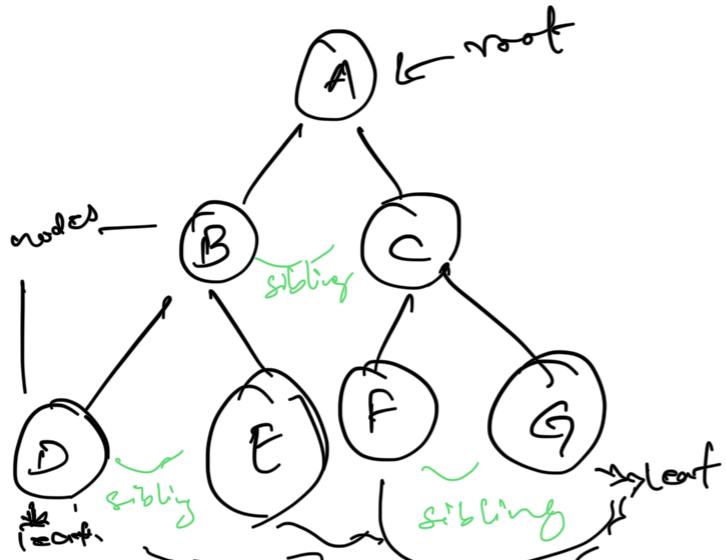
Tree Traversal

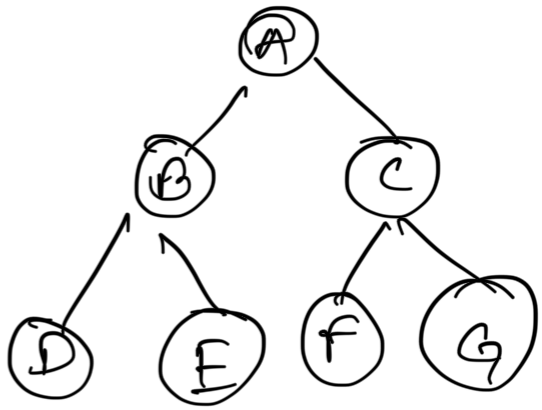


In PreOrder :-

ABDE CFE

Here, root → Parent 1
↓
leaves
↓
Parent 2
↓
leaves

(R P1 S1 S2 P2 S1 S2)

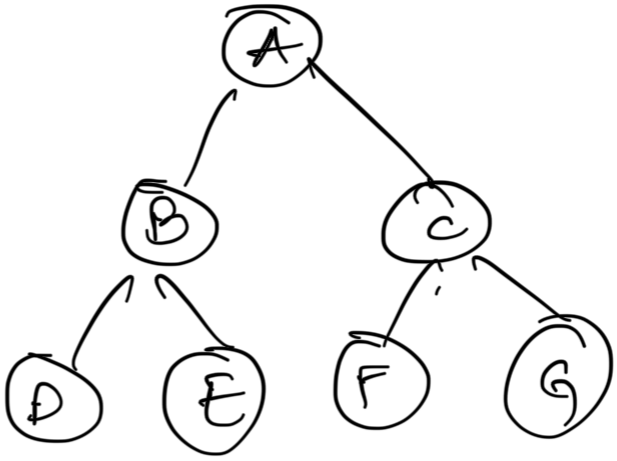


(LPLRLPL)

InOrder :-

DBE AFCG

Here, right most leaf → parent 1 → left most of P1 → root
↓
right most of P2
↓
P2
↓
left most of P2



(LLLP LLPR)

In Post Order :

DEBFGCA

leaves of P1 → P1 → leaves of P2 → P2 → root
(r, l) (r, l)

3 - Types of Binary tree :-

① Standard Binary tree :

1 Parent , 0 - 2 children/leaves

② Binary Search tree :

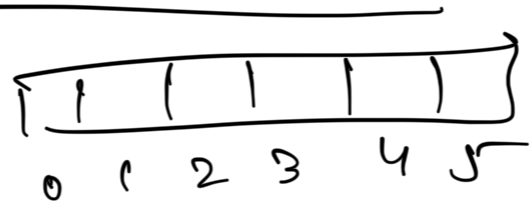
1 Parent , 0 - 2 children

left - side node always smaller or equal than parent & right side - bigger

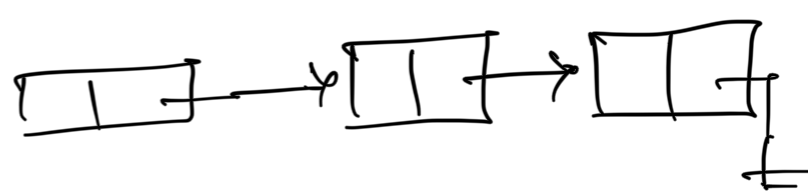
③ AVL trees : self balancing tree

Linear Data Structures :

