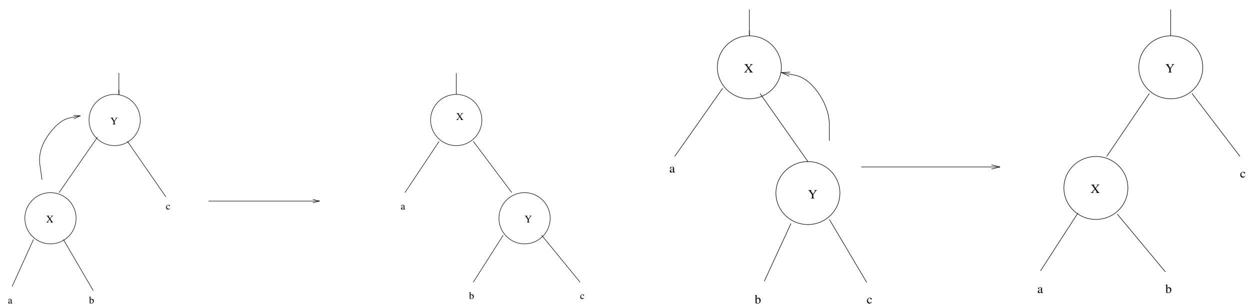
Rotations

The basic restructuring step for binary search trees are left and right rotation:



1. Rotation is a local operation changing $O(1)$ pointers.

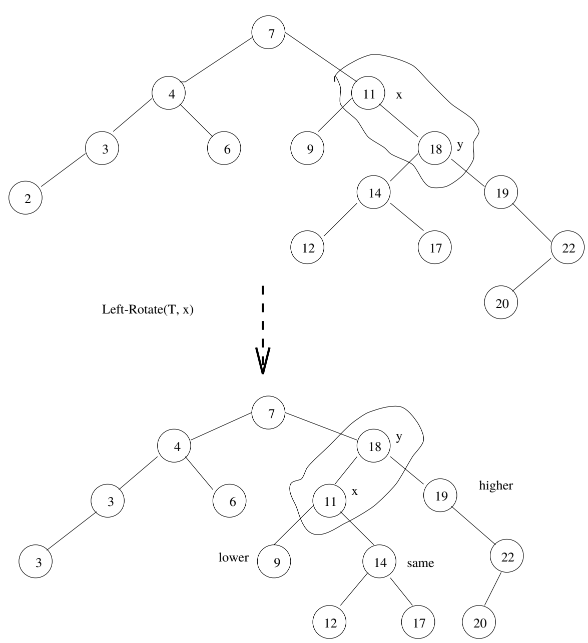
2. An in-order search tree before a rotation *stays* an in-order search tree.

3. In a rotation, one subtree gets one level closer to the root and one subtree one level further from the root.

LEFT-ROTATE(T,x)

$y \leftarrow right[x]$ (* Set y*)
$right[x] \leftarrow left[y]$ (* Turn y's left into x's right*)
if $left[y] \neq$ NIL
    then $p[left[y]] \leftarrow x$
$p[y] \leftarrow p[x]$ (* Link x's parent to y *)
if $p[x] =$ NIL
    then $root[T] \leftarrow y$
    else if $x = left[p[x]]$
        then $left[p[x]] \leftarrow y$
        else $right[p[x]] \leftarrow y$
$left[y] \leftarrow x$
$p[x] \leftarrow y$

Note the in-order property is preserved.

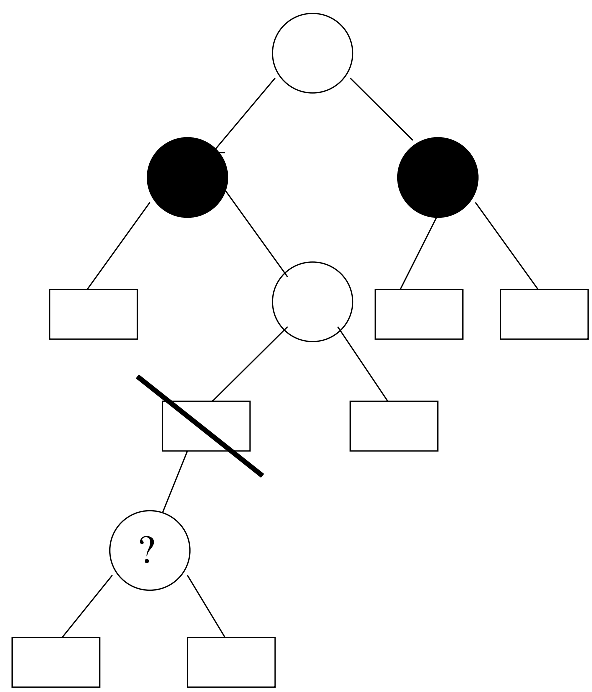


Red-Black Insertion

Since red-black trees have $\Theta(\lg n)$ height, if we can preserve all properties of such trees under insertion/deletion, we have a balanced tree!

