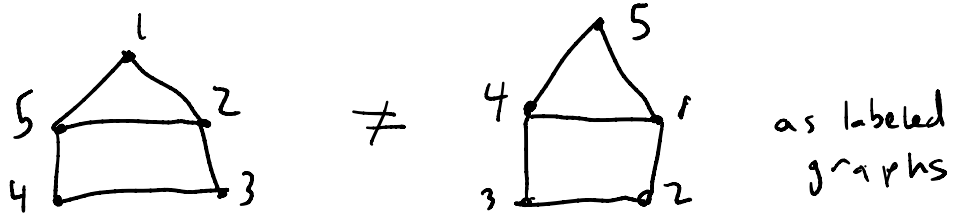
Math 4707: Counting trees

2/24
Ch. 8 of LPV

Reminder: • Midterm #2 is due today!

Today we are going to count graphs. As hinted at last class, there are two different possible interpretations of counting graphs on n vertices. If we're counting labeled graphs on $\{1, 2, \dots, n\}$ then we only count two graphs as the same is they have exactly the same edges:



If we're counting unlabeled graphs on n vertices, then we count two graphs as the same when they are isomorphic.

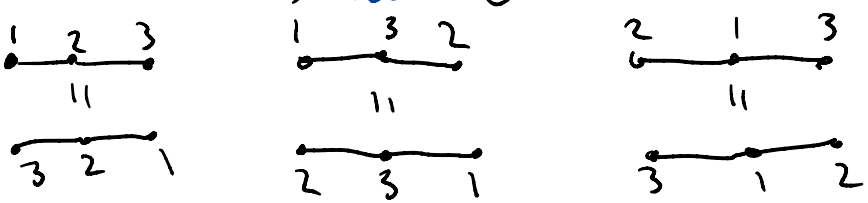
e.g. To count all simple labeled graphs on $\{1, 2, \dots, n\}$, we either include or don't each $\{i, j\}$, so the # is $2^{\binom{n}{2}}$. But counting unlabeled simple graphs is much harder....

Today we will focus specifically on counting trees on n vertices. You already did this a bit on the worksheet last class....

For instance, you probably figured out that there's only 1 unlabeled tree on 3 vertices:

How many labeled trees on $\{1,2,3\}$? Well, let's think about how many ways to label this tree?

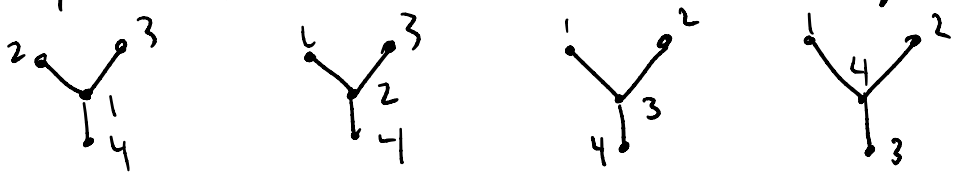
You might guess $3! = 6$, but actually there are only 3 ways because of reverses:



What about 4 vertices? Again, you saw last class there are 2 unlabeled trees on 4 vertices:

What about labeled trees on $\{1,2,3,4\}$? Well again, because of reverses, we have

$\frac{4!}{2}$ ways to label the path. But how many ways to label the star? Only 4:



Total = 12+4 = 16. Let's make a chart:

n	# unlabeled trees	# labeled trees
1	1	1
2	1	1
3	1	3
4	2	16
5	3	125
6	6	1296
7	11	16807

$1 \cdot 1$

$1 \cdot 1 \cdot 2$

Notice any patterns?

Thm (Cayley's Formula)

(labeled trees on $\{1, 2, \cdots, n\}$) $= n^{n-2}$.

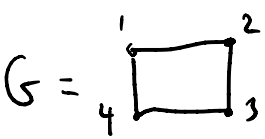
Very elegant formula! And somewhat deeper than any counting result we've seen so far.

To prove Cayley's formula, let's think abt. different ways to encode a labeled tree.

a) Adjacency matrix

For any labeled graph G on $\{1,2,\dots,n\}$, we can make an $n \times n$ matrix $A_G = (a_{ij})$ called the adjacency matrix w/ entries

$$a_{ij} = \begin{cases} 1 & \text{if } \{i,j\} \text{ an edge of } G \\ 0 & \text{otherwise} \end{cases}$$

e.g. 

$$\longrightarrow A_G = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

This uses $\sim n^2$ bits, too much for Cayley's formula, but we will talk about A_G next class...

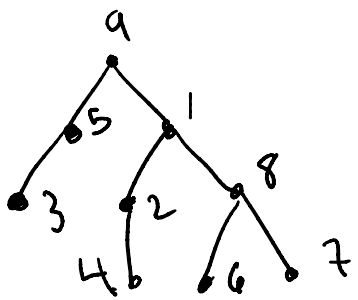
b) Father code

Let's consider our tree T on $\{1,2,\dots,n\}$ to be rooted at vertex n. Then ever vertex $i = 1,2,\dots,n-1$ has a unique father. The

father code of T is a 2-line array that records the fathers of these vertices in order:

e.g.

T=



1	2	3	4	5	6	7	8
9	1	5	2	9	8	8	1

father code

It's not hard to see that can recover T from its father code, and f.c. = length n-1 word in n letters, getting us very close to Cayley's formula...

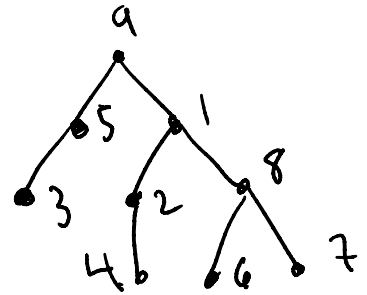
c) Extended Prüfer code

The extended Prüfer code of T is almost the same as the father code, except that we permute the columns in a special way.

Specifically, we look for the leaf with the smallest label, record it and its father, and then delete it! Then we repeat on our smaller tree, and keep going until all the non-root vertices have been deleted.

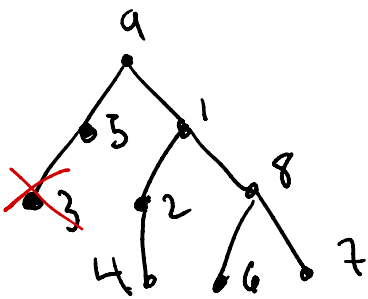
e.g.

T=



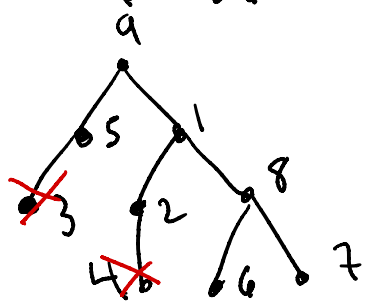
Smallest leaf = 3 ⇝

record
3
5



Smallest leaf = 4 ⇝

	record ↓
3	4
5	2



smallest leaf = 2 ⇝

		record ↓
3	4	2
5	2	1

⋮

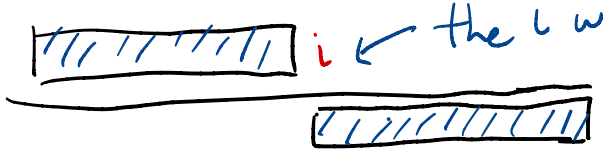
extended Prüfer code =

3	4	2	5	6	7	8	1
5	2	1	9	8	8	1	9

Since the extended Prüfer code is just a permutation of the father code, we can get the father code by putting 1st line in order, and hence extended P. code again determines T.

Prop. The 2nd line of the e.p.c. determines the 1st line.

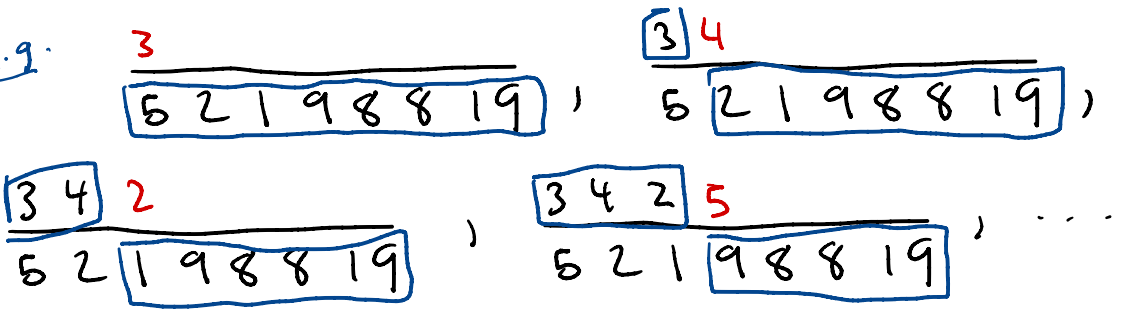
Pf:



