Mayank Yadav (002193976)

INFO 6205

Program Structures & Algorithms

Assignment 3

TASK

We were given the following tasks:

1. Complete the implementation of class UF_HWQUPC.java, i.e. implement the find(), mergeComponents() and doPathCompression() method that perform find and union for height weighted quick union with path compression.

2. Develop a client that takes an integer n, generates random pairs from 0 to n-1 and returns the connections that are generated.

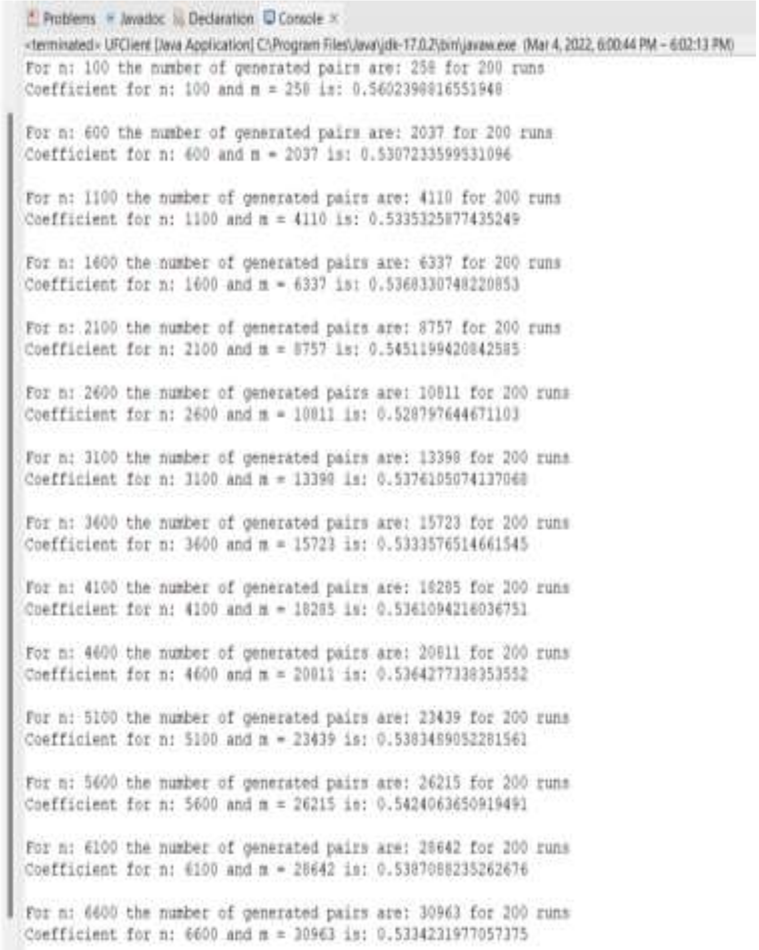
3. Derive the relationship between the generated pairs (m) and the number of objects (n).

Please find the code for the client in UFClient.java present in the /union_find directory.

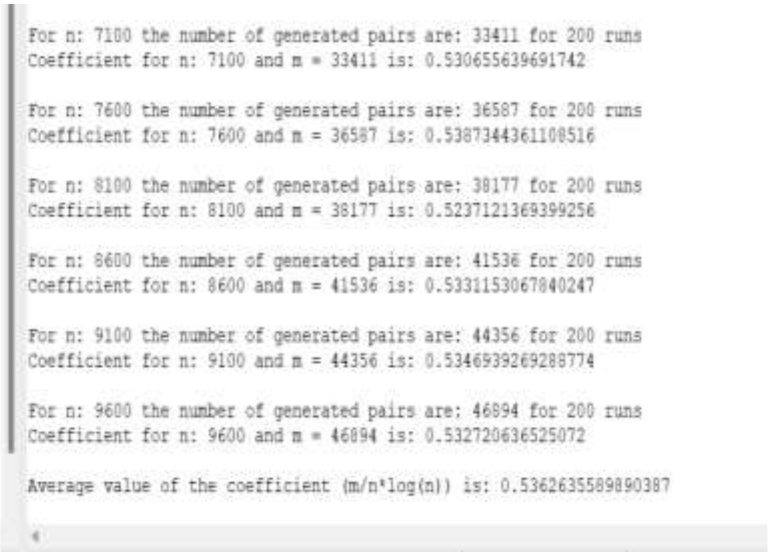
Output

After implementing the mergeComponents() find(), and doPathCompression() methods in UF_HWQUPC.java, I created a class called UFClient.java. In UFClient.java, there is a method count() which takes in an integer as an argument and generates random pairs between 0 and n-1. With these pairs, it checks whether they are connected, if not, it unions them. It does this till the number of components go from n to 1. It returns the generated pairs required to bring the components from n to 1. We call the number of generated pairs for a given n as 'm'. Since the value of m changed each time for a given n, I averaged the m values over 200 runs. I started from n = 100 and doubling it till n = 819,200 (total of 14 values of n). The console output is given below:

Mayank Yadav (002193976)



```
<terminated> UFClient [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Mar 4, 2022, 6:00:44 PM – 6:02:13 PM)
For n: 100 the number of generated pairs are: 258 for 200 runs
Coefficient for n: 100 and m = 258 is: 0.5602398816551948

For n: 600 the number of generated pairs are: 2037 for 200 runs
Coefficient for n: 600 and m = 2037 is: 0.5307233599331096

For n: 1100 the number of generated pairs are: 4110 for 200 runs
Coefficient for n: 1100 and m = 4110 is: 0.5335325877435249

For n: 1600 the number of generated pairs are: 6337 for 200 runs
Coefficient for n: 1600 and m = 6337 is: 0.5368330748220853

For n: 2100 the number of generated pairs are: 8757 for 200 runs
Coefficient for n: 2100 and m = 8757 is: 0.5451199420842585

For n: 2600 the number of generated pairs are: 10811 for 200 runs
Coefficient for n: 2600 and m = 10811 is: 0.528797644671103

For n: 3100 the number of generated pairs are: 13398 for 200 runs
Coefficient for n: 3100 and m = 13398 is: 0.5376105074137068

For n: 3600 the number of generated pairs are: 15723 for 200 runs
Coefficient for n: 3600 and m = 15723 is: 0.5333576514661545

For n: 4100 the number of generated pairs are: 18285 for 200 runs
Coefficient for n: 4100 and m = 18285 is: 0.5361094216036751

For n: 4600 the number of generated pairs are: 20811 for 200 runs
Coefficient for n: 4600 and m = 20811 is: 0.5364277338353552

For n: 5100 the number of generated pairs are: 23439 for 200 runs
Coefficient for n: 5100 and m = 23439 is: 0.5383489052281561

For n: 5600 the number of generated pairs are: 26215 for 200 runs
Coefficient for n: 5600 and m = 26215 is: 0.5424063650919491

For n: 6100 the number of generated pairs are: 28642 for 200 runs
Coefficient for n: 6100 and m = 28642 is: 0.5387088235262676

For n: 6600 the number of generated pairs are: 30963 for 200 runs
Coefficient for n: 6600 and m = 30963 is: 0.5334231977057375
```



```
For n: 7100 the number of generated pairs are: 33411 for 200 runs
Coefficient for n: 7100 and m = 33411 is: 0.530655639691742

For n: 7600 the number of generated pairs are: 36587 for 200 runs
Coefficient for n: 7600 and m = 36587 is: 0.5387344361108516

For n: 8100 the number of generated pairs are: 38177 for 200 runs
Coefficient for n: 8100 and m = 38177 is: 0.5237121369399256

For n: 8600 the number of generated pairs are: 41536 for 200 runs
Coefficient for n: 8600 and m = 41536 is: 0.5331153067840247

For n: 9100 the number of generated pairs are: 44356 for 200 runs
Coefficient for n: 9100 and m = 44356 is: 0.5346939269288774

For n: 9600 the number of generated pairs are: 46894 for 200 runs
Coefficient for n: 9600 and m = 46894 is: 0.532720636525072

Average value of the coefficient (m/n*log(n)) is: 0.5362635589890387
```

Relationship Conclusion

I have recorded the results for range of n = 100 to n = 4100 in the table below:

n	m	$\frac{m}{n}$
100	263	2.630
600	2155	3.591
1100	4206	3.823
1600	6544	4.09
2100	8813	4.196
2600	11002	4.231
3100	13200	4.258
3600	15690	4.358
4100	18485	4.508

These results weren't that surprising considering the logarithmic nature of height weighted quick union. In height (or even weight) weighted quick union, the height of a node with a tree having total of 2k nodes is at most k or in other words, the height of a tree having total of n nodes is at most log (n). Since in this mechanism we keep track of the height of each tree and connect the smaller tree to the larger rather than doing it arbitrarily, we are guaranteed a logarithmic performance. If we assume that we perform n unions and maximum height of a node is log(n), we can conclude that the number of operations required to perform n unions would be at most n*log(n). With this conclusion, I hypothesized that there must be a n*log(n) factor in the relationship between m and n.

