Program Structures & Algorithms
Spring 2022
Assignment No. 3

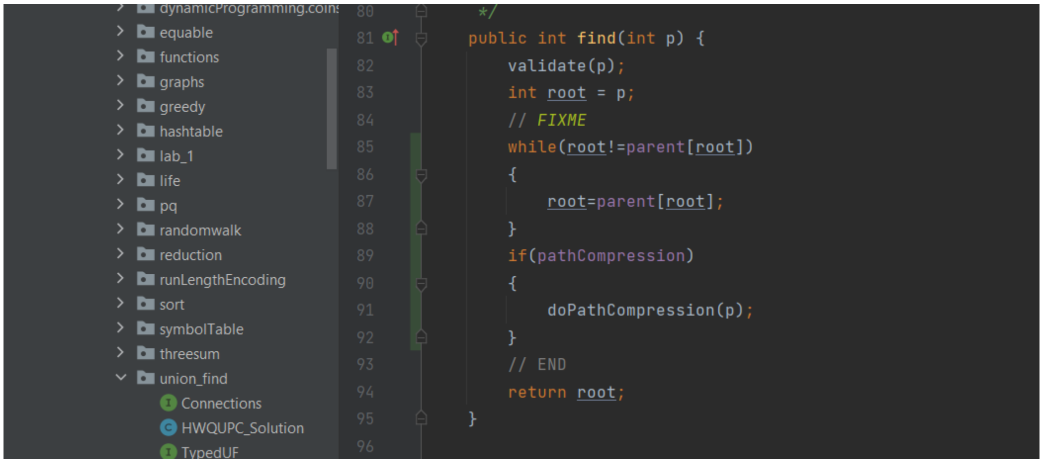
Name: Shashank Siripragada
NUID: 002193773

Task

Step 1:

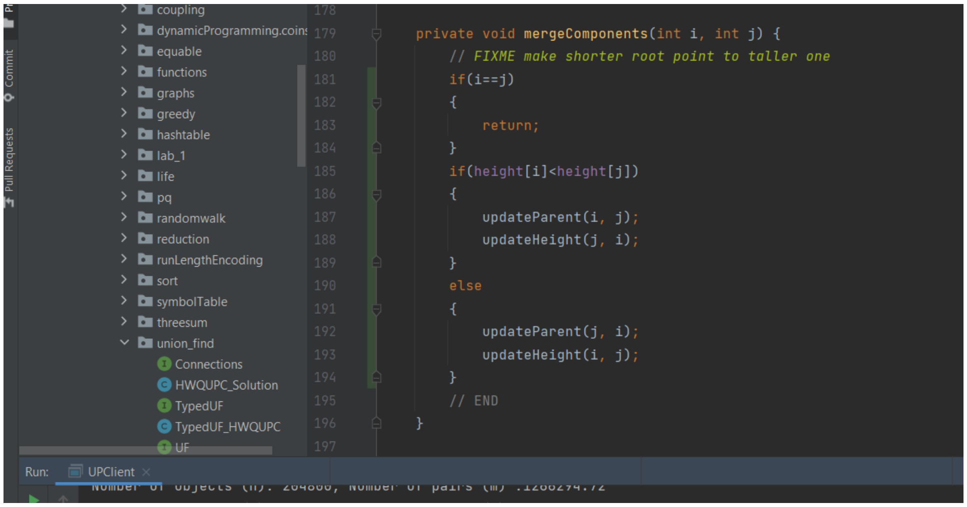
(a) Implement height-weighted Quick Union with Path Compression.

