Red Black Trees

Top-Down Insertion

Review of Bottom-Up Insertion

- In B-Up insertion, “ordinary” BST insertion was used, followed by correction of the tree on the way back up to the root
- This is most easily done recursively
  - Insert winds up the recursion on the way down the tree to the insertion point
  - Fixing the tree occurs as the recursion unwinds

Top-Down Insertion Strategy

- In T-Down insertion, the corrections are done while traversing down the tree to the insertion point.
- When the actual insertion is done, no further corrections are needed, so no need to traverse back up the tree.
- So, T-Down insertion can be done iteratively which is generally faster

Goal of T-D Insertion

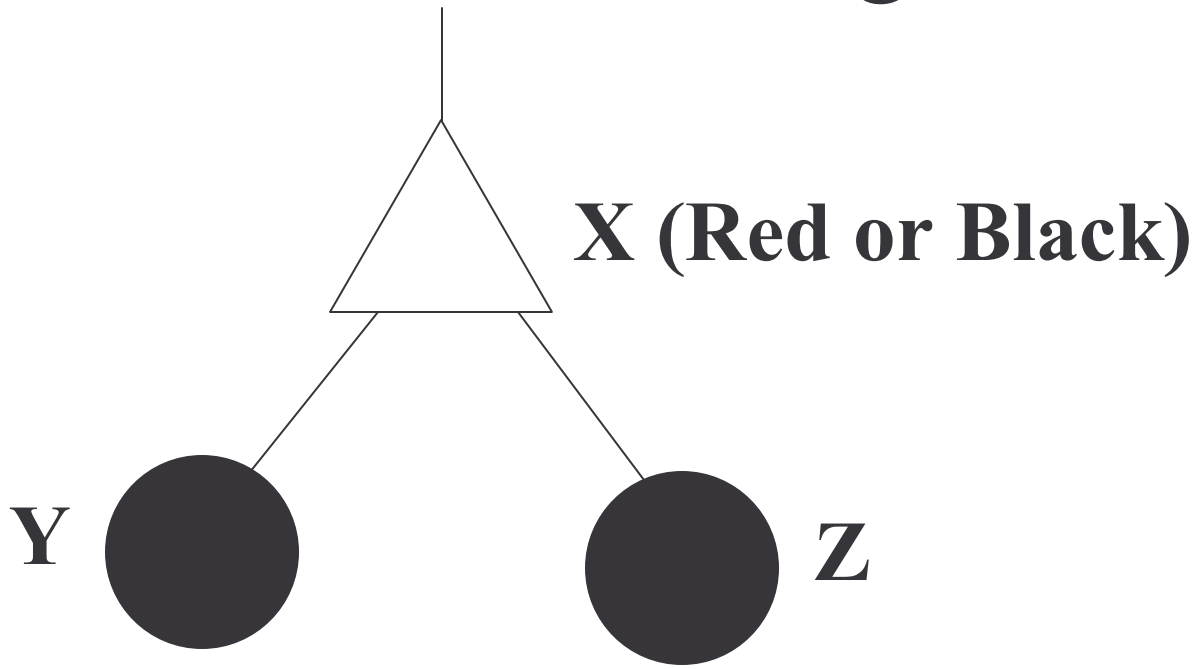
- Insertion is always done as a leaf (as in ordinary BST insertion)
- Recall from the B-Up flow chart that if the uncle of a newly inserted node is Black, we restore the RB tree properties by one or two local rotations and recoloring – we do not need to make changes further up the tree

Goal (2)

- Therefore, the goal of T-D insertion is to traverse from the root to the insertion point in such a way that RB properties are maintained, and at the insertion point, the uncle is Black.

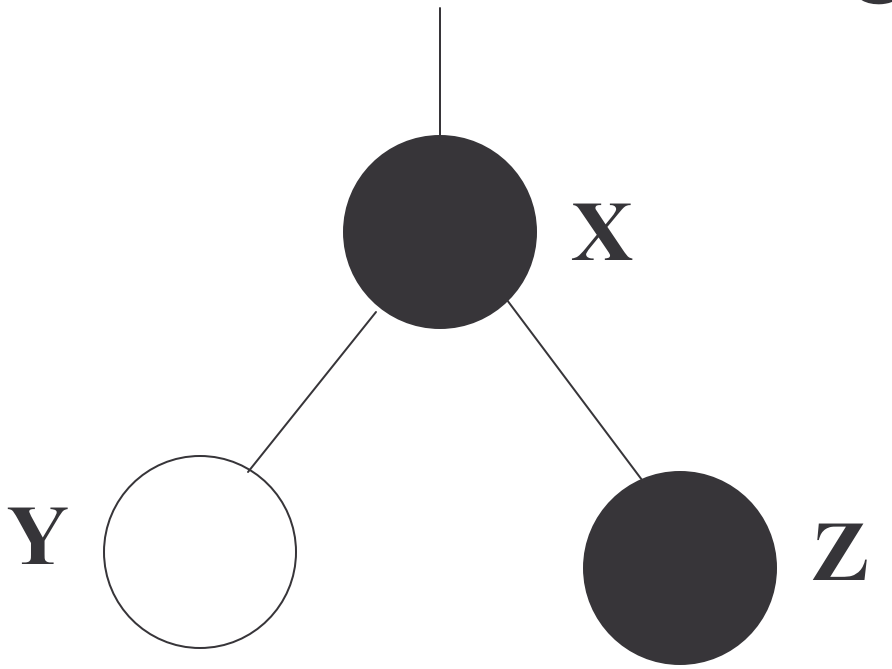
- That way we may have to rotate and recolor, but not propagate back up the tree

Possible insertion configurations



If a new node is inserted as a child of Y or Z, there is no problem since the new node's parent is Black

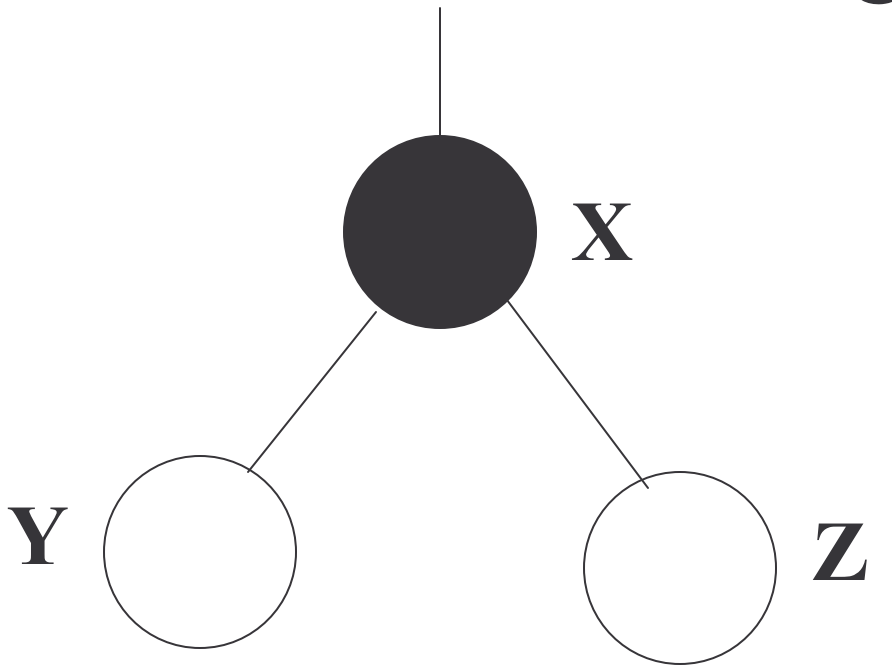
Possible insertion configurations



If new node is child of Z, no problem since Z is Black.

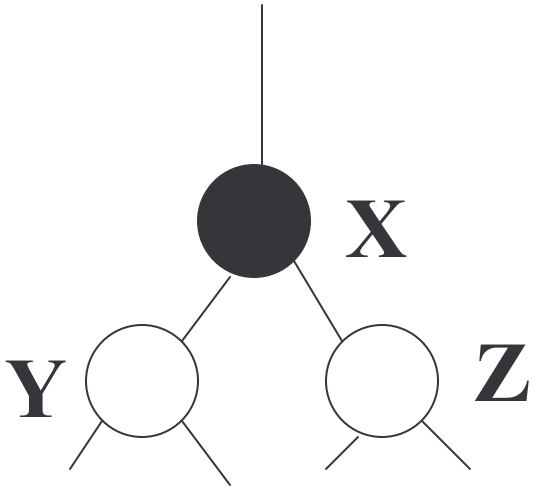
If new node is child of Y, fixable problem since the new node's uncle (Z) is Black – do a few rotations and recolor…. done

Possible insertion configurations



If new node is inserted as child of Y or Z, its uncle will be Red and we will have to go back up the tree. This is the only case we need to avoid.

Top-Down Traversal

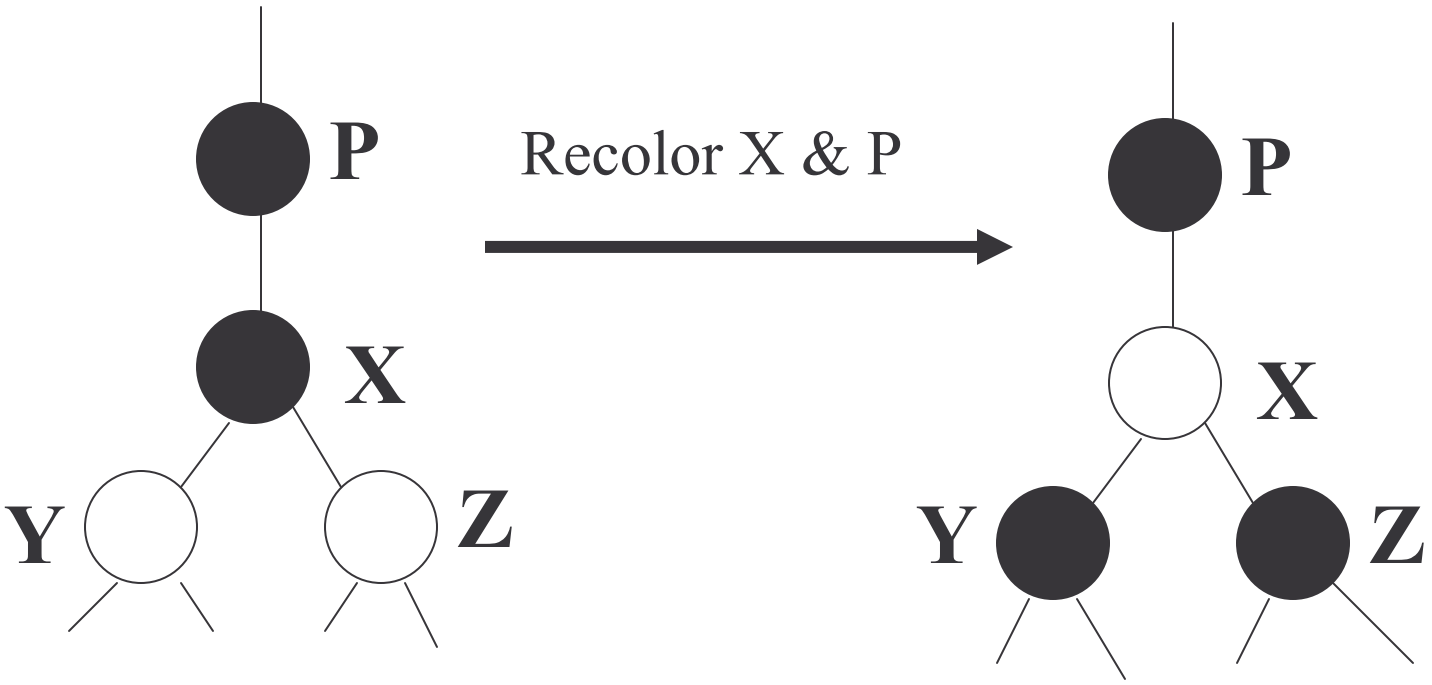


As we traverse down the tree and encounter this case, we recolor and possibly do some rotations.

There are 3 cases.

Remember the goal – to create an insertion point at which the parent of the new node is Black, or the uncle of the new node is Black.

