15. Under what conditions are the Fetch and Insert operations performed on a binary search tree structure fast?

The fetch and insert operations are faster when the tree is balanced. That means that the left and right subtrees of each node are close to the same size.

16. State the three cases of the binary search tree Delete algorithm.

a. Case 1: No children.

b. Case 2: One child.

c. Case 3: Two children.

17. The *linked implementation* is used in the coding of a Binary Search Tree structure. Calculate the structure's density assuming that it contains 200 nodes and:

a. Each node contains 10 bytes of information. D = 0.45

b. Each node contains 300 bytes of information. D = 0.96

18. The *array implementation* is used in the coding of a Binary Search Tree structure. Calculate the structure's density assuming that it contains 200 nodes and:

a. Left balanced and each node contains 10 bytes of information. D = 0.71

b. Skewed to the right and each node contains 10 bytes of information. D < 10-27

c. Left balanced and each node contains 300 bytes of information. D = 0.99

b. Skewed to the right and each node contains 300 bytes of information. D < 10-25

19. A balanced binary search tree stores 1,000,000 nodes. Assuming a memory access takes one nanosecond, calculate the average time required to fetch a node from the tree for the:

a. Linked implementation. 63.9 ns

b. Array-based implementation. 39.9 ns

20. Give the output produced by an NLR output scan of the tree shown in Figure 7.3f.

D B E A F C G

21. State the advantage of an AVL tree over a Binary Search Tree Structure.

The tree is kept balanced so that the search algorithm is as fast as possible.

22. Give the AVL balance factors for each of the nodes in Figure 7.3a.

A: 1 B: -2 D: 1 E: 0

F: 0 G: 0 H: 0

23. Of the 11 nodes in an AVL tree, 6 of the nodes have balance factors of 1, 4 nodes have balance factors of 0, and 1 node has a balance factor of −1.

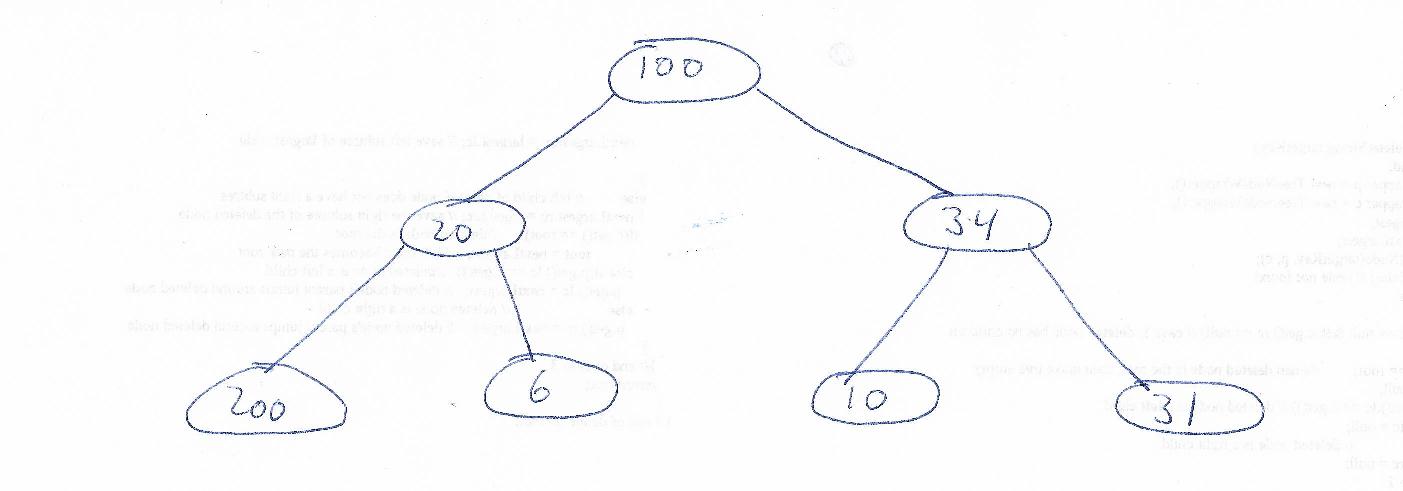
I've been trying to draw this tree. I believe it is impossible to draw. There are too few zeros. I do know this: It can't go above level three. The minimum number of nodes to reach level four is 12. (There is a simple proof of this that I found several different places online including geeksforgeeks.)

Here's the proof for this problem: We need 11 nodes. The total number of possible nodes for all the levels <=3 is 15. That means we need exactly 3 levels (four counting the root). Any tree which satisfies all three requirements must have all four nodes in level 2 filled (leaving a node out of level 2 and putting the remaining nodes in level 3 always gives an unbalanced tree). That leaves four nodes for level three. They can be placed in any of the eight slots in level three and still have the tree be balanced. There 35 possibilities but many of them are nearly similar after reflections. None have fewer than six zeros. (All of those last four leaf nodes are zeros which accounts for all we have. That would mean that NONE of the remaining nodes can be zero.) For instance, the tree in figure 7-22 has 11 nodes and eight with balance factors equal to zero.

a. Is the tree balanced? Yes, the tree, if it exists, is balanced by definition.

b. Is the tree complete? No.

24. An array contains the values 100, 20, 34, 200, 6, 10, and 31. Element 0 contains 100, element 1 contains 20, etc. Draw the binary tree representation of the array.



25. In the array implementation of a binary tree:

a. Where is the root node stored? At node 0.

b. Where is the right child of the node in element 10 stored? 22

c. How can we determine if the node stored at element 22 does *not* have a left child?

Node 45 will be null.

26. True or false, a binary tree containing 20 nodes can always be stored in a 20 element array?

False. There are many exceptions.

27. Give the size of an array that could store *any* binary tree containing 10 nodes.

1023 nodes is the smallest array. An array which is smaller could not accommodate a tree with ten nodes which are entered in sorted order.