INFO6205 Assignment 4 (WQUPC)

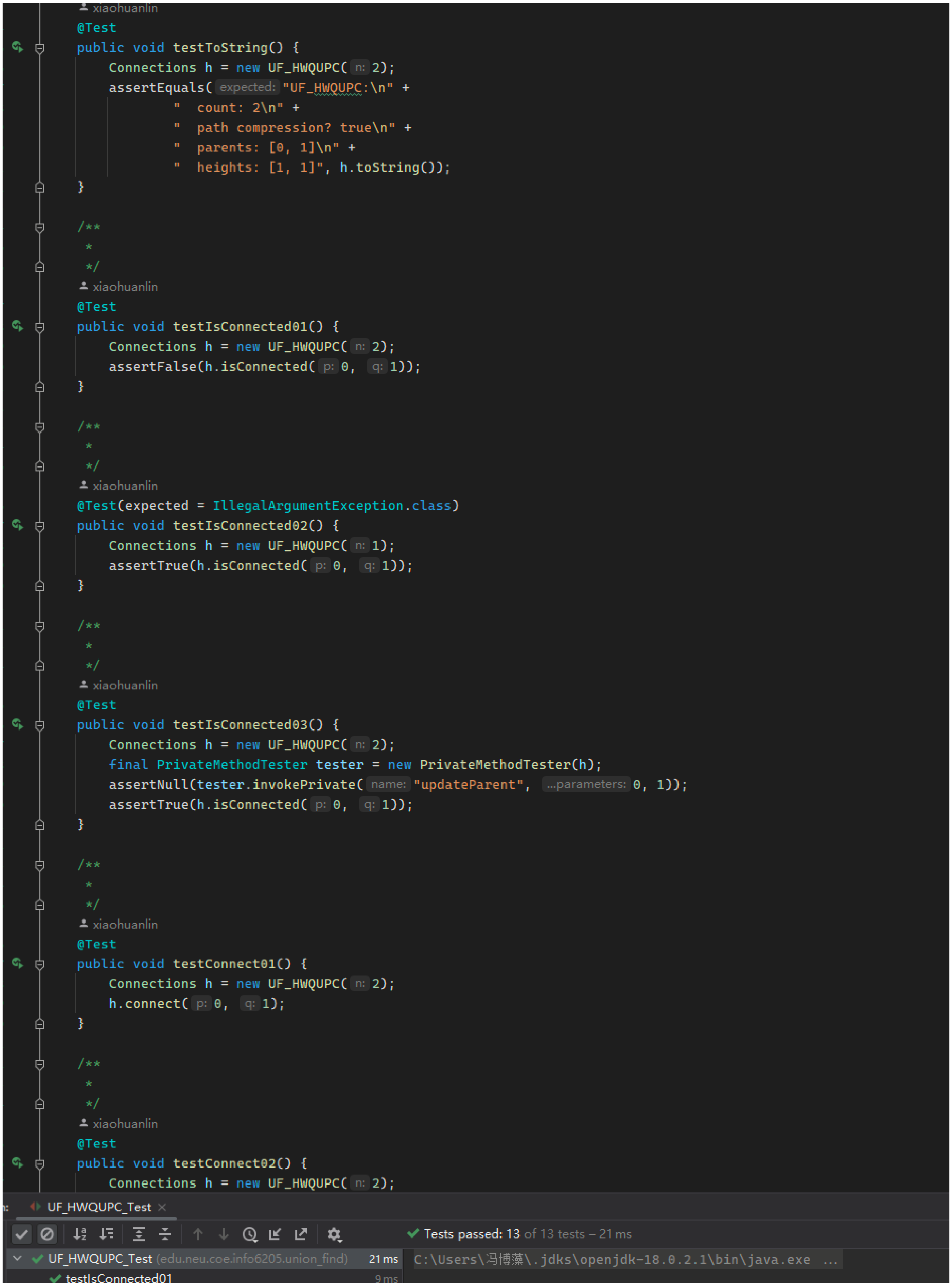
NAME: Bohan Feng

NUID: 001564249

Repository: https://github.com/fengb3/INFO6205

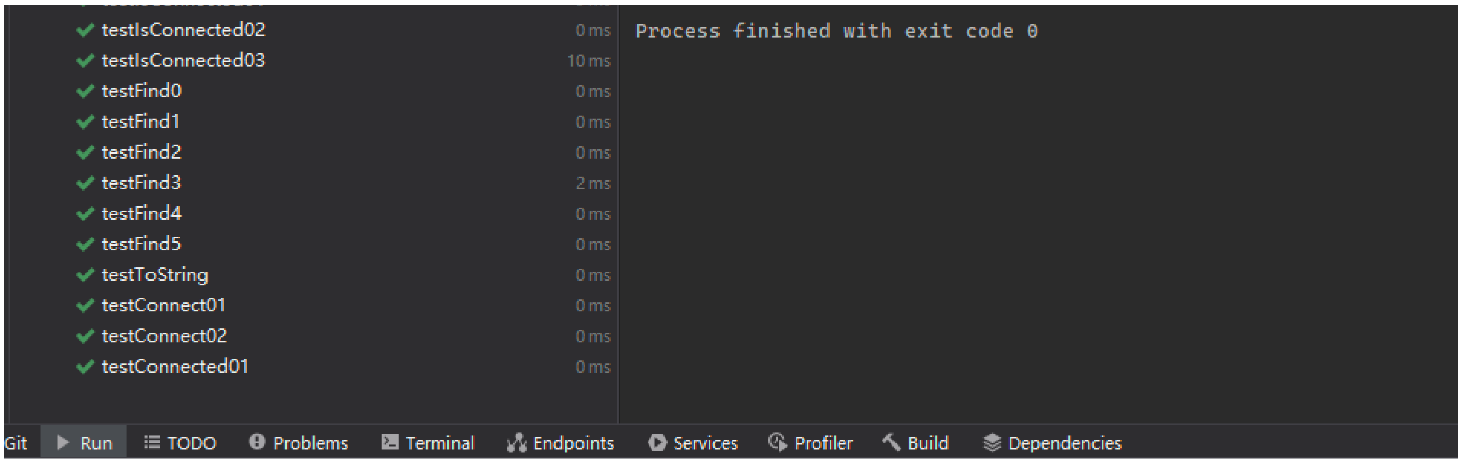
Step 01

Screen shot for test case passed without modifing test code



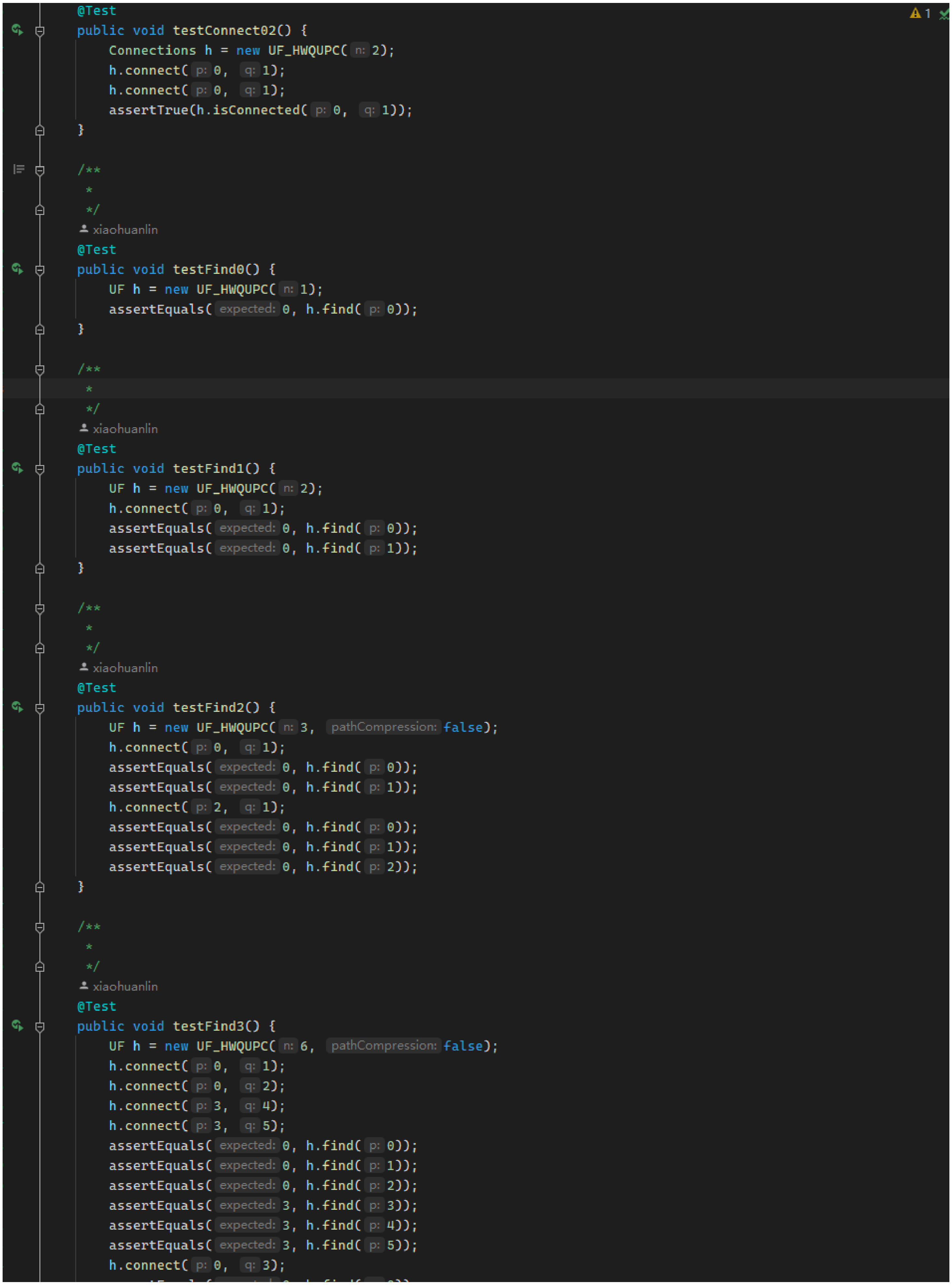
```
xiaohuanlin
@Test
public void testToString() {
    Connections h = new UF_HWQUPC( n: 2);
    assertEquals( expected: "UF_HWQUPC:\n" +
            "  count: 2\n" +
            "  path compression? true\n" +
            "  parents: [0, 1]\n" +