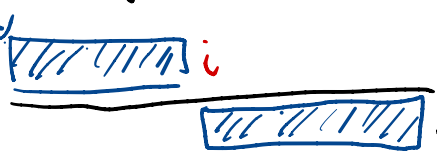
the i we put here is the minimum # not appearing in the shaded boxes

So we can build up 1st line left-to-right:

e.g.



Can we see why this min. property works?

The #'s in  have been deleted

by the time we're choosing leaf i, while #'s in are fathers of some future vertex, so are not leaves!

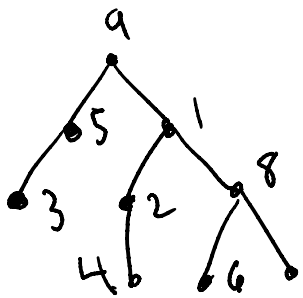
Prop. The last # in 2nd line of e.p.c. is n.

Pf: We always end at a tree like 

d) Prüfer code

The **Prüfer code** of T is obtained from the e.p.c. by removing the extraneous information: remove 1st line + last entry of 2nd line.

e.g.:



$T =$

e.p.c. $=$

3	4	2	5	6	7	8	1
5	2	1	9	8	8	1	9

Prüfer code $= 5\,2\,1\,9\,8\,8\,1$

By propositions, the Prüfer code determines the e.p.c., and hence the tree T. But also, the Prüfer code is a length $n-2$ word in n letters!

Thm Any length $n-2$ word $p_1 p_2 \dots p_{n-2}$ in letters $\{1, 2, \dots, n\}$ is Prüfer code of some tree T on vertices $\{1, 2, \dots, n\}$.

Pf: Exercise for you (or look it up...) //

Since $\exists\, n^{n-2}$ such words,

Thm $\Rightarrow$ Cayley's formula. ✓

Counting unlabeled trees

Now let's think about how to count unlabeled trees on n vertices. There is no exact formula like Cayley's formula, so we'll have to be content w/ giving bounds.

First lets find a lower bound. Each unlabeled tree on n vertices can be labeled in at most $n!$ ways:

$$\frac{\#\text{ labeled trees on } n \text{ vert.}}{n!} \leq \#\text{ unlabeled trees on } n \text{ vert's}$$

Cayley $\Rightarrow$

$$\frac{n^{n-2}}{n!} \leq \#\text{ unlabeled trees on } n \text{ vert.'s}$$

Stirling's approx. $\Rightarrow$ $\frac{n^{n-2}}{n!} \sim e^n / n^{5/2}\sqrt{2\pi}$

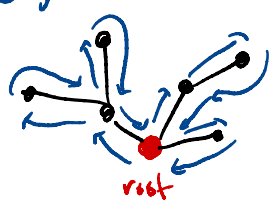
Certainly, for big enough n, we have

$$2^n \leq \#\text{ unlabeled trees}$$

For an upper bound, we need a different idea. We need some way to encode an unlabeled tree as a word of some kind. This is what the planar code of T does. To make the

planar code, we embed T in the plane, choose a root, and take a walk around T, recording a 1 when we walk out from root, 0 when we walk back towards root:

e.g.



planar code:

$\rightsquigarrow$ 1 1 0 1 0 0 1 1 0 0 1 0

Planar code = word in 0,1 of length $2(n-1)$ τ $2 \cdot \#E$

Each T can determine several codes (depending on choice of walk, etc.), and not every word is a planar code, but important fact is: (think abt.)

the planar code determines the tree!

So $\Rightarrow$ # unlabeled trees $\leq 2^{2(n-1)} = 4^{n-1}$

Altogether have

$$2^n \leq \#\text{unlabeled trees on } n \text{ vert.'s} \leq 4^n \quad \checkmark$$

Note Pólya proved that

$$\#\text{unlabeled trees} \sim a\, n^{-5/2}\, b^n$$

w/ $a = 0.5349\ldots$ and $b = 2.9557\ldots$