Vertical Order Traversal (Level Order / BFS)

Stepwise Algorithm Explanation: 1. **Problem Understanding:**
Vertical Order Traversal groups nodes by their horizontal distance from the root. Using BFS (level order) ensures that nodes are processed top-to-bottom in each vertical line. 2. **Idea:**
- Maintain a queue of pairs: (node, position). - Root starts at pos = 0. - Left child gets pos-1, right child gets pos+1. - Push data of each node into ans[pos]. 3. **Width Calculation:**
- First calculate the leftmost (Lmin) and rightmost (Rmax) horizontal positions using a recursive function. - This tells us how many vertical lines are required. 4. **BFS Traversal:**
- Use a queue to process nodes level by level. - For each node, push its data into the appropriate vertical index. - Push its left and right children with updated positions. 5. **Offset Handling:**
- Since positions can be negative, we use an offset = abs(Lmin). - Example: if Lmin = -3, we add +3 to all positions so they fit into a 0-based array index. 6. **Return:**
- After BFS traversal, return the 2D vector containing grouped vertical nodes.

Code Implementation:

```
void vertical(Node * root , vector<vector<int>> &ans , int pos){
    queue<pair<Node*, int>> q;
    q.push({root, pos});

    while(!q.empty()){
        Node *temp = q.front().first;
        int pos = q.front().second;
        q.pop();

        ans[pos].push_back(temp->data);
        if(temp->left) q.push({temp->left, pos-1});
        if(temp->right) q.push({temp->right, pos+1});
    }
}

void width(Node* root , int &Lmin , int &Rmax , int pos){
    if(!root) return;
    width(root->left , Lmin , Rmax , pos-1);
    Lmin = min(Lmin , pos);
    width(root->right , Lmin , Rmax , pos+1);
    Rmax = max(Rmax , pos);
}

class Solution {
  public:
    vector<vector<int>> verticalOrder(Node *root) {
        int Lmin = 0, Rmax = 0;
        width(root, Lmin, Rmax, 0);
        int size = (Rmax - Lmin) + 1;
        vector<vector<int>> result(size);
        vertical(root, result, abs(Lmin));
        return result;
    }
};
```