SCJ2013 Data Structure & Algorithms

Tree

Nor Bahiah Hj Ahmad

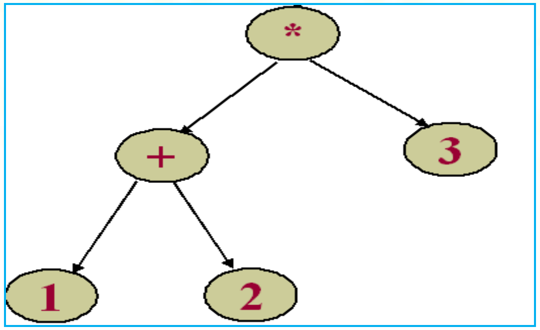
Course Objectives

At the end of the lesson students are expected to be able to:

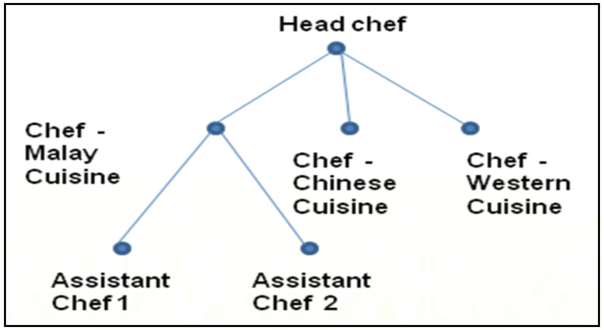
- Understand the tree concept and terms related to tree.
- Identify characteristics of general tree, binary tree and binary search tree
- Identify basic operations of a tree such as tree traversals, insert node, delete node, searching.
- Understand and know how to apply and implement tree in problem solving and in programming.

Introduction to tree - Definition

- Tree is a non-linear data structure.
- Data in a tree is stored in a hierarchy form.
- Example of tree application:
  - Represent algebraic formulas
  - Store data in hierarchy form. Ex: organization chart
  - Artificial intelligence – information is accessed based on certain decision which is stored in a tree.



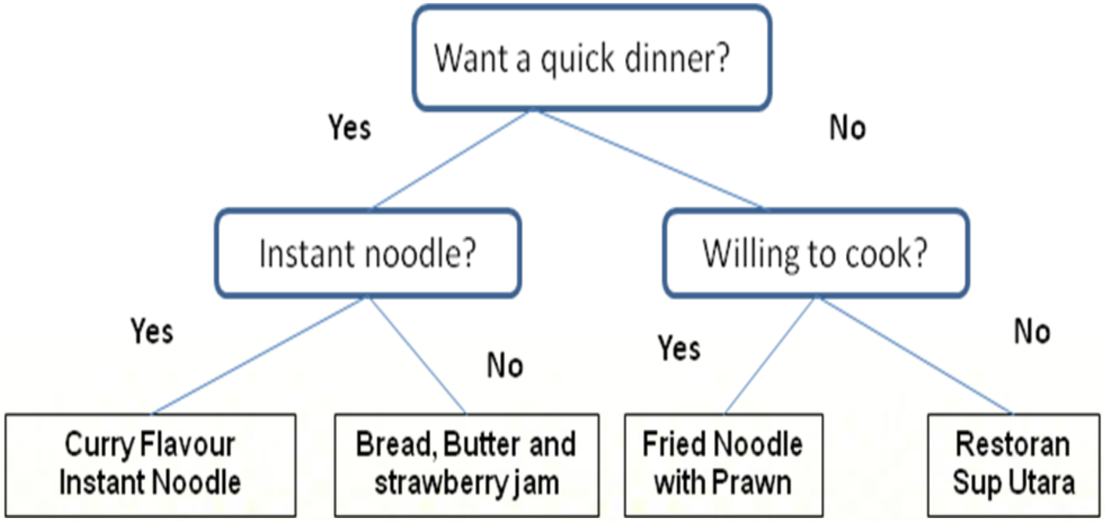
algebraic formulas : (1+2)*3



organization chart

Introduction to tree – Decision Tree

- Binary tree associated with a decision process
- Internal nodes: questions with yes/no answer
- External nodes: decisions
- Example: dining decision tree



Tree

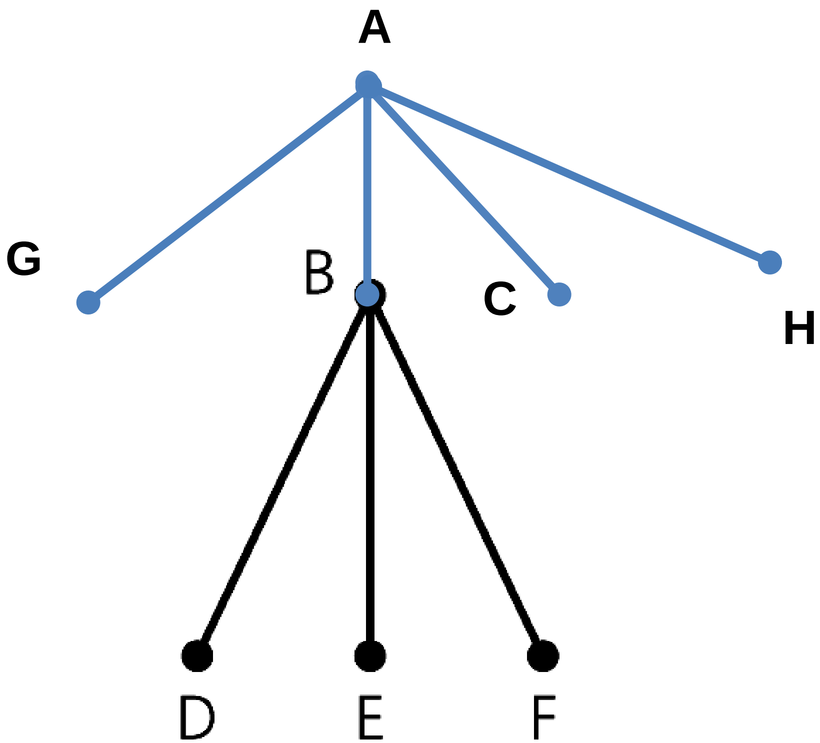
A tree is a collection of nodes and edges that connect the nodes.

- The collection can be empty.
- If not empty, a tree consists of a root, and zero or more nonempty subtrees.
- Any two vertices in a tree must have only one path between them or else its not a tree.
- Trees are hierarchical
  - Has parent-child relationship between two nodes.
  - Has ancestor-descendant relationships among nodes.

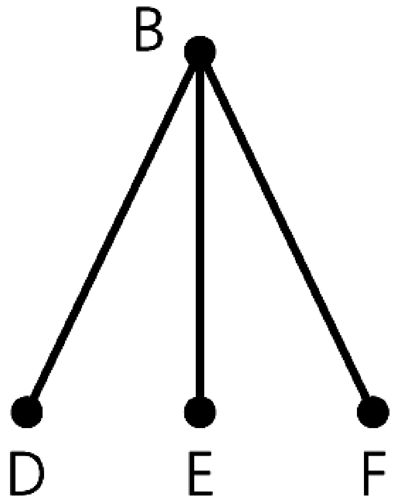
Tree terminology

- General tree
  - A general tree is a set of one or more nodes that is partitioned into :
    - The root
    - Sets that are general trees, called subtrees
  - Each node in general tree can have an unlimited children
- Subtree of a tree: Any node and its descendants

Tree Terminology



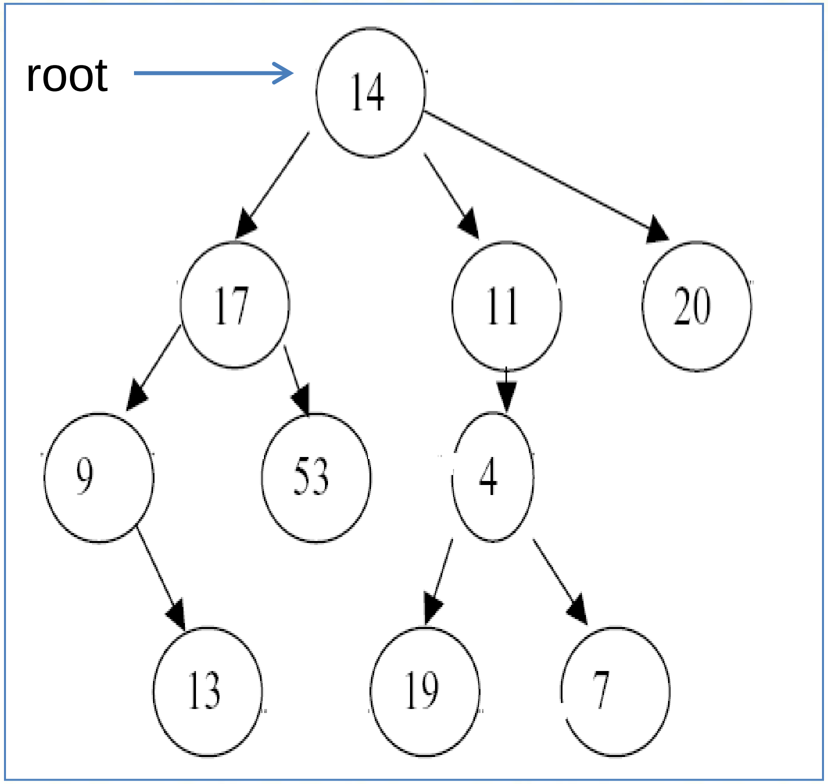
A general tree



A subtree of the tree in general tree

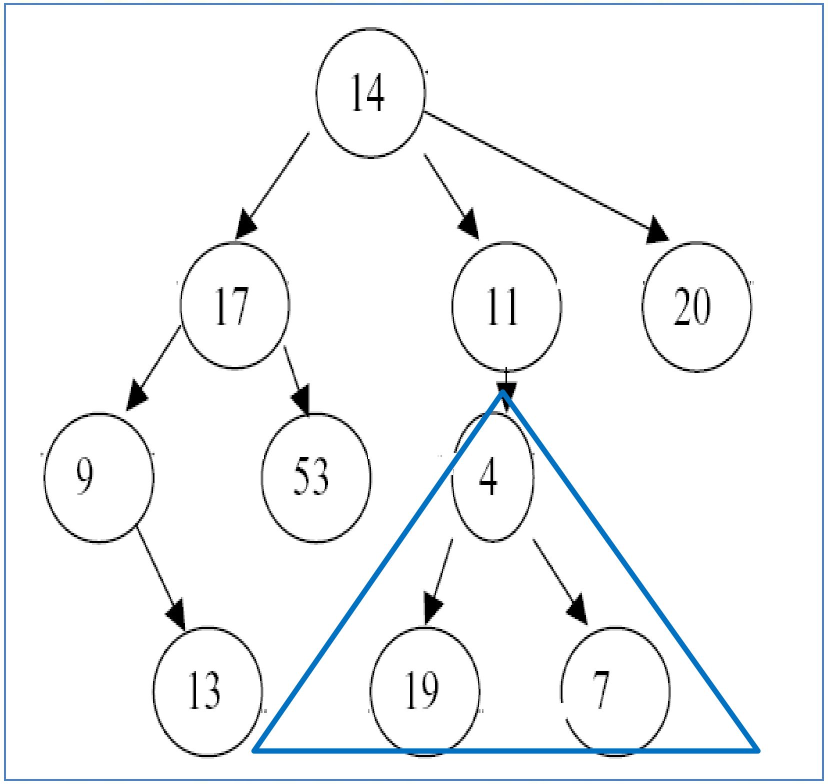
Tree Terminologies

- **Root**
  - The only node in the tree with no parent
  - *A tree has only one root*
  - *Root : 14*
- ***Child* and *parent***
  - Every node except the root has one parent
  - **Parent of node *n***
    - The node directly above node *n* in the tree
    - 14 is Parent to 17, 11, 20
  - A node can have an arbitrary number of children
  - **Child of node *n***
    - A node directly below node *n* in the tree
    - 17, 11, 20 are children of 14



Some Terminologies

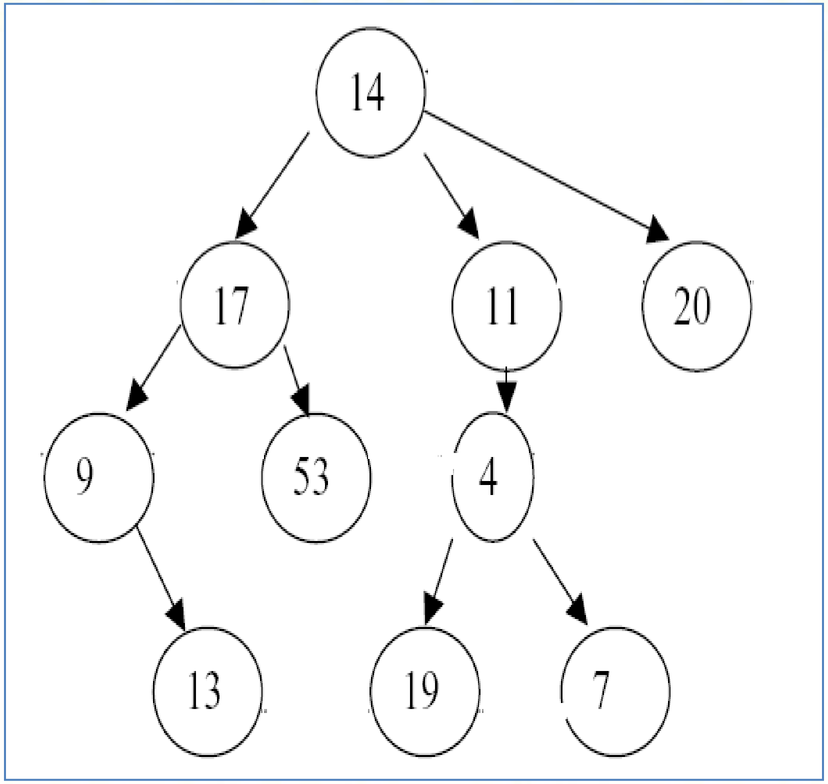
- ***Leaves***
  - Nodes with no children
  - 13, 53, 19,7, 20
- ***Sibling***
  - nodes with the same parent
  - 17, 11, 20 are siblings
  - 19 and 7 are siblings
- **Subtree of node *n***
  - A tree that consists of a child (if any) of node *n* and the child's descendants



