

Jiao Gong(001561450)
Program Structures & Algorithms
Fall2021
Assignment No.3

Task(WQUPC)

Your task is

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.

NOTE: although I'm not going to tell you in advance what the relationship is, I can assure you that it is a *simple* relationship.

Don't forget to follow the submission guidelines. And to use sufficient (and sufficiently large) different values of n .

Step1

UF_HWQUPC

a. Code

https://github.com/slowpeace2020/INFO6205/blob/Fall2021/src/main/java/edu/neu/coe/info6205/union_find/UF_HWQUPC.java

b. Unit test

```
1  /.../  
4  
5  package edu.neu.coe.info6205.uni  
6  
7  import ...  
11  
12 public class UF_HWQUPC_Test {  
13  
14     @Test  
15     public void testToToString() {  
16         Connections h = new UF_H  
17         assertEquals("expected: \"U  
18             \" count: 2\\n\" +  
19             \" path compress  
20             \" parents: [0,  
21             \" heights: [1,  
22     }  
23
```

Run: UF_HWQUPC_Test.testToToString x

Tests passed: 1 of 1 test – 3 ms

UF_HWQUPC_3 ms

testToStrin 3 ms

/Library/Java/JavaVirtualMachines/jdk-11.0.2.jdk

Process finished with exit code 0

(grap1. UF_HWQUPC_Test method testToToString)

```
23  
24  /**  
25  *  
26  */  
27  @Test  
28  public void testIsConnected01()  
29      Connections h = new UF_HWQU  
30      assertFalse(h.isConnected(  
31  }  
32  
33  /**  
34  *  
35  */  
36  @Test(expected = IllegalArgument  
37  public void testIsConnected02()  
38      Connections h = new UF_HWQU  
39      assertTrue(h.isConnected(  
40
```

Run: UF_HWQUPC_Test.testIsConnected01 x

Tests passed: 1 of 1 test – 5 ms

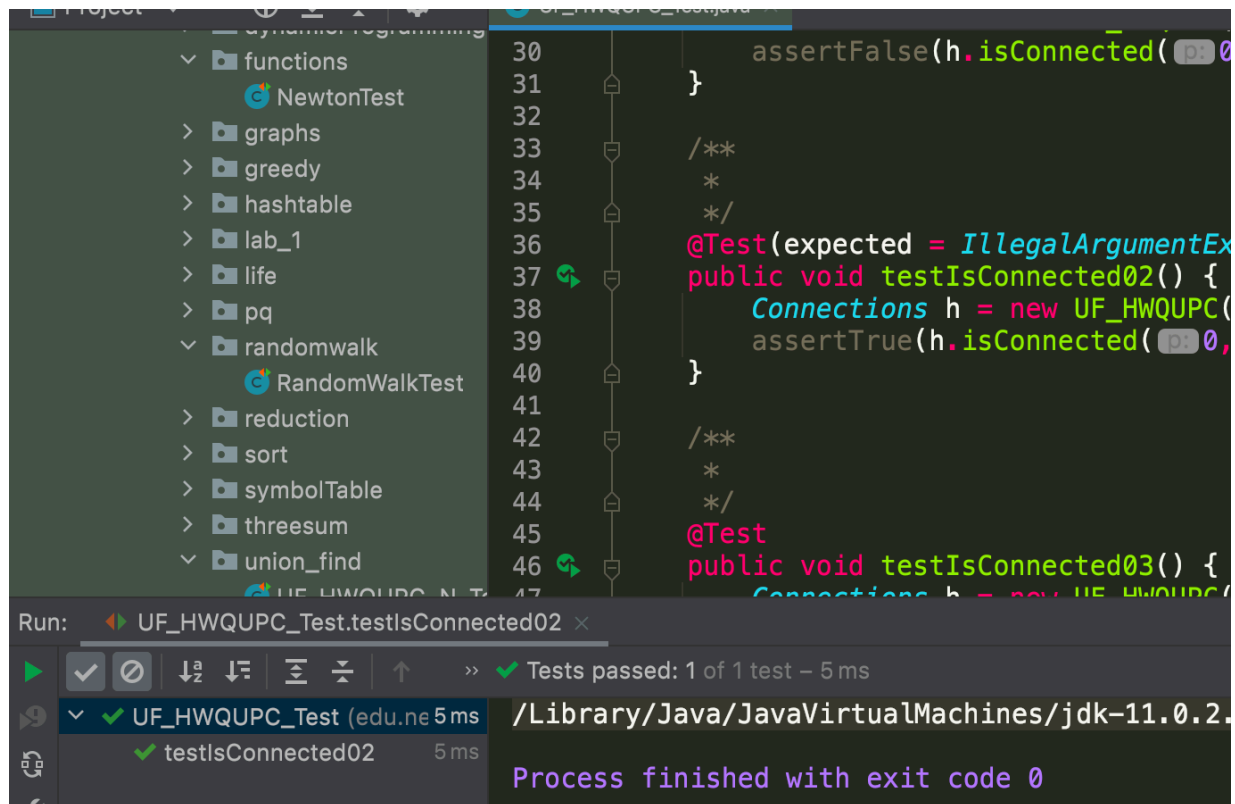
UF_HWQUPC_Test (edu.ne 5 ms

testIsConnected01 5 ms

/Library/Java/JavaVirtualMachines/jdk-11.0

Process finished with exit code 0

(graph 2. UF_HWQUPC_Test method testIsConnected01)



The screenshot displays an IDE with a project explorer on the left, a code editor in the center, and a run console at the bottom. The project explorer shows a directory structure with folders like 'functions', 'graphs', 'greedy', 'hashtable', 'lab_1', 'life', 'pq', 'randomwalk', 'reduction', 'sort', 'symbolTable', 'threesum', and 'union_find'. The code editor shows the implementation of the `UF_HWQUPC_Test` class, including methods `testIsConnected01`, `testIsConnected02`, and `testIsConnected03`. The run console shows the execution of `UF_HWQUPC_Test.testIsConnected02`, which passed successfully in 5 ms. The console also displays the Java version `/Library/Java/JavaVirtualMachines/jdk-11.0.2` and the exit code `0`.

```
30         assertFalse(h.isConnected(0, 0));
31     }
32
33     /**
34     *
35     */
36     @Test(expected = IllegalArgumentException.class)
37     public void testIsConnected02() {
38         Connections h = new UF_HWQUPC(10);
39         assertTrue(h.isConnected(0, 0));
40     }
41
42     /**
43     *
44     */
45     @Test
46     public void testIsConnected03() {
47         Connections h = new UF_HWQUPC(10);
```

Run: UF_HWQUPC_Test.testIsConnected02 x

Tests passed: 1 of 1 test – 5 ms

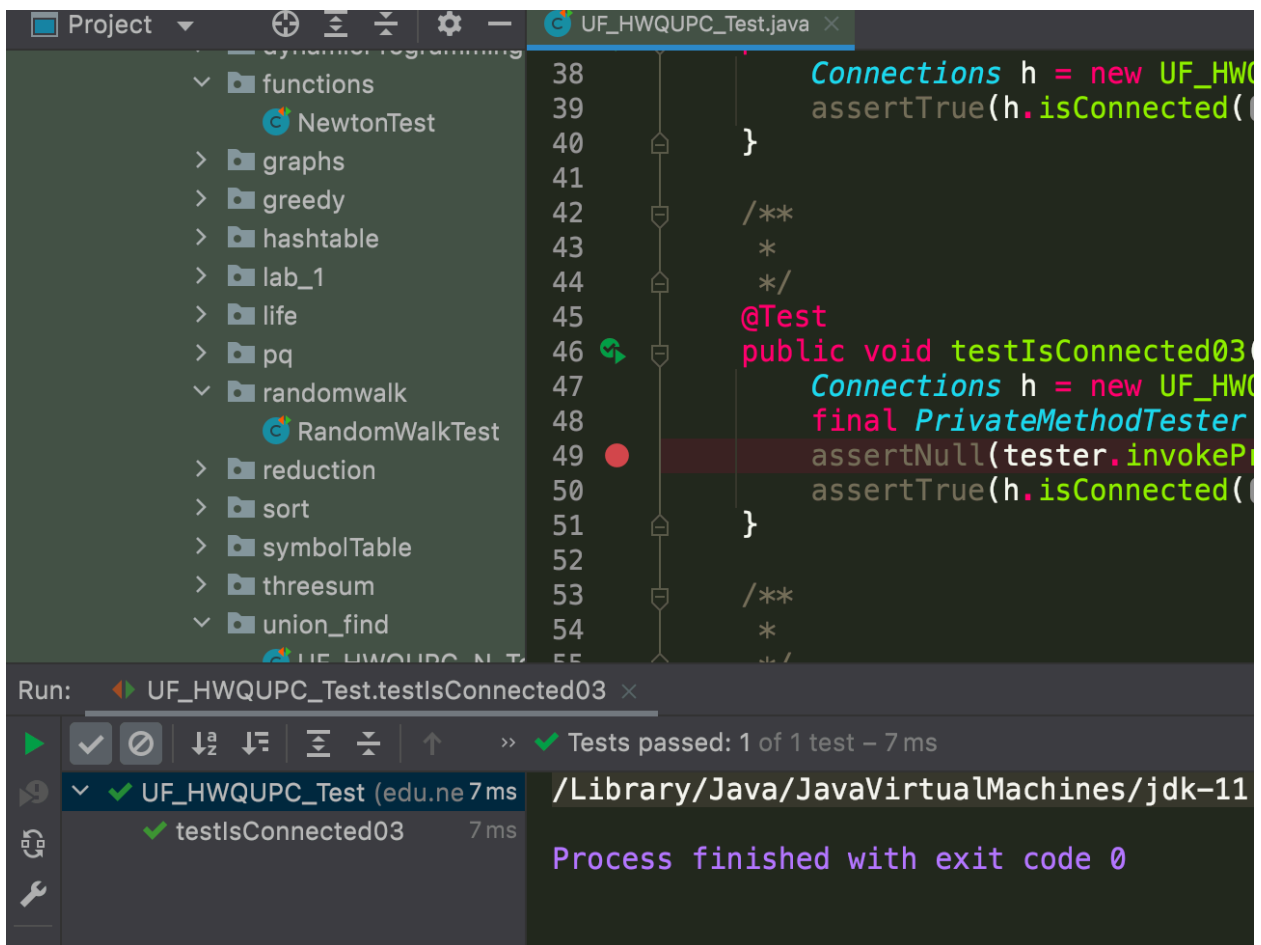
UF_HWQUPC_Test (edu.ne 5 ms)

testIsConnected02 5 ms

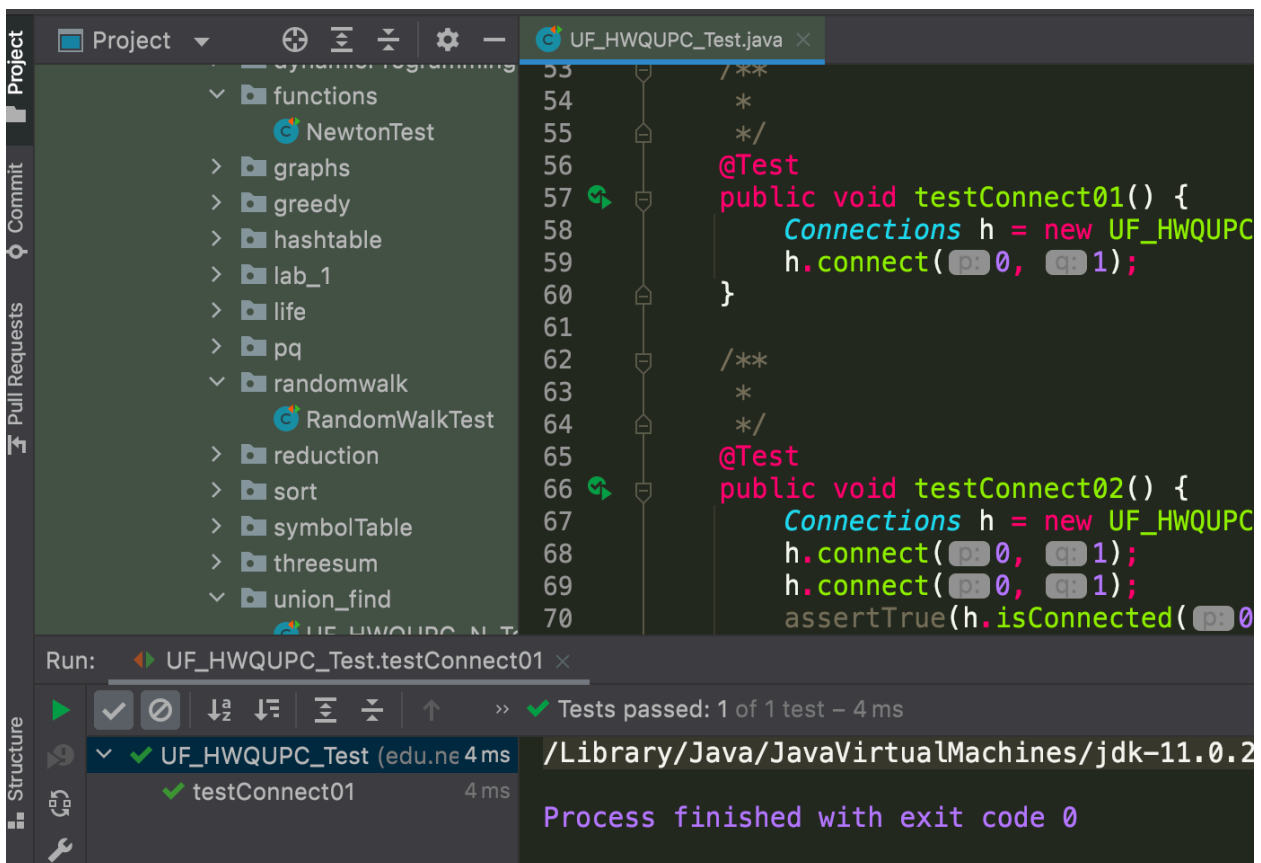
/Library/Java/JavaVirtualMachines/jdk-11.0.2

