# COSC 222 – Lab 9 – Social Network: Suggest-a-friend

You are given a graph data structure in Graph.java which maintains an adjacency list of a graph. It supports the ability to add String-type nodes, add edges (by specifying a pair of nodes), test if a node is in the graph, retrieve the list of contacts (friends) of a particular String, and it has a toString() method so you can view the adjacency list of the graph. It does not support deletions, and it requires nodes to be Strings, but this could easily be changed to generic types if you want to use this class for other future uses. You do not have to create any graph functionality here.

You are also given 4 graph files which are simply text files of pairs of names, each pair representing a friendship connection. You are also given a method that reads a graph file and creates the graph object. The sizes of these graphs is in the order graph1, graph2, graph3, graph4. The first file is small enough that you could display the adjacency list and troubleshoot your code development. Perhaps graph2 is small enough too. However, graph4 has almost 9000 edges.

**Your task: given a Graph g and a String s, suggest who s might know in g**. That is, implement a suggest-a-contact method for a social network. Your method should simply take in a name s and find a person who is not already a friend of s, and has the most number of common friends with s. If there are multiple people who are tied, they should all be returned. Specifically, the suggest() method should return an ArrayList of all suggested friends. The starter code includes a method header for suggest().

A poor solution would be to find every non-friend of s, and then count the number of common friends. This would not scale well when the social network has millions of nodes. The expected solution would find all friends of s, then the friends of the friends, and count from there (this is a breadth-first search which stops after two distance levels).

Several cases are given to you to ensure your code is working properly. For example, in graph1, suggest(g1, “Logan”) should return [“Chloe”]. In graph3, suggest(g3, “Kendra”) should return two names [“Olive”, “Weston”]. Note that this relationship is not commutative: in graph2, suggest(g2, “Thomas”) returns Dylan, but suggest(g2, “Dylan”) returns Paisley.

All your edits should be made in SuggestAFriend.java .

Submit the one file SuggestAFriend.java (or submit any and all edited file you edited or made).

+1 point per small test file = +3 marks total

+2 points for passing the large test file = +2 marks total

+3 points for proper naming, commenting and documenting, code style, etc = +3 marks total

+2 points for proper use of data structures (Lists and/or Sets and/or Maps) = +2 marks total