Data Structures and Algorithms

CSE2001

Lab - 3 - Assignment - 2

Yashwanth Reddy
19BCE7362
Date- 6thJuly2021

Problem : AVL Tree

```
class Node {
  int item, height;
  Node left, right;

  Node(int d) {
    item = d;
    height = 1;
  }
}

class AVLTree {
  Node root;

  int height(Node N) {
    if (N == null)
      return 0;
    return N.height;
  }
```

```
int max(int a, int b) {
  return (a > b) ? a : b;
}

Node rightRotate(Node y) {
  Node x = y.left;
  Node T2 = x.right;
  x.right = y;
  y.left = T2;
  y.height = max(height(y.left), height(y.right)) + 1;
  x.height = max(height(x.left), height(x.right)) + 1;
  return x;
}

Node leftRotate(Node x) {
  Node y = x.right;
  Node T2 = y.left;
  y.left = x;
  x.right = T2;
  x.height = max(height(x.left), height(x.right)) + 1;
  y.height = max(height(y.left), height(y.right)) + 1;
  return y;
}
int getBalanceFactor(Node N) {
  if (N == null)
    return 0;
  return height(N.left) - height(N.right);
}

// Insert a node
Node insertNode(Node node, int item) {
```

```
  if (node == null)
    return (new Node(item));
  if (item < node.item)
    node.left = insertNode(node.left, item);
  else if (item > node.item)
    node.right = insertNode(node.right, item);
  else
    return node;
  node.height = 1 + max(height(node.left), height(node.right));
  int balanceFactor = getBalanceFactor(node);
  if (balanceFactor > 1) {
    if (item < node.left.item) {
      return rightRotate(node);
    } else if (item > node.left.item) {
      node.left = leftRotate(node.left);
      return rightRotate(node);
    }
  }
  if (balanceFactor < -1) {
    if (item > node.right.item) {
      return leftRotate(node);
    } else if (item < node.right.item) {
      node.right = rightRotate(node.right);
      return leftRotate(node);
    }
  }
  return node;
}

Node nodeWithMimumValue(Node node) {
```

```
  Node current = node;
  while (current.left != null)
    current = current.left;
  return current;
}

void preOrder(Node node) {
  if (node != null) {
    System.out.print(node.item + " ");
    preOrder(node.left);
    preOrder(node.right);
  }
}

private void printTree(Node currPtr, String indent, boolean last) {
  if (currPtr != null) {
    System.out.print(indent);
    if (last) {
      System.out.print("R----");
      indent += "    ";
    } else {
      System.out.print("L----");
      indent += "|   ";
    }
    System.out.println(currPtr.item);
    printTree(currPtr.left, indent, false);
    printTree(currPtr.right, indent, true);
  }
}
```

```
  public static void main(String[] args) {
    AVLTree tree = new AVLTree();
    tree.root = tree.insertNode(tree.root, 36);
    tree.root = tree.insertNode(tree.root, 12);
    tree.root = tree.insertNode(tree.root, 62);
    tree.root = tree.insertNode(tree.root, 3);
    tree.root = tree.insertNode(tree.root, 23);
    tree.root = tree.insertNode(tree.root, 58);
    tree.root = tree.insertNode(tree.root, 2);
    tree.root = tree.insertNode(tree.root, 1);
    tree.printTree(tree.root, "", true);

  }
}
```

Output

```
C:\Users\yashw\Desktop>javac AVLTree.java --release 8

C:\Users\yashw\Desktop>java AVLTree
R----36
   L----12
   |  L----2
   |  |  L----1
   |  |  R----3
   |  R----23
   R----62
      L----58

C:\Users\yashw\Desktop>
```