*S.S.M.A.SRI HARSHA AM.EN.U4 AIE19066*

# DATA STRUCTURES AND ALGORITHMS – 3 A.V.L

*LAB SHEET – 1*

public class AVLNode{ protected int data; protected AVLNode left; protected AVLNode right; protected int bf;

public AVLNode() {}; public AVLNode(int d) { data = d;

bf = 0;

left = null; right = null;

}

public int max(int i1 , int i2) { int t;

if(i1>i2) { t = i1;

}

else { t = i2;

}

return t;

}

public int height() {

if (this.left==null && this.right==null) return 0;

else if (this.left!=null && this.right==null) return left.height() + 1;

else if (this.left==null && this.right!=null) return right.height() + 1;

else return max(left.height(), right.height()) + 1;

}

public int computeBalance() {

if (this.left==null && this.right== null) return 0;

else if (this.left!=null && this.right==null) return left.height()+1;

else if (this.left==null && this.right!=null)return -

1 \* (right.height()+1);

else return left.height() - right.height(); } public AVLNode rwlc() {

AVLNode lc = left; left = lc.right; lc.right = this; return lc;

}

public AVLNode rwrc() {

AVLNode rc = right; right = rc.left; rc.left =this; return rc;

}

public AVLNode insert(int key) { if(key<data) {

if(left == null) {

left = new AVLNode(key);

}

else

left = left.insert(key);

}

else {

if(right == null) {

right = new AVLNode(key);

}

else

right = right.insert(key);

}

bf = computeBalance(); switch(bf) {

case 2: if(left.bf >= 0) {

return rwlc();

}

else {

left =left.rwrc(); return rwlc();

}

case -2: if(right.bf <= 0) { return rwrc();

}

else {

right = right.rwlc(); return rwrc();

}

}

return this;

}

public void inorder() { if(left!=null) left.inorder(); System.out.print(data+" "); if(right!=null) right.inorder();

}

public void preorder() { System.out.print(data+" "); if(left!=null) left.preorder(); if(right!=null) right.preorder();

}

public void postorder() { if(left!=null) left.postorder(); if(right!=null) right.postorder(); System.out.print(data+" ");

}

public void levelorder() { int h=height();

for(int i=1;i<=h+1;i++) printLevel(i);

}

public void printLevel(int level) { if(level==1) System.out.print(data+" ");

else if(level>1) { if(left!=null) left.printLevel(level-1); if(right!=null) right.printLevel(level-1);

}

}

public boolean search(int key) { if(key==data)

return true;

else if(key<data&&left!=null) return left.search(key);

else if(key>data&&right!=null) return right.search(key);

else

return false;

}

}

public class AVLTree{ protected AVLNode root; public void insert(int key) { if(root == null) {

root = new AVLNode(key);

}

else { root.insert(key);

}

}

public void inorder() { if(root==null)

return; else {

System.out.println(); root.inorder();

}

}

public void preorder() { if(root==null)

return; else {

System.out.println(); root.preorder();

}

}

public void postorder() { if(root==null)

return; else {

System.out.println(); root.postorder();

}

}

public void levelorder() { if(root==null)

return; else {

System.out.println(); root.levelorder();

}

}

public boolean search(int key) {

if(root==null) return false; else

return root.search(key);

}

}