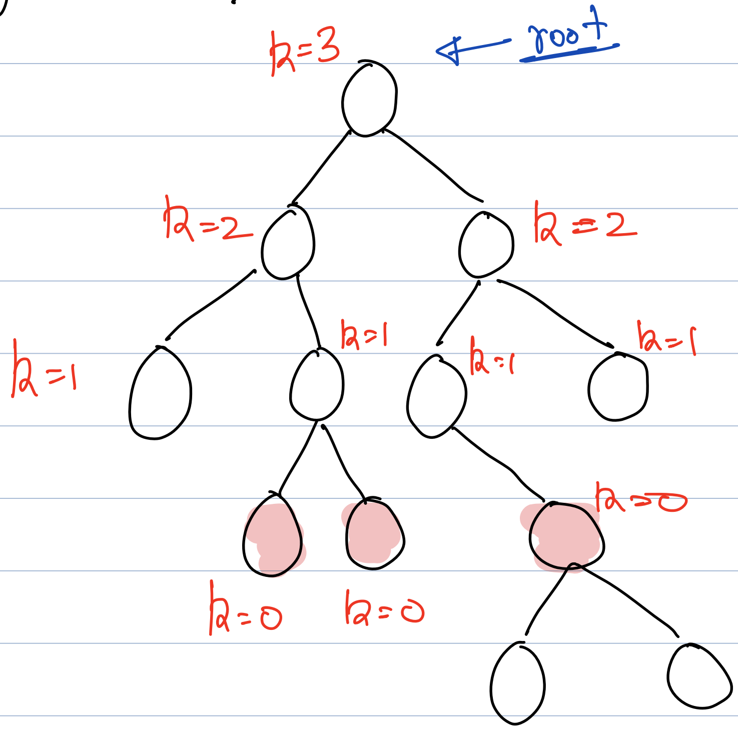
Q Given a BT. Calculate the no of nodes at distance K from root!.



K ⇒ 3

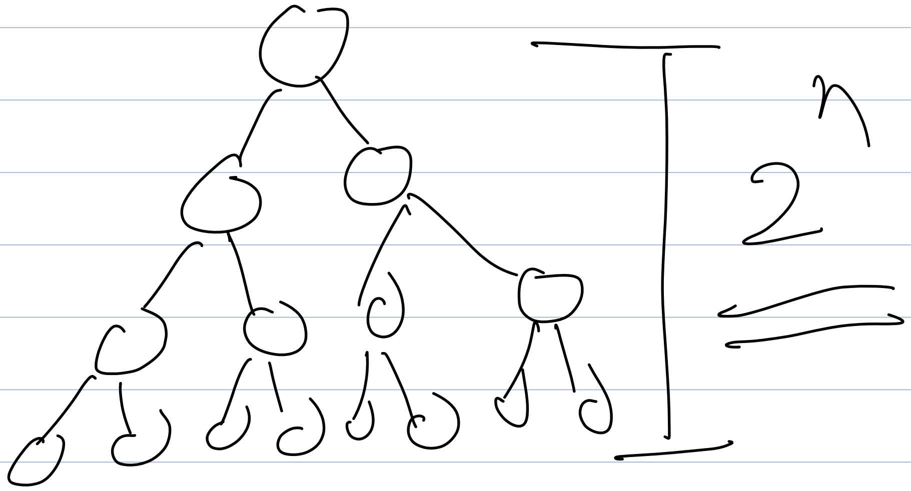
ans = 3

```
int  countNodes ( Node root, int k) {
                                                  → if(k<0)
     if (root ==NULL)  return 0;                       return 0;
     if (k == 0)
                  return 1;

     int l ⇒  countNodes (root.left, k-1);
     int r ⇒  countNodes (root.right, k-1);

     return l+r;
}
```

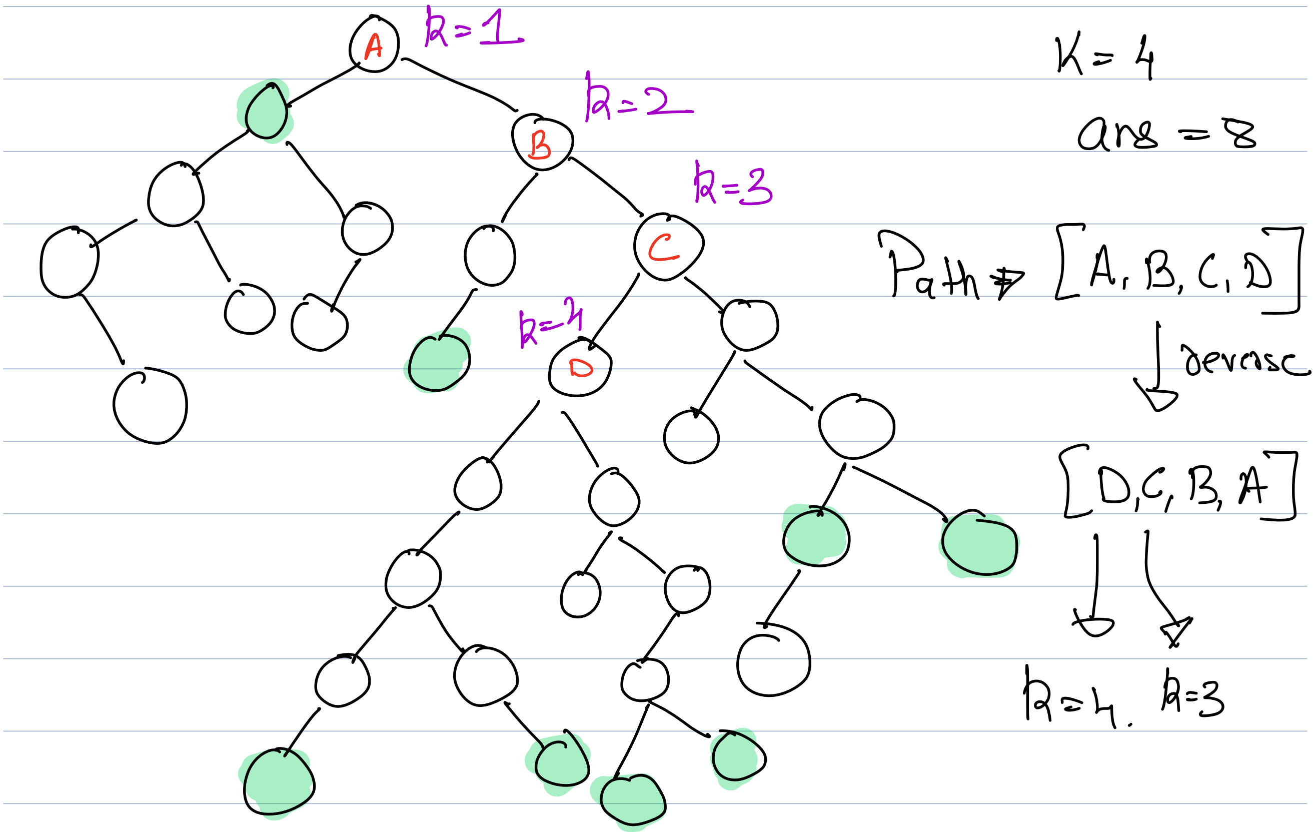
Tc: $O(N)$ | $O(2^k)$ | $\boxed{k=100}$

Tc $\Rightarrow$ $O(\min(N, 2^k))$;

SC $\Rightarrow$ $O(\min(h, k))$



Q Given a BT. Calculate the no of nodes at distance K from a given node !



