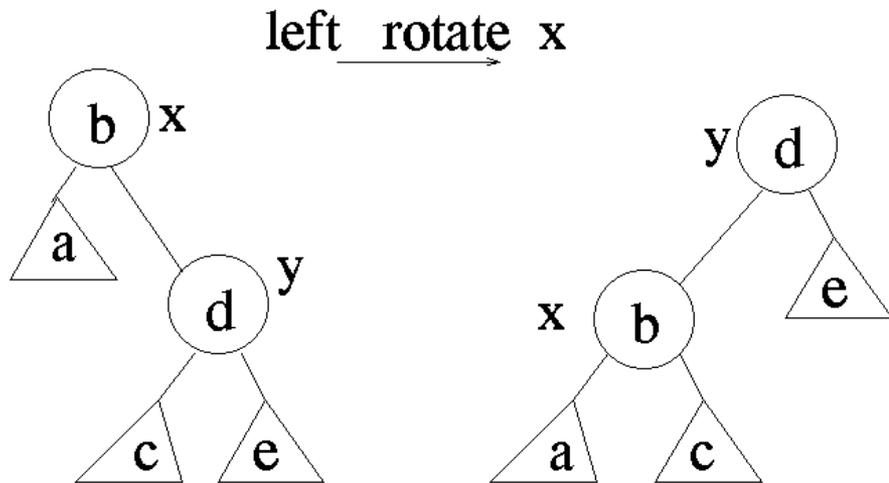


CompSci 100e

Program Design and Analysis II



April 21, 2011

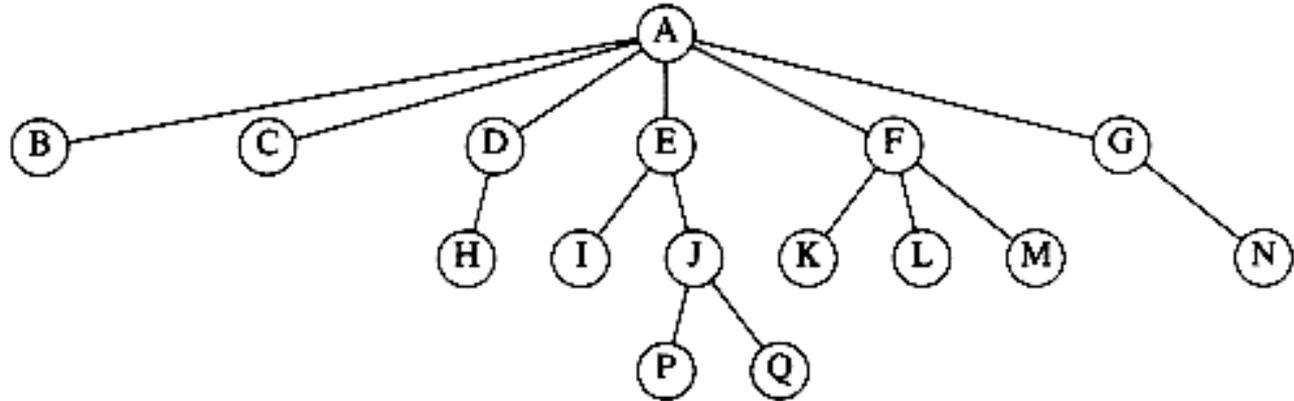
Prof. Rodger

Announcements

- Lab tomorrow
 - Focus on BoggleScore APT
- What is due:
 - Huffman (extended til Monday night, Apr 25)
 - APTS due Tue, April 26
 - Extra credit assignments due Wed, April 27
- Graphs and Red-Black trees today
 - Classwork 16
 - Internet APT – work through parts in class

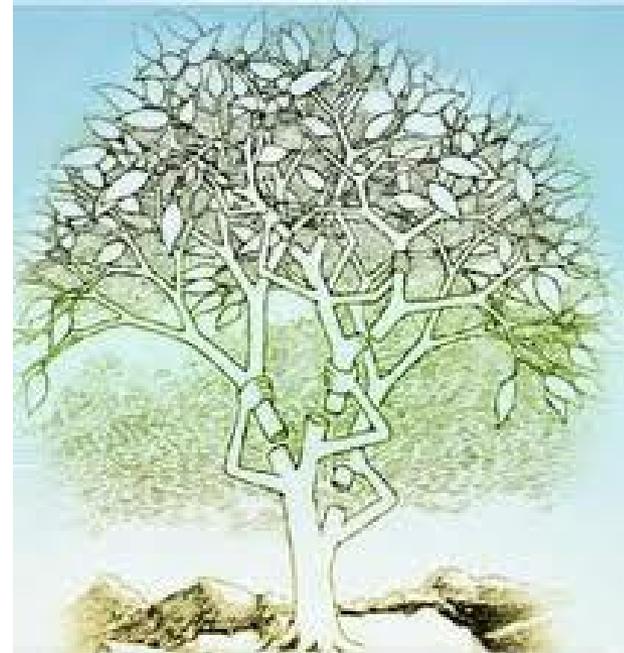
More on trees

- General tree



Balanced Trees

- Splay trees
- AVL trees
- Red-black trees
- B-trees



Balanced trees, we'll look at red-black

- Both kinds have worst-case $O(\log n)$ time for tree operations
- AVL (Adel'son-Velskii and Landis), 1962
 - Nodes are “height-balanced”, subtree heights differ by 1
 - Rebalancing requires per-node bookkeeping of height
 - <http://people.ksp.sk/~kuko/bak/>
- Red-black tree uses same rotations, but can rebalance in one pass, contrast to AVL tree
 - In AVL case, insert, calculate balance factors, rebalance
 - In Red-black tree can rebalance on the way down, code is more complex, but doable
 - Standard `java.util.TreeMap/TreeSet` use red-black

Red-Black Tree

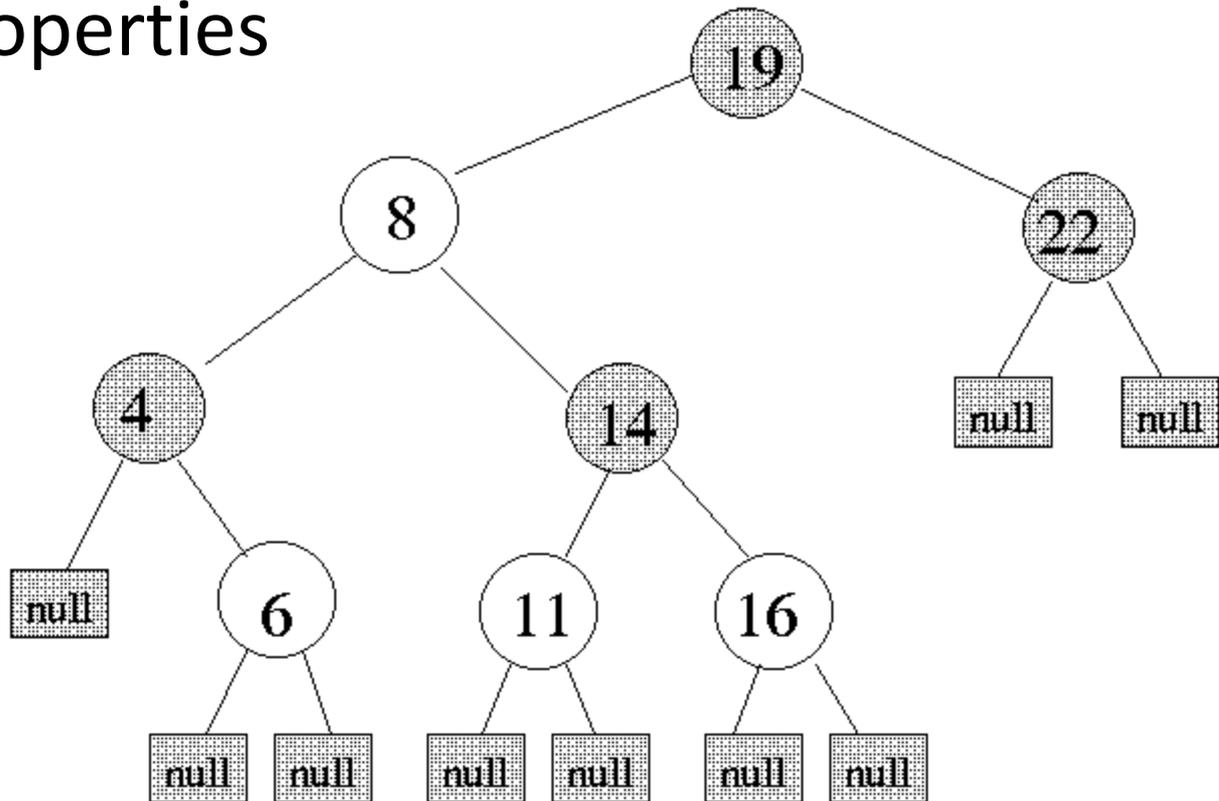
- Invented by Bayr – (though called them something else)
- Robert Tarjan (Turing Award Winner) – noticed the rotations were $O(1)$
- Type of balanced tree – uses color scheme, recoloring and rotations to balance

Red-Black Tree

- Is a Binary Search Tree
- Properties:
 - Every node is red or black
 - The root is black
 - If a node is red, then its children are black
 - Every leaf is a null node and black (external node)
 - Every simple path from a node to a descendant leaf contains the same number of black nodes.

