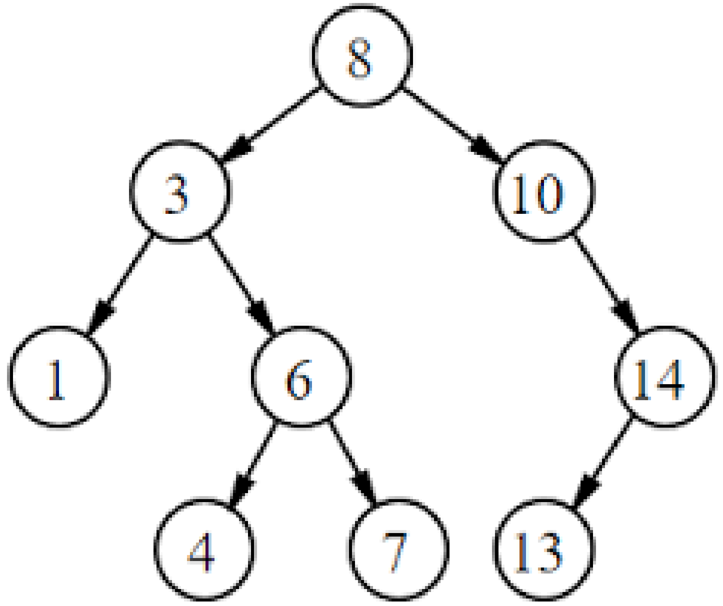
Data Structures and Analysis of Algorithms
CST 225-3

