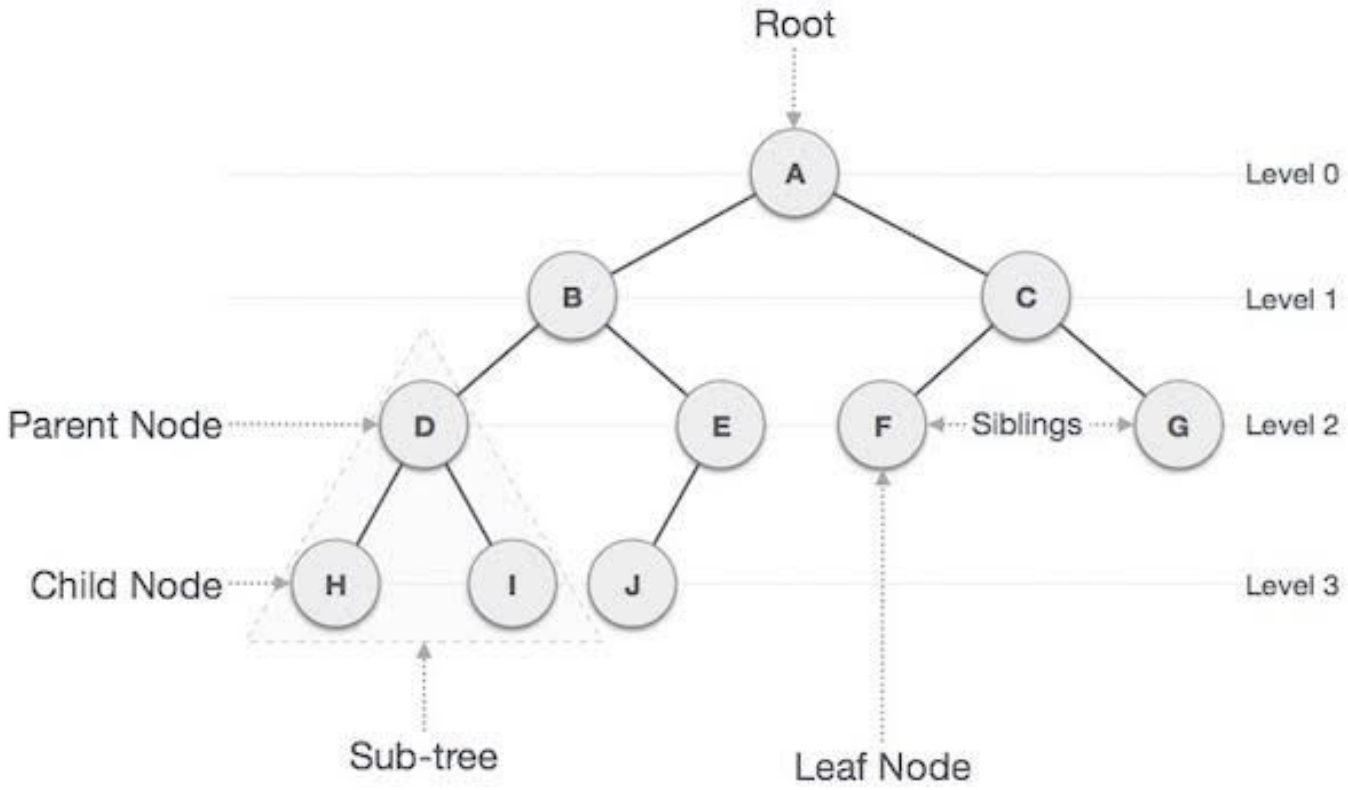
Competitive Programming

Lec 10-11
Trees

Bhavik Dhandhalya

Tree



- Root Node
- Parent Node
- Child Nodes
- Leaf Nodes
- Internal Nodes
- Degree of Node [in-degree, out-degree]

Tree

Can Tree have more than 1 Root ?

- No

Can Tree have more than 2 child ?

- Yes, Trees with 2 children called binary, trees with N children called N-ary trees.