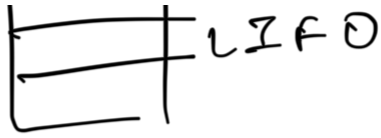
Array :



Linked List :



Stack :



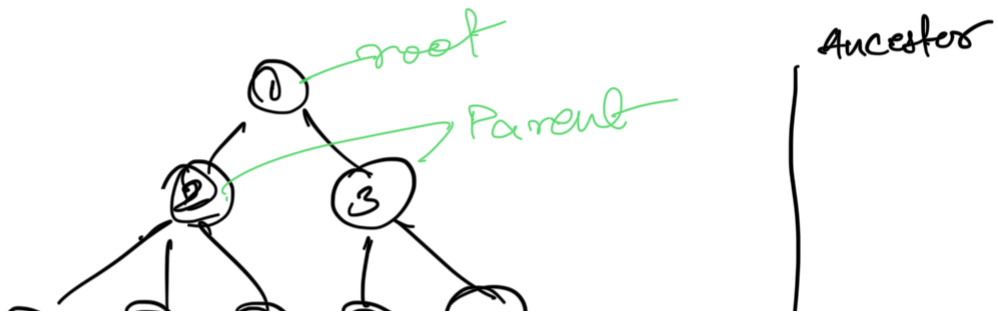
Queue:

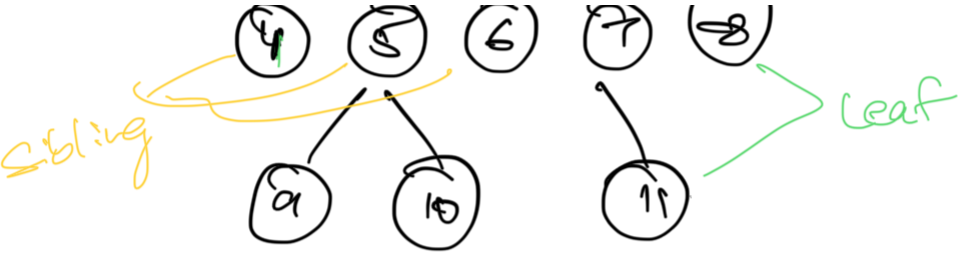
→ How do I decide which D.S. to use?

- What needs to be stored?
- Cost of Operations
- Memory usage
- Ease of implementation (may not be the best strategy)

Non-linear DS:

Trees: use whenever we need to store hierarchical data





Descendant

1, 2 are are ancestors of 5;

9, 10 are descendants of 5

Tree → recursive data structure

↳ N nodes

↳ N-1 edges

Dept of node x = length of path from root to x

OR

No. of edges in path from root to x.

For above tree, depth of 2, 3 is 1

1 is 0

4, 5, 6, 7, 8 is 2

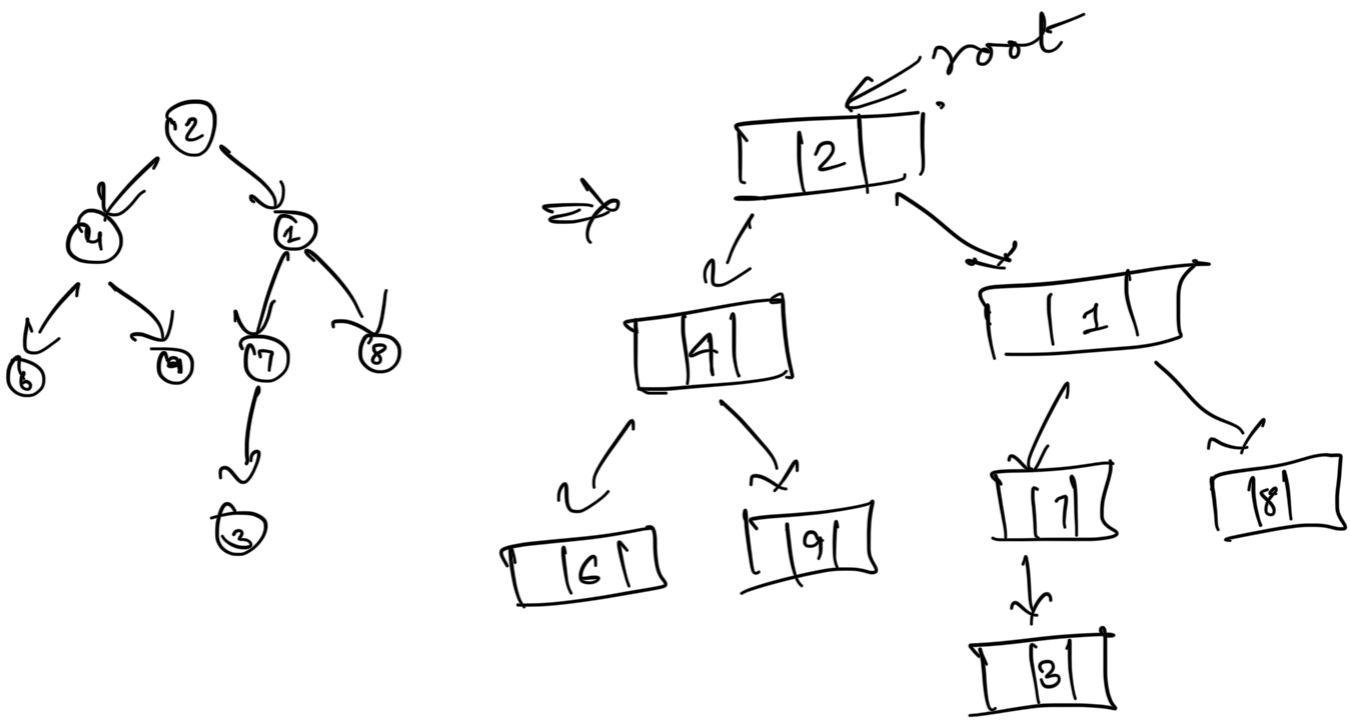
9, 10, 11 is 3

Height of node x : No. of edges in longest path from x to a leaf

for nodes 3, $h = 2$

1, $h = 3$

for leaf, $h = 0$



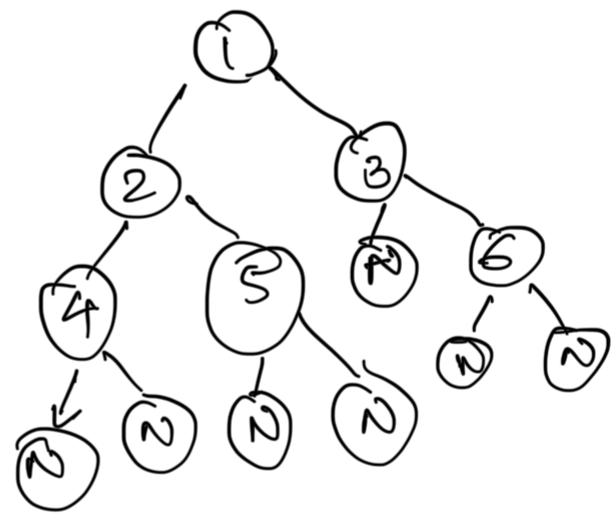
* Binary Search tree :-

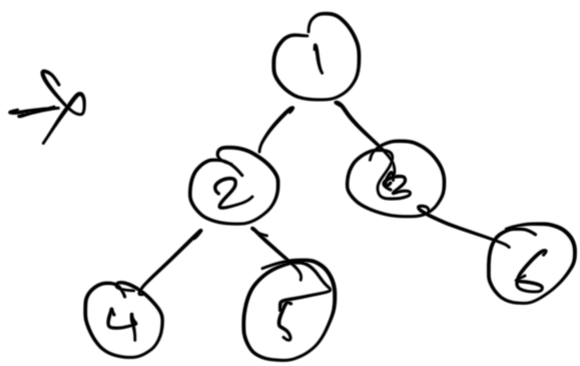
Ex:- Build Tree Preorder:-

Traversal order:-
- Root
- left subtree
- Right '''