To check this, I ran my client again and I found m/(n* log(n)) and recorded the results for range of n = 100 to n = 4100 below:

n	m	$\frac{\boldsymbol{m}}{n * \log{(n)}}$
100	263	0.571
600	2155	0.561
1100	4206	0.545
1600	6544	0.554
2100	8813	0.548
2600	11002	0.531
3100	13200	0.529
3600	15690	0.532
4100	18485	0.541

From the results above, we see that there is indeed a relation between n and m where the coefficient is given by m/(n* log(n)) . Averaging these for 80 values of n, I found the value of m/(n* log(n)) to be roughly 0.53.

From these results we can conclude that:

$$m \approx k * n * \log{(n)}$$

Where $k \approx 0.53$

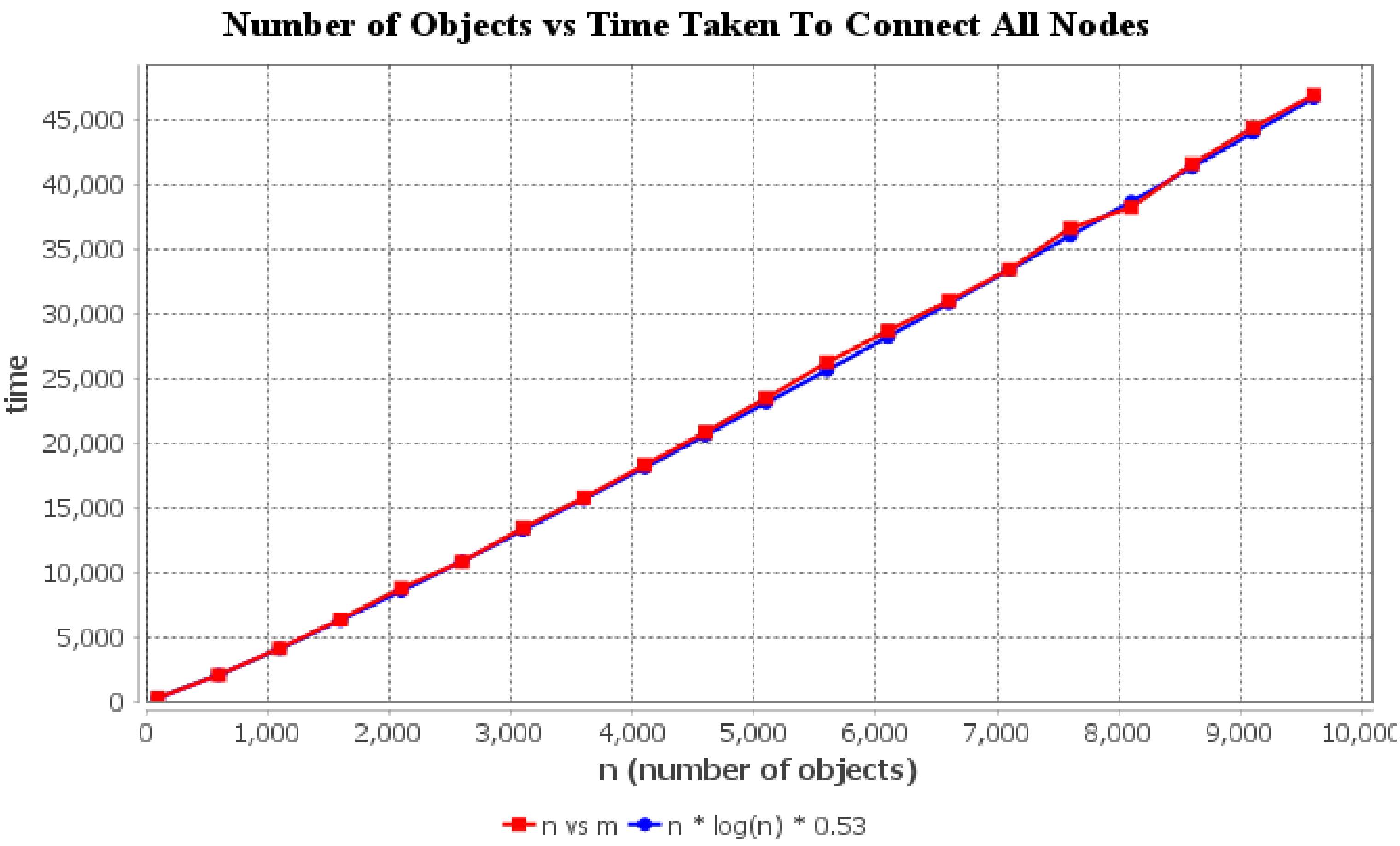
Mayank Yadav (002193976)

Relationship Evidence

To prove the relationship

$$m \approx 0.53 * n * \log(n)$$

I plotted the graph for n vs m and n vs n*log(n)*0.53. The graph is provided below:

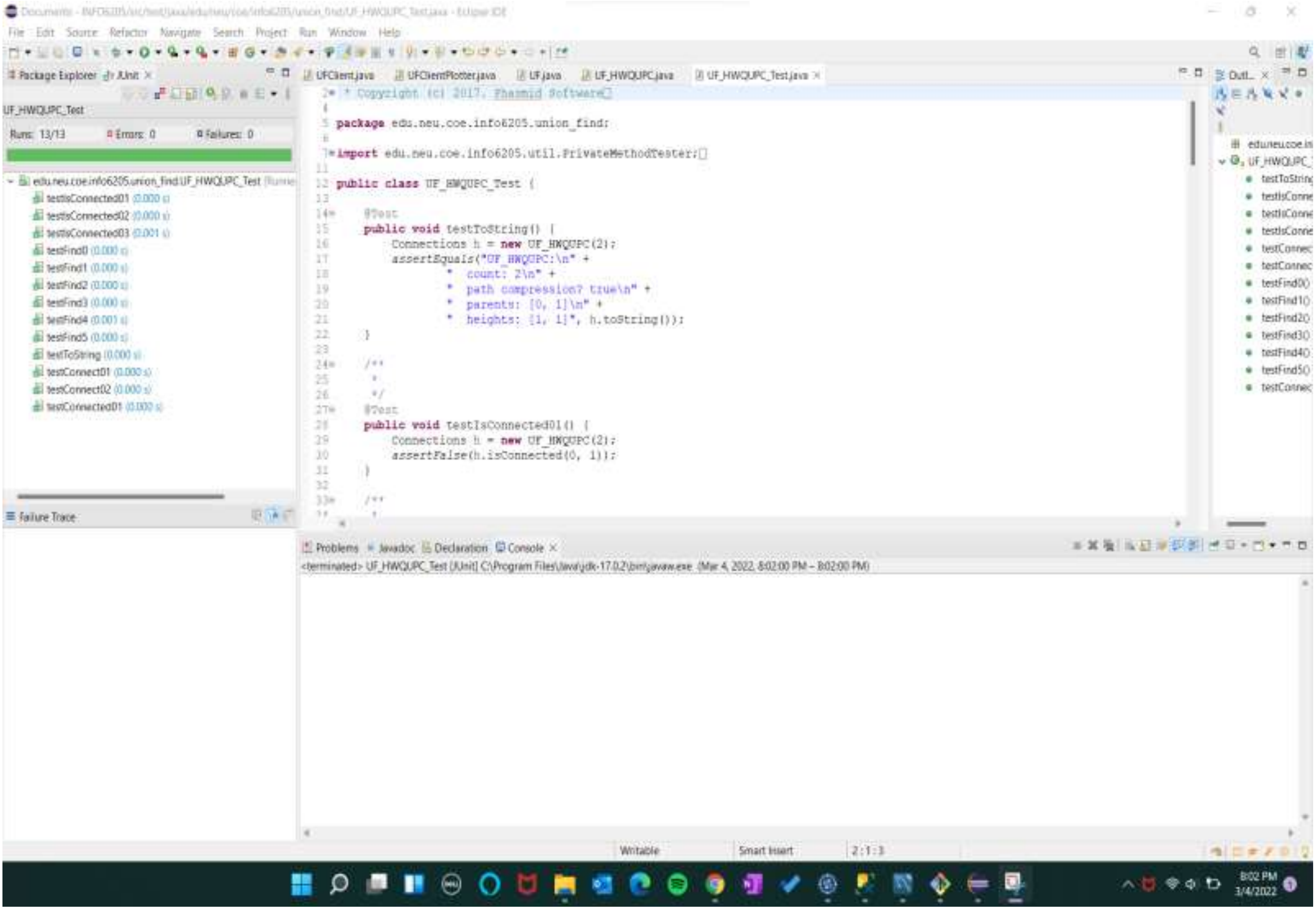


The overlapping line graph for both n vs m and n vs n*log(n)*0.53 supports our conclusion that the relationship between n and m can be defined by:

$$m \approx 0.53 * n * \log(n)$$

Mayank Yadav (002193976)

Screenshot of Unit Test Passing



UF_HWQUPC_Test

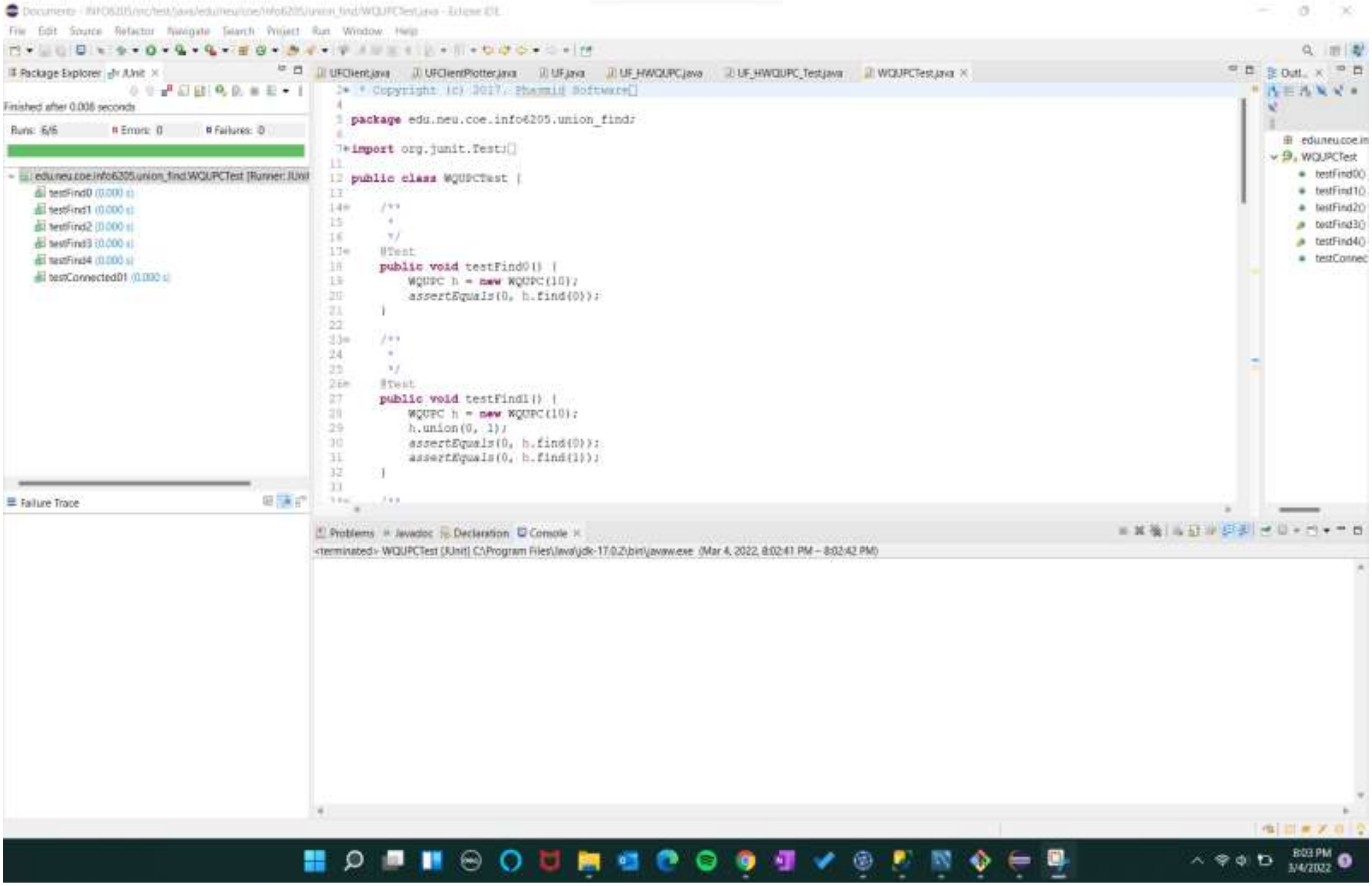
Runs: 13/13 Errors: 0 Failures: 0

```
edu.neu.coe.info6205.union_find.UF_HWQUPC_Test (Runne
    testIsConnected01 (0.000 s)
    testIsConnected02 (0.000 s)
    testIsConnected03 (0.001 s)
    testFind0 (0.000 s)
    testFind1 (0.000 s)
    testFind2 (0.000 s)
    testFind3 (0.000 s)
    testFind4 (0.001 s)
    testFind5 (0.000 s)
    testToString (0.000 s)
    testConnect01 (0.000 s)
    testConnect02 (0.000 s)
    testConnected01 (0.000 s)
```

```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));
    }

    /**
```

```
<terminated> UF_HWQUPC_Test [JUnit] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Mar 4, 2022, 8:02:00 PM – 8:02:00 PM)
```

