0 1 2 3 4 5 6

[6, 2, 5, 4, 5, 1, 6] n = 7

ls: [-1 -1 1 1 3 -1 5]

rs: [1 5 3 5 5 7 7]

area: [6 10 5 12 5 7 6] rs[i] - ls[i] -1

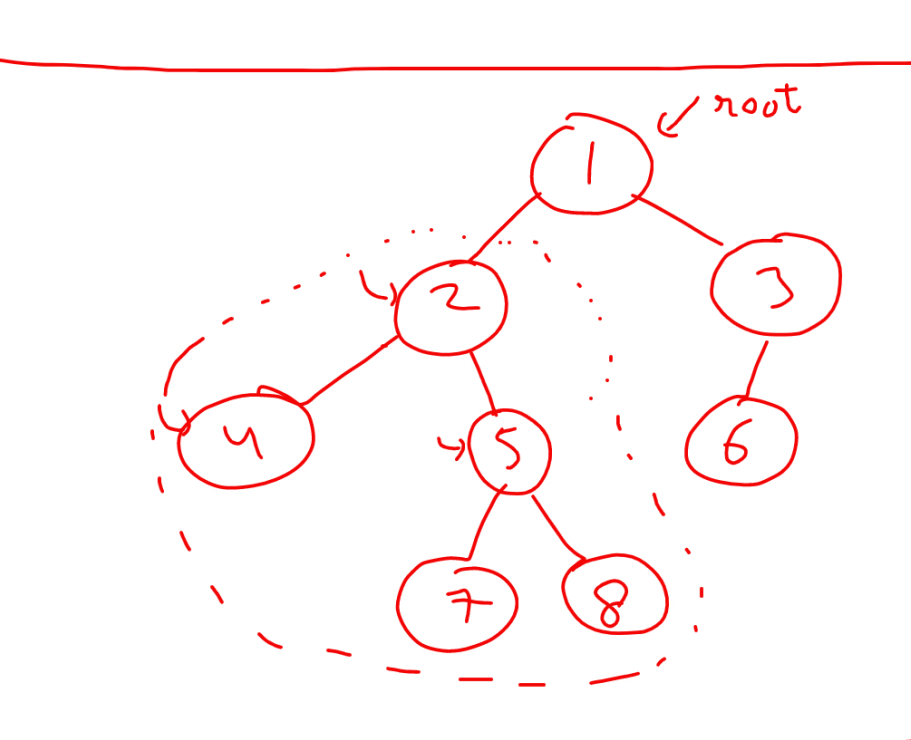
result.

O(n)

→ 1
→ 10
→ 11
100
101
110
111

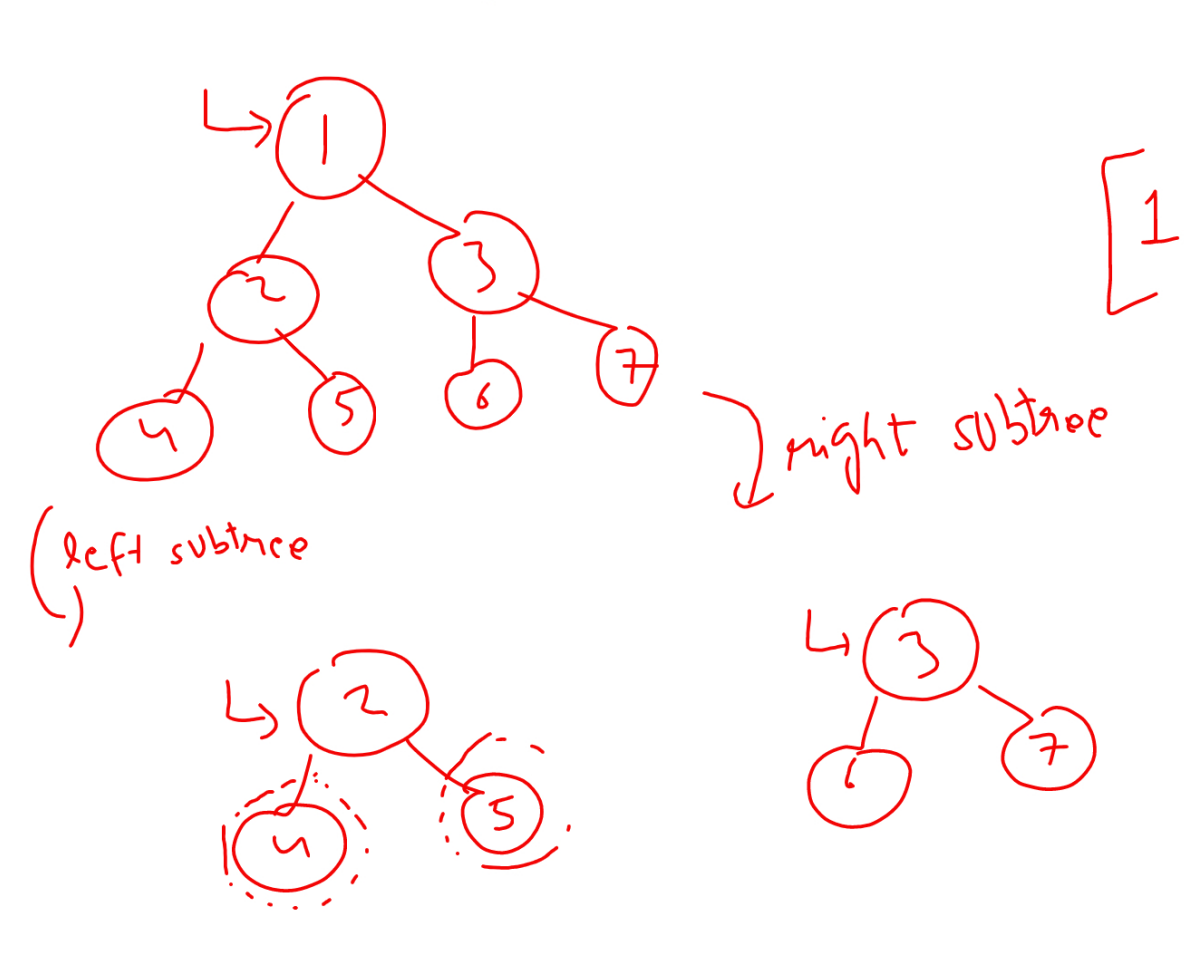
front → ~~1~~, ~~10~~, ~~11~~, 100, 101, 110, 111

[1, 10, 11]


root
1
2
3
4
5
6
7
8

Preorder:
→ Process the current node
→ " " left subtree
→ " " right subtree

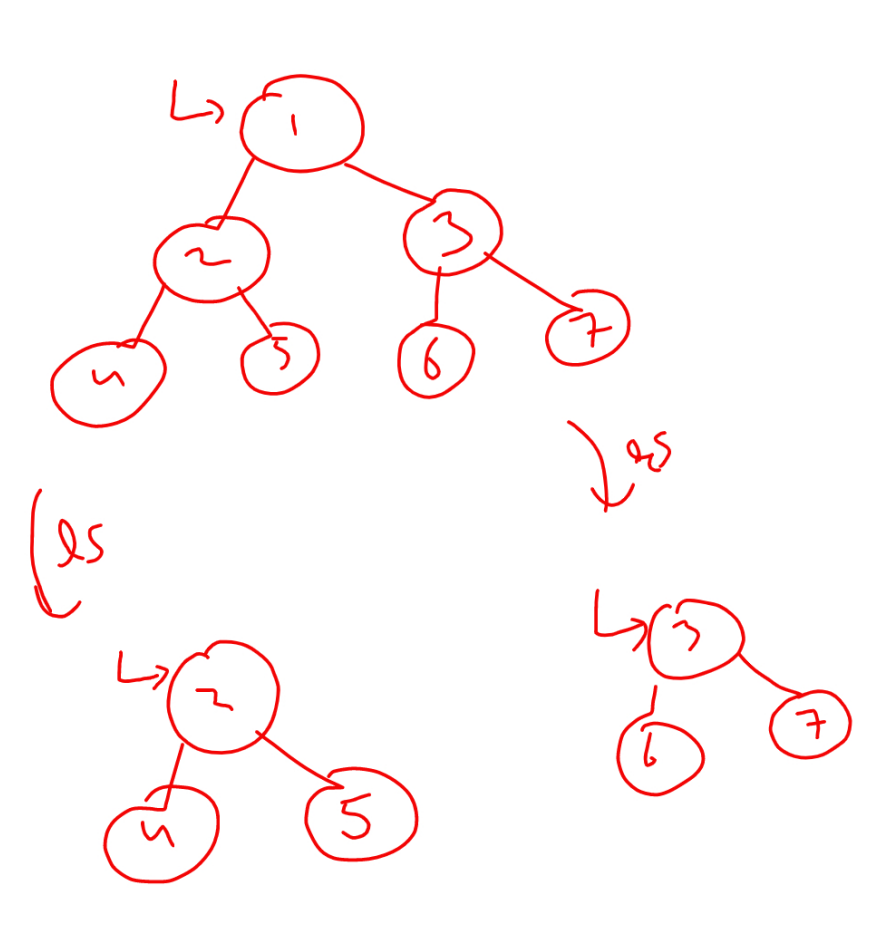
[1, 2, 4, 5, 7, 8, 3, 6]


1
2
3
4
5
6
7
right subtree
left subtree
2
4
5
3
6
7

[1, 2, 4, 5, 3, 6, 7]

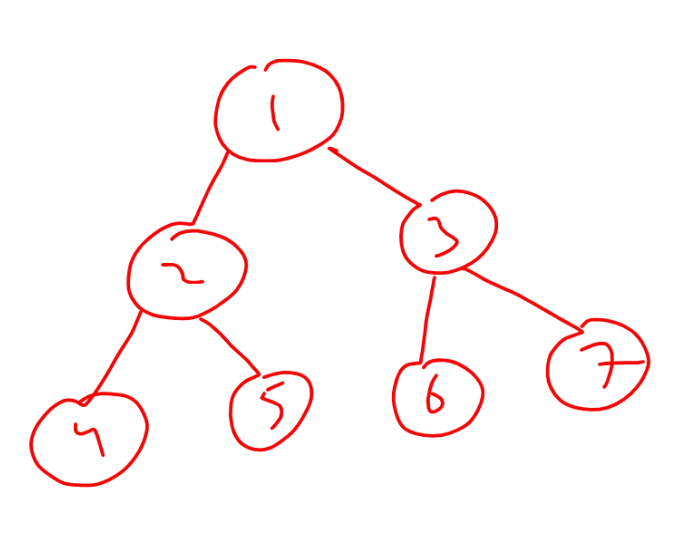
head

Postorder:
→ Process the left subtree
→ " " right subtree
→ " " current node


1
2
3
4
5
6
7
rs
ls
2
4
5
3
6
7

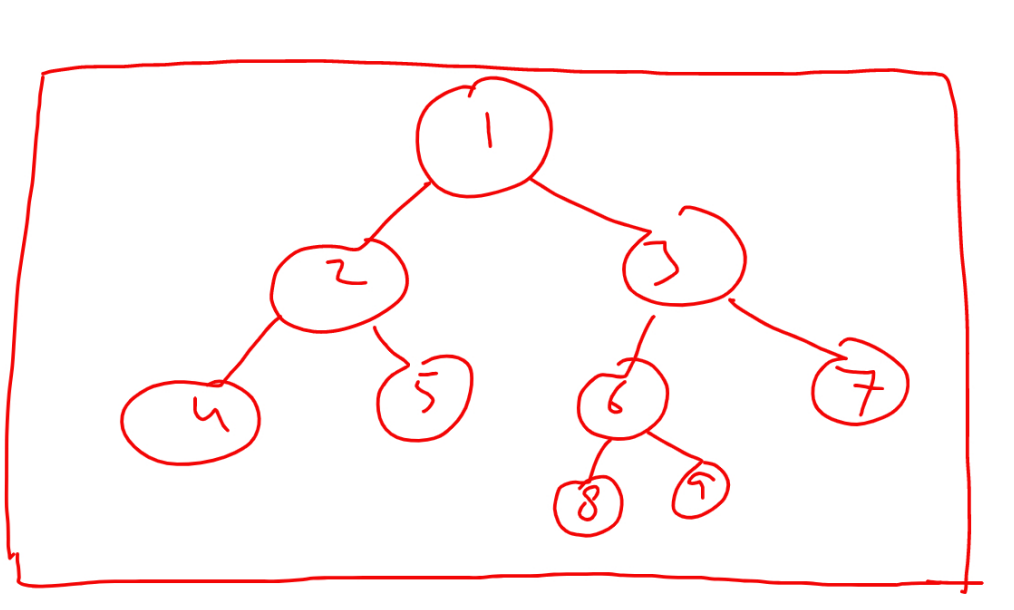
[4, 5, 2, 6, 7, 3, 1]

Inorder:
→ Process the left subtree
→ Process the current node
→ " " right subtree



