Topic 23
Red Black Trees

"People in every direction
 No words exchanged
 No time to exchange
 And all the little ants are marching
 Red and **Black** antennas waving"
          -*Ants Marching,* Dave Matthew's Band

"Welcome to L.A.'s Automated Traffic Surveillance and Control Operations Center. See, they use video feeds from intersections and specifically designed algorithms to predict traffic conditions, and thereby control traffic lights. So all I did was come up with my own... **kick ass algorithm** to sneak in, and now we own the place."

          -Lyle, the Napster, (Seth Green), *The Italian Job*

Clicker 1

‣ 2000 elements are inserted one at a time into an initially empty binary search tree using the simplenaive algorithm. What is the maximum possible height of the resulting tree?

A. 1

B. 11

C. 21

D. 500

E. 1999

Binary Search Trees

‣ Average case and worst case Big O for

– insertion

– deletion

– access

‣ Balance is important. Unbalanced trees give worse than log N times for the basic tree operations

‣ Can balance be guaranteed?

Red Black Trees

‣ A BST with more complex algorithms to ensure balance

‣ Each node is labeled as Red or Black.

‣ Path: A unique series of links (edges) traverses from the root to each node.

– The number of edges (links) that must be followed is the path length

‣ In Red Black trees paths from the root to elements with 0 or 1 child are of particular interest

Paths to Single or Zero Child Nodes

‣ How many?
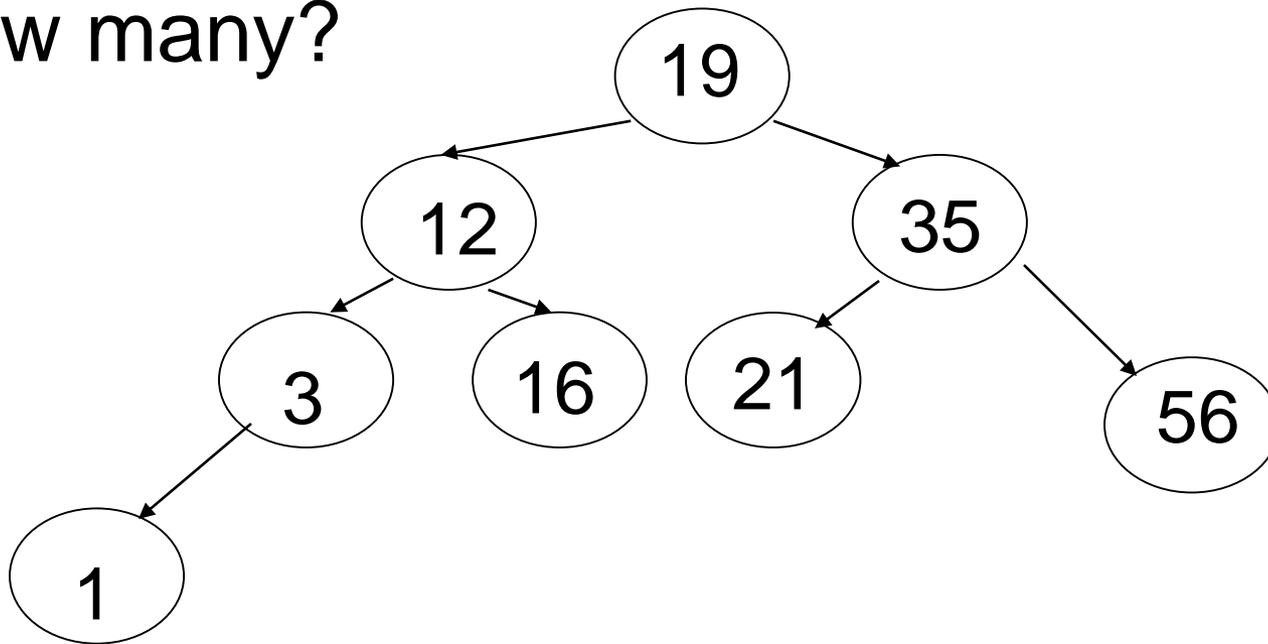
Red Black Trees

Red Black Tree Rules

1. Is a binary search tree

2. Every node is colored either red or black

3. The root of the whole tree is black

4. If a node is red its children must be black. (a.k.a. the red rule)

5. Every path from a node to a null link must contain the same number of black nodes (a.k.a. the path rule)

Example of a Red Black Tree

‣ The root of a Red Black tree is black

‣ Every other node in the tree follows these rules:

– Rule 3: If a node is Red, all of its children are Black

– Rule 4: The number of Black nodes must be the same in all paths from the root node to null nodes

Red Black Trees

Red Black Tree?

19

12

0

35

50
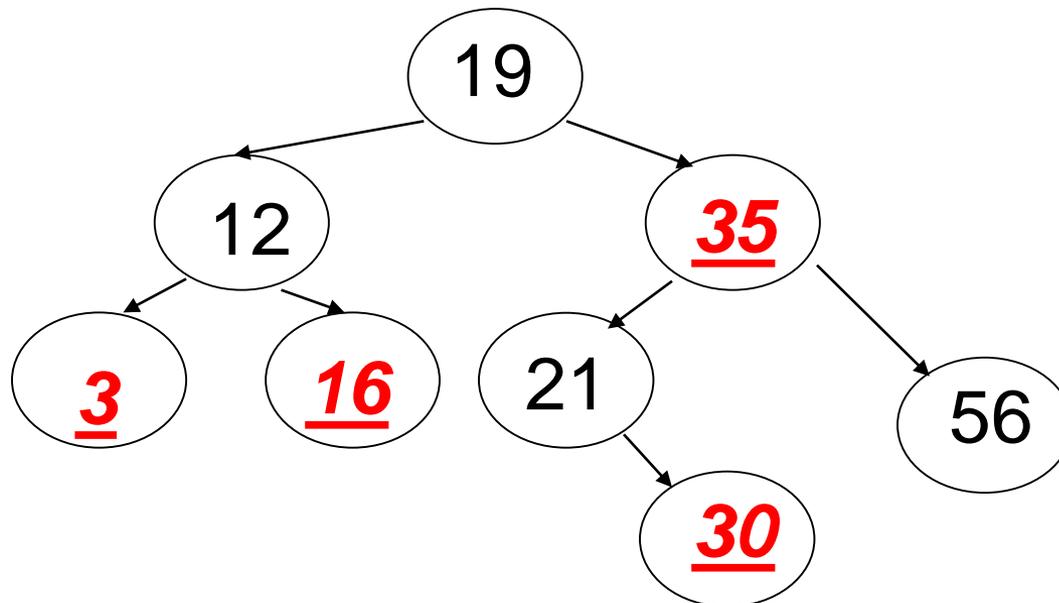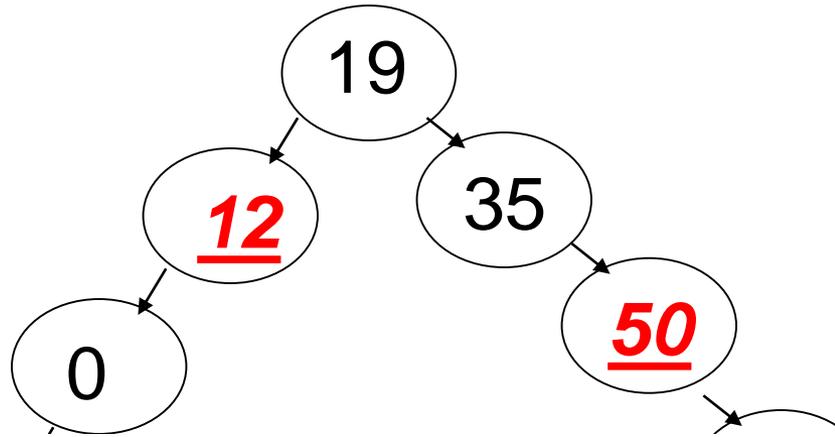
Clicker 2

