Specifications Grading

Before starting on this assignment, it's a good idea to consult the standard specifications and the syllabus. Here are the key points that you should keep in mind when working on this assignment.

- There is no partial credit on homework problems. You receive credit for a problem by completing the entire problem (including all parts, if applicable), to a high standard of correctness and communication. This standard is enumerated by *specifications*.
- You will have **multiple attempts** to complete each problem. After the initial submission by the first due date, your assignment will be assessed. Problems that meet specifications will receive credit. If you've attempted some problems but not met specifications, you can revise your solutions and resubmit them. If they now meet specs, you get credit!
- If you submitted a problem by the deadline with less than 50% of the problem completed, as determined by me, then you can resubmit for 50% credit.
- You don't actually have to do all homework problems assigned in this course. It's still a good idea to attempt all problems though, as this will allow you to make up for, say, a rough time on the midterm exam. The syllabus has details on how your final grade will be calculated.

Specifications

This is the list of specifications that you should meet for most problems in order to receive credit. These specifications apply to all problems that request you to write a **proof** or **argument** for a mathematical statement. I've reproduced these from the website for the first assignment; in future assignments I'll leave them off.

You can think of this like a checklist: if, for a given problem, you can check of each item, then you should expect to receive credit! Going down this checklist is exactly what the TA will do to grade your work.

Please remember that **these specifications apply to every part of a problem**. To receive credit on a problem with parts (a), (b), and (c), you need to meet the specifications on all three parts.

Correctness

- Each direction in the problem statement is followed.
  - *Note*: You are required to follow only *directions*, not *hints*. That said, I include the hints with the intention of making your life easier!
- The overall structure is mathematically sound and supports the required result.

Exposition

- Each step is carefully justified in terms of results from the course, results from the course text, or results from previous courses.
- The proof or argument is presented using clear and engaging English prose. The proof or argument is written in English sentences. There is at least as much English text as there are mathematical symbols. Grammar and spelling errors are acceptable provided that the meaning is clear.

Other

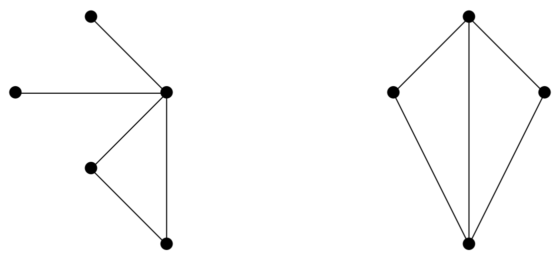
We'll also see problems in which you are expected to write some code, show a plot, write a brief reflection, or perform some other task. In this case, the specifications will be included with the problem statement.

Problem 1. (Degree, Laplacian, Connectivity)

Part (a)

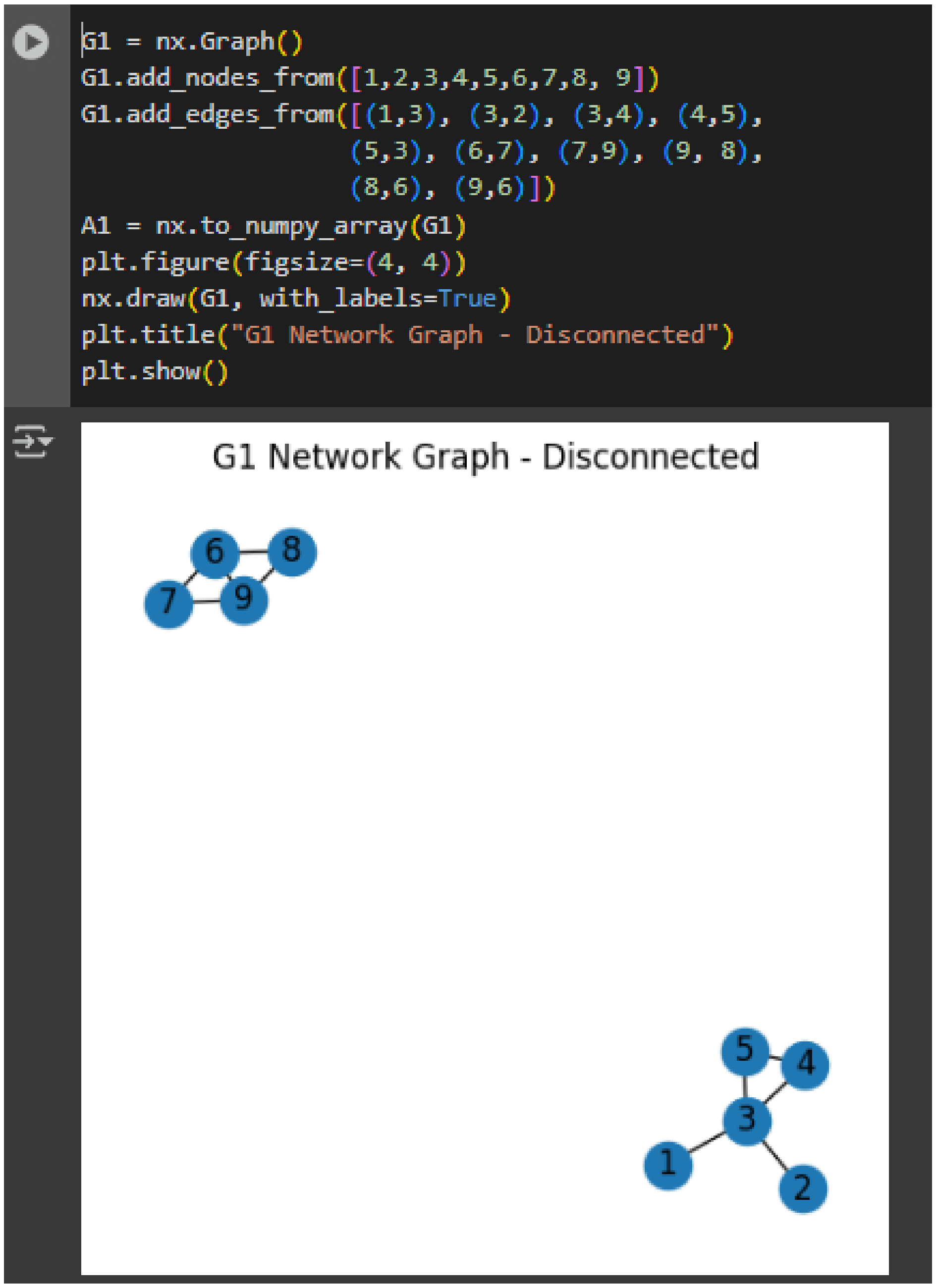
In class we have seen that for a simple undirected network the algebraic multiplicity of the eigenvalue 0 of the Laplacian is equal to the number of its components. For connected networks, the second smallest eigenvalue of the Laplacian is strictly greater than zero, and this eigenvalue also gives some information about the connectivity of the network. In this problem you will verify this via an example.

Consider the following network:

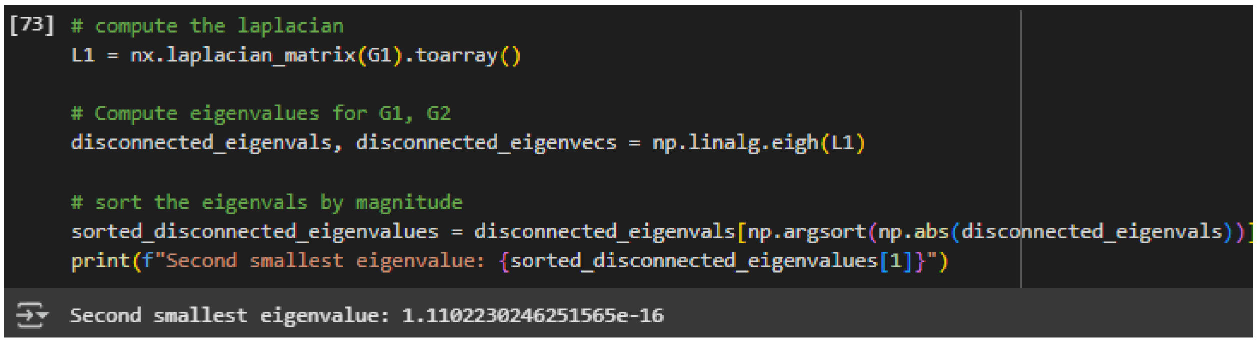


i. Compute the second smallest eigenvalue of its Laplacian. What is it?

    - "Compute" means "use a computer;" there's no need for you to try to do this by hand.



```python
G1 = nx.Graph()
G1.add_nodes_from([1,2,3,4,5,6,7,8, 9])
G1.add_edges_from([(1,3), (3,2), (3,4), (4,5),
                   (5,3), (6,7), (7,9), (9, 8),
                   (8,6), (9,6)])
A1 = nx.to_numpy_array(G1)
plt.figure(figsize=(4, 4))
nx.draw(G1, with_labels=True)
plt.title("G1 Network Graph - Disconnected")
plt.show()
```



