CS 3310 – Data and File Structures, Instructor: *<Gupta>*, Western Michigan University Lab TA: *<YG>*

Parker Loomis

**SOFTWARE LIFE CYCLE REPORT – FOR LAB ASSIGNMENT** *5*

**Trees and Binary Search Trees**

**PHASE 1: SPECIFICATION (“What do we build?”)**

1. Derive the relationships among height h, total number of nodes n, number of leaves L, and the number of edges m in a complete quad tree T. In particular, given h, what is n? Given n, what is h? Given m, what is h? Given m, what is n? Given L, what is h? Given h, what is L? Show all your work otherwise no credit.
2. Repeat question 1 if T is a full quad-tree.
3. Describe an implicit representation of a quad-tree T using arrays. Give the declaration of the array that stores the data of the nodes of T. Assume that the data consists of three fields: ssid, name, phoneNumber. Given a node v of T at index i, what are the indices of the four children of v and the parent of v?
4. Let T be a (unbalanced) binary search tree:
   1. A. Insert the sequence 65, 70, 60, 72, 87, 40, 35, 90, 75, 63, 68, 69, 61, 20, 25, 28, 37 (of integer keys) into T. Show the tree after each insert operation.
   2. B. Show the output of Preorder, Inorder and Postorder traversals of T after last insertion (i.e., after key 37 is inserted).
   3. C. Delete the sequence 28, 72, 65, 35, 63, 87, 70 from T. Show T after each deletion. Note that to be consistent in deletion, follow the same algorithm when you have to “borrow” a key to fill a hole.
5. 5. Design, develop and implement an object-oriented application to build an unbalanced binary search tree T using an array based implicit representation of T and starting with an empty tree T. The object representing T should support at least the following operations: *insert(mydata x), delete (mydata x), search (mydata x), preorderTraversal(), inorderTraversal( ), and postorderTraversal( )*. mydata is a record containing following three fields: stuName (a string of at most 15 characters, first and last name separated by a colon ‘:’)), courseNumber(an integer), grade (a char). Search key in T is based on stuName. Use recursion to implement the three traversals. Use dynamic allocation of array elements whenever possible (hint: c# and Java users can use arraylist).

**PHASE 2: DESIGN**

**Program should have 7 classes**

**1. Main method**

**2. Binary Search Tree**

**3. Node**

**4. MyComp**

**5. MyIterator**

**PHASE 3: RISK ANALYSIS (“What can go wrong, and how bad can it be?”)**

You could delete the wrong node and ruin the entire tree. You could also insert a node in the wrong place and mess it up as well.

**PHASE 4: VERIFICATION (“Are the algorithms correct?”)**

I hope they are. It seems to work correctly.

**PHASE 5: CODING**

package assignment5;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws FileNotFoundException {

readFile();

}

public static void readFile() throws FileNotFoundException {

BinaryTree<String> tree = new BinaryTree<String>();

@SuppressWarnings("resource")

Scanner fileScanner = new Scanner(new File("hw5cs3310F16data.txt"));// Scanner to read file

while (fileScanner.hasNext()) {

String instruction = fileScanner.next();

System.out.println(instruction);

if (instruction.equals("Preorder")) {

tree.preOrderTraversal();

/\*

\* calls the preorder method

\*/

System.out.println("\n");// adds a line to make it look nicer for the user

} else if (instruction.equals("Inorder")) {

tree.inOrderTraversal();

/\*

\* calls the inorder method

\*/

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

}

else if (instruction.equals("Postorder")) {

tree.preOrderTraversal();

/\*

\* calls the postorder method

\*/

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

} else if (instruction.equals("Insert:")) {

String add = fileScanner.nextLine();

System.out.println("Adding" + add);// shows user what is going on

System.out.println();

String[] parts = add.split(",");

tree.insert(parts[0]);

/\*

\* calls the insert method

\*/

} else if (instruction.equals("Search:")) {// searches the tree

String searchThis = fileScanner.nextLine();

System.out.println("Search for " + searchThis);// shows user what is going on

System.out.println("Found the node " + tree.search(searchThis));

