MATH3090U: Netwrok Science
Assignment 2

Syed Naqvi
100590852

October 13, 2023

1.

(a) The shortest path between nodes 1 and 7 is the path $[1, 3, 6, 7]$. The reason why this is the shortest path is because it contains a number of edges that is less than or equal to the number of edges contained in all other paths from nodes 1 to 7.

(b) The diameter of G is 4 because out of all distances between two nodes that exist in G, 4 is the maximum.

(c) The length of the longest simple cycle (circumference) of G is 4. This is because there exists no sequence beginning and ending at the same node (closed walk), that also doesnt repeat any vertices except the first and last and is longer than length 4.

(d) A bridge in graph G is defined as an edge $\{u, v\}$ such that $G - \{u, v\}$ is disconnected. Thus the only bridges in this graph are edges $\{8, 9\}$ and $\{8, 10\}$ since removing them disconnects nodes 9 and 10 from the rest of the graph. All other edges lie on cycles and so cannot be bridges since their adjacent nodes will remain connected by a path other than the removed edge.

(e) The closeness centrality of node 3 can be calculated as follows:

$$
\begin{aligned}
C_3 &= \frac{n-1}{\sum_{j \neq i} dist(i,j)} \\
C_3 &= \frac{9}{\sum_{j \neq 3} dist(3,j)} \\
&= \frac{9}{dist(3,1)+dist(3,2)+dist(3,4)+dist(3,5)+dist(3,6)+dist(3,7)+dist(3,8)+dist(3,9)+dist(3,10)} \\
&= \frac{9}{1+1+1+1+1+2+2+3+3} \\
&= \frac{9}{15} \\
&= \frac{3}{5}
\end{aligned}
$$

$\therefore$ the closesness centrality of node 3 is $\frac{3}{5}$.

(f) The betweenness centrality of node 5 is calculated as follows:

$$
\begin{aligned}
b_5 &= \sum_{\substack{s,t \\ s \neq t \neq i}} \frac{\sigma_{s,t}(i)}{\sigma_{s,t}} \\
&= \frac{\sigma_{1,2}(5)}{\sigma_{1,2}} + \frac{\sigma_{1,3}(5)}{\sigma_{1,3}} + \frac{\sigma_{1,4}(5)}{\sigma_{1,4}} + \frac{\sigma_{1,6}(5)}{\sigma_{1,6}} + \frac{\sigma_{1,7}(5)}{\sigma_{1,7}} + \frac{\sigma_{1,8}(5)}{\sigma_{1,8}} + \frac{\sigma_{1,9}(5)}{\sigma_{1,9}} + \frac{\sigma_{1,10}(5)}{\sigma_{1,10}} \\
&+ \frac{\sigma_{2,3}(5)}{\sigma_{2,3}} + \frac{\sigma_{2,4}(5)}{\sigma_{2,4}} + \frac{\sigma_{2,6}(5)}{\sigma_{2,6}} + \frac{\sigma_{2,7}(5)}{\sigma_{2,7}} + \frac{\sigma_{2,8}(5)}{\sigma_{2,8}} + \frac{\sigma_{2,9}(5)}{\sigma_{2,9}} + \frac{\sigma_{2,10}(5)}{\sigma_{2,10}} \\
&+ \frac{\sigma_{3,4}(5)}{\sigma_{3,4}} + \frac{\sigma_{3,6}(5)}{\sigma_{3,6}} + \frac{\sigma_{3,7}(5)}{\sigma_{3,7}} + \frac{\sigma_{3,8}(5)}{\sigma_{3,8}} + \frac{\sigma_{3,9}(5)}{\sigma_{3,9}} + \frac{\sigma_{3,10}(5)}{\sigma_{3,10}} \\
&+ \frac{\sigma_{4,6}(5)}{\sigma_{4,6}} + \frac{\sigma_{4,7}(5)}{\sigma_{4,7}} + \frac{\sigma_{4,8}(5)}{\sigma_{4,8}} + \frac{\sigma_{4,9}(5)}{\sigma_{4,9}} + \frac{\sigma_{4,10}(5)}{\sigma_{4,10}} \\
&+ \frac{\sigma_{6,7}(5)}{\sigma_{6,7}} + \frac{\sigma_{6,8}(5)}{\sigma_{6,8}} + \frac{\sigma_{6,9}(5)}{\sigma_{6,9}} + \frac{\sigma_{6,10}(5)}{\sigma_{6,10}} \\
&+ \frac{\sigma_{7,8}(5)}{\sigma_{7,8}} + \frac{\sigma_{7,9}(5)}{\sigma_{7,9}} + \frac{\sigma_{7,10}(5)}{\sigma_{7,10}} \\
&+ \frac{\sigma_{8,9}(5)}{\sigma_{8,9}} + \frac{\sigma_{8,10}(5)}{\sigma_{8,10}} \\
&+ \frac{\sigma_{9,10}(5)}{\sigma_{9,10}} \\
&= \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} \\
&+ \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} \\
&+ \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} \\
&+ \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} \\
&+ \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} \\
&+ \frac{0}{1} + \frac{0}{1} + \frac{0}{1} \\
&+ \frac{0}{1} + \frac{0}{1} \\
&+ \frac{0}{1} \\
&= 0
\end{aligned}
$$

$\therefore$ the betweenness centrality of node 5 is 0.

