Announcements

1. Exam 1 Window ends today!
    a. If you are sick or an emergency comes up, email cs225admin
2. MP Lists is released!

Exam OH Changes

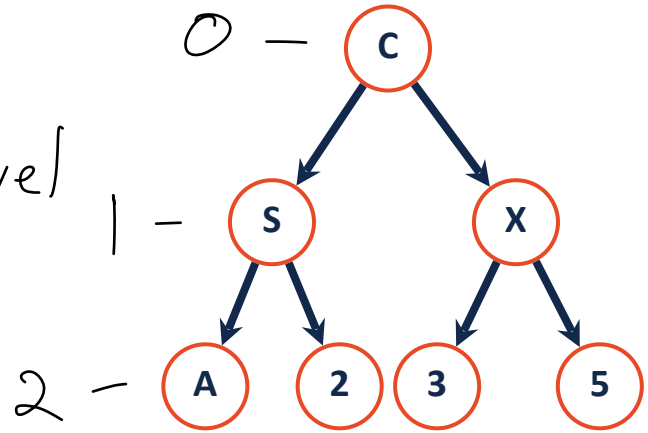
Warm Up Question: How many leaves are in a perfect tree of height, h?

Join Code: **225**

Perfect Trees

Given a perfect tree with height h:

1. $\emptyset$, or T_L & T_R were P_{h-1}
2. Full Tree w/ all leaves at same level
3. Total leaves is 2^h
4. Total # of nodes



Level	Nodes
0	1
1	2
2	4

How many leaf nodes are there?

height: h, 2^h

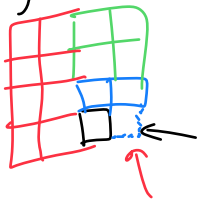
Perfect Trees

Given a perfect tree with height h:

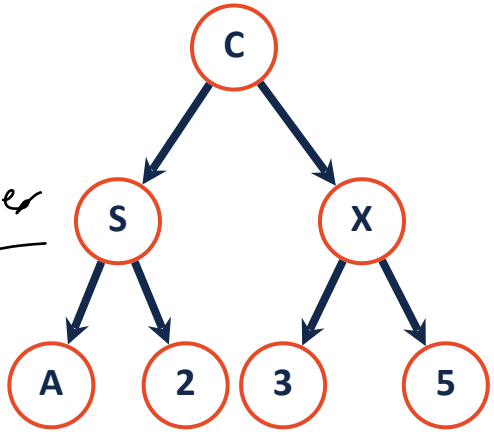
How many nodes in total?

$2^*(\text{nodes at last level}) - 1$

$2^*(2^h) - 1 = 2^{h+1} - 1$



# of Nodes	Level	Total Number
1	0	1
2	1	3
4	2	7
$8 = 2^h$	3	



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

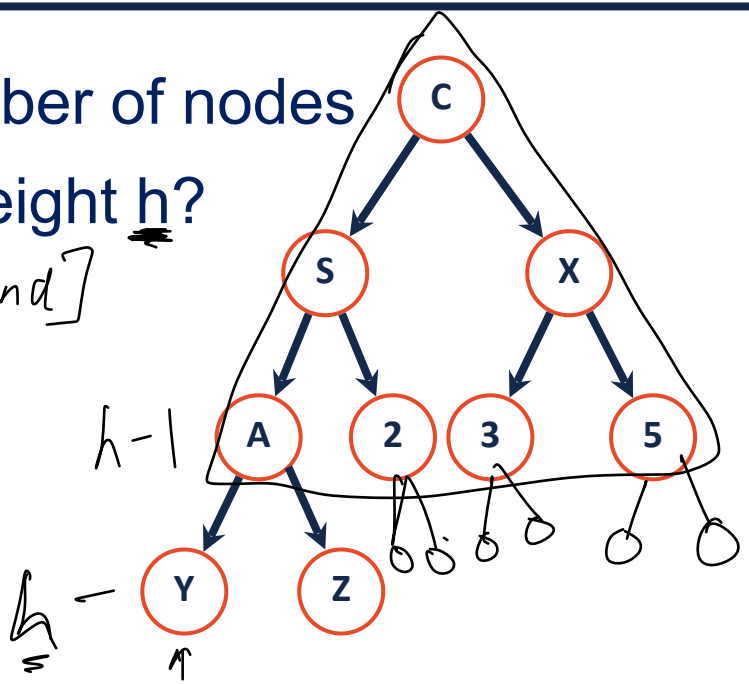
Complete Trees

What is the range of number of nodes in a complete tree with height h?

[lower bound, upper bound]

$2^{h-1+1} - 1 + 1 = \boxed{2^h}, \boxed{2^{h+1} - 1}$

# of nodes in perfect tree



Tree Traversals

Learning Objectives

1. Understand the Tree ADT
2. Prove how much overhead a tree has
3. Understand what a tree traversal is
4. Implement Tree Traversals

Tree ADT

insert, inserts an element to the tree.

remove, removes an element from the tree.

access, access elements from the tree.

