School of Computing & Informatics (SCI)
Universiti Teknologi Brunei

CI2142 - Data Structures & Algorithms

Tree data structure and its application

- Introduction to the Tree Data Structure
- Binary Trees
- Traversals in a Binary Tree
- Java Implementation

Tree based representations

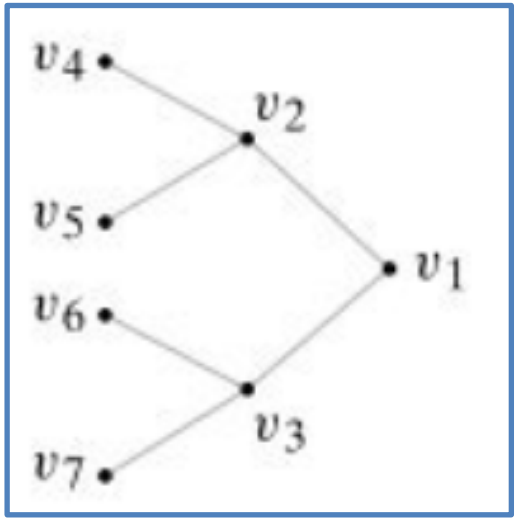
Monica Seles	**6**	6^{3}	**6**
Martina Navratilova	2	$\mathbf{7}^{7}$	4

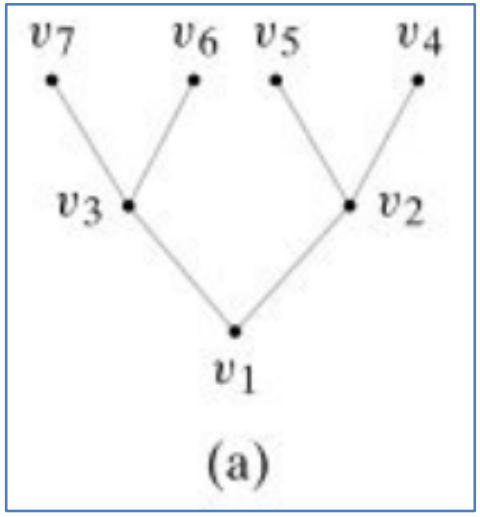
1	Monica Seles	2	1
2	**Steffi Graf**	**6**	**6**

1992 Wimbledon

Steffi Graf

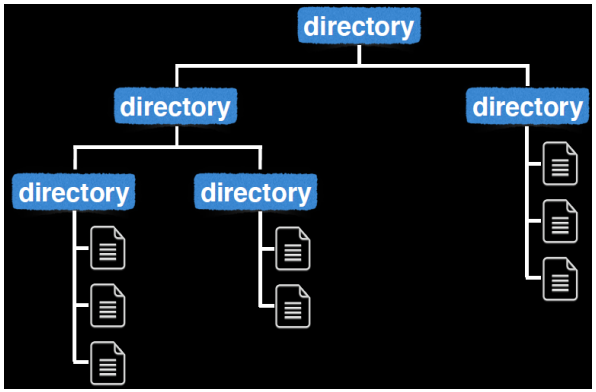
Gabriela Sabatini	3	3	
Steffi Graf	**6**	**6**	



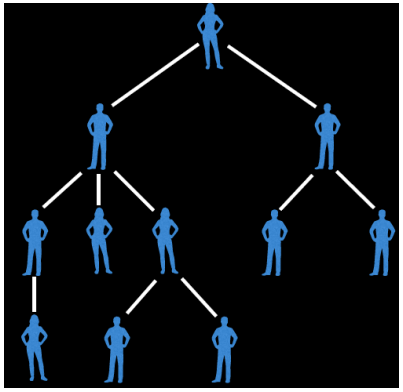


Potential Application of Trees:

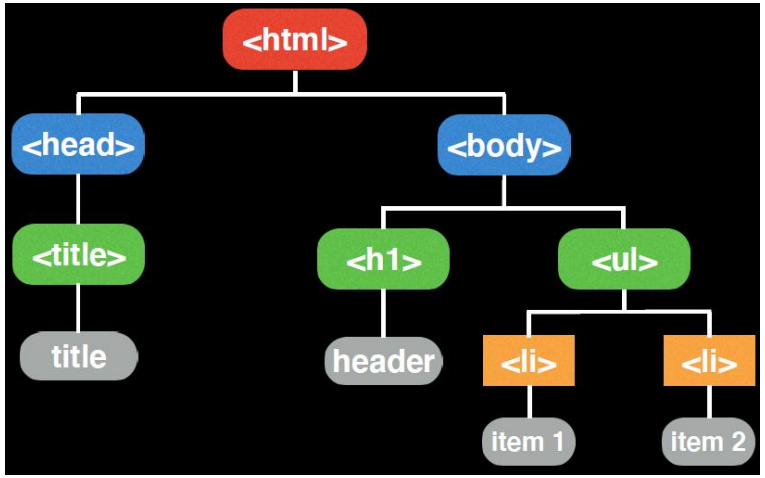
The following applications use a tree data structure as very efficient algorithms exist to process *big data* represented as trees.



File system in PCs



Social networks

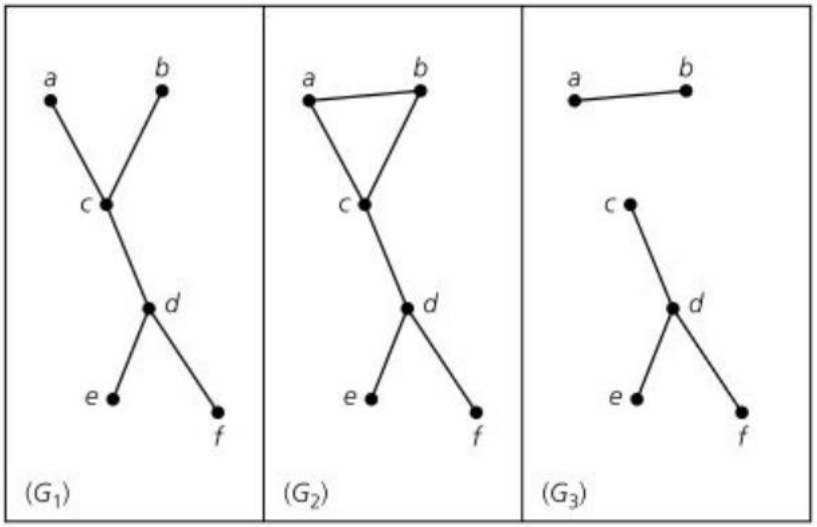


Web pages are basically trees!

Decision Trees, Domain Name Servers (DNS), Databases,….all use trees.

Terminology / Definitions

Let G = (V, E) be a simple undirected graph. The graph G is called a tree if G is connected and contains no cycles.



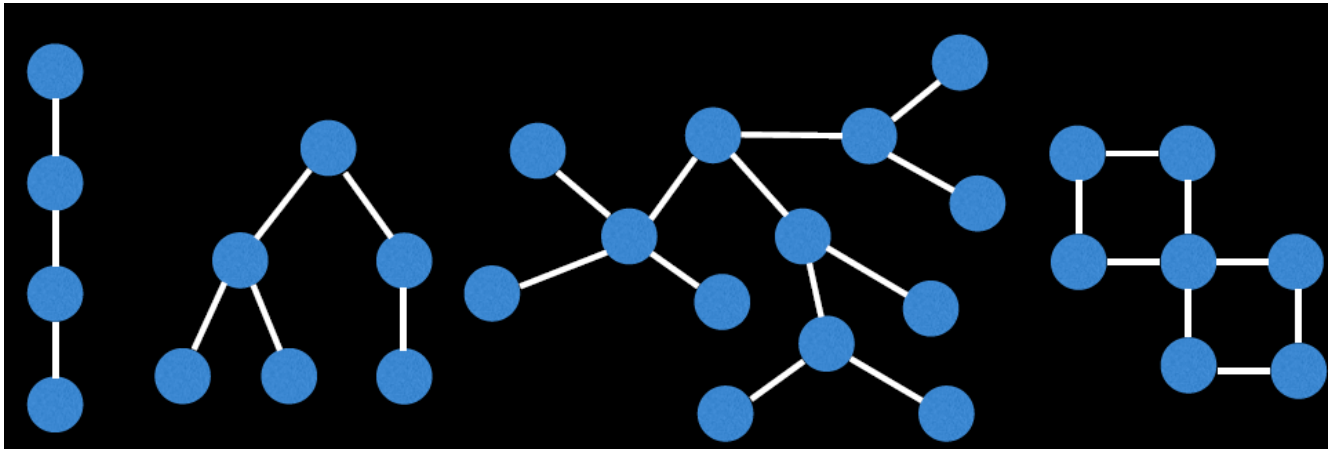
Identify trees?

Is G1 a tree? Justify?

