Sylvia Ssanyu

sylvia-ssanyu@lambdastudents.com

BASIC INFO

Test: Computer Science - Binary Search Trees - Module Project

Solved: 5/5

Similarity: none

Score: 900/900

Finished On: 10 Aug 2021

Duration: 178m/168h

Label: None

Task	Time Spent	Score	Similarity
csNodesPerfectBinaryTree	1min	100/100	-
csPrimaryWeaknessBinarySearchTree	21sec	100/100	-
csBinarySearchTreeInOrderSuccessor	1min	100/100	-
balancedBinaryTree	6min	300/300	none
minimumDepthBinaryTree	11min	300/300	none

Task details: csNodesPerfectBinaryTree

Description:

How can you compute the total number of nodes in a "perfect" binary tree if you know the height?

✓ `2^h - 1` where `h` is the height of the binary tree

(Correct)

○ `log_2(h + 1)` where `h` is the height of the binary tree

(Incorrect)

○ `h^2 - 1` where `h` is the height of the binary tree

(Incorrect)

○ `2^h + 1` where `h` is the height of the binary tree

(Incorrect)

Task details: **csPrimaryWeaknessBinarySearchTree**

Description:

What is one of the primary weaknesses of the binary search tree as a data structure?

- [✓ The performance degrades if it becomes unbalanced.] *(Correct)*
- ○ Insertions and deletions are slower than a sorted array. *(Incorrect)*
- ○ They are unsorted by default. *(Incorrect)*
- ○ It has slower lookups than a sorted array. *(Incorrect)*

Task details: **csBinarySearchTreeInOrderSuccessor**

Description:

What does the phrase "in-order successor" mean when we are talking about a node in a binary search tree?

✓ The node that has the next highest value.

(Correct)

○ The node that has the next lowest value.

(Incorrect)

○ The node that has the maximum value.

(Incorrect)

○ The node that has the minimum value.

(Incorrect)

Task details: balancedBinaryTree

Description:

You are given a binary tree and you need to write a function that can determine if it is height-balanced.

A height-balanced tree can be defined as a binary tree in which the left and right subtrees of every node differ in height by a maximum of 1.

Example 1:
Given the following tree `[5,10,25,None,None,12,3]`:

```
    5
   / \
 10  25
    /  \
   12   3
```

return `True`.

Example 2:
Given the following tree `[5,6,6,7,7,None,None,8,8]`:

```
       5
      / \
     6   6
    / \
   7   7
  / \
 8   8
```

return `False`.

Solution (main.py3):

```python
#
# Binary trees are already defined with this interface:
# class Tree(object):
#   def __init__(self, x):
#     self.value = x
#     self.left = None
#     self.right = None
def height(root) -> int:
        # An empty tree has height -1
    if not root:
        return -1
    return 1 + max(height(root.left), height(root.right))
def balancedBinaryTree(root):


    if not root:
        return True

    return abs(height(root.left) - height(root.right)) < 2 \
            and balancedBinaryTree(root.left) \
            and balancedBinaryTree(root.right)


```

Task details: **minimumDepthBinaryTree**

Description:

You are given a binary tree and you are asked to write a function that finds its minimum depth. The minimum depth can be defined as the number of nodes along the shortest path from the root down to the nearest leaf node. As a reminder, a leaf node is a node with no children.

Example:
Given the binary tree `[5,7,22,None,None,17,9]`,

```
    5
   / \
  7   22
     /  \
    17   9
```

your function should return its minimum depth = `2`.

Solution (main.py3):

```
#
# Binary trees are already defined with this interface:
# class Tree(object):
#   def __init__(self, x):
#     self.value = x
#     self.left = None
#     self.right = None
def minimumDepthBinaryTree(root):

    if not root:
        return 0
    children = [root.left, root.right]
        # if we're at leaf node
    if not any(children):
        return 1

    min_depth = float('inf')
    for c in children:
        if c:
            min_depth = min(minimumDepthBinaryTree(c), min_depth)
    return min_depth + 1


```