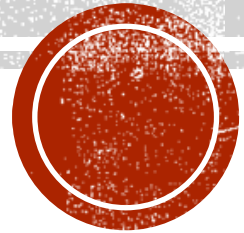
DATA STRUCTURES AND ALGORITHMS



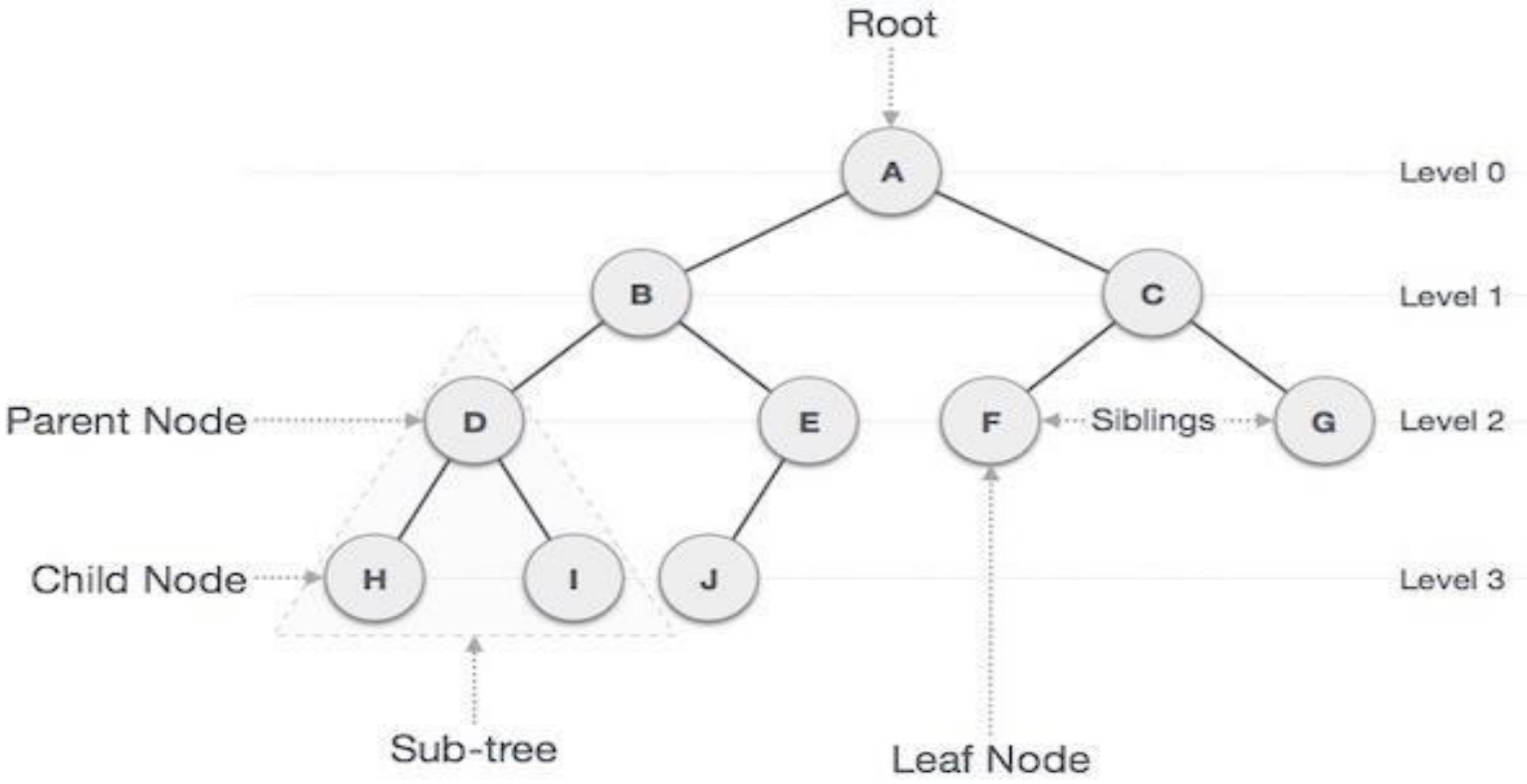
LINEAR AND NON-LINEAR DATA STRUCTURES

- Data structure where data elements are arranged sequentially or linearly where each and every element is attached to its previous and next adjacent is called a linear data structure.
- In linear data structure, single level is involved. Therefore, we can traverse all the elements in single run only. Linear data structures are easy to implement because computer memory is arranged in a linear way. Its examples are array, stack, queue, linked list, etc.
- Data structures where data elements are not arranged sequentially or linearly are called non-linear data structures.
- In a non-linear data structure, single level is not involved. Therefore, we can't traverse all the elements in single run only.
- Non-linear data structures are not easy to implement in comparison to linear data structure. It utilizes computer memory efficiently in comparison to a linear data structure. Its examples are trees and graphs.

TREES IN DATA STRUCTURES

- A tree is a non-linear abstract data type with a hierarchy-based structure. It consists of nodes (where the data is stored) that are connected via links
- Nodes represent value and nodes are connected by edges
- The tree data structure stems from a single node called a root node and has subtrees connected to the root
- A tree has the following properties:
  1. The tree has one node called root. The tree originates from this, and hence it does not have any parent
  2. Each node has one parent only but can have multiple children
  3. Each node is connected to its children via edge

TREE IN DATA STRUCTURES



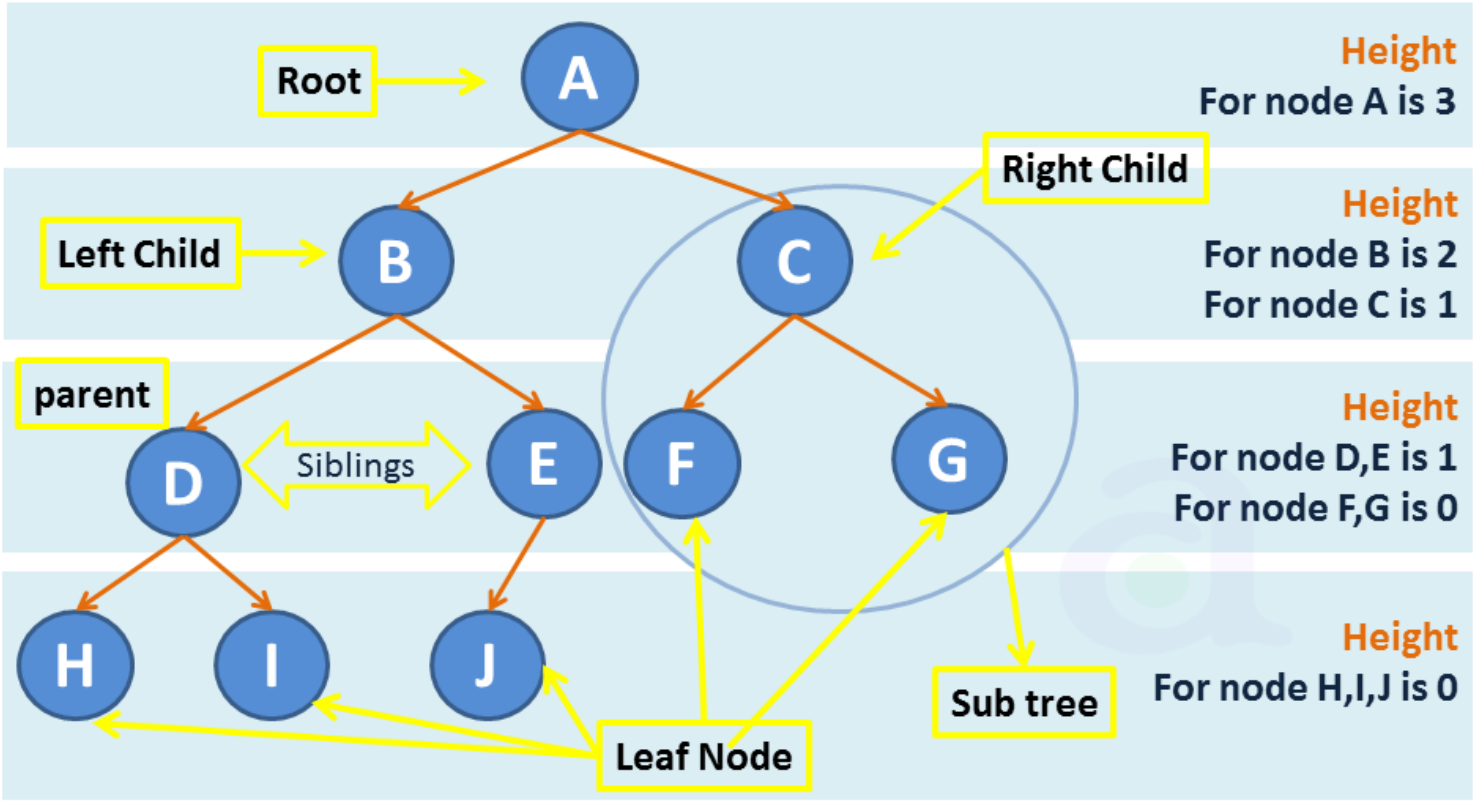
IMPORTANT TERMS RELATED TO TREES

- **Root** – The node at the top of the tree. There is only one root per tree
- **Path** – Sequence of nodes along the edges of a tree
- **Parent** – Any node except the root node has one edge upward to a node
- **Child** – The node below a given node connected by its edge downward
- **Siblings** - Nodes with the same parent
- **Leaf** – The node which does not have any child 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

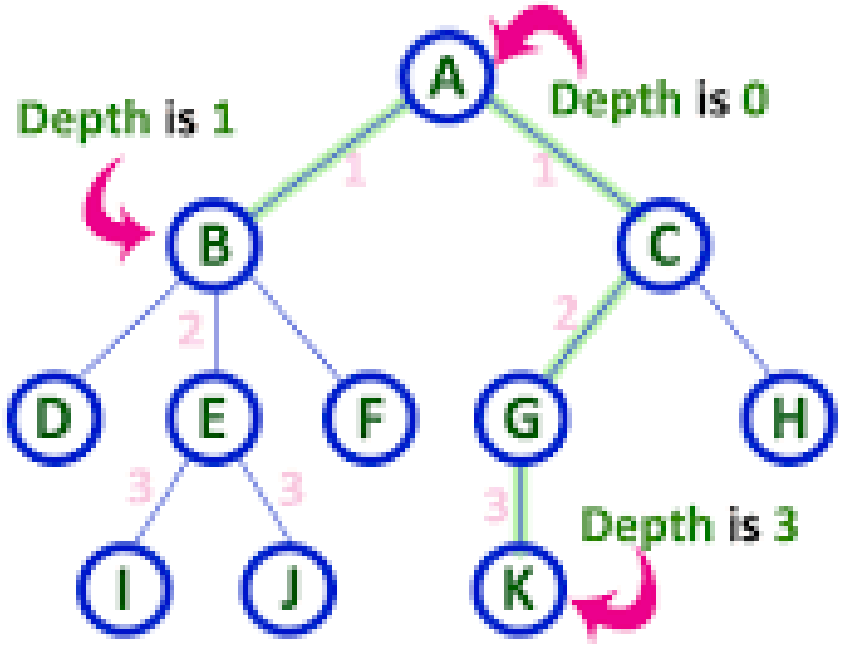
IMPORTANT TERMS RELATED TO TREES

- **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.
- **Degree of a node** - The number of children of a node
- **Height of a node** – The number of edges on the longest path between that node and a leaf
- **Depth of a node** – The number of edges from root node to that node
- **Height of Tree** – Height of root node
- **Depth of Tree** – Total number of edges from the root node to the leaf node in the longest path

HEIGHT OF NODE/ TREE



DEPTH OF NODE/ TREE



Here Depth of tree is 3

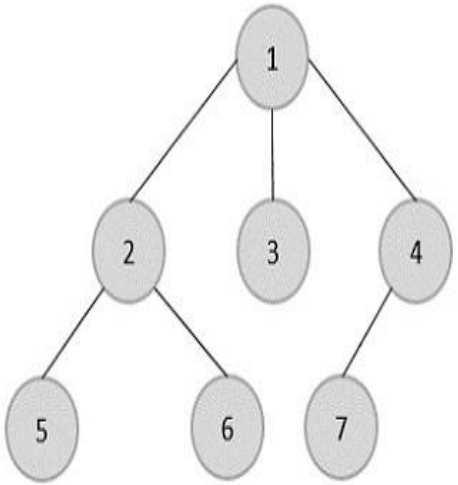
- In any tree, 'Depth of Node' is total number of Edges from root to that node.
- In any tree, 'Depth of Tree' is total number of edges from root to leaf in the longest path.

TYPES OF TREES

- There are three types of trees –
  - General Trees
  - Binary Trees
  - Binary Search Trees

GENERAL TREES

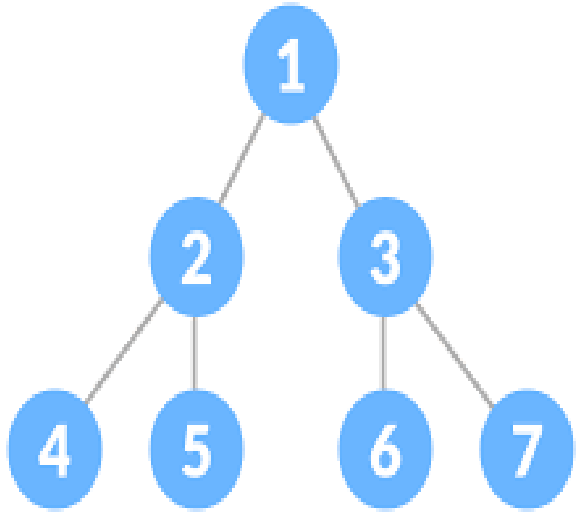
- General trees are unordered tree data structures
- The root node has minimum 0 or maximum ‘n’ subtrees
- No constraint placed on their hierarchy
- The root node acts like the superset of all the other subtrees



General Tree Data Structure

BINARY TREES

- Binary Tree is defined as a tree data structure where each node has at most 2 children.
- Since each element in a binary tree can have only 2 children, we typically name them the left and right child.



BINARY SEARCH TREES

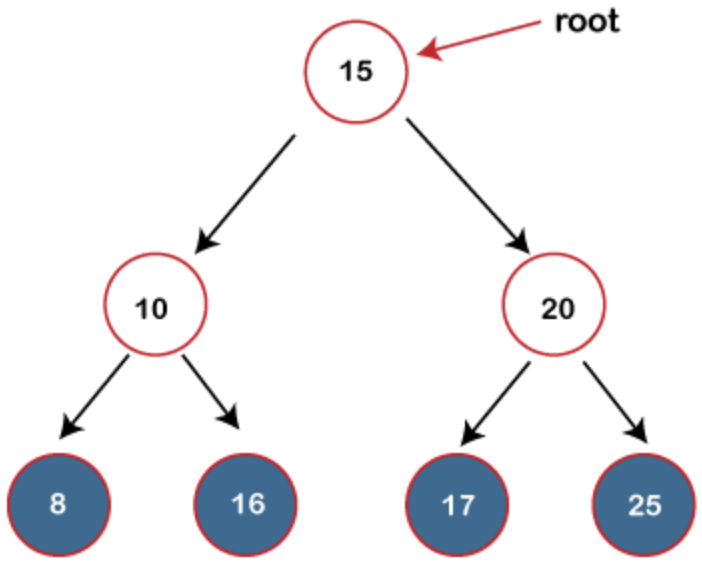
- The Binary Search Tree is a node-based, non-linear type of data structure that is derived from the binary tree.
- It is also abbreviated as the BST.
- Data retrieval, classification, and analysis can all be accomplished with its help.
- Due to the fact that its nodes are placed in a specific order, another name for this structure is the Ordered Binary Tree.

