**Problem Title**

Sophias Hierarchy Visualization

**Problem Description**

Sophia wants to visualize her organization's hierarchy. Each level of the hierarchy should be displayed on a new line. She needs to perform a level order traversal of the binary tree representing the hierarchy.

Given a binary tree, print the level order traversal of its nodes' values. (i.e., from left to right, level by level).

**Input Format**

* A single line contains the values of all the nodes in the binary tree in a pre-order format where true suggests the node exists and false suggests it is NULL.

**Output Format**

* Print the tree level order traversal, with each level on a new line.

**Constraints**

* The number of nodes in the tree is in the range [0, 2000].
* -1000 <= Node.val <= 1000

**Examples**

**Example 1**

Input:

3 true 9 false false true 20 true 15 false false true 7 false false

Output:

3

9 20

15 7

**Explanation**

**Example 1:** The binary tree is:

3

/ \

9 20

/ \

15 7

Level 1 has 3, Level 2 has 9 and 20, and Level 3 has 15 and 7.

**Example 2**

Input:

30 true 90 false false true 20 true 15 false false true 70 false false

Output:

30

90 20

15 70

**Example 2:** The binary tree is:

30

/ \

90 20

/ \

15 70

Level 1 has 30, Level 2 has 90 and 20, and Level 3 has 15 and 70.

**Test Case 1:**

Input:

8 true 6 true 5 false true 7 false false true 10 false true 9 false false true 12 false false

Output:

8

6 12

5 10

7 9

**Test Case 2:**

Input:

8 true 10 true 6 false false true 7 false false true 5 false true 3 false false

Output:

 8

10 5

6 7 3

**Extra Test Cases:**

Input:

 8 true 6 true 5 false true 7 false false true 10 false true 9 false false true 12 false false

Output:

8

6 12

5 10

7 9

Input:

 8 true 10 true 6 false false true 7 false false true 5 false true 3 false false

Output:

 8

10 5

6 7 3

Input:

12 true 8 true 5 true 3 false false true 7 false false true 10 true 9 false false true 11 false false true 15 false true 14 false false true 17 false false

Output:

12

8 15

5 10 14

3 7 9 11

Input:

18 true 15 true 10 true 5 false false true 12 false false true 17 true 16 false false true 20 true 19 false false true 25 false false

Output:

 18

15

10 17

5 12 16 20

19 25

Input:

25 true 20 true 15 true 10 false false true 22 false true 21 false false true 30 true 28 false false true 35 false false

Output:

25

20

15 30

10 22 28 35

21

**Solution**

To solve this problem, we will first parse the input to construct the binary tree and then perform a level order traversal to get the desired output.

Here's the implementation in Python:

python

Copy code

from collections import deque

class TreeNode:

def \_\_init\_\_(self, value=0, left=None, right=None):

self.value = value

self.left = left

self.right = right

def build\_tree(values):

if not values:

return None

def build\_tree\_helper(index):

if index >= len(values) or values[index] == 'false':

return None, index

node = TreeNode(int(values[index]))

index += 1

if index < len(values) and values[index] == 'true':

node.left, index = build\_tree\_helper(index + 1)

else:

index += 1

if index < len(values) and values[index] == 'true':

node.right, index = build\_tree\_helper(index + 1)

else:

index += 1

return node, index

root, \_ = build\_tree\_helper(0)

return root

def level\_order\_traversal(root):

if not root:

return []

result = []

queue = deque([root])

while queue:

level\_size = len(queue)

level\_nodes = []

for \_ in range(level\_size):

node = queue.popleft()

level\_nodes.append(node.value)

if node.left:

queue.append(node.left)

if node.right:

queue.append(node.right)

result.append(level\_nodes)

return result

def main():

import sys

input = sys.stdin.read().strip()

values = input.split()

root = build\_tree(values)

levels = level\_order\_traversal(root)

for level in levels:

print(" ".join(map(str, level)))

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

import sys

from io import StringIO

sys.stdin = StringIO("3 true 9 false false true 20 true 15 false false true 7 false false")

main() # Output:

# 3

# 9 20

# 15 7

**Explanation of the Code**

1. **TreeNode Class:** Defines the structure of a node in the binary tree.
2. **build\_tree Function:** Recursively constructs the binary tree from the pre-order input values.
3. **level\_order\_traversal Function:** Performs a level order traversal (BFS) on the binary tree and returns the values level by level.
4. **main Function:** Reads input, constructs the tree, and prints the level order traversal.