Data Structures

Insertion In A Red-Black Tree

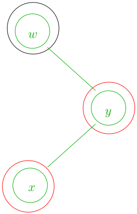
Design and Analysis of Algorithms I

High-Level Plan

Idea for Insert/Delete: Proceed as in a normal binary search tree, then recolor and/or perform rotations until invariants are restored.

Insert(x):

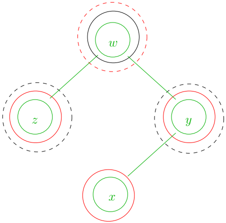
1. Insert x as usual (makes x a leaf).
2. Try coloring x red.
3. If x's parent y is black, done.
4. Else y is red $\Rightarrow$ y has a black parent w.



Insertion

Case 1

Case 1: The other child z of x's grandparent w is also red.
$\Rightarrow$ Recolor y, z black and w red. [key point: does not break invariant (4)]
$\Rightarrow$ Either restores invariant (3) or propagates the double red upward.
$\Rightarrow$ Can only happen $O(\log n)$ times. [If you reach the root, recolor it black $\Rightarrow$ Preserves invariant (4)].



Case 2

Case 2: Let x, y be the current double-red, x the deeper node. Let $w = x$'s grandparent. Suppose w's other child is NULL or is a black node z.

Exercise/case analysis (details omitted): Can eliminate double-red [$\Rightarrow$ All invariants satisfied] in $O(1)$ time via 2-3 rotations+ recolorings.