Assignment 1

This is our first assignment! It has three primary purposes:

- To help you re-engage with some important ideas and results from the two mathematical foundations of the study of networks: linear algebra and probability.
- To get you "set up" for the course by installing software for network computations and introducing yourself to the class on EdStem.
- To get you used to how the homework system works in MATH 168, as it's a bit different from what you may be used to.

This assignment should be submitted on Gradescope, with pages assigned to each problem. Writing your solutions in LaTeX is highly encouraged but not required.

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. (Newman 6.1)

Part (a)

Which word or words from the following list describe each of the five networks below?

Words to use: *directed, undirected, cyclic, acyclic, approximately acyclic, planar, approximately planar, tree, approximate tree.*

- *The internet, at the level of autonomous systems*: is most probably an example of a **directed** graph. We can think of a client connecting to a provider/host in a particular direction. In a peer to peer connection, these could perhaps be bi-directional arrows too. Whether we should consider the internet (in this interpretation) to be **cyclic** or **approximately acyclic** is dependent on what we are talking about; I claim routing is generally acyclic as it would be wasteful for a given node to route/facilitate some transaction back to itself. That is not to say, however, that peer to peer systems are acyclic; within their component, peers should be connected to one another forming a **cycle**.
- *A food web*: is an example of a **directed**, highly **cyclic**, highly **non-planar** graph. There is a natural notion of consumer/consumed organism in these relationships, justifying its directed-ness. At the macroscopic (multi-cellular) level, I would argue that there can be a somewhat hierarchical structure (almost tree-like) when we talk about the rankings of primary producers/primary consumers/secondary consumers, etc... Though, at the microscopic level, I would argue that this ordering is a little more ambiguous with bacteria, archaea, and other microorganisms forming cycles. I claim that none of the overly simplistic networks I describe, at the most basic level, will truly be able to capture relationships of predation, commensalism, mutualism, or parasitism.
- *The stem and branches of a plant*: is a clear example of a **tree** that is **acyclic** and **undirected** (unless we are to consider things such as the directional flow of nutrients, water, etc...). It is acyclic as branches and stems have an inherent hierarchy where branches and stems only connect to their parent branch once, and not to a parent's parent branch, etc....
- *A spider web*: a clear example of an **undirected**, **cyclic**, and **planar** graph. Assuming that intersections of edges on the web are to be considered nodes, there is no inherent direction to each edge from node to node (aside from the direction in which the edge was created by the spider). It is clearly cyclic, as for any node, there should be a cycle that starts and ends at the given node. Trivially, if we are to define nodes as intersections of the web over other webs, it is also planar.
- *A complete clique of four nodes*: Is a **cyclic**, **undirected**, and **non-planar** graph. That is, cliques are defined over undirected graphs, and with a 4-node clique, given that all nodes must be adjacent, at least one edge must cross another (non-planar). Given that all nodes of a clique are connected to all other nodes of the clique, it must also necessarily be cyclic.

Part (b)

Give one real-life example of each of the following types of networks, not including the five examples above:

- *An acyclic (or approximately acyclic) directed network*: A family tree is an acyclic directed network. The direction should be specified between a parent/child relationship, and children can not be parents to their ancestors (acyclic).
- *A cyclic directed network*: The circulatory system is an example of a directed network. There is a particular direction of flow of blood to and from the heart, where we can specify things such as capillaries, ventricles, arteries, and veins as nodes and the flow between them as edges. In a healthy and stable individual, nodes must have a at least one directed edge to it and from it to ensure the continuous flow of blood throughout the body.
- *A tree (or approximate tree)*: The Stockholm Tunnelbana (underground metro system) is an example of an approximate tree. There are only three underground lines or around 9 stops/nodes that are capable of forming a cycle, which is relatively small when you consider the total number of stations. All rail-lines radiate out from T-centralen (the shallowest node) and *mostly* do not reconnect with one another at other levels of the "approximate" tree.
- *A planar (or approximately planar) network*: The perfect/ideal circuit diagram can be understood as a planar or approximately planar graph whose edges are wires connecting components (nodes). We want to minimize the number of wires that cross, as that would imply on the actual PCB that we need to add a new layer for wiring, which costs more to manufacture.
- *A bipartite network*: The network of students and teachers. One set of nodes is the set of students being taught in a particular subject. Given that students are hopefully not teaching Middlebury college classes, there should be no edges between them. The other set of nodes is the teachers, professors are generally not instructing other professors. There should only be edges between the set of students and professors, and not among each set.

Part (c)

Describe briefly one empirical technique that could be used to measure the structure of each of the following networks (i.e., to fully determine the positions of all the edges):

- *The World Wide Web*: We can scrape all links on each web page and see which pages they refer to, thus creating a directed network of nodes (pages) referring to other nodes (pages). This is interesting, as we can apply the Page Rank algorithm to understand how important pages are.
- *A citation of scientific papers*: If we are given a set of scientific papers, we can scrape their references section and check which other research papers they refer to. If on a given paper it refers to another paper, we create a directed edge from that paper (node) to the other. We could also think about adding some weight for each edge according to how heavily or frequently a paper refers to another.
- *A food web*: As an ecologist, we would first identify the species present within a given ecological community; sample along some transects and count/identify organisms in the

area. For each species (node), we observe whether they are primary producers, consumers, etc..., and note predator/prey relationships. Based on these feeding relationships at different trophic levels, we would assign directed edges between consumer/consumed.

- *A network of friendships between a group of co-workers*: First I would specify the criteria for friendship. Is it according to the frequency that they spend time together in person? Friends on social media? If it is the former, I would probably give each participant (node) a survey and ask them to list the people they consider their friends. If it is the latter, we only have to check their follower/followed list on social media. For each participant (node) we construct a directed edge according to whether they consider someone else their friend.
- *A power grid*: We would need to identify the given substations, generators, and power consumers (all nodes) in the network first. We would then identify the layout of power lines (edges) between these nodes. If we did not have access to this information, a less exact way of identifying the layout of the network would be to observe historical power outage reports according to which substations/generators go down.

Problem 2. (Newman 6.4)

Let $\mathbf{A}$ be the adjacency matrix of an undirected network and $\mathbf{1}$ be the column vector whose elements are all 1. In terms of these quantities, write expressions for

i. The vector $\mathbf{k}$ whose ith element is the degree k_i of node i;

$\mathbf{k} = \mathbf{A} \cdot \mathbf{1}$ (as was shown in the warmup)

ii. The number m of edges in the network;

$m = \frac{1}{2}\mathbf{1}^T\mathbf{A}\mathbf{1}$, the sum of degrees of nodes / 2 so that we don't consider duplicate edges.

iii. The matrix $\mathbf{N}$ whose element N_{ij} is equal to the number of common neighbors of nodes i and j;

$\mathbf{N} = \mathbf{A}^T\mathbf{A} = \mathbf{A}^2$ for the undirected network (symmetric $\mathbf{A}$).

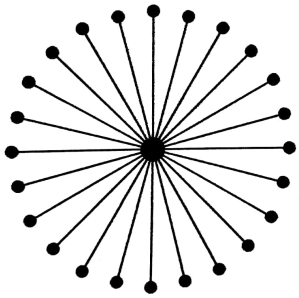
iv. The total number of triangles in the network, where a triangle means three nodes each connected by edges to both of the others.

Number of triangles $= \frac{1}{6}Tr(A^3)$ That is, $\mathbf{A^3}$ gives us the matrix of walks of length 3 (3 sides to our triangle). We then go along the diagonal as we want the set of walks starting from a node, of length 3, back to the same node (a triangle is a cycle). We divide by 1/6 to remove duplicate walks in the undirected graph.

We divide by 1/6, as every possible triangle in a graph consists of 3 nodes. In the undirected setting, which node we start from or go to next doesn't matter, only as long as we achieve a cycle in 3 nodes. With 3 nodes, there are 6 possible permutations/walks we could follow in a cycle. Given that A^3 contains ALL walks of length 3 from node i to j (and we select only walks from i to i by tracing the diagonal), we divide by 1/6 to remove this redundancy that would only matter in the directed case.

