ADS Lab Writeup

Shikha.N
1BM18CS149

AVL Trees

Insertion

```
Node* insert (Node *node, int key)
{ if node == NULL
        return newNode (key)
    //normal BST insertion
    if key < node -> key
        node -> left = insert (node -> left, key)
    else if key > node -> key
        node -> right = insert (node -> right,
                                key)
    else
        return node
//update height
node -> height = 1 + max (height (node -> left),
                          height (node -> right))

// calculate Balance factor
    int balance = getBalance (node)

// if Unbalanced.
// case 1 => Left rotation.
    if (balance < -1 && key > node -> right ->
                                key)
        return leftRotate (node)
// case 2 => right rotation.
    if (balance > 1 && key < node -> left ->
                                key)
        return rightRotate (node)
```

Shikhar N
1BM18CS149

```
// Left - Right Rotation.
if balance > 1 && key > node -> left -> key
    { node -> left = leftRotate (node -> left)
      return rightRotate (node)
    }
// Right - left Rotation
if balance < -1 && key < node -> right -> key
    {
      node -> right = RightRotate (node -> right)
    return leftRotate (node)

    }.
    return node
}.
```

Deletion ⇒

Shikha N
1MI18CS149

```
Node * 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
    {
        // node with only one child/no child
        if (node->left == NULL ||
            node->right == NULL)
        {
            Node *x = root->left ?
                      root->left :
                      root->right

            // no child
            if x == NULL
            {
                x = node
                node = NULL
            }
            // one child
            else
            {
                *node = *x
                free(x)
            }
    }
```

Shikha-N
1BM18CS149

```
// node with 2 children.
else
{
    Node * temp = inorder_successor
                  (node -> right)

    node -> key = temp -> key

    node -> right = deleteNode (node -> right,
                                temp -> key)
}
}
if node == NULL
    return node

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

int balance = getBalance (node)
// Right Rotation
if (balance > 1 && getBalance (node -> left) >=
                                              0)
    return rightRotate (node)
// Left Rotation + Right Rotation
if (balance > 1 && getBalance (node -> left) < 0)
{ node -> left = leftRotate (node -> left)
    return rightRotate (node)
}
// left Rotation
if (balance < -1 && getBalance (node -> right) <= 0)
    return leftRotate (node)
```

Shikha.N
1SM11CS149

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

Node * leftRotate (Node * temp)
{
    Node * y = temp -> right
    Node * x = y -> left

    y -> left = temp.
    temp -> right = x.
temp -> height = max (height(temp -> left),
                      height(temp -> right))
                  + 1
y -> height = max (height(y -> left),
                   height(y -> right)) +
                   1

    return y
}.
```

Shizna-N
1BM18CS149

```
Node * RightRotate (Node *y)
{
    Node *x = y->left
    Node *temp = x->Right

    x->right = y
    y->left = temp.
y->height = max (height (y->left),
                 height (y->right)) + 1

x->height = max (height (x->left),
                 height (x->right)) + 1

    return x
}.

int getBalance (Node * node)
{   if node == NULL
        return 0
    return height (node->left) -
           height (node->right)

}.
```