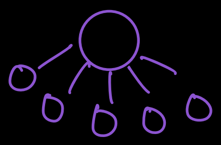
Trees :-

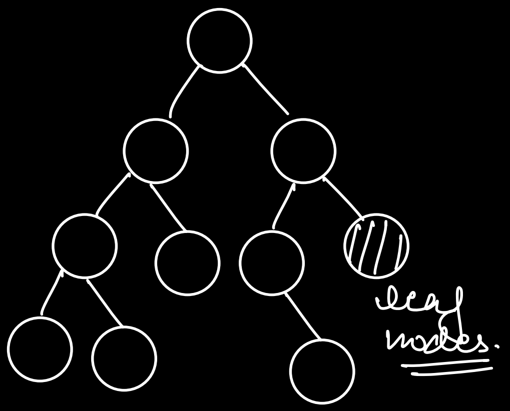
↳ hierarchical data.

- Directory structure
- XML / HTML
- Organisation structure
- Trie
- B / B+ Trees (DB indexes)
- Segment Trees
- RB / AVL Trees (Self Balancing BST)



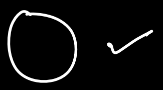
- FB | Google | LinkedIn . . .

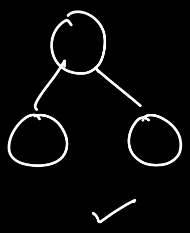
✱ Important for interviews.

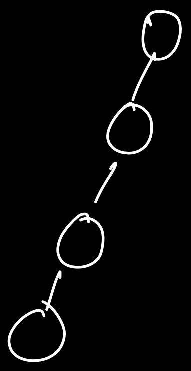


- Binary Tree

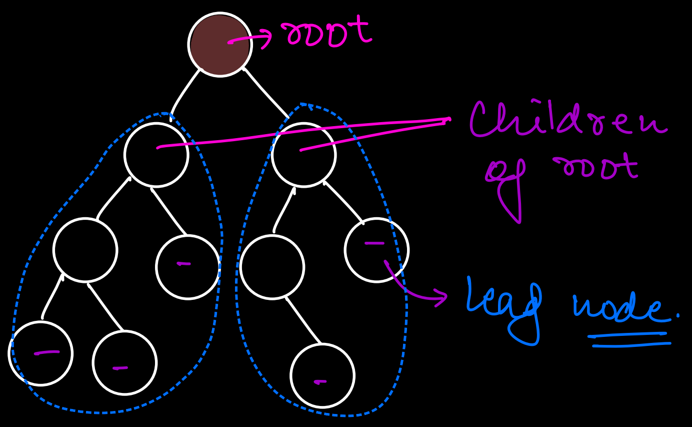
Tree in which every node can have at max 2 children

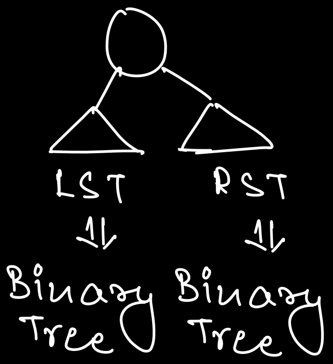






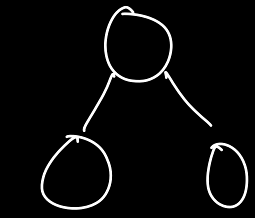
Skewed Tree.



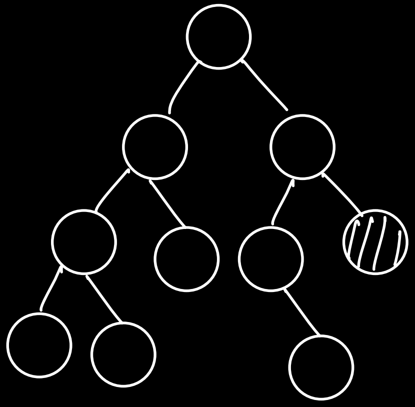


⇒ Recursive DS

Quiz # of edges in a Binary Tree with N nodes.



