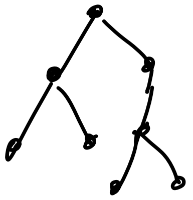
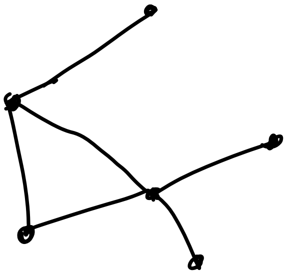
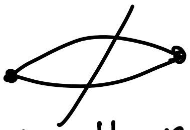
Counting Trees 8-31-22



Graph:

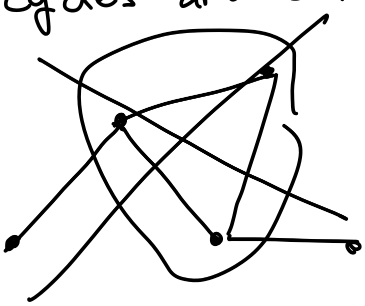


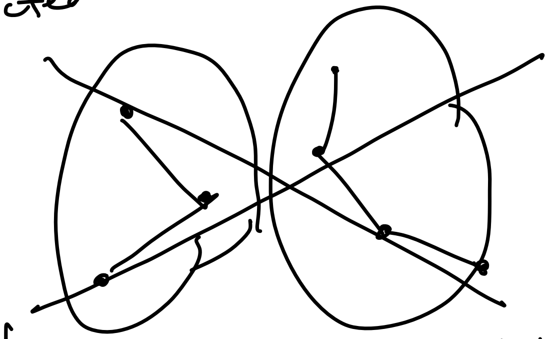
vertices connected by **edges** (no multiple edges for our purposes)



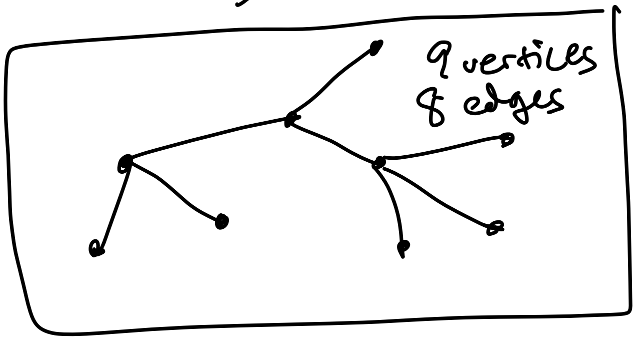
not allowed

Tree: no cycles and connected





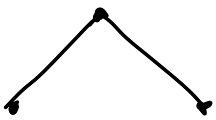
"forest" - multiple trees



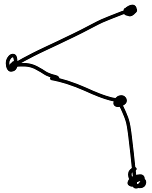
9 vertices
8 edges

If we have n vertices in a tree, how many edges?

n-1 edges



3 vertices
2 edges



4 vertices
3 edges

Pf: Induction.

Repeatedly remove a vertex and an edge to get a smaller tree.

Base case: 1 vertex, 0 edges

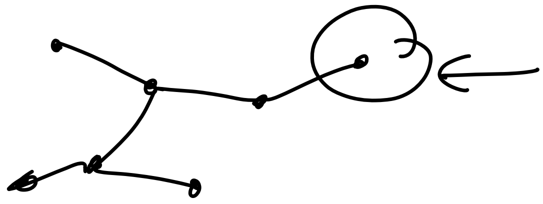
2 vertices, 1 edge

3 vertices, 2 edges

Inductive step: Assume statement holds for trees with $n-1$ vertices.

Can every tree with n vertices be formed by adding 1 vertex and 1 edge to a smaller tree? (Is there always a vertex and edge we can remove?)

Yes! Such a node is a leaf (meaning it has only one edge coming out /only one neighbor).

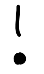


remove a leaf and corresponding edge

Question: We have n vertices labeled 1, 2, …, n.

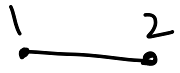
How many trees can we draw?

n=1:



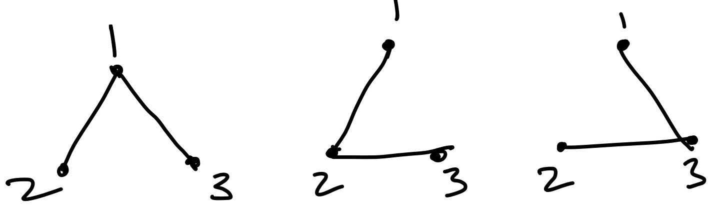
$\boxed{\text{1 tree}} = 1^{-1}$

n=2:



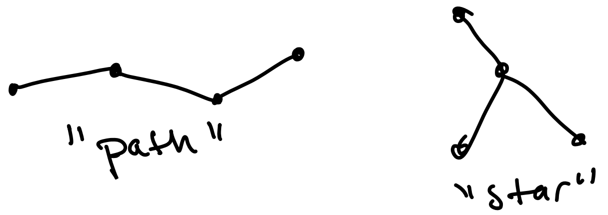
$\boxed{\text{1 tree}} = 2^{0}$

n=3:

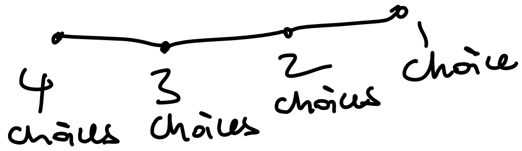


$\boxed{\text{3 trees}} = 3^{1}$

n=4:

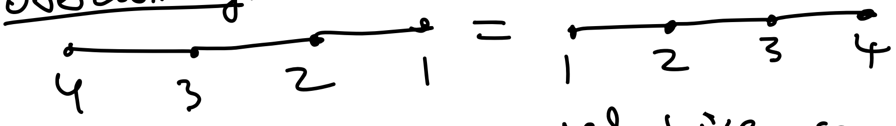


Paths:



4! orders

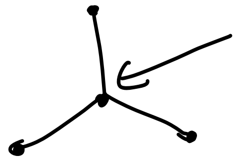
overcounting:



each tree has been counted twice, so

$$\frac{4!}{2} = 12 \text{ paths}$$

Stars:

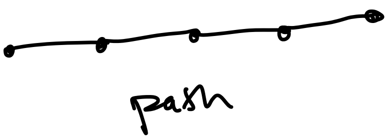


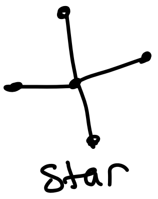
4 ways

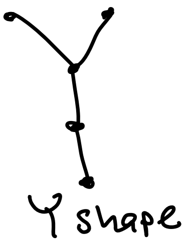
just need to choose middle vertex

$12 + 4 = \boxed{\text{16 trees}} = 4^{2}$

n=5:



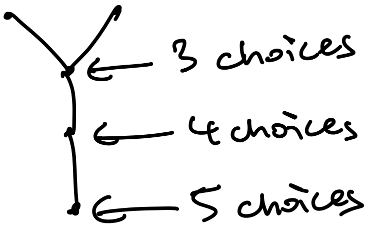




Paths: $\frac{5!}{2} = 60$ (same argument as before)

Stars: 5 (just choose middle vertex)

Y shapes:



$5 \cdot 4 \cdot 3 = 60$

$60 + 5 + 60 = \boxed{125 \text{ trees}} = 5^3$

Thm (Cayley's Tree Formula): The # of labeled trees on n vertices is n^{n-2}.

Induction? Hard to go from n^{n-2} to $(n-1)^{n+1}$

Degrees? Hard to check whether it's a tree from list of degrees...

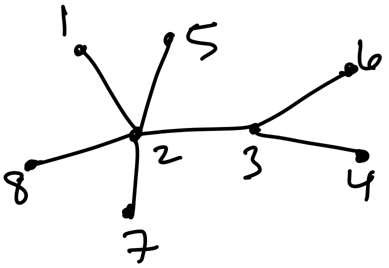
What does n^{n-2} count?