/\*

\* calls the search method

\*/

System.out.println("");

} else if (instruction.equals("Delete:")) {

String delete = fileScanner.nextLine();

System.out.println("Deleteing" + delete);// shows user what is going on

tree.delete(delete);

/\*

\* calls the delete method

\*/

System.out.println("");

}

}

System.out.println();

System.out.println("Here is the tree");

System.out.println(tree.toString());

}

}

package assignment5;

import java.util.Comparator;

import java.util.Iterator;

import java.util.Stack;

public class BinaryTree<T extends Comparable<T>> implements Iterable<T> {

private Node<T> root;

private Comparator<T> compare;

public BinaryTree() {

root = null;

compare = null;

}

public BinaryTree(Comparator<T> comp) {

root = null;

compare = comp;

}

private int compare(T p, T n) {

if (compare == null)

return p.compareTo(n);

else

return compare.compare(p, n);

/\*

\* campares nodes

\*/

}

public void insert(T data) {

root = insert(root, data);

//calls insert ,ethod

}

private Node<T> insert(Node<T> p, T insert) {

/\*

\* this method inserts the node into the tree

\*/

if (p == null)

return new Node<T>(insert);

if (compare(insert, p.data) == 0)

return p;

if (compare(insert, p.data) < 0)

p.left = insert(p.left, insert);

else

p.right = insert(p.right, insert);

return p;

}

public boolean search(T toSearch) {

return search(root, toSearch);

}

private boolean search(Node<T> p, T toSearch) {

/\*

\* searches to find the desired node and returns a boolean

\*/

if (p == null)

return false;

else if (compare(toSearch, p.data) == 0)

return true;

else if (compare(toSearch, p.data) < 0)

return search(p.left, toSearch);

else

return search(p.right, toSearch);

}

public void delete(T toDelete) {

root = delete(root, toDelete);//calls the delete method

}

private Node<T> delete(Node<T> p, T toDelete) {

/\*

\* searches tree and deletes the desired node

\*/

if (p == null)

System.out.println("This node doesn't exist in the tree");

else if (compare(toDelete, p.data) < 0)

p.left = delete(p.left, toDelete);

else if (compare(toDelete, p.data) > 0)

p.right = delete(p.right, toDelete);

else {

if (p.left == null)

return p.right;

else if (p.right == null)

return p.left;

else {

p.data = retrieveData(p.left);// get data from the rightmost node in the left subtree

p.left = delete(p.left, p.data);// delete the rightmost node in the left subtree

}

}

return p;

}

private T retrieveData(Node<T> p) {

while (p.right != null)

p = p.right;

return p.data;

}

public String toString() {

/\*

\* Makes it into one string

\*/

StringBuffer buff = new StringBuffer();

for (T data : this)

buff.append(data.toString());

return buff.toString();

}

public void preOrderTraversal() {

preOrder(root);//calls the preOrder method

}

private void preOrder(Node p) {

/\*

\* files through tree recursivley and prints the tree in preOrder

\*/

if (p != null) {

System.out.print(p + " ");

preOrder(p.left);

preOrder(p.right);

}

}

public void inOrderTraversal() {

inOrder(root);//calls the inOrder method

}

private void inOrder(Node p) {

/\*

\* files through tree recursivley and prints the tree in inOrder

\*/

if (p != null) {

inOrder(p.left);

System.out.print(p + " ");

inOrder(p.right);

}

}

public void postOrderTraversal() {

postOrder(root);//calls the postOrder method

}

private void postOrder(Node p) {

/\*

\* files through tree recursivley and prints the tree in postOrder

\*/

if (p != null) {

postOrder(p.left);

System.out.print(p + " ");

postOrder(p.right);

}

}

public Iterator<T> iterator() {

return new MyIterator();

}

private class MyIterator implements Iterator<T> {

Stack<Node<T>> stack = new Stack<Node<T>>();

public MyIterator() {

if (root != null)

stack.push(root);

