**Assignment3**

**FengyiZhang SEC02 001564247**

Program file path: [INFO6205-Fall2021 3/src/main/java/edu/neu/coe/info6205/union\_find/UF\_HWQUPC.java](https://github.com/Destination4298/FengyiZhang/commit/03ce00908d867240452b4bd69070bb55e8b5834b" \l "diff-4598fc6c03297b04e1e6b84d43e8159f9c65caabc61fec4cb247d2da995a610a" \o "INFO6205-Fall2021 3/src/main/java/edu/neu/coe/info6205/union_find/UF_HWQUPC.java)

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)

Screen shot for step1:

图形用户界面, 文本, 应用程序

描述已自动生成

Step2:

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).

Screen shots for Step2 :

I’ve run each n for 50 times to get the average connections of each n value.

图形用户界面, 文本, 应用程序

描述已自动生成

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.

Screen shots on Step3:

图表

描述已自动生成

After the calculation, the average ratio of n to n\* is 0.37881464.

So the conclusion is **n = 0.37881464 \* n \***