Mod 9 ICA. Community Detection using NetworkX

In this activity you will investigate the Karate Club Data and identify communities.

See notes here:

https://networkx.org/documentation/stable/auto_examples/graph/plot_karate_club.html

See Details about the Data and its collection here:
http://www1.ind.ku.dk/complexLearning/zachary1977.pdf

"BACKGROUND These are data collected from the members of a university karate club by Wayne Zachary. The edges represents the presence or absence of ties among the members of the club..."

Review the code below that investigates this data set and builds a graphical model. Answer the questions below which focus on community detection.

The Zachary Karate Club graph is a well-known social network dataset that represents the relationships between members of a karate club, as observed by sociologist Wayne Zachary in the 1970s. The dataset is often used in network analysis, particularly to study community structure and the dynamics of group division.

Here are the key aspects of the social network represented by the graph:

1. Nodes (Members):

The graph contains 34 nodes, each representing a member of the karate club. Each node is identified by a number (0 to 33). These are individuals who were part of the club at the time of the study. 2. Edges (Interactions): The edges represent social interactions between the members. There are 78 edges in total. Each edge indicates a relationship or communication between two members of the club. The edges were originally recorded based on the frequency and strength of interactions. 3. Club Split: The key feature of this network is that, due to a dispute between the club's instructor (referred to as "Mr. Hi") and the club's administrator (referred to as "Officer"), the club split into two factions. One faction, led by "Mr. Hi," was composed of certain members (e.g., node 0, the instructor). The other faction, led by the "Officer," contained those who sided with the officer (e.g., node 33). The 'club' attribute for each node indicates which faction they belong to, such as "Mr. Hi" or "Officer". 4. Communities: The social network can be analyzed for its community structure, which was naturally formed by the factional split. The two communities, Mr. Hi's Club and Officer's Club, were formed after the rift, and the structure of the graph shows how tightly-knit the members of each faction were. 5. Graph Properties: Nodes: 34 (members of the club) Edges:

78 (connections between members) Density: The graph is relatively dense, indicating many interactions. Connected Components: The graph is connected, meaning there is a path between every pair of nodes, though the structure of connections reflects the social dynamics of the split.

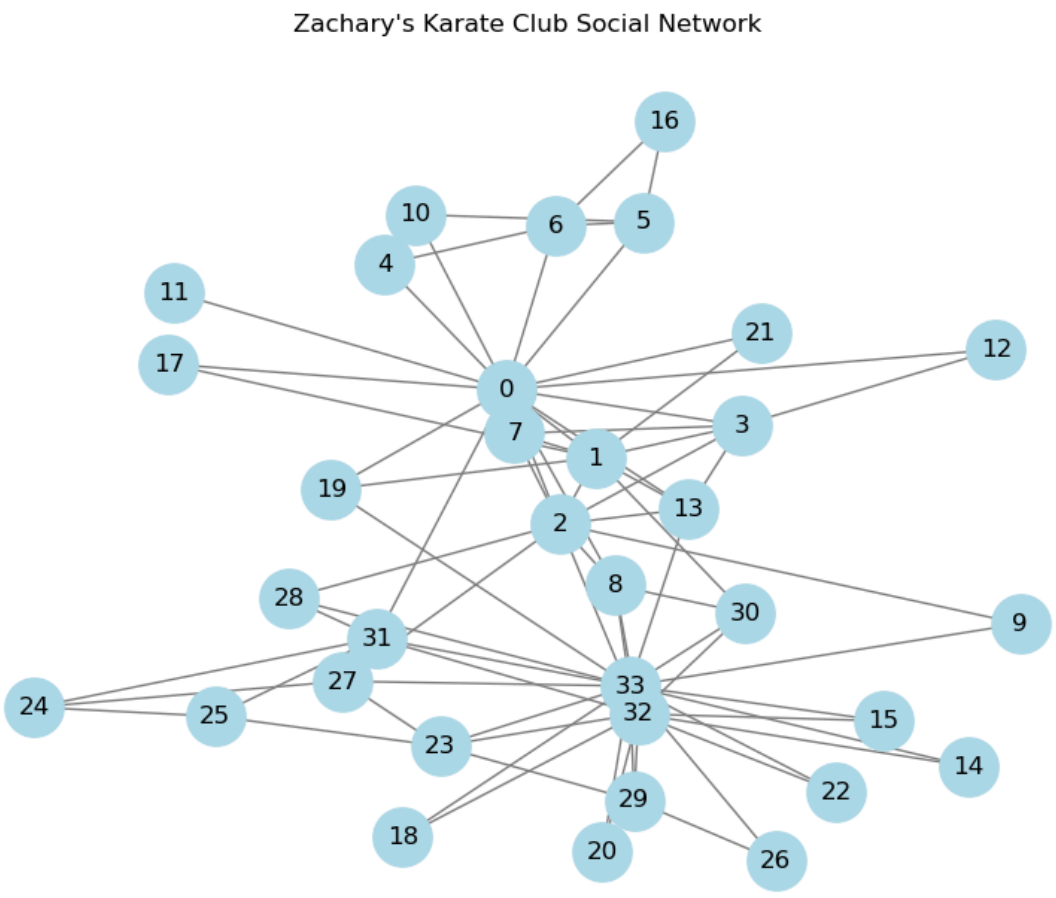
In [3]:
```
# Import Modules
import sys
import networkx as nx
import matplotlib.pyplot as plt
```

