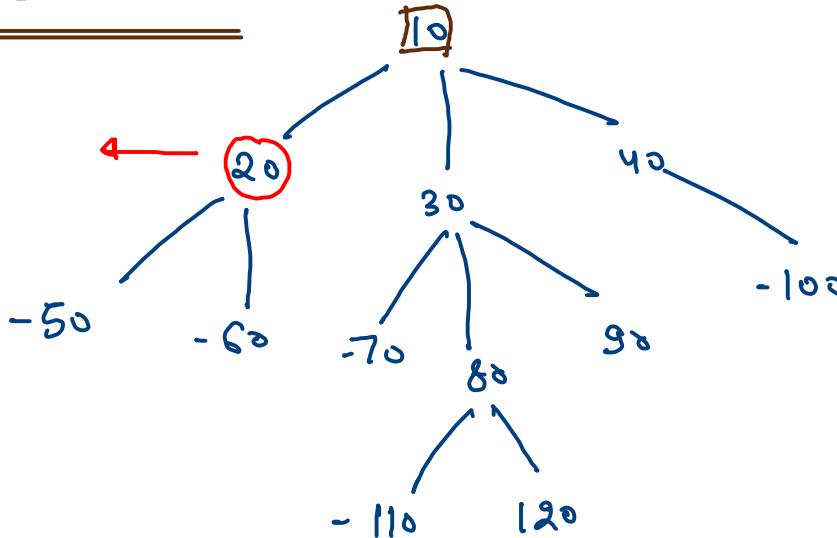


Node with Maximum Sum Subtree

Maximum sum subsequence



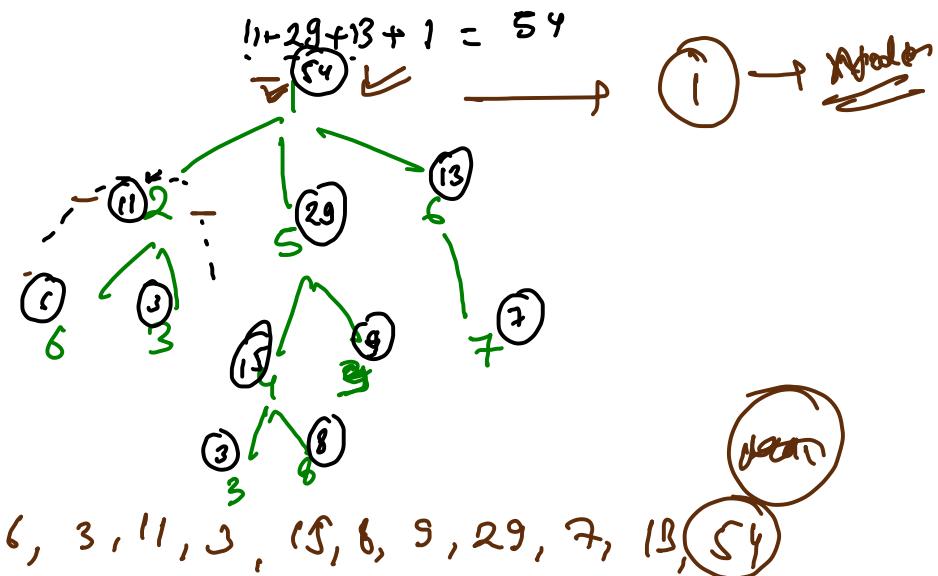
Maximus Subm Subr

condition →

if (node.data >= 0)

Node = Root

1



Sum of Generic Tree -

Recursion -

Expectation →

$\text{sum}(10) \rightarrow$ overall sum of all subtree having root is 10.

faith →

$\text{sum}(\text{node.children.get}(i)) \rightarrow$

E.g. 10 as root.

