TREES

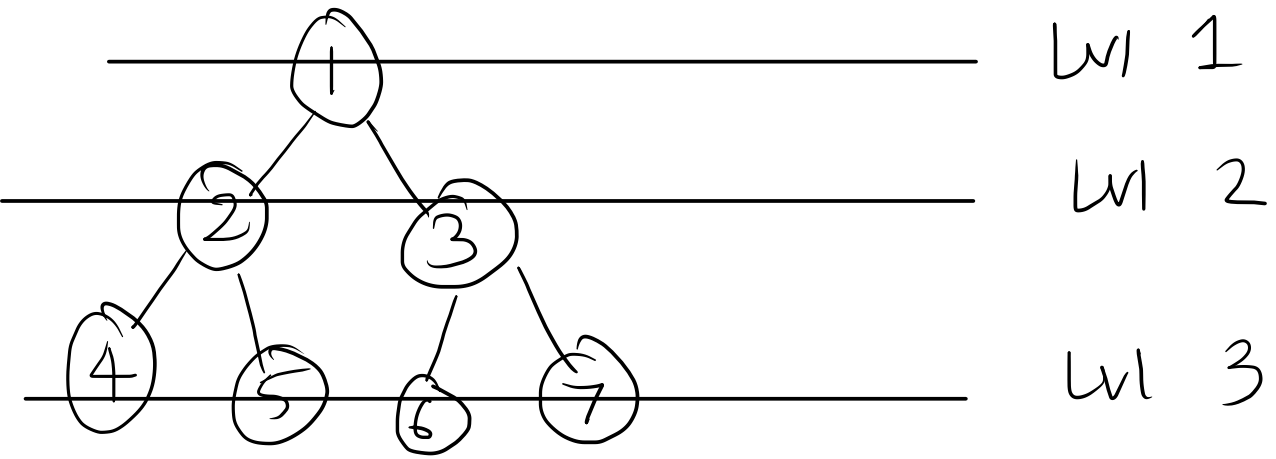
- Basics ✓
- Traversal ✓
  - DFS
    - Preorder
    - Inorder
    - Postorder
  - * BFS → Level Order Traversal
- * Height of Tree

TREES

- Basics ✓
- Traversal ✓
  - DFS
    - Preorder
    - Inorder
    - Postorder
  - *BFS → Level Order Traversal
- *Height of Tree

Height of the Tree

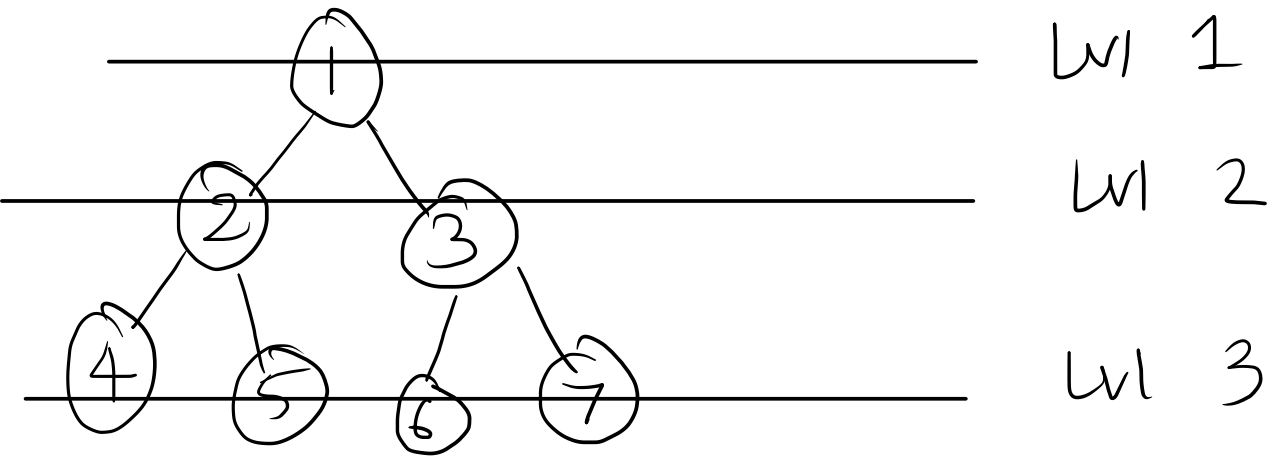
→ no. of levels



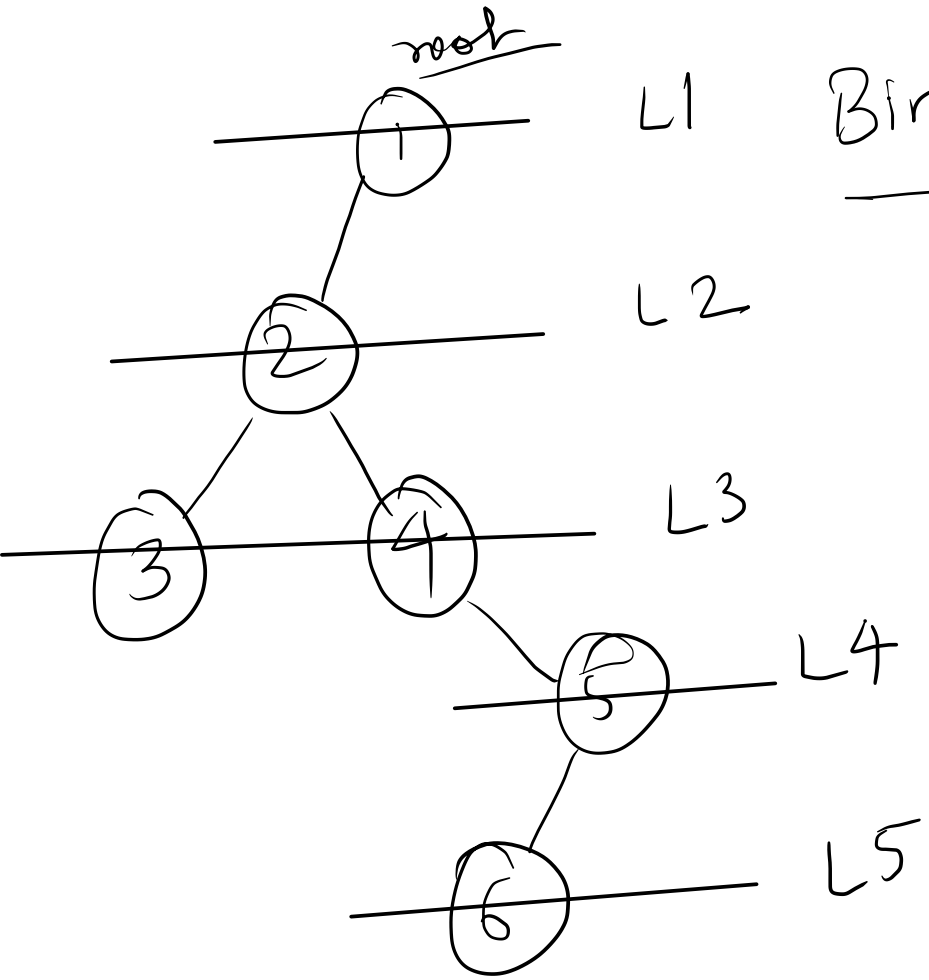
Height = 3

Height of the Tree

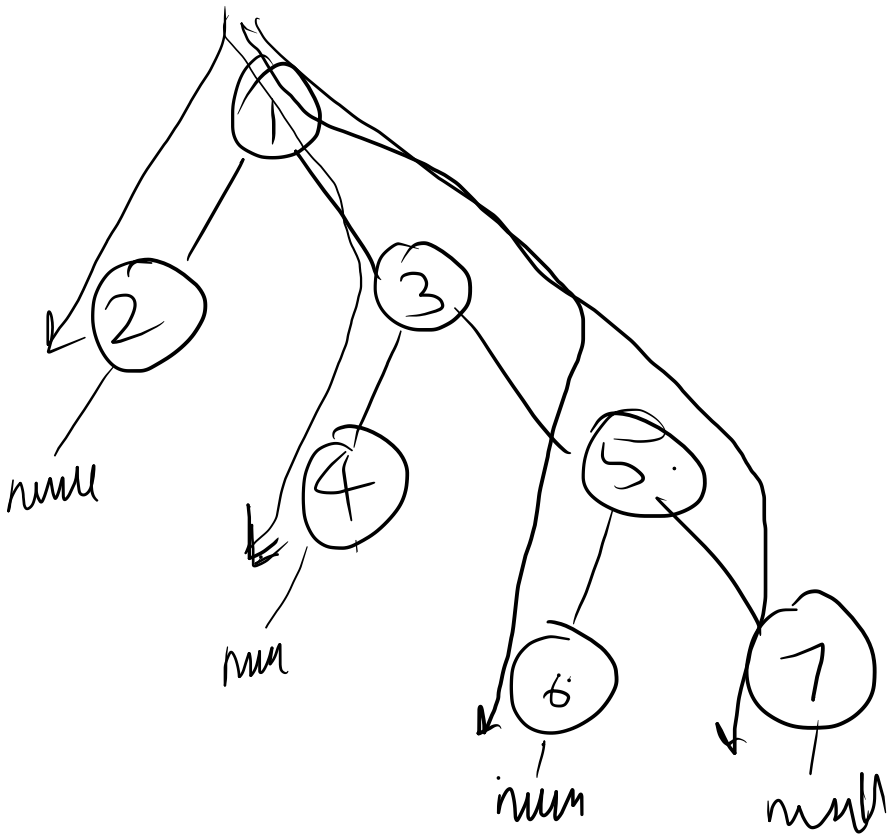
→ no. of levels

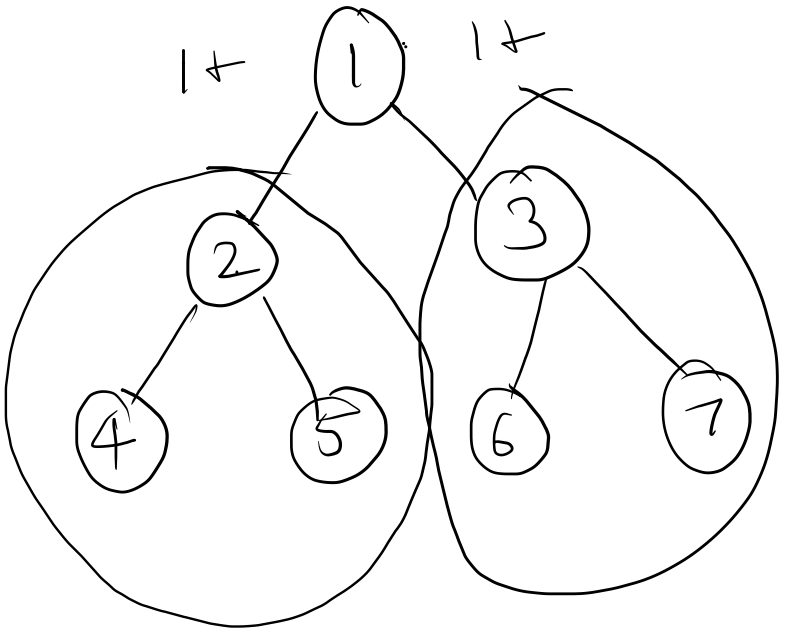


Height = 3