This answer makes sense because for all pairs of nodes in the graph where node 5 is not one of the nodes in the pair, any shortest path between the nodes does not include node 5.

(g) Yes, nodes 1, 2, 4, 7, 9, 10 also have betweenness centralities of 0. This makes sense because just like node 5, each of these nodes also does not lie on any shortest path between any two other nodes in the graph.

(h) This is **True**.
If we consider a node where all pairs of its adjacent nodes are connected then this node will have local clustering coefficient of 1. This also means that the shortest path from any of this node's neighbours to any other neighbour will always be 1 since they are all adjacent. It follows then that the betweenness centrality of the original node will be zero since it cannot lie on the shortest path between any pair of its neighbours.

2.

(a) To find the transitivity of graph G in question 1, we first find its adjacency matrix A as well as A^2 and A^3.

Matrix A:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Matrix A^2:

$$\begin{bmatrix} 3 & 2 & 2 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 3 & 2 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 2 & 5 & 2 & 1 & 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 4 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 2 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Matrix A^3:

$$\begin{bmatrix} 6 & 7 & 9 & 7 & 3 & 3 & 1 & 1 & 0 & 0 \\ 7 & 6 & 9 & 7 & 3 & 3 & 1 & 1 & 0 & 0 \\ 9 & 9 & 8 & 9 & 6 & 8 & 2 & 2 & 1 & 1 \\ 7 & 7 & 9 & 6 & 3 & 3 & 1 & 1 & 0 & 0 \\ 3 & 3 & 6 & 3 & 2 & 5 & 2 & 2 & 1 & 1 \\ 3 & 3 & 8 & 3 & 5 & 4 & 5 & 7 & 1 & 1 \\ 1 & 1 & 2 & 1 & 2 & 5 & 2 & 5 & 1 & 1 \\ 1 & 1 & 2 & 1 & 2 & 7 & 5 & 2 & 4 & 4 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 4 & 0 & 0 \end{bmatrix}$$

Each entry (i, j) of A^2 where $i \neq j$ is the number of simple paths of length two from node i to node j in graph G. Hence, the sum of all off-diagnoal entries in A^2 counts the total number of simple paths of length two in G (here $i \to k \to j$ and $j \to k \to i$ are different paths). Since A is an adjacency matrix for a simple undirected graph, we know that A will be symmetric, and because $(A^2)^T = (AA)^T = A^T A^T = AA = A^2$, A^2 is also symmetric. Since A^2 is symmetric, we know that the sum of all off-diagonal entries is the same as the sum of all entries above or below the main diagonal times 2.

Thus the total number of simple paths of length two in G is $2 \sum\limits_{i>j} \left(A^2\right)_{ij}$.

The trace of A^3 counts the number of simple cycles of length three in G.

We can now calculate the transitivity of G:

$$\begin{aligned} t(G) &= \frac{\text{number of simple cycles of length three in graph G}}{\text{number of simple paths of length two in G}} \\ &= \frac{trace(A^3)}{2\sum\limits_{i>j} \left(A^2\right)_{ij}} \\ &= \frac{36}{2(33)} \\ &= 0.545\overline{454} \end{aligned}$$

$\therefore$ the transitivity of the graph in question 1 is $t(G) = 0.545\overline{454}$

(b) $t(G)$ is equal to 1 only for complete graphs on n vertices. In such graphs, every 2-length simple path must necessarily be part of a closed triangle due to all nodes being connected. Furthermore, all closed triangles have the same number of 3-length simple cycles as they do 2-length simple paths. Thus for each additional 2-length simple path, we add one simple 3-length cycle and so the numerator and denominator of the transitivity fraction will always be the same for complete graphs on n nodes.

(d) For directed graph D, we count the number of edges and cycles of length two directly.

$$\begin{aligned} r(D) &= \frac{\text{cycles of length two}}{\text{number of edges in } D} \\ &= \frac{8}{13} \end{aligned}$$

$\therefore$ the reciprocity of D is $\frac{8}{13}$.

(e) The graph from part (c) is strongly connected because for all nodes $u, v \in V(D)$ where $u \neq v$, there is a directed path from u to v and from v to u.

(f) This is **True**.
If D is weakly connected, then there exists a path P for all nodes $u, v \in V(D)$ where $u \neq v$ such that, ignoring edge direction, P connects u and v. Consider now that $r(D) = 1$, this implies that for all $u, v \in V(D)$ s.t. $(u, v) \in E(D)$, it must also be the case that $(v, u) \in E(D)$ meaning that all exsiting edges in D have a reciprocal edge. Thus, each directed edge in P must have a reciprocal edge allowing us to follow P as a directed path from u to v as well as a directed path from v to u. Since u and v are arbitrary, D is strongly connected.

$\therefore$ if D is weakly-connected and $r(D) = 1$, then D must also be strongly connected.

