Lab class 1 Exercise Sheet

Preliminary

If you haven't had any prior experience with NetworkX already, please go through the Chapter 1 Tutorial. Then, move on to the following questions. Model answers will be provided a week later but do feel free to discuss your solutions/workings with the tutor(s)!

Question 1:

You study the mutual friendship links in your Karate club, when you notice the following: Any set of two people from the club have exactly one friend in common. What does this tell you about the club? Sketch the friendship network on paper for a club with 15 members. Once you have done that, you should be able to guess which of the NetworkX community graph generators corresponds to it. Write the code to generate the friendship network for the club.

Question 2 (a bit tricky):

A connected simple graph has a mean degree of $z \approx 1.9990123457$. The mean degree of an undirected network is given by $k = \frac{2L}{N}$ where L is the number of links and N is the number of nodes. What is the number of nodes? *(hint: draw some 'random' networks to build an intuition as to what an average degree of less than 2 could mean; you could also consult some of the network databases mentioned in the first lecture and see what typical average degrees look like)*

Question 3: The Friendship paradox

The Friendship paradox [1] is the phenomenon that, on average, your friends have more friends than you (don't feel bad about it, it's a form of sampling bias). In this exercise, I am asking you to verify this empirically in random networks. So, please complete the following code skeleton (replace ... by either numbers and/or simple line of code)

```python
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt

# Step 1: Generate a random graph.
# Here we will use a particular form of random graphs called Erdős-Rényi.
# We will discuss this model in a forthcoming lecture.
n = ...  # Number of people (nodes)
p = ...  # Probability of forming a friendship (edge)
G = nx.erdos_renyi_graph(n, p)

# Step 2: Compute degrees of all nodes
degrees = ...

# Step 3: Compute average neighbor degree for each node
neighbor_degrees = []
for node in G.nodes():
    friends = list(G.neighbors(node))
    if friends:
        avg_friend_degree = ...
    else:
        avg_friend_degree = ...
    neighbor_degrees.append(avg_friend_degree)

neighbor_degrees = np.array(neighbor_degrees)

# Step 4: Compute averages
overall_avg_degree = np.mean(degrees)
overall_avg_friend_degree = np.mean(neighbor_degrees)

print(f"Average degree of nodes: {overall_avg_degree:.2f}")
print(f"Average degree of friends: {overall_avg_friend_degree:.2f}")

# Step 5: Visualization
plt.figure(figsize=(8, 6))
plt.hist(degrees, bins=30, alpha=0.5, label="Node Degree")
plt.hist(neighbor_degrees, bins=30, alpha=0.5, label="Friend's Degree")
plt.axvline(overall_avg_degree,
    color='blue', linestyle='dashed', label="Avg Node Degree")
plt.axvline(overall_avg_friend_degree,
    color='red', linestyle='dashed', label="Avg Friend Degree")
plt.xlabel("Degree")
plt.ylabel("Frequency")
plt.legend()
plt.title("Friendship Paradox: Friends Tend to Have More Friends")
plt.show()
```

You will note that we used the erdos_renyi_graph generator. In an upcoming lecture we will discuss this graph in more detail. If you still have time in this lab class, consider using another generator, e.g., to generate a scale-free network. This would mean replacing

```
# Step 1: Generate a random graph.
n = ...  # Number of people (nodes)
p = ...  # Probability of forming a friendship (edge)
G = nx.erdos_renyi_graph(n, p)
```

by

```
# Step 1: Generate a Scale-Free (Barabási-Albert) graph
n =   # Number of nodes (people)
m =   # Number of edges a new node attaches to existing nodes
G = nx.barabasi_albert_graph(n, m)
```

Try it and see if it makes any difference. Can you explain why?

Question 4: Network visualisation

In the first lecture, I briefly mentioned that one needs to be careful with network visualisation. If you still have time, consider loading a small network from one of the many network repositories (pick a network that isn't too large) and experiment with the different layouts provided by Networkx. Remember also that within the same layout, you will get different visualisations (i.e., try the same layout many times). One useful feature of NetworkX is the ability to decide / impose positions for the nodes (i.e., you do not necessarily to let the layout function decides for you where the nodes go). A good idea then is to load file `adj.npy` containing 70 brain connectomes (each with 114 nodes). Write the code to let NetworkX decide for you how to place the nodes and then write the code to plot the nodes stored in file `coords.npy`, the 2nd scale in the Lausanne 2008 parcellation. What do you notice? *(hint: you will need the function `from_numpy_array` to load the adjacency matrix into a graph; you will also want to rotate the 3D plot interactively to recognize some of the major brain structures)*

References

1. S. L. Feld, Why Your Friends Have More Friends Than You Do. *American Journal of Sociology*, **96** (1991) 1464–1477. https://doi.org/10.1086/229693.