William Lin

CSCI 2226

April 27, 2016

Exam

1a. Trees 2 and 3 are not binary search trees because each one of them has a left child that is greater than the parent node, therefore making it not a binary search tree.

1b. Tree 2 is not a min heap because the 3rd level nodes are not all the way to the left like they should be. The nodes 17 and 37 should be children nodes of 6.

1c. Yes, it is balance, but by adding a 13 as a child node of 12, it would make it unbalanced.

1d. 2, 3, 5, 8, 17, 6, 37, 11, 12

1e. The resulting tree will have bad property because it is unbalanced. It will occur because when you delete the minimum elements you are left with just the max elements which will make one side of the tree larger than the other.

2a. You would add a black node with the value of 30 as the right child node of 25. And now the node 30 will have two null black nodes.

2b. 3 before a re-balance operation occurs because that is how many nodes are left that can be inserted at this level. I would insert 21, 85, and 90.

2c. Yes, you can force a rebalance by adding another node as a child of 30. If you add 35, this would cause the tree to become un-balanced.

2d. An AVL and a Red-black tree are self-balancing trees that are always balanced. Unlike binary tress, being balanced all the time give AVL and Red black trees an advantage in Big-O time.

3a. Add 13 to the end of the heap. But 13 is smaller than 14 which is its parent node. There 13 and 14 swap places. So the end heap is:

2, 4, 3, 5, 6, 3, 8, 29, 13, 25, 19, 32, 61, 21, 16, 30, 38, 15, 14

3b. You swap the last node in the heap with the top node and then you can go ahead and delete the last node in the heap which is technically the top item on the heap. Because of this the top node is 14 which is not in the right place because it is not the smallest compared to 4 and 3. So we take the smallest of the two children node which is 3 and we swap places with it. But 14 is still not in the right place because the children of 14, which are 3 and 8, are still smaller than 14. So we take the smallest of the two children node, which is 3 and we swap places with it. Now 14 is in the right spot. The resulting heap looks like this:

3, 4, 3, 5, 6, 14, 8, 29, 13, 25, 19, 32, 61, 21, 16, 30, 38, 15

4a. w.Pivot: 40, 38, 91, 30, 74, 25, 97, 63, 23, 25, 71, 49, 35, 68, 29, 41

4b. Steps:

40, 41, 38, 91, 30, 74, 25, 97, 63, 23, 25, 71, 49, 35, 68, 29

29, 40, 41, 38, 91, 30, 74, 25, 97, 63, 23, 25, 71, 49, 35, 68

29, 40, 41, 68, 38, 91, 30, 74, 25, 97, 63, 23, 25, 71, 49, 35

35, 29, 40, 41, 68, 38, 91, 30, 74, 25, 97, 63, 23, 25, 71, 49

35, 29, 40, 41, 68, 49, 38, 91, 30, 74, 25, 97, 63, 23, 25, 71

35, 29, 40, 41, 68, 49, 71, 38, 91, 30, 74, 25, 97, 63, 23, 25

25, 35, 29, 40, 41, 68, 49, 71, 38, 91, 30, 74, 25, 97, 63, 23

23, 25, 35, 29, 40, 41, 68, 49, 71, 38, 91, 30, 74, 25, 97, 63

23, 25, 35, 29, 40, 41, 68, 49, 71, 63, 38, 91, 30, 74, 25, 97

23, 25, 35, 29, 40, 41, 68, 49, 71, 63, 97, 38, 91, 30, 74, 25

25, 23, 25, 35, 29, 40, 41, 68, 49, 71, 63, 97, 38, 91, 30, 74

25, 23, 25, 35, 29, 40, 41, 68, 49, 71, 63, 97, 74, 38, 91, 30

30, 25, 23, 25, 35, 29, 40, 41, 68, 49, 71, 63, 97, 74, 38, 91

30, 25, 23, 25, 35, 29, 40, 41, 68, 49, 71, 63, 97, 74, 91, 38

38, 30, 25, 23, 25, 35, 29, 40, 41, 68, 49, 71, 63, 97, 74, 91

5a.

Prefix: * 3 + x 2

5b.

Postfix: 2 3 * x +

5c.

Infix: y = (x + 2) * 3