# CSci 433/502 Algorithms
## Programming Assignment #3: Social Graphs
## Spring 2016

**Due:** Monday, March 28 at midnight

For this program, you will read in two different types of social graphs and compute some statistics using them.

### Twitter followers

If the graph to be processed is a directed graph, then it represents a small part of a Twitter graph. An edge from node u to node v means that user *u* "follows" user *v* or alternatively, that user v is "followed by" user *u*. For Twitter graphs, your goal is to determine which users are popular. A person (user) *P* is considered popular if *P*'s popularity ratio (computed as the number of users who follow *P* divided by the number of users *P* follows) is at least 2. There is a special case. If user P does not follow any other users, the popularity ratio would be undefined. So, in that case, *P* is considered popular if he or she is followed by at least 3 people.

### Facebook friends

If the graph to be processed is an undirected graph, then it represents a portion of a Facebook friendship graph. Facebook users agree to be mutual friends, so an undirected graph is appropriate. In this case though, the graph is weighted. The weight on an edge ($u$, $v$) represents the number of days $u$ and $v$ have been friends (on Facebook). For this type of graph, you will need to compute several statistics for each user/node $P$:

- The number of friends of $P$
- The number of friends of friends of $P$
- The longest friendship $P$ has with a friend on Facebook

For the friend of friend (FoF) computation, don't count any FoF more than once. Do NOT count $P$ as a friend or FoF of himself.

### Overall Process

Once you read in and instantiate a graph, you must do graph traversals to verify that the graph is constructed correctly. The Depth-First Search (DFS) algorithm is on page 123-4 of the textbook. The BFS algorithm is in the textbook on page 126. Follow these algorithms as carefully as possible. Here is the overall process:

```
Prompt the user for the filename
Open the file and read header
If Directed (and unweighted)     // Twitter
     Instantiate directed graph (edge u --> v means u "follows" v)
     Traverse the graph with DFS and BFS // to show it read in correctly
     Find and list all people who are "popular",
         where popular means they have at least twice the number of
         followers as the number of people they follow.
     List #followers, #followed, ratio for each person
         (Special case: A person who follows no one must have at least 3
         followers to be considered popular)
Else Undirected (and weighted)   // Facebook
     An edge between u and v means u and v are friends, and the weight
         represents the number of days they have been friends.
     Instantiate the friendship graph
     Traverse the graph with DFS and BFS  // to show it read in correctly
     For each person p,
          List the number of friends of p and the number of
               Friends of Friends (FoF) of p
          List p's longest friendship (using the weights)
Close file
```

## File format

The data file will be formatted as discussed in class. Assume vertices are numbered 0..$n$-1. In this case, we will assume each file contains **exactly one graph**. Every graph has a two line "header".

- Line 1: $isDirected$ $isWeighted$
- Line 2: $n$ $m$

On line 1, if *isDirected*==0 the graph is undirected else it is directed. If $isWeighted$==0 the graph is unweighted else it is weighted.

On line 2, $n$ is the number of vertices (nodes) and $m$ is the number of edges.

The next $m$ lines contain information about the edges. If the graph $isWeighted$, the next $m$ lines each contain three integers, $u$, $v$ and $w$, where $u$ and $v$ are the endpoints of the edge and $w$ is the weight on that edge. If the graph is not weighted, each of the $m$ lines contains only $u$ and $v$. If the graph is undirected, there is an edge ($u$, $v$) and an edge ($v$, $u$). If the graph *isDirected*, the edge is from $u$ to $v$.

**Sample output** Note that the traversals were generated from an AL representation based on the order the edges were listed in the input file, not "ordered" as we generally do in class. If you use an AM representation, you will get a different order.

```
Enter the name of a social media graph file (EXIT when finished):
tgraph.txt
Using file: tgraph.txt
Graph is directed, unweighted and has 8 nodes and 19 edges.

DFS traversal: 0 1 3 5 7 4 6 2
BFS traversal: 0 1 7 6 2 3 5 4
Person 0 is NOT popular.  Popularity score:   0.3.  Followed by: 1  Follows: 4
Person 1 is NOT popular.  Popularity score:   0.5.  Followed by: 2  Follows: 4
Person 2 is NOT popular.  Popularity score:   0.2.  Followed by: 1  Follows: 5
Person 3 is NOT popular.  Followed by: 2  Follows: 0
Person 4 IS popular.  Followed by: 3  Follows: 0
Person 5 IS popular.  Popularity score:   3.0.  Followed by: 3  Follows: 1
Person 6 is NOT popular.  Popularity score:   1.0.  Followed by: 3  Follows: 3
Person 7 IS popular.  Popularity score:   2.0.  Followed by: 4  Follows: 2

Enter the name of a social media graph file (EXIT when finished):
fgraph.txt
Using file: fgraph.txt
Graph is undirected, weighted and has 13 nodes and 16 edges.

DFS traversal: 0 1 2 3 8 9 10 5 12 4 6 7 11
BFS traversal: 0 1 2 3 8 9 4 6 11 10 7 5 12
Person 0 has 4 friends and 7 FoFs, oldest friend is 3 (630 days)
Person 1 has 3 friends and 4 FoFs, oldest friend is 2 (220 days)
Person 2 has 3 friends and 4 FoFs, oldest friend is 1 (220 days)
Person 3 has 3 friends and 4 FoFs, oldest friend is 0 (630 days)
Person 4 has 2 friends and 6 FoFs, oldest friend is 9 (1024 days)
Person 5 has 1 friends and 2 FoFs, oldest friend is 10 (550 days)
Person 6 has 2 friends and 4 FoFs, oldest friend is 7 (573 days)
Person 7 has 1 friends and 1 FoFs, oldest friend is 6 (573 days)
Person 8 has 5 friends and 7 FoFs, oldest friend is 4 (960 days)
Person 9 has 3 friends and 7 FoFs, oldest friend is 4 (1024 days)
Person 10 has 3 friends and 2 FoFs, oldest friend is 5 (550 days)
Person 11 has 1 friends and 4 FoFs, oldest friend is 8 (313 days)
Person 12 has 1 friends and 2 FoFs, oldest friend is 10 (121 days)
```

**CSCI 502 Students**
Graduate students must implement one graph type (Twitter or Facebook) using the Adjacency Matrix representation and the other graph type with Adjacency List representation.

**Notes and Suggestions**

- Start early and ask me, the tutor or the TA for help if you need it!
- You must use the data file format as defined above. You should adapt to it, and not expect the grader to use your "special" version of the file. She will create test files that are formatted the same as the samples provided.
- You can assume the data files are formatted correctly and do not have to check for syntax for formatting errors. In this case, all files are guaranteed to be directed and unweighted or undirected and weighted.
- Your traversals may give different results if your process in a different order. The sample output assumes the edges are added to an Adjacency List in the order listed in the file, which may not be in increasing order. If you use an Adjacency Matrix, you will likely get a different order (e.g. for Twitter example, DFS order is 0 1 3 4 5 7 6 2, for AM).
- Undergrads MUST use one (or both) of the two graph representations discussed in class to represent your graphs, either Adjacency Matrix or Adjacency List.