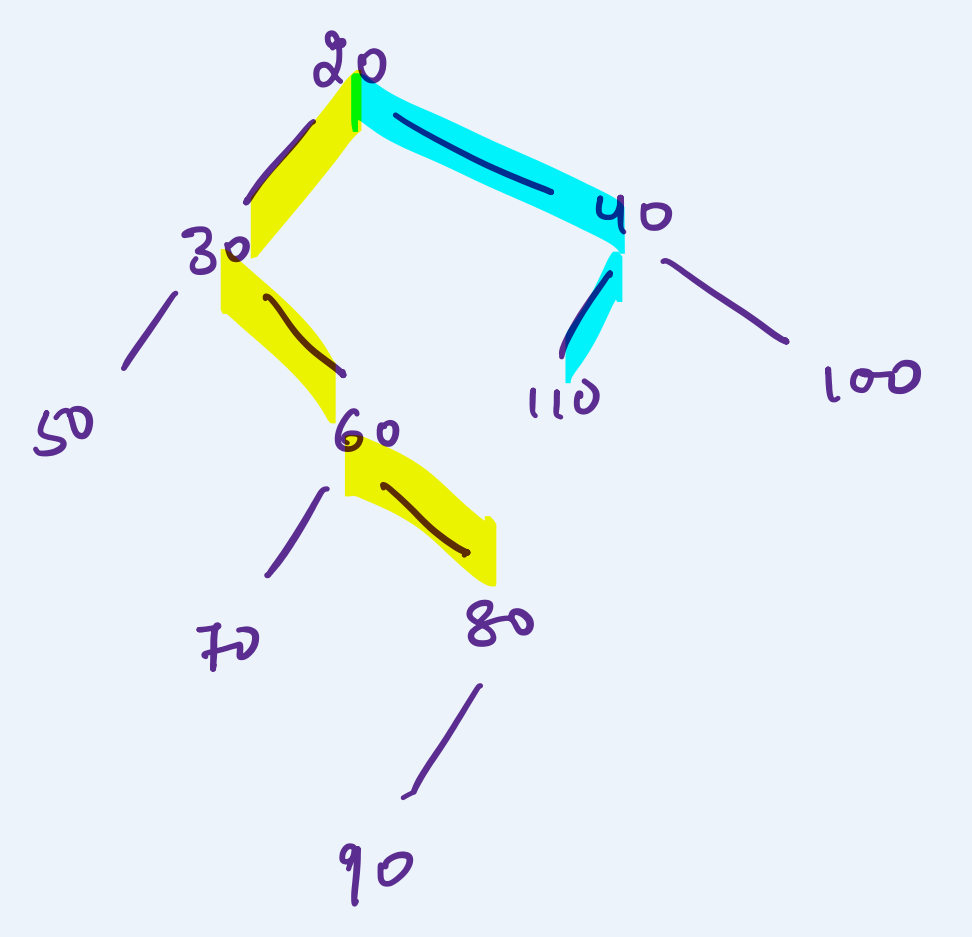
Wednesday, 27 October 2021 8:37 PM

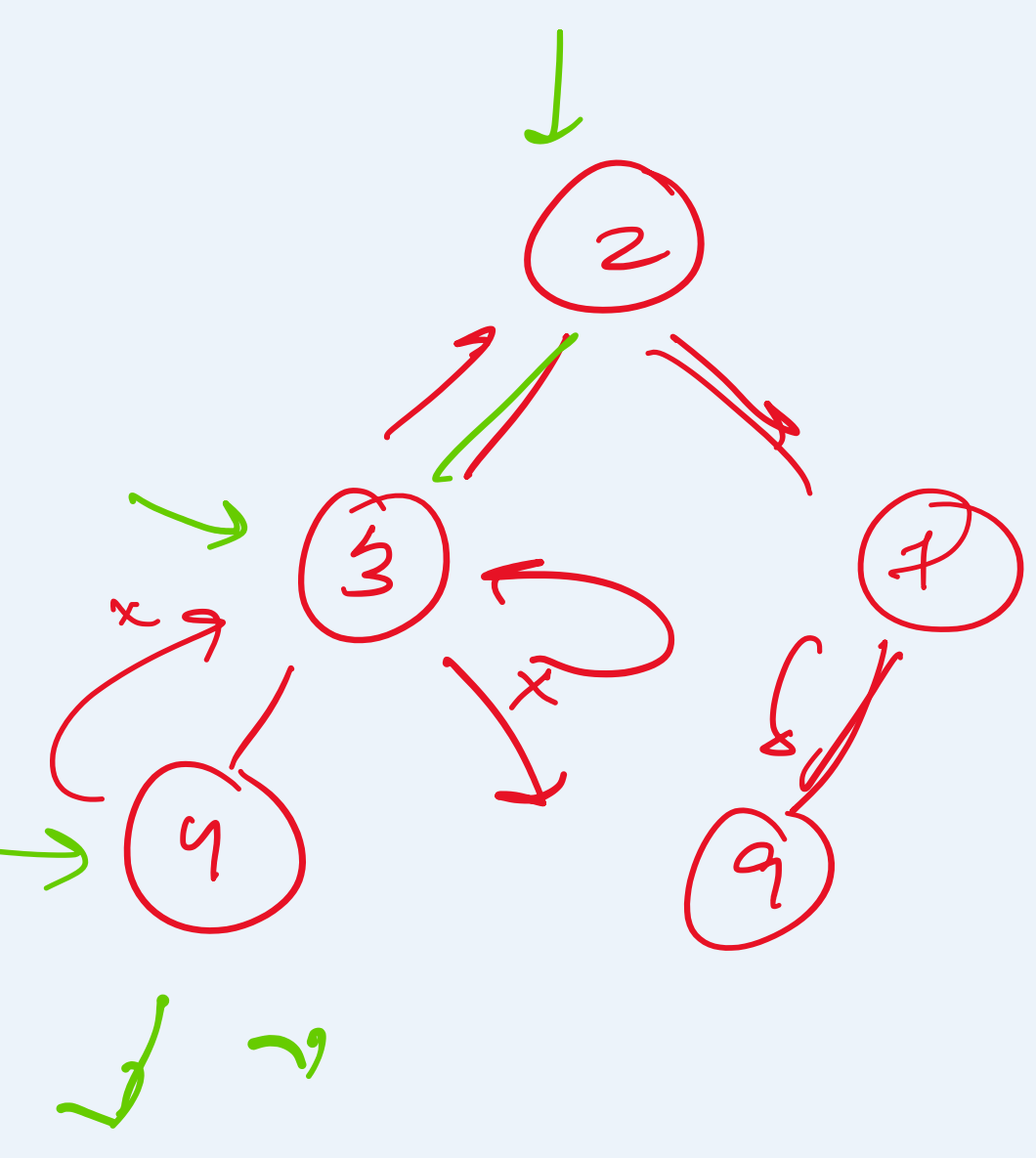
Q Given a BT, find the path from root to a given node



findPath (80)
↓
20, 30, 60, 80

findPath (110)
↓
20, 40, 110

```
hasPath ( root, Arraylist arr, x) {
    if (root == null)
        return false;

    arr. add (root. data);
    if (root. data == x) return true;
    if (hasPath (root. left, arr, x) ||
            hasPath (root. right, arr, x))
        return true;

    arr. remove (arr. size() - 1);
    return false;
}
```

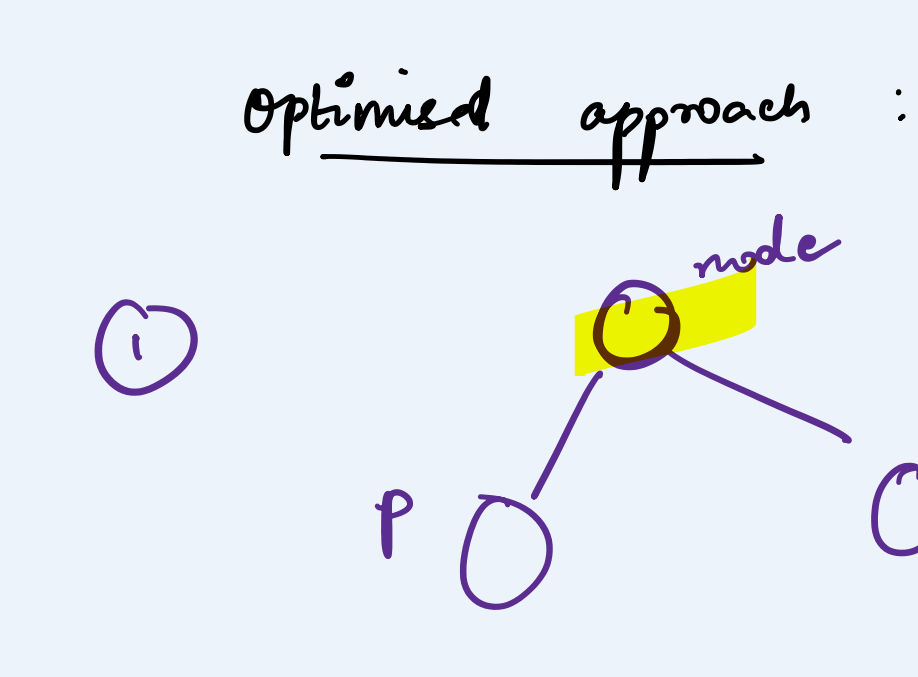


hasPath (9);

[2, 7, 9]

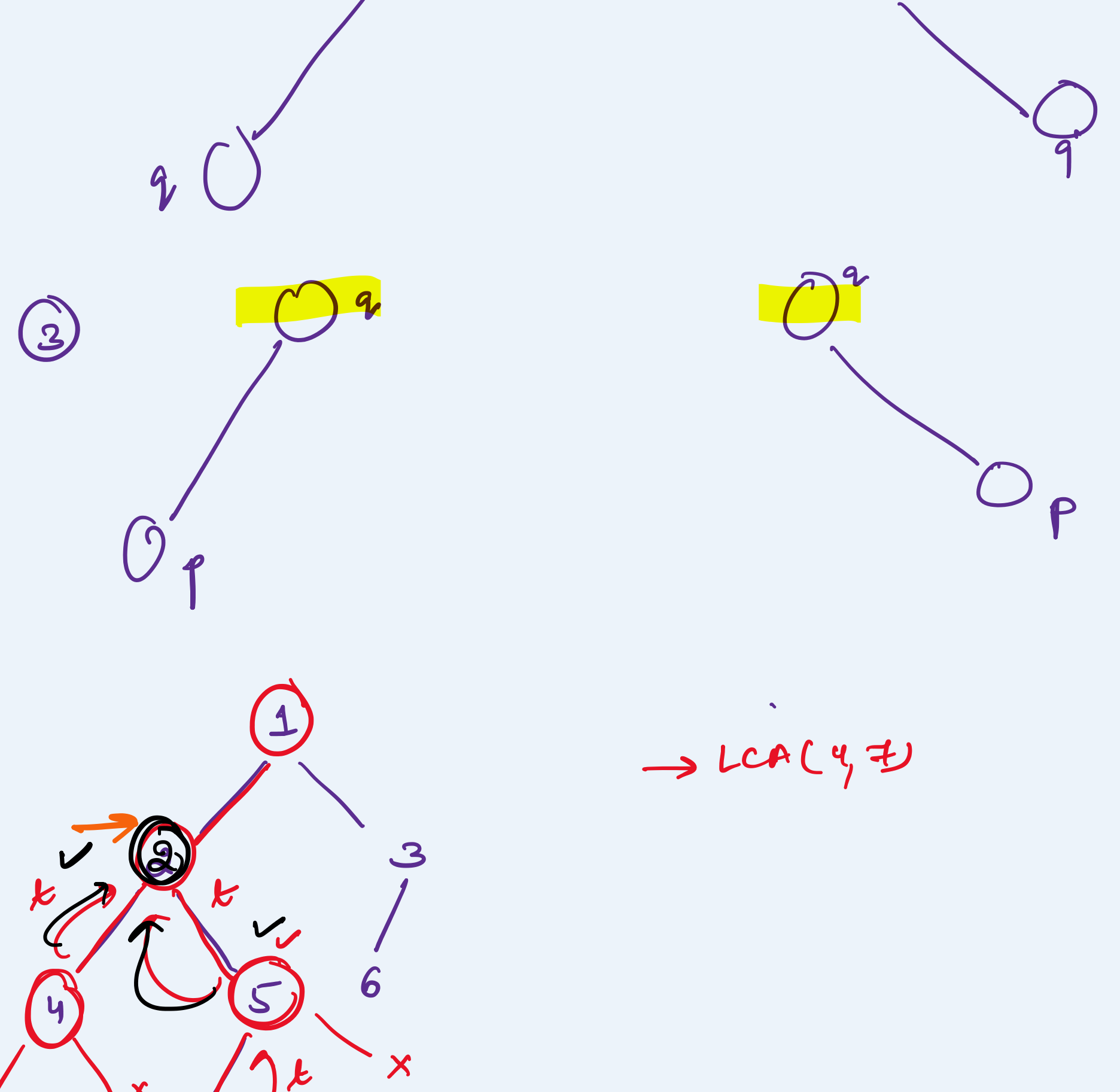
Q Given a BT, find the lowest common ancestor (LCA) of given two nodes.

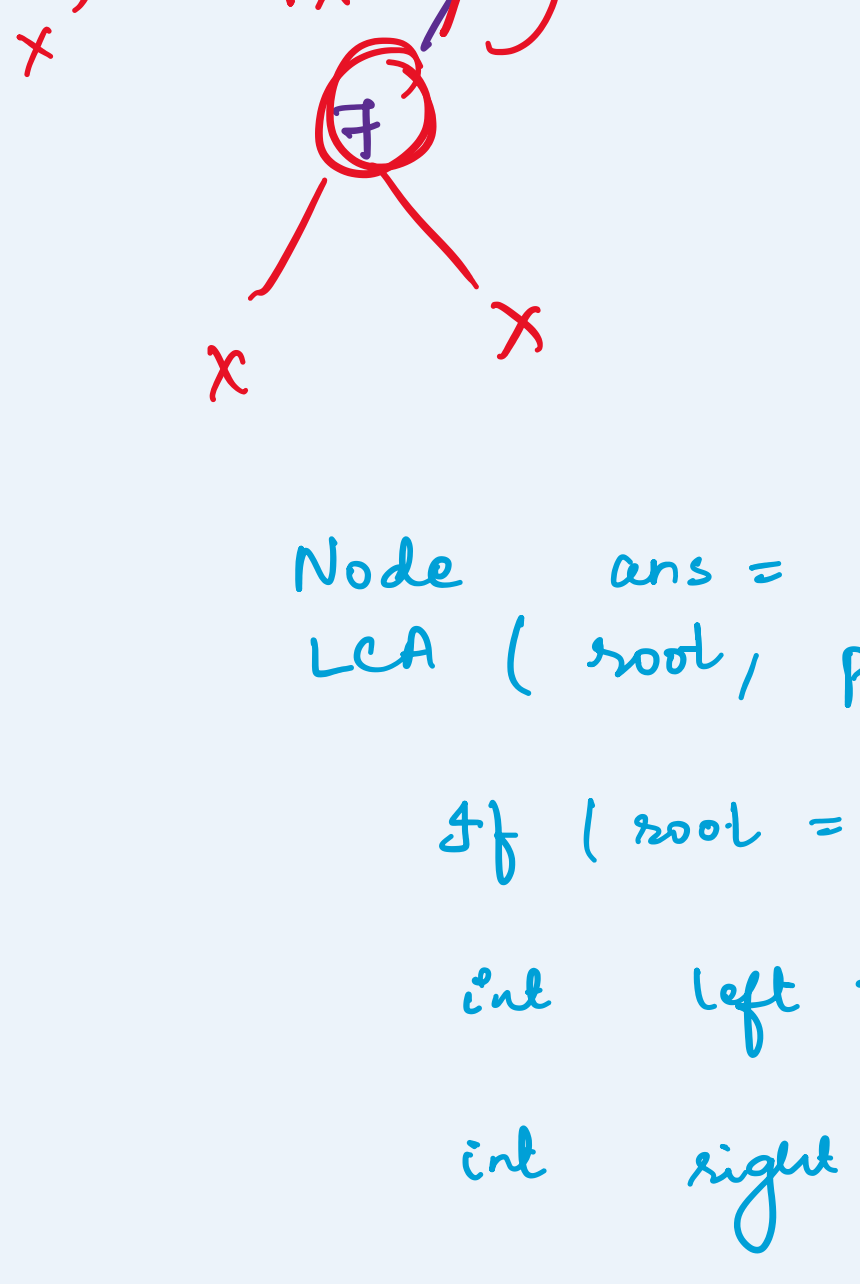
LCA : defined as the lowest node in a tree that has both p & q as descendents.



LCA (5, 7) → 2
LCA (2, 8) → 3
LCA (5, 8) → 5

Optimised approach :





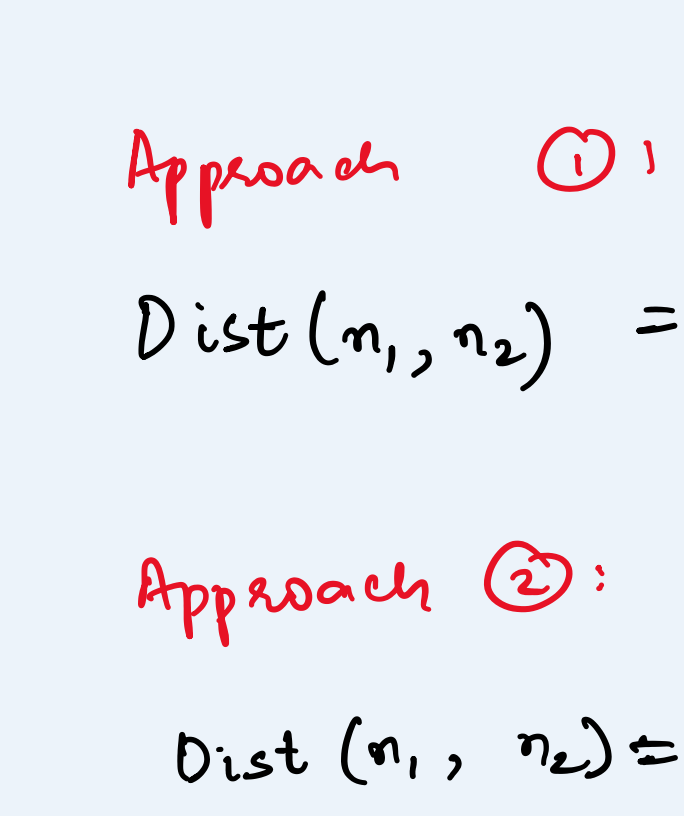
→ LCA (4, 7)

```
Node ans = null
LCA ( root, p, q) {

    if (root == null) return false;

    int left = LCA (root. left, p, q) ? 1 : 0
    int right = LCA (root. right, p, q) ? 1:0
    int mid = (root . data == p ||
                root. data == q) ? 1 : 0;

    if ( mid + left + right >= 2)
            ans = root ;

    return (mid + left + right > 0);

}
```

TC → O(n)

SC → O(n)

Q Given a BST, and two nodes, find the distance b/w 2 nodes.



