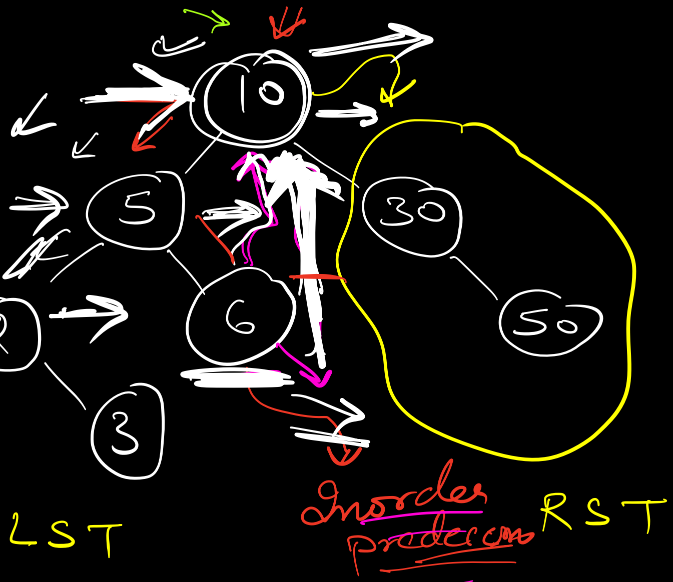
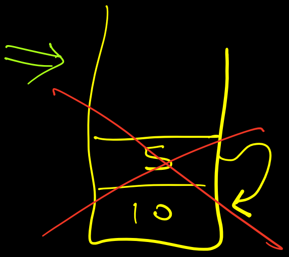
Inorder ⇒ LNR (BST)

Stack space ⇒ O(H)
||
O(N)

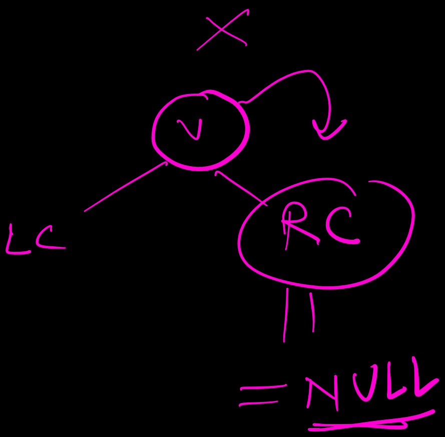
O(1) extra space





(LST) , 10 , RST

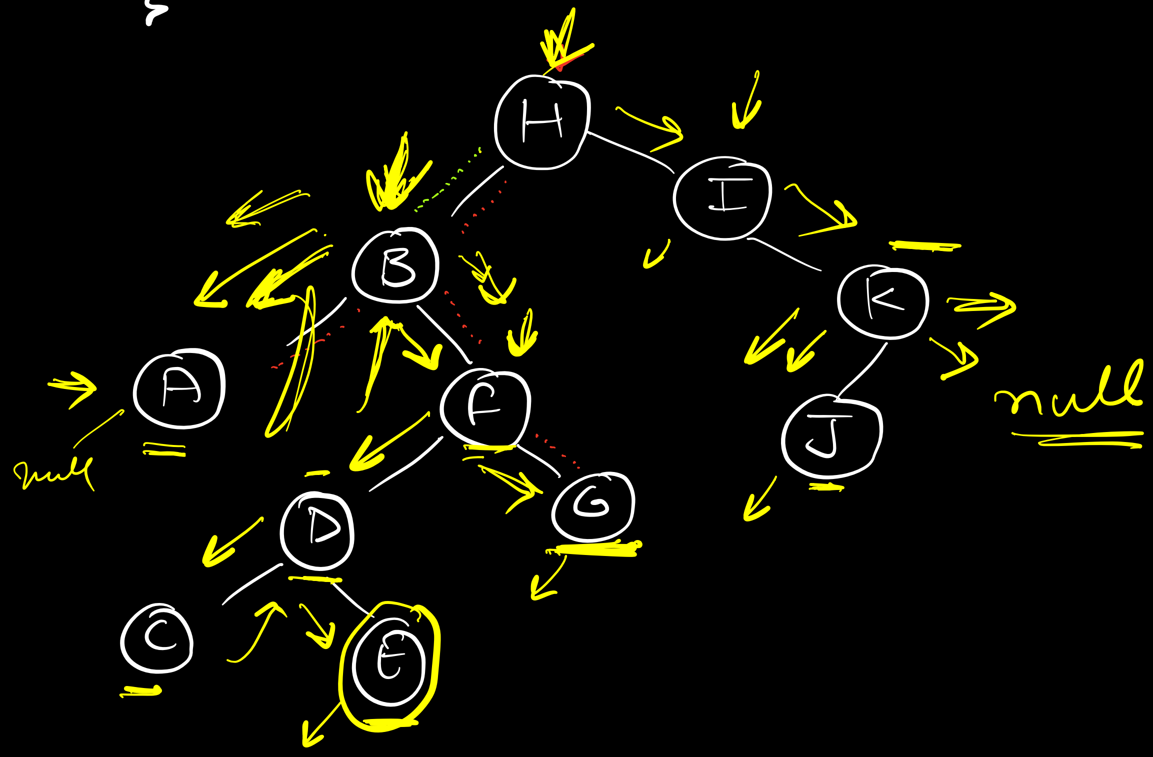
2, 3, 5, 6, 10



⇒ ∀ node, we try to find the inorder predecessor & update it's right to the node.

Code

```
  curr = root ;
→ while (curr != null) {
  → if (curr. left == null) {
          print curr;
          curr = curr. right;
     }
    → else {
         → pred = findPredecessor (curr);   ⇒ O(H)
      → ⇒ if (pred. right == null) {
                pred. right = curr;
                curr = curr. left;
            }
         → else {
                pred. right = null;
                print (curr. data);
curr = curr.right;
}
}
```

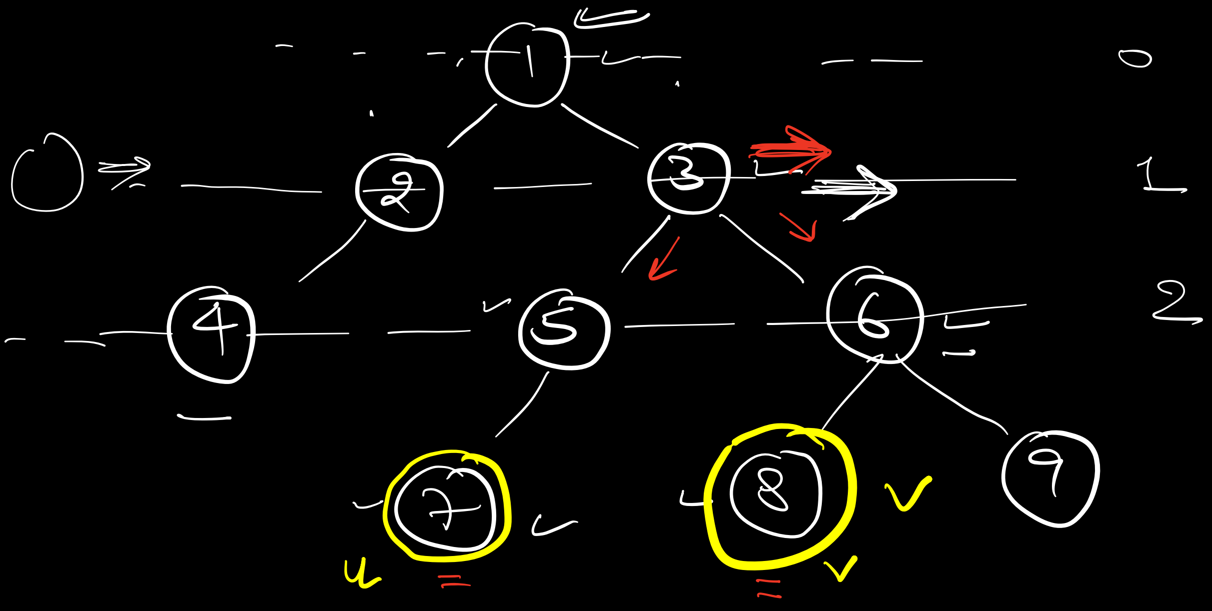


A, B, C, D, E, F, G, H, I, J, K

S.C. = $O(1)$

T.C. = $O(2N) = O(N)$

Morris's Inorder Traversal



Ancestors of 7 ⇒ 1, 3, 5, 7

6 ⇒ 1, 3, 6

8 ⇒ 1, 3, 6, 8

⇒ 1 & 3 are common ancestors

of 7, 6, & 8

⇒ Lowest common ancestors

⇓

(largest level) = 3

LCA(4, 6) = 1

LCA (3,8)
u, v

3 ⇒ 1, 3

8 ⇒ 1, 3, 6, 8

u = LCA

Q How to find LCA??

⇒ ∀ node, find u & v in the LST & RST

if we able to find, node is a common ancestor

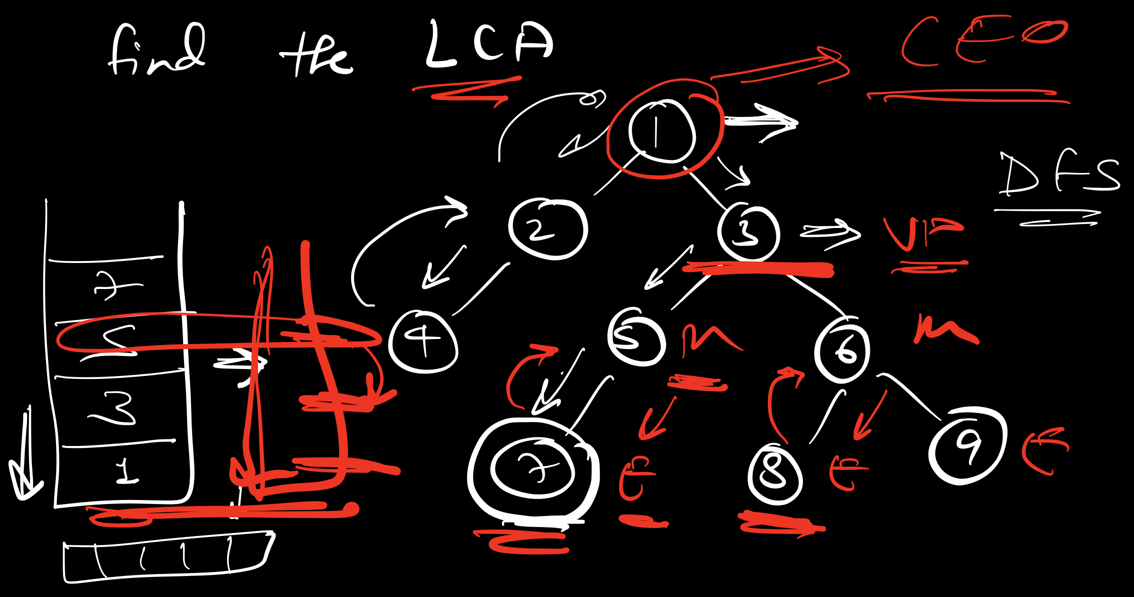
BF ⇒ find all CA
find the lowest one.

$$T.C. = O(\overset{\forall \text{ nodes}}{N} \times \overset{\text{traversal on subtree}}{N})$$

$$= O(N^2)$$

Q Given a binary tree & 2 nodes u & v.