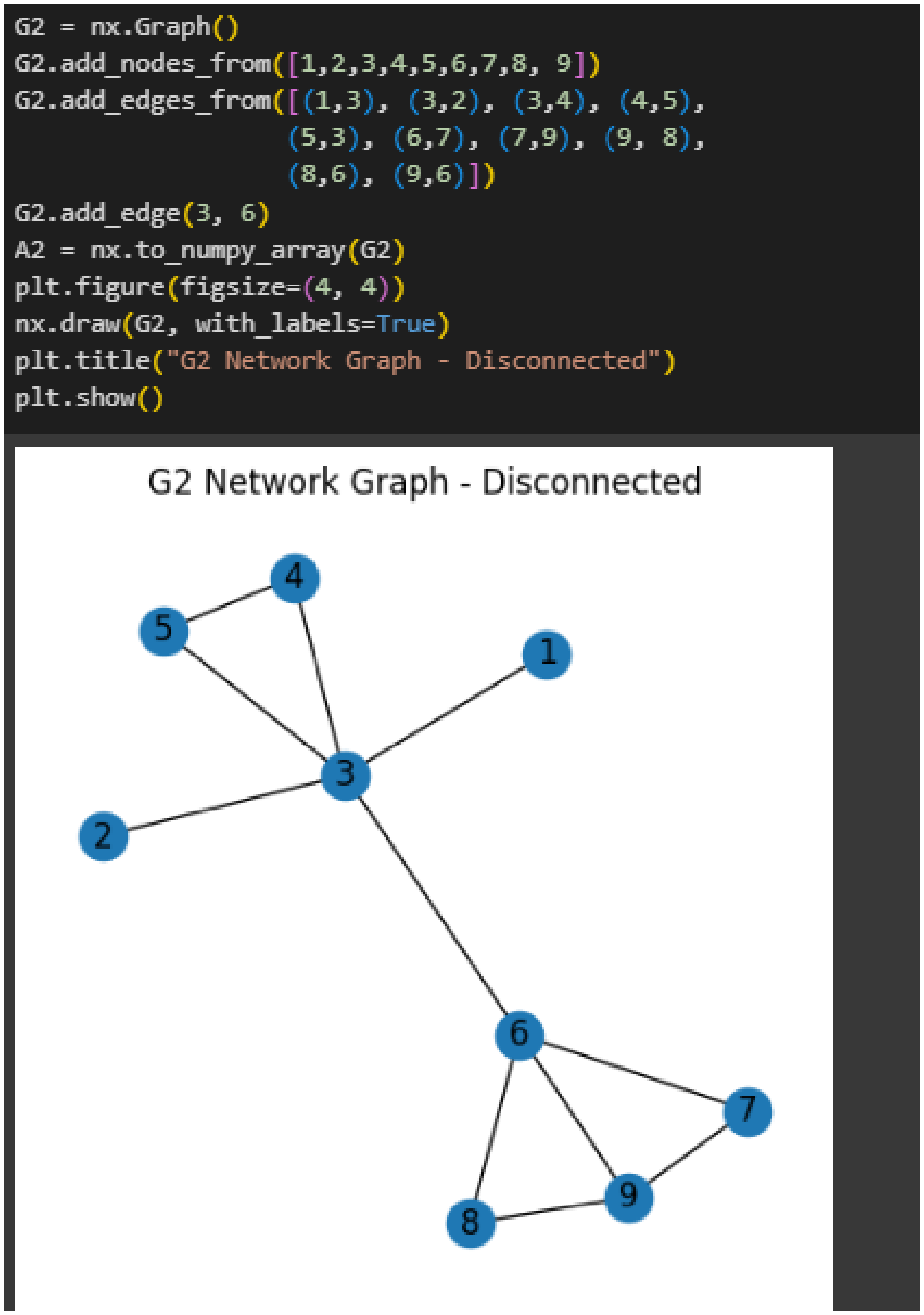
```python
[73] # compute the laplacian
     L1 = nx.laplacian_matrix(G1).toarray()

     # Compute eigenvalues for G1, G2
     disconnected_eigenvals, disconnected_eigenvecs = np.linalg.eigh(L1)

     # sort the eigenvals by magnitude
     sorted_disconnected_eigenvalues = disconnected_eigenvals[np.argsort(np.abs(disconnected_eigenvals))]
     print(f"Second smallest eigenvalue: {sorted_disconnected_eigenvalues[1]}")
```

```
Second smallest eigenvalue: 1.1102230246251565e-16
```

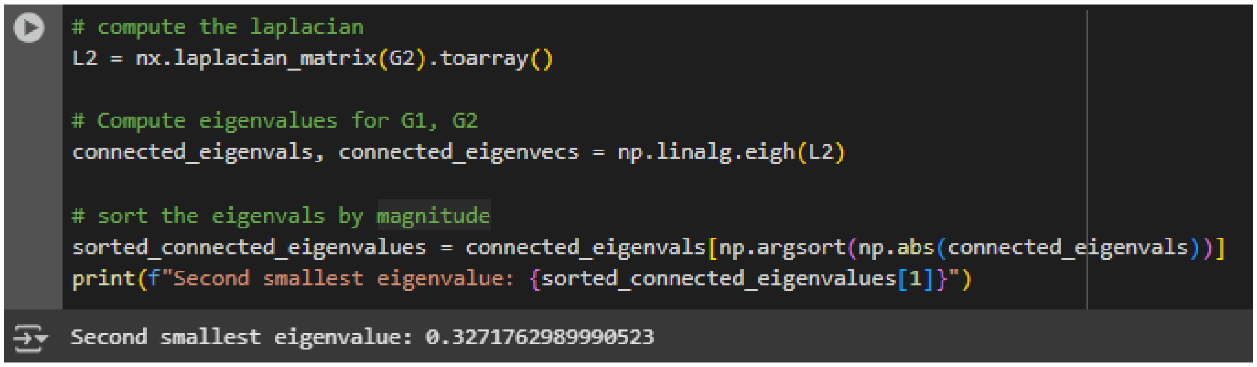
The second eigenvalue of the Laplacian for this graph $G1$ appears to be 0. The fact that the multiplicity of 0 eigenvalues is greater than one, indicates that the graph is disconnected into multiple components.

ii. Next, join two vertices in the two components of the network so that the network will have just one component. How does the value of the second smallest eigenvalue change, as you choose different vertices in the two components? Compute the value of the second smallest eigenvalue for three different choices of nodes.

Hint: the degree of the nodes plays a crucial role.



```
G2 = nx.Graph()
G2.add_nodes_from([1,2,3,4,5,6,7,8, 9])
G2.add_edges_from([(1,3), (3,2), (3,4), (4,5),
                   (5,3), (6,7), (7,9), (9, 8),
                   (8,6), (9,6)])
G2.add_edge(3, 6)
A2 = nx.to_numpy_array(G2)
plt.figure(figsize=(4, 4))
nx.draw(G2, with_labels=True)
plt.title("G2 Network Graph - Disconnected")
plt.show()
```

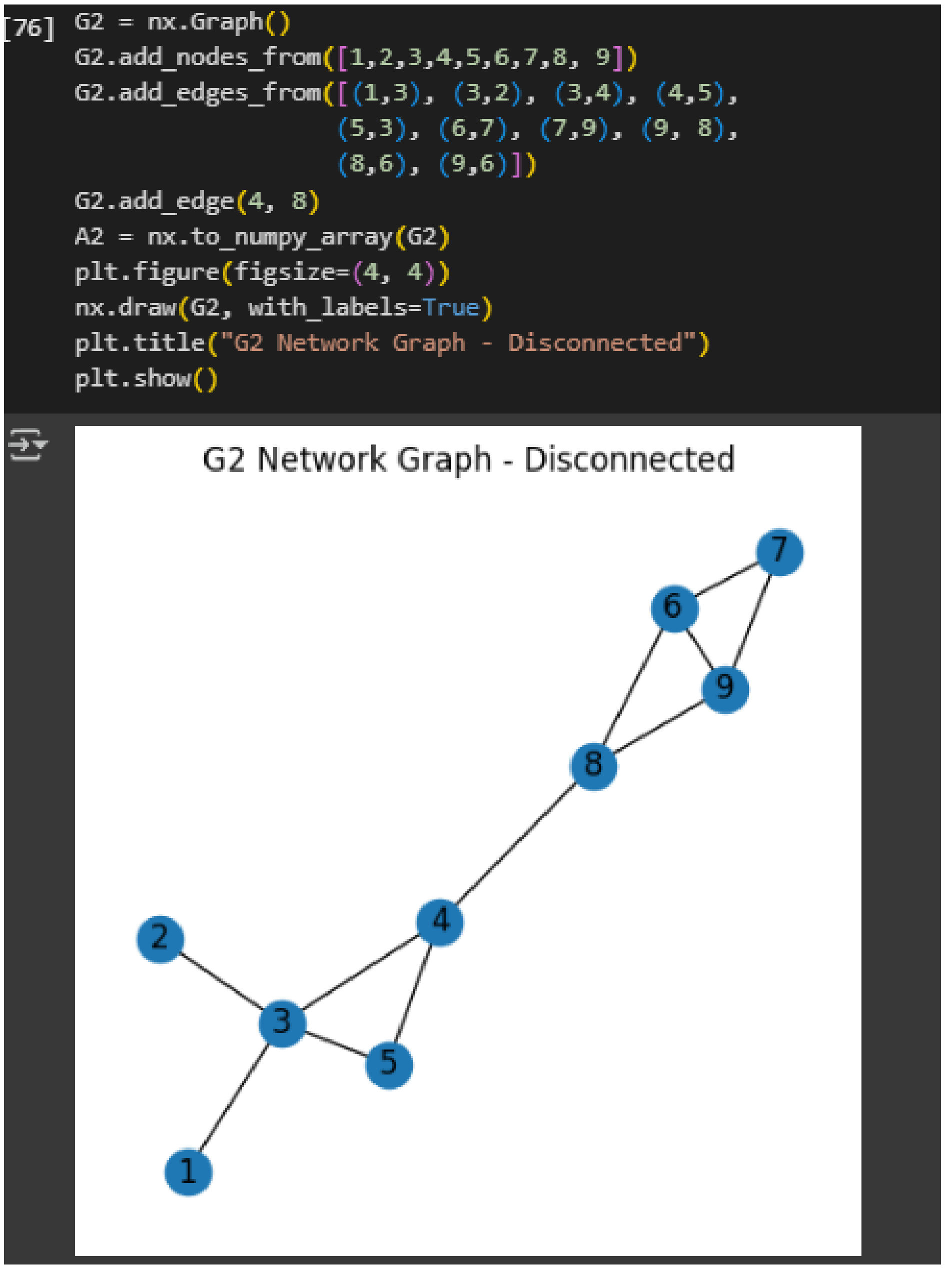


```
# compute the laplacian
L2 = nx.laplacian_matrix(G2).toarray()

# Compute eigenvalues for G1, G2
connected_eigenvals, connected_eigenvecs = np.linalg.eigh(L2)

# sort the eigenvals by magnitude
sorted_connected_eigenvalues = connected_eigenvals[np.argsort(np.abs(connected_eigenvals))]
print(f"Second smallest eigenvalue: {sorted_connected_eigenvalues[1]}")
```

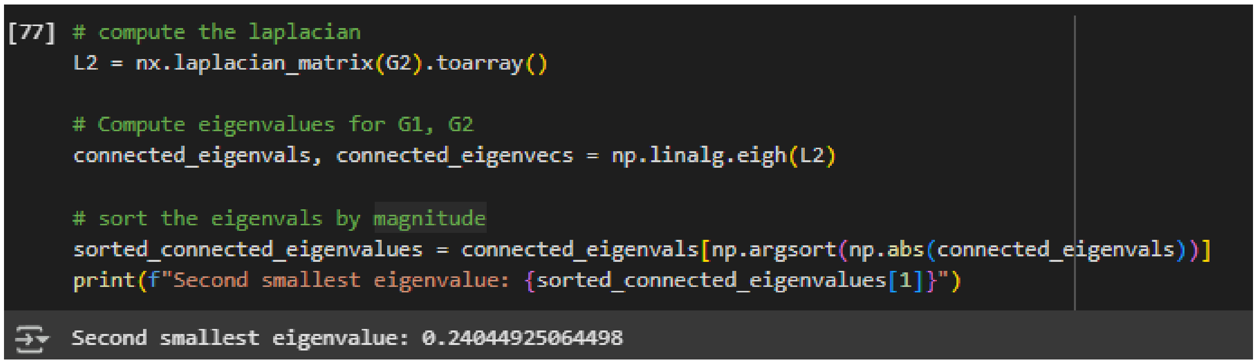
```
Second smallest eigenvalue: 0.3271762989990523
```

The Fiedler eigenvalue is now non-zero now that the graph $G2$ consists of a connected network. It appears that this second eigenvalue is fairly large, having connected node 3 which has a fairly high degree of 5 and node 6 with a degree of 4.

