Lecture 15 - March 6

Binary Trees

Binary Trees: Math Properties
Tree Traversals

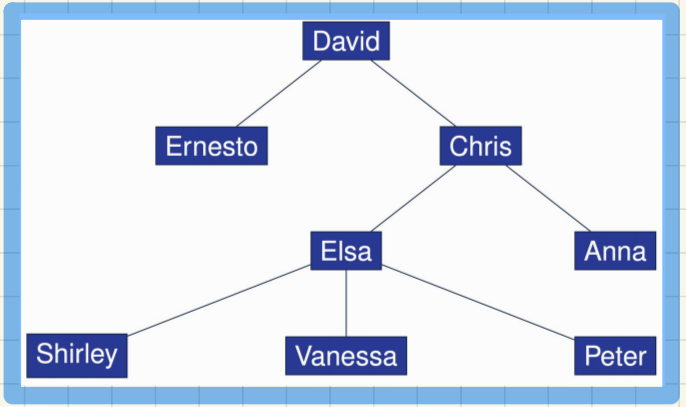
Announcements/Reminders

- 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;
  }
}
```

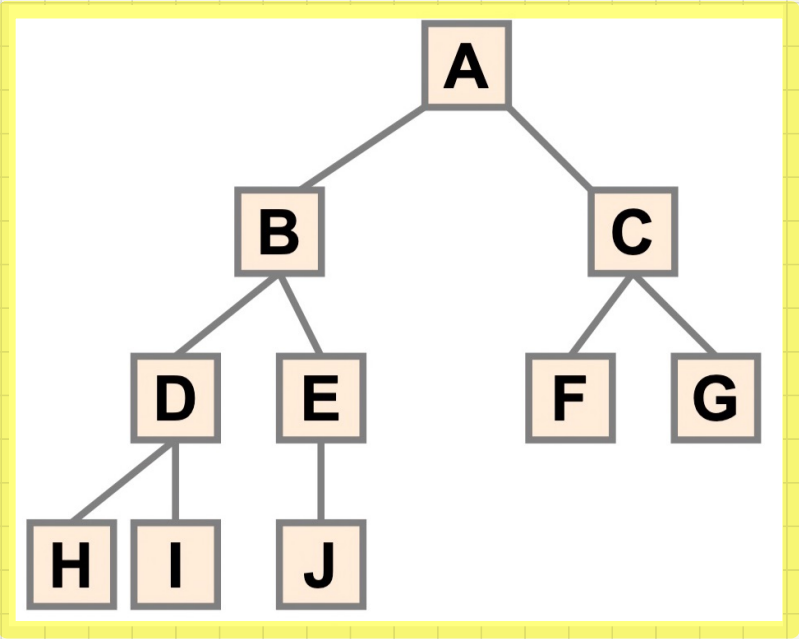
not



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



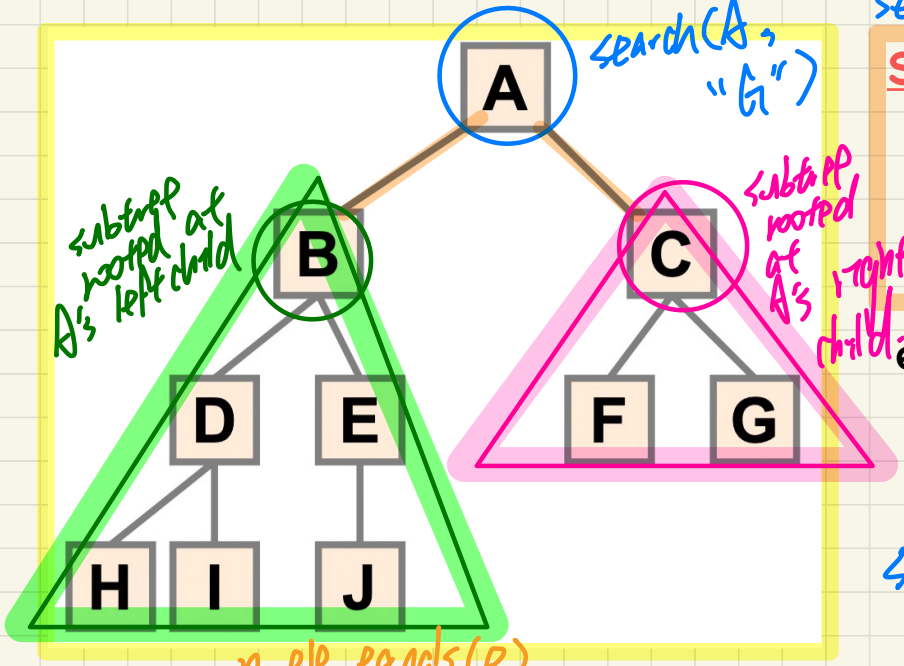
Strategy of Recursion on BT:

+ 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 item

↳ if n is external → n.data.equals(e)

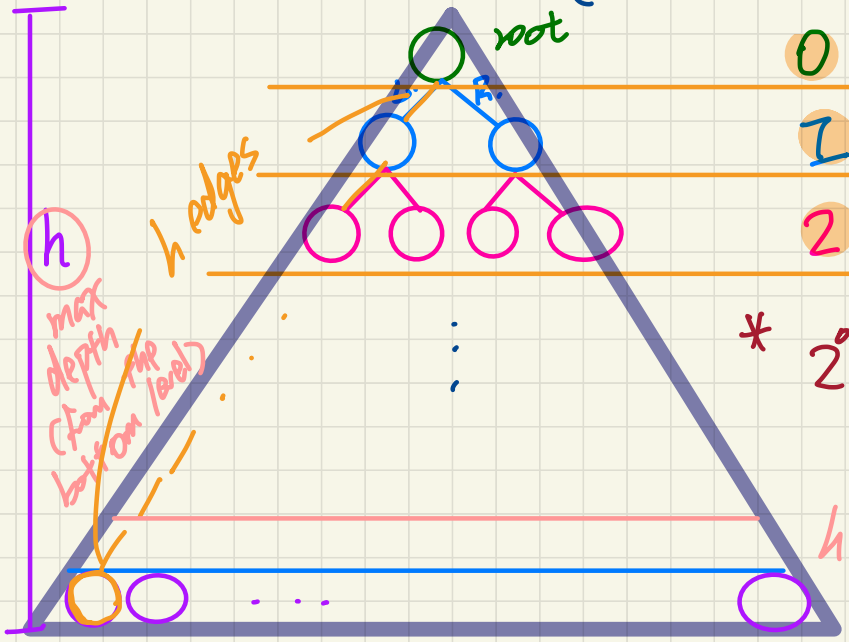
