Program Structures and Algorithms
Spring 2022
Assignment 3 (WQUPC)

Name: Vraj Himanshu Reshamdalal
(**NUID**): 002927484

Task:

Step 1:

(a) Implement height-weighted Quick Union with Path Compression. For this, you will flesh out the class UF_HWQUPC. All you have to do is to fill in the sections marked with // TO BE IMPLEMENTED ... // ...END IMPLEMENTATION.

(b) Check that the unit tests for this class all work. You must show "green" test results in your submission (screenshot is OK).

Step 2:

Using your implementation of UF_HWQUPC, develop a UF ("union-find") client that takes an integer value n from the command line to determine the number of "sites." Then generates random pairs of integers between 0 and n-1, calling connected() to determine if they are connected and union() if not. Loop until all sites are connected then print the number of connections generated. Package your program as a static method count() that takes n as the argument and returns the number of connections; and a main() that takes n from the command line, calls count() and prints the returned value. If you prefer, you can create a main program that doesn't require any input and runs the experiment for a fixed set of n values. Show evidence of your run(s).

Step 3:

Determine the relationship between the number of objects (*n*) and the number of pairs (*m*) generated to accomplish this (i.e. to reduce the number of components from *n* to 1). Justify your conclusion in terms of your observations and what you think might be going on.

Code Stubs:

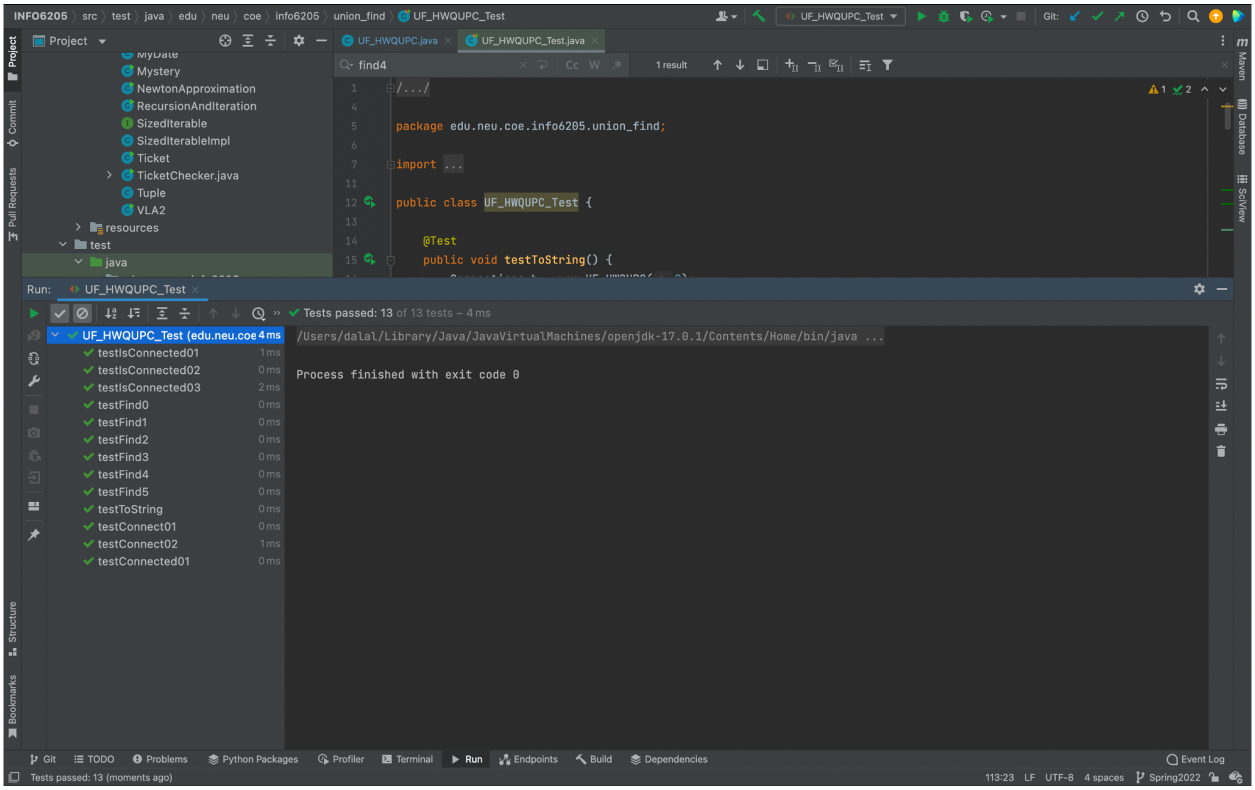
Part 1: Completed

```java
     */
    public int find(int p) {
        validate(p);
        int root = p;
        // FIXME
        while(root != getParent(root)) {
            if(this.pathCompression == true) {
                this.doPathCompression(root);
            }

            root = getParent(root);
        }
        // END
        return root;
    }

    /**
```

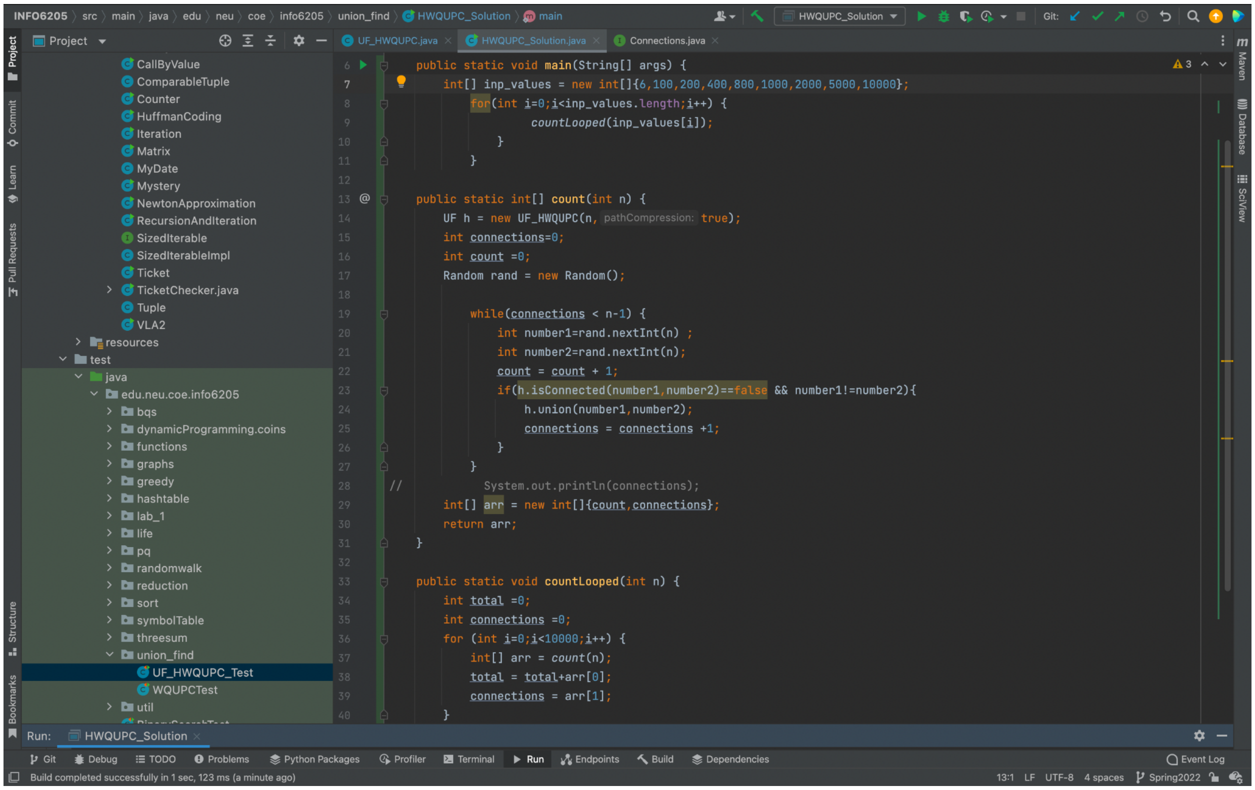
```java
    */
    private void doPathCompression(int i) {
        // FIXME update parent to value of grandparent
        parent[i] = parent[parent[i]];
        // END
    }
}
```

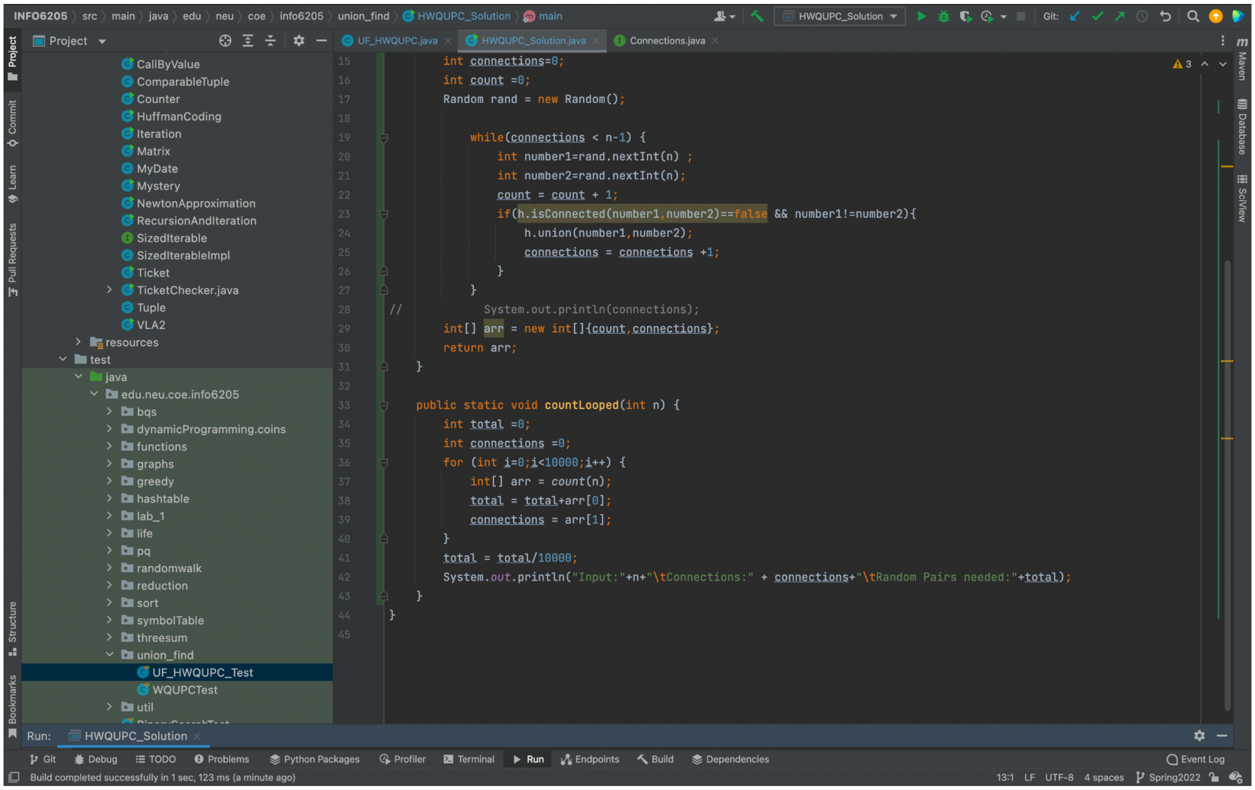
```java
    private void mergeComponents(int i, int j) {
        // FIXME make shorter root point to taller one
        int t1 = find(i);
        int t2 = find(j);
        if(t1 == t2) {
            return;
        }
        if(height[t1] < height[t2]) {
            parent[t1] = t2;
            height[t2] = height[t2] + height[t1];
        } else {
            parent[t2] = t1;
            height[t1] = height[t1] + height[t2];
        }
        // END
    }
```

