If this is a subtree and has a red parent, we go up and balance again.

(a) balance

(b) Insert 0. Get red-red violation

(c) Start with this tree

In this example, we insert 30 into the tree in (a). Bold nodes are black.

Insertion and Balancing Example

Red-Black Tree Insertion

Red-Black Tree is Balanced

Red-Black Tree

Time:

- By balancing, we return back to the root. This only takes logarithmic

But then, the parent of the new node may be red as well. We will fix this

- So we make it red.

- On paths, and this is expensive to report.

- If it is black, it will cause a difference in the number of black nodes

What should be the color of the new node?

First, insert the key as usual (keeping the tree as a binary search tree).

Properties:

How to insert a key in logarithmic time and still preserve the red-black tree

All paths have the same number of black nodes

- A red node has no red child

With these restrictions:

- data RB = a (insert a) (remove a)

A red-black tree is a binary search tree with every node colored red or
where node - x t1 t2 = true t.
insert d = node p x t1 t2

Solution: Simple. Just mark the root as black. Always.
root, we have no further black parent to help us balance it.
We need a finishing touch: when the red-red violation propogates to the

otherwise = n
if (t2 = balance c r t1 (true t2) t2) t2
drk = balance c r (true t) t2
t2 = drk = balance c r (true t) t2
t2 = drk = balance c r (true t) t2

Note: This function works on the contents of the black node instead of
See how it is called in insert below)
the black node itself, because it most probably gets thrown away anyway.

Balance Code

The relations are coded as follows:

Balance Details

There are four forms of red-red violation:

Insertion Code

Each is noted to obtain the following: