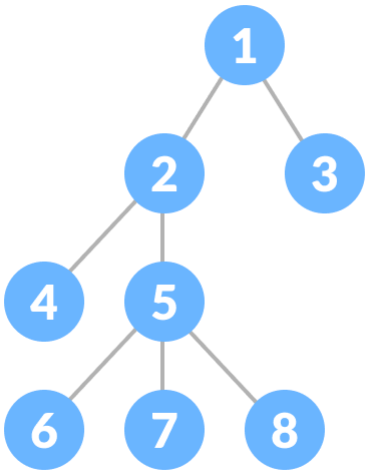
Tree Data Structure

A tree is a **nonlinear hierarchical data structure** that **consists of nodes connected by edges.**



Tree Terminologies

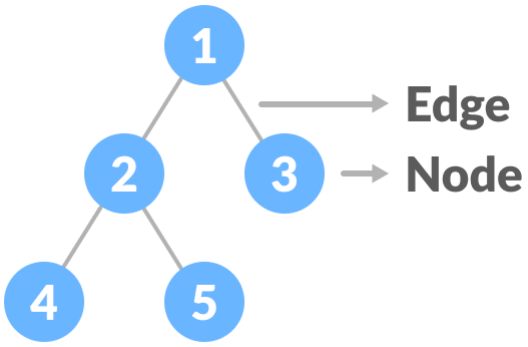
Node

A node is an entity that contains a key or value and pointers to its child nodes.

The last nodes of each path are called **leaf nodes or external nodes** that do not contain a link/pointer to child nodes.
The node having at least a child node is called an **internal node**.

Edge

It is the link between any two nodes.



Nodes and edges of a tree

Root

It is the topmost node of a tree.

Height of a Node

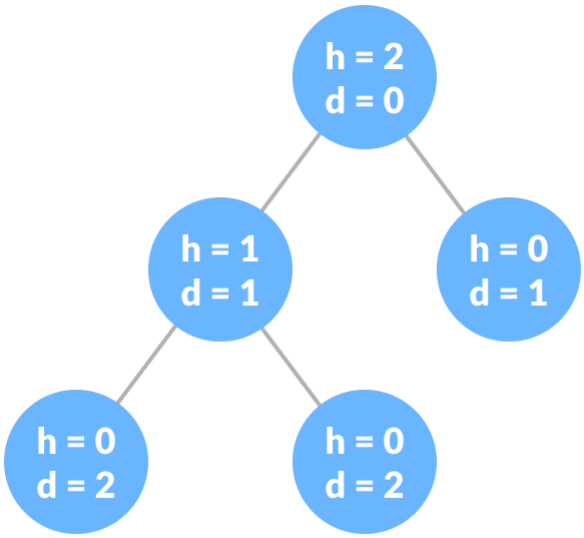
The height of a node **is the number of edges from the node to the deepest leaf** (i.e., the longest path from the node to a leaf node).

Depth of a Node

The depth of a node **is the number of edges from the root to the node.**

Height of a Tree

The height of a Tree is the height of the root node or the depth of the deepest node.



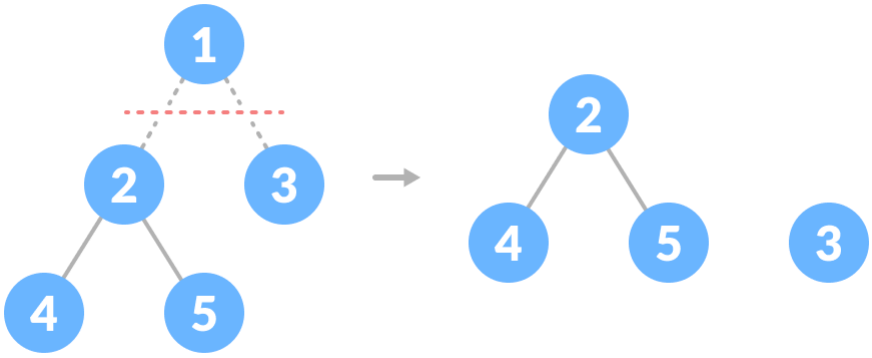
Height and depth of each node in a tree

Degree of a Node

The degree of a node is the total number of branches of that node.

Forest

A collection of disjoint trees is called a forest.



Creating forest from a tree

You can create a forest by cutting the root of a tree.

Types of Trees

1. Binary Tree
2. Binary Search Tree
3. AVL Tree
4. B-Tree

Tree Traversal

In order to perform any operation on a tree, you need to reach to the specific node.

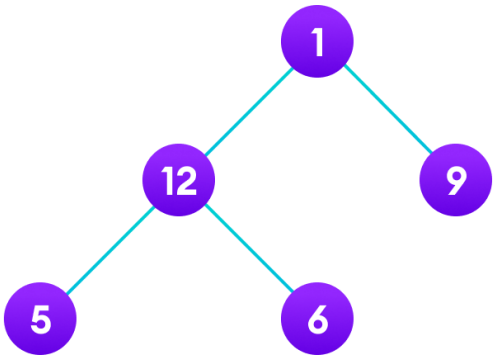
The tree traversal algorithm **helps in visiting a required node in the tree.**

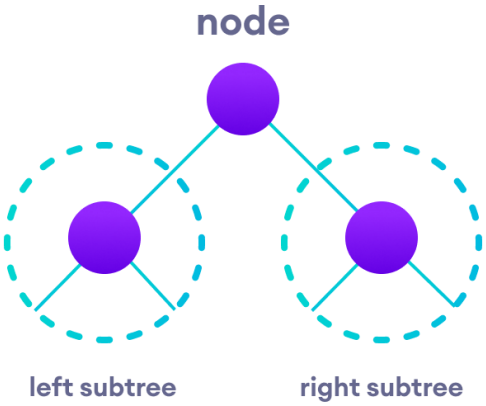
Tree Applications

- **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.**
- **Compilers use a syntax tree to validate the syntax of every program you write.**

Tree Traversal - inorder, preorder and postorder





Inorder traversal

1. **First, visit all the nodes in the left subtree**
2. **Then the root node**
3. **Visit all the nodes in the right subtree**

```
inorder(root->left)
display(root->data)
inorder(root->right)
```

Preorder traversal

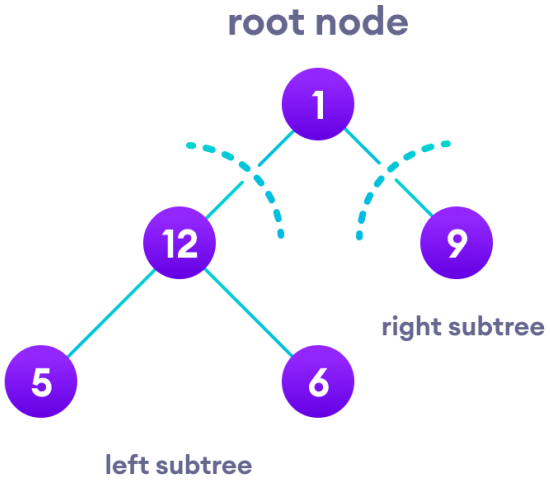
1. **Visit root node**
2. **Visit all the nodes in the left subtree**
3. **Visit all the nodes in the right subtree**

```
display(root->data)
preorder(root->left)
preorder(root->right)
```

Postorder traversal

1. **Visit all the nodes in the left subtree**
2. **Visit all the nodes in the right subtree**
3. **Visit the root node**

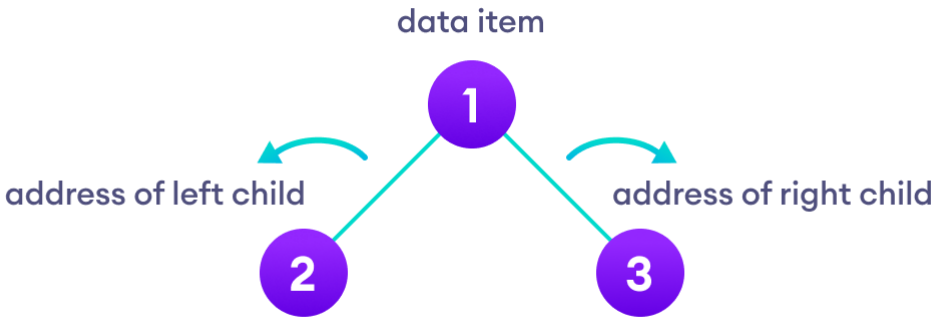
```
postorder(root->left)
postorder(root->right)
display(root->data)
```



Binary Tree

A binary tree is a tree data structure in which each parent node can have at most two children.
Each node of a binary tree consists of three items:

- data item
- address of left child
- address of right child

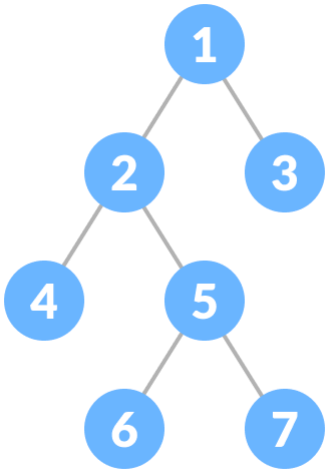


Binary Tree

Types of Binary Tree

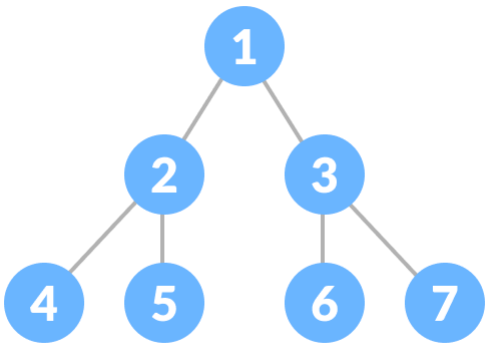
1. Full Binary Tree

A full Binary tree is a special type of binary tree **in which every parent node/internal node has either two or no children.**



2. Perfect Binary Tree

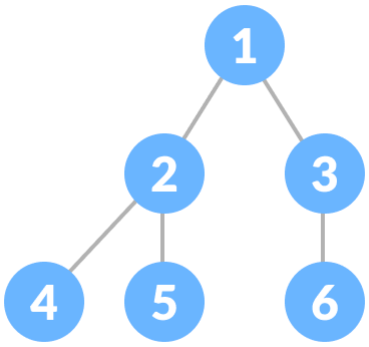
A perfect binary tree is a type of binary tree **in which every internal node has exactly two child** nodes and **all the leaf nodes are at the same level.**



3. Complete Binary Tree

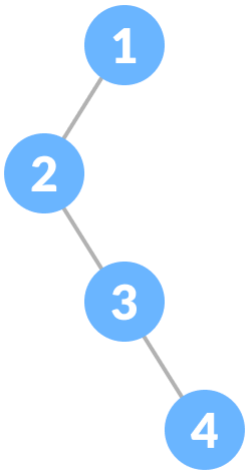
A complete binary tree is just like a full binary tree, but with two major differences

1. **Every level must be completely filled**
2. All the leaf elements must lean towards the left.
3. The last leaf element might not have a right sibling i.e., a complete binary tree doesn't have to be a full binary tree.



4. Degenerate or Pathological Tree

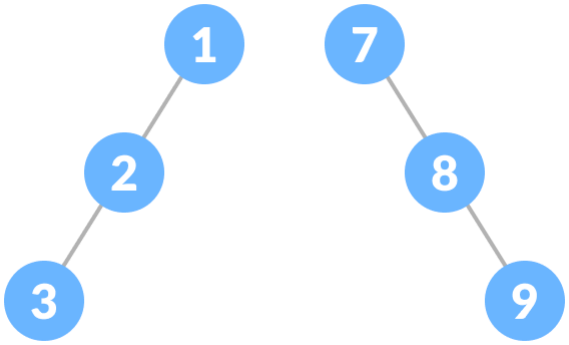
A degenerate or pathological tree is the tree having a single child either left or right.



5. Skewed Binary Tree

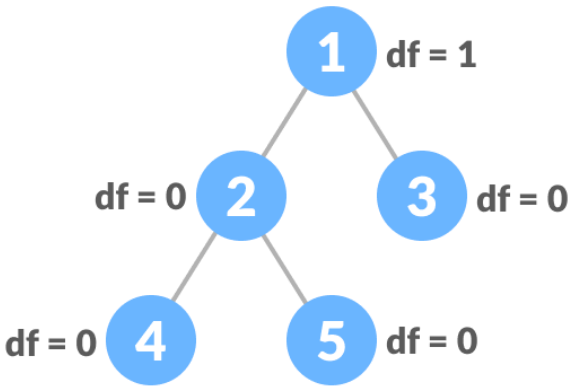
A skewed binary tree is a pathological/degenerate tree in which the tree is either dominated by the left nodes or the right nodes.

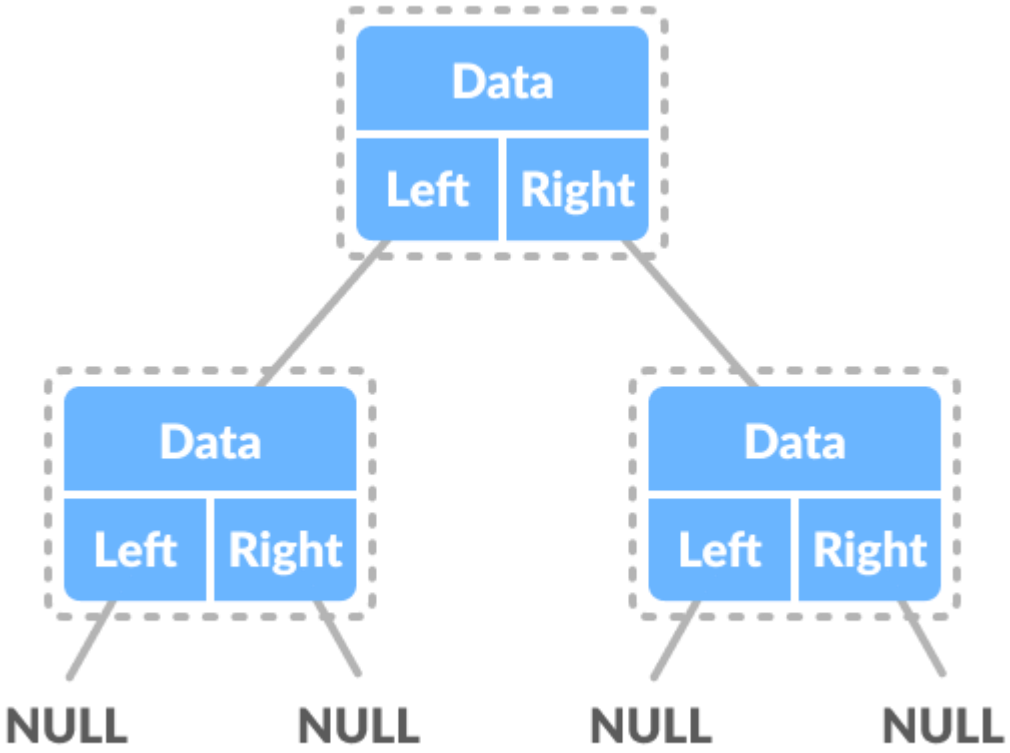
Thus, there are two types of skewed binary tree: **left-skewed binary tree** and **right-skewed binary tree**.



6. Balanced Binary Tree

It is a type of **binary tree in which the difference between the height of the left and the right subtree for each node is either 0 or 1.**

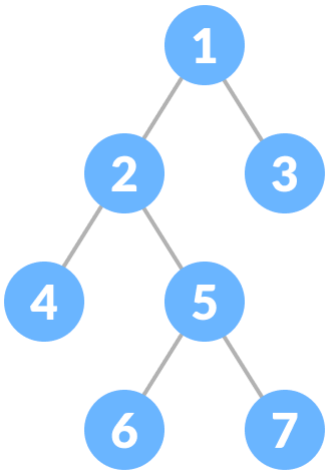




Full Binary Tree

A full Binary tree is a special type of binary tree in which every parent node/internal node has either two or no children.

