NAME:R.RAHINI ROLL NO:215229128

# LAB 7:DETECTING COMMUNITIES IN LARGE NETWORKS USING NETWORKX PACKAGE

## EXERCISE 1 ¶

### 1.Import the networkx package and set up the environment to initialise communities

In [1]:

```
import networkx as nx
import networkx.algorithms.community as nxcom
import matplotlib.pyplot as plt
plt.rcParams.update(plt.rcParamsDefault)
plt.rcParams.update({'figure.figsize': (15, 10)})
```

### 2.initialise the karate club graph and find the number of communities

In [2]:

```
G_karate = nx.karate_club_graph()

# Find the communities
communities = sorted(nxcom.greedy_modularity_communities(G_karate), key=len, reverse=Tru

# Count the communities
print(f"The karate club has {len(communities)} communities.")
```

```
The karate club has 3 communities.
```

### 3.Define utility functions to assgin the communities to nodes and edges

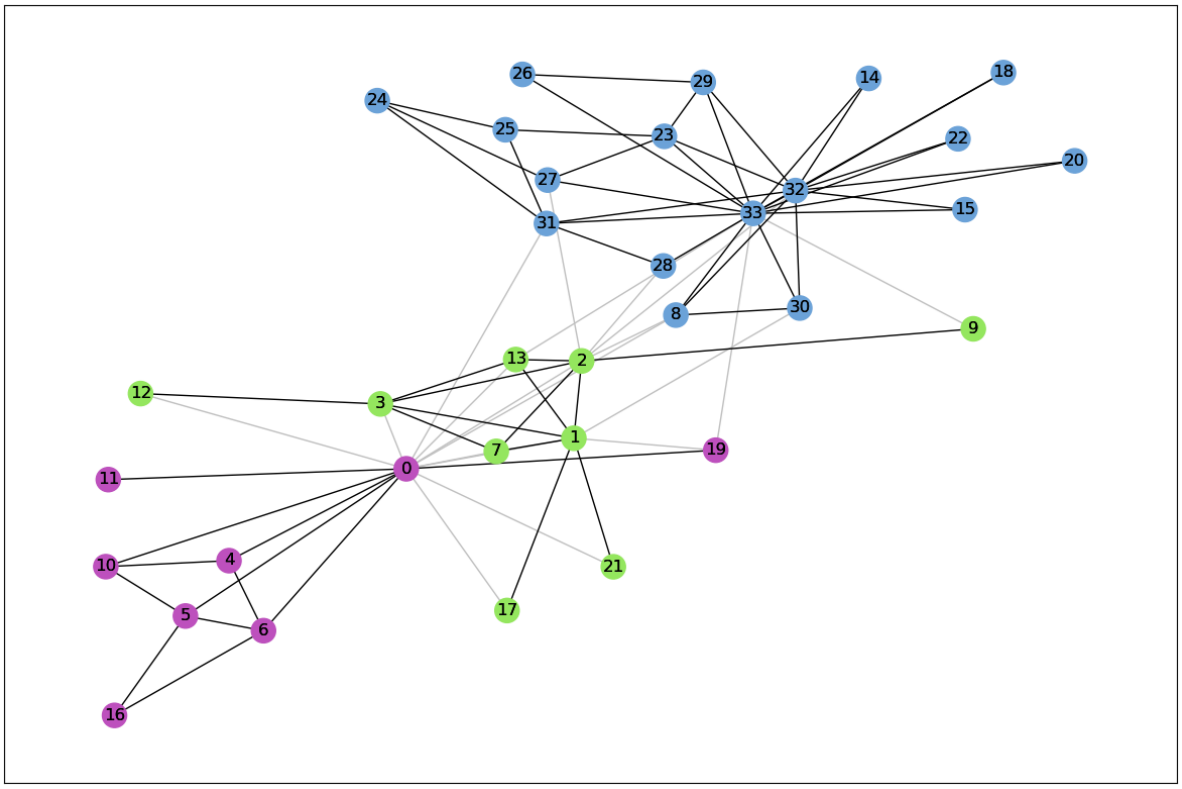
In [3]:

```
def set_node_community(G, communities):
        '''Add community to node attributes'''
        for c, v_c in enumerate(communities):
            for v in v_c:
                # Add 1 to save 0 for external edges
                G.nodes[v]['community'] = c + 1
def set_edge_community(G):
        '''Find internal edges and add their community to their attributes'''
        for v, w, in G.edges:
            if G.nodes[v]['community'] == G.nodes[w]['community']:
                # Internal edge, mark with community
                G.edges[v, w]['community'] = G.nodes[v]['community']
            else:
                # External edge, mark as 0
                G.edges[v, w]['community'] = 0
def get_color(i, r_off=1, g_off=1, b_off=1):
        '''Assign a color to a vertex.'''
        r0, g0, b0 = 0, 0, 0
        n = 16
        low, high = 0.1, 0.9
        span = high - low
        r = low + span * (((i + r_off) * 3) % n) / (n - 1)
        g = low + span * (((i + g_off) * 5) % n) / (n - 1)
        b = low + span * (((i + b_off) * 7) % n) / (n - 1)
        return (r, g, b)



# Set node and edge communities
set_node_community(G_karate, communities)
set_edge_community(G_karate)
node_color = [get_color(G_karate.nodes[v]['community']) for v in G_karate.nodes]
# Set community color for edges between members of the same community (internal) and int
external = [(v, w) for v, w in G_karate.edges if G_karate.edges[v, w]['community'] == 0]
internal = [(v, w) for v, w in G_karate.edges if G_karate.edges[v, w]['community'] > 0]
internal_color = ['black' for e in internal]
```

**4.Visualise the communities by plotting the graph**

In [4]:

```
karate_pos = nx.spring_layout(G_karate)
plt.rcParams.update({'figure.figsize': (15,10)})

# Draw external edges
nx.draw_networkx(G_karate,pos=karate_pos,node_size=0,edgelist=external,edge_color="silve

# Draw nodes and internal edges
nx.draw_networkx(G_karate,pos=karate_pos,node_color=node_color,edgelist=internal,edge_c

plt.show()
```




## EXERCISE 2

### 1.import the facebook dataset and find the communities using the above steps

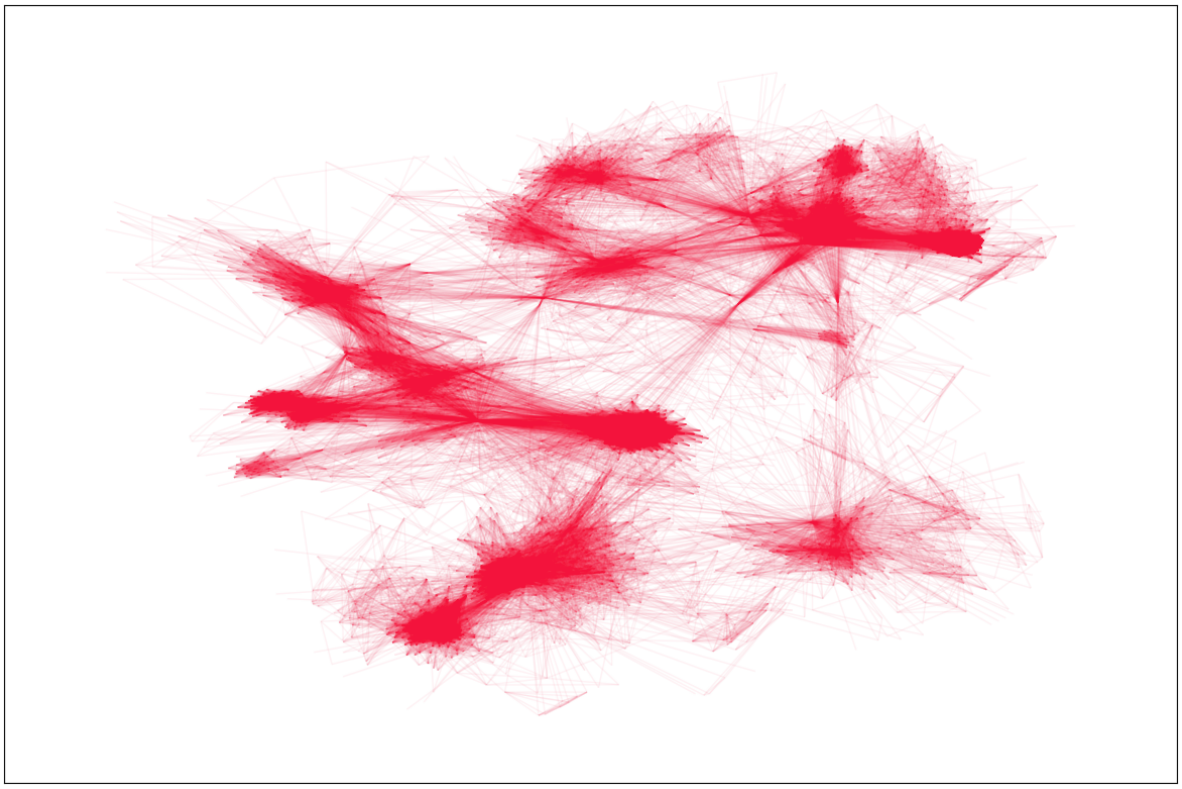
In [5]:

```
data_path = 'facebook_combined.txt'
G_social = nx.read_edgelist(data_path)
```

### 2.plot the graph and visualse it

In [6]:

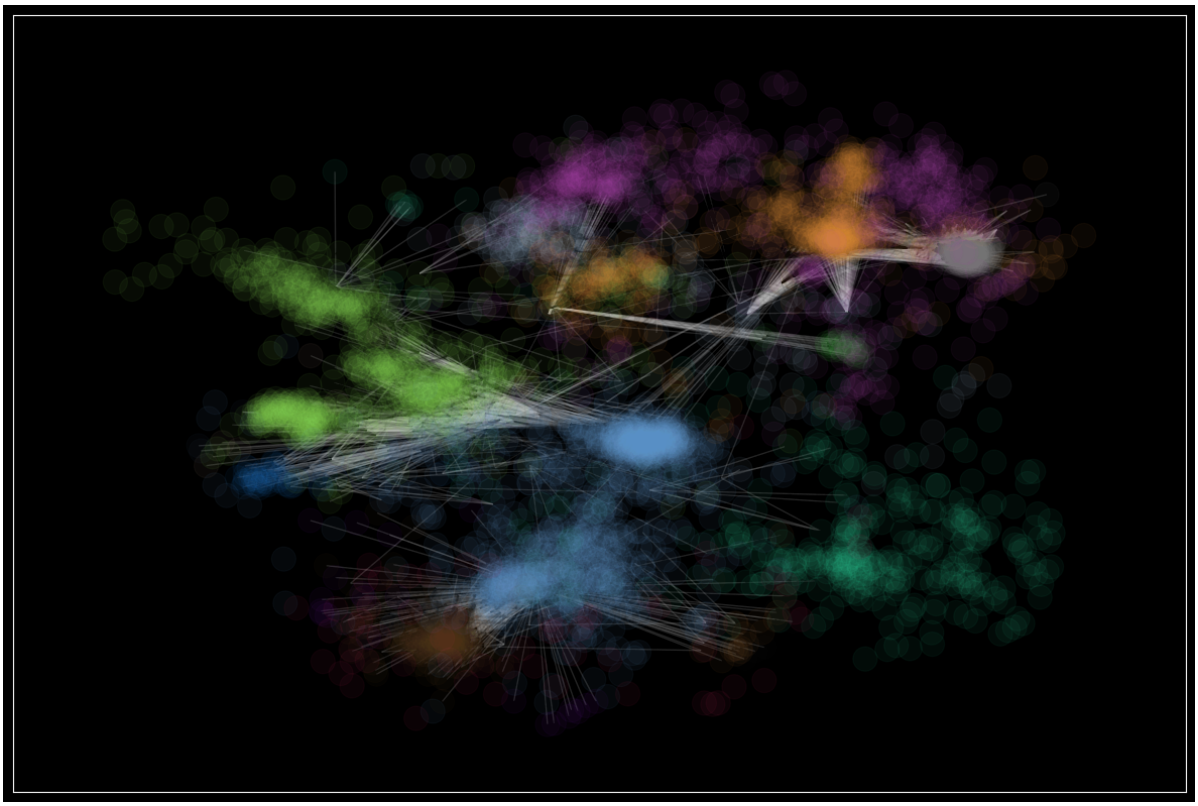
```
pos = nx.spring_layout(G_social, k=0.1)
plt.rcParams.update({'figure.figsize': (15, 10)})
nx.draw_networkx(G_social,pos=pos,node_size=0,edge_color="#F3133C",alpha=0.05,with_labe
plt.show()
```



### 3.Render the graph using the defined utility functions

In [7]:

```
communities = sorted(nxcom.greedy_modularity_communities(G_social), key=len, reverse=Tru
len(communities)

plt.rcParams.update(plt.rcParamsDefault)
plt.rcParams.update({'figure.figsize': (15, 10)})
plt.style.use('dark_background')

# Set node and edge communities
set_node_community(G_social, communities)
set_edge_community(G_social)

# Set community color for internal edges
external = [(v, w) for v, w in G_social.edges if G_social.edges[v, w]['community'] == 0]
internal = [(v, w) for v, w in G_social.edges if G_social.edges[v, w]['community'] > 0]
internal_color = ["black" for e in internal]
node_color = [get_color(G_social.nodes[v]['community']) for v in G_social.nodes]

# external edges
nx.draw_networkx(G_social,pos=pos,node_size=0,edgelist=external,edge_color="silver",node
                 alpha=0.2,with_labels=False)

# internal edges
nx.draw_networkx(G_social, pos=pos,edgelist=internal,edge_color=internal_color,node_colo
                 alpha=0.05,with_labels=False)

plt.show()
```



### 4.Apply girvan community detection to find the communities from the 2 datasets karate club and facebook

In [8]:

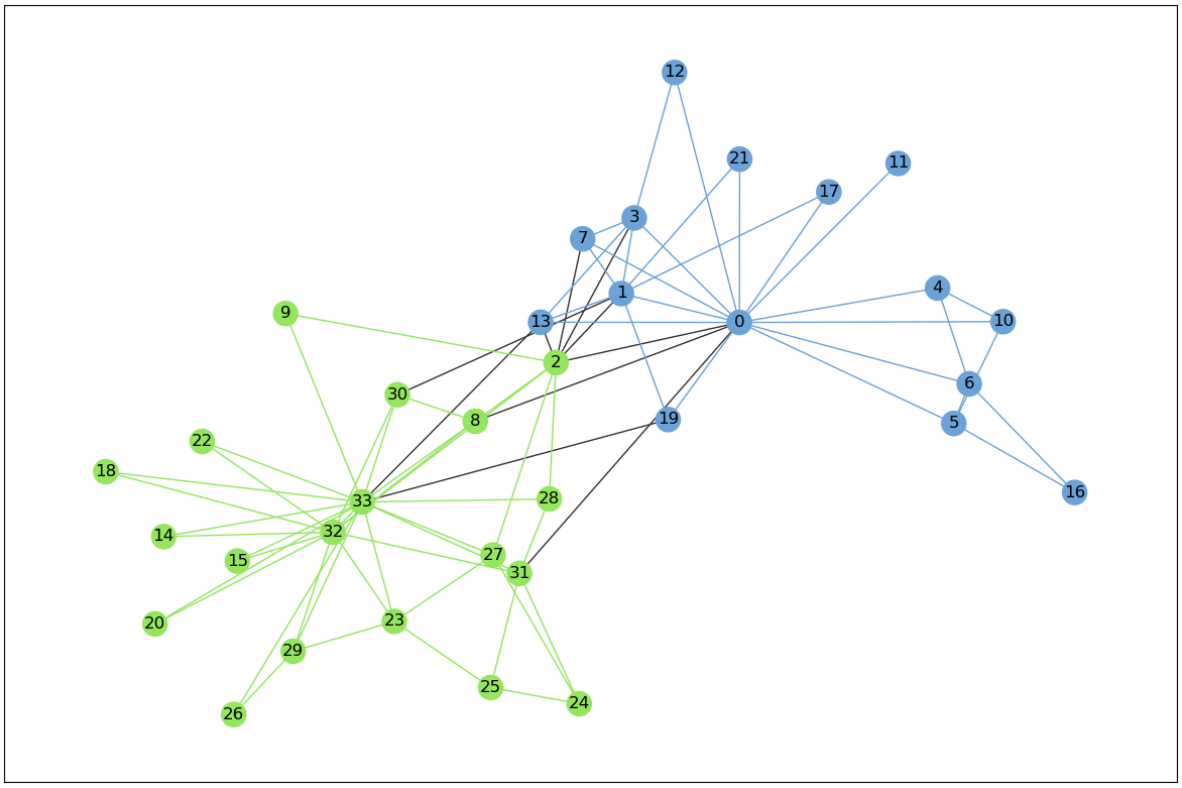
```
result = nxcom.girvan_newman(G_karate)
communities = next(result)
len(communities)
```

Out[8]:

2

In [9]:

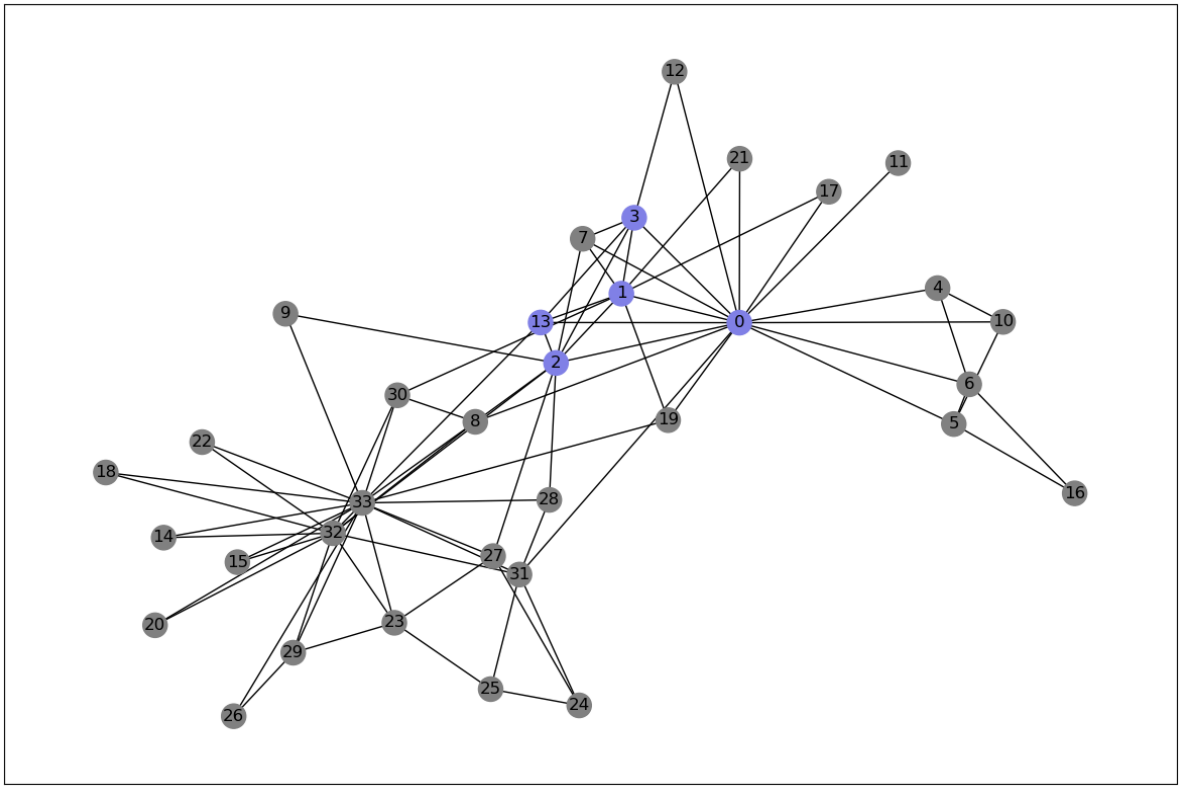
```
plt.rcParams.update(plt.rcParamsDefault)
plt.rcParams.update({'figure.figsize': (15, 10)})

# Set node and edge communities
set_node_community(G_karate, communities)
set_edge_community(G_karate)

# Set community color for nodes
node_color = [get_color(G_karate.nodes[v]['community']) for v in G_karate.nodes]

# Set community color for internal edges
external = [(v, w) for v, w in G_karate.edges if G_karate.edges[v, w]['community'] == 0]
internal = [(v, w) for v, w in G_karate.edges if G_karate.edges[v, w]['community'] > 0]
internal_color = [get_color(G_karate.edges[e]['community']) for e in internal]
karate_pos = nx.spring_layout(G_karate)

# Draw external edges
nx.draw_networkx(G_karate, pos=karate_pos, node_size=0,edgelist=external, edge_color="#
# Draw nodes and internal edges
nx.draw_networkx(G_karate, pos=karate_pos, node_color=node_color,edgelist=internal, edge
plt.show()
```




**5.Find the cliques,k-plex and k-core from the graph**

In [10]:

```
plt.rcParams.update(plt.rcParamsDefault)
plt.rcParams.update({'figure.figsize': (15, 10)})
cliques = list(nx.find_cliques(G_karate))
max_clique = max(cliques, key=len)
node_color = [(0.5, 0.5, 0.5) for v in G_karate.nodes()]
for i, v in enumerate(G_karate.nodes()):
    if v in max_clique:
        node_color[i] = (0.5, 0.5, 0.9)
nx.draw_networkx(G_karate, node_color=node_color, pos=karate_pos)

plt.show()
```



In [11]:

```
# cores with at lest degree 30
G_core_30 = nx.k_core(G_social, 30)

# similarly, with at least degree 60
G_core_60 = nx.k_core(G_social, 60)

# Visualize network and k-cores
plt.rcParams.update(plt.rcParamsDefault)
plt.rcParams.update({'figure.figsize': (15, 10)})
plt.style.use('dark_background')
pos = nx.spring_layout(G_social, k=0.1)
nx.draw_networkx(G_social, pos=pos, node_size=0, edge_color="#333333", alpha=0.05, with_
nx.draw_networkx(G_core_30, pos=pos, node_size=0, edge_color="pink", alpha=0.05, with_la
nx.draw_networkx(G_core_60, pos=pos, node_size=0, edge_color="blue", alpha=0.05, with_la
plt.show()
```