Red Black trees

A RB-tree is a binary search tree (BST), where each node x has following attrs:

- x.key — what you compare
- x.left, x.right } children
- x.p — parent
- x.color — red or black
- possibly satellite data

T is red-black tree

- T.root ← pointer to root node
- T.nil ← dummy leaf node
- T.nil.color = black

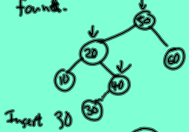
Initially, T.root = T.nil

Any BST: no duplicate keys!

Review of BST (pointer to root of tree)

```
node find(BST T, key x)   // Java style
{
  if (T==nil)
    return nil;
  else if(x==T.key)
    return T;
  else if(x < T.key)
    return find(T.left, x);
  else // x > T.key
    return find(T.right, x);
}
```

Insert is just like find, but insert as new leaf where the node would otherwise have been found.



Insert 30

Keep tree balanced.

Structural alterations

shape

2 types:

- left rotation (at a node)
- right rotation (at a node)

In any red-black tree:

0. Every node except a leaf has 2 children
1. The root is black
2. All leaves are black (T.nil)
3. If a node is red, then its parent is black.
   (equiv: if red, then both children are black)
4. For every node x, the same number of black nodes along any path from x to a leaf is the same, regardless of the path.
   [Black height of x (bh(x)): # of black nodes along any path from x to any leaf excluding x (but including the leaf)]

Show that in any RB-tree with n internal nodes, depth is O(lg n).

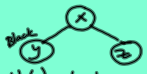
Consider a node with x with bh(x) = h

h = 1 : tree rooted at x has size 1 (not including leaves if x is black (same if x is red)

Oct 6-12:32 PM

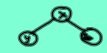
Now suppose $h > 1$.

Case 1: x is red



$bh(y) = h-1$

Case 2: x is black

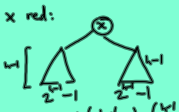


By induction:

if x is red, then size(tree at x) $\geq 2^h - 1$

if x is black, then size(tree at x) $\geq 2^h - 1$

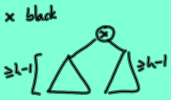
x red:



size at $x \geq (2^{h-1} - 1) + (2^{h-1} - 1) + 1$

$= 2^h - 1$

x black



size at $x \geq (2^{h-1} - 1) + (2^{h-1} - 1) + 1$

$= 2^h - 1$

$\therefore$ size at x is $\geq 2^{bh(x)} - 1$

regardless of x's color.

height(root) $\leq 2\, bh(root)$

$\leq H$

S = size at root = size of whole tree

$\geq 2^{bh(root)} - 1 \geq 2^{H/2} - 1$

$S \geq 2^{H/2} - 1$

$S + 1 \geq 2^{H/2}$

$\lg(S+1) \geq H/2$

$H \leq 2 \lg(S+1)$

$= O(\lg n)$

Insertion into a RB-tree

2 steps:

1. normal BST insert
2. fix-up

To insert x:

x.color := red

insert it into the RB-tree as you would with a normal BST

Easy case:

y.color = black

nothing further to do

check:

Suppose y is red.

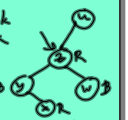
assume that y is the left child of z in all that follows. Otherwise, we do the same things reversing left & right.

If w is red,

y.color := black

w.color := black

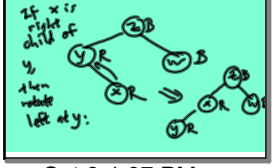
z.color := red



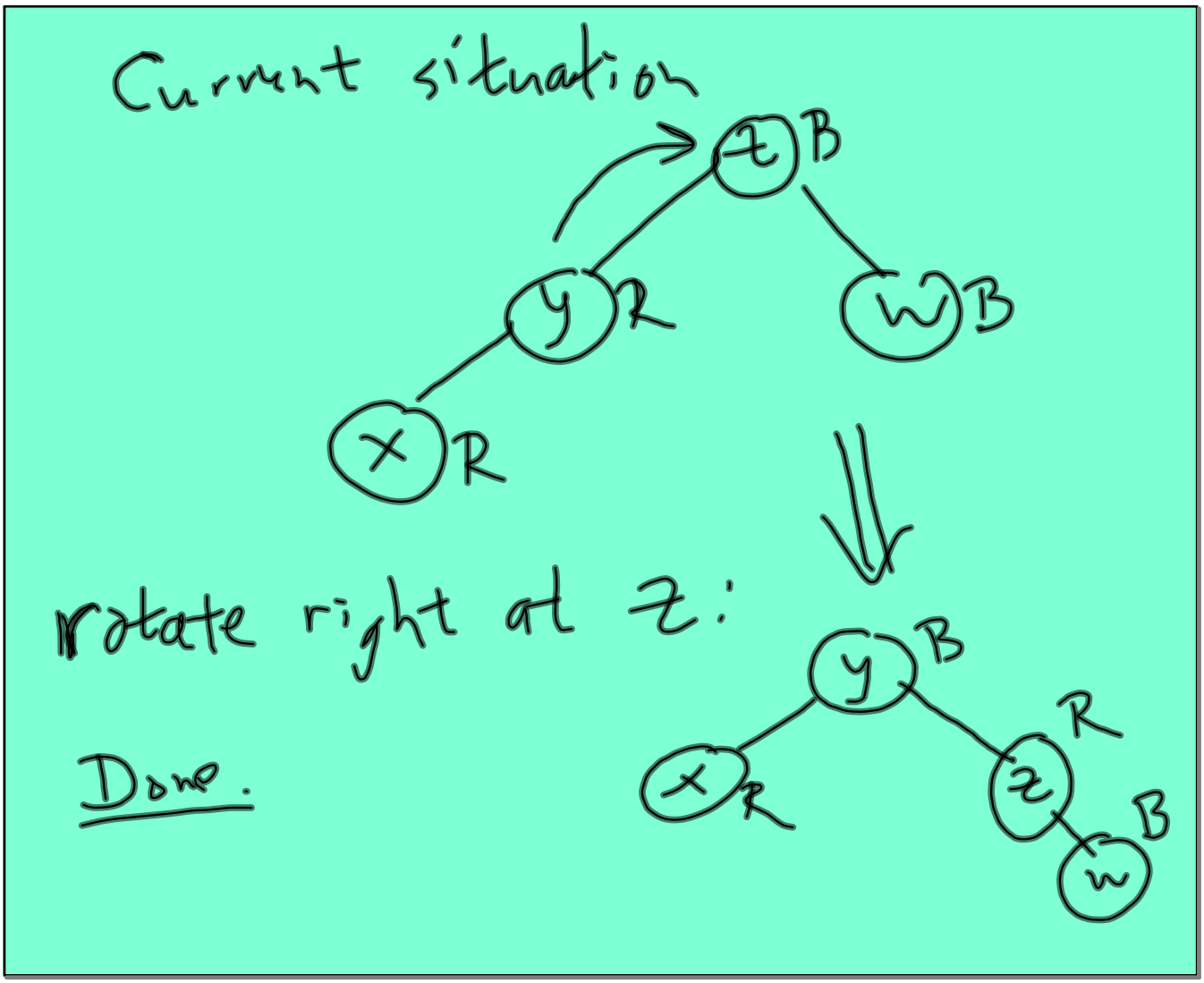
Repeat whole fix-up process at z. Process will terminate eventually, when parent of "problem" node is black (this could be the root).

Suppose w is black

If x is right child of y, then rotate left at y:



Oct 6-1:07 PM


Current situation
z B
y R
w B
x R
rotate right at z:
Done.
y B
x R
z R
w B