In [5]:
```
#Let's import the ZKC graph:
#karate_club_graph(): Returns Zachary’s Karate Club graph.
#Each node in the returned graph has a node attribute ‘club’ that indicates the nam
#Each edge has a weight based on the number of contexts in which that edge’s incide
ZKC_graph = nx.karate_club_graph()

# Prints a summary of the graph object, including the number of nodes and edges.
print(ZKC_graph)

# Retrieves and prints the attributes associated with node 0 in the graph.
# This will typically include information like its "club" membership in the dataset
print(ZKC_graph.nodes[0])

# Retrieves and prints the attributes associated with node 33 in the graph.
# This will typically include information like its "club" membership in the dataset
print(ZKC_graph.nodes[33])


# Prints a list of all the edges in the graph.
# Each edge is represented as a tuple (node1, node2), where node1 and node2 are the
# This shows the relationships or connections between nodes in the karate club grap
print(ZKC_graph.edges())


# Counts and returns the number of edges between node 0 and node 33.
# In this context, it will check if these nodes are directly connected (should be 1
print(ZKC_graph.number_of_edges(0,33))
```

```
Graph named "Zachary's Karate Club" with 34 nodes and 78 edges
{'club': 'Mr. Hi'}
{'club': 'Officer'}
[(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (0, 7), (0, 8), (0, 10), (0, 11),
(0, 12), (0, 13), (0, 17), (0, 19), (0, 21), (0, 31), (1, 2), (1, 3), (1, 7), (1, 1
3), (1, 17), (1, 19), (1, 21), (1, 30), (2, 3), (2, 7), (2, 8), (2, 9), (2, 13), (2,
27), (2, 28), (2, 32), (3, 7), (3, 12), (3, 13), (4, 6), (4, 10), (5, 6), (5, 10),
(5, 16), (6, 16), (8, 30), (8, 32), (8, 33), (9, 33), (13, 33), (14, 32), (14, 33),
(15, 32), (15, 33), (18, 32), (18, 33), (19, 33), (20, 32), (20, 33), (22, 32), (22,
33), (23, 25), (23, 27), (23, 29), (23, 32), (23, 33), (24, 25), (24, 27), (24, 31),
(25, 31), (26, 29), (26, 33), (27, 33), (28, 31), (28, 33), (29, 32), (29, 33), (30,
32), (30, 33), (31, 32), (31, 33), (32, 33)]
0
```

In [7]:
```
# Import the NetworkX library, which is used for creating, manipulating, and studyi
import networkx as nx
```

```
# Import the Matplotlib library, specifically for plotting graphs and visualization
import matplotlib.pyplot as plt

# Load the Karate Club graph from NetworkX. This dataset represents the social inte
# It is a built-in graph in NetworkX, containing 34 nodes (members) and 78 edges (i
ZKC_graph = nx.karate_club_graph()

# Create a new figure for plotting with a specified size of 8 inches by 6 inches.
plt.figure(figsize=(8, 6))

# Draw the graph using the nx.draw() function:
# - `with_labels=True` includes labels for each node, which are the node numbers (0
# - `node_color='lightblue'` sets the color of the nodes to light blue.
# - `edge_color='gray'` sets the color of the edges (connections between nodes) to
# - `node_size=800` sets the size of each node in the visualization.
nx.draw(ZKC_graph, with_labels=True, node_color='lightblue', edge_color='gray', nod

# Adds a title to the plot: "Zachary's Karate Club Social Network" to describe the
plt.title("Zachary's Karate Club Social Network")

# Display the plot with the graph visualization. This will render the graph in a ne
plt.show()
```



In [9]:
```
# Retrieves the 'club' attribute for each node in the Karate Club graph.
# This attribute represents which faction each member belongs to (either 'Mr. Hi' o
# The result is stored in the dictionary , where the keys are node IDs and the valu
```

```
club_labels = nx.get_node_attributes(ZKC_graph, 'club')

# Prints the `club_labels` dictionary, showing the faction (club) affiliation for e
print(club_labels)
```

```
{0: 'Mr. Hi', 1: 'Mr. Hi', 2: 'Mr. Hi', 3: 'Mr. Hi', 4: 'Mr. Hi', 5: 'Mr. Hi', 6: 'M
r. Hi', 7: 'Mr. Hi', 8: 'Mr. Hi', 9: 'Officer', 10: 'Mr. Hi', 11: 'Mr. Hi', 12: 'Mr.
Hi', 13: 'Mr. Hi', 14: 'Officer', 15: 'Officer', 16: 'Mr. Hi', 17: 'Mr. Hi', 18: 'Of
ficer', 19: 'Mr. Hi', 20: 'Officer', 21: 'Mr. Hi', 22: 'Officer', 23: 'Officer', 24:
'Officer', 25: 'Officer', 26: 'Officer', 27: 'Officer', 28: 'Officer', 29: 'Office
r', 30: 'Officer', 31: 'Officer', 32: 'Officer', 33: 'Officer'}
```

Club / Group

Members of the karate group were divided on some policy and split into groups / clubs of support. Members either belonged to Mr Hi's Support Club or Officer's (John A) Support Club.

In [11]:
```
# Converts the Karate Club graph into a NumPy array representation using NetworkX's
# The resulting matrix `A` is an adjacency matrix, where each element A[i, j] repre
# and if the edge exists, the value represents the weight (number of interactions).
#The edge weights in the Karate Club graph are based on the frequency of interactio
#These weights represent the number of interactions or the strength of the relation
A = nx.convert_matrix.to_numpy_array(ZKC_graph)
print(A)
```

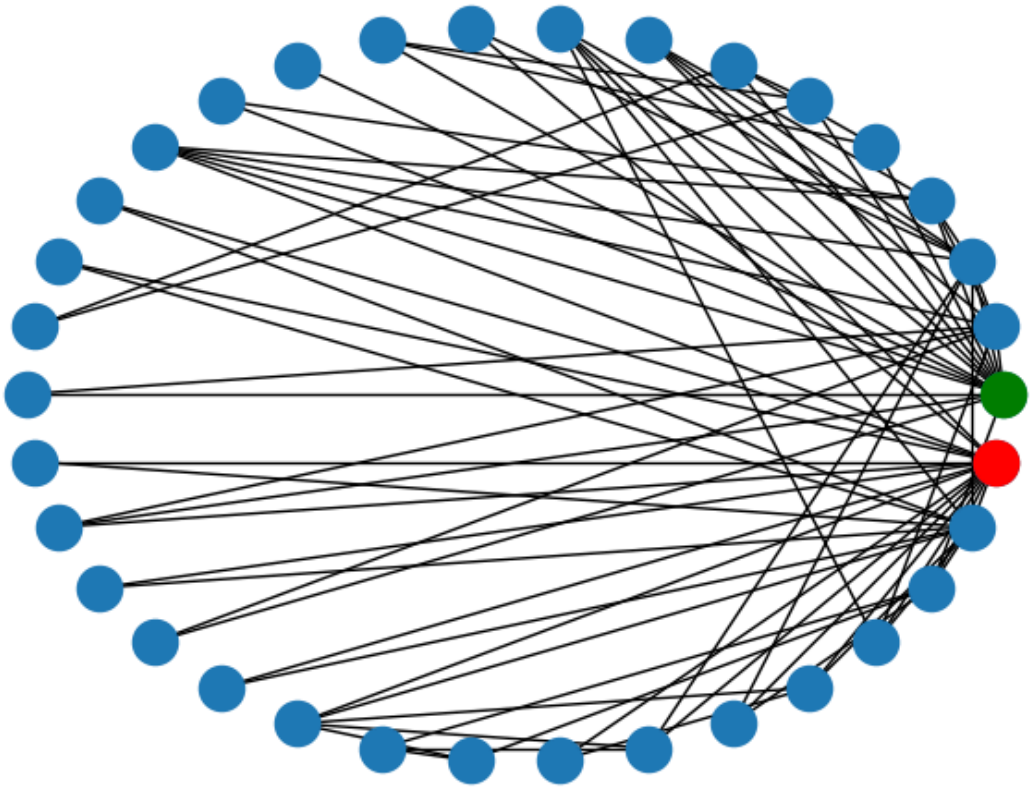
```
[[0. 4. 5. ... 2. 0. 0.]
 [4. 0. 6. ... 0. 0. 0.]
 [5. 6. 0. ... 0. 2. 0.]
 ...
 [2. 0. 0. ... 0. 4. 4.]
 [0. 0. 2. ... 4. 0. 5.]
 [0. 0. 0. ... 4. 5. 0.]]
```

Visualize the Network

In [13]:
```
# Assigns node 0 as Mr. Hi and node 33 as John A, to track which nodes represent th
Mr_Hi = 0
John_A = 33

# Generates positions for all nodes using a circular layout.
# This layout arranges the nodes in a circular pattern, but other layouts are also
circ_pos = nx.circular_layout(ZKC_graph)

# Draws the whole graph using the circular layout generated above.
# The nodes will be placed in a circular arrangement, and the edges will connect th
nx.draw(ZKC_graph, circ_pos)

# Highlights Mr. Hi by drawing node 0 (Mr. Hi) with green color.
# The `alpha=1` argument ensures that the node is fully opaque.
nx.draw_networkx_nodes(ZKC_graph, circ_pos, nodelist=[Mr_Hi], node_color='g', alpha
```

```
# Highlights John A by drawing node 33 (John A) with red color.
# Again, `alpha=1` ensures full opacity for the node.
nx.draw_networkx_nodes(ZKC_graph, circ_pos, nodelist=[John_A], node_color='r', alph
```

Out[13]: <matplotlib.collections.PathCollection at 0x244d831e600>



Connections

There are many graph measures to assess "connectivity" of the nodes such as densities, degree, modularity and more! This can help us to identify communities given this network construction.

1. Graph Density is a measure that helps to understand how interconnected the nodes are in a network. A density of 1 indicates a complete graph (every node is connected to every other node), while a density close to 0 indicates a sparse network.\

Density= $\frac{2|E|}{|V|(|V|-1)}$, where $|E|$ is the number of edges and $|V|$ is the number of vertices
The Karate Club graph has 34 nodes and 78 edges, so, the Density= $\frac{2\times 78}{34\times 33} \approx 0.14$
This means that the graph has about 14% of the edges that a fully connected graph with the same number of nodes would have. Therefore, the Karate Club graph is relatively sparse.
The 14% density indicates that there are many pairs of nodes (club members) that are not

directly connected by an edge. This is typical in social networks, where some individuals form close-knit groups, but not everyone is directly connected to everyone else.

2. Modularity is a measure used in network analysis to quantify the strength of the division of a network into communities or modules. A community is a group of nodes that are more densely connected to each other than to the rest of the network. In NetworkX, modularity is often used to evaluate how well a graph can be partitioned into communities.

3. The degree of a node in a graph is defined as the number of edges incident to that node. It is a measure of how connected the node is within the graph. In simple terms, the degree of a node indicates how many direct neighbors it has.

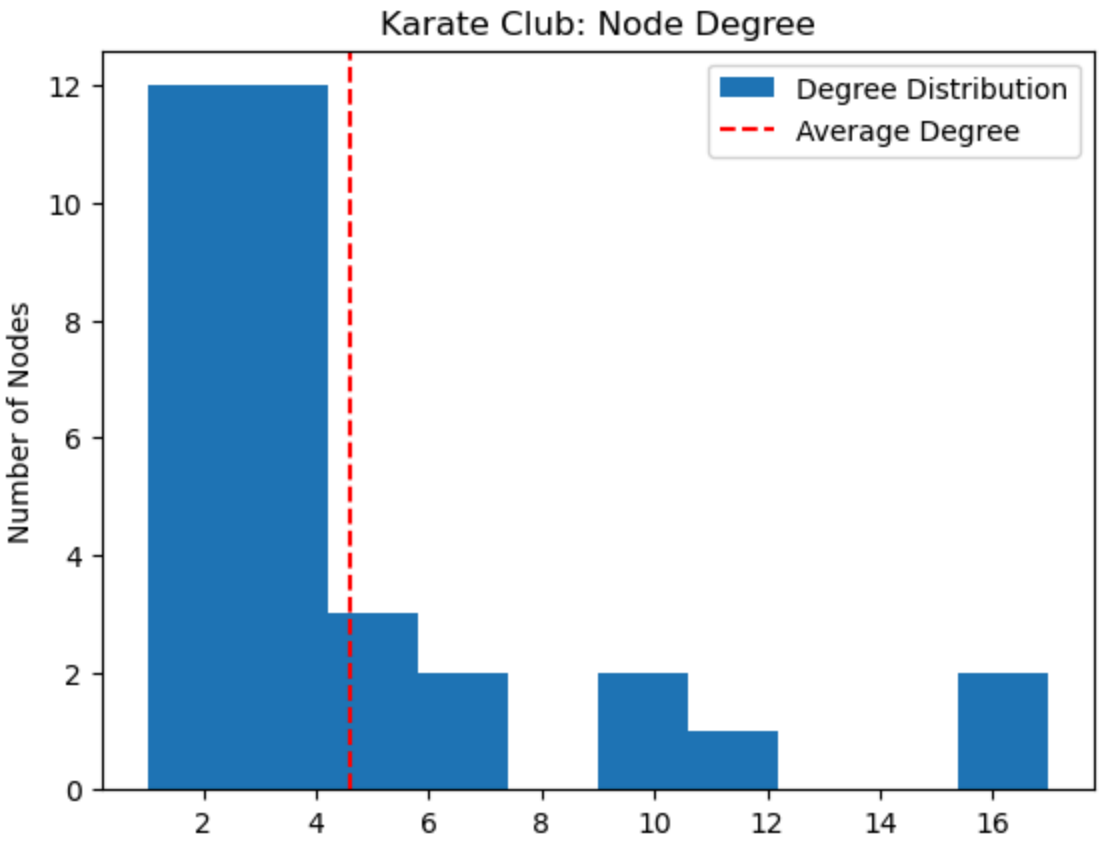
In [15]:
```
# Calculate the density of the Karate Club graph using NetworkX's density function.
# The density is the ratio of the number of edges to the maximum possible number of
density = nx.density(ZKC_graph)

# Print the calculated edge density. The density value is converted to a string and
# The result will display the edge density of the graph.
print('The edge density is: ' + str(density))
```

```
The edge density is: 0.13903743315508021
```

In [17]:
```
#the degree function in networkx returns a DegreeView object capable of iterating t
# Get the degree of each node in the Karate Club graph using the degree method from
# The degree of a node is the number of edges connected to that node.
degree = ZKC_graph.degree()

# print out the degree view of the nodes in the Karate Club graph
print(degree)

# Create an empty list to store the degree values of each node.
degree_list = []

# Loop through each node and its degree in the graph.
# Append the degree (number of edges) of each node to the degree_list.
for (n, d) in degree:
    degree_list.append(d)

# Calculate the average degree by summing all the degrees and dividing by the numbe
av_degree = sum(degree_list) / len(degree_list)

#In the Karate Club graph has an average degree of 4.5, this means that on average,
# Print the average degree of the graph. The degree values are converted to a strin
print('The average degree is ' + str(av_degree))
```

```
[(0, 16), (1, 9), (2, 10), (3, 6), (4, 3), (5, 4), (6, 4), (7, 4), (8, 5), (9, 2),
(10, 3), (11, 1), (12, 2), (13, 5), (14, 2), (15, 2), (16, 2), (17, 2), (18, 2), (1
9, 3), (20, 2), (21, 2), (22, 2), (23, 5), (24, 3), (25, 3), (26, 2), (27, 4), (28,
3), (29, 4), (30, 4), (31, 6), (32, 12), (33, 17)]
The average degree is 4.588235294117647
```

In [19]:
```
# Create a histogram of the degree distribution of the nodes in the Karate Club gra
# `degree_list` contains the degree values of all nodes.
# `bins=10` specifies the number of bins to group the degree values into for the hi
plt.hist(degree_list, label='Degree Distribution', bins=10)

# Add a vertical dashed line to the histogram at the average degree value.
# The line is drawn in red (`color='r'`) and is dashed (`linestyle='dashed'`).
# `label='Average Degree'` adds a label to the line for the legend.
plt.axvline(av_degree, color='r', linestyle='dashed', label='Average Degree')

# Display the legend, which shows labels for the histogram and the average degree l
plt.legend()

# Set the y-axis label as "Number of Nodes", which represents the frequency of node
plt.ylabel('Number of Nodes')

# Set the title of the plot to describe it as the degree distribution of the Karate
plt.title('Karate Club: Node Degree')

# Display the plot.
plt.show()
```



The clustering coefficient is a measure used in graph theory to describe how close a node's neighbors are to forming a clique (a complete subgraph). It is used to quantify the degree to which nodes in a graph tend to cluster together. The clustering coefficient can be computed in different ways, depending on whether you're looking at individual nodes or the whole graph.

Local Clustering Coefficient (LCC): The local clustering coefficient measures how connected the neighbors of a particular node are to each other. For a node v, it is defined as the ratio of the number of edges between the neighbors of v (i.e., the number of triangles that include v) to the total number of possible edges between those neighbors.

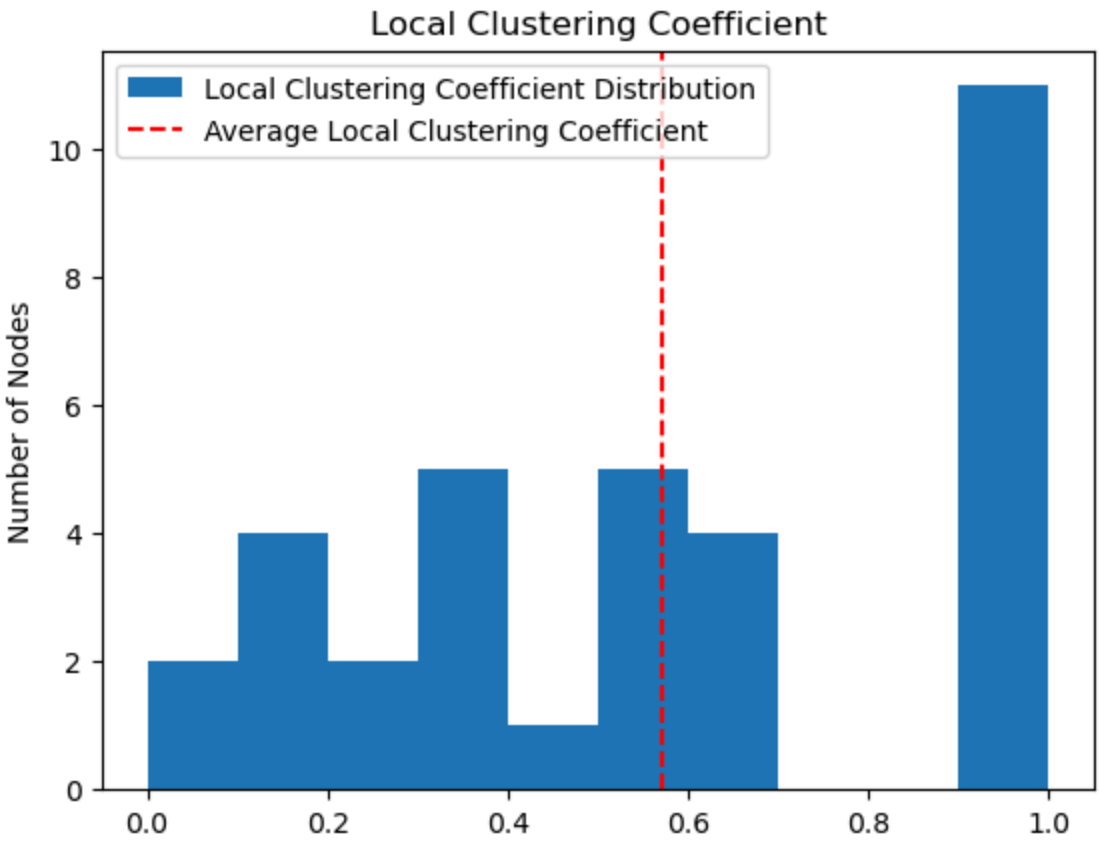
A clustering coefficient value of 1 indicates perfect clustering within a neighborhood. Specifically, for a node v, a clustering coefficient of 1 means that all of the node's neighbors are mutually connected. In other words, the neighbors of the node form a complete subgraph (or clique).

A clustering coefficient value of 0.5 for a node means that about 50% of the neighbors of a given node are connected to each other.

A clustering coefficient value of 0 indicates no clustering within the neighborhood of a node.

In [21]:
```
# Compute the local clustering coefficient for each node in the graph
# `nx.algorithms.cluster.clustering` returns a dictionary where keys are nodes
# and values are their respective local clustering coefficients.
local_clustering_coefficient = nx.algorithms.cluster.clustering(ZKC_graph)

# Calculate the average clustering coefficient across all nodes
# Sum the local clustering coefficients of all nodes and divide by the total number
av_local_clustering_coefficient = sum(local_clustering_coefficient.values()) / len(

# Print the average clustering coefficient
print(av_local_clustering_coefficient)

# Plot a histogram of the local clustering coefficient distribution
# `plt.hist` visualizes the frequency of local clustering coefficients for nodes in
plt.hist(local_clustering_coefficient.values(), label='Local Clustering Coefficient

# Add a vertical dashed line at the average clustering coefficient for reference
# `axvline` draws a vertical line on the plot.
plt.axvline(av_local_clustering_coefficient, color='r', linestyle='dashed', label='

# Add a legend to the plot to describe the histogram and average line
plt.legend()

# Label the y-axis to indicate that it shows the number of nodes in each bin
plt.ylabel('Number of Nodes')

# Add a title to the plot for better understanding of the visualization
plt.title('Local Clustering Coefficient')

# Display the plot
plt.show()
```

```
0.5706384782076823
```



Community Analysis

There are many algorithms that exist to identify communities based on the measures discussed aboved. Networkx has a number of community identification algorithms. A greedy approach is shown below.

Greedy modularity maximization (Clauset-Newman-Moore) begins with each node in its own community and joins the pair of communities that most increases modularity until no such pair exists.

See more about modularity here:
https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithm

In [24]:
```
# Import the `greedy_modularity_communities` function from NetworkX's community mod
# This function implements a greedy algorithm to detect communities by maximizing m
from networkx.algorithms.community.modularity_max import greedy_modularity_communit

# Perform community detection on the given graph `ZKC_graph`
# The function returns a list of sets, where each set represents a detected communi
c = list(greedy_modularity_communities(ZKC_graph))

# Print the number of communities detected
# `len(c)` gives the number of distinct communities identified by the algorithm.
print(len(c))
```

3

In [26]:
```
# Sort the nodes in the first community and assign it to `community_0`
# Sorting ensures the nodes in each community are listed in a consistent order.
community_0 = sorted(c[0])

# Sort the nodes in the second community and assign it to `community_1`
community_1 = sorted(c[1])

# Sort the nodes in the third community and assign it to `community_2`
community_2 = sorted(c[2])

# Print the nodes in the first community
print(community_0)

# Print the nodes in the second community
print(community_1)

# Print the nodes in the third community
print(community_2)
```

```
[8, 14, 15, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33]
[1, 2, 3, 7, 9, 12, 13, 17, 21]
[0, 4, 5, 6, 10, 11, 16, 19]
```

Greedy Modularity Maximization Algorithm

The Greedy Modularity Maximization Algorithm is a community detection algorithm used to identify groups of nodes (communities) in a network, where nodes within the same community are densely connected, and connections between communities are sparse.

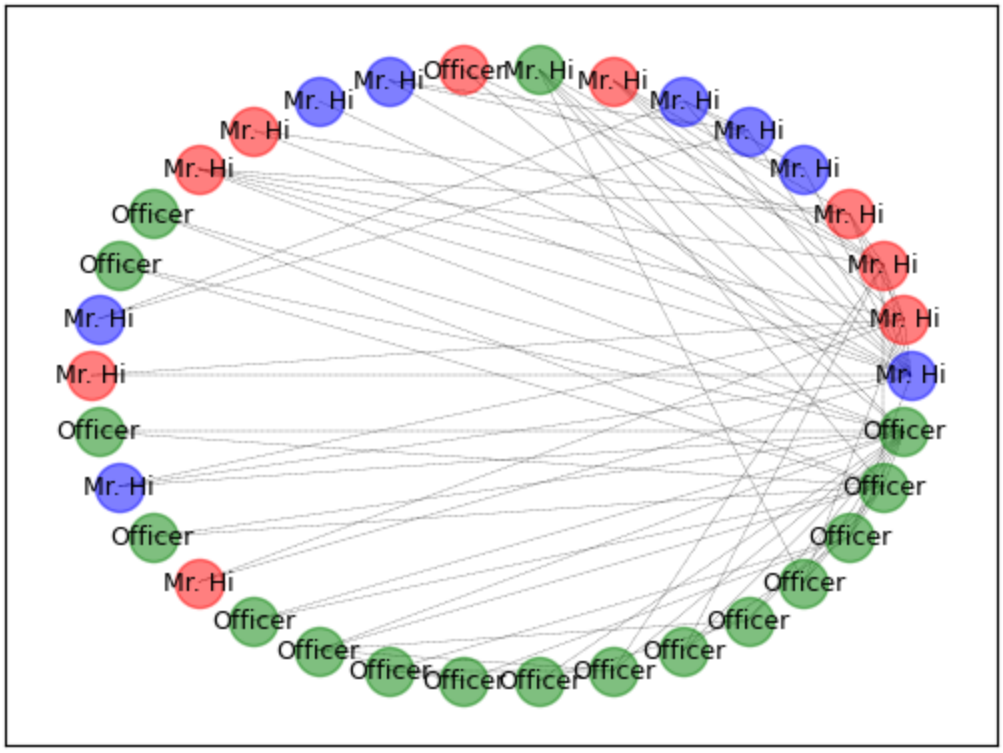
Question #1. Greedy Algorithms. (10 pts)

Given what you know about greedy algorithm schemes, on what factor is the greedy selection based in the Greedy modularity maximization algorithm (Clauset-Newman-Moore)? What computational problem (optimization or decision) does Greedy modularity maximization algorithm solve? Explain.

Answer:

In the Greedy modularity maximization algorithm, greedy selection is based upon connections between nodes. Specifically, the goal is to maximize the difference between groups in a network. The Greedy modularity maximiazation solves an optimization computational problem because it is attempting to find the solution that maximizes modulariy between groups and minimizes within group in a networkx

In [34]:
```
# Draw nodes in `community_0` with green color ('g') and alpha transparency of 0.5
# `circ_pos` contains the positions of nodes for the layout.
nx.draw_networkx_nodes(ZKC_graph, circ_pos, nodelist=community_0, node_color='g', a

# Draw nodes in `community_1` with red color ('r') and alpha transparency of 0.5
nx.draw_networkx_nodes(ZKC_graph, circ_pos, nodelist=community_1, node_color='r', a

# Draw nodes in `community_2` with blue color ('b') and alpha transparency of 0.5
nx.draw_networkx_nodes(ZKC_graph, circ_pos, nodelist=community_2, node_color='b', a

# Draw edges of the graph using dashed lines and set the width of the edges to 0.2
# `circ_pos` provides the layout positions of the nodes.
nx.draw_networkx_edges(ZKC_graph, circ_pos, style='dashed', width=0.2)

# Draw labels on the nodes corresponding to the club each member joined.
# `club_labels` should be a dictionary with node identifiers as keys and club membe
# The font size for the labels is set to 9.
nx.draw_networkx_labels(ZKC_graph, circ_pos, club_labels, font_size=9)

# Display the graph with all the visual elements
plt.show()
```



Question 2. Qualitative Assessment (20 pts)

In your opinion (and within the context of support club division) why did the community detection algorithm generate more than two communites? Did you find any other interesting observations?

Answer: Despite the data set being divided into two camps of support based on a specific policy, the algorithm detects more than two communities within the network. This could be because there is a discrepancy in how "tightly-knit" the Officer group is compared to the Mr. Hi Group, from a within group perspective. The Officer group, represented in green has the highest proportion of like-minded neighbors and often has edge connections with other Officer nodes. Whereas the Mr. Hi group has more within group variability, and is not consistent in its prioritization of edge connections to other like-minded nodes

"Tight Knit" Groups

In communities there are often subgroups that are "tightly knit". "Tight-knit" group refer to clusters of nodes that are highly interconnected, meaning there are strong relationships or connections between them. These groups exhibit a high clustering coefficient, indicating that the members of the group are closely connected to each other. A tight-knit group is not necessarily fully connected like a clique. It may have some missing edges but still exhibit a strong sense of cohesion.

Clique

A clique is a complete subgraph in which every pair of nodes is directly connected. In other words, a clique is a subset of nodes where every node has an edge to every other node in that subset.

In [31]:
```
# Find all cliques in the Zachary's Karate Club graph
cliques = list(nx.find_cliques(ZKC_graph))

# Print the list of cliques
print(cliques)
```

```
[[0, 1, 17], [0, 1, 2, 3, 13], [0, 1, 2, 3, 7], [0, 1, 19], [0, 1, 21], [0, 4, 10],
[0, 4, 6], [0, 5, 10], [0, 5, 6], [0, 8, 2], [0, 11], [0, 12, 3], [0, 31], [1, 30],
[2, 32, 8], [2, 9], [2, 27], [2, 28], [5, 16, 6], [33, 32, 8, 30], [33, 32, 14], [3
3, 32, 15], [33, 32, 18], [33, 32, 20], [33, 32, 22], [33, 32, 23, 29], [33, 32, 3
1], [33, 9], [33, 13], [33, 19], [33, 26, 29], [33, 27, 23], [33, 28, 31], [24, 25,
31], [24, 27], [25, 23]]
```

Question 3. Karate Cliques 30 pts

Lets see if we can use cliques to identify "core members" or core supporters of Mr. Hi's Group and the Officers Group. Use networkx to find the 3 or 4 largest cliques. Use find_cliques(...) or k_clique_communities(...). (Note these cliques may not be disjoint).

Visualize the results and analyze the results. Do the cliques seem to indicate "core" supporters of each group?

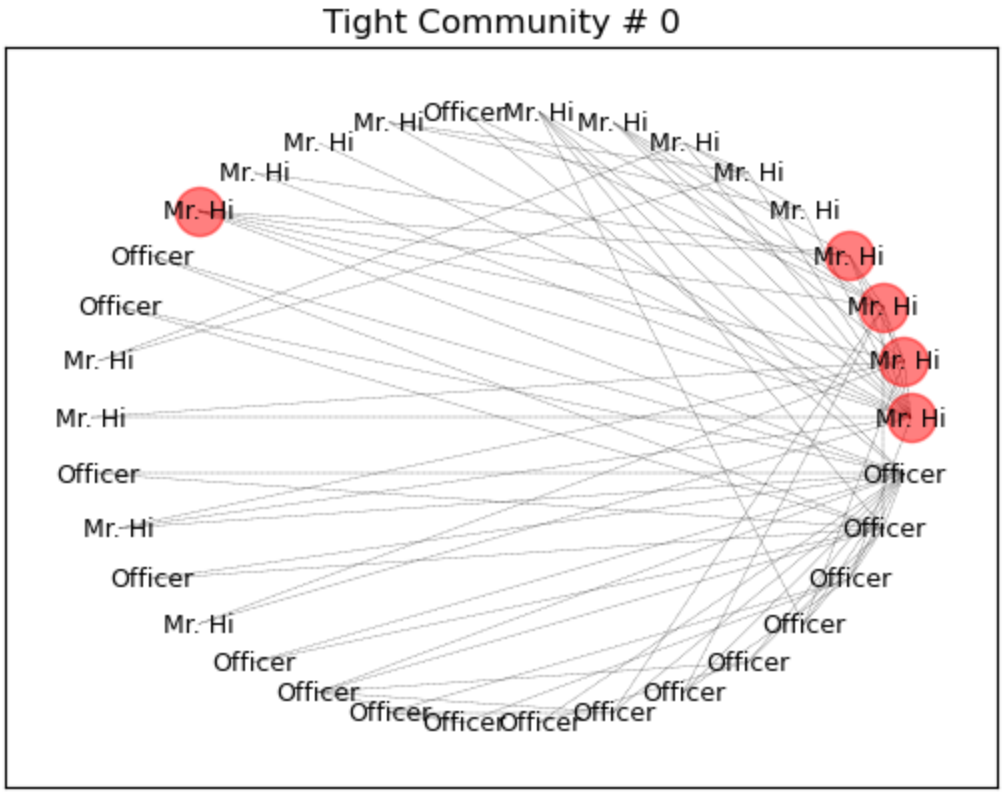
Are all clique members within the same support group? if not, explain what this might mean?

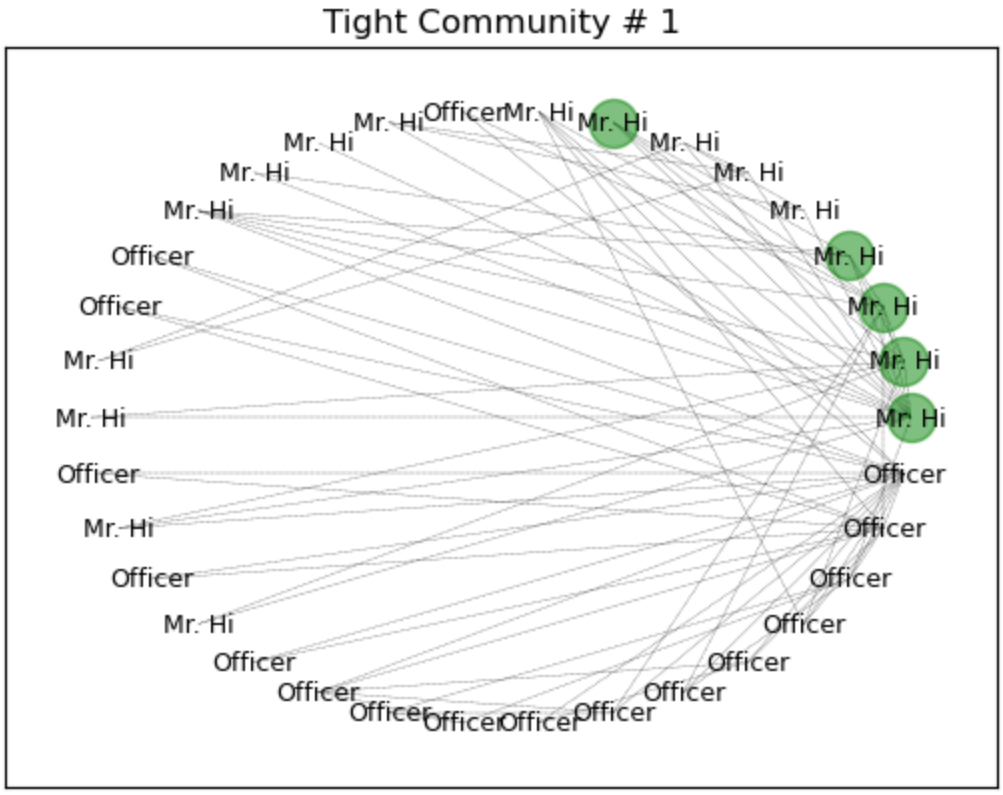
Is there any member that appears in more than one clique? if yes, explain what this might mean?