↳ find

Make Tree

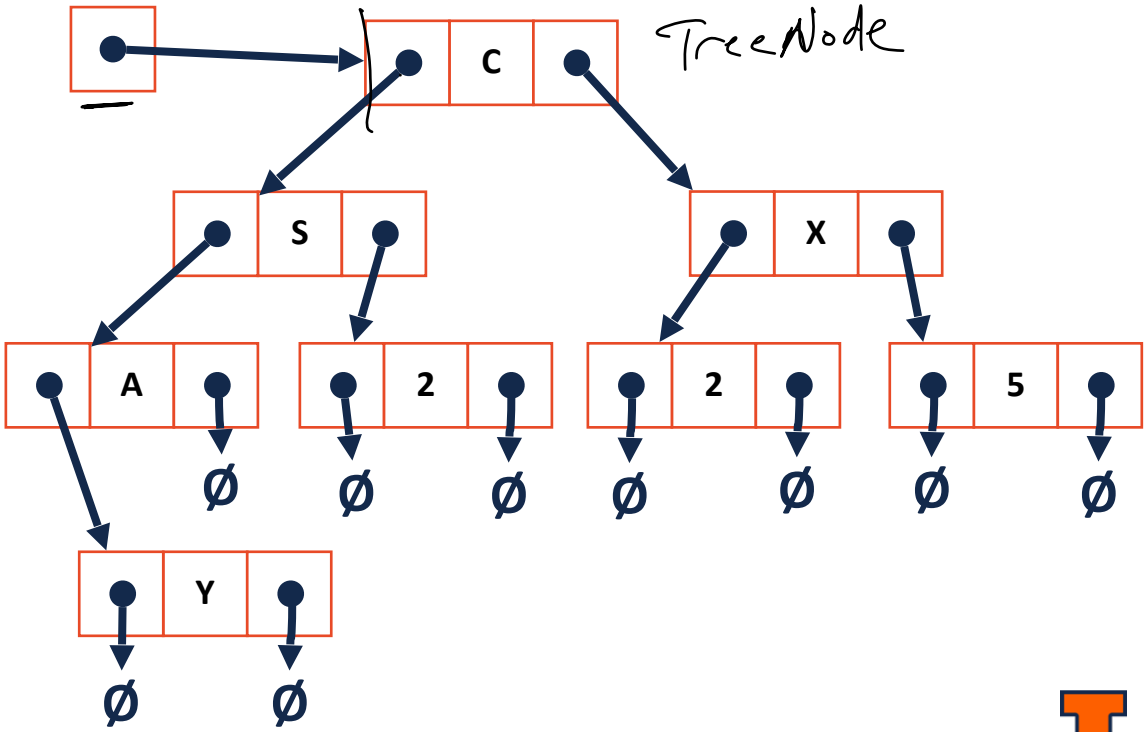
Empty

BinaryTree.h

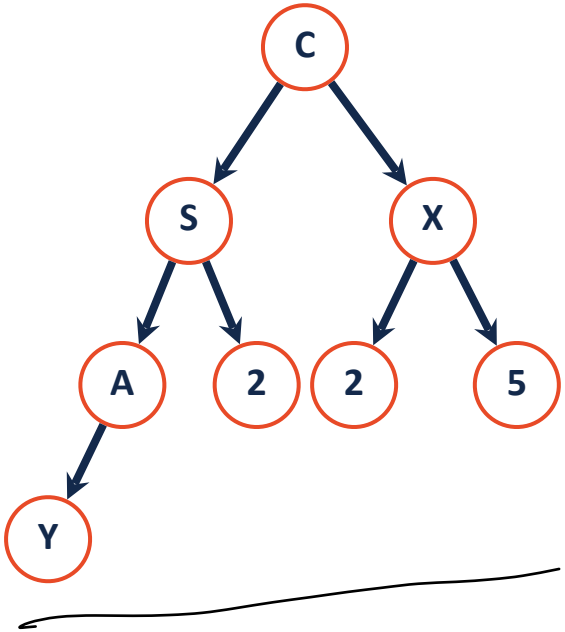
```
1  #pragma once
2
3  template <class T>
4  class BinaryTree {
5    public:
6      /* ... */
7
8    private:
9      struct TreeNode {
10       TreeNode* left;
11       T data;
12       TreeNode* right;
13     };
14     TreeNode* root_;
15
16 };
17
```