$\text{sum}(20) \rightarrow 20$

$\text{sum}(30) \rightarrow 140$

$\text{sum}(40) \rightarrow -60$

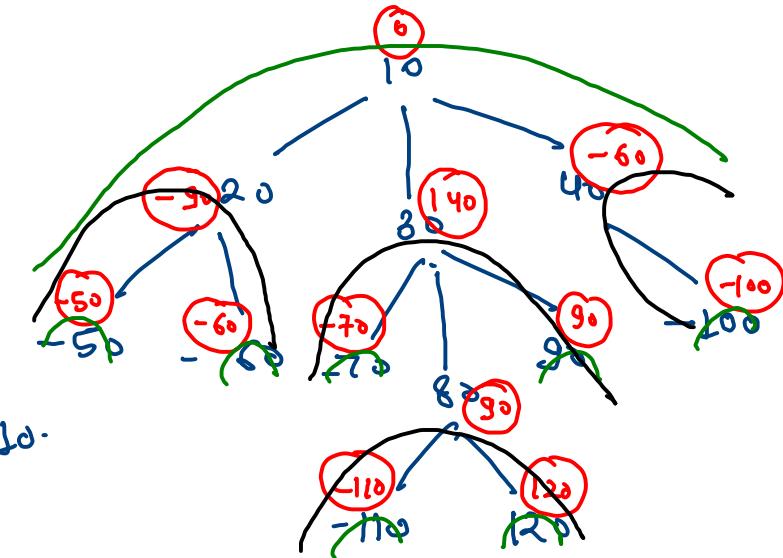
sum ↗

[+ children, → find sum till you

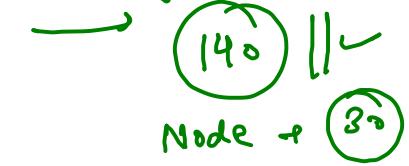
sum variable -

Merging → faith & Expectation

↳ return sum variable + node.data,



Max. sum of subtree.



30 @ 140

```

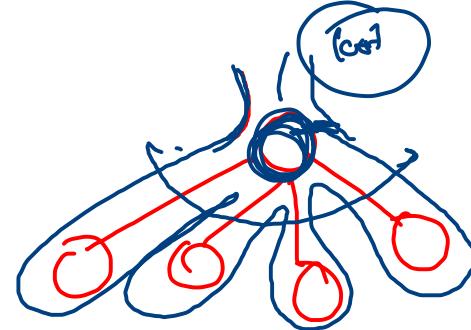
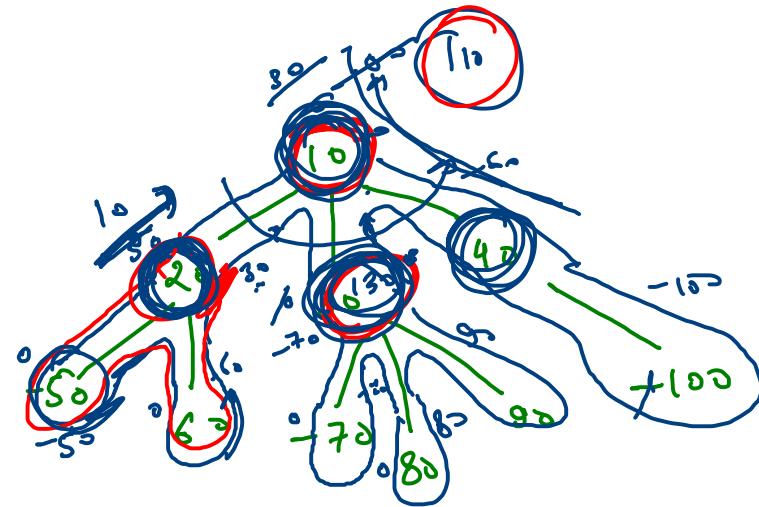
public static int sumOfTree(Node root) {
    int sum = 0; // sum from child
    for(Node child : root.children) {
        sum += sumOfTree(child);
    }
    Root Area → lgt
    {  

        int osum = sum + root.data; // overall sum
        return osum;
    }
}

```

~~MaxSum = 0~~

$\Rightarrow \text{Max sum} = \text{Math.max}(\text{maxSum vs. osum})$.



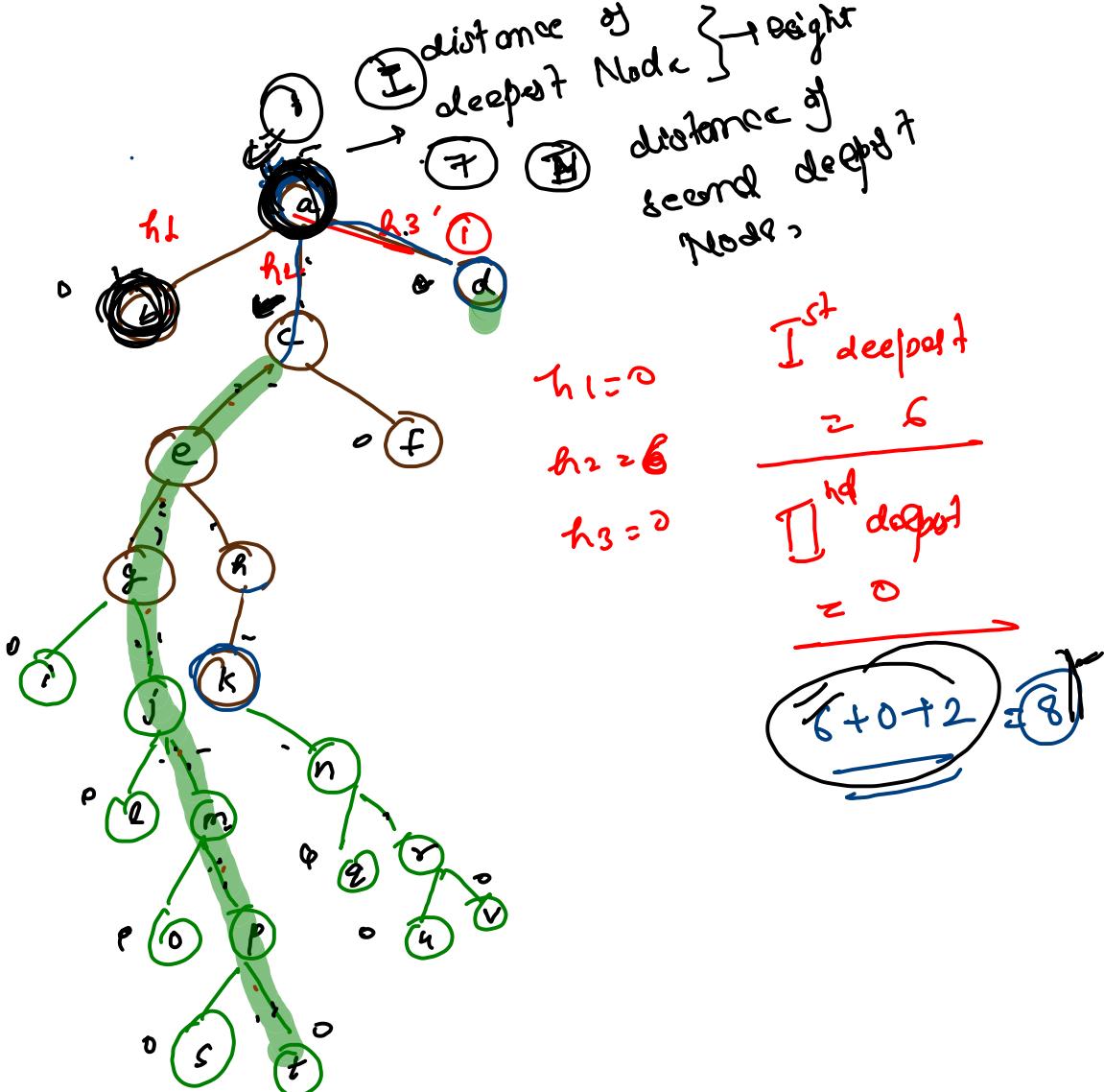
Diameter of Generic Tree

Diameter is Max. distance b/w any two nodes.

