Archil Lelashvili – NUID 001522269
Program Structures & Algorithms, Spring 2021
Assignment No 2

Task:

1. We 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. To check the implementation the unit tests BenchmarkTest and TimerTest are used.
2. Implement InsertionSort (in the InsertionSort class) by simply looking up the insertion code used by Arrays.sort. You should use the helper.swap method although you could also just copy that from the same source code. You should of course run the unit tests in InsertionSortTest.
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.

Output:

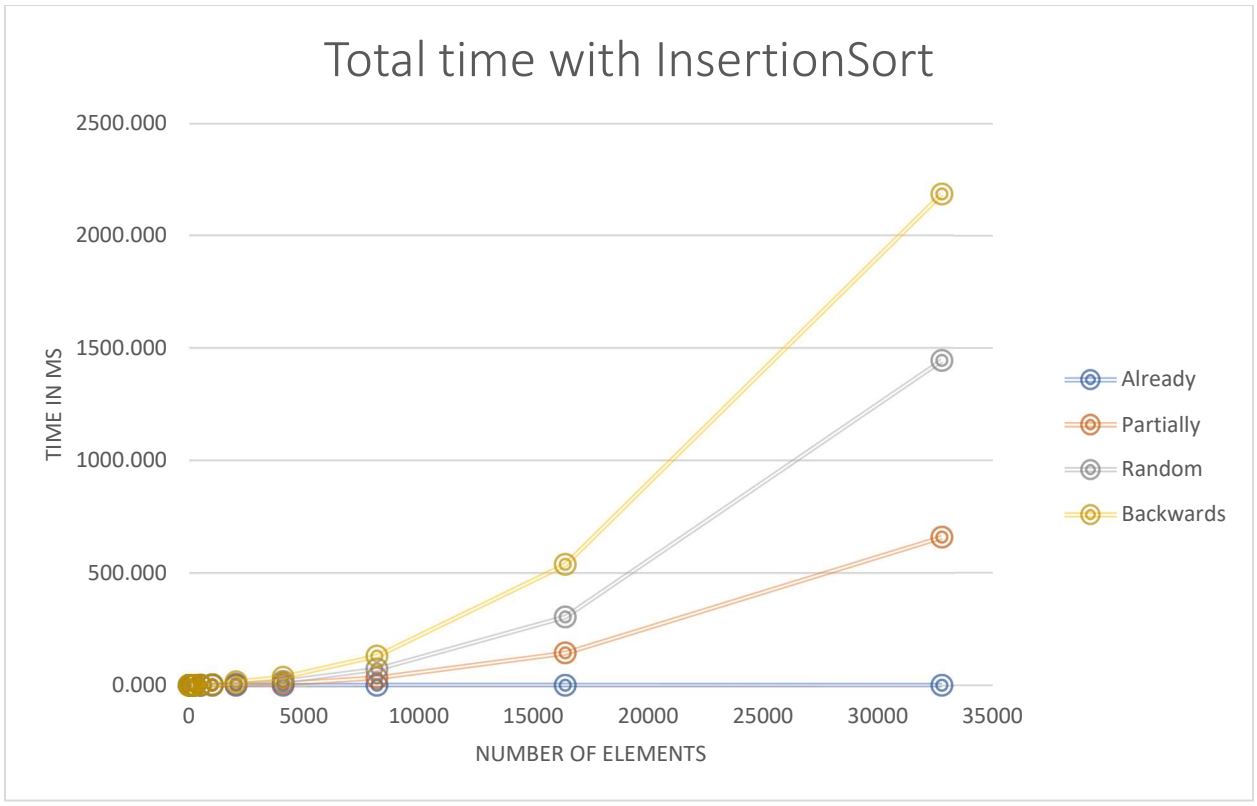
P.1 & P.2 were implemented as required. Please the unit tests below.

