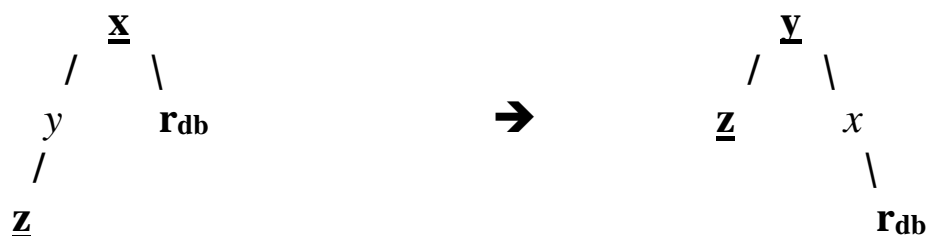


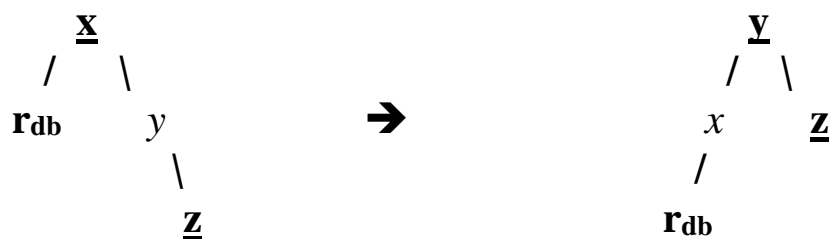
## Red-Black Tree Deletion (Case 3)

This is the last case where we have to deal with a double black node  $r$ . In this situation,  $y$ , the sibling of  $r$ , is red. If  $y$  is a right child of  $x$  (where  $x$  is  $r$ 's parent), let  $z$  be the right child of  $y$ . Otherwise, let  $z$  be the left child of  $y$ . (Note that both  $x$  and  $z$  must be black.) Perform a restructuring on the node  $z$ , placing  $y$  where  $x$  used to be:



Since  $y$ 's right child in the original picture **MUST BE** black,  $x$ 's new left child **MUST BE** black as well. This puts us in either case 1 or case 2 to deal with the "double black" node.

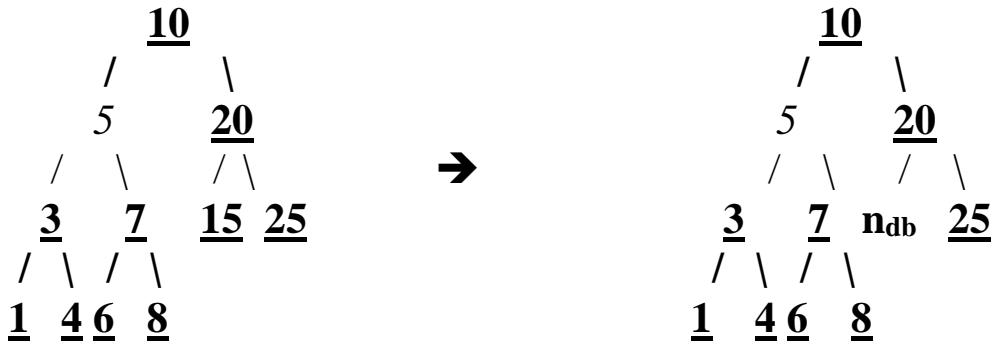
Here is the symmetric situation:



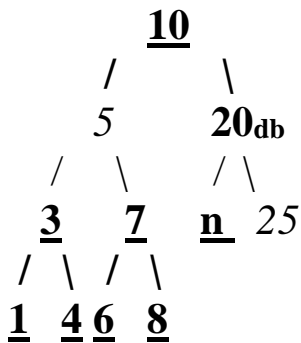
Here we have that  $y$ 's original left child must have been black, thus, in the restructured picture,  $x$ 's new right child must be black as well, putting us in either case 1 or case 2 as desired.

In both of these situations, since x is red, we will NOT propagate the "double black" node. Instead it will be handled in one operation of either case 1 or case 3.

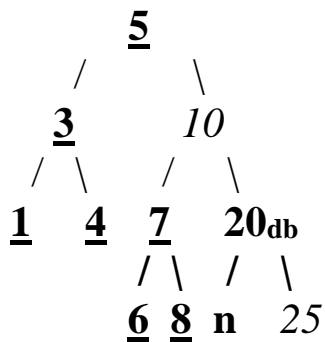
Consider deleting 15 from the Red-Black Tree below:



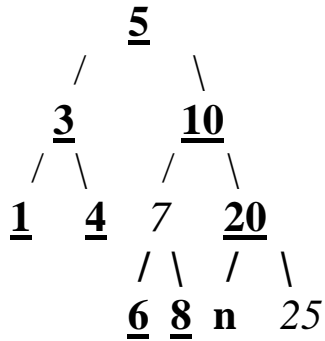
This is actually case 2. Here we will recolor as follows:



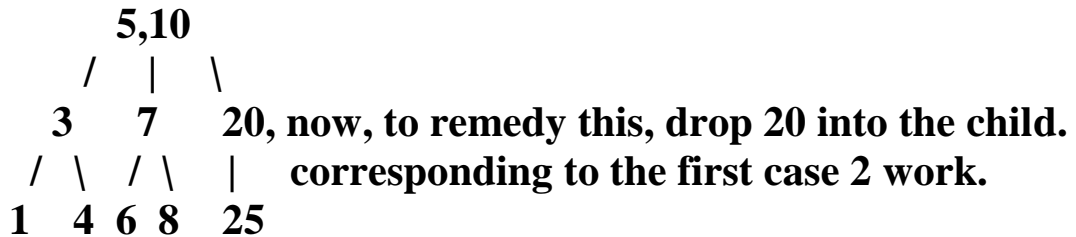
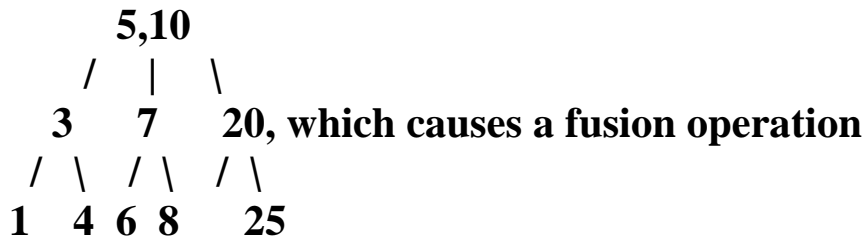
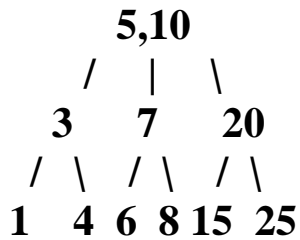
Now we have a situation where we are in case 3. Here we will have x=10, y=5, and z=3.

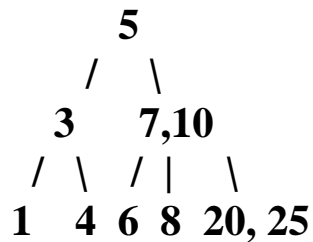
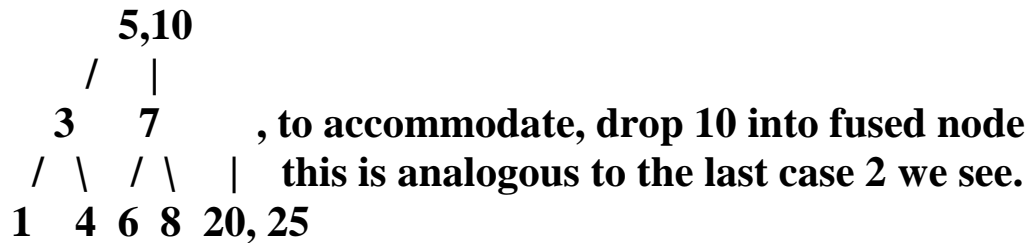
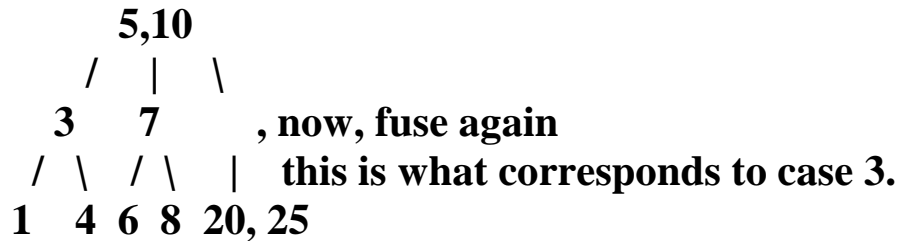


Now, we are in case 2, since both of 7's children are black. Deal with this case with a recoloring:



Consider the corresponding delete in a 2-4 Tree:





Notice that case three corresponds to the case when we have to drop a value from a parent node into a child after a fusion, where the parent node DOES have a value to spare. This is why after this occurs, only a single application of case 1 or case 2 is necessary. The difference in case corresponds to a difference in which value from the parent node will end up dropping into the fused node.

## Summarizing Red-Black Tree Delete

We have a couple simple cases to deal with, which we can do without any extra work. These correspond to removing a value from a 2-4 tree where there are other values at the node the removal is made from.

The rest of the cases result in a "double black" colored node. These cases all correspond to restructuring operations in the 2-4 Tree, such as fusions, and dropping elements into a node. The goal is to deal with the double black(DB) node to get rid of this property. Here is the outline of these cases:

### Case 1: DB node has black sibling with at least one red child.

This fixes the problem structurally. No extra work is required after this case completes. This corresponds to a transfer operation in a 2-4 Tree.

### Case 2: DB node has black sibling with two black children.

This uses a recoloring and no structural change. It may solve the problem, but may ALSO propagate the DB node to the parent of the current DB node. This corresponds to a fusion and drop operation in a 2-4 Tree.

### Case 3: DB Node has red sibling

A structural change here puts you in case 1 or case 2. At this point, a single application of either case is sufficient. This corresponds to a fusion where you have enough values in the parent node to drop one into the fused child.