

Program Structures & Algorithms

Spring 2022

Assignment No - 3 (WQUPC)

Name: Naina Rajan
NUID: 002922398

Task -

Step 1:

(a) Implement height-weighted Quick Union with Path Compression. For this, you will flesh out the class UF_HWQUPC. All you have to do is to fill in the sections marked with `// TO BE IMPLEMENTED ... // ...END IMPLEMENTATION`.

(b) Check that the unit tests for this class all work. You must show "green" test results in your submission (screenshot is OK).

Step 2:

Using your implementation of UF_HWQUPC, develop a UF ("union-find") client that takes an integer value n from the command line to determine the number of "sites." Then generates random pairs of integers between 0 and $n-1$, calling `connected()` to determine if they are connected and `union()` if not. Loop until all sites are connected then print the number of connections generated. Package your program as a static method `count()` that takes n as the argument and returns the number of connections; and a `main()` that takes n from the command line, calls `count()` and prints the returned value. If you prefer, you can create a main program that doesn't require any input and runs the experiment for a fixed set of n values. Show evidence of your run(s).

Step 3:

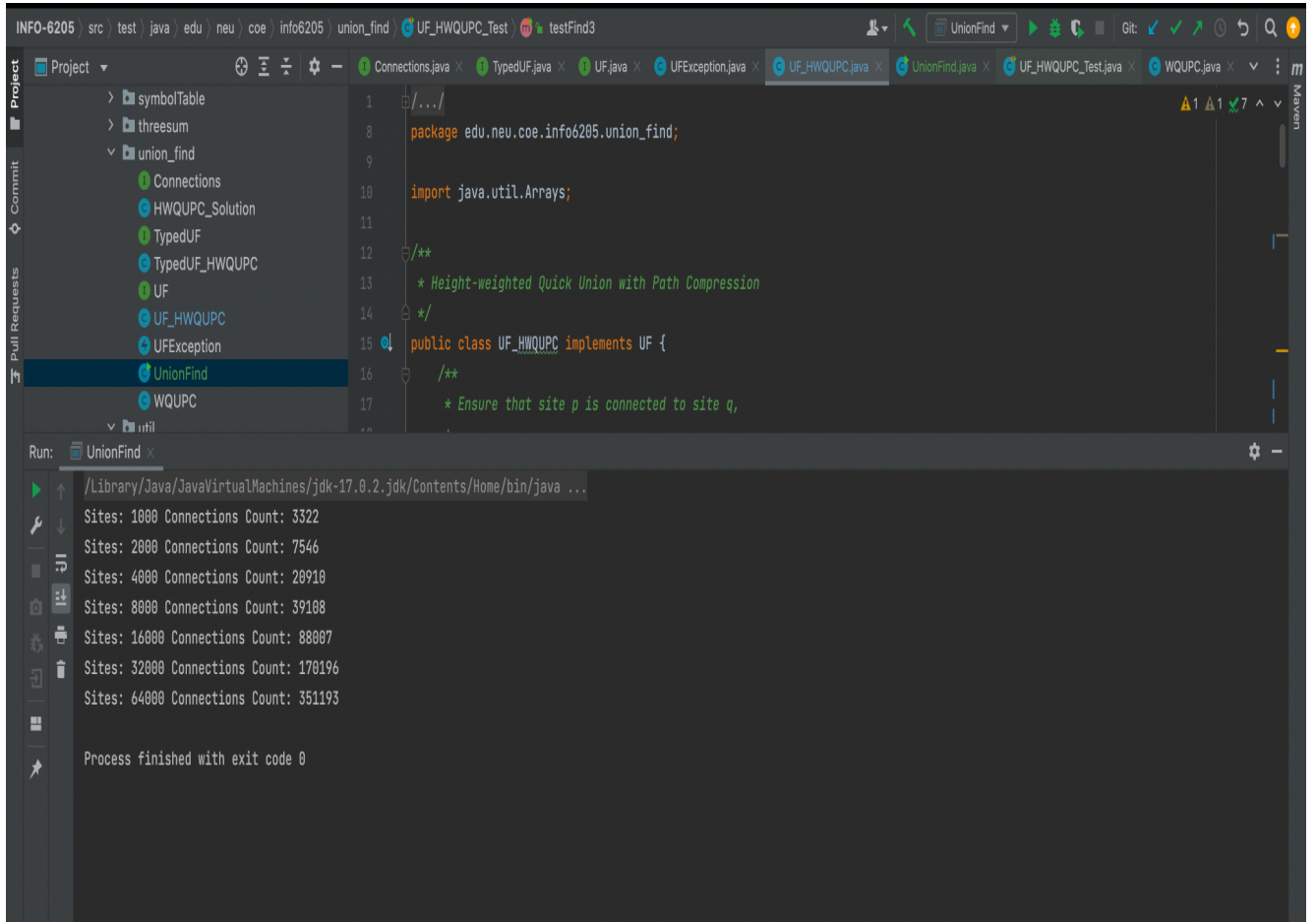
Determine the relationship between the number of objects (n) and the number of pairs (m) generated to accomplish this (i.e. to reduce the number of components from n to 1). Justify your conclusion in terms of your observations and what you think might be going on.

NOTE: although I'm not going to tell you in advance what the relationship is, I can assure you that it is a *simple* relationship.

Don't forget to follow the submission guidelines. And to use sufficient (and sufficiently large) different values of n .

Output Screenshot

- Implemented the functions **mergeComponents()**, **doPathCompression()** and **find()**
- Successfully ran the code



The screenshot displays an IDE with a project named 'union_find' and a file named 'UnionFind.java'. The code implements a Union-Find data structure with path compression. The output window shows the results of running the program, which tests the structure with increasing numbers of sites and connections.

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

import java.util.Arrays;

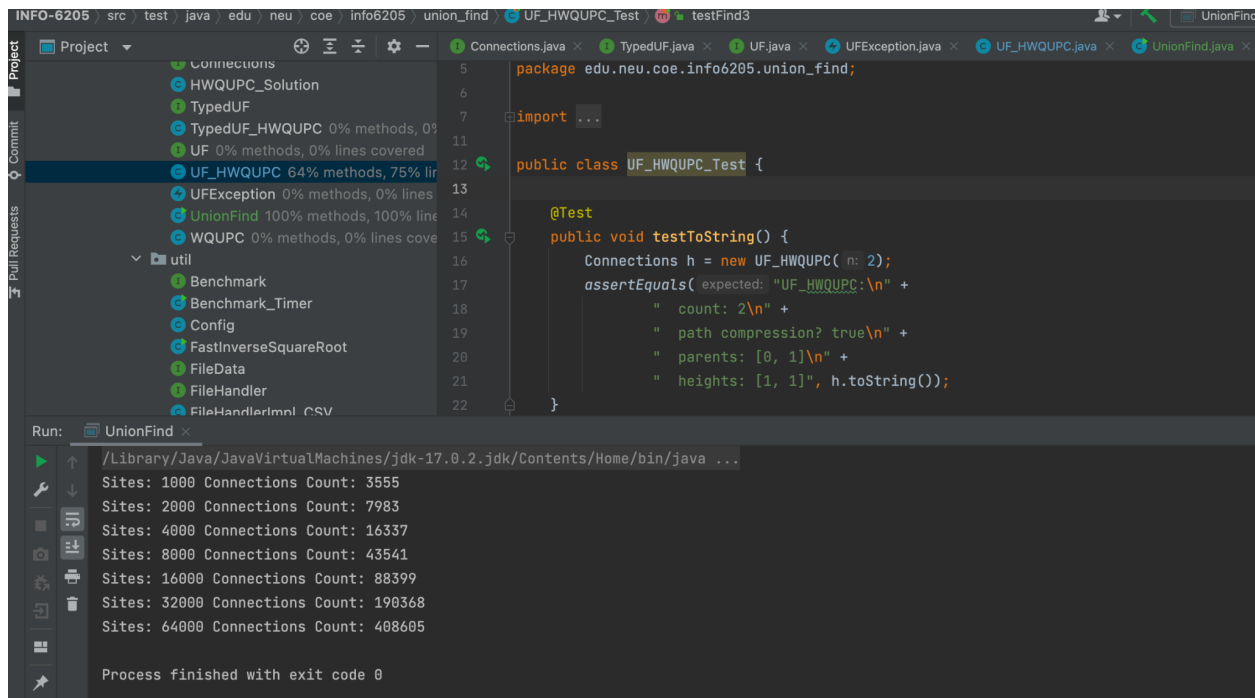
/**
 * Height-weighted Quick Union with Path Compression
 */
public class UF_HWQUPC implements UF {
    /**
     * Ensure that site p is connected to site q,
     */
}
```

Run: UnionFind x

```
/Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home/bin/java ...
Sites: 1000 Connections Count: 3322
Sites: 2000 Connections Count: 7546
Sites: 4000 Connections Count: 28918
Sites: 8000 Connections Count: 39108
Sites: 16000 Connections Count: 88087
Sites: 32000 Connections Count: 170196
Sites: 64000 Connections Count: 351193

Process finished with exit code 0
```

Relationship Conclusion



The screenshot shows an IDE with a project named 'INFO-6205'. The project structure includes a 'util' package with classes like 'Benchmark', 'Benchmark_Timer', 'Config', 'FastInverseSquareRoot', 'FileData', 'FileHandler', and 'FileHandlerImpl'. The 'Run' console shows the output of a Java application, displaying the number of connections for various site counts. The test class 'UF_HWQUPC_Test' is shown with a test method 'testToString()' that asserts the output of 'UF_HWQUPC' against a string representation of the connections.