else /* n is internal */

n.ele.equals(e)
||
return search(A.left, e)
||
search(A.right, e)

BT Terminology: Depths, Levels, Max # of Nodes

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

* Max # of nodes in BT with height h



BT structure

Level ? (d)

Max # Nodes at Level ?

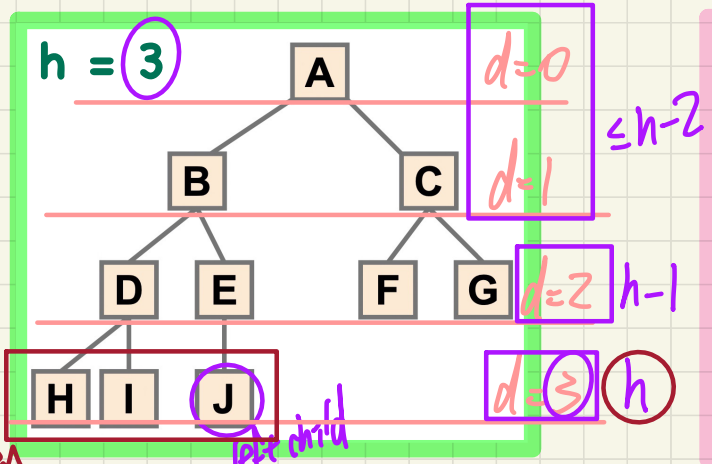
Level	Max # Nodes at Level
0	$1 = 2^0$
1	$2 = 2^1$
2	$4 = 2^2$
h-1	2^{h-1}
h	2^h

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

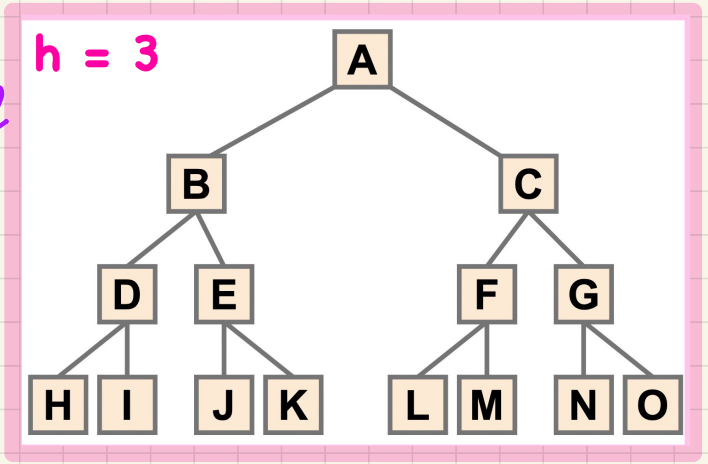
Exercise: Max # of nodes from level 0 to level h-1?

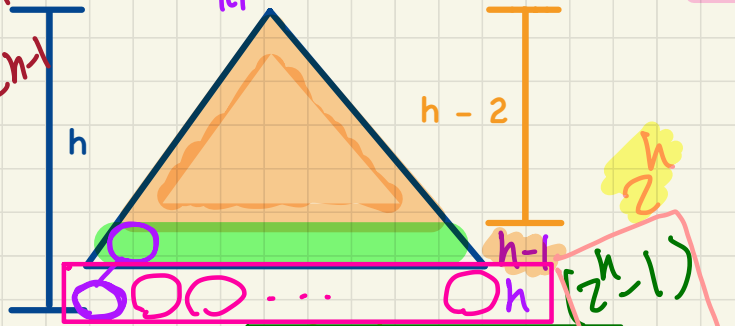
BT Terminology: Complete vs. Full BTs

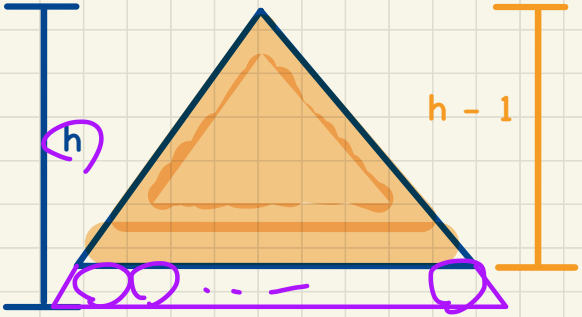
$\sum k = 2^k - 1$



- # nodes with $d = h$
- children of nodes with $d = h-1$







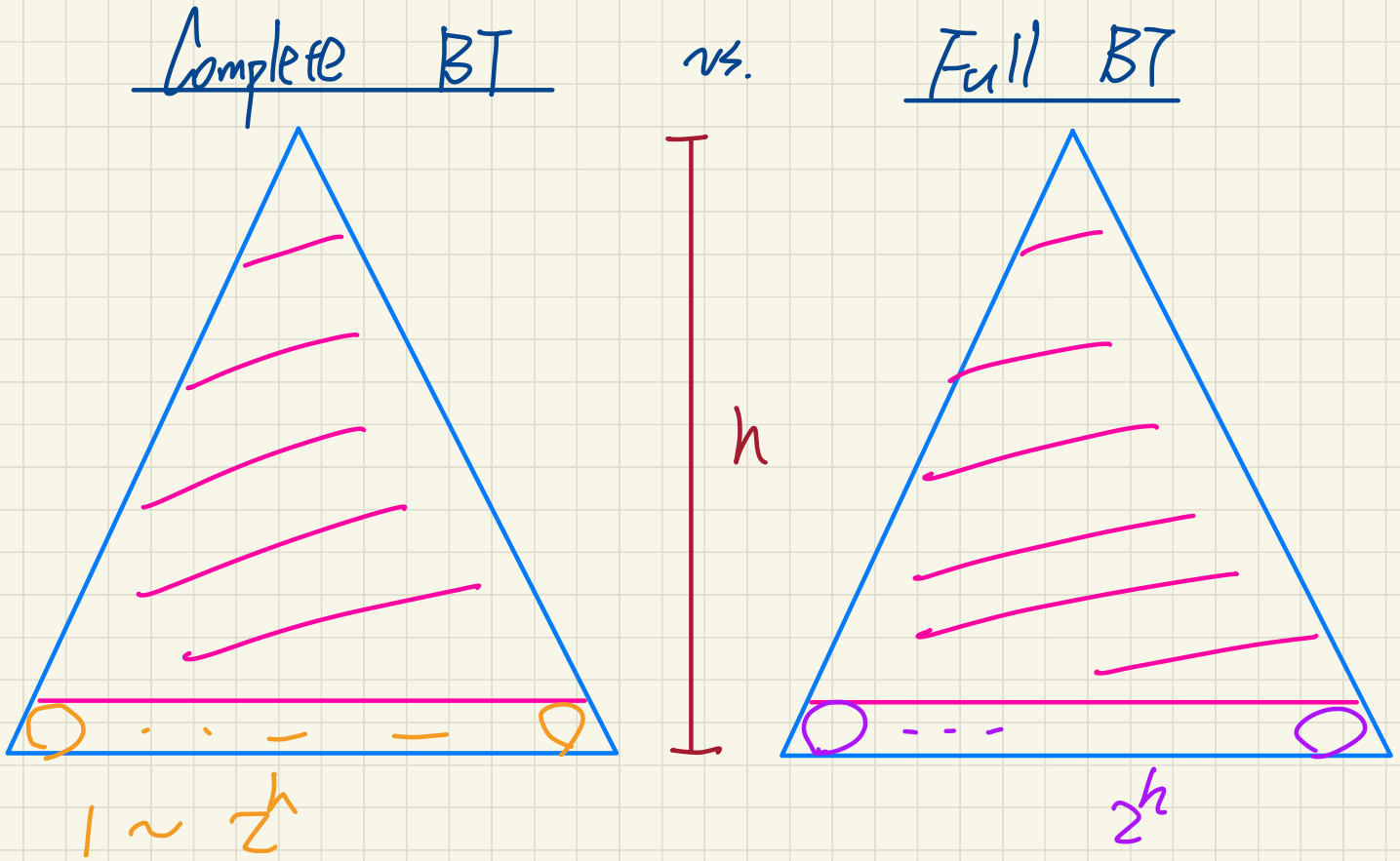
Min # nodes? $(2^0 + 2^1 + \cdots + 2^{h-1}) + 1$

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

$2^{h+1} - 1$

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

Max # nodes?


Complete BT
vs.
Full BT
h
1 ~ 2^h
2^h

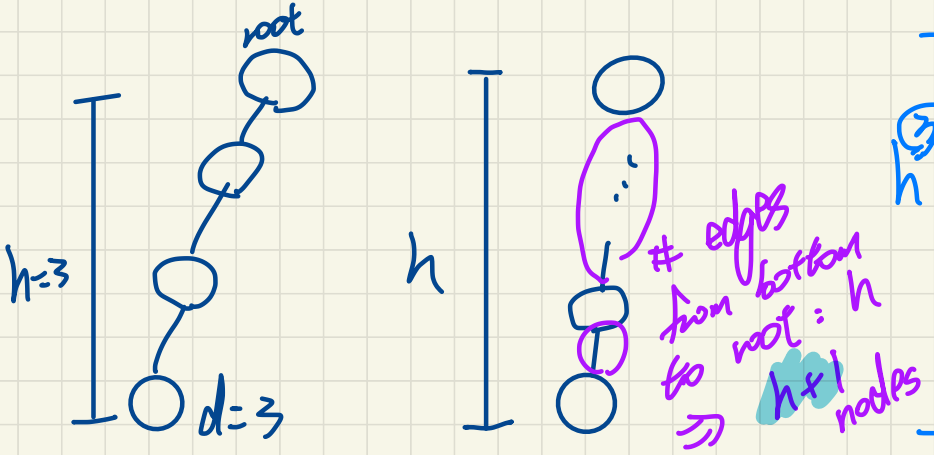
BT Properties: Bounding # of Nodes

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

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

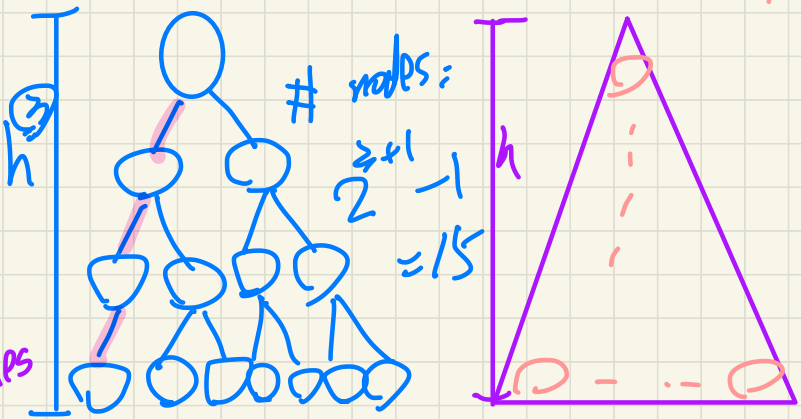
For example, say h = 3

Minimum # of nodes



Maximum # of nodes

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



BT Properties: Bounding Height of Tree

$\log 2^{h+1} = h+1$

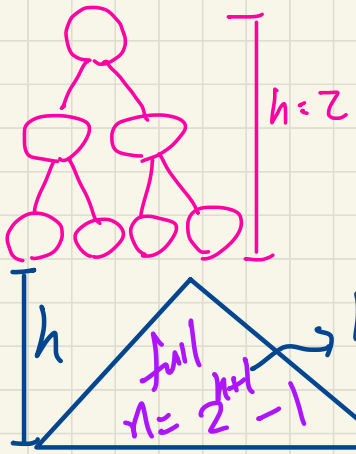
$\log 2^3 = 3$

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

