# Lecture 16 - March 11

## Binary Trees, Binary Search

***Bounding Internal vs. External Nodes***
***Proper Binary Trees***
***Binary Search: Ideas, Java***

## Announcements/Reminders

- Assignment 3 (on linked Trees) released
- WrittenTest guide & example questions released
- WrittenTest review session materials posted
- Makeup Lecture (on ADTs, Stacks) posted
- Lecture notes template, Office Hours, TA Contact

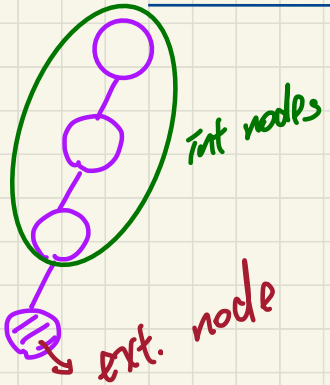
# BT Properties: Bounding # of External Nodes

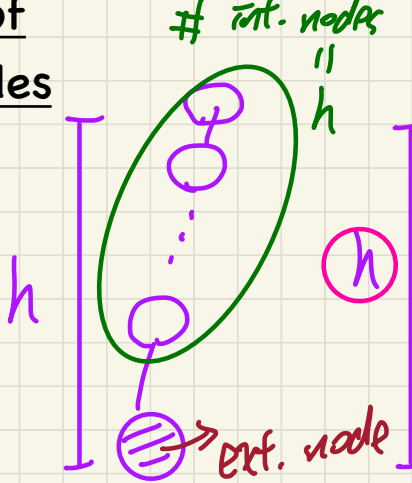
Given a *binary tree* with ***height*** $h$, the ***number of external nodes*** $n_E$ is bounded as:

$$1 \leq n_E \leq 2^h$$

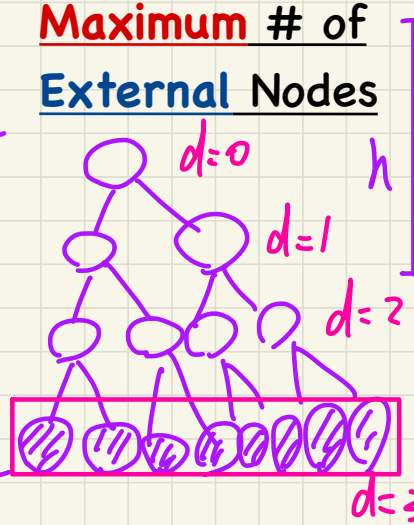
**For example, say h = 3** → max # of edges from bottom node to root.

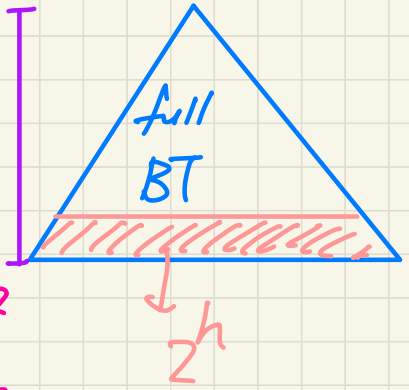
**Minimum # of External Nodes**




**Maximum # of External Nodes**



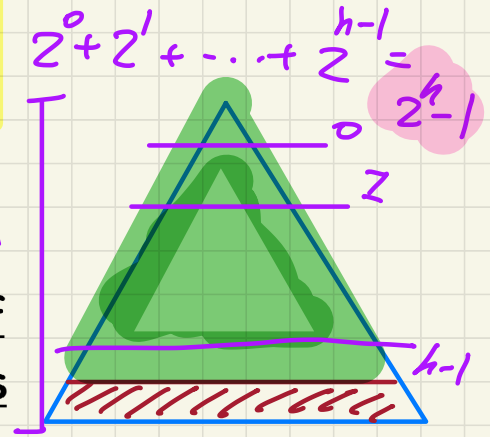



# BT Properties: Bounding # of Internal Nodes $S_k = 2^k - 1$

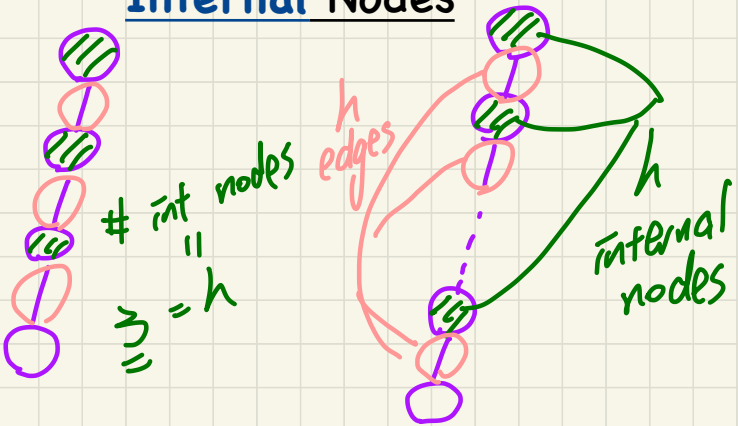
Given a *binary tree* with ***height*** $h$, the ***number of internal nodes*** $n_I$ is bounded as:

$$h \leq n_I \leq 2^h - 1$$

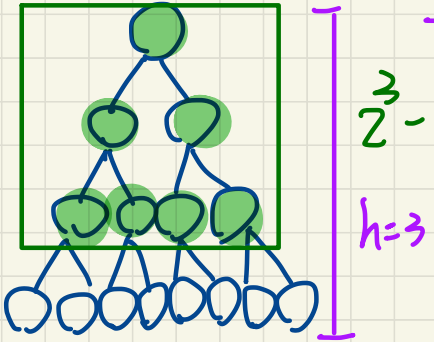
**For example, say h = 3**




**Minimum # of Internal Nodes**




**Maximum # of Internal Nodes**



# BT Properties: Relating #s of Ext. and Int. Nodes

before the ext ← $n_I$, $n_E$ | $n_I'$, $n_E'$ after ext

Given a *binary tree* that is:

- ***nonempty*** and ***proper***
- with $n_I$ ***internal nodes*** and $n_E$ ***external nodes***

We can then expect that: $n_E = n_I + 1$

## Induction on Size of Proper BT

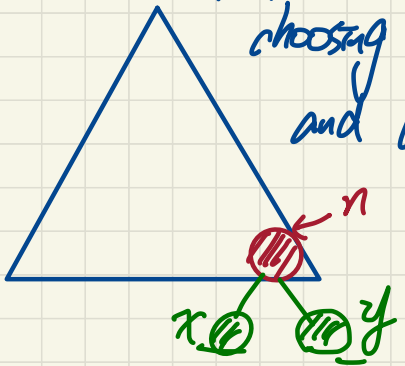
① Base Case: Smallest Proper BT REVIEW

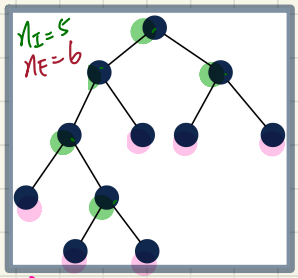
$n_E = 1$, $n_I = 0$ ) property holds

② Inductive Hypothesis (I.H.):

For a proper BT of size > 1: $n_E = n_I + 1$

③ Given a proper BT, extend it by choosing the right-most ext. node and converting it to an int. node with child nodes (x) and (y).



