Program Structures and Algorithms

Spring 2023(SEC 03)

NAME: Vipul Rajderkar

NUID: 002700991

**Task:**

Part-1: Implement height-weighted Quick Union with Path Compression. Test all unit test cases.

Part-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).

Part-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).

**Relationship Conclusion:**

The relationship between the number of objects(n) and the number of connections formed is n-1. As all n nodes can be connected with n-1 links.

In task 3, ‘m’ random pairs of integers between 0 and n-1 were formed, and it was checked if those 2 nodes are connected, and if not then connect them until all nodes are connected. The task was performed for 50 runs and then the average was taken.   
  
The relationship between (n) and (m) can be derived as,  
**m = c x n x log n**, Where c ≈ 1.1

Thus,  
**m ∝ n log n**

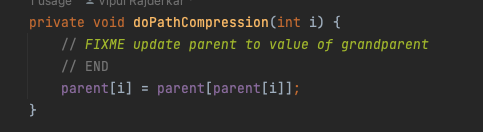
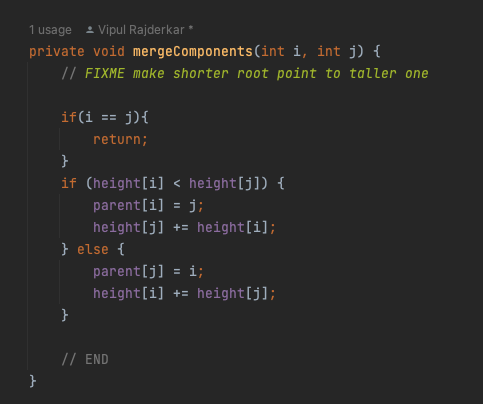
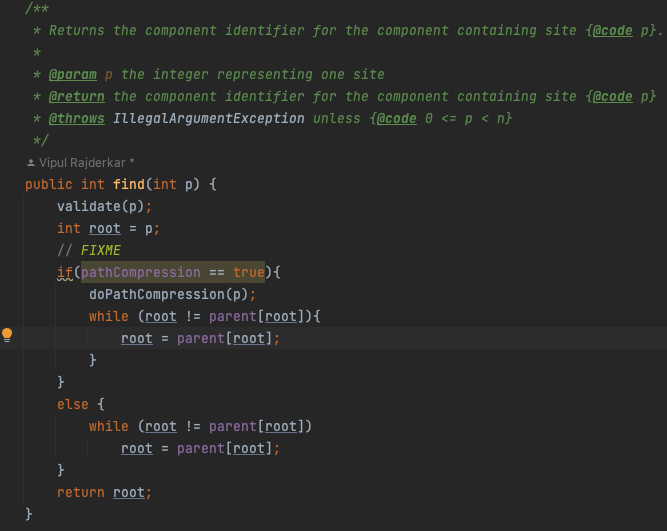
**Evidence to support that conclusion:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **n (total number of sites)** | **m (total number of pairs generated)** | **log n** | **n log n** | **m/n** | **m/n log n** |
| 1000 | 3802 | 3 | 3000 | 3.802 | 1.26733333 |
| 2000 | 8076 | 3.301029996 | 6602.059991 | 4.038 | 1.22325456 |
| 4000 | 18150 | 3.602059991 | 14408.23997 | 4.5375 | 1.25969584 |
| 8000 | 38694 | 3.903089987 | 31224.7199 | 4.83675 | 1.23921048 |
| 16000 | 80904 | 4.204119983 | 67265.91972 | 5.0565 | 1.20274874 |
| 32000 | 183202 | 4.505149978 | 144164.7993 | 5.7250625 | 1.27078178 |
| 64000 | 373664 | 4.806179974 | 307595.5183 | 5.8385 | 1.21479013 |

Code Snippet:

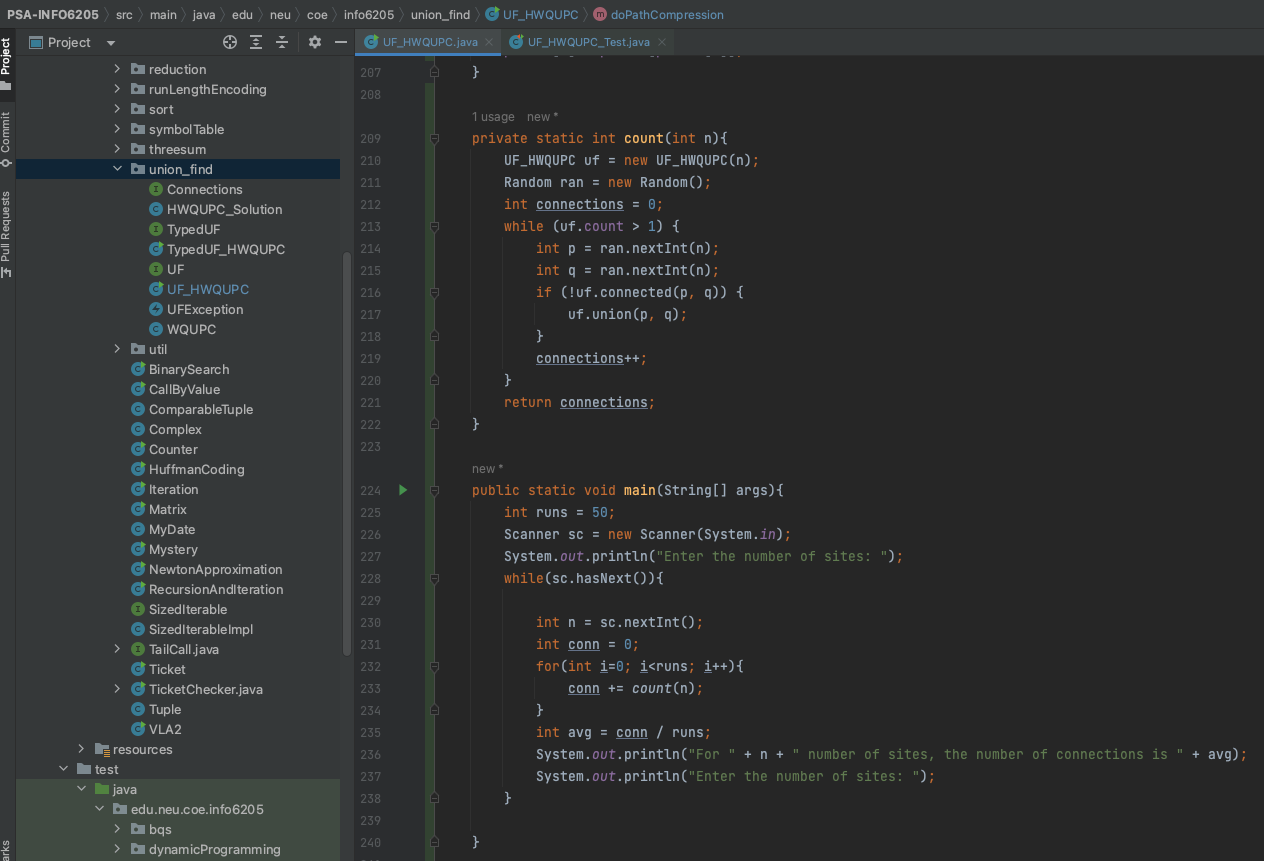
Part 1: Implement *find, mergeComponents*, and *doPathCompression* methods

1. find(): In this method, we check if the given node is connected to the parent root or not. Path compression is done until it is connected.
2. mergeComponents(): If both nodes are not the same, we map the node with a lesser number of connections to a larger number of connections.
3. doPathCompression(): The size of the connected nodes has to be reduced hence we map the current node’s child with the parent root of that node.



Part 2: Implement Client

Code Snippet:



Output:

For UF Client implementation main and count methods have been added in the same class. The count method returns the number of pairs generated. This method has been called for 50 times and then the average has been taken. The main method takes the input from the user for the value of n and then calls the count method.

Text

Description automatically generated

**Graphical Representation:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **n (total number of sites)** | **m (total number of pairs generated)** | **log n** | **n log n** | **m/n** | **m/n log n** |
| 1000 | 3802 | 3 | 3000 | 3.802 | 1.26733333 |
| 2000 | 8076 | 3.301029996 | 6602.059991 | 4.038 | 1.22325456 |
| 4000 | 18150 | 3.602059991 | 14408.23997 | 4.5375 | 1.25969584 |
| 8000 | 38694 | 3.903089987 | 31224.7199 | 4.83675 | 1.23921048 |
| 16000 | 80904 | 4.204119983 | 67265.91972 | 5.0565 | 1.20274874 |
| 32000 | 183202 | 4.505149978 | 144164.7993 | 5.7250625 | 1.27078178 |
| 64000 | 373664 | 4.806179974 | 307595.5183 | 5.8385 | 1.21479013 |

**Unit Test Screenshots:**

**Graphical user interface, text

Description automatically generated**