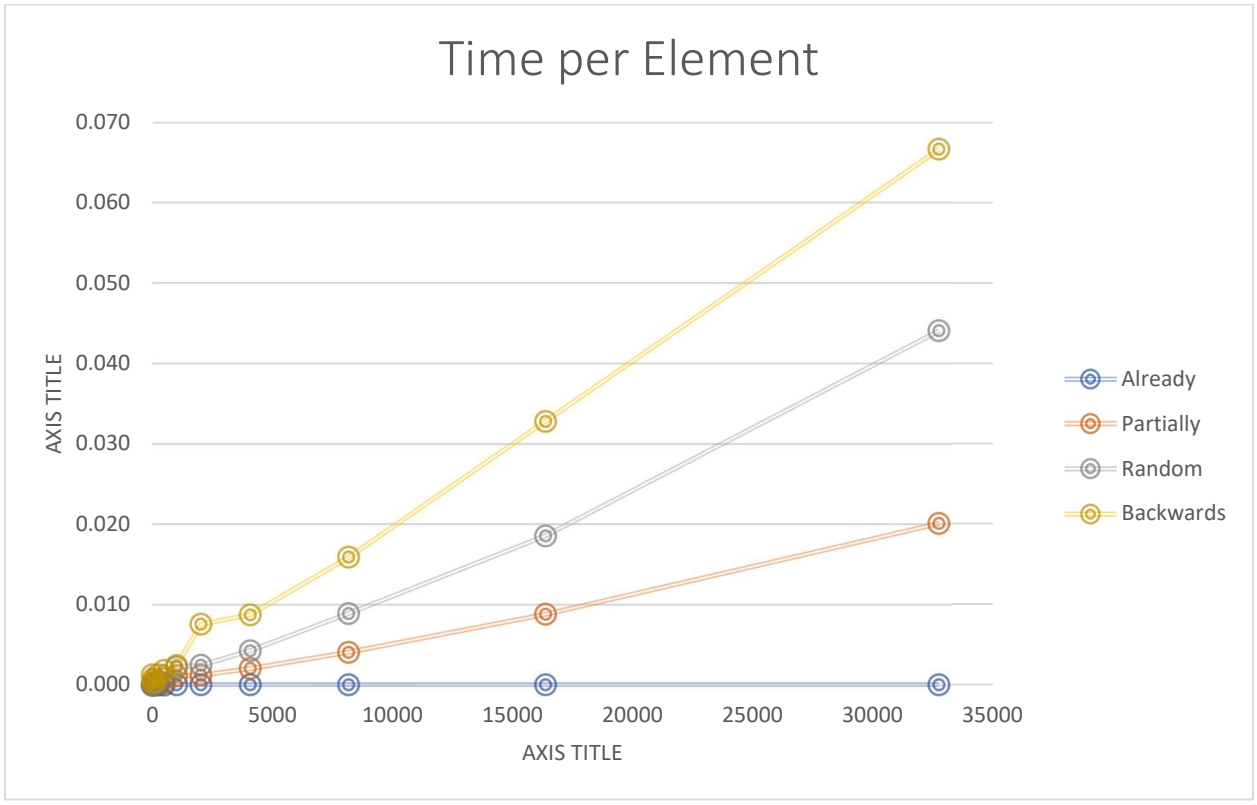
For P.3 – a new class is implanted:

test -> java -> edu.neu.coe.info6205 -> sort -> simple -> CustomBenchmarks.class

It runs the InsertionSort for arrays starting with size of 2^4(16 elements) up to 2^15 (32k elements) under four different scenarios with random, ordered, partially-ordered and reverse-ordered arrays. The simulations run for 100 times each. We use Excel to show the output in a table and a chart.