PROPERTIES OF BST

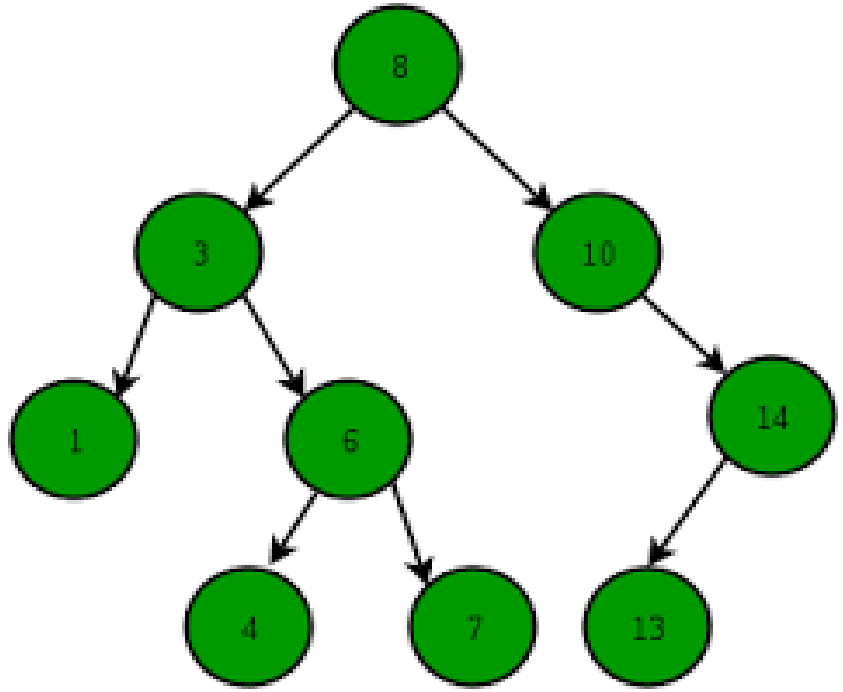
- It is necessary for the right subtree and the left subtree to each be a binary search tree.
- Only nodes with keys that are higher than the keys of the node itself can be found in the right subtree of a node.
- The left subtree of a node will only ever contain nodes with keys that are lower than the key of the node itself.
- In a Binary Search Tree, duplicate nodes are not permitted under any circumstances.
- Every node has a unique key.

DIFFERENCE BETWEEN BT AND BST

BT



BST



CONSTRUCT A BST

- Given the data, take first value to be inserted as root node value
- After insertion of Root node, take next value to be inserted
- Compare the value with root node value
- If the value is less than the root node, insert the new value to the left
- If the value is greater than the root node, insert the new value to the right

CONSTRUCT A BST

- Construct a BST of following data:

    a) 11, 6, 8, 19, 4, 10, 5, 17, 43, 49, 31

CONSTRUCT A BST

- Construct a BST of following data:

  70, 50, 25, 90, 10, 101, 85, 62, 120, 55, 77, 20, 110

  i. What is height and depth of tree in part b?

  ii. What is height and depth of node having value 90?

  iii. What is height depth of node having value 20?

  iv. Write all the leaf nodes of the tree.