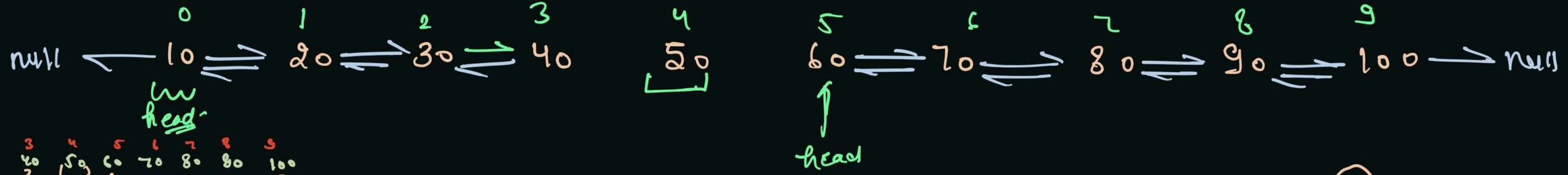
Convert Sorted DLL to BST

Sunday, 29 August 2021 10:17 AM



lo = 0
hi = 9

mid = $\frac{lo+hi}{2} = \frac{9}{2} = 4$

root = 50

left → lo to mid-1
right → mid+1 to hi
Even length → consider mid-1

mid = 50

left part break

```
mid.left.right = null
mid.left = null
```

TreeNode head2 = mid.right;

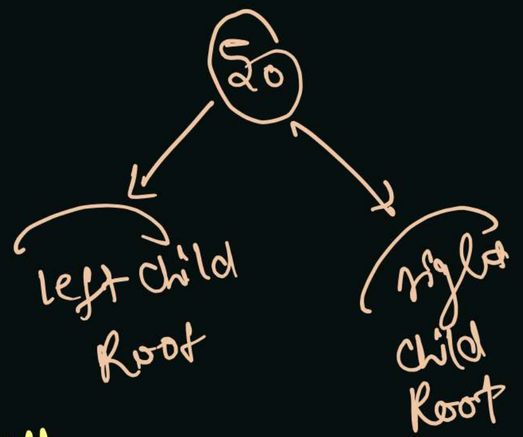
right part break

```
mid.right.left = null
mid.right = null
```

root = mid;

if()
mid.left != null

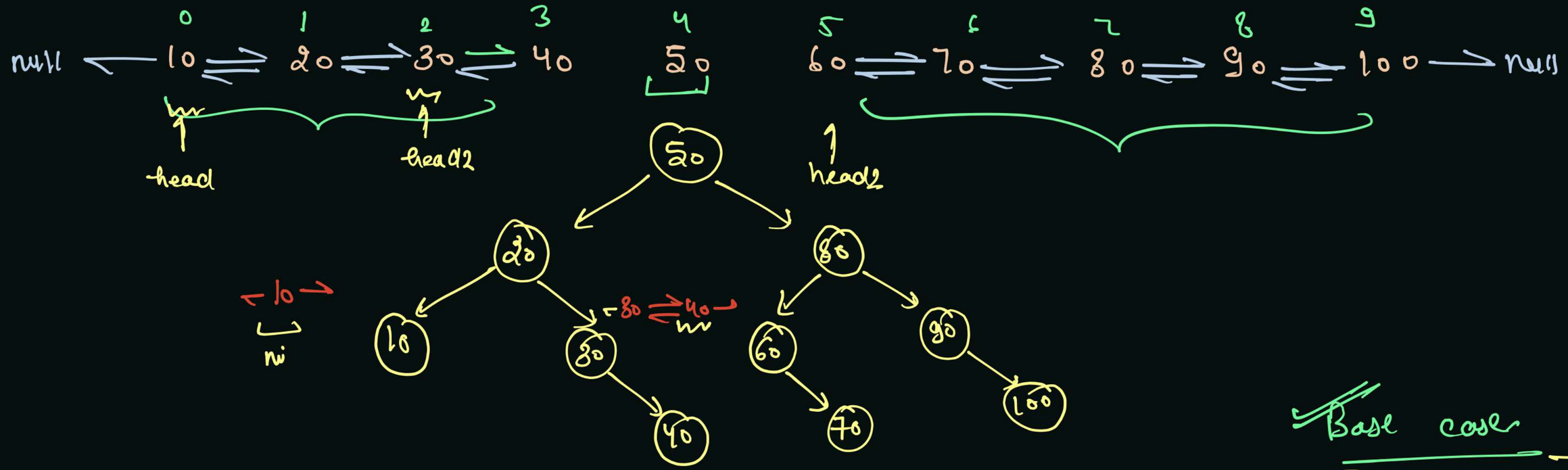
if()
mid.right != null



root.left = left child Root = creation(head);

root.right = right child Root = creation(head2);

return root;



