Laboratory practice No. 4: Hash Tables and Trees

Martin Ospina Uribe
Universidad Eafit
Medellín, Colombia
mospinau1@eafit.edu.co

María José Bernal Vélez
Universidad Eafit
Medellín, Colombia
mjbernalv@eafit.edu.co

3) Practice for final project defense presentation

3.1 Bee collisions data structures

The main goal of this exercise is to determine collisions between bees that are within 100 meters. To achieve this we used an octree, a tree data structure in which each node has exactly eight children. They are most used to divide a 3-dimensional space by recursively subdividing into eight octants. They are the equivalent of quadtrees in a 2-dimensional space.

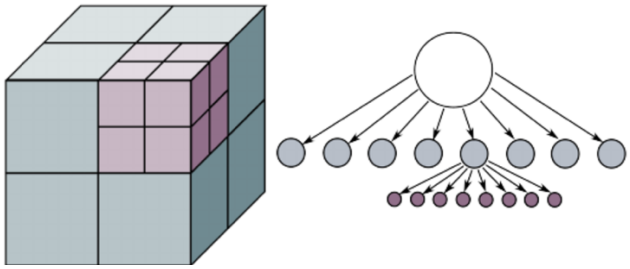


Figure 1: Octree example. Taken from: https://www.researchgate.net/figure/An-octree-mesh-left-and-its-representation-as-a-tree-right_fig1_286523372

This data structure is very efficient as is has a time complexity for the worst case of $O(logn)$for inserting and searching. Due to this, it is very efficient, compared to other methods used such as comparing every bee to the desired bee, which would have a complexity of $O(n^2)$.

3.2 Is it possible to implement a more efficient family tree so that searching and inserting could be done in logarithmic time? Why?

We think that it is not possible to implement a more efficient family tree due to the fact that one of the only ways to achieve a logarithmic time complexity would be by using a self-balancing binary search tree. This is not possible because usually there are people in the family tree whose names are unknown, which will cause the tree to be unbalanced, and if you try to balance it, some people may have to be moved in the tree, which causes a shift on the order, and thus, an incorrect family tree.

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

3.3/3.4/3.5 Explanation and complexity of project simulation exercises and meaning of variables

2.1 Postorder

In this exercise, our main goal was to find the postorder (left, right, root) of a binary search tree given the inorder (root, left, right). In order to do this, we used the methods that we had already created before, such as add an element in a tree, and we created a method that prints the tree in postorder. First, the elements of the tree are received in separate lines and are added to the tree with the add method; and finally, they are returned in the desired order using the pos-order function, which is a recursive function that starts by traveling the root of the tree; continues with the left brach; and finishes with the right branch.

- **Complexity:**

Postorder method:

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

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

$O(n)$

Insert method:

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

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

$O(n)$

Worst case:

$O(n^2) \rightarrow$ this is due to the fact that each node has to be added to the tree in a loop, and then use the method postorder, so it would be $O(n * n + n) = O(n^2 + n) = O(n^2)$.

- n = size of the tree, which corresponds to the number of nodes of the tree.

4) Practice for midterms

4.1 1. b
2. d

4.2 1. It returns the lowest common ancestor between nodes n1 and n2.
2. $T(n) = 2T(n/2) + c$ or $O(n)$
3. It can be improved by adding a conditional to see if n is less than or greater than the data. If it is smaller, it goes to search the nodes on the left; else, it goes to the right. This is due to the fact that it is a BST, so it is organized. Complexity would be improved to $O(logn)$.

4.3 1. return True
2. $O(n)$

4.4 1. c
2. a
3. d
4. a

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

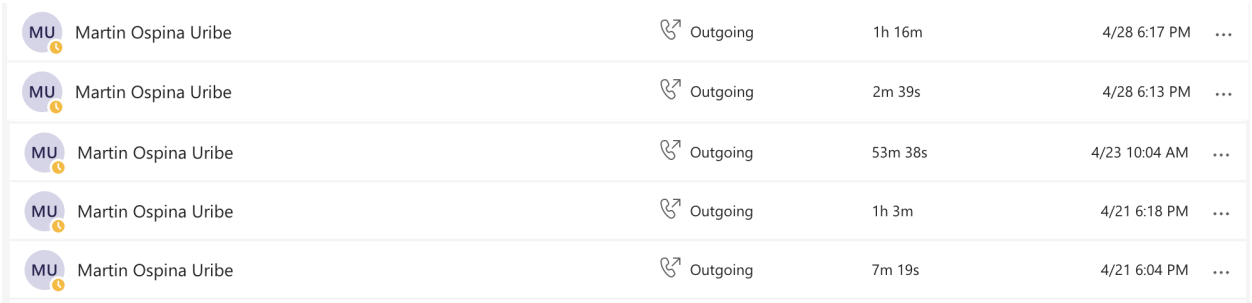
4.5 a. toInsert==p.data
b. toInsert>p.data
4.6 1. d
2. return 0
3. ==0
4.7 1. a
2. b
4.8 b
4.9 a
4.10 1. b
2. a
3. b
4.11 1. raiz.id
2. a
4.12 1. i
2. a
3. a

6) Teamwork and gradual progress (optional)

Member	Date	Done	Doing	To do
Martín and Maria José	21/04/2021	Practice for midterms	Project simulation exercises	Practice for final project defense presentation
Martín and Maria José	23/04/2021	Exercise 1.3	Finding exercise complexities	Practice for final project defense presentation
Martín and Maria José	28/04/2021	Exercises 1.1 and 2.1	Finish exercise 1.1	Work on the report
Martín and Maria José	2/04/2021	Work and finish the report. Upload laboratory on GitHub.		

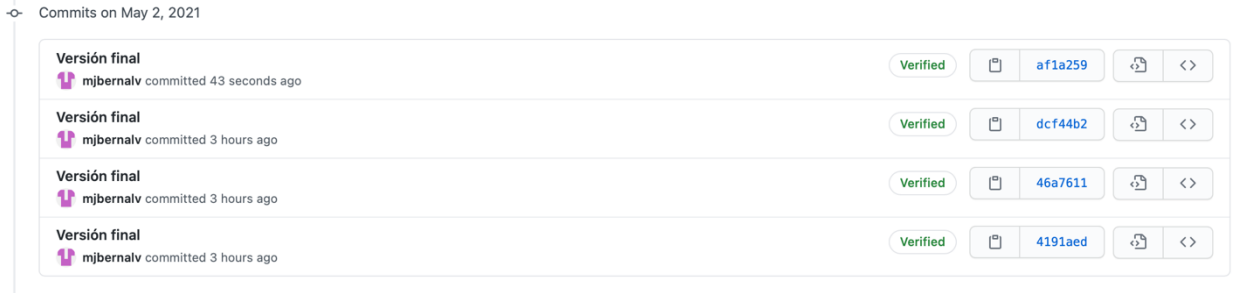
PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

6.1 Meeting minutes

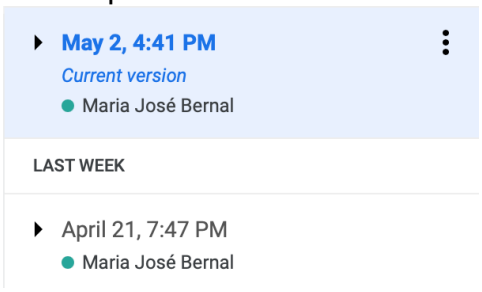


MU	Martin Ospina Uribe	Outgoing	1h 16m	4/28 6:17 PM
MU	Martin Ospina Uribe	Outgoing	2m 39s	4/28 6:13 PM
MU	Martin Ospina Uribe	Outgoing	53m 38s	4/23 10:04 AM
MU	Martin Ospina Uribe	Outgoing	1h 3m	4/21 6:18 PM
MU	Martin Ospina Uribe	Outgoing	7m 19s	4/21 6:04 PM

6.2 History of changes of the code



6.3 History of changes of the report



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