Announcements

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

Join Code: **225**

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.

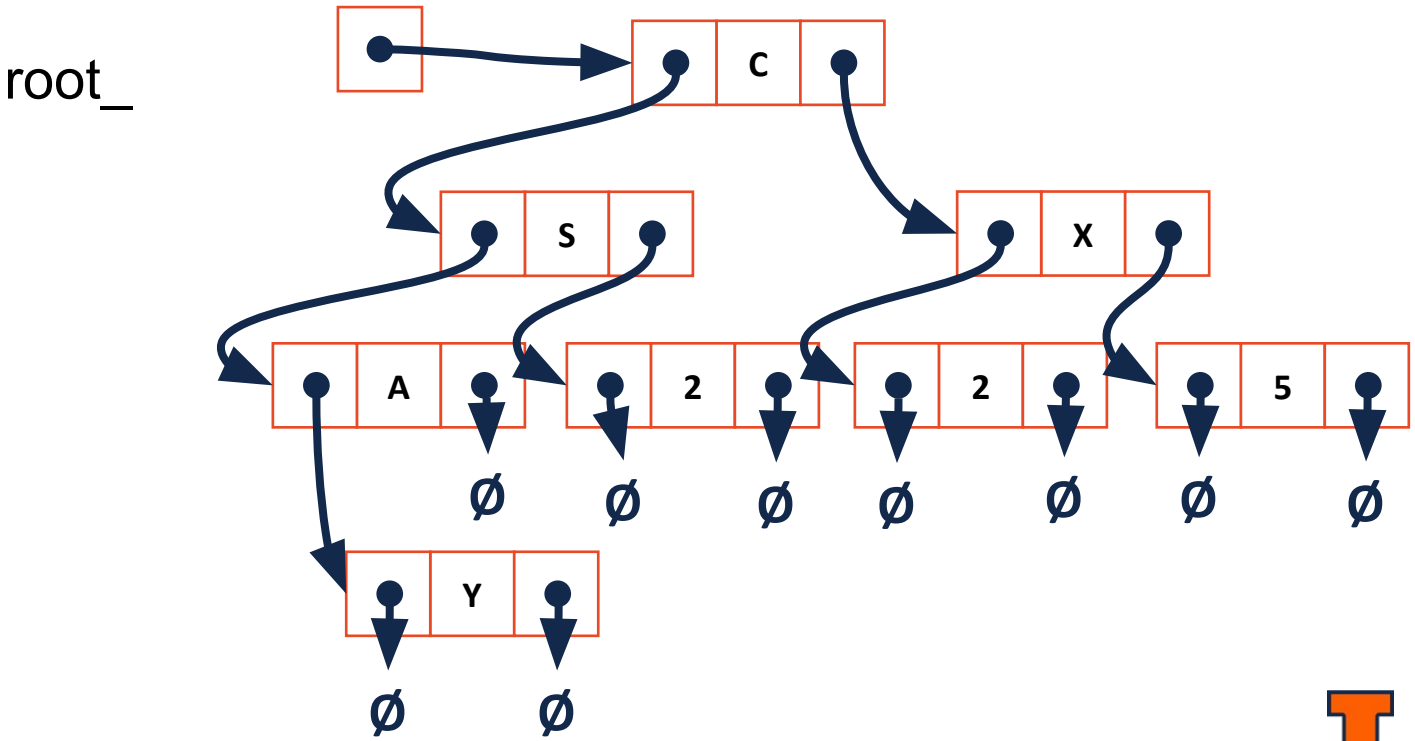
Make Tree

Empty

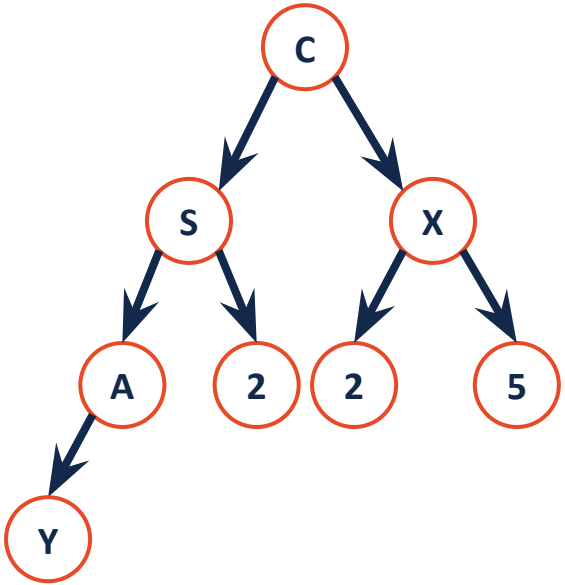
BinaryTree.h

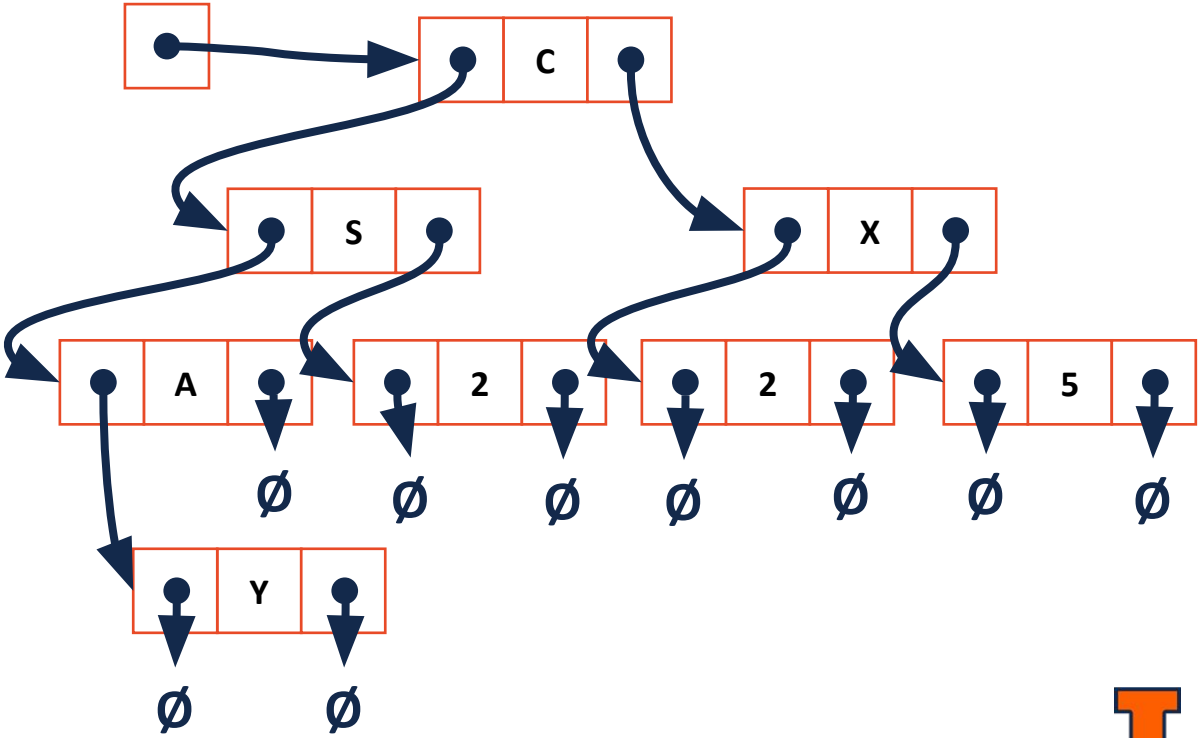
```
1  #pragma once
2
3  template <class T>
4  class BinaryTree {
5    public:
6      /* ... */
7
8    private:
9
10
11
12
13
14
15
16 };
17
```