N=3
E=2



N=10
E=9

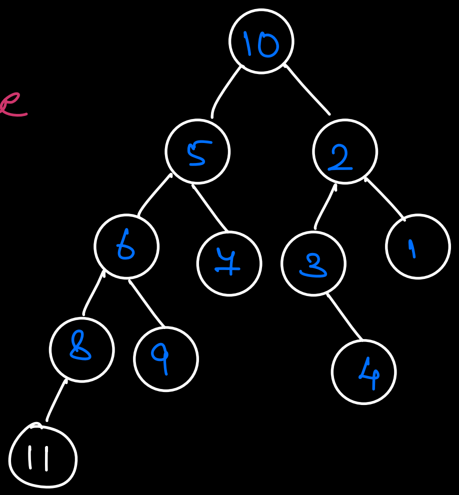
N nodes ⟶ $\boxed{(N-1)}$ Edges.

Height of a Node in Binary Tree

Ht of a node :-

↳ Distance of the node from the farthest reachable leaf node.

(# of Edges)



Ht(5) = 3

Ht(3) = 1

Ht(6) = 2

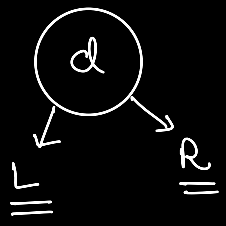
Ht(10) = 4

Ht(11) = Ht(9) = Ht(4) = 0

Ht(NULL) = −1

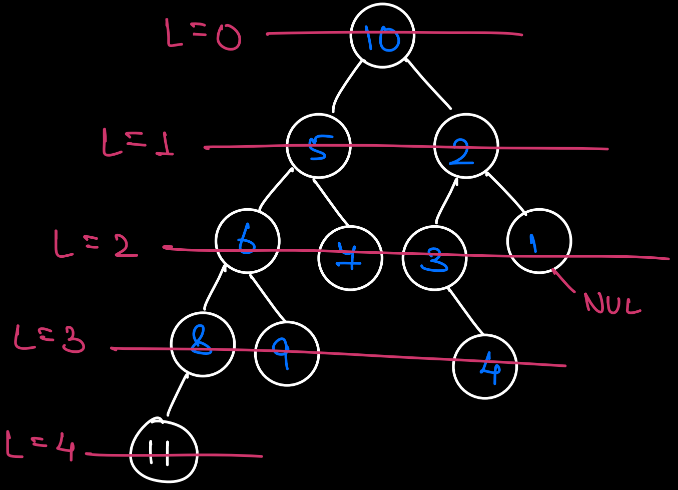
⇒ Height = 0.

```
Class Tree Node {
      int data;
      TreeNode left;
      Tree Node right;
      TreeNode(int x) {
            data = x;
            left = NULL;
            right = NULL;
      }
}
```



Depth of a node in a Binary Tree.

Distance of node from root node.



Dp(6) = 2

Dp(4) = 3

Dp(11) = 4

Dp(3) = 2

Dp(9) = 3

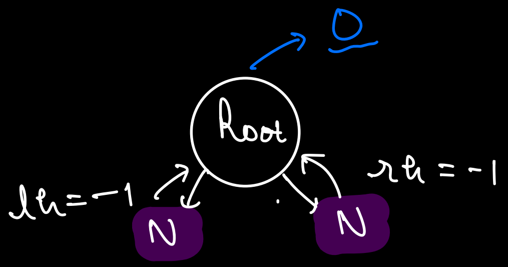
Dp(1) = 2

Dp(10) = 0

Q. Given a Binary Tree, find its height.

```
int height(root){
    if(root == NULL)
        return -1;

    int lh = height(root.left);   LST
    int rh = height(root.right);  RST
    return max(lh,rh) + 1;        Root
```

3



Post Order traversal

→ Traverse LST before RST

- * Root LST RST → PreOrder
- * LST Root RST → InOrder
- * LST RST Root → Post Order.

