

Program Structures and Algorithms  
Spring 2024

NAME: Ashish Nevan Gade

NUID: 002889005

GITHUB LINK: <https://github.com/AshishNevan/INFO6205>

**Task:**

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

**Relationship Conclusion:**

The relationship between the number of objects ( $n$ ) and the number of pairs ( $m$ ) generated to connect all objects and reduce the number of components to 1 is linear. For ' $n$ ' objects, ' $n-1$ ' pairs are generated. This linear relationship is expected because each pair of objects (sites) forms one union operation, and ' $n-1$ ' such operations are required to connect all objects into one component.

**Evidence to support that conclusion:**

In the union-find algorithm, we aim to connect all the objects (sites) until they form a single component. We repeatedly generate pairs of integers and use the `connected()` and `union()` operations to connect them until there is only one component left.

Each time we call `union()` operation, it merges two components into one. Initially, we have ' $n$ ' components (one component for each object). As we perform union operations, the number of components decreases until there is only one component left, indicating that all objects are connected.

The number of pairs ( $m$ ) generated to accomplish this can be analyzed as follows:

- Initially, we have ' $n$ ' components.
- With each `union()` operation, the number of components decreases by 1.
- So, after ' $n-1$ ' union operations, there will be only one component left.
- Therefore, the number of pairs ( $m$ ) generated to reduce the number of components from ' $n$ ' to 1 is ' $n-1$ '.

Number of Components (N)	Number of Connections
500	499
1000	999
2000	1999
4000	3999
8000	7999
16000	15999
32000	31999
64000	63999

Figure 1: Number of components vs Number of connections

The screenshot shows an IDE with the project 'INFO6205' and the file 'UnionFindClient.java' open. The code defines a 'UnionFindClient' class with a 'count' method that generates a random graph with 'n' components and 'connections' edges. The output window shows the results of running the client for various values of 'n' (500, 1000, 2000, 4000, 8000, 16000, 32000, 64000), each resulting in the expected number of connections (n-1). The process finished with exit code 0.

Figure 2: Union Find Client

Unit Test Screenshots:

The screenshot shows an IDE with the project 'INFO6205' and the file 'UF\_HWQUPC\_Test.java' open. The code defines a 'UF\_HWQUPC\_Test' class with a 'doPathCompression' method. The output window shows the results of running the tests, which all passed. The process finished with exit code 0.