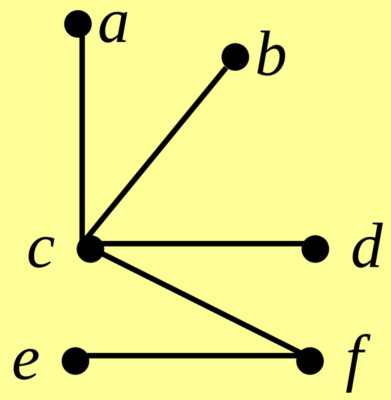
Chapter 7: Trees

- A **tree** is a connected simple undirected graph with no simple circuits.

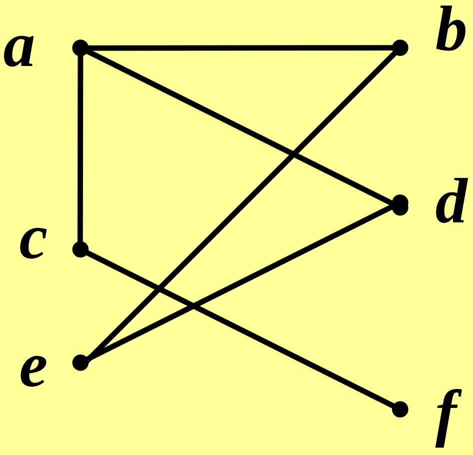
- **Properties:**
  - There is a unique simple path between any 2 of its vertices.
  - No loops.
  - No multiple edges.

Example 1



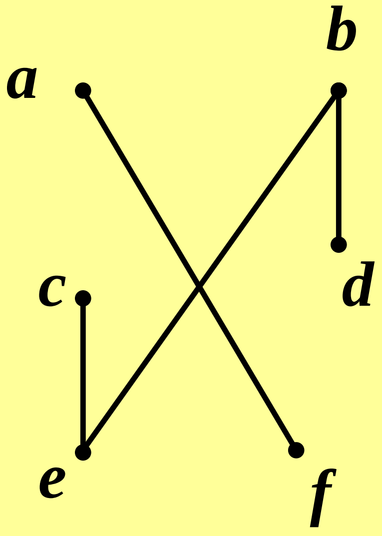
G1 : This graph is a <u>Tree</u> because it is a connected graph with no simple circuits

Example 2



G2: is <u>not a tree</u> " because there is a cycle *a*, *b*, *e*, *d*, *a*"

Example 3



G3: is <u>not a tree</u> "because it's not connected". In this case it's called forest in which each connected component is a tree.

Component 1: ***a, f***

Component 2: ***c, e, b, d***

Forest

- An undirected graph without simple circuits is called a **forest**.

  – You can think of it as a set of trees having disjoint sets of nodes.

