# CS 225

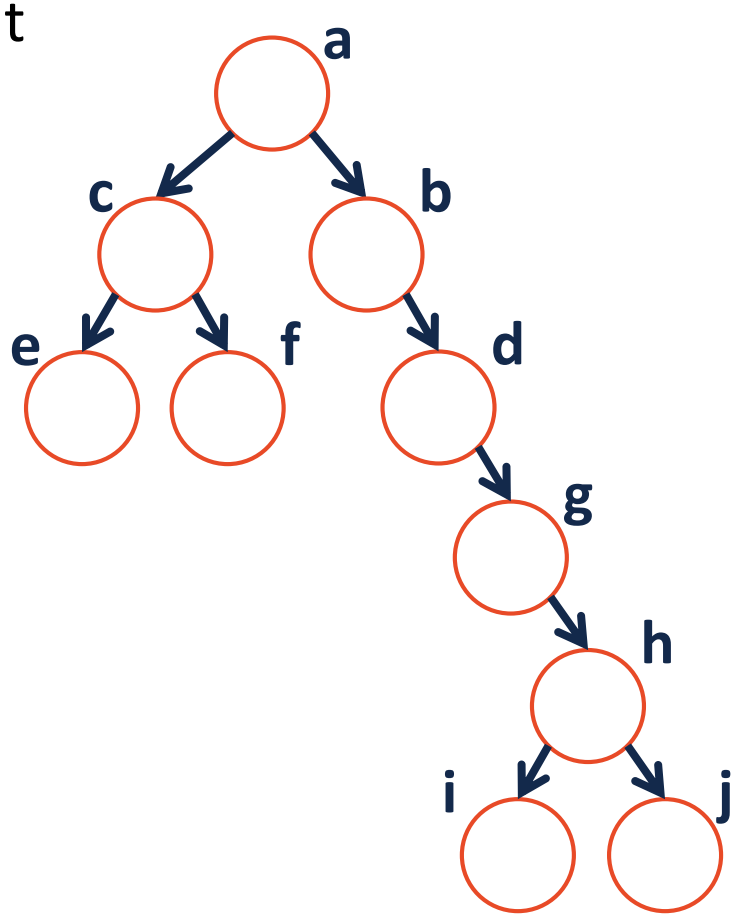
**Data Structures**

*September 28 – Tree Proof*

*Wade Fagen-Ulmschneider*

# Tree Terminology

- Find an **edge** that is not on the longest **path** in the tree. Give that edge a reasonable name. **Edge: ac**
- One of the vertices is called the **root** of the tree. Which one? **Vertex a is the root.**
- Make an "word" containing the names of the vertices that have a **parent** but no **sibling**. **No word with just bgh.** ☹
- How many parents does each vertex have?
- Which vertex has the fewest **children**?
- Which vertex has the most **ancestors**?
- Which vertex has the most **descendants**?
- List all the vertices is b's left **subtree**.
- List all the **leaves** in the tree.



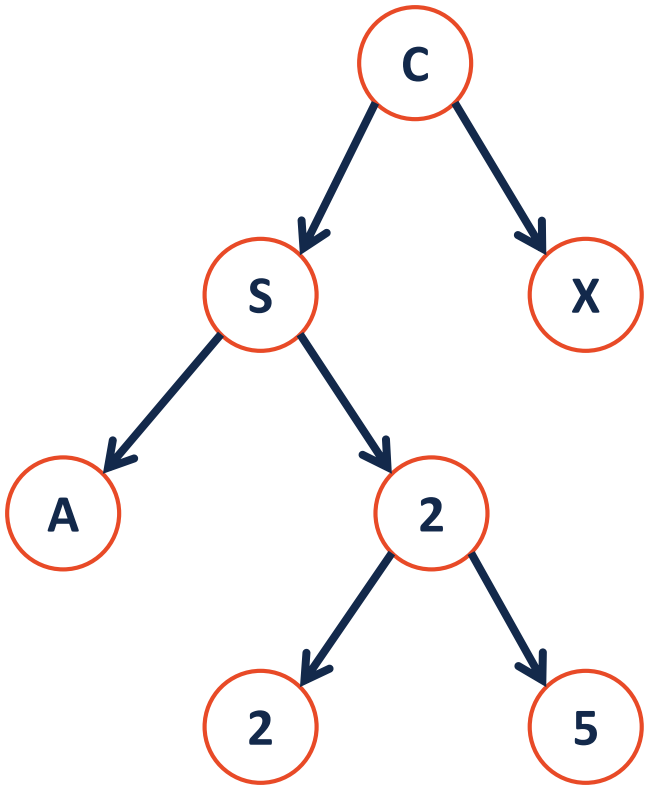
# Binary Tree – Defined

**A *binary tree* T is either:**

- 

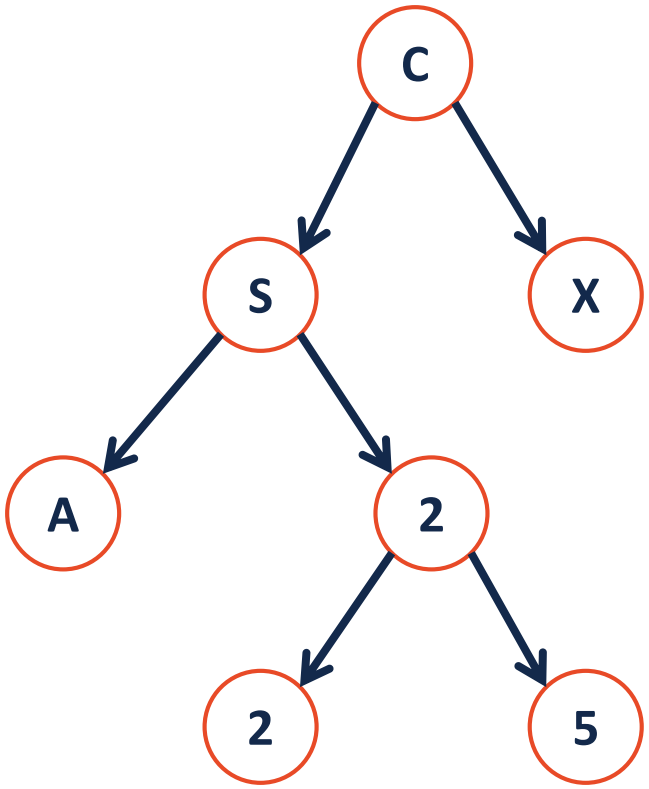
**OR**

- 



# Tree Property: height

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




**Given a binary tree T:**

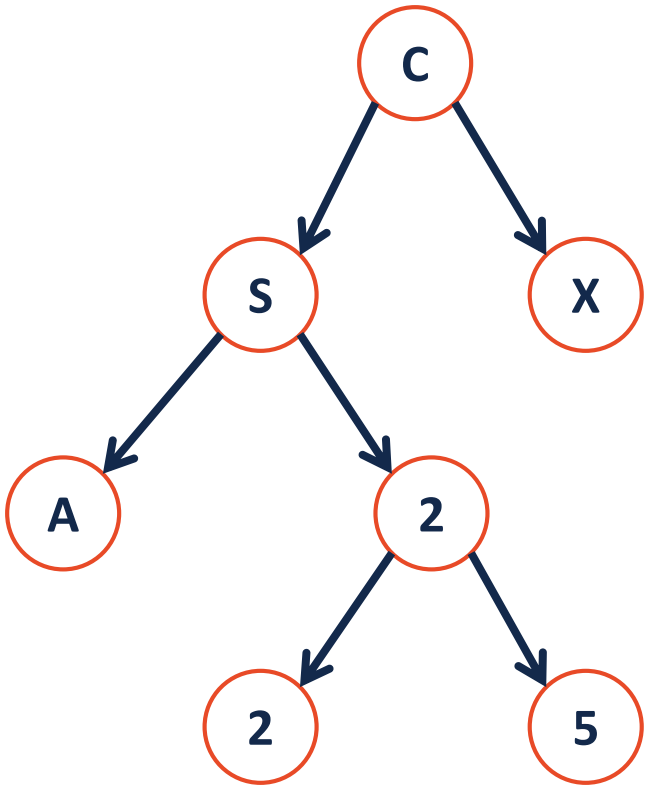
***height(T) =***

# Tree Property: full

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

1.

2.



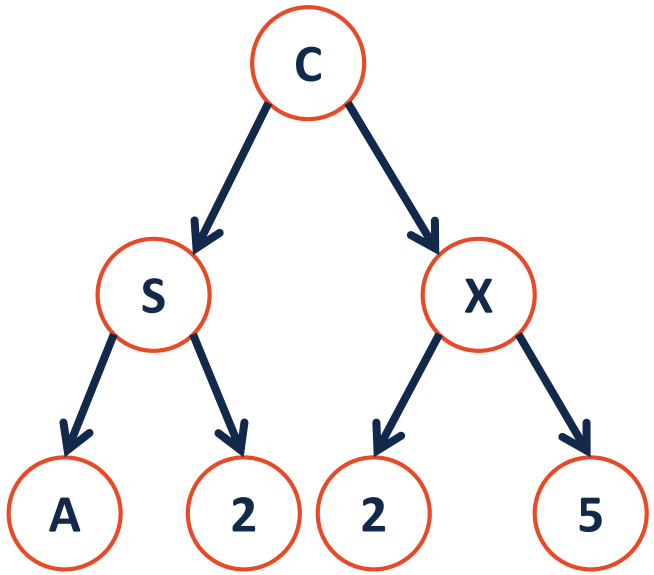
# Tree Property: perfect

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.

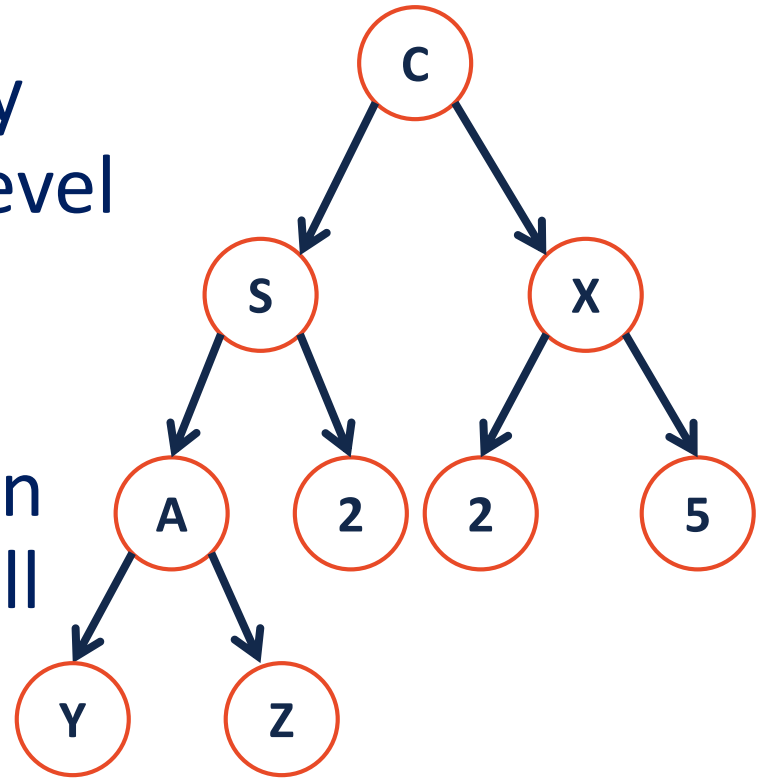
2.



# Tree Property: complete

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

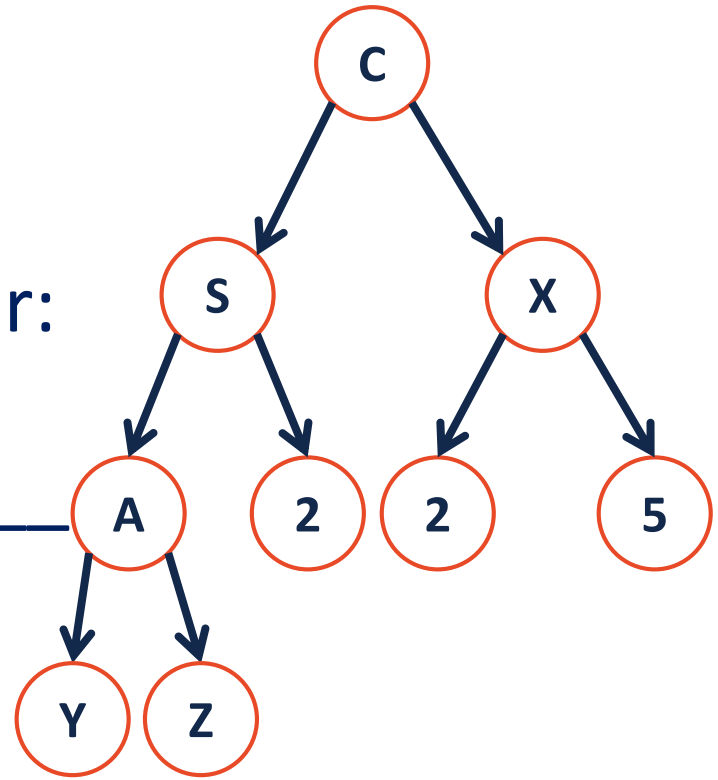
**Slightly more formal**: For all levels k in [0, h-1], k has $2^k$ nodes. For level h, all nodes are "pushed to the left".



# 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, $\mathbf{T_L}$, $\mathbf{T_R}$}** and either:




$T_L$ is _____ and $T_R$ is _____

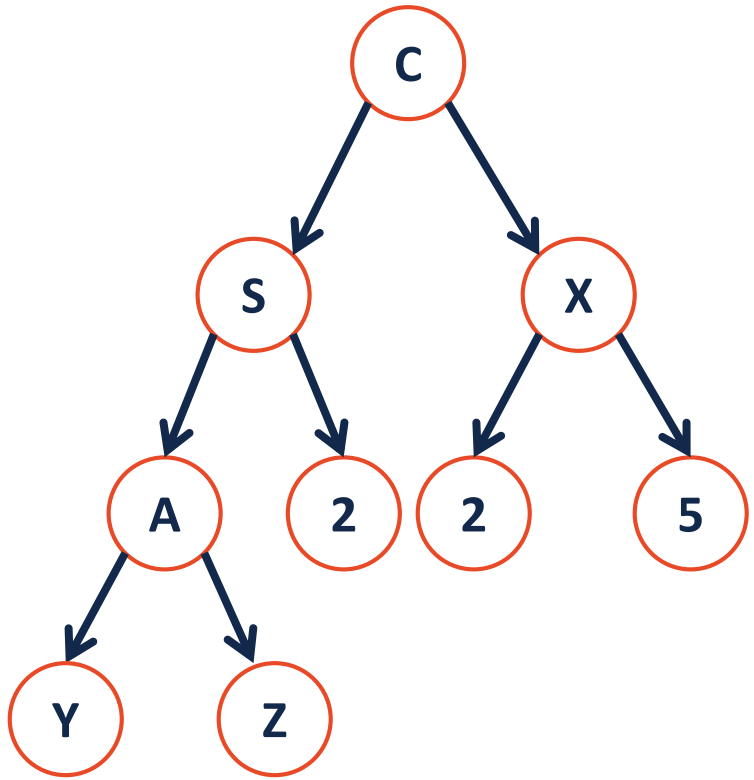
**OR**

$T_L$ is _____ and $T_R$ is _____

# Tree Property: complete

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

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



# Open Office Hours

# Open Office Hours

CS 225 has **over 50 hours of open office hours each week**, lots of time to get help!

# Open Office Hours

CS 225 has **over 50 hours of open office hours each week**, lots of time to get help!

**1.** Understand the problem, don't just give up.

- "I segfaulted" is not enough. *Where? Any idea why?*

# Open Office Hours

CS 225 has **over 50 hours of open office hours each week**, lots of time to get help!

**2.** Your topic must be specific to one function, one test case, or one exam question.

- Helps us know what to focus on before we see you!
- Helps your peers to ensure all get questions answered!

# Open Office Hours

CS 225 has **over 50 hours of open office hours each week**, lots of time to get help!

**3.** Get stuck, get help – not the other way around.

- If you immediately re-add yourself, you're setting yourself up for failure.

# Open Office Hours

CS 225 has **over 50 hours of open office hours each week**, lots of time to get help!

**4.** Be awesome.

# Tree ADT

# Tree ADT

**insert**, inserts an element to the tree.

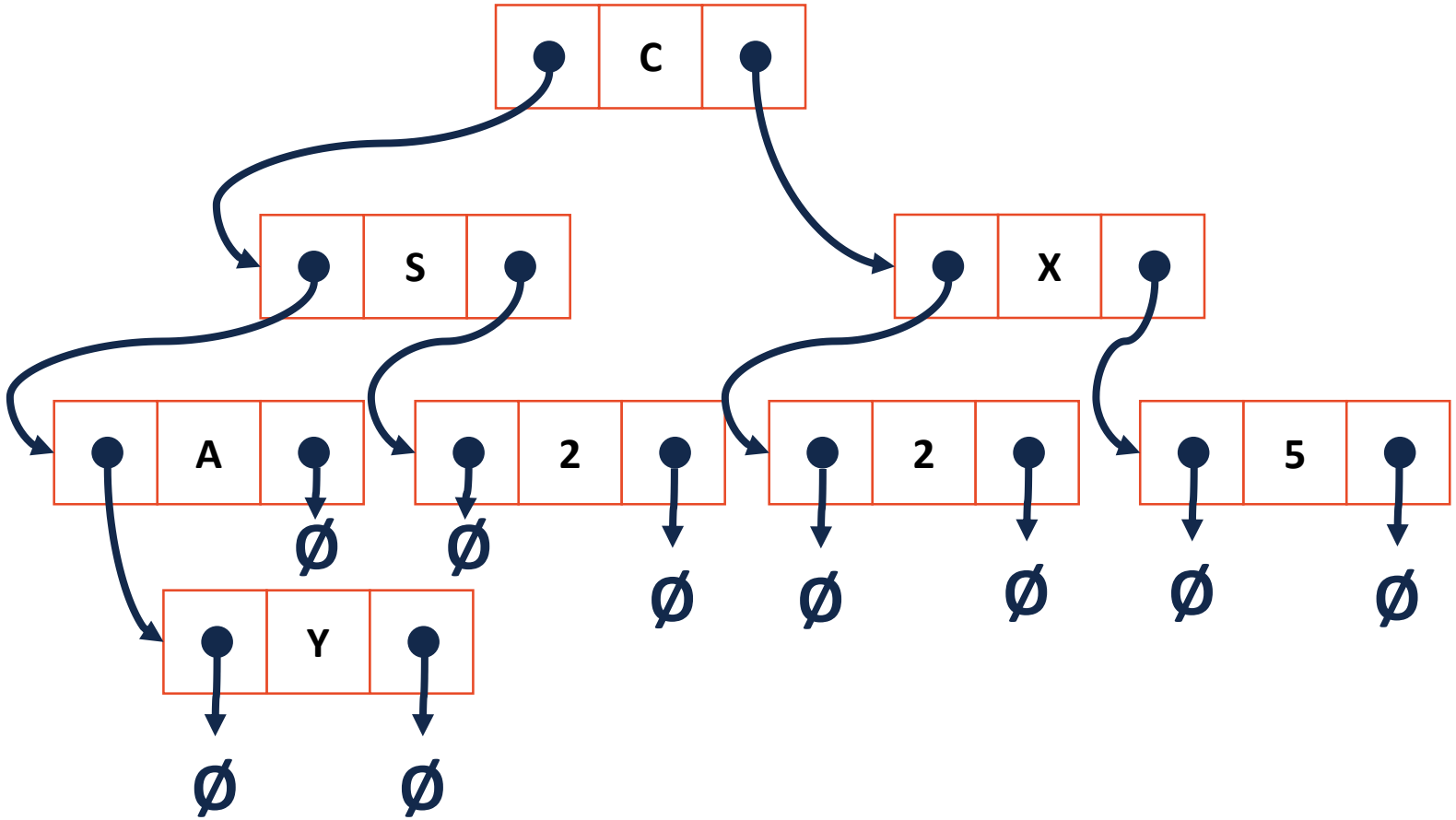
**remove**, removes an element from the tree.

**traverse**,

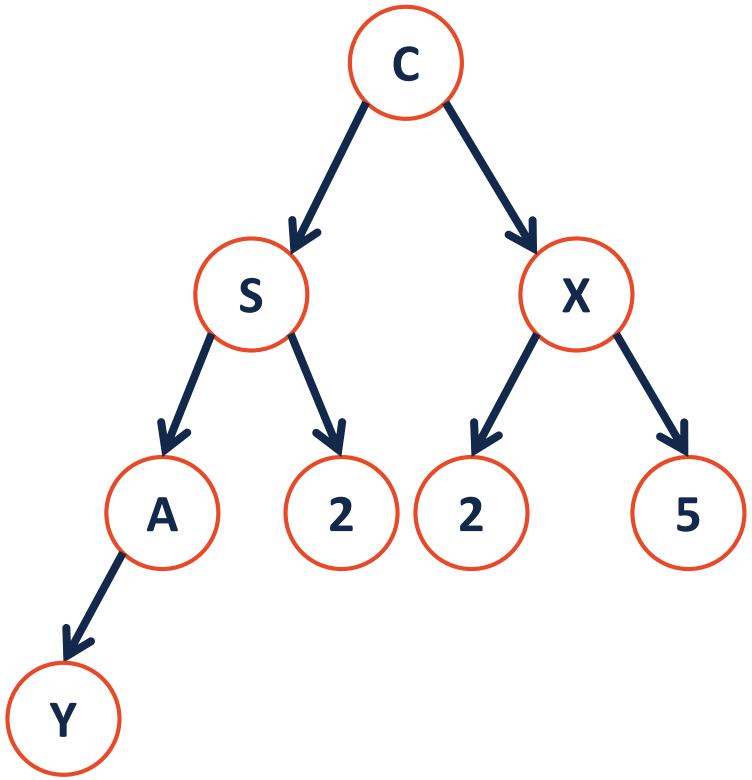
# BinaryTree.h

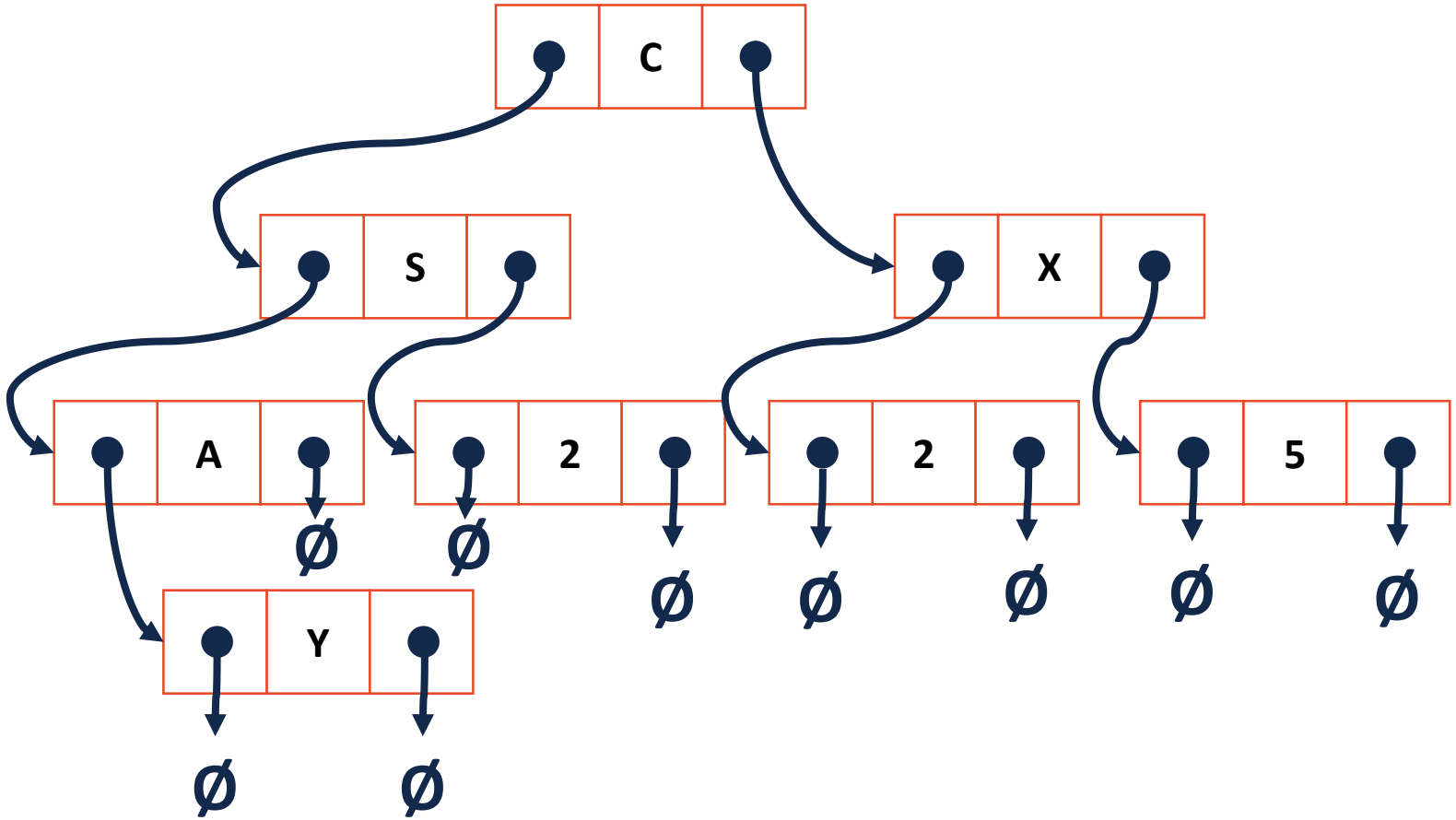
```
1  #pragma once
2
3  template <class T>
4  class BinaryTree {
5    public:
6      /* ... */
7
8    private:
9
10
11
12
13
14
15
16
17
18
19 };
```

# 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:**

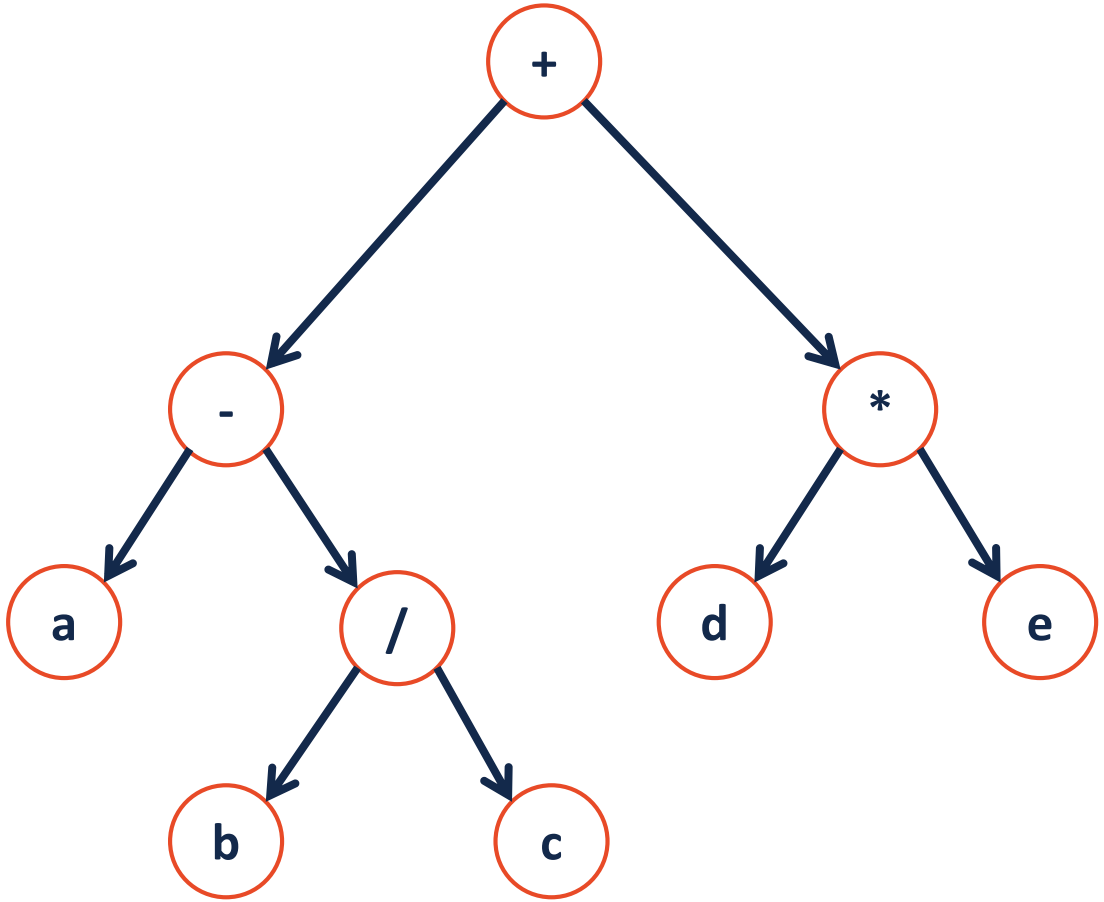
# How many NULLs?

**Induction Hypothesis:**

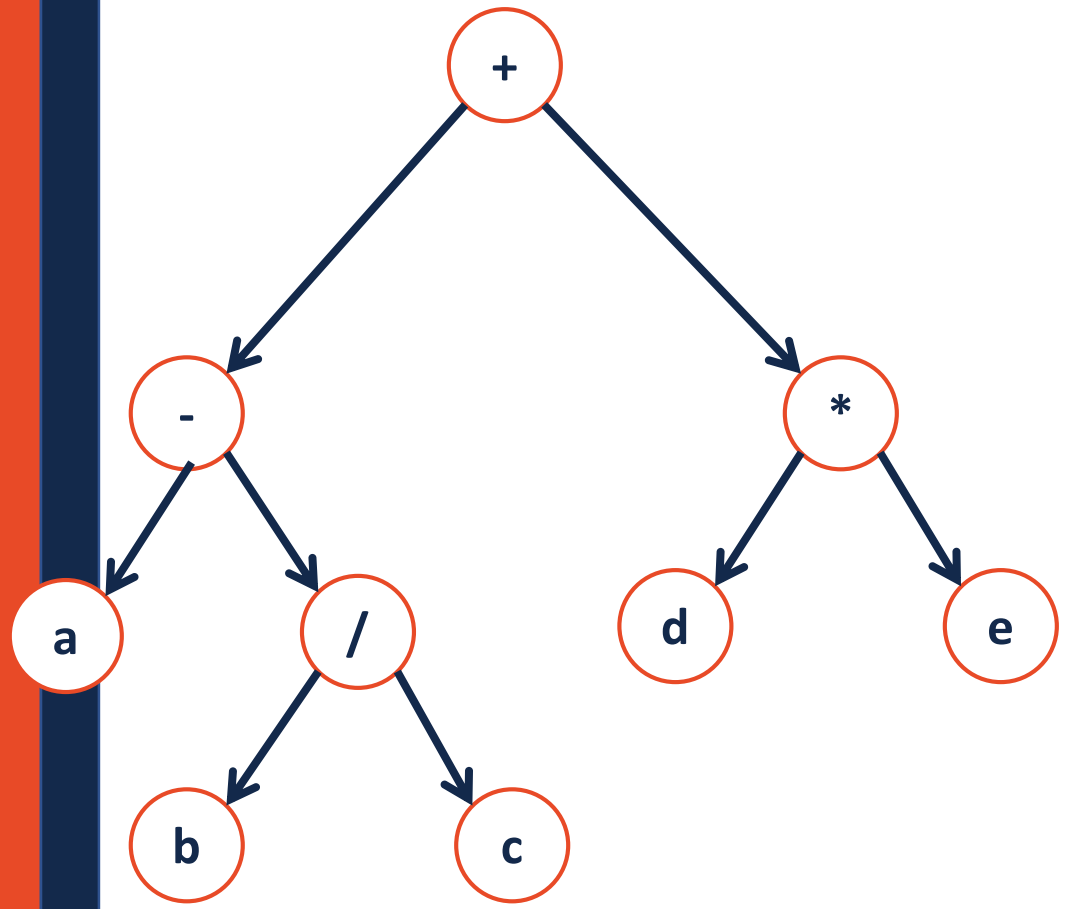
# How many NULLs?

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

# Traversals

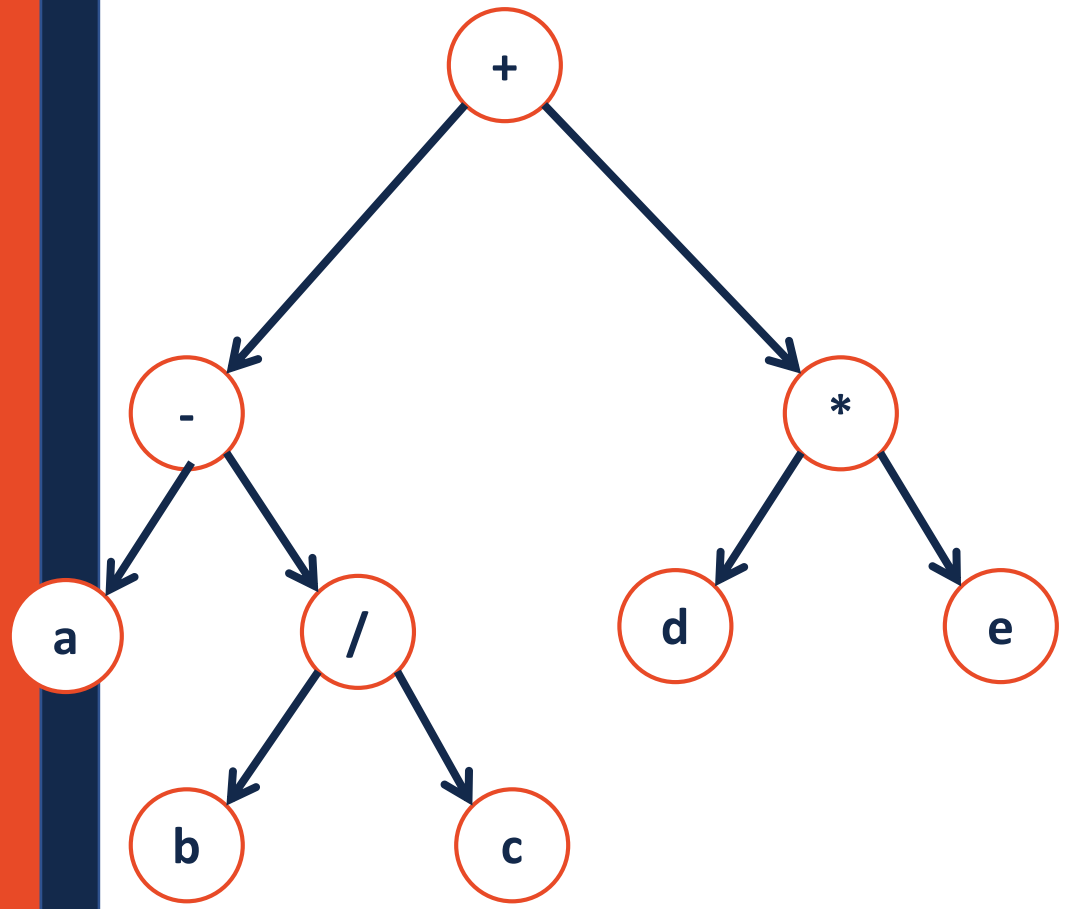


# Traversals



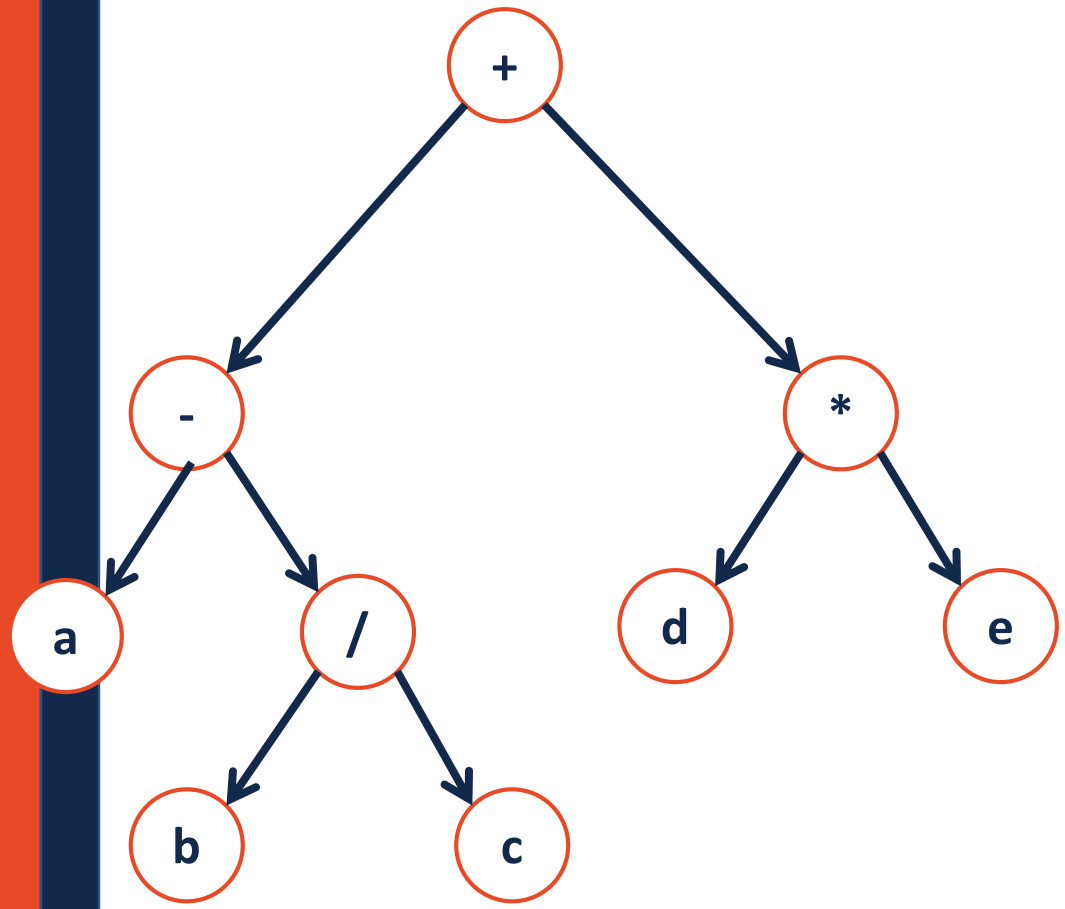

```
1  template<class T>
2  void BinaryTree<T>::__Order(TreeNode * root)
3  {
4     if (root != NULL) {
5
6        ____________________________;
7
8        ___Order(root->left);
9
10       ____________________________;
11
12       ___Order(root->right);
13
14       ____________________________;
15
16    }
17 }
```

# Traversals




```
1  template<class T>
2  void BinaryTree<T>::__Order(TreeNode * root)
3  {
4     if (root != NULL) {
5
6        ____________________________;
7
8        ___Order(root->left);
9
10       ____________________________;
11
12       ___Order(root->right);
13
14       ____________________________;
15
16    }
17 }
```

# Traversals




```
1  template<class T>
2  void BinaryTree<T>::__Order(TreeNode * root)
3  {
4     if (root != NULL) {
5
6        ____________________________;
7
8        ___Order(root->left);
9
10       ____________________________;
11
12       ___Order(root->right);
13
14       ____________________________;
15
16    }
17 }
```