*You are currently looking at **version 1.2** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ course resource.*

Assignment 2 - Network Connectivity

In this assignment you will go through the process of importing and analyzing an internal email communication network between employees of a mid-sized manufacturing company. Each node represents an employee and each directed edge between two nodes represents an individual email. The left node represents the sender and the right node represents the recipient.

In [1]:
```
import networkx as nx

# This line must be commented out when submitting to the autograder
#!head email_network.txt
```

Question 1

Using networkx, load up the directed multigraph from `email_network.txt`. Make sure the node names are strings.

This function should return a directed multigraph networkx graph.

In [2]:
```
def answer_one():

    email = nx.read_edgelist('email_network.txt',delimiter='\t', data = [('time',int)], create_using=nx.MultiDiGraph())


    return email

answer_one()
```

Out[2]:
```
<networkx.classes.multidigraph.MultiDiGraph at 0x7fa54417b2e8>
```

Question 2

How many employees and emails are represented in the graph from Question 1?

This function should return a tuple (#employees, #emails).

In [3]:
```
def answer_two():

    email = answer_one()


    return ( len(email.nodes()),len(email.edges()) )
answer_two()
```

Out[3]:
```
(167, 82927)
```

Question 3

- Part 1. Assume that information in this company can only be exchanged through email.

  When an employee sends an email to another employee, a communication channel has been created, allowing the sender to provide information to the receiver, but not vice versa.

  Based on the emails sent in the data, is it possible for information to go from every employee to every other employee?

- Part 2. Now assume that a communication channel established by an email allows information to be exchanged both ways.

  Based on the emails sent in the data, is it possible for information to go from every employee to every other employee?

This function should return a tuple of bools (part1, part2).

In [4]:
```
def answer_three():

    email = answer_one()

    return (nx.is_strongly_connected(email), nx.is_weakly_connected(email))
answer_three()
```

Out[4]:
```
(False, True)
```

Question 4

How many nodes are in the largest (in terms of nodes) weakly connected component?

This function should return an int.

In [5]:
```
def answer_four():

    email = answer_one()
    wccs = nx.weakly_connected_components(email)
    return len(max(wccs, key=len))
answer_four()
```

Out[5]:
```
167
```

Question 5

How many nodes are in the largest (in terms of nodes) strongly connected component?

This function should return an int

In [6]:
```
def answer_five():

    email = answer_one()
    sccs = nx.strongly_connected_components(email)
    return len(max(sccs, key=len))
answer_five()
```

Out[6]: 126

Question 6

Using the NetworkX function strongly_connected_component_subgraphs, find the subgraph of nodes in a largest strongly connected component. Call this graph G_sc.

This function should return a networkx MultiDiGraph named G_sc.

In [7]:
```
def answer_six():

    email = answer_one()
    subgraphs = nx.strongly_connected_component_subgraphs(email)
    G_sc = max(subgraphs,key=len)
    return G_sc
answer_six()
```

Out[7]: <networkx.classes.multidigraph.MultiDiGraph at 0x7fa5412cb710>

Question 7

What is the average distance between nodes in G_sc?

This function should return a float.

In [8]:
```
def answer_seven():

    G_sc = answer_six()

    return nx.average_shortest_path_length(G_sc)
answer_seven()
```

Out[8]: 1.6461587301587302

Question 8

What is the largest possible distance between two employees in G_sc?

This function should return an int.

In [9]:
```
def answer_eight():

    G_sc = answer_six()

    return nx.diameter(G_sc)
answer_eight()
```

Out[9]: 3

Question 9

What is the set of nodes in G_sc with eccentricity equal to the diameter?

This function should return a set of the node(s).

In [10]:
```
def answer_nine():

    G_sc = answer_six()
    return set(nx.periphery(G_sc))
answer_nine()
```

Out[10]: {'129', '134', '97'}

Question 10

What is the set of node(s) in G_sc with eccentricity equal to the radius?

This function should return a set of the node(s).

In [11]:
```
def answer_ten():

    G_sc = answer_six()

    return set(nx.center(G_sc))
answer_ten()
```

Out[11]: {'38'}

Question 11

Which node in G_sc is connected to the most other nodes by a shortest path of length equal to the diameter of G_sc?

How many nodes are connected to this node?

This function should return a tuple (name of node, number of satisfied connected nodes).

In [12]:
```
def answer_eleven():

    G_sc = answer_six()
    diameter = nx.diameter(G_sc)
    peripheries = nx.periphery(G_sc)
    max_count = -1
    result_node = None
    for node in peripheries:
        count = 0
        sp = nx.shortest_path_length(G_sc, node)
        for key, value in sp.items():
            if value == diameter:
                count += 1
            if count > max_count:
                result_node = node
                max_count = count
    return (result_node,max_count)
answer_eleven()
```

Out[12]: ('97', 63)

Question 12

Suppose you want to prevent communication from flowing to the node that you found in the previous question from any node in the center of G_sc, what is the smallest number of nodes you would need to remove from the graph (you're not allowed to remove the node from the previous question or the center nodes)?

This function should return an integer.

In [13]:
```
def answer_twelve():

    G = answer_six()
    center = nx.center(G)[0]
    node = answer_eleven()[0]
    return len(nx.minimum_node_cut(G, center, node))
answer_twelve()
```

Out[13]:
```
5
```

Question 13

Construct an undirected graph G_un using G_sc (you can ignore the attributes).

This function should return a networkx Graph.

In [14]:
```
def answer_thirteen():

    G = answer_six()
    undir_subgraph = G.to_undirected()
    G_un = nx.Graph(undir_subgraph)

    return G_un
answer_thirteen()
```

Out[14]:
```
<networkx.classes.graph.Graph at 0x7fa5440fc668>
```

Question 14

What is the transitivity and average clustering coefficient of graph G_un?

This function should return a tuple (transitivity, avg clustering).

In [15]:
```
def answer_fourteen():

    G = answer_thirteen()

    return nx.transitivity(G), nx.average_clustering(G)
```

In [ ]:
```

```