Varad Ratnakar Desai (001465732)

Program Structures & Algorithms

Spring 2021

Assignment No. 4

- **Task**

We mentioned two alternatives for implementing Union-Find:

1. For weighted quick union, store the depth rather than the size;
2. For weighted quick union with path compression, do two loops, so that all intermediate nodes point to the root, not just the alternates.

For both of these, code the alternative and benchmark it against the implementation in the repository. You have all of that available from a previous assignment.

If you can explain why alternative #1 is unnecessary to be benchmarked, you may skip benchmarking that one.

Usual submission rules apply. 40 points only for this one.

- **Output**

```
Comparison Between Height Weighted Quick Find vs Size Weighted Quick Find!

Union find method used: WQUPC
1: Count of Pairs = 579 for Size = 200
2: Count of Pairs = 1333 for Size = 400
3: Count of Pairs = 2936 for Size = 800
4: Count of Pairs = 6537 for Size = 1600
5: Count of Pairs = 13920 for Size = 3200
6: Count of Pairs = 30587 for Size = 6400
7: Count of Pairs = 62207 for Size = 12800
8: Count of Pairs = 139162 for Size = 25600
9: Count of Pairs = 288361 for Size = 51200
10: Count of Pairs = 614830 for Size = 102400
11: Count of Pairs = 1305463 for Size = 204800
12: Count of Pairs = 2739657 for Size = 409600
Union find method used: HWQUPC1
1: Count of Pairs = 589 for Size = 200
2: Count of Pairs = 1339 for Size = 400
3: Count of Pairs = 2912 for Size = 800
4: Count of Pairs = 6424 for Size = 1600
5: Count of Pairs = 13969 for Size = 3200
6: Count of Pairs = 29627 for Size = 6400
7: Count of Pairs = 63937 for Size = 12800
8: Count of Pairs = 136040 for Size = 25600
9: Count of Pairs = 291264 for Size = 51200
10: Count of Pairs = 625092 for Size = 102400
11: Count of Pairs = 1309799 for Size = 204800
12: Count of Pairs = 2790996 for Size = 409600

Benchmarking of No Compression Quick Find vs Total Compression Quick Find!

1: SIZE = 400
2021-03-02 09:56:25 INFO  Benchmark_Timer - Begin run: QU No Compression
Benchmark with 100 runs
0.32
2021-03-02 09:56:25 INFO  Benchmark_Timer - Begin run: QU Total Compression
Benchmark with 100 runs
0.25
2: SIZE = 800
2021-03-02 09:56:25 INFO  Benchmark_Timer - Begin run: QU No Compression
Benchmark with 100 runs
0.4
2021-03-02 09:56:25 INFO  Benchmark_Timer - Begin run: QU Total Compression
Benchmark with 100 runs
0.26
3: SIZE = 1600
```

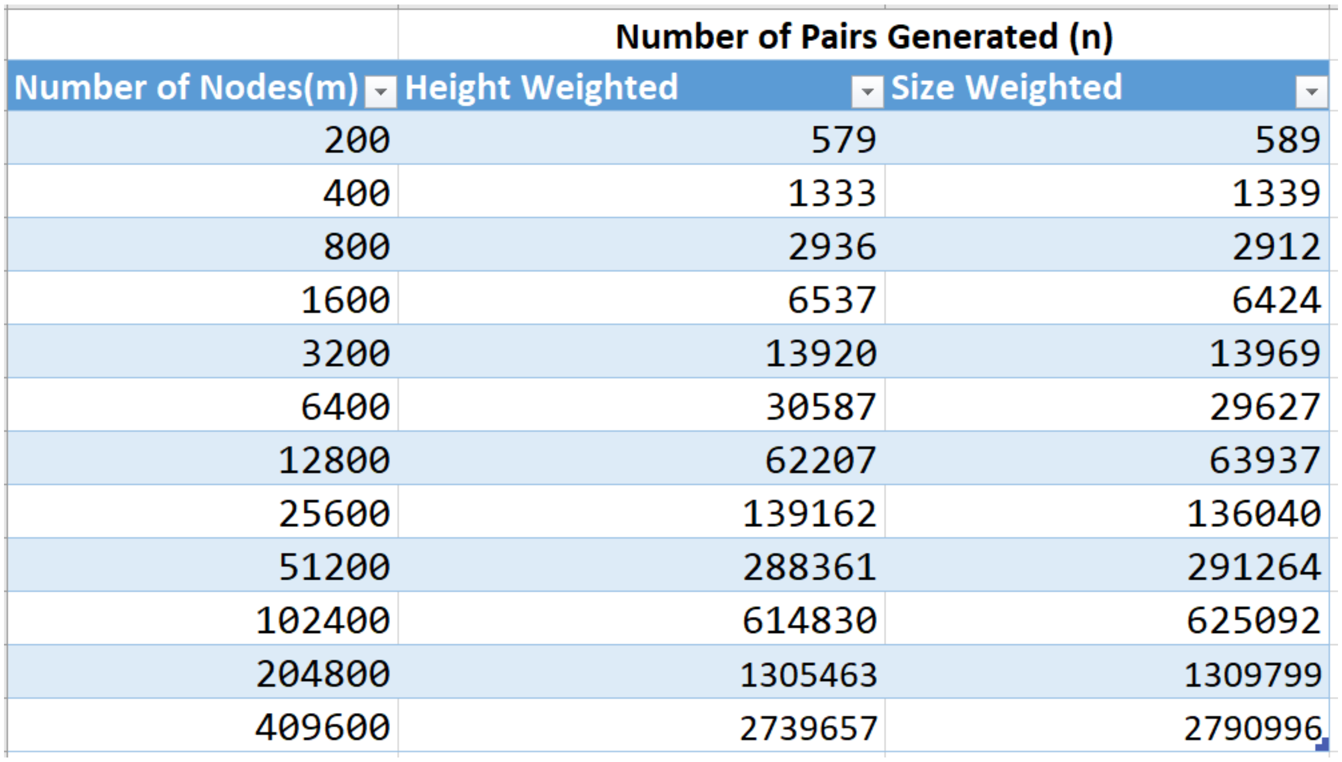
```
2021-03-02 09:56:25 INFO  Benchmark_Timer - Begin run: QU No Compression
Benchmark with 100 runs
0.63
2021-03-02 09:56:25 INFO  Benchmark_Timer - Begin run: QU Total Compression
Benchmark with 100 runs
0.57
4: SIZE = 3200
2021-03-02 09:56:26 INFO  Benchmark_Timer - Begin run: QU No Compression
Benchmark with 100 runs
1.32
2021-03-02 09:56:26 INFO  Benchmark_Timer - Begin run: QU Total Compression
Benchmark with 100 runs
1.2
5: SIZE = 6400
2021-03-02 09:56:26 INFO  Benchmark_Timer - Begin run: QU No Compression
Benchmark with 100 runs
3.04
2021-03-02 09:56:26 INFO  Benchmark_Timer - Begin run: QU Total Compression
Benchmark with 100 runs
2.56
6: SIZE = 12800
2021-03-02 09:56:26 INFO  Benchmark_Timer - Begin run: QU No Compression
Benchmark with 100 runs
6.38
2021-03-02 09:56:27 INFO  Benchmark_Timer - Begin run: QU Total Compression
Benchmark with 100 runs
5.66
7: SIZE = 25600
2021-03-02 09:56:28 INFO  Benchmark_Timer - Begin run: QU No Compression
Benchmark with 100 runs
13.8
2021-03-02 09:56:29 INFO  Benchmark_Timer - Begin run: QU Total Compression
Benchmark with 100 runs
11.86
8: SIZE = 51200
2021-03-02 09:56:31 INFO  Benchmark_Timer - Begin run: QU No Compression
Benchmark with 100 runs
30.59
2021-03-02 09:56:34 INFO  Benchmark_Timer - Begin run: QU Total Compression
Benchmark with 100 runs
26.14
9: SIZE = 102400
2021-03-02 09:56:37 INFO  Benchmark_Timer - Begin run: QU No Compression
Benchmark with 100 runs
71.47
2021-03-02 09:56:45 INFO  Benchmark_Timer - Begin run: QU Total Compression
Benchmark with 100 runs
57.04
10: SIZE = 204800
2021-03-02 09:56:51 INFO  Benchmark_Timer - Begin run: QU No Compression
Benchmark with 100 runs
```

```
156.63
2021-03-02 09:57:08 INFO  Benchmark_Timer - Begin run: QU Total Compression
Benchmark with 100 runs
126.76
11: SIZE = 409600
2021-03-02 10:18:18 INFO  Benchmark_Timer - Begin run: QU No Compression
Benchmark with 100 runs
367.47
2021-03-02 10:18:58 INFO  Benchmark_Timer - Begin run: QU Total Compression
Benchmark with 100 runs
279.16
12: SIZE = 819200
2021-03-02 10:19:29 INFO  Benchmark_Timer - Begin run: QU No Compression
Benchmark with 100 runs
823.61
2021-03-02 10:21:00 INFO  Benchmark_Timer - Begin run: QU Total Compression
Benchmark with 100 runs
667.41
```

