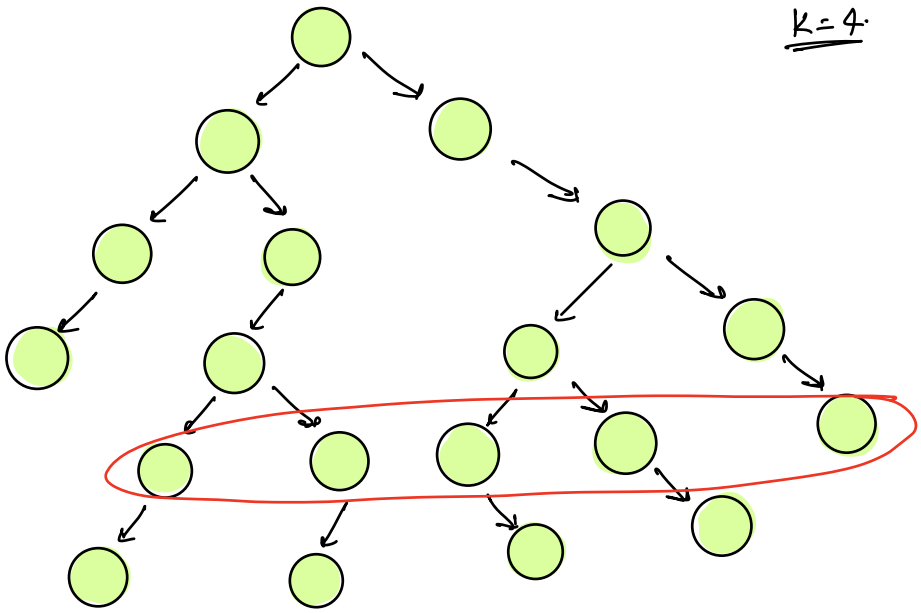
Q: Count of nodes which are at K-distance away from root node.

(in terms of edges)

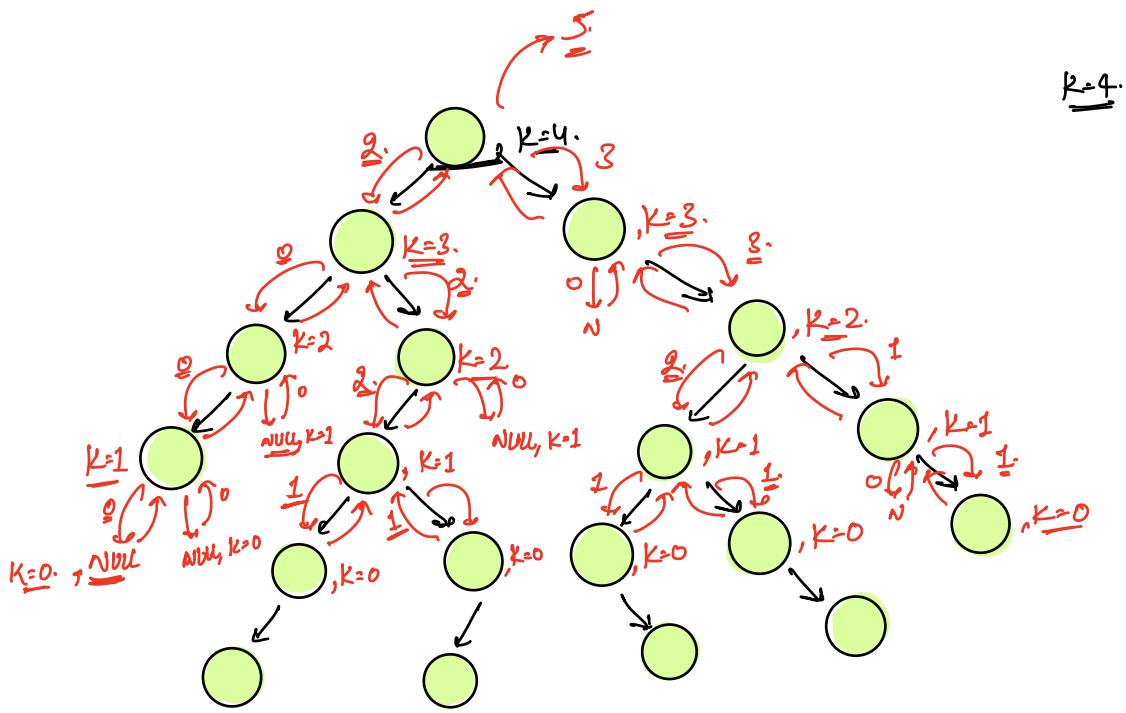
$K = 4$.



ans = count of nodes at level K.

⇕

Level Order Traversal.



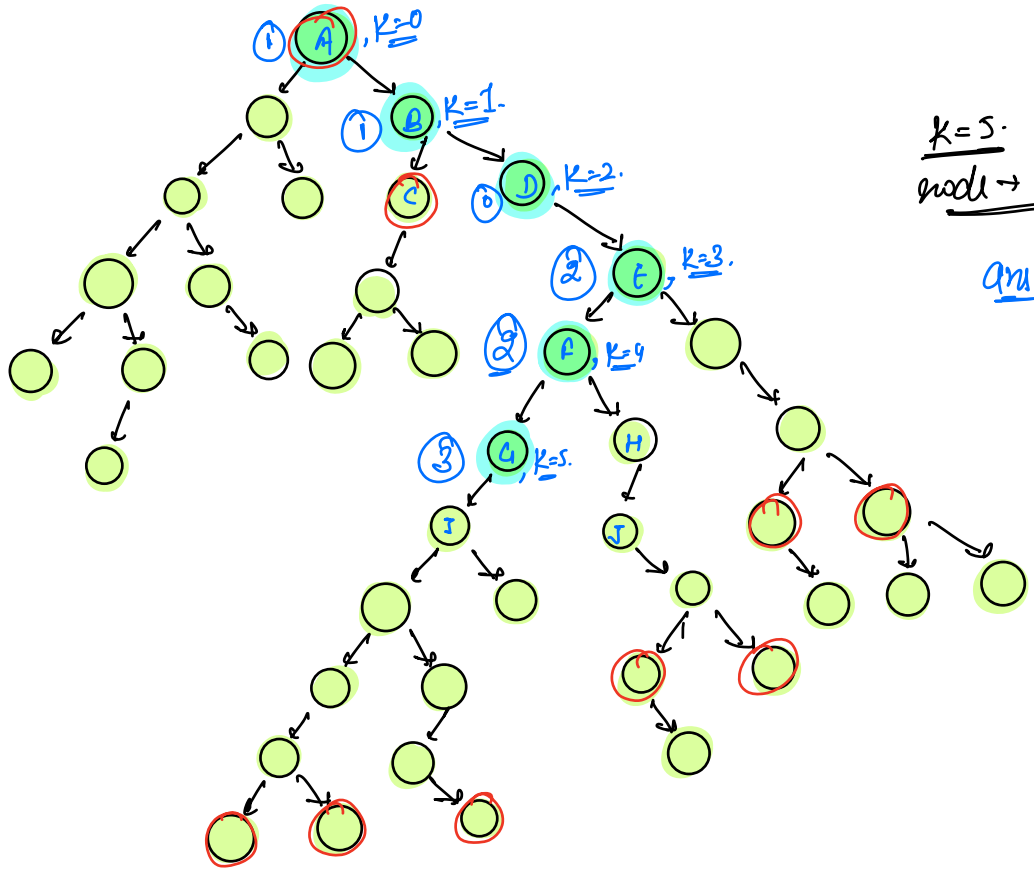
pseudocode.

```
int count (Node root, int K){
    if (root == NULL) { return 0 }
    if (K == 0) { return 1 }

    lans = count(root.left, K-1);
    rans = count(root.right, K-1);
    return lans + rans;
}
```

T.C → O(N)
S.C → O(K)

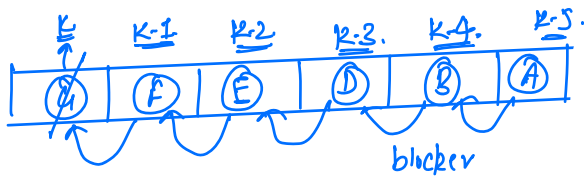
Q: Count no. of nodes which are K distance away from given node.



K=5.
node → G

ans → 9.

① - find path from node to root




```
ans += count( G, K, null );
ans += count( F, K-1, G );
ans += count( E, K-2, F );
ans += count( D, K-3, E );
ans += count( B, K-4, D );
ans += count( A, K-5, B );
```

```
int count (Node root, int K, Node blocker ){
    if (root == NULL || root == blocker || K < 0 ) { return 0 }
    if (K == 0){ return 1 }

    lans = count(root.left, K-1, blocker);
    rans = count(root.right, K-1, blocker);
    return lans + rans;
}
```

T.C → O(N), S.C → O(H)

// Store node to root path in a list.

