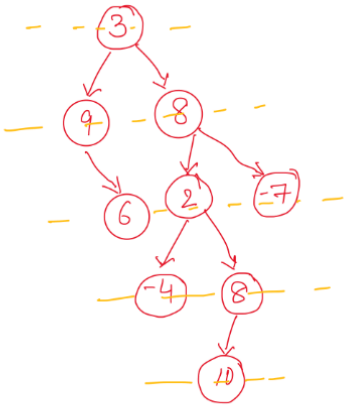
Trees 2

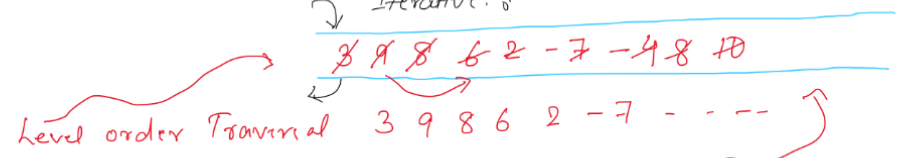
Level Order Traversal (R - L)
Left View And Right View
Vertical Order Traversal
Top And Bottom View
Types Of Binary Tree
Height Balanced Tree

Level Order Traversal (R - L)



Expected o/p → 3 9 8 6 2 -7 -4 8 10
Left to Right.

Idea :
Recursion will not work as all subtree cannot be same.
So
Iterative. or



Level order Traversal 3 9 8 6 2 -7 - - - -

Queue (FIFO)

Pseudo Code :-

```
Queue <Node> q;
q.enqueue(root);
while(!q.empty()){
    Node f = q.front();
    q.dequeue();
    Print(f.data());
    if(q.left != NULL)
        q.push(f.left)
    if(q.right != NULL)
        q.push(f.right);
}
```

Normal Inorder

T : O(N)
S : O(N)

```
while(q.size() >1)
    q.push(NULL)
if(f != NULL)
{
    Print("\n");
    q.push(NULL);
}
else{
    Print(f.data);
}
```

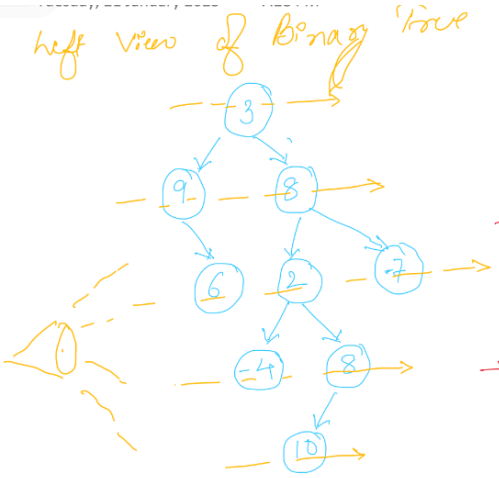
→ To print inorder line by line.

3 NULL 9 8 NULL

Inorder in different line

R - L → first Push Right than left Inside loop

Left View And Right View

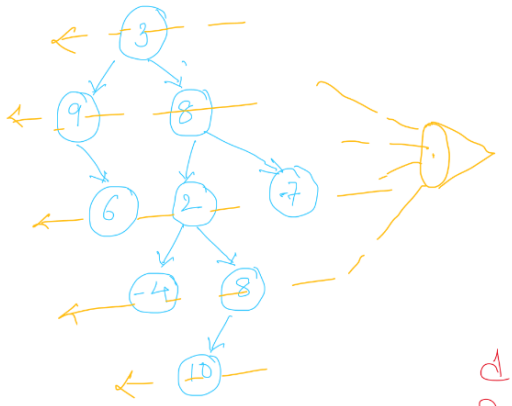


Left View: 3 9 6 -4 10

→ Start level order Traversal and in every iteration select the first element to get left view

→ For all first element in each level element previous to that, will be NULL except Root element. (prev == NULL)

