

Program Structures and Algorithms

Spring 2024

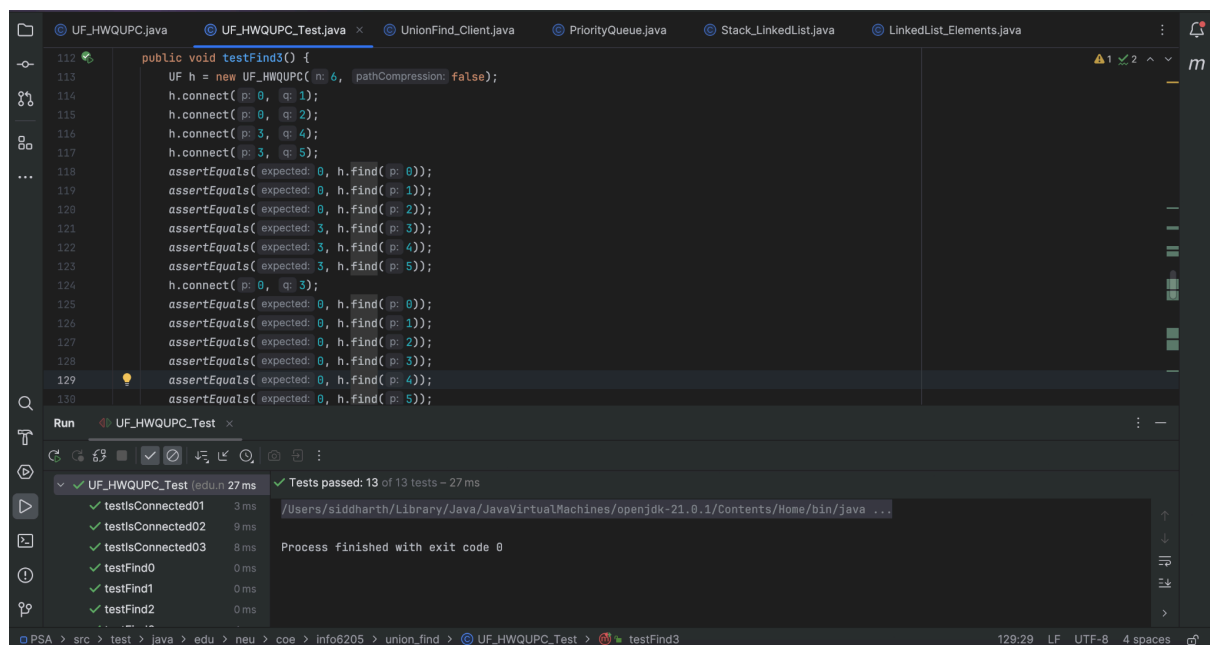
NAME: Siddharth Dumbre

NUID: 002247119

GITHUB LINK: <https://github.com/dumbresi/Info-6205-spring2024>

Task: WQUPC- Weighted Quick Union Path Compression

Output screenshot:



The screenshot displays an IDE with several open files. The active file is `UF_HWQUPC.java`, which contains a `testFind3()` method. This method initializes a `UF_HWQUPC` object with `n=6` and `pathCompression=false`, then performs a series of `connect` and `find` operations, each followed by an `assertEquals` assertion. The `Run` tab at the bottom shows the execution results for `UF_HWQUPC_Test`, indicating that all 13 tests passed successfully within 27 milliseconds. The test results list includes `testIsConnected01` through `testFind2`.

```
public void testFind3() {
    UF h = new UF_HWQUPC(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));
    assertEquals(expected: 0, h.find(p: 5));
}
```

Run: UF_HWQUPC_Test

Tests passed: 13 of 13 tests - 27 ms

- testIsConnected01: 3 ms
- testIsConnected02: 9 ms
- testIsConnected03: 8 ms
- testFind0: 0 ms
- testFind1: 0 ms
- testFind2: 0 ms

Process finished with exit code 0

The screenshot shows an IDE with several Java files open. The active file is `UnionFind_Client.java`, which contains the following code:

```
public class UnionFind_Client {  
    // siddharth_dumbre *  
    public static void main(String[] args){  
        int n=1000;  
        for(int i=1;i<6;i++){  
            System.out.println("n:"+n+", connections:"+count(sites: n+1));  
        }  
    }  
    // siddharth_dumbre  
    public static int count(int sites){  
        int connections=0;  
        Random random= new Random();  
        UF_HWQUPC union= new UF_HWQUPC(sites);  
        while (union.components()>1){  
            // ...  
        }  
        return connections;  
    }  
}
```

The Run console shows the output of the program:

```
n:1000, connections:999  
n:1000, connections:1999  
n:1000, connections:2999  
n:1000, connections:3999  
n:1000, connections:4999  
Process finished with exit code 0
```

Observation:

The relationship between the number of objects(n) and the number of pairs is that the number of pairs(m) is 1 less than the number of sites. I.e. $m=n-1$.