Unit - III


DATA STRUCTURES
ALGORITHMS
COMPUTER NETWORKS
DATABASE SYSTEMS


#include <iostream>
using namespace std;
int main()
{
cout << "Hello, World!";
return 0;
}

U3

Linked List.

* **Linked List -**

A Linked List is a linear data structure in which elements are stored in a non-contiguous memory locations.

Each element of a Linked List is called as node, and every node contains two parts:

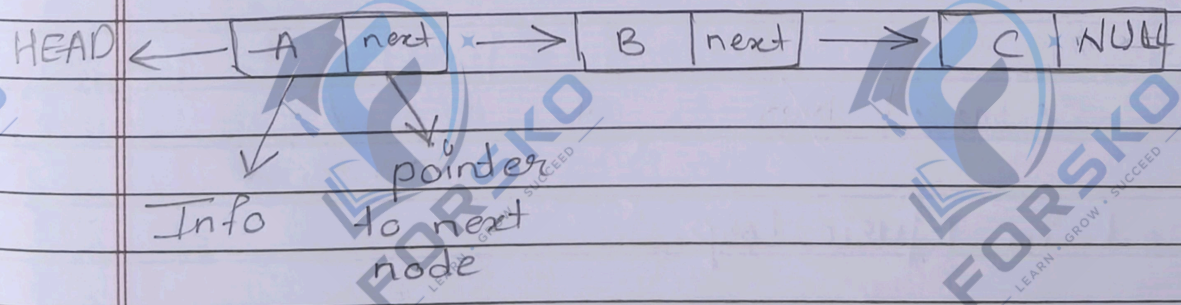
data and a Link (pointer) to the next node.

Node Structure. | Info | next |

- Data part - Stores the actual value
- Link part - stores the address of the next node.

The last node of the Linked List contains NULL in its Link part.

Representation -



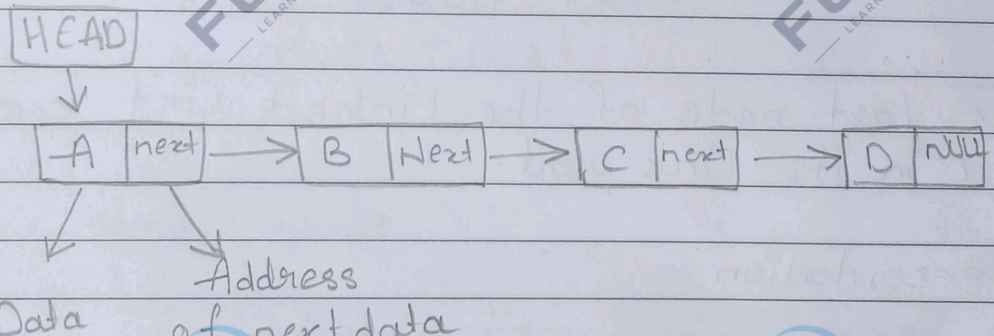
* Types of Linked List.

i) Singly Linked List.

Singly Linked List is the simplest type of Linked List in which every node contains some data and a pointer to the next node of the same data type.

The node contains a pointer to the next node means that the nodes stores the address of the next node in the squence.

A singly linked list allows the traversal of data only in one way.

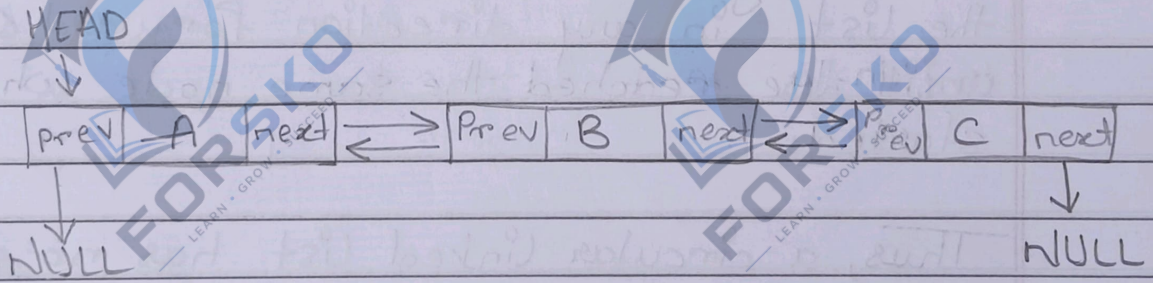


iii) Doubly Linked List -

A doubly linked list or a two way linked is a more complex type of a linked list that contains a pointer to the next as well as previous node in sequence.

Therefore, it contains three parts, a data and a pointer to the next node and a pointer to the previous node.

This would enable us to traverse the list in the backward direction as well.

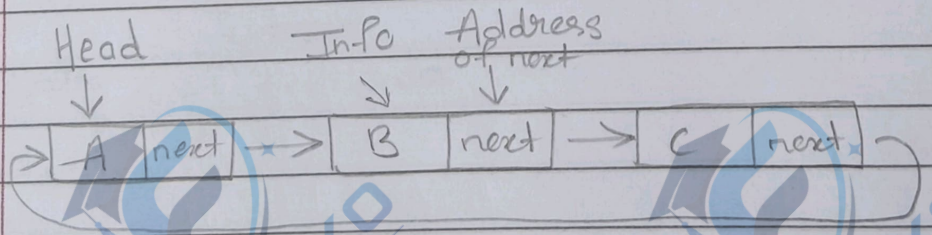


iii) Circular Linked List -

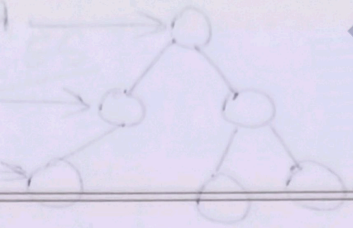
A circular linked list is a type of linked list in which the last node's next pointer points to the first node of the list, creating a circular structure. Thus it allows for continuous traversal of the list, as there is no null to end the list.

While traversing a circular linked list we can begin at any node and traverse the list in any direction forward & backward until we reached the same node where we started.

Thus, a circular linked list has no starting and no ending.



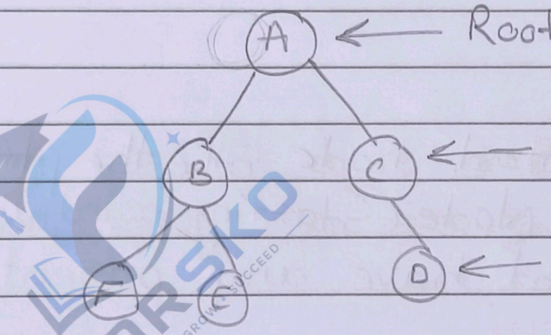
Used in Music Loop.



* Tree -

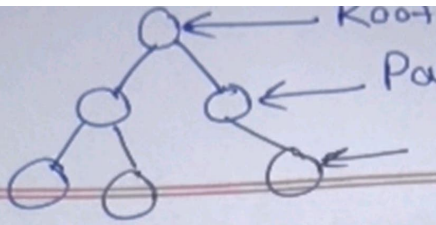
A tree is a non-linear data structure which is used to represent data in a hierarchical form / Network form.

It is consist of nodes connected by edges where one node is called the root and every other node is connected directly or indirectly to the root node.



* Tree Terminologies

T



Tree Terminologies.

The following terms are commonly used in a tree data structure to describes the relationship between nodes.

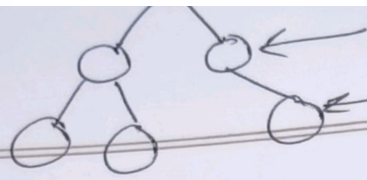
1) **Node-**
A node is a basic unit of a tree that contains data and links to its child nodes.

2) **Root Node-**
The topmost node of the tree is known as Root Node. (Starting point)
It does not have any parent node

3) **Parent Node-**
A node that has one or more child nodes is known as Parent node.

4) **Child Node-**
A node that are the deskends from parent is known as child node.

5) **Siblings-**
Nodes that have the same parents are called siblings.



Tree Terminologies.

The following terms are commonly used in a tree data structure to describes the relationship between nodes.

