Laboratory No. 4: Trees

Camila Barona Cabrera
Universidad Eafit
Medellín, Colombia
cbaronac@eafit.edu.co

Mariana Gómez Piedrahita
Universidad Eafit
Medellín, Colombia
Mgomezp10@eafit.edu.co

3) Mock project support questions

1. Having the family tree, you can treat the same and do the operations of a binary search tree, then you can highlight the search in this type of trees is very efficient, is represented in most cases a logarithmic function. A maximum number of comparisons that would be necessary to know if an element is in a tree. Binary search is between [log2 (N + 1)] and N, where N is the number of nodes.
   That is, the time necessary for the operations of the common tree to run is proportional to the base-2 log of N.
   In the Big-O notation we say that operations stories take time O (logN). Each time a trajectory is followed by a certain subtree, half of the nodes can be downloaded. This means, it is governed by the recurrence relation: T (n) = T (n / 2) + c, which has a logarithmic solution. This is how the structure of the binary search tree, many advantages to implement dynamic sets, related to the lists, which allow algorithms of complexity O (n).

2. Exercise 2.1 works as follows: We ask the user to enter the nodes that you want to store in the binary tree, within the method it is initially checked if the entered node is different from null, as it is, two recursive calls are made which The function is initially to print the nodes found in the left branch of the tree that are less than the root, then print the nodes found in the right branch that are greater than the root and finally print the root that will be the first element entered to show the user a post-order tree.

3. O(n)

4. The variable n in the previous complexity calculation is the number of elements in the tree.

4) Test simulation

1. ***a***. *raiz.izq.dato;*
   b.*raiz.der.dato*;
2. *c*
3. ***a.*** *false.*
   b. *nodo.dato*
   c. (*a.der,suma-a.dato)*
   d. *(a.izq,suma-a.dato)*
4. ***1.*** *b.*
   2. a.
   3. d.
   4. *a.*
5. ***a.*** *p.data==toInsert*
   b. *toInsert>=p.data*

6. 1. *d.*
   2. *return 0;*
   3. *raíz.hijos.size()>0*
7. 1. *1.*
   2. *2.*
8. *b.*
9. *a.*
10. *b.*

5) *Recommended reading (optional)*

Binary Tree

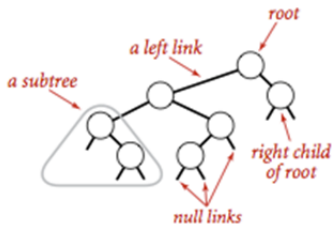
A binary tree consists of nodes connected by edges, in computer programs, nodes represent the typical items we store in any kind of data structures, like: car parts, airline reservations, and others, the lines(edges) between the nodes represent the way the nodes are related. Edges are likely to be represent in a program by references. It is good to use binary trees because it combines the advantages of two structures; ordered array and linked list and insert and delete items quickly.
To represent a tree in java it is necessary:
-*Class of node objects:*
These contains the data representing the objects being storeded and also references to each of the node's two children.

-*The tree class:*
The tree class has a number of methods: some for finding, inserting, and deleting nodes, several for different kinds of traverses, and one to display the tree
-*The treeApp class:*
we need a way to perform operations in the tree.
The time needed to carry out the common tree operations is proportional to the base-2 log of N. In Big-O notation we say such operations take O(logN)time.

conceptual map below

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

UNIVERSIDAD EAFIT	UNIVERSIDAD EAFIT ESCUELA DE INGENIERÍA DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS	Código: ST245 Estructura de Datos 1



Anatomy of a binary tree

Image take from:https://algs4.cs.princeton.edu/32bst/

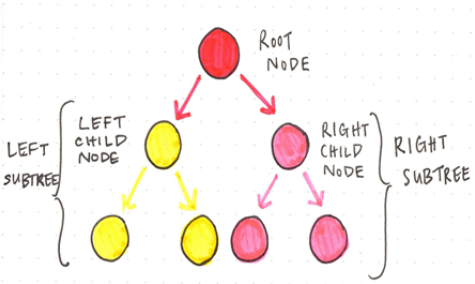


Image take from:https://medium.com/basecs/leaf-it-up-to-binary-trees-11001aaf746d

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co