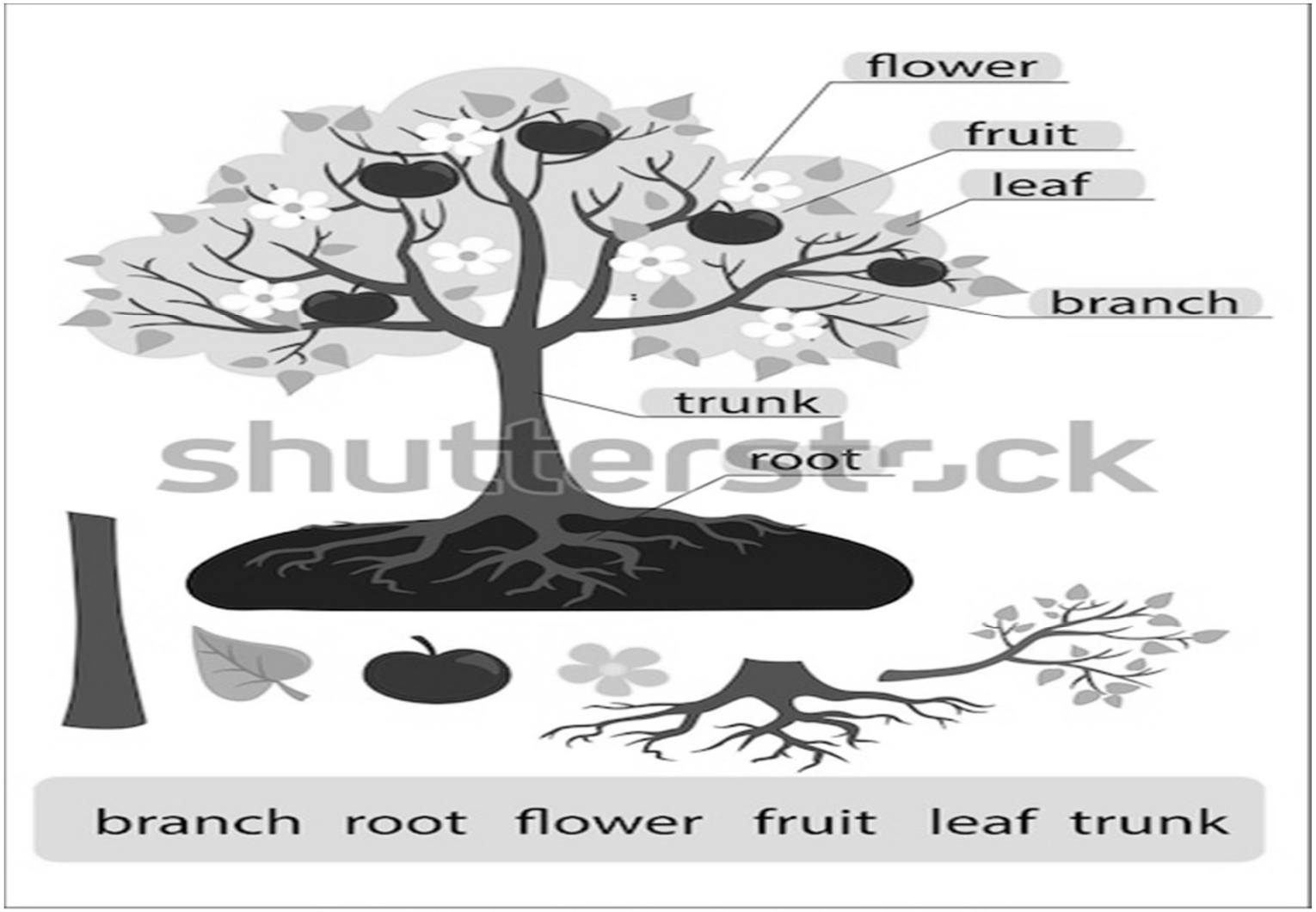
Data Structure
Algorithms & Applications

CT-159

Prepared by
Muhammad Kamran

Trees


flower
fruit
leaf
branch
trunk
root
shutterstock
branch root flower fruit leaf trunk
www.shutterstock.com · 252114028

Tree

A tree is a **hierarchical** data structure used to organize and represent data in a **parent–child** relationship. It consists of nodes, where the topmost node is called the **root**, and every other node can have one or more **child nodes**.

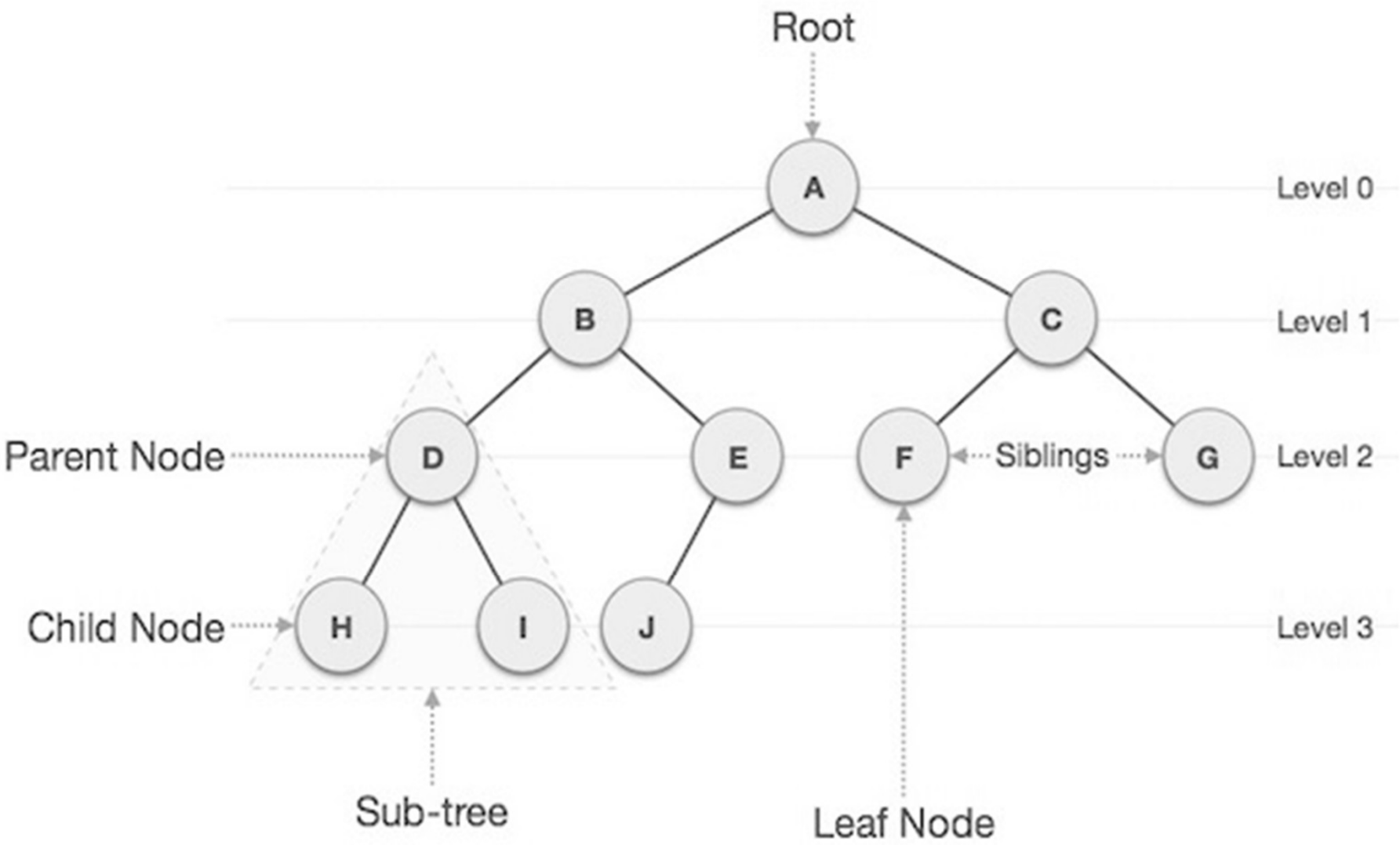
Binary Tree

A **Binary Tree Data Structure** is a hierarchical data structure in which each node has at most two children, referred to as the left child and the right child. It is commonly used in computer science for efficient storage and retrieval of data, with various operations such as insertion, deletion, and traversal.

Binary Tree

- Tree represents the nodes connected by edges.
- Binary Tree is a special data structure used for data storage purposes.
- A binary tree has a special condition that each node can have a maximum of two children.
- A binary tree has the benefits of both an ordered array and a linked list as search is as quick as in a sorted array and insertion or deletion operation are as fast as in linked list.

Binary Tree


Root
A
B
C
D
E
F
G
H
I
J
Level 0
Level 1
Level 2
Level 3
Parent Node
Siblings
Child Node
Sub-tree
Leaf Node

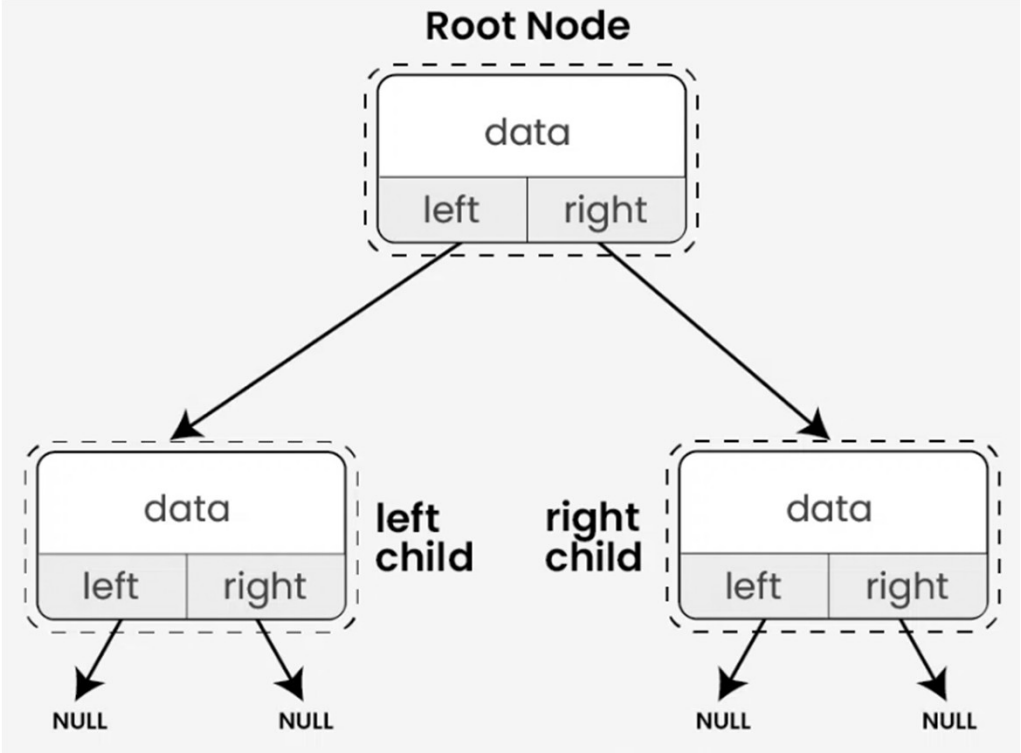
Important Terms

- **Path** − Path refers to the sequence of nodes along the edges of a tree.
- **Root** − The node at the top of the tree is called root. There is only one root per tree and one path from the root node to any node.
- **Parent** − Any node except the root node has one edge upward to a node called parent.
- **Child** − The node below a given node connected by its edge downward is called its child node.
- **Leaf** − The node which does not have any child node is called the leaf node.
- **Subtree** − Subtree represents the descendants of a node.
- **Levels** − Level of a node represents the generation of a node. If the root node is at level 0, then its next child node is at level 1, its grandchild is at level 2, and so on.

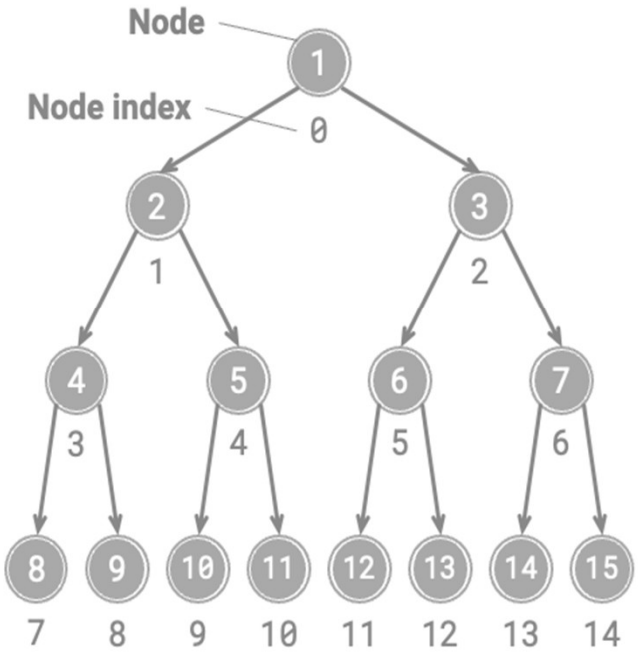
Representation

Each node in a Binary Tree has three parts:

- Data
- Pointer to the left child
- Pointer to the right child



Representation



Current node index: `i`

Mapping

Left child node index: `2i + 1`

Right child node index: `2i + 2`

Using arrays to represent binary trees

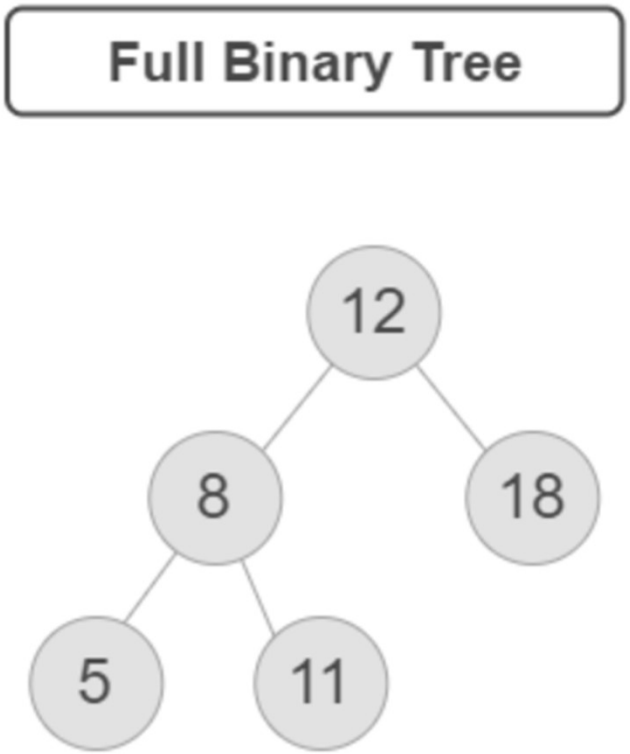
Properties

- The maximum number of nodes at level **L** of a binary tree is $\mathbf{2^L}$
- The maximum number of nodes in a binary tree of height **H** is $\mathbf{2^H - 1.}$
- In a Binary Tree with **N** nodes, the minimum possible height or the minimum number of levels is $\mathbf{Log_2(N+1).}$
- A Binary Tree with **L** leaves has at least

  $\mathbf{| Log_2 L |+ 1}$ levels.

