Apoorva Kethamaranahalli (001090928)

Program Structures & Algorithms

Fall 2021

Assignment No. 2

Task:

Task list:

- (Part 1) You are to implement three methods 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. The APIs of these class are as follows:

```
public <T, U> double repeat(int n, Supplier<T> supplier, Function<T, U> function, UnaryOperator<T> preFunction, Cons
umer<U> postFunction) {
// TO BE IMPLEMENTED
}
```

```
private static long getClock() {
    // TO BE IMPLEMENTED
}
```

```
private static double toMillisecs(long ticks) {
    // TO BE IMPLEMENTED
}
```

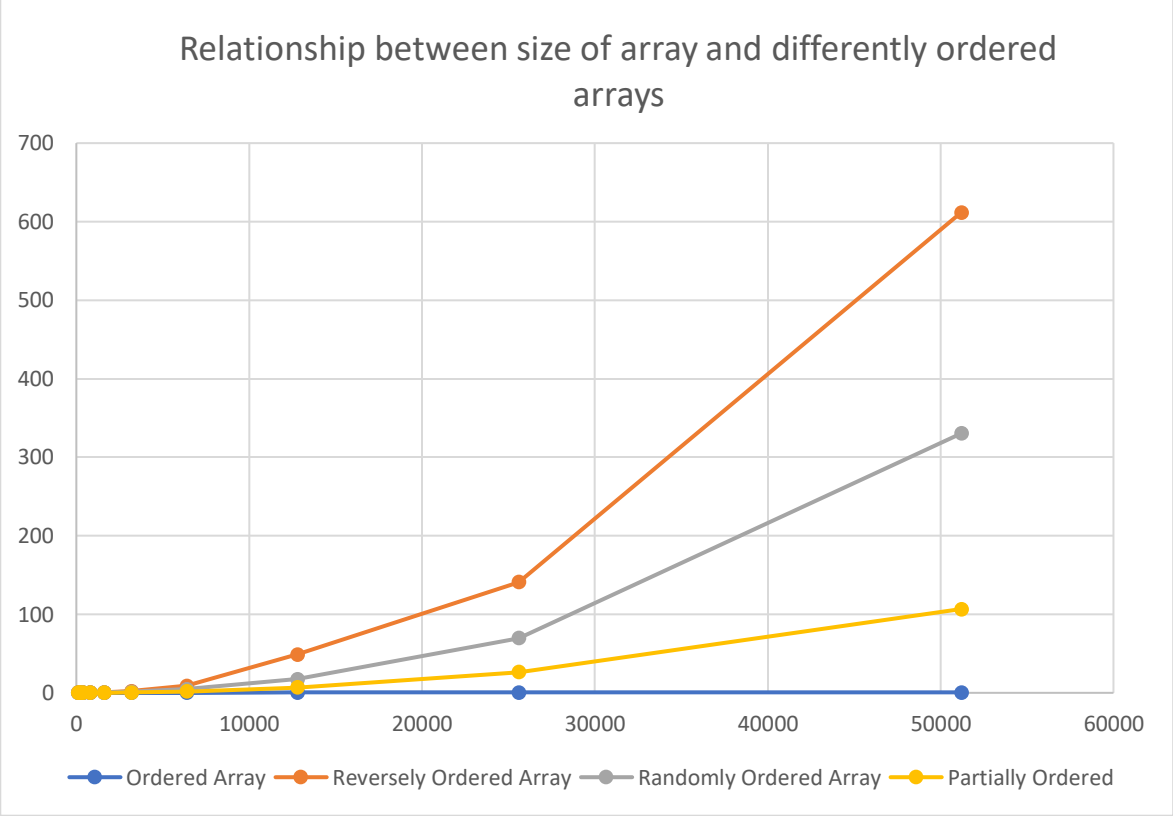
- (Part 2) Implement InsertionSort (in the InsertionSort class) by simply looking up the insertion code used by Arrays.sort. If you have the instrument = true setting in test/resources/config.ini, then you will need to use the helper methods for comparing and swapping (so that they properly count the 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:**

