

Restructuring in AVL trees

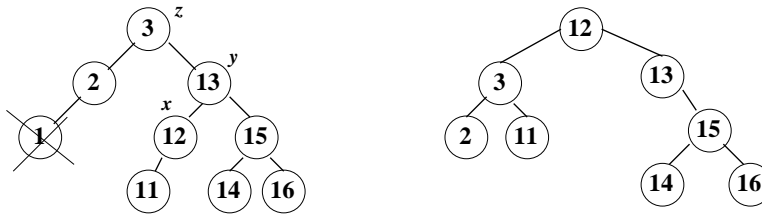
In the lecture, we considered the three nodes z, y, x which were defined as follows. Let z be the first node encountered with an imbalance. Let y be the child of z with the higher height. Let x be the child of y with the higher height. Then perform the restructuring using these three nodes.

The above description is correct in general but there are some subtleties that needs to be pointed out. In particular, Jason Overmier noticed that there was a problem in one of my examples which made me check the details more carefully.

During insertion. It's possible that both children of z have the same height. If so, choose y so that y is an ancestor of the node just inserted. Similarly, the children of y can have the same height. If so, choose x so that it's an ancestor of the node just inserted.

During deletion. In this situation, y is always the child of z that is *not* the ancestor of the node just deleted. Now, it's again possible that the children of y have the same height. If so, chose x so that it's on the "same side" as y ; that is, if y is a right child of z , then x should be the right child of y , etc.

Here's the error that Jason spotted:



The tree on the left is a valid AVL tree. We then deleted the node whose key is 1. An imbalance occurs at the node z . Choosing y is straightforward, but we have two available choices for x since the left and right subtrees of y have equal heights. Interestingly, when we chose the node whose key is 12, the restructuring creates a problem. Do you see it? It's at the node whose key is 13. We should have chosen the node whose key is 15 for x , and the restructuring would have been fine. So please bear in mind that the "tie-breaking" decision for choosing y and x are actually important.