Data Structures and Algorithms

Tree

PRASHANT HEMRAJANI

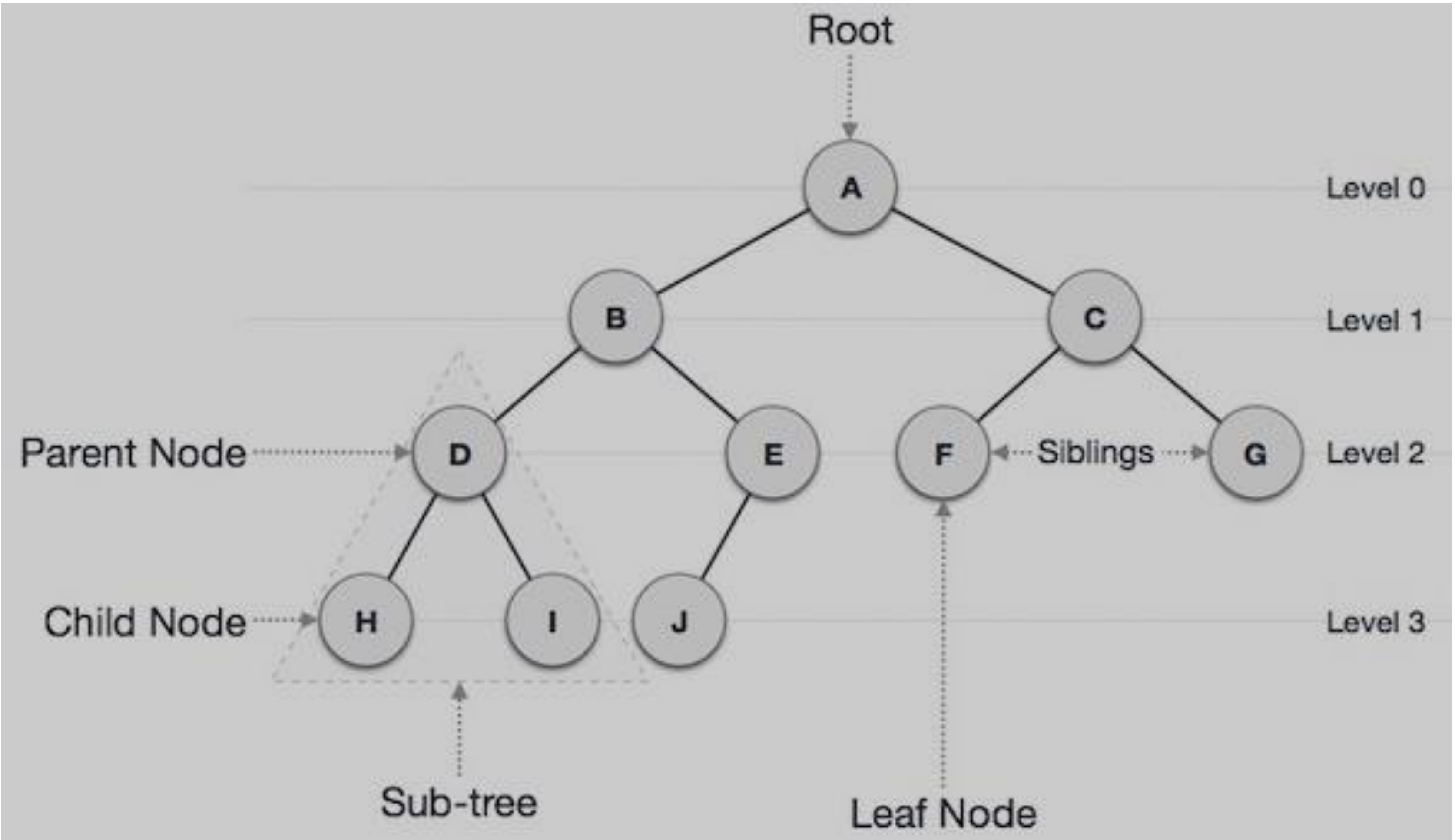
Assistant Professor

(Computer and Communication Engineering)

Introduction

- 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.

Representation



Application

- **Decision Making**
  - Next Move in games
  - Computer chess games build a huge tree (training) which they prune at runtime using heuristics to reach an optimal move.
- **Networking**
  - Router algorithms -Network Routing.
  - Social networking is the current buzzword in CS research.
- **Manipulate Hierarchical Data**
  - Make information easy to search.
  - Manipulate sorted lists of data.
- **Workflow**
  - As a workflow for compositing digital images for visual effects.
- **Organizing Things**
  - Folders/ Files in the Operating System
  - HTML Document Object Model (DOM)
  - Company Organization Structures
- **Faster Lookup**
  - Auto-correct applications and spell checker
  - Syntax Tree in Compiler
- **Task Tracker**
  - Undo function in a text editor

Important Terms

Following are the important terms with respect to tree.

- **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.
- **Visiting** – Visiting refers to checking the value of a node when control is on the node.
- **Traversing** – Traversing means passing through nodes in a specific order.
- **keys** – Key represents a value of a node based on which a search operation is to be carried out for a node.

Important Terms

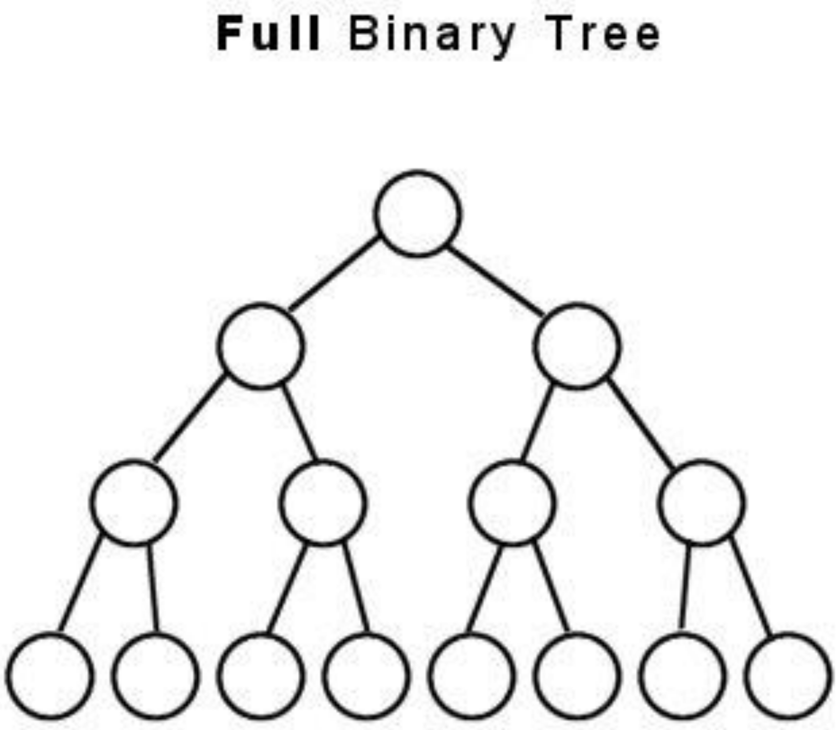
- **Levels –** Level of a node represents the generation of a node. If the root node is at level 0, then its next child node is at level 1, its grandchild is at level 2, and so on.
- **Siblings –** If N is a node in T that has a left successor s1, and a right successor s2, then N is called the parent of s1 and s2. Correspondingly s1 and s2 are called the left child and right child of N. Also s1 and s2 are said to be siblings.
- **Degree –** The degree of a node is equal to the number of children that a node has. The degree of a leaf node is zero. If a node has two child nodes then its degree will be one.
- **Depth –** The depth of a node N is given as the length of the path from the root R to the node N. the depth of the root node is zero. The height/depth of a tree is defined as the length of the path from the root node to the deepest node in the tree. The height of a binary tree with n nodes is at least n and at the most log(n+1).

