

Trees

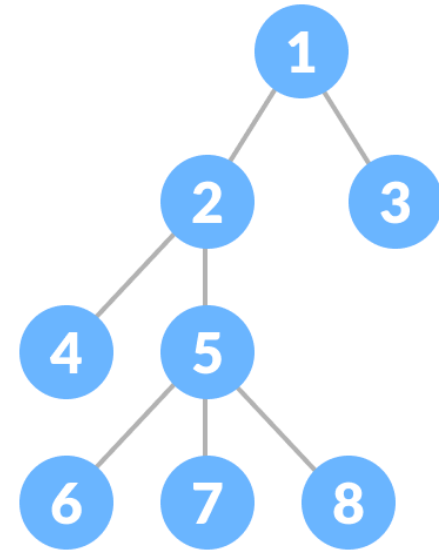
2022/2023

Tree Data Structure

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

Why Tree Data Structure?

- Other data structures such as arrays, linked list, stack, and queue are linear data structures that store data sequentially. In order to perform any operation in a linear data structure, the time complexity increases with the increase in the data size. But, it is not acceptable in today's computational world.

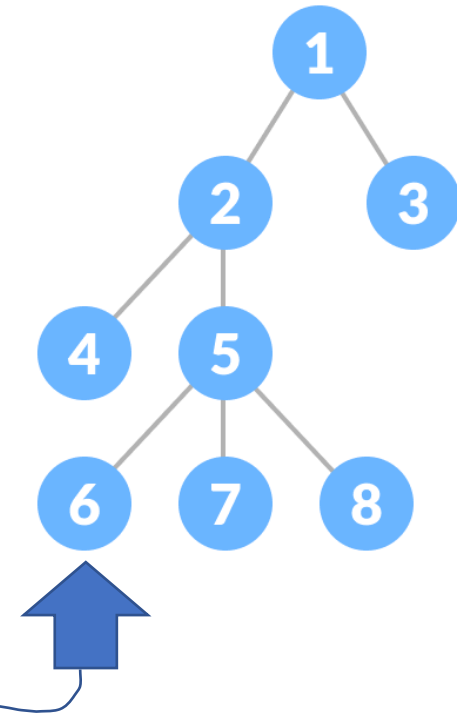


Tree Data Structure

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.



Tree Data Structure

Tree Terminologies

Edge

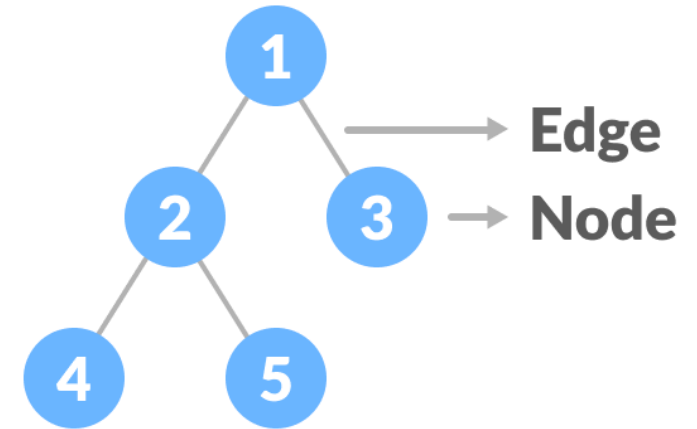
- It is the link between any two nodes.


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 (ie. the longest path from the node to a leaf node).



 Exo 02 in series 04

Tree Data Structure

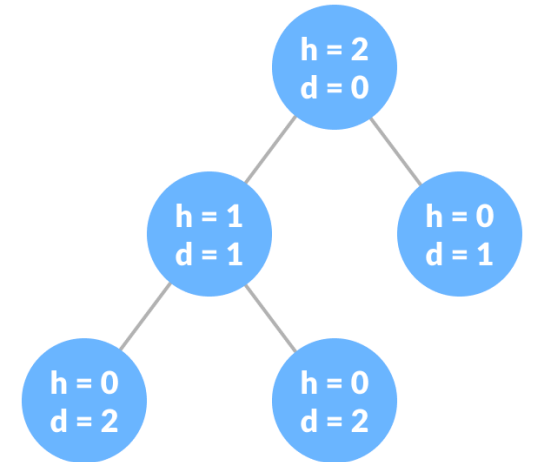
Tree Terminologies

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.

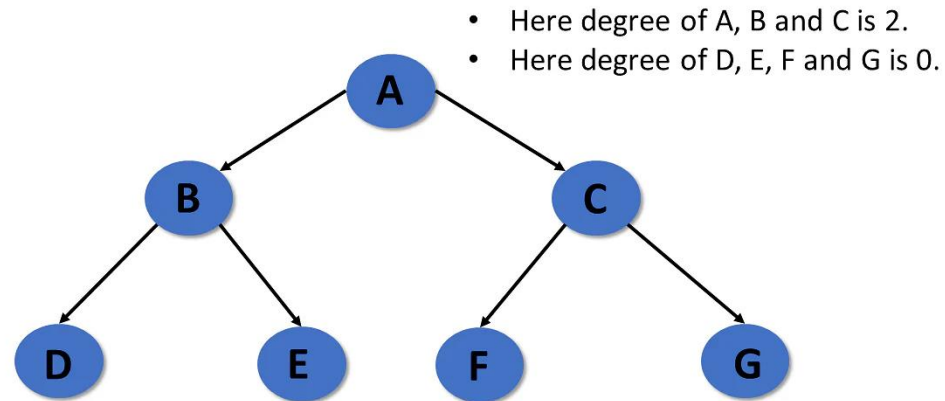


Tree Data Structure

Tree Terminologies

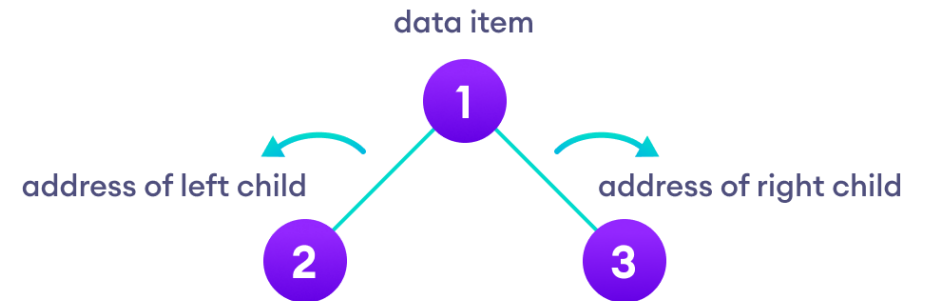
Degree of a Node

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



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

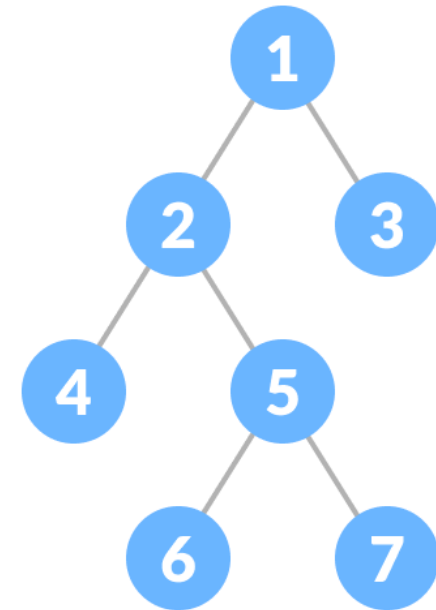


Binary Tree

Types of Binary Tree

I. 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.



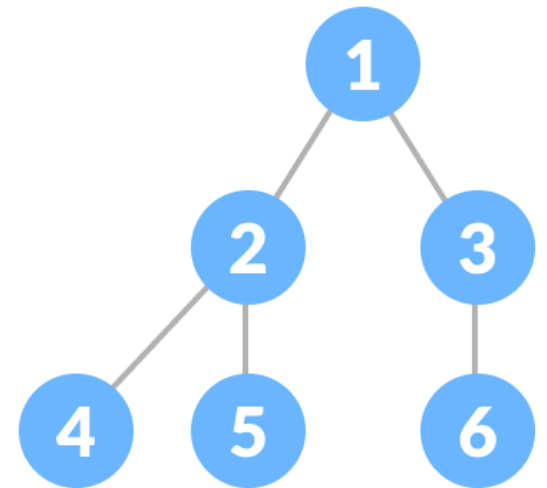
Binary Tree

Types of Binary Tree

2. Complete Binary Tree

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

- Every level must be completely filled
- All the leaf elements must lean towards the left.
- 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.



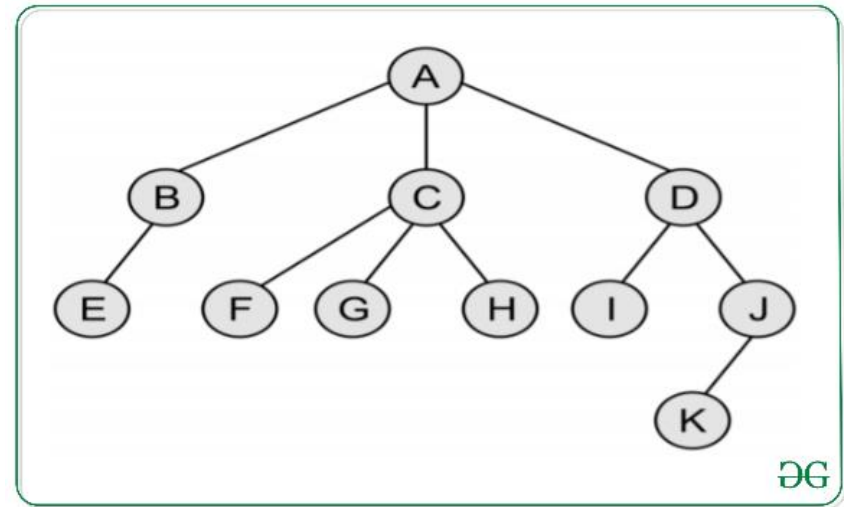
Convert a Generic Tree(N-array Tree) to Binary Tree

- The root of the Binary Tree is the Root of the Generic Tree.
- The left child of a node in the Generic Tree is the Left child of that node in the Binary Tree.
- The right sibling of any node in the Generic Tree is the Right child of that node in the Binary Tree.

Convert a Generic Tree(N-array Tree) to Binary Tree

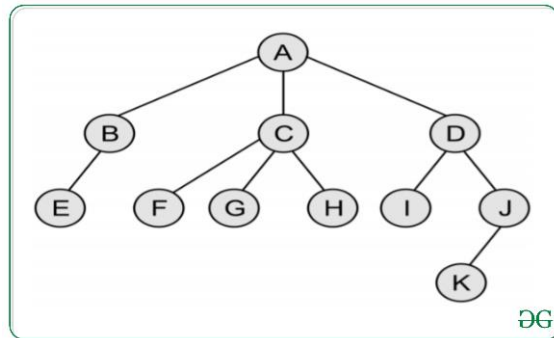
Example

- Convert the following Generic Tree to Binary Tree:



Convert a Generic Tree(N-array Tree) to Binary Tree

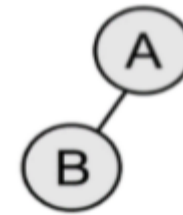
- I. As per the rules mentioned above, the root node of general tree A is the root node of the binary tree.



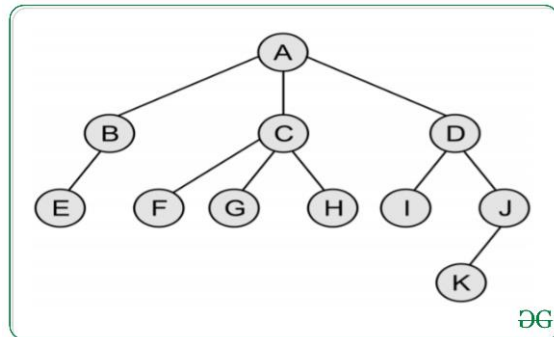
Step 1

Convert a Generic Tree(N-array Tree) to Binary Tree

2. Now the leftmost child node of the root node in the general tree is B and it is the leftmost child node of the binary tree.

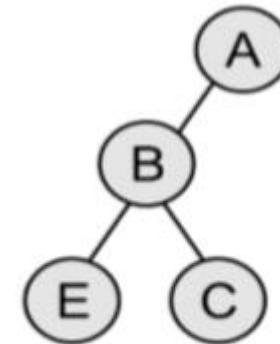
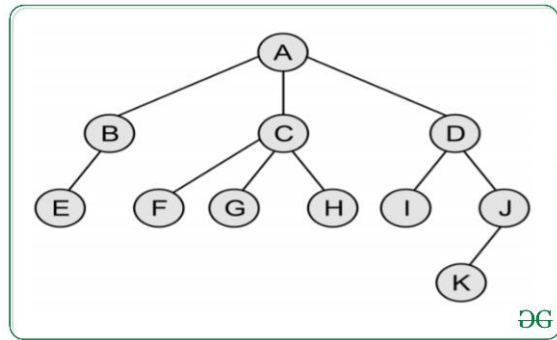


Step 2



Convert a Generic Tree(N-array Tree) to Binary Tree

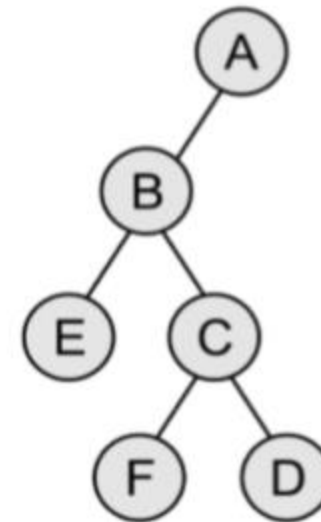
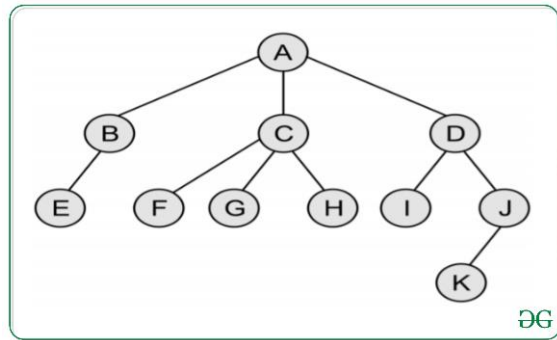
3. Now as B has E as its leftmost child node, so it is its leftmost child node in the binary tree whereas it has C as its rightmost sibling node so it is its right child node in the binary tree.



Step 3

Convert a Generic Tree(N-array Tree) to Binary Tree

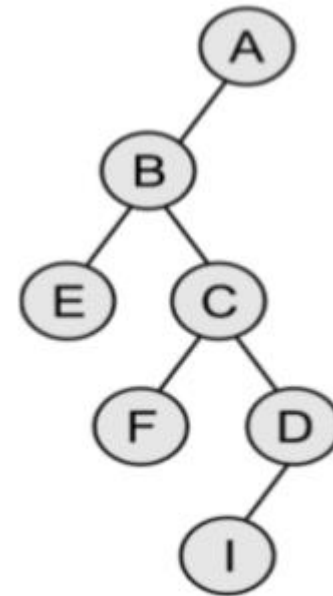
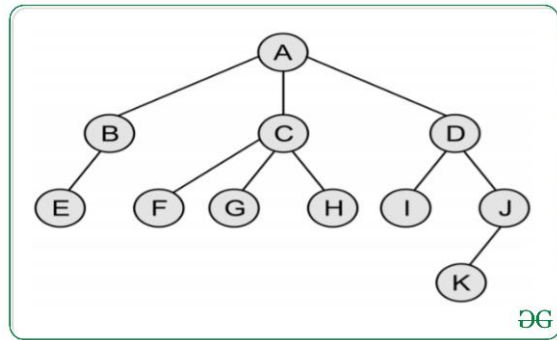
4. Now C has F as its leftmost child node and D as its rightmost sibling node, so they are its left and right child node in the binary tree respectively.



Step 4

Convert a Generic Tree(N-array Tree) to Binary Tree

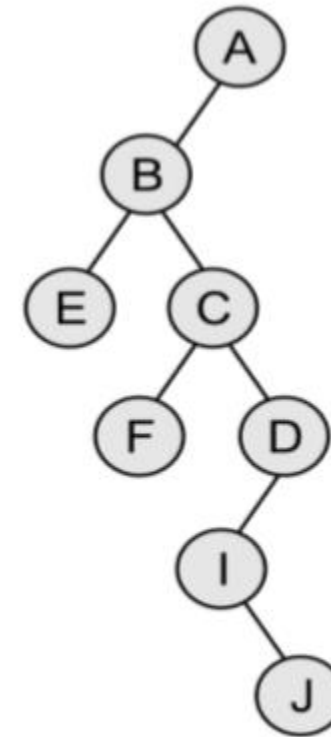
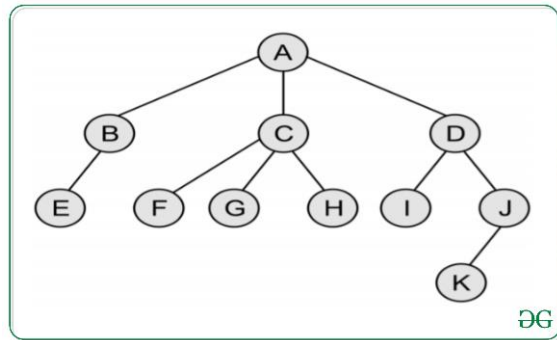
- Now D has I as its leftmost child node which is its left child node in the binary tree but doesn't have any rightmost sibling node, so doesn't have any right child in the binary tree.



Step 5

Convert a Generic Tree(N-array Tree) to Binary Tree

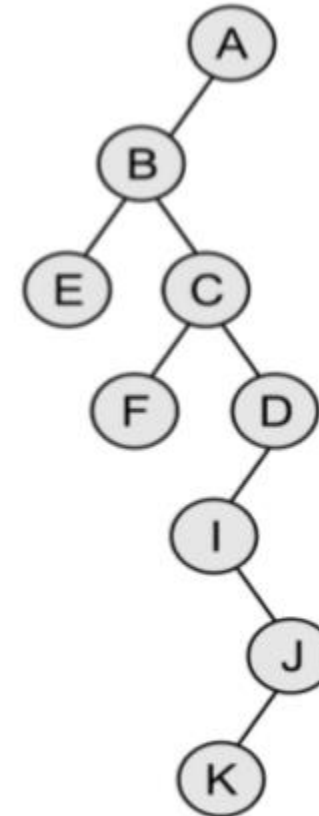
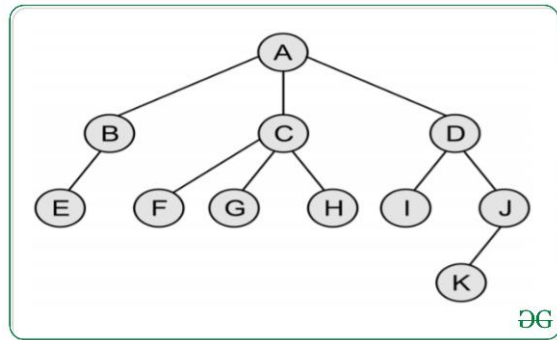
- Now for I, J is its rightmost sibling node and so it is its right child node in the binary tree.



Step 6

Convert a Generic Tree(N-array Tree) to Binary Tree

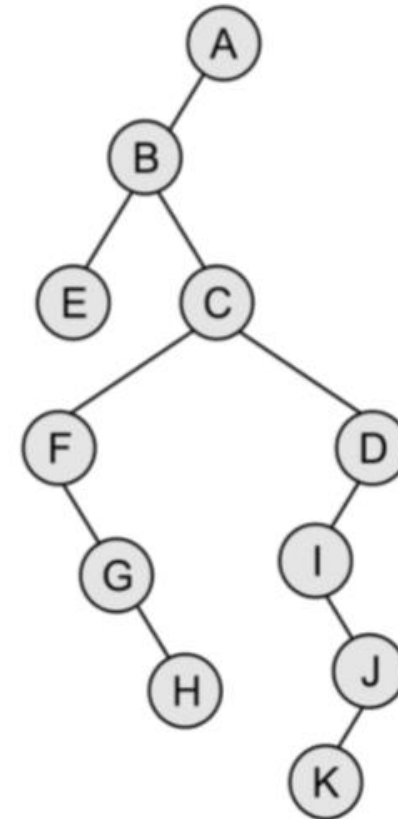
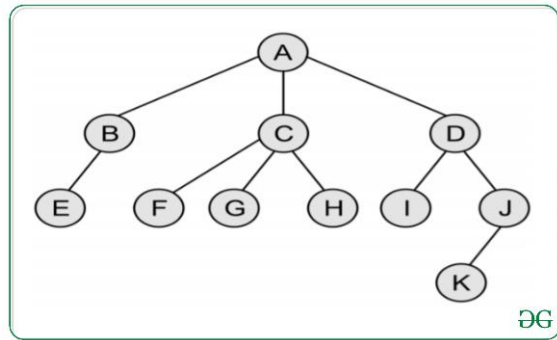
7. Similarly, for J, K is its leftmost child node and thus it is its left child node in the binary tree.



Step 7

Convert a Generic Tree(N-array Tree) to Binary Tree

- Now for C, F is its leftmost child node, which has G as its rightmost sibling node, which has H as its just right sibling node and thus they form their left, right, and right child node respectively.

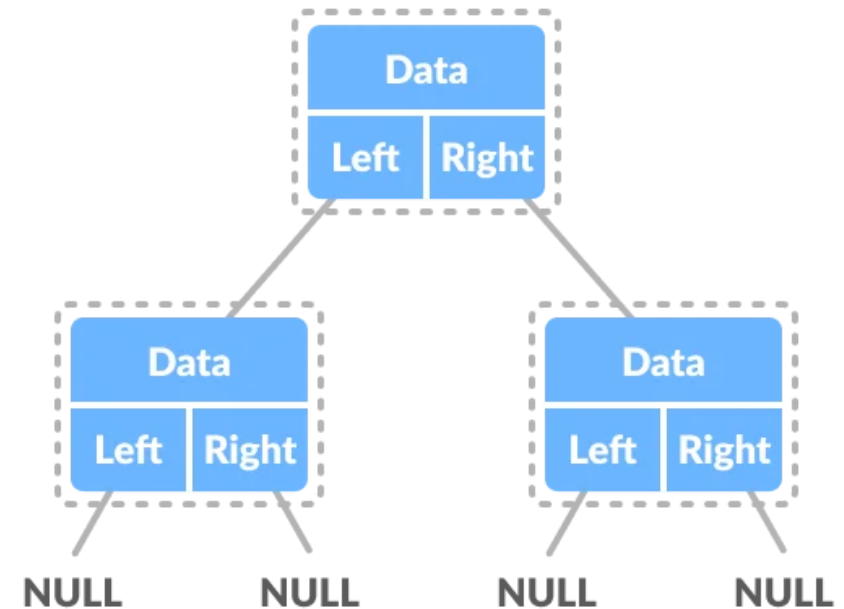


Step 8

Binary Tree Representation

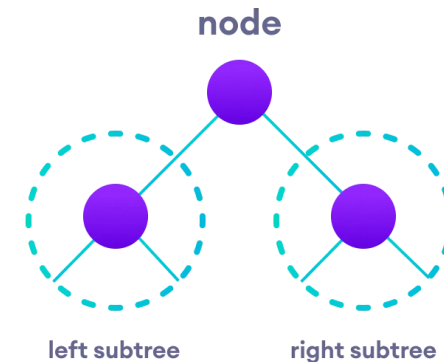
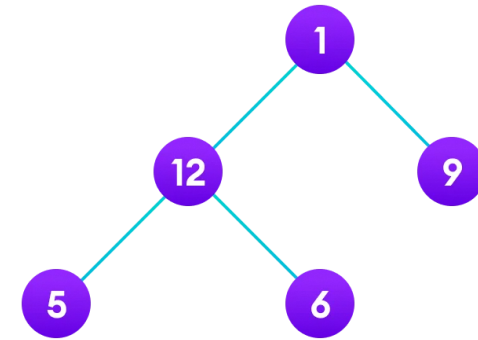
- A node of a binary tree is represented by a structure containing a data part and two pointers to other structures of the same type.

```
struct node
{
    int data;
    struct node *left;
    struct node *right;
};
```



Tree Traversal - inorder, preorder and postorder

- Linear data structures like arrays, stacks, queues, and linked list have only one way to read the data. But a hierarchical data structure like a tree can be traversed in different ways.
- According to this structure, every tree is a combination of
 - A node carrying data
 - Two subtrees



Inorder traversal

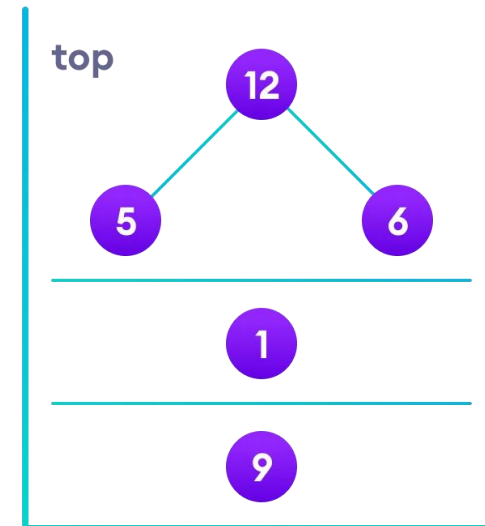
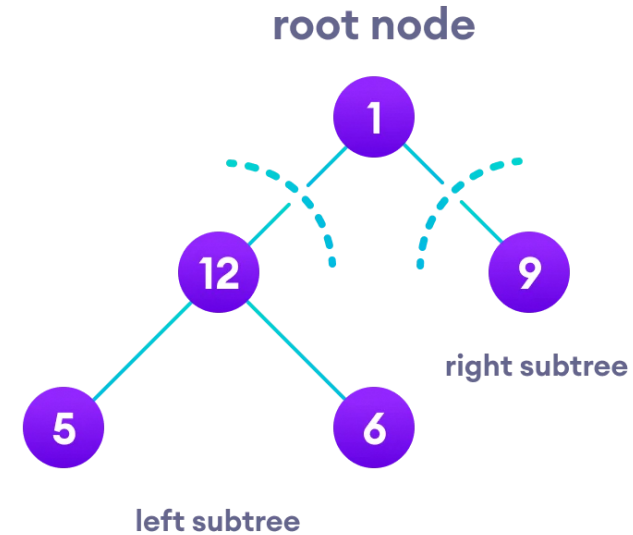
1. First, visit all the nodes in the left subtree
2. Then the root node
3. Visit all the nodes in the right subtree

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

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

- We traverse the left subtree first. We also need to remember to visit the root node and the right subtree when this tree is done.
- Let's put all this in a stack so that we remember.



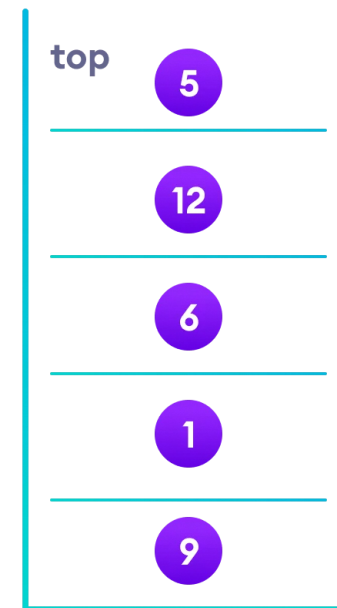
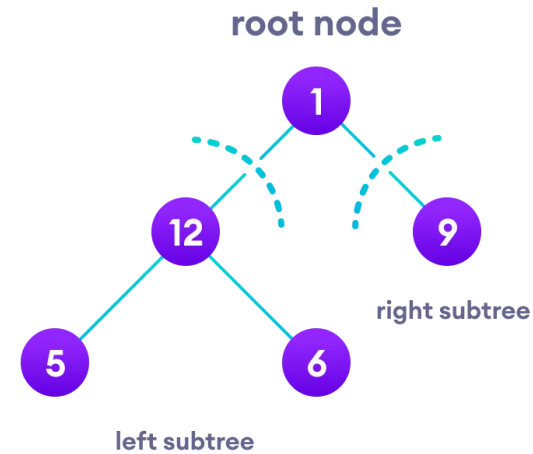
Inorder traversal

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

- Now we traverse to the subtree pointed on the TOP of the stack.
- Again, we follow the same rule of inorder

Left subtree -> root -> right subtree

- We don't have to create the stack ourselves because recursion maintains the correct order for us.



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 Representations

A binary tree data structure is represented using two methods. Those methods are as follows...

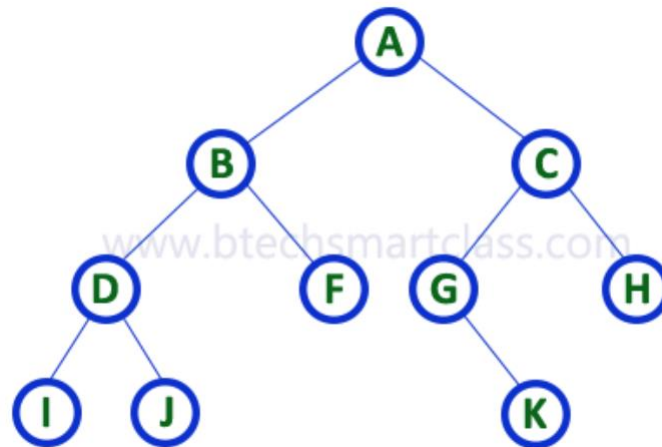
1. Array Representation

2. Linked List Representation

I. Array Representation of Binary Tree

In array representation of a binary tree, we use one-dimensional array (I-D Array) to represent a binary tree.

Consider the above example of a binary tree and it is represented as follows...

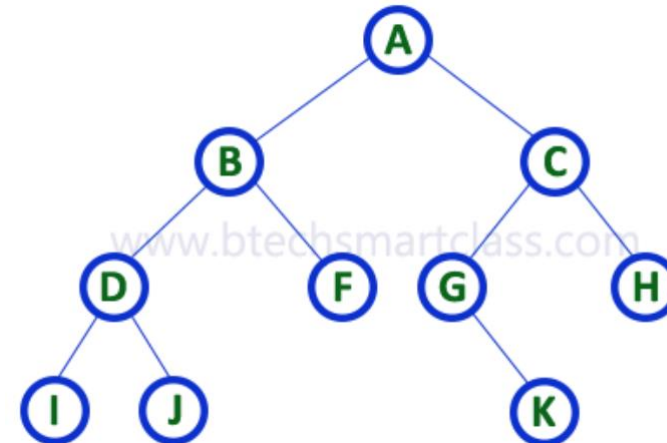
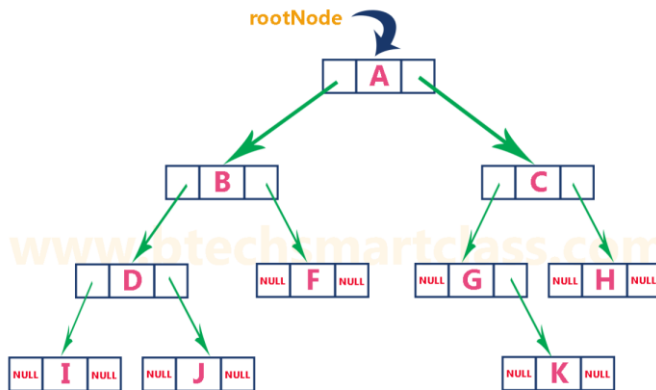


2. Linked List Representation of Binary Tree

- We use a double linked list to represent a binary tree. In a double linked list, every node consists of three fields. First field for storing left child address, second for storing actual data and third for storing right child address.

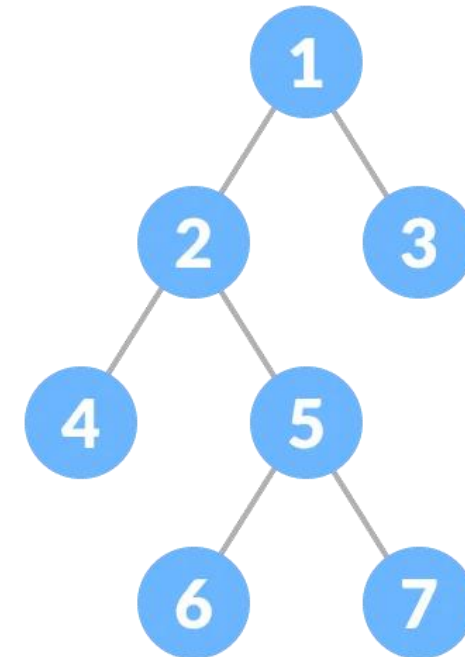


- The precedent example of the binary tree represented using Linked list representation is shown as follows...



Complete Binary Tree

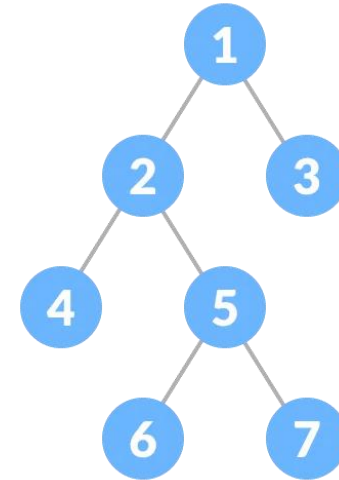
- A complete binary tree is a binary tree in which all the levels are completely filled except possibly the lowest one, which is filled from the left.
- A complete binary tree is just like a **full binary tree**, but with two major differences
- A **full Binary tree** is a special type of binary tree in which every parent node/internal node has either two or no children.
- It is also known as a **proper binary tree**.



Complete Binary Tree

Full Binary Tree Theorems

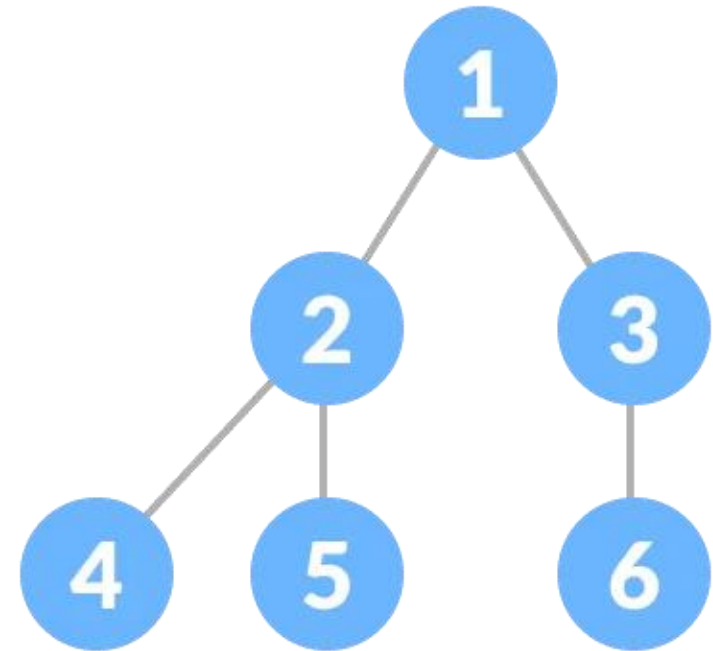
1. The number of leaves is $i + 1$.
2. The total number of nodes is $2i + 1$.
3. The number of internal nodes is $(n - 1) / 2$.
4. The number of leaves is $(n + 1) / 2$.
5. The total number of nodes is $2l - 1$.
6. The number of internal nodes is $l - 1$.
7. The number of leaves is at most $2^{\lambda}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



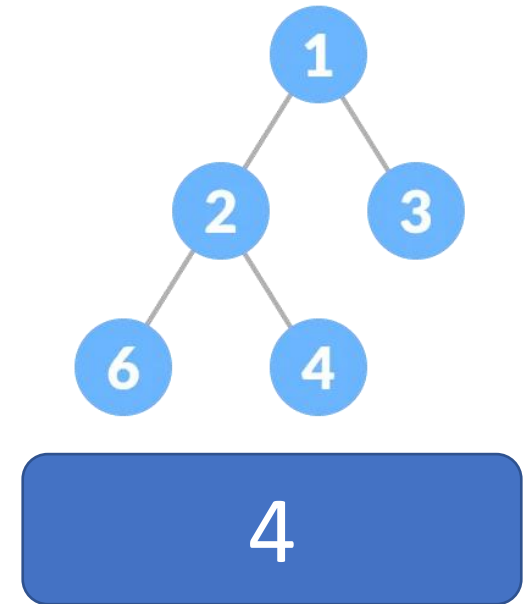
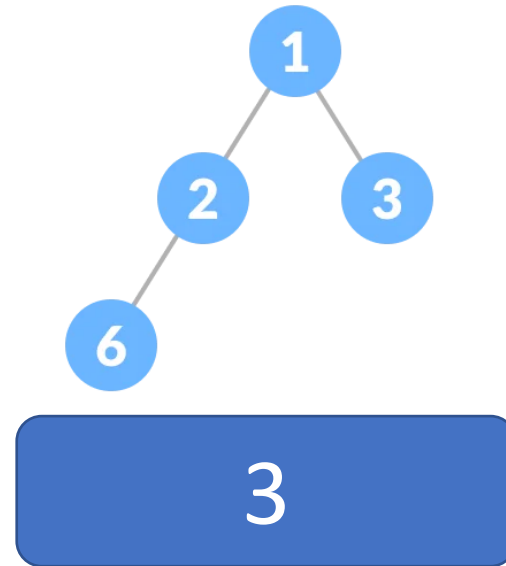
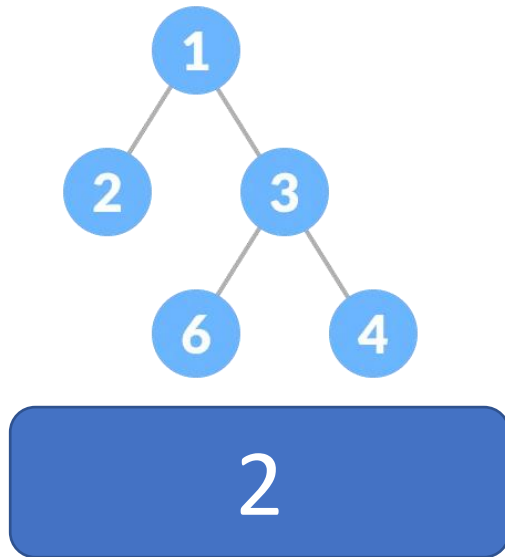
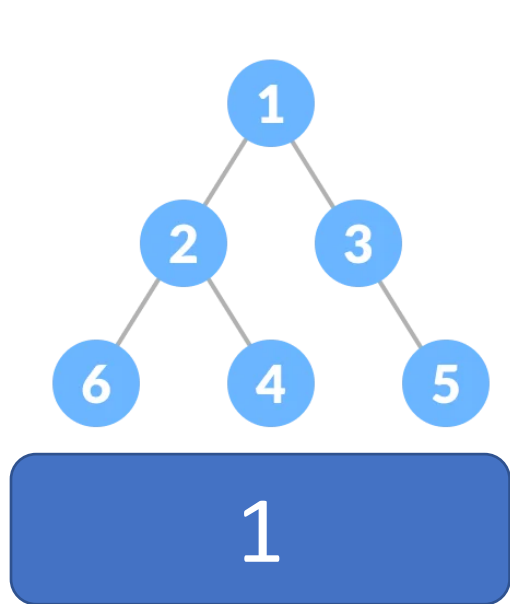
Let, i = the number of internal nodes
 n = be the total number of nodes
 l = number of leaves
 λ = number of levels

Complete Binary Tree

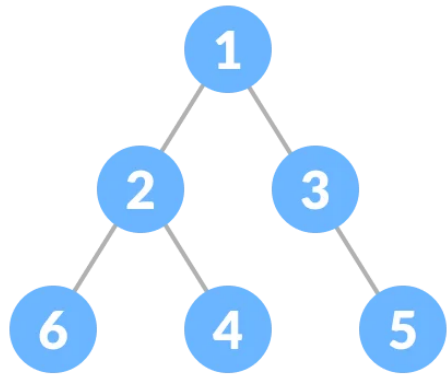
- A complete binary tree is just like a **full binary tree**, but with two major differences
 1. All the leaf elements must lean towards the left.
 2. The last leaf element might not have a right sibling i.e. a complete binary tree doesn't have to be a full binary tree.



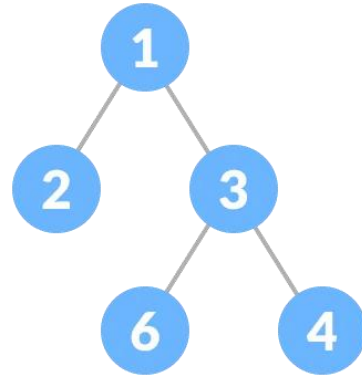
Complete Binary Tree



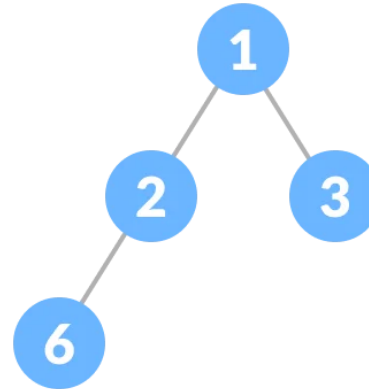
Complete Binary Tree



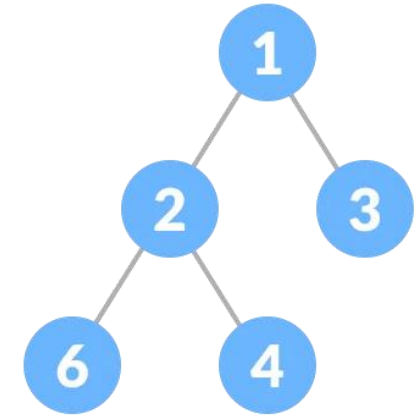
- ✗ Full Binary Tree
- ✗ Complete Binary Tree



2

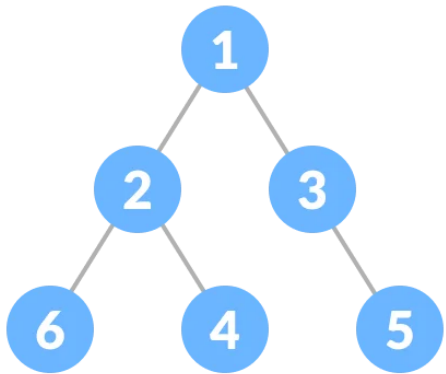


3

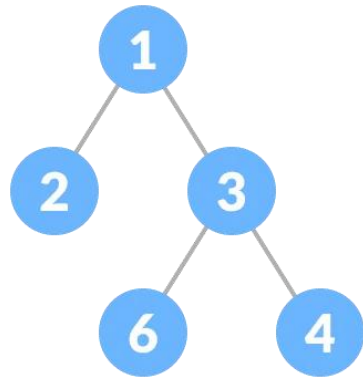


4

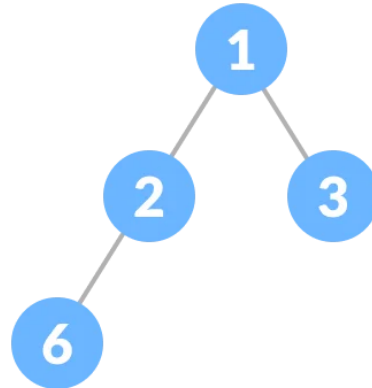
Complete Binary Tree



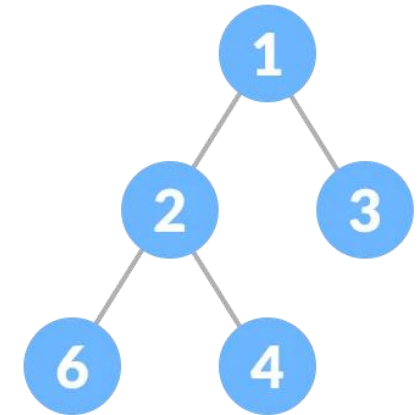
✗ Full Binary Tree
✗ Complete Binary Tree



✓ Full Binary Tree
✗ Complete Binary Tree

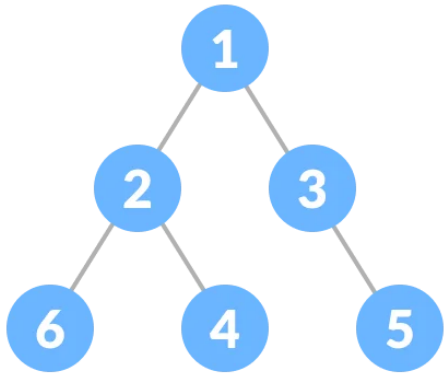


3

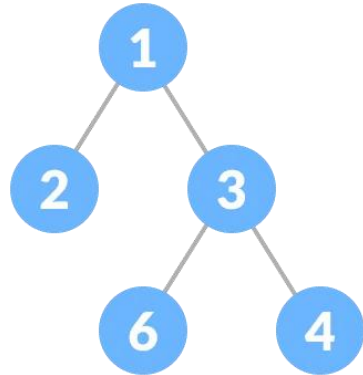


4

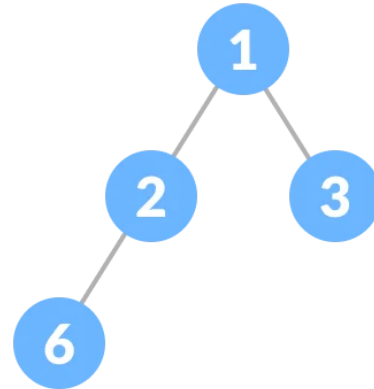
Complete Binary Tree



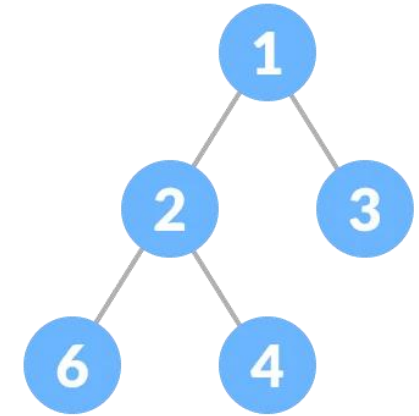
✗ Full Binary Tree
✗ Complete Binary Tree



✓ Full Binary Tree
✗ Complete Binary Tree

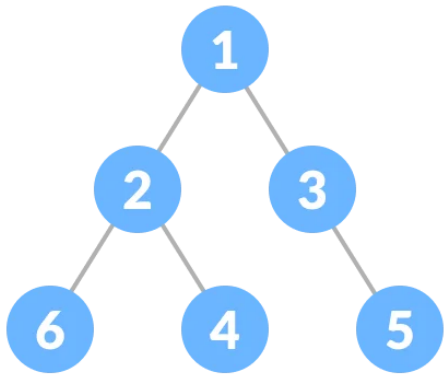


✗ Full Binary Tree
✓ Complete Binary Tree

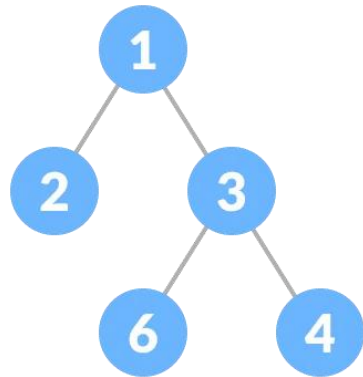


4

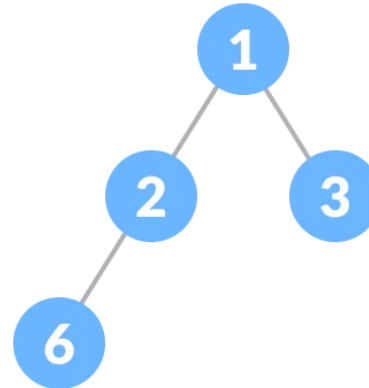
Complete Binary Tree



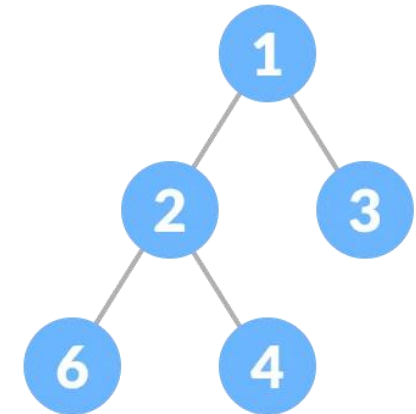
✗ Full Binary Tree
✗ Complete Binary Tree



✓ Full Binary Tree
✗ Complete Binary Tree



✗ Full Binary Tree
✓ Complete Binary Tree

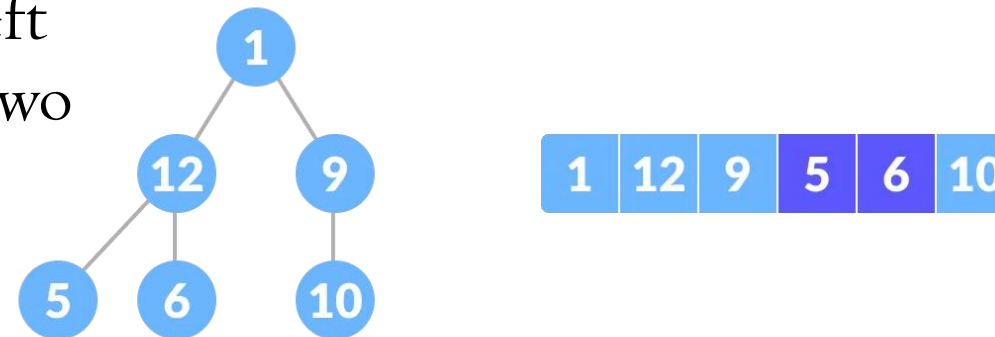


✓ Full Binary Tree
✓ Complete Binary Tree

Complete Binary Tree

How a Complete Binary Tree is Created?

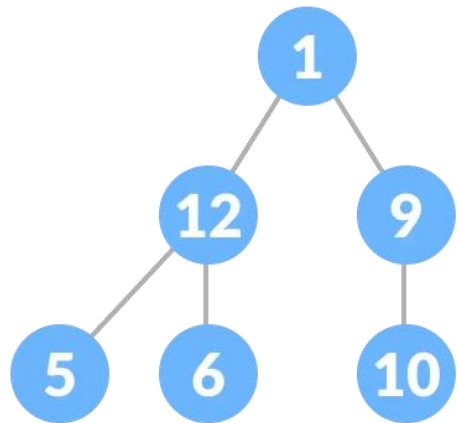
1. Select the first element of the list to be the root node. (no. of elements on level-I: 1)
2. Put the second element as a left child of the root node and the third element as the right child. (no. of elements on level-II: 2)
3. Put the next two elements as children of the left node of the second level. Again, put the next two elements as children of the right node of the second level (no. of elements on level-III: 4 elements).



Complete Binary Tree

How a Complete Binary Tree is Created?

- I. Keep repeating until you reach the last element.

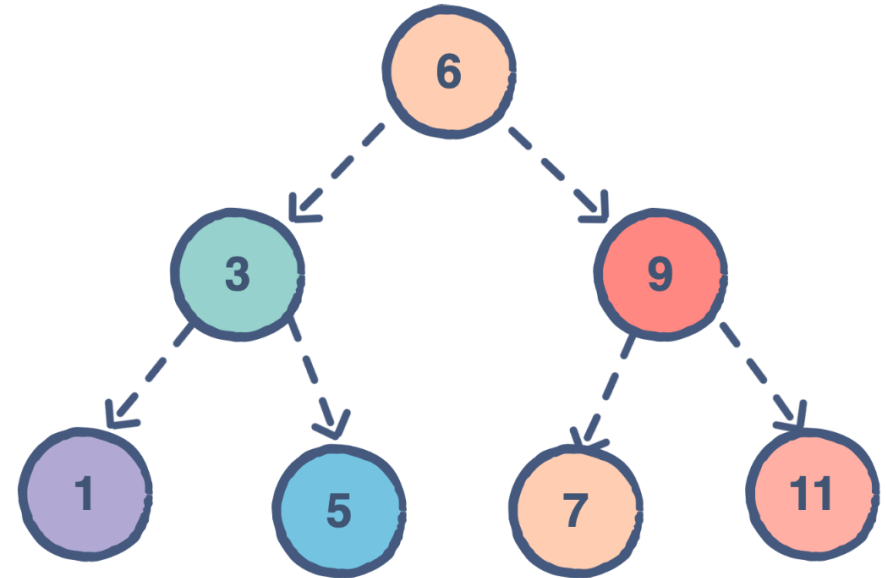


Binary Search Tree(BST)

How a Complete Binary Tree is Created?

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.

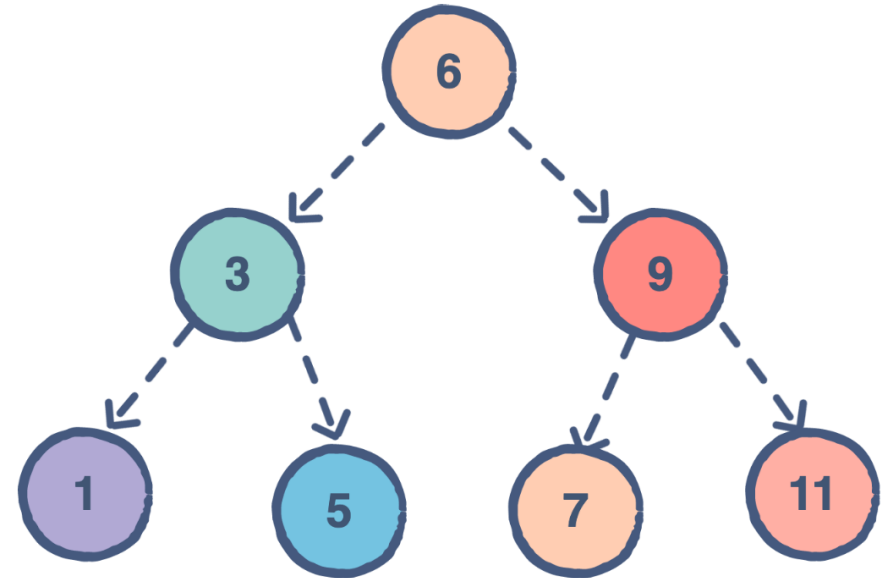


An example of a binary search tree

Binary Search Tree(BST)

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

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

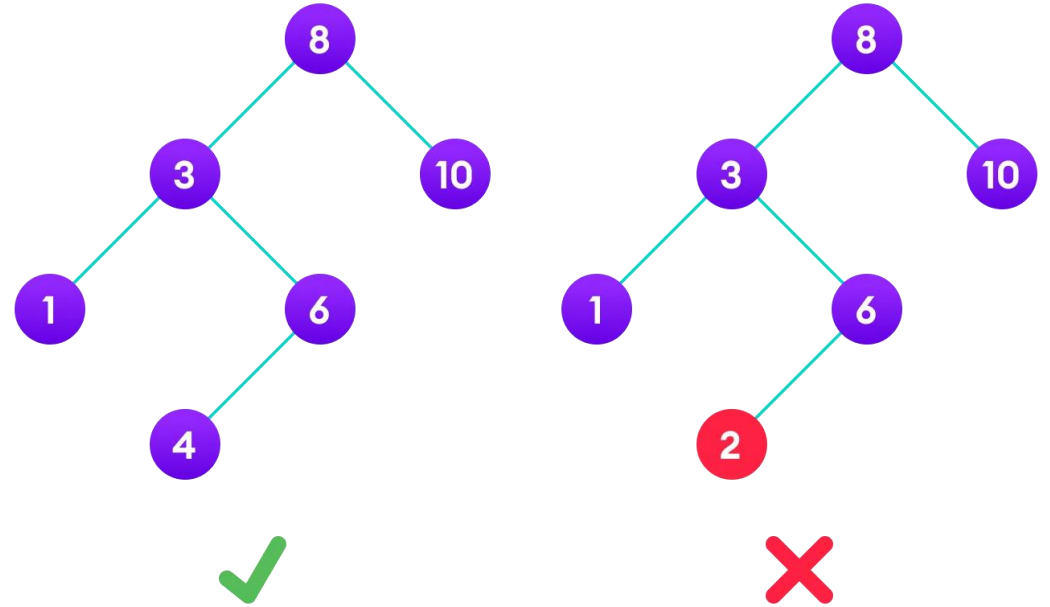


An example of a binary search tree

Binary Search Tree(BST)

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

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

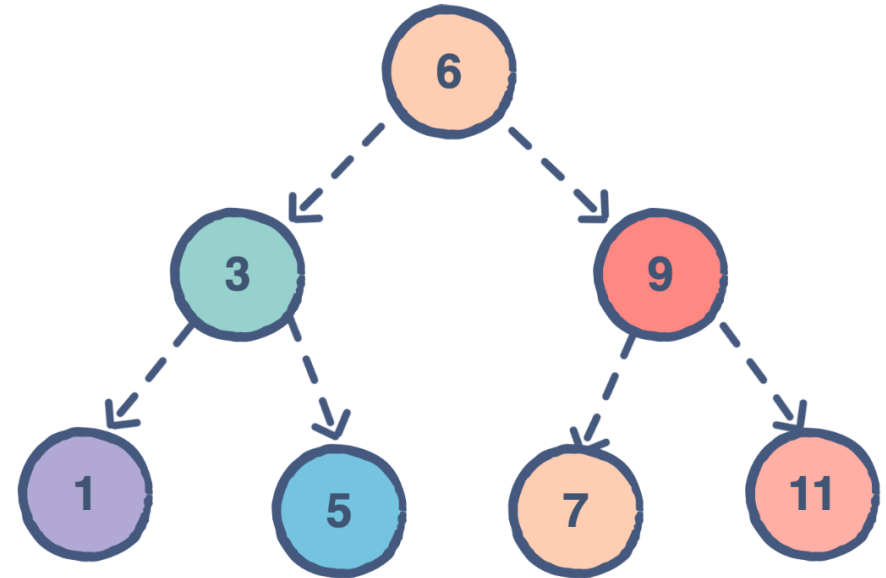


Binary Search Tree(BST)

There are two basic operations that you can perform on a binary search tree:

Search Operation

- If the value is below the root, we can say for sure that the value is not in the right subtree; we need to only search in the left subtree and if the value is above the root, we can say for sure that the value is not in the left subtree; we need to only search in the right subtree.



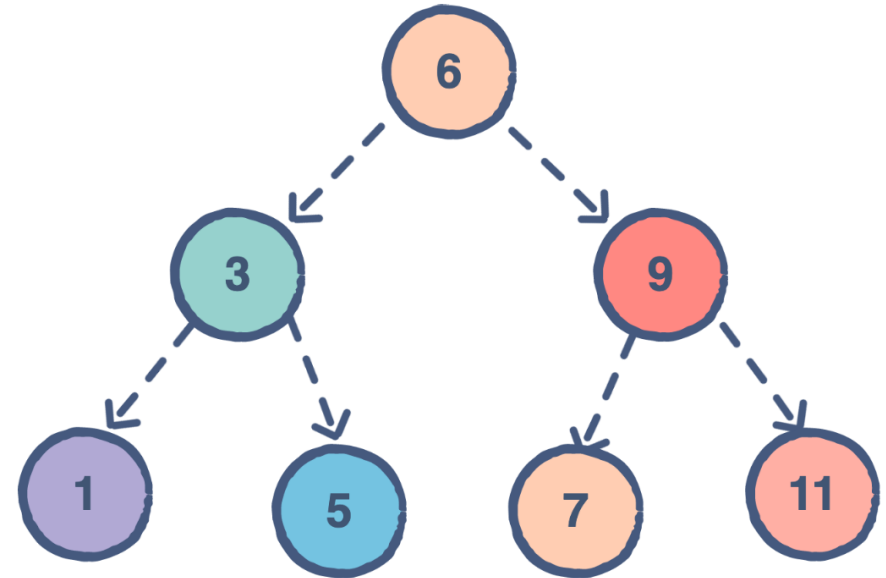
An example of a binary search tree

Binary Search Tree(BST)

There are two basic operations that you can perform on a binary search tree:

Insert Operation

- Inserting a value in the correct position is similar to searching because we try to maintain the rule that the left subtree is lesser than root and the right subtree is larger than root.
- We keep going to either right subtree or left subtree depending on the value and when we reach a point left or right subtree is null, we put the new node there.



An example of a binary search tree

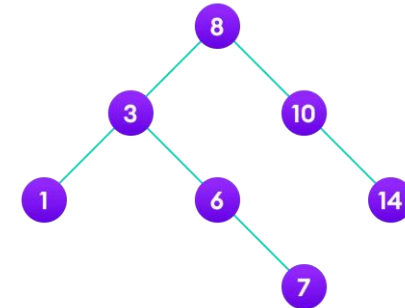
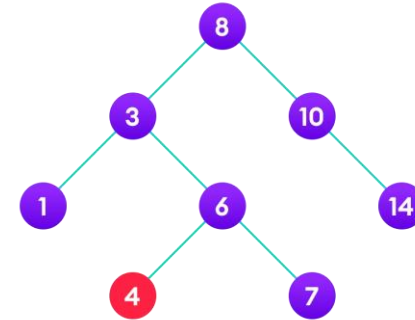
Binary Search Tree(BST)

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.

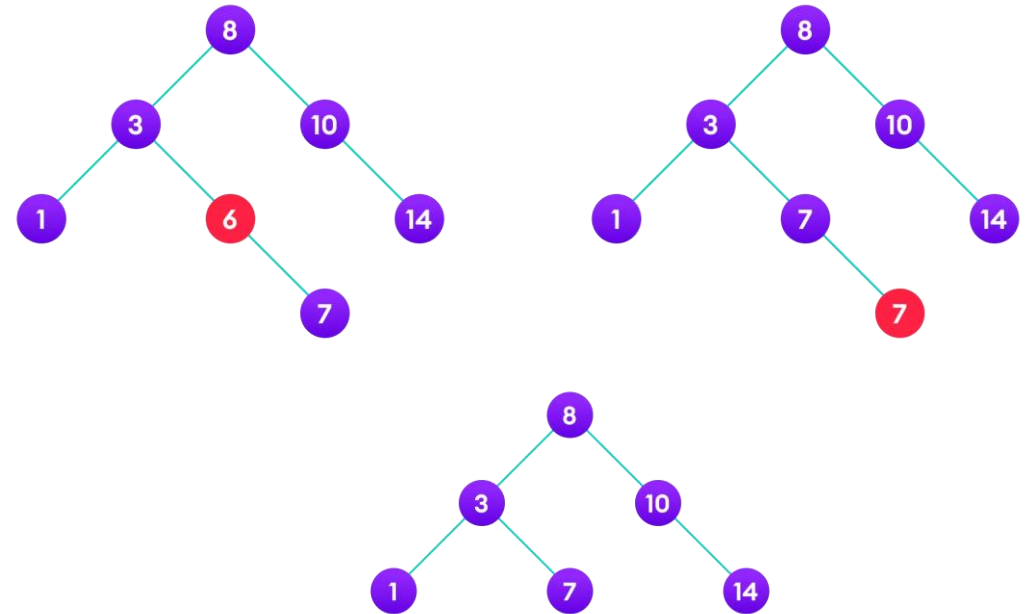


Binary Search Tree(BST)

Deletion Operation

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.



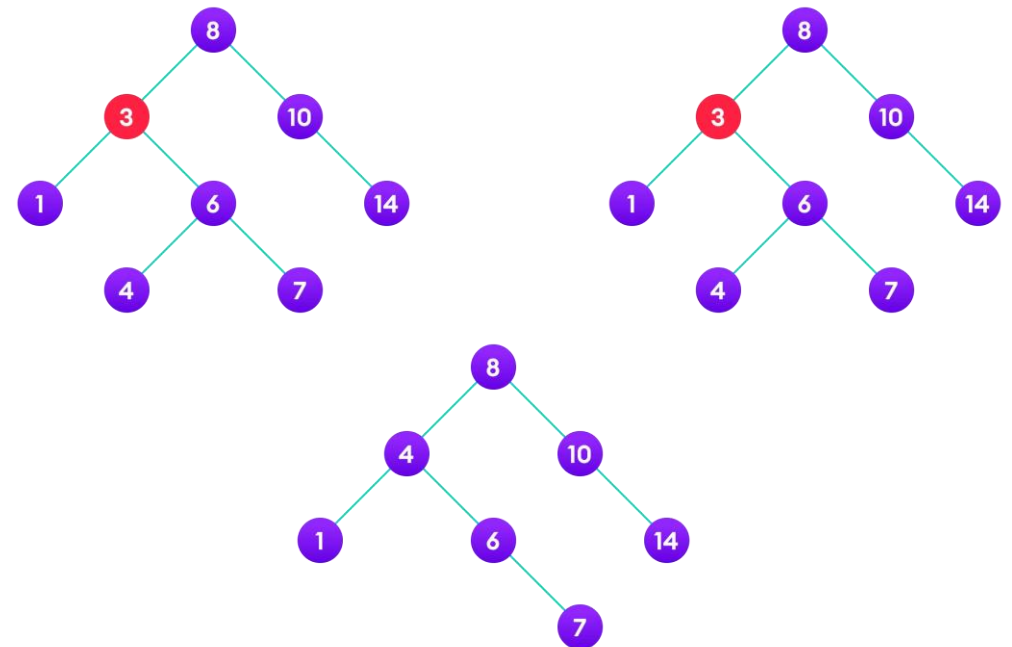
Binary Search Tree(BST)

Deletion Operation

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.



Binary Search Tree(BST)

Binary Search Tree Complexities

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.

Operation	Best Case Complexity	Average Case Complexity	Worst Case Complexity
Search	$O(\log n)$	$O(\log n)$	$O(n)$
Insertion	$O(\log n)$	$O(\log n)$	$O(n)$
Deletion	$O(\log n)$	$O(\log n)$	$O(n)$

يعطيك الله العافية

References

<https://www.programiz.com/dsa/trees>

