Kshitij Verma NUID: 001305285

# INFO 6205

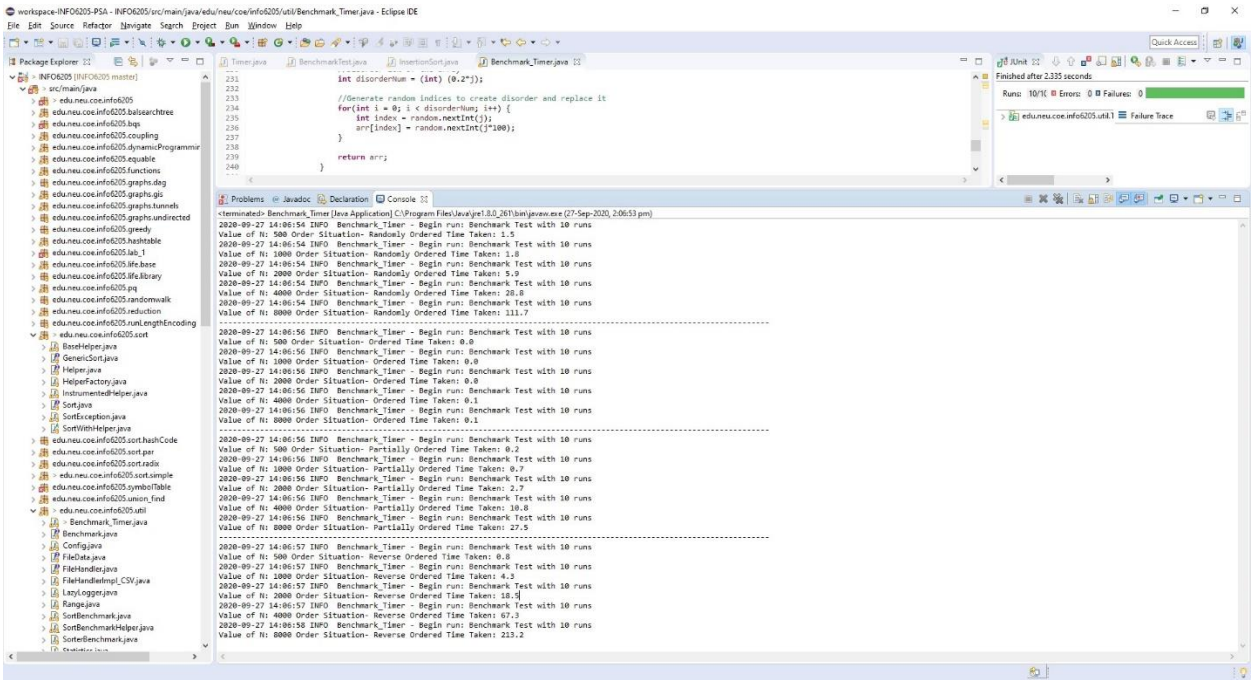
# Program Structures & Algorithms

# Fall 2020

# Assignment No: 2

- **Task:** To implement the class Timer.java and InsertionSort.java and also measure the running times of the sort using four different initial array ordering situations i.e. Random, Ordered, Partially-Ordered and Reverse-ordered array.

- **Output:** Below is the output received for 5 different values of n with different initial array ordering situations ran 10 times each and doubling the value of n each time-



