CSE 2320 Notes 10: Red-Black Trees

(Last updated 10/7/06 6:59 PM)

CLRS, 13.1-13.4, along with the test 2 cheat sheet

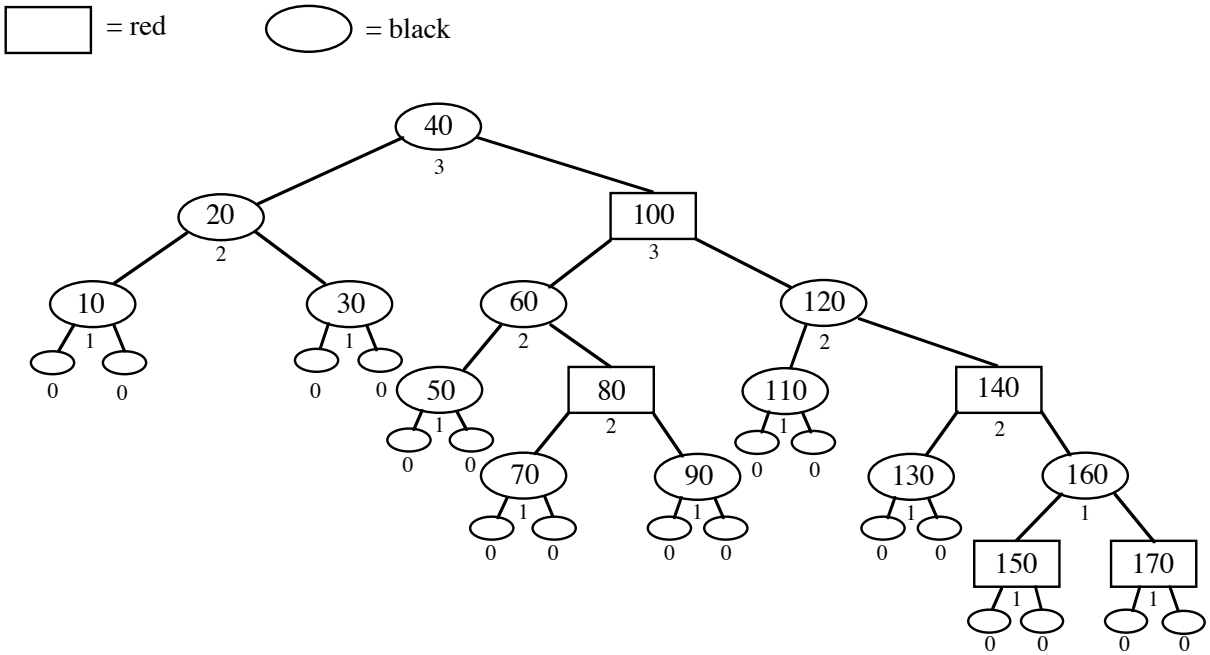
STRUCTURAL PROPERTIES

A *red-black tree* is a binary search tree whose height is logarithmic in the number of keys stored.

1. Every node is colored red or black. (Colors are only examined during insertion and deletion)
2. Every "leaf" (the sentinel) is colored black.
3. Both children of a red node are black.
4. Every simple path from a child of node X to a leaf has the same number of black nodes.

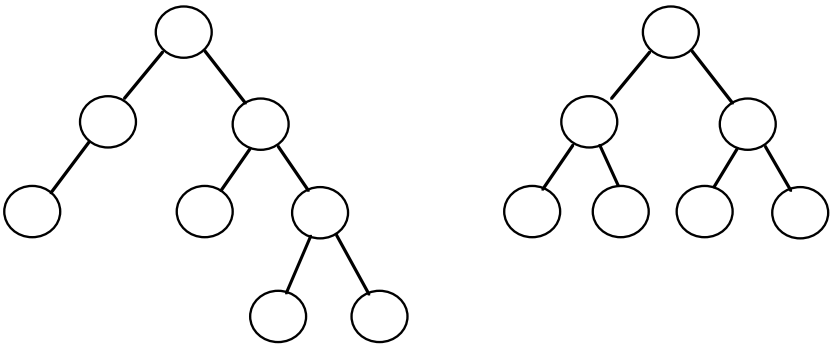
   This number is known as the *black-height* of X (bh(X)). These are not stored.

Example:



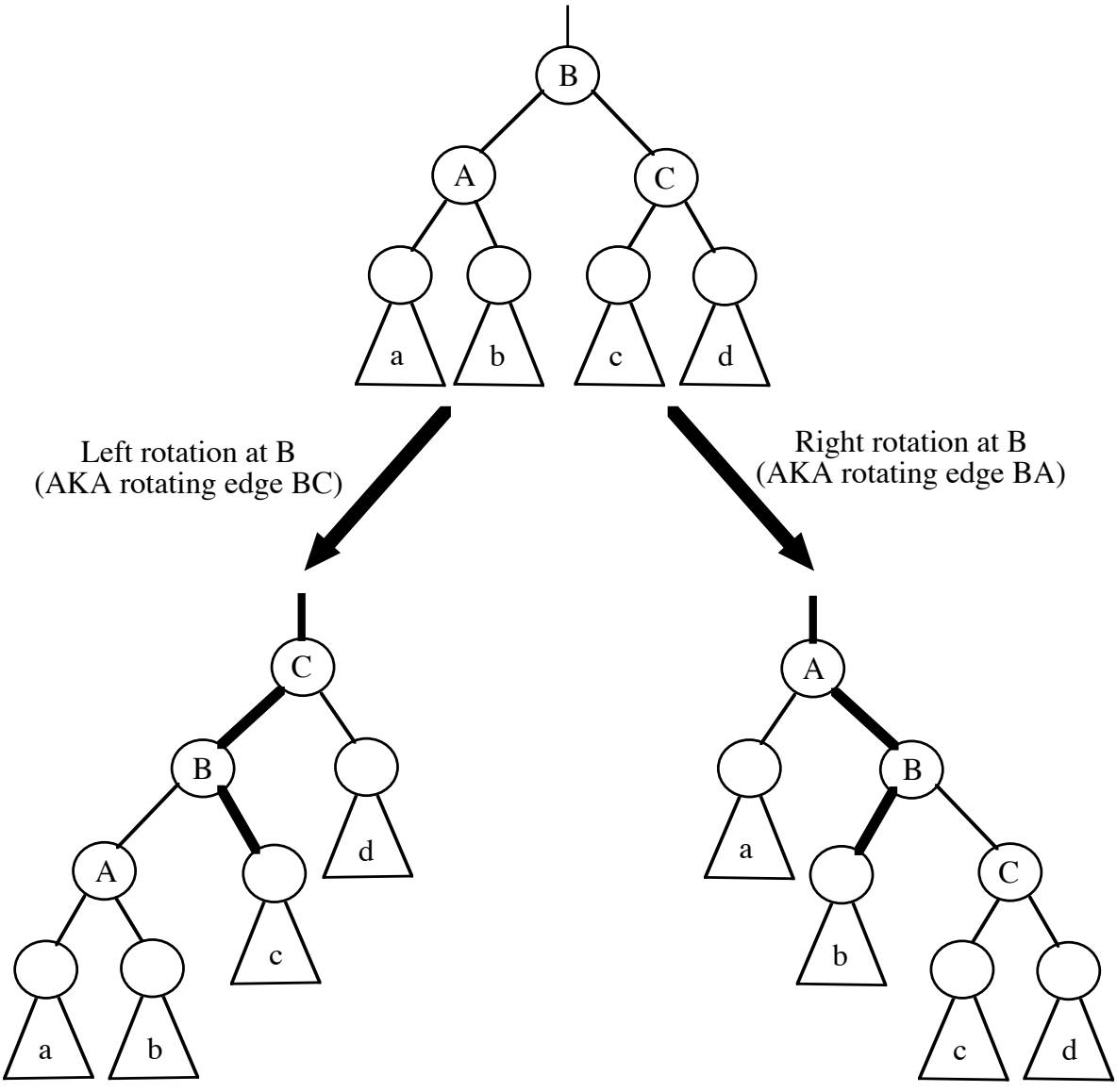
Observations:

1. A red-black tree with n internal nodes ("keys") has height at most $2 \lg(n+1)$.
2. If a node X is not a leaf and its sibling is a leaf, then X must be red.
3. There may be many ways to color a binary search tree to make it a red-black tree.
4. If the root is colored red, then it may be switched to black without violating structural properties.



ROTATIONS

Technique for rebalancing in most balanced binary search tree schemes. Takes $\Theta(1)$ time.

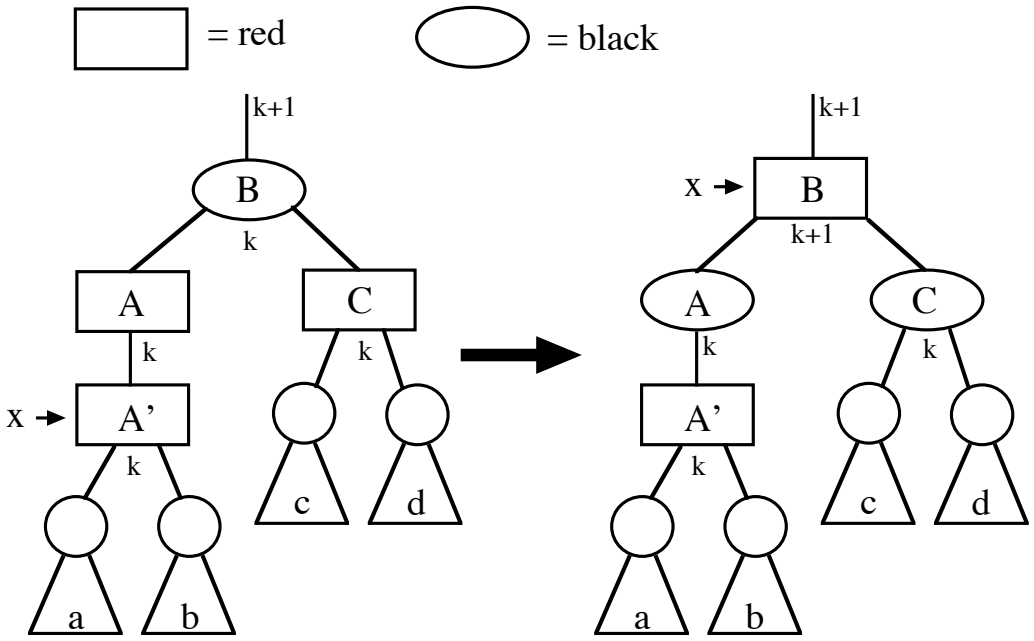


INSERTION

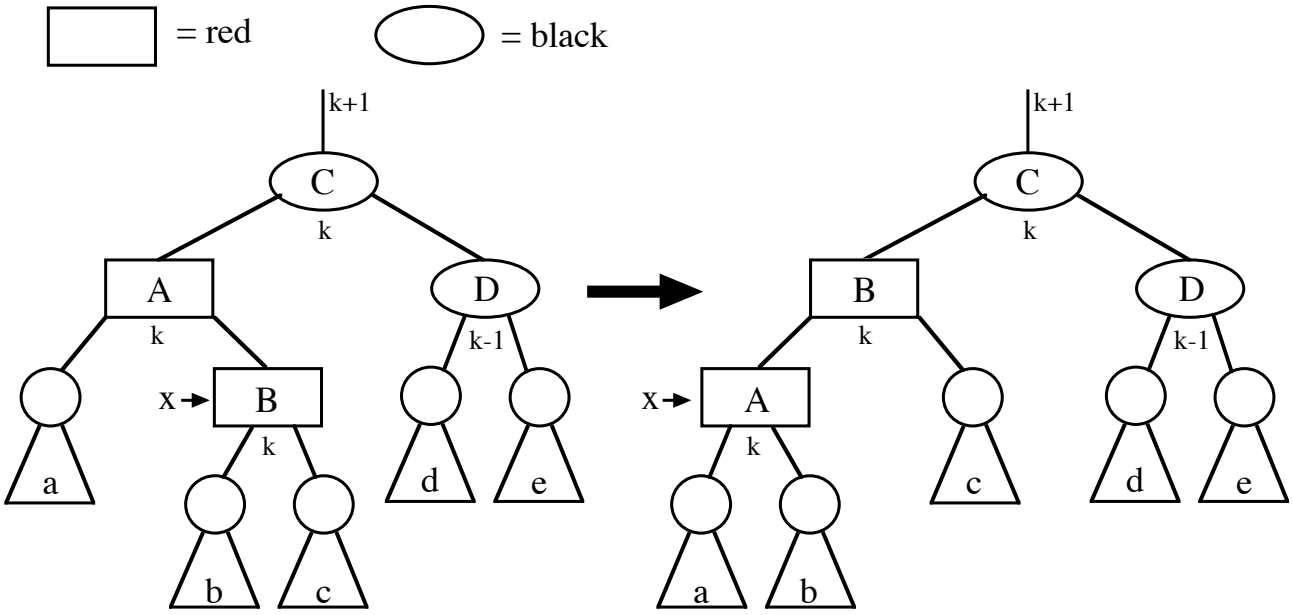
1. Start with unbalanced insert of a "data leaf" (both children are the sentinel).
2. Color of new node is _____.
3. May violate structural property 3. Leads to three cases, along with symmetric versions.

   The x pointer points at a red node whose parent might also be red.

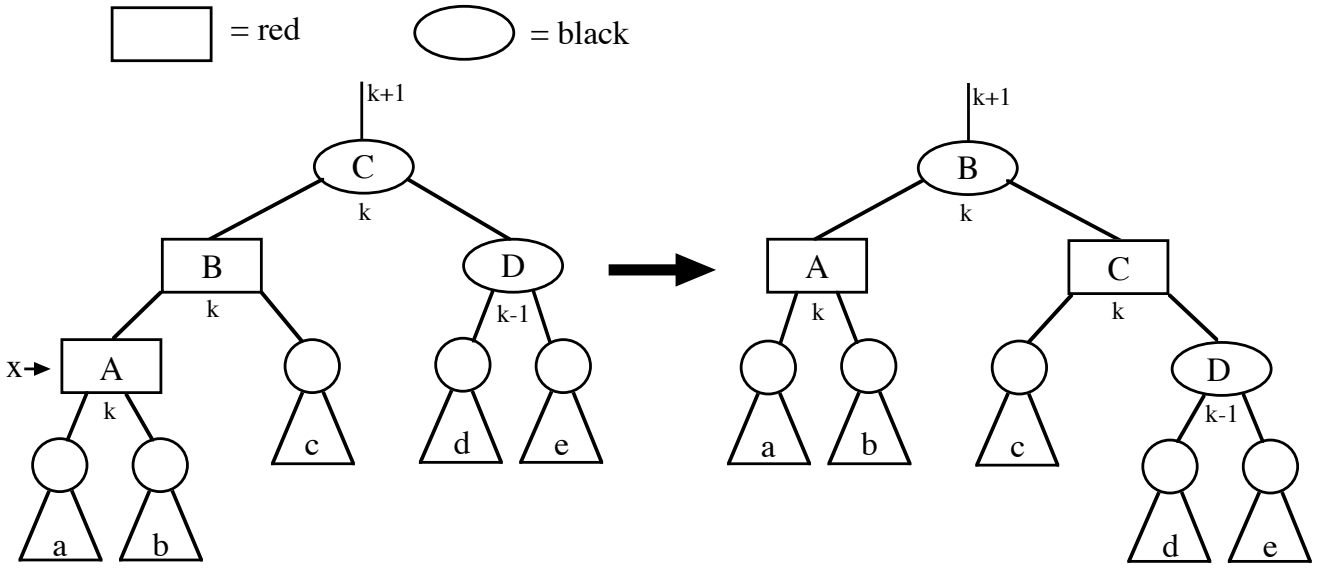
Case 1:



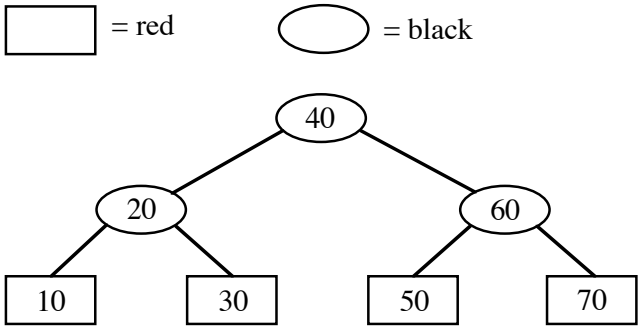
Case 2:



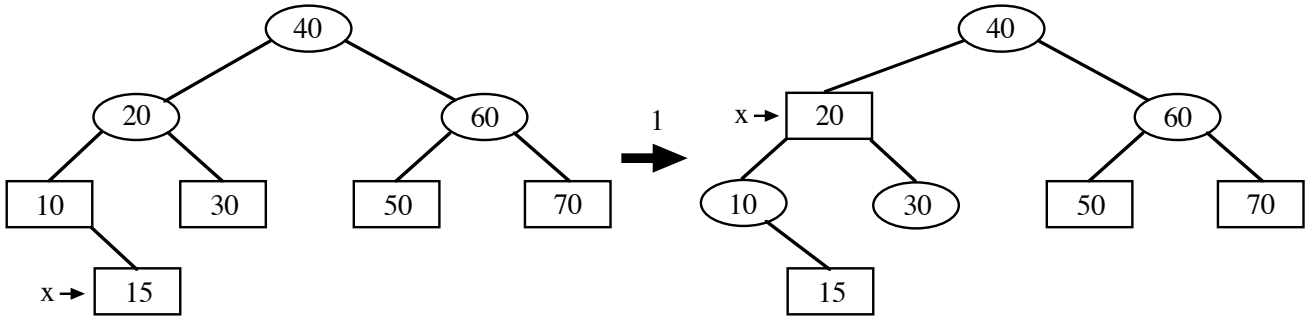
Case 3:



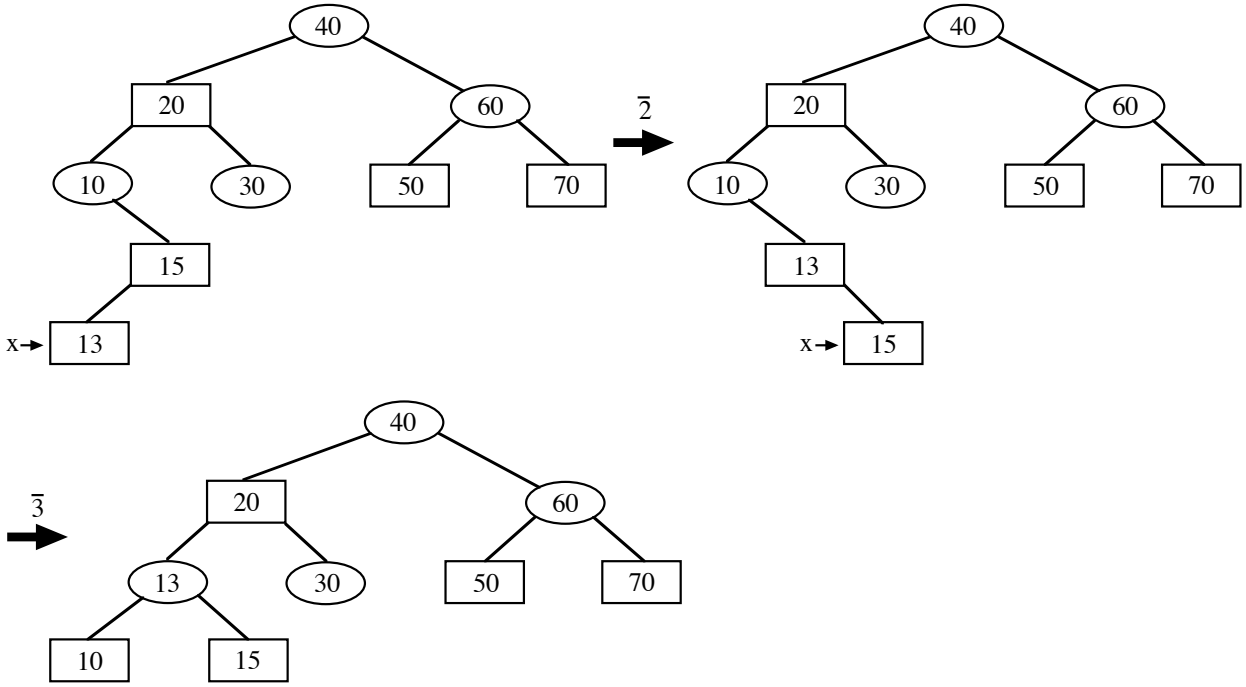
Example 1:



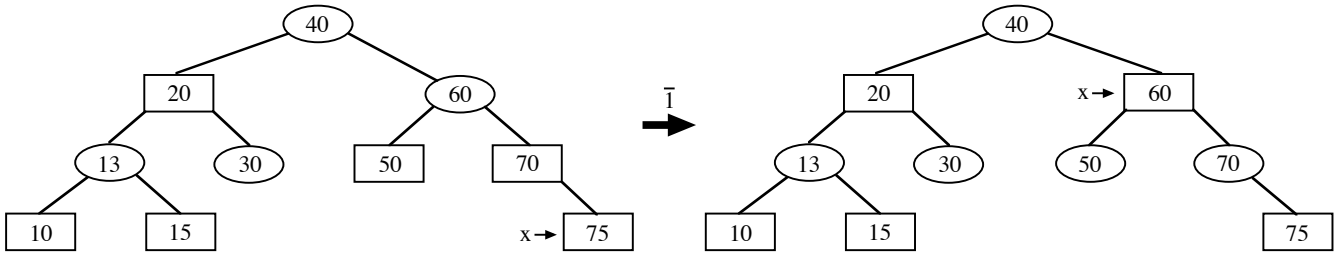
Insert 15



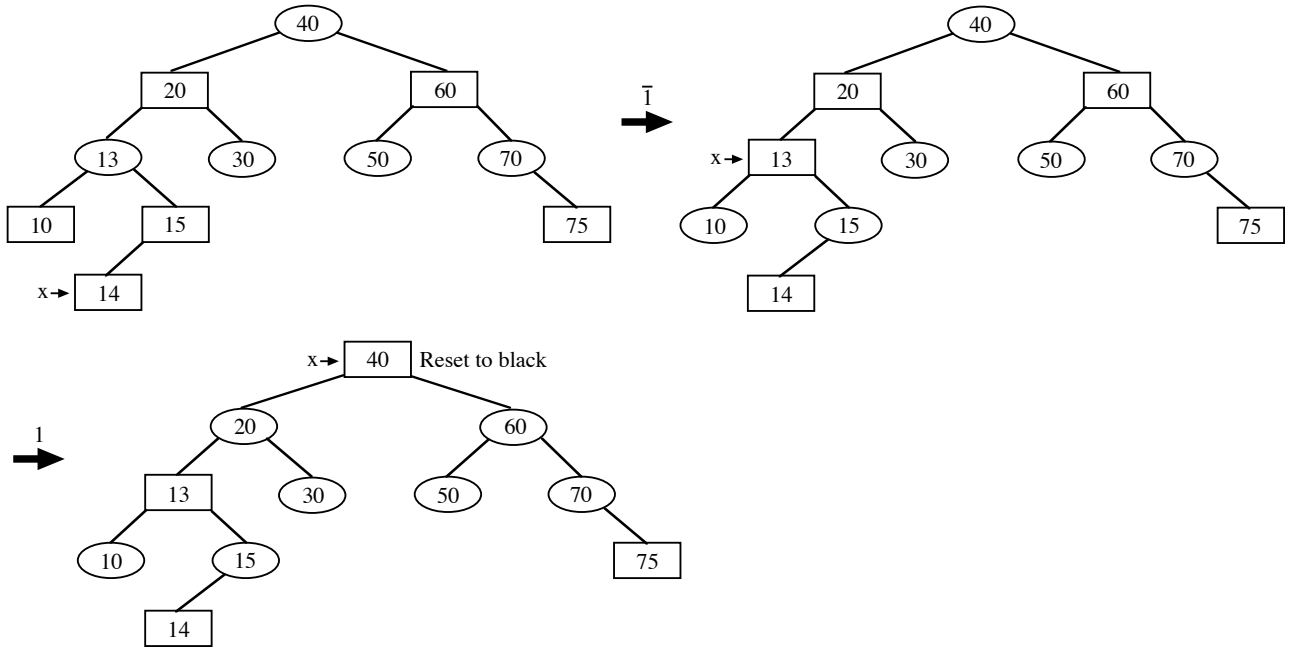
Insert 13



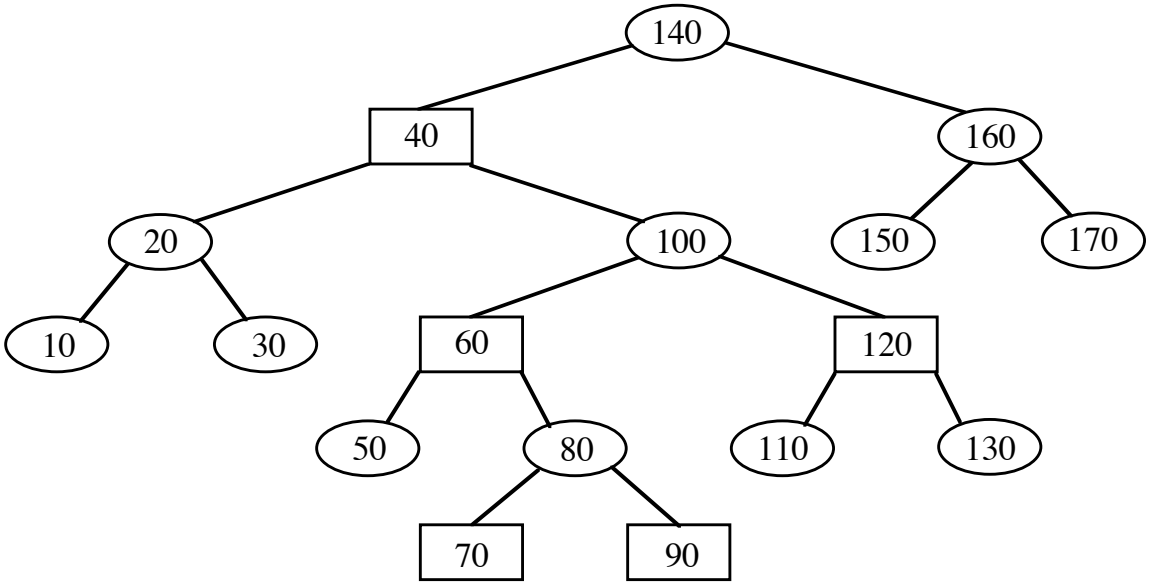
Insert 75



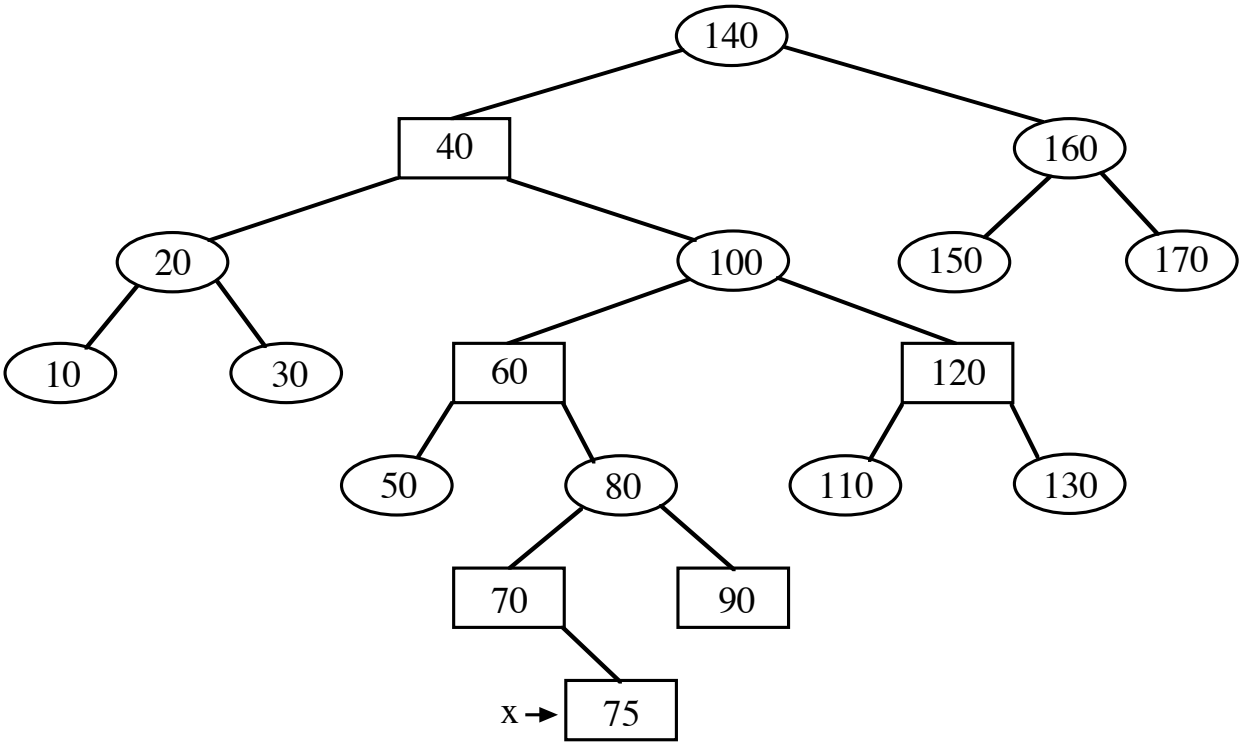
Insert 14

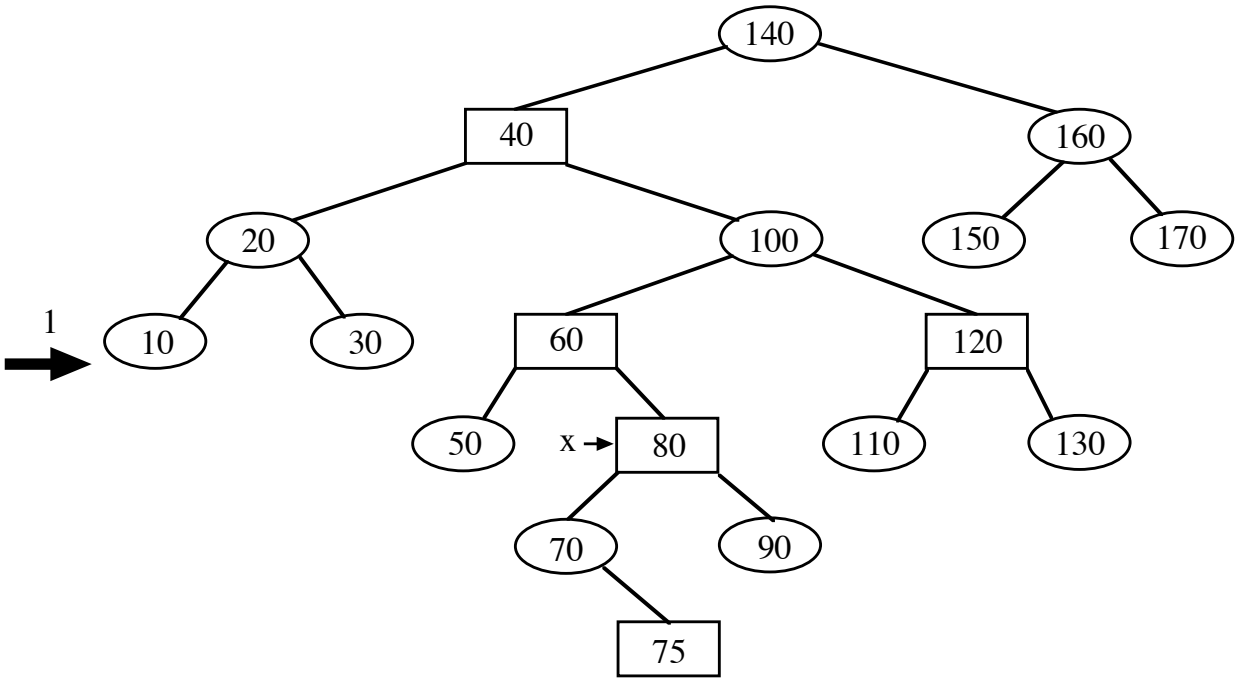


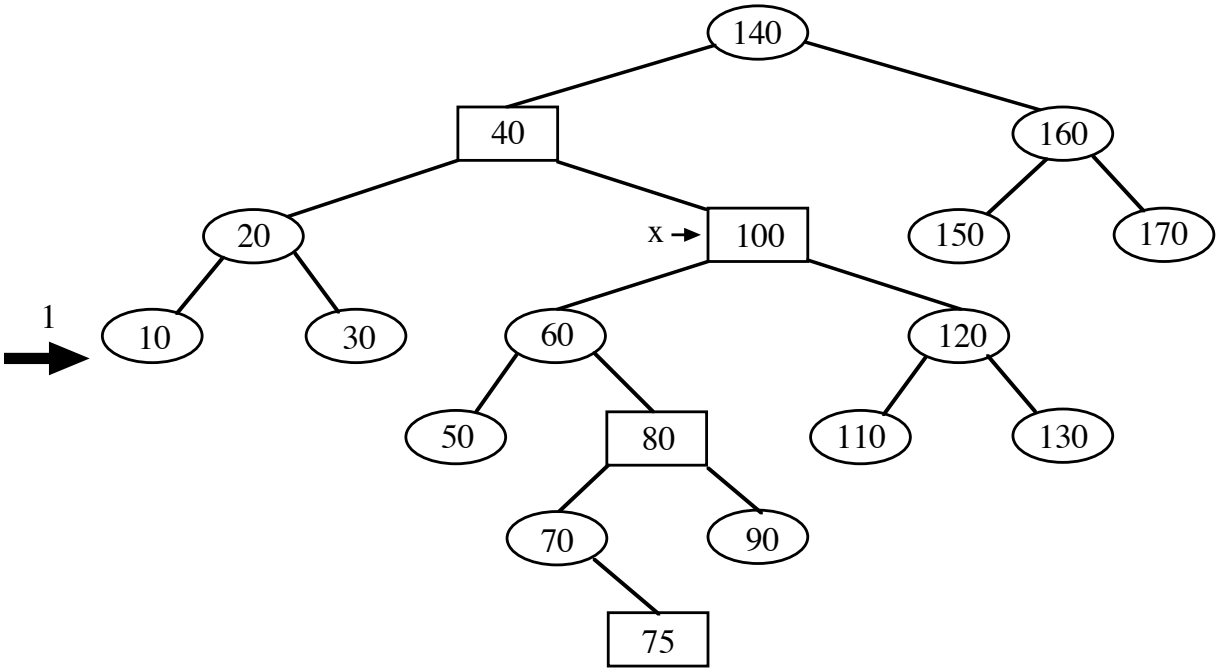
Example 2:

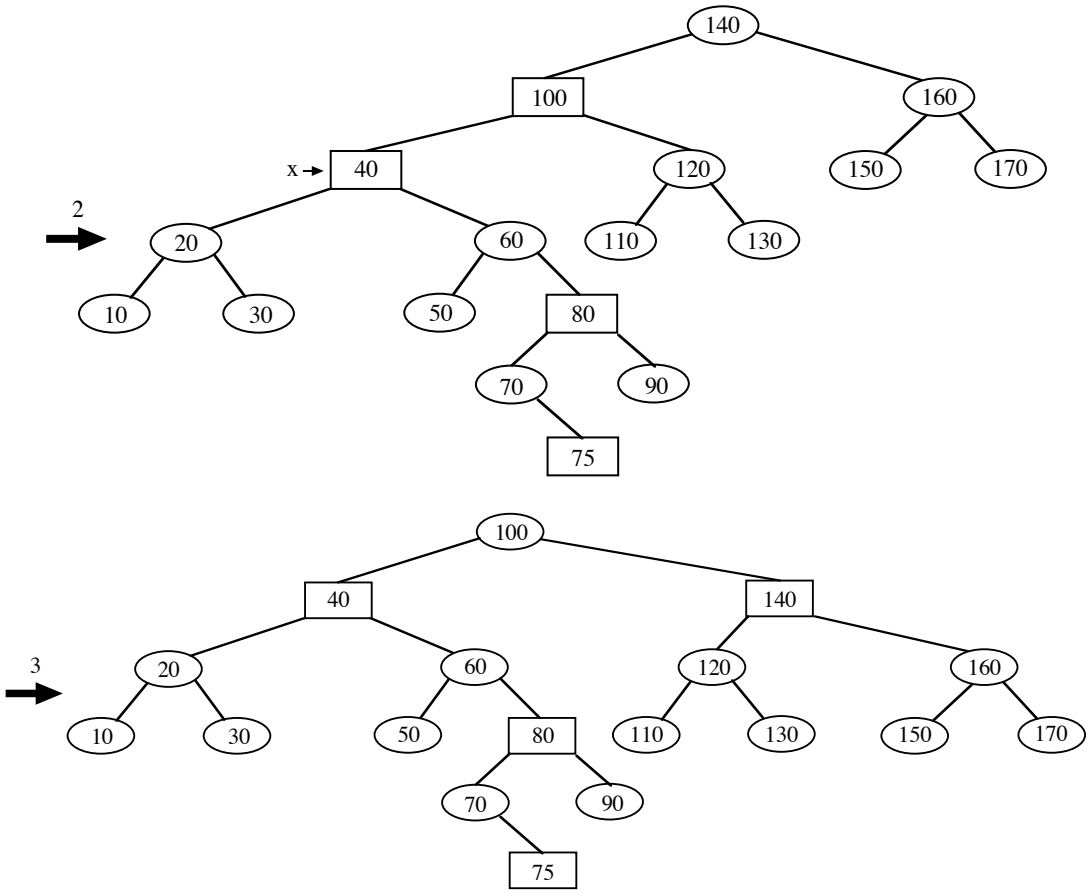


Insert 75




140
40
160
20
100
150
170
1
10
30
60
120
50
x
80
110
130
70
90
75


140
40
160
20
x
100
150
170
1
10
30
60
120
50
80
110
130
70
90
75



DELETION

Start with one of the unbalanced deletion cases:

1. Deleted node is a "data leaf".

   a. Splice around to sentinel.

   b. Color of deleted node?

      Red ⇒ Done

      Black ⇒ Set temporary "double black" pointer (x) at sentinel.
      Determine which of four rebalancing cases applies.

2. Deleted node is parent of one "data leaf".

   a. Splice around to "data leaf"

   b. Color of deleted node?

      Red ⇒ Not possible

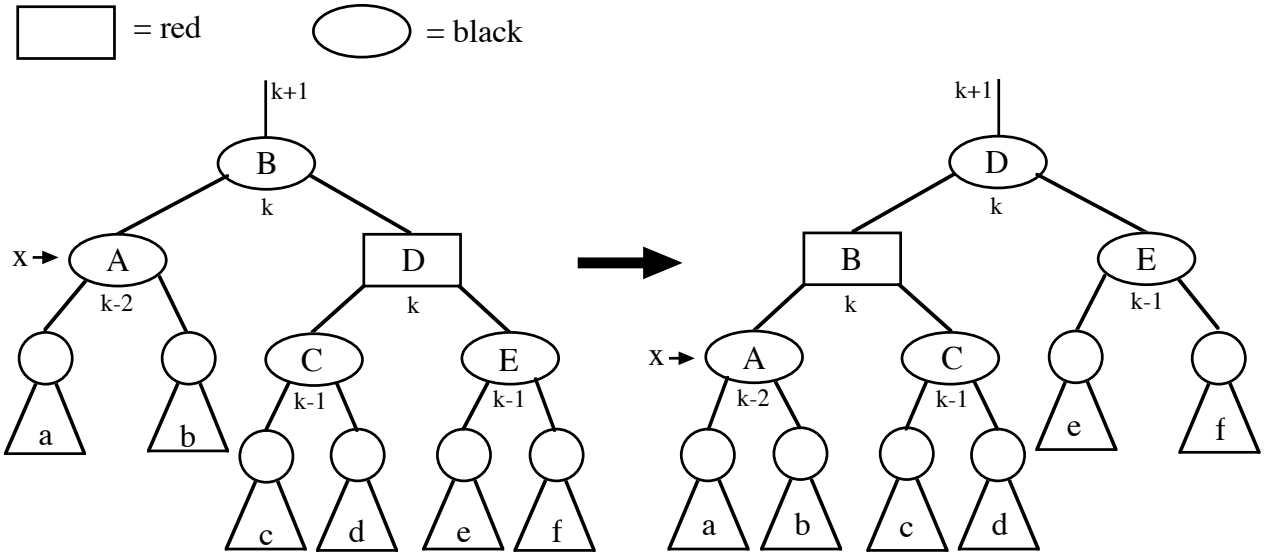
      Black ⇒ "data leaf" must be red. Change its color to black.

3. Node with key-to-delete is parent of two “data nodes”.

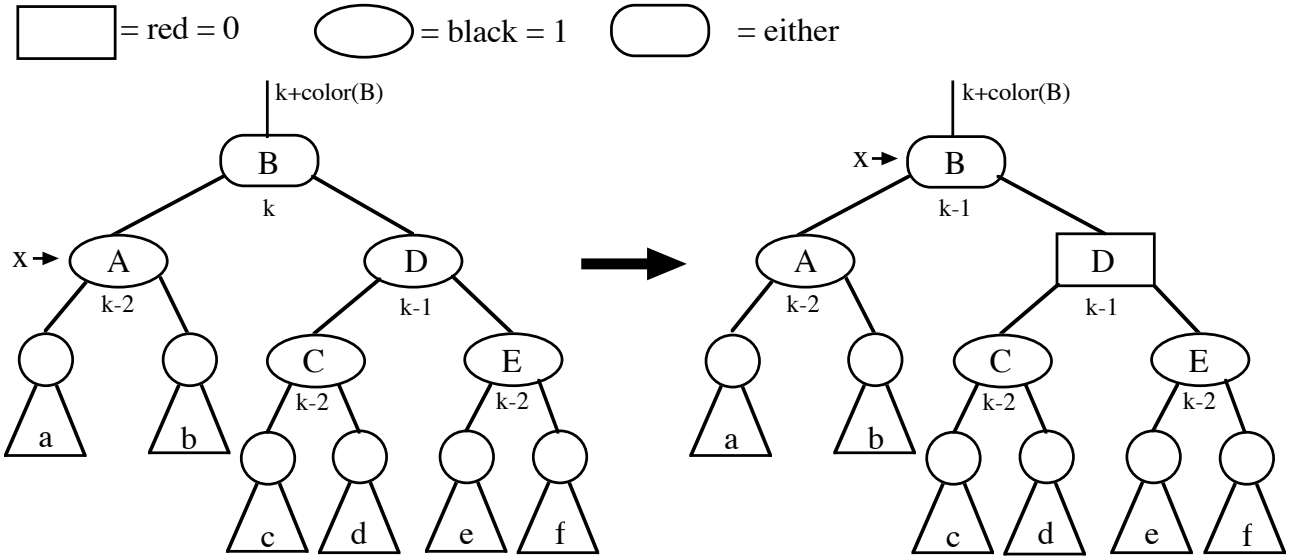
   a. Steal key and data from successor (but not the color).

   b. Delete successor using the appropriate one of the previous two cases.

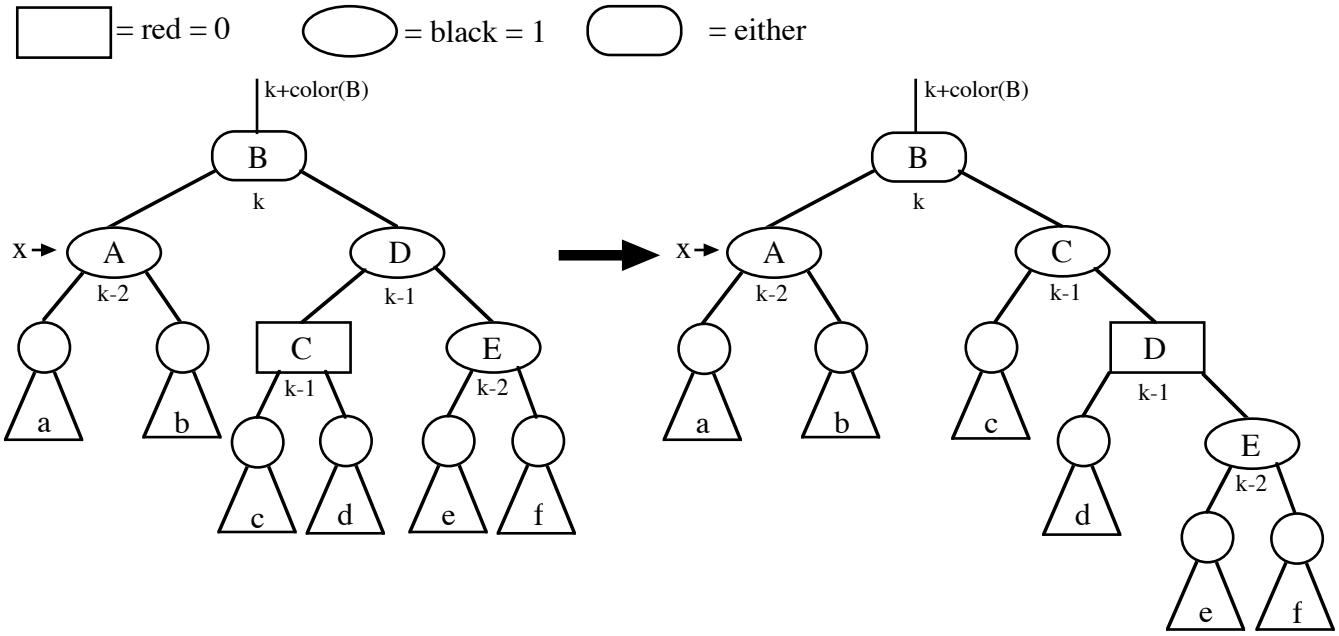
Case 1:



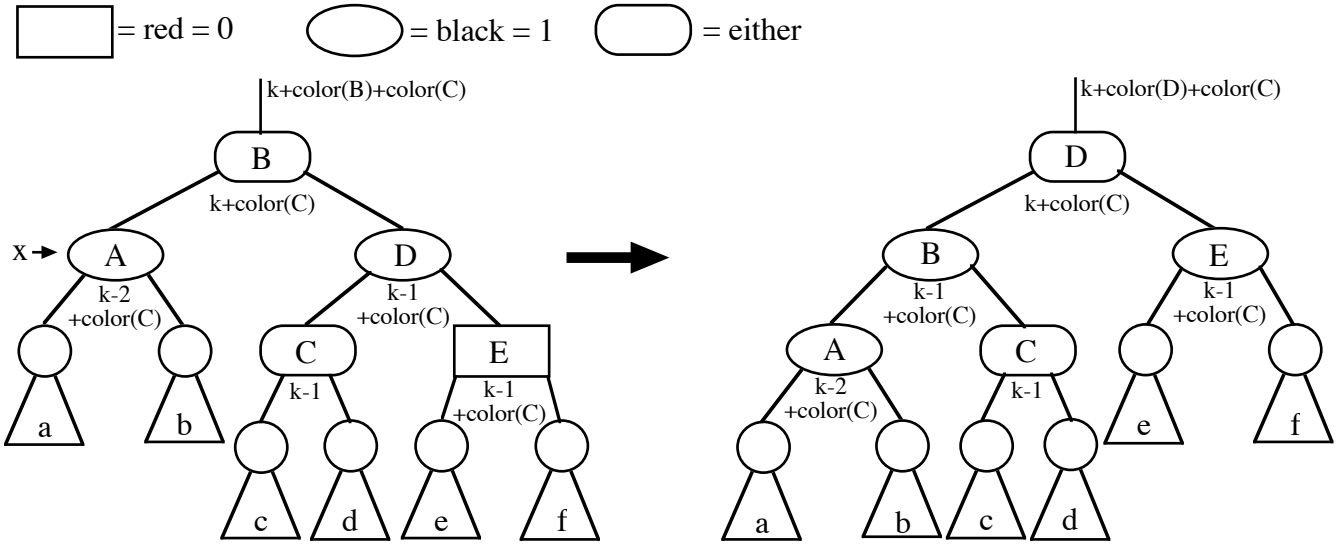
Case 2:



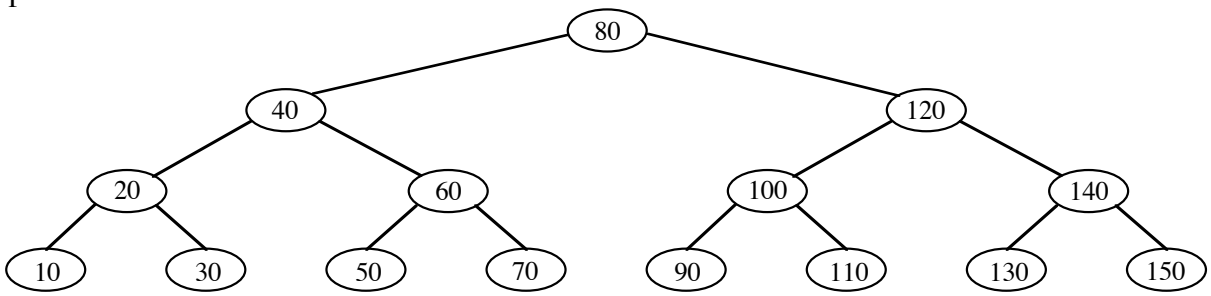
Case 3:



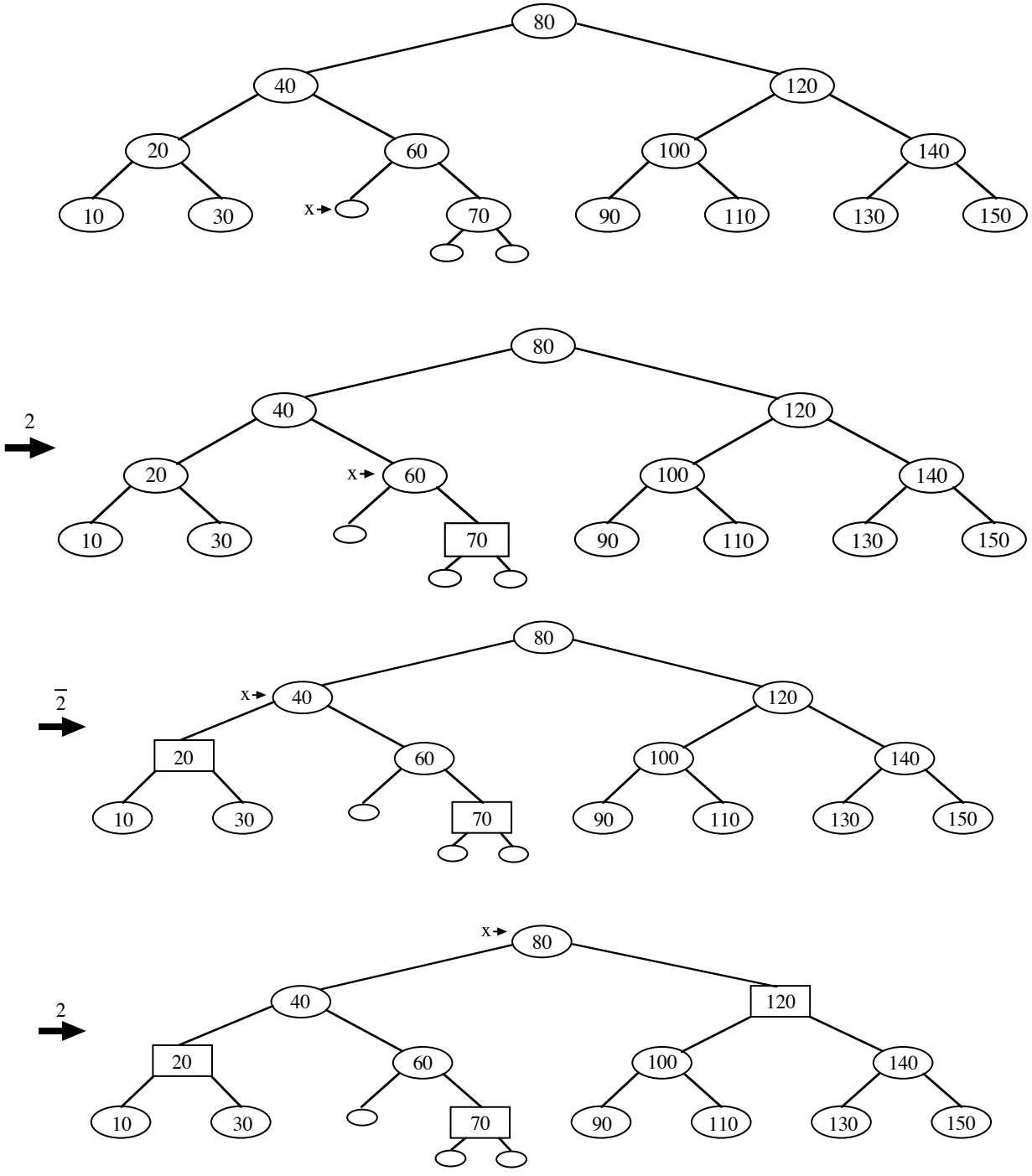
Case 4:



Example 3:

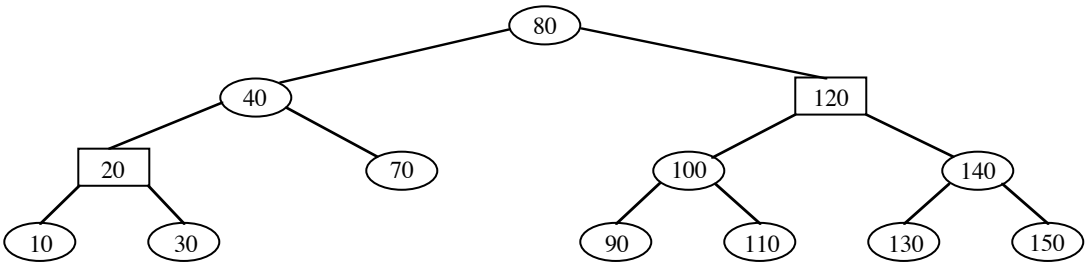


Delete 50

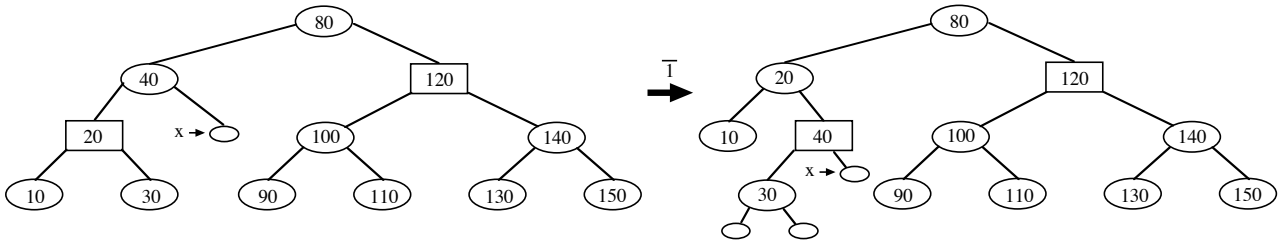


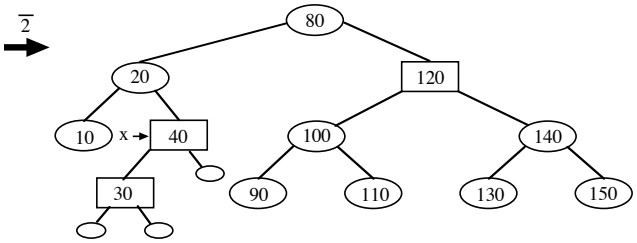
If x reaches the root, then done. Only place in tree where this happens.

Delete 60



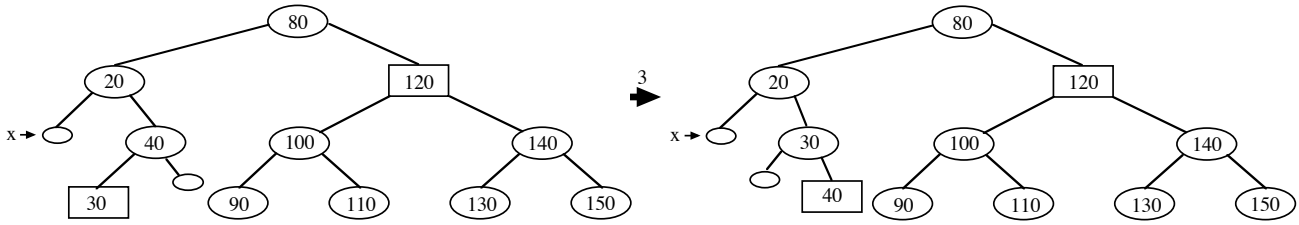
Delete 70

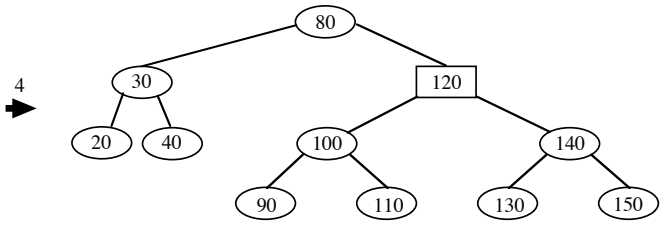




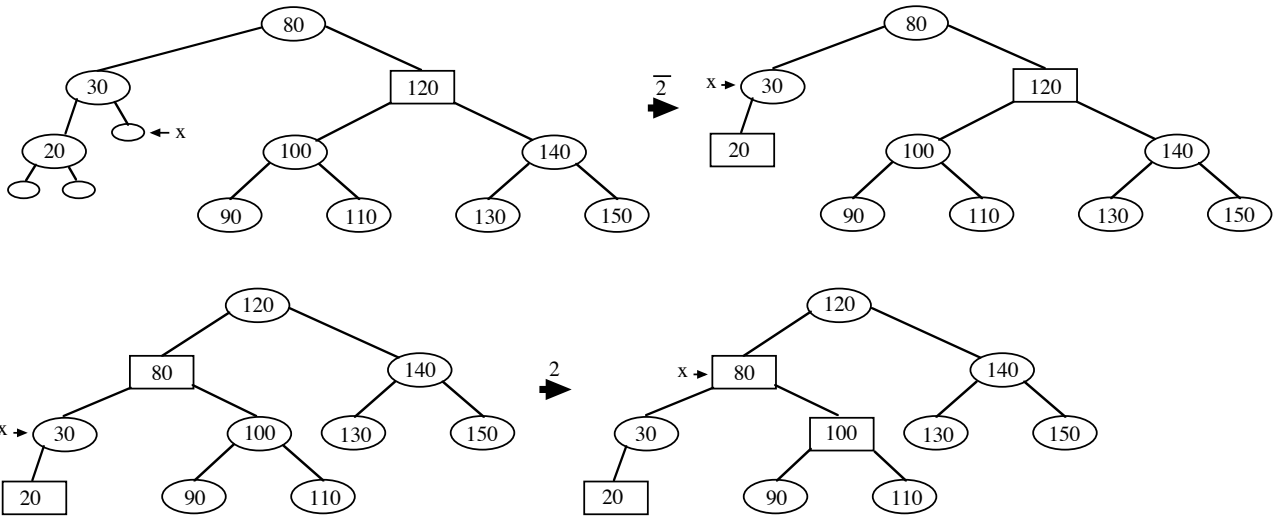
If x reaches a red node, then change color to black and done.

Delete 10

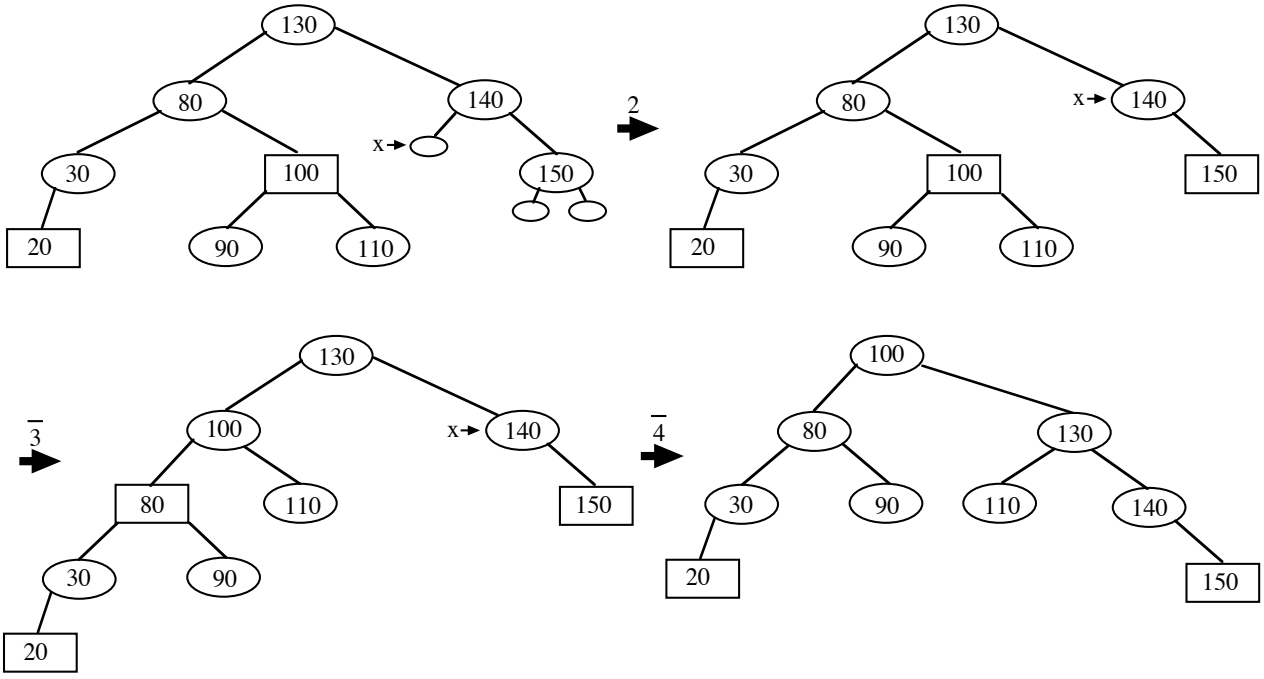




Delete 40



Delete 120



Delete 100