(g) Let B be the adjacency matrix of an arbitrary directed graph D. We note that even though D is a directed graph, each entry (i, j) in matrix B^2 still represents the number of walks of length 2 from node i to node j. This is because every entry (i, k) in row i of B is the number of directed edges from node i to node k and each entry (k, j) in column j of B is the number of directed edges from node k to node j. When we find entry (i, j) in B^2 we are calculating the sum $\sum_{k=1}^{n} (B)_{ik}(B)_{k,j}$ which is only incremememented when there exist paths from i to k and from k to j, resutlting in entry (i, j) of B^2 consisting of the number of walks of length 2 from node i to node j.

Following this logic we see that values in entry (i, i) of B^2 are the number of ways to get from node i back to itself. This counts the number of cycles of length 2 that i is a part of, meaning that $trace(B^2)$ is the total number of cycles of length 2 in graph D, where $i \to k \to i$ and $k \to i \to k$ count as two different cycles. Since the trace of a matrix is equal to the sum of its eigenvalues, and if matrix B has eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_n$ then matrix B^2 has eigenvalues $\lambda_1^2, \lambda_2^2, \ldots \lambda_n^2$ and $trace(B^2) = \lambda_1^2 + \lambda_2^2 + \cdots + \lambda_n^2$.

If m is the total number of edges in D then we can write an expression for $r(D)$ as follows:

$$\begin{aligned} r(D) &= \frac{\text{number of cycles of length two}}{\text{number of edges in } D} \\ &= \frac{trace(B^2)}{m} \\ &= \frac{(\lambda_1^2 + \lambda_2^2 + \cdots + \lambda_n^2)}{m} \end{aligned}$$

$\therefore$ an expression for $r(D)$ in terms of m and the eigenvalues of the adjacency matrix of D is $\frac{(\lambda_1^2 + \lambda_2^2 + \cdots + \lambda_n^2)}{m}$.

3.

(a) For any tree with n vertices, there are $n-1$ edges. If we let L be the number of leaves then L is also the total degree of all leaves since each leaf has degree 1. We can let I be the total number of internal nodes and I_{deg} be the total internal degree. Remembering that the total graph degree is 2 times the number of edges, we can express n and $n-1$ as follows:

$$|V(T)| = n = L + I \tag{1}$$

$$|E(T)| = n - 1 = \frac{L + I_{deg}}{2} \tag{2}$$

We know that tree T in our case has 50 leaves and an equal number, x, of internal nodes with degrees 2,3,4 and 5. Thus the total internal degree will always be $I_{deg} = x(2+3+4+5) = 14x$ while the total number of internal nodes will always be $I = 4x$. Based on this information we can substitute, solve for x and then find n:

$$\begin{aligned}
|V(T)| - 1 &= |E(T)| \\
n - 1 &= \frac{L + I_{deg}}{2} \\
(L + I) - 1 &= \frac{L + I_{deg}}{2} \\
(50 + 4x) - 1 &= \frac{50 + 14x}{2} \\
100 + 8x - 2 &= 50 + 14x \\
48 &= 6x \\
x &= 8
\end{aligned}$$

solving for n we get:

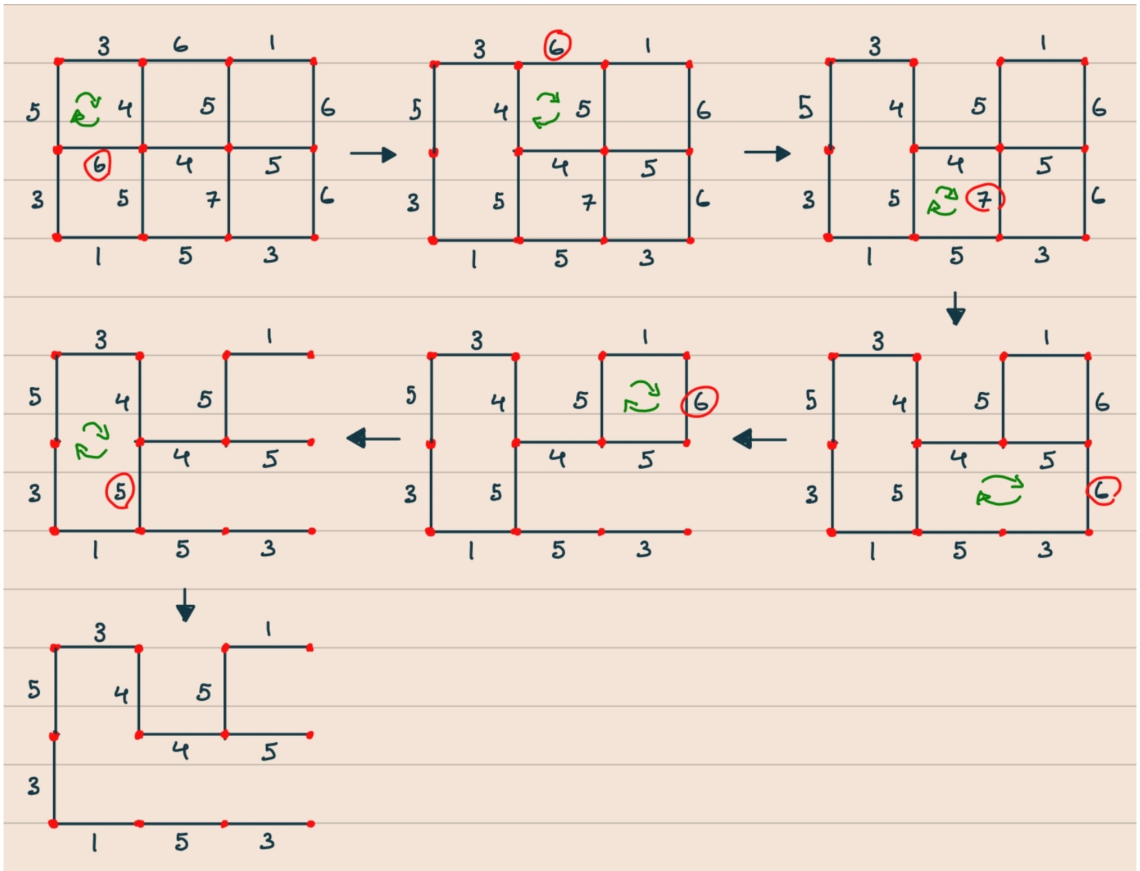
$$\begin{aligned}
n &= 50 + 4(8) \\
n &= 82
\end{aligned}$$