	Total time with InsertionSort				Time per Element (multiplied by 1000)				Growth			
N	**Already**	**Partially**	**Random**	**Backwards**	**Already**	**Partially**	**Random**	**Backwards**	**Already**	**Partially**	**Random**	**Backwards**
16	0.000	0.000	0.000	0.020	0.000	0.000	0.000	0.001				
32	0.000	0.000	0.010	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.250
64	0.000	0.000	0.000	0.050	0.000	0.000	0.000	0.001	0.000	0.000	0.000	2.500
128	0.000	0.010	0.020	0.050	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.500
256	0.000	0.050	0.090	0.320	0.000	0.000	0.000	0.001	0.000	2.500	2.250	3.200
512	0.000	0.160	0.390	0.930	0.000	0.000	0.001	0.002	0.000	1.600	2.167	1.453
1024	0.010	1.150	2.270	2.480	0.000	0.001	0.002	0.002	0.000	3.594	2.910	1.333
2048	0.020	2.480	5.020	15.470	0.000	0.001	0.002	0.008	1.000	1.078	1.106	3.119
4096	0.030	8.350	17.470	35.690	0.000	0.002	0.004	0.009	0.750	1.683	1.740	1.154
8192	0.130	33.240	72.690	130.140	0.000	0.004	0.009	0.016	2.167	1.990	2.080	1.823
16384	0.140	143.910	303.720	536.880	0.000	0.009	0.019	0.033	0.538	2.165	2.089	2.063
32768	0.190	657.480	1442.650	2182.400	0.000	0.020	0.044	0.067	0.679	2.284	2.375	2.032

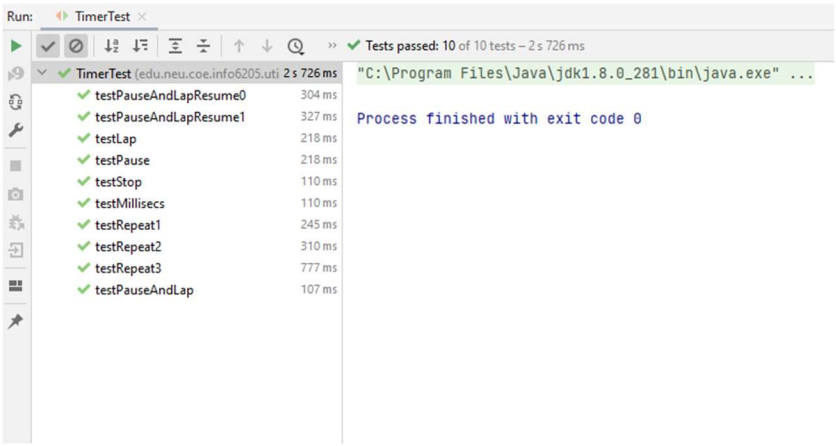

Total time with InsertionSort
2500.000
2000.000
1500.000
1000.000
500.000
0.000
TIME IN MS
0
5000
10000
15000
20000
25000
30000
35000
NUMBER OF ELEMENTS
Already
Partially
Random
Backwards

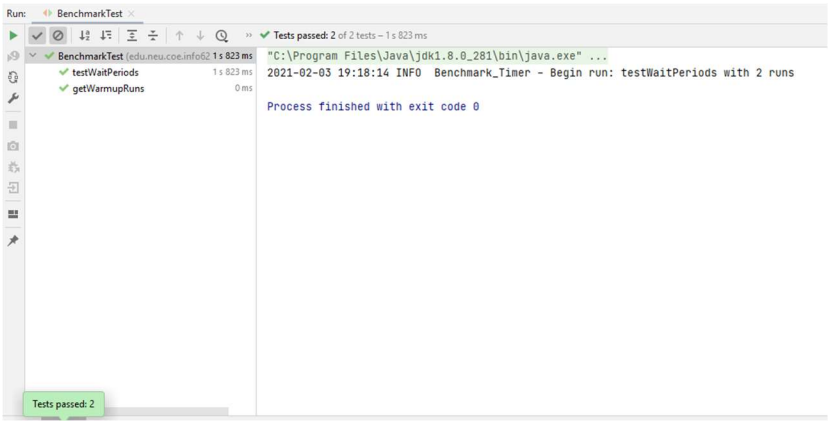

Time per Element
0.070
0.060
0.050
0.040
0.030
0.020
0.010
0.000
AXIS TITLE
0
5000
10000
15000
20000
25000
30000
35000
AXIS TITLE
Already
Partially
Random
Backwards

