AVL Tree Implementation
Data Structures

Cain Susko

Queen's University
School of Computing

February 28, 2022

Update Height

it is important to keep track of and update the height of an AVL tree, as mentioned in the lesson 5-3-2. the height is mainly changed during insertion and deletion. an implementation of this process could look like so. Note that this code is within a ALVTree class.

```
# step 1, perform normal BST insert
def insert(self, root, key):
        if not root:
                return TreeNode(key)
        elif key < root.val:
                root.left = self.insert(root.left, key)
        else:
                root.right = self.insert(root.right, key)

# step 2, Update the height of the ancestor node

        root.height = 1 + max(self.Height(root.left), self.Height(root.right))

# step 3, get the balance factor of the node

        balance = self.getBalance(root)

# step 4, if node is unbalanced then try out the 4 cases

# CASE 1 - LeftLeft
if balance < -1 and key < root.left.val:
        return self.rightRotate(root)

# CASE 2 - RightRight
if balance > 1 and key > root.right.val:
        return self.leftRotate(root)

# CASE 3 - LeftRight
if balance < -1 and key > root.left.val:
        root.left = self.lefttRotate(root.left)
        return self.rightRotate(root)

# CASE 4 - RightLeft
if balance > 1 and key < root.right.val:
        root.right = self.rightRotate(root.right)
        return self.leftRotate(root)
```

the implementation for the rotate functions mentioned above are the following:

```
def rightRotate(self, B):

    A= B.left
beta = A.right

    # Perform rotation
    A.right = B
    B.left = beta

    # Update heights
    B.height = 1 + max(self.getHeight(B.left),
                       self.getHeight(B.right))
    A.height = 1 + max(self.getHeight(A.left),
                       self.getHeight(A.right))

    # Return the new root
    return A
```

```
def leftRotate(self, A):

    B = A.right
beta = B.left

    # Perform rotation
    B.left = A
    A.right = beta

    # Update heights
    A.height = 1 + max(self.getHeight(A.left),
                       self.getHeight(A.right))
    B.height = 1 + max(self.getHeight(B.left),
                       self.getHeight(B.right))

    # Return the new root
    return B
```