Homework 4

Please carefully read the instructions under "Description" in the left sidebar (you may have to expand it).

Unless otherwise specified, all graphs in this assignment refer to simple undirected graphs.

Full Name: Maximilian Comfere

UNI: mkc2182

Problem 1 (16 points)

Recall that a directed graph $G = (V, E)$ corresponds to a relation on V, with the pairings given by the elements of E. A *simple directed graph* is one without self-loops. For each of the simple directed graphs described below, identify all properties (reflexive, irreflexive, symmetric, antisymmetric, connected, transitive) that the corresponding relation has. Briefly explain each of the properties that you identify.

1. A *semicomplete directed graph*, obtained by replacing every edge $\{v, w\}$ in an undirected complete graph with at least one of two directed edges (v, w) and (w, v).

- Reflexive. All Vertices in the first graph or in the second.
- Not irreflexive.
- Asymmetric. Not all vertices can be reached in every direction in the directed graph.
- Not neccesarily transitive. There could exist an edge from a to b and b to c but not from a to c because the direction matters.

2. A *complete directed graph*, obtained by replacing every edge $\{v, w\}$ in an undirected complete graph with two directed edges (v, w) and (w, v). (No need to re-explain any properties that follow from 1.)

- Reflexive
- symmetric; the graphs are the same, with edges and the ability to traverse between edges.
- Transitive; it is complete so it is transitive.

3. An *oriented graph*, obtained by replacing every edge $\{v, w\}$ in an undirected graph with either directed edge (v, w) or (w, v) (but not both). (No need to re-explain any properties that follow from 1 or 2.)

- Reflexive
- Asymmetric
- Not neccesarily transitive.

4. A *tournament*, obtained by replacing every edge $\{v, w\}$ in an undirected complete graph with either directed edge (v, w) or (w, v) (but not both). (No need to re-explain any properties that follow from 1, 2, or 3.)

- Reflexive
- symmetric; the graphs are the same, with edges and the ability to traverse between edges.
- Transitive; it is complete so it is transitive.

YOUR SOLUTION

1.

- Reflexive. All Vertices in the first graph or in the second.
- Not irreflexive.
- Asymmetric. Not all vertices can be reached in every direction in the directed graph.
- Not neccesarily transitive. There could exist an edge from a to b and b to c but not from a to c because the direction matters.

2.

- Reflexive
- symmetric; the graphs are the same, with edges and the ability to traverse between edges.
- Transitive; it is complete so it is transitive.

3.

- Reflexive
- Asymmetric
- Not neccesarily transitive.

4.

- Reflexive

- symmetric; the graphs are the same, with edges and the ability to traverse between edges.
- Transitive; it is complete so it is transitive.

Problem 2 (12 points)

1. Prove (e.g., by contradiction) that if a graph is bipartite, then it does not contain a cycle with an odd number of vertices.

2. Let G be a graph with $n \geq 2$ vertices. Prove (e.g., by contrapositive) that if the minimum degree of G is at least $\frac{n}{2}$, then G must be connected.

YOUR SOLUTION

1.

- A bipartite graph contains an odd number of vertices
- There are an equal amount of Vertices from the two sets of the bipartite graph.
- Alternating between the two as we go through graph means that we will end on a vertex of a set different than where we started.

*Contradiction: vertices in cycle are even.

2.

- If G is not connected then the minimum degree of G is at least $n/2$
- G is not connected so some vertices are isolated and so have degree 0
- This means that the degree in one of the components is 0 and so the minimum degree in the other component must be less than $n/2$
- This means it is not at least $n/2$ so it must be connected for it to have at least $n/2$ degree.

Problem 3 (12 points)

Provide a brief explanation for each of the following statements.

1. A cycle is formed if any edge is added to a tree.

2. A tree becomes disconnected if any edge is removed from it.

3. Derive an expression for the average vertex degree of a tree in terms of $|V|$ and/or $|E|$. Is it less than, equal to, or greater than 2?

YOUR SOLUTION

1. Yes because a tree has no cycles to begin with and so by adding an edge between two vertices you make a second path from which to get between the two vertices. This creates two paths between two vertices and so a cycle.
2. Yes, a tree becomes disconnected if any edge is removed because there is only one path from each vertex back to the root so removing any edge removes that path and so disconnects the node from the tree.
3. $E = V - 1$ in a tree because adding an edge would make it a cycle. The average vertex degree would then be $(V - 1 + internalnodes)/V$. Since the internal nodes are all the nodes excluding the root, $internalnodes = V - 1 = E$ so $(V - 1 + E)/V$ which is essentially $2 - 2/V$ so the average is less than 2.

Problem 4 (14 points)

Suppose we have a *disconnected* graph in which every vertex has even degree. No Eulerian cycle exists on this graph. (Draw an example if you don't see why.)

1. Clearly explain which step(s) of the Eulerian cycle construction proof given in the Eulerian Cycle notes become invalid and why.

2. Suppose our graph consists of k components, and again all vertices have even degree. Use induction to prove that for $k \geq 2$, we can connect the graph using $k - 1$ edges so that an Eulerian trail is guaranteed to exist.

YOUR SOLUTION

1.

- Constructing a cycle in G is invalid because you cant create an arbitrary cycle, only a cycle between vertices in the same component.
- The While loop is also invalid because it will run infinetly because the graph is disconnected it will be impossible to eliminate every edge through the process it proposes.

