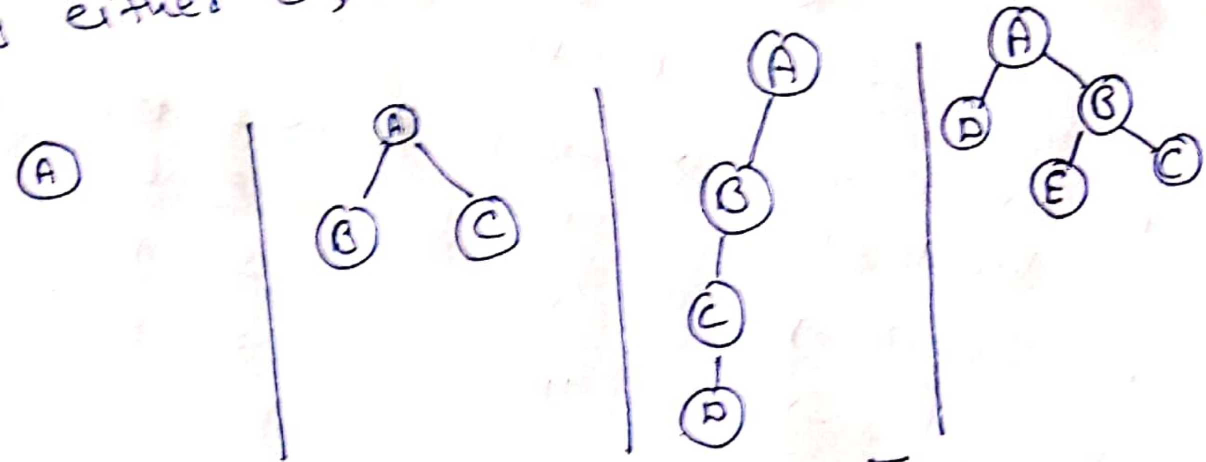
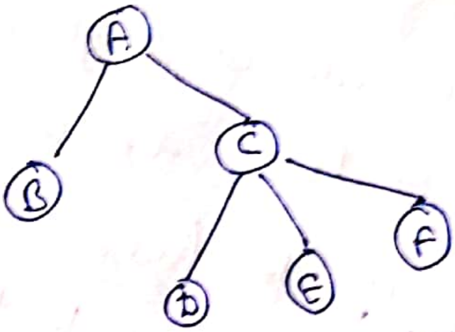
Introduction to Binary Tree

Binary Tree is a tree in which each node has atmost 2 children. i.e. each node has either 0, 1 or 2 child.

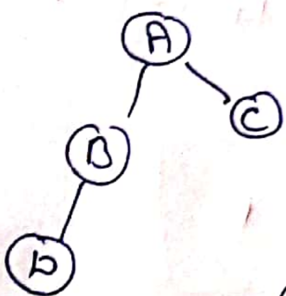
eg.

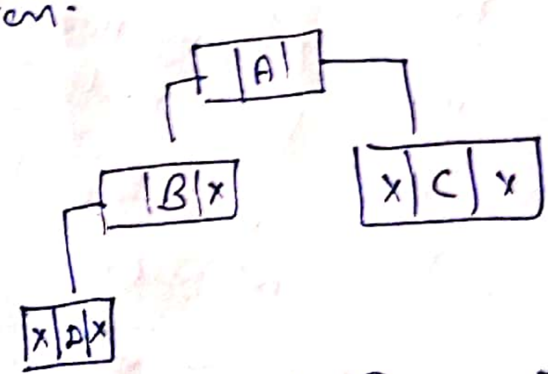


all these are examples of Binary Tree.

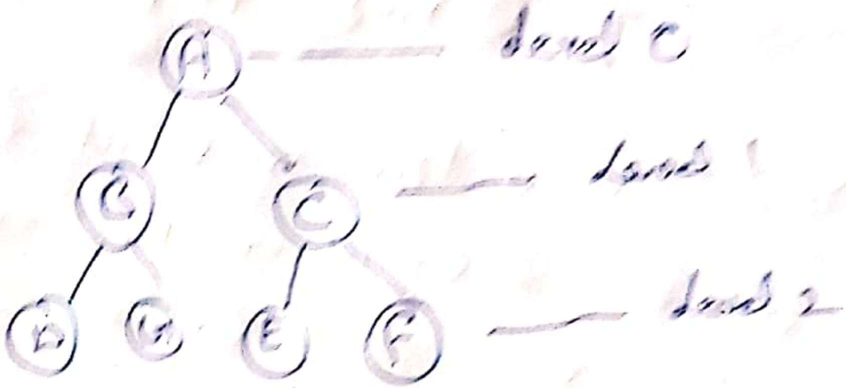


Not a Binary Tree since node 'C' has three children.





Representation of Binary Tree.



fig(i)

At level 0 → max. no. of possible nodes = 1 = 2^0

At level 1 → " = 2 = 2^1

At level 2 → " = 4 = 2^2

At level 3 → " = 8 = 2^3

and so on.

① At each level i, the max. no. of possible nodes = 2^i

② The max. no. of nodes at height (h) =

(i) height of tree = height of root node

eg. In fig(i), the height = 2

height of tree is equal to the longest path from root node to leaf node.

Similarly height of any node is equal to longest path from that node to leaf node.

In fig(i), ~~height~~ max. no. of nodes at height 2 = 1+2+4 = 7

Similarly $2^0+2^1+2^2+ \cdots 2^h = \boxed{2^{h+1}-1}$

Max. no. of node at height (h)

Scanned with CamScanner

Min. no. of nodes at height h = $(h+1)$

eg. if height = 0 ⇒ tree will be A

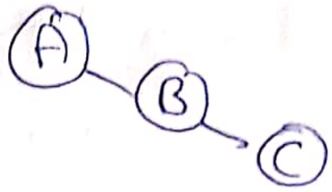
⇒ min. no. of node = 1

if height = 1

min. no. of node = 2

if height = 2

or



min. no. of nodes = 3

min. no. of nodes at height h = $\boxed{h+1}$

Similarly min. no. of nodes

Suppose we have given 'n' max. no. of nodes in the tree. The how to find possible height?

Since max. no. of nodes = n

we know if height of a tree is h then max. no. of node = $2^{h+1} - 1$

So. $2^{h+1} - 1 = n$

$2^{h+1} = n+1$

$h+1 = \log_2 (n+1)$

$$\boxed{h = \log_2 (n+1) - 1}$$

So if max. nodes are 'n' then height = $\lceil \log_2 (n+1) - 1 \rceil$

⇓

(Min. height)

Case ii) if min. no. of nodes are 'n'

we know min. no. of nodes at height $h = h+1$

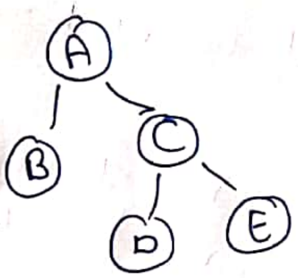
So $n = h+1$

$\boxed{h = n-1}$ → Max. height.

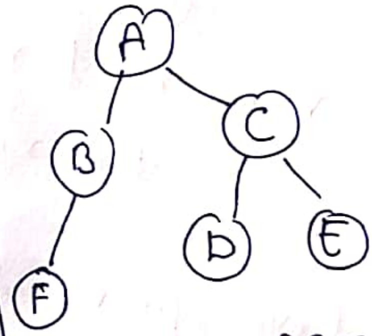
Types of Binary Tree

① **Full Binary Tree (FBT)** : It is a binary tree where each node contains either 0 or 2 children. or each node contain exactly 2 children except leaf nodes.

eg



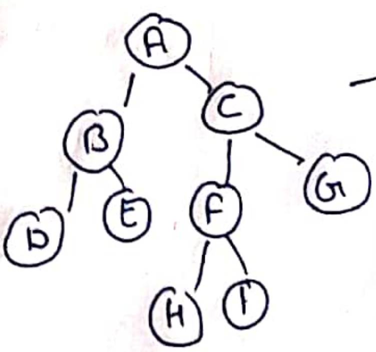
FBT



Not a FBT

No. of leaf node = no. of internal nodes +1

No. of leaf nodes = 3
No. of internal node = 2 which are A, C



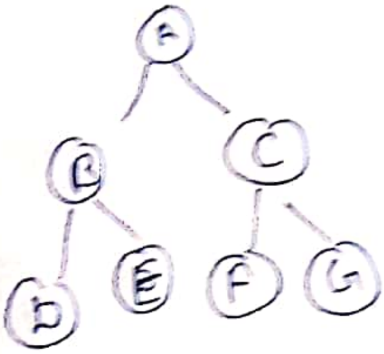
→ No. of leaf nodes = 5
No. of internal node = 4 i.e. A, B, C, F

Scanned with CamScanner

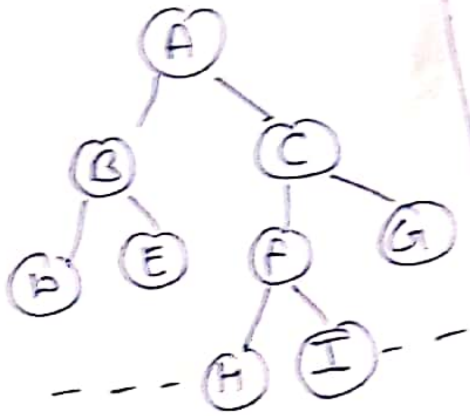
Complete Binary Tree (CBT)

(i) All the levels are completely filled (except possibly the last level)

(ii) The last level has nodes as left as possible. i.e. fill the last level from left to right.