- **Conclusion:**

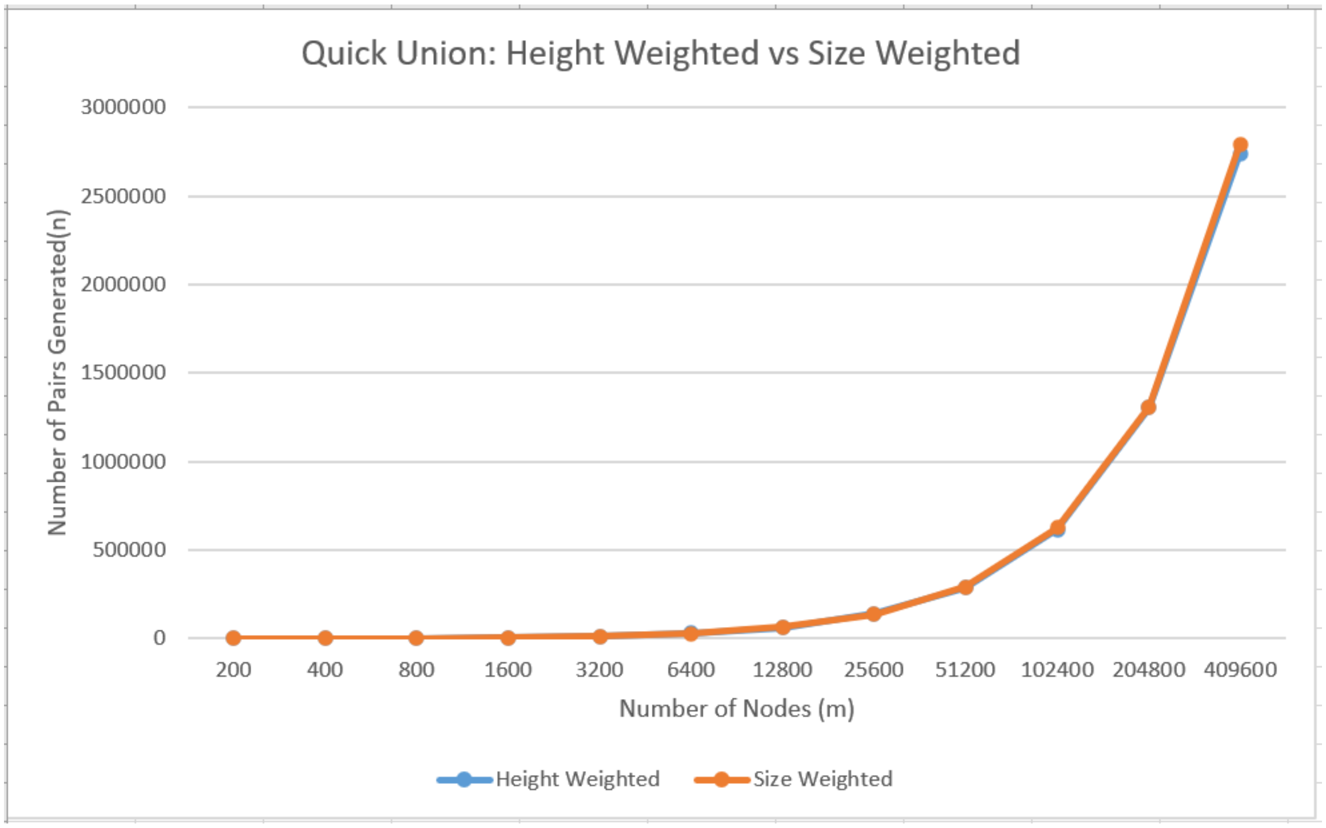
1. As we can see from table and graph below, we can say that Height Weighted and Size Weighted Quick Union show same result with same time complexity. Hence, we can say that, Benchmarking is not required for this comparison.
2. As we can see from the table and graph below, we can say that with No compression of path Quick union algorithm runs is 1.29 times slower than with Total Compression of path.