Example red-black tree

- In the figure, black nodes are shaded and red nodes are non-shaded
- Check properties



Example

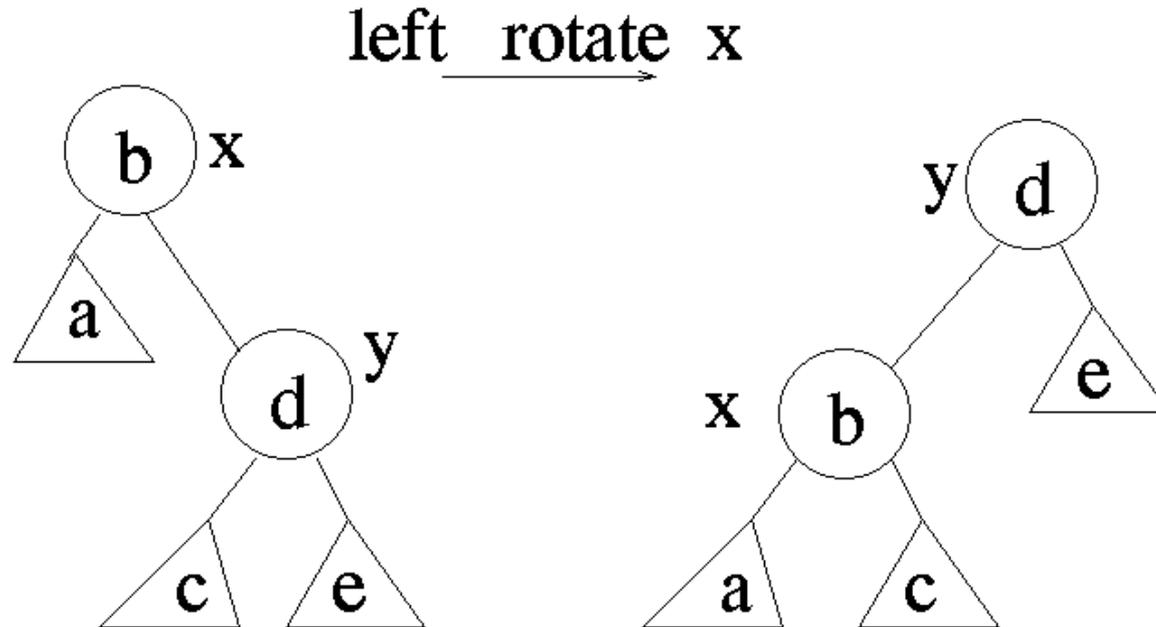
- The five properties ensure that no path is more than twice as long as any other path
- Def. The *height* (h) of a node is the length of the longest path from the node (downward) to a leaf (including external nodes).
- Def. The *black height* (bh) of a node x is the number of black nodes on any path from x (not including x) to a leaf
- Examples: $h(19)$? $bh(19)$? $h(8)$? $bh(8)$?

Height of Red-Black Tree

- Lemma: A red-black tree with n internal nodes has height at most $2 \log (n+1)$
- Operations:
 - Time for search for x :
 - Time for min:
 - Time for list inorder:

