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

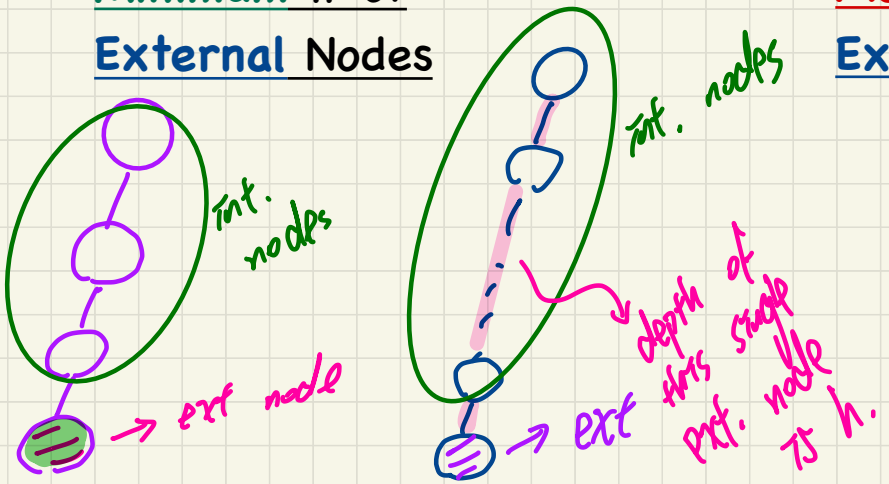
Given a *binary tree* with ***height*** h, the ***number of external nodes*** n_E is bounded as:

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

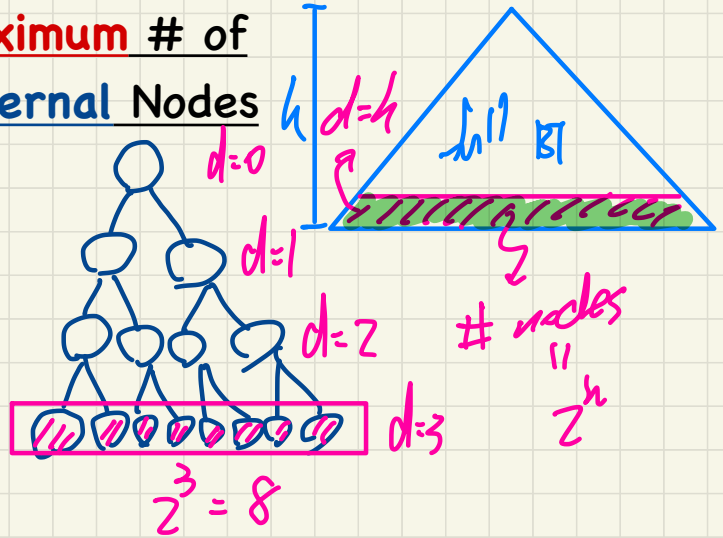
For example, say h = 3

max # edges from bottom node to root

Minimum # of External Nodes



Maximum # of External Nodes

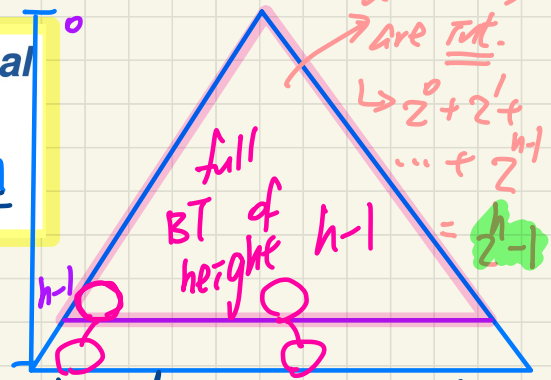


BT Properties: Bounding # of Internal Nodes

$S_k = 2^k - 1$

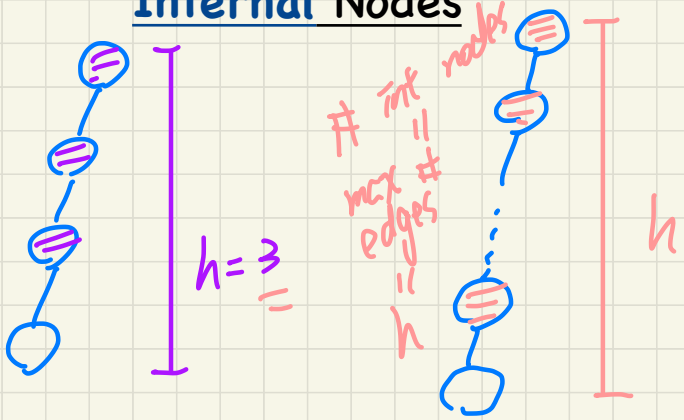
Given a *binary tree* with ***height*** h, the ***number of internal nodes*** n_I is bounded as:

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

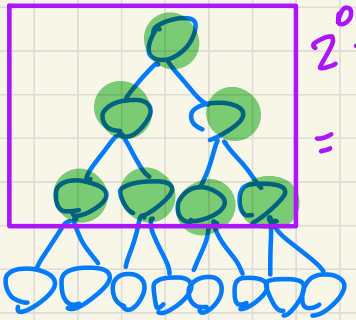


For example, say h = 3

Minimum # of Internal Nodes



Maximum # of Internal Nodes



$2^0 + 2^1 + 2^2 = 2^3 - 1 = 7$ lb. d

↳ as long as each node at level h-1 has at least one child; it's ok.

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: Singleton Proper BT REVIEW!

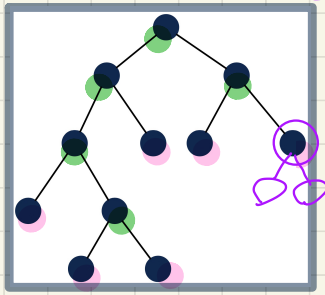
$n_E = 1$

$n_I = 0$

↳ property holds.

② Inductive Hypothesis (I.H.): for a proper BT of size >1: $n_E = n_I + 1$

③ Given a proper binary tree, extend it by choosing the right-most ext. node n and adding two child nodes to it x, y

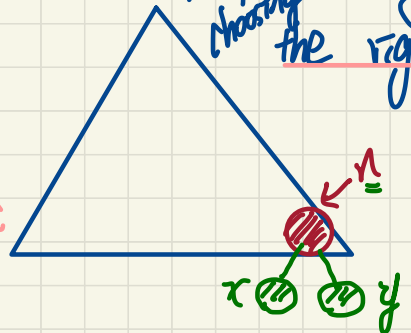


④ $n_E' \overset{(1)}{=} n_E + 1 \overset{IH}{=} (n_I + 1) + 1 \overset{(2)}{=} n_I' + 1$

