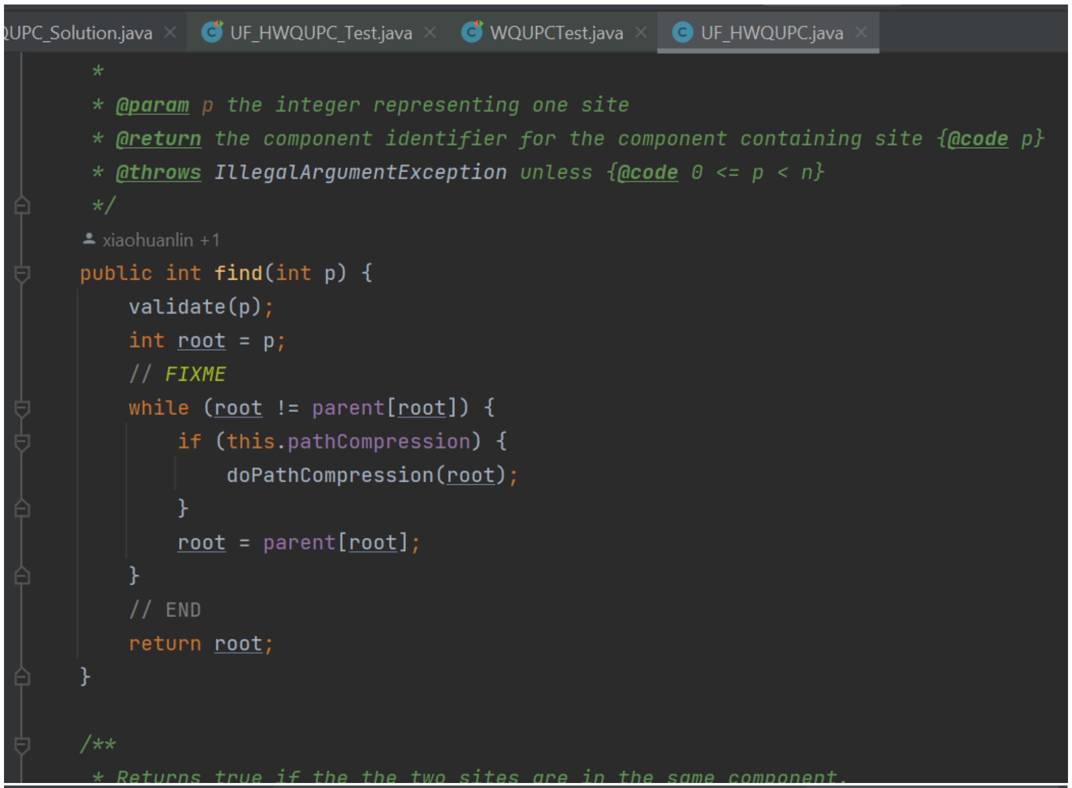
PSA – Assignment 4

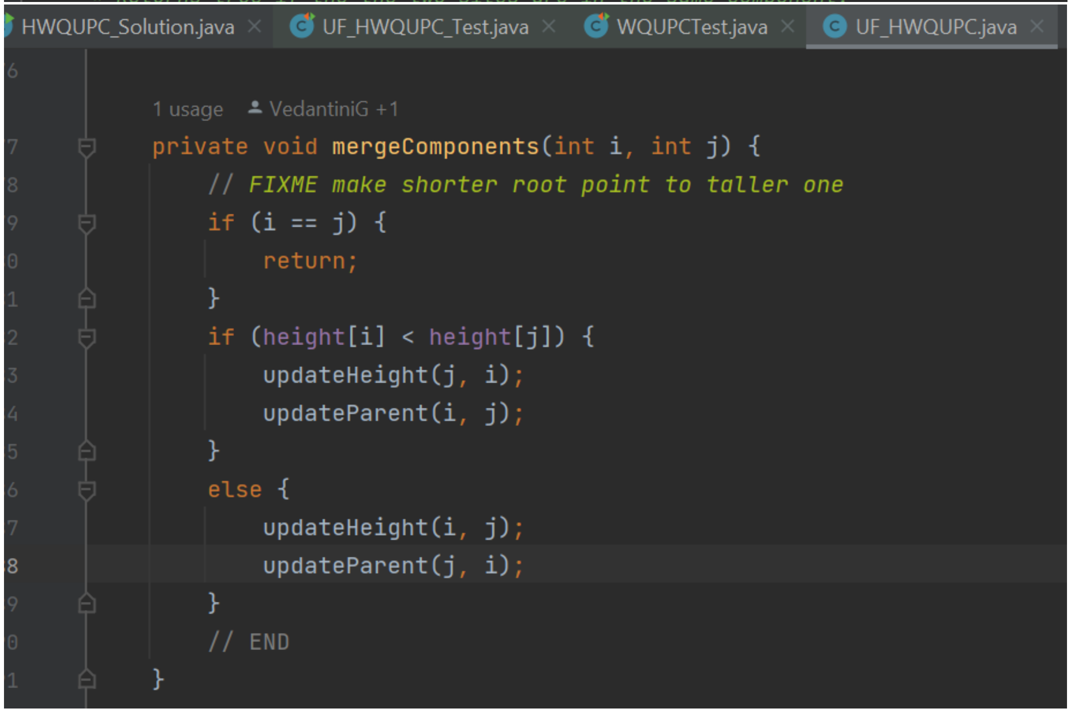
WQUPC

Name: Vedantini Dilip Gaikwad
NUID: 002998254

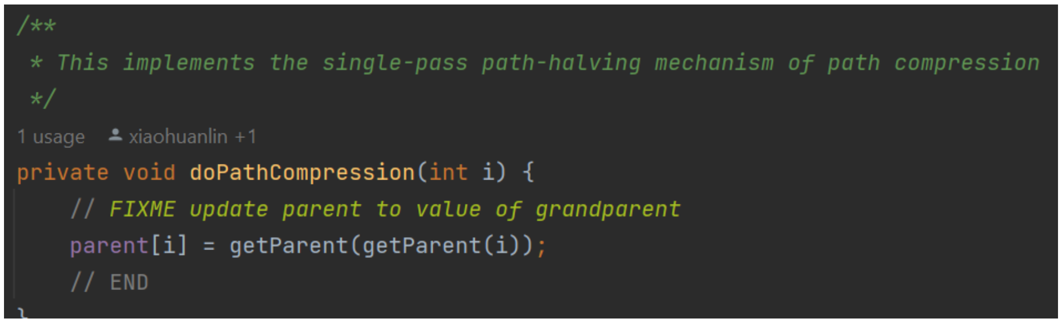
- **Part 1**


UPC_Solution.java × | UF_HWQUPC_Test.java × | WQUPCTest.java × | UF_HWQUPC.java ×

```java
     *
     * @param p the integer representing one site
     * @return the component identifier for the component containing site {@code p}
     * @throws IllegalArgumentException unless {@code 0 <= p < n}
     */
    xiaohuanlin +1
    public int find(int p) {
        validate(p);
        int root = p;
        // FIXME
        while (root != parent[root]) {
            if (this.pathCompression) {
                doPathCompression(root);
            }
            root = parent[root];
        }
        // END
        return root;
    }

    /**
     * Returns true if the the two sites are in the same component.
```

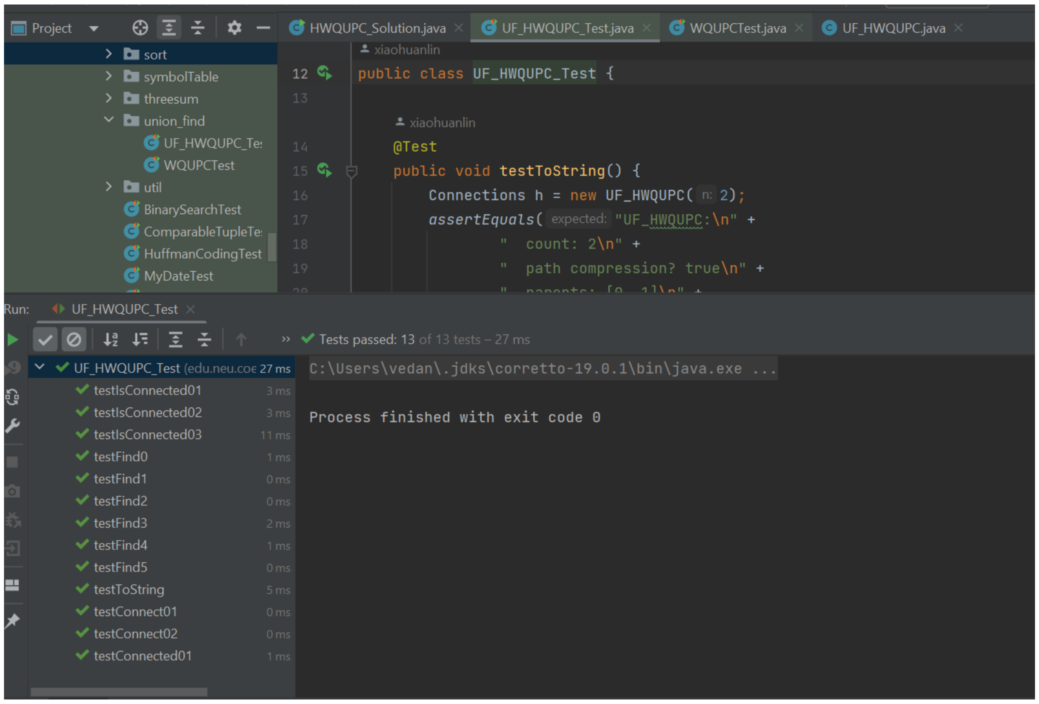

HWQUPC_Solution.java × | UF_HWQUPC_Test.java × | WQUPCTest.java × | UF_HWQUPC.java ×

```java
    1 usage   VedantiniG +1
    private void mergeComponents(int i, int j) {
        // FIXME make shorter root point to taller one
        if (i == j) {
            return;
        }
        if (height[i] < height[j]) {
            updateHeight(j, i);
            updateParent(i, j);
        }
        else {
            updateHeight(i, j);
            updateParent(j, i);
        }
        // END
    }
```



```
/**
 * This implements the single-pass path-halving mechanism of path compression
 */
1 usage  xiaohuanlin +1
private void doPathCompression(int i) {
    // FIXME update parent to value of grandparent
    parent[i] = getParent(getParent(i));
    // END
```

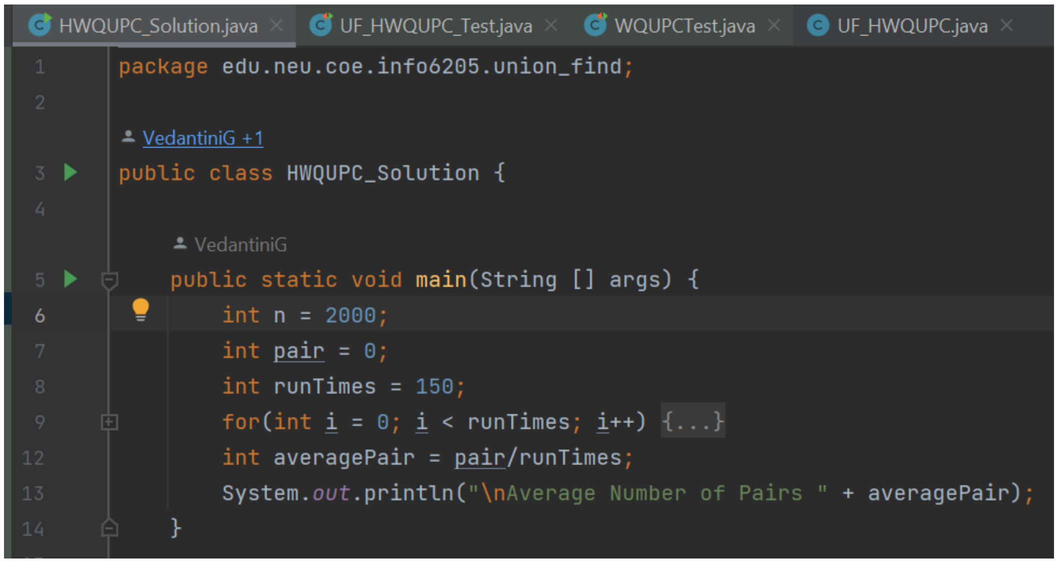
Unit Tests:



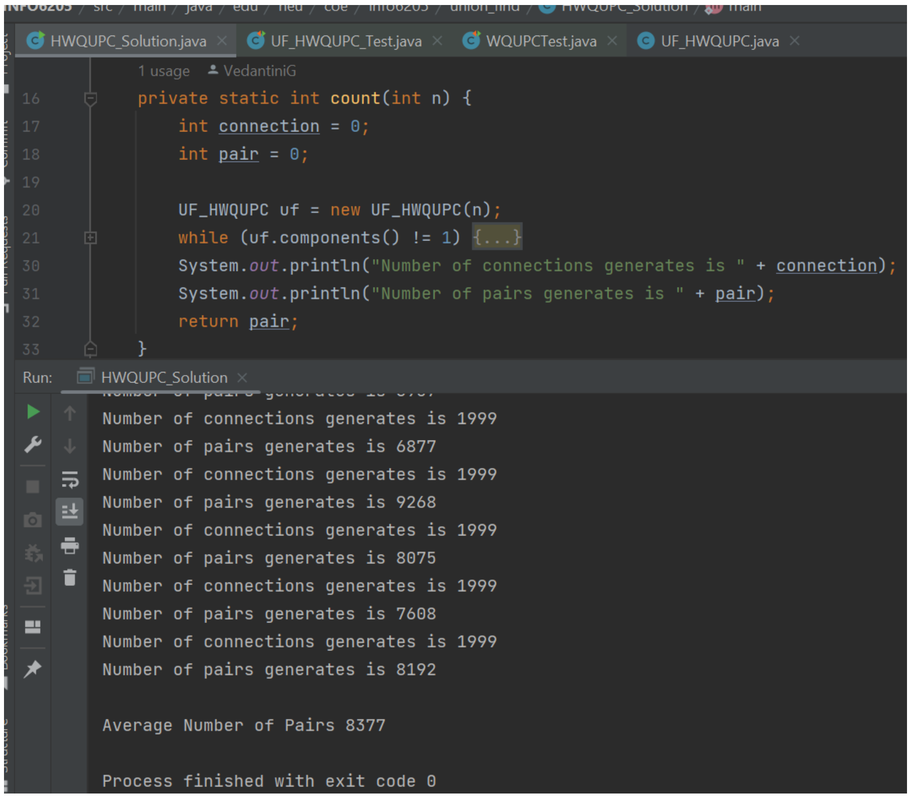
```
xiaohuanlin
public class UF_HWQUPC_Test {

    xiaohuanlin
    @Test
    public void testToString() {
        Connections h = new UF_HWQUPC( n: 2);
        assertEquals( expected: "UF_HWQUPC:\n" +
                "  count: 2\n" +
                "  path compression? true\n" +
```

```
✓ Tests passed: 13 of 13 tests – 27 ms

✓ UF_HWQUPC_Test (edu.neu.coe 27 ms
    ✓ testIsConnected01    3 ms
    ✓ testIsConnected02    3 ms
    ✓ testIsConnected03   11 ms
    ✓ testFind0            1 ms
    ✓ testFind1            0 ms
    ✓ testFind2            0 ms
    ✓ testFind3            2 ms
    ✓ testFind4            1 ms
    ✓ testFind5            0 ms
    ✓ testToString         5 ms
    ✓ testConnect01        0 ms
    ✓ testConnect02        0 ms
    ✓ testConnected01      1 ms

C:\Users\vedan\.jdks\corretto-19.0.1\bin\java.exe ...

Process finished with exit code 0
```

- **Part 2**



```
package edu.neu.coe.info6205.union_find;

VedantiniG +1
public class HWQUPC_Solution {

    VedantiniG
    public static void main(String [] args) {
        int n = 2000;
        int pair = 0;
        int runTimes = 150;
        for(int i = 0; i < runTimes; i++) {...}
        int averagePair = pair/runTimes;
        System.out.println("\nAverage Number of Pairs " + averagePair);
    }
```



```
1 usage  VedantiniG
private static int count(int n) {
    int connection = 0;
    int pair = 0;

    UF_HWQUPC uf = new UF_HWQUPC(n);
    while (uf.components() != 1) {...}
    System.out.println("Number of connections generates is " + connection);
    System.out.println("Number of pairs generates is " + pair);
    return pair;
}
```

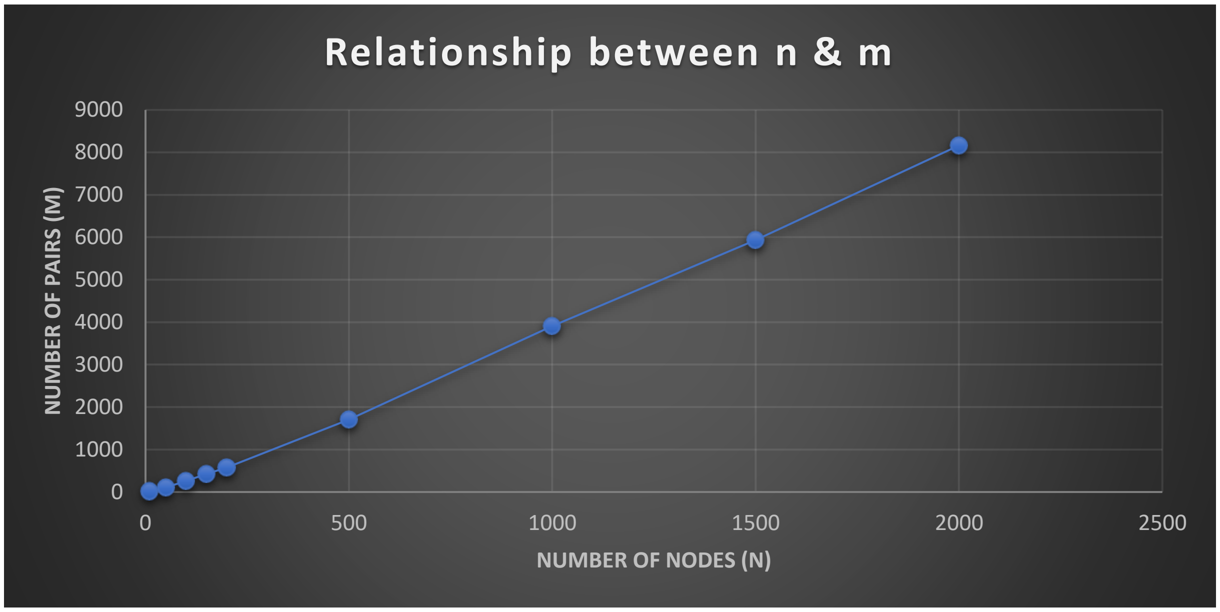
```
Number of connections generates is 1999
Number of pairs generates is 6877
Number of connections generates is 1999
Number of pairs generates is 9268
Number of connections generates is 1999
Number of pairs generates is 8075
Number of connections generates is 1999
Number of pairs generates is 7608
Number of connections generates is 1999
Number of pairs generates is 8192

Average Number of Pairs 8377

Process finished with exit code 0
```

- **Part 3**

  Relationship:-

  The relationship can be represented as $y = mx + c$, where y represents the number of pairs connected by the union-find method and x represents the number of nodes.

Number of nodes (n)	Number of pairs (m)
10	17
50	110
100	257
150	423
200	575
500	1710
1000	3907
1500	5934
2000	8163


Relationship between n & m
NUMBER OF PAIRS (M)
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
500
1000
1500
2000
2500
NUMBER OF NODES (N)