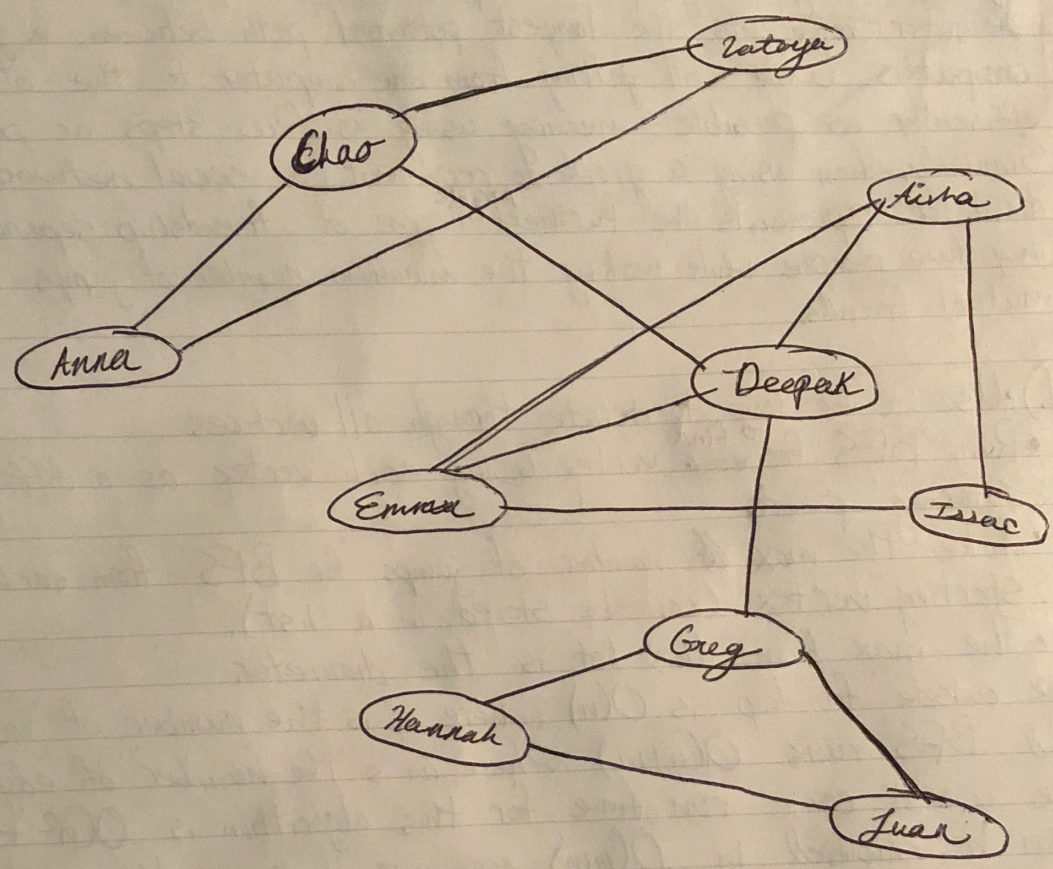
Daniel Licht and Marc Ojalvo
CMPS 1500
Thursday 3:30-4:45
Lab 10 Part 2
25 April 2018

a.)



Vertices = 10
Edges = 13
Connected Components = 1
Largest degree of friendship seperation = 4

b.) G = {'Chao': ['Latoya', 'Anna', 'Deepak'], 'Latoya': ['Anna', 'Chao'], 'Anna': ['Latoya', 'Chao'], 'Deepak': ['Chao', 'Aisha', 'Emma', 'Greg'], 'Aisha': ['Emma', 'Deepak', 'Issac'], 'Emma': ['Aisha', 'Deepak', 'Issac'], 'Issac': ['Emma', 'Aisha'], 'Greg': ['Deepak', 'Hannah', 'Juan'], 'Hannah': ['Greg', 'Juan'], 'Juan': ['Greg', 'Hannah']}

c.) When using a graph to represent a computer network, graph diameter represents the longest potential path between a given two computers while still getting from one computer to the other as efficiently as possible - meaning using as few steps as possible. Similarly, when using a graph to represent a social network, graph diameter represents the furthest degree of friendship separation between any two people while making the minimum number of jumps between mutual friends.

d.) • Use a for loop to iterate through all vertices
• Run BFS ~~for each~~ to find vertex using each vertex as a different starting point.
• Take the max number of jumps for BFS from each different starting vertex (can be stored in a list).
• The max from this list is the diameter.

The outside for loop is $O(n)$ where n is the number of vertices and BFS runs $O(n+m)$ where m is the number of edges so the worst case run time for this algorithm is $O(n^2+nm)$, which can be simplified to $O(nm)$, assuming ~~it is a~~ the graph is connected and not a tree.