Apoorva Kethamaranahalli(001090928)

Program Structures & Algorithms

Fall 2021

Assignment No. 3

- **Tasks**

1. (a) Implement height-weighted Quick Union with Path Compression. Fill out the sections marked as “TO BE IMPLEMENTED” in the class UF_HWQUPC
1. (b) Check unit test cases all work for this class.

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).

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 randomly generated pairs (m) in order to reduce the number of components/objects from n to 1, in other words in order for all the pairs to be connected can be given as follows :
After taking the average of the value of m over 50 runs of the program(for the same value of n), we can say that

$$m = c * n * \log(n)$$

$$\text{where } c = m/n * \log(n)$$

which is approximately equal to 1.22 as observed over n ranging from 1000 to 256000(doubling)
Therefore we can summarize the relationship as

$$m \propto n * \log(n)$$

(log taken to the base of 2)

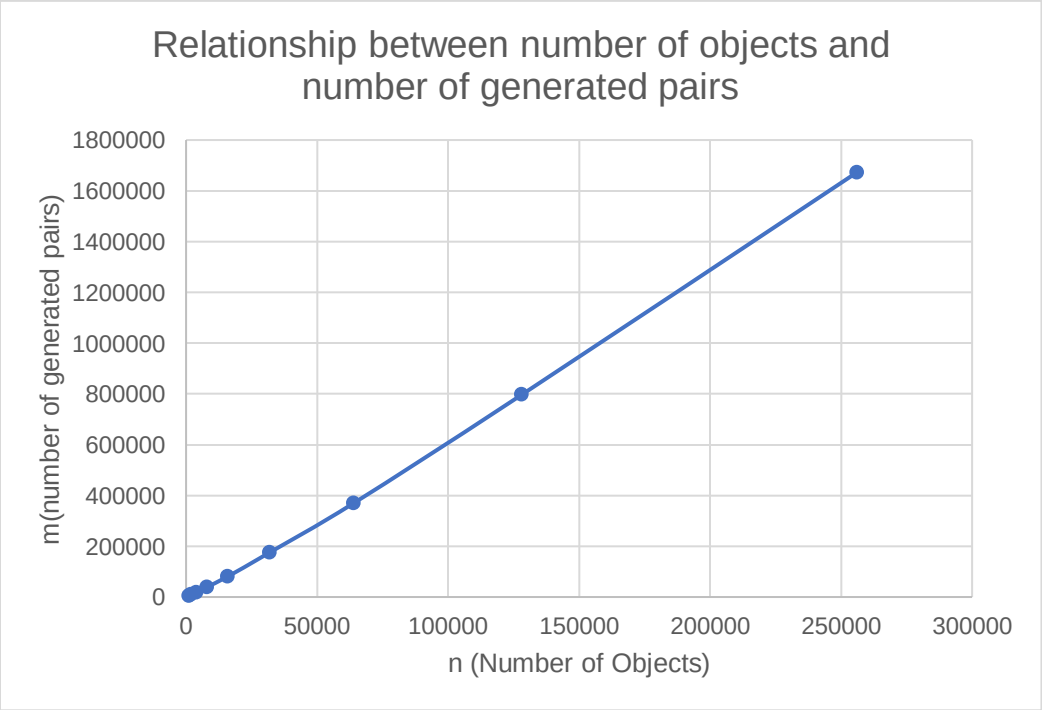
- **Evidence to support the conclusion:**

1. n values ranging from 1000 to 256000(doubling each time)
2. For each value of n, the program is run for 50 times, and the average value of m is noted down as shown below.
3. The constant c = m/n * log(n) can be approximated to 1.22

