BIRZEIT UNIVERSITY

# Trees and Traversals


## Dr. Abdallah Karakra

Computer Science Department

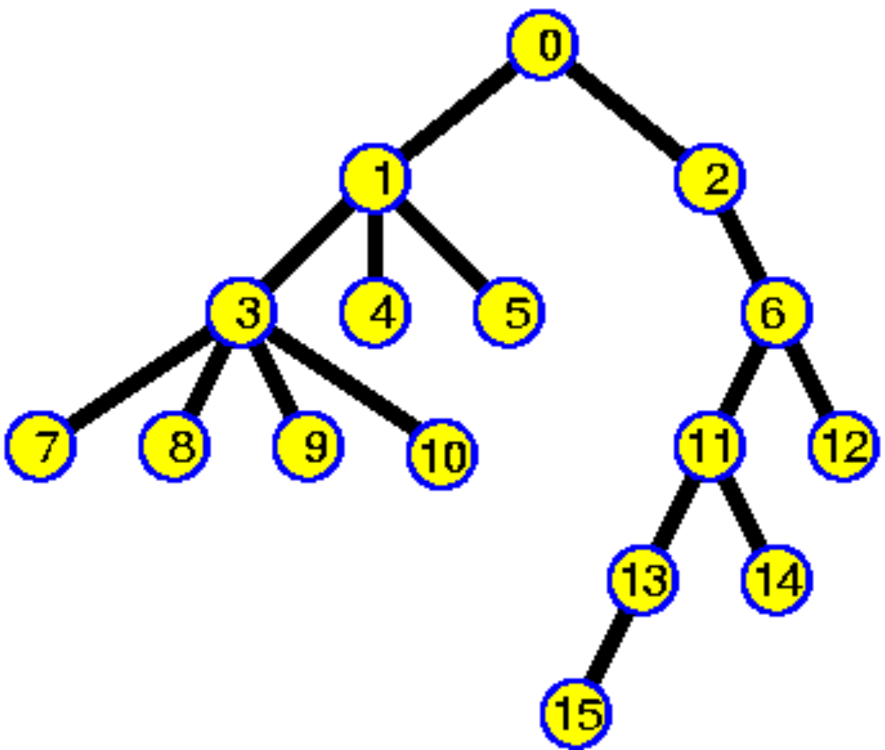
COMP242


Monday, April 8, 2024

# Tree

# Motivation

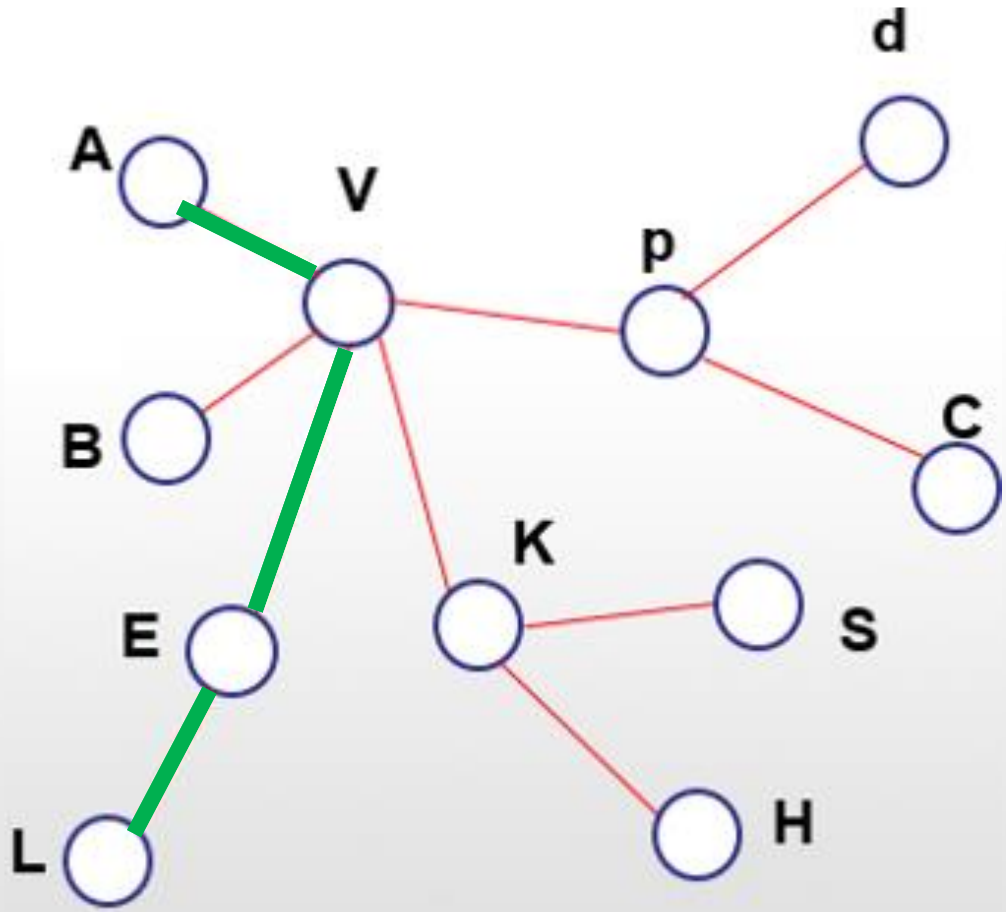
## Tree Definitions




Tree has 16 nodes
Tree has degree 4
Tree has depth 5
Node 0 is the root
Node 1 is internal
Node 4 is a leaf
4 is a child of 1
1 is the parent of 4
0 is grandparent of 4
3, 4 and 5 are siblings

# Tree

- **<u>Tree:</u> Set of nodes and edges that connect them.**
- **Exactly one path between any two nodes.**
- **<u>Path</u>: connected sequence of edges.**

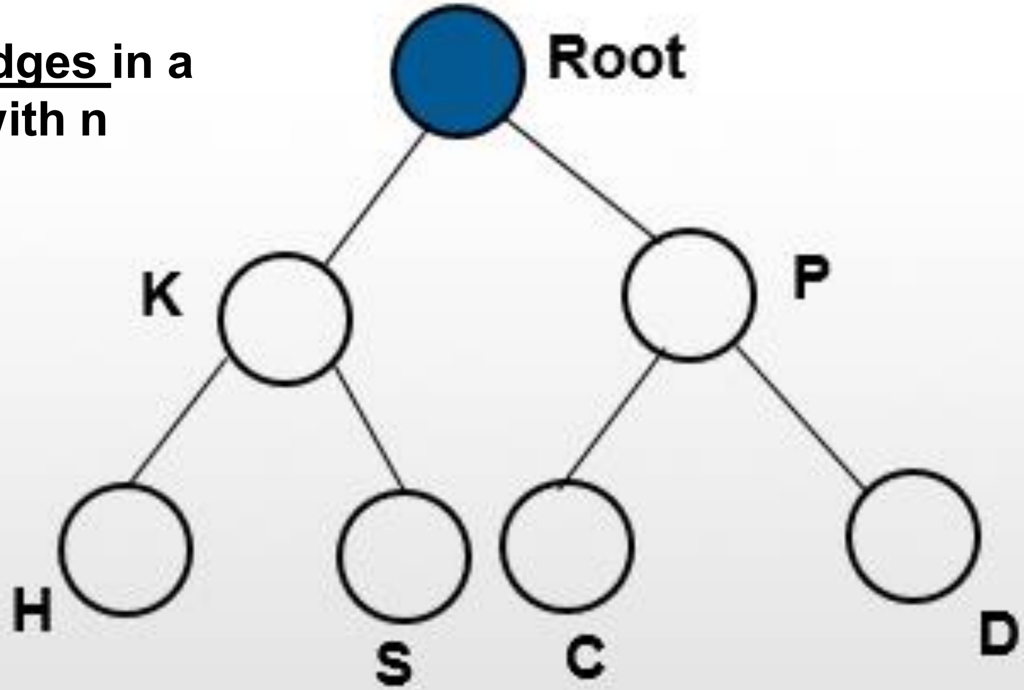



A **path** is a sequence of nodes such that the next node in the sequence is a child of the previous {A,V,E,L}

# Rooted Tree

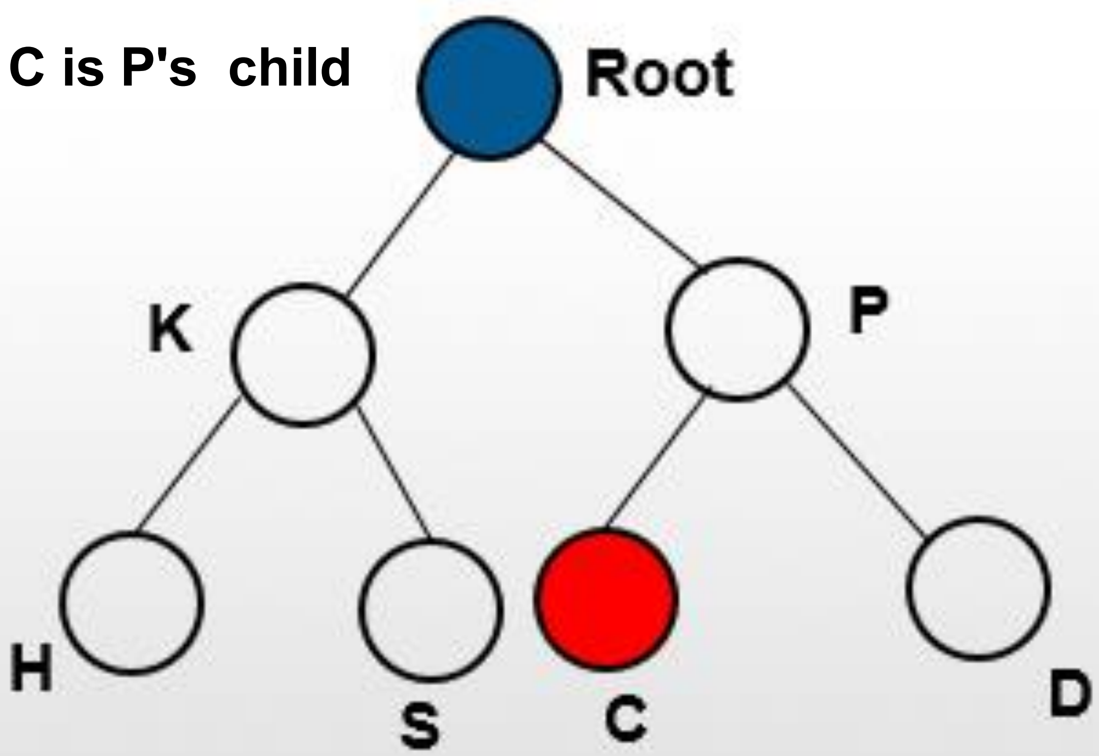
- **<u>Rooted tree</u>: One distinguished node is called the root.**

- **<u>Number of edges</u> in a rooted tree with n nodes = n-1**



# Rooted Tree

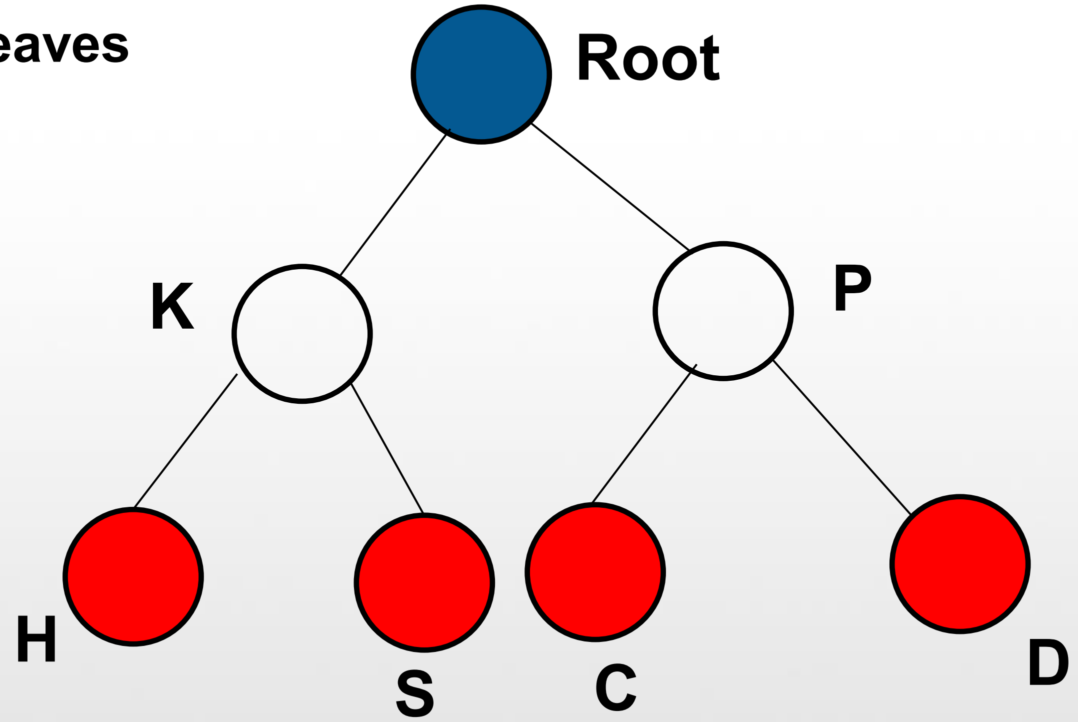
- Every node C, except root, has one parent P, the first node on path form c to the root.
- if p is parent of C,then C is P's child
- Root has no parents



# Rooted Tree

**<u>Leaf</u> : Node with no children.**

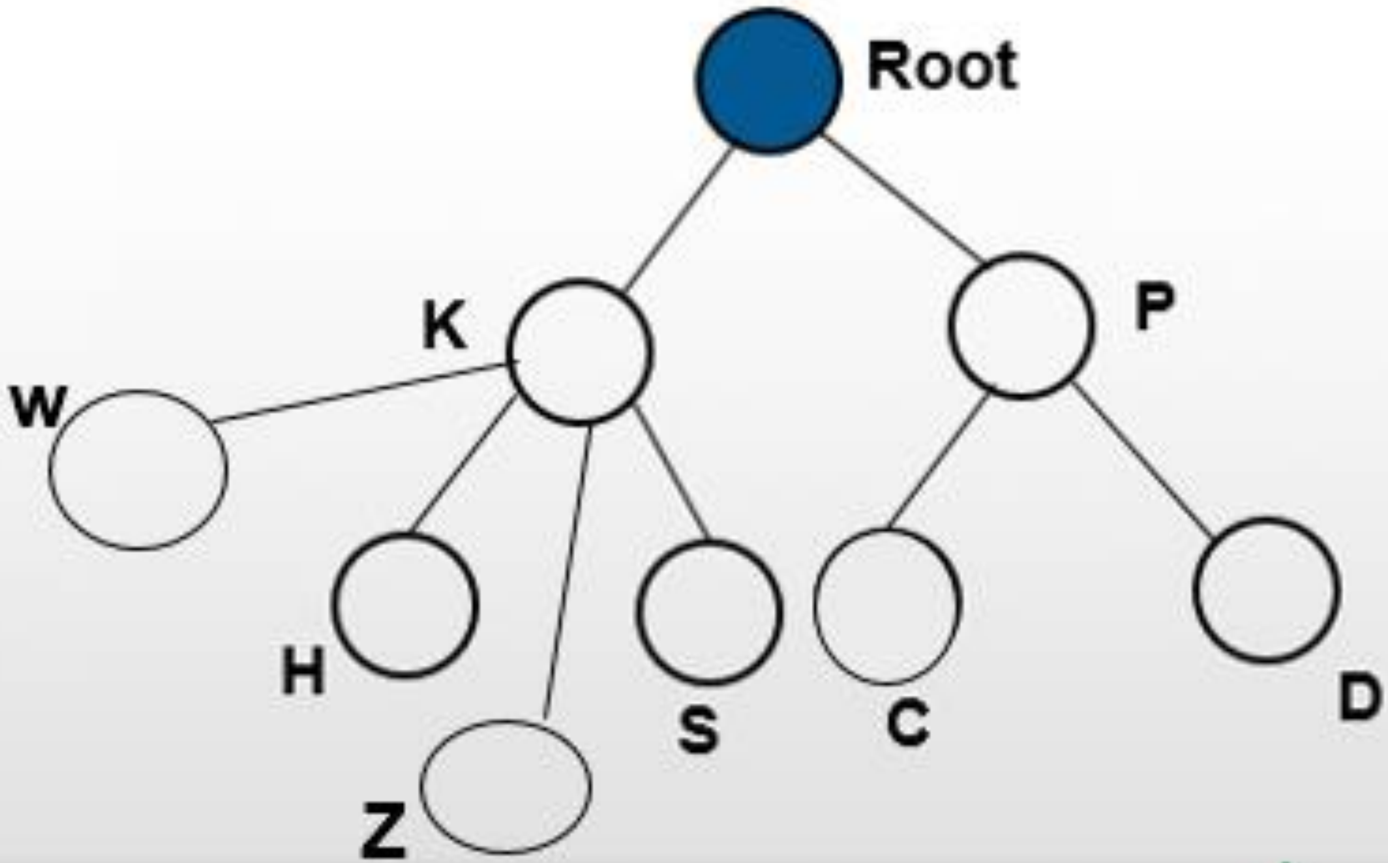
**Example. H,S,C, and D are leaves in our tree**




**Leaf node also called external node, all other nodes are internal**

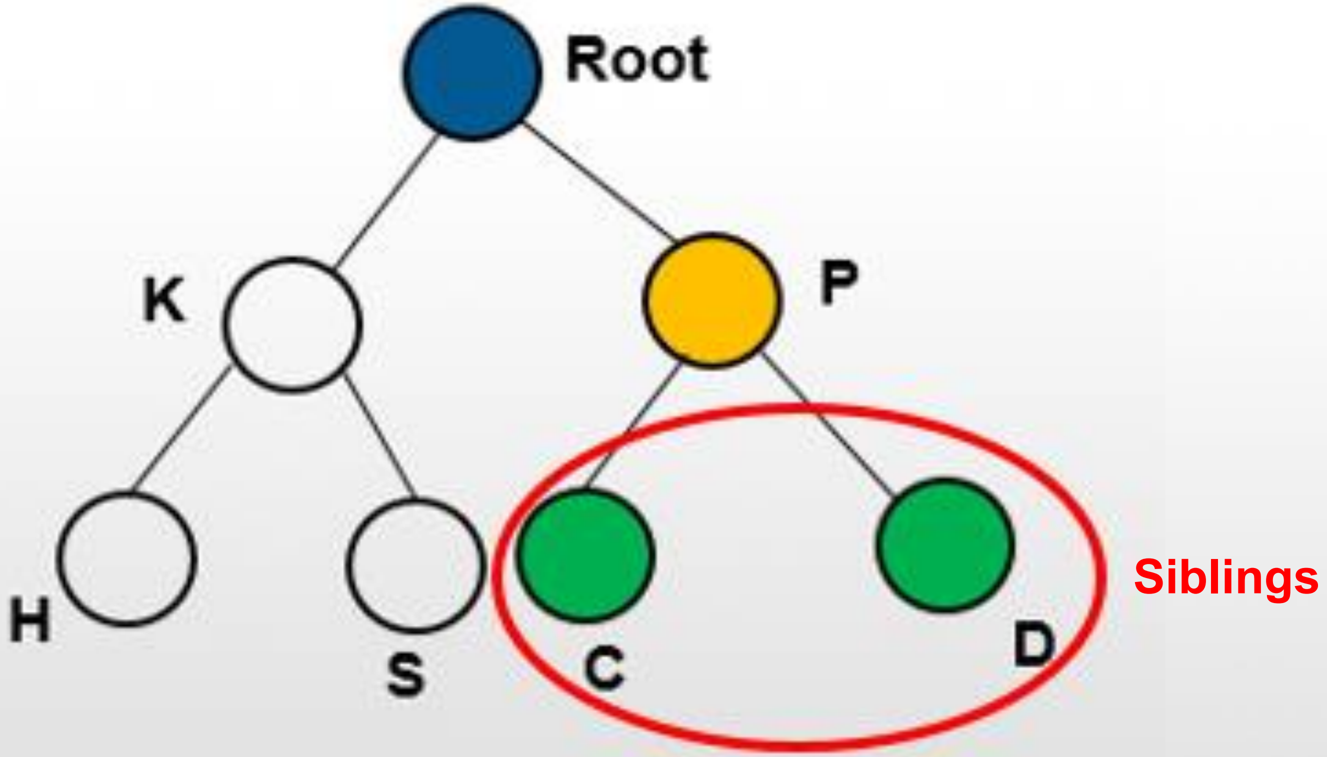
# Rooted Tree

- A node can have any number of children



# Rooted Tree

- **<u>Siblings</u> : Nodes with same parent.**

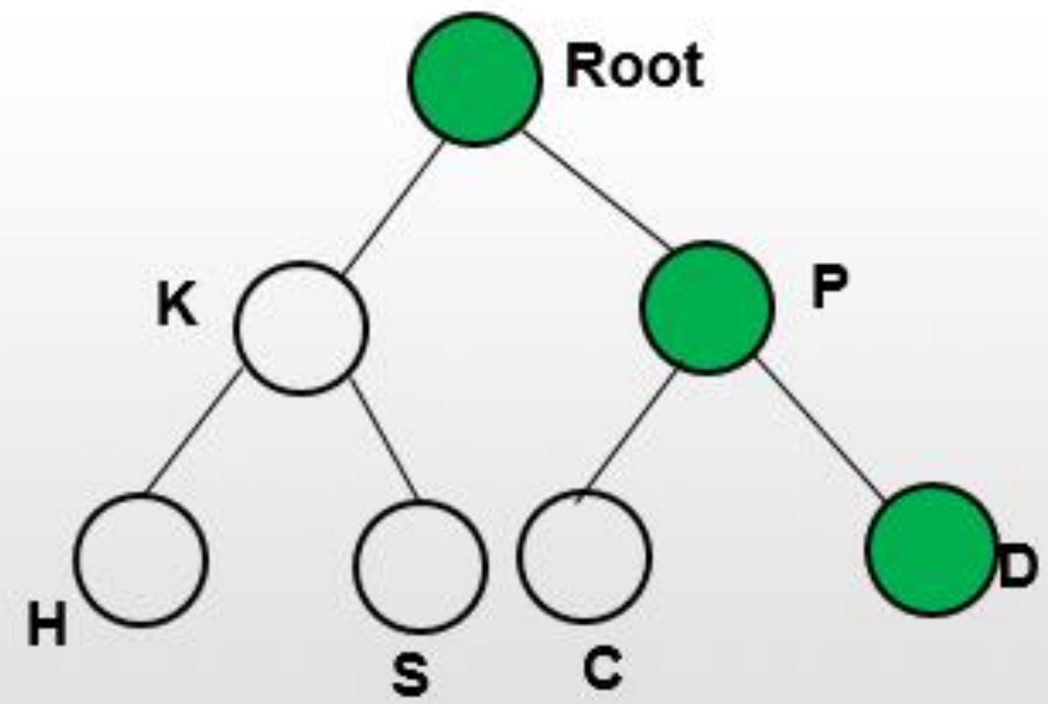



**P is parent of C, and P is parent of D**

# Rooted Tree

- ❑ **Ancestors of a node D : nodes on path from D to root, including D, D's parent, D's grandparent,...root (included).**

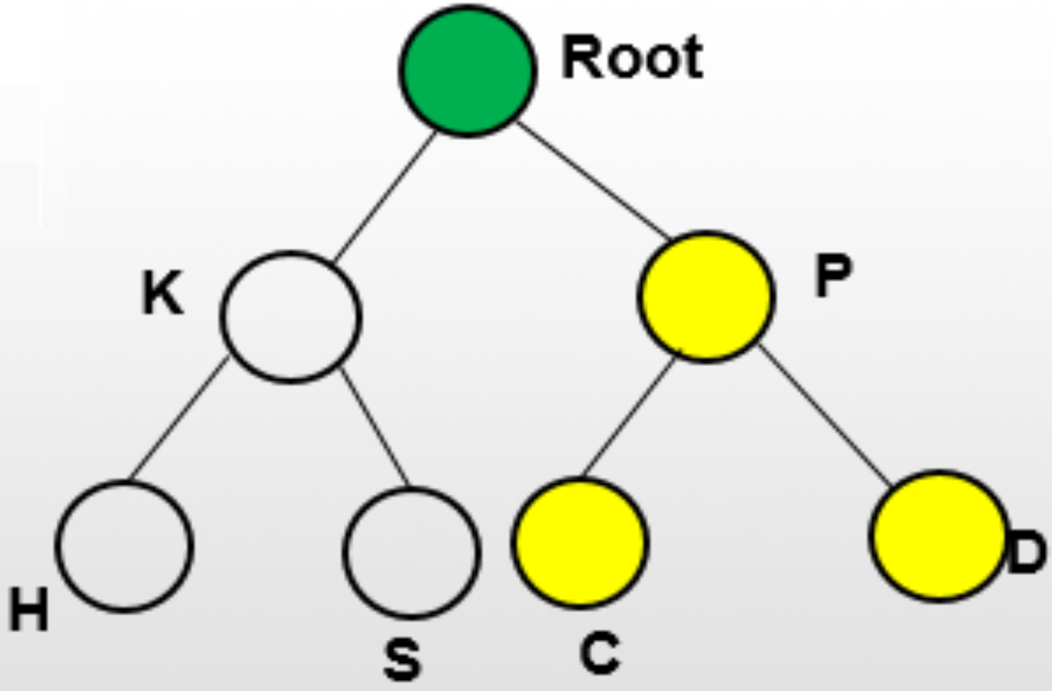
- ❑ **If P is <u>ancestor</u> of D, then D is <u>descendant</u> of P**



# Rooted Tree

***Example:*** **Descendants of P are** ***P,C, and D***

***Ancestors of H are H,K, and Root***

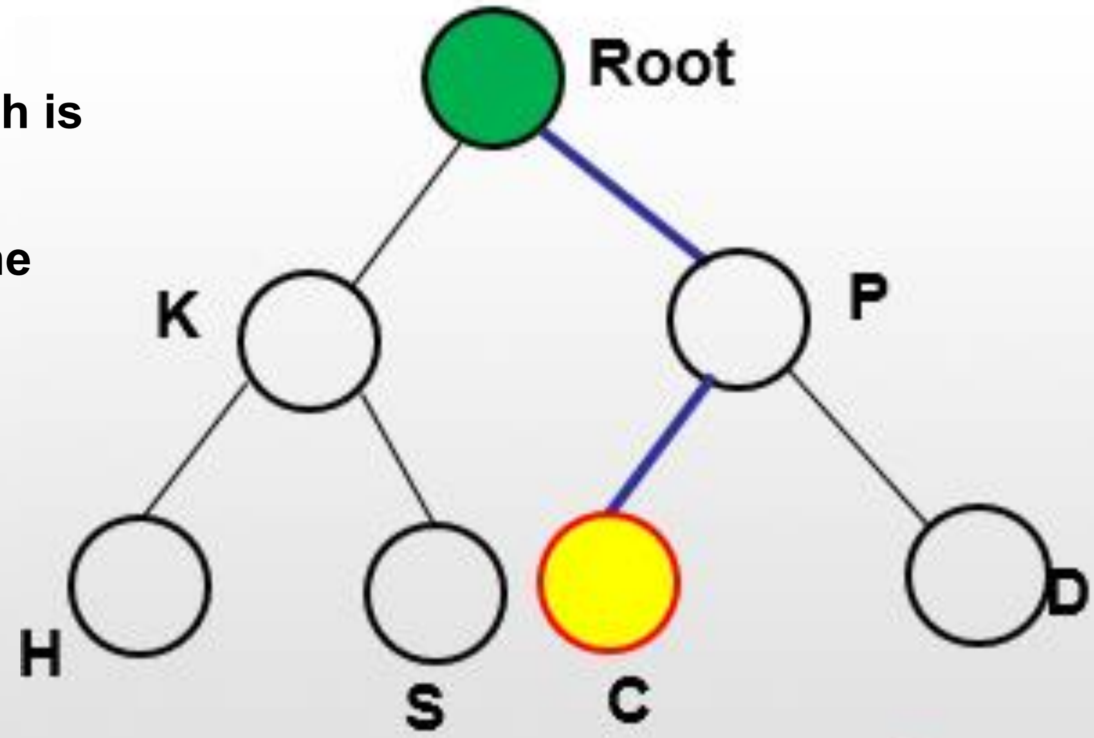


# Rooted Tree

- ❑ **Length of path: number of edges in path.**

**Example:**

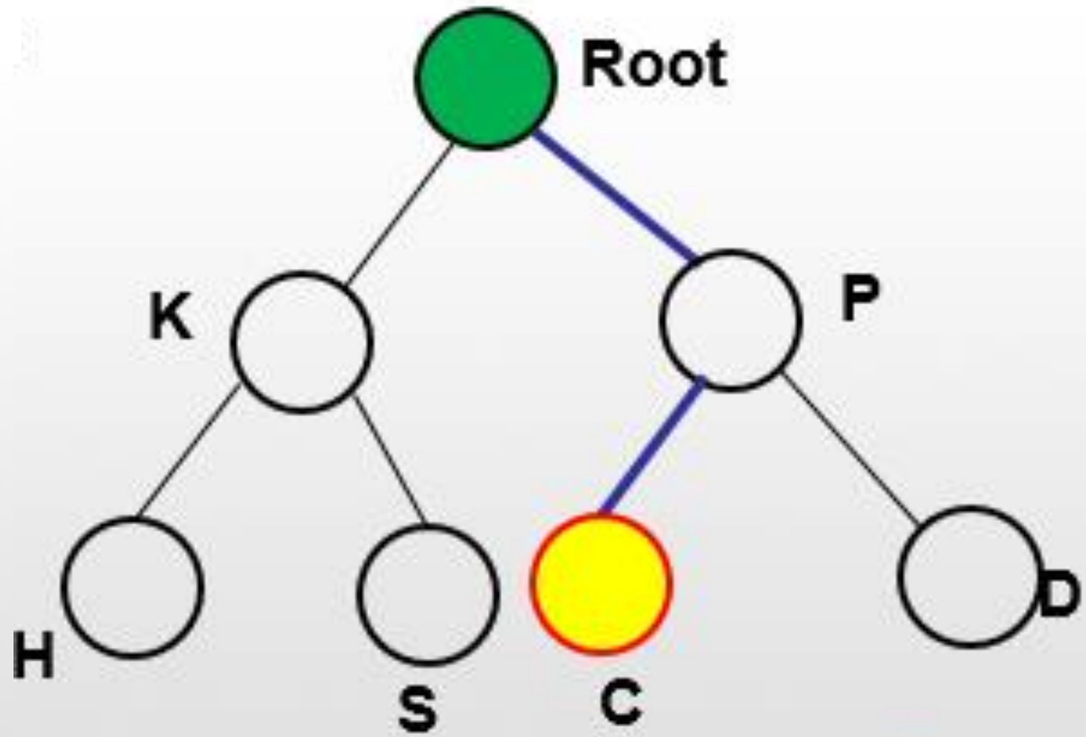
- ❑ **path from c to itself , the length is zero (empty path or no path)**
- ❑ **path from c to p, length is 1 (one edge in path)**
- ❑ **path from c to root, length is 2**



# Rooted Tree

- Depth of node n is length of path from n to root.

**EX: (Depth of root is zero)**
**EX: (Depth of C is 2)**

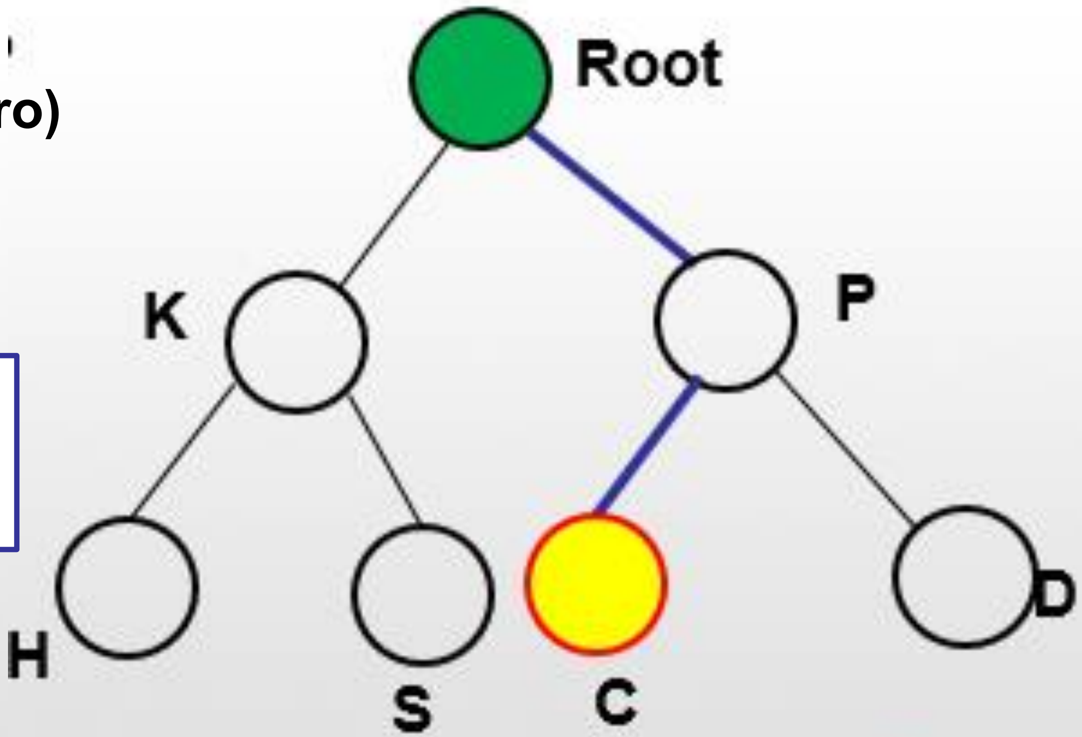


# Rooted Tree

- Height of node n is the length of path from n to its deepest descendant

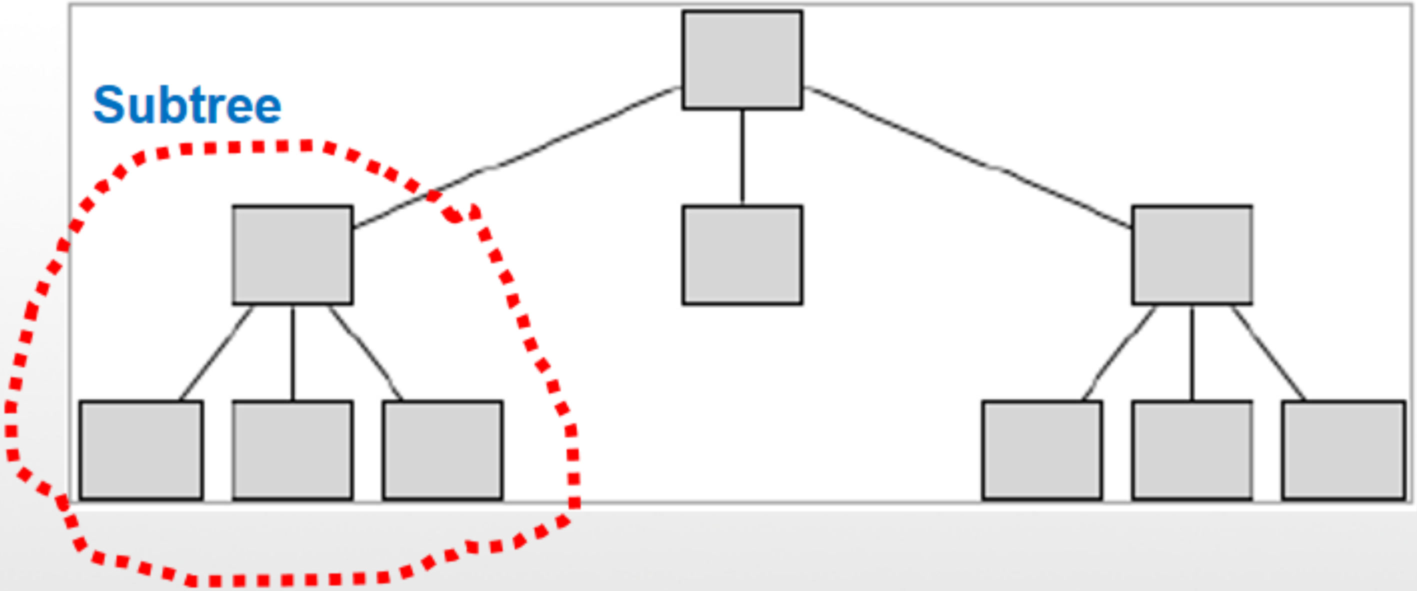
Examples:
(Height of any leaf node is zero)
(Height of Root node is 2)
(Height of P node is 1)

Height of a tree= height of the root
(The longest path length from
the root to a leaf)



# Rooted Tree

- **<u>Degree</u> : the maximum number of possible children for every node in the tree.**




**The height of tree is 2 and the degree 3**

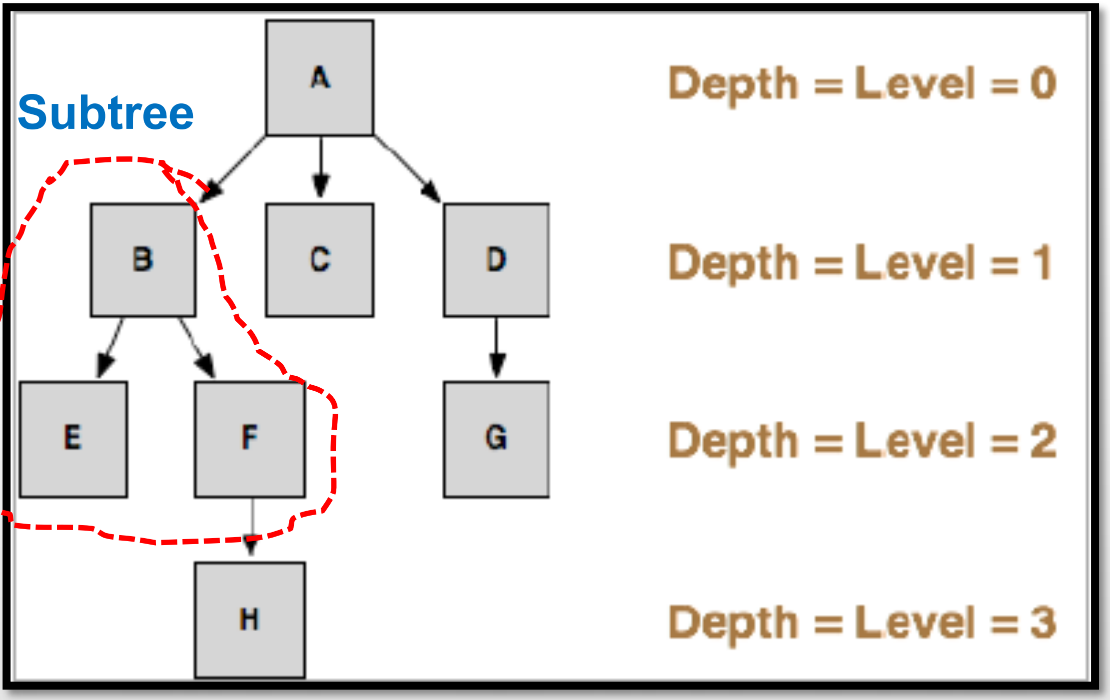
# Rooted Tree

**Node level & node depth : is the path length from the root**

- **The root is level 0 and depth 0**
- **Other nodes depth is 1 + depth of parent**

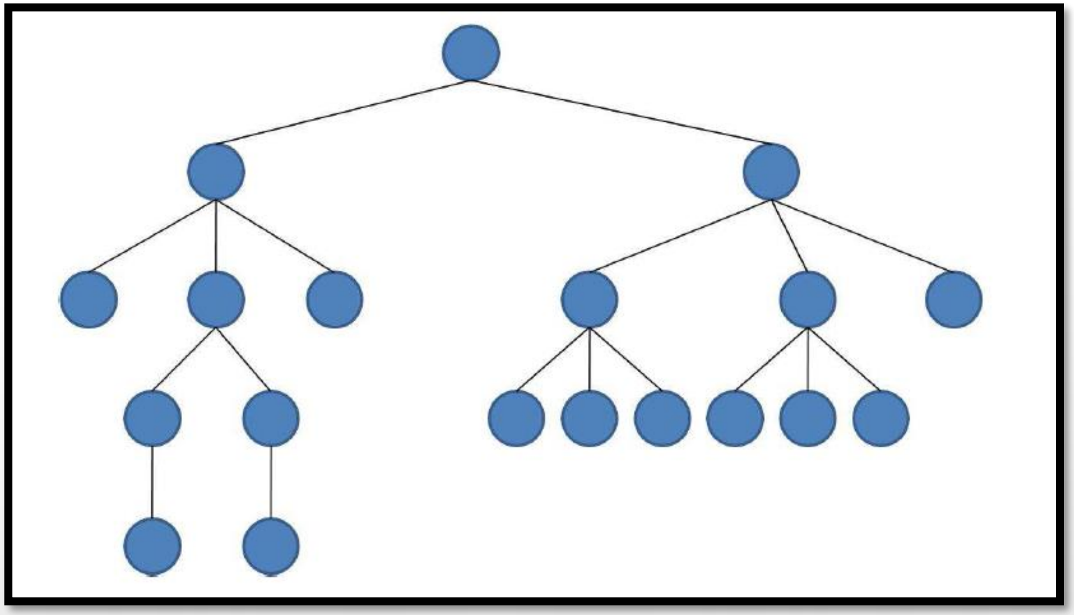
depth(A)=0
depth(B)=1
depth(E)=2
depth(H)=3

height(A)=3
height(F)=1
height(C)=0
height(E)=0
height(D)=1
height(B)=2



# Rooted Tree: H.W

**You have one week to do the following**

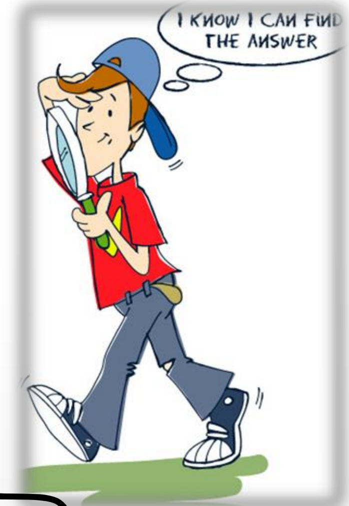



- ❑ **Explain the values of the main characteristics of the tree shown in the figure.**

