Author

Kang Shentu

001569432

Part1

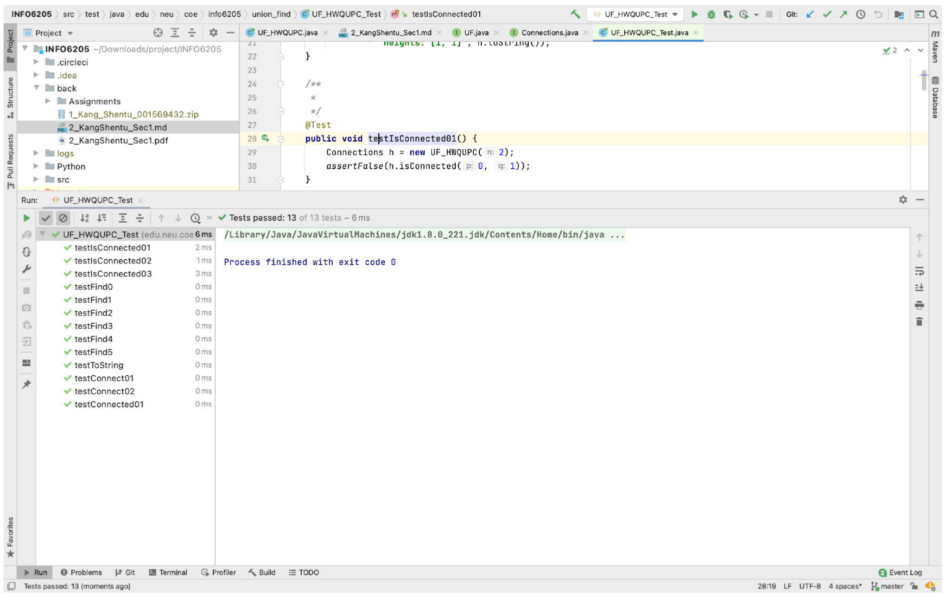
Implementation

```
    // 1. find method
    public int find(int p) {
        validate(p);
        int root = p;
        while (root != getParent(root)) {
            if (pathCompression) {
                doPathCompression(root);
            }
            root = getParent(root);
        }
        return root;
    }

    // 2. mergeComponents
    private void mergeComponents(int i, int j) {
        // TO BE IMPLEMENTED make shorter root point to
taller one
        if (height[i] < height[j]) {
            updateParent(i, j);
            updateHeight(j, i);
        } else {
            updateParent(j, i);
            updateHeight(i, j);
        }
    }

    // 3. doPathCompression
    private void doPathCompression(int i) {
        // TO BE IMPLEMENTED update parent to value of
grandparent
        updateParent(i, getParent(getParent(i)));
    }
```

Screenshot



```
    }

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

```
✓ Tests passed: 13 of 13 tests – 6 ms

✓ UF_HWQUPC_Test (edu.neu.coe  6 ms
    ✓ testIsConnected01         2 ms
    ✓ testIsConnected02         1 ms
    ✓ testIsConnected03         3 ms
    ✓ testFind0                 0 ms
    ✓ testFind1                 0 ms
    ✓ testFind2                 0 ms
    ✓ testFind3                 0 ms
    ✓ testFind4                 0 ms
    ✓ testFind5                 0 ms
    ✓ testToString              0 ms
    ✓ testConnect01             0 ms
    ✓ testConnect02             0 ms
    ✓ testConnected01           0 ms

/Library/Java/JavaVirtualMachines/jdk1.8.0_221.jdk/Contents/Home/bin/java ...

Process finished with exit code 0
```

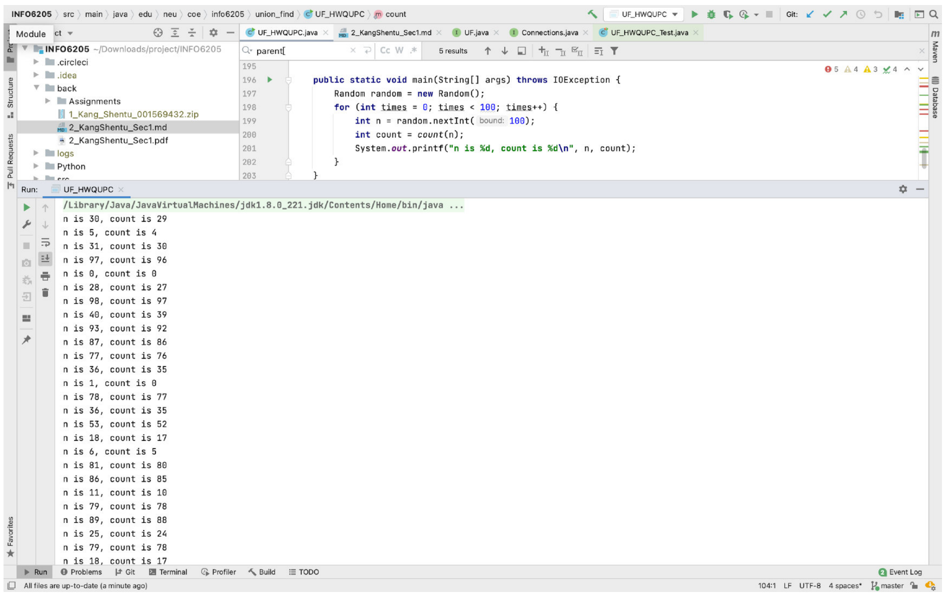
Tests passed: 13 (moments ago)

Part2

Implementation

```
    /**
     * @param n the number of sites
     * @return the number of connections
     */
    private static int count(int n) {
        UF_HWQUPC client = new UF_HWQUPC(n);
        int count = 0;
        for (int i = 0; i < n; i++) {
            for (int j = i; j < n; j++) {
                if (!client.isConnected(i,j)){
                    client.union(i,j);
                    count++;
                }
            }
        }
        return count;
    }
    public static void main(String[] args) throws
IOException {
        Random random = new Random();
        for (int times = 0; times < 100; times++) {
            int n = random.nextInt(100);
            int count = count(n);
            System.out.printf("n is %d, count is %d\n",
n, count);
        }
    }
```

Screenshot



```
    public static void main(String[] args) throws IOException {
        Random random = new Random();
        for (int times = 0; times < 100; times++) {
            int n = random.nextInt( bound: 100);
            int count = count(n);
            System.out.printf("n is %d, count is %d\n", n, count);
        }
    }
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_221.jdk/Contents/Home/bin/java ...
n is 30, count is 29
n is 5, count is 4
n is 31, count is 30
n is 97, count is 96
n is 0, count is 0
n is 28, count is 27
n is 98, count is 97
n is 40, count is 39
n is 93, count is 92
n is 87, count is 86
n is 77, count is 76
n is 36, count is 35
n is 1, count is 0
n is 78, count is 77
n is 36, count is 35
n is 53, count is 52
n is 18, count is 17
n is 6, count is 5
n is 81, count is 80
n is 86, count is 85
n is 11, count is 10
n is 79, count is 78
n is 89, count is 88
n is 25, count is 24
n is 79, count is 78
n is 18, count is 17
```

All files are up-to-date (a minute ago)

Part3

Relationship

N is the number of objects and M is the number of pairs generated to accomplish the requirement.

$$M = N - 1$$

For the purpose of connecting all sites, each site should connect a certain site, to more specific, the site which has the biggest value.

So the conclusion is obvious.