Part **1** output:


INFO6205 src test java edu neu coe info6205 union_find UF_HWQUPC_Test
UF_HWQUPC_Test
Project
UF_HWQUPC.java
UF_HWQUPC_Test.java
find4
1 result
Mystery
NewtonApproximation
RecursionAndIteration
SizedIterable
SizedIterableImpl
Ticket
TicketChecker.java
Tuple
VLA2
resources
test
java
/.../
package edu.neu.coe.info6205.union_find;
import ...
public class UF_HWQUPC_Test {
@Test
public void testToString() {
Run: UF_HWQUPC_Test
Tests passed: 13 of 13 tests – 4 ms
UF_HWQUPC_Test (edu.neu.coe 4 ms
testIsConnected01 1 ms
testIsConnected02 0 ms
testIsConnected03 2 ms
testFind0 0 ms
testFind1 0 ms
testFind2 0 ms
testFind3 0 ms
testFind4 0 ms
testFind5 0 ms
testToString 0 ms
testConnect01 0 ms
testConnect02 1 ms
testConnected01 0 ms
/Users/dalal/Library/Java/JavaVirtualMachines/openjdk-17.0.1/Contents/Home/bin/java ...
Process finished with exit code 0
Git TODO Problems Python Packages Profiler Terminal Run Endpoints Build Dependencies Event Log
Tests passed: 13 (moments ago)
113:23 LF UTF-8 4 spaces Spring2022

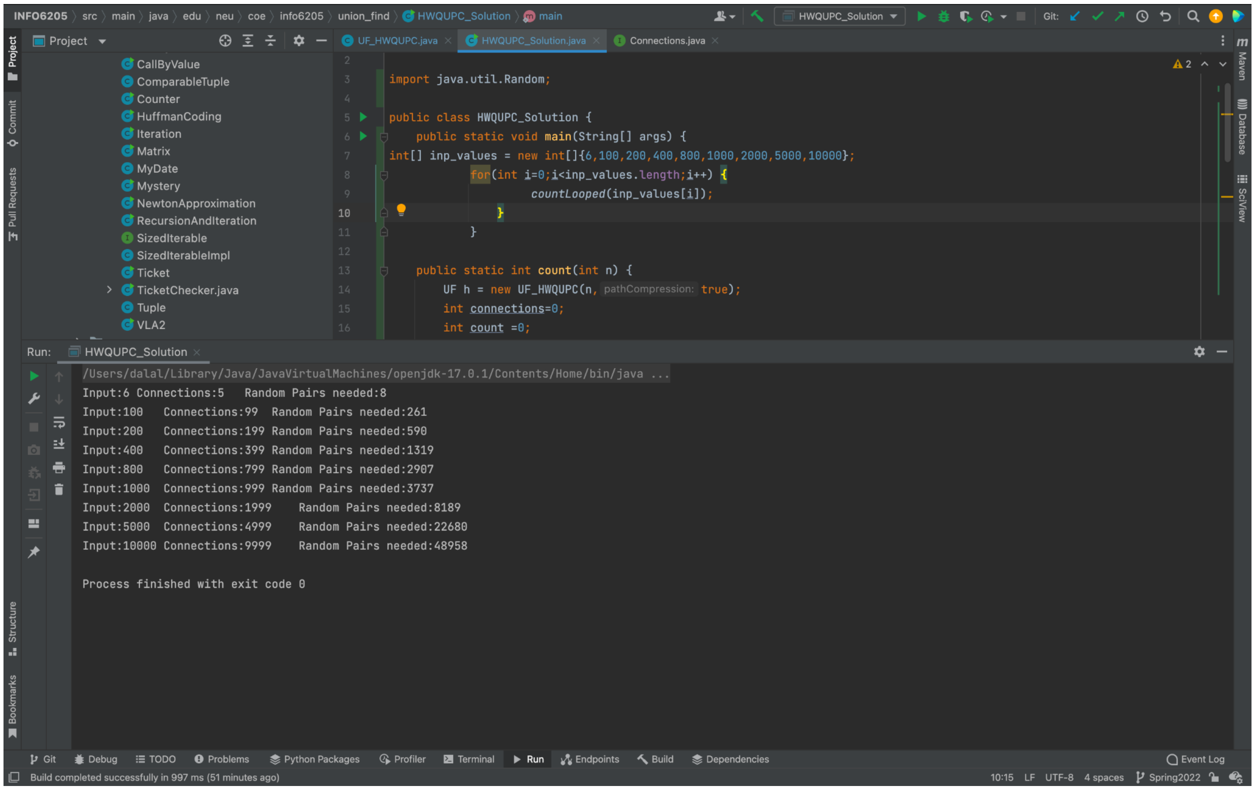
Part2: Completed





Part3: Completed

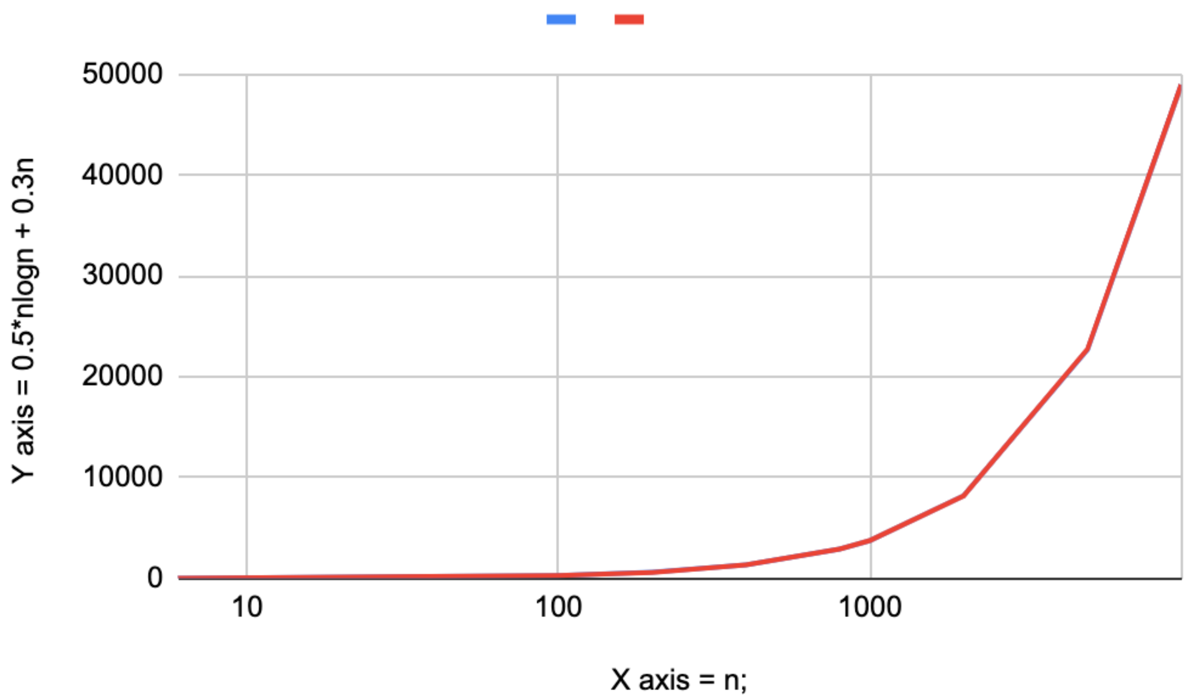
Part **3** output:



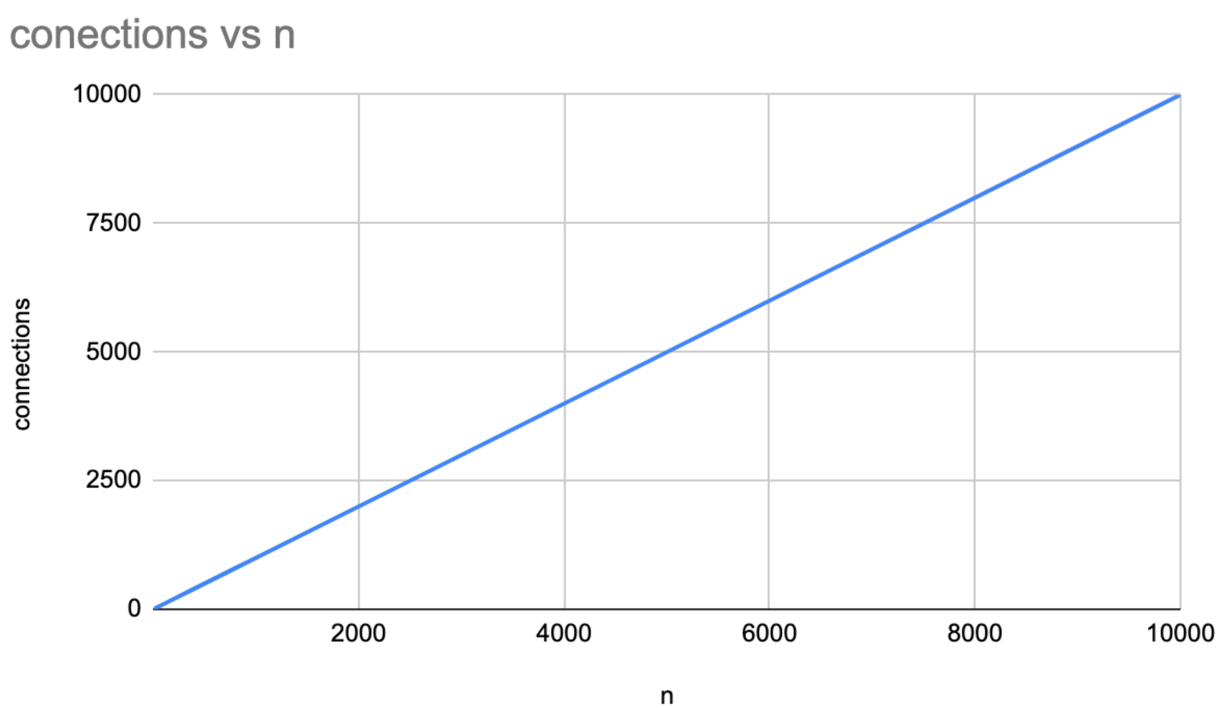
Evidence/graph:

connections	n	0.5*nlogn + 0.29n	m
5	6	7.115278408	8
99	100	259.2585093	261
199	200	587.8317367	590
399	400	1314.292909	1319
799	800	2905.844691	2907
999	1000	3743.877639	3737
1999	2000	8180.90246	8189
4999	5000	22742.98298	22680
9999	10000	48951.70186	48958

Graph



50000
40000
30000
20000
10000
0
Y axis = 0.5*nlogn + 0.3n
10
100
1000
X axis = n;



conections vs n
10000
7500
5000
2500
0
connections
2000
4000
6000
8000
10000
n

<u>Relationship conclusion</u>:

Based on the calculations and the table shown above we can conclude that,
m=number of pairs generated
connections=number of connections
c = constant

Connections = n-1

m = 0.5*nlogn + 0.3n OR m = 0.5*nlogn + c

Justification:

As per the Erdös-Renyi model - the number of pairs generated to get one component is ~ 1⁄2N ln N.

References:

1. **PSA text book - Algorithhms 4th Edition by Robert Sedgewick, Kevin Wayne**
2. **Slides from class**
3. **https://en.wikipedia.org/wiki/Erd%C5%91s%E2%80%93R%C3%A9nyi_model**