Trees - Basics 1

DS learnt

Array

Linked List

Stack

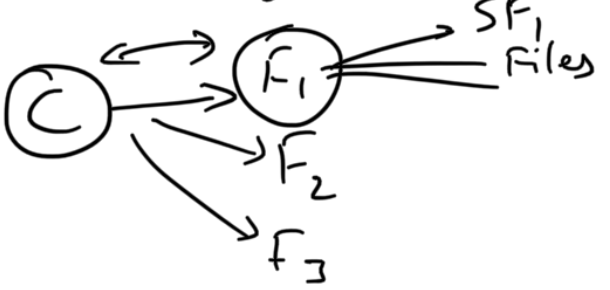
LIFO

Queue

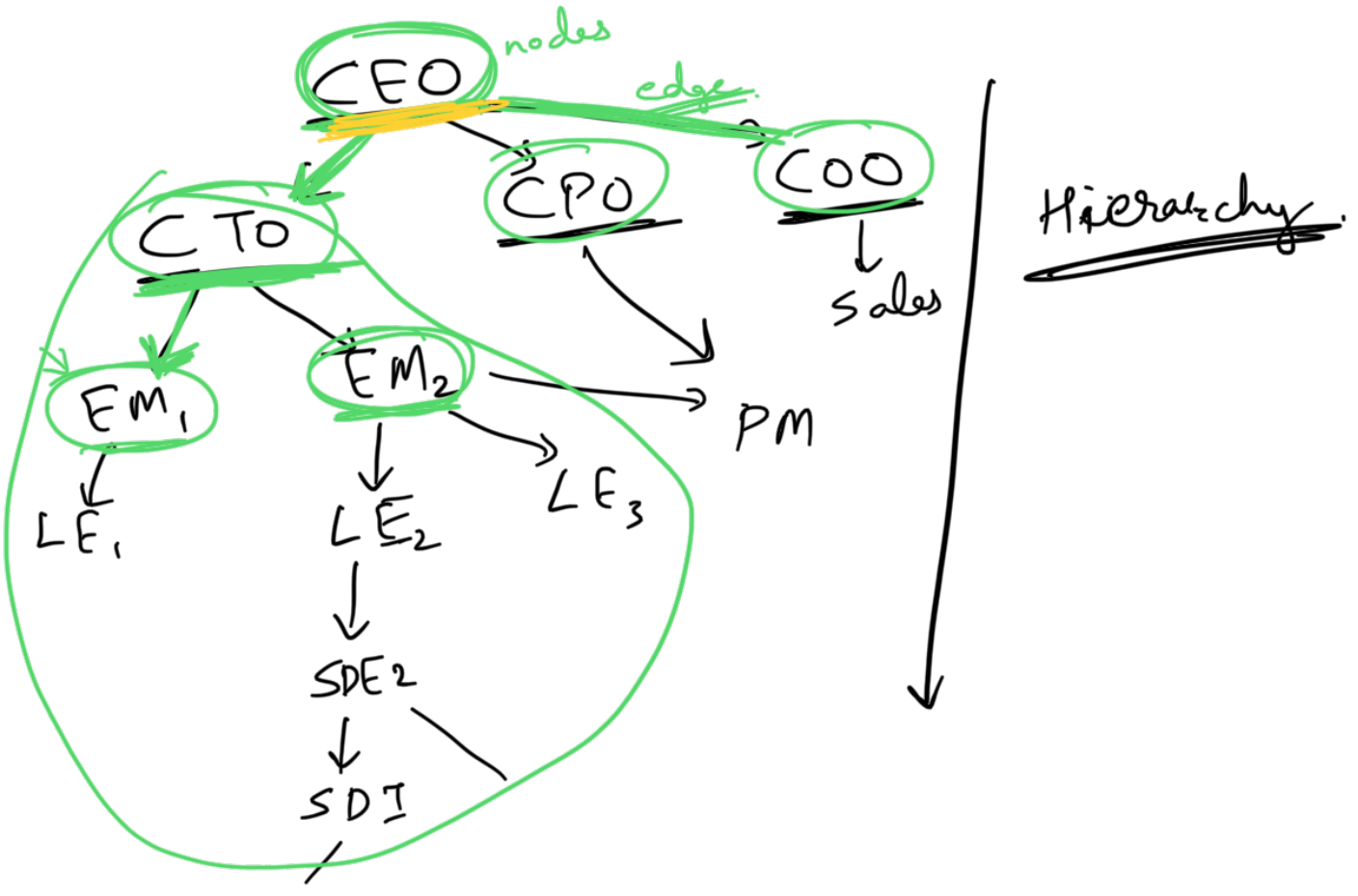
FIFO

Linear D.S

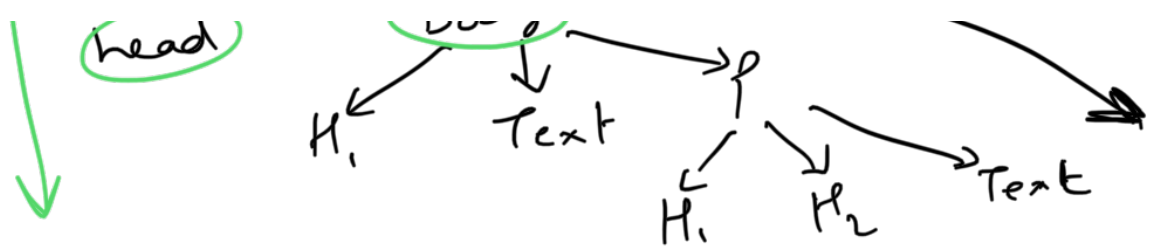
How to store hierarchy in the data → filesystem



There's a parent ↓ child

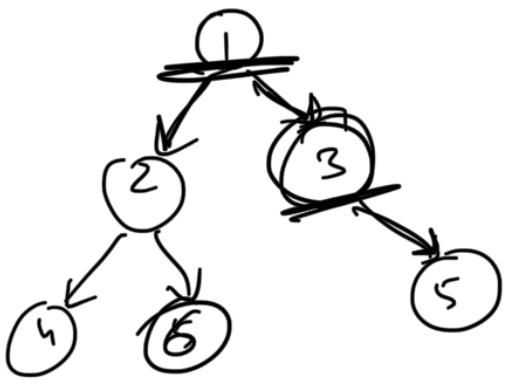


HTML → DOM

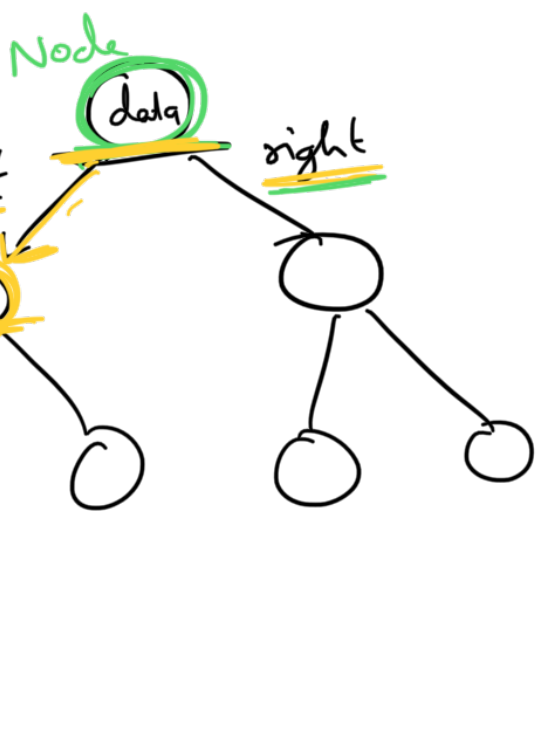


Tree (Graph → advanced)

Binary Tree → (0 / 1 / 2 children)



Node



B.T ?

L.L.

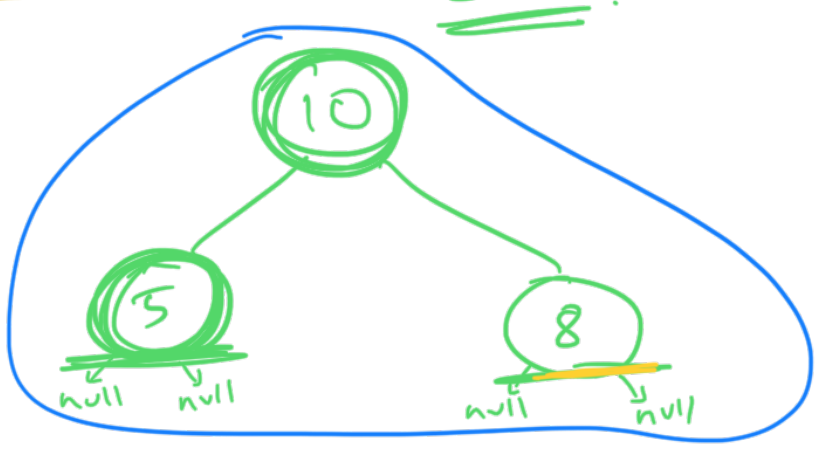
next.

B-T

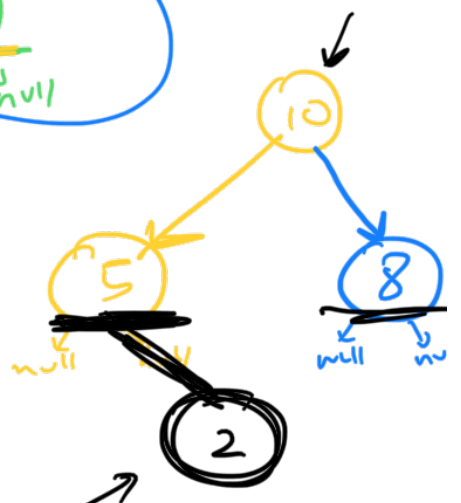

```
class TreeNode
{
    int data;
    TreeNode left;
    TreeNode right;
    public TreeNode(K)
    {
        this.data = K
        this.left = null
        this.right = null
    }
}
```

TreeNode t1 = new TreeNode(10);

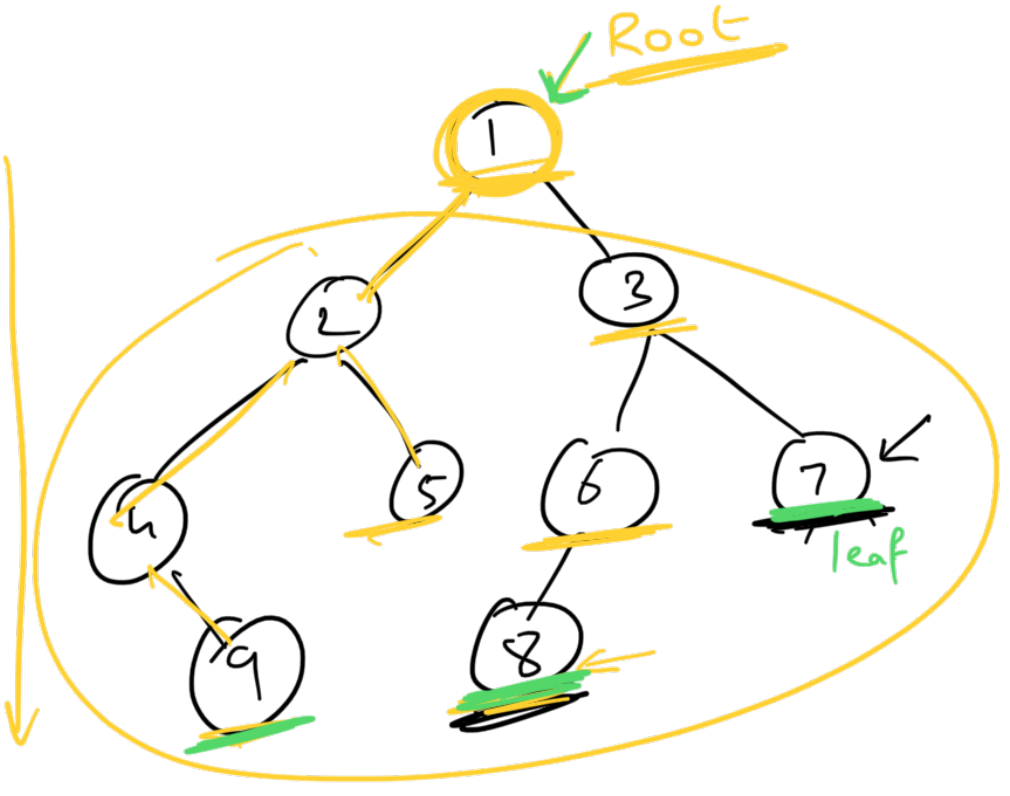
t1.data = 10

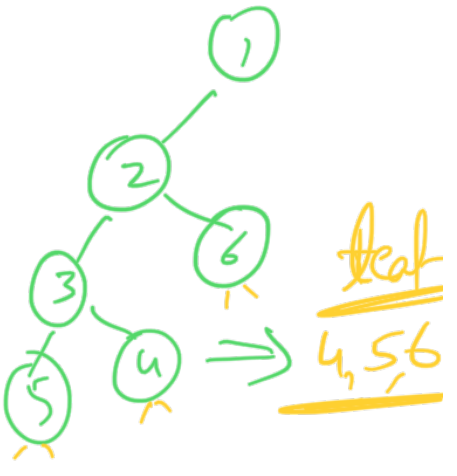



```
TreeNode t = new TreeNode(10)
t.left  = new TreeNode(5)
t.right = new TreeNode(8)
```




```
t.left.right = new TreeNode(2)
```





Binary Tree

root → topmost Node

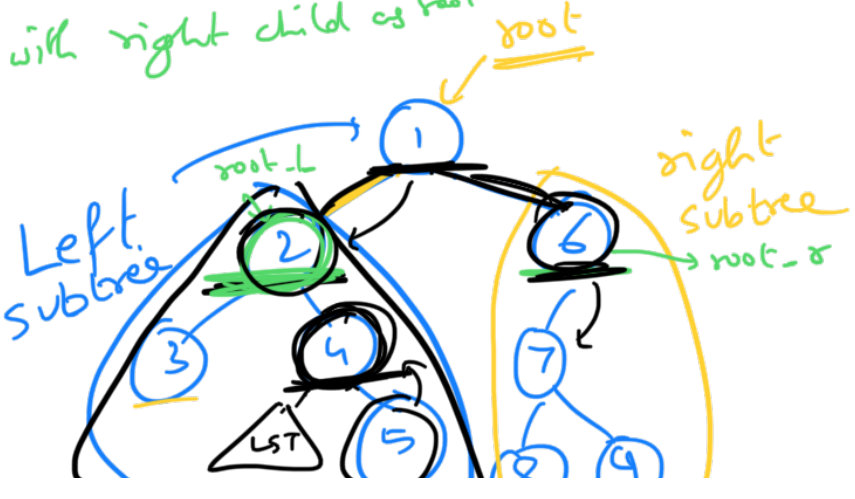
leaf → Nodes with 0 children

left subtree → subtree with left child as root.

right subtree → subtree with right child as root.

parent

child



root

1

2 3

4 5 6

7 8 9 10

11

LST

Subtree

Subtree

Recursion

to traverse the B.T.

Search (1)

Search 2

Search 3

Left

Fla

Root

left

LST

RST

Standard Rule

First LST & then RST visit

RST Root LST

LST comes before RST

root

RST

LST

Types of traverse

Root	LST	LST
LST	Root	RST
RST	RST	Root
preorder	Inorder	Post order

root

RST

root

LST

root

RST

LST

root.

1

2 3

1 2 3

Preorder: 1 2 3 4 5

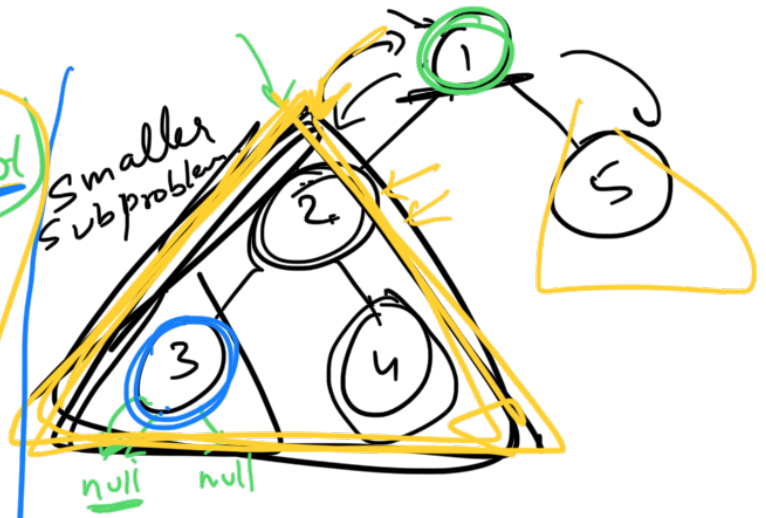
Print

Use recursion

2 1 3

2 3 1

```
Void preOrder(TreeNode root)
{ //Base if (root==null) return
  print(root)
  preOrder(root.left)
  preOrder(root.right)
}
```




```
Void InOrder(TreeNode root)
{ // if (root==null) return
  Inorder(root.left)   <= LST
  print(root)
  Inorder(root.right)  <= RST
}
```

Q. How to print the preorder if OS doesn't support recursion. => (use stack!)

↳ advanced class.

Preorder

root

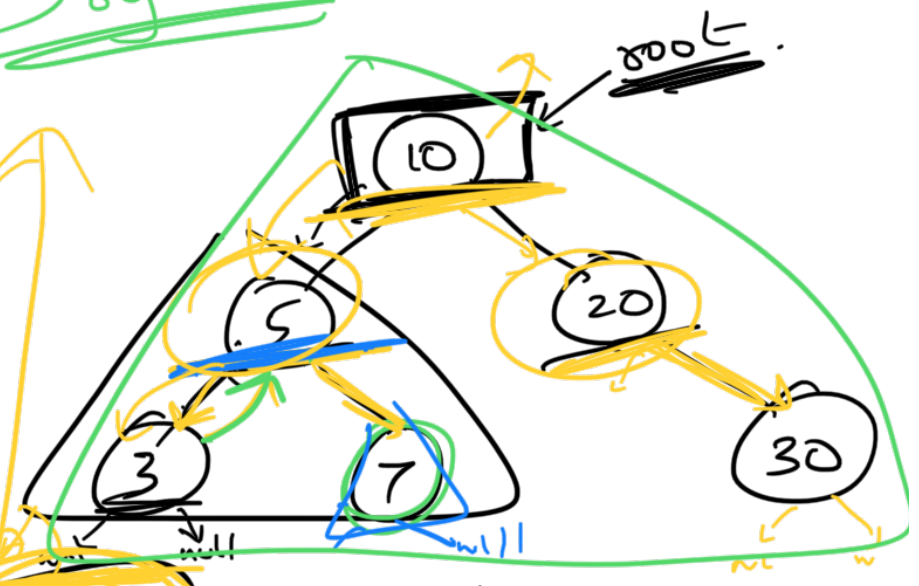
LST

RST

```
Preorder(Root)
{ if (root==null) return
  print(root)
  Preorder(root.left)
  preorder(root.right)
}
```

return;

