# Zachary's Karate Club Network Analysis

In [2]:
```
import networkx as nx
import matplotlib.pyplot as plt
```

## Table of Contents

1. Creating the Graph and displaying the node and edge info
2. Metadata of the actors in the network
3. Centrality-Measure
4. k-Components and Clustering Coefficient
5. Community Detection using Girvan-Newman algorithm and Louvain Method

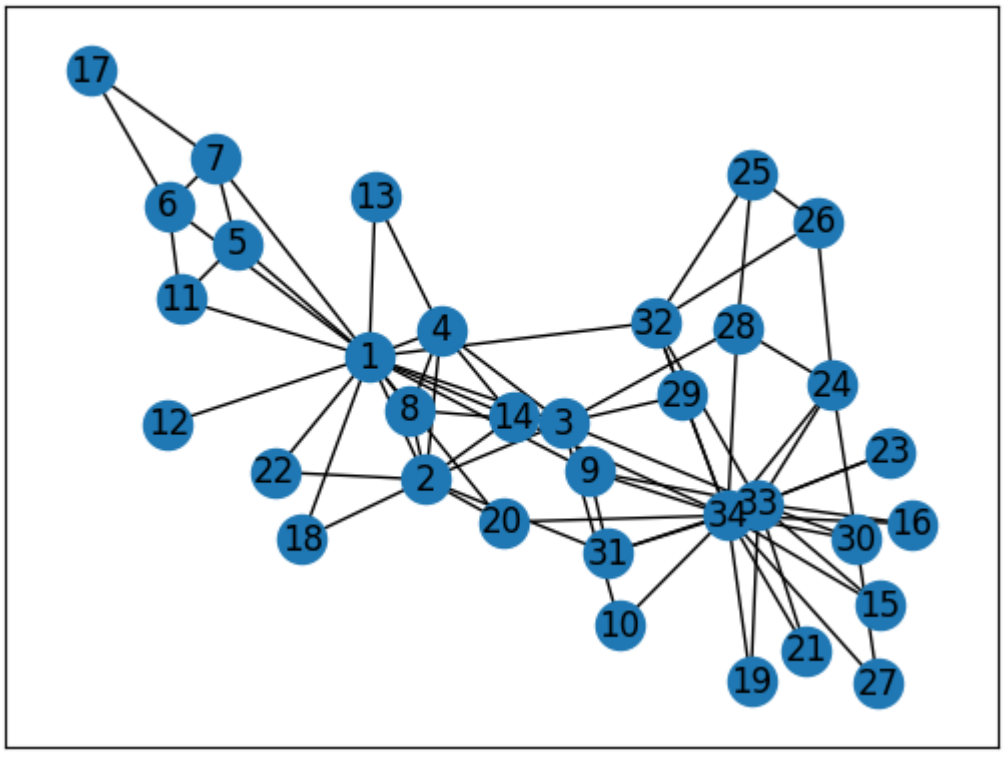
## 1. Creating the graph and displaying info on nodes and edges

In [3]:
```
G = nx.read_gml('karate.gml')
```

In [4]:
```
print(G)
print(f'Average Degree {nx.average_clustering(G):.4f}')
print("Connectivity:", nx.is_connected(G))
print("Diameter:", nx.diameter(G))
print("Average Shortest Path Length:", nx.average_shortest_path_length(G))
print("\nNodes\n")
for node in G.nodes():
    print(node, end=" ")

print("\n\nEdges\n")
for edge in G.edges():
    print(edge, end="\t")
```

```
Graph with 34 nodes and 78 edges
Average Degree 0.5706
Connectivity: True
Diameter: 5
Average Shortest Path Length: 2.408199643493761

Nodes

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 2
9 30 31 32 33 34

Edges

('1', '2')      ('1', '3')      ('1', '4')      ('1', '5')      ('1', '6')
('1', '7')      ('1', '8')      ('1', '9')      ('1', '11')     ('1', '12')
('1', '13')     ('1', '14')     ('1', '18')     ('1', '20')     ('1', '22')
('1', '32')     ('2', '3')      ('2', '4')      ('2', '8')      ('2', '14')
('2', '18')     ('2', '20')     ('2', '22')     ('2', '31')     ('3', '4')
('3', '8')      ('3', '9')      ('3', '10')     ('3', '14')     ('3', '28')
('3', '29')     ('3', '33')     ('4', '8')      ('4', '13')     ('4', '14')
('5', '7')      ('5', '11')     ('6', '7')      ('6', '11')     ('6', '17')
('7', '17')     ('9', '31')     ('9', '33')     ('9', '34')     ('10', '34')
('14', '34')    ('15', '33')    ('15', '34')    ('16', '33')    ('16', '34')
('19', '33')    ('19', '34')    ('20', '34')    ('21', '33')    ('21', '34')
('23', '33')    ('23', '34')    ('24', '26')    ('24', '28')    ('24', '30')
('24', '33')    ('24', '34')    ('25', '26')    ('25', '28')    ('25', '32')
('26', '32')    ('27', '30')    ('27', '34')    ('28', '34')    ('29', '32')
('29', '34')    ('30', '33')    ('30', '34')    ('31', '33')    ('31', '34')
('32', '33')    ('32', '34')    ('33', '34')
```

In [5]:
```
#graph representation of the data
nx.draw_networkx(G)
```



## 2. Metadata of the actors in the network

> According to information provided by Zachary, once the unity of the club was unsustainable, two groups formed and the membership for each of them is listed in the paper.

- Nodes 1 to 9, 11 to 14, 17, 18, 20 and 22 became Mr. Hi's club
- Nodes 10, 15, 16, 19, 21, 23 to 34 remained with John A

In [6]:
```
# adding metadata to the network
mr_hi = [*range(1,10), *range(11,15), 17, 18, 20, 22]

club = {}

for member in range(1,35):
    if member in mr_hi:
        club[str(member)] = "Mr. Hi"
    else:
        club[str(member)] = "John A."

nx.set_node_attributes(G, club, 'club')
```

In [7]:
```
metadata = dict(G.nodes(data=True))

metadata
```

Out[7]:
```
{'1': {'club': 'Mr. Hi'},
 '2': {'club': 'Mr. Hi'},
 '3': {'club': 'Mr. Hi'},
 '4': {'club': 'Mr. Hi'},
 '5': {'club': 'Mr. Hi'},
 '6': {'club': 'Mr. Hi'},
 '7': {'club': 'Mr. Hi'},
 '8': {'club': 'Mr. Hi'},
 '9': {'club': 'Mr. Hi'},
 '10': {'club': 'John A.'},
 '11': {'club': 'Mr. Hi'},
 '12': {'club': 'Mr. Hi'},
 '13': {'club': 'Mr. Hi'},
 '14': {'club': 'Mr. Hi'},
 '15': {'club': 'John A.'},
 '16': {'club': 'John A.'},
 '17': {'club': 'Mr. Hi'},
 '18': {'club': 'Mr. Hi'},
 '19': {'club': 'John A.'},
 '20': {'club': 'Mr. Hi'},
 '21': {'club': 'John A.'},
 '22': {'club': 'Mr. Hi'},
 '23': {'club': 'John A.'},
 '24': {'club': 'John A.'},
 '25': {'club': 'John A.'},
 '26': {'club': 'John A.'},
 '27': {'club': 'John A.'},
 '28': {'club': 'John A.'},
 '29': {'club': 'John A.'},
 '30': {'club': 'John A.'},
 '31': {'club': 'John A.'},
 '32': {'club': 'John A.'},
 '33': {'club': 'John A.'},
 '34': {'club': 'John A.'}}
```

## 3. Centrality Measure

In [8]:
```python
# Degree Centrality
dc = nx.degree_centrality(G)

# Betweenness Centrality
bc = nx.betweenness_centrality(G)

# Closeness Centrality
cc = nx.closeness_centrality(G)

# Eignenvector Centrality
ec = nx.eigenvector_centrality(G)

# PageRank Centrality
prc = nx.pagerank(G)
```

In [9]:
```python
def sort_by_value_desc(input):
    return sorted(input, key=input.get, reverse=True)[:5]
```

```
def sort_by_value_asce(input):
    return sorted(input, key=input.get)[:5]

print(sort_by_value_desc(dc))
print(sort_by_value_asce(dc))
```

```
['34', '1', '33', '3', '2']
['12', '10', '13', '15', '16']
```

