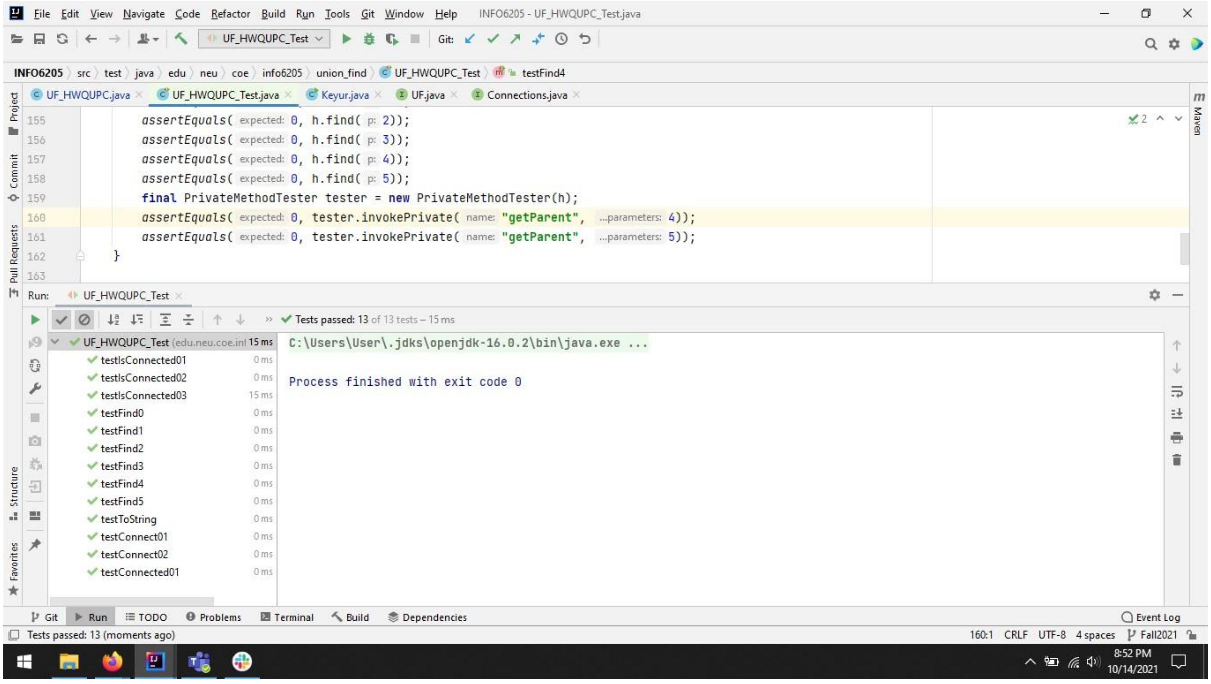
Keyur Ashokbhai Barot (NUID: 001568664)

# Program Structures and Algorithms

## Fall 2021

## Assignment No.3 (WQUPC)

1. Implemented height-weighted Quick Union with Path Compression showing “green” test results.



```
        assertEquals( expected: 0, h.find( p: 2));
        assertEquals( expected: 0, h.find( p: 3));
        assertEquals( expected: 0, h.find( p: 4));
        assertEquals( expected: 0, h.find( p: 5));
        final PrivateMethodTester tester = new PrivateMethodTester(h);
        assertEquals( expected: 0, tester.invokePrivate( name: "getParent",  ...parameters: 4));
        assertEquals( expected: 0, tester.invokePrivate( name: "getParent",  ...parameters: 5));
    }
```

