Laboratory practice No. 4: Trees

Ana Sofía Gutiérrez
Universidad Eafit
Medellín, Colombia
asgutierrt@eafit.edu.co

Santiago Hidalgo Ocampo
Universidad Eafit
Medellín, Colombia
shidalgoo1@eafit.edu.co

3) Practice for final project defense presentation

3.1 The tree is n-ary and has a complexity of O(n*m)
3.2
3.3 A binary search tree is built with the given entry, then the tree is printed with the posorder method
3.4 Exercise 2.1: Build the tree is O(n) and and cross the tree in a post-order way is O(n).
3.5 Exercise 2.1: n is the number of nodes in the tree

4) Practice for midterms

4.1 A altura(node.izq)+1
B altura(node.der)+1
4.2. Option: C)3
4.3 A false
B a.val
C a.izq, suma-a.val
D a.der, suma-a.val
4.4 .1 Option c) T(n) = 2*T(n/2) +C
.2 Option a: O(n)
.3 Option: d)
.4 Option A is the most appropriate, None is correct!
4.5. A toInsert==p.value
B toInsert>p.value
4.6 .1 Option: d) 4
.2 return 0;
.3 if(raiz.hijos.size() ==0)...
4.7 .1 Option: a) 0, 2, 1, 7, 5, 10, 13, 11, 9, 4
.2 Option: b) 2
.3 Document error (No question)
4.8 Option: b) 2
4.9 Option: a) 5, 3, 6, 1, 7, 4, 8, 0, 2

4.10 Option: b) No

4.11

.1 Option: b)
.2 Option: a) 5
.3 Option: b) No

4.12.

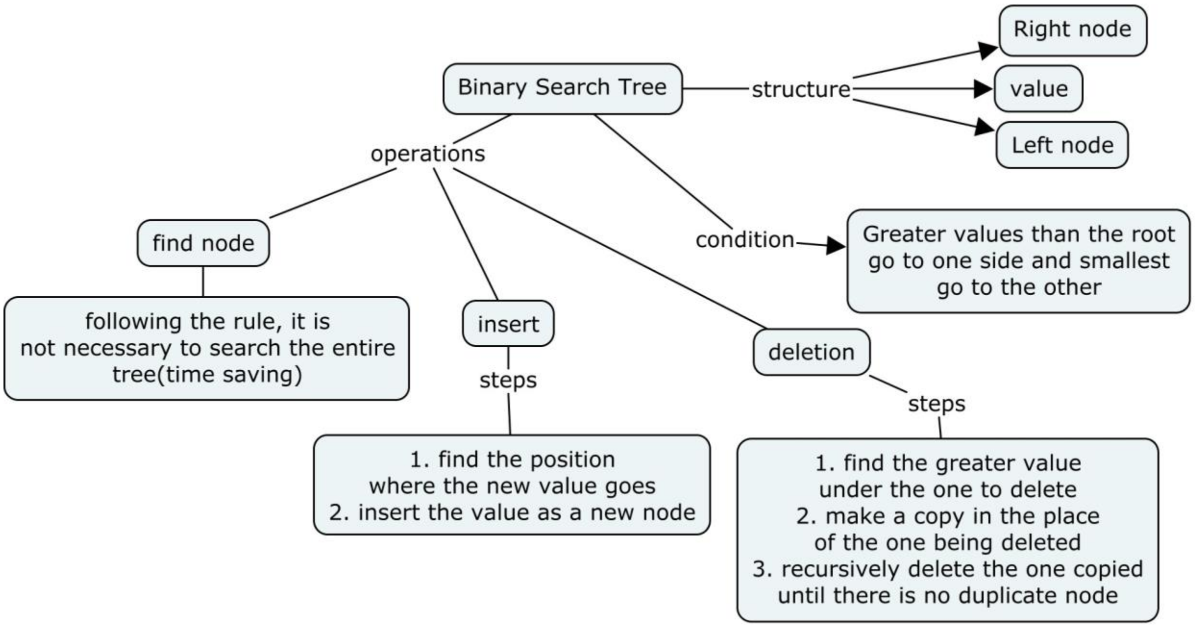
1. Option: i)
2. Option: a)
3. Option: b) O(log(n))

4.13

1. suma[raiz.id]...
2. Option: d) $T(n) = nT(n - 1) + c$, que es $O(n!)$

5) Binary search trees (BSTs)

The main use of this data structure is for searching, reducing the average time for this operation to O(log n) in a balanced tree. The tree's elements follow the condition: all the greater values go to one side and all the smallest go to the other in each node. Just like in other trees, each node has a right node, a left node and a value.
Main operations are finding a value, insert and delete nodes. The first one is more time efficient than other trees because it only investigates one of the sides depending on whether the value is greater or less than the root, which means it doesn't have to look into all of the paths. Due to the display of the tree, the in-order print function will output a sorted list of the elements. The insertion operation executes two sub-operations: first, it finds the place where the given data must appear and then it adds the value in this position. On the other hand, the deletion operation is more complex; in order to delete a node(that is followed by more values), you have to put the greater value that appears below that node in the current node you want to delete and then delete the duplicated node the same way in a recursive call to the method until there are no duplicated nodes left.


Binary Search Tree
structure
Right node
value
Left node
operations
condition
Greater values than the root go to one side and smallest go to the other
find node
following the rule, it is not necessary to search the entire tree(time saving)
insert
steps
1. find the position where the new value goes
2. insert the value as a new node
deletion
steps
1. find the greater value under the one to delete
2. make a copy in the place of the one being deleted
3. recursively delete the one copied until there is no duplicate node