Find:



```
     */
    public int find(int p) {
        validate(p);
        int root = p;
        // FIXME
        while(root!=parent[root])
        {
            root=parent[root];
        }
        if(pathCompression)
        {
            doPathCompression(p);
        }
        // END
        return root;
    }
```

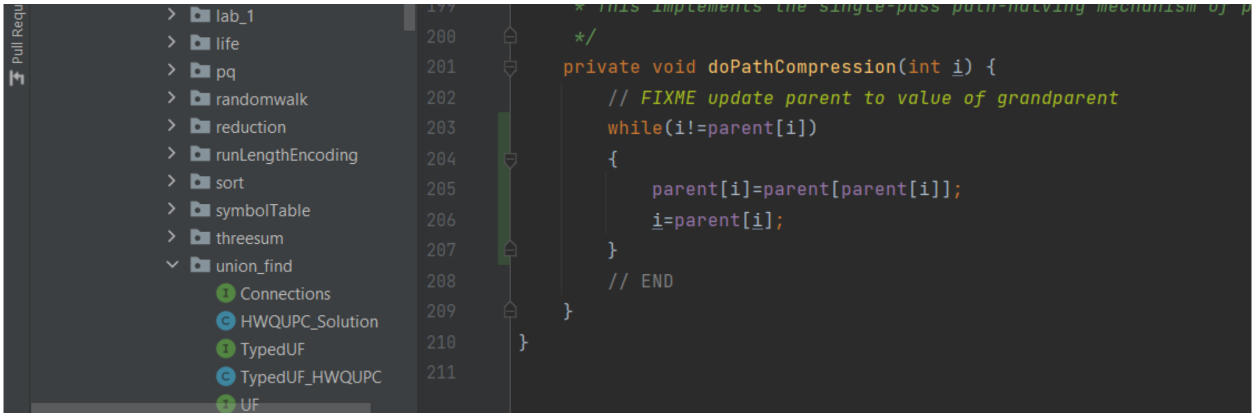
mergeComponents:



```
    private void mergeComponents(int i, int j) {
        // FIXME make shorter root point to taller one
        if(i==j)
        {
            return;
        }
        if(height[i]<height[j])
        {
            updateParent(i, j);
            updateHeight(j, i);
        }
        else
        {
            updateParent(j, i);
            updateHeight(i, j);
        }
        // END
    }
```

```
Number of objects (n): 204800, Number of pairs (m) :1268294.72
```

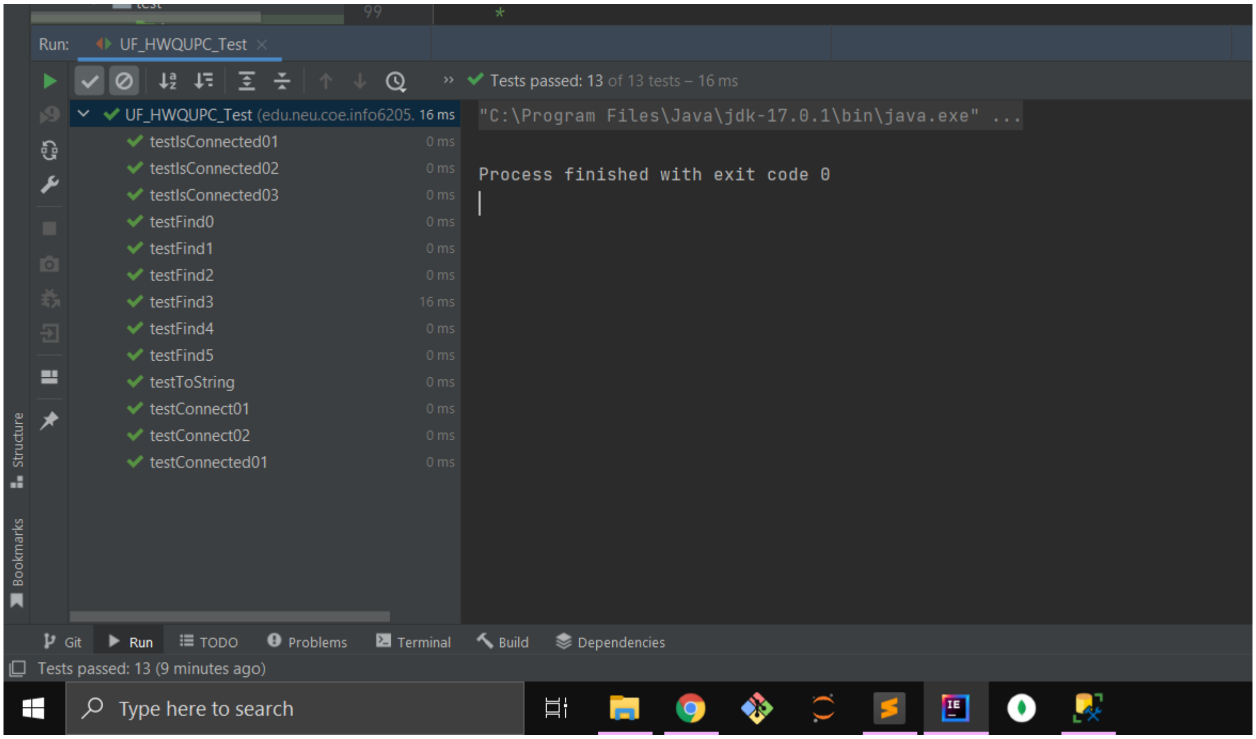
pathCompression:



```
     * This implements the single-pass path-halving mechanism of p
     */
    private void doPathCompression(int i) {
        // FIXME update parent to value of grandparent
        while(i!=parent[i])
        {
            parent[i]=parent[parent[i]];
            i=parent[i];
        }
        // END
    }
}
```

(b) Check that the unit tests for this class all work.

UF_HWQUPC_Test

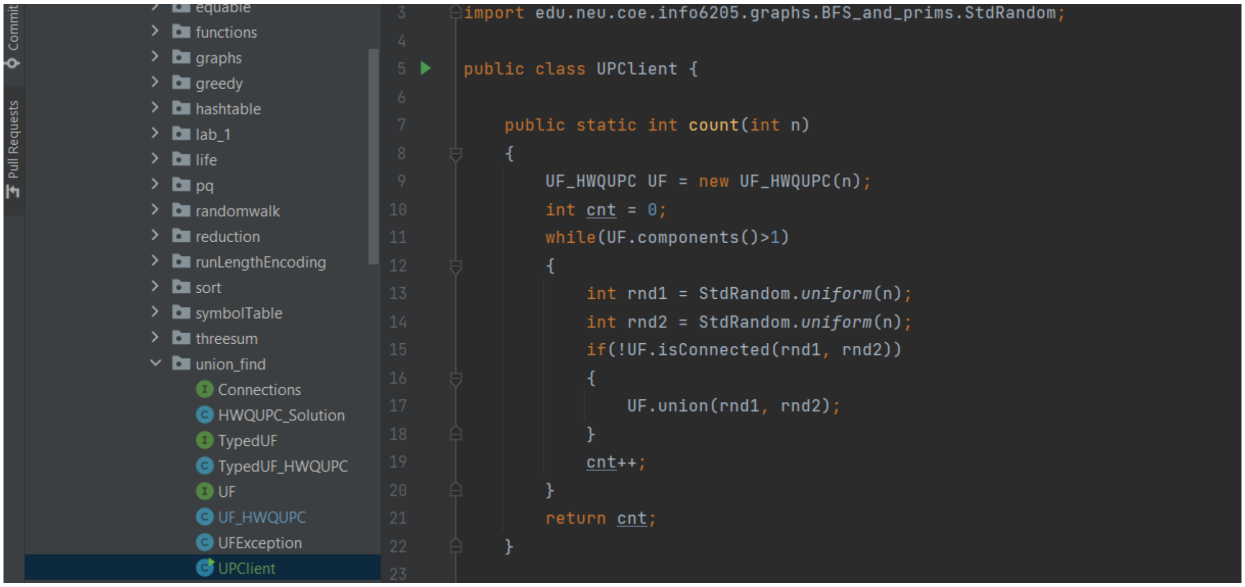


```
Tests passed: 13 of 13 tests – 16 ms

UF_HWQUPC_Test (edu.neu.coe.info6205. 16 ms
    testIsConnected01    0 ms
    testIsConnected02    0 ms
    testIsConnected03    0 ms
    testFind0            0 ms
    testFind1            0 ms
    testFind2            0 ms
    testFind3           16 ms
    testFind4            0 ms
    testFind5            0 ms
    testToString         0 ms
    testConnect01        0 ms
    testConnect02        0 ms
    testConnected01      0 ms

"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...

Process finished with exit code 0

Tests passed: 13 (9 minutes ago)
```

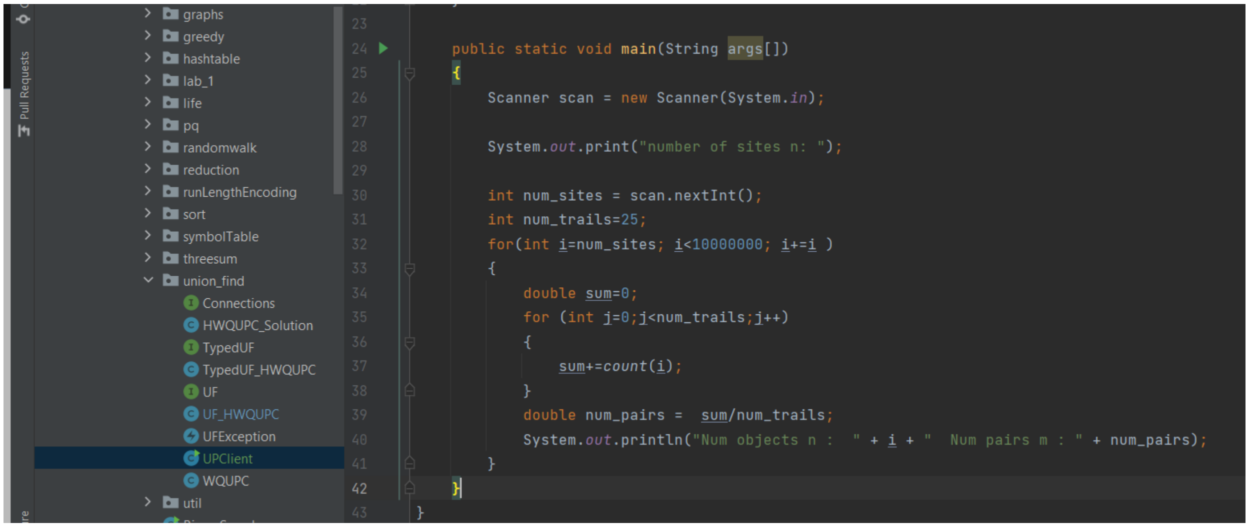
Step 2:

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."