**NOTE: These characteristics are grade(degree) of the tree, height, number of nodes, external and internal nodes.**

# Rooted Tree: H.W

**You have one week to do the following**



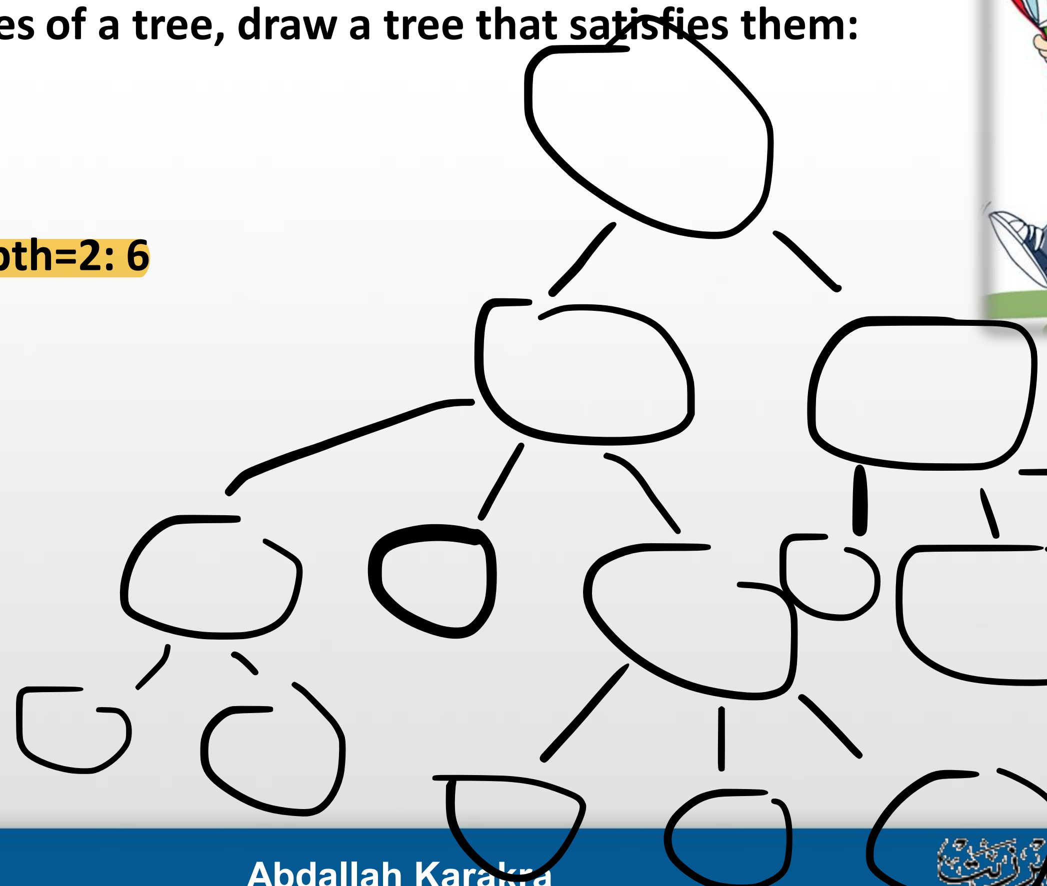

**Given the following properties of a tree, draw a tree that satisfies them:**

**1. Degree of the tree: 3**

**2. Number of nodes: 14**

**3. Height of the tree: 3**

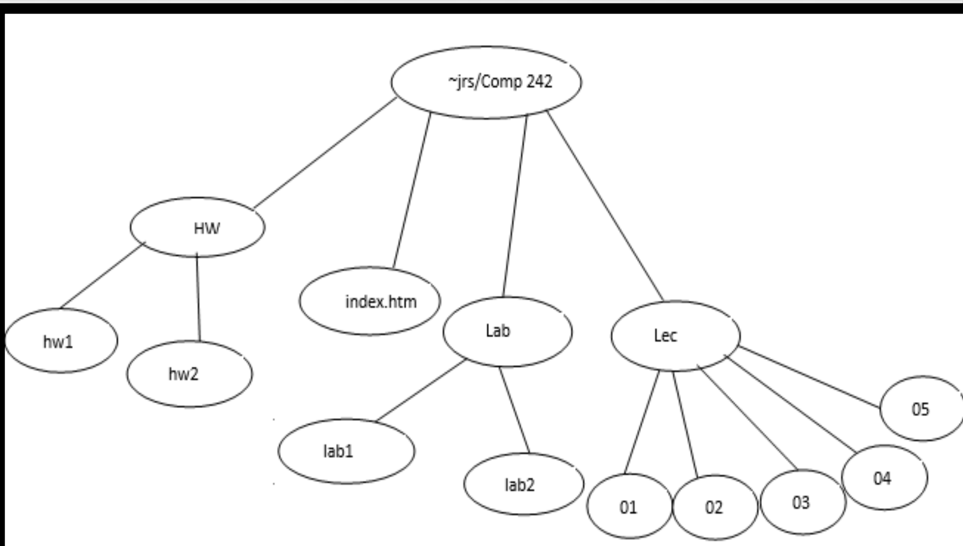
**4. Number of nodes with depth=2: 6**

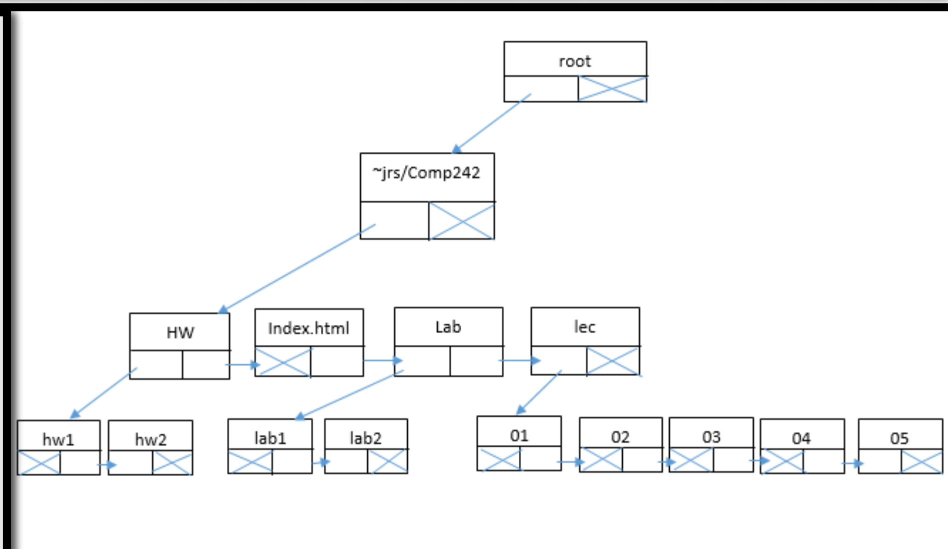


# Implementation: Rooted Tree

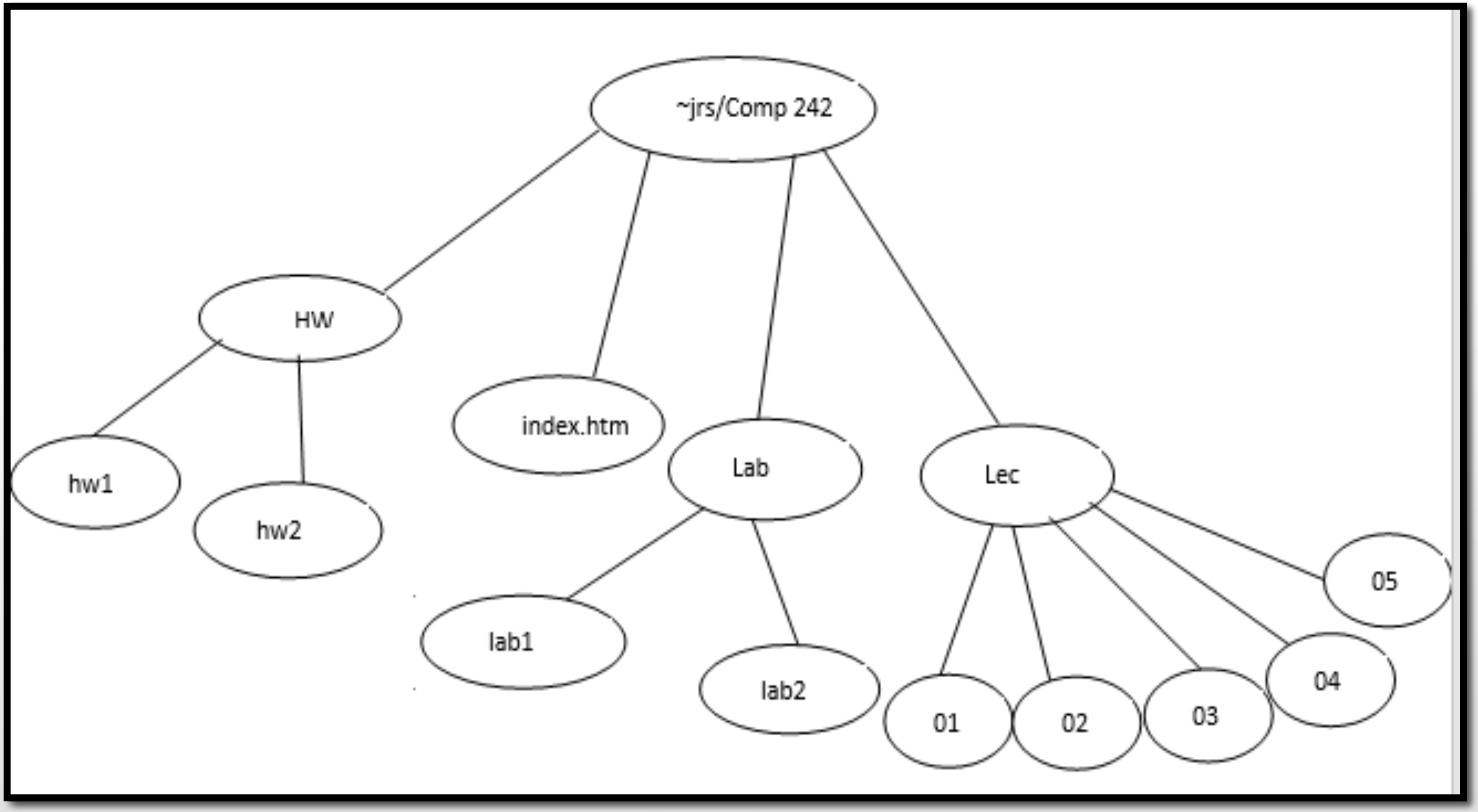
```java
//Class Node for the rooted Tree
public class RootedTreeNode {
   Object element;          //store data
   RootedTreeNode firstChild;    // reference for left tree
   RootedTreeNode nextSibling; //reference for right tree

}
```

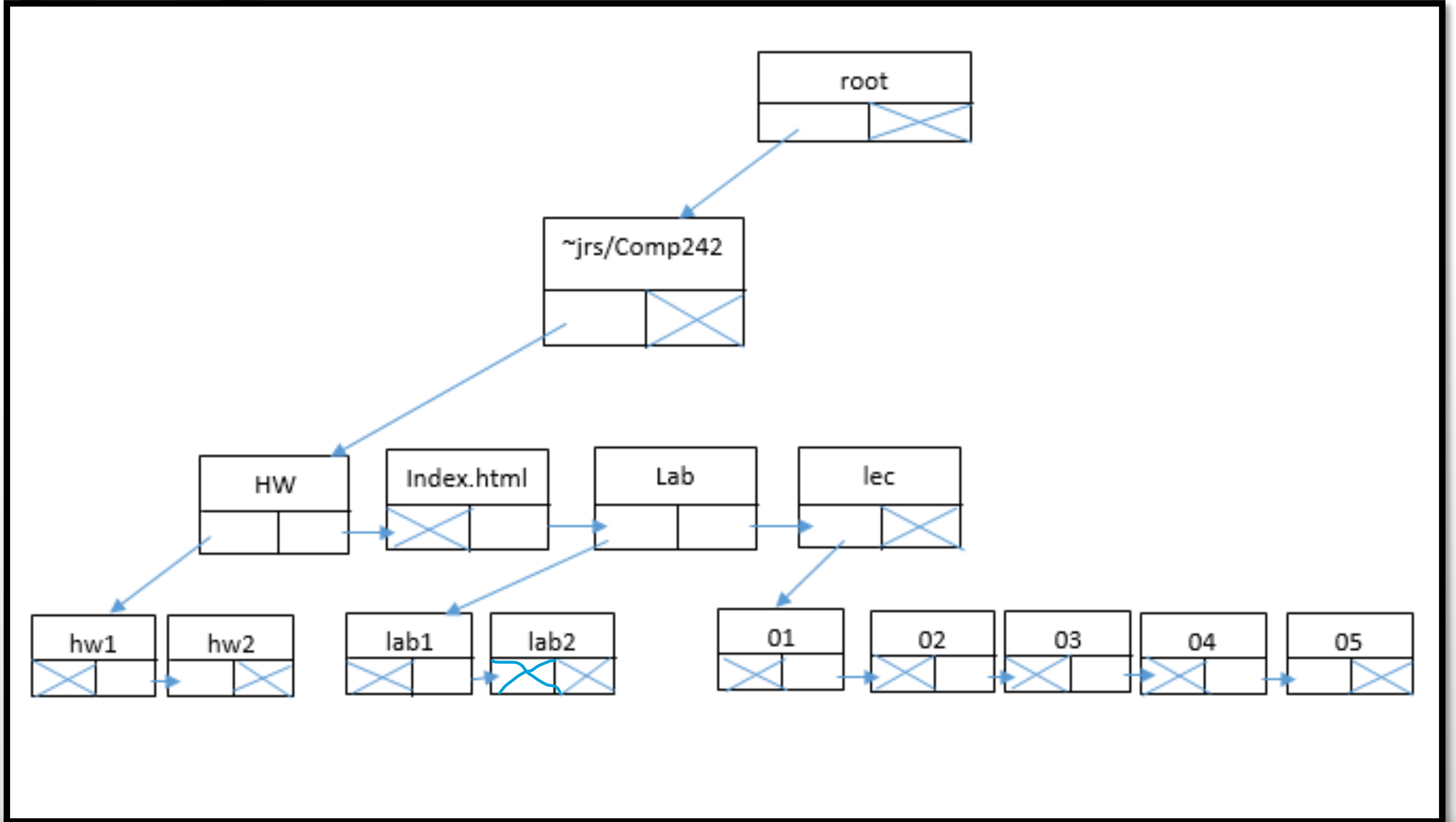


# Implementation: Rooted Tree



# Implementation: Rooted Tree



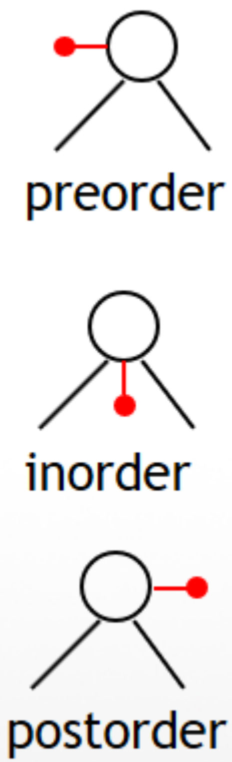
# Tree Traversals

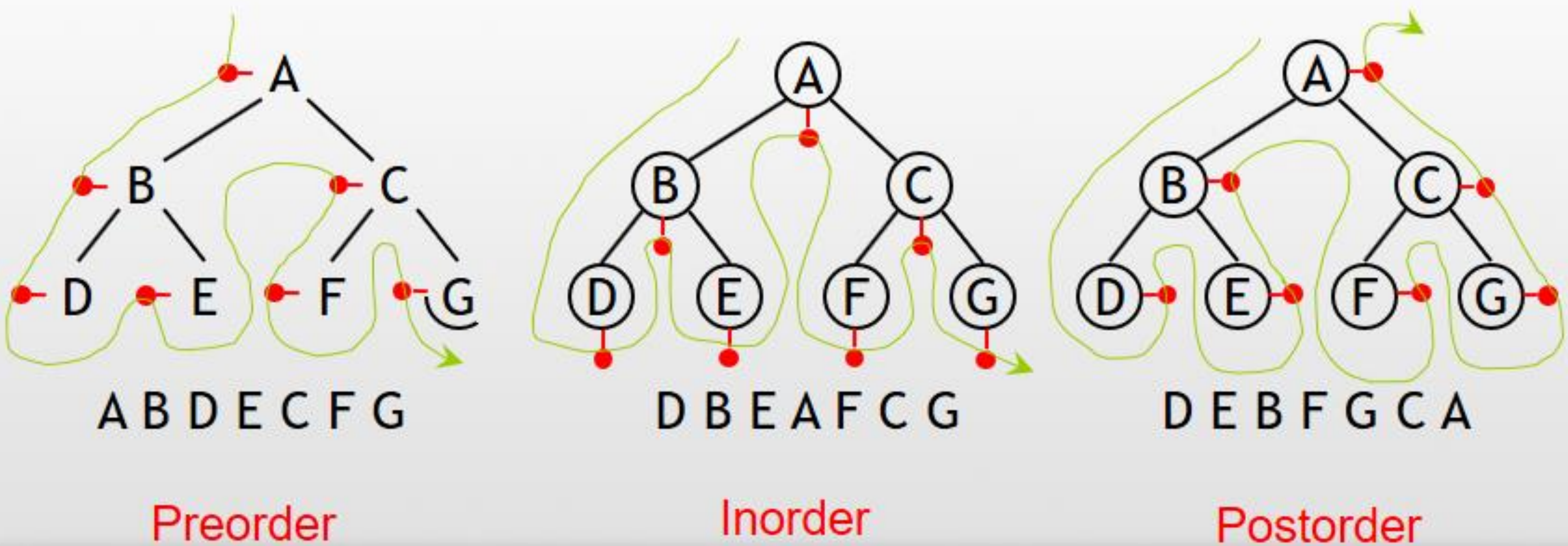
- A traversal is a manner of visiting each node in a tree once.

- What you do when visiting any particular node depends on the application; for instance, you might print a node's value, or perform some calculation upon it. There are several different traversals, each of which orders the nodes differently

# Tree Traversal

- Preorder: visits nodes as root → left → right
- Inorder: visits nodes as left → root → right
- Postorder: visits nodes as left → right → root



# Tree Traversal: More Details

**Preorder traversal**

Let $T$ be a tree with root $r$ and subtrees $T_1, T_2, \ldots, T_n$. In Preorder traversal, we visit the root $r$ first, then traverse the subtree $T_1$ in preorder, then traverse the subtree $T_2$ in preorder, and so on up to the traversal of the subtree $T_n$ in preorder.

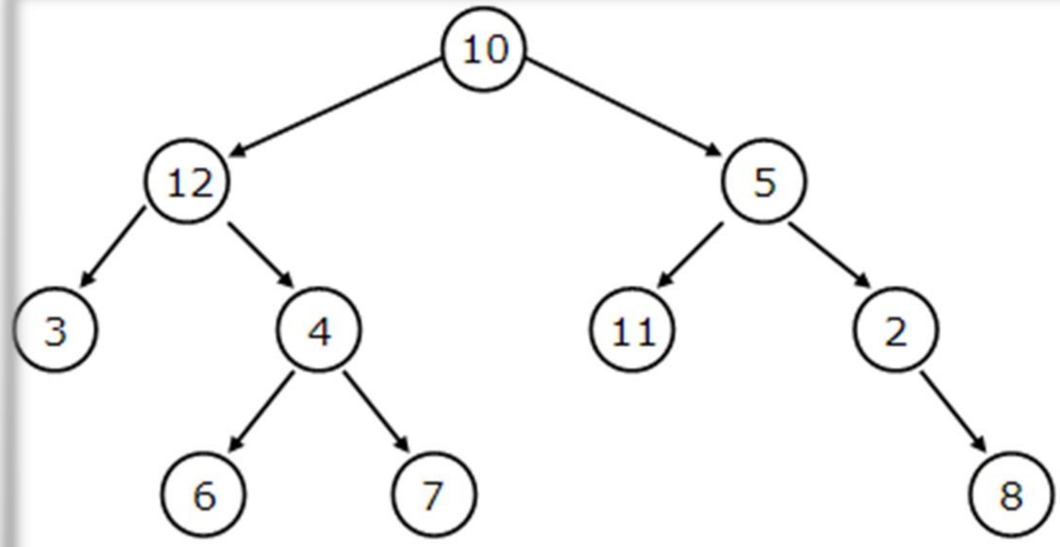
**Inorder traversal**

Let $T$ be a tree with root $r$ and subtrees $T_1, T_2, \ldots, T_n$. In an Inorder traversal, we traverse the subtree $T_1$ in inorder, then we visit the root $r$, then traverse the subtree $T_2$ in inorder, and so on up to the traversal of the subtree $T_n$ in inorder.

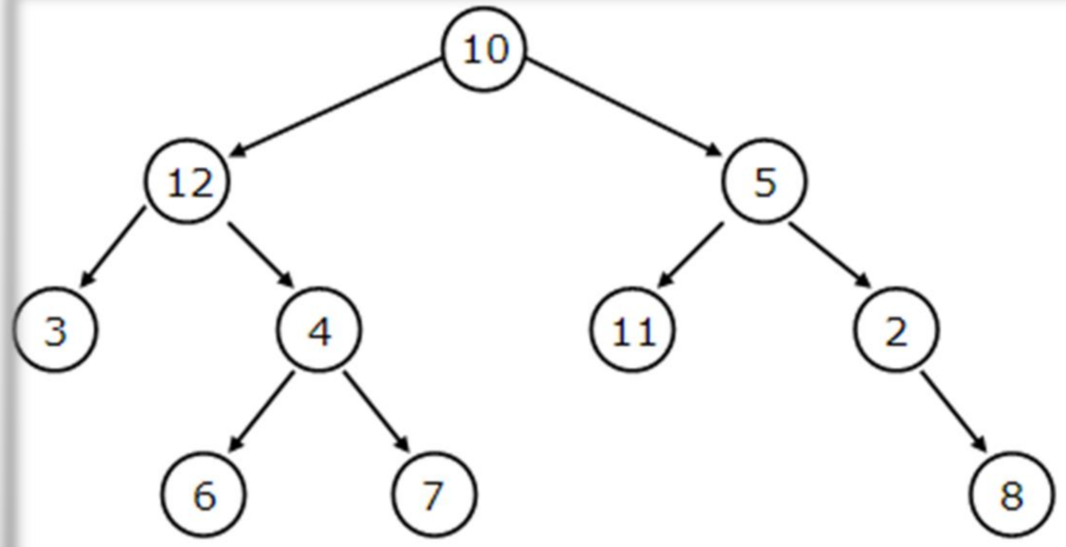
**Postorder traversal**

Let $T$ be a tree with root $r$ and subtrees $T_1, T_2, \ldots, T_n$. In a Postorder traversal, we traverse the subtree $T_1$ in postorder, then traverse the subtree $T_2$ in postorder, and so on up to the traversal of the subtree $T_n$ in postorder, and finally we visit the root $r$.

# Tree Traversal



# Tree Traversal



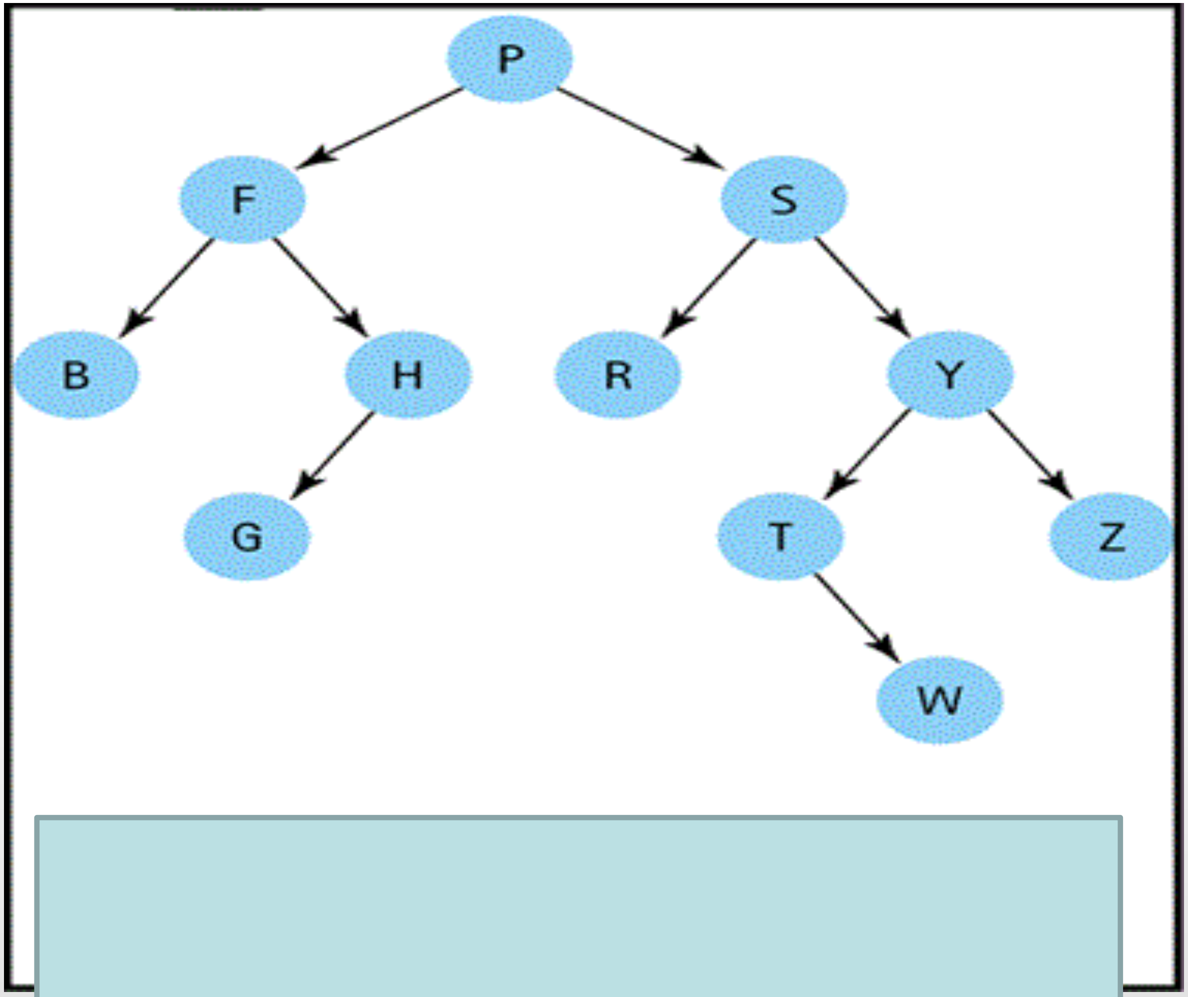

Levelorder tree traversal
10, 12, 5, 3, 4, 11, 2, 6, 7, 8

Inorder tree traversal
3, 12, 6, 4, 7, 10, 11, 5, 2, 8

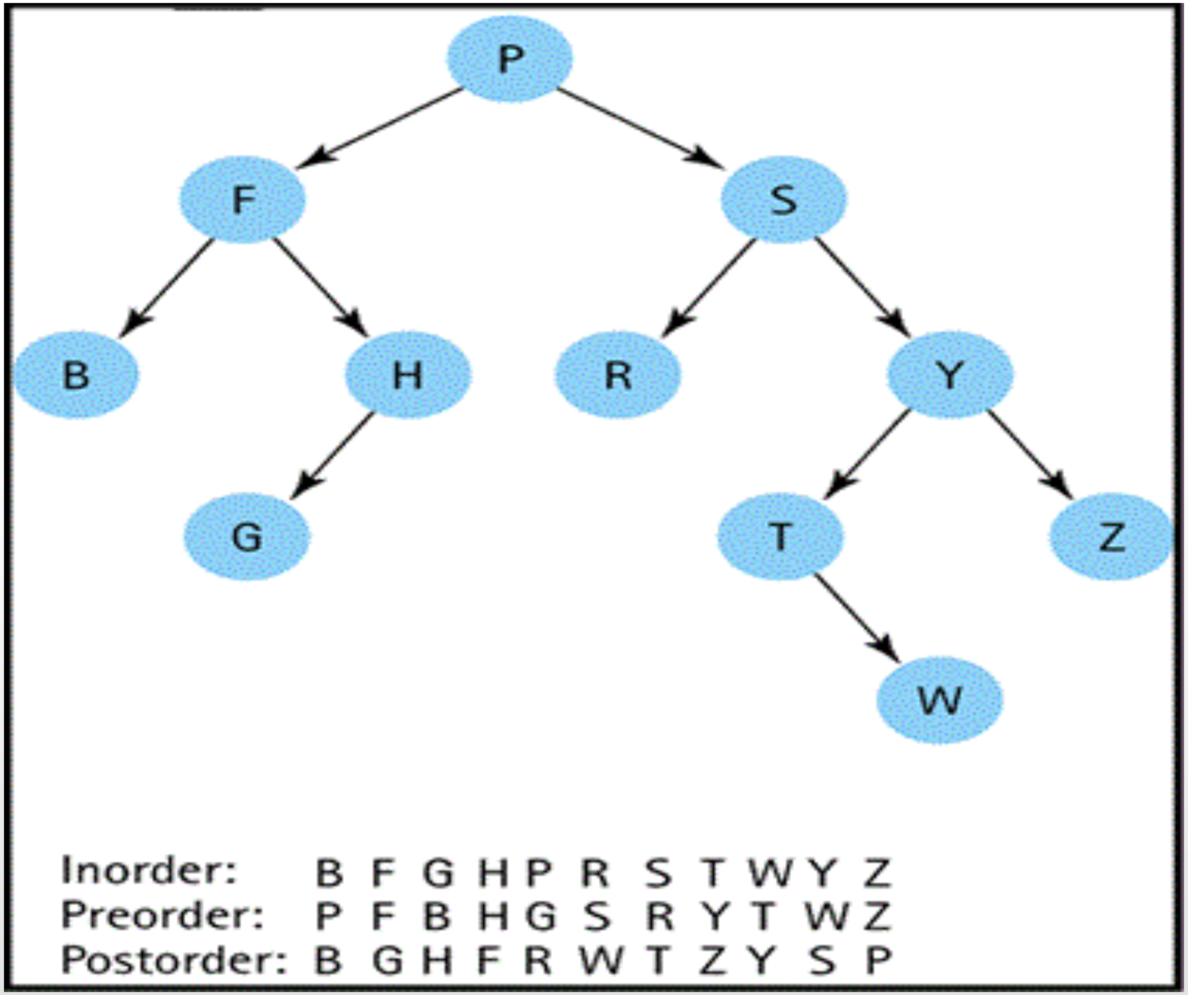
Preorder tree traversal
10, 12, 3, 4, 6, 7, 5, 11, 2, 8

Postorder tree traversal
3, 6, 7, 4, 12, 11, 8, 2, 5, 10

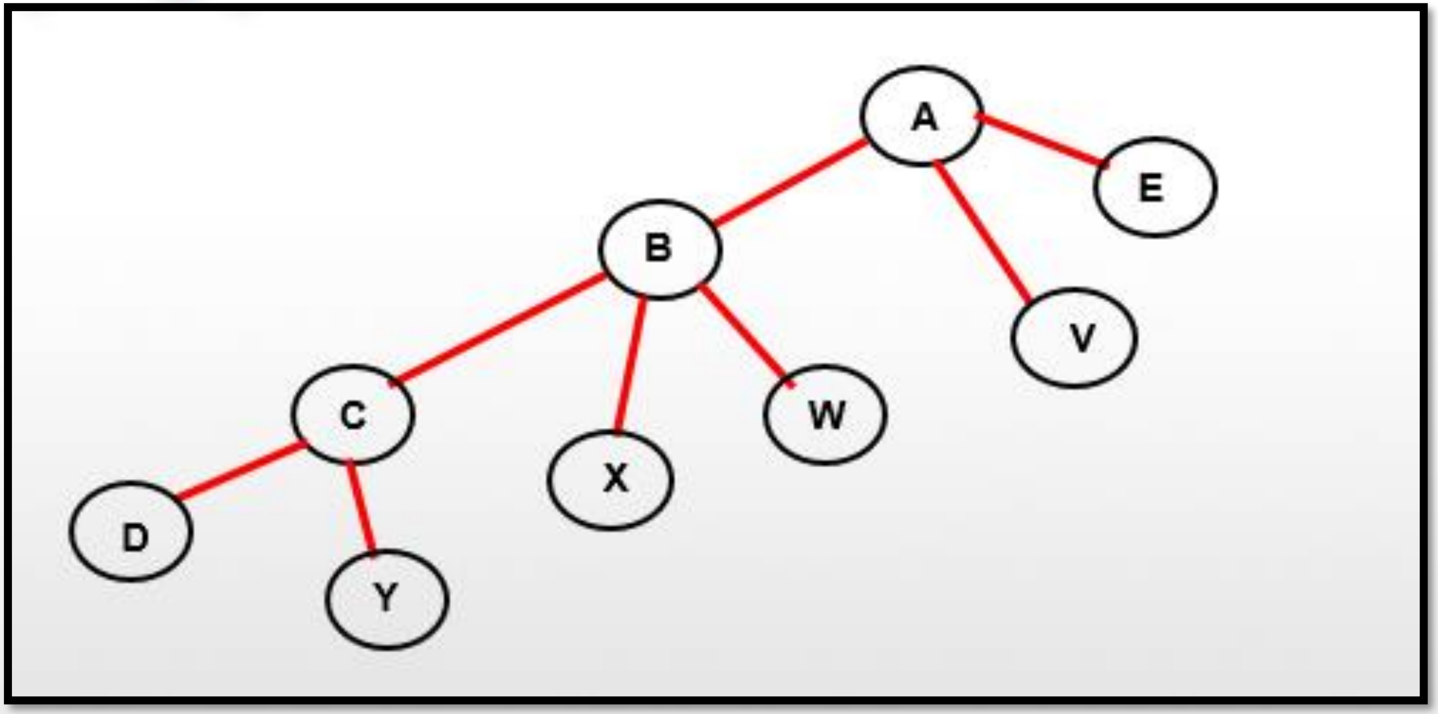
# Tree Traversal



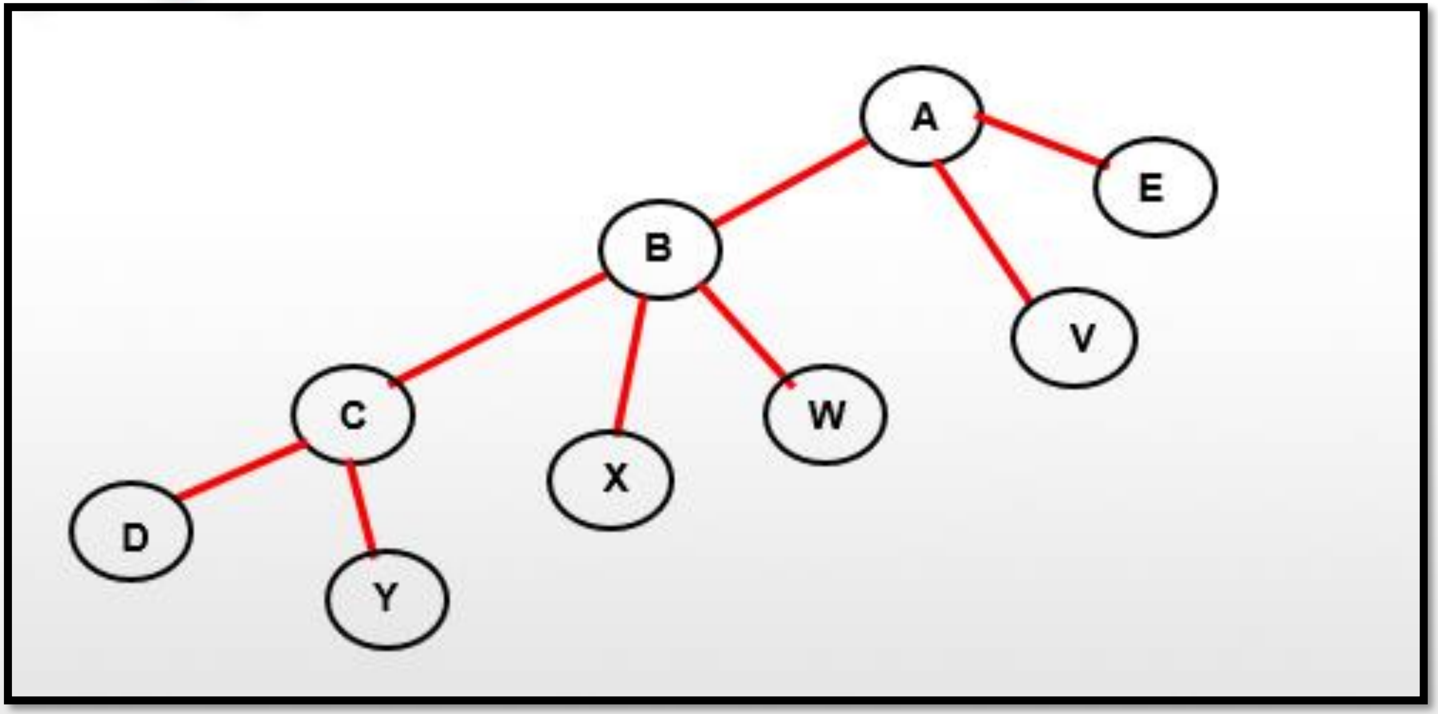
# Tree Traversal



# Tree Traversal



# Tree Traversal

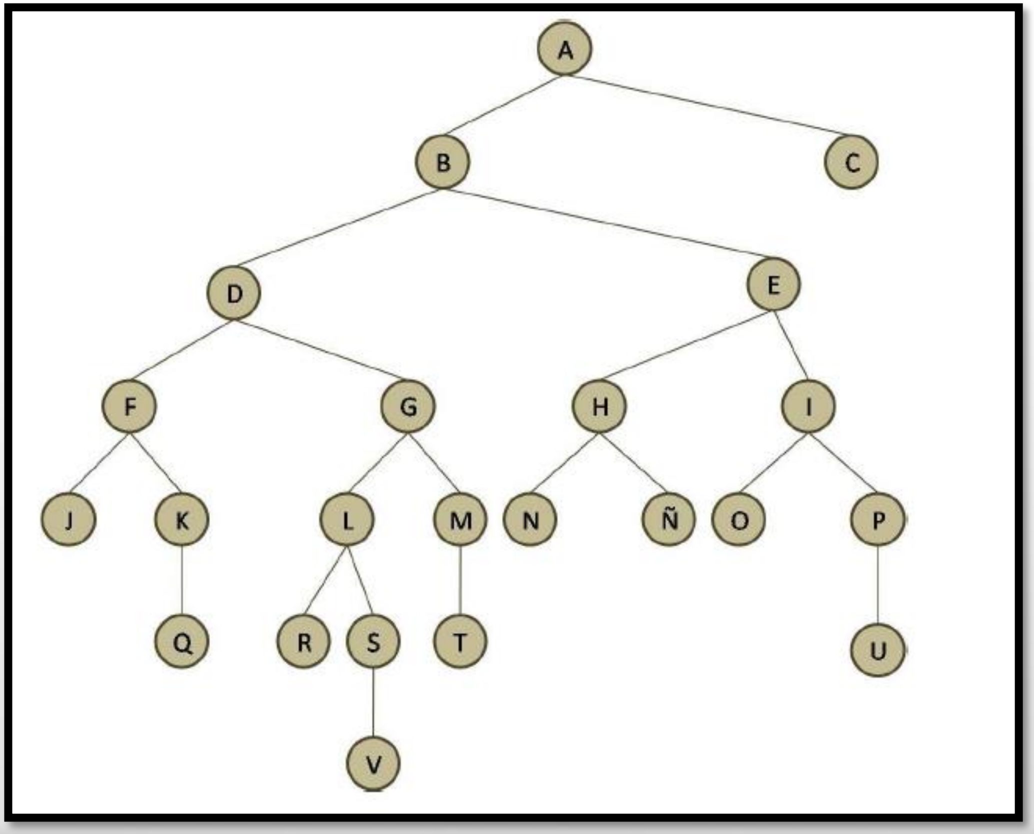



**Pre-order: A,B,C,D,Y,X,W,V,E**
**In-order: D,C,Y,B,X,W,A,V,E**
**Post-order: D,Y,C,X,W,B,V,E,A**

# Rooted Tree: H.W

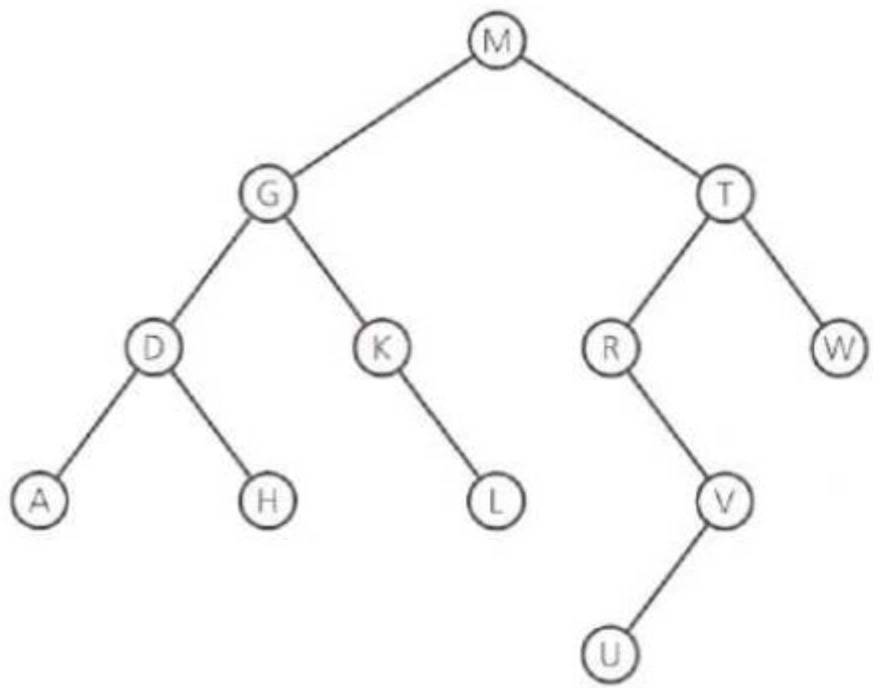
**You have one week to do the following**

**Given the following tree, write the pre-order, in-order and post-order traversals.**




# Extra Exercises

Carrano, 4th edition, Chapter 10, Exercise 2: What are the preorder, inorder, and postorder traversals of the following binary trees:



# Question?


**“Success is the sum of small efforts, repeated day in and day out.”**
Robert Collier

## References:


1. Algorithms and Data Structures, Julian Moreno Schneider et al.
2. Fundamentals of Data Structures in C, Ellis Horowitz et al.
3. Data Structures and Problem Solving with C++: Walls and Mirrors
4. Analysis of algorithms robert Sedgewick
5. Prof. Sin-Min Lee Lecture Notes
6. Prof. Evan Korth Lecture Notes