Dry Run



pre(10)

10

print(10)

pre(5)

pre(20)

10, 5, 3, 7, 20, 30

pre(r)
5
pre(3)
pre(7)
pre(3)
3
pre(NULL)
pre(NULL)
pre(NULL)
pre(7)
7
pre(NULL)
pre(NULL)
pre(NULL)
pre(NULL)
pre(20)
20
pre(NULL)
pre(30)
pre(NULL)
pre(30)
20
pre(NULL)

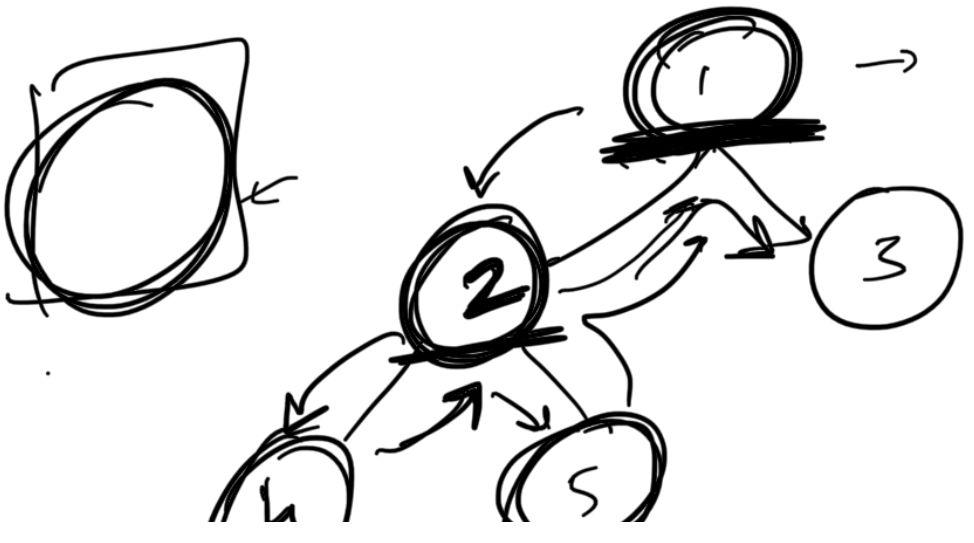
no. of operations ⇒ O(N).

S.C ⇒ O(Height / depth)

⇒ O(N) in worst case

depth

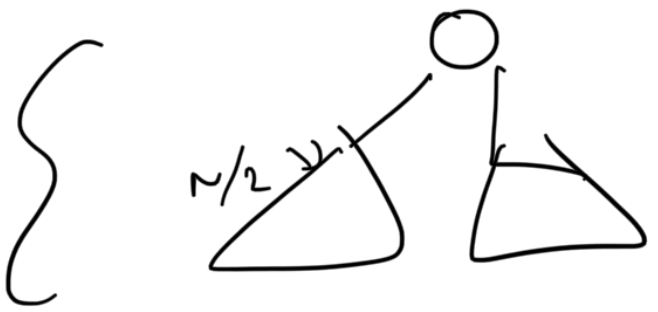
B-T



preorder

1 2 4 5 3

T.C ⇒ O(N)