            "  heights: [1, 1]", h.toString());
}

/**
 *
 */
xiaohuanlin
@Test
public void testIsConnected01() {
    Connections h = new UF_HWQUPC( n: 2);
    assertFalse(h.isConnected( p: 0,  q: 1));
}

/**
 *
 */
xiaohuanlin
@Test(expected = IllegalArgumentException.class)
public void testIsConnected02() {
    Connections h = new UF_HWQUPC( n: 1);
    assertTrue(h.isConnected( p: 0,  q: 1));
}

/**
 *
 */
xiaohuanlin
@Test
public void testIsConnected03() {
    Connections h = new UF_HWQUPC( n: 2);
    final PrivateMethodTester tester = new PrivateMethodTester(h);
    assertNull(tester.invokePrivate( name: "updateParent",  ...parameters: 0, 1));
    assertTrue(h.isConnected( p: 0,  q: 1));
}

/**
 *
 */
xiaohuanlin
@Test
public void testConnect01() {
    Connections h = new UF_HWQUPC( n: 2);
    h.connect( p: 0,  q: 1);
}

/**
 *
 */
xiaohuanlin
@Test
public void testConnect02() {
    Connections h = new UF_HWQUPC( n: 2);
```

```
UF_HWQUPC_Test ×
✓ Tests passed: 13 of 13 tests – 21 ms
✓ UF_HWQUPC_Test (edu.neu.coe.info6205.union_find)  21 ms    C:\Users\冯博藻\.jdks\openjdk-18.0.2.1\bin\java.exe ...
  ✓ testIsConnected01                                9 ms
```

