Spanning Trees from complete graphs, enumerating them, and their effect on Networks

Sonny Alcius

April 2024

Graph Theory Fundamentals

Graph theory, a fundamental area of discrete mathematics, concerns the study of graphs—structures consisting of nodes (or vertices) connected by edges. This theory is instrumental in analyzing various types of networks, whether social, computational, or biological, providing insights into connectivity, flow, and traversal within these networks. A key concept within this framework is that of a spanning tree, which is a subset of a graph that includes all the vertices connected by the fewest possible number of edges, forming a tree structure. Spanning trees are crucial for the efficient design and analysis of networks because they ensure maximum connectivity with minimal complexity, facilitating operations like network traversal and optimization without redundancy. Some basic formulas within graph theory, particularly ones that will be used throughout this paper, include:

1. The degree of a vertex in an undirected graph is the number of edges incident to it. For a graph $G = (V, E)$ with vertices V and edges E, the degree of vertex v is denoted as:

$$\deg(v).$$

2. In any undirected graph, the sum of the degrees of all vertices is twice the number of edges. This can be expressed as:

$$\sum_{v \in V} \deg(v) = 2|E|.$$

This lemma is useful for proving properties about the number of vertices of a given degree within a graph.

3. A complete graph is one in which there is exactly one edge between every pair of vertices. For a complete graph with n vertices, the number of edges is given by:

$$\frac{n(n-1)}{2}.$$

An example of a complete graph is below:

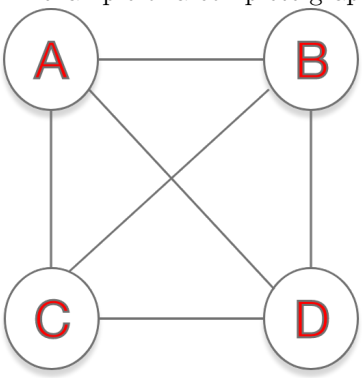

Figure 1: *Complete graph example*

We know there are 4 vertices (A, B, C, D), so applying the formula we get:

$$\frac{4(4-1)}{2} = \frac{12}{2} = 6$$

Trees:

Before introducing spanning trees, we first define what a tree is. First discussed by Arthur Cayley in 1857 while working on finding recursive formulas for counting the number of trees having finite vertices, a tree is a type of subgraph with 3 attributes: connection (there being a path between any 2 vertices), ascyclicness (there being a lack of a cycle, implying that you cannot return to a vertice after already transversing it), and the calculation of the number of edges differs from that of a traditional graph: $n-1)\}$ edges.. An outline of a basic tree can be visualized in the infographic below:

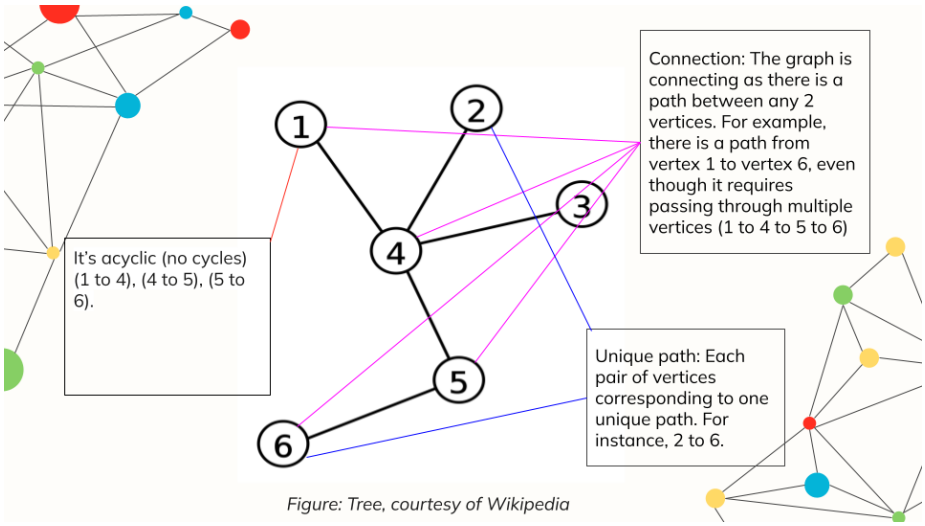

Figure 2: *A tree, courtesy of Wikipedia.*

Spanning Trees:

Continuing from the foundation of graph theory and its fundamental concepts, I now turn our focus more towards spanning trees. Spanning trees were first applied by Otakar Borůvka, a Czech mathematician who developed the Borůvka's algorithm, one of the earliest algorithms for finding a minimum spanning tree in a graph, specifically for electrical grid planning in the city of Moravia in 1026. A spanning tree of a graph is a subgraph that, for the most part, inherits the majority of its properties from trees. It must contain the minimum number of edges possible to ensure connection among all vertices (calculated once again through the formula $n - 1$), it must be ascylic, have a unique path - but it has one key differentiator: it must be all inclusive of vertices from the graph that it spans.

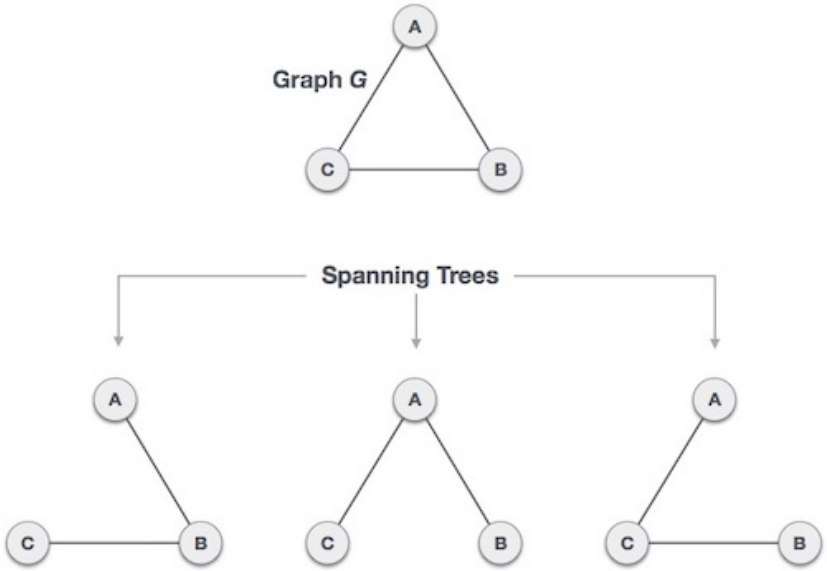

Figure 3: *The image illustrates a connected graph G with vertices A, B, and C, and its three possible spanning trees, each omitting one edge to prevent cycles and maintain connectivity.*

We can see in the image that a connected graph G with vertices A, B, and C, and its three possible spanning trees, each omitting one edge to prevent cycles and maintain connectivity, with each spanning tree being inclusive of all vertices from Graph G. This is in contrast with general trees, which, despite coming from the same graph, does not have to be inclusive of all the vertices.

Let's apply these concepts to a real life scenario: Suppose We have network of 9 cities, each represented by a vertice, with roads that sometimes loop back (implying a cyclic nature). Multiple paths exist between some cities, like between C and G: B-D-G-B and C-F-G-B. Each edge in the graph denotes a direct communication link (fiber optic cables for internet service, for instance) between two cities. While this network offers multiple paths between any two cities, which can be advantageous for redundancy, it also means higher costs for installation and maintenance. Each link needs to be established and maintained, and the overall cost increases with the number of links.

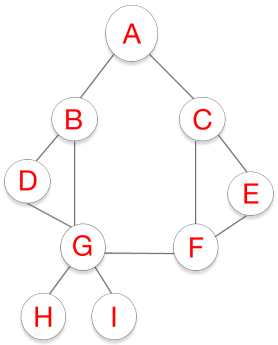

Figure 4: *Complete graph of a network of 9 cities*

For the same set of cities from the original graph, a spanning tree would connect all cities without forming any cycles, there'd be a unique path between any 2 cities, and there would be 8 edges for all 9 cities, the minimum number needed to ensure connection among them all. So, with this spanning tree, each city is still reachable from every other city, but with a smaller amount of edges, and via unique paths that may pass through other cities. This reduces the amount of direct connections that need to be maintained. This effectively means that there's greater cost efficiency, as fewer links means less infrastructure to maintain, which lowers the ongoing operational costs for the network provider in this city.

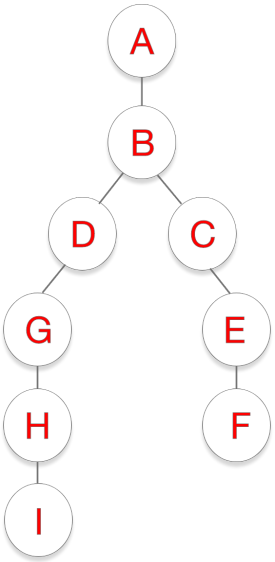

Figure 5: *Spanning tree of the complete graph of a network of 9 cities.*

Following from the example above, it becomes apparent why panning trees are particularly significant in network design and optimization because they repre-

sent the minimal connection necessary to ensure that all nodes are reachable without redundancy. This makes spanning trees essential for minimizing redundancy while maximizing efficiency in data structures, network design, and algorithm development.

The enumeration of spanning trees is a crucial aspect in understanding the versatility and capability of a network's infrastructure - and in particular, refers to the process of counting all the different ways in which a spanning tree can be formed from a graph. It is important because it provides insights on optimal network designs, and network reliability. Take for instance the 9 city scenario we've evaluated - the spanning tree derived from the original graph was just one of many possible spanning trees, and the higher amount of spanning trees a network has, the more alternative paths exist for data to travel through in case the failure of some connections. The graph of the network of cities has many spanning trees (as we'll soon learn how to calculate), which means there are many alternative paths for the fiber optic cables for internet service to travel through, ensuring reliability always. To this end, enumerating all possible spanning trees of a graph not only provides insight into the robustness and reliability of the network but also allows us to gauge the complexity inherent in network topologies.

The most popular formula for enumerating spanning trees in graph is known as Cayley's formula, a combinatorial technique.

Enumeration Technique:

Cayley's formula:

For a complete graph with n labeled vertices, Cayley's formula gives the number of spanning trees directly as $T_n = n^{n-2}$, where T_n is the number of spanning trees and n is the number of vertices in the graph.

Proof of Cayley's Formula:

Theorem: *The number of spanning trees in a complete graph K_n with n vertices is n^{n-2}.*

Before presenting the proof, we define a Prüfer code. For a labeled tree with n vertices, a Prüfer code is a sequence of $n-2$ integers where each integer corresponds to a vertex label from the set $\{1, 2, \ldots, n\}$. The Prüfer code for a tree is constructed by sequentially removing the leaf with the smallest label and recording the label of its neighbor until only two vertices remain.

Establishing the Bijection:

We assert that there is a bijection between the set of all spanning trees on n labeled vertices and the set of all possible Prüfer codes of length $n-2$. This bijection works as follows:

- **From Tree to Prüfer Code:** Given a tree, the corresponding Prüfer code is generated by repeatedly removing the smallest labeled leaf and noting the label of its neighbor.
- **From Prüfer Code to Tree:** Conversely, a tree can be reconstructed uniquely from a given Prüfer code by replenishing the vertices not present in the remaining code as leaves in each step, attaching them to the vertices as indicated by the sequence, until the tree is fully reconstructed.

Inductive Justification for Uniqueness:

To rigorously demonstrate that each Prüfer code corresponds to a unique tree, consider the process of reconstructing the tree from a Prüfer code using mathematical induction:

Base Case: Assume $n = 3$, the smallest case for a meaningful Prüfer code, which would have a length of 1. The Prüfer code has only one number, corresponding to one of the vertices. The only two vertices not in the code are connected to this vertex, forming a unique tree (a star tree with the central node being the one in the Prüfer code).

Inductive Step: Assume that for a Prüfer code of length k (where $k \geq 1$), corresponding to a tree with $k + 2$ vertices, the tree reconstructed from the code is unique. Now consider a Prüfer code of length $k + 1$. The process to reconstruct the tree involves:

- Adding one vertex to the tree structure determined by the Prüfer code of length k. This vertex is the smallest numbered vertex not present in the remaining Prüfer code, and it is attached to the vertex corresponding to the next number in the code.
- The position and connection of this new vertex are uniquely determined by the structure already built from the Prüfer code of length k and the next number in the sequence of the Prüfer code of length $k + 1$.

Thus, if the tree corresponding to a Prüfer code of length k is unique, then so is the tree corresponding to a Prüfer code of length $k + 1$, as the added vertex and its connection depend solely on the defined next step in the Prüfer code and cannot vary.

By the principle of mathematical induction, we conclude that for all $n \geq 3$, a Prüfer code of length $n - 2$ uniquely determines a labeled tree with n vertices, confirming that each Prüfer code corresponds to a unique tree.

Counting Prüfer Codes:

Each element of a Prüfer code is an integer from 1 to n, with no restrictions other than its length, $n - 2$. Therefore, the total number of such sequences (Prüfer codes) is n^{n-2} because there are n choices for each position in the sequence and the length of the sequence is $n - 2$.

To conclude, by the established bijection, the number of different Prüfer codes exactly equals the number of spanning trees in the complete graph K_n. Since there are n^{n-2} different Prüfer codes, it follows that there are n^{n-2} spanning trees in K_n.

Applications

Now that we have established the proof for Cayley's formula through Prufers code, we examine the applications of this formula for enumerating trees in graphs, and the effects that it has on networks.

Let's take for example social media platforms like Facebook, which rely on a network of data centers to provide seamless user experiences. These data centers need to be interconnected with redundancy to ensure that users can still access the platform even if some network links fail. A complete graph of this model is below.

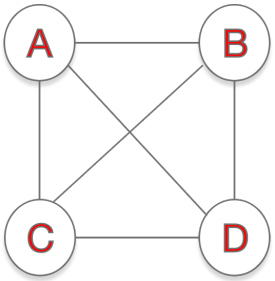

Figure 6: *Complete graph to model 4 data centers of Facebook*

In this example, let's say our vertices, n, are 4 data centers responsible for handling data across the Facebook network. Let's say the edges are the network links, which are AB,AC,AD,BC,BD, CD. This graph is also cyclic, so it allows multiple paths for data transmission between any 2 data centers.

Calculating the number of spanning trees in this graph, we get:

$$T_n = n^{n-2}$$

$$T_5 = 4^{4-2} = 4^2 = 16$$

Thus, the number of spanning trees in the graph is 16.

By ensuring all vertices are included, all have a connection with at least another, and all have a unique path, we can extrapolate many spanning trees from this one graph. 3 are visualized below:

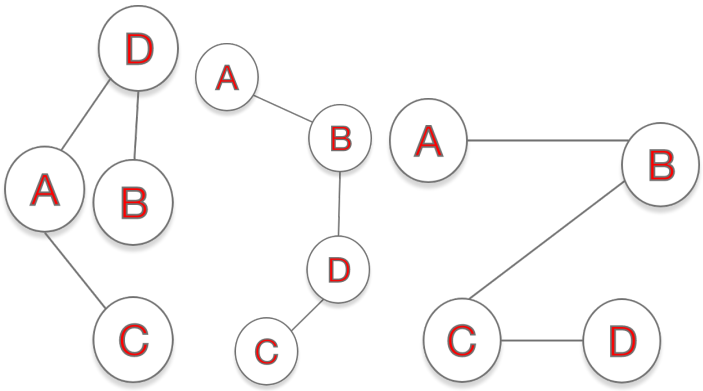

Figure 7*: Spanning tree of Figure 6*

From the option of spanning trees available, we'll take the first and observe its implications on the network.

This spanning tree reduces, or at least maintains redundancy at a manageable level by ensuring unique paths between any pair of data centers. This allows for selective redundancy without introducing network loops.

Following from this selective redundancy, because of its acyclic nature (no network loops), which ensures a unique path between each pair of data centers while maintaining connectivity, the tree reduces the risk of network problems, like broadcast storms and link saturations/network congestions.

Oncemore, the original complete graph has

$$\frac{n(n-1)}{2}$$

links, which could lead to high network overhead. Spanning trees, alternatively, only have n-1 links, which minimizes the number of links required to connect all data centers, which greatly reduces infrastructure cost, and enables more straightforward routing. This increases cost efficiency, and helps Facebook save money on its data infrastructural management - and with hundreds of millions of users, that's a lot of money.
†

‡

[†]In a network loop, broadcast frames (which are intended to reach all network devices) continue to circulate and be replicated, causing excessive traffic that overwhelms the network. This situation, known as a broadcast storm, can consume substantial network bandwidth and lead to network slowdowns or failures.

[‡]Loops can lead to link saturation where the network bandwidth is fully consumed by unnecessary traffic, preventing legitimate network communications from occurring. This results in significant network congestion and latency, impacting the performance of critical applications and services.

Conclusion

In summary, the investigation of spanning trees and their enumeration within the framework of graph theory has uncovered significant insights that extend far beyond the abstract mathematical realm. Through the application of Kirchhoff's Matrix-Tree Theorem to ecological networks, we have demonstrated the theorem's utility in quantifying the resilience of ecosystems. The ability to enumerate all possible spanning trees in a network is not just a theoretical exercise, but rather a crucial analytical tool for understanding the stability and robustness of ecological systems. This approach provides a vital foundation for the development of effective conservation strategies, enabling ecologists to pinpoint the key interactions necessary for maintaining biodiversity and ecosystem health.

Furthermore, the principles elucidated through graph theory have wide-ranging applications in diverse fields such as telecommunications, urban planning, and epidemiology. These applications highlight the importance of spanning trees in optimizing network design, ensuring uninterrupted connectivity, and enhancing the resilience of infrastructural systems. As we face increasingly complex challenges in managing and preserving interconnected systems, from digital networks to natural habitats, the role of graph theory becomes more critical than ever.

Bibliography:

1. T Kyle Petersen, Inquiry Based Enumerative Combinatorics, 2019, in reference to graph theory fundamentals discussed in Graph theory section.

2. Diestel, Reinhard. Graph Theory. 5th ed., Springer, 2017, in reference to trees and spanning trees, their corollaries (rules/properties), and the Cayley formula.

3. Newman, M. E. J. Networks: An Introduction. Oxford University Press, 2010, in reference to the network application under Applications and the concepts of network infrastructure, network reliability, and network optimization.

4. Harary, F. (n.d.). http://deepblue.lib.umich.edu— request PDF, in reference to Arthur Cayleys discovery of trees.

5. Otakar Boruvka on minimum spanning tree problem. (n.d.). http://www.cmat.edu.uy/ marclan/TAG/Sellanes/boruvka.pdf, in reference to the Otakar Boruvka's application of spanning trees and the development of the Boruvka algorithm.

6. Libretexts. (2022, March 20). 5.6: Counting labeled trees. Mathematics LibreTexts. https://math.libretexts.org/Bookshelves/CombinatoricsandDiscrete$_M$*athematics* in reference to constructing the Cayley formula proof using Prufers codes.