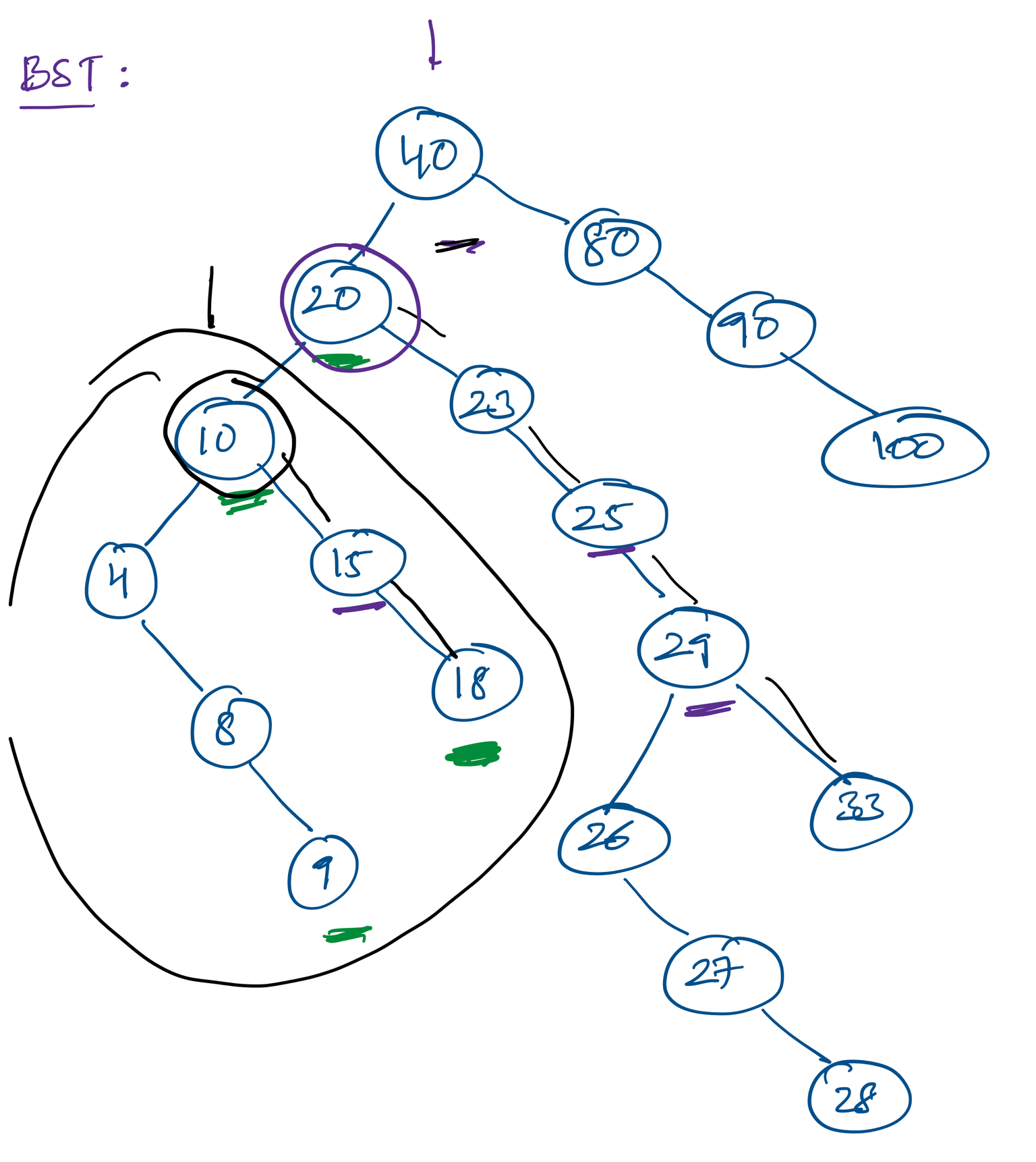
Binary Trees 5

Sunday, 13 June 2021 10:04 AM

BST:



→ Data come befor given data in inorder

- inorder predecessor of 40 : 33
- inorder " of 20 : 18
- " " of 10 : 9
- " " 25 : 23
- " " 15 : 10
- of " " 29 : 28

Obs1: If node has no LST then inorder predecessor is it's parent

Obs2: If node has LST Than inorder predecessor is max in (LST)

⇒ Given a node with leftsubtree find its inorder predecessor

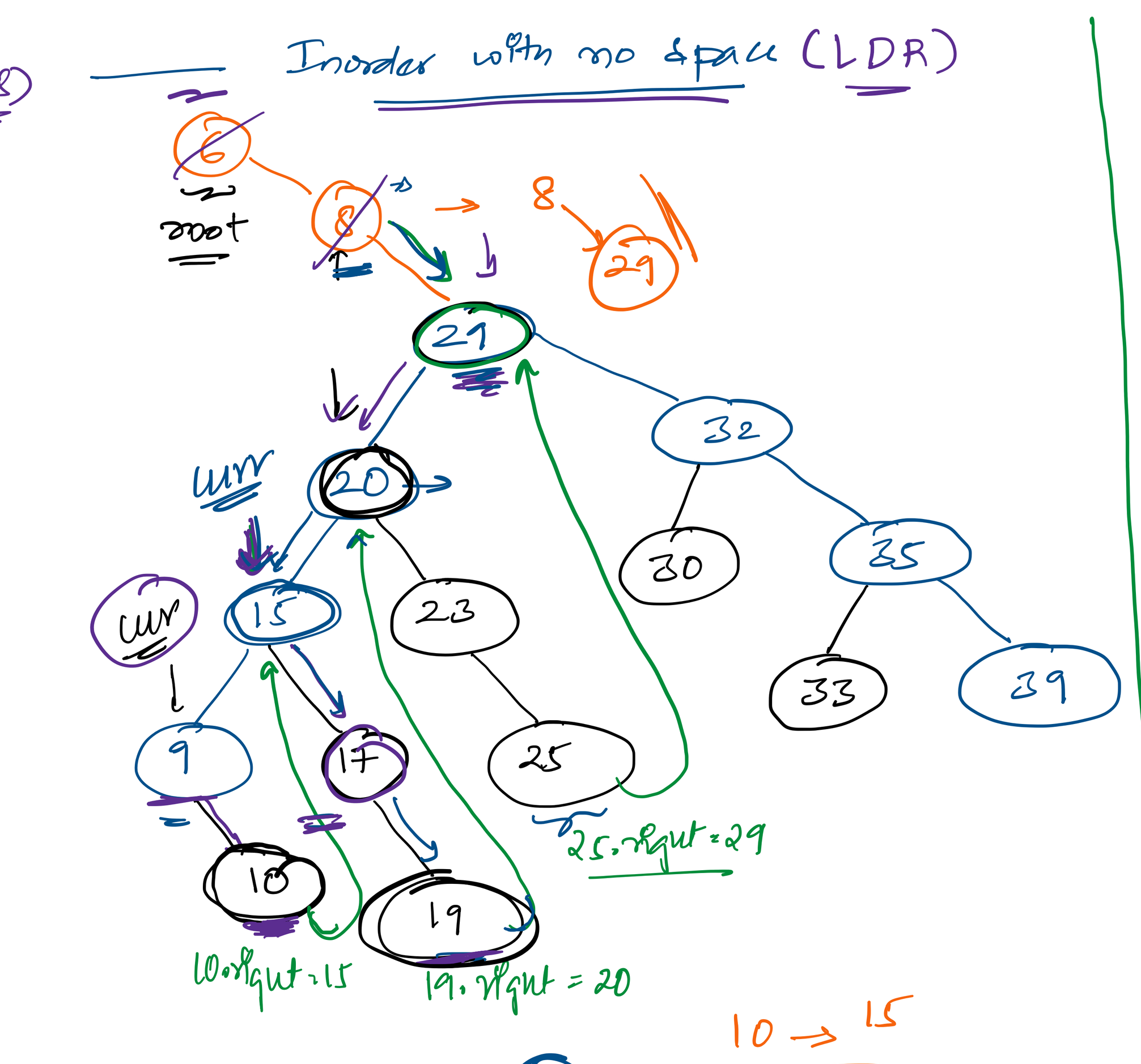
```
Node* inorderpred(Node *root){
    Node *temp = root;
    root = root->left // goto LST
    In LST, goto rightmost node
    while( root->right != NULL && root->right != temp){
        root = root->right
    }
    return root;
}
```

