R-B Trees

Three Properties of a Red-Black Tree

that must always be true for the tree to be red-black

- 1. The root must always be BLACK (white in our pictures)

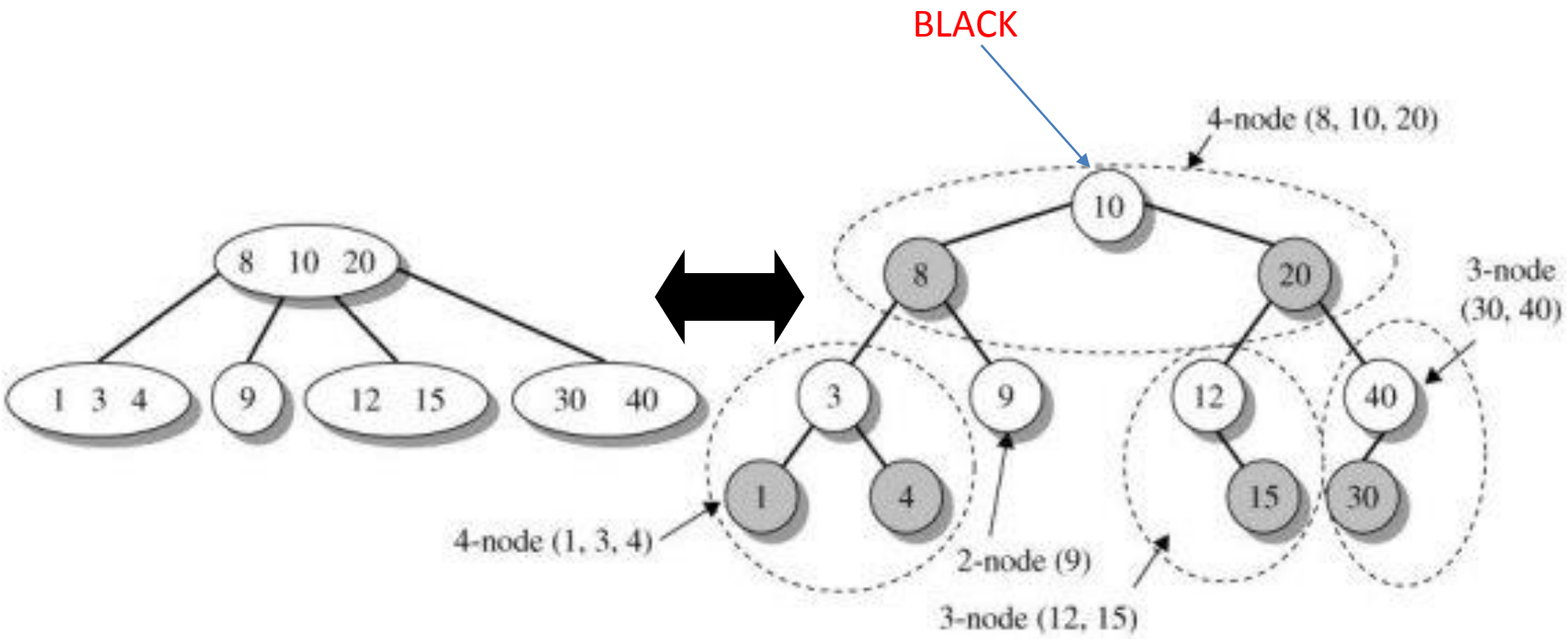
- 2. A RED parent never has a RED child
  - in other words: there are never two successive RED nodes in a path

continued

- 3. Every path from a node to an null leaf (node) contains the same number of BLACK nodes
  - called the *black height*

- We can use black height to measure the balance of a red-black tree.

Check the Example Properties



The black-height of the red-black tree is 2. Each path from the root contains exactly two BLACK nodes.

Inserting a Node

Three things to do.

