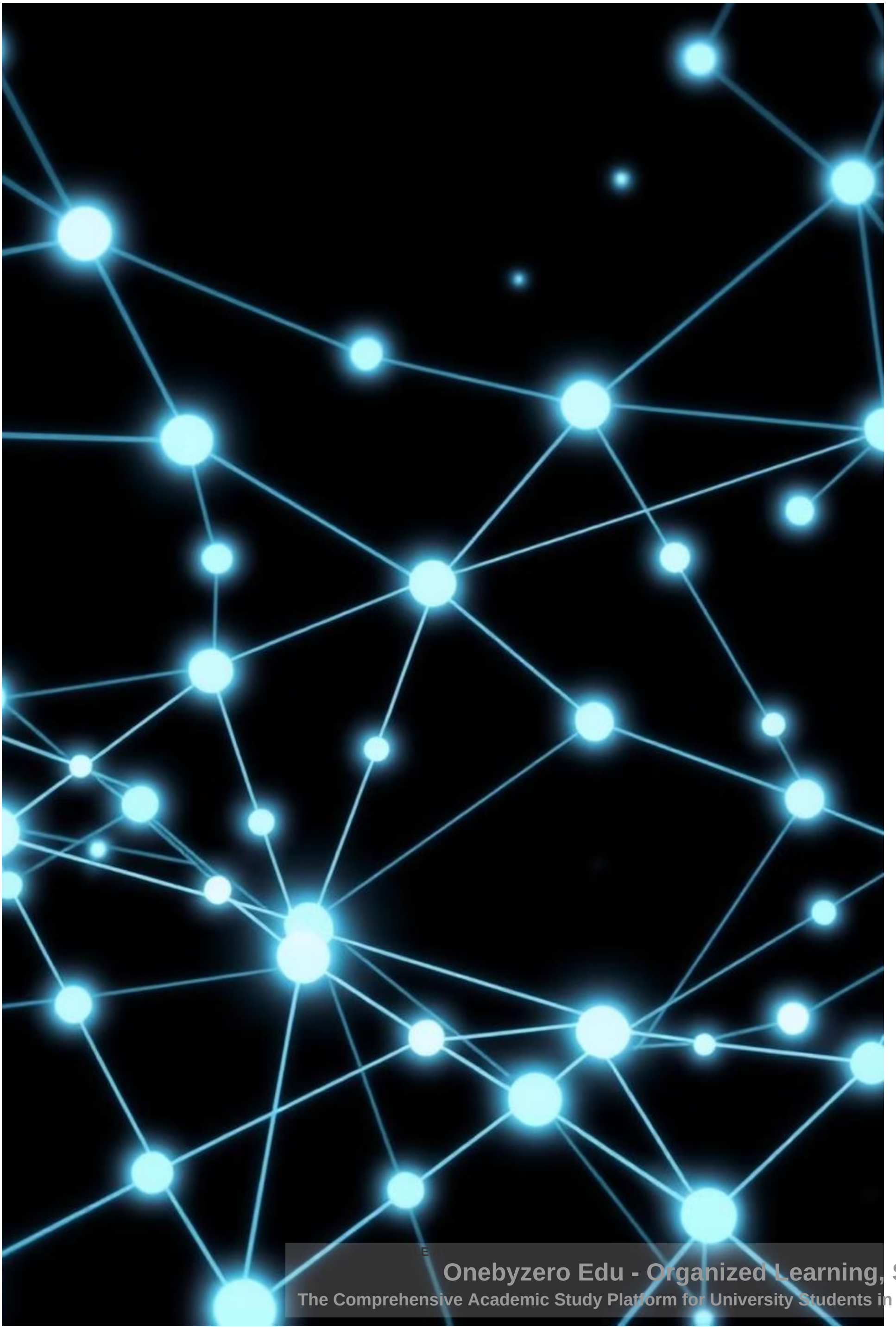


Data Structures

Lecture 6: Tree

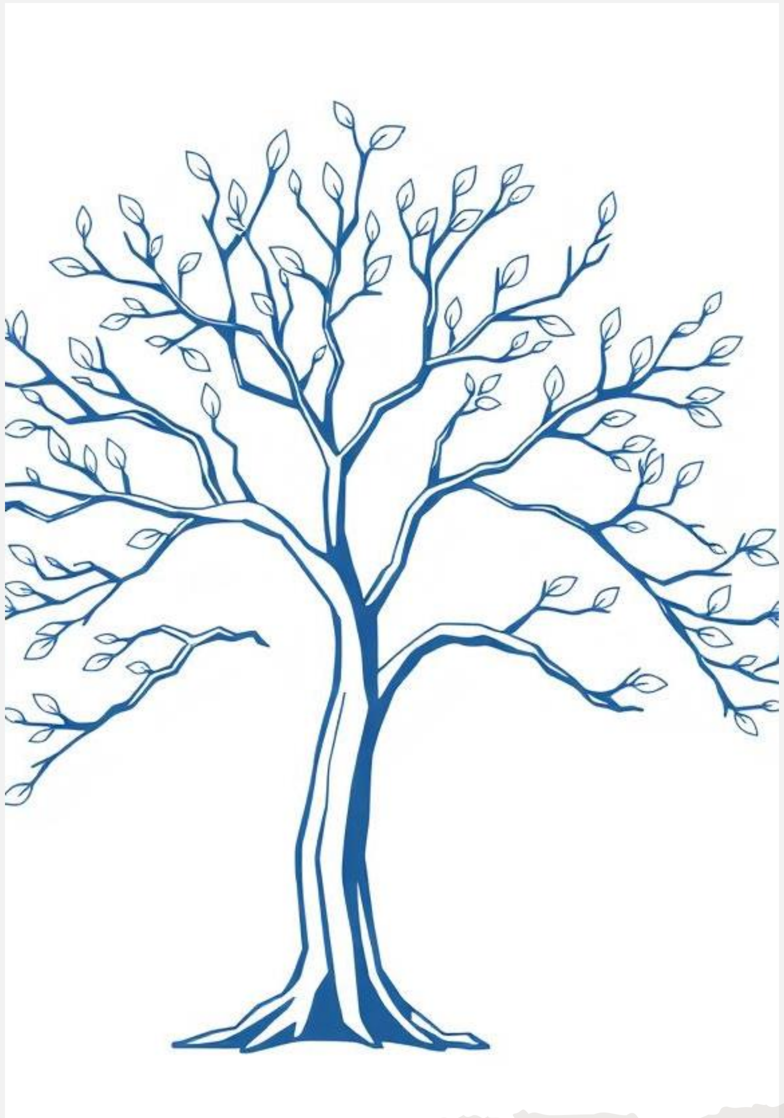
Instructor:
Md Samsuddoha
Assistant Professor
Dept of CSE, BU

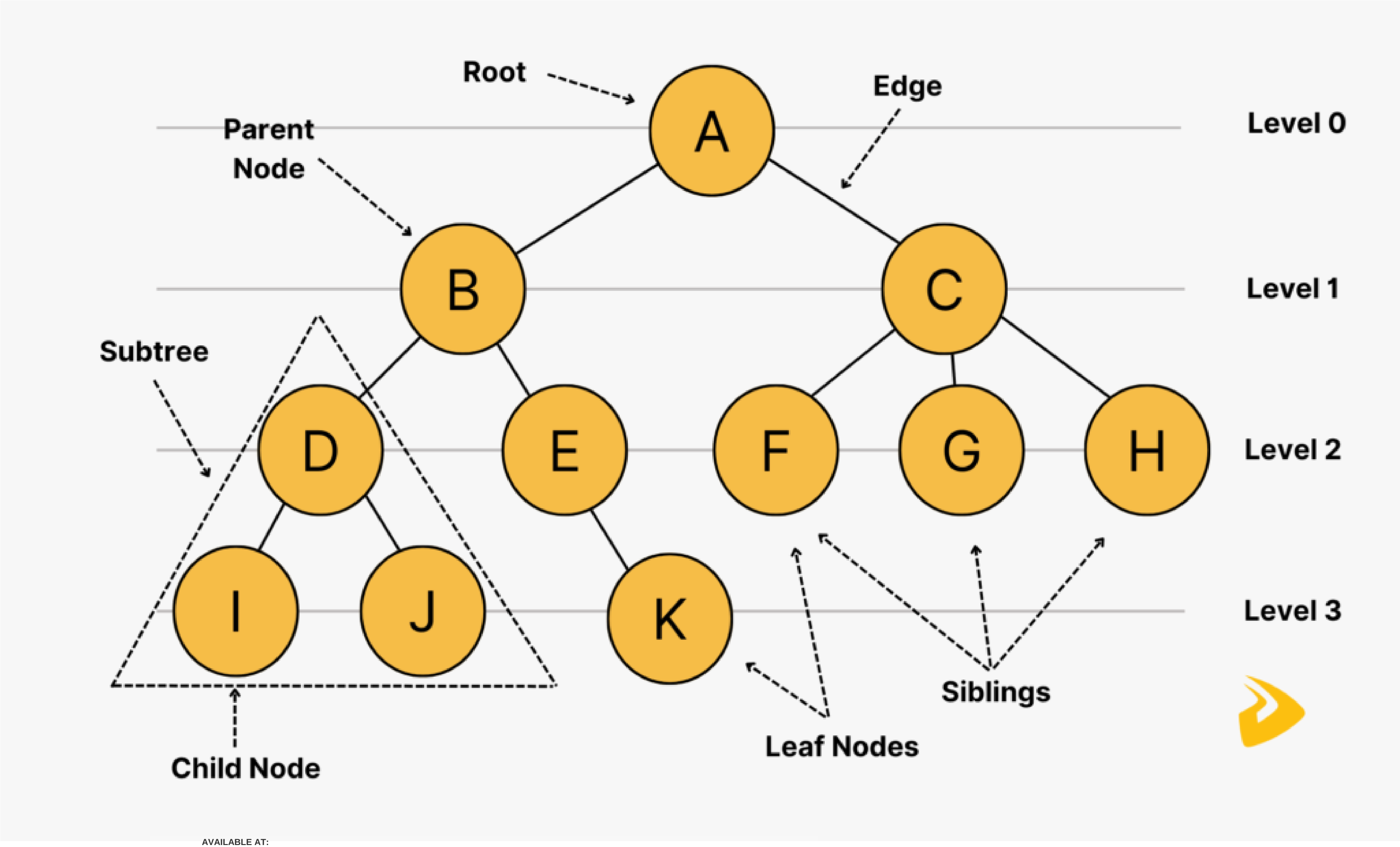
Contents

- Concept of Tree
- Types of Tree
- Binary Tree
- Types of Binary Tree
- Traversal in BT
- Construction of BT

Trees: Hierarchical Data Structures

- Trees organize data in a **hierarchical structure**, with nodes connected in parent-child relationships. They start with a single root node and branch downwards, with no cycles, ensuring a clear path from root to any node.
- There are many specialized tree types, such as **binary trees** (each node has at most two children), **binary search trees** (ordered nodes for efficient searching), **heaps** (used in priority queues), and **tries** (for string retrieval).
- Trees are extensively used in **file systems** to represent directories and files, in **databases for indexing**, in parsing expressions (abstract syntax trees), and for efficiently organizing and searching hierarchical data like the Document Object Model (DOM) in web browsers.




Root
Edge
Parent Node
Level 0
A
B
C
Level 1
Subtree
D
E
F
G
H
Level 2
I
J
K
Level 3
Siblings
Leaf Nodes
Child Node

Properties of Tree

- **Root Node**: The root node is the very top node of the tree. It has no parent. ***A*** is the root node.
- **Node**: Nodes are the individual circles (entities) in the tree. In the image, ***A, B, C, D, E, F, G, H, I, J, K*** are all nodes.
- **Edges:** Edges are the lines that connect one node to another. They show relationships between nodes (parent to child). For example, the lines A–B, A–C, B–D are all edges.
- **Parent**: A parent node is a node that has at least one child connected below it. ***A, B, C, D, E*** are parent Nodes.
- **Children**: A child node is directly connected below a parent node. All nodes are child except ***A***.
- **Leaf Nodes**: All the nodes that have no children are called leaves. Here, ***I, J, K, F, G, H*** are leaf nodes.
- ***Internal Node:*** All nodes Except Parent and Leaf (***B, C, D, E).***
- **Sibling**: All the child nodes of a parent node are siblings.

Properties of Tree

- ***Degree:*** The degree of a tree means the maximum number of children any single node has.

  A → 2 children, B → 2 children, C → 2 children, D → 2 children, E → 1 child
  F, G, H, I, J, K → 0 children

  Here, maximum is **2**. So, **Degree of the tree = 2.**

- ***Tree Size:*** Tree size means the total number of nodes in the tree. Just count all nodes: ***A, B, C, D, E, F, G, H, I, J, K = 11*** nodes.

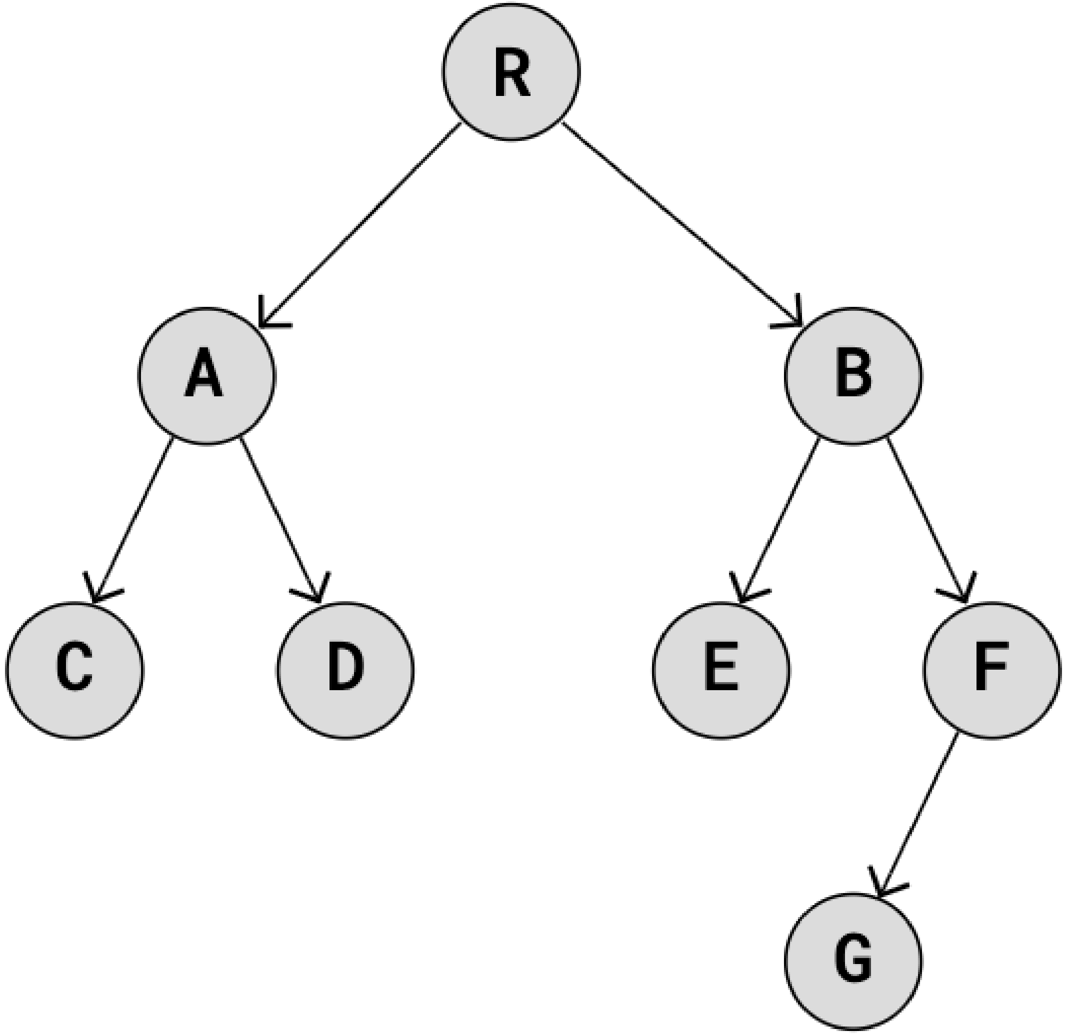
- **Tree Height:** Tree height is the number of ***levels*** from the ***root to the deepest leaf.***
  Level counting usually starts from 0 (as shown).
  Here ***Levels***:
  Level 0 → A
  Level 1 → B, C
  Level 2 → D, E, F, G, H
  Level 3 → I, J, K
  So the height = **3**.

Types of Trees

- General TreeS
- Binary TreeS
- Binary Search Trees (BST)

Binary Tree

- A **binary tree** is a *hierarchical data structure* where each node can have at most **most two children**. ***No of Children of a Node (0, 1, 2).***
- These children are typically labeled **left child** and **right child.**
- This restriction, that a node can have a maximum of two child nodes, gives us many benefits:
  - Algorithms like ***traversing, searching, insertion and deletion*** become easier to understand, to implement, and run faster.
  - Keeping data sorted in a ***Binary Search Tree (BST)*** makes searching very efficient.
  - ***Balancing trees is easier*** to do with a limited number of child nodes, using an AVL Binary Tree for example.
  - Binary Trees can be represented as ***arrays***, making the tree more memory efficient.


R
A
B
C
D
E
F
G

Finding number of Nodes and Height for a Tree

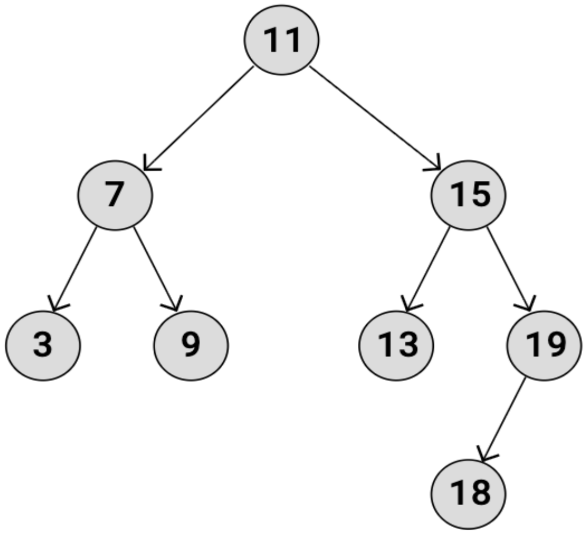
- Maximum Number of Nodes in a BT
  $1+2+4+....+ 2^h =$ **$2^{h+1} - 1$**, **Here,** *h is the height of Tree.*
- Minimum Number of Nodes in a BT
  - **h+1** (Height of tree +1)
- Max Height of a tree
  - This happens when the tree is totally skewed. Every node has only one child. This happens for min number of nodes.
  - N=h+1 => ***h=N-1*** (N, number of nodes)
- Min height for tree
  - This happens when the tree is perfectly balanced and fully filled. For maximum number of nodes.
  - $N= 2^{h+1} – 1$ => ***$h= log2(N + 1) - 1$***

Exercises

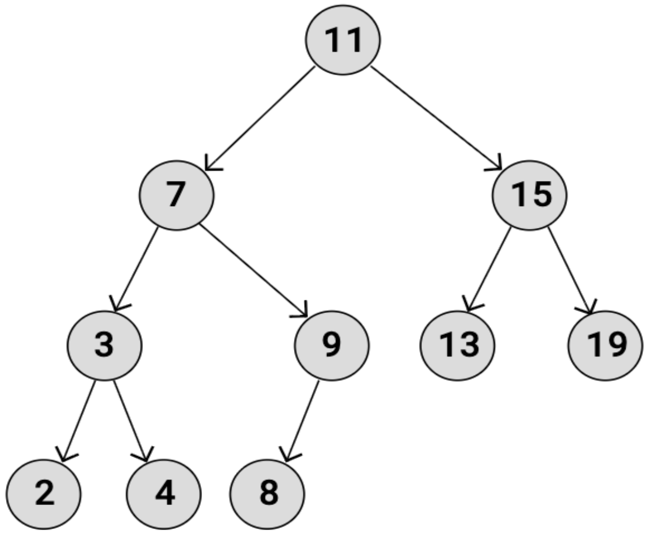
- If a binary tree has a height of 4, what are the maximum and minimum numbers of nodes it can have?
- If a binary tree has 7 nodes, what are the minimum and maximum possible heights of the tree?

Types fo Binary Tree

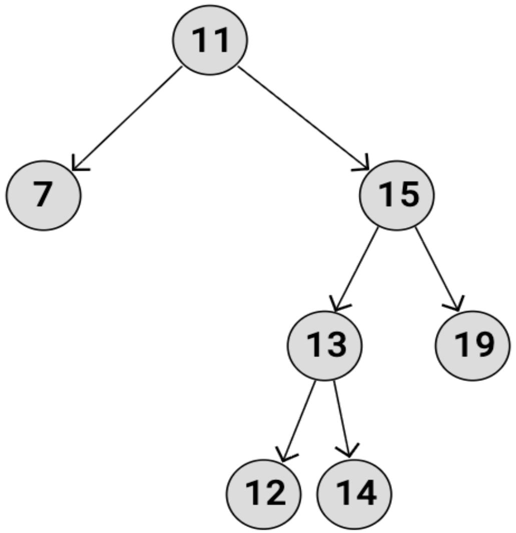
- ***Balanced Binary Tree***: A balanced Binary Tree has at most 1 in difference between its ***left and right subtree heights***, for each node in the tree.
- ***Complete Binary Tree***: A complete Binary Tree has all ***levels full of nodes, except the last level***, which is can also be full, or filled from left to right. The properties of a complete Binary Tree means it is also balanced.
- ***Full Binary Tree***: A full Binary Tree is a kind of tree where each node has either ***0 or 2 child nodes***.
- ***Perfect Binary Tree:*** A **perfect** Binary Tree has all leaf nodes on the same level, which means that ***all levels are full of nodes,*** and all internal nodes have two child nodes. The properties of a perfect Binary Tree means it is also full, balanced, and complete.



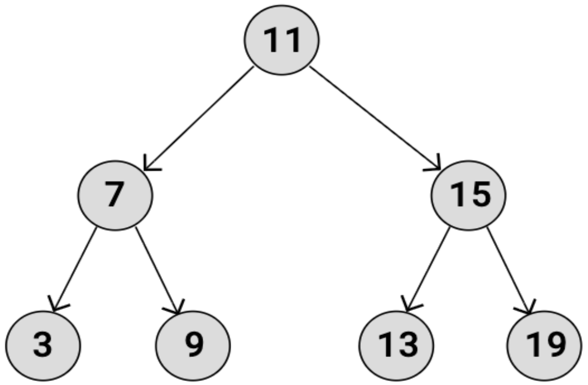
Balanced



Complete and balanced



Full



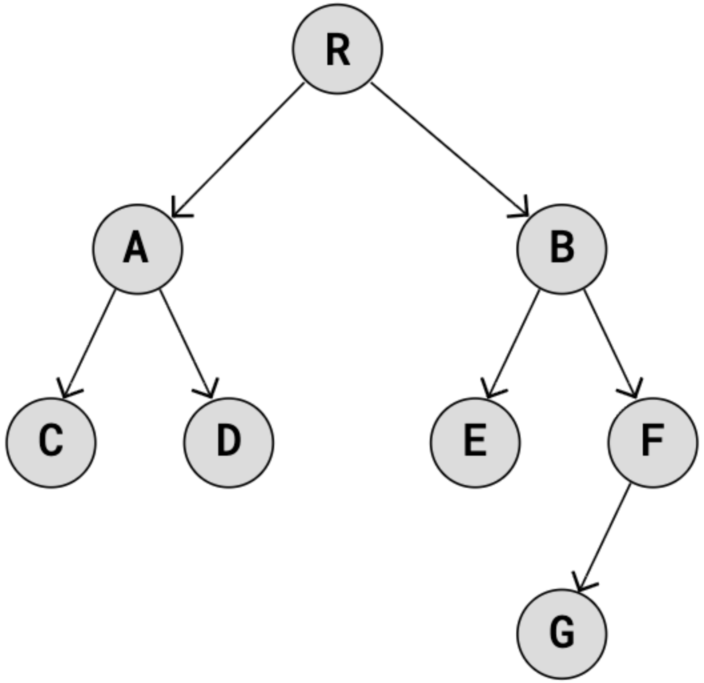
Perfect, full, balanced and complete

Binary Tree Traversal

- Traversal is a process to ***visit all the nodes of a tree*** and may print their values too.
- All nodes are connected via edges (links) we always start from the ***root (head) node***.
- Random access of a node in a tree is not possible.
- There are three ways which we use to traverse a tree –
  - Pre-order Traversal [Root, Left , Right / N-L-R]
  - In-order Traversal [Left, Root, Right / L-N-R]
  - Post-order Traversal [Left, Right, Root / L-R-N]

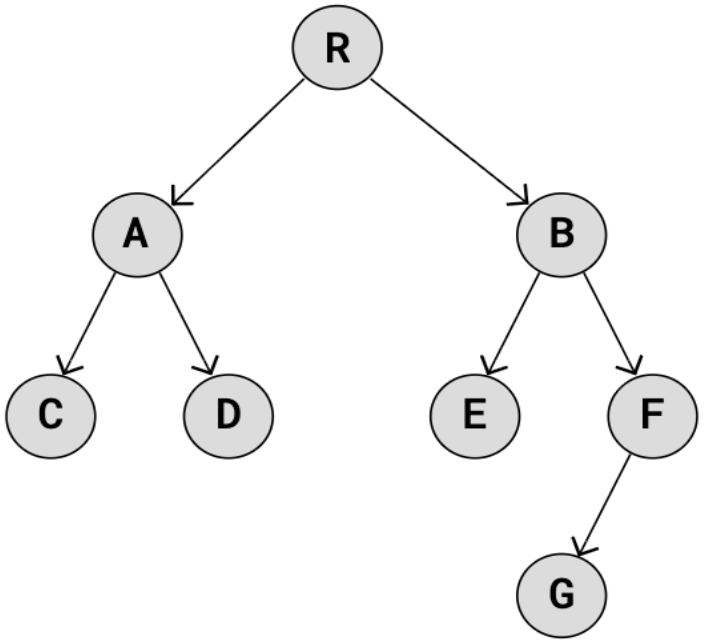
Pre-order Traversal

- In this traversal method, the root node is visited first, then the left subtree and finally the right subtree ***[Root-Left-Right]***.
- Traversal Sequence for the following tree : ***R,A,C,D,B,E,F,G***



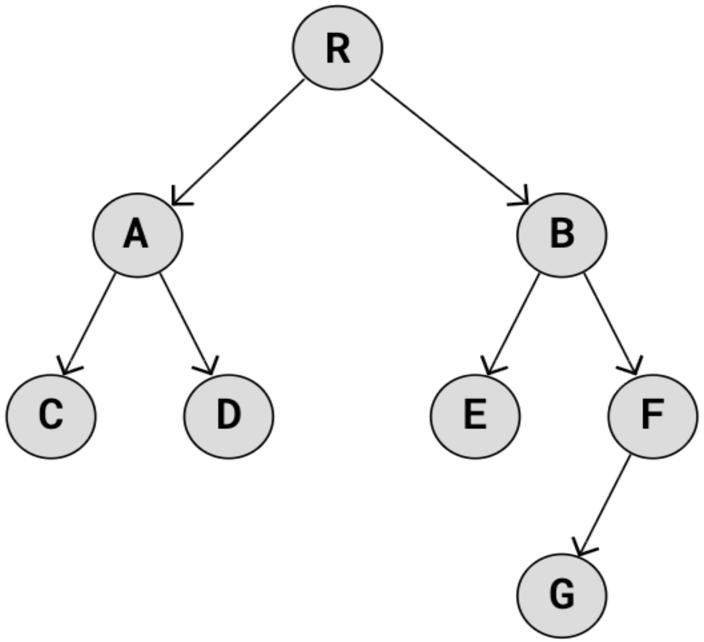
In-order Traversal

- In this traversal method, the left subtree is visited first, then the root and later the right sub-tree. We should always remember that every node may represent a subtree itself ***[Left- Root-Right]***.
- Traversal Sequence for the following tree : ***C,A,D,R,E,B,G,F***



Post-order Traversal

- In this traversal method, the root node is visited last, hence the name. First we traverse the left subtree, then the right subtree and finally the root node ***[Left- Right-Root]***.
- Traversal Sequence for the following tree :
- ***C→D →A →E→G→F→B→R***



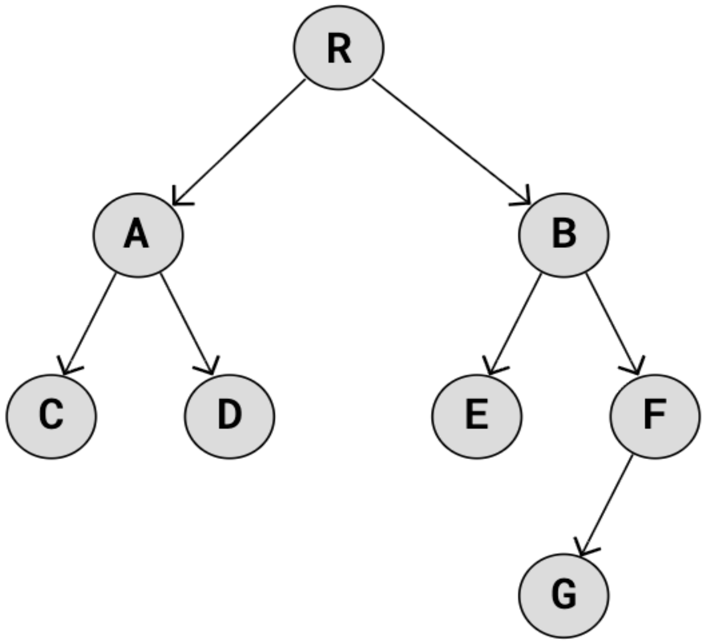
Construct binary Tree from a Sequence

- To construct a unique binary tree following combination of node sequences require:
  - Inorder and preorder
  - Inorder and postorder

Construct Tree from inorder and preorder

- Preorder: ***R,A,C,D,B,E,F,G***
- Inorder: ***C,A,D,R,E,B,G,F***

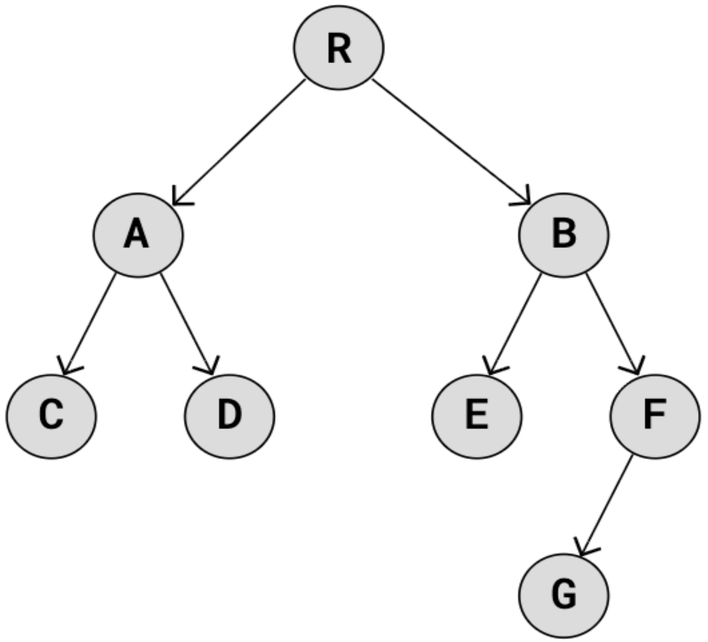
- *Exercises:*
  - *Preorder: A, B, D, G, K, H, L, M, C, E*
  - *Inorder: K, G, D, L, H, M, B, A, E, C*



Construct Tree from inorder and postorder

- Postorder: ***C, D, A, E, G, F,B, R***
- Inorder: ***C,A,D,R,E,B,G,F***

- *Exercises:*
  - *Postorder: K, G, L, M, H, D, B, E, C, A*
  - *Inorder: K, G, D, L, H, M, B, A, E, C*



Tree Applications

- Binary Search Trees(BSTs) are used to quickly check whether an element is present in a set or not.
- Heap is a kind of tree that is used for heap sort.
- A modified version of a tree called Tries is used in modern routers to store routing information.
- Most popular databases use B-Trees and T-Trees, which are variants of the tree structure we learned above to store their data
- Compilers use a syntax tree to validate the syntax of every program you write.

References

- **Chapter 8: Tree (Data Structures using C** by E. Balagurusamy)

Thank You