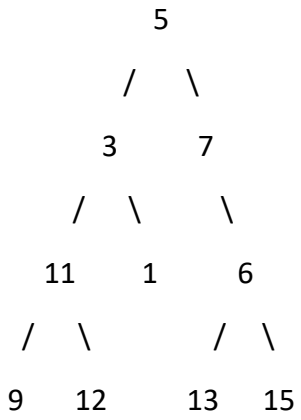
# Vertical Order Traversal Of Binary Tree GFG

Given a Binary Tree, find the vertical traversal of it starting from the leftmost level to the rightmost level.
If there are multiple nodes passing through a vertical line, then they should be printed as they appear in **level order** traversal of the tree.

Example:




Output: The Vertical Traversal of Binary Tree: 9, 11, 3, 12, 5, 1, 7, 13, 6, 15

**Approach 1: Function to perform vertical order traversal of a binary tree using a recursive approach.**

- The recursive approach performs vertical order traversal by maintaining a map that associates each node with its horizontal distance from the root and level.
- It starts the traversal from the root and recursively explores the left and right subtrees.
- Nodes are inserted into the map based on their horizontal distance and level.
- The result is obtained by extracting nodes from the map and adding them to a result vector.

**Time Complexity:**

- **The time complexity is O(N log N) in the worst case, where N is the number of nodes in the tree. This is due to the use of a map for sorting nodes by horizontal distance and level.**

**Space Complexity:**

- **The space complexity is O(N) in the worst case, where N is the number of nodes in the tree, due to the storage of nodes in the map.**

**Approach 2: Function to perform vertical order traversal of a binary tree using an iterative approach.**

- The iterative approach performs vertical order traversal using a queue for level-order traversal.
- It also maintains a map to associate nodes with their horizontal distance and level.
- Starting from the root, it enqueues nodes with their horizontal distance and level.
- Nodes are dequeued, and their values are inserted into the map.
- Finally, nodes are extracted from the map and added to a result vector.

**Time Complexity:**

- **The time complexity is O(N log N) in the worst case, where N is the number of nodes in the tree. This is due to the use of a map for sorting nodes by horizontal distance and level.**

**Space Complexity:**

- **The space complexity is O(N) in the worst case, where N is the number of nodes in the tree, due to the storage of nodes in the map.**

**Conclusion:**

Both the recursive and iterative approaches effectively perform vertical order traversal of a binary tree. They share the same time complexity of O(N log N) due to the sorting by horizontal distance and level. However, their space complexities are also O(N), making them suitable for relatively small trees.

In terms of memory efficiency, both approaches have similar space complexities, as they use maps to store nodes. The choice between the recursive and iterative approaches depends on coding preferences and requirements.