Math 3322: Graph Theory[1] Mikhail Lavrov

Lecture 12: Spanning trees

September 23, 2021 *Kennesaw State University*

1 Spanning trees

If G is a connected n-vertex graph, then it has a connected n-vertex subgraph which is a tree. This has a proof by the extremal principle: let H be some connected n-vertex subgraph of G which has as few edges as possible. Then every edge of H must be a bridge (or else deleting that edge would give a smaller connected subgraph) and therefore H must be a tree.

Such a subgraph is called a **spanning tree**. (In general, a subgraph is **spanning** if it includes all the vertices of G, but possibly not all the edges; the term "spanning tree" is by far the most common way this adjective gets used.)

There are some interesting algorithms involving minimum spanning trees, which we won't talk about. These appear when every edge of a graph G has an extra piece of information attached to it, representing the "cost" of that edge. We might want to find a spanning tree of G with the lowest total cost: this is the "cheapest" way to connect G.

Spanning trees are also useful theoretically: lots of results about connected graphs hold because we can prove them for the spanning tree. Here is an example: a theorem which appears much earlier in your textbook, and has a page-long proof.

Theorem 1.1. *If G is a connected graph with at least 2 vertices, then it has a vertex v such that $G - v$ is still connected.*

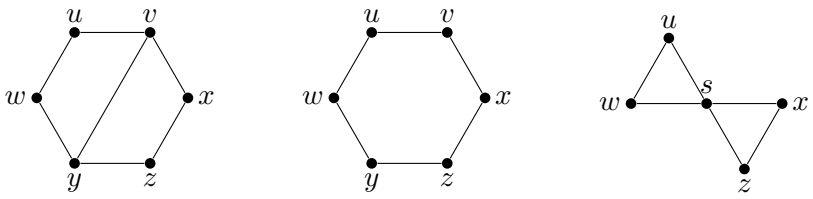
Proof. Let T be a spanning tree of G, and let v be a leaf of T. Then $T - v$ is also a tree, so it is connected. $T - v$ is a subgraph of $G - v$, so $G - v$ is also connected. □

Such a strategy works for any property which is never made false by adding more edges. If we can prove it for all trees, we automatically get it for all connected graphs. There's never a downside to doing this: if we were going to try to prove it for all connected graphs, our proof would have to work in particular for trees, anyway.

2 Counting the number of spanning trees

Our main question today is this: given a graph G, how many spanning trees does it have? Before jumping to the general answer, let's look at a special case. In the first graph below, how many spanning trees are there?

[1] This document comes from the Math 3322 course webpage: http://facultyweb.kennesaw.edu/mlavrov/courses/3322-fall-2021.php



This graph is small enough that we might be able to solve this problem by brute force—but also large enough that we probably wouldn't want to. To handle it systematically, let's split it up into cases based on the following: does our spanning tree use edge vy or not?

Case 1: spanning trees without vy. If we don't include edge vy, then we might as well delete it. All such spanning trees are also spanning trees of the second graph above: $G - vy$.

Out of the 6 edges in this graph, if we delete any one, we're left with a 5-edge tree isomorphic to P_6. So there are 6 spanning trees in this case.

Case 2: spanning edges with vy. If we include edge vy, then we must delete one of the edges yw, wu, or uv (otherwise there will be a cycle) and one of the edges vx, xz, or zy (otherwise there will be a cycle). Once we do both, there will be 5 edges left, so we'll have a tree. There are 9 spanning trees in this case.

Total: 15 trees.

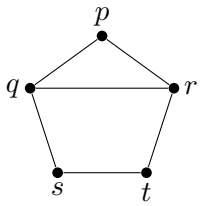
There's another way to think about case 2. Once we insist on using edge vy being used, then we have already made sure that v and y are in the same connected component. So it doesn't matter if the edge from another vertex like z goes to v or to y: that edge can connect it to both of them anyway.

As a result, we can keep track of everything we need to if we **contract** edge vy: replace it by a single vertex (called s in the third diagram above) that has an edge to every neighbor of v and every neighbor of y.

3 Deletion and contraction

Let's talk a bit more about contraction: the operation that takes graph G (leftmost graph) to graph $G \bullet vy$ (rightmost graph).

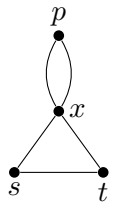
The definition we used to contract edge vy in the example *mostly* works. There's one more detail to watch out for. Let H be the graph below, and say we want to contract edge qr:



Our logic was: "it doesn't matter if an edge from another vertex like p goes to q or to r: once edge qr is part of our spanning tree, an edge from p to either vertex will connect it to both of them".

But what if there's an edge from p both to q and to r?

Although both edges are equally good, if our goal is to count spanning trees, we want to keep track of both of them. If we use edge pq and not pr, we get a different spanning tree, compared to if we use edge pr and not pq. So we keep them both, $H \bullet qr$ should be the **multigraph** shown below:



A multigraph is a graph in which we allow multiple edges to exist between the same pair of vertices. In general, we might also allow loops in a multigraph: an edge from a vertex to itself. However, in the case of spanning trees, loops will not be relevant: an edge from a vertex to itself should never be part of a tree.

We can count spanning trees in multigraphs in the same way as in ordinary, or **simple** graphs. In the multigraph $H \bullet qr$ above, to get a spanning tree:

- We must use one of the two edges between p and x. (We can't say "edge px", anymore: that's ambiguous.)
- Separately from that, we must keep two of the edges on the cycle (x, s, t, x) and delete the third.

So there are $2 \cdot 3 = 6$ ways to choose a spanning tree of $H \bullet qr$, which means there are 6 spanning trees of H that use edge qr. There are also 5 spanning trees of H that do not use edge qr: they are the 5 spanning trees of the 5-cycle $H - qr$. Altogether, H has 11 spanning trees.

To generalize our logic from the two examples above, let $\tau(G)$ denote the number of spanning trees in a connected graph G. Then:

Theorem 3.1 (Deletion-contraction formula for spanning trees)**.** *Let G be a connected graph (or connected multigraph) and let e be any edge of G. Then*

$$\tau(G) = \tau(G - e) + \tau(G \bullet e).$$

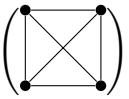
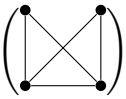
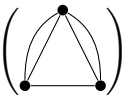
Keep in mind that if we contract edges in a multigraph, then parallel edges can accumulate! For example, suppose that u has 2 edges to v and 3 edges to w, and we contract edge vw. Then u should have $2 + 3 = 5$ edges to the new contracted vertex.

In the two examples we did, we applied this theorem once, and then used other logic. In general, we can try to find $\tau(G)$ by applying this theorem multiple times. The great thing about it is that both $G - e$ and $G \bullet e$ have fewer edges than G: the graphs become less and less complicated. So as we apply this theorem, we eventually get to graphs we can understand.

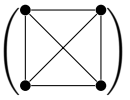
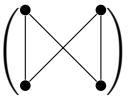
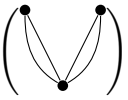
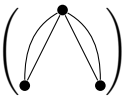
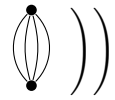
4 Complete graphs

Let's first do a complicated example: K_4.

If we apply the deletion-contraction formula to K_4, we get two graphs with 5 edges that still aren't easy to understand:

τ  $= \tau$  $+ \tau$ 

So let's do this again! In each of the graphs on the right-hand side (where we either deleted or contracted the top edge), let's also delete and contract the bottom edge. This gives us:

τ  $=$ $\left(\tau$  $+ \tau$  $\right) + \left(\tau$  $+ \tau$  $\right)$

Now we can look at each of the four resulting graphs.

- The first graph we got is isomorphic to C_4. Delete any edge of this graph, and you get a tree; there are 4 spanning trees.
- The second graph is already almost a tree: it just has doubled edges. For each adjacent pair of vertices, select one of the two edges to use, and you get a tree; there are $2 \cdot 2 = 4$ spanning trees.
- The third graph is isomorphic to the second, and also has 4 spanning trees.
- Finally, the last graph becomes a tree if we choose any one of the 4 parallel edges. There are also 4 spanning trees.

Altogether, $\tau(K_4) = 4 + 4 + 4 + 4 = 16$.

This would be a very frustrating calculation to do for K_5. But there is a formula:

Theorem 4.1 (Cayley's formula)**.** *The complete graph K_n has n^{n-2} spanning trees: $\tau(K_n) = n^{n-2}$.*

Cayley actually proved something stronger, and considerably weirder, than this formula. Suppose we take the expression

$$\begin{aligned}(v_1 + v_2 + v_3 + v_4)^2 v_1 v_2 v_3 v_4 = {} & v_1^3 v_2 v_3 v_4 + 2v_1^2 v_2 v_3 v_4^2 + 2v_1^2 v_2 v_3^2 v_4 + 2v_1^2 v_2^2 v_3 v_4 \\ & + v_1 v_2 v_3 v_4^3 + 2v_1 v_2 v_3^2 v_4^2 + 2v_1 v_2^2 v_3 v_4^2 \\ & + v_1 v_2 v_3^3 v_4 + 2v_1 v_2^2 v_3^2 v_4 + v_1 v_2^3 v_3 v_4.\end{aligned}$$

If we add up all the coefficients of all 10 terms, we get $16 = 4^2$, the number of spanning trees of K_n. That's just because "add up all the coefficients" is the same as "set $v_1 = v_2 = v_3 = v_4 = 1$", and doing that means the formula simplifies to $(1 + 1 + 1 + 1)^2 \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 4^2 = 16$.

But what's more, if you look any term like $2v_1^2 v_2^2 v_3 v_4$, the coefficient tells us the number of spanning trees of K_n in which v_1 has degree 2, v_2 has degree 2, v_3 has degree 1, and v_4 has degree 1. In general, the coefficient on each term tells us the number of spanning trees in which the degree (number of edges) of v_i in the tree equals the degree (power) of v_i in the term.

This works for any n, generalizing the formula to $(v_1 + v_2 + \cdots + v_n)^{n-2} v_1 v_2 \cdots v_n$.