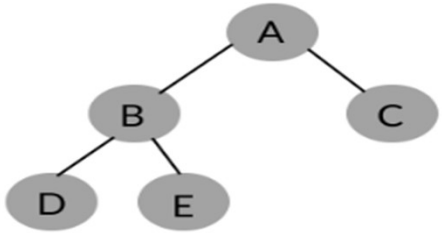
Types
On the basis of Number of Children

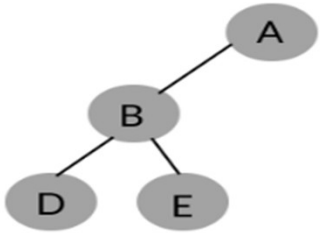
- Full Binary Tree
- Degenerate Binary Tree
- Skewed Binary Trees

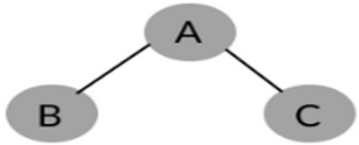


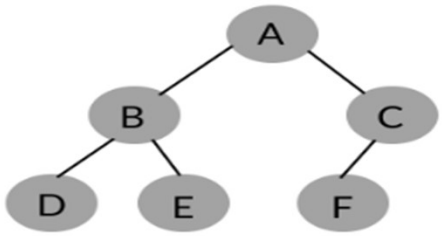
Types
On the basis of Completion of Levels

- Complete Binary Tree
- Perfect Binary Tree
- Balanced Binary Tree

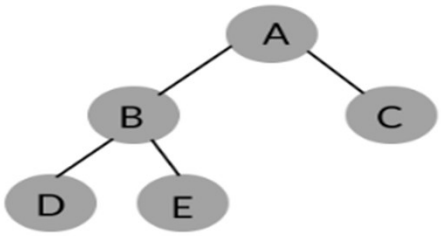




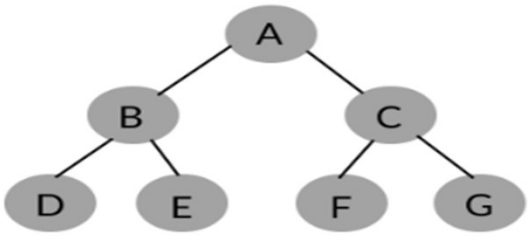




Complete Binary Tree



Full Binary Tree



Perfect Binary Tree

Other Types

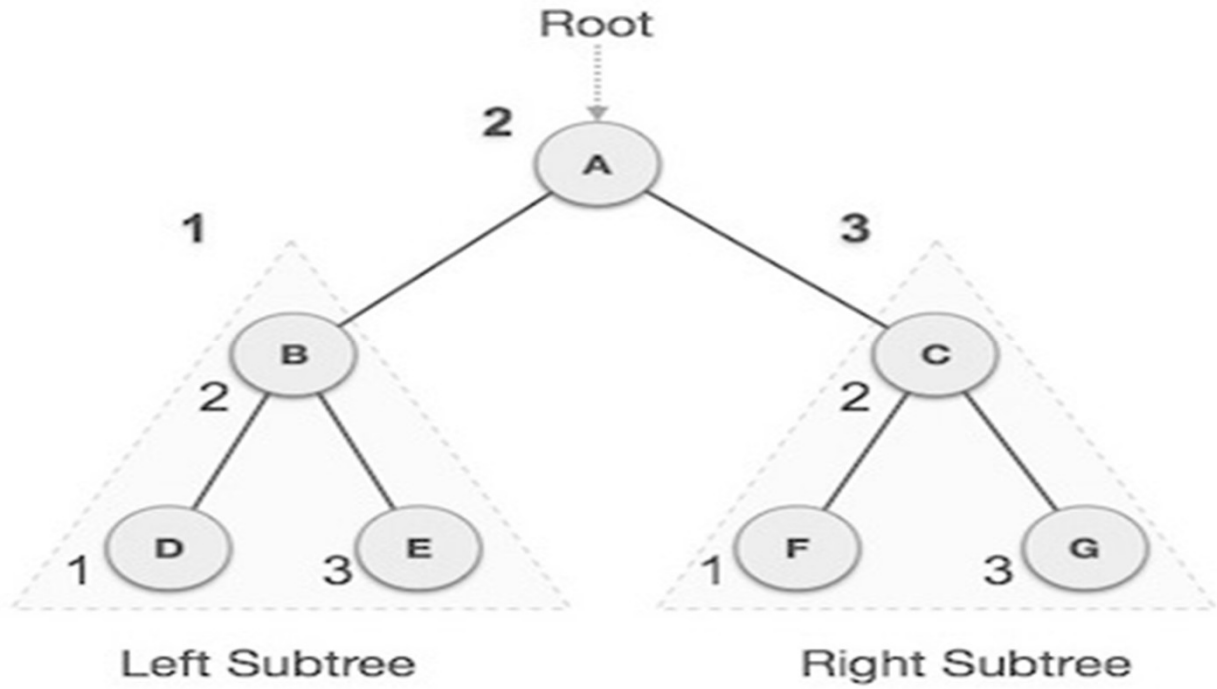
- Binary Search Tree
- AVL Tree
- Binary Heap Tree, etc.

Operations On Binary Tree

Traversal in Binary Tree involves visiting all the nodes of the binary tree. Tree Traversal algorithms can be classified broadly into two categories, **DFS** and **BFS**:

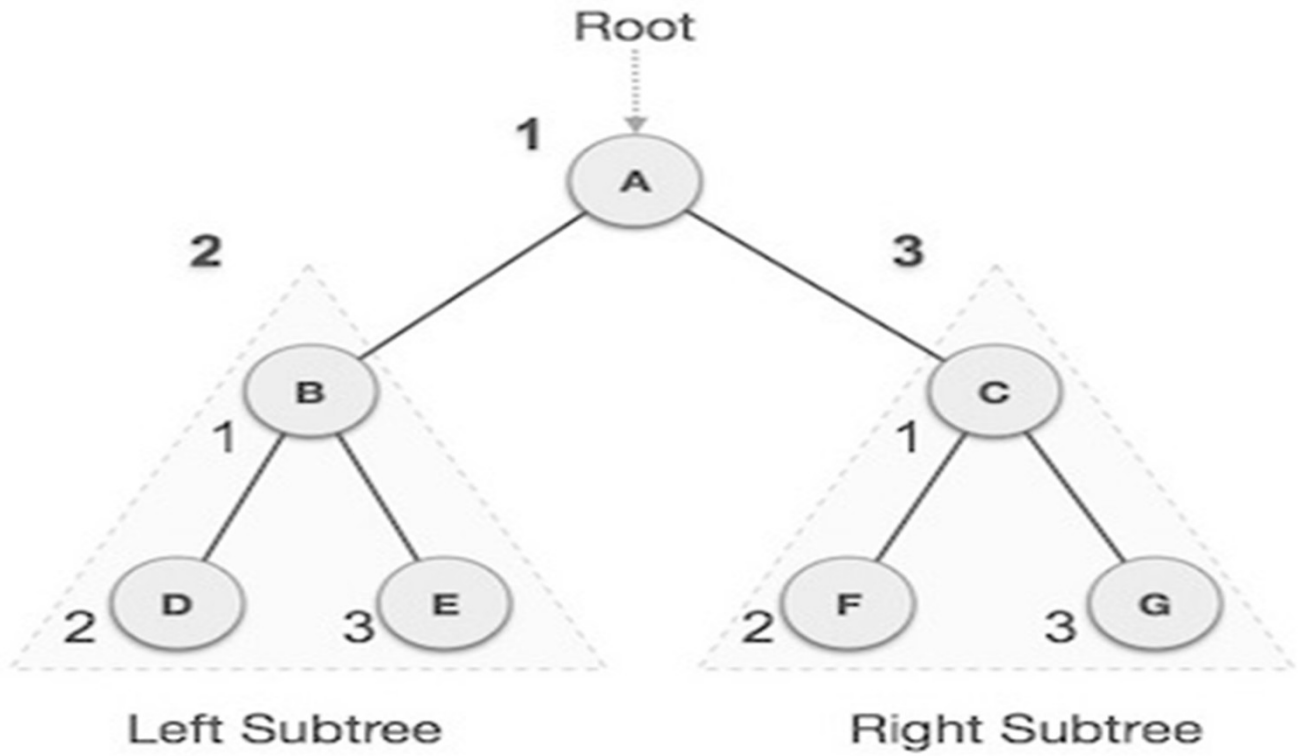
- **Depth-First Search (DFS) algorithms:** DFS explores as far down a branch as possible before backtracking. It is implemented using recursion. The main traversal methods in DFS for binary trees are:

In-order Traversal



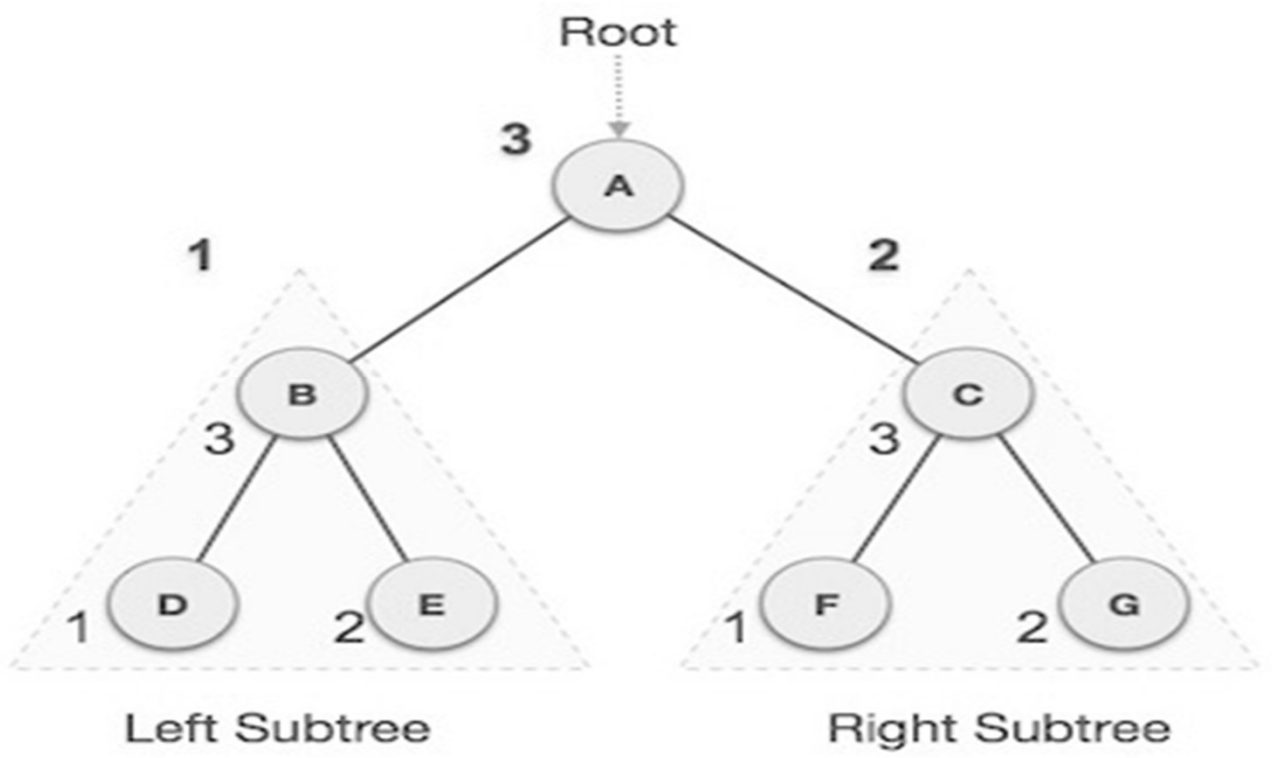
$$D \rightarrow B \rightarrow E \rightarrow A \rightarrow F \rightarrow C \rightarrow G$$

Pre-order Traversal



$$A \rightarrow B \rightarrow D \rightarrow E \rightarrow C \rightarrow F \rightarrow G$$

Post-order Traversal



$$D \rightarrow E \rightarrow B \rightarrow F \rightarrow G \rightarrow C \rightarrow A$$

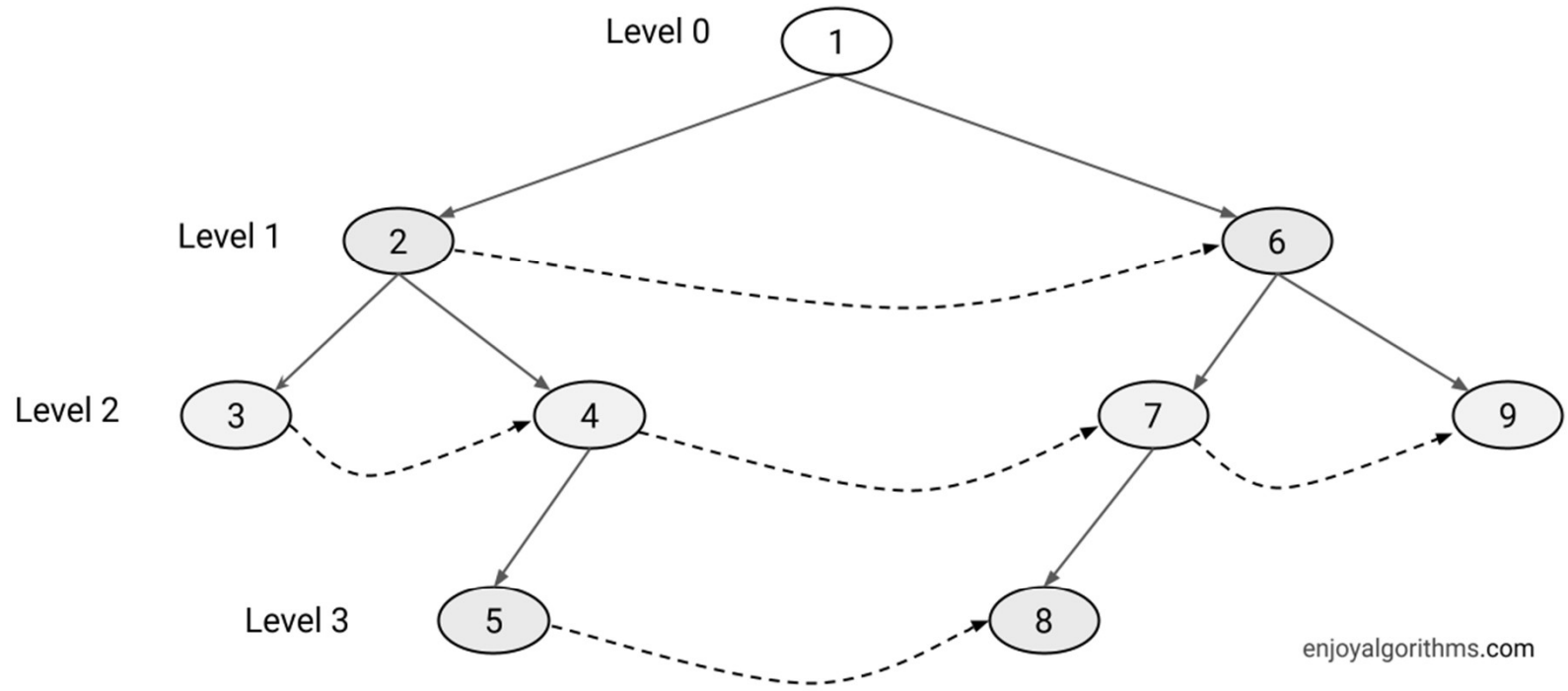
Traversal in Binary Tree

Breadth-First Search (BFS) algorithms:

BFS explores all nodes at the present depth before moving on to nodes at the next depth level. It is typically implemented using a queue.

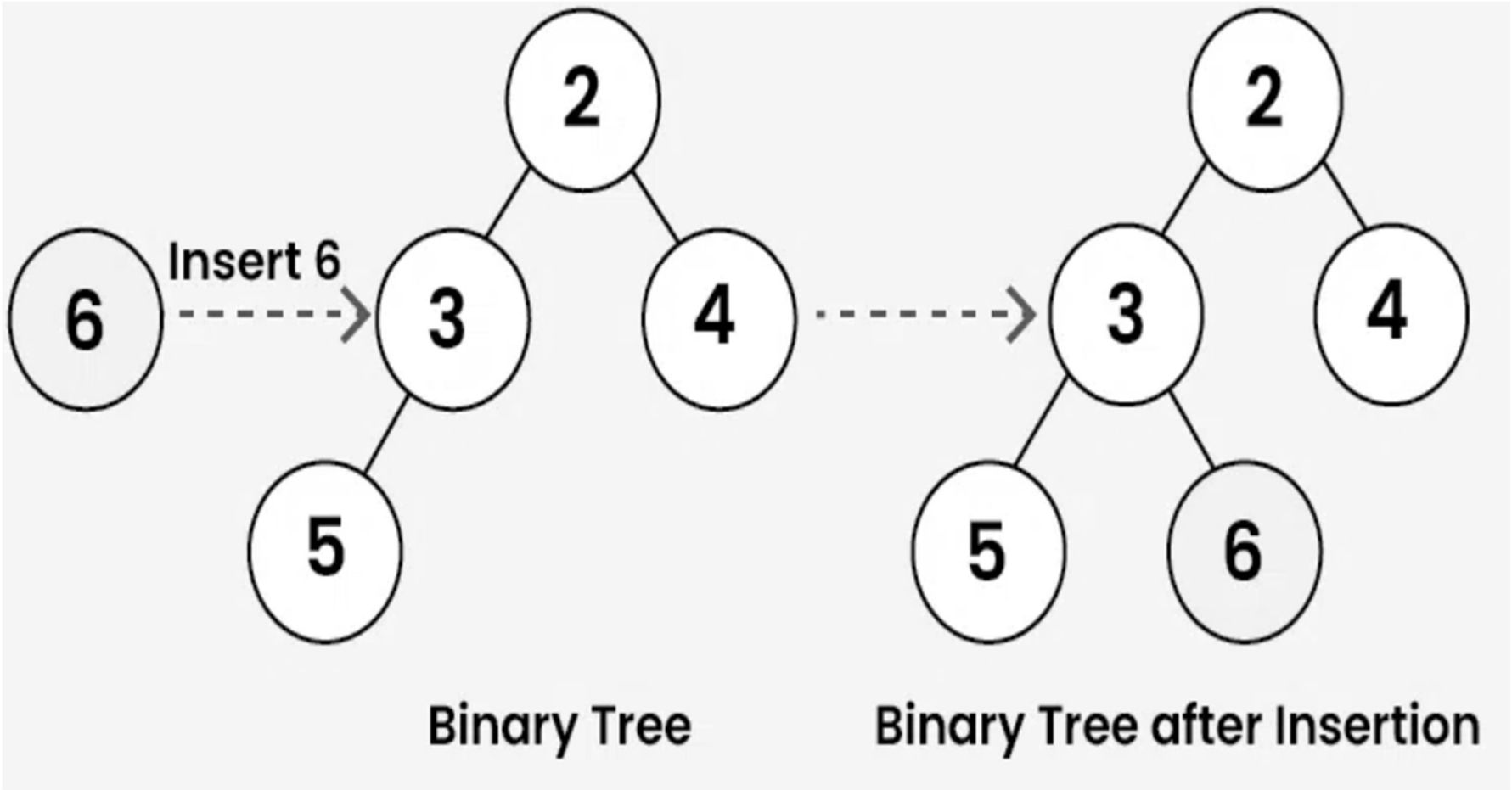
BFS in a binary tree is commonly referred to as Level Order Traversal.