- **<u>Evidence to support the conclusion 1:</u>**
- **Tabular representation:**

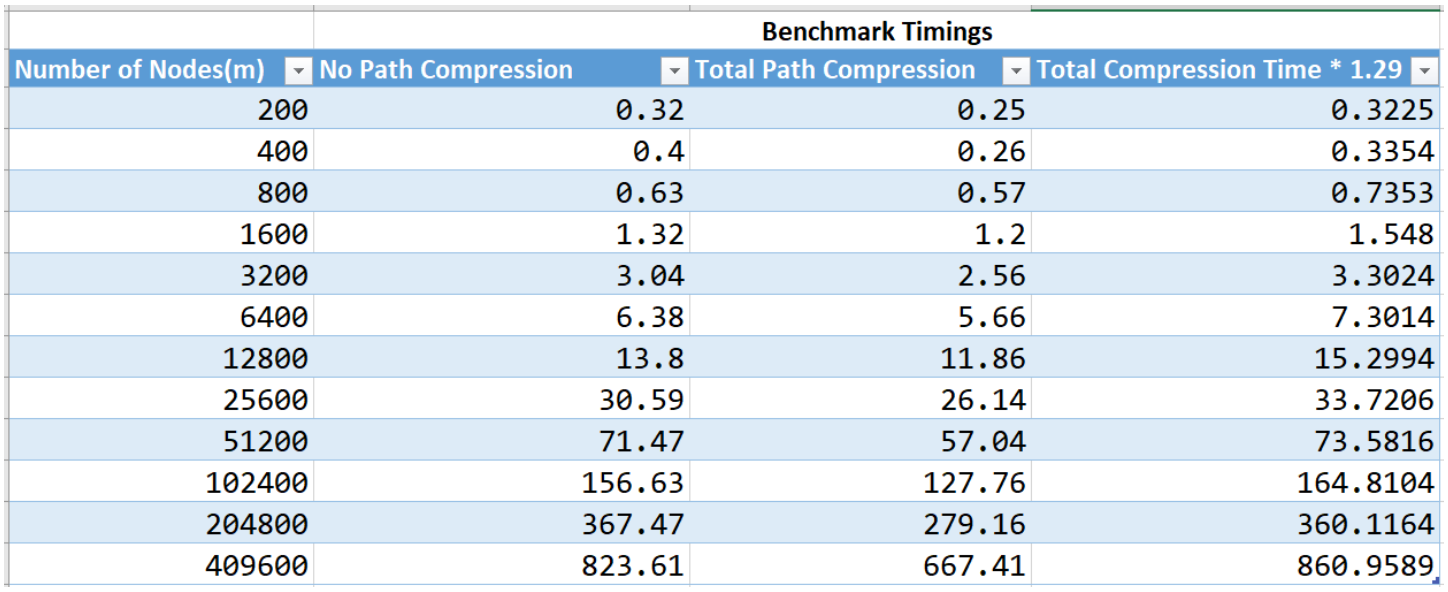


	Number of Pairs Generated (n)	
Number of Nodes(m)	**Height Weighted**	**Size Weighted**
200	579	589
400	1333	1339
800	2936	2912
1600	6537	6424
3200	13920	13969
6400	30587	29627
12800	62207	63937
25600	139162	136040
51200	288361	291264
102400	614830	625092
204800	1305463	1309799
409600	2739657	2790996

