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**

Minimum # of External Nodes

Maximum # of External Nodes

BT Properties: Bounding # of Internal Nodes

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

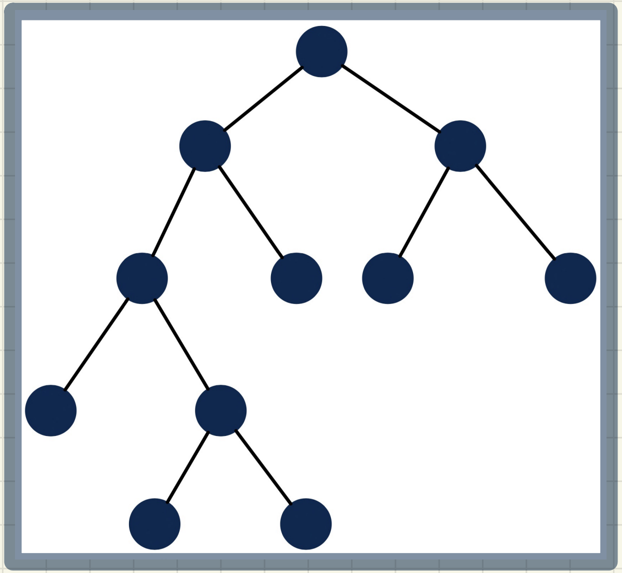
BT Properties: Relating #s of Ext. and Int. Nodes

Given a *binary tree* that is:

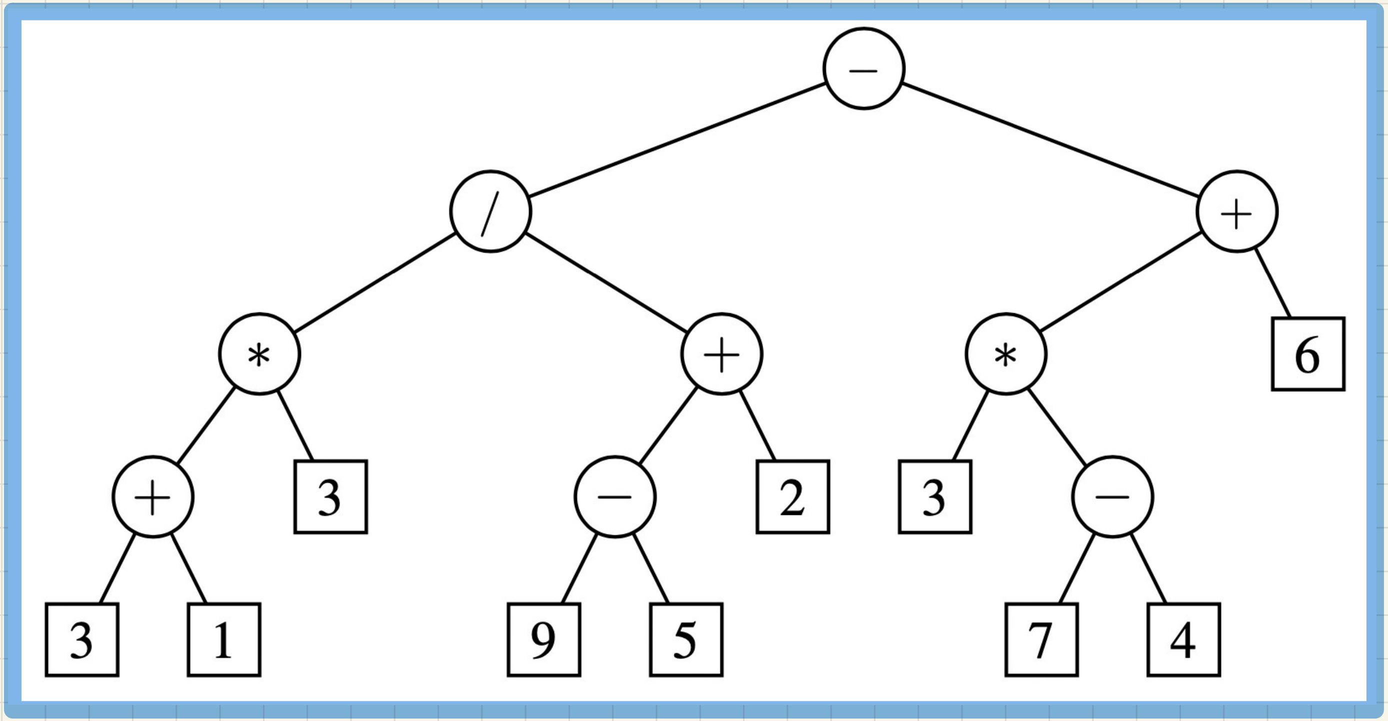
- ***nonempty*** and ***proper***
- with n_I ***internal nodes*** and n_E ***external nodes***

