Barla Venkat Sandeep (002135635)

Program Structures & Algorithms

Fall 2021

Assignment No. 3 (WQUPC)

- **Task (List down the tasks performed in the Assignment)**

1) Implemented height-weighted Quick Union with Path Compression

2) Developed Union Find using the above UF_HWQUPC by taking random n (number of sites) values. And returned the value m (number of pairs generated).

3) Derived relationship between n (number of sites) and m (number of pairs generated).

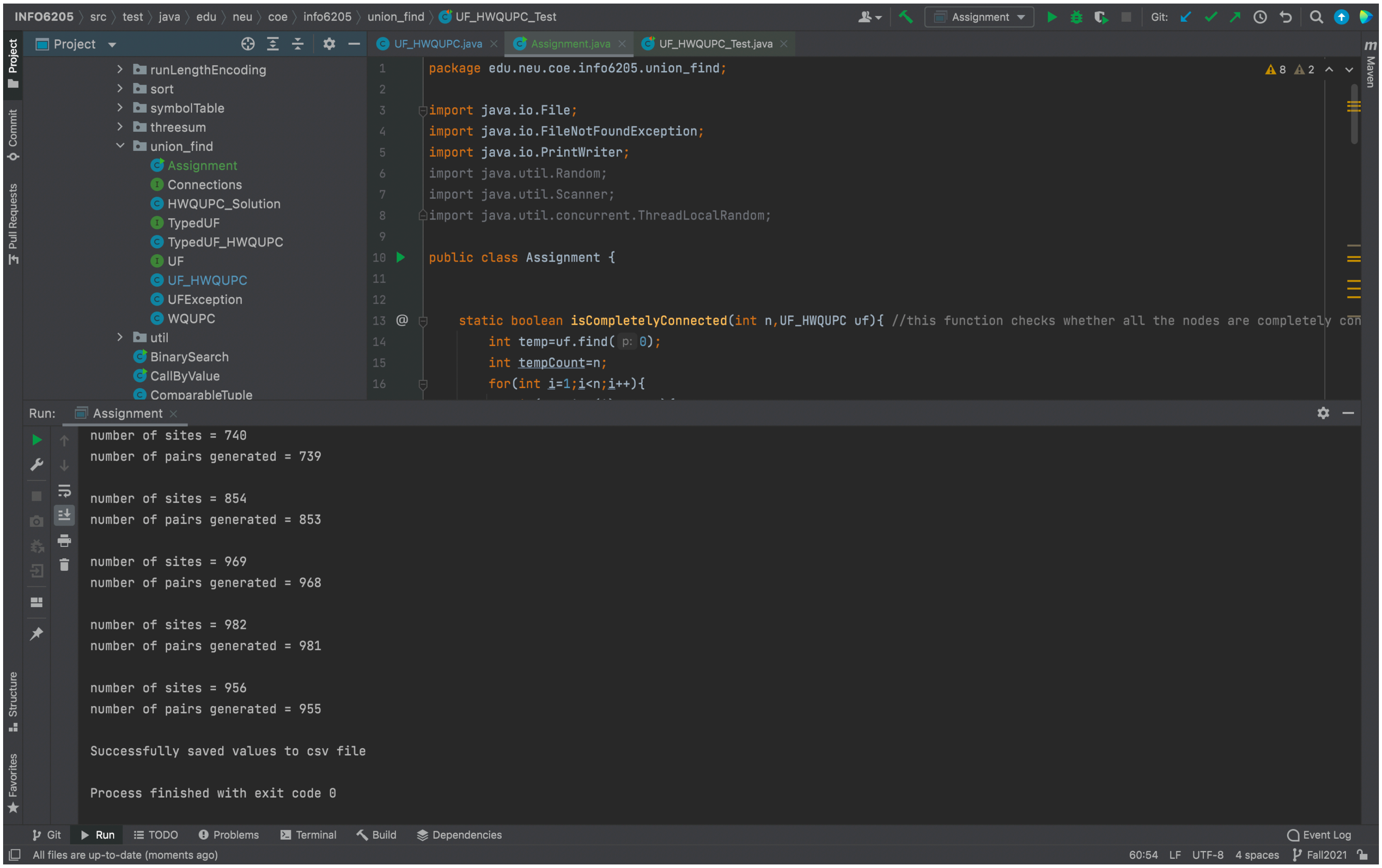
- **Relationship Conclusion: (For ex : z = a * b)**