1, 2, 4, -1, -1, 5, -1, -1, 3, -1, 6, -1, -1

-1 represents null node





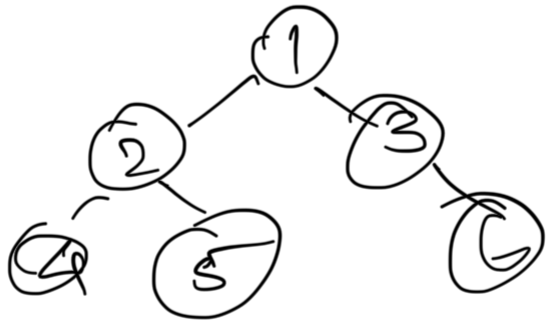
DFS

Inorder :- Traversal

- Left Sub

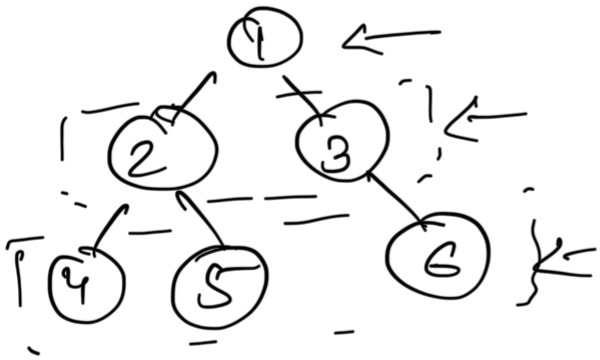
— Root
— Right "

Ex:— 4 2 5 1 3 6



Post Order Traversal: Left Sub
Right "
Root

Level Order Traversal:— FIFO
using Queue :—



~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~

1 2 3 4 5 6

to add next line ∵ its level order
we add null

1	null	2	3	null	4 5	6	null

→ BFS

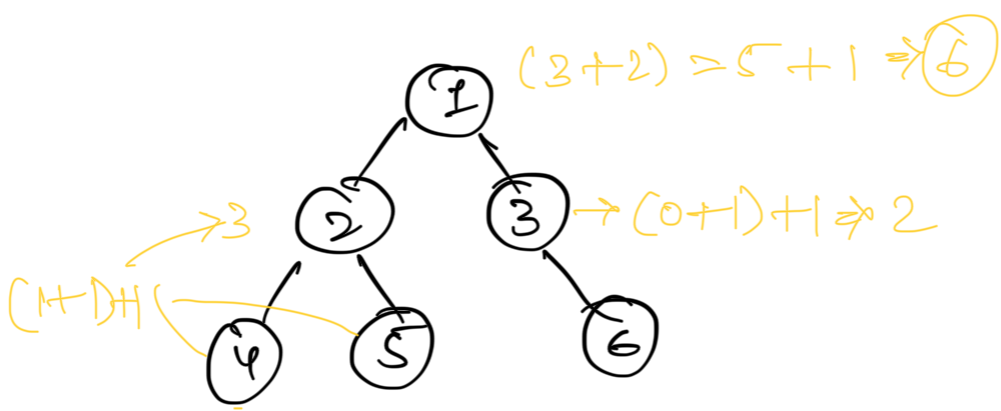
→ To count nodes :- (Code Concept)

use recursion

left node of (sub) Tree +

Right node " " " +

1

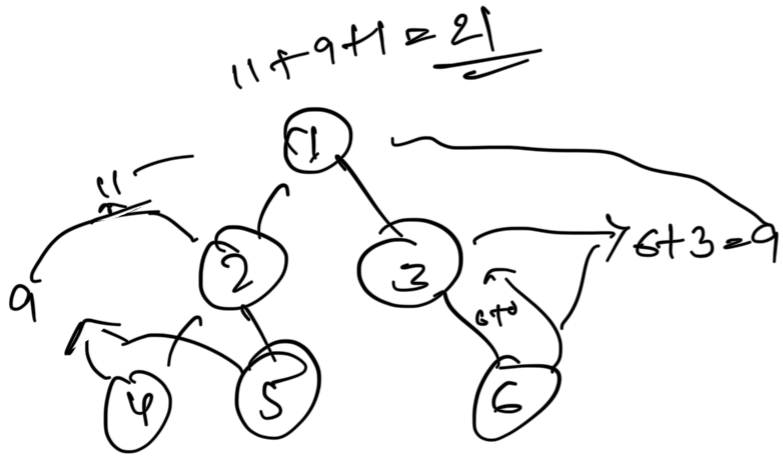


→ To get Sum of nodes :-

To get Sum subtrees wise :-

left subt + Right subtree.

+ root



→ Diameter of a tree :-

No. of nodes in the longest path between 2 nodes.

→ Subtree of another tree :-