Shrusti Rajesh Chheda

NUID : 002196756

Program Structures and Algorithms

Fall 2021

Assignment No. 2 – Benchmark

- Tasks in the assignment:

Part 1: Timer and Benchmark

1. Implemented method getClock to get system time in nanosec keeping in mind precision and method toMillisecs which converts clock ticks currently in nanosec to millisec.
2. Implemented method repeat to calculate time taken to run a function based on the given requirements.
3. Ran the TimerTest and BenchmarkTest to verify values of meanLapTime based on the given input values.

Part 2: Insertion Sort

1. Implemented the sort method using the helper class to sort a given array.
2. Verified using the InsertionSortTest for different scenarios which included mutatingSort and static insertion sort.

Part 3: Benchmarking Insertion Sort

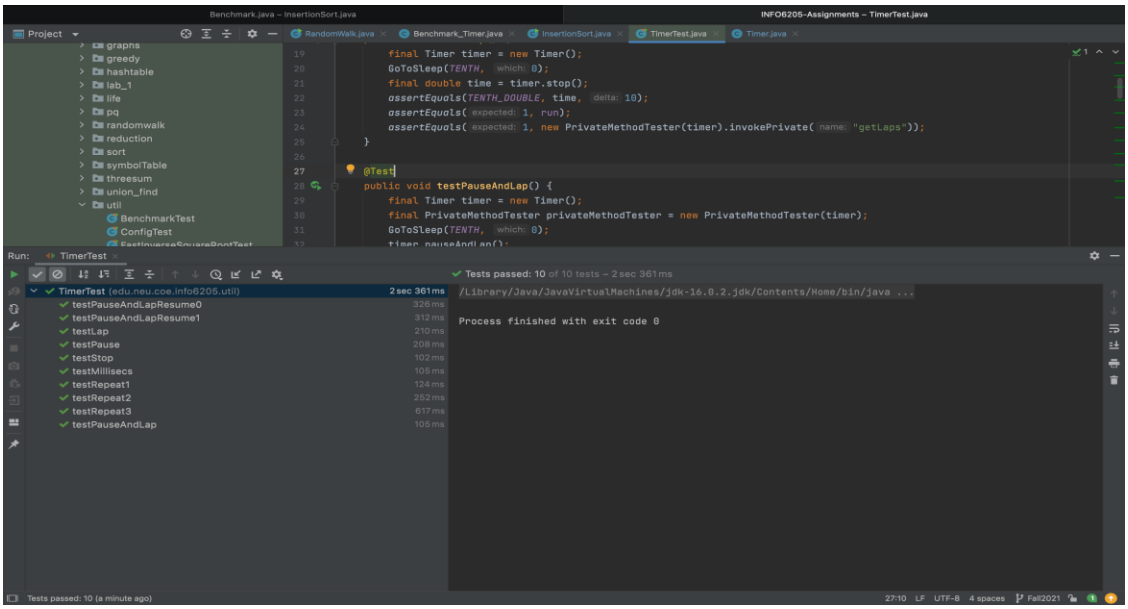
1. Created a class InsertionSortBenchmark which contains a main method.
2. This main method creates an object of the benchmark_timer and runs the timer to find running time for this sort. Here 4 different types of arrays are used ie four different initial array ordering situations: random, ordered, partially-ordered and reverse-ordered.
3. For each of these arrays doubling method is used to determine the order of growth.

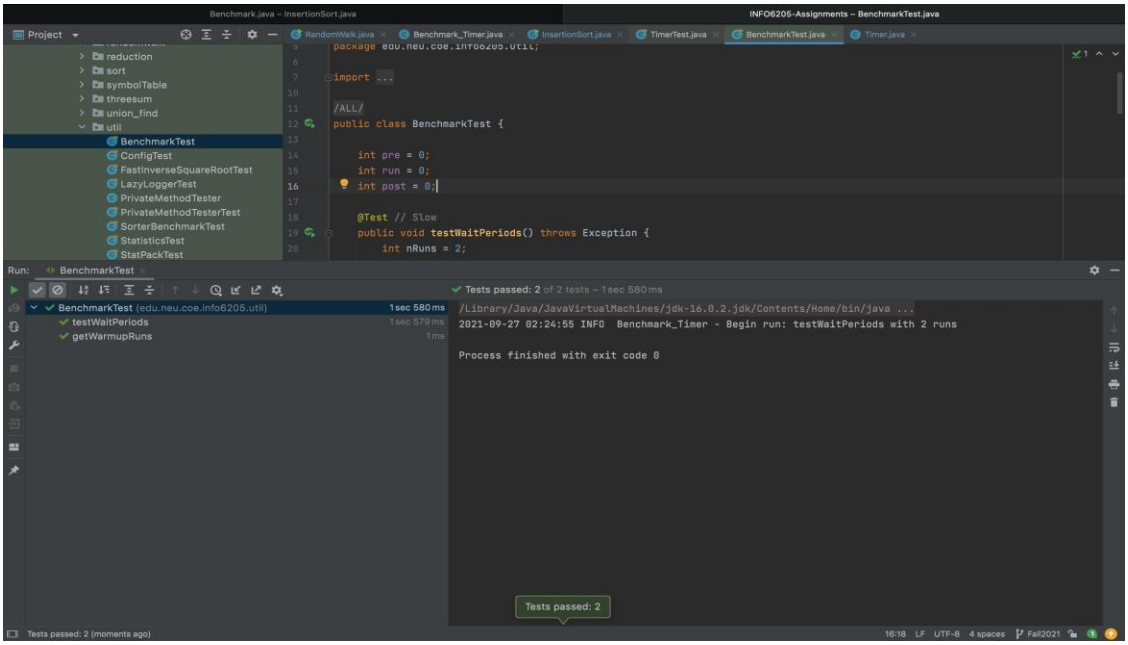
- Relationship Conclusion

  Complexity ranges between $O(N^2)$ and $O(N)$

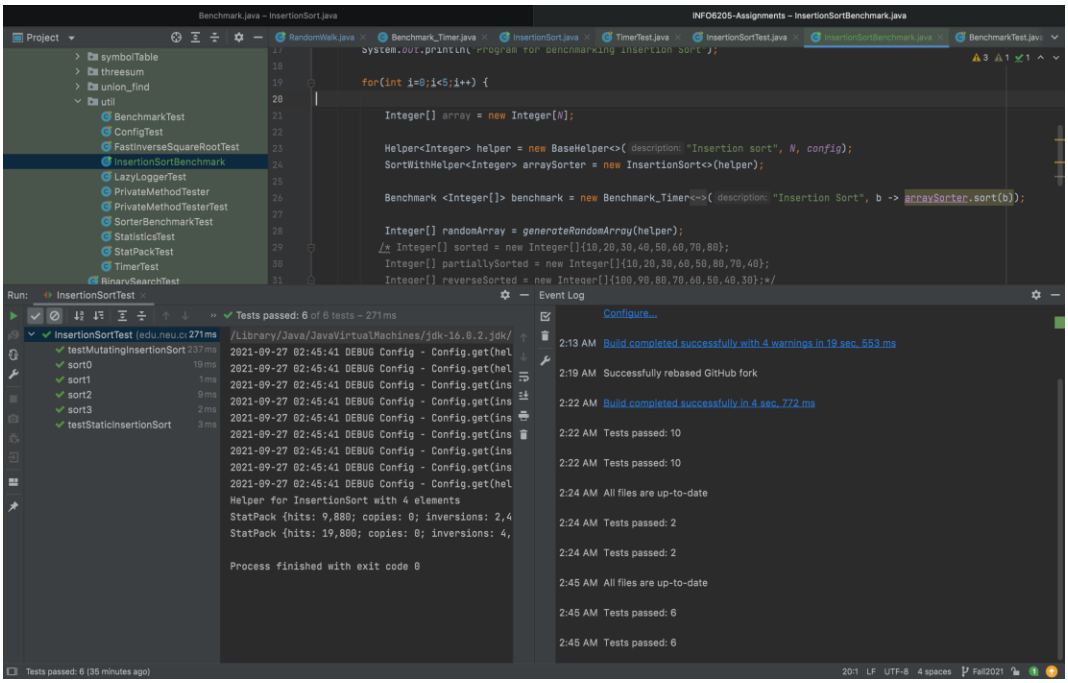
- Evidence to support conclusions

1. Snapshot of unit test part 1

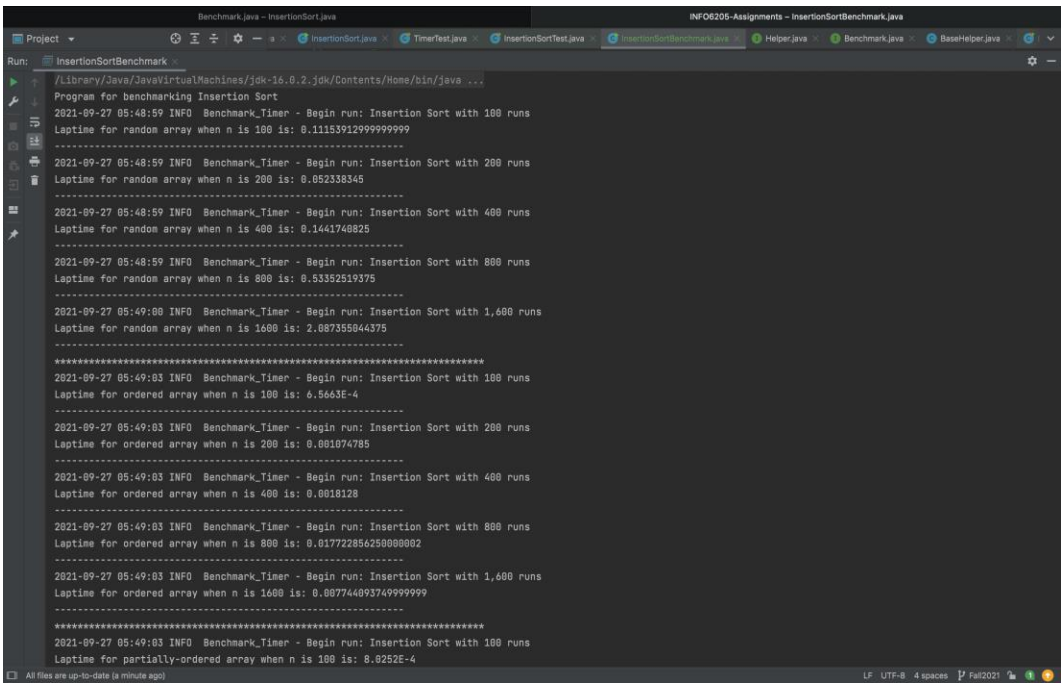


