Table of Contents

CS4423 Assignment 1: Solution Template

This is a template for your solution to the `networkx` questions on Assignment 1.

Collaboration Policy

This is a homework assignment. You are welcome to collaborate with class-mates if you wish. Please note:

- You may collaborate with at most two other people;
- Each of you must submit your own copy of your work;
- The file(s) you submit must contain a statement on the collaboration: who you collaborated with, and on what part of the assignment.
- The use of any AI tools, such as ChatGPT or DeepSeek is prohibited, and will be subject to disciplinary procedures.

Instructions

This assignment involves a mix of questions, some of which require use of the `networkx` Python module, and some which you solve by hand. You can decide the best way to submit your work (e.g., do everything in Jupyter, or a combination of hand-written work and Jupyter notebook). However:

- Any file you submit must include your name and ID number.
- All files must be in PDF format. To convert your notebook to `pdf` the easiest method maybe to first export as 'html', then open that in a browser, and then print to pdf.

Preliminaries

Your name goes here

Your ID number goes here

Place your collaboration statement here

Usual list of Python modules

In [ ]:
```
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
opts = { "with_labels": True, "node_color": 'y' } # show labels; yellow noodes
```

Q1: Bipartite Graphs

Define and draw the following graph

Let G_1 be the graph on the nodes $\{0, 1, 2, 3, 4, 5, 6\}$ with edges $0-1, 1-2, 1-4, 1-6, 2-3$, $3-4, 4-5, 5-6$.
Define this graph in `networkx` and draw it.

In [ ]:
```
### Give your answer in this cell. Add more cells if needed.
```

Determine if G_1 is bipartite.

If G_1 is bipartite, draw it in `networkx` with a two-colouring of the nodes. If not, explain why it is not bipartite.

In [ ]:
```
### Give your answer in this cell. Add more cells if needed.
```

Q2: A Network of friends

At a party with $n = 6$ people, some people know each other already while others don't. Each of the 6 guests is asked how many friends they have at this party.
One person says they know all of the others.
One person says they know four of the others.
Two report that they know three of the others.
One person agrees they know two of the other guests, while
one person says they know only one other.

Define a graph in `networkx` that represents this scenario.

In [ ]:
```
### Give your answer in this cell. Add more cells if needed.
```

Verify that the graph has the correct properties by displaying the diagonal of the square of the graphs adjacency matrix.

Hint: `np.diag(X)` returns the entries on the main diagonal of X.

In [ ]:
```
### Give your answer in this cell. Add more cells if needed.
```

Q3: Isomorphic graphs

In `networkx` we can check if two (smallish) graphs, G and H, are **isomorphic** by using the `nx.is_isomorphic()` function: `nx.is_isomorphic(G,H)` evaluates as `True` if they are isomorphic.

Self-complementary cycle graph

Use `networkx` to check which of the cycle graphs C_3, C_4, . . ., C_{10} are isomorphic to its own complement.
Notes:

- You can use the constructor `nx.cycle_graph(n)` to make C_n
- You can use the method `nx.complement(G)` to make construct the complement of the graph G.

In [ ]:
```
### Give your answer in this cell. Add more cells if needed.
```

Graphs that are isomorphic to their line graphs.

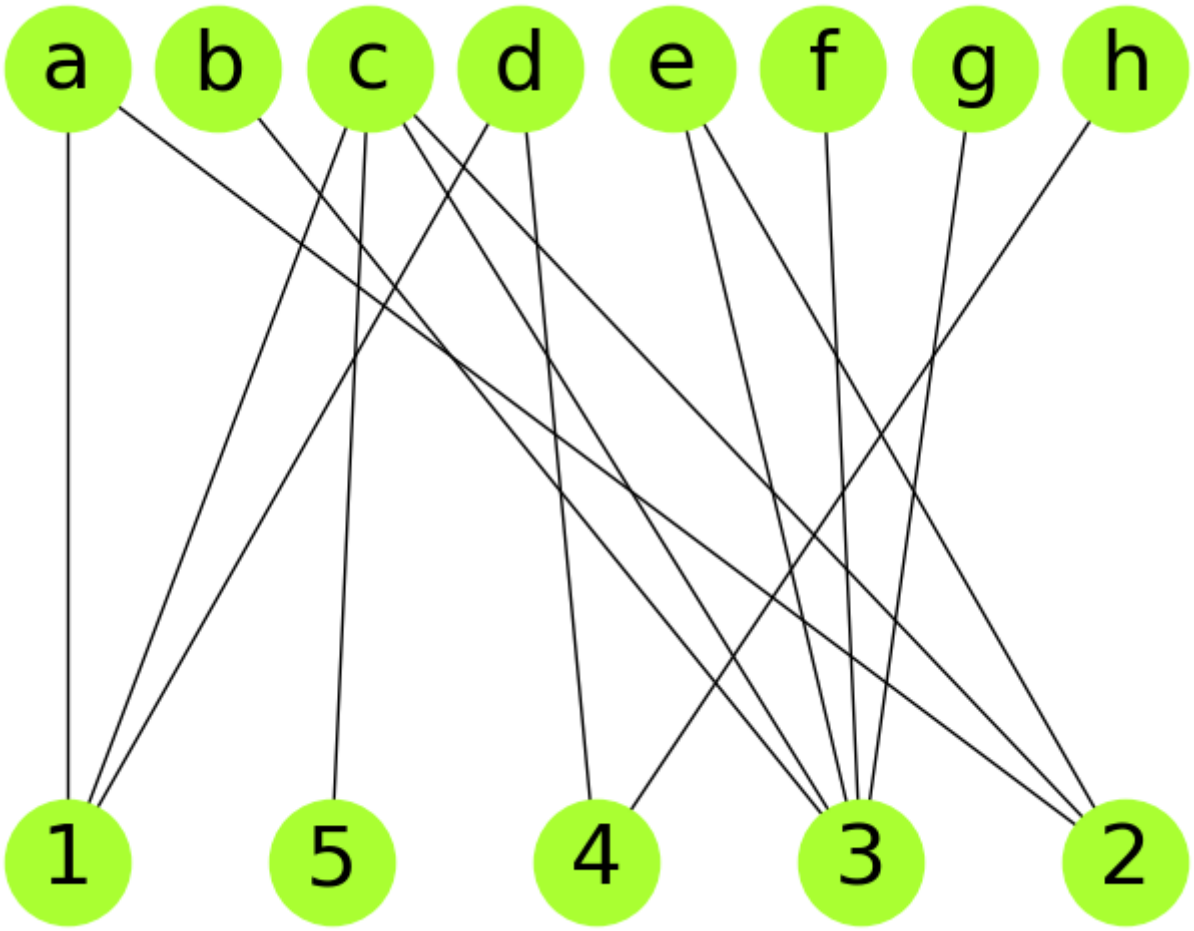
Use `networkx` to check that all cycle graphs C_3, C_4, . . ., C_{10} are isomorphic to their line graphs.

You can use the `nx.line_graph()` function.

In [ ]:
```
### Give your answer in this cell. Add more cells if needed.
```

Q4: Bipartite Graphs

Consider the following affiliation network, G_4, with 8 people labelled a to h, and five foci ("focal points" of interaction) labelled 1 to 5:



1. Create this graph in `networkx` and draw it with a two-colouring.
2. Compute the adjacency matrix of G.
3. Draw the projection on (just) the people, in which two people are joined by an edge if they have a common focus. (You can do this by hand, or in `networkx`.)

In [ ]:
```
# 1. Making and drawing the grap
### Give your answer in this cell. Add more cells if needed.
```

In [ ]:
```
# 2. Adjacency matrix
### Give your answer in this cell. Add more cells if needed.
```

In [ ]:
```
# 3. Compute and draw the projection
### Give your answer in this cell. Add more cells if needed.
```

Q5: Directed Graphs

For this question, the use of `networkx` is optional. You may write out your solution if you prefer.

So far in CS4423 we have only considered **undirected graphs**. That is, the edge $a - b$ is the same as the edge $b - a$.

Now I want you to think about *directed graphs* (also called *digraphs*): where the edge $a \to b$ is not the same as the edge $b \to a$. One can think of such edges as "one way streets": an edge that can be used to get from a to b can't be used to get from b to a.

There are numerous differences between directed and undirected graphs, including:

- When you draw a digraph you add arrows to edges to indicate its direction.

- If there is an edge $u \to v$ and $v \to u$, this can be represented by either having two edges between these nodes (with arrows in opposite directions), or by adding two arrows to a single edge.
- The adjacency matrix is not necessarily symmetric.
- The graph may have a path from node u to node v, but not from v to u.
- We talk of a digraph being
  - **Strongly Connected** meaning there is a path between every pair of nodes
  - **Weakly Connected** meaning, for every pair of nodes, u and v, there is a path from u to v, or from v to u.
  - **Disconnected** (same as the usual meaning of disconnected).
- In `networkx` we construct a directed graph with the `nx.DiGraph()` constructor.

Here is an example of a digraph in `networkx` which is strongly connected:

In [ ]:
```
G5a = nx.DiGraph(["ab", "bc", "cd", "da"])
nx.draw(G5a, **opts)
```

And here is one that is weakly connected: there is no path from c to a, for example (since d is a "dead end").

In [ ]:
```
G5b = nx.DiGraph(["ab", "bc", "cd", "ad"])
nx.draw(G5b, **opts)
```

Construct and draw a digraph

Let G_5 be the directed graph on the nodes $0, 1, 2, 3, 4$ and 5, with edges $0 \to 1, 1 \to 2, 1 \to 3, 1 \to 4$, $1 \to 5, 2 \to 4, 3 \to 2, 3 \to 4, 4 \to 3, 5 \to 0$ and $5 \to 1$.

Either by hand, or in `networkx`, draw G_5.

In [ ]:
```
## A drawing of the digraph
### Give your answer in this cell. Add more cells if needed.
```

G_5 is not strongly connected.

Show that this digraph is *not* strongly connected (i.e., find a pair of nodes between which there is no path).

In [ ]:
```
### Give your answer in this cell. Add more cells if needed.
```

Permuting the adjacency matrix.

Suppose that A is a the adjacency matrix of a digraph. Say there is a permutation matrix, P, such that

$$P^T A P = \begin{pmatrix} X & Y \\ O & Z \end{pmatrix}$$

where X and Z are square matrices and O is a zero matrix.

Explain why, if there is such a P, the graph is not strongly connected.

Write down the adjacency matrix for G_5, and also a permutation matrix P such that $P^T A P$ has the structure described above.

In [ ]:
```
### Give your answer in this cell. Add more cells if needed.
```