ELEC 278

Lab 4: AVL Trees

Weeks 7 and 8
Fall, 2019

Contents

1. Objective

The purpose of this lab is to give you some practice writing an AVL Tree insert function, and the necessary support code to determine if the tree is balanced, and if not, to rebalance it.

2. Organization

This lab includes three steps, all of which involve adding code to the AVL code provided to you in the zipped directory **lab4code.zip**

In Section 3, you will implement a function to calculate the height and balance factor of any node in the AVL tree.

In Section 4, you will implement 3 functions to rebalance an AVL node after insertion.

If you inspect the header file, avl.h, you will find the definition of a node in the tree:

```
typedef int Key;

// Node in tree has key and pointer to value associated with key.
// Also contains structural components - two pointers to left and
// right subtrees.
typedef struct Node   {
      Key  key;
      void *value;
      struct Node     *leftChild, *rightChild;
      int height;
} Node;
```

All of the function will be tested using the code in **`main.c`**. After solving all of the problems, show your programs to a TA in order to get your mark for this lab. You are allowed to work in groups of a maximum of two people. You are not allowed to change the code in **`main.c`** because it is used to check your code.

You should plan to build your own version of main.c and use it to check your code thoroughly. Make sure that you have tested all possible cases – the TA's will be checking your code to make sure you have considered all situations possible.

3. Step 1: Calculating the Height and Balance Factor of an AVL node

- From the node definition above, you can see that each node has a field recording its own height. Start by assuming that the heights of the left and right subtrees are correct. Calculate the height of the node root using the heights of its subtrees. No recursion is needed here.

- The balance factor of a node root is :

$$Balance\ Factor = H_L - H_R$$

3.1 Task

Your task is to implement the function: **`int calcHeight(Node *root)`** and **`int getBalanceFactor (Node* root).`** Write your implementation in **`avl.c`**.

3.2 Deliverable

1) Compile your code

```
cc main.c avl.c –o avlTree
```

2) Show the output by running **`./avltree`** as well as demonstrating your source code, to a TA during the lab.

The output shall look similar to this:

```
Inserting 10
Inorder: 10(h=0,bf=0) -
-------
Inserting 3
Inorder: 3(h=0,bf=0) - 10(h=1,bf=1) -
-------
Inserting 1
Inorder: 1(h=0,bf=0) - 3(h=1,bf=1) - 10(h=2,bf=2) -
-------
Inserting 7
Inorder: 1(h=0,bf=0) - 3(h=1,bf=0) - 7(h=0,bf=0) - 10(h=2,bf=2) -
-------
Inserting 20
Inorder: 1(h=0,bf=0) - 3(h=1,bf=0) - 7(h=0,bf=0) - 10(h=2,bf=1) -
20(h=0,bf=0) -
-------
Inserting 15
Inorder: 1(h=0,bf=0) - 3(h=1,bf=0) - 7(h=0,bf=0) - 10(h=2,bf=0) -
15(h=0,bf=0) - 20(h=1,bf=1) -
-------
Inserting 18
Inorder: 1(h=0,bf=0) - 3(h=1,bf=0) - 7(h=0,bf=0) - 10(h=3,bf=-1) -
15(h=1,bf=-1) - 18(h=0,bf=0) - 20(h=2,bf=2) -
-------
Inserting 17
Inorder: 1(h=0,bf=0) - 3(h=1,bf=0) - 7(h=0,bf=0) - 10(h=4,bf=-2) -
15(h=2,bf=-2) - 17(h=0,bf=0) - 18(h=1,bf=1) - 20(h=3,bf=3) -
-------
```

```
Inserting 16
Inorder: 1(h=0,bf=0) - 3(h=1,bf=0) - 7(h=0,bf=0) - 10(h=5,bf=-3) -
15(h=3,bf=-3) - 16(h=0,bf=0) - 17(h=1,bf=1) - 18(h=2,bf=2) - 20(h=4,bf=4) -
-------
Inserting 22
Inorder: 1(h=0,bf=0) - 3(h=1,bf=0) - 7(h=0,bf=0) - 10(h=5,bf=-3) -
15(h=3,bf=-3) - 16(h=0,bf=0) - 17(h=1,bf=1) - 18(h=2,bf=2) - 20(h=4,bf=3) -
22(h=0,bf=0) -
```

4. Step 2: Rebalance tree after insertion

After a value is inserted in the tree, the tree may be unbalanced. If the balance factor if a node is less than -1 or greater than 1, then the node is unbalanced and needs rebalancing.

As discussed in class, there are 4 types of rotations:

Name	Rotation
Left of Left (LoL)	Right rotation
Right of Right (RoR)	Left rotation
Right of Left (RoL)	Left rotation on subtree Right rotation
Left of Right (LoR)	Right rotation on subtree Left rotation

Determine the type of rotation needed and perform it on the tree.

4.1 Task

You will implement 3 function.

1. RotateRight (Node* root)
   Perform right rotation on node root, and returns new root.
2. RotateLeft (Node *root)
   Perform left rotation on node root, and returns new root.
3. Rebalance (Node *root)
   This function will determine the type of rotations needed and call the appropriate functions.

4.2 Deliverable

1. Recompile the code
2. Run **./avltree** and show the output to the TA.

The output from your program shall look similar to the following:

```
Inserting 10
Inorder: 10(h=0,bf=0) -
-------
Inserting 3
Inorder: 3(h=0,bf=0) - 10(h=1,bf=1) -
-------
```

```
Inserting 1
Node 10 is unbalanced. Left of Left: Rotate Right
Inorder: 1(h=0,bf=0) - 3(h=1,bf=0) - 10(h=0,bf=0) -
-------
Inserting 7
Inorder: 1(h=0,bf=0) - 3(h=2,bf=-1) - 7(h=0,bf=0) - 10(h=1,bf=1) -
-------
Inserting 20
Inorder: 1(h=0,bf=0) - 3(h=2,bf=-1) - 7(h=0,bf=0) - 10(h=1,bf=0) -
20(h=0,bf=0) -
-------
Inserting 15
Node 3 is unbalanced. Right of Right: Rotate Left
Inorder: 1(h=0,bf=0) - 3(h=1,bf=0) - 7(h=0,bf=0) - 10(h=2,bf=0) -
15(h=0,bf=0) - 20(h=1,bf=1) -
-------
Inserting 18
Node 20 is unbalanced. Right of Left: Rotate Left
Rotate Right
Inorder: 1(h=0,bf=0) - 3(h=1,bf=0) - 7(h=0,bf=0) - 10(h=2,bf=0) -
15(h=0,bf=0) - 18(h=1,bf=0) - 20(h=0,bf=0) -
-------
Inserting 17
Inorder: 1(h=0,bf=0) - 3(h=1,bf=0) - 7(h=0,bf=0) - 10(h=3,bf=-1) -
15(h=1,bf=-1) - 17(h=0,bf=0) - 18(h=2,bf=1) - 20(h=0,bf=0) -
-------
Inserting 16
Node 15 is unbalanced. Left of Right: Rotate Right
Rotate Left
Inorder: 1(h=0,bf=0) - 3(h=1,bf=0) - 7(h=0,bf=0) - 10(h=3,bf=-1) -
15(h=0,bf=0) - 16(h=1,bf=0) - 17(h=0,bf=0) - 18(h=2,bf=1) - 20(h=0,bf=0) -
-------
Inserting 22
Inorder: 1(h=0,bf=0) - 3(h=1,bf=0) - 7(h=0,bf=0) - 10(h=3,bf=-1) -
15(h=0,bf=0) - 16(h=1,bf=0) - 17(h=0,bf=0) - 18(h=2,bf=0) - 20(h=1,bf=-1) -
22(h=0,bf=0) -
-------
```

5. Stretch Goal

There is no stretch goal for lab 4.

ELEC 278 LAB EXERCISE 4 CHECK-OFF SHEET

Print this sheet and bring it to the lab with you. The TA will sign off the steps as you complete them. Make sure you hand in your signed sheet before you leave the lab.

Student Name (optional)	
Student Number (8 digits)	

Part Number	Step Description	Mark	TA Sign Off
Step 1 (Section 3)	Implement and demonstrate calcHeight() and getBalanceFactor()	4	
Step 2 (Section 4)	Implement and demonstrate Rotateright(), Rotateleft() and Rebalance()	6	
	No stretch goal this lab.		
	Full marks (complete lab requirements)	10	
	Maximum possible marks	10	