AVL Trees

Amogh Kasbekar
IBM18CS014

```
insert(value, root){
  if (root == NULL) return Newnode(value)
  if (root->val > value)  root->left = insert(root->left, value)
  else if (root->val < value) root->right = insert(root->right, value)
  else return root.

  check balance of root.
  if balance < -1 and value > root->right->val
        rotateLeft(root)
  if balance < -1 and value < root->right->val
        root->right = rotateRight(root->right);
        return rotateLeft(root)
  if balance > 1 and value < root->left->val
        return rotateRight(root)
  if balance > 1 and data > root->left->val
        root->left = rotateLeft(root->left)
        return rotateRight(root);

  return root
}

RotateRight(node){
  r = node->right
  leftofr = r->left
  r->left = node
  node->right = leftofr
  update height of r and node.
  return r
}
```

```
Rotate right (node) {
    l = node -> left
    right of l = l -> right
    l -> right = node
    node -> left = right of l
    update heights of l and node
    return l
}

delete (root, data) {
    if (root == NULL) return root
    if (data < root->val)
        root -> left = delete (root -> left, data)
    else if (data > root -> val)
        root -> right = delete (root -> right, data)
    else {
        if (root -> left is NULL or root -> right is NULL) {
            t = NULL
            if (root -> left != NULL) t = root -> left
            else t = root -> right
            if (t == NULL) {
                t = root
                root = NULL
            }
            else root = t
            free (t)
        }
        else {
            t = root -> right
```


Acey

```
while (t->left != NULL){
    t = t->left
    }
root->val = t->val
root->right = delete(root->right, t->val)
}
if (root == NULL) return root
```

update height of root.

check the balance of root.

if (balance > 1 and left of left chid height is greater that right of left child)

return rotateRight(root)

if (balance > 1 and left of left child height is less than right of left child)

root->left = rotateLeft(root->left)

return rotateRight(root)

if(balance < -1 and height of left of right child is less than height of right of right child)

return rotateRight(root)

if(balance < -1 and height of left of rightchild is greater than height of right of right):

root->right = rotateRight(root->right)

return rotateLeft(root)

return root

}