BFS Traversal

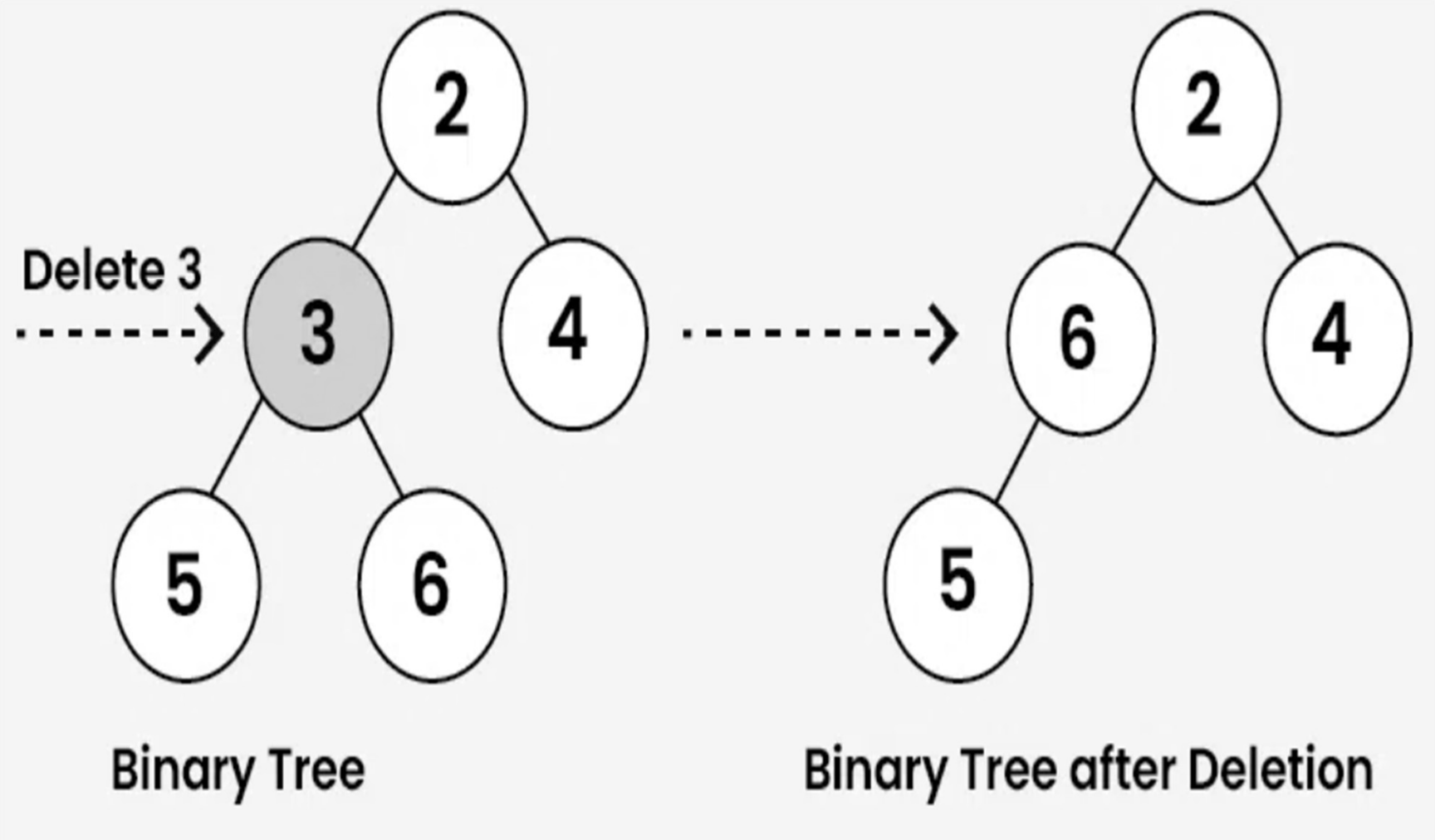


Level Order Traversal

Insertion in Binary Tree



Deletion in Binary Tree



Advantages of Binary Tree

- Efficient Search
- Memory Efficient
- Binary trees are relatively easy to implement and understand as each node has at most two children, left child and right child.

Disadvantages of Binary Tree

- Limited structure
- Unbalanced trees
- Space inefficiency
- Slow performance in worst-case scenarios

Applications of Binary Tree

- Binary Tree can be used to represent hierarchical data.
- Huffman Coding trees are used in data compression algorithms.
- Priority Queue is another application of binary tree that is used for searching maximum or minimum in O(1) time complexity.
- Useful for indexing segmented at the database is useful in storing cache in the system,
- Binary trees can be used to implement decision trees, a type of machine learning algorithm used for classification and regression analysis.

Thank You