Plan for Week

Monday, 16 August 2021 10:32 PM

17th August 2021

1. ~~Recover BST~~
2. ~~Construct BST from level order~~
3. ~~Serialise and Deserialise N-ary Tree~~
4. ~~Serialise and Deserialise Binary Tree~~
5. ~~left View of Binary Tree~~
6. ~~Right View of Binary Tree~~

19th August 2021

1. width of Shadow of Binary Tree
2. Vertical order traversal of B-Tree
3. Vertical order traversal of B-Tree - II
4. Bottom View of a Binary Tree
5. Top View of a Binary Tree

21st August 2021 (Morning)

1. Diagonal order of a Binary Tree
2. Diagonal order of a Binary Tree (Anti-clockwise)
3. Vertical order sum of B Tree
4. Diagonal order sum of B-Tree
5. Node to root path in B-Tree.

21st August 2021 (Evening)

1. Iterator → Generic Tree from LL
   → BST Iterator - 2.
2. Inorder Morris Traversal
3. Preorder Morris Traversal
4. Post order Morris Traversal

24th August 2021

1. Root to all leaf path in B.T.
2. All Single child in B.T.
3. Count of single child parent
4. All nodes distance K in B.T.
5. Burning Tree

26th August 2021

1. Burning Tree 2
2. max-width of Binary Tree
3. Convert BST to Doubly LL
4. Convert Sorted DLL to BST
5. Path Sum in Binary Tree

28th August 2021 (Morning)

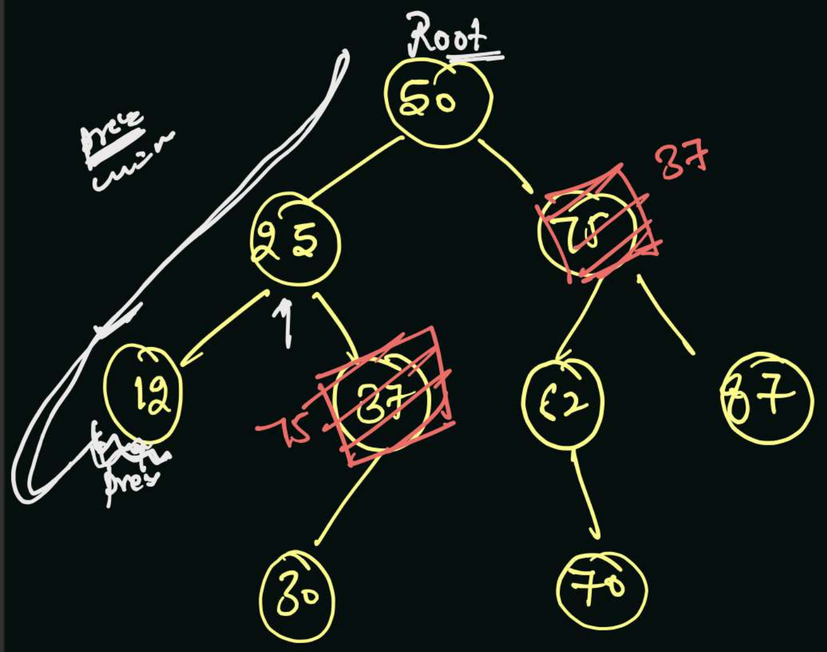
1. Path Sum in Binary Tree 2
2. Diameter of Binary Tree (All Methods)
3. Maximum path sum in B/w two leaf
4. Miscellaneous

28th August 2021 (Evening)

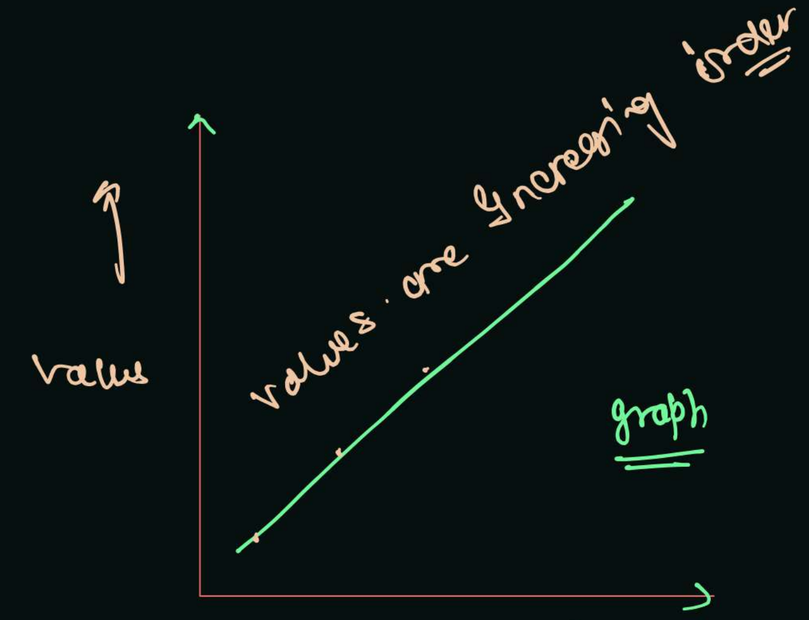
1. Maximum path sum of B-Tree
2. Path Sum Equal to given value
3. Lowest comon ancestor of B-Tree
4. Miscellaneous

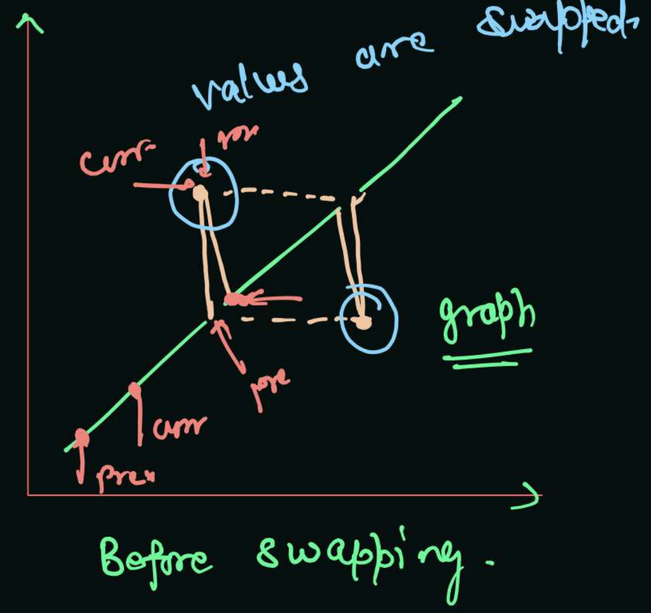
Recover BST

Tuesday, 17 August 2021 8:33 PM



Inorder (BST) → Sorted





Inorder → → 12 25 30 75 50 62 70 37 87

(After swapping)

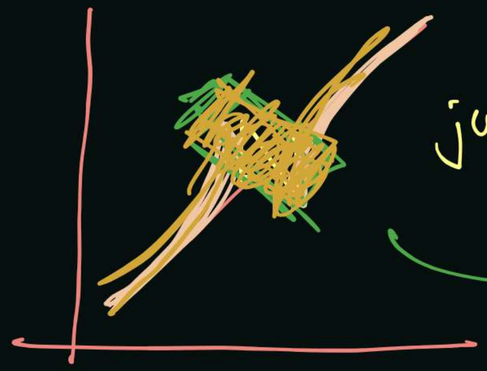
a b

First Encounter → Prev

Second Encounter → current

prev curr

a = null
b = null



just adjacent Elements are swapped.

→ problem will encounter only single time.

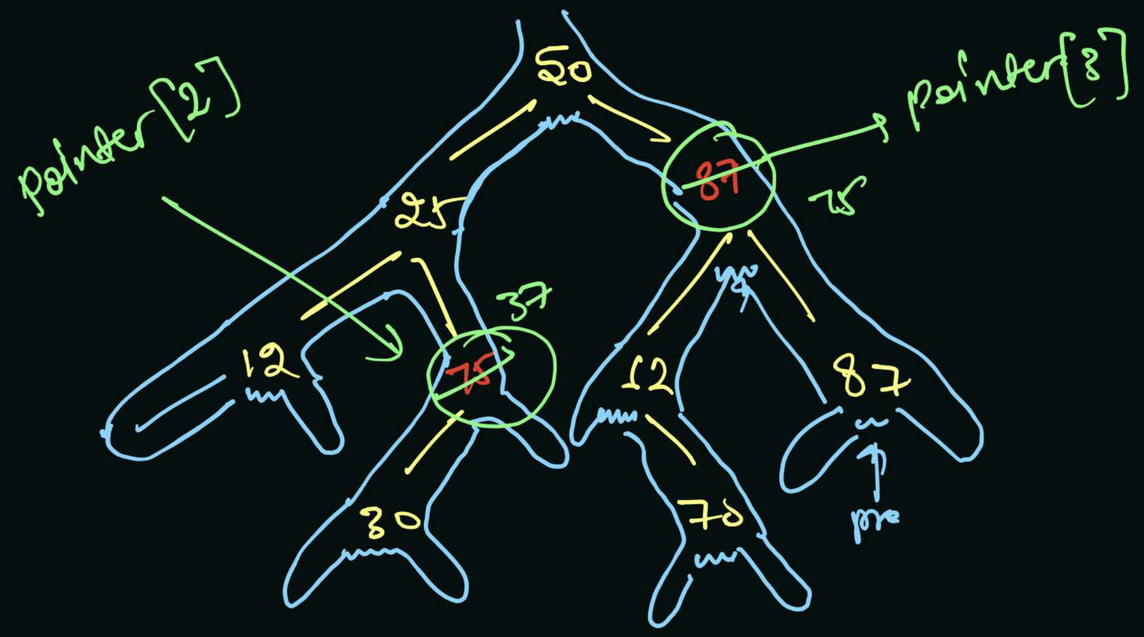
First Encounter, 62

Inorder → 12 30 37 62 60 70 75 87 60


```
if( curr < prev) {
    if( b == null) {
        // first occurance  → a = prev
                              b = curr
    } else {
        // second occurance
        b = curr
    }
}
```

How to manage prev and curr ??

→ Inorder traversal



pointers[0] = prev = null = ~~12~~ ~~25~~ ~~30~~ ~~75~~ ~~50~~ ~~12~~ ~~70~~ ~~37~~ 87

pointers[1] = curr = null = ~~25~~ ~~30~~ ~~75~~ ~~50~~ ~~12~~ ~~70~~ ~~37~~ 87

pointers[2] = a = null = 75

Pointers[3] = b = null = ~~50~~ 37

Data Swap?

In Real

```
// pointers[0] -> prev
// pointers[1] -> curr
// pointers[2] -> a
// pointers[3] -> b
public void recover_Tree(TreeNode root, TreeNode[] pointers) {
    if(root == null) return;

    recover_Tree(root.left, pointers);
    if(pointers[0] == null) {
        // prev == null
        pointers[0] = root;
    } else {
        pointers[1] = root;
        if(pointers[0].val > pointers[1].val) {
            // prev > curr
            if(pointers[3] == null) {
                // first encounter
                pointers[2] = pointers[0];
                pointers[3] = pointers[1];
            } else {
                // second encounter
                pointers[3] = root;
            }
        }
        // move prev and curr
        pointers[0] = root;
    }
    recover_Tree(root.right, pointers);
}

public void recoverTree(TreeNode root) {
    TreeNode[] pointers = new TreeNode[4];
    recover_Tree(root, pointers);
    // swap value for a and b, i.e. pointers[2], pointers[3]
    int temp = pointers[2].val;
    pointers[2].val = pointers[3].val;
    pointers[3].val = temp;
}
```

