

Program Structure and Algorithms

Spring 2022

Assignment No. 3

Name: Shreya Patil
(NUID): **002193245**

Tasks implemented:

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).

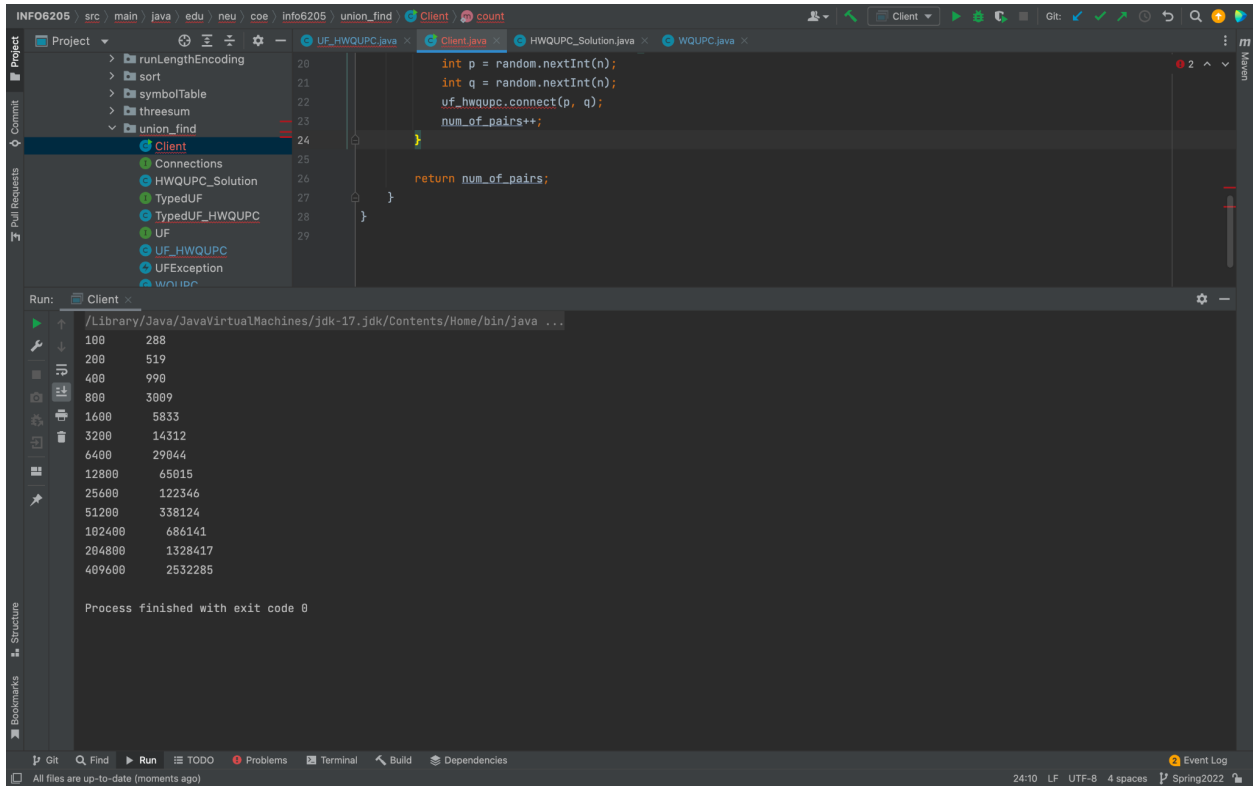
Step 3:

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:



The screenshot shows an IDE with a project named 'info6205'. The 'Client.java' file is open, showing a method that generates random pairs and connects them using 'uf_hwqupc.connect(p, q)'. The 'Run' console shows the output of the program, which is a list of pairs of numbers. The output is as follows:

n	num_of_pairs
100	288
200	519
400	990
800	3009
1600	5833
3200	14312
6400	29044
12800	65015
25600	122346
51200	338124
102400	686141
204800	1328417
409600	2532285

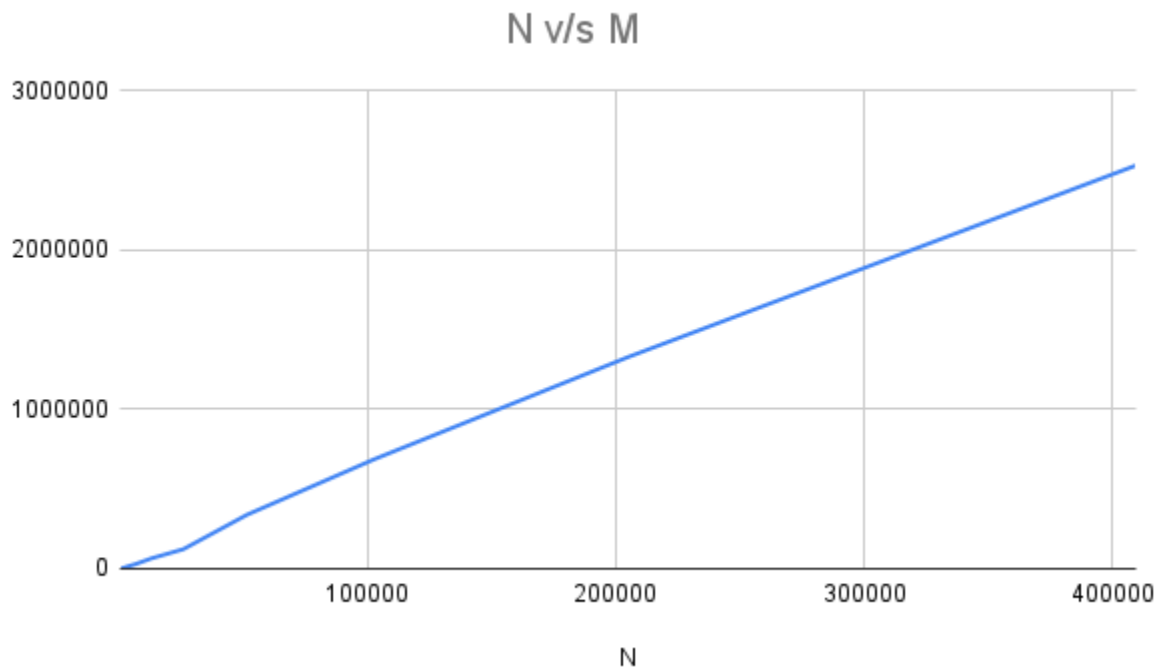
Process finished with exit code 0

Conclusion:

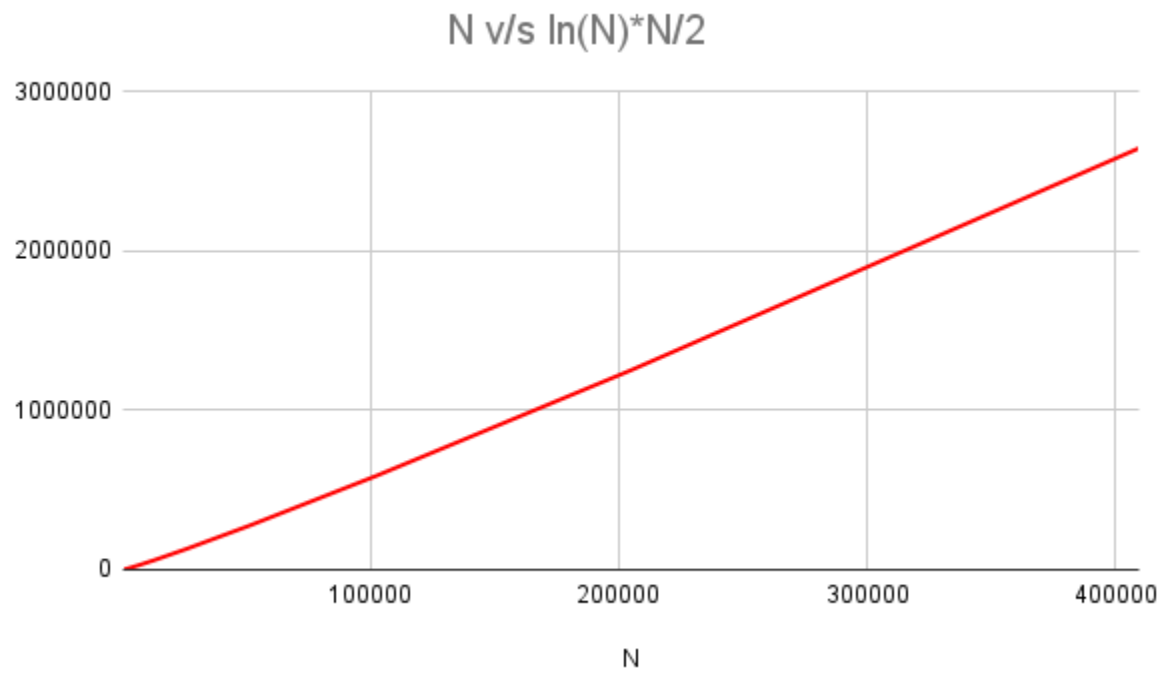
The number of pairs M is approximately equal to $\ln(N) \cdot N/2$, where N is the number of objects.

Evidence:

- Plot of N (number of elements) vs M (number of pairs)

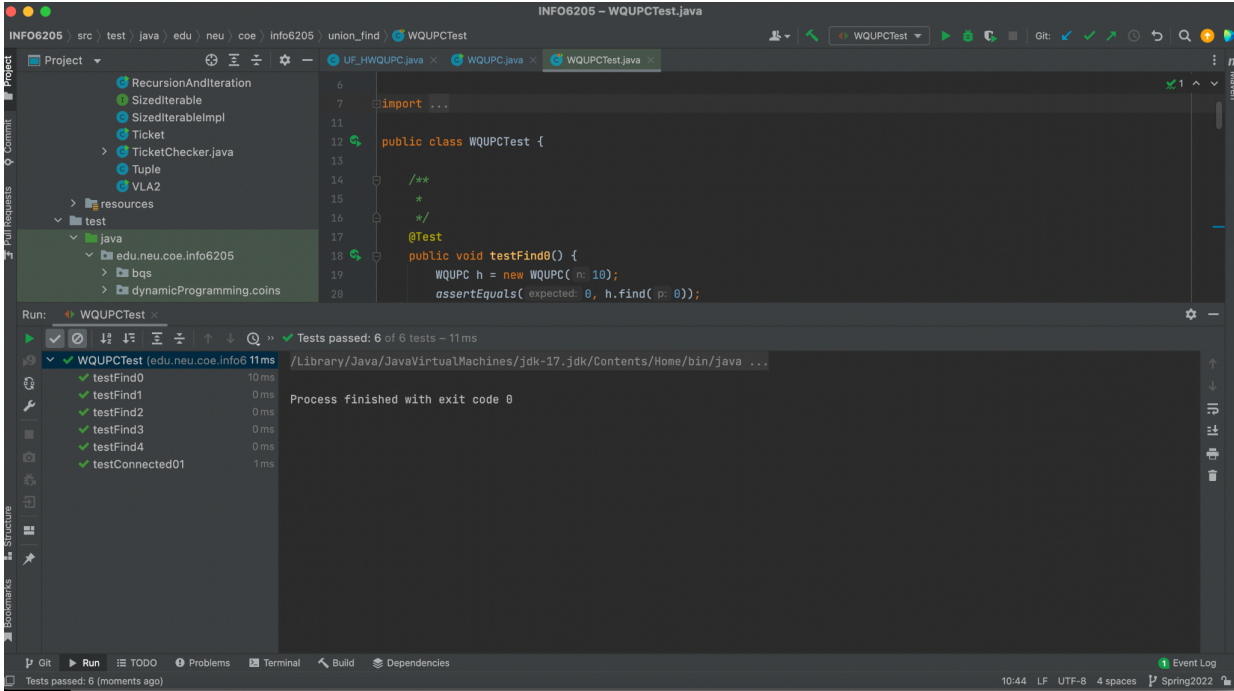


- Plot of N(number of elements) vs $\ln(N) \cdot N/2$



Unit Test cases screenshot:

1. WQUPCTest.java



2. UF_HWQUPC_Test.java