$2\,T(N/2)$

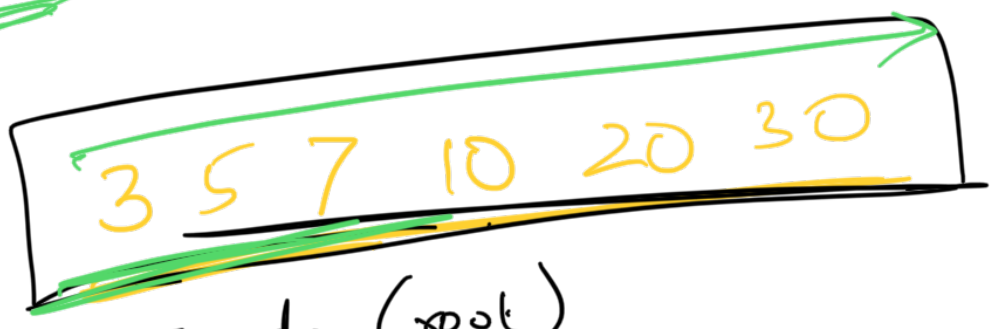
$$T(N) = 1 + T(K) + T(N-K)$$

Inorder

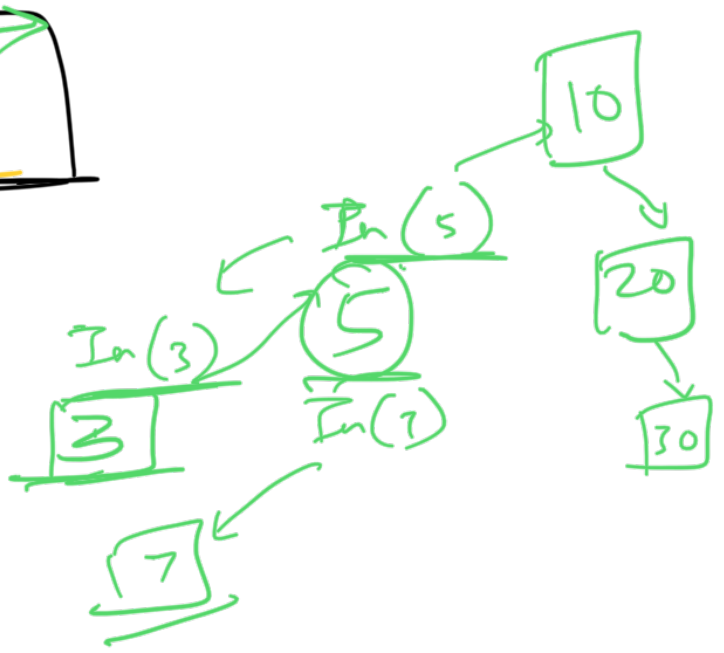
LST

root

RST




```
void Inorder (root)
{  // Base
   if (root == null) return;
   Inorder (root.left)
   Print (root.data)
   Inorder (root.right)
}
```

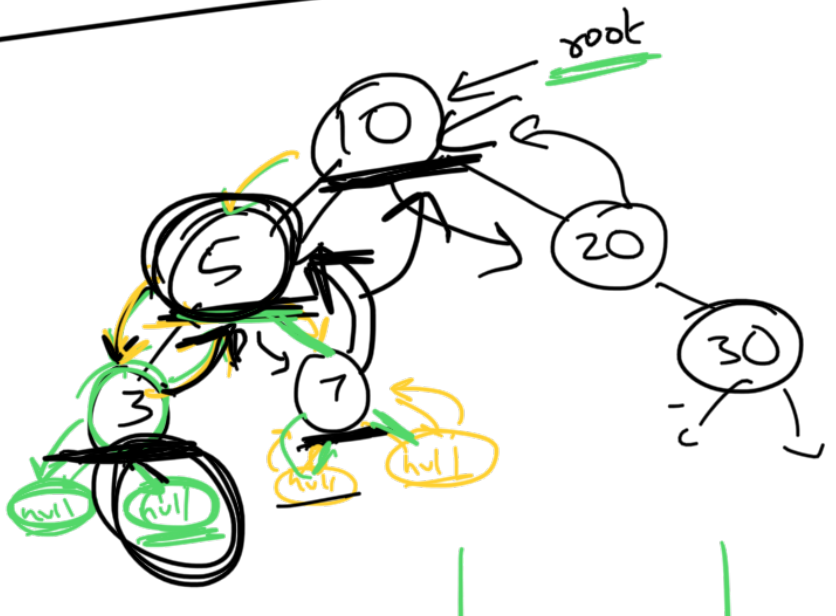


Postorder

LST

RST

root.val()




```
void postorder (root)
{  if (root == null) return
   postorder (root.left)
   postorder (root.right)
   print (root.data)
}
```

print (root.data)

return

37 5 30 20 10

P(3)
P(5)
P(10)

P(10)
P(5)
P(20)

P(5)
P(3)
P(7)
Pre(5)

P(7)
P(null)
P(null)
Pre(3)

P(7)
P(null)
P(null)
Pre(7)

P(20)
P(null)
P(30)
20

P(30)

30

Break

10:50

Postorder

void postorder(root)

if (root == null)
return

postorder (root.left)
postorder (root.right)
print (root.data)
return

1
2
3
null
null

root

LST
RST
root

postOrder(1)

root != 1

P(3)

P(1)

P(1)

P(2)

P(3)

P(2)

P(NULL)

P(NULL)

Print(2)

P(NULL)

return

P(NULL)

return

P(3)

int main()

int i = 5

int j = 6

int K = 9

int sum = i + j + K

i = 5

5

4 byte

1010

Preorder

root

Inorder

LST

root

Postorder

LST

RST

LST→
RST

RST

CFO
100+1
data
145
101
SS
data
CTO
LST
LE
RST
CPO
10
CFO
5
5K
CTO
COO
2-5
2-5
2-5
2-5
Ready

Q. Find height of a B.T.

↳ length of the largest path from root to the leaf.

