CS 225

Data Structures

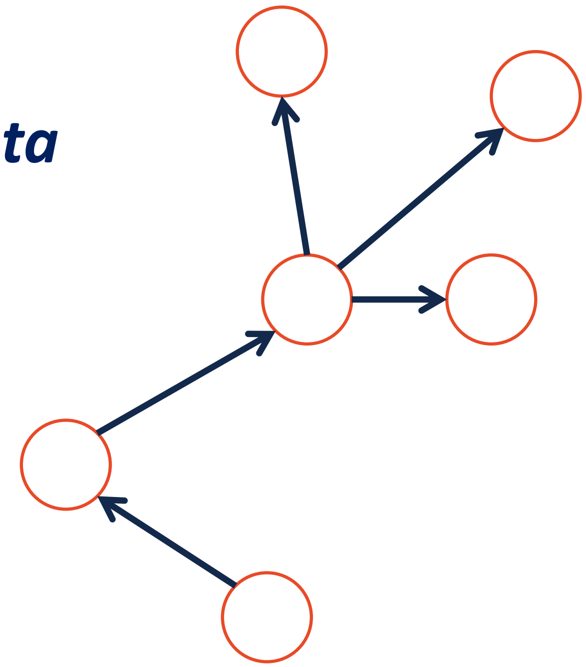
February 5 –Trees

G Carl Evans

Trees

"The most important non-linear data structure in computer science."

- David Knuth, The Art of Programming, Vol. 1



A tree is:

-

-

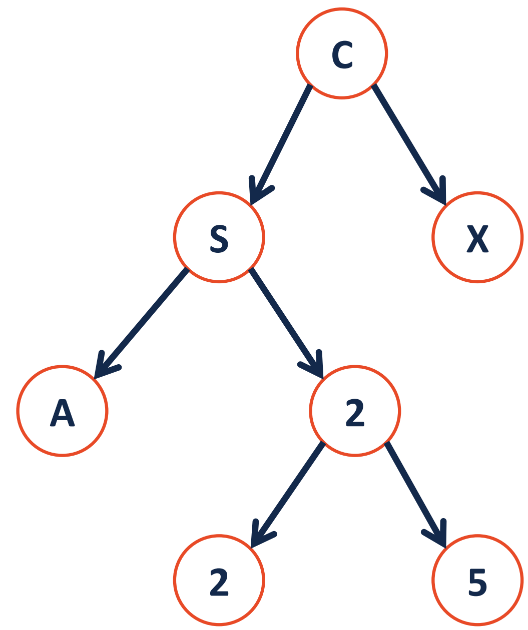
Binary Tree – Defined

A *binary tree* T is either:

-

OR

-



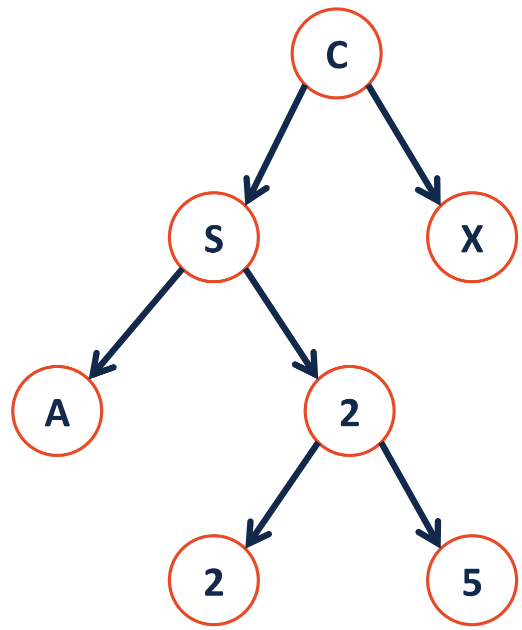
Binary Tree – Defined

A *binary tree* T is either:

- $T = \emptyset$

OR

- $T = (r, T_L, T_R)$

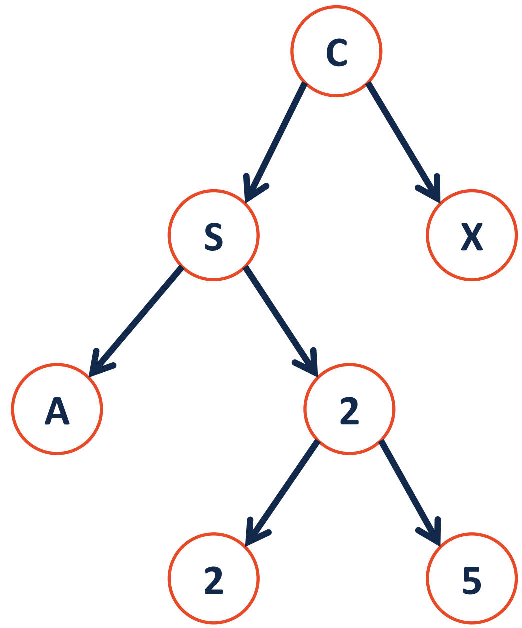


Tree Property: height

height(T): length of the longest path from the root to a leaf

Given a binary tree T:

height(T) =



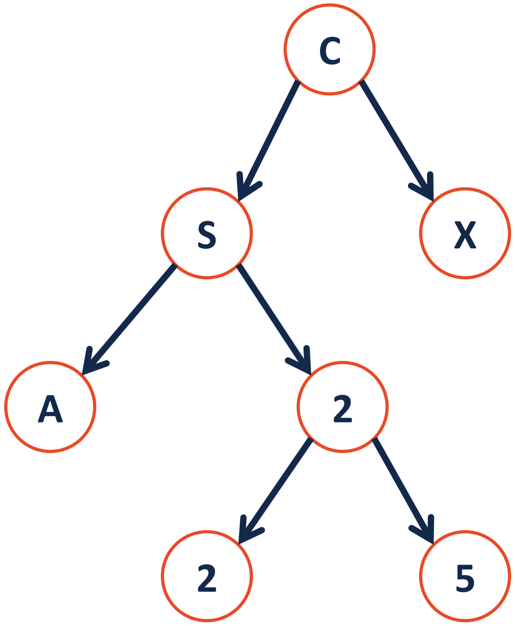
Tree Property: height

height(T): length of the longest path from the root to a leaf

Given a binary tree T:

$$\textit{height}(T) = \max(\textit{height}(T_L),\ \textit{height}(T_R)) + 1$$

$$\textit{height}(\emptyset) = -1$$

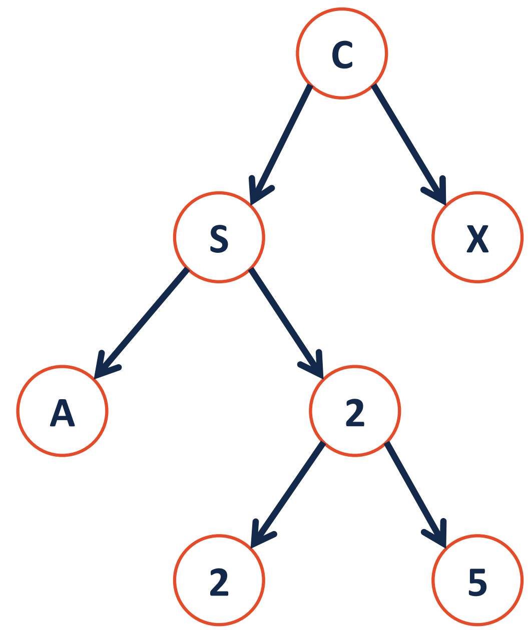


Tree Property: full

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

1.

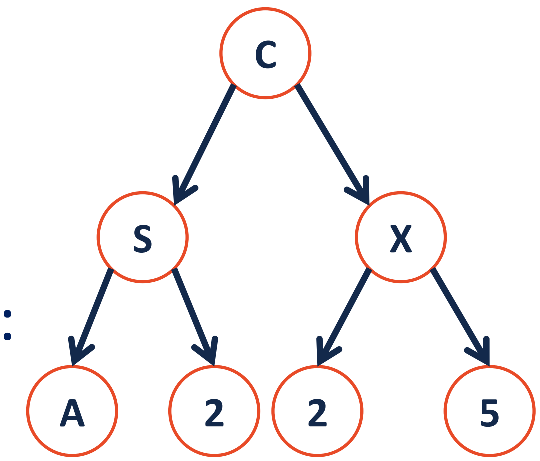
2.



Tree Property: perfect

A **perfect** tree ***P*** is defined in terms of the tree's height.

Let $\mathbf{P_h}$ be a perfect tree of height **h**, and:



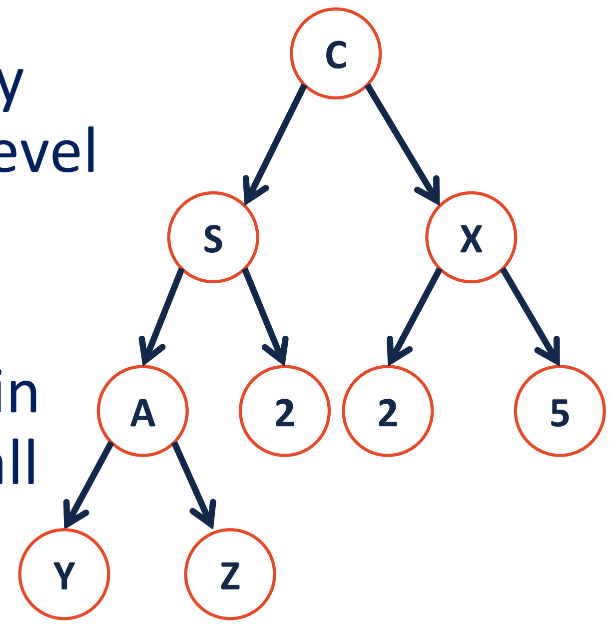
1.

2.

Tree Property: complete

Conceptually: A perfect tree for every level except the last, where the last level if "pushed to the left".

Slightly more formal: For all levels k in [0, h-1], k has 2^k nodes. For level h, all nodes are "pushed to the left".



Tree Property: complete

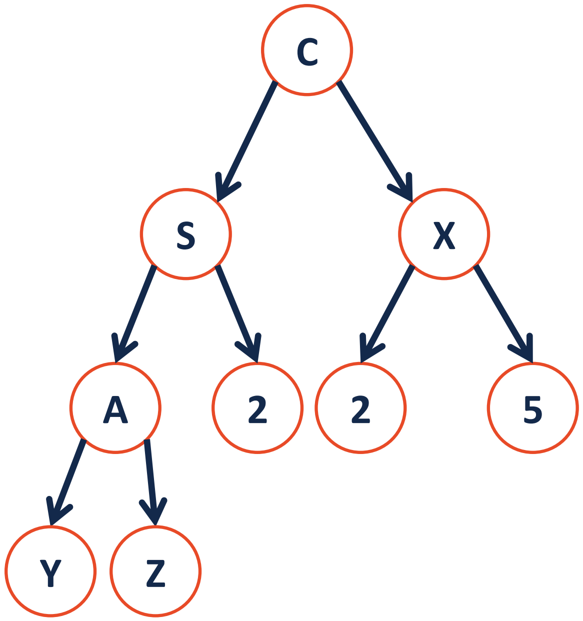
A **complete** tree ***C*** of height **h**, $\mathbf{C_h}$:

1. $\mathbf{C_{-1}}$ = {}
2. $\mathbf{C_h}$ *(where h>0)* **= {r, T_L, T_R}** and either:

T_L is ___________ and T_R is __________

OR

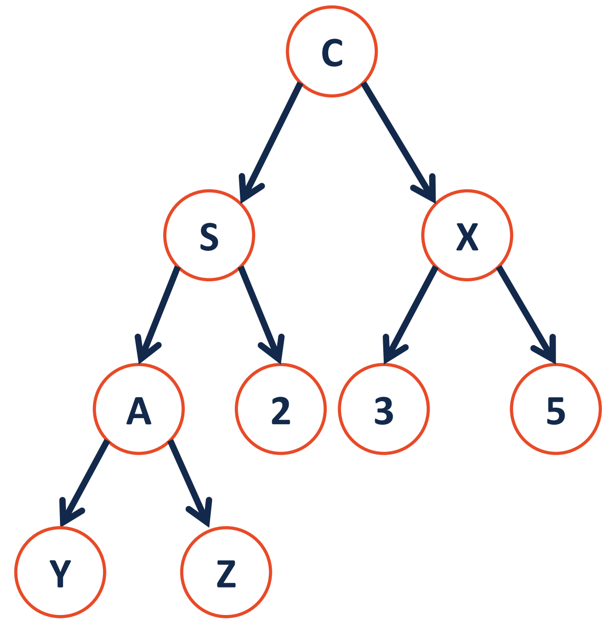
T_L is ___________ and T_R is __________



Tree Property: complete

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

If every **complete** tree **full**?



Tree ADT

insert, inserts an element to the tree.

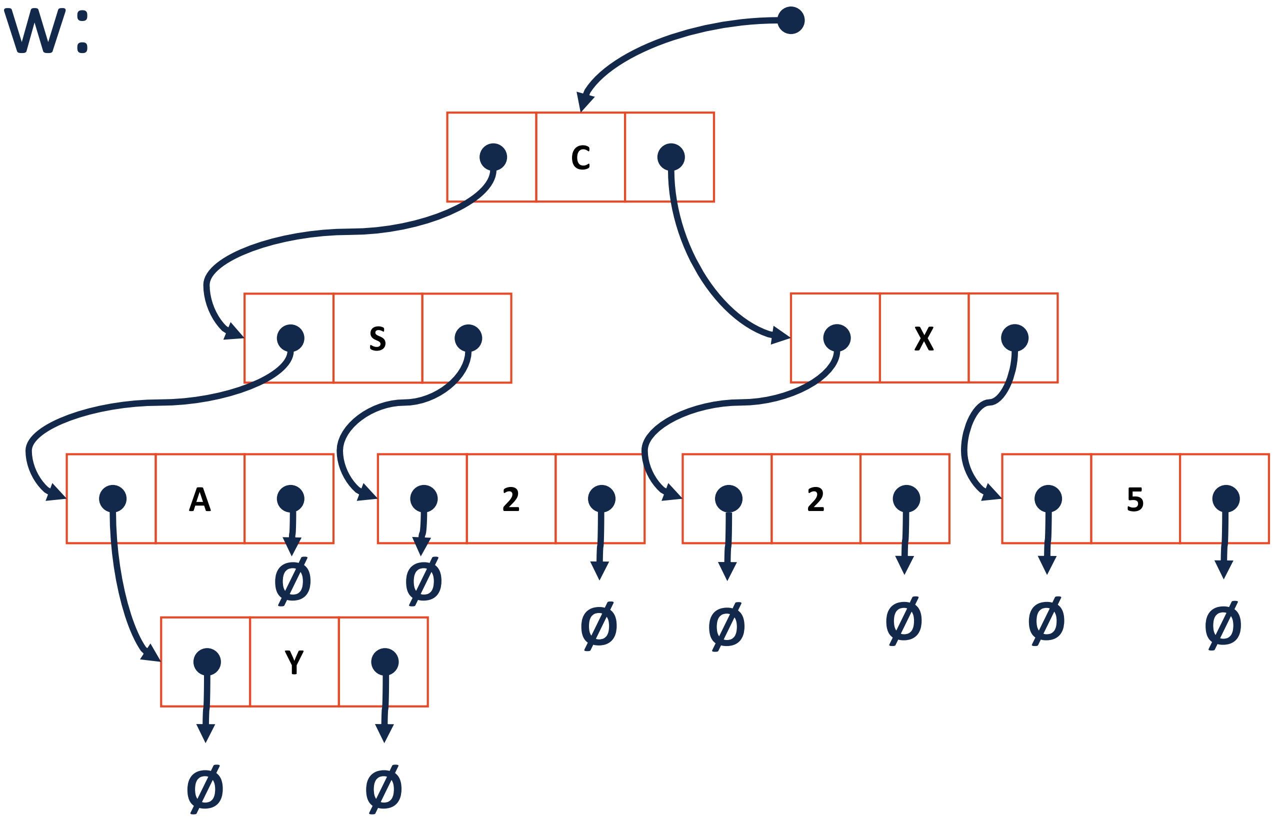
remove, removes an element from the tree.

access, access elements from the tree.

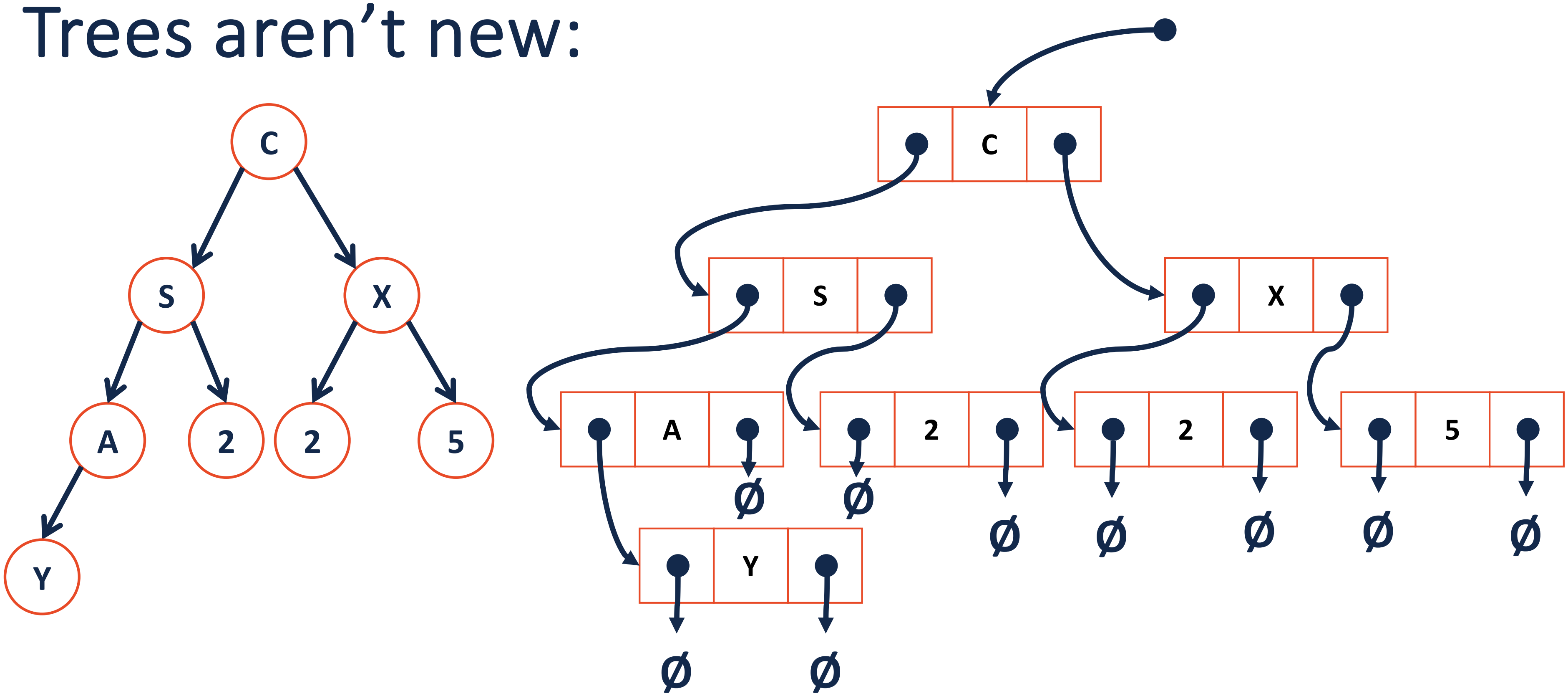
BinaryTree.h

```
#pragma once

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

  private:










};
```

Trees aren’t new:



Trees aren't new:



How many NULLs?

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

How many NULLs?

Base Cases:

n = 0:

n = 1:

n = 2:

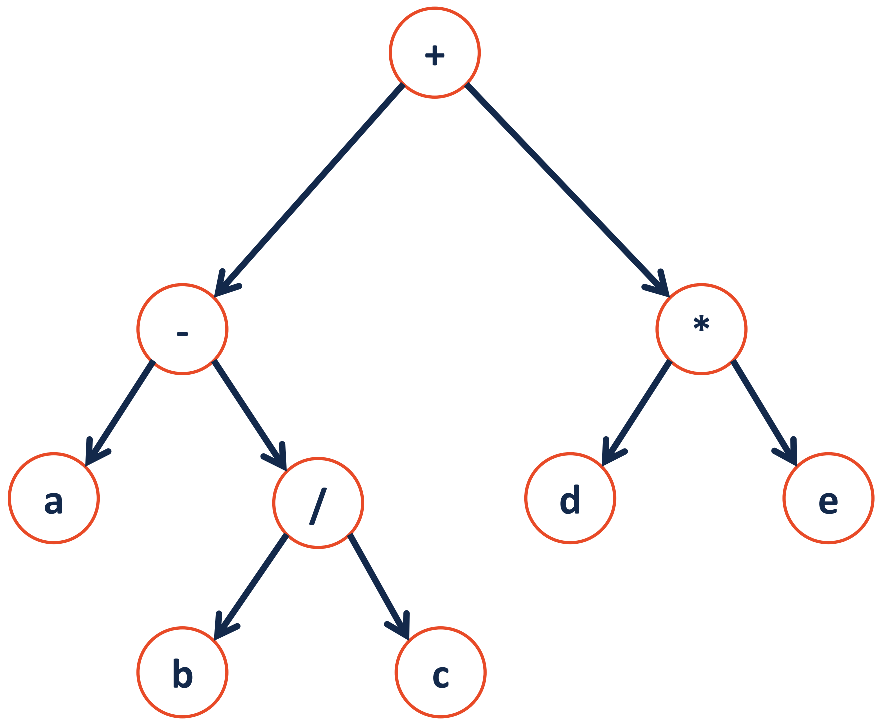
How many NULLs?

Induction Hypothesis:

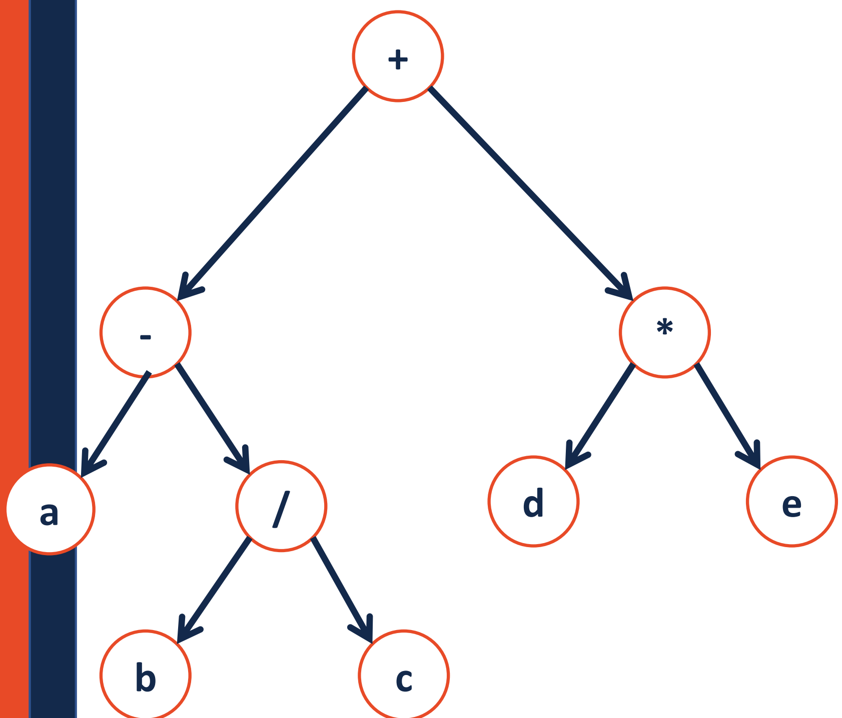
How many NULLs?

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

Access All the Nodes - Traversals

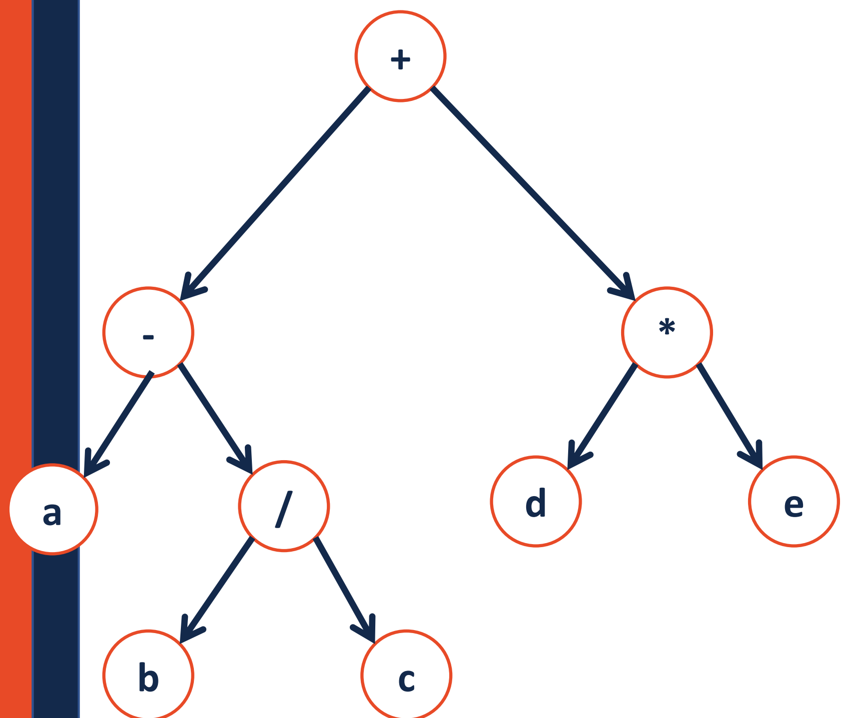


Traversals



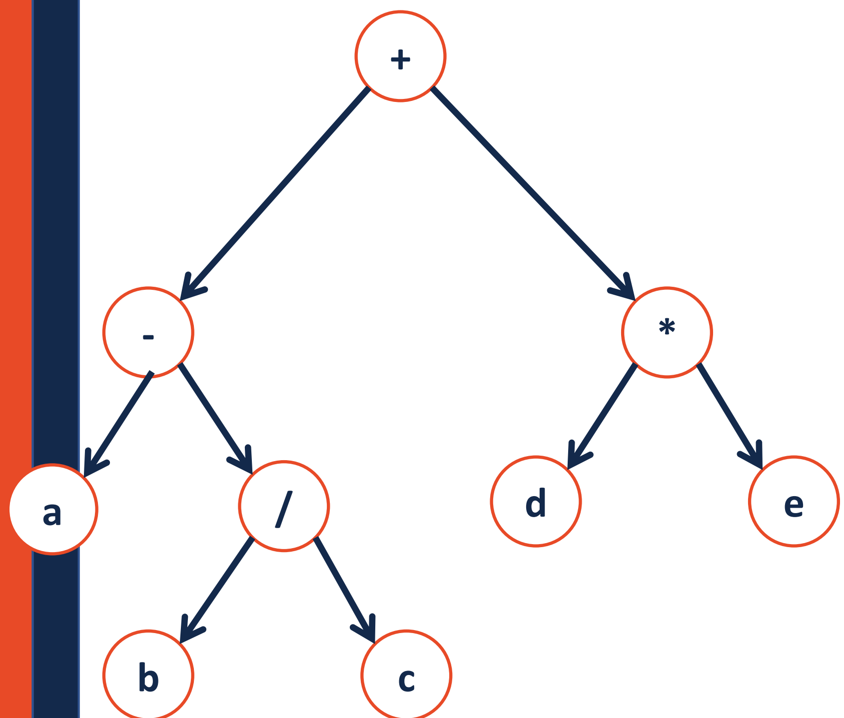

```
template<class T>
void BinaryTree<T>::__Order(TreeNode * cur)
{







}
```

Traversals



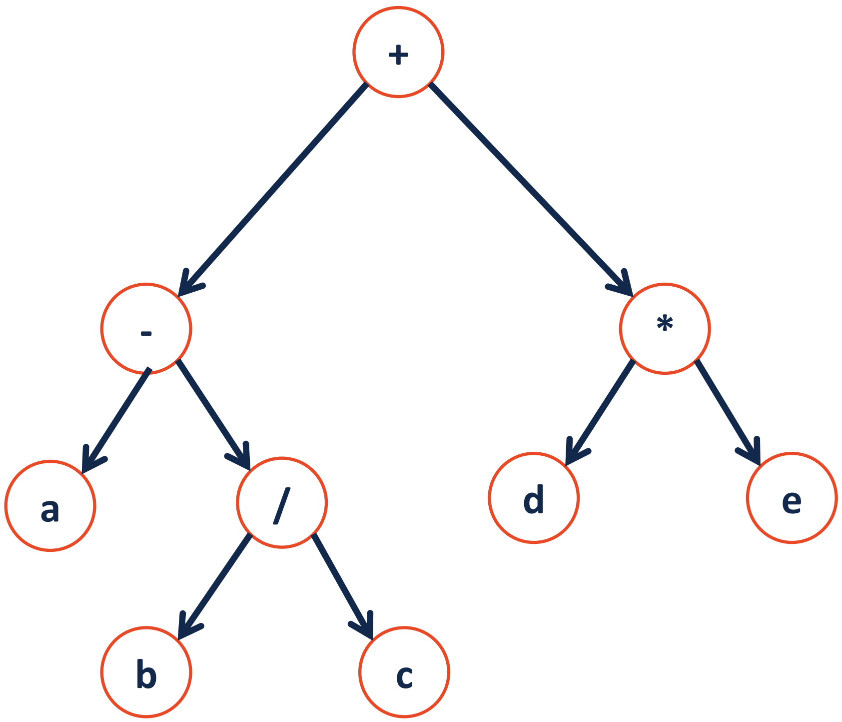

```
template<class T>
void BinaryTree<T>::___Order(TreeNode * cur) {
   if (cur != NULL) {

      _______________________;

      ___Order(cur->left);

      _______________________;

      ___Order(cur->right);

      _______________________;

   }
}
```

Traversals

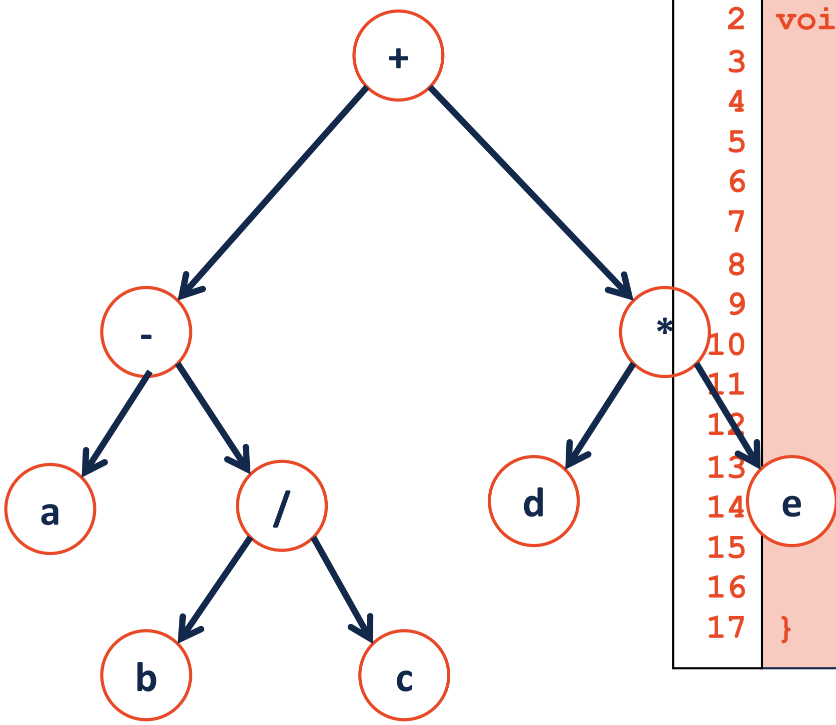



```
template<class T>
void BinaryTree<T>::___Order(TreeNode * cur) {
   if (cur != NULL) {

      _______________________;

      ___Order(cur->left);

      _______________________;

      ___Order(cur->right);

      _______________________;

   }
}
```

A Different Type of Traversal



A Different Type of Traversal




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

Traversal vs. Search

Traversal

Search

Search: Breadth First vs. Depth First

Strategy: Breadth First Search (BFS)

Strategy: Depth First Search (DFS)