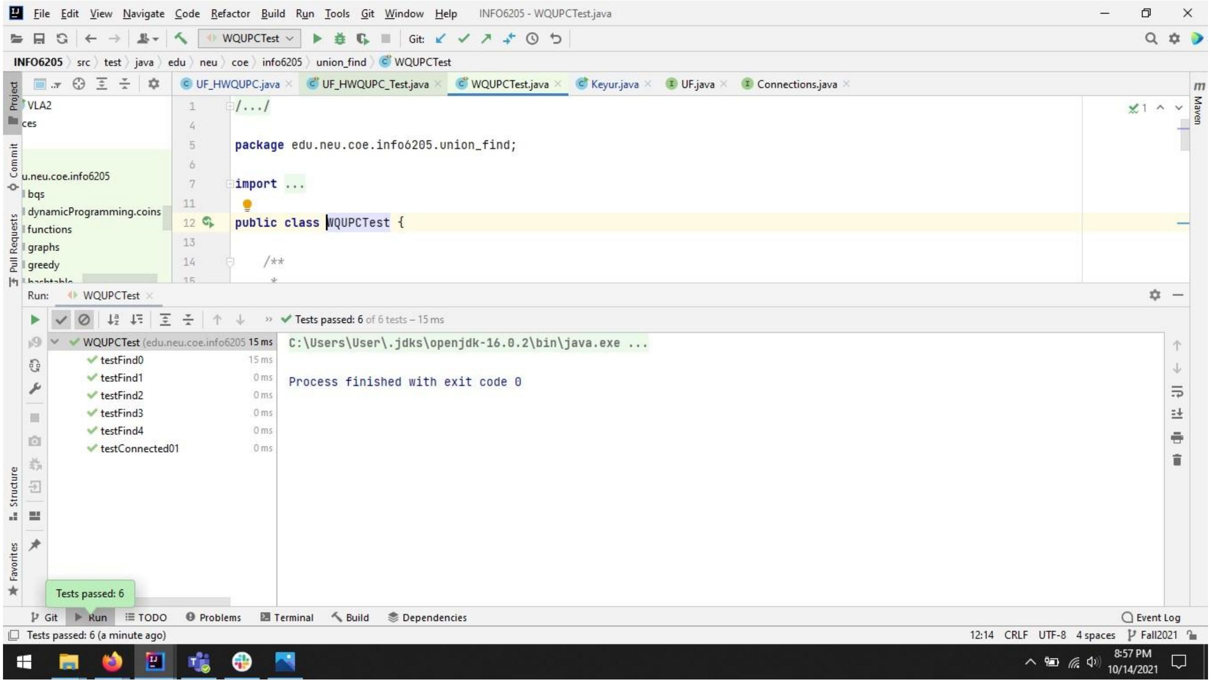
Tests passed: 13 of 13 tests – 15 ms

```
UF_HWQUPC_Test (edu.neu.coe.inf 15 ms
    testIsConnected01   0 ms
    testIsConnected02   0 ms
    testIsConnected03  15 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       0 ms
    testConnected01     0 ms
```

```
C:\Users\User\.jdks\openjdk-16.0.2\bin\java.exe ...

Process finished with exit code 0
```

Tests passed: 13 (moments ago)



```
/.../

package edu.neu.coe.info6205.union_find;

import ...

public class WQUPCTest {

    /**
     *
```

Tests passed: 6 of 6 tests – 15 ms

```
WQUPCTest (edu.neu.coe.info6205 15 ms
    testFind0          15 ms
    testFind1           0 ms
    testFind2           0 ms
    testFind3           0 ms
    testFind4           0 ms
    testConnected01     0 ms
```

