Bryan Arnold

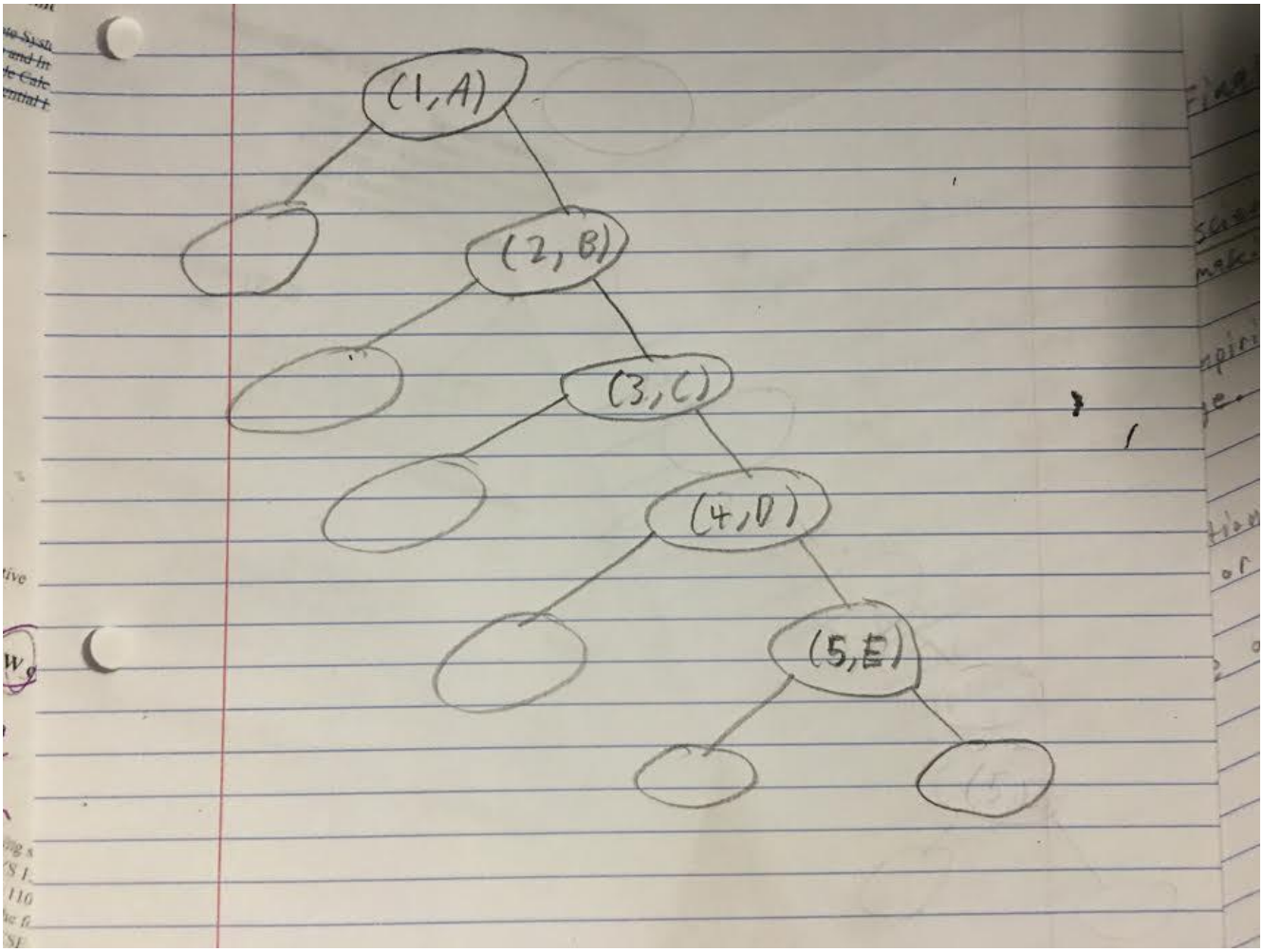
CSE 2100

11/20/16

Homework 2

[R-11.1]

After insertion of the entries (1, A), (2, B), (3, C), (4, D), and (5, E) into an empty binary search tree, the tree will look like the following:

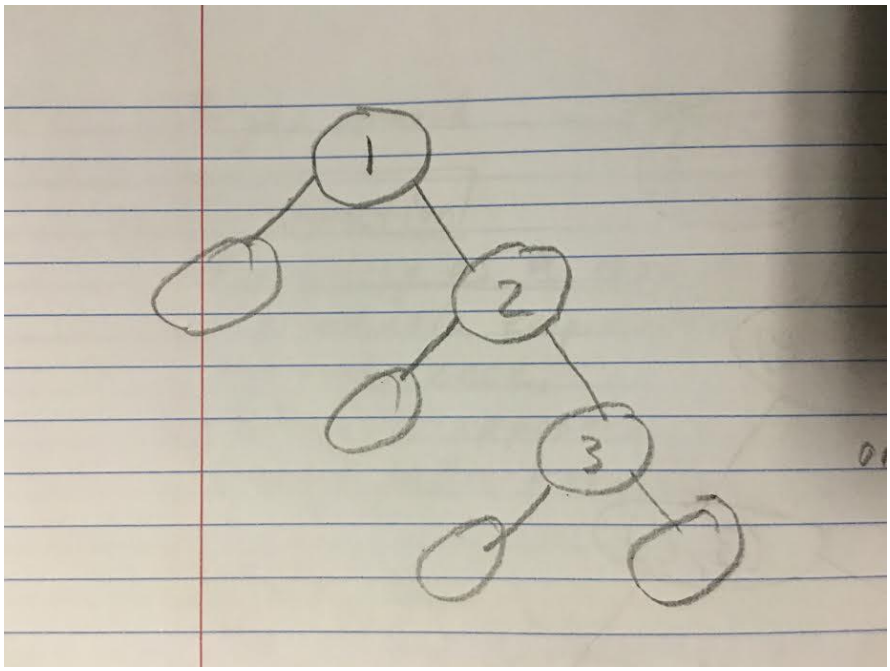


In an empty binary search tree, the first node will be the root node of the tree. Since any entries with greater keys always go to the right of the parent node, these entries will be the right child of each node placed before it. All parents must have two children in a binary search tree as well, but since to entries were put in these, they are left black (null).

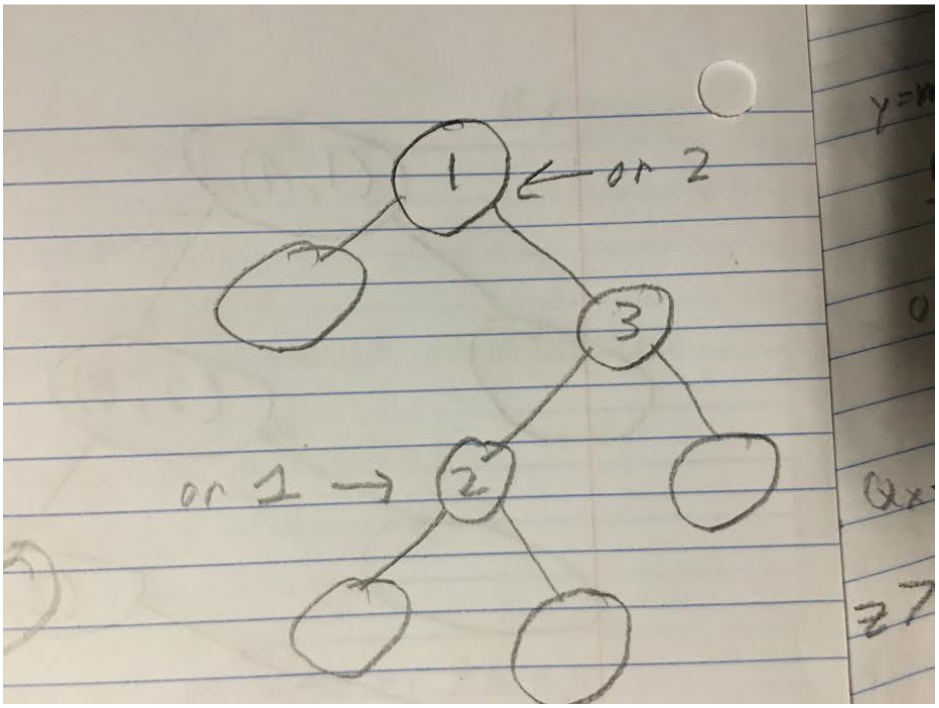
[R-11.3]

A binary search tree given three keys can have a total of 5 different trees. Below are the 6 possible orders that the keys that can be inserted into a tree and a picture of the tree is displayed afterwards. Two key insertion orders make the same tree, hence why there are not 6 possible trees and only 5. Any empty nodes are null (blank children in tree):

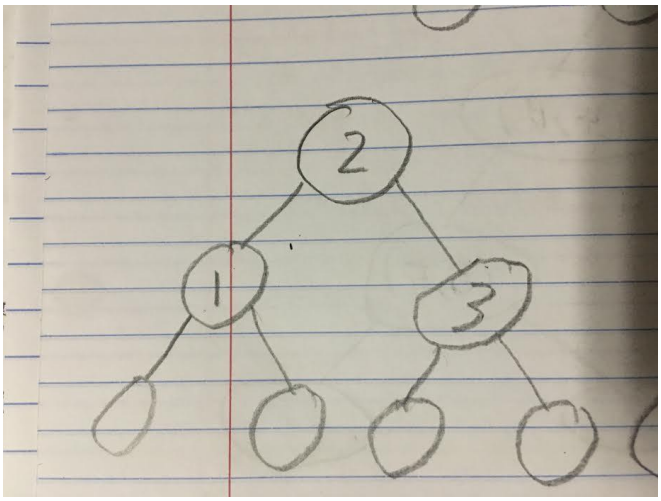
{1, 2, 3}



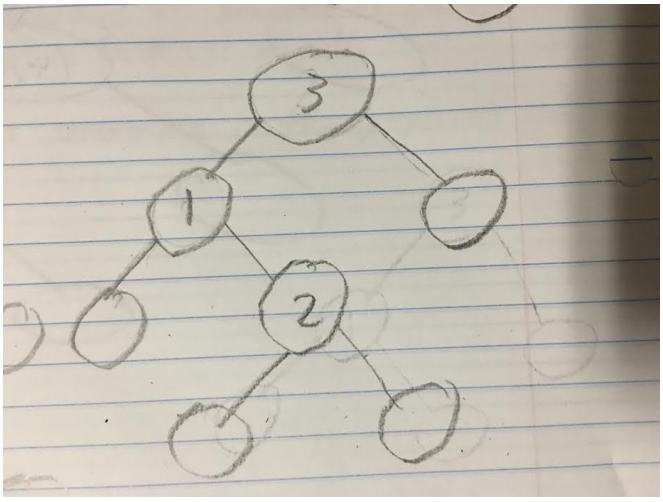
{1, 3, 2} / {2, 3, 1}



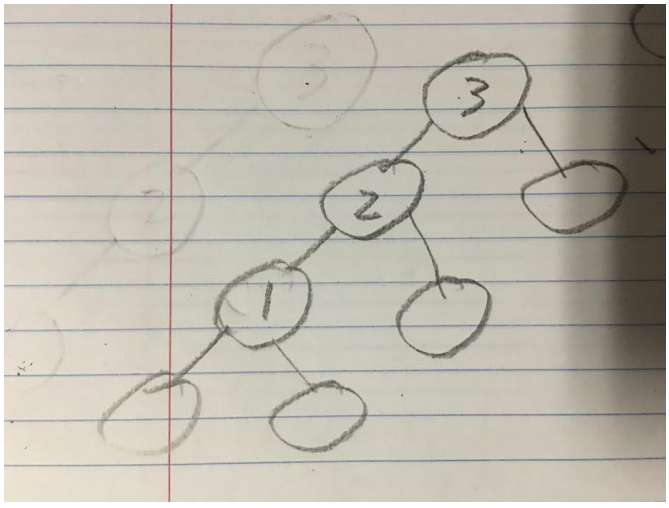
{2, 1, 3}



{3, 1, 2}



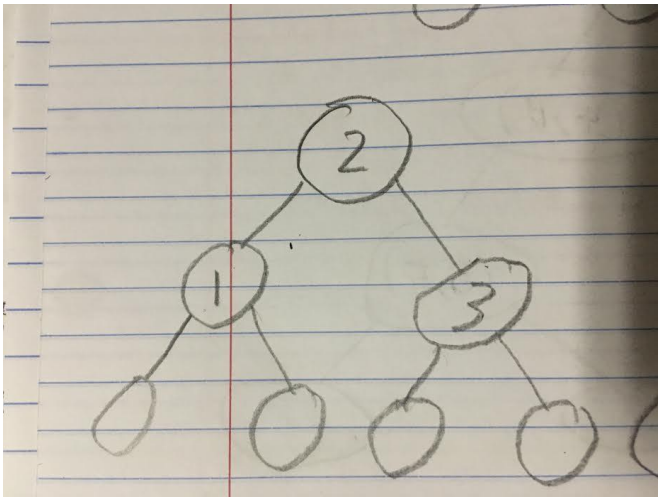
{3, 2, 1}



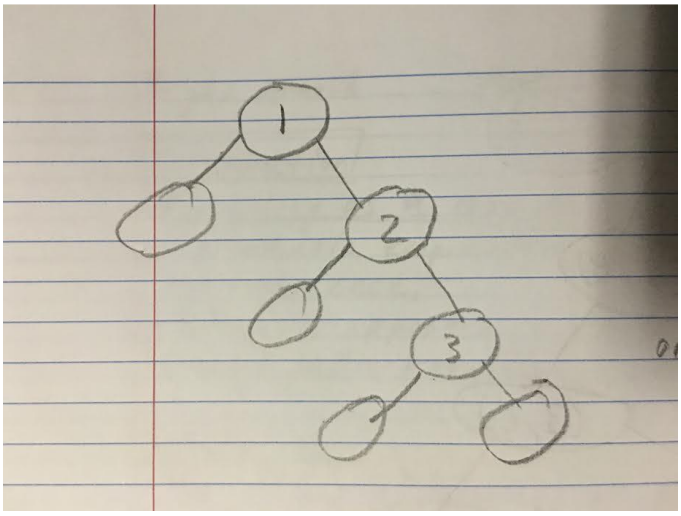
[R-11.4]

An example that proves Dr. Amongus' statement incorrect can be seen in two of the pictures in the previous problem, the insertion of {2, 1, 3} and {1, 2, 3}. These are both the same key values, but their ordering is different. If Dr. Amongus is correct, then the two trees should be the same, but they are not. The following pictures show this:

{2, 1, 3}



{1, 2, 3}

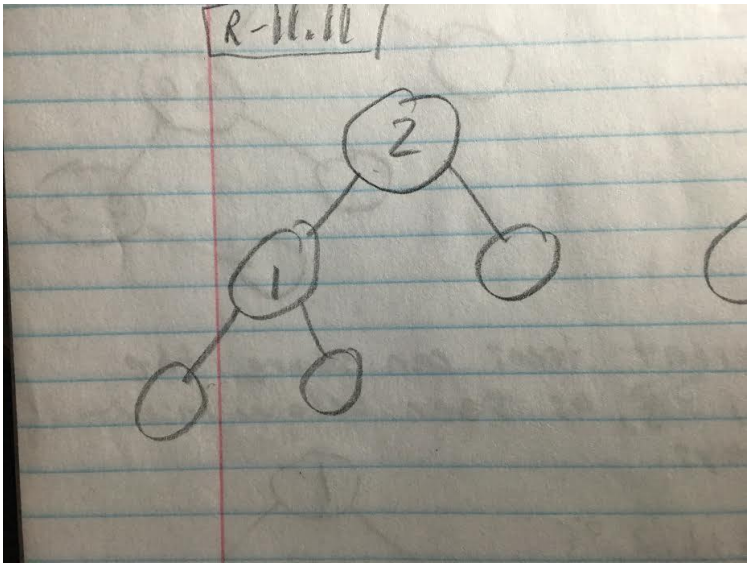


As you can see, both trees are proper binary search trees with the same key values, but their structures are different.

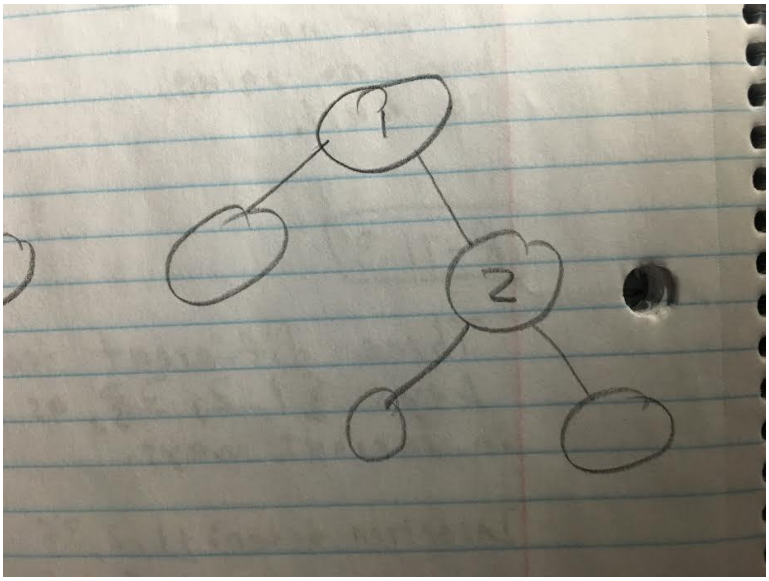
[R-11.5]

