Matrix Tree Theorem & the number of spanning trees

Omar Mohamed Said

IRPW

Abstract

We will introduce the foundations & explore four proofs of the Matrix-Tree theorem, revealing connections between graph enumeration, random walks, and spectral theory and its application.

Contents

1 Introduction & an Overview

The problem of counting the number of spanning trees in a graph is a question that no one wants to answer through enumeration. I can tell no one loves to count the number of people in the class to take attendance, and the same goes for spanning trees. The solution to this problem reveals beautiful connections between Graph theory and Linear Algebra. This paper provides a comprehensive exploration of the Matrix-Tree Theorem. We begin by establishing the necessary graph-theoretic and linear algebra foundations, introducing concepts like spanning trees, incidence matrices, and the Laplacian. We will present four distinct proofs: a classical proof using the Cauchy-Binet formula, a second using the spectral properties of the Laplacian's eigenvalues, a third employing combinatorial arguments, and a fourth using random walks. While we are reading the paper, we will try to get insights and a lot of Aha! Moments, we will also try to prove everything we can. There is a lot of work covered in this expository paper, but also there are things we didn't cover, so take a look at the table of contents. We will also explore Cayley's theorem as a consequence of the Matrix-Tree theorem; moreover, we will dive into the applications of the theorem. To excite you, here is a fact you will come across in the paper: Uniform weighted graphs have the same Laplacian Matrix of the unweighted version.

2 Graph Theory Preliminaries

In this section, we define the basic terminology of graph theory that will be used throughout this work. We focus on the essential properties of graphs that are relevant to the study of spanning trees.

Definition 2.1 (Graph)**.** A simple graph G is an ordered pair (V, E), where V is a finite set of elements called vertices, and E is a set of unordered pairs of distinct vertices, called edges. Throughout this text, "graph" will refer to a simple, undirected graph unless otherwise specified.

Definition 2.2 (Degree of a Vertex)**.** The degree of a vertex v, denoted $\deg(v)$, is the number of edges incident to it. A vertex of degree 0 is called an isolated vertex.

Proposition 2.3 (Handshaking Lemma)**.** *The sum of the degrees of all vertices in a graph is equal to twice the number of edges:* $\sum_{v \in V} \deg(v) = 2|E|$.

Proof. This is a direct consequence of double counting. Each edge connects exactly two vertices. Therefore, when summing the degrees of all vertices, each edge is counted precisely once for each of its two endpoints, contributing 2 to the total sum. ■

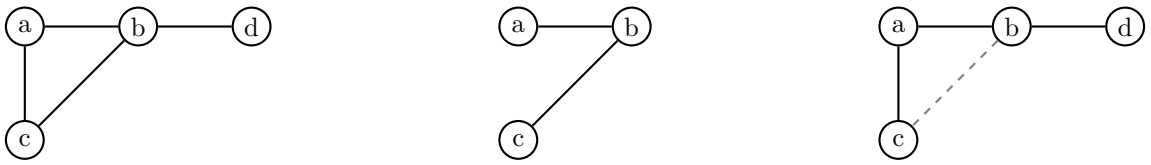
Definition 2.4 (Subgraph)**.** A graph $H = (V', E')$ is a subgraph of a graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. If H contains all edges from G that connect vertices in V', it is called an induced subgraph.

Definition 2.5 (Path and Cycle)**.** A path is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$ where $\{v_i, v_{i+1}\}$ is an edge for all $1 \leq i < k$. A cycle is a path that starts and ends at the same vertex, with at least three vertices. A graph is acyclic if it contains no cycles.

Definition 2.6 (Connected Graph)**.** A graph is connected if there exists a path between any two distinct vertices. A graph that is not connected is called disconnected and is composed of one or more connected components, which are its maximal connected subgraphs.

Definition 2.7 (Tree and Spanning Tree)**.** A tree is a connected acyclic graph. A spanning tree of a connected graph $G = (V, E)$ is a subgraph $T = (V, E')$ that is a tree and includes all the vertices of G (i.e., it is spanning). The number of spanning trees of a graph G is denoted by $\tau(G)$.

Example 2.8 (Graph Illustrations)**.** The following figures illustrate the concepts defined above. Figure 2.1a shows a simple graph G_1. Figure 2.1b shows a subgraph of G_1, and Fig. 2.1c shows a spanning tree of G_1. In G_1, the degrees are $\deg(a) = 2$, $\deg(b) = 3$, $\deg(c) = 2$, and $\deg(d) = 1$.



(a) A simple graph G_1. (b) A subgraph of G_1. (c) A spanning tree of G_1.

Figure 2.1: Illustrations of graph theory concepts.

3 Matrix-Tree Theorem Statement

To count the number of spanning trees in a graph can be computationally heavy, so luckily we have the Matrix-Tree Theorem that states if we have a graph G that is connected with n vertices and L is its Laplacian matrix. The number of spanning trees of G, denoted by $\tau(G)$, is equal to any co-factor of L. That is, for any choice of row i and column j,

$$\tau(G) = (-1)^{i+j} \det(L_{i,j})$$

where $L_{i,j}$ is the sub-matrix of L that we obtain by deleting row i and column j.

This section 3 introduces Matrix-Tree theorem statement. You can check more and see explanations at [Ope14]

Definition 3.1 (Complete Graph)**.** A complete graph on n vertices, denoted K_n, is a simple graph in which every pair of distinct vertices is connected by a unique edge. In K_n, the degree of every vertex is $n-1$, and the total number of edges is $\binom{n}{2}$.

Example 3.2. Count the Number of spanning trees of the following complete graph (K_4).

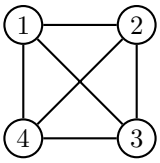


Figure 3.1: The complete graph on 4 vertices (K_4).

By Enumeration, the total number of spanning trees is 16.

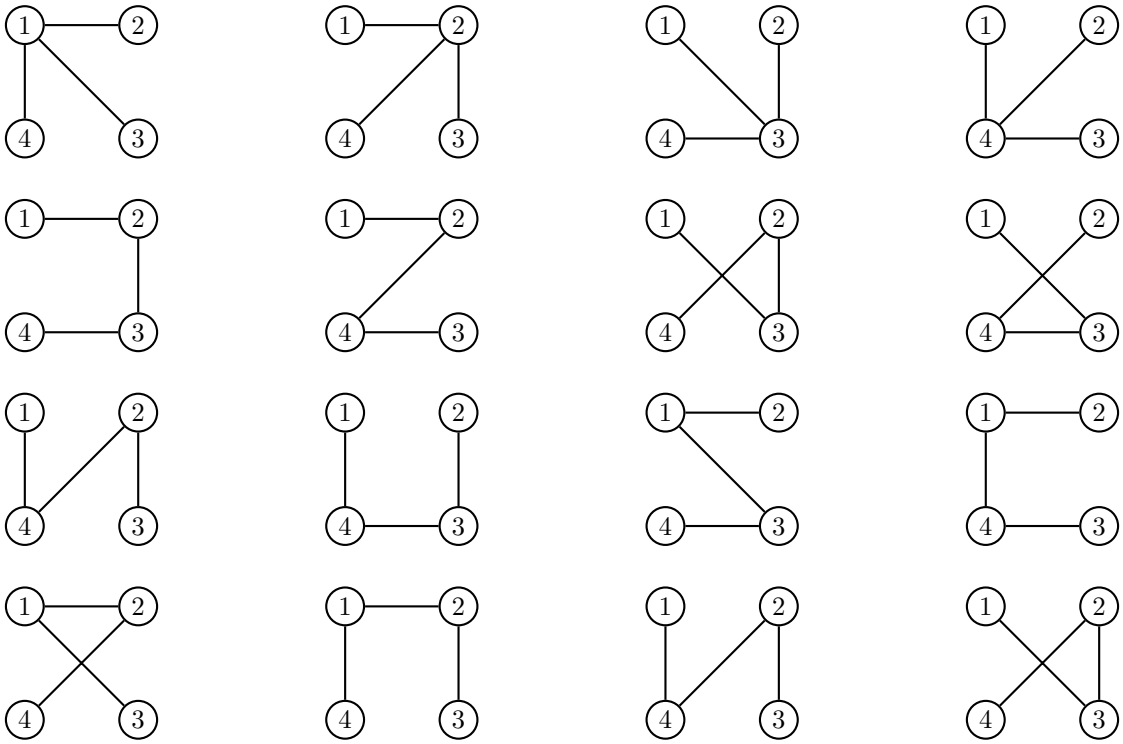


Figure 3.2: The 16 Spanning Trees of the Complete Graph K_4

$$L_{K_4} = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix} \qquad L'_{K_4} = \begin{pmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{pmatrix}$$