Types of Binary Trees

- Binary Search Tree (BST)
- AVL Tree
- Heap Tree
- Threaded Binary Tree

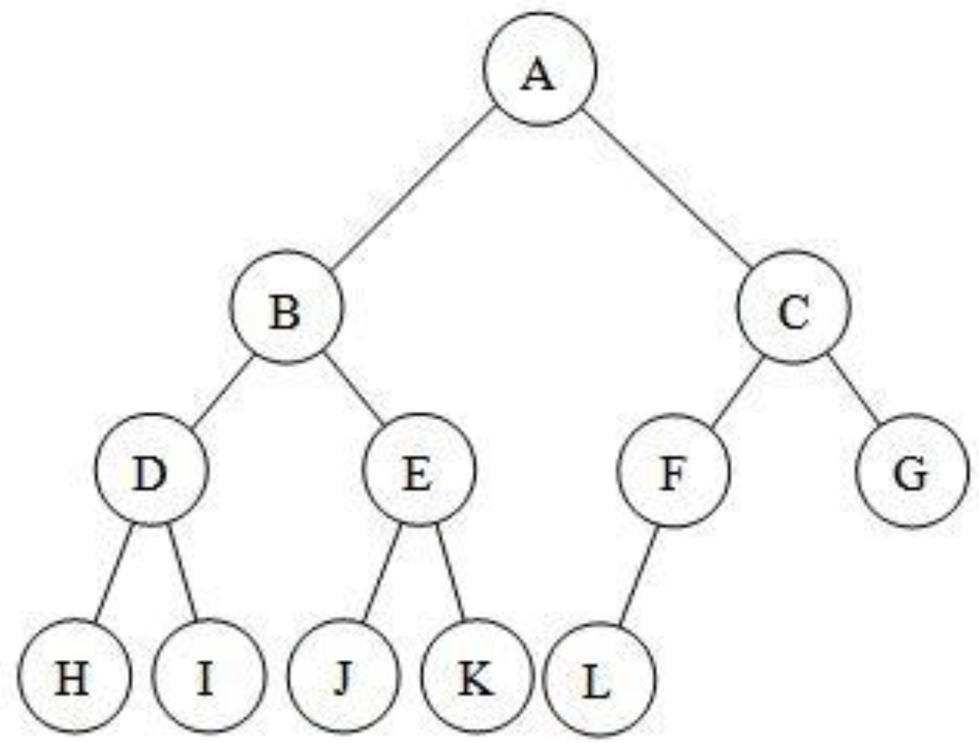
Full /Strict Binary Tree

- A full binary tree (sometimes proper binary tree or 2-tree) is a tree in which every node other than the leaves has two children.

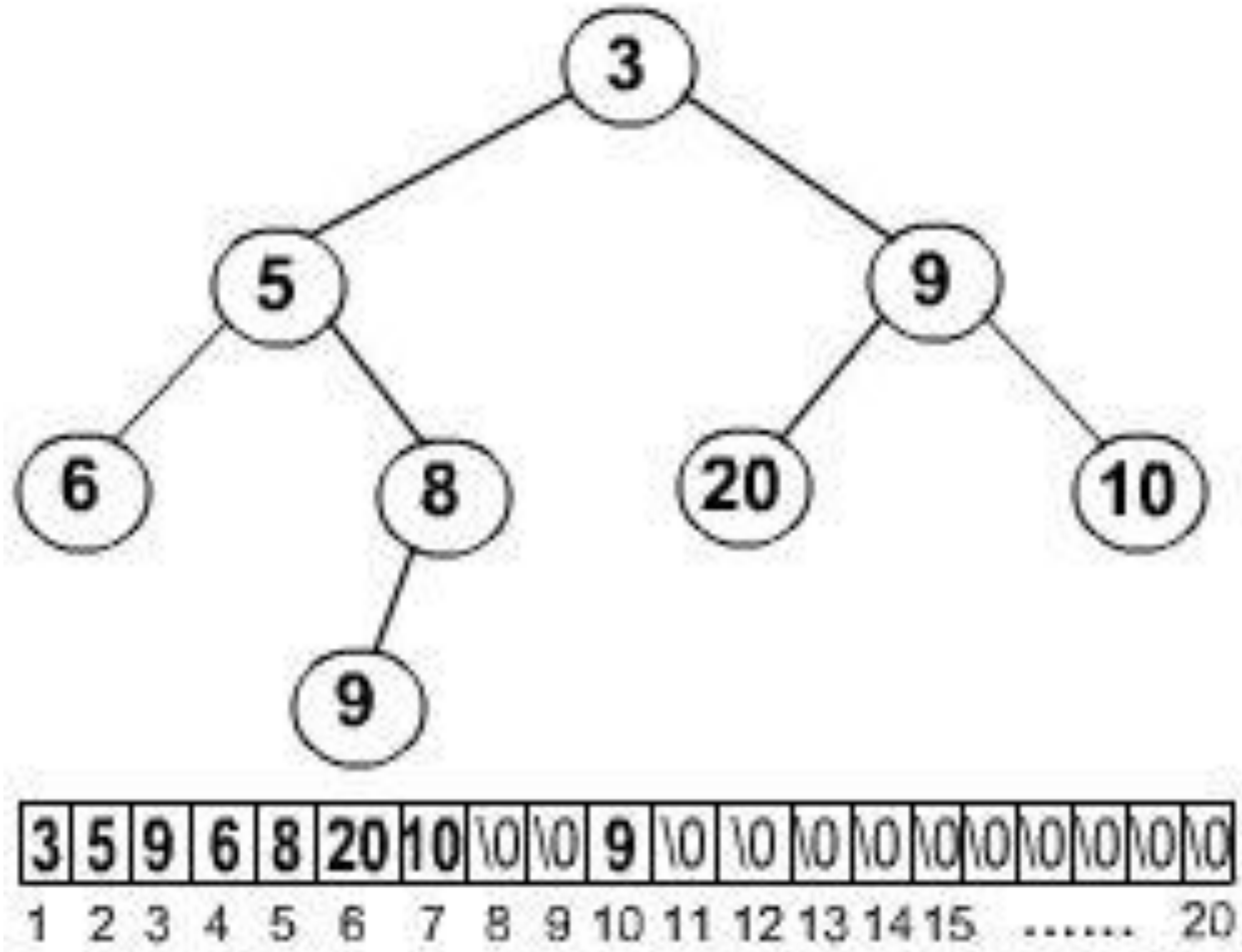


Complete Binary Tree

- A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.



Array Representation of Tree in Memory

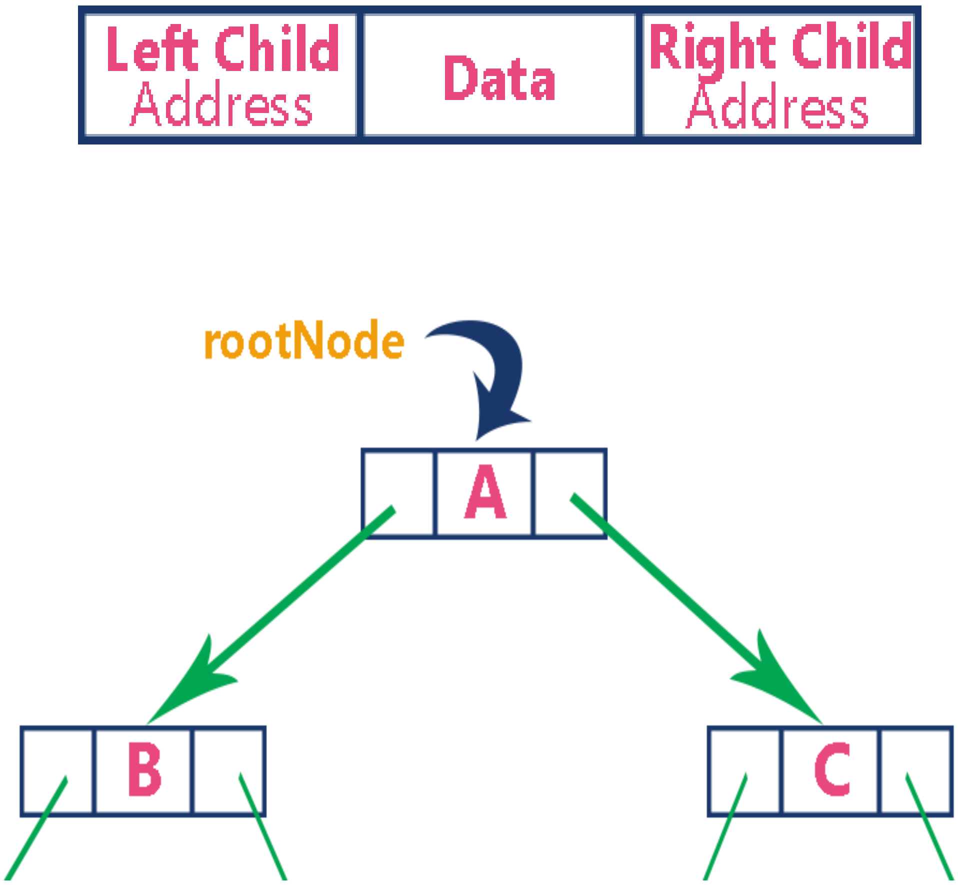


Method

- root of the tree: array position 1
- root's left child : array position 2
- root's right child: array position 3
- left child of node in array position : 2*K
- right child of node in array position : 2*K+1

Note: K is representing node number whose child is going to be inserted in array.

