Red-Black Tree | Set 1 (Introduction)

Red-Black Tree is a self-balancing Binary Search Tree (BST) where every node follows following rules.  
[](https://www.geeksforgeeks.org/wp-content/uploads/RedBlackTree.png)  
**1)** Every node has a color either red or black.

**2)** Root of tree is always black.

**3)** There are no two adjacent red nodes (A red node cannot have a red parent or red child).

**4)** Every path from root to a NULL node has same number of black nodes.

**Why Red-Black Trees?**  
Most of the BST operations (e.g., search, max, min, insert, delete.. etc) take O(h) time where h is the height of the BST. The cost of these operations may become O(n) for a skewed Binary tree. If we make sure that height of the tree remains O(Logn) after every insertion and deletion, then we can guarantee an upper bound of O(Logn) for all these operations. The height of a Red-Black tree is always O(Logn) where n is the number of nodes in the tree.

Comparison with AVL Tree  
The AVL trees are more balanced compared to Red-Black Trees, but they may cause more rotations during insertion and deletion. So if your application involves many frequent insertions and deletions, then Red Black trees should be preferred. And if the insertions and deletions are less frequent and search is a more frequent operation, then AVL tree should be preferred over Red-Black Tree.