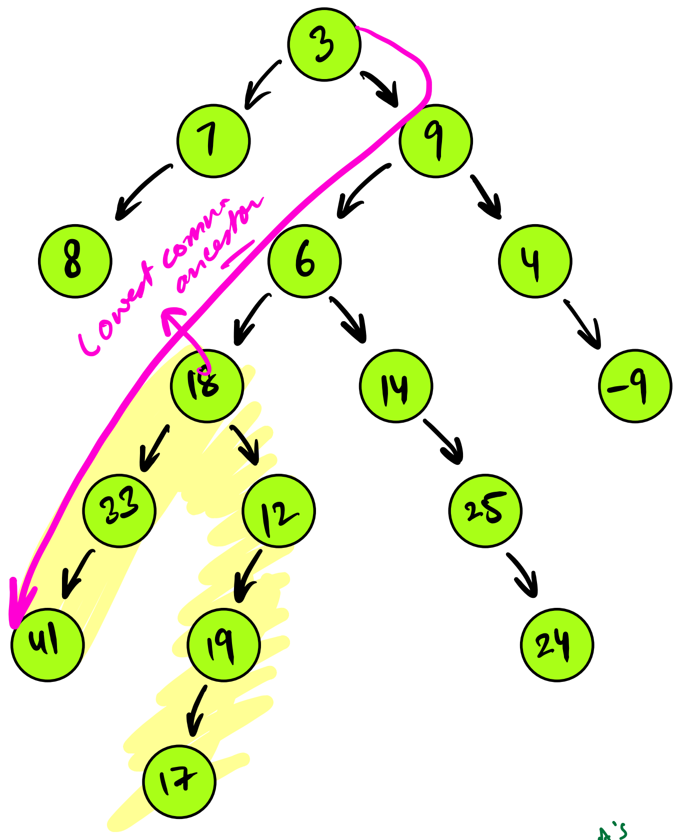


41 ⇒ 17

41 — 33 — 18 — 12 — 19 — 17

Root to 41: 3 9 6 18 33 41

Root to 17: 3 9 6 18 12 19 17

12 to 25

12 18 6 14 25

i

R to 12: 3 9 6 18 12

R to 25: 3 9 6 14 25

i-1th

41 to 18:

41 → 33 → 18

i-1 i

R to 41: 3 9 6 18 33 41

R to 18: 3 9 6 18

total nodes in root → A's path

A: 0 → x-1

B: 0 → y-1

~~total~~ nodes in root → B's path

i ↴ first non-common node

T.C: $O(n)$

x-1 → i-1 & i → y-1

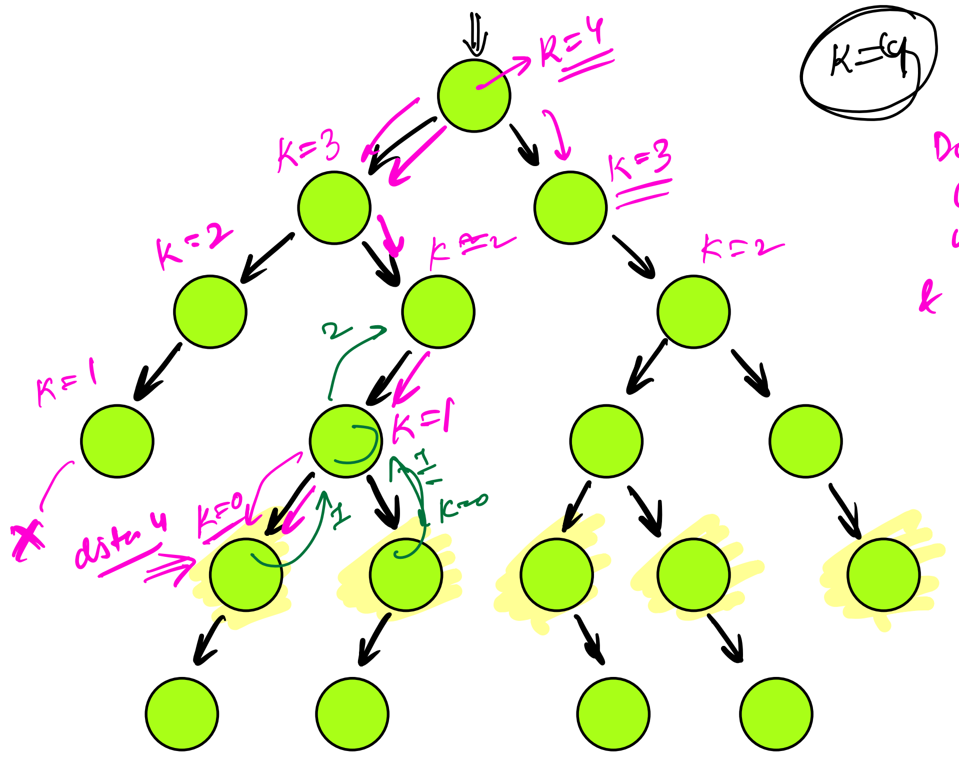
i-1 i

	0	1	2	3	4	5	6
41:	3	9	6	18	33	41	
17:	3	9	6	18	12	19	17

x=6

y=7

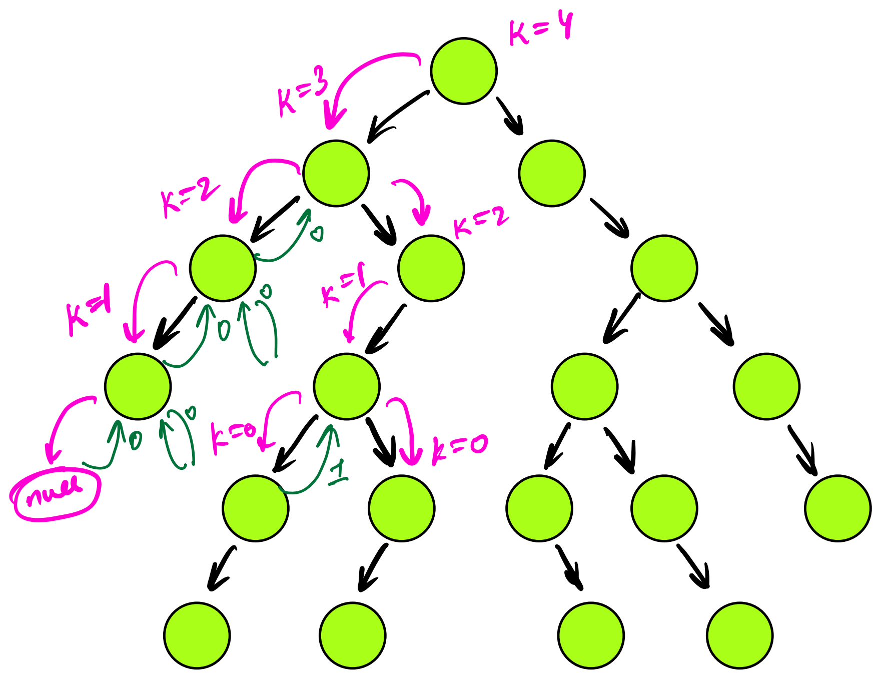
• count all nodes which are at a distance k from root.



$\boxed{K=4}$

Do the level order traversal with null node & start count after K^{th} null node

$\boxed{ans=5}$



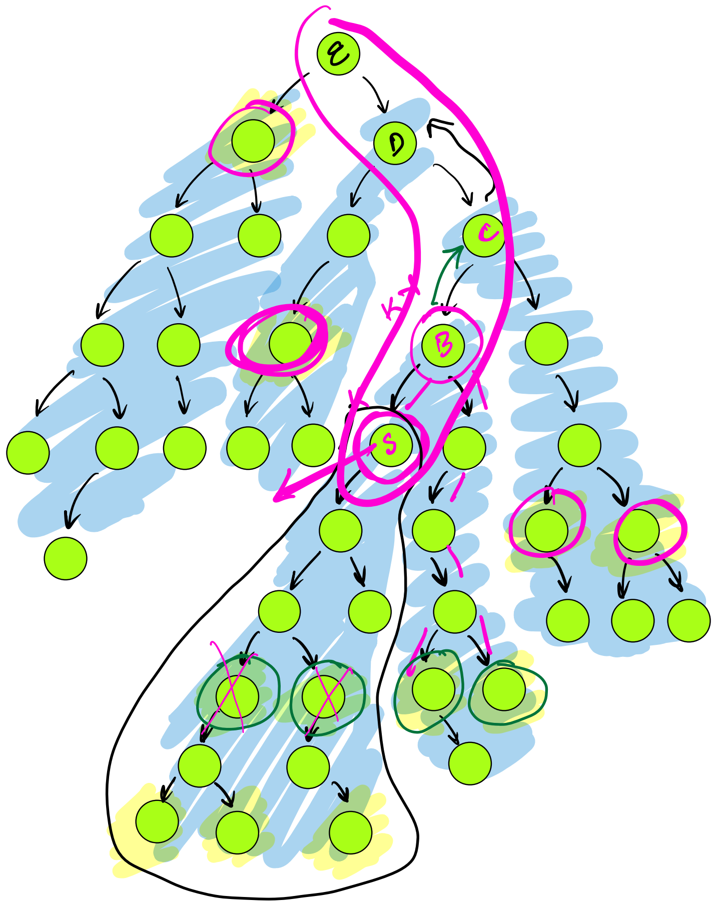
```
int  count(  root,  int k)
{
        if ( root==null) return 0;

        if (k==0) retun 1;

      Int x  = count( root.left, k-1);
      int y  = cout(  root.right ,k-1);

            return x+y;
}
```

count of nodes at a distance from any given node



$\boxed{K=5}$

count (node , K) = 3 $\xrightarrow{} 5$

~~count (B , K-1)~~ $\to 4$

4 $\;$ K = K-1

count (B.right , K-1) = 2 $\to 3$

3 ← K = K-1

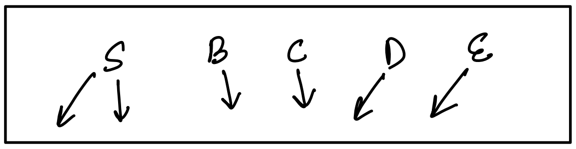
count (C.right , K-1) = 2 $\to 2$

2 ← K = K-1

count (D.left , K-1)
$\boxed{1}$

1 ← K = K-1

count (E.left , K-1)
$\boxed{0}$



// store the nodes in the path from root to give node

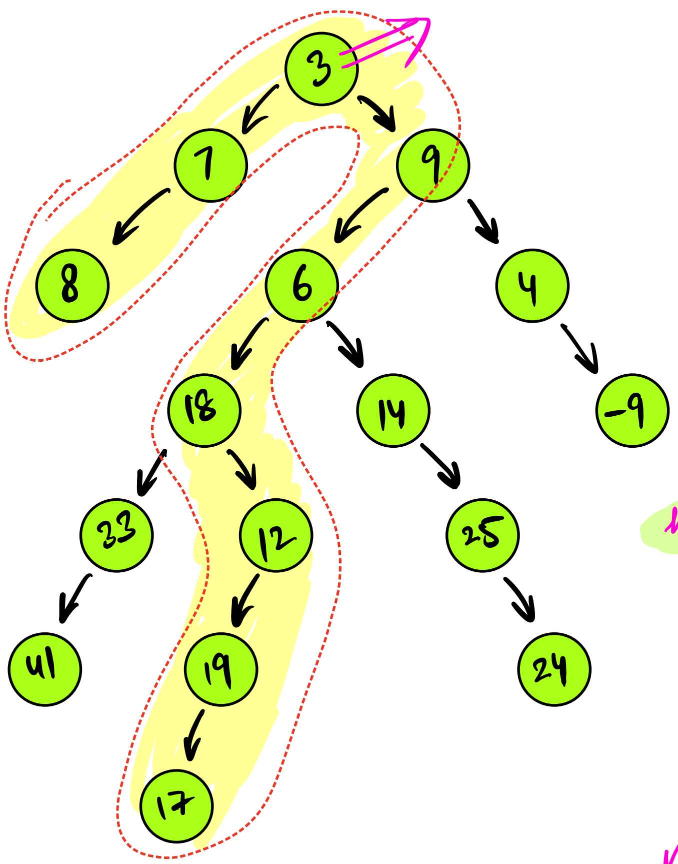
path →

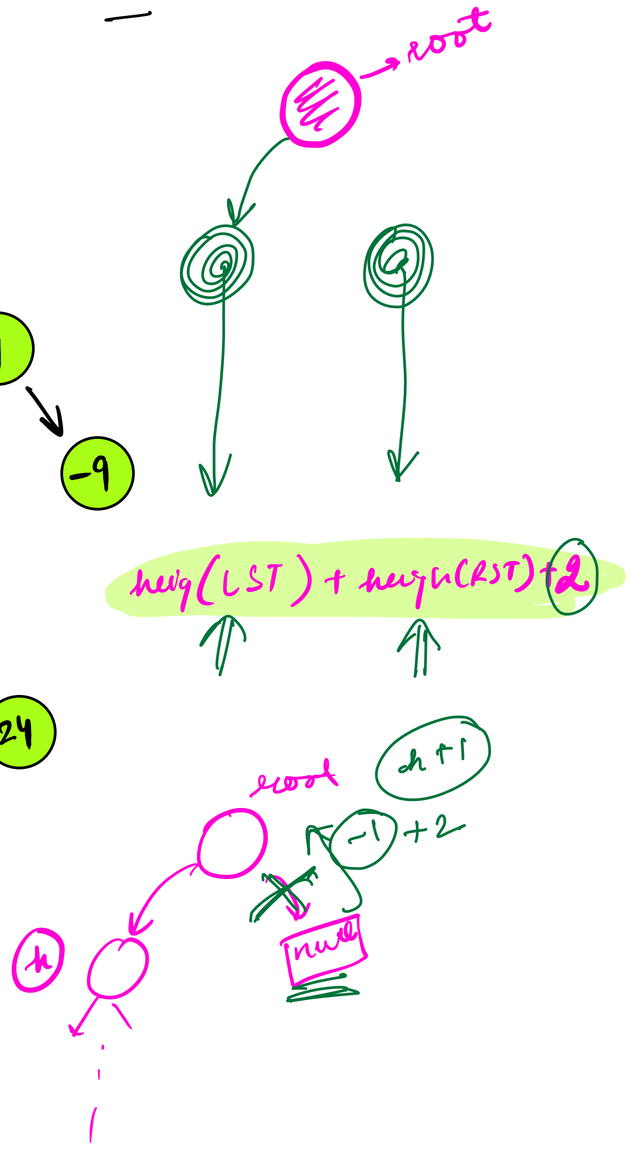
S	B	C	D	E

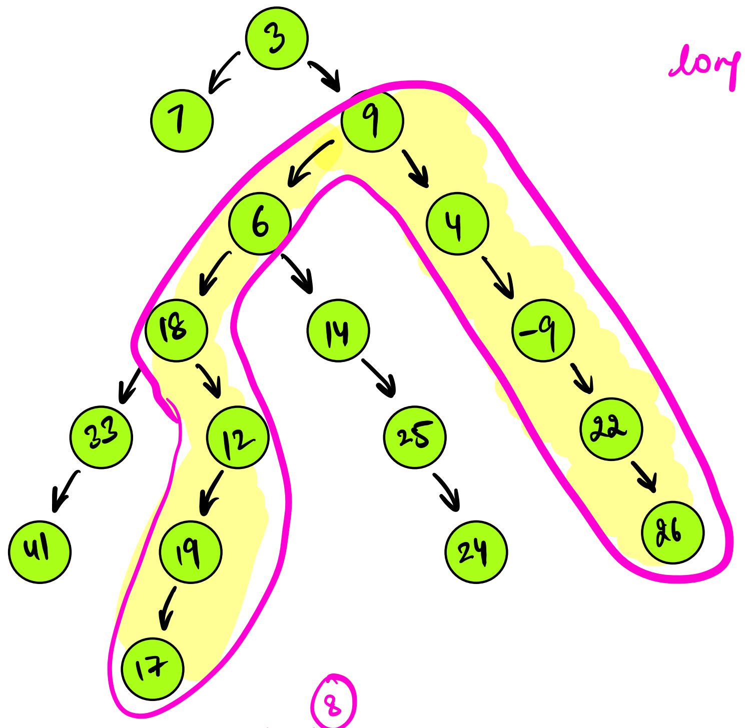
```
ans += count( path[0], K);
K = K-1;
for( i=1;   i< path.size ; i++)
{
        if(K==0){ ans++; break;}
        if( path[i].left == path[i-1])
        {
        ans += count ( path[i].right, K-1);
        }
        else
        {
        ans += count (path[i].left, K-1);
        }

        K=K-1;
}
```

• Find longest path across the root.



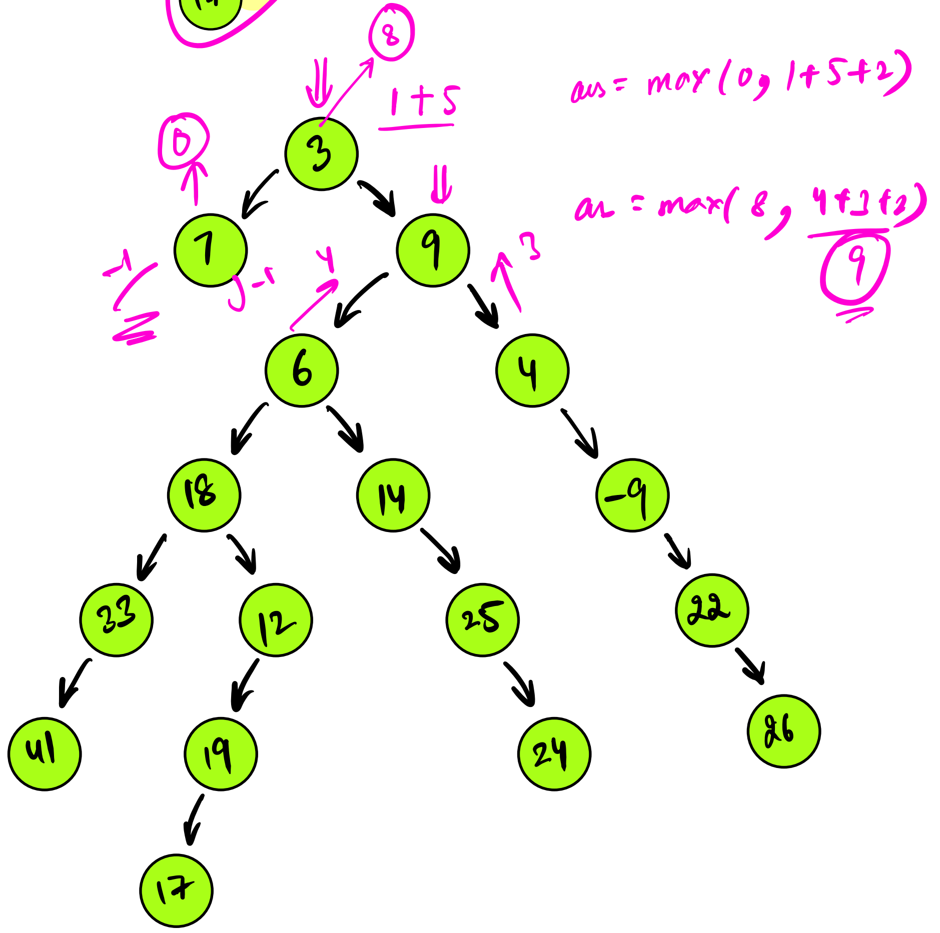




longest path b/w any 2 nodes!

↓

diameter/width of tree



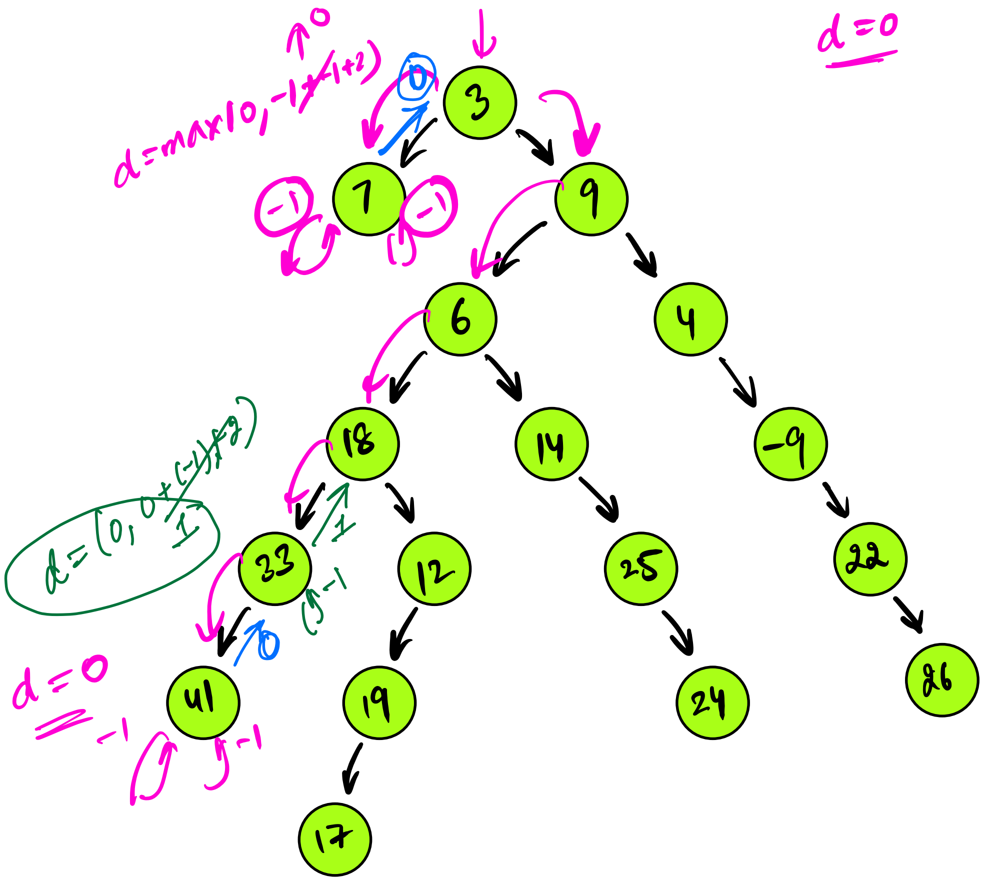
ans = max(0, 1+5+2)

ans = max(8, $\underline{4+3+2}$)

9

```
int diameter =0;
int    height( Node root)
{
        if( root = =null) retur -1;

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



d=0
d=max(0,-1+-1+2)
0
d=(0,0+(-1)+2)
1
d=0
-1
-1
0
1