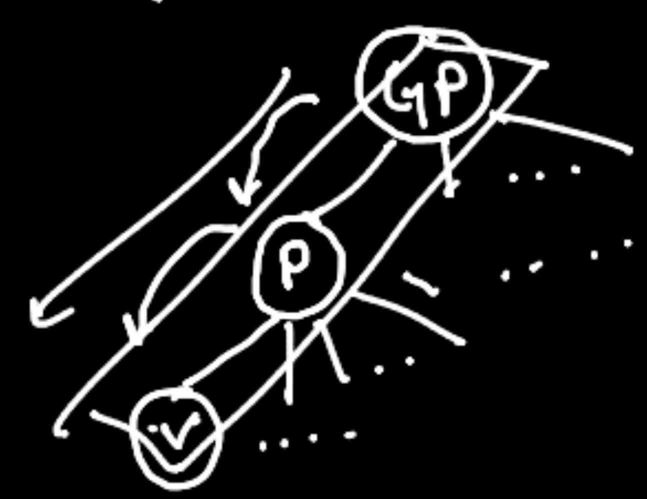
= Tree = non-linear

= seq. containers

= Linear → arrays, LL, vectors (dyn. arrays), stacks, queues

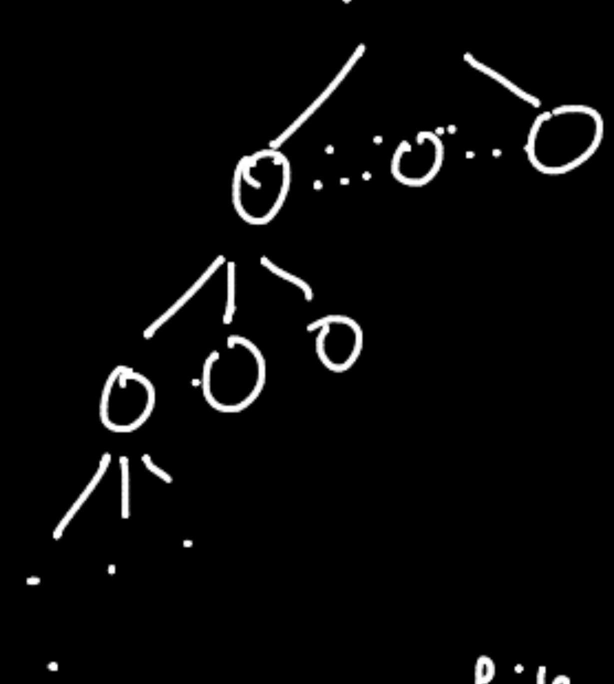
hierarchical info.



