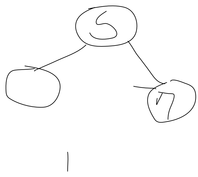

if(!root) return null;
let stack = [], newRoot = null, newTree = null;

stack = []

newRoot ===null

newTree === null

//1st while loop
while(root){ //root === [5, 1, 7]
stack.push(root); //[5, 1, 7]
//stack === [[5, 1, 7]]
console.log("root.left", root.left) //[1]

root = root.left; // root = [1]
console.log('stack', stack)
//[[[5, 1, 7]]
}

outer while loop(has 1st and 2nd while loops)

first while loop 1st iteration

root === [5,1,7]
stack.push(root)

root = root.left
root === [1, null, null]

first while loop 2nd iteration

root === [1, null, null]
stack.push(root)

root = root.left
root === null

2nd while LOOP - 1st iteration

stack

[1, null, null] < top
[5,1,7]

Pop

stack.length == 2

node === [1, null, null]

newRoot === [1, null, null]

newTree === [1, null, null]

node = [1, null, null]

stack = []

2nd while loop
while(stack.length > 0){
let node = stack.pop()

if(!newRoot){

newRoot = newTree = node;

} else {
newTree.right = node;
newTree = newTree.right;
}

node.left = null;

// right
if(!node.right){
break;
}
}

newRoot === [1, null, null]

newTree === [1, null, null]

node = [1, null, null]

2nd while LOOP - 2nd iteration

stack

[5,1,7]

Pop

stack.length == 1

node === [5, 1, 7]

newRoot isn't empty newRoot === [1, null, null]

newTree.right = node // newTree.right = [5,1,7]
newTree = newTree.right // newTree = [5 , 1, 7]
node.left = null
node = [5, null, 7]

stack = []

2nd while loop
while(stack.length > 0){
let node = stack.pop()

if(!newRoot){

newRoot = newTree = node;

} else {
newTree.right = node;
newTree = newTree.right;
}

node.left = null;

// right
if(!node.right){
break;
}
}

newRoot === [1, null, null]

newTree === [5, 1, 7]

node = [5, null, 7]

if(node.right){ //node.right === [7]
root = node.right;

break;
}

root ==== [7]

newRoot ===[1] or [1, null, null]

newTree === [5,1,7]

first while loop 1st iteration

root === [7]
stack.push(root)

root = root.left
root === null

[7]

while(stack.length > 0){

let node = stack.pop();
if(!newRoot){
newRoot = newTree = node;
} else {
newTree.right = node;
newTree = newTree.right;
}

node.left = null;

newRoot ===[1] or [1, null, null]

newTree === [5,1,7]

2nd while LOOP - 1st iteration

[7]

Pop

stack.length == 1

node === [7]

newTree.right ===
[7]

newTree = newTree.right
newTree = [7]

newRoot ===[1] or [1, null, null]

node.left === null
node === [7, null, null]

newTree = [7]

if(node.right){ //node.right === null
root = node.right;

break;
}