  On running the main method in the Benchmark_output class multiple times, by 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, running them for 10 runs each, I established the following conclusion between length of the input(n) and the mean-time taken for the Insertion Sort after plotting the values in an excel sheet

n(Length of Array)	Ordered Array	Reversely Ordered Array	Randomly Ordered Array	Partially Ordered
100	0.0358125	0.0824861	0.0032788	3.13E-02
200	0.0105593	0.0595656	0.0080354	0.0339712
400	0.0348994	0.1915932	0.0231562	0.0350932
800	0.0288127	0.2279096	0.0824395	0.1168869
1600	0.0585436	0.5433482	0.2959718	0.1648076
3200	0.1139909	2.1686604	1.1775513	0.4651877
6400	0.1457443	8.7474809	4.5498351	1.6947624
12800	0.3627047	48.7065737	17.4374589	6.7495448
25600	0.4467751	141.0240318	69.6801161	26.3680016
51200	0.6404642	611.8534447	330.3842517	106.7523721



- Insertion sort compares consecutive elements in the array, swapping them if they aren't in order
- As seen from the graph above, Insertion sort takes the most time if the array is reversely sorted, because it needs to perform swapping for every consecutive element in the array, which can be the worst possible case for an insertion sort algorithm
- Also, it takes the least amount of time when the array is already sorted
- We can provide the following order of increase in time for an insertion sort for the different types of array input provided as

  Ordered Array< Partially Ordered < Randomly Ordered < Reversely Ordered

- **Evidence to support Conclusion:**

1.Output

In order to test different types of array(ordered, partially ordered, randomly ordered, reverse ordered) of varying lengths (n) starting from 100 going up to 51200(doubling each time), I have created a Benchmark_Output class with a main function to handle the different inputs.

OUTPUT:

INSERTION SORT FOR ORDERED ARRAY:

```
----------------------------------------INSERTION SORT FOR ORDERED ARRAY----------------------------------------
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 100 takes a meantime of  0.035812500000000004
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 200 takes a meantime of  0.0105593
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 400 takes a meantime of  0.034899400000000004
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 800 takes a meantime of  0.028812700000000004
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 1600 takes a meantime of  0.058543599999999994
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 3200 takes a meantime of  0.1139909
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 6400 takes a meantime of  0.1457443
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 12800 takes a meantime of  0.3627047
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 25600 takes a meantime of  0.4467751
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Ordered Array of size 51200 takes a meantime of  0.6404642
```

INSERTION SORT FOR REVERSELY ORDERED ARRAY

```
----------------------------------------INSERTION SORT FOR REVERSELY ORDERED ARRAY----------------------------------------
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 100 takes a meantime of  0.08248609999999999
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 200 takes a meantime of  0.059565599999999996
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 400 takes a meantime of  0.1915932
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 800 takes a meantime of  0.2279096
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 1600 takes a meantime of  0.5433482
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 3200 takes a meantime of  2.1686604
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 6400 takes a meantime of  8.7474809
2021-09-26 22:48:32 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 12800 takes a meantime of  48.7065737
2021-09-26 22:48:33 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 25600 takes a meantime of  141.0240318
2021-09-26 22:48:36 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for  Reversely Ordered Array of size 51200 takes a meantime of  611.8534447
```

INSERTION SORT FOR PARTIALLY ORDERED ARRAY

```
----------------------------------------INSERTION SORT FOR PARTIALLY ORDERED ARRAY----------------------------------------
2021-09-26 22:48:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 100 takes a meantime of  0.031306099999999996
2021-09-26 22:48:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 200 takes a meantime of  0.0339712
2021-09-26 22:48:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 400 takes a meantime of  0.035093200000000005
2021-09-26 22:48:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 800 takes a meantime of  0.11688689999999999
2021-09-26 22:48:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 1600 takes a meantime of  0.1648076
2021-09-26 22:48:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 3200 takes a meantime of  0.4651877
2021-09-26 22:48:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 6400 takes a meantime of  1.6947624000000001
2021-09-26 22:48:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 12800 takes a meantime of  6.7495448
2021-09-26 22:48:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 25600 takes a meantime of  26.368001600000003
2021-09-26 22:48:49 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Partially Ordered Array of size 51200 takes a meantime of  106.7523721
```

