7. Lab 6: Trees

7.1. Objectives

- Know how binary Tree works.
- Implementing additional methods to deal with binary trees in Java programming.

You are provided with two files

- Tree.java
- TreeApp.java

7.2. Problems 1

(10 points) Add a method that counts the elements in a binary tree into the *Tree Class*. Specifically, the method takes no parameters and returns an integer equal to the number of elements in the tree.

7.3. Problems 2

(10 points) Add a method that computes the height of a binary tree into the *Tree Class*. Specifically, this method has no parameters and returns an integer equal to the height of the tree.

7.4. Problems 2

(10 points) Add a method that counts a binary tree's leaves tree into the *Tree Class*. Specifically, this method has no parameters and returns an integer equal to the number of leaves in the tree.

7.5. Problems 3

(10 points) Add a method that determines whether a binary tree is fully balanced. This method takes no parameters and returns a Boolean value: true if the tree is fully balanced and false if not.

7.6. Problems 4

(10 points) Define two binary trees to be identical if both are empty or their roots are equal, their left subtrees are identical, and their right subtrees are identical. Design a method that determines whether two binary trees are identical *(this method takes a second binary tree as its only parameter and returns a Boolean value: true if the tree receiving the message is identical to the parameter, and false otherwise)*.

7.7. Huffman coding

(15 points) Draw a Huffman coding tree for the following text with **YOUR_NAME** below is your full name.

"I am a student at International University. My name is YOUR_NAME. I am working on a DSA lab"

7.8. Tree.java

Create different arithmetic expressions, write a method to read prefix notation and create a tree.

Hint: https://www.cs.colostate.edu/~cs165/.Fall19/recitations/L15/doc/traversal-order.html