



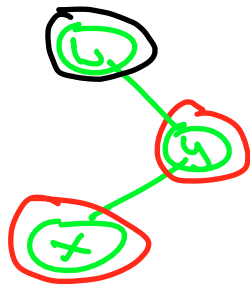
Design and Analysis
of Algorithms I

Data Structures

Insertion In A Red-Black Tree

High-Level Plan

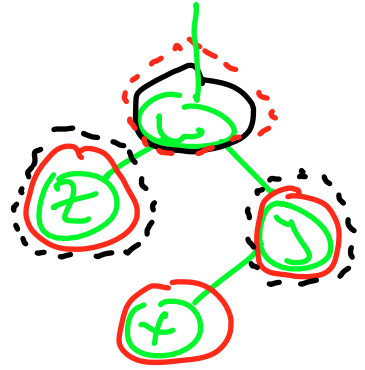
Idea for Insert/Delete : proceed as in a normal binary search tree, then recolor and/or perform rotations until invariants are restored.



- 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

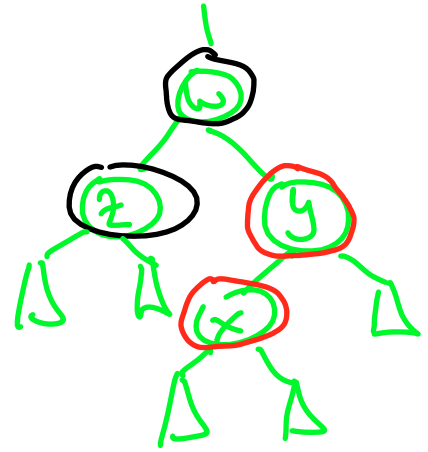
[key point: does not break invariant (4)]

\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 grand parent. Suppose w 's other child ($\neq y$) 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.