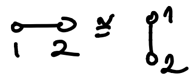
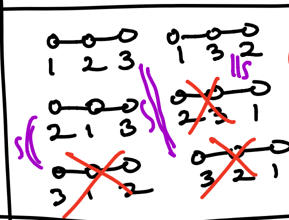
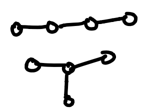
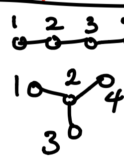
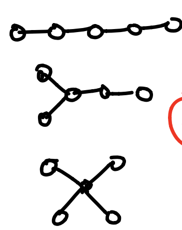
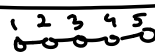
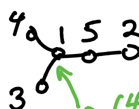
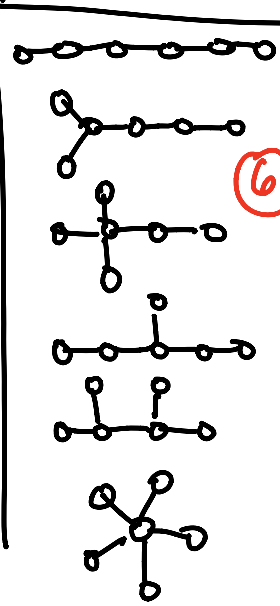
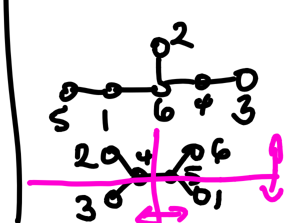
| $n$ | unlabeled trees on $n$ vertices | | labeled | |
|---|---|---|---|---|
| 1 |  | 1 | 1 ∘ | $1 = 1^{-1}$ |
| 2 |  | 1 |  | $1 = 2^{0}$ |
| 3 |  | 1 |  | $3 = 3^{1}$ |
| 4 |  | 2 |  | $\frac{4!}{2} = 12$ + $\binom{4}{1} = 4$; $16 = 4^{2}$ |
| 5 |  | 3 |    | $\frac{5!}{2} = 60$ + $\frac{5!}{2} = 60$ + $3\binom{4}{2}2 = 60$; $\binom{5}{1} = 5$; $125 = 5^{3}$ |
| 6 |  | 6 |  | $6!/2 = 360$ + $6!/2 = 360$ + $6!/3! \stackrel{?}{=} \binom{6}{3}\cdot 3! = 120$ + $6!/2 = 360$ + $6!/(2\cdot2\cdot2) = 90$ + $\binom{6}{1} = 6$; $1296 = 6^{4}$ |

Math 4707 Oct. 28, 2020

**THEOREM** (Cayley 1889, Borchardt 1860)

$$\#\left\{\text{labeled trees on } \underset{\{1,2,\ldots,n\}}{n \text{ vertices}}\right\} = n^{n-2}$$

**THEOREM**

$$\frac{n^{n-2}}{n!} \leq \#\left\{\begin{array}{c}\text{unlabeled trees}\\ \text{on } n \text{ vertices}\end{array}\right\} \leq 4^{n-1}$$

$\wr\wr$ Stirling's approximation

$$2^n \leq \frac{e^n}{\sqrt{2\pi}\; n^{5/2}}$$

Borchardt–Cayley Theorem is deep;
every proof (and there are many)
uses an interesting idea!

[A generating function in Wilf's book on syllabus]

# Proof of Borchardt–Cayley theorem via Prüfer coding:

There's a bijection

$$\left\{ \begin{array}{l} \text{labeled trees on vertices} \\ \{1,2,\dots,n\} \end{array} \right\} \xrightarrow{\text{Prüfer coding}} \left\{ \begin{array}{l} \text{sequences} \\ (c_1, c_2, \dots, c_{n-2}) \end{array} \right\}$$

$\underbrace{\text{with } c_i \in \{1,2,\dots,n\}}$

this set has size $n^{n-2}$ via multiplication principle!

$$\underset{T}{\text{tree}} \longmapsto (c_1, c_2, \dots, c_{n-2})$$

## Algorithm:

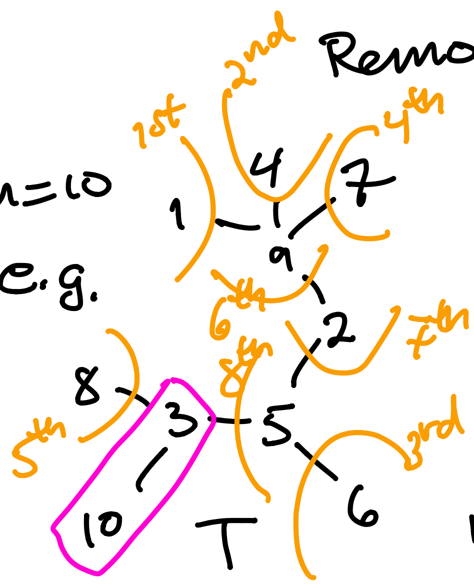
Let $c_1 :=$ label on the unique neighbor of the smallest leaf

Remove that leaf, and let

$c_2 :=$ label on unique neighbor of the smallest leaf left

Repeat and stop when left with an edge.

e.g. $n=10$




$\longmapsto$ Prüfer code $(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8)$

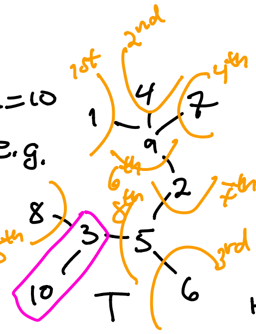
$= (9, 9, 5, 9, 3, 2, 5, 3)$

How to reverse this (the inverse bijection)??

