

Program Structures & Algorithms

Spring 2022

Assignment No. 2

Name: Jayanth Vakkalagadda
(NUID): 002950342

Task

- (Part 1) Implementation of three methods in *Timer.java*, & check this implementation by running the unit tests in *BenchmarkTest.java* and *TimerTest.java*
- (Part 2) implementation of *InsertionSort* (in the *InsertionSort* class) & check this implementation by running the unit tests in *InsertionSortTest*
- (Part 3) Implementation of a main program to 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.
- Using doubling method for choosing n and test for at least five values of n
- Drawing conclusions from the observations regarding the order of growth

Relationship Conclusion

- Order of growth of the running time of Insertion Sort (Randomly ordered array of size N) is $\approx N^{2.04}$
- Order of growth of the running time of Insertion Sort (Ordered array of size N) is $\approx N^{0.92}$
- Order of growth of the running time of Insertion Sort (Partially ordered array of size N) is $\approx N^{1.97}$
- Order of growth of the running time of Insertion Sort (Reverse ordered array of size N) is $\approx N^{2.07}$
- In terms of order of growth, for the running time of Insertion sort:

Ordered < Partially Ordered < Randomly Ordered < Reverse Ordered

Evidence to the Conclusion

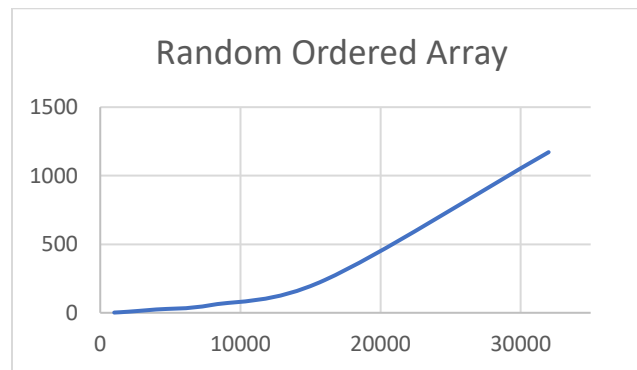
- Running time of the insertion sort for an array of 'n' numbers has been captured
- Each time the size of the array would be doubled and running time would be captured again (5 different sizes of array)
- Every time, we run the insertion sort algorithm, we make sure to test on four different states of the array (Ordered, Partially Ordered, Randomly Ordered, Reverse Ordered)

Random Ordered Array

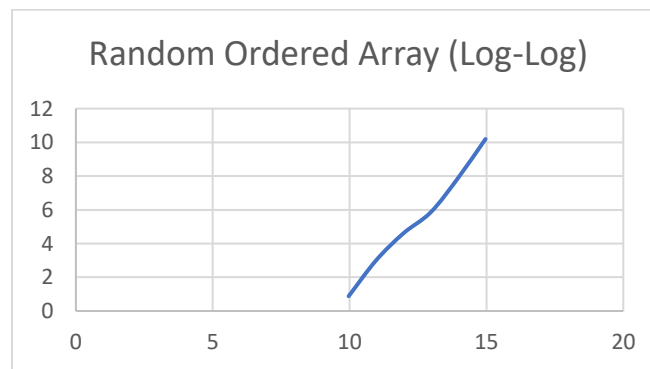
Various sizes of the Array and the running time of the Insertion sort

Randomly Ordered Array						
Array Size	Time	Ratio (Time/Previous Time)	lg(Array Size)	lg(Time)	Log Ratio	Slope
1000	1.84	-	9.97	0.88	11.33	
2000	7.99	4.34	10.97	3.00	3.66	2.12
4000	24.35	3.05	11.97	4.61	2.60	1.61
8000	57.85	2.38	12.97	5.85	2.21	1.25
16000	237.68	4.11	13.97	7.89	1.77	2.04
32000	1171.63	4.93	14.97	10.19	1.47	2.30
Avg slope						2.04

Analysis of experimental data (the running time of insertion sort with random ordered input)



Standard Plot: Running time T(n) Vs Array size N



Log-Log Plot: lg(T(n)) Vs lg(N)

The equation of the log-log plot is

$$\lg(T(N)) = 2.04 \lg N + \lg a$$

Which is equivalent to,

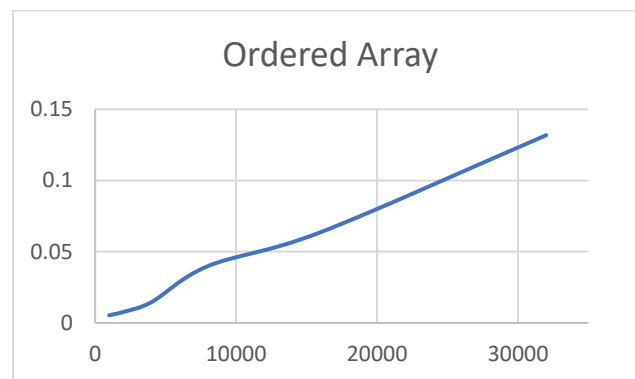
$$T(N) = aN^{2.04}$$

Ordered Array

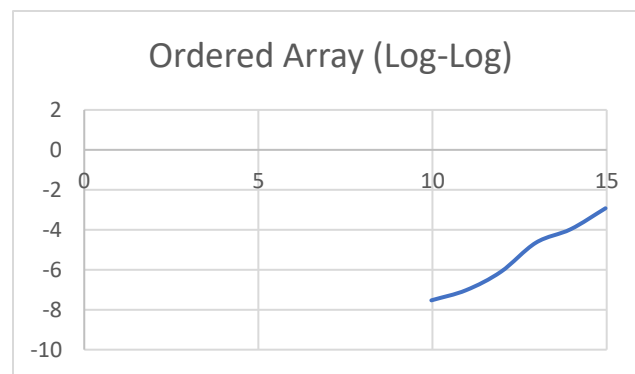
Various sizes of the Array and the running time of the Insertion sort

Ordered Array						
Array Size	Time	Ratio (Time/Previous Time)	lg(Array Size)	lg(Time)	Log Ratio	Slope
1000	0.0054		9.97	-7.53	-1.32	
2000	0.0077	1.43	10.97	-7.02	-1.56	0.51
4000	0.0146	1.90	11.97	-6.10	-1.96	0.92
8000	0.0399	2.73	12.97	-4.65	-2.79	1.45
16000	0.0637	1.60	13.97	-3.97	-3.52	0.67
32000	0.1318	2.07	14.97	-2.92	-5.12	1.05
Avg slope						0.92

Analysis of experimental data (the running time of insertion sort with random ordered input)



Standard Plot: Running time $T(n)$ Vs Array size N



Log-Log Plot: $\lg(T(n))$ Vs $\lg(N)$

The equation of the log-log plot is

$$\lg(T(N)) = 0.92 \lg N + \lg a$$

Which is equivalent to,

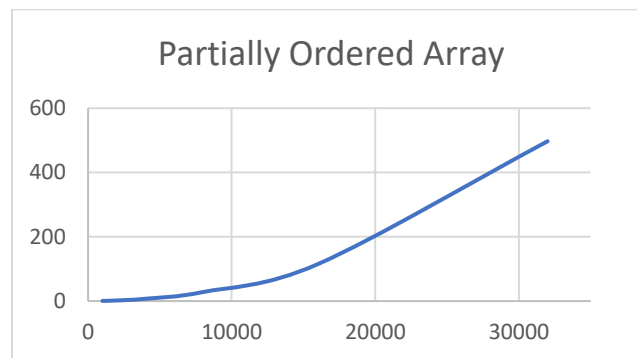
$$T(N) = aN^{0.92}$$

Partially Ordered Array

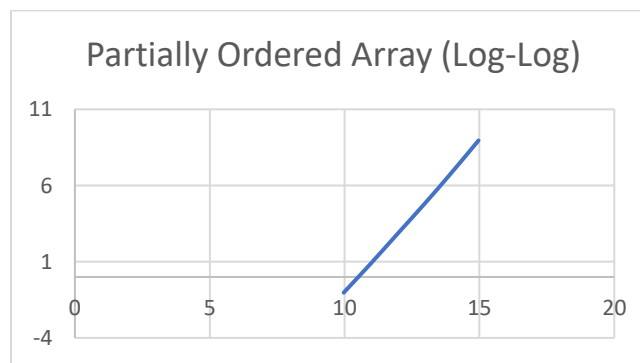
Various sizes of the Array and the running time of the Insertion sort

Partially Ordered Array						
Array Size	Time	Ratio (Time/Previous Time)	lg(Array Size)	lg(Time)	Log Ratio	Slope
1000	0.49		9.97	-1.03	-9.68	
2000	1.83	3.73	10.97	0.87	12.58	1.90
4000	7.16	3.91	11.97	2.84	4.21	1.97
8000	27.97	3.91	12.97	4.81	2.70	1.97
16000	114.85	4.11	13.97	6.84	2.04	2.04
32000	497.21	4.33	14.97	8.96	1.67	2.11
Avg slope						1.97

Analysis of experimental data (the running time of insertion sort with random ordered input)



Standard Plot: Running time T(n) Vs Array size N



Log-Log Plot: lg(T(n)) Vs lg(N)

The equation of the log-log plot is

$$\lg(T(N)) = 1.97 \lg N + \lg a$$

Which is equivalent to,

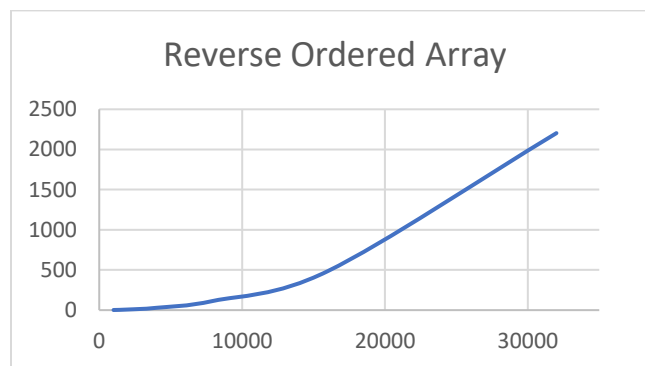
$$T(N) = aN^{1.97}$$

Reverse Ordered Array

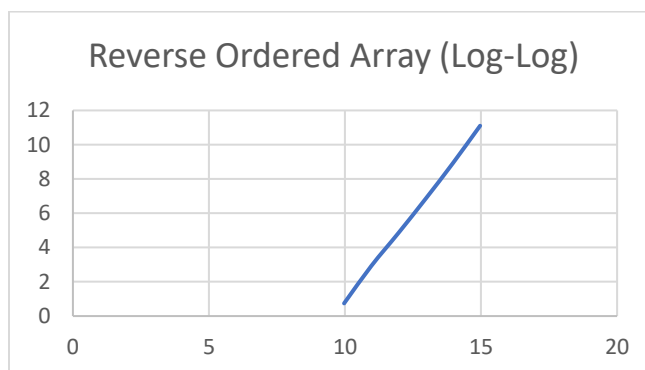
Various sizes of the Array and the running time of the Insertion sort

Reverse Ordered Array						
Array Size	Time	Ratio (Time/Previous Time)	lg(Array Size)	lg(Time)	Log Ratio	Slope
1000	1.67		9.97	0.74	13.47	
2000	7.58	4.54	10.97	2.92	3.75	2.18
4000	28.86	3.81	11.97	4.85	2.47	1.93
8000	115.39	4.00	12.97	6.85	1.89	2.00
16000	485.59	4.21	13.97	8.92	1.57	2.07
32000	2202.95	4.54	14.97	11.11	1.35	2.18
Avg slope						2.07

Analysis of experimental data (the running time of insertion sort with random ordered input)



Standard Plot: Running time T(n) Vs Array size N



Log-Log Plot: lg(T(n)) Vs lg(N)

The equation of the log-log plot is

$$\lg(T(N)) = 2.07 \lg N + \lg a$$

Which is equivalent to,

$$T(N) = aN^{2.07}$$

Output Screenshot

The first screenshot shows the output of a benchmark test for a randomly ordered array. The test runs Insertion Sort on arrays of sizes 1000, 2000, 4000, 8000, 16000, and 32000, each for 30 runs. The times increase significantly with array size. The second screenshot shows the output for a partially ordered array, where the times are much lower for the same array sizes. Both tests completed successfully in 1 second and 284 milliseconds.

```
INFO6205 > src > main > java > edu > neu > coe > info6205 > util > Benchmark_Timer > main
Project > life > .../
Run: Benchmark_Timer x
C:\Users\Jayanth\.jdk\openjdk-17.0.2\bin\java.exe ...
**Randomly Ordered Array**
2022-02-12 21:04:30 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 1000, Time: 1.84235
2022-02-12 21:04:30 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 2000, Time: 7.991303333333334
2022-02-12 21:04:31 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 4000, Time: 24.351119999999998
2022-02-12 21:04:32 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 8000, Time: 57.851236666666665
2022-02-12 21:04:33 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 16000, Time: 237.68453333333335
2022-02-12 21:04:41 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 32000, Time: 1171.62858

**Ordered Array**
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 1000, Time: 0.005353333333333333
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 2000, Time: 0.007703333333333334
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 4000, Time: 0.014633333333333333
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 8000, Time: 0.03994333333333333
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 16000, Time: 0.06371666666666667
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 32000, Time: 0.13181666666666667

Build completed successfully in 1 sec, 284 ms (13 minutes ago)
```

```
INFO6205 > src > main > java > edu > neu > coe > info6205 > util > Benchmark_Timer > main
Project > life > .../
Run: Benchmark_Timer x
**Partially Ordered Array**
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 1000, Time: 0.48889
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 2000, Time: 1.82751
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 4000, Time: 7.16458
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 8000, Time: 27.970056666666668
2022-02-12 21:05:21 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 16000, Time: 114.85384666666667
2022-02-12 21:05:25 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 32000, Time: 497.20865

**Reverse Ordered Array**
2022-02-12 21:05:42 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 1000, Time: 1.6678833333333332
2022-02-12 21:05:42 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 2000, Time: 7.575003333333333
2022-02-12 21:05:42 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 4000, Time: 28.861373333333333
2022-02-12 21:05:43 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 8000, Time: 115.39204
2022-02-12 21:05:47 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 16000, Time: 485.58698
2022-02-12 21:06:03 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 32000, Time: 2202.9538833333336

Process finished with exit code 0
Build completed successfully in 1 sec, 284 ms (14 minutes ago)
```

Output

****Randomly Ordered Array****

2022-02-12 21:04:30 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 1000, Time: 1.84235
2022-02-12 21:04:30 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 2000, Time: 7.9913033333333334
2022-02-12 21:04:31 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 4000, Time: 24.351119999999998
2022-02-12 21:04:32 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 8000, Time: 57.851236666666665
2022-02-12 21:04:33 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 16000, Time: 237.68453333333335
2022-02-12 21:04:41 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 32000, Time: 1171.62858

****Ordered Array****

2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 1000, Time: 0.0053533333333333333
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 2000, Time: 0.0077033333333333334
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 4000, Time: 0.014633333333333333
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 8000, Time: 0.039943333333333333
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 16000, Time: 0.06371666666666667
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 32000, Time: 0.13181666666666667

****Partially Ordered Array****

2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 1000, Time: 0.48889
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 2000, Time: 1.82751
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 4000, Time: 7.16458
2022-02-12 21:05:20 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 8000, Time: 27.970056666666668
2022-02-12 21:05:21 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 16000, Time: 114.85384666666667
2022-02-12 21:05:25 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
N= 32000, Time: 497.20865

****Reverse Ordered Array****

2022-02-12 21:05:42 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
 N= 1000, Time: 1.6678833333333332
 2022-02-12 21:05:42 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
 N= 2000, Time: 7.575003333333333
 2022-02-12 21:05:42 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
 N= 4000, Time: 28.861373333333333
 2022-02-12 21:05:43 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
 N= 8000, Time: 115.39204
 2022-02-12 21:05:47 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
 N= 16000, Time: 485.58698
 2022-02-12 21:06:03 INFO Benchmark_Timer - Begin run: Insertion Sort with 30 runs
 N= 32000, Time: 2202.9538833333336

Process finished with exit code 0

Unit Tests

- *TimerTest*

The screenshot shows an IDE with the following components:

- Project Explorer:** Displays a project structure with folders like 'greedy', 'hashtable', 'lab_1', 'life', 'pq', 'randomwalk', 'reduction', 'sort', 'symbolTable', 'threesum', 'union_find', and 'util'. The 'util' folder is expanded, showing classes like 'BenchmarkTest', 'ConfigTest', 'FastInverseSquareRootTest', 'LazyLoggerTest', 'PrivateMethodTester', 'PrivateMethodTesterTest', and 'SorterBenchmarkTest'.
- Code Editor:** Shows the source code of 'TimerTest.java'. The code includes package declarations, imports, and two methods: '@Before' 'setup()' and '@Test' 'testStop()'. The 'testStop()' method creates a 'Timer' object and calls 'GoToSleep(TENTH, which: 0);'.
- Run Console:** Displays the output of the test run. It shows 'Tests passed: 10 of 10 tests - 2 sec 816 ms'. Below this, a list of test methods and their execution times is shown:

Test Method	Time (ms)
testPauseAndLapResume0	282
testPauseAndLapResume1	337
testLap	219
testPause	226
testStop	113
testMillisecs	119
testRepeat1	189
testRepeat2	330
testRepeat3	881
testPauseAndLap	120
- Terminal:** Shows the command 'C:\Users\Jayanth\jdk\openjdk-17.0.2\bin\java.exe ...' and the message 'Process finished with exit code 0'.

• BenchmarkTest

The screenshot shows an IDE with the `BenchmarkTest` class open. The class is located in the package `edu.neu.coe.info6205.util`. It contains a `testWaitPeriods()` method that uses `Benchmark_Timer` to measure the execution time of a test. The test is named `testWaitPeriods` and has a description `"testWaitPeriods"`. The test is run with `nRuns = 2` and `warmups = 2`. The test is run with `GoToSleep(mSecs: 100L, which: -1)`.

The Run window shows the following output:

```

Tests passed: 2 of 2 tests - 1 sec 562 ms
BenchmarkTest (edu.neu.coe.info6205.util) 1 sec 562 ms
  testWaitPeriods 1 sec 562 ms
  getWarmupRuns 0 ms
2022-02-12 19:26:30 INFO Benchmark_Timer - Begin run: testWaitPeriods with 2 runs
Process finished with exit code 0

```

• InsertionSortTest

The screenshot shows an IDE with the `InsertionSortTest` class open. The class is located in the package `edu.neu.coe.info6205.sort.elementary`. It contains a `sort0()` method that sorts a list of integers. The test is named `sort0` and has a description `"sort0"`. The test is run with `nRuns = 2` and `warmups = 2`. The test is run with `GoToSleep(mSecs: 100L, which: -1)`.

The Run window shows the following output:

```

Tests passed: 6 of 6 tests - 199 ms
InsertionSortTest (edu.neu.coe.info6205.sort.elementary) 199 ms
  testMutatingInsertionSort 186 ms
  sort0 10 ms
  sort1 3 ms
  sort2 0 ms
  sort3 0 ms
  testStaticInsertionSort 0 ms
2022-02-12 19:50:35 DEBUG Config - Config.get(helper, instrument) = true
2022-02-12 19:50:35 DEBUG Config - Config.get(helper, seed) = 0
2022-02-12 19:50:35 DEBUG Config - Config.get(instrumenting, copies) = true
2022-02-12 19:50:35 DEBUG Config - Config.get(instrumenting, swaps) = true
2022-02-12 19:50:35 DEBUG Config - Config.get(instrumenting, compares) = true
2022-02-12 19:50:35 DEBUG Config - Config.get(instrumenting, inversions) = 1
2022-02-12 19:50:35 DEBUG Config - Config.get(instrumenting, fixes) = true
2022-02-12 19:50:35 DEBUG Config - Config.get(instrumenting, hits) = true
2022-02-12 19:50:35 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}

```