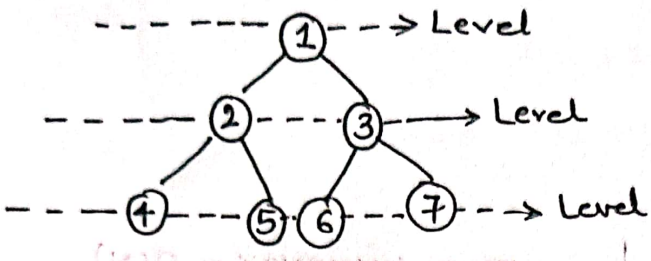
TREE

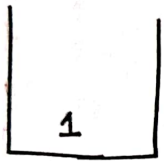
(BFS)

Level Order Traversal of Binary Tree



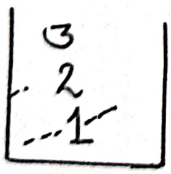
1
2 3
4 5 6 7

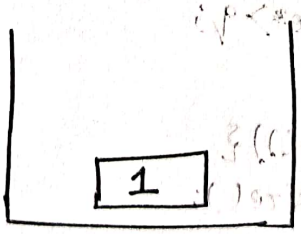
We initially require a Queue Data Structure



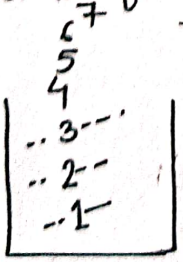
→ In order to store the traversal level wise, we will be requiring some data structure, vector of vector which store level of traversal level wise.

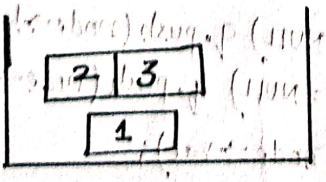
→ When you remove 1 from the Data Structure make sure you add both its left and right child into the Queue.

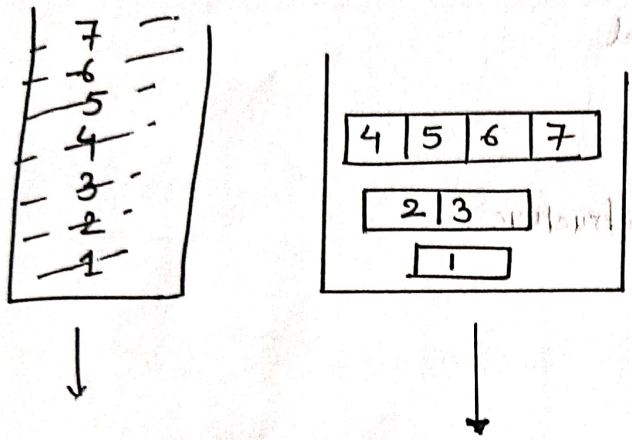




→ Whatever is left in your Queue, just remove and put it in the Queue







now queue data structure is empty.

This data structure now stored level order traversal.

TIME COMPLEXITY = O(N)

SPACE COMPLEXITY = O(N)

Because in worst case you have to store all the element into the Queue.

we dont consider ds → O(N) because you ultimately have to return the answer.

C++ Code:-

```
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>> ans;
        if(root == NULL) return ans;
        queue<TreeNode*> q;
        q.push(root);
        while(!q.empty()) {
            int size = q.size();
            vector<int> level;
            for(int i=0; i<size; i++) {
                TreeNode *node = q.front();
                q.pop();
                if(node->left != NULL) q.push(node->left);
                if(node->right != NULL) q.push(node->right);
                level.push_back(node->val);
            }
            ans.push_back(level);
        }
        return ans;
    }
};
```