**KEY OBSERVATION:**
A vertex $v \in V$ has
$$\deg_T(v) = 1 + \left(\begin{matrix}\#\text{times } v \\ \text{appears in} \\ (c_1, c_2, \ldots, c_{n-2})\end{matrix}\right)$$

$n=10$

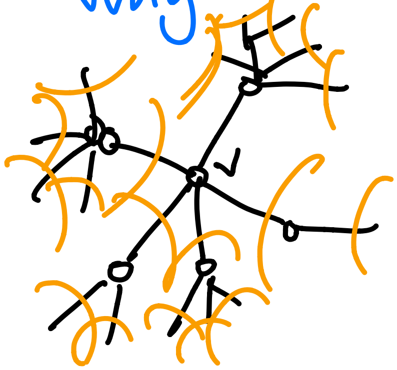
e.g.




$\longmapsto$ Prüfer code $(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8)$
$= (9, 9, 5, 9, 3, 2, 5, 3)$

In particular, $v$ is a leaf $\Longleftrightarrow$ $v$ is not in the Prüfer code at all.

Why the key observation?




Algorithm for recovering $T$ from $(c_1, c_2, \ldots, c_{n-2})$ is to create the leaf list and create an edge for the smallest ~~of the~~ leaves attached to $c_1$. Erase $c_1$, erase that leaf, and repeat.

When a vertex $v$ gets crossed off the Prüfer code the last time, it enters the leaf-list!

$n=10$    code $\rightsquigarrow$ leaf-list

$c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7\ c_8$

(9, 9, 5, 9, 3, 2, 5, 3)    1, 4, 6, 7, 8, 10

(9, 5, 9, 3, 2, 5, 3)    4, 6, 7, 8, 10

(5, 9, 3, 2, 5, 3)    6, 7, 8, 10

(9, 3, 2, 5, 3)    7, 8, 10

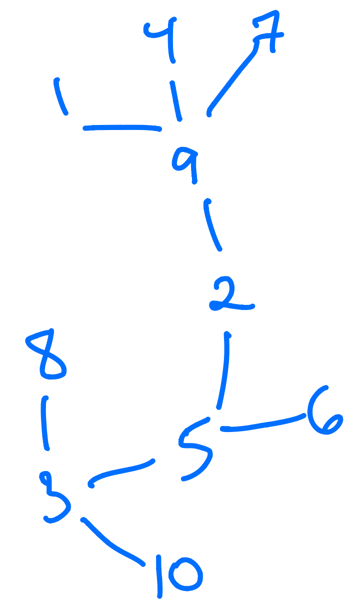
(3, 2, 5, 3)    8, 9, 10    9 entered because it left the code

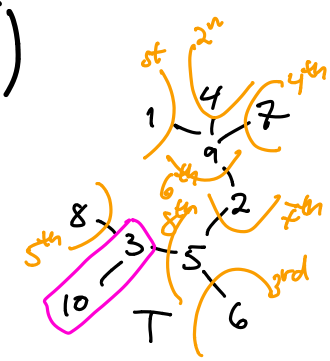
(2, 5, 3)    9, 10

(5, 3)    2, 10

(3)    5, 10

()    3, 10



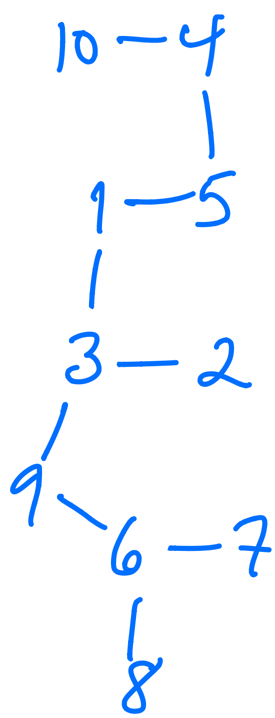

$(V, E)$

$V = \{1, 2, \ldots, n\}$

$E = \{\{i, j\}\}$

Our randomly chosen Prüfer code is...

| $c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7\ c_8$ | leaf-list |
| --- | --- |
| (3, 6, 6, 9, 3, 1, 5, 4) | 2, 7, 8, 10 |
| (6, 6, 9, 3, 1, 5, 4) | 7, 8, 10 |
| (6, 9, 3, 1, 5, 4) | 8, 10 |
| (9, 3, 1, 5, 4) | 6, 10 |
| (3, 1, 5, 4) | 9, 10 |
| (1, 5, 4) | 3, 10 |
| (5, 4) | 1, 10 |
| (4) | 5, 10 |
| () | 4, 10 |

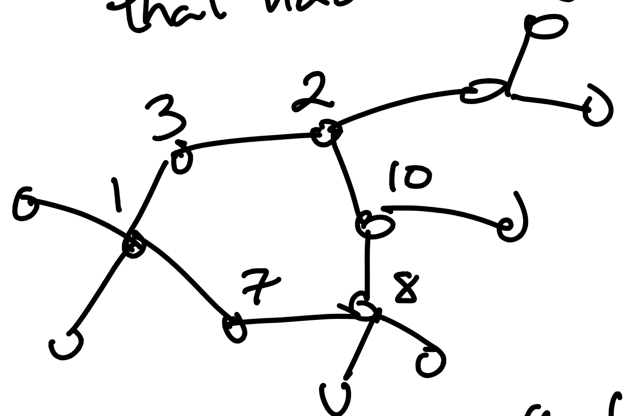



Why is the resulting graph <u>connected</u>?

