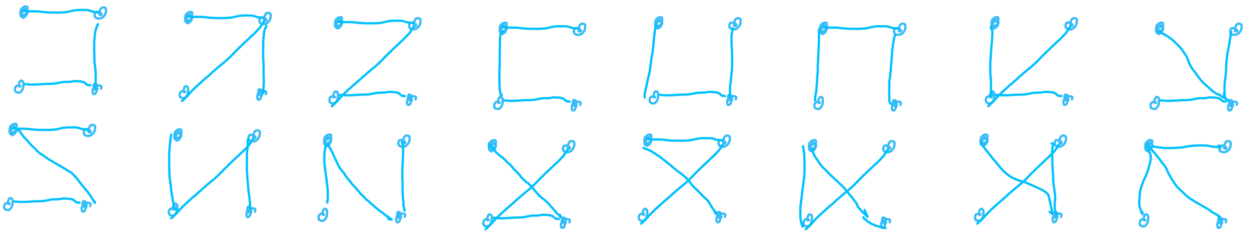
Math 38 – Graph Theory
Counting spanning trees

Nadia Lafrenière
04/20/2022

Question: How many spanning trees does the complete graph with n vertices have?



For the complete graph, there is an easy way of answering:

However, the following question is much harder:

Question: Given any graph G (simple or not), how many spanning trees are subgraphs of it?

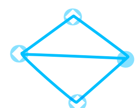
Finding a closed formula to count the number of spanning trees would be a lot to ask for trees that don't have a specific structure. Instead, we see an algorithm to answer this question easily.

Proposition
There are as many spanning trees in a graph G as in the graph obtained from G by deleting all its loops.

Proof

However, we cannot use the same strategy for multiple edges, as there can be more than one associated spanning tree. Here is an example:

Example: Count the number of spanning trees of the kite ($K_4 - e$)



4 trees passing through the outer cycle (of length 4).

4 trees passing through the diagonal, since we need to choose one edge from each triangle.

So there are 8 spanning trees in the kite.

To count the number of trees, there are two cases. These two cases span all the possibilities: either we use one specific edge or we don't use it. Of course, we cannot be in both situations at the same time. Combinatorially speaking, that means the total number of spanning trees is

#(spanning trees using that edge) + #(spanning trees not using it).

If the latter seems easy to count in general, the former needs the introduction of the following operation.

In a graph G, the contraction of the edge $e=uv$ is the replacement of both vertices u and v by a single vertex, by keeping all the edges incident to it, except e. The resulting graph, $G\cdot e$, has one fewer edge than G, and one fewer vertex.

Example

In the kite, the contraction of the central edge gives the following:

Proposition

The number of spanning trees of G, noted $\tau(G)$, satisfies, for any single edge e, $\tau(G)=\tau(G-e)+\tau(G\cdot e)$.

Proof

We already noted above that the total number of spanning trees is the sum of the trees with and without edge e. The thing we need to prove is that $\tau(G\cdot e)$ is the number of spanning trees using edge e.

This proposition will be the key to count, recursively, the number of spanning trees. We could also benefit from some shortcuts, like the following proposition:

Proposition

If G has no loop and does not have cycles of length at least 3, its number of spanning trees is the product of the multiplicities of the edges.

Proof

Corollary

If there are k edges $\{e_1, e_2, \dots, e_k\}$ between endpoints u and v in G, the number of spanning trees of G is given by

$$\tau(G - \{e_1, e_2, \dots, e_k\}) + k\tau(G \cdot e),$$

where $G \cdot e$ is the graph obtained by merging u and v and deleting $\{e_1, e_2, \dots, e_k\}$.

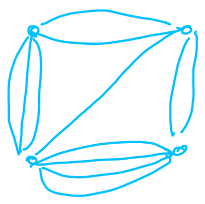
That yields an algorithm to count the number of spanning trees in G:

- If G is disconnected, it has no spanning tree; if G has a single vertex, it has only one spanning tree.
- Delete all loops in G.
- If G has no cycles of length at least 3:
  - The number of spanning trees is the product of the multiplicities of edges.
- Otherwise, choose a (multiple) edge e with multiplicity k, that is in a cycle of length at least 3. The number of spanning trees is $\tau(G-e) + k\,\tau(G \cdot e)$.

In the last step, $G-e$ has fewer cycles than G, and $G \cdot e$ has shorter cycles. That means that the algorithm eventually terminates.

Example

We count the spanning trees in the graph below:



Total number of spanning trees

$30+1(24+2(12+2\times 7)) = 106$ spanning trees

That process works for small trees, but the recursive procedure makes it very long to do for large connected graphs. We will see next class a theorem to make this computation efficient.

Reference: Douglas B. West. Introduction to graph theory, 2nd edition, 2001. Section 2.2