WK 6-1

REVIEW PROOF BY INDUCTION: CH5!

- we make a claim about N, prove N+1 — LADDER PROOF!
- BASIS STEP / INDUCTIVE HYPOTHESIS

TIP :)

. . -2 -1 0 1 2 ...

NON-POSITIVE NON-NEGATIVE

NEGATIVE POSITIVE

EXAMPLE:

LET $a_n = 2a_{n-1} + 1$ for $n \geq 1$; $a_0 = 0$

USING THIS FORMULA

n	a_n	
0	0	(given)
1	1	2(0) + 1
2	3	2(1) + 1
3	7	2(3) + 1
4	15	2(7) + 1
5	31	2(15) + 1

we want a **CLOSE FORM SOLUTION**

↳ a_n = [something in terms of n, but not other terms in the sequence]

NOW LET'S DEFINE IT IN TERMS OF n FOR A CLOSED FORM SOLUTION

$a_n = 2^n - 1$

$\{a_n\}$ = A SEQUENCE. FOR EACH VALUE n, a_n IS SOME INTEGER

↳ IN THIS CASE:

$\{a_n\} = \{0, 1, 3, 7, 15, 31\}$

- defined by $2(a_{n-1}) + 1$

LET'S USE **PROOF BY INDUCTION TO PROVE THIS:**

- **DEFINITION** 4 SEQUENCE $\{a_n\}$ = let $a_n = 2a_{n-1} + 1$ for $n \geq 1$; $a_0 = 0$
- **CLAIM:** for any integer $n \geq 0$ $a_n = 2^n - 1$

① **BASIS:**

WHAT DO WE WANT TO PROVE?

↳ 1ST STEP: "IT HOLDS FOR 0"

meaning $a_0 = 2^0 - 1$

LHS		RHS
$a_0 = 0$	=	$2^0 - 1 = 1 - 1 = 0$

by def. of the sequence $\{a_n\}$

since a_0 + $2^0 - 1$ both = 0, $a_0 = 2^0 - 1$ ✓

② **INDUCTIVE STEP:**

ASSUME FOR INDUCTION $a_k = 2^k - 1$

CLAIM: $a_{k+1} = 2^{k+1} - 1$ TRY TO PROVE THIS

STEP 1: LOOK AT CONCLUSION, SEE HOW TO WORK TOWARDS IT

START WITH LHS:

$a_{k+1} = 2 * a_k + 1$ BY DEF. OF THE SEQUENCE

STEP 2: SUB IN WHAT a_k is assumed : $a_k = 2^k - 1$

$a_{k+1} = 2 * (2^k - 1) + 1$ BY INDUCTIVE HYPOTHESIS

STEP 3: SIMPLIFY

$= 2^{k+1} - 2 + 1$ DISTRIBUTION

$a_{k+1} = 2^{k+1} - 1$

PROVING FIBBONACHI SEQUENCE:

$\{f_n\} \rightarrow 1, 1, 2, 3 . .$

DEF. → $f_n = f_{n-1} + f_{n-2}$

$f_0 = 0, f_1 = 1$

n	f_n	$\sum_{i=0}^{n} f_i$ SUM OF FIBBONACHIS
0	0	0
1	1	1
2	1	2
3	2	4
4	3	7
5	5	12
6	8	20
7	13	33

$\sum_{i=0}^{n} f_i = f_{n+2} - 1$

CAN WE PROVE THIS!

