Red Black Trees

Top-Down Deletion

Reminder: Rules for Binary Search Tree Deletion

1. If the node to be deleted is a leaf, just delete it.
2. If the node to be deleted has just one child, replace it with that child
3. If the node to be deleted has two children, replace the **value** of it by it's in-order predecessor's value then delete the in-order predecessor (a recursive step) This case reduces to case 1.

Issues in Deleting from Red Black Trees

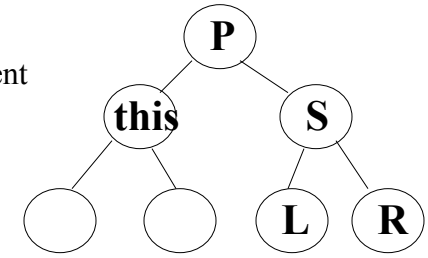
1. The node to be deleted is red:
   Not a problem – no RB properties are violated
2. The node to be deleted is black:
   If the node is not the root, deleting it will change the black-height along paths going through it.

Goal of Top Down Deletion

- Delete a red node
- How do we ensure this?
  - As we traverse the tree looking for the node to delete, we change the nodes along the path to red nodes.
  - If this causes a violation of the RB properties, we fix it through rotations.
  - Recall: Null nodes are **black**.

Terminology

- **this** is the node being examined
- *S* is **this**'s sibling
- *P* is **this**'s (and S's) parent
- *R* is S's right child
- *L* is S's left child



- This discussion assumes **this** is the left child of P. As usual, there are left-right symmetric cases.

Step 1 – Set-up

Examine the root.

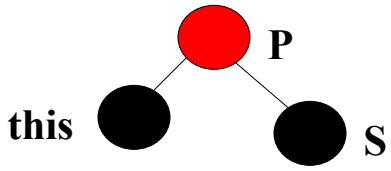
If both of the root's children are **black**

a. Make the root red
b. If the root is to be deleted go to step 3
c. Else, move **this** to the appropriate child of the root and proceed to step 2

Otherwise proceed directly to step 2B on **this**.

Step 2 – The main case

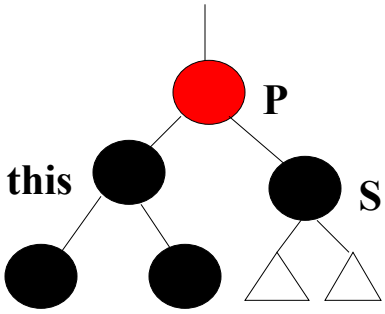
When reaching this part of the algorithm, we should have the following situation: **this** is **black**, P is red, T is **black:**



If **this** has 2 **black** children, go to step 2A.

If **this** has at least one red child, go to step 2B.

Case 2A
this has two **Black** Children



In this case, we consider S:

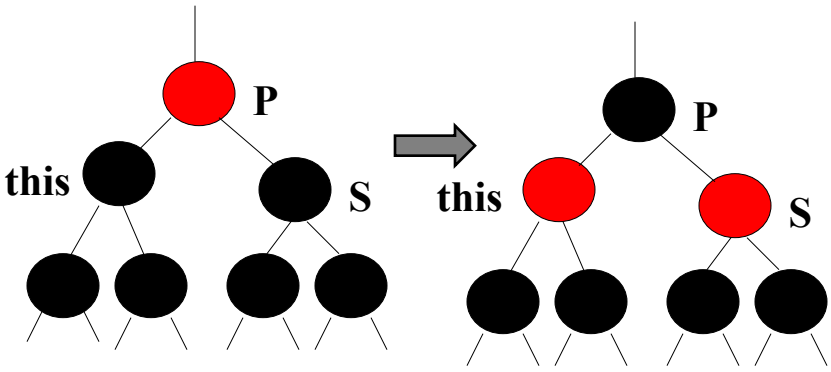
2A1. S has 2 **black** Children

2A2. S's inside child is red

2A3. S's outside child is red

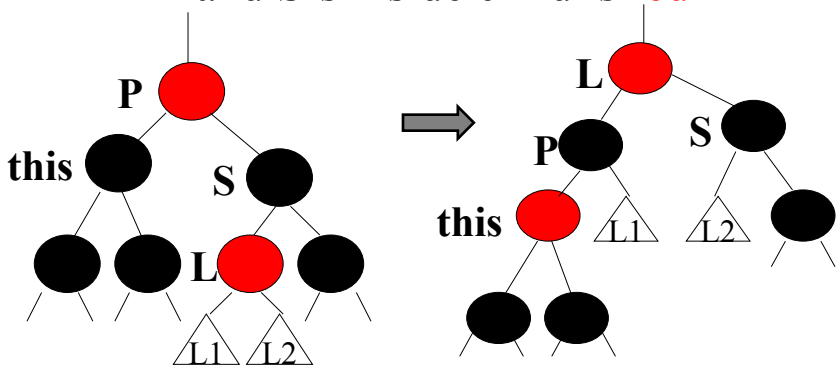
2A4. If both of S's children are red, we can do either 2A2 or 2A3. The test cases assume you do case 2A3.

Case 2A1: **this** and S have 2 **Black** Children



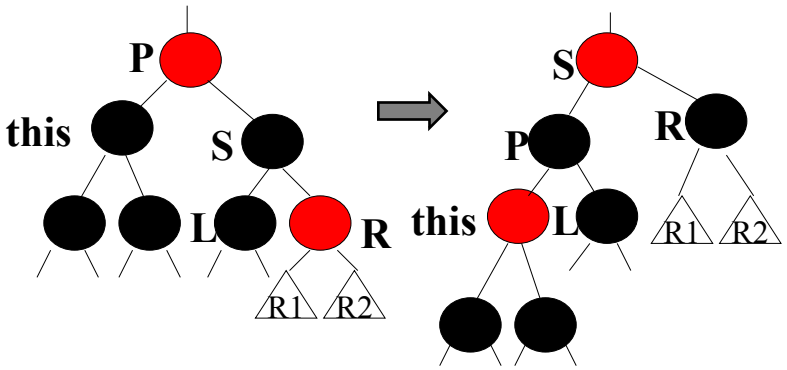
Action: 1) Recolor **this**, P and S
2) If **this** is the node to be deleted, go to step 3
else move down the tree and go back to step 2

Case 2A2: **this** has 2 **black** children and S's inside child is red



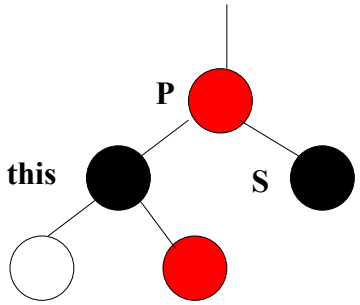
Action: 1) Perform a *double* rotation on P, S, and L.
2) Recolor **this** and P
3) If **this** is the node to be deleted, go to step 3
else move down the tree and go back to step 2

Case 2A3: **this** has 2 **black** children and S's outside child is red



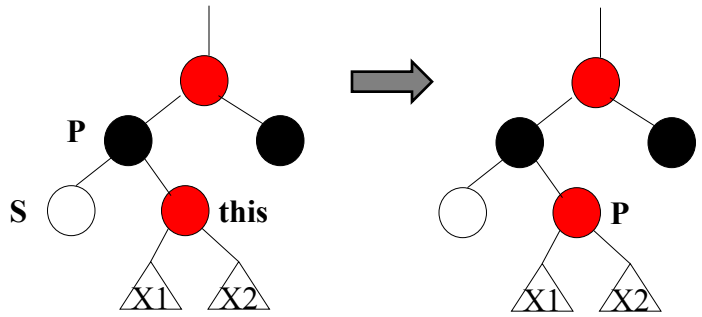
Action: 1) Perform a *single* rotation on P, S, and R.
2) Recolor **this**, P, S and R
3) If **this** is the node to be deleted, go to step 3
else move down the tree and go back to step 2

Case 2B: **this** has at least one Red child



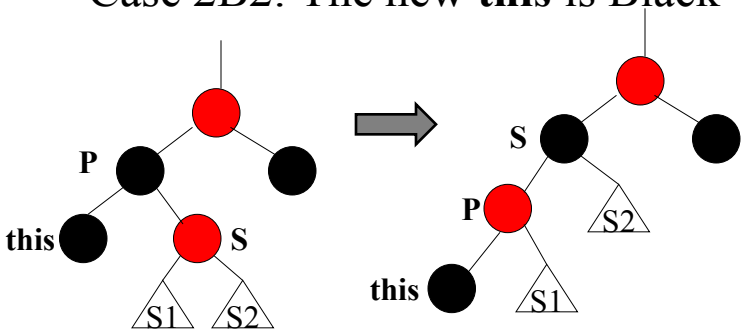
Action: If **this** is the node to be deleted, go to step 3
else move down the tree. The new node is either red or black. If it is red, go to step 2B1, if it is black go to step 2B2.

Case 2B1: The new **this** is Red



Action: If **this** is to be deleted, go to step 3
else move down the tree again and go back to step 2

Case 2B2: The new **this** is Black



Action: 1) Rotate S around P
2) Recolor P and S
3) Go back to step 2 (We will check again whether **this** is the node to be removed. It won't reduce to this case.)

Step 3: Deletion

a. If **this** has two children, obtain the largest value V of the left sub-tree. If **this** is red, copy V to **this**, move **this** to the left child and go to step 2 with value V. If **this** is black, store V in a temporary variable, do Not move **this** and go to step 2b with V. When finished with the removal, place V into **this**.
b. If **this** is a leaf, it must be red and can be safely deleted. Do so and go to step 4.
c. If **this** has a single child, then **this** may be red or black. It is black only in case 2B. In this case, **this** has exactly one red child. Color that child black and remove **this**. Then go to step 4.

Step 4: Finish up

- Color the Root Black