Nama : Andyan Yogawardhana

NIM : 21/482180/PA/21030

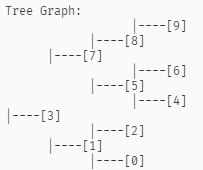
Kelas : KOMB1

Tugas 4 - AVL

1. Implementasi AVL

1. package Tree;
2. public class AVL {
3. public static void main(String[] args) {
4. int[] data = {0,1,2,3,4,5,6,7,8,9};
5. Tree tree = new Tree();
6. for(int i = 0; i < data.length; i++) {
7. tree.setRoot(tree.insert(tree.getRoot(), data[i]));
8. }
9. tree.printTree();
10. }
11. }
12. class Node {
13. private int data, height;
14. private Node left, right;
15. public Node(int d) {
16. this.data = d;
17. this.height = 1;
18. }
19. public int getData() {
20. return this.data;
21. }
22. public int getHeight() {
23. return this.height;
24. }
25. public void setHeight(int h) {
26. this.height = h;
27. }
28. public Node getLeft() {
29. return this.left;
30. }
31. public Node getRight() {
32. return this.right;
33. }
34. public void setLeft(Node n) {
35. this.left = n;
36. }
37. public void setRight(Node n) {
38. this.right = n;
39. }
40. }
41. class Tree {
42. private Node root;
43. public Node getRoot() {
44. return this.root;
45. }
46. public void setRoot(Node n) {
47. this.root = n;
48. }
49. public int height(Node n) {
50. return n == null ? 0 : n.getHeight();
51. }
52. public int max(int a, int b) {
53. return a > b ? a : b;
54. }
55. public int getBalance(Node n) {
56. return n == null ? 0 : (height(n.getLeft()) - height(n.getRight()));
57. }
58. public Node rotateLeft(Node x) {
59. Node y = x.getRight();
60. Node T2 = y.getLeft();
61. x.setRight(T2);
62. y.setLeft(x);
63. x.setHeight(1 + max(height(x.getLeft()), height(x.getRight())));
64. y.setHeight(1 + max(height(y.getLeft()), height(y.getRight())));
66. return y;
67. }
69. public Node rotateRight(Node y) {
70. Node x = y.getLeft();
71. Node T2 = x.getRight();
73. x.setRight(y);
74. y.setLeft(T2);
76. y.setHeight(1 + max(height(y.getLeft()), height(y.getRight())));
77. x.setHeight(1 + max(height(x.getLeft()), height(x.getRight())));
78. return x;
79. }
80. public Node insert(Node n, int d) {
81. if(n == null) {
82. return new Node(d);
83. }
84. if(d < n.getData()) {
85. n.setLeft(insert(n.getLeft(), d));
86. }
87. else if(d > n.getData()) {
88. n.setRight(insert(n.getRight(), d));
89. }
90. else {
91. return n;
92. }
93. n.setHeight(1 + max(height(n.getLeft()), height(n.getRight())));
94. int balance = getBalance(n);
96. if(balance > 1 && d < n.getLeft().getData()) {
97. return rotateRight(n);
98. }
100. if(balance < -1 && d > n.getRight().getData()) {
101. return rotateLeft(n);
102. }
104. if(balance > 1 && d > n.getLeft().getData()) {
105. n.setLeft(rotateLeft(n.getLeft()));
106. return rotateRight(n);
107. }
109. if(balance < -1 && d < n.getRight().getData()) {
110. n.setRight(rotateRight(n.getRight()));
111. return rotateLeft(n);
112. }
113. return n;
114. }
115. public void printTree() {
116. System.out.println("Tree Graph: ");
117. printTree("", root, false);
118. }
120. public void printTree(String prefix, Node n, boolean isLeft) {
121. if (n != null) {
122. printTree(prefix + "      ", n.getRight(), false);
123. System.out.println (prefix + ("|----[") + n.getData() + "]");
124. printTree(prefix + "      ", n.getLeft(), true);
125. }
126. }
127. }

