Difficulty: **Category:** Binary Trees **Successful Submissions:** 60,563+

Branch Sums

Write a function that takes in a Binary Tree and returns a list of its branch sums ordered from leftmost branch sum to rightmost branch sum.

A branch sum is the sum of all values in a Binary Tree branch. A Binary Tree branch is a path of nodes in a tree that starts at the root node and ends at any leaf node.

Each `BinaryTree` node has an integer `value`, a `left` child node, and a `right` child node. Children nodes can either be `BinaryTree` nodes themselves or `None` / `null`.

Sample Input

```
tree =      1
         /     \
        2       3
      /   \    /  \
     4     5  6    7
   /   \  /
  8     9 10
```

Sample Output

```
[15, 16, 18, 10, 11]
// 15 == 1 + 2 + 4 + 8
// 16 == 1 + 2 + 4 + 9
// 18 == 1 + 2 + 5 + 10
// 10 == 1 + 3 + 6
// 11 == 1 + 3 + 7
```

Hints

Hint 1

Try traversing the Binary Tree in a depth-first-search-like fashion.

Hint 2

Recursively traverse the Binary Tree in a depth-first-search-like fashion, and pass a running sum of the values of every previously-visited node to each node that you're traversing.

Hint 3

As you recursively traverse the tree, if you reach a leaf node (a node with no "left" or "right" Binary Tree nodes), add the relevant running sum that you've calculated to a list of sums (which you'll also have to pass to the recursive function). If you reach a node that isn't a leaf node, keep recursively traversing its children nodes, passing the correctly updated running sum to them.

Optimal Space & Time Complexity

O(n) time | O(n) space - where n is the number of nodes in the Binary Tree

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```python
# This is the class of the input root. Do not edit it.
class BinaryTree:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

'''O(n) Time and O(n) Space'''
def branchSums(root):
    def BS(root, rsum=0, result=[]):
        if root is None:
            return

        if root.left is None and root.right is None:
            result.append(rsum+root.value)

        BS(root.left, rsum+root.value, result)
        BS(root.right, rsum+root.value, result)
        return result

    return BS(root)


# Kunal Wadhwa

# https://github.com/kunal5042
# https://leetcode.com/kunal5042/
# https://www.hackerrank.com/kunalwadhwa_cs
# https://www.linkedin.com/in/kunal5042/
```

Tests Quick Test Sandbox

Test Case 1

{
  "tree": {
    "nodes": [
      {"id": "1", "left": "2", "right": "3", "value": 1},
      {"id": "2", "left": "4", "right": "5", "value": 2},
      {"id": "3", "left": "6", "right": "7", "value": 3},
      {"id": "4", "left": "8", "right": "9", "value": 4},
      {"id": "5", "left": "10", "right": null, "value": 5},
      {"id": "6", "left": null, "right": null, "value": 6},
      {"id": "7", "left": null, "right": null, "value": 7},
      {"id": "8", "left": null, "right": null, "value": 8},
      {"id": "9", "left": null, "right": null, "value": 9},
      {"id": "10", "left": null, "right": null, "value": 10}
    ],
    "root": "1"
  }
}

Test Case 2

{
  "tree": {
    "nodes": [
      {"id": "1", "left": null, "right": null, "value": 1}
    ],
    "root": "1"
  }
}

Test Case 3

{

Custom Output Raw Output Submit Code

Yay, your code passed all the test cases!

9 / 9 test cases passed.

- Test Case 1 passed!
- Test Case 2 passed!
- Test Case 3 passed!
- Test Case 4 passed!
- Test Case 5 passed!
- Test Case 6 passed!
- Test Case 7 passed!
- Test Case 8 passed!