SAI KUMAR GANGA (002191288)

INFO 6205 PROGRAM STRUCTURES AND ALGORITHMS

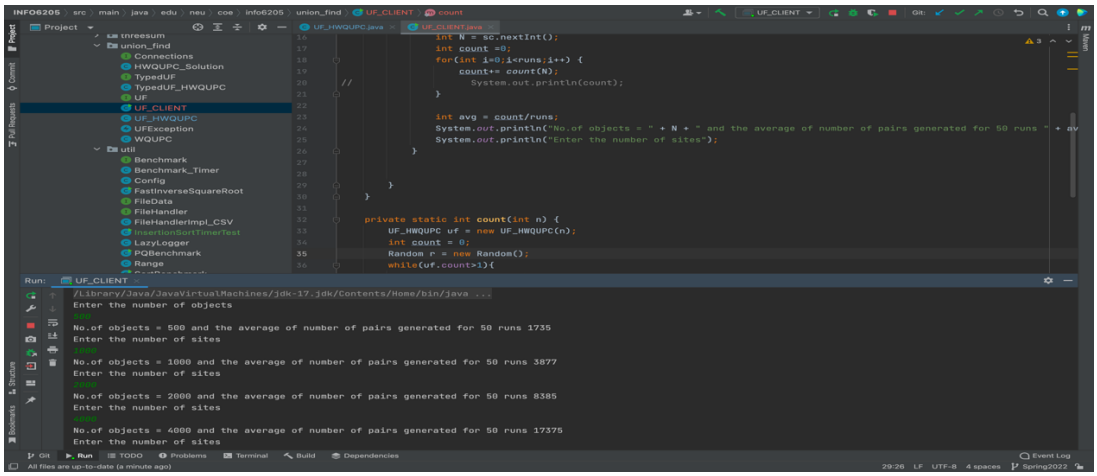
Assignment-3(WQUPC)

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.

Solution:

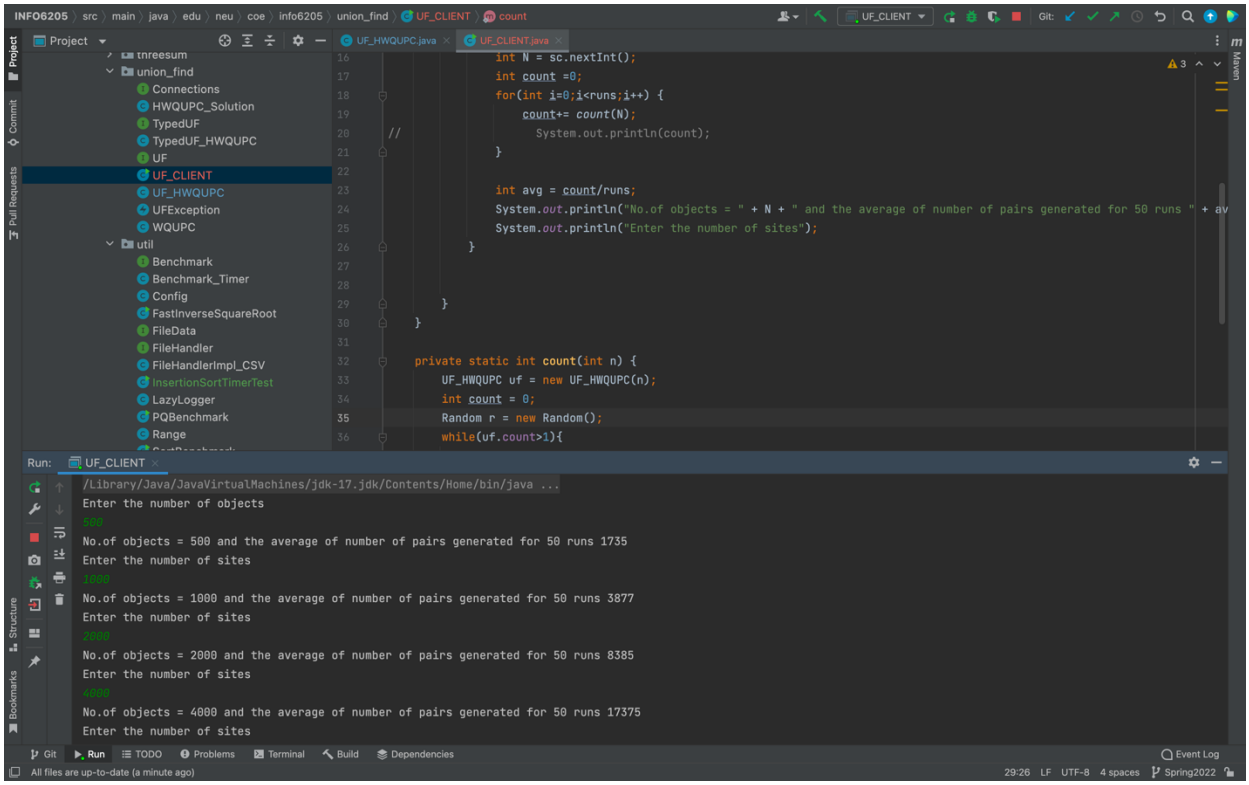
The solution can be accessed through the UF_HWQUPC.java.