- **Graphical representation:**

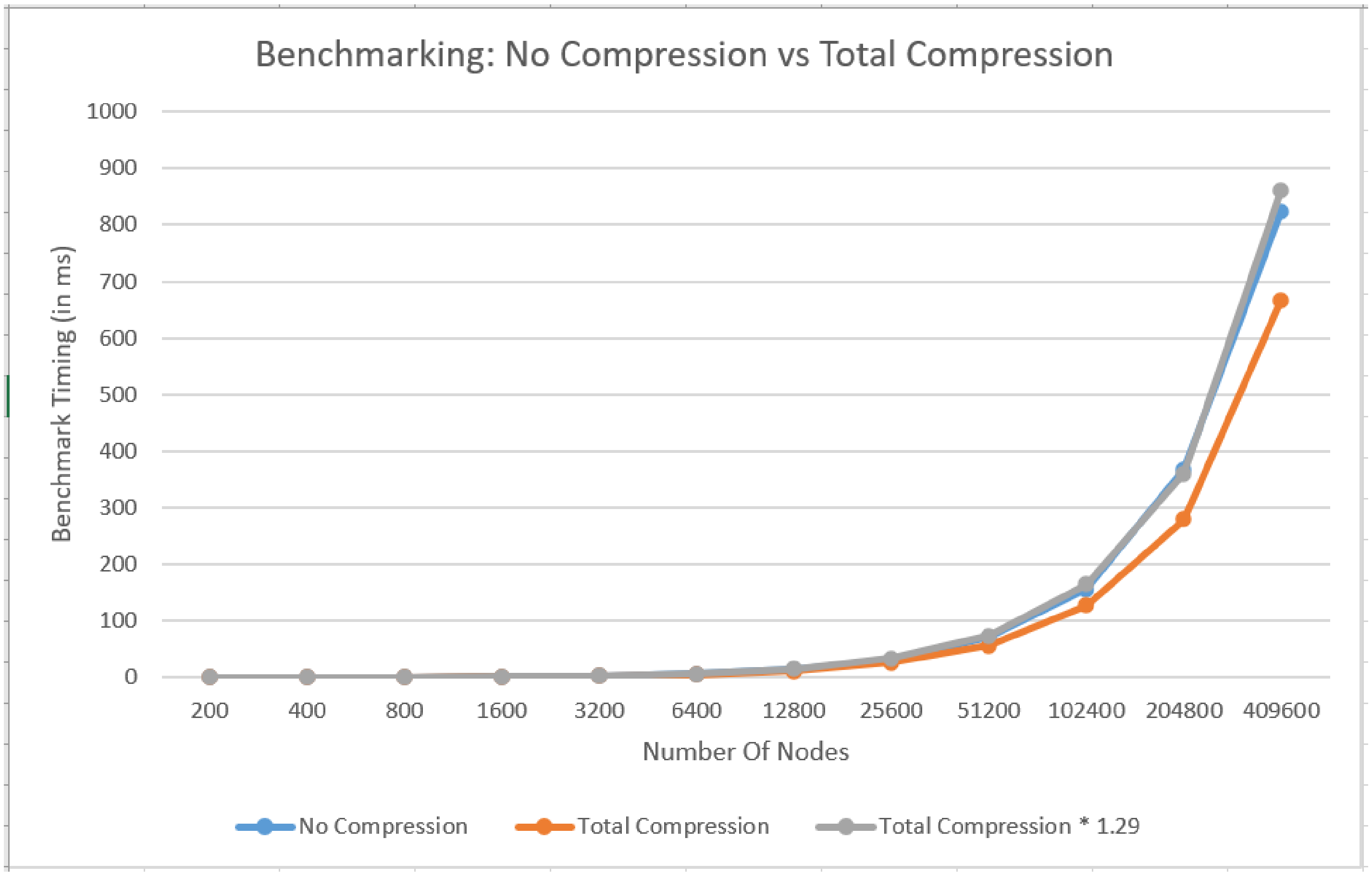


- **Evidence to support the conclusion 2:**
- **Tabular representation:**



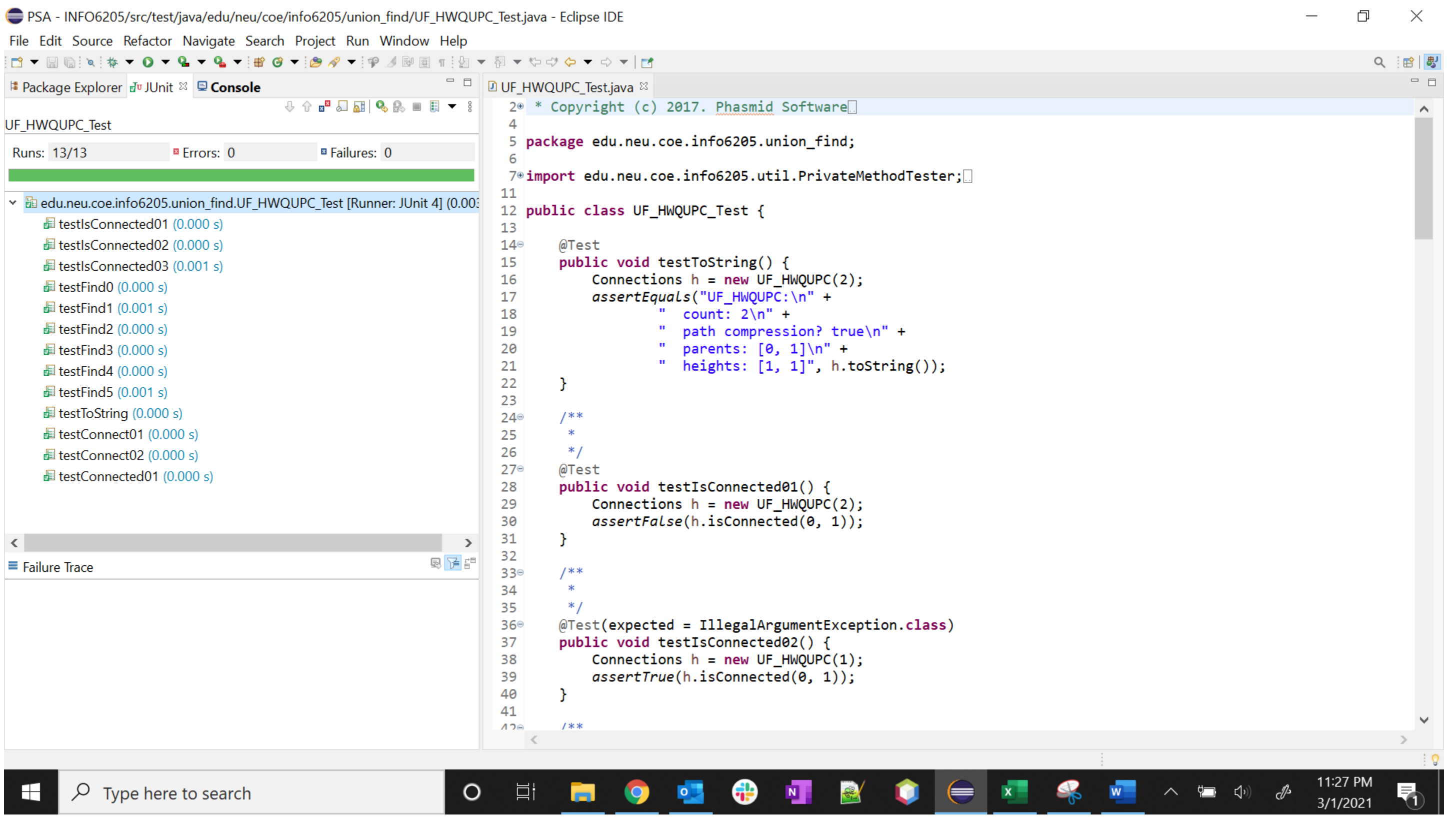
	Benchmark Timings		
Number of Nodes(m)	No Path Compression	Total Path Compression	Total Compression Time * 1.29
200	0.32	0.25	0.3225
400	0.4	0.26	0.3354
800	0.63	0.57	0.7353
1600	1.32	1.2	1.548
3200	3.04	2.56	3.3024
6400	6.38	5.66	7.3014
12800	13.8	11.86	15.2994
25600	30.59	26.14	33.7206
51200	71.47	57.04	73.5816
102400	156.63	127.76	164.8104
204800	367.47	279.16	360.1164
409600	823.61	667.41	860.9589

- **Graphical representation:**



- **Unit tests result**

1. **UF_HWQUPC_Test.java**



PSA - INFO6205/src/test/java/edu/neu/coe/info6205/union_find/UF_HWQUPC_Test.java - Eclipse IDE

UF_HWQUPC_Test

Runs: 13/13 Errors: 0 Failures: 0

```
edu.neu.coe.info6205.union_find.UF_HWQUPC_Test [Runner: JUnit 4] (0.00
    testIsConnected01 (0.000 s)
    testIsConnected02 (0.000 s)
    testIsConnected03 (0.001 s)
    testFind0 (0.000 s)
    testFind1 (0.001 s)
    testFind2 (0.000 s)
    testFind3 (0.000 s)
    testFind4 (0.000 s)
    testFind5 (0.001 s)
    testToString (0.000 s)
    testConnect01 (0.000 s)
    testConnect02 (0.000 s)
    testConnected01 (0.000 s)
```

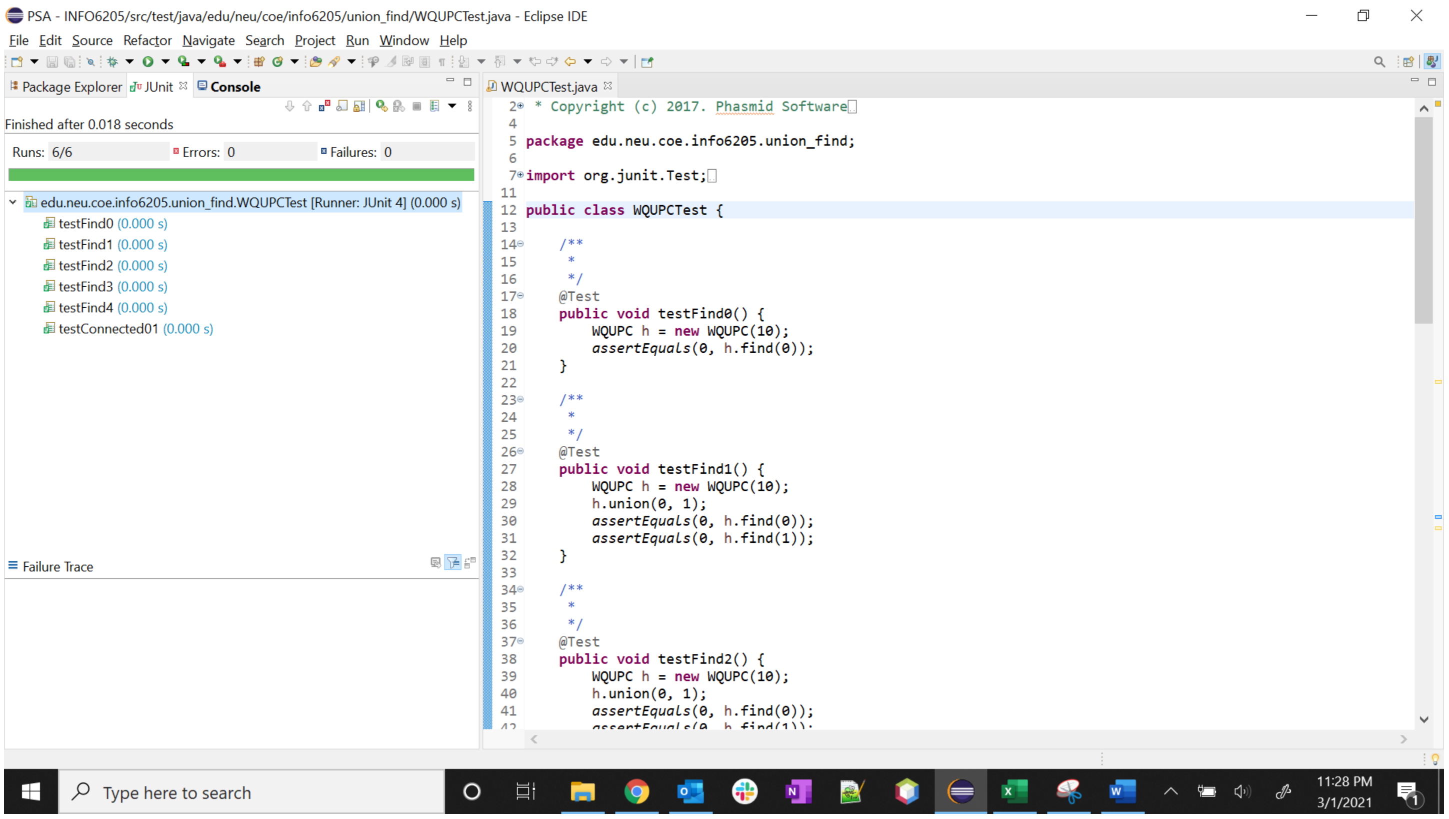
Failure Trace

```java
 * Copyright (c) 2017. Phasmid Software

package edu.neu.coe.info6205.union_find;

import edu.neu.coe.info6205.util.PrivateMethodTester;

public class UF_HWQUPC_Test {

    @Test
    public void testToString() {
        Connections h = new UF_HWQUPC(2);
        assertEquals("UF_HWQUPC:\n" +
                "  count: 2\n" +
                "  path compression? true\n" +
                "  parents: [0, 1]\n" +
                "  heights: [1, 1]", h.toString());
    }

    /**
     *
     */
    @Test
    public void testIsConnected01() {
        Connections h = new UF_HWQUPC(2);
        assertFalse(h.isConnected(0, 1));
    }

    /**
     *
     */
    @Test(expected = IllegalArgumentException.class)
    public void testIsConnected02() {
        Connections h = new UF_HWQUPC(1);
        assertTrue(h.isConnected(0, 1));
    }

    /**
```

Type here to search

11:27 PM
3/1/2021

2. **WQUPCTest.java**



PSA - INFO6205/src/test/java/edu/neu/coe/info6205/union_find/WQUPCTest.java - Eclipse IDE

Finished after 0.018 seconds

Runs: 6/6 Errors: 0 Failures: 0

```
edu.neu.coe.info6205.union_find.WQUPCTest [Runner: JUnit 4] (0.000 s)
    testFind0 (0.000 s)
    testFind1 (0.000 s)
    testFind2 (0.000 s)
    testFind3 (0.000 s)
    testFind4 (0.000 s)
    testConnected01 (0.000 s)
```