Binary Tree

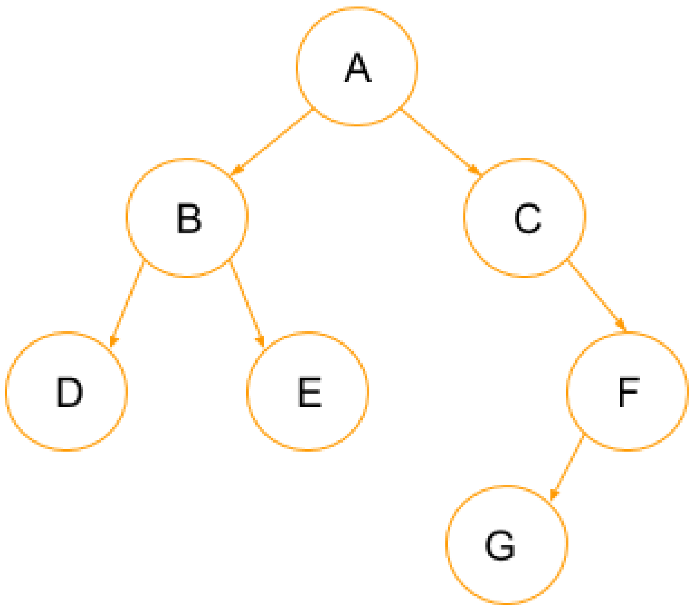
Uva Wellassa University

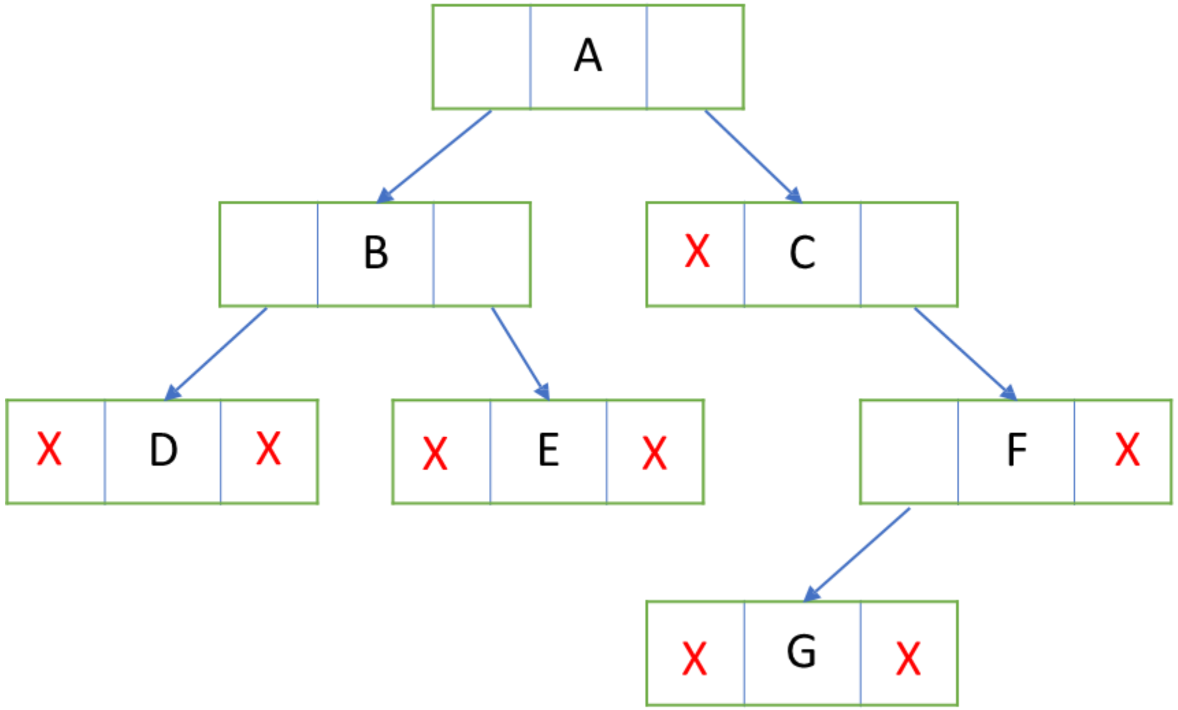
What is Binary Tree?

- A binary tree is a non-linear data structures where in each node there can be only 0, 1 or 2 child nodes.
- There can be maximum of two child nodes for each parent node.
- Each node contains;
  - Data
  - A pointer to the left child
  - A pointer to the right child



Logical Representation





Properties of Binary Tree

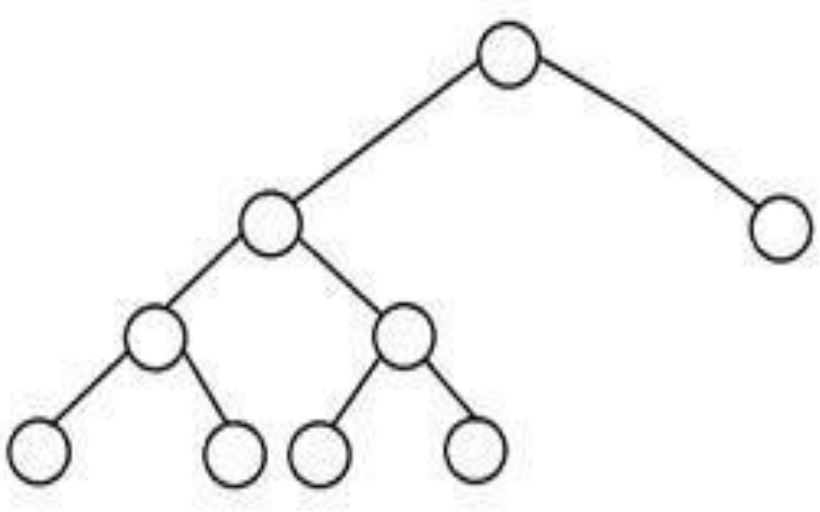
- Maximum number of nodes possible at any level i is $\mathbf{2^i}$.
- Maximum number of nodes of height h is $\mathbf{2^{h+1} - 1}$.
- Minimum number of nodes of height h is **h+1**.
- Maximum height given n number of min nodes is **n-1**.
- Minimum height given n number of max nodes is $\lfloor \mathbf{\log_2(n+1)} \rfloor \mathbf{-1}$.

Types of Binary Tree

- Full/ Proper / Strict
- Complete
- Perfect
- Degenerate
- Balanced / AVL

Full/Proper/Strict Binary Tree

- Every node in the tree has either 0 or 2 children.
- Each node have exactly 2 children except leaf node.

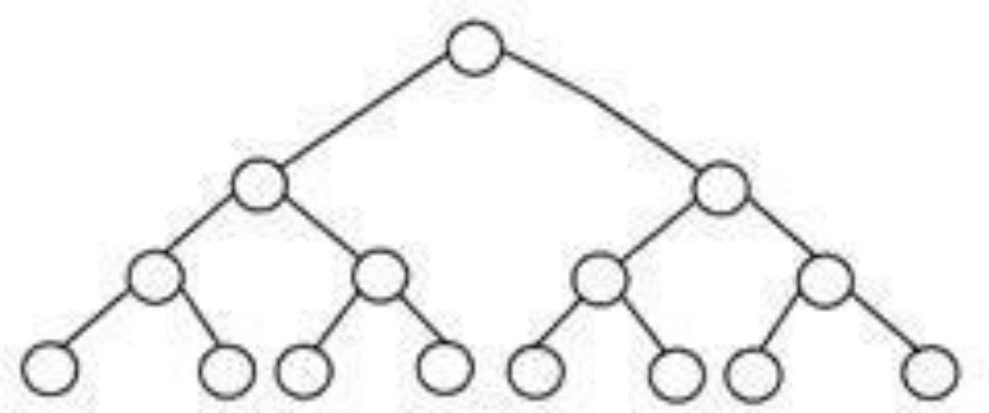


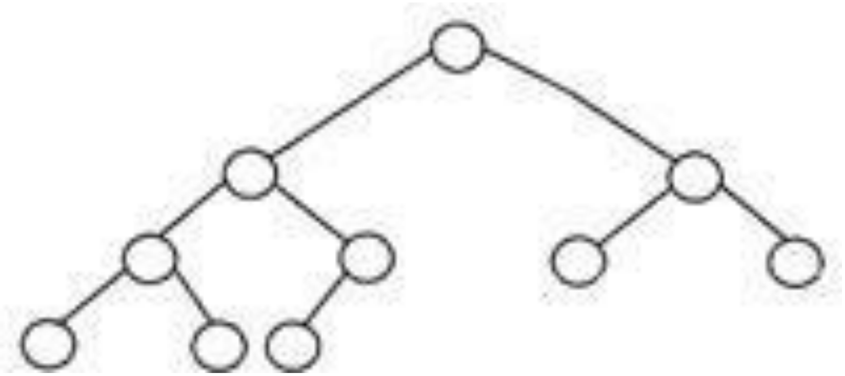
Full/Proper/Strict Binary Tree

- No. of leaf nodes = No. of internal nodes +1.
- Maximum number of nodes of height h is $\mathbf{2^{h+1} -1}$.
- Minimum number of nodes of height h is **2h+1**.
- Minimum height given n number of max nodes is $\lfloor \mathbf{\log2(n+1)} \rfloor \mathbf{-1}$.
- Maximum height given n number of min nodes is **(n-1)/2**.

Complete Binary Tree

- All the levels are completely filled except the last level.
- Last level has nodes as left as possible.



