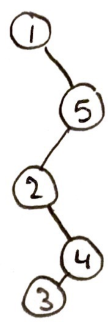
Jonathan Tso
CS 251
Homework 3

1.

Part A:

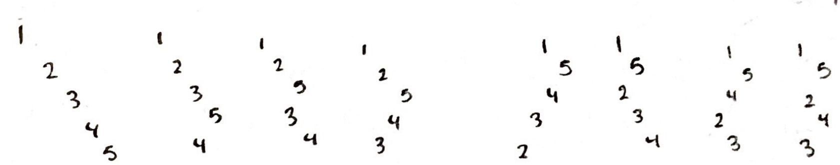


Part B: 1, 5, 2, 4, 3

Part C: The first insertion to disturb AVL tree properties is the ② insertion, because the root ① has left height ∅ and right height 2.

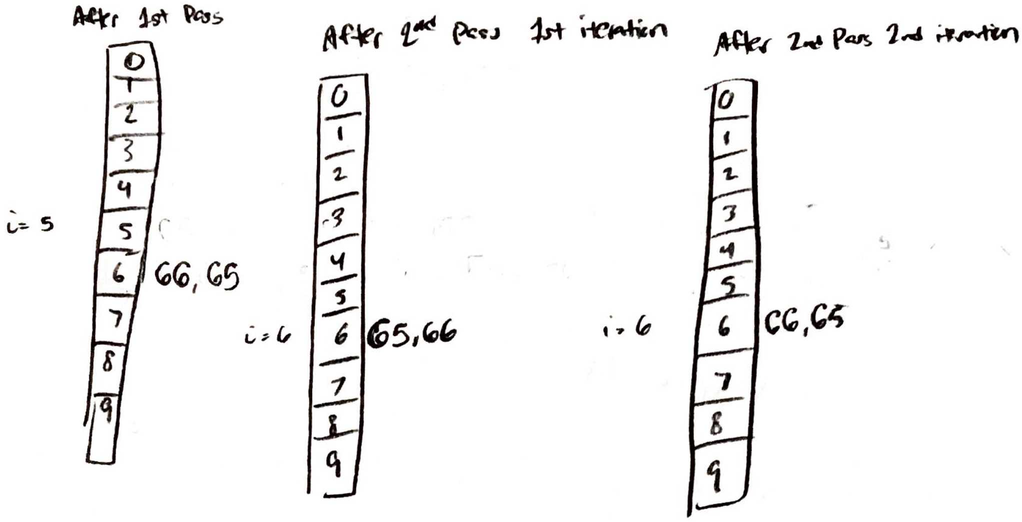
Part D: To have a height of 4, everything below the root must be either greater or less than the root, so the root must be ① or ⑤.

When we enumerate possibilities for where root is ①, we get the following



In each of these, we can see that the second number in the tree also dictates the rest of the tree, because then each value corresponding must also be less than or greater than the next value. It must be the next extreme. For example, after 1, it must be 2 or 5. Then we see this trend continue. The following number must be the next extreme, and this continues the pattern. Thus, for root 1, each following has 2 options (one for each extreme). Then, for every level down, we have 2 options, thus, for a four height tree, we have 8 trees for root 1 and 8 trees for root 5, so we have a total of 16 trees.

Problem 2



Consider the 2 input values of 66, 65 in that order. The above shows that on the left after we have gone into the 10's place, we have move 65 from the 5's box to the 6's box. Then, we move i = 6. At this point, we run through 2nd pass on 66 and pop and push it back to the 6 box. Then, we do 2nd pass on 65 and pop and push to the back. This sets a discrepancy such that when we extract this, the ordering is incorrect.

The problem stems from when we look at numbers that initially end up in the wrong order before you run the pop/push on the i^{th} box. In the above case, we have 66 before 65 because we move 65 prior to the 6 box but not 66. Then, because we update everything up to the length of the box, we will always retain the incorrect order. Thus, the given one-array radix sort will not work for general case sorting.

Problem 4.

Base Case: $h=0$, with node root R

R

By observation: This has 1 node

By formula: $\dfrac{3^{h+1}-1}{2} = \dfrac{3^{0+1}-1}{2} = 1$

Thus, for base case, it holds true

IH: Consider that for given height K, that the number of nodes is determined by $\dfrac{3^{K+1}-1}{2}$, where $K \geq 0$.

Prove: That for a height K+1, the above will hold true that

$$\frac{3^{(K+1)+1}-1}{2}.$$

Consider that for every level deeper into the tree when we add a height, or $(K)+1$, we ~~are~~ adding in a set of 3 nodes for each original root.

Thus, we can think of it such that when $(K+1)$ is the height, our nodes become $\text{Root} + 3 \cdot \dfrac{(3^{K+1}-1)}{2}$ where the root is just 1 node, so Root $= 1$.

