Deleting the first Node in Singly Linked List

Algorithms →

Deleted first (START)

Step1: Check for under flow

If Start = NULL, then

print Linked List Empty

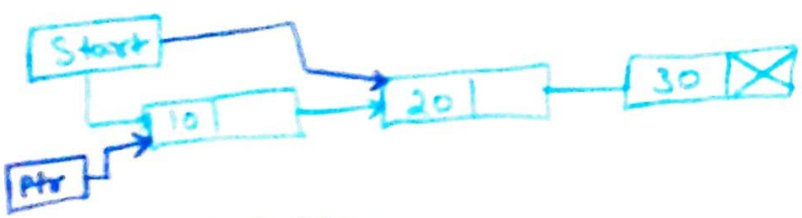
Exit

Step 2: Set PTR = START

Step 3: Set START = START → Next

Step 4 : print Element deleted is ptr → info

Step 5: free (ptr).



After deletion

Deleting the Last node in Singly Linked List

Algorithm →

Deleting (START)

Step1: Check for underflow

If Start = NULL then

print Link List is Empty

Exit

Step 2: if Start → Next = NULL then

Set Ptr = Start

Set Start = NULL

print element deleted is = PTR → Info

free (PTR)

End if

Step 3: Set PTR = START

Step 4: Repeat Step 5 and 6 untill

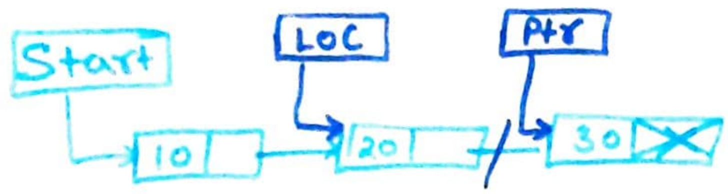
PTR → Next ! = NULL

Step 5: Set LOC = PTR

Step 6 : Set PTR = PTR → Next

Step7: Set LOC → Next = NULL

Step8: free (PTR)



After deletion

Deleting the Node from Specified Position In Singly Linked List

Algorithm →

Delete-Location (START, LOC)

Step1: Check for Underflow
 if PTR = NULL then
 print underflow
 Exit

Step2: Initialize the Counter I and pointers
 Set I = 0;
 Set ptr = Start;

Step3: Repeat step 4 to 6 until I < LOC

Step4: Set temp = PTR

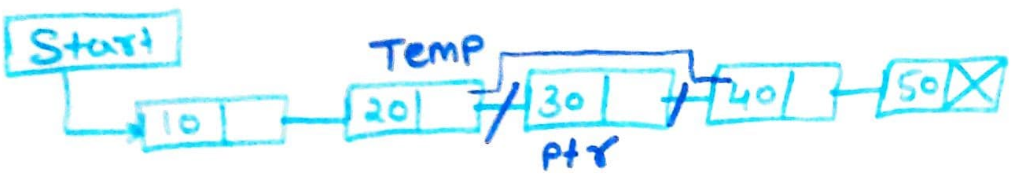
Step5: Set PTR = PTR → Next

Step6: Set I = I+1

Step 7 : print Element deleted is = ptr → info

Step 8 : set Temp → Next = ptr → Next

Step 9 : free(ptr)

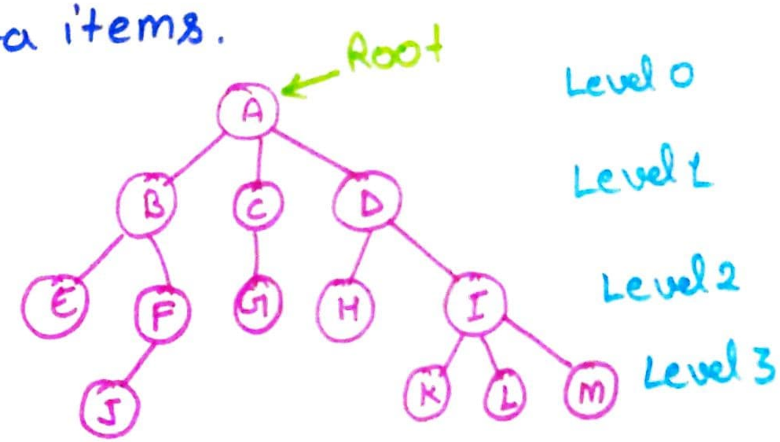


After deletion

Trees in Data Structure

Tree ⇒ • A Tree is a non-linear data Structure in which items are arranged in a sorted sequence.