An example that proves Dr. Amongus' statement incorrect are the insertion of {2, 1} and {1, 2}. These are both the same key values, but their ordering is different. If Dr. Amongus is correct, then the two trees should be the same, but they are not. The following pictures show this:

{2, 1}



{1, 2}



Since AVL trees are just rebalanced binary search trees, the same principles for how insertion works applies when only adding two keys. So, the key values are the same, but the trees are different structures, proving Dr. Amongus wrong.

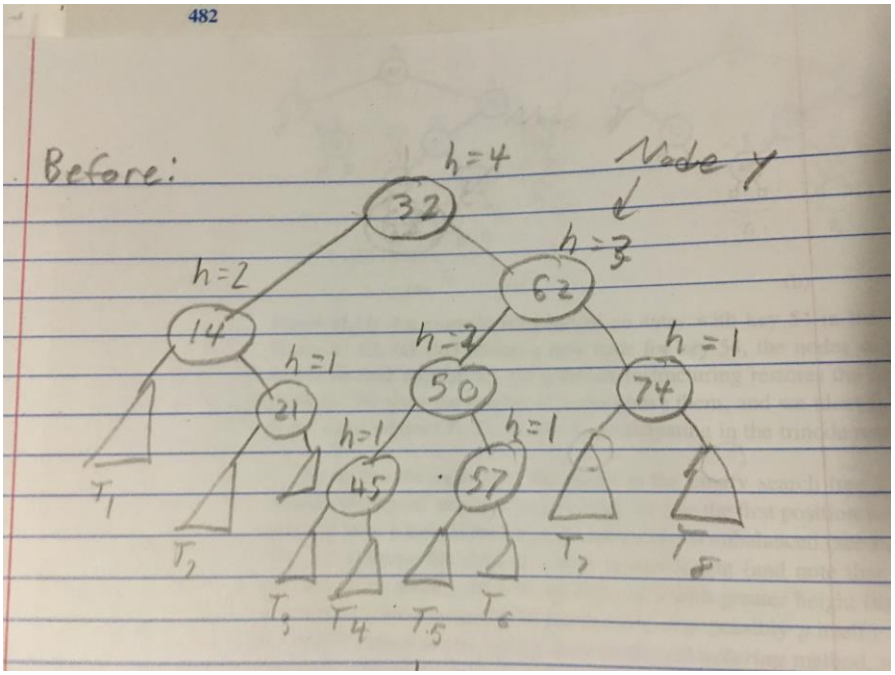
[R-11.7]

Figure 11.11 would need a double rotation to insert the new key 54, while figure 11.3 would only need a single rotation for its removal.

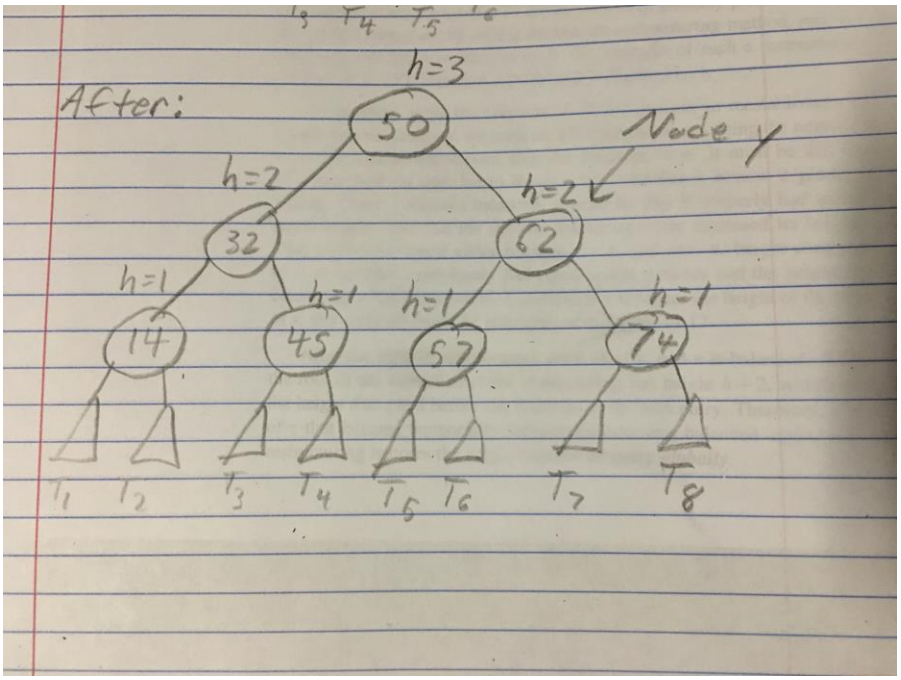
[R-11.11]

(All blank nodes are null)

Here is an example of an AVL tree before deletion:



The node y in the question is noted, it has two children with equal height as stated and is a subtree. The heights of each node are listed. Now, delete the node 21 to cause a trinode reconstruction:

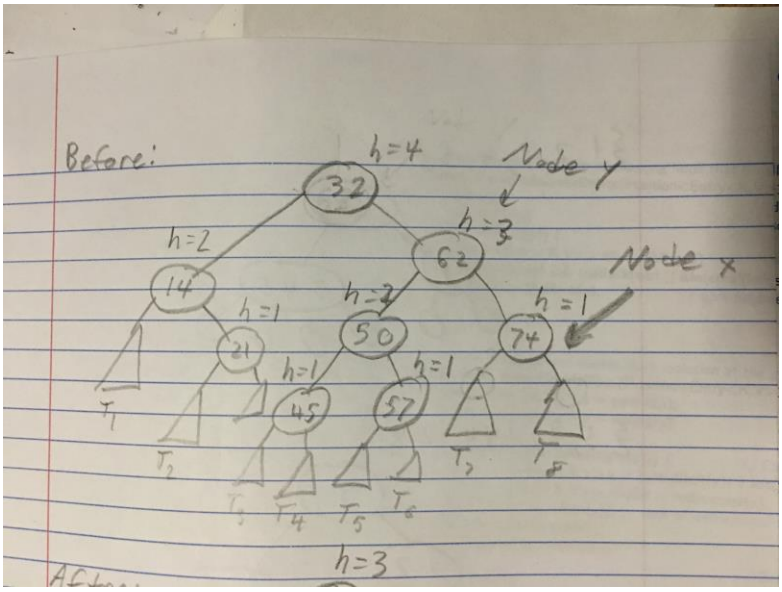


Notice the node y still has two children with them same height and is a subtree. The overall net change in the height of this tree is 1. Any node that had a height greater than 1, decreased by 1, and any node equal or less than 1 remained the same height. The total net loss of height, the combined loss of height of each node, is 4.

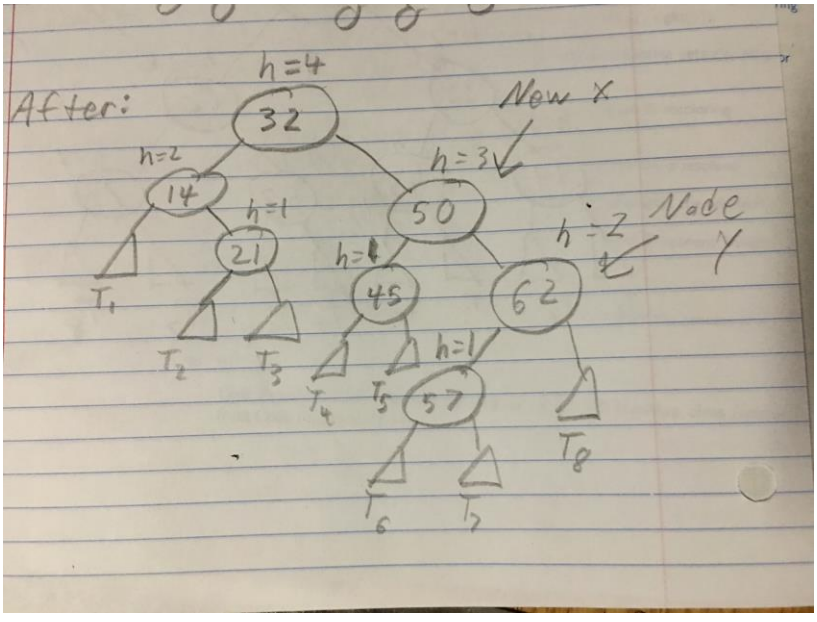
[R-11.13]

(All blank nodes are null)

Here is an example of an AVL tree before deletion:



This is the same tree in the previous question, except the child node x is outlined like specified in the question, it's the one the is aligned with the node y after deletion in the previous problem making it node x in this problem. Now, delete node 74:



Notice the node y still has two children and node x is now a new node that satisfies the requirement stated in the question. The overall net change in the height of this tree is 0. The could be a problem in restoring the AVL property because if the aligned child node x is has children, the property of having keys less than the parent on the left and greater than the parent on the right can be violated as well as the height difference change between children would be greater than one, making it impossible or nearly impossible to rebalance it to the height rule.

[R-11.17]

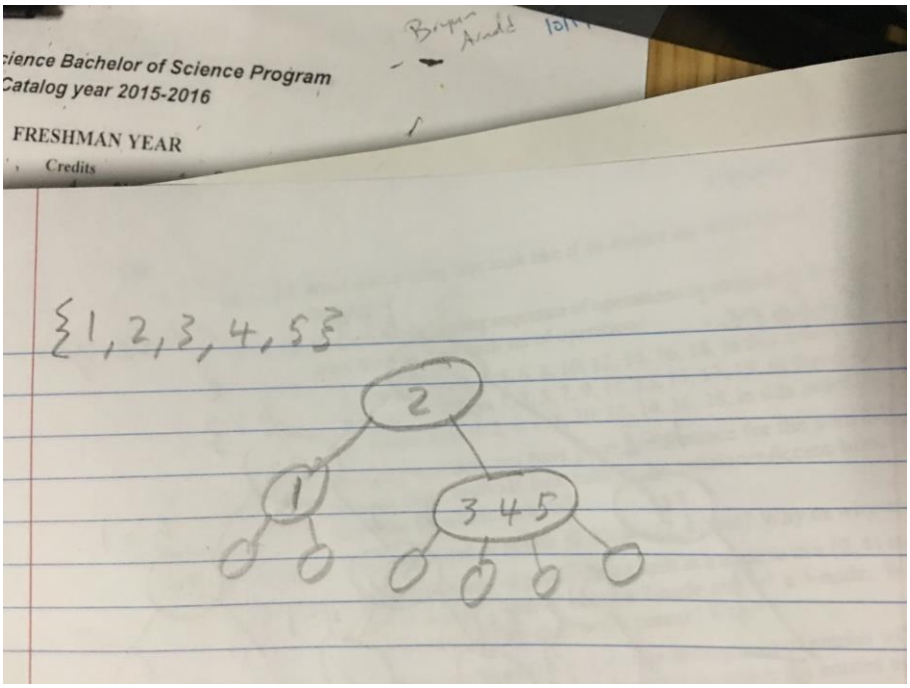
The search tree in figure 11.22(a) is not a (2,4) tree. It satisfies all the other qualities of a multi-search tree, but doesn't satisfy the depth property of a (2,4) tree. The depth property states that all leaves must have the same depth, or that each leaf must have the same number of ancestors. In the figure, the leaves that contain 11 and thirteen and the other leave that contains 17, both have a depth of 3. All other leaves have a depth of 2. So, this violates the depth property, making it not a (2,4) tree.

[R-11.19]

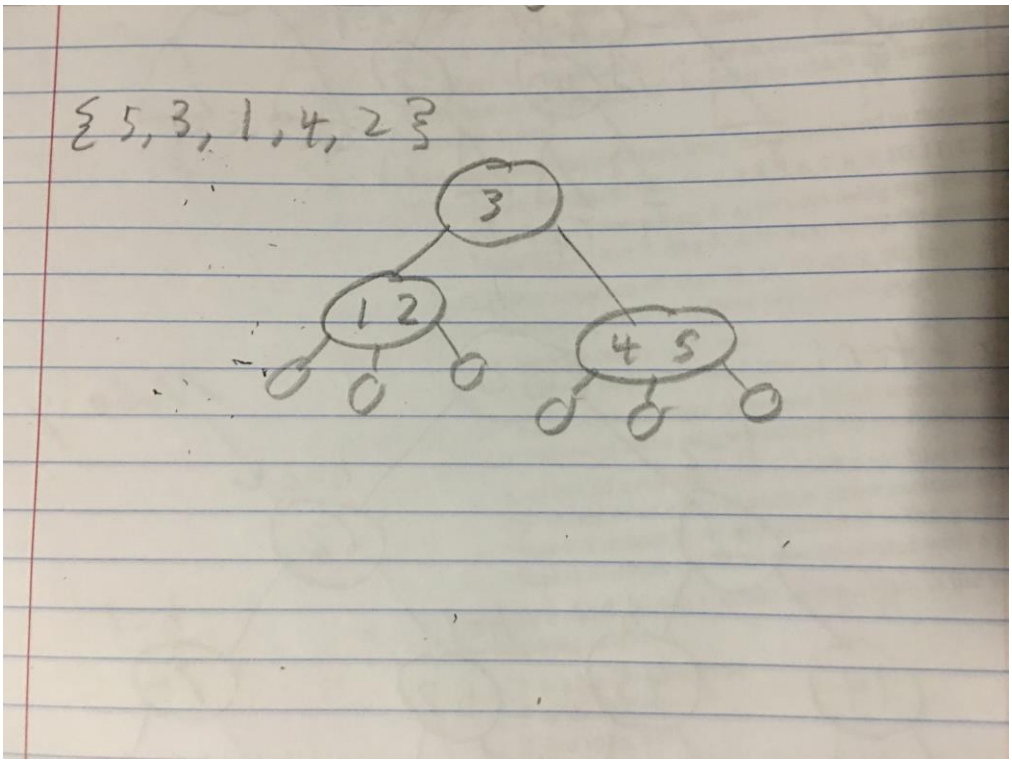
(All blank nodes are null)

An instance where Dr. Amongus is incorrect is the insertion orders of the following keys:

{1, 2, 3, 4, 5} and {5, 3, 1, 4, 2}. These will create two different (2,4) tree structures, as follows:



Note, there is one child with 2 nodes, and one child with 4 nodes.



Now note that the two children have 3 nodes. These are two different tree structures, but have the same key entries, just in different orders. So, Dr. Amongus is yet again incorrect.

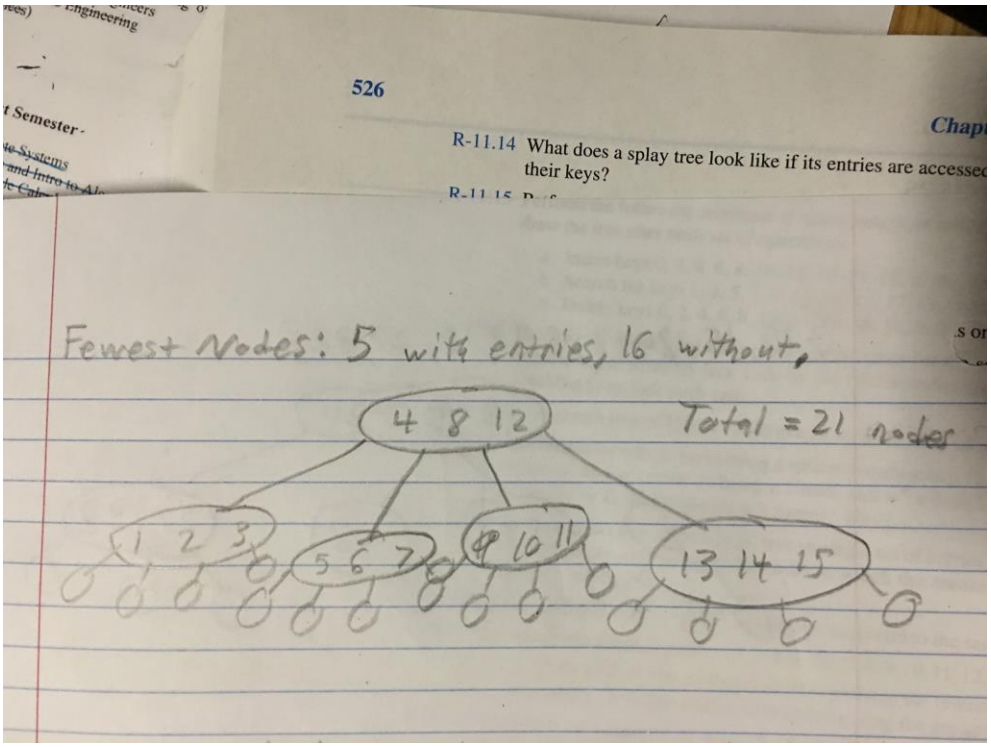
[R-11.21]

(All blank nodes are null)

526

R-11.14 What does a splay tree look like if its entries are accessed their keys?

a.



b.