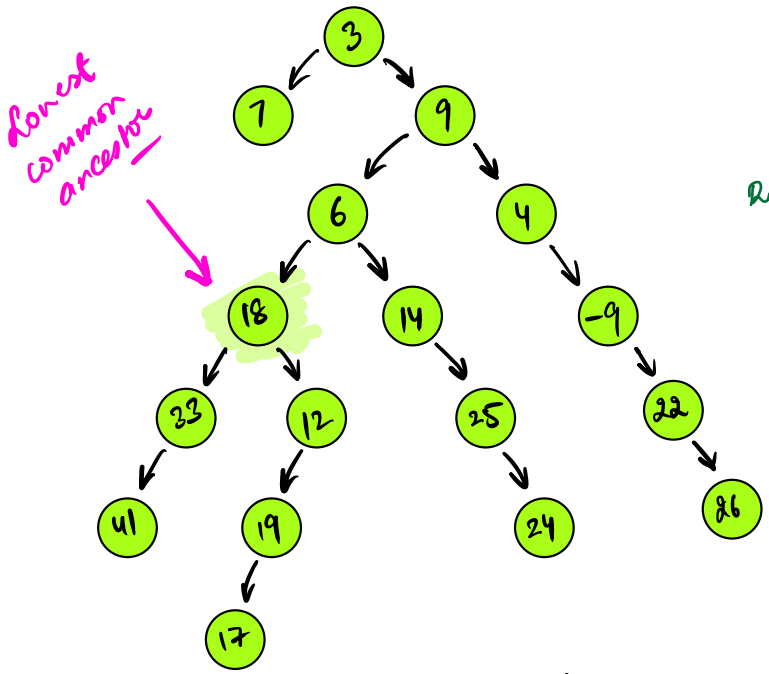


41 → 17

41 → 33 → 18 → 12 → 19 → 17

Root 41 :- 3 9 6 18 33 41

17 :- 3 9 6 18 12 19 17

12 to 25

R to 12 : 3 9 6 18 12

R to 25 : 3 9 6 14 25

O(N)

