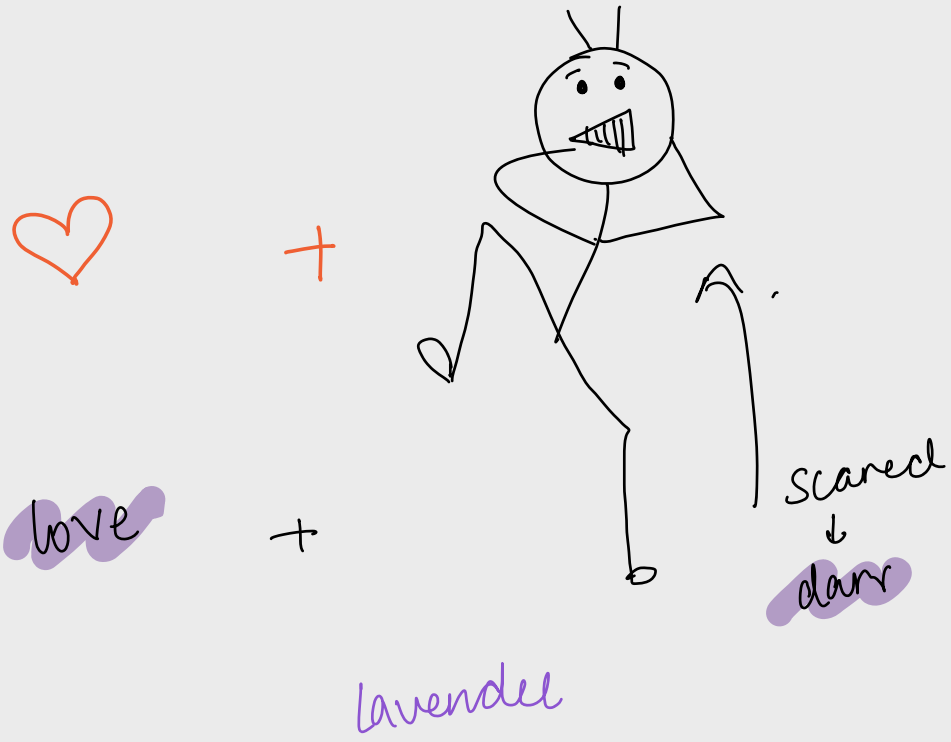
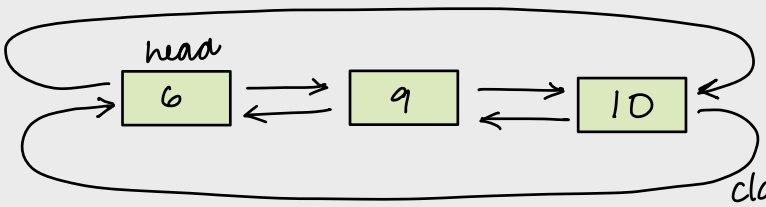
- Binary Search Tree to Circular Doubly Linked List
- All nodes at K distance
- All nodes at K distance from node n
- Longest path across the root
- Diameter
- Two nodes in BST are swapped

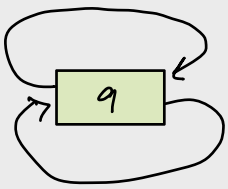


lavender

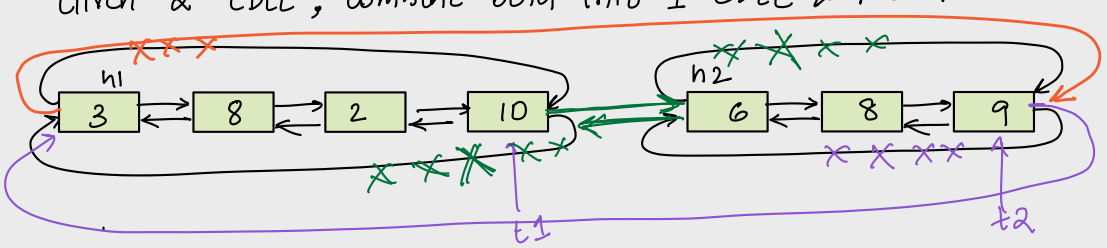
Circular Doubly Linked list

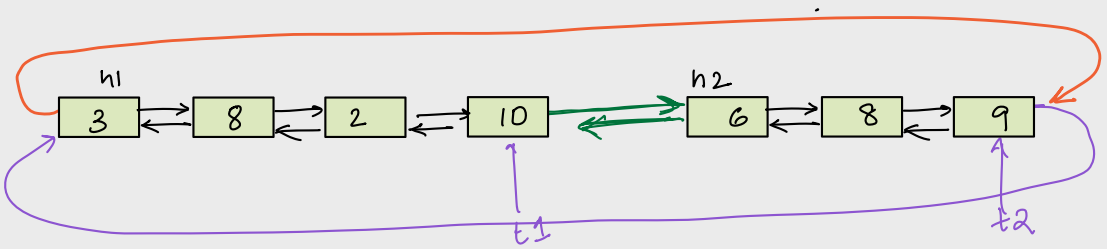



```
class CNode {
    CNode prev
    CNode next
    int data
}
```



Given 2 CDLL, combine both into 1 CDLL & return head





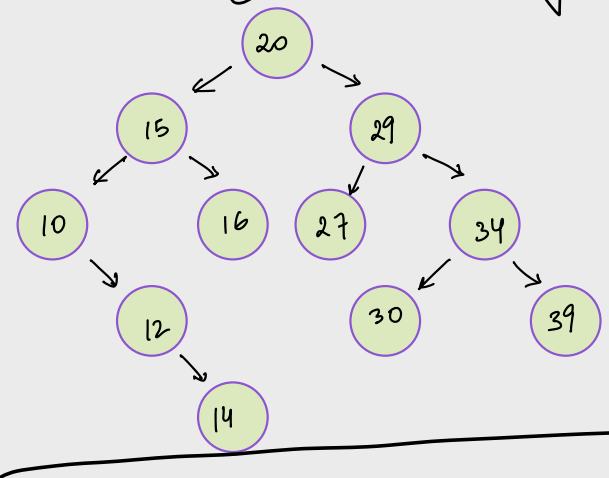

```
CNode combine(CNode h1, CNode h2)
{
    if(h1 == NULL) return h2
    if(h2 == NULL) return h1
    CNode t1 = h1.prev
    CNode t2 = h2.prev

    t1.next = h2
    h2.prev = t1
    h1.prev = t2
    t2.next = h1

    return h1
}
```

TC: O(1)
SC: O(1)

Que: Given BST, convert into sorted circular DLL.

↳ Can't use array to store BST

MICROSOFT.



10 ⇄ 12 ⇄ 14 ⇄ 15 ⇄ 16 ⇄ 20 ⇄ 27 ⇄ 29 ⇄ 30 ⇄ 34 ⇄ 39

// Recursion:

Assumption: Given BST, func converts it to SCDLL & returns the head.

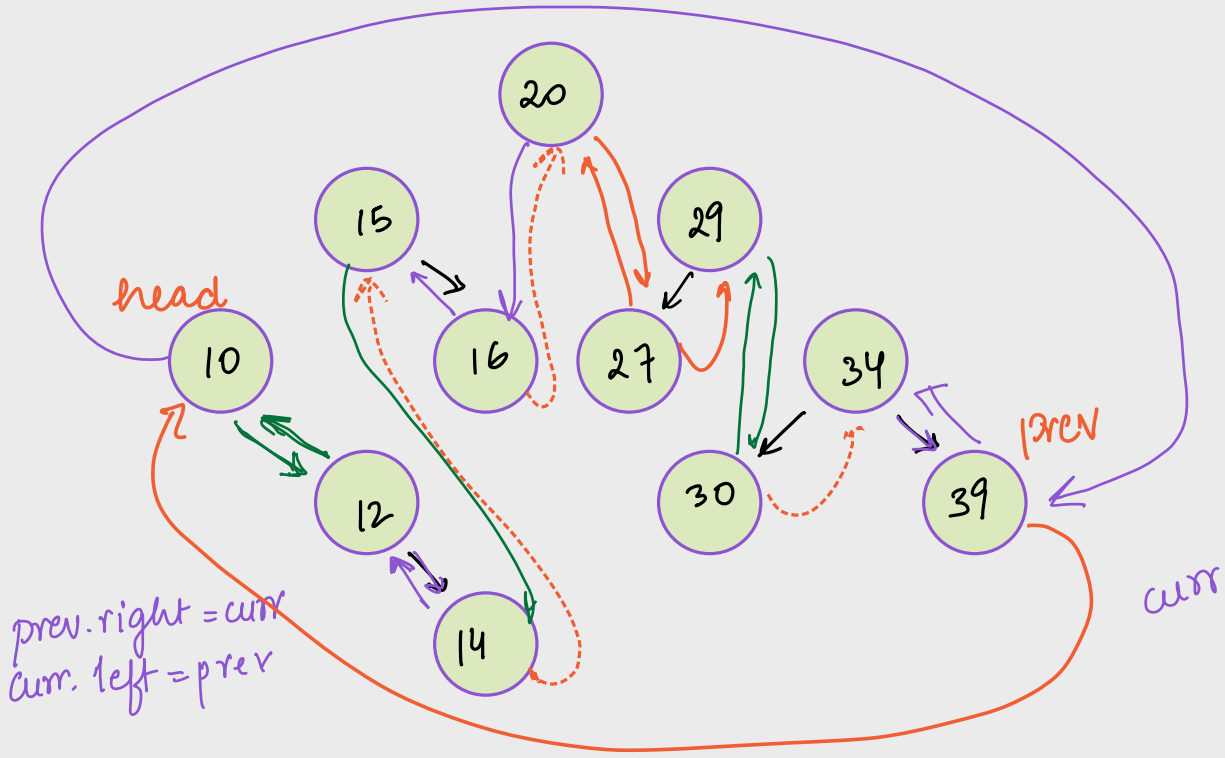
```
CNode convert (CNode root) {
    if (root == NULL) return NULL
    CNode h1 = convert (root.left)
    CNode h2 = convert (root.right)
    root.left = root
    root.right = root
    return combine(combine(h1, root), h2)
}
```

```
class TreeNode {
    left
    right
}
```

TC: O(N)

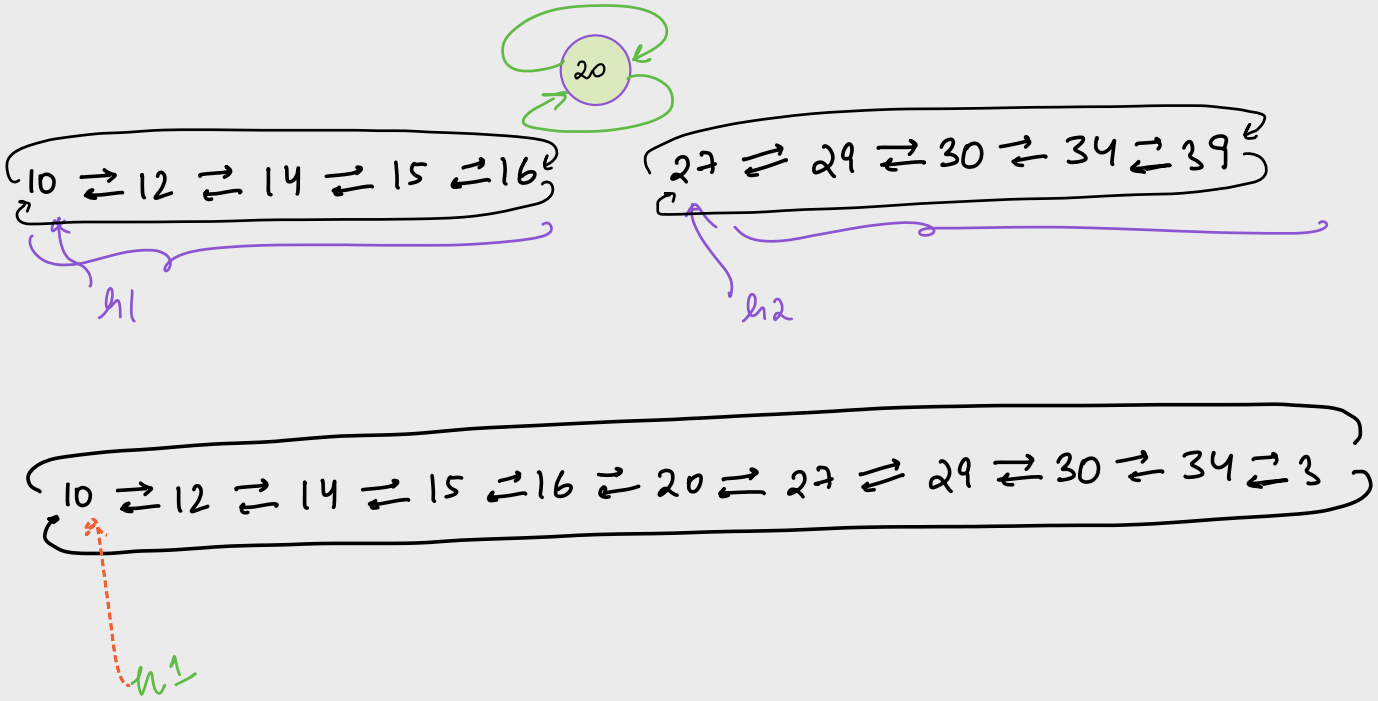
SC: O(H) → SC O(1)? Morris

BST → SCDLL


20
15
29
head
10
16
27
34
prev
12
30
39
curr
14
prev.right = curr
curr.left = prev

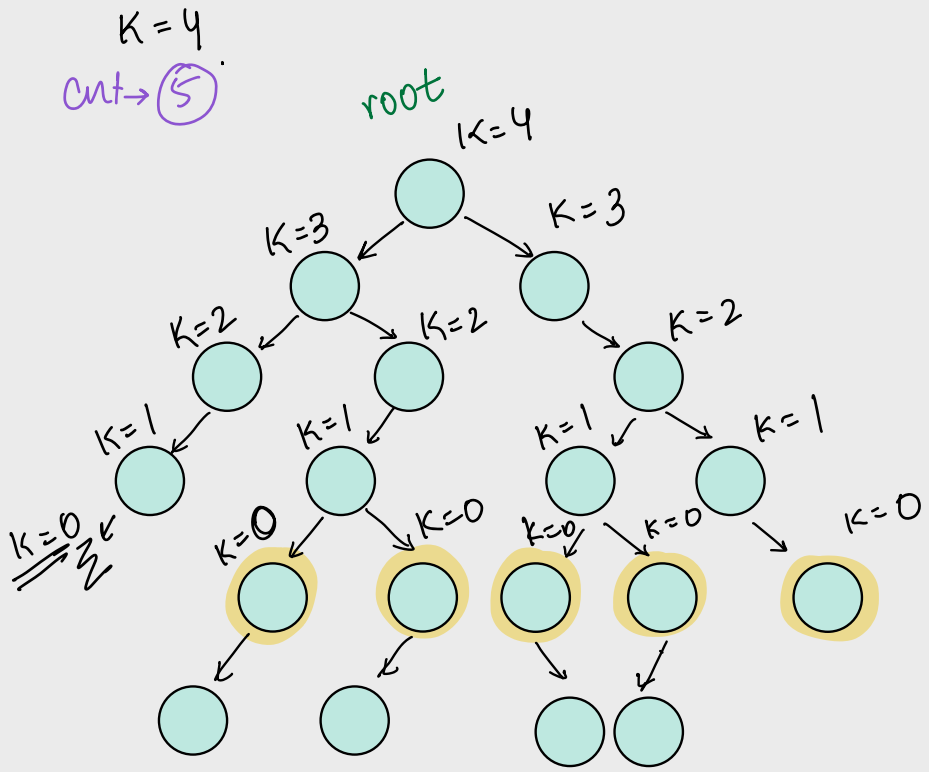
isBST

Adhoc


20
10 ⇄ 12 ⇄ 14 ⇄ 15 ⇄ 16
27 ⇄ 29 ⇄ 30 ⇄ 34 ⇄ 39
h1
h2

10 ⇄ 12 ⇄ 14 ⇄ 15 ⇄ 16 ⇄ 20 ⇄ 27 ⇄ 29 ⇄ 30 ⇄ 34 ⇄ 3
n1

Que. Calculate no. of nodes at a distance K from root node.



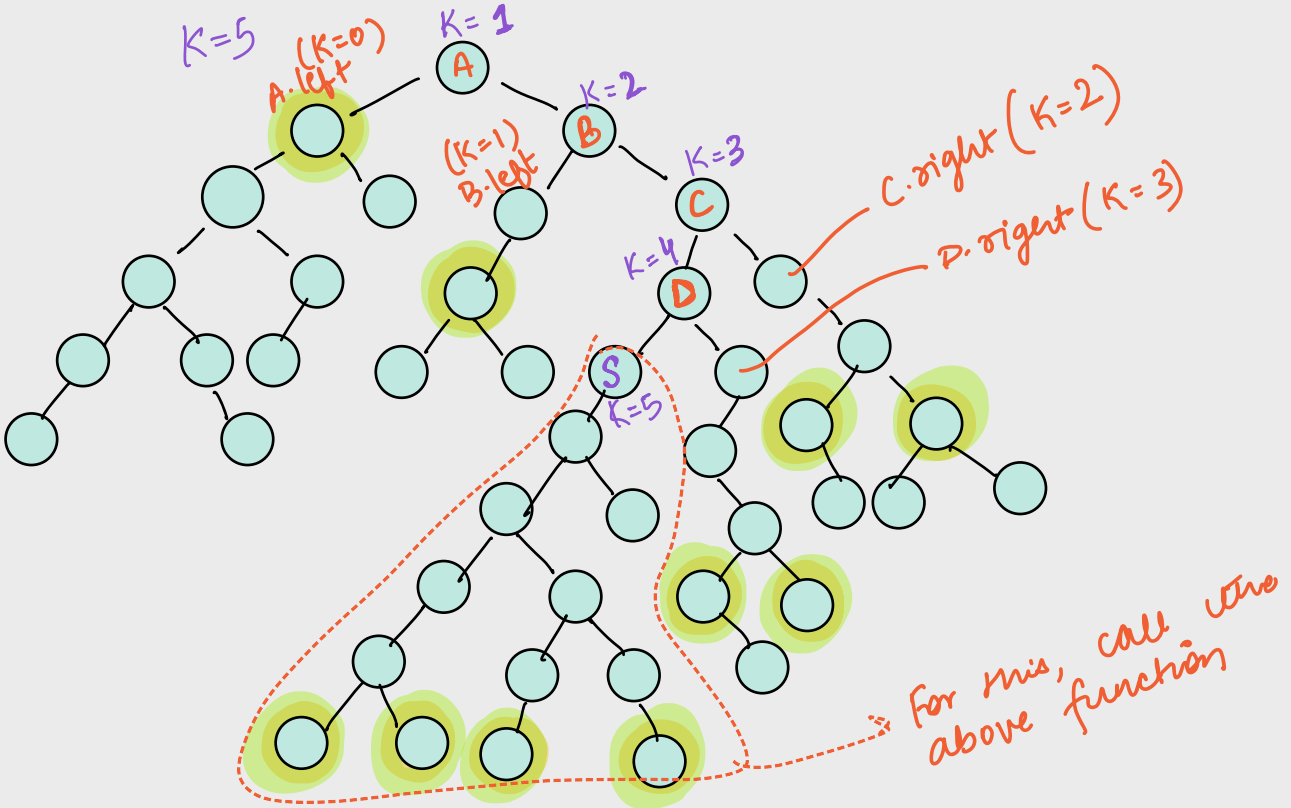

```
int distance (root, int K){
    if (root == NULL || K < 0 ) return 0
    if (K == 0) return 1
    int l = distance (root.left , K-1 )
    int r = distance (root.right, K-1)
    return l+r
}
```

TC : O(N)
SC : O(H)

Break : 10:30 PM

NUTANIX

Que. Calculate no. of nodes at a distance K from given source node



distance (source , 5)

{ distance (D , 4) } ✗

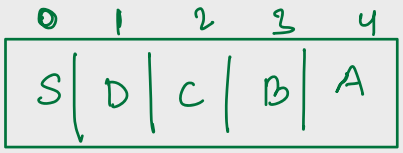
distance (D.right , 3)

distance (C.right , 2)

distance (B.left , 1)

distance (A.left , 0)

list<Node> p



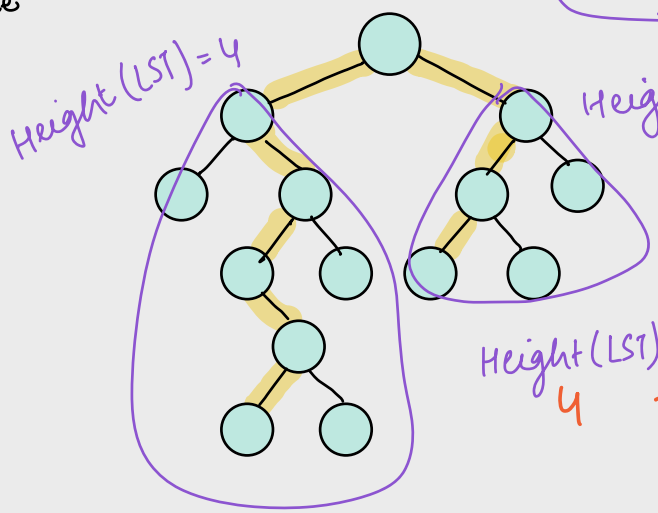

```
int cnt = distance(s, k)
k = k-1
for(i=1; i<p.size(); i++){
    if(p[i].left == p[i-1]){
        cnt += distance(p[i].right, k-1)
    }
    else{
        cnt += distance(p[i].left, k-1)
    }
    k=k-1
}
return cnt
```

TC: O(N)

SC: O(H)

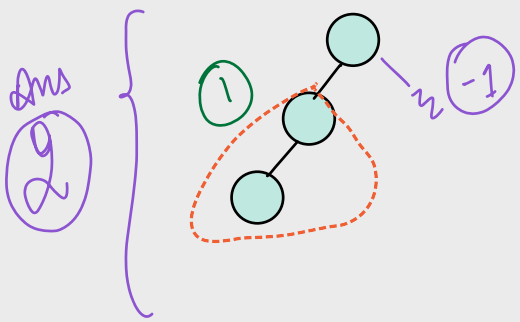
Que. Given a BT, find length of max path going passing through root node

→ no. of edges



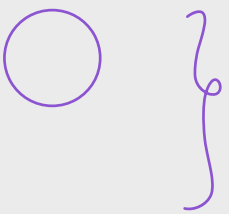
Height (LST) + Height (RST) + 2

4 + 2 + 2



1 + (-1) + 2

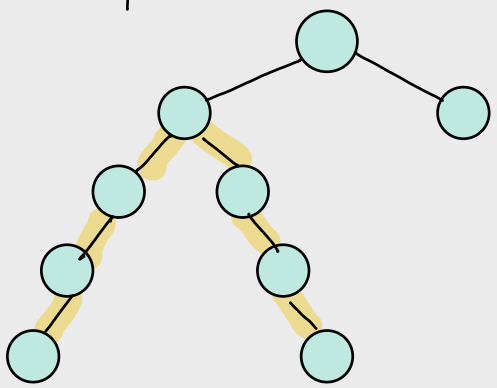
2



-1 + (-1) + 2

-2 + 2 = 0

Que. Find max path in the entire given Binary Tree.



Ans: 6

It's not necessary that it passes through root

Above Que., we found path across root.

For entire tree, path has to pass across one of the nodes.

If we perform above technique across all the nodes, we'll get the max path

For a node, Ht (LST) + Ht (RST) + 2

max of all the max paths will be the answer.

```
int ans = -∞
int height (Node root){
    if (!root) return -1
    int l = height (root.left)
    int r = height (root.right)
    path = l+r+2
    ans = max(ans, path)
    return max(l, r) + 1
}
```

TC: O(N)
SC: O(H)