If I instead connect a node with slightly lower degree, 4 and 8:



```
[76] G2 = nx.Graph()
     G2.add_nodes_from([1,2,3,4,5,6,7,8, 9])
     G2.add_edges_from([(1,3), (3,2), (3,4), (4,5),
                        (5,3), (6,7), (7,9), (9, 8),
                        (8,6), (9,6)])
     G2.add_edge(4, 8)
     A2 = nx.to_numpy_array(G2)
     plt.figure(figsize=(4, 4))
     nx.draw(G2, with_labels=True)
     plt.title("G2 Network Graph - Disconnected")
     plt.show()
```

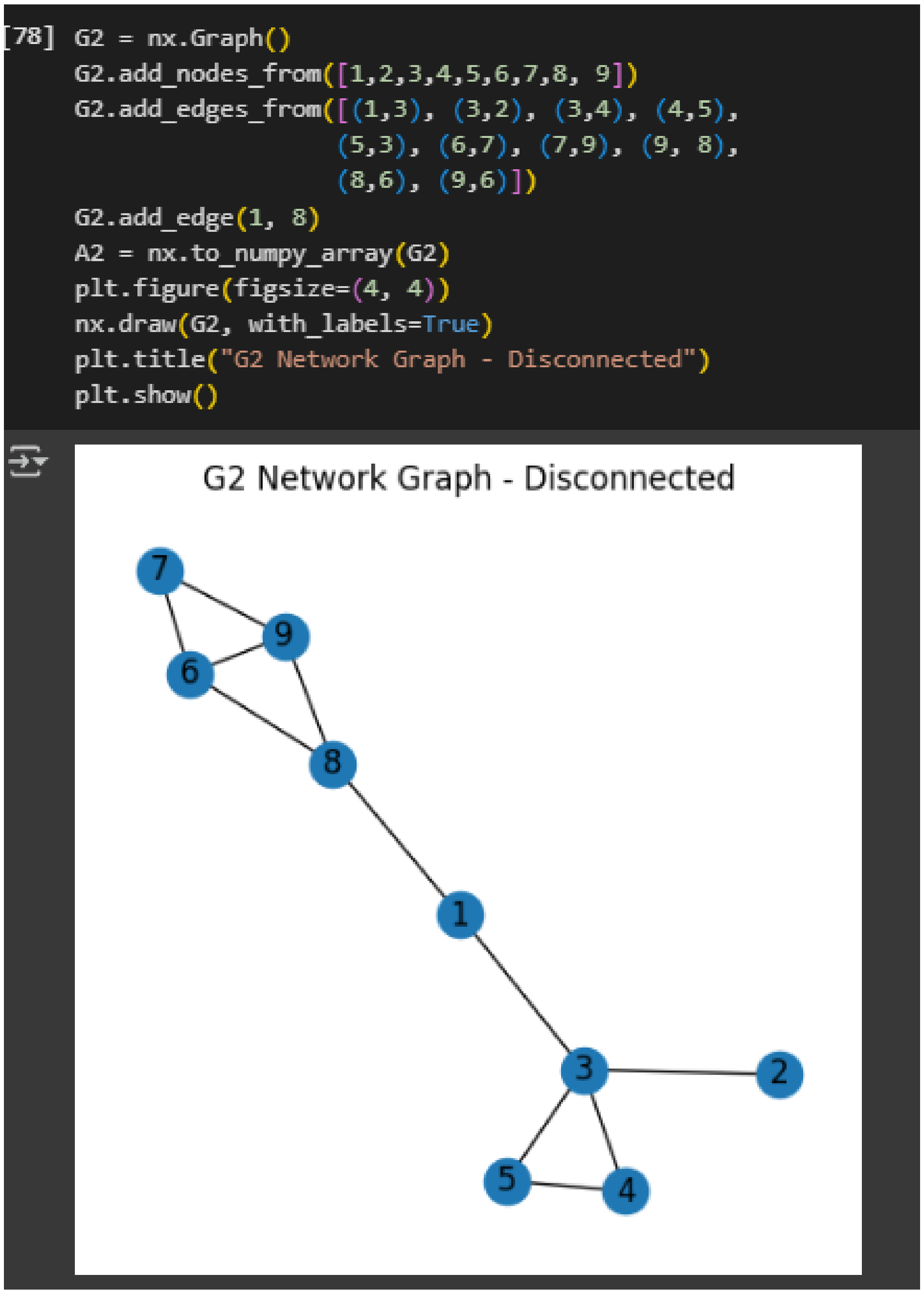


```
[77] # compute the laplacian
     L2 = nx.laplacian_matrix(G2).toarray()

     # Compute eigenvalues for G1, G2
     connected_eigenvals, connected_eigenvecs = np.linalg.eigh(L2)

     # sort the eigenvals by magnitude
     sorted_connected_eigenvalues = connected_eigenvals[np.argsort(np.abs(connected_eigenvals))]
     print(f"Second smallest eigenvalue: {sorted_connected_eigenvalues[1]}")
```

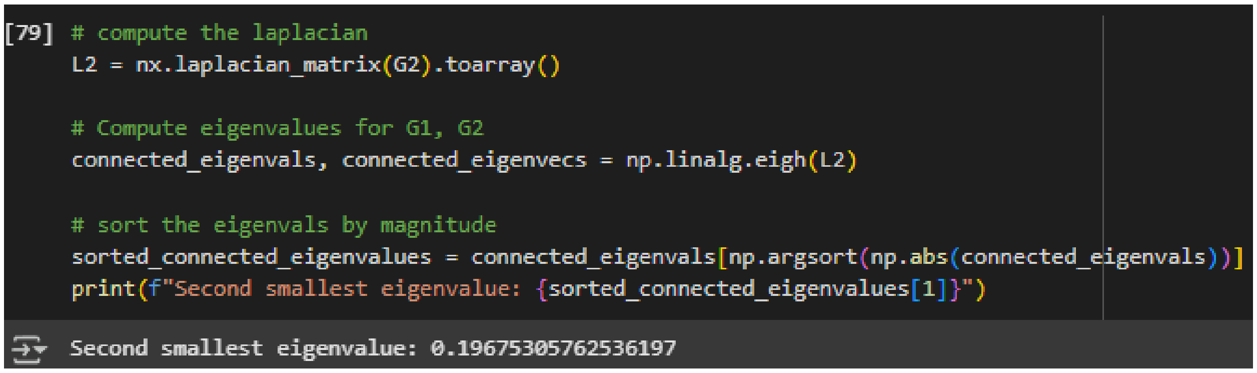
```
Second smallest eigenvalue: 0.24044925064498
```

I notice that the second smallest eigenvalue in magnitude has decreased as the nodes that were connected are lower in degree at either end.

If I choose nodes with an extremely low degree (1 and 8):



```
[78] G2 = nx.Graph()
     G2.add_nodes_from([1,2,3,4,5,6,7,8, 9])
     G2.add_edges_from([(1,3), (3,2), (3,4), (4,5),
                        (5,3), (6,7), (7,9), (9, 8),
                        (8,6), (9,6)])
     G2.add_edge(1, 8)
     A2 = nx.to_numpy_array(G2)
     plt.figure(figsize=(4, 4))
     nx.draw(G2, with_labels=True)
     plt.title("G2 Network Graph - Disconnected")
     plt.show()
```



```
[79] # compute the laplacian
     L2 = nx.laplacian_matrix(G2).toarray()

     # Compute eigenvalues for G1, G2
     connected_eigenvals, connected_eigenvecs = np.linalg.eigh(L2)

     # sort the eigenvals by magnitude
     sorted_connected_eigenvalues = connected_eigenvals[np.argsort(np.abs(connected_eigenvals))]
     print(f"Second smallest eigenvalue: {sorted_connected_eigenvalues[1]}")
```

```
Second smallest eigenvalue: 0.19675305762536197
```

This Fiedler value is even smaller. Thus there is a general trend that as the degree of the most weakly connected nodes decreases, so does this eigenvalue. Once these nodes are completely disconnected such that there exists two distinct components, this Fiedler value becomes 0.

Problem 2. (Node-Incidence Matrix and the Laplacian)

The *node-edge incidence matrix* of a graph with n nodes and m edges is a matrix $\mathbf{B} \in \mathbb{R}^{2m \times n}$. There are two rows of $\mathbf{B}$ for each edge e. If e links nodes j and ℓ, then there is a row for the $j \to \ell$ "direction" and a row for the $\ell \to j$ "direction". So, we can write an individual entry of $\mathbf{B}$ as $b_{(j\to\ell),i}$. These entries are given by:

$$b_{(j\to\ell),i} = \begin{cases} -1 & i = j \\ +1 & i = \ell \\ 0 & \text{otherwise.} \end{cases}$$

i. The Laplacian matrix $\mathbf{L}$ of a graph is defined in eq. (6.29) of Newman. Prove using direct matrix multiplication that $\mathbf{L}$ can be computed using one of the two formulae below (and figure out which one):

$$\mathbf{L} = \frac{1}{2}\mathbf{B}^T\mathbf{B} \quad \text{or} \quad \mathbf{L} = \frac{1}{2}\mathbf{B}\mathbf{B}^T \ .$$

Let us first figure out which of $L = \frac{1}{2}B^TB$ or $L = \frac{1}{2}BB^T$ is valid given the definition of B. We say that $B \in R^{2m\times n}$ where $G = (V, E), |V| = n, |E| = m$, thus $A \in R^{n\times n}$. B has shape $2m \times n$ and B^T has $n \times 2m$. Matrix multiplying BB^T gives a $2m \times 2m$ matrix, but we know the Laplacian matrix should have the same dimensions as A. B^TB, under our definition, gives us a $n \times n$ matrix, so for now we will try to prove the above with $L = \frac{1}{2}B^TB$.