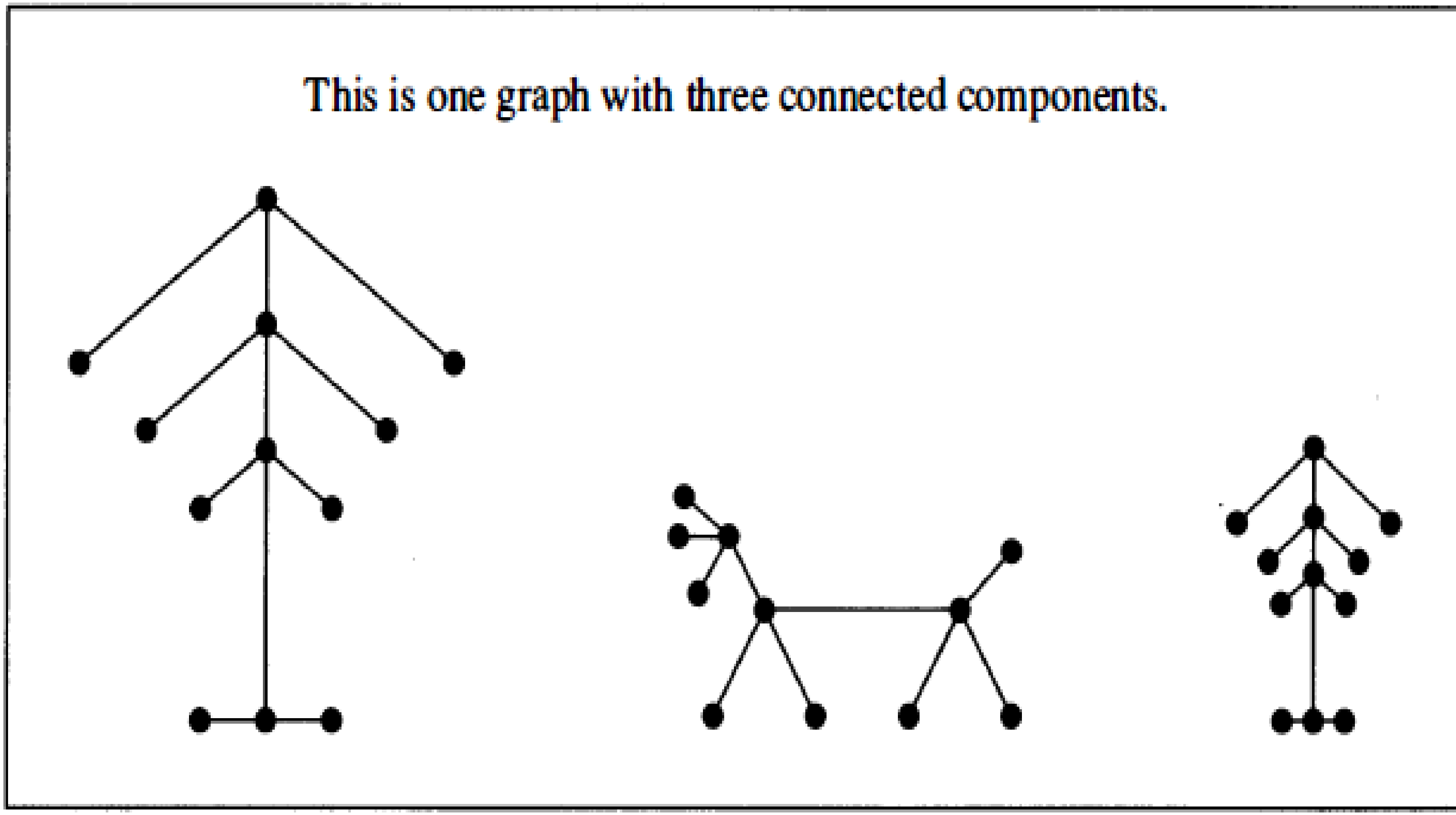
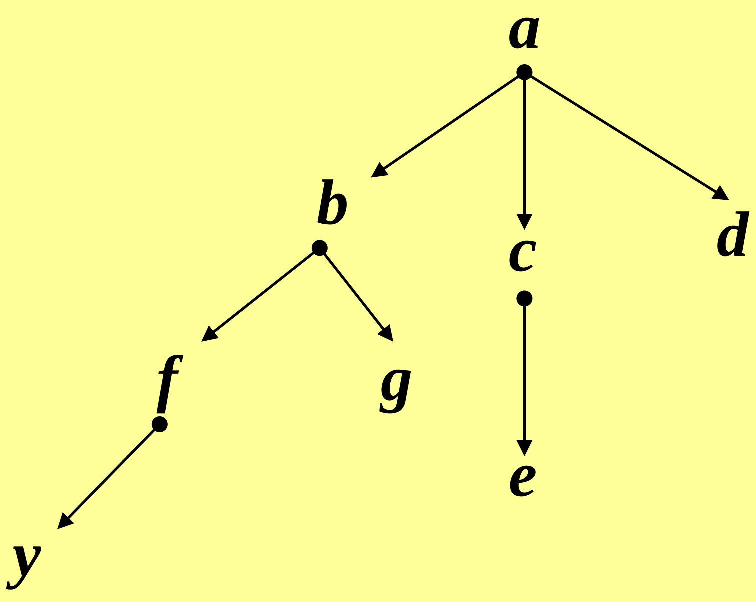


FIGURE 3 Example of a Forest.

Rooted (Directed) Trees

- A **rooted tree** is a tree in which one node has been designated the **root** and every edge is directed away from the root.
- You should know the following terms about rooted trees:
  - Root, Parent, Child, Siblings, Ancestors, Descendents, Leaf, Internal node, Subtree.

Definitions



- **Root**: Vertex with in-degree 0

[Node *a* is the root]

Definitions

- **Parent**: Vertex *u* is a parent, such that there is directed edge from *u* to *v*.

  [*b* is parent of *g* and *f*]
- **Child:** If *u* is parent of *v*, then *v* is child of *u*.

  [*g* and *f* are children of *b*]
- **Siblings:** Vertices with the same parents.

  [*f* and *g*]
- **Ancestors:** Vertices in path from the root to vertex *v*, excluding *v* itself, including the root.

  [Ancestors of *g* : *b, a*]

Definitions

- **Descendents**: All vertices that have *v* as ancestors.

  [**Descendents of *b* : *f*, *g*, *y***]

- **Leaf:** Vertex with no children.

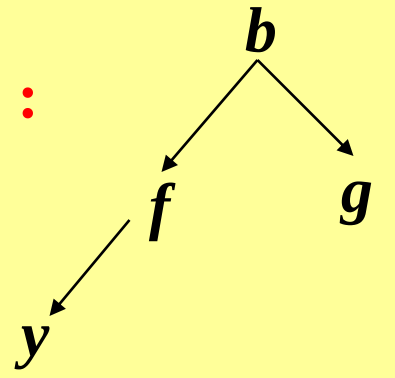
  [*y*, *g*, *e*, *d*]

- **Internal vertices:** Vertices that have children.

  [*a*, *b*, *c*, *f*]

- **Subtree:** Subgraphs consisting of *v* and its descendents and their incident edges.

  Subtree rooted at *b* :



Definitions

- **Level (of *v*)** is length of unique path from root to *v*.

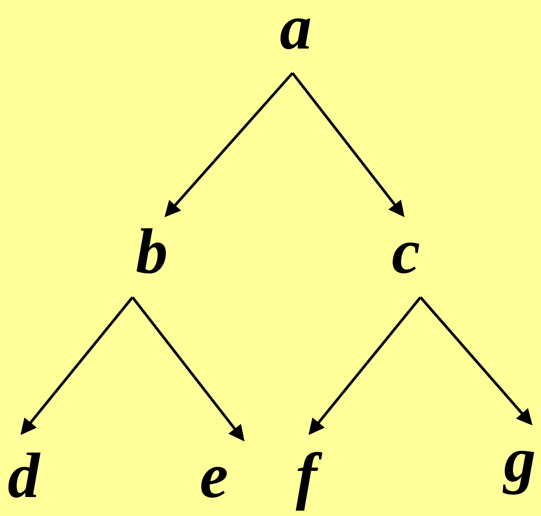
  [level of root = 0, level of *b* = 1, level of *g* = 2]
- **Height** is maximum of vertices levels.

  [Height = 3]

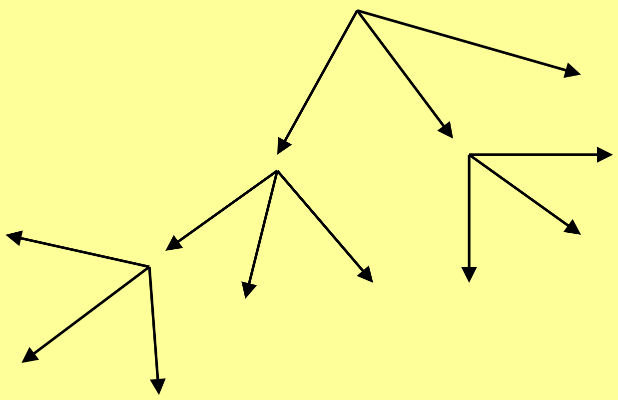
m-ary Trees

- A rooted tree is called ***m*-ary** if every internal vertex has no more than *m* children.
- It is called **full *m*-array** if every internal vertex has **exactly** *m* children.
- A 2-ary tree is called a **binary tree**.