- It is used to represent hierarchical relationship Existing amongst several data items.



Tree Terminology ⇒ Tree has different terminology such as :-

1> **Root** ⇒ It is specially designed data item in a tree. It is the first in the hierarchical Arrangement of data item. In the above tree, A is root item.

2> **Node** ⇒ Each data item in a tree is called a node. In the given

Tree there are 13 Node such as-
A, B, C, D, E, F, G, H, I, J, K, L, M

3⇒ **Degree of a node** ⇒ It is the no. of Subtrees of a node in a given tree:
The degree of A = 3
The degree of C = 1
The degree of L = 0

4⇒ **Degree of a tree** ⇒ It is the maximum degree of nodes in a given tree. In the given tree the Node A and node I has maximum degree (3). so the degree of tree is 3.

5⇒ **Terminal node** ⇒ A node with degree zero is Called terminal node. In given tree- E, J, G, H, K, L and M are terminal node.

6⇒ **None-terminal Node** ⇒ Any Node whose degree is not zero is called non-terminal node. In given tree - A, B, C, D, F, I are Non-terminal Node.

7⇒ **Siblings** ⇒ The Child nodes of a given parent node are Called Siblings. They are also Called brothers.
In the given table.
- B, C, D are Siblings of parent nod A.
- H & I are Siblings of parent node D.

8⇒ **Level** ⇒ The entire tree structure is Levelled in Such a way that the root node is always at level 0.

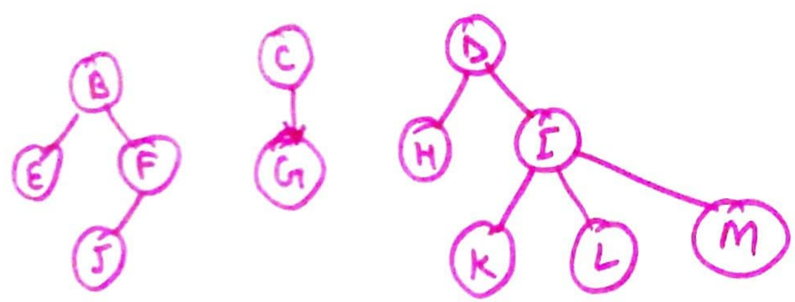
9⇒ **Edge** ⇒ It is a Connecting Line of two nodes. that is, the Line drawn from one node to another node is Called an Edge.

10⇒ **Path** ⇒ It is a sequence of Consecutive edges from the Source node to the destination node. In the given tree the path between A and J is as.
(A, B) (B, F) and (F, J)
A → B → F → J

11 ⇒ **Depth** ⇒ It is the maximum level of any node in a given tree. In the given tree, the root node A has the maximum level.

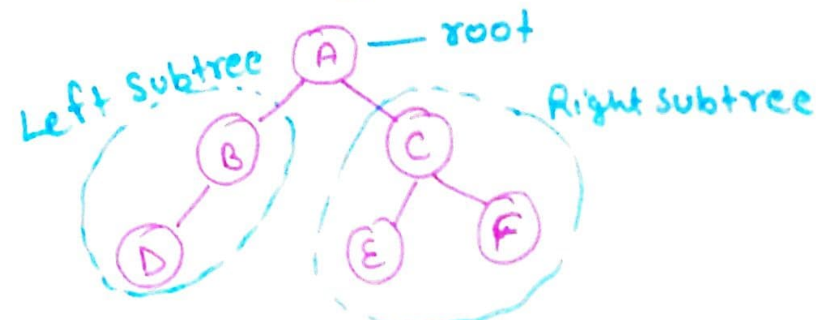
12 ⇒ **forest** ⇒ It is a set of disjoint trees. In a given tree if you remove its root node then it becomes a forest. In the given tree, there is forest with three tree. Such as.

After removing root A. forest is.



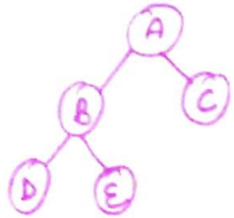
BINARY TREES

- Binary tree is a finite set of data item which is either empty of consists of a single item called root and two disjoint binary tree called the Left subtree and right subtree
- In Binary tree, every node can have maximum of 2 children which are known as Left Child and Right child