Mayank Yadav (002193976)



Finished after 0.008 seconds

Runs: 6/6 Errors: 0 Failures: 0

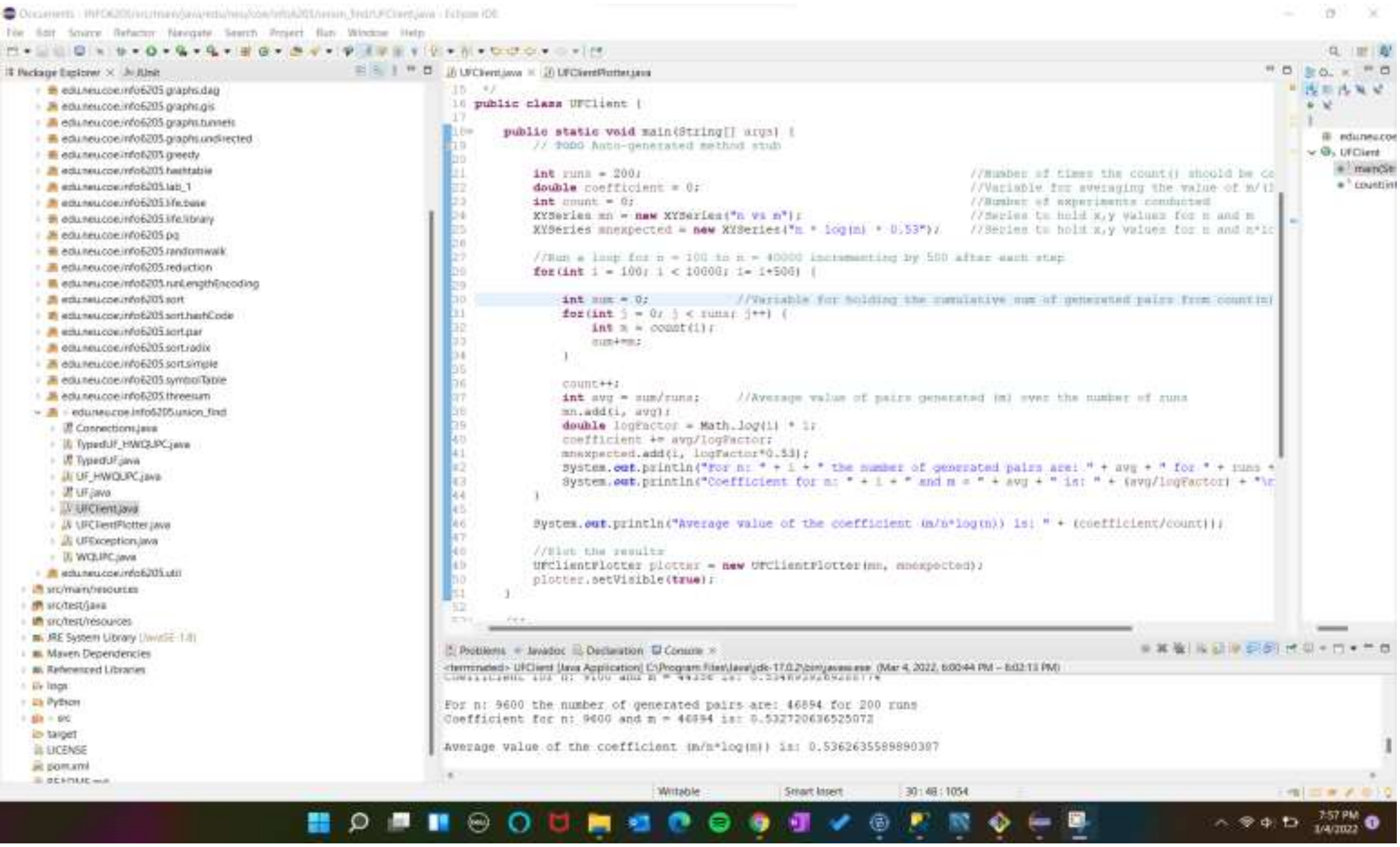
```
edu.neu.coe.info6205.union_find.WQUPCTest [Runner: JUnit
    testFind0 (0.000 s)
    testFind1 (0.000 s)
    testFind2 (0.000 s)
    testFind3 (0.000 s)
    testFind4 (0.000 s)
    testConnected01 (0.000 s)
```

```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));
    }

    /**
```

```
<terminated> WQUPCTest [JUnit] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Mar 4, 2022, 8:02:41 PM – 8:02:42 PM)
```

Mayank Yadav (002193976)

Code

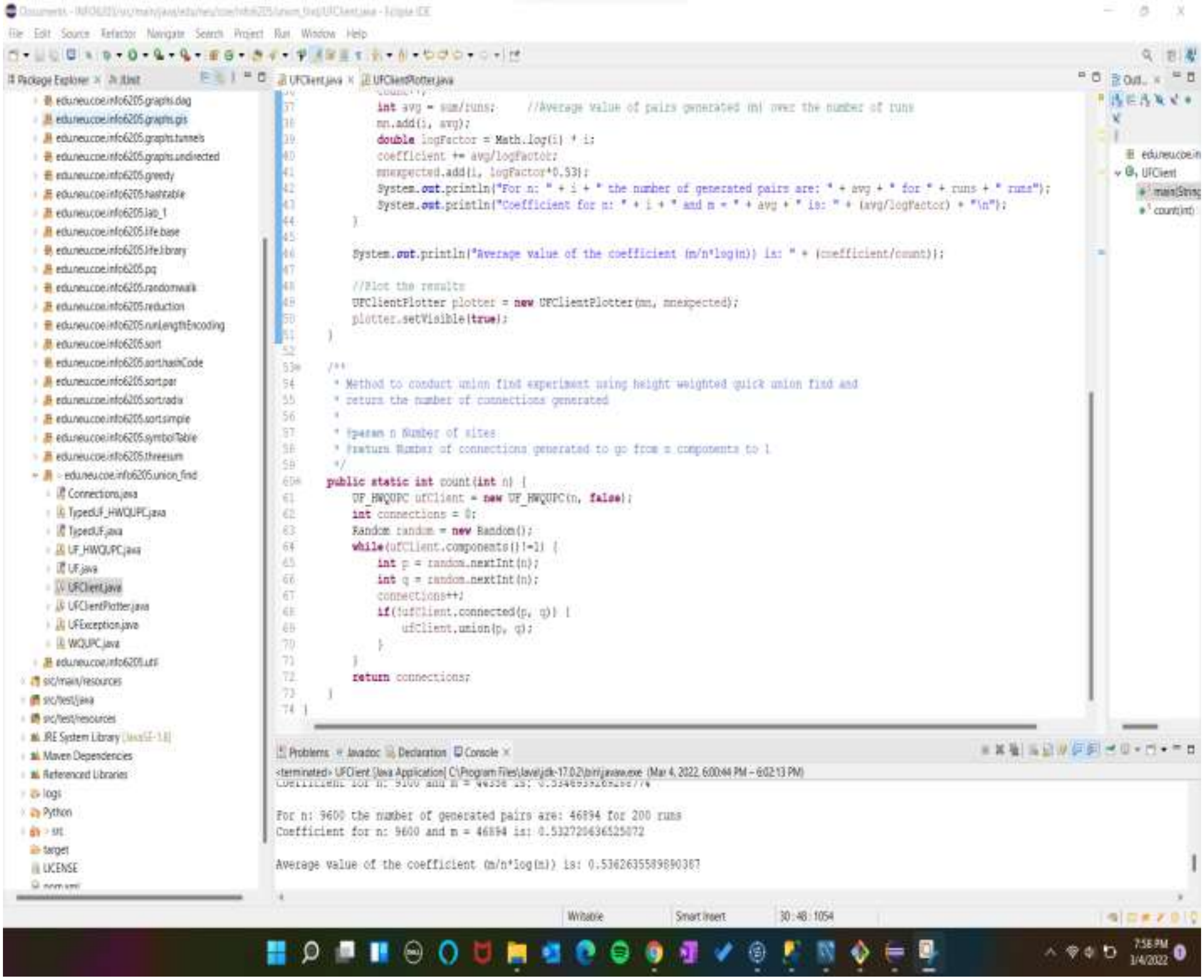


```
*/
public class UFClient {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        int runs = 200;                                         //Number of times the count() should be co
        double coefficient = 0;                                 //Variable for averaging the value of m/(l
        int count = 0;                                          //Number of experiments conducted
        XYSeries mn = new XYSeries("n vs m");                   //Series to hold x,y values for n and m
        XYSeries mnexpected = new XYSeries("n * log(n) * 0.53");    //Series to hold x,y values for n and n*lo

        //Run a loop for n = 100 to n = 40000 incrementing by 500 after each step
        for(int i = 100; i < 10000; i= i+500) {

            int sum = 0;            //Variable for holding the cumulative sum of generated pairs from count(n)
            for(int j = 0; j < runs; j++) {
                int m = count(i);
                sum+=m;
            }

            count++;
            int avg = sum/runs;     //Average value of pairs generated (m) over the number of runs
            mn.add(i, avg);
            double logFactor = Math.log(i) * i;
            coefficient += avg/logFactor;
            mnexpected.add(i, logFactor*0.53);
            System.out.println("For n: " + i + " the number of generated pairs are: " + avg + " for " + runs +
            System.out.println("Coefficient for n: " + i + " and m = " + avg + " is: " + (avg/logFactor) + "\r
        }

        System.out.println("Average value of the coefficient (m/n*log(n)) is: " + (coefficient/count));

        //Plot the results
        UFClientPlotter plotter = new UFClientPlotter(mn, mnexpected);
        plotter.setVisible(true);
    }
```

