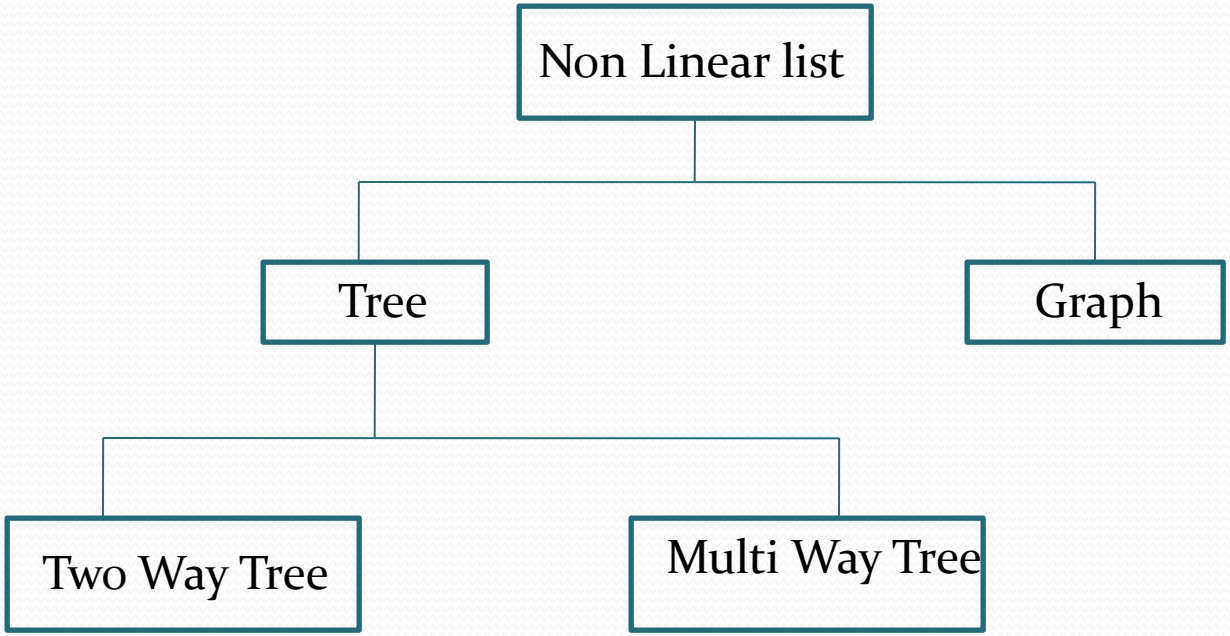
TREES

Topics Covered

- Basic Trees concepts and terminology
- Memory Representation of tree
- Binary Trees
- Traversing Binary trees
- Complete and extended binary tree
- General Trees

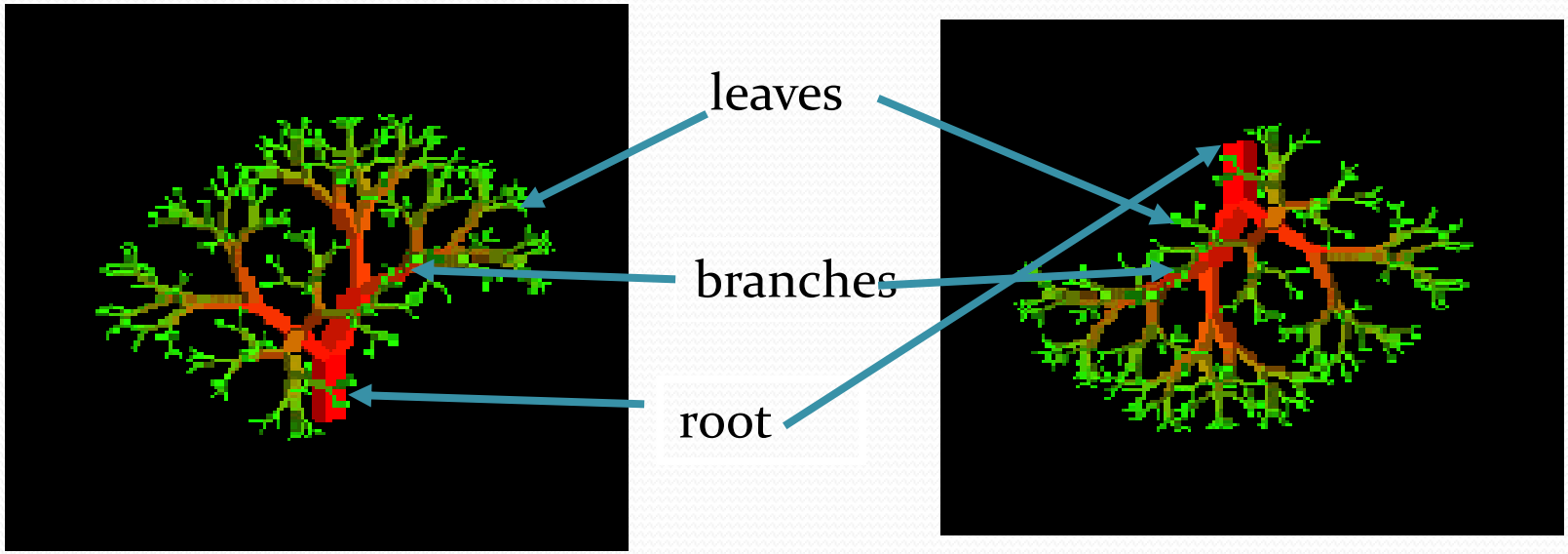
Non-linear List

- A non linear list in data structure is a special type of list having one or more successor
- Non-linear list divided into two categories



Tree

- A tree is a finite nonempty set of elements.
- It is an abstract model of a hierarchical structure.
- Trees are used extensively in computer science to represent algebraic formula, an efficient method for searching large , dynamic list and in file system

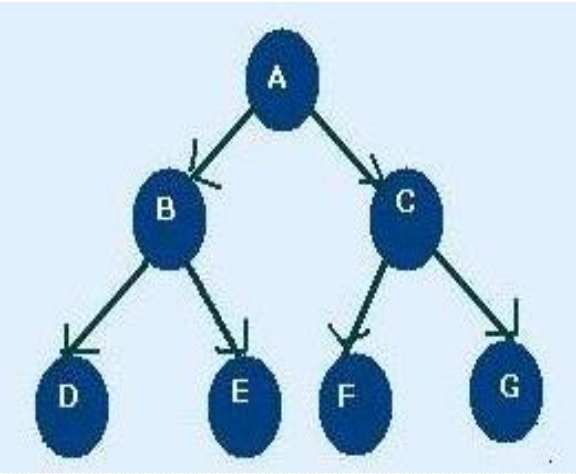


Nature View of a Tree

Computer Scientist's View

Tree Terminology

- **Node** : each of the elements or data that constructs tree. i.e. A, B, C, D, E, F
- **Edges:** Finite set of directed lines that connects node
- **Root:** The first Node of tree(If tree is not empty) i.e. A
- **In Degree :** The number of edges comes into particular node