find the LCA



Sol^n

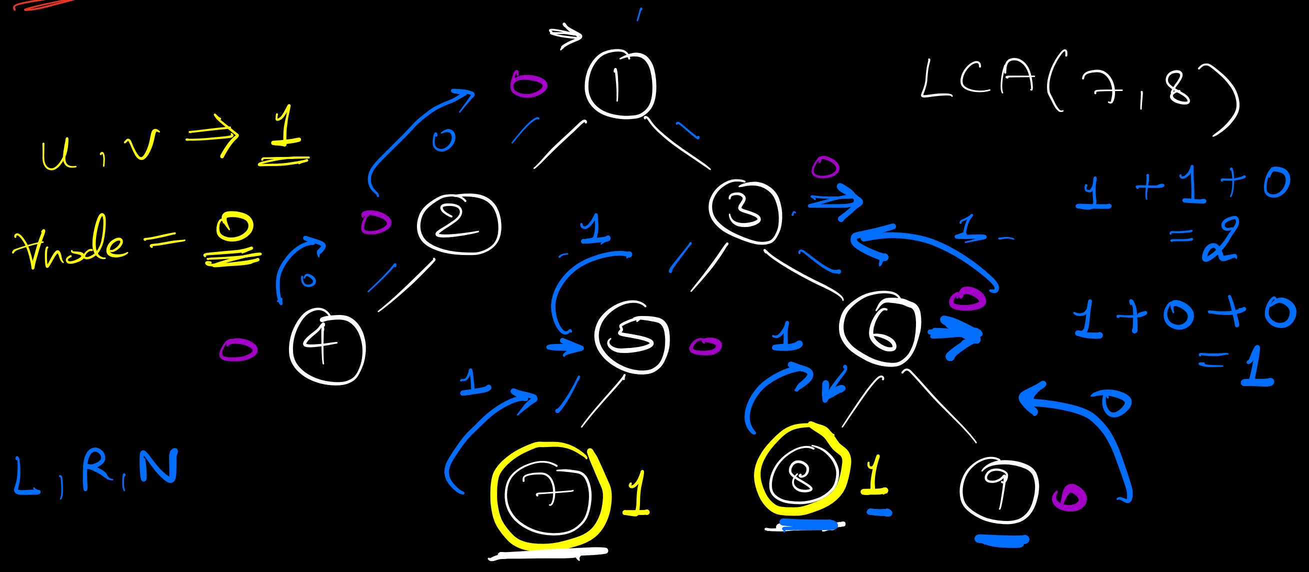
① Find ancestors of both & compare.

Use Stack to find out ancestors

T.C = O(N)

S.C = O(H) = O(N)

② Reduce space Complexity ??



1) 1 in $\overrightarrow{LST}$, 1 in $\overrightarrow{RST}$

2) 1 = node, 2nd in either LST or RST

⇒ Smallest subtree which contains both u & v will be rooted at the LCA.

⇒ Assign 1 to u & v
0 to every other node

⇒ Use post order to calculate the sum of every subtree.

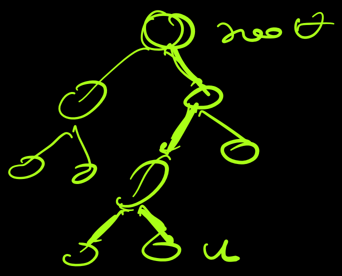
Root of

⇒ Smallest subtree for which the sum becomes 2 is the LCA

T.C. = O(N)

S.C. = O(1)

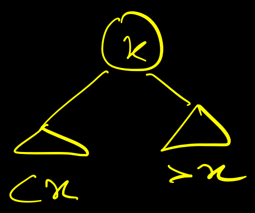
Adobe ⇒ Using bits   H.W.

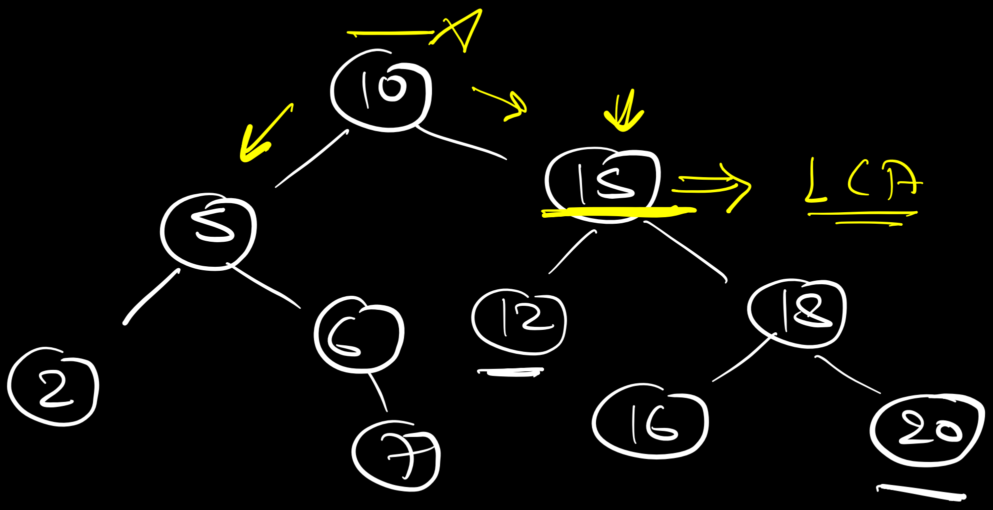


Try to represent in terms of string

& try to find mismatch

Q LCA in BST





$LCA[\overset{u}{12}, \overset{v}{20}]$

LST (u < root.value

RST (v ≥ root. value

node. value == u || node. value == v.