family tree

President
↓
Vice Pres

root



file system

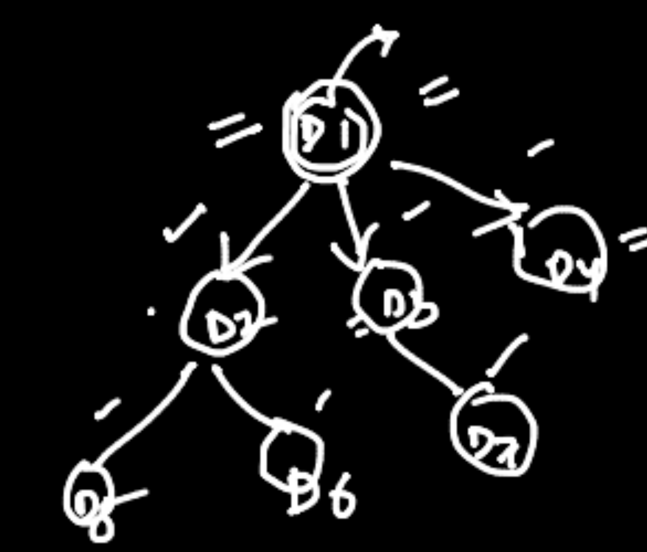
Linked List

Tree → Node =

data

link

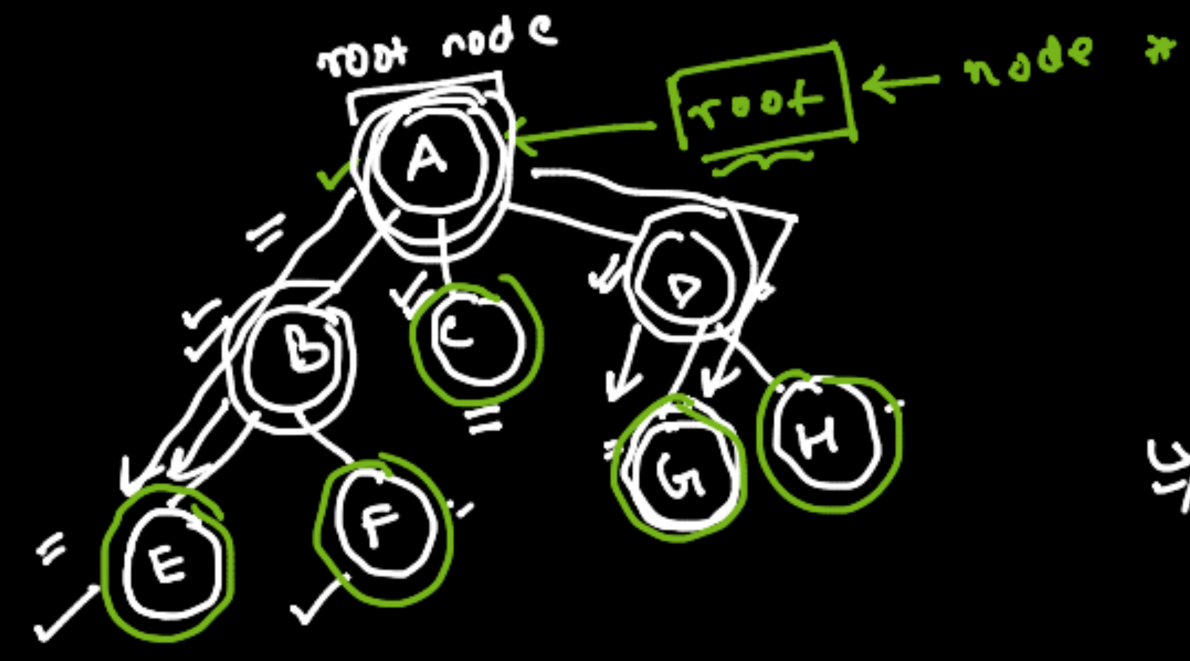
0 or more links



Trees → each is node is created dynamically.

Tree terminology

unidirectional



⇒ B, C, and D are $\boxed{\text{children}}$ of A

or

A is the $\boxed{\text{parent}}$ of B, C, D

B, C, and D are $\boxed{\text{siblings}}$

for any X and Y if there exists a path from X to Y ⇒ X is an $\boxed{\text{ancestor}}$ of Y

or

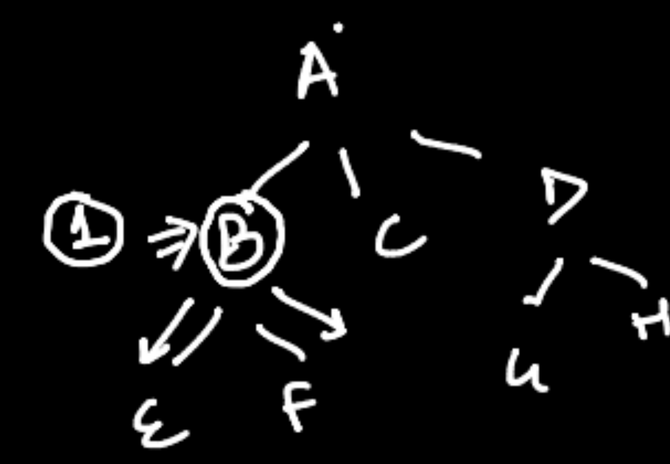
Y is a descendant of X

Node w/o children ⇒ leaf nodes

Property

⇒ height of a node

⇒ length of the $\boxed{\text{longest}}$ ⇒ max edges path from the node to a leaf node

