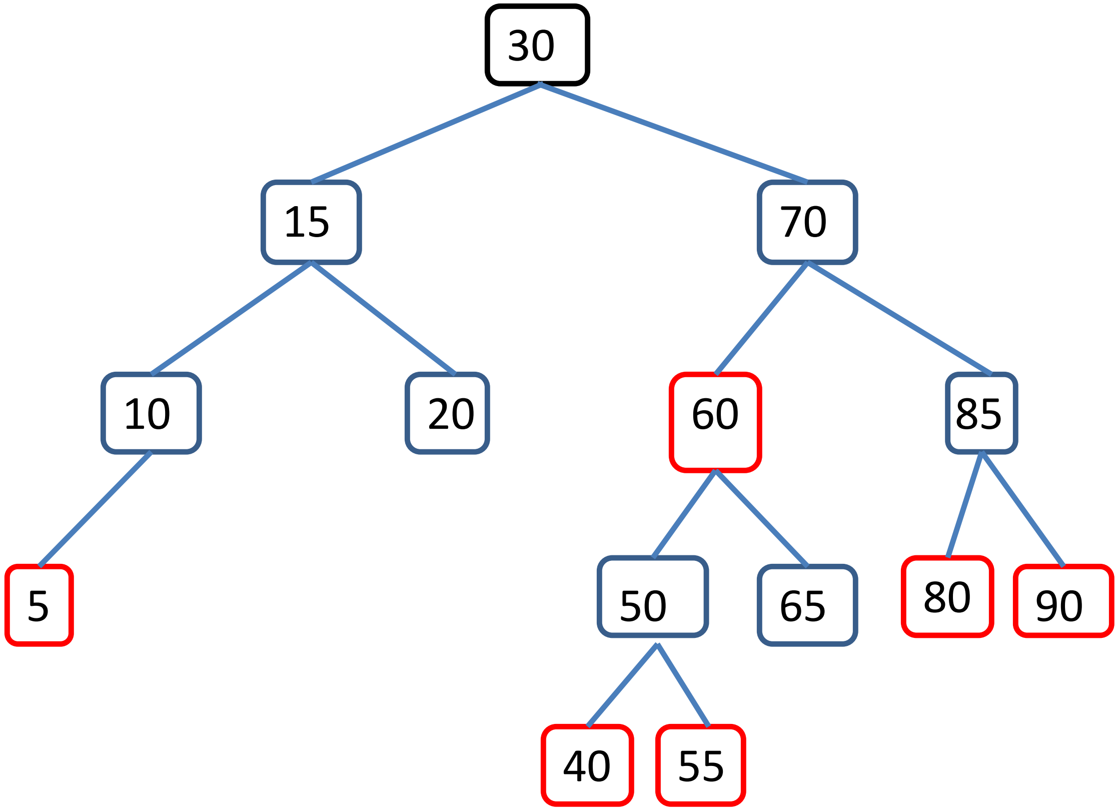
Other Balanced Binary Search Trees

Instead of a height condition Red-Black trees give each node a color, either red or black. The colors must satisfy 3 properties:

a) The root must be black.
b) If a node is red its children must be black.
c) Every path from a node to a null link must contain the same number of black nodes.


30
15
70
10
20
60
85
5
50
65
80
90
40
55

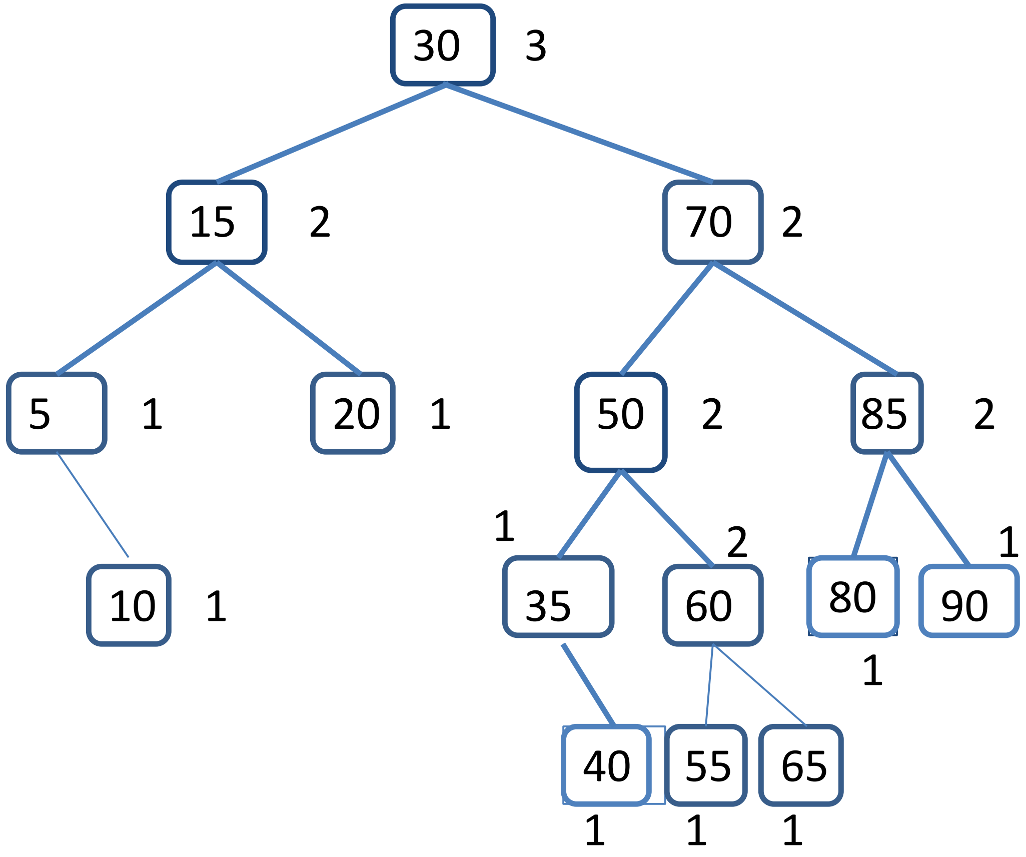
You can show that the height of a red-black tree with N nodes is at most 2*log(N+ 1), so these again get logarithmic performance.

Insertions and deletions with red-black trees are similar to those with AVL trees, with rotations used to rebalance the tree and preserve the red-black conditions. The difference is that both insertions and deletions can be performed with a single top-down pass, so these algorithms can be done with loops rather than recursions.   The difficulty is that both algorithms and their rotations are a bit more complex than the corresponding algorithms for AVL trees.

AA trees, named after their inventor Arne Anderson, are a variation on red-black trees. One way to explain them is that they are red-black trees in which left-children cannot be red.   The usual discussion is in terms of *level*. Here are the rules:

- Every leaf node has level 1
- The level of a left child is one less than the level of its parent.
- The level of every right child is equal to or one less than the level of its parent.
- The level of every right grandchild is strictly less than the level of its grandparent.
- Every node of level greater than 1 has two children.

Here is an AA tree with the level of each node indicated.



AA trees have simpler implementations than red-black trees.