Cheewin Thawornjaroenpong

CSCI323.25 Designs and Analysis of Algorithms (Spring 2023)

Project4

The implementation of 2-3 trees insertion

03/19/2023

Algorithm Steps:
Step 1: inFile open with args[0] outFile open with args[1] deBugFile open with args[2]
Step 2: listHead get a new listNode with (“dummy”), as the dummy node for listHead to point to.
Step 3: constructLL (listHead, inFile, deBugFile)
Step 4: printList (listHead, outFile) // Print the complete list to outFile
Step 5: middleNode findMiddleNode (listHead, deBugFile)
Step 6: if middleNode != null // in case the list is empty outFile middleNode’s data // with caption “the word in the middle of list is”
Step 7: Close all files

Illustrations:

Source code:

```
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class ThawornjaroenpongC_Project4 {
	public static class treeNode{
		private int key1 = 0;
		private int key2 = 0;
		private int rank = 0;
		private treeNode child1 = null;
		private treeNode child2 = null;
		private treeNode child3 = null;
		private treeNode father = null;

		public treeNode(int key1, int key2, int rank,
				treeNode child1, treeNode child2, treeNode child3, treeNode
father)
		{
			this.key1 = key1;
			this.key2 = key2;
			this.rank = rank;
			this.child1 = child1;
			this.child2 = child2;
			this.child3 = child3;
			this.father = father;
		}

		public void printNode(treeNode Tnode, BufferedWriter outFile)
```

```
		{
			try {
				if(Tnode.father == null)
				{
					if(Tnode.child1 == null)
					{
						outFile.write("RootNode's (" + Tnode.key1 + ", " +
Tnode.key2 + ", " + Tnode.rank +
							", " + "null" + ", " + "null" + ", " + "null"
+ ", " + "null" + ")");
					}

					if(Tnode.child1 != null && Tnode.child2 != null &&
Tnode.child3 == null)
					{
						outFile.write("RootNode's (" + Tnode.key1 + ", " +
Tnode.key2 + ", " + Tnode.rank +
							", " + Tnode.child1.key1 + ", " +
Tnode.child2.key1 + ", " + "null" + ", " + "null" + ")");
					}

					if(Tnode.child1 != null && Tnode.child2 != null &&
Tnode.child3 != null)
					{
						outFile.write("RootNode's (" + Tnode.key1 + ", " +
Tnode.key2 + ", " + Tnode.rank +
							", " + Tnode.child1.key1 + ", " +
Tnode.child2.key1 + ", " + Tnode.child3.key1 + ", " + "null" + ")");
					}
				}

				if(Tnode.child1 == null && Tnode.father != null)
				{
					outFile.write("Tnode's (" + Tnode.key1 + ", " + Tnode.key2
+ ", " + Tnode.rank +
						", " + "null" + ", " + "null" + ", " + "null" + ", " +
Tnode.father.key1 + ")");
				}

				if(Tnode.child1 != null && Tnode.child2 != null && Tnode.child3 ==
null && Tnode.father != null)
				{
					outFile.write("Tnode's (" + Tnode.key1 + ", " + Tnode.key2
+ ", " + Tnode.rank +
```

```
                                                        ", " + Tnode.child1.key1 + ", " +
Tnode.child2.key1 + ", " + "null" + ", " + Tnode.father.key1 + ")");
                                }

                                if(Tnode.child1 != null && Tnode.child2 != null && Tnode.child3 !=
null && Tnode.father != null)
                                {
                                        outFile.write("Tnode's (" + Tnode.key1 + ", " + Tnode.key2
+ ", " + Tnode.rank +
                                                        ", " + Tnode.child1.key1 + ", " +
Tnode.child2.key1 + ", " + Tnode.child3.key1 + ", " + Tnode.father.key1 + ")");
                                }

                                outFile.write("\n");

                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }

                }
        }

        public static class Trees
        {
                private treeNode Root;
                public Trees()
                {

                }

                public treeNode initialTree(Scanner inFile, BufferedWriter deBugFile)
                {
                        try {

                                deBugFile.write("*********Entering initialTree() method**********
\n");

                                this.Root = new treeNode(-1, -1, -1, null, null, null, null);

                                int data1;
                                int data2;
                                data1 = inFile.nextInt();
                                data2 = inFile.nextInt();
```

```
                                deBugFile.write("********before swap data1 and data2 are " +
data1 + ", " + data2 + "********** \n");
                                if (data2 < data1)
                                {
                                        int temp = data1;
                                        data1 = data2;
                                        data2 = temp;
                                }
                                deBugFile.write("*******after swap data1 and data2 are " + data1 +
", " + data2 + "********** \n");
                                treeNode newNode1 = new treeNode(data1, -1, 1, null, null, null,
this.Root);
                                treeNode newNode2 = new treeNode(data2, -1, 2, null, null, null,
this.Root);
                                this.Root.child1 = newNode1;
                                this.Root.child2 = newNode2;
                                this.Root.key1 = data2;
                                this.Root.printNode(this.Root, deBugFile);
                                deBugFile.write("*********Exiting initialTree() method********** \n");

                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }

                        return this.Root;
                }


                public void build23Tree (Scanner inFile, treeNode root, BufferedWriter deBugFile)
                {
                        try {
                                this.Root = root;
                                deBugFile.write("********Entering build23Tree() method**********
\n");
                                int data = 0;
                                treeNode Spot = new  treeNode(data, -1, 5, null, null, null, null);
                                while(inFile.hasNext())
                                {
                                        data = Integer.parseInt(inFile.next());
                                        Spot = this.findSpot(this.Root, data, deBugFile);
                                        while(Spot == null && inFile.hasNext())
                                        {
                                                data = Integer.parseInt(inFile.next());
```

```
                                        Spot = this.findSpot(this.Root, data, deBugFile);
                                        if(data == 19)
                                        {
                                                deBugFile.write("Data = 19 \n");
                                        }
                                }

                                if(Spot != null)
                                {
                                        if(data == 19)
                                        {
                                                deBugFile.write("Data = 19 \n");
                                        }
                                        deBugFile.write("********In build23Tree; printing
Spot info******** \n");
                                        Spot.printNode(Spot, deBugFile);
                                        treeNode leafNode = new  treeNode(data, -1, 5,
null, null, null, null);
                                        this.treeInsert(Spot, leafNode, deBugFile);
                                }

                        }

                        /*
                                Spot = this.findSpot(this.Root, data, deBugFile);
                                deBugFile.write("********In build23Tree; printing Spot
info******** \n");
                                Spot.printNode(Spot, deBugFile);
                                treeNode leafNode = new  treeNode(data, -1, 5, null, null,
null, null);
                                this.treeInsert(Spot, leafNode, deBugFile);

                        */
                        deBugFile.write("******In build23Tree; printing preOrder() after one
treeInsert********* \n");
                        this.preOrder(this.Root, deBugFile);



                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }
```

```
	public treeNode findSpot(treeNode Spot, int data, BufferedWriter deBugFile)
	{
		try {

			deBugFile.write("**********Entering findSpot() method********* \n");
			deBugFile.write("Spot's key1 and key2 and data are " + Spot.key1
+ ", " + Spot.key2 + ", " + data + "\n");

			if(Spot.child1 == null)
			{
				deBugFile.write("In findSpot() You are at the leaf level, you
are too far down the tree!! \n");
			}
			if(data == Spot.key1 || data == Spot.key2)
			{
				deBugFile.write("In findSpot(): data is already in Spot's
keys, no need to search further!! \n");
				return null;
			}
			if(Spot.child1.child1 == null)
			{
				if(data == Spot.child1.key1 || data == Spot.child2.key1)
				{
					deBugFile.write("*****in findSpot(): data is already
in a leaf node.****** \n");
					return null;
				}else {
					return Spot;
				}
			}
			else
			{
				if(data < Spot.key1)
				{
					return findSpot(Spot.child1, data, deBugFile);
				}
				else if(Spot.key2 == -1 || data < Spot.key2)
				{
					return findSpot(Spot.child2, data, deBugFile);
				}
				else if(Spot.key2 != -1 && data >= Spot.key2)
				{
					return findSpot(Spot.child3, data, deBugFile);
```

```
                                        }
                                        else
                                        {
                                                deBugFile.write("*******in findSpot(), something is
wrong about data****** \n");
                                                return null;