Case 1 – X's Parent is Black

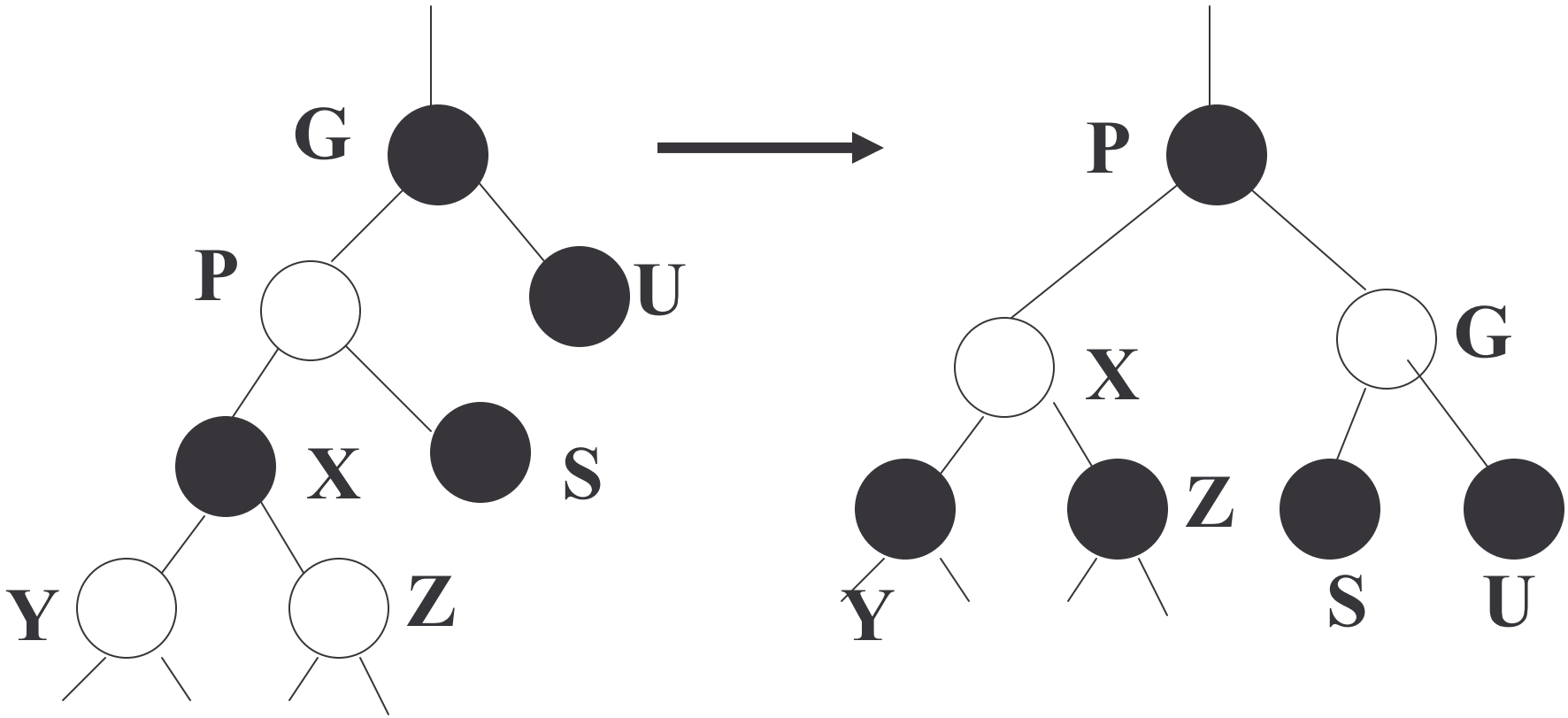


Just recolor and continue down the tree

Case 2

- X's Parent is Red (so Grandparent is Black) and X and P are both left/right children
  - Rotate P around G
  - Color P Black
  - Color G Red
- Note that X's uncle, U, must be Black because it (a) was initially Black, or (b) would have been made Black when we encountered G (which would have had two Red children -- X's Parent and X's uncle)

Case 2 diagrams

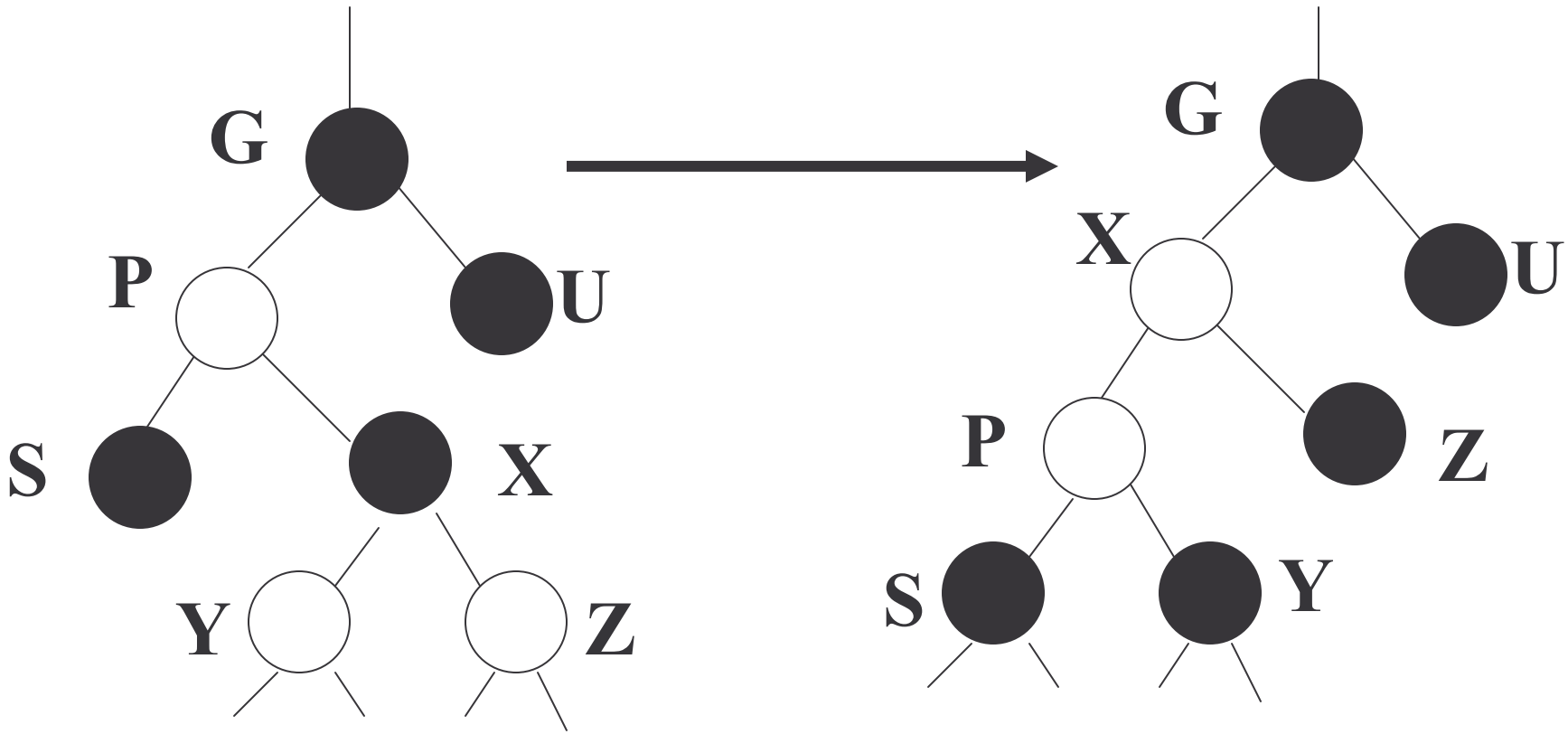


Rotate P around G. Recolor X, Y, Z, P and G

Case 3

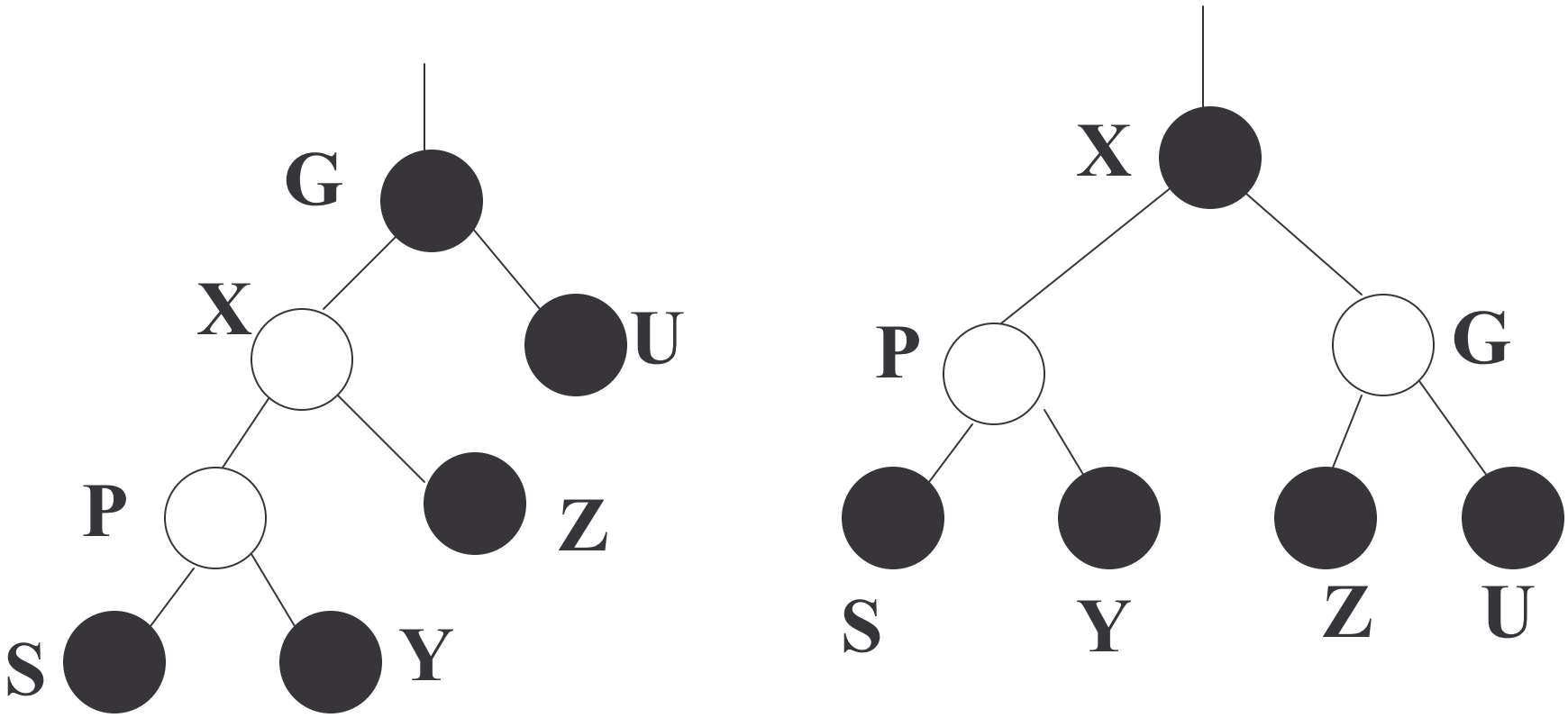
- X's Parent is Red (so Grandparent is Black) and X and P are opposite children
  - Rotate P around G
  - Color P Black
  - Color G Red
- Again note that X's uncle, U, must be Black because it (a) was initially Black, or (b) would have been made Black when we encountered G (which would have had two Red children -- X's Parent and X's uncle)

Case 3 Diagrams (1 of 2)



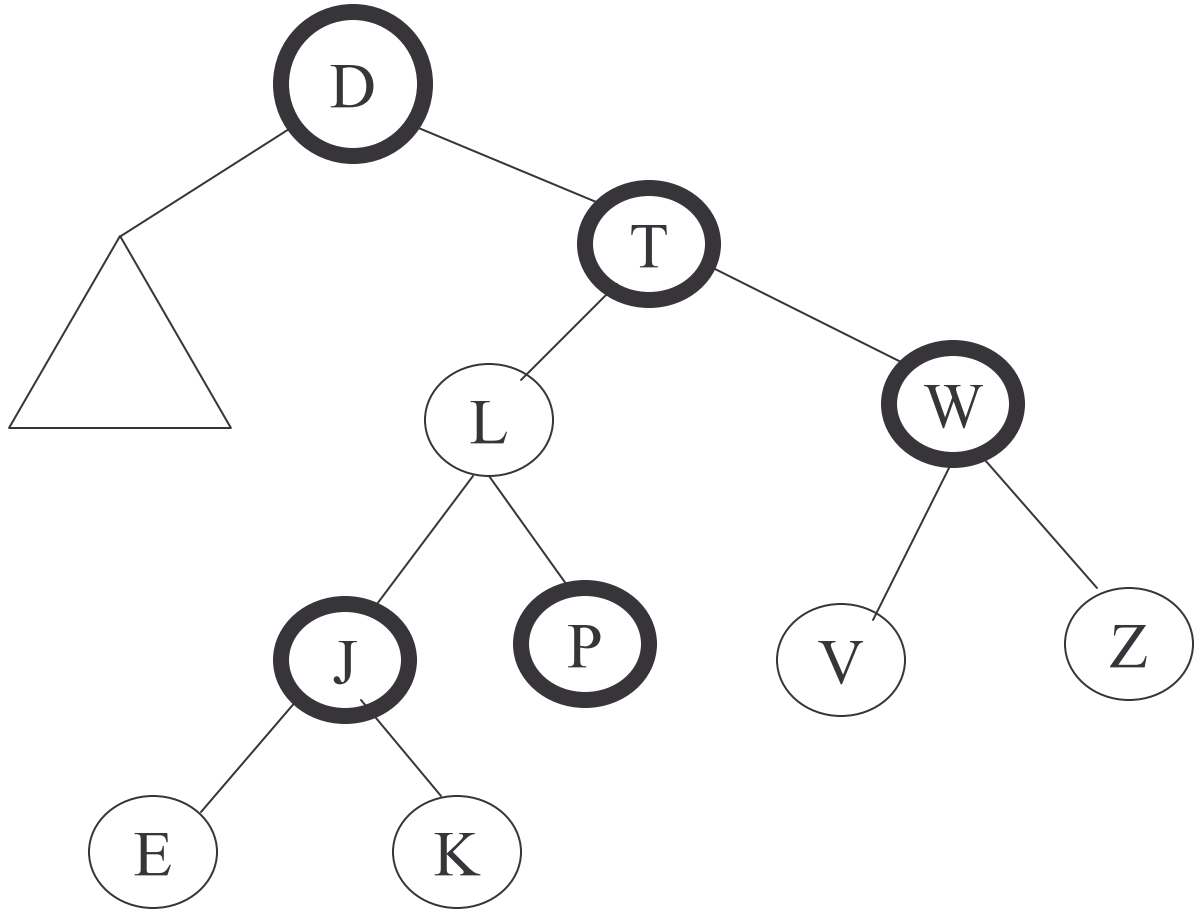
Step 1 – recolor X, Y and Z. Rotate X around P.

Case 3 Diagrams (2 of 2)



Step 2 – Rotate X around G. Recolor X and G

An exercise – insert F

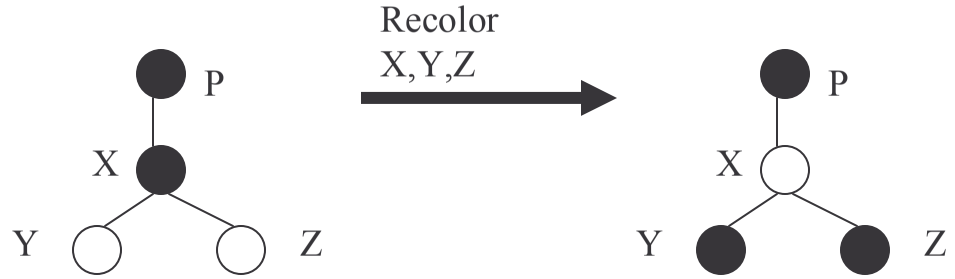


Top-Down Insert Summary

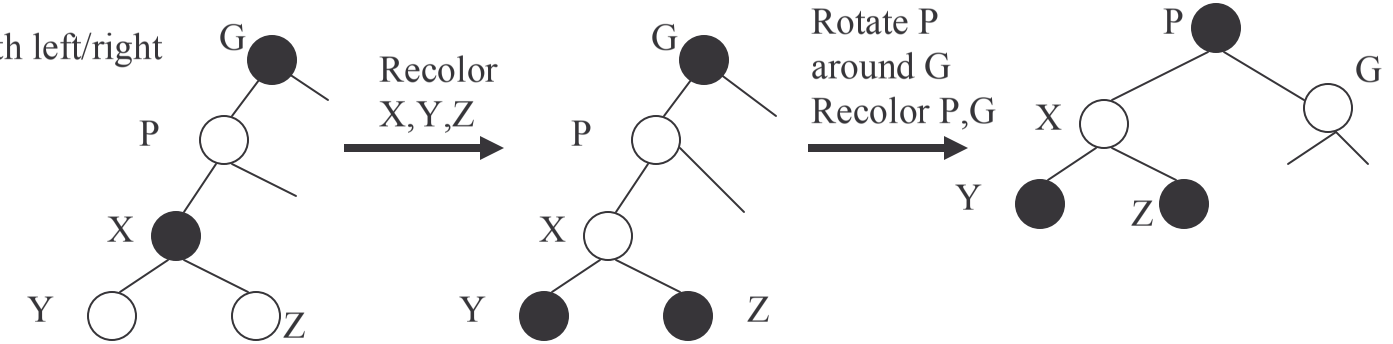
Case 1

P is Black

Just Recolor



Case 2
P is Red
X & P both left/right



Case 3
P is Red
X and P are
opposite children