Introduction to Computers and Programming

Prof. I. K. Lundqvist

Lecture 7
April 2 2004

Concept Question

A graph G(V, E) is a finite nonempty set of vertices and a set of edges

G1(V1,E1) where V1 = {}, E1 = {}

G2(V2,E2) where V2 = {a,b}, E2 = {}

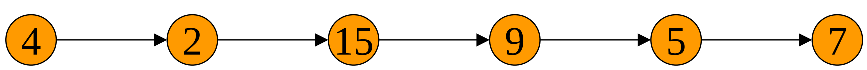
1. Both G1 and G2 are Graphs
2. Only G1 is a Graph
3. Only G2 is a Graph
4. Neither G1 nor G2 are Graphs

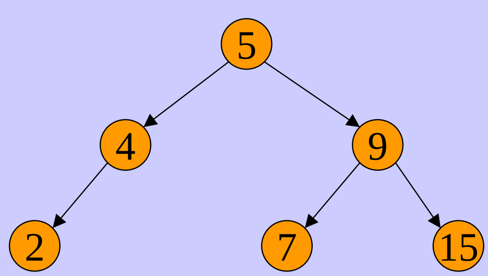
[Theorem]

Importance

- **Theorem**: a mathematical statement that can be shown to be true
  - Can be proved using other theorems, axioms (statements which are given to be true) and rules of inference
- **Lemma**: a pre-theorem or result needed to prove a theorem
- **Corollary**: post-theorem or result which follows directly from a theorem
- Proposition
- Claim
- Remark

Why should we use trees?





Binary Search Tree

Trees

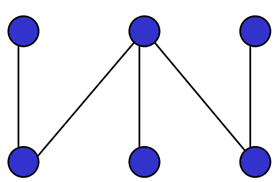
A **tree** is a connected undirected graph with no simple circuits.

– it cannot contain multiple edges or loops

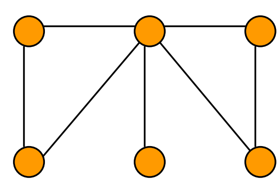
Theorem : An undirected graph is a tree if and only if there is a **unique simple** path between any two of its vertices.

Which graphs are trees?

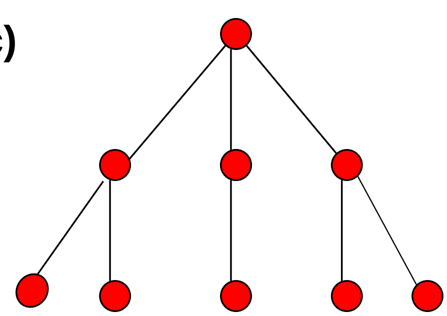
a)



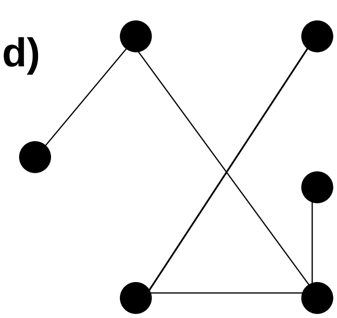
b)



c)



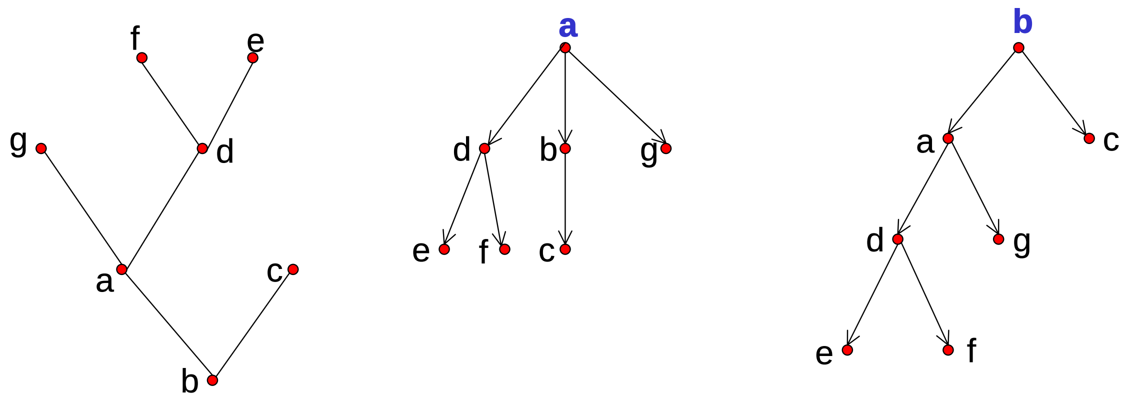
d)



Rooted Tree

- A directed graph G is called a **rooted tree** if there exists a vertex *u* so that for each *v*∈V, there is exactly one path between *u* and *v*
  - The in-degree of *u* is 0 and the in-degree of all other vertices is 1
- For an undirected graph, different choices of the root produces different trees

Choice of Root



Examples of Rooted Trees?

Internal Vertex

- A vertex that has children is called an **internal vertex**

- A graph H(W, F) is a **subgraph** of a graph G(V,E) iff $W \subseteq V$ and $F \subseteq$ E

- The **subtree** at vertex v is the subgraph of the tree consisting of vertex v and its descendants and all edges incident to those descendants

Tree Properties

- The **parent** of a non-root vertex v is the unique vertex u with a directed edge from u to v.

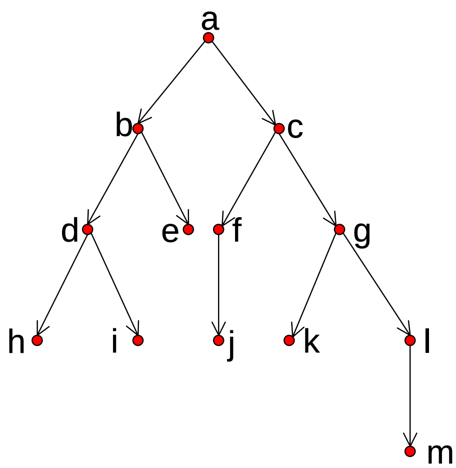
- A vertex is called a **leaf** if it has no children.

- The **ancestors** of a non-root vertex are all the vertices in the path from root to this vertex.

- The **descendants** of vertex v are all the vertices that have v as an ancestor.

Tree Properties

- The **level** of vertex *v* in a rooted tree is the length of the unique path from the root to *v*.
- The **height** of a rooted tree is the maximum of the levels of its vertices.



Level of vertex **f** = 2
Height of tree = 4

Binary Tree

- An **m-ary tree** is a rooted tree in which each internal vertex has *at most* m children
- A rooted tree is called a **binary tree** if every internal vertex has *no more than* 2 children.
- The tree is called a **full** binary tree if every internal vertex has exactly 2 children.

Tree Properties

Theorem: A tree with N vertices has N-1 edges.

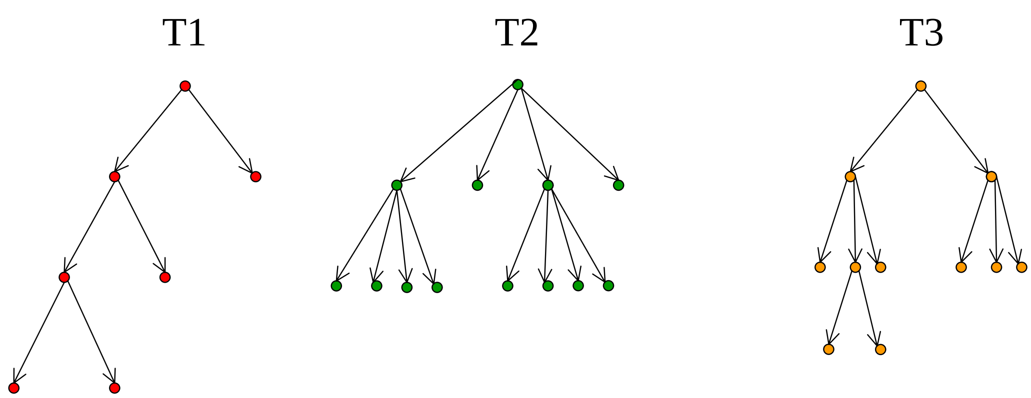
Theorem: There are at most 2^H leaves in a binary tree of height H.

Corallary: If a binary tree with L leaves is full and balanced, then its height is

$$H = \lceil \log_2 L \rceil$$

A **balanced** tree with height *h* is a m-ary tree with all leaves being at levels *h* or *h-1*

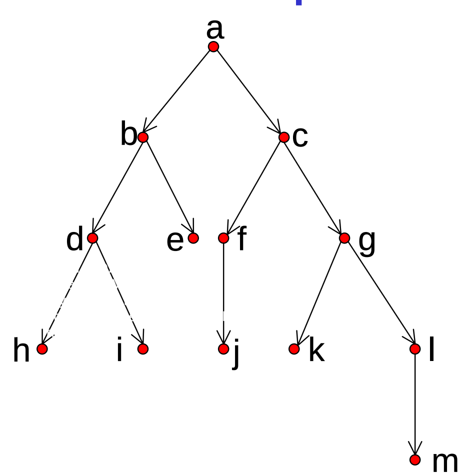
Examples



Ordered Binary Tree

- An **ordered** rooted tree is a rooted tree where the children of each internal vertex are ordered.
- In an ordered binary tree, the two possible children of a vertex are called the **left** child and the **right** child, if they exist.

Example



Children of **b**? d, e
Parent of **b**? a
Ancestors of **g**? c, a
Descendants of **b**? d, e, h, i

Leafs? h, i, e, j, k, m
Internal vertices? a, b, c, d, f, g
Left child of **g**? k
Right child of **g**? l

Traversal Algorithms

- A **traversal algorithm** is a procedure for systematically visiting every vertex of an ordered binary tree
- Tree traversals are defined recursively
- Three commonly used traversals are:
  - **preorder**
  - **inorder**
  - **postorder**

PREORDER Traversal Algorithm

Let T be an ordered binary tree with root R

If T has only R then

R is the preorder traversal

Else

Let T_1, T_2 be the left and right subtrees at R

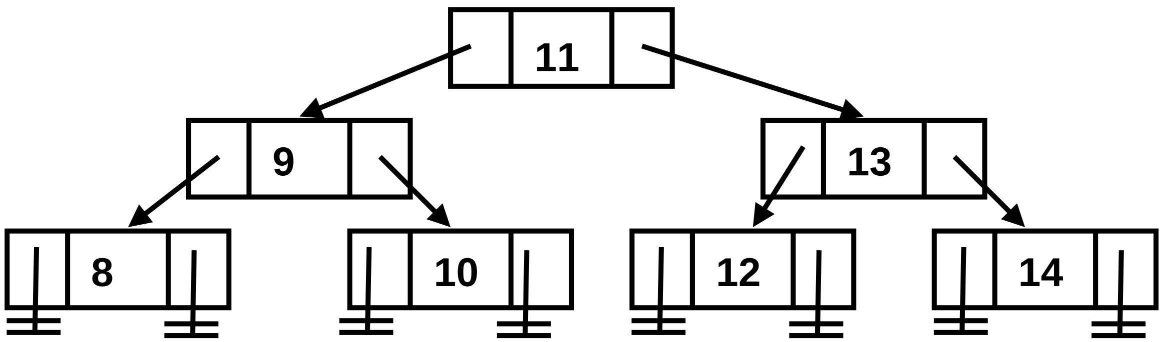
Visit R

Traverse T_1 in preorder

Traverse T_2 in preorder

Record Definition

```
type Node;
type Nodeptr is access Node;
type Node is record
  Element      : Elementtype;
  Left_Child   : Nodeptr;
  Right_Child  : Nodeptr;
end record;
```



INORDER Traversal Algorithm

Let T be an ordered binary tree with root R

If T has only R then

R is the inorder traversal

Else

Let T_1, T_2 be the left and right subtrees at R

Traverse T_1 in inorder

Visit R

Traverse T_2 in inorder

POSTORDER Traversal Algorithm

Let T be an ordered binary tree with root R

If T has only R then

R is the postorder traversal

Else

Let T_1, T_2 be the left and right subtrees at R

Traverse T_1 in postorder

Traverse T_2 in postorder

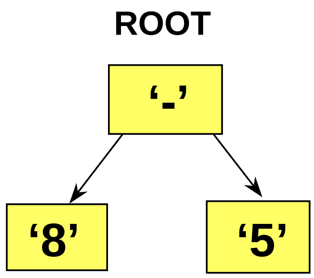
Visit R

Binary Expression Tree

A special kind of binary tree in which:

- Each leaf node contains a single operand
- Each inner vertex contains a single binary operator
- The left and right subtrees of an operator node represent sub-expressions that must be evaluated before applying the operator at the root of the subtree.

Binary Expression Tree

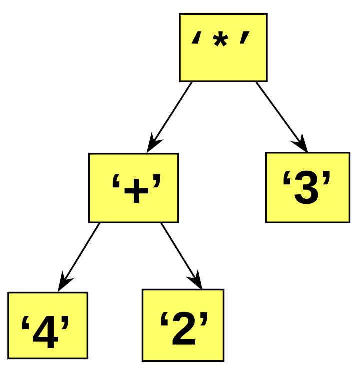


INORDER TRAVERSAL: 8 - 5 has value 3

PREORDER TRAVERSAL: - 8 5

POSTORDER TRAVERSAL: 8 5 -

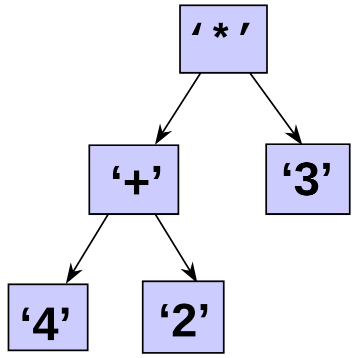
Binary Expression Tree



What value does it have?

(4 + 2) * 3 = 18

Binary Expression Tree

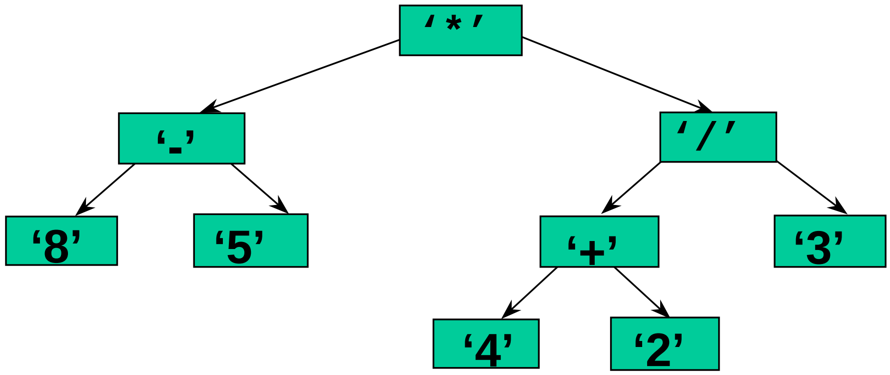


Infix:	**((4 + 2) * 3)**
Prefix:	*** + 4 2 3**
Postfix:	**4 2 + 3 ***

Levels Indicate Precedence

- When a binary expression tree is used to represent an expression, the levels of the nodes in the tree indicate their relative precedence of evaluation.
- Operations at higher levels of the tree are evaluated later than those below them. The operation at the root is always the last operation performed.

Binary Expression Tree

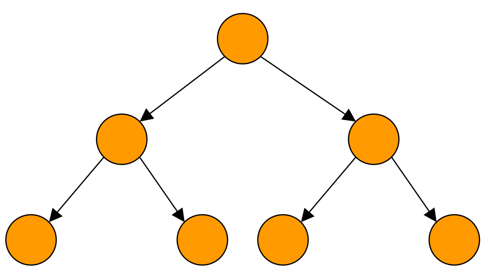


Infix: ((8 - 5) * ((4 + 2) / 3))

Prefix: * - 8 5 / + 4 2 3

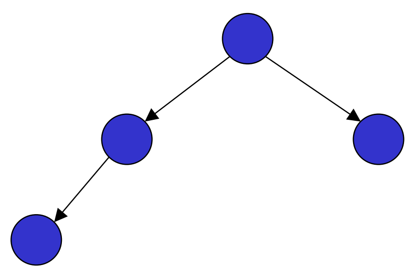
Postfix: 8 5 - 4 2 + 3 / *

Trees - Glossary

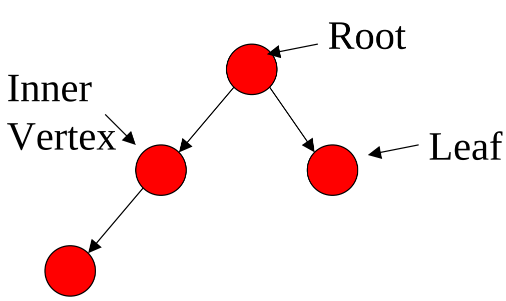


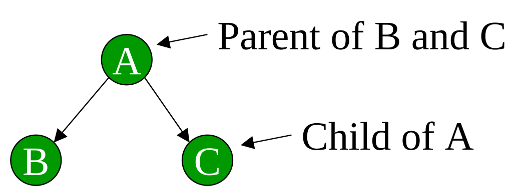
Perfectly balanced tree

M-ary tree



Height balanced tree





B and C are siblings