



Lecture 10 – UNIT 23

Abstract Data Types

抽象資料型態

Outlines

- 10.1 Background
- 10.2 Stacks
- 10.3 Queues
- 10.4 General linear lists
- 10.5 Trees
- 10.6 Binary trees
- 10.7 Binary search trees
- 10.8 Graphs

10.5

Trees

Trees

- A **tree** consists of a finite set of elements, called **nodes** (or **vertices, 結點**) and a finite set of directed **lines**, called **arcs**, that connect pairs of the nodes.
- We can divided the vertices in a tree into three categories: the ***root*** (根**結點**), ***leaves*** (葉子)and the *internal nodes* (分支結點或非終端結點).

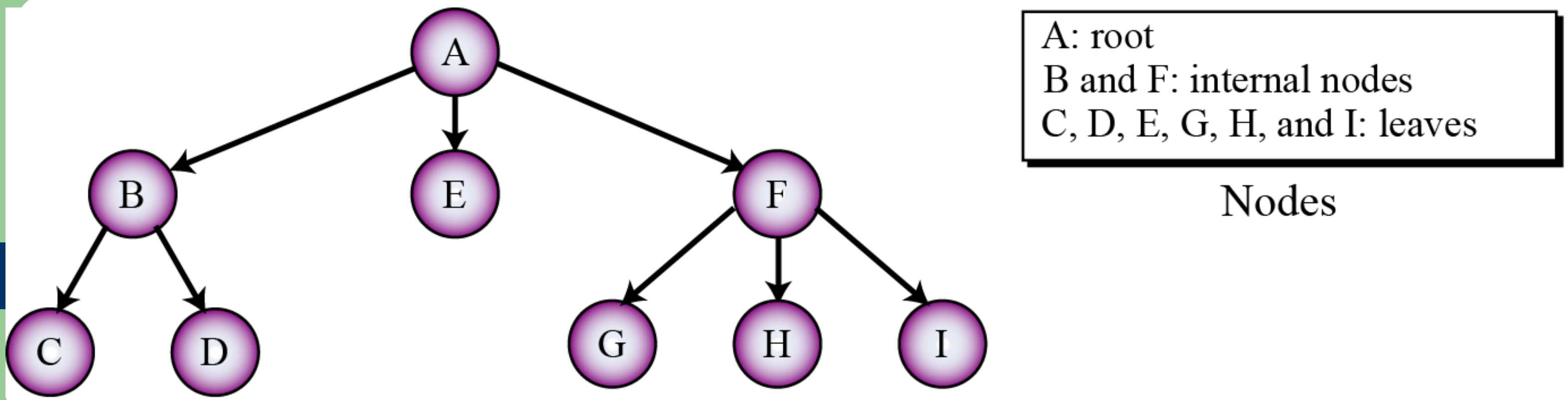


Figure 10.20 **Tree representation**

Table 10.1 Number of incoming and outgoing arcs

Type of node	*Incoming arc*	*Outgoing arc*
root	0	0 or more
leaf	1	0
internal	1	1 or more

Subtree (子樹)

- Each node in a tree may have a **subtree**.

- The subtree of each node includes one of its children(孩子) and all descendents(子孫) of that child. Figure 10.21 shows all subtrees for the tree in Figure 10.20.

Subtree (continued)

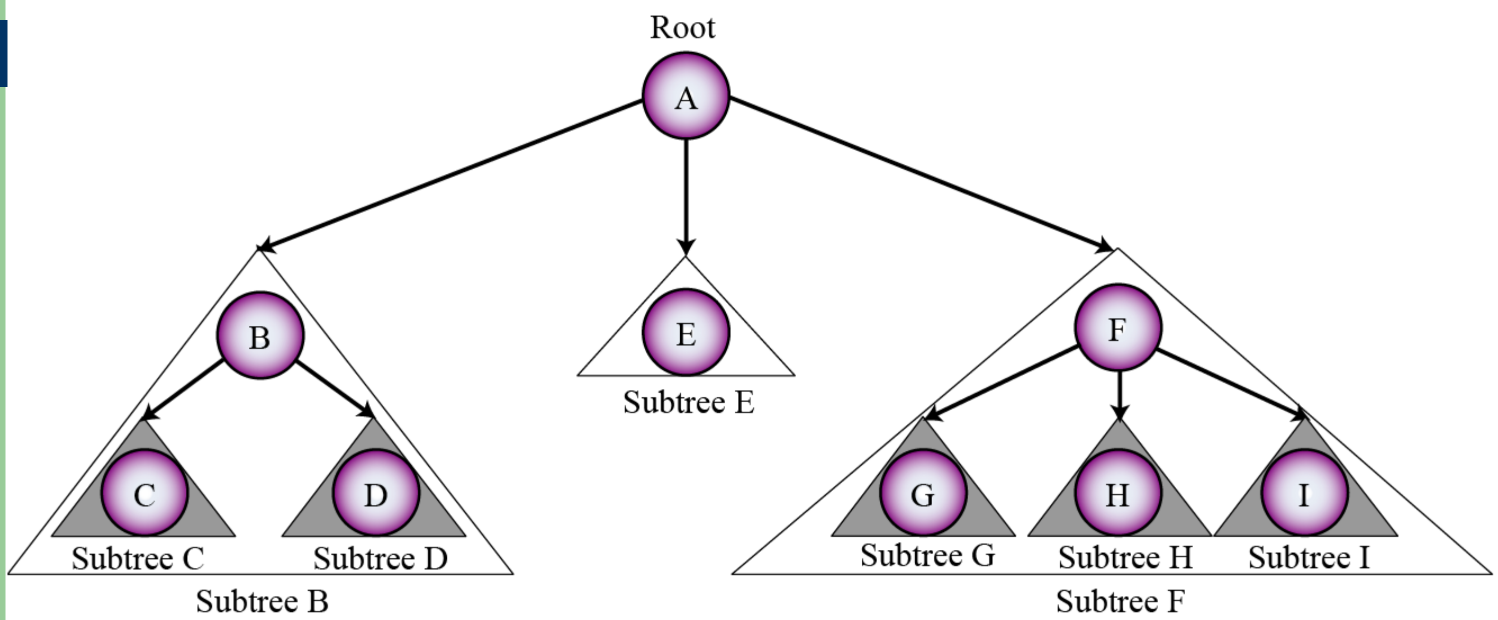


Figure 10.21 Subtrees

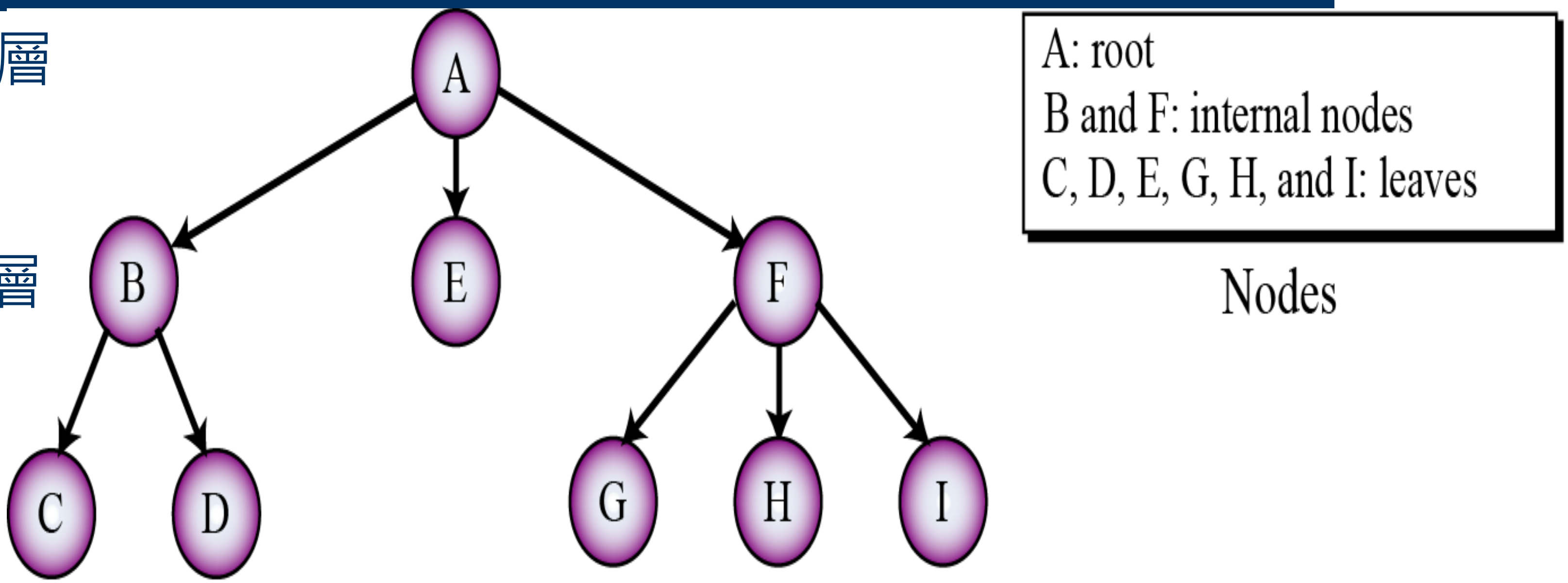
常用術語

- 度 (degree)
  indegree, outdegree (degree)
- 兄弟 (siblings), 堂兄弟
- 祖先 (ancestor)
- 層次 (level), root (level 1), level 2, …
- 深度 (depth): 樹的最大層數
- 有序樹 (ordered tree)
- 森林 (forest)

第一層

第二層

第三層



Nodes

G, H, I 為兄弟

此樹深度為3

D 與G, H, I 互為堂兄弟

10.6

Binary Trees

Binary Tree

- A binary tree is a tree in which no node can have more than two subtrees. In other words, a node can have zero, one or two subtrees.

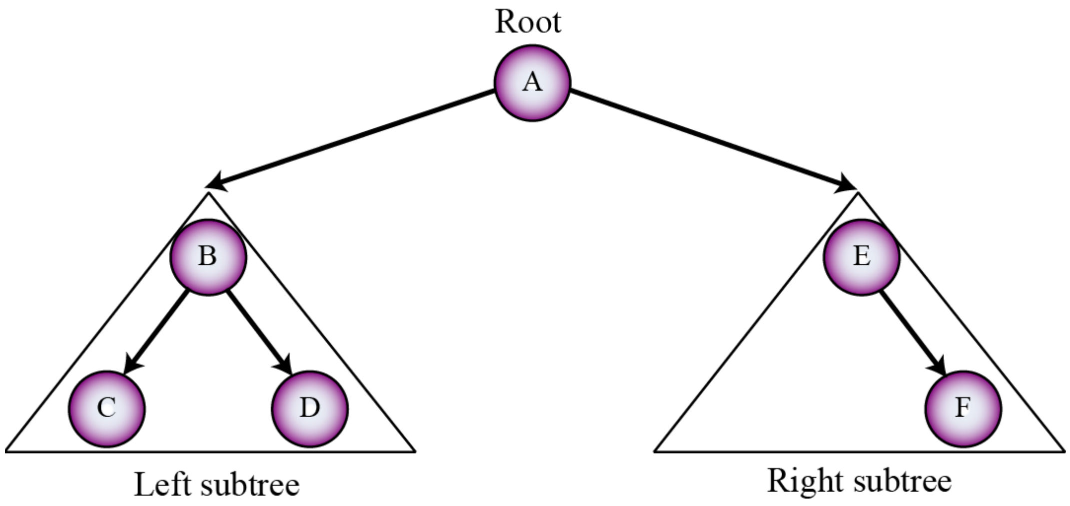


Figure 10.22 **A binary tree**

Recursive definition of binary trees

- The following gives the recursive definition of a binary tree. Note that, based on this definition, a binary tree can have a root, but each subtree can also have a root.

Binary tree

Definition	A binary tree is either empty or consists of a node, *root*, with two subtrees, in which each subtree is also a binary tree.

Examples of binary trees

- The figure shows eight trees, the first of which is an empty binary tree (sometimes called a null binary tree).

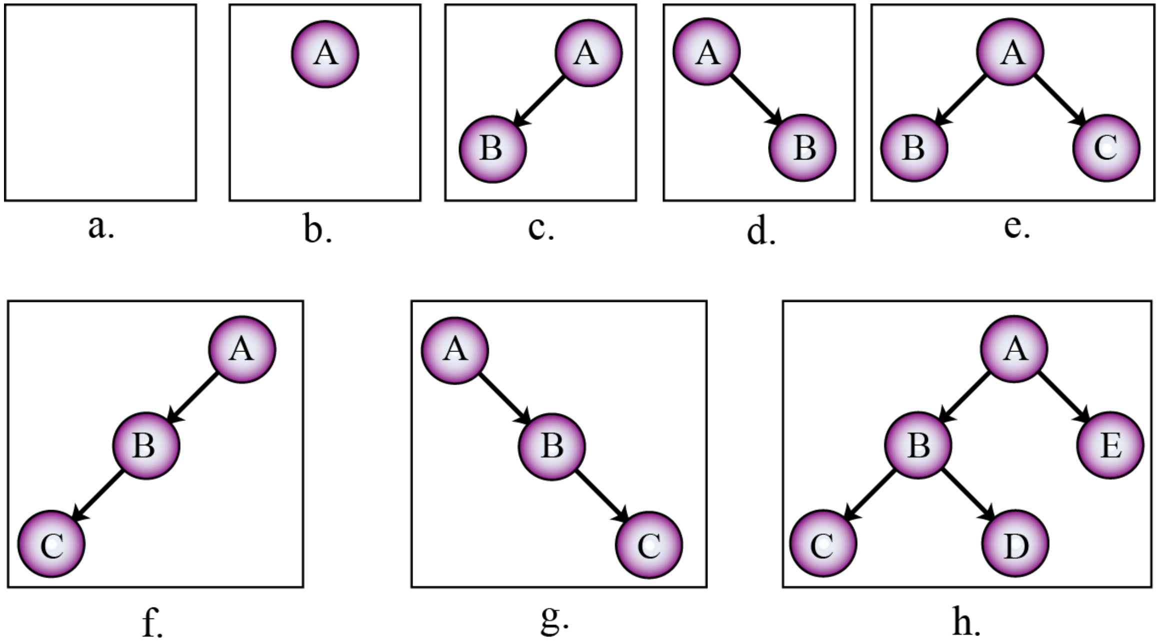


Figure 10.23 Examples of binary trees

Full Binary Tree

- 除葉子結點外，每個結點都有兩個孩子
- 第一層1個結點(root),第二層2個結點，第三層4個結點，…，第k層2^{k-1} 個結點
- 深度為K的full binary tree共有2^k - 1個結點

- Complete binary tree(完全二叉樹)

類似full binary tree ，但最後一層未填滿葉子，只有左邊有部分葉子

思考一下

- 任何一棵binary tree ，若葉子結點數為 x，outdegree(度)為2的結點數為 y

  $x = y + 1$

  Why?

  $x + y + m = 2y + m + 1$

Operations on binary trees

- The six most common operations defined for a binary tree are ***tree*** (creates an empty tree), ***insert***, ***delete***, ***retrieve***, ***empty*** and ***traversal***.
- The first five are complex and beyond the scope of this book. We discuss binary **tree traversal** in this section.

Binary tree traversals

- A binary tree traversal requires that each node of the tree be processed once and only once in a predetermined sequence.
- The two general approaches to the traversal sequence are **depth-first** and **breadth-first** traversal.

中左右　　左中右　　左右中

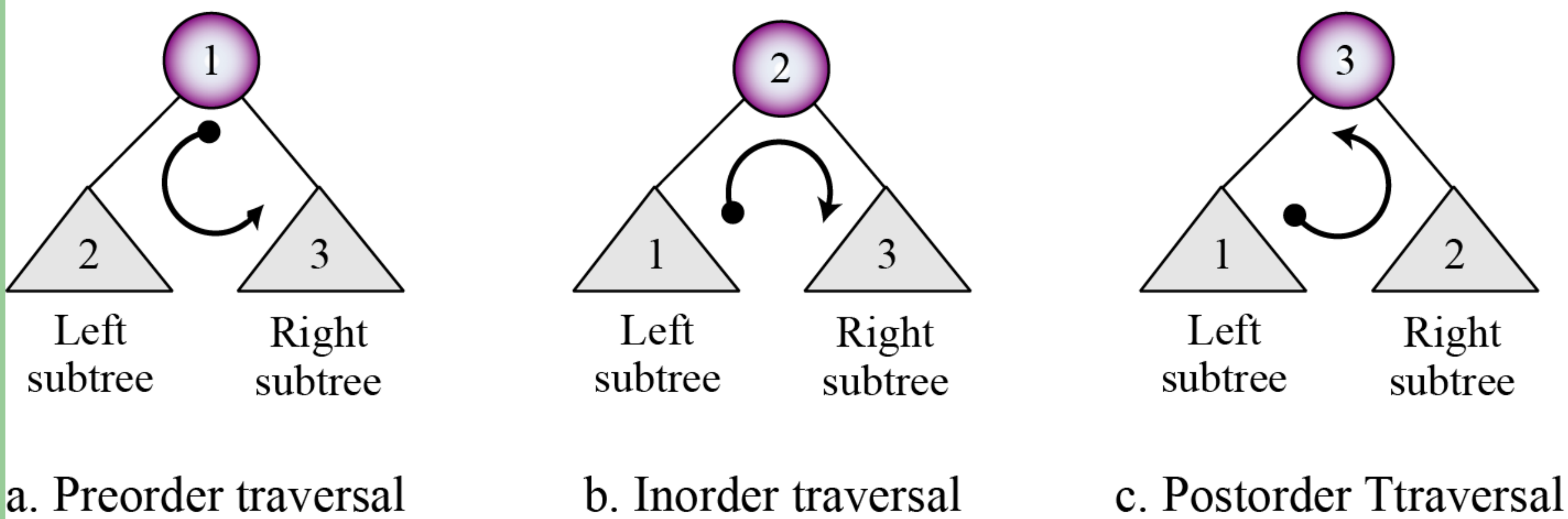


Figure 10.24 **Depth-first traversal of a binary tree**

Example 10.10

- Figure 10.25 shows how we visit each node in a tree using preorder traversal. The figure also shows the walking order.

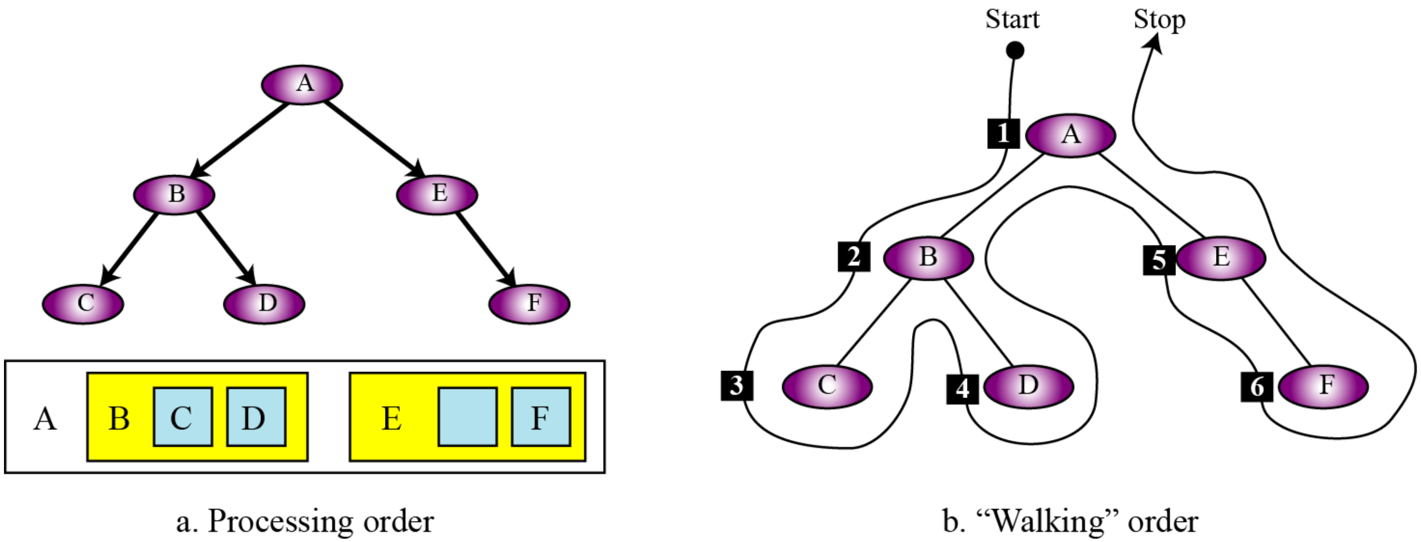


Figure 10.25 Example 9.10

Example 10.11

- Figure 10.26 shows how we visit each node in a tree using breadth-first traversal. The figure also shows the walking order.

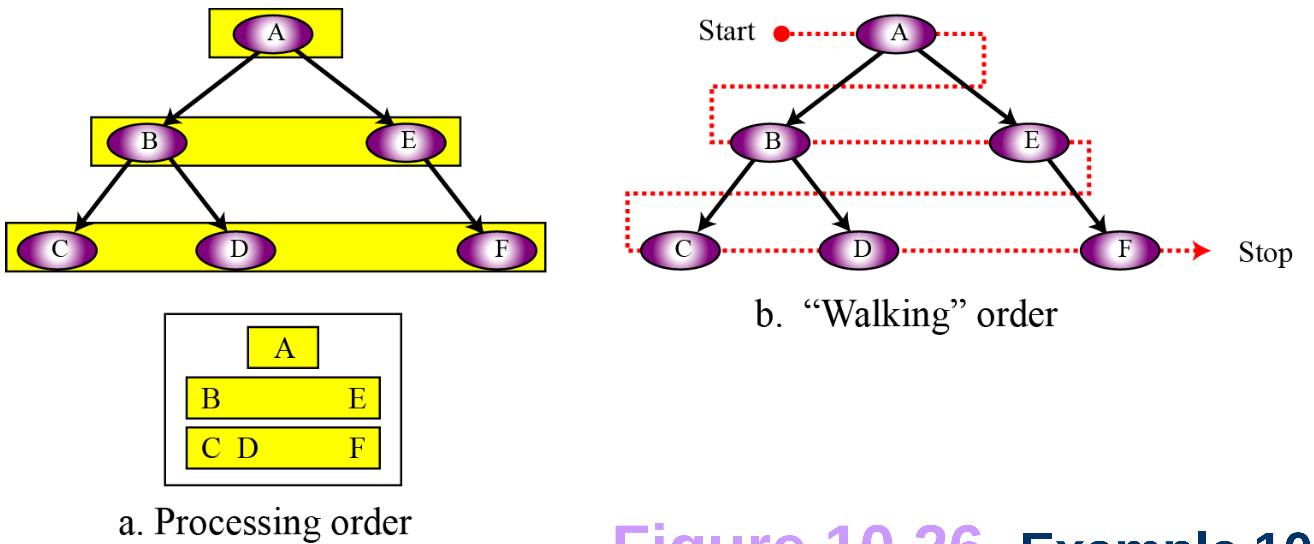


Figure 10.26 Example 10.11

Binary tree applications

- Binary trees have many applications in computer science. In this section we mention only two of them: Huffman coding and expression trees.
- **Huffman coding**
  - Huffman coding is a compression technique that uses binary trees to generate a variable length binary code from a string of symbols.

Expression trees

- An arithmetic expression can be represented in three different formats: **infix**, **postfix** and **prefix**.
  - In an infix notation, the operator comes between the two operands.
  - In postfix notation, the operator comes after its two operands.
  - In prefix notation it comes before the two operands. These formats are shown below for addition of two operands A and B.

Prefix: + A B **Infix:** A + B **Postfix:** A B +

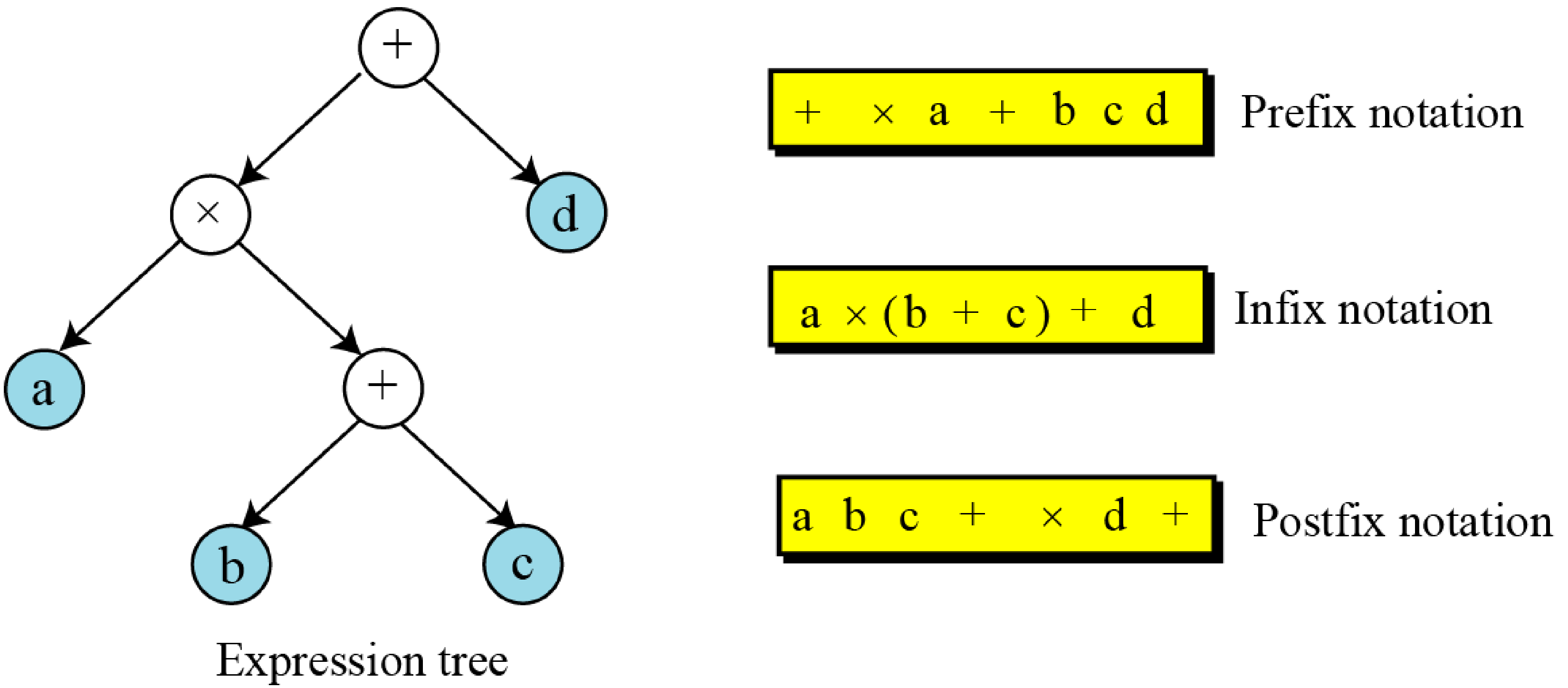


Figure 10.27 **Expression tree**