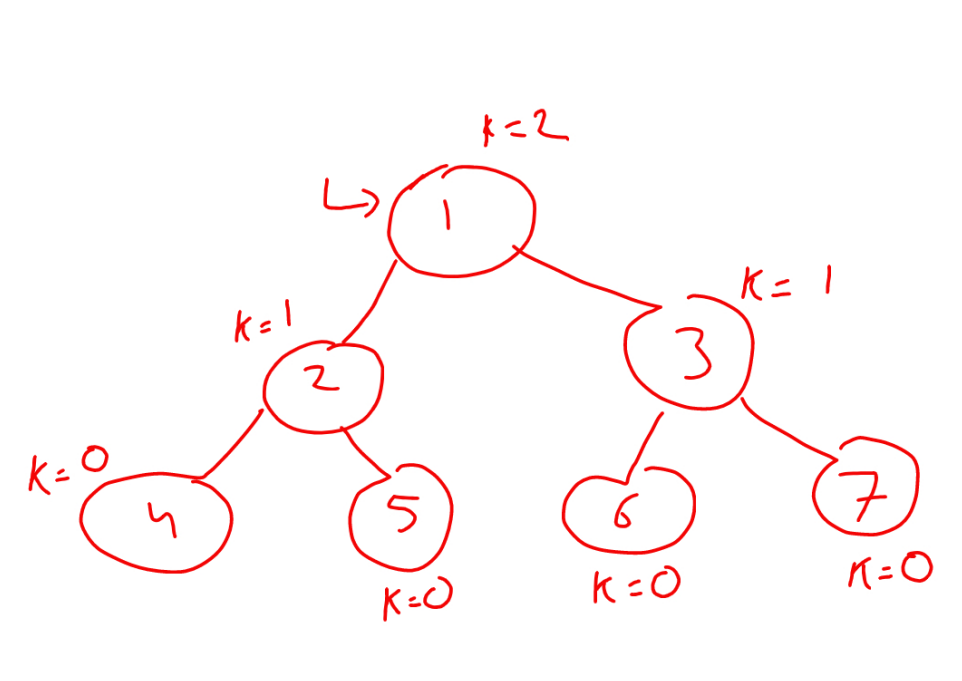
[4, 2, 5, 1, 6, 3, 7]

Level-order:



[1, 2, 3, 4, 5, 6, 7, 8, 9]

⇒ [get_kth_level(root, k)]


k=2
1
k=1
2
k=1
3
k=0
4
5
k=0
6
k=0
7
k=0

k=2

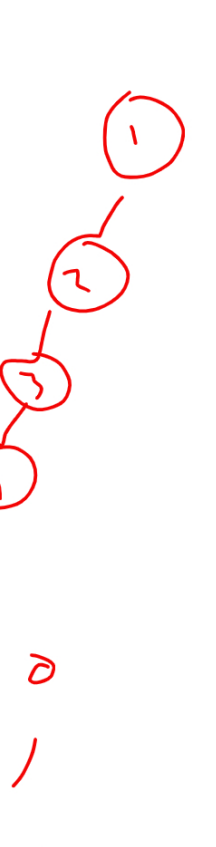
[1, 2, 4, 5, 3, 6, 7]

k=0 ⇒ [1]
k=1 ⇒ [2, 3]
k=2 ⇒ [4, 5, 6, 7]
k=3 ⇒ [ ]

$[2^0 + 2^1 + 2^2 + \dots + 2^{n-1}]$ 2^n

⇒ [4, 5, 6, 7]

[→ O(height * n)
→ O(n^2)]



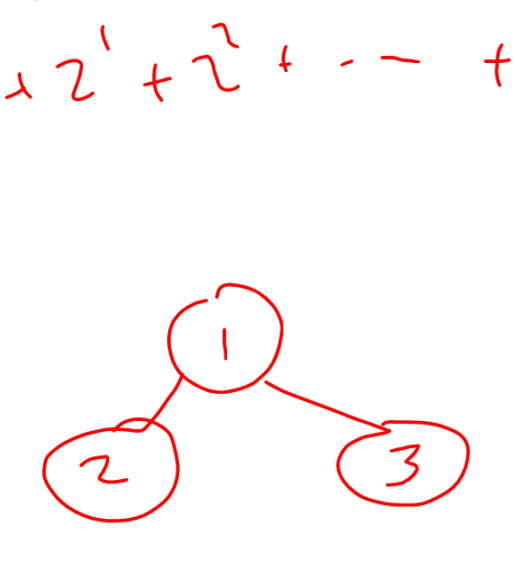
n n-1 0
0 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0

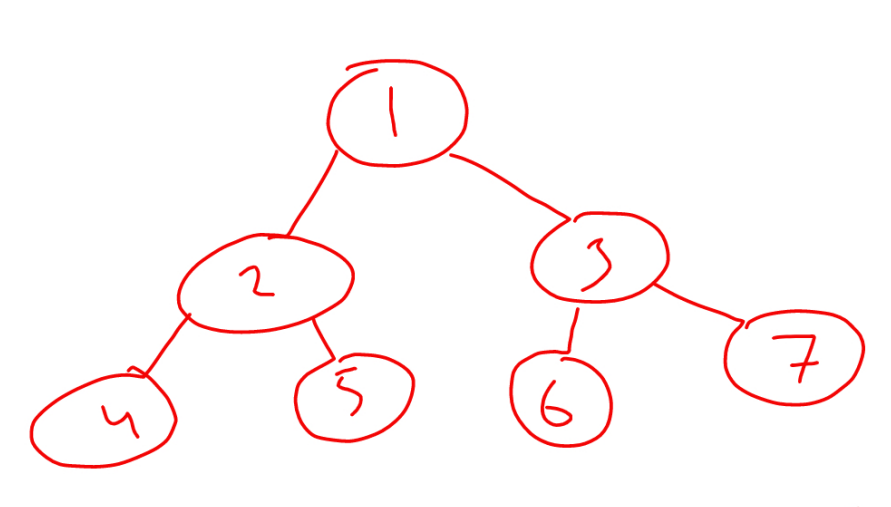
[0 → 1
0 → 2
0 → 3
...]

n^n

[0 → 1
0 → n]

n^2





[1, 2, 3, 4, 5, 6, 7]

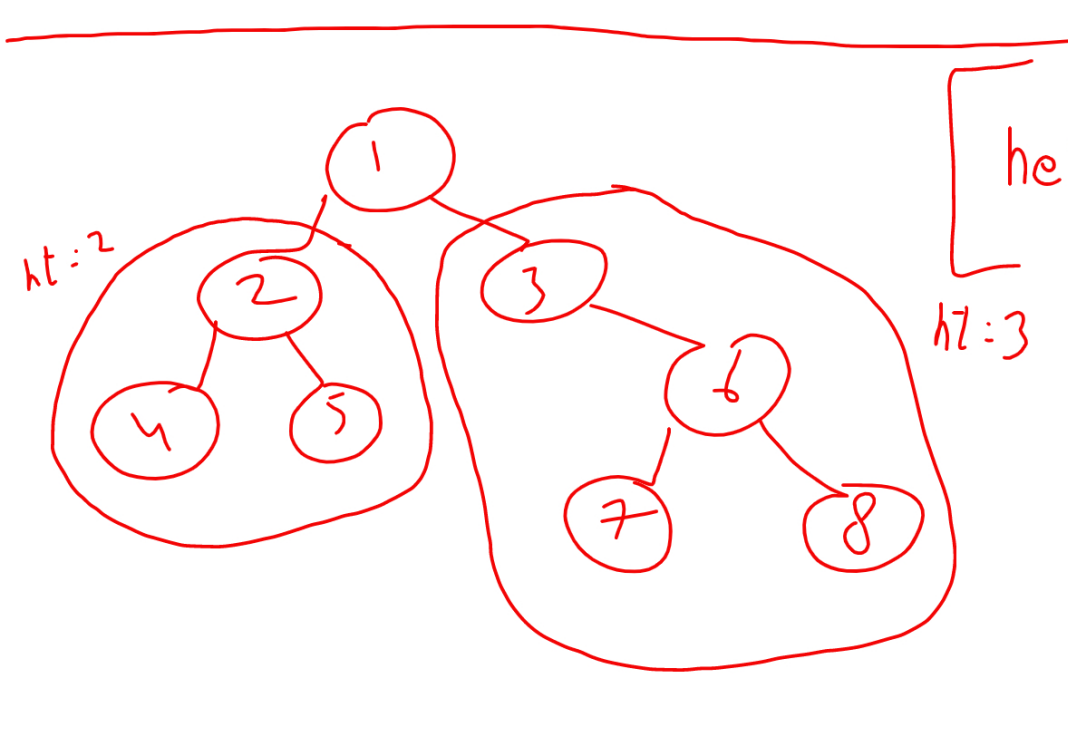
↓ ↓ ↓

1, 2, 3, 4, 5, 6, 7

queue: front → ~~1~~, ~~2~~, ~~3~~, ~~4~~, ~~5~~, ~~6~~, ~~7~~ STOP

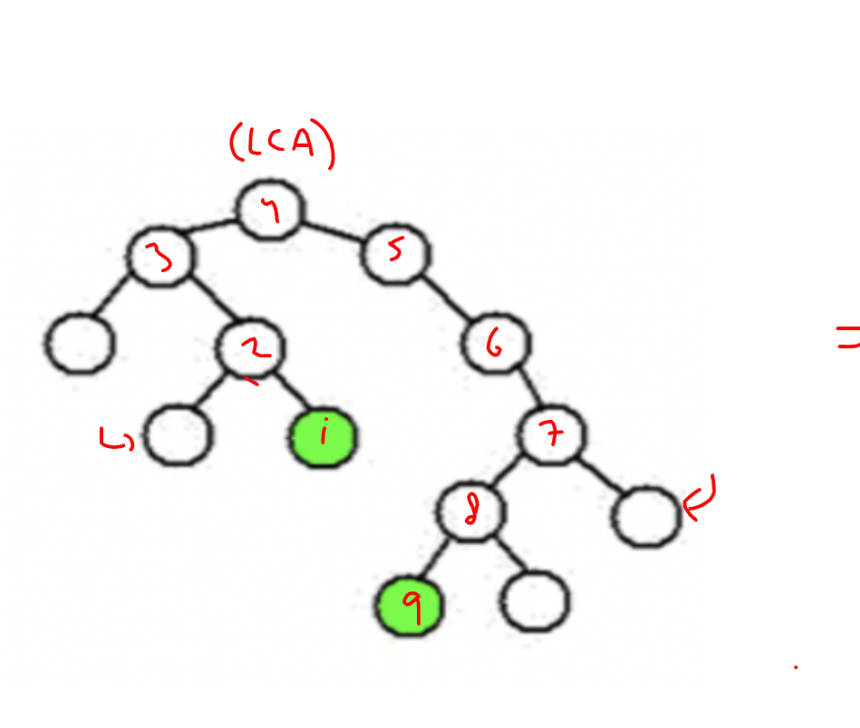
O(n)

res: [1, 2, 3, 4, 5, 6, 7]


1
ht:2
2
4
5
3
6
7
8
ht:3

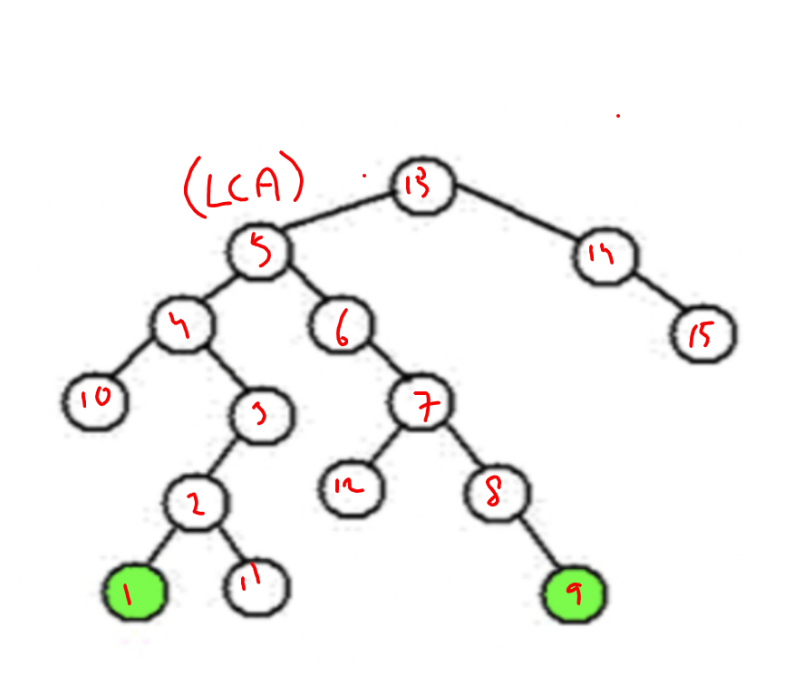
[height(1) = max(ht(2), ht(3)) + 1]

[height(root) = max(height(root→left), height(root→right)) + 1]



⇒ diameter = maximum no. of nodes connecting any two nodes in a BT.

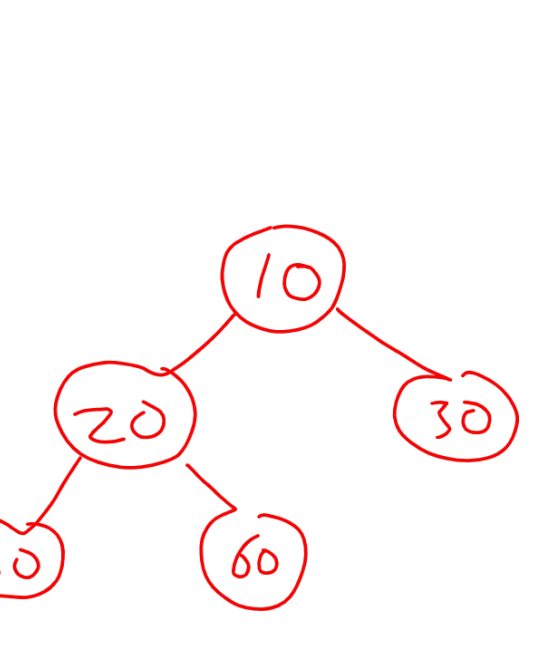
= 9

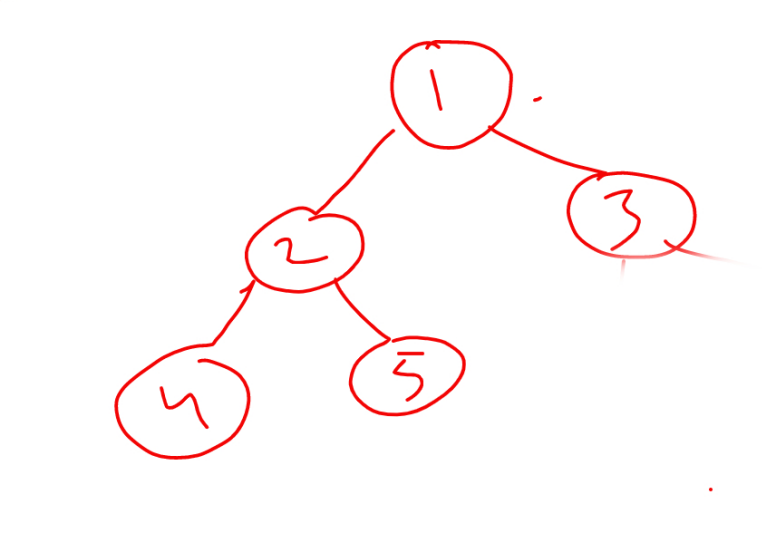

(LCA)
13
5
14
15
4
6
10
3
7
2
12
8
1
11
9

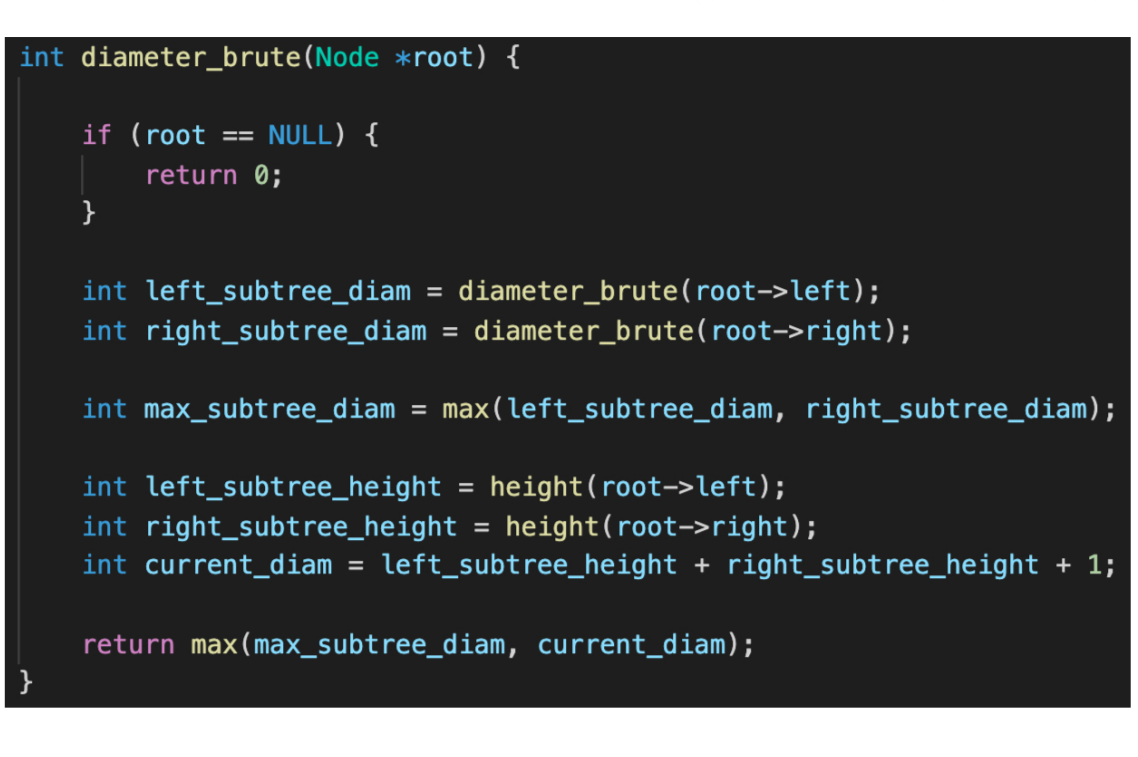
[LCA = 13
lh = 5, rh = 2
diam = 5+2+1 = 8]

[LCA = 5
lh = 4, rh = 4
diam = 4+4+1 = 9]

diameter(root) = max[diam of the left subtree,
diam of the right subtree,
left_ht + right_ht + 1]


10
20
30
40
60





```
int diameter_brute(Node *root) {

    if (root == NULL) {
        return 0;
    }

    int left_subtree_diam = diameter_brute(root->left);
    int right_subtree_diam = diameter_brute(root->right);

    int max_subtree_diam = max(left_subtree_diam, right_subtree_diam);

    int left_subtree_height = height(root->left);
    int right_subtree_height = height(root->right);
    int current_diam = left_subtree_height + right_subtree_height + 1;

    return max(max_subtree_diam, current_diam);
}
```

4

root = 1 → 1 → root = 3

3

root = 2 → 1 → root = 5 lh=1, rh=1

1

root = 4 0 lh=0, rh=0

0

root=N rt=N

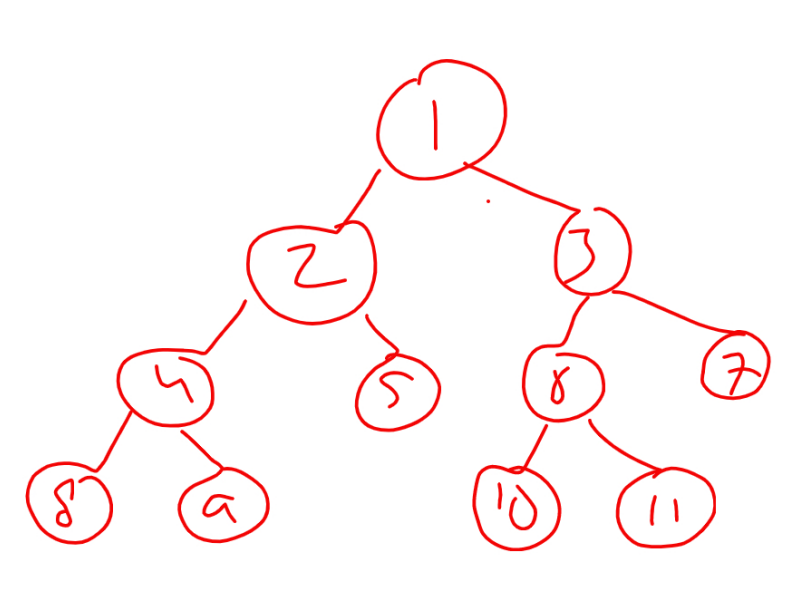
[ht(4)
ht(5)
ht(2)
ht(3)]

ht(4)
ht(5)

ht(8)
ht(9)
ht(4) → 8, 9
ht(5)

ht(10)
ht(11)
ht(6) → 10, 11
ht(7)

ht(2) → ht(4), ht(5), ht(8), ht(9)
ht(3) → ht(6), ht(7), ht(10), ht(11)


1
2
3
4
5
6
7
8
9
10
11

⇒ [Node* → int]

O(n^2) → O(n)

[max of (lh + rh + 1) value for any node in the BT]

height ⇒ [lh
rh]

(lh + rh + 1)