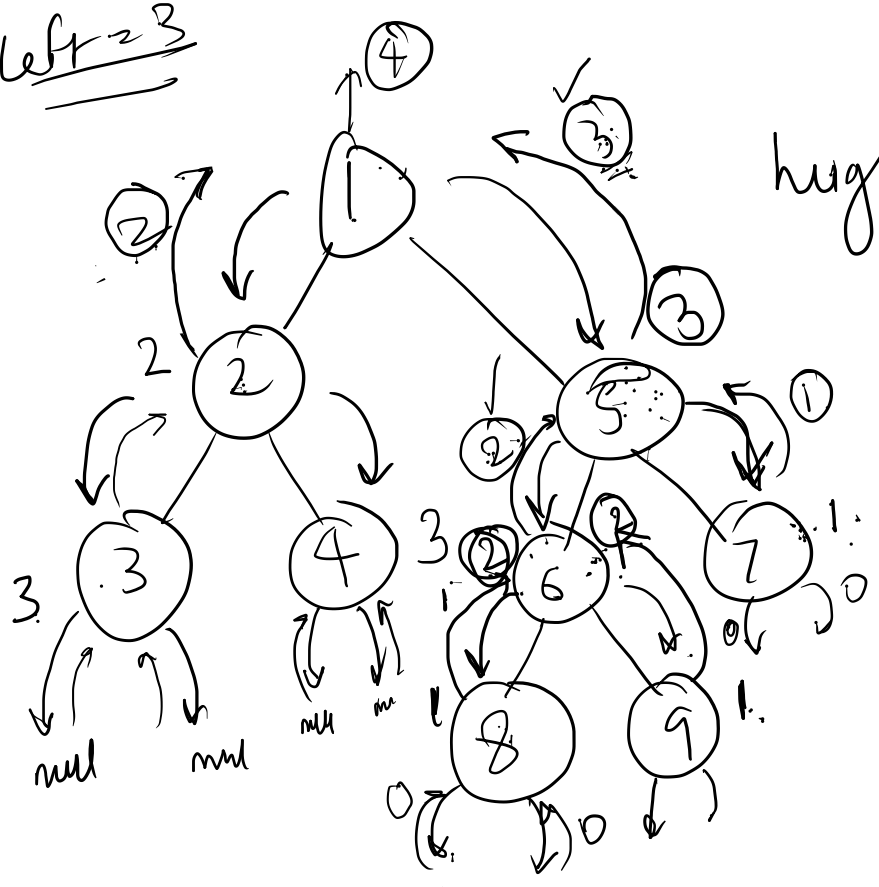
Binary Tree - Yes


1
2
3
4
5
6
7
null
null
null
null



root → 1 + max(height(left) , height(right))

left = 3



height = 1 + bottom tree height

= 1 + 1

= 2

= 1 + 1

= 2

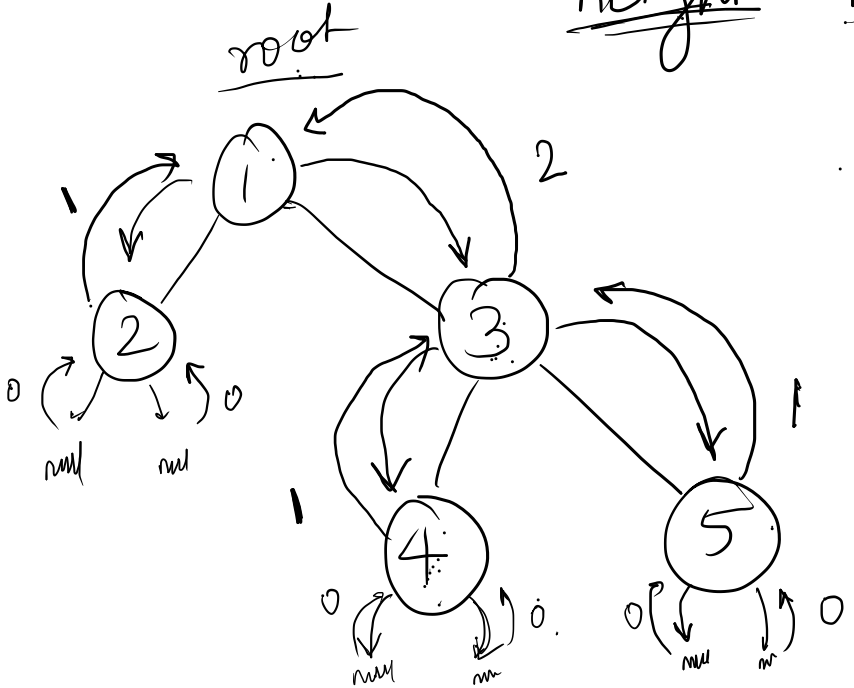
max(2, 1)

max(left, right)

1 + max (2, 3)

height = 3

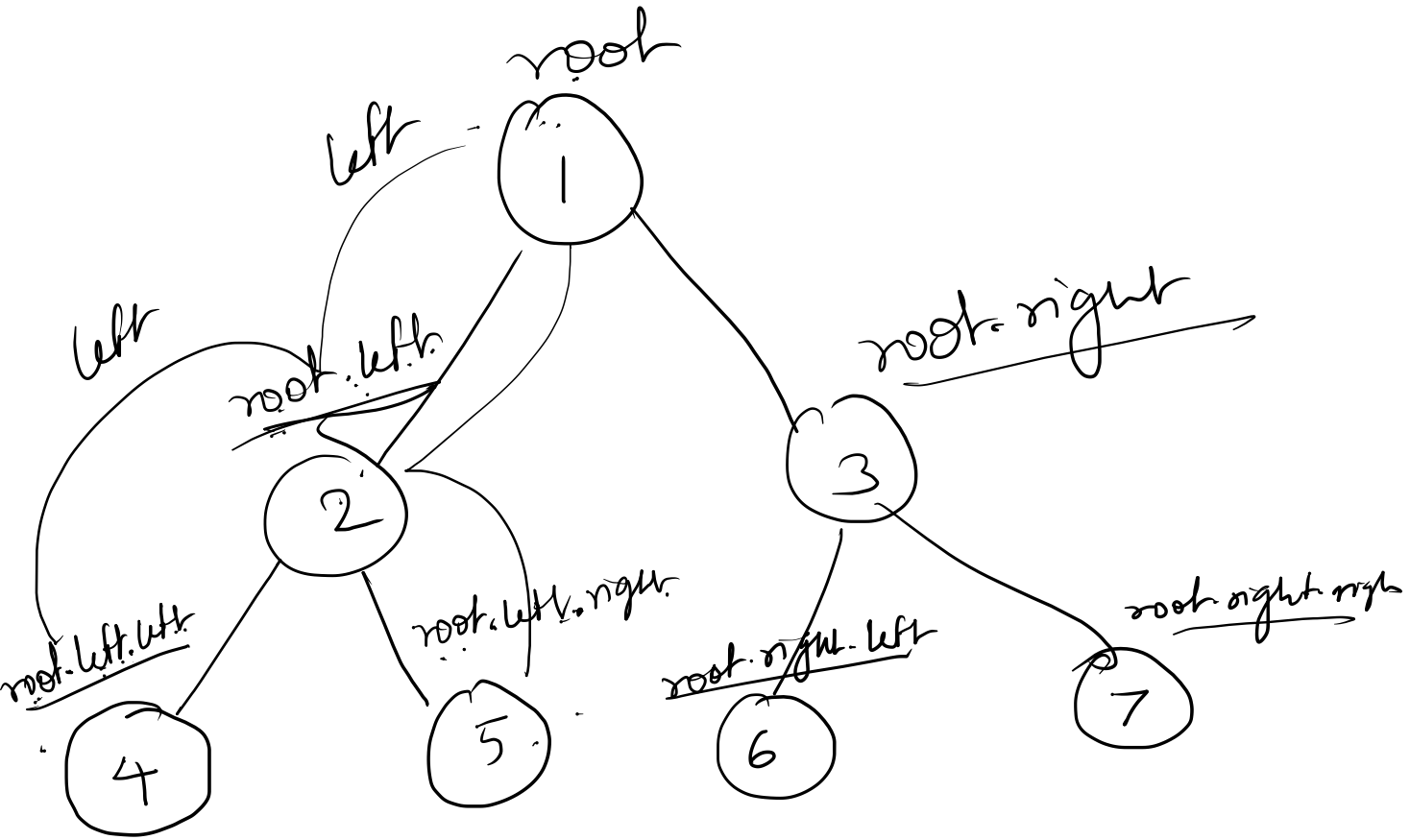
height = 1 + Max(height(left), height(right))

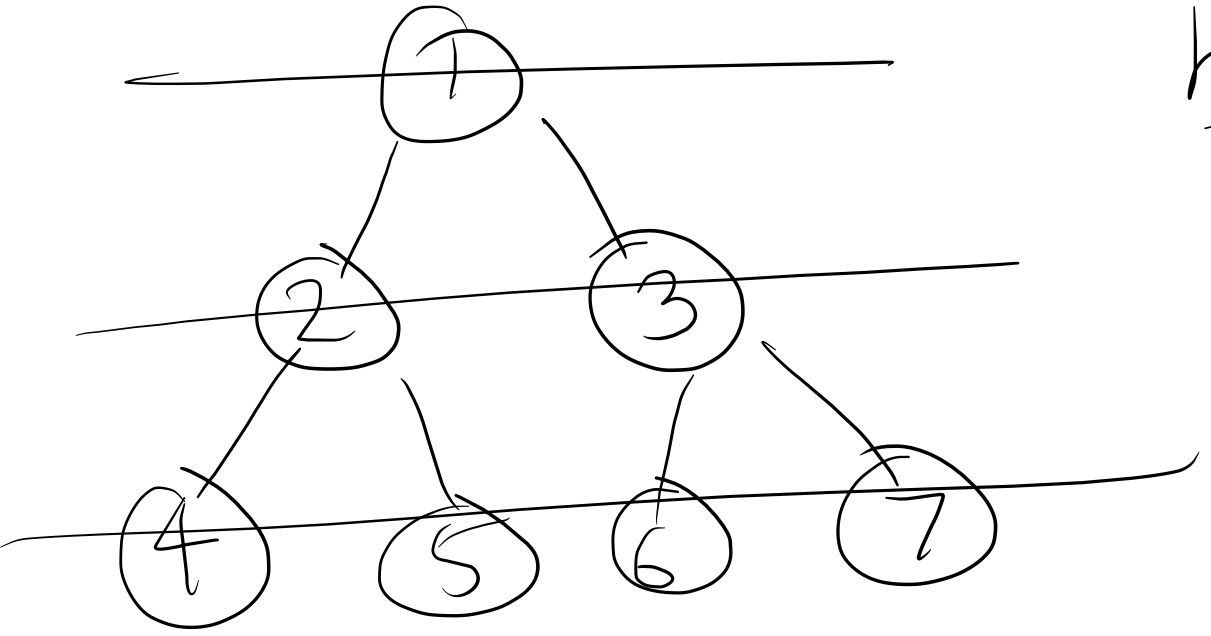


1 + max(0,0) = 1 + 0 = 1

1 + max(1,1) = 1 + 1 = 2

1 + max(1,2) = 1 + 2 = 3


root
left
1
left
root.left
root.right
2
3
root.left.left
root.left.right
root.right.left
root.right.right
4
5
6
7

