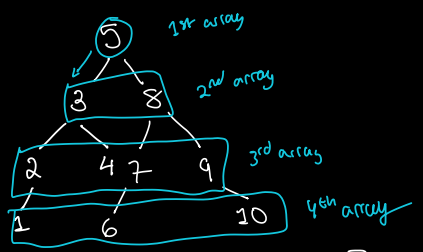
Problem: Given binary tree, return LEVEL order traversal of nodes' values as an array -> bfs

Intro:
- Verify Constraints
- Create Testcases



[[5], [3,8], [2,4,7,9], [1,6,10]]

bfs returns [6,3,8,2,4,7,9,1,6,10]

at every level

add #s to new list

Optimal:
- Brainstorming & Pattern Observations

[5]

if curr.left
queue: 3
if curr.right: 8
queue: [3,8]

append(queue)

[3]

if curr.left
queue: 8,2
if curr.right
queue: 8,2,4

BFS pseudo:

res = [], q = [root]
while (q):
node = q[0]
q = q[1:]
res.append(node.val)
if node.left:
q.append(node.left)
if node.right:
q.append(node.right)

1 identify level
2 initialize array

return res;

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

len = 1
count = 1
↳ list.append([3])

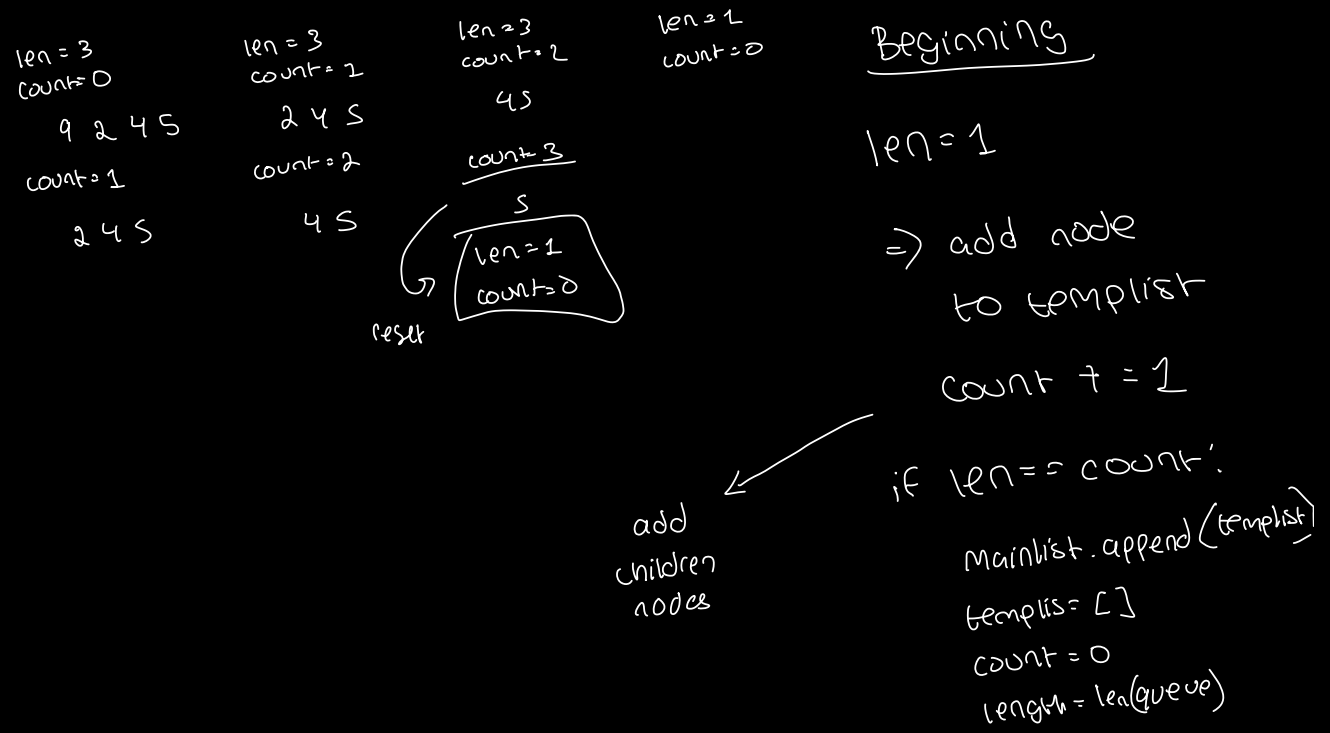
add the children
[6,1]

len = 2
count = 0
6 1 9 2
count = 1
1 9 2

len = 2
count = 1
1 9 2 4
count = 2
len = 3 => count = 0
reset length and count

q[9,2,4]

len = 3
count = 0



- Pseudocode
- Write code
- Run through testcases
- Analyze time and space complexity
  Time: O(n) (going thru all nodes)
  Space: O(n)