And the determinant of the Cofactor L'_{K_4} made by removing the first row and column is the number of spanning trees $\tau(G)$. It doesn't matter what entry we choose to remove, as we will see in the Cauchy-Binet theorem part.

$$\det(L'_{K_4}) = 3 \cdot \begin{vmatrix} 3 & -1 \\ -1 & 3 \end{vmatrix} - (-1) \cdot \begin{vmatrix} -1 & -1 \\ -1 & 3 \end{vmatrix} + (-1) \cdot \begin{vmatrix} -1 & 3 \\ -1 & -1 \end{vmatrix} = 16$$

This section 3 introduces Matrix-Tree theorem statement. You can check more and see explanations at [Ope14]

We confirmed that K_4 has 16 spanning trees, and we are sure that the Matrix-Tree Theorem works. To sense how much that means, calculating the number of spanning trees can help us create better electrical networks as we will see in section 12. Now you know what a spanning tree is before going to the next section that will help us to prove the matrix tree theorem. I want to introduce a nice formula for planar graphs, which is simply a graph with no two intersecting edges. We will meet this formula again in Section 12

Theorem 3.3.

$$V - E + C = 1$$

Where v, E, and C are the number of vertices, the number of edges, and the number of loops, respectively.

Proof. Consider a spanning tree, T, within the connected graph, G. By definition, this spanning tree connects all V vertices using the minimum number of edges required, which is always $V-1$. The edges in the original graph that are not part of this spanning tree are the ones that create the cycles, and the number of these edges is the total number of edges, E, minus the edges in the spanning tree, so $E-(V-1)$. Each of these leftover edges, when added back to the spanning tree, creates one independent cycle. Therefore, the total number of cycles, C, must be equal to this number of leftover edges, and this gives us the equation $C = E-(V-1)$, which simplifies to our desired equation, completing our proof.

■

4 Linear Algebra Representations of Graphs

We now introduce matrix representations of graphs, and introduce interesting properties and prove important propositions for Matrices like the Incidence Matrix, Degree matrix, Adjacency matrix, and the Laplacian Matrix for directed and undirected graphs. Let $G = (V, E)$ be a graph with $n = |V|$ vertices labeled $v_1, \ldots, v_n$ and $m = |E|$ edges labeled $e_1, \ldots, e_m$.

Example 4.1 (Matrix Example Graph)**.** For this section, we will use the graph $G_M = (V, E)$ with $V = \{1, 2, 3, 4\}$ and $E = \{e_1 = \{1, 2\}, e_2 = \{1, 4\}, e_3 = \{2, 3\}, e_4 = \{3, 4\}\}$.

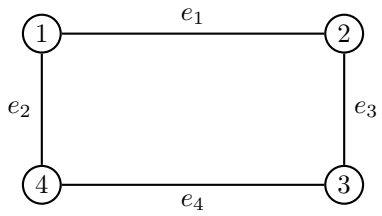


Figure 4.1: The graph G_M for Matrices.

Definition 4.2 (Adjacency Matrix)**.** The adjacency matrix $A(G)$ is an $n \times n$ matrix where $A_{iej} = 1$ if $\{v_i, v_j\} \in E$ and 0 otherwise, so the diagonal of the matrix can only contain zero entries, as there is no edge connecting the vertex to itself except for a self-loop.

Definition 4.3 (Degree Matrix)**.** The degree matrix $D(G)$ is an $n \times n$ diagonal matrix where $D_{ii} = \deg(v_i)$.

Definition 4.4 (Incidence Matrix)**.** To define an incidence matrix $I(G)$ for an undirected graph, we first assign an arbitrary orientation to each edge as in an electric network. I is an $n \times m$ matrix where $I_{ie} = +1$ if edge e starts at v_i, -1 if it ends at v_i, and 0 otherwise. For G_M, orient each edge from the smaller to larger indexed vertex.

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \quad I = \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & -1 \end{pmatrix}$$

This section 4 shows how to present graphs as matrices and propose and prove interesting properties. For more on this you can check [Gil10], [Seb21], & [Sid18]

Definition 4.5 (Laplacian Matrix)**.** The Laplacian matrix $L(G)$ is defined as $L(G) = D(G) - A(G)$. Its entries are given by:

$$L_{ij} = \begin{cases} \deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } \{v_i, v_j\} \in E \\ 0 & \text{otherwise} \end{cases}$$

For our graph G_M:

$$L = D - A = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}$$

Definition 4.6 (Rank)**.** The rank of the matrix is the number of linearly independent columns in the matrix. In other words, the number of columns that you cannot get by combining other columns.

Proposition 4.7. *The rank of I is $n - c$, where c is the number of connected components of G. For a connected graph, the rank is $n - 1$.*

Proof. Let the rows of I be $r_1, \ldots, r_n$. Each column of I contains exactly one $+1$ and one -1, so the sum of its entries is 0. This implies $\sum_{i=1}^{n} r_i = 0$, so the rows are linearly dependent and $\text{rank}(I) < n$. To show the rank is exactly $n - 1$ for a connected graph, one can show that any set of $n - 1$ rows is linearly independent. ■

Proposition 4.8. *Every square sub-matrix in I has determinant $0, +1,$ or -1.*

Proof. We prove by induction on the size k of the square sub-matrix. For the base case ($k = 1$), a 1×1 sub-matrix is just a single entry of I. Since the entries are $0, +1$, or -1, the determinant is also in this set. The base case holds. We assume that for some $k \geq 2$, any $(k-1) \times (k-1)$ sub-matrix of I has a determinant in $\{0, +1, -1\}$. Let S be a $k \times k$ sub-matrix. We analyze the columns of the matrix S. The first case is when S has a column that contains only zeros, then $\det(S) = 0$. The second case is when every column of S has at least one non-zero entry. Since the columns of S are sub-vectors of columns of I, each non-zero column must contain either one non-zero entry (± 1) or two non-zero entries ($+1$ and -1). If every column of S has exactly two non-zero entries, then it must contain one $+1$ and one -1. If we sum all the row vectors of S, the result will be the zero vector. This means the rows of S are linearly dependent, and therefore $\det(S) = 0$. If there exists a column in S with exactly one non-zero entry, then the j-th column of S has exactly one non-zero entry, say $S_{ij} = \pm 1$. We can compute the determinant of S by expanding along this column:

$$\det(S) = \sum_{r=1}^{k} (-1)^{r+j} S_{rj} \det(S_{r,j})$$

Since only one entry S_{ij} in this column is non-zero, the sum simplifies to

$$\det(S) = (-1)^{i+j} S_{ij} \det(S_{i,j}) = (\pm 1) \cdot \det(S_{i,j})$$

Here, $S_{i,j}$ is the $(k-1) \times (k-1)$ sub-matrix obtained by removing row i and column j. By our inductive hypothesis, $\det(S_{i,j}) \in \{0, +1, -1\}$. Therefore, $\det(S)$ must also be in $\{0, +1, -1\}$.
All possible cases are covered, so the induction is complete. ■

Fact 4.9. A square matrix M is invertible if and only if its columns are linearly independent.

Proof. The definition of linear independence states that the columns of M are linearly independent if the only solution to the equation $Mx = 0$ is the trivial solution $x = 0$. We assume M is invertible. This means an inverse matrix M^{-1} exists. We consider the equation $Mx = 0$. We can multiply both sides from the left by M^{-1}:

$$M^{-1}(Mx) = M^{-1}0$$

By associativity, this becomes $(M^{-1}M)x = 0$. Since $M^{-1}M$ is the identity matrix I, we get $Ix = 0$, which simplifies to $x = 0$. Thus, the only solution is the trivial one, which proves that the columns of M are linearly independent. ■

This section 4 shows how to present graphs as matrices and propose and prove interesting properties. For more on this you can check [Gil10], [Seb21], & [Sid18]

Proposition 4.10. *If G is connected, a subgraph formed by $n-1$ edges is a spanning tree if and only if the corresponding $(n-1)\times(n-1)$ sub-matrix of I (with any one row removed) is invertible.*

Proof. From Fact 4.9, a matrix M is invertible if its columns are linearly independent. We will prove that the columns of the sub-matrix are linearly independent if and only if the corresponding set of edges S is acyclic. Since G is connected and S contains $n-1$ edges, S being acyclic is the definition of a spanning tree. Suppose the set of edges S contains a cycle, C. Let the vertices of the cycle be $(v_1, v_2, \dots, v_k)$ and the corresponding edges be $e_1 = \{v_1, v_2\}, \dots, e_k = \{v_k, v_1\}$. The columns of the incidence matrix corresponding to these edges are linearly dependent. This linear dependence is preserved in the sub-matrix, so its columns are linearly dependent, and it is not invertible.
Conversely, suppose the columns of the sub-matrix are linearly dependent. Then there exists a minimal non-empty subset of columns corresponding to edges $S' \subseteq S$ that is linearly dependent. A minimal set of linearly dependent columns in an incidence matrix corresponds to a cycle in the graph. Therefore, S' contains a cycle, and thus S contains a cycle. ■

Another Proof. Using the same notation, we will give another beautiful proof by relying on the previously established result that the rank of an incidence matrix of a graph with n vertices and c connected components is $n-c$ (Proposition 4.7). Assume H is a spanning tree. Since it is, by definition, connected, its number of connected components is $c=1$. According to the rank theorem, the rank of its incidence matrix I_H is

$$\operatorname{rank}(I_H) = n - c = n - 1$$

The sum of the n rows of any incidence matrix is the zero vector, which means the rows are linearly dependent. We have a set of n row vectors whose span is an $(n-1)$-dimensional space and which sum to zero. This implies that any subset containing $n-1$ of these rows must be linearly independent. If a subset of $n-1$ rows were linearly dependent, their span would have a dimension less than $n-1$. Since the final row is just a linear combination of the others, adding it back would not increase the dimension, which would contradict the fact that the rank is $n-1$. Therefore, any $n-1$ rows of I_H form a basis for its row space. The matrix M is formed by taking exactly such a set of $n-1$ rows, so its rows are linearly independent, which makes the square matrix M invertible, as we proved in Fact 4.9.
Proving it the other way, if the $(n-1)\times(n-1)$ matrix M is invertible, its rank must be $n-1$. Because the rows of M are a subset of the rows of the full incidence matrix I_H, the dimension of the row space of I_H must be at least the dimension of the space spanned by the rows of M.

$$\operatorname{rank}(I_H) \geq \operatorname{rank}(M) = n - 1$$

But, the matrix I_H only has $n-1$ columns, so its rank cannot possibly exceed $n-1$. Combining these two facts, we must have that $\operatorname{rank}(I_H) = n-1$. We again use the rank theorem, which states that $\operatorname{rank}(I_H) = n-c$, where c is the number of connected components of the subgraph H.

$$n - 1 = n - c \implies c = 1$$

This shows that the subgraph H must be connected. A connected graph with n vertices and $n-1$ edges is, by definition, a tree. Since it includes all the vertices of G, it is a spanning tree of G. ■

Proposition 4.11. *For a graph G with an oriented incidence matrix I, $L = II^T$.*

Proof. We check the entries of II^T, where we analyze the diagonal entries and the non-diagonal entries. The diagonal entry $(II^T)_{ii} = \sum_{k=1}^{m} I_{ik}^2$. The term I_{ik}^2 is 1 if vertex v_i is incident to edge e_k and 0 otherwise. The sum is therefore the total number of edges incident to v_i, which is $\deg(v_i) = L_{ii}$. For the non-diagonal entries $(II^T)_{ij} = \sum_{k=1}^{m} I_{ik}I_{jk}$. A term $I_{ik}I_{jk}$ is non-zero only if both v_i and v_j are incident to edge e_k. This occurs only if $e_k = \{v_i, v_j\}$. In this case, one of I_{ik}, I_{jk} is $+1$ and the other is -1, so their product is -1. If no edge connects v_i and v_j, the sum is 0. This matches $L_{ij} = -1$ for an edge and 0 otherwise, and this makes the Laplacian matrix L, completing the proof. ■

Definition 4.12 (Symmetric Matrix)**.** A square matrix A is said to be symmetric if it is equal to its own transpose, $A = A^T$. That is, the entry in the i-th row and j-th column is equal to the entry in the j-th row and i-th column for all i and j.

This section 4 shows how to present graphs as matrices and propose and prove interesting properties. For more on this you can check [Gil10], [Seb21], & [Sid18]

Definition 4.13 (Eigenvalue and Eigenvector)**.** A non-zero vector $v \in \mathbb{R}^n$ is called an eigenvector of an $n \times n$ square matrix L, if there exists a scalar λ such that $Lv = \lambda v$. The scalar λ is called the eigenvalue corresponding to the eigenvector v. For the Laplacian matrix L, there is always a linearly dependent column, which means there is always a 0 eigenvalue.

Proposition 4.14. *For any vector $u \in \mathbb{R}^n$, the quadratic form is $u^T L u = \sum_{\{x,y\}\in E}(u_x - u_y)^2$.*

Proof. We use the result from Proposition 4.11, $L = II^T$, and using $(AB)^T = B^T A^T$.

$$u^T L u = u^T (II^T) u = (u^T I)(I^T u) = (I^T u)^T (I^T u)$$

This is the dot product of the vector $v = I^T u$ with itself, $\|v\|^2$. The components of v are indexed by edges. For an edge e oriented from x to y, the corresponding component of v is $u_x - u_y$. So by summing over the edges, $u^T L u = \sum_{e=\{x,y\}\in E}(u_x - u_y)^2$. ■

Proposition 4.15. *L is a real symmetric matrix. Its smallest eigenvalue is 0, and the multiplicity of this eigenvalue is equal to c, the number of connected components of G.*

Proof. Let $\lambda = 0$ be an eigenvalue with eigenvector x. Then $Lx = 0$. From the proof of Proposition 4.14, this implies $x^T L x = 0$, which means $\sum_{\{v_i,v_j\}\in E}(x_i - x_j)^2 = 0$. Since each term is non-negative, every term must be zero. Thus, for every edge $\{v_i, v_j\}$, we must have $x_i = x_j$. This means that all vertices in the same connected component must have the same corresponding value in the eigenvector x. If G has k connected components $C_1, \ldots, C_k$, we can construct k linearly independent eigenvectors for $\lambda = 0$. For each component C_i, define a vector $x^{(i)}$ where entries are 1 for vertices in C_i and 0 otherwise. Each such vector is in the null space of L, and they are mutually orthogonal. The dimension of the null space (eigenspace of $\lambda = 0$) is therefore k. For a connected graph ($k = 1$), the eigenvalue 0 has multiplicity 1, and its eigenvector is the all-ones vector 1. ■

Proposition 4.16. $\operatorname{rank}(L) = n - c$.

Proof. The rank of a symmetric matrix is n minus the multiplicity of the eigenvalue 0. From the previous proposition, this is $n - c$. ■

Definition 4.17. The adjugate of an $n \times n$ matrix M, denoted $\operatorname{adj}(M)$, is the transpose of its cofactor matrix. It satisfies the following identity $M \cdot \operatorname{adj}(M) = \operatorname{adj}(M) \cdot M = \det(M) \cdot I$, where I is the $n \times n$ identity matrix.

Now, we are comfortable to go to the next section where we will use the Cauchy-Binet theorem to give a first proof for the Matrix-Tree theorem.

5 Cauchy-Binet Theorem & Matrix-Tree Theorem First Proof

Theorem 5.1. *Let G be a connected graph with n vertices and L its Laplacian matrix. The number of spanning trees of G, denoted by $\tau(G)$, is equal to any cofactor of L. That is, for any choice of row i and column j,*

$$\tau(G) = (-1)^{i+j} \det(L_{ij})$$

where L_{ij} is the submatrix of L obtained by deleting row i and column j.

Theorem 5.2 (The Cauchy-Binet Theorem)**.** *Let A be an $m \times n$ matrix, and let B be an $n \times m$ matrix. If $m > n$, then $\det(AB) = 0$. If $m \leq n$, then*

$$\det(AB) = \sum_S (\det A[S])(\det B[S]),$$

where S ranges over all m-element subsets of the columns of A and rows of B.

Example 5.3. Let A be a 2×3 matrix and B be a 3×2 matrix.

This section 5 introduces The cauchy-Binet Theorem and a first proof for the Matrix-Tree Theorem. You can see more on this at [Ope14], [Seb21], & [Ste23]

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix} \quad B = \begin{bmatrix} c_1 & d_1 \\ c_2 & d_2 \\ c_3 & d_3 \end{bmatrix}$$

$$\det(AB) = \overbrace{\begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \cdot \begin{vmatrix} c_1 & d_1 \\ c_2 & d_2 \end{vmatrix}}^{\text{Cols 1 \& 2}} + \overbrace{\begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix} \cdot \begin{vmatrix} c_1 & d_1 \\ c_3 & d_3 \end{vmatrix}}^{\text{Cols 1 \& 3}} + \overbrace{\begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} \cdot \begin{vmatrix} c_2 & d_2 \\ c_3 & d_3 \end{vmatrix}}^{\text{Cols 2 \& 3}}$$

Now our goal is to prove that $\text{adj}(L) = \tau(G) \cdot J$, where J is the all-ones matrix. This is equivalent to proving that the statement of the Matrix-Tree theorem is true because it implies all the co-factors of L are equal to each other, and their common value is the number of spanning trees.
The proof proceeds in two main steps. First, we show that $\text{adj}(L)$ must be a multiple of the all-ones matrix J. Second, we identify that multiple as $\tau(G)$.

Proposition 5.4. *For any graph G with Laplacian L, $\text{adj}(L)$ is a constant multiple of the all-ones matrix J.*

Proof. From the ad-jugate identity, we have $L \cdot \text{adj}(L) = \det(L) \cdot I$. Because 0 is an eigenvalue of L, $\det(L) = 0$. Therefore:

$$L \cdot \text{adj}(L) = 0 \quad \text{(the zero matrix)}$$

We consider two cases for the graph G.
Case 1: G is disconnected. If G is disconnected, it has $c \geq 2$ components. We know that $\text{rank}(L) = n - c \leq n - 2$. This means that any $(n-1) \times (n-1)$ sub-matrix of L must be singular, so its determinant is 0. By definition, every co-factor of L is zero. Thus, the co-factor matrix is the zero matrix, and its transpose, $\text{adj}(L)$, is also the zero matrix. In this case, $\text{adj}(L) = 0 \cdot J$, so the proposition holds.
Case 2: G is connected. If G is connected, then $\text{rank}(L) = n - 1$. The relation $L \cdot \text{adj}(L) = 0$ implies that every column vector of $\text{adj}(L)$ is in the null space of L. Since G is connected, we proved that the null space of L (the eigen-space of the eigenvalue 0) is one-dimensional and is spanned by the all-ones vector, 1. Therefore, every column of $\text{adj}(L)$ must be a scalar multiple of 1. Let the j-th column be $\alpha_j 1$. The ad-jugate matrix has the form:

$$\text{adj}(L) = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \end{bmatrix}$$

Since L is a symmetric matrix, its ad-jugate $\text{adj}(L)$ must also be symmetric. For the matrix above to be symmetric, we must have $\alpha_1 = \alpha_2 = \cdots = \alpha_n$. Let this common scalar be α. Then:

$$\text{adj}(L) = \alpha \cdot J$$

Since $\text{rank}(L) = n - 1$, at least one co-factor is non-zero, so $\alpha \neq 0$. This completes the proof. ■

We now prove that $\alpha = \tau(G)$. Since all co-factors of L are equal to α, we only need to compute one of them. Let's compute the $(1,1)$ cofactor, which is $\det(L_{11})$, where L_{11} is L with the first row and column removed.

Proof of Kirchhoff's Theorem. Let I be an oriented incidence matrix for G. We know $L = II^T$. Let I_1 be the matrix I with its first row removed. Then the sub-matrix L_{11} can be written as $L_{11} = I_1 I_1^T$. So, we want to compute $\det(I_1 I_1^T)$.
Here, I_1 is an $(n-1) \times |E|$ matrix. We use the Cauchy-Binet Formula for the determinant of a product of non-square matrices.

$$\det(I_1 I_1^T) = \sum_{F \subseteq E, |F| = n-1} \det((I_1)_F) \cdot \det((I_1^T)_F)$$

where $(I_1)_F$ is the $(n-1) \times (n-1)$ sub-matrix of I_1 with columns corresponding to the edges in the set F. This simplifies to

$$\det(L_{11}) = \sum_{F \subseteq E, |F| = n-1} (\det((I_1)_F))^2$$

This section 5 introduces The cauchy-Binet Theorem and a first proof for the Matrix-Tree Theorem. You can see more on this at [Ope14], [Seb21], & [Ste23]

By combining Proposition 4.8 and Proposition 4.10, we know $(\det((I_1)_F))^2$ is equal to 1 if F forms a spanning tree, and 0 otherwise. The sum therefore counts a '1' for every subset of $n-1$ edges that forms a spanning tree, and this is the definition of $\tau(G)$.

$$\alpha = \det(L_{11}) = \tau(G)$$

This completes the proof of the Matrix-Tree Theorem. ■

6 Cayley's Formula via Prüfer Codes

We will try the Matrix-Tree theorem on a complete graph K_n for illustrative purposes.

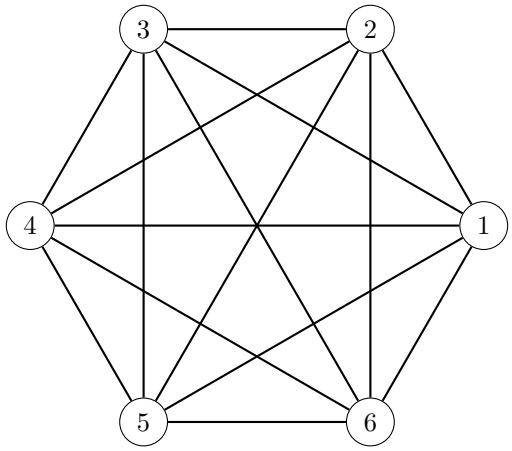


Figure 6.1: The complete graph on 6 vertices (K_6).

$$\tau(K_6) = \det \begin{pmatrix} 5 & -1 & -1 & -1 & -1 \\ -1 & 5 & -1 & -1 & -1 \\ -1 & -1 & 5 & -1 & -1 \\ -1 & -1 & -1 & 5 & -1 \\ -1 & -1 & -1 & -1 & 5 \end{pmatrix} = 6^4 = 1296$$

This value, 1296, turns out to be n^{n-2}, and so it is for every other complete graph.

Theorem 6.1 (Cayley's Formula, 1889)**.** *The number of spanning trees of a complete graph K_n, which is the number of labeled trees on n vertices, is n^{n-2}.*

We will prove this theorem by proving that the number of labeled trees on n vertices is n^{n-2}. Any such labeled tree is by definition a spanning subgraph of K_n, and since K_n contains all possible edges, any spanning tree of K_n is a labeled tree on n vertices. The two sets are therefore identical.

Definition 6.2 (Prüfer Code Encoding)**.** Let a labeled tree T have a vertex set $V = \{1, 2, \ldots, n\}$. We start with $T_1 = T$. For each step $i = 1, \ldots, n-2$:

1. Find the leaf with the smallest label in the current tree T_i. Let this leaf be b_i.
2. Identify the unique neighbor of b_i in T_i and call this neighbor a_i.
3. The tree for the next step, T_{i+1}, is formed by removing the vertex b_i and its incident edge from T_i.

The sequence of neighbors, $P(T) = (a_1, a_2, \ldots, a_{n-2})$, is the Prüfer code of the tree T.

Example 6.3 (A tree on 6 vertices)**.** Let's apply this algorithm to the following tree on $n = 6$ vertices. The steps to generate the code are summarized in the following table: The procedure stops after $n - 2 = 4$ steps and the generated Prüfer code is the sequence of neighbors: $A = (a_1, a_2, a_3, a_4) = (1, 1, 4, 4)$.

This section 6 is about cayley's theorem and a proof via Prüfer Codes. See [Rub23], [Lin19a], & for more proofs [AZ99]

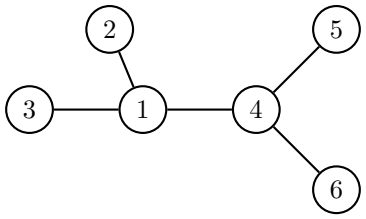


Figure 6.2: A labeled tree on 6 vertices.

Table 6.1: Prüfer code generation for the tree in Fig. 6.2.

Step (i)	Smallest Leaf (b_i)	Neighbor (a_i)	Remaining Tree T_{i+1} Vertices
1	2	1	$\{1,3,4,5,6\}$
2	3	1	$\{1,4,5,6\}$
3	1	4	$\{4,5,6\}$
4	5	4	$\{4,6\}$

Remark 6.4. In our example, after removing $\{2,3,1,5\}$, the vertices $\{4,6\}$ are left. The final edge is therefore $\{4,6\}$. This encoding gets the entire tree structure in only $n-2$ numbers.

Now, to prove that the correspondence between the tree and the code is a bijection, we must show that any sequence $A=(a_1,\ldots,a_{n-2})$ can be decoded into a unique tree.

Definition 6.5 (Prüfer Code Decoding)**.** Given a code $A=(a_1,\ldots,a_{n-2})$ and a set of vertex labels $S=\{1,\ldots,n\}$:

1. For $i=1,\ldots,n-2$, we find the smallest number $b_i\in S$ that is not in the tail of the code $\{a_i,\ldots,a_{n-2}\}$.
2. We add the edge $\{b_i,a_i\}$ to the tree.
3. We remove b_i from S.
4. After the loop, S contains two vertices, say u and v We add the final edge $\{u,v\}$.

Example 6.6 (Decoding a code for n=7)**.** We let $n=7$ and the code be $A=(2,2,4,1,7)$. Let $S=\{1,\ldots,7\}$.

- Step 1: Code tail is $(2,2,4,1,7)$. The set of available vertices not in the code is $\{3,5,6\}$. The smallest is $b_1=3$. Edge: $\{3,2\}$. Remove 3 from S.
- Step 2: Code tail is $(2,4,1,7)$. Vertices in $S\setminus\{3\}$ not in the tail are $\{5,6\}$. Smallest is $b_2=5$. Edge: $\{5,2\}$. Remove 5 from S.
- Step 3: Code tail is $(4,1,7)$. Vertices in $S\setminus\{3,5\}$ not in the tail are $\{2,6\}$. Smallest is $b_3=2$. Edge: $\{2,4\}$. Remove 2 from S.
- Step 4: Code tail is $(1,7)$. Vertices in $S\setminus\{3,5,2\}$ not in the tail are $\{4,6\}$. Smallest is $b_4=4$. Edge: $\{4,1\}$. Remove 4 from S.
- Step 5: Code tail is (7). Vertices in $S\setminus\{3,5,2,4\}$ not in the tail are $\{1,6\}$. Smallest is $b_5=1$. Edge: $\{1,7\}$. Remove 1 from S.

The remaining vertices in S are $\{6,7\}$. The final edge is $\{6,7\}$. The reconstructed tree has edges $\{\{3,2\},\{5,2\},\{2,4\},\{4,1\},\{1,7\},\{6,7\}\}$.

Proof of Cayley's Formula. The encoding and decoding procedures are inverses of each other. This establishes a bijection between the set of labeled trees on n vertices and the set of sequences of length $n-2$ with elements from $\{1,\ldots,n\}$. The number of such sequences is n^{n-2}. ■

Section 7 shows a proof of the Matrix-Tree theorem via eigenvalues. See [GR13] & [Gil10]

7 The Eigenvalue Connection: A Second Proof

We now present a second, powerful proof for the number of spanning trees, this time using the eigenvalues of the Laplacian matrix. This approach provides a beautiful link between the combinatorial structure of a graph and the algebraic properties of its matrix representation. We will prove the following theorem.

Theorem 7.1 (Matrix-Tree Theorem, Eigenvalue Form)**.** *Let G be a connected graph with n vertices and Laplacian matrix L. Let $\lambda_1, \lambda_2, \dots, \lambda_{n-1}$ be the non-zero eigenvalues of L. The number of spanning trees is given by:*

$$\tau(G) = \frac{1}{n} \prod_{i=1}^{n-1} \lambda_i$$

The proof hinges on analyzing the characteristic polynomial of the Laplacian, $P(\lambda) = \det(\lambda I - L)$, from two different viewpoints.
By definition, the roots of the characteristic polynomial are the eigenvalues of the matrix. If the eigenvalues of L are $\lambda_1, \lambda_2, \dots, \lambda_n$, we can write the polynomial in a factored form:

$$P(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2) \cdots (\lambda - \lambda_n) = \prod_{i=1}^{n} (\lambda - \lambda_i)$$

From Proposition 4.15, we know that for a connected graph, exactly one eigenvalue is 0. Let's set $\lambda_n = 0$. The expression becomes:

$$P(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2) \cdots (\lambda - \lambda_{n-1}) \cdot (\lambda - 0) = \lambda \prod_{i=1}^{n-1} (\lambda - \lambda_i)$$

Our goal is to find the coefficient of the λ^1 term in this polynomial. Let's expand the product part.

$$\prod_{i=1}^{n-1} (\lambda - \lambda_i) = \lambda^{n-1} - \left(\sum_{i=1}^{n-1} \lambda_i \right) \lambda^{n-2} + \cdots + (-1)^{n-1} \prod_{i=1}^{n-1} \lambda_i$$

The last term shown in fuchsia is the constant term (the term with λ^0) of this expansion. When we multiply the entire expression by the leading λ, this constant term becomes the coefficient of our target λ^1 term.

$$P(\lambda) = \lambda \left(\lambda^{n-1} - \cdots + (-1)^{n-1} \prod_{i=1}^{n-1} \lambda_i \right) = \lambda^n - \cdots + \left((-1)^{n-1} \prod_{i=1}^{n-1} \lambda_i \right) \lambda$$

So, from the eigenvalue perspective, the coefficient of λ is $(-1)^{n-1} \prod_{i=1}^{n-1} \lambda_i$.
There is a general formula for the characteristic polynomial of any $n \times n$ matrix M, where the coefficients are given by the sums of its principal minors.

$$\det(\lambda I - M) = \lambda^n - c_1 \lambda^{n-1} + c_2 \lambda^{n-2} - \cdots + (-1)^{n-1} c_{n-1} \lambda + (-1)^n c_n$$

where c_k is the sum of all $k \times k$ principal minors of M. A $k \times k$ principal minor is the determinant of a sub-matrix formed by selecting k rows and the *same* k columns.
We are interested in the coefficient of the λ^1 term, which is $(-1)^{n-1} c_{n-1}$. The term c_{n-1} is the sum of all $(n-1) \times (n-1)$ principal minors. An $(n-1) \times (n-1)$ principal minor of L is $\det(L_{ii})$, where L_{ii} is the matrix L with row i and column i removed.

$$c_{n-1} = \sum_{i=1}^{n} \det(L_{ii})$$

So, from this perspective, the coefficient of λ is $(-1)^{n-1} \sum_{i=1}^{n} \det(L_{ii})$.

Section 7 shows a proof of the Matrix-Tree theorem via eigenvalues. See [GR13] & [Gil10]

Now we have found two different expressions for the same coefficient of the characteristic polynomial. By equating them, we can build our final proof.

$$\underbrace{(-1)^{n-1}\prod_{i=1}^{n-1}\lambda_i}_{\text{From Eigenvalues}} = \underbrace{(-1)^{n-1}\sum_{i=1}^{n}\det(L_{ii})}_{\text{From Minors}}$$

$$\prod_{i=1}^{n-1}\lambda_i = \sum_{i=1}^{n}\det(L_{ii})$$

Now, we use the main result of the Matrix-Tree Theorem, which we proved in the previous section using the Cauchy-Binet formula. It states that all the co-factors $\det(L_{ii})$ are equal to the number of spanning trees, $\tau(G)$. By substituting in the right-hand side and some easy manipulation we get that

$$\det(L_{11}) = \det(L_{22}) = \cdots = \det(L_{nn}) = \tau(G)$$

$$\sum_{i=1}^{n}\det(L_{ii}) = \sum_{i=1}^{n}\tau(G) = n\cdot\tau(G)$$

$$\prod_{i=1}^{n-1}\lambda_i = n\cdot\tau(G)$$

$$\tau(G) = \frac{1}{n}\prod_{i=1}^{n-1}\lambda_i$$

Completing our colored proof. ■

8 More Proofs of the Matrix Tree Theorem

Combinatorial Proof. By the Leibniz formula, the determinant of the $(n-1)\times(n-1)$ matrix L_{nn} is:

$$\det(L_{nn}) = \sum_{\sigma\in S_{n-1}}\operatorname{sgn}(\sigma)\prod_{i=1}^{n-1}(L_{nn})_{i,\sigma(i)}$$

where the sum is over all permutations σ of the set $\{1,2,\ldots,n-1\}$. The entries of L_{nn} are given by the Laplacian definition:

$$(L_{nn})_{ij} = L_{ij} = \begin{cases}\deg(v_i) & \text{if } i=j\\ -1 & \text{if } i\neq j \text{ and } (v_i,v_j)\in E\\ 0 & \text{otherwise}\end{cases}$$

We can interpret each term in the determinant sum as a collection of signed, directed graphs. A single term corresponds to a permutation σ. From this σ, we construct a set of functional directed graphs, denoted $\mathcal{D}(\sigma)$. Each graph $D\in\mathcal{D}(\sigma)$ has vertex set V and $n-1$ directed edges, one originating from each non-root vertex $\{v_1,\ldots,v_{n-1}\}$. The edges are chosen as follows:

- For each i where $\sigma(i)=i$ (a fixed point), we select the matrix entry $L_{ii}=\deg(v_i)$. We draw a directed edge from v_i to one of its $\deg(v_i)$ neighbors in G.
- For each i where $\sigma(i)\neq i$, we select the matrix entry $L_{i,\sigma(i)}$, which is non-zero only if it is -1. We must draw the directed edge $(v_i,v_{\sigma(i)})$.

This section shows other proofs of the Matrix-Tree theorem. See [GR13], [Lin19b], & [Ric13]

Let S be the set of all such functional digraphs obtainable from all permutations. The value of the determinant is the sum of the signs of all these graphs:

$$\det(L_{nn}) = \sum_{D \in S} \operatorname{sign}(D)$$

where the sign of a graph D derived from permutation σ is defined as $\operatorname{sign}(D) = \operatorname{sgn}(\sigma) \cdot (-1)^k$, with k being the number of edges (v_i, v_j) where $i \neq j$.
Every vertex v_i for $i < n$ has an out-degree of 1, and v_n has an out-degree of 0. Therefore, any path starting from a non-root vertex must either terminate at the root v_n or enter a directed cycle. Thus, every graph $D \in S$ consists of a set of disjoint cycles on vertices $\{v_1, \ldots, v_{n-1}\}$ and a set of directed paths forming a tree rooted at v_n.
Let $S_C \subset S$ be the subset of graphs containing at least one cycle. We define an involution $\phi : S_C \to S_C$ that pairs up graphs so their signs cancel. For any $D \in S_C$:

1. Identify the directed cycle C in D that contains the vertex v_i with the smallest index i.
2. Define $\phi(D) = D'$ to be the graph identical to D, but with the direction of every edge in C reversed.

The map ϕ is an involution because applying it twice restores the original graph: $\phi(\phi(D)) = D$. We now show that $\operatorname{sign}(D') = -\operatorname{sign}(D)$.
Let the cycle C have length m. Reversing its edges corresponds to changing the underlying permutation σ to a new permutation σ' by composition with an m-cycle. This multiplies the permutation sign by $(-1)^{m-1}$. Additionally, reversing m edges means we are choosing m different off-diagonal entries, which multiplies the $(-1)^k$ part of the sign by $(-1)^m$. The total change in sign is:

$$\Delta(\text{sign}) = (-1)^{m-1} \cdot (-1)^m = -1$$

Thus, $\operatorname{sign}(D') = -\operatorname{sign}(D)$. Every graph in S_C is paired with another of opposite sign, so their total contribution to the sum is zero:

$$\sum_{D \in S_C} \operatorname{sign}(D) = 0$$

The only graphs in S that are not in S_C are those with no cycles. A functional digraph on V with $n-1$ edges and no cycles must be a spanning tree where all paths are directed toward the root v_n. Let S_T be this set of cycle-free graphs. The determinant is now simply:

$$\det(L_{nn}) = \sum_{D \in S_T} \operatorname{sign}(D)$$

For any spanning tree $D \in S_T$, all of its edges must be of the form (v_i, v_j) where v_j is a neighbor of v_i, corresponding to diagonal entries L_{ii} in the determinant product. This implies:

- The underlying permutation must be the identity permutation ($\sigma(i) = i$ for all i), for which $\operatorname{sgn}(\sigma) = +1$.
- No off-diagonal entries are used, so the factor $(-1)^k$ is $(-1)^0 = +1$.

Therefore, the sign of every surviving spanning tree is $\operatorname{sign}(D) = (+1) \cdot (+1) = 1$. Then

$$\det(L_{nn}) = \sum_{D \in S_T} 1 = |S_T| = \tau(G)$$

This completes the proof. ■

This proof uses probability to count the number of spanning trees. The strategy is as follows. We use something called Wilson's Algorithm, which generates every possible spanning tree of a graph with equal probability, which means that the probability of generating a specific tree is $\mathbb{P}(\mathcal{T}) = 1/\tau(G)$. We will then calculate $\mathbb{P}(\mathcal{T})$ using a different method based on the properties of the random walks in the algorithm, and this will give us a second expression for the probability. By setting the two expressions equal to each other, we can solve for $\tau(G)$.

This section shows other proofs of the Matrix-Tree theorem. See [GR13], [Lin19b], & [Ric13]

Definition 8.1 (Simple Random Walk)**.** Imagine a person walking on the vertices of the graph. At each step, they move from their current vertex to one of its neighbors, choosing each neighbor with equal probability. This process is a simple random walk. We can describe this with a matrix P, where P_{ij} is the probability of moving from vertex v_i to v_j in one step.

Definition 8.2 (Wilson's Algorithm)**.** This algorithm builds a random spanning tree step-by-step. We first pick any vertex v to be the root, the start of our tree, then we pick any vertex x that is not yet in the tree. After that, we start a random walk from x and keep walking until we hit any vertex that is already part of the tree. The path we just walked might have cycles but we only care about the direct path, so we erase any loops. This direct path is called a Loop-Erased Random Walk (LERW). Finally, we add this new, simple path to your tree and go back picking a new vertex not yet in the tree until all vertices are included. The final result is a spanning tree.

Random Walks Proof. Because Wilson's algorithm is uniform, we have

$$\mathbb{P}(\mathcal{T}) = \frac{1}{\text{Total number of spanning trees}} = \frac{1}{\tau(G)}$$

Now for the clever part, the probability of generating the tree $\mathcal{T}$ is the product of the probabilities of generating each of its branches in a specific order, and the probability of generating a single branch (a LERW) depends on the choices made at each step of the random walk.
Imagine the walk is at a vertex x; the probability of it moving to a specific neighbor y is $1/\deg(x)$. However, the walk might loop around before committing to a path. The mathematics of these loops and paths can get complicated, involving concepts like the Green's function. However, a remarkable thing happens when we multiply all these step-by-step probabilities together for the entire process of building the tree $\mathcal{T}$. All the complex intermediate terms cancel each other out in a beautiful way.
The probability of Wilson's algorithm, starting with root v, generating the specific tree $\mathcal{T}$ simplifies to

$$\mathbb{P}(\mathcal{T}) = \frac{1}{\det(L_{ij})}$$

where L_{ij} is the Laplacian matrix with the row and column for the root vertex v removed. We now have two different but equal expressions for the same probability:

$$\frac{1}{\tau(G)} = \frac{1}{\det(L_{ij})}$$

$$\tau(G) = \det(L_{ij})$$

Since our choice of the root vertex v was arbitrary, this remarkable formula holds for any vertex in the graph. This completes our proof. ■

9 Matrix Tree Theorem In other types of graphs

Till now, we have proved the Matrix Tree theorem for Simple Graphs and complete graphs, but I still want to calculate the number of spanning trees in Complete bipartite graphs and weighted graphs.

