Program Structure and Algorithms

1)Implement height-weighted Quick Union with Path Compression.

```
 */
public int find(int p) {
    validate(p);
    int root = p;
    while (root != parent[root]) {
        if (this.pathCompression) {
            doPathCompression(root);
        }
        root = parent[root];
    }
    return root;
}
```

```java
    private void mergeComponents(int i, int j) {
        // FIXME make shorter root point to taller one
        // END
        int x = parent[i];
        int y = parent[j];
        if (height[x] >= height[y]) {
            parent[y] = parent[x];
            height[x] += height[y];
        } else {
            parent[x] = y;
            height[y] += height[x];
        }

    }

    /**
     * This implements the single-pass path-halving mecha
     */
    private void doPathCompression(int i) {
        // FIXME update parent to value of grandparent
        // END
        parent[i] = parent[parent[i]];
    }
```

```java
/.../

package edu.neu.coe.info6205.union_find;

import ...

public class UF_HWQUPC_Test {

    @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());
    }
```

Tests passed: 13 of 13 tests – 3 ms

```
/Users/ap/Library/Java/JavaVirtualMachines/openjdk-17.0.2/Contents/Home/bin/java ...

Process finished with exit code 0
```

```java
    @Test
    public void testFind0() {
        WQUPC h = new WQUPC( n: 10);
        assertEquals( expected: 0, h.find( p: 0));
    }

    /**
     *
     */
    @Test
    public void testFind1() {
        WQUPC h = new WQUPC( n: 10);
        h.union( p: 0,  q: 1);
        assertEquals( expected: 0, h.find( p: 0));
        assertEquals( expected: 0, h.find( p: 1));
    }

    /**
```

Tests passed: 6 of 6 tests – 1 ms

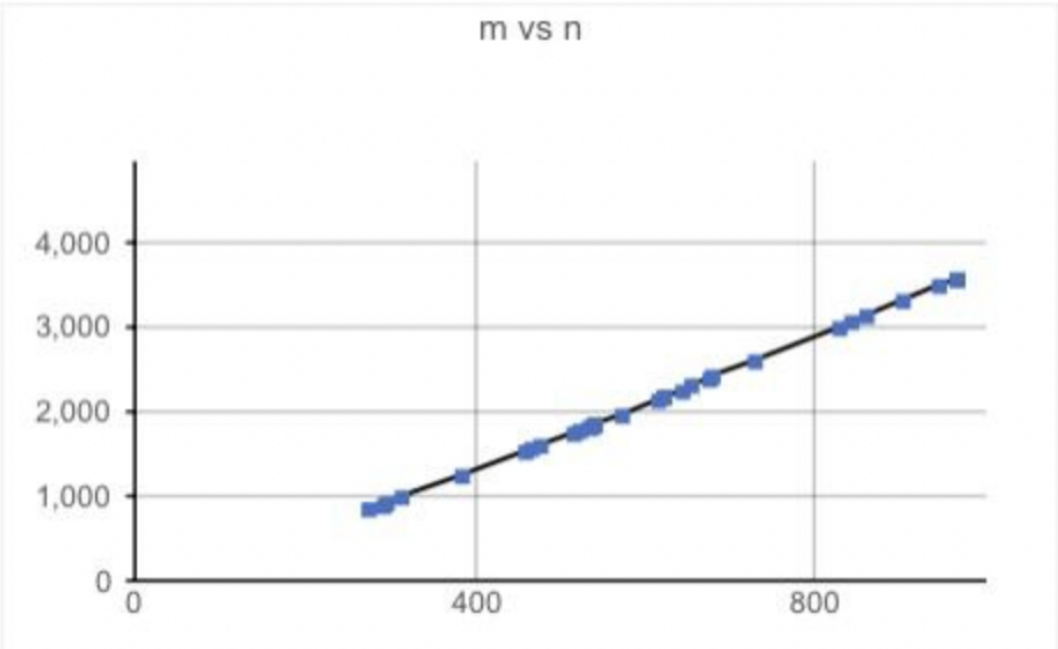
```
/Users/ap/Library/Java/JavaVirtualMachines/openjdk-17.0.2/Contents/Home/bin/java ...

Process finished with exit code 0
```

2.Use WQUPC class to get multiple values of generated pairs for N components

```java
    public static void main(String args[]) {

        Random rand = new Random();
        for (int i = 0; i < 30; i++) {

            int conCount = 0;
            Random random = new Random();
            int J =rand.nextInt( origin: 300, bound: 1000);
            WQUPC wqupc = new WQUPC(J);
            while (wqupc.count > 1) {
                int x = random.nextInt(J);
                int y = random.nextInt(J);
                wqupc.union(x, y);
                conCount++;
            }
            System.out.println(J + " sites are generating " + conCount + " connections ");

        }
    }

}
```

3.Deriving a relation between number of objects(N), and number of Pairs(M) generated.

```
473 sites are generating 1156 connections
460 sites are generating 1674 connections
116 sites are generating 274 connections
330 sites are generating 891 connections
420 sites are generating 1833 connections
220 sites are generating 601 connections
488 sites are generating 2189 connections
406 sites are generating 981 connections
110 sites are generating 196 connections
277 sites are generating 919 connections
163 sites are generating 480 connections
257 sites are generating 837 connections
178 sites are generating 485 connections
113 sites are generating 259 connections
277 sites are generating 953 connections
320 sites are generating 919 connections
291 sites are generating 787 connections
278 sites are generating 897 connections
455 sites are generating 1327 connections
214 sites are generating 528 connections
185 sites are generating 388 connections
165 sites are generating 425 connections
333 sites are generating 1033 connections
481 sites are generating 1416 connections
471 sites are generating 1572 connections
384 sites are generating 1576 connections
236 sites are generating 694 connections

Process finished with exit code 0
```



Conclusion:
Relationship is– M=(0.5)*N(log(N))
I made several runs and plotted the graph to see the above result.
I calculated the average value of the above function for all values and it is equal to the above function.