Thursday, 22 April 2021 9:24 PM

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

```
path (root, data, p) {
    if (root == null)
        return false;

    ar = add (root, data);

    if (root.data == p)
        return true;

    if (path(root.left, data, p) ||
        path(root.right, data, p))
        return true;

    ans = remove (ar.size -1);
    return false;
}
```

② Given a BT, find LCA

The LCA of two nodes u and v in a rooted tree is the node that is the ancestor of both u and v and has the greatest depth.

ⓑ Case:

Case 1:

Case 2:

```
LCA (root, p, q) {
    if (root == null)
        return false;

    if (root == p || root == q)
        [ int left = LCA(root.left, p, q) ] & [ int r = LCA(root.right, p, q) ] & [ int c = root.data ];
        → return c;
    if (root == p && root == q)
        if (root.left && root.right)
            → left;
    else
        if (root == p || root == q) && (root.left || root.right)
            return (root.left + root.right) > 0;
```

LCA of BST

Q. Given a BT, and two nodes, add the distance b/w nodes.

① Dist (7, 12) = Dist (root, 7) + Dist (root, 12)

② Find LCA of both nodes.

find distance from LCA to the nodes.

Order statistics

→ 2th order statistic for a set of n distinct elements is the 2th smallest element.

→ 1st order → 1st order statistic

→ largest elm → nth order "

[10, 2, 3, 8, 12, 15, 19]

1st → 2 → 2

4th → 3 → 8

① Insert (y, z)

② O-rank (y, z)

③ Rank (y, z)

④ Select (y, z) → find smallest / largest element in set S.

Balanced Search tree

A search tree data structure for which a height O(log n)

Eg. AVL Tree

Dynamic order statistics

① select (S, i) → returns the elm with rank i in the set S.

② Rank (S, x) → returns the rank of x in the sorted set of elements.

Approach - Balanced binary search tree.

Node → key, left, right, size

size [x] denotes the no of keys in the subtree rooted at [x]

i
y
z
select (y, z) → 4th order statistic
returns the 2th smallest key
select (root, 4)
root
r = size[left[y]] + 1
if r == i
return select (y, i)
else if r > i
return select (left[y], i)
else
return select (right[y], i - (r + 1))
i = 4
r = 3
r = 5
r = 2
r = 3
r = 1
r = 1
r = 1