# DESIGN AND ANALYSIS OF ALGORITHMS
# UE19CS251

**Shylaja S S**
Department of Computer Science
& Engineering

# DESIGN AND ANALYSIS OF ALGORITHMS

## Binary Tree

Major Slides Content: Anany Levitin

**Shylaja S S**

Department of Computer Science & Engineering

## Binary Tree

- A *binary tree* **T** is defined as a finite set of nodes that is either empty or consists of a root and two disjoint binary trees $\mathbf{T_L}$ and $\mathbf{T_R}$ called as the left and right subtree of the root
- The definition itself divides the Binary Tree into two smaller structures and hence many problems concerning the binary trees can be solved using the Divide – And – Conquer technique
- The binary tree is a Divide – And – Conquer ready structure ☺

## Height of a Binary Tree

- Height of a Binary Tree: Length of the longest path from root to leaf

```
ALGORITHM  Height(T)
//Computes recursively the height of a binary tree
//Input: A  binary tree T
//Output: The height of T
if T = Ø  return -1
else return max(Height($T_L$), Height($T_R$))+ 1
```

## Height of a Binary Tree - Analysis

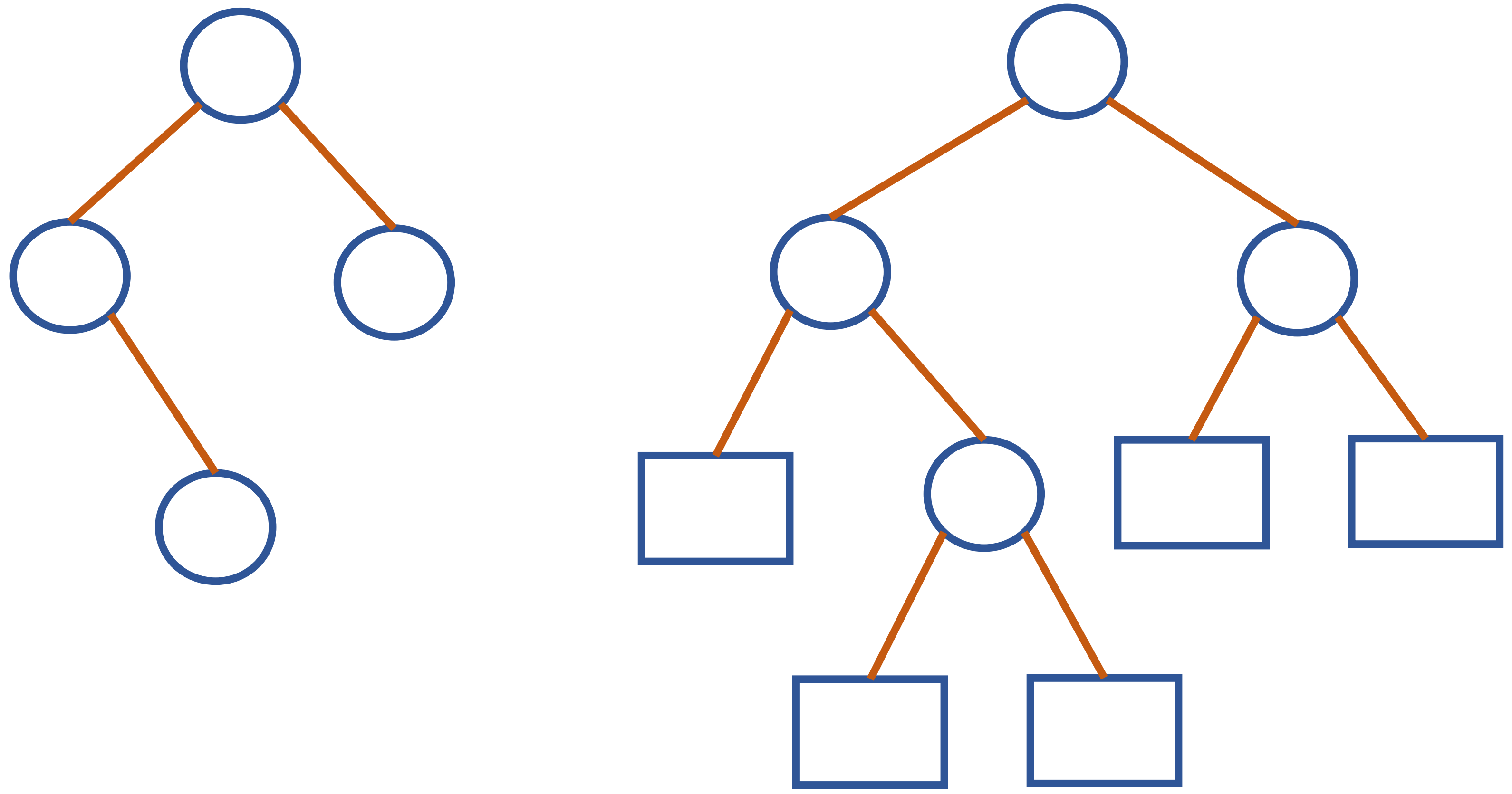
- The measure of input's size is the number of nodes in the given binary tree. Let us represent this number as n(T)
- Basic Operation: Addition
- The recurrence relation is setup as follows:

$$A(n(T)) = A(n(T_L)) + A(n(T_R)) + 1, \quad \text{for } n(T) > 0$$

$$A(0) = 0$$

## Height of a Binary Tree - Analysis

- In the analysis of tree algorithms, the tree is extended by replacing empty subtrees by special nodes called external nodes



## Height of a Binary Tree - Analysis

- x – Number of external nodes
- n – Number of internal nodes

$$x = n + 1$$

- The number of comparisons to check whether a tree is empty or not:

$$C(n) = n + x = 2n + 1$$

- The number of additions is:

$$A(n) = n$$

## Binary Tree Traversals

- The three classic traversals for a binary tree are inorder, preorder and postorder traversals

- In the preorder traversal, the root is visited before the left and right subtrees are visited (in that order)
- In the inorder traversal, the root is visited after visiting its left subtree but before visiting the right subtree
- In the postorder traversal, the root is visited after visiting the left and right subtrees (in that order)

## Binary Tree Traversals

```
Algorithm Inorder(T)
if T $\neq \varnothing$
    Inorder(Tleft)
    print(root of T)
    Inorder(Tright)
```

```
Algorithm Preorder(T)
if T $\neq \varnothing$
    print(root of T)
    Preorder($T_{left}$)
    Preorder($T_{right}$)
```

```
Algorithm Postorder(T)
if T $\neq \varnothing$
    Postorder($T_{left}$)
    Postorder($T_{right}$)
    print(root of T)
```

## THANK YOU

**Shylaja S S**

Department of Computer Science & Engineering

**shylaja.sharath@pes.edu**