Definition 9.1. Complete bipartite graphs A bipartite graph is divided into two sets A and B containing vertices n and m where each edge connects a vertex from A with a vertex from B, and because it is a complete graph, each vertex in A or B is connected to all the other vertices in the opposing set. You can see an example in Figure 9.1.

Theorem 9.2. *The Number of spanning trees in a Complete bipartite graph $K_{n,m}$ is $n^{m-1}m^{n-1}$. I have a habit of trying to see if the theorem works, so we are expecting the number of spanning trees to be* 32000*. I am sure we are both lazy to count* 32000 *trees, so let's prove it. It's easier.*

This section 9 counts the number of spanning trees for other types of graphs. See [Oca11]

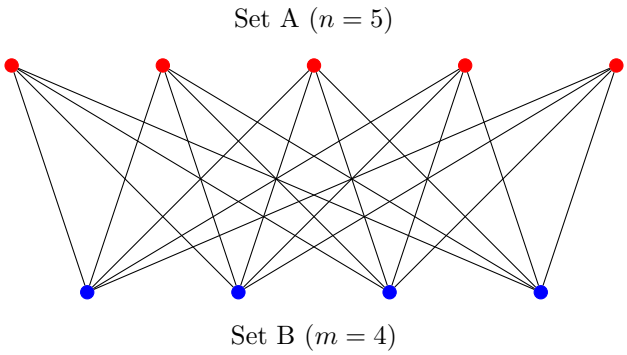


Figure 9.1: A complete bipartite graph $K_{5,4}$.

Proof. We use the Matrix Tree Theorem, which states that the number of spanning trees $t(G)$ for a graph with k vertices is given by the product of its non-zero Laplacian eigenvalues (λ_i), divided by k

$$t(G) = \frac{1}{k}\prod_{i=1}^{k-1}\lambda_i$$

For $K_{n,m}$, the number of vertices is $k = n + m$.
The product of all these non-zero eigenvalues, $\prod \lambda_i$, is the product of $(n + m)$ appearing once, n appearing $m - 1$ times, and m appearing $n - 1$ times. This can be written out explicitly as:

$$(n+m) \times \underbrace{(n \times n \times \cdots \times n)}_{m-1 \text{ times}} \times \underbrace{(m \times m \times \cdots \times m)}_{n-1 \text{ times}}$$

$$\prod_{i=1}^{n+m-1}\lambda_i = (n+m)^1 \cdot n^{m-1} \cdot m^{n-1}$$

By substituting this product back into the theorem's formula

$$t(K_{n,m}) = \frac{1}{n+m}\left((n+m)\cdot n^{m-1}\cdot m^{n-1}\right)$$

$$t(K_{n,m}) = n^{m-1}m^{n-1}$$

■

Definition 9.3. Weighted graphs $G = (V, E, w)$ where V is a set of vertices, E is a set of edges connecting pairs of vertices, and w is a weight function assigning a real-valued weight to each edge is a weighted graph.

Conjecture 9.4. *The Number of spanning trees in the weighted graph shown in Figure 9.2 with a weighted edge of 5 is five times the number of spanning trees a non-weighted version has. This is the first thing that came to my mind.*

Now let's prove or disprove it.

Proof. The Matrix Tree Theorem states that the number of spanning trees is any co-factor of the graph's Laplacian matrix, L. The Laplacian is defined as $L = D - A$.

$$L_{weighted} = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 4 & -1 & -1 \\ 0 & -1 & -1 & 2 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{pmatrix} = L_{unweighted}$$

This section 9 counts the number of spanning trees for other types of graphs. See [Oca11]

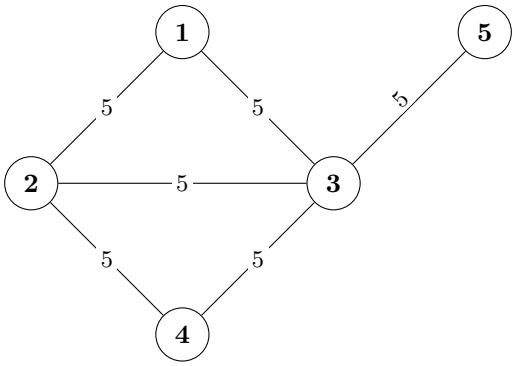


Figure 9.2: A weighted Graph

Both D and A depend only on the graph's structure. Adding weights does not change the degree of any vertex or the existence of any edge. Thus, the Laplacian matrix L is identical for both the unweighted graph and its weighted version and since the Laplacian matrix does not change, the number of spanning trees calculated from it cannot change either. This formally proves that the number of spanning trees is independent of uniform edge weights. Disproving our proposition ■

10 Algorithms Comparison

The Matrix-Tree Theorem provides the theoretical basis for several distinct algorithms to compute the number of spanning trees, $\tau(G)$. The choice of method depends on the graph's properties and the computational context. This section provides a detailed comparison, followed by pseudo-code for each primary algorithm.

Table 10.1: Detailed comparison of algorithms for counting spanning trees.

Method	**Core Mathematical Idea**	**Use Case & Remarks**	**Complexity**
Matrix Determinant	The number of spanning trees equals any co-factor of the Laplacian matrix, L.	This is the most practical algorithm for general graphs. It is computationally efficient and numerically stable, with complexity determined by the determinant calculation (e.g., using LU decomposition).	$O(n^3)$
Eigenvalue Method	The number of spanning trees is the normalized product of the non-zero eigenvalues of L.	Theoretically insightful but computationally less efficient. While having the same big-O complexity, eigenvalue algorithms often have larger constant factors and can be less stable than determinant calculations. Best for graphs with known spectra.	$O(n^3)$
Cayley's Formula	A direct combinatorial result from bijection like Prüfer sequences.	Extremely fast but highly specialized. This is not a general algorithm; it applies exclusively to the complete graph K_n. Its speed comes from being a closed-form formula.	$O(\log n)$

Method 1: The Determinant Algorithm

This method is the standard for computational software. It directly implements the co-factor form of the Matrix-Tree Theorem.

This section 10 basically compress time complexities. For more see [Gee25]

```
function CountTreesDeterminant(Graph G):
  // G has n vertices and m edges
  n = G.vertex_count()
  if n == 0: return 0

  // Construct the n x n Laplacian matrix L
  L = construct_laplacian(G)

  // Create an (n-1) x (n-1) submatrix
  L_sub = remove_row_and_col(L, index=0)

  // The result is the determinant of this submatrix
  return determinant(L_sub)
```

Listing 1: Pseudocode for the Determinant Algorithm

Explanation: The algorithm first constructs the Laplacian 'L'. It then forms a sub-matrix by removing the first row and column (any index would work). The final step is to compute the determinant of this smaller matrix, which directly yields $\tau(G)$.

Method 2: The Eigenvalue Algorithm

This algorithm implements the eigenvalue form of the theorem. It is less common in practice due to higher computational overhead and potential numerical precision issues.

```
function CountTreesEigenvalue(Graph G):
  // G has n vertices and m edges
  n = G.vertex_count()
  if n <= 1: return n

  // Construct the n x n Laplacian matrix L
  L = construct_laplacian(G)

  // Compute all eigenvalues of L
  eigenvalues = compute_all_eigenvalues(L)

  // Multiply all non-zero eigenvalues
  product = 1.0
  for λ in eigenvalues:
    if abs(λ) > 1e-9:  // Check if λ is non-zero
      product = product * λ

  // Normalize the product and round to the nearest integer
  return round(product / n)
```

Listing 2: Pseudocode for the Eigenvalue Algorithm

Explanation: After constructing the Laplacian, this algorithm finds all its eigenvalues. It then iterates through them, multiplying all values that are not close to zero (to account for floating-point inaccuracies). The final product is normalized by dividing by 'n'.

11 Electrical Networks

The origins of the Matrix-Tree Theorem are deeply connected to the analysis of electrical circuits. Gustav Kirchhoff's 1847 paper established the fundamental laws governing current and voltage distribution and

Section 11 introduces Electric networks with a connection between graph theory. See [Kir03], & [Ric13]

these laws, when expressed in the language of linear algebra, reveal a connection between physics and graph theory. To formalize this, consider a network (graph) with n nodes and m edges. We arbitrarily assign a direction to each edge.