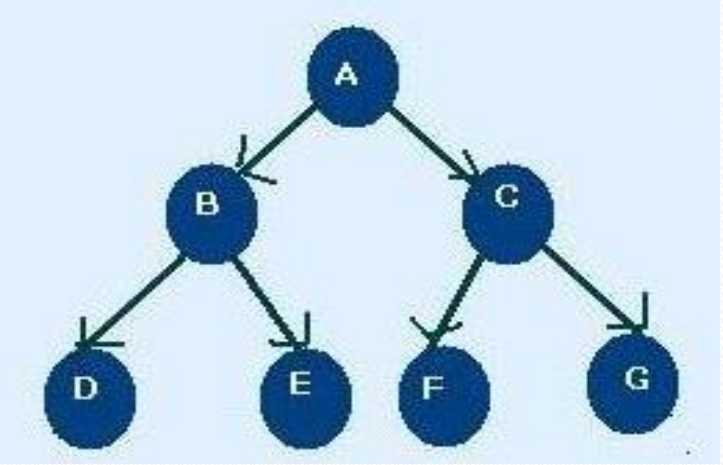


Node	In Degree
A	0
B	1
C	1
D	1
E	1
F	1
G	1

Tree Terminology

- **Out Degree :** The number of edges comes out particular node

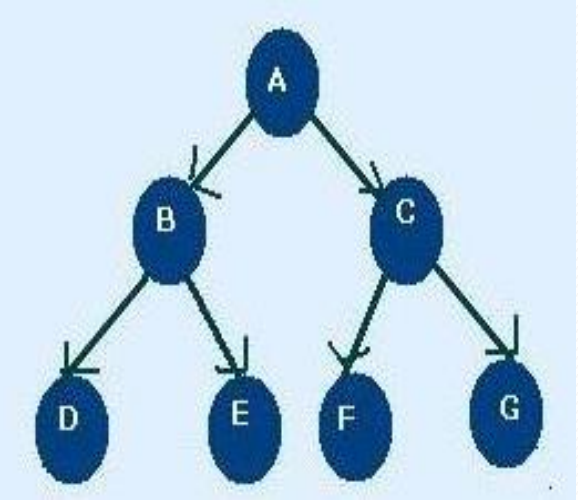
Node	Out Degree
A	2
B	2
C	2
D	0
E	0
F	0
G	0



Tree Terminology

- **Total Degree :** Summation of Indegree and outdegree

Node	Indegree	outdegree	degree
A	0	2	2
B	1	2	3
C	1	2	3
D	1	0	1
E	1	0	1
F	1	0	1
G	1	0	1

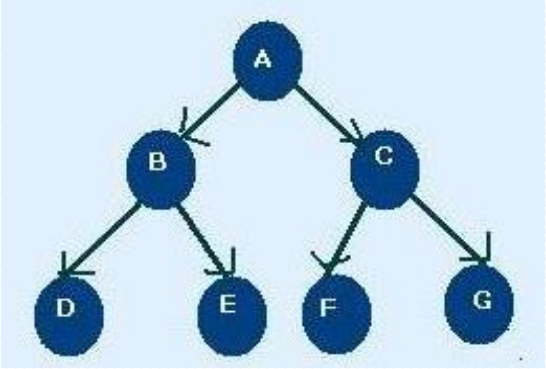


N.B: Each node except root node of tree must have an indegree of 1

Tree Terminology

- **Leaf** : Nodes that have outdegree 0 i.e. no successor. i.e. D, E,F,G
- **Internal Node** : A node that is not root or leaf node. It is found in middle portion of tree i.e. B,C
- **Parent Node:** Nodes that have outdegree greater than 0 i.e. it has successor node i.e. A, B, C
- **Child Node:** Nodes that have indegree 1 i.e. one predecessor

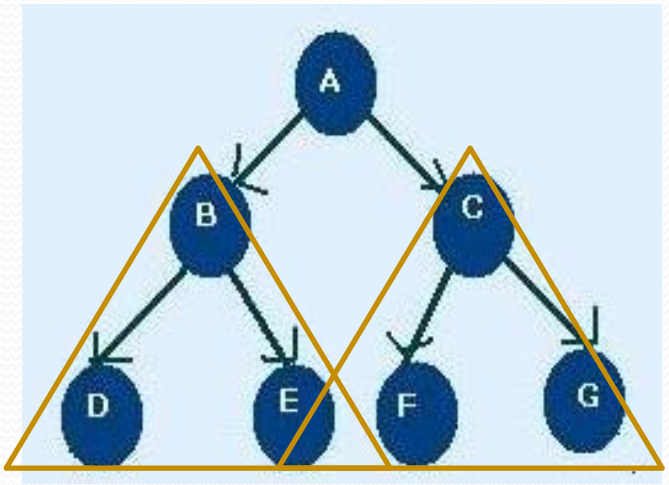
  i.e. B, C, D, E, F, G, H
- **Sibling :** Nodes having same parent i.e. B,C are sibling and D, E and F,G are sibling
- **Path:** A sequence of connected node from one node to another. i.e. path from root to D is ABD, path from C to G is CG
- **Ancestor of a Node :** Any node in the path from root to that node. i.e. Ancestor of D is A,B
- **Descendent of a Node :** All node in the path from that node to the leaf node. i.e. Descendent of : B is D

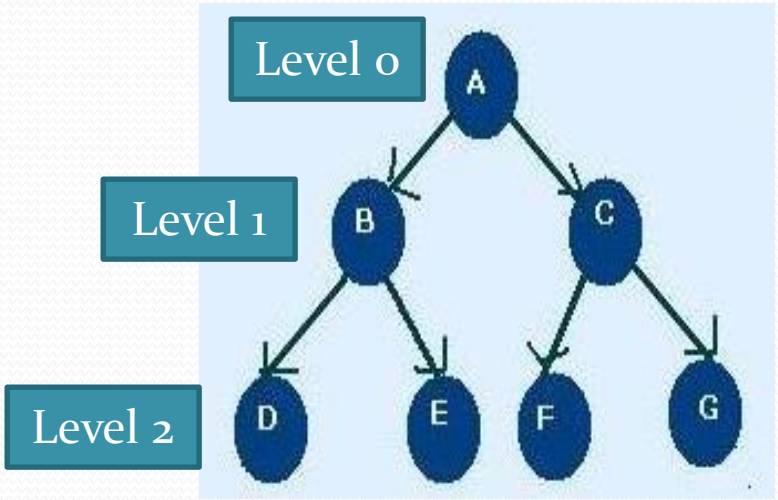


Tree Terminology

- **Level/depth:** level of a node is its distance from root. i.e. root has 0 distance from itself, so root is at level 0

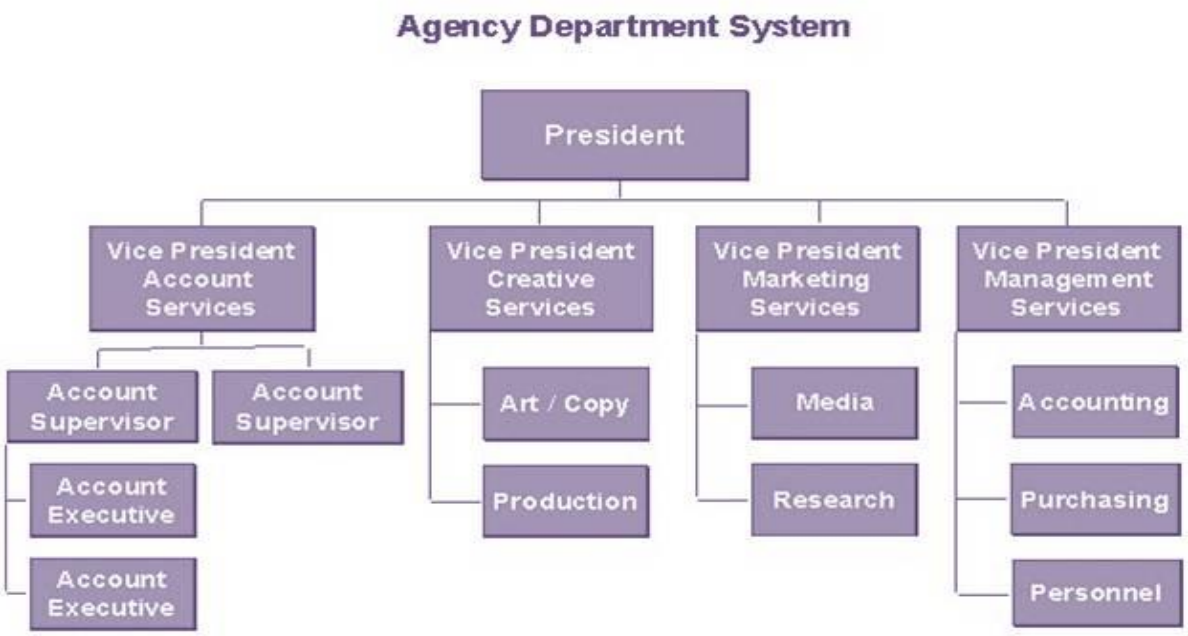
  the children of the root are at level 1 and their children are at level 2 and so on
- **Height :** Maximum level of any node in the tree.i.e. 2
- **Subtrees:** A tree may be divided into subtrees. The first node of the subtree is considered as root and is used to name the subtree. i.e BDE, CFG





Use of tree

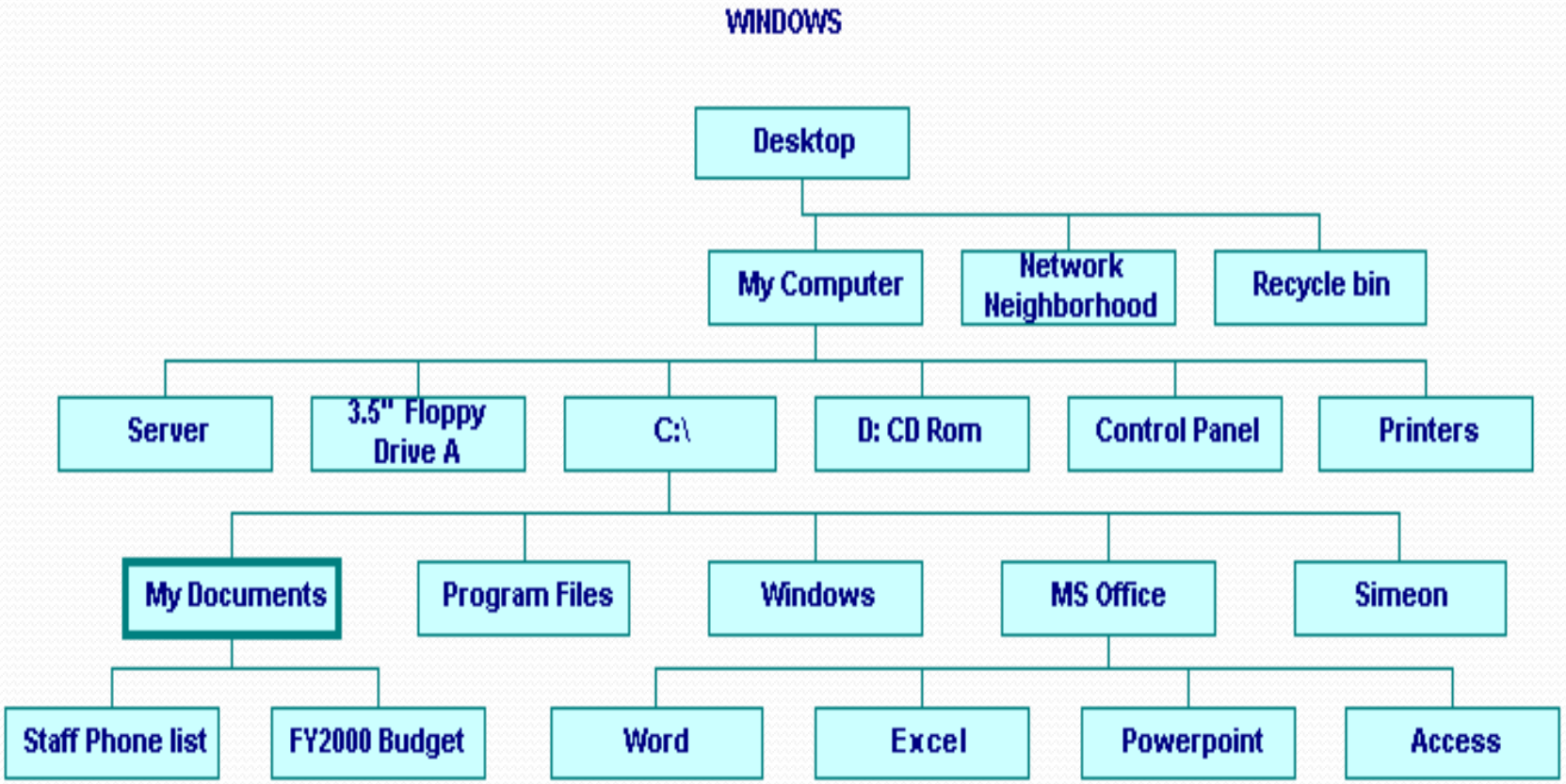
- Tree represents a hierarchy for. e.g. the organization structure of a company.



- Table of contents of a book
- Unix or Windows file system

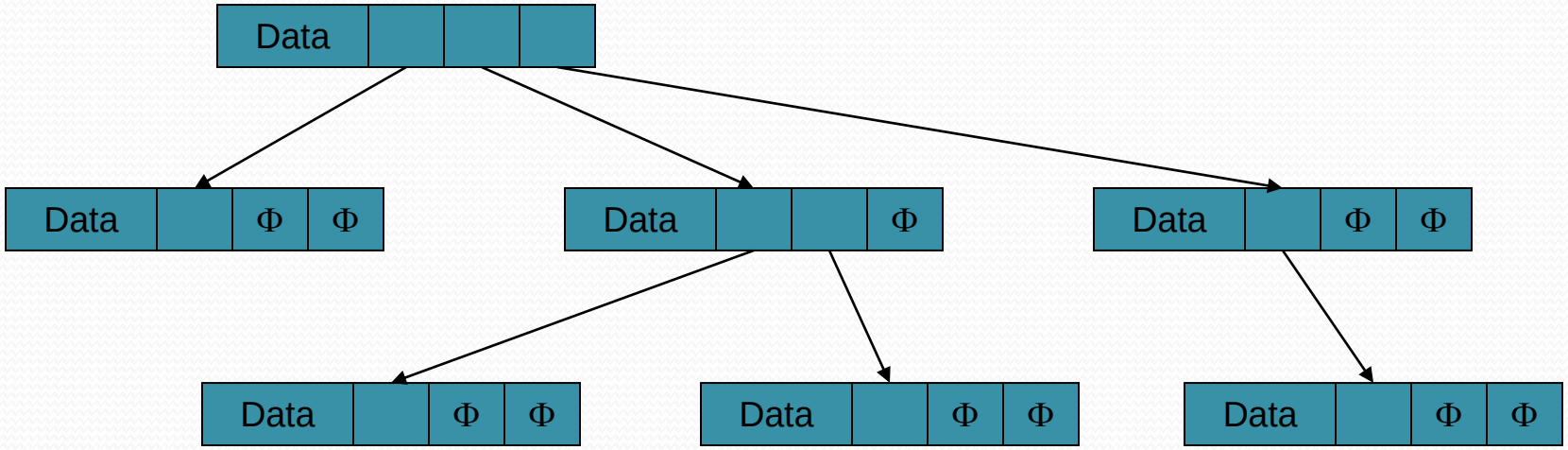
Use of tree

- Unix or Windows file system



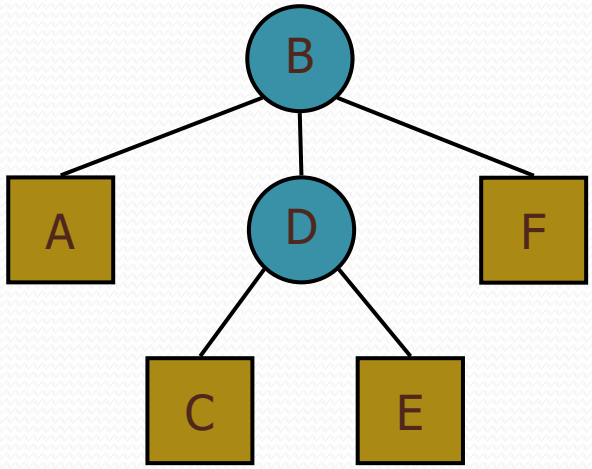
Trees

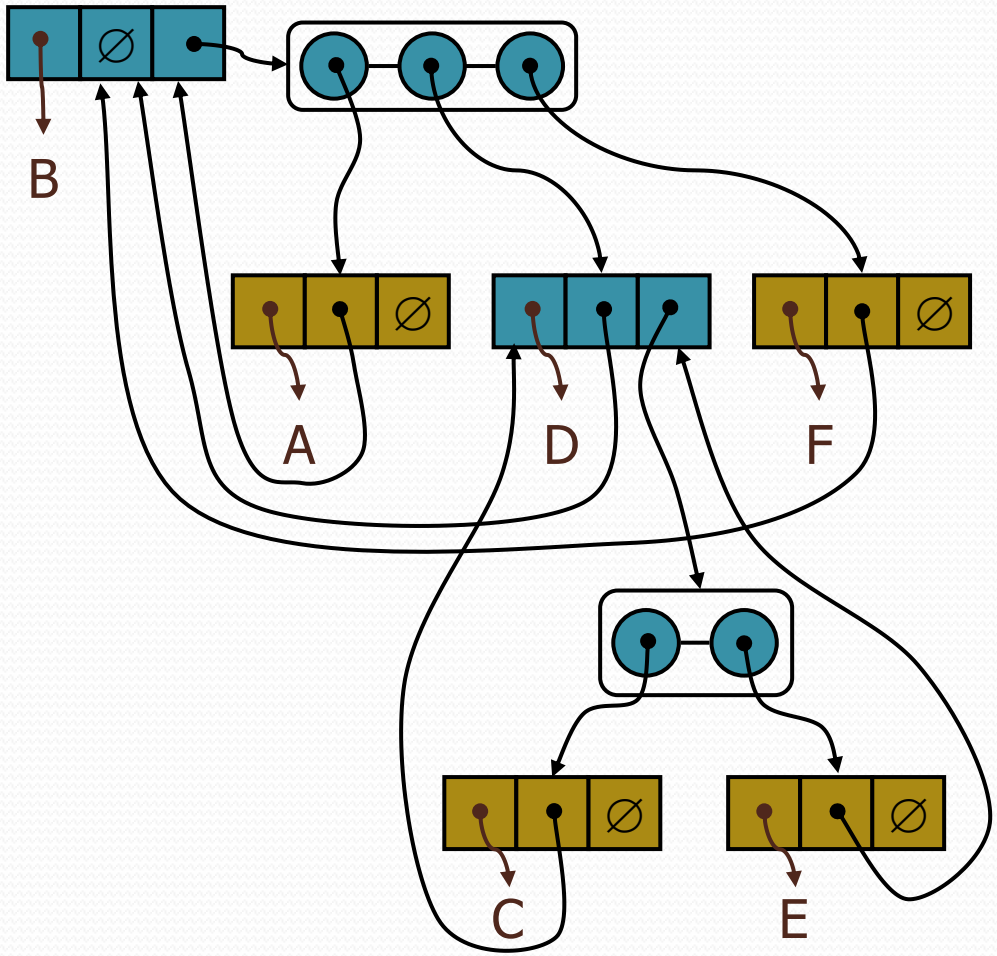
- Every tree node:
  - object – useful information
  - children – pointers to its children



A Tree Representation

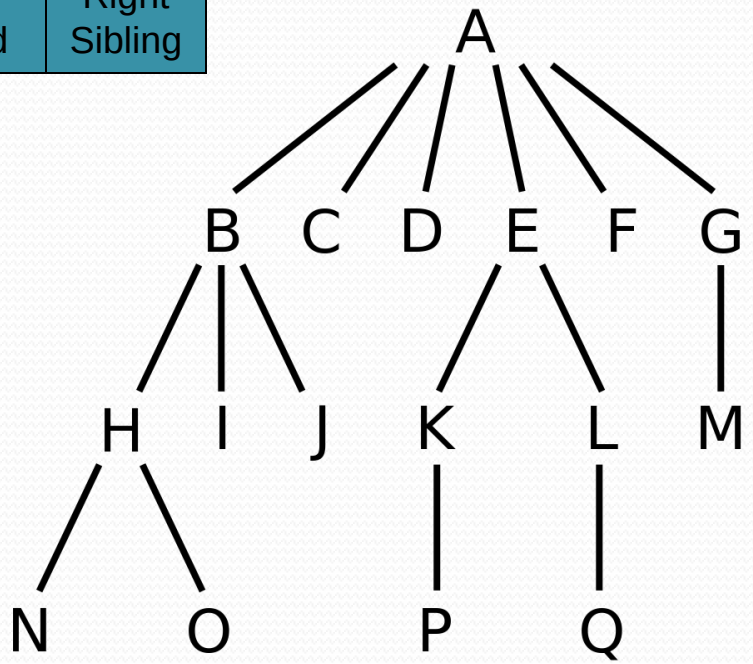
- A node is represented by an object storing
  - Element
  - Parent node
  - Sequence of children nodes

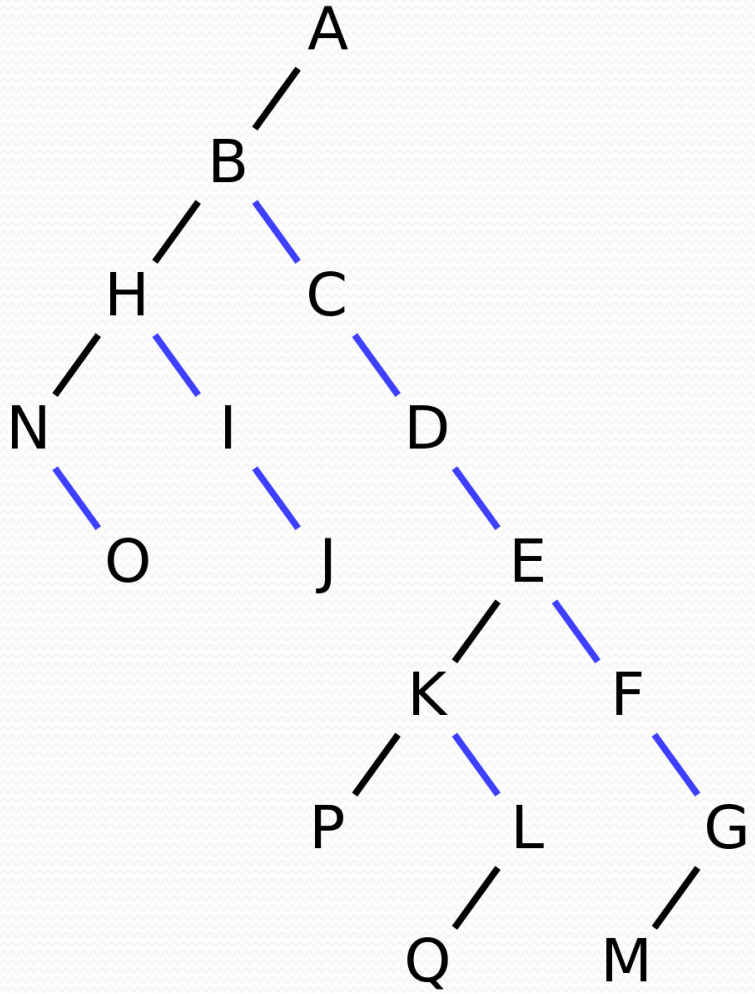




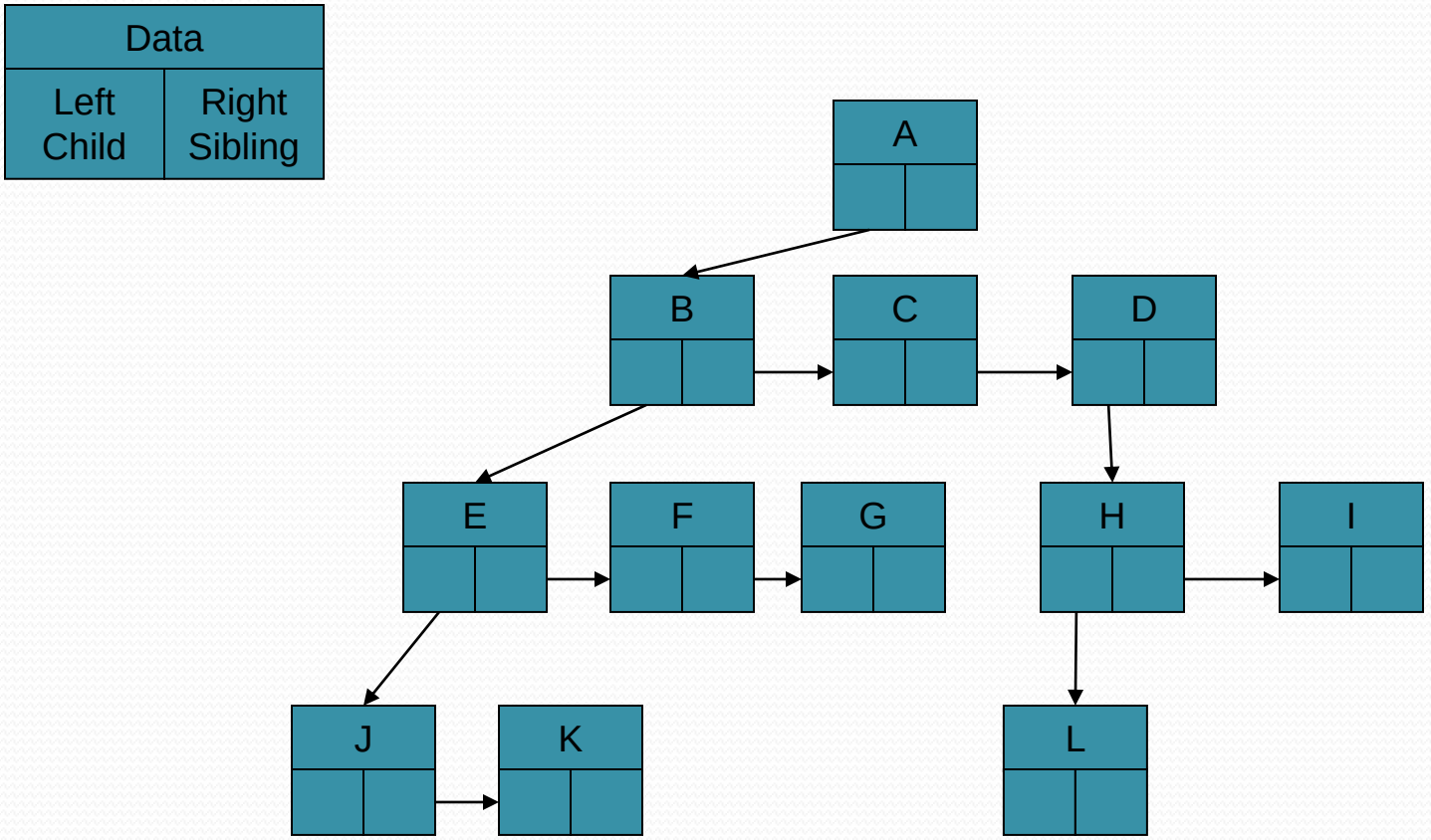
Left Child, Right Sibling Representation

Data	
Left Child	Right Sibling





Inclass Exercise Draw the tree



Tree Traversal

- Two main methods:
  - Preorder
  - Postorder
- Recursive definition

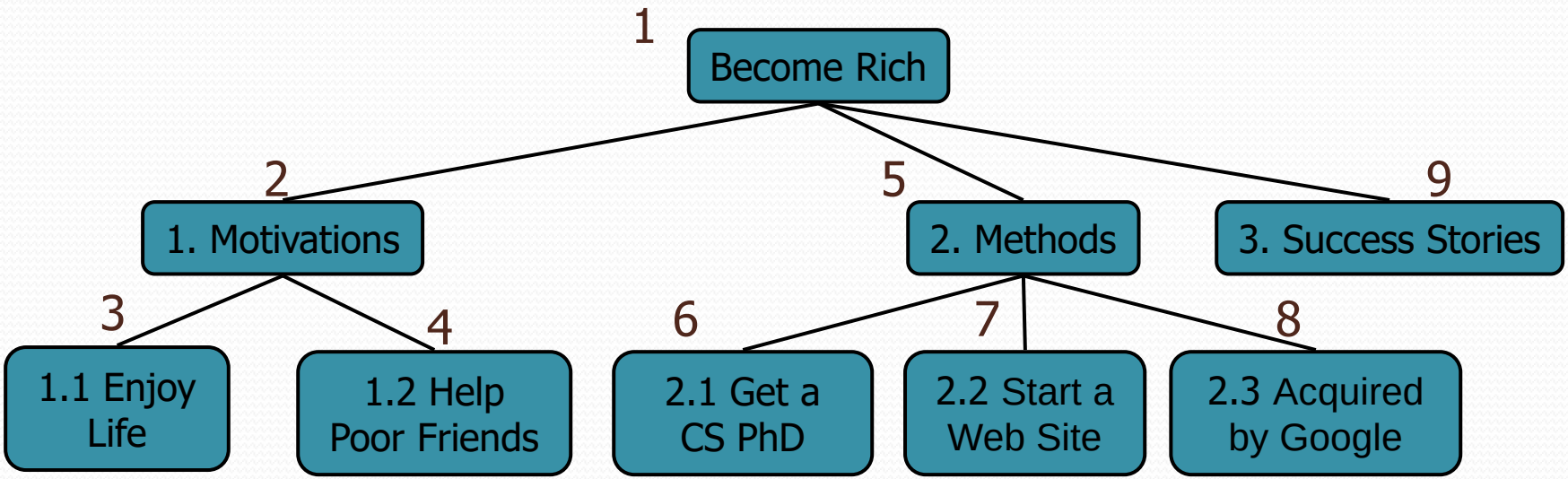
- Preorder:
  - visit the root
  - traverse in preorder the children (subtrees)

- Postorder
  - traverse in postorder the children (subtrees)
  - visit the root

Preorder Traversal

- A traversal visits the nodes of a tree in a systematic manner
- In a preorder traversal, a node is visited before its descendants
- Application: print a structured document

