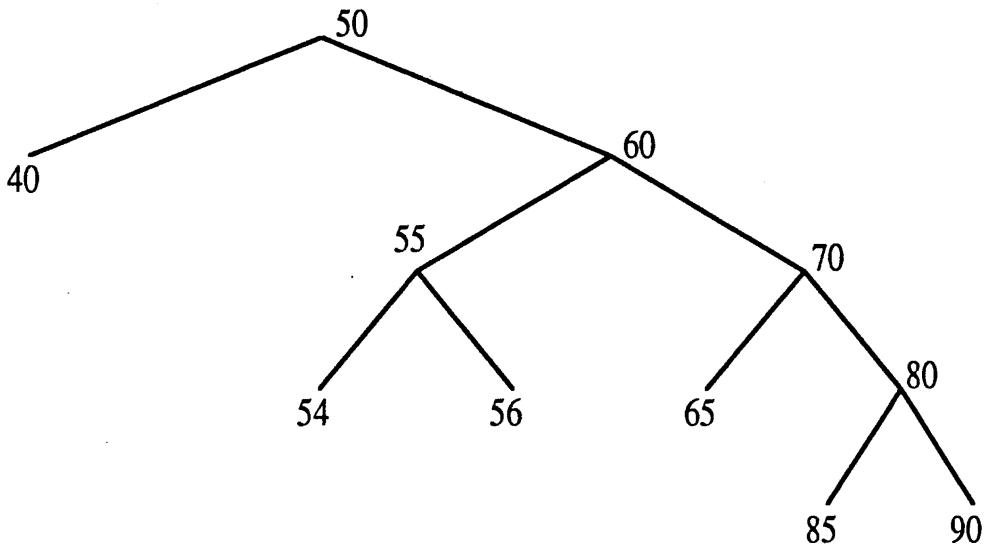
SOME REMARKS ON DELETING A NODE FROM A BINARY TREE

HENG Aik-Koan
Dept of Information Systems and Computer Science
National University of Singapore
Lower Kent Ridge Road, Singapore 0511

To delete an arbitrary node from a lexically ordered binary tree, the algorithm is given as follows:

1. Determine the parent node of the node marked for deletion, if it exists, note that it will not exist if we are deleting the root node.
2. If the node being deleted has either a left or a right empty subtree, then append this nonempty subtree to its grandparent node (that is, the node found in STEP 1) and EXIT.
3. Obtain the inorder successor of the node to be deleted. Append the right subtree of this successor node to its grandparent. Replace the node to be deleted by its inorder successor.This is accomplished by appending the left and right subtrees of the node marked for deletion to the successor node. Also, the successor node is appended to the parent of the node just delteted (that is, the node obtained in STEP 1).

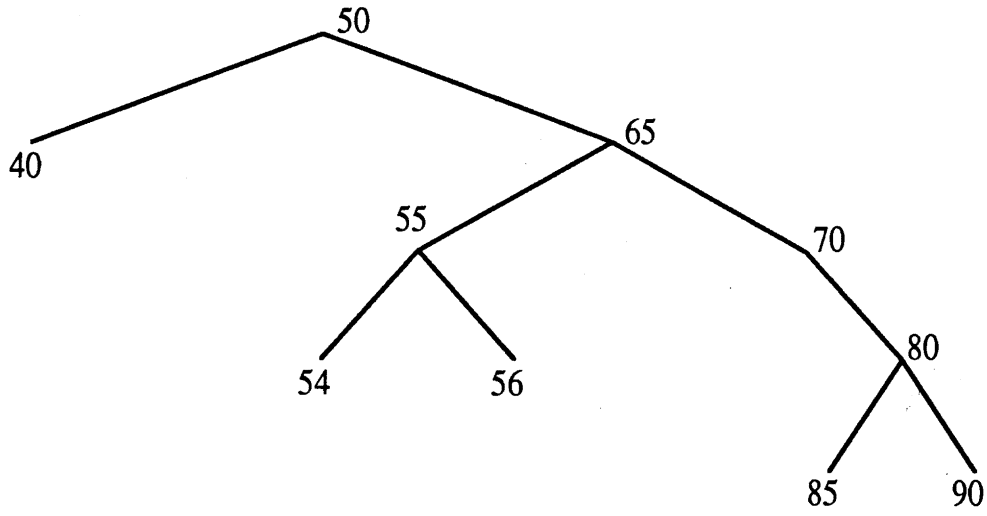
For example, given the following binary tree:



If we wish to delete node 60, according to the algorithm, we should replace 60 by the inorder successor which is 65.

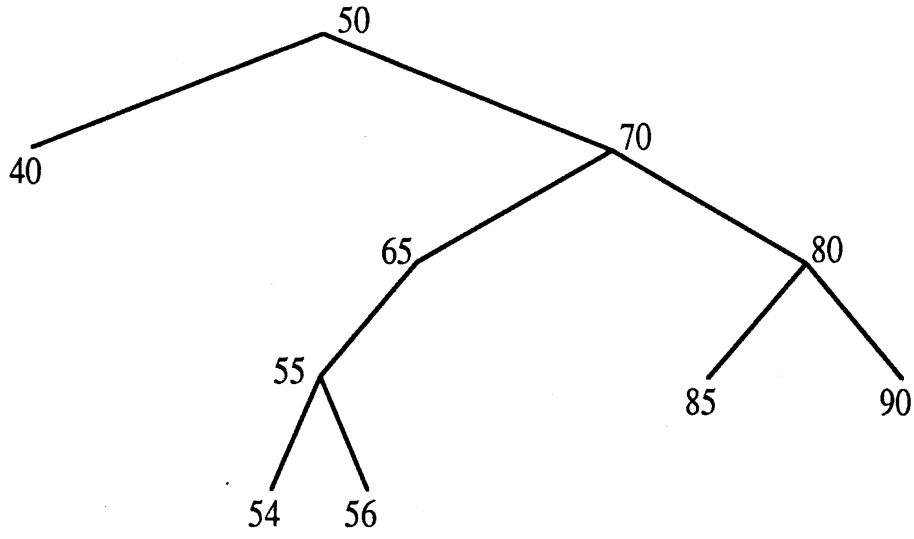
But if our purpose is to obtain just a lexically ordered tree, then the replacement of this deleting node could be any node on its right subtree i.e. 65,70,80,85 and 90 in our example.

If we consider the number of changes in the deletion, using the algorithm, the deleting node is replaced by the inorder successor node, the left and right subtrees of the deleting node will be appended to this successor node, and the parent of this inorder successor node has now an empty left subtree. The resulting tree becomes, in our example,as follows:



1 replacement + 3 changes

But if we replace the deleting node by the root node of its right subtree. Then the left subtree of the deleting node will be appended to the left subtree of the replaceing node. In our example, the resulting tree becomes:



1 replacement + 1 change

In the second case, the height of the resulting tree could be modified and become more unbalanced, all these will depend on the structure of the left subtree of the deleting node, which is a disadvantage.

REFERENCE

1. Tremblay and Sorenson, An Introduction to Data Structures with Application, (2nd edition) McGraw-Hill, 1984.