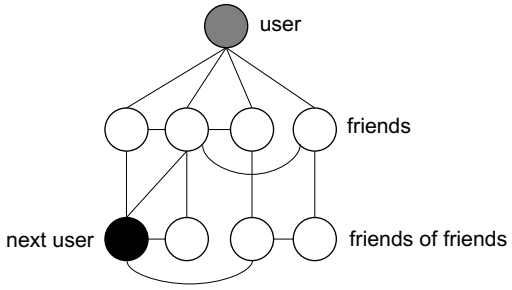
Name: Trevor Buck

ID: 109081318

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: Trevor Buck

ID: 109081318

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.

Thank you for sharing your algorithm with me. After reading through your goals, and your code, I have a few suggestions as to how you can make your algorithm more accurate. I noticed that once your algorithm starts, it never keeps track of which users you've already visited. For instance, you could visit the same user twice and you would never know. I think that it would prove helpful to keep a list of users that have already been visited. Then make a quick check that before you set your 'max_foaf', that they aren't already in that list.

Secondly, and here's where I think the main problem is, you ever only recurse on one friend of a friend. You are assuming that whoever has the most foaf's will lead you to find the optimal path. You could go on an unnecessary loop, and not realize it. Let me put this into perspective. Let's pretend that you are look for a path from A to Y. Now let's say that A matches with both B and Z, however if B has more foaf's, your path would start with A – B. Then that continues and you end up with a path of A - B – C –D –E –F -…. - W – X – Y and this gives you a length of 25. The ideal path would be A – Z – Y with a length of 3, but your algorithm doesn't explore that option because B had more foaf's than Z and that forces your algorithm to take the B path.

To solve this problem, you need an algorithm that explores more options than just one. You need to recurse on every friend that you find, and not just the one with the most foafs. Now this will slow down your algorithm, but it will ensure that you get the right answer, because it explores every option. I think it is intuitive that there are multiple possible paths to get where you want to go, and right now, your algorithm is only taking one.

Now if you want, I also have a few ideas to make work as fast as possible. Something called memoization would really come in handy. However, in this email, I want to focus on the concerns that your current algorithm presents.

Name:

ID:

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