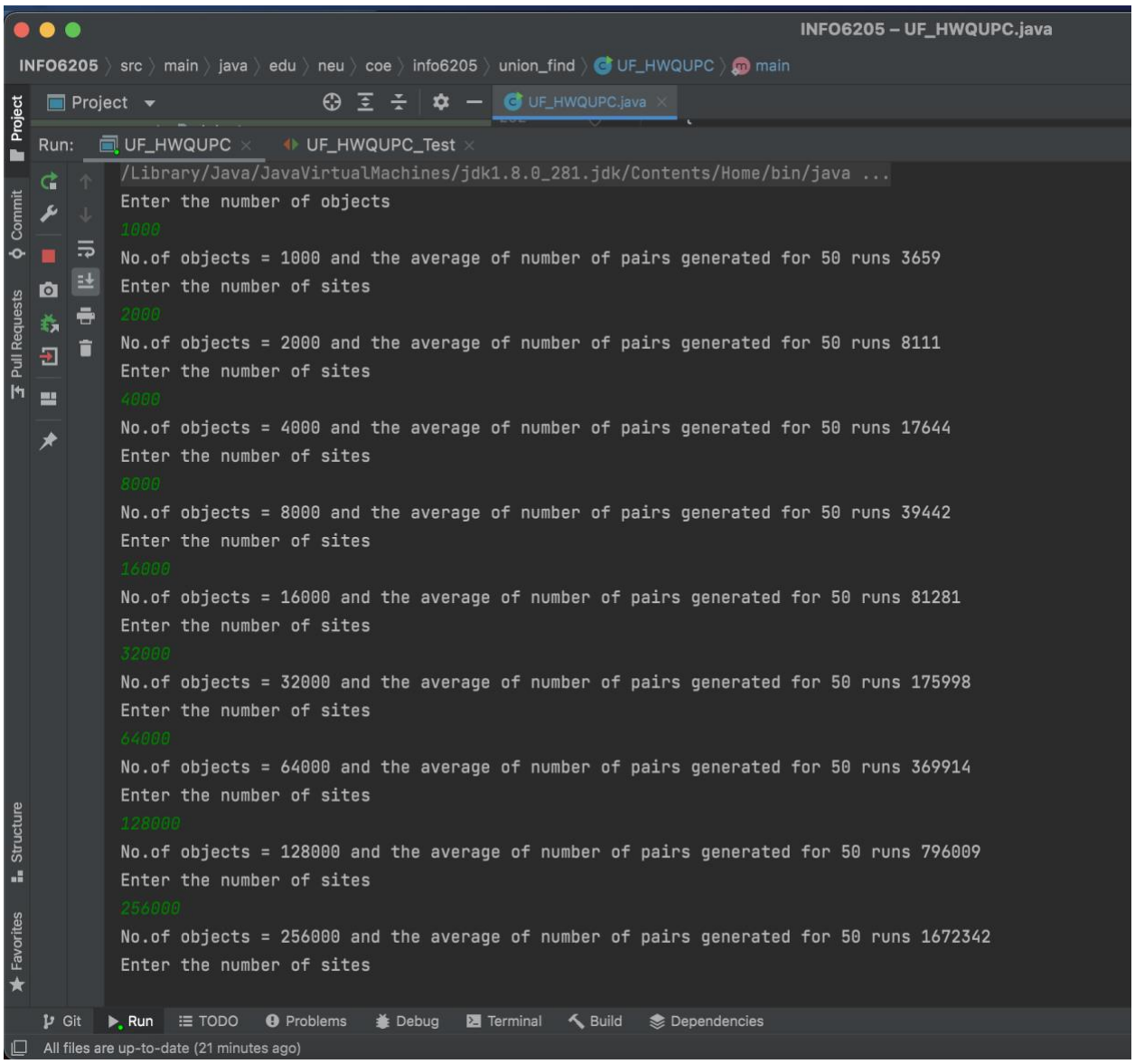
And therefore the relationship can be established as

$m = c* n* \log(n)$ or $m \propto n* \log(n)$

n (Number of Objects)	m (number of generated pairs) over an average of 50 runs	n log n	m / n log n
1000	3659	3000	1.21966667
2000	8111	6602.05999	1.22855594
4000	17644	14408.24	1.22457705
8000	39442	31224.7199	1.26316585
16000	81281	67265.9197	1.20835336
32000	175998	144164.799	1.22081119
64000	369914	307595.518	1.2025988
128000	796009	653722.876	1.21765511
256000	1672342	1384509.43	1.20789499

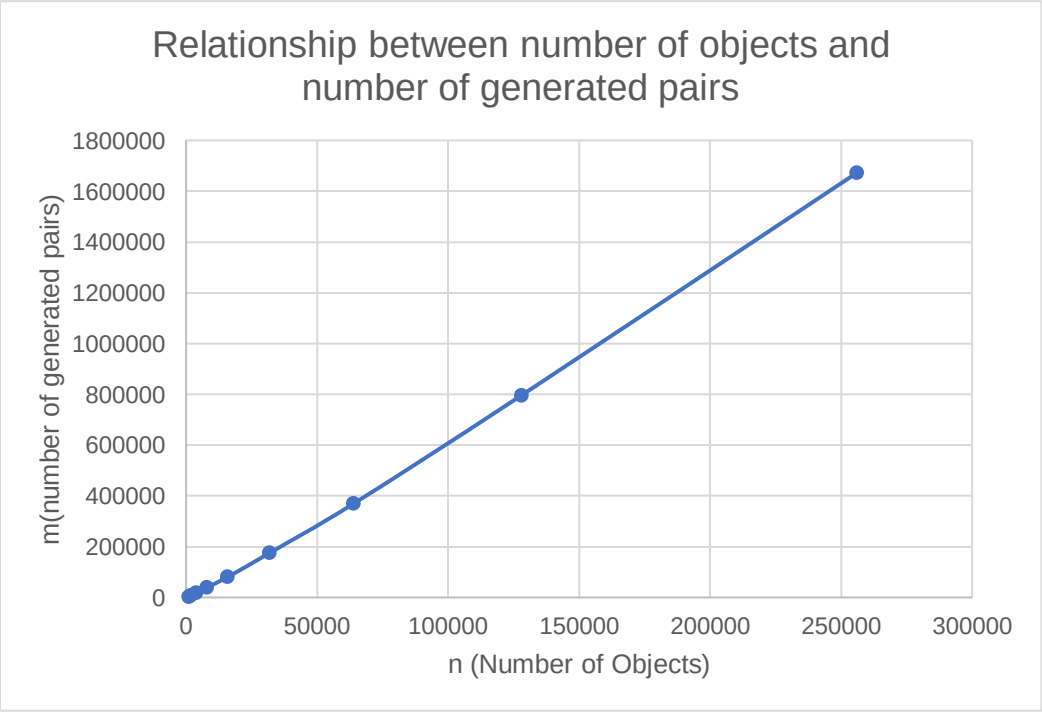


1. **Output :**


INFO6205 – UF_HWQUPC.java
INFO6205 〉 src 〉 main 〉 java 〉 edu 〉 neu 〉 coe 〉 info6205 〉 union_find 〉 UF_HWQUPC 〉 main
Project
UF_HWQUPC.java
Run: UF_HWQUPC
UF_HWQUPC_Test
/Library/Java/JavaVirtualMachines/jdk1.8.0_281.jdk/Contents/Home/bin/java ...
Enter the number of objects
1000
No.of objects = 1000 and the average of number of pairs generated for 50 runs 3659
Enter the number of sites
2000
No.of objects = 2000 and the average of number of pairs generated for 50 runs 8111
Enter the number of sites
4000
No.of objects = 4000 and the average of number of pairs generated for 50 runs 17644
Enter the number of sites
8000
No.of objects = 8000 and the average of number of pairs generated for 50 runs 39442
Enter the number of sites
16000
No.of objects = 16000 and the average of number of pairs generated for 50 runs 81281
Enter the number of sites
32000
No.of objects = 32000 and the average of number of pairs generated for 50 runs 175998
Enter the number of sites
64000
No.of objects = 64000 and the average of number of pairs generated for 50 runs 369914
Enter the number of sites
128000
No.of objects = 128000 and the average of number of pairs generated for 50 runs 796009
Enter the number of sites
256000
No.of objects = 256000 and the average of number of pairs generated for 50 runs 1672342
Enter the number of sites
Git
Run
TODO
Problems
Debug
Terminal
Build
Dependencies
All files are up-to-date (21 minutes ago)

2. **Graphical Representation:**

n (Number of Objects)	m (number of generated pairs) over an average of 50 runs	n log n	m / n log n
1000	3659	3000	1.21966667
2000	8111	6602.05999	1.22855594
4000	17644	14408.24	1.22457705
8000	39442	31224.7199	1.26316585
16000	81281	67265.9197	1.20835336
32000	175998	144164.799	1.22081119
64000	369914	307595.518	1.2025988
128000	796009	653722.876	1.21765511
256000	1672342	1384509.43	1.20789499



- **Unit tests result:**