Trees, Caylees Theorem

Thomas Cohn

9/4/2018

Defn: A graph is the ordered pair $G = (V, E)$, where V is the set of vertices and E is the set of edges.

Defn: A graph is said to be connected if you cannot write $V = V_1 \sqcup V_2$ such that every pair of vertices $v_1 \in V_1$ and $v_2 \in V_2$ is not adjacent.

Defn: A tree is a connected graph with no cycles.

Defn: A forest is a collection of disjoint trees.

Defn: The degree of a vertex v, $\deg(v)$, is the number of edges incident to v.

Thm: $\sum_{v \in V} \deg(v) = 2|E|$
The reason for this should be obvious.

Defn: A vertex of degree 1 in a tree is called a leaf.

$[n]$ is a set of n labelled vertices. $C(n)$ is defined as the number of distinct trees on $[n]$.
For example, $C(2) = 1$, $C(3) = 3$, and $C(4) = 16$. Is there some sort of pattern? Perhaps even a formula?
Note that $C(n+1)$ is the number of rooted forests on $[n]$.
Thm: Cayley's Theorem
$C(n) = n^{n-2}$

Defn: A rooted tree is a tree on $[n]$ with a distinguished vertex (the root).

Defn: A rooted forest is a forest where every tree is a rooted tree.

$\overrightarrow{C}(n)$ is defined as the number of rooted trees on $[n]$. A tree with n vertexes could be made into n distinct rooted trees, depending on where the root is placed. So if Cayley's Theorem is true, we would expect $\overrightarrow{C}(n) = n \cdot n^{n-2} = n^{n-1}$.

Observe that if we have rooted forest $\overrightarrow{F}$, and we remove an edge $\overrightarrow{e}$, we get a rooted forest $\overrightarrow{F}'$ that has one more tree than $\overrightarrow{F}$.

Cayley's Theorem Proof 1: Double Counting
Let $F_{n,k} = \{\text{rooted } k\text{-forests on } [n]\}$. Thus, $F_{n,1} = \{\text{rooted trees on } [n]\}$.
Consider some $F_1 \in F_{n,1}$. We can remove an edge, and call this new rooted forest $F_2 \in F_{n,2}$. We can repeat this process all the way to $F_n \in F_{n,n}$. This will leave us with n vertices, and no edges connecting any of them; we can see that $|F_{n,n}| = 1$. We can also see that there are $(n-1)!$ possible ways to remove the edges from any F_1 to reach F_n.

But how many ways are there to add edges from $F_n \in F_{n,n}$ up to $F_1 \in F_{n,1}$? We can pick any two vertices in F_n, and the edge between them could face either direction, so there are $\binom{n}{2} \cdot 2 = n(n-1)$ ways to grow from $F_{n,n}$ to $F_{n,n-1}$. For growing from $F_{n,k}$ to $F_{n,k-1}$, we can choose any vertex, and chain an edge to it from any tree *other than the one it is a part of.* So there are $n(k-1)$ ways to grow from $F_{n,k}$ to $F_{n,k-1}$.

Therefore, we have $\prod_{k=n}^{2} n(k-1) = n^{n-1} \cdot (n-1)!$ ways to grow from $F_{n,n}$ to $F_{n,1}$. And we have $(n-1)!$ ways to remove the edges from each of the trees in $F_{n,1}$ (sending it back to $F_{n,n}$).
So we must have $\overrightarrow{C}(n) \cdot (n-1)! = n^{n-1} \cdot (n-1)!$, and if $\overrightarrow{C}(n) = n^{n-1}$, then $C(n) = n^{n-2}$. $\square$

Cayley's Theorem Proof 2: Prüfercode
We will look at a function $f : T \mapsto w \in [n]^{n-2}$, where w is obtained via a recursive process:

1. Select the smallest leaf in T, denoted v.
2. Look at the neighbor of v, denoted v'.
3. Put v' into w, then delete v from T, and return to step 1.

This process ends when there are only 2 vertices left.

Lemma: $v \in T$ is a leaf $\leftrightarrow v \notin w$

Proof: Assume that v is a leaf. Then we know if does not have a child pointing to it, so it could never be inserted into w as per our recursive algorithm. Thus, $v \notin w$.
Assume that v is not a leaf. Then we know that v has at least 2 neighbors. Since the algorithm terminates when there are only 2 vertices remaining, we know that at least one of the neighbors must be deleted, so $v \in w$. $\square$

Now, we must define the inverse function $g = f^{-1} : w \in [n]^{n-2} \mapsto T$ in order to obtain a bijection. Given $w = (w_1, w_2, \ldots, w_{n-2})$, define $v = \min\{[n] \setminus \{w_1, \ldots, w_{n-2}\}\}$. Then let T be the graph with vertices v and w_1 and a single edge connecting them, and let $w' = (w_2, \ldots, w_{n-2})$. We can then repeat this process recursively on w'.

There's probably more to come on Thursday's lecture.
I don't think we ever actually finished this in class.