$\therefore T$ has 82 vertices.

4.

(a) My process for finding a minimum spanning tree is to scan the graph and choose a cycle at random. I then remove the highest weighted edge from the cycle and then select another cycle at random. I repeat this process until i arrive at a minimum spanning tree. This approach ensures that out of all edges whose removal would not disconnect the graph, only the highest weighted are removed.

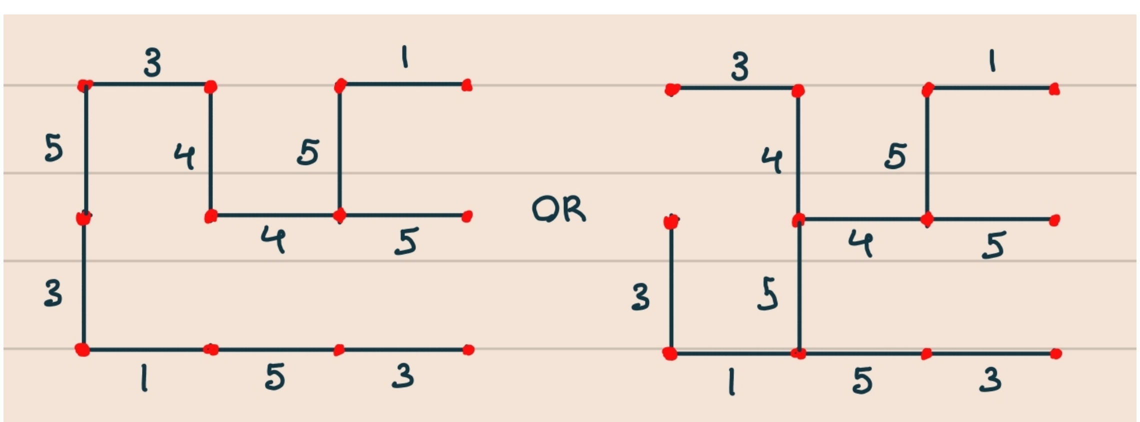
Minimum Spanning Tree



(b) The minimum spanning tree for an arbitrary weighted undirected graph is not unique in general. Take the case where there are multiple edges of the same weight in a single cycle. Removing any such edge results in the same final weight but not necessarily the same final graph.

We can illustrate by presenting two different minimum spanning trees of the graph in the previous question:

Multiple Minimum Spanning Trees



(c) This is **False**.
Counting the $n-1$ smallest edge weights arbitrarily does not gurantee the weight of a minimum spanning tree for the graph. This is due to the possibility of bridges whose weights may not fall within the $n-1$ smallest, but must be included in the count since their removal would disconnect the graph and by definition, minimum spanning trees are both connected and spanning.

5.

(a)

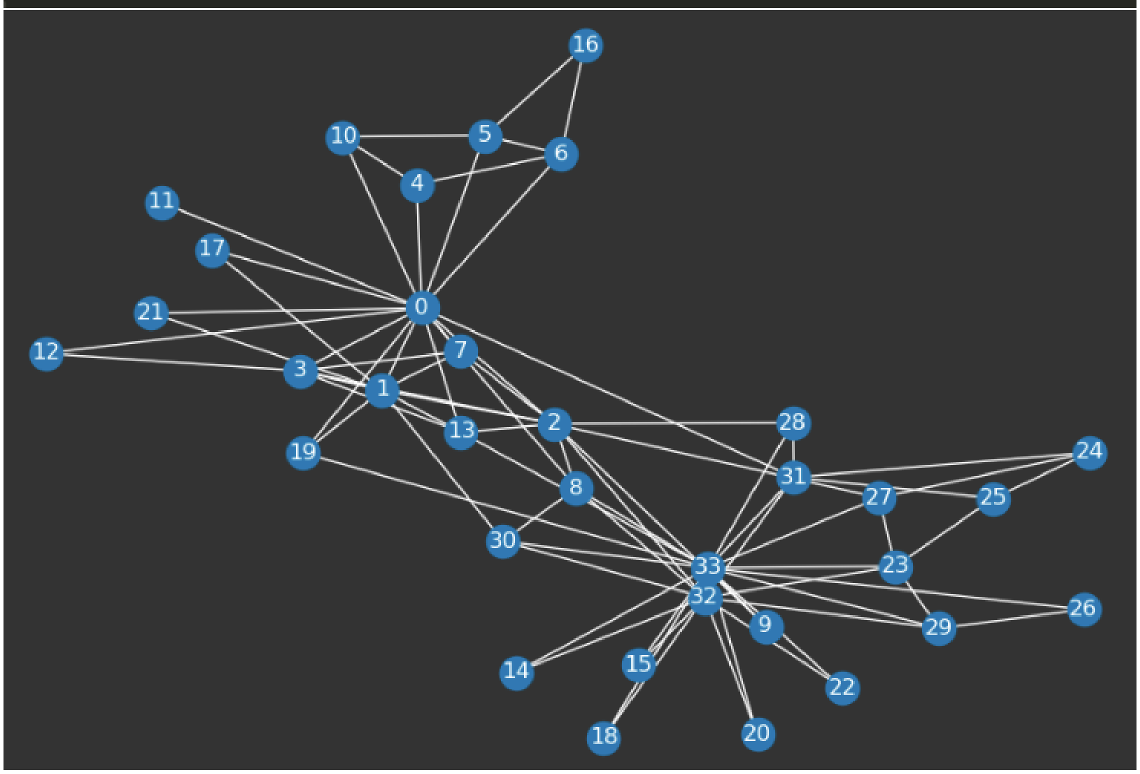
```python
#5.(a)
G = nx.karate_club_graph()
```

[14] ✓ 0.0s Python

(b)

```python
#5.(b)
fig, ax = plt.subplots()
nx.draw(G, with_labels=True, edge_color='white', font_color='white')
fig.set_size_inches(30,20)
fig.set_facecolor('#333333')
```

[32] Python



(c) Based purely on visuals of the graph drawing, nodes 0, 2, 33 and 32 seem to be the most central. Because nodes 0, 33 and 32 appear connected to so many other nodes, they likely have the highest degree centralities. Due to the sheer amount of connections, they likely fall on many shortest paths between nodes in the network also resulting in high betweenness centralities. Similarly, the large number of neighbours would mean there are many distences between them and other nodes of length 1, likely also resulting in high closeness centralities.
Although node 2 doesnt seem to have a very high degree centralitiy, its placement near the center of the graph, almost splitting the graph evenely into two, suggests that it would have a short average distance to other nodes while also lying on the shortest paths between many nodes on either ends of the graph. These aspects suggest high closeness and betweenness centralities for node 2.

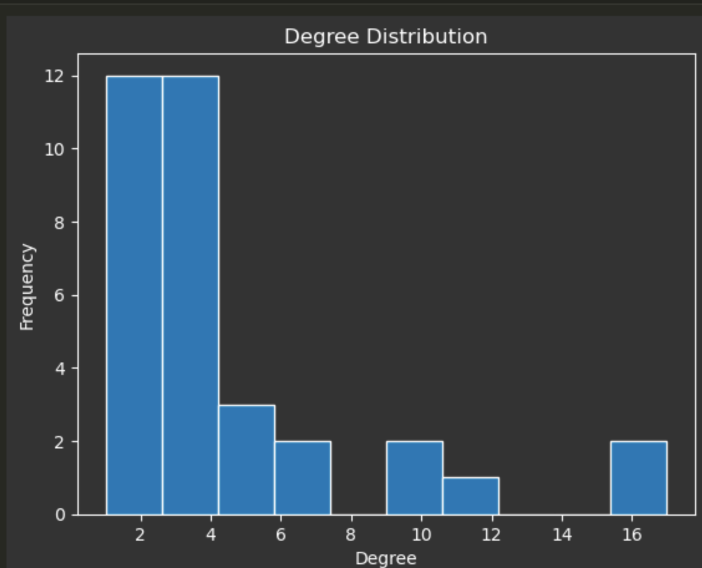
(d)

```python
#5. (d)
def degreedist(G, logs=False):
    degrees = dict(G.degree()).values()

    # Set the figure and axis background color
    fig, ax = plt.subplots()
    fig.patch.set_facecolor('#333333')
    ax.set_facecolor('#333333')

    plt.hist(degrees, log=logs, edgecolor="white")
    plt.title('Degree Distribution')
    plt.xlabel('Degree')
    plt.ylabel('Frequency')

    # Set label colors to white for visibility against dark background
    ax.xaxis.label.set_color('white')
    ax.yaxis.label.set_color('white')
    ax.title.set_color('white')
    ax.tick_params(axis='both', colors='white')
    for spine in ax.spines.values():
        spine.set_edgecolor('white')

    plt.show()

def network_statistics(G, logs=False):
    degreedist(G, logs=logs)
    print("Edge density: ", nx.density(G))
    print("Transitivity: ", nx.transitivity(G))
    print("Average path length: ", nx.average_shortest_path_length(G))
```

[105] ✓ 0.0s Python

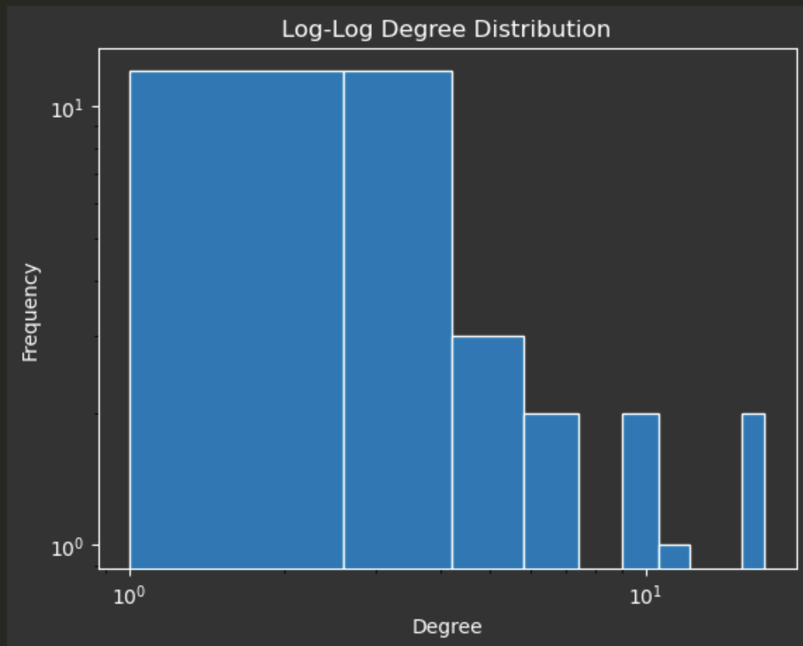
```python
network_statistics(G)
```

[106] ✓ 0.1s Python



```python
network_statistics(G, logs=True)
```

[130] 0.2s Python



From the Degree Distribution histogram we see that the majority of nodes have a low degree, around 2,3 and 4, while a smaller amount of nodes have higher degrees. This is reflected in the shape of the histogram which depicts a "long-tail" and an "L" shape, both of which are characteristic of the power-law distribution. Although the log-log graph of the distribution doesnt perfectly show a linear trend, this could be due to the relatively small size of the network so based on initial inspection the degree distribution does seem to indicate a power-law.

(e)

```python
#5. (e)
degrees = dict(nx.degree(G))

sorted_degrees = sorted(degrees.items(), key=lambda x: x[1], reverse=True)

sorted_degrees_p1 = sorted_degrees[0:len(sorted_degrees)//3]
sorted_degrees_p2 = sorted_degrees[len(sorted_degrees)//3:2*len(sorted_degrees)//3]
sorted_degrees_p3 = sorted_degrees[2*len(sorted_degrees)//3:]

print('All node degrees, greatest to least: ' )
print(dict(sorted_degrees_p1))
print(dict(sorted_degrees_p2))
print(dict(sorted_degrees_p3))
```

[19] 0.0s Python

```
All node degrees, greatest to least:
{33: 17, 0: 16, 32: 12, 2: 10, 1: 9, 3: 6, 31: 6, 8: 5, 13: 5, 23: 5, 5: 4}
{6: 4, 7: 4, 27: 4, 29: 4, 30: 4, 4: 3, 10: 3, 19: 3, 24: 3, 25: 3, 28: 3}
{9: 2, 12: 2, 14: 2, 15: 2, 16: 2, 17: 2, 18: 2, 20: 2, 21: 2, 22: 2, 26: 2, 11: 1}
```

The highest degree nodes are 33, 0 and 32. In this network, connections deptict pairs of members that interacted outside of the karate club. We could then reasonably deduce that this metric indicates how interpersonally influential or involved a particular indivdual is in the club. A high degree individual would be someone that interactes with many members outside of the club indicating high sociability, multiple friendships or large overall influence. A low degree could indicate someone that does not have many ties to other memebers and so does not have much influence.

(f)

```python
#5. (f)
closeness = sorted(nx.closeness_centrality(G).items(), key=lambda x: x[1], reverse=True)
betweenness = sorted(nx.betweenness_centrality(G).items(), key=lambda x: x[1], reverse=True)

print('All closeness centralities, greatest to least: ' )
print(dict(closeness))

print('\nAll betweenness centralities, greatest to least: ' )
print(dict(betweenness))
```

[28] ✓ 0.0s Python

```
All closeness centralities, greatest to least:
{0: 0.5689655172413793, 2: 0.559322033898305, 33: 0.55, 31: 0.5409836065573771, 8: 0.515625, 13: 0.5156

All betweenness centralities, greatest to least:
{0: 0.43763528138528146, 33: 0.30407497594997596, 32: 0.145247113997114, 2: 0.14365680615680618, 31: 0.
```

(g) The three nodes with the highest closeness centrality based on the the values calculated above are nodes 0, 2 and 33.
The three nodes with the highest betweenness centralitiy are nodes 0, 33 and 32.

These rankings largely agree other than node 2 having a higher closeness centrality than node 33 and 32 while node 32 having a higher betweenness centrality than node 2. For both centrality measures, node 0 is higher than node 33 and both are among the top three.

For the closeness centrality the node rankings make sense where 0 and 33 have low average distance to other nodes since they have large amounts of adjacent connections with distance 1. Node 2 being among the top three for this measure also make sense due to its placement near the ceneter of the graph resutling in a short average distance to other nodes on either half of the graph.

For the betweeness centrality, the top three nodes are also the nodes with the top three degree. This also makes sense because a high number of connections to other nodes means that a given node likely falls on many shortest paths between other nodes.

(h) **(i) Coloured by Degree**

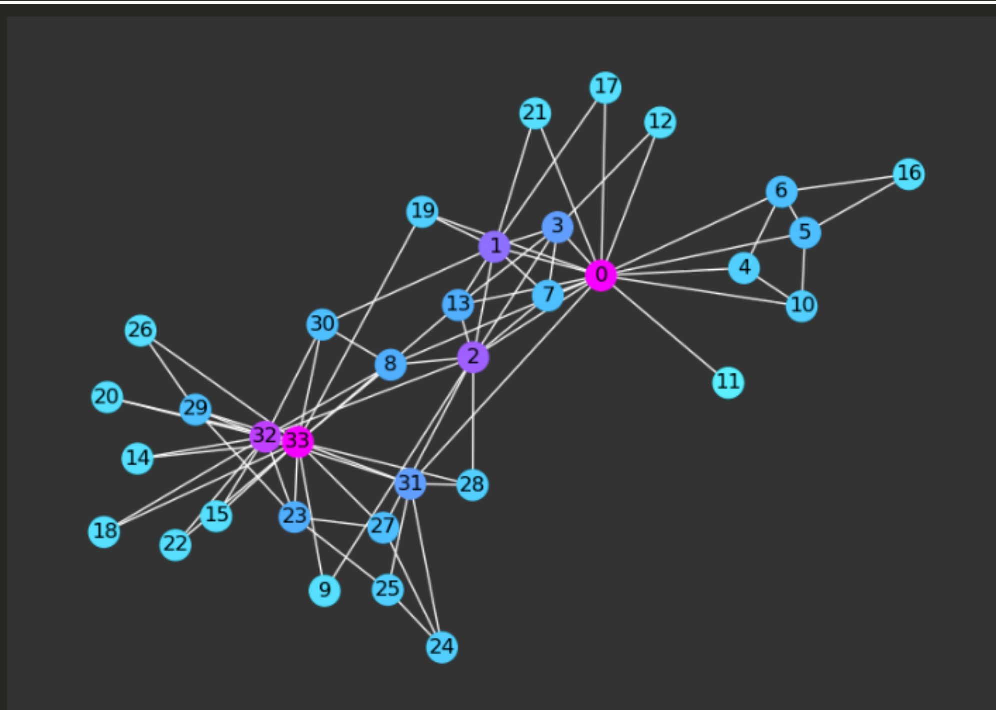
```python
#5. (h)
pos = nx.spring_layout(G)
```

[428] ✓ 0.0s Python

```python
# Pick a colormap to map degrees to colors
degrees = dict(G.degree())

cmap = plt.get_cmap('cool', max(degrees.values()) - min(degrees.values()) + 1)

# Get a list of colors for nodes based on their degrees
node_colors = [cmap(degrees[node]) for node in G.nodes()]
# Draw the graph with node colors
fig, ax = plt.su (function) with_labels: Literal[True]
nx.draw(G, pos, with_labels=True, node_color=node_colors, edge_color='white', font_color='black')
fig.set_size_inches(10,7)
fig.set_facecolor('#333333')
#plt.colorbar()
plt.show()
```

[447] ✓ 0.1s Python

...