In [10]:
```
print('Degree Centrality\n')

print("Highest 5 Nodes\n\nNode\tDegree Centrality")
for node in sort_by_value_desc(dc):
    print(f'{node} \t: {dc[node]:.4f}')

print("\n\nLowest 5 Nodes\n\nNode\tDegree Centrality")
for node in sort_by_value_asce(dc):
    print(f'{node} \t: {dc[node]:.4f}')
```

```
Degree Centrality

Highest 5 Nodes

Node    Degree Centrality
34      : 0.5152
1       : 0.4848
33      : 0.3636
3       : 0.3030
2       : 0.2727


Lowest 5 Nodes

Node    Degree Centrality
12      : 0.0303
10      : 0.0606
13      : 0.0606
15      : 0.0606
16      : 0.0606
```

- Node 34, 1 and 33 have the highest degree centrality, which implies that these nodes are important actors in the karate club. They are connected to most of the and probably play lead roles on behalf of the karate club.
- Whereas node 12, 10, 13, 15 and 16 are the nodes with least degree centrality.

In [11]:
```
print('Betweenness Centrality\n')

print("Highest 5 Nodes\n\nNode\tBetweenness Centrality")
for node in sort_by_value_desc(bc):
    print(f'{node} \t: {bc[node]:.4f}')

print("\n\nLowest 5 Nodes\n\nNode\tBetweenness Centrality")
```

```
for node in sort_by_value_asce(bc):
    print(f'{node} \t: {bc[node]:.4f}')
```

```
Betweenness Centrality

Highest 5 Nodes

Node    Betweenness Centrality
1       : 0.4376
34      : 0.3041
33      : 0.1452
3       : 0.1437
32      : 0.1383


Lowest 5 Nodes

Node    Betweenness Centrality
8       : 0.0000
12      : 0.0000
13      : 0.0000
15      : 0.0000
16      : 0.0000
```

- Node 1 has the highest Betweenness centrality, and the actor acts as the connecting bridge between different communitites. Node 1 also relays most the information that flows throught the network.
- The lowest 5 nodes all have 0 betweenness, therefore they do not participate in information sharing and have no mutual connections.

In [12]:
```
print('Closeness Centrality\n')

print("Highest 5 Nodes\n\nNode\tCloseness Centrality")
for node in sort_by_value_desc(cc):
    print(f'{node} \t: {cc[node]:.4f}')

print("\n\nLowest 5 Nodes\n\nNode\tCloseness Centrality")
for node in sort_by_value_asce(cc):
    print(f'{node} \t: {cc[node]:.4f}')
```

```
Closeness Centrality

Highest 5 Nodes

Node    Closeness Centrality
1       : 0.5690
3       : 0.5593
34      : 0.5500
32      : 0.5410
9       : 0.5156


Lowest 5 Nodes

Node    Closeness Centrality
17      : 0.2845
27      : 0.3626
12      : 0.3667
13      : 0.3708
15      : 0.3708
```

- From the above output, all the highest 5 nodes are closely connected to each other, which loosely defines an unofficial group or community. The members interact with each other.
- It can be said that, the lowest ranked nodes lie on the boundary of the network.

In [13]:
```
print('Eigenvector Centrality\n')

print("Highest 5 Nodes\n\nNode\tEigenvector Centrality")
for node in sort_by_value_desc(ec):
    print(f'{node} \t: {ec[node]:.4f}')

print("\n\nLowest 5 Nodes\n\nNode\tEigenvector Centrality")
for node in sort_by_value_asce(ec):
    print(f'{node} \t: {ec[node]:.4f}')
```

```
Eigenvector Centrality

Highest 5 Nodes

Node    Eigenvector Centrality
34      : 0.3734
1       : 0.3555
3       : 0.3172
33      : 0.3087
2       : 0.2660


Lowest 5 Nodes

Node    Eigenvector Centrality
17      : 0.0236
12      : 0.0529
25      : 0.0571
26      : 0.0592
27      : 0.0756
```

- The nodes with highest values are closely connected group comprising of influencial individuals. Every individual in this group is an important and influencial figure.
- The least score nodes are those which are least influential and least influeneced.