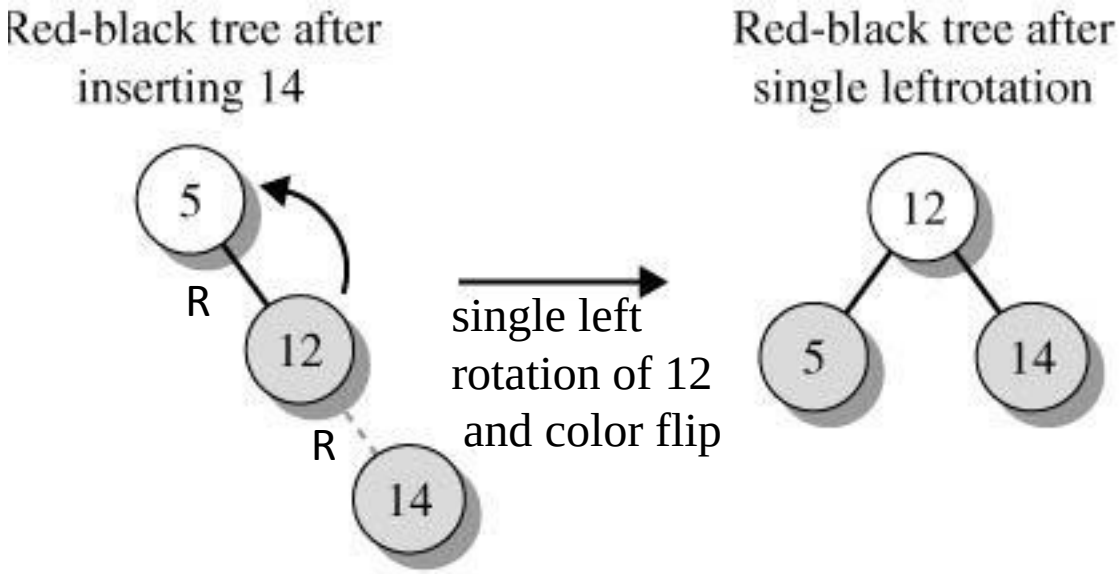
1. Search down the tree to find the insertion point
2. Once the insertion point is found, add the new item as a RED *leaf node*
   - this may create two successive RED nodes
     - again use rotation and recoloring to reorder/rebalance the tree
3. Keep the root as a BLACK node.

Inserting a New node

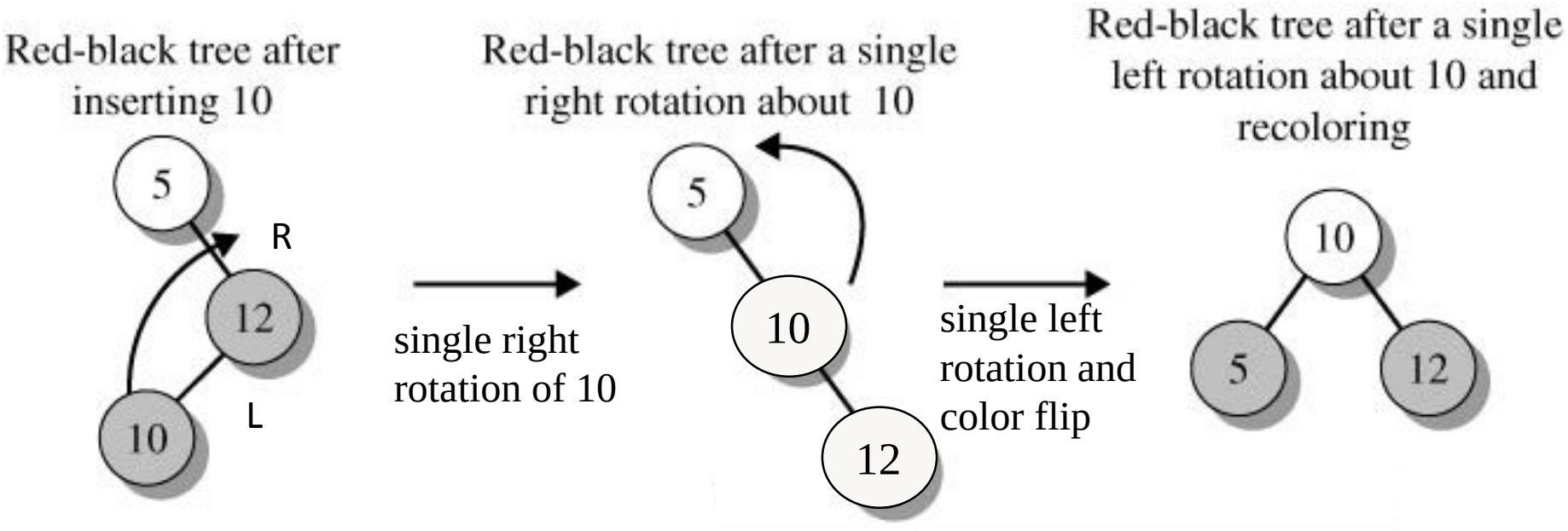
- Always add a new item to a tree as a RED leaf node
  - this may create two successive RED nodes, which breaks property 2
  - Assumption: Null is treated as Black node

```
RB-INSERT(T, x)
 1  TREE-INSERT(T, x)
 2  color[x] ← RED
 3  while x ≠ root[T] and color[p[x]] = RED
 4        do if p[x] = left[p[p[x]]]
 5              then y ← right[p[p[x]]]
 6                   if color[y] = RED
 7                      then color[p[x]] ← BLACK                    ▷ Case 1
 8                           color[y] ← BLACK                       ▷ Case 1
 9                           color[p[p[x]]] ← RED                   ▷ Case 1
10                           x ← p[p[x]]                            ▷ Case 1
11                      else if x = right[p[x]]
12                              then x ← p[x]                       ▷ Case 2
13                                   LEFT-ROTATE(T, x)              ▷ Case 2
14                           color[p[x]] ← BLACK                    ▷ Case 3
15                           color[p[p[x]] ← RED                    ▷ Case 3
16                           RIGHT-ROTATE(T, p[p[x]])               ▷ Case 3
17              else (same as then clause
                      with "right" and "left" exchanged)
18  color[root[T]] ← BLACK
```

Example: Insert 14

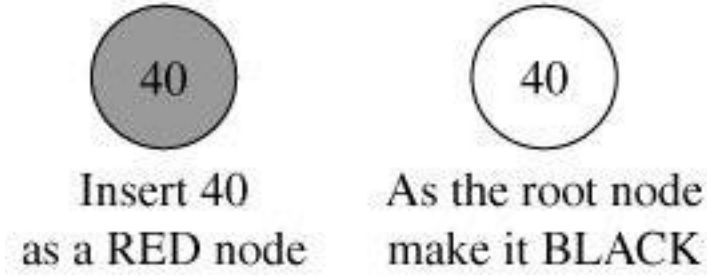


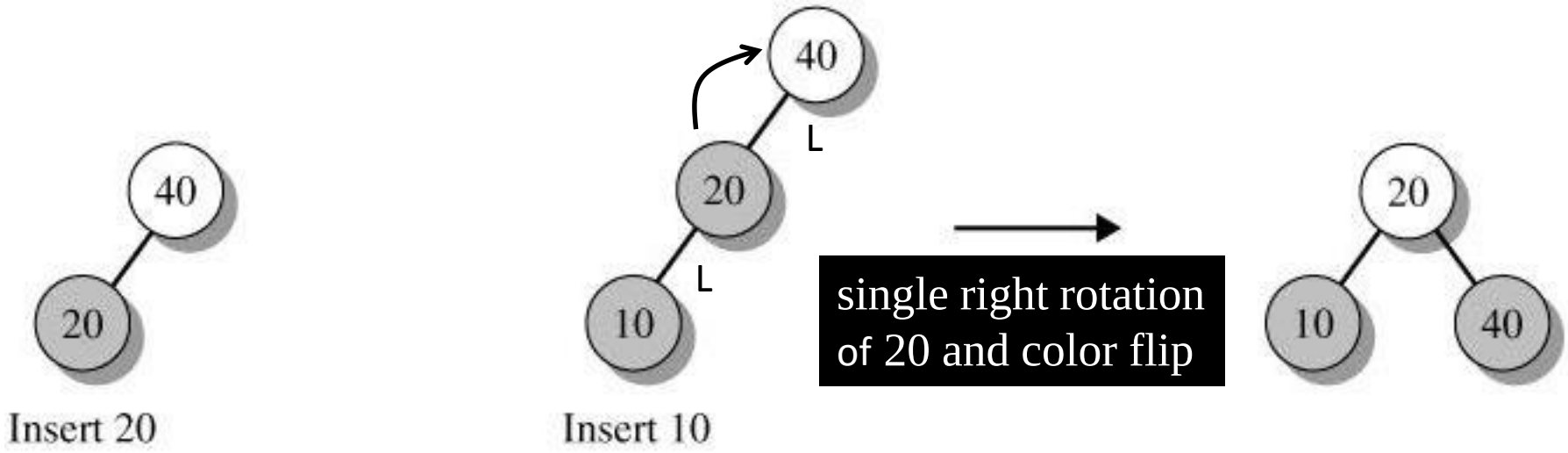
Insert 10 instead of 14



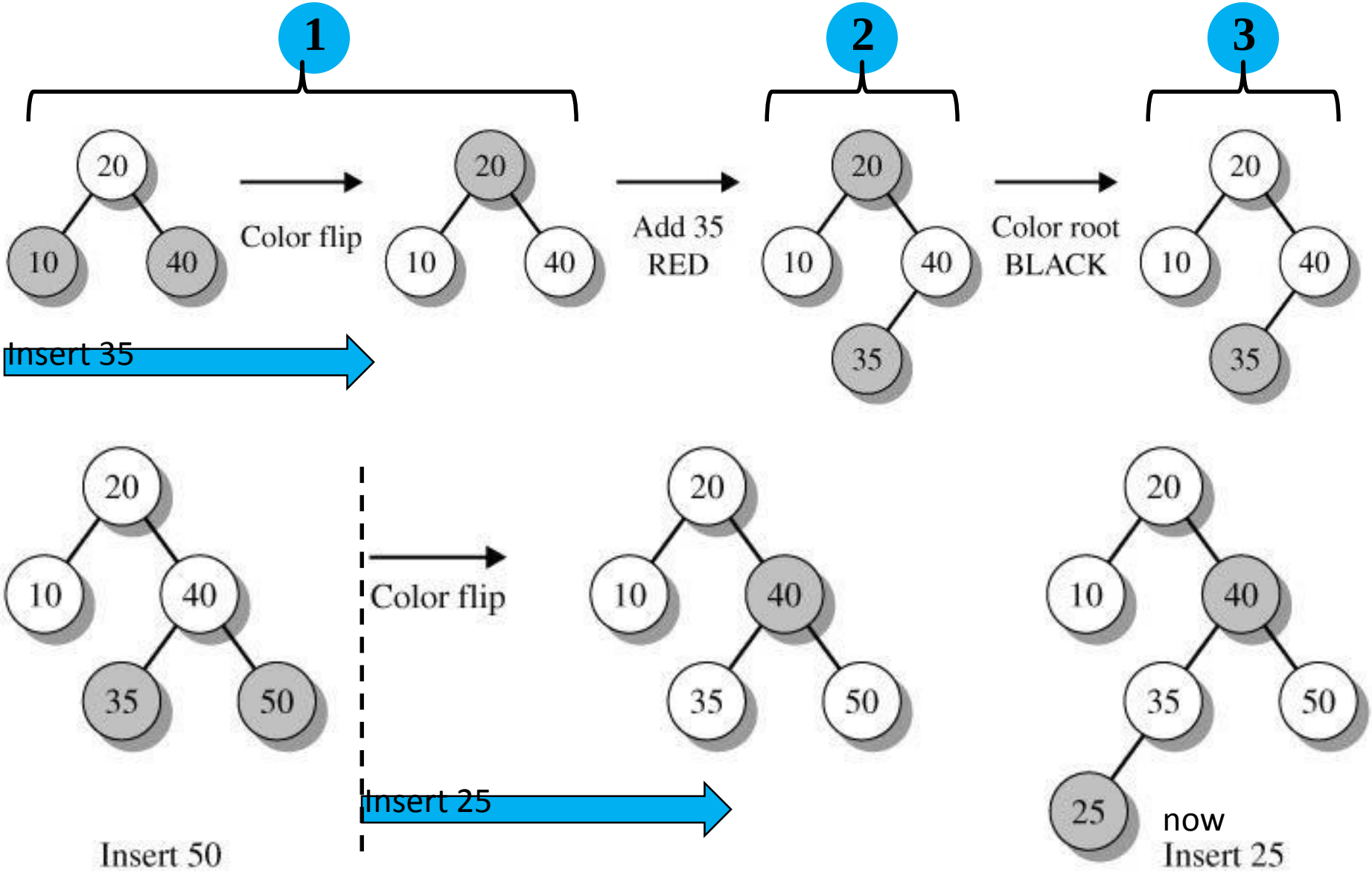
RL = double left rotation of node 10
(right then left)

Building a Red-Black Tree



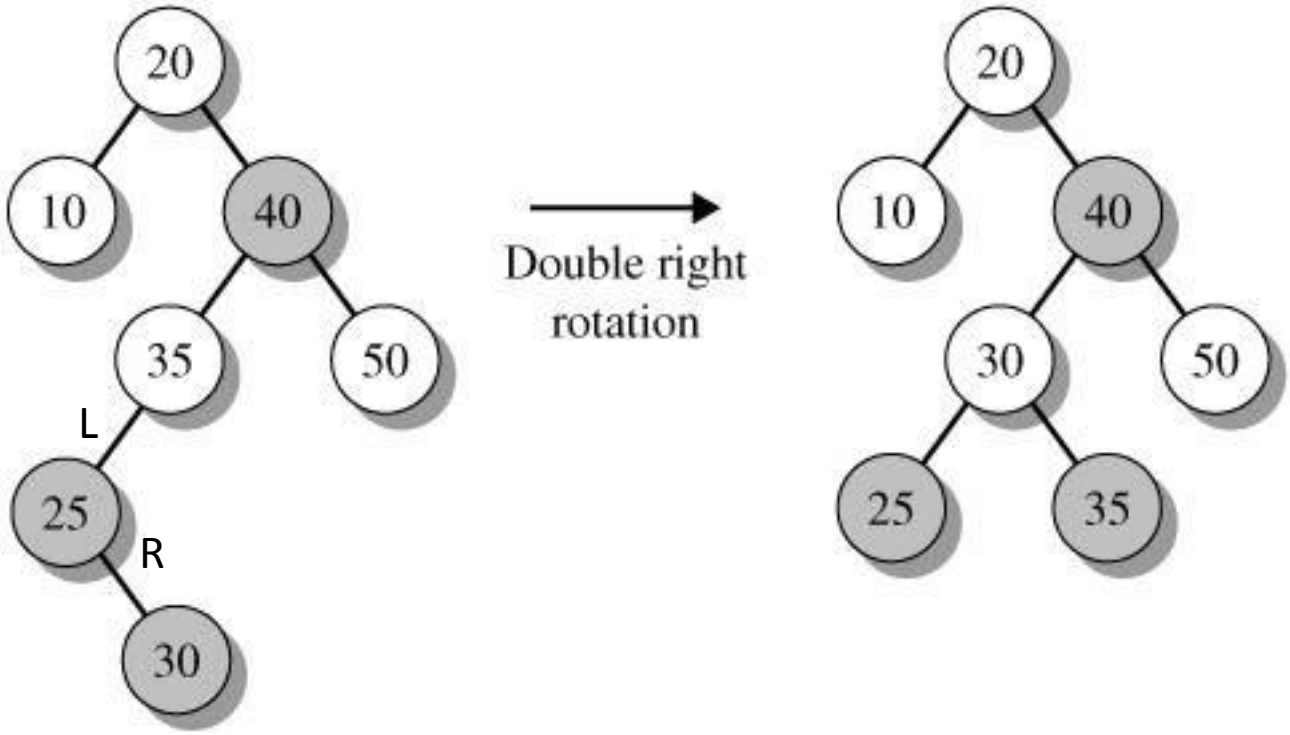


continued


1
2
3
20
10
40
Color flip
20
10
40
Add 35
RED
20
10
40
35
Color root
BLACK
20
10
40
35
Insert 35
20
10
40
35
50
Insert 50
Color flip
20
10
40
35
50
Insert 25
20
10
40
35
50
25
now
Insert 25

continued

Insert 30



LR = double right rotation of node 30
(left then right)