Vertical Order Traversal Method 2

```python
import sys


class TreeNode:
    def __init__(self, val):
        self.data = val
        self.left = None
        self.right = None


class Pair:
    def __init__(self, level, node):
        self.level = level
        self.node = node


def printVerticalOrder(root):
    if root is None:
        return []
    result = [sys.maxsize, -sys.maxsize]
    findWidth(root, 0, result)
    length = result[1] - result[0] + 1
    ans = [None] * length
    queue = [Pair(abs(result[0]), root)]
    while len(queue):
        temp = queue.pop(0)
        if ans[temp.level] is None:
            ans[temp.level] = [temp.node.data]
        else:
            ans[temp.level].append(temp.node.data)
        if temp.node.left:
            queue.append(Pair(temp.level - 1, temp.node.left))
        if temp.node.right:
            queue.append(Pair(temp.level + 1, temp.node.right))
    return ans


def findWidth(root, level, result):
    if root is None:
        return None
    result[0] = min(level, result[0])
    result[1] = max(level, result[1])
    findWidth(root.left, level - 1, result)
    findWidth(root.right, level + 1, result)


root = TreeNode(1)
root.left = TreeNode(2)
root.right = TreeNode(3)
root.left.left = TreeNode(4)
root.left.right = TreeNode(5)
root.right.left = TreeNode(6)
root.right.right = TreeNode(7)

print(printVerticalOrder(root))
```