Lecture 16 - part 2: Red Black Trees 3- Deletion

COSC242: Algorithms and Data Structures

Brendan McCane

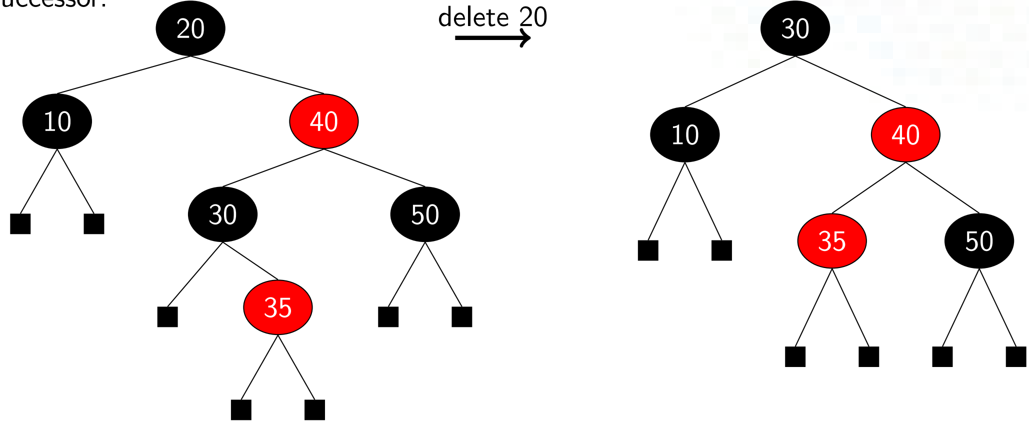
Department of Computer Science, University of Otago

Reminder: BST delete

To delete a node z, BST-deletion recursively searches for z, and then:

1. if z has < 2 children, replace it by a child (possibly nil);
2. if z has two children, replace it by its successor.

Some node, call it y, eventually gets spliced out. It may be that $y = z$, or y may be z's successor.



RBT Deletion

To delete a node, z, in an RBT:

1. delete z as for a BST
2. fix any RBT violations

Call the spliced-out node y (remember that the spliced out node is the node that is removed from the tree - which is not necessarily z).

RBT Deletion

Here are the RBT properties again:

1. Every node is either red or black
2. The root is black
3. All dummy leaves are black (think of dummy leaves as NULL subtrees)
4. If a node is red, then the roots of both its subtrees are black
5. For each node, all paths from the node to the leaves contain the same number of black nodes (including the dummy leaves).

If y is red, can any of those properties be violated?

If y is black, can any of those properties be violated?

If y is black

If y is black, what can go wrong:

- If y was the root, the new root might be red (property 2). This is easy to fix - just make the root black.
- Splicing out y might leave two red nodes in a parent-child relationship.
- The black-height of at least one path will be reduced by one (an imbalance). This is always the case when splicing out a black node.

Let's start by defining some labels:

- z is the node to be deleted
- y is the node that gets spliced out (sometimes $y = z$ and sometimes y is z's successor)
- x is the child that replaced y
- w is the new sibling of x

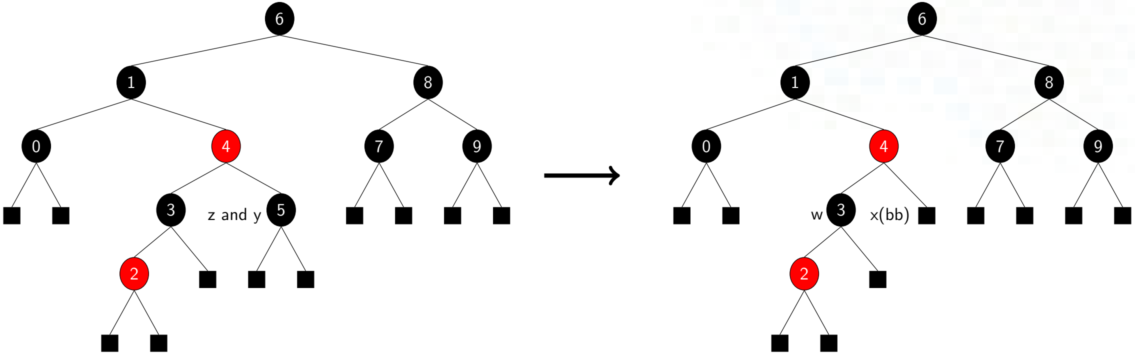
Four cases

1. x's sibling, w, is red, fix then fall to case 2, 3 or 4
2. w is black and has two black children, fix then traverse up the tree
3. w is black and w's inner child is red and outer child is black, fix then fall to case 4
4. w is black and w's outer child is red, fix and terminate

The trick to fixing all of these is to give x an extra black value. Since we've removed a black node, giving x the value of 2×black restores the black-height property of the tree. We move this extra black value up the tree until we can either give it to a red node, or until we reach the root.

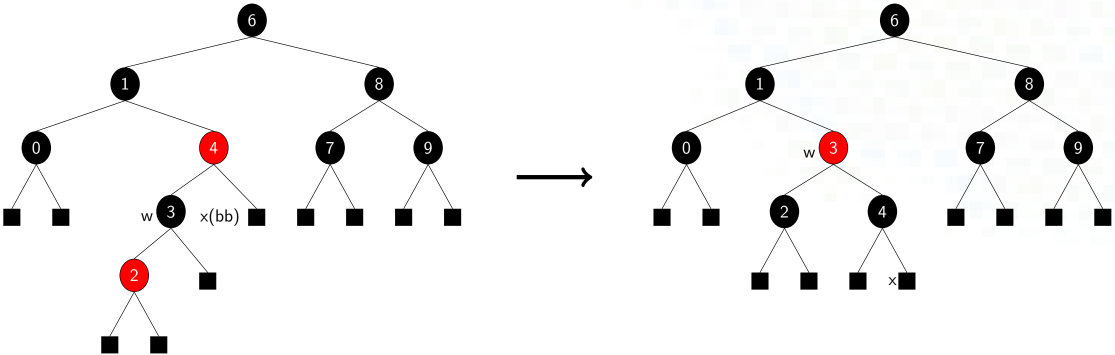
Case 4 Example: w is black and w's outer child is red

Let's try some examples. Consider the following RBT:



See if you can recolor or rotate and recolor to maintain the RBT properties when when you delete 5 from the tree.

Case 4 Example: w is black and w's outer child is red

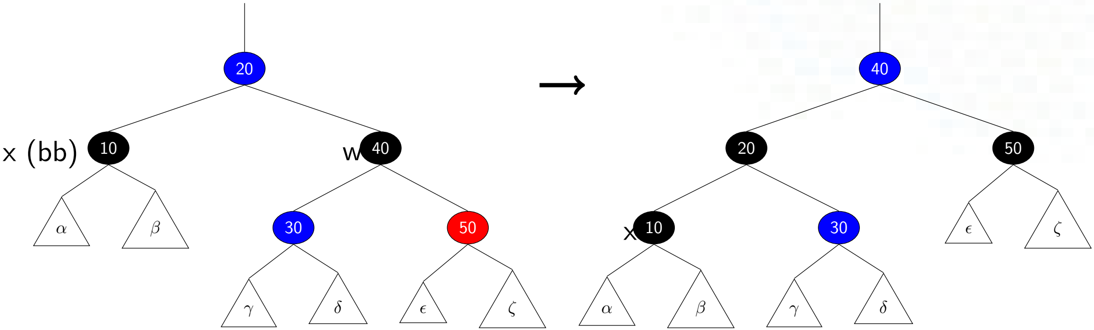


Solution: rotate w towards x and keep all colors in the same positions.

The black-height of the sub-trees rooted at w are now equal, and we can stop.

Case 4: w is black and w's outer child is red

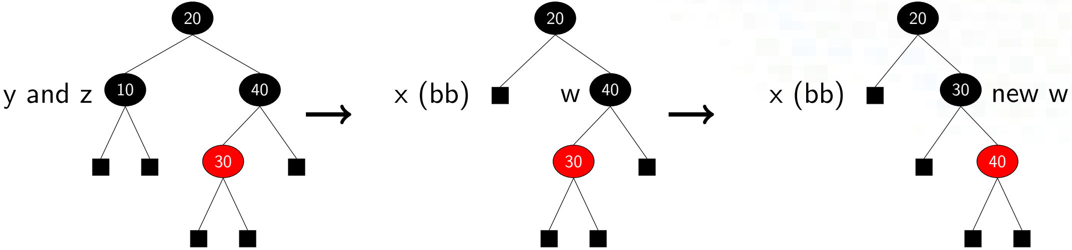
Note that blue means the node could be red or black.



And now we are done.

Case 3 Example: w's inner child is red and outer child is black

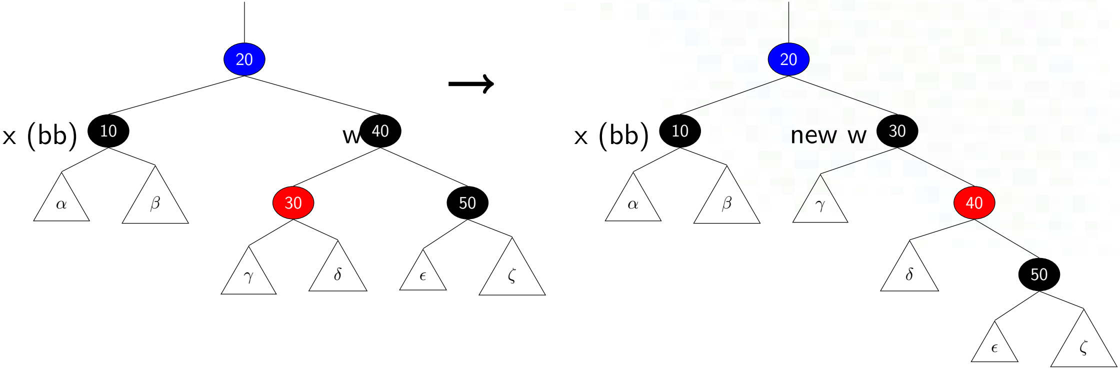
Delete 10 from the following tree:



Solution: rotate around w towards the outside.

Now we have case 4.

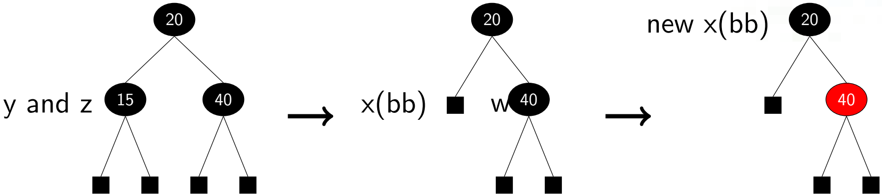
Case 3: w's inner child is red and outer child is black



Now we have Case 4.

Case 2 Example: x’s sibling is black and has two black children

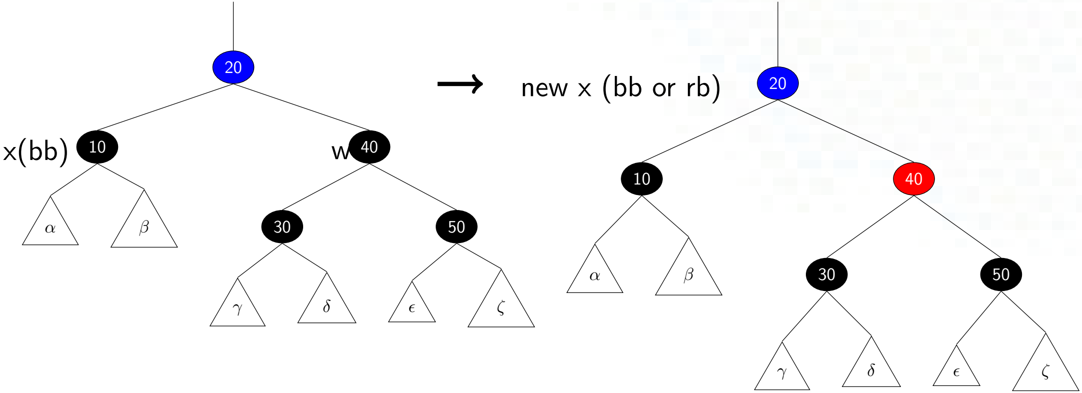
Delete 15 from the following tree:



Solution: move a black color up one level (borrow a black from x and w)

Either we are done (rb) or we head up the tree with a new x.

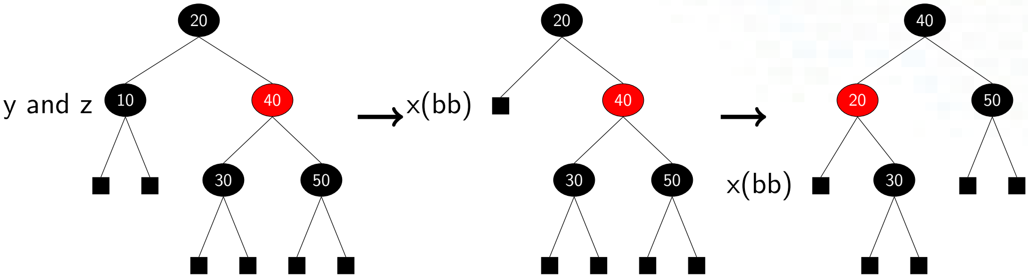
Case 2: x's sibling is black and has two black children



Either we are done (rb) or we head up the tree with a new x.

Case 1 Example: x's sibling w is red

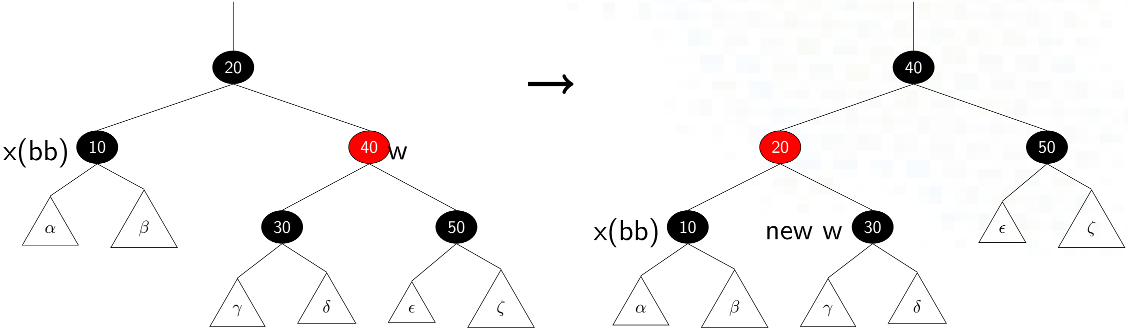
Delete 10 from the following RBT:



Solution: rotate towards x around x's parent.

Now we have case 2 (but it could have been case 3 or 4).

Case 1: x's sibling w is red



Now we have case 2, 3 or 4.

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.