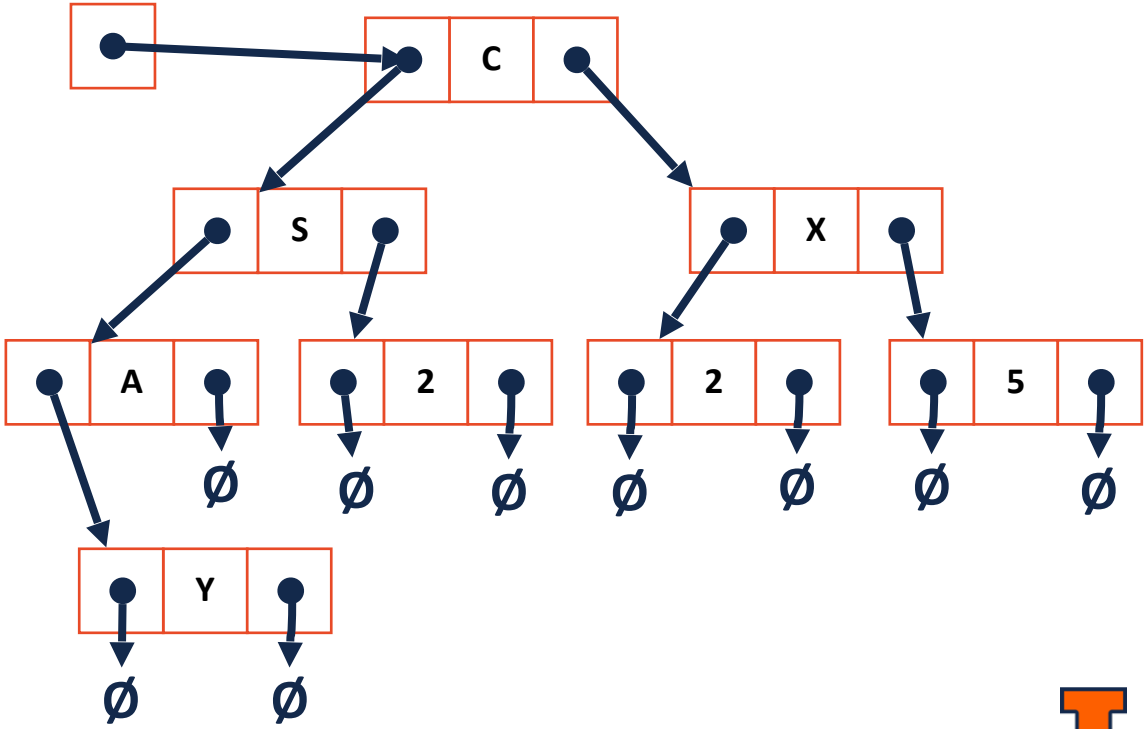
In Memory Representation

root_

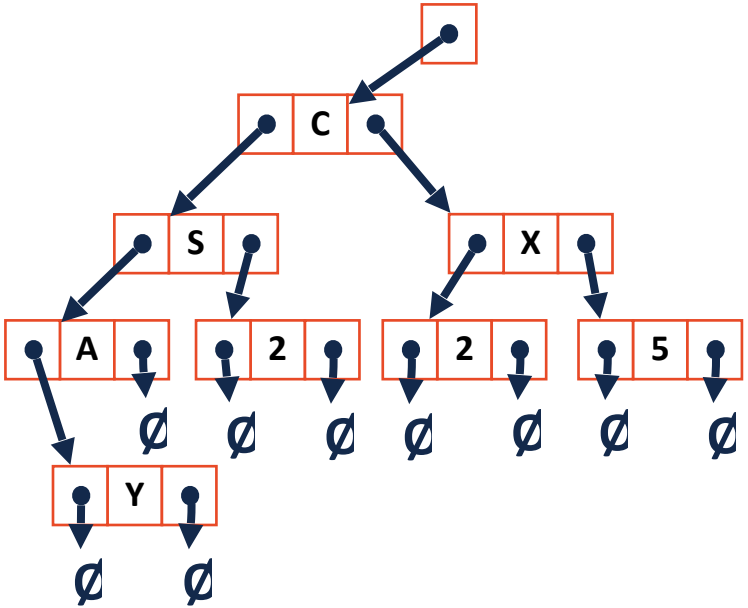


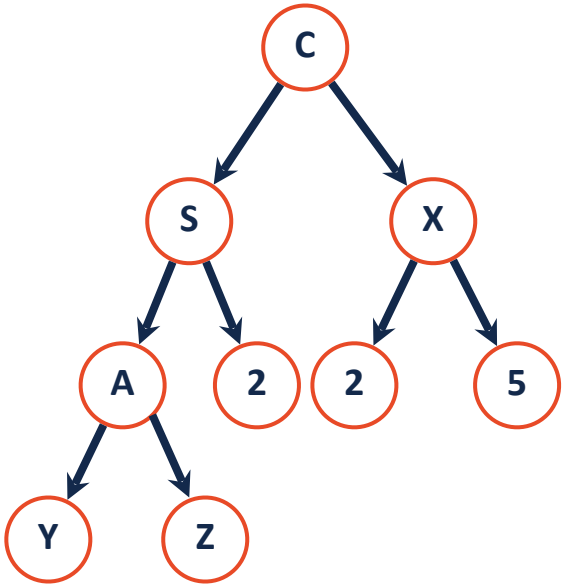
In Memory Representation





Wasted Pointers in a binary tree?





How many Null Pointers?

Base Cases:

Null(n):= the number of null pointers with n nodes in the binary tree

n = 0: Null (0) = 1

n = 1: Null (1) = 2

h = 0, n = 1

n = 2: Null (2) = 3

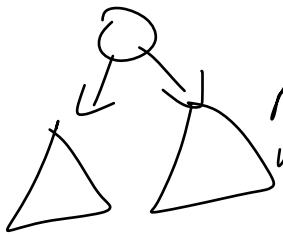
How many Null Pointers?

Theorem: If there are **n** data items in our representation of a binary tree, then there are n+1 NULL pointers.

Induction:

Given: Binary Tree w/ n nodes & n+1 NuLL pointers

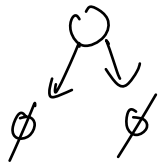
Prove: Binary Tree w/ n+1 nodes has n+2 Null pointers



n nodes
n+1 null

Consider any leaf nodes

Before

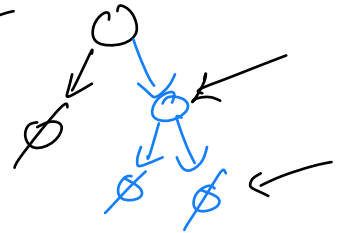


n+1

n+1-1+2

= n+2

After



Induction Hypothesis

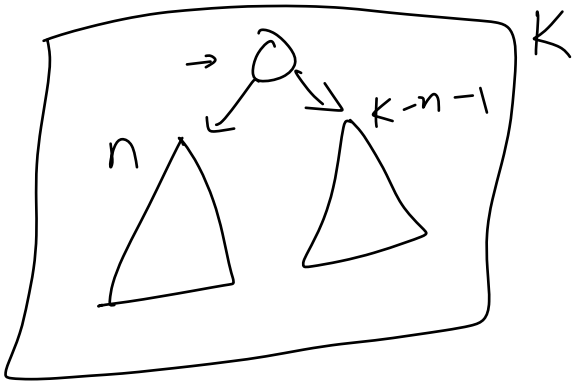
In class activity: Another Inductive Proof

2:00

Inductive Hypothesis: Null(n) = n+1 for all n, 0<= n < k

Goal: prove null(k) = k + 1

Consider an arbitrary tree **T** containing **k** data elements:



Null(k) = null(n) + null(k-n-1)

left right

Null(k) = n+1 + k-n-1+1

Null(k) = k+1

Traversals

::begin	::end
ArrayIterator(0)	ArrayIterator(size_)
ListIterator(head_)	ListIterator(nullptr)