↳ We need to add this condition to get inorder predecessor for 2nd time

SC: O(1)

preorder: DLR

2(8) Inorder with no space (LDR)



print: 6 8 9 10 15

10 → 15

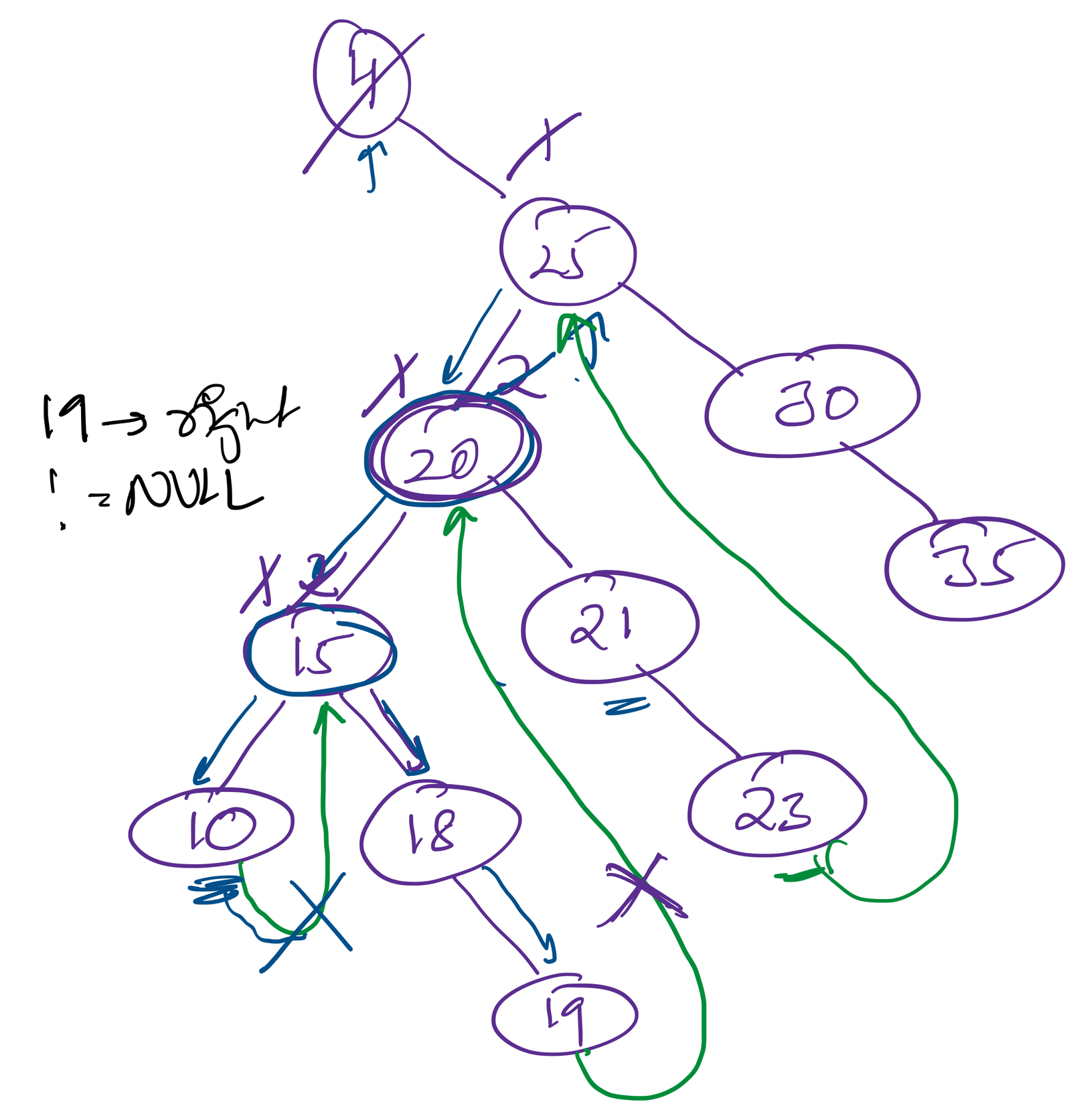
```
while( curr != NULL){
    if( curr->left == NULL){
        print(curr->data)
        curr = curr->right
    }
    else{
        Node *temp = inorderpred(curr)
        if(temp->right == NULL){
            temp->right = curr
            curr = curr->left;
        }
        else{
            temp->right = NULL
            print(curr->data)
            curr = curr->right
        }
    }
}
```