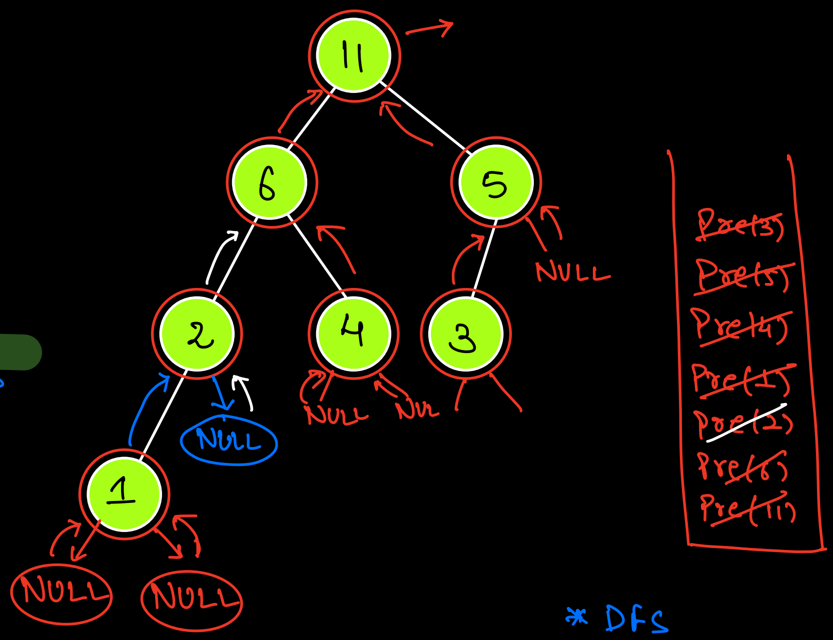
```
Void preOrder (root) {
    if (root == NULL) return;
    print (root.data);
    preOrder(root.left);
    preOrder(root.right);
}

Void inOrder (root) {
    if (root == NULL) return;
    inOrder(root.left);
    print (root.data);
    inOrder(root.right);
}

Void postOrder (root) {
    if (root == NULL) return;
    Post Order(root.left);
    postOrder(root.right);
    print (root.data);
}
```

* preOrder

root
LST
Root

11, 6, 2, 1, 4, 5, 3

$\underset{\text{Root}}{\uparrow}$ $\underbrace{\quad\quad\quad}_{\text{LST}}$ $\underbrace{\quad\quad}_{\text{RST}}$

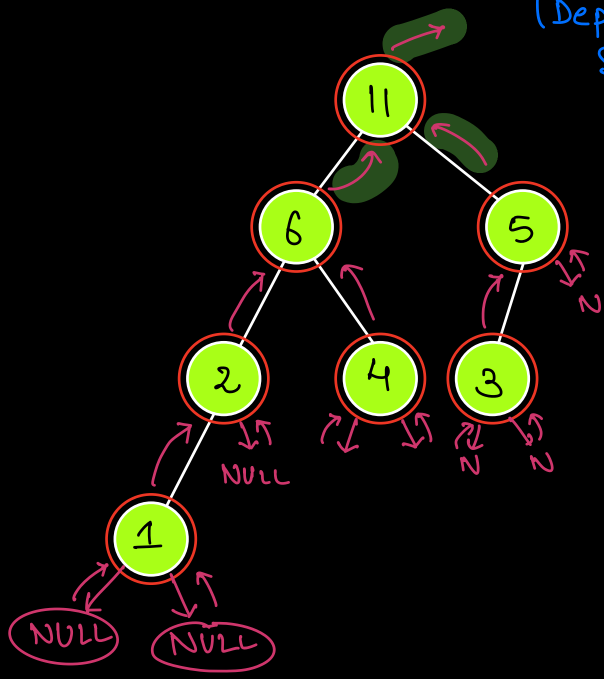


* DFS
(Depth First Search)

* Post Order

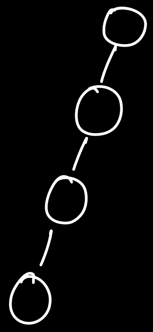
LST
RST
root

$\underbrace{1, 2, 4, 6, 3, 5, 11}$



TC : O(N)

SC : O(Height)
↳ O(N)
{WC}



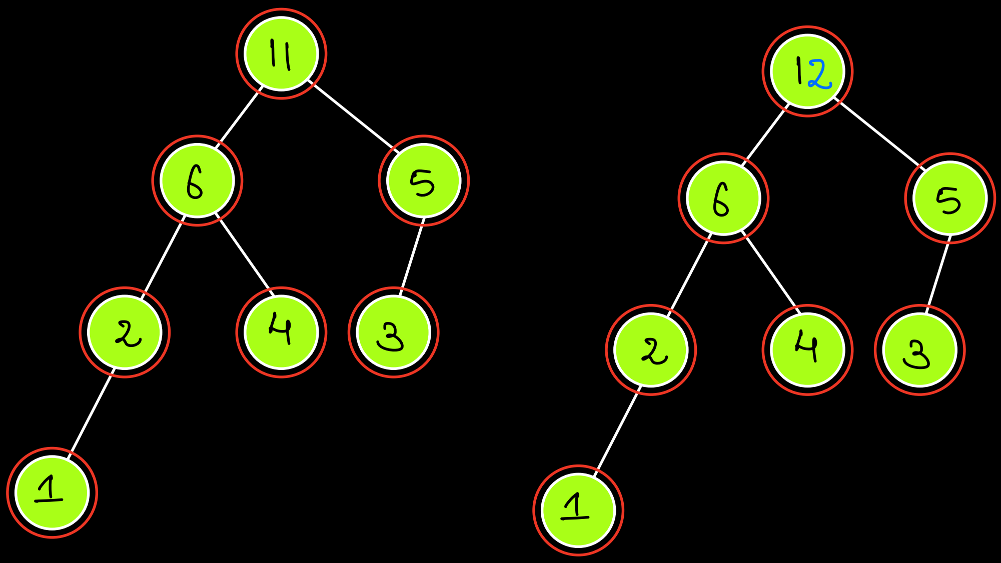
Skewed

H = N

→ PreOrder :

Root
LST
RST

* Search in a B.T
* Check if 2 trees are identical



→ Fail fast Approach

* Post Order
  - ↳ Height
  - → No. of nodes in B.T

Q. Given a B.T, search a value k in it

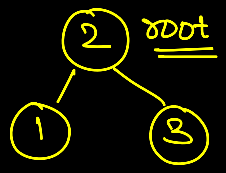
```
bool search(root, k){
        if(root==NULL) return false;
        if(root.data == k)
            return true;
        return search(root.left, k)
                              ||
                 search(root.right, k)
}
```

