RGPVNOTES.IN

Program : **B.Tech**

Subject Name: **Data Structure**

Subject Code: **CS-303**

Semester: **3rd**

CLASS NOTES SUBJECT DATA STRUCTURE
UNIT-3 TREE

Tree: Definitions - Height, depth, order, degree etc. Binary Search Tree - Operations, Traversal, Search. AVL Tree, Heap, Applications and comparison of various types of tree; Introduction to forest, multi-way Tree, B tree, B+ tree, B* tree and red-black tree

Tree: Definitions

In linear data structure, data is organized in sequential order and in non-linear data structure, data is organized in random order. Tree is a very popular data structure used in wide range of applications. A tree data structure can be defined as follows:-
Tree is a non-linear data structure which organizes data in hierarchical structure and this is a recursive definition.
In tree data structure, every individual element is called as **Node**. Node in a tree data structure, stores the actual data of that particular element and link to next element in hierarchical structure.
In a tree data structure, if we have **N** number of nodes then we can have a maximum of **N-1** number of links.

Example

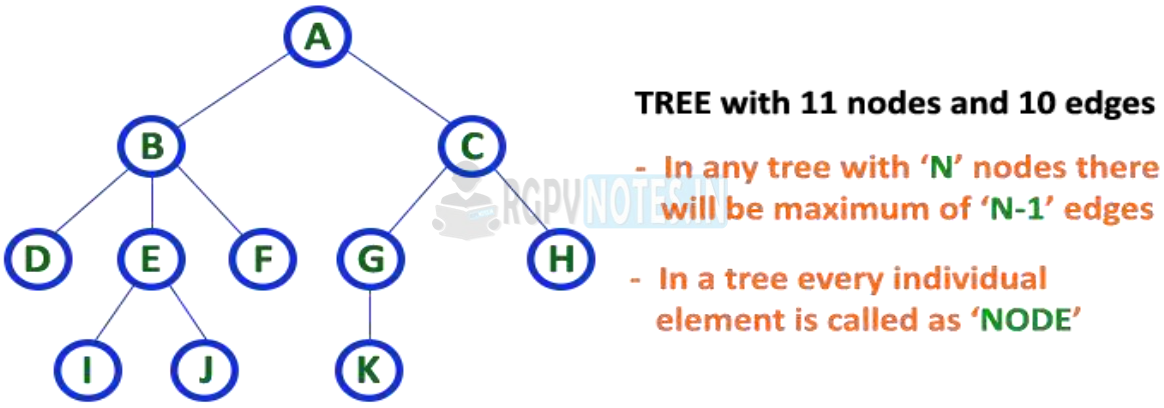


Figure 3.1 Tree

In a tree data structure, we use the following terminology:-

1. Root

In a tree data structure, the first node is called as Root Node. Every tree must have root node. We can say that root node is the origin of tree data structure. In any tree, there must be only one root node. We never have multiple root nodes in a tree.

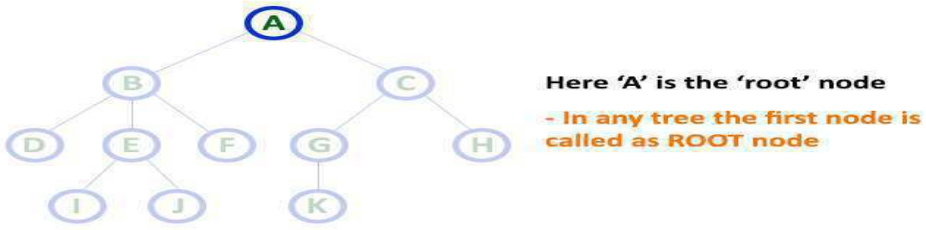


Figure 3.2 Root Node

2. Edge

In a tree data structure, the connecting link between any two nodes is called as EDGE. In a tree with 'N' number of nodes there will be a maximum of 'N-1' number of edges.

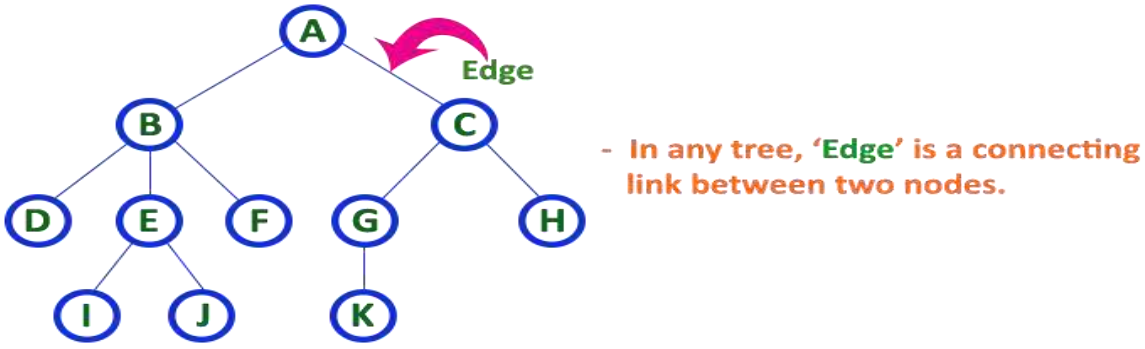


Figure 3.3 Edge

3. Parent

In a tree data structure, the node which is predecessor of any node is called as PARENT NODE. In simple words, the node which has branch from it to any other node is called as parent node. Parent node can also be defined as "The node which has child / children".

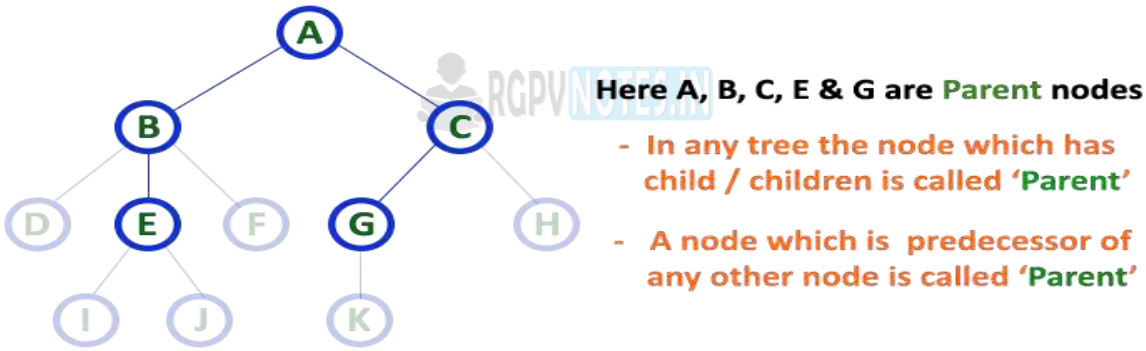


Figure 3.4 Parent Node

4. Child

In a tree data structure, the node which is descendant of any node is called as CHILD Node. In simple words, the node which has a link from its parent node is called as child node. In a tree, any parent node can have any number of child nodes. In a tree, all the nodes except root are child nodes.