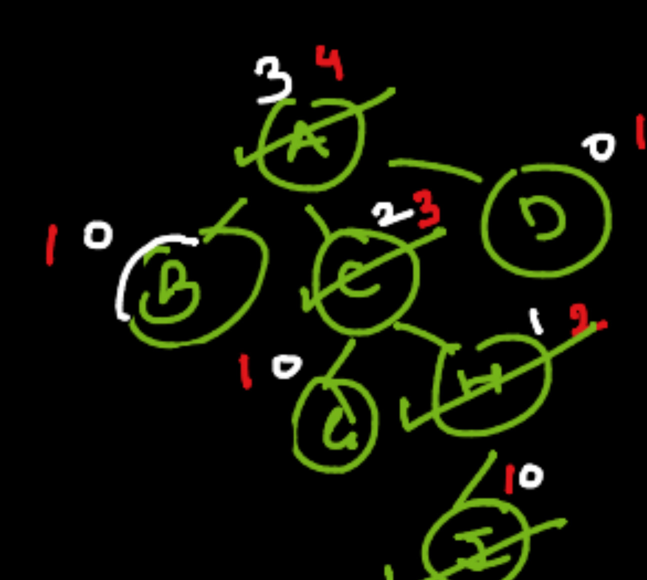


Hgt ⇒ Hgt of root node of tree

empty tree ⇒ no nodes

root = NULL

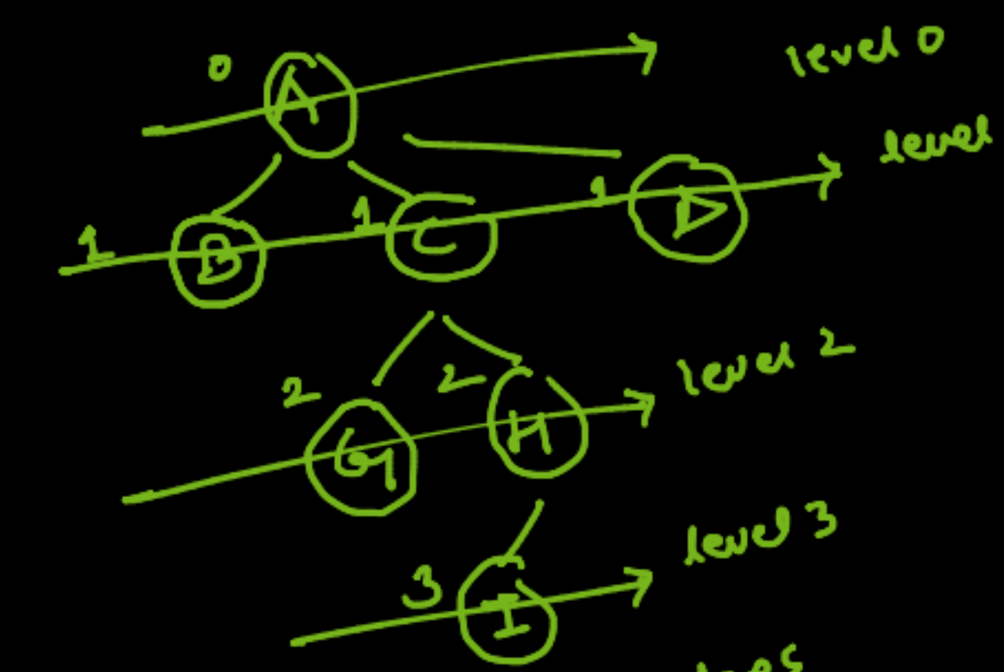
h(NULL) = −1



empty tree hgt = 0

⇒ $\boxed{\text{CLRS}}$ → Cormen

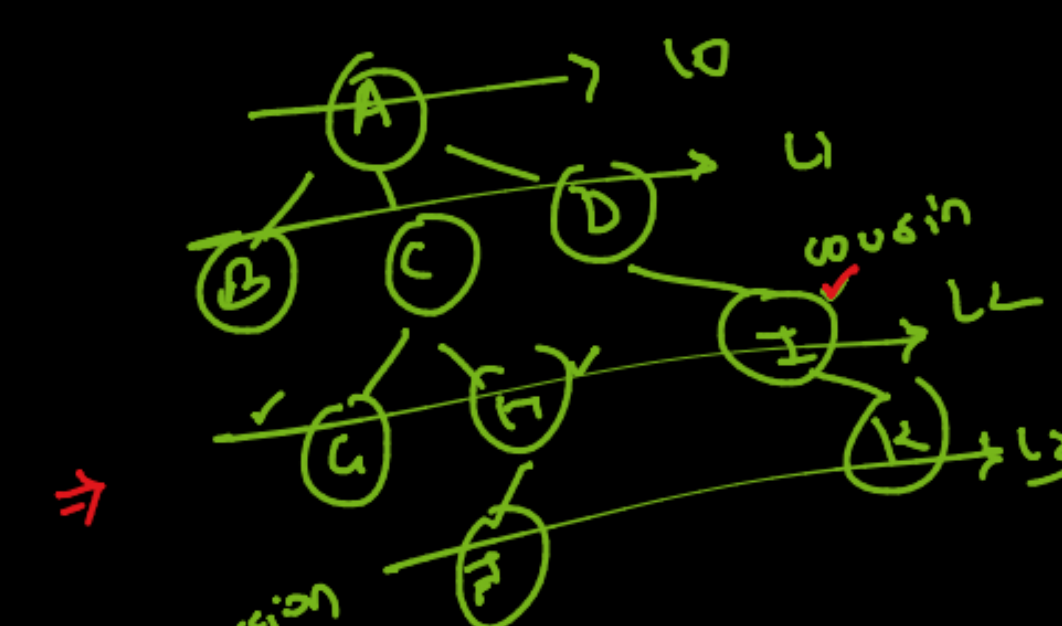
⇒ wikipedia } 90%



⇒ Depth of a node

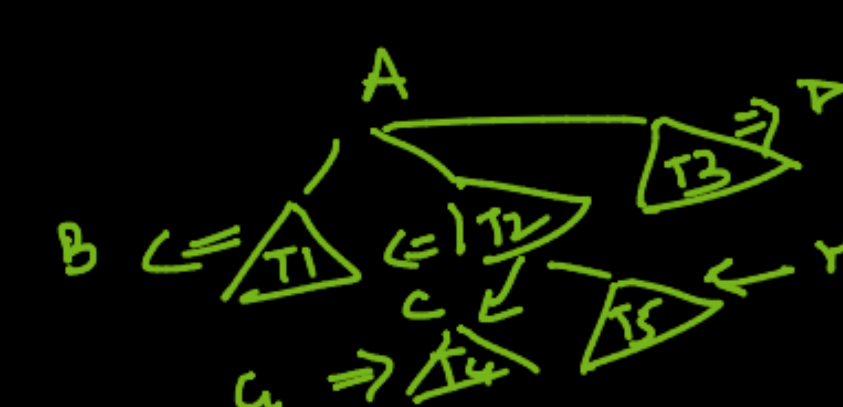
⇒ length of the path from node to the root