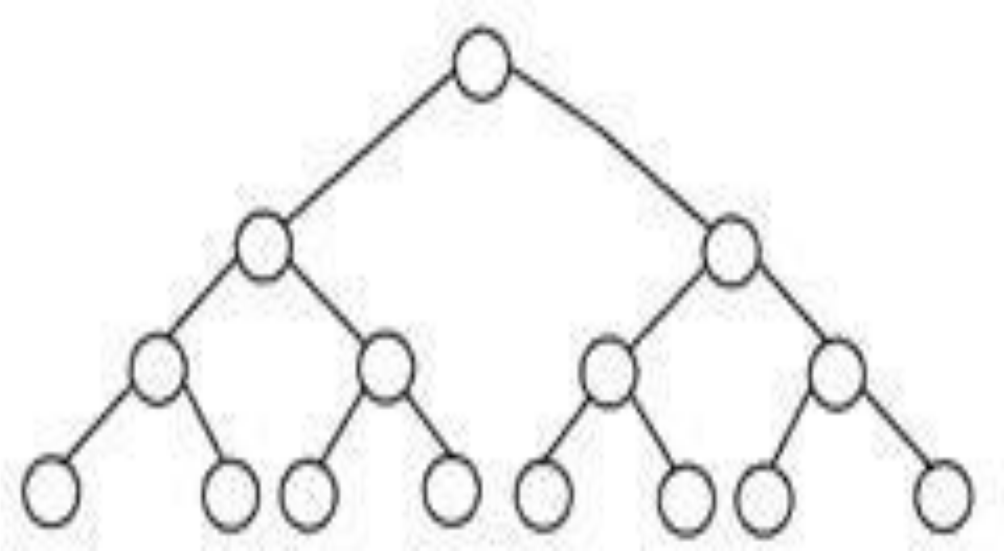


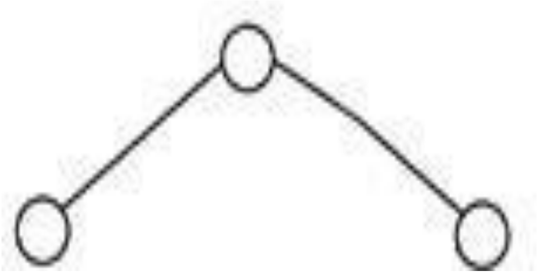
Complete Binary Tree

- Maximum number of nodes of height h is $\mathbf{2^{h+1} -1}$.
- Minimum number of nodes of height h is $\mathbf{2^h}$.
- Minimum height given n number of max nodes is $\lfloor$ **log2(n+1)** $\rfloor$ **-1**.
- Maximum height given n number of min nodes is **log n**.

Perfect Binary Tree

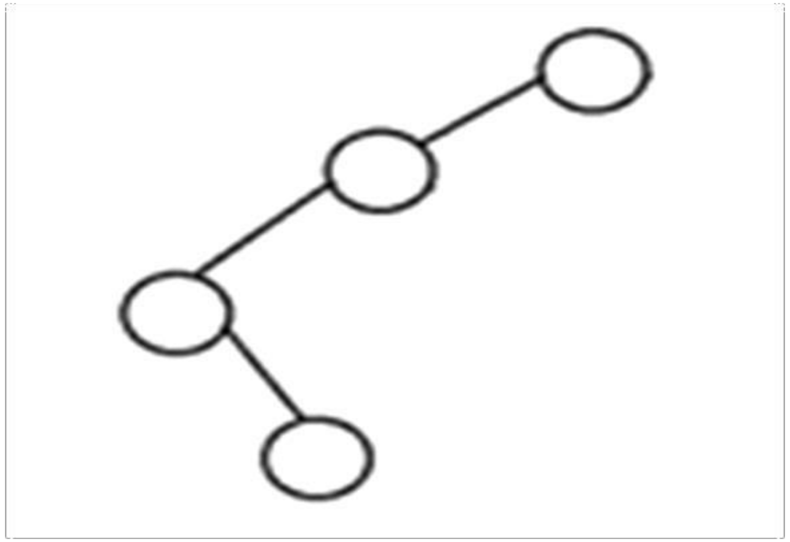
- All internal nodes have 2 children.
- All leaf nodes should be at same level.
- All perfect binary trees are full and complete binary trees.





Degenerate Binary Tree

- All the internal nodes are having only one child.



Balanced Binary Tree

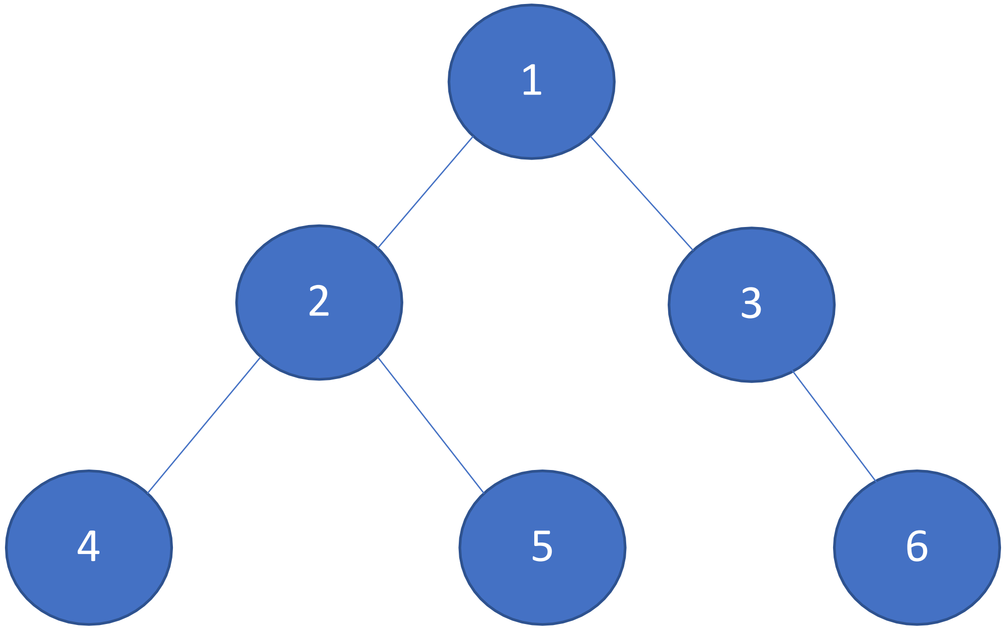
- Both the left and right trees differ by at most 1.
- Ex: AVL

Tree Traversal

- Traversing is the way of accessing nodes of a tree in different ways.
- There are different approaches like;
  - Breadth First Traversal
  - Depth First Traversal
    - In-order
    - Pre-order
    - Post-order

Breadth First Tree Traversal

- Each node is accessed level by level from left to right.



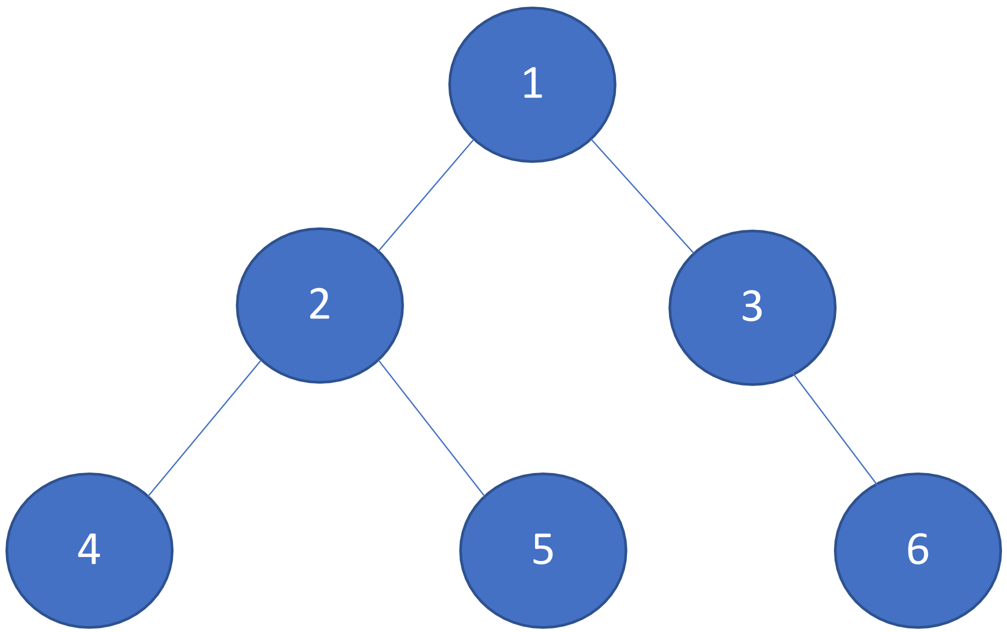
Breadth First (Level First) : 1 2 3 4 5 6