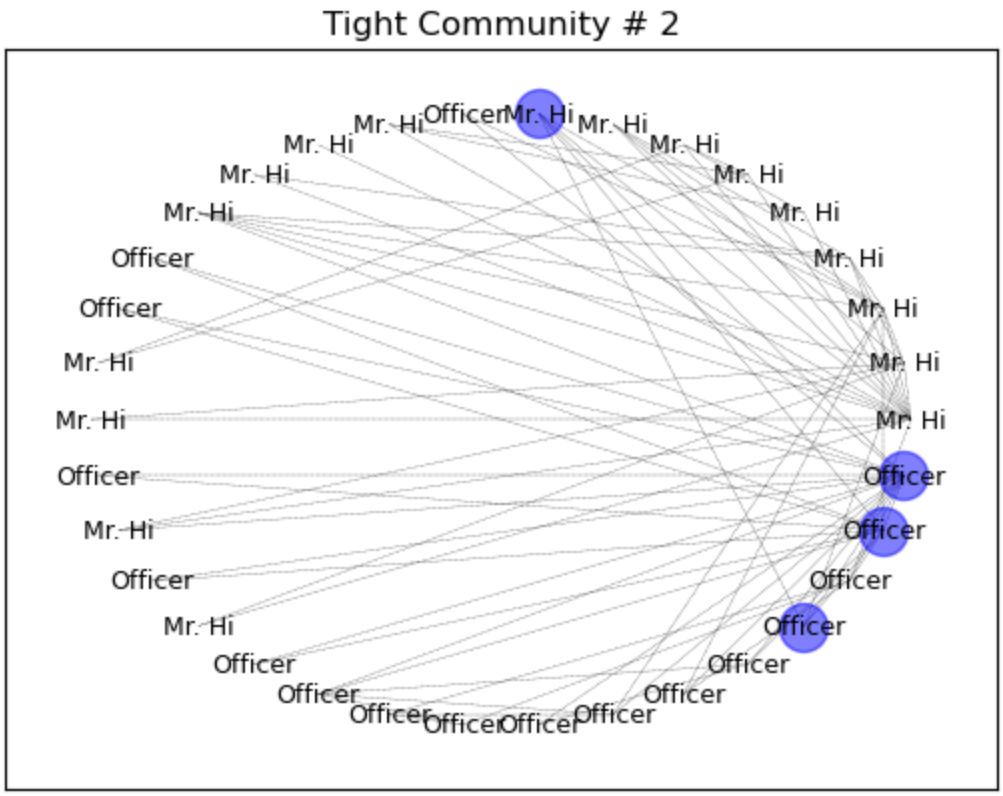
Answer:

In [157…

```
from networkx.algorithms import community

# Find largest cliques here ...
#
# INSERT YOUR CODE HERE :)

largest_cliques = list(nx.find_cliques(ZKC_graph))

# highest possible value is 5, and second highest is 4, so we'll set our algorithm
#instead of length > previous of 0 length (because the 5 length cliques occur befor
values = []
communities = []
prev = 4
for i in range(1,36):

    clique_length = len(largest_cliques[i])

    if clique_length >= prev:
        communities.append(clique_length)
        values.append(largest_cliques[i])
    else:
        None

# values of largest cliques
values

communities
# two communities of value 5, three communities of value 4

# commune =>  list that contains the communities identified by the clique communiti


color = ['r', 'g', 'b', 'c','k','m']
for i in range(len(c)):
  nx.draw_networkx_nodes(ZKC_graph,circ_pos, nodelist=values[i], node_color=color[i
  #now we can add edges to the drawing
  nx.draw_networkx_edges(ZKC_graph,circ_pos,style='dashed',width = 0.2)
  #finally we can add labels to each node corresponding to the final club each memb
  nx.draw_networkx_labels(ZKC_graph,circ_pos,club_labels,font_size=9)
  plt.title(str("Tight Community # " + str(i)))
  plt.show()
```


Tight Community # 0
Mr. Hi
Officer


Tight Community # 1
Mr. Hi
Officer



3b)

It appears that most of the largest cliques are comprised of neighbors, plus on more distant node. The cliques do not necessarily inform core supporters of the group because, even though each node is connected to the others in the clique, some nodes are connected to nodes of the opposing group; for example, as in Tight Community #1, there is a Mr Hi node with two edges connected to Officer nodes, indicating a potential lack of politically strategic selection of a network. Generally speaking, there is an element of core support because most members of this clique have majority in-group connections, however the reasons for this may be more social than political.

3c)

It is shown here that the top two clique groups are in fact within group, specifically, all for Mr. Hi's support group. The Mr. Hi group member who is in a clique with three Officer group members appears to have their degree split between Mr. Hi, and Officer nodes. This weakens the argument that large community cliques are representative of the core group of supporters. This potentially reinforces the fact that the cliques may be more socially than politically motivated.

3d)

There are a series of nodes in the second and third groups that are repeated. These appear to be mini-sub cliques inside the larger cliques. Here, I would infer that the sub clique is closer to a cohesive group, and the additional member is indicative of a variant on the sub-cliques goal. For example, maybe the sub clique shares a niche interest with these two

nodes. It should also be mentioned the members of these sub-cliques are also individually connected to a variety of other nodes, so they may simply stand out as being more social than the average node

In [147…
```
# highest possible value is 5, and second highest is 4, so we'll set our algorithm
#instead of length > previous of 0 length (because the 5 length cliques occur befor

values = []
communities = []
prev = 4
for i in range(1,36):

    clique_length = len(largest_cliques[i])

    if clique_length >= prev:
        communities.append(clique_length)
        values.append(largest_cliques[i])
    else:
        None
```

In [151…
```
# values of largest cliques
values
```

Out[151… `[[0, 1, 2, 3, 13], [0, 1, 2, 3, 7], [33, 32, 8, 30], [33, 32, 23, 29]]`

In [153…
```
communities
# two communities of value 5, three communities of value 4
```

Out[153… `[5, 5, 4, 4]`

In [ ]: