Ethan Wong

Math HW #8

1. Number of non-isomorphic plane rooted trees with n edges is C_n, where C_n is the n^{th} Catalan number

WTS: C_n is also number of non-isomorphic plane rooted full binary trees with $2n$ edges

- By definition, a non-isomorphic plane rooted full binary tree is a specific type of rooted tree where each internal vertex has $2n$ edges. Trees always have 1 more vertex than it does edges, so a full binary tree with n internal vertices will have $2n+1$ vertices, $2n$ edges, and $n+1$ leaves.
- To show that C_n is also the number of non-isomorphic plane rooted full binary trees with $2n$ edges, we can use an argument similar to what was covered during class. Start off with the question: How many full binary trees are there with n internal vertices?

$n = 1 \rightarrow$ 1 possible full binary trees

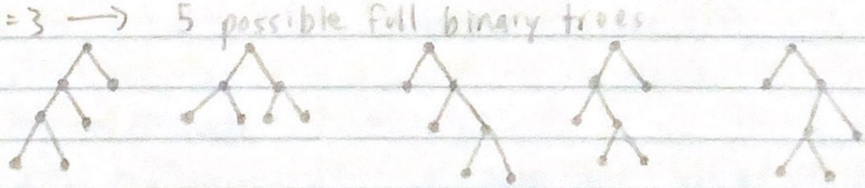
$C_1 = 1$ $\qquad$ C_1 = # of possible full binary trees ($n=1$)

$n = 2 \rightarrow$ 2 possible full binary trees

$C_2 = 2$

C_2 = # of possible full binary trees ($n=2$)

$n=3 \rightarrow$ 5 possible full binary trees



$C_3 = 5$

C_3 = # of possible full binary trees ($n=3$)

This pattern continues for $n=4$ (14 possible full binary trees, $C_4 = 14$), $n=5$ (42 possible full binary trees, $C_n = 42$) and so forth.

Note: When $n=0$, there is also one full binary tree – the single vertex. $C_0 = 1$, so the pattern holds for $n=0$ as well

9.8.12 A different approach:

Claim: The number C_n can be used to describe the number of binary trees with $n+1$ leaf nodes, or $2n+1$ total nodes.

Explanation:

$C_0 = 1$

$C_0, C_1, \dots, C_n$ represent the number of full binary trees with up to $n+1$ leaf nodes

We want C_{n+1}

When given a root node, we need k leaf nodes on one side and $n+1-k$ leaf nodes on the other, assuming k is a value between 1 and n. This yields C_k ways of choosing the tree for the first side and C_{n+1-k} on the second side. This means there are $C_k \cdot C_{n-k}$ trees for a given k.

This gives the following recurrence relation

$$C_0 = 1 \qquad C_{n+1} = \sum_{k=0}^{n} C_k C_{n-k}$$

The solution to this recurrence relation is the Catalan numbers C_n

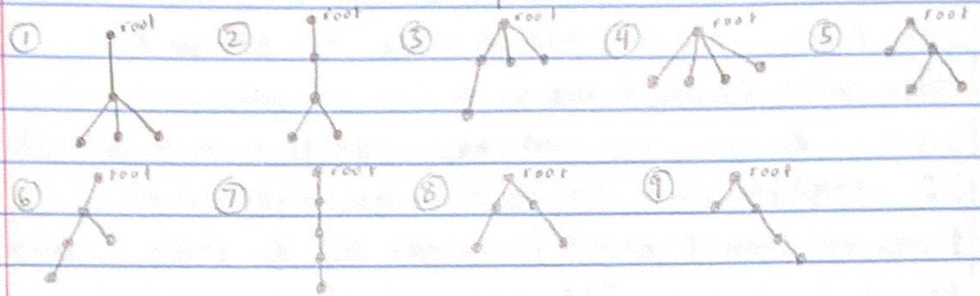
$$C_n = \frac{(2n)!}{(n+1)!\,n!}$$

This proves C_n is the number of non-isomorphic plane rooted full binary trees with $2n$ edges.

Ethan Wong

Math HW #8

2. There are 9 distinct non-isomorphic rooted trees with 5 vertices.

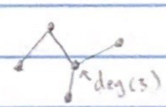


3 trees consist of a path on 5 vertices.
4 trees consist of a path on 4 vertices with one extra edge.
2 trees consist of a vertex with degree 4 surrounded by leaves.
All 9 of these rooted trees are non-isomorphic because they can not be redrawn to accurately represent the same data. It is impossible for any of these trees to preserve the same degree, number of levels, number of vertices per level. Simply put, swapping children of any of these trees does not yield any other tree, making them non-isomorphic.

3. There are 3 distinct non-isomorphic trees with 5 vertices. This can be determined by the maximum possible degree of a vertex. If the maximum degree of any vertex in the graph is only 2, that would mean the tree looks like this:

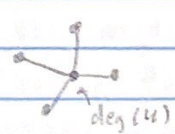
(ALL deg(2))

If the maximum degree of a vertex in the tree is 3, the fifth and final vertex would just be attached to a random leaf of the tree like this:



No matter where you put the fifth vertex, the result is the same.

Finally, if the maximum degree of a vertex in the tree is 4, the tree would look like this:



These 3 equivalence classes are all non-isomorphic by the same logic as in number two.

4. G : connected weighted graph   x : vertex in G

e : edge incident to x with minimal weight

WTS : e is contained in the minimal spanning tree of G

- By definition, a minimal spanning tree of G is a spanning tree (touches all vertices) T such that the weights of the edges of T is minimal among all spanning trees of G.
- It is easy to find a minimal spanning tree of a graph G by using Prim's algorithm. The concept is to add edges to the tree one by one. Each time we add an edge, we choose the edge with minimal weight that is incident to a vertex in the tree and to a vertex outside of the tree.
- When creating a minimal spanning tree for our given graph G, x must be added at some point. e must also be contained within the minimal spanning tree because e is the edge of minimal weight among all edges incident to x. According to Prim's algorithm, we should always choose the edges with minimal weight. e will be included in the minimal spanning tree in one of two ways:

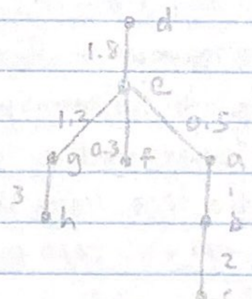
1. x will already by a vertex in the minimal spanning tree (i.e, it might be the starting vertex). The next edge to be added to the tree would be e, because it is the edge with minimal weight that is incident to a vertex in the tree (x) and to a vertex outside of the tree.
2. The vertex on the other side of e will already be in the minimal spanning tree. Then e will be added to the tree as the vertex connects to x. This is essentially the vice versa of above.

All in all, e will be contained in the minimal spanning tree because it is an edge with minimal weight between 2 vertices.

There may also be another scenario where x was added into the minimal spanning tree by another edge (Let's call this edge Z). This edge Z might be the only edge containing x that can be added to the tree at the moment. This edge Z would presumably weigh more than e. If x was added through Z, we could assume Z was the lowest weight edge currently incident to the tree. After adding Z, however, e would have to be added because there won't be any other incident edges to the current tree with a lesser weight than e. All of the other edges (if there are any) would have weights greater than Z and e.

No matter what scenario happens, e will eventually be added to the Minimal Spanning Tree since the vertex incident to e that isn't x will need to be added to the tree. Since e has the lowest weight connecting the two vertices, it must be part of the minimal spanning tree.

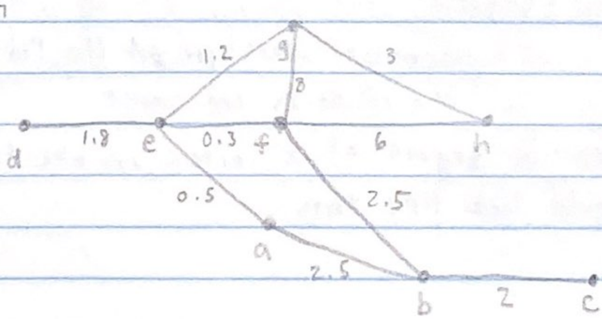
5. (i) Prim's Algorithm : We add edges to the tree one by one. Each time we add an edge, we choose the edge with minimal weight that is incident to a vertex in the tree and to a vertex outside of the tree. If we start investigating the minimum spanning tree at d, it must lead into e. e leads directtly to g, f, and a. The paths e -> g, e -> f, e -> a are all the lowest weight to reach these nonincluded vertices, so they should all be used. g -> h has the lowest weight to reach h. e -> f has the lowest weight to reach f. a -> b -> c has the lowest weights to reach b and c. Therefore all of these edges/paths should be included in the minimum spanning tree. Therefore by Prim's algorithm, our minimum spanning tree looks like this :



Weight of Minimum Spanning Tree : 9.8.

$1.8 + 1.2 + 0.3 + 0.5 + 3 + 1 + 2 = 9.8.$

(ii) G'



This graph G' is isomorphic to G because the degrees of the vertices and the overall weights are the same.
There are 2 minimal spanning trees as f -> b and a -> b have same weight.