}

public boolean hasNext() {

return !stack.isEmpty();//sees if it has any children

}

public T next() {

Node<T> cur = stack.peek();

if (cur.left != null) {

stack.push(cur.left);

} else {

Node<T> tmp = stack.pop();

while (tmp.right == null) {

if (stack.isEmpty())

return cur.data;

tmp = stack.pop();

}

stack.push(tmp.right);

}

return cur.data;

}

public void remove() {

}

}

private class Node<T> {

/\*

\* Node class that is binary

\*/

private T data;

private Node<T> left, right;

public Node(T data, Node<T> l, Node<T> r) {

left = l;

right = r;

this.data = data;

}

public Node(T data) {

this(data, null, null);

}

public String toString() {

return data.toString();

}

}

}

class MyComp implements Comparator<Integer> {

public int compare(Integer x, Integer y) {

return y - x;

}

}

**PHASE 6: TESTING (“Did we build it correctly?”)**

Insert:

Adding John:Doe, 3310, A

Delete:

Deleteing John:Doe

Postorder

Search:

Search for John:Doe

Found the node false

Insert:

Adding Jane:Dane, 1120, B

Delete:

Deleteing June:Doe

This node doesn't exist in the tree

Delete:

Deleteing John:Doe

This node doesn't exist in the tree

Insert:

Adding Aj:Gup, 3310, B

Insert:

Adding Foo:Done, 2240, C

Inorder

Aj:Gup Foo:Done Jane:Dane

Preorder

Jane:Dane Aj:Gup Foo:Done

Insert:

Adding Aj:Gupt, 3310, D

Postorder

Jane:Dane Aj:Gup Foo:Done Aj:Gupt

Search:

Search for John:Doe

Found the node false

Search:

Search for Jane:Dane

Found the node true

Inorder

Aj:Gup Aj:Gupt Foo:Done Jane:Dane

Insert:

Adding Maria:Anders, 2050,B

Insert:

Adding Ana:Trujillo, 1120, B

Delete:

Deleteing June:Doe

This node doesn't exist in the tree

Delete:

Deleteing Maria:Anders

Insert:

Adding Antonio:Moreno, 3310, B

Insert:

Adding Foo:Done, 2240, C

Inorder

Aj:Gup Aj:Gupt Ana:Trujillo Antonio:Moreno Foo:Done Jane:Dane

Insert:

Adding Antonio:Morenot, 3310, D

Postorder

Jane:Dane Aj:Gup Foo:Done Aj:Gupt Ana:Trujillo Antonio:Moreno Antonio:Morenot

Search:

Search for Maria:Anders

Found the node false

Search:

Search for Ana:Trujillo

Found the node true

Inorder

Aj:Gup Aj:Gupt Ana:Trujillo Antonio:Moreno Antonio:Morenot Foo:Done Jane:Dane

Insert:

Adding John:Doe, 2050,B

Insert:

Adding Christina:Berglund, 1120, B

Preorder

Jane:Dane Aj:Gup Foo:Done Aj:Gupt Ana:Trujillo Antonio:Moreno Antonio:Morenot Christina:Berglund John:Doe

Delete:

Deleteing June:Doe

This node doesn't exist in the tree

Delete:

Deleteing John:Doe

Insert:

Adding Thomas:Hardy, 3310, B

Insert:

Adding Foo:Done, 2240, C

Preorder

Jane:Dane Aj:Gup Foo:Done Aj:Gupt Ana:Trujillo Antonio:Moreno Antonio:Morenot Christina:Berglund Thomas:Hardy

Insert:

Adding Thomas:Hardyt, 3310, D

Postorder

Jane:Dane Aj:Gup Foo:Done Aj:Gupt Ana:Trujillo Antonio:Moreno Antonio:Morenot Christina:Berglund Thomas:Hardy Thomas:Hardyt

Search:

Search for John:Doe

Found the node false

Search:

Search for Christina:Berglund

Found the node true

Insert:

Adding Hanna:Moos, 2050,B

Insert:

Adding Elizabeth:Lincoln, 1120, B

Delete:

Deleteing June:Doe

This node doesn't exist in the tree

Insert:

Adding Liz:Clay, 3510, C

Insert:

Adding Clay:Liz, 3410, B

Insert:

Adding Doe:John, 2020, A

Insert:

Adding Morenot:Antonio, 3310, A

Insert:

Adding Blah:Blah, 1110, E

Insert:

Adding John:Blew, 3450, I

Insert:

Adding Quincey:Joes, 4441, A

Insert:

Adding Deer:Zoo, 1010, B

Insert:

Adding :Aqua, 0000, C

Insert:

Adding :Clay, 1110, F

Insert:

Adding Walt:, 2330, A

Insert:

Adding Walt:Disney, 3310, A

Delete:

Deleteing Hanna:Moos

Insert:

Adding Hard:Marry, 3310, B

Insert:

Adding Waka:Young, 2240, A

Inorder

:Aqua :Clay Aj:Gup Aj:Gupt Ana:Trujillo Antonio:Moreno Antonio:Morenot Blah:Blah Christina:Berglund Clay:Liz Deer:Zoo Doe:John Elizabeth:Lincoln Foo:Done Hard:Marry Jane:Dane John:Blew Liz:Clay Morenot:Antonio Quincey:Joes Thomas:Hardy Thomas:Hardyt Waka:Young Walt: Walt:Disney

Insert:

Adding Hard:Marryt, 3310, D

Postorder

Jane:Dane Aj:Gup :Aqua :Clay Foo:Done Aj:Gupt Ana:Trujillo Antonio:Moreno Antonio:Morenot Christina:Berglund Blah:Blah Elizabeth:Lincoln Clay:Liz Doe:John Deer:Zoo Hard:Marry Hard:Marryt Thomas:Hardy Liz:Clay John:Blew Morenot:Antonio Quincey:Joes Thomas:Hardyt Walt: Waka:Young Walt:Disney

Search:

Search for Hanna:Moos

Found the node false

Insert:

Adding Alina:Sofia, 3310, C

Insert:

Adding Kayla:Jessie, 3310, B

Insert:

Adding Amma:Mia, 3310, A

Insert:

Adding Aiden:Maden, 3310, D

Insert:

Adding Caleb: Caden, 3310, A

Search:

Search for Elizabeth:Lincoln

Found the node true

Inorder

:Aqua :Clay Aiden:Maden Aj:Gup Aj:Gupt Alina:Sofia Amma:Mia Ana:Trujillo Antonio:Moreno Antonio:Morenot Blah:Blah Caleb: Caden Christina:Berglund Clay:Liz Deer:Zoo Doe:John Elizabeth:Lincoln Foo:Done Hard:Marry Hard:Marryt Jane:Dane John:Blew Kayla:Jessie Liz:Clay Morenot:Antonio Quincey:Joes Thomas:Hardy Thomas:Hardyt Waka:Young Walt: Walt:Disney

Insert:

Adding Steeve:Thebig, 3310, A

Delete:

Deleteing June:Doe

This node doesn't exist in the tree

Search:

Search for Nice

Found the node false

Delete:

Deleteing Steeve:Thebig

Insert:

Adding Hodah:Thomas, 2240, D

Insert:

Adding Nice:Lolah, 2240, B

Insert:

Adding Hodah:Thomast, 3310, D

Postorder

Jane:Dane Aj:Gup :Aqua :Clay Aiden:Maden Foo:Done Aj:Gupt Ana:Trujillo Alina:Sofia Amma:Mia Antonio:Moreno Antonio:Morenot Christina:Berglund Blah:Blah Caleb: Caden Elizabeth:Lincoln Clay:Liz Doe:John Deer:Zoo Hard:Marry Hard:Marryt Hodah:Thomas Hodah:Thomast Thomas:Hardy Liz:Clay John:Blew Kayla:Jessie Morenot:Antonio Quincey:Joes Nice:Lolah Thomas:Hardyt Walt: Waka:Young Walt:Disney

Search:

Search for Steeve:Thebig

Found the node false

Search:

Search for Nona:Lana

Found the node false

Insert:

Adding Brown:Yellow, 3310, A

Insert:

Adding Nona:Lana, 1120, B

Insert:

Adding Victoria:Ashworth, 1120, B

Delete:

Deleteing June:Doe

This node doesn't exist in the tree

Delete:

Deleteing Brown:Yellow

Insert:

Adding Patricio:Simpson, 2240, A

Insert:

Adding Francisco:Chang, 2240, B

Delete:

Deleteing Hanna:os

This node doesn't exist in the tree

Preorder

Jane:Dane Aj:Gup :Aqua :Clay Aiden:Maden Foo:Done Aj:Gupt Ana:Trujillo Alina:Sofia Amma:Mia Antonio:Moreno Antonio:Morenot Christina:Berglund Blah:Blah Caleb: Caden Elizabeth:Lincoln Clay:Liz Doe:John Deer:Zoo Hard:Marry Francisco:Chang Hard:Marryt Hodah:Thomas Hodah:Thomast Thomas:Hardy Liz:Clay John:Blew Kayla:Jessie Morenot:Antonio Quincey:Joes Nice:Lolah Nona:Lana Patricio:Simpson Thomas:Hardyt Walt: Waka:Young Victoria:Ashworth Walt:Disney

Inorder

:Aqua :Clay Aiden:Maden Aj:Gup Aj:Gupt Alina:Sofia Amma:Mia Ana:Trujillo Antonio:Moreno Antonio:Morenot Blah:Blah Caleb: Caden Christina:Berglund Clay:Liz Deer:Zoo Doe:John Elizabeth:Lincoln Foo:Done Francisco:Chang Hard:Marry Hard:Marryt Hodah:Thomas Hodah:Thomast Jane:Dane John:Blew Kayla:Jessie Liz:Clay Morenot:Antonio Nice:Lolah Nona:Lana Patricio:Simpson Quincey:Joes Thomas:Hardy Thomas:Hardyt Victoria:Ashworth Waka:Young Walt: Walt:Disney

Postorder

Jane:Dane Aj:Gup :Aqua :Clay Aiden:Maden Foo:Done Aj:Gupt Ana:Trujillo Alina:Sofia Amma:Mia Antonio:Moreno Antonio:Morenot Christina:Berglund Blah:Blah Caleb: Caden Elizabeth:Lincoln Clay:Liz Doe:John Deer:Zoo Hard:Marry Francisco:Chang Hard:Marryt Hodah:Thomas Hodah:Thomast Thomas:Hardy Liz:Clay John:Blew Kayla:Jessie Morenot:Antonio Quincey:Joes Nice:Lolah Nona:Lana Patricio:Simpson Thomas:Hardyt Walt: Waka:Young Victoria:Ashworth Walt:Disney

Here is the tree

Jane:Dane Aj:Gup :Aqua :Clay Aiden:Maden Foo:Done Aj:Gupt Ana:Trujillo Alina:Sofia Amma:Mia Antonio:Moreno Antonio:Morenot Christina:Berglund Blah:Blah Caleb: Caden Elizabeth:Lincoln Clay:Liz Doe:John Deer:Zoo Hard:Marry Francisco:Chang Hard:Marryt Hodah:Thomas Hodah:Thomast Thomas:Hardy Liz:Clay John:Blew Kayla:Jessie Morenot:Antonio Quincey:Joes Nice:Lolah Nona:Lana Patricio:Simpson Thomas:Hardyt Walt: Waka:Young Victoria:Ashworth Walt:Disney

**PHASE 7: REFINING THE PROGRAM (“Add bells and whistles to the program”)**

No refinements are needed. In this program, I have already included all required features.

**PHASE 8: PRODUCTION**

I prepared a copy of the entire program for Lab TA’s evaluation, as specified by the TA. Then, I sent electronically the copy to the Lab TA.

**PHASE 9: MAINTENANCE**

I wish there was an easier way to show the user what is going on. I could refine by not telling it what it is actually doing and just print out the pre, post, and in orders.