

# INFO6205 Assignment 3 (Benchmark)

NAME: Bohan Feng

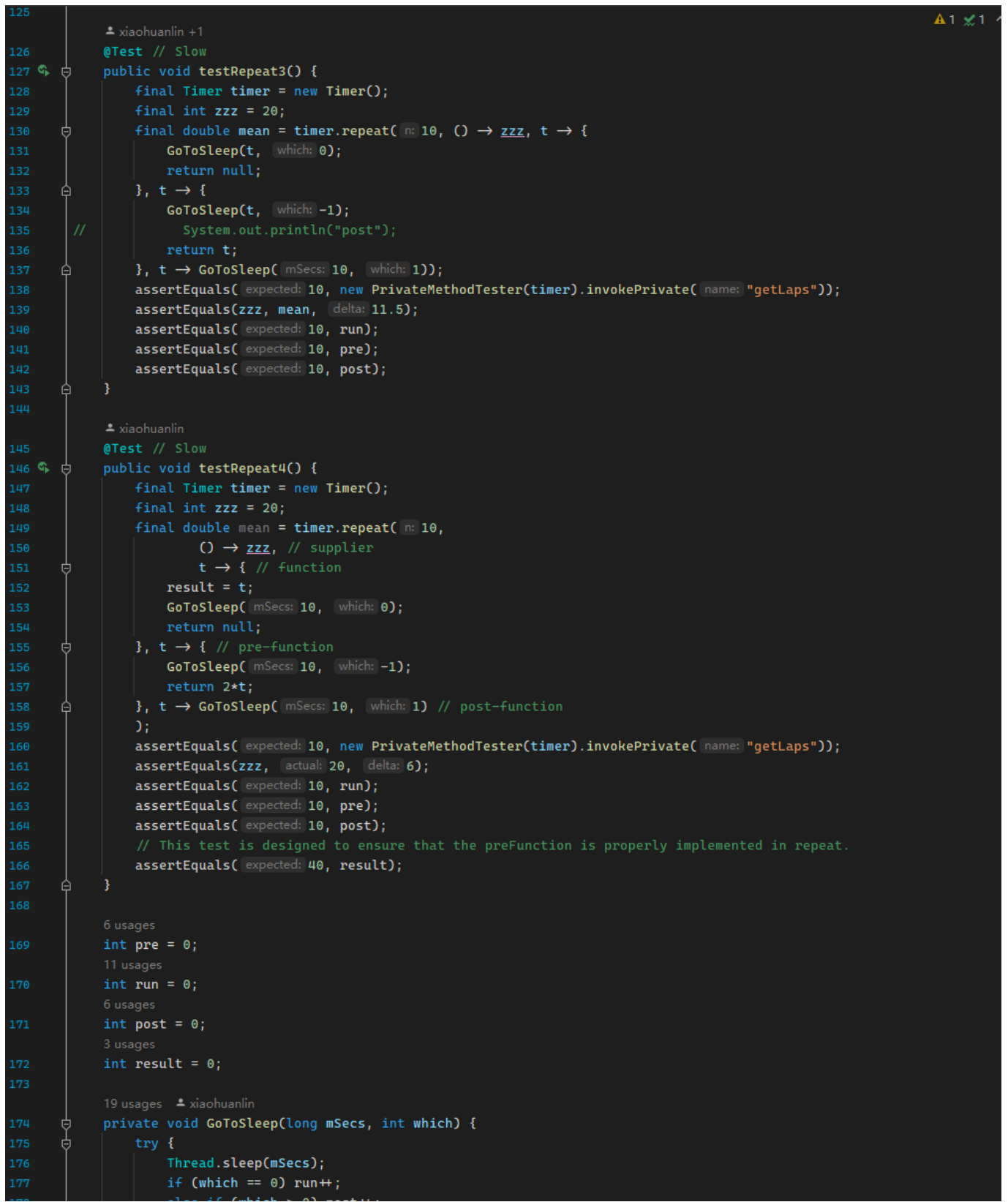
NUID: 001564249

Repository: <https://github.com/fengb3/INFO6205>

## Part 1 Timer & Benchmark

### Unit Test Screenshot

Timer:



```
125
126  xiaohuanlin +1
127  @Test // Slow
128  public void testRepeat3() {
129      final Timer timer = new Timer();
130      final int zzz = 20;
131      final double mean = timer.repeat(n: 10, () -> zzz, t -> {
132          GoToSleep(t, which: 0);
133          return null;
134      }, t -> {
135          GoToSleep(t, which: -1);
136          // System.out.println("post");
137          return t;
138      }, t -> GoToSleep(mSecs: 10, which: 1));
139      assertEquals(expected: 10, new PrivateMethodTester(timer).invokePrivate(name: "getLaps"));
140      assertEquals(zzz, mean, delta: 11.5);
141      assertEquals(expected: 10, run);
142      assertEquals(expected: 10, pre);
143      assertEquals(expected: 10, post);
144  }
145
146  xiaohuanlin
147  @Test // Slow
148  public void testRepeat4() {
149      final Timer timer = new Timer();
150      final int zzz = 20;
151      final double mean = timer.repeat(n: 10,
152          () -> zzz, // supplier
153          t -> { // function
154              result = t;
155              GoToSleep(mSecs: 10, which: 0);
156              return null;
157          }, t -> { // pre-function
158              GoToSleep(mSecs: 10, which: -1);
159              return 2*t;
160          }, t -> GoToSleep(mSecs: 10, which: 1) // post-function
161      );
162      assertEquals(expected: 10, new PrivateMethodTester(timer).invokePrivate(name: "getLaps"));
163      assertEquals(zzz, actual: 20, delta: 6);
164      assertEquals(expected: 10, run);
165      assertEquals(expected: 10, pre);
166      assertEquals(expected: 10, post);
167      // This test is designed to ensure that the preFunction is properly implemented in repeat.
168      assertEquals(expected: 40, result);
169  }
170
171  6 usages
172  int pre = 0;
173  11 usages
174  int run = 0;
175  6 usages
176  int post = 0;
177  3 usages
178  int result = 0;
179
180  19 usages xiaohuanlin
181  private void GoToSleep(long mSecs, int which) {
182      try {
183          Thread.sleep(mSecs);
184          if (which == 0) run++;
185          else if (which > 0) post++;
186      } catch (InterruptedException e) {
187          // ...
188      }
189  }
```

```

178         else if (which > 0) post++;
179         else pre++;
180     } catch (InterruptedException e) {
181         e.printStackTrace();
182     }
183 }
184

```

11 usages

Run: TimerTest X

Tests passed: 11 of 11 tests - 3 sec 397 ms

Test Name	Duration
testPauseAndLapResume0	608 ms
testPauseAndLapResume1	324 ms
testLap	217 ms
testPause	213 ms
testStop	104 ms
testMillisecs	108 ms
testRepeat1	154 ms
testRepeat2	309 ms
testRepeat3	773 ms
testRepeat4	477 ms
testPauseAndLap	110 ms

Process finished with exit code 0

```

17
18  xiaohuanlin
19  @Test
20  public void testStop() {
21      final Timer timer = new Timer();
22      GoToSleep(TENTH, which: 0);
23      final double time = timer.stop();
24      assertEquals(TENTH_DOUBLE, time, delta: 10);
25      assertEquals(expected: 1, run);
26      assertEquals(expected: 1, new PrivateMethodTester(timer).invokePrivate(name: "getLaps"));
27  }
28
29  xiaohuanlin
30  @Test
31  public void testPauseAndLap() {
32      final Timer timer = new Timer();
33      final PrivateMethodTester privateMethodTester = new PrivateMethodTester(timer);
34      GoToSleep(TENTH, which: 0);
35      timer.pauseAndLap();
36      final Long ticks = (Long) privateMethodTester.invokePrivate(name: "getTicks");
37      assertEquals(TENTH_DOUBLE, actual: ticks / 1e6, delta: 12);
38      assertFalse((Boolean) privateMethodTester.invokePrivate(name: "isRunning"));
39      assertEquals(expected: 1, privateMethodTester.invokePrivate(name: "getLaps"));
40  }
41
42  xiaohuanlin
43  @Test
44  public void testPauseAndLapResume0() {
45      final Timer timer = new Timer();
46      final PrivateMethodTester privateMethodTester = new PrivateMethodTester(timer);
47      GoToSleep(TENTH, which: 0);
48      timer.pauseAndLap();
49      timer.resume();
50      assertTrue((Boolean) privateMethodTester.invokePrivate(name: "isRunning"));
51      assertEquals(expected: 1, privateMethodTester.invokePrivate(name: "getLaps"));
52  }
53
54  xiaohuanlin
55  @Test
56  public void testPauseAndLapResume1() {
57      final Timer timer = new Timer();
58      GoToSleep(TENTH, which: 0);
59      timer.pauseAndLap();
60      GoToSleep(TENTH, which: 0);
61      timer.resume();
62      GoToSleep(TENTH, which: 0);
63      final double time = timer.stop();
64      assertEquals(TENTH_DOUBLE, time, delta: 10.0);
65      assertEquals(expected: 3, run);
66  }
67
68  xiaohuanlin
69  @Test
70  public void testLap() {
71      final Timer timer = new Timer();
72      GoToSleep(TENTH, which: 0);
73      timer.lap();

```

```

69         GoToSleep(TENTH, which: 0);
70         final double time = timer.stop();
71         assertEquals(TENTH_DOUBLE, time, delta: 10.0);
72         assertEquals(expected: 2, run);
73     }
74
75     xiaohuanlin
76     @Test
77     public void testPause() {
78         final Timer timer = new Timer();
79         GoToSleep(TENTH, which: 0);

```

Run: TimerTest X

Tests passed: 11 of 11 tests – 3 sec 397 ms

Test Name	Duration
testPauseAndLapResume0	608 ms
testPauseAndLapResume1	324 ms
testLap	217 ms
testPause	213 ms
testStop	104 ms
testMillisecs	108 ms
testRepeat1	154 ms
testRepeat2	309 ms
testRepeat3	773 ms
testRepeat4	477 ms
testPauseAndLap	110 ms

Process finished with exit code 0

```

75     xiaohuanlin
76     @Test
77     public void testPause() {
78         final Timer timer = new Timer();
79         GoToSleep(TENTH, which: 0);
80         timer.pause();
81         GoToSleep(TENTH, which: 0);
82         timer.resume();
83         final double time = timer.stop();
84         assertEquals(TENTH_DOUBLE, time, delta: 10.0);
85         assertEquals(expected: 2, run);
86     }
87
88     xiaohuanlin
89     @Test
90     public void testMillisecs() {
91         final Timer timer = new Timer();
92         GoToSleep(TENTH, which: 0);
93         timer.stop();
94         final double time = timer.millisecs();
95         assertEquals(TENTH_DOUBLE, time, delta: 10.0);
96         assertEquals(expected: 1, run);
97     }
98
99     xiaohuanlin
100    @Test
101    public void testRepeat1() {
102        final Timer timer = new Timer();
103        final double mean = timer.repeat(n: 10, () → {
104            GoToSleep(HUNDREDTH, which: 0);
105            return null;
106        });
107        assertEquals(expected: 10, new PrivateMethodTester(timer).invokePrivate(name: "getLaps"));
108        assertEquals(expected: TENTH_DOUBLE / 10, mean, delta: 6);
109        assertEquals(expected: 10, run);
110        assertEquals(expected: 0, pre);
111        assertEquals(expected: 0, post);
112    }
113
114    xiaohuanlin +1
115    @Test
116    public void testRepeat2() {
117        final Timer timer = new Timer();
118        final int zzz = 20;
119        final double mean = timer.repeat(n: 10, () → zzz, t → {
120            GoToSleep(t, which: 0);
121            return null;
122        });
123        assertEquals(expected: 10, new PrivateMethodTester(timer).invokePrivate(name: "getLaps"));
124        assertEquals(zzz, mean, delta: 11);
125        assertEquals(expected: 10, run);
126        assertEquals(expected: 0, pre);
127        assertEquals(expected: 0, post);

```

```
124 }
125
126 @Test // Slow
127 public void testRepeat3() {
128     final Timer timer = new Timer();
129     final int zzz = 20;
130     final double mean = timer.repeat(n: 10, () -> zzz, t -> {
131         GoToSleep(t, which: 0);
132         return null;
133     }, t -> {
134         GoToSleep(t, which: -1);
135         System.out.println("post");
136         return t;
137     });
138     t -> GoToSleep(mSecs: 10, which: 1);
139 }
140
```

Run: TimerTest x

Tests passed: 11 of 11 tests - 3 sec 397 ms

Test	Duration
testPauseAndLapResume0	608 ms
testPauseAndLapResume1	324 ms
testLap	217 ms
testPause	213 ms
testStop	104 ms
testMillisecs	108 ms
testRepeat1	154 ms
testRepeat2	309 ms
testRepeat3	773 ms
testRepeat4	477 ms
testPauseAndLap	110 ms

Process finished with exit code 0

Benchmark:

```
1 // ...
2
3
4 package edu.neu.coe.info6205.util;
5
6
7 import ...
8
9
10
11 /ALL/
12 public class BenchmarkTest {
13
14     2 usages
15     int pre = 0;
16     2 usages
17     int run = 0;
18     2 usages
19     int post = 0;
20
21     xiaohuanlin
22     @Test // Slow
23     public void testWaitPeriods() throws Exception {
24         int nRuns = 2;
25         int warmups = 2;
26         Benchmark<Boolean> bm = new Benchmark_Timer<>(
27             description: "testWaitPeriods", b -> {
28                 GoToSleep(mSecs: 100L, which: -1);
29                 return null;
30             },
31             b -> {
32                 GoToSleep(mSecs: 200L, which: 0);
33             },
34             b -> {
35                 GoToSleep(mSecs: 50L, which: 1);
36             }
37         );
38         double x = bm.run(t: true, nRuns);
39         assertEquals(nRuns, post);
40         assertEquals(expected: nRuns + warmups, run);
41         assertEquals(expected: nRuns + warmups, pre);
42         assertEquals(expected: 200, x, delta: 10);
43     }
44
45     3 usages xiaohuanlin
46     private void GoToSleep(long mSecs, int which) {
47         try {
48             Thread.sleep(mSecs);
49             if (which == 0) run++;
50             else if (which > 0) post++;
51             else pre++;
52         }
53     }
54 }
```

```
46 } catch (InterruptedException e) {
47     e.printStackTrace();
48 }
49 }
50
51 xiaohuanlin
52 @Test
53 public void getWarmupRuns() {
54     assertEquals(expected: 2, Benchmark_Timer.getWarmupRuns(m: 0));
55     assertEquals(expected: 2, Benchmark_Timer.getWarmupRuns(m: 20));
56     assertEquals(expected: 3, Benchmark_Timer.getWarmupRuns(m: 30));
57     assertEquals(expected: 10, Benchmark_Timer.getWarmupRuns(m: 100));
58     assertEquals(expected: 10, Benchmark_Timer.getWarmupRuns(m: 1000));
59 }
}
```

Run: BenchmarkTest x

Tests passed: 2 of 2 tests - 1 sec 462 ms

Test Name	Duration	Output
BenchmarkTest (edu.neu.coe.info6205.util)	1 sec 462 ms	C:\Users\冯博藻\.jdk\openjdk-18.0.2.1\bin\java.exe ...
testWaitPeriods	1 sec 462 ms	2022-10-18 12:37:11 INFO Benchmark_Timer - Begin run: testWaitPeriods
getWarmupRuns	0 ms	Process finished with exit code 0

## Part 2 Insertion Sort

Unit Test Screenshot:

```
21 xiaohuanlin
22 @Test
23 public void sort0() throws Exception {
24     final List<Integer> list = new ArrayList<>();
25     list.add(1);
26     list.add(2);
27     list.add(3);
28     list.add(4);
29     Integer[] xs = list.toArray(new Integer[0]);
30     final Config config = ConfigTest.setupConfig(instrumenting: "true", seed: "0", inversions: "1", cutoff: "", interimInversion: "1");
31     Helper<Integer> helper = HelperFactory.create(description: "InsertionSort", list.size(), config);
32     helper.init(list.size());
33     final PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
34     final StatPack statPack = (StatPack) privateMethodTester.invokePrivate(name: "getStatPack");
35     SortWithHelper<Integer> sorter = new InsertionSort<>(helper);
36     sorter.preProcess(xs);
37     Integer[] ys = sorter.sort(xs);
38     assertTrue(helper.sorted(ys));
39     sorter.postProcess(ys);
40     final int compares = (int) statPack.getStatistics(InstrumentedHelper.COMPARES).mean();
41     assertEquals(expected: list.size() - 1, compares);
42     final int inversions = (int) statPack.getStatistics(InstrumentedHelper.INVERSIONS).mean();
43     assertEquals(expected: 0L, inversions);
44     final int fixes = (int) statPack.getStatistics(InstrumentedHelper.FIXES).mean();
45     assertEquals(inversions, fixes);
46 }
47
48 xiaohuanlin
49 @Test
50 public void sort1() throws Exception {
51     final List<Integer> list = new ArrayList<>();
52     list.add(3);
53     list.add(4);
54     list.add(2);
55     list.add(1);
56     Integer[] xs = list.toArray(new Integer[0]);
57     BaseHelper<Integer> helper = new BaseHelper<>(description: "InsertionSort", xs.length, Config.load(InsertionSortTest::config));
58     helper.init(xs.length);
59     final PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
60     final StatPack statPack = (StatPack) privateMethodTester.invokePrivate(name: "getStatPack");
61     SortWithHelper<Integer> sorter = new InsertionSort<>(helper);
62     sorter.preProcess(xs);
63     Integer[] ys = sorter.sort(xs);
64     assertTrue(helper.sorted(ys));
65     sorter.postProcess(ys);
66     final int compares = (int) statPack.getStatistics(InstrumentedHelper.COMPARES).mean();
67     assertEquals(expected: list.size() - 1, compares);
68     final int inversions = (int) statPack.getStatistics(InstrumentedHelper.INVERSIONS).mean();
69     assertEquals(expected: 0L, inversions);
70     final int fixes = (int) statPack.getStatistics(InstrumentedHelper.FIXES).mean();
71     assertEquals(inversions, fixes);
72 }
}
```

```

56         GenericSort<Integer> sorter = new InsertionSort<>>(helper);
57         Integer[] ys = sorter.sort(xs);
58         assertTrue(helper.sorted(ys));
59         System.out.println(sorter.toString());
60     }
61
62     @Test
63     public void testMutatingInsertionSort() throws IOException {
64         final List<Integer> list = new ArrayList<>();
65         list.add(3);
66         list.add(4);
67         list.add(2);
68         list.add(1);
69         Integer[] xs = list.toArray(new Integer[0]);
70         BaseHelper<Integer> helper = new BaseHelper<>(description: "InsertionSort", xs.length, Config.load(InsertionSortTest
71         GenericSort<Integer> sorter = new InsertionSort<>>(helper);
72         sorter.mutatingSort(xs);
73         assertTrue(helper.sorted(xs));
74     }
75
76     @Test
77     public void testStaticInsertionSort() throws IOException {
78         final List<Integer> list = new ArrayList<>();
79         list.add(3);
80         list.add(4);
81         list.add(2);
82         list.add(1);
83         Integer[] xs = list.toArray(new Integer[0]);
84     }

```

Run: InsertionSortTest ×

✓ Tests passed: 6 of 6 tests – 150 ms

Test Name	Duration	Output
InsertionSortTest (edu.neu.coe.info6205.sortelementa)	150 ms	C:\Users\冯博藻\.jdk\openjdk-18.0.2.1\bin\java.exe ...
testMutatingInsertionSort	107 ms	2022-10-18 11:46:14 DEBUG Config - Config.get(helper, instrument) = tr
sort0	33 ms	2022-10-18 11:46:14 DEBUG Config - Config.get(helper, seed) = 0
sort1	1 ms	2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, copies) =
sort2	5 ms	2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, swaps) =
sort3	2 ms	2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, compares) =
testStaticInsertionSort	2 ms	2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, inversion
		2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, fixes) =
		2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, hits) = t
		2022-10-18 11:46:14 DEBUG Config - Config.get(helper, cutoff) =
		Helper for InsertionSort with 4 elements
		StatPack {hits: 9,880, normalized=21.454; copies: 0, normalized=0.000;
		StatPack {hits: 19,800, normalized=42.995; copies: 0, normalized=0.000
		Process finished with exit code 0

```

82         list.add(1);
83         Integer[] xs = list.toArray(new Integer[0]);
84         InsertionSort.sort(xs);
85         assertTrue(condition: xs[0] < xs[1] && xs[1] < xs[2] && xs[2] < xs[3]);
86     }
87
88     @Test
89     public void sort2() throws Exception {
90         final Config config = ConfigTest.setupConfig(instrumenting: "true", seed: "0", inversions: "1", cutoff: "", interimInversion
91         int n = 100;
92         Helper<Integer> helper = HelperFactory.create(description: "InsertionSort", n, config);
93         helper.init(n);
94         final PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
95         final StatPack statPack = (StatPack) privateMethodTester.invokePrivate(name: "getStatPack");
96         Integer[] xs = helper.random(Integer.class, r -> r.nextInt(bound: 1000));
97         SortWithHelper<Integer> sorter = new InsertionSort<Integer>(helper);
98         sorter.preProcess(xs);
99         Integer[] ys = sorter.sort(xs);
100        assertTrue(helper.sorted(ys));
101        sorter.postProcess(ys);
102        final int compares = (int) statPack.getStatistics(InstrumentedHelper.COMPARES).mean();
103        // NOTE: these are supposed to match within about 12%.
104        // Since we set a specific seed, this should always succeed.
105        // If we use true random seed and this test fails, just increase the delta a little.
106        assertEquals(expected: 1.0, actual: 4.0 * compares / n / (n - 1), delta: 0.12);
107        final int inversions = (int) statPack.getStatistics(InstrumentedHelper.INVERSIONS).mean();
108        final int fixes = (int) statPack.getStatistics(InstrumentedHelper.FIXES).mean();
109        System.out.println(statPack);
110        assertEquals(inversions, fixes);
111    }
112
113    @Test
114    public void sort3() throws Exception {
115        final Config config = ConfigTest.setupConfig(instrumenting: "true", seed: "0", inversions: "1", cutoff: "", interimInversion

```



```
116     int n = 100;
117     Helper<Integer> helper = HelperFactory.create( description: "InsertionSort", n, config);
118     helper.init(n);
119     final PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
120     final StatPack statPack = (StatPack) privateMethodTester.invokePrivate( name: "getStatPack");
121     Integer[] xs = new Integer[n];
122     for (int i = 0; i < n; i++) xs[i] = n - i;
123     SortWithHelper<Integer> sorter = new InsertionSort<>(helper);
124     sorter.preProcess(xs);
125     Integer[] ys = sorter.sort(xs);
126     assertTrue(helper.sorted(ys));
127     sorter.postProcess(ys);
128     final int compares = (int) statPack.getStatistics(InstrumentedHelper.COMPARES).mean();
129     // NOTE: these are supposed to match within about 12%.
130     // Since we set a specific seed, this should always succeed.
131     // If we use true random seed and this test fails, just increase the delta a little.
132     assertEquals( expected: 4950, compares);
133     final int inversions = (int) statPack.getStatistics(InstrumentedHelper.INVERSIONS).mean();
134     final int fixes = (int) statPack.getStatistics(InstrumentedHelper.FIXES).mean();
135     System.out.println(statPack);
136     assertEquals(inversions, fixes);
137 }
138
139 final static LazyLogger logger = new LazyLogger(InsertionSort.class);
140
141 }
```

Run: InsertionSortTest ×

✓ Tests passed: 6 of 6 tests – 150 ms

InsertionSortTest (edu.neu.coe.info6205.sort.elementa 150 ms)

- ✓ testMutatingInsertionSort 107 ms
- ✓ sort0 33 ms
- ✓ sort1 1 ms
- ✓ sort2 5 ms
- ✓ sort3 2 ms
- ✓ testStaticInsertionSort 2 ms

C:\Users\冯博藻\.jdk\openjdk-18.0.2.1\bin\java.exe ...

2022-10-18 11:46:14 DEBUG Config - Config.get(helper, instrument) = tr  
2022-10-18 11:46:14 DEBUG Config - Config.get(helper, seed) = 0  
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, copies) =  
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, swaps) =  
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, compares) =  
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, inversion  
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, fixes) =  
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, hits) = t  
2022-10-18 11:46:14 DEBUG Config - Config.get(helper, cutoff) =  
Helper for InsertionSort with 4 elements  
StatPack {hits: 9,880, normalized=21.454; copies: 0, normalized=0.000;  
StatPack {hits: 19,800, normalized=42.995; copies: 0, normalized=0.000  
Process finished with exit code 0

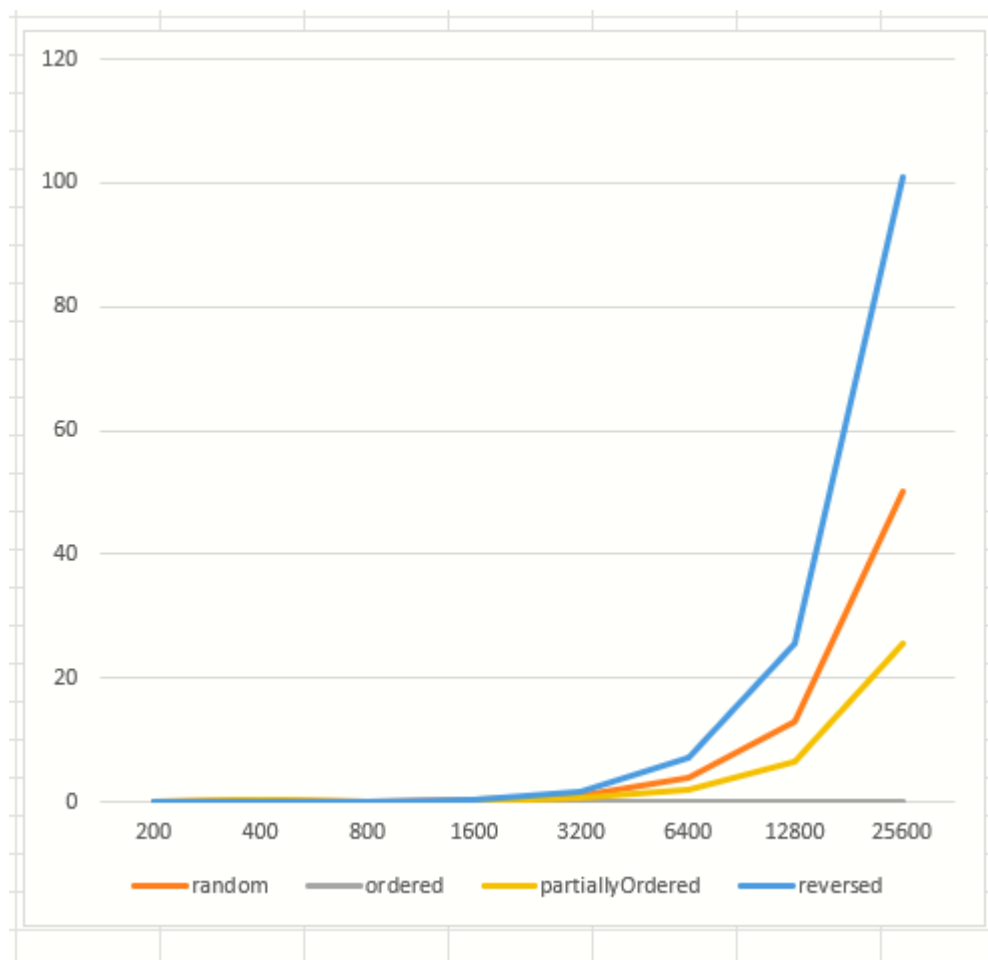
## Part 3 Measure Running Times

### Measured Data

Table:

n	random	ordered	partiallyOrdered	reversed
200	0.0374	0.00516	0.01614	0.04288
400	0.32443	0.00359	0.19526	0.05365
800	0.10849	0.0065	0.03351	0.11643
1600	0.25289	0.00811	0.12406	0.45562
3200	0.93144	0.01623	0.56324	1.79383
6400	3.84342	0.03391	1.83933	7.2329
12800	12.86309	0.05785	6.54417	25.50461
25600	50.28089	0.0905	25.56846	100.8463

Graph:



## Obervation

I used doubling method to test the time usage of InsertionSort. For each different array length  $(n)$ , I tried four array sorting situations (random, ordered, inverted, and partially ordered). Based on the output data and charts, we can briefly know that the speed of sorting ordered array is the fastest, sorting random and partially ordered array is slower, and sorting for reversed is the slowest.

For best situation (ordered), InsertionSort does not require a swap operation, it needs  $(n - 1)$  times comparison. For worst situation (reversed), InsertionSort needs  $n * (n - 1) / 2$  times comparison, because when we insert  $n$ th element, we need to compare previous  $(n - 1)$  elements. InsertionSort's swap times is number of comparison operations minus  $(n - 1)$ . On the average, the time complexity of the InsertionSort is  $O(n^2)$ .