ID 2550814

Exam for Data Structures, Algorithms, and Databases

After inserting your student ID and the module name in the title, header, and footer, write your answers between here and the statement of good academic conduct. Your ID and the module name will automatically appear on any subsequent pages.

QUESTION 1

(a) Neither of the trees is perfectly balanced as both have nodes in their trees that do not have zero balance. For example, in the first tree, the balance at node 2 is 1, and in the second tree, the balance at node 1 is -1. Therefore, both trees are not perfectly balanced.

(b) AVL trees are defined as binary trees that only consist of nodes with a balance of either -1, 0, or 1. For the first tree, the balance at node 8 is -2, so it is not an AVL tree.

For the second tree, all the nodes have a balance of -1, 0, or 1. For example, the balance at node 3 is -1. Hence, it is an AVL tree.

(c) Assuming that all the nodes in the tree have a balance of zero, the maximum number of nodes in the tree would be that of a perfectly balanced tree. The formula to find the number of nodes in a perfectly balanced tree is $2^{h+1} - 1$, where h is the height of the tree. For the tree with a height of 6, the maximum possible number of nodes in this variant would be 127.

(d) By modifying the formula used to find the number of nodes in an AVL tree such that the balance of the nodes can also extend up to 2, we get T(h) = 1 + T(h-1) + T(h-3), where T(x) refers to the number of nodes in the tree/subtrees. Substituting h as 6, the minimal number of nodes in this variant of the binary search tree would be 18.

QUESTION 2

(a) Once element 76 is deleted, there will be an empty node at the root of the AVL tree. This is replaced by the leftmost node of the right subtree, which is 82. The resulting tree is shown in Figure 1:

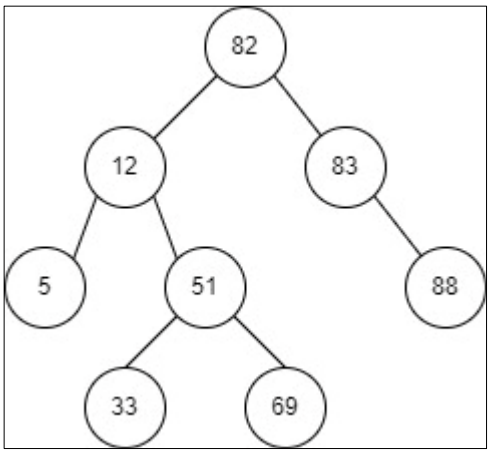

Figure 1: Resulting AVL after deletion of element 76. [1]

Since the balance of each node after the deletion is still -1, 0, or 1, there is no rotation required.

(b)

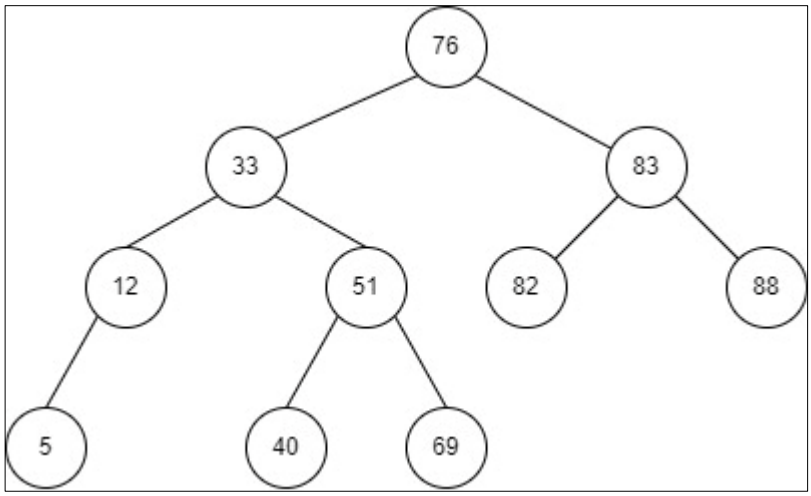

Figure 2: Final AVL tree after insertion of element 40 and RL rotation. [1]

Once element 40 is inserted into the AVL tree (Figure 3), the balance at node 12 becomes -2. To fix this, RL rotation is performed. First, R rotation at node 51 is done as shown in Figure 4. Then L rotation is done at node 12, giving the final balanced AVL tree in Figure 2.

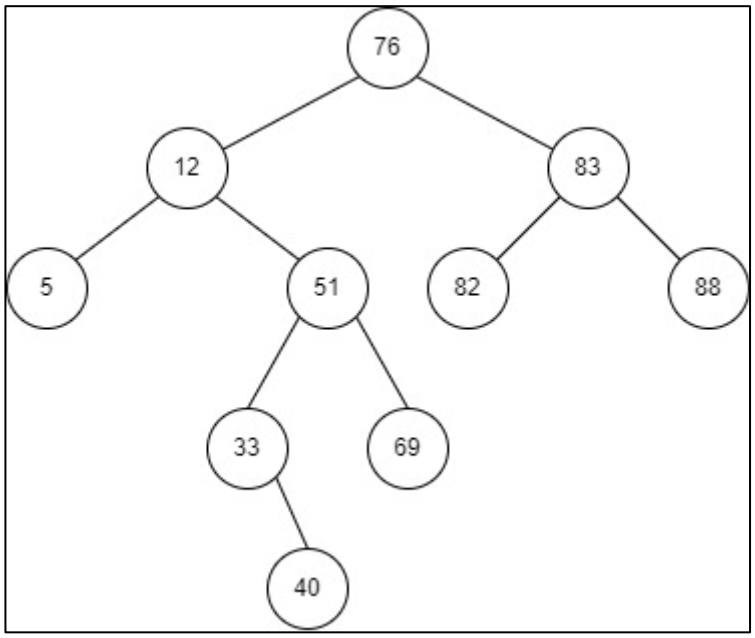

Figure 3: Tree after inserting element 40. [1]

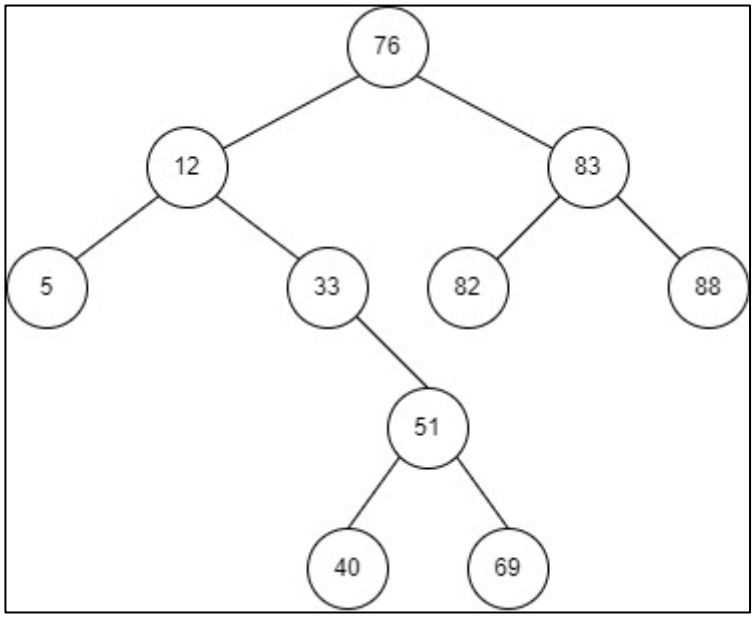

Figure 4: Tree after performing R rotation. [1]

QUESTION 3

(a) The algorithm takes an array a (may be sorted or unsorted) of length n and sorts it into a binary search tree T. After inserting all the elements of the array a into the tree T, the algorithm calls the visit function to construct a list L with the elements in the tree. The list L will contain the elements of the array a in a sorted descending order.

(b) For the invariant and post condition to hold true, all the elements in array a must be inserted into the binary tree T without exceptions. If the insert function fails to insert an element from the array a into tree T, then the #nodes(T) function would return a value that is less than i, which would make the invariant not true. Subsequently, the postcondition would also be not true.

Additionally, the visit function should add all the elements from the tree T to the list L without any errors. If the visit function is not performed correctly, then the length of list L will be different from the length of array a. This will lead to the failure of the postcondition.

(c) Assuming the result of doSomething is a regular binary search tree, the worst-case time complexity of the insertion step would be O(n). Since the algorithm performs an insertion for all the n elements of the array a, the worst-case time complexity of the algorithm would be $O(n^2)$. In addition, considering that the visit function runs through all the elements of the tree T, its time complexity is O(n). This still leads to the algorithm having an overall worst-case time complexity of $O(n^2)$.

References

[1] “Diagrams.net,” [Online]. Available: https://app.diagrams.net/. (used for making AVL tree representations)

Do not write below this line

Statement of good academic conduct

By submitting this assignment, I understand that I am agreeing to the following statement of good academic conduct:

- I confirm that this assignment is **my own work** and I have not worked with others in preparing this assignment.
- I confirm this assignment was written by me and is in my own words, except for any materials from published or other sources which are clearly indicated and acknowledged as such by appropriate referencing.
- I confirm that this work is not copied from any other person's work (published or unpublished), web site, book or other source, and has not previously been submitted for assessment either at the University of Birmingham or elsewhere.
- I confirm that I have not asked, or paid, others to prepare any part of this work for me.
- I confirm that I have read and understood the University regulations on plagiarism (https://intranet.birmingham.ac.uk/as/registry/policy/conduct/plagiarism/index.aspx).