Prove $L = \frac{1}{2}B^TB$: Let us rewrite for a given B^TB:

$$(B^TB)_{ij} = \sum_{k=1}^{2m} B^T_{ik}B_{kj}$$

We are given (I changed the notation slightly for clarity later on):

$$b_{(m\to\ell),i} = \begin{cases} -1 & i = m \\ +1 & i = \ell \\ 0 & \text{otherwise.} \end{cases}$$

Let us first choose the case where $i = j$ in $(B^TB)_{ij}$:

$$(B^TB)_{ii} = \sum_{k=1}^{2m} B^T_{ik}B_{ki}$$

We know that $B^T_{ik} = B_{ki}$, and we know that each edge $e \in E$ contributes two rows in B: the $m \to l$ direction and the $l \to m$ direction. We can reconsider this quantity in terms of e instead of a summation over all $2m$ rows of B:

$$= \sum_{e\in E} b_{(m\to l),i}b_{(m\to l),i} + b_{(l\to m),i}b_{(l\to m),i}$$

When $i = m$ for $b_{(m\to l),i}$ we have -1. And for $b_{(l\to m),i}$ we have 1 (i.e. i is the end of some edge $m \to l$). Thus, over each of the edges where i appears, we have $(-1)^2 + (1)^2 = 2$. If there is no edge to i, this counts for 0. In other words, this is $2 \times k_i$ where k_i denotes the degree of node i. It would follow that $1/2$ of this quantity would give the degree for $i = j$ (along the diagonal).

Now let us do the case $i \neq j$, the off-diagonal of B^TB:

We will again use:

$$(B^TB)_{ij} = \sum_{k=1}^{2m} B^T_{ik}B_{kj}$$

And rewrite it as:

$$= \sum_{e\in E} b_{(m\to l),i}b_{(m\to l),j} + b_{(l\to m),i}b_{(l\to m),j}$$

In other words, for each edge, the product of entries for nodes i and j in B where $i \neq j$ with each of the directions of the edge $m \to l$ and $l \to m$.

If there does not exist an edge m to l, the quantity $b_{(m\to l),i}b_{(m\to l),j} + b_{(l\to m),i}b_{(l\to m),j} = 0$, since for $b_{(m\to l),i} = 0$ where $i \neq m$ and $i \neq l$ (in other words, i is not at either end of this edge). The same applies to $b_{(l\to m),j}$.

Otherwise:

$$b_{(m\to l),i}b_{(m\to l),j} + b_{(l\to m),i}b_{(l\to m),j} = (-1) \times (1) + (1) \times (-1) = -2$$

Where the product of the entry for node i and row $m \to \ell = -1$ (that is, $i = m$ - an end of the edge), and the entry for node j and row $m \to \ell = +1$ (that is, $j = \ell$ - the other end of the edge), gives -1. Summed over both directions, we have -2. If we divide by our factor of $1/2$, we get -1.

Put together we have:

$$(\frac{1}{2}B^TB)_{ij} = \begin{cases} k_i & i = j \\ -1 & (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

This is just the definition of the Laplacian matrix if we look at it entry-wise, thus $L = \frac{1}{2}B^TB$.

I WOULD APPRECIATE FEEDBACK ON MY NOTATION FOR $i \neq j$ CASE. I continually made changes to the indexing notation for clarity in my different drafts of this proof, and I may have forgotten something/made a mistake.

ii. A matrix $\mathbf{M}$ is *positive-semidefinite* if it holds that $\mathbf{v}^T\mathbf{M}\mathbf{v} \geq 0$ for all vectors $\mathbf{v}$. Use your result from Part (a) to give a very short proof that $\mathbf{L}$ is a positive-semidefinite matrix.

Let us first rewrite the problem as $v^T L v = \frac{1}{2} v^T B^T B v$, applying the result of the previous proof. We can rewrite then as:

$$= \frac{1}{2}(v^T B^T)(Bv)$$

We know that Bv is a vector and $v^T B^T = (Bv)^T$ is the transpose of that vector.

$$= \frac{1}{2}(Bv \cdot Bv) = \frac{1}{2}\sum_i (Bv)_i^2$$

Any quantity squared, and then summed over for the entries of the vector must be strictly ≥ 0. In other words, $v^T L v \geq 0$ for all v, so positive-semidefinite.

Problem 3. (Laplacian and Graph Partitioning)

In section 6.14.1, Newman considers the role of the Laplacian $\mathbf{L}$ in partitioning or "cutting" graphs into groups. Let's focus on the two-group case. In eq. (6.37), Newman defines an objective function

$$R(\mathbf{s}) = \frac{1}{4}\mathbf{s}^T\mathbf{L}\mathbf{s}\ , \tag{1}$$

where $\mathbf{s} \in \mathbb{R}^n$ is the vector with entries

$$s_i = \begin{cases} +1 & \text{node } i \text{ is in group 1,} \\ -1 & \text{node } i \text{ is in group 2.} \end{cases} \tag{2}$$

The idea is that a choice of s corresponds to a choice of groups for the nodes, and that the function $R(\mathbf{s})$ measures the number of edges which are cut. Newman writes—somewhat uncarefully—that "... our goal is to find the vector s that minimizes the cut size (6.37) for given $\mathbf{L}$."

Assume throughout this problem that we are considering the Laplacian matrix $\mathbf{L}$ of a connected graph.

i. Find a vector s that minimizes $R(\mathbf{s})$.

   Let us take the gradient of $R(s)$ with respect to s:

$$\nabla_s R(s) = \nabla_s \frac{1}{4} s^T L s$$

$$= \frac{1}{4} \times 2Ls = \frac{1}{2}Ls$$

   We set the gradient to 0:

$$0 = \frac{1}{2}Ls$$
$$0 = Ls$$

   Since L is the Laplacian matrix, we know it must have eigenvector $\mathbf{1}$ with eigenvalue 0, and we also know that s is in the null space of L. We have s as some multiple of $\mathbf{1}$.

ii. Comment briefly (2-3 sentences is fine) on whether this vector is useful in the context of the graph partitioning problem.

   This is a completely useless vector in the context of the graph partitioning problem. We are effectively taking every node i in a graph and assigning it to the same group, and none to the other (just the original graph). We would probably actually want the the next s that minimizes $R(s)$.

Problem 4. (Properties of the Graph Laplacian)

Prove the following properties of the graph Laplacian that were given in the lecture notes. Let $\mathbf{L}$ be the combinatorial graph Laplacian for an undirected, unweighted network and $\mathbf{1}$ be the vector containing all ones.

Note: *Some of these properties may already have appeared in warmups or in class. It's ok to draw on those solutions; what I'm asking for here is a careful, rigorous presentation.*

i. $\mathbf{L}$ is real and symmetric.

L is real: Let $L = D - A$ where A is the undirected adjacency matrix of a graph G (implying $A_{ij} = A_{ji}$). D is the diagonal matrix of whose entries are the degrees of the given nodes.

By definition A is defined over the reals: $A \in R^{n \times n}$. We can not have negative edges or non-integer number of edges between nodes.

Additionally, the only non-zero entries of D are diagonal entries corresponding to degrees. Since a graph G can't have negative or non-integer number of neighbors, it must also be real.

The difference between two real matrices $D - A$ must also produce a matrix that is real.

L is symmetric: The degree matrix D is diagonal, meaning $D^T = D$.

Because A is undirected, $A_{ij} = A_{ji}$. This means an edge between nodes i and j is the same as an edge between nodes j and i. Thus, $A^T = A$.

Together, $L^T = D^T - A^T$, substituting in D and A gives us:

$$L^T = D^T - A^T$$

$$L^T = D - A$$

$$L^T = L$$

That is, L is symmetric.

ii. $\mathbf{L1} = \mathbf{0}$. That is, every row sums to 0.

We can substitute into the definition $L = D - A$:

$$L\mathbf{1} = (D - A)\mathbf{1}$$

$$= D\mathbf{1} - A\mathbf{1}$$

We can then rewrite $D\mathbf{1}$:

$(D\mathbf{1})_i = \sum_{i=1}^{n} D_{ij} \cdot 1 = d_i = [d_1, ..., d_n]^T$

That is, we have the summation over a given column of D. Since D is diagonal (only one non-zero entry per column or row), that entry of the vector $D\mathbf{1}_i$ is just the sole entry.

We can then rewrite $A\mathbf{1}$:

$$(A\mathbf{1})_i = \sum_{i=1}^{n} A_{ij} \cdot 1 = d_i = [d_1, ..., d_n]^T$$

That is, for each row of A, we sum up the instances of edges from node i, which can be understand as the degree. We end up with a vector of degrees.

$$L\mathbf{1} = [d_1, ..., d_n]^T - [d_1, ..., d_n]^T = 0$$

Trivially, for the vector $\mathbf{1}$, we have an eigenvalue 0 for the Laplacian matrix L.

iii. $\mathbf{L}$ is not invertible.

A matrix M is said to be invertible if its determinant is non-zero by the Inverse Matrix Theorem. Given that $\mathbf{1}$ is an eigenvector of the Laplacian matrix with eigenvalue 0 as we have shown in the previous proof, we can rewrite the determinant as the product of its eigenvalues:

$$\det(L) = 0 \times \lambda_2 \times ... \times \lambda_n$$

Since the product of anything times zero is zero, $\det(L) = 0$. Based upon the Inverse Matrix Theorem, L is not invertible.

iv. The eigenvalues of $\mathbf{L}$ are real and nonnegative. *Hint: Use the fact that* $\mathbf{L}$ *is positive-semidefinite.*

We have alread proven the fact that the L Laplacian matrix is positive-semidefinite. Let us express eigenvalues of L:

$$Lv = \lambda v$$

We can rewrite this as:

$$v^T L v = v^T \lambda v$$

Since we know that $v^T L v \geq 0$, this implies that $v^T \lambda v \geq 0$. Since $v^T v$ is non-negative, $\lambda \geq 0$ for this to hold.

To show that λ is real, we can use the Spectral Theorem: a symmetric matrix M with real entries is diagonizable where $M = U^{-1} D U$, such that D contains the eigenvalues of M along the diagonal, and U contains eigenvectors of M. By extension of the Spectral Theorem, eigenvalues of M are real, since the eigenvalues of M are found by $0 = \det(M - \lambda I)$; the solutions to the characteristic equation are real.

Thus, since L is symmetric, eigenvalues of L must also be real. In sum, L must have non-negative and real eigenvalues.

Problem 5. (Connected Components and the Graph Laplacian)

Prove that a graph has c connected components if and only if its graph Laplacian has exactly c zero eigenvalues (that is, the eigenvalue $\lambda = 0$ has algebraic multiplicity c.) Throughout this problem, you may use without proof a theorem from linear algebra which says that, for a symmetric matrix, the algebraic and geometric multiplicities of all eigenvalues are equal.

Here's the approach we recommend.

i. First, show that the algebraic multiplicity of the zero eigenvalue in a connected graph is exactly 1.

   - It may be helpful to consider the function $S(\mathbf{x}) = \frac{1}{2}\sum_{i,j} A_{ij}(x_i - x_j)^2 = \mathbf{x}^T\mathbf{L}\mathbf{x}$.