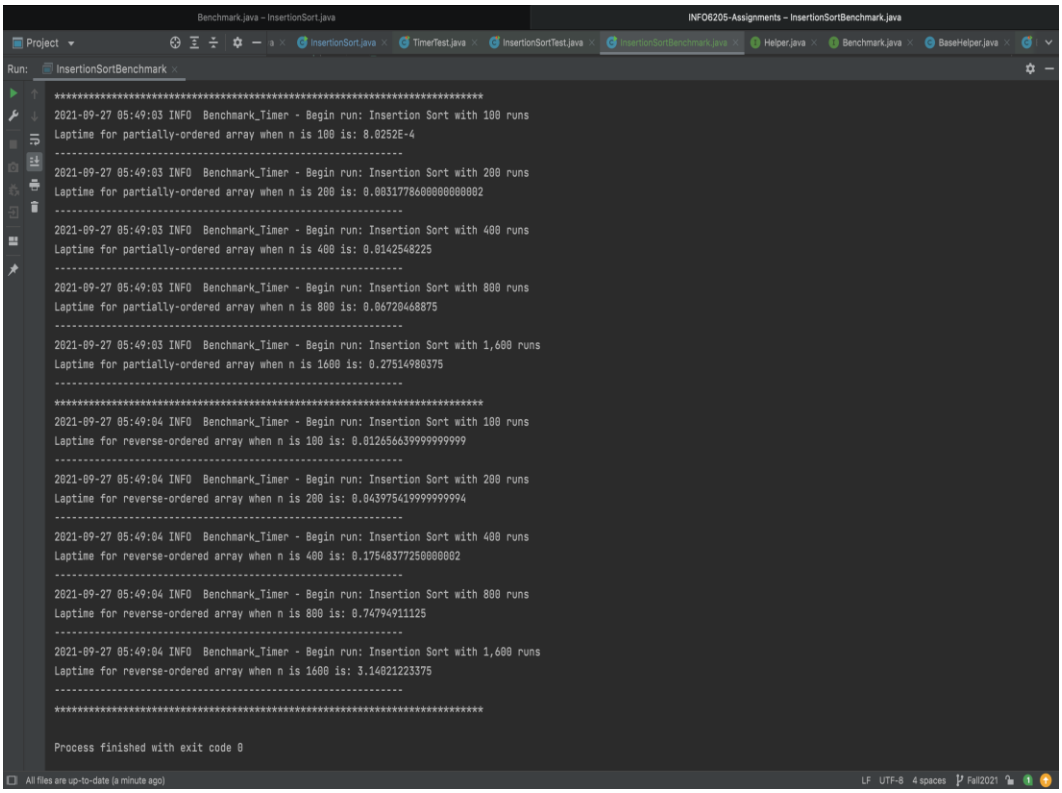


2. Snapshot of unit test part 2



3. Snapshot of output part 3





Randomly Ordered Array

N	Running time
100	0.1115
200	0.0523
400	0.1442
800	0.5335
1600	2.0873

Ordered Array

N	Running time
100	6.5663E-4
200	0.0011
400	0. 0018