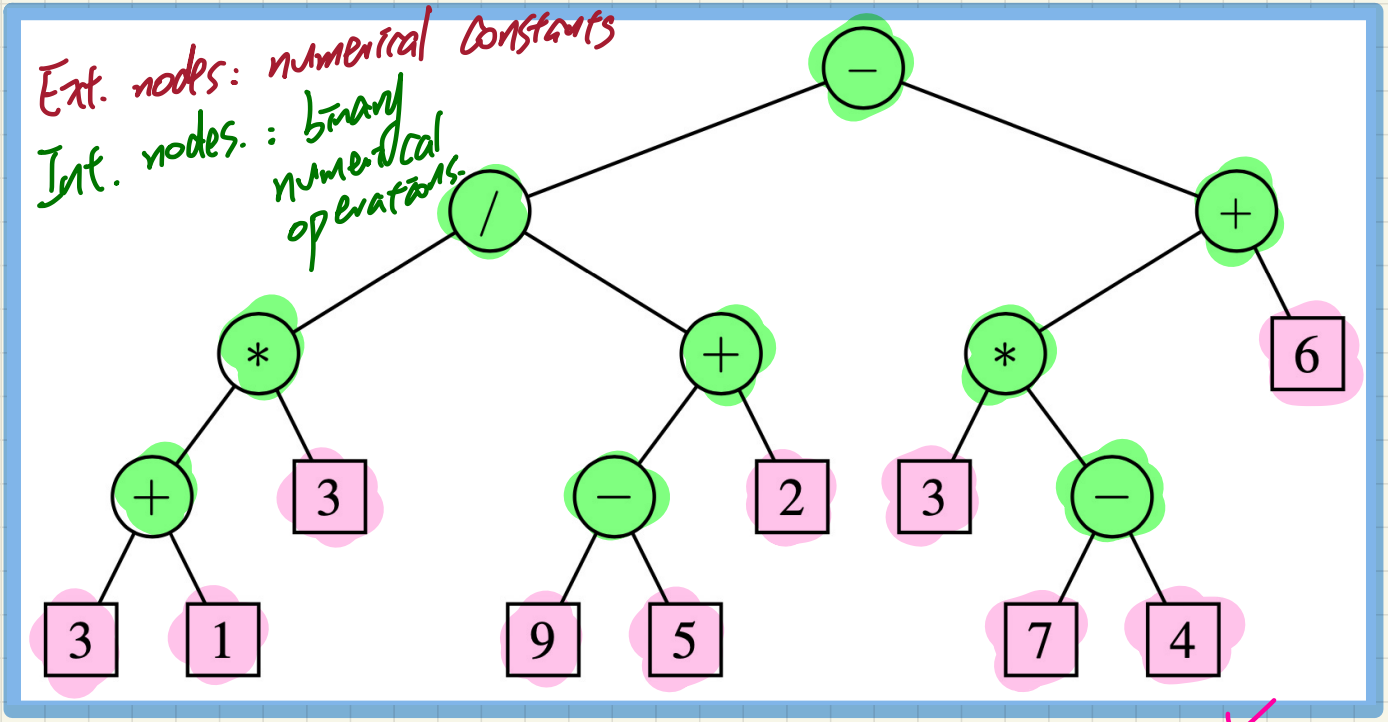

④ $n_E' \overset{(1)}{=} n_E + 1 \overset{I.H.}{=} (n_I + 1) + 1 \overset{(2)}{=} n_I' + 1$

(1) $n_E' = n_E \underbrace{-1}_{n} \underbrace{+2}_{x,y} = n_E + 1$

(2) $n_I' = n_I \underbrace{+1}_{n}$

show: $n_E' = n_I' + 1$

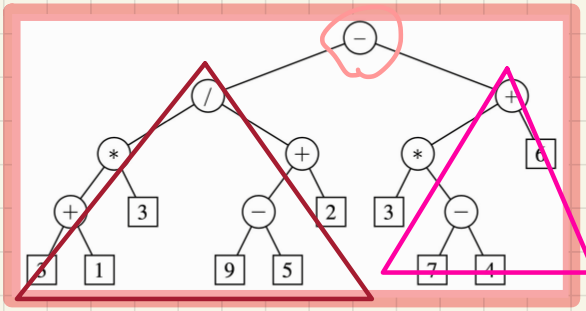
# Applications of Binary Trees: Infix Notation




BT's not proper.

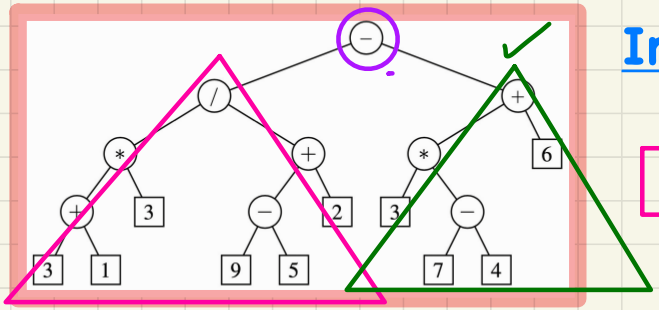

**Q.** Is the binary tree necessarily proper?

# Binary Tree Traversals



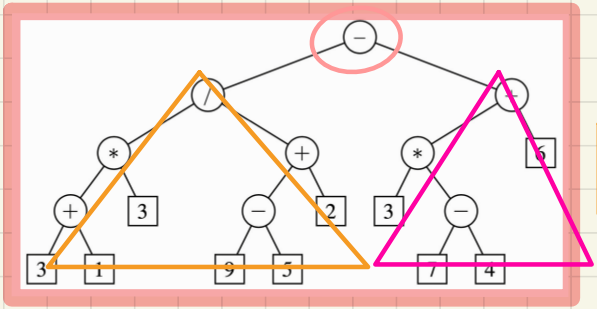

## Pre-Order Traversal

− / * + 3 1 3 + − 9 5 2 + * 3 − 7 4 6




## In-Order Traversal

3 + 1 * 3 / 9 − 5 + 2 − 3 * 7 − 4 + 6

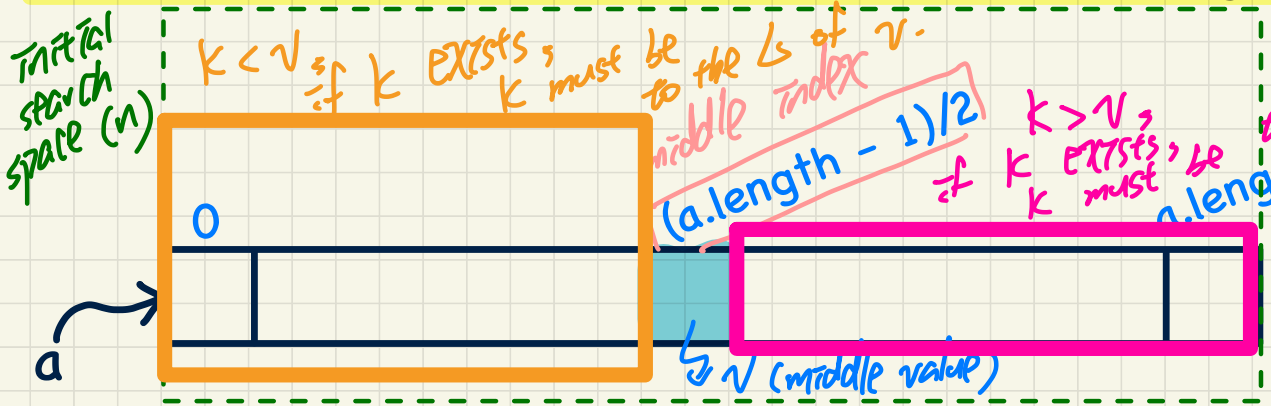



## Post-Order Traversal

3 1 + 3 * 9 5 − 2 + / 3 7 4 − * 6 + −

# Binary Search: Ideas


**Precondition**: Array **sorted** in non-descending order




Size of Search Space

n, n/2, n/4, ..., 1

# comparisons

log n (worst case)

**Search**: Does key **k** exist in array **a**?

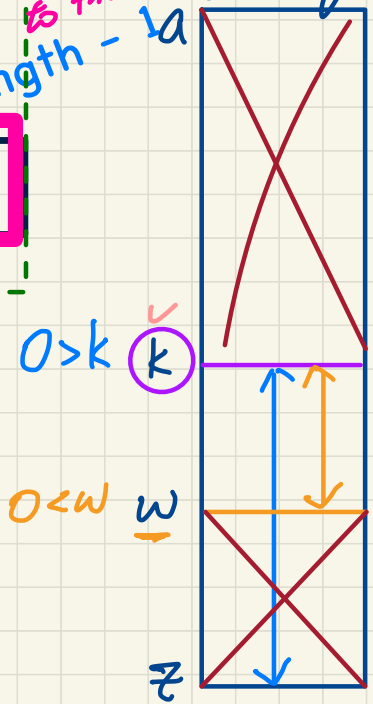
① Access the middle of search space

② Compare k against v:

k == v → found

k < v → Recur on the left

k > v → Recur on the right.



# Binary Search in Java

```
boolean binarySearch(int[] sorted, int key) {
  return binarySearchH(sorted, 0, sorted.length - 1, key);
}
boolean binarySearchH(int[] sorted, int from, int to, int key) {
  if (from > to) { /* base case 1: empty range */
    return false; }
  else if(from == to) { /* base case 2: range of one element */
    return sorted[from] == key; }
  else {
    int middle = (from + to) / 2;
    int middleValue = sorted[middle];
    if(key < middleValue) {
      return binarySearchH(sorted, from, middle - 1, key);
    }
    else if (key > middleValue) {
      return binarySearchH(sorted, middle + 1, to, key);
    }
    else { return true; }
  }
}
```

base

recursive cases

k == m.v.