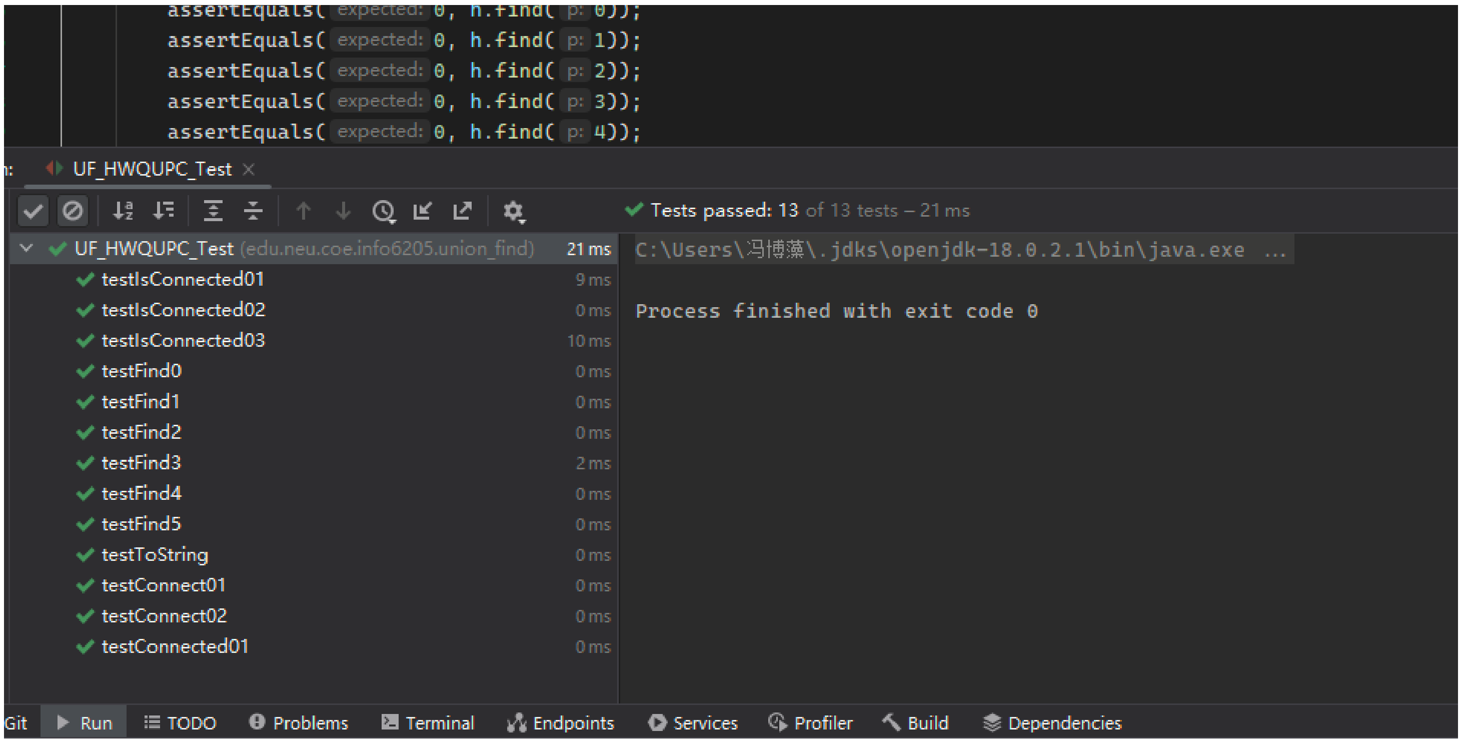


```
✓ testIsConnected02          0 ms
✓ testIsConnected03         10 ms
✓ testFind0                  0 ms
✓ testFind1                  0 ms
✓ testFind2                  0 ms
✓ testFind3                  2 ms
✓ testFind4                  0 ms
✓ testFind5                  0 ms
✓ testToString               0 ms
✓ testConnect01              0 ms
✓ testConnect02              0 ms
✓ testConnected01            0 ms
```

```
Process finished with exit code 0
```

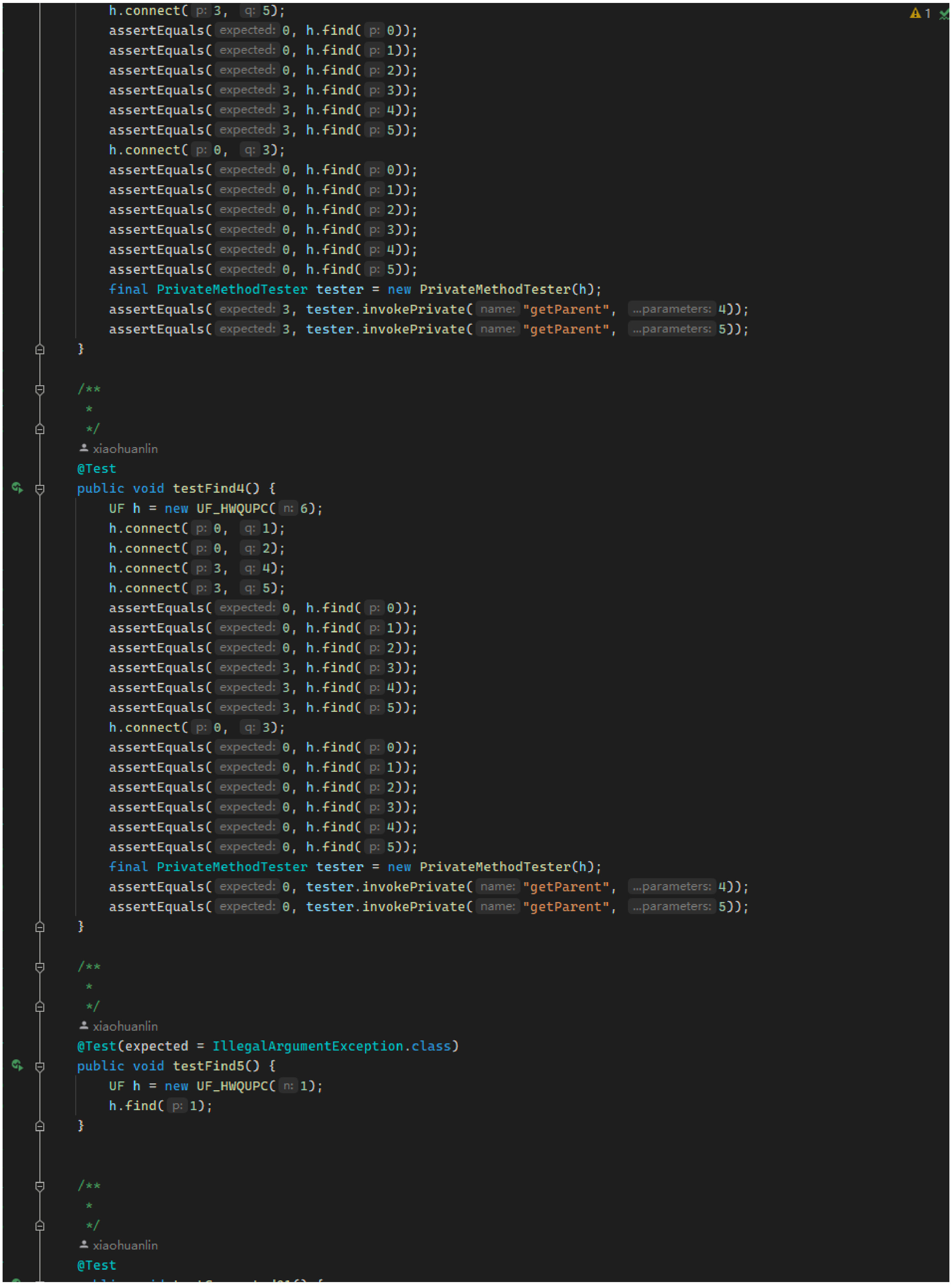


```
    @Test
    public void testConnect02() {
        Connections h = new UF_HWQUPC( n: 2);
        h.connect( p: 0,  q: 1);
        h.connect( p: 0,  q: 1);
        assertTrue(h.isConnected( p: 0,  q: 1));
    }

    /**
     *
     */
    xiaohuanlin
    @Test
    public void testFind0() {
        UF h = new UF_HWQUPC( n: 1);
        assertEquals( expected: 0, h.find( p: 0));
    }

    /**
     *
     */
    xiaohuanlin
    @Test
    public void testFind1() {
        UF h = new UF_HWQUPC( n: 2);
        h.connect( p: 0,  q: 1);
        assertEquals( expected: 0, h.find( p: 0));
        assertEquals( expected: 0, h.find( p: 1));
    }

    /**
     *
     */
    xiaohuanlin
    @Test
    public void testFind2() {
        UF h = new UF_HWQUPC( n: 3,  pathCompression: false);
        h.connect( p: 0,  q: 1);
        assertEquals( expected: 0, h.find( p: 0));
        assertEquals( expected: 0, h.find( p: 1));
        h.connect( p: 2,  q: 1);
        assertEquals( expected: 0, h.find( p: 0));
        assertEquals( expected: 0, h.find( p: 1));
        assertEquals( expected: 0, h.find( p: 2));
    }

    /**
     *
     */
    xiaohuanlin
    @Test
    public void testFind3() {
        UF h = new UF_HWQUPC( n: 6,  pathCompression: false);
        h.connect( p: 0,  q: 1);
        h.connect( p: 0,  q: 2);
        h.connect( p: 3,  q: 4);
        h.connect( p: 3,  q: 5);
        assertEquals( expected: 0, h.find( p: 0));
        assertEquals( expected: 0, h.find( p: 1));
        assertEquals( expected: 0, h.find( p: 2));
        assertEquals( expected: 3, h.find( p: 3));
        assertEquals( expected: 3, h.find( p: 4));
        assertEquals( expected: 3, h.find( p: 5));
        h.connect( p: 0,  q: 3);
```