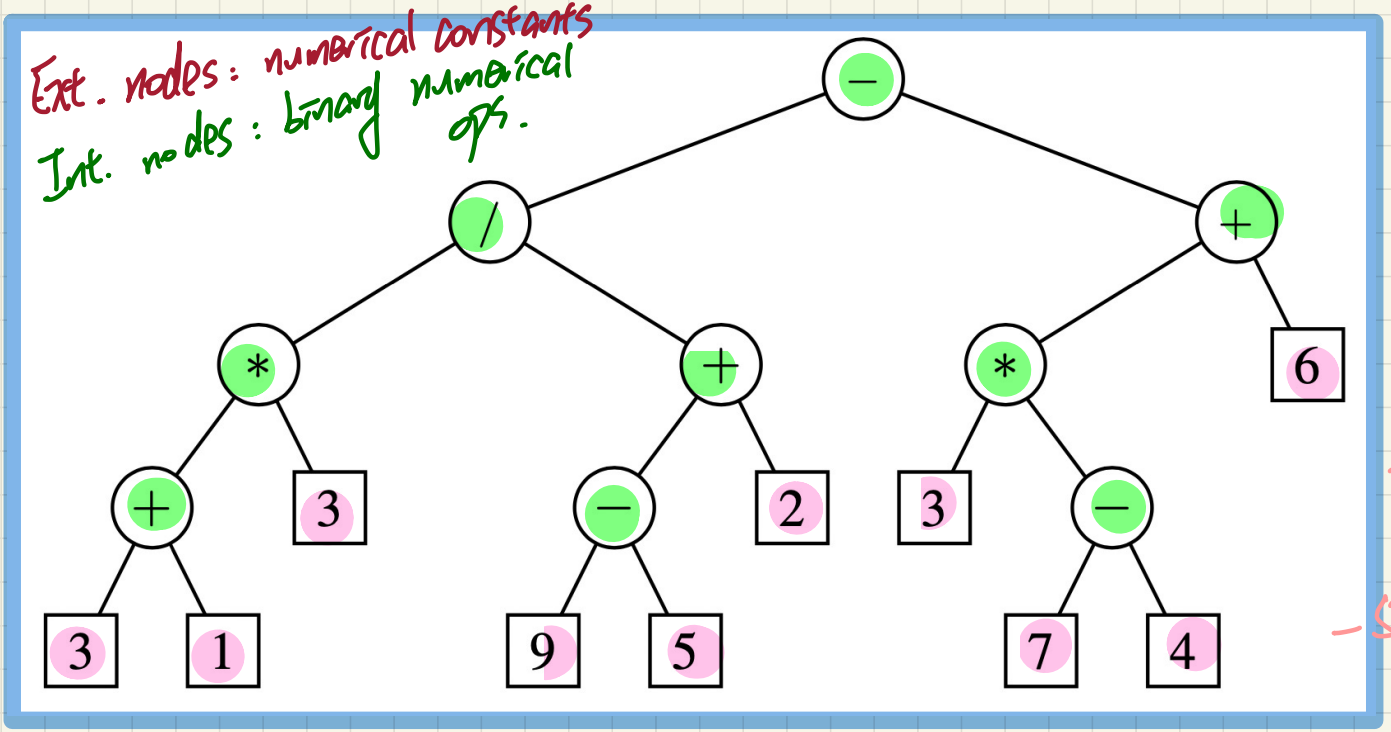
(1) $n_E' = n_E - 1 + 2 = n_E + 1$ (n: -1; x, y: $+2$)

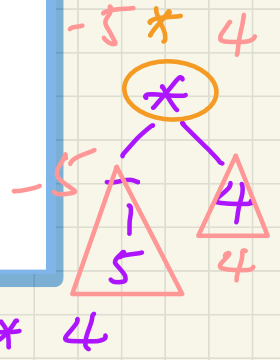
(2) $n_I' = n_I + 1$ (n: $+1$)

Show: $n_E' = n_I' + 1$



Applications of Binary Trees: Infix Notation



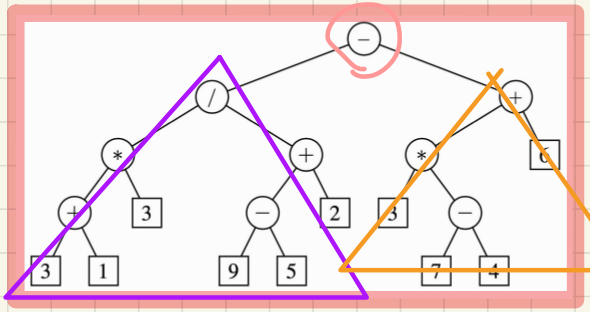


-5 * 4

Q. Is the binary tree necessarily proper?

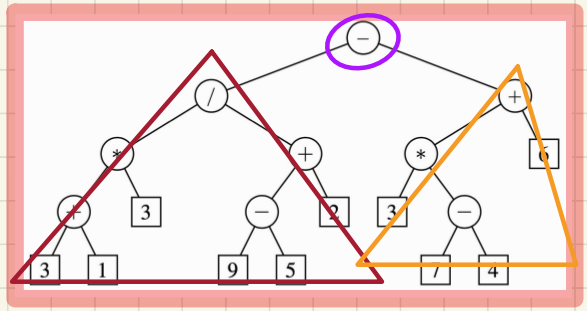
Binary Tree Traversals

TRAVEL THE WORLD



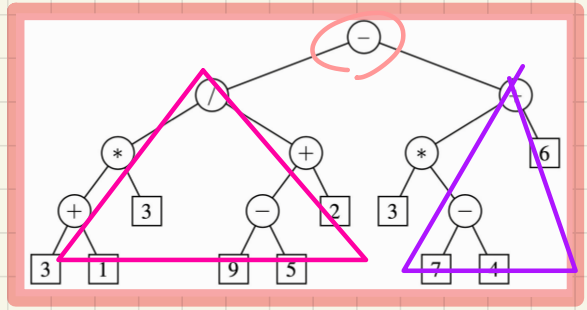
Pre-Order Traversal

− / * + 3 1 3 + − 9 5 2 + * 3 − 7 4 6



In-Order Traversal

3 + 1 * 3 / 9 − 5 + 2 − 3 * 7 − 4 + 6

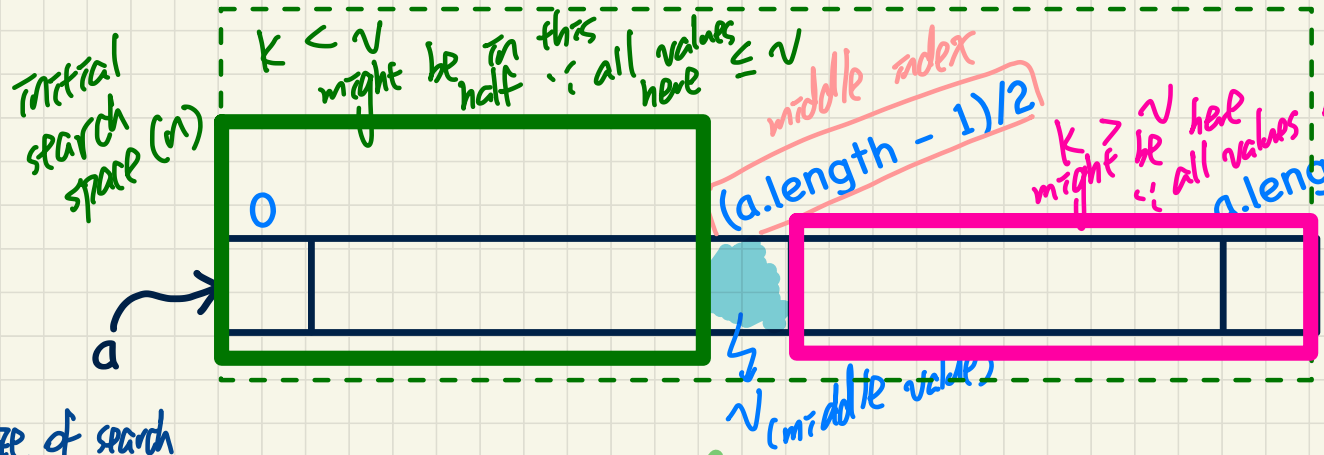


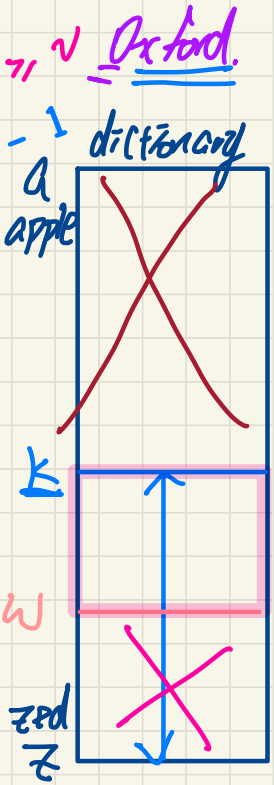
Post-Order Traversal

3 1 + 3 * 9 5 − 2 + / 3 7 4 − * 6 + −

Binary Search: Ideas

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

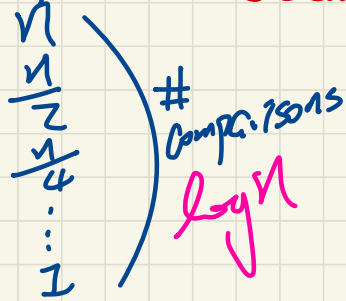




Search: Does key **k** exist in array **a**?

① keep accessing the middle of the search space (halved each time)

② Recursion: (A) left if k < v
(B) right if k > v



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; }
  }
}
```

recursive cases

k == middleValue.