Your Day 23: BST Level-Order Traversal submission got 30.00 points. Share Tweet
Try the Next Challenge | Try a Random Challenge

Day 23: BST Level-Order Traversal

by vatsalchanana

Problem Submissions Leaderboard Discussions Editorial Tutorial

Objective
Today, we're going further with Binary Search Trees. Check out the Tutorial tab for learning materials and an instructional video!

Task
A level-order traversal, also known as a breadth-first search, visits each level of a tree's nodes from left to right, top to bottom. You are given a pointer, $root$, pointing to the root of a binary search tree. Complete the *levelOrder* function provided in your editor so that it prints the level-order traversal of the binary search tree.

Hint: You'll find a queue helpful in completing this challenge.

Input Format

The locked stub code in your editor reads the following inputs and assembles them into a BST:
The first line contains an integer, T (the number of test cases).
The T subsequent lines each contain an integer, $data$, denoting the value of an element that must be added to the BST.

Output Format

Print the $data$ value of each node in the tree's level-order traversal as a single line of N space-separated integers.

Sample Input

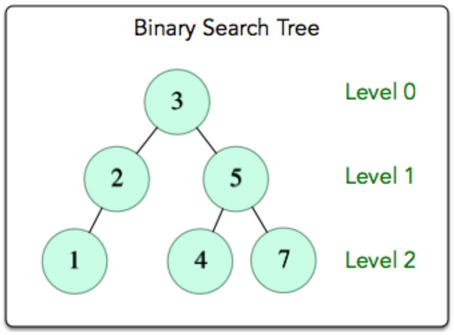
```
6
3
5
4
7
2
1
```

Sample Output

```
3 2 5 1 4 7
```

Explanation

The input forms the following binary search tree:



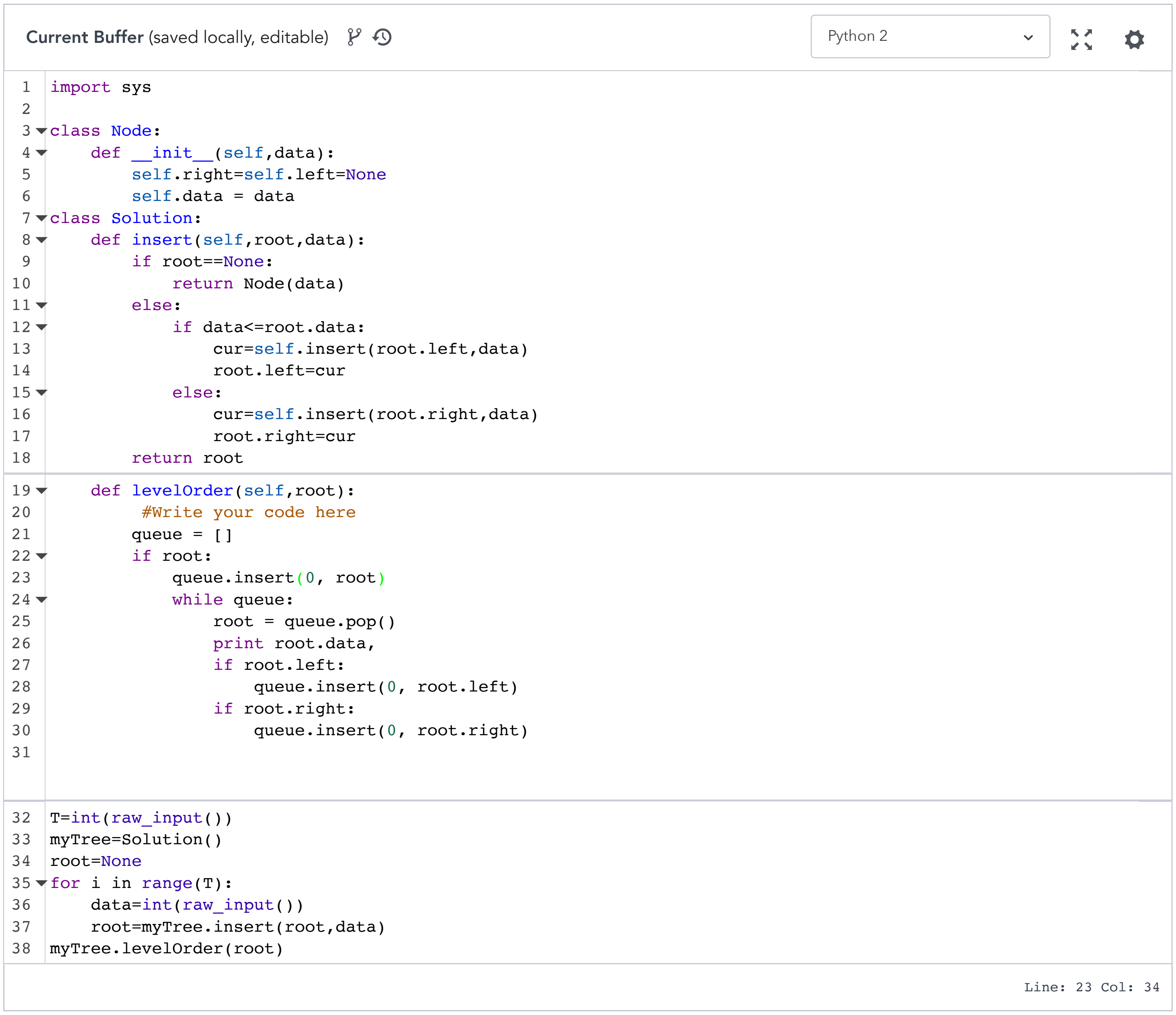
We traverse each level of the tree from the root downward, and we process the nodes at each level from left to right. The resulting level-order traversal is $3 \to 2 \to 5 \to 1 \to 4 \to 7$, and we print these data values as a single line of space-separated integers.

Submissions: 3828

Max Score: 30

Difficulty: Easy

More



```
import sys

class Node:
    def __init__(self,data):
        self.right=self.left=None
        self.data = data
class Solution:
    def insert(self,root,data):
        if root==None:
            return Node(data)
        else:
            if data<=root.data:
                cur=self.insert(root.left,data)
                root.left=cur
            else:
                cur=self.insert(root.right,data)
                root.right=cur
        return root
    def levelOrder(self,root):
         #Write your code here
        queue = []
        if root:
            queue.insert(0, root)
            while queue:
                root = queue.pop()
                print root.data,
                if root.left:
                    queue.insert(0, root.left)
                if root.right:
                    queue.insert(0, root.right)


T=int(raw_input())
myTree=Solution()
root=None
for i in range(T):
    data=int(raw_input())
    root=myTree.insert(root,data)
myTree.levelOrder(root)
```

Line: 23 Col: 34

Upload Code as File ☐ Test against custom input Run Code Submit Code

Congrats, you solved this challenge!

✓ Test Case #0 ✓ Test Case #1 ✓ Test Case #2

Next Challenge

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature