

## Second unit: Algorithms

Some topics in Decision Maths; mostly “easy” points on exam.

Largely about optimization problems:

- ▶ Kruskal and Prim's algorithms for cheapest spanning trees
- ▶ Dijkstra's algorithm: shortest path between two points
- ▶ Travelling Salesperson problem

In the real world: computers run these algorithm

From pure math perspective, interesting bits are:

- ▶ Proving the algorithm works as advertized
- ▶ Analyzing speed of algorithm – can you go faster?

First topic: Prüfer code

How many trees on  $n$  vertices are there?

# Two interpretations of counting trees

Count isomorphism classes of trees. “unlabelled trees”

- ▶ This is what we do when we count isomers
- ▶ No nice answer

Count labelled trees on  $n$  vertices

- ▶ Vertices are no longer interchangeable
- ▶  $n!$  ways to label an unlabeled tree
- ▶ Symmetries mean some produce the same labelled tree

n	1	2	3	4	5	6	7
Unlabelled trees	1	1	1	2	3	6	11
Labelled trees	1	1	3	16	125	1296	16807

# “Cayley’s” formula

Theorem (Borchardt 1860, Cayley 1889)

*There are  $n^{n-2}$  labelled trees on  $n$  vertices.*

Original proof used determinants.

Prüfer code: another way to prove Cayley’s formula

Bijection between *Trees* and *Codes*

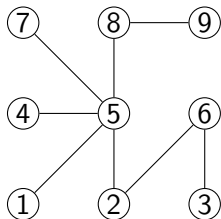
- ▶  $T_n = \{\text{labelled trees on } n \text{ vertices}\}$
- ▶  $C_n = \{(a_1, a_2, \dots, a_{n-2}) : a_i \in \{1, 2, \dots, n\}\}$
- ▶ Build a bijection between  $T_n$  and  $C_n$
- ▶  $|C_n| = n^{n-2}$

## Bijjective Proofs

In combinatorics, bijective proofs often give more...

# How to write down a labelled trees?

Record the edges:



Each column is an edge

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

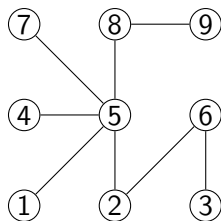
- ▶ Records  $2n - 2$  numbers between 1 and  $n$
- ▶ *Many* ways of recording same tree

Need to record the edges in a canonical way

“Canonical” means: without arbitrary choices

# Prüfer code: iteratively remove lowest leaf

1. Find lowest leaf  $\ell$  of  $T$
2. Record edge  $e$  connecting it to rest of tree
3. Delete  $\ell$  and  $e$  to get a simpler tree  $T'$
4. Repeat process with  $T'$



Parent	<u>5</u>	<u>6</u>	<u>5</u>	<u>2</u>	<u>5</u>	<u>5</u>	<u>8</u>	9
Leaf	1	3	4	6	2	7	5	8

The underlined numbers form the Prüfer code

Non-underlined numbers are a permutation of 1-9. Why?

To see Prüfer code is a bijection, construct inverse

Given a Prüfer code, how to fill in empty boxes?

Parent	5	6	5	2	5	5	8	
Leaf								

Numbers in the Prüfer code were parents

- ▶ So the numbers in Prüfer code can't be leaves
- ▶ We deleted lowest leaf first
- ▶ Thus, first leaf is lowest number not in the Prüfer code

To reconstruct tree from code:

- ▶ Find lowest number not used yet that's not in remaining code
- ▶ Delete the first column
- ▶ Iterate; last two numbers left are the last edge

## Cayley's enrichment: keep track of degrees of vertices

Parent	<u>5</u>	<u>6</u>	<u>5</u>	<u>2</u>	<u>5</u>	<u>5</u>	<u>8</u>	9
Leaf	1	3	4	6	2	7	5	8

Degree of vertex  $i$  = number of times  $i$  appears in table  
= number of times  $i$  appears in Prüfer code + 1

### Corollary

*The number of labeled trees on  $n$  vertices where for each  $i$ , vertex  $i$  has degree  $d_i$  is:*

$$\frac{(n-2)!}{\prod (d_i - 1)!}$$

Side point: consistent with original formula

Handshaking and multinomial formula.