Assignment3 (Score: 9.0 / 9.0)

1. Test cell (Score: 1.0 / 1.0)
2. Test cell (Score: 1.0 / 1.0)
3. Test cell (Score: 1.0 / 1.0)
4. Test cell (Score: 1.0 / 1.0)
5. Test cell (Score: 1.0 / 1.0)
6. Test cell (Score: 1.0 / 1.0)
7. Test cell (Score: 1.0 / 1.0)
8. Test cell (Score: 1.0 / 1.0)
9. Test cell (Score: 1.0 / 1.0)

Assignment 3

In this assignment you will explore measures of centrality on two networks, a friendship network in Part 1, and a blog netw in Part 2.

Part 1

Answer questions 1-4 using the network `G1`, a network of friendships at a university department. Each node corresponds person, and an edge indicates friendship.

The network has been loaded as networkx graph object `G1`.

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

G1 = nx.read_gml('assets/friendships.gml')
```

Question 1

Find the degree centrality, closeness centrality, and betweeness centrality of node 100.

This function should return a tuple of floats `(degree_centrality, closeness_centrality, betweenness_centrality)`.

In [2]:

```
import networkx as nx

# Function to calculate centralities
def calculate_centralities(G, node):
    # Ensure the node exists in the graph
    if node not in G:
        raise ValueError(f"Node {node} not found in the graph.")

    # Degree Centrality
    degree_centrality = nx.degree_centrality(G)[node]

    # Closeness Centrality
    closeness_centrality = nx.closeness_centrality(G, u=node)

    # Betweenness Centrality
    betweenness_centrality = nx.betweenness_centrality(G, normalized=True)
de]

    return (degree_centrality, closeness_centrality, betweenness_centralit

# Function to return the centralities as a tuple for node 100
def answer_one():
    G1 = nx.read_gml('assets/friendships.gml')
    centralities = calculate_centralities(G1, 100)
    return centralities

# Call the function and store the result in ans_one
ans_one = answer_one()

# Print the result to check the output
print(ans_one)

# Assert the result is a tuple
assert type(ans_one) == tuple, "You must return a tuple"
```

```
(0.0026501766784452294, 0.2654784240150094, 7.142902633244772e-05)
```

In [3]:

```
cell-cd6a99ae1fdd71a9
```

```
ans_one = answer_one()
assert type(ans_one) == tuple, "You must return a tuple"
```

Use centrality measures to answer questions 2-4

Question 2

Suppose you are employed by an online shopping website and are tasked with selecting one user in network G1 to send a online shopping voucher to. We expect that the user who receives the voucher will send it to their friends in the network. Y want the voucher to reach as many nodes as possible. The voucher can be forwarded to multiple users at the same time, b the travel distance of the voucher is limited to one step, which means if the voucher travels more than one step in this netv it is no longer valid. Apply your knowledge in network centrality to select the best candidate for the voucher.

This function should return an integer, the chosen node.

In [4]:

```
import operator

def answer_two():
    # Load the network graph
    G1 = nx.read_gml('assets/friendships.gml')

    # Calculate degree centrality for all nodes
    degree_centrality = nx.degree_centrality(G1)

    # Find the node with the highest degree centrality
    best_node = max(degree_centrality.items(), key=operator.itemgetter(1))

    return best_node

# Call the function and print the result
ans_two = answer_two()
print(ans_two)
```

```
105
```

In [5]:

```
cell-d890b05007b8cce5
```

```
ans_two = answer_two()
```

Question 3

Now the limit of the voucher's travel distance has been removed. Because the network is connected, regardless of who yo pick, every node in the network will eventually receive the voucher. However, we now want to ensure that the voucher reac nodes as quickly as possible (i.e. in the fewest number of hops). How will you change your selection strategy? Write a func to tell us who is the best candidate in the network under this condition.

This function should return an integer, the chosen node.

In [6]:

```
def answer_three():
    # Load the network graph
    G1 = nx.read_gml('assets/friendships.gml')

    # Calculate closeness centrality for all nodes
    closeness_centrality = nx.closeness_centrality(G1)

    # Find the node with the highest closeness centrality
    best_node = max(closeness_centrality.items(), key=operator.itemgetter(
[0]

    return best_node

# Call the function and print the result
ans_three = answer_three()
print(ans_three)
```

```
23
```

In [7]:

```
        cell-467293083292b0ee

ans_three = answer_three()
```

Question 4

Assume the restriction on the voucher's travel distance is still removed, but now a competitor has developed a strategy to remove a person from the network in order to disrupt the distribution of your company's voucher. You competitor plans to remove people who act as bridges in the network. Identify the best possible person to be removed by your competitor?

This function should return an integer, the chosen node.

In [8]:
```
def answer_four():
    # Load the network graph
    G1 = nx.read_gml('assets/friendships.gml')

    # Calculate betweenness centrality for all nodes (normalized)
    betweenness_centrality = nx.betweenness_centrality(G1, normalized=True
ndpoints=False)

