

Red-Black Trees

Bottom-Up Deletion

Recall “ordinary” BST Delete

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

Bottom-Up Deletion

1. Do ordinary BST deletion. Eventually a “case 1” or “case 2” deletion will be done (leaf or just one child).
 - If deleted node, U , is a leaf, think of deletion as replacing U with the NULL pointer, V .
 - If U had one child, V , think of deletion as replacing U with V .
2. What can go wrong??

Which RB Property may be violated after deletion?

1. If U is Red?
Not a problem – no RB properties violated
2. If U is Black?
If U is not the root, deleting it will change the black-height along some path

Fixing the problem

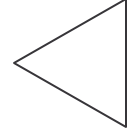
- Think of V as having an “extra” unit of blackness. This extra blackness must be absorbed into the tree (by a red node), or propagated up to the root and out of the tree.
- There are four cases – our examples and “rules” assume that V is a left child. There are symmetric cases for V as a right child.

Terminology

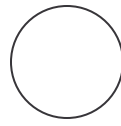
- The node just deleted was U
- The node that replaces it is V, which has an extra unit of blackness
- The parent of V is P
- The sibling of V is S



Black Node



Red or Black and don't care



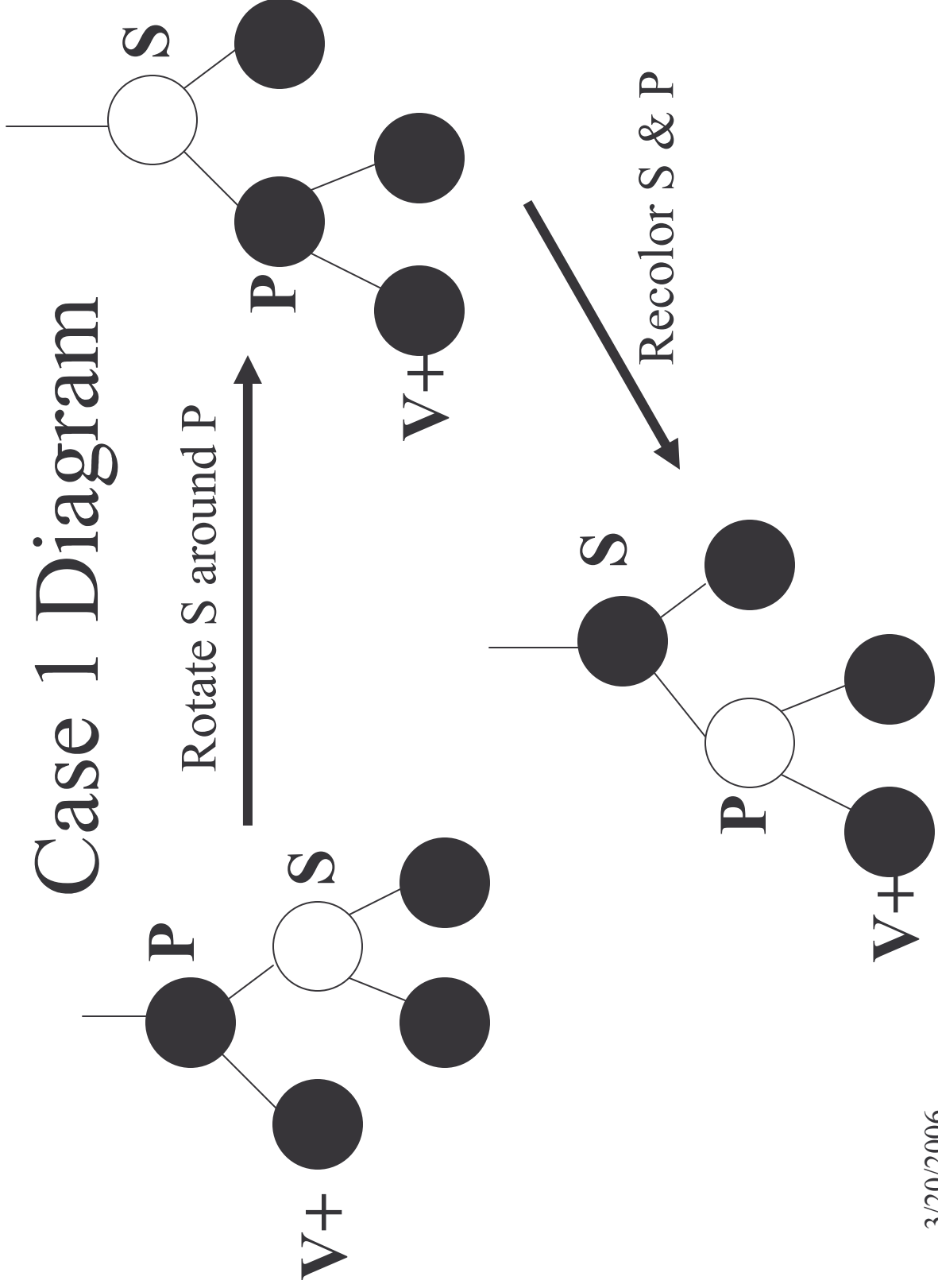
Red Node

Bottom-Up Deletion

Case 1

- V's sibling, S, is Red
 - Rotate S around P and recolor S & P
- NOT a terminal case – One of the other cases will now apply
- All other cases apply when S is Black

Case 1 Diagram

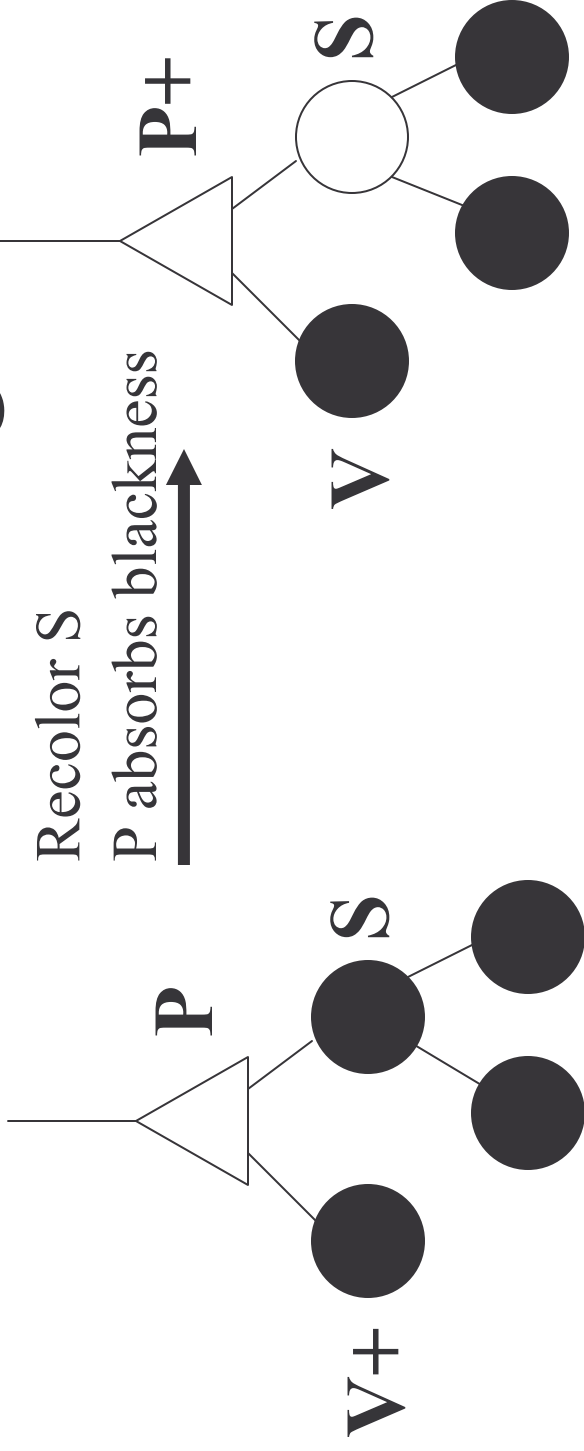


Bottom-Up Deletion

Case 2

- V's sibling, S, is Black and has two Black children.
 - Recolor S to be Red
 - P absorbs V's extra blackness
 - If P is Red, we're done (it absorbed the blackness)
 - If P is Black, it now has extra blackness and problem has been propagated up the tree

Case 2 diagram



Either extra Black absorbed by P

or

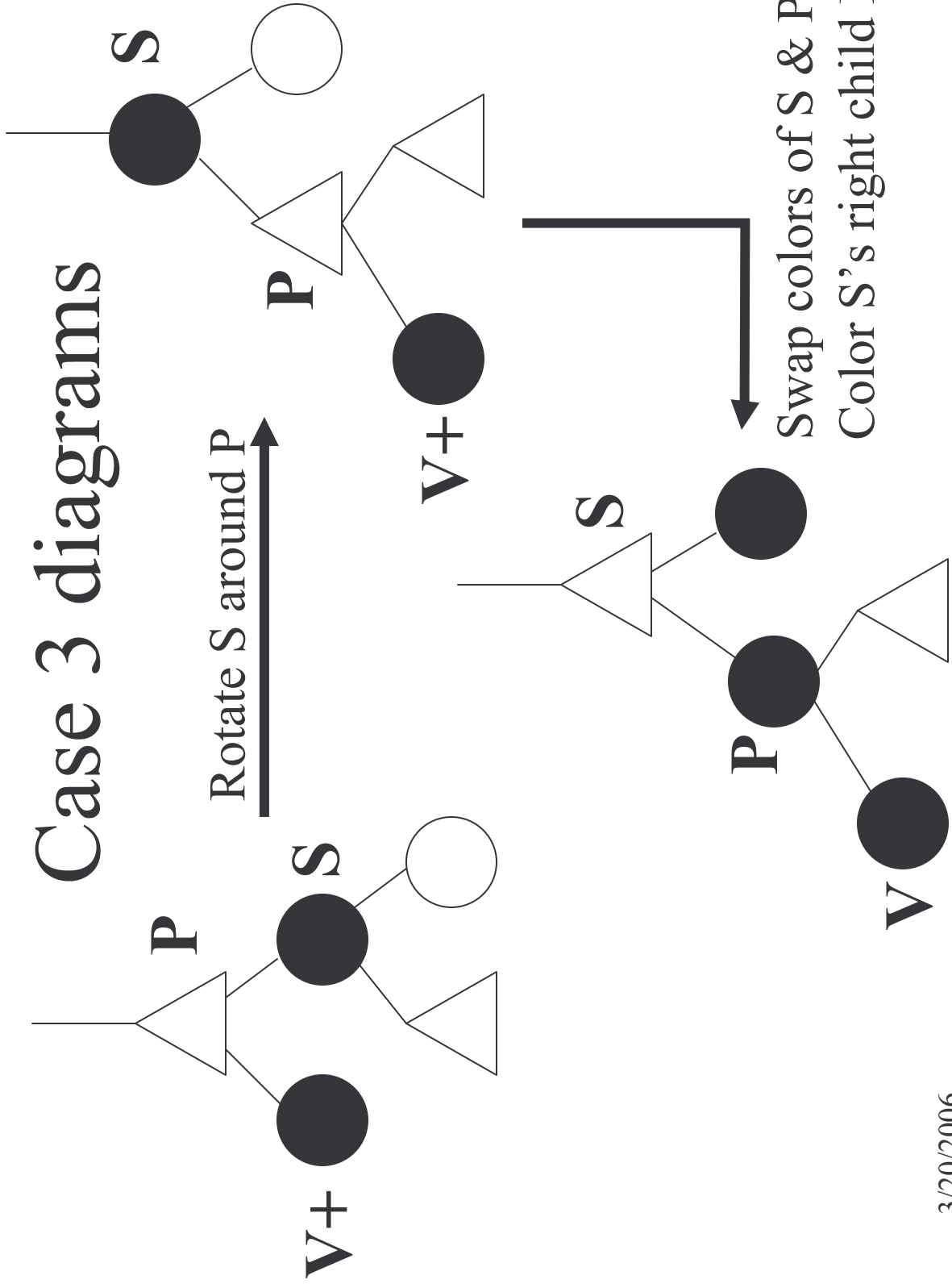
P now has extra blackness

Bottom-Up Deletion

Case 3

- S is Black
- S's right child is RED (Left child either color)
 - Rotate S around P
 - Swap colors of S and P,
and color S's right child Black
- This is the terminal case – we're done