1) **Node-**
A node is a basic unit of a tree that contains data and links to its child nodes

2) **Root Node-**
The topmost node of the tree is known as Root Node. (Starting point)
It does not have any parent node

3) **Parent Node-**
A node that has one or more child nodes is known as Parent node.

4) **Child Node-**
A node that are the descends from parent is known as child node.

5) **Siblings-**
Nodes that have the same parents are called siblings.

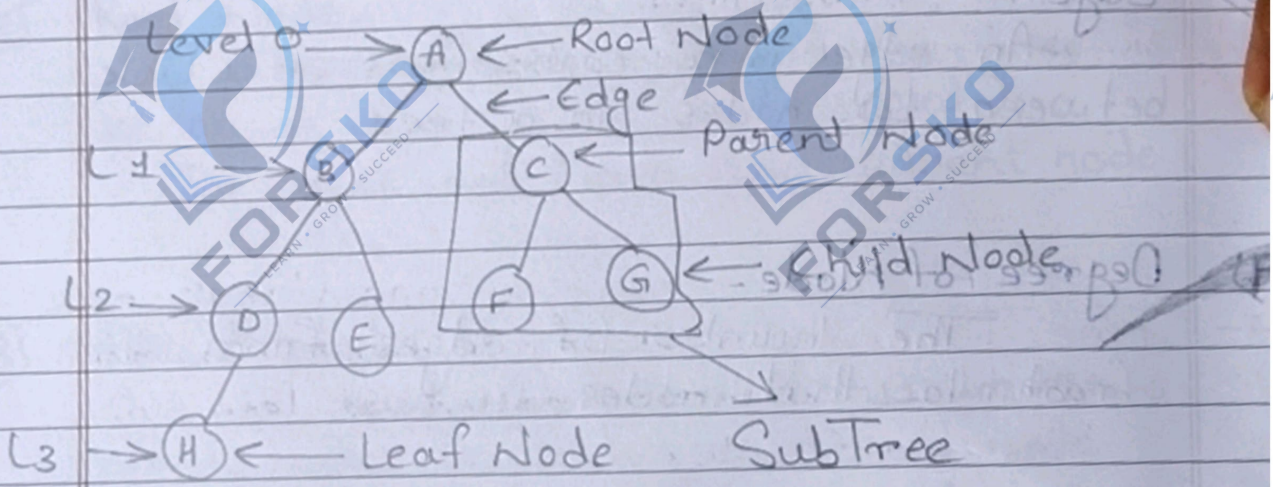
12) Level

A Level of a node represent its position in the tree.

13) Hight of Tree

The length of the longest path from root to any leaf node.

It represent the longest path from root to leaf.



