

HW 3

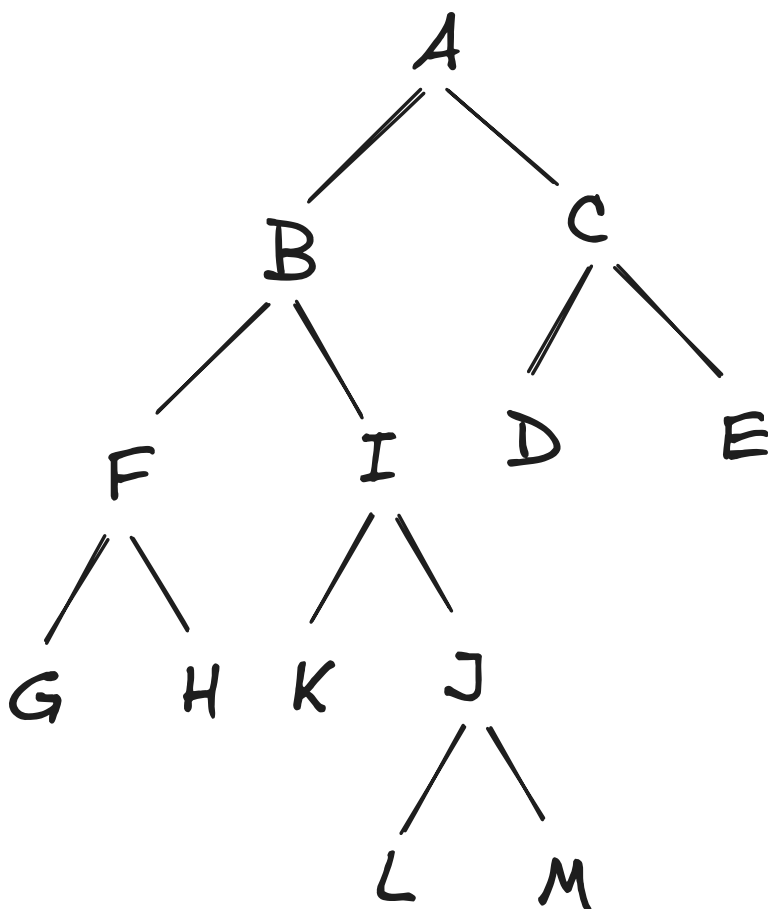
Charles Liu

Advanced Data Structures - Dr. Subu Kandaswamy

March 26th 2025

Tree Features

1. For the tree shown above, answer the following:



a) Number of nodes in the tree (include root, leaves, and internal nodes).

13 nodes

b) Number of leaves in the tree.

7 leaves

c) Height of the tree = ?

4

d) Depth of node J = ?

3

e) Height of node J = ?

1

f) Give the post-order and pre-order traversals

Pre-order: *A, B, F, G, H, I, K, J, L, M, C, D, E*

Post-order: *G, H, F, K, L, M, J, I, B, D, E, C, A*

g) Give the level-order and in-order traversals of the tree.

Level-order: *A, B, C, F, I, D, E, G, H, K, J, L, M*

In-order: *G, F, H, B, I, K, J, L, M, A, D, C, E*

2. (6 points) Reconstructing the tree from its traversals:

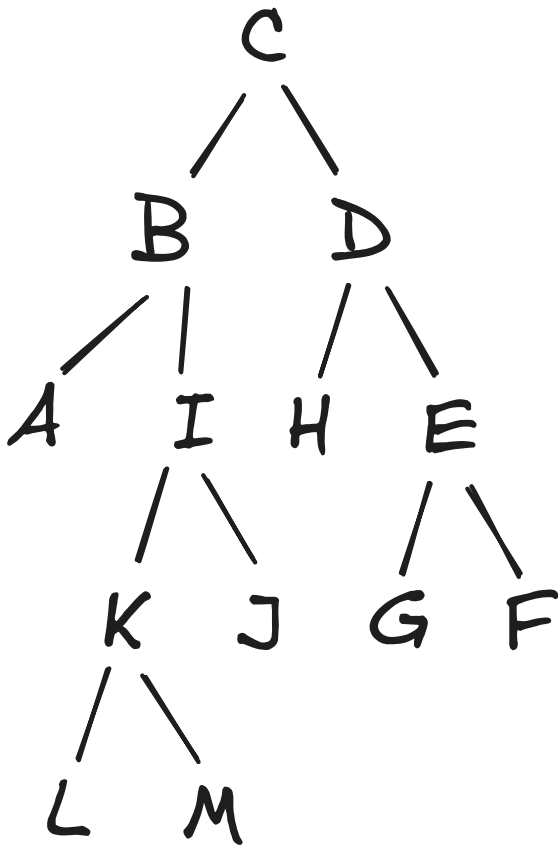
It is possible to reconstruct (meaning, draw) the binary tree given only the tree's pre-order and in-order traversals. To illustrate this, draw the binary tree by reconstructing it from the following traversals:

• Pre-order : *C, B, A, I, K, L, M, J, D, H, E, G, F*

• In-order: *A, B, L, K, M, I, J, C, H, D, G, E, F*

Note: You must draw a single binary tree that works for both traversals.

Hint: it would help if you draw smaller toy examples of a tree, write down their in-order and pre-order traversals, understand what node goes where in those traversals, before attempting this question.



3. (6 points) Basic tree property:

How many edges are there in any tree with n nodes? Provide your answer in terms of n along with an explanation (proof, even better) of why that should be true. Answers with no explanations will not be given points.

Note that the tree can be any tree with a root (i.e., binary tree, 3-way tree, etc.). There is no restriction it is a binary tree. Secondly, recall that “nodes” include the root, any internal nodes, and any leaves.

Hint: It might help if you draw a few examples for yourself of arbitrary trees to arrive at an answer.

An arbitrary tree of n nodes must have $n - 1$ edges.

For any given tree, we can break it up into the root (1 node) and all other nodes ($n - 1$ nodes). Every non-root node has to be a child of a parent as it is a descendent of the root, so there must be one edge connecting from it to its parent.

Since the root has no edge connecting upwards to a parent and all edges have unique nodes connecting below, we can count all the edges as $(n - 1) \cdot 1 = n - 1$.

Thus, an arbitrary tree with n nodes has $n - 1$ edges.

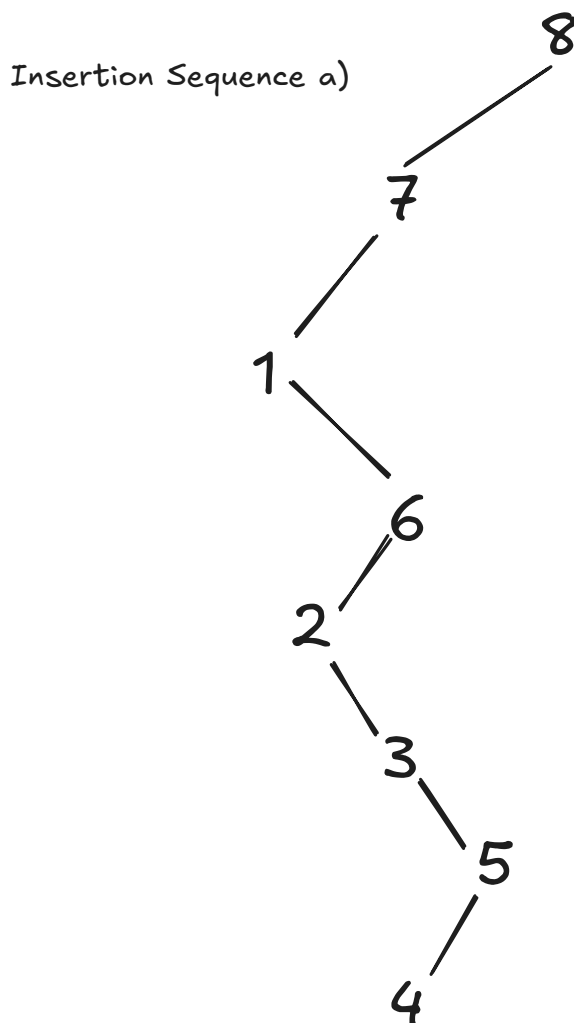
4. (7 points)