Rotations

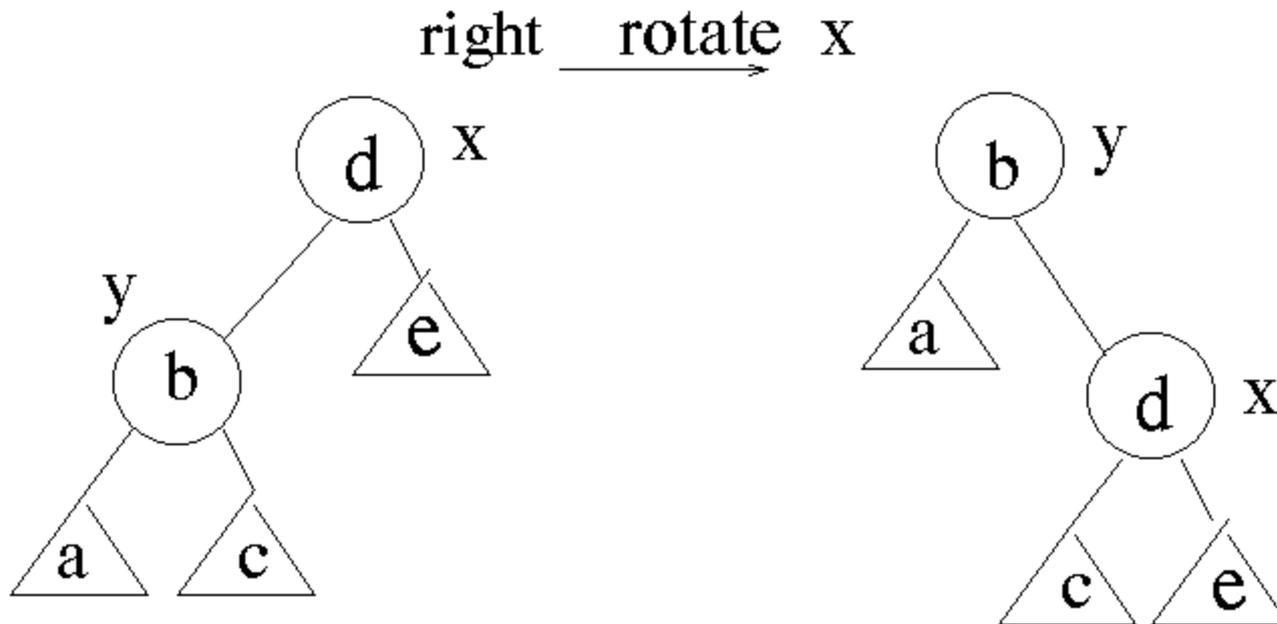
- We want to perform insertions and deletions in $O(\log n)$ time. Adding or deleting a node may disrupt one of its properties, so in addition to some recolorings, we may also have to restructure the tree by performing a rotation (change some pointers).



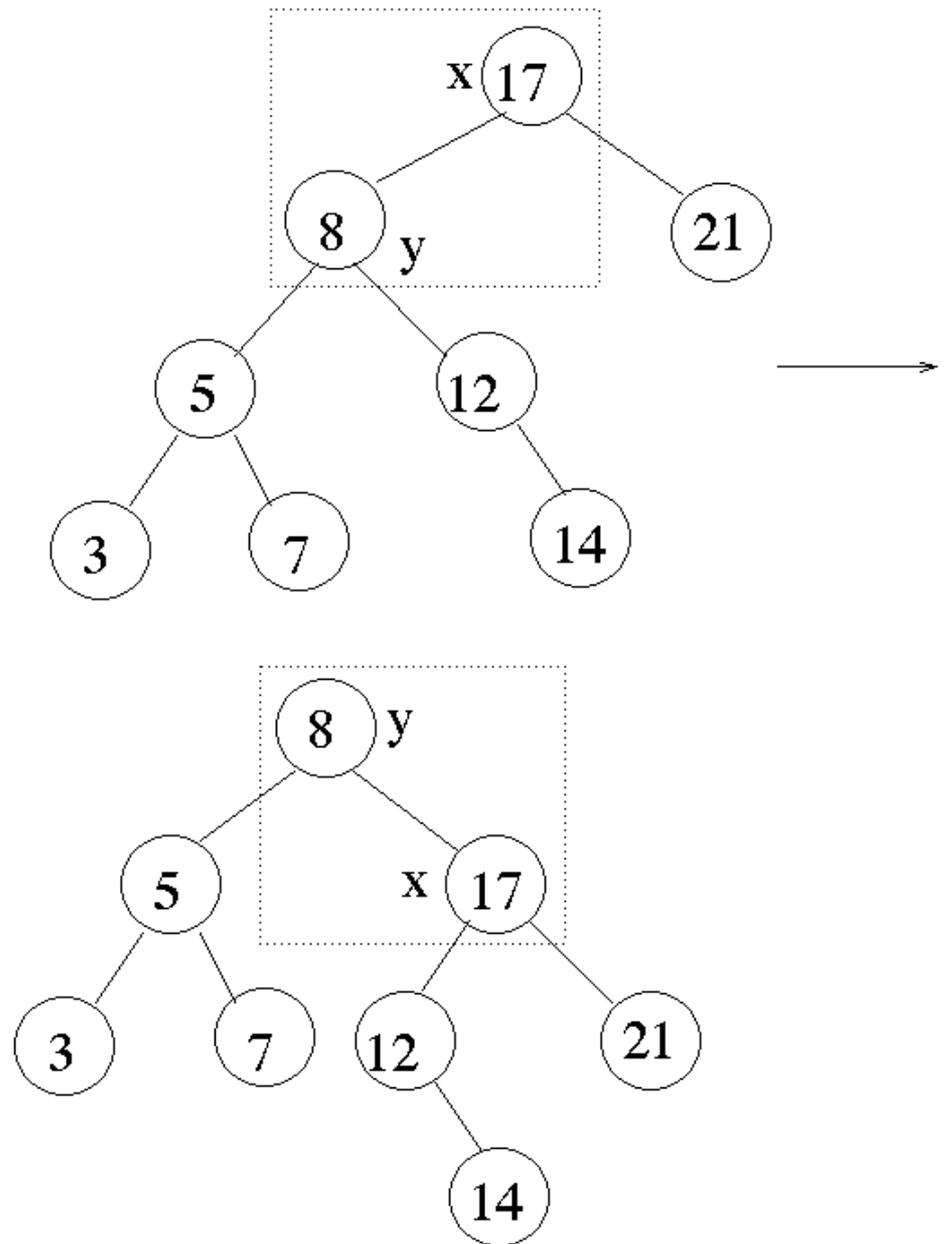
- Note the inorder traversal in both is: abcde

Right Rotate

- Note the rotations change the pointer structure while preserving the inorder property.



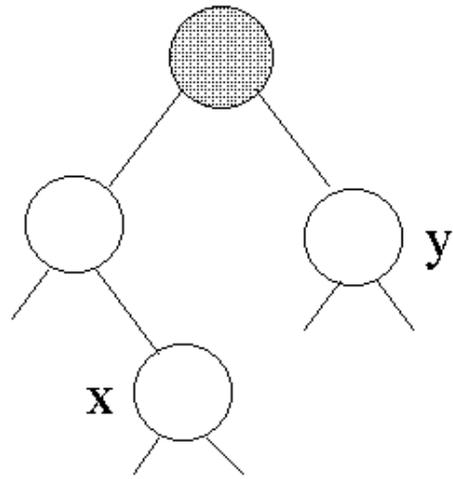
Example of rotation



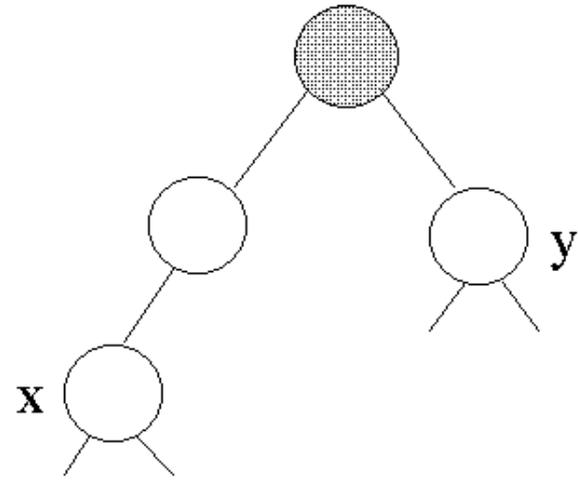
Insertion

- Insert node as RED using a binary search tree insert
 - Means insert as a Red leaf with two black NULL nodes
- Then fix-up so that properties still hold
 - Recoloring and/or 1-2 rotations
- Several cases to consider