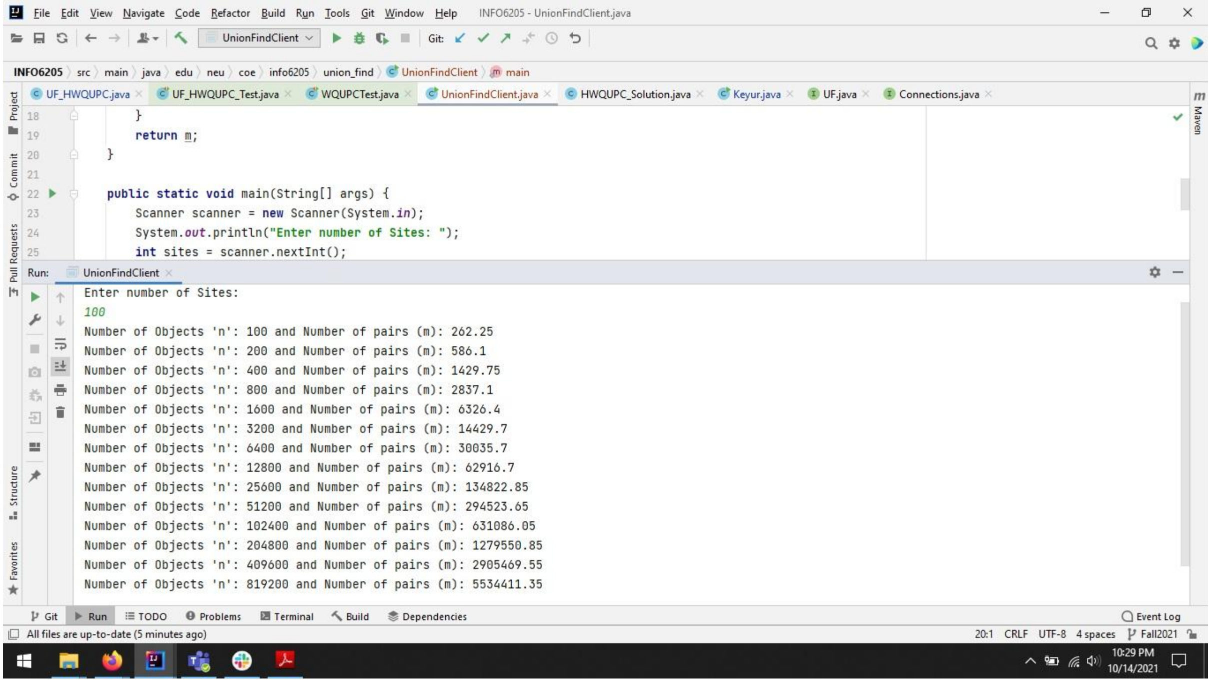
```
C:\Users\User\.jdks\openjdk-16.0.2\bin\java.exe ...

Process finished with exit code 0
```

Tests passed: 6

Tests passed: 6 (a minute ago)

2. Using the implementation of UF_HWQUPC, developed a UnionFindClient function 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.



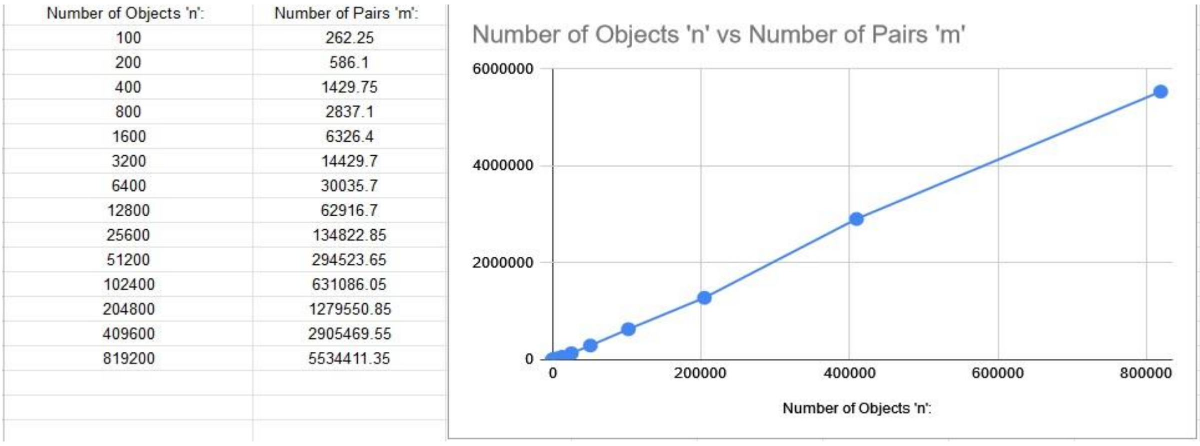
```
        }
        return m;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter number of Sites: ");
        int sites = scanner.nextInt();
```