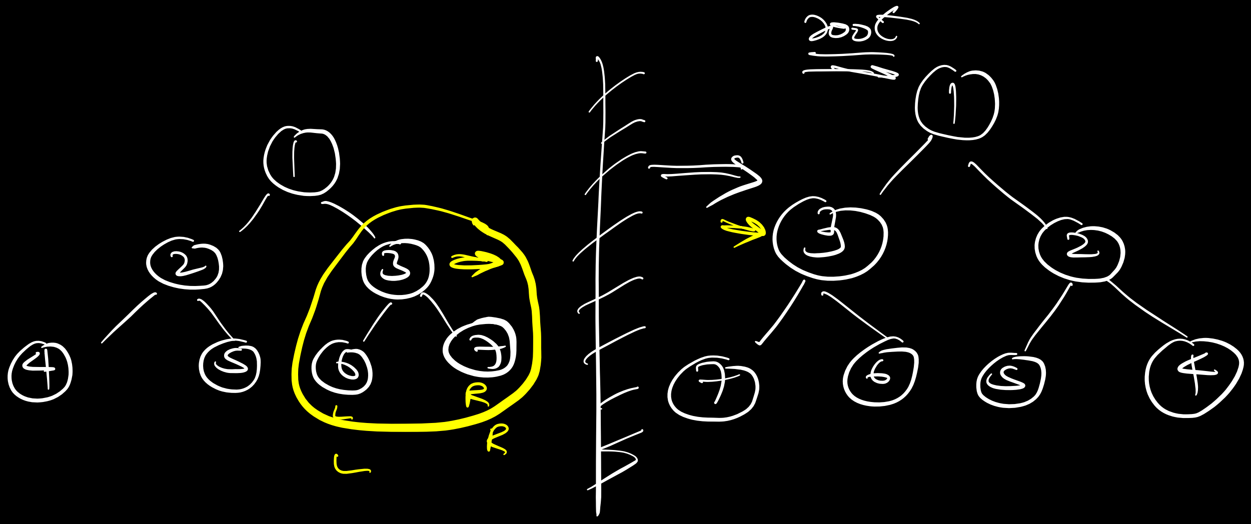
↳ node is the LCA

T. C. = O(H)

log N

N

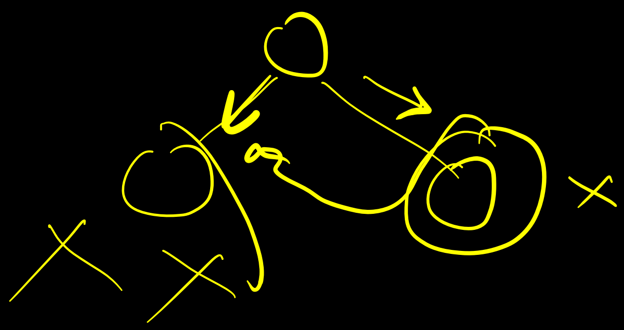
Q Given a binary tree
Invert the binary tree.



No extra space allowed.

⇒ Recursively swap the left & the right.

```
node invert (root) {
      if (root == null) { return null; }
temp = root.left
      root.left = invert (root.right);
      root.right = invert (~~root.left~~ temp);
      return root;
}
```



T.C. = O(N)

Unorderd ⇒ HashMap ⇒ O(1)

Hashing

Tree Map ⇒ Balanced BST ⇒

(log(N)) ✓

log(N) ⇒ ✓

✓

Unorder

Sorted list

Best ✓

Worst

Average ✓

BST → $O(H)$

N (Worst)

log N