① **BASIS:** SHOW $\sum_{i=0}^{0} f_i = f_2 - 1$

LHS	RHS
$\sum_{i=0}^{0} f_i = 0$	$f_2 = 1$ AS CALCULATED BY DEF.
PLUG IN 0 DIRECTLY	$f_2 - 1 = 0$

0 v 0

② **INDUCTIVE STEP:**

ASSUME FOR INDUCTION $\sum_{i=0}^{k} f_i = f_{k+2} - 1$

CLAIM : $\sum_{i=0}^{k+1} f_i = f_{k+3} - 1$

SPLIT OFF LAST ELEM OF SUM

$\underbrace{\sum_{i=0}^{k+1} f_i}_{\text{LHS}} = \underbrace{\sum_{i=0}^{k} f_i + f_{k+1}}_{\text{RHS:}}$

$= (f_{k+2} - 1) + f_{k+1}$ BY INDUCTIVE HYPOTHESIS

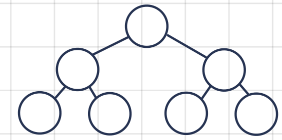
$= (f_{k+2} + f_{k+1}) - 1$ COMMUNITIVE/ ASSOCIATIVE LAW OF +

✓ $= f_{k+3} - 1$ DEF. OF FIBBONACHI

THEREFORE: ALL INTEGERS $n \geq 0$, $\sum_{i=0}^{n} f_i = f_{n+2} - 1$

TREES ~ INDUCTION

CH 5.3 + 11.1
WK 6-2



TREE ~ ITS RECURSIVE DEFINITION:

"RECURSIVE DEFINITION" 4 A FULL BINARY TREE
(THERE'S ANOTHER DEF FOR IT)

THE FULL BINARY TREE

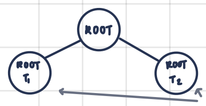
BASIS: A SINGLE VERTEX IS A FULL BINARY TREE, VERTEX = ROOT

RECURSIVE STEP: WE USE PREVIOUSLY KNOWN BINARY TREES TO CONSTRUCT MORE BINARY TREES

IF T_1 + T_2 ARE FULL BINARY TREES, THEN THERE IS A FULL BINARY TREE $T_1 \cdot T_2$

CONSTRUCTED AS FOLLOWS:

- CREATE A NEW VERTEX THAT WILL BE THE ROOT, + DRAW DIRECT EDGES FROM IT TO THE ROOT OF T_1 + ROOT OF T_2



WE JUST USED PREVIOUS FULL BINARY TREE (THE ONE NODE)

TO MAKE A NEW FULL BINARY TREE

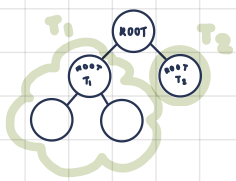
JUST 1
NODE = ROOT

NOW CONSTRUCT ANOTHER FULL BINARY TREE USING KNOWN ONES: WHAT WE KNOW RN → A, B

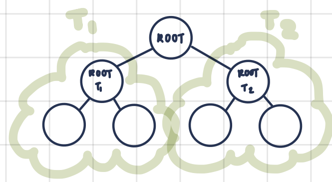
T_1: LET'S USE B

T_2: LET'S USE A

COMBINE IT



IF T_1 + T_2 = B, WE COULD CONSTRUCT THIS:



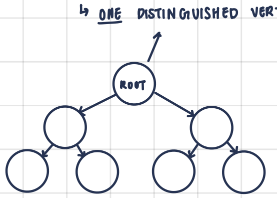
RECURSIVE:

TAKING RESULT OF A FUNCTION N USING IT IN THE NEXT FUNCTION CALL OF THE SAME FUNCTION

EX: FIBBONACHI: USING $\underbrace{f_{n-2} \text{ and } f_{n-1}}_{\text{PREVIOUSLY FOUND}}$ to find f_n

TREE ~ ANOTHER PERSPECTIVE

- A ROOTED TREE = A DIRECTED GRAPH
  - ↳ ONE DISTINGUISHED VERTEX, THE ROOT

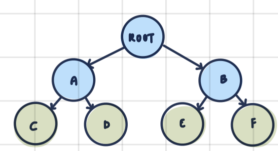


→ ~ there is exactly one directed path from the root to each other vertex

SINCE WE KNOW ALL THE ARROWS POINT AWAY, DON'T NORMALLY DRAW THEM

- THE ROOT HAS NO INCOMING EDGES
- EACH OTHER VERTEX HAS EXACTLY ONE INCOMING EDGE

→ SO IF THERE IS AN EDGE (U, V), WE SAY U IS THE PARENT OF V, V IS THE CHILD OF U



PARENT OF A = ROOT

CHILDREN OF A = C, D

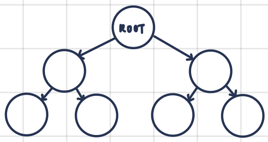
~ DEFINITION HUB:

INTERNAL VERTEX: HAS AT LEAST ONE CHILD - ROOT, A, B

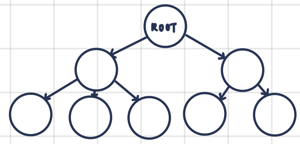
LEAF: NO CHILDREN - C, D, E, F

SO EVERY VERTEX IS AN INTERAL VERTEX / LEAF

N-ARY TREE : A ROOTED TREE IN WHICH EVERY VERTEX HAS AT MOST N CHILDREN



THIS IS A 2-ARY TREE ✓
3-ARY TREE ✓
⋮

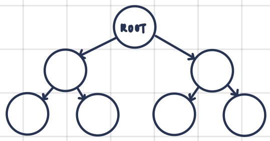


THIS IS A 3-ARY TREE

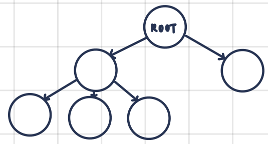
2-ARY = BINARY TREE

FULL 2-ARY = FULL BINARY TREE

A FULL N-ARY TREE : EVERY INTERNAL VERTEX HAS EXACTLY N CHIDREN



THIS IS A FULL 2-ARY TREE ✓
FULL 3-ARY TREE X
⋮



NOT A FULL N-ARY TREE FOR ANY N

RECURSIVE FORMULA 4 # OF STUFF IN A FULL BINARY TREE

BASED ON A RECURSIVE DEFINITION

LET'S DEFINE FUNCTION $v(T)$, the number of **VERTICES** in a full binary tree T =

○ NEED 2 CASES : BASIS / RECURSIVE

#1: if T is a single vertex, then $v(T) = 1$

#2: if $T = T_1 \cdot T_2$, for full bin. tree T_1 & T_2,

then $v(T)$ = [DEPENDS ON T_1 & T_2]

$= v(T_1 \cdot T_2)$

$= 1 + v(T_1) + v(T_2)$

FIND $L(T)$ # OF **LEAVES** IN A FULL BIN. TREE T.

#1 : BASIS

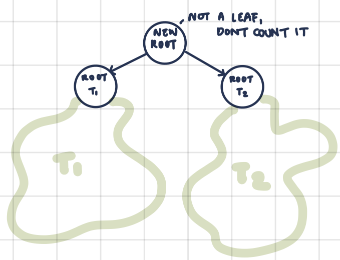
IF T IS A SINGLE VERTEX, THEN $L(T) = 1$

1 VERTICE
1 LEAF
0 INTERNAL VERTICES

#2 : RECURSIVE

IF $T = T_1 \cdot T_2$ for a full bin. tree T_1 + T_2,

then $L(T) = L(T_1 \cdot T_2) = L(T_1) + L(T_2)$



FIND $i(T)$ # OF **INTERNAL VERTICES** in a full bin. tree T:

↳ EZ METHOD : $v(T) - L(T)$:)

BUT, RECURSIVELY →

#1 : BASIS

IF T IS A SINGLE VERTEX $i(T) = 0$

#2 : RECURSIVE

IF $T = T_1 \cdot T_2$ for full bin. tree T_1, T_1 + T_2, then

$i(T) = i(T_1 \cdot T_2) = i(T_1) + i(T_2) + 1$

↑ THE NEW ROOT