Every vertex v has a path to the last two vertices in the leaf list, using reverse induction on the order in which v got crossed off the leaf-list.

Why is the resulting graph acyclic (no cycles)?

Can't create cycles working backwards (reverse order in which v gets crossed off leaf-list) because you always add an edge to a new vertex v that had no edges before.




Once you believe that any code $(c_1, \ldots, c_{n-2})$ produces a tree T, it's not hard to check the two algorithms give inverse bijections. ∎

Let's return to...

<u>THEOREM</u>: $T_n := \#\left\{\begin{array}{l}\text{unlabeled trees}\\ \text{on } n \text{ vertices}\end{array}\right\}$

has

$$\left(2^n \le\right) \frac{n^{n-2}}{n!} \underset{(1)}{\le} T_n \underset{(2)}{\le} 4^{n-1}$$

<u>proof:</u> To show (1), let's show

$$n! \cdot T_n \ge n^{n-2} \quad \text{Cayley-Borchardt}$$

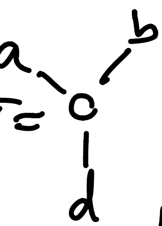
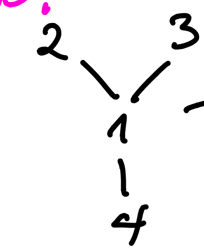
i.e.

$$\#\left\{\begin{array}{l}\text{permutations}\\ \text{of } \{1,2,\ldots,n\}\end{array}\right\} \cdot \#\left\{\begin{array}{l}\text{unlabeled}\\ \text{trees on}\\ n \text{ vertices}\end{array}\right\} \ge \#\left\{\begin{array}{l}\text{labeled}\\ \text{trees on}\\ \{1,2,\ldots,n\}\end{array}\right\}$$

via a surjective (onto) map

$$\left\{ (\sigma, T) : \begin{array}{l}\sigma \text{ a permutation of } \{1,2,\ldots,n\},\\ T \text{ an unlabeled tree on } n \text{ vertices}\end{array} \right\} \longrightarrow \left\{\begin{array}{l}\text{labeled tree } T' \\ \text{on } \{1,2,\ldots,n\}\end{array}\right\}$$

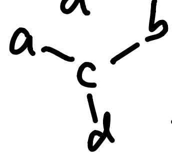
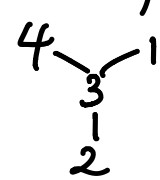
e.g. $n=4$

$\left( \sigma = \begin{pmatrix} a & b & c & d \\ 2 & 3 & 1 & 4 \end{pmatrix}, \ T = \right.$  $\left.\right) \longmapsto$  $T'$



surjective!



$\left( \sigma' = \begin{pmatrix} a & b & c & d \\ 4 & 1 & 3 & 2 \end{pmatrix}, \ T = \right.$  $\left.\right) \longmapsto$  $T'$

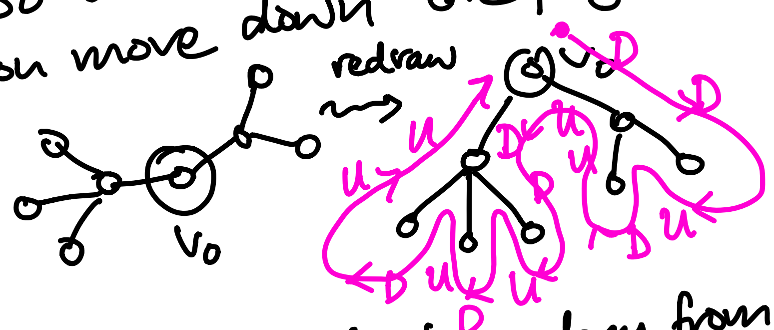
Once we know $T_n \geq \dfrac{n^{n-2}}{n!} \underset{\text{as } n\to\infty}{\approx} \dfrac{n^{n-2}}{\left(\frac{n}{e}\right)^n \sqrt{2\pi n}} = \dfrac{e^n}{n^{5/2}\cdot\sqrt{2\pi}} \geq 2^n$

---

To show $T_n \leq 4^{n-1}$,
let's do some encoding of an unlabeled tree $T$ by a sequence of ~~letters~~ of letters $D, U$ of length $2(n-1)$:

① Start with any vertex $v_0$ as root vertex

② Draw $T$ in the plane with no crossings so that as you move away from $v_0$ you move down the page:

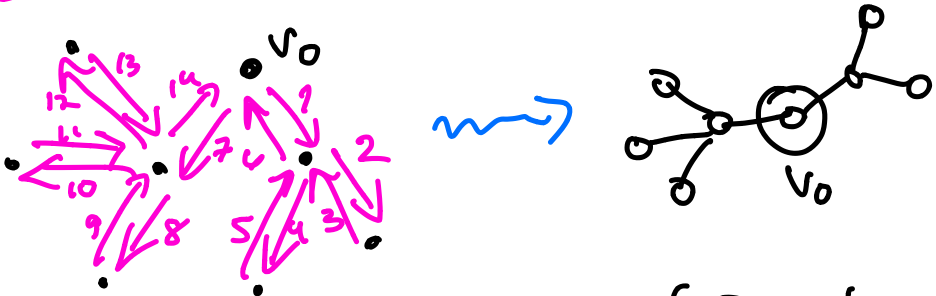



③ Traverse the outer boundary from $v_0$, returning to $v_0$, writing $D, U$ as you go down and up along edges.

$$\underbrace{DDUDUUDDUDUDUU}_{2(n-1) = 2\#E}$$

1 2 3 4 5 6 7 8 9 10 11 12 13 14

DDUDUUDDUDUDUU




Hence $T_n \le$ # sequences of D, U's of length $2(n-1)$

$$= 2^{2(n-1)} = 4^{n-1}$$

$$\left(\text{in fact, } \le \binom{2(n-1)}{n-1} \underset{\text{Stirling}}{\approx} 4^n \right) \blacksquare$$

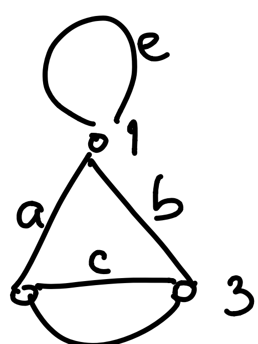
# Math 4707 Nov 2, 2020

Counting spanning trees (not in our book) and then finding minimum-cost spanning trees (§9.1)

---

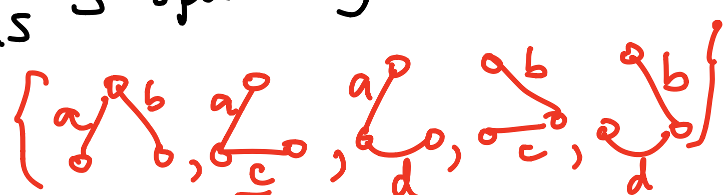
**DEF'N:** Given a graph $G=(V,E)$ with multiple edges and loops allowed, then a <u>spanning tree</u> $T$ in $G$ is a subset $T \subset E$ such that $(V,T)$ is a tree.

e.g. $G=$



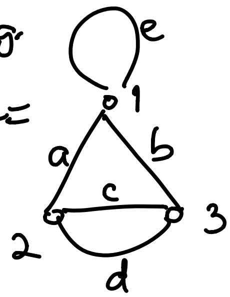

$=(V,E)$ $\quad V=\{1,2,3\}$, $E=\{a,b,c,d,e\}$

has 5 spanning trees:

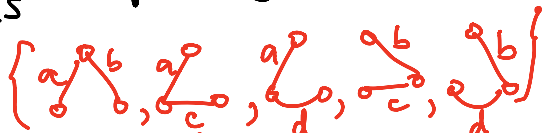



$T_1 = \{a,b\}$, $T_2 = \{a,c\}$, $T_3 = \{a,d\}$, $T_4 = \{b,c\}$, $T_5 = \{b,d\}$

e.g. $G =$




has 5 spanning trees:




$\left\{ T_1, T_2, T_3, T_4, T_5 \right\}$

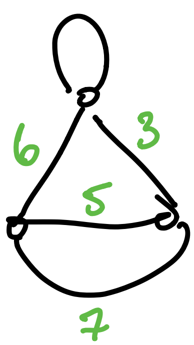
$T_1 = \{a,b\}$, $T_2 = \{a,c\}$, $T_3 = \{a,d\}$, $T_4 = \{b,c\}$, $T_5 = \{b,d\}$

## 2 problems:

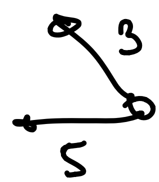
① How to count the number of spanning trees in $G$?

② How to find the cheapest spanning tree if the edges have costs?

edge costs in green



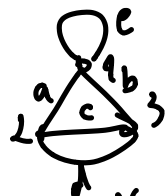

has cheapest spanning tree



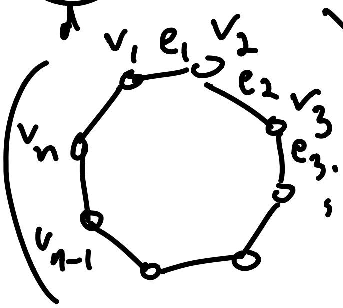

$= T_4$

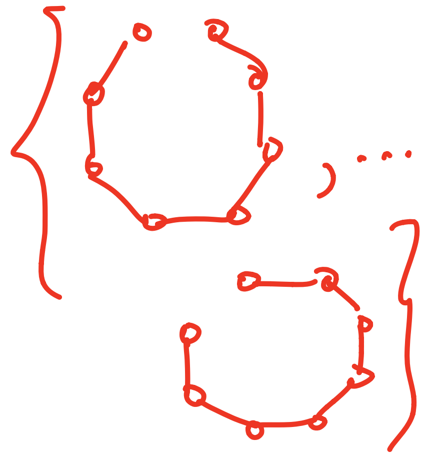
**DEFINITION:** Let $\tau(G) :=$ # of spanning trees in $G$

**EXAMPLES:**

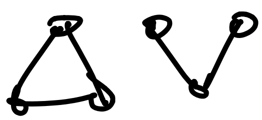
① $\tau\Bigg($ 

$\Bigg) = 5$

② $\tau\Bigg($ 

$\Bigg) = n$ 

n-cycle graph $C_n$

③ $\tau\Bigg($  $\Bigg) = ? = 0$

There is a recursive way to compute $\tau(G)$ (useful for HW; inefficient in general), uses the notion of deletion and contraction of an edge in $G$..

