# CIS600 Principles of Social Media and Data Mining
# Spring 2025

## Homework 2 - Network Analysis

**Due: 11:59pm, February 9, 2025**

**Total Points (2.5 points)**

**Introduction**

In this in-class exercise, you will demonstrate some of the basic network analysis on directed network using the Python library NetwokX. An email network is considered a directed network because when you send an email, the connection is directional, meaning it goes from the sender to the recipient, making the relationship between individuals in the network clearly one-way. You are provided with a set of email communication data of employees in a medium size tech start-up company. When you construct the email network, a node represents an employee of company and a directed edge between two nodes represents an email.

**Data file:** email_net.txt

**Please complete the following tasks:**

**Task 1:** import `NetworkX` and `Matplotlib`. You may use the following code:

```
import networkx as nx
import matplotlib.pyplot as plt
```

**Task 2:** Write a function to load up the directed multigraph from email data. Your function should return a directed graph. (Note: Node names should be string)

**Task 3:** Draw the email network.

**Task 4:** Write a function to find the number employees (nodes) and number of emails (edges) are represented in the graph. The function should return a tuple (#employees, #emails).

**Task 5:** Write a function to find whether the graph is strongly connected or weakly connected. Strongly connected graph means that if the email network graph is strongly connected, the email information will go from every employee to every other employee. Weakly connected graph means that if email information go from every employee to every other employee. This function should return a tuple of Booleans as:

```
return (nx.is_strongly_connected(Graph),nx.is_weakly_connected(Graph))
```

**Task 6:** Write a function to find number of nodes in the largest weakly connected component. Function returns an `int`.

**Task 7:** Write a function to find number of nodes in the largest strongly connected component. Function returns an `int`.

**Task 8:** Write a function to return the subgraph of nodes in the largest strongly connected component. You may use `strongly_connected_component_subgraphs()` and name this sub-graph `Graph_sub`.

**Task 9:** Write a function which returns the average distance (average shortest path length) between nodes in `Graph_sub`. This function returns a `float`.

Task 10: Write a function that returns the largest possible distance between two employees in Graph_sub. This function returns a `int`.