Example



Full binary tree

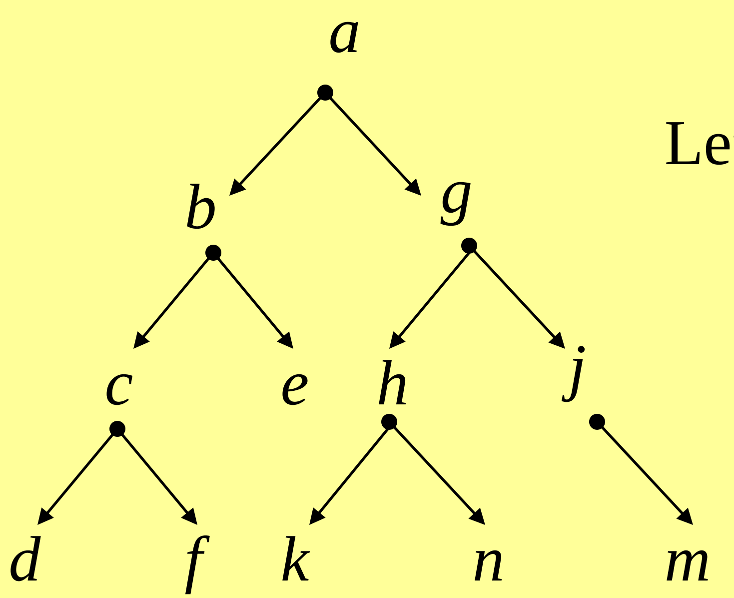


Full 3-ary tree

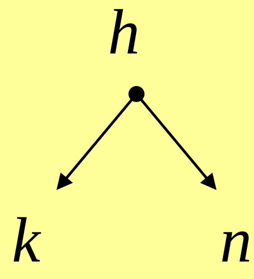
Ordered Rooted Tree

- A rooted tree where the children of each internal node are ordered.
- In ordered binary trees, we can define:
  - **left child, right child**
  - **left subtree, right subtree**
- For m-ary trees with $m > 2$, we can use terms like "leftmost", "rightmost," etc.

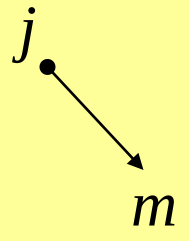
Examples



Left subtree of *g*



Right subtree of *g*



Left child of *c* is *d* , Right child of *c* is *f*

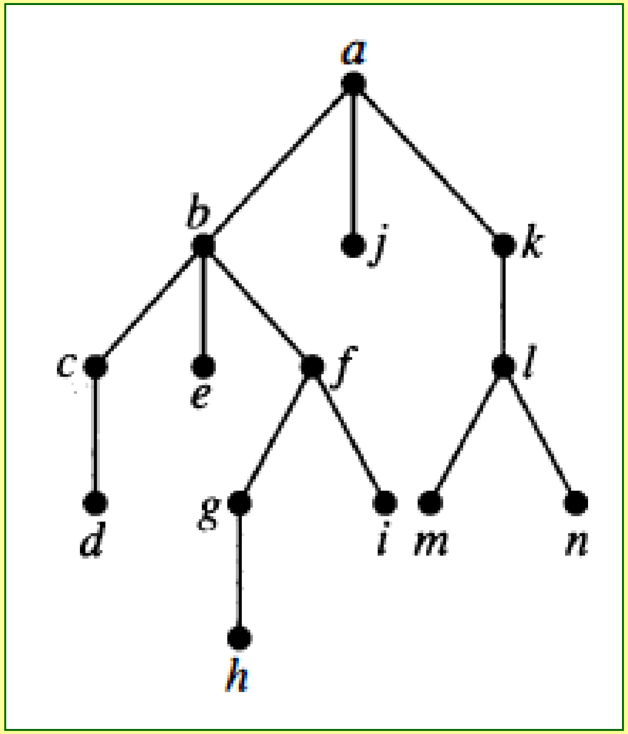
Properties of Trees

1- A tree with n vertices has

***n* - 1 edges.**

e.g. The tree in the figure has

14 vertices and 13 edges

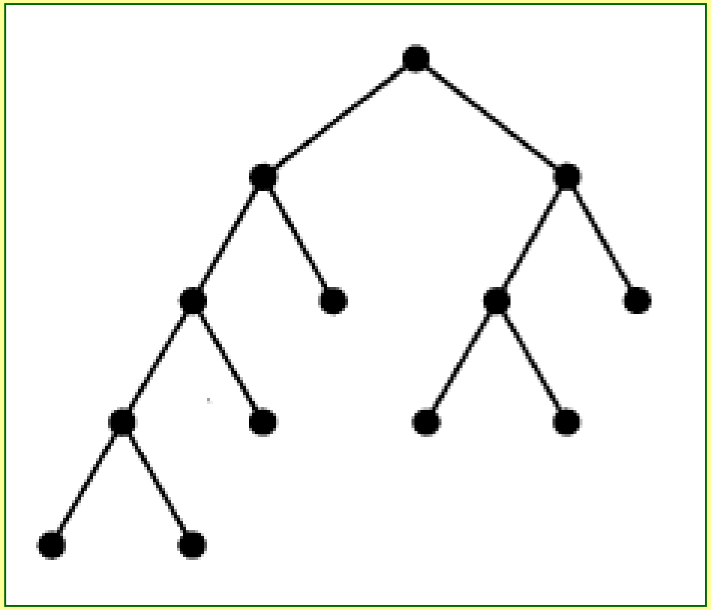


Properties of Trees

2- A full *m*-ary tree with *I* internal vertices and *L* leaves contains:

$$\boldsymbol{n = m \times I + 1} \textbf{ vertices}$$

$$\boldsymbol{n = I + L} \textbf{ vertices}$$



e.g. The full binary tree in the figure has:

Internal vertices $I = 6$

Leaves $L = 7$

Vertices $13 = (2)(6) + 1$

Summary

For a full *m*-ary tree:

(i) Given n vertices, $I = (n - 1) / m$ internal vertices and

$L = n - I = [(m - 1) \times n + 1] / m$ leaves.

(ii) Given I internal vertices, $n = m \times I + 1$ vertices and

$L = n - I = (m - 1) \times I + 1$ leaves.

(iii) Given L leaves, $n = (m \times L - 1) / (m - 1)$ vertices and

$I = n - L = (L - 1) / (m - 1)$ internal vertices.

In the previous example:

$m = 2$, $n = 13$, $I = 6$ and $L = 7$

(i) $I = (13 - 1) / 2 = \mathbf{6}$ and $L = 13 - 6 = \mathbf{7}$

(ii) $n = 2 \times 6 + 1 = \mathbf{13}$ and $L = 13 - 6 = \mathbf{7}$

(iii) $n = (2 \times 7 - 1) / (2 - 1) = \mathbf{13}$ and $I = 13 - 7 = \mathbf{6}$

Properties of Trees

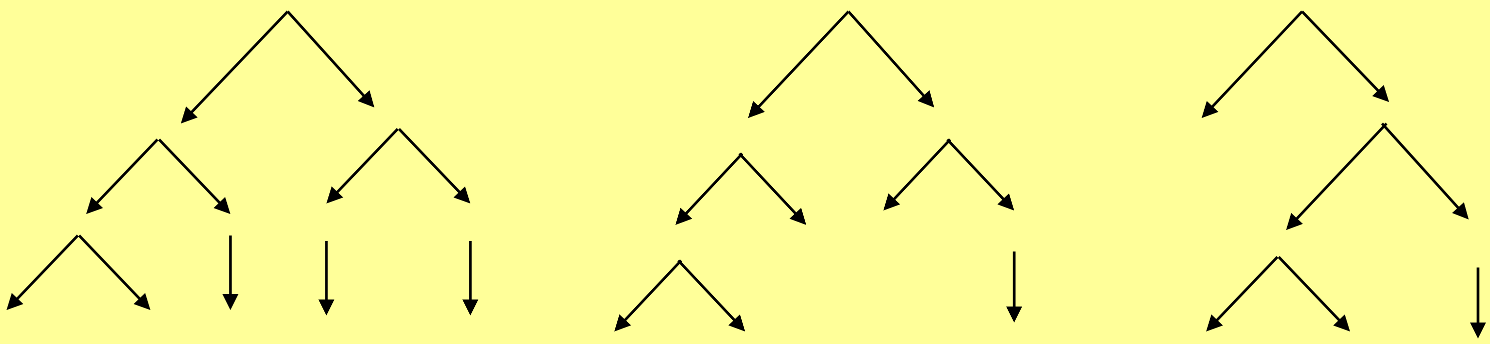
3- The **level** of a vertex in a rooted tree is the length of the path from the root to the vertex (level of the root is 0)

4- The **height** of the rooted tree is the maximum of the levels of vertices (length of the longest path from the root to any vertex)

Balanced Trees

- **Balanced Tree**

A rooted ***m*-ary** tree of height ***h*** is balanced if all **leaves** are at levels ***h*** or ***h* - 1**.



Balanced **Balanced** **Not Balanced**

7.2 Tree Traversal

- Traversal algorithms
  - Pre-order traversal
  - In-order traversal
  - Post-order traversal

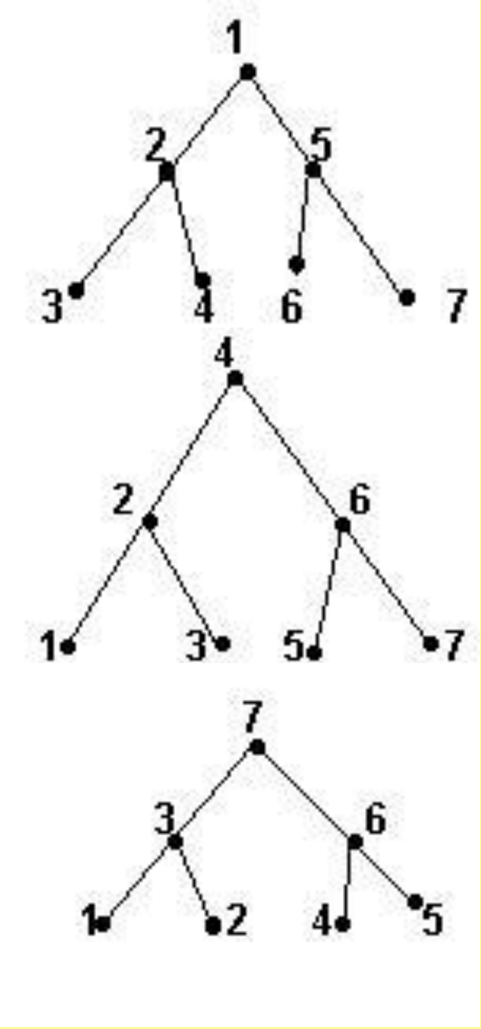
- Prefix / Infix / Postfix notation

Traversal Algorithms

Is visiting every vertex of ordered rooted tree.

- **Pre-order:** **R**oot, **L**eft, **R**ight.
- **In-order:** **L**eft, **R**oot, **R**ight.
- **Post-order:** **L**eft, **R**ight, **R**oot.

Tree Traversals



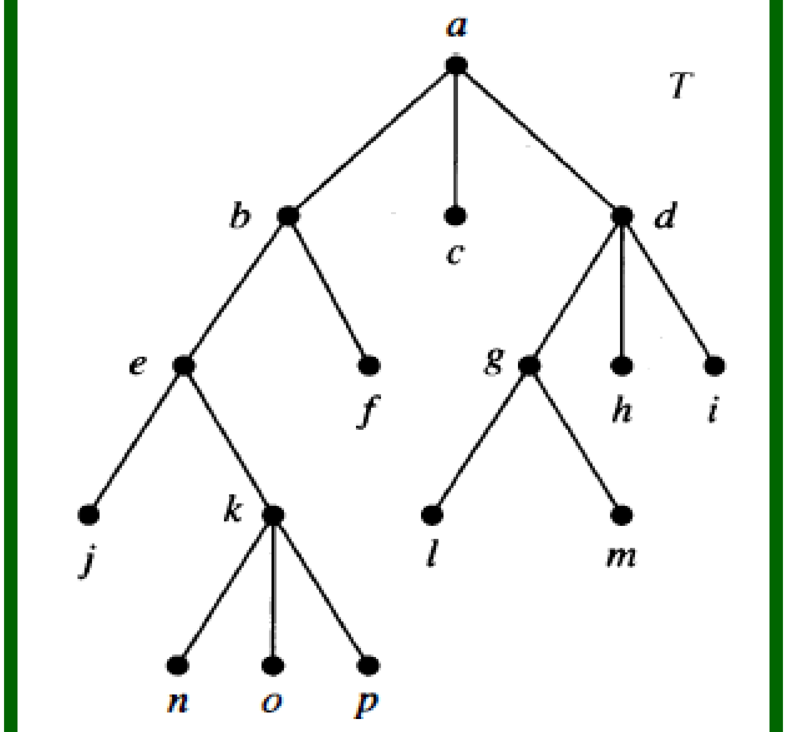
- Pre-order traversal
- In-order traversal
- Post-order traversal

Tree Traversals

Pre-order: **R**oot, **L**eft, **R**ight.

In-order: **L**eft, **R**oot, **R**ight.

Post-order: **L**eft, **R**ight, **R**oot.



Pre-order: a b e j k n o p f c d g l m h i

In-order: j e n k o p b f a c l g m d h i

Post-order: j n o p k e f b c l m g h i d a

Infix, Prefix, and Postfix Notation

- The way to write arithmetic expression is known as a **notation**. An arithmetic expression can be written in three different but equivalent notations

These notations are

- Infix Notation
- Prefix (Polish) Notation
- Postfix (Reverse-Polish) Notation

Infix, Prefix, and Postfix Notation

- Infix Notation:

We write expression in **infix** notation

e.g. a - b + c

- where operators are used **in**-between operands. It is easy for us humans to read, write, and speak in infix notation but the same does not go well with computing devices. An algorithm to process infix notation could be difficult and costly in terms of time and space consumption.

Infix, Prefix, and Postfix Notation

- Prefix Notation :

In this notation, operator is **prefix**ed to operands, i.e. operator is written ahead of operands.

- For example, **+ab**. This is equivalent to its infix notation **a + b**.
- Prefix notation is also known as **Polish Notation**.

Infix, Prefix, and Postfix Notation

- Postfix Notation:
  This notation style is known as **Reversed Polish Notation**.
- In this notation style, the operator is **postfix**ed to the operands i.e., the operator is written after the operands.

For example, **ab+**. This is equivalent to its infix notation **a + b**.

Infix, Prefix, and Postfix Notation

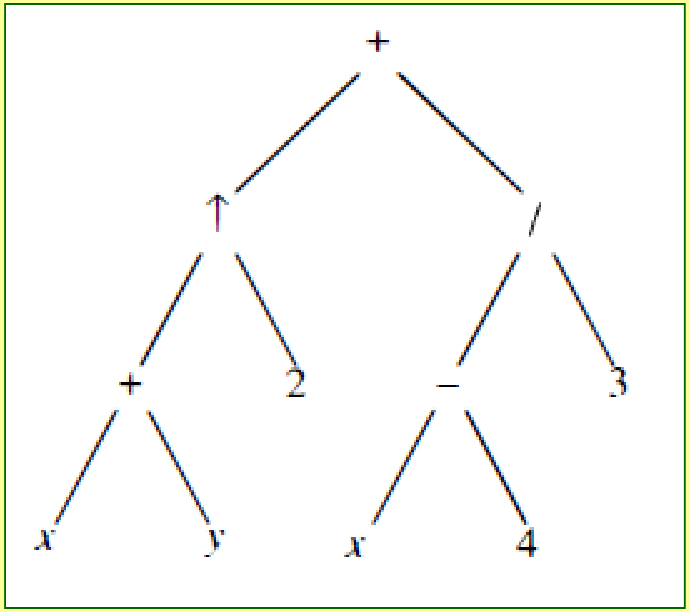
The following table briefly tries to show the difference in all three notations −

Sr.No.	Infix Notation	Prefix Notation	Postfix Notation
1	a + b	+ a b	a b +
2	(a + b) ∗ c	∗ + a b c	a b + c ∗
3	a ∗ (b + c)	∗ a + b c	a b c + ∗
4	a / b + c / d	+ / a b / c d	a b / c d / +
5	(a + b) ∗ (c + d)	∗ + a b + c d	a b + c d + ∗
6	((a + b) ∗ c) - d	- ∗ + a b c d	a b + c ∗ d -

Infix, Prefix, and Postfix Notation

A tree can be used to represents mathematical expressions.

Example: $((x + y)\hat{}2) + ((x - 4) / 3)$



Prefix: $+ \ \hat{} \ + x \ y \ 2 \ / - \ x \ 4 \ 3$

Postfix: $x \ y \ + 2 \ \hat{} \ x \ 4 \ - 3 \ / +$

Infix, Prefix, and Postfix Notation

Infix: In-order traversal of tree must be fully parenthesized to remove ambiguity.

Example: $x + 5 / 3$: $(x + 5) / 3$, $x + (5 / 3)$

Prefix (polish): Pre-order traversal of tree (no parenthesis needed)

Example: From the above tree → $+ * + x\ y\ 2 / - x\ 4\ 3$

Postfix: Post-order traversal (no parenthesis needed)

Example: From the above tree → $x\ y + 2 * x\ 4 - 3 / +$

1. Evaluating a **Prefix** Expression: (Pre-order: **Right to left**)

$$+ \quad - \quad * \quad 2 \quad 3 \quad 5 \quad / \quad \underbrace{\uparrow \quad 2 \quad 3}_{2 \uparrow 3 = 8} \quad 4$$

$$+ \quad - \quad * \quad 2 \quad 3 \quad 5 \quad \underbrace{/ \quad 8 \quad 4}_{8 / 4 = 2}$$

$$+ \quad - \quad \underbrace{* \quad 2 \quad 3}_{2 \quad 3 = 6} \quad 5 \quad 2$$

$$+ \quad \underbrace{- \quad 6 \quad 5}_{6 - 5 = 1} \quad 2$$

$$\underbrace{+ \quad 1 \quad 2}_{1 + 2 = 3}$$

Value of expression 3

2. Evaluating a **Postfix** Expression: (**Post-order**: **Left to right**)

$$7\quad \underbrace{2\quad 3\quad *}_{2 * 3 = 6}\quad -\quad 4\quad \uparrow\quad 9\quad 3\quad /\quad +$$

$$\underbrace{7\quad 6\quad -}_{7 - 6 = 1}\quad 4\quad \uparrow\quad 9\quad 3\quad /\quad +$$

$$\underbrace{1\quad 4\quad \uparrow}_{1^4 = 1}\quad 9\quad 3\quad /\quad +$$

$$1\quad \underbrace{9\quad 3\quad /}_{9 / 3 = 3}\quad +$$

$$\underbrace{1\quad 3\quad +}_{1 + 3 = 4}$$

Value of expression: 4

Infix, Prefix, and Postfix Notation

- **Exercise**

Draw the tree for the following expression and find the infix, prefix, and postfix

$\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$