on the basis of →
 ↑ Edge
 ↑ Node

diameter = 5

$t \rightarrow u$, } max dist
 $s \rightarrow v$



- ✓ First deepest] - \max height || -
 - ✓ Second deepest] - $2\max$ height

firstmax = -1

Second max = 1

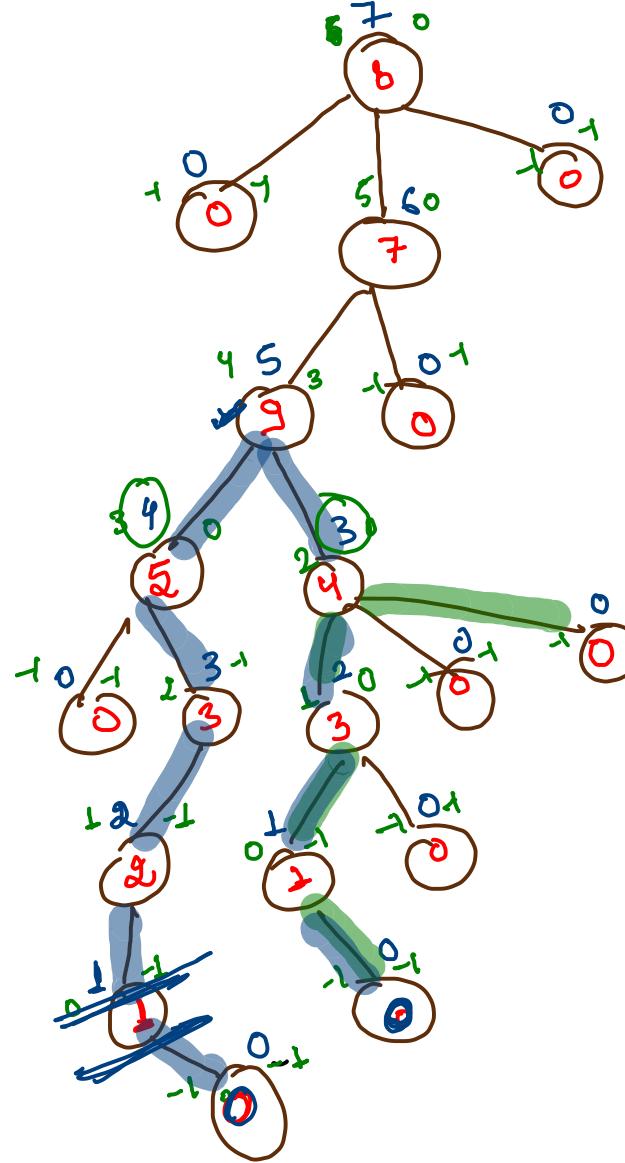
$$\text{return} \rightarrow \underline{\underline{FM + SM + L}}$$

A hand-drawn diagram of a circle. A horizontal red line segment passes through the center of the circle, labeled "diameter" in red. A vertical red line segment extends from the center to the top of the circle, labeled "radius" in red. The text "fm hl sm" is written above the circle in green.

$h \rightarrow$ freight

EM → first Max

gpa → Second Mar



A hand-drawn diagram of a circle. Inside the circle, the letter 'g' is written. A horizontal line segment passing through the center of the circle is labeled 'Max' above it and has two parallel lines below it, indicating it is the maximum diameter. A radius extending from the center to the right is labeled 'First f' above it and 'Second f' below it, with the number '2' written under 'Second f'. An arrow points upwards from the end of the radius.

```

public static int diameter = 0;

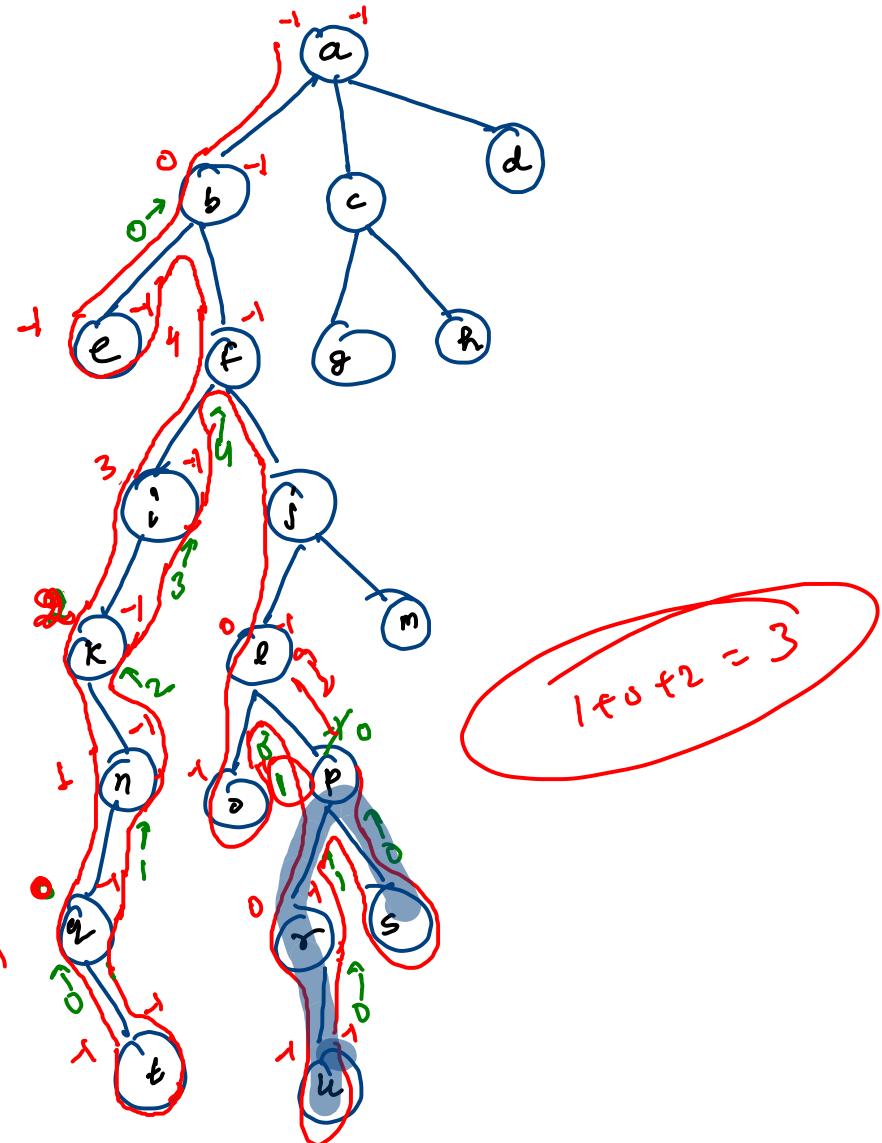
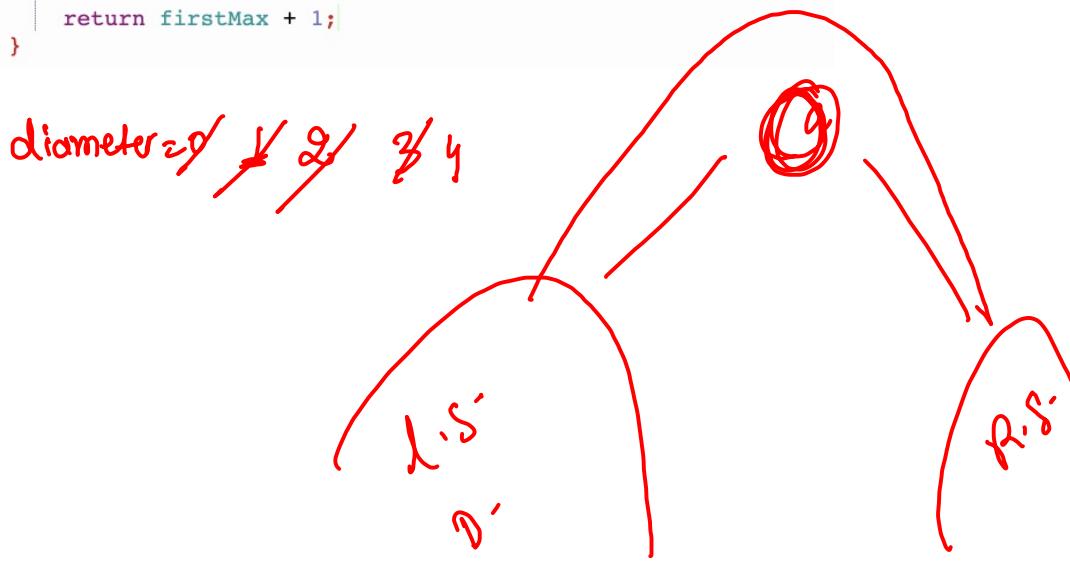
public static int heightForDia(Node root) {
    int firstMax = -1; // actual height
    int secondMax = -1;

    for(Node child : root.children) {
        int rht = heightForDia(child); // rec height
        if(rht >= firstMax) {
            secondMax = firstMax;
            firstMax = rht;
        } else if(rht > secondMax) {
            secondMax = rht;
        }
    }

    diameter = Math.max(diameter, firstMax + secondMax + 2);

    return firstMax + 1;
}

```



linearize → expectation → linearize(λ)

preorder

Recursive
space → Allowed.

time complexity.
 $O(n^2)$

faith.

linearize(20)

linearize(80)

linearize(40)

$110 \rightarrow 120 \rightarrow 90 \rightarrow 40 \rightarrow 100$

↓
individually linearize

10

↓

20

↓

50

↓

60

↓

80

↓

70

↓

80

↓

70

↓

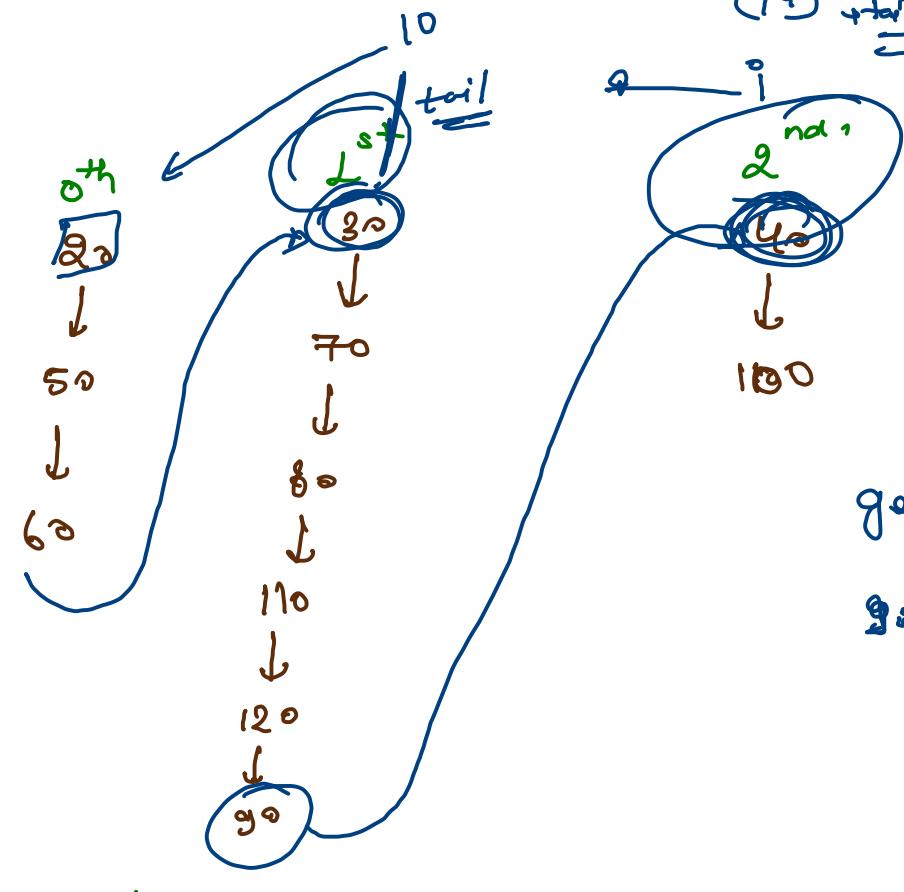
80

↓

40

↓

100

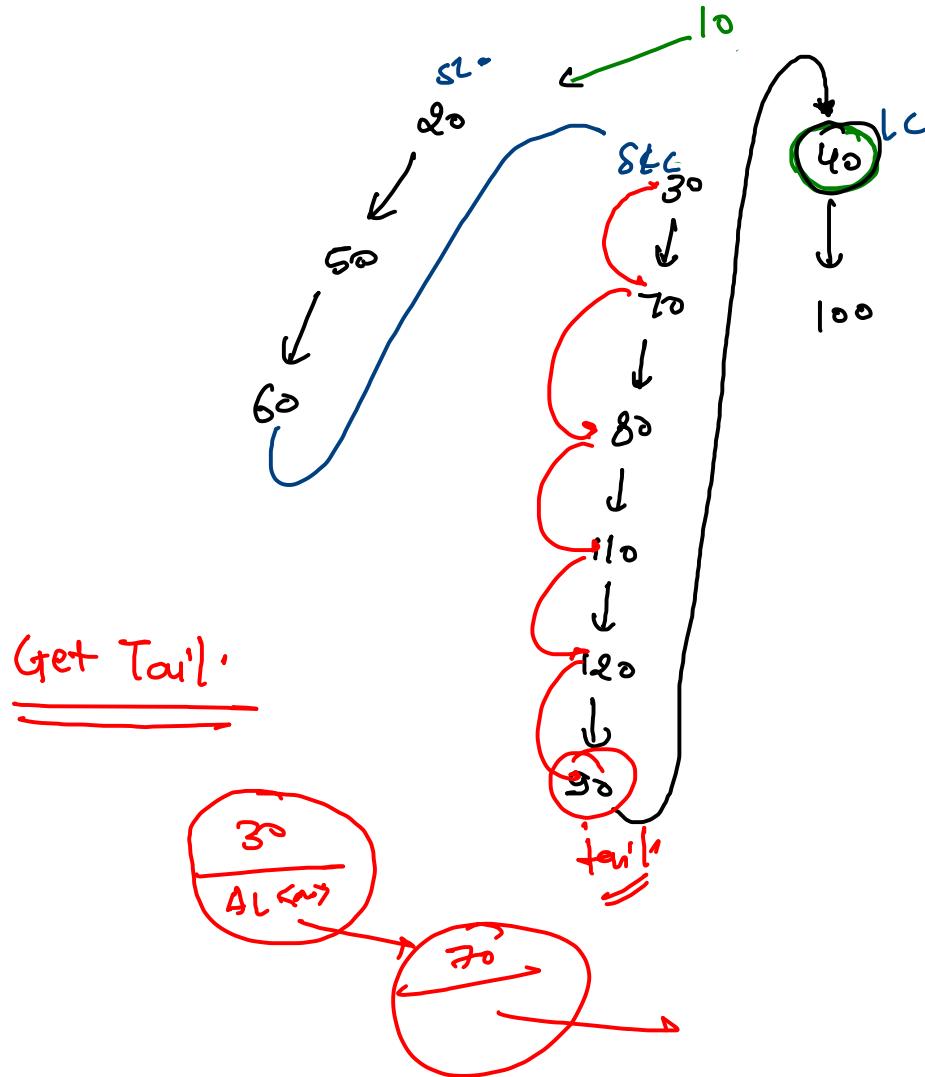


get i → Remove
from Root

for(i-1)

get tail → 90

90.children.add(40)



Steps -

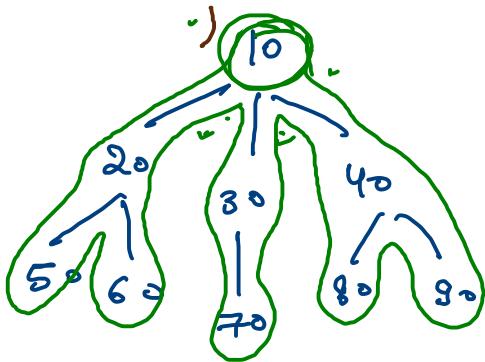
- ① linearize child from Recursion.

faith,

- ② Get last child.
 - ③ Get second last child.
 - ④ Remove last child from root·children.