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

Proof:

UF_HWQUPC_Test.java is the code for all the unit test cases, and the following screenshot is proof of the test cases being passed

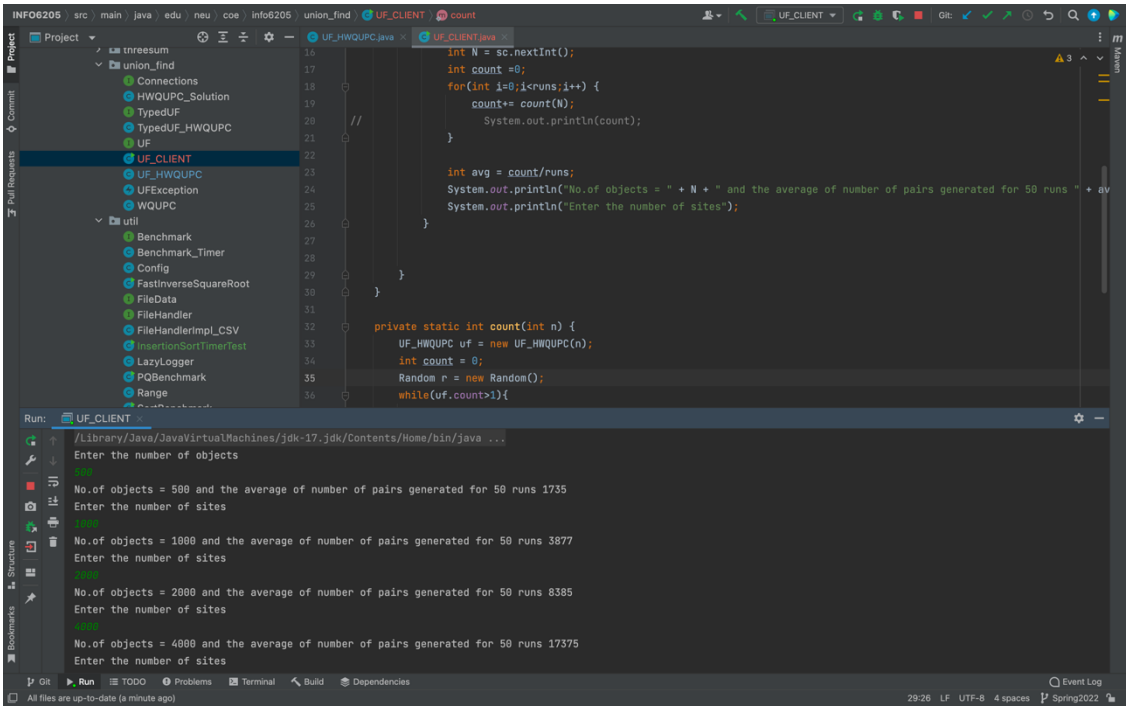


Step 2:

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).

Solution:

UF_CLIENT.java. is the code and following screenshot is the evidence that the given input and the code is executing properly and with the outputs.



Step 3:

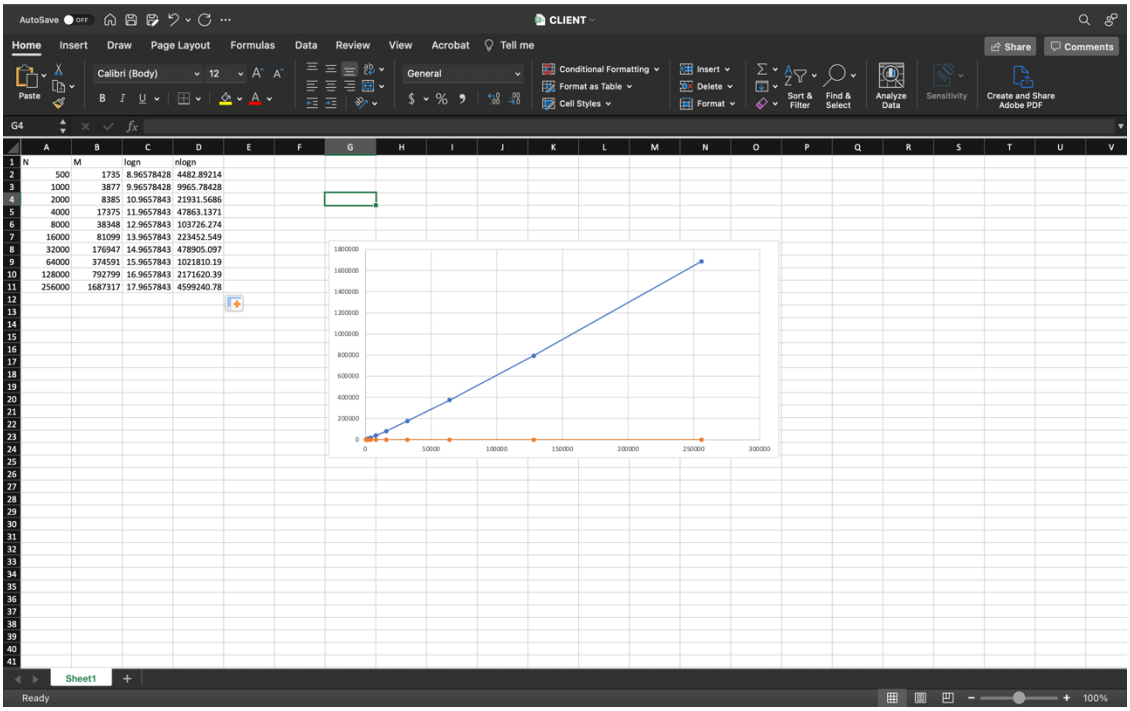
Determine the relationship between the number of objects (*n*) and the number of pairs (*m*) generated to accomplish this (i.e. to reduce the number of components from *n* to 1). Justify your conclusion in terms of your observations and what you think might be going on.

Conclusion and Relationship:

N values ranging from 500 to 256000(doubling every time), for each value of n the program runs 50 times and the avg value of m is noted down. The constant can be approximated, and relationship can be determined as

$$m = \frac{1}{2} * n * \log_2 n$$

Evidence & Graph:



N	M	logn	nlogn
500	1735	8.96578428	4482.89214
1000	3877	9.96578428	9965.78428
2000	8385	10.9657843	21931.5686
4000	17375	11.9657843	47863.1371
8000	38348	12.9657843	103726.274
16000	81099	13.9657843	223452.549
32000	176947	14.9657843	478905.097
64000	374591	15.9657843	1021810.19
128000	792799	16.9657843	2171620.39
256000	1687317	17.9657843	4599240.78