Data Structures

Tree Definitions

CS 225
Brad Solomon

September 16, 2023

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

Exam 1 (9/18 — 9/20)

Autograded MC and one coding question

Manually graded short answer prompt

Practice exam will be released on PL

Topics covered can be found on website

Registration started August 22

https://courses.engr.illinois.edu/cs225/fa2024/exams/

Learning Objectives

Review trees and binary trees

Practice tree theory with recursive definitions and proofs

Discuss the tree ADT

Explore tree implementation details

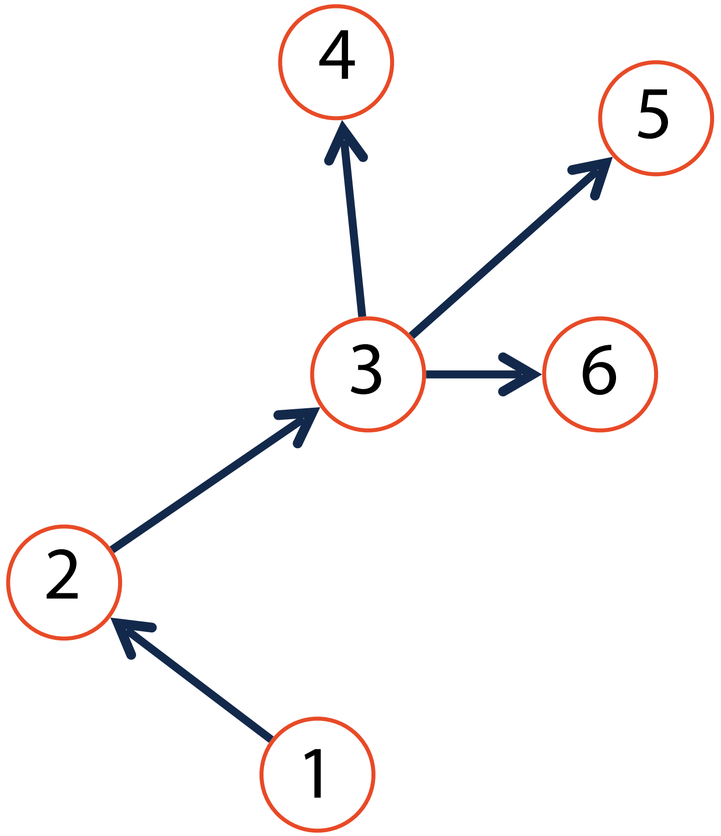
Trees

A non-linear data structure defined recursively as a collection of nodes where each node contains a value and zero or more connected nodes.

[In CS 225] a tree is also:

1) Acyclic — No path from node to itself

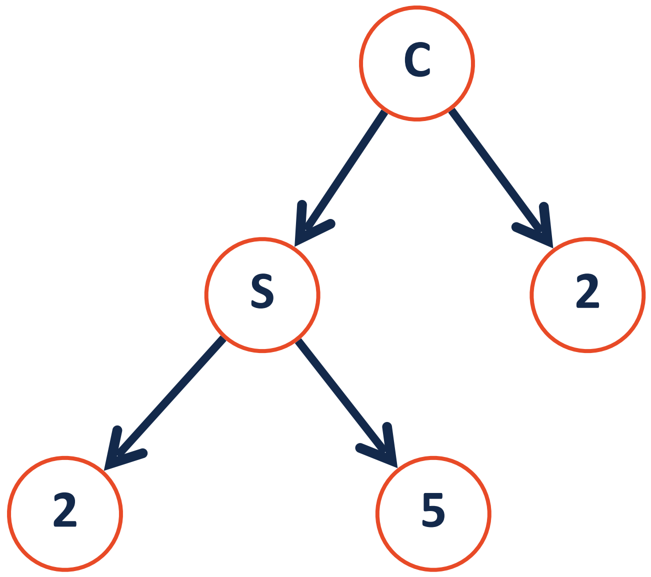
2) Rooted — A specific node is labeled root



Binary Tree

A **binary tree** is a tree T such that:

1. $T = \emptyset$



2. $T = (data, T_L, T_R)$

Binary Tree

Lets define additional terminology for different **types** of binary trees!

1.

2.

3.

Binary Tree: full

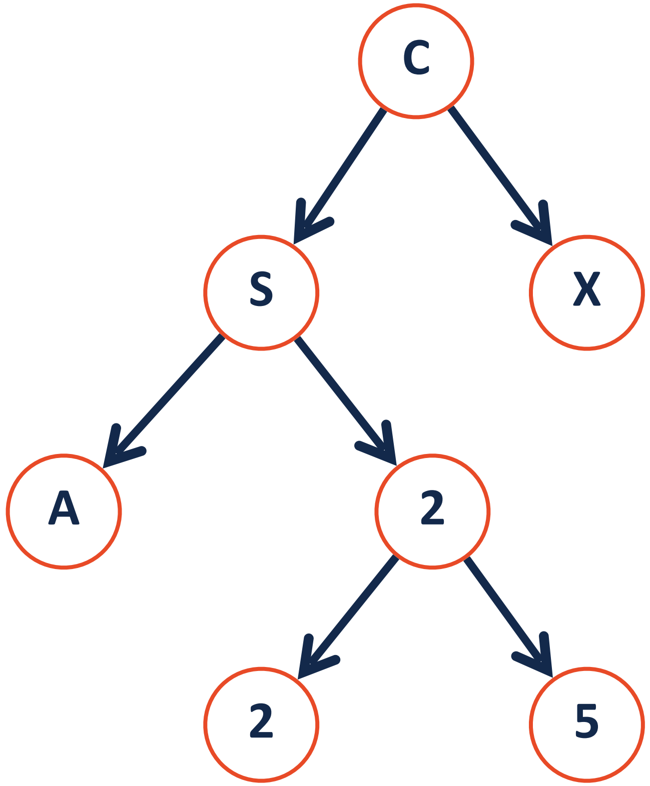
A **full tree** is a binary tree where every node has either 0 or 2 children

A tree **F** is **full** if and only if:

1.

2.

3.



Binary Tree: full

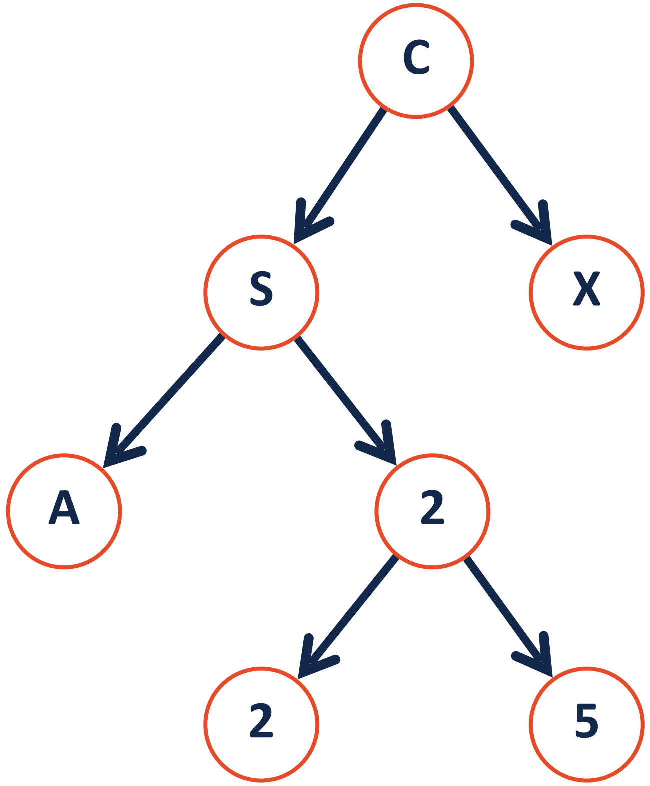
A **full tree** is a binary tree where every node has either 0 or 2 children

A tree **F** is **full** if and only if:

1. $F = \emptyset$

2. $F = (data, \emptyset, \emptyset)$

3. $F = (data, F_l \neq \emptyset, F_r \neq \emptyset)$



Binary Tree: perfect

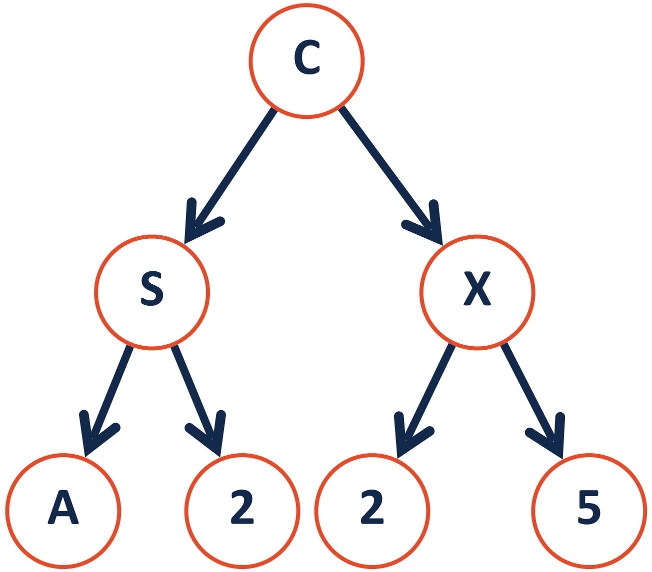
A **perfect tree** is a binary tree where…

Every internal node has 2 children and all leaves are at the same level.

A tree **P** is **perfect** if and only if:

1.

2.



Binary Tree: perfect

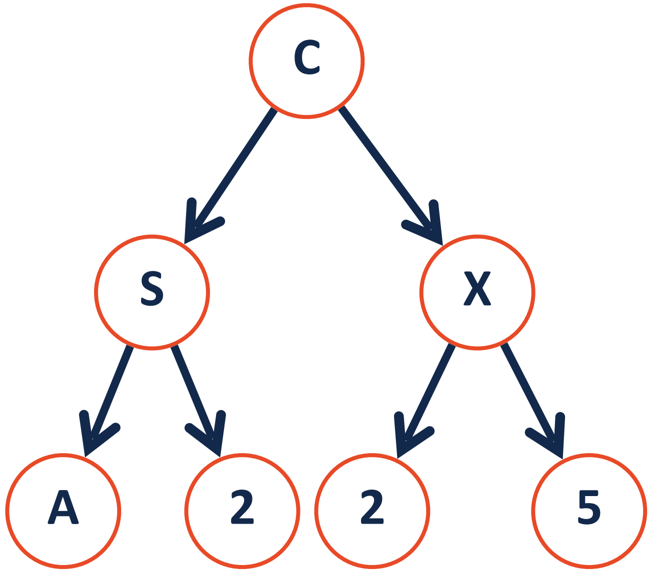
A **perfect tree** is a binary tree where…

Every internal node has 2 children and all leaves are at the same level.

A tree **P** is **perfect** if and only if:

1. $P_h = (data, P_{h-1}, P_{h-1})$

2. $P_0 = (data, \emptyset, \emptyset) \equiv \; P_{-1} = \emptyset$



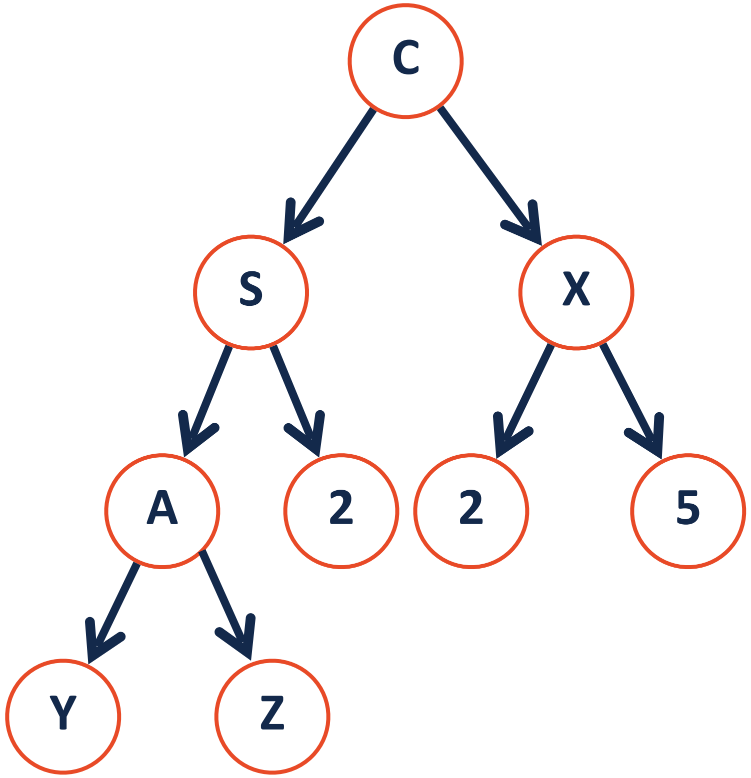
Binary Tree: complete

A **complete tree** is a B.T. where…

All levels except the last are completely filled.