Console Output-

```
2020-09-27 14:06:54 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 500 Order Situation- Randomly Ordered Time Taken: 1.5
2020-09-27 14:06:54 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 1000 Order Situation- Randomly Ordered Time Taken: 1.8
2020-09-27 14:06:54 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 2000 Order Situation- Randomly Ordered Time Taken: 5.9
2020-09-27 14:06:54 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 4000 Order Situation- Randomly Ordered Time Taken: 28.8
2020-09-27 14:06:54 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
```

```
Value of N: 8000 Order Situation- Randomly Ordered Time Taken: 111.7
--------------------------------------------------------------------------------
2020-09-27 14:06:56 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 500 Order Situation- Ordered Time Taken: 0.0
2020-09-27 14:06:56 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 1000 Order Situation- Ordered Time Taken: 0.0
2020-09-27 14:06:56 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 2000 Order Situation- Ordered Time Taken: 0.0
2020-09-27 14:06:56 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 4000 Order Situation- Ordered Time Taken: 0.1
2020-09-27 14:06:56 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 8000 Order Situation- Ordered Time Taken: 0.1
--------------------------------------------------------------------------------
2020-09-27 14:06:56 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 500 Order Situation- Partially Ordered Time Taken: 0.2
2020-09-27 14:06:56 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 1000 Order Situation- Partially Ordered Time Taken: 0.7
2020-09-27 14:06:56 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 2000 Order Situation- Partially Ordered Time Taken: 2.7
2020-09-27 14:06:56 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 4000 Order Situation- Partially Ordered Time Taken: 10.8
2020-09-27 14:06:56 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 8000 Order Situation- Partially Ordered Time Taken: 27.5
--------------------------------------------------------------------------------
2020-09-27 14:06:57 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 500 Order Situation- Reverse Ordered Time Taken: 0.8
2020-09-27 14:06:57 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 1000 Order Situation- Reverse Ordered Time Taken: 4.3
2020-09-27 14:06:57 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 2000 Order Situation- Reverse Ordered Time Taken: 18.5
2020-09-27 14:06:57 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 4000 Order Situation- Reverse Ordered Time Taken: 67.3
2020-09-27 14:06:58 INFO  Benchmark_Timer - Begin run: Benchmark Test with 10 runs
Value of N: 8000 Order Situation- Reverse Ordered Time Taken: 213.2
```

- **Relationship conclusion:** It can be concluded from the results of the benchmark test:
    - In best-case scenario which happens when the array is sorted, the insertion sort runs in O(n) time as it simply compares the elements with no swapping required
    - In the average-case when the array is sorted in randomly-ordered or partially ordered manner the insertion sort runs in O($n^2$) time
    - Similarly, in the worst-case scenario, when the array is sorted in the reverse order manner the insertion sort takes O($n^2$) time because for inserting the last element, the algorithm will need n-1 comparisons and perform n-1 swaps and so on.

      Therefore, we get - 2(1+2+3+……+n-2+n-1)

      = n(n-1) [Using Sum of n formula]

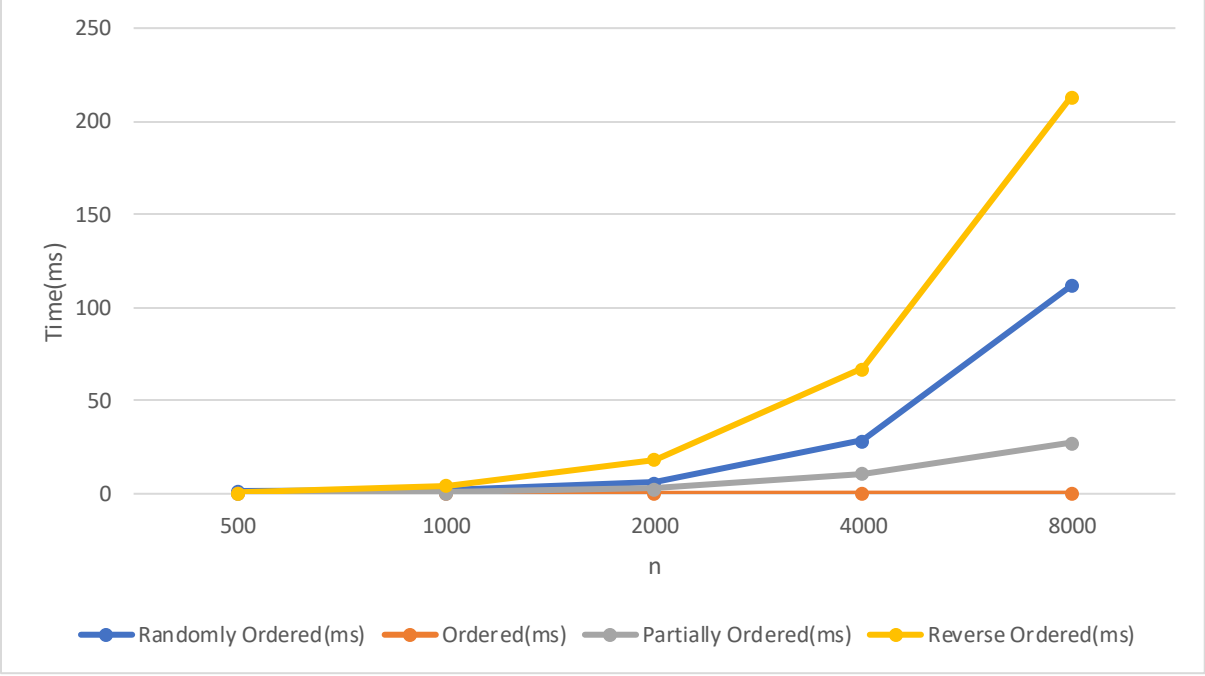
      = $n^2$ – n

      ~ $n^2$

- Thus, it can be derived that the following pattern is followed by the run time of insertion sort for different order situation in arrays-
  Ordered < Partially Ordered < Randomly Ordered < Reverse Ordered

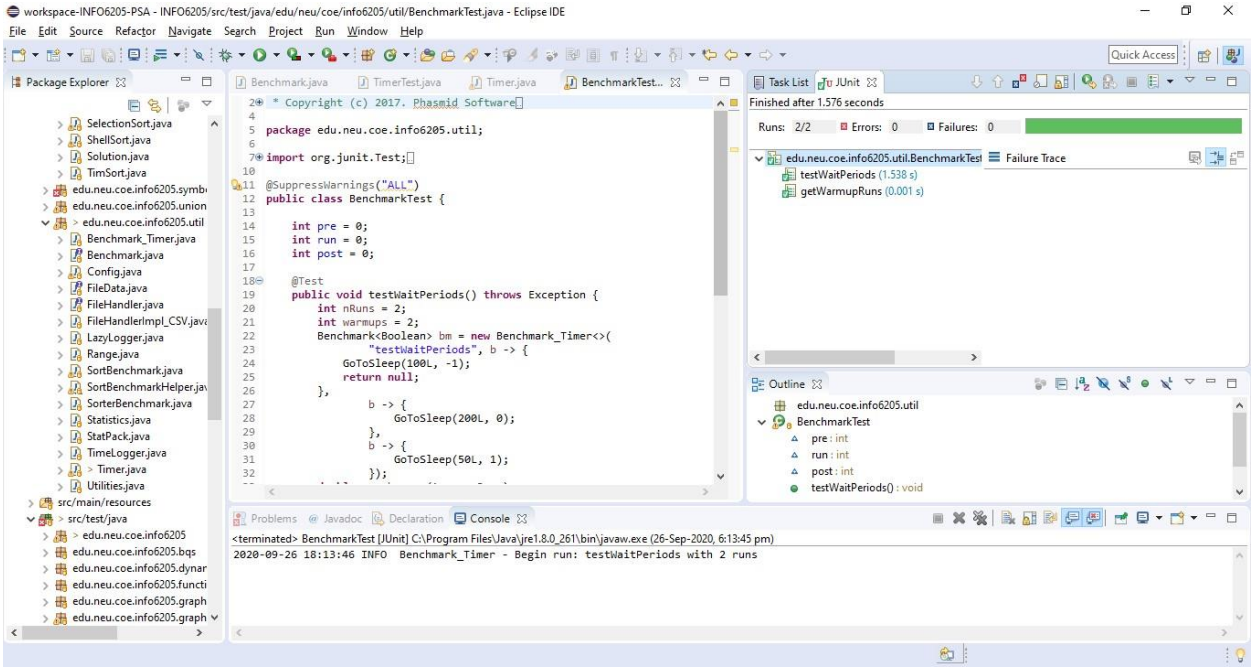
- **Evidence to support relationship**: I have attached a chart and a table stating the data of the different output observed for the different set of inputs of n. As a result, we can see the proportionate increase in the run time of insertion sort with different sorted situation of arrays-

| Value of n | Randomly Ordered(ms) | Ordered(ms) | Partially Ordered(ms) | Reverse Ordered(ms) |
|---|---|---|---|---|
| 500 | 1.5 | 0.0 | 0.2 | 0.8 |
| 1000 | 1.8 | 0.0 | 0.7 | 4.3 |
| 2000 | 5.9 | 0.0 | 2.7 | 18.5 |
| 4000 | 28.8 | 0.1 | 10.8 | 67.3 |
| 8000 | 111.7 | 0.1 | 27.5 | 213.2 |



- **Screenshot of Unit test passing:** Below is the screenshot of all the unit tests which ran successfully

**BenchmarkTest**




**TimerTest**