Project #2

Due Dates: Monday, October 17 at 11:59pm

Submit: eLearning

Late Policy: 24-hour late period, then zero

Instructions: This is an individual assignment. Answers should be your own work.

Introduction:

In this project you will modify the author's BinarySearchTree code

to implement some new methods.

Description:

Modify the author's BinarySearchTree code to implement the methods

shown below.

Each method is 10 points.

a) nodeCount

Recursively traverses the tree and returns the count of nodes.

b) isFull

Returns true if the tree is full. A full tree has every node as either a leaf or a parent with two children.

c) compareStructure

Compares the structure of current tree to another tree and returns

true if they match.

For example, these two trees have the same structure:

5 10

/ \ / \

3 8 5 15

/ \ / \

1 4 2 7

d) equals

Compares the current tree to another tree and returns true

if they are identical.

e) copy

Creates and returns a new tree that is a copy of the original tree.

f) mirror

Creates and returns a new tree that is a mirror image of the original tree. For example, for the tree on the left, the tree on the right is returned.(Note that the resulting tree will no longer be in search tree order.)

100 100

/ \ / \

50 150 --> 150 50

/ \

40 40

\ /



45 45

g) isMirror

Returns true if the tree is a mirror of the passed tree.

h) rotateRight

Performs a single rotation on the node having the passed value.

If a RotateRight on 100 is performed:



100 50



/ \ / \

50 150 --> 40 100



/ \ \



40 45 150



\



45



g) rotateLeft



As above but left rotation.



i) printLevels - performs a level-by-level printing of the tree.

j) main - demonstrate in your main method that all of your new methods work.

Submit to eLearning:

BinarySearchTree.java

/\*

private static class BinaryNode<AnyType>

{

// Constructors

BinaryNode(AnyType theElement)

{

this(theElement,null,null,null);

}

BinaryNode(AnyType theElement,BinaryNode<AnyType> lt,BinaryNode<AnyType> rt,BinaryNode<AnyType> pt)

{

element=theElement;

left =lt;

right =rt;

parent =pt;

}

AnyType element; // The data in the node

BinaryNode<AnyType> left; // Left child

BinaryNode<AnyType> right; // Right child

BinaryNode<AnyType> parent; // parent

}

private void insert(AnyType x)

{

BinaryNode<AnyType> currentNode = new BinaryNode<>(x,null,null,null);

BinaryNode<AnyType> parentNode = null;

BinaryNode<AnyType> newNode = root;

while (newNode != null)

{

parentNode = newNode;

if (currentNode.element.compareTo(newNode.element)<0)

{

newNode = newNode.left;

}

else

{

newNode = newNode.right;

}

}

currentNode.parent=parentNode;

if (parentNode == null)

{

root=currentNode;

}

else if (currentNode.element.compareTo(parentNode.element)<0)

{

parentNode.left=currentNode;

}

else

{

parentNode.right=currentNode;

}

}

public void rotateRight(AnyType x)

{

BinaryNode<AnyType> foundNode= (find(x,root));

if(foundNode.left!=null)

{

BinaryNode<AnyType> tempChild = foundNode.left;

if(tempChild.right!=null)

{

BinaryNode<AnyType> tempChildRight = tempChild.right;

foundNode.left=tempChildRight;

tempChildRight.parent=foundNode;

}

else

{

foundNode.left=null;

}

if(foundNode.parent==null)

{

root=tempChild;

tempChild.parent=null;

}

else if(foundNode.parent.left==foundNode)

{

foundNode.parent.left=tempChild;

tempChild.parent=foundNode.parent;

}

else

{

foundNode.parent.right=tempChild;

tempChild.parent=foundNode.parent;

}

tempChild.right=foundNode;

foundNode.parent=tempChild;

}

else

{

System.out.println("rotation not possible\n");

}

}

public void rotateLeft(AnyType x)

{

BinaryNode<AnyType> foundNode= (find(x,root));

if(foundNode.right!=null)

{

BinaryNode<AnyType> tempChild = foundNode.right;

if(tempChild.left!=null)

{

BinaryNode<AnyType> tempChildLeft = tempChild.left;

foundNode.right=tempChildLeft;

tempChildLeft.parent=foundNode;

}

else

{

foundNode.right=null;

}

if(foundNode.parent==null)

{

root=tempChild;

tempChild.parent=null;

}

else if(foundNode.parent.left==foundNode)

{

foundNode.parent.left=tempChild;

tempChild.parent=foundNode.parent;

}

else

{

foundNode.parent.right=tempChild;

tempChild.parent=foundNode.parent;

}

tempChild.left=foundNode;

foundNode.parent=tempChild;

}

else

{

System.out.println("rotation not possible\n");

}

}

\*/