    # Find the node with the highest betweenness centrality
    best_node = max(betweenness_centrality.items(), key=operator.itemgette
1))[0]

    return best_node

# Call the function and print the result
ans_four = answer_four()
print(ans_four)
```

```
333
```

In [9]:
```
cell-338a3ca88864385b
```

```
ans_four = answer_four()
```

Part 2

`G2` is a directed network of political blogs, where nodes correspond to a blog and edges correspond to links between blo
Use your knowledge of PageRank and HITS to answer Questions 5-9.

In [10]:
```
G2 = nx.read_gml('assets/blogs.gml')
```

Question 5

Apply the Scaled Page Rank Algorithm to this network. Find the Page Rank of node 'realclearpolitics.com' with damping v
0.85.

This function should return a float.

In [11]:

```
def answer_five():
    # Load the network graph
    G2 = nx.read_gml('assets/blogs.gml')

    # Calculate PageRank with damping factor 0.85
    pagerank = nx.pagerank(G2, alpha=0.85)

    # Return the PageRank of the specified node
    return pagerank['realclearpolitics.com']

# Call the function and print the result
ans_five = answer_five()
print(ans_five)
```

```
0.004636694781649098
```

In [12]:

```
cell-5ade64a5771dcbce

ans_five = answer_five()
```

Question 6

Apply the Scaled Page Rank Algorithm to this network with damping value 0.85. Find the 5 nodes with highest Page Rank.

This function should return a list of the top 5 blogs in desending order of Page Rank.

In [13]:

```
def answer_six():
    # Load the network graph
    G2 = nx.read_gml('assets/blogs.gml')

    # Calculate PageRank with damping factor 0.85
    pagerank = nx.pagerank(G2, alpha=0.85)

    # Sort the blogs by PageRank, in descending order, and get the top 5
    top_5_blogs = sorted(pagerank.items(), key=operator.itemgetter(1), rev
e=True)[:5]

    # Return only the blog names (keys)
    return [blog[0] for blog in top_5_blogs]

# Call the function and print the result
ans_six = answer_six()
print(ans_six)
```

```
['dailykos.com', 'atrios.blogspot.com', 'instapundit.com', 'blogsforbush.co
'talkingpointsmemo.com']
```

In [14]:
```
cell-fa118135cb4998f4
```

```
ans_six = answer_six()
assert type(ans_six) == list, "You must return a list"
```

Question 7

Apply the HITS Algorithm to the network to find the hub and authority scores of node 'realclearpolitics.com'.

Your result should return a tuple of floats `(hub_score, authority_score)`.

In [15]:
```
def answer_seven():
    # Load the network graph
    G2 = nx.read_gml('assets/blogs.gml')

    # Apply the HITS algorithm
    hits_scores = nx.hits(G2)

    # Return the hub and authority scores of 'realclearpolitics.com'
    return (hits_scores[0]['realclearpolitics.com'], hits_scores[1]['realc
rpolitics.com'])

# Call the function and print the result
ans_seven = answer_seven()
print(ans_seven)
```

```
(0.0003243556140278731, 0.003918957644934254)
```

In [16]:
```
cell-43b3de064e549ef6
```

```
ans_seven = answer_seven()
assert type(ans_seven) == tuple, "You must return a tuple"
```

Question 8

Apply the HITS Algorithm to this network to find the 5 nodes with highest hub scores.

This function should return a list of the top 5 blogs in desending order of hub scores.

In [17]:

```
def answer_eight():
    # Load the network graph
    G2 = nx.read_gml('assets/blogs.gml')

    # Apply the HITS algorithm to get hub and authority scores
    hits_scores = nx.hits(G2)

    # Sort the blogs by hub scores in descending order and get the top 5
    top_5_hub_blogs = sorted(hits_scores[0].items(), key=operator.itemgett
1), reverse=True)[:5]

    # Return only the blog names (keys)
    return [blog[0] for blog in top_5_hub_blogs]

# Call the function and print the result
ans_eight = answer_eight()
print(ans_eight)
```

```
['politicalstrategy.org', 'madkane.com/notable.html', 'liberaloasis.com', '
efour.typepad.com/commonprejudice', 'bodyandsoul.typepad.com']
```

In [18]:

```
cell-72499b780b38eb2c
```

```
ans_eight = answer_eight()
assert type(ans_eight) == list, "You must return a list"
```

Question 9

Apply the HITS Algorithm to this network to find the 5 nodes with highest authority scores.

This function should return a list of the top 5 blogs in desending order of authority scores.

In [19]:

```
def answer_nine():
    # Load the network graph
    G2 = nx.read_gml('assets/blogs.gml')

    # Apply the HITS algorithm to get hub and authority scores
    hits_scores = nx.hits(G2)

    # Sort the blogs by authority scores in descending order and get the t
5
    top_5_authority_blogs = sorted(hits_scores[1].items(), key=operator.it
etter(1), reverse=True)[:5]

    # Return only the blog names (keys)
    return [blog[0] for blog in top_5_authority_blogs]

# Call the function and print the result
ans_nine = answer_nine()
print(ans_nine)
```

```
['dailykos.com', 'talkingpointsmemo.com', 'atrios.blogspot.com', 'washingto
thly.com', 'talkleft.com']
```

In [20]:

```
cell-bbc73cedc13c80ca
```

```
ans_nine = answer_nine()
assert type(ans_nine) == list, "You must return a list"
```

In [ ]: