TODO BEFORE

To read Before #ToBeReady

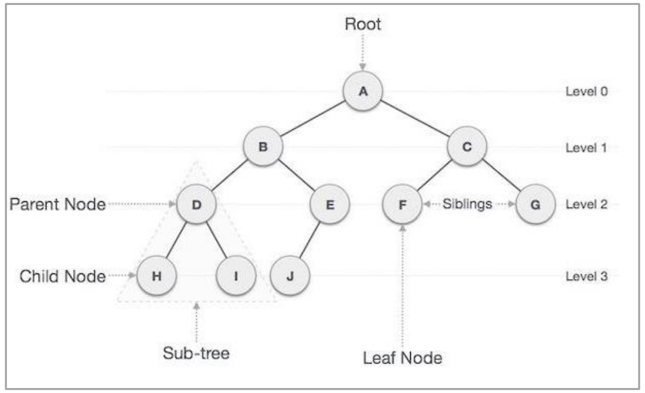
https://github.com/aish21/Algorithms-and-Data-Structures

Great doc covering
All ADTS

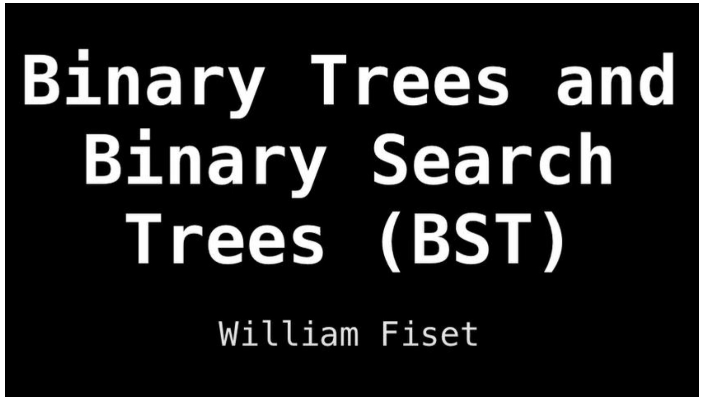
✓ Introduction to Trees

Introduction to
Tree
Data Structure

✓ Different types of trees



✓ Binary Trees

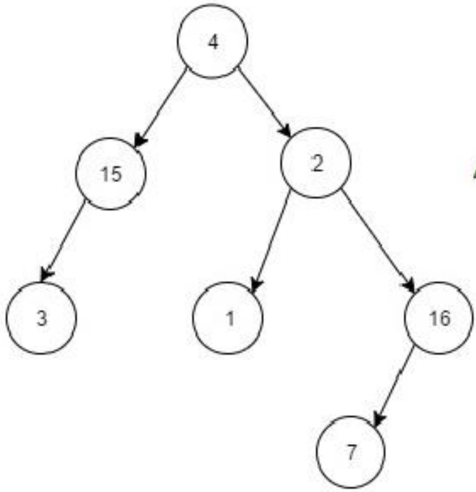


Binary Trees and
Binary Search
Trees (BST)

William Fiset

ADVANCED ALGORITHM

W8-S1 – Trees



CADT
IDT

Objectives for today

- Understand the concept of the Tree
- Explore Tree terminology
- Identify the type of the Tree
- Navigation in the Binary Tree

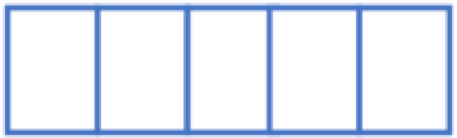
EXPLAIN

Abstract Data Structures

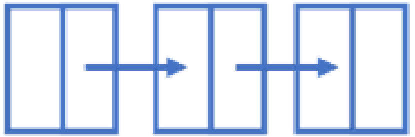
Linear

Data elements are arranged **sequentially**

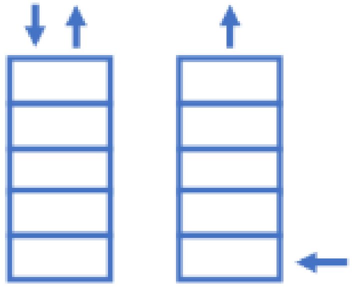
Array



Linked List



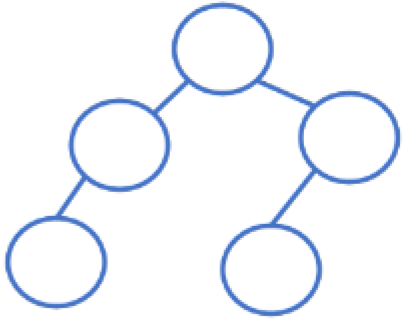
Stack & Queue



Non Linear

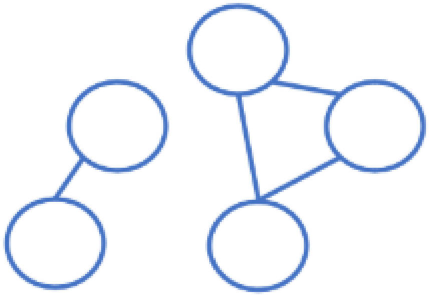
Data elements are **not** arranged sequentially

Tree



Every node can have 1 parent only

Graph



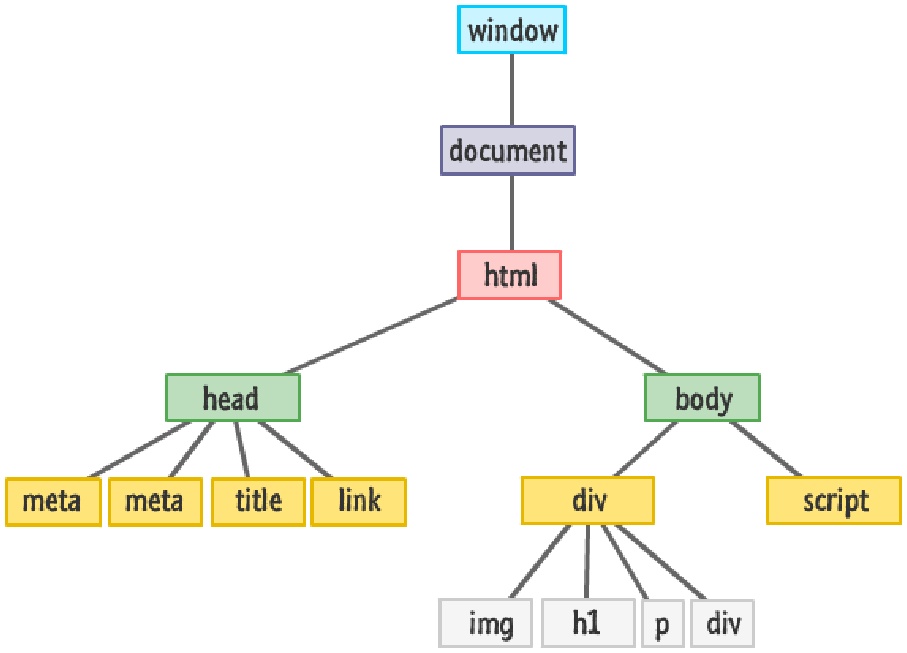
There is not rules for the connection of nodes

EXPLAIN

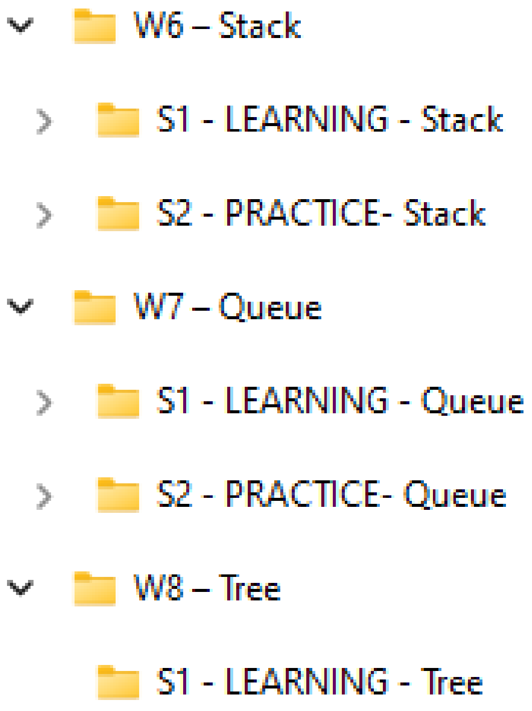
Why **trees** ?

A tree can store information that naturally forms a hierarchy.

HTML document (DOM)



Files in an Operating System



- W6 – Stack
  - S1 - LEARNING - Stack
  - S2 - PRACTICE- Stack
- W7 – Queue
  - S1 - LEARNING - Queue
  - S2 - PRACTICE- Queue
- W8 – Tree
  - S1 - LEARNING - Tree

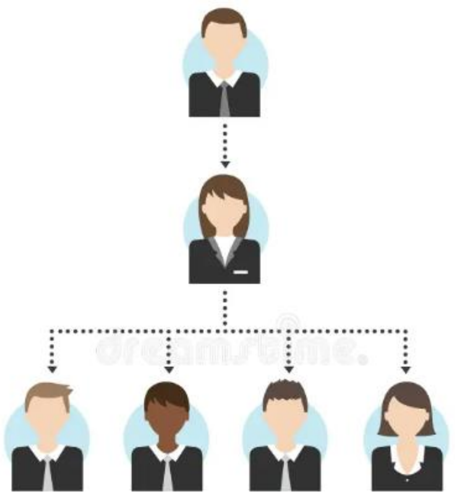
EXPLAIN

Why **trees** ?

A tree can store information that naturally forms a hierarchy.

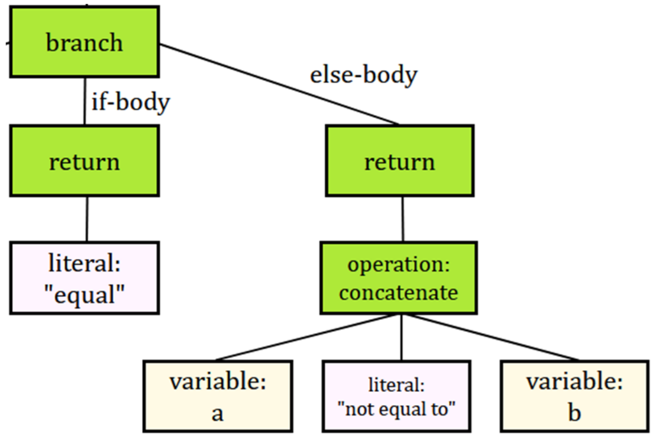
Organization Trees

Nodes represent
employees or sub-departments.



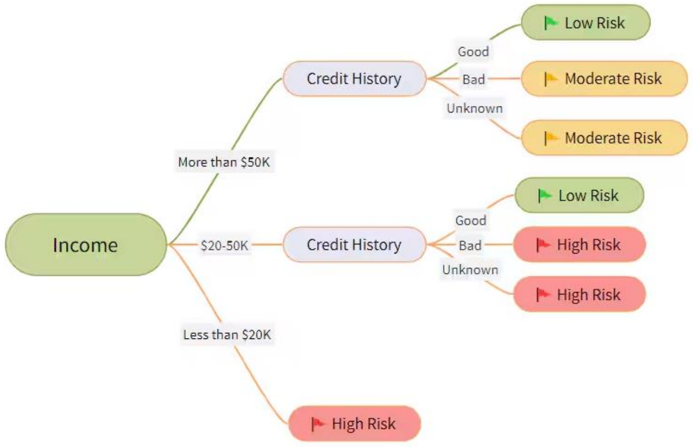
Compiler Syntax Trees

Nodes represent
programming language constructs
(e.g., expressions, statements).



Decision-Making Trees

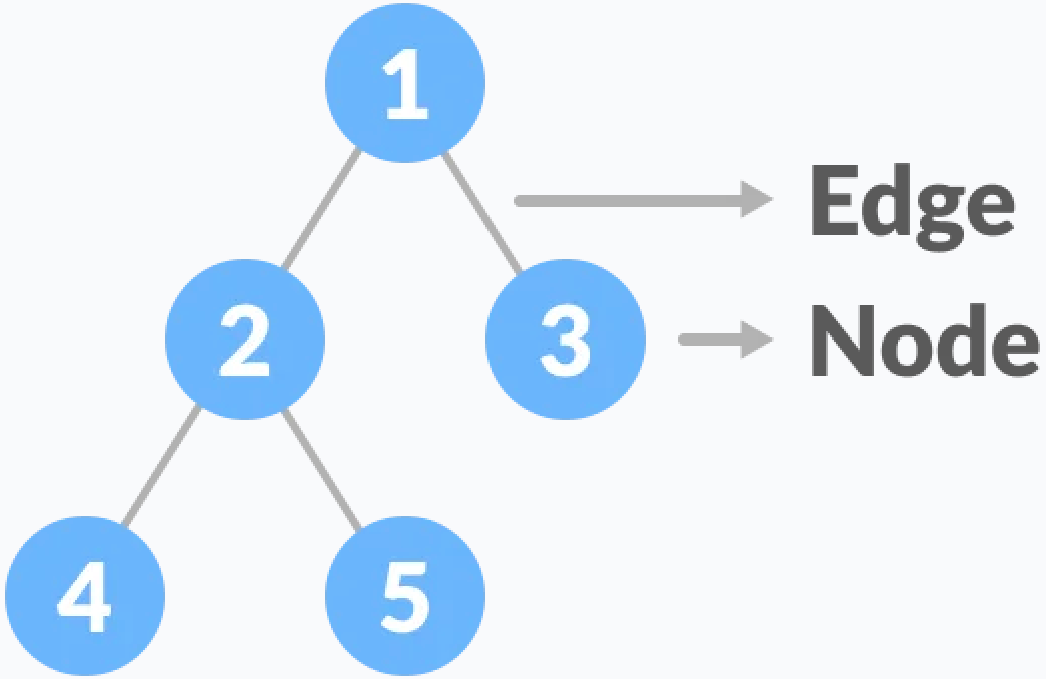
Nodes represent
a condition or decision point.



EXPLAIN

Tree **Definition**

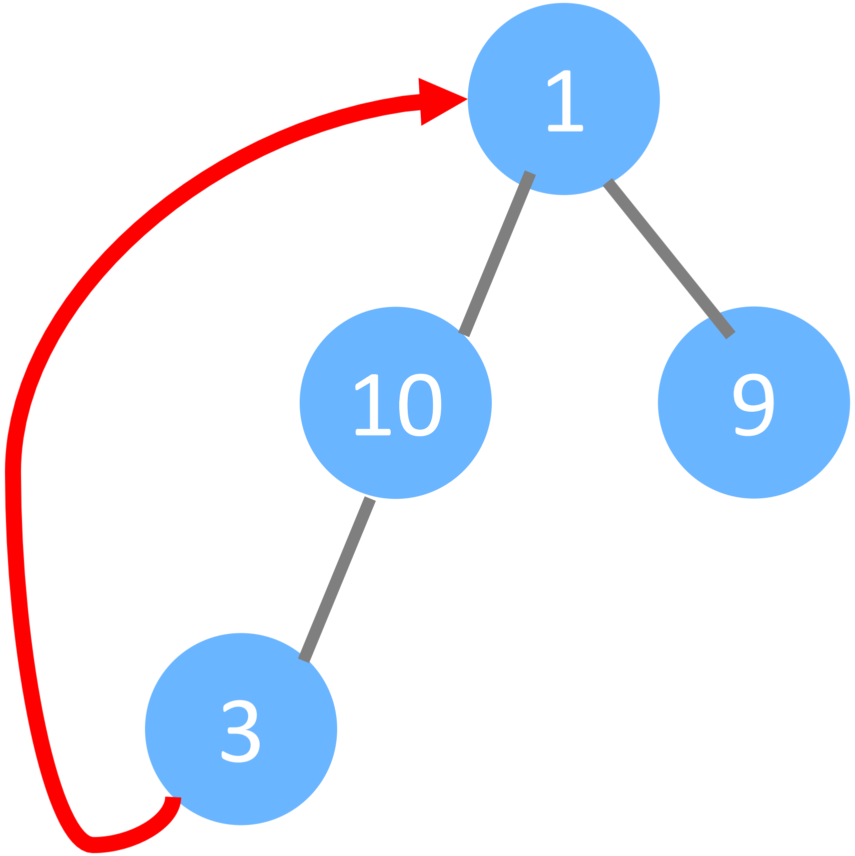
A tree is a **non-linear hierarchical** data structure that consists of nodes connected via edges.



EXPLAIN

Tree **Definition**

A node **cannot** be both the child and the parent of another node

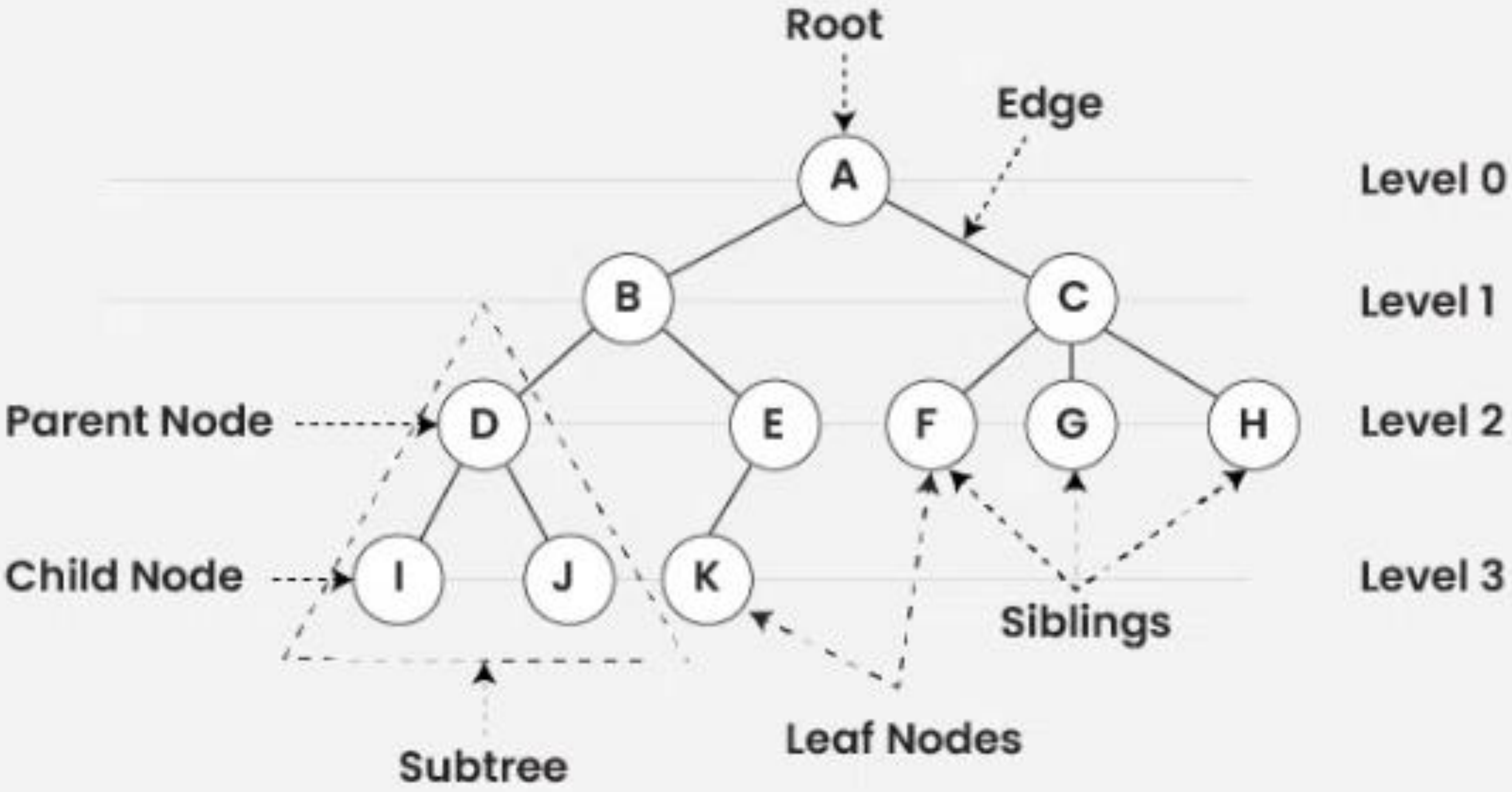


In this situation, we would need a graph data structure

EXPLAIN

Tree **Terminology**

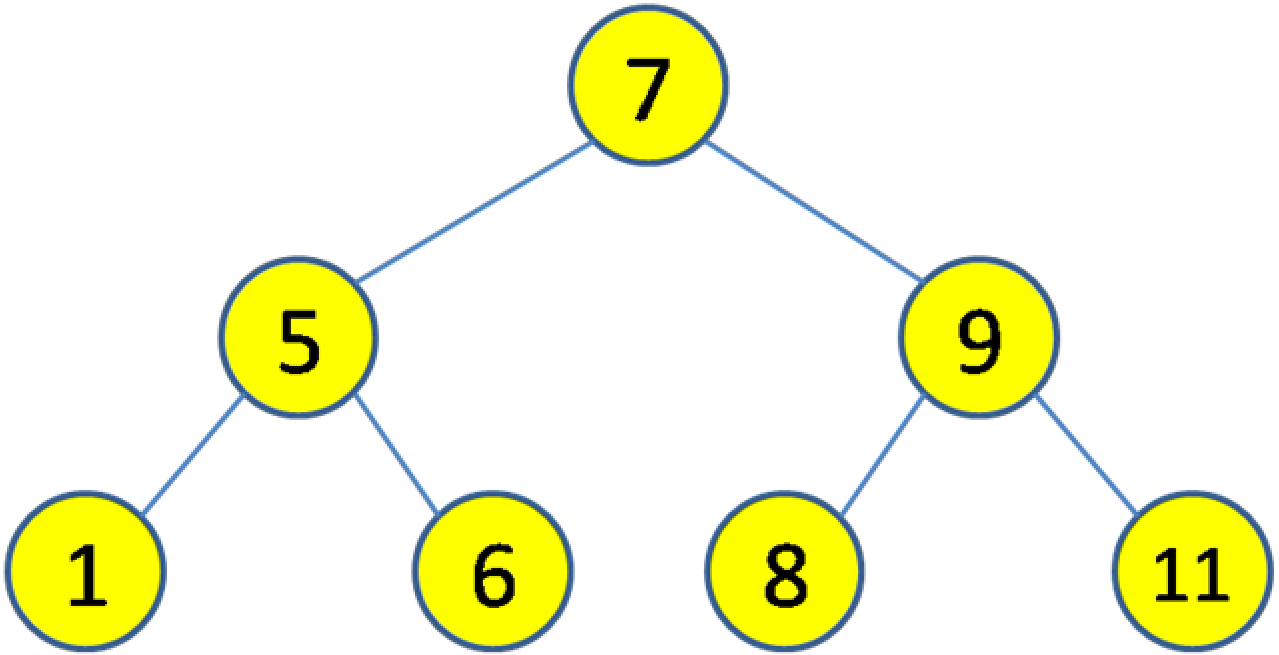
Parent Node, Child Node, Root Node, Leaf Node, Sibling, Edge, Level



QUIZ

Tree **Terminology**

Look at this tree and answer the questions



Q1 - What is the root node?

Q2 - What is the parent of node 1 ?

Q3 - What are the sibling of 5?

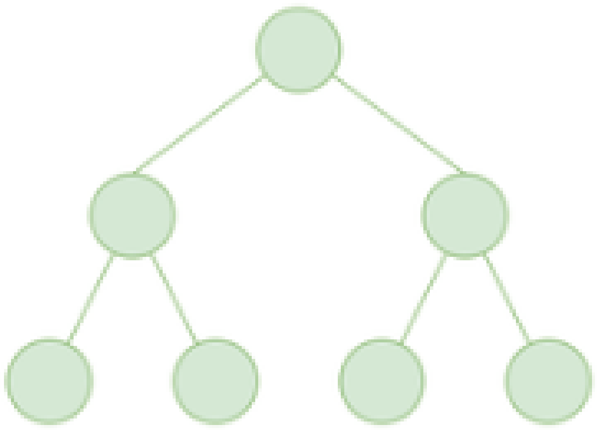
Q4 - What are the leaves nodes ?

Q5 – What is the **level** of node 8?

EXPLAIN

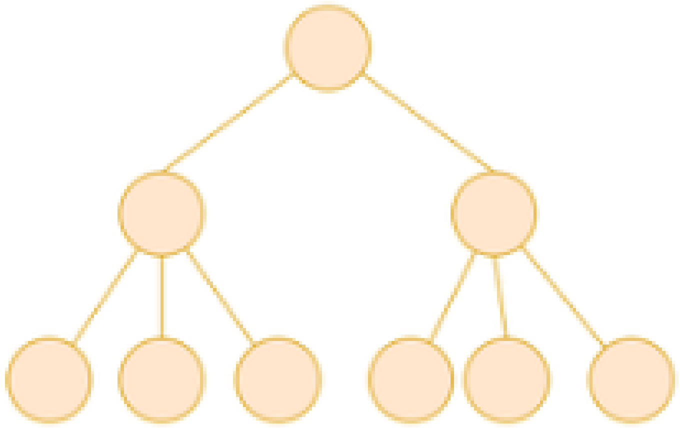
Types of Trees

MORE INFO HERE



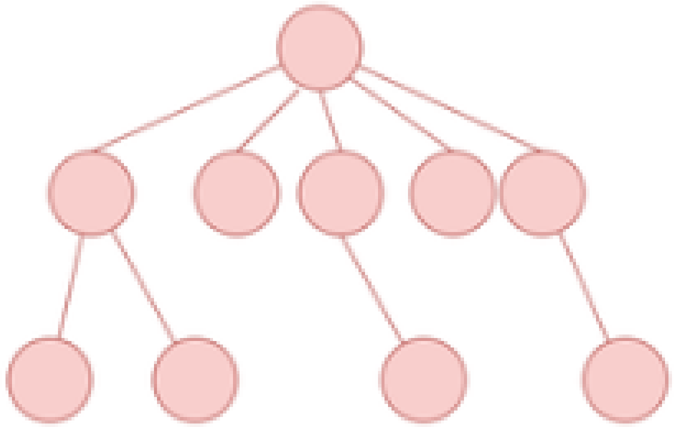
Binary Tree

Each node can have only **2** children
We call them **left** and **right**



Ternary Tree

Each node has **3** children: **left child**, **middle child**, and **right child**.



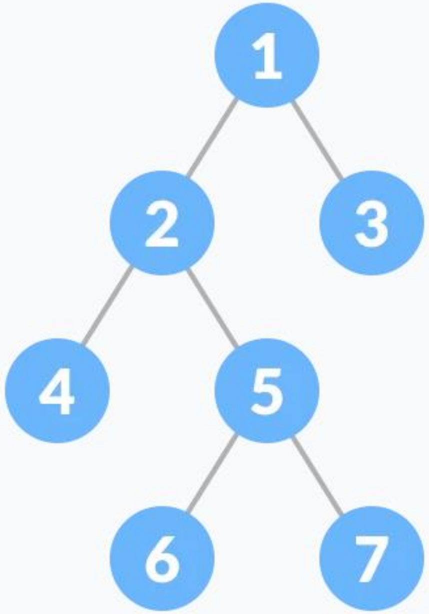
N-Ary Tree

Each node can have a list of children

EXPLAIN

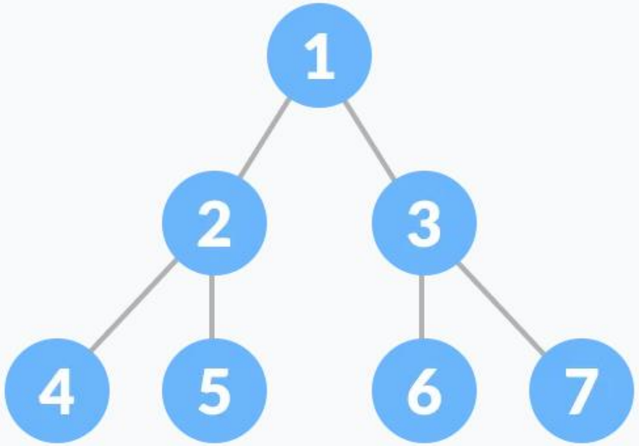
Some specific **Binary** Tree Types

MORE INFO HERE



Full Binary Tree

Every node has either
2 or no children nodes.

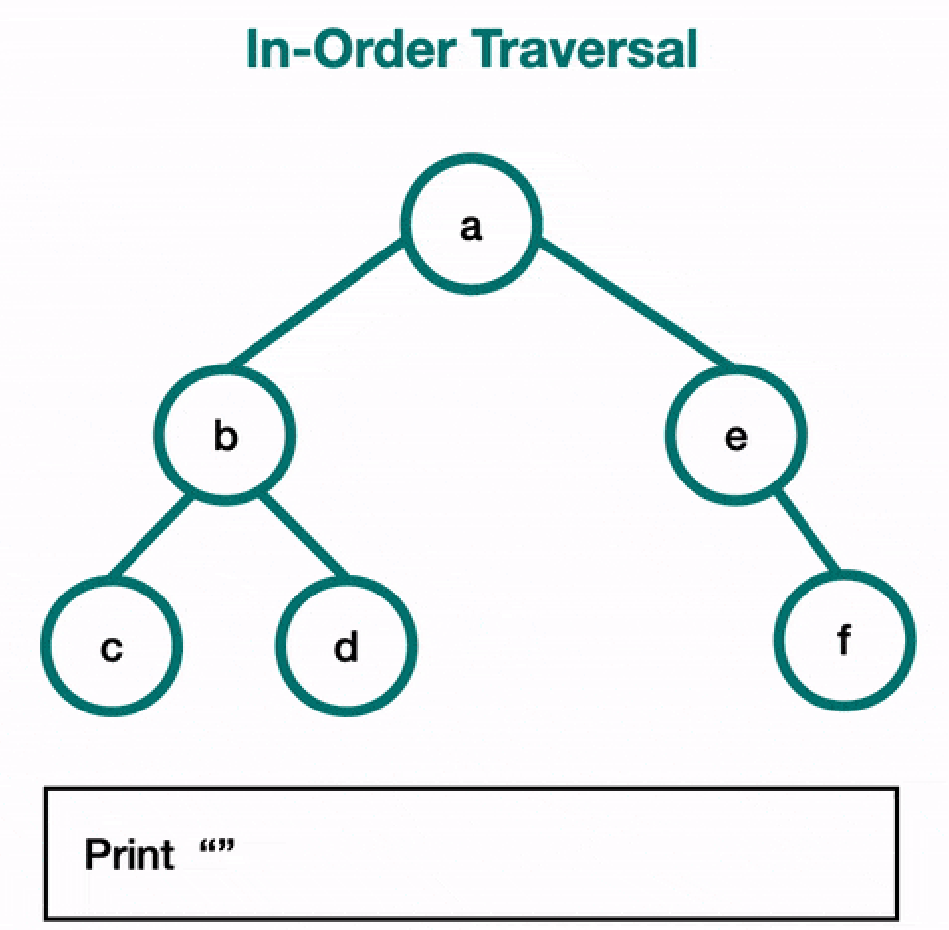


Perfect Binary Tree

Every internal node has **exactly** 2 child nodes
All the leaf nodes are at the same level.

EXPLAIN

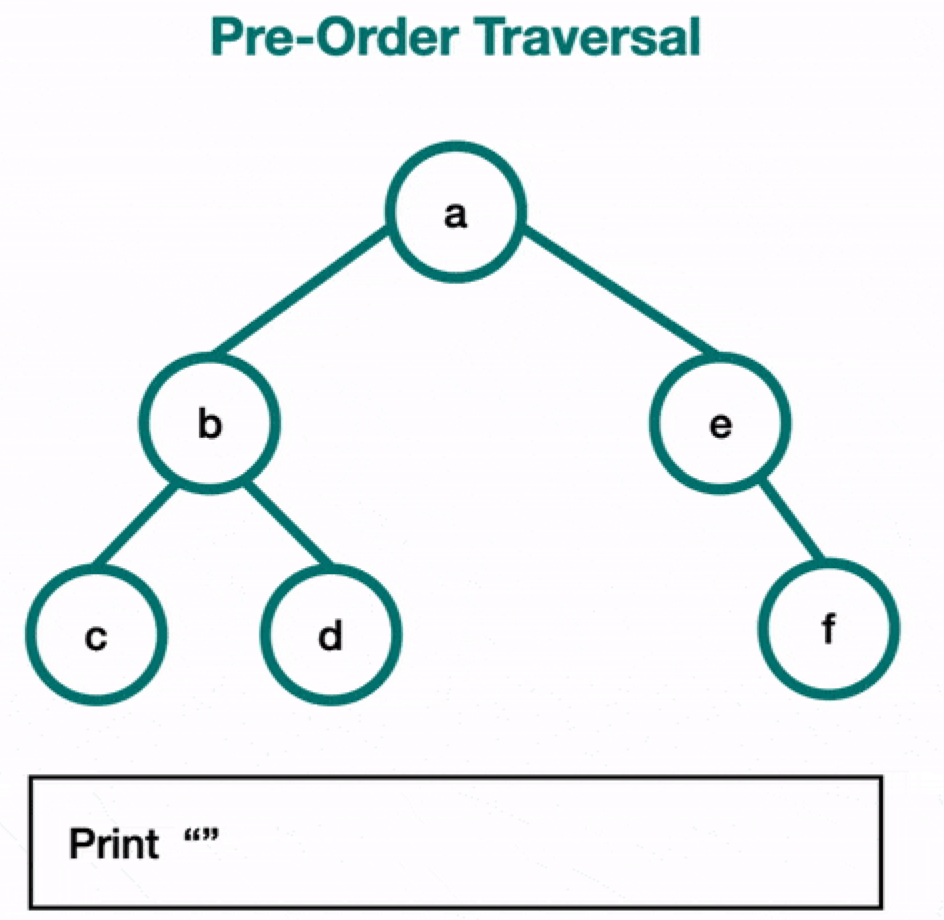
Binary Trees - Navigation



Visit all the nodes in the **left subtree** → Visit **root node** → Visit all the nodes in the **right subtree**

EXPLAIN

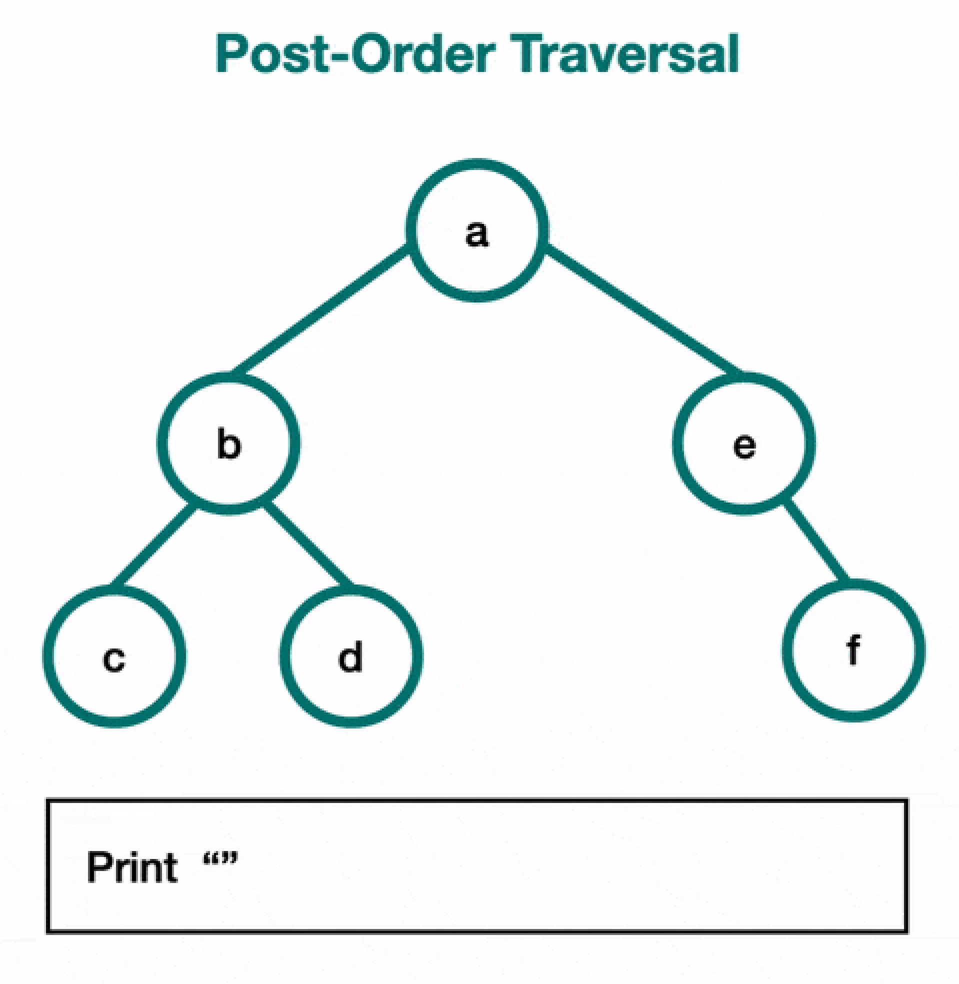
Binary Trees - Navigation



Visit **root node** → Visit all the nodes in the **left subtree** → Visit all the nodes in the **right subtree**

EXPLAIN

Binary Trees - Navigation

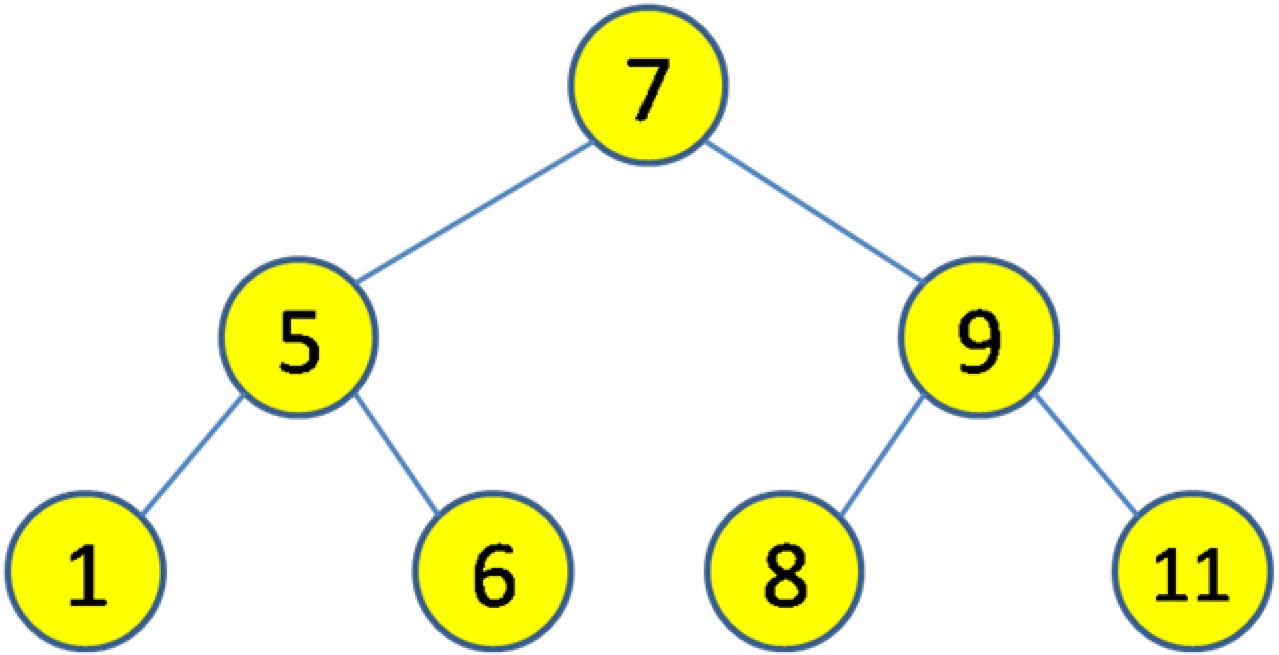


Visit all the nodes in the **left subtree** → Visit all the nodes in the **right subtree** → Visit **root node**

QUIZ

In-Order traversal

Write the list of nodes visited, in case of a **IN ORDER** transversal



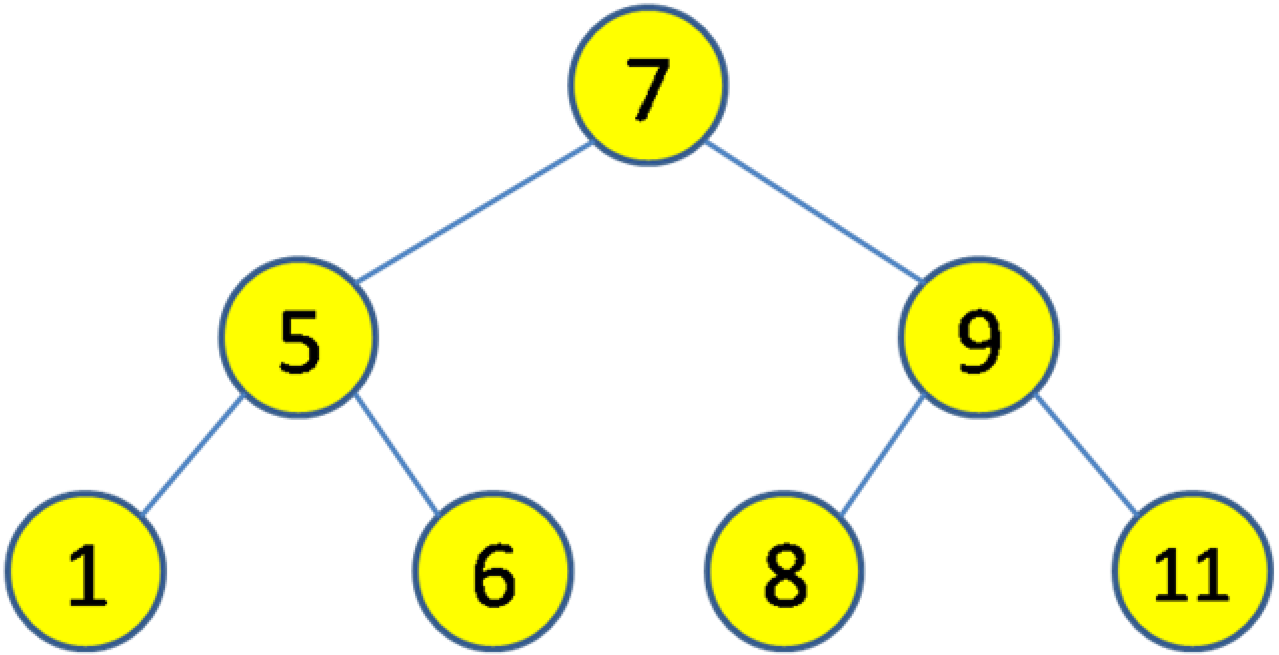
AS REMINDER :

1. Visit all the nodes in the left subtree
2. Visit root node
3. Visit all the nodes in the right subtree

QUIZ

Pre-Order traversal

Write the list of nodes visited, in case of a **PRE ORDER** transversal



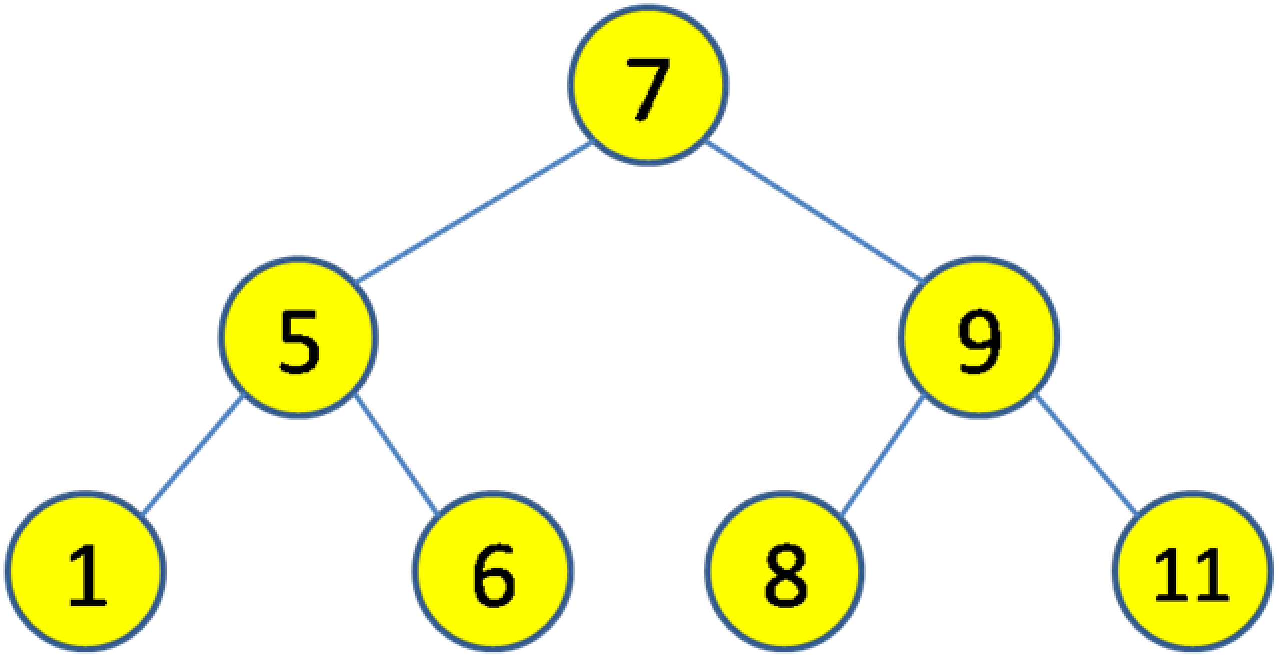
AS REMINDER :

1. Visit **root node**
2. Visit all the nodes in the **left subtree**
3. Visit all the nodes in the **right subtree**

QUIZ

Post-Order traversal

Write the list of nodes visited, in case of a **POST ORDER** transversal



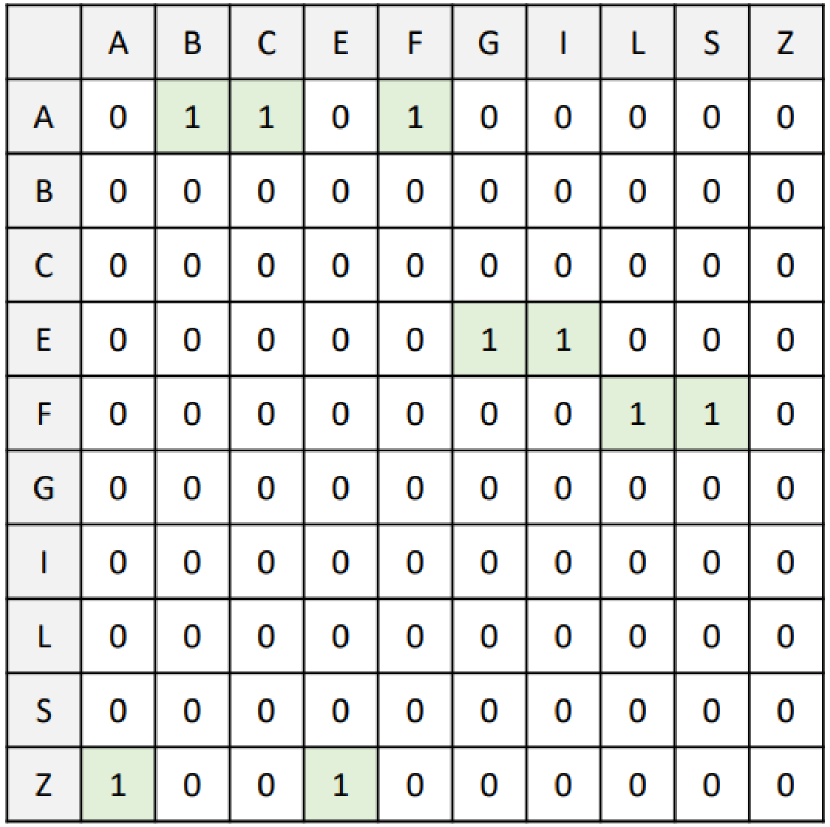
AS REMINDER :

1. Visit all the nodes in the **left subtree**
2. Visit all the nodes in the **right subtree**
3. Visit **root node**

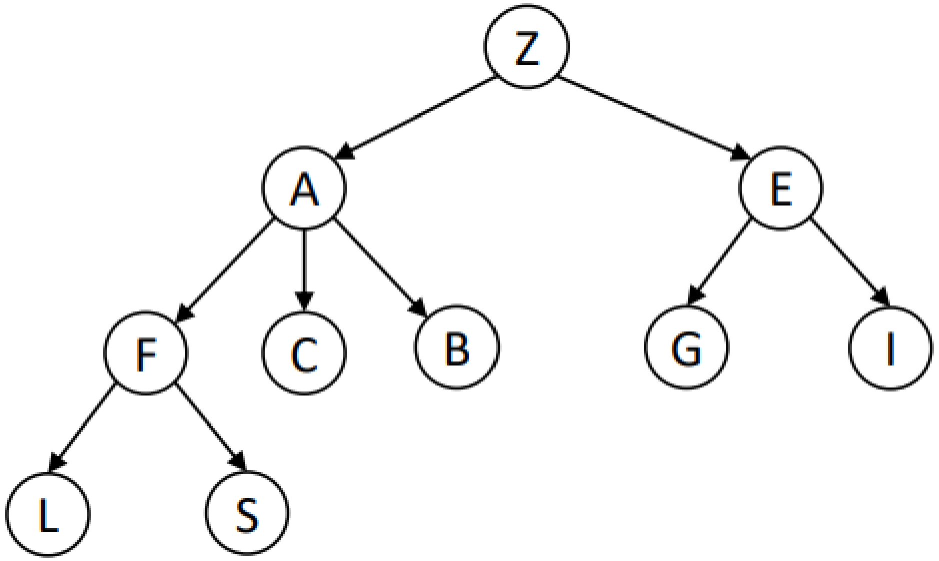
EXPLAIN

Tree Implementation- **With Array 2D**

*Each columns of the array represent a potential **children** of each node*



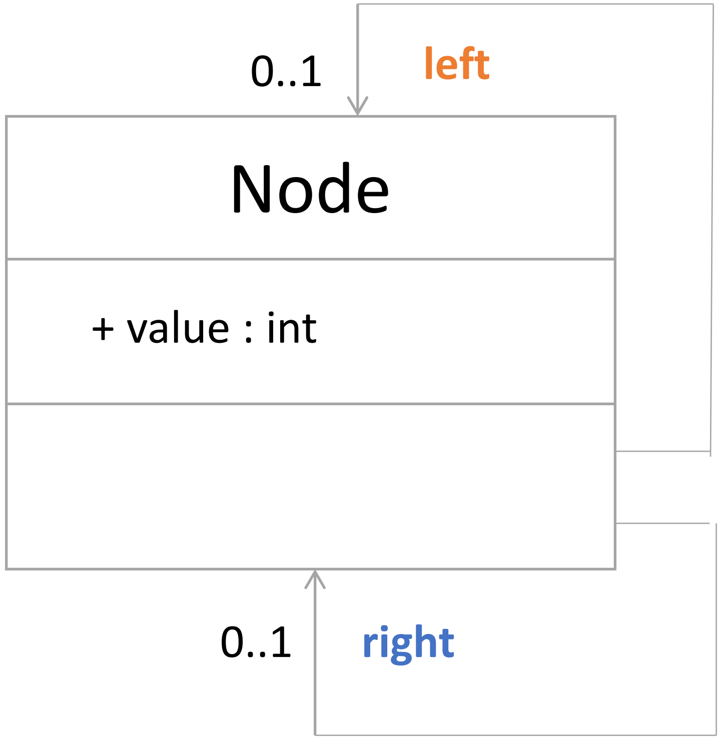
	A	B	C	E	F	G	I	L	S	Z
A	0	1	1	0	1	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	1	1	0	0	0
F	0	0	0	0	0	0	0	1	1	0
G	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0
Z	1	0	0	1	0	0	0	0	0	0



EXPLAIN

Tree Implementation- **With Pointers**

Pointers are responsible for connecting one node of the tree to another.



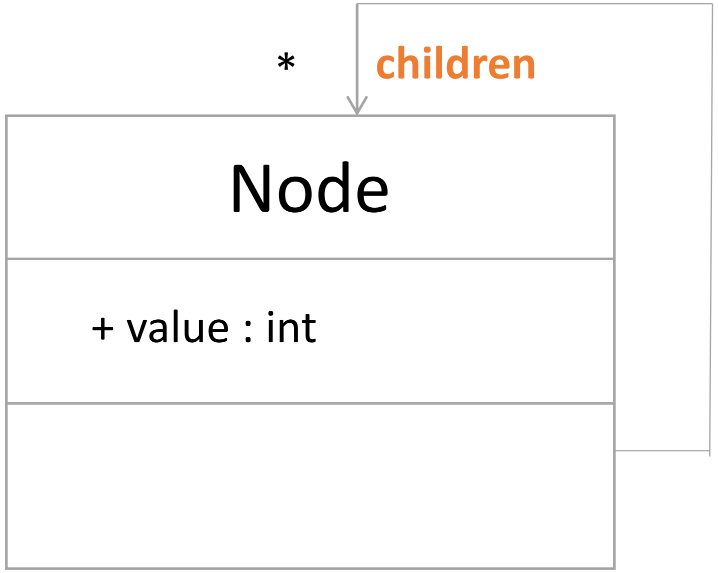
BINARY TREE

- one pointer pointing to the **left child** of the node
- another pointer pointing to the **right node** of the tree

EXPLAIN

Tree Implementation- **With Pointers**

Pointers are responsible for connecting one node of the tree to another.



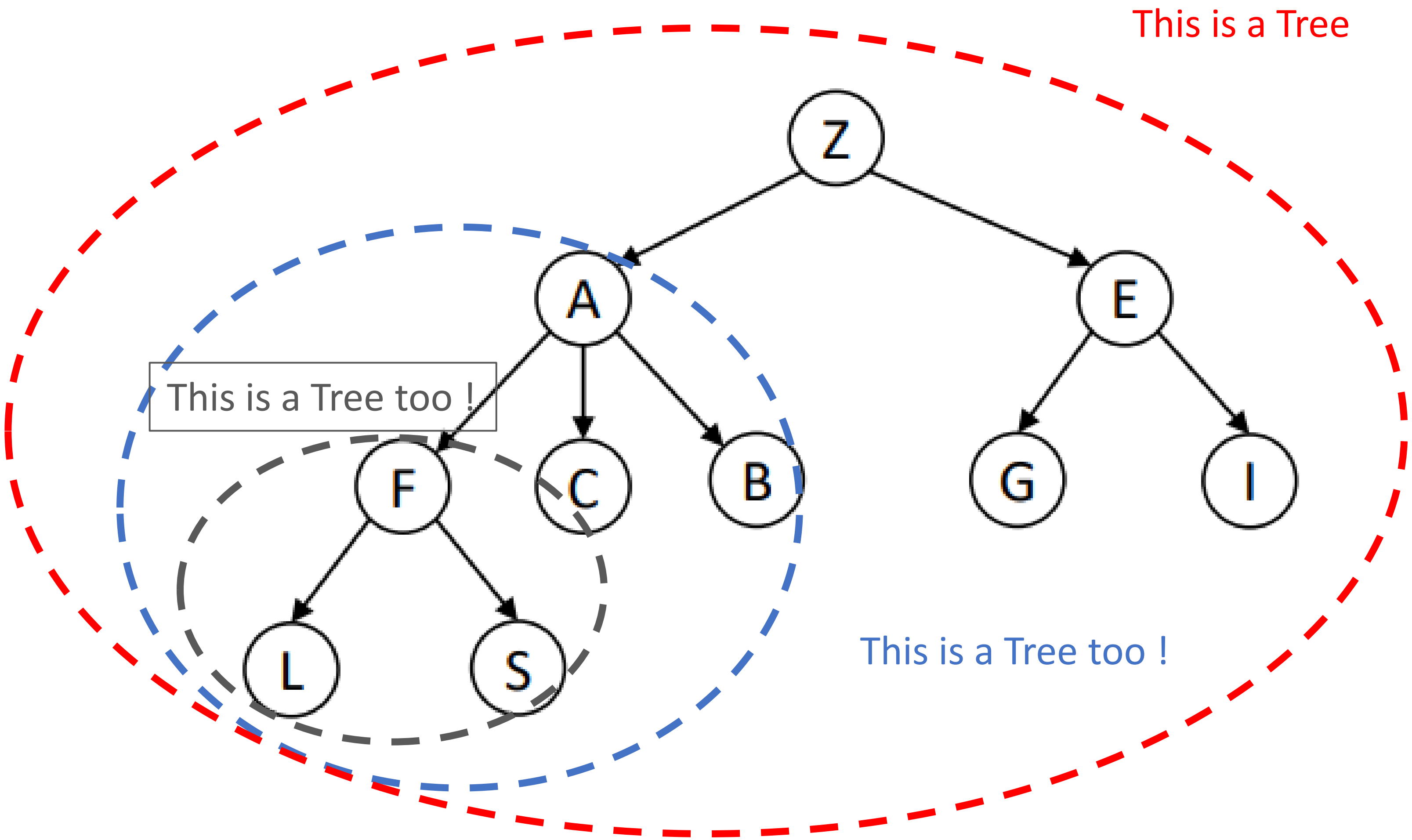
N-ARY TREE

- A list of pointer pointing to every **children**

EXPLAIN

Tree & Recursion

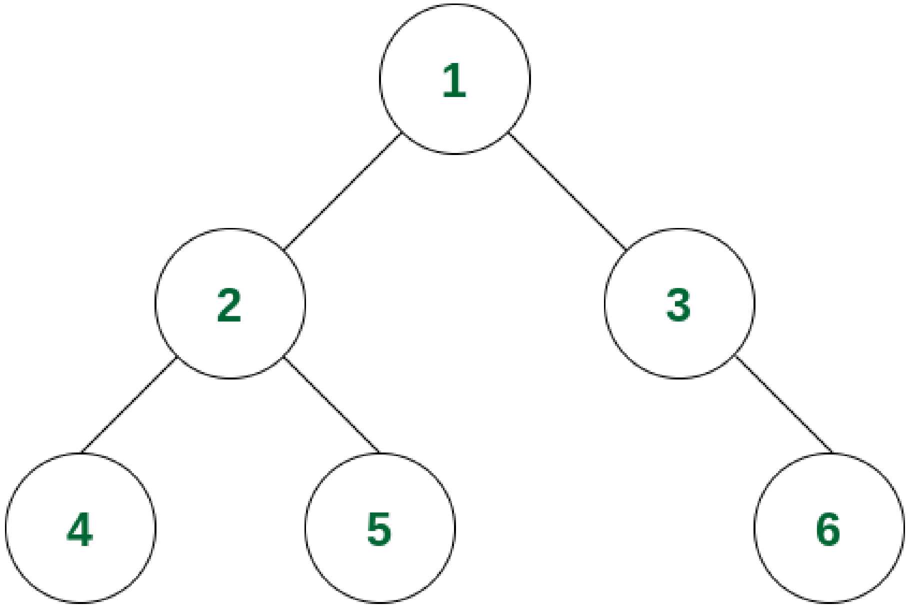
*We can use **recursion** to iterate on tree elements, as **everything is a node***



DEMO

In-Order Traversal

*We define the following pseudo code for **in-orde**r traversal*



Binary Tree to be traversed

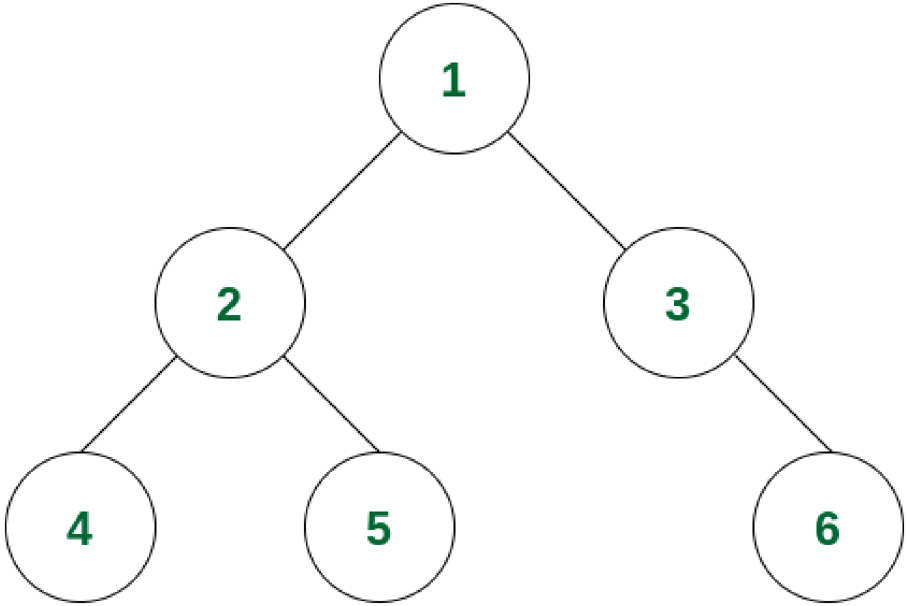
```
in_order(node):

    if node.hasLeft:
      in_order(root.leftNode)

    print(root.value)

    if node.hasRight:
      in_order(root.rightNode)
```

✓ *Execute the code to check and write the output* : - - - - - -

ACTIVITY

Pre-Order Traversal

Define the pseudo code for a pre-order traversal



Binary Tree to be traversed

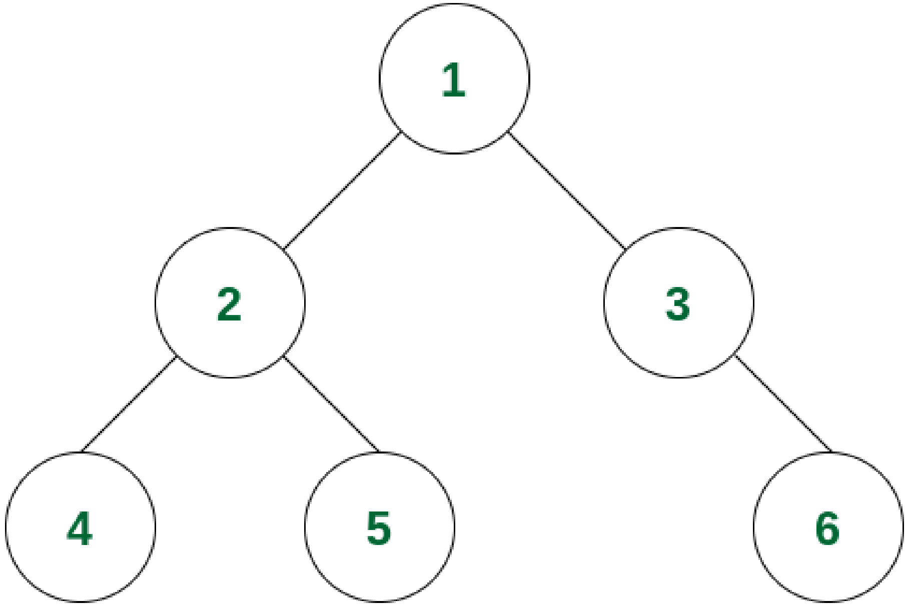
```
pre_order(node):
```

✓ *Execute the code to check and write the output* : - - - - - -

ACTIVITY

Post-Order Traversal

Define the pseudo code for a post-order traversal



Binary Tree to be traversed

```
post_order(node):
```

✓ *Execute the code to check and write the output* : - - - - - -

3-2-1 Challenge

- ✓ List three things you **learned** today.
- ✓ List two **questions** you still have.
- ✓ List one aspect of the lesson or topic you **enjoyed**.