```
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());
    }
}
```

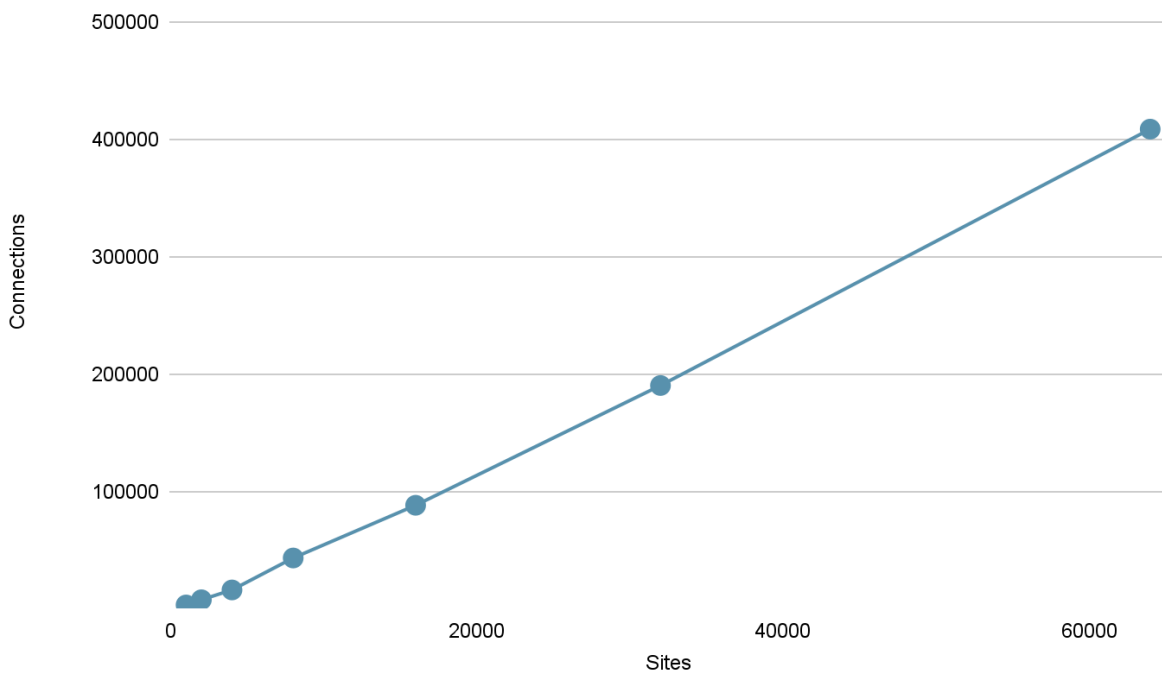
Run: UnionFind x

/Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home/bin/java ...

Sites: 1000 Connections Count: 3555
Sites: 2000 Connections Count: 7983
Sites: 4000 Connections Count: 16337
Sites: 8000 Connections Count: 43541
Sites: 16000 Connections Count: 88399
Sites: 32000 Connections Count: 190368
Sites: 64000 Connections Count: 408605

Process finished with exit code 0

Sites Vs Connections



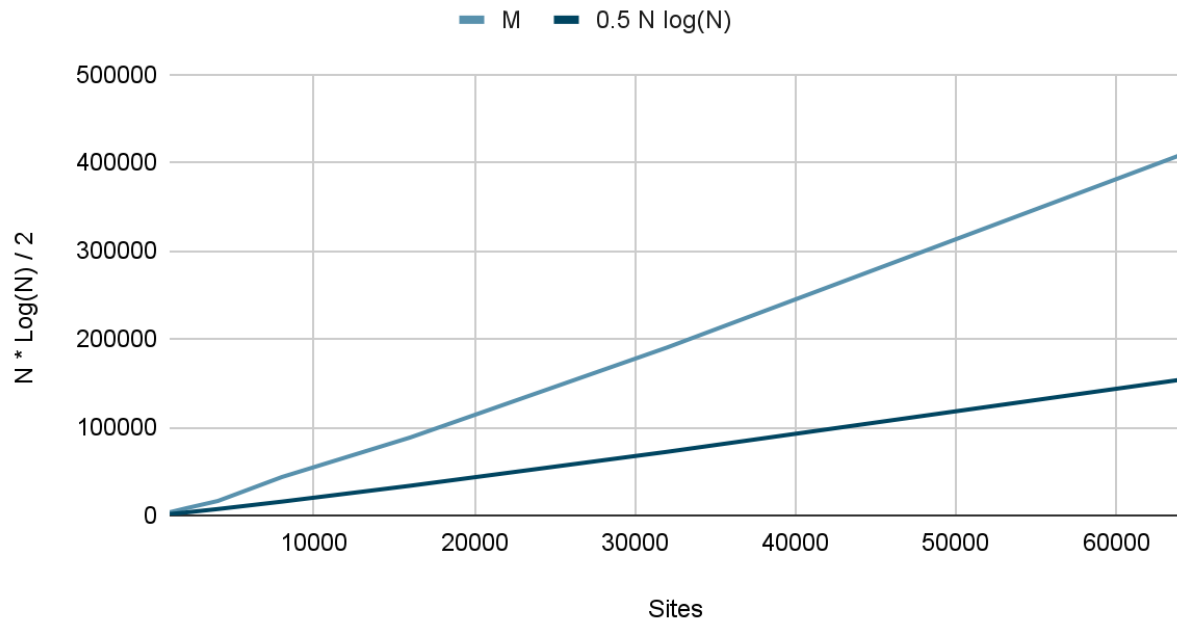
1. From the above Output Screenshot and Table, we can conclude:

Relationship between n(number of sites) and m(number of connections)is: $M = (N * \log(N)) / 2$

The relationship is linear. As the number of sites grow, the number of connections grow linearly.

Evidence/Graph

Sites v/s $N * \log(N) / 2$

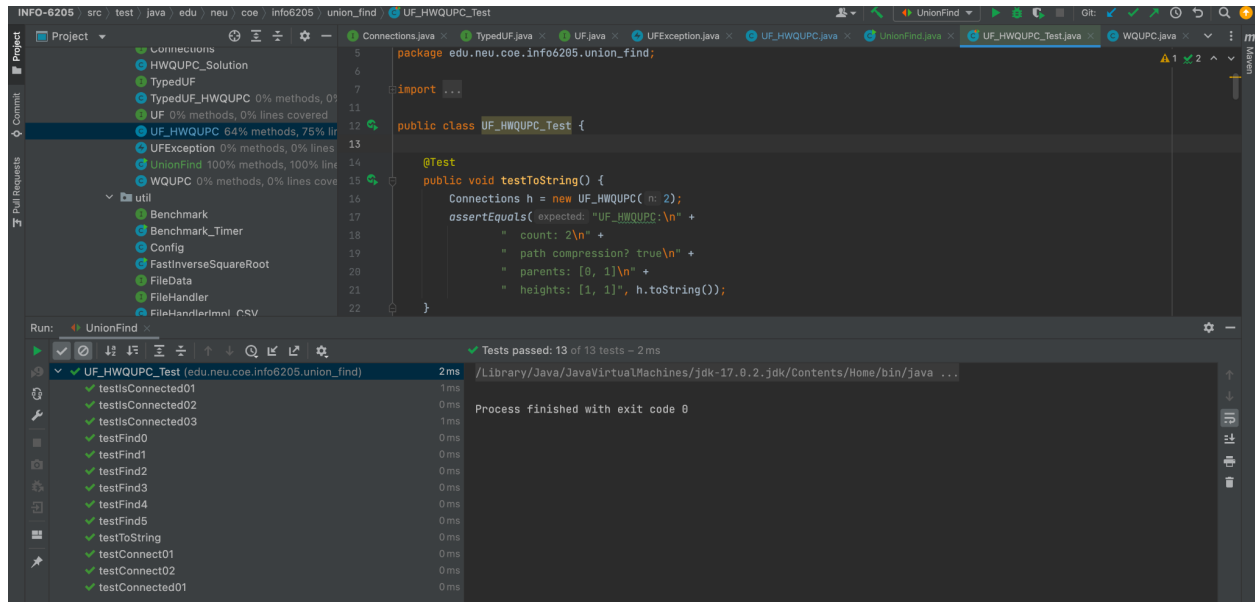


Hence the relation should be $M = (N * \log(N)) / 2$

Where M = Connection Count and N = Sites

Sites	M	0.5 N log(N)
1000	3555	1500
2000	7983	3301
4000	16337	7204
8000	43541	15612
16000	88399	33633
32000	190368	72083
64000	408605	153798

Unit Tests Result



```
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());
    }
}
```

Run: UnionFind

Tests passed: 13 of 13 tests - 2ms

Test Name	Duration
UF_HWQUPC_Test (edu.neu.coe.info6205.union_find)	2ms
testIsConnected01	1ms
testIsConnected02	0ms
testIsConnected03	1ms
testFind0	0ms
testFind1	0ms
testFind2	0ms
testFind3	0ms
testFind4	0ms
testFind5	0ms
testToString	0ms
testConnect01	0ms
testConnect02	0ms
testConnect03	0ms

Process finished with exit code 0

Git Repository -

<https://github.com/Naina-NEU/INFO6205/commit/930e76142b4ed7bfab481176257c6381cc0cd81f>