Laboratory practice No. 4: Binary Tree

Omar Alexis Becerra Sierra
Universidad EAFIT
Medellín, Colombia
oabecerras@eafit.edu.co

Juan Jose Tamayo
Universidad EAFIT
Medellín, Colombia
jjtamayoa@eafit.edu.co

April 23, 2018

1) Codes to deliver in GitHub

1.a. ***Investigate what's your names, your parents names, your grandparents names and your great grandparents names. With this information, build your family tree in Java using the class BinaryTree.***

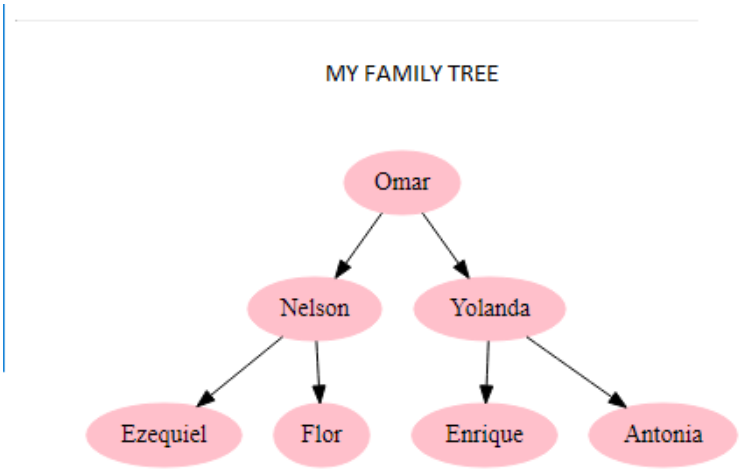

Figure 1: Family Tree

1.b. Can be implemented an more efficient family tree to make the search and the insertion be made in logaritmic time? Or not? Why?

To improve the efficienty of the family tree can be used an other class tree that is self-balancing like an AVL tree or an Red-Black tree, the problem with a tree of these is that balancing a family tree is that it would mean changes in the "offspring" where changes the hierarchical position of every family member, so it's not can improve the family tree implementing trees to solve the exercise.

1.c. Explain with your own words how work the exercises 2.1 and 2.2

The first that we do is write a main method where is created an object with tree type and then we start to insert the data given previously and then starts to save the numbers inside the tree, ,when you have finished entering the data, the PosOrden class is called to print the tree in PostOrden.

1.d. Complexity print on PosOrden

The complexity of print in PostOrden is $O(n * log(n))$ because inserting into a tree, the complexity is $log(n)$ and travel a tree , the complexity is n.

```
class Arbol {
    public Nodo raiz;
    Arbol(){
        raiz = null;
    }
    public  void insertar(int n){                    complejidad log(n)
        Nodo nuevo = new Nodo(n);
        if(raiz == null){
            raiz = nuevo;
        }else{
            Nodo aux = raiz;
            Nodo padre;
            while(true){
                padre = aux;
                if(n<aux.dato){
                    aux = aux.izquierda;
                    if(aux == null){
                        padre.izquierda = nuevo;
                        return;
                    }
                }else{
                    aux = aux.derecha;
                    if(aux == null){
                        padre.derecha = nuevo;
```

```
                    return;
                }
            }
        }
    }
    public void recorridoPosorden(){        complejidad n
        auxPosorden(raiz);
    }
    private void auxPosorden(Nodo nodo){
        if( nodo == null ){
            return;
        }

        auxPosorden(nodo.izquierda);
        auxPosorden(nodo.derecha);
        System.out.print(nodo.dato + " ");
    }

}
```

1.e. Explain with your own words the variables (what is 'n', what is 'm',...)of the complexity calculation of the exercise 3.4

In the previous complexity calculation the variable 'n' refers to the number of times that the user inserts a number, now, as the size of the tree is equal to the number of numbers that inserts the variable 'n' also refers to the size of the tree

1.f. Mock exam

i. line 4 : 1 + altura(Nodo.izq)
line 5: 1 + altura(Nodo.der)

ii. c

iii.
- **i.** 0
- **ii.** a.dato
- **iii.** a.der,suma + a.dato
- **iv.** a.izq,suma + a.dato

iv. a) T(n) = T(n-1) + c

v. a) p == toInsert
b) p < toInsert

Profoessor Mauricio Toro Bermudez
Phone: (+57)(4)2619500 Ext. 9473. Office: 19 – 627
E-mail: mtorobe@eafit.edu.co

vi. a) d
b) return 0;
c) == suma

1.g. Recommended reading

Binary trees

Why use binary trees? Because it combines the advantages of two others structures: an ordered array and a linked list. You can search a tree quickly, as you can an ordered array, and you can also insert and delete items quickly, as you can with a linked list.
What is a Tree? Although we are interested in binary tree, let us begin seeing what is a tree. A tree consists of nodes connected by edges. In computer programs, nodes often represent te typical items we store in any kind of data structure.The lines (edges) between the nodes represent the way the nodes are related. Typically there is one node in the top row of a tree, with lines connecting to more nodes on the second row, even more on the third, and so on. Thus trees are small on the top and large on the bottom.
Unbalanced trees: Trees becomes unbalanced because of the order in which the data items are inserted. If these key values are inserted randomly, the tree will be more or less balanced. If a tree is created by data items whose key values arrive in random order, the problem of unbalanced trees may not be too much of a problem for larger trees, because the chances of a long run of numbers in sequence is small.

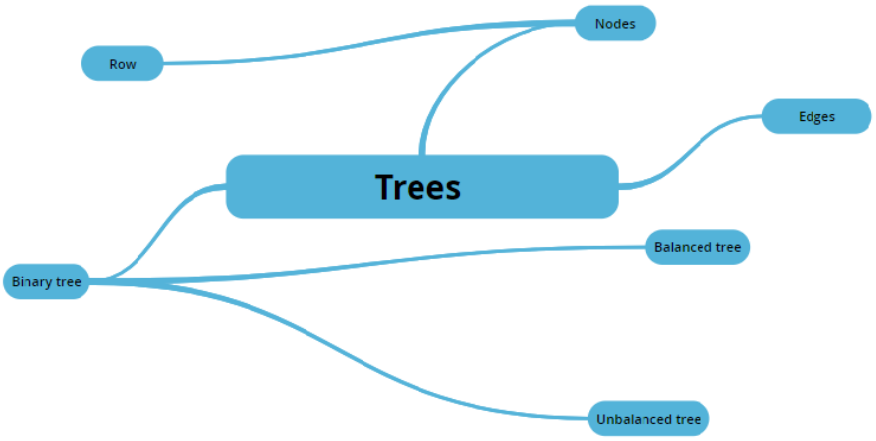

Figure 2: Mental map of the recommended reading