copy: root Node



3 8 -7 8 10

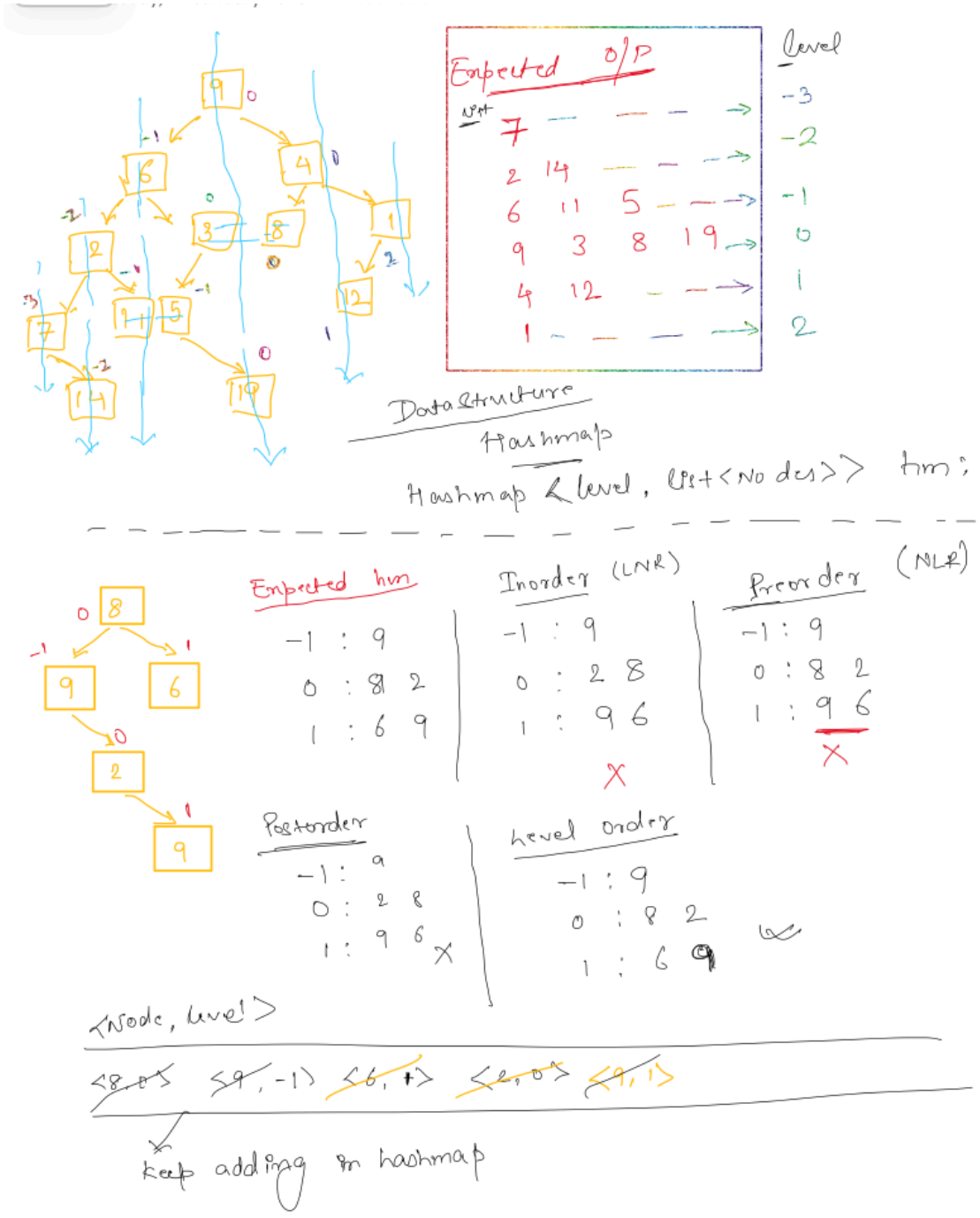
→ Start level order Traversal Last node of every level will be Right view of B-Tree.

→ For R.V node in each level the next will be NULL.

OR

Do Level order Traversal from Right to left, & follow approach for left view.