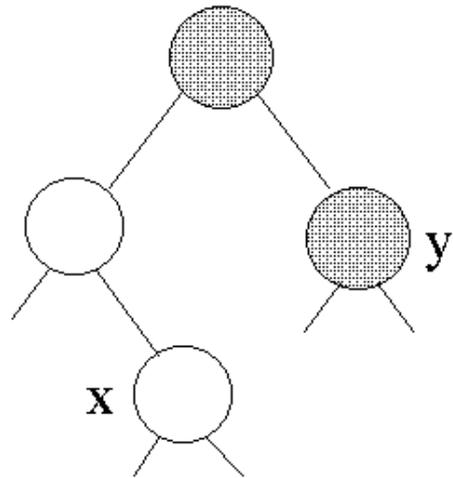
Cases for Insert



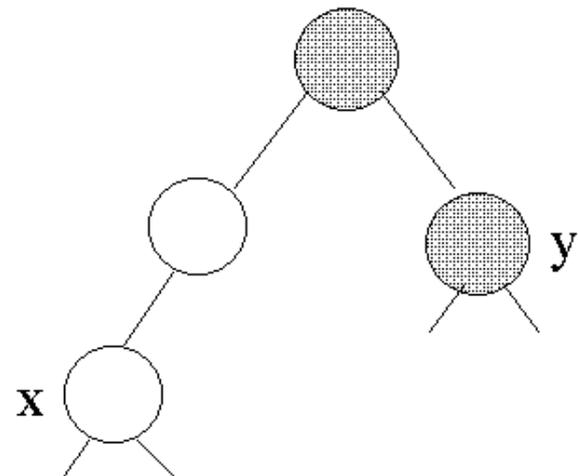
case 1



case 1



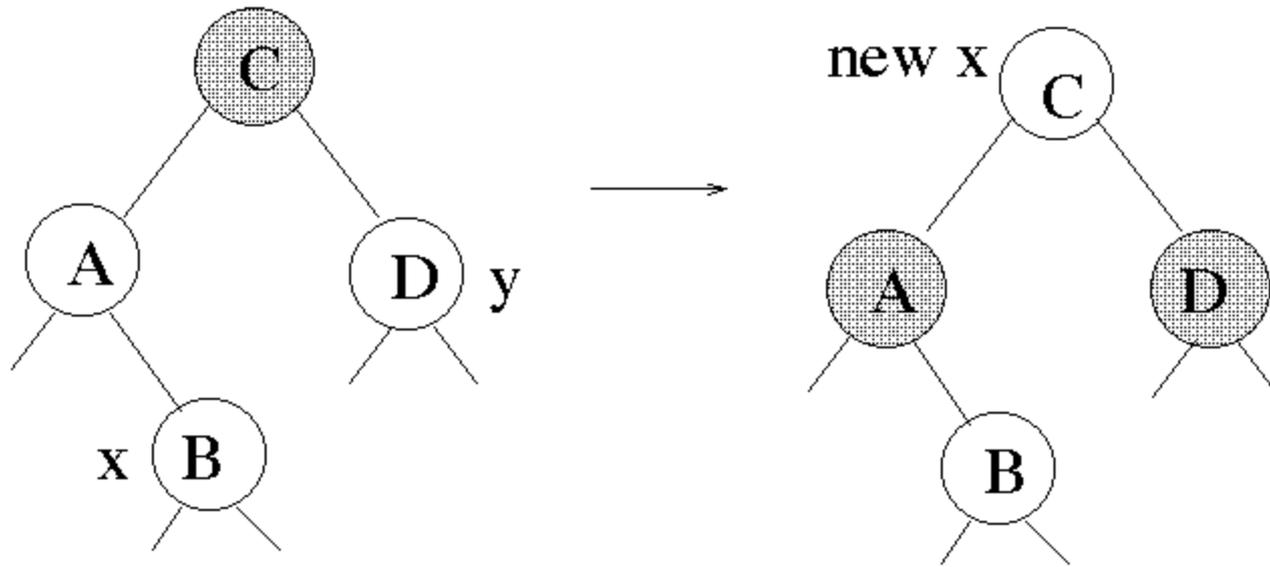
case 2



case 3

Insertion – Case 1 – How to Fix

- Case 1 – sibling of parent of x (called y) is red

