**Vertical Order Traversal**

vector<int> verticalOrderTraversal(TreeNode<int> \*root)

{

    vector<int> result;

    if (root == NULL)

        return result;

    // Create a map to store the vertical order traversal

    map<int, vector<int>> verticalMap;

    // Create a queue for level order traversal

    queue<pair<TreeNode<int> \*, int>> q;

    // Enqueue the root node with its horizontal distance

    q.push({root, 0});

    while (!q.empty())

    {

        TreeNode<int> \*curr = q.front().first;

        int currHD = q.front().second;

        q.pop();

        // Add the current node's data to the map

        verticalMap[currHD].push\_back(curr->data);

        // Enqueue the left child with a decreased horizontal distance

        if (curr->left != NULL)

            q.push({curr->left, currHD - 1});

        // Enqueue the right child with an increased horizontal distance

        if (curr->right != NULL)

            q.push({curr->right, currHD + 1});

    }

    // Traverse the map and collect the vertical order traversal

    for (auto it = verticalMap.begin(); it != verticalMap.end(); ++it)

    {

        vector<int> verticalNodes = it->second;

        for (int i = 0; i < verticalNodes.size(); i++)

        {

            result.push\_back(verticalNodes[i]);

        }

    }

    return result;

}