   First, suppose by contradiction that the connected graph G has a Laplacian L such that it's multiplicity of 0 eigenvalues is greater than 1. There must exist at least two corresponding eigenvectors x and v such that they are linearly independent with eigenvalue 0.

   Note that with $\frac{1}{2}\sum_{i,j} A_{ij}(x_i - x_j)^2$, this sum for $S(v)$ and $S(x)$ is 0 only when $x_i = x_j$ and $v_i = v_j$ for all i, j . Consequently, if we were to mutate even a single entry of x or v, where c is some constant:

   $$S(x) = x^T L x > 0 \text{ where } x \neq c\mathbf{1}$$
   $$S(v) = v^T L v > 0 \text{ where } v \neq c\mathbf{1}$$

   By extension, x and v are implied to be multiples of the constant vector if they are to satisfy $S(x) = 0$. This is a contradiction, as x and v are supposed to be linearly independent as eigenvectors with corresponding eigenvalue 0. Therefore, the multiplicity of eigenvalue 0 of L is exactly 1 in the connected graph.

ii. Next, argue that it is possible to label the nodes so that $\mathbf{L}$ is a block-diagonal matrix.

   Now, let G be the graph that is disconnected that consists of components G_1 through G_n. For a given component G_i, we can treat its nodes and edges as its own distinct adjacency matrix A_i as if it were its own graph. We can then think of the overall adjacency matrix A as a combination of adjacency matrices A_1 through A_c, where c is the number of components.

   By the definition of a disconnected graph, given that there are no edges from a given component to another, we would not expect there to be any node in A_i that connects to another node in A_j. Thus, in the larger adjacency matrix A, any given A_i and A_j should never occupy or overlap in the same row or column. For the adjacency matrix A, where c is the number of components, we would be able to arrange it as:

   $$\mathbf{A} = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_c \end{bmatrix}$$

Each adjacency matrix A_i would have its own corresponding Laplacian matrix $L_i = D_i - A_i$. This follows from the fact that we can calculate the degrees of each node in each individual component G_i from their corresponding adjacency matrix A_i.

Since the matrix D_i is itself a diagonal matrix, the diagonal matrix D of diagonal matrices D_i along the diagonal must also be diagonal. We have also just shown that the larger adjacency matrix A is diagonal, thus $L = D - A$ must also be a block diagonal matrix by extension, as it is the difference of two block diagonal matrices.

iii. Finally, use the block-diagonal structure of $\mathbf{L}$ to show that the algebraic multiplicity of $\lambda = 0$ is exactly c.

Given that L is the block diagonal matrix containing the Laplacian matrices L_i for each G_i, given c components, we can write it as:

$$\mathbf{L} = \begin{bmatrix} L_1 & 0 & \cdots & 0 \\ 0 & L_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & L_c \end{bmatrix}$$

We have previously just proven that the connected graph G_i has an eigenvalue for L_i of 0 with multiplicity of no more than 1, and additionally that given $L\mathbf{1} = 0$, 0 is always an eigenvalue of L. Thus, given that for connected G_i, L_i has exactly one eigenvalue 0 of multiplicity 1, and that there are c Laplacian matrices L_i (one for each distinct connected component G_i), there must also be exactly c total eigenvalues of 0 in L. In other words, the multiplicity of 0 eigenvalues in the Laplacian matrix corresponds to the number of distinct components.

This works!

We have just shown that it is possible to label the nodes such that L is a block-diagonal matrix consisting of L_i through L_c. By the Block Diagonal Matrix Theorem, the block diagonal matrix T's determinant is the product of the determinants of its diagonal blocks B_i. By extension, the eigenvalues of a block diagonal matrix T are the union over eigenvalues of each block B_i in T. We have previously just proven that a given connected graph G_i has an eigenvalue for L_i of 0 with multiplicity of exactly 1. Applying the Block Diagonal Matrix Theorem, if there is exactly one eigenvalue 0 for each L_i with multiplicity c, the union over all 0 eigenvalues of L_i must also have multiplicity c. Thus, a graph has only c connected components if and only if its graph Laplacian has exactly c 0 eigenvalues.

Problem 6. (Data Analysis with Laplacian Eigenvectors)

Several parts of this problem can be appropriately solved by lightly adapting code from relevant lecture notes.

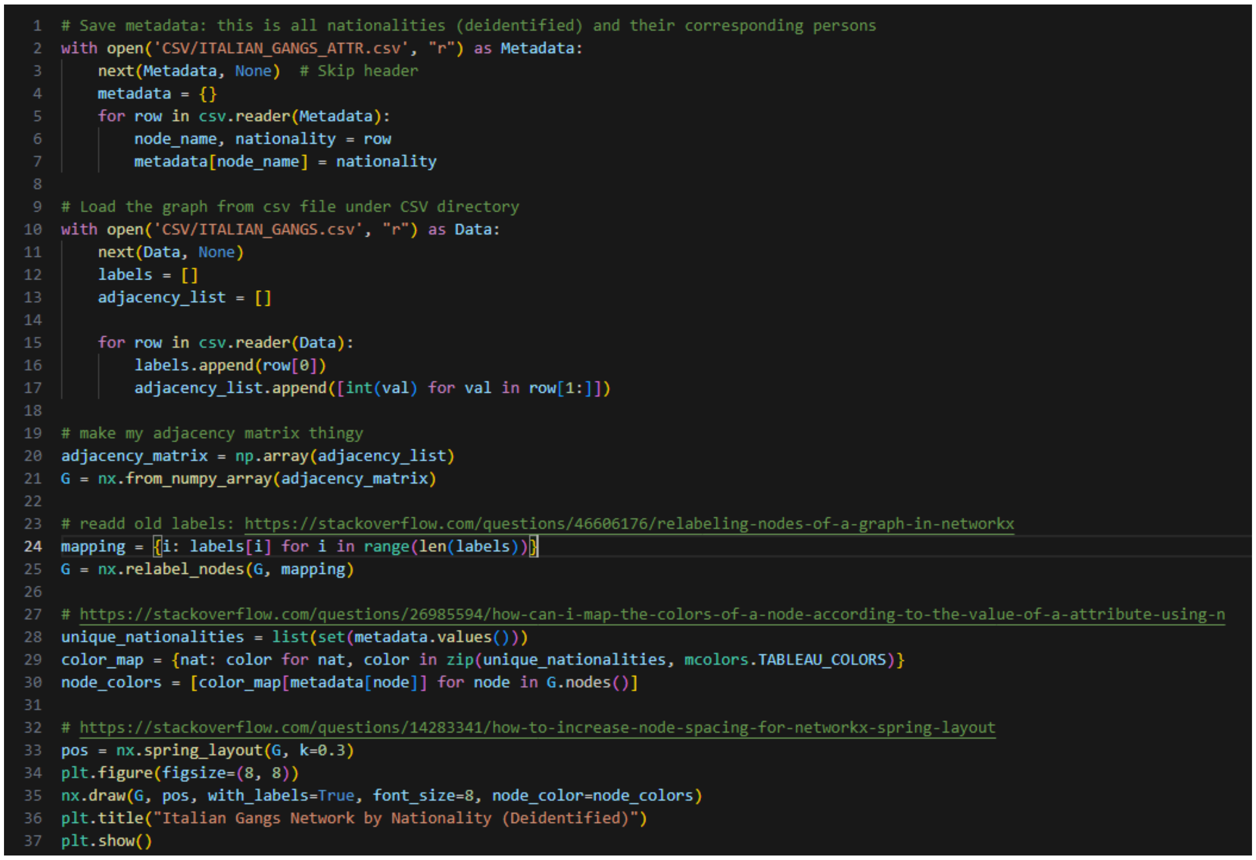
Part (a)

First, obtain an undirected network data set in which the nodes possess *metadata labels* of some type. For example:

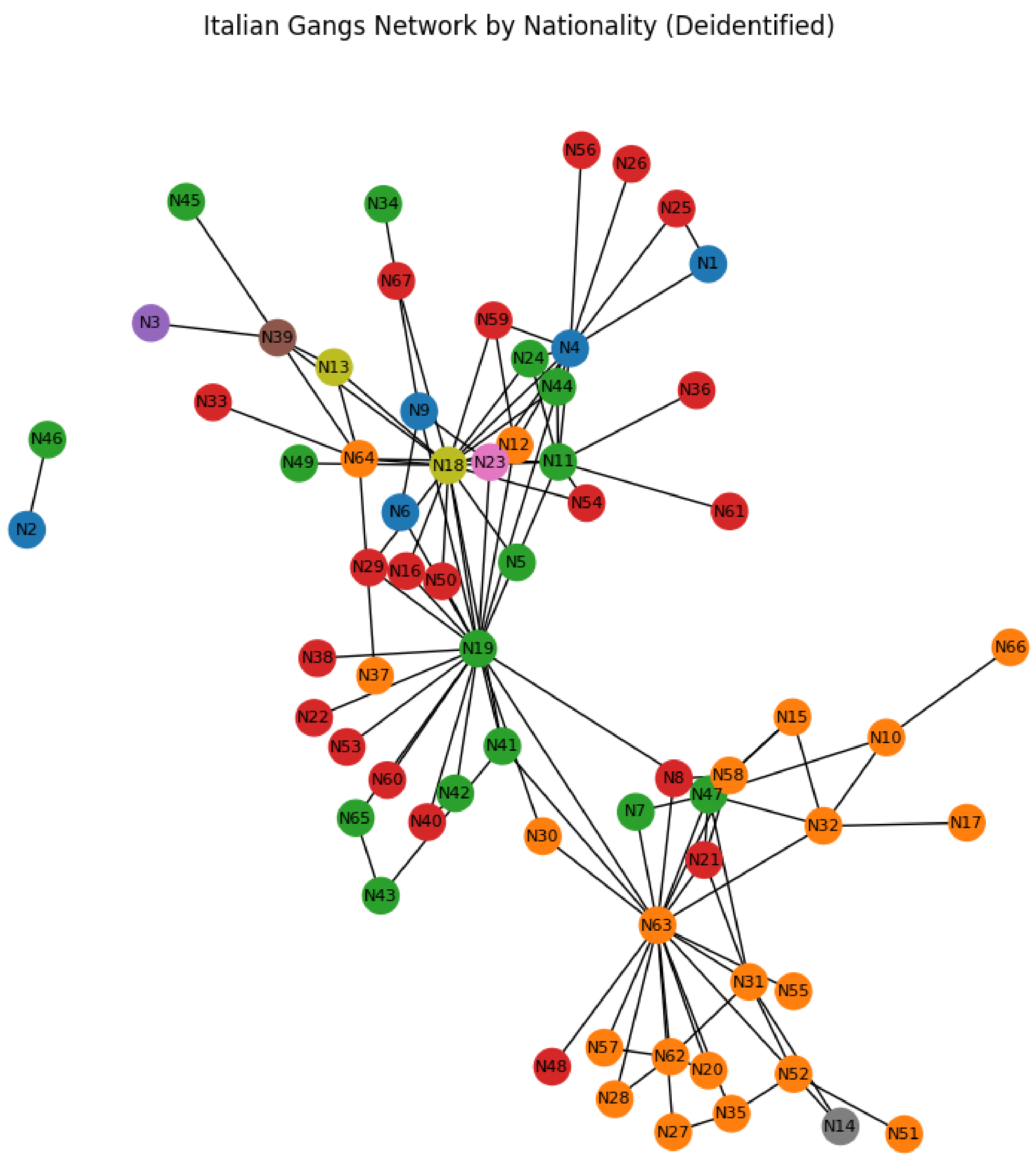
- Gender, race, age, or other demographic in a social network.
- Political party, ideology, or other political affiliation in a network of political actors.

Our favorite source of data sets is the Colorado Index of Complex Networks (ICON), operated by Aaron Clauset's group at the University of Colorado Boulder: https://icon.colorado.edu/. You can filter the index to include only graph data sets that include metadata. Download the network and read it into Python using NetworkX. You'll also need to read in the metadata.

You probably want a network with at most a few hundred nodes.

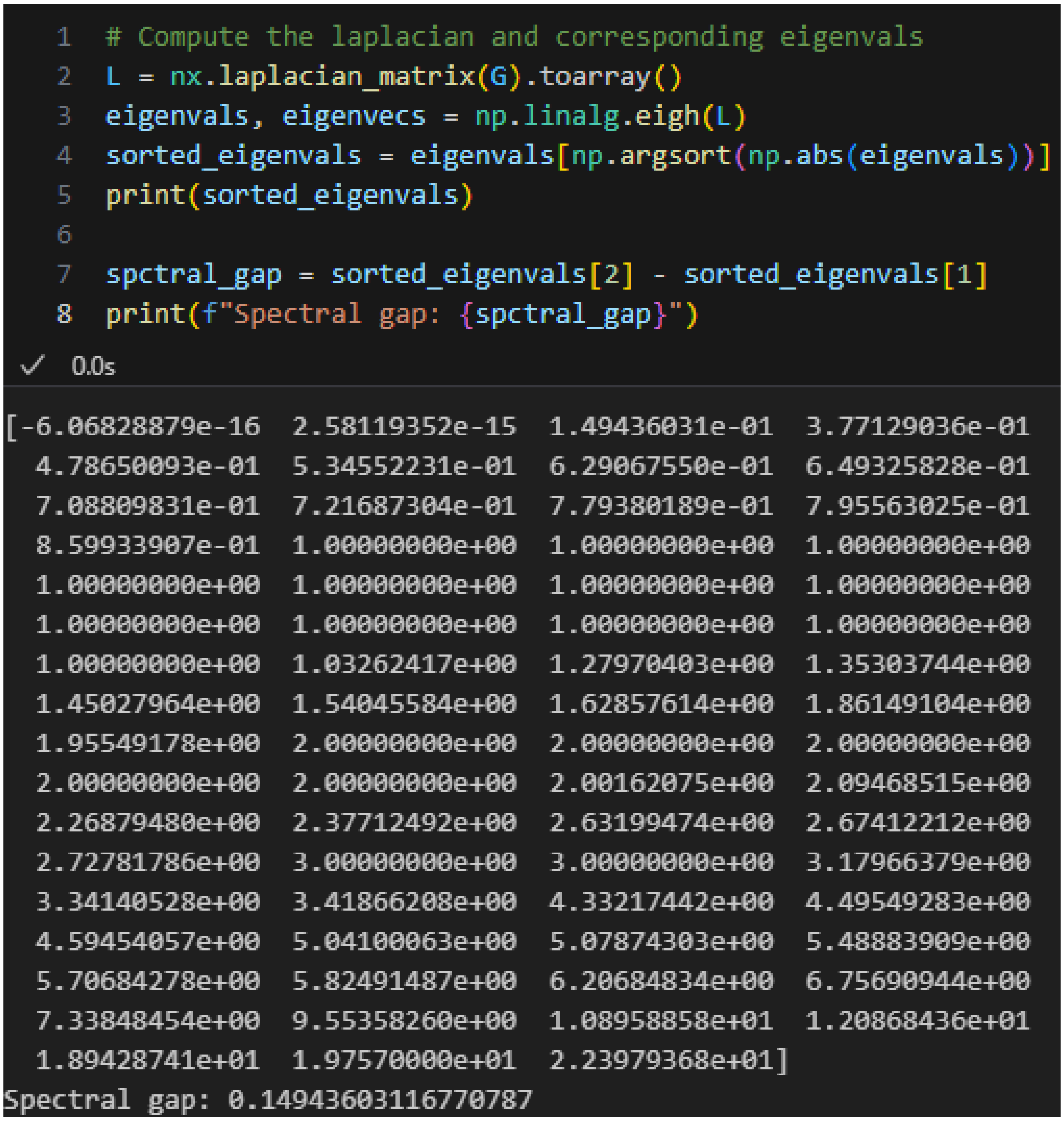


```python
# Save metadata: this is all nationalities (deidentified) and their corresponding persons
with open('CSV/ITALIAN_GANGS_ATTR.csv', "r") as Metadata:
    next(Metadata, None)  # Skip header
    metadata = {}
    for row in csv.reader(Metadata):
        node_name, nationality = row
        metadata[node_name] = nationality

# Load the graph from csv file under CSV directory
with open('CSV/ITALIAN_GANGS.csv', "r") as Data:
    next(Data, None)
    labels = []
    adjacency_list = []

    for row in csv.reader(Data):
        labels.append(row[0])
        adjacency_list.append([int(val) for val in row[1:]])

# make my adjacency matrix thingy
adjacency_matrix = np.array(adjacency_list)
G = nx.from_numpy_array(adjacency_matrix)

# readd old labels: https://stackoverflow.com/questions/46606176/relabeling-nodes-of-a-graph-in-networkx
mapping = {i: labels[i] for i in range(len(labels))}
G = nx.relabel_nodes(G, mapping)

# https://stackoverflow.com/questions/26985594/how-can-i-map-the-colors-of-a-node-according-to-the-value-of-a-attribute-using-n
unique_nationalities = list(set(metadata.values()))
color_map = {nat: color for nat, color in zip(unique_nationalities, mcolors.TABLEAU_COLORS)}
node_colors = [color_map[metadata[node]] for node in G.nodes()]

# https://stackoverflow.com/questions/14283341/how-to-increase-node-spacing-for-networkx-spring-layout
pos = nx.spring_layout(G, k=0.3)
plt.figure(figsize=(8, 8))
nx.draw(G, pos, with_labels=True, font_size=8, node_color=node_colors)
plt.title("Italian Gangs Network by Nationality (Deidentified)")
plt.show()
```



Part (b)

Visualize the small eigenvalues of the graph Laplacian; it's ok to show only the smallest 10 or 20. The *spectral gap* is the gap between the 2nd-smallest eigenvalue and third-smallest eigenvalue of the Laplacian. Would you say that your data shows a clear spectral gap?



```python
# Compute the laplacian and corresponding eigenvals
L = nx.laplacian_matrix(G).toarray()
eigenvals, eigenvecs = np.linalg.eigh(L)
sorted_eigenvals = eigenvals[np.argsort(np.abs(eigenvals))]
print(sorted_eigenvals)

spctral_gap = sorted_eigenvals[2] - sorted_eigenvals[1]
print(f"Spectral gap: {spctral_gap}")
```

```
✓ 0.0s
```

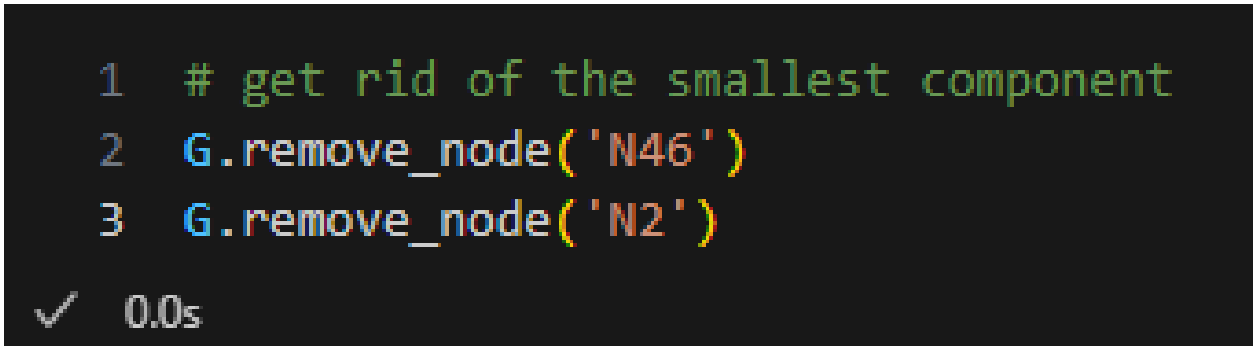
```
[-6.06828879e-16  2.58119352e-15  1.49436031e-01  3.77129036e-01
  4.78650093e-01  5.34552231e-01  6.29067550e-01  6.49325828e-01
  7.08809831e-01  7.21687304e-01  7.79380189e-01  7.95563025e-01
  8.59933907e-01  1.00000000e+00  1.00000000e+00  1.00000000e+00
  1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
  1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
  1.00000000e+00  1.03262417e+00  1.27970403e+00  1.35303744e+00
  1.45027964e+00  1.54045584e+00  1.62857614e+00  1.86149104e+00
  1.95549178e+00  2.00000000e+00  2.00000000e+00  2.00000000e+00
  2.00000000e+00  2.00000000e+00  2.00162075e+00  2.09468515e+00
  2.26879480e+00  2.37712492e+00  2.63199474e+00  2.67412212e+00
  2.72781786e+00  3.00000000e+00  3.00000000e+00  3.17966379e+00
  3.34140528e+00  3.41866208e+00  4.33217442e+00  4.49549283e+00
  4.59454057e+00  5.04100063e+00  5.07874303e+00  5.48883909e+00
  5.70684278e+00  5.82491487e+00  6.20684834e+00  6.75690944e+00
  7.33848454e+00  9.55358260e+00  1.08958858e+01  1.20868436e+01
  1.89428741e+01  1.97570000e+01  2.23979368e+01]
Spectral gap: 0.14943603116770787
```

I would argue that there is a fairly substantial spectral gap between the 2nd and 3rd smallest eigenvalues by magnitude. This makes sense, given the visual of the network clearly depicts two distinct components, thus we expect multiplicity of 2 for the 0 eigenvalue (which we do observe as the first two of the "sorted_eigenvals"). The difference (spectral gap) between the Fiedler 0 eigenvalue and the next eigenvalue on a 2-component network is bound to be larger.

Part (c)

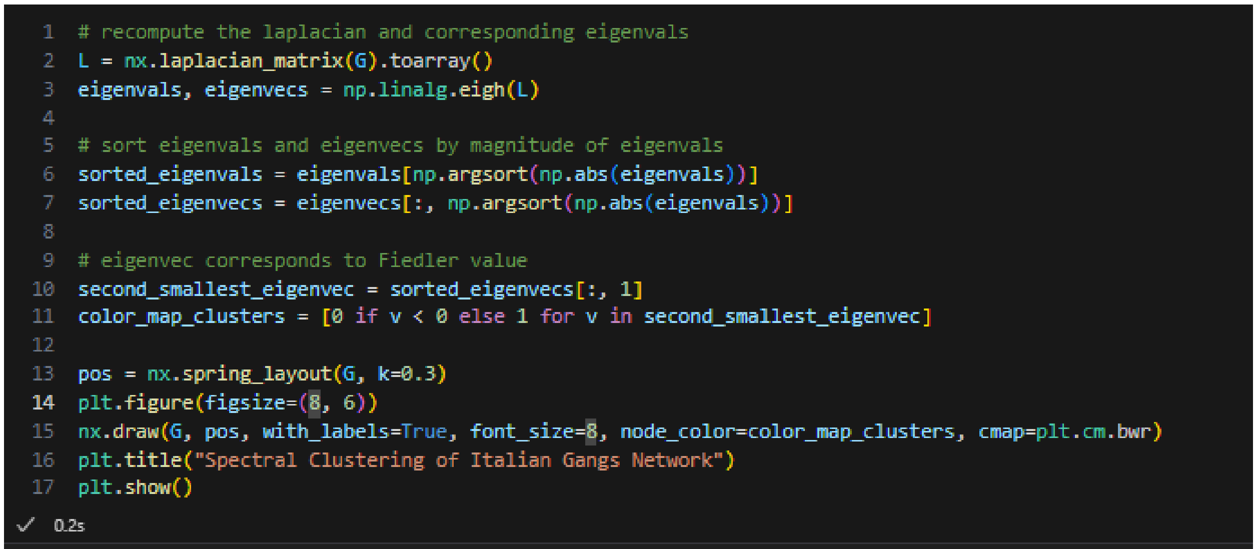
Compute the Fiedler eigenvector for the data and use it to cluster the nodes into two groups. Then, use any method to compare the values of the Fiedler eigenvector to the metadata labels from the data. Do these appear to be correlated?

To make things more interesting, I will get rid of the smallest component.



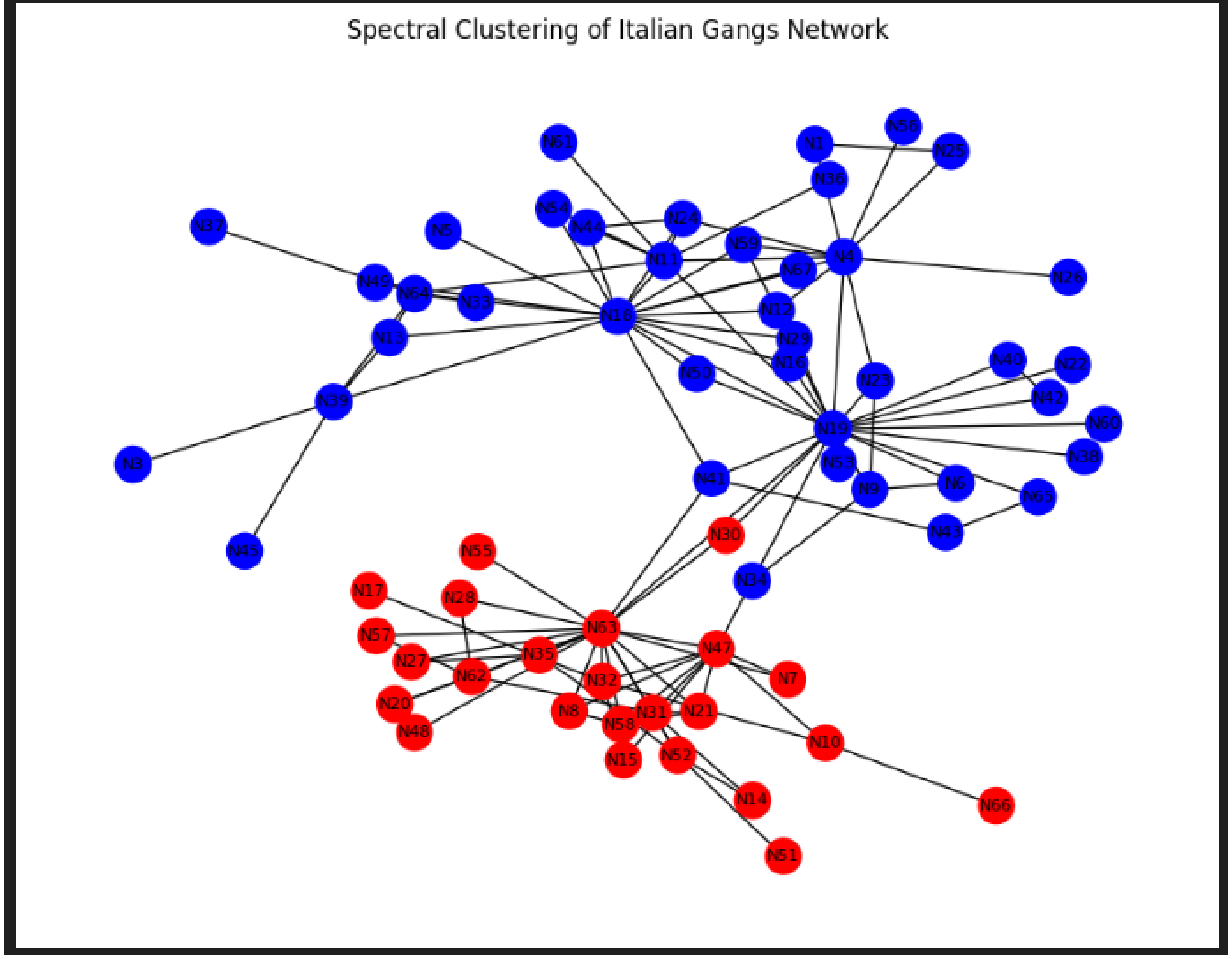
```
# get rid of the smallest component
G.remove_node('N46')
G.remove_node('N2')
```

```
✓ 0.0s
```



```python
# recompute the laplacian and corresponding eigenvals
L = nx.laplacian_matrix(G).toarray()
eigenvals, eigenvecs = np.linalg.eigh(L)

# sort eigenvals and eigenvecs by magnitude of eigenvals
sorted_eigenvals = eigenvals[np.argsort(np.abs(eigenvals))]
sorted_eigenvecs = eigenvecs[:, np.argsort(np.abs(eigenvals))]

# eigenvec corresponds to Fiedler value
second_smallest_eigenvec = sorted_eigenvecs[:, 1]
color_map_clusters = [0 if v < 0 else 1 for v in second_smallest_eigenvec]

pos = nx.spring_layout(G, k=0.3)
plt.figure(figsize=(8, 6))
nx.draw(G, pos, with_labels=True, font_size=8, node_color=color_map_clusters, cmap=plt.cm.bwr)
plt.title("Spectral Clustering of Italian Gangs Network")
plt.show()
```

✓ 0.2s



I was unable to superimpose the nationality and clustering labels simultaneously, but if you inspect the individual labels, you will notice that there is substantial overlap between the "orange" nationalities (of which I can not possibly know for certain with de-identified data) and the red cluster. Many members of the red cluster (i.e. N55, N57, N17, N28, N62, etc...) are also of the same ”orange” nationality. The other (blue) cluster seems to predominantly consist of ”red” and ”green” nationalities. I would definitely say that the metadata labels and Fiedler eigenvector are likely correlated.

Problem 7. (Exploring Network Literature, 2 points)

This problem has two main aims:

- To help you practice producing mathematical documents using the LaTeX typesetting language.
- To inspire you to discover networks that interest you and what kinds of questions you might wish to investigate about them.

Prompt

Find a *scholarly source*, such as a published journal article, book, or preprint, that discusses a network that interests you. Any network is fine — social, biological, information, physical, etc. — but it must be described in a scholarly source ***other*** **than Newman's book**. Cite this source and address the following questions:

- Where did this network come from? How was the data for it collected? What is one way in which the data collection methodology could potentially have been flawed, relative to the intentions of the researchers who collected the data?
- What is the main question that the researchers asked about the network? What did they do to answer this question? Describe at least one of their analyses using some mathematical symbols and at least one display equation (i.e. a centered equation on its own line). It's fine to replicate symbols or equations from the source, with appropriate acknowledgment.
- What are two questions that *you* have that you feel would be interesting questions for future work?

Requirements

This problem will be graded out of 2 points, with partial credit possible. An essay that meets the following requirements is very likely to receive full credit. An essay that is missing one or more of the requirements is very likely to not receive full credit.

i. Your essay should include discussions of each of the questions above.

ii. Your essay should be written in LaTeX.

iii. Your essay should include multiple mathematical symbols and at least one display equation (like this one):

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{3}$$

iv. Your essay should be no longer than 1,000 words.

v. Your essay should be organized into paragraphs, each of which discusses a separate and clearly indicated set of topics. A valid and "safe" way to organize your essay is to have one paragraph corresponding to each of the three bullet points.

vi. Your essay should be written in clear, scholarly prose. Errors related to spelling and grammar are not an issue provided that your meaning is clear.

vii. Your essay should include at least one **graphic with a caption** that is relevant to the network. This could be a graphic from the paper, an image that helps the reader interpret the meaning of nodes or edges, or any other relevant graphic. Your caption must be original. Make sure to briefly discuss this graphic in your written essay.

0.1 Reading Summary: 10 - Covid-19 pandemic model: a graph theoretical perspective

In this paper, authors Osaye and Alochukwu investigated graph theoretical approaches to modeling the Covid-19 pandemic. More generally, the authors ask how networks can be applied to address flaws in compartmental models, which make large assumptions about the dynamics of disease spread. While the authors discussed the foundations of a pandemic network simulation, they did not empirically test or validate on any actual data, and thus avoided specifying any data collection methodology altogether. Nevertheless, the authors propose a model that incorporates aspects of both compartmental models (specifically, SEIRD) and contact networks to better simulate more complex disease dynamics, rather than assuming equal likelihood of interactions among people.