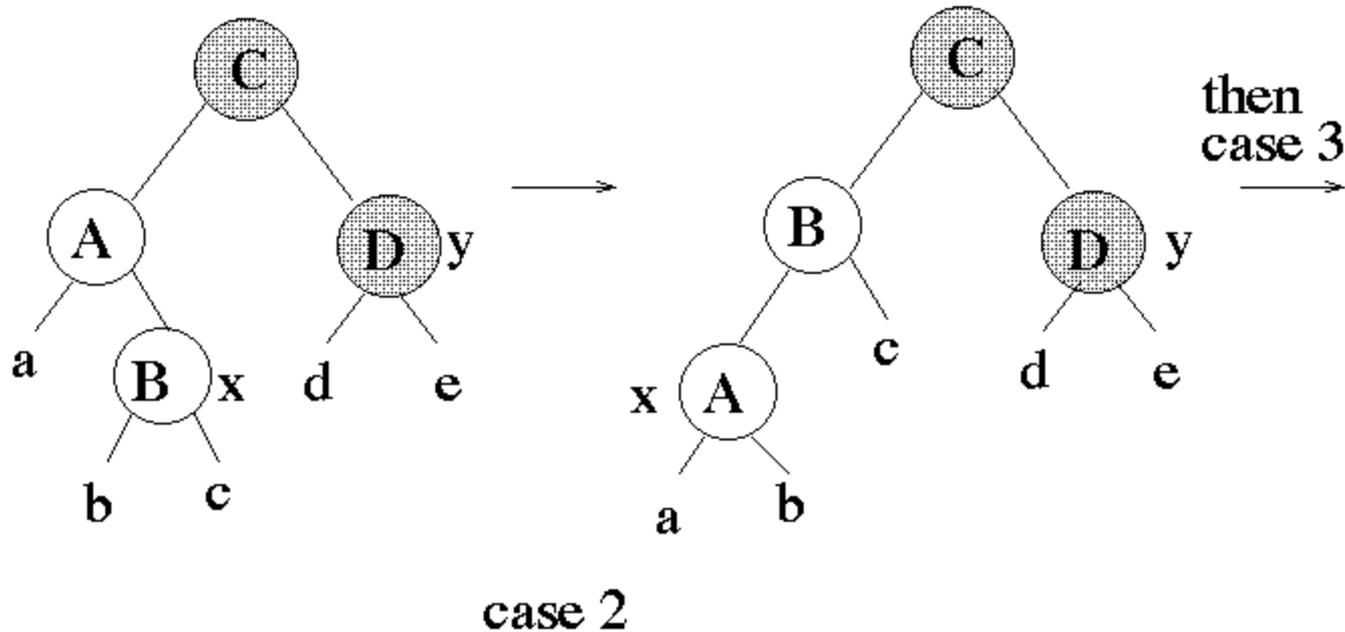


case 1

- To fix: recolor three nodes, then fix up new “x”

Insertion – Case 2 How to Fix

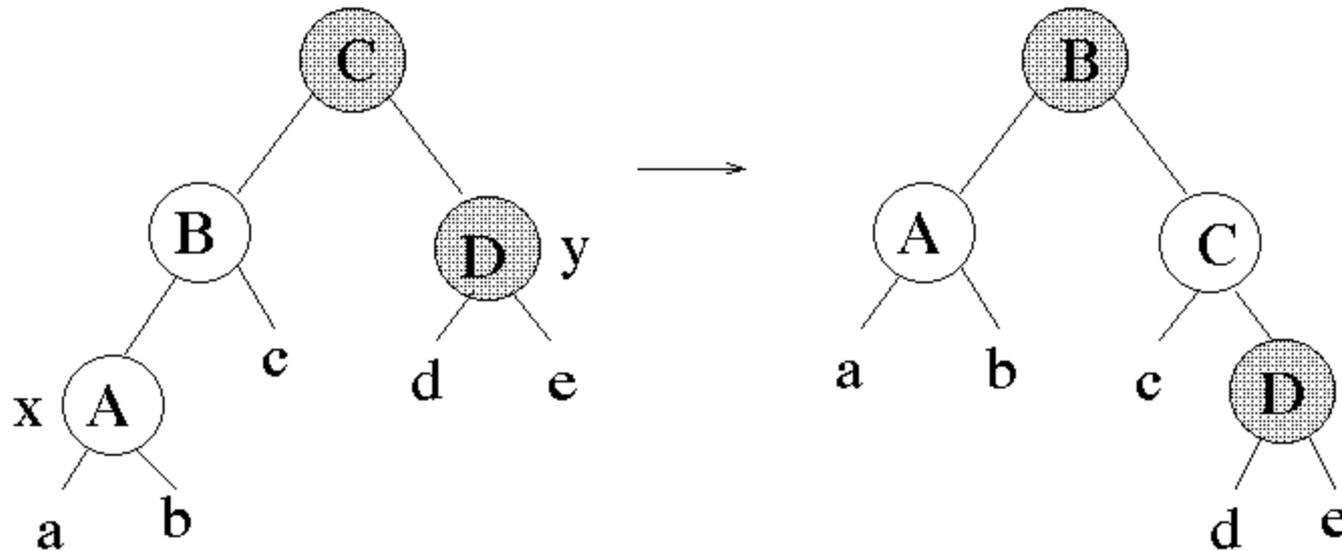
- Sibling of parent of x (call y) is black, x right child



- To fix: set x to parent of x and left rotate x, then it becomes a case 3

Insertion – Case 3 – How to Fix

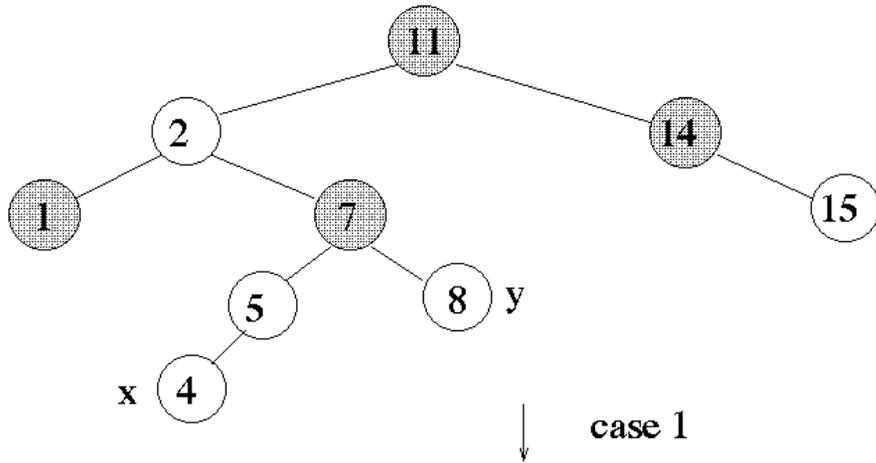
- Case 3 – sibling of parent of x (call y) is black, x left child



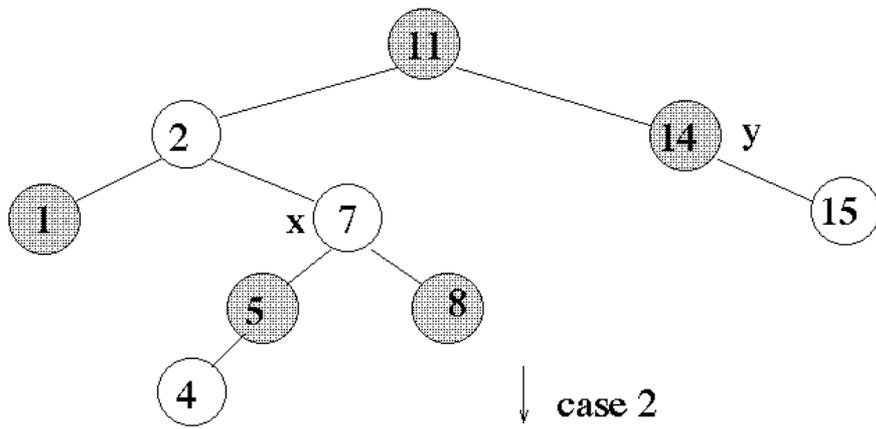
case 3

- To fix: two recolorings and one right rotate of grandparent of x

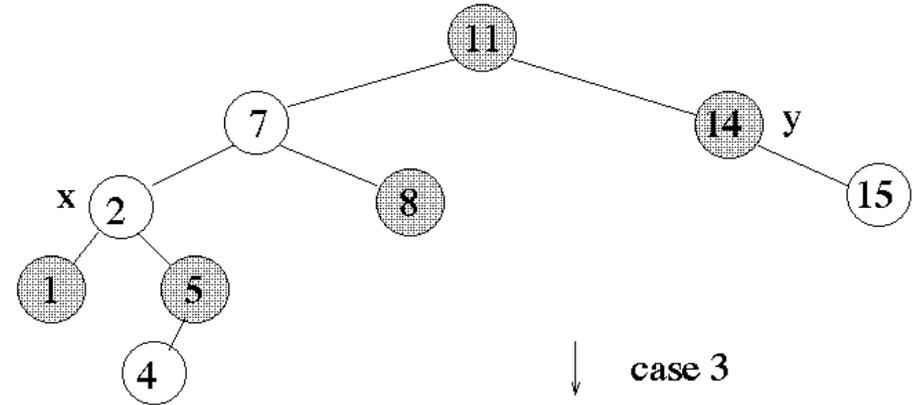
Example of Insert 4 w/ double rotation



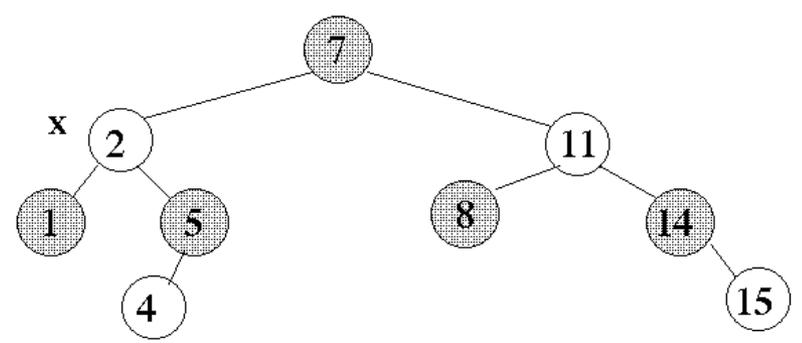
↓ case 1



↓ case 2



↓ case 3



Analysis – Red Black Tree

- Insert
- Deletion