root

3

4

3

⇒ 4

No. of nodes

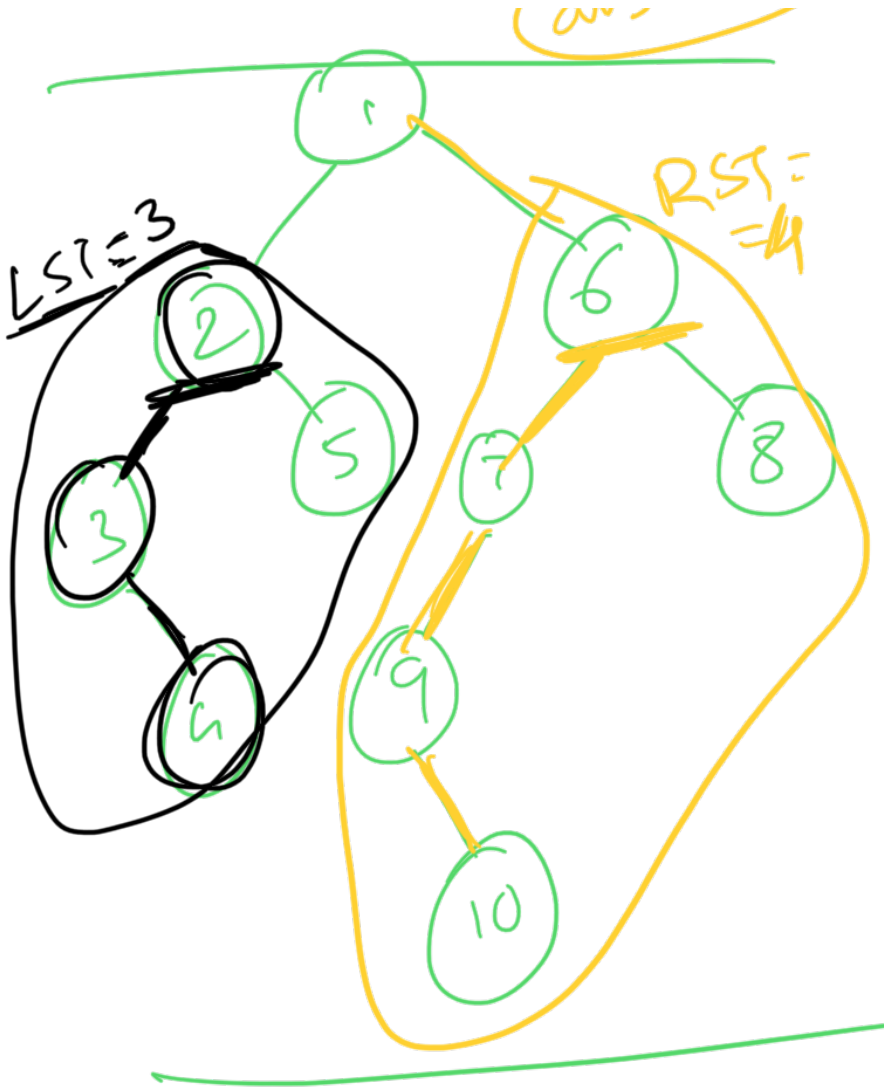
No of edges

3

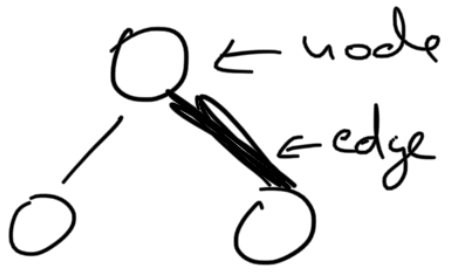
→ no. of nodes

root

ans = 5



max(3,4) + 1

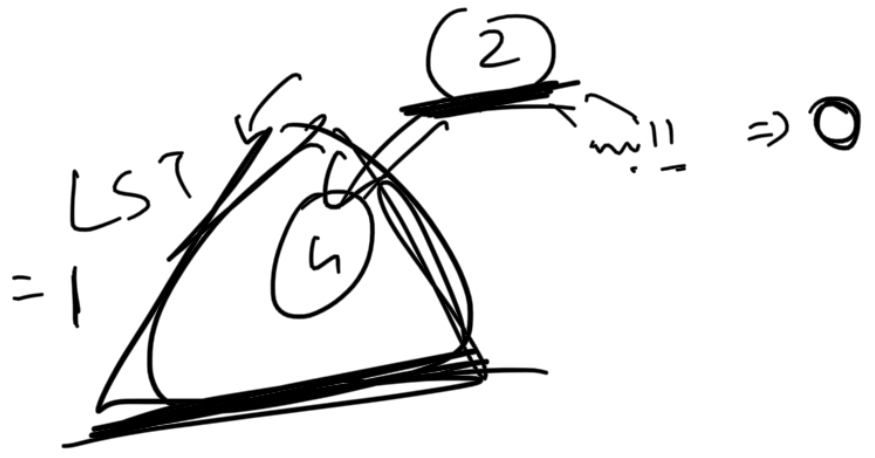


Height BT = max(Height LST, Height RST) + 1

assumption

```
int height (root)
{
//Base if(root==null) return 0;
leftHt = height(root.left)
rightHt = height(root.right)
return max(leftHt, rightHt)+1
}
```

