Program Structures & Algorithms

Spring 2022 Assignment No. 3

Name: ANKUR THAKUR (NUID): 002927572

# Task

* Step 1:
  + 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.
  + 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.

**Program and Output:**

Task 1a:  
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.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Task 1b:  
A screenshot of a computer

Description automatically generated with medium confidence

Task 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).  
  
  
  
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Step 3:

# Relationship and Conclusion

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.  
  
Based on my observations as noted below, I can conclude that the relationship between the M & N is linear.

It can be defined as m = n – 1, Where m is total connections and n is number of inputs given by user.

**Supporting Evidence:**

A screenshot of a computer

Description automatically generated

Thanking you,

Ankur Thakur