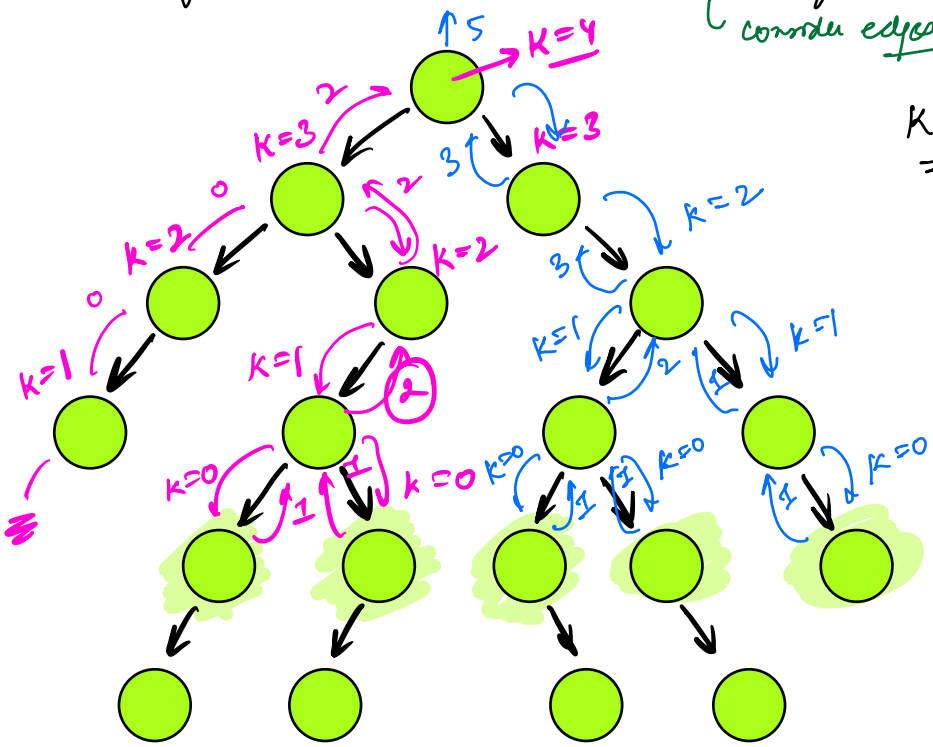
CCA

path from root to A

path from root to B

start traversing until you find a non-common node

• count of nodes which are K distance from the root
↳ consider edges



K=4
↓
count of nodes in K^{th} level

→ count no of nodes at a distance K from the root in its subtree

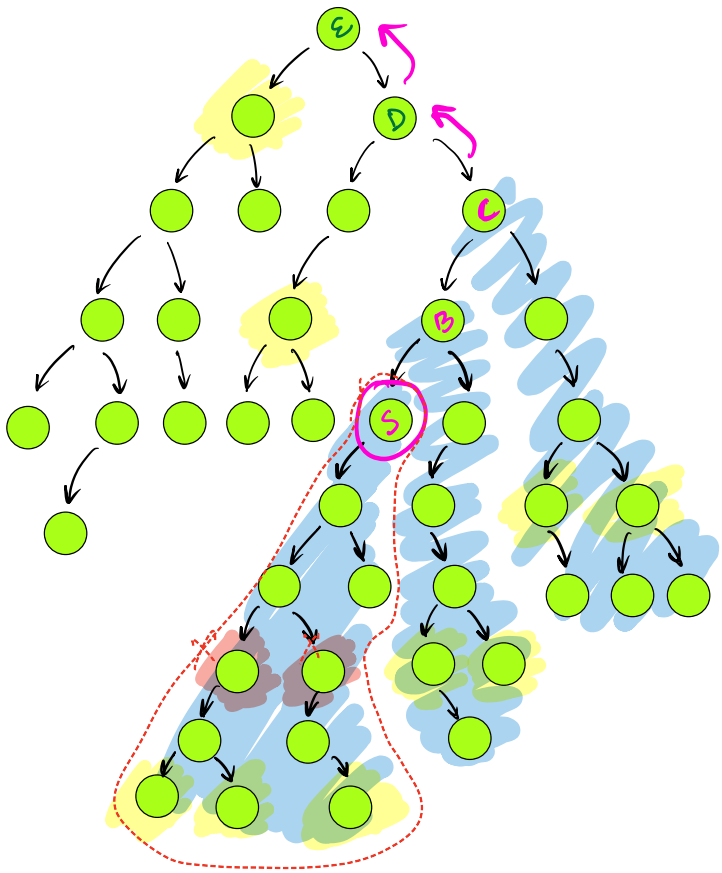
```
int count ( Node root, int K)
{
    if ( root == NULL) return 0;

    if ( K == 0) return 1;

    int l = count ( root.left, K-1);
    int r = count ( root.right, K-1);

    return l + r;
}
```

o count of nodes at k^{th} distance from any particular node



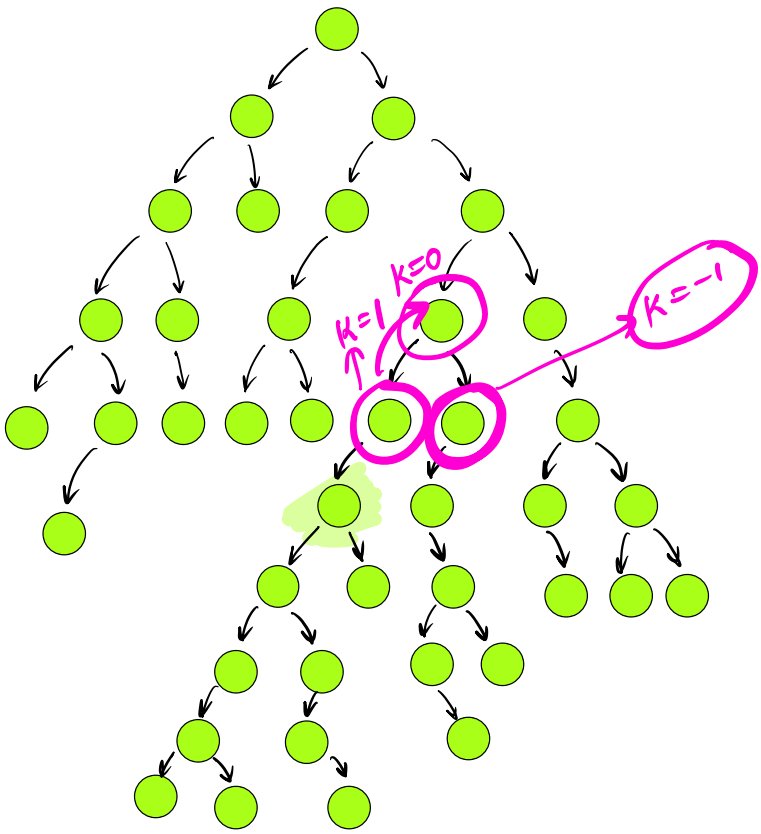
K=5

count(S, K) = 3 (5)
K = K-1; (K=4)
~~count(B, K);~~
count(B.right, K-1) = 2 (3)
K = K-1; (K=3)
count(C.right, K-1) = 2
K = K-1 (K=2)
count(D.left, K-1) = 1
K = K-1 (K=1)
count(E.left, K-1) = 1 (0)

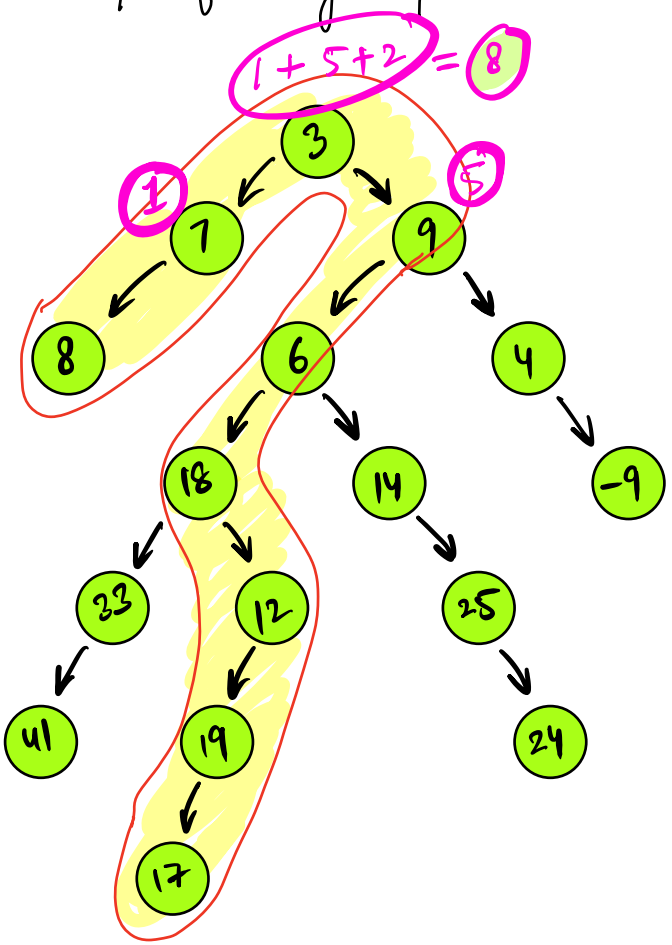
path : S B C D E (path)

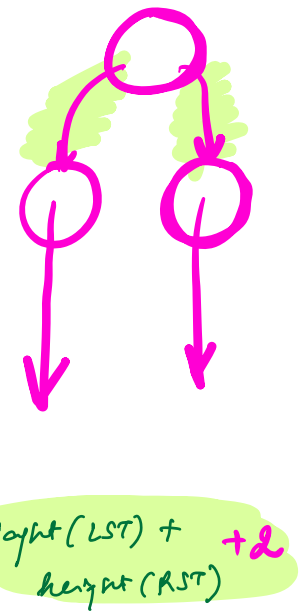
path

```
ans += count(path[0], K);
K = K-1;
for( i=1; i< path.size(); i++)
{
    if(K==0) {ans++; break;}
    if( path[i] == path[i].left)
        ans += count( path[i].right, K-1);
    else
        ans += count( path[i].left, K-1);
    K = K-1;
```

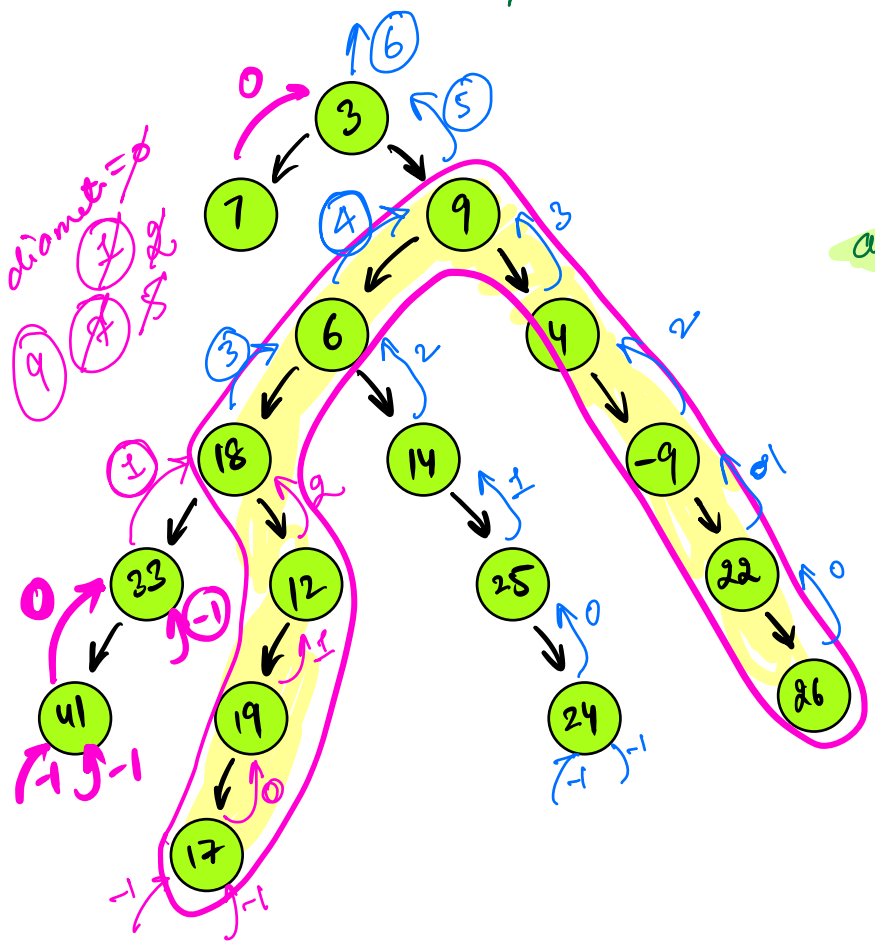

K=1
K=0
K=-1

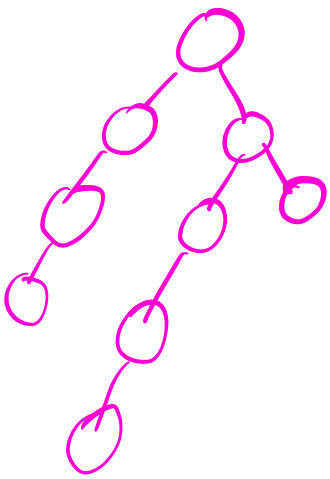
- Find the lngth of longest path across root.

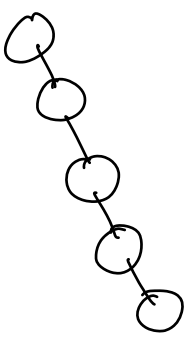




- Diameter/width of the tree
  - ↳ longest distance b/w any 2 nodes







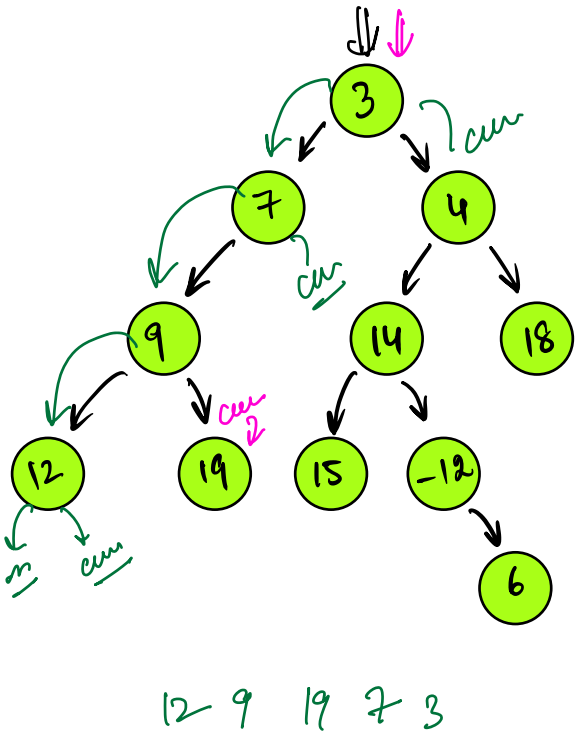
```
diameter = 0;
int height (Node root)
{
    if (root == null) return -1;
    int l = height(root.left);
    int r = height(root.right);
    diameter = max(diameter, l+r+2);
    return max(l, r)+1;
}
```

T.C: O(n)

S.C: O(H)

0 inorder iterative

left root right



12 9 19 7 3

curr = root

stack s;

while (!s.empty() || curr != null)

{ while (curr != null)

{ s.push(curr);

curr = curr.left;

}

curr = s.top();

print (curr.data);

s.pop();

curr = curr.right;

}