                                        }
                                }
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                        return null;
                }

                public void treeInsert(treeNode Spot, treeNode newNode, BufferedWriter
deBugFile)
                {
                        try {
                                int count = 0;
                                deBugFile.write("********Entering treeInsert() method******** \n");
                                if(Spot == null)
                                {
                                        deBugFile.write("*******in treeInsert(), Spot is null,
something is wrong****** \n");
                                        return;
                                }
                                else
                                {
                                        deBugFile.write("*****In treeInsert(). Printing Spot and
newNode info******* \n");
                                        Spot.printNode(Spot, deBugFile);
                                        newNode.printNode(newNode, deBugFile);
                                }

                                if(Spot.key2 == -1)
                                {
                                        count = 2;
                                }
                                else
                                {
                                        count = 3;
                                }
```

```
                    deBugFile.write("In treeInsert() method; Spot kids count is " +
count + "\n");

                    if(count == 2)
                    {
                        this.spotHas2kidsCase(Spot, newNode, deBugFile);
                    }
                    else if (count == 3)
                    {
                        this.spotHas3kidsCase(Spot, newNode, deBugFile);
                    }

                    deBugFile.write("*********Leaving treeInsert() method********* \n");

                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }

            public void spotHas2kidsCase(treeNode Spot, treeNode newNode,
BufferedWriter deBugFile)
            {
                try {
                    deBugFile.write("*******Entering spotHas2kidCase() method*****
\n");
                    deBugFile.write("In spotHas2kidCase() method; Spot's rank is " +
Spot.rank+ "\n");
                    if(newNode.key1 < Spot.child2.key1)
                    {
                        Spot.child3 = Spot.child2;
                        Spot.child2 = newNode;
                    }
                    else
                    {
                        Spot.child3 = newNode;
                    }


                    if(Spot.child2.key1 < Spot.child1.key1)
                    {
                        treeNode tmpNode = Spot.child1;
                        Spot.child1 = Spot.child2;
                        Spot.child2 = tmpNode;
```

```
                                }

                                Spot.child1.father = Spot;
                                Spot.child1.rank = 1;
                                Spot.child2.father = Spot;
                                Spot.child2.rank = 2;
                                Spot.child3.father = Spot;
                                Spot.child3.rank = 3;

                                this.updateKeys(Spot, deBugFile);

                                if(Spot.rank > 1)
                                {
                                        this.updateKeys(Spot.father, deBugFile);
                                }

                                deBugFile.write("***********Leaving spotHas2kidCase()
method********** \n");
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }

                }

                public void spotHas3kidsCase(treeNode Spot, treeNode newNode,
BufferedWriter deBugFile)
                {
                        try {
                                deBugFile.write("*********Entering spotHas3kidCase()
method**********  \n");
                                deBugFile.write("In spotHas3kidCase() method; Spot's rank is " +
Spot.rank+ "\n");

                                treeNode sibling = new treeNode(-1, -1, 5, null, null, null,null);
                                if(newNode.key1 > Spot.child3.key1)
                                {
                                        sibling.child2 = newNode;
                                        sibling.child1 = Spot.child3;
                                        Spot.child3 = null;
                                }
                                else if(newNode.key1 < Spot.child3.key1)
                                {
                                        sibling.child2 = Spot.child3;
```

```
        Spot.child3 = newNode;
}

if(Spot.child3 != null)
{
        if(Spot.child3.key1 > Spot.child2.key1)
        {
                sibling.child1 = Spot.child3;
                Spot.child3 = null;
        }
        else
        {
                sibling.child1 =  Spot.child3;
                Spot.child3 = newNode;
        }
}
else if (Spot.child2.key1 < Spot.child1.key1)
{
        treeNode tmpNode = Spot.child1;
        Spot.child1 = Spot.child2;
        Spot.child2 = tmpNode;
}

//Spot
Spot.child1.father = Spot;
Spot.child1.rank = 1;

Spot.child2.father = Spot;
Spot.child2.rank = 2;

Spot.child3 = null;

//sibling
sibling.child1.father = sibling;
sibling.child1.rank = 1;


sibling.child2.father = sibling;
sibling.child2.rank = 2;

sibling.child3 = null;

this.updateKeys(Spot, deBugFile);
this.updateKeys(sibling, deBugFile);
```

```
                        if(Spot.rank == -1 && Spot.father == null)
                        {
                                this.Root = this.makeNewRoot(Spot, sibling, deBugFile);
                        }
                        else
                        {
                                treeInsert(Spot.father, sibling, deBugFile);
                        }

                        if(Spot.rank > 1)
                        {
                                this.updateKeys(Spot.father, deBugFile);
                        }

                        deBugFile.write("*****Leaving spitHas3kidCase() method********
\n");
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }

        public treeNode makeNewRoot(treeNode Spot, treeNode Sibling, BufferedWriter
deBugFile)
        {
                try {

                        deBugFile.write("******Entering makeNewRoot() method****** \n");
                        treeNode newRoot = new treeNode(-1, -1, -1, null, null, null, null);
                        newRoot.child1 = Spot;
                        newRoot.child2 = Sibling;
                        newRoot.child3 = null;
                        newRoot.key1 = this.findMinLeaf(Sibling);
                        newRoot.key2 = -1;
                        Spot.father = newRoot;
                        Spot.rank = 1;
                        Sibling.father = newRoot;
                        Sibling.rank = 2;
                        deBugFile.write("********Leaving makeNewRoot() method********
\n");

                        return newRoot;
```

```
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		return null;

	}

	public int findMinLeaf(treeNode Tnode)
	{
		if(Tnode == null)
		{
			return -1;
		}
		if(Tnode.child1 ==  null)
		{
			return Tnode.key1;
		}
		else
		{
			return findMinLeaf(Tnode.child1);
		}
	}

	public void updateKeys(treeNode Tnode, BufferedWriter deBugFile)
	{
		try {
			deBugFile.write("****Entering updateKeys() method****** \n");


			if(Tnode == null)
			{
				return;
			}

			deBugFile.write("In updateKeys Key1 and Key2 are " +
Tnode.key1 + ", " + Tnode.key2 + "\n");
			Tnode.key1 = this.findMinLeaf(Tnode.child2);
			Tnode.key2 = this.findMinLeaf(Tnode.child3);

			if(Tnode.rank > 1)
			{
				this.updateKeys(Tnode.father, deBugFile);
			}
```

```
                    deBugFile.write("********Leaving updateKeys() method****** \n");
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
        }

        public void preOrder(treeNode Tnode, BufferedWriter outFile)
        {
                if(Tnode.child1 == null)
                {
                    Tnode.printNode(Tnode, outFile);
                }
                else if(Tnode.child1 != null && Tnode.child3 == null)
                {
                    Tnode.printNode(Tnode, outFile);
                    preOrder(Tnode.child1, outFile);
                    preOrder(Tnode.child2, outFile);
                }
                else if(Tnode.child1 != null && Tnode.child3 != null)
                {
                    Tnode.printNode(Tnode, outFile);
                    preOrder(Tnode.child1, outFile);
                    preOrder(Tnode.child2, outFile);
                    preOrder(Tnode.child3, outFile);
                }
        }

        public treeNode getRoot()
        {
                return this.Root;
        }
}

public static void main(String[] args) {

        try {
                Scanner inFile = new Scanner(new FileReader(args[0]));
                BufferedWriter treeFile = new BufferedWriter(new FileWriter(args[1]));
           BufferedWriter deBugFile = new BufferedWriter(new FileWriter(args[2]));

           Trees TwoThreeTree = new Trees();
           treeNode root = TwoThreeTree.initialTree(inFile, deBugFile);
           TwoThreeTree.build23Tree(inFile, root, deBugFile);
```

```
                root = TwoThreeTree.getRoot();
                TwoThreeTree.preOrder(root, treeFile);

                //inFile.close();
                treeFile.close();
                deBugFile.close();
            } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }



        }

}
```

Program output:

Data1:

RootNode's (8, 16, -1, 5, 12, 19, null)
Tnode's (5, 7, 1, 2, 5, 7, 8)
Tnode's (2, -1, 1, null, null, null, 5)
Tnode's (5, -1, 2, null, null, null, 5)
Tnode's (7, -1, 3, null, null, null, 5)
Tnode's (12, -1, 2, 8, 12, null, 8)
Tnode's (8, -1, 1, null, null, null, 12)
Tnode's (12, -1, 2, null, null, null, 12)
Tnode's (19, -1, 3, 16, 19, null, 8)
Tnode's (16, -1, 1, null, null, null, 19)
Tnode's (19, -1, 2, null, null, null, 19)

Data2 :

RootNode's (16, -1, -1, 2, 36, null, null)
Tnode's (2, -1, 1, 4, 8, null, 16)
Tnode's (4, -1, 1, 7, 10, null, 2)
Tnode's (7, -1, 1, 1, 7, null, 4)
Tnode's (1, -1, 1, null, null, null, 7)
Tnode's (7, -1, 2, null, null, null, 7)
Tnode's (10, -1, 2, 4, 10, null, 4)
Tnode's (4, -1, 1, null, null, null, 10)
Tnode's (10, -1, 2, null, null, null, 10)
Tnode's (8, -1, 2, 5, 11, null, 2)
Tnode's (5, 6, 1, 2, 5, 6, 8)
Tnode's (2, -1, 1, null, null, null, 5)
Tnode's (5, -1, 2, null, null, null, 5)
Tnode's (6, -1, 3, null, null, null, 5)
Tnode's (11, 12, 2, 8, 11, 12, 8)
Tnode's (8, -1, 1, null, null, null, 11)
Tnode's (11, -1, 2, null, null, null, 11)
Tnode's (12, -1, 3, null, null, null, 11)
Tnode's (36, -1, 2, 19, 55, null, 16)
Tnode's (19, -1, 1, 18, 25, null, 36)
Tnode's (18, -1, 1, 16, 18, null, 19)
Tnode's (16, -1, 1, null, null, null, 18)
Tnode's (18, -1, 2, null, null, null, 18)
Tnode's (25, 33, 2, 19, 25, 33, 19)
Tnode's (19, -1, 1, null, null, null, 25)
Tnode's (25, -1, 2, null, null, null, 25)

Tnode's (33, -1, 3, null, null, null, 25)
Tnode's (55, -1, 2, 44, 66, null, 36)
Tnode's (44, -1, 1, 36, 44, null, 55)
Tnode's (36, -1, 1, null, null, null, 44)
Tnode's (44, -1, 2, null, null, null, 44)
Tnode's (66, 72, 2, 55, 66, 72, 55)
Tnode's (55, -1, 1, null, null, null, 66)
Tnode's (66, -1, 2, null, null, null, 66)
Tnode's (72, -1, 3, null, null, null, 66)

debugFile from Data1 :

*********Entering initialTree() method**********
********before swap data1 and data2 are 7, 2**********
*******after swap data1 and data2 are 2, 7**********
RootNode's (7, -1, -1, 2, 7, null, null)
*********Exiting initialTree() method**********
********Entering build23Tree() method**********
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 7, -1, 12
********In build23Tree; printing Spot info********
RootNode's (7, -1, -1, 2, 7, null, null)
********Entering treeInsert() method********
*****In treeInsert(). Printing Spot and newNode info*******
RootNode's (7, -1, -1, 2, 7, null, null)
RootNode's (12, -1, 5, null, null, null, null)
In treeInsert() method; Spot kids count is 2
*******Entering spotHas2kidCase() method*****
In spotHas2kidCase() method; Spot's rank is -1
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 7, -1
********Leaving updateKeys() method******
**********Leaving spotHas2kidCase() method**********
*********Leaving treeInsert() method*********
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 7, 12, 8
********In build23Tree; printing Spot info********
RootNode's (7, 12, -1, 2, 7, 12, null)
********Entering treeInsert() method********
*****In treeInsert(). Printing Spot and newNode info*******
RootNode's (7, 12, -1, 2, 7, 12, null)
RootNode's (8, -1, 5, null, null, null, null)
In treeInsert() method; Spot kids count is 3
*********Entering spotHas3kidCase() method**********
In spotHas3kidCase() method; Spot's rank is -1
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 7, 12
********Leaving updateKeys() method******
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are -1, -1
****Entering updateKeys() method******
********Leaving updateKeys() method******
******Entering makeNewRoot() method******
********Leaving makeNewRoot() method********
*****Leaving spitHas3kidCase() method********

*********Leaving treeInsert() method*********
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 8, -1, 7
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 7, -1, 7
In findSpot(): data is already in Spot's keys, no need to search further!!
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 8, -1, 19
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 12, -1, 19
Data = 19
Data = 19
********In build23Tree; printing Spot info********
Tnode's (12, -1, 2, 8, 12, null, 8)
********Entering treeInsert() method********
*****In treeInsert(). Printing Spot and newNode info*******
Tnode's (12, -1, 2, 8, 12, null, 8)
RootNode's (19, -1, 5, null, null, null, null)
In treeInsert() method; Spot kids count is 2
*******Entering spotHas2kidCase() method*****
In spotHas2kidCase() method; Spot's rank is 2
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 12, -1
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 8, -1
********Leaving updateKeys() method******
********Leaving updateKeys() method******
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 8, -1
********Leaving updateKeys() method******
**********Leaving spotHas2kidCase() method**********
*********Leaving treeInsert() method*********
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 8, -1, 5
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 7, -1, 5
********In build23Tree; printing Spot info********
Tnode's (7, -1, 1, 2, 7, null, 8)
********Entering treeInsert() method********
*****In treeInsert(). Printing Spot and newNode info*******
Tnode's (7, -1, 1, 2, 7, null, 8)
RootNode's (5, -1, 5, null, null, null, null)
In treeInsert() method; Spot kids count is 2
*******Entering spotHas2kidCase() method*****

In spotHas2kidCase() method; Spot's rank is 1
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 7, -1
********Leaving updateKeys() method******
**********Leaving spotHas2kidCase() method**********
*********Leaving treeInsert() method*********
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 8, -1, 16
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 12, 19, 16
********In build23Tree; printing Spot info********
Tnode's (12, 19, 2, 8, 12, 19, 8)
********Entering treeInsert() method********
*****In treeInsert(). Printing Spot and newNode info*******
Tnode's (12, 19, 2, 8, 12, 19, 8)
RootNode's (16, -1, 5, null, null, null, null)
In treeInsert() method; Spot kids count is 3
*********Entering spotHas3kidCase() method***********
In spotHas3kidCase() method; Spot's rank is 2
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 12, 19
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 8, -1
********Leaving updateKeys() method******
********Leaving updateKeys() method******
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are -1, -1
****Entering updateKeys() method******
********Leaving updateKeys() method******
********Entering treeInsert() method********
*****In treeInsert(). Printing Spot and newNode info*******
RootNode's (8, -1, -1, 5, 12, null, null)
RootNode's (19, -1, 5, 16, 19, null, null)
In treeInsert() method; Spot kids count is 2
*******Entering spotHas2kidCase() method*****
In spotHas2kidCase() method; Spot's rank is -1
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 8, -1
********Leaving updateKeys() method******
**********Leaving spotHas2kidCase() method**********
*********Leaving treeInsert() method*********
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 8, 16
********Leaving updateKeys() method******

*****Leaving spitHas3kidCase() method********
*********Leaving treeInsert() method*********
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 8, 16, 8
In findSpot(): data is already in Spot's keys, no need to search further!!
******In build23Tree; printing preOrder() after one treeInsert*********
RootNode's (8, 16, -1, 5, 12, 19, null)
Tnode's (5, 7, 1, 2, 5, 7, 8)
Tnode's (2, -1, 1, null, null, null, 5)
Tnode's (5, -1, 2, null, null, null, 5)
Tnode's (7, -1, 3, null, null, null, 5)
Tnode's (12, -1, 2, 8, 12, null, 8)
Tnode's (8, -1, 1, null, null, null, 12)
Tnode's (12, -1, 2, null, null, null, 12)
Tnode's (19, -1, 3, 16, 19, null, 8)
Tnode's (16, -1, 1, null, null, null, 19)
Tnode's (19, -1, 2, null, null, null, 19)

debugFile from Data2 :

The debugFile for Data2 contains more than 6 pages.

```
*********Entering initialTree() method**********
********before swap data1 and data2 are 7, 2**********
*******after swap data1 and data2 are 2, 7**********
RootNode's (7, -1, -1, 2, 7, null, null)
*********Exiting initialTree() method*********
********Entering build23Tree() method**********
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 7, -1, 12
********In build23Tree; printing Spot info********
RootNode's (7, -1, -1, 2, 7, null, null)
********Entering treeInsert() method********
*****In treeInsert(). Printing Spot and newNode info*******
RootNode's (7, -1, -1, 2, 7, null, null)
RootNode's (12, -1, 5, null, null, null, null)
In treeInsert() method; Spot kids count is 2
*******Entering spotHas2kidCase() method*****
In spotHas2kidCase() method; Spot's rank is -1
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 7, -1
********Leaving updateKeys() method******
**********Leaving spotHas2kidCase() method**********
*********Leaving treeInsert() method*********
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 7, 12, 8
********In build23Tree; printing Spot info********
RootNode's (7, 12, -1, 2, 7, 12, null)
********Entering treeInsert() method********
*****In treeInsert(). Printing Spot and newNode info*******
RootNode's (7, 12, -1, 2, 7, 12, null)
RootNode's (8, -1, 5, null, null, null, null)
In treeInsert() method; Spot kids count is 3
*********Entering spotHas3kidCase() method**********
In spotHas3kidCase() method; Spot's rank is -1
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 7, 12
********Leaving updateKeys() method******
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are -1, -1
****Entering updateKeys() method******
********Leaving updateKeys() method******
******Entering makeNewRoot() method******
```

********Leaving makeNewRoot() method********
*****Leaving spitHas3kidCase() method********
*********Leaving treeInsert() method*********
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 8, -1, 7
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 7, -1, 7
In findSpot(): data is already in Spot's keys, no need to search further!!
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 8, -1, 19
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 12, -1, 19
Data = 19
Data = 19
********In build23Tree; printing Spot info********
Tnode's (12, -1, 2, 8, 12, null, 8)
********Entering treeInsert() method********
*****In treeInsert(). Printing Spot and newNode info*******
Tnode's (12, -1, 2, 8, 12, null, 8)
RootNode's (19, -1, 5, null, null, null, null)
In treeInsert() method; Spot kids count is 2
*******Entering spotHas2kidCase() method*****
In spotHas2kidCase() method; Spot's rank is 2
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 12, -1
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 8, -1
********Leaving updateKeys() method******
********Leaving updateKeys() method******
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 8, -1
********Leaving updateKeys() method******
**********Leaving spotHas2kidCase() method**********
*********Leaving treeInsert() method*********
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 8, -1, 5
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 7, -1, 5
********In build23Tree; printing Spot info********
Tnode's (7, -1, 1, 2, 7, null, 8)
********Entering treeInsert() method********
*****In treeInsert(). Printing Spot and newNode info*******
Tnode's (7, -1, 1, 2, 7, null, 8)
RootNode's (5, -1, 5, null, null, null, null)

In treeInsert() method; Spot kids count is 2
*******Entering spotHas2kidCase() method*****
In spotHas2kidCase() method; Spot's rank is 1
****Entering updateKeys() method******
In updateKeys Key1 and Key2 are 7, -1
********Leaving updateKeys() method******
***********Leaving spotHas2kidCase() method**********
*********Leaving treeInsert() method*********
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 8, -1, 16
**********Entering findSpot() method*********
Spot's key1 and key2 and data are 12, 19, 16