Advanced Algorithms & Data Structures

AIM OF THE SESSION

To familiarize students with the basic concept of Binary tree

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1. Demonstrate The Binary tree.
2. Describe the types of Binary tree traversals.
3. Constructing Binary tree and traversals.
4. Constructing an expression tree.
5. List out the advantages and applications of Binary tree.

LEARNING OUTCOMES

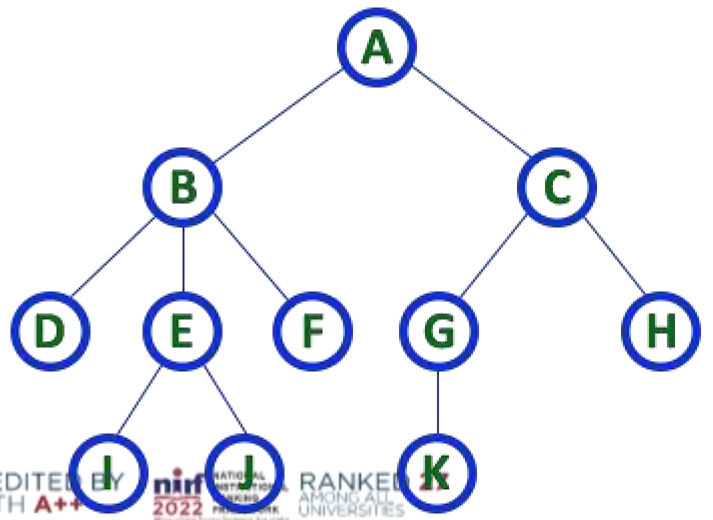
At the end of this session, you should be able to:

1. Define Binary tree
2. Describe the various traversals in Binary tree Expression tree construction
3. Summarize definition, types and operations of Binary tree and its applications

Binary tree
INTRODUCTION

- ➢ A binary tree is a special type of tree data structure in which every node can have a **maximum of 2 children**. One is known as a left child and the other is known as right child.

Example



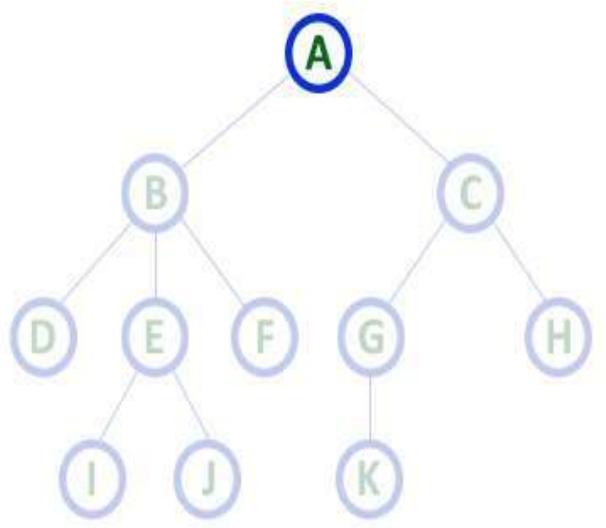
TREE with 11 nodes and 10 edges

- In any tree with 'N' nodes there will be maximum of 'N-1' edges
- In a tree every individual element is called as 'NODE'

Tree - Terminology

1. Root

In a tree data structure, the first node is called as **Root Node**.

