Laboratory practice No. 4: Hash Tables and Trees

Alejandro Mc Ewen
Universidad Eafit
Medellín, Colombia
amce@eafit.edu.co

Felipe Henao
Universidad Eafit
Medellín, Colombia
fhenaog3@eafit.edu.co

3) Practice for final project defense presentation

3.1 For the collision of bees we used a hash table where it was defined that the points were divided by 100 and rounded down. This made that all numbers from –100 to 100 were put into the point 0. This groups the bees into sections making the search the nearest bees much faster and less time expensive.

3.2 The family tree cannot be implemented more efficiently so that the search and insertion is in logarithmic time because there is no way to order the members of the family.

3.3

2.1 To find the pos-order path of a tree binary search tree having the pre-order path what we do is build the tree again from the pre-order, knowing that the root is the first element and adding the rest of the elements bases on how a binary search tree is built, the left subtree of a node only contains nodes with lower values than the value of the node and the right subtree of a node only contains nodes with higher values than the value of the node. Then, to make the pos-order, we call a function recursively until it reaches the last node on the left on the tree, and then see if there is anything on the right and do it recursively until we reach a leaf of the tree, then we and then we save the node on the left and the node on the right along with the node that is parent to those nodes on a string in that order (left-right-parent), and then, recursively, we do the same for the right of the parent of that subtree and join the string of the left subtree with the string of the right subtree and then the parent of those subtrees, and all of that is done until we reach the root.

2.2 For this problem we did all the possible sum of the tree. We did this with a recursive algorithm that would go summing the tree branches until it got to the leaves. Then we checked if the sum was equal to the target and return the result all the way up the 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

3.4/3.5

Exercises	Complexity
2.1	O(n*log(n)) n: number of nodes in the tree
2.2	O(n) n: number of nodes in the tree

4) Practice for midterms

4.1.1 b

4.1.2 d

4.2.1 Nearest common ancestor

4.2.2 O(n) because it goes through all the nodes once.

4.2.3 Add ifs to see that if the value is higher there is no need to check that value on the left and if it is lower, we don't need to check the value on the right.

4.3.1 return True

4.3.2 O(n) because you go through two arrays of length n. This happens because for the worst case to happen both arrays have to have the same length

4.4.1 a

4.4.2 a

4.4.3 d

4.4.4 a

4.5.a false

4.5.b p.data < toInsert

4.6.1 d

4.6.2 return 0;

4.6.3 == 0

4.7.1 a

4.7.2 b

4.7.3 d

4.8 b

4.9 a

4.10.1 b

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

4.10.2 5

4.10.3 b

4.9.3 c

4.10.1 d

4.10.2 c

4.10.3 b

4.11.1 raiz.id

4.11.2 a

4.12.1 i

4.12.2 a

4.12.3 c

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