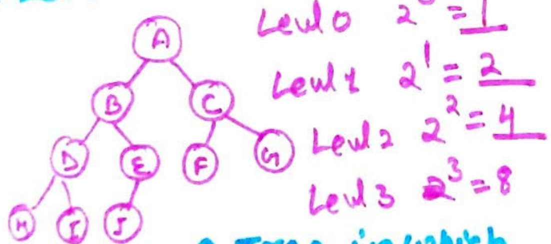


Types of Binary trees ⇒

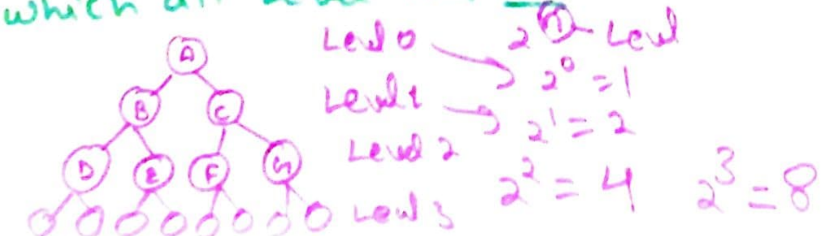
1) **full Binary tree** ⇒ A Binary Tree is full if Every node has 0 or 2 child.



2) **Complete Binary tree** ⇒ A Binary tree is Complete Binary Tree if all Levels are completely filled Except possibley the Last Level and the Last Level has all keys as Left as possible.



3) **Perfect Binary Tree** ⇒ A Tree in which all internal nodes has two children and all Leaves are at the Same Level.
in which all Level has 2^n child



Traversal of a Binary Tree

It is a way in which Each node in the tree is visited Exactly once in a Systematic manner.

There are three ways which we use to traverse a tree - Node Left, Right

1- Preorder traversal (N L R)
2- In order traversal (L N R)
3- Postorder traversal (L R N)

1→ **Preorder Traversal** ⇒ In this Traversal method, the root node (N) is visited first, then the Left subtree (L) and finally the right subtree (R).

Algorithm ⇒

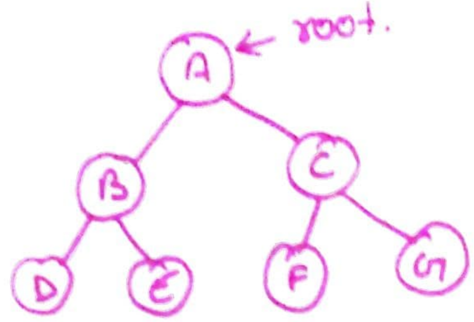
Until all nodes are traversed -

Step1: Visit root node.

Step2: Recursively traverse Left Subtree.

Step3: Recursively traverse Right Subtree.

Ex ⇒



Pre-order traversal is →

A, B, D, E, C, F, G.

2 → Inorder Traversal ⇒

In this traversal method, the Left subtree (L) is visited first, then the root (N) and later the right subtree (R).

Algorithm ⇒

Untill all nodes are traversed –

Step1: Recursively traverse Left subtree.

Step2: Visit root node.

Step3: Recursively traverse Right Subtree.

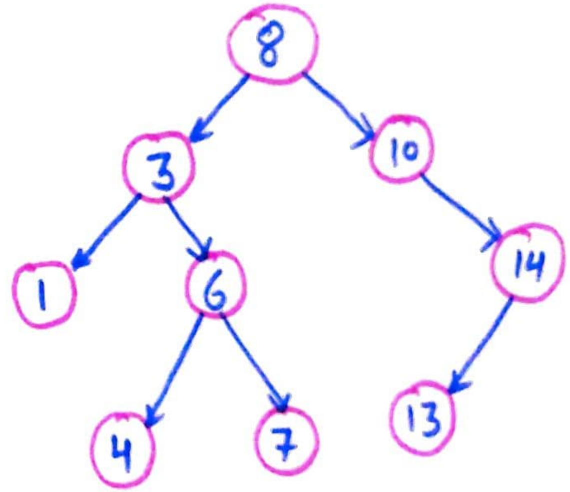
Binary Search tree (BST)

• Binary Search tree is a node-based binary tree data structure which has the following Rules:

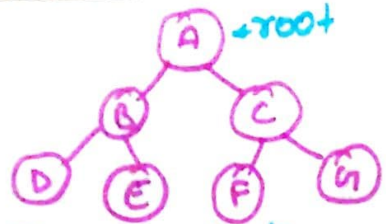
1 → The value of the key in the Left child or left subtree is less than the value of root.

2 ⇒ The value of the key in the right child or right subtree is more than or Equal to the root.

3 ⇒ The right and Left subtree Each must also be a binary search tree (BST).



Ex ⇒



Inorder Traversal is -

D, B, E, A, F, C, G.