Problem 3. (Inspired by Newman 6.6)

A "star graph" consists of a single central node with $n-1$ other nodes connected to it, like this one:



We want to determine the largest (most positive) eigenvalue of the adjacency matrix of a star graph with n nodes.

This is a challenging problem that you'll want to spend some time with. Key concepts that you may need to brush up on include determinants, characteristic polynomials, expansion by minors, and diagonalization.

Part (a)

Compute the eigenvalues in the case $n = 2$ and $n = 3$. It is not necessary to compute the eigenvectors.

For $n = 2$:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$det(A - \lambda I) = 0$$

$$\begin{vmatrix} -\lambda & 1 \\ 1 & -\lambda \end{vmatrix} = (-\lambda * -\lambda) - (1 * 1) = \lambda^2 - 1 = 0$$

So, $\lambda_1 = 1, \lambda_2 = -1$ for $n = 2$.

For $n = 3$:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$det(A - \lambda I) = 0$$

I go over the first row for the determinant:

$$\begin{vmatrix} -\lambda & 1 & 1 \\ 1 & -\lambda & 0 \\ 1 & 0 & -\lambda \end{vmatrix} = -\lambda \begin{vmatrix} -\lambda & 0 \\ 0 & -\lambda \end{vmatrix} - 1 \begin{vmatrix} 1 & 0 \\ 1 & -\lambda \end{vmatrix} + 1 \begin{vmatrix} 1 & -\lambda \\ 1 & 0 \end{vmatrix} = -\lambda^3 + 2\lambda = 0$$

$$-\lambda(\lambda^2 - 2) = 0$$

$\lambda^2 = 2$ or $\lambda = 0$ are valid solutions to $-\lambda(\lambda^2 - 2) = 0$, thus:

$$\lambda = 0, \lambda = -\sqrt{2}, \lambda = \sqrt{2}$$

Part (b)

Find all the eigenvalues of the matrix $\mathbf{A}^2$ (this is easier than finding the eigenvalues of $\mathbf{A}$ directly).

~~If $\mathbf{A}x = \lambda x$, then $\mathbf{A}^2x = \lambda^2 x$. Trivially, we have: $\lambda = 0, \lambda = 2, \lambda = 2$~~

Given some symmetric matrix A, by definition of **Spectral Theorem**: $\mathbf{A} = \mathbf{U}^{-1}\mathbf{D}\mathbf{U}$, where D is the diagonal matrix of eigenvalues of A and eigenvectors in U.

A^2 is therefore:

$$\mathbf{A}^2 = (\mathbf{U}^{-1}\mathbf{D}\mathbf{U})(\mathbf{U}^{-1}\mathbf{D}\mathbf{U})$$

Since $U^{-1}U = I$:

$$\mathbf{A}^2 = \mathbf{U}^{-1}\mathbf{D}^2\mathbf{U}$$

Let $\lambda_1, ...\lambda_n$ be the eigenvalues of the $n \times n$ symmetric matrix A

$$\mathbf{D}^2 = \begin{bmatrix} \lambda_1^2 & 0 & \cdots & 0 \\ 0 & \lambda_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n^2 \end{bmatrix}$$

If a $n \times n$ matrix $\mathbf{A}$, for $i = 1, ..., n$ has eigenvalues λ_i, matrix $\mathbf{A}^2$ must have eigenvalues λ_i^2 by the **Spectral Theorem**. Simply put, the eigenvalues of $\mathbf{A}^2$ are the eigenvalues squared of $\mathbf{A}$.

Let G be the star graph with n nodes where we call the central node 1. By definition, G has $n - 1$ other nodes connected to that central node, and thus $n - 1$ different "spokes". Here are some of the key features of this graph and A^2, the matrix representing walks of length 2:

i. There exist exactly $n - 1$ walks of length 2 from node 1 back to itself (moving to any "end" of a given spoke and back to node 1).

ii. There exists no walks of length 2 from node 1 to any other node that is not node 1 in the graph.

iii. There exists exactly one walk of length 2 from u to v where 1 is not u nor v.