Sequences of length n-2
n choices for each term — say each term is a #
from 1 to n

Idea: Find one-to-one correspondence / bijection between trees and sequences of length n-2.

Given a tree, how do we construct the corresponding sequence?

Approach: remove vertices one leaf at a time, add one entry to the sequence each time



added to the sequence is the neighbor of the leaf we remove

(can't just use the # of the vertex removed because then all entries of the sequence would be unique — thus we'd have n! sequences, not n^{n-2})

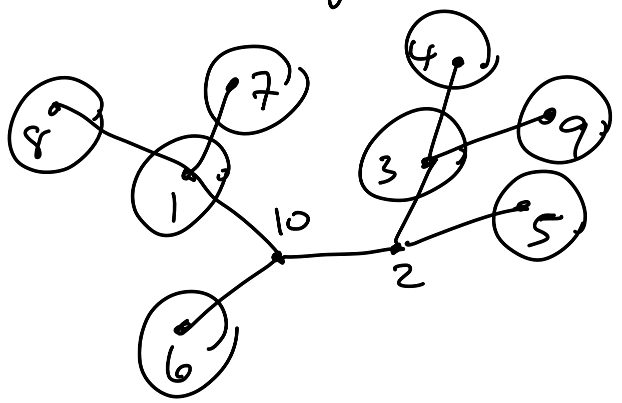
How do we choose which leaf to remove?

The one with the smallest number.

Algorithm: (trees to sequences)

Remove smallest leaf at each step.

Add its neighbor to the sequence.



n=10

sequence has length 8

Prüfer sequence:

3, 2, 10, 1, 1, 10, 3, 2

1. Remove 4
   3
2. Remove 5
   3, 2
3. Remove 6
   3, 2, 10
4. Remove 7
   3, 2, 10, 1
5. Remove 8
   3, 2, 10, 1, 1
6. Remove 1
   3, 2, 10, 1, 1, 10
7. Remove 9
   3, 2, 10, 1, 1, 10, 3
8. Remove 3
   3, 2, 10, 1, 1, 10, 3, 2

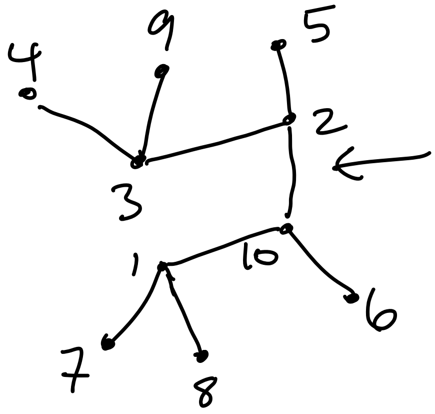
Stop!

Question: How do we go from a sequence to a tree?

Ex: ~~3~~, ~~2~~, ~~10~~, ~~1~~, ~~1~~, ~~10~~, ~~3~~, ~~2~~

Want to reconstruct the tree

First leaf removed is smallest # not in sequence!



Last step: connect the 2 that never showed up as leaves.

Reverse algorithm: (sequences to trees)

Find smallest # not in our sequence. (leaf)

Connect it to current first #.

Remove that # from the sequence.

Repeat until no terms left in our sequence.

Connect the 2 vertices that never showed up as leaves.

Second proof:

What's another counting problem about functions whose answer is n^{n-2}?

functions $f: \{1, 2, \dots, n\} \to \{1, 2, \dots, n\}$?