Output Terminal



2. Cek AVL atau bukan

1. package Tree;
2. public class AVLCheck {
3. public static void main(String[] args) {
4. int[] data = {6,8,9,4,7,2,1,5,3};
6. BSTree tree = new BSTree();
8. for(int i = 0; i < data.length; i++) {
9. Node node = new Node(data[i]);
10. tree.addNode(node);
11. }
13. tree.checkAVL();
14. int[] data2 = {5,2,6,3,1,4,8,7,9};
16. BSTree tree2 = new BSTree();
17. for(int i = 0; i < data2.length; i++) {
18. Node node2 = new Node(data2[i]);
19. tree2.addNode(node2);
20. }
21. tree2.checkAVL();
22. }
23. }
24. class Node {
25. private int data, height;
26. private Node left, right;
27. public Node(int d) {
28. this.data = d;
29. this.height = 1;
30. }
31. public int getData() {
32. return this.data;
33. }
34. public int getHeight() {
35. return this.height;
36. }
37. public void setHeight(int h) {
38. this.height = h;
39. }
40. public Node getLeft() {
41. return this.left;
42. }
43. public Node getRight() {
44. return this.right;
45. }
46. public void setLeft(Node n) {
47. this.left = n;
48. }
49. public void setRight(Node n) {
50. this.right = n;
51. }
52. }
53. class BSTree {
54. private Node root;
55. private boolean isAVL = true;
56. public Node getRoot() {
57. return this.root;
58. }
59. public void addNode(Node n) {
60. if(root == null) {
61. root = n;
62. }
63. else {
64. insertNode(root, n);
65. }
66. }
67. public void insertNode(Node parent, Node n) {
68. if(parent.getData() > n.getData()) {
69. if(parent.getLeft() == null) {
70. parent.setLeft(n);
71. }
72. else {
73. insertNode(parent.getLeft(), n);
74. }
75. }
76. else {
77. if(parent.getRight() == null) {
78. parent.setRight(n);
79. }
80. else {
81. insertNode(parent.getRight(), n);
82. }
83. }
84. }
85. public int height(Node n) {
86. if(n == null) {
87. return -1;
88. }
89. else {
90. int leftHeight = height(n.getLeft());
91. int rightHeight = height(n.getRight());
92. return leftHeight > rightHeight ? leftHeight - 1 : rightHeight - 1;
93. }
94. }
95. public int max(int a, int b) {
96. return a > b ? a : b;
97. }
98. public Node rotateLeft(Node x) {
99. Node y = x.getRight();
100. Node T2 = y.getLeft();
101. y.setLeft(x);
102. x.setRight(T2);
103. x.setHeight(1 + max(height(x.getLeft()), height(x.getRight())));
104. y.setHeight(1 + max(height(y.getLeft()), height(y.getRight())));
106. return y;
107. }
109. public Node rotateRight(Node y) {
110. Node x = y.getLeft();
111. Node T2 = x.getRight();
113. x.setRight(y);
114. y.setLeft(T2);
116. y.setHeight(1 + max(height(y.getLeft()), height(y.getRight())));
117. x.setHeight(1 + max(height(x.getLeft()), height(x.getRight())));
118. return x;
119. }
120. public int getBalance(Node n) {
121. return n == null ? 0 : height(n.getLeft()) - height(n.getRight());
122. }
123. public Node insert(Node n, int d) {
124. if(n == null) {
125. return new Node(d);
126. }
127. if(d < n.getData()) {
128. n.setLeft(insert(n.getLeft(), d));
129. }
130. else if(d > n.getData()) {
131. n.setRight(insert(n.getRight(), d));
132. }
133. else {
134. return n;
135. }
136. n.setHeight(1 + max(height(n.getLeft()), height(n.getRight())));
137. int balance = getBalance(n);
139. if(balance > 1 && d < n.getLeft().getData()) {
140. return rotateRight(n);
141. }
143. if(balance < -1 && d > n.getRight().getData()) {
144. return rotateLeft(n);
145. }
147. if(balance > 1 && d > n.getLeft().getData()) {
148. n.setLeft(rotateLeft(n.getLeft()));
149. return rotateRight(n);
150. }
152. if(balance < -1 && d < n.getRight().getData()) {
153. n.setRight(rotateRight(n.getRight()));
154. return rotateLeft(n);
155. }
156. return n;
157. }
158. public boolean isAVL(Node n) {
159. if(n.getLeft() != null) {
160. isAVL(n.getLeft());
161. }
162. if(n.getRight() != null) {
163. isAVL(n.getRight());
164. }
165. if(getBalance(n) > 1 || getBalance(n) < -1) {
166. isAVL = false;
167. }
168. return isAVL;
169. }
171. public void checkAVL() {
172. boolean isAVL = this.isAVL(root);
173. printTree();
174. printLevelOrder();
175. System.out.println();
176. if(isAVL) {
177. System.out.println("This is an AVL Tree");
178. }
179. else {
180. System.out.println("This is not an AVL Tree");
181. }
182. System.out.println("\n- - - - - - - - - - - - -\n");
183. }
184. public void printTree() {
185. System.out.println("Tree Graph:");
186. printTreeFunction("", root, false);
187. }
189. public void printTreeFunction(String prefix, Node n, boolean isLeft) {
190. if (n != null) {
191. printTreeFunction(prefix + "      ", n.getRight(), false);
192. System.out.println (prefix + ("|----[") + n.getData() + "]");
193. printTreeFunction(prefix + "      ", n.getLeft(), true);
194. }
195. }
196. public void printLevelOrder() {
197. System.out.println("\nTree Level Order:");
198. for(int i = 1; i <= 99; i++) {
199. levelOrderFunction(root, i);
200. }
201. System.out.println();
202. }
204. public void levelOrderFunction(Node node, int level) {
205. if(node != null) {
206. if(level == 1){
207. System.out.print(node.getData() + " ");
208. }
209. else if(level > 1) {
210. levelOrderFunction(node.getLeft(), level - 1);
211. levelOrderFunction(node.getRight(), level - 1);
212. }
213. }
214. }
215. }

Output Terminal