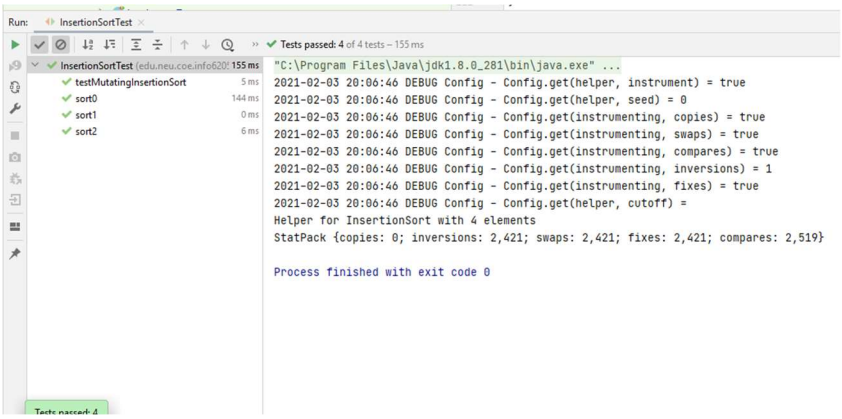
Relationship Conclusion:

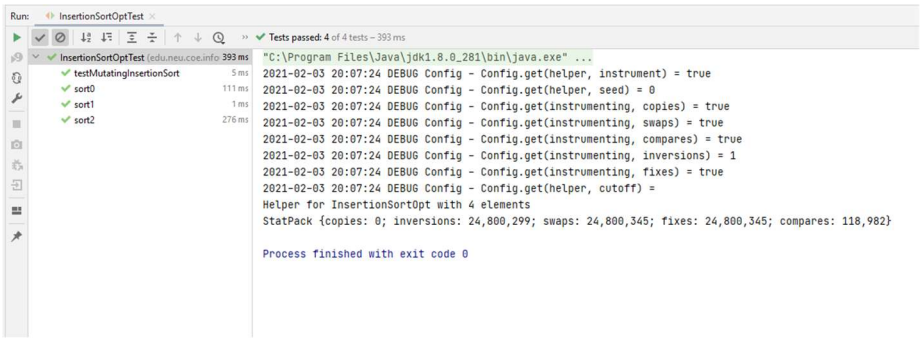
We can see that the time spent **per element** in worst case scenario **increases linearly** and looks like in worst case scenario we are getting O(N^2) complexity. Even though with Random and Partially sorted arrays we have slower growth, it is still far from constant.

Unit Test Results:








Run: InsertionSortOptTest
Tests passed: 4 of 4 tests – 393 ms
InsertionSortOptTest (edu.neu.coe.info 393 ms
testMutatingInsertionSort 5 ms
sort0 111 ms
sort1 1 ms
sort2 276 ms
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...
2021-02-03 20:07:24 DEBUG Config - Config.get(helper, instrument) = true
2021-02-03 20:07:24 DEBUG Config - Config.get(helper, seed) = 0
2021-02-03 20:07:24 DEBUG Config - Config.get(instrumenting, copies) = true
2021-02-03 20:07:24 DEBUG Config - Config.get(instrumenting, swaps) = true
2021-02-03 20:07:24 DEBUG Config - Config.get(instrumenting, compares) = true
2021-02-03 20:07:24 DEBUG Config - Config.get(instrumenting, inversions) = 1
2021-02-03 20:07:24 DEBUG Config - Config.get(instrumenting, fixes) = true
2021-02-03 20:07:24 DEBUG Config - Config.get(helper, cutoff) =
Helper for InsertionSortOpt with 4 elements
StatPack {copies: 0; inversions: 24,800,299; swaps: 24,800,345; fixes: 24,800,345; compares: 118,982}
Process finished with exit code 0