n^n : for each of the n inputs, n ways to choose the output

To get n^{n-2}: fix output for 2 inputs

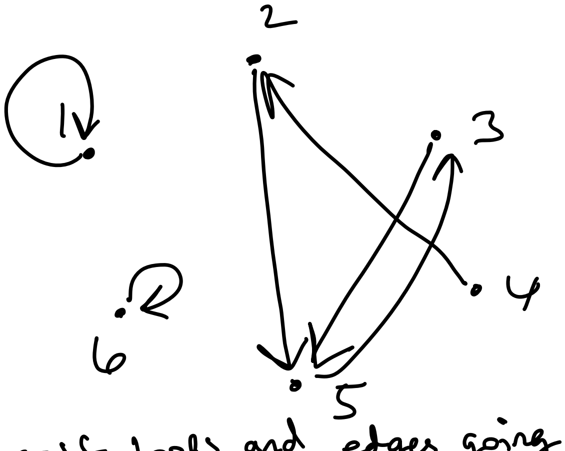
$f(1)=1$

$f(n)=n$

Goal: Find a bijection between trees on n vertizes and functions $f: \{1, 2, \dots, n\} \to \{1, 2, \dots, n\}$ such that $f(1)=1$ and $f(n)=n$.

How to represent a function using a graph?

Directed graph — each edge is an arrow in one direction

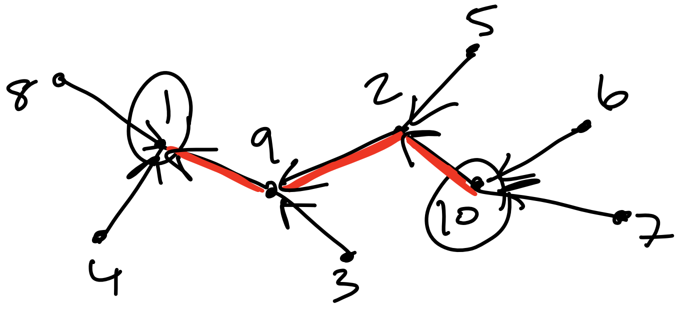


Draw an arrow from n to f(n) for each n.

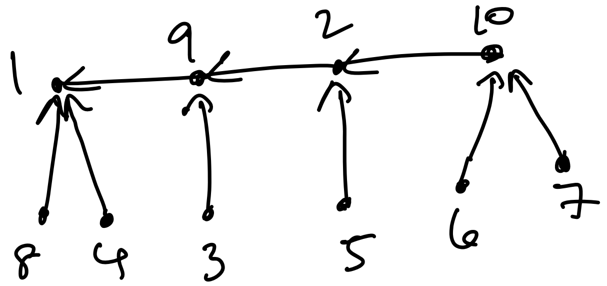
$f(1)=1$
$f(2)=5$
$f(3)=5$
$f(4)=2$
$f(5)=3$
$f(6)=6$

Allow self loops and edges going both ways.

First: Turn tree into a directed graph.



There's a unique path from 1 to n (because no cycles). Direct these edges toward 1.

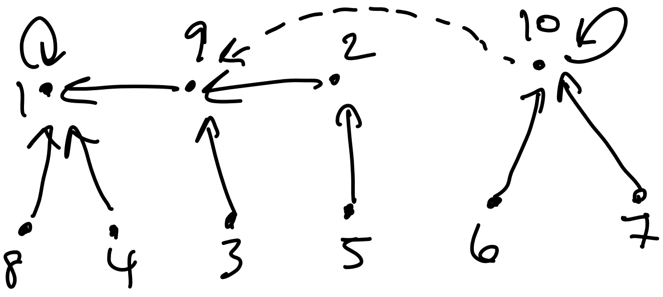


Direct others toward the path

Can we interpret this graph as a function?

Add loop from 1 to itself.

Want 10 to map to itself.

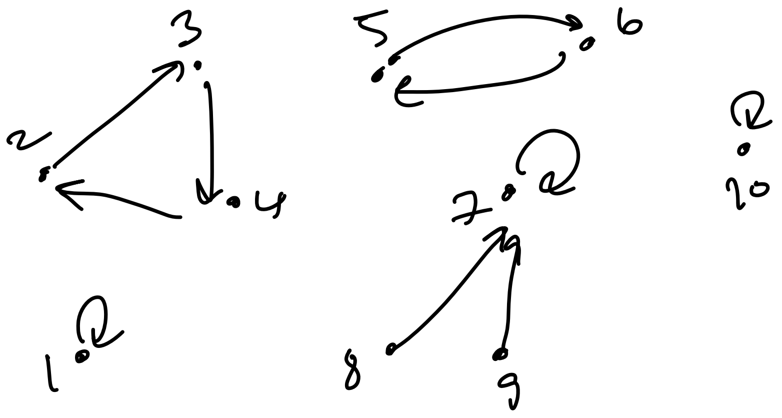


Can this mapping be one-to-one?

