Algorithm Analysis and Data Structures: Assignment 2 – Binary Search Tree

Write a program to implement the deletion from a binary search tree.

You first make a BST (algorithm given in class.)

DO NOT USE ARRAY REPRESENTATION FOR BST.

THE NODE structure must NOT have parent pointer.

Your tree must have at least 15 nodes.

1. Make the BST and do a inorder traversal

2. Now delete the nodes as follows:

     a. a leaf node.  Then do inorder traversal.

     b. a node that has a subtree.  Then do inorder traversal again.

Code:

import java.util.\*;

public class BinarySearchTree{

class Node{

int key;

Node left;

Node right;

Node(int value)

{

key = value;

left = right = null;

}

}

Node root = null;

public static void main(String[] args)

{

BinarySearchTree bst = new BinarySearchTree();

bst.InsertNewNode(21);

bst.InsertNewNode(28);

bst.InsertNewNode(18);

bst.InsertNewNode(16);

bst.InsertNewNode(15);

bst.InsertNewNode(17);

bst.InsertNewNode(12);

bst.InsertNewNode(19);

bst.InsertNewNode(23);

bst.InsertNewNode(32);

bst.InsertNewNode(29);

bst.InsertNewNode(25);

bst.InsertNewNode(22);

bst.InsertNewNode(24);

bst.InsertNewNode(27);

System.out.println("\nInorder traversal of Binary Search Tree");

bst.InorderTraversal(bst.root);

System.out.print("\n");

bst.DeleteLeafNode(bst.root);

System.out.println("\nInorder traversal of Binary Search Tree after deleting the leaf node");

bst.InorderTraversal(bst.root);

System.out.print("\n");

System.out.println("\nA node with sub-tree is "+ bst.root.key);

bst.DeleteSubTreeNode(bst.root, bst.root.key);

System.out.println("\nInorder traversal of Binary Search Tree after deleting a node that has sub-tree");

bst.InorderTraversal(bst.root);

System.out.println("\n");

}

// InsertNewNode - Inserts a new node

public void InsertNewNode(int value)

{

root = InsertNewNodeRecursion(root,value);

}

public Node InsertNewNodeRecursion(Node root, int value)

{

if(root == null)

{

root = new Node(value);

return root;

}

if(value < root.key)

root.left = InsertNewNodeRecursion(root.left,value);

else if(value > root.key)

root.right = InsertNewNodeRecursion(root.right, value);

return root;

}

// InorderTraversal - Prints all the nodes of the given Binary Search Tree in InOrder format

public void InorderTraversal(Node root)

{

if(root != null)

{

InorderTraversal(root.left);

System.out.print(root.key+" ");

InorderTraversal(root.right);

}

}

// DeleteLeafNode - Deletes a leaf node from the Binary Search Tree

public Node DeleteLeafNode(Node root)

{

if(root == null)

return root;

if(root.left != null)

root.left = DeleteLeafNode(root.left);

else if(root.right != null)

root.right = DeleteLeafNode(root.right);

else

{

System.out.println("\nLeaf Node that is going to be deleted is "+root.key);

root = null;

}

return root;

}

// DeleteSubTreeNode - Deletes a node which has a sub-tree

public Node DeleteSubTreeNode(Node root, int value)

{

Node successor = null;

if(root == null)

return root;

if(value < root.key)

root.left = DeleteSubTreeNode(root.left, value);

else if(value > root.key)

root.right = DeleteSubTreeNode(root.right, value);

else

{

if(root.right != null)

{

successor = GetSuccessor(root,true);

System.out.println("\nSuccessor is "+successor.key);

Delete(root,successor.key);

root.key = successor.key;

}

}

return root;

}

// GetSuccessor - Fetches the successor of the node to be deleted

public Node GetSuccessor(Node root, boolean bFirst)

{

Node successor = root;

if(bFirst)

root = (root.right != null) ?root.right: root;

successor = (root.left != null) ? GetSuccessor(root.left,false) : root;

return successor;

}

// Delete - Deletes the node from the tree

public Node Delete(Node root, int value)

{

if(root == null)

return root;

if(value < root.key)

root.left = Delete(root.left, value);

else if(value > root.key)

root.right = Delete(root.right,value);

else if(value == root.key)

root = null;

return root;

}

}

Instructions to compile:

1. Create a .java file in Sublime text or any text editor by the name BinarySearchTree.java
2. Place the code in BinarySearchTree.java and save
3. Open command prompt and redirect to the path where BinarySearchTree.java is saved
4. Use javac BinarySearchTree.java command, to compile the code
5. After successful compilation use java BinarySearchTree command, to run the code and see the output

Screen Shot of the output

