Lecture 15 - March 6

Binary Trees

Binary Trees: Math Properties
Tree Traversals

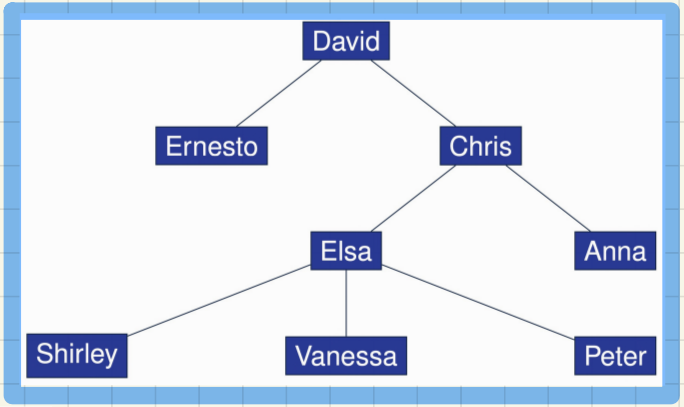
Announcements/Reminders

- Assignment 3 (on linked Trees) released
- WrittenTest
  + guide released
  + example questions to be release
- Makeup Lecture (on ADTs, Stacks) posted
- Lecture notes template, Office Hours, TA Contact

Tracing: Computing a Tree's Height

```
public int height(TreeNode<E> n) {
  TreeNode<E>[] children = n.getChildren();
  if(children.length == 0) { return 0; }
  else {
    int max = 0;
    for(int i = 0; i < children.length; i ++) {
      int h = 1 + height(children[i]);
      max = h > max ? h : max;
    }
    return max;
  }
}
```

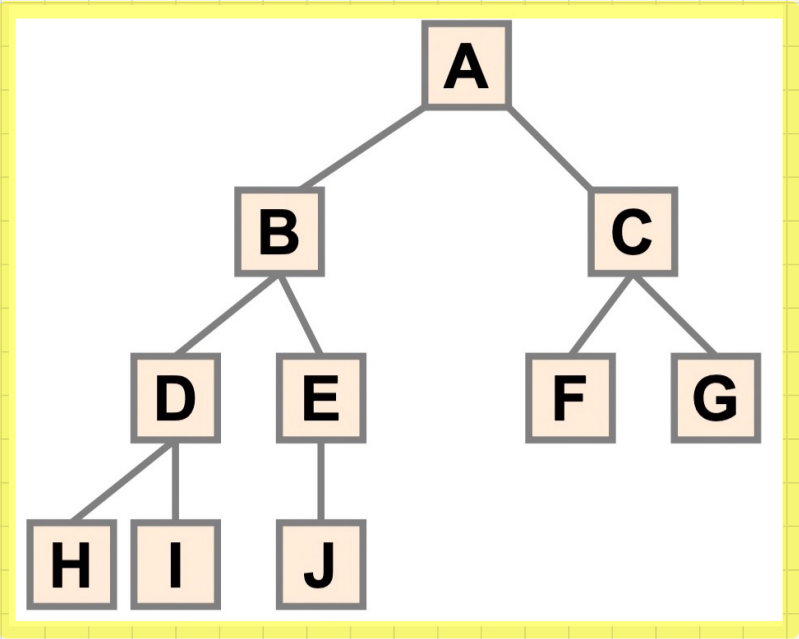
noC



height(chris)

```
@Test
public void test_general_trees_heights() {
  ... /* constructing a tree as shown above */
  TreeUtilities<String> u = new TreeUtilities<>();
  /* internal nodes */
  assertEquals(3, u.height(david));
  assertEquals(2, u.height(chris));
  assertEquals(1, u.height(elsa));
  /* external nodes */
  assertEquals(0, u.height(ernesto));
  assertEquals(0, u.height(anna));
  assertEquals(0, u.height(shirley));
  assertEquals(0, u.height(vanessa));
  assertEquals(0, u.height(peter));
}
```

BT Terminology: LST vs. RST



Strategy of Recursion on BT:

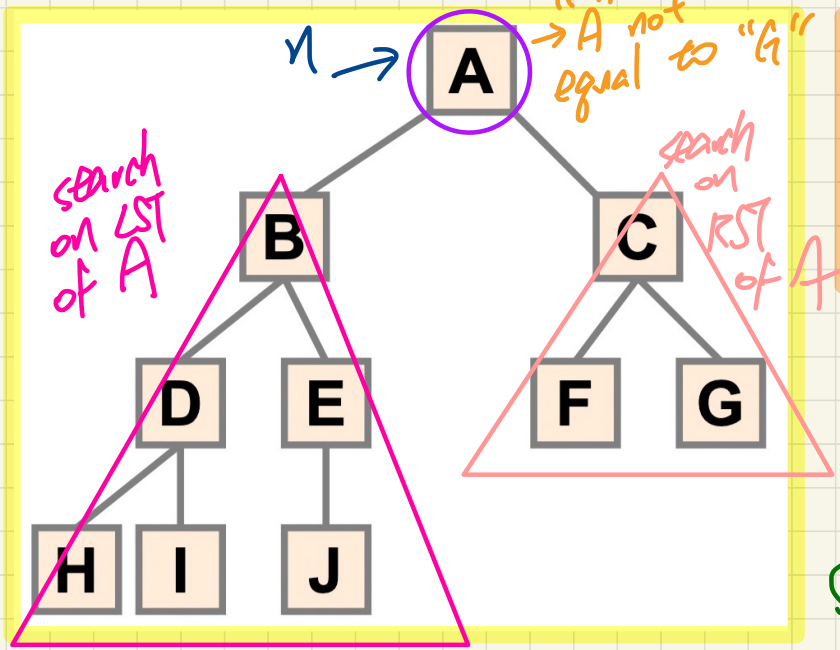
+ Do something on root
+ Recur on LST
+ Recur on RST

e.g.,

+ counting size

BT Terminology: LST vs. RST

search(A, "G") → T
search(A, "Z") → F



Strategy of Recursion on BT:

+ Do something on root
+ Recur on LST
+ Recur on RST

e.g.,

+ searching item

search(n, e) n: tree node, e: data ele.

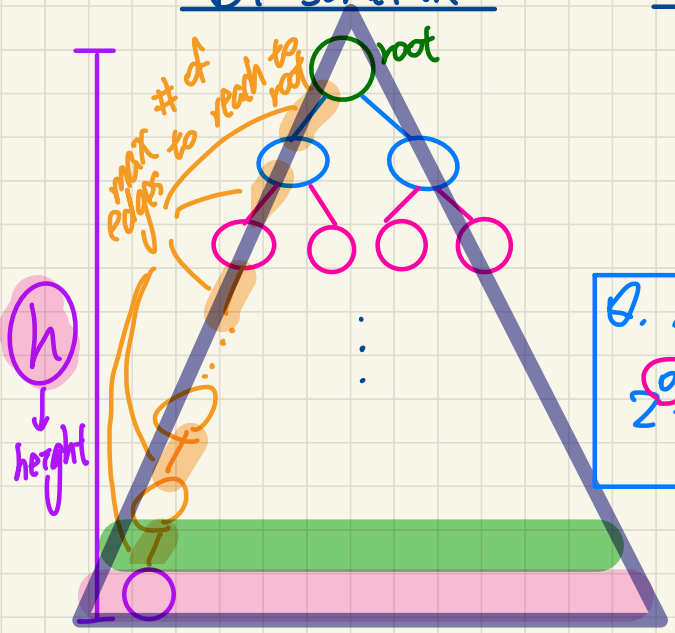
```
① if (n.element.equals(e)){
      return true;
  }
② if(n.left != null){
      return search(n.left, e);
  }
③ if(n.right != null){
      return search(n.right, e);
  }
④ return false; // if n is external, n.element ≠ e
```

BT Terminology: Depths, Levels, Max # of Nodes

height.

$S_k = \frac{I \cdot (r^k - 1)}{r - 1}$ BT: $\frac{1 \cdot (2^k - 1)}{2 - 1} = 2^k - 1$ *

BT structure



Level ??

d

0

1

2

Exercise
Max # of nodes from Level 0 to Level h-1?

Max # nodes at Level ??

$1 = 2^0$

$2 = 2^1$

$4 = 2^2$

Q. Max # of nodes in a BT with height h?

$2^0 + 2^1 + 2^2 + \cdots + 2^{h-1} + 2^h = 2^{h+1} - 1$

$h - 1$

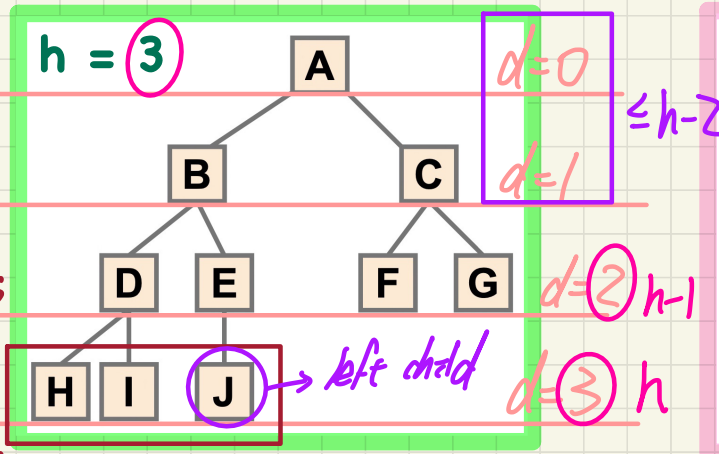
h

2^{h-1}

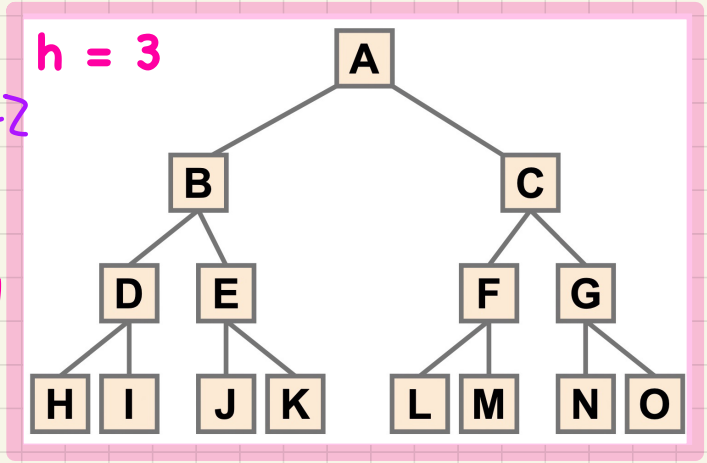
2^h

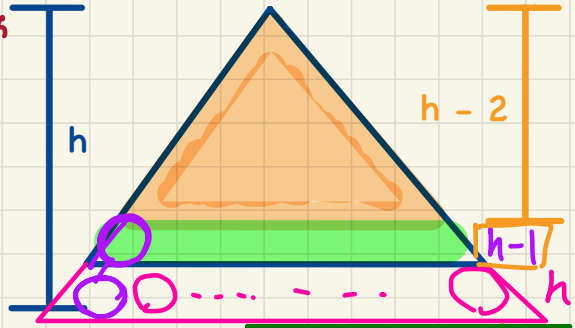
BT Terminology: Complete vs. Full BTs

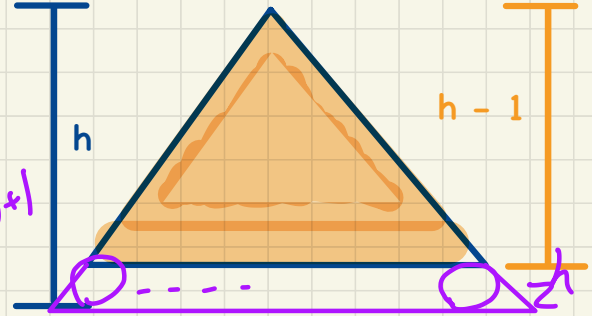
$S_k = 2^k - 1$



- nodes with d = h
- children of nodes at level h-1







Min # nodes? $(2^0 + 2^1 + \cdots + 2^{h-1}) + 1$

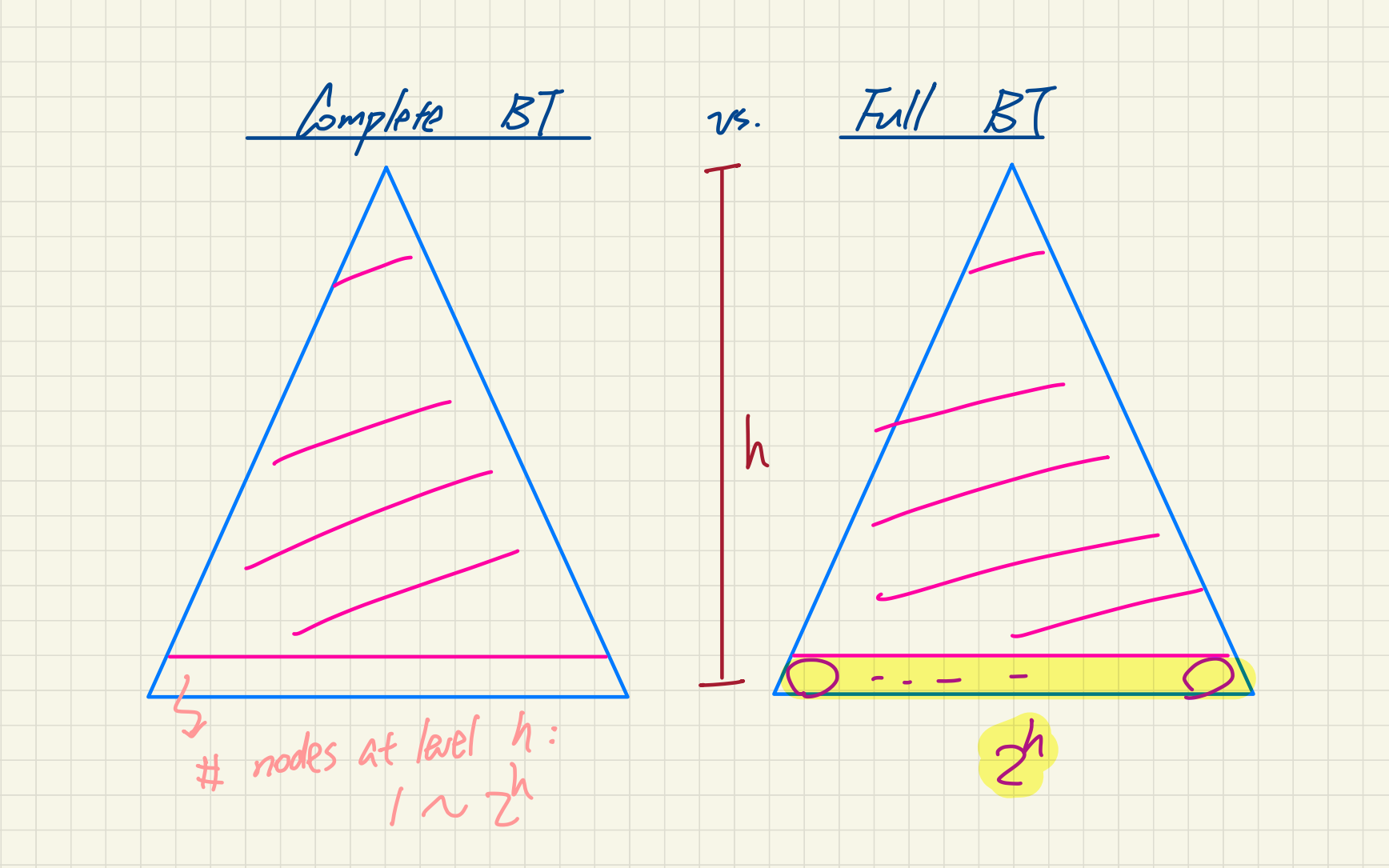
Max # nodes? $(2^0 + 2^1 + \cdots + 2^{h-1}) + 2^h$

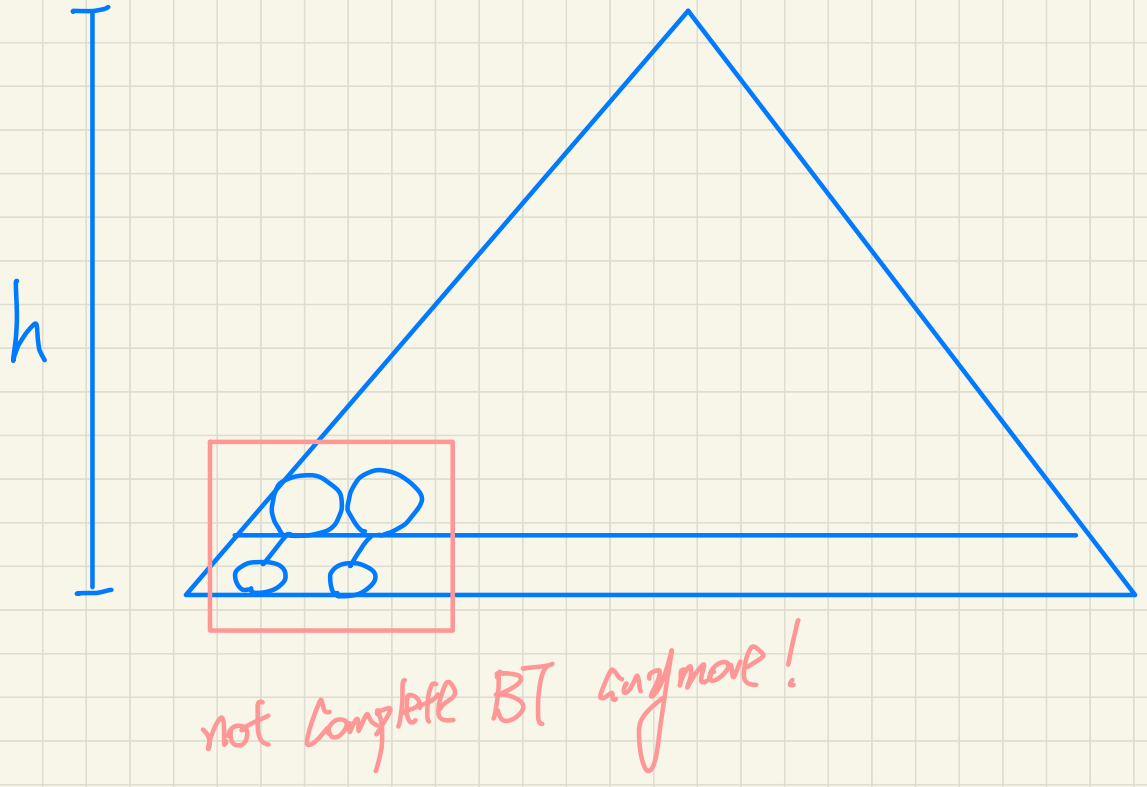
$(2^h - 1) + 1 = 2^h$

$2^{h+1} - 1$

Min # nodes?

Max # nodes? $2^0 + 2^1 + \cdots + 2^h$


Complete BT
vs.
Full BT
h
nodes at level h:
$1 \sim 2^h$
2^h



not Complete BT anymore!

BT Properties: Bounding # of Nodes

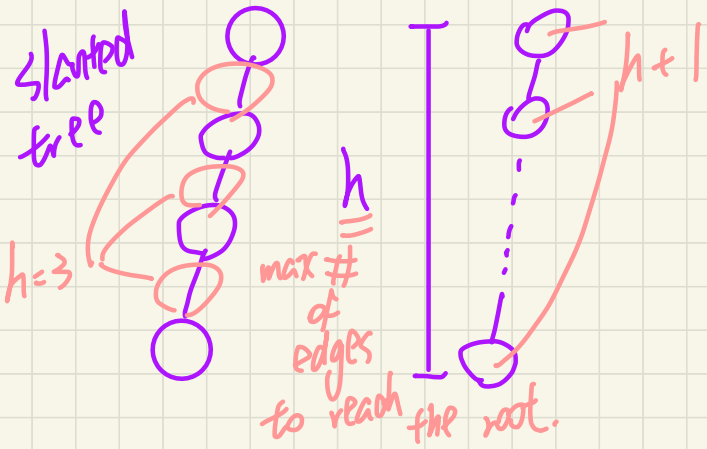
Given a *binary tree* with ***height*** *h*, the ***number of nodes*** *n* is bounded as:

$$h + 1 \leq n \leq 2^{h+1} - 1$$

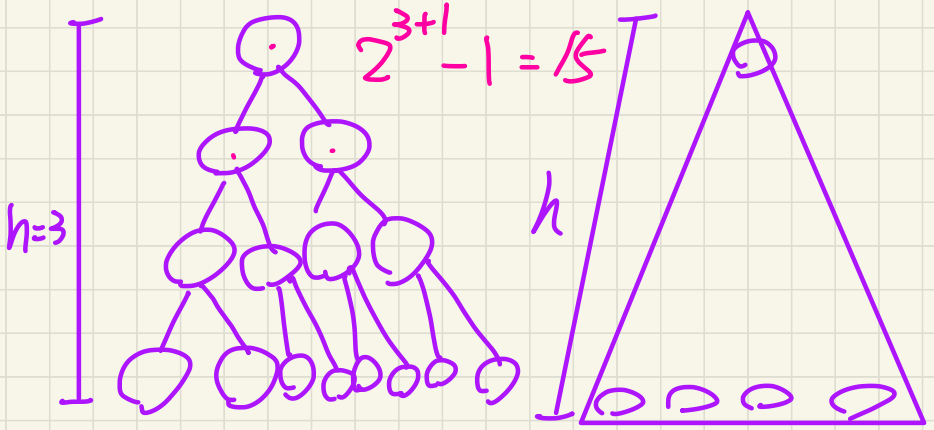
For example, say h = 3 → max depth is 3

$2^0 + 2^1 + \cdots + 2^h = 2^{h+1} - 1$

Minimum # of nodes



Maximum # of nodes



BT Properties: Bounding Height of Tree

$\log z^x = x$

Given a *binary tree* with *n* **nodes**, the ***height*** *h* is bounded as:

$n = 7$

max: $\log(7+1) - 1 = 2$

$$\log(n+1) - 1 \le h \le n - 1$$

For example, say n = 7

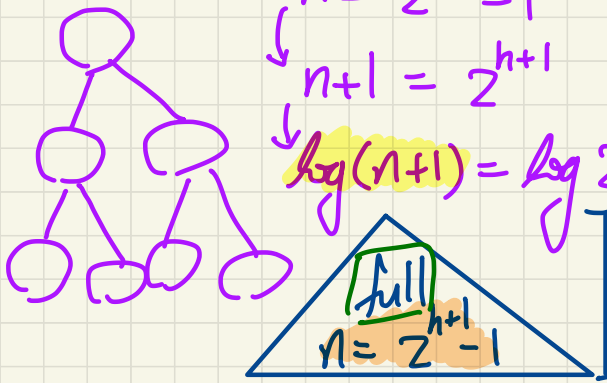
Minimum height → # nodes is max for that min height.

