AVL Trees Insertion & Deletion

Krishnaprasad V
1BM17CS047
14.10.20

Insert

```
insert (node, key):
{   if (!node)          // perform normal BST insertion
        return TreeNode(key)
    else if (key < node->val)
        node->left = insert (node->left, key)
    else if (key > node->key)
        node->right = insert (node->right, key)
    else
        return node
// update height of ancestor node
    node->height = 1 + max (height (node->left), height (node->right))
// get balance factor
    balance = getBalance (node)

//4 cases
if (balance > 1 && key < node->left->key)      // left
    return leftRotate (node)
if (balance < -1 && key > node->right->key)    // right
    return rightRotate
if (balance > 1 && key > node->right->key)     // right left
{   node->right = rightRotate (node->right)
    return leftRotate (node)
}
```

```
if (balance < -1 && key < node->right->key)
{
    node->right = rightRotate(node->right)
    return leftRotate(node)
}
return node
}
```

Delete

```
deleteNode(Node node, int key)
{
    if (node == NULL)
        return node
    if (key < node->key)
        node->left = deleteNode(node->left, key)
    else if (key > node->key)
        node->right = deleteNode(node->right, key)
    else
    {
        if (node->left == NULL || node->right == NULL)
        {
            Node temp = node->left ? node->left : node->right;
            if (temp == NULL)
            {
                temp = node
                node = NULL
            }
            else
                *node = *temp
        }
        else
        {
            Node temp = minValueNode(node->right)
            node->key = temp->key
            node->right = deleteNode(node->right, temp->key)
        }
```

```
if (node == NULL)
    return 0;

// height
node->height = 1 + max(height(node->left), height(node->right));

// balance
balance = getBalance(node);

// 4 cases
if (balance > 1 && getBalance(node->left) >= 0)   // left
    return rightRotate(node);

if (balance < -1 && getBalance(node->right) <= 0)   // right
    return leftRotate(node);

if (balance > 1 && getBalance(node->left) < 0)   // left right
{
    node->left = leftRotate(node->left);
    return rightRotate(node);
}

if (balance < -1 && getBalance(node->right) > 0)
{
    node->right = rightRotate(node->right);
    return leftRotate(node);
}

return node;
}
```