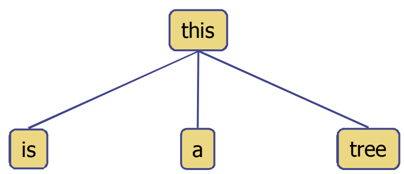
CSE 2215:
Data Structures and Algorithms-I

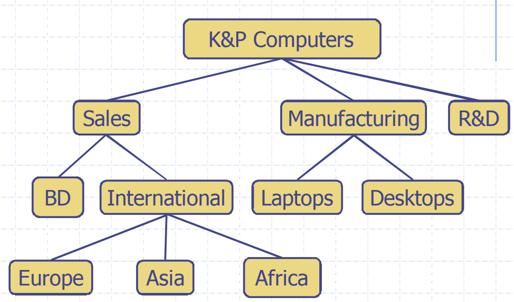
Trees and Tree Traversals



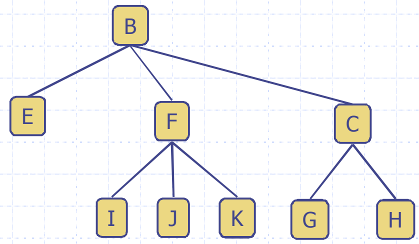
Dr. Md. Abul Kashem Mia, Professor, CSE Dept and Pro-Vice Chancellor, UIU

What is a Tree?

- A connected acyclic graph is a tree
- In computer science, a tree is an abstract model of a hierarchical structure
- A tree consists of nodes with a parent-child relationship
- Applications:
  - Organizational charts
  - File systems
  - Programming environments



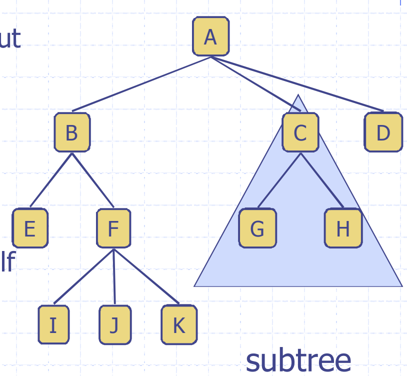
What is a Tree?



- A connected acyclic graph is a tree.
- A tree *T* is a set of nodes in a parent-child relationship with the following properties:
  - *T* has a special node *r*, called the root of *T*, with no parent node
  - Each node *v* of *T*, different from *r*, has a unique parent node *u*

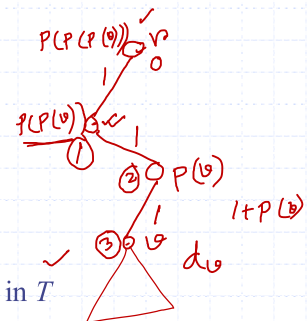
Tree Terminology

- Root: node without parent (A)
- Internal node: node with at least one child (A, B, C, F)
- External node (Leaf): node without children (E, I, J, K, G, H, D)
- Ancestors of a node: parent, grandparent, grand-grandparent, etc.
- Descendants of a node: child, grandchild, grand-grandchild, etc.
- Depth of a node: number of ancestors, excluding the node itself
- Height of a tree: maximum depth of any node (3)
- Siblings: two nodes that are children of the same parent
- Subtree: tree consisting of a node and its descendants

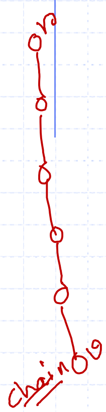


Depth and Height

- The depth of a node v can be recursively defined as follows
  - If v is the root, then the depth of v is 0.
  - Otherwise, the depth of v is one plus the depth of the parent of v



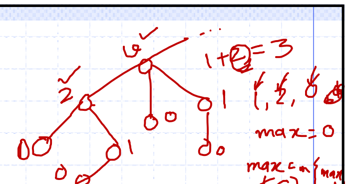
Algorithm depth(T, v)
  if T.isRoot(v) **then**
    return 0
  else
    return 1 + depth(T, T.parent(v))



- Running time: $O(1 + d_v)$, d_v is depth of v in T

In worst case $O(n)$, n is the number of nodes in T

Depth and Height



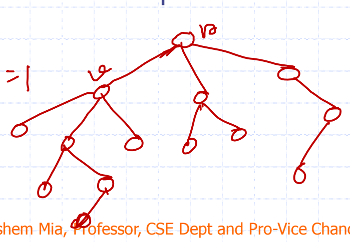
- The height of a node v can be recursively defined as follows
  - If v is a leaf node, then the height of v is 0.
  - Otherwise, the height of v is one plus the maximum height of a child of v

The height of a tree T is the height of the root of T

h(T)

The height of a tree T is equal to the maximum depth of a leaf node of T

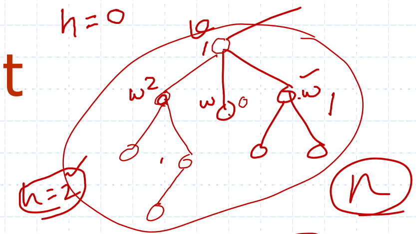
Ans: depth(v) = 1
depth(r) = 0
height(v) = 3
height(r) = 4



what is the depth of nodes v and r?

what is the height of v and r?

Depth and Height



Algorithm height(T, v)
  if T.isLeaf(v) **then**
    return 0
  else
    h = 0
    for each w ε T.children(v) **do**
      h = max(h, height(T, w))
    return 1 + h

1 + 2 = 3

O(n)

n-1

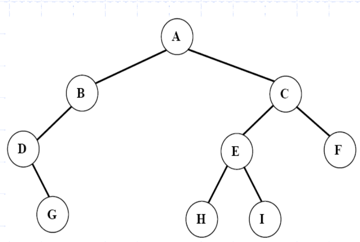
Running time: $O(\Sigma_{v \in T}(1 + c_v))$, where c_v is the number of children of v in T

Since $\Sigma_{v \in T} c_v = n - 1$, we have $O(\Sigma_{v \in T}(1 + c_v)) = O(n)$.

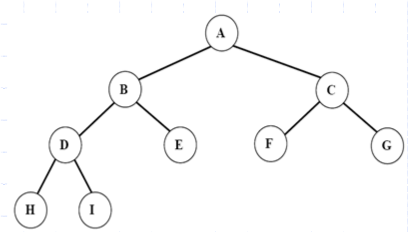
- The height of tree T is obtained by calling height(T, r).

Ordered Trees

- A tree is **ordered** if there is a linear ordering defined for each child of each node.
- A **binary tree** is an ordered tree in which every node has at most two children.
- If each node of a tree has either zero or two children, the tree is called a **proper binary tree**.



A binary tree



A proper binary tree

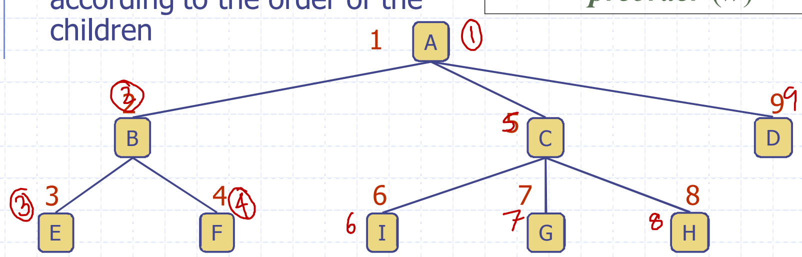
Traversal of Trees

- A traversal of a tree T is a systematic way of visiting all the nodes of T
- Traversing a tree involves visiting the root and traversing its subtrees
- There are the following traversal methods:
  - Preorder Traversal
  - Postorder Traversal
  - Inorder Traversal (of a binary tree)

Preorder Traversal (General Tree)

- In a preorder traversal, a node is visited before its descendants
- If a tree is ordered, then the subtrees are traversed according to the order of the children

Algorithm *preOrder*(*v*)
    visit(*v*)
    for each child *w* of *v*
        preorder (*w*)



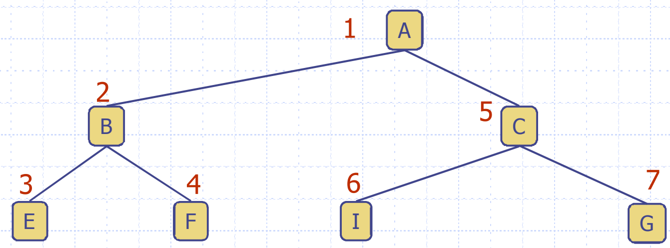
Preorder*: A B E F C I G H D*

Preorder Traversal (Binary Tree)

pre
root L R

Traversing a binary tree in *Preorder*

1. Visit the ***root***.
2. Traverse the ***left subtree*** in Preorder.
3. Traverse the ***right subtree*** in Preorder.

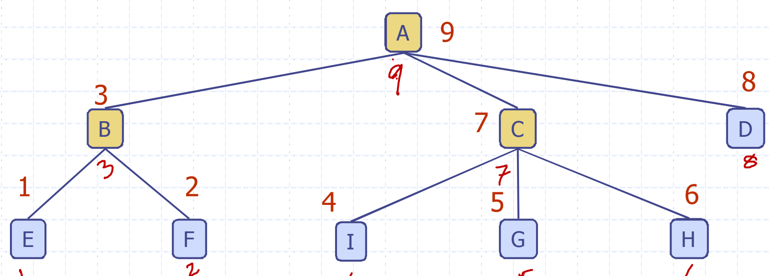


Preorder: *A B E F C I G*

Postorder Traversal (General Tree)

- In a postorder traversal, a node is visited after its descendants

Algorithm ***postOrder(v)***
    for each child ***w*** of ***v***
        postOrder (***w***)
    visit(***v***)



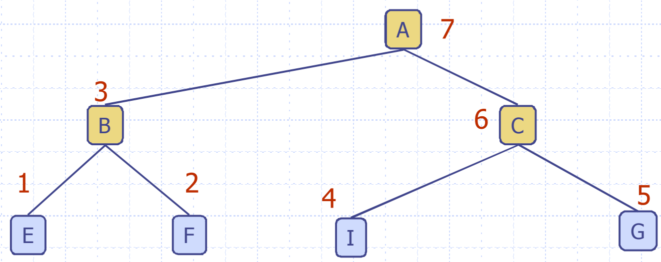
Postorder: *E F B I G H C D A*

Postorder Traversal (Binary Tree)

post
L R root

Traversing a binary tree in *Postorder*

1. Traverse the ***left subtree*** in Preorder.
2. Traverse the ***right subtree*** in Preorder.
3. Visit the ***root***.



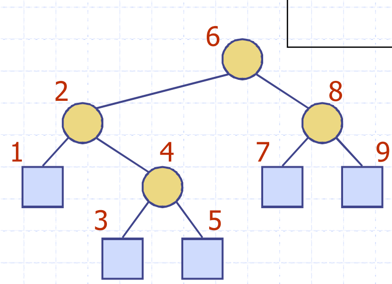
Postorder*: E F B I G C A*

Inorder Traversal

in
L root R

- In an inorder traversal a node is visited after its left subtree and before its right subtree

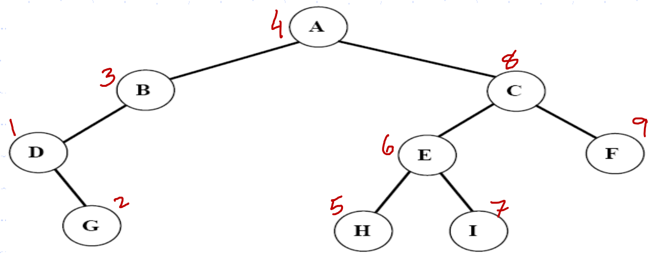
Algorithm *inOrder*(*v*)
  if *isInternal* (*v*)
    inOrder (*leftChild* (*v*))
  visit(*v*)
  if *isInternal* (*v*)
    inOrder (*rightChild* (*v*))



Inorder Traversal

Traversing a binary tree in *inorder*

1. Traverse the ***left subtree*** in inorder.
2. Visit the ***root***.
3. Traverse the ***right subtree*** in inorder.



Inorder*: D G B A H E I C F*

Dr. Md. Abul Kashem Mia, Professor, CSE Dept and Pro-Vice Chancellor, UIU 15