Program Structures and Algorithms

Spring 2024

Name : Shaofan Wei

NUID: 002815198

GITHUB LINK:https://github.com/Arthurccone123/INFO6205

Task:

This task is divided into three steps:

Step 1: Implement the UF_HWQUPC Class

Implement the height-weighted Quick Union with Path Compression in the `UF_HWQUPC` class. Conduct unit tests to ensure that all functionalities work as expected.

Step 2: Develop a UF Client Program

Using your implementation of the `UF_HWQUPC` class, develop a client program. The program should accept an integer `n` from the command line, which represents the number of sites. The program will generate random pairs of integers `m`, ranging from 0 to `n-1`, and use the `connected()` method to check if they are already connected. If not, it will use the `union()` method to connect them. Continue this process in a loop, and then print the number of connections made.

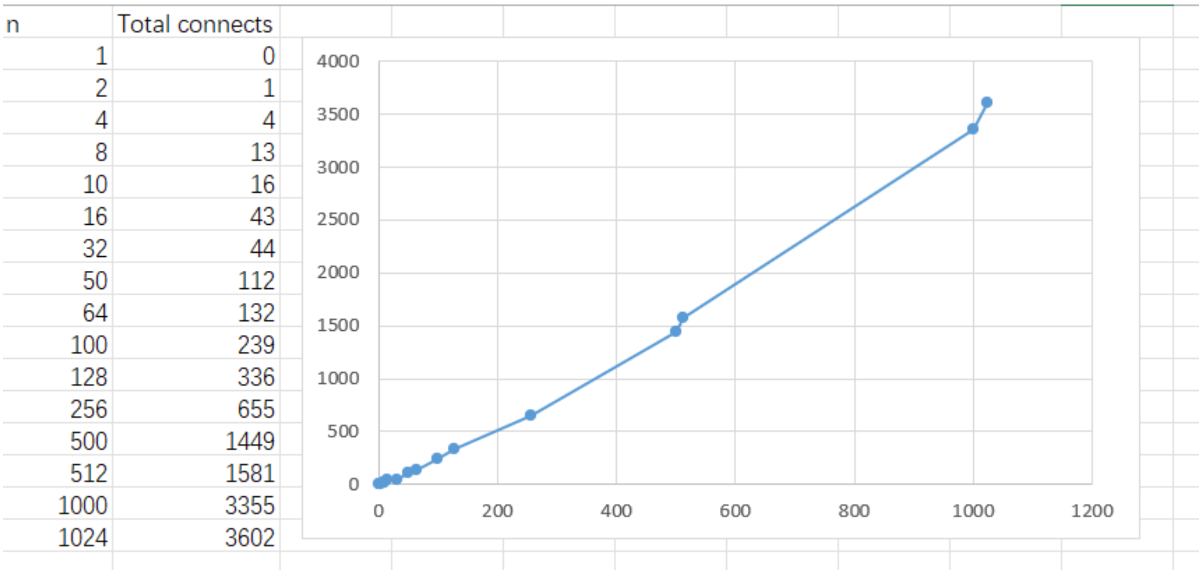
Step 3: Determine the Relationship Between the Number of Objects (n) and the Number of Pairs (m)

Analyze and determine the relationship between the number of objects (`n`) and the number of pairs (`m`) needed to reduce the component count from `n` to 1.

Relationship Conclusion:

The relationship between (m) and (n) tends to exhibit an almost linear relationship.

Evidence to support that conclusion:



n	Total connects
1	0
2	1
4	4
8	13
10	16
16	43
32	44
50	112
64	132
100	239
128	336
256	655
500	1449
512	1581
1000	3355
1024	3602

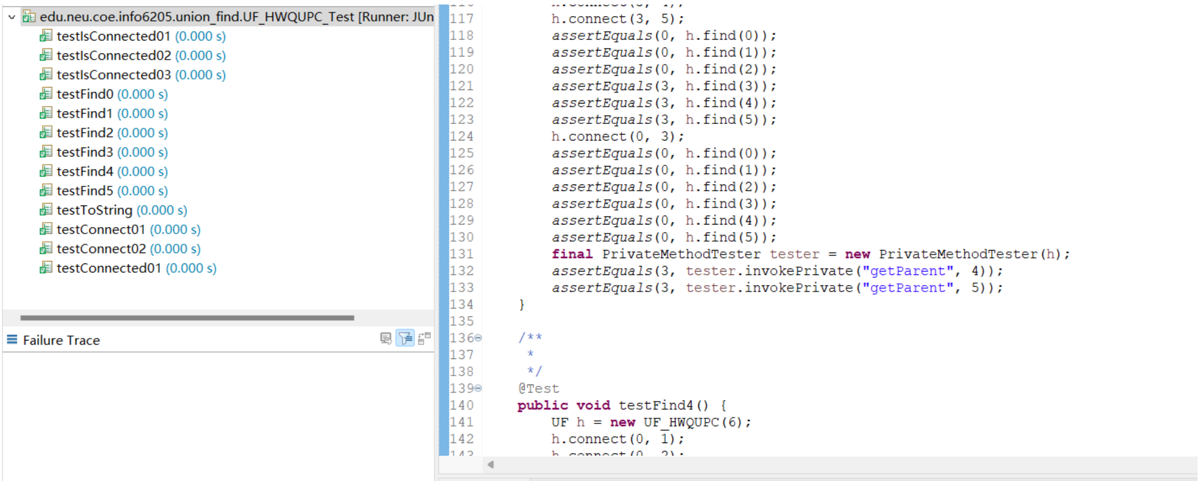
Here is the code:

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

```
 private void mergeComponents(int i, int j) {
         if (height[i] < height[j]) {
                parent[i] = j;
            } else if (height[i] > height[j]) {
                parent[j] = i;
            } else {
                parent[j] = i;
                height[i]++;
            }
    }
```

```
private void doPathCompression(int i) {
        while (i != parent[i]) {
            parent[i] = parent[parent[i]];
            i = parent[i];
        }
    }
```

Unit Test Screenshots:



edu.neu.coe.info6205.union_find.UF_HWQUPC_Test [Runner: JUn
- testIsConnected01 (0.000 s)
- testIsConnected02 (0.000 s)
- testIsConnected03 (0.000 s)
- testFind0 (0.000 s)
- testFind1 (0.000 s)
- testFind2 (0.000 s)
- testFind3 (0.000 s)
- testFind4 (0.000 s)
- testFind5 (0.000 s)
- testToString (0.000 s)
- testConnect01 (0.000 s)
- testConnect02 (0.000 s)
- testConnected01 (0.000 s)

```
        h.connect(3, 4);
        h.connect(3, 5);
        assertEquals(0, h.find(0));
        assertEquals(0, h.find(1));
        assertEquals(0, h.find(2));
        assertEquals(3, h.find(3));
        assertEquals(3, h.find(4));
        assertEquals(3, h.find(5));
        h.connect(0, 3);
        assertEquals(0, h.find(0));
        assertEquals(0, h.find(1));
        assertEquals(0, h.find(2));
        assertEquals(0, h.find(3));
        assertEquals(0, h.find(4));
        assertEquals(0, h.find(5));
        final PrivateMethodTester tester = new PrivateMethodTester(h);
        assertEquals(3, tester.invokePrivate("getParent", 4));
        assertEquals(3, tester.invokePrivate("getParent", 5));
    }

    /**
     *
     */
    @Test
    public void testFind4() {
        UF h = new UF_HWQUPC(6);
        h.connect(0, 1);
        h.connect(0, 2);
```