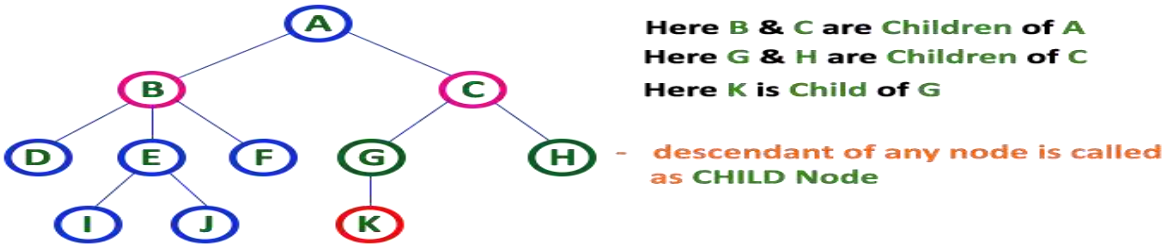


Figure 3.5 Child Node

5. Siblings

In a tree data structure, nodes which belong to same Parent are called as SIBLINGS. In simple words, the nodes with same parent are called as Sibling nodes.

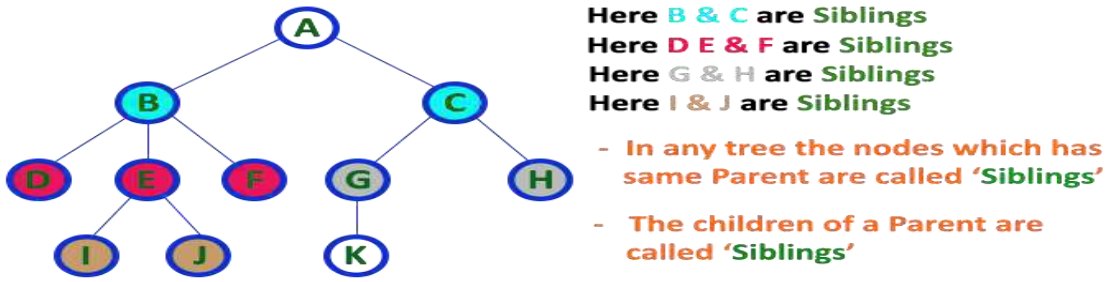


Figure 3.6 Siblings

6. Leaf

In a tree data structure, the node which does not have a child is called as LEAF Node. In simple words, a leaf is a node with no child.
In a tree data structure, the leaf nodes are also called as External Nodes. External node is also a node with no child. In a tree, leaf node is also called as 'Terminal' node.

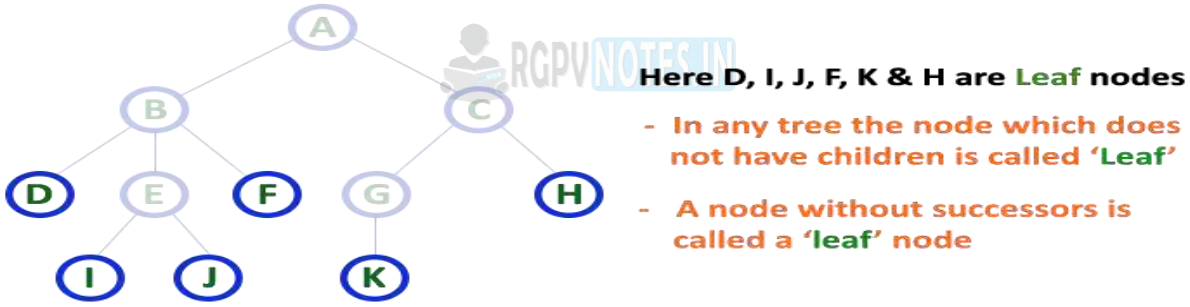


Figure 3.7 Leaf Node

7. Internal Nodes

In a tree data structure, the node which has atleast one child is called as INTERNAL Node. In simple words, an internal node is a node with atleast one child.In a tree data structure, nodes other than leaf nodes are called as Internal Nodes. The root node is also said to be Internal Node if the tree has more than one node. Internal nodes are also called as 'Non-Terminal' nodes.

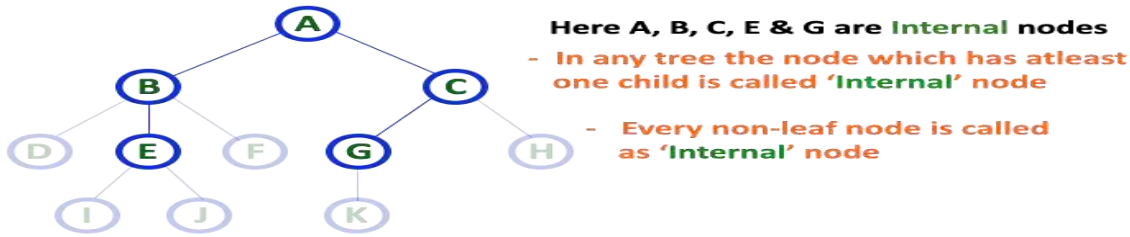


Figure 3.8 Internal Nodes

8. Degree

In a tree data structure, the total number of children of a node is called as DEGREE of that Node. In simple words, the Degree of a node is total number of children it has. The highest degree of a node among all the nodes in a tree is called as 'Degree of Tree'

