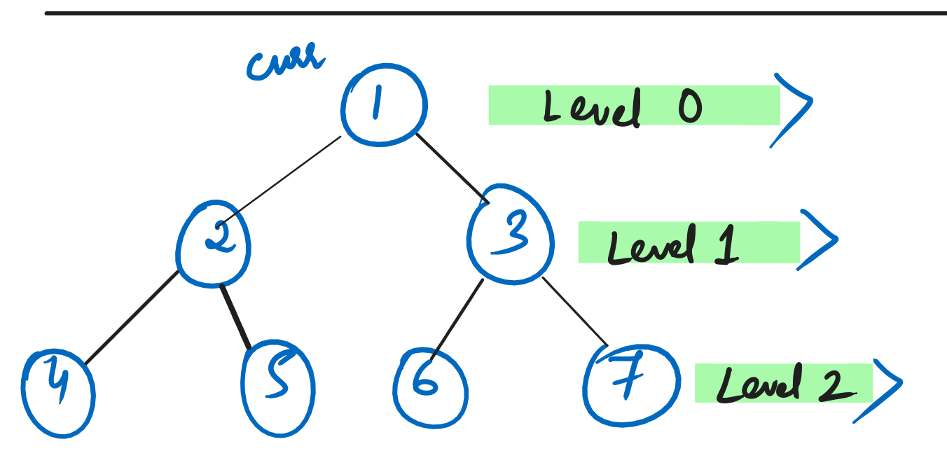
Level Order Traversal or Breadth First Search



In this traversal, we travel the tree via the root node taking a path from top to bottom & left to right, level by level.

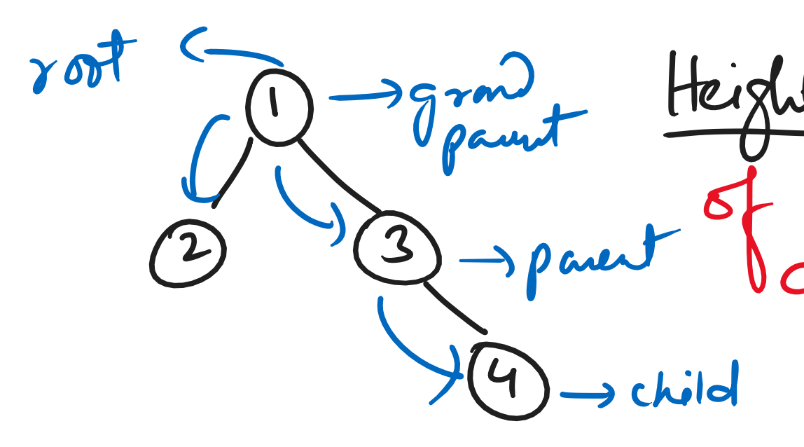
For this purpose, we use the queue data structure because of it's "FIFO" property.

Output :→ BFS :→ $\underbrace{1,}_{L0}$ $\underbrace{2, 3,}_{L1}$ $\underbrace{4, 5, 6, 7}_{L2}$

Binary Trees Important Interview Questions for Placements:

① Height of a Binary Tree
② Diameter of a Binary Tree
③ Identical Trees
④ Mirror of a Binary Tree
⑤ Symmetric Tree
⑥ Sum of Nodes with Even Grand Parents
⑦ Lowest Common Ancestor → LCA
⑧ Serialize, Deserialize Binary Tree
⑨ House Robber III Leetcode.
⑩ Left & Right Views of a Binary Tree.

{ LeetCode
GFG
Coding Ninjas }



Height of a Tree: The maximum number of nodes from the root to any of it's descendants.

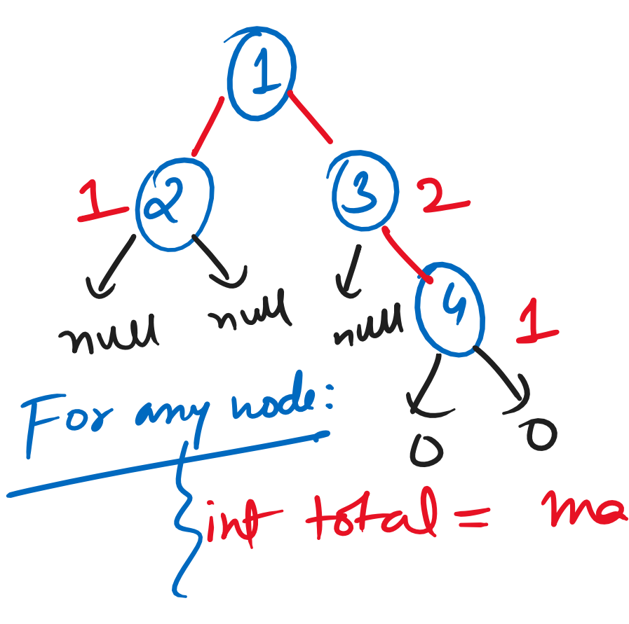
h = 3

(F2F)

Pseudo code: total height = max(left height, right height) + 1.

Root → 1
(2, 3, 4) descendants

*(Recursion Tree) → (4.5)

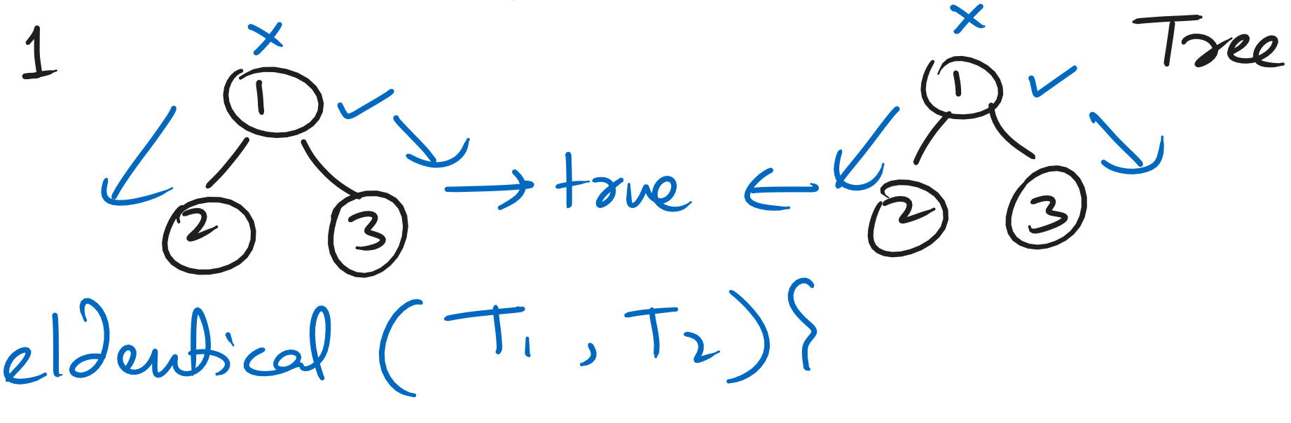


For any node:
{int total = max(lh, rh)+1}

findHeight(root)

findHeight ① → 2+1 = ③
→ findHeight(2) max(0,0)+1 = ①
  → fh(null) 0
  → fh(null) 0
→ 2
→ findHeight (3) → 1+1 = ②
  → fh (null) → 0
  → ①
  → fh(4) → 0+1
  1

Identical Trees : (Cognizant & TCS) Bulk Hiring/April '2025



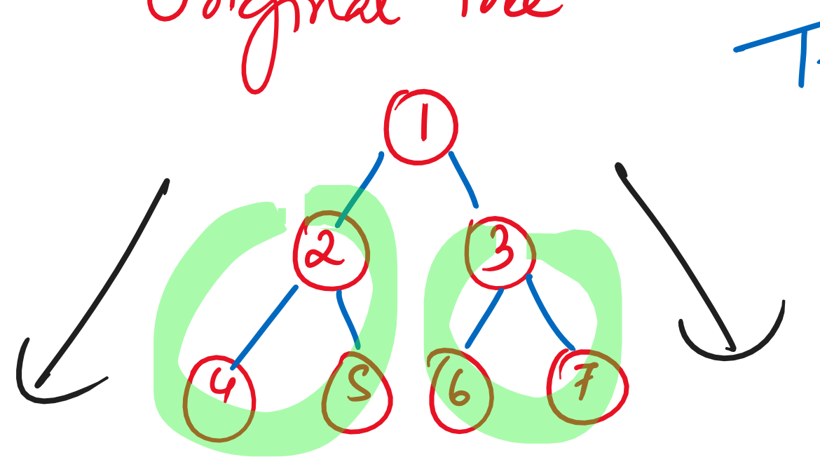
boolean areIdentical (T₁, T₂){

}

Mirror of a Binary Tree:→ *** Important

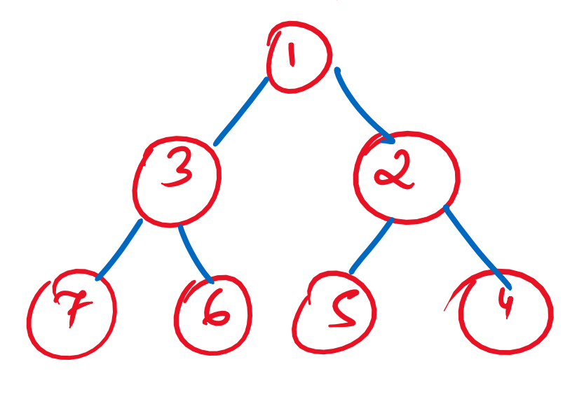
90% recursion

Original Tree



Mirror (root)
TreeNode temp = r.l
r.l = r.r
r.r = temp;

Mirror Tree

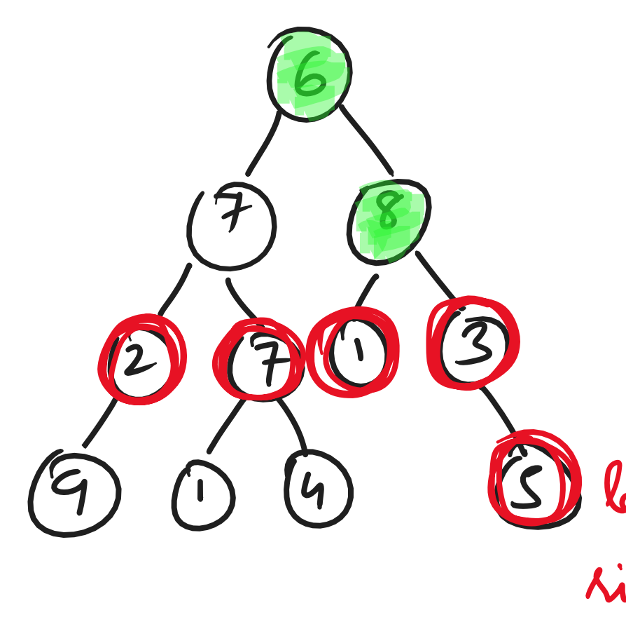


In-order: 4, 2, 5, 1, 6, 3, 7 ↑

7, 3, 6, 1, 5, 2, 4 ↑

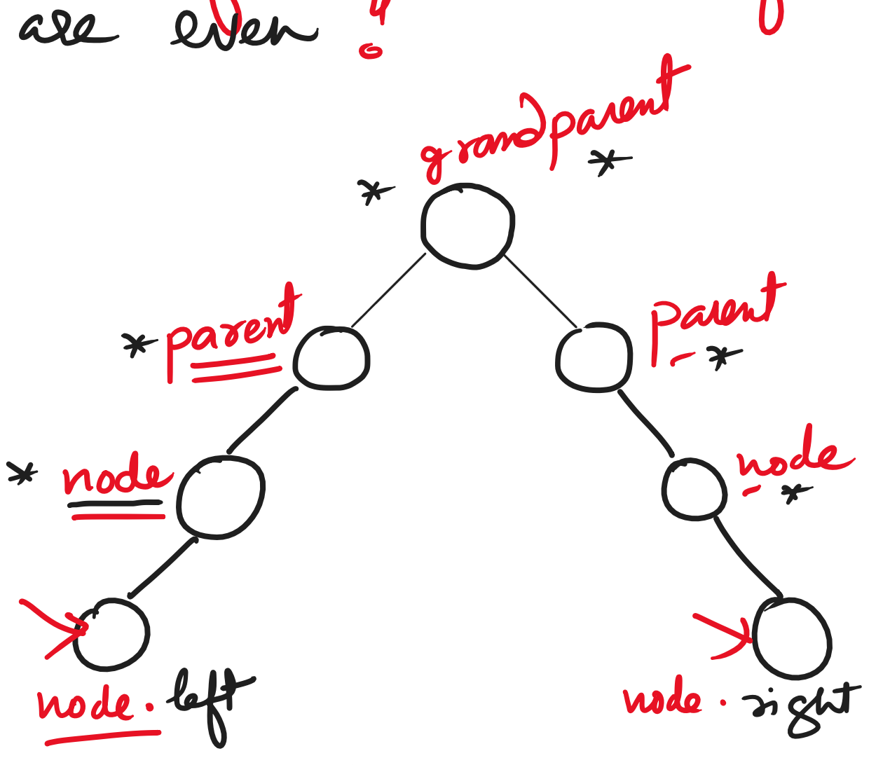
Cognizant | Capgemini | HP | Dell | LG Soft

* Given a binary tree, find the sum of those nodes of the tree whose "grand parents are even"!



Sum of nodes with EG =
2+7+1+3+5 = ⑱

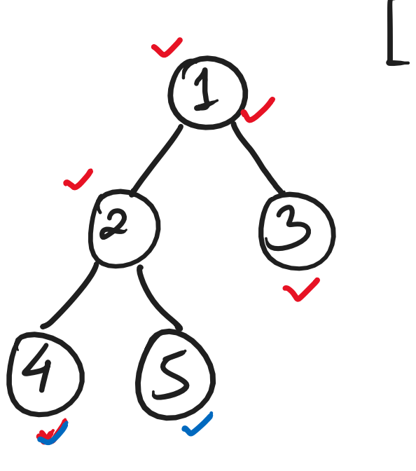
rgs(n, p, g, sum)
left: rgs(n.l, n, p, sum)
right: rgs(n.r, n, p, sum)
if(g.data % 2 == 0)



sum += gp.data;

Most frequently asked Binary Tree Problem in Interviews:

"Lowest Common Ancestor of Two Nodes In a Binary Tree" [LeetCode - 236]



lca(root, p, q)
4, 5

① if(root == null || root is p || root is q)
return root;

② if left != null && right != null
return root

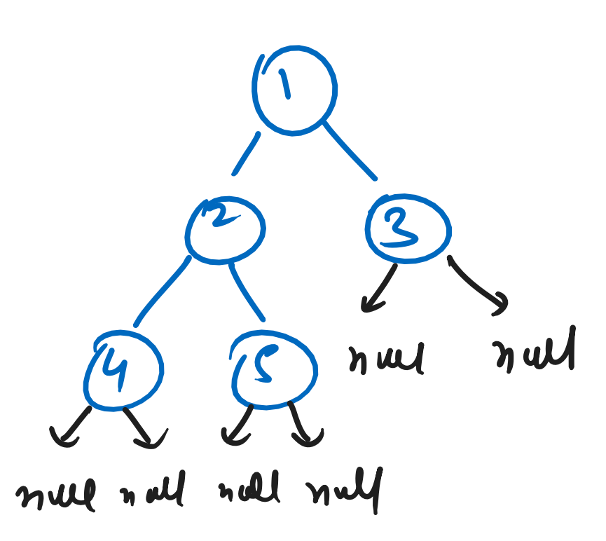
③ if left is null or right is null
→ return non null child

4, (2 & 1)
5, (2 & 1)

3 (1)
2 (1)
1 (1)

Tracing the lowestCommonAncestor lca(root, 4, 5) :→

Recursion Tree

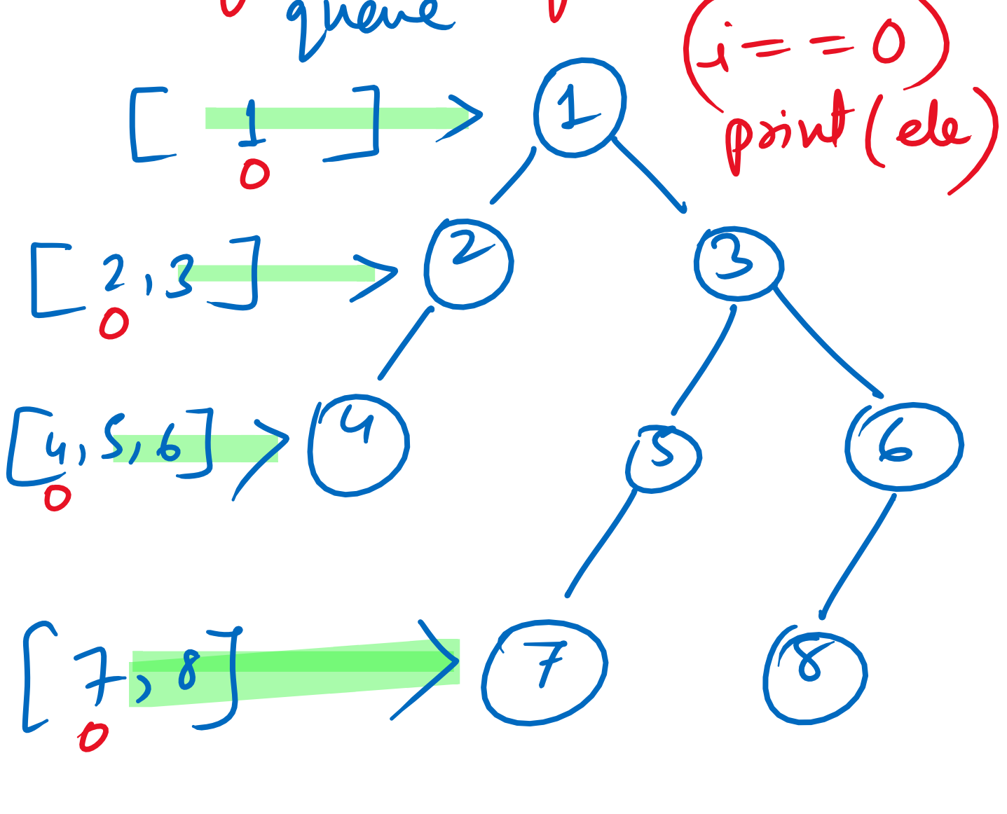


LCA(1, 4, 5)
left → LCA(2, 4, 5) ②
  left → LCA(4, 4, 5) → (matched) returns 4
  right → LCA(5, 4, 5) → (matched) returns 5
  left → 4 right → 5 LCA = ②
right → LCA(3, 4, 5) (null)
  → LCA(null, 4, 5) → null
  → LCA(null, 4, 5) → null
  → (null)
→ LCA = ②

Left View of a Tree

queue

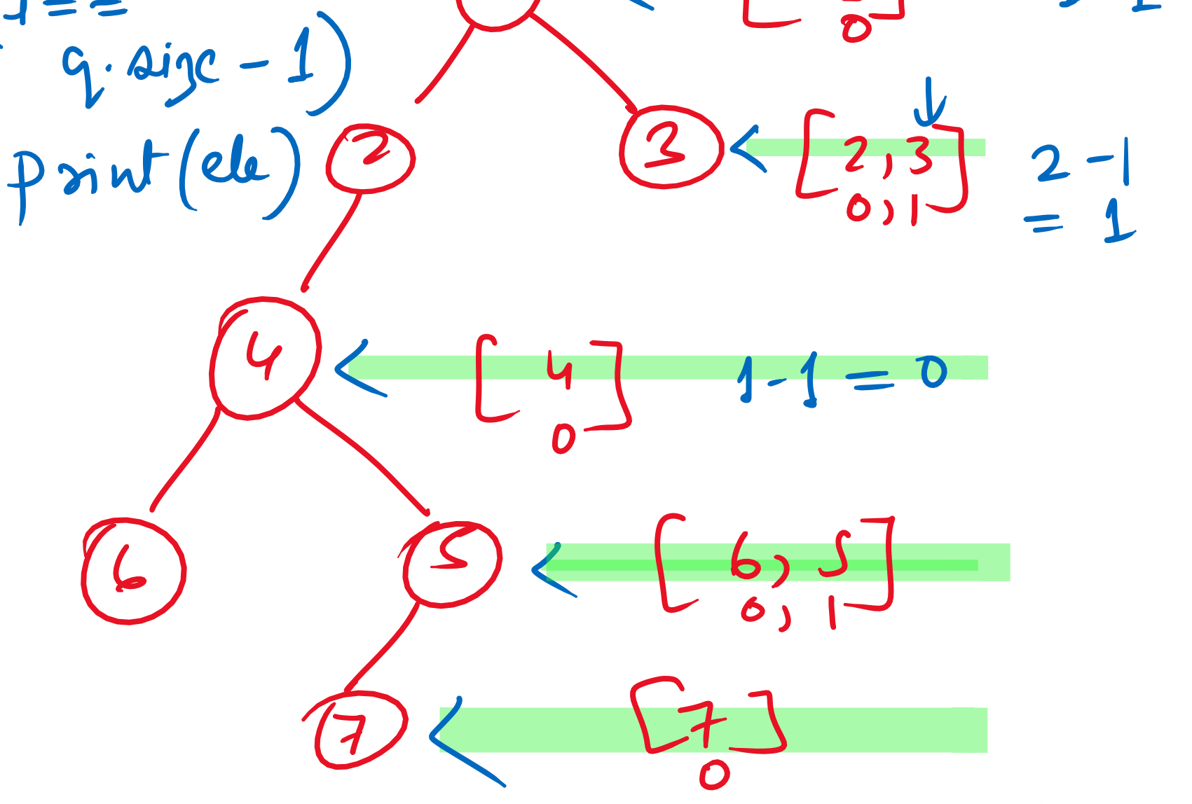
(i == 0)
print(ele)



O/p: 1, 2, 4, 7

Right View of a Tree

if (i == q.size - 1)
print(ele)



O/p: 1, 3, 4, 5, 7