# AVL Insertion

This lesson will cover the insertion operation in AVL trees, discussing all the four insertion cases.

**We'll cover the following**



## Introduction #

Insertion in AVL trees is done the same way that BST insertion is done. However, when a node is inserted into a BST it usually becomes *unbalanced*, i.e., the tree has a node which has a left-right subtree height difference greater than 1. So, AVL trees have to be rebalanced after insertion, unlike BSTs. To re-balance the tree, we need to perform a 'rotation'. But before going deep let's look at AVL tree rebalancing case-by-case.

Let's look at terms that we will be using while re-balancing the tree.

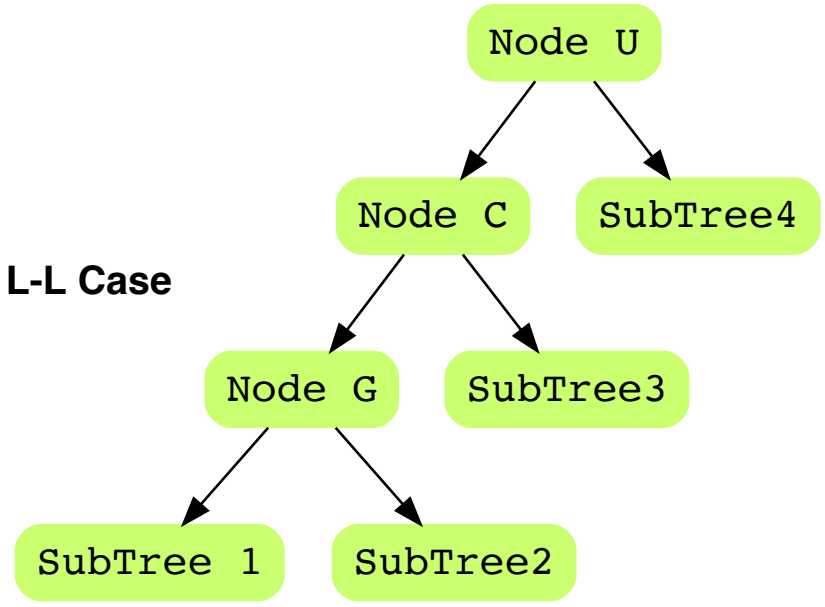
```
Node U – an unbalanced node
Node C – child node of node U
Node G – grandchild node of node U
```

# Insertion Cases #

To rebalance the tree, we will perform rotations on the subtree with Node U being the root node. There are two types of rotations (left and right). We came across four different scenarios based on the arrangements of Nodes U, C, and G.

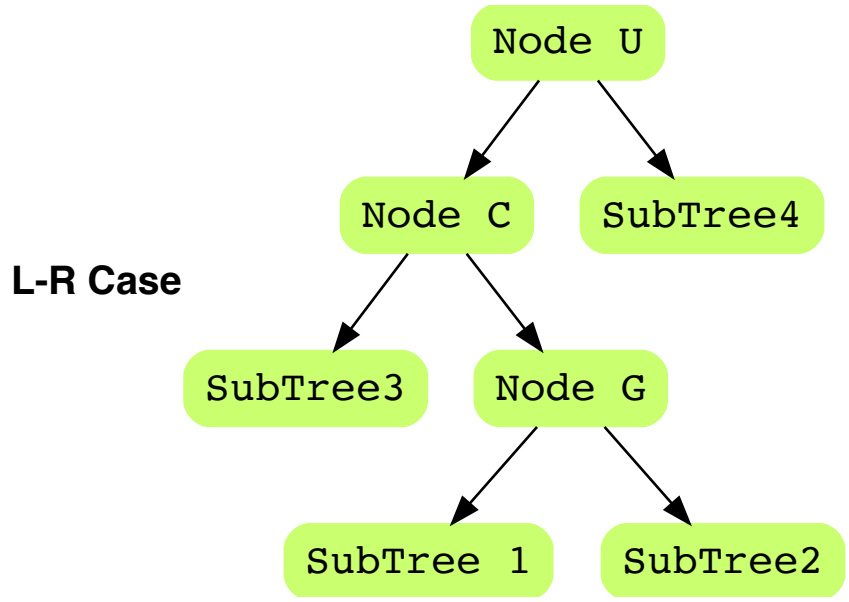
```
Left-Left: Node C is the left-child of Node U, and Node G is left-child of Node C

Left-Right: Node C is the left-child of Node U, and Node G is right-child of Node C

Right-Right: Node C is the right-child of Node U, and Node G is right-child of Node C

Right-Left: Node C is right-child of Node U, and Node G is left-child of Node C
```

## Case 1: Left-Left #



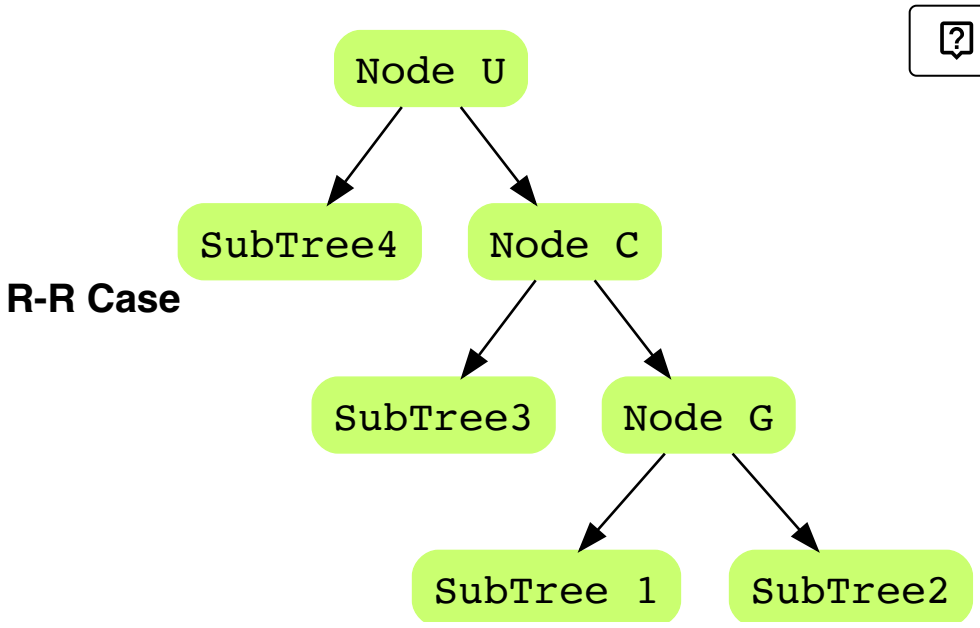

1 of 4

## Case 2: Left-Right #



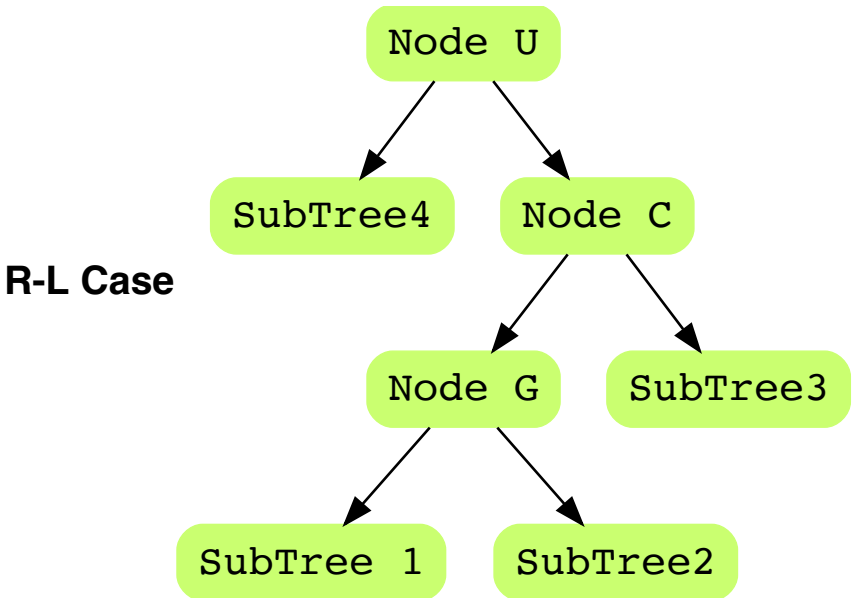

1 of 7

## Case 3: Right-Right #




1 of 4

## Case 4: Right-Left #




1 of 8

That's it on AVL tree insertion! Lets move on to AVL tree dele[illegible] chapter!