K= 4

ans = 8

Path ⇒ [A, B, C, D]

↓ reverse

[D, C, B, A]

k=4. k=3

```
int CountAllNodes ( Node source, Node root, int k) {

    list<Node> path => findPath (source, root);
    reverse (path);

    int count => countNodes (path[0], k);

   for (int i = 1 ; i < path.size() ; i++)  {

      if ( path[i].left == path[i-1])
         count = count + countNodes ( path[i].right,
                                      k-i-1 )
      else
         count = count + countNodes ( path[i].left,
                                      k-i-1 )
   }

   return count;
}
```

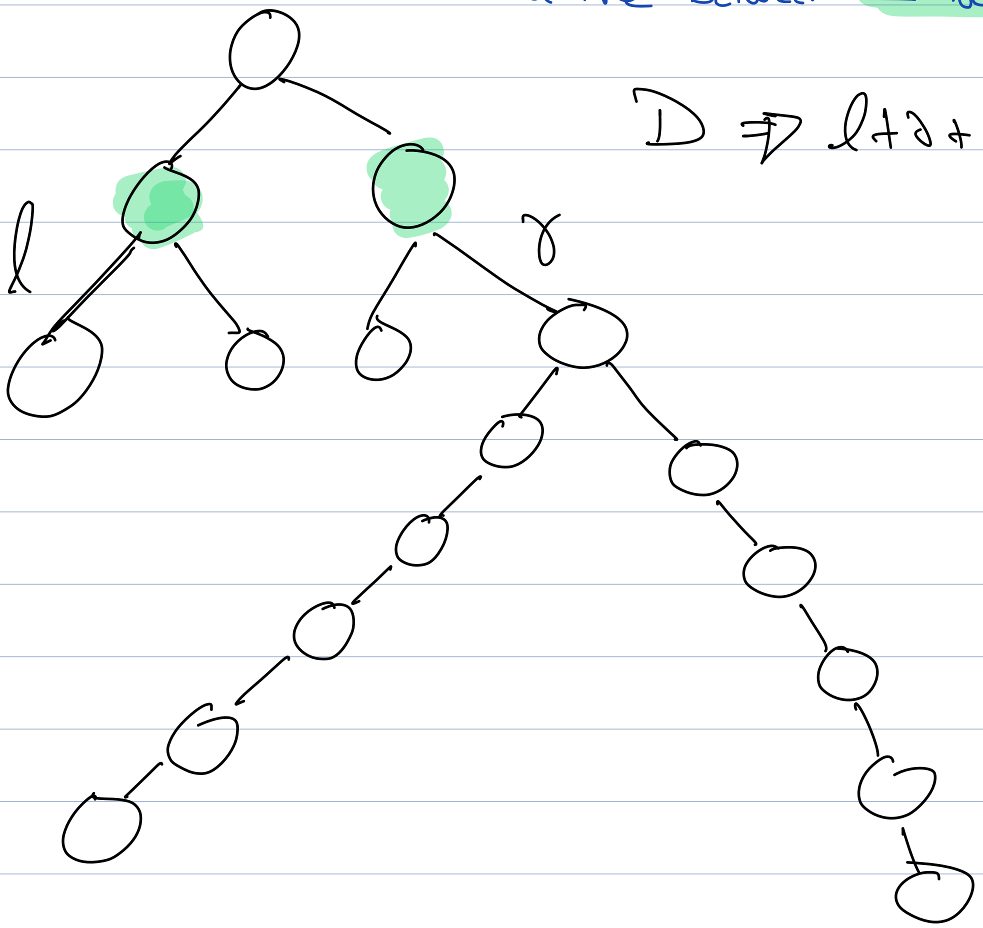
TC: O(n)

SC: O(H)

Q Calculate the length of the diameter of a tree.

$\downarrow$

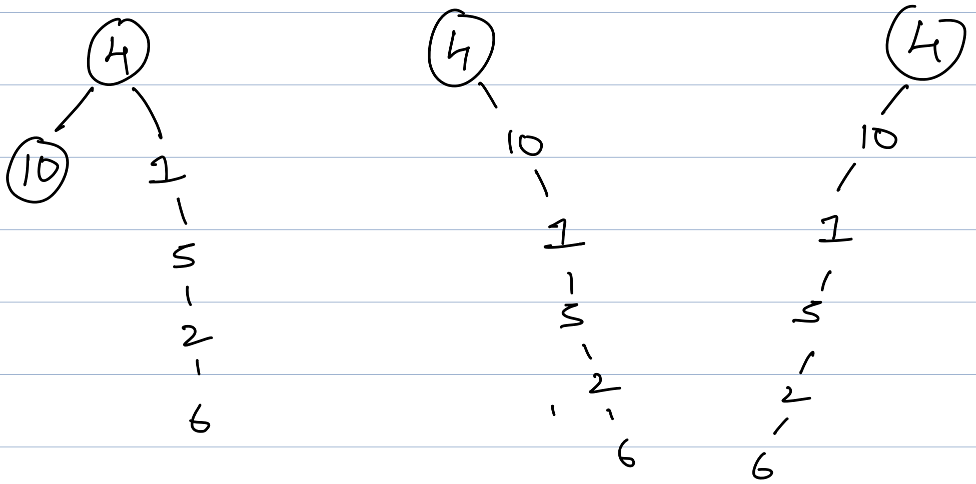
longest path in a tree between 2 nodes!

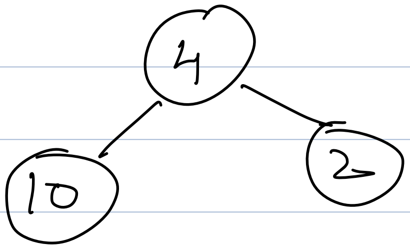


D $\Rightarrow$ l+r+2

Preorder ⇒ Root LsT RsT

4 10 1 5 2 6





⇒ CAN'T DRAW BT.

Postorder : 4 10 1 5 2 6

Inorder : 4 10 1 5 2 6

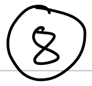
⇒ Pre order + Inorder

⇒ Post order + Inorder.

Distinct Element

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Pre :	8	6	5	15	19	9	18	25	4	7	41	30	39	48
In :	15	5	19	6	18	9	25	8	7	41	4	39	30	48
	is							ind						

ps+1, ps + lengt



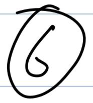
ind − is + ~~1~~ − ~~1~~

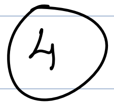
In : 15 5 19 6 18 9 25

Pre : 6 5 15 19 9 18 25

In : 7 41 4 39 30 48

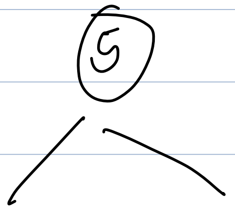
Pre : 4 7 41 30 39 48

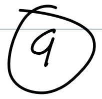




In : 15 5 19 18 9 25

Pre : 5 15 19 9 18 25



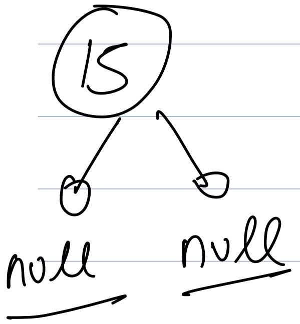


In : 15

Pr 15

In : 19

Pre : 19



```
Node constroct Tree ( in[], pre[], int ps, int pe, int is
                                                   int ie) {
        if ( ps > pe)
               return null;

      int root-val = pre[ps];
      Node x = new Node (root-val);

     int ind = -1;
    for (int i = is ; i ≤ ie ; i++ ) {
           if ( in[i] == root-val)
                   ind = i, break;
    }

       int lst-lenght = ind - is;

  x.left = constroct Tree ( in, pre, ps+1, ps+lst-length
                                    is, ind-1);
  x.right = construct Tree ( in, pre, ps+lst-length+1, pe
                                    ind+1, ie);

   return x;

}
```

// O(1)
↓
using a
map;

TC: O(n)
SC: O(n) → using a

map!!

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Pre :	8	6	5	15	19	9	18	25	4	7	41	30	39	48

Balanced BT

$$|h_{lst} - h_{RST}| \le 1$$

6 5 15 19 9 18 25 ~~1~~

Inorder

Postorder

LST Root RST