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

For example, say n = 7

Minimum height



h is min if the BT is full.

min height $= \lceil \log(7+1) \rceil - 1$

$= 2$

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

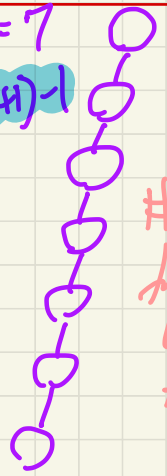
$n = 2^{h+1} - 1$

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

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

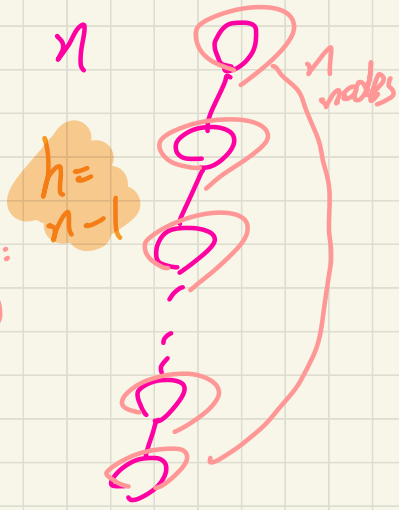
Maximum height

$n = 7$

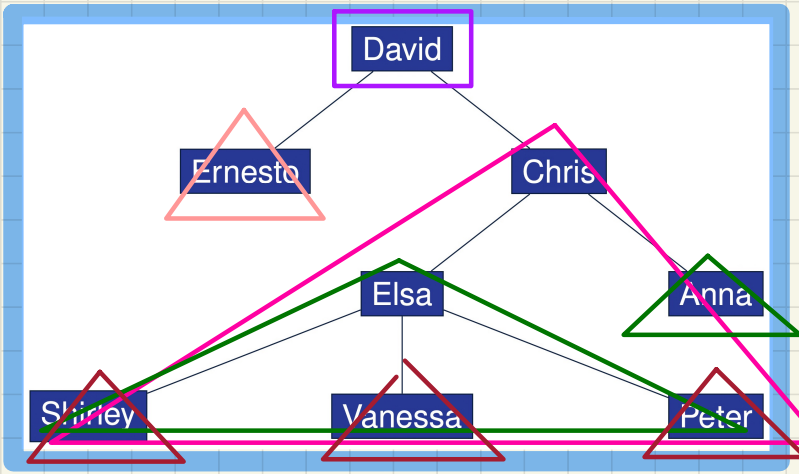


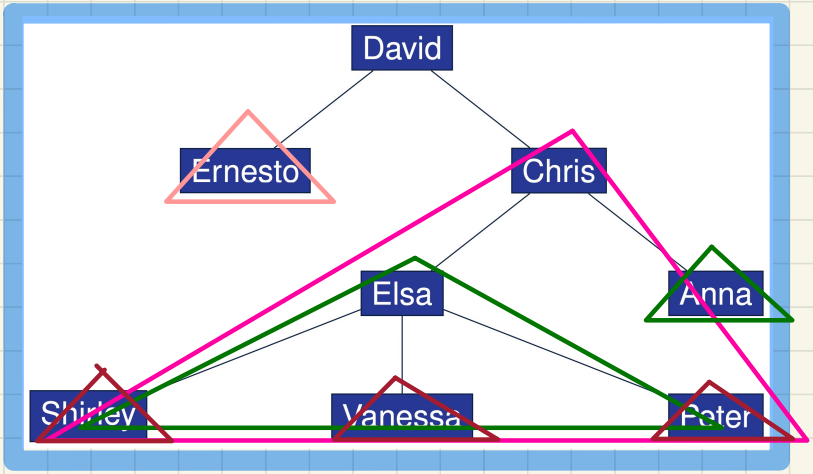
edges from bottom: (max depth)

$7 - 1 = 6$



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(Elsa) po(Anna) po(Chris)

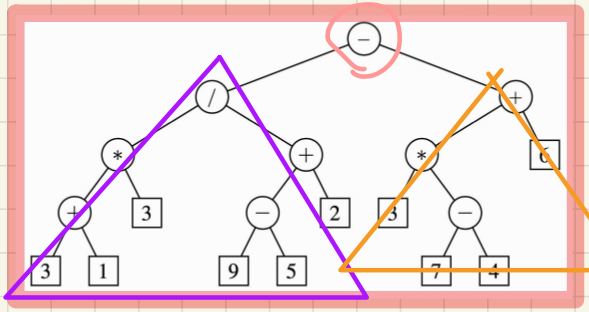
Post-Order Traversal from the Root

S V P Elsa Anna Chris David

Ernesto po(Elsa) po(1) po(E) po(C)

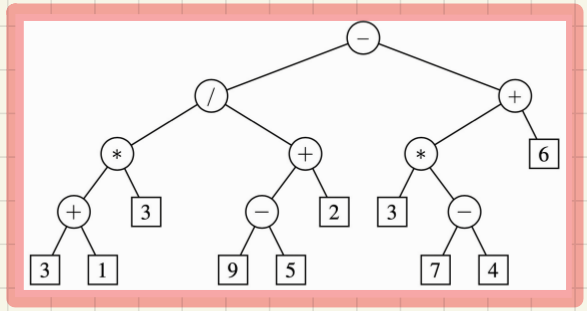
TRAVEL THE WORLD

Binary Tree Traversals

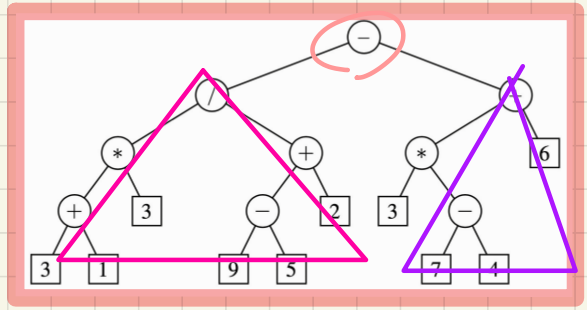


Pre-Order Traversal

− / * + 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 + −