In [14]:
```python
print('Page Rank\n')

print("Highest 5 Nodes\n\nNode\tPage Rank")
for node in sort_by_value_desc(prc):
    print(f'{node} \t: {prc[node]:.4f}')

print("\n\nLowest 5 Nodes\n\nNode\tPage Rank")
for node in sort_by_value_asce(prc):
    print(f'{node} \t: {prc[node]:.4f}')
```

```
Page Rank

Highest 5 Nodes

Node    Page Rank
34      : 0.1009
1       : 0.0970
33      : 0.0717
3       : 0.0571
2       : 0.0529


Lowest 5 Nodes

Node    Page Rank
12      : 0.0096
10      : 0.0143
15      : 0.0145
16      : 0.0145
19      : 0.0145
```

- PageRank defines how important a node is.
- Node 34, 1, 33 and 3 are the most important(in ascending order) actors in the Karate Club, with probablity of more links to them
- All the lowest ranked nodes are the least important figures in the network

## 4. k-Components and Clustering Coefficient

In [15]:
```python
components = nx.k_components(G)
for k,v in components.items():
    print(f'{k}-Component nodes\n----------------------------')
    for nodeset in v:
        print('Node Set:', end = '')
        for node in nodeset:
            print(node, end=' ')
        print('\n')
```

```
4-Component nodes
----------------------------
Node Set:34 8 3 1 31 33 4 9 14 2

3-Component nodes
----------------------------
Node Set:34 8 3 26 1 30 33 24 29 20 25 31 32 4 9 28 14 2

Node Set:6 5 7 1 11

2-Component nodes
----------------------------
Node Set:34 8 3 26 1 30 33 18 21 27 15 19 29 20 24 28 13 25 31 10 16 32 4 9
22 14 2 23

Node Set:6 5 7 1 17 11

1-Component nodes
----------------------------
Node Set:34 8 1 30 33 21 27 17 29 19 5 20 13 25 31 6 4 22 14 2 18 11 23 7 26
15 24 10 16 32 9 28 3 12
```

In [16]:
```python
#Clustering Coefficient
print('Clustering Coefficient for each node\n')
coeff = nx.clustering(G)
for k,v in coeff.items():
    print(f'{k} : {v:.2f}', end='\t')
print('')

print('\n\nAverage Clustering Coefficient\n')

avg_coeff = nx.average_clustering(G)
print(f'{avg_coeff:.4}')
```

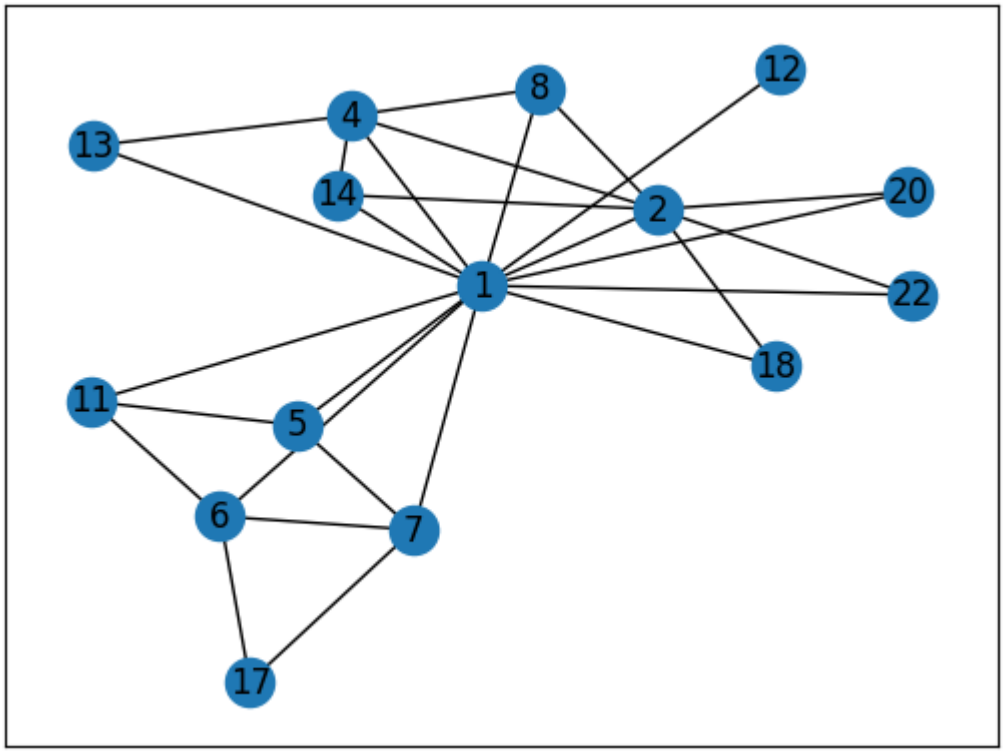
```
Clustering Coefficient for each node

1 : 0.15        2 : 0.33        3 : 0.24        4 : 0.67        5 : 0.67
6 : 0.50        7 : 0.50        8 : 1.00        9 : 0.50        10 : 0.00
11 : 0.67       12 : 0.00       13 : 1.00       14 : 0.60       15 : 1.00
16 : 1.00       17 : 1.00       18 : 1.00       19 : 1.00       20 : 0.33
21 : 1.00       22 : 1.00       23 : 1.00       24 : 0.40       25 : 0.33
26 : 0.33       27 : 1.00       28 : 0.17       29 : 0.33       30 : 0.67
31 : 0.50       32 : 0.20       33 : 0.20       34 : 0.11


Average Clustering Coefficient

0.5706
```

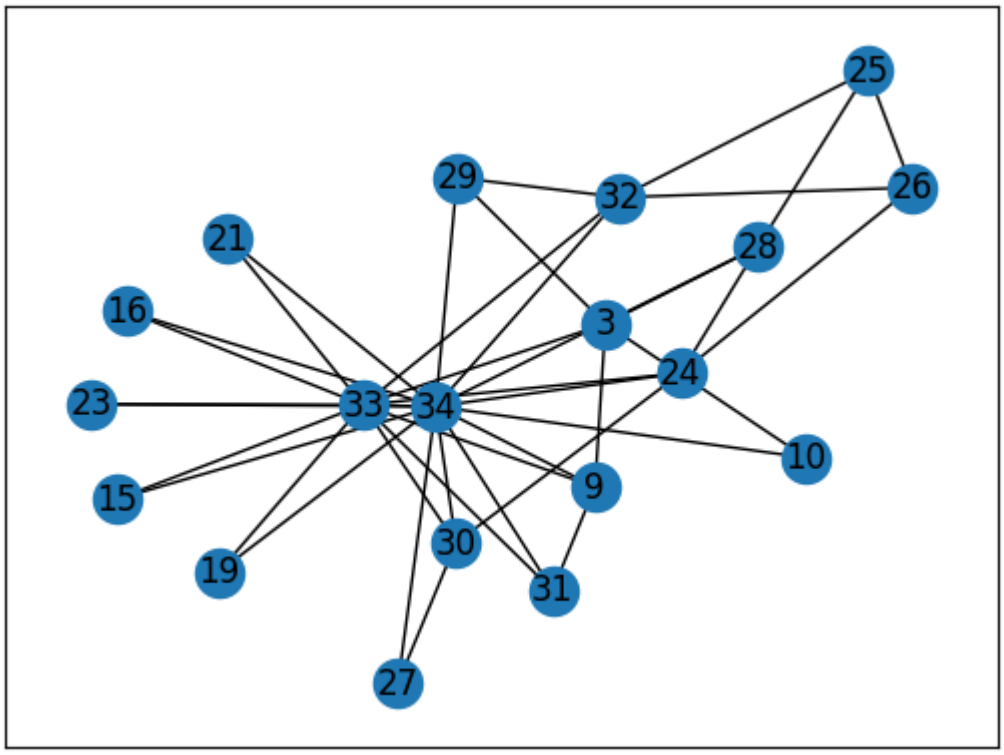
## 5. Community Detection using Girvan-Newman algorithm and Louvain Method

In [17]:
```python
# Using Girvan-Newman
from networkx.algorithms import community
```

```
comp = community.girvan_newman(G)
communities = tuple(sorted(c) for c in next(comp))
print('\nUsing Girvan-Newman Method\n')
print('No of communities found is ', len(communities),'\n')

c_1 = G.subgraph(communities[0])
c_2 = G.subgraph(communities[1])

plt.figure(1)
nx.draw_networkx(c_1)
plt.figure(2)
nx.draw_networkx(c_2)
plt.show()
```

```
Using Girvan-Newman Method

No of communities found is  2
```






In [18]:
```
#Using Louvain method
print('\nUsing Louvain Method\n')

import community

louvain = community.best_partition(G)
members_list = []

for i in set(louvain.values()):
    print(f'Community {i}')
    members = [n for n in louvain.keys() if louvain[n] == i]
    members_list.append(members)
    print(members, end='\n\n')
```

```
Using Louvain Method

Community 0
['1', '2', '3', '4', '8', '12', '13', '14', '18', '20', '22']

Community 1
['5', '6', '7', '11', '17']

Community 2
['24', '25', '26', '28', '29', '32']

Community 3
['9', '10', '15', '16', '19', '21', '23', '27', '30', '31', '33', '34']
```

In [ ]: