Lecture 16: Red Black Trees 2 - Insertion

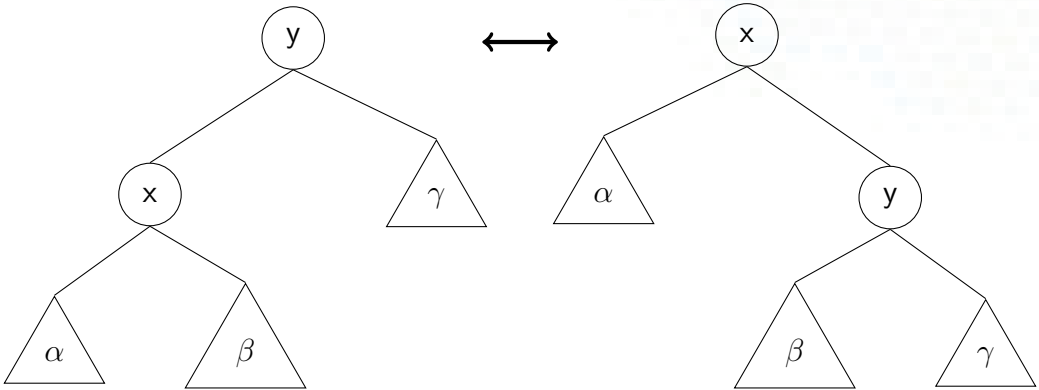
COSC242: Algorithms and Data Structures

Brendan McCane

Department of Computer Science, University of Otago

Rotations

The basis for efficient rebuilding is tree rotation (AVL trees also use rotations):



Rotation algorithm

For right rotation around y: x keeps its left subtree but its right subtree becomes the left subtree of y, y keeps its right subtree, and the parents of x and y change.

```
1:  procedure RBT_ROTATE_RIGHT(x)
2:      y = x→parent
3:      y→left = x→right
4:      x→right = y
5:      x→parent = y→parent
6:      if y→parent == Nil then
7:          root = x
8:      else
9:          y→parent→[left or right] = x
10:     end if
11:     y→parent = x
12: end procedure
```


RBT Insert

The basic algorithm for inserting a node into an RBT is:

```
1: procedure RBT_INSERT(T, x)
2:     BST_insert(T, x)
3:     colour[x] ← red
4:     if parent[x] = red then
5:         RBT_insert_fixup(T, x)
6:     end if
7: end procedure
```

By colouring x red, we may violate the property that says red nodes have black children. Think of x as the *problem node*.

All fixups push the problem back up the tree, so RBT_insert_fixup needs to traverse the tree upwards until either there is no problem anymore, or we reach the root of the tree. This can be done recursively or iteratively.

Try some examples

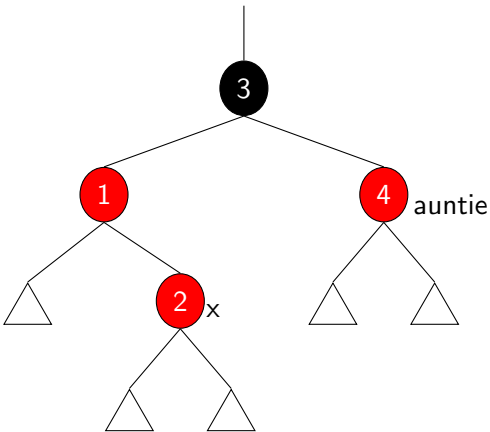
Let's see if we can get an intuitive feel for RBT_insert_fixup. The rules are that you can either recolor the nodes or rotate and recolor the nodes. But you want to do that in the minimum number of operations.

Insert into an RBT and perform fixups after each insert if necessary while inserting the following keys in the following order:

1. 3, 1, 4, 2
2. 3, 2, 1
3. 3, 1, 2

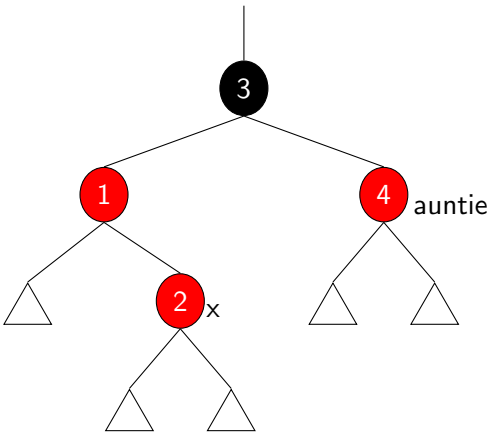
Case 1 Violation

Initial:

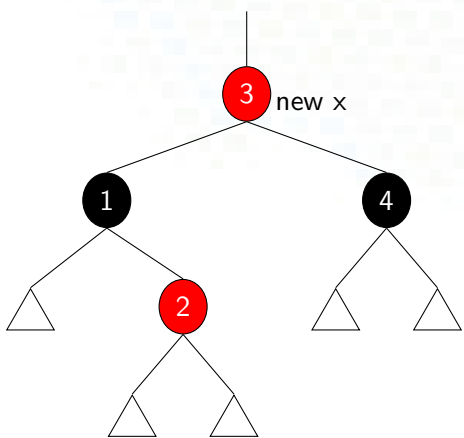


Case 1 Violation

Initial:

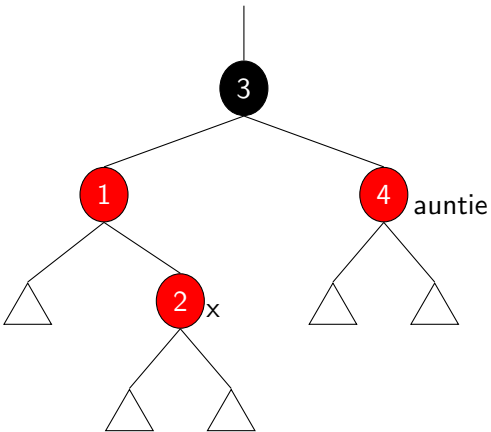


Fixup:

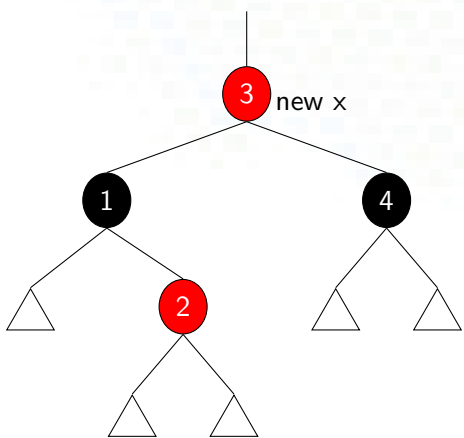


Case 1 Violation

Initial:



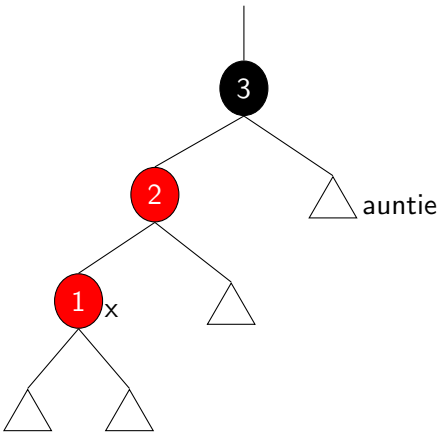
Fixup:



Case 1 is the case when x's auntie is red. It might cause a violation further up the tree.

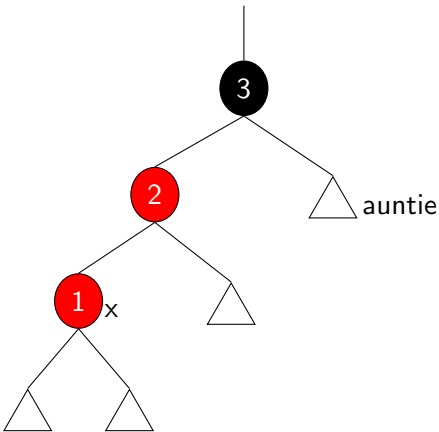
Case 3 Violation

Initial:

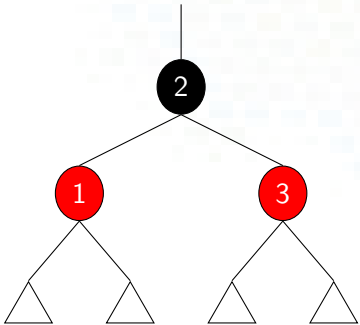


Case 3 Violation

Initial:

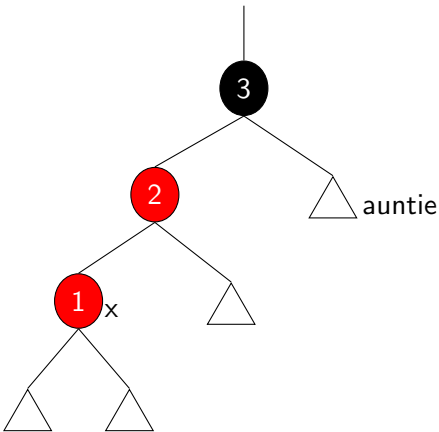


Fixup:

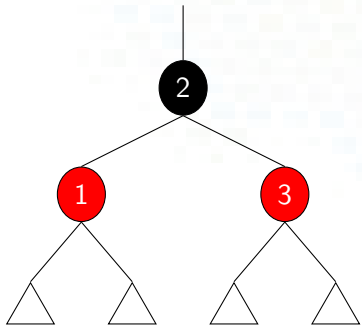


Case 3 Violation

Initial:



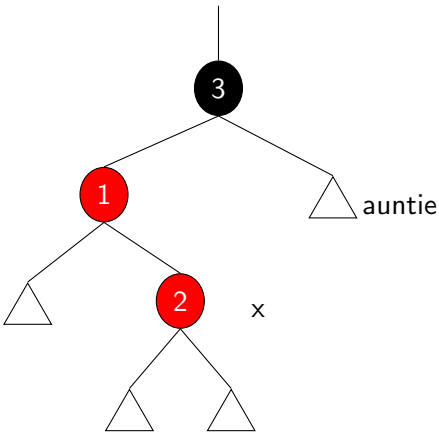
Fixup:



Case 3 is the case when x's auntie is black and x is an outer child.

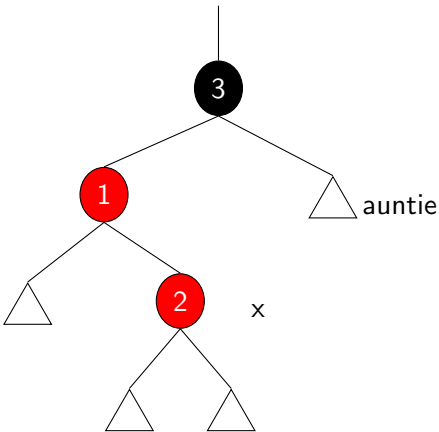
Case 2 Violation

Initial:

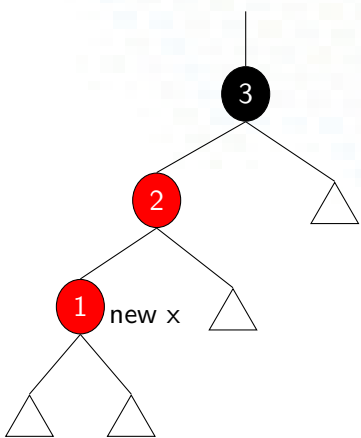


Case 2 Violation

Initial:

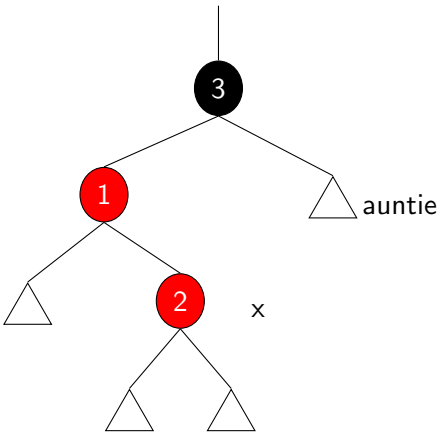


Fixup:

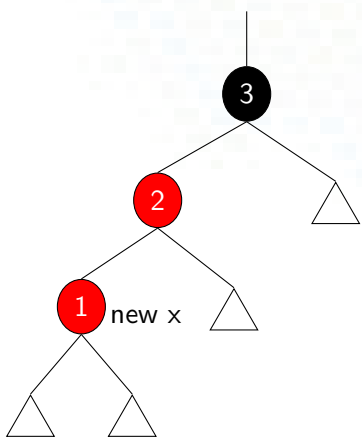


Case 2 Violation

Initial:



Fixup:

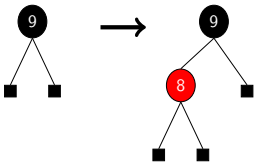


Case 2 is the case when x's auntie is black and x is an inner child.

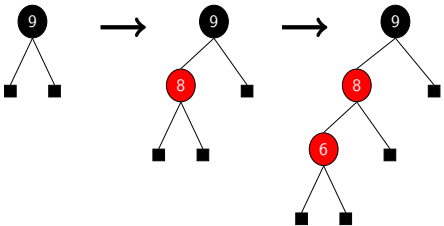
Example Insertions (9, 8, 6, 3, 5)

Example Insertions (9, 8, 6, 3, 5)

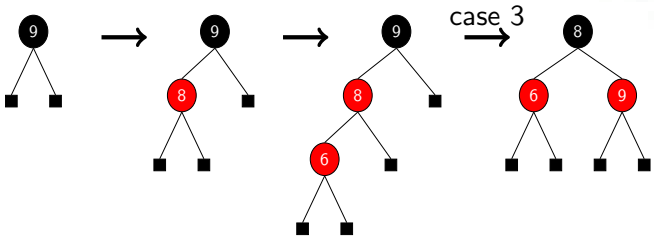
Example Insertions (9, 8, 6, 3, 5)



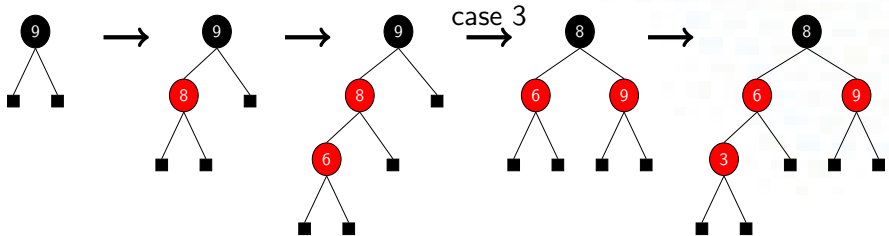
Example Insertions (9, 8, 6, 3, 5)



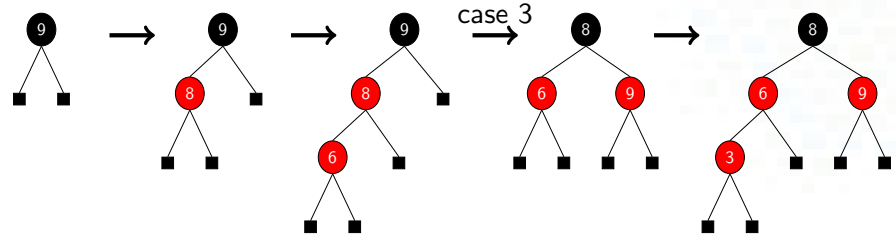
Example Insertions (9, 8, 6, 3, 5)

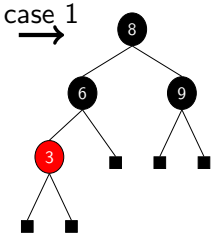


Example Insertions (9, 8, 6, 3, 5)

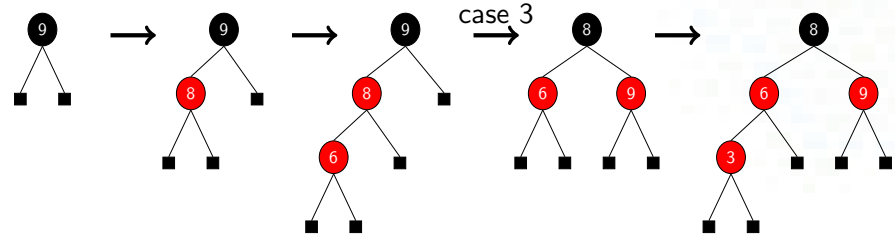


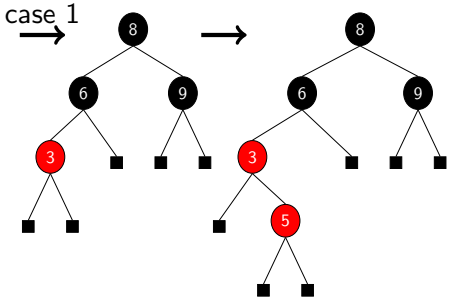
Example Insertions (9, 8, 6, 3, 5)



