Data Structures

Binary Tree Homework 2

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Problem #1: Inorder iterative

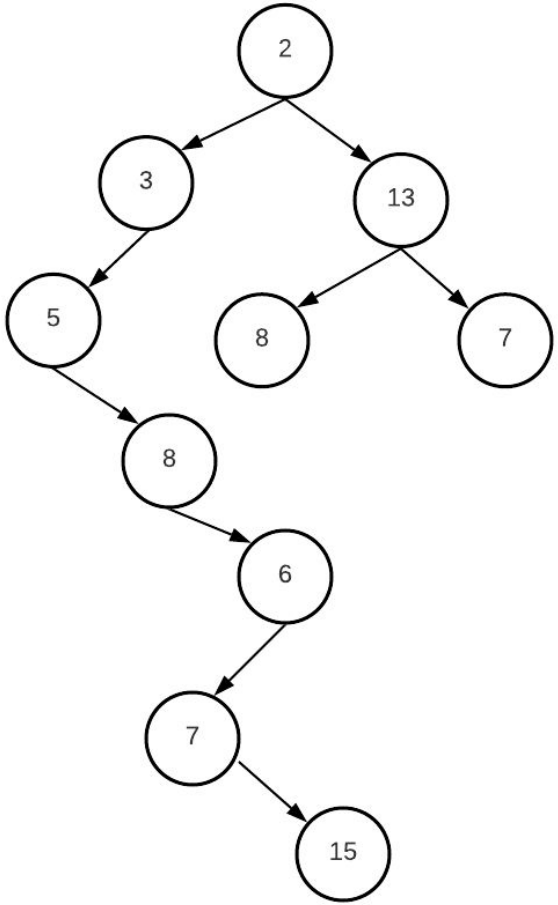
- Develop a function to return the tree inorder
- However, this time you won't use recursion. Code the recursion in an iterative way (use a stack)

```
tree = BinaryTree(1)
tree.add([2, 4, 7], ['L', 'L', 'L'])
tree.add([2, 4, 8], ['L', 'L', 'R'])
tree.add([2, 5, 9], ['L', 'R', 'R'])
tree.add([3, 6, 10], ['R', 'R', 'L'])

assert tree.inorder_iterative() == \
       [7, 4, 8, 2, 5, 9, 1, 3, 10, 6]
```

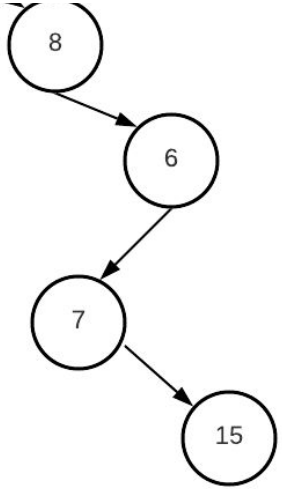
Problem #2: Left Tree boundary

- **def** traverse_left_boundry(self)
  - Returns a **list** of all values along the left tree boundary
- Nodes of the tree left boundary are nodes from the root to the **left-most** node in a tree
- Node 15 here is the **most-left** node
  - Most-left doesn't mean you just keep going left until there are no more left nodes
  - Can you see why we call it a boundary?
- **list is: 2 3 5 8 6 7 15**
- **Code in O(n) time**



Problem #2: Left Tree boundary

- **list is: 8 6 7 15**



Problem #3: LeetCode 543 - Diameter of Binary Tree

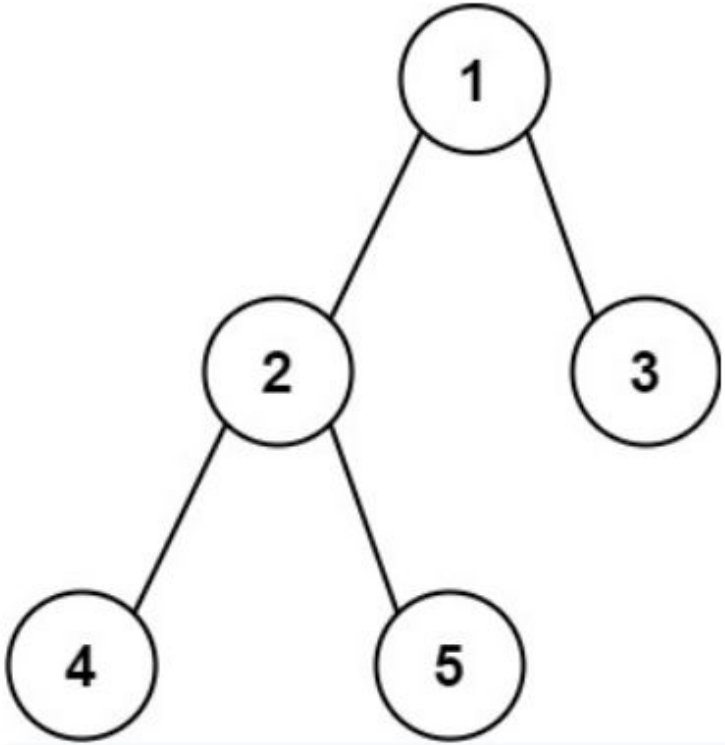
Given the `root` of a binary tree, return *the length of the **diameter** of the tree*.

The **diameter** of a binary tree is the **length** of the longest path between any two nodes in a tree. This path may or may not pass through the `root`.

The **length** of a path between two nodes is represented by the number of edges between them.

- Hint
  - Make use of the the tree height function
  - Given a node: the diameter either pass with the node or pass at the children
    - Develop logic for each of these 2 cases

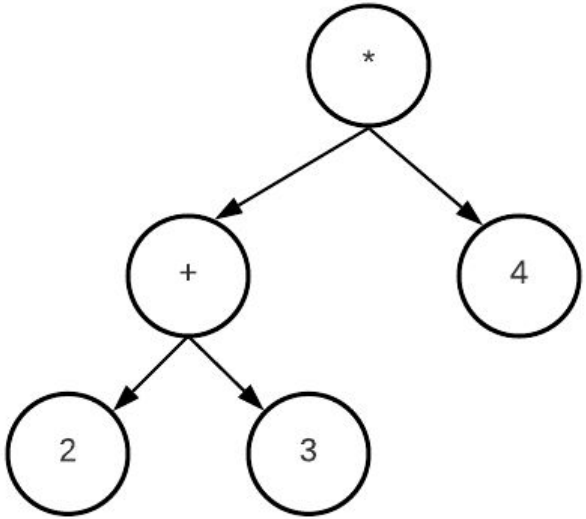
Example 1:




```
Input: root = [1,2,3,4,5]
Output: 3
Explanation: 3 is the length of the path [4,2,1,3] or [5,2,1,3].
```

Problem #4: Expression Tree

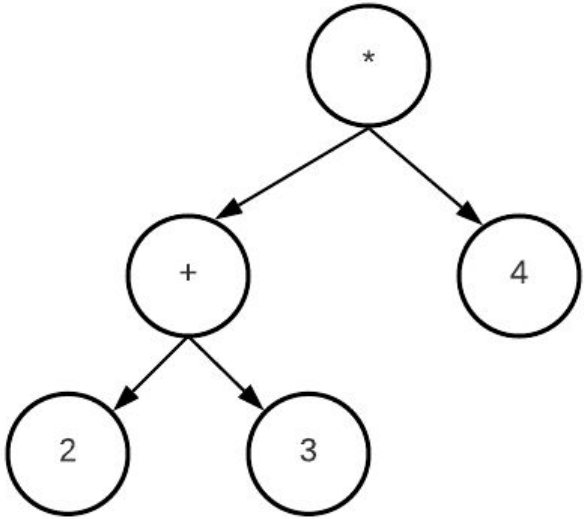
- Change the init method in the binary tree to accept a string representing a postfix expression
  - Single digits, no unary
  - Operators: + - * / ^
- The function builds an expression tree based on the given postfix expression
  - On right side the tree for expression: 23+4*
- To validate, you can make sure the postorder of the tree is as same as the postfix




```
class BinaryTree:
    def __init__(self, postfix):
```

Problem #5: Expression Tree Inorder Traversal

- Extend the previous code to print the tree in the inorder format
- However, when we try to print the tree inorder for this expression tree, the output is: 2+3*4
  - But this is wrong, as it should be (2+3)*4
- This function adds **proper parentheses** to give a valid infix output
  - 51+2/ ⇒ (5+1)/2
  - 534*2^+ ⇒ 5+((3*4)^2)



```
tree = BinaryTree('23+4*')
tree.print_inorder_expression()
# (2+3)*4

tree = BinaryTree('51+2/')
tree.print_inorder_expression()
# (5+1)/2

tree = BinaryTree('534*2^+')
tree.print_inorder_expression()
# 5+((3*4)^2)
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."