Process finished with exit code 0

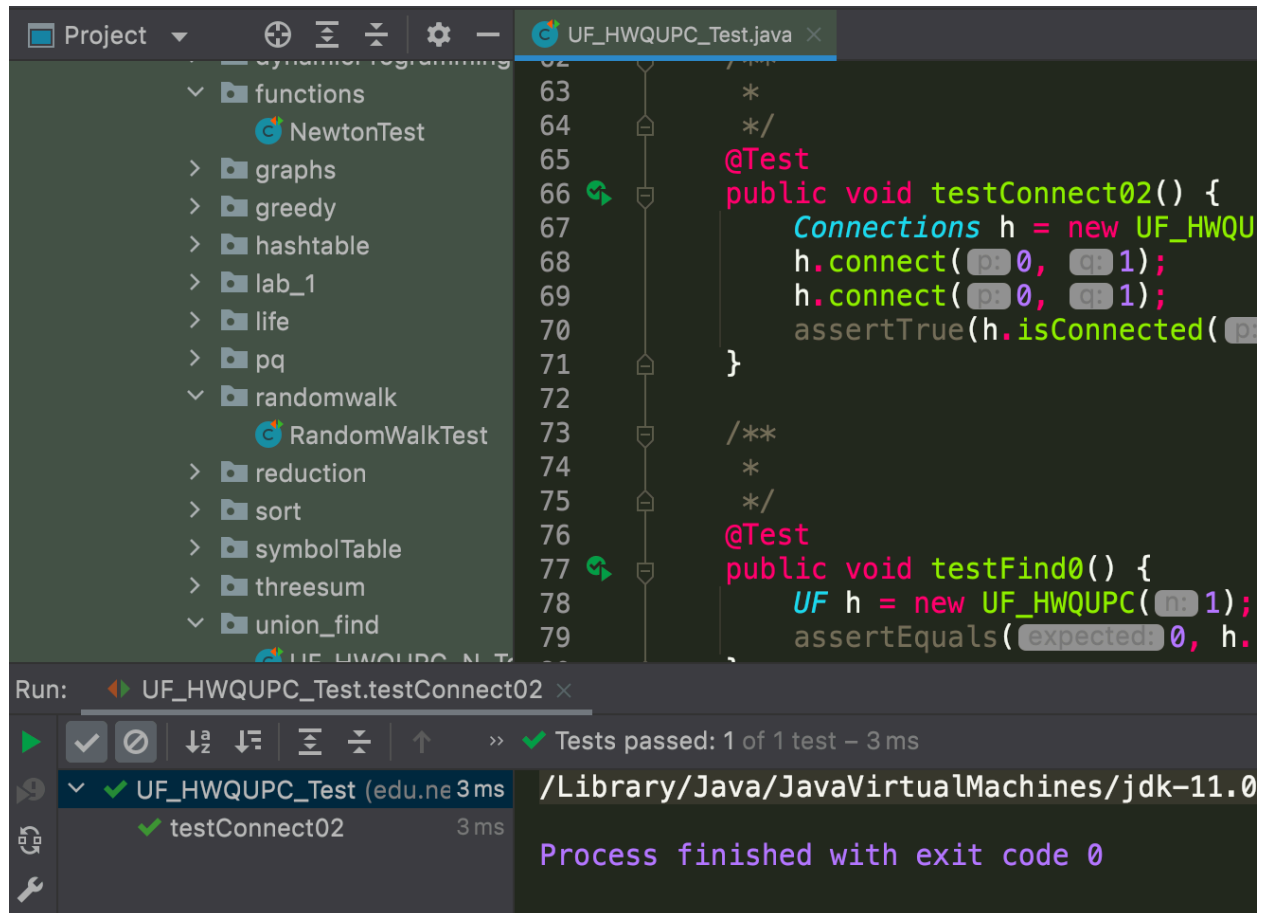
(graph 3. UF_HWQUPC_Test method testIsConnected02)



(graph 4. UF_HWQUPC_Test method testIsConnected03)



(graph 5. UF_HWQUPC_Test method testConnect01)



(graph 6. UF_HWQUPC_Test method testConnect02)

```
72
73 /**
74  *
75  */
76 @Test
77 public void testFind0() {
78     UF h = new UF_HWQUPC(
79         assertEquals(expected:
80     }
81
82 /**
83  *
84  */
85 @Test
86 public void testFind1() {
87     UF h = new UF_HWQUPC(
88     h.connect(p: 0, q: 1)
89     assertEquals(expected:

```

Run: UF_HWQUPC_Test.testFind0

Tests passed: 1 of 1 test – 10 ms

UF_HWQUPC_Test (edu.n 10 ms)

testFind0 10 ms

/Library/Java/JavaVirtualMachines/jd

Process finished with exit code 0

(graph 7. UF_HWQUPC_Test method testFind0)

```
80 }
81
82 /**
83  *
84  */
85 @Test
86 public void testFind1() {
87     UF h = new UF_HWQUPC(
88     h.connect(p: 0, q: 1)
89     assertEquals(expected:
90     assertEquals(expected:
91 }
92
93 /**
94  *
95  */
96 @Test
97 public void testFind2() {

```

Run: UF_HWQUPC_Test.testFind1

Tests passed: 1 of 1 test – 5 ms

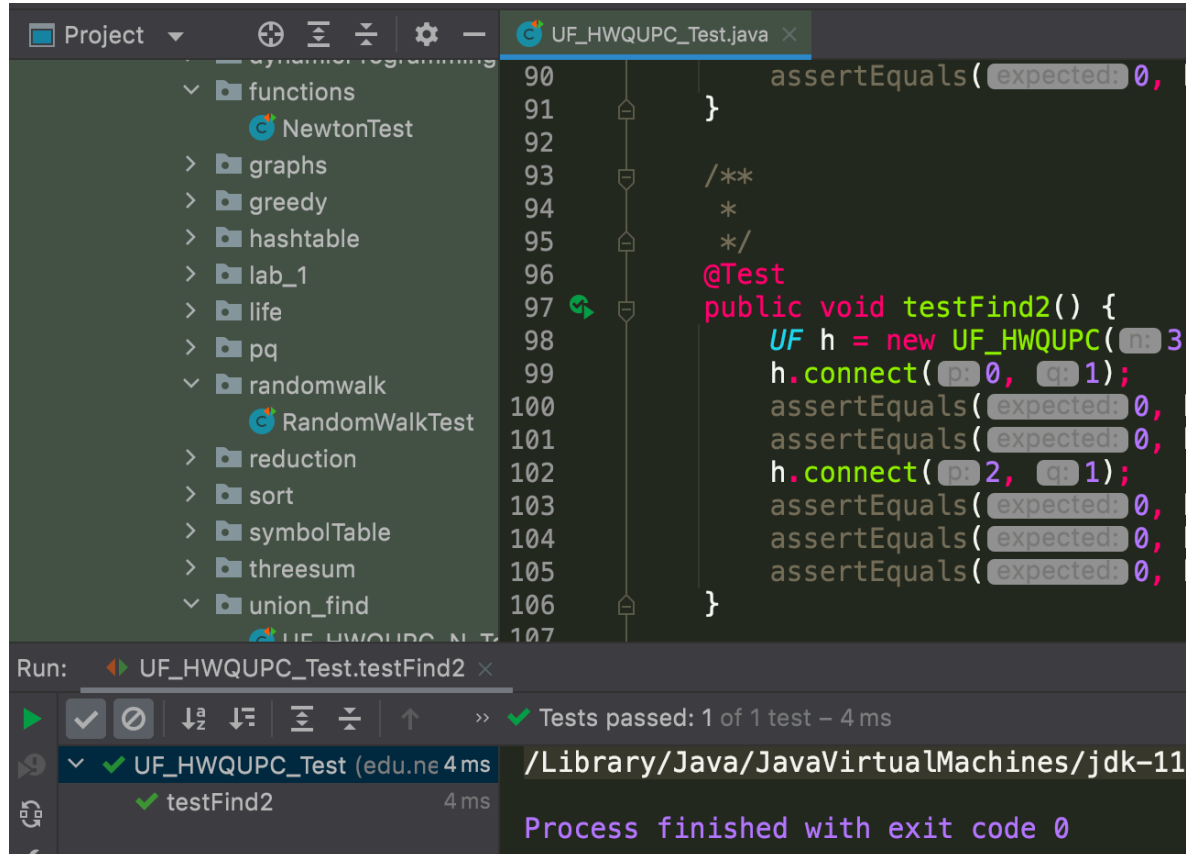
UF_HWQUPC_Test (edu.ne 5 ms)

testFind1 5 ms

/Library/Java/JavaVirtualMachines/jd

Process finished with exit code 0

(graph 8. UF_HWQUPC_Test method testFind1)



```
90      assertEquals(expected: 0,
91      }
92
93      /**
94      *
95      */
96      @Test
97      public void testFind2() {
98          UF h = new UF_HWQUPC(n: 3
99          h.connect(p: 0, q: 1);
100          assertEquals(expected: 0,
101          assertEquals(expected: 0,
102          h.connect(p: 2, q: 1);
103          assertEquals(expected: 0,
104          assertEquals(expected: 0,
105          assertEquals(expected: 0,
106      }
107
```

