The Catalan Numbers

CS 70, Summer 2019

Bonus Lecture, 7/19/19

Parentheses

How many ways can I **properly** arrange: ( )) (

open ↓ ↓ close close

- Zero pairs of parentheses? 1
- One pair of parentheses? ( ) 1
- Two pairs of parentheses? ( )( ) (( )) 2 (( )( ))
- Three pairs of parentheses? ( )( )( ) ((( ))) (( ))( ) ( )(( )) 5
- Four pairs of parentheses? Already getting hard... 14

Five Pairs?

1, 1, 2, 5, 14, 42, ..

Catalan Number Formula

The n-th Catalan number is given by:

$$C_n = \frac{1}{n+1} \cdot \binom{2n}{n}$$

We'll see how to get this formula later...

Why is C_n **always an integer**? We can rewrite:

$$C_n = \binom{2n}{n} - \binom{2n}{n+1}$$

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

$$= \frac{(2n)!}{n!\,(n-1)!} \cdot \left(\frac{1}{n} - \frac{1}{n+1}\right) = \binom{2n}{n}\left(\frac{1}{n+1}\right)$$

$$\frac{\cancel{n}+1-\cancel{n}}{n(n+1)} = \frac{1}{\cancel{n}(n+1)}$$

Recursion In Parentheses

The first character will always be a **left parenthesis.**

(( ))(( )( ))

1. Identify the **right parenthesis** it is matched with.
2. What goes **inside** the first left parenthesis and its partner? How many of them?
   (( ) ( ))
   valid arrangement. of parens.
   0 pairs, 1 pairs, ... (n-1) pairs
3. What goes **after** its partner? How many of them?
   ( ) ( )(( ))
   valid arrangement of parens.
   If i pairs inside, $(n-1)-i$ outside

Recursion In Parentheses: Count By Cases

Cases based on **how many pairs of parentheses inside**: Let C_i = # arr. of i pairs

- Case 0: No pairs inside.
  $( )\underbrace{\quad\quad}_{n-1 \text{ pairs}} = C_{n-1} = C_0 \cdot C_{n-1}$
- Case 1: One pair inside.
  $(( ))\underbrace{\quad\quad}_{n-2 \text{ pairs}} = C_1 C_{n-2}$
- Case 2: Two pairs inside.
  $(\underbrace{( )( ) \text{ or } (( ))}_{2 \text{ pairs}})\underbrace{\quad\quad}_{n-3 \text{ pairs}} = C_2 C_{n-3}$
- Case i: i pairs inside.
  $C_i C_{(n-1)-i}$

Catalan Number Recurrence

The n-th Catalan number is also given by:

$$C_n = \sum_{i=0}^{n-1} C_i \cdot C_{(n-1)-i}$$

ind. counts from prev. slide.

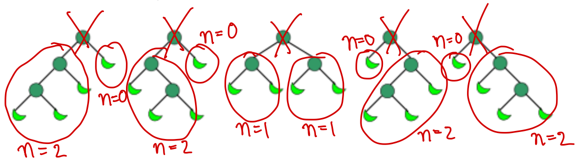
cases on # pairs inside.

If an object satisfies this recurrence, it can be counted by C_n!

pic: 4 leaves, n=3

Example: **Full** binary trees with $(n+1)$ leaves

Def: Every vertex either has 2 children, or no children

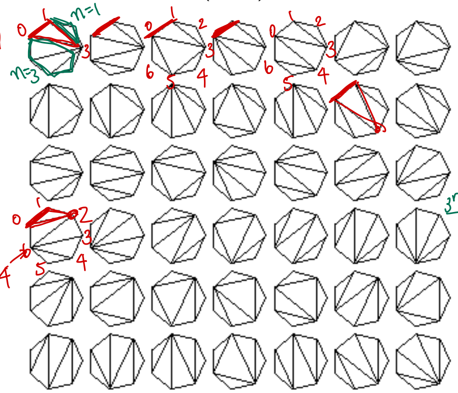


Other Items Counted By Recurrence

Example: **Triangulations** in $(n+2)$-sided polygons

7-sided n=5

Fix base edge



count by cases

3rd vertex of 0-1 triangle

3rd vertex

2: C_0C_4

3: C_1C_3

4: C_2C_2

5: C_3C_1

6: C_4C_0

Counting Using Bijections

Example: **Bitstrings** of n 0's and n 1's where, if we read from left to right, there are always more 0's than 1's.

≥ or equal.

Q: Bijection to something that we already counted?

↳ to parentheses

(maps to 0

) maps to 1.

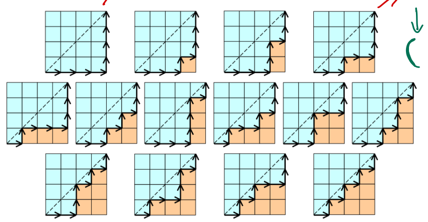
Counting Using Bijections

Example: **Lattice paths** on an $n \times n$ grid which do not cross above the diagonal.

RRRRUUUU

RRURRUUU

( )



Q: Bijection to something that we already counted?

Deriving the Formula

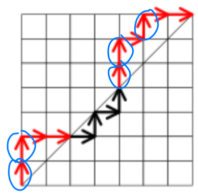
Let's study the formula again...

$$C_n = \frac{1}{n+1} \cdot \binom{2n}{n}$$

- What kinds of objects does $\binom{2n}{n}$ count?
  - all strings of n 0's n 1's
  - all lattice paths n ↑'s n →'s.
- What does the $\frac{1}{n+1}$ clue into?
  - partition elements into n+1 equal sets.

Exceedence

Exceedence: the number of **vertical edges** in the lattice path that lie **above the diagonal**



Exceed = 5.

Exceedence = 0:

↳ counted w/ Catalan #s!!

How many possible values are there for exceedence?

0, 1, 2, .., n ← (n+1) values

Bijection Between Exceedences

Goal: We can show that 0-exceedence paths are counted by C_n if:

$$\#\{0\text{-exc.}\} = \#\{1\text{-exc.}\} = \#\{2\text{-exc.}\} = \ldots = \#\{n\text{-exc.}\}$$

Want a bijection (**read: invertible transformation**) from

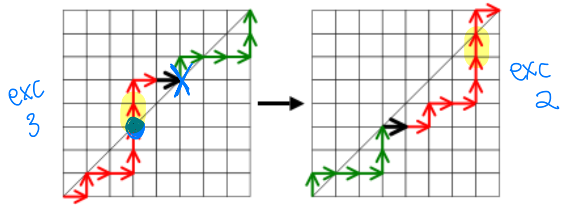
$$\#\{(i+1)\text{-exc.}\} \text{ to } \{i\text{-exc.}\}$$

$n \Longleftrightarrow n-1 \Longleftrightarrow n-2 \Longleftrightarrow \ \ldots \ \Longleftrightarrow 1 \Longleftrightarrow 0$

same size.

Bijection Between Exceedences

- Follow path **under diagonal** until it first goes above the diagonal. (It can be in the corner!)
- Mark the intersection with the diagonal.
- Continue following (now **above the diagonal**) until we hit the diagonal again. Mark edge e that occurs before this hit
- Swap the portion before e and the portion after e.



Summary

- All about the **recursive structure**
  - Complicated objects actually have **nice structure**
  - Reduces problem to a **smaller version of itself**
- Can also define **bijections** between interesting objects
  - Lets you count a huge variety of objects