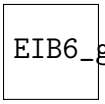
Name:

ID:

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 5 **Spring 2019, CU-Boulder**

Your social science colleagues are interested in quantifying the differences in the news sources of "distant" groups. Their data is from a social network and consists of users and their friends. Part of their research involves quantifying the "maximum social distance" of individuals. To accomplish they need an algorithm that takes in two users as input and returns the maximum number of social groups that connects them. They propose using a friend of a friend (FOAF) approach (`foaf()` below) that starts with one of the input users, finds their FOAFs, then selects the FOAF who has the largest number of friends in common. For example, in the figure below the user (grey) has four friends and four FOAFs. One FOAF (black) has 2 friends in common and that user is selected. This process repeated for each FOAF until the second input user is one of the FOAFs. We can assume that a path between any two users exists.




```
foaf(user_1, user_2, d):
  F = friends(user_1) // set user who are friends with user_1
  if user_2 is in F: return d
  FOAF = []
  for f in F:
    if user_2 is in friends(f): return d
    FOAF.append(friends(f) - F) // set subtraction
  max_foaf_count = 0
  max_foaf = NULL
  for f in FOAF:
    foaf_count = | intersect(F,friends(f)) |
    if max_foaf_count < foaf_count:
      max_foaf_count = foaf_count
      max_foaf = f
  return foaf(max_foaf, user_2, d+1)
```

Name:

ID:

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 5 **Spring 2019, CU-Boulder**

They seem surprised that, while the algorithm is very fast and give reasonable results in most cases, every once in a while the algorithm returns a distance that is different than they expected. Help them understand what assumptions required for the algorithm they developed and why those are not met here.

Name:

ID:

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 5 **Spring 2019, CU-Boulder**

The algorithm will generally work. However, there are a few major assumptions that are being made that, if not met can result in distances different than from what my colleagues thought. If the conditions are not met where user2 is a friend or a FOAF of the current person (hence returning the distance), then the algorithm looks for a FOAF who has the most friends. If we use this approach, it is possible that we will begin traversing down a path of friends that can not end up at user2. Hence, we must make the assumption that any path we take eventually has the ability to reconnect with user2's friend tree. Otherwise, the algorithm would continue to run recursively because user2 will have never of met any of the conditions to return the current distance value. Every time the recursion is called, d is increased by one. If there is no end path from user1 to user2 then d will be equal to infinite and the program will produce a run time error. Additionally, the algorithm is assuming that each jump to a friend is only worth an additional +1 for distance (because the recursion only allows for d to increment by 1 each time). However, each hop should be increase the value of d by the max_foaf_count because this now accounts for the strength of the max_foaf that is being passed into the next recursion. So, the return statement at the end of the algorithm should look like: return foaf(max_foaf, $user_2$, $d + max_foaf_count$). Additionally, the second if statement, which checks if user2 is in friends(f), should be located in the next for loop. We want to see if user2 is in friends(f) but only when f is in FOAF of user1. Therefore in the current algorithm, we must assume that if the condition is true and user2 is in friends(f) then f must also be in FOAF of the previous person.