find (6, 8) = 5
find (5, 4) = 3
find (4, 8) = 2

Approach ① : 2 + 3 − 2 * (0) = 5

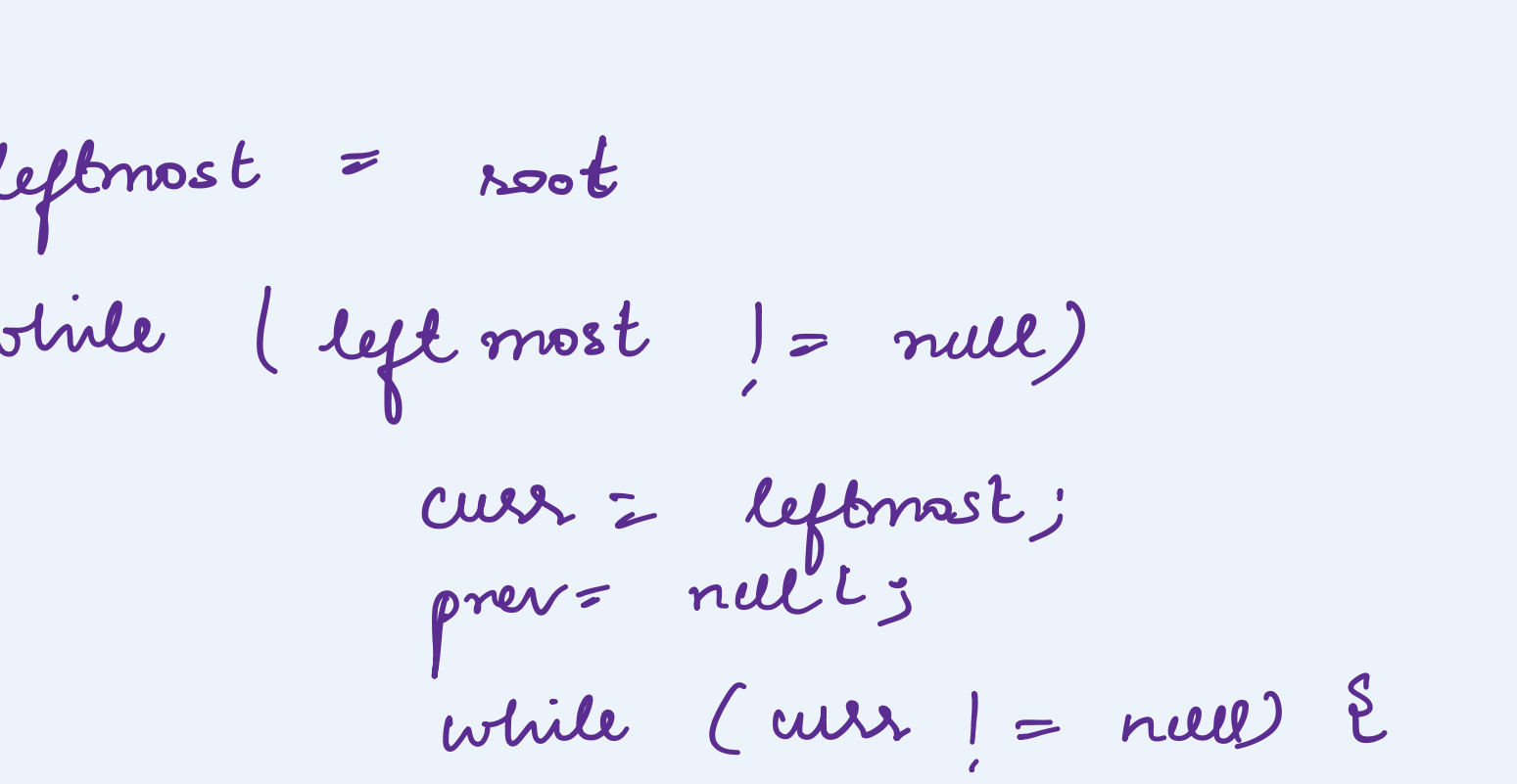
Dist(n₁, n₂) = Dist(root, n₁) + Dist(root, n₂) − 2 * Dist (root, LCA);

Approach ②:

Dist (n_1, n_2) = Dist (n_1, LCA) + Dist (n_2, LCA)

2 + 3 → 5

Q Populating right next pointers.




```
leftmost = root
while ( left most != null)

        curr = leftmost;
        prev= null;
        while (curr != null) {

            → process left child
            → process right child
            → move leftmost to next level
            curr = curr. next ;

        }
```