No! Can't uniquely recover tree from function (10 could have mapped to 9).

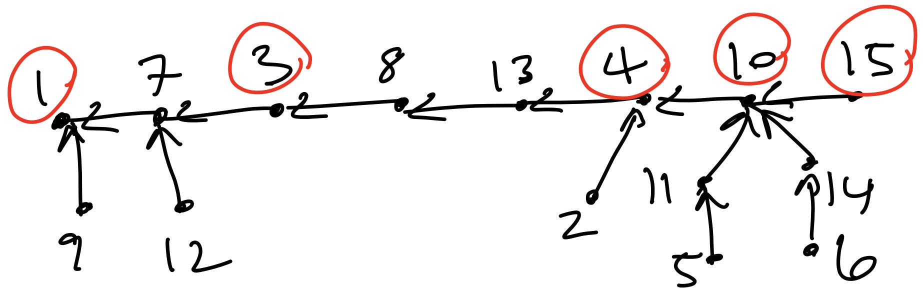
Lots of functions currently don't have preimages (ones with self-loops or other loops besides 1 and 10).

Need to change this to introduce some cycles.

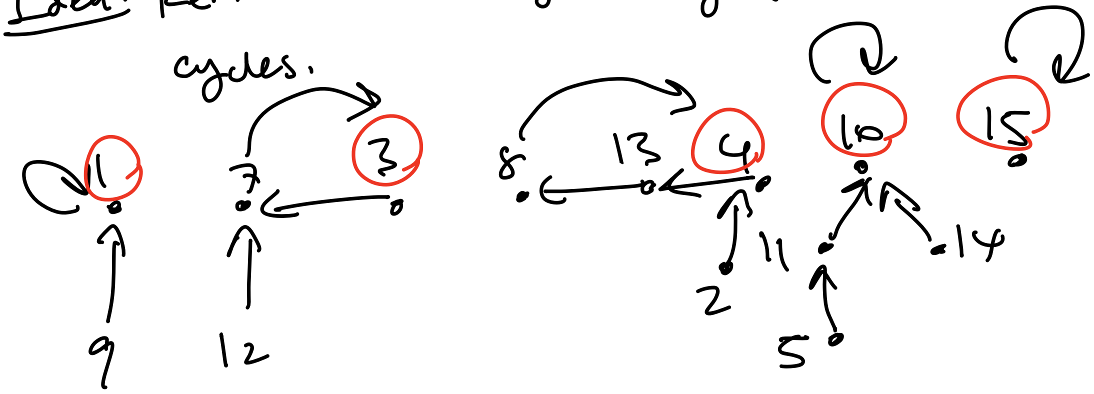


Things like this need to correspond to a tree.

Bigger example:



Idea: Remove some edges along path and create cycles.



- Mark vertices along the path which are smaller than all the ones to their right (circled in red)
- Erase edges going into those vertices.
- Add edges from each # that now needs one to the next red vertex to the right (including itself).