If predec of node is pointing == null visiting it for first

If predec of node is not pointing to null it is visited 2nd time → print(curr->data) & goto right

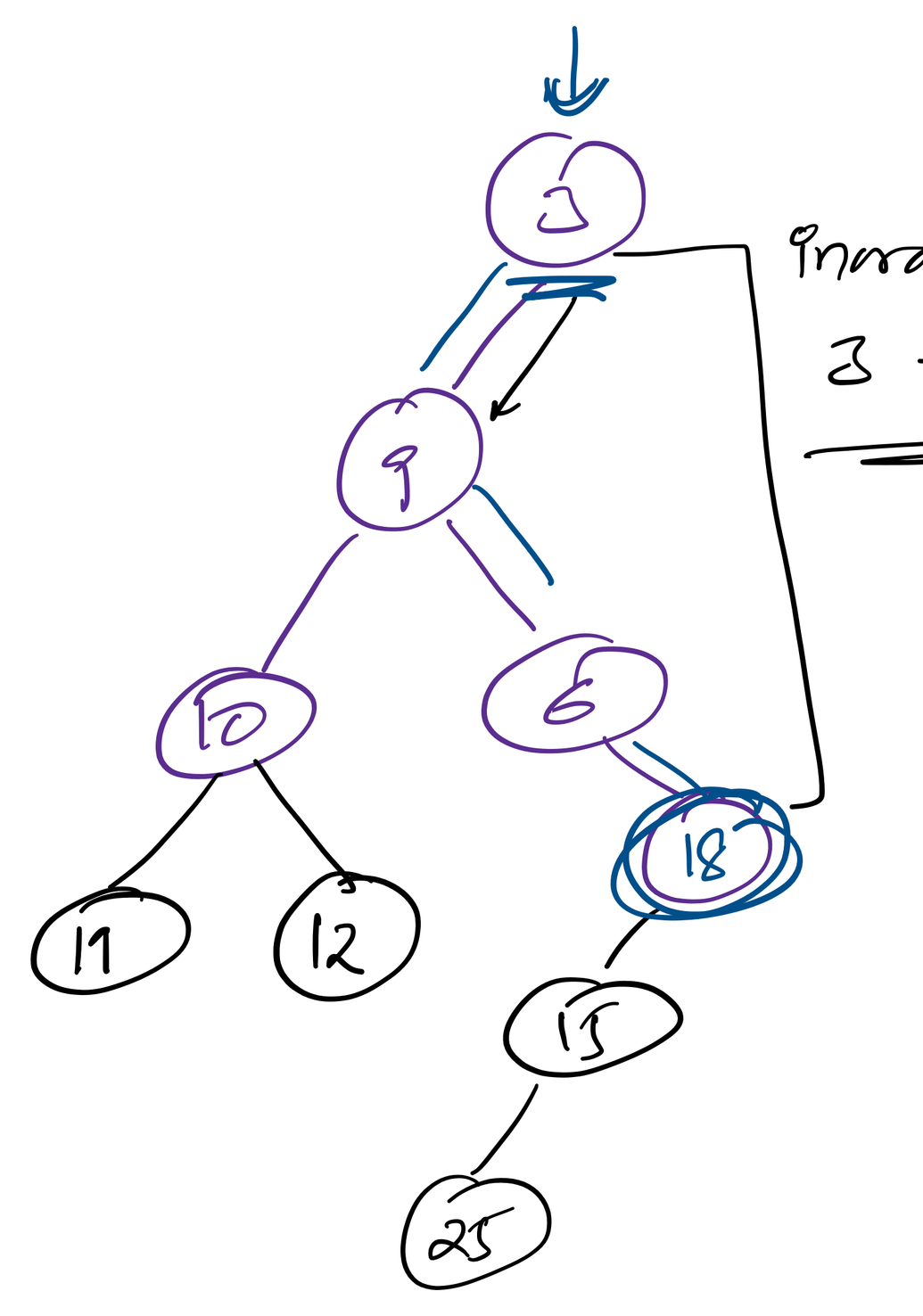
Note: If at all we need moris preorder, print() statement inorder keep it in if()

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



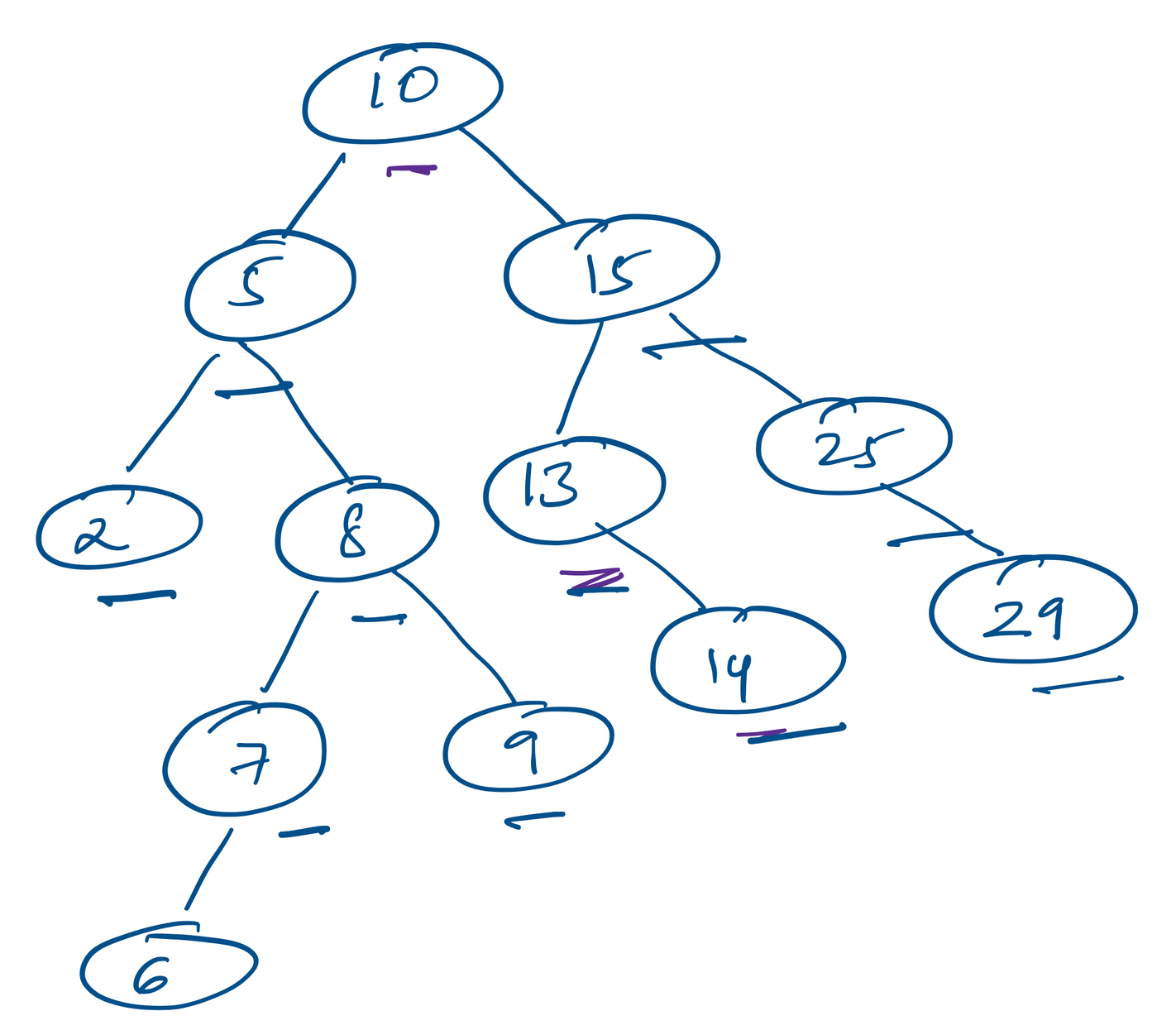
output:

4 10 15 18 19 20 21 →

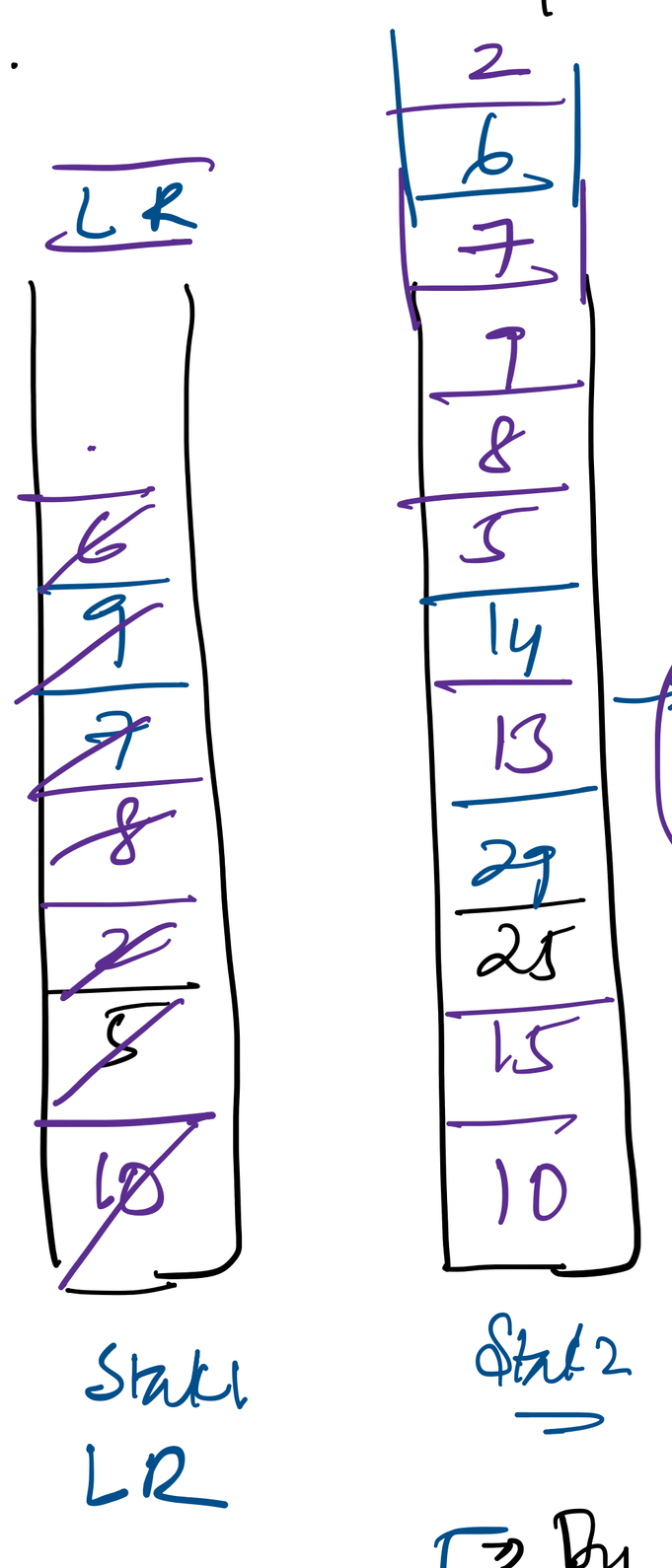


inorder predecesr of 3 = 18

3(8) Iterative postorder with 2 stacks.



postorder: LRD



2 6 7 9 8 5 14 13 29 25 15 10

LRD

final order

In stack 2 DRL

→ By popping data in stack 2 say we are getting postorder

Idea:

```
stack<Node*> s1, s2
s1.push(root)
while( s1.size() > 0){
    Node* temp = s1.top();
    s1.pop();
    s2.push(temp);
    if( temp->left != NULL)
        s1.push(temp->left)
    if( temp->right != NULL){
        s1.push(temp->right)
}
// print all elements in s2
```

Todo:

- ⇒ inorder iterative with stack
- ⇒ preorder iterat with stack
- ⇒ post iterat with 1stack