800	0. 0177
1600	0.0077

Partially Ordered Array

N	Running time
100	8.0252E-4
200	0. 0032
400	0. 0143
800	0. 0672

1600	0. 2751

Reverse Ordered Array

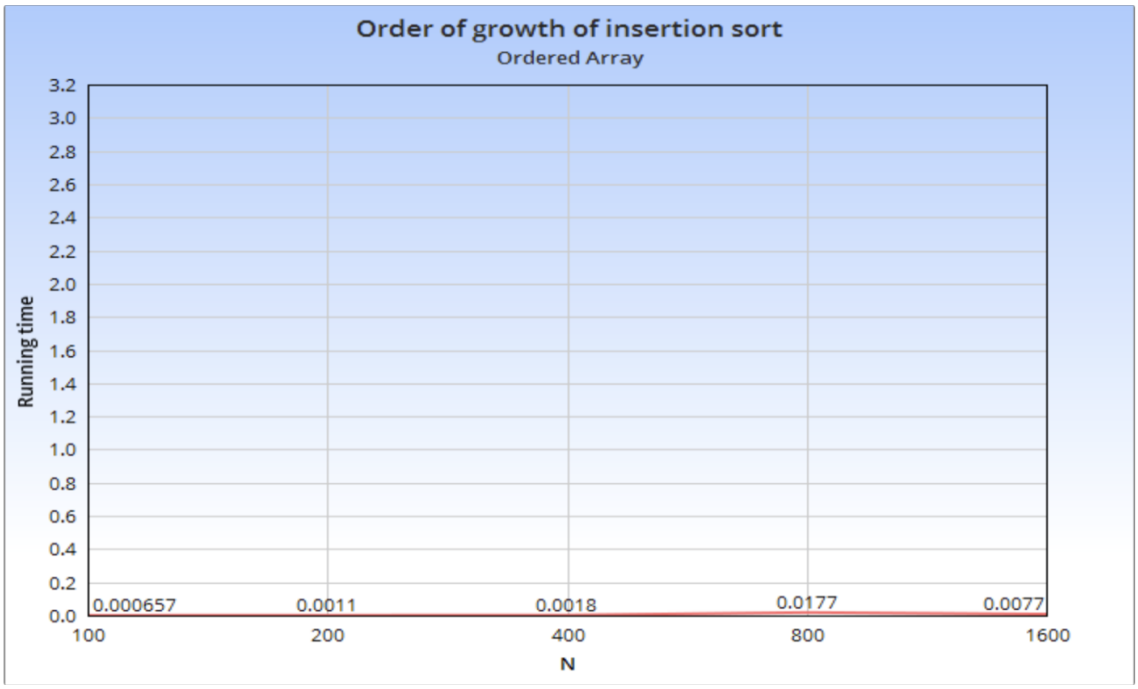
N	Running time
100	0. 0127
200	0. 0440
400	0. 1755
800	0. 7480
1600	3.1402

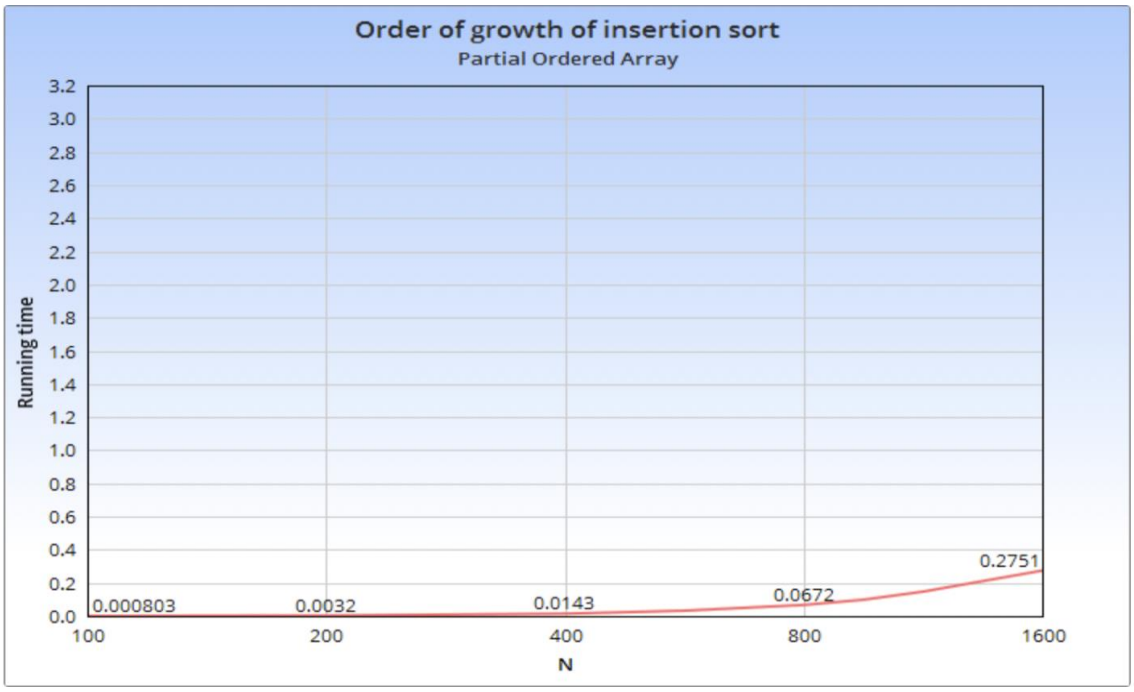
- Observations:

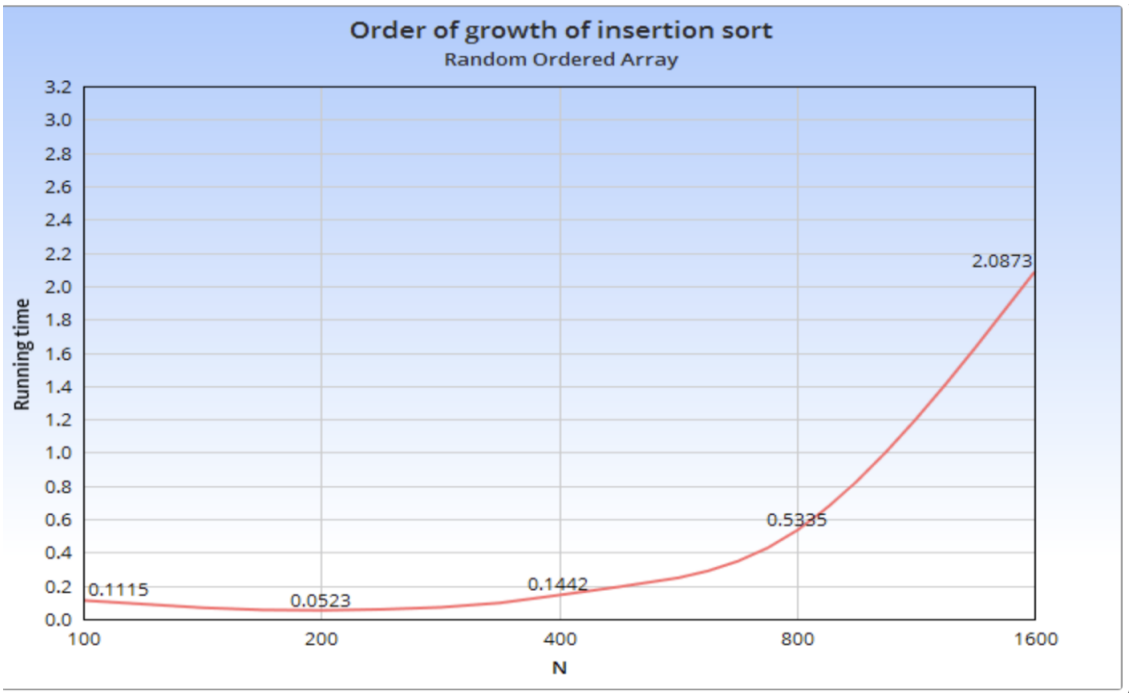
As the length of array increases, running time of the algorithm increases.

