CMPT-335 Discrete Structures

TREES

Basic Tree Concepts

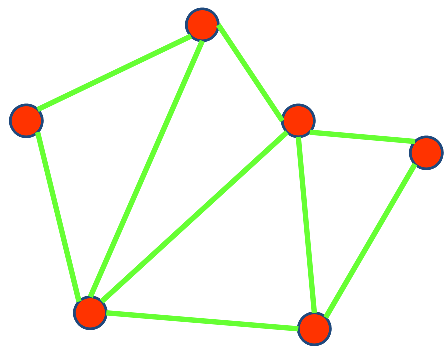
- A tree is a specific kind of a graph
- A **tree** is a discrete structure, which is a connected undirected graph with no simple circuits
- A tree consists of finite set of elements, called nodes (vertices), and a finite set of called branches (edges), that connect the nodes
- The number of branches associated with a node is the degree of the node

Basic Tree Concepts

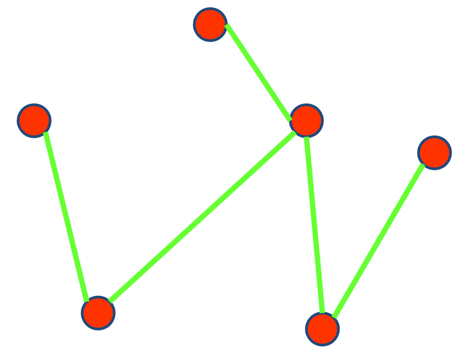
- Since a tree cannot have a simple circuit, a tree cannot contain multiple edges or loops.
- Therefore, any tree must be a simple graph
- Theorem. An undirected graph is a tree if and only if there is a unique simple path between any of its vertices

Basic Tree Concepts

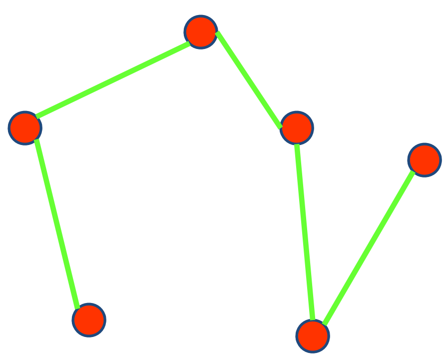
- **Example:** Are the following graphs trees?



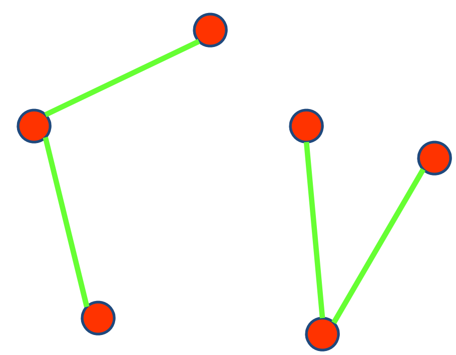
No.



Yes.



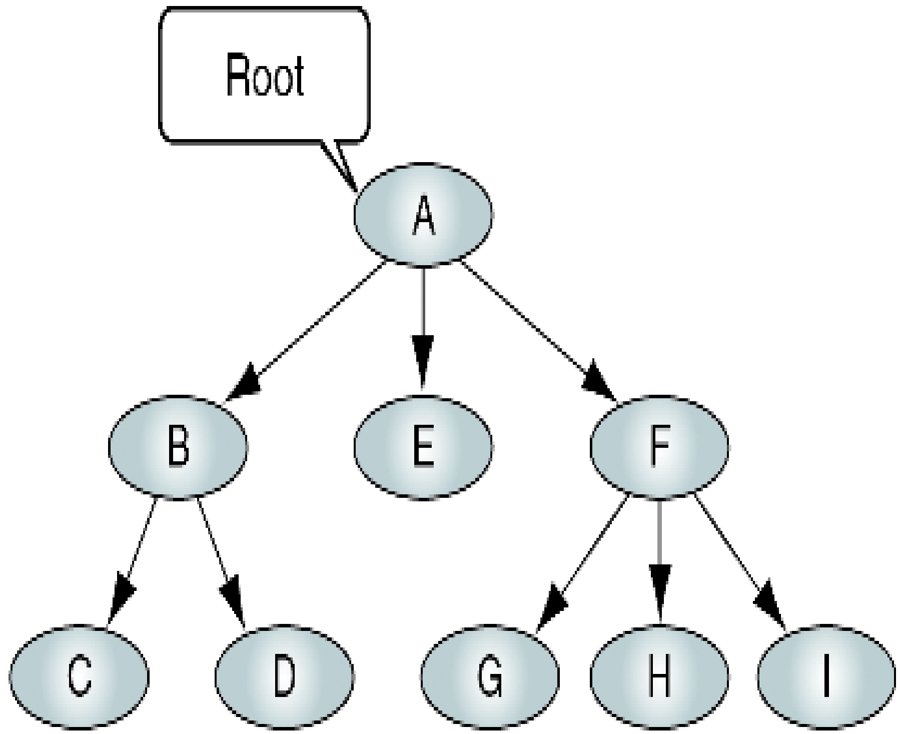
Yes.



No.

Basic Tree Concepts

- An undirected graph that does not contain simple circuits and is not necessarily connected is called a **forest**
- In general, we use trees to represent **hierarchical structures**
- We often designate a particular vertex of a tree as the **root**. Since there is a unique path from the root to each vertex of the graph, we direct each edge away from the root
- Thus, a tree together with its root produces a directed graph called a rooted tree



Tree

Basic Tree Concepts

- **In a rooted tree**:

➢ When the branch is directed toward the node, it is indegree branch

➢ When the branch is directed away from the node, it is an outdegree branch

➢ The sum of the indegree and outdegree branches is the degree of the node

Basic Tree Concepts

- **In a rooted tree**:
- The indegree of the root is, by definition, zero.
- With the exception of the root, all of the nodes in a tree must have an indegree of exactly one; **that is, they may have only one predecessor**
- All nodes in the tree can have zero, one, or more branches leaving them; **that is, they may have outdegree of zero, one, or more**

Basic Tree Concepts

- **In a rooted tree**:
- A leaf is any node with an outdegree of zero, that is, a node with no successors
- A node that is not a root or a leaf is known as an internal node.
- A node is a parent if it has successor nodes; that is, if it has outdegree greater than zero.
- A node with a predecessor is called a child

Basic Tree Concepts

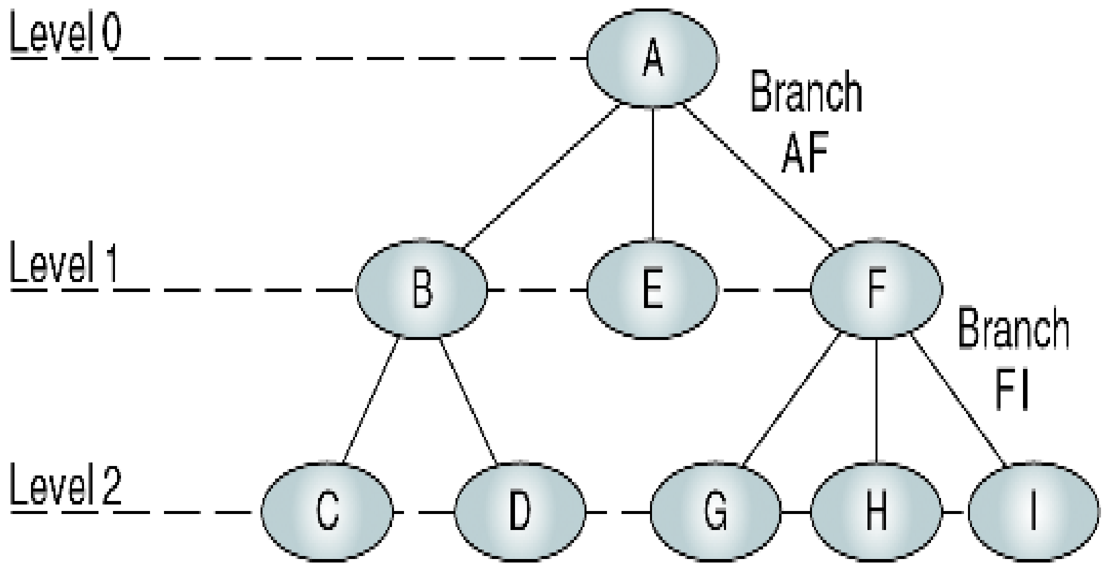
- **In a rooted tree**:
- Two or more nodes with the same parents are called siblings
- An ancestor is any node in the path from the root to the node.
- A descendant is any node in the path below the parent node; that is, all nodes in the paths from a given node to a leaf are descendants of that node.

Basic Tree Concepts

- A path is a sequence of nodes in which each node is adjacent to the next node.
- The level of a node is its distance from the root. The root is always at level 0, its children are at level 1, etc. …

Basic Tree Concepts

- The height of the tree is the level of the leaf in the longest path from the root plus 1.
  By definition the height of any empty tree is -1
- A subtree is any connected structure below the root. The first node in the subtree is known as the root of the subtree.



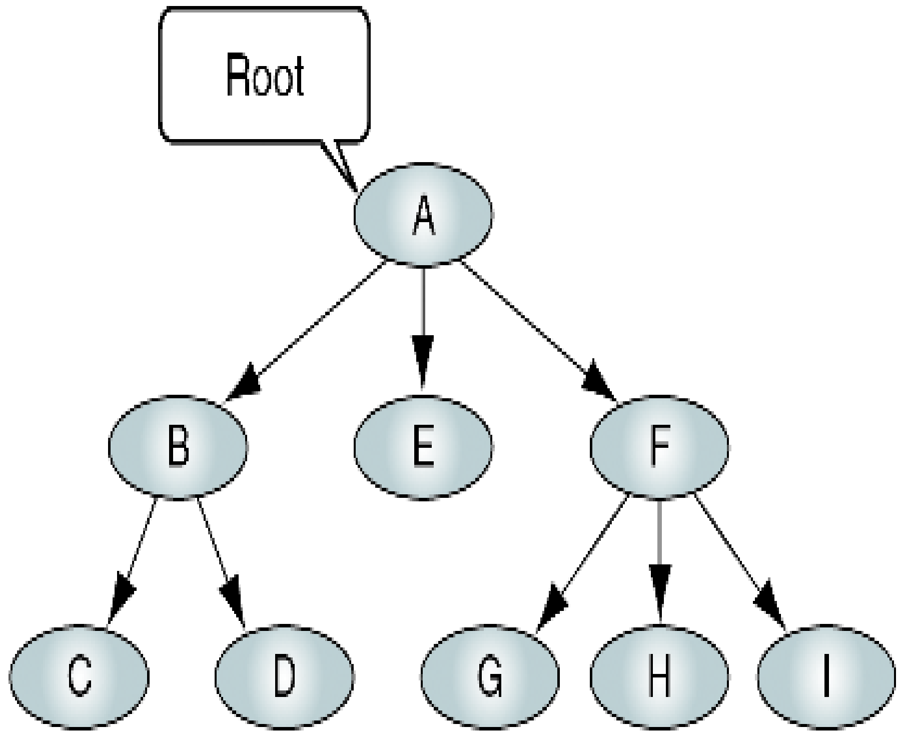
Root:	A	Siblings:	{B,E,F}, {C,D}, {G,H,I}
Parents:	A, B, F	Leaves:	C,D,E,G,H,I
Children:	B, E, F, C, D, G, H, I	Internal nodes:	B,F

Tree Nomenclature

Parenthetical Listing

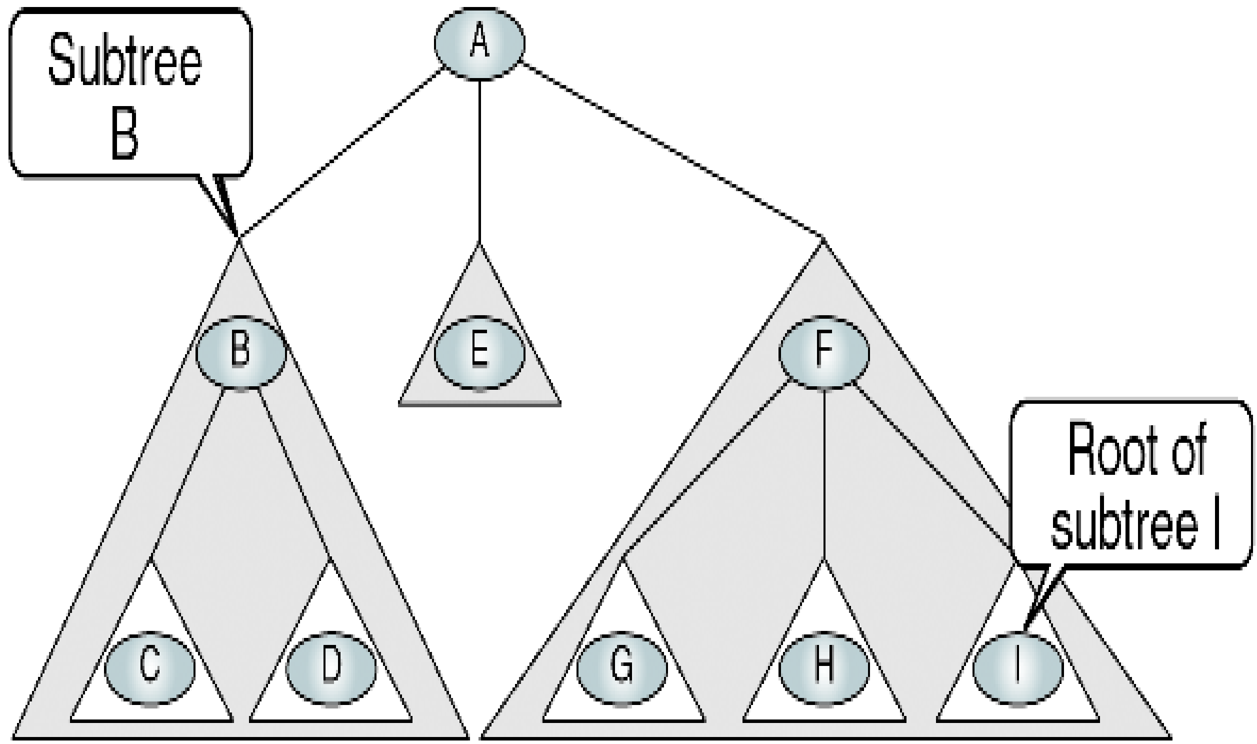
- Parenthetical Listing – the algebraic expression, where each open parenthesis indicates the start of a new level and each closing parenthesis completes the current level and moves up one level in the tree
- Parenthetical Listing is a primary method used for definition of and performing operations with the trees

Parenthetical Listing



Tree

A (B (C D) E F (G H I))



Subtrees

Recursive definition of a tree

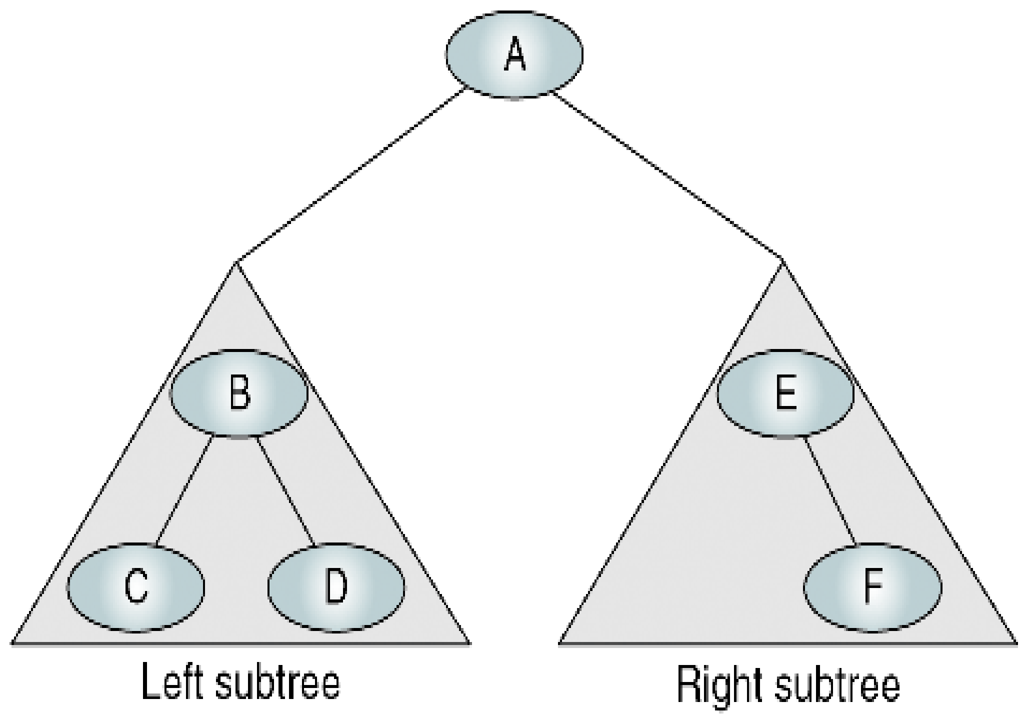
- A tree is a set of nodes that either:
  - is empty or
  - has a designated node, called the root, from which hierarchically descend zero or more subtrees, which are also trees

m-ary Trees

- A rooted tree is called an m-ary tree if every internal vertex has no more than m children
- The tree is called a full m-ary tree if every internal vertex has exactly m children
- An m-ary tree with $m = 2$ is called a **binary tree**
- A binary tree with n vertices has $(n – 1)$ edges
- Theorem. A full m-ary tree with i internal vertices contains $n = mi + 1$ vertices

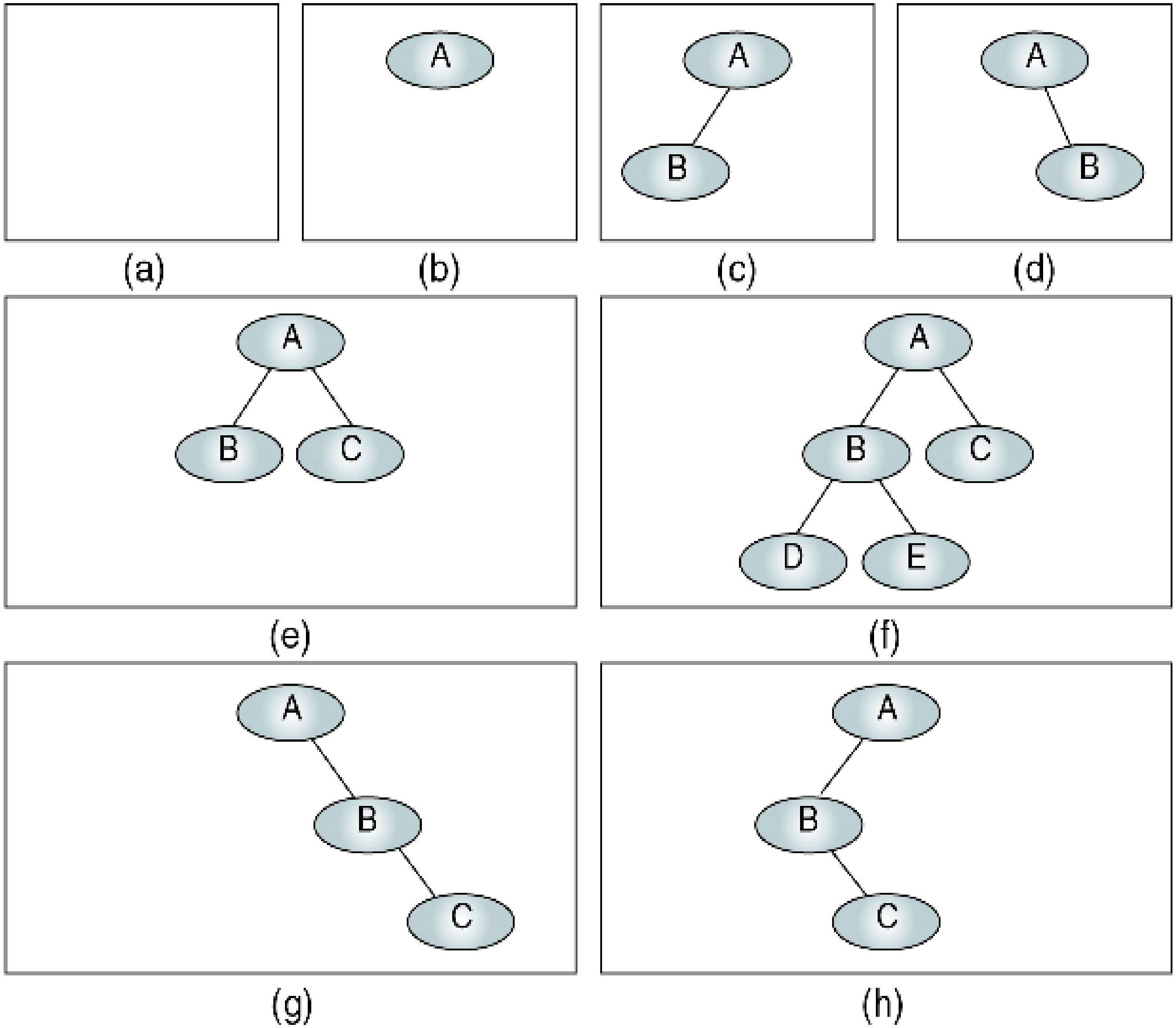
Binary Trees

- A **binary tree** is a tree in which no node can have more than two subtrees; the maximum outdegree for a node is 2
- In other words, a node can have 0, 1, or 2 subtrees
- These subtrees are designated as the left subtree and the right subtree



Binary Tree

A **null tree** is a tree with no nodes



Collection of Binary Trees

Some Properties of Binary Trees

- The height of binary trees can be mathematically predicted
- Given that we need to store N nodes in a binary tree, the maximum height is

$$H_{\max} = N$$

A tree with a maximum height is rare. It occurs when all of the nodes in the entire tree have only one successor.

Some Properties of Binary Trees

- The minimum height of a binary tree is determined as follows:

$$H_{\min} = [\log_2 N] + 1$$

For instance, if there are three nodes to be stored in the binary tree (N=3) then $H_{\min}$=2.

Some Properties of Binary Trees

- Given a height of the binary tree, H, the minimum number of nodes in the tree is given as follows:

$$N_{\min} = H$$

Some Properties of Binary Trees

- The formula for the maximum number of nodes is derived from the fact that each node may have only two descendents. Given a height of the binary tree, H, the maximum number of nodes in the tree is given as follows:

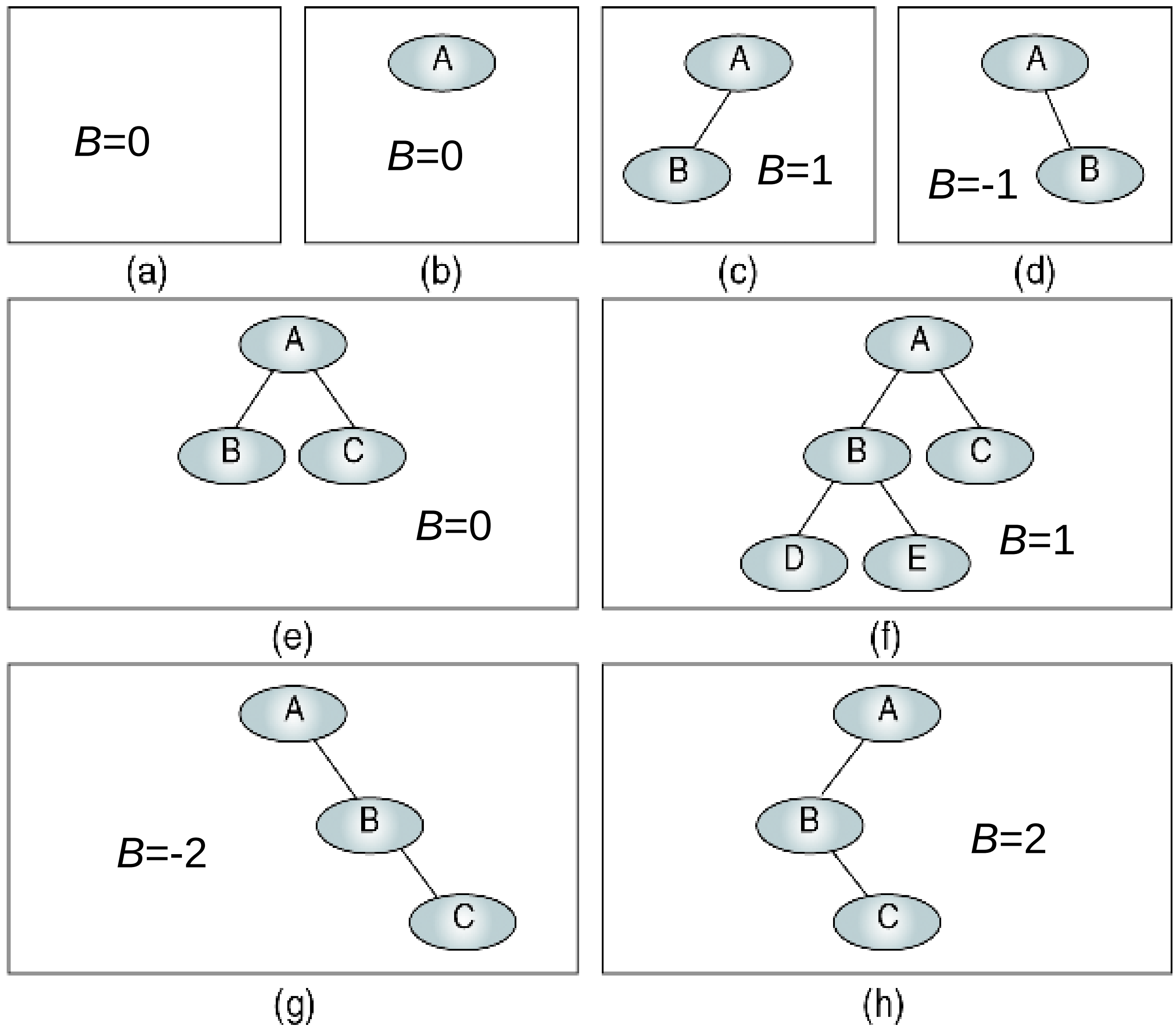
$$N_{\max} = 2^{H} - 1$$

Some Properties of Binary Trees

- The children of any node in a binary tree can be accessed by following only one branch path, the one that leads to the desired node
- The nodes at level 1, which are children of the root, can be accessed by following only one branch; the nodes of level 2 of a tree can be accessed by following only two branches from the root, etc.
- The **balance factor** of a binary tree is the difference in height between its left and right subtrees:

$$B = H_L - H_R$$

Balance of the binary tree



Collection of Binary Trees

Some Properties of Binary Trees

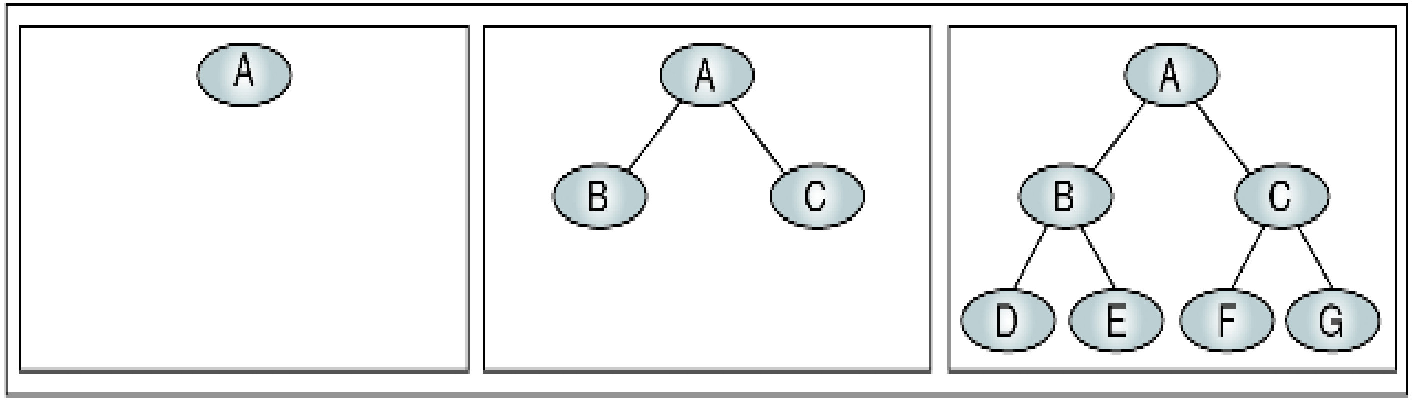
- A binary tree is called balanced (definition by Russian mathematicians Adelson-Velskii and Landis) if the height of its left and right subtrees differs by no more than one (its balance factor is -1, 0, or 1), and its subtrees are also balanced

Generalization for m-ary Trees

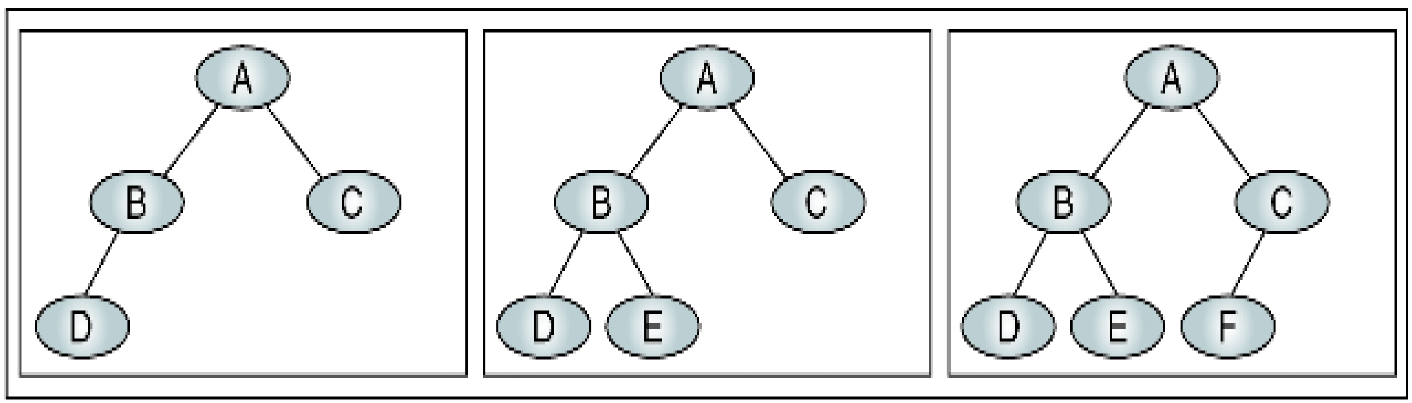
- In general, a rooted m-ary tree with height h is called balanced if all leaves are at levels h-1 or $h-2$
- Theorem. There are at most m^h leaves in an m-ary tree of height h
  - **Corollary:** An m-ary tree with λ leaves has height $h \geq \lceil \log_m \lambda \rceil$. If the tree is full and balanced then $h = \lceil \log_m \lambda \rceil$

Complete and nearly complete binary trees

- A complete tree has the maximum number of entries for its height. The maximum number is reached when the last level is full
- A tree is considered nearly complete if it has the minimum height for its nodes and all nodes in the last level are found on the left



(a) Complete trees (at levels 0, 1, and 2)



(b) Nearly complete trees (at level 2)

Complete and Nearly Complete Trees

Binary Search Trees

- If we want to perform a large number of searches in a particular list of items, it can be worthwhile to arrange these items in a binary search tree to facilitate the subsequent searches
- A binary search tree is a binary tree in which each child of a vertex is designated as a right or left child, and each vertex is labeled with a key
- When we construct the tree, vertices are assigned keys so that the key of a vertex is: larger than the keys of all vertices in its left subtree and smaller than the keys of all vertices in its right subtree

Binary Search Trees

- To perform a search in such a tree for an item ***x***, we can start at the root and compare its key to x. If ***x*** is less than the key, we proceed to the left child of the current vertex, and if ***x*** is greater than the key, we proceed to the right one

- This procedure is repeated until we either found the item we were looking for, or we cannot proceed any further.

- In a balanced tree representing a list of n items, search can be performed with a maximum of $\lceil \log(n+1) \rceil$ steps

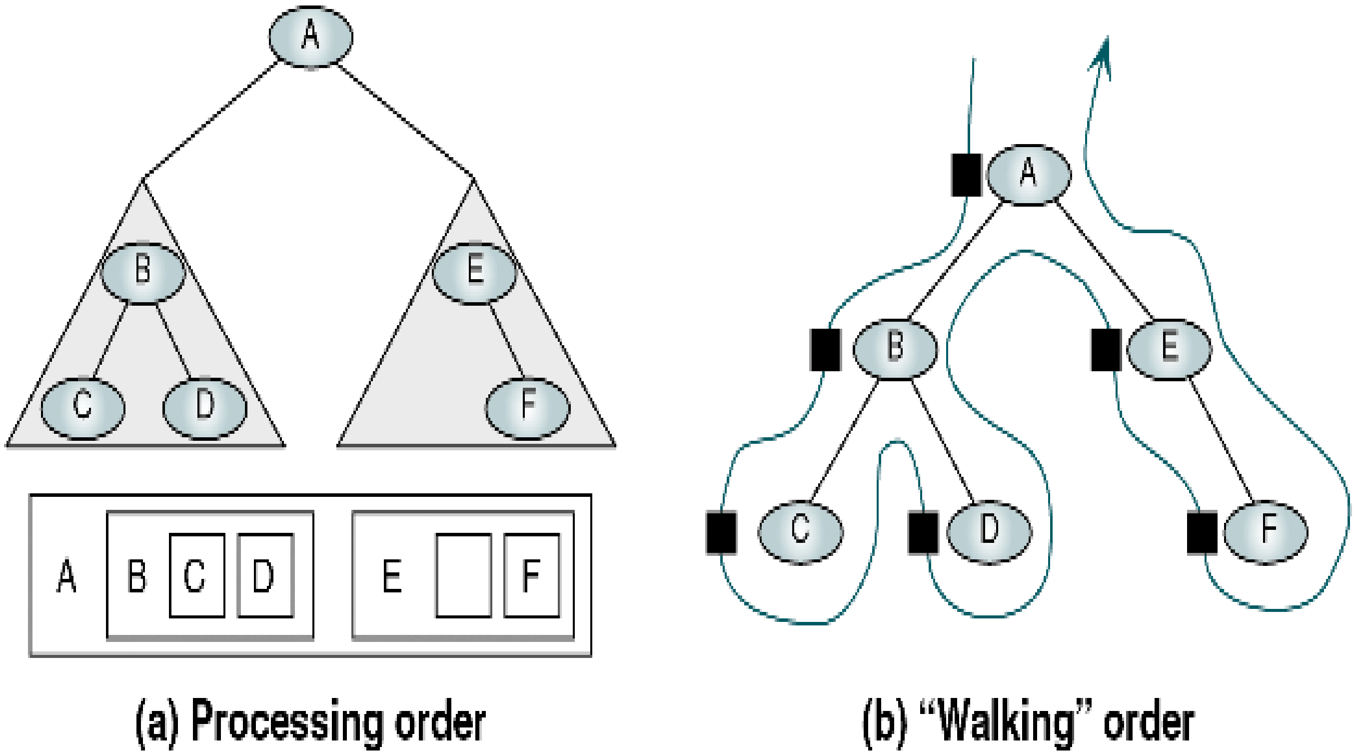
Decision Trees

- A decision tree represents a decision-making process.
  - Each possible "decision point" or situation is represented by a node.
  - Each possible choice that could be made at that decision point is represented by an edge to a child node.
- In the extended decision trees used in decision analysis, we also include nodes that represent random events and their outcomes.

Binary Tree Traversal

- A binary tree traversal requires that each node of the tree be processed once and only once in a predetermined sequence.
- In the depth-first traversal processing goes along a path from the root through one child to the most distant descendant of that first child before processing a second child.

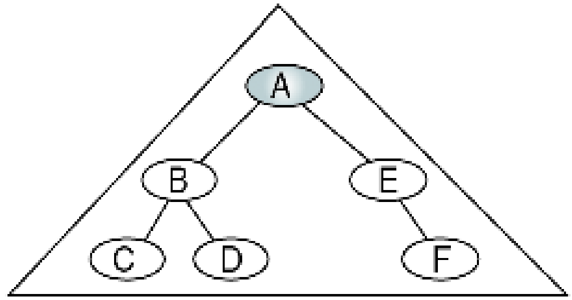
Depth-First and Preorder Traversal



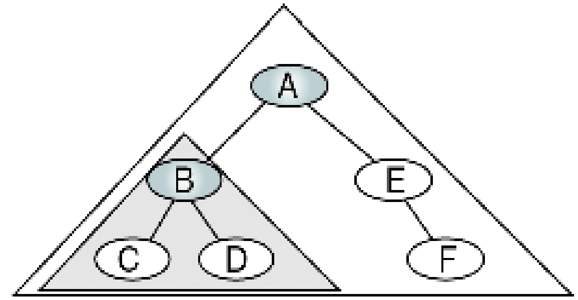
Depth-First and Preorder

Traversal—A B C D E F

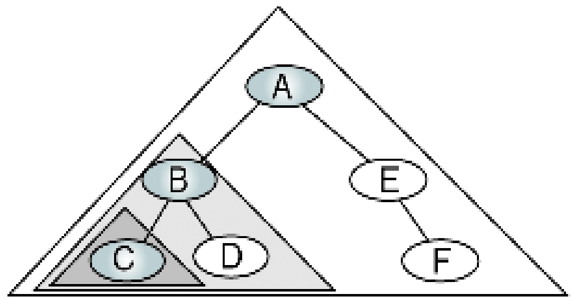
Preorder traversal starts at the root and always continues from the left-most branch to the right-most branch



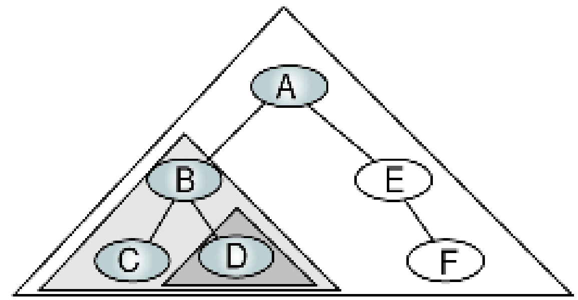
(a) Process tree A



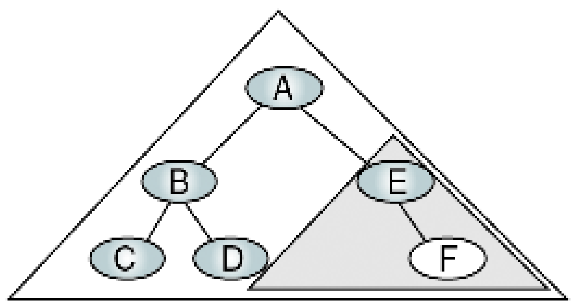
(b) Process tree B



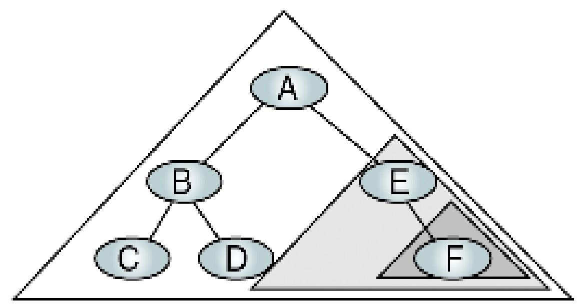
(c) Process tree C



(d) Process tree D



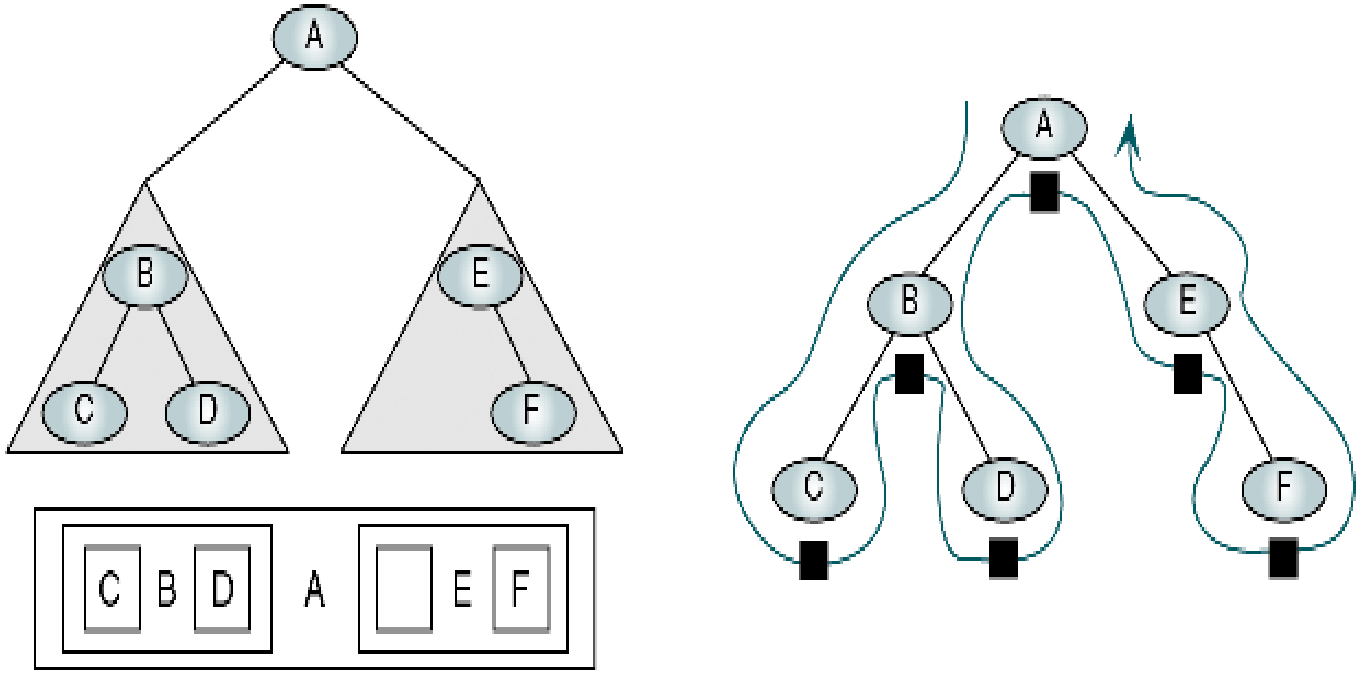
(e) Process tree E



(f) Process tree F

Depth-First Traversal of Binary Tree

Inorder Traversal

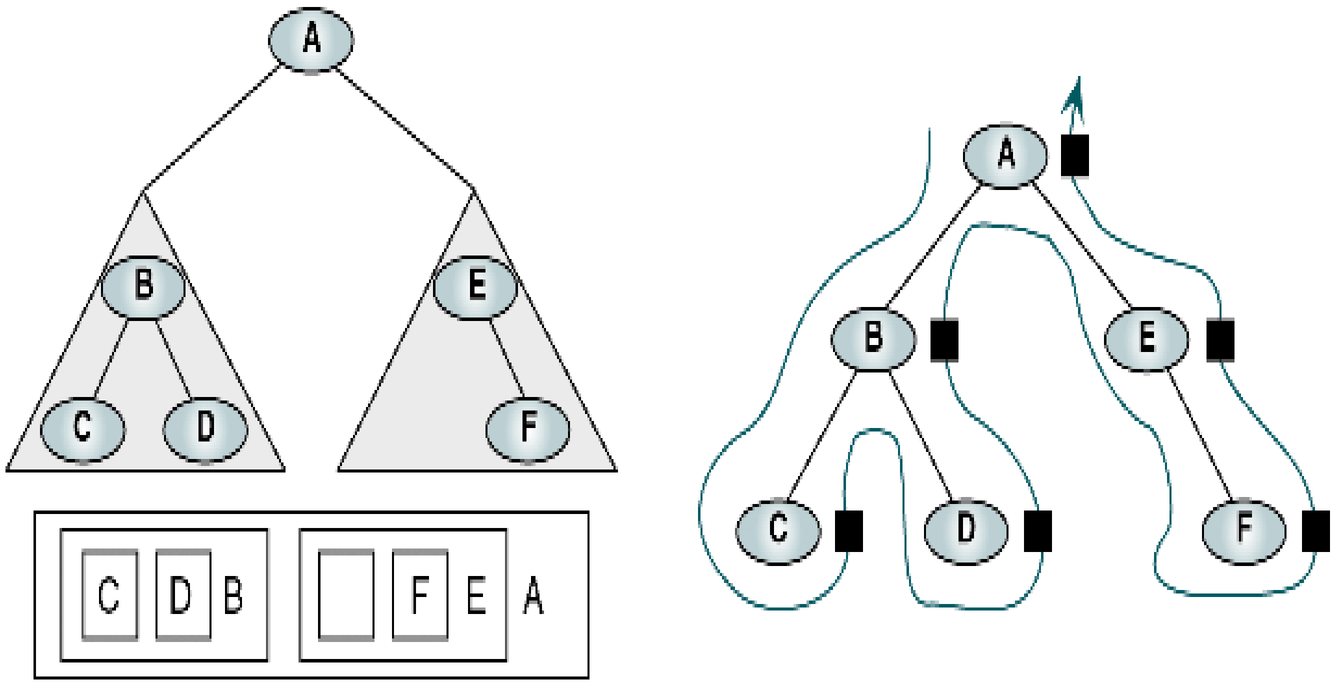


(a) Processing order (b) "Walking" order

Inorder Traversal—C B D A E F

Inorder traversal starts at the left-most leave and always continues from the left-most branch to the right-most branch through the root

Postorder Traversal

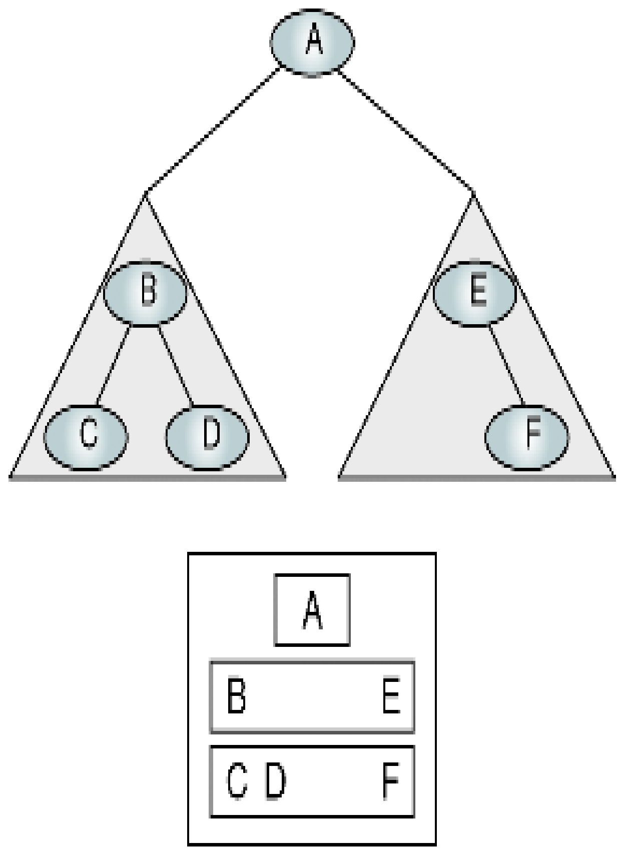


(a) Processing order

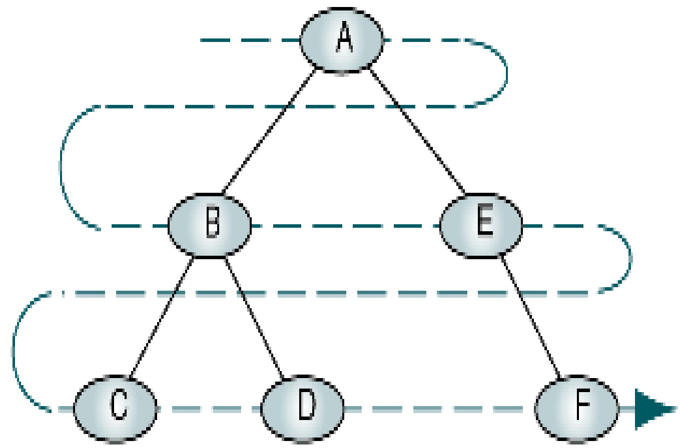
(b) "Walking" order

Postorder Traversal—C D B F E A

Postorder traversal starts at the left-most leave and always continues from the last level of the left subtree to the root through the right subtree, from the bottom to the top



(a) Processing order



(b) "Walking" order

Breadth-first Traversal

Breadth-first traversal starts at the root and always continues from the first level to the last level pathing levels from left to right

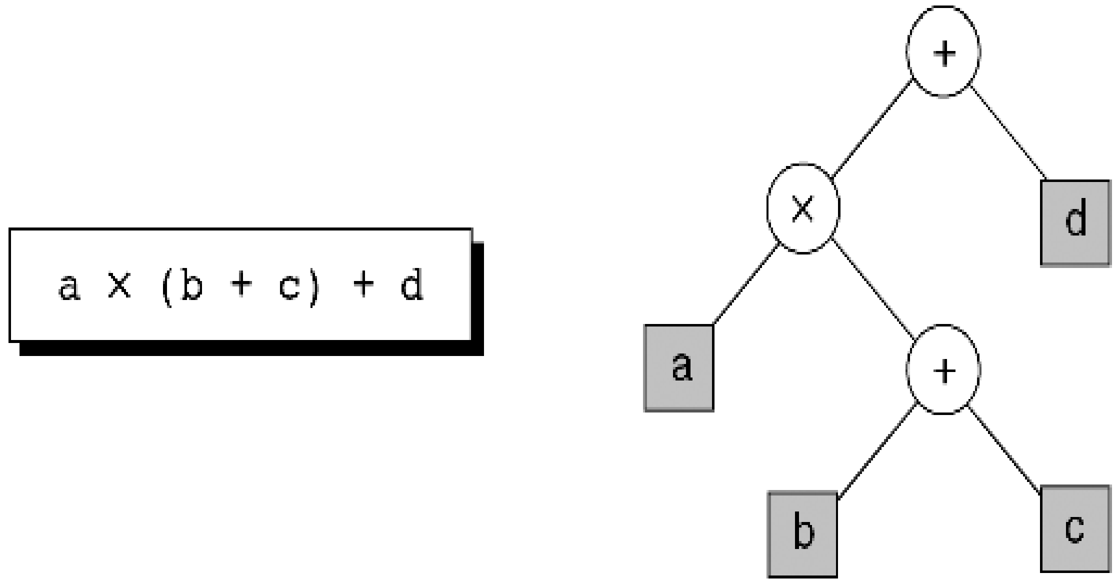
Decomposition of an arithmetic or logical expression by a compiler

- An expression shall be represented using a rooted tree in such a way that
  - ➢ The internal nodes represent operations
  - ➢ The leaves represent the variables or constant
  - ➢ Each operation operates on its left and right subtrees

Decomposition of an arithmetic or logical expression by a compiler

- We obtain the **infix form** of an expression if the inorder traversal is used. This requires to include parentheses in the traversal to avoid ambiguity
- We obtain the **prefix (Polish) form** of an expression if the preorder traversal is used. This representation is unambiguous, so no parentheses are needed

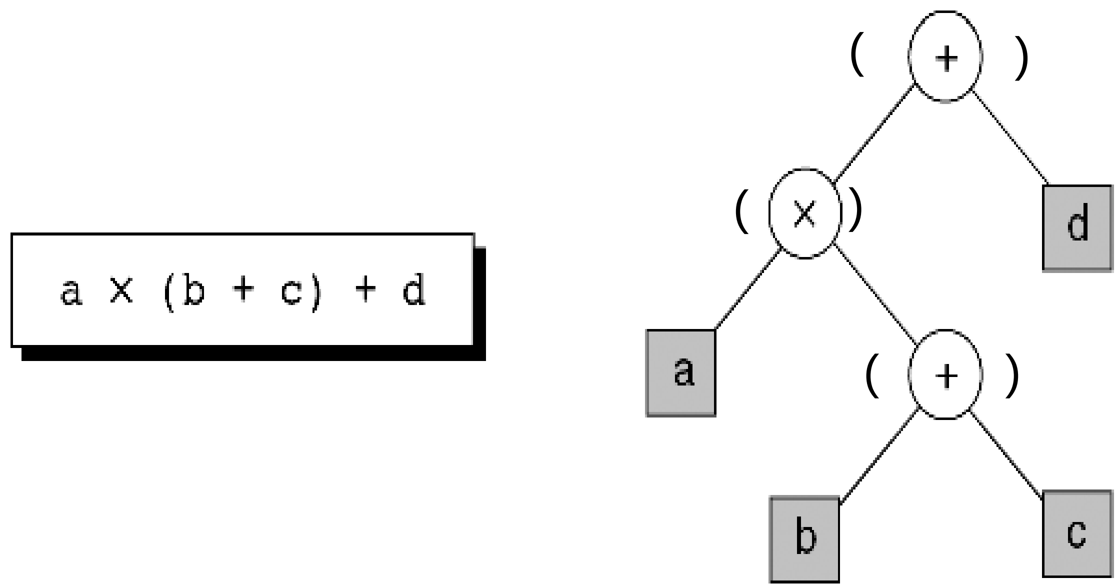
Example: Decomposition of an arithmetic expression by a compiler



Prefix Expression and Its Expression Tree

Preorder traversal returns + x a + b c d → +((x(a,+(b c)),d)

Example: Decomposition of an arithmetic expression by a compiler

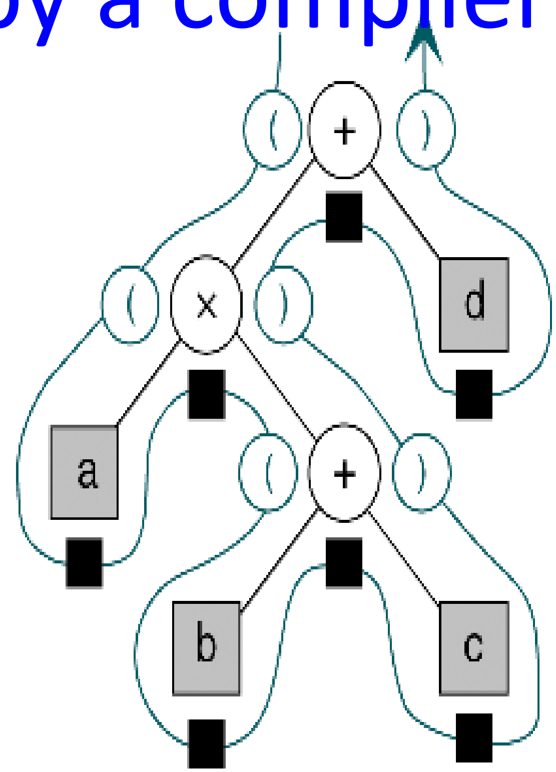


Infix Expression and Its Expression Tree

Inorder traversal returns a x (b + c) + d

Example: Decomposition of an arithmetic expression by a compiler

((a × (b + c)) + d)



Infix Traversal of an Expression Tree

Inorder traversal is employed