IV: Section (V) Spanning Trees, Enumeration, Optimization & Kruskal's Algorithm.

Next we focus our attention on the problem of spanning trees in a graph or networks. We begin with the problem of enumerating all (spanning) Trees in a graph, and then we discuss a solution to the "minimum spanning tree" problem using a famous algorithm designed by Kruskal (1956).

In relation to complex networks, one can imagine the solution to the min. spanning Tree problem as presenting a "cheapest" (i.e. most efficient) connected subgraph of a digraph G without using redundancy (i.e. no cycles).

To this end, suppose that one wants to build a transportation network connecting n given cities, in such a way that a passenger travels from one city to the next with maximum efficiency (i.e. minimal total mileage). This is an instance of the min. spanning tree problem.

By the same token, a great variety of analogous problems - e.g. finding a min. connector set for internet servers, or, less

obviously: applications in data clustering & automatic speech recognition (To name but a few examples) - can all be solved efficiently with Kruskal's Algorithm (or related results such as Prim's).

We begin with Two basic enumeration formulas.

(1) Q: How many distinct (simple) graphs (up To isomorphism) are possible for a set of n vertices?

A: A Total of $2^{\binom{n}{2}}$ such graphs are possible.

Proof: In Total, There are $\binom{n}{2}$ possible possible edges for a simple graph on n vertices (Think of K_n being an extreme case). Of These edges, a given graph 6 either includes Re edge or Re edge is excluded, giving rise To Two possibilities for each edge.

(2) Q: For a vertex set of size n, how many distinct spanning trees (up To isomorphism) exist?

A: There are a Total of $\boxed{n^{n-2}}$ spanning trees on n vertices (Cayley).

Proof: One way To prove This result is by way of Mathematical induction. Alternatively, however - and perhaps more intuitively -

one can think of the expression: n^{n-2} as representing the number of ways to form a list of length $(n-2)$ using the vertex set of size n.

With some work, it is not hard to show that for a given tree on n vertices (with the vertices labelled), each distinct tree can be encoded (through a "leaf-pruning" algorithm) as a unique string of length $(n-2)$ using the vertex set. This code is called a "Prüfer Code", in graph/network theory. Because there exists a bijection between each (distinct) tree & its so-called Prüfer code, this proves the result.

Two caveats are needed with respect to the previous result.

First, note that the expression: n^{n-2} grows super-exponentially in the size of the vertex set, n. Most obviously, this indicates that because the number of trees grows rapidly with respect to n, the problem of finding a minimum spanning tree for a given graph is far from trivial.

Second, observe that the n^{n-2} expression counts the number of trees in a complete graph (i.e. K_n). Because of the high degree of symmetry present in K_n, it is consequently unreasonable

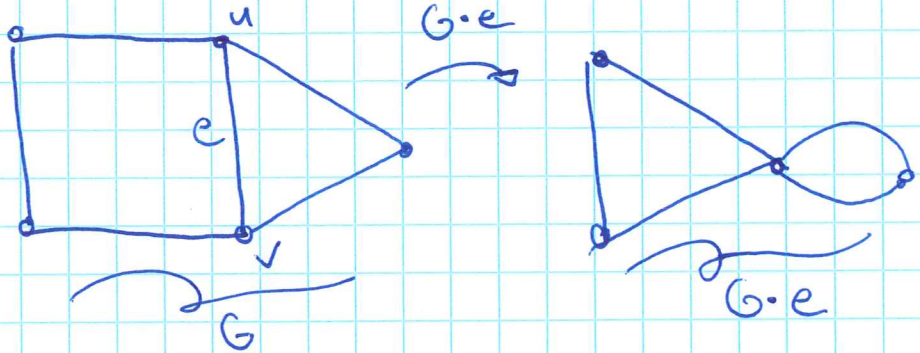
To expect that a tree-counting formula for a generic graph G will be quite this "clean." Next we develop such a formula, using the aforementioned adjacency matrix of a graph. This beautiful result is known as Kirchhoff's Theorem.

Prior to our exposition of Kirchhoff's Theorem, it is useful to divulge an important result about counting spanning trees. The following result should help us "see" the validity of K's Theorem in a deeper way.

Notation: Let $\boxed{t(G)}$ denote the number of spanning trees for a graph, G.

Let: $\boxed{G-e}$ indicate the subgraph of G induced by the deletion of edge "e." Furthermore, let $\boxed{G \cdot e}$ connote the "contraction" of edge e in G. By contraction, we mean the replacement of the vertices incident with edge e by a single vertex.

$\boxed{G \cdot e}$: Contraction of edge e in G

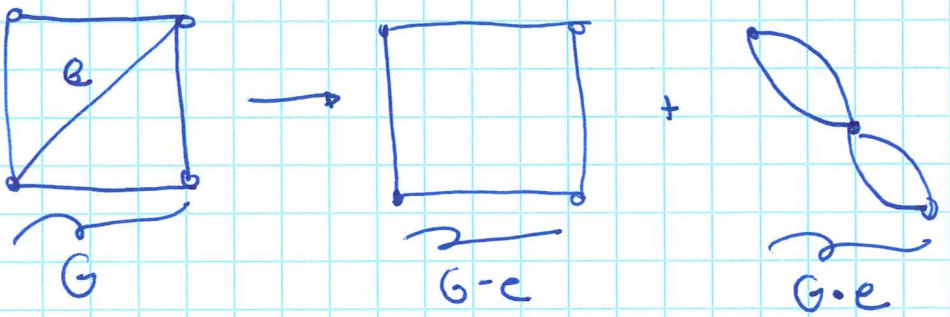


Fact: If $e \in E(G)$ is not a loop, Then:

$$\boxed{\tau(G) = \tau(G-e) + \tau(G \cdot e)}.$$

Why? Note That $\tau(G-e)$ counts The number of spanning trees of G That omit e, while $\tau(G \cdot e)$ counts The number of spanning trees of G That contain e. The result necessarily follows.

Ex.



The reader may confirm for G & e above, $\tau(G) = \tau(G-e) + \tau(G \cdot e)$, $8 = 4 + 4$.

observe The recursive nature of The formula for $\tau(G)$ given above, as This recursion will form a basis for Kirchoff's Theorem.

Def. Define $D(G)$ as The degree matrix of The graph G, so That $D(G)$ is diagonal, with $v_{ii} = d(v_i)$ for $1 \le i \le n$ & zero otherwise.

Def. Define $L(G)$ as The Laplacian Matrix of The graph G, where $\boxed{L(G) = D(G) - A(G)}$ (where $D(G)$: degree matrix, $A(G)$: adjacency matrix).

Theorem Kirchhoff's Theorem (also called: Matrix Tree Theorem):

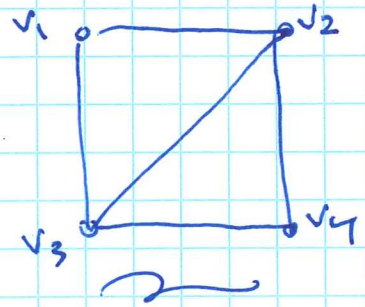
Let G be a loopless graph, & Q^* be a matrix obtained by deleting row: s & column: t of $L(G)$ (The Laplacian Matrix of G).

Then, $\tau(G) = (-1)^{s+t} \det(Q^*)$.

Proof (sketch): why does this work?

It can be shown that: $L(G)$ factors as: $L(G) = M^T M$, where M is the incidence matrix of G. Removing a row + column from M is tantamount to removing an edge & vertex (via contraction) from G, whereupon $Q^* = (M^T)^* M^*$. Inductively, then, since $\tau(G) = \tau(G-e) + \tau(G \cdot e)$, this shows that $\tau(G) = \pm \det(Q^*)$.

Ex. We use Kirchhoff's Theorem to find $\tau(G)$ for G as follows.



G

$$D(G) = \begin{matrix} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} \begin{matrix} \begin{matrix} v_1 & v_2 & v_3 & v_4 \end{matrix} \\ \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \end{matrix}, \quad A(G) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

($D(G)$: degree matrix)

$$L(G) = D - A = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

remove col: 2, row: 2

$$Q^* = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

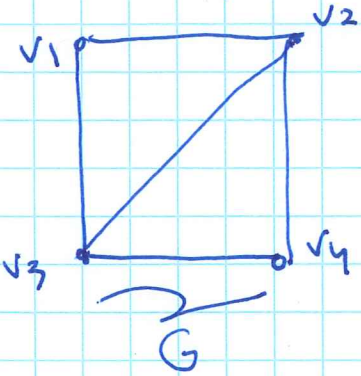
$(s = t = 2)$

$$(-1)^{2+2} \det(Q^*) = 1 \cdot \begin{vmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{vmatrix}$$

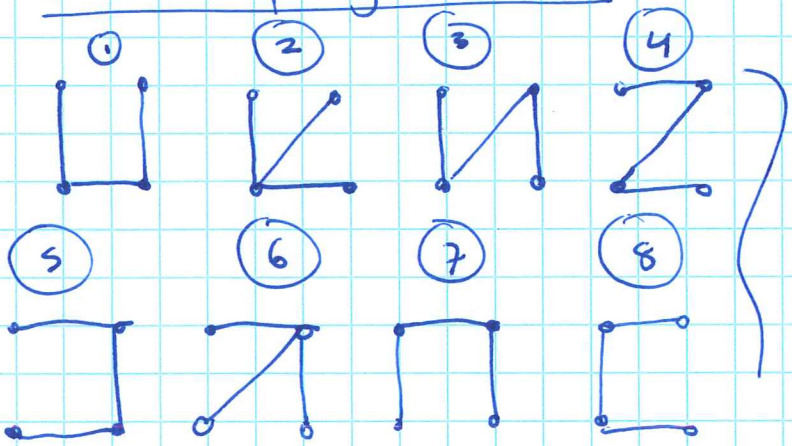
$$= 2 \cdot \begin{vmatrix} 3 & -1 \\ -1 & 2 \end{vmatrix} - (-1) \begin{vmatrix} -1 & -1 \\ 0 & 2 \end{vmatrix} = 2 \cdot (6-1) + (-2-0)$$

$$= \boxed{8} = \tau(G).$$

We verify the result $\tau(G) = 8$, directly.



Distinct spanning trees of G:



confirmed!

Note that since the computational complexity of computing the determinant of an $n \times n$ matrix is $O(n^3)$, Kirchhoff's Theorem provides us with a relatively efficient means to count the number of distinct spanning trees for any loopless graph.

Next we present another computationally efficient algorithm - this time as a solution to the minimum spanning tree problem - known as Kruskal's Algorithm.

Kruskal's Algorithm: Let G be a weighted graph, where $w(e) \in \mathbb{R}$. Choose an edge e_1 with minimal weight.

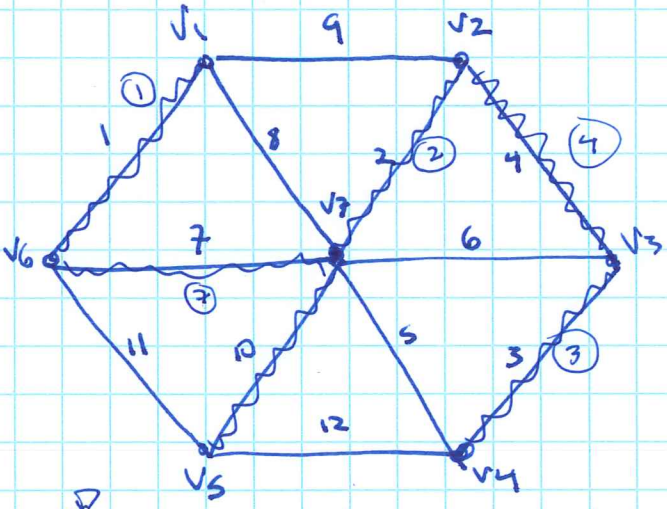
(A) Suppose that edges $E_k = \{e_1, \ldots, e_k\}$ have been chosen so far. Choose a next edge e_{k+1} from $E(G) \setminus E_k$ such that the following two conditions are met:

(1) The induced subgraph $G_{k+1} = G[\{e_1, e_2, \ldots, e_{k+1}\}]$ is acyclic. (Note that G_{k+1} need not be connected).

(2) The weight $w(e_{k+1})$ is minimal, i.e. for all $e \in E(G) \setminus E_k$, we know that $w(e) \geq w(e_{k+1})$.

(3) Stop when there are no more edges to select in the previous step.

Ex. We iterate Kruskal's Algorithm for the following weighted graph, G.



As the reader may check, after (10) steps Kruskal's Algorithm produces a minimum spanning tree.

Step	edge	weight	comment
(1)	v_1v_6	1	added
(2)	v_2v_7	2	added
(3)	v_3v_4	3	added
(4)	v_2v_3	4	added
(5)	v_4v_7	5	Rejected - creates cycle
(6)	v_3v_7	6	Rejected - cycle
(7)	v_6v_7	7	added
(8)	v_1v_7	8	Rejected - cycle
(9)	v_1v_2	9	Rejected - cycle
(10)	v_5v_7	10	Accepted

→ Halt: $|V(T)| = 6 = n-1$ ✓