Data Structures
Trees

CS 225
Brad Solomon

July 12, 2022

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

Learning Objectives

Review fundamental tree terminology

Introduce the concept and properties of a binary tree

Conceptualize and code tree traversals

Introduce fundamental tree search strategies

Trees

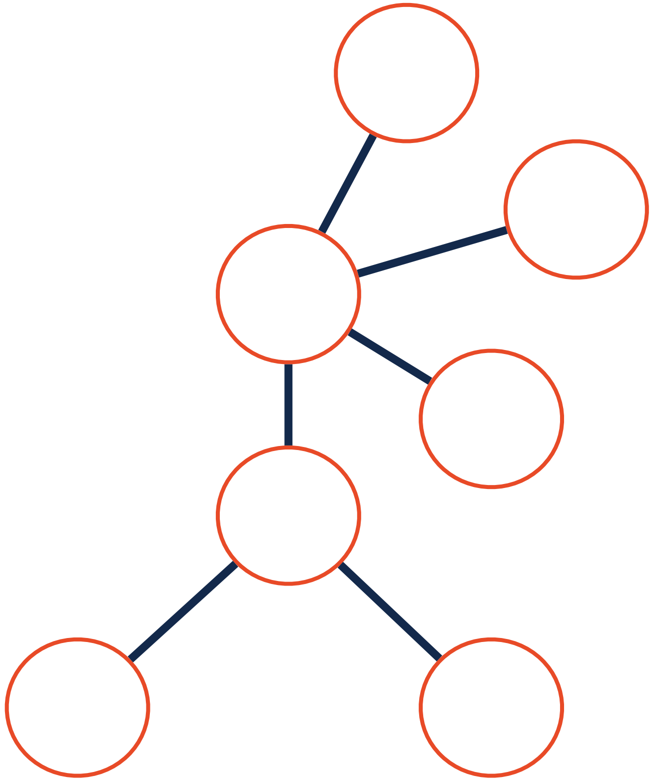
"The most important non-linear data structure in computer science."

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

A tree is:

-

-



Tree Terminology Review

Find an **edge** that is not on the longest **path** in the tree.

Which vertex is the **root** of the tree?

How many parents does each vertex have?

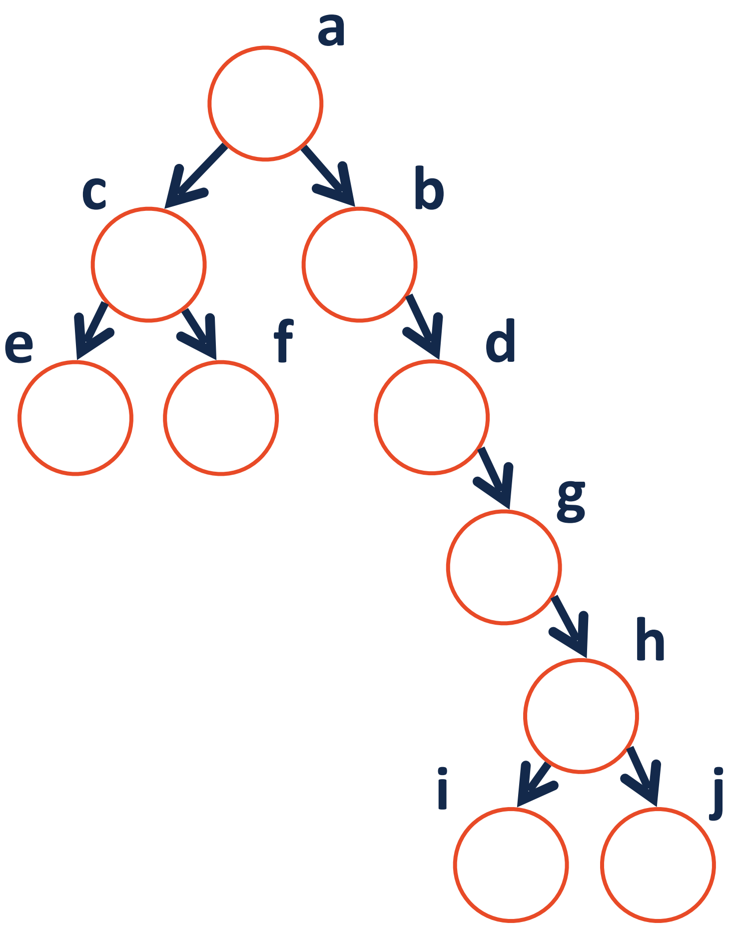
Which vertex has the fewest **children**?

Which vertex has the most **ancestors**?

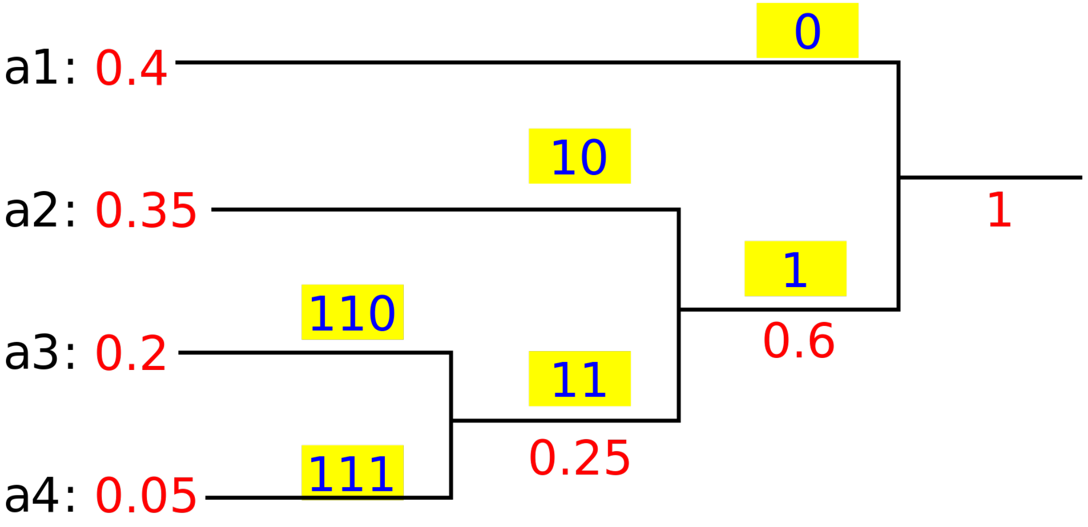
Which vertex has the most **descendants**?

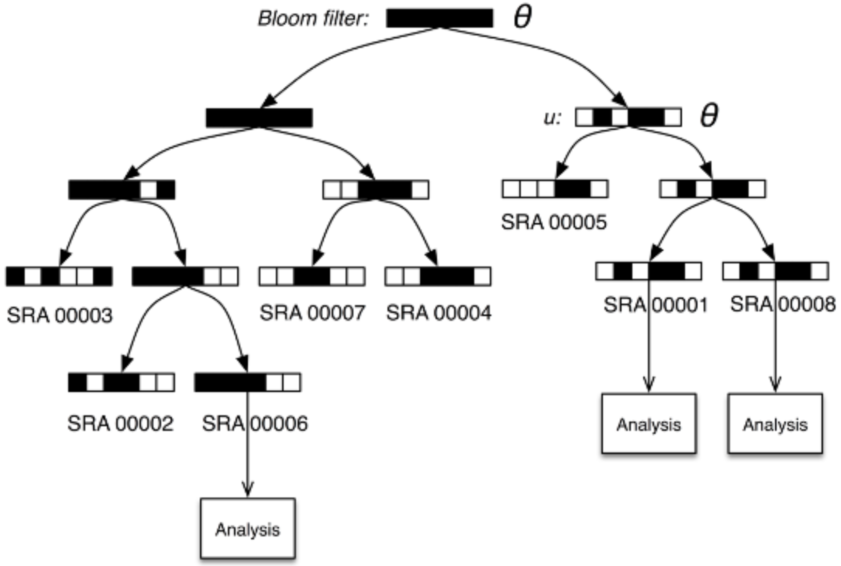
List all vertices in b's left **subtree**? In a's?

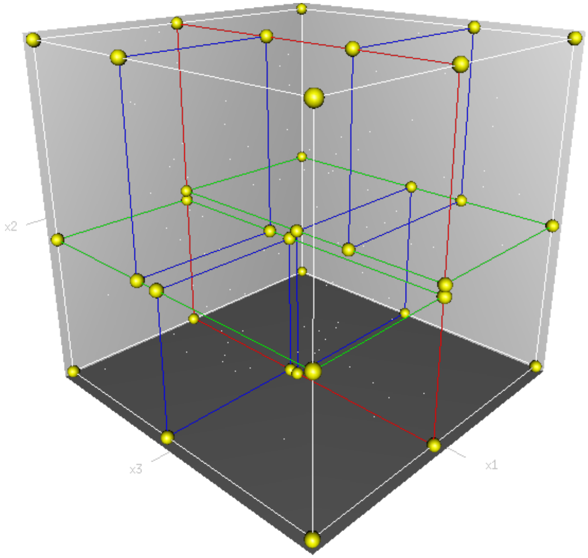
List all **leaves** in the tree.