max(2,3) + 1



$\Rightarrow 0$

$\max(1, 0) + 1$

$= 1 + 1 = 2$

$\max(0,0) + 1 = 1$

H.W

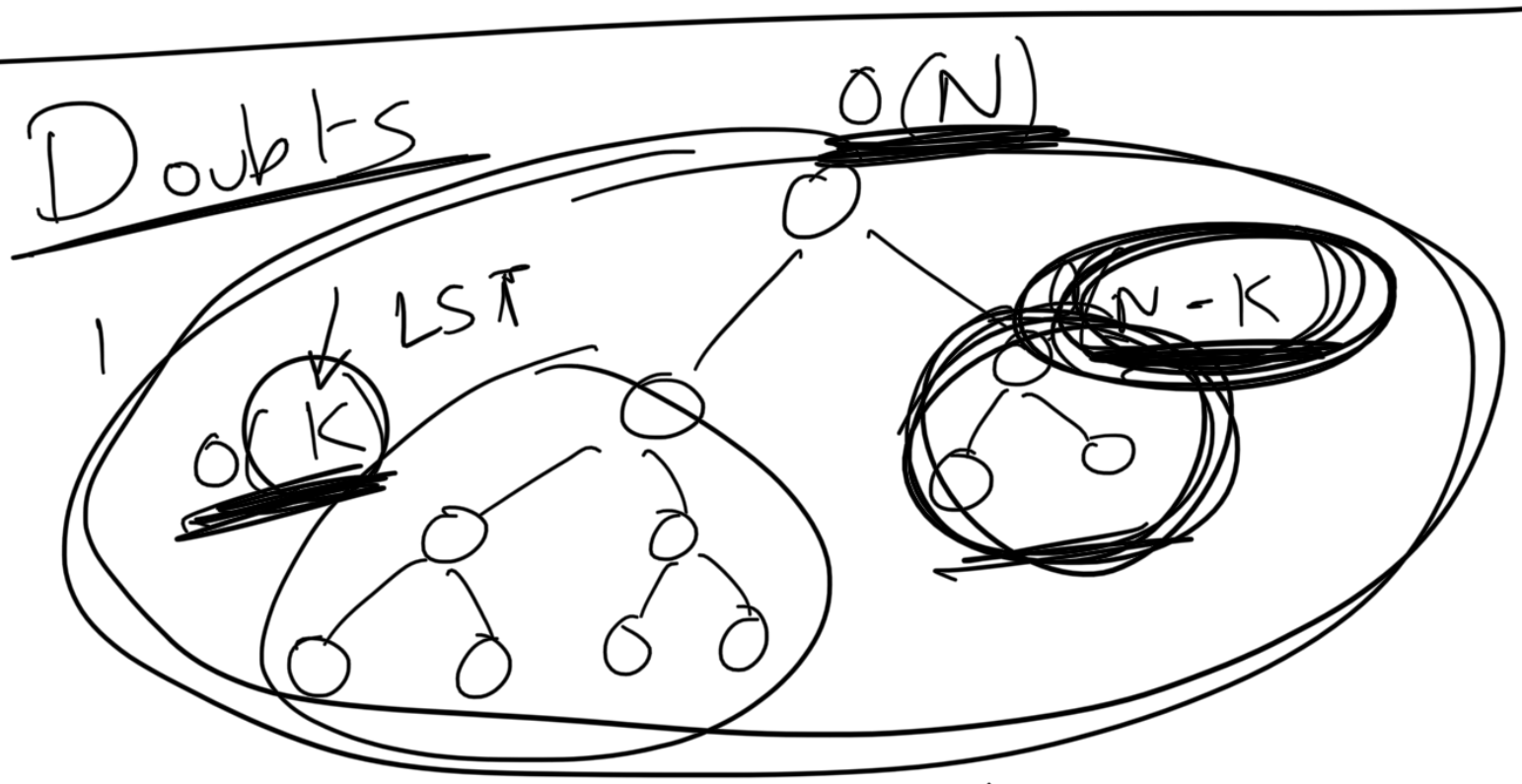
height → no. of edges

T.C $\Rightarrow O(N)$

S.C $\Rightarrow O(\text{Height})$

$\rightarrow O(N)$ skewed B.T.

Doubts



$K \leq N$

Java { TreeMap, Set, Map } C++ => BBST.

$\log(N)$

HashMap → O(1)

value → BBST

Node

edge

Coul.

Coul

max(left, right

left + right + 1

1

2