Week 9: Recursive Definition (of Functions) (5.3) & Intro to Counting (6.1, 6.3)

Recursive Definition of Functions:

Goal: Define functions like $f: \mathbb{Z}_{\geq 0} \to \mathbb{R}$

Familiar way: write down an explicit formula

e.g., $f(n) = n^2$.

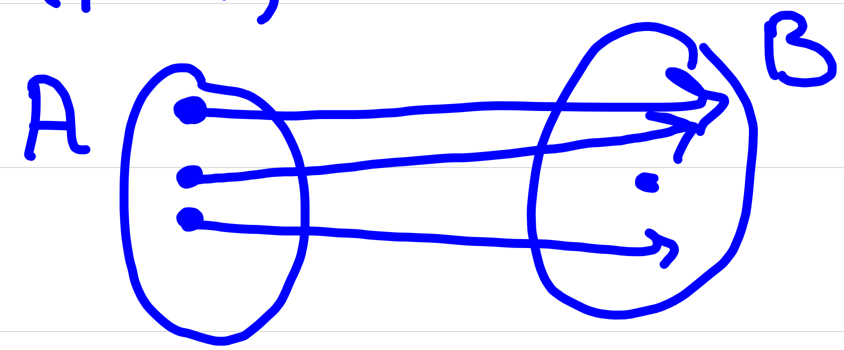
New way (recursive definition):

- **Basis Step:** Define $f(0), f(1), \ldots, f(b)$ for some finite b.

- **Recursive Step:** For any $K \geq b$, Define $f(K+1)$ in terms of $f(0), f(1), \ldots, f(K)$

Example: Fibonacci Numbers. $f: \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$

- Basis step: $f(0) = 0$, $f(1) = 1$
- Recursive step: For any $k \geq 1$, $f(k+1) = f(k) + f(k-1)$



$f(0)$	$f(1)$	$f(2)$	$f(3)$	$f(4)$	
0	1	1	2	3	...

It is a valid way of defining the function!

(every n eventually appears on this list, i.e., this is a mapping from each element of domain to an element of codomain)

(Q) Is this a circular defn? NO

Example:

Basis Step: $f(0) = 1, \quad f(1) = 1$

Recursive Step: $f(k+1) = f(k+2) + f(k+3) \quad \forall k \geq 1$

$f(0)$	$f(1)$	$f(2)$	$f(3)$	$f(4)$
1	1			

? ? ? ?

Example: Basis Step: $f(0) = 1$

Recursive Step: for all $k \geq 0$, $f(k+1) = \sqrt{1 + f(k)}$

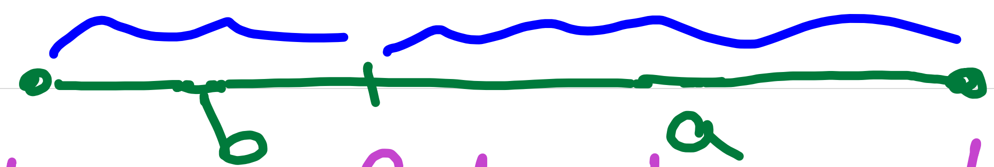
$f(0)$	$f(1)$	$f(2)$	$f(3)$	
1	$\sqrt{2}$	$\sqrt{1+\sqrt{2}}$	$\sqrt{1+\sqrt{1+\sqrt{2}}}$	$\dots$

- As you may have noticed, this way of recursive definition of functions is closely related to induction.
- In fact, we can prove nice properties about these functions with induction.

EX: Fibonacci Numbers are related Golden ratio.

Golden ratio: An age old interested number.

Consider a rod of unit length. For some reason ancient people were interested in a very specific way of splitting the rod such that:



$$\frac{a}{b} = \frac{a+b}{a}, \quad a,b>0$$

this ratio is called the "Golden

Let's find the golden ratio:

Solve it for $x = \frac{a}{b}$: $\quad x = 1 + \frac{1}{x} \xrightarrow{x>0} x^2 = x+1$

$$\longrightarrow x^2 - x - 1 = 0$$

$$x = \frac{1 \pm \sqrt{5}}{2}$$

$$\longrightarrow \frac{a}{b} = \frac{1+\sqrt{5}}{2} \approx 1.618..$$

Observe that $\phi^2 = \phi + 1$

Golden Ratio

ϕ

Fibonacci Numbers and Golden Ratio:

■ **Theorem:** for any $n \geq 3$, $f(n) > \phi^{n-2}$.

Proof: We proceed by strong induction.

Base case ($n=3$): observe that $2 > \phi^1 \approx 1.6\ldots$.

Inductive step: Assume arbitrary $n \geq 3$. Assume that for any $0 \leq k \leq n$, $f(k) > \phi^{k-2}$. We want to show that $f(n+1) > \phi^{n-1}$.

$$f(n+1) = f(n) + f(n-1) > \phi^{n-2} + \phi^{n-3} = \phi^{n-3}(\phi+1) = \phi^{n-3} \cdot \phi^2 = \phi^{n-1}$$

as desired. $\square$

Observe that: ϕ is the largest number s.t.

$$\phi^{k-3} + \phi^{k-2} \geq \phi^{k-1} \quad \forall\, k \geq 3.$$

You can also show that $\lim_{k \to \infty} \frac{f(k+1)}{f(k)} = \phi$.

You can also show that $f(n) \leq 2^n$. <homework>

Counting

Big picture: So far: Logic, Sets, Functions, Number Theory
Proof, Induction

Main achievements: (Definitions ... Theorems)

1. Built everything from first principles
2. Leared how to creat Mathematics
3. How to read/write Proofs

Next: Counting, Probability

I) Counting Rules: product & Sum

II) Substraction III) Division

(6.1, 6.3)

Counting

Goal: Count the number of objects with some specified properties. $\equiv$ Find the cardinality of $S = \{x : x \text{ has some property}\}$

EX1: Bit strings of length 10 $S = \{(b_1, \ldots, b_{10}) : b_i \in \{0,1\}\}$

EX2: Bit string of length 5 ending in 00 and starting 1.

EX3: # ranking of the set {UofT, UW, UBC, McGill}

EX4: # Ordered pairs of 2 distinct cards from a deck 52

EX5: Unordered set of 5-card hands from a deck of 52.

Ex1: Bit strings of Length 10

Naive way: List all of them:

0000000000
0000000001
0000000010
⋮

Better way: This is a set of object. We ask how do we construct an object in this Set, i.e.,
"What choices do we need to make to uniquely specify an object of this set."

The question we need to answer is:
What is the process that can generate the element of the set

EX1: Cnt'd

Observe that every $\underline{b} \in S$ is uniquely specified by a sequence of 10 choices.

first choice: C_1 : choose $b_1 \in \{0,1\}$

C_2 : Given b_1, choose $b_2 \in \{0,1\}$

C_3 ; Given b_1 and b_2, choose $b_3 \in \{0,1\}$

$\vdots$

C_{10} : Given previous choices, choose $b_{10} \in \{0,1\}$

What that "every object is uniquely specified" mean?

- after we make the choices there's only one string and every string can be constructed with this process.

ie., for any given string there's only way to make the choices

EX1: cnt'd

Upshot:

- We have broken down the complicated task of choosing an element of this big set into a bunch of small tasks of making little choices of one bit at a time.
- And for these little choices, it is very easy to count the number of ways to make that choice

How many ways of making the first choice, i.e., C_1? 2

How many ways of making C_2? 2

How many ways of making C_3? 2

$\vdots$

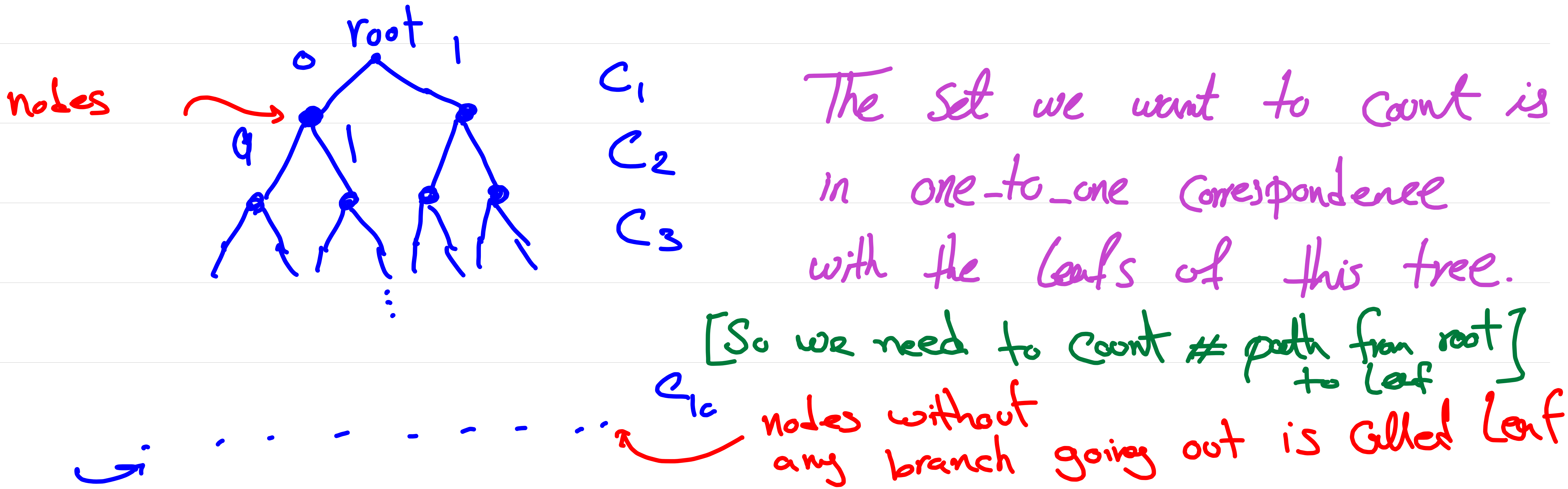
How many ways of making C_{10}? 2

EX1: Cnt'd

Total number of making all choices $c_1, c_2, \ldots, c_{10}$ is the product of the # ways to make each of them:

$$2 \times 2 \times \cdots \times 2 = 2^{10} = 1024.$$

A good visualization tool to demontrate this is Tree Diagram (Suitable for better understanding but should not be used as a proof).

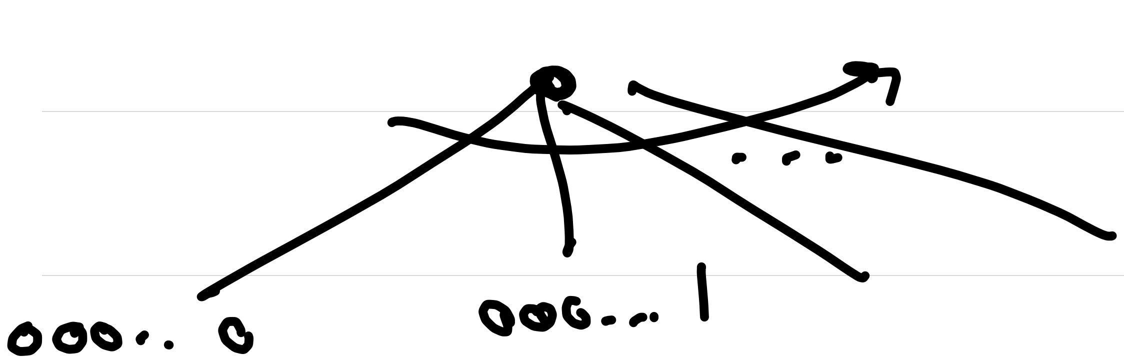


The set we want to count is in one-to-one correspondence with the leafs of this tree.

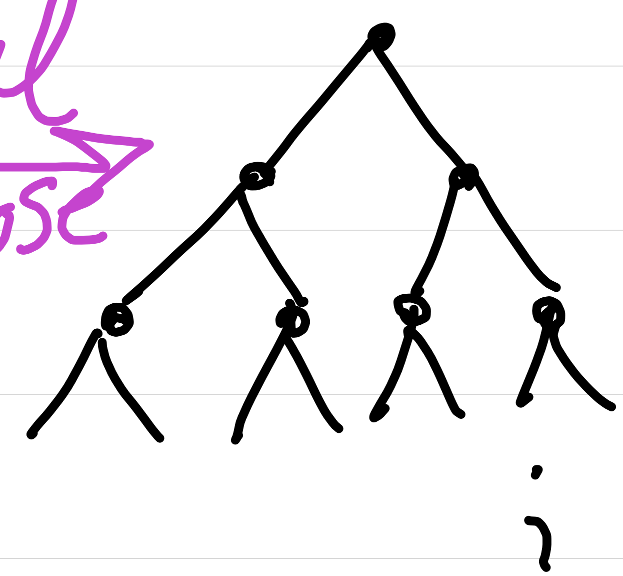
[So we need to count # path from root to leaf]

Idea:

Decompose a big choice into many little choices.



Vertical Decompose



Let's do a more interesting example:

EX4: Ordered pairs of 2 distinct cards from a deck of 52.

Step 1: Identify a sequence of choices for uniquely specifying an object with the desired property.

C_1: choose the first card

C_2: Given the first card choose the second card.

Step 2: Find # of ways to mak each choice.

$C_1: 52$ $\quad C_2: 51$

Step 3: Total # ordered pairs = total # choices $= 52 \times 51$.

Question : Why wasn't it 52×52 ?

Answer : Because repetition were not allowed.

(after choosing the first card, it will be removed the deck)

You must always be aware if ...

Repetition is allowed or not

Product rule: Suppose each object with the desired property can be uniquely specified by a sequence of k choices $C_1, C_2, \ldots, C_k$ and the number of ways to make C_i is n_i for any $1 \leq i \leq k$. Then the total # of objects is $n_1 \times n_2 \times \cdots \times n_k$.

tree diagram

"vertical" decomposition of big choice into smaller choices

Given the Product rule, how do you Solve a Counting problem?

Step 1: Find a sequence of choices $C_1, \ldots, C_K$ uniquely specifying an object

Step 2: Count # ways to make each C_i given previous choices

$C_1 : n_1 \quad C_2 : n_2 \quad \ldots \quad C_K : n_K$

Step 3: Combine using product rule

$$\# \text{ total} = n_1 \times n_2 \times \cdots \times n_K$$