Practice □ Tutorials □ 30 Days of Code □ Day 23: BST Level-Order Traversal

COMPETE

JOBS

LEADERBOARD

6 more challenges to get your gold badge!
Days of
Code

25% 24/30

joel_h_healy

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, , 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:

Submitted 32697 times
Max Score 30

Need Help?

View Tutorial

View Discussions

The first line contains an integer, (the number of test cases).
The subsequent lines each contain an integer, , denoting the value of an element that must be added to the BST.

Output Format

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

Sample Input

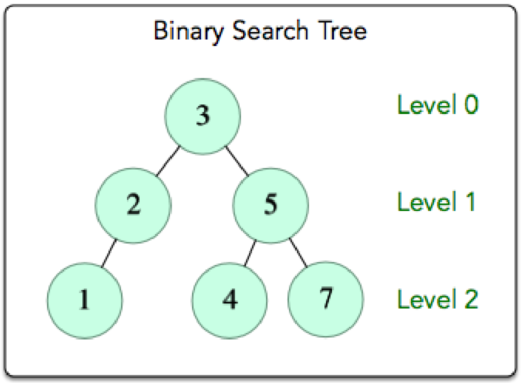
```
6
3
5
4
7
2
1
```

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 , and we print these data values as a single line of space-separated integers.

View Editorial Solution

View Top Submissions

Rate This Challenge:

□ □ □ □ □

Download problem statement

Download sample test cases

Suggest Edits

Current Buffer (saved locally, editable)

Python 3

```
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
        q = [root]
        while q:
            if q[0].left:
                q.append(q[0].left)
            if q[0].right:
                q.append(q[0].right)
            print("{} ".format(q[0].data), end='')
            q.pop(0)

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

```

Line: 27 Col: 42

□ Test against custom input

Run Code

Submit Code

Congrats, you solved this challenge!

Challenge your friends:

□ Test Case #0 □ Test Case #1 □ Test Case #2

Generating Badges...

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