It is also known as **a proper binary tree**.



Full Binary Tree Theorems

Let, i = the number of internal nodes
n = be the total number of nodes
l = number of leaves
λ = number of levels

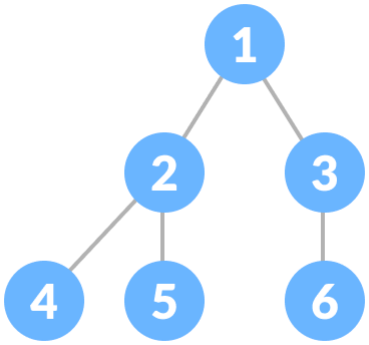
1. **The number of leaves is $i + 1$.**
2. **The total number of nodes is $2i + 1$.**
3. **The number of internal nodes is $(n - 1) / 2$.**
4. **The number of leaves is $(n + 1) / 2$.**
5. **The total number of nodes is $2l - 1$.**
6. **The number of internal nodes is $l - 1$.**
7. The number of leaves is at most $2^{\lambda - 1}$.

Complete Binary Tree

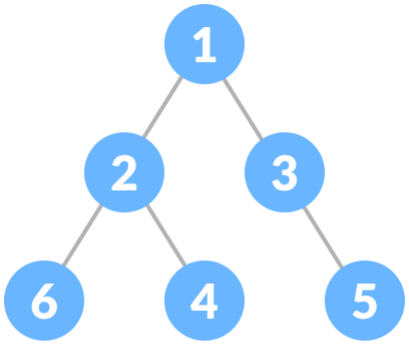
A complete binary tree is a binary tree in which all the levels are completely filled except possibly the lowest one, **which is filled from the left.**

A complete binary tree is just like a full binary tree, but with two major differences

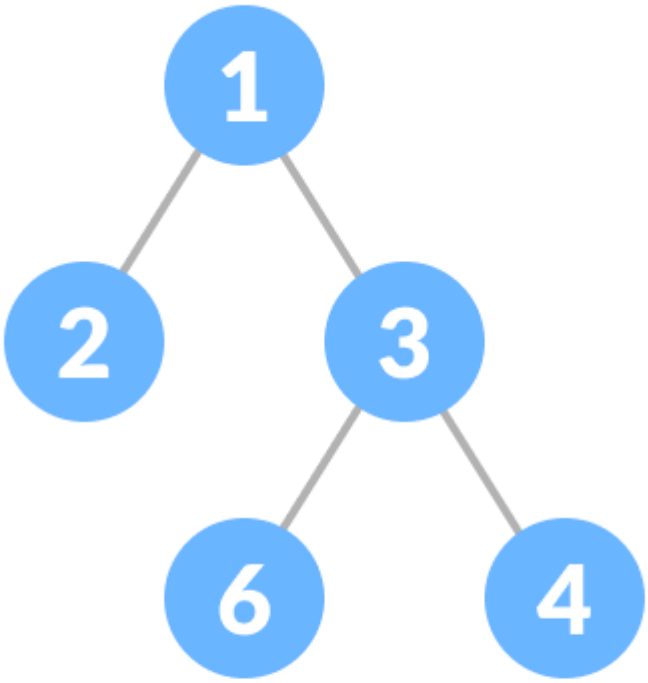
1. All the leaf elements must lean towards the left.
2. The last leaf element might not have a right sibling i.e. a complete binary tree doesn't have to be a full binary tree.



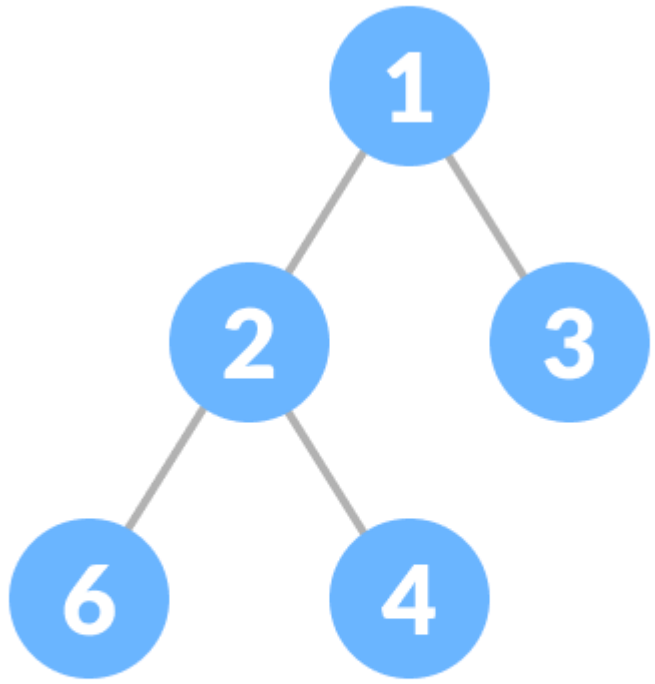
Full Binary Tree vs Complete Binary Tree



✗ Full Binary Tree
✗ Complete Binary Tree



✓ Full Binary Tree
✗ Complete Binary Tree



✔ Full Binary Tree

✔ Complete Binary Tree

How a Complete Binary Tree is Created?

1. Select the **first element of the list to be the root node.** (no. of elements on level-I: 1)

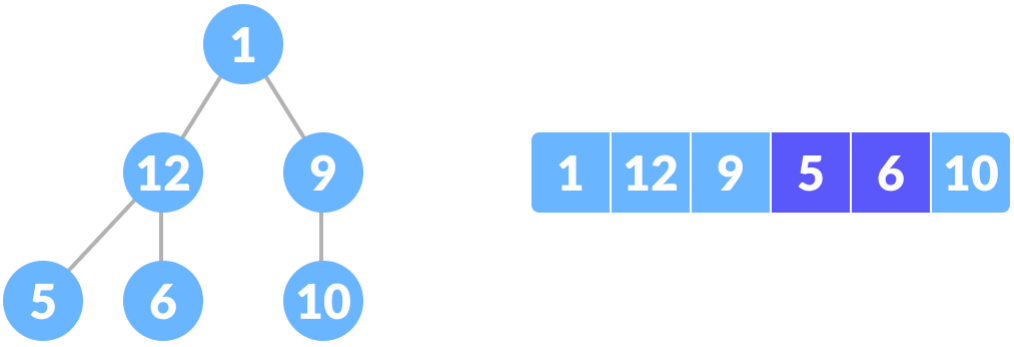
Select the first element as root

2. **Put the second element as a left child of the root node and the third element as the right child.** (no. of elements on level-II: 2)

12 as a left child and 9 as a right child

3. Put the **next two elements as children of the left node of the second level.** Again, **put the next two elements as children of the right node of the second level** (no. of elements on level-III: 4) elements).

4. Keep repeating until you reach the last element.



5 as a left child and 6 as a right child

Relationship between array indexes and tree element

A complete binary tree has an interesting property that we can use to find the children and parents of any node.

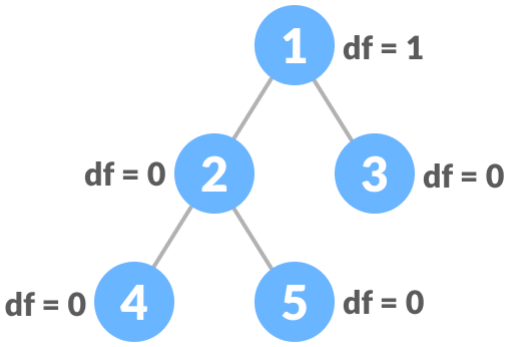
If the **index of any element in the array is i, the element in the index 2i+1 will become the left child and element in 2i+2 index will become the right child.** Also, **the parent of any element at index i is given by the lower bound of (i-1)/2.**

Balanced Binary Tree

A balanced binary tree, also referred to as a height-balanced binary tree, is defined as a binary tree in **which the height of the left and right subtree of any node differs by not more than 1.**

Following are the conditions for a height-balanced binary tree:

1. **difference between the left and the right subtree for any node is not more than one**
2. the left subtree is balanced
3. the right subtree is balanced



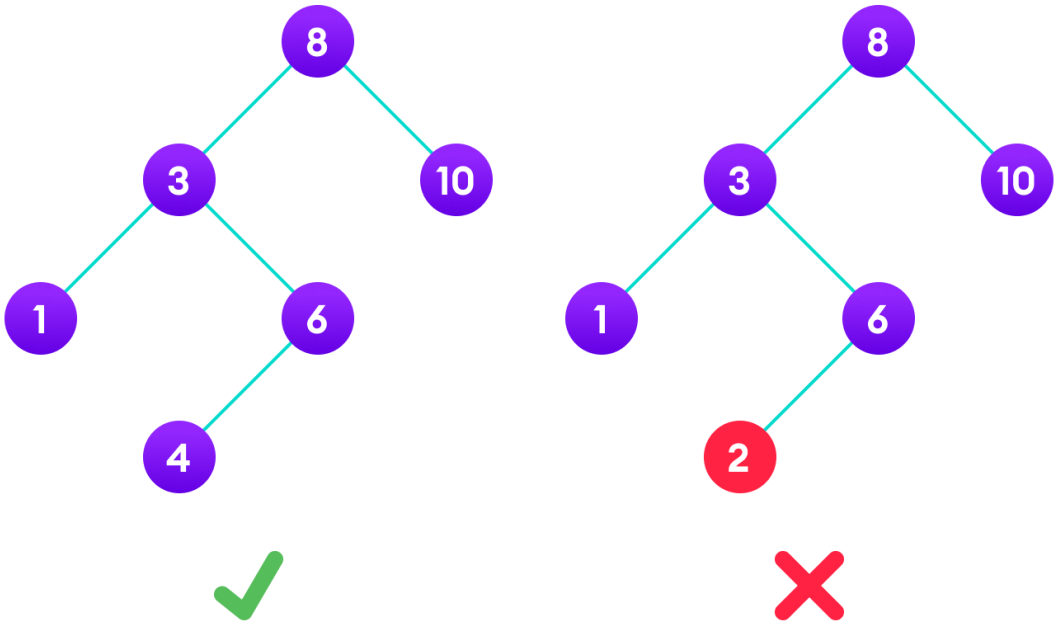
Binary Search Tree (BST)

Binary search tree is a data structure that quickly allows us to maintain a sorted list of numbers.

- It is called a binary tree because **each tree node has a maximum of two children.**
- It is called a search tree because **it can be used to search for the presence of a number in O(log(n)) time.**

The properties that separate a binary search tree from a regular binary tree is

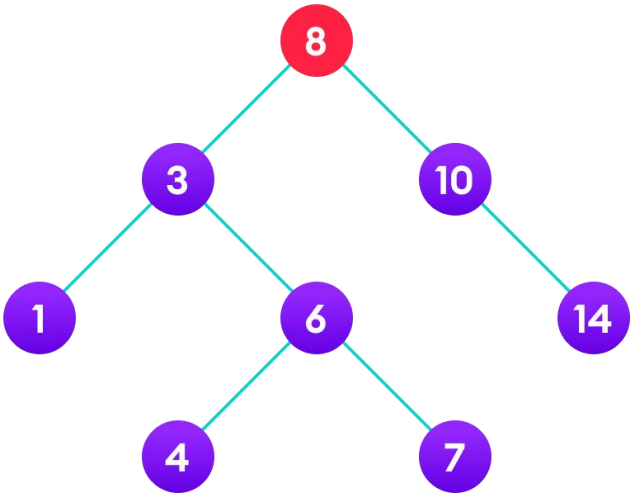
1. All nodes of left subtree are less than the root node
2. All nodes of right subtree are more than the root node
3. Both subtrees of each node are also BSTs i.e., they have the above two properties

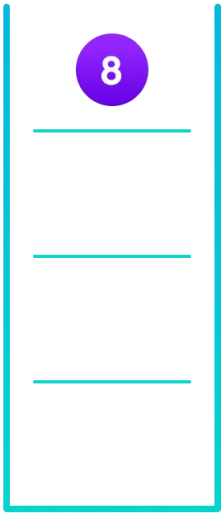


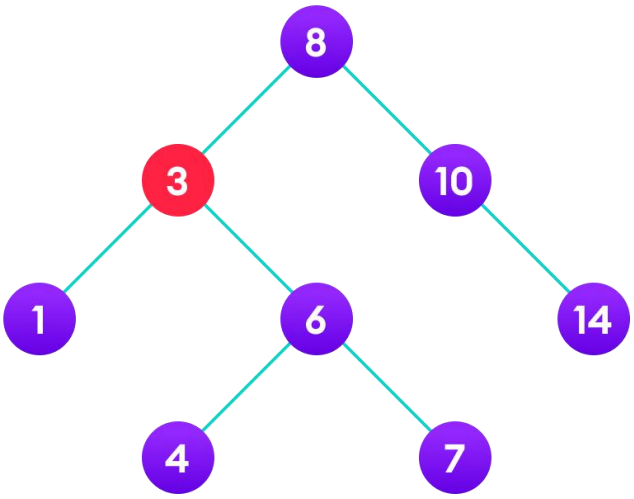
Search Operation

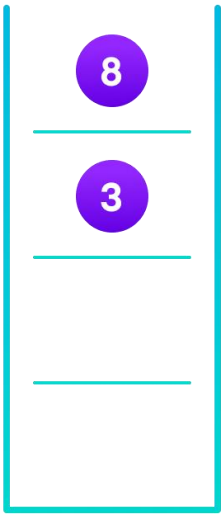
Algorithm:

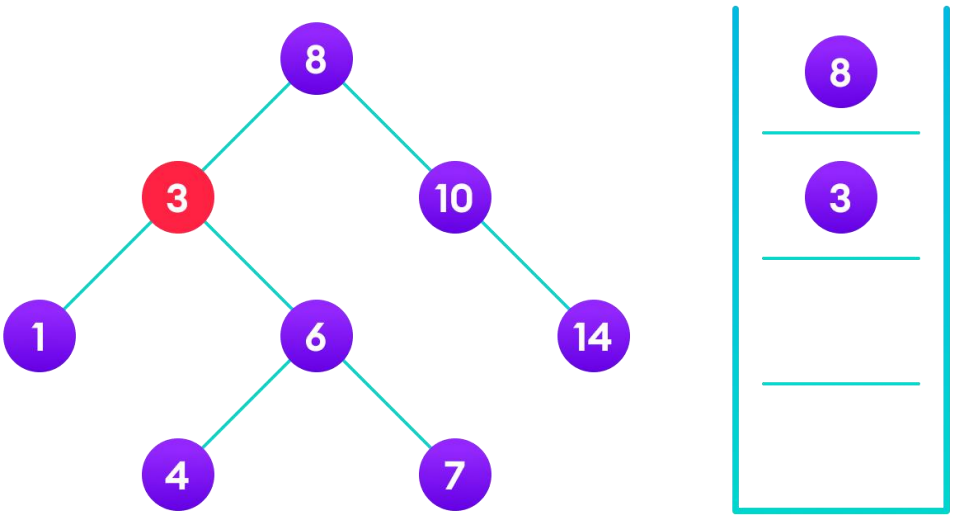
```
If root == NULL
   return NULL;
If number == root->data
   return root->data;
If number < root->data
   return search(root->left)
If number > root->data
   return search(root->right)
```

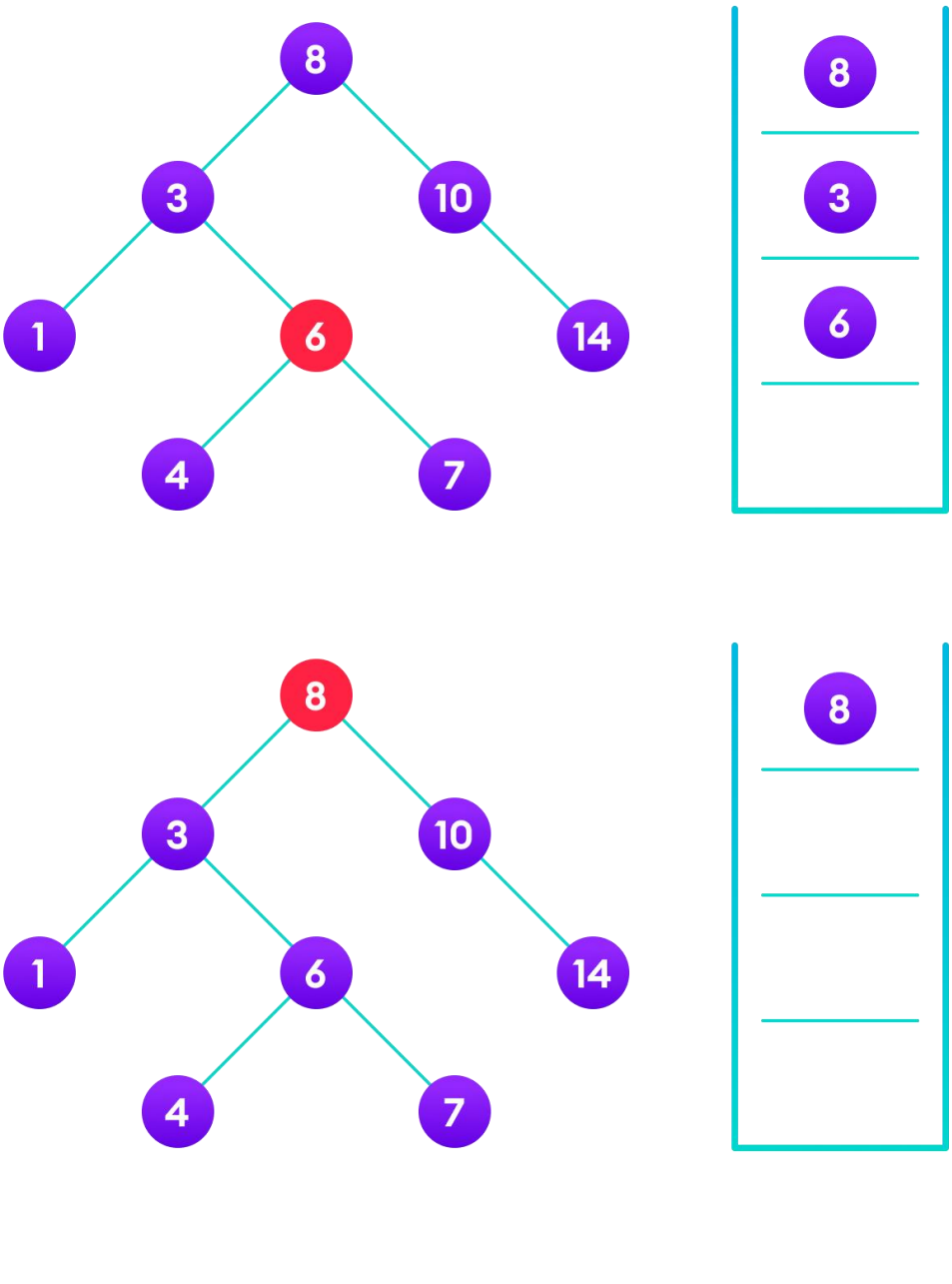

8
3
10
1
6
14
4
7


8

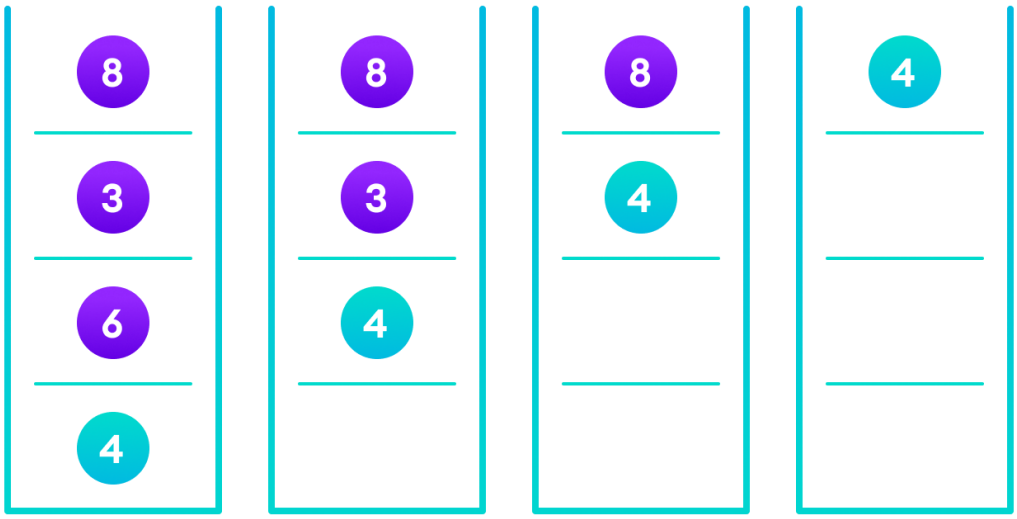

8
3
10
1
6
14
4
7


8
3


8
3
10
1
6
14
4
7
8
3


8
3
10
1
6
14
4
7
8
3
6

8
3
10
1
6
14
4
7
8



Insert Operation

Algorithm:

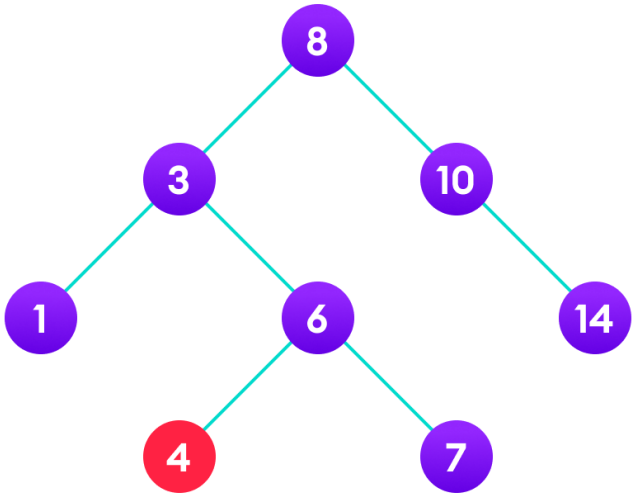
```
If node == NULL
    return createNode(data)
if (data < node->data)
    node->left  = insert(node->left, data);
else if (data > node->data)
    node->right = insert(node->right, data);
return node;
```

Deletion Operation

There are three cases for deleting a node from a binary search tree.

Case I

In the first case, the node to be deleted is the leaf node. In such a case, simply delete the node from the tree.



Case II

In the second case, the node to be deleted **lies has a single child node.** In such a case follow the steps below:

1. **Replace that node with its child node.**
2. **Remove the child node from its original position.**

Case III

In the third case, the node to be deleted has two children. In such a case follow the steps below:

1. **Get the inorder successor of that node.**
2. **Replace the node with the inorder successor.**
3. **Remove the inorder successor from its original position.**

AVL Tree

AVL tree is a self-balancing binary search tree in which each node maintains extra information called a balance factor whose value is either -1, 0 or +1.

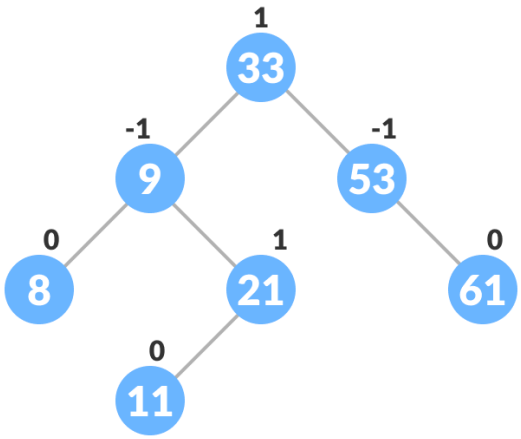
Balance Factor

Balance factor of a node in an AVL tree is the **difference between the height of the left subtree and that of the right subtree of that node.**

Balance Factor = (Height of Left Subtree - Height of Right Subtree) or (Height of Right Subtree - Height of Left Subtree)

The self-balancing property of an avl tree is maintained by the balance factor. The value of balance factor should always be -1, 0 or +1.

An example of a balanced avl tree is:



Avl tree

Operations on an AVL tree

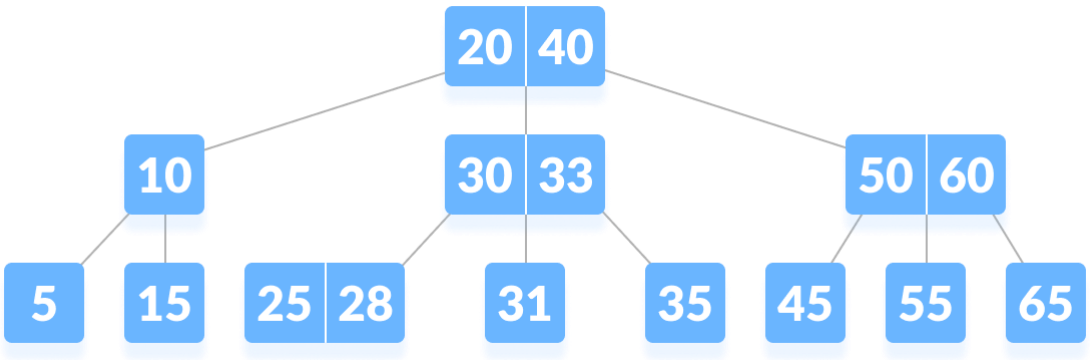
Various operations that can be performed on an AVL tree are:

Rotating the subtrees in an AVL Tree

In rotation operation, **the positions of the nodes of a subtree are interchanged.**

B-tree

B-tree **is a special type of self-balancing search tree in which each node can contain more than one key and can have more than two children.** It is a generalized form of the binary search tree.



B-tree