Height - 4

Siblings - B, E / F G / B C

Leaf Node - H, F, G, E

Internal Node - B, C, D

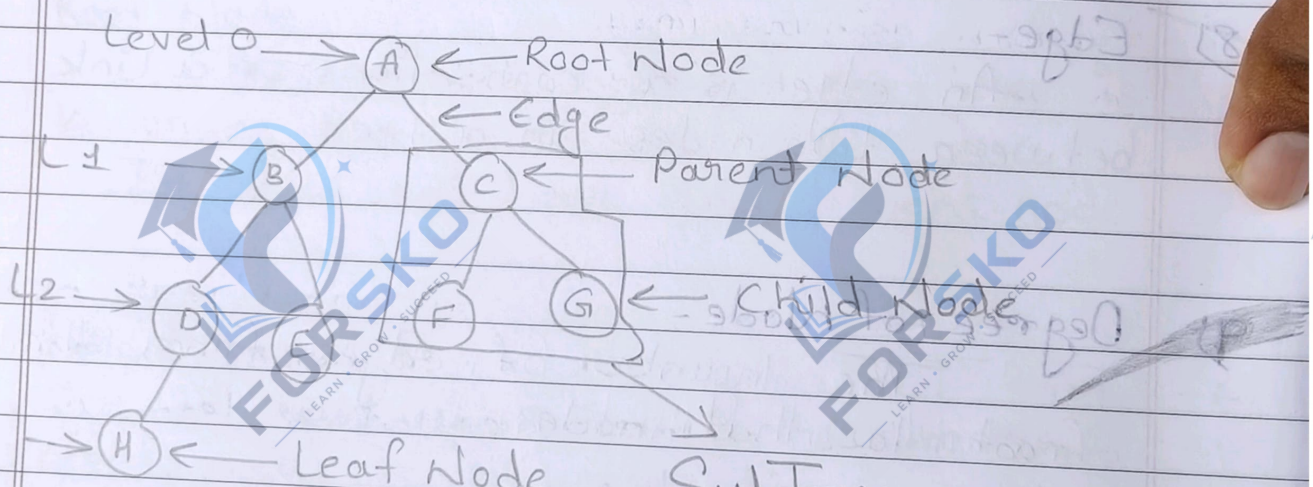
12) Level

A Level of a node represent its position in the tree.

13) Hight of Tree

The length of the longest path from root to any leaf node.

It represent the longest path from root to leaf.



Height - 4

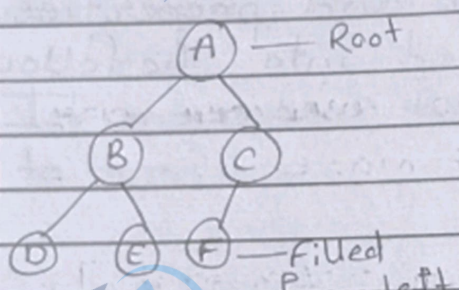
Siblings - D, E / F G / B C.

Leaf Node - H, F, G, E

Internal Node - B, C, D

ii) Complete Binary Tree =

A complete binary tree is a type of binary tree in which all levels are completely filled expect possibly the last level and the last level are filled from left to right.

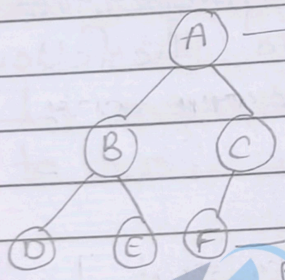


Application -

- Heap
- Priority Queue
- Heap Sort
- Implimentation of Binary Tree

ii) Complete Binary Tree -

A complete binary tree is a type of binary tree in which all levels are completely filled except possibly the last level and the last level are filled from left to right.



Application -

- Heap
- Priority Queue
- Heap Sort
- Implimentation of Binary Tree

* Tree Traversal-

Tree traversal is the process of visiting every node of a tree exactly once in a specific order to perform operations like displaying data, searching or updating values.

Following are the types of traverse

1) Preorder Traversal- Root → Left → Right
RoLeR

In preorder traversal, the root node is visited first, then the left subtree, and finally the right subtree.

Steps-
1) Visit Root
2) Traverse Left Subtree
3) Traverse Right Subtree

Eg-

Preorder -
A, B, D, E, C

Uses-
Used to copy tree.
Used for prefix expression evaluation

* Tree Traversal-

Tree traversal is the process of visiti
every node of a tree exactly once in a
specific order to perform operations (
displaying data, searching or updating v

Following are the types of traverse

1) Preorder Traversal- Root → Left → Righ
RoLeR

In preorder traversal, the root node i
visited first, then the left subtree, and
finally the right subtree.

Steps-

1) Visit Root
2) Traverse Left Subtree
3) Traverse Right Subtree

Eg- 

Preorder -
A, B, D, E, C

Uses-

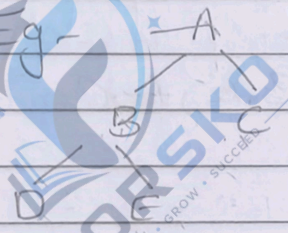
Used to copy tree.
Used for prefix expression evalution

3) Postorder Traversal - (Left → Right → Root)

In postorder traversal, the left subtree is visited first, then the right subtree, and the root node is visited last.

Steps:

1) Traverse Left Subtree
2) Traverse Right Subtree
3) Visit Root.

Eg- 

PostOrder-

D, B, E, C, A

Uses-

Used to delete a tree
Used for postfix expression evaluation