Subtree for node 11

Some Terminologies

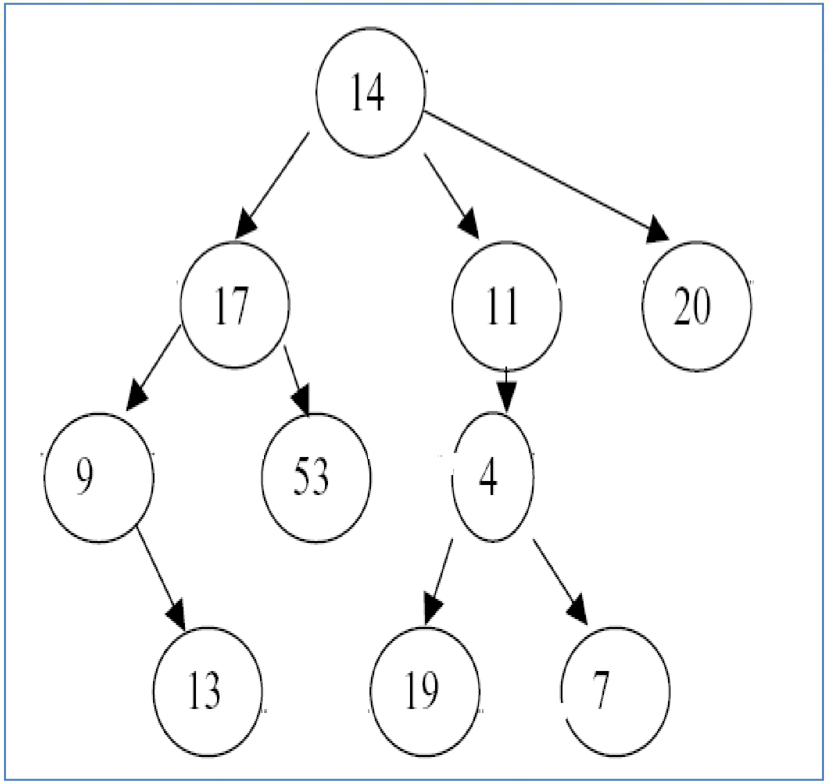
- **Ancestor of node *n***
  - A node on the path from the root to *n*
  - Ancesstor 13 : 9,17,14
  - Nod 14 is ancestor for all node in the tree
- **Descendant of node *n***
  - A node on a path from *n* to a leaf
  - Descendant 11: 4,19,7
  - All nodes in the tree are descendant to the root.



Some Terminologies

- ***Path*** *– sequence of nodes in which each node is adjacent o the next one. Example: Path from root to 13: 14,17,9,13*

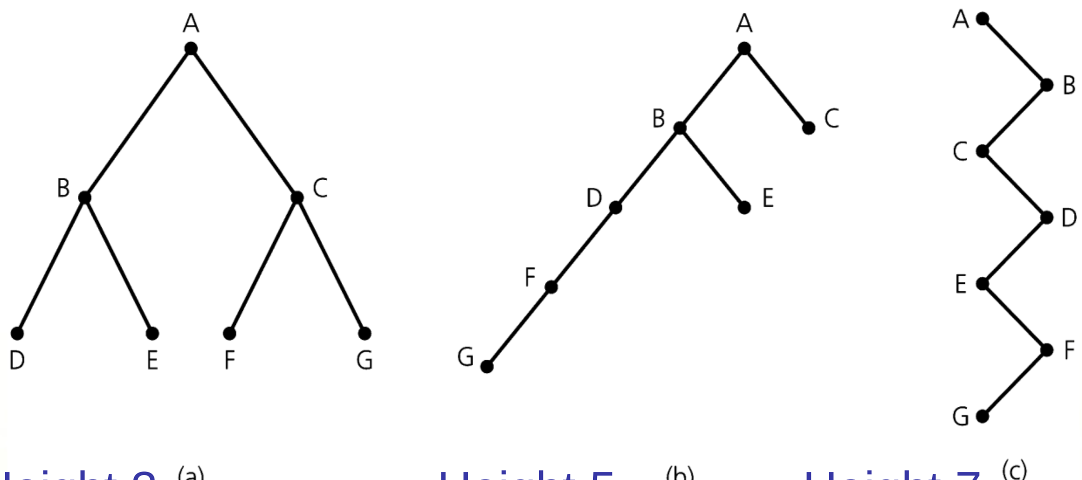
  Path from root to 19: 14,11,4,19
- ***Length***
  - number of edges on the path
  - Length of Tree : 3
- ***Depth* of a node**
  - length of the unique path from the root to that node
  - The depth of a tree is equal to the depth of the deepest leaf
  - Depth of Tree : 3
  - Depth of 4 : 2



The Height of Trees

Height of a tree

- Number of nodes along the longest path from the root to a leaf. *(Carrano,2007)*
- *If Tree is empty, its height is 0*



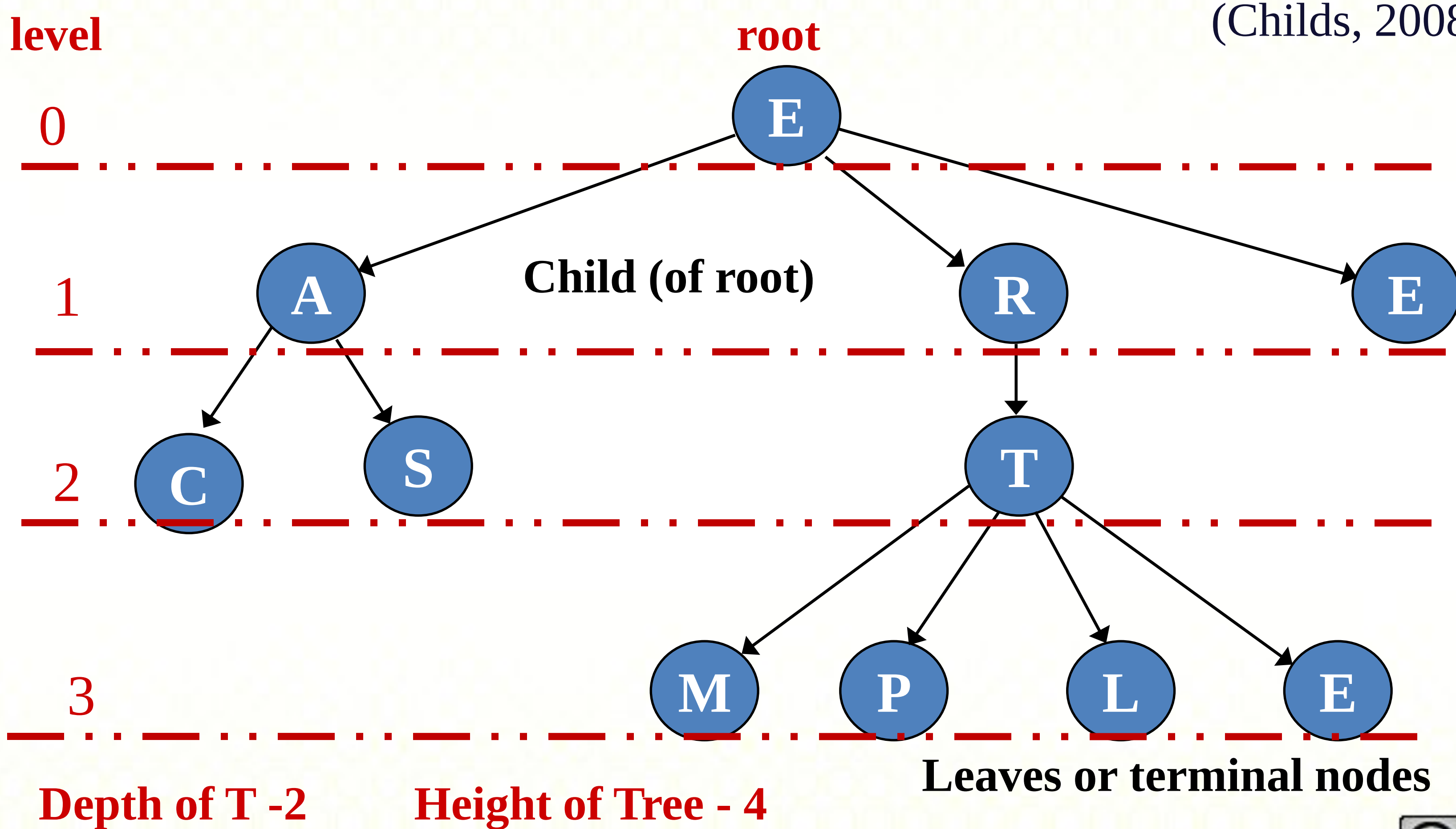
Height 3 Height 5 Height 7

Binary trees with the same nodes but different heights

Tree Terminologies

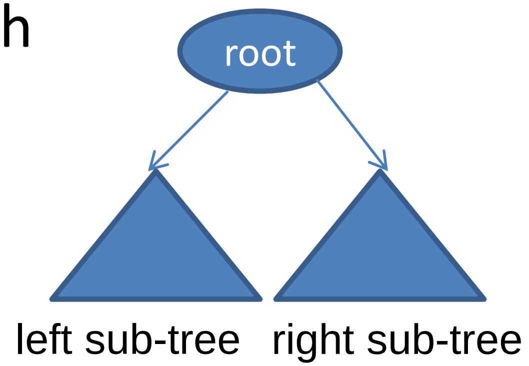
Level – the number of edges in the path from the root node to that node.

(Childs, 2008)

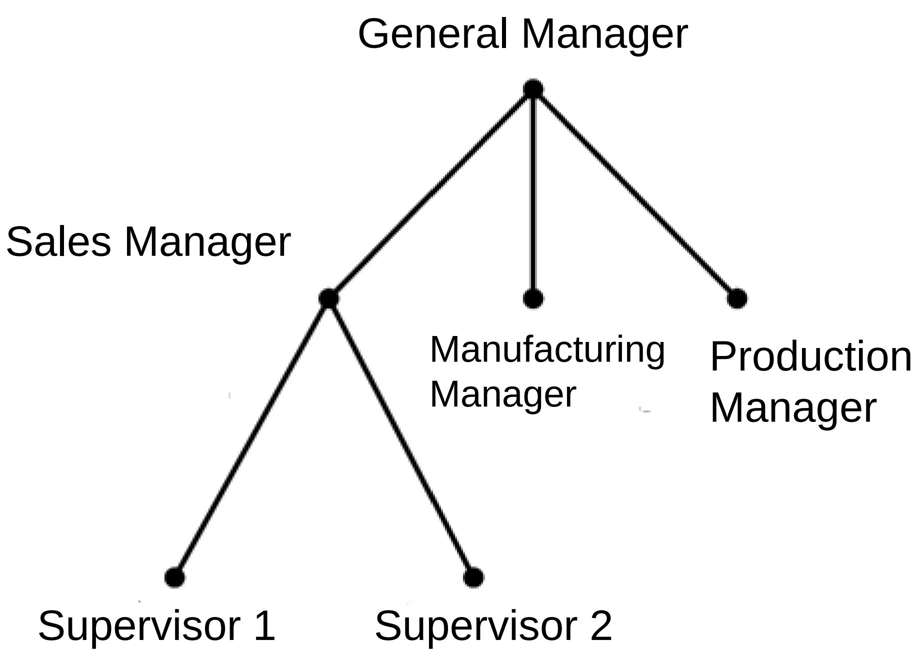


Binary Tree Definition

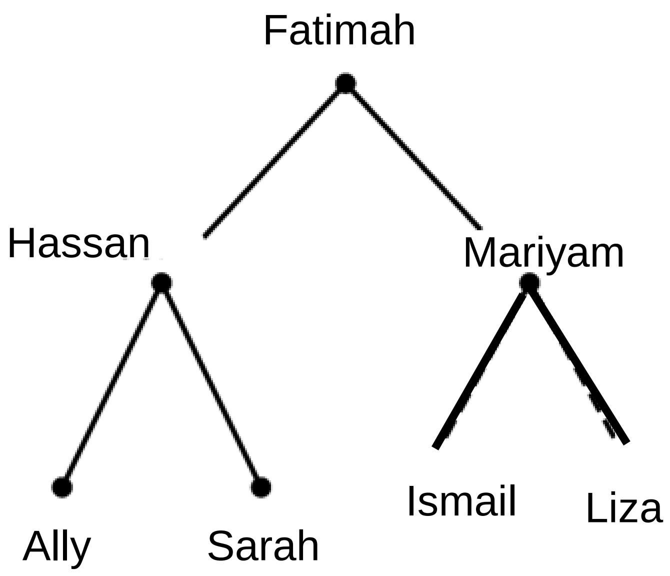
- A tree with restrictions, such that any given node can have at most two child nodes.
- A binary tree consists of a set of nodes such that either :
  - *Tree* is empty, or
  - *Tree* is partitioned into three disjoint subsets:
    - The root
    - Two possibly empty sets that are binary trees, called the left subtree of the *root* and the right subtree of the *root*



A General Tree vs A Binary Tree

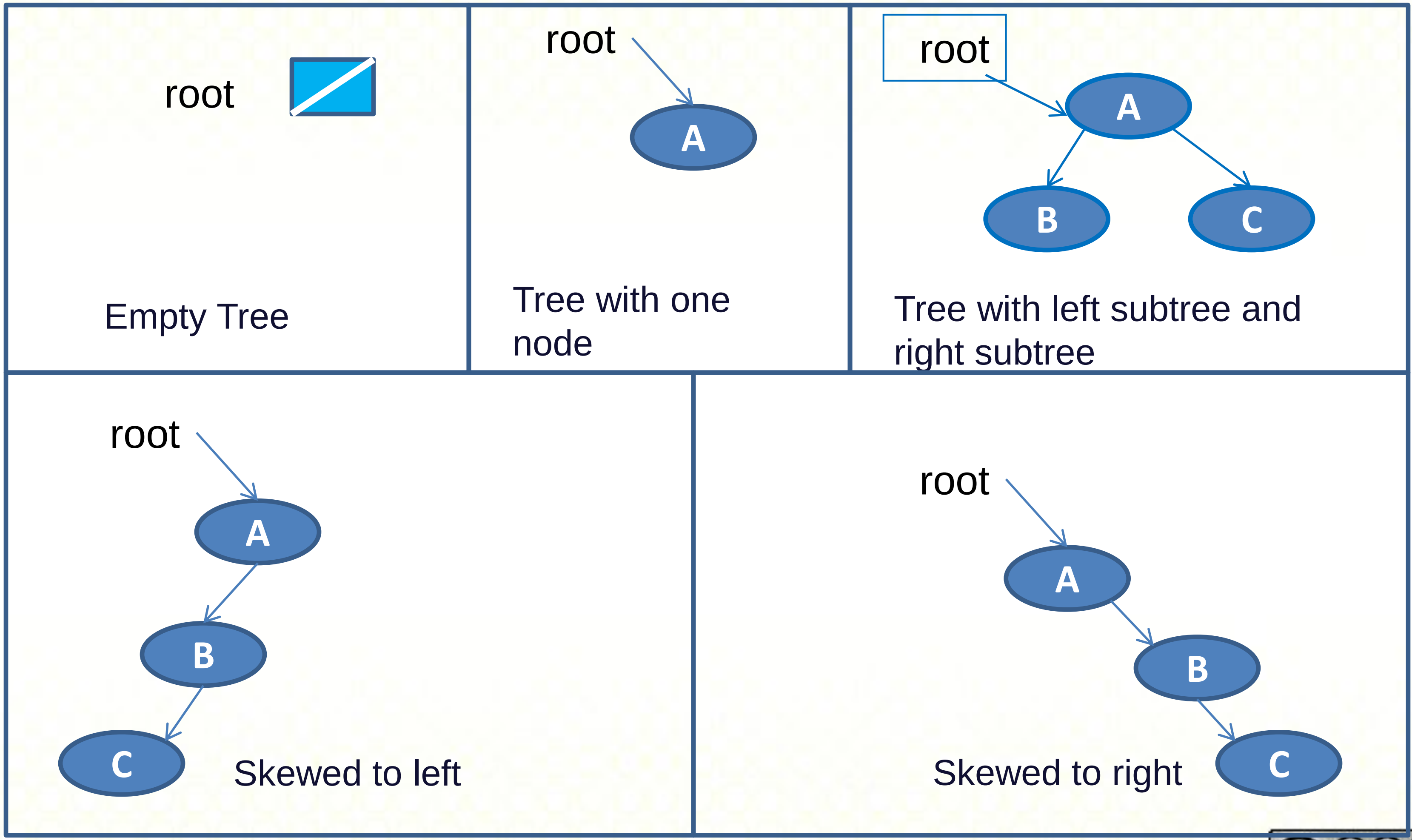


An organization chart

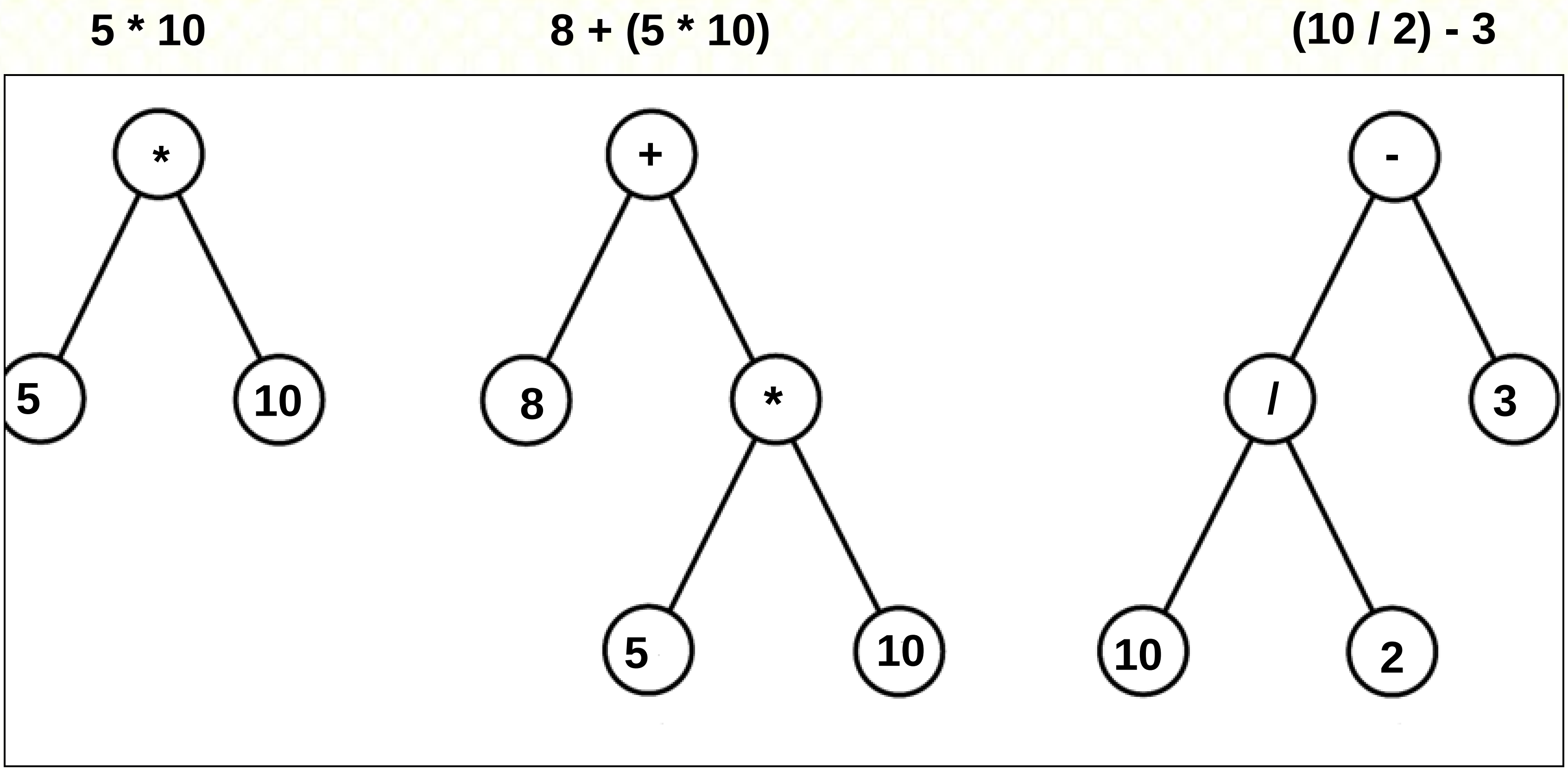


Family Tree

Collection of Binary Trees



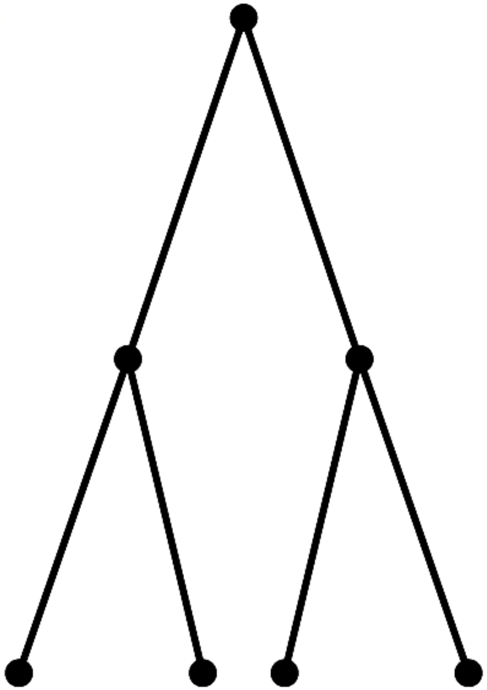
More Binary Trees



Binary trees that represent algebraic expressions.

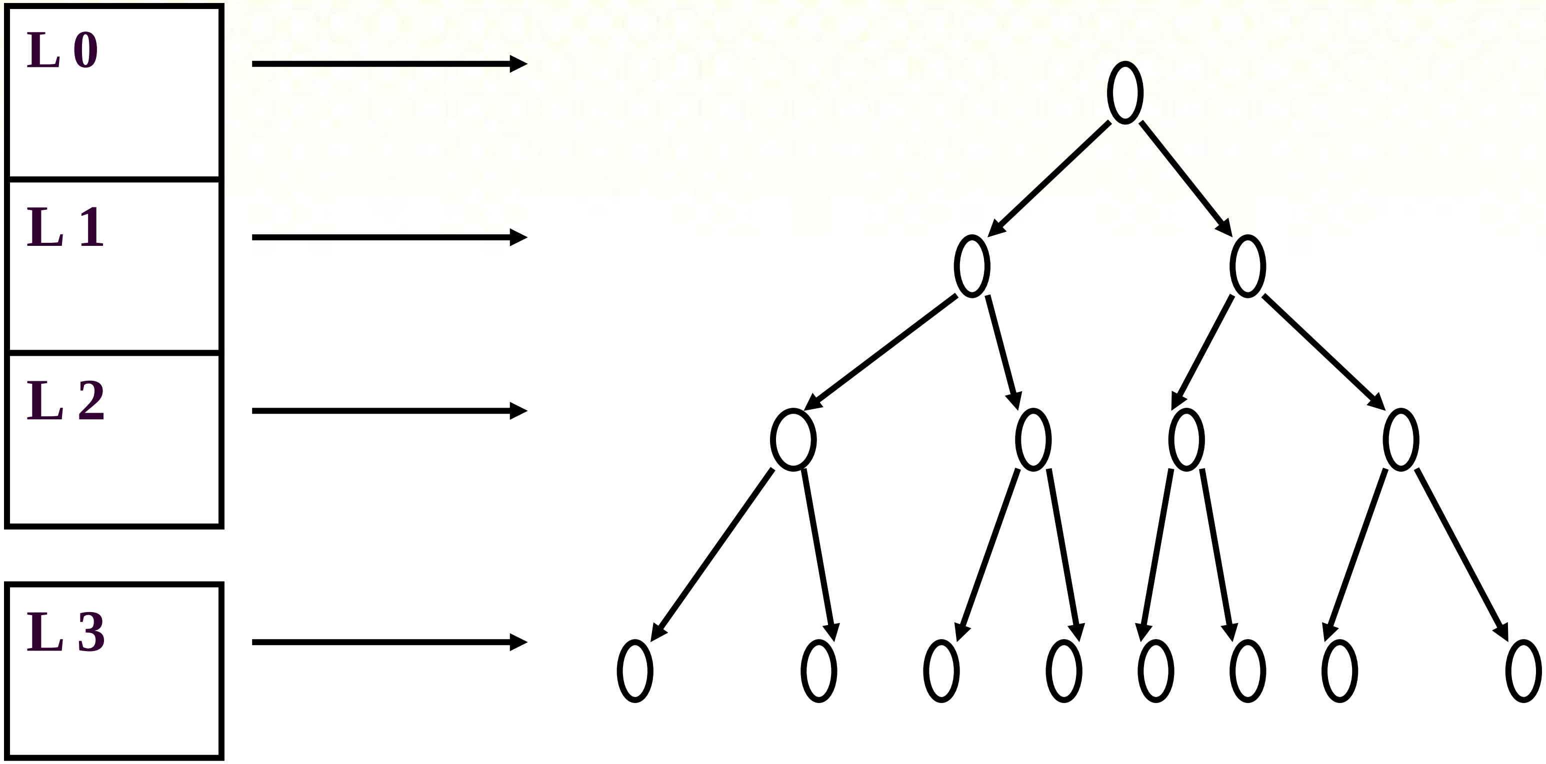
Full Binary Trees

- A binary tree of height h is *full* if
  - Nodes at levels < h have two children each
- Recursive definition
  - If T is empty, T is a full binary tree of height 0
  - If T is not empty and has height $h > 0$, T is a full binary tree if its root's subtrees are both full binary trees of height $h - 1$



A full binary tree of height 3

Full Binary Trees

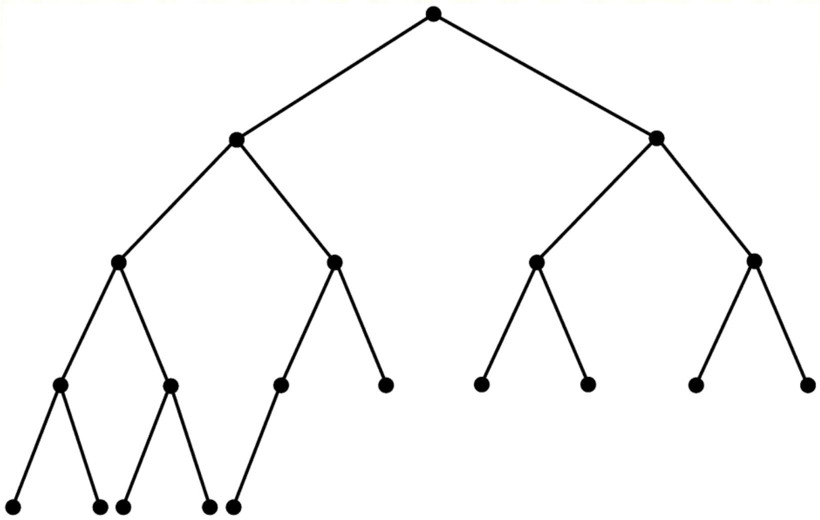


At each level the number of the nodes is doubled.

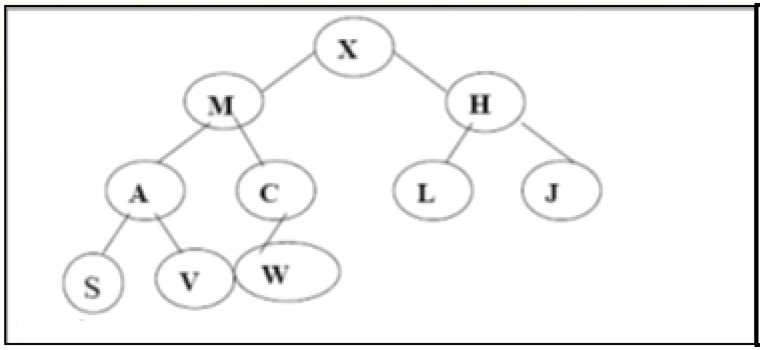
Total number of nodes: $\mathbf{1 + 2 + 2^2 + 2^3 = 2^4 - 1 = 15}$

Complete Binary Trees

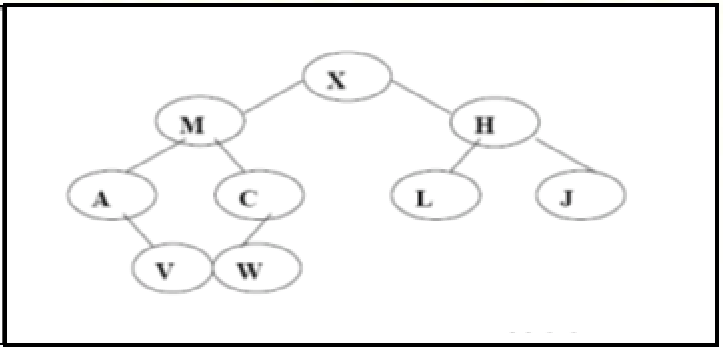
- A binary tree of height *h* is *complete* if
  - It is full to level *h*–1, and
  - Level *h* is filled from left to right



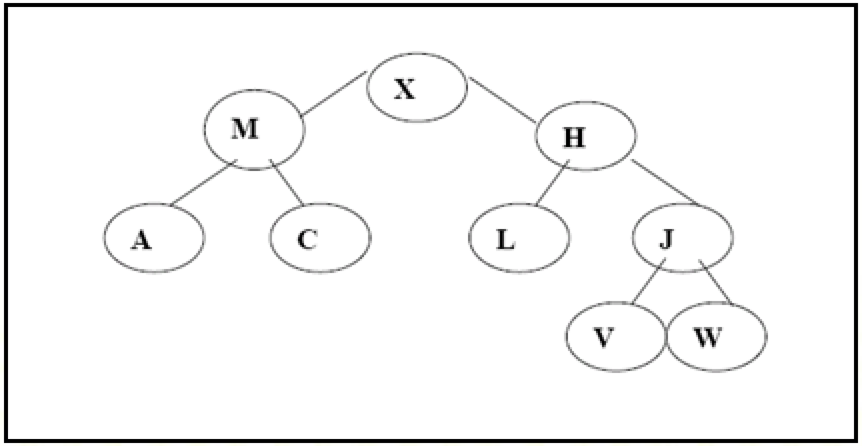
Tree Examples



Binary tree that is complete but not full



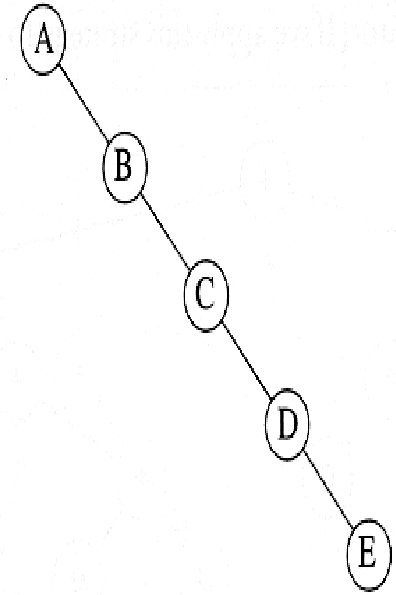
Binary tree that is not complete and not full



Binary tree that is not complete and not full

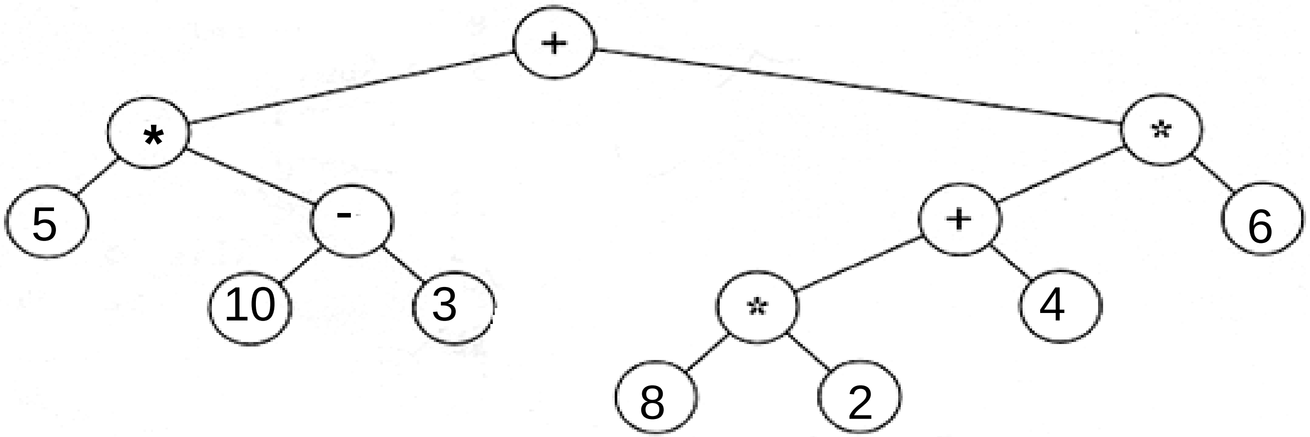
Balanced Binary Trees

- A binary tree is balanced if the heights of any node's two sub-trees differ by no more than 1
- Complete binary trees are balanced.
- Full binary trees are complete and balanced.
- The depth of an average binary tree is considerably smaller than n, even though in the worst case, the depth can be as large as n – 1.



Unbalanced tree : skewed to the right. Depth = n-1 (4)

Example: Expression Trees



Expression tree for ((5 * (10 – 3)) + ((8 * 2 + 4) * 6)

Expression Tree

- Leaves are operands (constants or variables)
- The other nodes (internal nodes) contain operators

Tree traversal

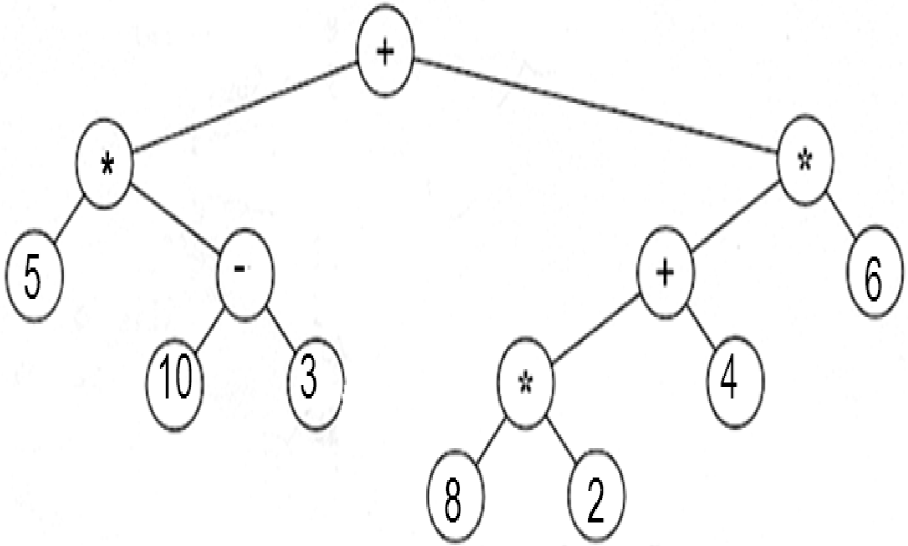
- Traverse a tree is to visit every node in a tree.
- Some operations can be done with the node during a visit.
  - For example, modify or update the data in the node
  - Used to print out the data in a tree in a certain order
- Type of Traversal
  - Inorder traversal
  - Preorder traversal
  - Postorder traversal

Pre-order traversal

Pre-order traversal

- Print the data at the root
- Recursively print out all data in the left subtree
- Recursively print out all data in the right subtree
- Give prefix expression

  +*5-103*+*8246

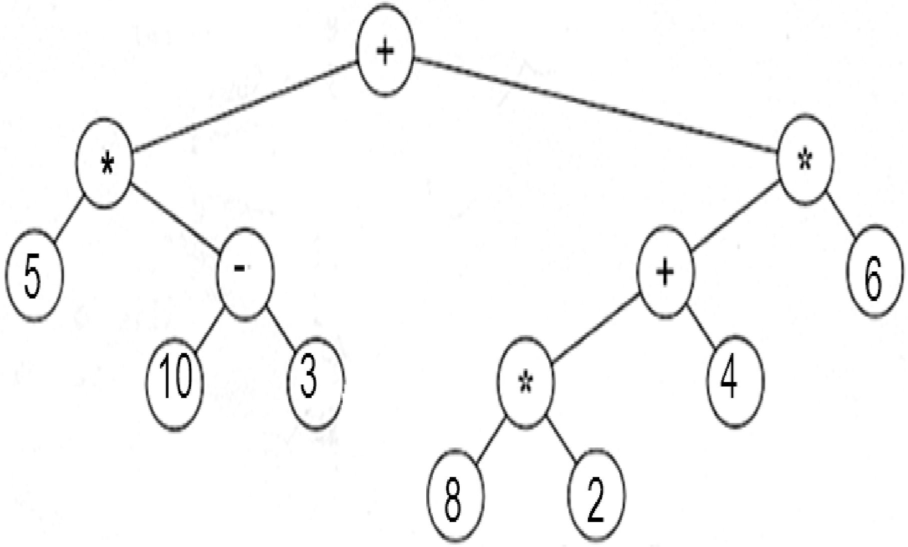


Expression tree for ((5 * (10 – 3)) + ((8 * 2 + 4) * 6)

Postorder traversal

Postorder traversal

- Recursively print out all data in the left subtree
- Recursively print out all data in the right subtree
- Print the data at the root
- Give postfix expression
  5103-*82*4+6*+



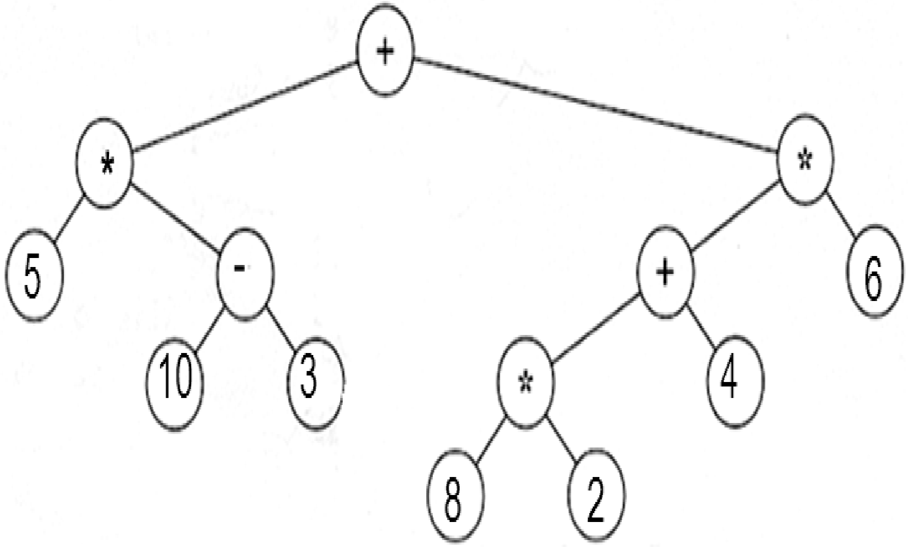
Expression tree for ((5 * (10 – 3)) + ((8 * 2 + 4) * 6)

Inorder traversal

Inorder traversal

- Recursively print out all data in the left subtree
- Print the data at the root
- Recursively print out all data in the right subtree
- Give infix expression

  5*10-3+8*2+4*6



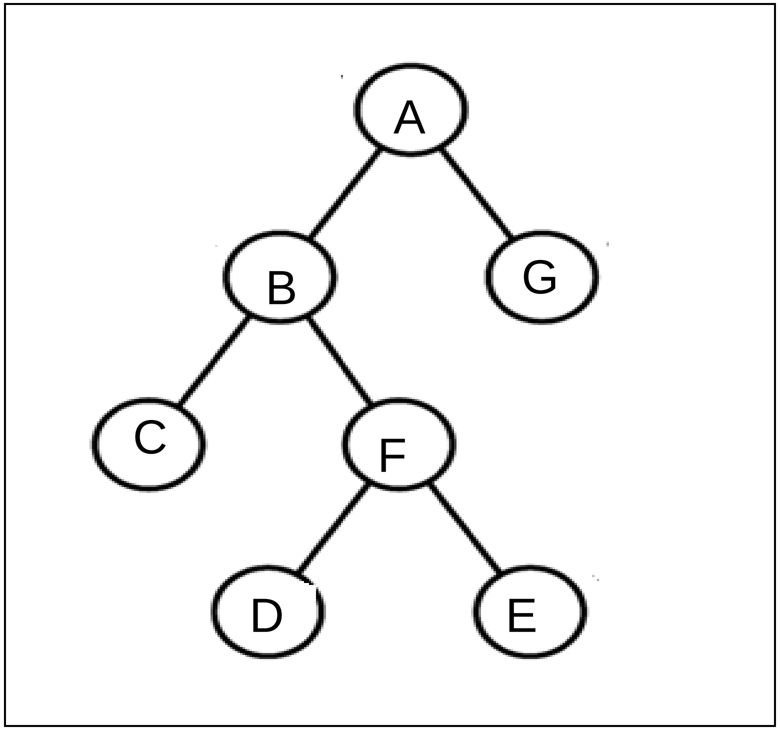
Expression tree for ((5 * (10 – 3)) + ((8 * 2 + 4) * 6)

Traversals of a Binary Tree

Pre-order traversal :
ABCFDEG

In-order traversal:
CBDFEAG

Post-order traversal:
CDEFBGA



Summary and Conclusion

- Tree provide a hierarchical organization of data with parent-child relationship.
- There are many types of tree such as general tree, binary tree and binary search tree.
- Terms related to tree : root, siblings, parent, leaf
- Traversing a tree is to visit every node in a tree either pre-order, in-order and post-order traversal.
- An in-order traversal of a binary search tree visits the tree's nodes in sorted search-key order

References

- Frank M. Carano, Janet J Prichard. "*Data Abstraction and problem solving with C++*" *Walls and Mirrors*. 5th edition (2007). Addision Wesley.
- Nor Bahiah et al. "*Struktur data & algoritma menggunakan C++". Penerbit UTM. 2005.*