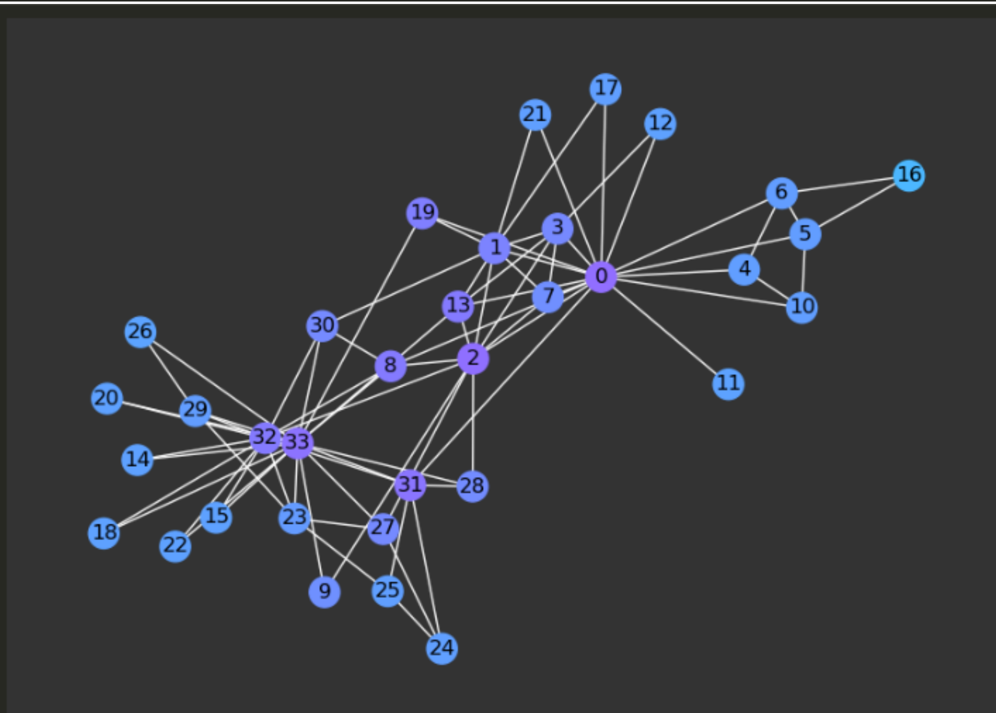


(ii) Coloured by Closeness Centrality

```
# Pick a colormap to map closeness centralities to colors
c_centrals = nx.closeness_centrality(G)

cmap = plt.get_cmap('cool')

# Get a list of colors for nodes based on their degrees
node_colors = [cmap(c_centrals[node]) for node in G.nodes()]
# Draw the graph with node colors
fig, ax = plt.su (function) with_labels: Literal[True]
nx.draw(G, pos, with_labels=True, node_color=node_colors, edge_color='white', font_color='black')
fig.set_size_inches(10,7)
fig.set_facecolor('#333333')
#plt.colorbar()
plt.show()
```

[22] ✓ 0.1s Python

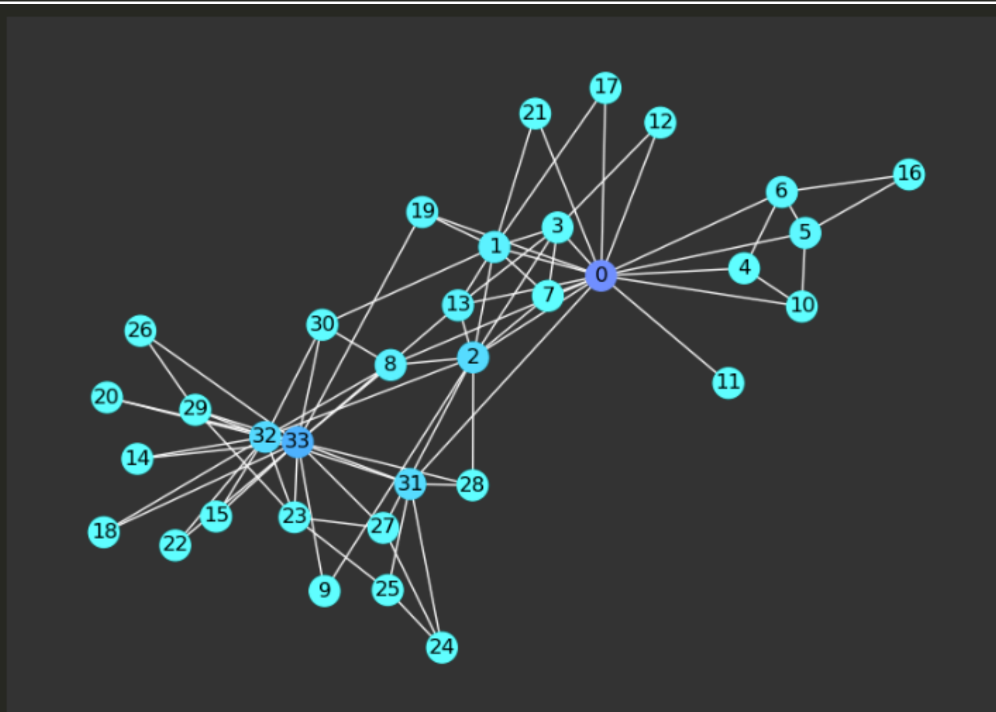
...



(iii) Coloured by Betweenness Centrality

```python
# Pick a colormap to map betweenness centralities to colors
b_centrals = nx.betweenness_centrality(G)

cmap = plt.get_cmap('cool')

# Get a list of colors for nodes based on their degrees
node_colors = [cmap(b_centrals[node]) for node in G.nodes()]
# Draw the graph with node colors
fig, ax = plt.subplots()
nx.draw(G, pos, with_labels=True, node_color=node_colors, edge_color='white', font_color='black')
fig.set_size_inches(10,7)
fig.set_facecolor('#333333')
#plt.colorbar()
plt.show()
```

[23] ✓ 0.1s Python



(i) If we are considering degree centrality as one of the centrality measures then this aspect would offer the greatest insight into the member playing the most influential role in a disagreement. It is most natural to reason that the members with the highest degrees are directly interacting with the most other members outside of the club and so will have the most opportunities to influence opinions. Thus, for important decisions and big disagreements, such memebers would cettainly hold much more sway than those who have a low degree centrality.

If we are only considering betweenness and closeness centralities then a case could be made for both. We could argue that the nodes with the leaset average distance from all other nodes (highest closeness centrality), are in a position to have their message heard by the most people since they are the fewest interactions away from everyone else. This could play a large role in influcencing others.

We could also make the point that those with the highest betweenness centralities have the largest amount of information flow through them since they stand in the way of the highest number of interaction chains between other memebrs. This could allow them to understand the points of views of many other memebera and the nuances of the disagreement better than most.