In G2 {a,b}, {b,c} and {c,a} is called as a _____

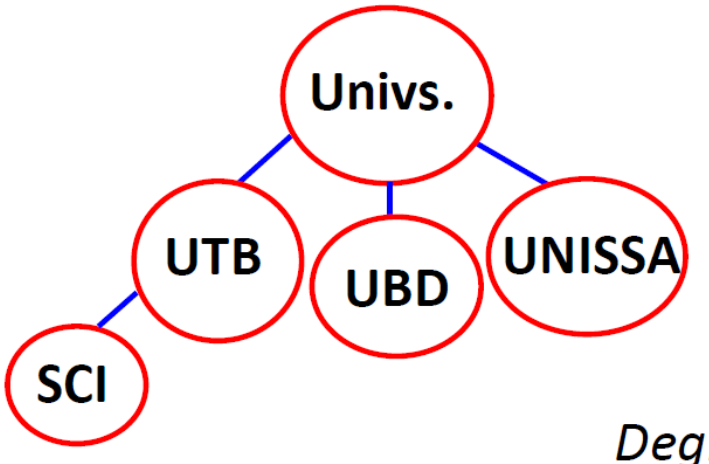
G3 is called as a _____

What is a Forest?



G4 G5 G6 G7

Terminology / Definitions



A tree is a _____ data structure that has a finite set of _____ and _____

The number of subtrees of a node is called the _____ of the node.

Degree of the node "Univs." is _____

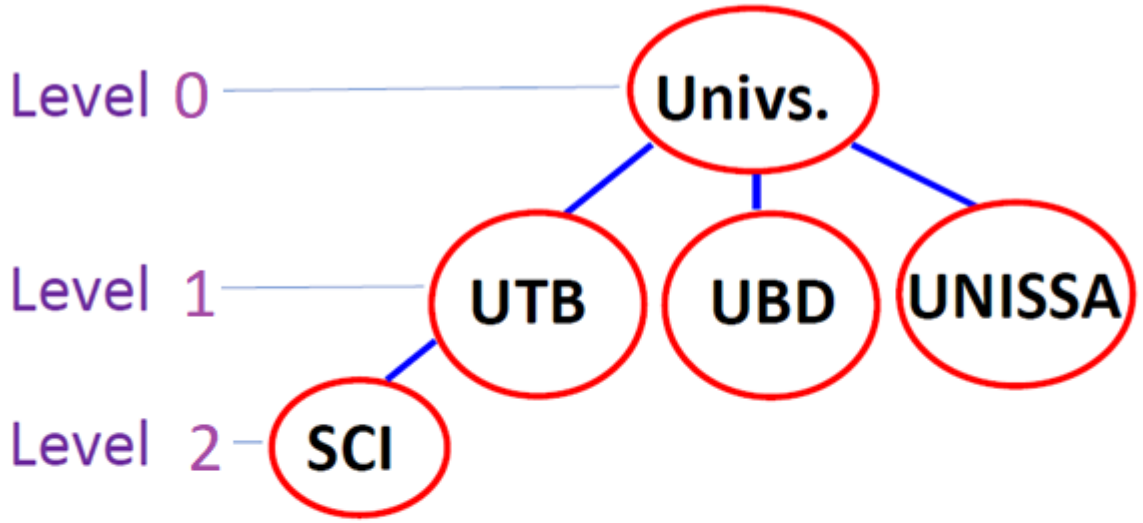
A node of degree zero is called a *terminal node* or ___ *node*.

Is the node "UTB" a leaf? What are leaf nodes in the eg.?

The *degree of a tree* is the maximum degree of a node in the tree.

What is the degree of the eg. tree?

Terminology / Definitions



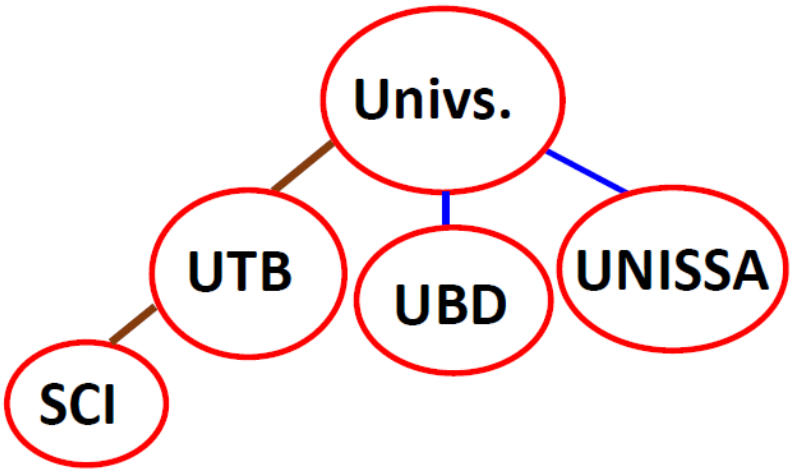
The ____ of a node is a measure of its distance from the root.

Recursively,

- if node n is the root of the tree T, its ____ is 0
- if not then the level is 1+ ____ of its parent

The alternative term for _____ is depth

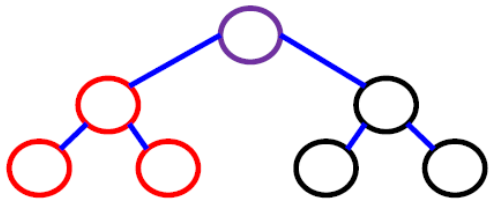
Terminology / Definitions



The height of a tree is the number of nodes in the longest path from the root node to a leaf node. (Height of the Eg. Tree is 3)

- if T is empty, its height is zero
- if not then the height is the max. depth of its nodes.

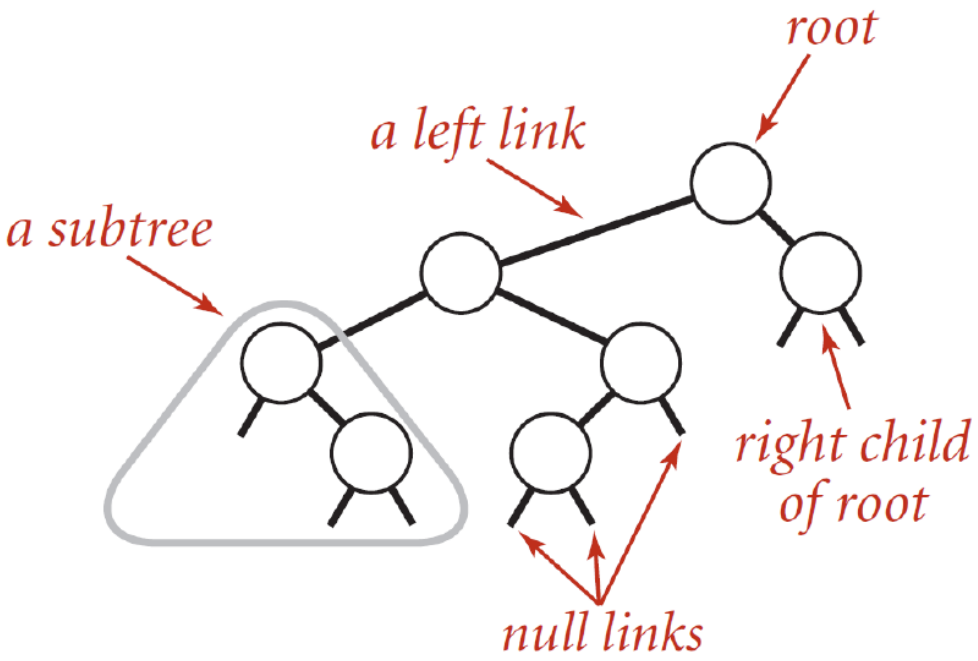
Binary Tree

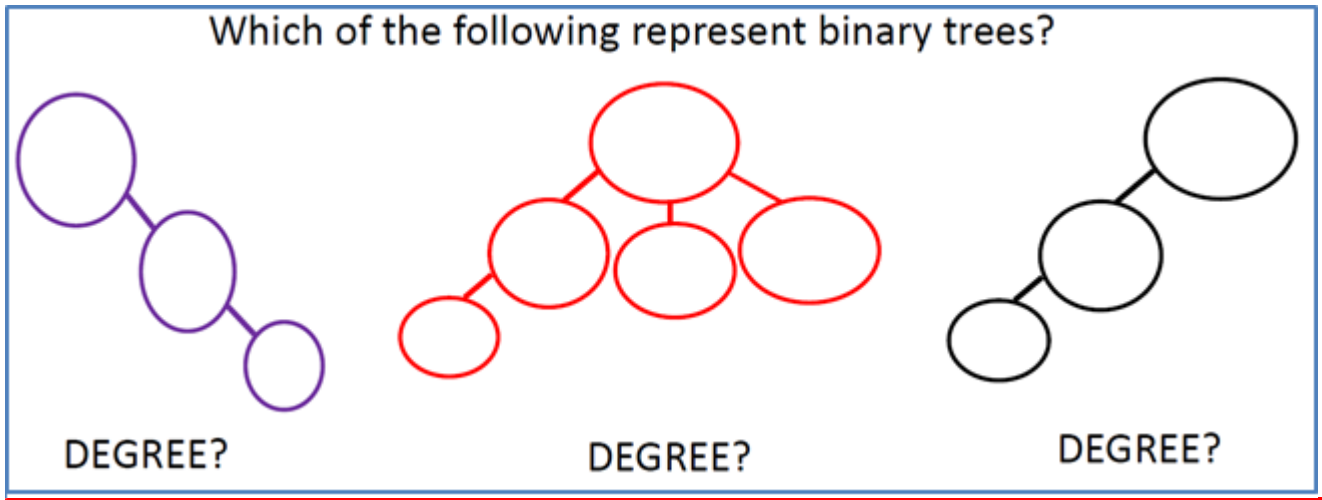


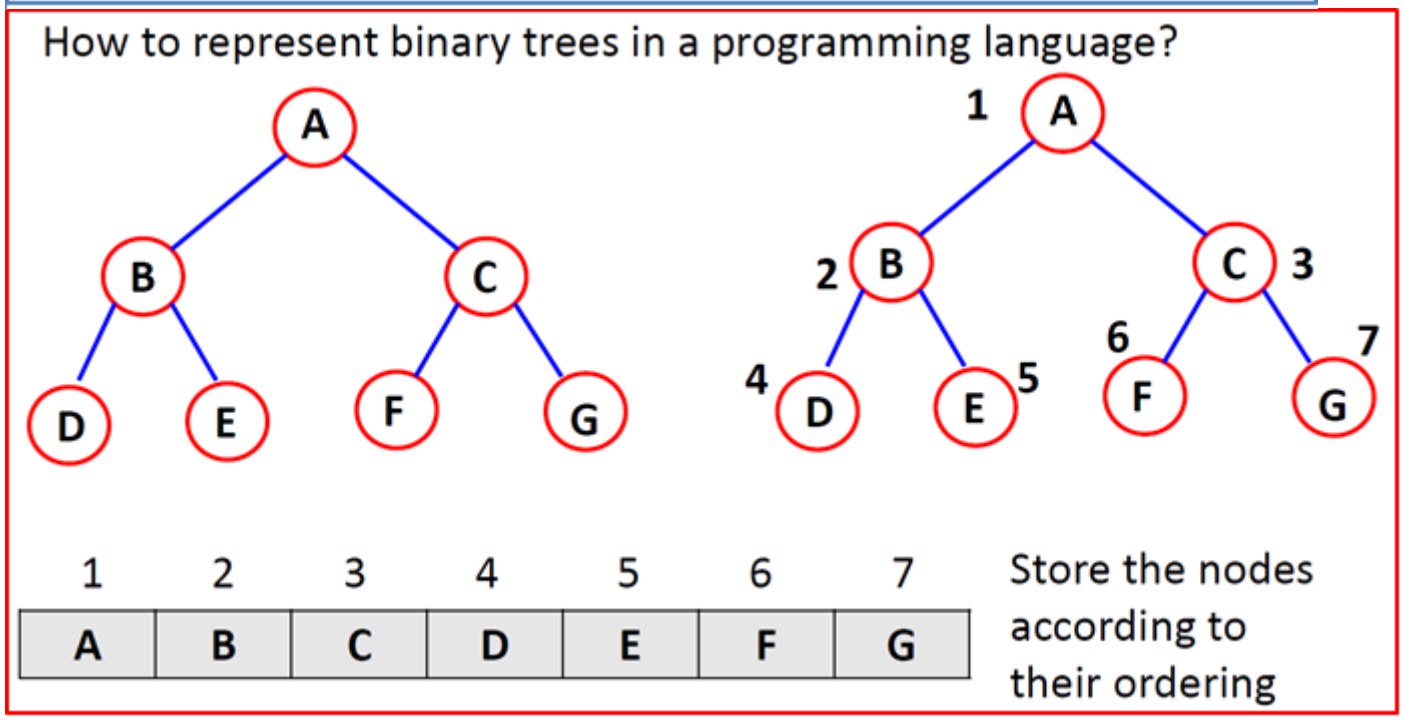
A set of nodes T is a binary tree if either of the following is true

- *T is empty*
- *If not then its root has two subtrees T_L and T_R*

Characteristics of a Binary Trees



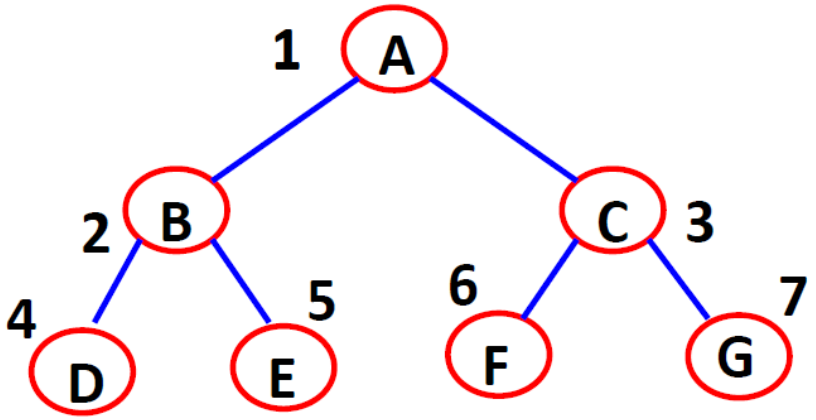




Sequentially number the nodes from root (left to right) until the lowest level

Note: The starting index for some programming languages (Eg. MATLAB) is one

How to represent binary trees in a programming language?



For any ith node

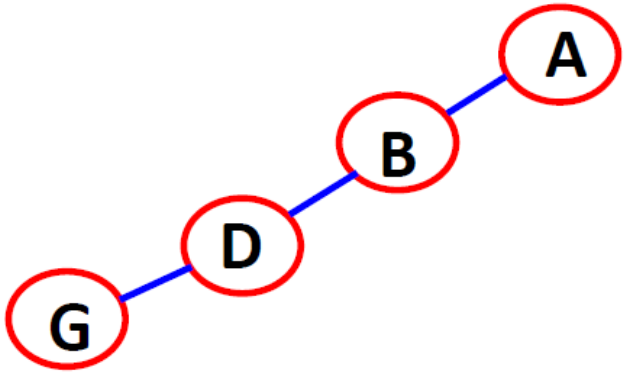
Parent(i) => i /2

Left Child(i) => 2i

Right Child(i) => 2i+1

1	2	3	4	5	6	7
A	B	C	D	E	F	G

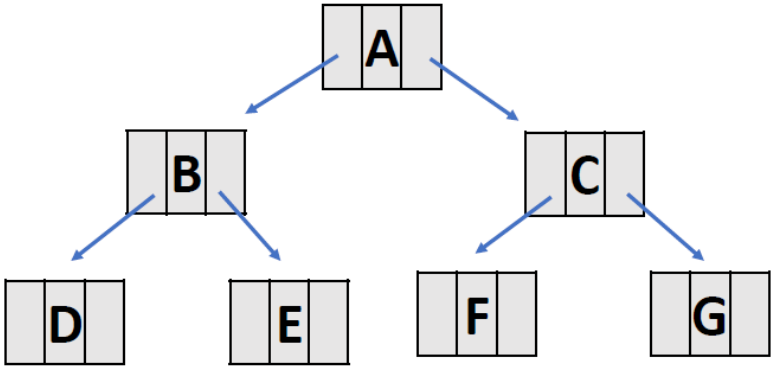
Represent the following binary tree (skewed BT) in an array

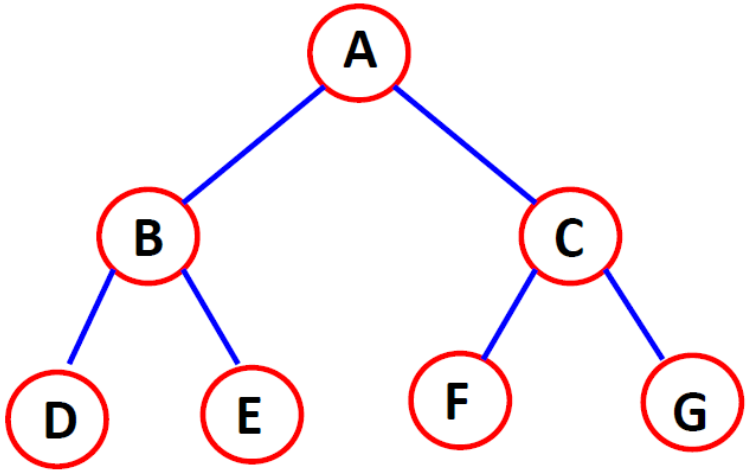


Linked list representation of binary trees

Left Child	Data	Right Child

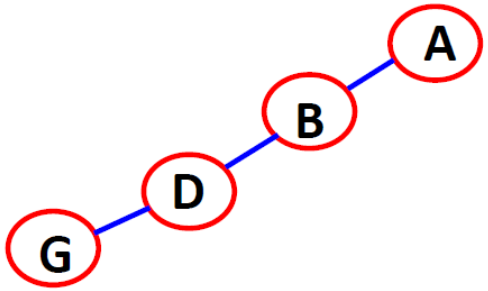
Left child and Right child are pointers





While traversing a binary tree, each node is visited exactly once.

Represent the following binary tree (skewed BT) in an array



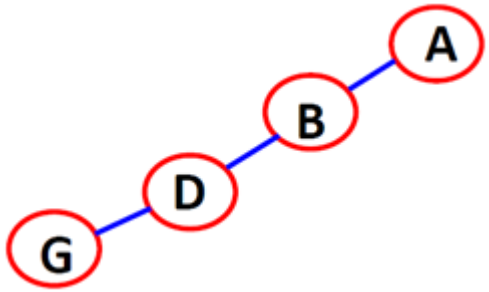
For any ith node

Parent(i) => i /2

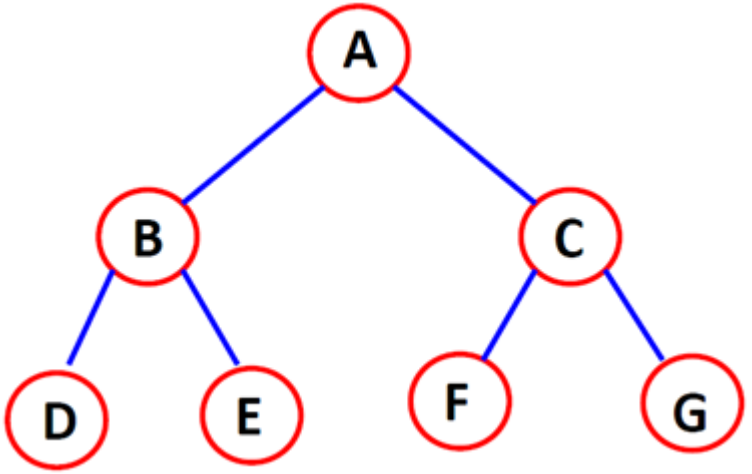
Left Child(i) => 2i

Right Child(i) => 2i+1

1	2	3	4	5	6	7	8
A	B		D				G



Imbalanced binary tree



Balanced binary tree

Traversing in a Binary tree

Pre-Order Traversal: Root, Left ,Right

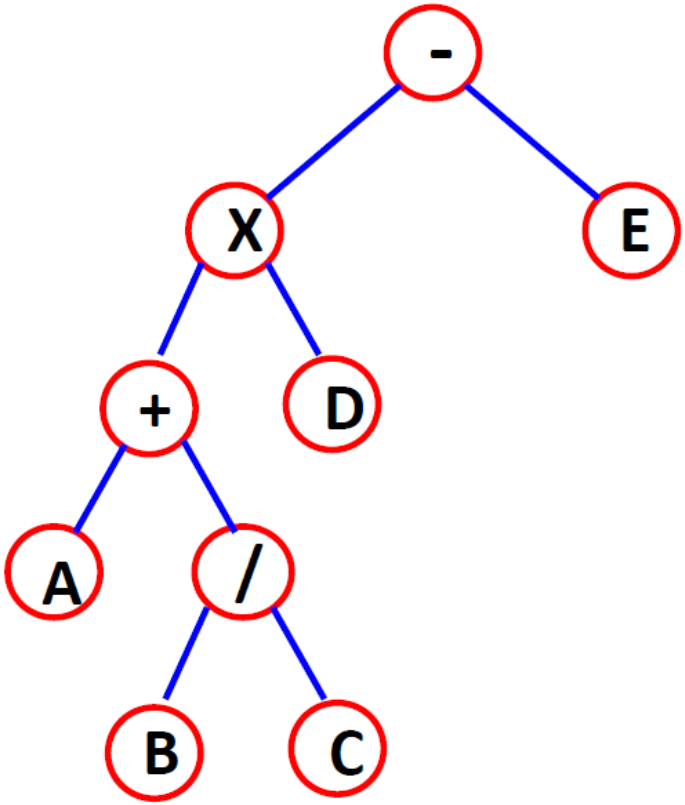
-	x	+	A					

Since there are no left or right
Subtrees after A we climb up
to + and move down right

-	x	+	A	/	B	C		

Move down to the right of X

-	x	+	A	/	B	C	D	E



Do in-order and post-order yourself !

In-Order Traversal: Left Most, Root, Right

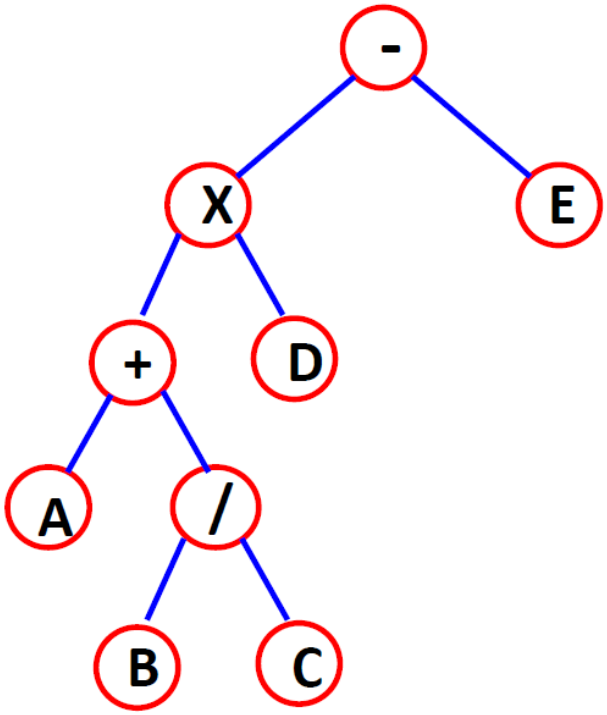
Post-Order Traversal: Left , Right, Root

Traversing in a Binary tree

Level-Order Traversal:

The root is visited first and then all
Nodes at depth 1 (going from left to right),
Then all nodes at depth 2 (going fr left to right)
and so on…

-	X	E	+	D	A	/	B	C



Application of BTs: Develop Compilers,
Interpreters, Virtual Machines (Java),
DBMS, OS and many more

Traversing in a Binary tree

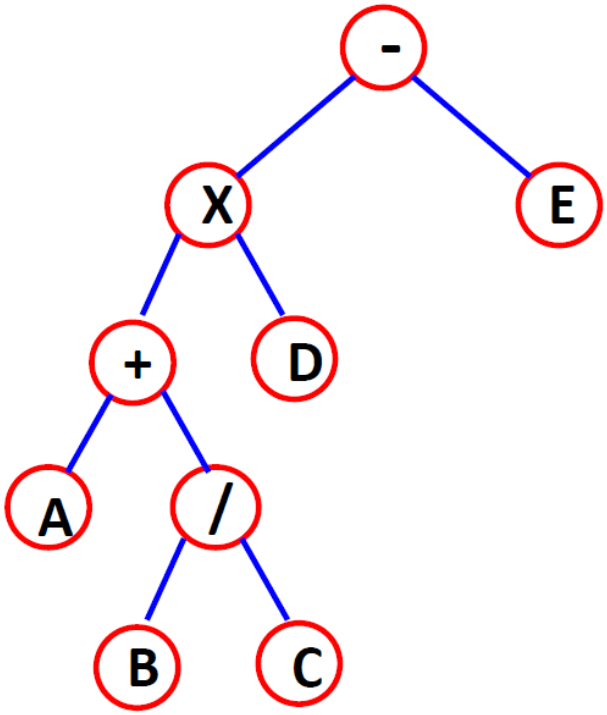
In-Order Traversal: Left Most, Root, Right

A	+	B	/	C	X	D	-	E

Post-Order Traversal: Left , Right, Root

A	B	C	/	+	D	X	E	-

Level – Order Traversal: Can you guess?



```java
/* Defines an ADT for a Binary tree */
public abstract class AbstractBinaryTree {
    protected Node rootNode;
    public AbstractBinaryTree(Node rootNode) {
        this.rootNode = rootNode;
    }
    /* Represents a node in binary tree. */
    protected static class Node{
        private int value; private Node left; private Node right;
        public Node(int value, Node left, Node right) {
            this.value = value; this.left = left; this.right = right;
        }
        public int getValue(){ return value;}
        public Node getLeft() {return left; }
        public Node getRight() { return right;}
    }//end of class Node

    /* Find the height of the binary tree given the root node.
    input parameter root; Returns height of the longest path. */
    protected int computeHeight(Node root) {
        if(null == root){ return 0;}
        int leftHeight = computeHeight(root.getLeft());
        int rightHeight = computeHeight(root.getRight());
        return leftHeight > rightHeight ? leftHeight + 1 : rightHeight + 1;
    }
}//end of class AbstractBinaryTree
```

Note: Refer to the Terminology/Definitions and linked-list representation (early slides); We implement those concepts what we have learned.

```java
public class BinaryTreeTraversals extends AbstractBinaryTree{
    public BinaryTreeTraversals(Node rootNode) {
        super(rootNode);
    }
    public static void main(String[] args) {
        /* Construct tree */
        BinaryTreeTraversals.Node r4 = new BinaryTreeTraversals.Node(4, null, null);
        BinaryTreeTraversals.Node r5 = new BinaryTreeTraversals.Node(5, null, null);
        BinaryTreeTraversals.Node r3 = new BinaryTreeTraversals.Node(3, null, null);
        BinaryTreeTraversals.Node r2 = new BinaryTreeTraversals.Node(2,r4,r5);
        BinaryTreeTraversals.Node r1 = new BinaryTreeTraversals.Node(1,r2,r3);
        BinaryTreeTraversals tree = new BinaryTreeTraversals(r1);
        System.out.println("InOrder traversal");
        tree.traverseInOrder(r1);
    }
    public void traverseInOrder(Node node){
        if(null == node){
        return;
        }
        /* Recur left */
        traverseInOrder(node.getLeft());
        /* Visit root */
        System.out.print(node.getValue()+" ");
        /* Rcur right */
        traverseInOrder(node.getRight());
        }
    }
```

```
D:\IBM\JavaWork>javac AbstractBinaryTree.java

D:\IBM\JavaWork>javac BinaryTreeTraversals.java

D:\IBM\JavaWork>java BinaryTreeTraversals
InOrder traversal
4 2 5 1 3
```

Lab Task

Write methods for pre-order, post-order and level-order.

Workout all the traversals by hand (pen and paper) for few binary tree examples.

Develop clients to verify the outputs with your hand-worked examples.