**DEF'N:** If $e$ is a non-loop edge of $G$,

$e = \{v, v'\}$, $G = (V, E)$

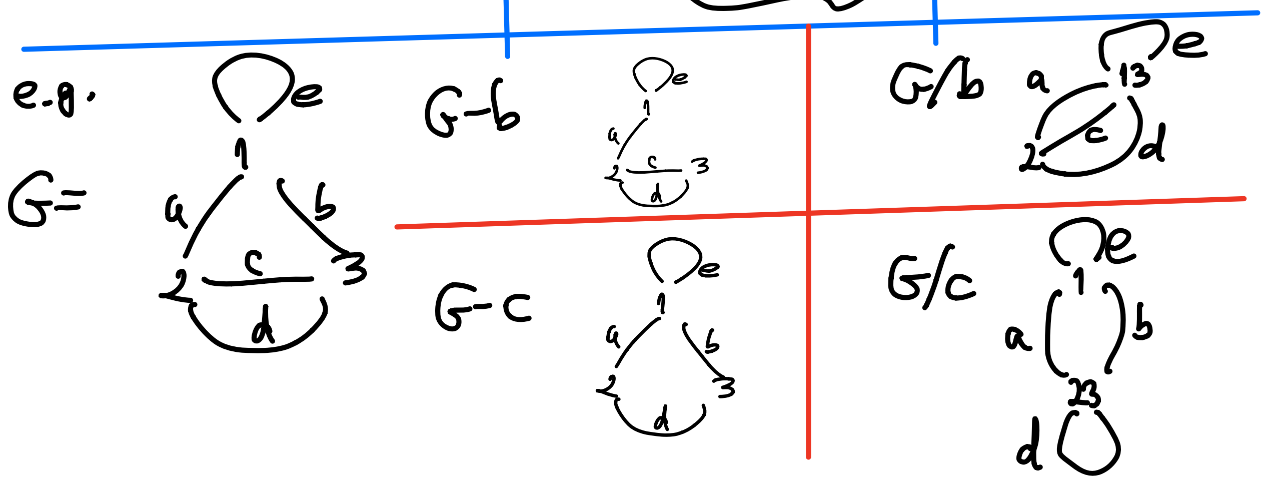
then $G \setminus e :=$ <u>deletion</u> of $e$ in $G$

$$= (V, E \setminus \{e\})$$

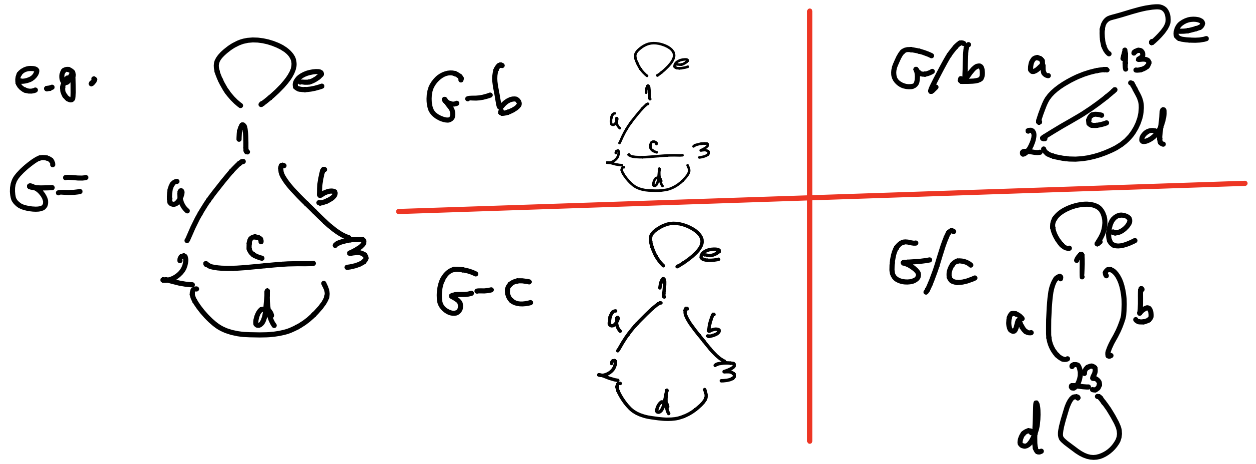
$G/e :=$ <u>contraction</u> of $e$ in $G$

$$= (V/e, E \setminus \{e\})$$

$V/e =$ squeeze together $v, v'$

e.g.




PROPOSITION: One can compute $\tau(G)$ recursively by induction on $|E|$ via these rules:

(a) $\tau(\bullet) = 1$ [$T = \emptyset$ no edges is the unique spanning tree]

(b) $\tau(G) = 0$ if $G$ is disconnected.

(c) $\tau(G) = \tau(G$ with all loops removed$)$

(d) if $e$ is any non-loop edge of $G$,

then $\tau(G) = \tau(G \setminus e) + \tau(G/e)$

**PROPOSITION:** One can compute $\tau(G)$ recursively by induction on $|E|$ via these rules.

(a) $\tau(\circ) = 1$ [$T = \emptyset$ no edges is the unique spanning tree]

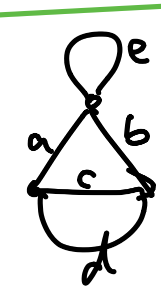
(b) $\tau(G) = 0$ if $G$ is disconnected.

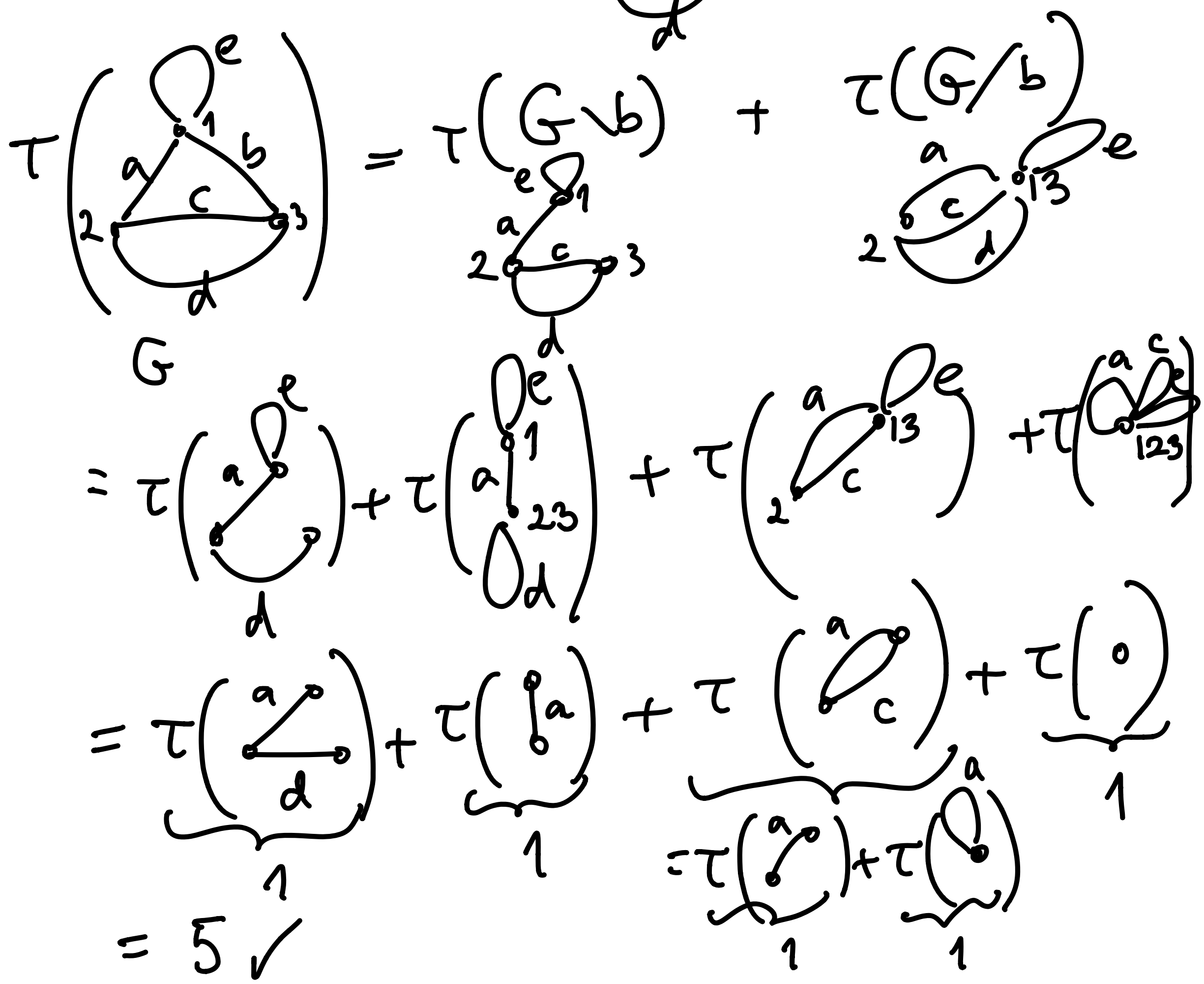
(c) $\tau(G) = \tau(G \text{ with all loops removed})$

(d) if $e$ is any non-loop edge of $G$, then $\tau(G) = \tau(G \setminus e) + \tau(G/e)$

---

# EXAMPLE:

Let's compute $\tau\Big($  $\Big)$ recursively:




$= 5$ ✓

**PROPOSITION:** One can compute $\tau(G)$ recursively by induction on $|E|$ via these rules.

(a) $\tau(\circ) = 1$ $\quad$ [$T = \emptyset$ no edges is the unique spanning tree]

(b) $\tau(G) = 0$ if $G$ is disconnected.

(c) $\tau(G) = \tau(G \text{ with all loops removed})$

(d) if $e$ is any non-loop edge of $G$, then $\tau(G) = \tau(G \setminus e) + \tau(G/e)$

proof: If we believe (a), (b), (c), (d) hold, then it's easy to see how the algorithm works. Properties (a), (b), (c) don't need more proof.

For (d), assuming $e$ is a non-loop edge of $G$

$$\tau(G) = \#\{\text{spanning trees } T \text{ in } G\}$$

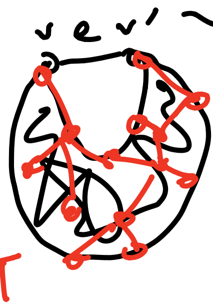
$$= \underbrace{\#\left\{\begin{matrix}\text{such } T\\ \text{that avoid}\\ \text{using } e\end{matrix}\right\}}_{\tau(G \setminus e)} + \#\left\{\begin{matrix}\text{such } T\\ \text{that use } e\end{matrix}\right\}$$

$$\text{bijection} \updownarrow$$

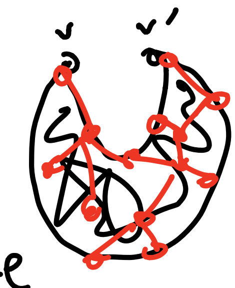
$$\underbrace{\#\left\{\begin{matrix}\text{spanning}\\ \text{trees of}\\ G/e\end{matrix}\right\}}_{\tau(G/e)}$$

This is not 100% obvious! Think about why.

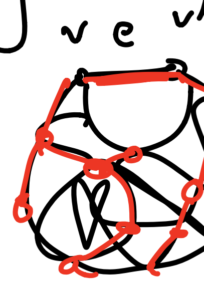
■



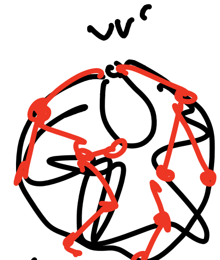
G
T



$G \setminus e$
T



G
T



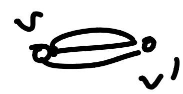
$G/e$ $\quad$ $T/e$

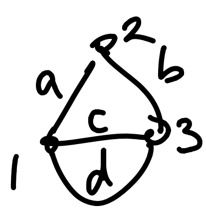
Let's get a better method out of this.

<u>DEFINITION</u>: Given $G=(V,E)$ (multiple edges OK; loops irrelevant, so remove them) define its $\#V \times \#V$ <u>Laplacian matrix</u> $L(G)$ having rows & columns index by $V$

with

$$\underbrace{L(G)_{v,v'}}_{\substack{(v,v')\text{-entry of}\\ \text{the matrix}}} = \begin{cases} \deg_G(v) & \text{if } v=v' \\ -\#\left(\substack{\text{edges } e \text{ in } E\\ \text{from } v \text{ to } v'}\right) & \text{if } v\neq v' \end{cases}$$



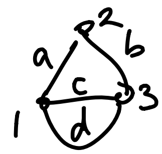

EXAMPLE $G=$ 

has

$$L(G)= \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 3 & -1 & -2 \\ -1 & 2 & -1 \\ -2 & -1 & 3 \end{bmatrix} \end{array}$$

THEOREM (Kirchhoff's 1848 Matrix-Tree Thm) For any vertex $v \in V$,

$$\tau(G) = \det \underbrace{\overline{L(G)}^{v}}_{\substack{\text{reduced Laplacian}\\ \text{matrix}}} = \det\left(L(G) \text{ with row } v \text{ and column } v \text{ removed}\right)$$

EXAMPLE $G =$  has $L(G) = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 3 & -1 & -2 \\ 2 & -1 & 2 & -1 \\ 3 & -2 & -1 & 3 \end{array}$

THEOREM (Kirchhoff's 1848 Matrix-Tree Thm) For any vertex $v \in V$,

$$\tau(G) = \det \underbrace{\overline{L(G)}^{v}}_{\text{reduced Laplacian matrix}} = \det \left( L(G) \text{ with row } v \text{ and column } v \text{ removed} \right)$$

<u>EXAMPLE:</u> $\tau\Big($  $\Big) = \det \overline{L(G)}^{1} = \det \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 3 & -1 & -2 \\ 2 & -1 & 2 & -1 \\ 3 & -2 & -1 & 3 \end{array}$

$$= \det \begin{array}{c|cc} & 2 & 3 \\ \hline 2 & 2 & -1 \\ 3 & -1 & 3 \end{array} = 2\cdot 3 - (-1)(-1) = 6 - 1 = 5 \checkmark$$

$$= \det \overline{L(G)}^{2} = \det \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 3 & -1 & -2 \\ 2 & -1 & 2 & -1 \\ 3 & -2 & -1 & 3 \end{array}$$

$$= \det \begin{array}{c|cc} & 1 & 3 \\ \hline 1 & 3 & -2 \\ 3 & -2 & 3 \end{array} = 3\cdot 3 - (-2)(-2) = 9 - 4 = 5 \checkmark$$

This is very useful both...

- <u>computationally</u>, because one can compute $n \times n$ determinants in $\leq C \cdot n^3$ steps for some constant $C$ via Gaussian elimination since ...

