1. Detecting your first Community Structure

To detect communities, you can use the `cdlib` package. It also contains functions for evaluation and comparison of partitions. For details, check the documentation at [https://cdlib.readthedocs.io/en/latest/](https://cdlib.readthedocs.io/en/latest/)

If for some reason you cannot install it, you can use the `louvain_communities` function from networkx, and NMI function from sklearn, although you will need to do some transformations.

(a) Using `networkx`, load the airport dataset

(b) Using `cdlib`, detect communities on this network using the louvain method. You have to use the `algorithms.louvain` method (and do `from cdlib import algorithms` before).

(c) Visualize the communities found. In order to interpret them, you should draw each node at its geographical location, with a color per community.

There are several ways to draw a spatial network with colors corresponding to communities, from using Gephi to plotting points on an interactive map using folium.

Here, I provide a simple code to plot the data as a simple scatter plot

```python
import seaborn as sns
import matplotlib.pyplot as plt
x = list(nx.get_node_attributes(g, "lon").values())
y = list(nx.get_node_attributes(g, "lat").values())
coms_dict = coms.to_node_community_map()
hues = list(coms_dict[n][0] for n in g.nodes())
plt.figure(figsize=(12, 8))
sns.scatterplot(x=x, y=y, hue=hues, palette=sns.color_palette("tab20", len(coms.communities)), s=5)
```

(d) Vary the resolution parameter and observe changes in the community structure.

2. Comparing Partitions

(a) The provided airport data also contains information about the country of each airport, which can be interpreted as a ground truth partition of the network.

You can obtain it using `nx.get_node_attributes(g, "country")`. Transform this information into a `NodeClustering` object of `cdlib`

```python
nc = NodeClustering(partition, graph, "GroundTruth")
```

(b) Compute the AMI, NMI, or another similarity score ([https://cdlib.readthedocs.io/en/latest/reference/evaluation.html](https://cdlib.readthedocs.io/en/latest/reference/evaluation.html)) between the community structure and the partition in countries

(c) By exploring with a for loop the values of the resolution parameter for modularity, find the partition with the highest similarity to the partition in countries.

(d) Compare visually the results, and try to interpret it. Is the partition in country a meaningful topological partition, i.e., is studying this network by considering that nodes in a same country form a coherent/homogeneous group a good approach? (yes and no, probably...)
3. Understanding Modularity. To be sure that you understand correctly the modularity, write a function that compute it for a given partition and a given graph (you can start with a small graph, for instance the karate_club_graph of networkx. Make your code efficient by remembering that the score depends only on pairs of nodes inside communities. Check that you obtain the same results than using the cdlib function newman_girvan_modularity. (Be careful that if your network has weights, by default they are used by cdlib modularity)

4. Other methods
   (a) Run the kclique algorithm, with k=10.
   (b) Check the communities obtained (if you plot them, be careful that there can be one node belonging to several communities)
   (c) Identify the nodes belonging to several communities.
   (d) optional, infomap might be hard to install on some local configurations: Install the infomap additional package (pip install infomap wurlitzer). Run it and compare the communities found with the one found with louvain, in term of community sizes and in term of similarity to the country ground truth.

5. Internal evaluation
   (a) Using countries as ground truth does not makes much sense. To evaluate which partition is the most interesting, we can evaluate them with internal scores. You can check https://cdlib.readthedocs.io/en/latest/reference/evaluation.html#internal-evaluation-fitness-scores to see the ones available in CDlib.
   (b) Check the conductance definition, and then compute it on your different solutions. Check the communities of highest and lowest conductance, and the average conductance for each partition
   (c) Do the same with the avg_transitivity.
   (d) Draw a scatterplot, in which the x axis corresponds to conductance score, y corresponds to avg_transitivity, and each point is a community. You can compare two methods, by using different colors for communities from different methods.

6. Going further: Intuitions on the SBM
   (a) Starting from a reasonable partition of the graph (i.e., countries or the result of the infomap method) Compute the block matrix. You thus need to count the number of edges between and inside each community.
   (b) Using networkx stochastic_block_model method, generates a graph based on the computed block matrix
   (c) Using the network and node descriptors that you know, compare the properties of this generated graph with the properties of your original graph (and with a simple ER or configuration model). How is it different? How is it similar? Think about clustering coefficient, average distance, distribution of node centralities, degree distribution, internal transitivity, etc.
   (d) How do these properties change when you increase/decrease the number of blocks?