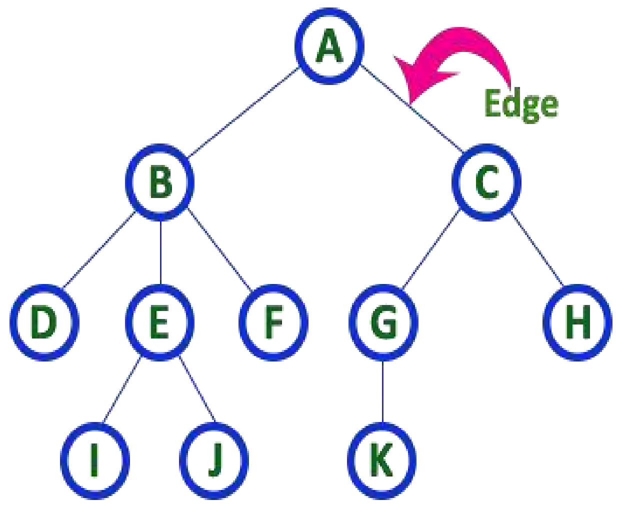


Here 'A' is the 'root' node

- In any tree the first node is called as ROOT node

2. Edge

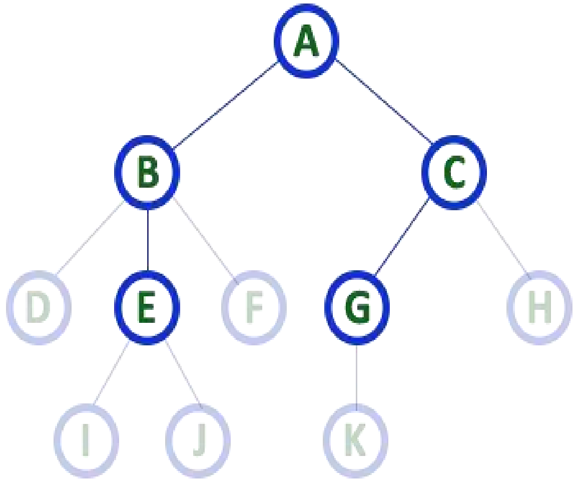
In a tree data structure, the connecting link between any two nodes is called as EDGE. In a tree with 'N' number of nodes there will be a maximum of 'N-1' number of edges.



- In any tree, 'Edge' is a connecting link between two nodes.

Tree - Terminology

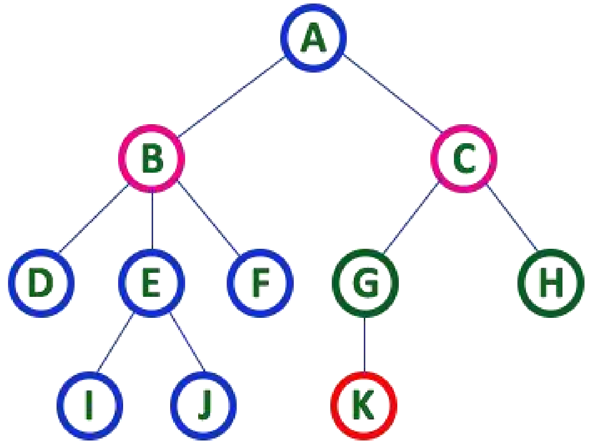
- **3. Parent**



Here A, B, C, E & G are Parent nodes

- In any tree the node which has child / children is called 'Parent'
- A node which is predecessor of any other node is called 'Parent'

