CS 2420 Program 6 – 20 points
Spring 2018

Social Network – Union Find

You have a social network containing N members. Use N=1000 for the final version. Friendships form randomly, one each day. It is possible for a friendship to be generated for two people that were already connected (as friendships are random). Implement an algorithm to determine when all members are connected (i.e, every member is a friend of a friend of a friend...of a friend). Output the number of days required. Note we are interested in all unions (not just the successful ones).

Be sure to implement a Union-Find class which is reusable. Perform union by height and path compression (during finds). In order to verify this is working correctly, print the contents of the data structure periodically to make sure it works properly.

We think that path compression gives almost constant time. So you should see a simple relationship between the number of unions and the recursive count of finds. In order to verify this, output the following:

a. The total number of unions that are performed (each union does two finds).
b. The total number of times find was called. Since find is recursive, the total number of times the routine is called will give us a measure of work.

Hints:

a. A "new" friendship may not change the disjoint sets. That is okay. That friendship still "counts" as the friendship for the day.
b. If two members are already in the same set, do NOT link them again. That could invalidate the data values.
c. It is useful if the union tells you whether or not the items were already in the same set. We say a union is "successful" if it joins two members who were not already connected.
d. To join n different elements (via union), you need (n-1) successful unions.
e. Path compression can change the height of a tree, but you won't know for sure. Code the union as though height was accurate. It will be a rough measure of height, which is still worth doing. Sometimes people call this "union by rank".