m = number of pairs generated, n = number of sites

Relationship: m = n-1

- **Evidence to support the conclusion:**

1. **Output (Snapshot of Code output in the terminal)**

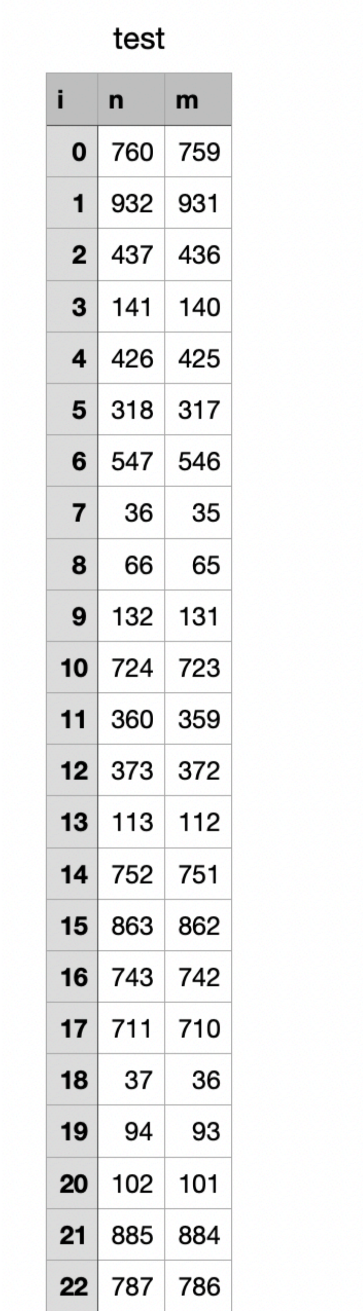


```java
package edu.neu.coe.info6205.union_find;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Random;
import java.util.Scanner;
import java.util.concurrent.ThreadLocalRandom;

public class Assignment {


    static boolean isCompletelyConnected(int n,UF_HWQUPC uf){ //this function checks whether all the nodes are completely con
        int temp=uf.find( p: 0);
        int tempCount=n;
        for(int i=1;i<n;i++){
```

```
number of sites = 740
number of pairs generated = 739

number of sites = 854
number of pairs generated = 853

number of sites = 969
number of pairs generated = 968

number of sites = 982
number of pairs generated = 981

number of sites = 956
number of pairs generated = 955

Successfully saved values to csv file

Process finished with exit code 0
```

2. **Graphical Representation(Observations from experiments should be tabulated and analyzed by plotting graphs(usually in excel) to arrive on the relationship conclusion)**

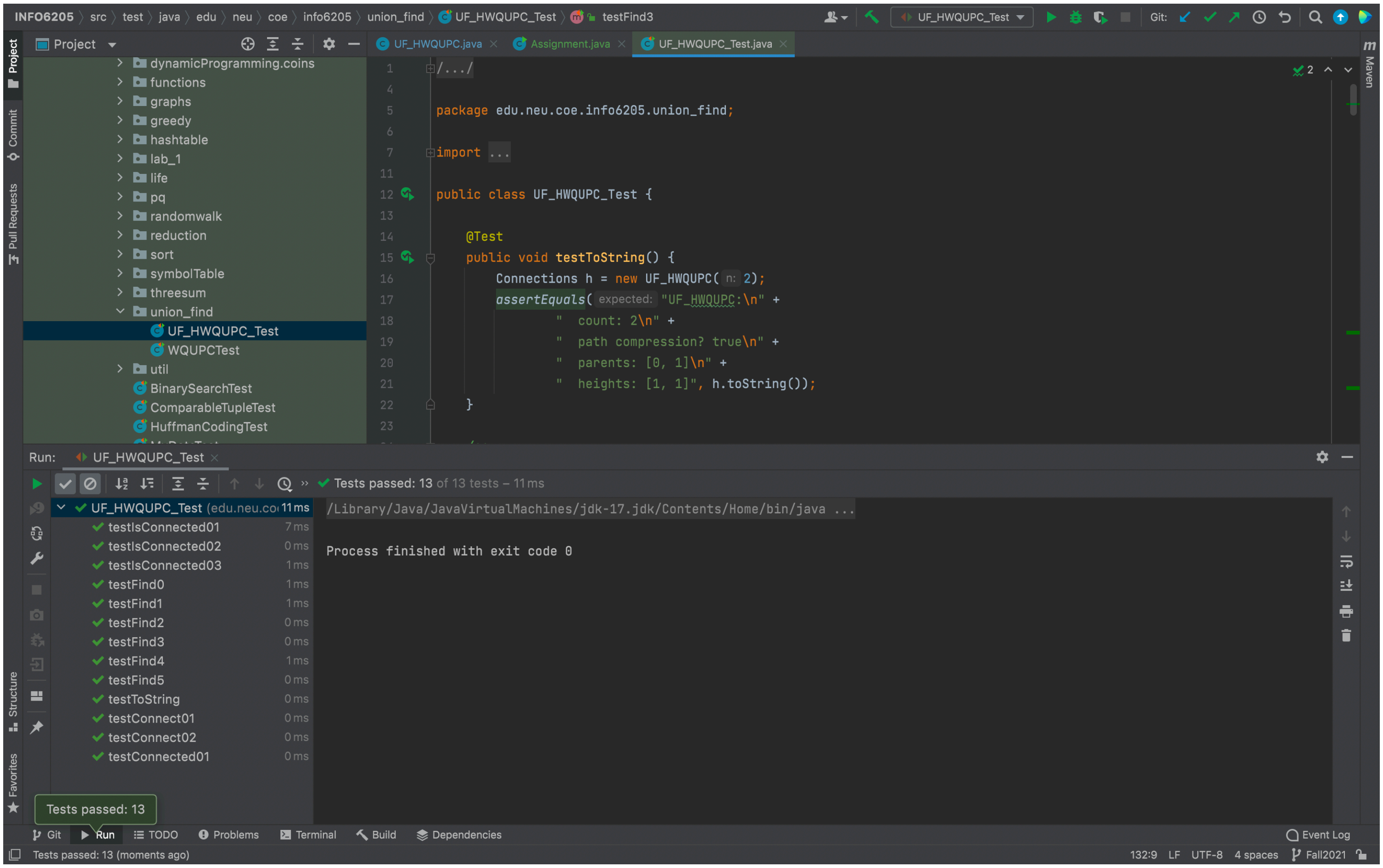
By generating random number of sites for 1000 times, we can observe that every time m=n-1 because we generate pairs only when those two sites are not connected. Thus we generate only those pairs, whose connection has not been established yet.



