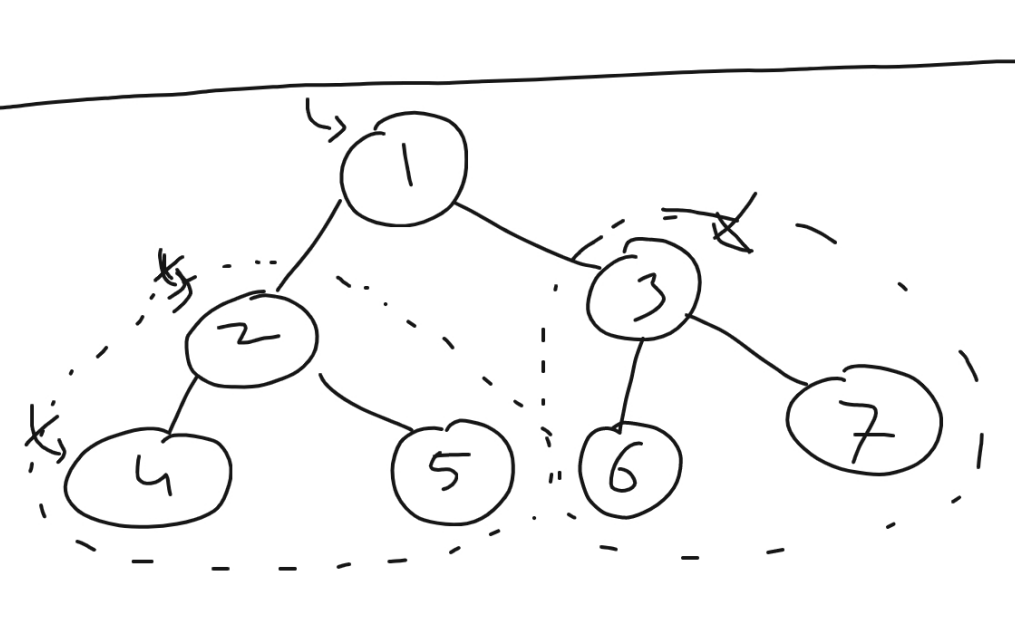
BINARY TREES

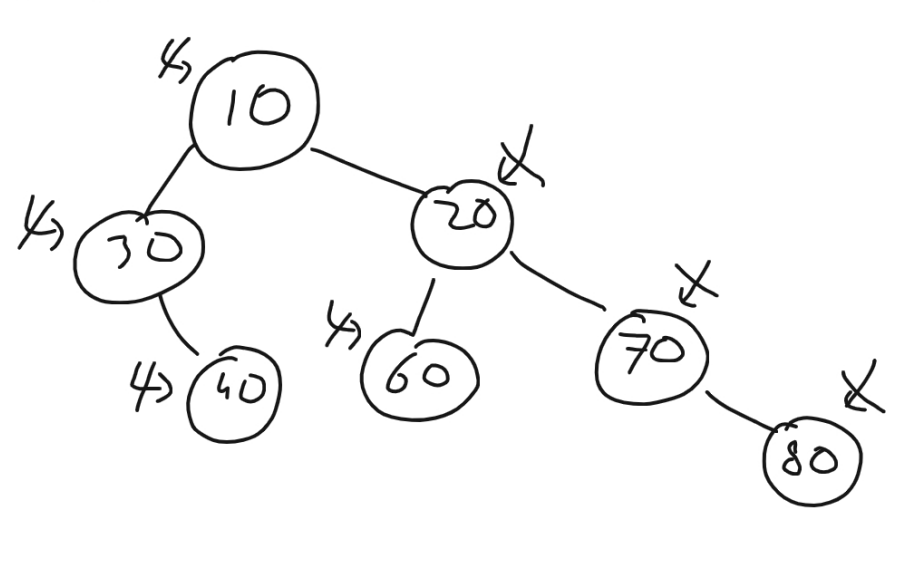
```
class TreeNode {
  public:
      int data;
      TreeNode* left;
      TreeNode* right;

};
```



Preorder Traversal:

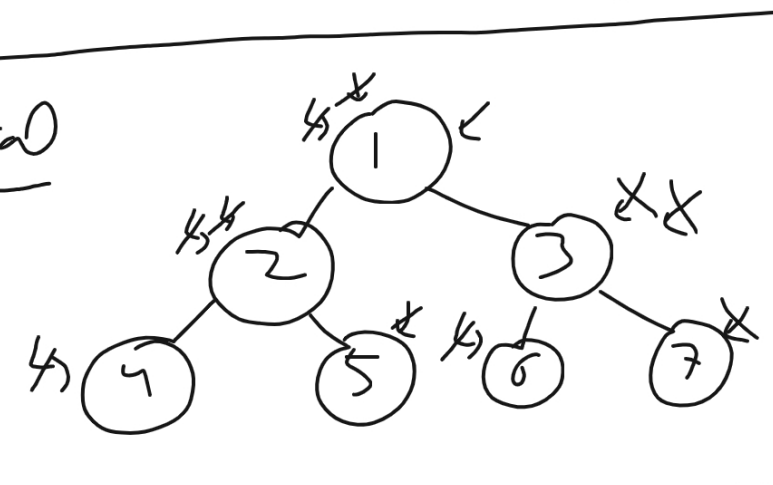
→ Process the current node
→ Recursively process the left subtree
→ Recursively process the right subtree

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


10
30
20
40
60
70
80

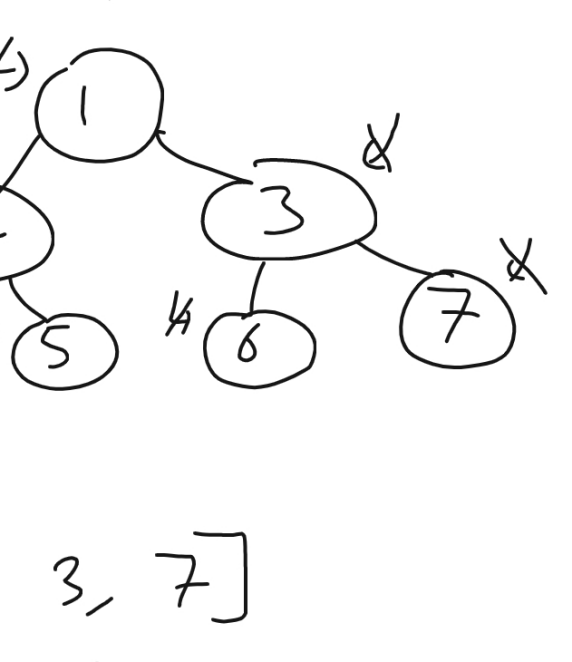
[10, 30, 40, 20, 60, 70, 80]

Post order Traversal



→ Recursively process the left subtree
→ Recursively process the right subtree
→ Process the current node.

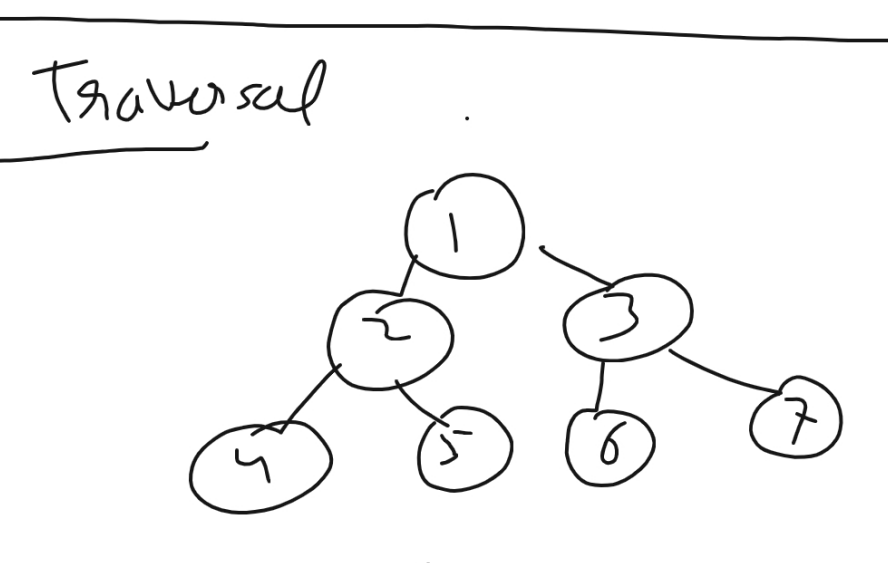
[4, 5, 2, 6, 7, 3, 1]


10
30
20
40
60
70
80

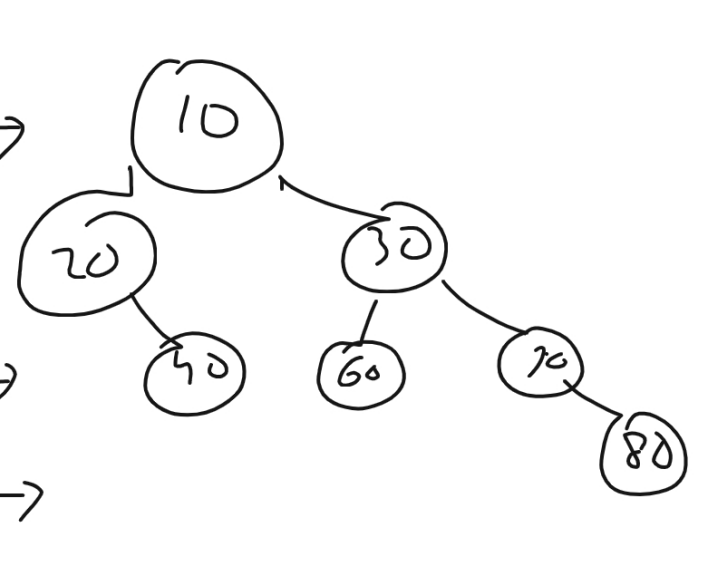
[40, 30, 60, 80, 70, 20, 10]

Inorder Traversal

→ Recursively process the left subtree
→ Process the current node
→ Recursively process the right subtree

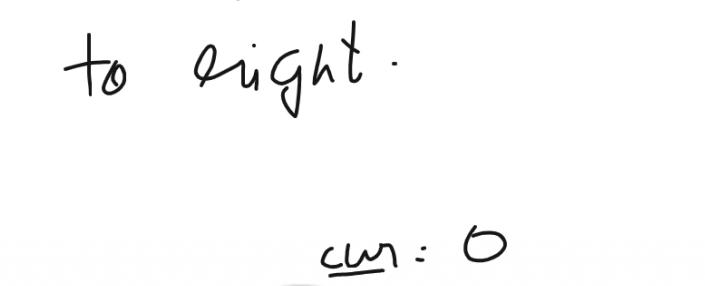


[4, 2, 5, 1, 6, 3, 7]

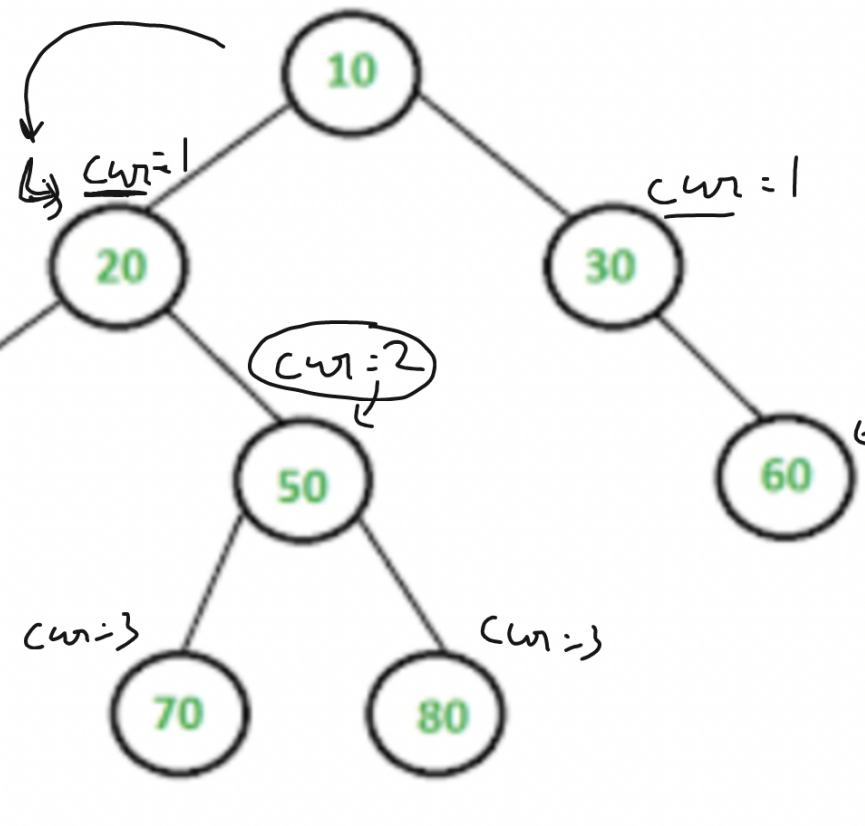

10
30
20
40
60
70
80

[30, 40, 10, 60, 20, 70, 80]

Level-order Traversal

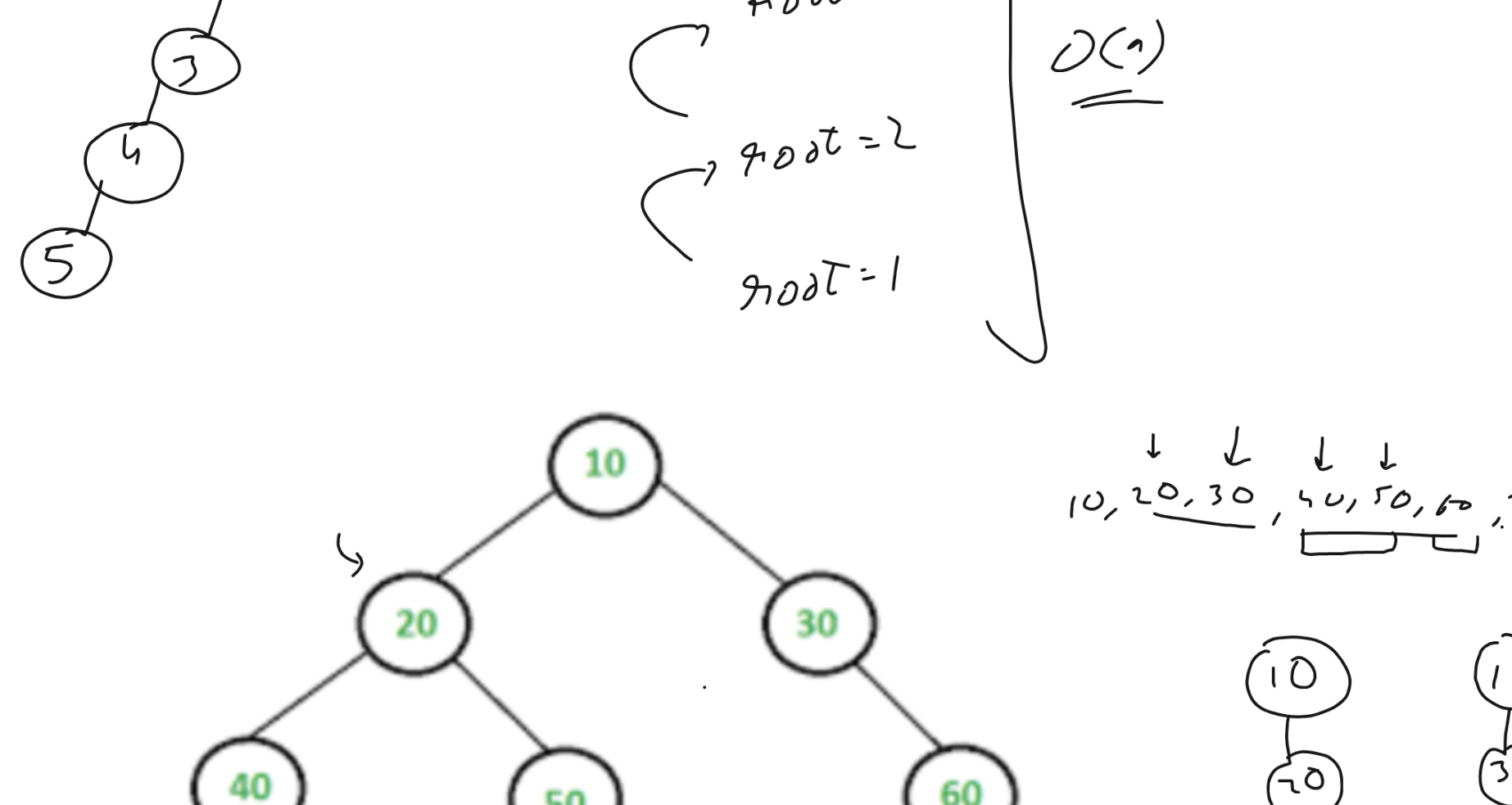


[1, 2, 3, 4, 5, 6, 7]

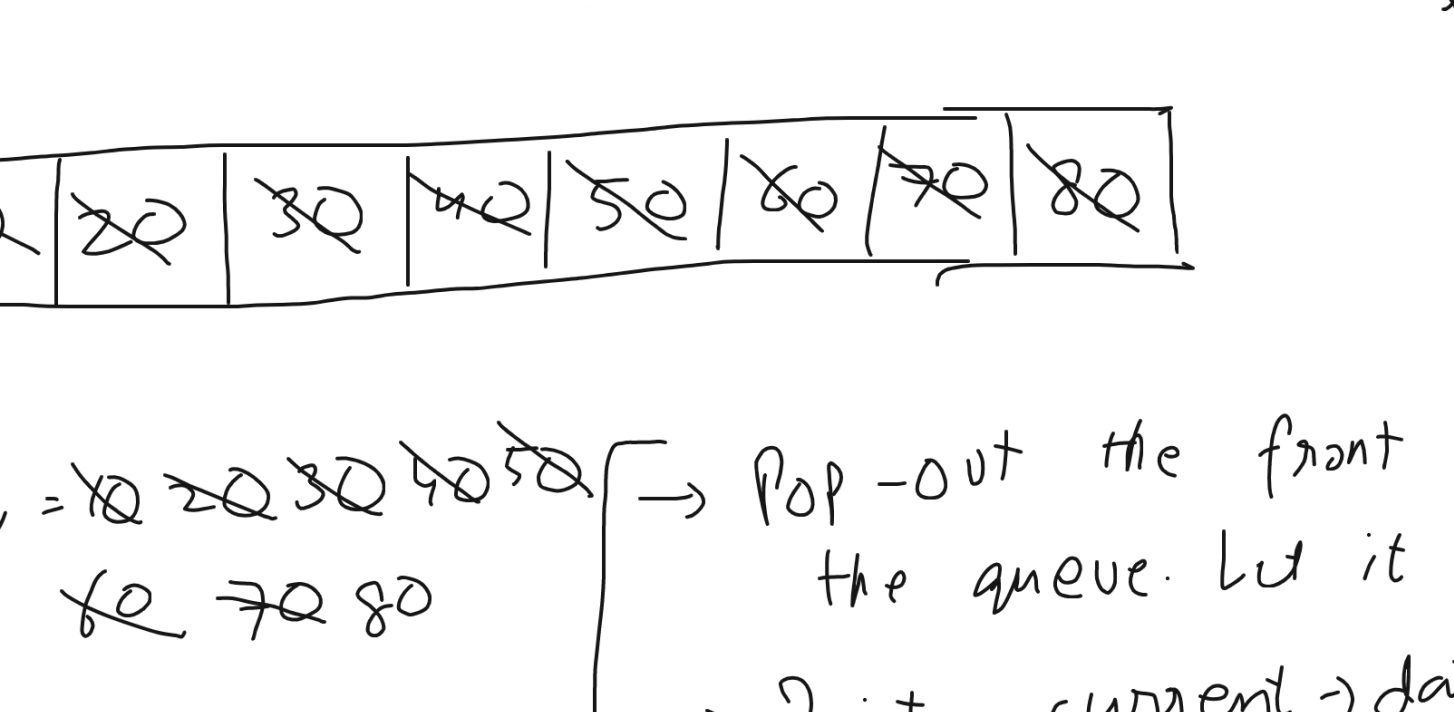

10
20
30
40
60
70
80

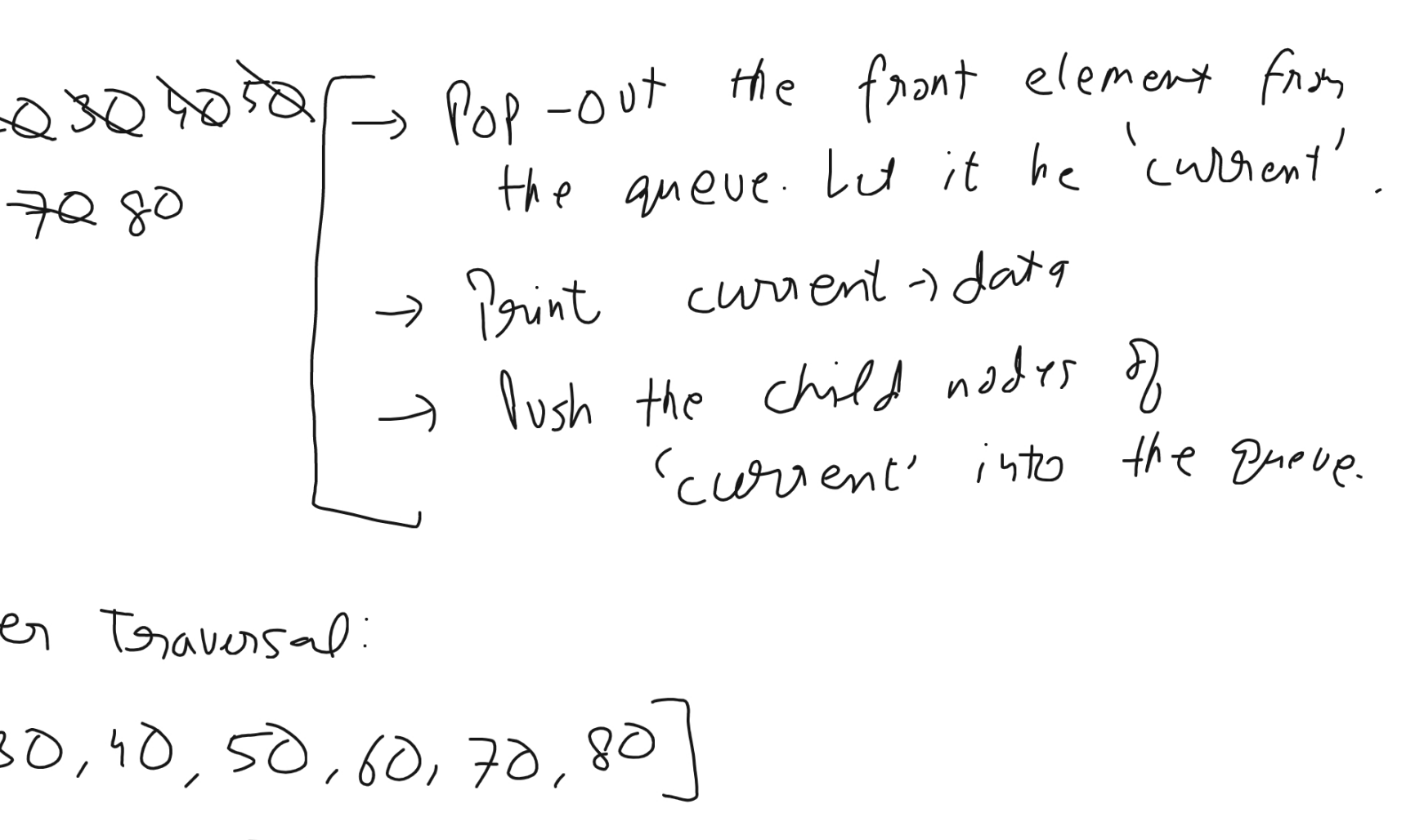
[10, 20, 30, 40, 60, 70, 80]

Write a function to print the k-th level of a Binary Tree from left to right.


k=0
k=1
k=2
k=3
curr: 0
curr=1
curr=1
curr=2
curr=2
curr=2
curr=3
curr=3
10
20
30
40
50
60
70
80
curr
[40, 50, 60]

Preorder: [10, 20, 40, 50, 70, 80, 30, 60]


1
2
3
4
5
root=5
root=4
root=3
root=2
root=1
O(n)


10
20
30
40
50
60
70
80
10, 20, 30, 40, 50, 60, 70, 80
left-view
right-view

~~10~~	~~20~~	30	~~40~~	~~50~~	~~60~~	~~70~~	80

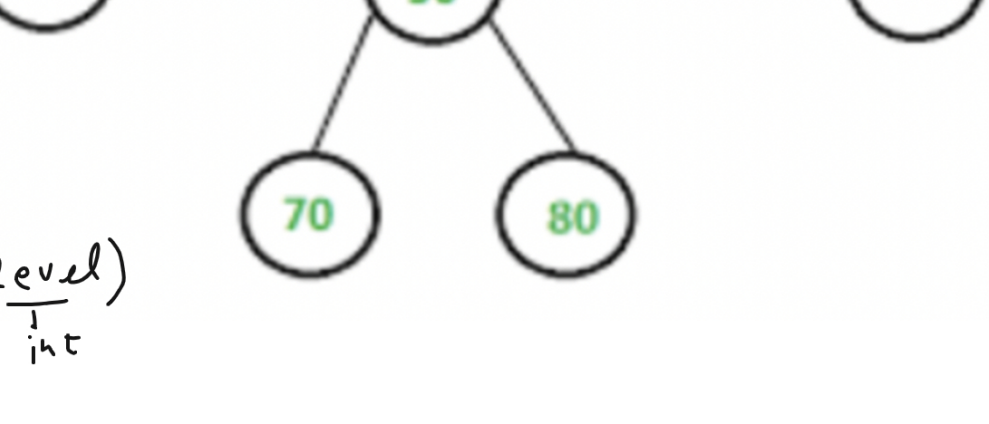
current = ~~10~~ ~~20~~ 30 ~~40~~ ~~50~~ ~~60~~ ~~70~~ 80

→ Pop-out the front element from the queue. Let it be 'current'.
→ Print current → data
→ Push the child nodes of 'current' into the queue.

Level-order Traversal:

[10, 20, 30, 40, 50, 60, 70, 80]

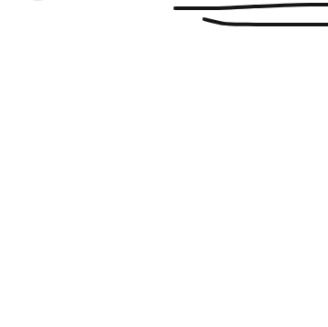
TC = O(n)
Aux. space: O(n)



15.

no. of nodes in the last level = $2^{(h-1)}$

Total no. of nodes: $\underbrace{2^0 + 2^1 + 2^2 + \dots + 2^{h-1}}_{\text{GP with } a=1,\ r=2,\ \text{no. of terms}=h} = \frac{1(2^h-1)}{2-1} = 2^h - 1$


10
20
30
40
50
60
70
80

0 → ~~False~~ True
1 → ~~F~~ True
2 → ~~F~~ True
3 → ~~F~~ True

pair <TreeNode*, level>
int

~~10, 0~~	~~20, 1~~	~~30, 1~~	~~40, 2~~	~~50, 2~~	~~60, 2~~	~~70, 3~~	~~80, 3~~

current_node = ~~10~~ ~~20~~ ~~30~~ ~~40~~ ~~50~~ ~~60~~ ~~70~~ 80
current_level = ~~0~~ ~~1~~ ~~1~~ ~~2~~ ~~2~~ ~~2~~ ~~3~~ 3

[10, 20, 40, 70]