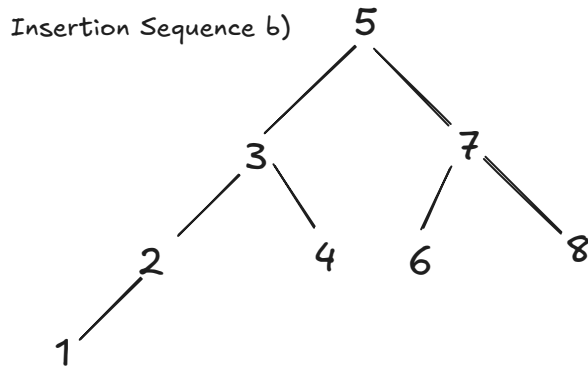
a) Draw the BSTs that result from the following two different insertion sequences of the same set of elements:

a) Insertion sequence: 8 7 1 6 2 3 5 4

b) Insertion sequence: 5 7 3 2 8 6 4 1

For both cases, start with an empty tree. Just showing the final tree in your answer is sufficient. Alternatively, if you want to show the BST after every insertion step by step, that is fine too. But make sure to highlight the final tree.





b) Briefly state what is so markedly different between the shapes of the two resulting BSTs.

The first linked list is extremely unbalanced, having all nodes in one single line down. This is basically a linked list, and is characteristic of a degenerate BST.

In contrast, the second linked list is much more balanced, producing a triangle shape. The left subtree and right subtree have similar amounts of nodes

5. (12 points)

a) Starting with an empty tree T_0 , show the set of AVL trees $T_0 \Rightarrow T_1 \Rightarrow T_2 \Rightarrow \dots$ resulting from performing the following sequence of operations (in that order):

Insert(7), Insert(5), Insert(2), Insert(4), Insert(3), Insert(1)

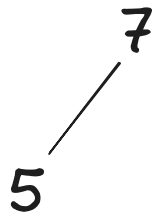
If at any step you need to rebalance the tree using rotation, then clearly identify: i) the node that has the imbalance (i.e., violation), and ii) the corresponding rotation “case” that applies there (i.e., case 1 or 2 or 3 or 4).

T_0

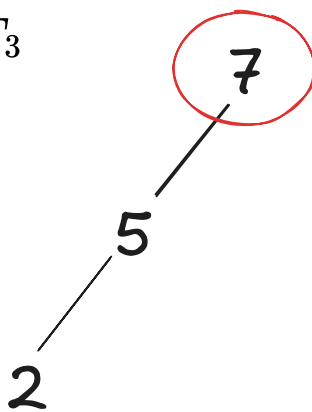
T_1

7

T_2

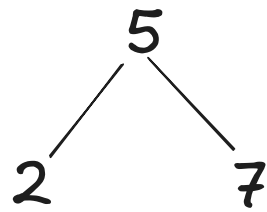


T_3

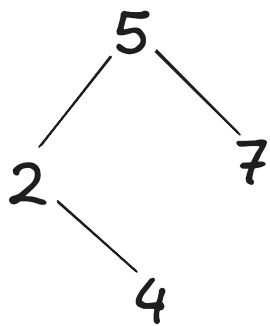


Imbalance,
Left heavy
BF = -2

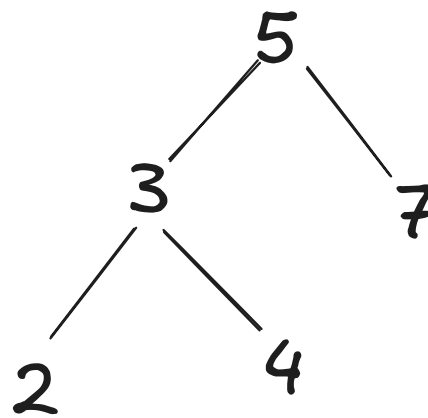
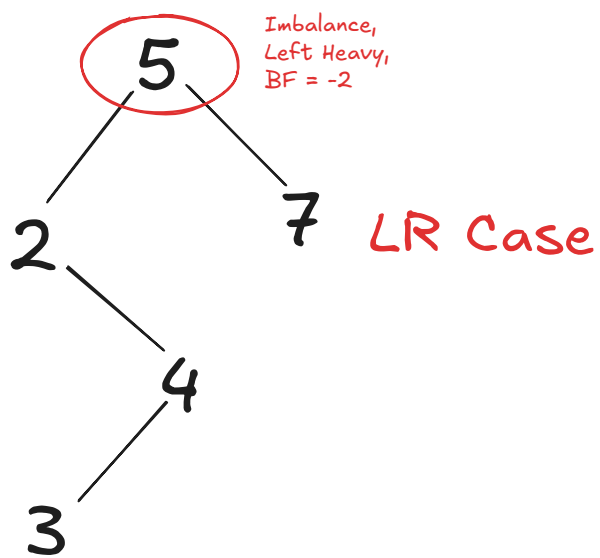
LL Case



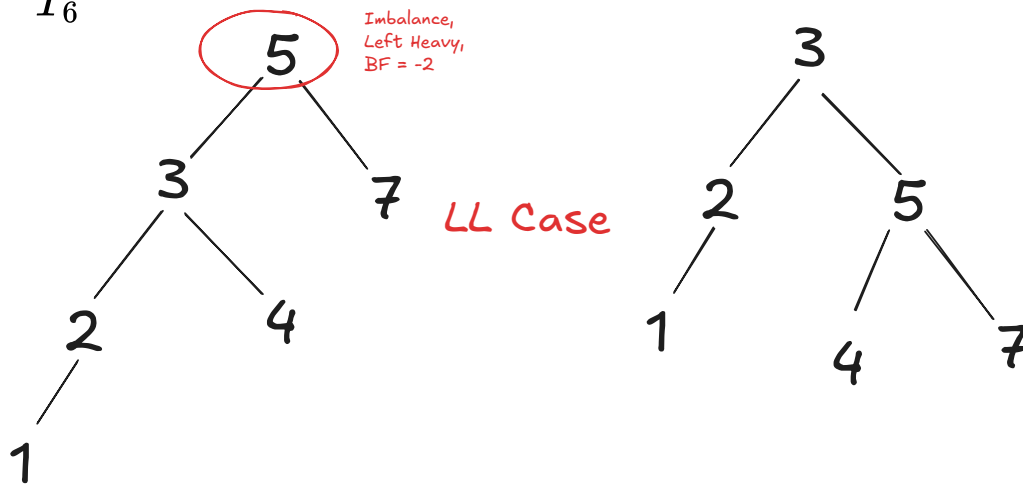
T_4



T_5



T_6



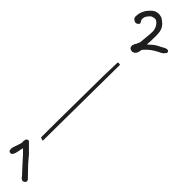
b) Answer the same question as in part (a) but with this insertion sequence instead:

Insert(2), Insert(1), Insert(4), Insert(5), Insert(9), Insert(3), Insert(6), Insert(7)

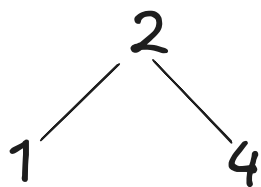
T_0

2

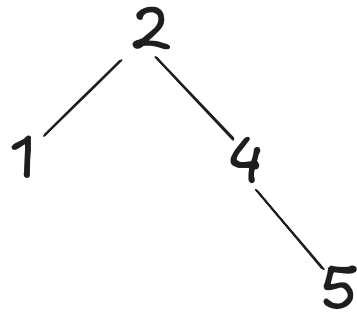
T_1



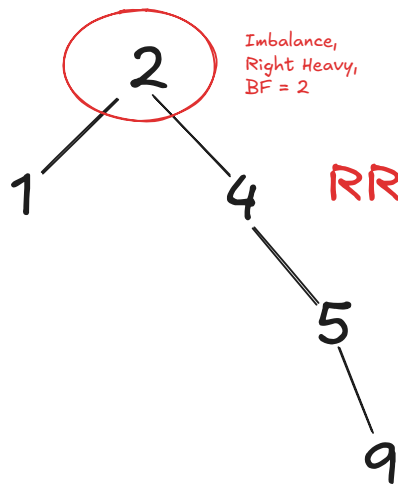
T_2



T_3

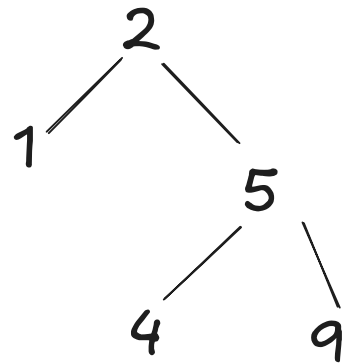


T_4

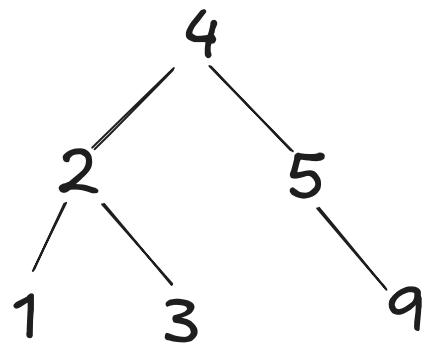
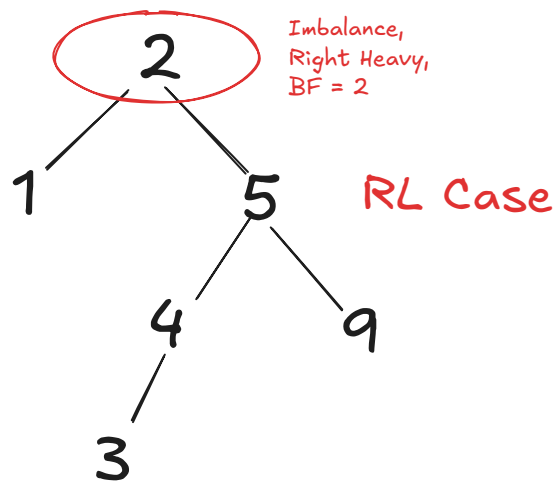


Imbalance,
Right Heavy,
BF = 2

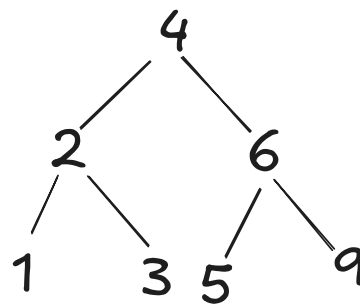
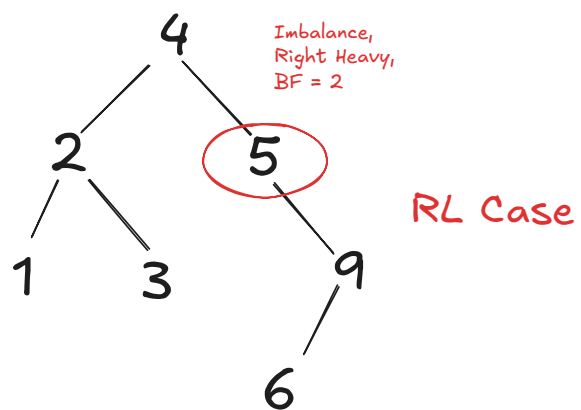
RR Case



T_5



T_6



T_7

