

## Program Structures and Algorithms – Assignment 3

Spring 2023 (SEC – 3)

**NAME:** Sharun Kumar Kakkad Sasikumar

**NUID:** 002774079

### Task:

To implement the timer functions, insertion sort and benchmark the insertion sort on various type of array inputs.

### Conclusion for growth based on 4 types of inputs:

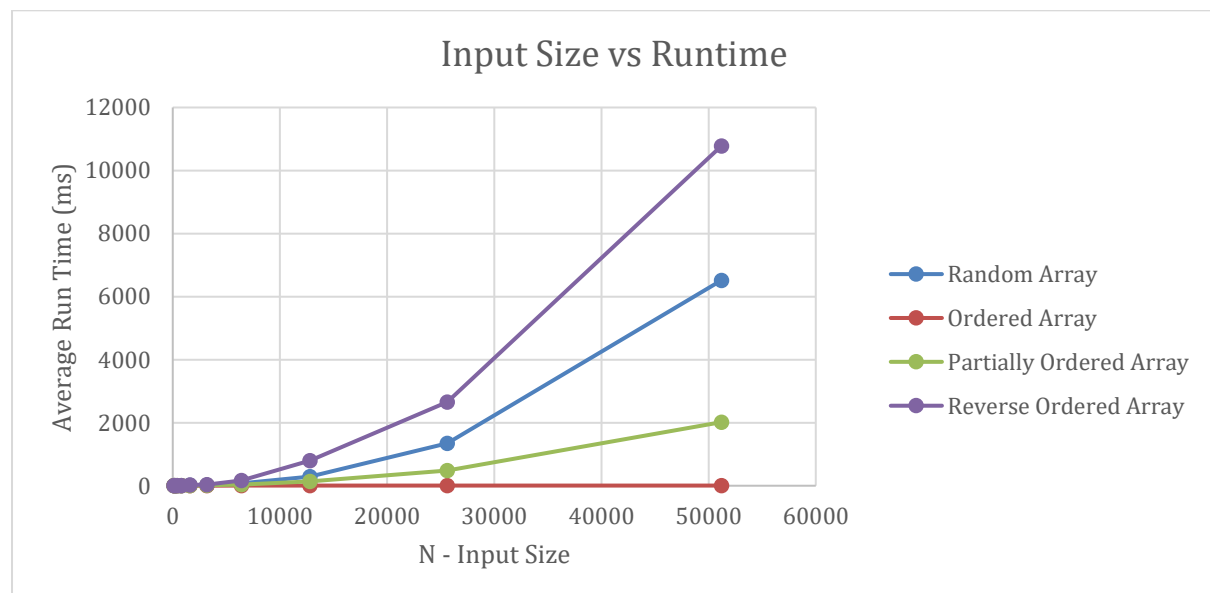
- **Random array (average case):** The elements are randomly placed, so on average it will take  $O(n^2)$  time for the algorithm to sort the array.
- **Ordered array (best case):** The elements are already in the correct order, so the algorithm will perform the best in this case, with a time complexity of  $O(n)$ .
- **Partially ordered array:** The elements are partially sorted, which means that some elements are in the correct order, while others are not. In this case, the time complexity will be between  $O(n)$  and  $O(n^2)$  depending on how many elements are already in the correct order.
- **Reverse-ordered array (worst case):** The elements are in the opposite order of what they should be. In this case, the algorithm will take the longest to sort the array, with a time complexity of  $O(n^2)$  since it will have to swap every element in the array multiple times to sort it.

### Code:

Submitted to GitHub Repository: <https://github.com/sharunkumar-ks/INFO6205/pull/2/files>

### Graphical Representation:

Complete data is available in the **Insertion Sort Benchmark.xlsx** file.



## Unit Test Screenshots:

**Note:** I have run the unit tests in GitHub Codespaces environment as my personal laptop is a bit slow

Codespaces Link: <https://sharunkumar-ks-potential-space-palm-tree-5jj5pprrggphp6vr.github.dev/>

The screenshot displays a GitHub Codespaces environment with three main panes. The left pane shows a list of tests under the heading "TESTING". The middle pane shows the source code for `BenchmarkTest.java` and `TimerTest.java`. The right pane shows the test results for `TimerTest.java`.

**TESTING**

- 0/0 tests passed (0.00%)
- INFO6205 4.1s
- edu.neu.coe.info6205.util 4.1s
- BenchmarkTest 1.6s
  - testWaitPeriods() 1.6s
  - getWarmupRuns() 0.0ms
  - TimerTest 2.5s
    - testRepeat4() 306ms
    - testPauseAndResume() 102ms
    - testPauseAndResume() 326ms
    - testPauseAndResume() 300ms
    - testPauseAndResume() 200ms
    - testPauseAndResume() 18ms
    - testRepeat10 104ms
    - testRepeat20 204ms
    - testRepeat30 506ms
    - testRepeat40 306ms

**BenchmarkTest.java**

```
src > test > java > edu > neu > coe > info6205 > util > BenchmarkTest.java > BenchmarkTest > testWaitPeriods()
12 public class BenchmarkTest {
13
14     int pre = 0;
15     int run = 0;
16     int post = 0;
17
18     @Test // slow
19     public void testWaitPeriods() throws Exception {
20         int nRuns = 2;
21         int warmups = 2;
22         BenchmarkTimer bm = new BenchmarkTimer();
23         Description "testWaitPeriods", b -> {
24             gotoSleep(mSecs, 100L, -1);
25             return null;
26         },
27         b -> {
28             gotoSleep(mSecs, 200L, which: 0);
29         },
30         b -> {
31             gotoSleep(mSecs, 50L, which: 1);
32         });
33         double x = bm.run(E, true, nRuns);
34         assertEquals(nRuns, post);
35         assertEquals(nRuns + warmups, run);
36         assertEquals(nRuns + warmups, pre);
37         assertEquals(200, x, 10);
38     }
39
40     private void gotoSleep(long mSecs, int which) {
41         try {
42             Thread.sleep(mSecs);
43             if (which == 0) run++;
44             else if (which > 0) post++;
45             else pre++;
46         } catch (InterruptedException e) {
47             e.printStackTrace();
48         }
49     }
50
51     @Test
52     public void getWarmupRuns() {
53         assertEquals(2, BenchmarkTimer.getWarmupRuns(0));
54         assertEquals(2, BenchmarkTimer.getWarmupRuns(10));
55         assertEquals(3, BenchmarkTimer.getWarmupRuns(20));
56     }
57 }
```

**TimerTest.java**

```
src > test > java > edu > neu > coe > info6205 > util > TimerTest.java > TimerTest > testRepeat4()
66 public void testRepeat4() {
67     final Timer timer = new Timer();
68     gotoSleep(TENTH, which: 0);
69     timer.start();
70     gotoSleep(TENTH, which: 0);
71     final double time = timer.stop();
72     assertEquals(TENTH_DOUBLE, time, 10.0);
73     assertEquals(2, run);
74 }
75
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, 10.0);
85     assertEquals(2, run);
86 }
87
88 @Test
89 public void testMillisecs() {
90     final Timer timer = new Timer();
91     gotoSleep(TENTH, which: 0);
92     timer.start();
93     final double time = timer.millisecs();
94     assertEquals(TENTH_DOUBLE, time, 10.0);
95     assertEquals(1, run);
96 }
97
98 @Test
99 public void testRepeat1() {
100     final Timer timer = new Timer();
101     final double mean = timer.repeat(10, 1) -> {
102         gotoSleep(MILLISEC, which: 0);
103         return null;
104     });
105     assertEquals(10, new PrivateMethodTester(timer).invokePrivate("name", "g
106     assertEquals(TENTH_DOUBLE / 10, mean, 0);
107     assertEquals(10, run);
108     assertEquals(0, pre);
109     assertEquals(0, post);
110 }
```

**PROBLEMS**

- OUTPUT
- DEBUG CONSOLE
- TERMINAL
- PORTS
- COMMENTS

`@sharunkumar-ks -> /workspaces/INF6205 (assignment-3) $ git diff src/test/java/edu/neu/coe/info6205/util/TimerTest.java`  
`@sharunkumar-ks -> /workspaces/INF6205 (assignment-3) $ git diff src/test/java/edu/neu/coe/info6205/util/BenchmarkTest.java`  
`@sharunkumar-ks -> /workspaces/INF6205 (assignment-3) $`