We can therefore write A^2, where it is a $n \times n$ matrix, as:

$$\mathbf{A}^2 = \begin{bmatrix} n-1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \cdots & 1 \end{bmatrix}$$

We can find the eigenvalues of this matrix by performing a cofactor expansion along the first row, where $A_{\hat{i}\hat{j}}$ denotes the $n-1 \times n-1$ matrix from $A - \lambda I$ without the ith row and jth column, and J is the $n-1 \times n-1$ matrix of ones:

$$\det(A^2 - \lambda I) = (n-1-\lambda) \times \det(J - \lambda I) - 0 \times \det(A_{\hat{2}\hat{2}}) + \ldots - \ldots + 0 \times \det(A_{\hat{n}\hat{n}})$$

Note that only the first term is non-zero, as all other entries along the first row are 0. Thus, we can cancel the rest out.

$$= (n-1-\lambda) \times \det(J - \lambda I)$$

We know that the determinant is the product of eigenvalues, and that the $n-1 \times n-1$ matrix of ones has eigenvalues $n-1$ with multiplicity 1 and 0 with multiplicity $n-2$, giving us the characteristic polynomial:

$$(n-1-\lambda)((n-1-\lambda)(-\lambda)^{n-2}) = (n-1-\lambda)^2(-\lambda)^{n-2}$$

It follows that our eigenvalues are $\lambda = n-1, \lambda = 0$ for the star graph with multiplicities 2 and $n-2$ respectively.

Part (c)

Justify the claim that the largest eigenvalue of $\mathbf{A}$ is the positive square root of the largest positive eigenvalue of $\mathbf{A}^2$. **This is not generally true for arbitrary square matrices**! You'll need to use the fact that $\mathbf{A}$ is symmetric (and a big theorem from linear algebra) to justify your argument.

We are given that $\mathbf{A}$ is a symmetric matrix (i.e. $\mathbf{A}_{ij} = \mathbf{A}_{ji}$). The Spectral Theorem for Real Symmetric Matrices tells us a symmetric matrix, $\mathbf{M} = \mathbf{U}^{-1}\mathbf{D}\mathbf{U}$, where $\mathbf{D}$ is the diagonal matrix of eigenvalues of $\mathbf{A}$.

We can express this as $\mathbf{A}^2 = (\mathbf{U}^{-1}\mathbf{D}\mathbf{U})(\mathbf{U}^{-1}\mathbf{D}\mathbf{U})$. Since $\mathbf{U}\mathbf{U}^{-1} = \mathbf{I}$, $\mathbf{A}^2 = \mathbf{U}^{-1}\mathbf{D}^2\mathbf{U}$

$$\mathbf{D}^2 = \begin{bmatrix} \lambda_1^2 & 0 & \cdots & 0 \\ 0 & \lambda_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n^2 \end{bmatrix}$$

That is, the eigenvalues squared of $\mathbf{A}$ are the eigenvalues of $\mathbf{A}^2$. The largest eigenvalue of $\mathbf{A}^2$ is: $\max \text{of} \lambda_1^2, \lambda_2^2, ..., \lambda_n^2$, so the largest eigenvalue of $\mathbf{A}$ is: $\sqrt{\max \text{of} \lambda_1^2, \lambda_2^2, ..., \lambda_n^2}$.

Problem 4. (Eigenpairs of Symmetric Matrices and Optimization)

You're likely familiar with the algebraic definition of the eigenvalues and eigenvectors of a matrix $\mathbf{A}$ as solutions to the equation $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$. In this problem, we will prove an additional viewpoint on eigenvalues and eigenvectors that is especially important in network science and many other fields of applied mathematics:

For symmetric matrices, eigenpairs are solutions of optimization problems.

Let $\|\mathbf{v}\|$ be the Euclidean norm of a vector $\mathbf{v}$. Here's our theorem:

Theorem 1. *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $\mathbf{A}$ ordered from smallest (λ_1) to largest (λ_n). Then,*

$$\lambda_n = \max_{\mathbf{v}\in\mathbb{R}^n} \left\{ \mathbf{v}^T\mathbf{A}\mathbf{v} \;\middle|\; \|\mathbf{v}\| = 1 \right\} \ .$$

In words, the largest value of the function $f(\mathbf{v}) = \mathbf{v}^T\mathbf{A}\mathbf{v}$ that can be obtained by picking a vector $\mathbf{v} \in \mathbb{R}^n$ satisfying $\|\mathbf{v}\| = 1$ is λ_n. The reason that this theorem matters from an applied standpoint is that many network science and data analysis tasks correspond to maximizing functions that look like $f(\mathbf{v}) = \mathbf{v}^T\mathbf{A}\mathbf{v}$.