size = 1
int[] index

b == null ||

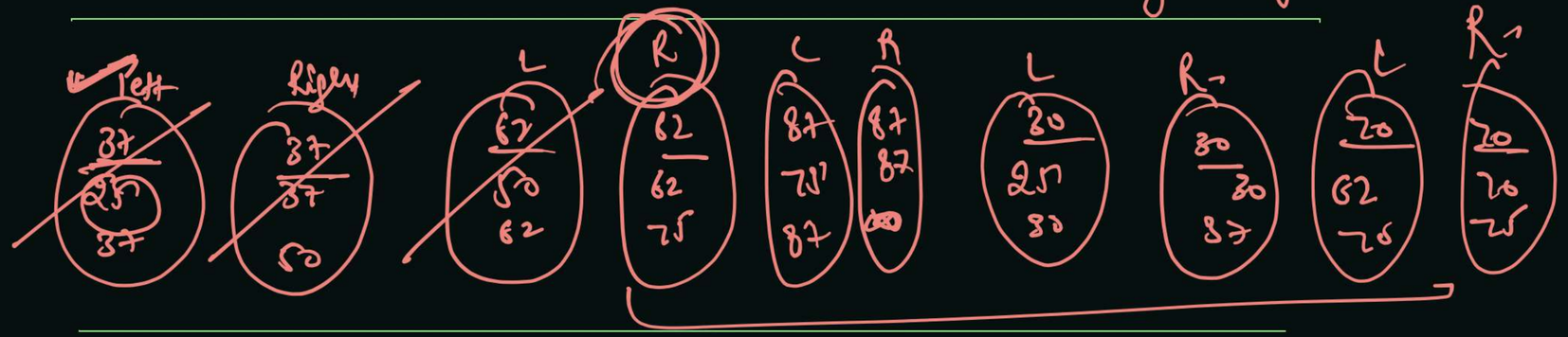
Construct BST From Level Order

Tuesday, 17 August 2021 9:26 PM

level order → 50, 25, 75, 12, 37, 62, 87, 30, 70

Class
→ parent
→ left Range
→ Right Range

Hint - D.S → queue



Parent
left R.
Right R.

-∞ ⇒ [12]

[12] ⇒ 25

for in in

while Nest in Range

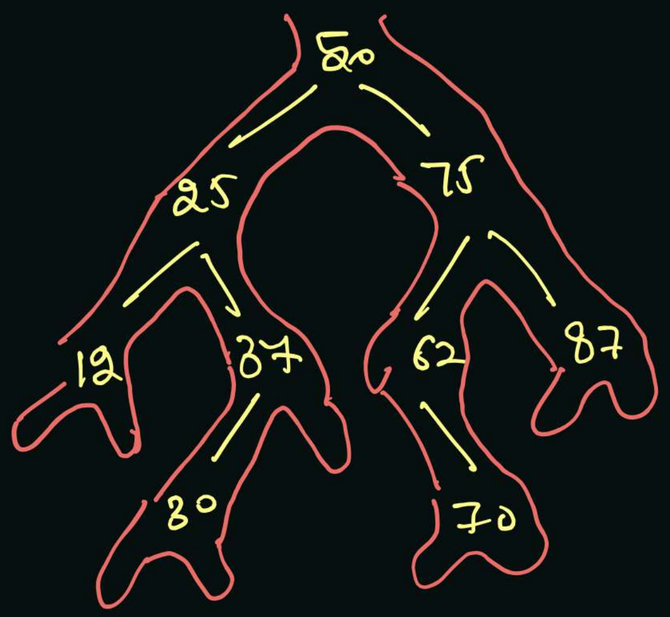
pop out from Que

-add condition if in Range
push left Range and Right Range

Serialize and Deserialize Binary Tree

Tuesday, 17 August 2021 10:28 PM

Data → String } Encription of Binary tree so that we can decript it and we can make same Binary tree again.



Serialise-

StringBuilder →

Convert this into String.

"50 # 25 # 12 # null # null # 37 # 80 # null # null # null # 75 # 62 # null # 70 # null # null # 87 # null # null #

