node to root path

```python
class TreeNode:
    def __init__(self, val):
        self.data = val
        self.left = None
        self.right = None


class Pair:
    def __init__(self, level, node):
        self.level = level
        self.node = node


import collections


def nodeToLeafPath(root, data, res):
    if root is None:
        return False
    if root.data == data:
        res.append(root.data)
        return True
    if nodeToLeafPath(root.left, data, res) or nodeToLeafPath(root.right,
data, res):
        res.append(root.data)
        return True
    return False


root = TreeNode(1)
root.left = TreeNode(2)
root.right = TreeNode(3)
root.left.left = TreeNode(4)
root.left.right = TreeNode(5)
root.right.left = TreeNode(6)
root.right.right = TreeNode(7)

res  = []
```

```python
nodeToLeafPath(root,5,res)
print(res)
```

```python
class TreeNode:
    def __init__(self, val):
        self.data = val
        self.left = None
        self.right = None


class Pair:
    def __init__(self, level, node):
        self.level = level
        self.node = node


import collections


def nodeToLeafPath(root, data):
    if root is None:
        return None
    if root.data == data:
        res = [root.data]
        return res
    leftL = nodeToLeafPath(root.left, data)
    if leftL is not None:
        leftL.append(root.data)
        return leftL
    rightL = nodeToLeafPath(root.right, data)
    if rightL is not None:
        rightL.append(root.data)
        return rightL
    return None


root = TreeNode(1)
root.left = TreeNode(2)
root.right = TreeNode(3)
root.left.left = TreeNode(4)
root.left.right = TreeNode(5)
root.right.left = TreeNode(6)
root.right.right = TreeNode(7)
```

```
print(nodeToLeafPath(root, 5))
```