3 ⇒ **Post-order Traversal** ⇒ In this method the root node is visited last, hence the name first we traverse Left subtree (L), then the right subtree (R) and finally the root node (N).

Algorithm ⇒

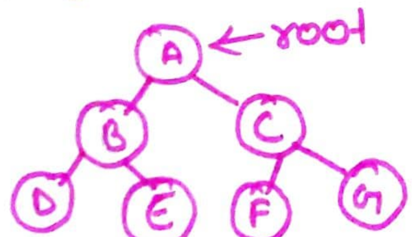
Until All nodes are traversed -

Step1: Recursively traverse Left subtree.

Step2: Recursively traverse right subtree.

Step3: Visit root node.

Ex ⇒



Post order Traversal is -

D, E, B, F G, C, A

Difference between Stack and Queue

Stack	Queue
1→ It represents the collection of elements in Last in first Out (LIFO) order.	1→ It represents the collection of elements in first In first out (FIFO) order.
2→ Objects are inserted and removed at the same end called Top of Stack (TOS).	2→ Objects are inserted and removed from different ends called front and rear ends.
3→ Insert operation is called push operation.	3→ Insert operation is called Enqueue operation.
4→ Delete operation is called pop operation	4→ Delete operation is called Dequeue operation.
5→ In Stack There is no wastage of memory space.	5→ In Queue there is a wastage of memory space.
6→ plate Counter at marriage Reception is an Example of Stack.	6→ Students Standing in a line at fees Counter is an Example of Queue.

Difference between Singly and Doubly Linked List

Singly Linked List	doubly Linked List
1 ⇒ Singly Linked List has nodes with data fied and next Link field (forward link). &⇒ | Data | next |	1 ⇒ Doubly Linked List has nodes with data field and two pointer field. (Backword and forward Link). &⇒ | Previous | Data | Next |
2 ⇒ It allows traversal only in one way.	2 ⇒ It allows a two way traversal.
3 ⇒ It requires one List poiter variable (Start)	3 ⇒ It requires two List pointer variable (Start and Last).
4 ⇒ It occupies Less memory	4 ⇒ It occupies more memory.
5 ⇒ Complexity of Insertion and Deletion at known position is $O(n)$	5 ⇒ Complexity of Insertion and Deletion at known position is $O(1)$.

Difference between Linear and Non-Linear data Structure

Linear data Structure	Non-Linear data Structure
1→ In this data structure The Elements are organized in a sequence such as ÷	1→ In this data structure data is organized without any sequence.
E→ Array, stack, queue etc.	E→ Tree, Graph etc.
2→ In Linear data Structure Single Level is involved.	2→ In non-Linear D.S multiple Levels are involved.
3→ It is Easy to implement.	3→ It is difficult to implement.
4→ Data Elements can be traversed in a single Run only.	4→ Data Elements can't be traversed in a single Run only.
5→ Memory is not utilized in a efficient way.	5→ memory ~~is~~ utilization in an efficient way.
6→ Applications of Linear D.S are mainly in Application software development.	6→ Applications of non-Linear D.S are in Artificial Intelligence and image processing.

Difference between Array and Linked List

Array	Linked-List
1→ Size of an Array is fixed	1→ Size of a List is not fixed.
2→ Array is a Collection of Homogeneous (Similar) data type.	2→ Linked-List is a Collection of node (data & address)
3→ Memory is allocated from Stack.	3→ Memory is allocated from heap.
4→ Array work with Static data Structure.	4→ Linked-List work with Dynamic data Structure.
5→ Elements are Stored in Contiguous memory Locations.	5→ Elements Can be Stored anywhere in the memory.
6→ Array Elements are independent to Each other.	6→ Linked List Elements are depend to Each other.
7→ Array take more time. (Insertion & Deletion)	7→ Linked-List take Less time. (Insertion & Deletion)

Difference between Tree and Graph

Tree	Graph
1→ Tree is a Collection of nodes and Edges. Ex→ T = {node, Edges}	1→ Graph is a collection of vertices/nodes and Edges. Ex→ G = {V, E}
2→ There is a unique node called root in tree.	2→ There is no unique node.
3→ There will not be any Cycle/Loops.	3→ There can be loops/Cycle.
4→ Represents data in The form of a tree structure, in a hierarchical manner	4→ Represents data Similar to a network.
5→ In tree only one path between two nodes.	5→ In Graph one or more then one path between two nodes.
6→ In this Preorder, Inorder and Postorder Traversal.	6→ In this BFS and DFS traversal.
Ex→ 	Ex→