Sutiksh Verma

002122052

INFO 6205

Program Structures & Algorithms

Fall 2021

Assignment No. 3

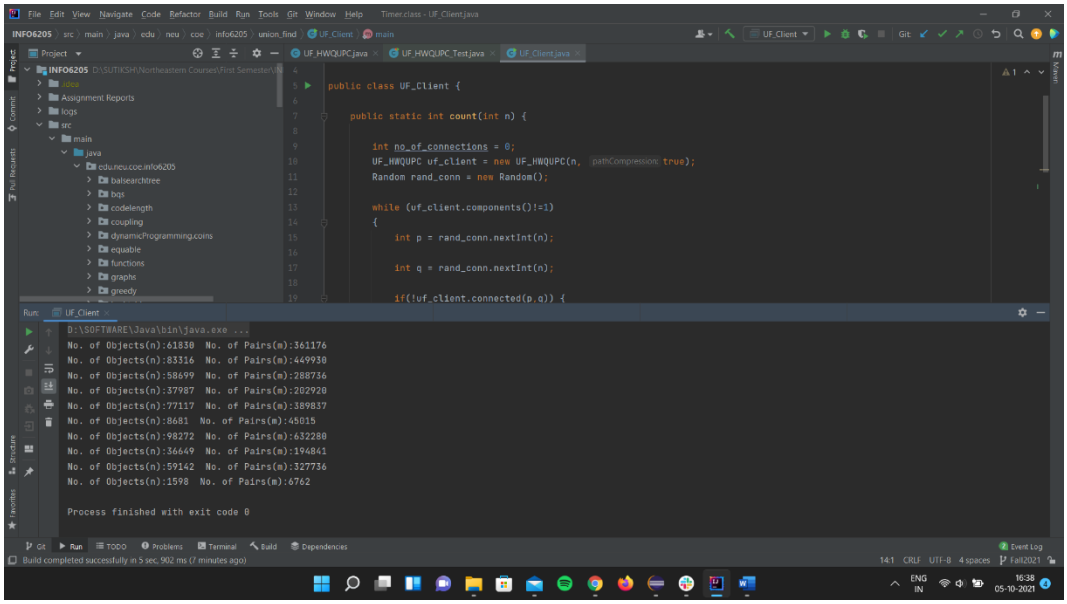
Task: To implement height-weighted Quick union with Path compression.

For this task UF_HWQUPC java class was used and following methods were implemented

- find () – method to update the root of input object if pathCompression is performed
- mergeComponents() – method to merge 2 subtrees such that smaller root points to larger root
- doPathCompression() – method that implements the single-pass process of pathCompression method.

Also, UF_Client java class was created to perform and test the implementation of UF_HWQUPC class

Output:



Console Output:

No. of Objects(n):61830 No. of Pairs(m):361176

No. of Objects(n):83316 No. of Pairs(m):449930

No. of Objects(n):58699 No. of Pairs(m):288736

No. of Objects(n):37987 No. of Pairs(m):202920

No. of Objects(n):77117 No. of Pairs(m):389837

No. of Objects(n):8681 No. of Pairs(m):45015

No. of Objects(n):98272 No. of Pairs(m):632280

No. of Objects(n):36649 No. of Pairs(m):194841

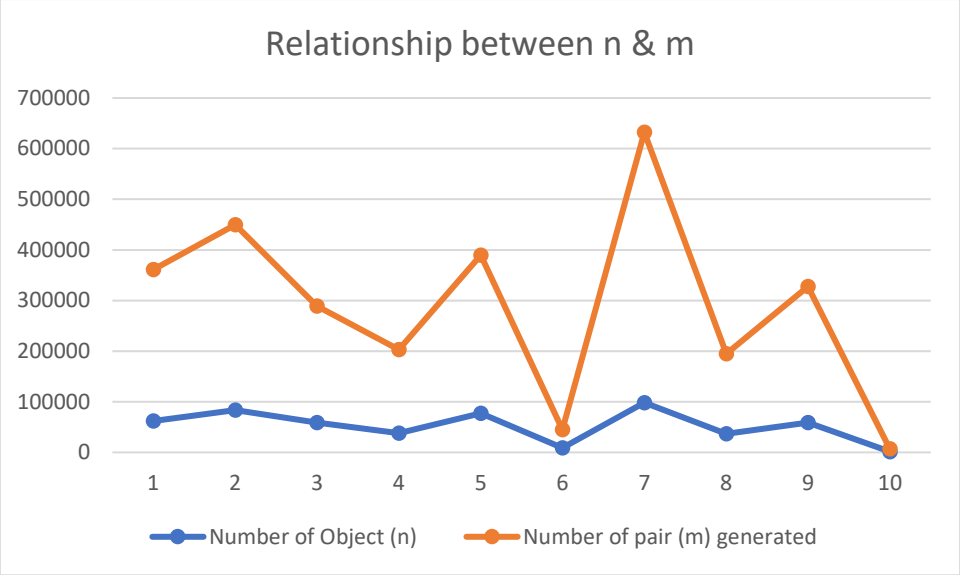
No. of Objects(n):59142 No. of Pairs(m):327736

No. of Objects(n):1598 No. of Pairs(m):6762

Relationship: It can be concluded from the results mentioned above that the number of pairs(m) generated are proportional to the number of objects provided as input. i.e $m \sim 5 * n$

Evidence: I have attached a table and a chart to show the relationship between the number of object (n) and number of pair (m) generated with different set of values for both n and m. As a result, we can see the proportionate result between n and m.

Number of Object (n)	Number of pair (m) generated
61830	361176
83316	449930
58699	288736
37987	202920
77117	389837
8681	45015
98272	632280
36649	194841
59142	327736
1598	6762



The left side of the chart is the value of object (n)

Screenshots of Passed Unit tests: I have attached the screenshot of successfully passed unit test for the class UF_HWQUPC test class.

UF_HWQUPC_Test.java