Depth First Tree Traversal

- Visit nodes by depth.
  - ✓ **In-order**-Left Root Right
  - ✓ **Pre-order**-Root Left Right
  - ✓ **Post-order**-Left Right Root

In-order Tree Traversal

- Each node is processed between subtrees.

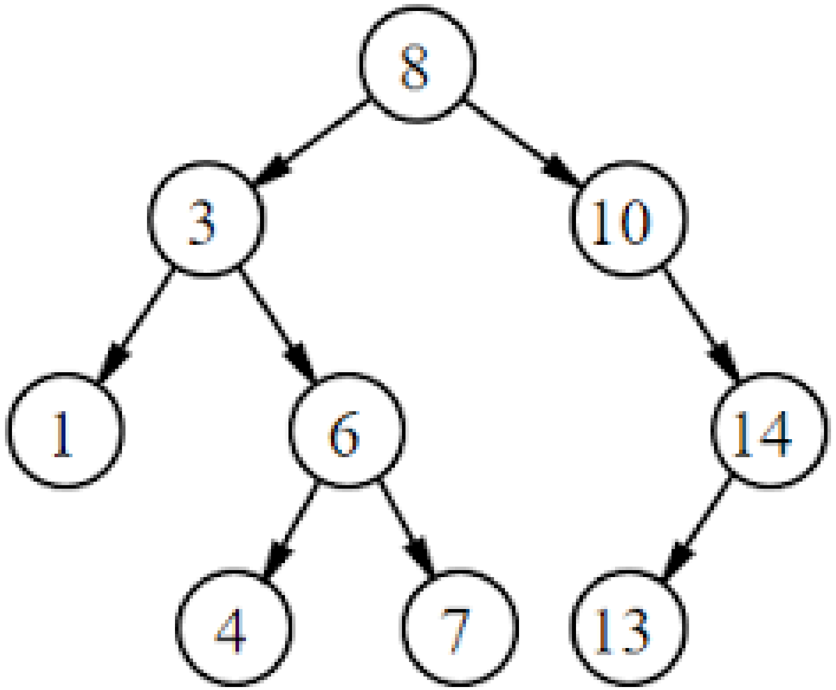


Inorder (Left, Root, Right) : 4 2 5 1 3 6

Algorithm
1. Traverse the left subtree
2. Visit the root
3. Traverse the right subtree

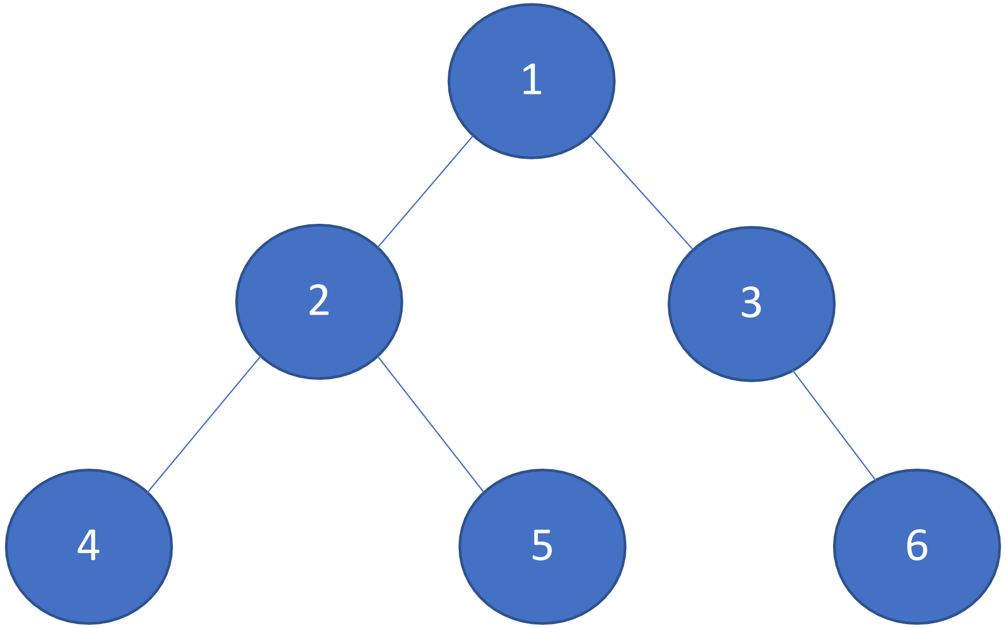
Exercise

Find in-order traversal.



Pre-order Tree Traversal

- Each node is processed before its sub-trees.

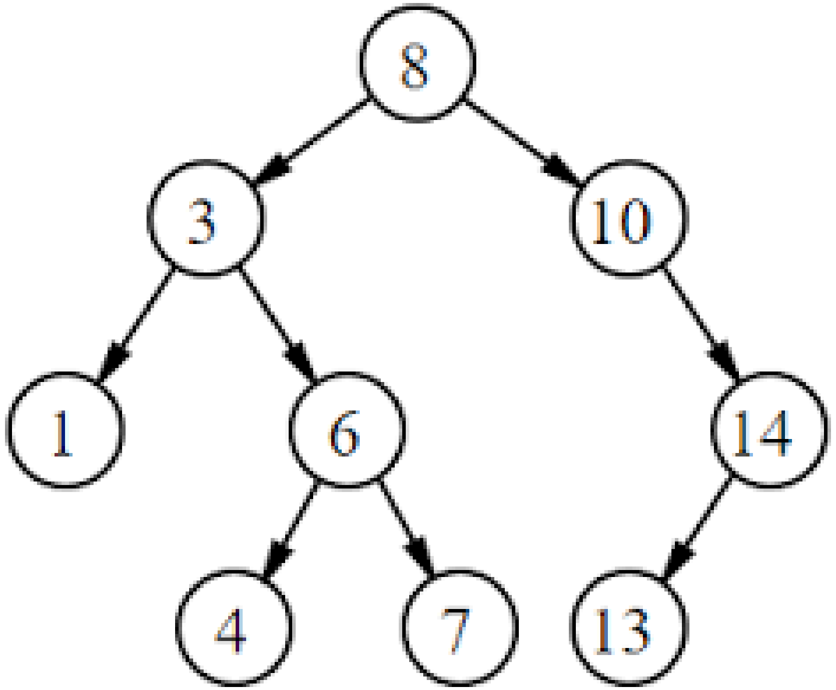


Preorder (Root, Left, Right) : 1 2 4 5 3 6

Algorithm
1. Visit the root
2. Traverse the left subtree
2. Traverse the right subtree

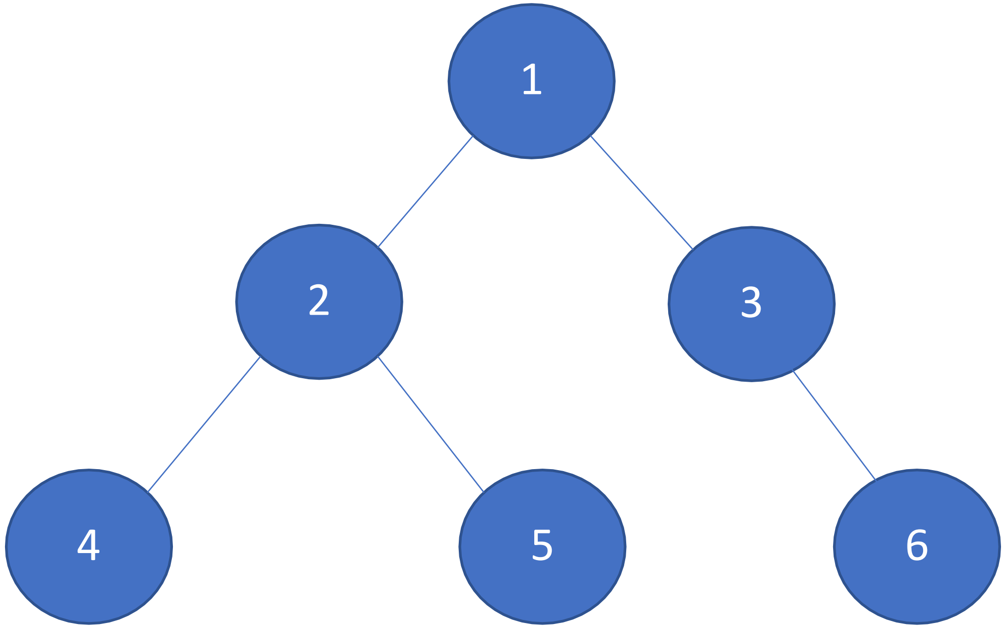
Exercise

Find pre-order traversal.



Post-order Tree Traversal

- Each node is processed after its subtrees.

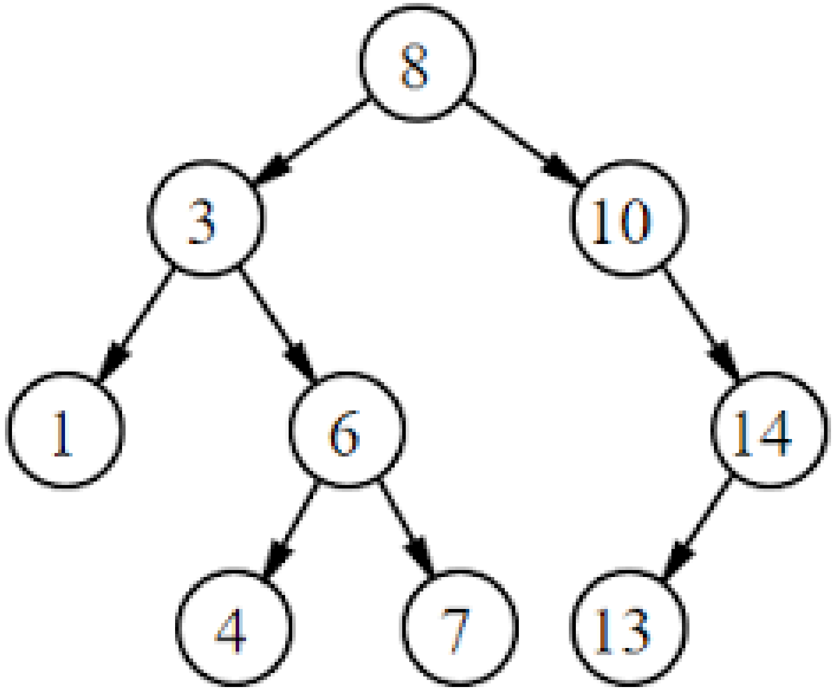


Postorder (Left, Right, Root) : 4 5 2 6 3 1

Algorithm
1. Traverse the left subtree
2. Traverse the right subtree
3. Visit the root

Exercise

Find post-order traversal.



Questions?