$f(1)=1$

$f(2)=4$

$f(3)=7$

$f(4)=13$

$f(5)=11$

$f(6)=14$

$f(7)=3$

$f(8)=4$

$f(9)=1$

$f(10)=10$

$f(11)=10$

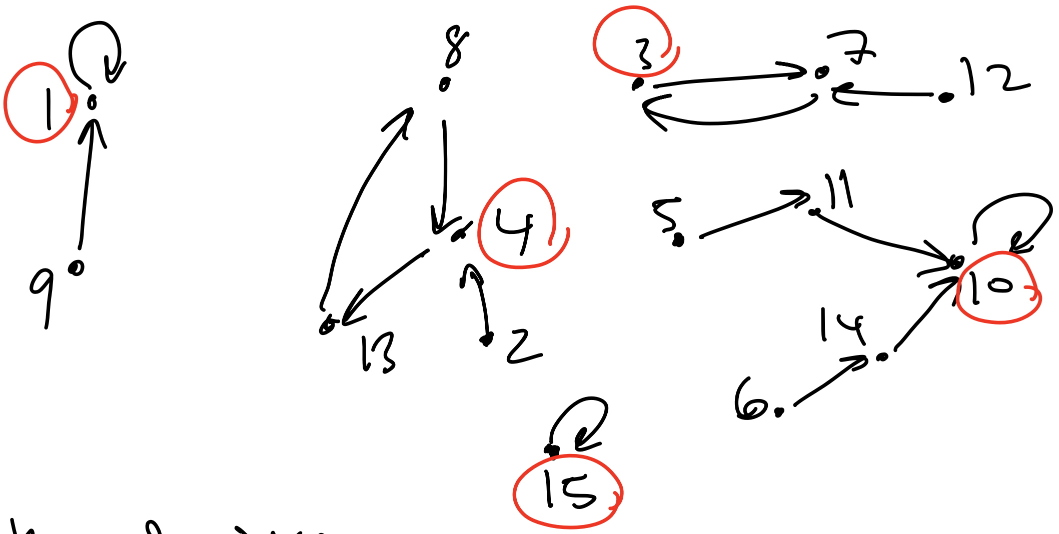
$f(12)=7$

$f(13)=8$

$f(14)=10$

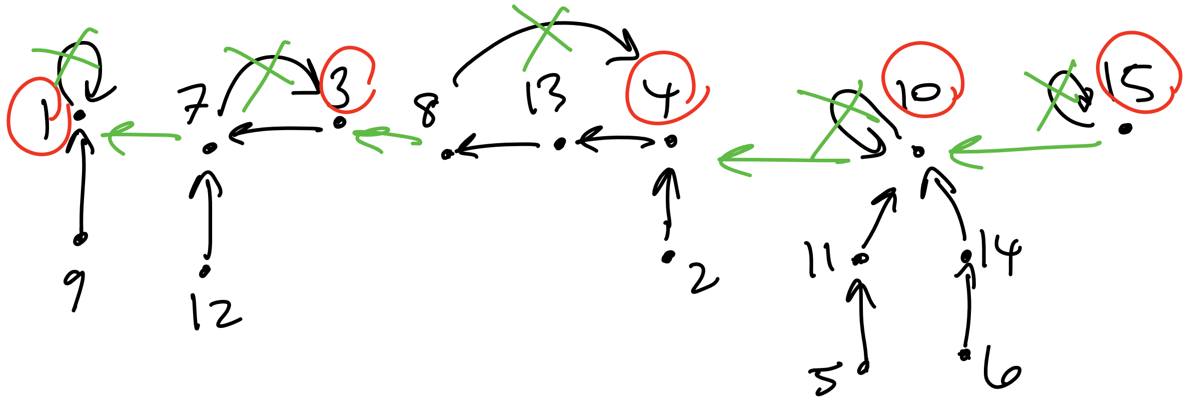
$f(15)=15$

How do we go from a function to a tree?



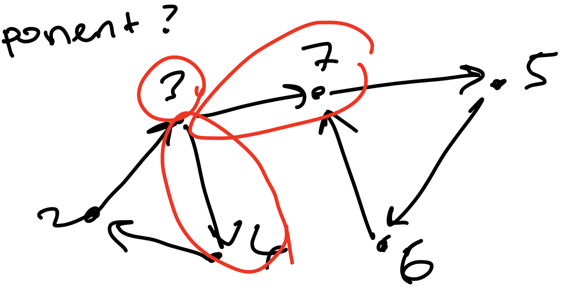
Find the red points: smallest ones in each cycle.

Remake top line by ordering red points.



Get rid of loops and fill in the path! Always point to next smallest red point.

Why not multiple cycles in one connected component?



There would have to be an edge coming out of a node in one cycle, and then that node is trying to map to 2 places!

Don't have to worry about this!

Pf due to Egecioglu and Remmel