A or B
⇓ ⇓
T ×

Q. Given the preorder & inorder traversal of a Binary Tree in 2 Arrays. (No duplicates)
Construct the Tree

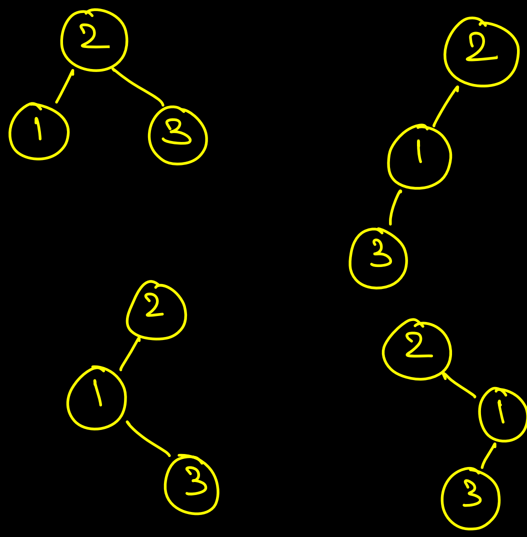
Amazon
MS

Pre-Order : 2 1 3
In-Order : 1 2 3

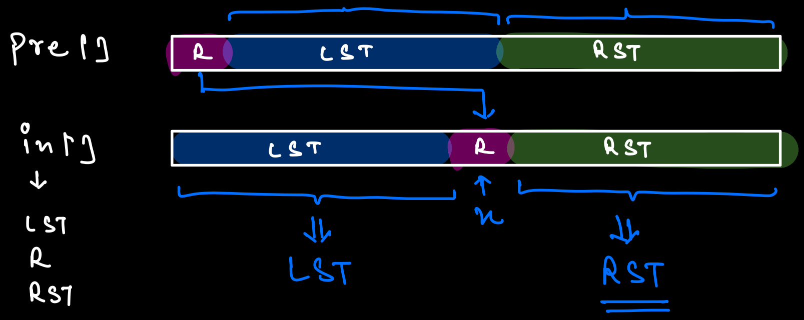


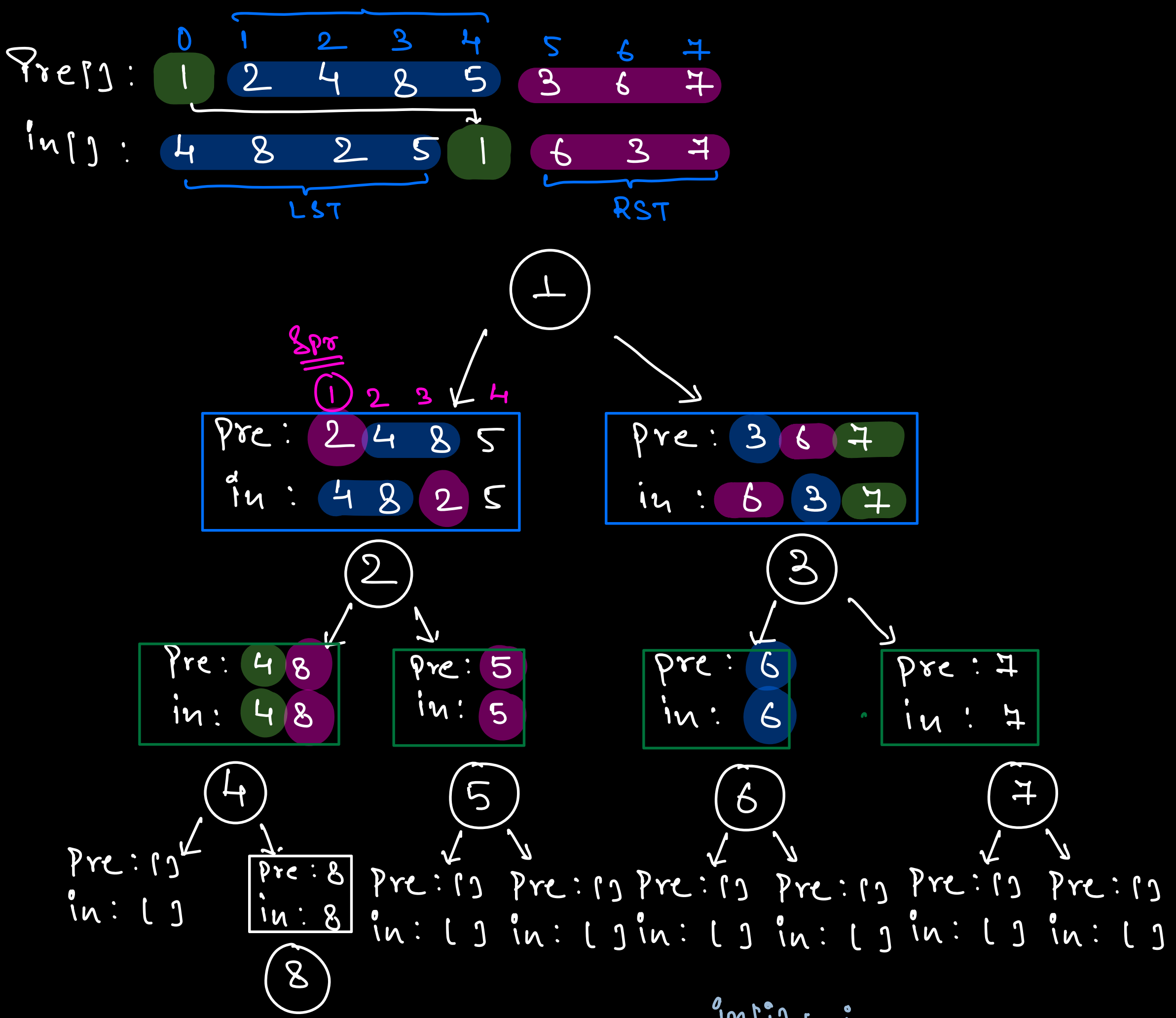
Pre-Order : 2 1 3

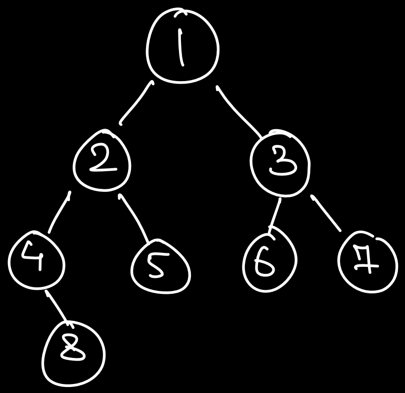
→ If only one tree traversal is given then we can't construct a unique tree.



→ first element of pre[] will be root
pre[0] → root.







* HashMap < int, int >

in[]	i
4	0
8	1
2	2
5	3
1	4
6	5
3	6
7	7

```
TreeNode construct( pre[], in[], Sin, ein, Spr, epr){
   // Assumption: Construct(pre, in, Sin, ein, Spr, epr)
   // returns the root of the tree with inorder
   // from Sin to ein in in[] & preorder from Spr, epr in pre[].
   if (Spr > epr) return NULL;

