TECNOLÓGICO DE MONTERREY

CAMPUS PUEBLA

Final Project

by

Dafne Linette Badillo Campuzano A01275298

Andrea Múñoz Gris A01733058

Data Structure

Daniel Pérez Rojas

June 10th, 2020

IMPLEMENTATION

1. First we create the AVL tree, with different functions.

* With a getBalanced function, this if for getting our BST tree to a AVL with a balance factor, this return an integer with the difference of the height between both sides.
* The function max, this is to compare the two nodes.
* Rotations, we use two functions one for the left and other for the right.
* Height, with the height we can know the node height.
* With the function Insert, we insert a node in the tree and make it balanced, using the other functions, actually this is the most complicated for the avl tree, however we saw it in out zoomclass.

1. We use linked list to add and removes nodes using pointer to pointer.

-newSpecial, newList, with this function we create the linked list.

-push,we use pointer to pointer (for going to the head) and inserts a new node on the front of the list.

-deletedList, this delete the node on the front of the list.

-swap, with this function we exchange the position of two nodes.

-printList,this prints the content contents of the linked list starting from head.

1. We also use stacks and queue for the implementation, actually we were use this data structures, to make it easier some methods in the program, like “edit entity” and “delete entity”.
2. The pointers that we implement in the program, have a big responsibility to connect the methods between them.
   1. For example, we use the pointer, head, pth, root, this for the AVL tree structure.
   2. Also, the pointers input and output, output is used to create the type FILE, this to save a backup of the information, and the input is used to call the type FILE, to use an existing file.
      1. This are implemented in the method “admintools” corresponding to the option 7, also is used at the end of the program, when you choose the option 0 to exit, the method “admintools” save the data in file with extension .txt.

DIFFICULTIES

An AVL tree stores in each node the height of the subtrees rooted at this node. Then, for any node, we can check if the height is balanced: that the height of the left subtree and the height of the right subtree differ by no more than one. This prevents situations where the three gets too lopsided.

balance(n) = n.left.height – n.right.height

-1 <= balance(n) <= 1

When you insert a node, the balance of some nodes might change to -2 or 2. Therefore, when we “unwind” the recursive stack, we check and fix the balance at each node. We do this through a series of rotations.

Rotations can be either left or right rotations. The right is an inverse of the left rotation.

We experiment difficulties mainly in the part of structures, firstly because we decided to implement past tasks, for example the AVL Homework. We take the structure of the tree, but we have to do some modifications. For example, to put the ID using strings was difficult. Specially when we want to delete and recover the identity. Other difficulty we had was with the Search, specially with the displation of the values.

But the most difficult part of the program was to implement more pointers with stack and queues, this working together in the structure of the AVL tree. Because sometimes, when we prove the program, the program presents a “Segmentation Fault” or some Warning at the moment to compile, so for this reason we have to be so careful with the specifications of the parameters.

CONCLUSION

This final project was very challenging, we can apply all the knowledge that we see in class, including all the topics. We found the technical implementation of using Linked list, queue, and stacks all together really hard, but despite that, everything else went great. The online classes recorder was very useful to reinforce the knowledge to apply in this project.

BIBLIOGRAPHIC REFERENCES

Gayle LaakMann McDowell. (2016). Advanced Topics. En Cracking the Coding Interview(637,638). Palo Alto, CA: CareerCup.

Rajput-Ji. (2017). Linked List | Set 2 (Inserting a node). 9 de junio de 2020, de geeksforgeeks Sitio web: <https://www.geeksforgeeks.org/linked-list-set-2-inserting-a-node/>

Ritchie, D., Kernighan, B. (1988). The C programming language. 263 pp. New Jersey: Pretince Hall.