The last level contains at least one node (and is pushed to left)

A tree **C** is **complete** if and only if:



1.

2.

3.

Binary Tree: complete

A **complete tree** is a B.T. where…

All levels except the last are completely filled.

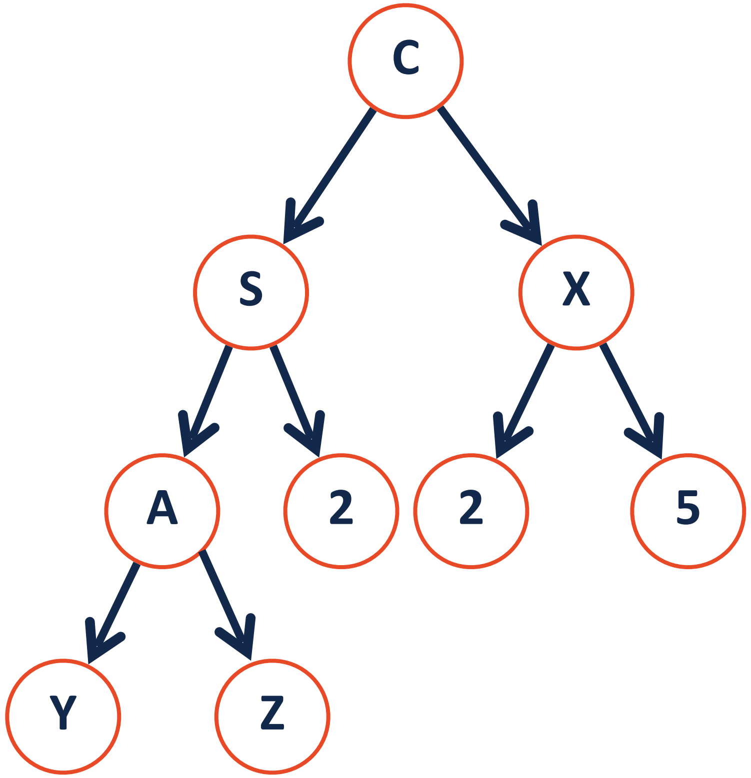
The last level contains at least one node (and is pushed to left)

A tree **C** is **complete** if and only if:

1. $C_h = (data, C_{h-1}, P_{h-2})$

2. $C_h = (data, P_{h-1}, C_{h-1})$

3. $C_{-1} = \emptyset$



Binary Tree

Why do we care?

1. Terminology instantly defines a particular tree structure

2. Understanding how to think 'recursively' is very important.

Binary Tree: Thinking with Types

Is every **full** tree **complete**?

Is every **complete** tree **full**?

Binary Tree: Practicing Proofs

Theorem: If there are **n** objects in our representation of a binary tree, then there are _________ NULL pointers.

Binary Tree: Practicing Proofs

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

Base Case:

Binary Tree: Practicing Proofs

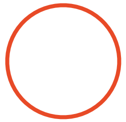
Theorem: If there are **n** objects in our representation of a binary tree, then there are **n+1** NULL pointers.

Base Case:

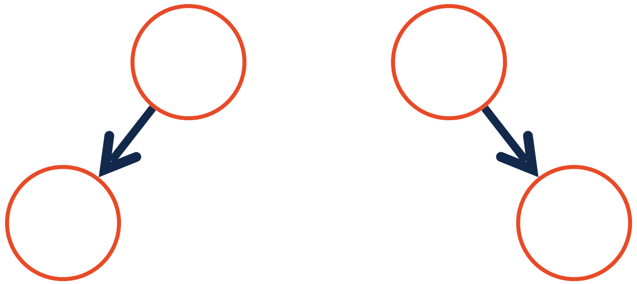
Let F(n) be the max number of NULL pointers in a tree of n nodes

N=0 has one NULL

N=1 has two NULL



N=2 has three NULL



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

Induction Step:

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

IS: Assume claim is true for $|T| \leq k-1$**, prove true for** $|T| = k$

By def, $T = r, T_L, T_R$. Let q be the # of nodes in T_L

Since r exists, $0 \leq q \leq k-1$. By IH, T_L has $q+1$ NULL

All nodes not in r or T_L exist in T_R. So T_R has $k-q-1$ nodes

$k-q-1$ is also smaller than k so by IH, T_R has $k-q$ NULL

Total number of NULL is the sum of T_L and T_R: $q+1+k-q = k+1$

Tree ADT

Insert

Remove

Traverse

Find

Constructor

BinaryTree.h

```
#pragma once

template <class T>
class BinaryTree {
  public:
    /* ... */

  private:
















};
```

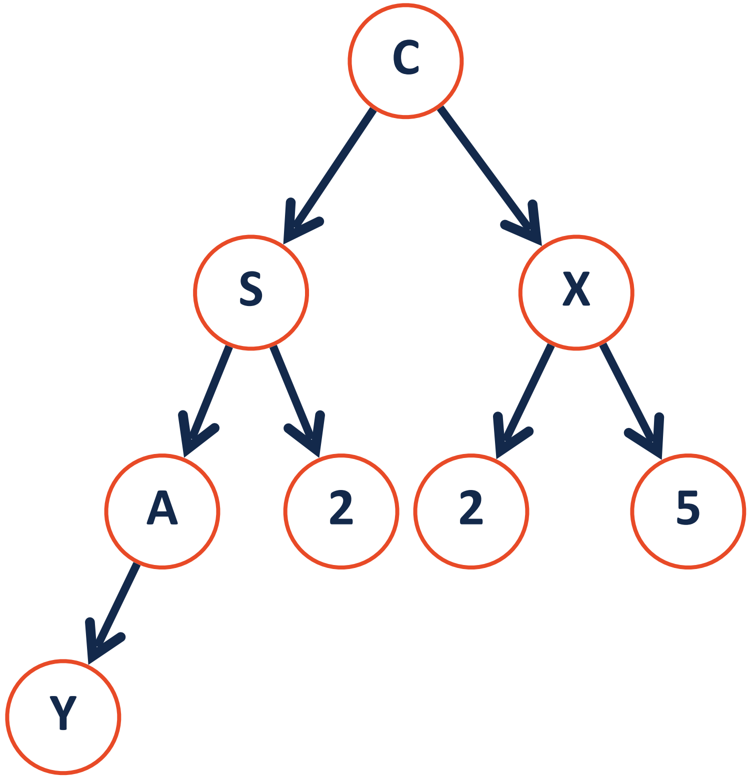
List.h

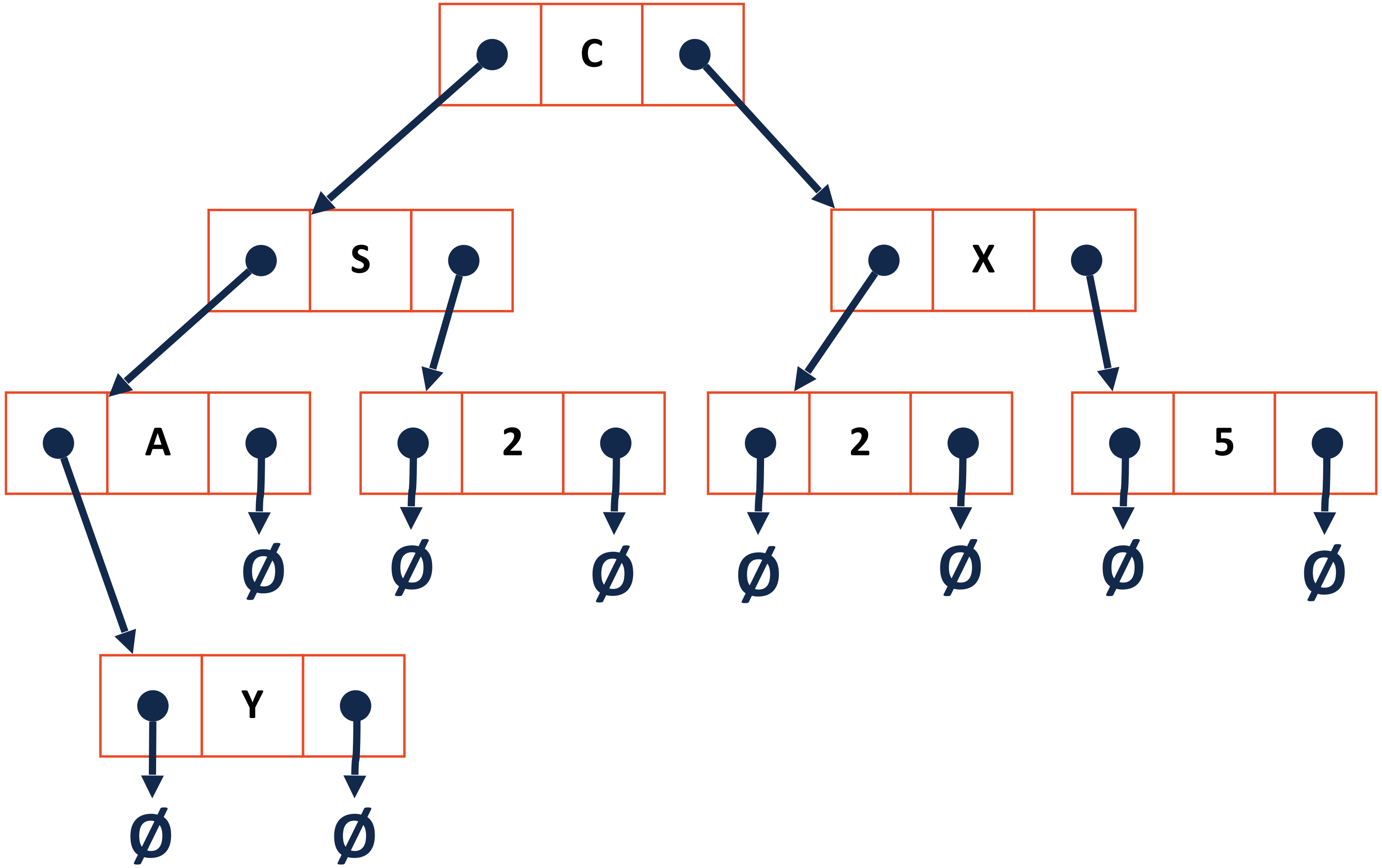
```
#pragma once

template <typename T>
class List {
  public:
    /* ... */
  private:
    class ListNode {
      T & data;

      ListNode * next;



      ListNode(T & data) :
       data(data), next(NULL) { }
    };



    ListNode *head_;
    /* ... */
};
```

Tree.h

```
#pragma once

template <typename T>
class BinaryTree {
  public:
    /* ... */
  private:
    class TreeNode {
      T & data;

      TreeNode * left;

      TreeNode * right;

      TreeNode(T & data) :
       data(data), left(NULL),
right(NULL) { }

    };

    TreeNode *root_;
    /* ... */
};
```

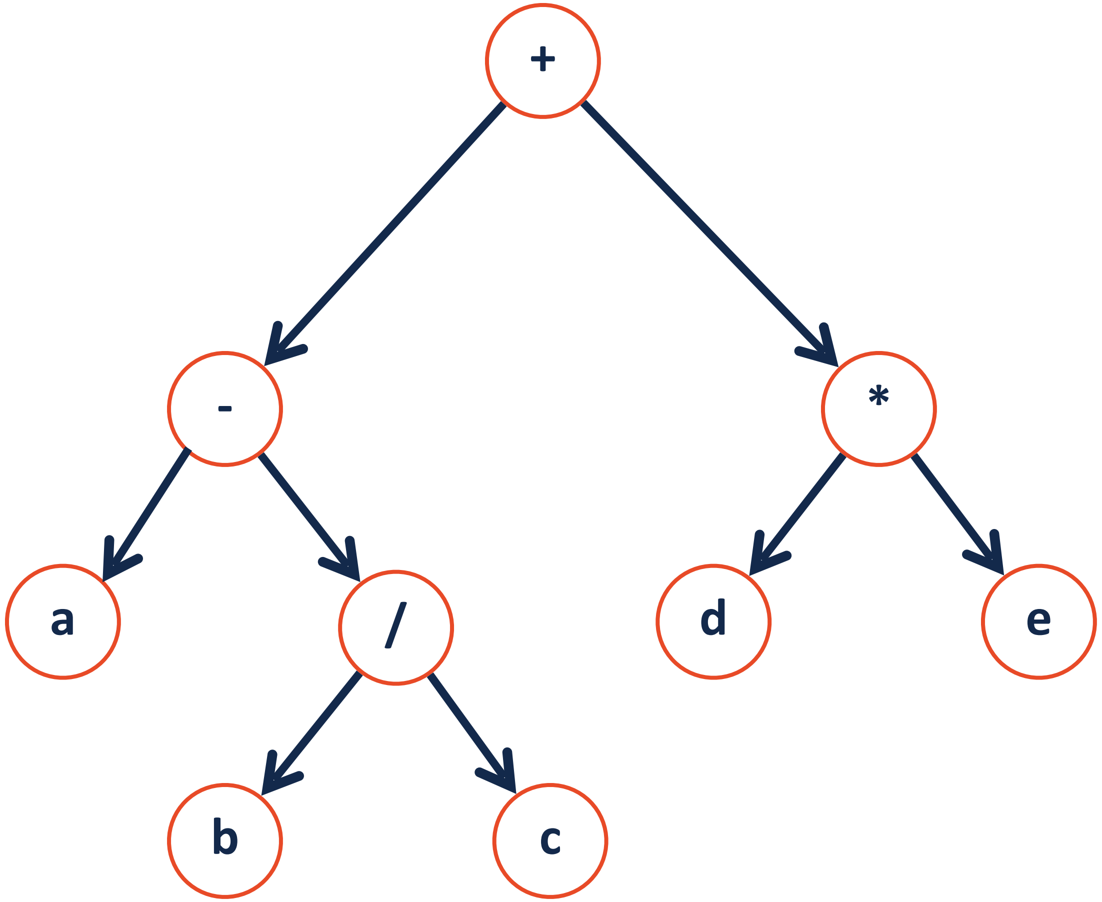
Visualizing trees



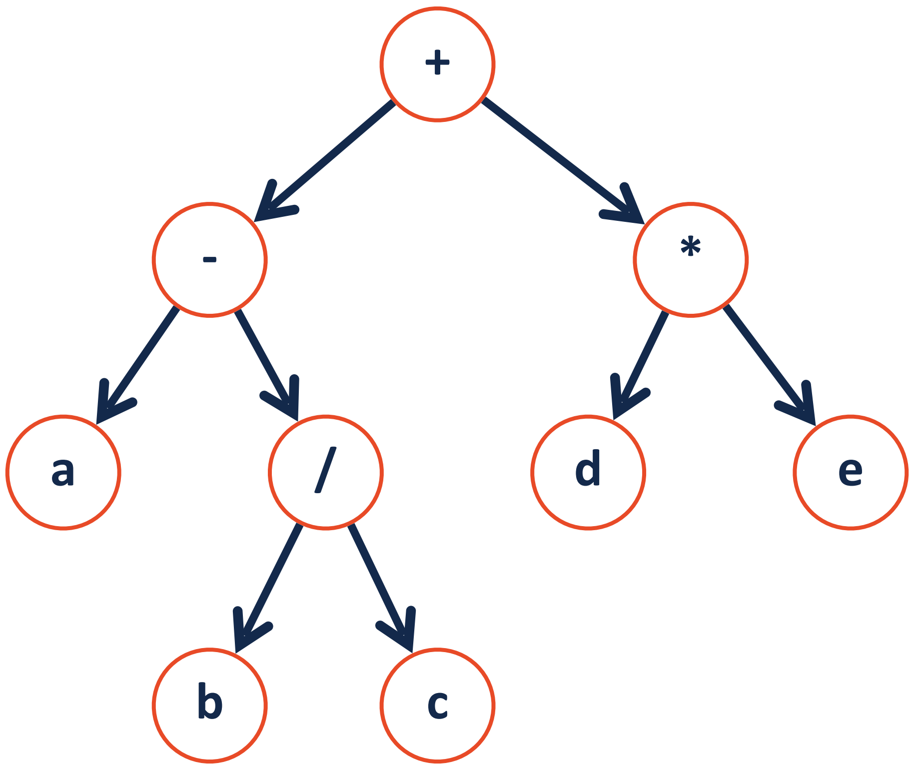


Tree Traversal

A **traversal** of a tree T is an ordered way of visiting every node once.



Traversals




```
template<class T>
void BinaryTree<T>::______Order(TreeNode * root)
{


















}
```