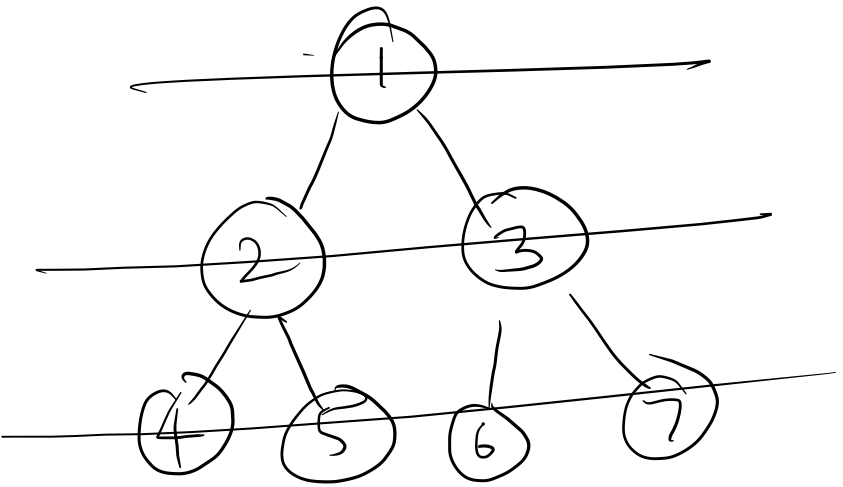


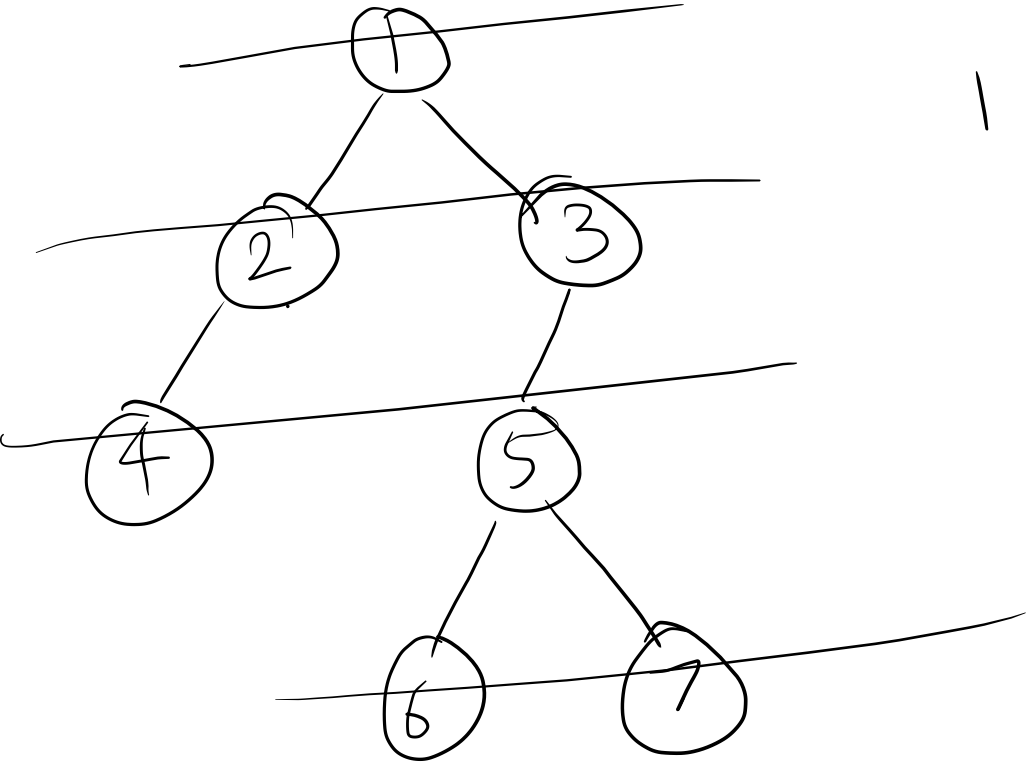
Height = 3

BFS

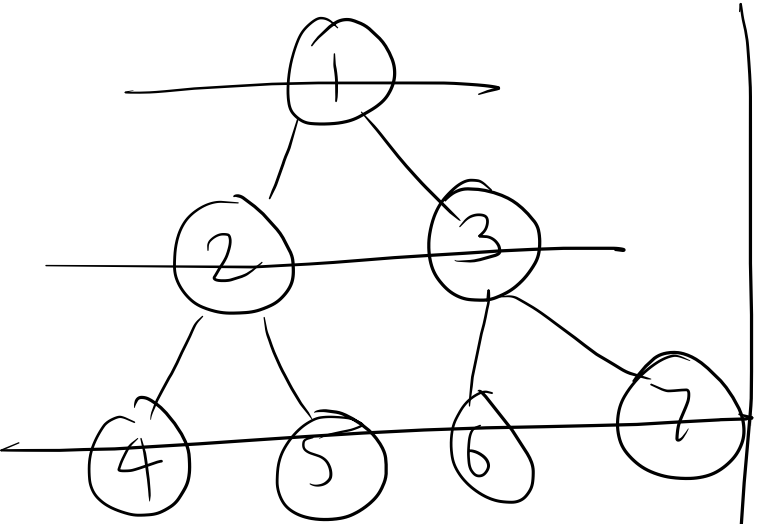
Breadth First Search

↳ Level Order Traversal





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

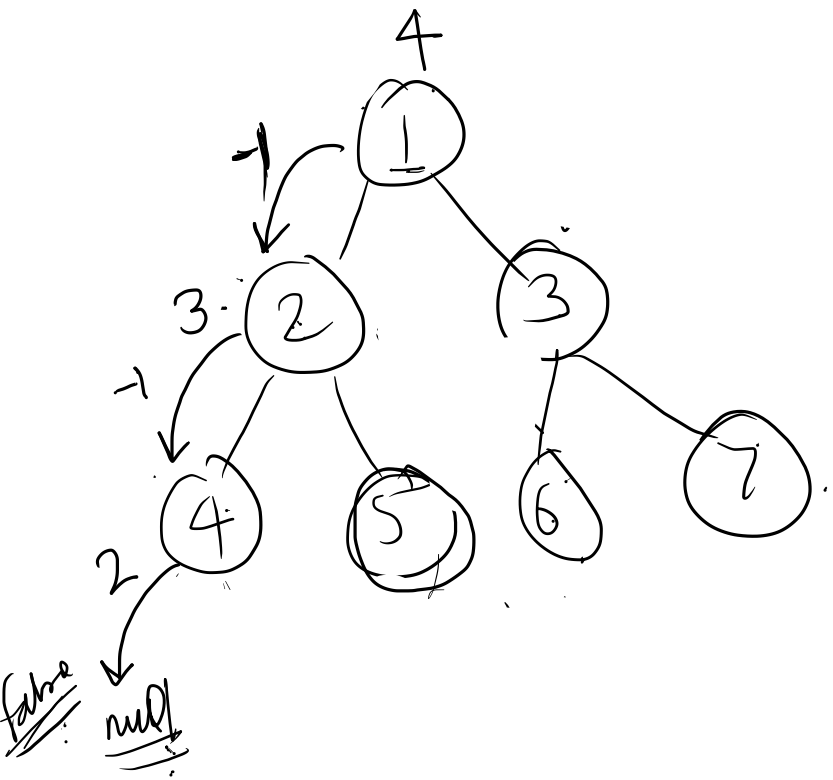


print (level) → True

print (level 2) → True

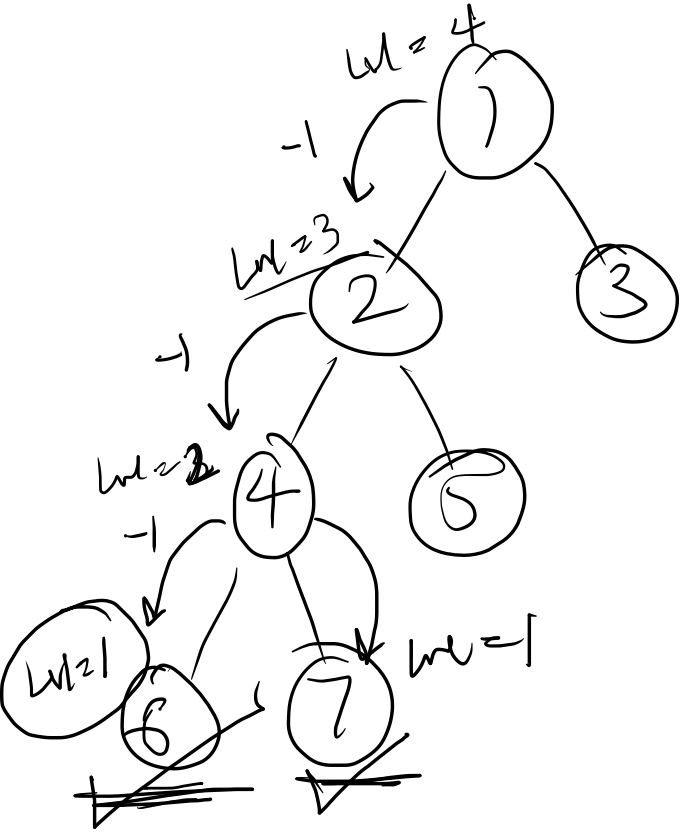
print (level 3) → True

print (level 4) → false

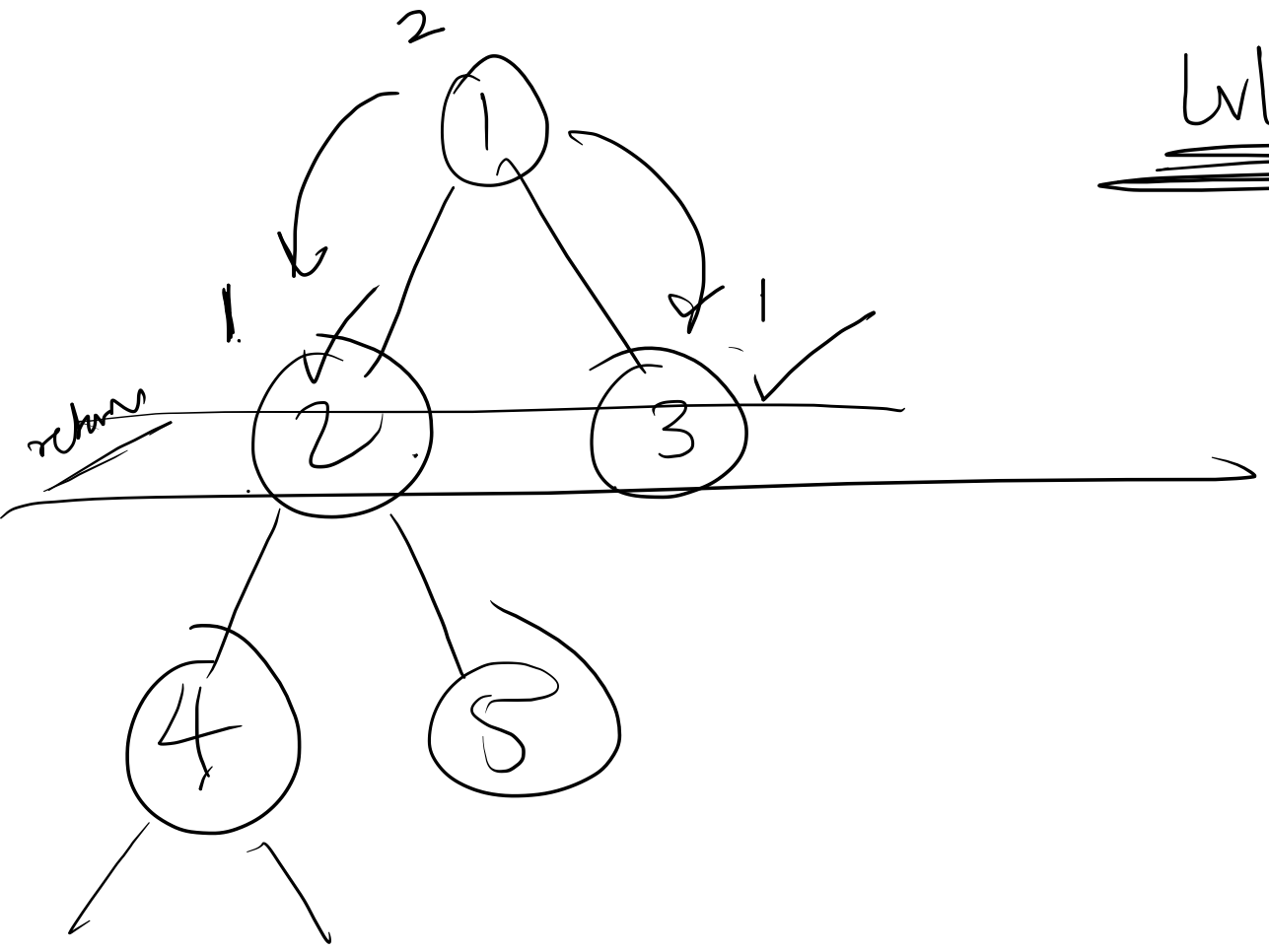


lvl = 4

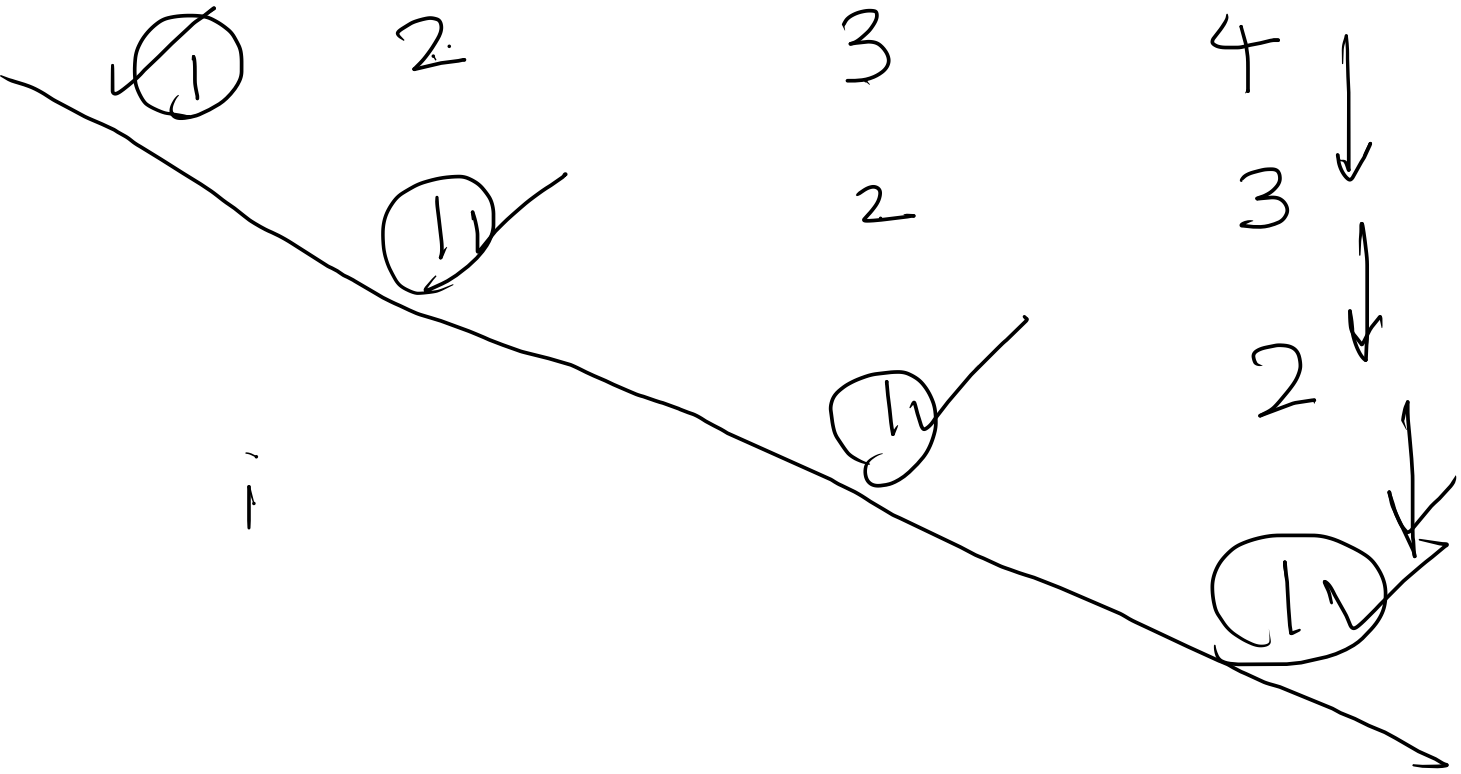
1, 2, 3, 4, 5, 6, 7



printLevel(4)



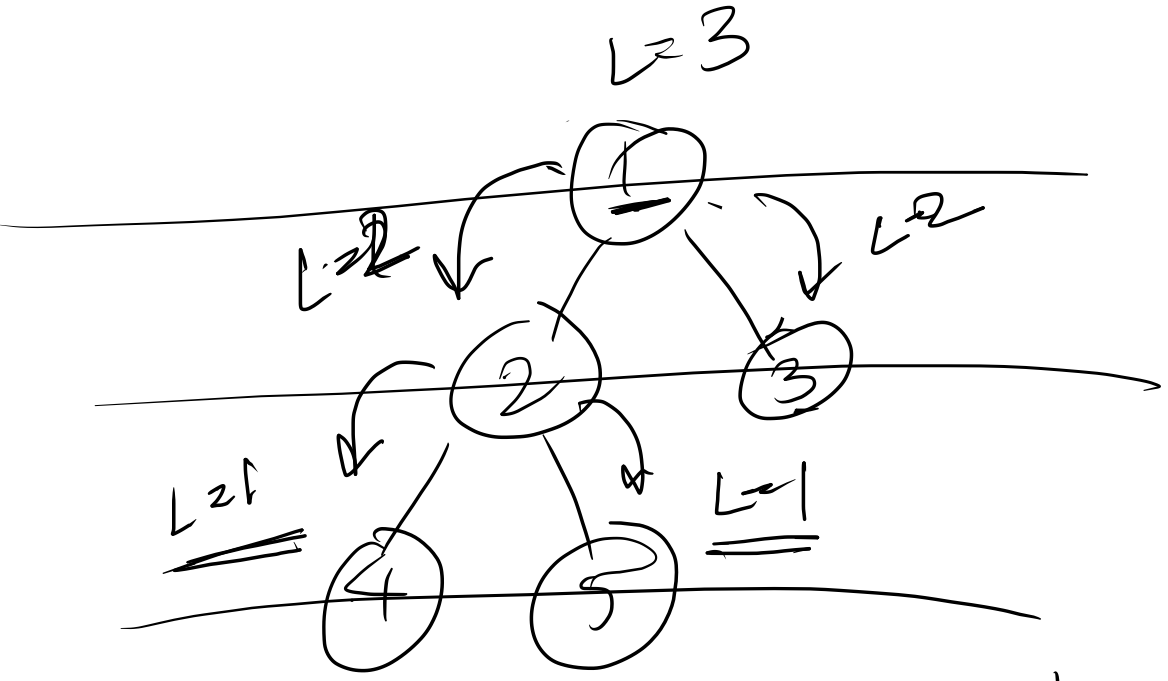
lvl = 2


1
2
3
4
3
2
2
1
1
1
i

n calls → n

T.C = O(n x n) = $O(n^2)$

Aux space = recursive stack O(n)



L = 3

1 2 3 4 5