Suppose we just did a regular insertion. Under what conditions does it stay a red-black tree?

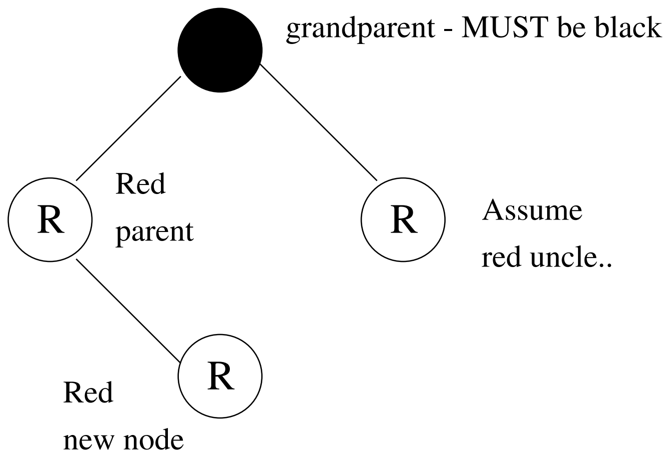
Since every insertion take places at a leaf, we will change a black NIL pointer to a node with two black NIL pointers.



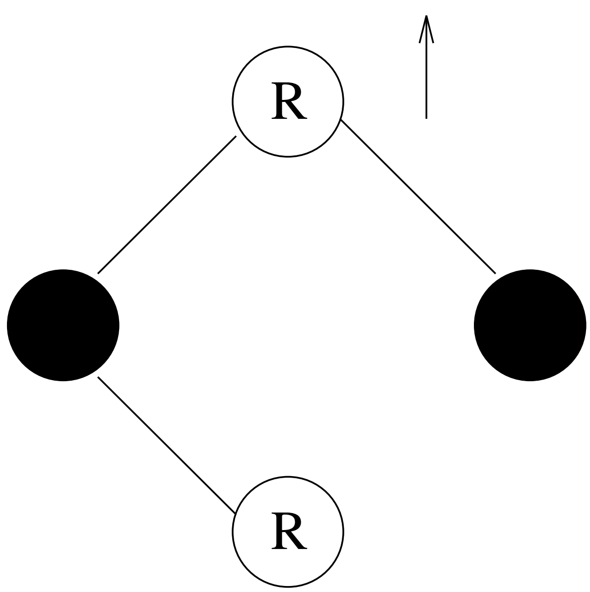
To preserve the black height of the tree, the new node must be *red*. If its *new* parent is black, we can stop, otherwise we must restructure!

How can we fix two reds in a row?

It depends upon our uncle's color:



If our uncle is red, reversing our relatives' color either solves the problem or pushes it higher!



Note that after the recoloring:

1. The black height is unchanged.

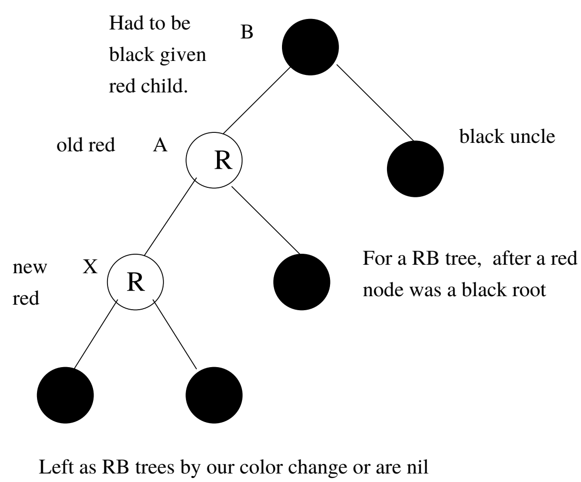
2. The shape of the tree is unchanged.

3. We are done if our great-grandparent is black.

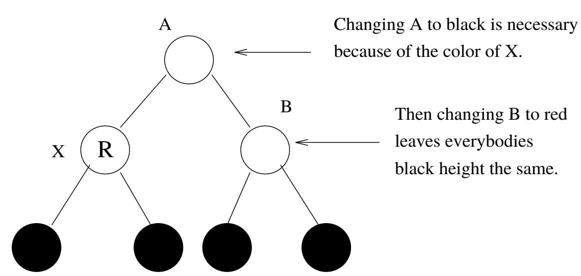
If we get all the way to the root, recall we can always color a red-black tree's root black. We always will, so initially it *was* black, and so this process terminates.

The Case of the Black Uncle

If our uncle was black, observe that all the nodes around us have to be black:



Solution - rotate right about B:



Since the root of the subtree is now black with the same black-height as before, we have restored the colors and can stop!