In Memory Representation

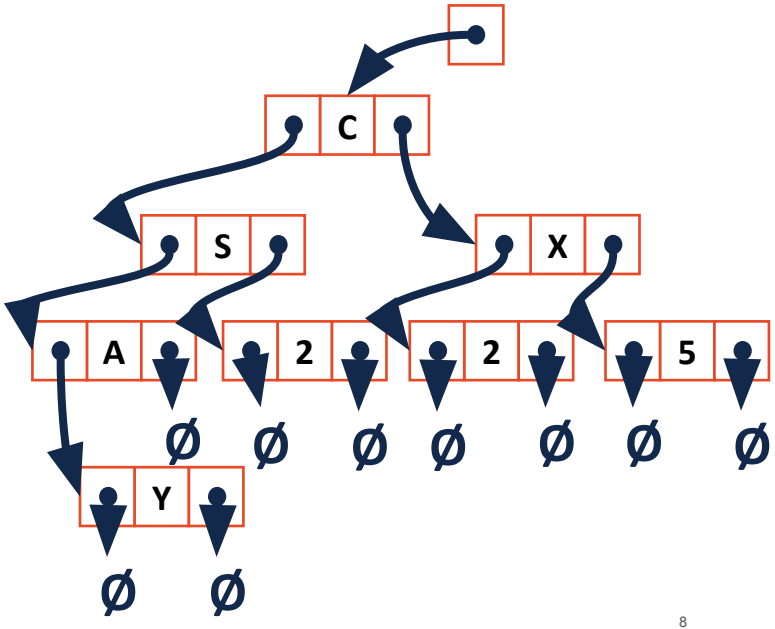


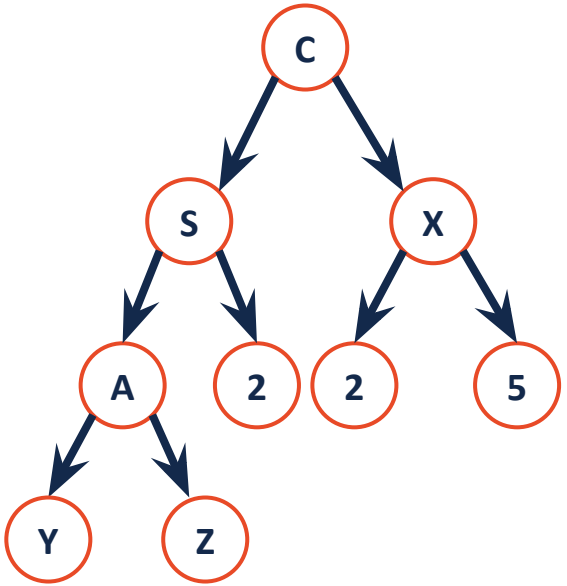
In Memory Representation





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:

n = 1:

n = 2:

How many Null Pointers?

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

Induction Hypothesis

In class activity: Another Inductive Proof

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

Goal: prove null(k) = k + 1

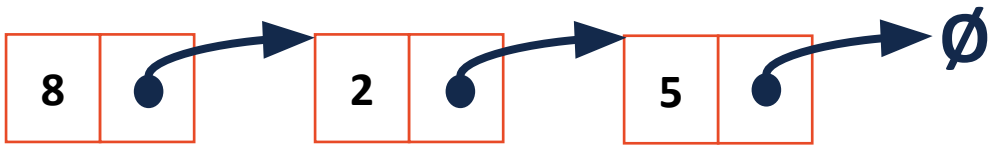
Consider an arbitrary tree **T** containing **k** data elements:

Traversals

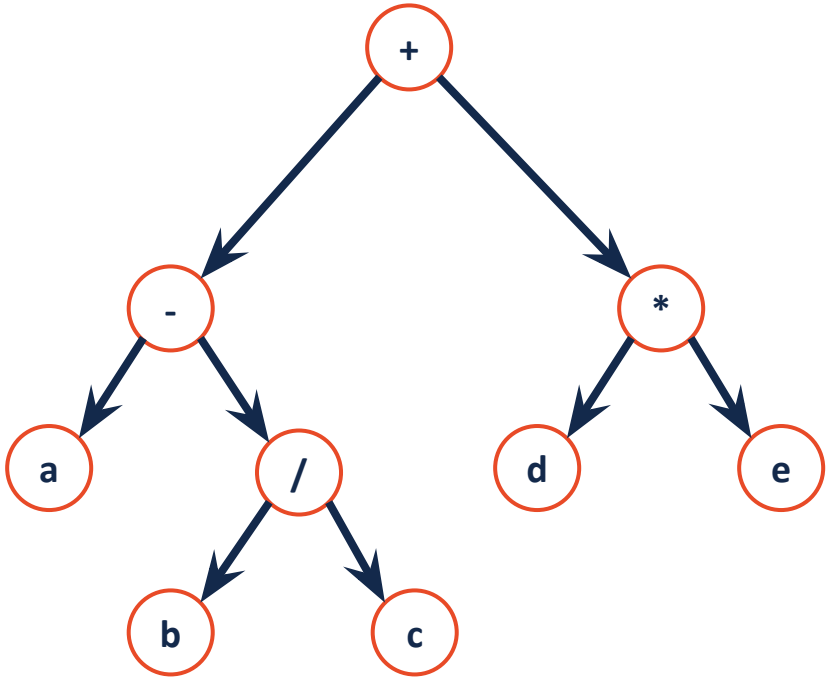
::begin	::end
ArrayIterator(0)	ArrayIterator(size_)
ListIterator(head_)	ListIterator(nullptr)