$n = 2^{h+1} - 1$

$n+1 = 2^{h+1}$

$\log(n+1) = \log 2^{h+1} = h+1$

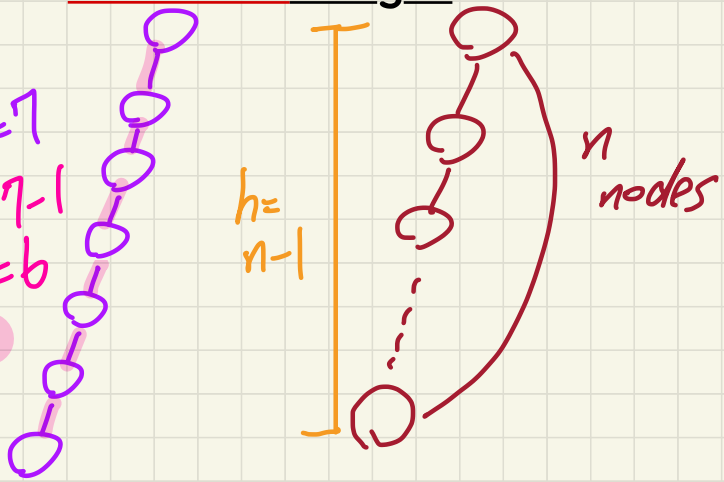
$h = \log(n+1) - 1$ min



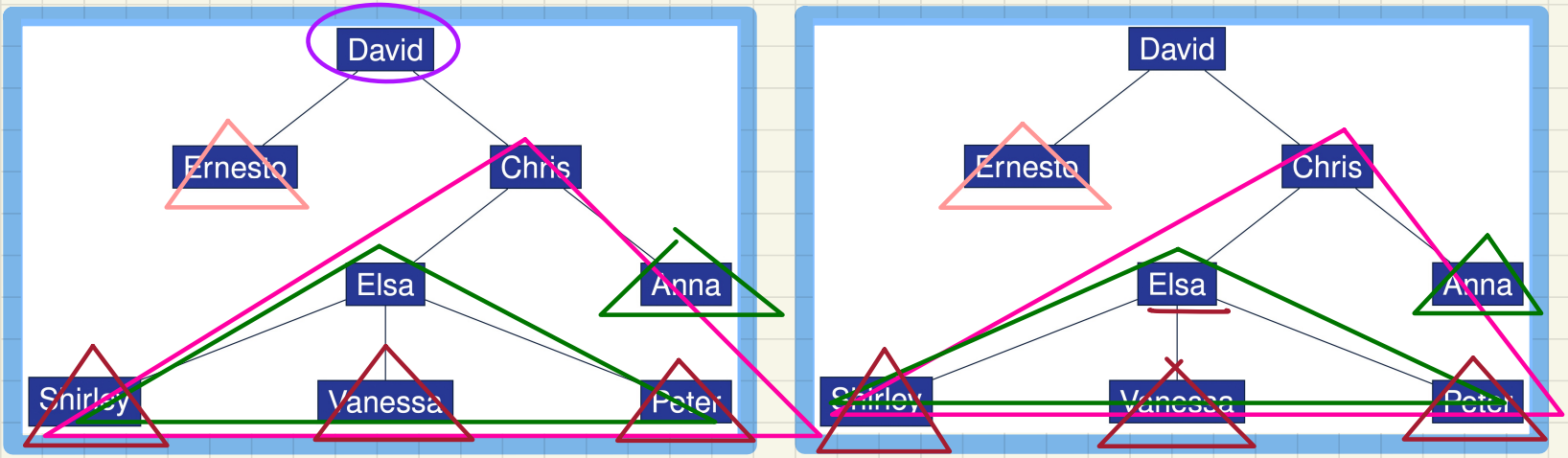
Maximum height

$n = 7$

$h = 7 - 1 = 6$



General Tree Traversals: Pre-Order vs. Post-Order



Pre-Order Traversal from the Root

David Ernesto Chris Elsa S V P Anna

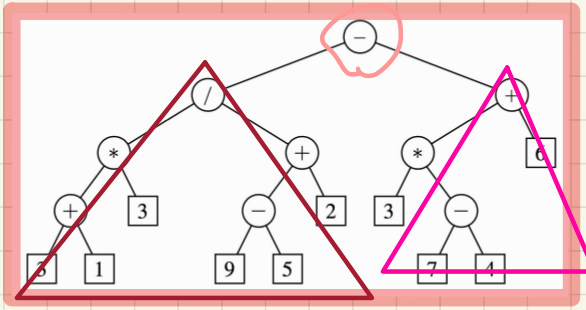
po(E) po(Chris) po(Anna)

Post-Order Traversal from the Root

Ernesto S V P Elsa Anna Chris David

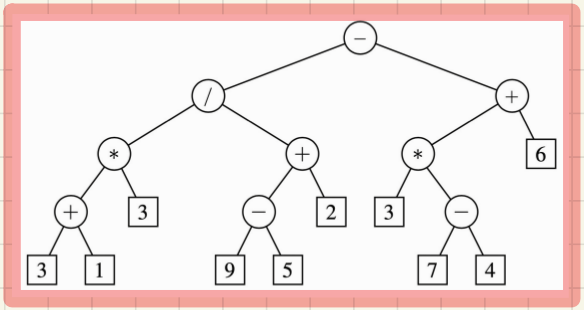
po(E) po(Elsa) po(Anna) po(Chris)

Binary Tree Traversals

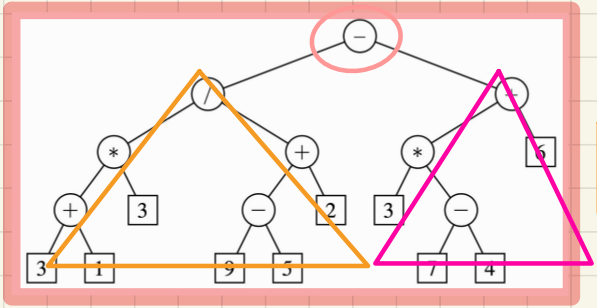


Pre-Order Traversal

− / * + 3 1 3 + − 9 5 2 + * 3 − 7 4 6



In-Order Traversal



Post-Order Traversal

3 1 + 3 * 9 5 − 2 + / 3 7 4 − * 6 + −