Difference between Graph and Tree

- Graph can have cycle

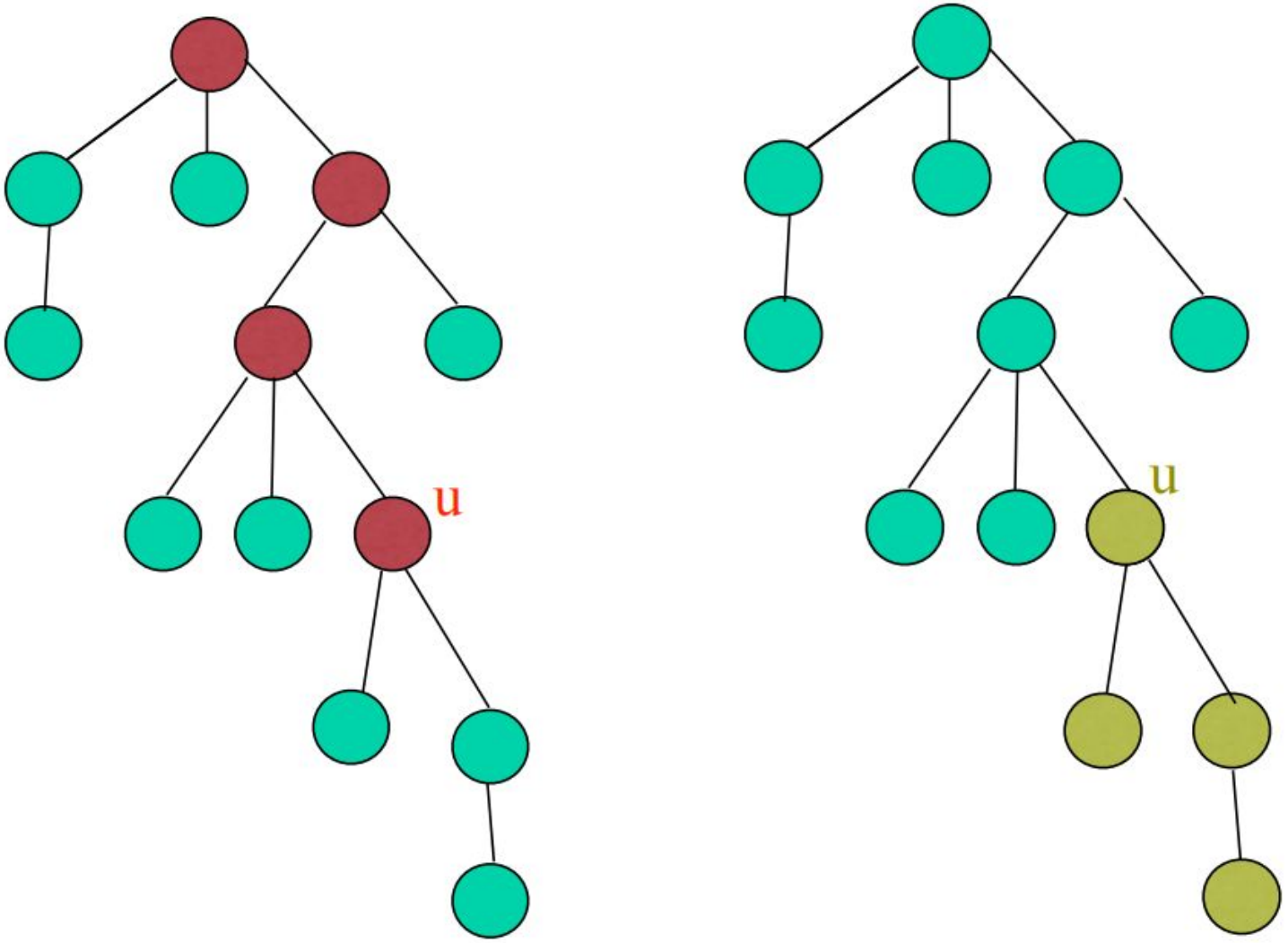
Type of Edges,

1. Forward Edges
2. Backward edges
3. Sibling edges

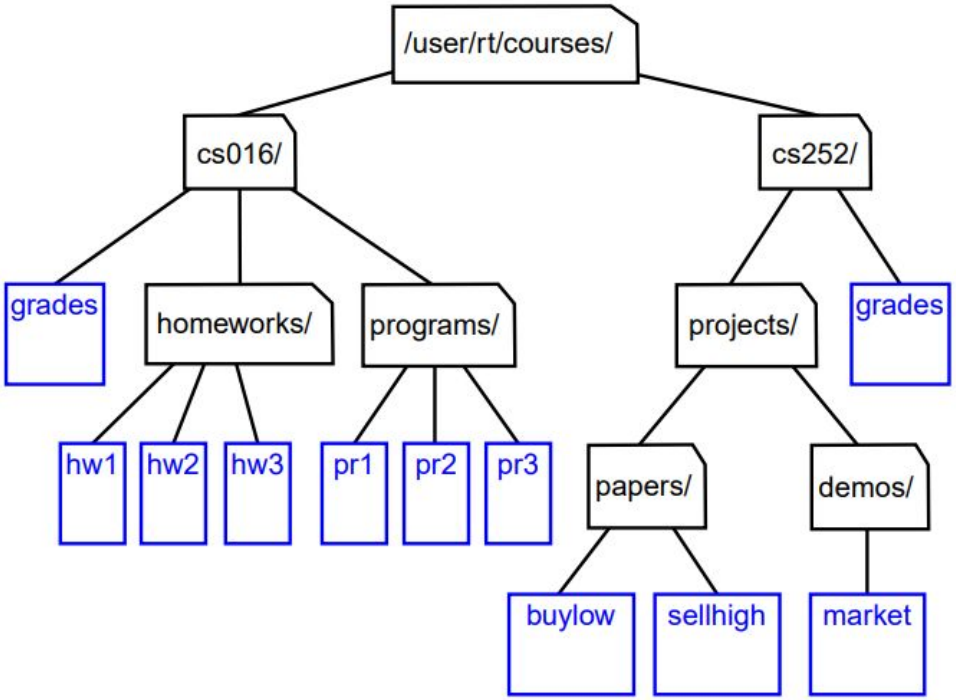
Total Edges = Total Nodes - 1

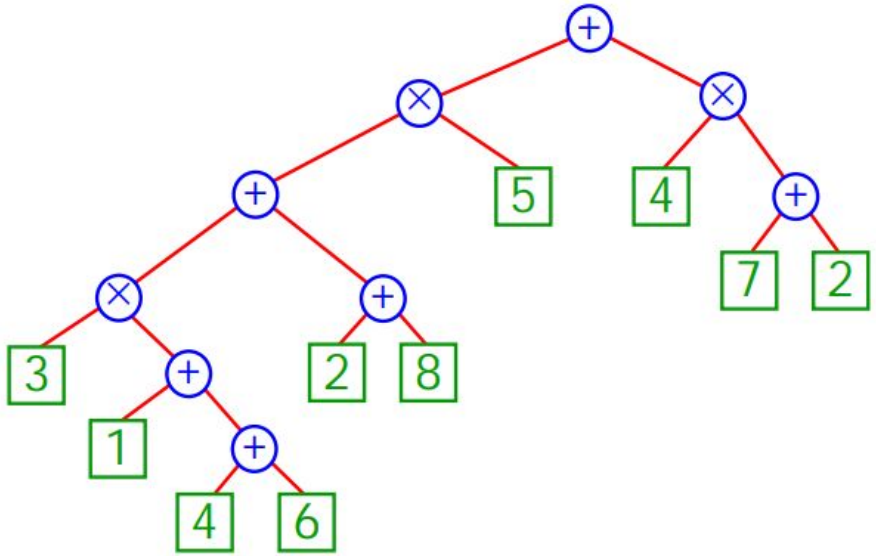
Nodes at **i-th** level = 2^{i}

Ancestors & Descendants of u



Tree Example





Bhavik Dhandhalya

Types of Binary Trees

1. Binary Tree

2. Complete binary tree
   - All leaf nodes are at same level or in other words, all internal nodes have degree two

3. Strict or Full binary tree
   - If every non-leaf node in a binary tree has non-empty left and right subtrees

Watch these videos.

Tree Representation

1. Using Array
2. Using Pointers ✓

C++ Implementation

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

Python Implementation

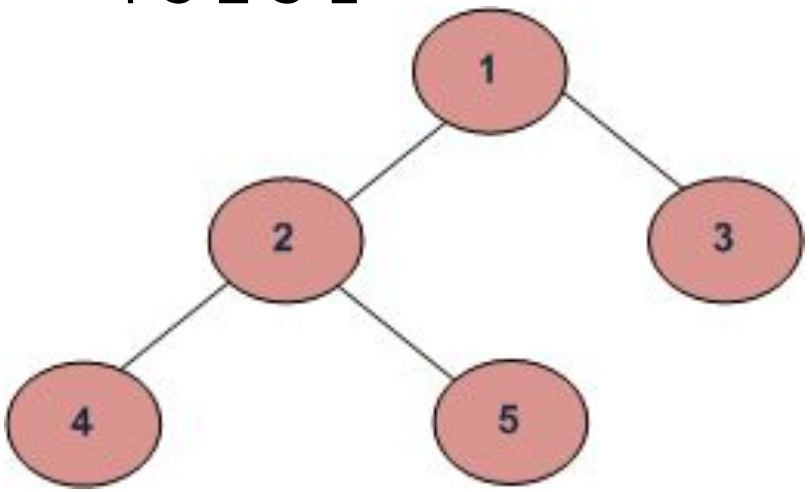
```
class Node:
    def __init__(self,key):
        self.left = None
        self.right = None
        self.val = key
```

Java Implementation

```
class Node
{
    int key;
    Node left, right;

    public Node(int item)
    {
        key = item;
        left = right = null;
    }
}
```

Tree Traversal

1. Inorder
   <Left Node, Root Node, Right Node>
   = 4 2 5 1 3
2. Preorder
   <Root Node, Left Node, Right Node>
   = 1 2 4 5 3
3. Postorder
   <Left Node, Right Node, Root Node>
   = 4 5 2 3 1



Tree Traversal

1. Inorder

```
void Inorder(struct Node* node) {
    if (node == NULL)
        return;

    /* first recur on left child */
    Inorder(node->left);

    /* then print the data of node */
    cout << node->data << " ";

    /* now recur on right child */
    Inorder(node->right);
}
```

Tree Traversal

2. Preorder

```
void Preorder(struct Node* node) {
    if (node == NULL)
        return;

    /* first print data of node */
    cout << node->data << " ";

    /* then recur on left subtree */
    Preorder(node->left);

    /* now recur on right subtree */
    Preorder(node->right);
}
```

Tree Traversal

3. Postorder

```
void Postorder(struct Node* node) {
    if (node == NULL)
        return;

    // first recur on left subtree
    Postorder(node->left);

    // then recur on right subtree
    Postorder(node->right);

    // now deal with the node
    cout << node->data << " ";
}
```

Tree Traversal

3. Postorder

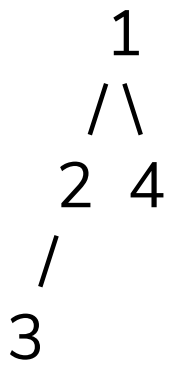
```
void Postorder(struct Node* node) {
    if (node == NULL)
        return;

    // first recur on left subtree
    Postorder(node->left);

    // then recur on right subtree
    Postorder(node->right);

    // now deal with the node
    cout << node->data << " ";
}
```

Problems

Max Depth or Height of Binary Tree

Given a binary tree, find its maximum depth.

The maximum depth of a binary tree is the number of nodes along the longest path from the root node down to the farthest leaf node.

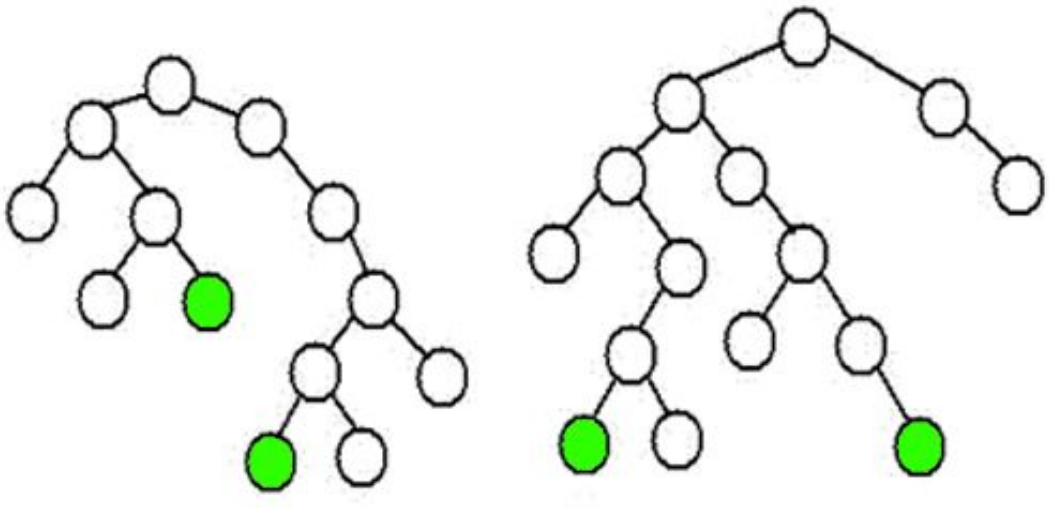


Ans: 3

Solution

Diameter of Binary Tree

Given a Binary Tree, find diameter of it.
The diameter of a tree is the number of nodes on the longest path between two leaves in the tree. The diagram below shows two trees each with diameter nine, the leaves that form the ends of a longest path are shaded (note that there is more than one path in each tree of length nine, but no path longer than nine nodes).



Diameter, 9 nodes, through root

Diameter, 9 nodes, NOT through root

Solution

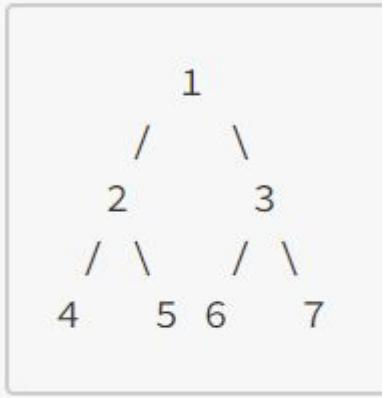
Bhavik Dhandhalya

Invert the Binary Tree, Google

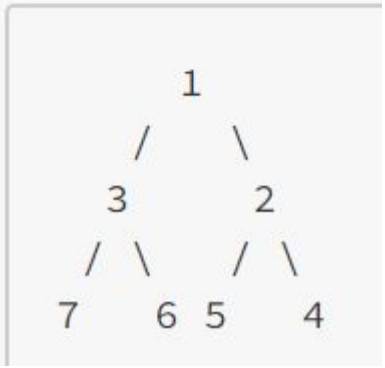
Given a binary tree, invert the binary tree and return it.

Example :

Given binary tree



invert and return

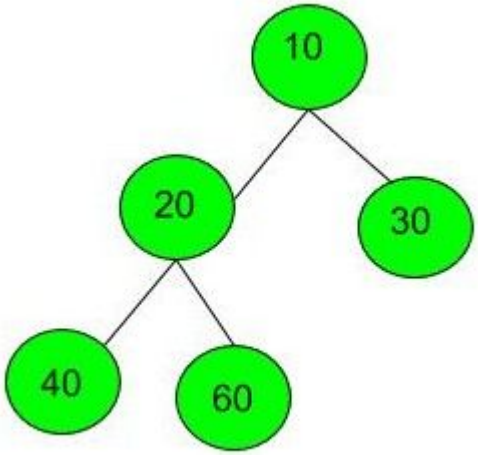


“Homebrew story”

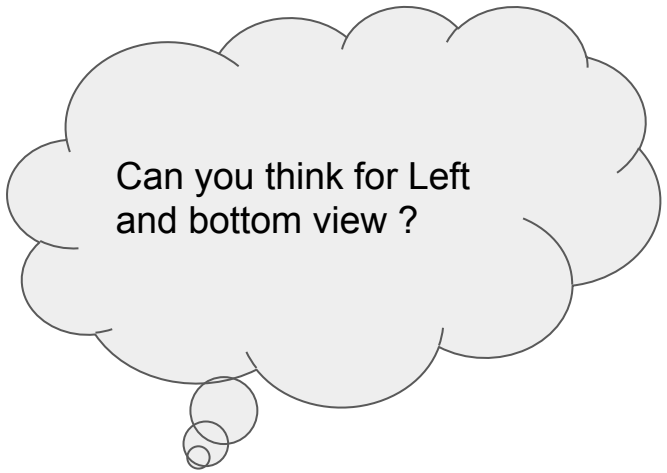
Solution

Bhavik Dhandhalya

Print a Binary Tree in Vertical Order



Vertical order = 40 20 10 60 30

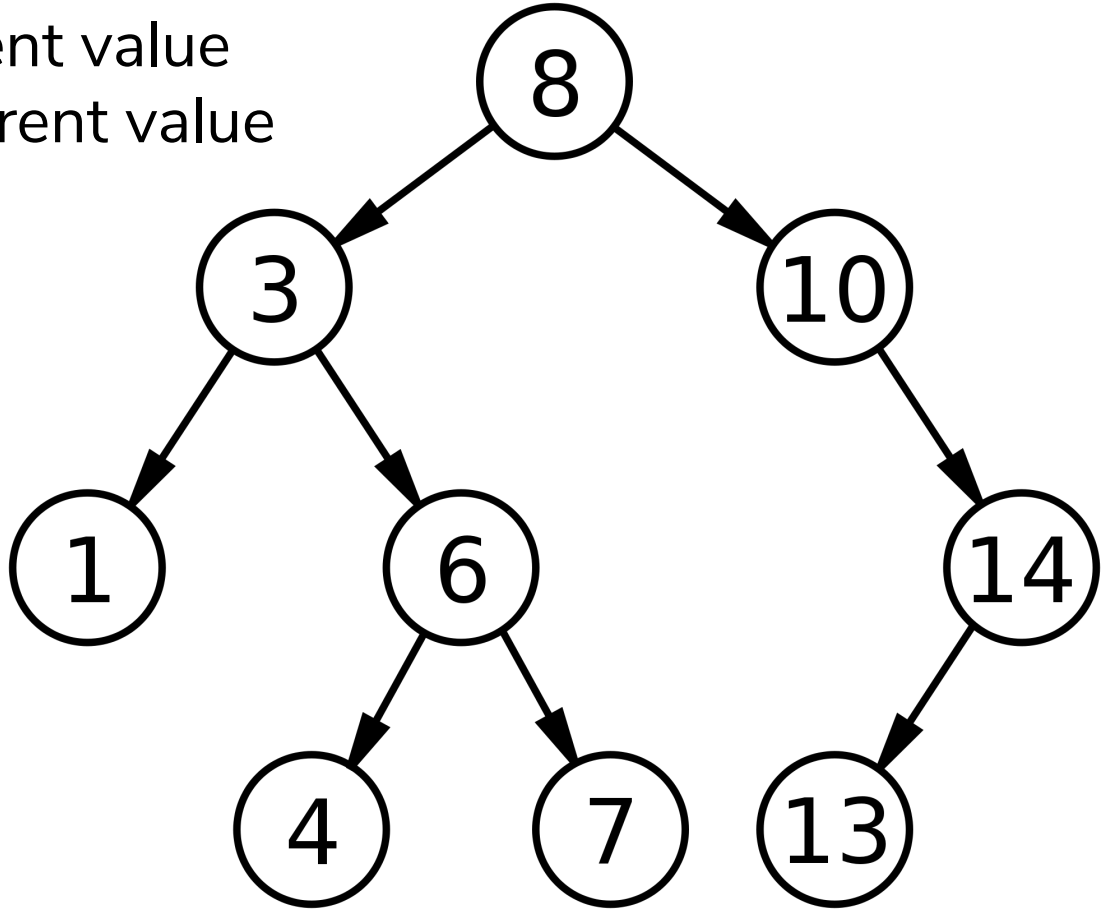


Solution

BST

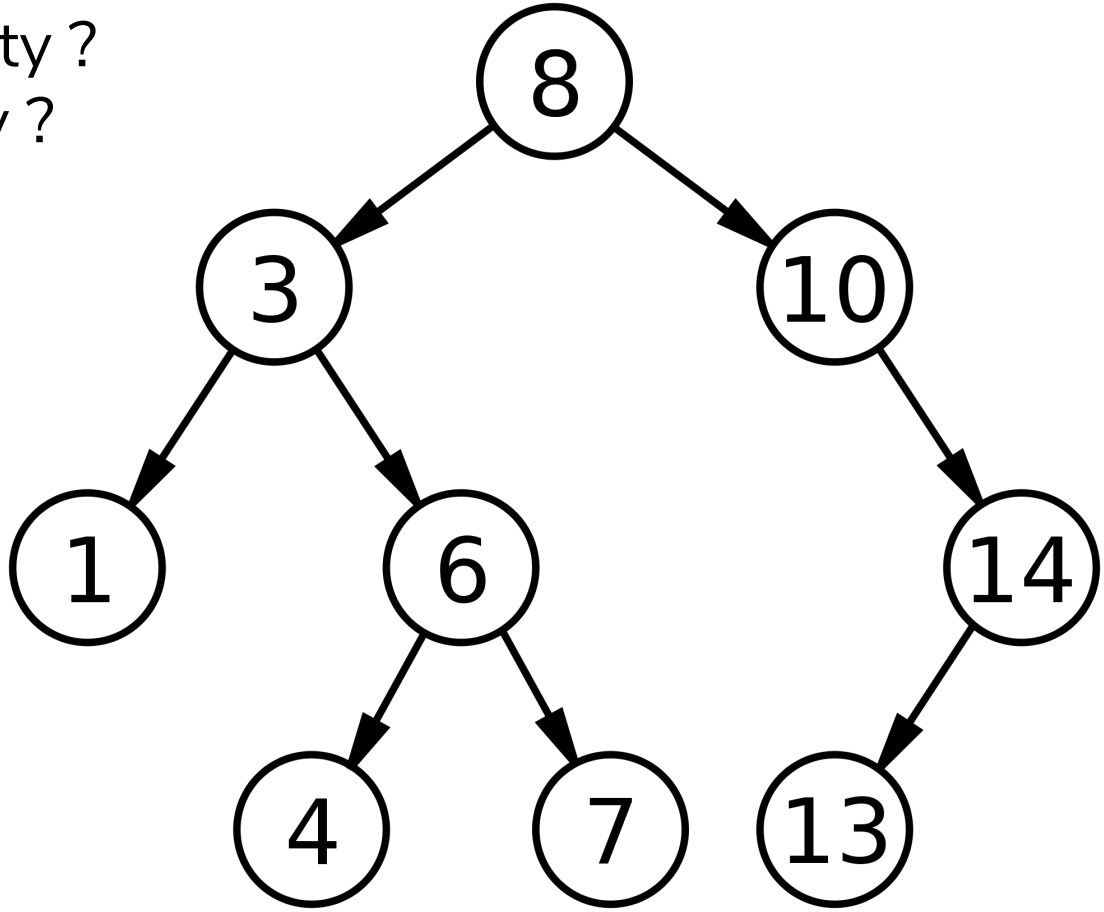
Binary Search Tree

Left child value < parent value
Right child value > parent value



BST

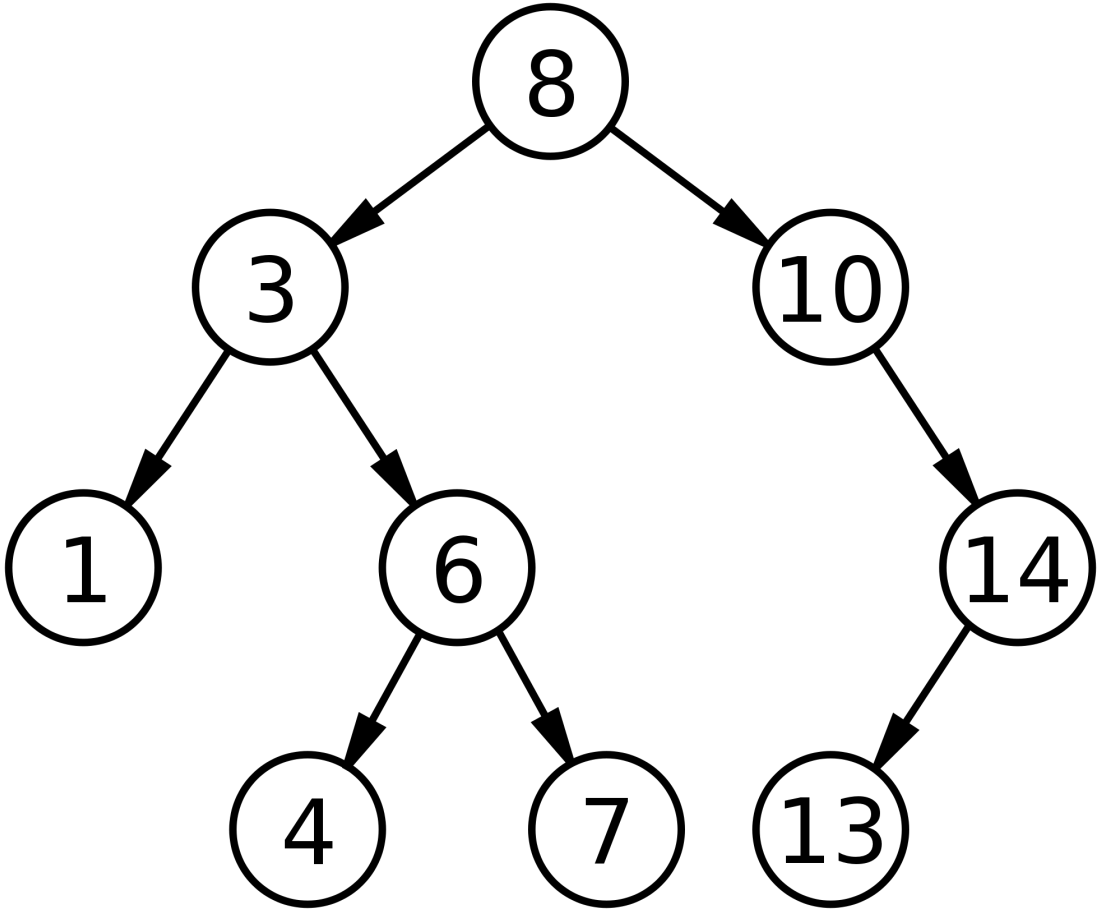
- How to construct BST ?
- Why BST ? because Insertion, search
- Insertion complexity ?
- Search Complexity ?



BST

Q. How to avoid worst case O(N) complexity ?

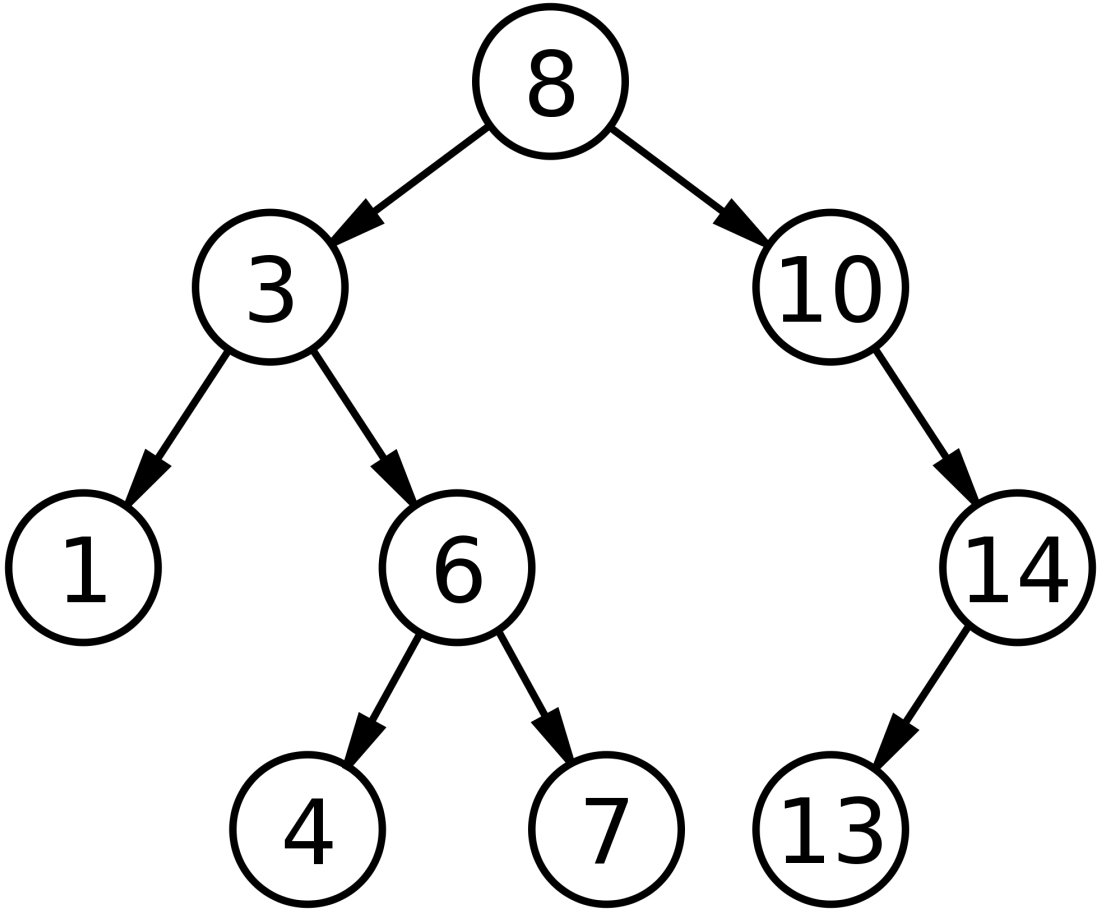
A. Use AVL Tree.



BST

Inorder Traversal : 1 3 4 6 7 8 10 13 14

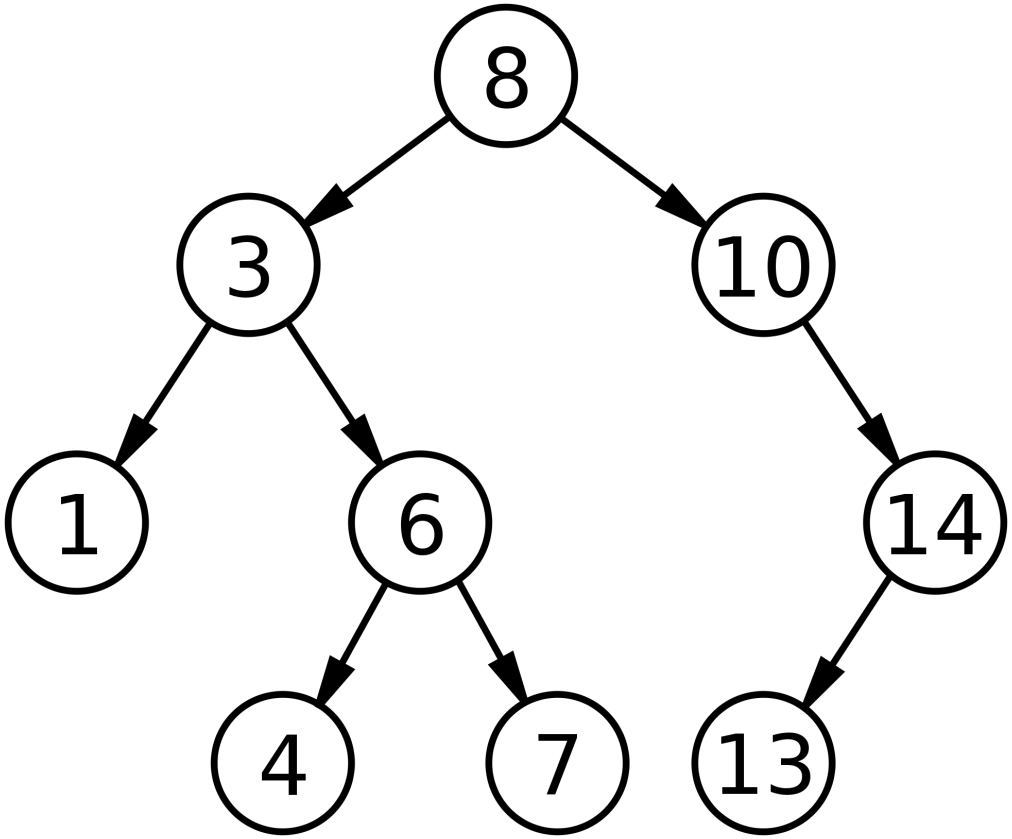
Noticed something ?



