Fizz Buzz Tree

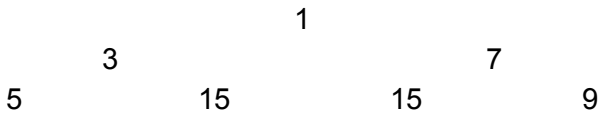
Collaborators: Joe Zabaleta & Tia Low

Problem Domain

- Write function FizzBuzzTree that takes in a tree (k-ary tree) as an argument
- Modify the values of the Nodes in the tree according to the following:
    - If value divisible by 3, replace with “Fizz”
    - If value divisible by 5, replace with “Buzz”
    - Divisible by 3 AND 5, replace with “FizzBuzz”
    - Not divisible by 3 OR 5, turn value into string

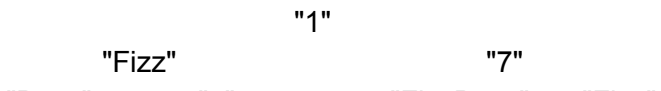
Input / Output examples:

INPUT:



INPUT PREORDER = [1, 3, 5, 15, 7, 15, 9]

OUTPUT:



OUTPUT preOrder = ["1", "Fizz", "Buzz", "4" , "7" , "FizzBuzz", "Fizz"]

Algorithm

- Check if there’s a root, if not throw an Error (“empty tree”)
- Traverse the tree to visit each Node, at each Node check the value (modular 3 and/or 5)
    - If the value falls under one of the rules, chance current Node’s value
- Go through entire tree (until !root) changing values if necessary, no specific extra step for return

Pseudo / Real Code
define a function called FizzBuzz(tree) {

```
  function _fizzBuzz(root) {

    if(!root) {
      return;
    }

      if (root.value % 3 === 0 && root.value % 5 === 0) {
            root.value = 'FizzBuzz';
      } else if(root.value % 3 === 0){
            root.value = 'Fizz';
      } else if (root.value % 5 === 0) {
            root.value = 'Buzz';
      } else {
            root.value = root.value.toString();
            (something to try - does root.value.toString() work?)
      }

    _fizzBuzz(root.left);
    _fizzBuzz(root.right);

  }

  _fizzBuzz(tree.root);

}
```

}

Walk-through using solution code above

fizzBuzz(tree1)

1: function call
ROOT.VALUE = 1
1.1 - check if root is null ------ false
1.2 - check if value % 3 equals 0 ---- false

1.3 - check if value % 5 equals 0 --- false
1.4 - check if value 3 / 5 ---- false
1.5 else --- true
root.value = string of "1"

call fizzbuzz root.left-- true

2: function call
ROOT.VALUE = 3
2.1 - check if root is null ------ false
2.2 - check if value % 3 equals 0 ----true
root.value = 'Fizz"

call fizzbuzz root.left-- true

3: function call
ROOT.VALUE = 5
3.1 - check if root is null ------ false
3.2 - check if value % 3 equals 0 ----false
3.3 - check if value % 5 ------ true
root.value = 'Buzz"

call fizzbuzz root.left-- true

call fizzbuzz root.right

4: function call
ROOT.VALUE = 5
4.1 - check if root is null ------ false
4.2 - check if value % 3 equals 0 ----true
15 = "fizz"
call fizzbuzz root.left-- true

call fizzbuzz root.right

CALL STACK:
4: function call root.value 15
3. function call root.value 5
2. function call root.value 3
1: function call root.value 1

Big O

- Space: O(1)
    - Reassigning variables, not creating anything new
- Time: O(n)
    - At the minimum, but also consider that with recursion each step is calling function two more times