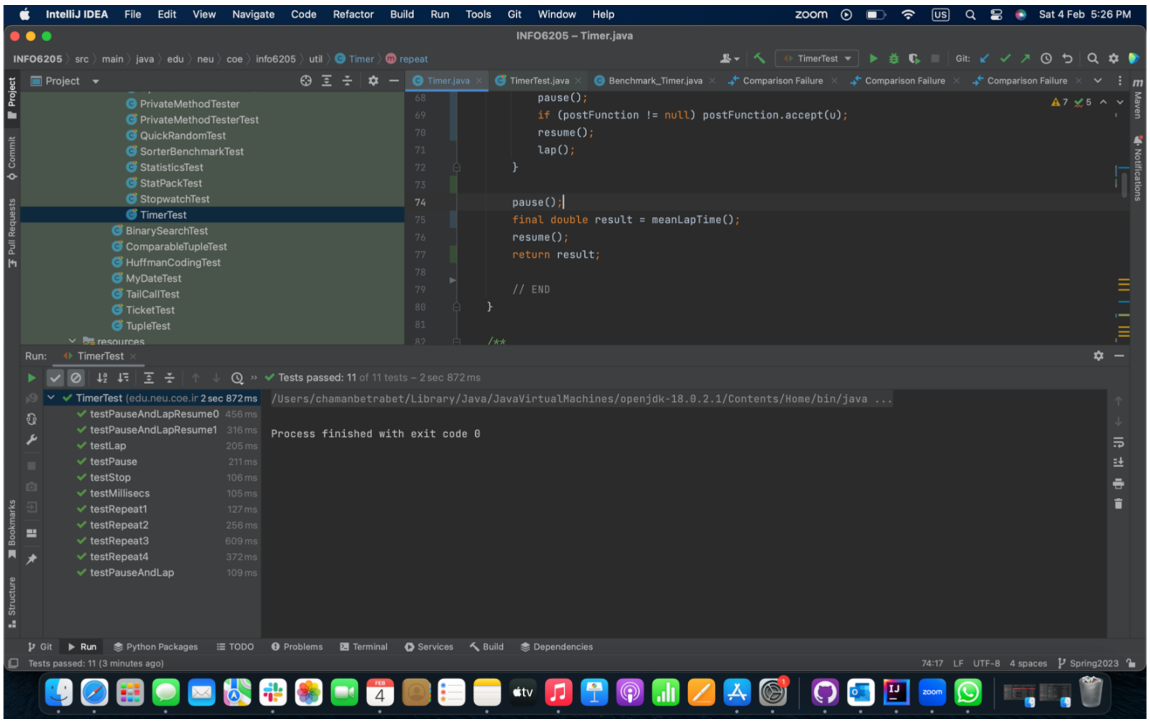
```java
            GoToSleep(t, which: -1);
            return t;
        }, t -> GoToSleep( mSecs: 10, which: 1));
        assertEquals( expected: 10, new PrivateMethodTester(timer).invokePrivate( name: "getLaps"));
        assertEquals(zzz, mean, delta: 6);
        assertEquals( expected: 10, run);
        assertEquals( expected: 10, pre);
        assertEquals( expected: 10, post);
    }

    @Test // Slow
    public void testRepeat4() {
        final Timer timer = new Timer();
```

Run: BenchmarkTest

```
Tests passed: 2 of 2 tests – 1 sec 775 ms

BenchmarkTest (edu.neu.  1 sec 775 ms
  testWaitPeriods        1 sec 773 ms
  getWarmupRuns          2 ms

/Users/chamanbetrabet/Library/Java/JavaVirtualMachines/openjdk-18.0.2.1/Contents/Home/bin/java ...
2023-02-04 17:36:10 INFO  Benchmark_Timer - Begin run: testWaitPeriods with 2 runs

Process finished with exit code 0
```

Tests passed: 2 (moments ago)

Timer tests:



INFO6205 – Timer.java

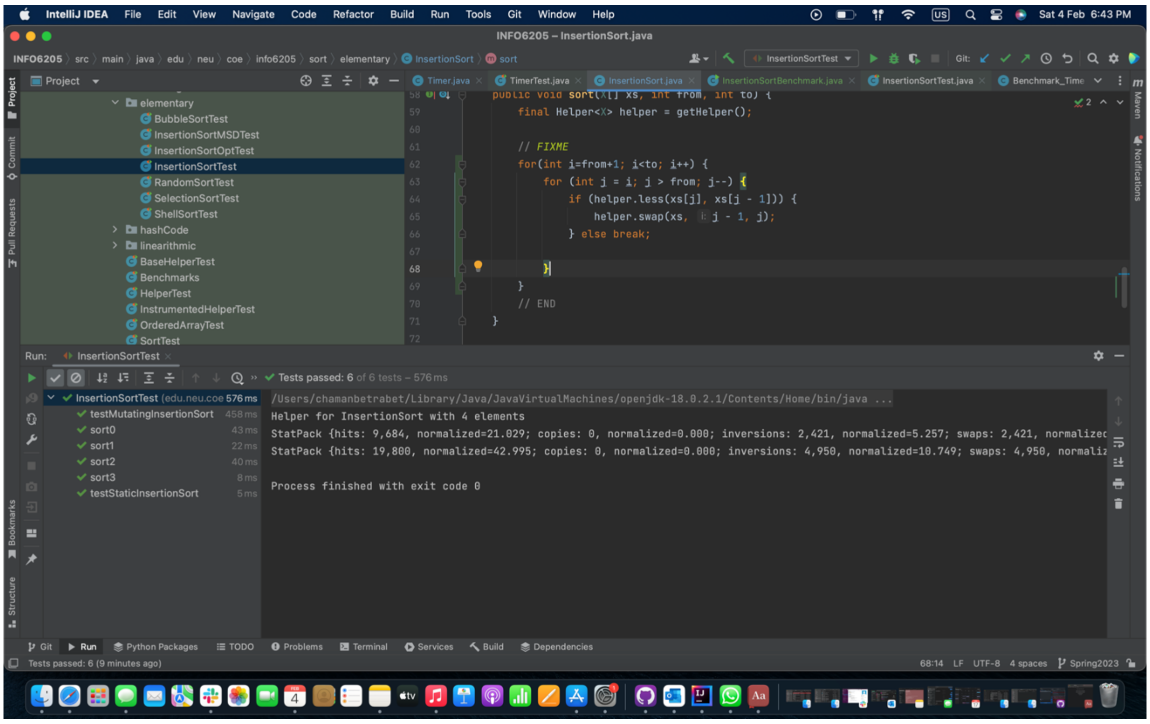
```java
            pause();
            if (postFunction != null) postFunction.accept(u);
            resume();
            lap();
        }

        pause();
        final double result = meanLapTime();
        resume();
        return result;

        // END
    }

    /**
```

Run: TimerTest

```
Tests passed: 11 of 11 tests – 2 sec 872 ms

TimerTest (edu.neu.coe.ir  2 sec 872 ms
  testPauseAndLapResume0   456 ms
  testPauseAndLapResume1   316 ms
  testLap                  205 ms
  testPause                211 ms
  testStop                 106 ms
  testMillisecs            105 ms
  testRepeat1              127 ms
  testRepeat2              256 ms
  testRepeat3              609 ms
  testRepeat4              372 ms
  testPauseAndLap          109 ms

/Users/chamanbetrabet/Library/Java/JavaVirtualMachines/openjdk-18.0.2.1/Contents/Home/bin/java ...

Process finished with exit code 0
```

Tests passed: 11 (3 minutes ago)

Insertion sort tests:



```java
    public void sort(X[] xs, int from, int to) {
        final Helper<X> helper = getHelper();

        // FIXME
        for(int i=from+1; i<to; i++) {
            for (int j = i; j > from; j--) {
                if (helper.less(xs[j], xs[j - 1])) {
                    helper.swap(xs, j - 1, j);
                } else break;

            }
        }
        // END
    }
```

Tests passed: 6 of 6 tests – 576 ms

InsertionSortTest (edu.neu.coe 576 ms
testMutatingInsertionSort 458 ms
sort0 43 ms
sort1 22 ms
sort2 40 ms
sort3 8 ms
testStaticInsertionSort 5 ms

```
/Users/chamanbetrabet/Library/Java/JavaVirtualMachines/openjdk-18.0.2.1/Contents/Home/bin/java ...
Helper for InsertionSort with 4 elements
StatPack {hits: 9,684, normalized=21.029; copies: 0, normalized=0.000; inversions: 2,421, normalized=5.257; swaps: 2,421, normalized
StatPack {hits: 19,800, normalized=42.995; copies: 0, normalized=0.000; inversions: 4,950, normalized=10.749; swaps: 4,950, normaliz

Process finished with exit code 0
```

Tests passed: 6 (9 minutes ago)