 - ⑤ Get tail of SLC & add LC as children in SLC
- Perform these steps until size of root·children > 1-

PreOrder and PostOrder

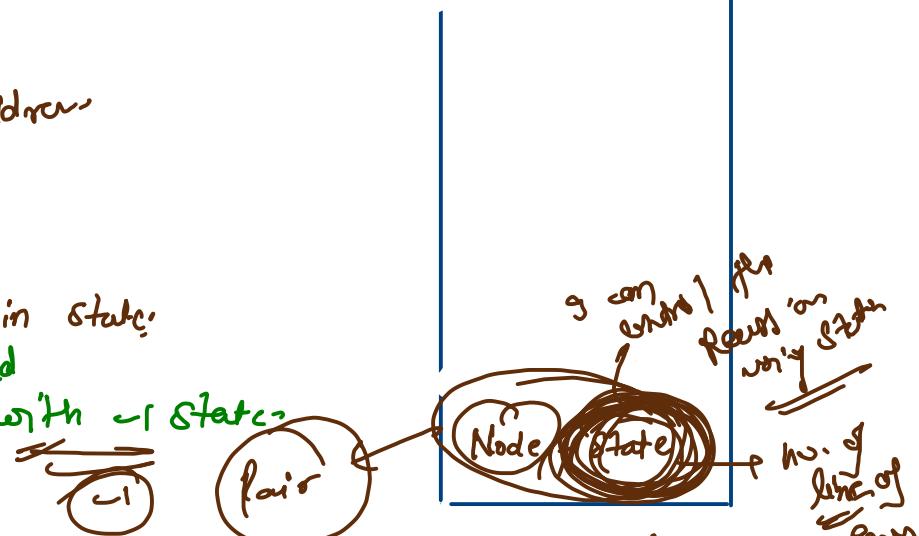
from Iterative



while ($stack.size() > 0$) {

- = State = -1
↳ preOrderOrder
- = Stateff;
- State = no. of children
↳ PostOrder
- . pop.

otherwise, increase in state:
 { state under child
 add in stack with -1 state
 ↳ c1



'Node → State'

State → -1 → Node Pre.

0 → edge pre of 0th child

1 → Edges of 1st child loop

2 → Edges pre of 2nd child loop

(no. of children) 3 → Node Post

↳ preOrd

10
20
50
60

80
30
70
40

60
30
70
40

90

↳ PostOrder

50 40

60 10

20

70

30

80

90