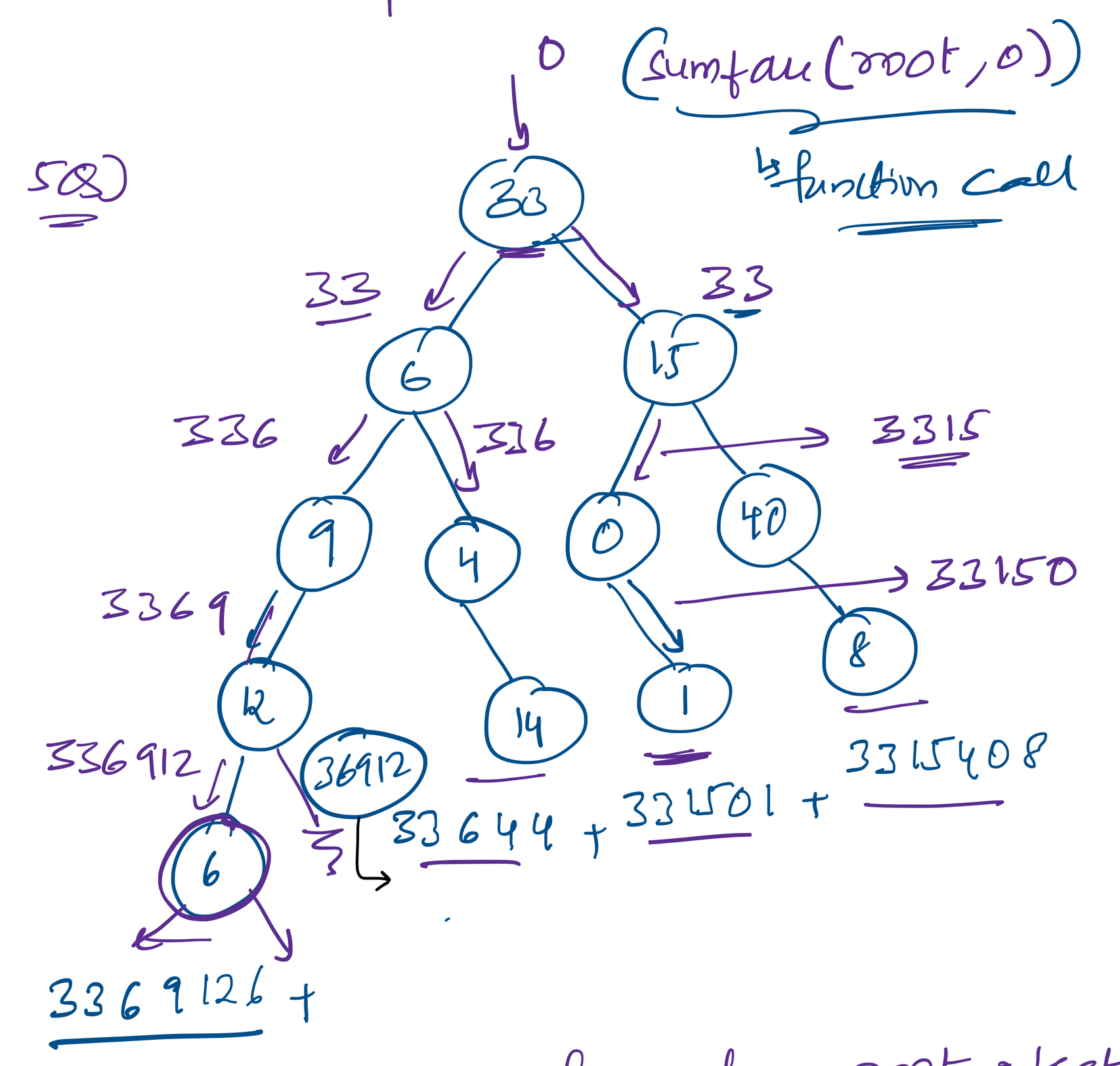
If sum = 234, root->data = 54 } 23454 → (2) 234*100+54

sum = 3000, root->data = 965 } 3000965 → 3000*1000 + 965

If sum = 29, root->data = 0 } 290 → 29*10+0

5(8)

0 (sumfall(root,0)) ↳ function call



33369126 + 33644 + 331501 + 3315408

⇒ Sum of all numbers formed from root → leaf node

output: It should be in int

↳ {thank for overflow, final output return % mod}

```
long ans;
void sumfall(Node *root, long sum){
    if(root == NULL) return;   } Dont add when at all;
    int d = count_digit(root->data)
    sum = (sum * power(10,d) + root->data)%mod
    if( root->left == NULL && root->right == NULL)
        { at leaf node
          ans = (ans + sum)%mod
          return; }
    sumfall(root->left, sum)
    sumfall(root->right, sum))
}
```

[Implement your own power function, with mod]

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

↳ > O(N) becaz of power function

```
int countdigit(int n){
    int c = 0;
    while(n > 0){
        c++;
        n = n/10
    }
    return c;
}
```

if(n==0) return 1; ↳ very very Edge