THEOREM (Kirchhoff's 1848 Matrix-Tree Thm) For any vertex $v \in V$,

$$\tau(G) = \det \underbrace{\overline{L(G)}^{v}}_{\text{reduced Laplacian matrix}} = \det\left(L(G) \text{ with row } v \text{ and column } v \text{ removed}\right)$$

This is very useful both...

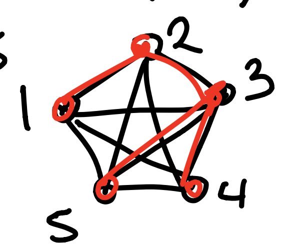
- computationally, because one can compute $n \times n$ determinants in $\leq C \cdot n^3$ steps for some constant $C$ via Gaussian elimination since $\det(A)$ is
  - unchanged by add a multiple of a row to another
  - negated by swapping rows
  - scaled by scaling a row.

- theoretically, because some graphs (or families) have enough structure to compute $\det \overline{L(G)}^{v}$ via eigenvalues of $\overline{L(G)}^{v}$!

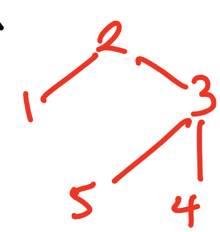
EXAMPLE: Let's give a $2^{nd}$ proof of Borchardt-Cayley Theorem this way.

$$\tau(K_n) = \#\left(\text{spanning trees on vertices } \{1,2,\ldots,n\}\right) = n^{n-2}$$

$n=5$



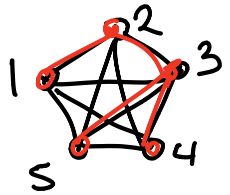
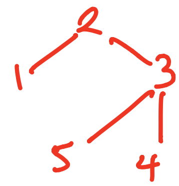

=



## Borchardt-Cayley Theorem

$$\tau(K_n) = \#\left(\text{spanning trees on vertices } \{1,2,\dots,n\}\right) = n^{n-2}$$

$n=5$

$K_5$  $=$ 

$$L(K_5) = \begin{array}{c} \\ 1\\2\\3\\4\\5 \end{array}\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & -1 & -1 & -1 & 4 \end{bmatrix} \end{array}$$

$$L(K_n) = \begin{array}{c|cccccc} & 1 & 2 & 3 & \cdots & & n \\ \hline 1 & n-1 & -1 & -1 & \cdots & & -1 \\ 2 & -1 & n-1 & & & & -1 \\ 3 & -1 & -1 & n-1 & & & \vdots \\ \vdots & \vdots & & & \ddots & & -1 \\ n & -1 & \cdots & & & -1 & n-1 \end{array}$$

$$\overline{L(K_n)}^{\vee} = \underbrace{\begin{bmatrix} n-1 & -1 & -1 & \cdots & -1 \\ -1 & n-1 & -1 & & \vdots \\ -1 & -1 & \ddots & & -1 \\ \vdots & & & & \\ -1 & -1 & \cdots & & n-1 \end{bmatrix}}_{n-1} \Bigg\} n-1 = \underbrace{\begin{bmatrix} n & & 0 \\ & n & \\ & & \ddots \\ 0 & & n \end{bmatrix}}_{nI_{n-1}} - \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & & & \\ 1 & 1 & \cdots & 1 \end{bmatrix}}_{J_{n-1}}$$

all 1's matrix of size $(n-1)\times(n-1)$

Hence Kirchhoff says

$$\tau(K_n) = \det\left(nI_{n-1} - J_{n-1}\right)$$

Hence Kirchhoff says

$$\tau(K_n) = \det\left(nI_{n-1} - J_{n-1}\right) \quad \text{where } J_{n-1} = \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \ddots & & 1 \\ \vdots & & \ddots & 1 \\ 1 & 1 & \cdots & 1 \end{bmatrix}}_{n-1} \Big\}\, n-1$$

We claim $v_1, v_2, \ldots, v_{n-2}$ are $n-1$ linearly independent eigenvectors for $nI_{n-1} - J_{n-1}$ with eigenvalue $n$

$$v_1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad v_2 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \ldots, \quad v_{n-2} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ -1 \end{bmatrix} \Bigg\}\, n-1$$

since $J_{n-1} v_i = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 0 \cdot v_i$

$$nI_{n-1} v_i = n \cdot v_i$$

$$\left(nI_{n-1} - J_{n-1}\right) v_i = n \cdot v_i - 0 \cdot v_i = n \cdot v_i$$

We also claim $\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = v_0$ is an eigenvector for $nI_{n-1} - J_{n-1}$

with eigenvalue 1, since $J_{n-1} v_0 = \begin{bmatrix} n-1 \\ n-1 \\ \vdots \\ n-1 \end{bmatrix} = (n-1) v_0$

$$\left(nI_{n-1} - J_{n-1}\right) v_0 = n \cdot v_0 - (n-1) v_0 = 1 \cdot v_0$$

So $nI_{n-1} - J_{n-1}$ has eigenvalues $(\underbrace{n, n, \ldots, n}_{n-2 \text{ times}}, 1)$

and $\det\left(nI_{n-1} - J_{n-1}\right) = n^{n-2} \cdot 1 = n^{n-2}$