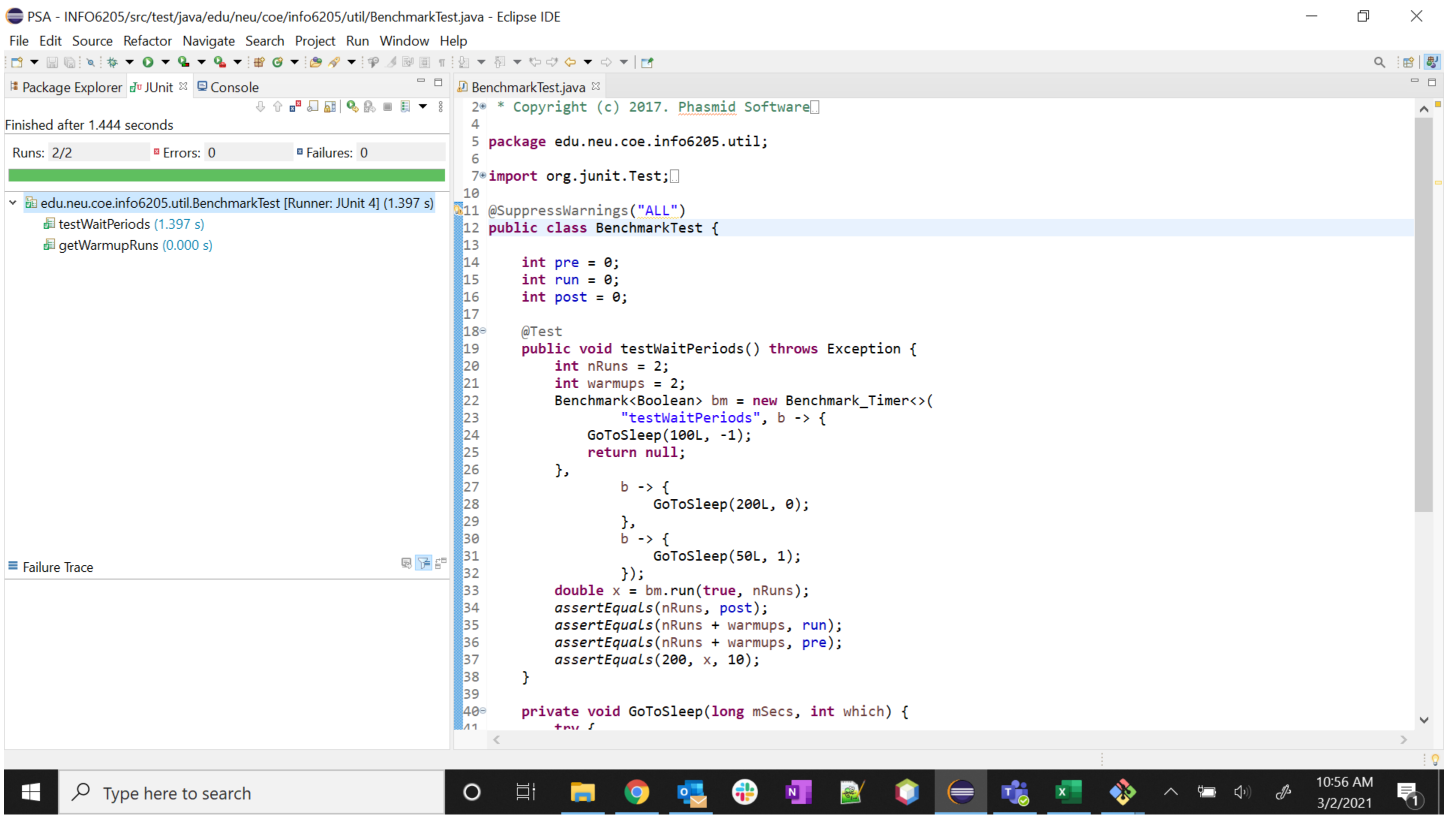
Failure Trace

```java
 * Copyright (c) 2017. Phasmid Software

package edu.neu.coe.info6205.union_find;

import org.junit.Test;

public class WQUPCTest {

    /**
     *
     */
    @Test
    public void testFind0() {
        WQUPC h = new WQUPC(10);
        assertEquals(0, h.find(0));
    }

    /**
     *
     */
    @Test
    public void testFind1() {
        WQUPC h = new WQUPC(10);
        h.union(0, 1);
        assertEquals(0, h.find(0));
        assertEquals(0, h.find(1));
    }

    /**
     *
     */
    @Test
    public void testFind2() {
        WQUPC h = new WQUPC(10);
        h.union(0, 1);
        assertEquals(0, h.find(0));
```

Type here to search

11:28 PM
3/1/2021

3. BenchmarkTest.java



PSA - INFO6205/src/test/java/edu/neu/coe/info6205/util/BenchmarkTest.java - Eclipse IDE

Finished after 1.444 seconds

Runs: 2/2 Errors: 0 Failures: 0

edu.neu.coe.info6205.util.BenchmarkTest [Runner: JUnit 4] (1.397 s)
testWaitPeriods (1.397 s)
getWarmupRuns (0.000 s)

```java
 * Copyright (c) 2017. Phasmid Software

package edu.neu.coe.info6205.util;

import org.junit.Test;

@SuppressWarnings("ALL")
public class BenchmarkTest {

    int pre = 0;
    int run = 0;
    int post = 0;

    @Test
    public void testWaitPeriods() throws Exception {
        int nRuns = 2;
        int warmups = 2;
        Benchmark<Boolean> bm = new Benchmark_Timer<>(
                "testWaitPeriods", b -> {
            GoToSleep(100L, -1);
            return null;
        },
                b -> {
                    GoToSleep(200L, 0);
                },
                b -> {
                    GoToSleep(50L, 1);
                });
        double x = bm.run(true, nRuns);
        assertEquals(nRuns, post);
        assertEquals(nRuns + warmups, run);
        assertEquals(nRuns + warmups, pre);
        assertEquals(200, x, 10);
    }

    private void GoToSleep(long mSecs, int which) {
        try {
```