Shuo Wang (001533020)

**Program Structures & Algorithms**

**Fall 2021**

**Assignment No. 2**

* **Task (List down the tasks performed in the Assignment)**

(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.

(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.

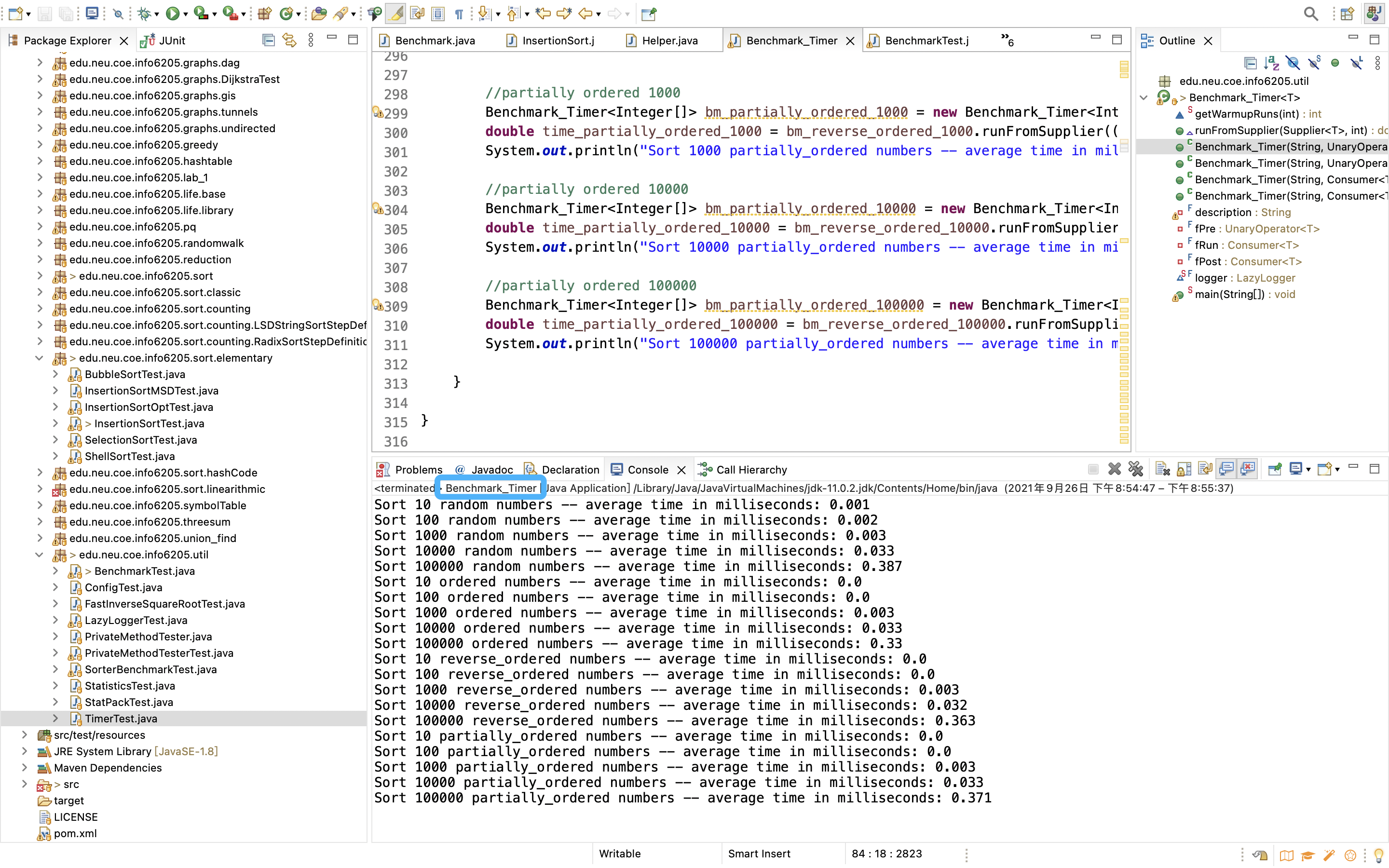
**(Main function is in Benchmark\_Time.java)**

Timer.java/ TimerTest.java/ Benchmark.java/ Benchmark\_Time.java/ BenchmarkTast.java/ InsertionSort.Java/ InsertionSortTest.Java

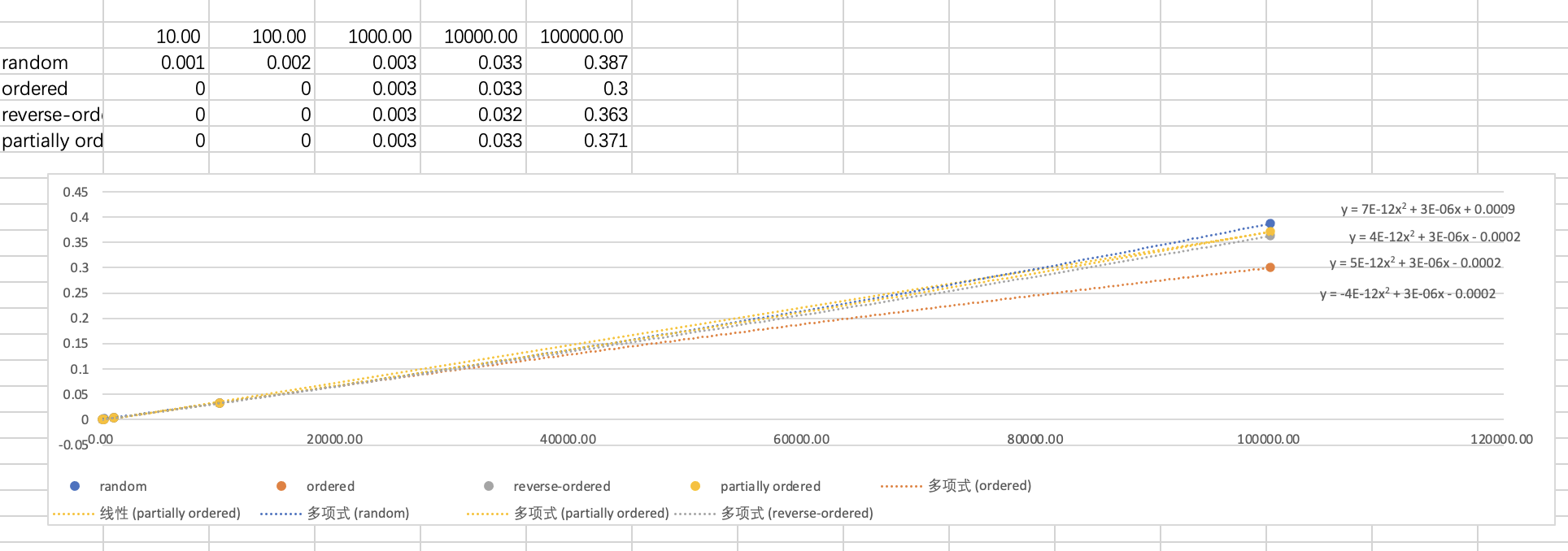
* **Relationship Conclusion:** **O(n2)**
* **Evidence to support the conclusion:**

It’s about O(n2), though is not very clearly in the picture. In my program, when it sorting ordered array, it is fastest. And when it sorting random array, it is slowest. But logically, sorting reverse ordered array should be the slowest one.

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



1. **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)**