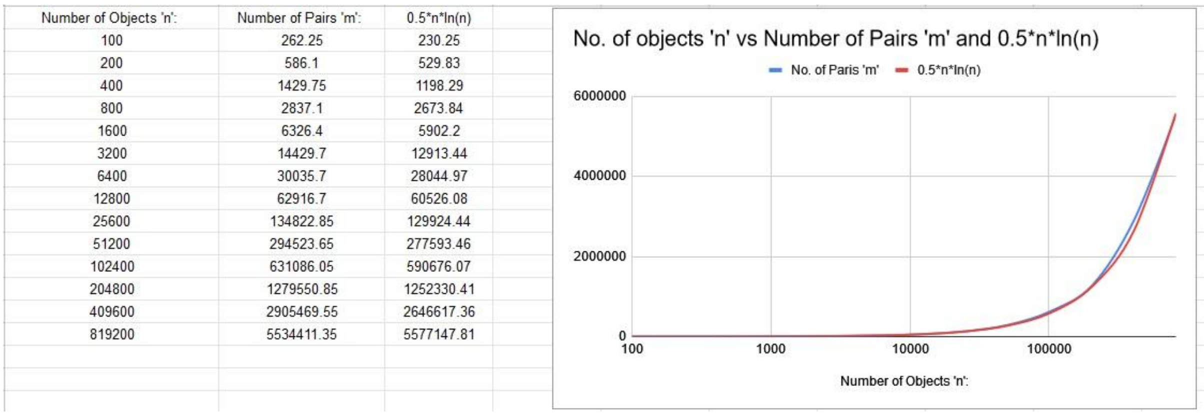
```
Enter number of Sites:
100
Number of Objects 'n': 100 and Number of pairs (m): 262.25
Number of Objects 'n': 200 and Number of pairs (m): 586.1
Number of Objects 'n': 400 and Number of pairs (m): 1429.75
Number of Objects 'n': 800 and Number of pairs (m): 2837.1
Number of Objects 'n': 1600 and Number of pairs (m): 6326.4
Number of Objects 'n': 3200 and Number of pairs (m): 14429.7
Number of Objects 'n': 6400 and Number of pairs (m): 30035.7
Number of Objects 'n': 12800 and Number of pairs (m): 62916.7
Number of Objects 'n': 25600 and Number of pairs (m): 134822.85
Number of Objects 'n': 51200 and Number of pairs (m): 294523.65
Number of Objects 'n': 102400 and Number of pairs (m): 631086.05
Number of Objects 'n': 204800 and Number of pairs (m): 1279550.85
Number of Objects 'n': 409600 and Number of pairs (m): 2905469.55
Number of Objects 'n': 819200 and Number of pairs (m): 5534411.35
```

3. 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) is

=> m = f(n)=0.5*n*ln(n)



| Number of Objects 'n': | Number of Pairs 'm': |
|---|---|
| 100 | 262.25 |
| 200 | 586.1 |
| 400 | 1429.75 |
| 800 | 2837.1 |
| 1600 | 6326.4 |
| 3200 | 14429.7 |
| 6400 | 30035.7 |
| 12800 | 62916.7 |
| 25600 | 134822.85 |
| 51200 | 294523.65 |
| 102400 | 631086.05 |
| 204800 | 1279550.85 |
| 409600 | 2905469.55 |
| 819200 | 5534411.35 |



| Number of Objects 'n': | Number of Pairs 'm': | 0.5*n*ln(n) |
|---|---|---|
| 100 | 262.25 | 230.25 |
| 200 | 586.1 | 529.83 |
| 400 | 1429.75 | 1198.29 |
| 800 | 2837.1 | 2673.84 |
| 1600 | 6326.4 | 5902.2 |
| 3200 | 14429.7 | 12913.44 |
| 6400 | 30035.7 | 28044.97 |
| 12800 | 62916.7 | 60526.08 |
| 25600 | 134822.85 | 129924.44 |
| 51200 | 294523.65 | 277593.46 |
| 102400 | 631086.05 | 590676.07 |
| 204800 | 1279550.85 | 1252330.41 |
| 409600 | 2905469.55 | 2646617.36 |
| 819200 | 5534411.35 | 5577147.81 |