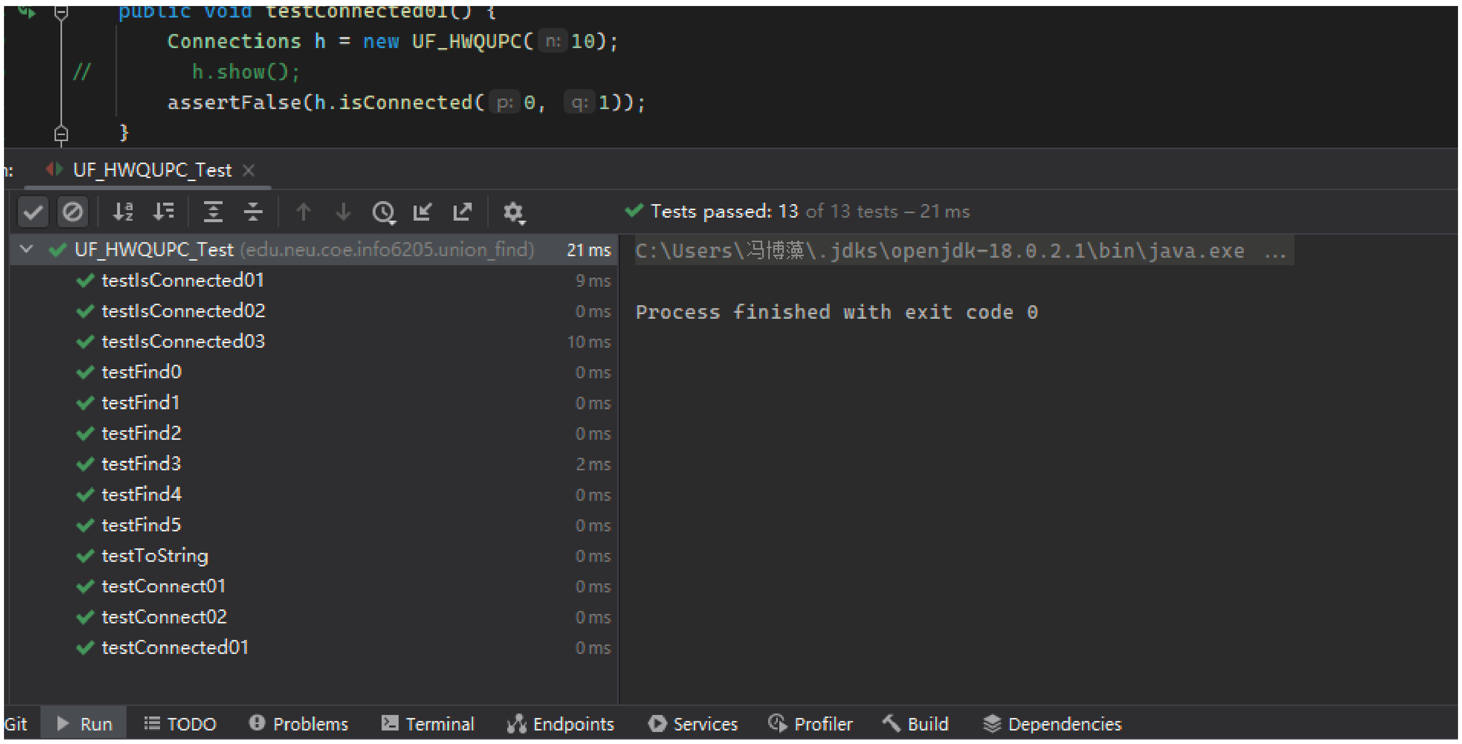


```
        assertEquals( expected: 0, h.find( p: 0));
        assertEquals( expected: 0, h.find( p: 1));
        assertEquals( expected: 0, h.find( p: 2));
        assertEquals( expected: 0, h.find( p: 3));
        assertEquals( expected: 0, h.find( p: 4));
```

```
UF_HWQUPC_Test
✓ Tests passed: 13 of 13 tests – 21 ms

✓ UF_HWQUPC_Test (edu.neu.coe.info6205.union_find)  21 ms
    ✓ testIsConnected01     9 ms
    ✓ testIsConnected02     0 ms
    ✓ testIsConnected03    10 ms
    ✓ testFind0             0 ms
    ✓ testFind1             0 ms
    ✓ testFind2             0 ms
    ✓ testFind3             2 ms
    ✓ testFind4             0 ms
    ✓ testFind5             0 ms
    ✓ testToString          0 ms
    ✓ testConnect01         0 ms
    ✓ testConnect02         0 ms
    ✓ testConnected01       0 ms

C:\Users\冯博藻\.jdks\openjdk-18.0.2.1\bin\java.exe ...

Process finished with exit code 0
```



```
        h.connect( p: 3,  q: 5);
        assertEquals( expected: 0, h.find( p: 0));
        assertEquals( expected: 0, h.find( p: 1));
        assertEquals( expected: 0, h.find( p: 2));
        assertEquals( expected: 3, h.find( p: 3));
        assertEquals( expected: 3, h.find( p: 4));
        assertEquals( expected: 3, h.find( p: 5));
        h.connect( p: 0,  q: 3);
        assertEquals( expected: 0, h.find( p: 0));
        assertEquals( expected: 0, h.find( p: 1));
        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: 3, tester.invokePrivate( name: "getParent",  ...parameters: 4));
        assertEquals( expected: 3, tester.invokePrivate( name: "getParent",  ...parameters: 5));
    }

    /**
     *
     */
    xiaohuanlin
    @Test
    public void testFind4() {
        UF h = new UF_HWQUPC( n: 6);
        h.connect( p: 0,  q: 1);
        h.connect( p: 0,  q: 2);
        h.connect( p: 3,  q: 4);
        h.connect( p: 3,  q: 5);
        assertEquals( expected: 0, h.find( p: 0));
        assertEquals( expected: 0, h.find( p: 1));
        assertEquals( expected: 0, h.find( p: 2));
        assertEquals( expected: 3, h.find( p: 3));
        assertEquals( expected: 3, h.find( p: 4));
        assertEquals( expected: 3, h.find( p: 5));
        h.connect( p: 0,  q: 3);
        assertEquals( expected: 0, h.find( p: 0));
        assertEquals( expected: 0, h.find( p: 1));
        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));
    }

    /**
     *
     */
    xiaohuanlin
    @Test(expected = IllegalArgumentException.class)
    public void testFind5() {
        UF h = new UF_HWQUPC( n: 1);
        h.find( p: 1);
    }


    /**
     *
     */
    xiaohuanlin
    @Test
```



```
    public void testConnected01() {
        Connections h = new UF_HWQUPC( n: 10);
//        h.show();
        assertFalse(h.isConnected( p: 0,  q: 1));
    }
```

UF_HWQUPC_Test

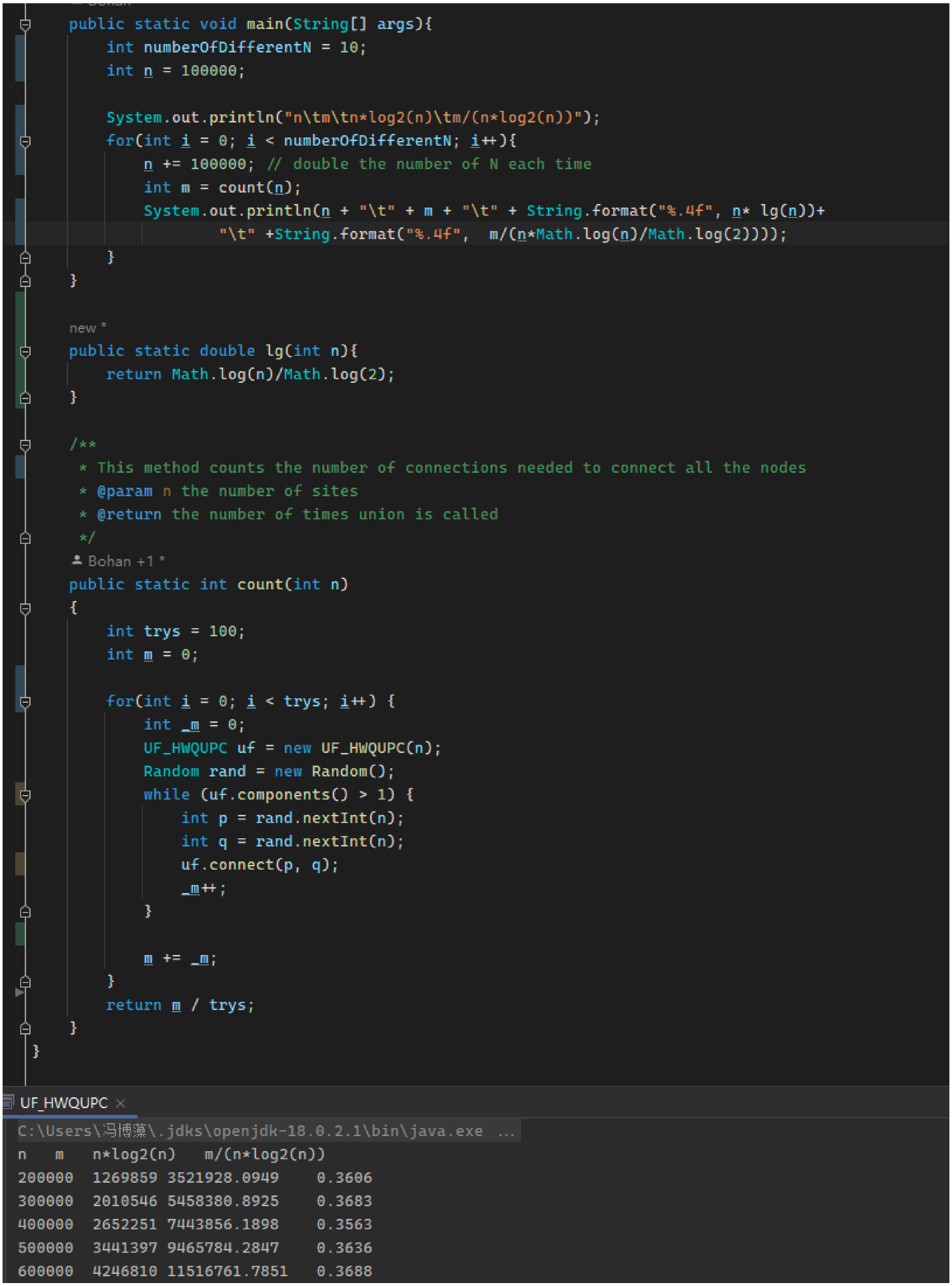
✓ Tests passed: 13 of 13 tests – 21 ms

```
UF_HWQUPC_Test (edu.neu.coe.info6205.union_find)   21 ms
    testIsConnected01   9 ms
    testIsConnected02   0 ms
    testIsConnected03   10 ms
    testFind0           0 ms
    testFind1           0 ms
    testFind2           0 ms
    testFind3           2 ms
    testFind4           0 ms
    testFind5           0 ms
    testToString        0 ms
    testConnect01       0 ms
    testConnect02       0 ms
    testConnected01     0 ms
```

```
C:\Users\冯博藻\.jdks\openjdk-18.0.2.1\bin\java.exe ...

Process finished with exit code 0
```

Step 02

Use UF_HWQUPC to implement static method to findout the how may connection will be created with a given number of "sites"



```
    public static void main(String[] args){
        int numberOfDifferentN = 10;
        int n = 100000;

        System.out.println("n\tm\tn*log2(n)\tm/(n*log2(n))");
        for(int i = 0; i < numberOfDifferentN; i++){
            n += 100000; // double the number of N each time
            int m = count(n);
            System.out.println(n + "\t" + m + "\t" + String.format("%.4f", n* lg(n))+
                    "\t" +String.format("%.4f",  m/(n*Math.log(n)/Math.log(2))));
        }
    }

    new *
    public static double lg(int n){
        return Math.log(n)/Math.log(2);
    }

    /**
     * This method counts the number of connections needed to connect all the nodes
     * @param n the number of sites
     * @return the number of times union is called
     */
    Bohan +1 *
    public static int count(int n)
    {
        int trys = 100;
        int m = 0;

        for(int i = 0; i < trys; i++) {
            int _m = 0;
            UF_HWQUPC uf = new UF_HWQUPC(n);
            Random rand = new Random();
            while (uf.components() > 1) {
                int p = rand.nextInt(n);
                int q = rand.nextInt(n);
                uf.connect(p, q);
                _m++;
            }

            m += _m;
        }
        return m / trys;
    }
}
```

UF_HWQUPC

```
C:\Users\冯博藻\.jdks\openjdk-18.0.2.1\bin\java.exe ...
n	m	n*log2(n)	m/(n*log2(n))
200000	1269859 3521928.0949	0.3606
300000	2010546 5458380.8925	0.3683
400000	2652251 7443856.1898	0.3563
500000	3441397 9465784.2847	0.3636
600000	4246810 11516761.7851	0.3688
```

```
700000  4870757 13591896.7775   0.3584
800000  5648808 15687712.3795   0.3601
900000  6376504 17801608.9283   0.3582
1000000 7351317 19931568.5693   0.3688
1100000 7984221 22075979.3024   0.3617

Process finished with exit code 0
```

Step 03

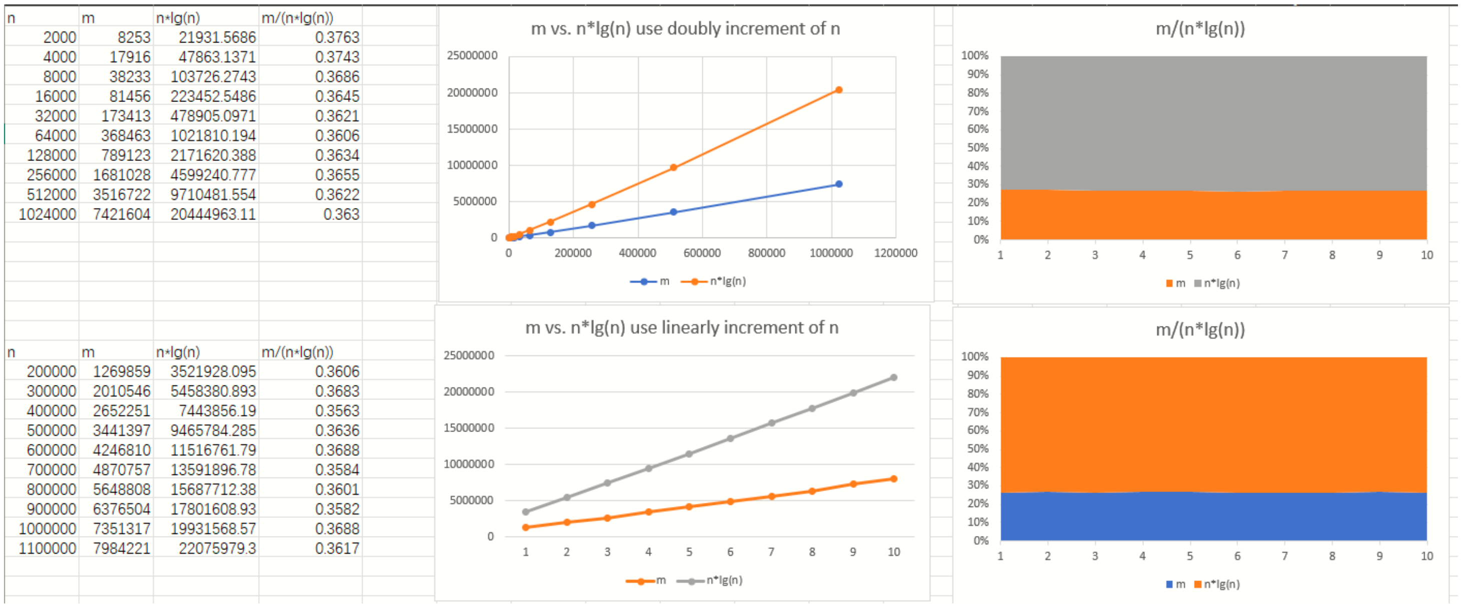
Determine the relationship between the number of objects (n) and the number of pairs (m) generated to accomplish reducing the number of components from n to 1

Based on my observation the releationshoip between m and n is

$$m = k * n * lg(n)$$

where k is a constant which approximately equal to 3.7

Evidence:



n	m	n*lg(n)	m/(n*lg(n))
2000	8253	21931.5686	0.3763
4000	17916	47863.1371	0.3743
8000	38233	103726.2743	0.3686
16000	81456	223452.5486	0.3645
32000	173413	478905.0971	0.3621
64000	368463	1021810.194	0.3606
128000	789123	2171620.388	0.3634
256000	1681028	4599240.777	0.3655
512000	3516722	9710481.554	0.3622
1024000	7421604	20444963.11	0.363

n	m	n*lg(n)	m/(n*lg(n))
200000	1269859	3521928.095	0.3606
300000	2010546	5458380.893	0.3683
400000	2652251	7443856.19	0.3563
500000	3441397	9465784.285	0.3636
600000	4246810	11516761.79	0.3688
700000	4870757	13591896.78	0.3584
800000	5648808	15687712.38	0.3601
900000	6376504	17801608.93	0.3582
1000000	7351317	19931568.57	0.3688
1100000	7984221	22075979.3	0.3617

m vs. n*lg(n) use doubly increment of n
m
n*lg(n)
m/(n*lg(n))
m vs. n*lg(n) use linearly increment of n
m/(n*lg(n))

On the screenshot of spreadsheet and graph above, I found that $m/n * lg(n)$ is always a constant (when n is a large number).