We can then expect that: $\boxed{n_E = n_I + 1}$

Induction on Size of Proper BT



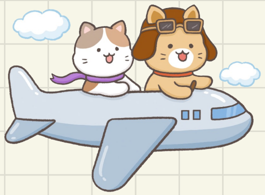
Applications of Binary Trees: Infix Notation

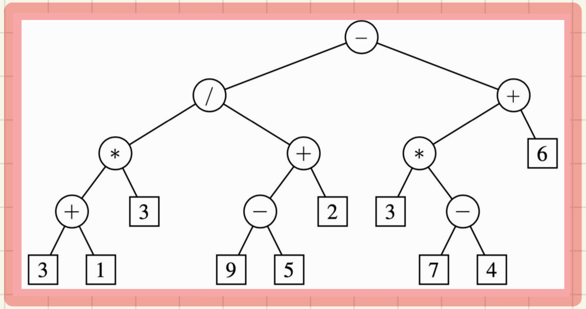


Q. Is the binary tree necessarily **proper**?

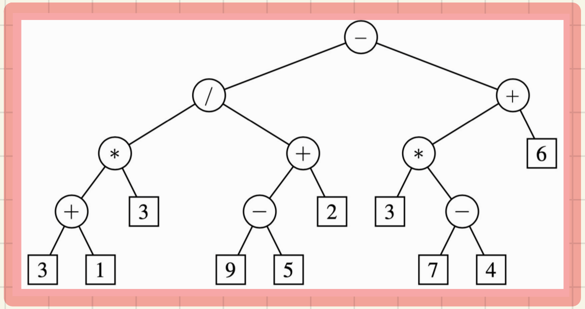
Binary Tree Traversals

TRAVEL THE WORLD

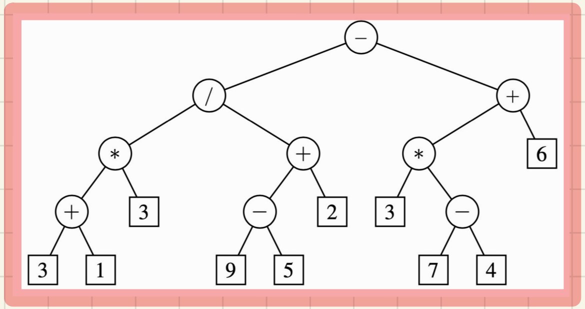




Pre-Order Traversal



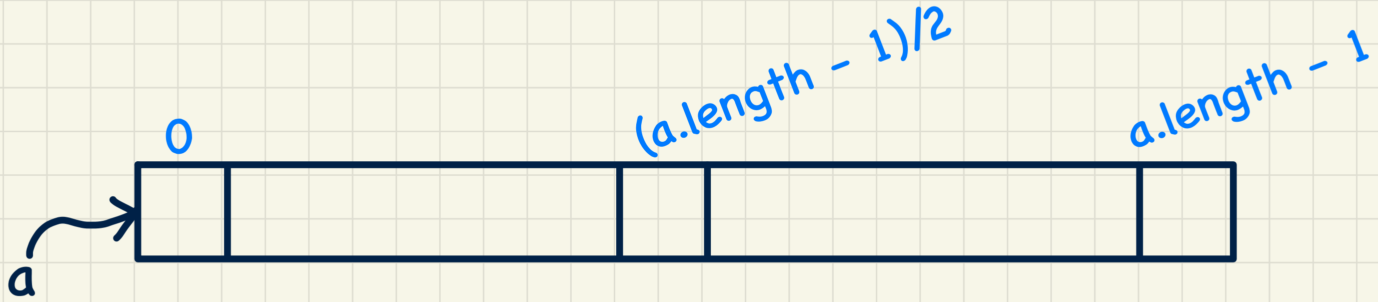
In-Order Traversal



Post-Order Traversal

Binary Search: Ideas

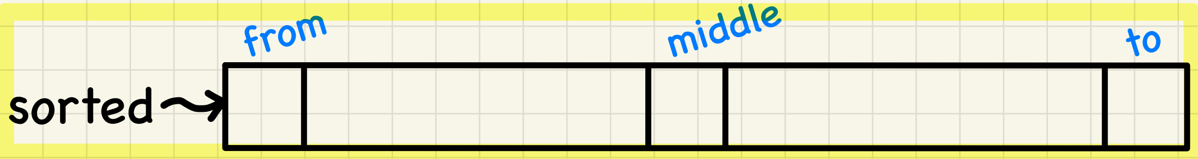
Precondition: Array sorted in non-descending order



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

Binary Search in Java

```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; }
  }
}
```



from
middle
to
sorted