Linked Representation of tree in Memory



Traversing of a Binary Tree

- Pre - Order Traversal
- In - Order Traversal
- Post – Order Traversal

Pre Order	**Root**	**Left node**	**Right node**
In Order	**Left node**	**Root**	**Right node**
Post Order	**Left node**	**Right node**	**Root**

Pre Order Traversal

The algorithm starts with the root node of the tree and continues by:

1. Visiting the root node,
2. Traversing the left sub-tree, and finally
3. Traversing the right sub-tree

In Order Traversal

The algorithm starts with the root node of the tree and continues by:

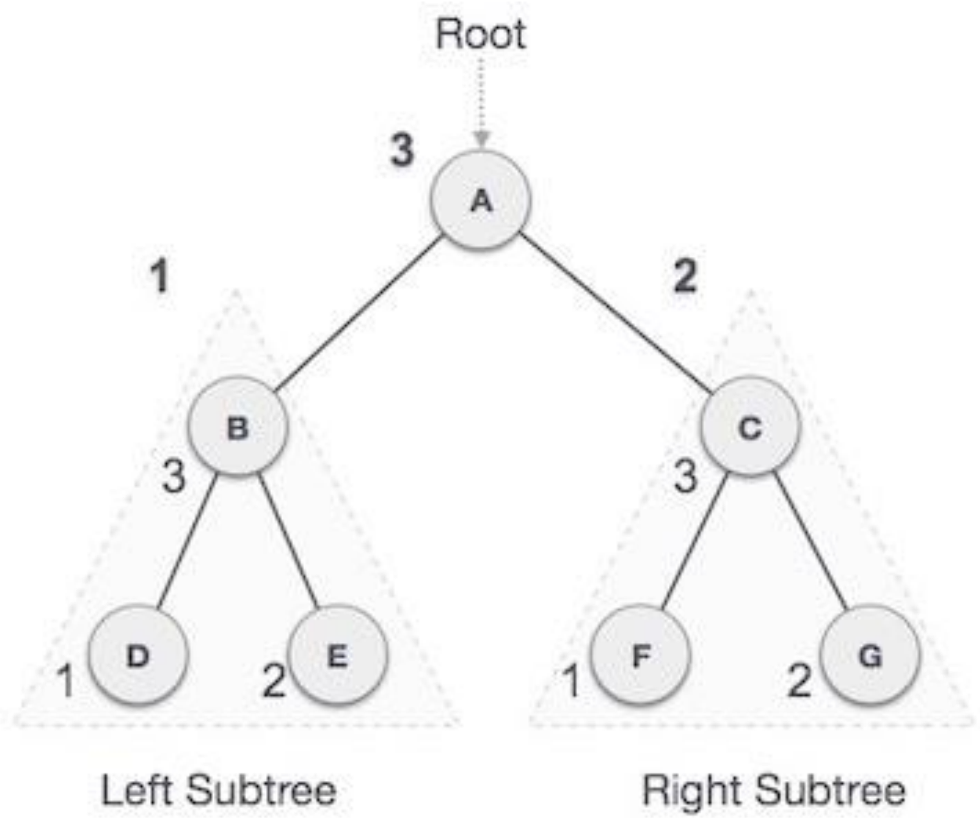
1. Traversing the left sub-tree,
2. Visiting the root node, and finally
3. Traversing the right sub-tree

Post Order Traversal

The algorithm starts with the root node of the tree and continues by:

1. Traversing the left sub-tree,
2. Traversing the right sub-tree, and finally
3. Visiting the root node

Example



- ***In Order =*** ***D → B → E → A → F → C → G***
- ***Pre Order =*** ***A → B → D → E → C → F → G***
- ***Post Order =*** ***D → E → B → F → G → C → A***

Practice Work

Construct Trees by following combinations:

- Inorder sequence: D B E A F C
  Preorder sequence: A B D E C F

- Inorder sequence: 4, 8, 2, 5, 1, 6, 3, 7
  Postorder sequence: 8, 4, 5, 2, 6, 7, 3, 1

- Inorder sequence: 4, 2, 5, 1, 6, 7, 3, 8
  Preorder sequence: 1, 2, 4, 5, 3, 7, 6, 8

- Preorder sequence: 1, 2, 4, 8, 9, 5, 3, 6, 7 **(Note: In this case Tree**
  Postorder sequence: 8, 9, 4, 5, 2, 6, 7, 3, 1 **must be full binary tree)**

Practice Work

- Given post order sequence
  1 3 2 6 9 8 5

Find out in order and pre order sequences.

Note: Here tree must be binary search tree.

Any Queries
????