Part (a)

Prove the theorem in three steps.

- Expand $\mathbf{A}$ in an orthonormal basis of eigenvectors:

  $$\mathbf{A} = \sum_{i=1}^{n} \lambda_i \mathbf{u}_i\mathbf{u}_i^T \ ,$$

  where each $\mathbf{u}_i$ is an eigenvector $\mathbf{A}$ with eigenvalue λ_i. As a reminder, orthonormality means that $\|\mathbf{u}_i\| = 1$ and $\mathbf{u}_i^T\mathbf{u}_j = 1$ if $i = j$ and 0 otherwise. Explain carefully why you are allowed to perform this expansion (cite a theorem).

- Write $\mathbf{v} = \sum_{i=1}^n \alpha_i\mathbf{u}_i$ for some undetermined coefficients $\{\alpha_i\}$, and explain why you are allowed to do this. Compute $\|\mathbf{v}\|$ in terms of the coefficients $\{\alpha_i\}$, and determine what the constraint $\|\mathbf{v}\| = 1$ implies about $\{\alpha_i\}$.

- Finally, use the first two steps to compute $\mathbf{v}^T\mathbf{A}\mathbf{v}$ directly in terms of $\{\lambda_i\}$ and $\{\alpha_i\}$. Show that the result is maximized when $\alpha_n = 1$ and $\alpha_i = 0$ for $i < n$. Include a closing sentence to help your reader connect this final result to Theorem 1.

The expansion is valid provided the **Spectral Theorem of Matrix Decomposition**. That is, for a matrix $\mathbf{M}$ that is symmetric, we can express: $\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{U}^T$, where $\mathbf{D}$ is the diagonal matrix of eigenvalues $\lambda_1, ..., \lambda_n$ of $\mathbf{M}$, and the columns of $\mathbf{U}$ form eigenvectors of $\mathbf{M}$ that correspond to their eigenvalues and that can form an orthogonal basis. This is equivalent to writing $\mathbf{A} = \sum_{i=1}^n \lambda_i\mathbf{u}_i\mathbf{u}_i^T$ but using summation notation instead, specifying each eigenvalue from along the diagonal in D and the individual columns/rows of U, U^T.

We can write $\mathbf{v}$ as a linear combination of α_i and the basis vectors of u_i, per the definition of the basis; we can obtain any element in a vector space as a linear combination of a set of vectors that form the basis of that vector space. The norm of $||v|| = \sqrt{v^T v} = \sqrt{(\mathbf{v} = \sum_{i=1}^n \alpha_i \mathbf{u}_i)^T (\mathbf{v} = \sum_{j=1}^n \alpha_j \mathbf{u}_j)} = \sqrt{\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j u_i^T u_j}$. As was stated above, $\mathbf{u}_i^T \mathbf{u}_j = 1$ if $i = j$ and 0 otherwise (we can simplify by removing one of the inner summations as we are multiplying by 0 when $i \neq j$). So, $||v|| = \sqrt{\sum_{i=1}^n \alpha_i^2}$. Thus the constraint of $||v|| = 1$ is the constraint that the squared sum of α_i must also be 1.

$v^T A v = (\sum_{i=1}^n \alpha_i \mathbf{u}_i)^T A (\sum_{j=1}^n \alpha_j \mathbf{u}_j)$, obtained by substituting my above results, can be expressed as $(\sum_{i=1}^n \alpha_i \mathbf{u}_i)^T (\sum_{j=1}^n \alpha_j \lambda_j \mathbf{u}_j)$ since $A u_j = \lambda_j u_j$. We expand it as $\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \lambda_j u_i^T u_j$ and simplify (using the previous technique) to $\sum_{i=1}^n \alpha_i^2 \lambda_i$. When $\alpha_n = 1$ under the constraint, $v^T A v = \lambda_n$. That is, by maximizing the weight of the n-th eigenvalue (constrained to $||v|| = 1$), we are applying the largest weight on the greatest eigenvalue and thus maximizing $v^T A v$, which can also be expressed in terms of its corresponding eigenvector/eigenvalue/alpha.

Part (b)

Using Part A, give an *extremely short* proof that

$$\lambda_1 = \min_{\mathbf{v} \in \mathbb{R}^n} \left\{ \mathbf{v}^T \mathbf{A} \mathbf{v} \;\middle|\; \|\mathbf{v}\| = 1 \right\} \ .$$

By the **Spectral Theorem**, A has basis of eigenvectors $u_1, ...u_n$ and eigenvalues $\lambda_1, \leq ..., \leq \lambda_n$. We are given the constraint $\sum_{i=1}^n \alpha_i^2 = 1$.

We have previously shown that $v^T A v = \sum_{i=1}^n \alpha_i^2 \lambda_i$. That is, the weighted summation of eigenvalues gives us $v^T A v$. Given that the smallest eigenvalue is λ_1, the smallest possible quantity for $v^T A v$ can be found when we apply the entirety of the weight on $\alpha_1^2 \lambda_1$ such that $v^T A v = \lambda_1$. In other words, $\lambda_1 = \min_{v \in R^n} \{v^T A v | \|v\| = 1\}$.

Part (c)

Suppose that $\mathbf{u}_n$ is an eigenvector of $\mathbf{A}$ with eigenvalue λ_n. Prove that

$$\lambda_{n-1} = \max_{\mathbf{v} \in \mathbb{R}^n} \left\{ \mathbf{v}^T \mathbf{A} \mathbf{v} \;\middle|\; \|\mathbf{v}\| = 1 \ , \ \mathbf{v}^T \mathbf{u}_n = 0 \right\} \ .$$

You'll find many of the same ideas that you used in Part (a) to be helpful.

By the **Spectral Theorem**, A has basis of eigenvectors $u_1, ...u_n$ and eigenvalues $\lambda_1, \leq ..., \leq \lambda_n$. We are given the constraint $\sum_{i=1}^n \alpha_i^2 = 1$. We previously showed that for $v \in R^n$: $v = \sum_{i=1}^n \alpha_i u_i$ where $\sum_{i=1}^n \alpha_i^2 = 1$, can be substituted into $v^T A v$:

$$v^T A v = (\sum_{i=1}^n \alpha_i \mathbf{u}_i)^T A (\sum_{j=1}^n \alpha_j \mathbf{u}_j)$$

$$= (\sum_{i=1}^{n} \alpha_i \mathbf{u}_i)^T (\sum_{j=1}^{n} \alpha_j \lambda_j \mathbf{u}_j)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \lambda_j u_i^T u_j$$

$$= \sum_{i=1}^{n} \alpha_i^2 \lambda_i$$

Given $v^T u_n = 0, \|v\| = 1$, v must be orthogonal to u_n. We can substitute v:

$$v^T u_n = (\sum_{i=1}^{n} \alpha_i u_i)^T u_n$$

$$= \sum_{i=1}^{n} \alpha_i u_i^T u_n$$

Given $i = n$, $u_i^T u_n = 1$ (as was stated in part a. of the problem) and for $i \neq n$, $u_i^T u_n = 0$, we can get rid of the summation as only when $i = n$, is the inside expression non-zero:

$$v^T u_n = \alpha_n \cdot 1$$

Since $v^T u_n = 0$:

$$\alpha_n = 0$$

That is to say, for $v^T A v = \sum_{i=1}^{n} \alpha_i^2 \lambda_i$, because of the constraint $v^T u_n = 0$ (and consequently $\alpha_n = 0$):

$$\sum_{i=1}^{n} \alpha_i^2 \lambda_i = \sum_{i=1}^{n-1} \alpha_i^2 \lambda_i$$

Thus, the maximum possible value happens when the entirety of the weight is assigned to the largest eigenvalue of $\lambda_1, ..., \lambda_{n-1}$. Thus:

$$\lambda_{n-1} = \max_{\mathbf{v} \in \mathbb{R}^n} \left\{ \mathbf{v}^T \mathbf{A} \mathbf{v} \;\middle|\; \|\mathbf{v}\| = 1 \;,\; \mathbf{v}^T \mathbf{u}_n = 0 \right\}$$

Problem 5. (Trees)

Recall that a *tree* is a connected graph which contains no cycles. Throughout this problem, you may use without proof the fact that every tree has at least two *leaves*: nodes of degree exactly equal to 1.

Part (a)

Prove that a tree with n nodes has exactly $n-1$ edges.

Let T_n be a tree with n nodes ($T_n = (V, E)$ where $|V| = n$). We can say without proof that every tree has at least two leaves (nodes of degree exactly 1).

For this proof we will use proof by induction.

Base case: trivially, for the tree with $n = 1$ nodes (T_1), it can have no edges with a degree of 0 and no nodes to connect to. In other words, for $n = 1$ node, there are $n - 1 = 0$ edges, meaning our base case holds.

~~As our inductive assumption, we assume that for n nodes, there are $n-1$ edges in T_n.~~

~~For $n+1$ nodes, the number of edges in $T_{n+1} = (n-1) +$ number edges created by adding 1 more node. Since T_n is connected and contains no cycles, when we add a new node, generating T_{n+1}, we add a single edge. That is, if we add more than one edge to the new node, we create a cycle, if we add no edges, we generate a cycle. The number of edges in T_{n+1} is thus $(n-1)+1 = n$~~

~~By induction, given that the base case holds and our assumption holds for $n+1$ nodes, we can say that any tree with n nodes has exactly $n-1$ edges.~~

For our inductive assumption, let us assume for some n nodes greater than 1, a tree T_n has exactly $n-1$ edges.

We will apply the induction hypothesis on the tree T_{n+1}: the tree of $n+1$ nodes, by pruning a leaf node l. Let leaf node l have a parent p in T_{n+1}. Given that leaves are nodes of degree exactly one, there is exactly one edge from l; the edge to to p. Suppose that we were to prune l in T_{n+1}: $T_{n+1} - l = T_n$. By removing a single node l with degree 1, T_n must be the tree with the number of nodes of $T_{n+1} - 1 = n$. Since T_n has n nodes, by our inductive hypothesis, it must also have $n-1$ edges. If we were to reintroduce the leaf node l which must have degree 1, the number of edges in T_{n+1} must be $(n-1)+1 = n$.

Since our inductive hypothesis holds for the base case of the tree of a single node T_1, and for a tree T_{n+1} with $n+1$ nodes and n edges, any tree T_n with n nodes must have exactly $n-1$ edges.

Part (b)

Prove that any connected graph with exactly $n-1$ edges and n nodes is a tree.

Let G be a graph with n nodes and $n-1$ edges (i.e. $G = (V, E)$ where $|V| = n, |E| = n-1$). To prove that G is a tree, we must demonstrate that it meets the criteria of the definition: acyclic and connected. We will do so by contradiction.

First we must show that G is acyclic. Let us assume G is a graph with n nodes and $n-1$ edges. By contradiction, let us suppose that G forms at least one cycle. If we were to remove enough edges to break the cycle in G, forming a graph T, T would still be connected, leaving us with n nodes and less than $n-1$ edges. Given that this new graph T is acyclic and connected, it is a tree. However, we showed in the previous proof that a tree T has exactly $n-1$ edges for n nodes. That is, $n-1 < |E|$, the edge set of G. Yet we said that $|E| = n-1$ for G. By contradiction, the graph must be acyclic.

~~We must then show that G is connected. Let us assume by contradiction that G is disconnected. By the definition of a disconnected graph, let G_1 and G_2 be components of the graph (meaning no edges between G_1 and G_2). Let n_1 and n_2 be the number of nodes in each G_1 and G_2 components respectively. We must have $n_1 + n_2 = n$, since there are n total nodes in graph G. Let e_1, e_2 denote the number of edges in G_1, G_2 respectively. We must have $e_1 + e_2 = n-1$ according to $|E|$ of G, where $e_1 \geq n_1 - 1, e_2 \geq n_2 - 1$ such that each component is connected. $e_1 + e_2 \geq n-1$ can be rewritten as $(n_1 - 1) + (n_2 - 1) \geq n-1$, and then $n_1 + n_2 - 2 \geq n-1$, and $n-2 \geq n-1$. By contradiction, G must therefore be connected.~~

Since G with n nodes and $n-1$ edges is both acyclic and connected, it must necessarily be a tree.