test

i	n	m
0	760	759
1	932	931
2	437	436
3	141	140
4	426	425
5	318	317
6	547	546
7	36	35
8	66	65
9	132	131
10	724	723
11	360	359
12	373	372
13	113	112
14	752	751
15	863	862
16	743	742
17	711	710
18	37	36
19	94	93
20	102	101
21	885	884
22	787	786

- **Unit tests result:(Snapshot of successful unit test run)**



```java
/.../

package edu.neu.coe.info6205.union_find;

import ...

public class UF_HWQUPC_Test {

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

✓ Tests passed: 13 of 13 tests – 11 ms

```
✓ UF_HWQUPC_Test (edu.neu.coe  11 ms
    ✓ testIsConnected01        7 ms
    ✓ testIsConnected02        0 ms
    ✓ testIsConnected03        1 ms
    ✓ testFind0                1 ms
    ✓ testFind1                1 ms
    ✓ testFind2                0 ms
    ✓ testFind3                0 ms
    ✓ testFind4                1 ms
    ✓ testFind5                0 ms
    ✓ testToString             0 ms
    ✓ testConnect01            0 ms
    ✓ testConnect02            0 ms
    ✓ testConnected01          0 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java ...

Process finished with exit code 0
```

Tests passed: 13

Tests passed: 13 (moments ago)

Assignment.java

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

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Random;
import java.util.Scanner;
import java.util.concurrent.ThreadLocalRandom;

public class Assignment {


    static boolean isCompletelyConnected(int n,UF_HWQUPC uf){ //this function checks whether all the nodes are completely connected
        int temp=uf.find(0);
        int tempCount=n;
        for(int i=1;i<n;i++){
            if(uf.find(i)!=temp){
                break;
            }else{
                tempCount--;
            }
        }
        return tempCount==1;

    }
    static int count(int n){
        int count=0;
        UF_HWQUPC uf=new UF_HWQUPC(n);
        while(!isCompletelyConnected(uf.size(),uf)){
            int p= (int) (Math.random()*n);
            int q= (int) (Math.random()*n);
            if(!uf.connected(p,q)){
//                System.out.println("p="+p+" q="+q);
                uf.connect(p,q);
                count++;
            }
        }
//        System.out.println("Root is: "+uf.find(0));
        return count;
    }
    public static void main(String[] args) {
//        Scanner sc=new Scanner(System.in);
//        System.out.println("Enter number of sites: ");
//        int n=sc.nextInt();
        int n=0;
        try(PrintWriter writer = new PrintWriter(new File("test.csv"))){
            StringBuilder sbh = new StringBuilder();
            sbh.append("i,n,m\n");
            writer.write(sbh.toString());
            for(int i=0;i<1000;i++){
                n = ((int)(Math.random()*1000))+1;
                System.out.println("number of sites (n) = "+n);
                StringBuilder sb = new StringBuilder();
                sb.append(String.valueOf(i));
                sb.append(",");
                sb.append(String.valueOf(n));
                sb.append(",");
                sb.append(String.valueOf(count(n)));
                System.out.println("number of pairs generated (m) = "+count(n));
                System.out.println();
                sb.append("\n");
                writer.write(sb.toString());
//                System.out.println("Number of pairs generated: "+ count(n));
//                System.out.println();
//                  if(n-1!=count(n)) System.out.println("n =
"+n+"count(n) = "+count(n));
//                  else System.out.println("Not found");
            }
            System.out.println("Successfully saved values
to csv file");

        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
        }


    }
}
```