G	E	E	D	B	A
0	1	2	3	4	5

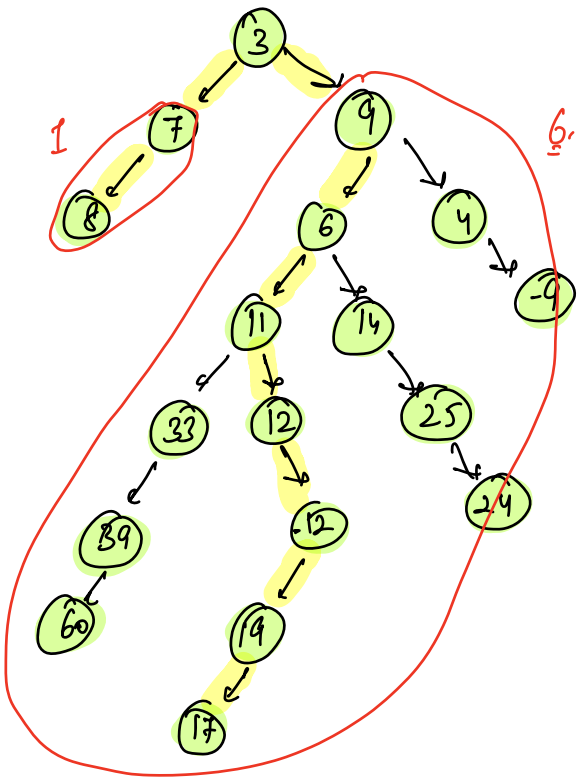
```
ans += count( list.get(0), K, NULL);

for( i = 1; i < list.size() ; i++){
        ans += count( list.get(i), K-i, list.get(i-1));
}

return ans;
```

{T.C → O(N), S.C → O(H)}

Q) Find the largest path across the root.



ans = 9.

ans :

[ht of lst + ht. of rst + 2]

{ ans → height(root.left) + height(root.right) + 2. }

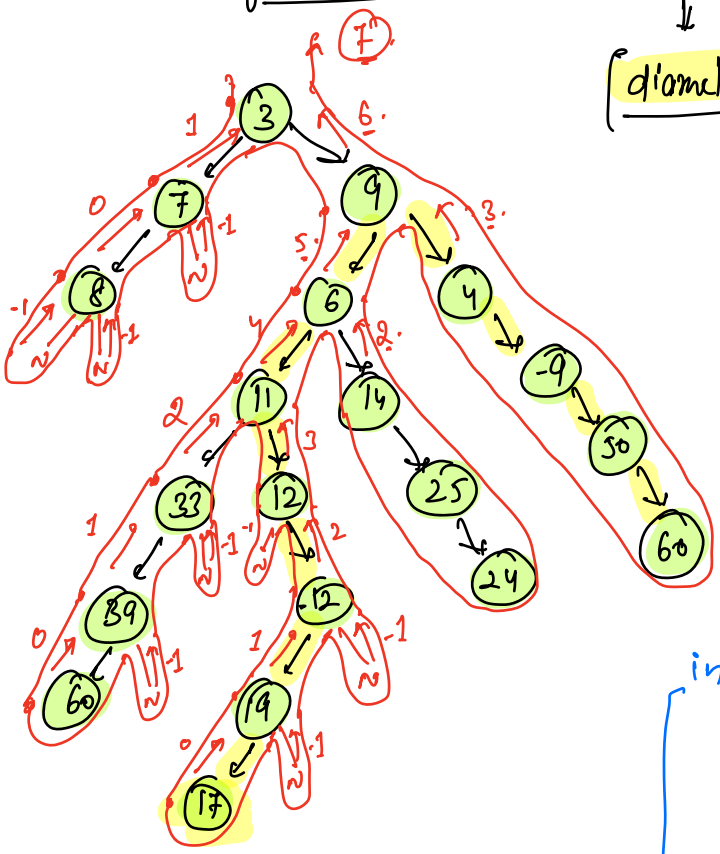
{ T.C → O(N)
S.C → O(H) }

Q: Find longest path b/w any two nodes in the tree.

↓

[diameter of tree]

ans → 10.



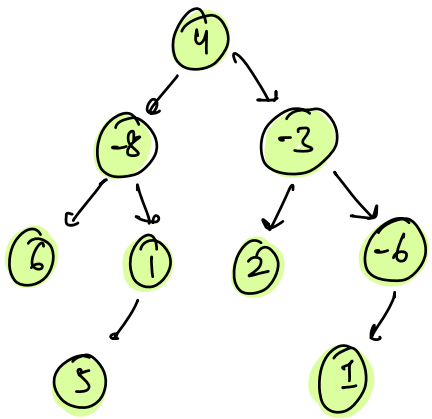
diameter → ~~8~~ 10

diameter = 0

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

T.C → O(N)
S.C → O(H)

Q.) **Maximum Path Sum.**



maxPathSum = $-\infty$

```
int sum( Node root) {
    if (root == null) {return 0}
    l = sum(root.left);
    r = sum(root.right);
    [maxPathSum → ? ]
    return l + r + root.data;
}
```

40-minutes.