INSERTION SORT FOR RANDOMLY ORDERED ARRAY

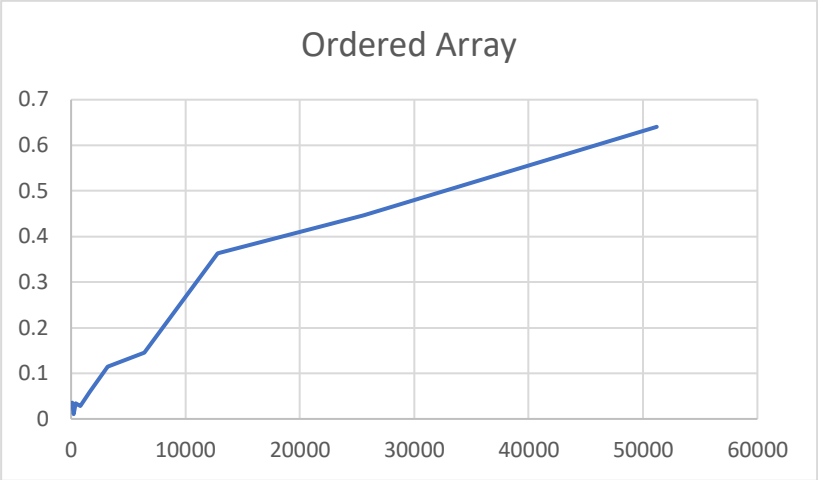
```
----------------------------------------INSERTION SORT FOR RANDOMLY ORDERED ARRAY----------------------------
2021-09-26 22:48:51 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 100 takes a meantime of  0.0032787999999999997
2021-09-26 22:48:51 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 200 takes a meantime of  0.0080354
2021-09-26 22:48:51 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 400 takes a meantime of  0.0231562
2021-09-26 22:48:51 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 800 takes a meantime of  0.0824395
2021-09-26 22:48:51 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 1600 takes a meantime of  0.2959718
2021-09-26 22:48:51 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 3200 takes a meantime of  1.1775513
2021-09-26 22:48:52 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 6400 takes a meantime of  4.5498351
2021-09-26 22:48:52 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 12800 takes a meantime of  17.4374589
2021-09-26 22:48:52 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 25600 takes a meantime of  69.68011609999999
2021-09-26 22:48:53 INFO  Benchmark_Timer - Begin run: Benchmarking testing for Insertion Sort  with 10 runs
Insertion sort for Randomly Ordered Array of size 51200 takes a meantime of  330.3842517

Process finished with exit code 0
```

2. Graphical Representation:

The graphical representations shown below are for mean times for each type of arrays (ordered, partially ordered, reversely ordered, randomly ordered)

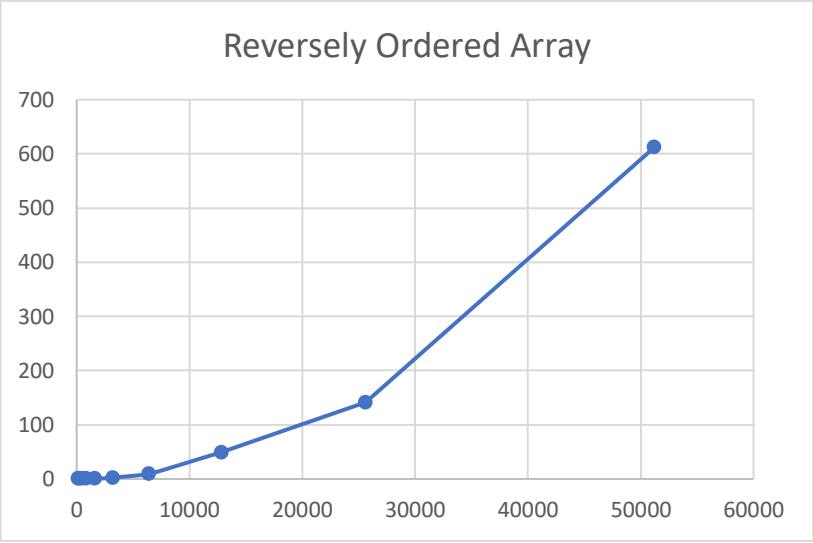
a) Ordered Array meantime vs length of input(n)

n(Length of Array)	Ordered Array
100	0.0358125
200	0.0105593
400	0.0348994
800	0.0288127
1600	0.0585436
3200	0.1139909
6400	0.1457443
12800	0.3627047
25600	0.4467751
51200	0.6404642



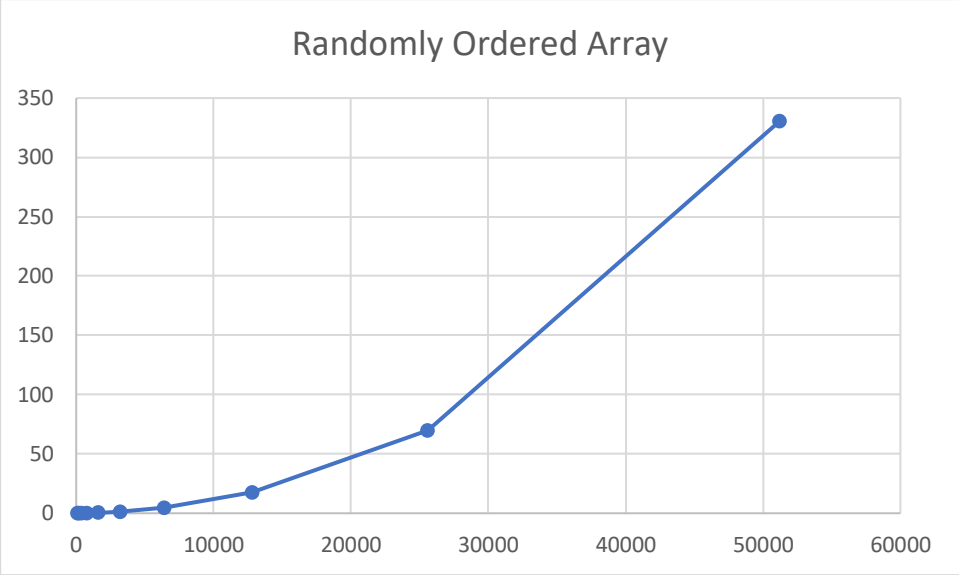
b) Reversely Ordered Array meantime vs length of input (n)

n(Length of Array)	Reversely Ordered Array
100	0.0824861
200	0.0595656
400	0.1915932
800	0.2279096
1600	0.5433482
3200	2.1686604
6400	8.7474809
12800	48.7065737
25600	141.0240318
51200	611.8534447



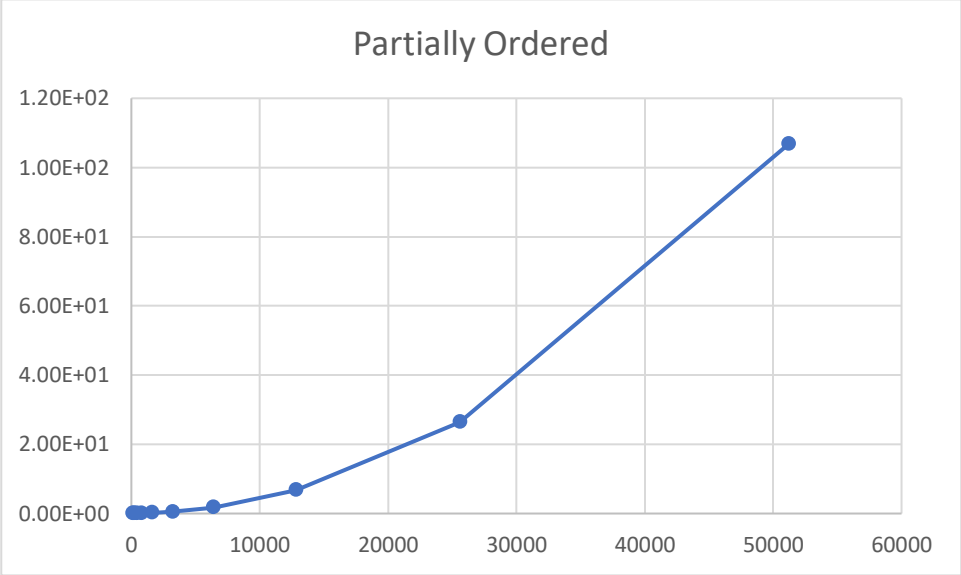
c) Randomly Ordered Array meantime vs length of input (n)

n(Length of Array)	Randomly Ordered Array
100	0.0032788
200	0.0080354
400	0.0231562
800	0.0824395
1600	0.2959718
3200	1.1775513
6400	4.5498351
12800	17.4374589
25600	69.6801161
51200	330.3842517