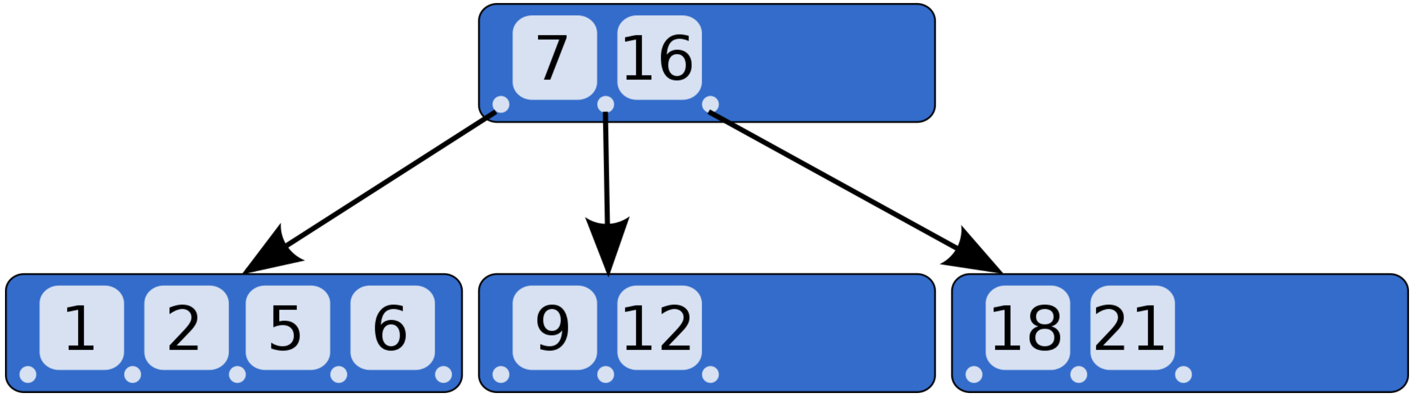


There are many *types* of trees









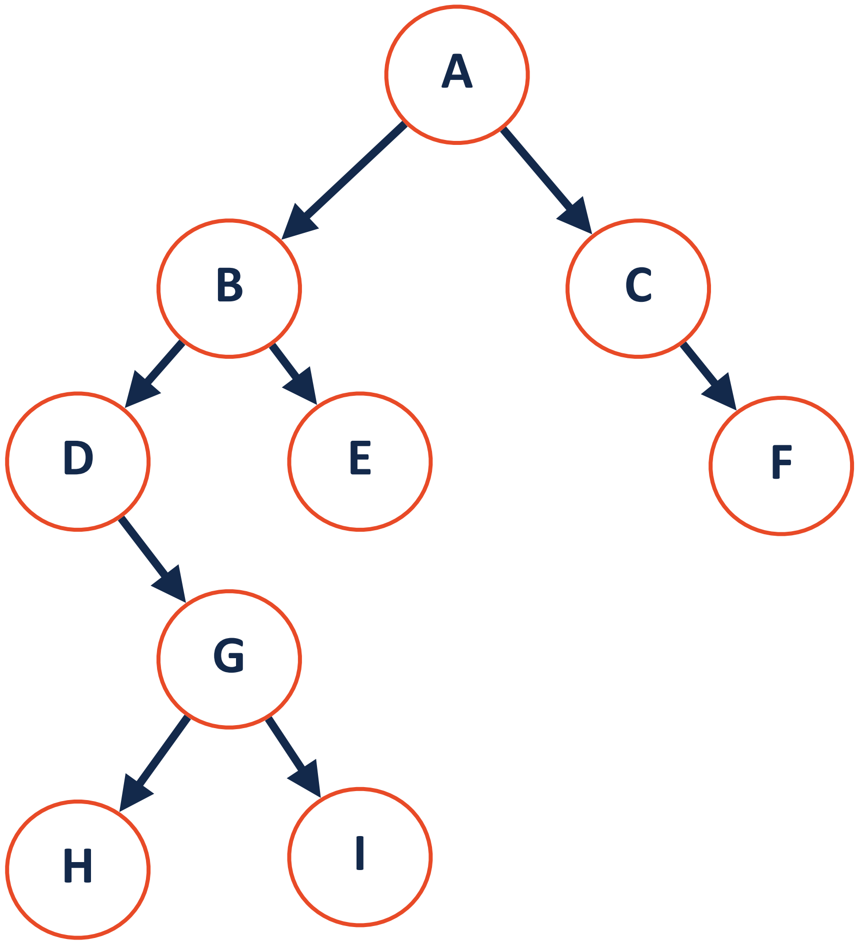
Binary Tree

A **binary tree** T is either:

-

OR

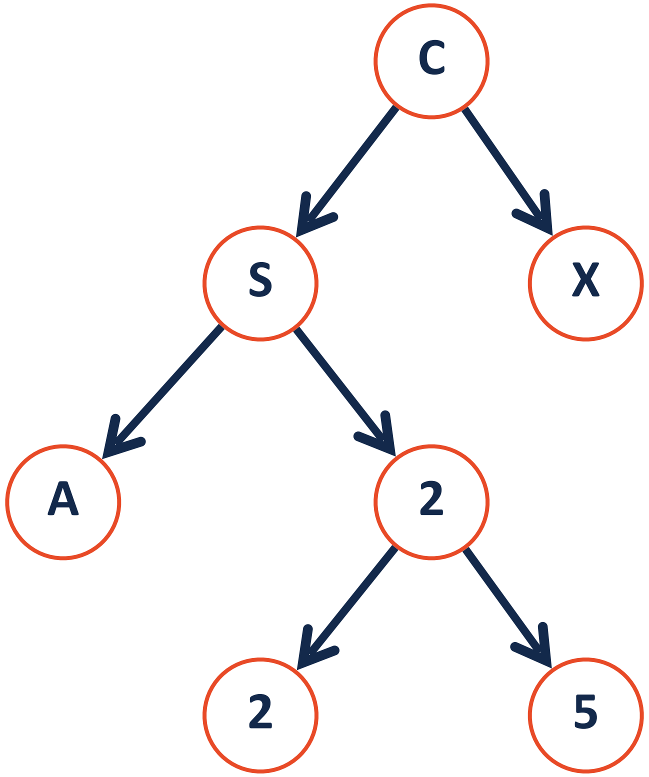
-



Tree Property: height

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

Given an arbitrary binary tree T, write a recursive equation for height:

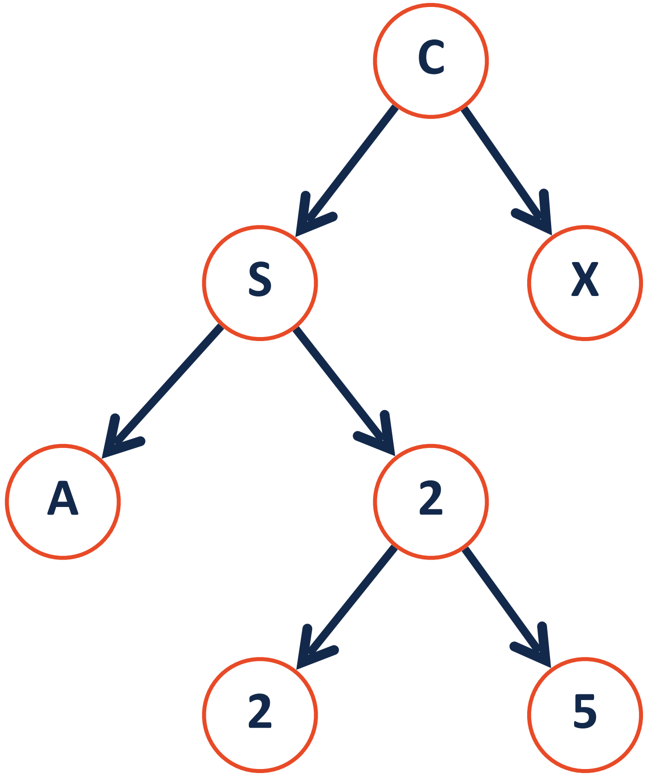


Tree Property: full

A tree **F** is **full** iff one of two things is true:

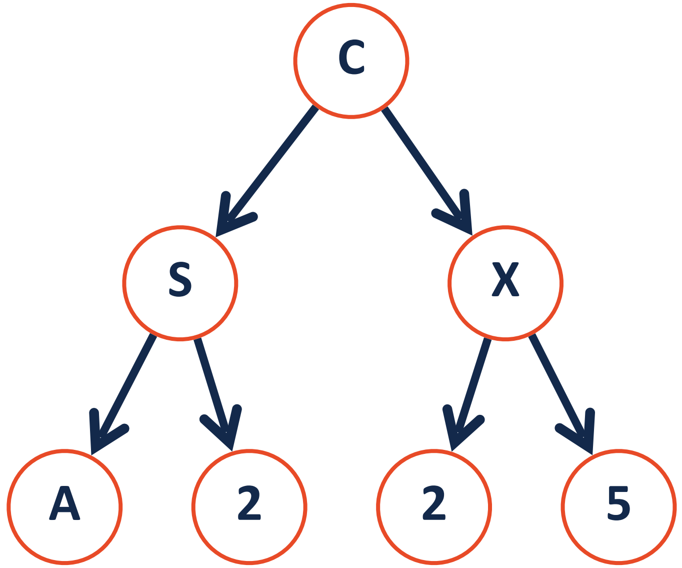
1.

2.



Tree Property: perfect

A tree **P** of height **h** is **perfect** iff one of two things is true:



1.

2.

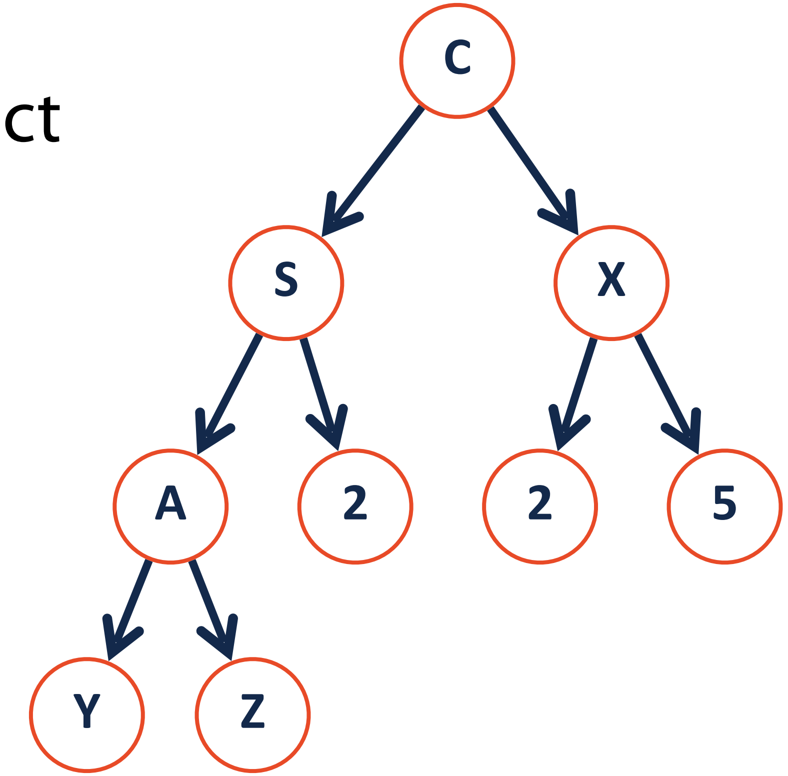
Tree Property: complete

A tree **P** of height **h** is **complete** if:

1. For every level except the last the tree is perfect

2. The last level is 'pushed to the left'

How many nodes are at level k in a complete tree?



Tree Property: complete

A **complete** tree **C** of height **h, C_h**:

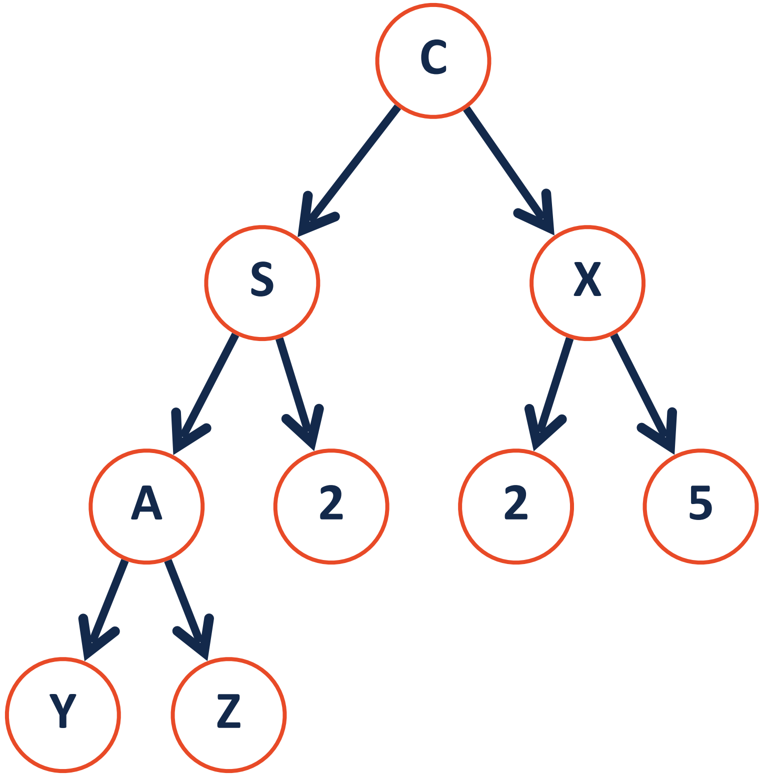
1. $\mathbf{C_{-1}} = \{\}$

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

T_L is ____________ and T_R is ____________

OR

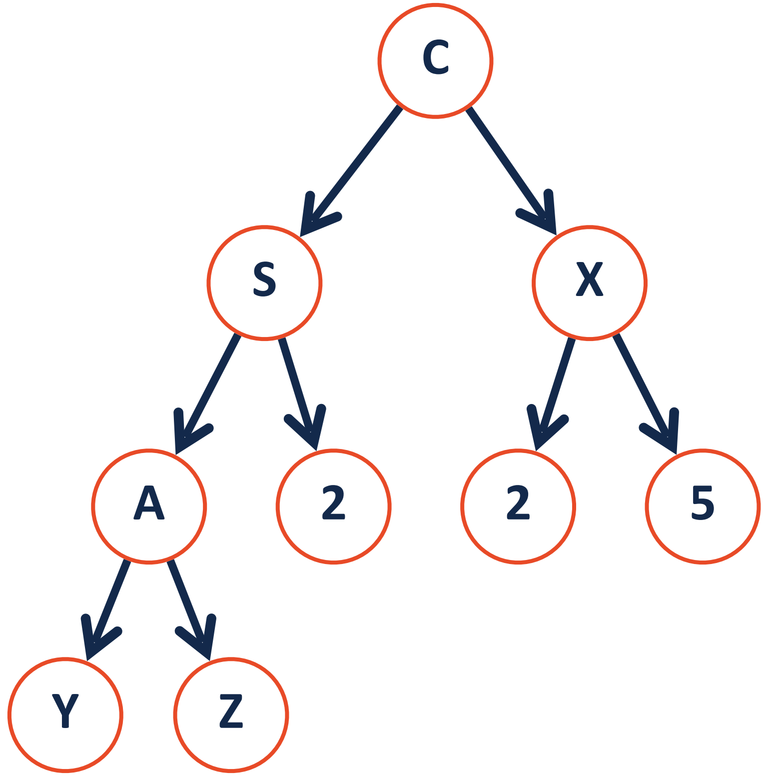
T_L is ____________ and T_R is ____________



Tree Property: complete

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

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



Tree ADT

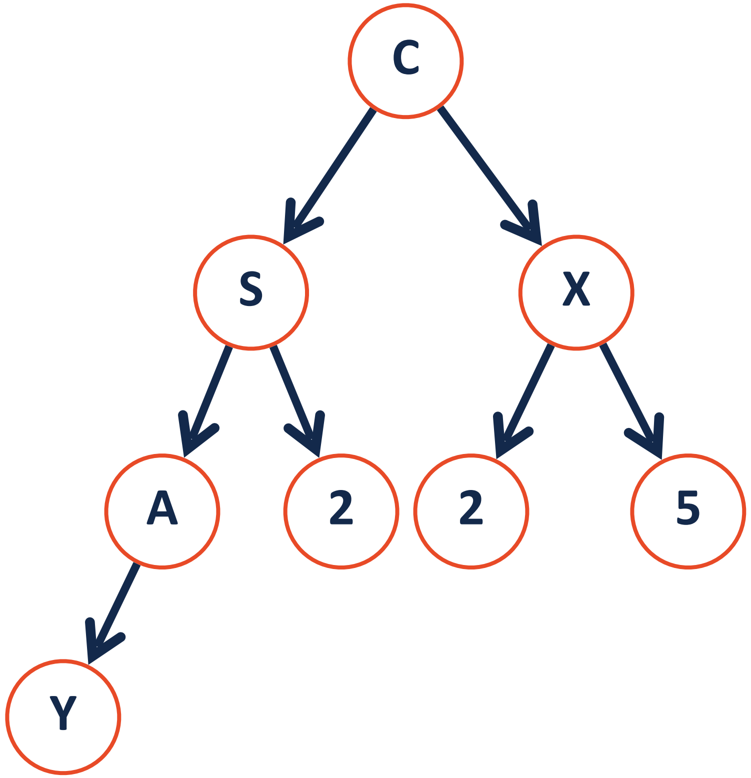
BinaryTree.h

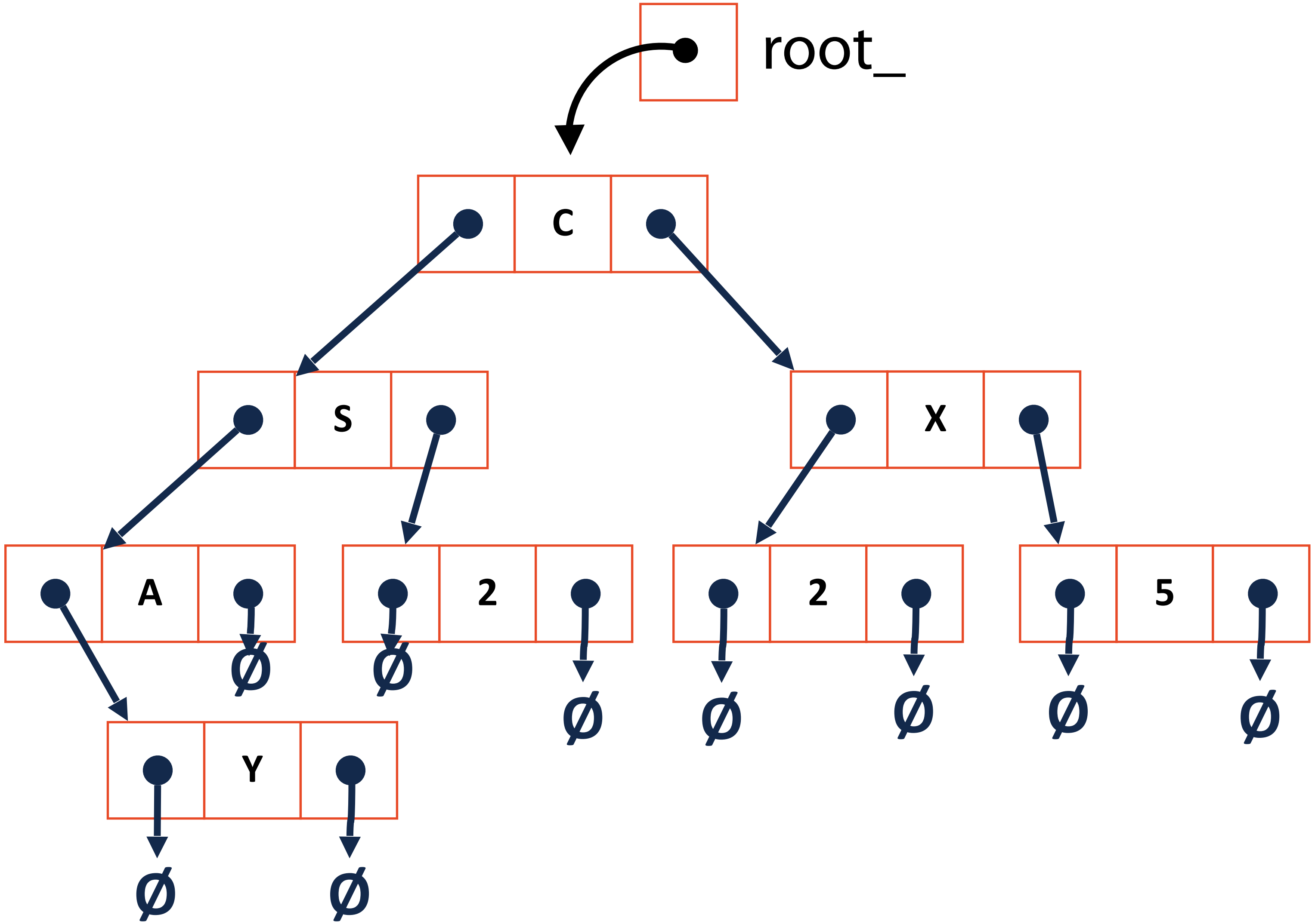
```
#pragma once

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

  private:














};
```

BinaryTree.h

```
#pragma once

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

  private:

    struct TreeNode {
      T data;
      TreeNode *left;
      TreeNode *right;
    }

    TreeNode *root_;






};
```

Trees aren't new:





“Wasted Overhead” in Binary Tree

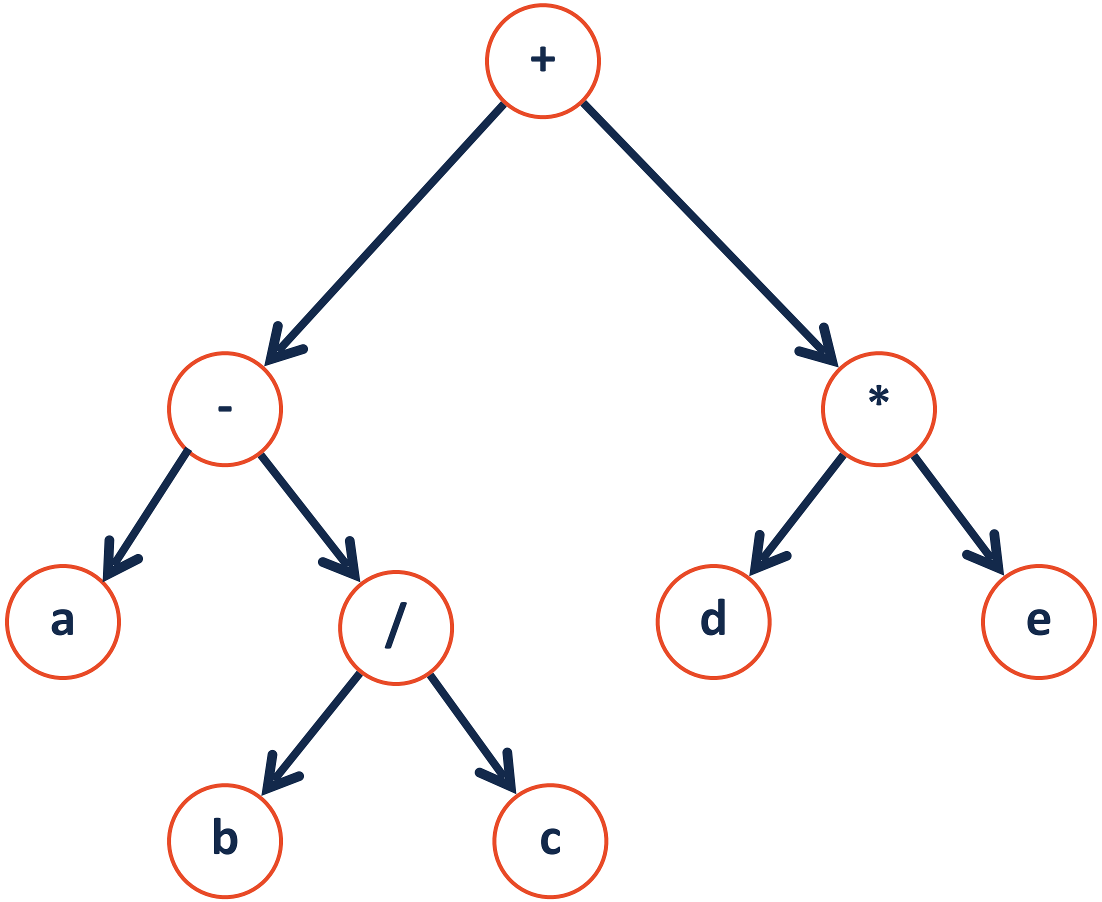
Theorem: If there are **n** objects in our representation of a binary tree, then there are _________ NULL pointers.

"Wasted Overhead" in Binary Tree

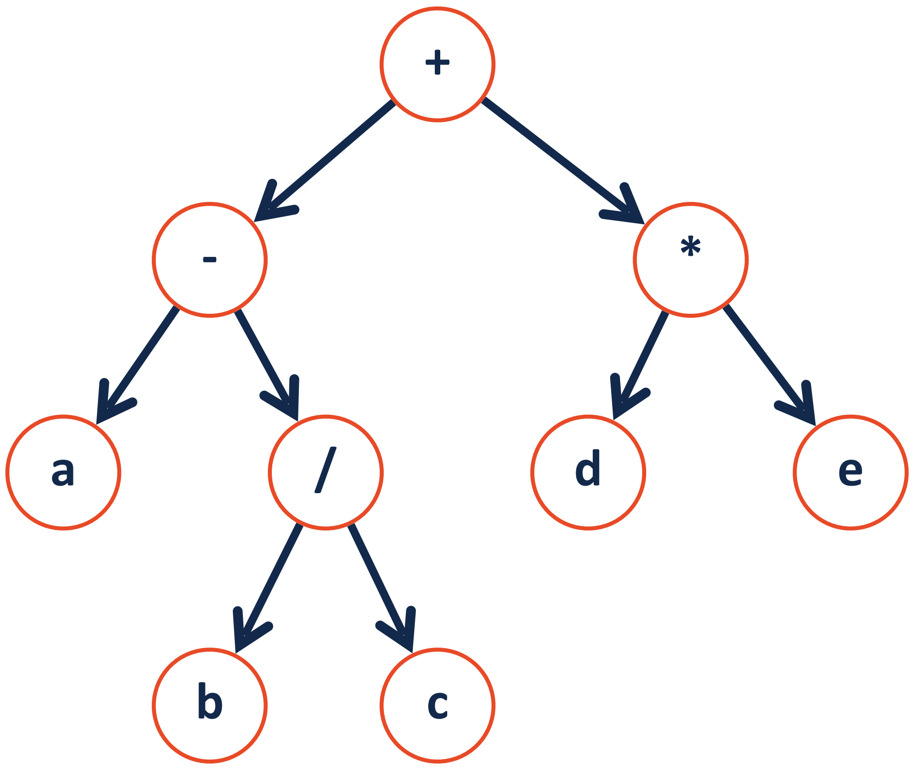
Theorem: If there are **n** objects in our representation of a binary tree, then there are **n+1** NULL pointers.

Induction Step:

Traversal

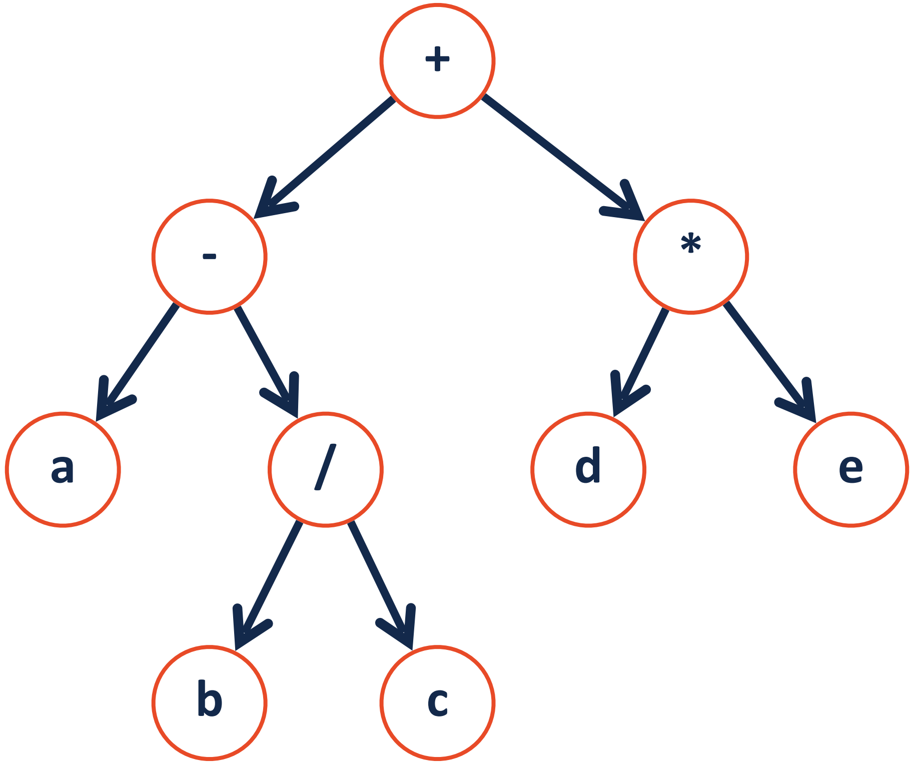


Traversals



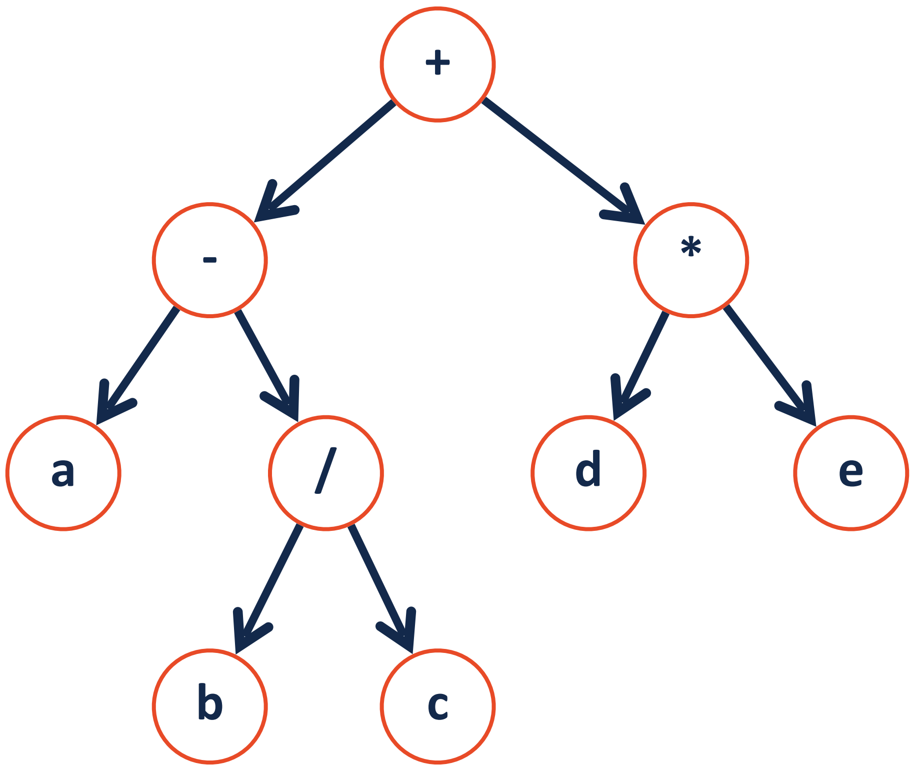

```
template<class T>
void BinaryTree<T>::__Order(TreeNode * root)
{



















}
```

Traversals



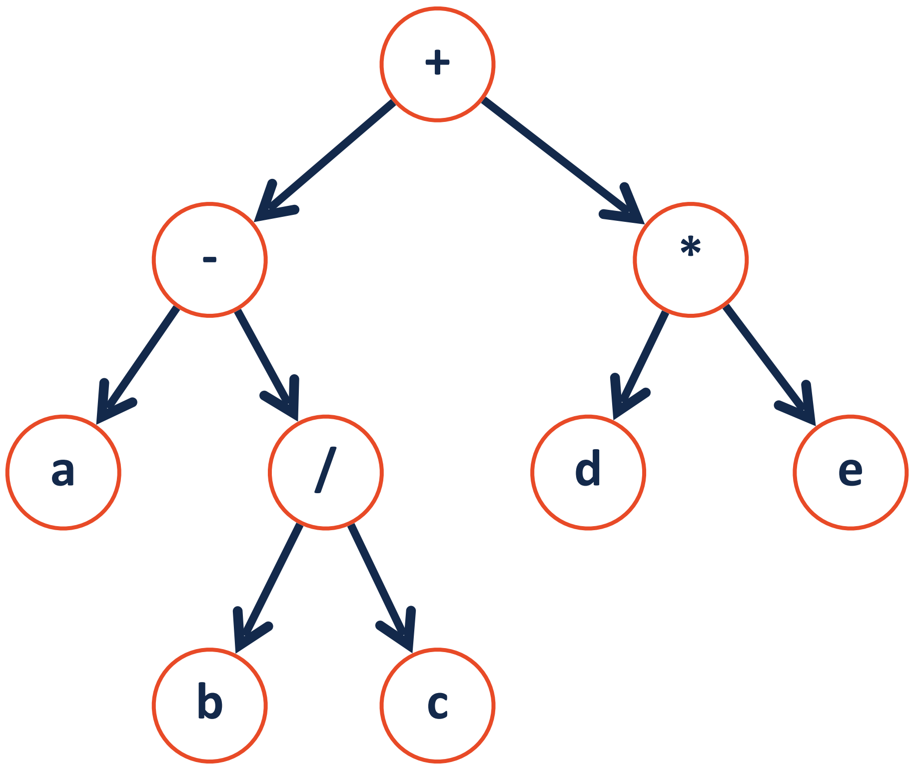

```
template<class T>
void BinaryTree<T>::__Order(TreeNode * root)
{
    if (root != NULL) {

        ______________________;

        ___Order(root->left);

        ______________________;

        ___Order(root->right);

        ______________________;

    }
}
```

Traversals

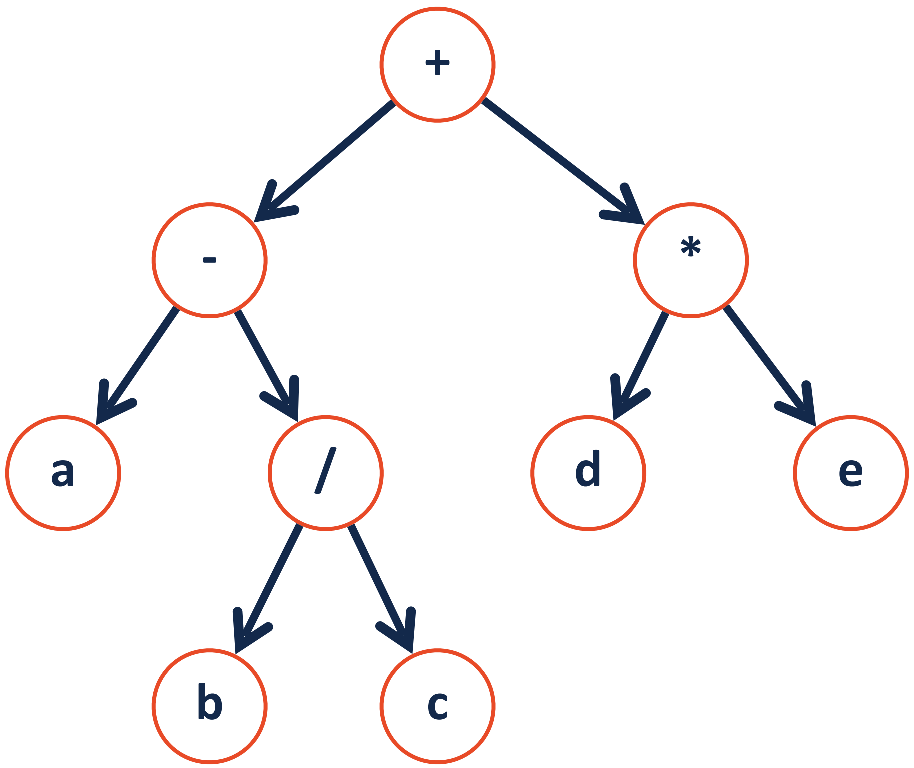



```
template<class T>
void BinaryTree<T>::__Order(TreeNode * root)
{
    if (root != NULL) {

        ______________________________;

        ___Order(root->left);

        ______________________________;

        ___Order(root->right);

        ______________________________;

    }
}
```

A Different Type of Traversal

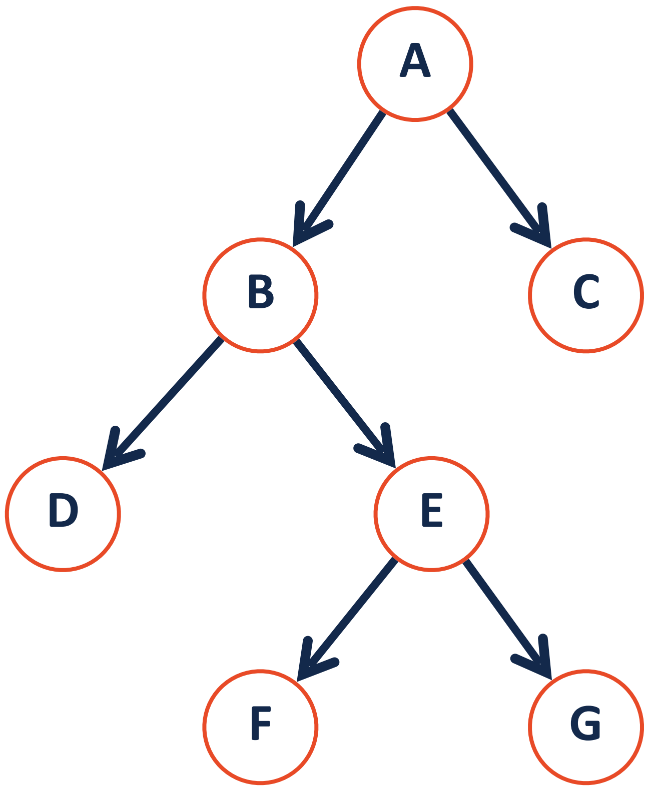


A Different Type of Traversal




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

    Queue<TreeNode*> q;
    q.enqueue(root);

    while( q.empty() == False){

        TreeNode* temp = q.head();
        process(temp);

        q.dequeue();

        q.enqueue(temp->left);
        q.enqueue(temp->right);
```

Traversal vs Search



Traversal

Search