Level Order Traversal

Created	@July 30, 2022 1:14 PM
Tags	Tree Traversal breath first search
Name:	Jayaprakash

Level Order Traversal

Trees can also be traversed in level order, where we visit every node on a level before going to a lower level. This search is referred to as level order traversal or BFS ,as the search tree is broadened as much as possible on each depth before going to the next depth.

Algorithm:

It’s same as a normal breath first search

1. Initialize a queue and append the root element of the tree from left to right.
2. while the queue is not empty ,initialize a empty list callled $level$ and traverse through the queue using a for loop from 0 to length of the queue and pop the leftmost element .
3. Append the value of leftmost element to the list $level$ and append the left and right child nodes of the root node (or the leftmost element present in the queue).
4. Continue this process untill the queue gets empty, If the level list is not empty append that to the final result list and return .

```
from collections import deque


class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.right = right
        self.left = left
```

```
class Solution:
    def levelOrder(self, root):

        res = []
        queue = deque()
        queue.append(root)

        while queue:
            level = []
            lenQ = len(queue)
            for i in range(lenQ):
                node = queue.popleft()
                if node:
                    level.append(node.val)
                    queue.append(node.left)
                    queue.append(node.right)
            if level:
                res.append(level)
        return res


root1 = TreeNode(1)
root1.left = TreeNode(2)
root1.right = TreeNode(3)

sln = Solution()
print(sln.levelOrder(root1))
```