Laboratory practice No. 4: Trees

Vincent Alejandro Arcila Larrea
Universidad Eafit
Medellín, Colombia
vaarcilal@eafit.edu.co

Isabel Piedrahita Velez
Universidad Eafit
Medellín, Colombia
ipiedrahiv@eafit.edu.co

3) Practice for final project defense presentation

3.1 The data structure implemented to represent the filesystem is a binary tree on which each node has two branches. One branch for a first son and one for a first brother. This way, the first file (let it be A) of a directory found while parsing the output of *tree -shaul* will be the son of a node representing its father directory. The next file (let it be B) under the same directory will be A's brother. The next one would be B's brother, and so on. The root would have no brother and the leafs would have no son or brother.

The search complexity is O(n).

3.2 It is not possible given the fact that in order to improve the complexity to insert or search one would have to make the tree a balanced tree, and by doing this one would end up with a tree that no longer represents a genealogical tree.

3.3 Firstly, in order to solve exercise 2.1 we used an algorithm that would take the information in the specified format from a .txt file and transformed it into an array by reading each line of the file and adding it to an arraylist, that is then transformed into an array of the appropriate size.

Then we used a method findPostOrderAux in order to actually produce the desired output. First, it declares a base case, in which once all of the preorder array has been transversed, it will return. The second conditional is a cautionary method that ensures that the value is inside the range specified, otherwise we can assume that it does not belong to the current subtree and return.

Then the value on the current position of the preOrder array is stored on an int variable and the value of preIndex is increased by one, so that the next time the method is called, it will follow this same procedure for the next element in the array.

Then the method is called in the following way `findPostOrderAux(pre, n, minval, val, preIndex);` this is to transverse all the elements in the left subtree, and like this `findPostOrderAux(pre, n, val, maxval, preIndex);` allto transverse all of the elements that would be on the right subtree.

Finally a simple `System.out.println(val)` command is used to print the previously stored value in the specified format.

3.4 As stated, mainly two methods were used to solve numeral 2.1, the first one, used to create an array from a plain text document has a complexity of T(n) = n+n → T(n) = n → O(n). The second method used to generate the postorder transversal is recursive, and has a complexity of T(n) = $\frac{n}{2} + \frac{n}{2}$ → T(n) = n+n → T(n) = n → O(n). This means the whole implementation has a time complexity of O(n) + O(n) = O(n).

3.5 In our complexity analysis n is the number of elements provided by the input.

4) Practice for midterms

4.3.a) false **4.3.b)** a.dato **4.3.c)** a.der, suma-a.dato **4.3.d)** a.izq, suma-a.dato	**4.9)** a
4.7.1) a **4.7.2)** b	**4.10)** No
4.8) b	**4.11.1)** b **4.11.2)** a **4.11.3)** No

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473