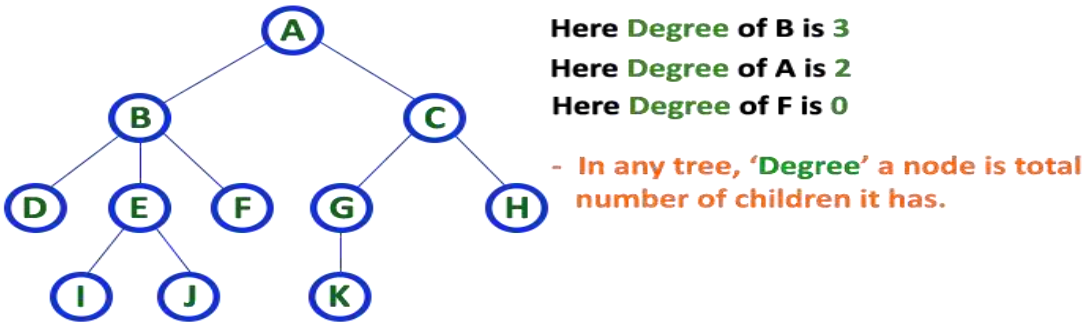


Figure 3.9 Degree

9. Level

In a tree data structure, the root node is said to be at Level 0 and the children of root node are at Level 1 and the children of the nodes which are at Level 1 will be at Level 2 and so on... In simple words, in a tree each step from top to bottom is called as a Level and the Level count starts with '0' and incremented by one at each level (Step).

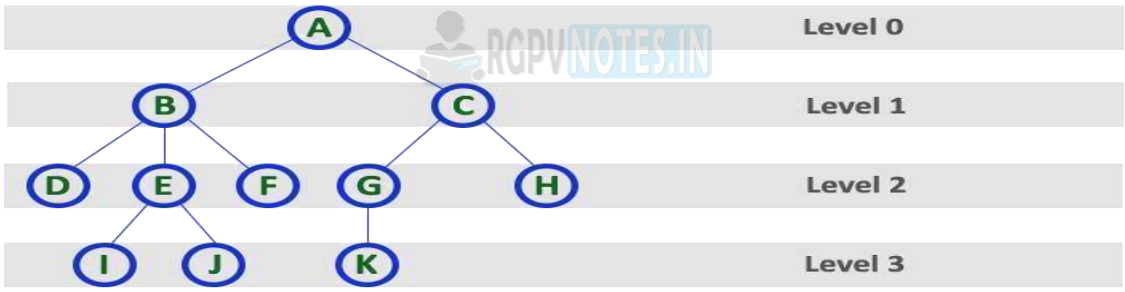


Figure 3.10 Level of Tree

10. Height

In a tree data structure, the total number of egdes from leaf node to a particular node in the longest path is called as HEIGHT of that Node. In a tree, height of the root node is said to be height of the tree. In a tree, height of all leaf nodes is '0'.

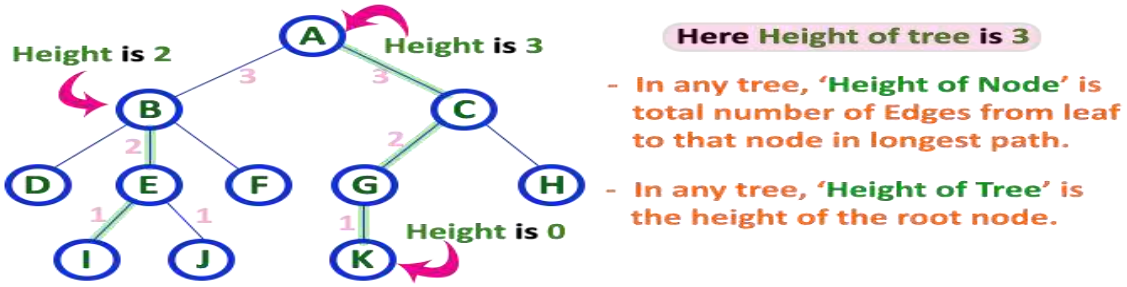


Figure 3.11 Height Of Tree

11. Depth

In a tree data structure, the total number of egdes from root node to a particular node is called as DEPTH of that Node. In a tree, the total number of edges from root node to a leaf node in the longest path is said to be Depth of the tree. In simple words, the highest depth of any leaf node in a tree is said to be depth of that tree. In a tree, depth of the root node is '0'.

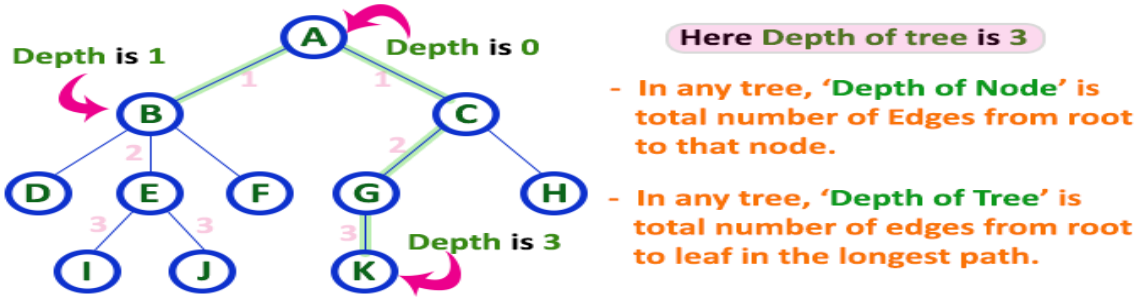


Figure 3.12 Depth Of Tree

12. Path

In a tree data structure, the sequence of Nodes and Edges from one node to another node is called as PATH between that two Nodes. Length of a Path is total number of nodes in that path. In below example the path A - B - E - J has length 4.

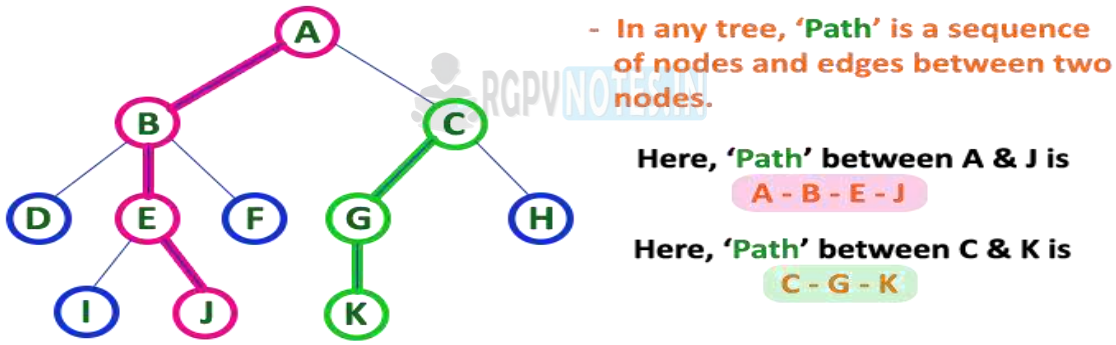


Figure 3.13 Path

13. Sub Tree

In a tree data structure, each child from a node forms a subtree recursively. Every child node will form a subtree on its parent node.

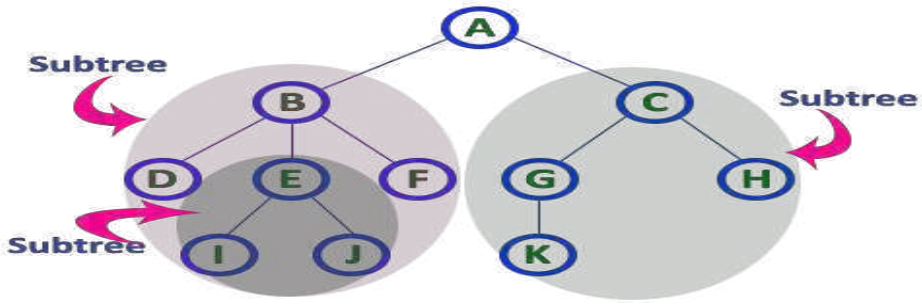


Figure 3.14 Sub Tree

Binary Search Tree - Operations, Search

In a binary tree, every node can have maximum of two children but there is no order of nodes based on their values. In binary tree, the elements are arranged as they arrive to the tree, from top to bottom and left to right. To enhance the performance of binary tree, we use special type of binary tree known as **Binary Search Tree**. Binary search tree mainly focus on the search operation in binary tree. Binary search tree can be defined as follows:-
Binary Search Tree is a binary tree in which every node contains only smaller values in its left subtree and only larger values in its right subtree.

Example

The following tree is a Binary Search Tree. In this tree, left subtree of every node contains nodes with smaller values and right subtree of every node contains larger values.

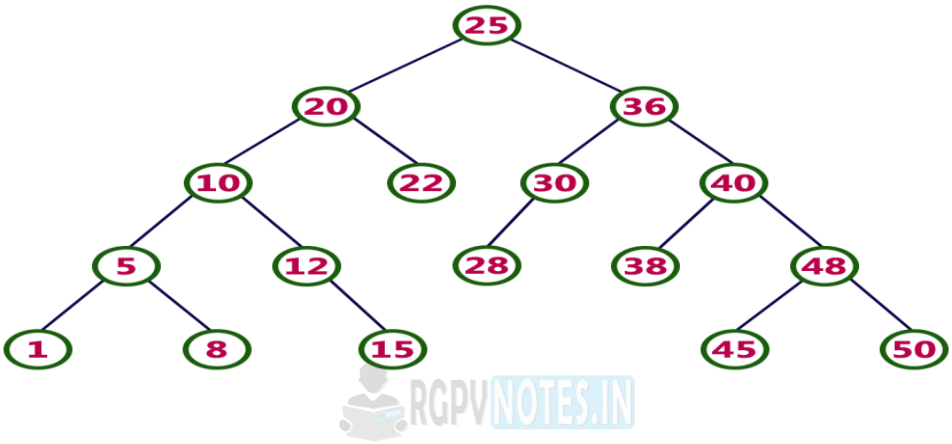


Figure 3.15 Binary Search Tree

The following operations are performed on a binary search tree:-1) Search 2) Insertion 3) Deletion

Search Operation in BST

In a binary search tree, the search operation is performed with O(log n) time complexity. The search operation is performed as follows:-

- Step 1: Read the search element from the user
- Step 2: Compare, the search element with the value of root node in the tree.
- Step 3: If both are matching, then display "Given node found!!!" and terminate the function
- Step 4: If both are not matching, then check whether search element is smaller or larger than that node value.
- Step 5: If search element is smaller, then continue the search process in left subtree.
- Step 6: If search element is larger, then continue the search process in right subtree.
- Step 7: Repeat the same until we found exact element or we completed with a leaf node
- Step 8: If we reach to the node with search value, then display "Element is found" and terminate the function.
- Step 9: If we reach to a leaf node and it is also not matching, then display "Element not found" and terminate the function.

Insertion Operation in BST

In a binary search tree, the insertion operation is performed with O(log n) time complexity. In binary search tree, new node is always inserted as a leaf node. The insertion operation is performed as follows:-
Step 1: Create a newNode with given value and set its left and right to NULL.
Step 2: Check whether tree is Empty.
Step 3: If the tree is Empty, then set set root to newNode.
Step 4: If the tree is Not Empty, then check whether value of newNode is smaller or larger than the node (here it is root node).
Step 5: If newNode is smaller than or equal to the node, then move to its left child. If newNode is larger than the node, then move to its right child.
Step 6: Repeat the above step until we reach to a leaf node (e.i., reach to NULL).
Step 7: After reaching a leaf node, then isert the newNode as left child if newNode is smaller or equal to that leaf else insert it as right child.

Deletion Operation in BST

In a binary search tree, the deletion operation is performed with O(log n) time complexity. Deleting a node from Binary search tree has following three cases:-

Case 1: Deleting a leaf node
We use the following steps to delete a leaf node from BST:-
Step 1: Find the node to be deleted using search operation
Step 2: Delete the node using free function (If it is a leaf) and terminate the function.

Case 2: Deleting a node with one child
We use the following steps to delete a node with one child from BST:-
Step 1: Find the node to be deleted using search operation
Step 2: If it has only one child, then create a link between its parent and child nodes.
Step 3: Delete the node using free function and terminate the function.

Case 3: Deleting a node with two children
We use the following steps to delete a node with two children from BST:-
Step 1: Find the node to be deleted using search operation
Step 2: If it has two children, then find the largest node in its left subtree (OR) the smallest node in its right subtree.
Step 3: Swap both deleting node and node which found in above step.
Step 4: Then, check whether deleting node came to case 1 or case 2 else goto steps 2
Step 5: If it comes to case 1, then delete using case 1 logic.
Step 6: If it comes to case 2, then delete using case 2 logic.
Step 7: Repeat the same process until node is deleted from the tree.

Binary Search Tree – Traversal

There are three types of binary search tree traversals.

1. **In - Order Traversal**
2. **Pre - Order Traversal**
3. **Post - Order Traversal**

1. In - Order Traversal (leftChild - root - rightChild)

In In-Order traversal, the root node is visited between left child and right child. In this traversal, the left child node is visited first, then the root node is visited and later we go for visiting right child node. This in-order

traversal is applicable for every root node of all subtrees in the tree. This is performed recursively for all nodes in the tree.

2. Pre - Order Traversal (root - leftChild - rightChild)

In Pre-Order traversal, the root node is visited before left child and right child nodes. In this traversal, the root node is visited first, then its left child and later its right child. This pre-order traversal is applicable for every root node of all subtrees in the tree.

3. Post - Order Traversal (leftChild - rightChild - root)

In Post-Order traversal, the root node is visited after left child and right child. In this traversal, left child node is visited first, then its right child and then its root node. This is recursively performed until the right most node is visited.

AVL Tree

AVL tree is a self balanced binary search tree. That means, an AVL tree is also a binary search tree but it is a balanced tree. A binary tree is said to be balanced, if the difference between the hieghts of left and right subtrees of every node in the tree is either -1, 0 or +1. In other words, a binary tree is said to be balanced if for every node, height of its children differ by at most one. In an AVL tree, every node maintains a extra information known as **balance factor**. The AVL tree was introduced in the year of 1962 by G.M. Adelson-Velsky and E.M. Landis.

An AVL tree is defined as follows:-An AVL tree is a balanced binary search tree. In an AVL tree, balance factor of every node is either -1, 0 or +1.Balance factor of a node is the difference between the heights of left and right subtrees of that node. The balance factor of a node is calculated either **height of left subtree - height of right subtree** (OR) **height of right subtree - height of left subtree**.

There are four rotations and they are classified into two types.

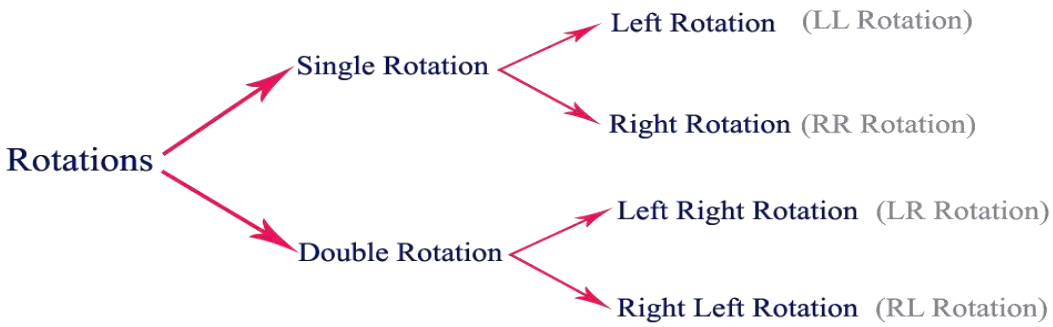


Figure 3.16 Rotation AVL Tree

The following operations are performed on an AVL tree:-

1. Search
2. Insertion
3. Deletion

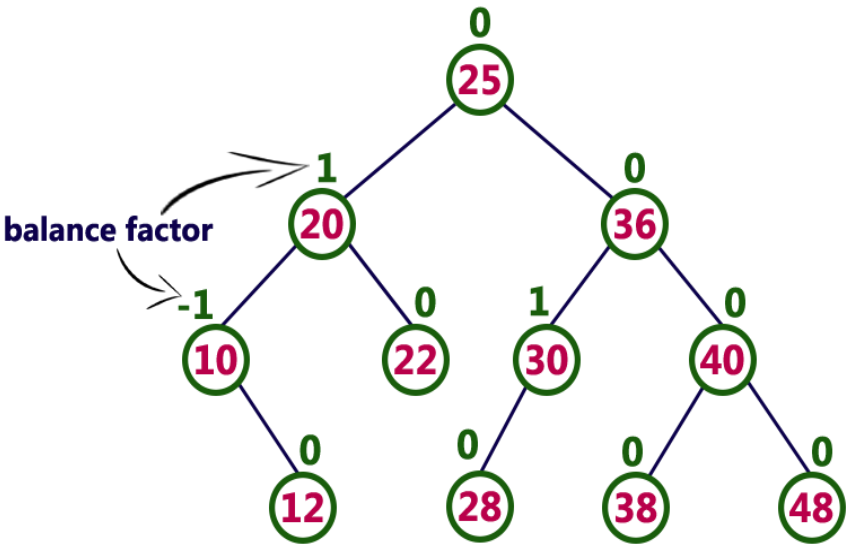


Figure 3.17 AVL Tree

The above tree is a binary search tree and every node is satisfying balance factor condition. So this tree is said to be an AVL tree.

Heap

Heap data structure is a specialized binary tree based data structure. Heap is a binary tree with special characteristics. In a heap data structure, nodes are arranged based on thier value. A heap data structure, some time called as Binary Heap.

There are two types of heap data structures and they are as follows...

1. Max Heap
2. Min Heap

Every heap data structure has the following properties...
Property #1 (Ordering): Nodes must be arranged in a order according to values based on Max heap or Min heap.
Property #2 (Structural): All levels in a heap must full, except last level and nodes must be filled from left to right strictly.
Max Heap
Max heap data structure is a specialized full binary tree data structure except last leaf node can be alone. In a max heap nodes are arranged based on node value.

Max heap is defined as follows:- Max heap is a specialized full binary tree in which every parent node contains greater or equal value than its child nodes. And last leaf node can be alone.

Min-Heap − Where the value of the root node is less than or equal to either of its children.

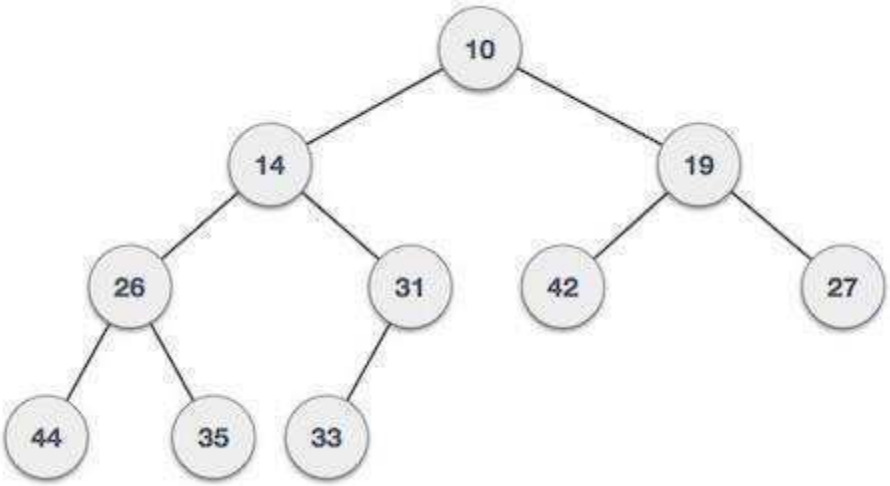


Figure 3.18 Min Heap

Max-Heap − Where the value of the root node is greater than or equal to either of its children.

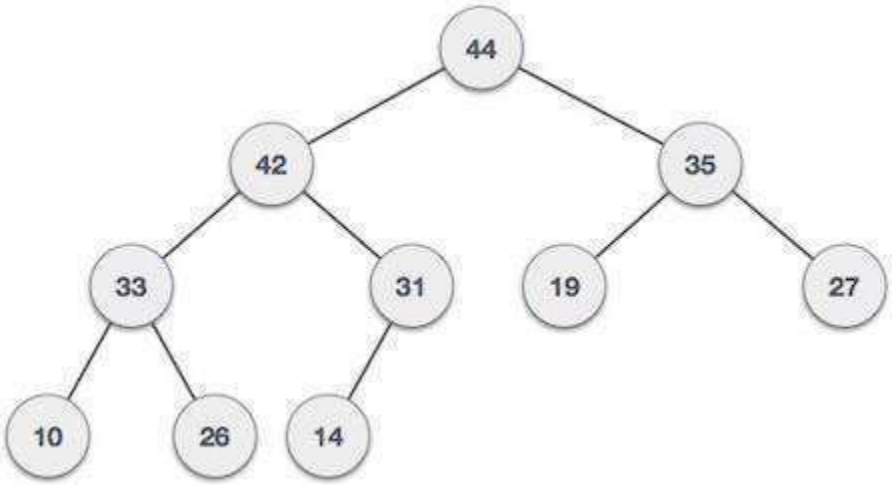


Figure 3.19 Max Heap

Heap application

Priority Queues: Priority queues can be efficiently implemented using Binary Heap because it supports insert(), delete() and extractmax(), decreaseKey() operations in O(logn) time. Binomoial Heap and Fibonacci Heap are variations of Binary Heap. These variations perform union also in O(logn) time which is a O(n) operation in Binary Heap. Heap Implemented priority queues are used in Graph algorithms like Prim's Algorithm and Dijkstra's algorithm.

Comparison of various types of tree

There are different types of binary trees and they are

1. Strictly Binary Tree

In a binary tree, every node can have a maximum of two children. But in strictly binary tree, every node should have exactly two children or none. That means every internal node must have exactly two children. A strictly

Binary Tree can be defined as follows:- A binary tree in which every node has either two or zero number of children is called Strictly Binary Tree

Strictly binary tree is also called as Full Binary Tree or Proper Binary Tree or 2-Tree

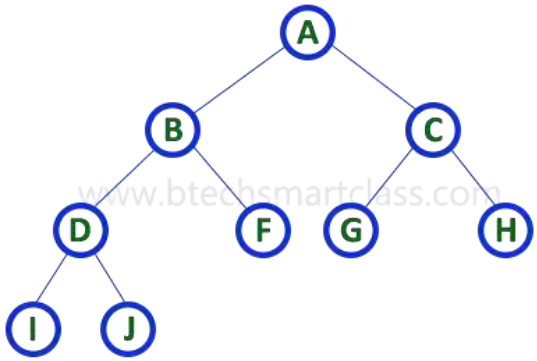


Figure 3.20 Strictly Binary Tree

2. Complete Binary Tree

In a binary tree, every node can have a maximum of two children. But in strictly binary tree, every node should have exactly two children or none and in complete binary tree all the nodes must have exactly two children and at every level of complete binary tree there must be 2level number of nodes. For example at level 2 there must be 22 = 4 nodes and at level 3 there must be 23 = 8 nodes. A binary tree in which every internal node has exactly two children and all leaf nodes are at same level is called Complete Binary Tree.
Complete binary tree is also called as Perfect Binary Tree

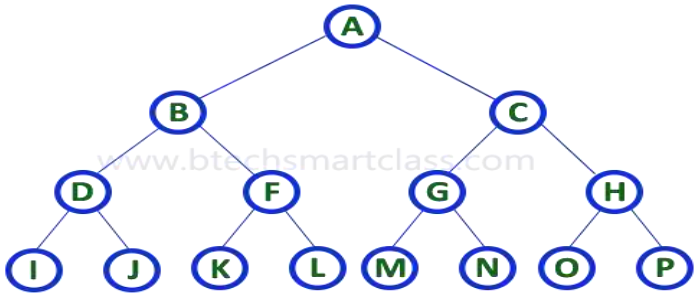


Figure 3.21 Complete Binary Tree

3. Extended Binary Tree

A binary tree can be converted into Full Binary tree by adding dummy nodes to existing nodes wherever required. The full binary tree obtained by adding dummy nodes to a binary tree is called as Extended Binary Tree.

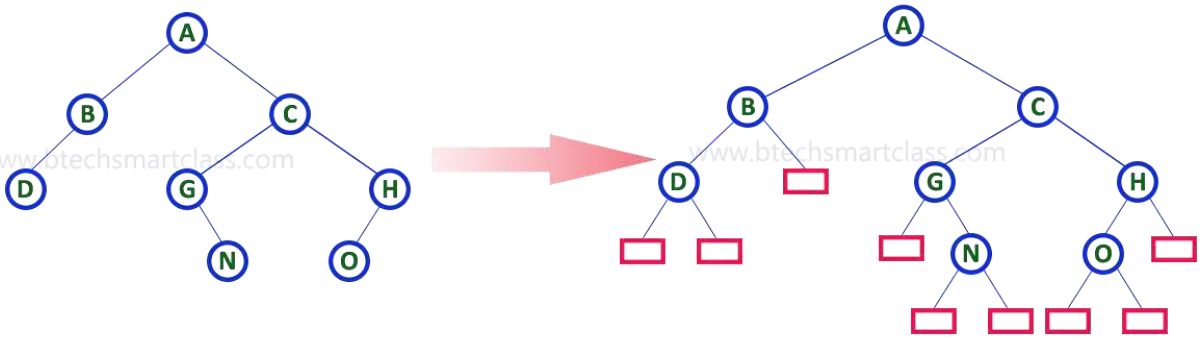


Figure 3.22 Extended Binary Tree

In above figure, a normal binary tree is converted into full binary tree by adding dummy nodes (In pink colour).

Introduction to Forest

Forest is a collection of disjoint trees. In other words, we can also say that forest is a collection of an acyclic graph which is not connected. Here is a pictorial representation of a forest.

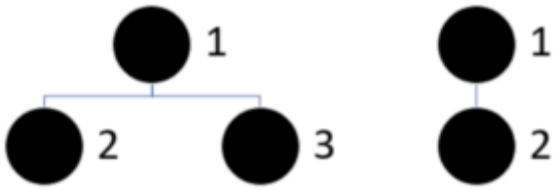


Figure 3.23 Forest

Here you can see that there is no connected tree in the example. A single tree and empty graph is also an example of forest data structure.

Uses of Forest Data Structure

Social Networking Websites

Social networking sites (Such as Facebook, LinkedIn, Twitter etc) are using tree and graphdata structure for their data representation. When you work on adding two people as a friend, you are creating a forest of two people.

Big Data and Web Scrapers

Websites are organized in the form of a tree data structure where the main page forms the root node and the subsequent hyperlinks from that page represents the rest of the tree. When Web scrapers scrap multiple such websites, they represent it in the form of a forest of several trees.

Operating System Storage

If you are working on a Windows-based operating system, you would be able to see various disks in the system such as C drive (C:\), D drive (D:\), etc. You can think of each drive as different trees and the collection of all storage as the forest.

Introduction to B & B * tree

In a binary search tree, AVL Tree, Red-Black tree etc., every node can have only one value (key) and maximum of two children but there is another type of search tree called B-Tree in which a node can store more than one value (key) and it can have more than two children. B-Tree was developed in the year of 1972 by Bayer and McCreight with the name Height Balanced m-way Search Tree. Later it was named as B-Tree.

B-Tree can be defined as follows... B-Tree is a self-balanced search tree with multiple keys in every node and more than two children for every node.

Here, number of keys in a node and number of children for a node is depend on the order of the B-Tree. Every B-Tree has order.

B-Tree of Order m has the following properties...

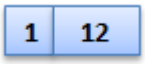
- Property #1 - All the leaf nodes must be at same level.
- Property #2 - All nodes except root must have at least [m/2]-1 keys and maximum of m-1 keys.
- Property #3 - All non leaf nodes except root (i.e. all internal nodes) must have at least m/2 children.
- Property #4 - If the root node is a non leaf node, then it must have at least 2 children.
- Property #5 - A non leaf node with n-1 keys must have n number of children.
- Property #6 - All the key values within a node must be in Ascending Order.

Construct a B-Tree of Order 5 by inserting following numbers

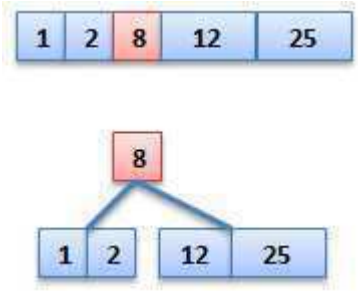
1,12,8,2,25,6,14,28,17,7,52,16,48,68,3,26,29,53,55,45,67.