Base case

head == null

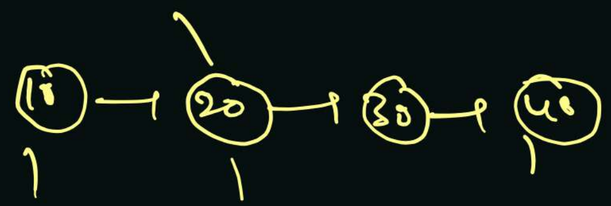
return null

mid == head

↳ Don't call toward left

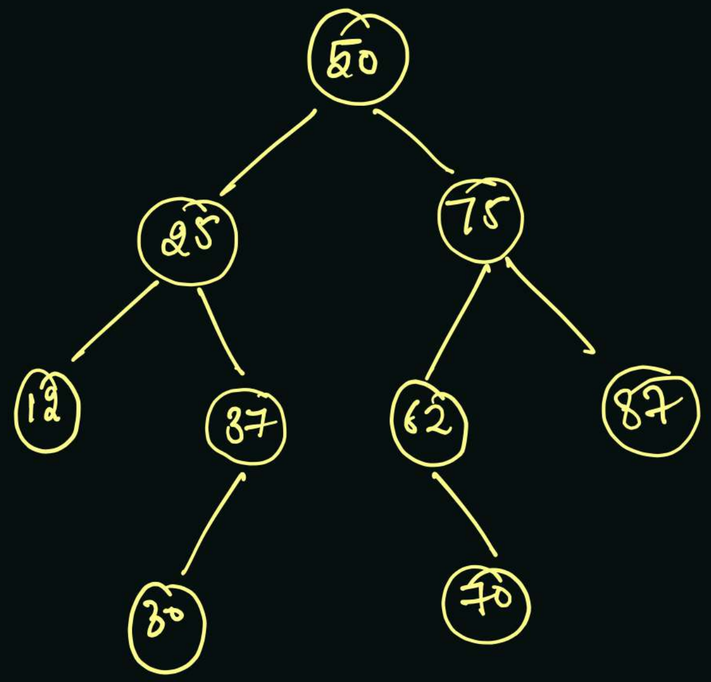
mid != head

↳ Make call for left side



Path Sum in Binary Tree

Sunday, 29 August 2021 10:17 AM



Root to leaf path having sum is equal to target value

if present → Return True

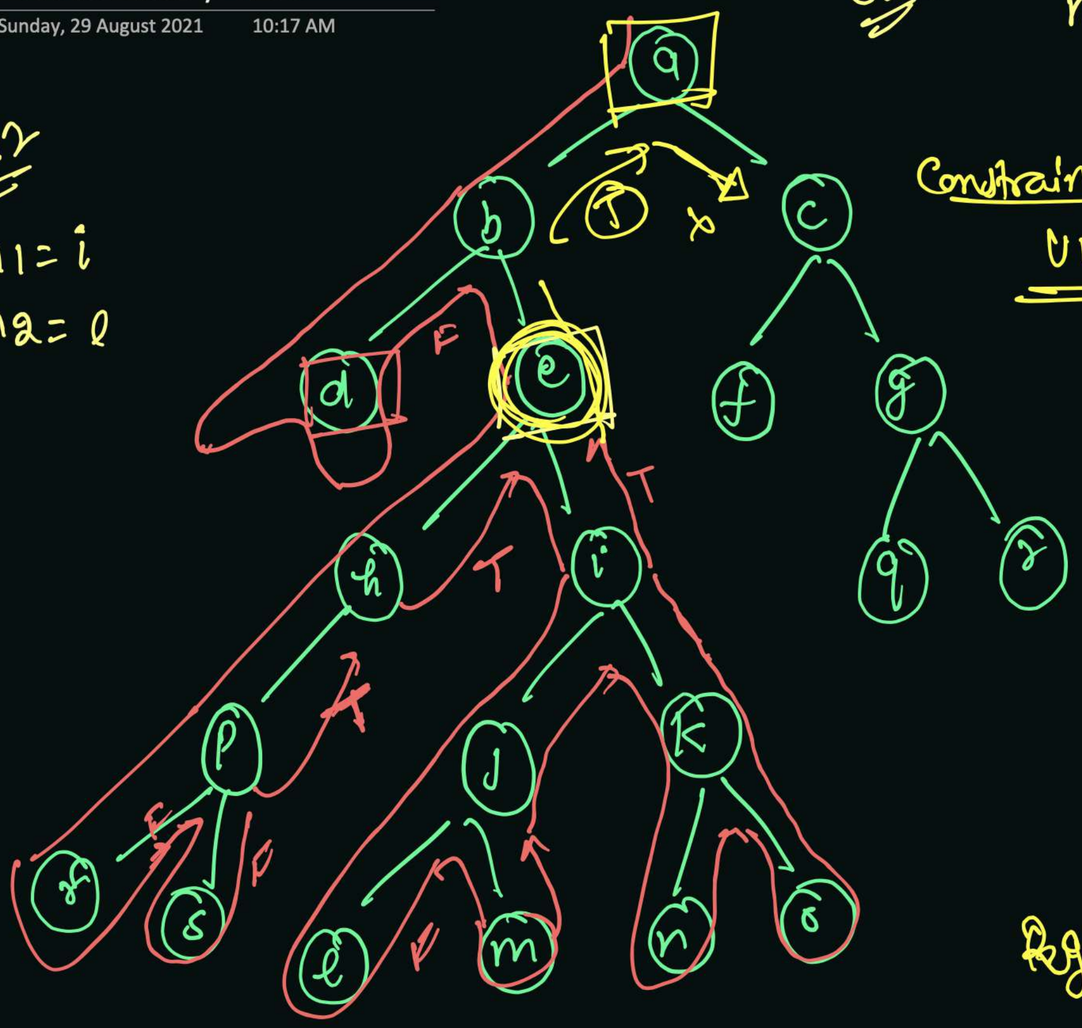
otherwise → Return False

Sum so far ⟶ if sf == target] return True

if True Encountered ⟶ No further calls

LCA in Binary Tree

Sunday, 29 August 2021 10:17 AM



Ex 1

node n_1 = p

node n_2 = m

Constraint → Unique data

Ex 2

$n_1 = i$

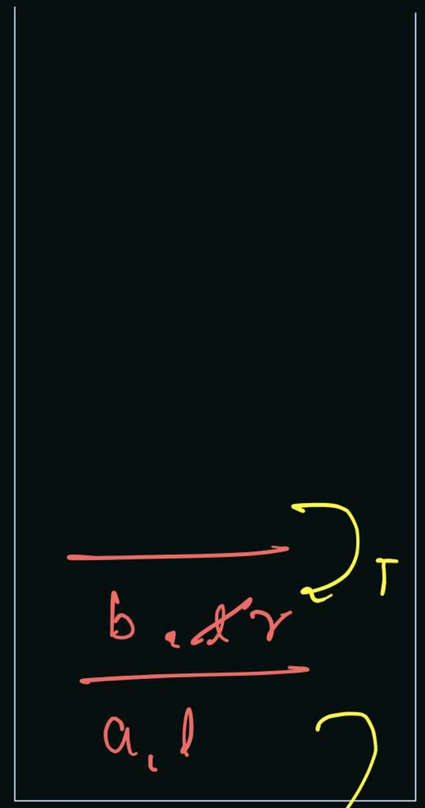
$n_2 = e$

Self check,
left check
right check
} Important

· How to stop next call →

if (LCA == null) make call

Reg



Set

↳ [self && left] || [self && Right] || [left && Right]

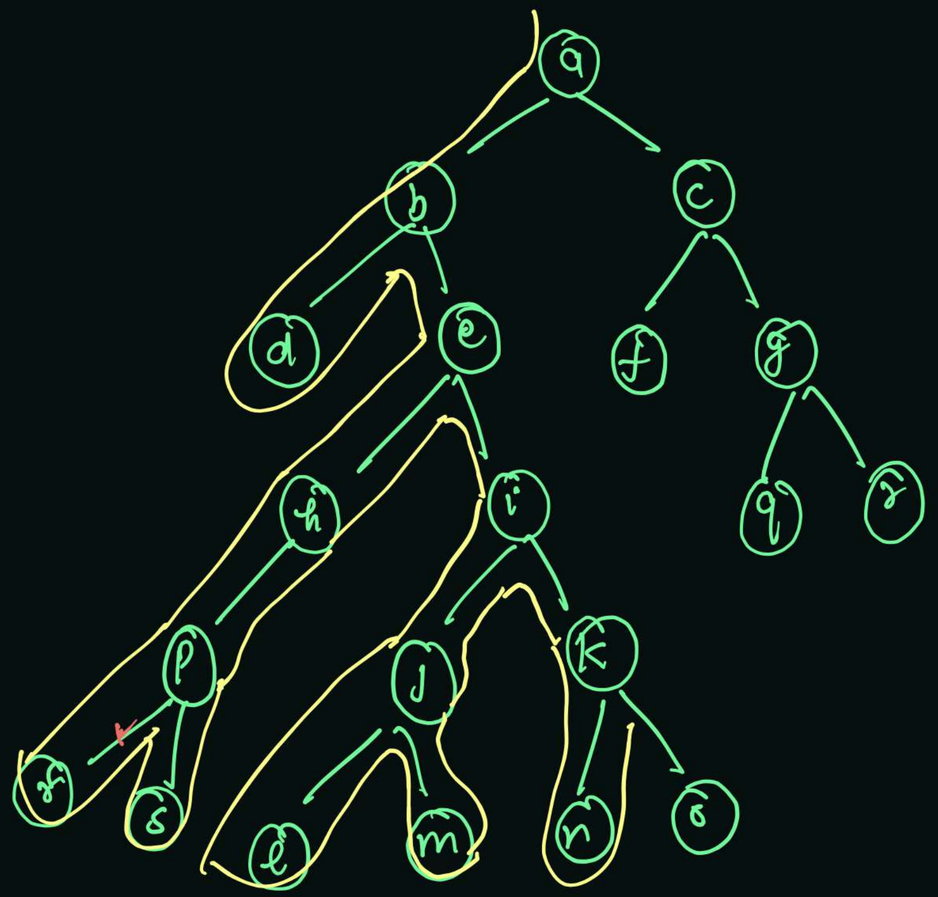
F T

selfcheck left Right check are.

Static Node LCA = null

LCA = e

data1 = i
data 2 = l



Static Node LCA = i

```
static TreeNode lca = null;

private static boolean solveLCA(TreeNode node, int data1, int da
    if(node == null) return false;
    boolean self = node.val == data1 || node.val == data2;
    boolean left = false, right = false;
    left = solveLCA(node.left, data1, data2);
    if(lca == null)
        right = solveLCA(node.right, data1, data2);
    if((self && left) || (self && right) || (left && right)) {
        lca = node;
    }
    return self || left || right;
}

public static TreeNode lowestCommonAncestor(TreeNode node, int p
    lca = null;
    solveLCA(node, p, q);
    return lca;
}
```

Self = True
left = True
Right = false.