LCA in BST

Find Lowest Common Ancestor.
Note: No duplicates in BST.

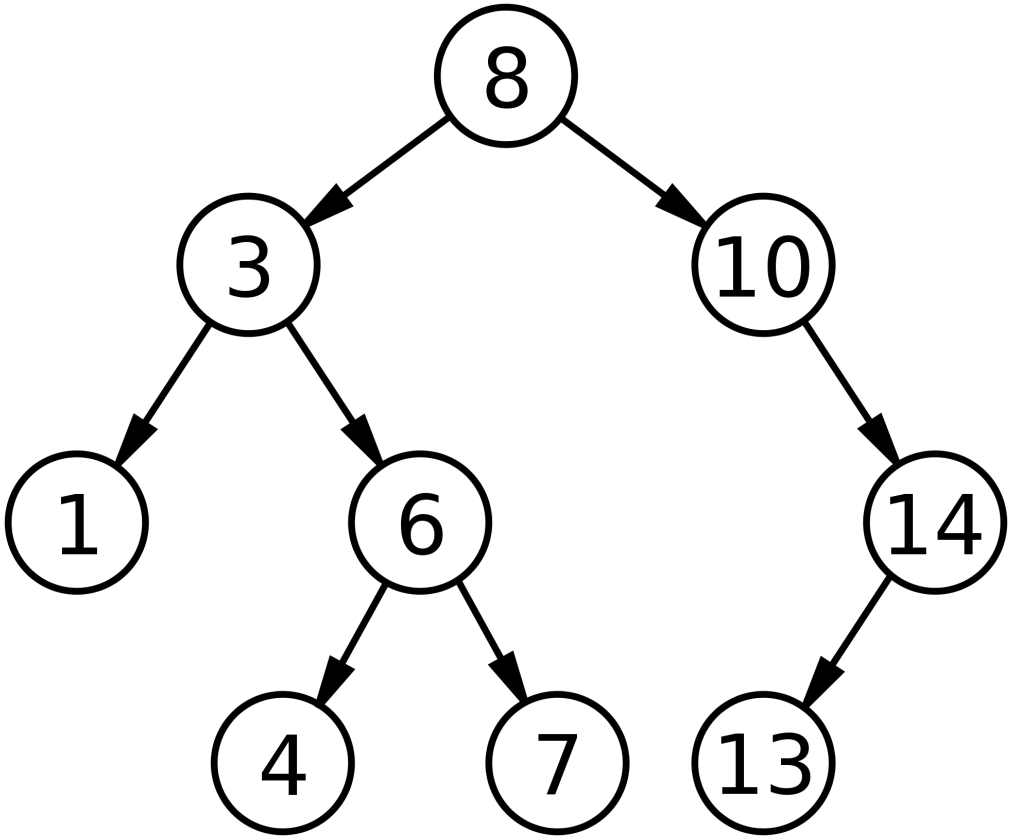
For 1 13 LCA = 8
For 3 7 LCA = 3
For 4 7 LCA = 6
For 1 6 LCA = 3



Solution

Check BST

Given a binary tree, return true if it is BST, else false



Solution

Inorder, Preorder, Postorder Iterative

1. Inorder
2. Preorder
3. Postorder

Homework part 1

Before you start solving Homework part 2, you should be clear with implementation of binary tree and BST.

Task:

1. Implement Binary Tree & Binary Search Tree.
2. Insert Operation
3. Delete Operation
   - Delete Leaf Node
   - Delete Non-leaf Node/internal Node
   - Delete Root Node

Homework part 2

Count Leaves in Binary Tree	Easy, G4G	
Min Depth of Binary Tree	Easy, Interviewbit	Similar to Max depth we discussed
Maximum path sum	Easy, G4G	Use concept of height for sum
Inorder Traversal	Medium, Interviewbit	Don't use recursion, Very Important for Interview
Postorder Traversal	Medium, Interviewbit	Don't use recursion, Very Important for Interview
Preorder Traversal	Medium, Interviewbit	Don't use recursion, Very Important for Interview
Symmetric Binary Tree	Medium, Interviewbit	Similar to Inverted tree we discussed
Left View of Binary Tree	Easy, Geeks4geeks	
Bottom View of Binary Tree	Easy, Geeks4geeks	
Balanced Binary Tree	Easy, interviewbit	Use height to find out
Serialize and Deserialize a Binary Tree	Medium, G4G	After homework part 1, it should be easy

Homework part 2

Sorted Array To Balanced BST	Hard, Interviewbit	
Binary Tree From Inorder And Postorder	Hard, Interviewbit	
Construct Binary Tree From Inorder And Preorder	Hard, Interviewbit	
2-Sum Binary Tree	Hard, Interviewbit	Variation asked in Goldman Sachs
Recover Binary Search Tree	Hard, Interviewbit	
Least Common Ancestor	Hard, Interviewbit	
ZigZag Level Order Traversal BT	Hard, Interviewbit	
Flatten Binary Tree to Linked List	Hard, Interviewbit	
Populate Next Right Pointers Tree	Hard, Interviewbit	