To begin, the authors looked at prior compartmental models such the Susceptible-Exposed-Infected-Recovered (SEIR) model.

✓

Equation	**Purpose**
$\frac{dS}{dt} = S_{\text{rec}} - \lambda S$	Change in susceptible individuals
$\frac{dE}{dt} = E_{\text{rec}} + \lambda S - kE$	Change in exposed individuals
$\frac{dI}{dt} = kE - \alpha I$	Change in infected individuals
$\frac{dR}{dt} = \alpha I$	Change in recovered individuals

Table 1: The SEIR model describes a dynamic model where individuals move between different "categories." For example, the given infected at time t can be modeled as the progression rate k of exposures E minus the recovery rate α among infected I. This figure was sourced from Osaye & Alochukwu.

In turn, certain variables such as λ vary as a function of t and the consequent time that lockdown occurs, the rate of infection, and rate of social distancing. The major drawback of many of these models, however, is that they make major assumptions, including that mixing/contacts are homogeneous, among others.

The authors additionally described comparisons of compartmental models against contact networks for severe acute respiratory syndrome (SARS). They start by exploring some of the useful insights provided by using network-based simulations of disease dynamics. As Osaye explained, the reproduction number (a quantity often used to gauge the expected cases generated from a single infection), could be modeled as $R_0 = t(\frac{<k^2>}{<k>} - 1)$ in the contact network, where t represents the transmissibility and k, the degree of a person, with $< k >$ as the mean degree.

Moreover, the authors outlined several additional expressions using the adjacency matrix A that describes contact among individuals, where each individual is a node i. First is the following: $\bar{n} = (1/N)\sum_{ij} A_{ij} = \frac{1}{N} trace(A^2)$. This quantity allows us to understand the average number of contacts for a given individual. That is, $1/N$ scales the quantity of walks of length two from a given person back to themselves (total contacts divided by number of people).

Another useful topic to consider in contact networks is the clustering of individuals in social groups. If groups are highly clustered, transmission may be high among individuals of the cluster, but rarer between clusters. In the most extreme case where we have distinct components, there would be no transmission, assuming spread only occurs through contacts. According to the authors, this quantity can be expressed as $\psi = \frac{trace(A^3)}{||A||^2 - trace(A^2)}$. That is, in the numerator, a high trace value over A^3 would likely indicate a high degree of connectivity.

Given the usefulness of describing contacts in the network representation, the authors extend the SEIRD model over contact networks, choosing to define contact as individuals/nodes i and j who come within 6 feet of one another - social distancing. Thus, the given state of a node i at a time t is a function of its k_i neighbors, and their "state" or "compartment" at a previous t.

$$n = s_t + e_t + i_t + r_t.$$

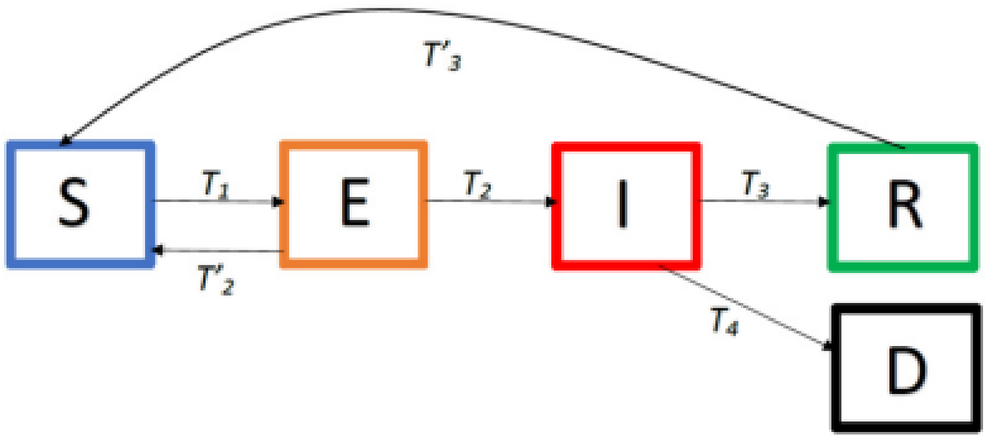
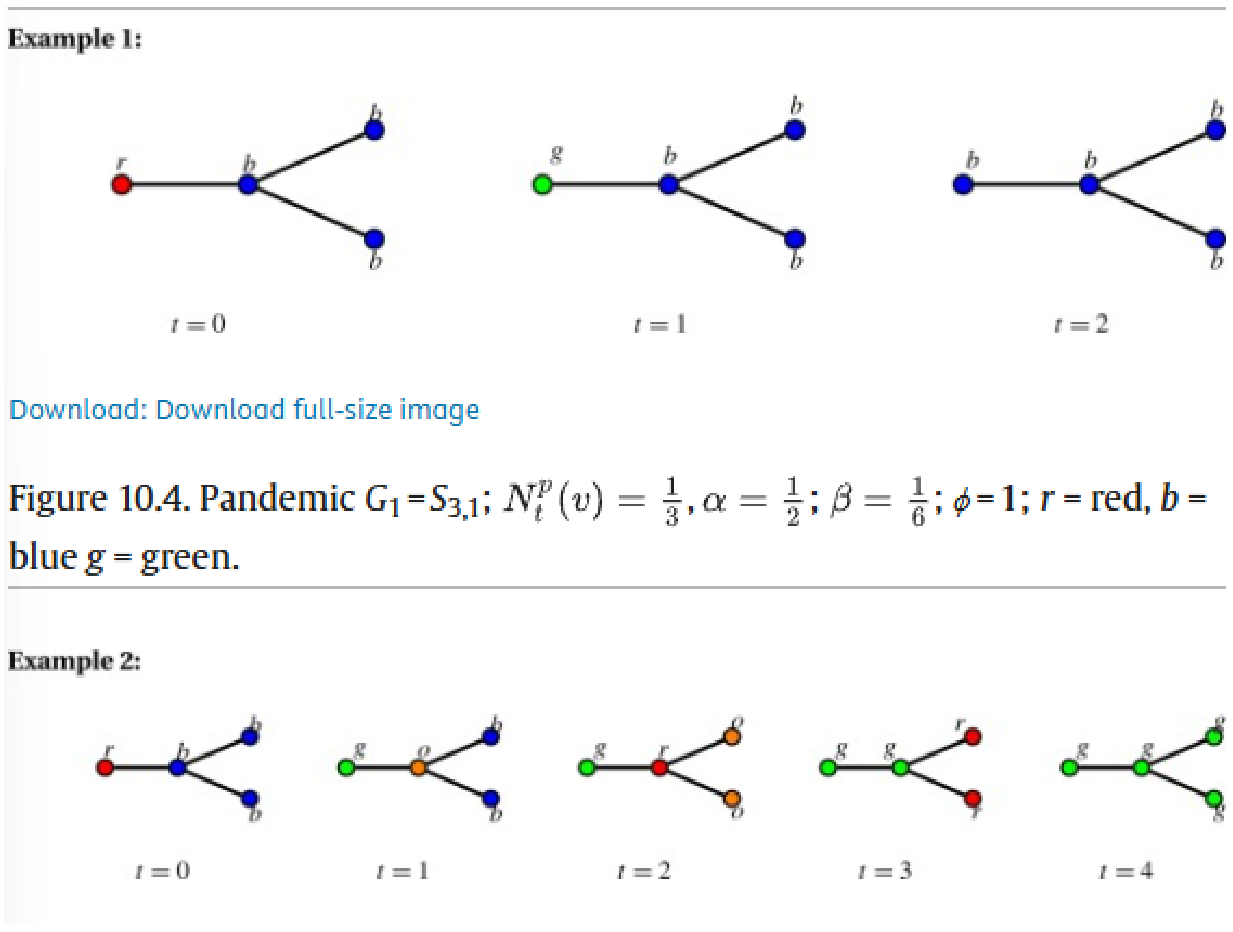


Figure 1: Figure 2. A given node i is colored with the corresponding compartment at time t. The transition of i depends upon its current compartment, t, and the state of its neighboring nodes. The comparments are as follows: S(t) susceptible, E(t) exposed, I(t) infected, R(t) recovered, and D(t) dead. This figure was source from Osaye & Alochukwu.

As the authors describe, for a given node v, the "neighborhood prevalence" of v can be understood as $N_t^p(v) = \frac{n_t(v, exposed) + n_t(v, infected)}{k_v}$. This quantity represents the ratio of v's neighbors that are "exposed" or "infected" at some t. In the following figure (Figure 3), in example 1, an infected node recovers without transmitting the disease despite having contact with one other node. In the second example, the infected individual transmits the disease to its neighbor, which propagates across the graph over time. Notably, this kind of network allows for variable degrees of contact among individuals, and by extension, the social clusters that they tend to form.



Example 1:

Download: Download full-size image

Figure 10.4. Pandemic $G_1 = S_{3,1}$; $N_t^p(v) = \frac{1}{3}$, $\alpha = \frac{1}{2}$; $\beta = \frac{1}{6}$; $\phi = 1$; *r* = red, *b* = blue *g* = green.

Example 2:

Figure 2: Figure 3. This figure shows the SEIRD model applied over a contact network in two cases. The first case where no transmission occurs despite contact, and the other in which the disease propagates down the network. This figure was source from Osaye & Alochukwu.

In this framework, susceptible individuals (blue) may become exposed from being connected to individuals who are infected (red) or exposed (orange). Exposed individuals may either transition to a state of infection, or back to susceptibility. A parameter ϕ and ψ can additionally be specified for the recovery and death rates, which vary under different kinds of disease. Lastly, an infected individual may transition to a state of recovery (green) or death (black). Some of the key assumptions of this model included a fixed population size n and no addition of vertices, which may not adequately reflect shifting social groups, migration, and birth rates. In my opinion, an additional parameter that was a big oversight was the exclusion of the length of infection, as this will likely change the rate of probability of transmission with a greater duration.

Ultimately, Osaye and Alochukwu propose a model for Covid-19 that incorporates compartmental and graph-based approaches, which would theoretically better represent the variation of degree of contacts for individuals within a social network. Several key questions remain about how they could improve or test such a model in the future. Firstly, how could this approach be used to understand local dynamics of disease spread within clusters of the larger network, and how do these local dynamics affect the global spread? Secondly, how would such a model look when calibrated or referenced against actual pandemic data? Lastly, how would these networks vary when modeled over different population demographics and locations?

Works Cited

Osaye, F. J., & Alochukwu, A. (2023). Covid-19 pandemic model: a graph theoretical perspective. In Advances in Epidemiological Modeling and Control of Viruses (pp. 285–303). Elsevier. https://doi.org/10.1016/b978-0-32-399557-3.00015-6