1) **insert 1**

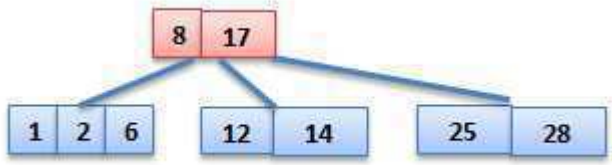
2) **insert 12**



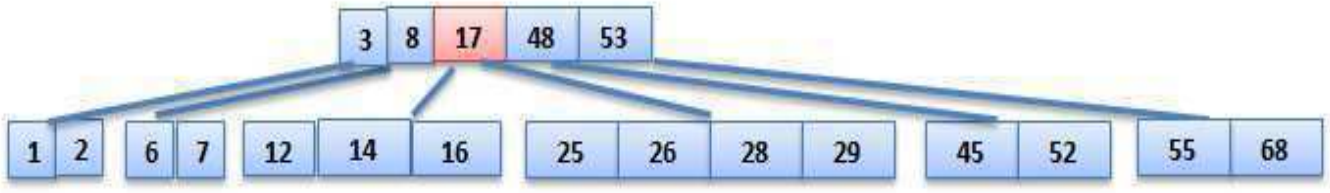
3) **insert 8,2 and 25**

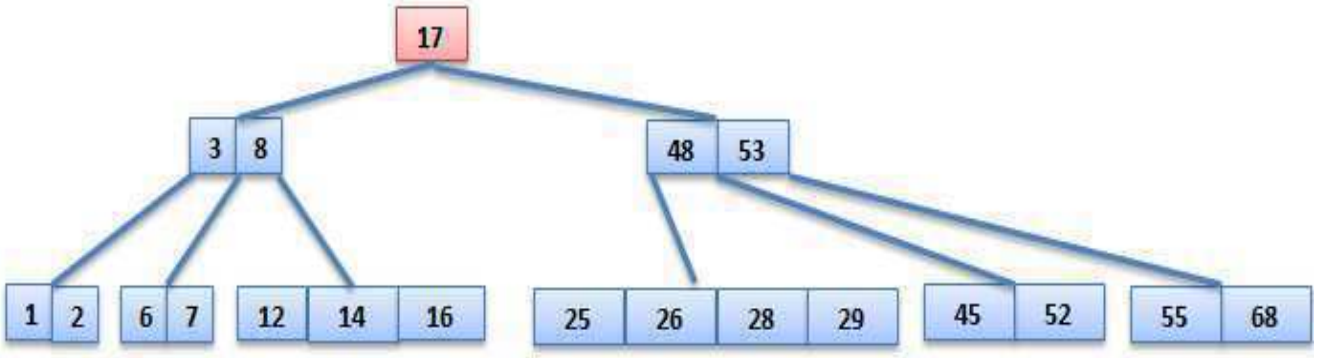


4) **insert 6,14,28,17**



5) **insert 7,52,16,48,68,3,26,29,53,55,45.**





Introduction to red-black tree

Red - Black Tree is another varient of Binary Search Tree in which every node is colored either RED or BLACK. We can define a Red Black Tree as follows:- Red Black Tree is a Binary Search Tree in which every node is colored eigther RED or BLACK.

In a Red Black Tree the color of a node is decided based on the Red Black Tree properties. Every Red Black Tree has the following properties.

Properties of Red Black Tree

- Property #1: Red - Black Tree must be a Binary Search Tree.
- Property #2: The ROOT node must colored BLACK.
- Property #3: The children of Red colored node must colored BLACK. (There should not be two consecutive RED nodes).
- Property #4: In all the paths of the tree there must be same number of BLACK colored nodes.
- Property #5: Every new node must inserted with RED color.
- Property #6: Every leaf (e.i. NULL node) must colored BLACK

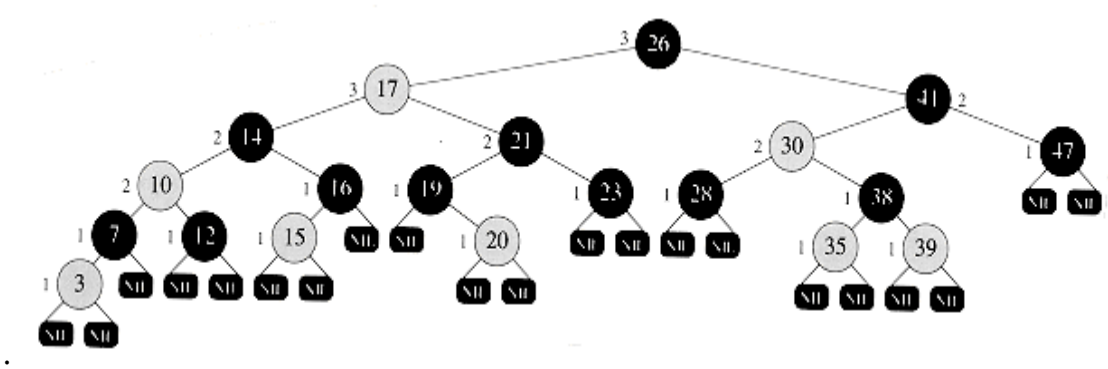


Figure 3.24 Red Black Tree

Introduction to B+ tree

A B+ tree is a data structure often used in the implementation of database indexes. Each node of the tree contains an ordered list of keys and pointers to lower level nodes in the tree. These pointers can be thought of as being between each of the keys. To search for or insert an element into the tree, one loads up the root node, finds the adjacent keys that the searched-for value is between, and follows the corresponding pointer to the next node in the tree. Recursing eventually leads to the desired value or the conclusion that the value is not present.

B+ trees use some clever balancing techniques to make sure that all of the leaves are always on the same level of the tree, that each node is always at least half full (rounded) of keys, and (therefore) that the height of the tree is always at most ceiling(log(k)) of base ceiling(n/2) where k is the number of values in the tree and n is the maximum number of pointers (= maximum number of nodes + 1) in each block. This means that only a small number of pointer traversals is necessary to search for a value if the number of keys in a node is large. This is crucial in a database because the B+ tree is on disk.

Properties of a B+ Tree

The root node points to at least two nodes.

1. All non-root nodes are at least half full.
2. For a tree of order m, all internal nodes have m-1 keys and m pointers.
3. A B+-Tree grows upwards.
4. A B+-Tree is balanced.
5. Sibling pointers allow sequential searching.

Introduction to multi-way Tree

A multiway tree is a tree that can have more than two children
A multiway tree of order m88 (or an **m-way tree) is one in which a tree can have m children.
An m-way search tree is a m-way tree in which:

- Each node has m children and m-1 key fields
- The keys in each node are in ascending order.
- The keys in the first i children are smaller than the ithith key
- The keys in the last m-i children are larger than the ithith key
- In a binary search tree, m=2. So it has one value and two sub trees.
- The figure above is a m-way search tree of order 3.
- M-way search trees give the same advantages to m-way trees that binary search trees gave to binary trees - they provide fast information retrieval and update.
- However, they also have the same problems that binary search trees had - they can become unbalanced, which means that the construction of the tree becomes of vital importance.
- In m-way search tree, each sub-tree is also a m-way search tree and follows the same rules.
- An extension of a multiway search tree of order m is a B-tree of order m.
- This type of tree will be used when the data to be accessed/stored is located on secondary storage devices because they allow for large amounts of data to be stored in a node.

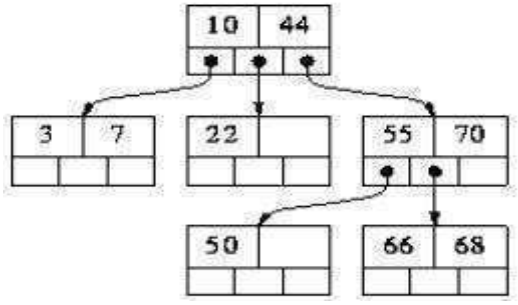


Figure 3.25 Multi Way Tree

RGPVNOTES.IN

We hope you find these notes useful.

You can get previous year question papers at https://qp.rgpvnotes.in .

If you have any queries or you want to submit your study notes please write us at rgpvnotes.in@gmail.com

LIKE & FOLLOW US ON FACEBOOK
facebook.com/rgpvnotes.in