Laboratory practice No. 4: Trees

Juliana Lalinde Velásquez
Universidad Eafit
Medellín, Colombia
jlalindev@eafit.edu.co

Isabel Urrego Gómez
Universidad Eafit
Medellín, Colombia
iurregog@eafit.edu.co

3) Practice for final project defense presentation

1. Can a family tree be implemented so that insertion and search can be made in a logarithmic time? Why?

The answer is yes. If we build a family tree as a binary search tree, it would be capable of doing these functions in O(logN) time.
The reason we can rebuild our family tree as a binary search tree is because it would fulfill the aspects needed to tag it as that type of data structure. For example, each node has at most two children, in this case father and mother and it would be perfectly balanced because the left root are the female members and the right one the male members.

2. Explanation of the algorithms

Exercise 2.1
This algorithm works by taking an integer array as an input, which is organized as a binary tree. Then, it works its way through the tree starting by the root and then proceeding recursively with two actions: the first recursive call examines the left node to the node we are currently in, and the second recursive call examines the right node. After both actions are executed the algorithm prints the data stored in the node. This way the algorithm examines all the left subtree, stops when it reaches a leaf and starts printing the elements from bottom to top. Then this process is executed with the right subtree and, finally, the root value is printed.

Exercise 2.2
This algorithm works recursively and stops when it detects a leaf node returning true if there is a path which sum equals the number wanted. While that statement isn't true it keeps calling itself recursively and takes the left and right children as the new nodes that is why there are two recursive calls, one for each child. Besides, on each new call the data of the actual node will be rested to the sum since it will always be included on the path. This way the algorithm verifies every single path and returns true if it finds the one that sums the number on the parameter. Otherwise it returns false. The other possibility is that we have an empty tree and in that case the algorithm also returns false.

3. Complexity of the algorithms

Exercise 2.1
Complexity equation:

$$T(n) = 2T(n/2) + c_1$$

Solution

$T(n) = \frac{c_1 n}{2} + c_1 (n-1)$ (c_1 is an arbitrary parameter)

O(n)

Exercise 2.2
Complexity equation:

$$T(n) = 2T(n/2) + c_1$$

Solution

$T(n) = \frac{c_1 n}{2} + c_1 (n-1)$ (c_1 is an arbitrary parameter)

O(n)

4. **Variables of the complexity**

Exercise 2.1: O(n), where n is the number of nodes in the tree
Exercise 2.2: O(n), where n is the number of nodes in the tree

4) Practice for midterms

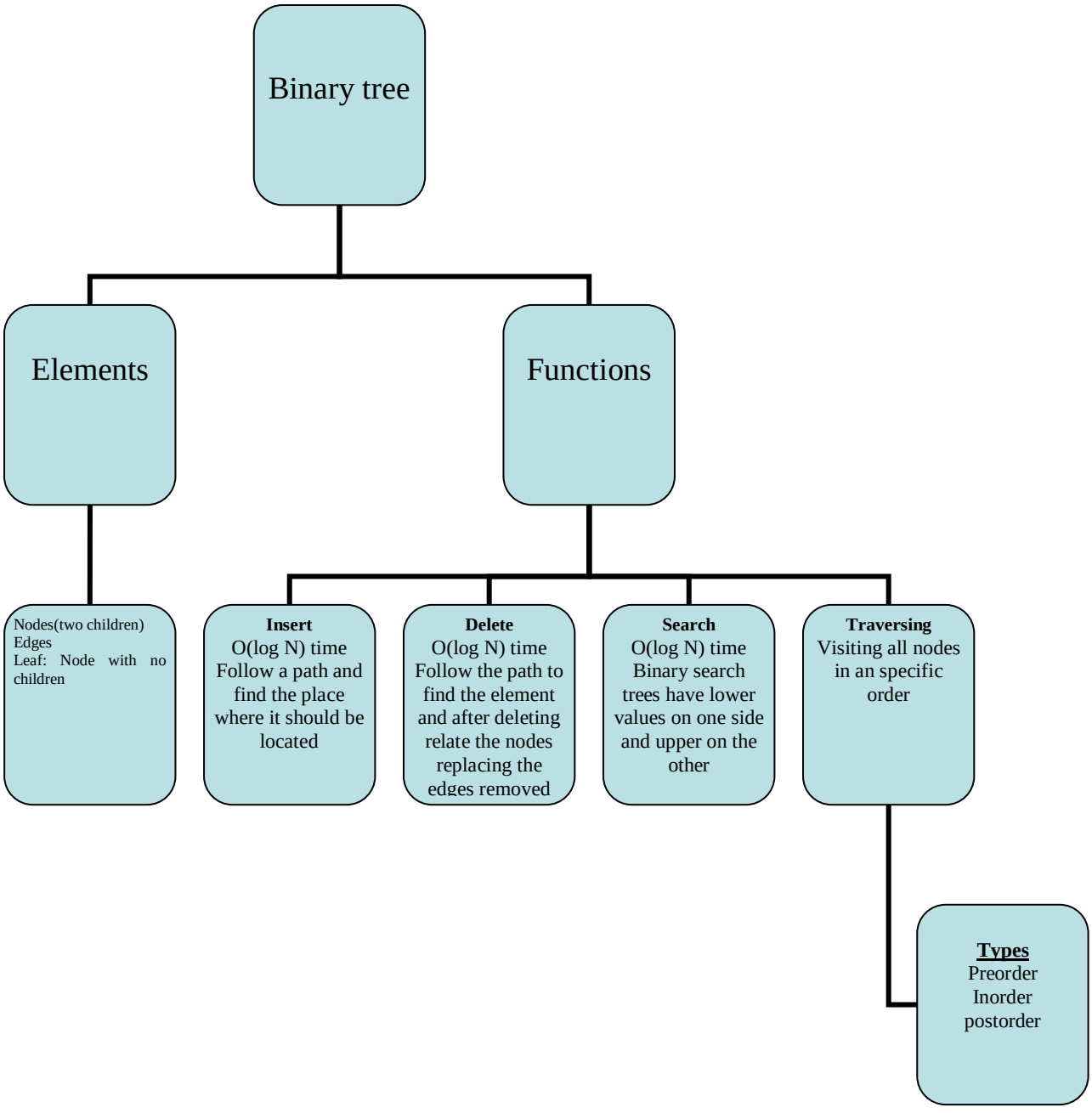
1. a. altura(raiz.izq)
   b. altura(raiz.der)
2. c
3. a. false
   b. a.dato
   c. a.izq, suma - a.dato
   d. a.der, suma - a.dato
4. 4.1. c
   4.2. a
   4.3. d
   4.4. a
5. a. p.data== toInsert
   b. p.data> toInserr
6. 6.1. d
   6.2. return 0
   6.3. ==0
7. 7.1. 1
   7.2. 2
8. b
9. a
10. b

5) Recommended reading (optional)

a) Chapter 8: Binary Trees
b) A tree is a tree made of nodes and edges that represent the way how the nodes are related. What differences a binary tree from the others is the maximum number of children of each node, two and that is why they are called the right child and left child. Additionally, this data structure combines the two main advantages of an array and a linked list searching, adding and deleting very fast.

There is also a specific case of a binary tree commonly known as binary search tree, which separates the data between bigger than and less than. Therefore, what is in the left o a node are the values that are smaller than the data contained in that node and the right values are the ones bigger, this repartition of the data makes it easier to search, insert and add.

c) Concept map



6) Team work and gradual progress (optional)

a) Meeting minutes

Date	Time	Description
13.10.18	1 hour	Distribution of the work
15.10.18	1 hour (by phone)	Discussion of parts 1,2 and 3
Additional work		Discussion of 2.2 Watching videos about traversing

b) History of changes of the code

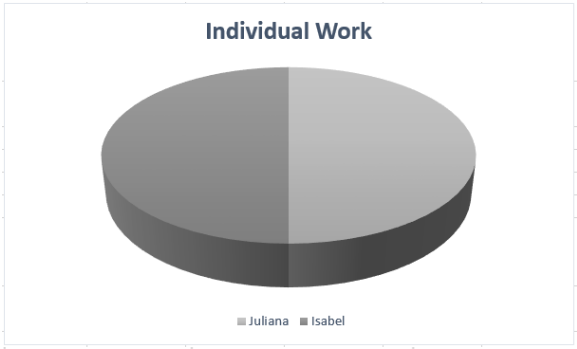
Version	Includes	Left	Status
1.0	1.1	1.2 2.1	
2.0	1 and 1.2	2.1	
3.0	1 and 2		Complete

c) History of changes of the report

Version	Includes	Left	Status
1.0	4 (odd numbered excercises) 5	3 and rest of 4	
2.0	3.1 and 5 complete Half of 4	3.2, 3.3, 3.4 4 (even numbers)	
3.0	Parts 3, 4 and 5		

d. Individual work

PROFESSOR MAURICIO TORO BERMÚDEZ
Phone: (+57) (4) 261 95 00 Ext. 9473. Office: 19 - 627
E-mail: mtorobe@eafit.edu.co



Additional parts		
Part 2	2.2	done
Part 3	2.2 complexity	done
Part 5	Lecture	done
Part 6		done
Part 7		done

Member	Part of the Laboratory	Description
Juliana	1	
	2	2.1
	3	3.2 of 2.1 3.3 of 2.1 and 2.2 3.4 of 2.1 and 2.2
	4	Even excercises (2,4,6,8,10)
	6 and 7	Team work and english
Isabel	1	1.1 and 1.2
	2	2.2
	3	3.1 3.2 of 2.2
	4	Odd excercises (1,3,5,7,9)
	5	Lecture
	6 and 7	Team work and english

PROFESSOR MAURICIO TORO BERMÚDEZ
Phone: (+57) (4) 261 95 00 Ext. 9473. Office: 19 - 627
E-mail: mtorobe@eafit.edu.co