2.

Base case

- $k = 2$
  - There are two components. They are connected and the degree is even. An Eulerian Cycle exists

Inductive Step

- Assume that $k = m$ where $m \geq 2$, m components can be connected by $m - 1$ edges
- $k = m + 1$
- G is a graph with m components
- All m components can be connected by $m - 1$ edges.
- Add any edge between connected components of m. For example C_m and C_{m+1} so that the total number of edges is m.
- That means m edges and m connected components
- $m + 1$ components can be connected by m edges.

Problem 5 (8 points)

1. Explain why it is not possible for $K_{n,n+1}$ to contain an Eulerian cycle for any $n > 0$.

2. Suppose that we have a complete bipartite graph $K_{n,n}$ with vertex sets $V_1 = \{v_1, \ldots, v_n\}$ and $V_2 = \{w_1, \ldots, w_n\}$. Give a Hamiltonian cycle on this graph and briefly explain why this is guaranteed to exist.

YOUR SOLUTION

1.

K is a bipartite graph with n vertices in the first section and n+1 vertices in the second. That means there is an imbalance which means that either n or $n + 1$ is odd. This means that it is not possible for it to contain an Eulerain Cycle because there will exist vertices with odd degree.

2.

- Start at any Vertex in V_1
- Move to matching Vertex in V_2
- Alternate between V_1 and V_2. Repeat this process, moving from one vertex to the

next closest vertex in the other section until all nodes have been visited. This cycle will include every vertex exaclty once and finish at the starting vertex.

Explanation

- The graph is complete and bipartite so every vertex in one section is connected to all the others in the other section. Everybody has a corresponding vertex so it is possible to alteranate between vertices in sections so that every vertex is visited.

Problem 6 (12 points)

Recall that two graphs are isomorphic iff for some ordering of their vertices, their adjacency matrices are equal. There is currently no known efficient way to check this for two arbitrary matrices, but we can always perform an exhaustive search for an ordering.

In other words, given two matrices M_1 and M_2, we can keep M_1 constant and attempt to reconstruct every possible variation of M_2 by reordering the rows and corresponding columns. If any variation of M_2 is equal to M_1, the corresponding graphs are isomorphic. If we find that no variation of M_2 is equal to M_1, they are not isomorphic.

Implement a function in Python following the above description. As usual, M_1 and M_2 are given as NumPy arrays. You may find the following two Python/NumPy constructs to be useful:

- `list(itertools.permutations(range(n)))` returns a list of tuples containing all possible orderings of $\{0, \ldots, n-1\}$.
- `M[np.ix_(order,order)]` returns the array `M` but with its rows and columns reordered according to the index ordering in `order`.

In [104…

```python
import numpy as np
import itertools

def isIsomorphic(M1, M2):
    # TODO: your code here
    # Returns True if M1 and M2 are isomorphic, False otherwise
    x, y = M2.shape
    p = list(itertools.permutations(range(x)))
    for i in range(len(p)):
        for j in range(len(p)):
            holder = M2[np.ix_(p[i], p[j])]
            if (holder == M1).all():
                return True
```

```
    return False
```

In [105…
```
# Johnsonbaugh 8.6, Exercise 2 (True)

M1 = np.array([[0,1,0,1,0],
               [1,0,1,1,1],
               [0,1,0,1,1],
               [1,1,1,0,0],
               [0,1,1,0,0]])

M2 = np.array([[0,1,0,1,1],
               [1,0,0,1,0],
               [0,0,0,1,1],
               [1,1,1,0,1],
               [1,0,1,1,0]])


isIsomorphic(M1,M2)
```

Out[105… True

In [106…
```
# Johnsonbaugh 8.6, Exercise 3 (True)

M1 = np.array([[0,0,0,1,1,1],
               [0,0,1,1,1,0],
               [0,1,0,1,0,1],
               [1,1,1,0,0,0],
               [1,1,0,0,0,1],
               [1,0,1,0,1,0]])

M2 = np.array([[0,1,1,0,1,0],
               [1,0,1,0,0,1],
               [1,1,0,1,0,0],
               [0,0,1,0,1,1],
               [1,0,0,1,0,1],
               [0,1,0,1,1,0]])

isIsomorphic(M1,M2)
```

Out[106… True

In [107…
```
# Johnsonbaugh 8.6, Exercise 10 (False)

M1 = np.array([[0,1,1,0,1,0],
               [1,0,1,0,0,1],
               [1,1,0,1,0,0],
               [0,0,1,0,1,1],
               [1,0,0,1,0,1],
```

```
                [0,1,0,1,1,0]])

M2 = np.array([[0,1,0,1,1,0],
                [1,0,1,0,0,1],
                [0,1,0,1,1,0],
                [1,0,1,0,0,1],
                [1,0,1,0,0,1],
                [0,1,0,1,1,0]])

isIsomorphic(M1,M2)
```

Out[107… False

In [108…

```
# Johnsonbaugh 8.6, Exercise 11 (False)

M1 = np.array([[0,1,0,1,1],
                [1,0,1,1,1],
                [0,1,0,1,0],
                [1,1,1,0,1],
                [1,1,0,1,0]])

M2 = np.array([[0,1,1,0,1],
                [1,0,1,1,1],
                [1,1,0,1,0],
                [0,1,1,0,1],
                [1,1,0,1,0]])

isIsomorphic(M1,M2)
```

Out[108… False