Implemented UPClient

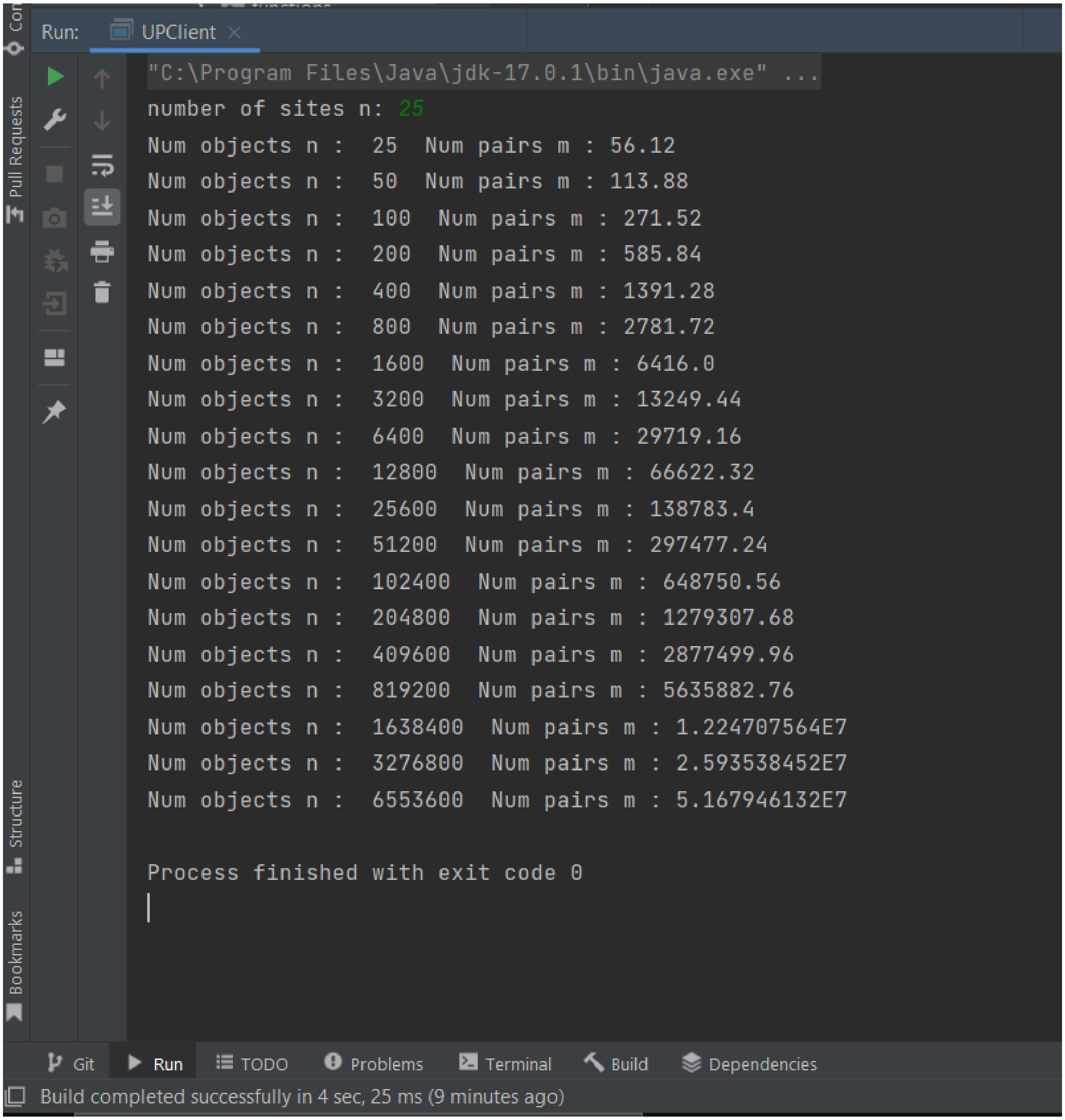


```java
import edu.neu.coe.info6205.graphs.BFS_and_prims.StdRandom;

public class UPClient {

    public static int count(int n)
    {
        UF_HWQUPC UF = new UF_HWQUPC(n);
        int cnt = 0;
        while(UF.components()>1)
        {
            int rnd1 = StdRandom.uniform(n);
            int rnd2 = StdRandom.uniform(n);
            if(!UF.isConnected(rnd1, rnd2))
            {
                UF.union(rnd1, rnd2);
            }
            cnt++;
        }
        return cnt;
    }

```



```java

    public static void main(String args[])
    {
        Scanner scan = new Scanner(System.in);

        System.out.print("number of sites n: ");

        int num_sites = scan.nextInt();
        int num_trails=25;
        for(int i=num_sites; i<10000000; i+=i )
        {
            double sum=0;
            for (int j=0;j<num_trails;j++)
            {
                sum+=count(i);
            }
            double num_pairs =  sum/num_trails;
            System.out.println("Num objects n :  " + i + "  Num pairs m : " + num_pairs);
        }
    }
}
```