4. Child



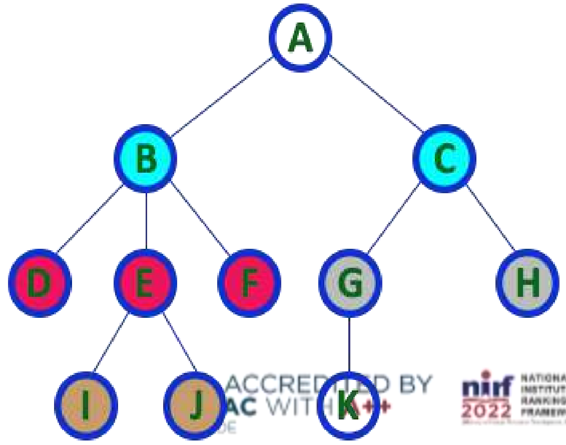
Here B & C are Children of A

Here G & H are Children of C

Here K is Child of G

- descendant of any node is called as CHILD Node

5. Siblings



Here B & C are Siblings

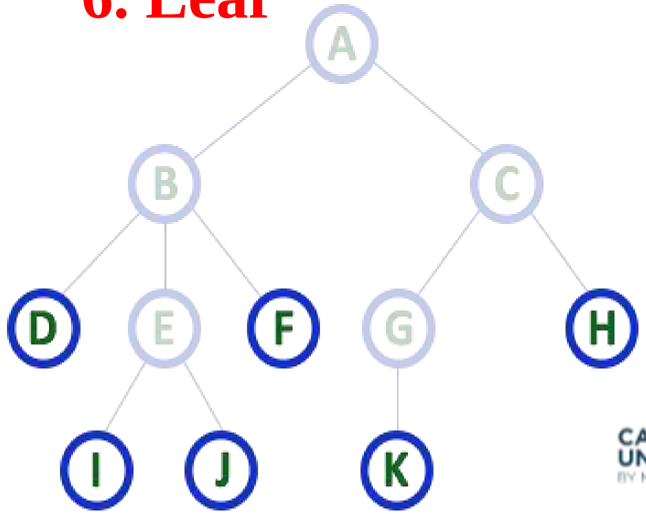
Here D E & F are Siblings

Here G & H are Siblings

Here I & J are Siblings

- In any tree the nodes which has same Parent are called 'Siblings'
- The children of a Parent are called 'Siblings'

6. Leaf

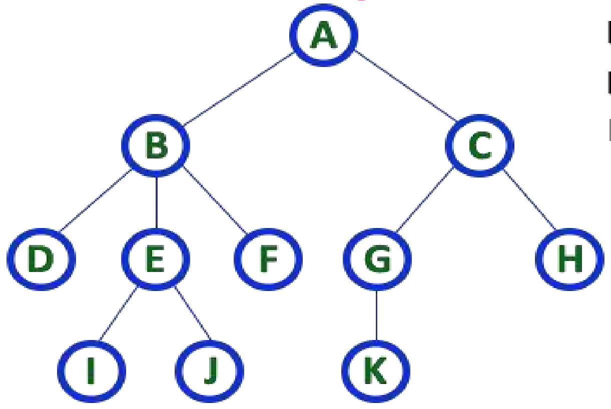


Here D, I, J, F, K & H are Leaf nodes

- In any tree the node which does not have children is called 'Leaf'
- A node without successors is called a 'leaf' node

Tree - Terminology

7. Degree



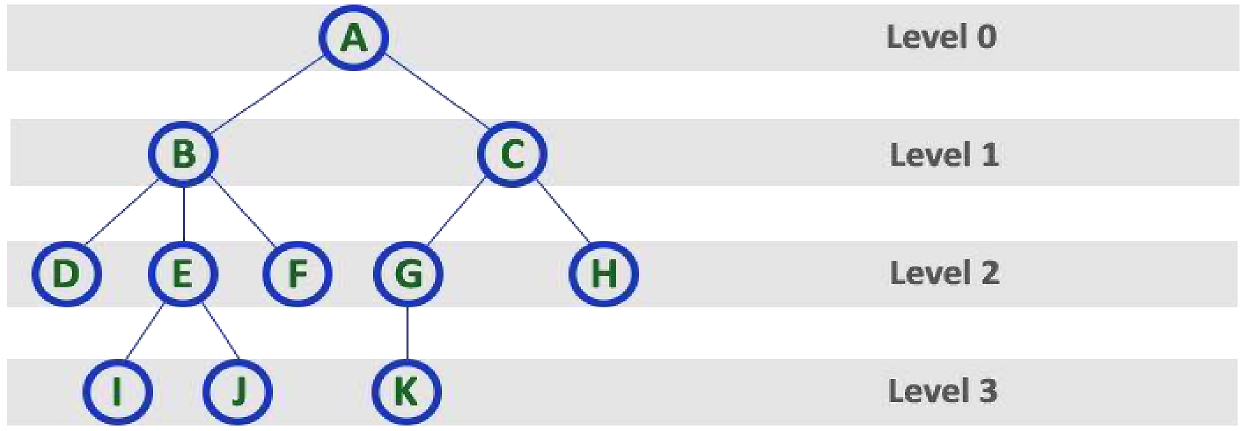
Here Degree of B is 3

Here Degree of A is 2

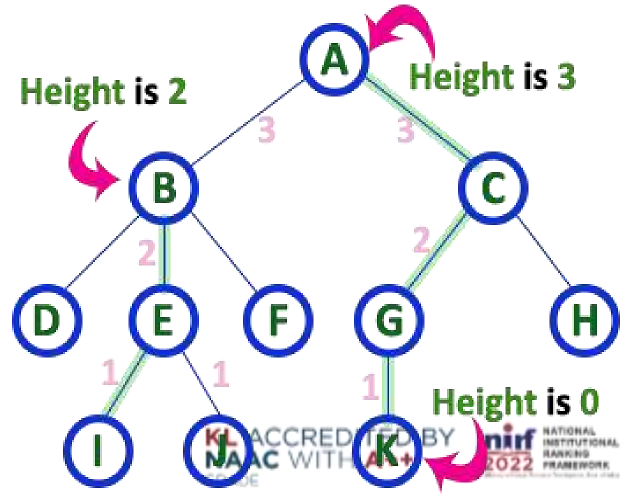
Here Degree of F is 0

- In any tree, 'Degree' of a node is total number of children it has.

8. Level



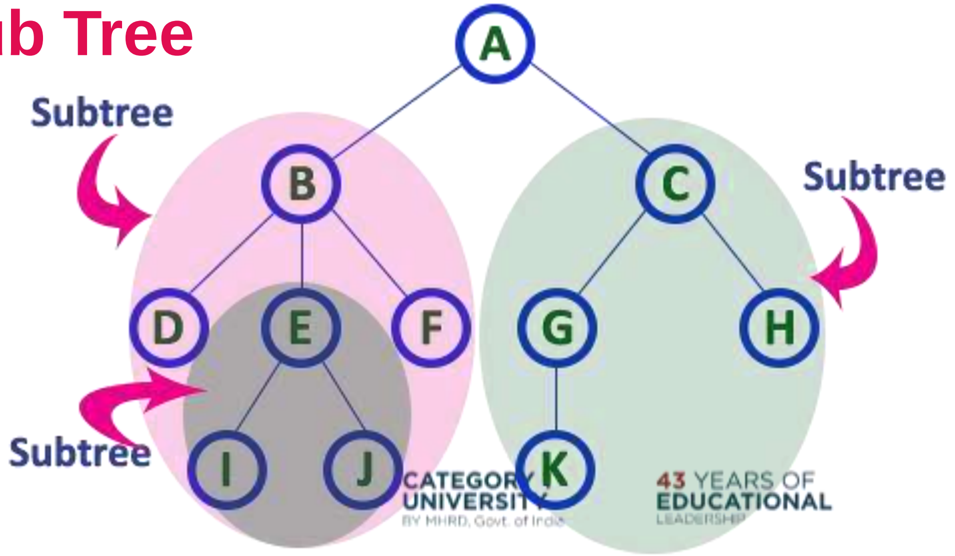
9. Height



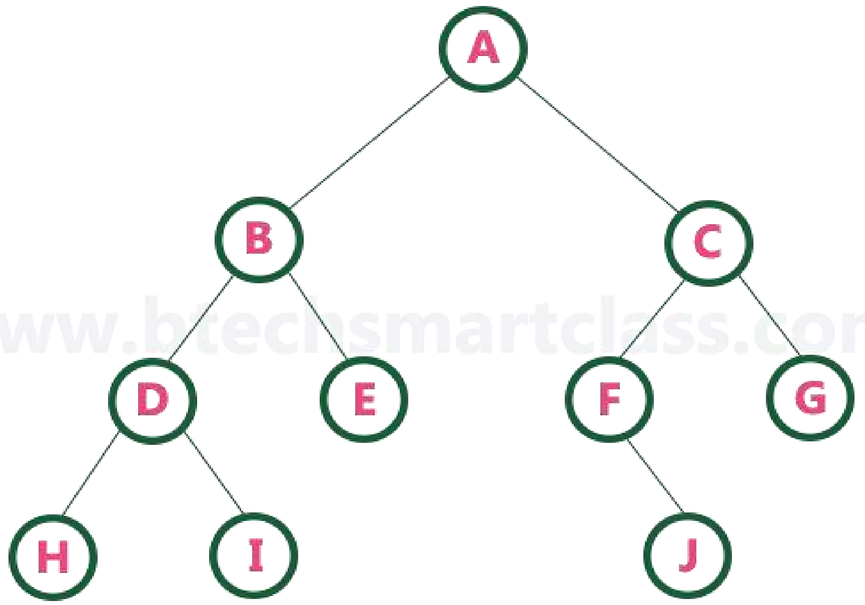
Here Height of tree is 3

- In any tree, 'Height of Node' is total number of Edges from leaf to that node in longest path.
- In any tree, 'Height of Tree' is the height of the root node.

10. Sub Tree



Example of Binary Tree with nodes



Types of Binary Tree Representations

1. Array Representation of Binary Tree
2. Linked List Representation of Binary Tree

Types of Binary Tree

1. Array Representation of Binary Tree

In array representation of a binary tree, we use one-dimensional array (1-D Array) to represent a binary tree. Consider the above example of a binary tree and it is represented as follows...

A	B	C	D	F	G	H	I	J	-	-	-	K	-	-	-	-	-	-	-	-

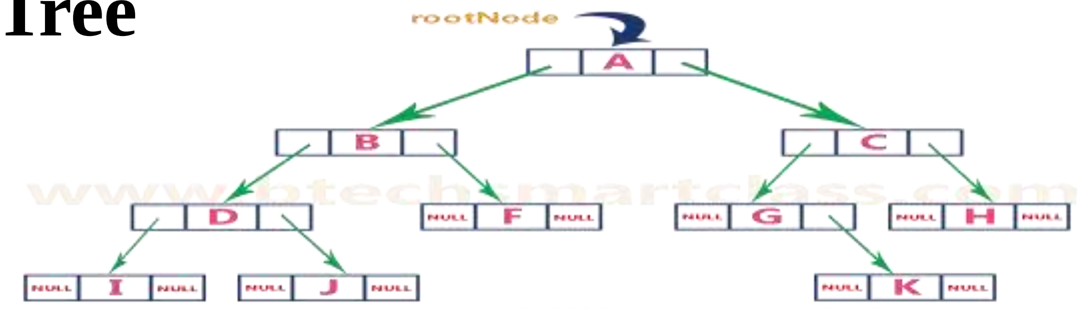
Node occupies a[i]

Left child->[2*i+1]

Right child->[2*i+2]

Parent child->[(i-1)/2] ex:55,44,66,33,10,20

2. Linked List Representation of Binary Tree



Types of binary tree traversals.

There are three types of binary tree traversals.

1.In - Order Traversal
2.Pre - Order Traversal
3.Post - Order Traversal

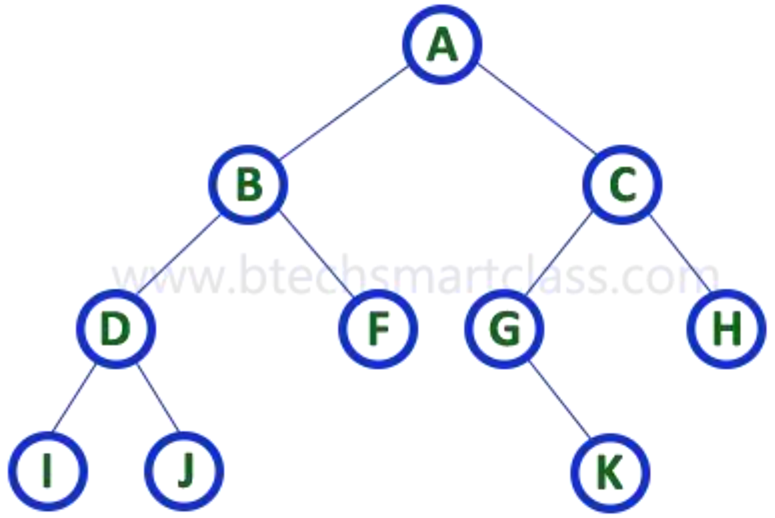
1.In - Order Traversal (leftChild - root - rightChild)

In In-Order traversal, the root node is visited between the left child and right child. In this traversal, the left child node is visited first, then the root node is visited and later we go for visiting the right child node. This in-order traversal is applicable for every root node of all subtrees in the tree. This is performed recursively for all nodes in the tree.

Example of In-Order Traversal (Cont..)

In-Order Traversal for belo example of binary tree is

I - D - J - B - F - A - G - K - C - H



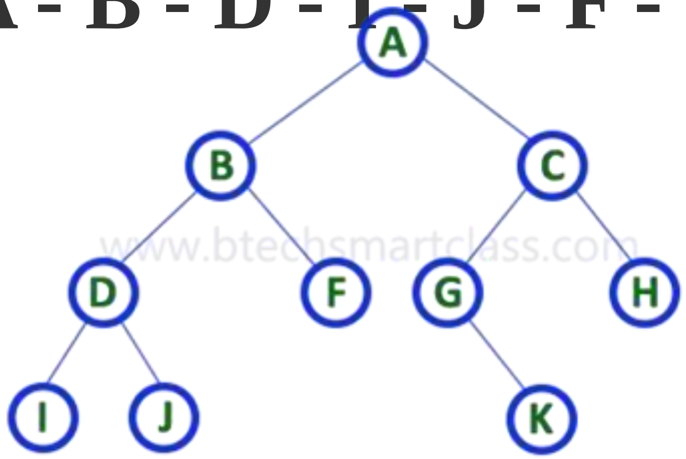
2. Pre - Order Traversal (root - leftChild - rightChild)

In Pre-Order traversal, the root node is visited before the left child and right child nodes. In this traversal, the root node is visited first, then its left child and later its right child. This pre-order traversal is applicable for every root node of all subtrees in the tree.

Example of Pre -Order Traversal

Pre-Order Traversal for below example binary tree is

A - B - D - I - J - F - C - G - K - H



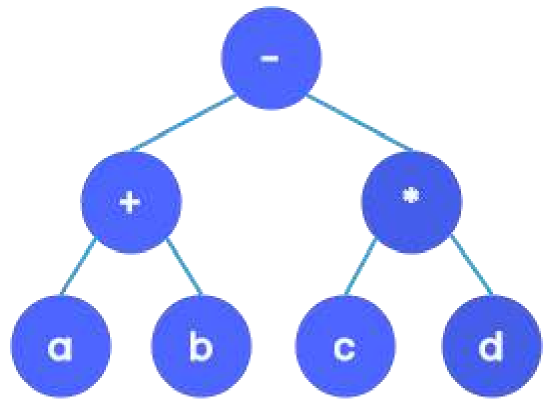
- **3. Post - Order Traversal (leftChild - rightChild - root)**

In Post-Order traversal, the root node is visited after left child and right child. In this traversal, left child node is visited first, then its right child and then its root node. This is recursively performed until the right most node is visited.

Expression Tree

An expression tree is a special type of **binary tree** that is used to store algebraic expressions.

- Consider the algebraic expression given as: X = (a + b) - (c * d). This can be represented using a binary tree as follows:

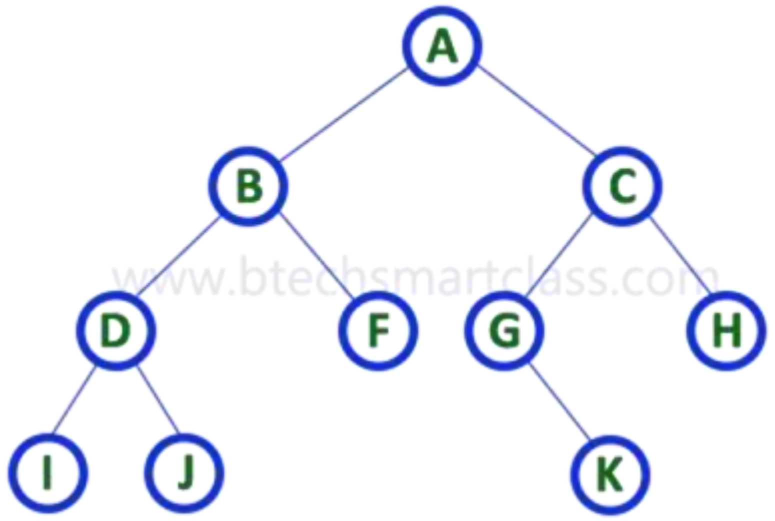


- The **in-order traversal** of the tree returns the **infix expression**.
- Similarly, the **pre-order** and **post-order** traversal of the expression tree will return **prefix** and **postfix** expression respectively.

Example of Pre -Order Traversal

Post-Order Traversal for above example binary tree is

I - J - D - F - B - K - G - H - C - A



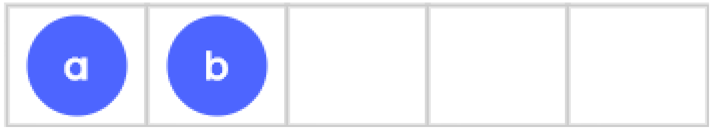
Constructing an expression tree

- An expression tree can be constructed with the help of a **stack**.
- Consider the postfix expression a b - c *

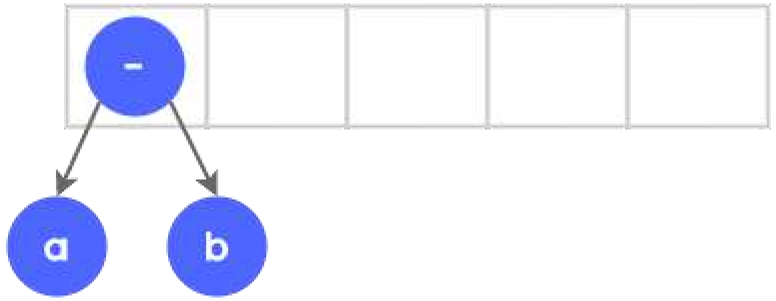
The following steps are done to create an expression tree.

- If the character is an operand, PUSH it on to the stack.
- If the character is an operator, POP two values from the stack and insert them as the child of operator. A pointer to the new root is pushed on to the stack.

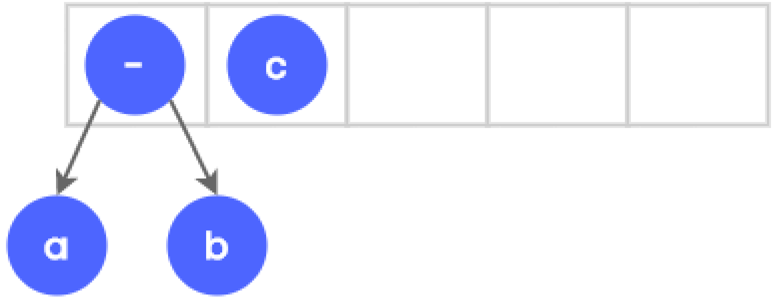
Since the first two characters are operands, PUSH it onto the stack.



When the (-)symbol is read, a and be are popped from the stack and is added as the child of node - A pointer to the new node is now pushed onto the stack.

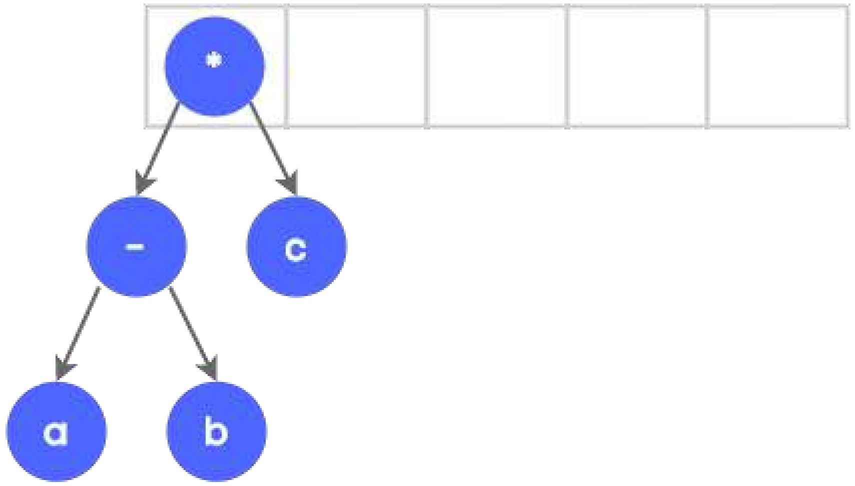


- Now c is read and is pushed onto the stack.



