Daily Coding Problem #110

Problem

This problem was asked by Facebook.

Given a binary tree, return all paths from the root to leaves.

For example, given the tree:

```
   1
  / \
 2   3
    / \
   4   5
```

Return `[[1, 2], [1, 3, 4], [1, 3, 5]]`.

Solution

A binary tree is a data structure in which each node has at most two children (left / right). To solve this question, we can iterate over the tree to check nodes without children, also known as leaf nodes.

The `BSTNode` class might be defined like this:

```
class BSTNode:
    def __init__(self, value, left=None, right=None):
        self.parent = None
        self.value = value
```

```
        self.left = left
        self.right = right
        if left:
            left.parent = self
        if right:
            right.parent = self

    def path(self):
        path = []
        current = self
        while current:
            path = [current.value] + path
            current = current.parent
        return path
```

We conveniently store the parent node for the `path` function, so we can find the ancestors to get their path to the root. We can initialize our binary tree with this code:

```
"""
    For the binary tree shown below:

       1
      / \
     2   3
        / \
       4   5
"""
root = Node(
    value=1,
    left=Node(2),
    right=Node(3, Node(4), Node(5))
)
```

Sometimes programmers may be tempted to use a recursive function, like this:

```
def find_leaves(node):
    if not node.left and not node.right:
        return [node.path()]
    leaves = []
```

```
    if node.left:
        leaves += find_leaves(node.left)
    if node.right:
        leaves += find_leaves(node.right)
    return leaves
```

We should avoid this practice on large or unknown trees because of the possibility of a call stack overflow.

A safer way is to use a FIFO queue, linked list or a simple list. Like this:

```
def find_leaves(node):
    leaves = []
    queue = list()
    queue.append(node)
    while len(queue):
        node = queue.pop()
        if not node.left and not node.right:
            leaves += [node.path()]
            continue
        if node.right:
            queue.append(node.right)
        if node.left:
            queue.append(node.left)
    return leaves
```