Part (c)

Prove that, between any two nodes i and j in a tree, there exists exactly one path.

Let i, j denote two arbitrary nodes in a tree T. As per our last proofs, we have demonstrated that T must be acyclic, connected, and have exactly $n-1$ edges for n nodes. We will use proof by contradiction.

Suppose by contradiction that i and j are connected by at least two distinct paths in a tree T. Let $P_1 = (i, ..., x, ..., j)$, $P_2 = (i, ..., y, ..., j)$. That is, both paths start at i and end at j. At some common node, both paths from i to j diverge, but reconnect eventually at or before j. P_1 may pass through x, while P_2 does not, and P_2 may pass through y while P_1 does not. With distinct paths in the graph from i to j, there exists a cycle. This contradicts the assumption that T is a tree; a tree cannot be cyclical.

Suppose by contradiction that are is no path from i to j in tree T. A tree, by our last proof, must be connected. However, if no path exists between i and j, there are at least two distinct components of T, making it disconnected. By contradiction, there must be a path from i to j in a tree T.

Thus, given that T must have no more than one path between arbitrary nodes i and j, and no less than one path, there is exactly one path between i and j such that the graph forms a tree.

Part (d)

A *bipartite graph* can be partitioned into node sets A and B such that no node in A connects to any other nodes in A and no nodes in B connect to an y other nodes in B. Prove that a tree is bipartite.

Let i be the root node of a tree $T = (V, E)$. As was shown in the previous proof, any node j in T can be accessed by exactly one unique path from i. Let V_A be the set of vertices at which the depth is even (i.e. $(0, 2, 4..., L)$, where L denotes the height of the tree - 1). Let V_B be the set of vertices at which the depth is odd $(0, 2, 4..., L - 1)$. Thus V_A contains the given root i and nodes at the depth of node $i + e$, where e is some even integer, and V_B contains nodes at the depth of node $i + o$, where o is odd.

Suppose by contradiction that tree T is not bipartite; there exists a V_A and V_B such that there exists an edge within one of either set of nodes. If such an edge existed between v_x and v_y in V_A or V_B, that would imply that v_x is connected to another node v_y of the same depth, or to a grandchild node, or vice versa. If v_x were connected to its grandchild, v_y, v_y would both be a child and grandchild of v_x. As was show in the previous proof, this would form a contradiction as there are two distinct paths from v_x to v_y (a cycle), and thus not a tree. If v_x were connected by an edge at the same depth as v_y in T, by definition, both nodes would have to share the same parent node while also being parents of each other. Given that a parent should occupy a depth 1 removed from a child, these nodes must also necessarily form a cycle, which is a contradiction from the original statement that T is a tree. Thus, no edges in a tree T can exists between nodes within V_A or V_B.

We must also demonstrate that there exists an edge that connects a given node in V_A to V_B. We assume by contradiction that for some given node a in V_a, there is not a b in V_B that it connects to. That is, there exists a parentless node (in terms of T) in V_A or V_B that is not the root node. This contradicts the result of the previous proof that for any given node i in T, there exists a j such that there is exactly one unique path between them. Therefore, every node in V_A must be connected to some node in V_B.

Since every node in V_A is connected to at least one node in V_B, and there are no edges within nodes of V_A or V_B, tree T must be bipartite by definition.

Problem 6. (Getting Set Up)

Install Python and the NetworkX package. The easiest way for you to get set up with Python is to download Anaconda, which includes NetworkX built in. You can then start Anaconda and open a Jupyter Notebook in which to write and run code.

***Note**: if you've previously installed Anaconda, it's recommended to uninstall and reinstall it. I encountered some errors with NetworkX that went away after a clean reinstallation.*

Open your software of choice. Write code to do two things:

- Produce a visualization of a simple network.
- Print your name.

Take a screencap of your code and output, including both your network and your name and include it with your submission.