Run Results:



```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...
number of sites n: 25
Num objects n :  25  Num pairs m : 56.12
Num objects n :  50  Num pairs m : 113.88
Num objects n :  100  Num pairs m : 271.52
Num objects n :  200  Num pairs m : 585.84
Num objects n :  400  Num pairs m : 1391.28
Num objects n :  800  Num pairs m : 2781.72
Num objects n :  1600  Num pairs m : 6416.0
Num objects n :  3200  Num pairs m : 13249.44
Num objects n :  6400  Num pairs m : 29719.16
Num objects n :  12800  Num pairs m : 66622.32
Num objects n :  25600  Num pairs m : 138783.4
Num objects n :  51200  Num pairs m : 297477.24
Num objects n :  102400  Num pairs m : 648750.56
Num objects n :  204800  Num pairs m : 1279307.68
Num objects n :  409600  Num pairs m : 2877499.96
Num objects n :  819200  Num pairs m : 5635882.76
Num objects n :  1638400  Num pairs m : 1.224707564E7
Num objects n :  3276800  Num pairs m : 2.593538452E7
Num objects n :  6553600  Num pairs m : 5.167946132E7

Process finished with exit code 0
```

Build completed successfully in 4 sec, 25 ms (9 minutes ago)

Step 3:

Determine the relationship between the number of objects (n) and the number of pairs (m).

Relationship:
The relationship between the number of objects (n) and the number of pairs (m) generated to reduce the number of components from n to 1 is:

$$m = f(n) = \mathbf{1/2} \times n \times \ln(n)$$

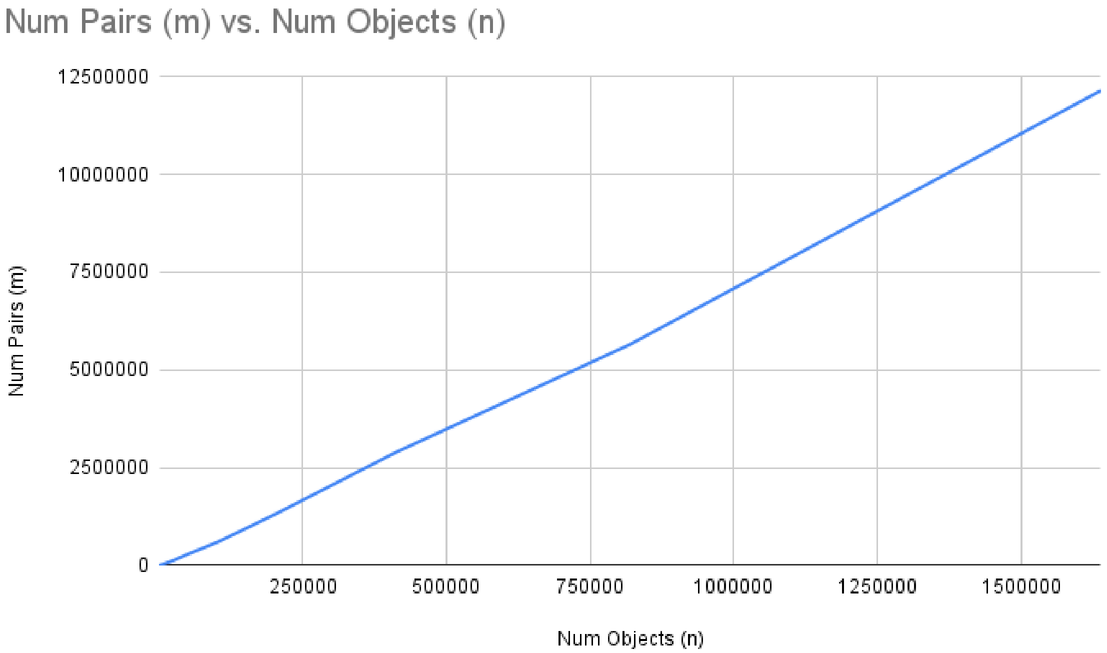
Where, m = number of pairs, n = number of objects

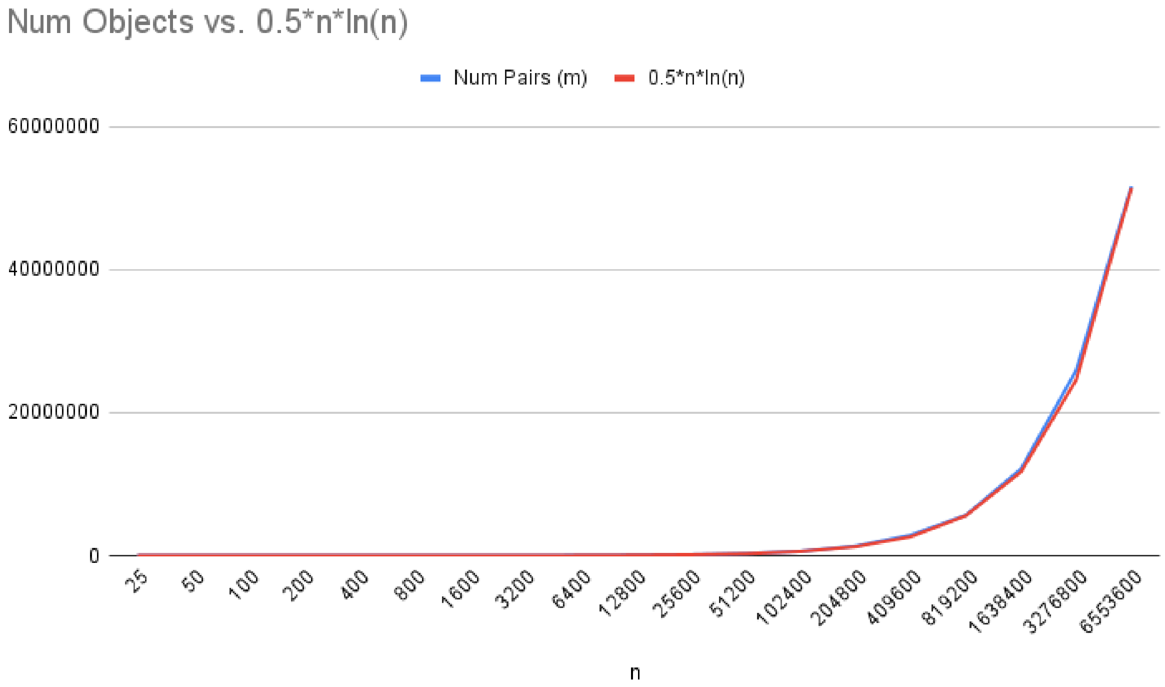
Evidence:
With the initial value of n as 25 and the doubling method, we can calculate the num pairs (m) generated to reduce the number of components from n to 1.

For the values of n, we can see that the average number of pairs needed to reduce the component 1 follows pretty closely to ½ * × n × ln(n), although not equal.

Num Objects (n)	Num Pairs (m)	0.5*n*ln(n)
25	57.2	40.23594781
50	116.16	97.80057514
100	272.28	230.2585093
200	550.92	529.8317367
400	1323.92	1198.292909
800	2671.76	2673.844691
1600	6121.36	5902.207127
3200	14588.72	12913.44974
6400	31000.96	28044.97046
12800	63623.04	60526.08288
25600	134619.16	129924.4497
51200	297053.76	277593.4672
102400	601920.52	590676.07
204800	1331145.48	1252330.411
409600	2877144.36	2646617.365
819200	5652306.36	5577147.815
1638400	1.21E+07	11722121.8
3276800	2.59E+07	24579895.94
6553600	5.17E+07	51431096.57

The below diagrams show the result of plotting the above table data, with the number of objects n on the Xaxis and number of pairs m generated to reduce the number of components from n to 1 on the Yaxis.


Num Pairs (m) vs. Num Objects (n)
Num Pairs (m)
Num Objects (n)


Num Objects vs. 0.5*n*ln(n)
Num Pairs (m)
0.5*n*ln(n)
n