   TreeNode root = new TreeNode(pre[Spr]);
    // find the index of root in in[].
    idx = map.get(pre[Spr]);
    int a = idx - Sin; // # of elements in in[]/pre[]
                                of LST.

     root.left = Construct(pre, in, Sin,idx-1, Spr+1,Spr+a);
                                    in[]          pre[]

     root.right = Construct(pre, in, idx+1, ein, Spr+a+1, epr)'

      return root;
}
```

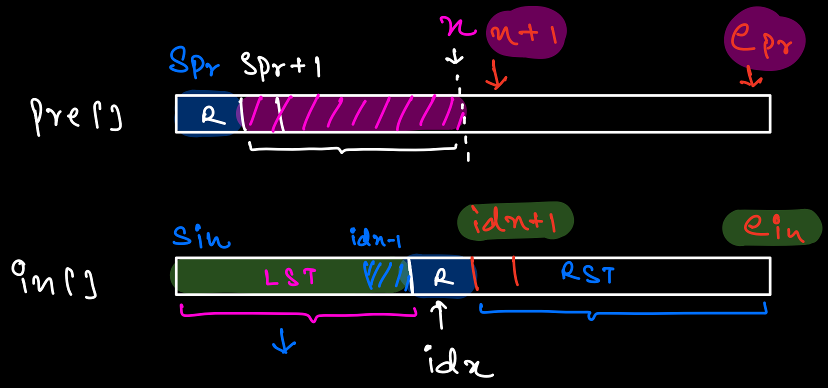
TC: O(N)

SC: O(N)

↳ Recursion
+
Hash Map.

Without HM

TC: O(N^2)



$[sin, idx-1] \rightarrow \boxed{idx - sin} = a$

$[spr+1, x] \rightarrow idx - sin$

$\Downarrow$

$x - (spr+1) + 1 = idx - sin$

$x - spr - \cancel{1} + \cancel{1} = idx - sin$

$x = spr + idx - \underline{\underline{sin}}.$

$= \underbrace{spr + a}$

* PreOrder[]
  PostOrder[] > Unique Tree ?

R LST RST

LST RST Root

——*——