**RED-BLACK TREES (a self-balancing tree)**

Red-black tree is used to implement TreeSet and TreeMap in Java.

Chart

Description automatically generated

Length of the path from root to a leaf could be as much as twice the number of black nodes (in the path) since there could be a red node between each 2 black nodes.

Path also could be as minimum as number of black nodes (in the path).

Height is determined by counting only black nodes

A red-black tree is always balanced bc the root node’s left and right subtrees must be the same height

The algorithm follows the same recursive search process used for all bst to reach the insertion point

When a leaf is found, the new item is inserted and initially given the color red

If the parent is black, we are done; otherwise there is some rearranging to do

We introduce 3 situations (“cases”) that may occur when a node is inserted; more than one can occur after an insertion

**CASE 1**

Chart

Description automatically generated A picture containing diagram

Description automatically generated

At right figure, if this tree is a subtree of a red node, then there is a problem. After rearrangement, we moved problem at leaf to the root of the subtree. We can continue to carry this problem up until the root of all tree and solve the problem.  
Number of black nodes on the path root to a leaf is the same.

A picture containing diagram

Description automatically generated A picture containing diagram

Description automatically generated

Now we have balanced tree.

**CASE 2**

A picture containing chart

Description automatically generated A picture containing chart

Description automatically generated

Diagram

Description automatically generated with low confidence Diagram

Description automatically generated

**CASE 3**

A picture containing text

Description automatically generated A picture containing diagram

Description automatically generated

Diagram

Description automatically generated with medium confidence A picture containing diagram

Description automatically generated

Text

Description automatically generated with medium confidence A picture containing chart

Description automatically generated

A picture containing chart

Description automatically generated Diagram

Description automatically generated with low confidence

Chart

Description automatically generated with medium confidenceEXAMPLE 🡪 insert 4

This is case 1 since parent and sibling are both red.

CASE 1 SAYS : If a parent is red, and its sibling is also red, they can both be changed to black, and the grandparent to red.

Chart

Description automatically generated

Chart

Description automatically generated with low confidence

A picture containing chart

Description automatically generated

A picture containing graphical user interface

Description automatically generated

A picture containing graphical user interface

Description automatically generated

A picture containing diagram

Description automatically generated

EXAMPLE: Build a Red-Black tree for the words in “The quick brown fox jumps over the lazy dog”

Diagram

Description automatically generatedThere is no problem while adding “The” and “quick”

Adding “brown” creates case 3.

Rotate “quick” right, change colors and rotate “The” left.

Diagram

Description automatically generatedRotated “quick” right.

Graphical user interface, diagram

Description automatically generated

Diagram

Description automatically generatedChanged colors.

Rotated “The” left.

Diagram

Description automatically generatedAdding “fox” creates case 1.

“fox”’s parent and its parent’s sibling are both red. Change colors.

Diagram

Description automatically generatedChange colors.

Diagram

Description automatically generated

Adding “jumps” creates case 3.

Rotate “fox” left (so that red child is on same side of its parent as its parent is to the grandparent), change colors, and rotate “quick” right.

Diagram

Description automatically generated

Rotated “fox” left.

Diagram

Description automatically generatedChanged colors.

Diagram

Description automatically generatedRotate “quick” right.

Diagram

Description automatically generatedCase 1.

Diagram

Description automatically generated

Diagram

Description automatically generatedAdding “the” doesn’t violate anything.

Diagram, radar chart

Description automatically generated

Adding “lazy” creates case 1.

Chart, radar chart

Description automatically generatedNow we have case 2.

We moved the problem at “jumps”-“quick”

Rotation over “brown” is needed.

Diagram

Description automatically generatedRotated “brown” left.

Adding “dog” doesn’t violate anything Diagram

Description automatically generated.

Implementation of a Red-Black Tree Class

Diagram

Description automatically generated

Removal from a Red-Black Tree

Remove a node only if it is a leaf or has only one child

Otherwise, the node containing the inorder predecessor of the value being removed is removed

If the node removed is red, nothing further is done

If the node removed is black, and has a red child, then the red child takes its place and is colored black

If a black leaf is removed, the black height becomes unbalanced

Performance of a Red-Black Tree

The upper limit in the height for a Red-Black tree is 2logn + 2 which is still O(logn)

As with AVL trees, the average performance is significantly better than the worst-case performance

Empirical studies show that the average cost of searching a Red-Black tree built from random values is 1.002logn

Red-Black trees and AVL trees both give performance close to that of a complete binary tree

TreeMap and TreeSet Classes

The Java API has a TreeMap class that implements a Red-Black tree

It implements SortedMap so some of the methods it defines are:

* get
* put
* remove
* containsKey

All are O(logn) operations

TreeSet implements SortedSet and is an adapter of the TreeMap class