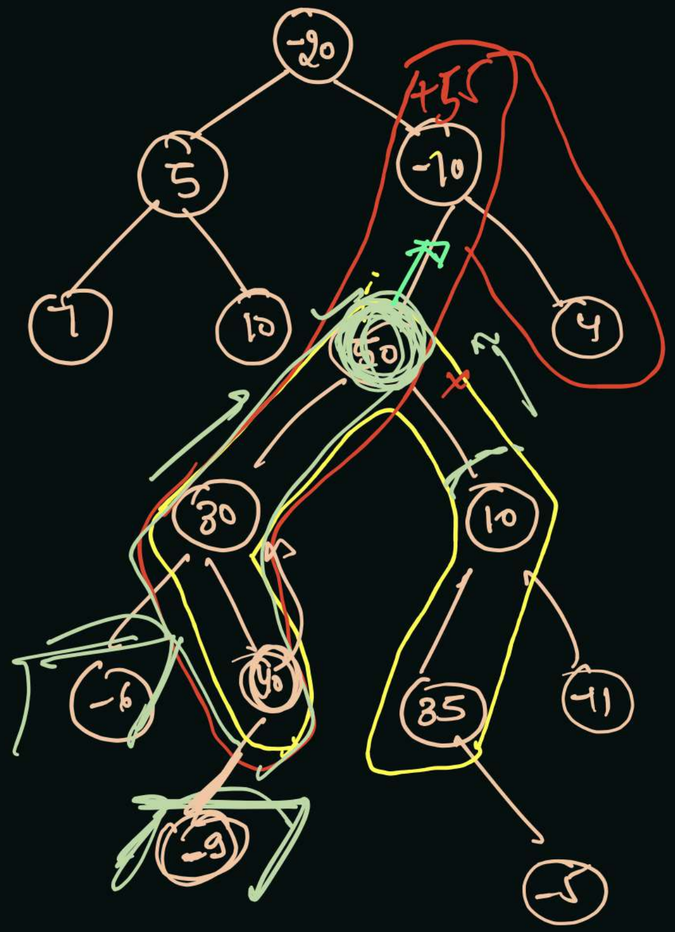
Self && left = T
Self && Right = F
left && Right = f

if data1 Not found
||
data2 Not found
LCA = null

e F F T
b F F T
a F

Maximum path Sum In Binary Tree

Sunday, 29 August 2021 10:17 AM



Max path sum from any Node to any Node.

Travel and change ↔

↳ overall max → val1

↳ left side subtree Root to node → val2

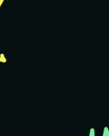
↳ right side subtree Root to node → val3

overall max

this figure is for overall max

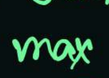
⇒ overall max

vs



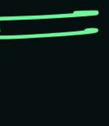
⇒ val1 + node.val + val2

vs



⇒ val1 + node.val

vs



⇒ node.val + val2

vs

node.val

What we return to parent →

val1 + node.val

vs

val2 + node.val

vs

node.val

Static int overallMax = $-\infty$ // Identity of max is $-\infty$

val1 = Max Path sum from left subtree Root to Node

val2 = Max path sum from Right subtree Root to Node

$-\infty + 1 = -\infty$

Fight for overall max

overallMax

vs

val1 + node.val

vs

val2 + node.val

vs

val1 + node.val + val2

vs

node.val

val

What should be return to parent

val1 + node.val

vs

val2 + node.val

vs

node.val

val ⇒ Return ans

Path Sum equal to given Value in Binary Tree

Sunday, 29 August 2021 10:17 AM

No. of subarray having sum equal to target.

HashMap →

target = 30

	0	1	2	3	4	5	6	7	8	9	10	11
array →		9	10	20	5	2	4	16	7	3	12	8	15
prefix Sum →	0	9	19	39	44	49	53	69	76	79	91	99	114
				39-30		49-30	53-30	69-30	76-30	79-30	91-30	99-30	114-30
				9		19	23	39	46	49	61	69	84

Bruteforce →

→ generate all sub arrays and check if sum is equal to target or not.

Sum = target

project(prefix Sum - target) == true

79-30 → 49

Count = 1 2 3 4 5

Count ++

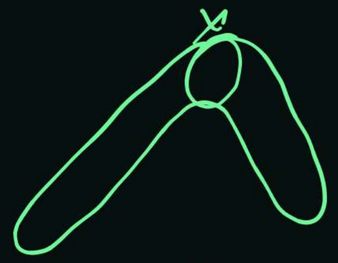
10 - 20 → 30

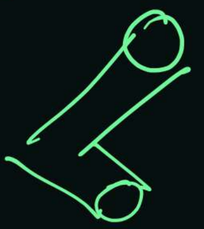
20 - 5 - 5 → 30

5 - 5 - 4 - 16 → 30

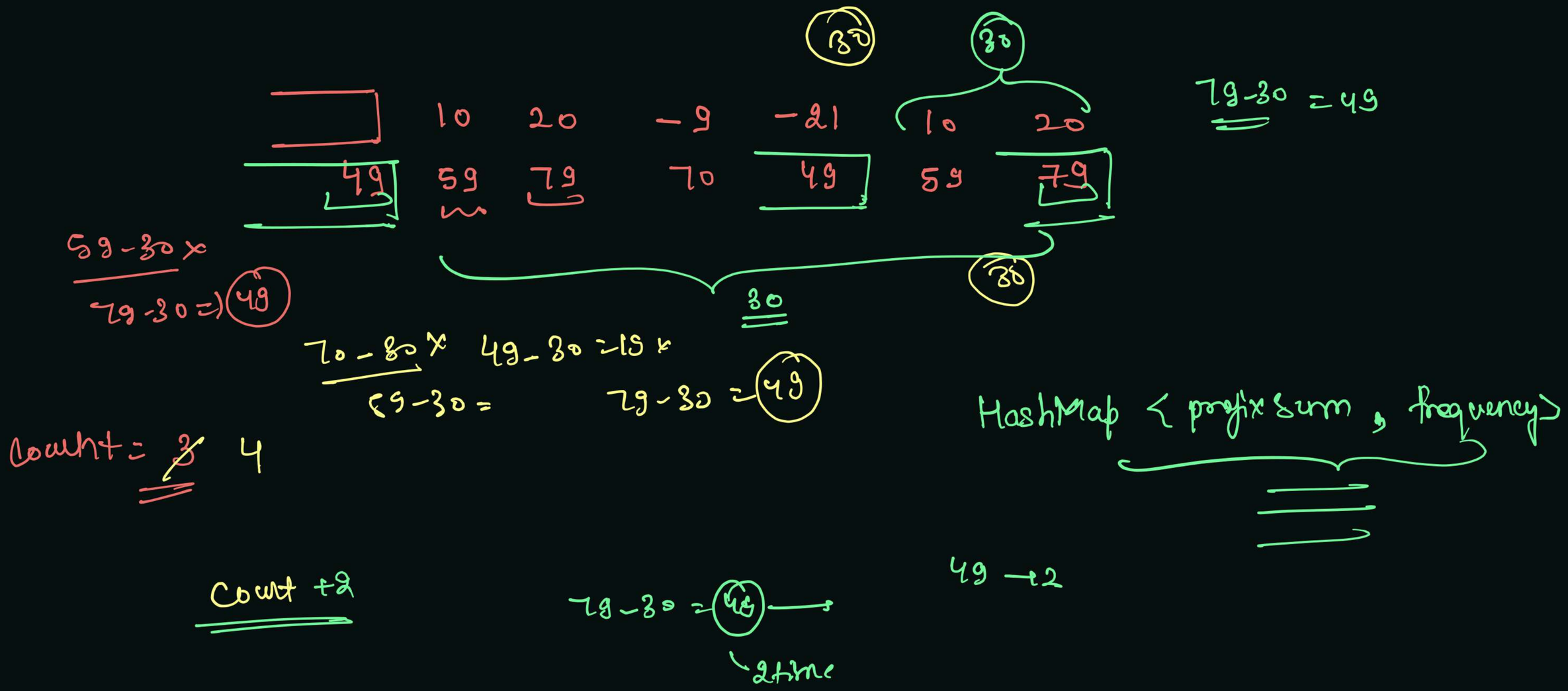
4 - 16 - 7 - 3 → 30

7 - 3 - 12 - 8 → 30



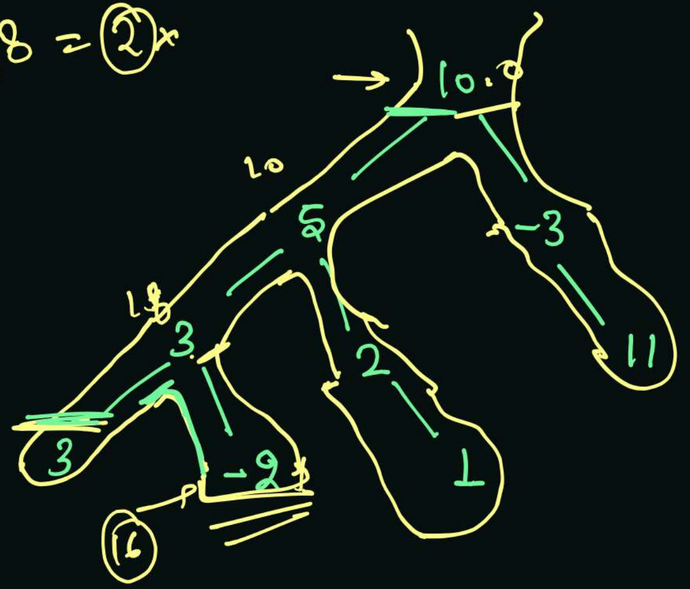


Manage prefix Sum variable and add prefix sum in HashSet

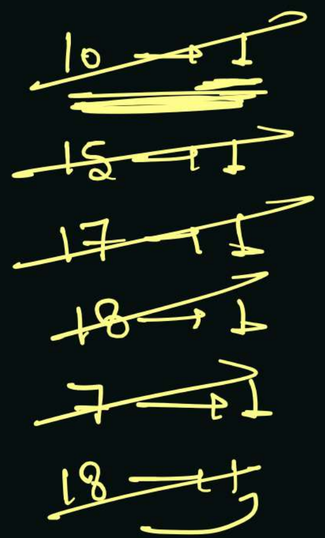

30
30
10 20 -9 -21 10 20
49 59 79 70 49 59 79
79-30 = 49
59-30 x
79-30 = 49
30
70-30 x 49-30 = 19 x
59-30 = 79-30 = 49
count = 3 4
HashMap < prefix sum , frequency >
Count +2
79-30 = 49
2 time
49 → 2

target Sum = 8

10 - 8 = 2



HashMap → prefix Sum vs freq.



17 - 8 = 9

18 - 8 = 10

7 - 8 = -1

18 - 8 = 16

ans = 3

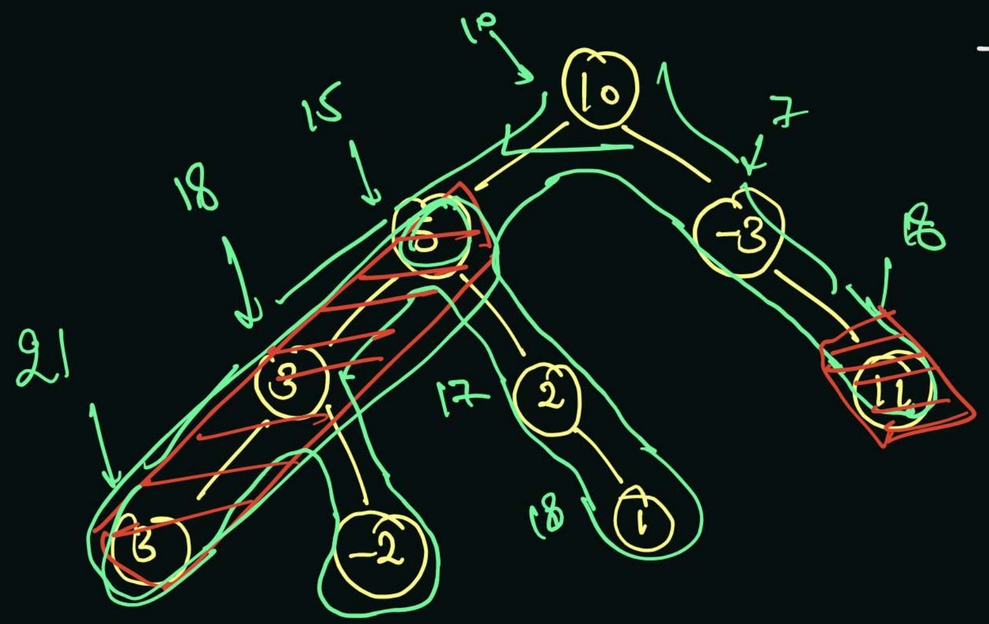
-3

Most Ans → map.put(prefix Sum, map.get(prefix Sum) - 1);

Decrease Freq.

if (map.get(prefix Sum) == 0)

map.remove(prefix Sum)



target = 11

prefix → arguments -

root → "

HashMap → "

count → Static

if prefixSum is available is HashMap then change freq.

Otherwise add prefix sum with value -1

prefixSum t = node.val

X map.put(

→ Change freq of prefix sum.

→ if freq became 0, then remove it from HashMap.

prefixSum	freq
10	1
7	1
18	1

ans = 0 ~~1~~ 2

11-

10-11 = -1

7-11 = -4 10-11 = 4

18-11 = 7

18-8 (7)

18-11 = 7 21-11 = 10

16-11 = 5

18-11 = 7 17-11 = 6