Name: Georges Hatem

CS 590 Homework 6: Union-Find Structures Reinforcement Exercises

Due Date: March 6, 2022

<u>Problem 7.5.1:</u>

In Section 7.1, connected components is described as follows:

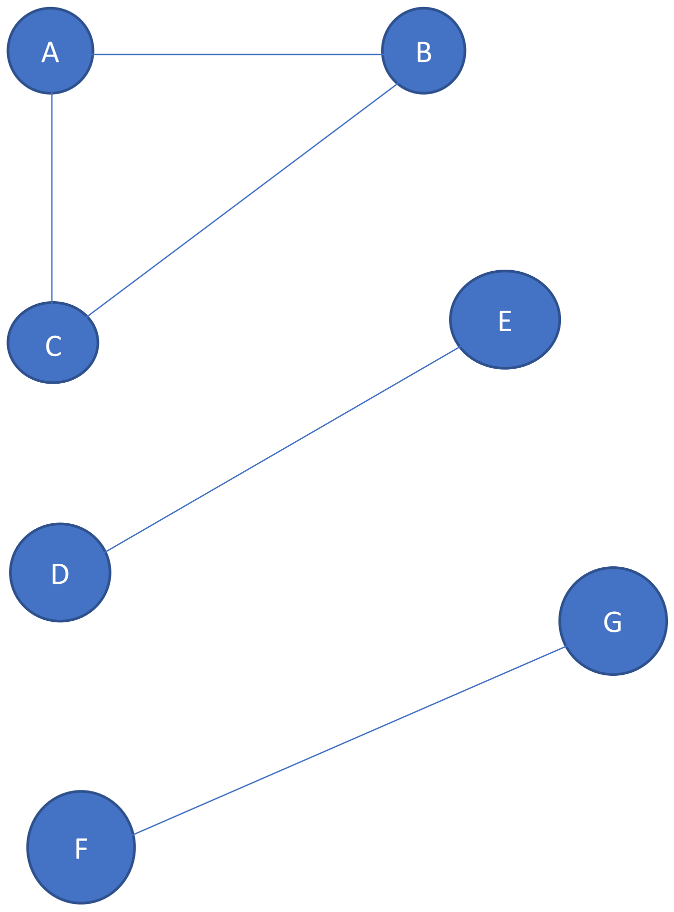
A connected component is a subset, T, from S that satisfies the following properties:

1) Every person in T is related through friendship, that is, for any x and y in T, either x and y are friends or there is a chain of friendship, such as through a friend of a friend of a friend that connects x and y.
2) No one in T is friends with anyone outside of T.

In this exercise, A and B are friends, B and C are friends, and C and A are friends. This means that A, B, and C form a single connected component. D and E are friends with each other only. This means that D and E

form a single connected component. F and G are friends with each others only. This means that F and G form a single connected component.

Let's draw the connected component graph below to further explain my analysis:



A and B are friends so a line has been drawn between them. B and C are friends so a line has been drawn between them. And C and A are friends so a line has been drawn between them. Therefore, A, B, and C are all connected together and form a single connected component. D and E are only friends with each other so a line has been drawn between them. This means that D and E are connected together and form a single connected component. F and G are only friends with each others so a line has been drawn between them. This means that F and G are connected together and form a single connected component.

So, the connected components are as follows:

{(A, B, C), (D, E), (F, G)}

Problem 7.5.2:

Algorithm 7.2.1 is as follows:

```
Algorithm UFConnectedComponents(S,E);
    Input: A set, S, of n people and a set, E, of m
    pairs of people from S defining pairwise
    relationships
    Output: An identification, for each x in S, of
    the connected component containing x

    for each x in S do
        makeSet(x)
    for each (x, y) in E do
        if find(x) != find(y) then
            union(find(x), find(y))
    for each x in S do
        Output "Person x belongs to connected
        component" find(x)
```

From the Algorithm above, n is the number of people in Set, S. makeSet(x) will run through every number n in Set S and make a set from it. This means that makeSet(x) run n times or O(n).

From the Algorithm above, we can see that the number of pairs of people given in Set E are m pairs. The for loop statement (for each (x,y) in E) will run m times because it is running for every pairs in Set E. Considering that no repeated pairs (e.g. (A, B) and then later on (B,A), etc) is included in Set E, the if statement (find(x) != find(y)) will run m times as well. This means that union runs m times as well. To express this in function of m, we can think of it as the following:

This looks like a Tree because we are taking the pairs m (if every pair m has elements they have not seen in any other pair elements, meaning that every element in each pair is distinct from all the other elements in all pairs, then this means that we need to divide n by 2 to get m, which means that we are traversing union similar to O(log(n)). This means that Union runs m times, which is O(log(n)) times in function of n.

As for find, in the if statement (if (find(x) != find(y)), find is getting called 2 times (for both x and y in each loop run) so find is running n times. Now, find in the (for each x in S do), is running c times because the number of connected components is c. So, find is running O(n+c) times. Therefore, find is running O(n) times.

So, according to my analysis above, makeSet(x) is running n times (O(n) times), union is running m times (O(log(n)) times), and find is running n+c times (O(n) times).