Run: UF_HWQUPC_Test.testFind2

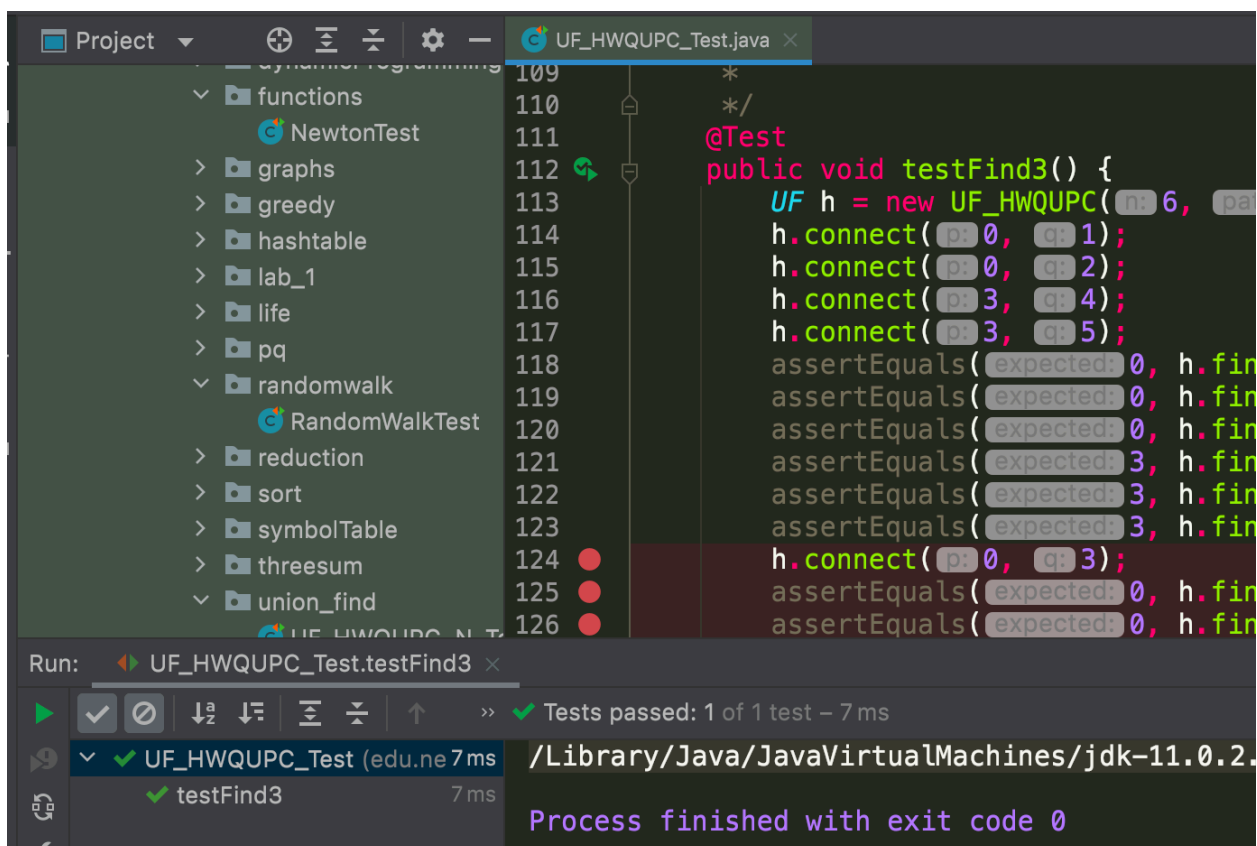
Tests passed: 1 of 1 test – 4 ms

UF_HWQUPC_Test (edu.ne 4 ms) /Library/Java/JavaVirtualMachines/jdk-11

testFind2 4 ms

Process finished with exit code 0

(graph 9. UF_HWQUPC_Test method testFind2)



```
109      *
110      */
111      @Test
112      public void testFind3() {
113          UF h = new UF_HWQUPC(n: 6, pa
114          h.connect(p: 0, q: 1);
115          h.connect(p: 0, q: 2);
116          h.connect(p: 3, q: 4);
117          h.connect(p: 3, q: 5);
118          assertEquals(expected: 0, h.fin
119          assertEquals(expected: 0, h.fin
120          assertEquals(expected: 0, h.fin
121          assertEquals(expected: 3, h.fin
122          assertEquals(expected: 3, h.fin
123          assertEquals(expected: 3, h.fin
124          h.connect(p: 0, q: 3);
125          assertEquals(expected: 0, h.fin
126          assertEquals(expected: 0, h.fin
```

Run: UF_HWQUPC_Test.testFind3

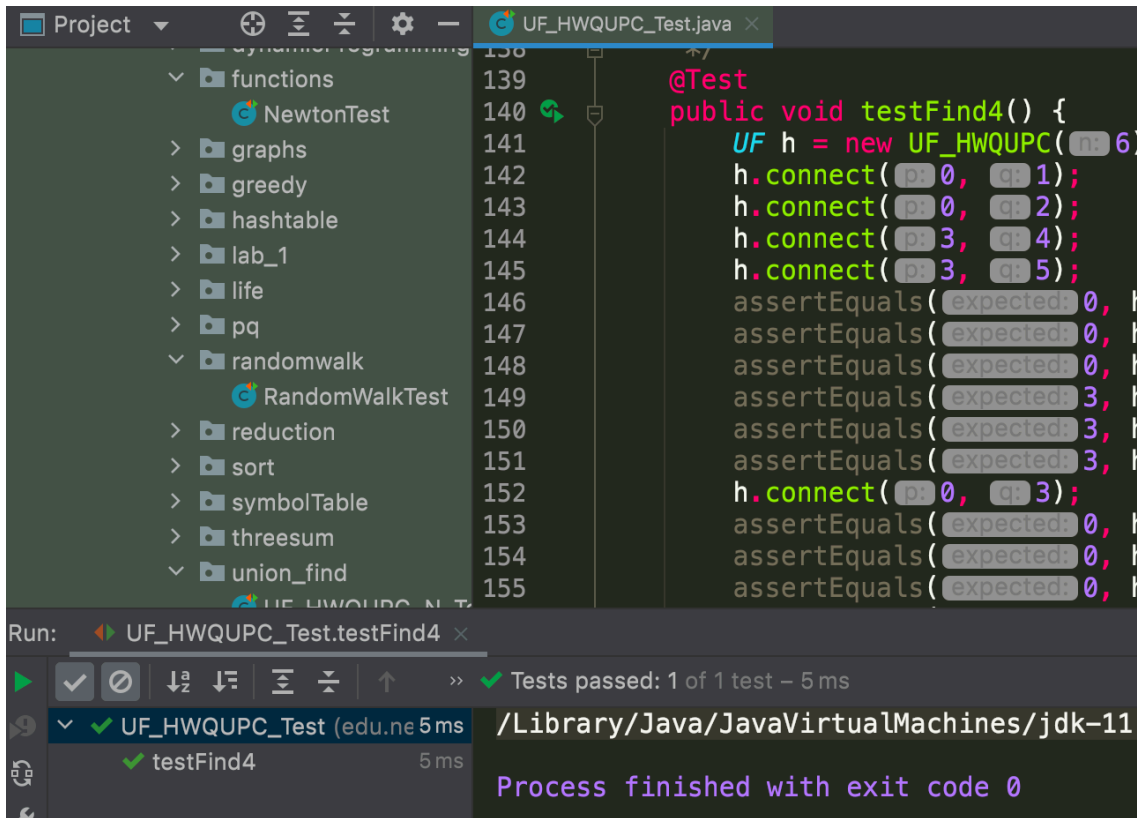
Tests passed: 1 of 1 test – 7 ms

UF_HWQUPC_Test (edu.ne 7 ms) /Library/Java/JavaVirtualMachines/jdk-11.0.2.

testFind3 7 ms

Process finished with exit code 0

(graph 10. UF_HWQUPC_Test method testFind3)



```
139 @Test
140 public void testFind4() {
141     UF h = new UF_HWQUPC(6);
142     h.connect(0, 1);
143     h.connect(0, 2);
144     h.connect(3, 4);
145     h.connect(3, 5);
146     assertEquals(expected: 0, h.find(0));
147     assertEquals(expected: 0, h.find(1));
148     assertEquals(expected: 3, h.find(2));
149     assertEquals(expected: 3, h.find(3));
150     assertEquals(expected: 3, h.find(4));
151     assertEquals(expected: 3, h.find(5));
152     h.connect(0, 3);
153     assertEquals(expected: 0, h.find(0));
154     assertEquals(expected: 0, h.find(1));
155     assertEquals(expected: 0, h.find(2));
156 }
```

Run: UF_HWQUPC_Test.testFind4

Tests passed: 1 of 1 test – 5 ms

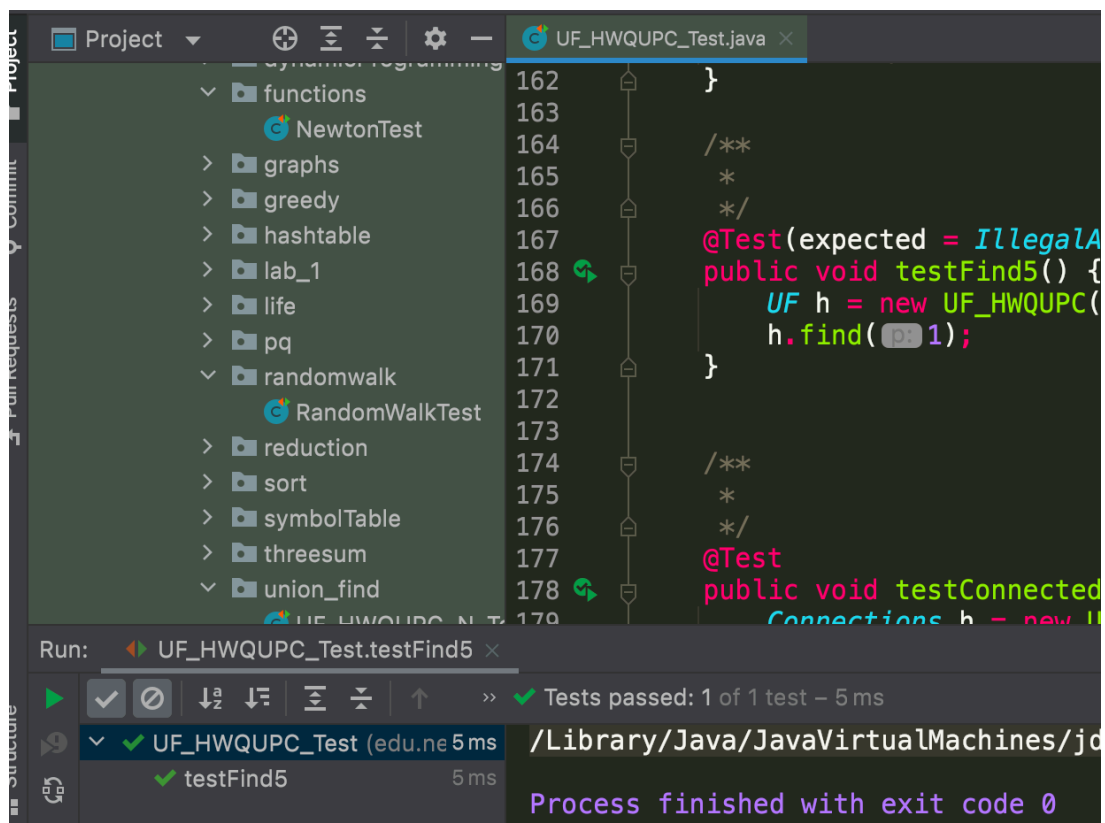
UF_HWQUPC_Test (edu.ne 5 ms)

testFind4 5 ms

/Library/Java/JavaVirtualMachines/jdk-11

Process finished with exit code 0

(graph 11. UF_HWQUPC_Test method testFind4)



```
162 }
163
164 /**
165  *
166  */
167 @Test(expected = IllegalArgumentException.class)
168 public void testFind5() {
169     UF h = new UF_HWQUPC(1);
170     h.find(1);
171 }
172
173 /**
174  *
175  */
176
177 @Test
178 public void testConnected() {
179     Connections h = new UF_HWQUPC(6);
180 }
```

Run: UF_HWQUPC_Test.testFind5

Tests passed: 1 of 1 test – 5 ms

UF_HWQUPC_Test (edu.ne 5 ms)

testFind5 5 ms

/Library/Java/JavaVirtualMachines/jdk-11

Process finished with exit code 0

(graph 12. UF_HWQUPC_Test method testFind5)

The screenshot shows an IDE with a project explorer on the left and a code editor on the right. The project explorer lists a directory structure with folders like 'functions', 'graphs', 'greedy', 'hashtable', 'lab_1', 'life', 'pq', 'randomwalk', 'reduction', 'sort', 'symbolTable', 'threesum', and 'union_find'. The code editor displays the 'UF_HWQUPC_Test.java' file with the following code:

```
171 }
172
173
174 /**
175  *
176  */
177 @Test
178 public void testConnect
179     Connections h = new
180     // h.show();
181     assertFalse(h.isCon
182 }
183 }
```

Below the code editor, the 'Run' tab shows the execution of 'UF_HWQUPC_Test.testConnected01'. The test passed, and the output window displays the following message:

```
Tests passed: 1 of 1 test - 5 ms
UF_HWQUPC_Test (edu.ne 5 ms)
testConnected01 5 ms
Process finished with exit code 0
```

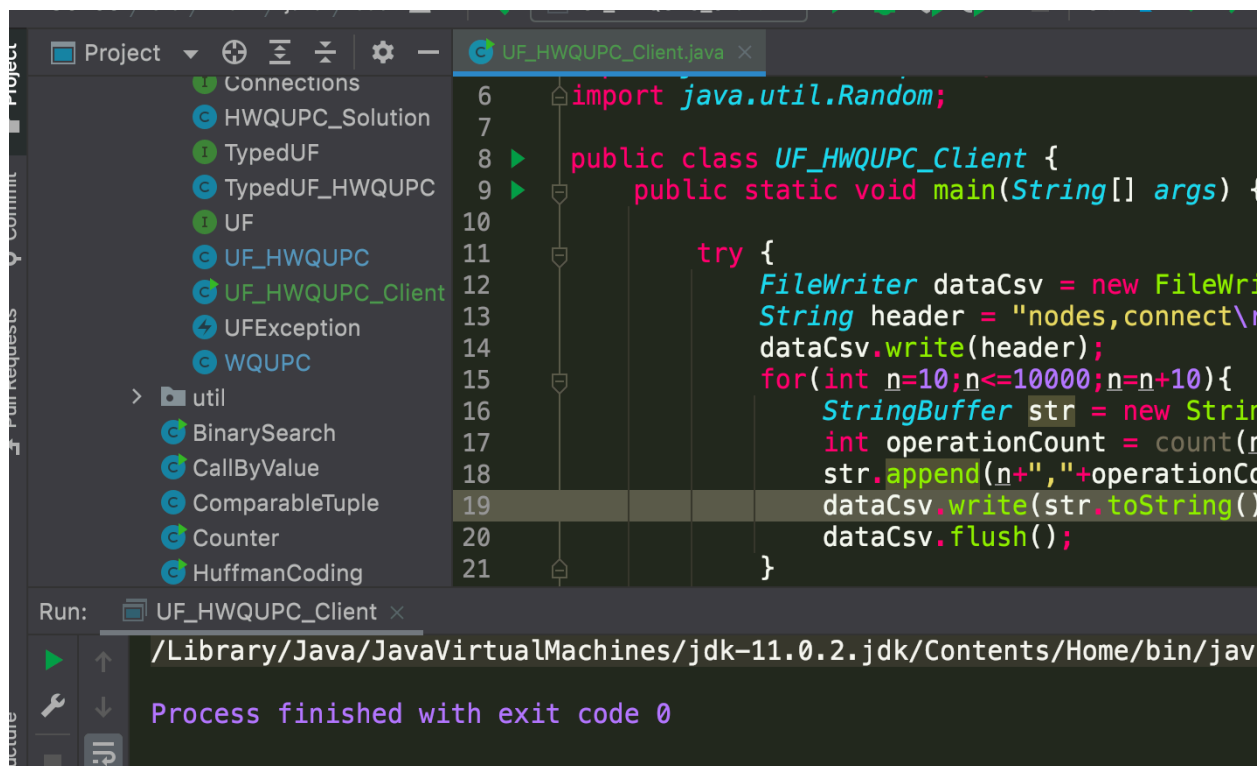
(graph 13. UF_HWQUPC_Test method testConnected01)

Step2

Implementation of UF_HWQUPC, I created a main program that doesn't require any input and runs the experiment for a fixed set of n values(10-10000, steps=10), for every n value. Experiment for 100 times, get the average value.

Code:

[https://github.com/slowpeace2020/INFO6205/blob/Fall2021/src/main/java/edu/neu/coe/info6205/union find/UF_HWQUPC_Client.java](https://github.com/slowpeace2020/INFO6205/blob/Fall2021/src/main/java/edu/neu/coe/info6205/union%20find/UF_HWQUPC_Client.java)



The screenshot shows an IDE with the following components:

- Project Explorer:** A list of project files including `Connections`, `HWQUPC_Solution`, `TypedUF`, `TypedUF_HWQUPC`, `UF`, `UF_HWQUPC`, `UF_HWQUPC_Client` (selected), `UFException`, `WQUPC`, and a `util` package containing `BinarySearch`, `CallByValue`, `ComparableTuple`, `Counter`, and `HuffmanCoding`.
- Editor:** Displays the code for `UF_HWQUPC_Client.java`. The code is as follows:

```
6 import java.util.Random;
7
8 public class UF_HWQUPC_Client {
9     public static void main(String[] args) {
10
11         try {
12             FileWriter dataCsv = new FileWriter("data.csv");
13             String header = "nodes,connect\n";
14             dataCsv.write(header);
15             for(int n=10;n<=10000;n=n+10){
16                 StringBuffer str = new StringBuffer();
17                 int operationCount = count(n);
18                 str.append(n+","+operationCount+"\n");
19                 dataCsv.write(str.toString());
20                 dataCsv.flush();
21             }
22         } catch (Exception e) {
23             e.printStackTrace();
24         }
25     }
26 }
```
- Run Console:** Shows the execution of `UF_HWQUPC_Client` using the Java Virtual Machine `jdk-11.0.2.jdk/Contents/Home/bin/java`. The output is `Process finished with exit code 0`.

(graph 13. UF_HWQUPC_Client experient)