Array Lists

Linked Lists



Accessing all nodes



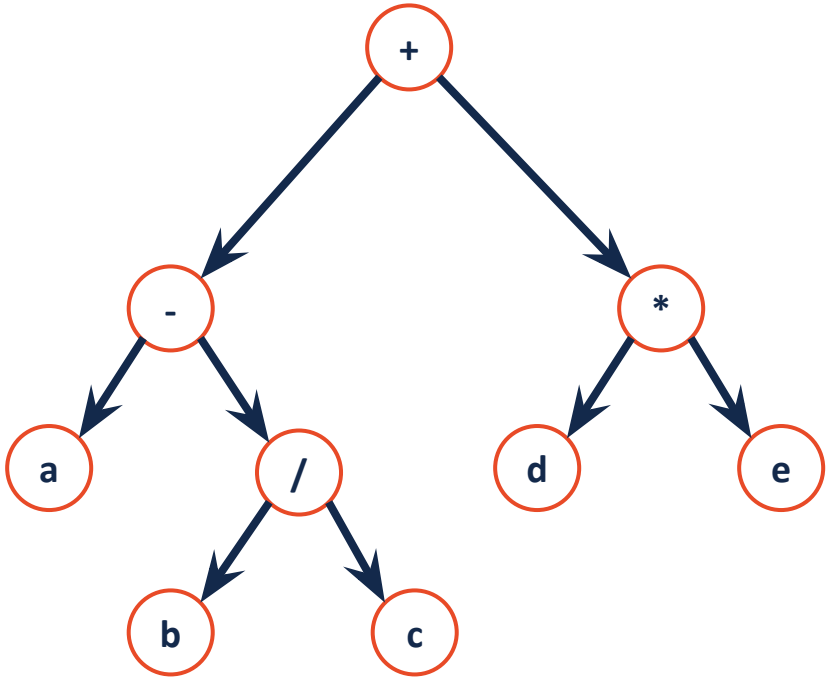
Types of Traversals

InOrder

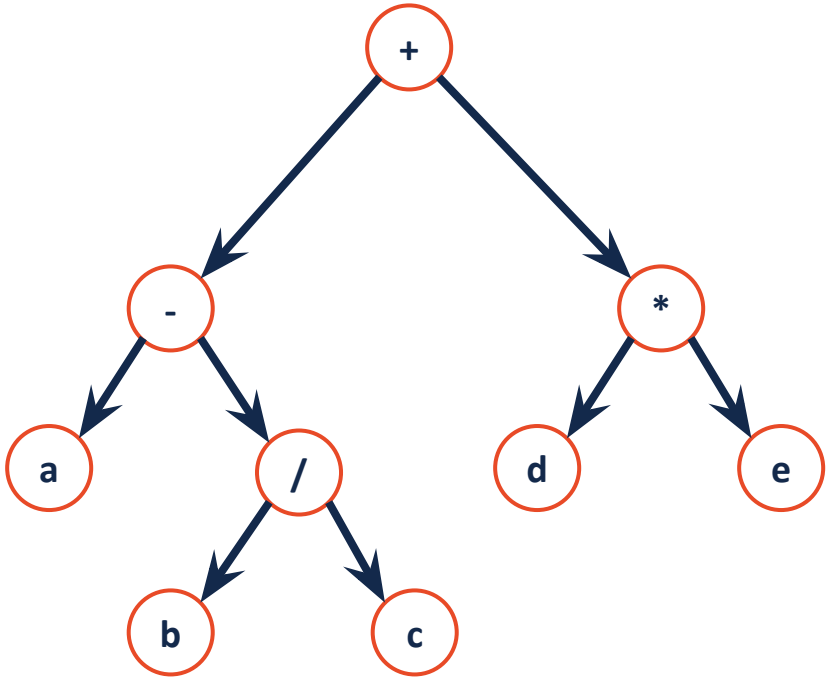
PreOrder

PostOrder

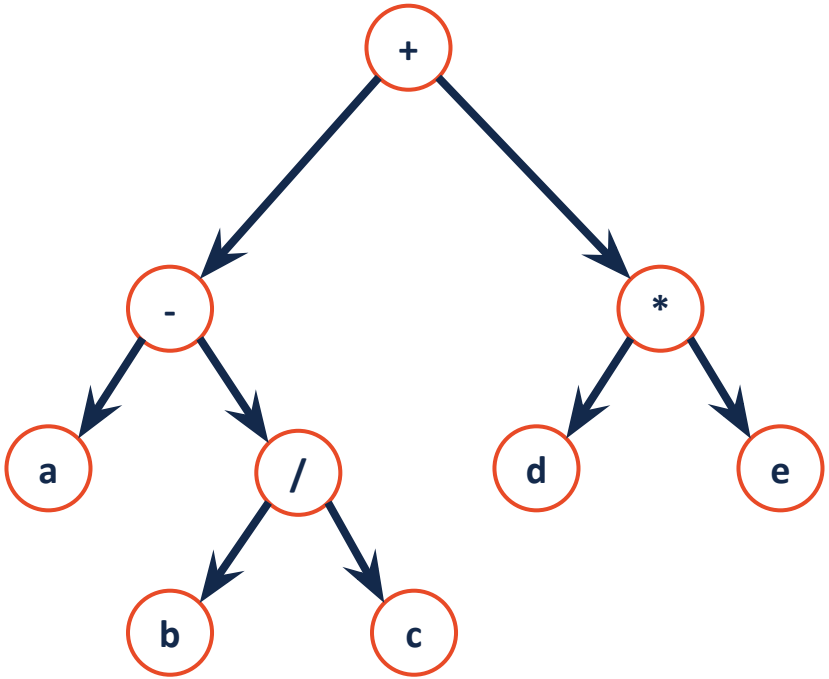
LevelOrder



Inorder Traversal

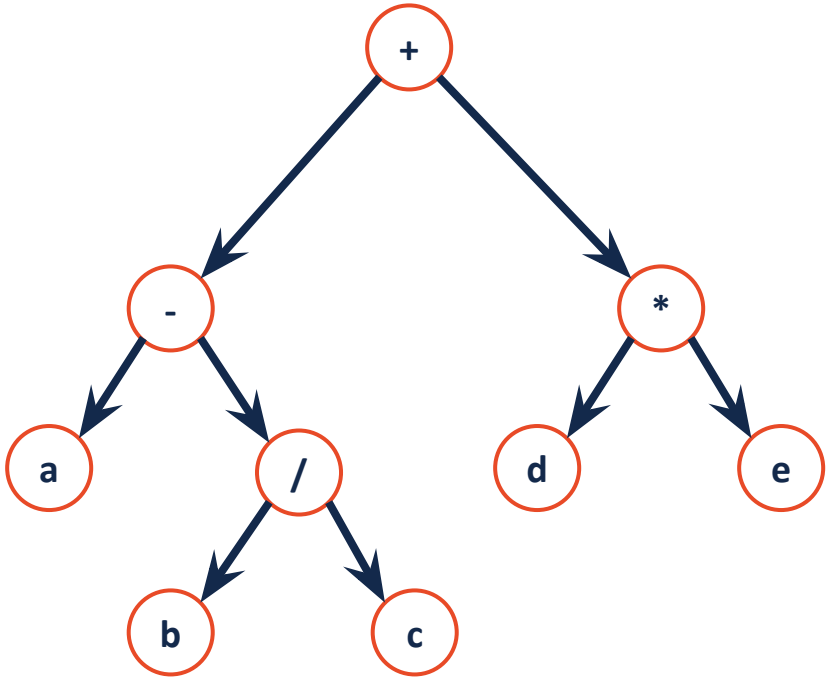


Inorder Traversal




```
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
```

Accessing all nodes



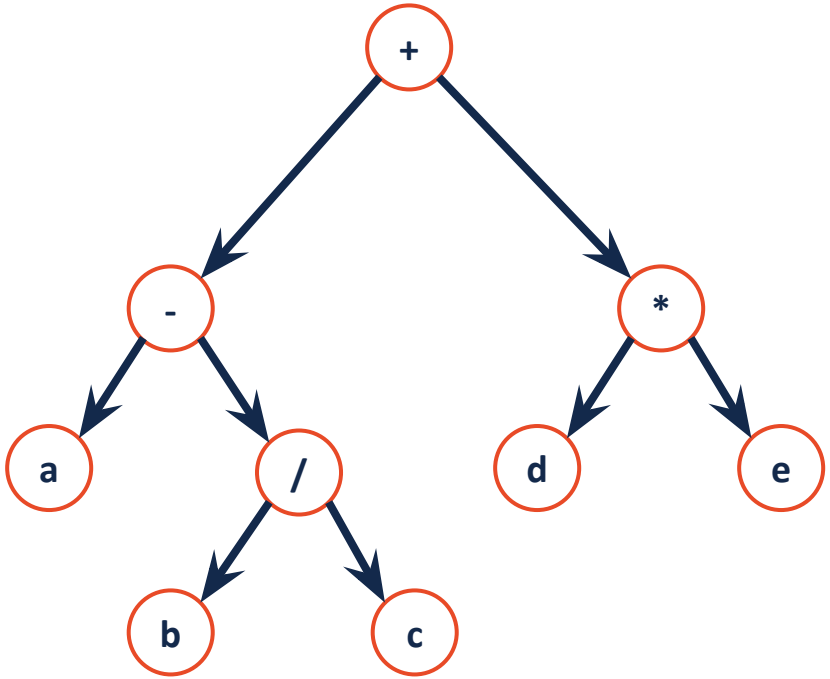
Types of Traversals

InOrder

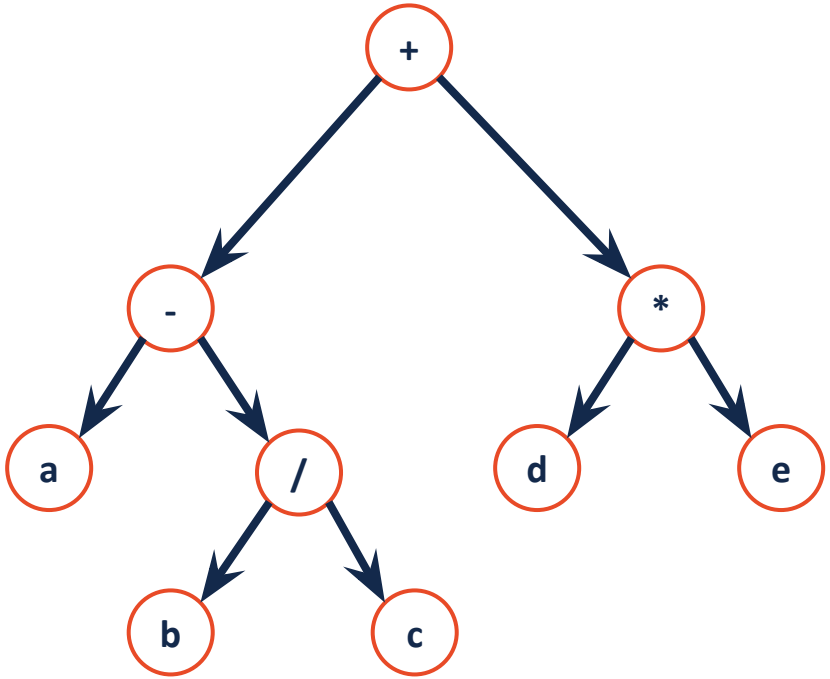
PreOrder

PostOrder

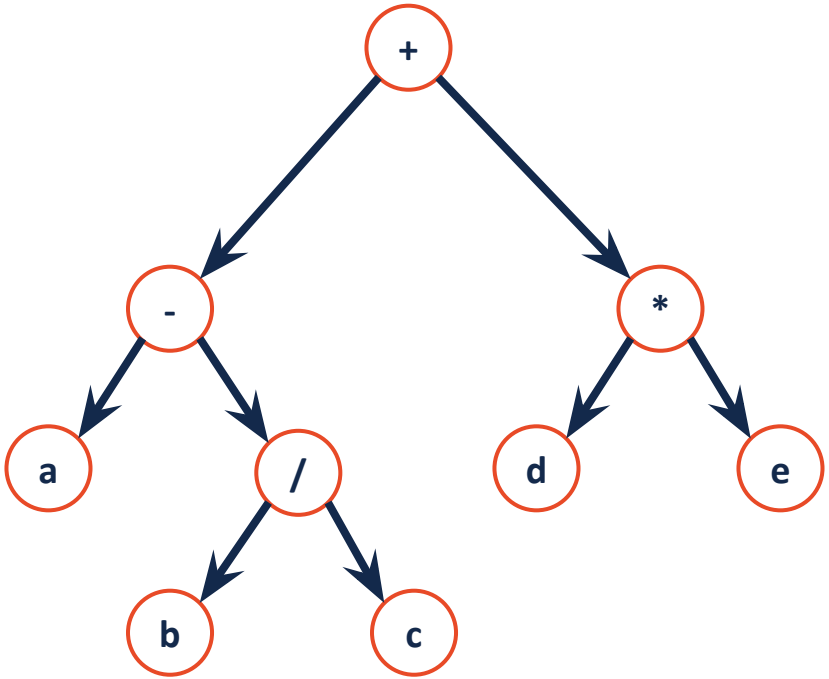
LevelOrder



Preorder Traversal

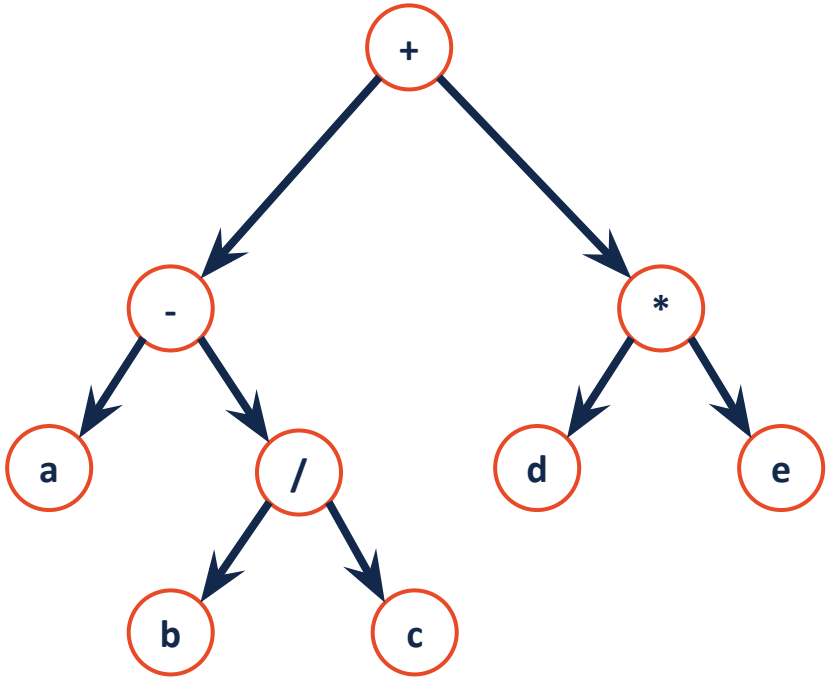


Preorder

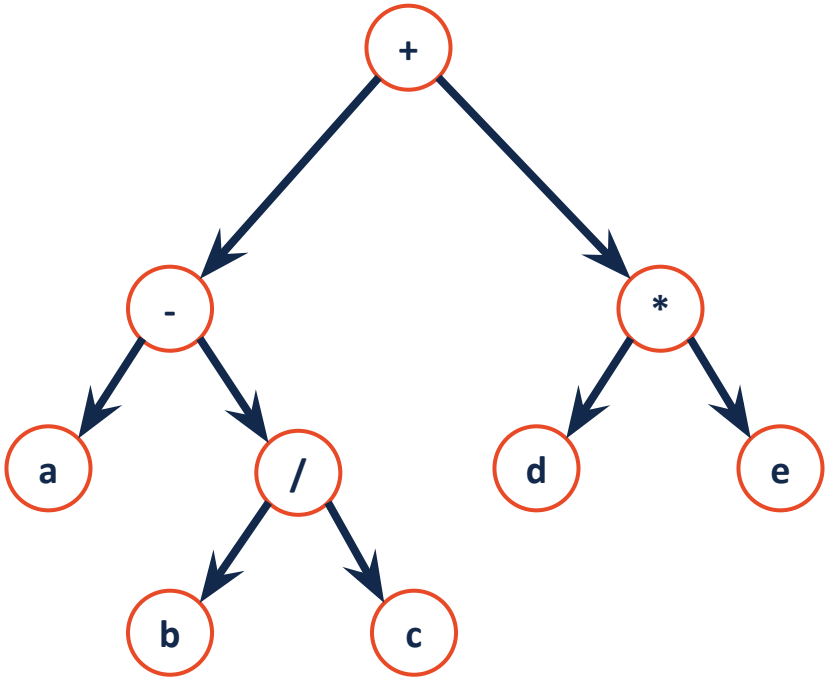



```
49  template<class T>
50  void BinaryTree<T>::PreOrder(TreeNode * cur) {
51     if (cur != NULL) {

52        std::cout << cur->data << std::endl;

53        InOrder(cur->left);

54        InOrder(cur->right);


55    }
56  }
57
```

Postorder Traversal



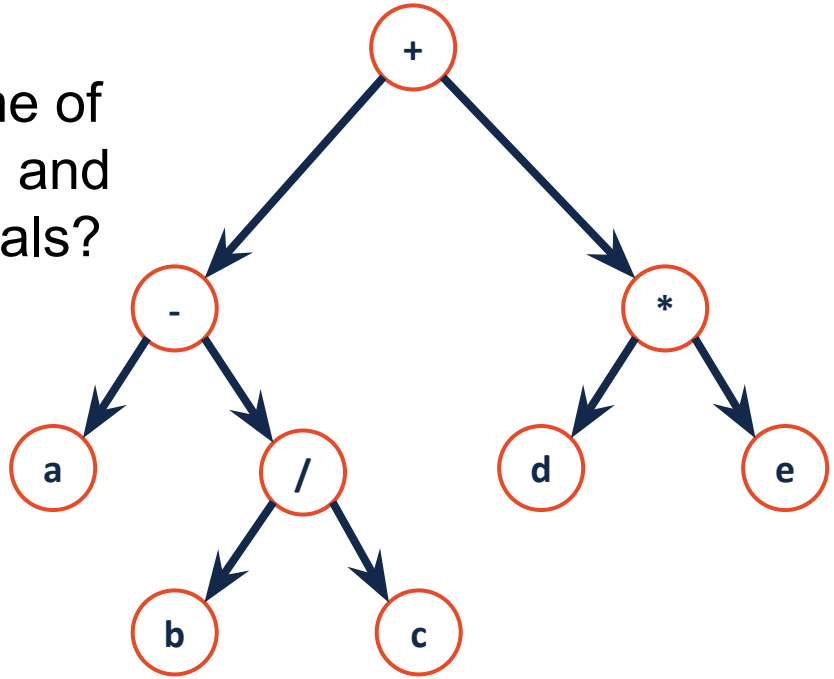
Postorder




```
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
```

Traversal Runtimes

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



Combining information

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

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