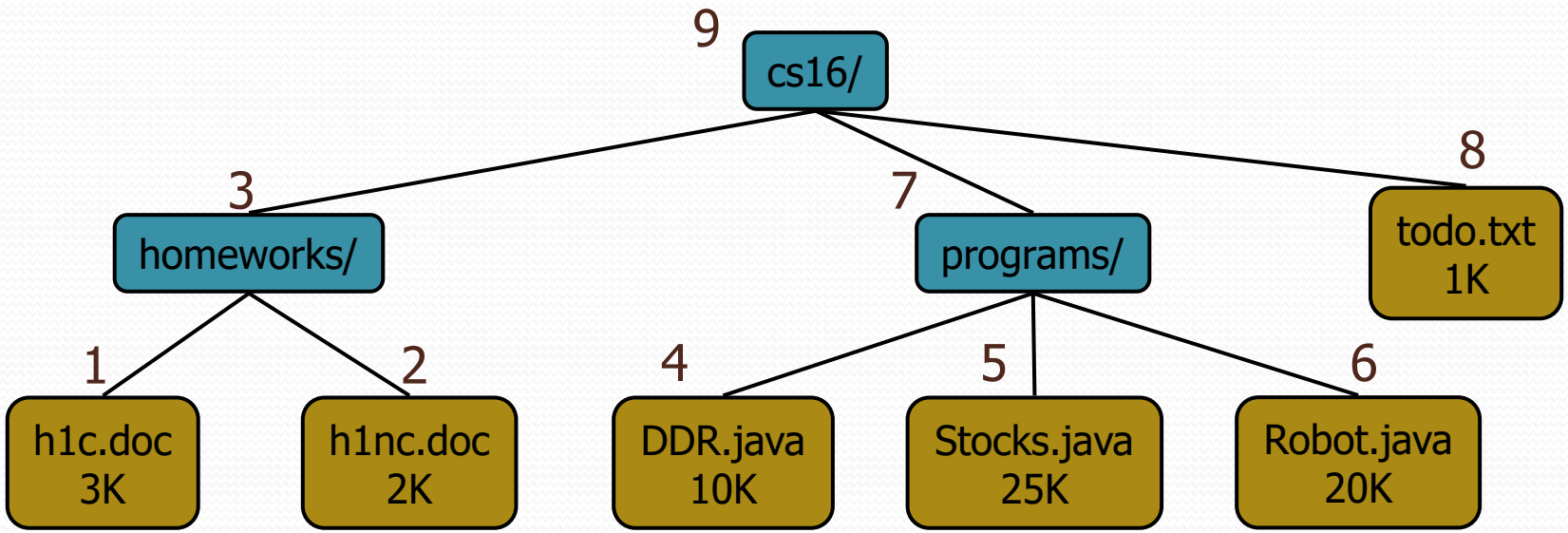
Algorithm ***preOrder*(*v*)**
    ***visit*(*v*)**
    for each child ***w*** of ***v***
        preorder **(*w*)**



Postorder Traversal

- In a postorder traversal, a node is visited after its descendants
- Application: compute space used by files in a directory and its subdirectories

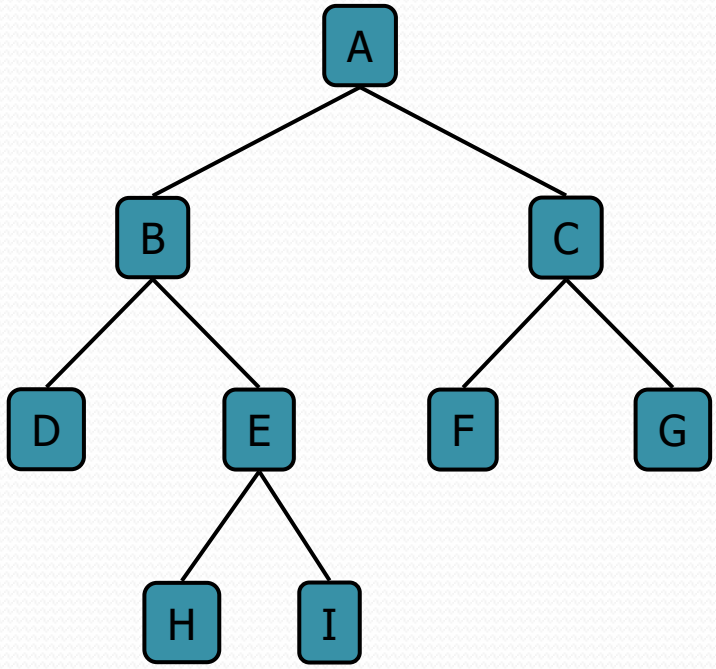
Algorithm ***postOrder***(***v***)
    for each child ***w*** of ***v***
        postOrder (***w***)
    visit(***v***)



Binary Tree

- A binary tree is a tree with the following properties:
  - Each internal node has at most two children (degree of two)
  - The children of a node are an ordered pair
- We call the children of an internal node left child and right child
- Alternative recursive definition: a binary tree is either
  - a tree consisting of a single node, OR
  - a tree whose root has an ordered pair of children, each of which is a binary tree

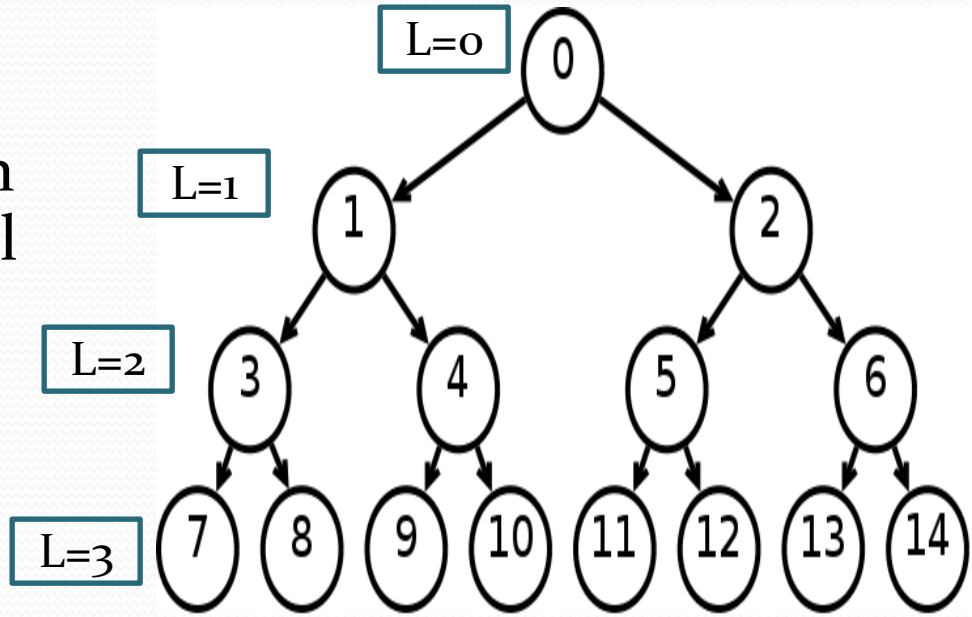
- Applications:
  - arithmetic expressions
  - decision processes
  - searching



Complete Binary tree

- Every nodes (except leaf) having two nodes.
- Maximum number of nodes in Level i is 2^i. i.e. nodes in level 0= 2^0 = 1, level 1 = 2^1 = 2
- In a tree of height h
  - No. of leaf/leaves at level h is 2^h
- If the tree has n nodes then,
- $n <= 2^{h+1} - 1$

  $=> h >= \log_2 (n+1) - 1$
- A binary tree has at least height of $\log_2 (n+1) - 1$ which is a complete binary tree



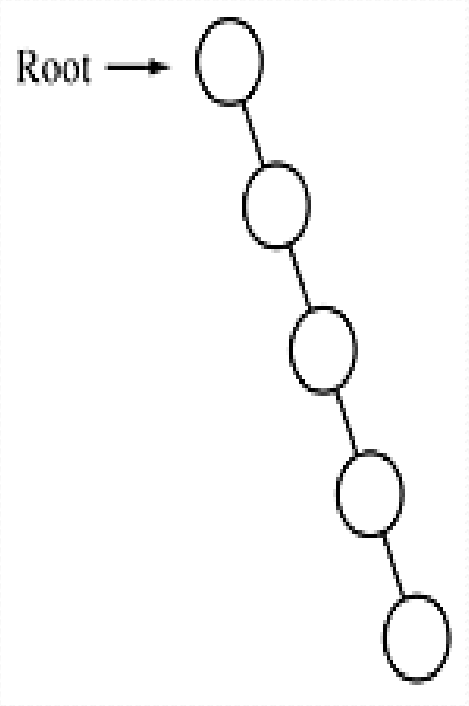
Minimum height of a Binary trees

- **A binary tree of height h has,**
  - At most 2^i nodes in level i
  - At most $2^0 + 2^1 + 2^2 + \ldots\ldots\ldots + 2^h = 2^{h+1} - 1$ nodes
- **If the tree has n nodes then,**
  - $n <= 2^{h+1} - 1$

  $=> h >= \log_2 (n+1) - 1$
- A binary tree has at least height of $\log_2 (n+1) - 1$ which is a complete binary tree

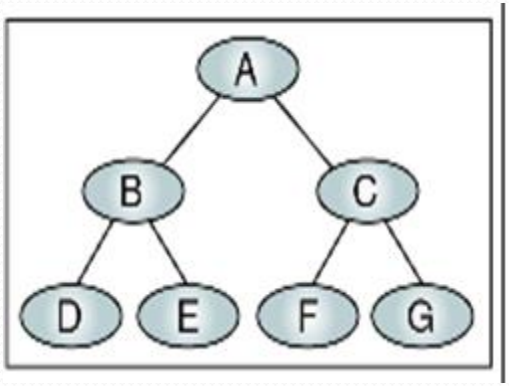
Maximum height of a Binary trees

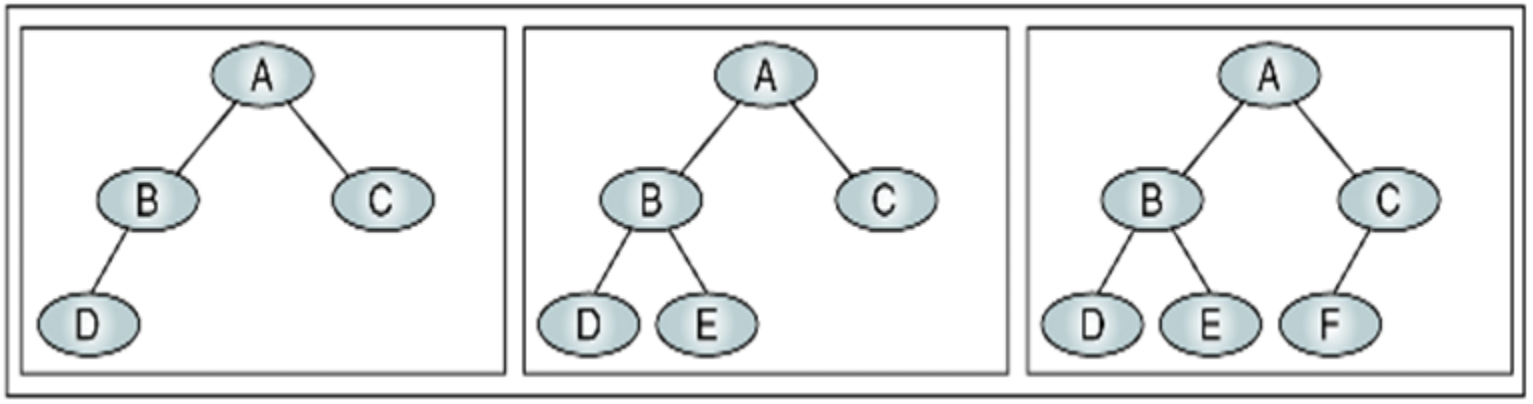
- If the tree has n nodes then the height is n-1
- This is obtained when every node has exactly one child (except leaf node)



Near Complete Binary Tree

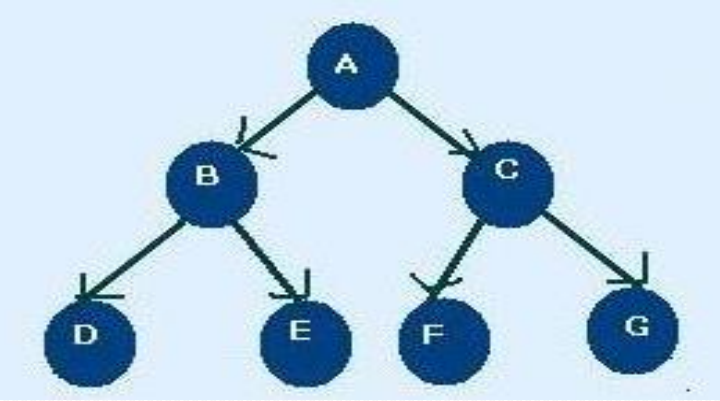
- If a tree has the minimum height for its nodes and all nodes in the last level are found on the left is called near complete binary tree.





Binary Tree Traversal

- Each Node of the tree must be processed once and only once in a predetermined sequence
- Two general approach-
  - Depth First traversal
  - Breadth first traversal
- **Depth First traversal,** the processing proceeds along a path from the root through one child to the most descendent of that first child before processing a second child
- **Breadth first traversal (level by level),** the processing proceeds horizontally from the roots to all its children (left to right usually) i.e. A B C D E F G



Depth First Traversal

- Three Standard traversal techniques :

1. Preorder traversal

   <root > <left> <right>

   - Visit the root
   - Visit the left subtree
   - Visit the right subtree

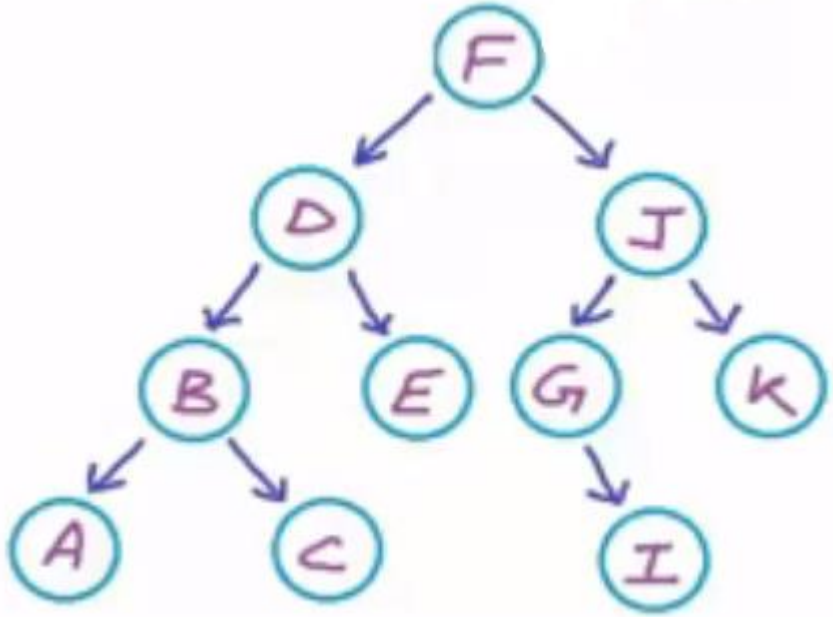
   i.e. F D B A C E J G I K

2. Inorder traversal

   <left> <root > <right>

   - Visit the left subtree
   - Visit the root
   - Visit the right subtree

   i.e. A B C D E F G I J K



Depth First Traversal

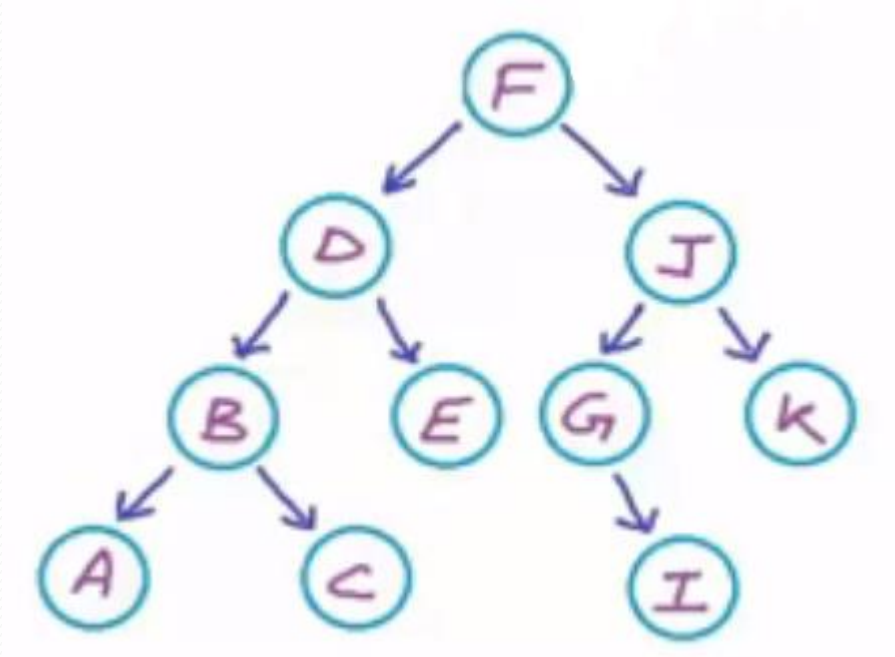
- Three Standard traversal techniques

1. Postorder traversal

   <left> <right> <root >

   - Visit the left subtree
   - Visit the right subtree
   - Visit the root

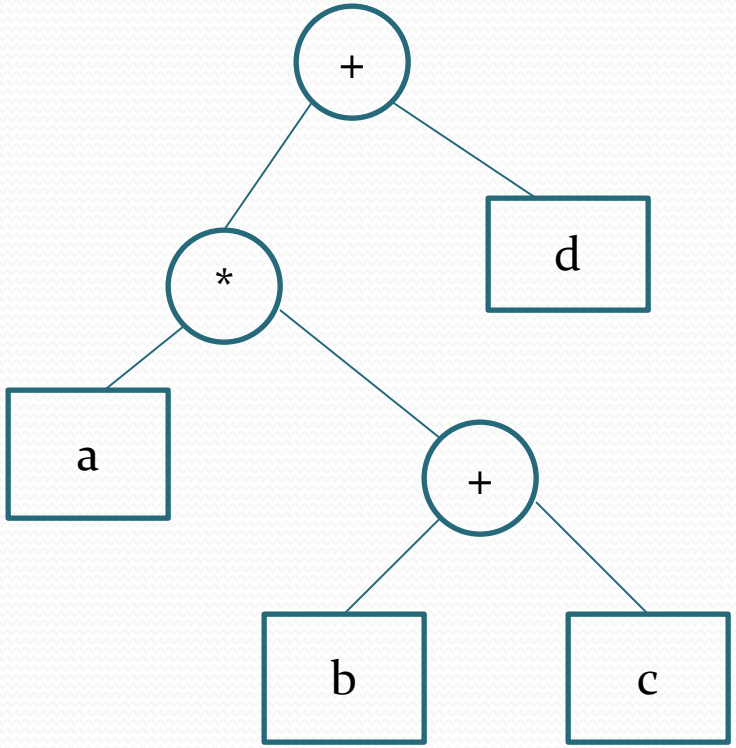
   i.e. A C B E D I G K J F



Expression Trees

- An expression tree is a binary tree with following properties :
  - Each leaf is an operand
  - The root and internal nodes are operator
  - Sub trees are sub expressions with the root being an operator

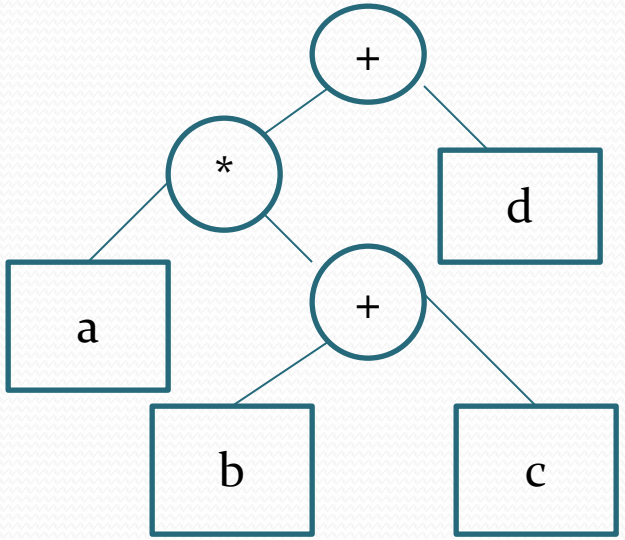
a * (b + c) +d



Infix Traversal

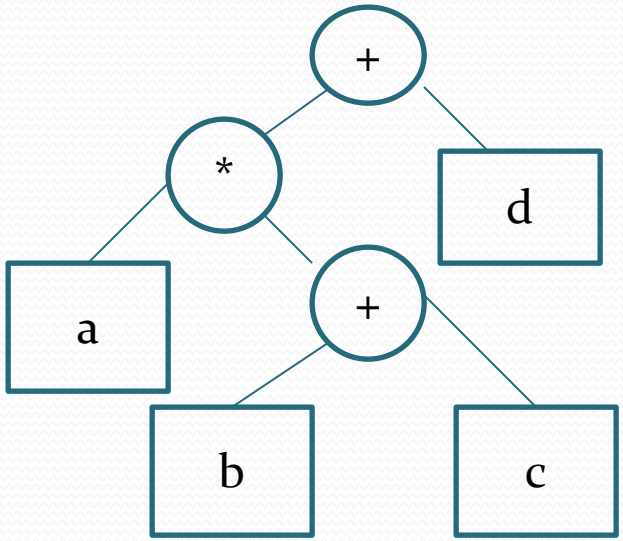
```
Algorithm infix (tree)
    if (tree not empty)
        if(tree token is an operand)
          print (tree token)
        else
          print ( open parenthesis)
          infix (tree left subtree)
          print (tree token)
           infix (tree right subtree)
          print (close parenthesis)
         end if
      end if;
```

((a * (b + c)) +d)



Prefix Traversal

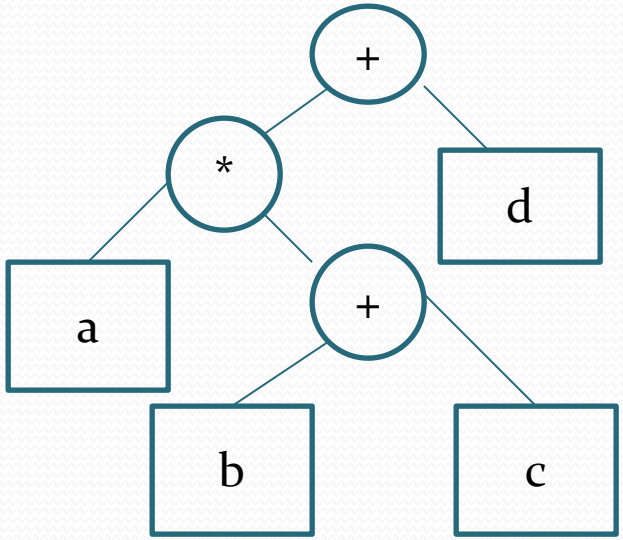
```
Algorithm prefix (tree)
    if (tree not empty)
        print (tree token)
        prefix (tree left subtree)
        prefix (tree right subtree)
     end if;
```



+*a+bcd

Postfix Traversal

```
Algorithm  postfix (tree)
    if (tree not empty)
        postfix (tree left subtree)
        postfix (tree right subtree)
        print (tree token)
    end if;
```



abc+*d+