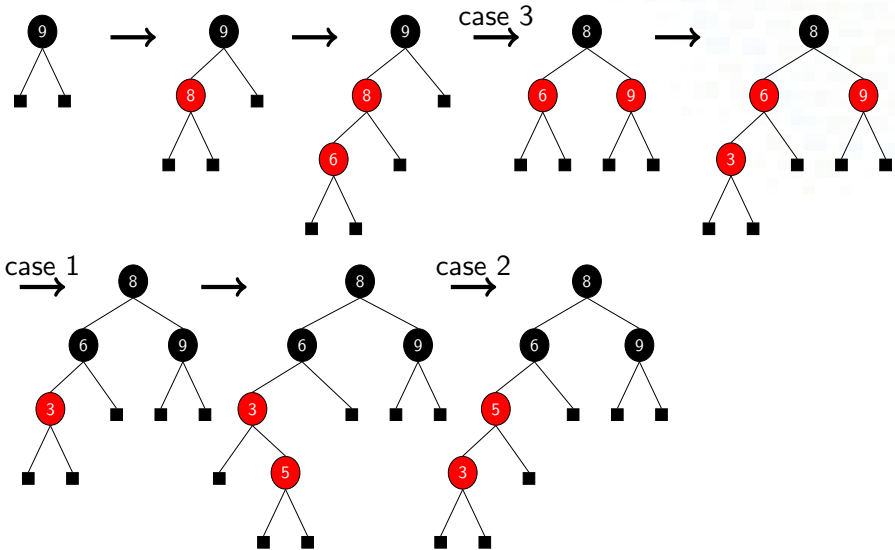


Example Insertions (9, 8, 6, 3, 5)

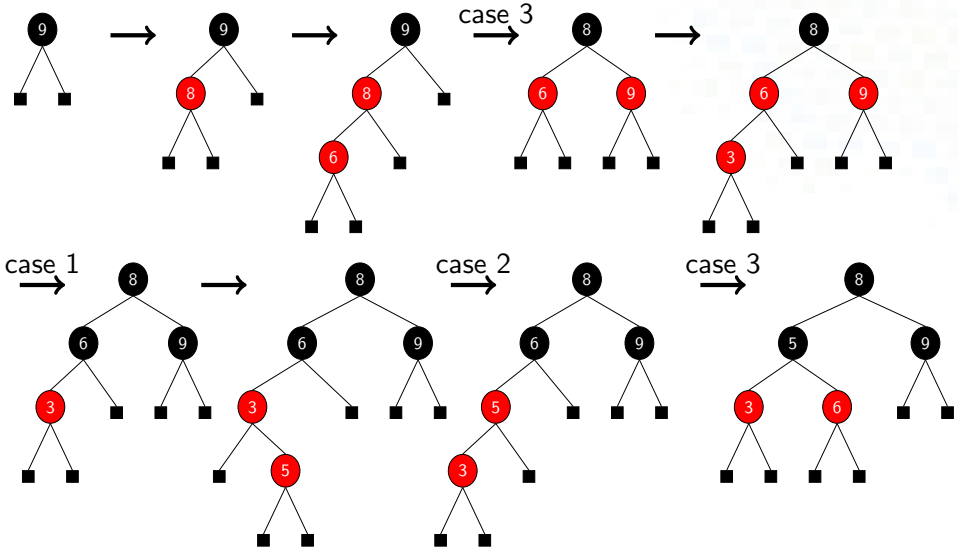




Example Insertions (9, 8, 6, 3, 5)



Example Insertions (9, 8, 6, 3, 5)



Relevant parts of the textbook

Red-Black trees are the topic of Chapter 13 of the textbook.

Today's lecture covered sections 13.3.

Several other tree-based data structures are discussed in the Problems section of Chapter 13.