Then, we have the following:

$(K+1)$ height has

$$\text{Root} + 3(\#\text{ nodes})$$

$$= 1 + \frac{3(3^{K+1}-1)}{2} \quad \text{by IH}$$

$$= 1 + \frac{3^{K+2}-3}{2}$$

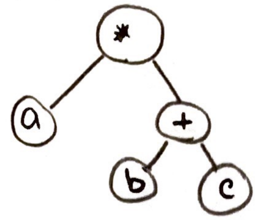
$$= \frac{2}{2} + \frac{3^{K+2}-3}{2}$$

$$= \frac{3^{K+2}-1}{2}$$

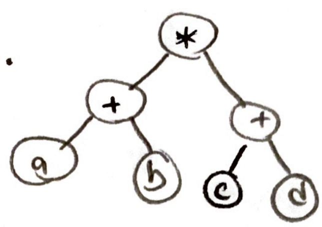
Thus, we have proven that the inductive hypothesis holds true.

Problem 5.

1.

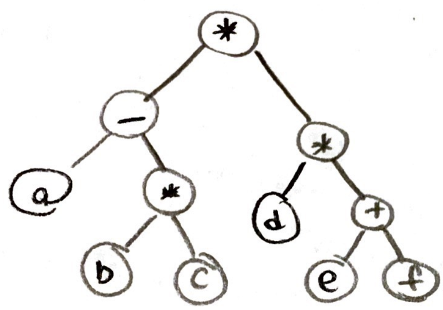


2.



Part A

3.



1. * a + bc
2. * + ab + cd
3. * - a * bc * d + ef

Part B

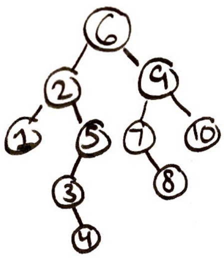
1. a bc + *
2. ab+ cd+ *
3. a bc* - def+ * *

Part C

Problem 6.

6, 9, 2, 1, 5, 7, 10, 8, 3, 4

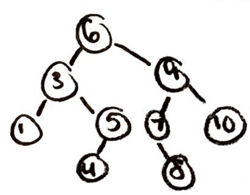
a)



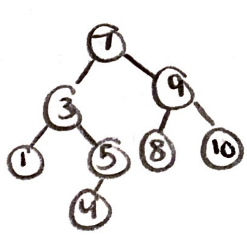
b)

Pre-order	Print(#) Left Right	6, 2, 1, 5, 3, 4 9, 7, 8, 10
In-Order	Left Print(#) Right	1, 2 3, 4, 5, 6, 7, 8, 9, 10
Post-Order	Left Right Print(#)	1 4, 3, 5, 2, 8, 7, 10, 9, 6

c) We take and copy the lowest value from the right subtree, which is 3 (b/c 2 has 2 subnodes). We then replace the 2 with the new value 3, which preserves the tree. Then, delete the old 3 node and move the connection to be between 5 and 4.



We follow the same approach, because 6 has 2 subnodes. We look for the smallest value in the right subtree, which is 7. Then we replace the 6 and then delete the 7 node (old one) and relate 8 to 9. This will retain the tree such that all left is < 7 and all right > 7.

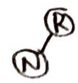


Problem 3.

Base Case: Where non-empty tree and the only root node R

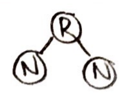
By observation, there is not a full node in the tree and 1 leaf.
By formula, # FN (full nodes) = # L (leaves) − 1 = 1 − 1 = 0

Case 2: consider where full nodes based off root node R has 1 node below N



By observation, we can see still no full nodes, but we do have 1 leaf N,
By formula, we can see #FN = #L − 1 = 1 − 1 = 0

Case 3: Consider where full nodes based off of root node R which has 2 nodes below N



By observation, we can see we have 1 full node R and 2 leaves N
By formula, #FN = #L − 1 = 2 − 1 = 1

Thus, for base case, the two are equivalent.

IH: Consider for a number of K leaves, that the number of full nodes = K − 1, where $K \geq 1$.

Prove: For (K+1) leaves, our IH still holds true and #FN+1 = (K+1) − 1 leaves

Observe that for all binary trees, each terminal and non-terminal node can provide a number of full nodes that sum up to the total #FN. Every node can be summed up as the total number of full nodes on the left and right. We declare any left pattern of nodes to be L_1 and the right pattern as L_2. This can be recursively done until we reach terminal nodes. When there is a L_1 and L_2, we also account that their root node is then a full node, thus, #FN for a root that has an L_1 and L_2 is $\#FN_1 + \#FN_2 + 1$ (the base root). Thus, $FN = \#FN_1 + \#FN_2 + 1$

$FN + 1 = (\#L_1 - 1 + \#L_2 - 1) + 1$
$= \#L_1 + \#L_2 - 1$, still only off by 1

Because we can determine each possible base root of a binary tree and have proven that these make up all possible binary trees and also still follow our IH, we have proven that for any k+1 leaves, our IH will still hold no matter the makeup of the tree.

If we were to add to our N node in case 2 above, N would lose the status of leaf and so we would only have 1 leaf still, meaning we do not change # FN.

If we were to add to our R node in case 2, we end up with case 3, which we have proven does not change our formula and we have 1 new leaf and then consider R to be a full node.