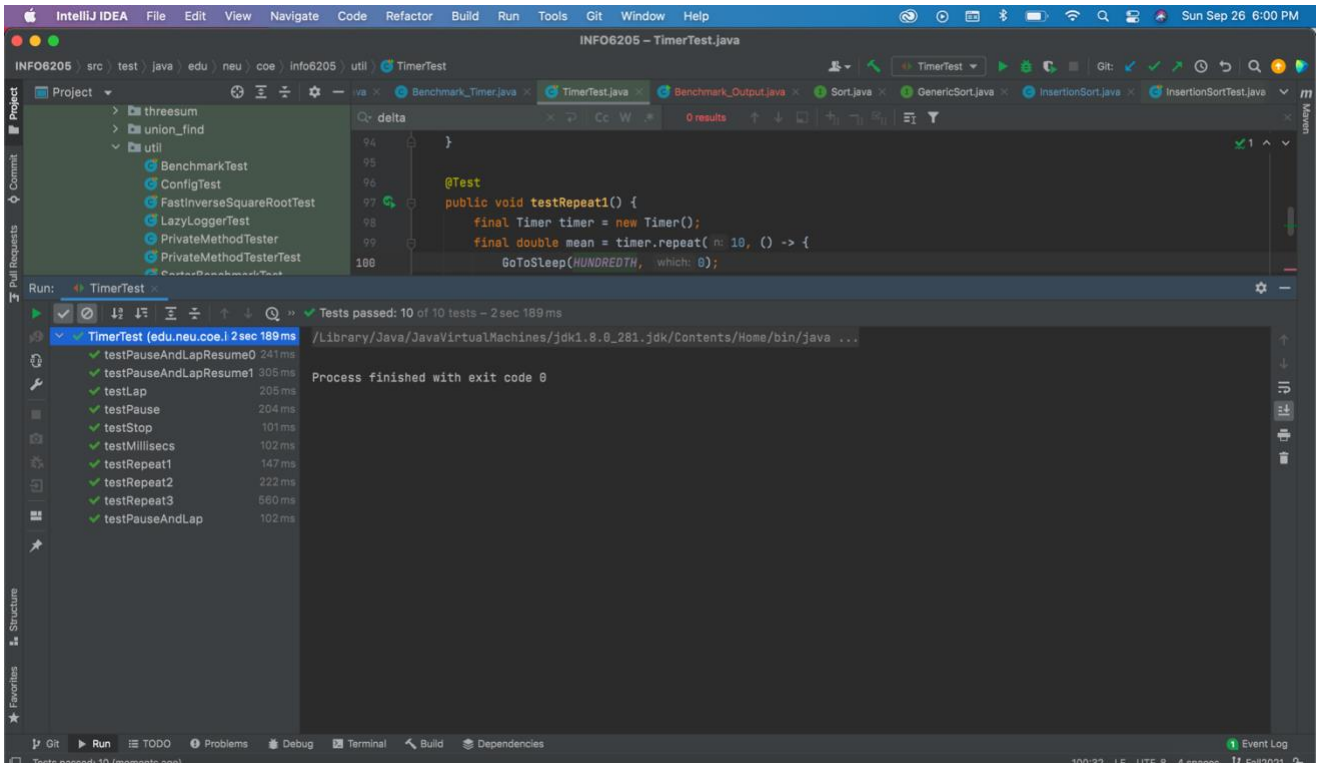


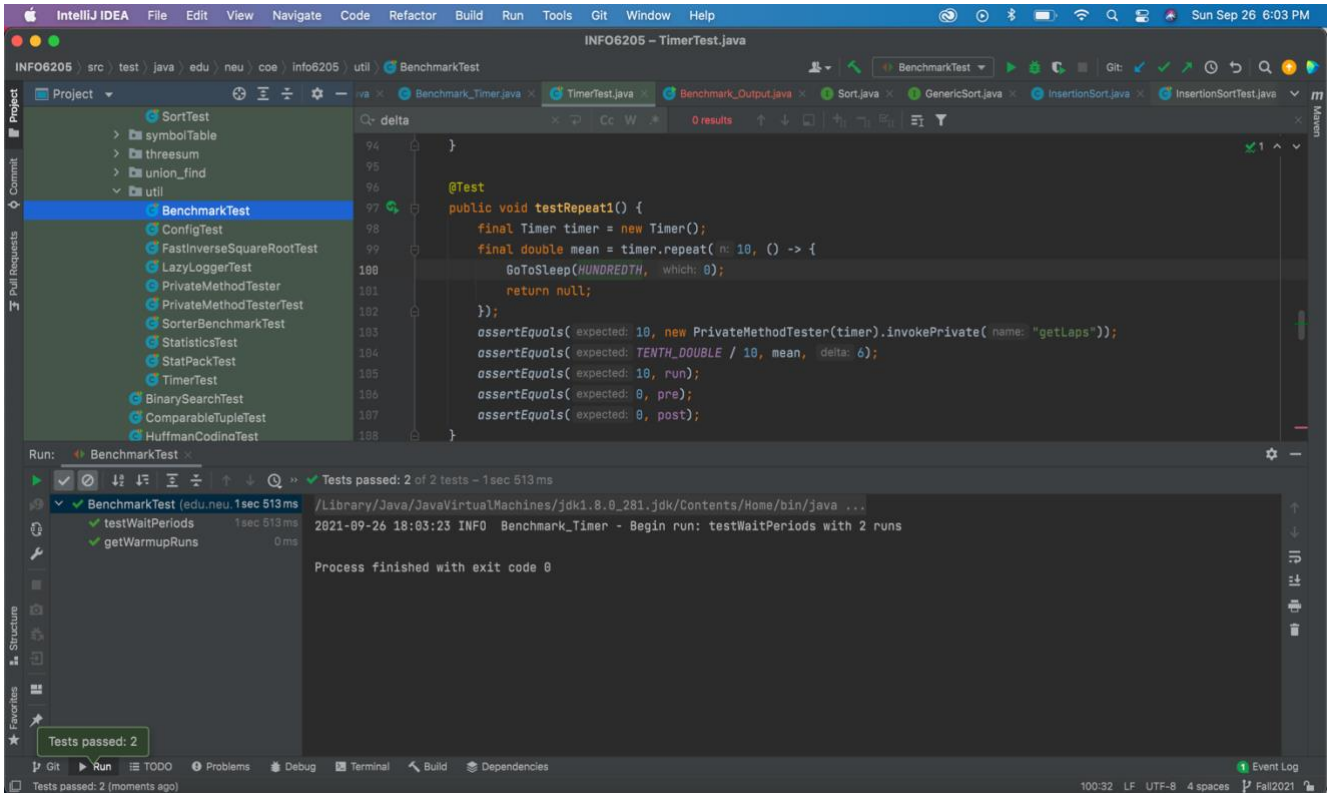
d) Partially Ordered Array meantime vs length of input (n)

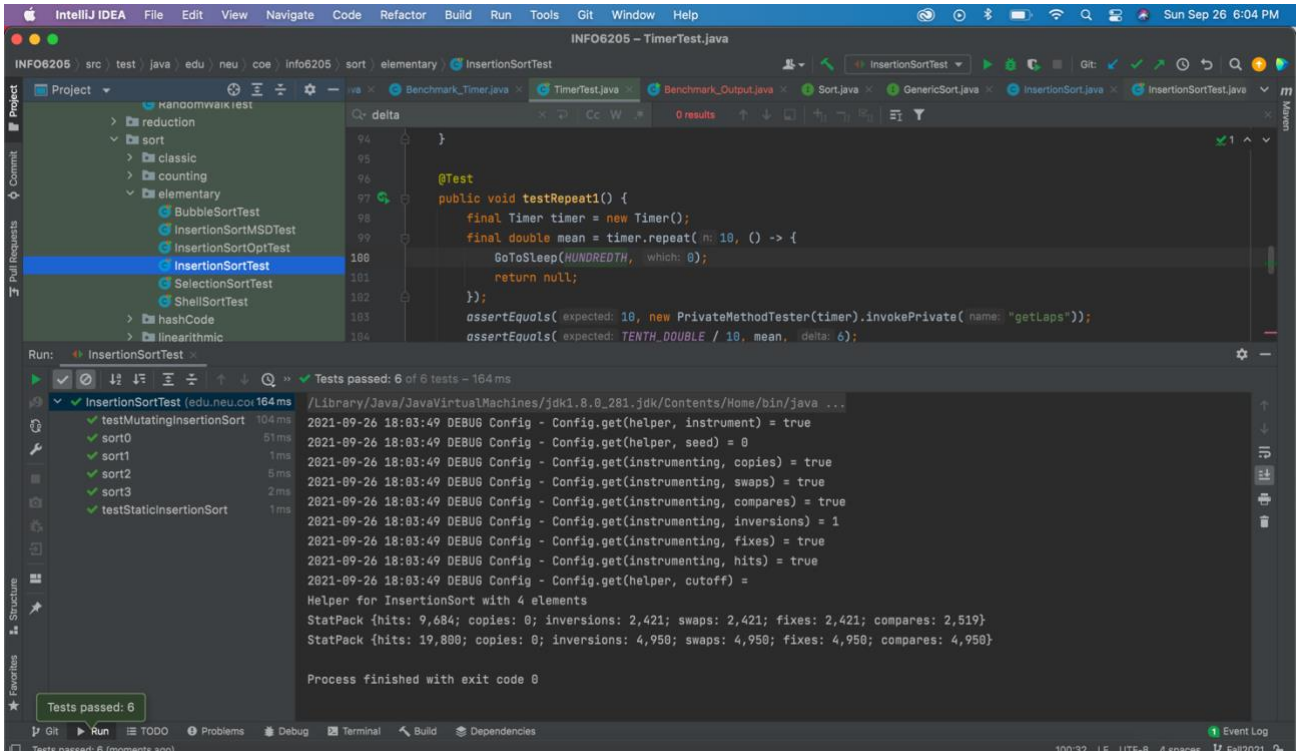
n(Length of Array)	Partially Ordered
100	3.13E-02
200	0.0339712
400	0.0350932
800	0.1168869
1600	0.1648076
3200	0.4651877
6400	1.6947624
12800	6.7495448
25600	26.3680016
51200	106.7523721



- **Unit tests result:**






IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Sun Sep 26 6:04 PM
INFO6205 – TimerTest.java
INFO6205 src test java edu neu coe info6205 sort elementary InsertionSortTest
Project
reduction
sort
classic
counting
elementary
BubbleSortTest
InsertionSortMSDTest
InsertionSortOptTest
InsertionSortTest
SelectionSortTest
ShellSortTest
hashCode
linearithmic
Benchmark_Timer.java
TimerTest.java
Benchmark_Output.java
Sort.java
GenericSort.java
InsertionSort.java
InsertionSortTest.java
delta
0 results
94 }
96 @Test
97 public void testRepeat1() {
98 final Timer timer = new Timer();
99 final double mean = timer.repeat(n: 10, () -> {
100 GoToSleep(HUNDREDTH, which: 0);
101 return null;
102 });
103 assertEquals(expected: 10, new PrivateMethodTester(timer).invokePrivate(name: "getLaps"));
104 assertEquals(expected: TENTH_DOUBLE / 10, mean, delta: 6);
Run: InsertionSortTest
Tests passed: 6 of 6 tests – 164 ms
InsertionSortTest (edu.neu.coe 164 ms
testMutatingInsertionSort 104 ms
sort0 51 ms
sort1 1 ms
sort2 5 ms
sort3 2 ms
testStaticInsertionSort 1 ms
/Library/Java/JavaVirtualMachines/jdk1.8.0_281.jdk/Contents/Home/bin/java ...
2021-09-26 18:03:49 DEBUG Config - Config.get(helper, instrument) = true
2021-09-26 18:03:49 DEBUG Config - Config.get(helper, seed) = 0
2021-09-26 18:03:49 DEBUG Config - Config.get(instrumenting, copies) = true
2021-09-26 18:03:49 DEBUG Config - Config.get(instrumenting, swaps) = true
2021-09-26 18:03:49 DEBUG Config - Config.get(instrumenting, compares) = true
2021-09-26 18:03:49 DEBUG Config - Config.get(instrumenting, inversions) = 1
2021-09-26 18:03:49 DEBUG Config - Config.get(instrumenting, fixes) = true
2021-09-26 18:03:49 DEBUG Config - Config.get(instrumenting, hits) = true
2021-09-26 18:03:49 DEBUG Config - Config.get(helper, cutoff) =
Helper for InsertionSort with 4 elements
StatPack {hits: 9,684; 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}
Process finished with exit code 0
Tests passed: 6
Git Run TODO Problems Debug Terminal Build Dependencies
Event Log