

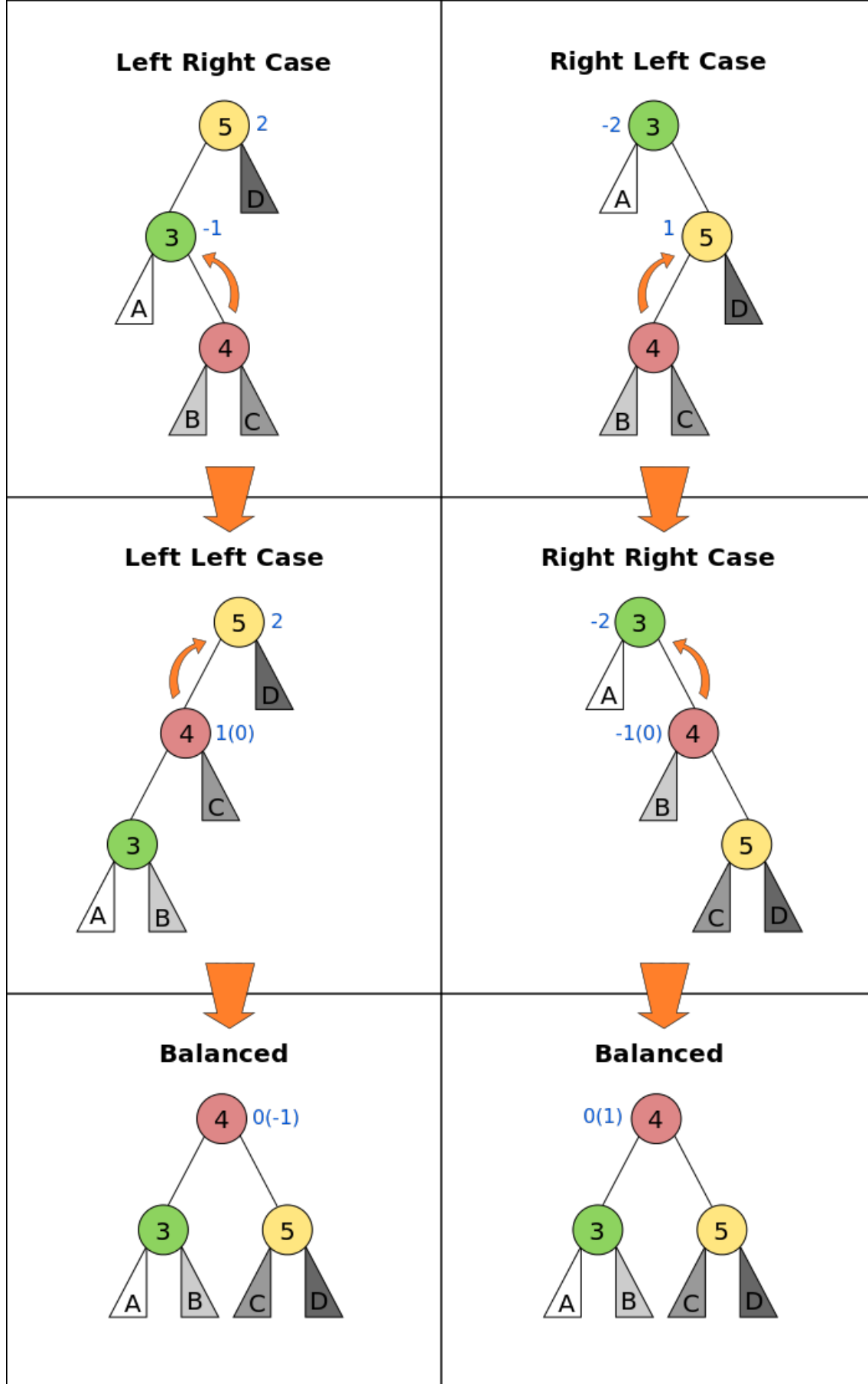
Self Balancing Tree

An AVL tree (Georgy Adelson-Velsky and Landis' tree, named after the inventors) is a self-balancing binary search tree. In an AVL tree, the heights of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, rebalancing is done to restore this property.

We define balance factor for each node as :

$$\text{balanceFactor} = \text{height}(\text{left subtree}) - \text{height}(\text{right subtree})$$

The balance factor of any node of an AVL tree is in the integer range [-1,+1]. If after any modification in the tree, the balance factor becomes less than -1 or greater than $+1$, the subtree rooted at this node is unbalanced, and a rotation is needed.



(https://en.wikipedia.org/wiki/AVL_tree)

You are given a pointer to the root of an AVL tree. You need to insert a value into this tree and perform the necessary rotations to ensure that it remains balanced.

Input Format

You are given a function,

```
node *insert(node * root,int new_val)
```

```
{  
  
}
```

'node' is defined as :

```
struct node  
{  
    int val;          //value  
    struct node* left; //left child  
    struct node* right; //right child  
    int ht;           //height of the node  
} node;
```

You only need to complete the function.

Note: All the values in the tree will be distinct.

Output Format

Insert the new value into the tree and return a pointer to the root of the tree. Ensure that the tree remains balanced.

Sample Input

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

The value to be inserted is 6.

Sample Output

```
  3  
 / \  
2   5  
  \  
  /\  
 4  6
```

Explanation

After inserting 6 in the tree. the tree becomes:

```
  3 (Balance Factor = -2)  
 / \  
2   4 (Balance Factor = -2)  
  \  
   5 (Balance Factor = -1)  
  \  
   6 (Balance Factor = 0)
```

Balance Factor of nodes 3 and 4 is no longer in the range $[-1,1]$. We need to perform a rotation to balance the tree. This is the right right case. We perform a single rotation to balance the tree.

After performing the rotation, the tree becomes :

3 (Balance Factor = -1)

/ \

(Balance Factor = 0) 2 5 (Balance Factor = 0)

/ \

(Balance Factor = 0)4 6 (Balance Factor = 0)