**Venkata Nitin Reddy Byreddy(002191804)**

**INFO 6205 PROGRAM STRUCTURES AND ALGORITHMS**

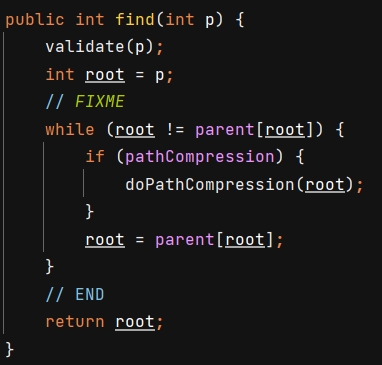
**Assignment3(WQUPC)**

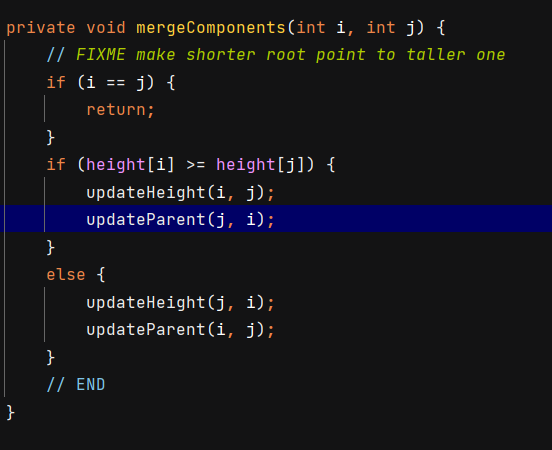
**Step 1:**

1. 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.

**Solution:**

The code can found under the file name UF\_HWQUPC.java.





Text

Description automatically generated

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

**Proof:**

The code can found under the file name UF\_HWQUPC\_Test.java.

**A screenshot of a computer

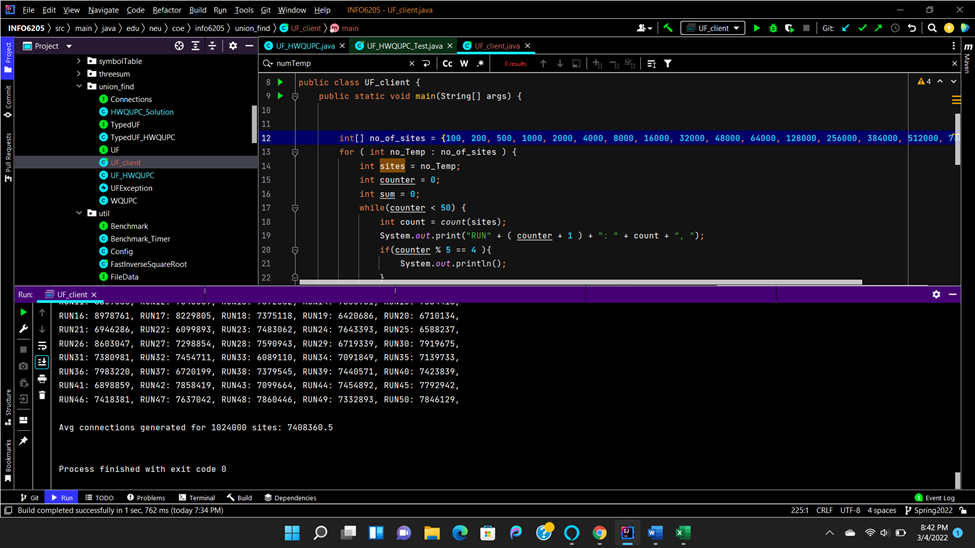
Description automatically generated**

**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).

**Solution:**

The code can be found under the name UF\_client.java.



**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.

**Solution:**

Total 15 sites have been ran to determine the relation between number of objects(n) and number of pairs(m). 50 runs were conducted for each number of objects(n) to get average connections.

**Conclusion and Relationship:**

After calculating the relationship between the number of objects(n) and number of pairs (m), I found the relation based on graph. So after trying different methods, I can conclude approximately that

**m = ½\*n\*log2n**

**Evidence(Graph):**

**Relationship between m and n:**

**Chart

Description automatically generated**

**Relationship between m and ½\*n\*logn:**

**Chart

Description automatically generated**

So the final relation is **m=1/2\*n\*logn.**