edges



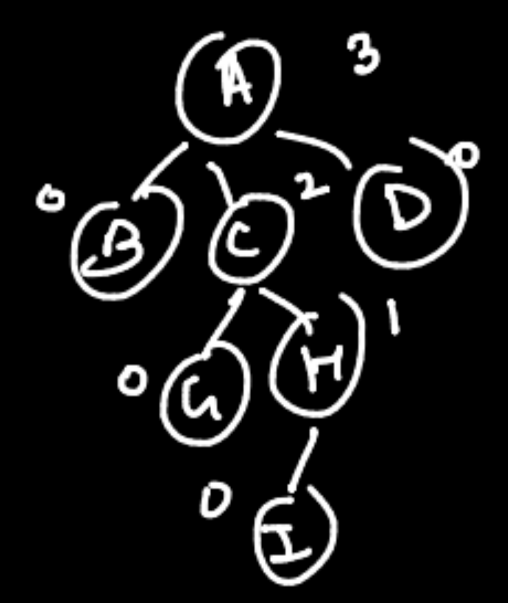
90-95% ⇒ recursion

Tree → recursive datastructure

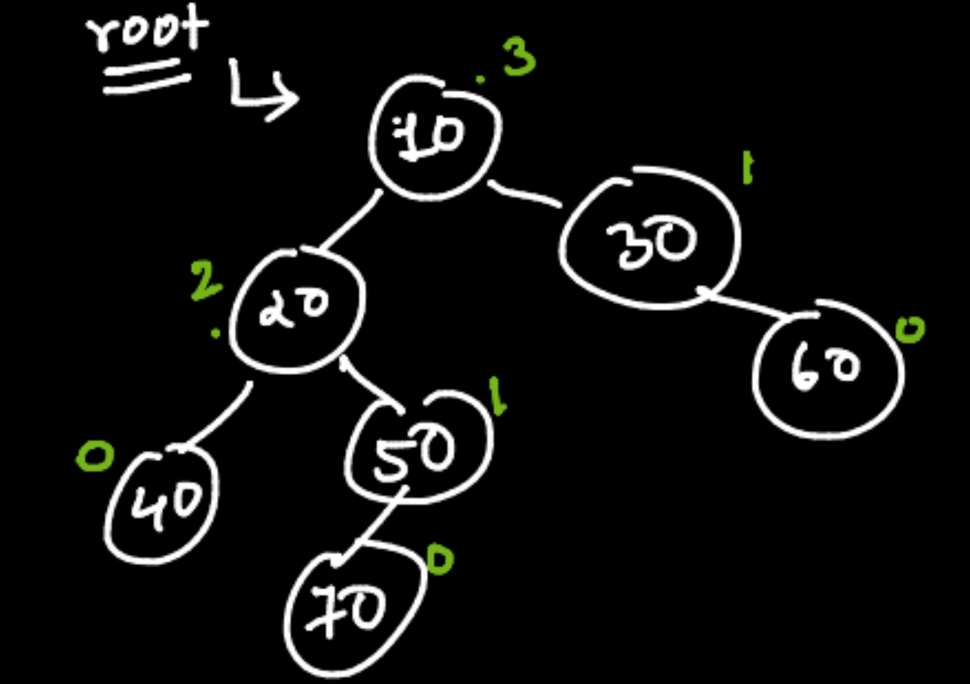


Generic Trees

↳ each node has 0 or more child nodes



root ↳



tree traversal ⇒ visit all the nodes in the tree $\boxed{\text{exactly once}}$

Binary Tree

↳ each node has atmost 2 children ≤ 2

0 or 1 or 2

node of BT

↳ data → store value

↳ left → ptr to left node or left child

↳ right → ptr to right node or right child

0

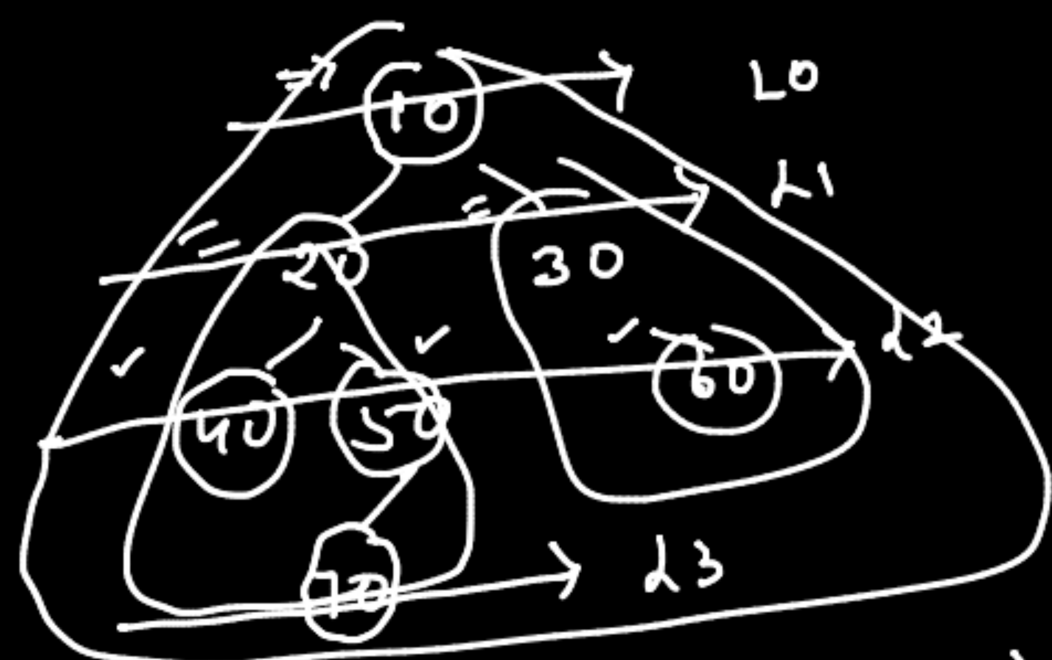
⇒ Breadth First

⇒ Depth First

Graph =

↓

Tree (special case)



↳ 10
↳ 20 30
↳ 40 50 60
↳ 70

⇒ Breadth First ⇒ How? (Next class)

⇓

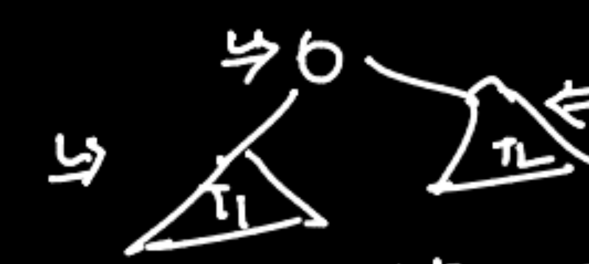
Level Order Traversal

O/P: 10 20 30 40 50 60 70

⇒ for root. Before visiting GC, we visit all the children

root, left subtree, right subtree

↳ pre-order : root <left> <right>



↳ in-order : <left> root <right>

↳ post-order : <left><right> root

Depth-First

for the root node, once you visit one of its child you'll first completely traverse the sub-tree rooted by that child before going to the other child of root.

root

root

root <left> <right>

Pre-order root <left> <right>

10 [20 [40 -1 -1] [50 [70 -1 -1] [-1]]] [30 [-1] [60 -1 -1]]

root left right

In-order

<left> root <right>

left: [-1 40 -1 20 -1 70 -1 50 -1]

10 [-1 30 -1 60 -1]

right

Tree ⇒ n nodes

↳ pre : root <left> <right> (rec, rec)

↳ in : <left> root <right> (rec, rec)

↳ post : <left> <right> root (rec, rec)

Post order

⇒ <left> <right> root

⇒ -1 -1 40 -1 -1 70 -1 50 20

left

-1 -1 -1 60 30 10

right root