Case 3 diagrams

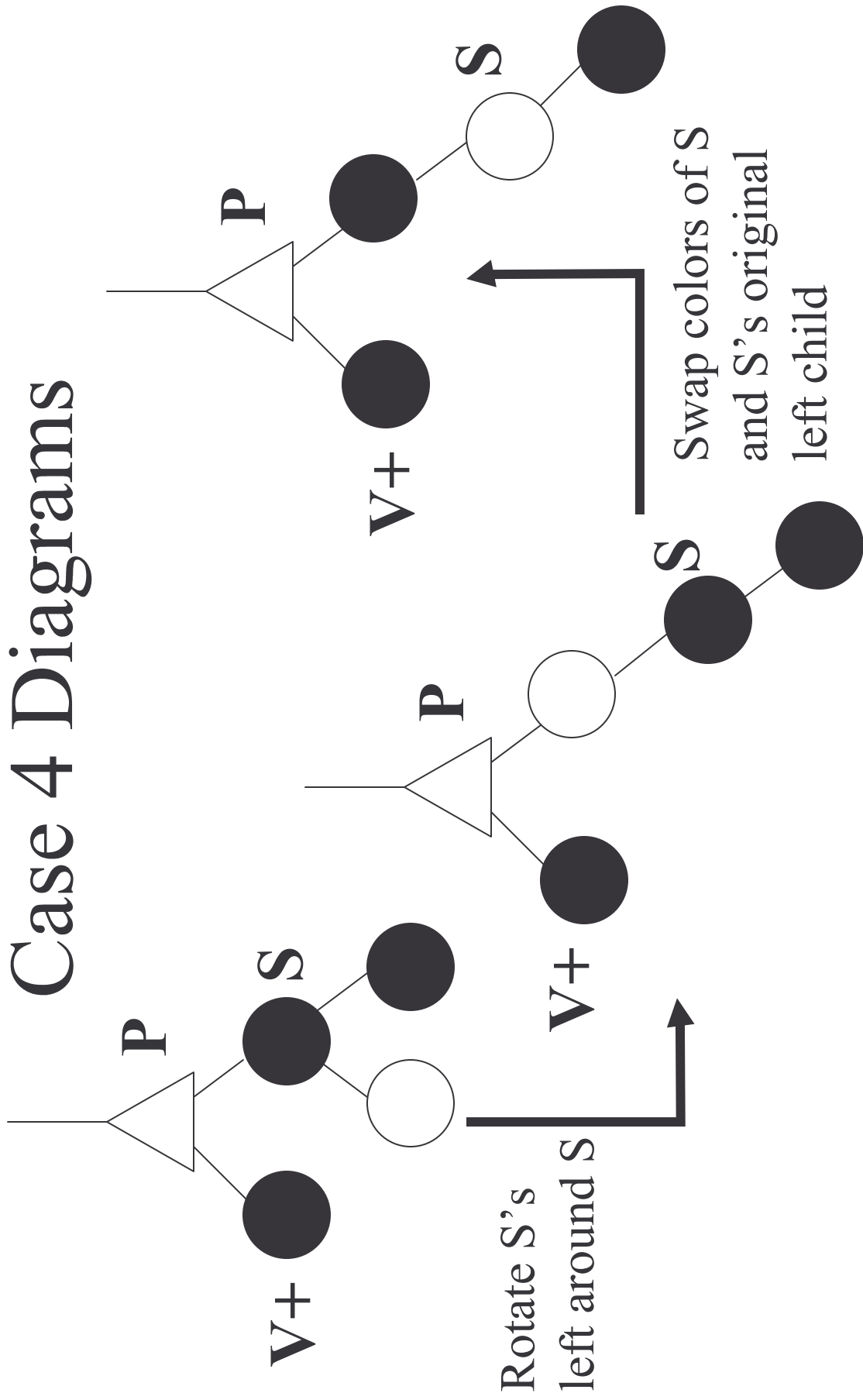


Bottom-Up Deletion

Case 4

- S is Black, S's right child is Black and S's left child is Red
 - Rotate S's left child around S
 - Swap color of S and S's left child
 - Now in case 3

Case 4 Diagrams

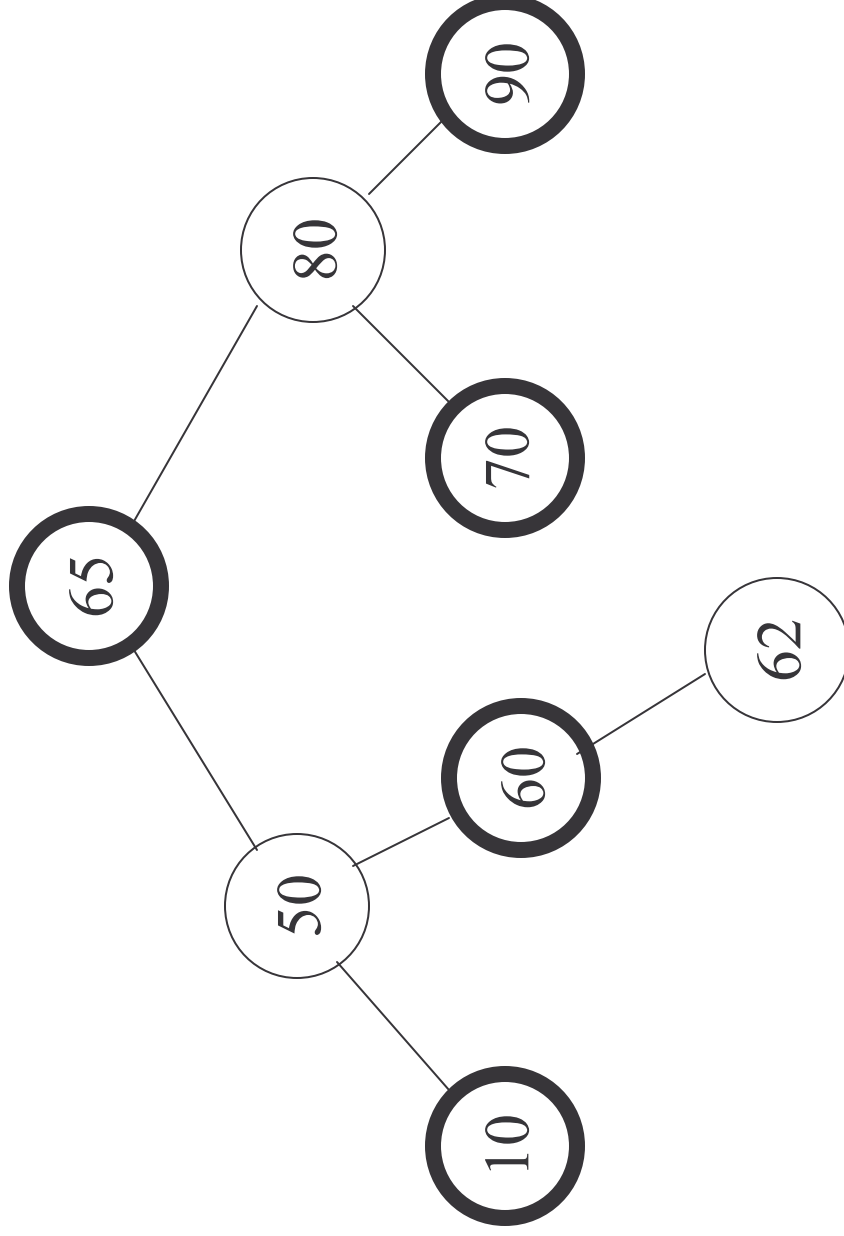


Top-Down Deletion

An alternative to the recursive “bottom-up” deletion is “top-down” deletion.

This method is iterative. It moves down the tree only, “fixing” things as it goes.

What is the goal of top-down deletion?



Perform the following deletions, in the order specified
Delete 90, Delete 80, Delete 70