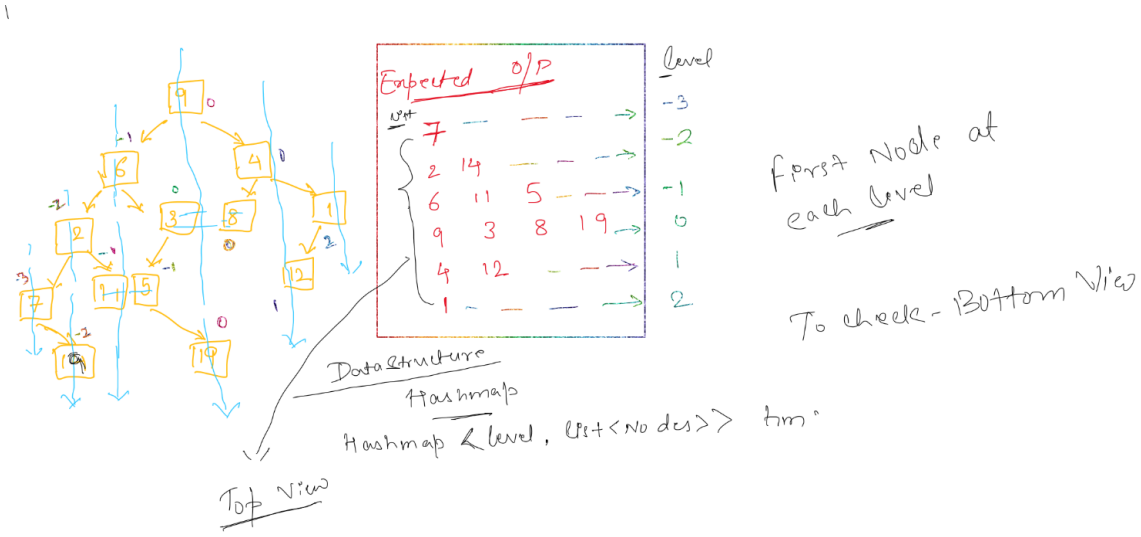
Vertical Order Traversal



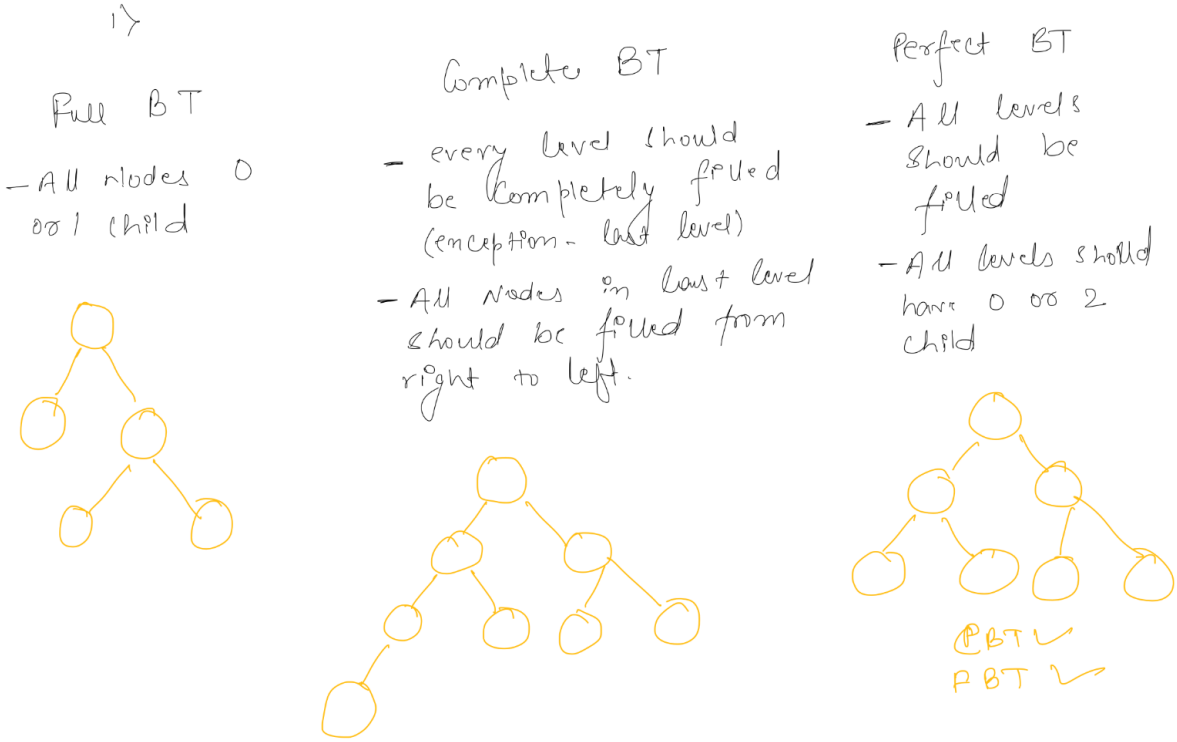
Pseudo Code:

```
Queue <Node, Int> q
Hashmap <int, list<Node>> hm
q.push ({root, 0});
while(!q.empty())
{   Pair <Node, int> p = q.front
    Node f = p.first;
    int l = p.second;
    q.dequeueue();
    hm[l].add(f); //Insert Node f at level l.
    //Add left & right child in the queue.
      if(f.left != NULL)
          q.push({f.left, l-1});
      if(f.right != NULL)
          q.push({f.right, l+1});
}
```

Top And Bottom View



Types Of Binary Tree



Height Balanced Tree

```
int height (Node root)
{   if (root == NULL)
        return 0;
    int l = height (root.left)
    int r = height (root.right)
    return max(l, r) + 1;
}
```

Height of a Tree

Q Check whether a given tree is height balanced or not ?

$|h(LST) - h(RST)| <= 1$

eg:


```
bool ans = true;

bool isBalanced (Node root)
{   height(root)
    return ans;
}
```

```
int height (Node root)
{   if (root == NULL)
        return 0;
    int l = height (root.left)
    int r = height (root.right)
    if (abs(l-r) > 1) {
        ans = false;
    }
    return max(l, r) + 1;
}
```