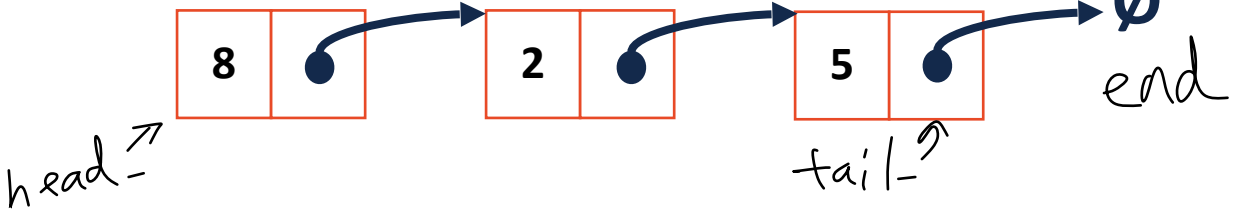
Array Lists

- Index

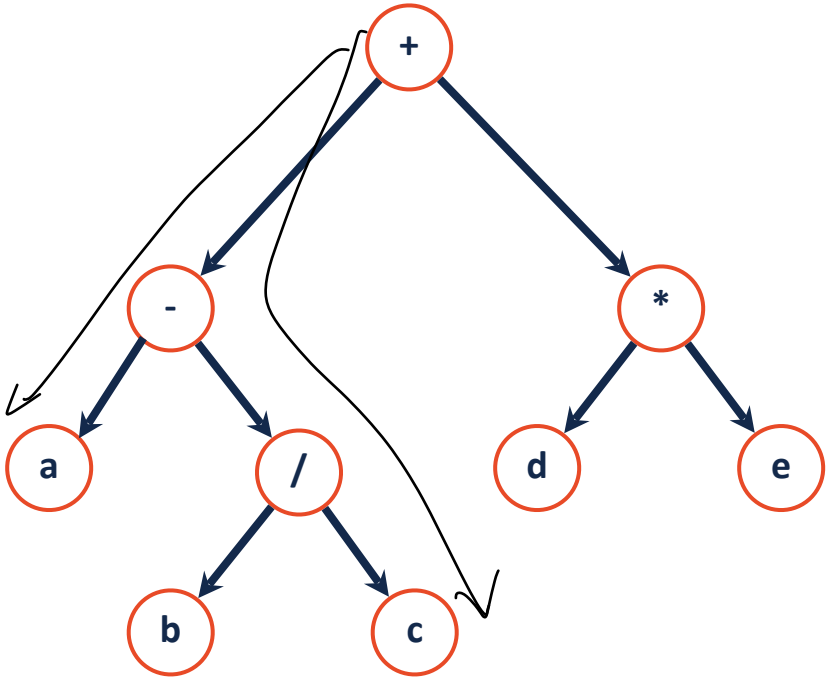
0 size_

What order do I traverse my data structure?