PreOrder String-

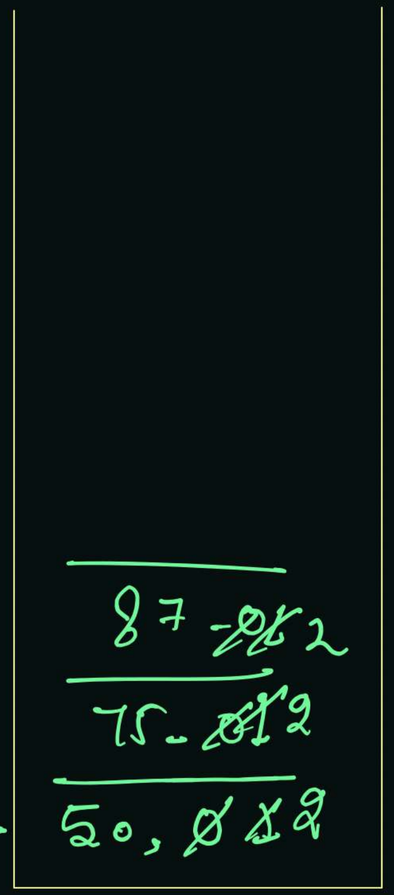
Empty String

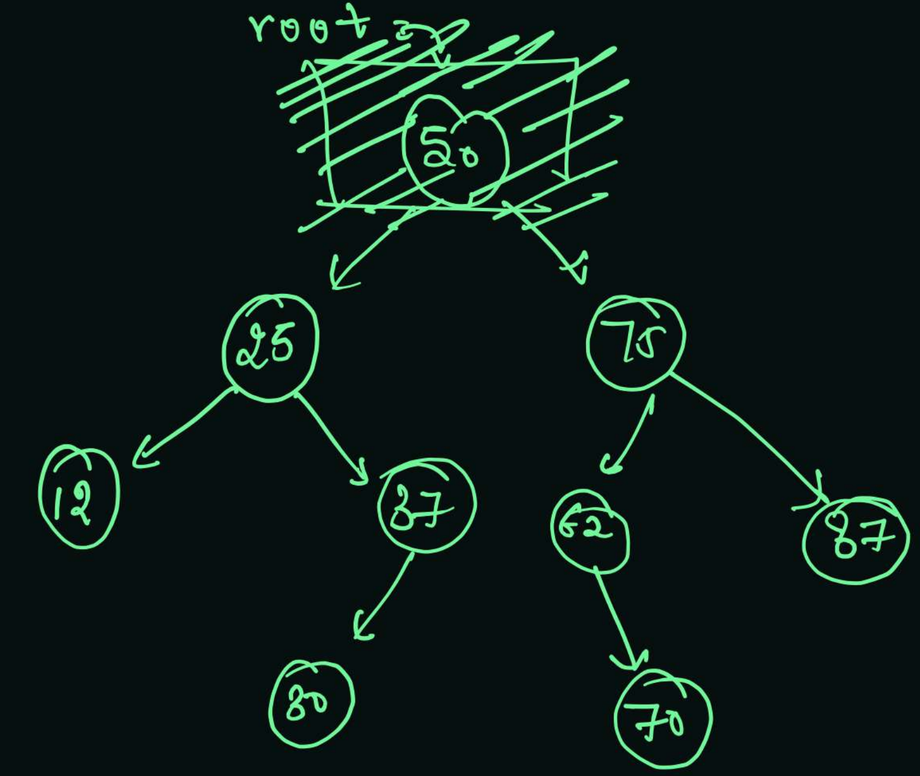
Steps for Desorialise

1. Split on the basis of delimeter.
2. treat string as Integer using parseInt Except null.
3. Construct Binary tree

data → 50 25 12 null null 87 30 null null null 75 62 null 70 null

null 87 null null →



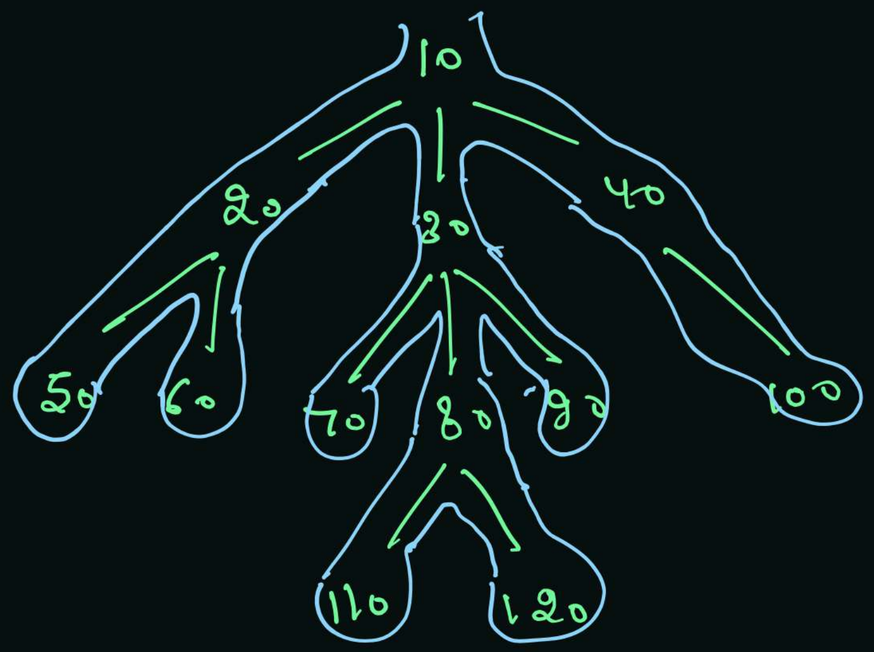



```
while( indx < arr.length) {
    if(st.peek().state == 0){
        // node, left ans
        // if null
        // state++
    } else if (state == 1) {

    } else {
        // st.pop()
    }
}
```

Serialize and Deserialize N-ary Tree

Tuesday, 17 August 2021 10:28 PM



Serialise

→ pre Area → node.data + '#'

→ post Area → "null #"

10 # 20 # 50 # null # 60 # null # null # 30 # 70 #

null # 80 # 110 # null # 120 # null # null #

90 # null # null # 40 # 100 # null # null # null #

data in String

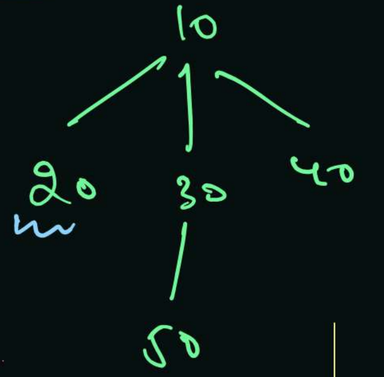
Deserialise

① split on basis of delimiter: "#"

② use Integer.parse int and solve, also merge null

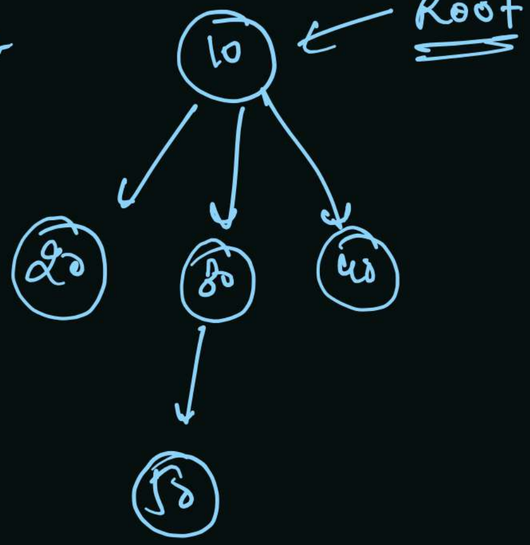
③ Return root.

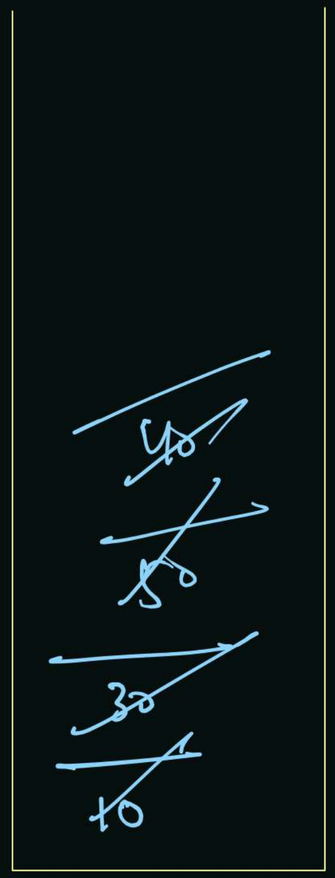
Generic Tree-



10 20 null 30 50 null null 40 null null

Back have no children





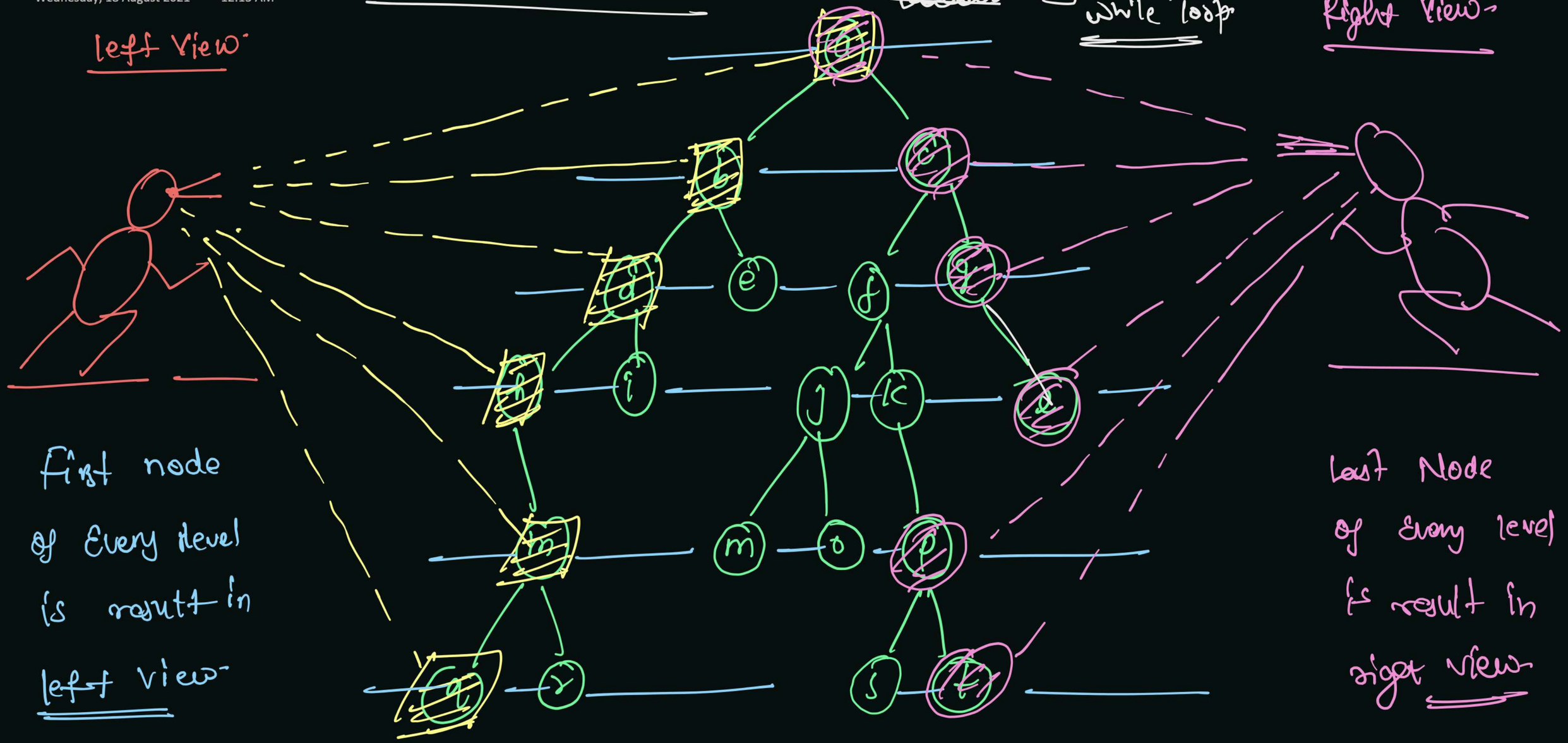
Left View and Right View

Wednesday, 18 August 2021 12:13 AM

level order traversal ⟶ 2 while] complete level present in while loop

Left View

Right View



First node of Every level is result in left view

Last Node of Every level is result in right view