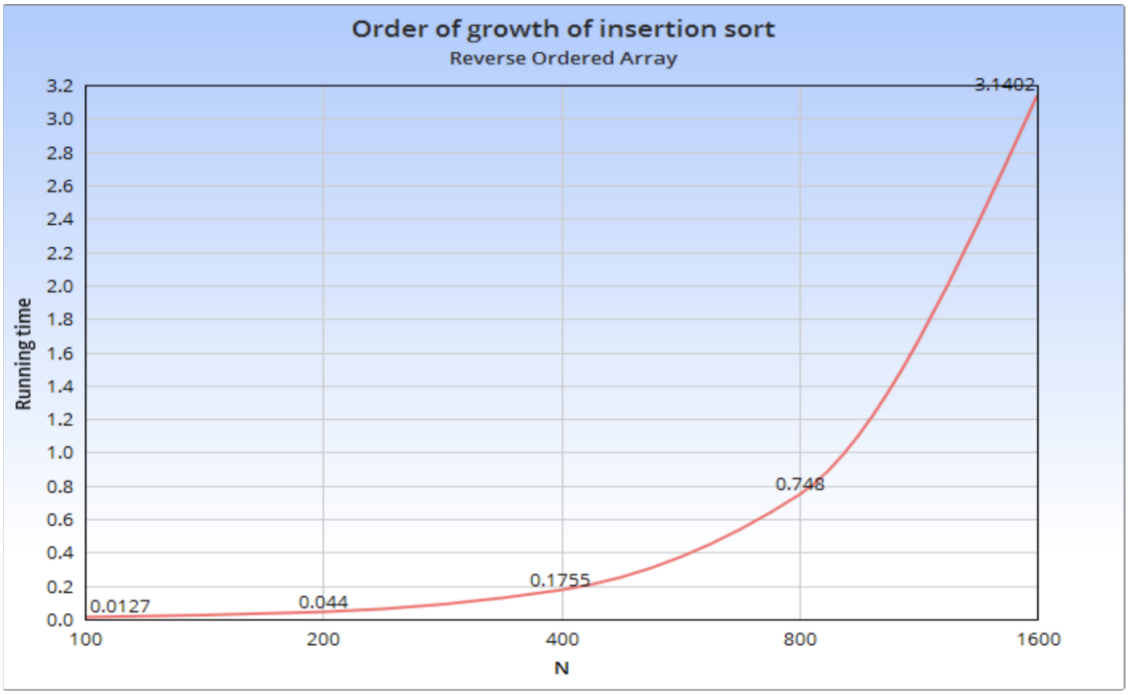
In case of a reverse-order array the running time for the algorithm is the highest when N is highest. Thus with this input we get the worst case complexity of insertion sort which is $O(N^2)$

The random array has the next highest running time while running times of ordered array is the least with O(N) complexity which is almost a linear line.




Order of growth of insertion sort
Partial Ordered Array
Running time
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.000803
0.0032
0.0143
0.0672
0.2751
100
200
400
800
1600
N


Order of growth of insertion sort
Random Ordered Array
Running time
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.1115
0.0523
0.1442
0.5335
2.0873
100
200
400
800
1600
N



5. All test cases passed successfully.