<terminated> UFClient [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Mar 4, 2022, 6:00:44 PM – 6:02:13 PM)

```
For n: 9600 the number of generated pairs are: 46894 for 200 runs
Coefficient for n: 9600 and m = 46894 is: 0.532720636525072

Average value of the coefficient (m/n*log(m)) is: 0.5362635589890397
```

Mayank Yadav (002193976)


Documents - INFO6205/src/main/java/edu/neu/coe/info6205/union_find/UFClient.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
edu.neu.coe.info6205.graphs.dag
edu.neu.coe.info6205.graphs.gis
edu.neu.coe.info6205.graphs.tunnels
edu.neu.coe.info6205.graphs.undirected
edu.neu.coe.info6205.greedy
edu.neu.coe.info6205.hashtable
edu.neu.coe.info6205.lab_1
edu.neu.coe.info6205.life.base
edu.neu.coe.info6205.life.library
edu.neu.coe.info6205.pq
edu.neu.coe.info6205.randomwalk
edu.neu.coe.info6205.reduction
edu.neu.coe.info6205.runLengthEncoding
edu.neu.coe.info6205.sort
edu.neu.coe.info6205.sort.hashCode
edu.neu.coe.info6205.sort.par
edu.neu.coe.info6205.sort.radix
edu.neu.coe.info6205.sort.simple
edu.neu.coe.info6205.symbolTable
edu.neu.coe.info6205.threesum
edu.neu.coe.info6205.union_find
Connections.java
TypedUF_HWQUPC.java
TypedUF.java
UF_HWQUPC.java
UF.java
UFClient.java
UFClientPlotter.java
UFException.java
WQUPC.java
edu.neu.coe.info6205.util
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-1.8]
Maven Dependencies
Referenced Libraries
logs
Python
src
target
LICENSE
pom.xml
UFClient.java
UFClientPlotter.java
Out...
edu.neu.coe.in
UFClient
main(String
count(int)
Problems
Javadoc
Declaration
Console
Writable
Smart Insert
30 : 48 : 1054
7:58 PM
3/4/2022

```java
            int avg = sum/runs;     //Average value of pairs generated (m) over the number of runs
            mn.add(i, avg);
            double logFactor = Math.log(i) * i;
            coefficient += avg/logFactor;
            mnexpected.add(i, logFactor*0.53);
            System.out.println("For n: " + i + " the number of generated pairs are: " + avg + " for " + runs + " runs");
            System.out.println("Coefficient for n: " + i + " and m = " + avg + " is: " + (avg/logFactor) + "\n");
        }

        System.out.println("Average value of the coefficient (m/n*log(n)) is: " + (coefficient/count));

        //Plot the results
        UFClientPlotter plotter = new UFClientPlotter(mn, mnexpected);
        plotter.setVisible(true);
    }

    /**
     * Method to conduct union find experiment using height weighted quick union find and
     * return the number of connections generated
     *
     * @param n Number of sites
     * @return Number of connections generated to go from n components to 1
     */
    public static int count(int n) {
        UF_HWQUPC ufClient = new UF_HWQUPC(n, false);
        int connections = 0;
        Random random = new Random();
        while(ufClient.components()!=1) {
            int p = random.nextInt(n);
            int q = random.nextInt(n);
            connections++;
            if(!ufClient.connected(p, q)) {
                ufClient.union(p, q);
            }
        }
        return connections;
    }
}
```

```
<terminated> UFClient [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Mar 4, 2022, 6:00:44 PM – 6:02:13 PM)

For n: 9600 the number of generated pairs are: 46894 for 200 runs
Coefficient for n: 9600 and m = 46894 is: 0.5327206365250 72

Average value of the coefficient (m/n*log(n)) is: 0.5362635589890387
```