Kae-Yang Hsieh(001092745)

Program Structures & Algorithms

Fall 2021 Assignment No. 2

Tasks

* Observations of the insertion sort regarding the order of growth

Conclusion:

There are four different initial array ordering type: random, ordered, partially ordered, and reverse ordered and each array were be benchmarked with 10 different sized elements. With the removal of the first 3 smaller elements, the four benchmark experiments matched the expectation with the time complexity $O(n)$ of ordered array and $O(n^2)$ of the remaining types. Also, with the same elements, the order of time consumed by the four type were: ordered < partially ordered < random < reverse ordered, which is also matched the expectation.

Preparations

The four different initial arrays were created by the following methods:
* random: randomly shuffled an ordered array
* ordered: an array start from 1 to n
* partially ordered: random shuffled half of an ordered array, start from index 1 to n/2
* reverse ordered: an array start from n to 1

Each type of the array was initialized with 11 different lengths, followed the doubling method, where n starts from 50 to 51200

The ordered array where n starts from 500 to 512000
Also, the first 50 element is for warmup and will not be accounted into the experiments.

Experiments

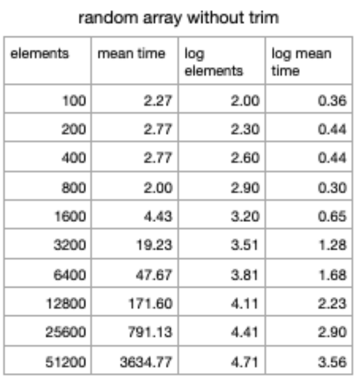
The experiments were implemented with a class called InsertionSortBenchmarks, located at “src/test/java/edu/neu/coe/info6205/util/InsertionSortBenchmarks.java”

All the arrays were set up using @before methods and timer.repeat to calculate the mean time of the sort.

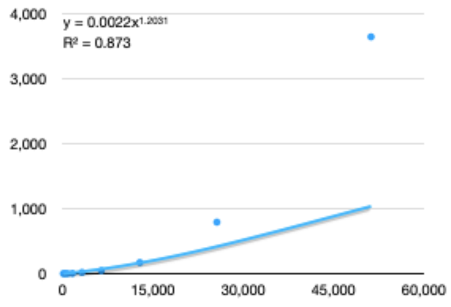
Benchmark Results and Analysis

Random array, partially ordered array, and reverse ordered array:
The average case (random and partially ordered) and the worst case (reverse ordered) of the insertion sort will be $O(n^2)$, which was expected to fit with the statistics. However, the equation showed a large difference with the expectation (Figure 1-1, 1-2, 1-3). With the removal of the first four abnormal statistics, the graph (Figure 2-1, 2-2, 2-3) is more close to the expectation: $y = n^2$

Through this experiments, the elements under 400 seems like has more deviation with the expectation. Here is a theory that may cause this situation: the cpu time contained the cache reading time and processing time, the cache reading will be almost the same while the data are smaller than the bandwidth. Also, the cache reading time may have a large proportion when measuring the sorting of little elements. This may affect a lot while we want to measure the processing time but not the cache reading time.


random array without trim

elements	mean time	log elements	log mean time
100	2.27	2.00	0.36
200	2.77	2.30	0.44
400	2.77	2.60	0.44
800	2.00	2.90	0.30
1600	4.43	3.20	0.65
3200	19.23	3.51	1.28
6400	47.67	3.81	1.68
12800	171.60	4.11	2.23
25600	791.13	4.41	2.90
51200	3634.77	4.71	3.56



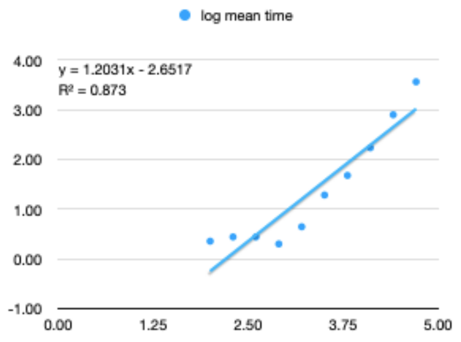
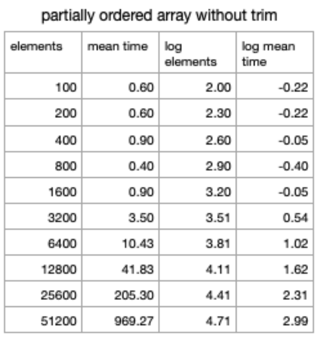
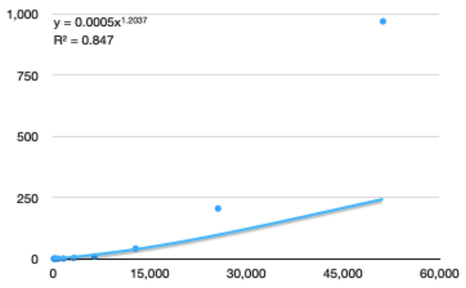


Figure 1-1 Random array without trimming


partially ordered array without trim

elements	mean time	log elements	log mean time
100	0.60	2.00	-0.22
200	0.60	2.30	-0.22
400	0.90	2.60	-0.05
800	0.40	2.90	-0.40
1600	0.90	3.20	-0.05
3200	3.50	3.51	0.54
6400	10.43	3.81	1.02
12800	41.83	4.11	1.62
25600	205.30	4.41	2.31
51200	969.27	4.71	2.99



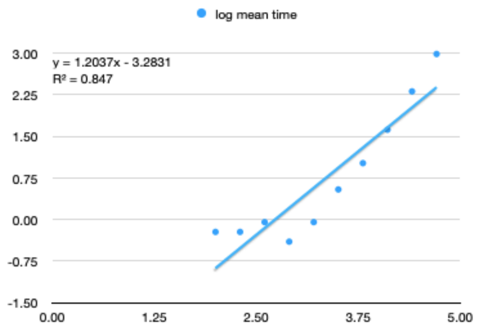
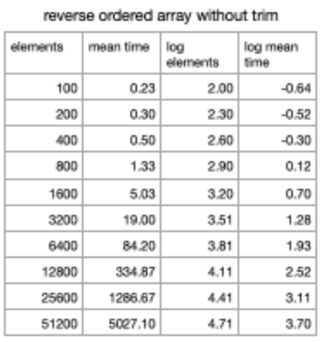
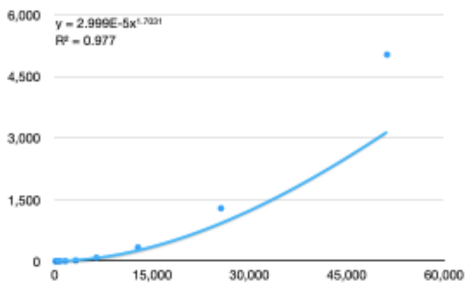


Figure 1-2 Partially ordered array without trimming


reverse ordered array without trim

elements	mean time	log elements	log mean time
100	0.23	2.00	-0.64
200	0.30	2.30	-0.52
400	0.50	2.60	-0.30
800	1.33	2.90	0.12
1600	5.03	3.20	0.70
3200	19.00	3.51	1.28
6400	84.20	3.81	1.93
12800	334.87	4.11	2.52
25600	1286.67	4.41	3.11
51200	5027.10	4.71	3.70



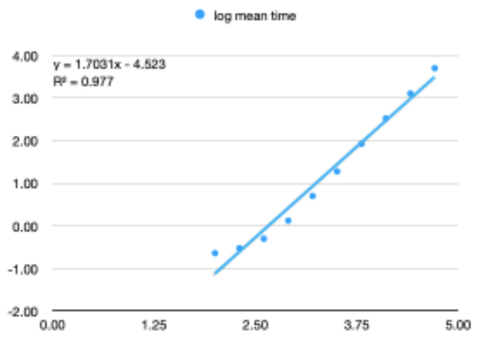


Figure 1-3 Reverse ordered array without trimming

random array without trim

elements	mean time	log elements	log mean time
800	2.00	2.90	0.30
1600	4.43	3.20	0.65
3200	19.23	3.51	1.28
6400	47.67	3.81	1.68
12800	171.60	4.11	2.23
25600	791.13	4.41	2.90
51200	3634.77	4.71	3.56

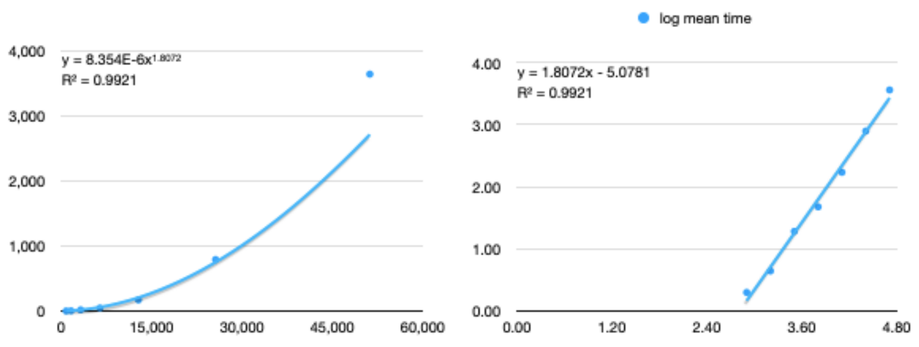


Figure 2-1 Random array with trimming

partially ordered array with trim

elements	mean time	log elements	log mean time
800	0.40	2.90	-0.40
1600	0.90	3.20	-0.05
3200	3.50	3.51	0.54
6400	10.43	3.81	1.02
12800	41.83	4.11	1.62
25600	205.30	4.41	2.31
51200	969.27	4.71	2.99

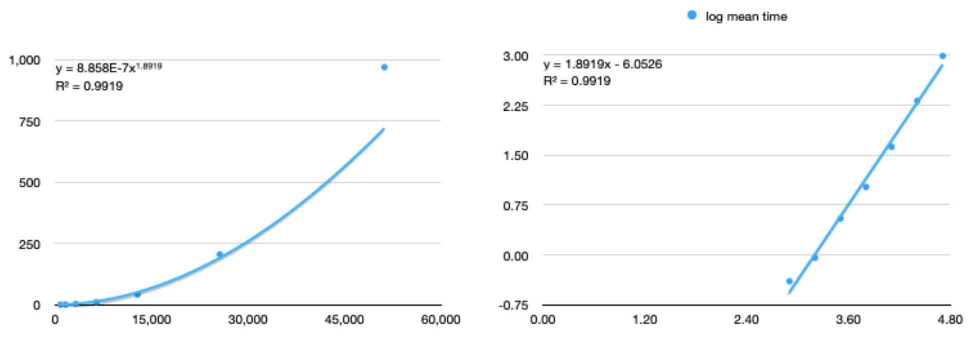


Figure 2-2 Partially ordered array with trimming

reverse ordered array without trim

elements	mean time	log elements	log mean time
800	1.33	2.90	0.12
1600	5.03	3.20	0.70
3200	19.00	3.51	1.28
6400	84.20	3.81	1.93
12800	334.87	4.11	2.52
25600	1286.67	4.41	3.11
51200	5027.10	4.71	3.70

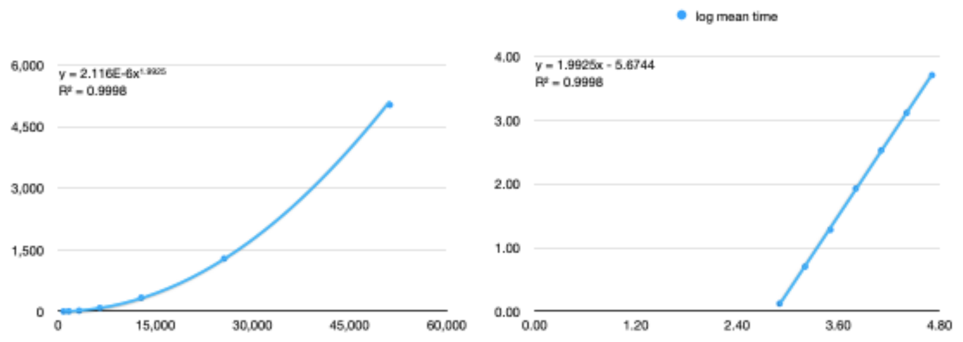


Figure 2-3 Reverse ordered array with trimming

Ordered array:

The ordered array matched the best case of the insertion sort, which has $O(n)$ time complexity. In the ordered array experiments, the original element number was too small to exclude the deviations. The statistic showed the original numbers has no linearity(Figure 3-1). With the ten times of the elements(Figure 3-2), the R^2 is near 1, which is expected to show a $O(n)$ time complexity.

ordered array

elements	mean time
100	1.23
200	1.03
400	1.27
700	0.77
1600	0.83
3200	0.43
6400	0.50
12800	0.33
25600	0.67
51200	1.20

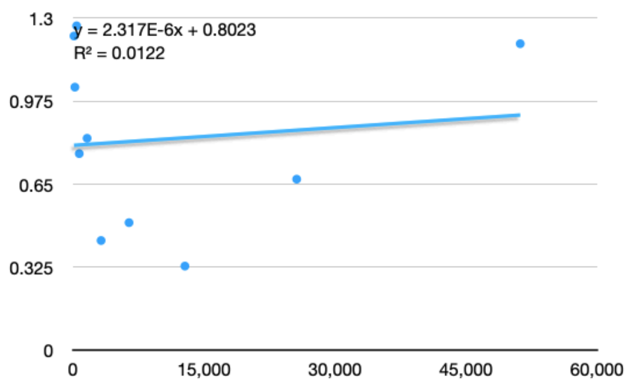


Figure 3-1 Ordered array with original numbers

ordered array

elements	mean time
1000	1.80
2000	2.00
4000	3.07
8000	2.00
16000	1.93
32000	2.40
64000	3.27
128000	5.50
256000	9.37
512000	16.10

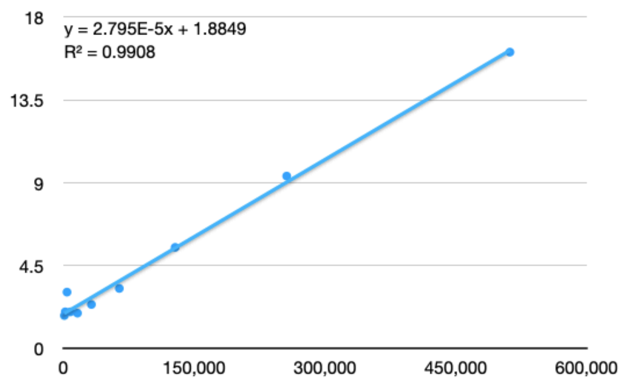
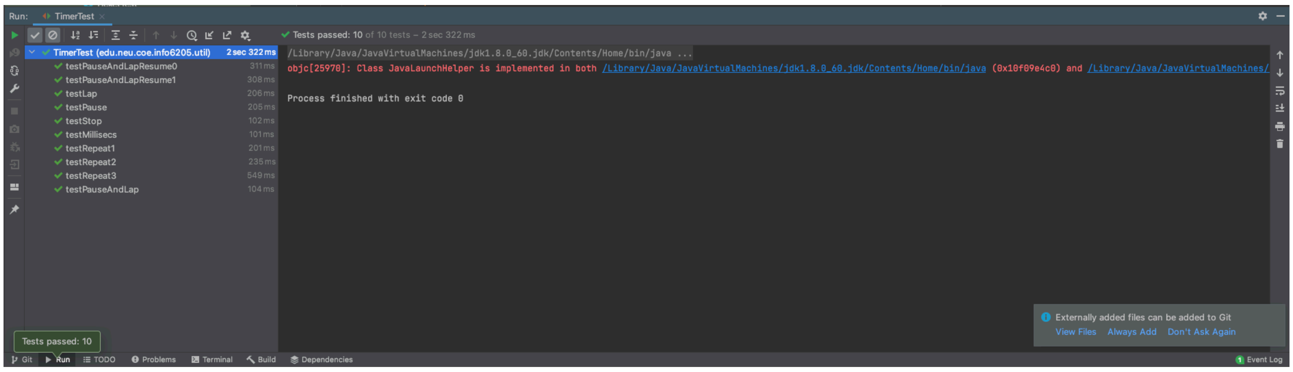


Figure 3-2 Ordered array with ten times numbers

* Unit test results

TimerTest

Note: The delta of the testRepeast1 has been adjusted from 6 to 13 to fit my environment.



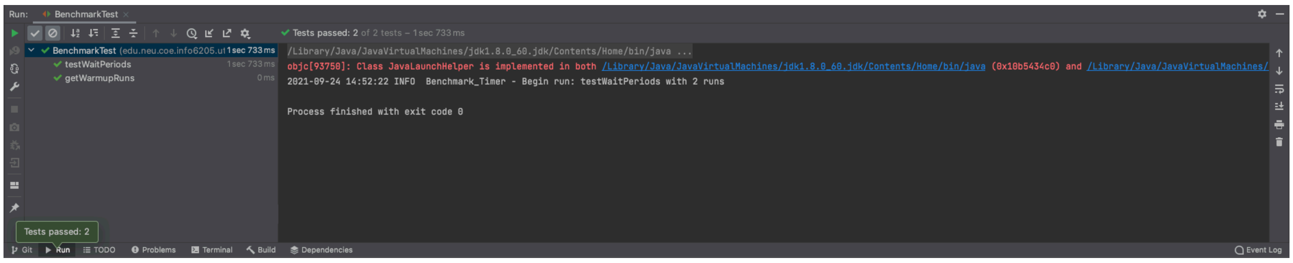
```
Tests passed: 10 of 10 tests – 2 sec 322 ms

TimerTest (edu.neu.coe.info6205.util)   2 sec 322 ms
    testPauseAndLapResume0              311 ms
    testPauseAndLapResume1              308 ms
    testLap                             206 ms
    testPause                           205 ms
    testStop                            102 ms
    testMillisecs                       101 ms
    testRepeat1                         201 ms
    testRepeat2                         235 ms
    testRepeat3                         549 ms
    testPauseAndLap                     104 ms

/Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java ...
objc[25970]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java (0x10f09e4c0) and /Library/Java/JavaVirtualMachines/

Process finished with exit code 0
```

BenchmarkTest



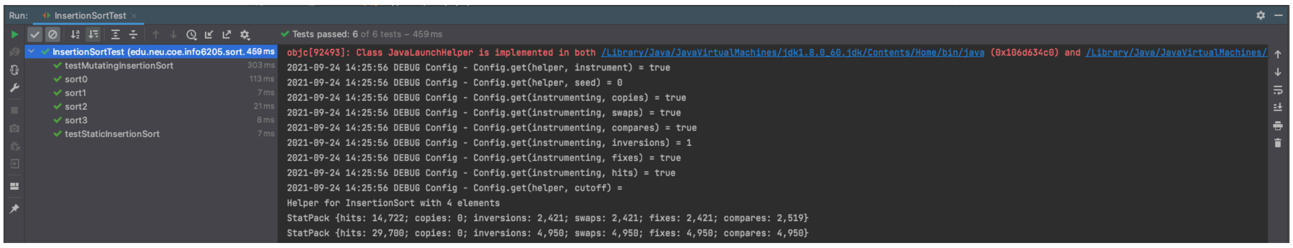
```
Tests passed: 2 of 2 tests – 1 sec 733 ms

BenchmarkTest (edu.neu.coe.info6205.ut 1 sec 733 ms
    testWaitPeriods                     1 sec 733 ms
    getWarmupRuns                       0 ms

/Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java ...
objc[93750]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java (0x10b5434c0) and /Library/Java/JavaVirtualMachines/
2021-09-24 14:52:22 INFO  Benchmark_Timer - Begin run: testWaitPeriods with 2 runs

Process finished with exit code 0
```

InsertionSortTest



```
Tests passed: 6 of 6 tests – 459 ms

InsertionSortTest (edu.neu.coe.info6205.sort. 459 ms
    testMutatingInsertionSort           303 ms
    sort0                               113 ms
    sort1                               7 ms
    sort2                               21 ms
    sort3                               8 ms
    testStaticInsertionSort             7 ms

objc[92493]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java (0x106d634c0) and /Library/Java/JavaVirtualMachines/
2021-09-24 14:25:56 DEBUG Config - Config.get(helper, instrument) = true
2021-09-24 14:25:56 DEBUG Config - Config.get(helper, seed) = 0
2021-09-24 14:25:56 DEBUG Config - Config.get(instrumenting, copies) = true
2021-09-24 14:25:56 DEBUG Config - Config.get(instrumenting, swaps) = true
2021-09-24 14:25:56 DEBUG Config - Config.get(instrumenting, compares) = true
2021-09-24 14:25:56 DEBUG Config - Config.get(instrumenting, inversions) = 1
2021-09-24 14:25:56 DEBUG Config - Config.get(instrumenting, fixes) = true
2021-09-24 14:25:56 DEBUG Config - Config.get(instrumenting, hits) = true
2021-09-24 14:25:56 DEBUG Config - Config.get(helper, cutoff) =
Helper for InsertionSort with 4 elements
StatPack {hits: 14,722; copies: 0; inversions: 2,421; swaps: 2,421; fixes: 2,421; compares: 2,519}
StatPack {hits: 29,700; copies: 0; inversions: 4,950; swaps: 4,950; fixes: 4,950; compares: 4,950}
```