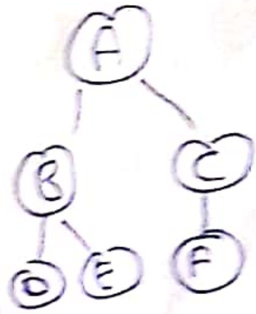
CBT



Not CBT

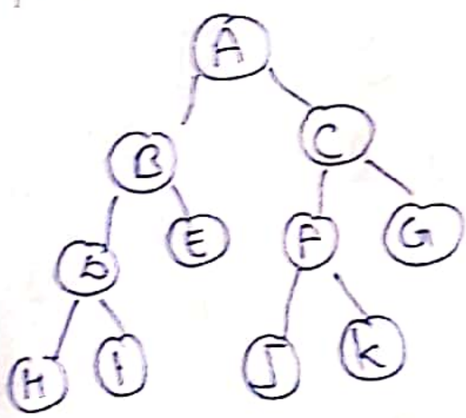
↓

Since elements are to be filled from left (last line)



CBT

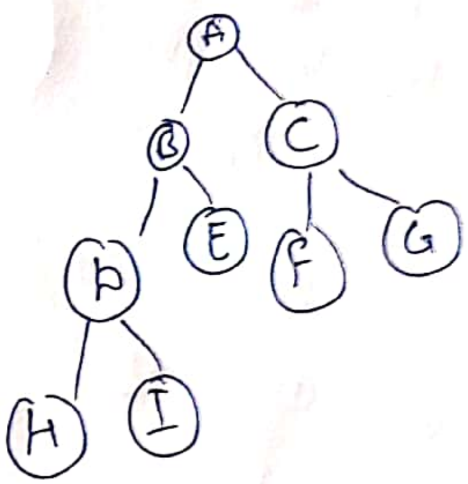
if any node at last level has one child then it should be from left to right.



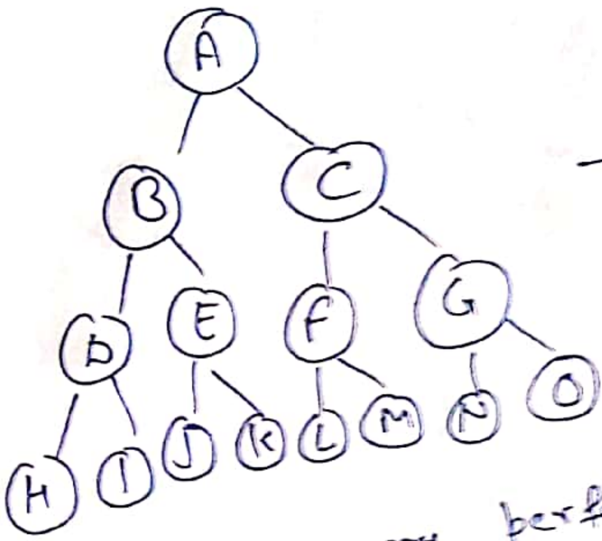
NOT CBT

Perfect Binary Tree

(i) All internal nodes have 2 children
(ii) All leaves are at same level.



→ In this tree condition (i) is satisfied but condition (ii) is not satisfied. (verify it).



→ Perfect Binary Tree

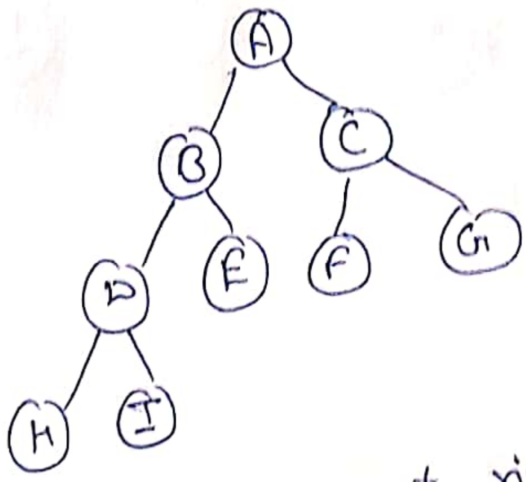
So every perfect Binary tree is Complete as well as full Binary Tree. But vice versa is not true.

	Max Nodes	Min. Nodes
Binary Tree	$2^{h+1}-1$	$h+1$
Full Binary Tree	$2^{h+1}-1$	$2h+1$
Complete Binary Tree	$2^{h+1}-1$	2^{h}

	~~Max~~ Min Height	Max. height
Binary Tree	$\lceil \log_2(n+1)-1 \rceil$	$n-1$
Full Binary Tree	$\lceil \log_2(n+1)-1 \rceil$	$\frac{(n-1)}{2}$
Complete Binary Tree	$\lceil \log_2(n+1)-1 \rceil$	$\log n$

Binary Tree Traversal

(i) Inorder Traversal
(ii) Pre order Traversal
(III) Post order Traversal



Inorder Traversal → (Left, root, right)

H D I B E A F C G

Pre order Traversal (Root, left, right)

A B D H I E C F G

Post order Traversal (Left, right, root)

H I D E B F G C A