Theorem 11.1 (Kirchhoff's First Law - The Current Law (KCL))**.** *The sum of currents entering any node from the network, plus any external current supplied to that node, is zero.*

Kirchhoff's First Law - The Current Law (KCL)

This law describes the conservation of charge as we can represent it using the incidence matrix, I, of the graph. For a graph with n nodes and m edges, I is an $m \times n$ matrix where

$$I_{ev} = \begin{cases} +1 & \text{if edge } e \text{ starts at node } v \text{ (flows away)} \\ -1 & \text{if edge } e \text{ ends at node } v \text{ (flows towards)} \\ 0 & \text{otherwise} \end{cases}$$

We let $\boldsymbol{i}$ be the $m \times 1$ vector of currents on each edge and $\boldsymbol{f}$ be the $n \times 1$ vector of external currents supplied to each node. KCL is then stated

$$I^T \boldsymbol{i} = \boldsymbol{f}$$

The matrix-vector product $I^T \boldsymbol{i}$ calculates the net current flowing away from each node and for a closed system with no external sources ($\boldsymbol{f} = \boldsymbol{0}$), the equation becomes $I^T \boldsymbol{i} = \boldsymbol{0}$. This means the vector of physically possible currents, $\boldsymbol{i}$, must lie in the null space of I^T.

Theorem 11.2 (Kirchhoff's Second Law - The Voltage Law (KVL))**.** *For any closed loop which is a cycle in the language if graphs in the network, the sum of the potential differences across the edges of the loop is zero.*

Kirchhoff's Second Law - The Voltage Law (KVL)

This law implies that the node potentials are well-defined. Let $\boldsymbol{x}$ be the $n \times 1$ vector of potentials at the nodes. The potential difference (voltage) $\boldsymbol{p}_e$ across an edge e from node u to node v is $p_e = x_u - x_v$. This relationship for all edges can be captured in a single matrix equation:

$$\boldsymbol{p} = I\boldsymbol{x}$$

Here, the incidence matrix I acts as a discrete **gradient operator** (∇), mapping node potentials (a 0-form) to edge potential differences (a 1-form). KVL is the statement that the vector of potential differences, $\boldsymbol{p}$, must be in the **column space of the incidence matrix, Col**(I). A vector is in $\text{Col}(I)$ if and only if the sum of its components around any cycle is zero. This space is the orthogonal complement of the cycle space. Therefore, KVL is the discrete analogue of the vector calculus identity $\text{curl}(\text{grad}(f)) = \boldsymbol{0}$.

Theorem 11.3 (Ohm's Law)**.** *The current $\boldsymbol{i}_e$ on an edge e is directly proportional to the potential difference $\boldsymbol{p}_e$ across it, mediated by the edge's conductance c_e (where $c_e = 1/r_e$ and r_e is the resistance).*

$$i_e = c_e p_e$$

Ohm's Law

This provides a link between voltage and current. We can express this for the entire network using a diagonal $m \times m$ conductance matrix, C, where $C_{ee} = c_e$. The matrix form of Ohm's Law is

$$\boldsymbol{i} = C\boldsymbol{p}$$

By combining the three laws, we can solve for the node potentials $\boldsymbol{x}$ from the external currents $\boldsymbol{f}$.

1. $I^T \boldsymbol{i} = \boldsymbol{f}$ (KCL)

Section 11 introduces Electric networks with a connection between graph theory. See [Kir03], & [Ric13]

2. $\boldsymbol{i} = C\boldsymbol{p}$ (Ohm's Law)
3. $\boldsymbol{p} = I\boldsymbol{x}$ (KVL satisfied by definition)

Substituting (3) into (2) gives us $\boldsymbol{i} = C(I\boldsymbol{x})$. Substituting this into (1) gives the fundamental equation of network analysis

$$(I^T CI)\boldsymbol{x} = \boldsymbol{f}$$

$I^T CI$ is the graph's Laplacian matrix.

Kirchhoff's 1847 Spanning Tree Formula

The most remarkable part of Kirchhoff's original paper was his general solution for the currents. He found that the solution for every current in the network shared a common denominator. He calculated this denominator, D, by stating it is the sum of all expressions of the form $c_1 c_2 \cdots c_{n-1}$, where the edges corresponding to the conductances c_i form a spanning tree of the network.
This is the Matrix-Tree Theorem! Decades before it was formalized with modern linear algebra, Kirchhoff discovered that the key to solving the entire system was a combinatorial sum over all spanning trees. The quantity he calculated by hand is exactly what we now compute as the determinant of any co-factor of the Graph Laplacian matrix L.

$$\det(L_{\text{co-factor}}) = \sum_{T \in \text{spanning trees}} \prod_{e \in T} c_e$$

This determinant represents the total conductance of the network and is fundamental to determining the effective resistance between any two nodes and as the number of spanning trees goes higher, the more redundant the network is, bearing more errors with less loss.

Acknowledgments

I would love to thank Dr. Simon Rubestensien-Salzedo & Dr. Dean Menezes for their mentorship, attending Euler Circle, revising, and writing this paper.

References

[AZ99] Martin Aigner and Günter M. Ziegler. *Proofs from The Book*. Springer, 1999.

[Gee25] Geeksforgeeks. *Total number of Spanning Trees in a Graph*. Jan. 2025. URL: https://www.geeksforgeeks.org/dsa/total-number-spanning-trees-graph/ (visited on 07/08/2025).

[Gil10] Gilbert Stang. *Linear Algebra*. OCW. Available at: http://ocw.mit.edu/18-06S05. Accessed: 2025-07-08. 2010.

[GR13] Chris Godsil and Gordon F Royle. *Algebraic Graph Theory*. Vol. 207. Springer Science & Business Media, 2013.

[Hug09] Jacob Hughes. *Electrical Networks—A Graph Theoretical Approach*. 2009.

[Kir03] Gustav Kirchhoff. "On the solution of the equations obtained from the investigation of the linear distribution of galvanic currents". In: *IRE transactions on circuit theory* 5.1 (2003), pp. 4–7.

[Lin19a] Andrew Lin. *Algebraic Combinatorics*. Apr. 2019. URL: https://ocw.mit.edu/courses/18-212-algebraic-combinatorics-spring-2019/1c947fa02a84f4538bdd3caf95e67ee5_MIT18_212S19_lec26.pdf (visited on 07/08/2025).

[Lin19b] Andrew Lin. *The Matrix-Tree Theorem: Connecting graphs with matrices*. 2019. URL: https://web.stanford.edu/~lindrew/matrixtree.pdf (visited on 07/08/2025).

[Oca11] Evans Doe Ocansey. *The Matrix-Tree Theorem*. Essay, African Institute for Mathematical Sciences. 2011.

[Ope14] MIT OpenCourseWare. *Combinatorial Analysis: The Matrix-Tree Theorem*. 2014. URL: https://ocw.mit.edu/courses/18-314-combinatorial-analysis-fall-2014/2724112ea36679f82dc04f0b2f4f355e_MIT18_314F14_mt.pdf (visited on 07/08/2025).

[Ric13] Larissa Richards. *A Random Walk Proof of Matrix Tree Theorem*. 2013. URL: https://uregina.ca/~kozdron/Research/UgradTalks/KRS_MTT.pdf (visited on 07/08/2025).

[Rub23] Simon Rubinstein-Salzedo. *Transition to Proofs*. World Scientific, 2023.

[Seb21] Sebastian Cioaba. *Combinatorics 1 Lectures*. YouTube. Available at: https://youtube.com/playlist?list=PLiR7fd-rXGgjurtwz5Rw7tKuUPYrvpi8M&si=QahSlHjb_E7FBYFH. Accessed: 2025-07-08. Nov. 2021.

[Sid18] Aaron Sidford. *Lecture 10 - Spectral Graph Theory*. Feb. 2018. URL: https://web.stanford.edu/class/cme305/Files/l10.pdf (visited on 07/08/2025).

[Ste23] Steve Butler. *(Spectral; Fall 23) 16 - Matrix Tree Theorem*. YouTube. Available at: https://youtu.be/AcVPX2JOM1U?si=qTFPXG_nGAuqxor8. Accessed: 2025-07-08. Oct. 2023.

[Wra19] Wrath of Math. *Graph Theory*. Youtube. Available at: https://youtube.com/playlist?list=PLztBpqftvzxXBhbYxoaZJmnZF6AUQr1mH&si=LFUjqW__e08q74Bn. Accessed: 2025-07-08. 2019.