Finally, the operator (*) is read, so we pop c and (-) from the stack and add it as the child of node *

.



IMPORTANT FACTS RELATED TO THE SESSION

Advantages of a Binary Trees

- Binary trees are conceptually simple and easy to implement.
- Binary trees provide efficient search and retrieval operations.
- Traversing a binary tree is a well-defined process, and various algorithms (in-order, pre-order, post-order) allow for easy navigation and processing of all elements in the tree.

Disadvantages of Binary Trees

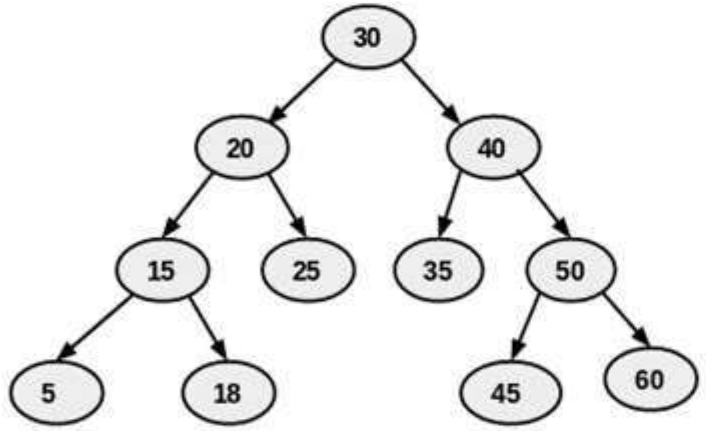
- Binary trees can have higher memory overhead compared to other data structures.
- Although binary trees are conceptually simple, ensuring proper balance and implementing self-balancing mechanisms can add complexity to the code.
- The performance of a binary search tree can be sensitive to the order in which elements are inserted.

Applications/Uses of Binary Trees in real life

- File systems in computers often use binary trees to represent the hierarchical structure of directories and file.
- Binary trees can represent organizational hierarchies in business structures
- Binary trees are used to represent mathematical expressions in compilers and evaluators

EXAMPLES

1.Write Inorder, Preorder, Post Order to below tree



2. Draw an Expression Tree to A * B + C / D.

SUMMARY

- Tree data structure is a collection of data (Node) which is organized in hierarchical structure recursively
- A tree in which every node can have a maximum of two children is called Binary Tree.
- Several terminologies are used in tree like follows..
  root, parent, child, height, level, deapth and subtree etc….
- In a Binary tree, we perform the following traversals
  i) Inorder ii) Preorder iii) post order

- An expression tree is a special type of binary tree that is used to store algebraic expressions. In an expression tree, each internal node corresponds to the operator and each leaf node corresponds to the operand.

- Advantages, Disadvantages and Applications/Uses of Binary Trees in real life.

SELF-ASSESSMENT QUESTIONS

What is a tree in data structures?

a. A hierarchical data structure

b. A linear data structure

c. A non-linear data structure

d. A sequential data structure

In a binary tree, each node has at most how many children?

a. 1
b. 2
c. 3
d. 4

TERMINAL QUESTIONS

1. Define Binary Tree?

2. Define the height of a binary tree?

3. What is an in-order traversal of a binary tree?

3. Define a leaf node in the context of a binary tree?

4. Define the term "binary tree traversal.“?

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books:

1. "Mark Allen Weiss, Data Structures and Algorithm Analysis in C, 2010 , Second Edition, Pearson Education.
2. 2. Ellis Horowitz, Fundamentals of Data Structures in C: Second Edition, 2015.
3. A.V.Aho, J. E. Hopcroft, and J. D. Ullman, “Data Structures And Algorithms”, Pearson Education, First Edition Reprint 2003.

4. Sites and Web links:

1. https://nptel.ac.in/courses/106102064
2. https://in.udacity.com/course/intro-to-algorithms--cs215
3. https://www.coursera.org/learn/data-structures?action=enroll

DEEMED TO BE UNIVERSITY

THANK YOU