Linked Lists

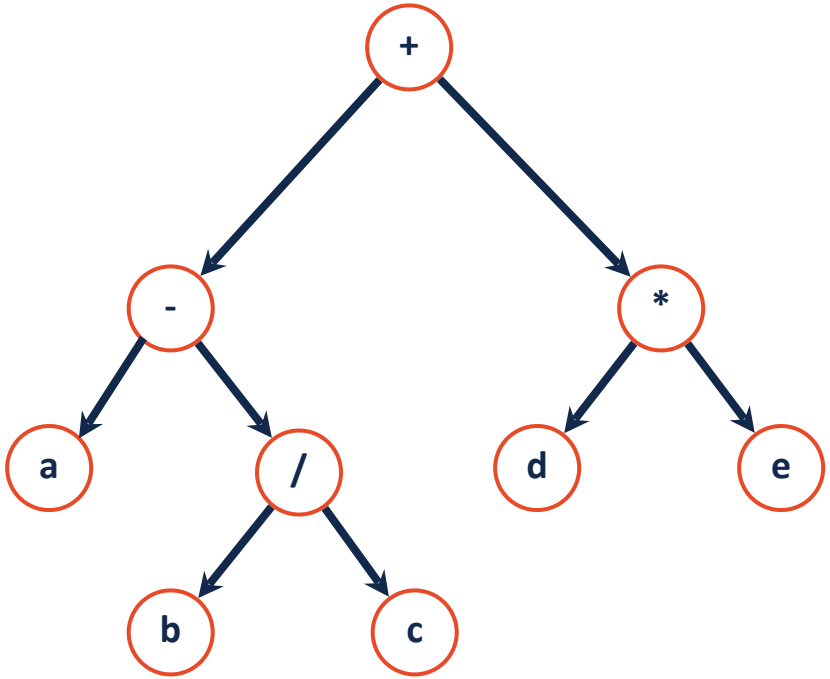


Accessing all nodes



Types of Traversals

- InOrder
- PreOrder
- PostOrder
- LevelOrder



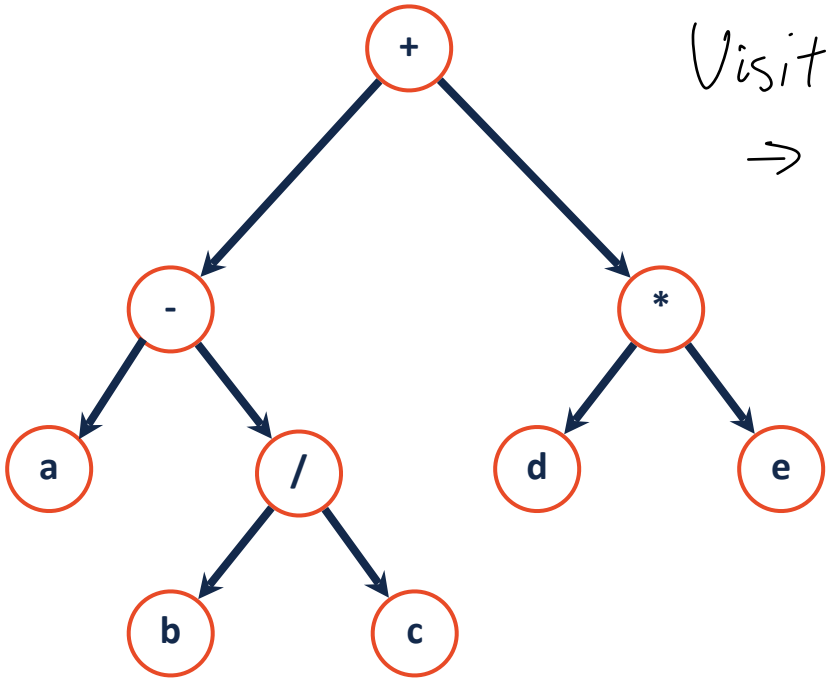
Inorder Traversal

Inorder:

Left Subtree

Root

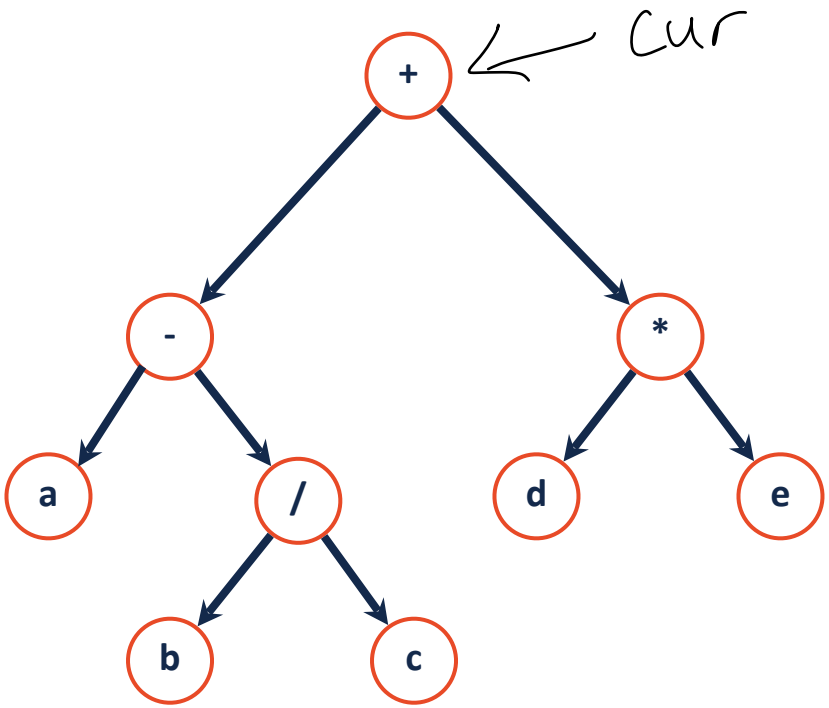
Right Subtree



Visit a node

⇒ Print it out

Inorder Traversal

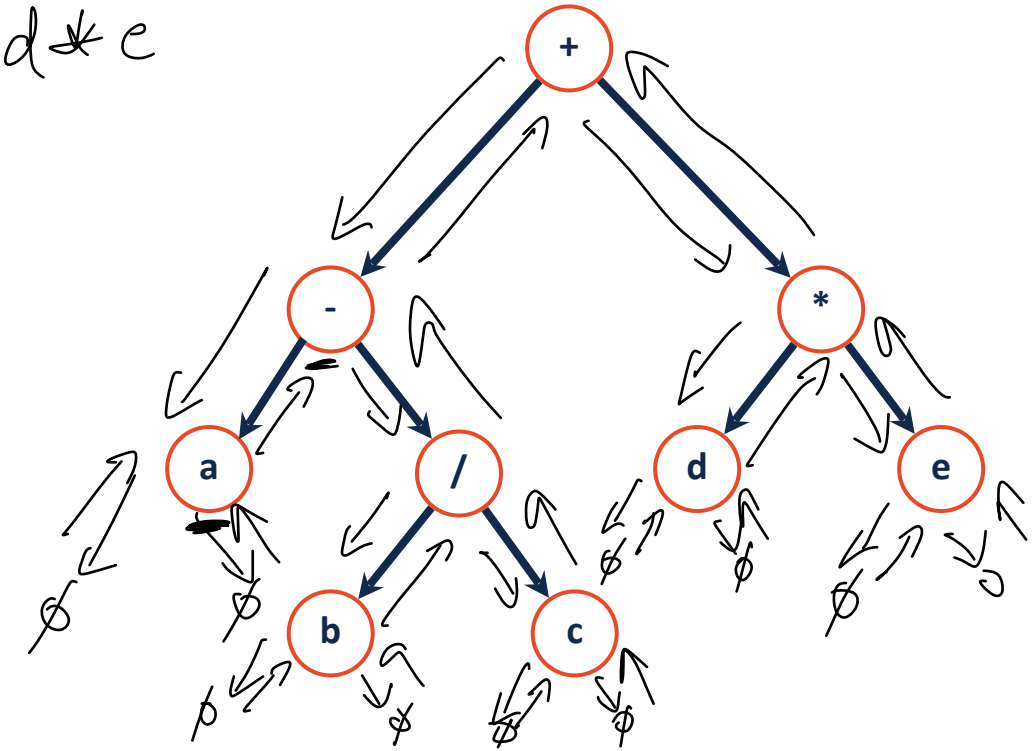



```
49  template<class T>
50  void BinaryTree<T>::InOrder(TreeNode * cur) {
51     if (cur != NULL) {
52        InOrder(cur->left);
53        std::cout << cur->data << std::endl;
54        InOrder(cur->right);
55     }
56  }
57
```

Visit

Inorder Traversal: Which node is visited first?

a - b / c + d * e



Left
Root
Right

Join Code: **225**

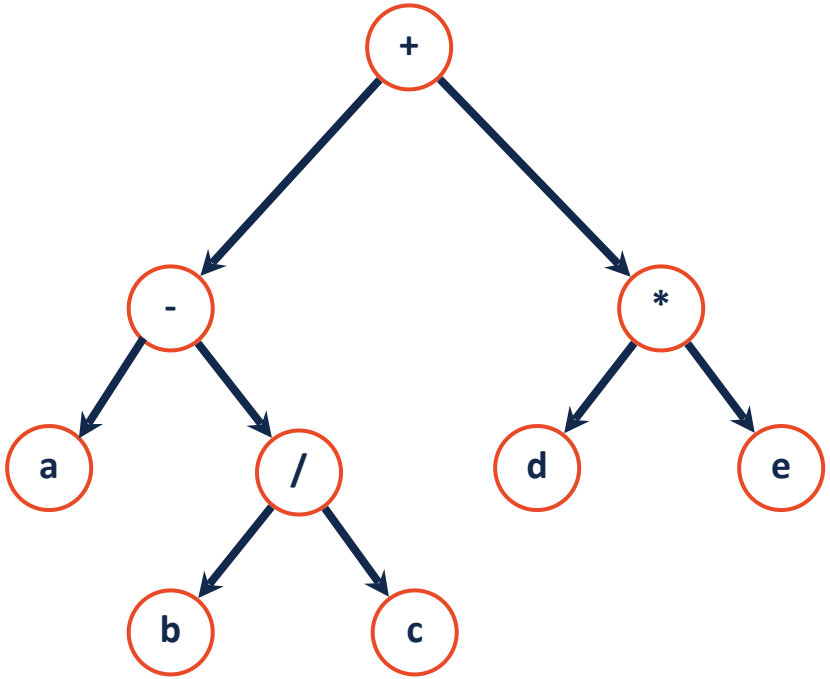
Types of Traversals

InOrder

PreOrder

PostOrder

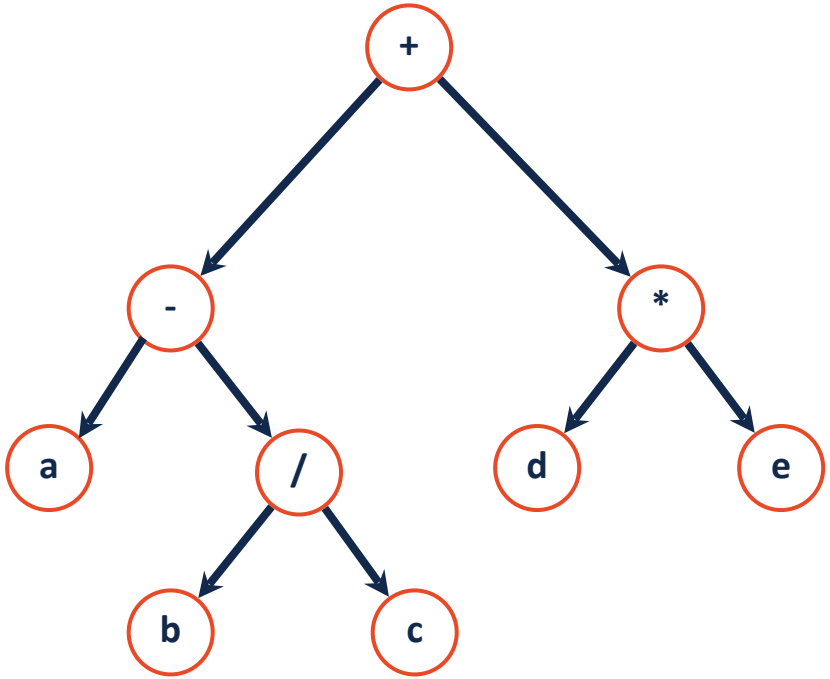
LevelOrder



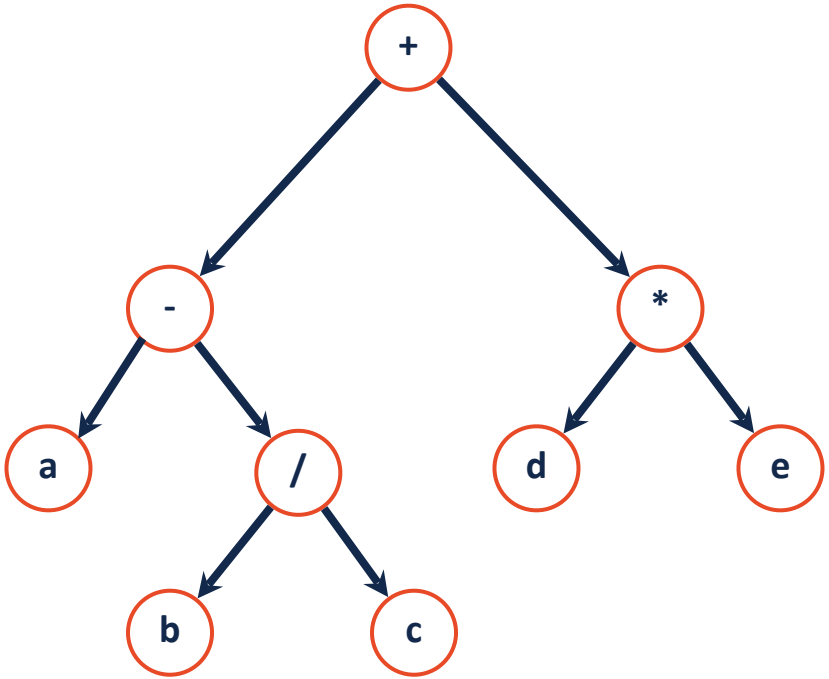
Preorder Traversal

Pre Order:

Root
Left
Right



Preorder




```
49  template<class T>
50  void BinaryTree<T>::PreOrder(TreeNode * cur) {
51     if (cur != NULL) {
52        std::cout << cur->data << std::endl;
53        InOrder(cur->left);
54        InOrder(cur->right);
55     }
56  }
57
```

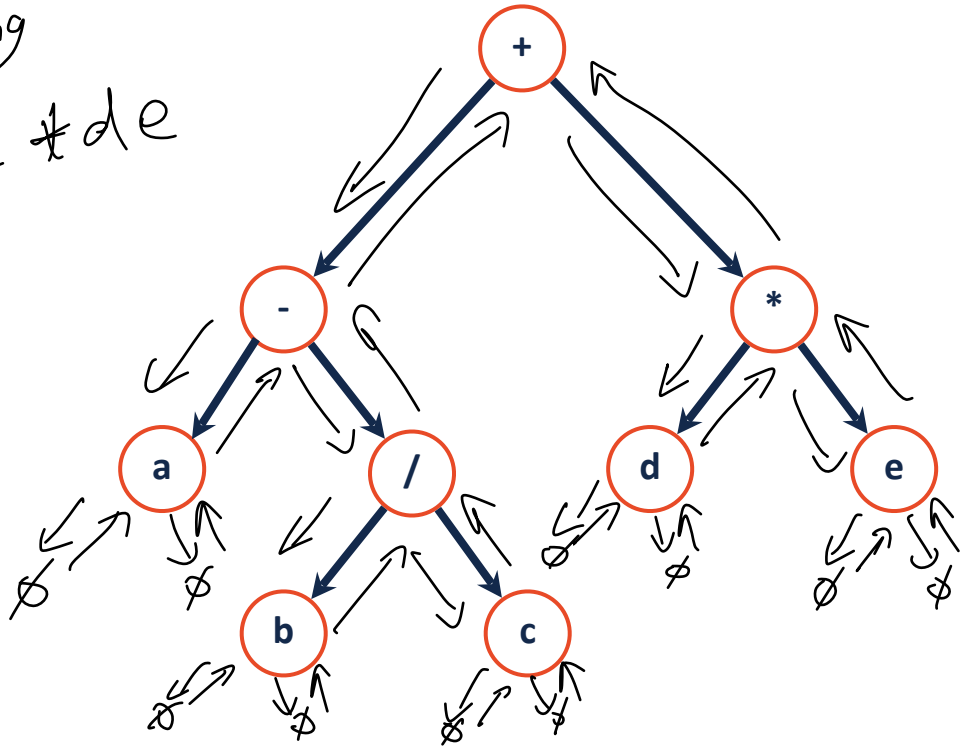
Pre

Pre

Preorder Traversal: Which node is visited first

Order of visiting

+ - a / b c * d e



Join Code: **225**

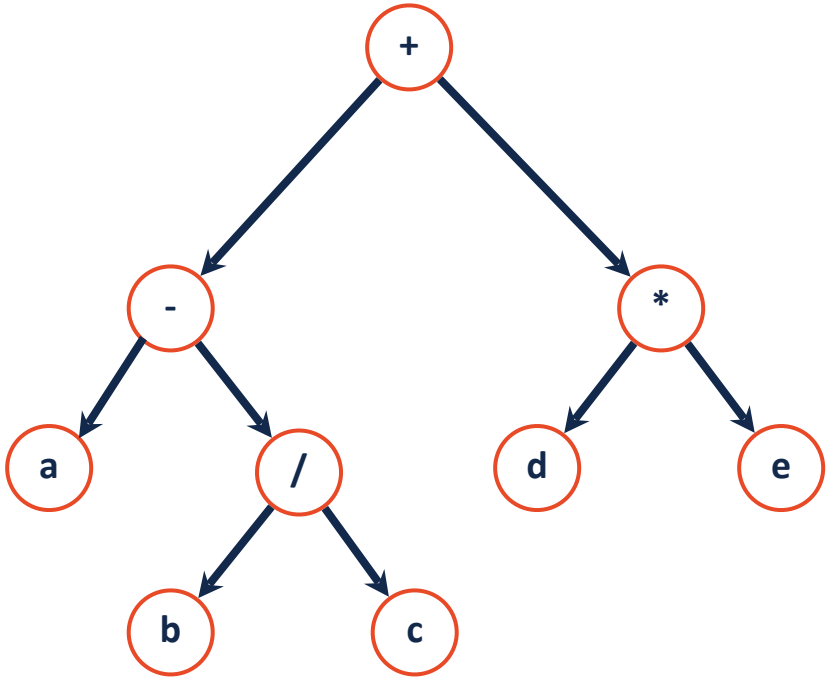
Postorder Traversal

Postorder

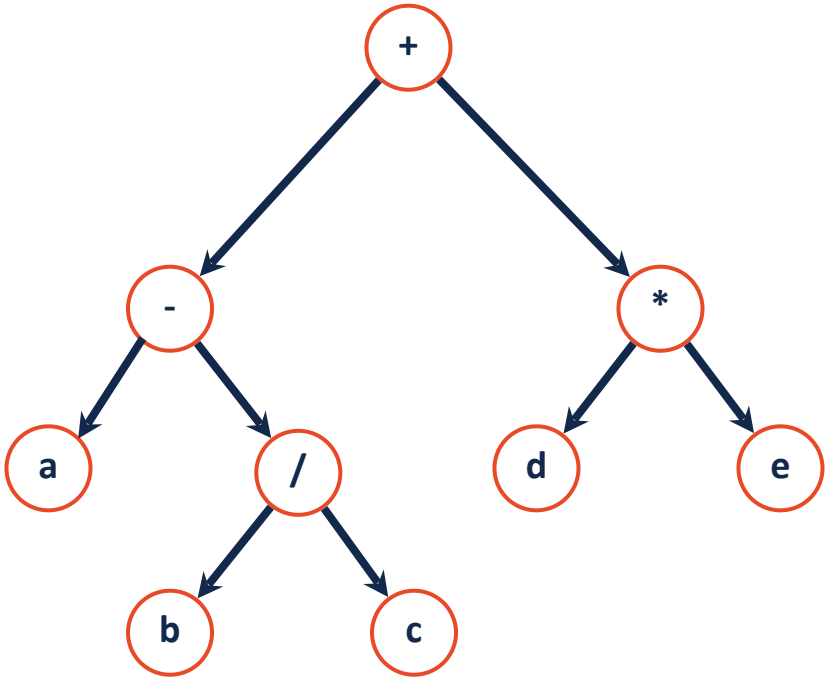
Left Subtree

Right Subtree

Root



Postorder

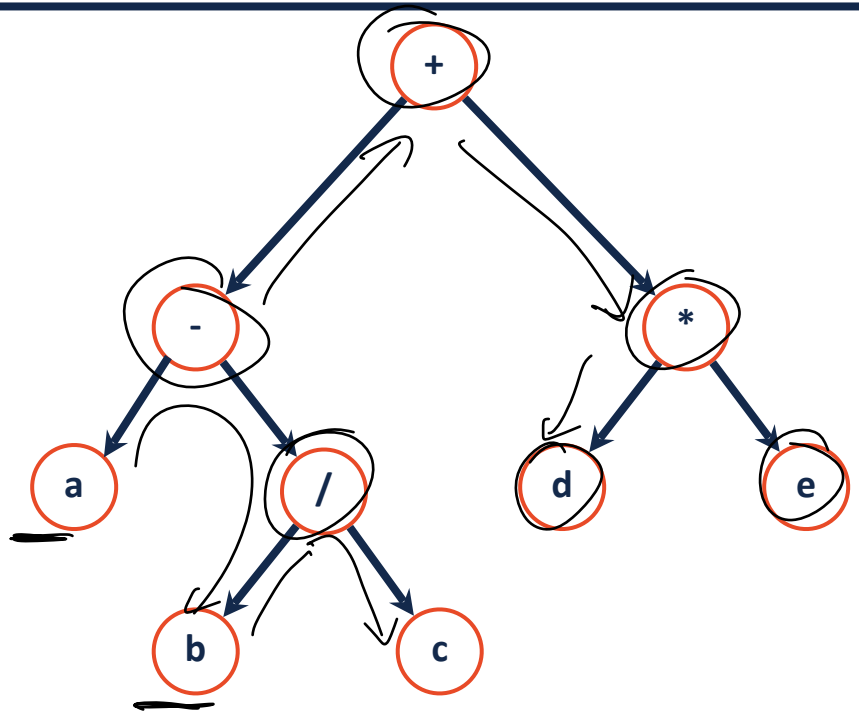



```
49  template<class T>
50  void BinaryTree<T>::PostOrder(TreeNode * cur) {
51     if (cur != NULL) {
52        InOrder(cur->left);
53        InOrder(cur->right);
54        std::cout << cur->data << std::endl;
     }
  }
55
56
57
```

Postorder Traversal

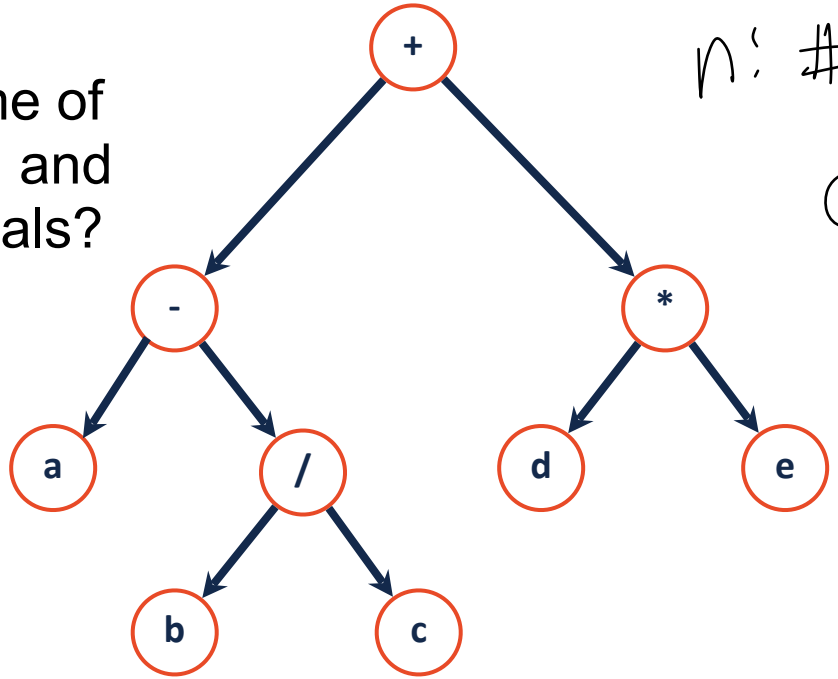
Order

a b c / - d e * +



Traversal Runtimes

What is the runtime of pre-order, inorder, and post-order traversals?



n: # of nodes

O(n)

Combining information

Preorder Output: + - a / b c * d e

Postorder Output: a b c / - d e * +