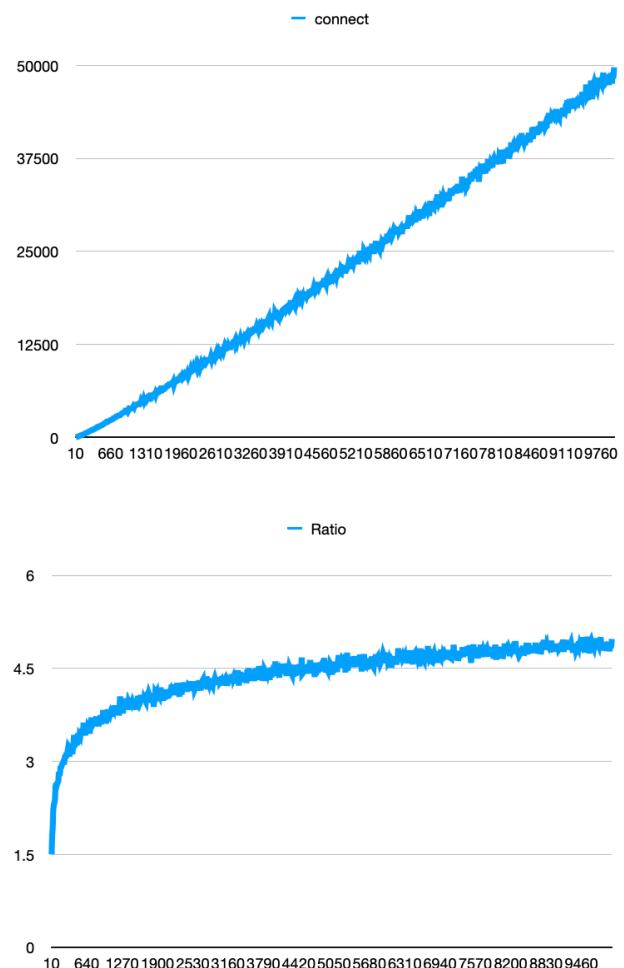
Step3

Evidence

Csv data

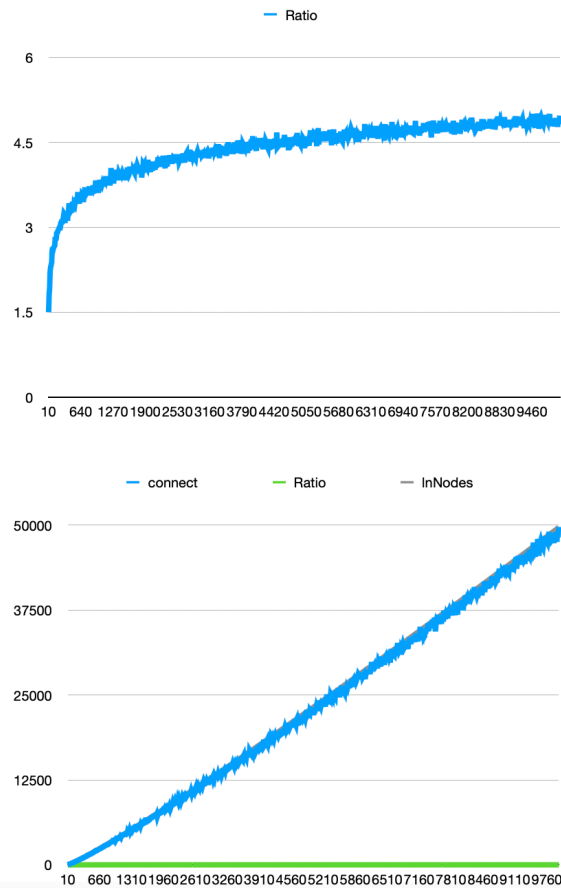
Write the experimental results into csv, and draw graphs to show the relationship the number of objects (n) and the number of pairs (m) .

union_find_data			
nodes	connect	Ratio	lnNodes
10	15	1.5	13.6226072054083
20	36	1.8	35.5515605136088
30	58	1.93333333333333	60.2639436790051
40	89	2.225	86.7599794668995
50	115	2.3	114.57082181277
60	140	2.33333333333333	143.422840225395
70	167	2.38571428571429	173.136474565066
80	208	2.6	203.584802542307
90	238	2.64444444444444	234.673351737575
100	261	2.61	266.329071620481
110	291	2.64545454545455	298.493795594115
120	318	2.65	331.12011789812
130	348	2.67692307692308	364.168664354718
140	396	2.82857142857143	397.606213782558
150	416	2.77333333333333	431.404362611314
160	449	2.80625	465.53854961807
170	497	2.92352941176471	499.987327080984
180	520	2.88888888888889	534.731805167822
190	558	2.93684210526316	569.755220993984
200	591	2.955	605.04259925339
210	620	2.95238095238095	640.580481335916
220	659	2.99545454545455	676.356706492794
230	679	2.95217391304348	712.360233128223
240	721	3.00416666666667	748.580991427956
250	775	3.1	785.00976074824
260	792	3.04615384615385	821.638066777525
270	847	3.13703703703704	858.458094641907
280	860	3.07142857142857	895.462614981579
290	907	3.12758620689655	932.644920666545
300	939	3.13	969.998772305223
310	1006	3.24516129032258	1007.51835107101
320	1034	3.23125	1045.19821765905
330	1015	3.07575757575758	1083.03327640938
340	1096	3.22352941176471	1121.01874380889
350	1107	3.16285714285714	1159.15012072427



(graph 14. UF_HWQUPC_Client experiments data the figure between n and m and their ratio)

160	449	2.80625	465.53854961807
170	497	2.92352941176471	499.987327080984
180	520	2.88888888888889	534.731805167822
190	558	2.93684210526316	569.755220993984
200	591	2.955	605.04259925339
210	620	2.95238095238095	640.580481335916
220	659	2.99545454545455	676.356706492794
230	679	2.95217391304348	712.360233128223
240	721	3.00416666666667	748.580991427956
250	775	3.1	785.00976074824
260	792	3.04615384615385	821.638066777525
270	847	3.13703703703704	858.458094641907
280	860	3.07142857142857	895.462614981579
290	907	3.12758620689655	932.644920666545
300	939	3.13	969.998772305223
310	1006	3.24516129032258	1007.518351071101
320	1034	3.23125	1045.19821765905
330	1015	3.07575757575758	1083.03327640938
340	1096	3.22352941176471	1121.01874380889
350	1107	3.16285714285714	1159.15012072427
360	1168	3.24444444444444	1197.42316782976
370	1209	3.26756756756757	1235.8338837834
380	1184	3.11578947368421	1274.37848577817
390	1256	3.22051282051282	1313.05339215377
400	1314	3.285	1351.85520680336
410	1404	3.42439024390244	1390.78070514962
420	1420	3.38095238095238	1429.82682149768
430	1405	3.26744186046512	1468.99063760014
440	1444	3.28181818181818	1508.26937229244
450	1513	3.36222222222222	1547.66037207629
460	1499	3.25869565217391	1587.16110254535
470	1582	3.36595744680851	1626.7691405609



At first I thought the relationship between them was linear, because from the first graph it looked like a straight line, but I calculated the ratio between them and found that this ratio becomes more and more as the number of nodes increases. The larger the coming, the ratio itself grows in a log function.

Conclusion

So it is guessed that this coefficient should be related to the log function. After trying, they almost overlap. It was found that the relationship the number of objects (N) and the number of pairs (M):

$$M = N \cdot \log(N \cdot \ln N)$$