▸ Is the tree on the previous slide a binary search tree? Is it a red black tree?

|	BST?	Red-Black?
A.	No	No
B.	No	Yes
C.	Yes	No
D.	Yes	Yes

Red Black Tree?
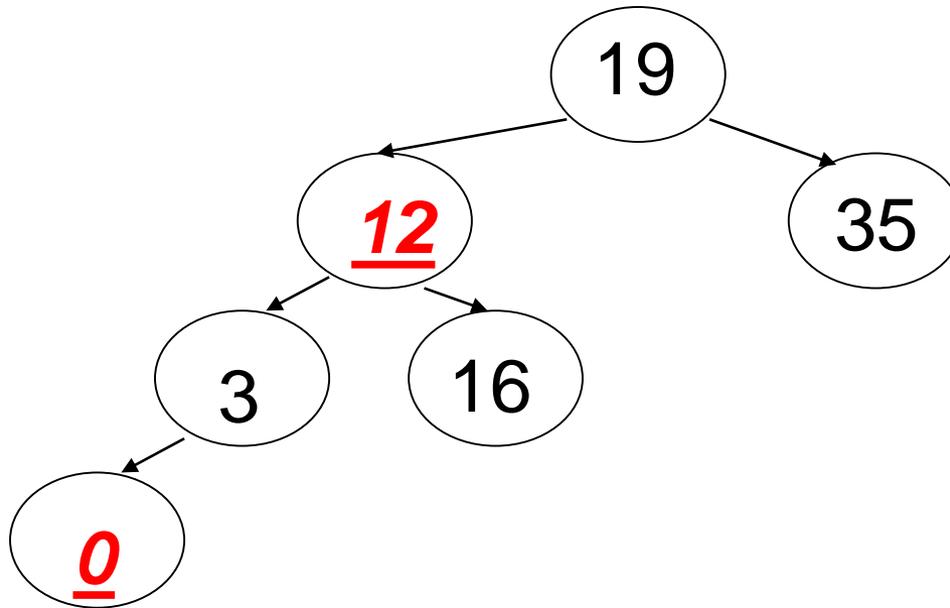
Perfect?
Full?

Complete?

Red Black Trees

Clicker 3

‣ Is the tree on the previous slide a binary search tree? Is it a red black tree?

	BST?	Red-Black?
A.	No	No
B.	No	Yes
C.	Yes	No
D.	Yes	Yes

Implications of the Rules

‣ If a Red node has any children, it must have two children and they must be Black. (Why?)

‣ If a Black node has only one child that child must be a Red leaf. (Why?)

‣ Due to the rules there are limits on how unbalanced a Red Black tree may become.

  – on the previous example may we hang a new node off of the leaf node that contains 0?

Properties of Red Black Trees

- If a Red Black Tree is complete, with all Black nodes except for Red leaves at the lowest level the height will be minimal, ~log N

- To get the max height for N elements there should be as many Red nodes as possible down one path and all other nodes are Black
  - This means the max height would b **approximately** 2 * log N (don't use this as a formula)
  - typically less than this
  - see example on next slide
  - interesting exercise, draw max height tree with N nodes

Max Height Red Black Tree
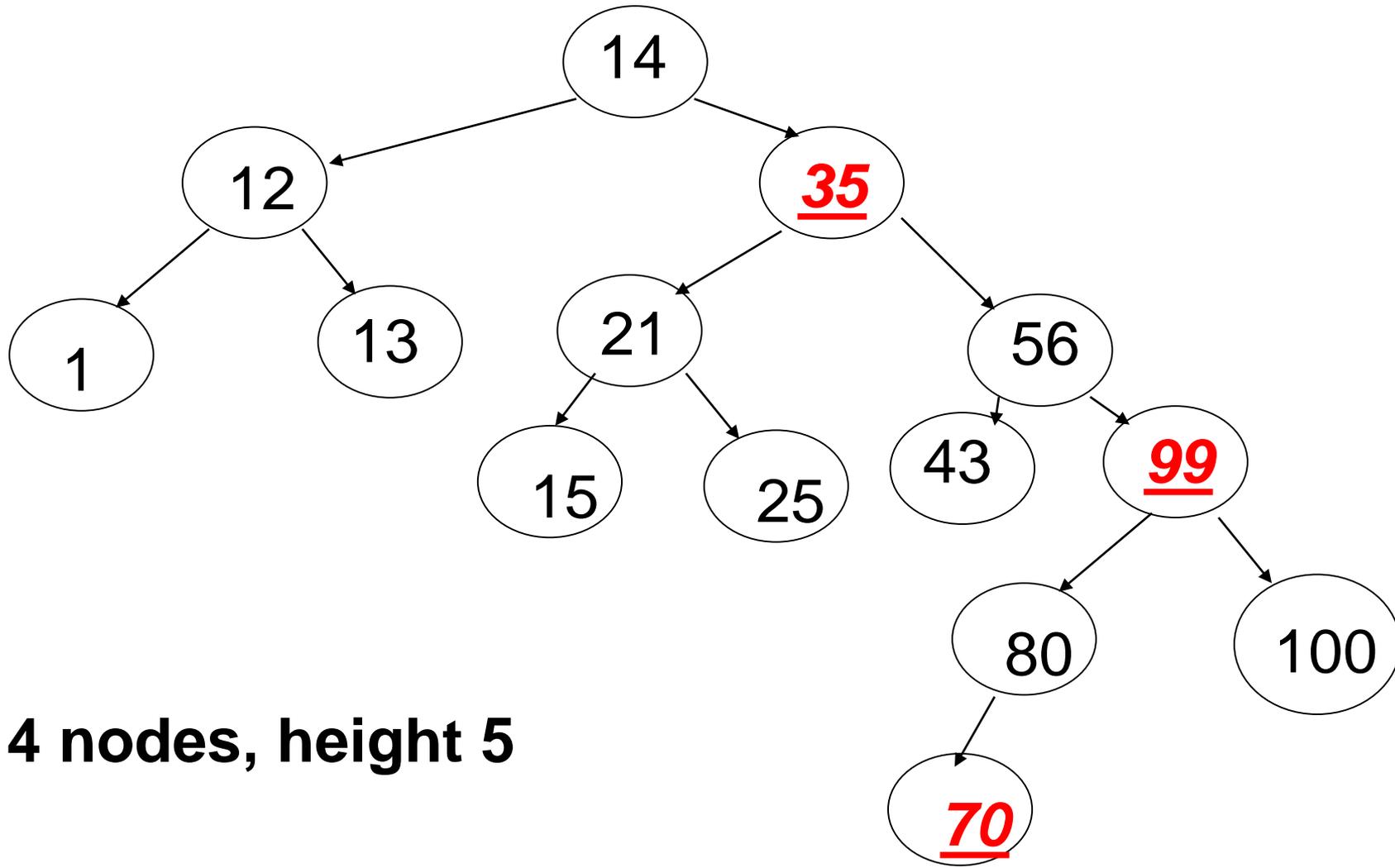
14 nodes, height 5

Red Black Trees

Maintaining the Red Black Properties in a Tree

‣ Insertions

‣ Must maintain rules of Red Black Tree.

‣ New Node always a leaf

– can't be black or we will violate rule 4

– therefore the new leaf must be red

– If parent is black, done (trivial case)

– if parent red, things get interesting because a red leaf with a red parent violates rule 3

Insertions with Red Parent - Child

Must modify tree when insertion would result in Red Parent - Child pair using color changes and *rotations.*

Red Black Trees
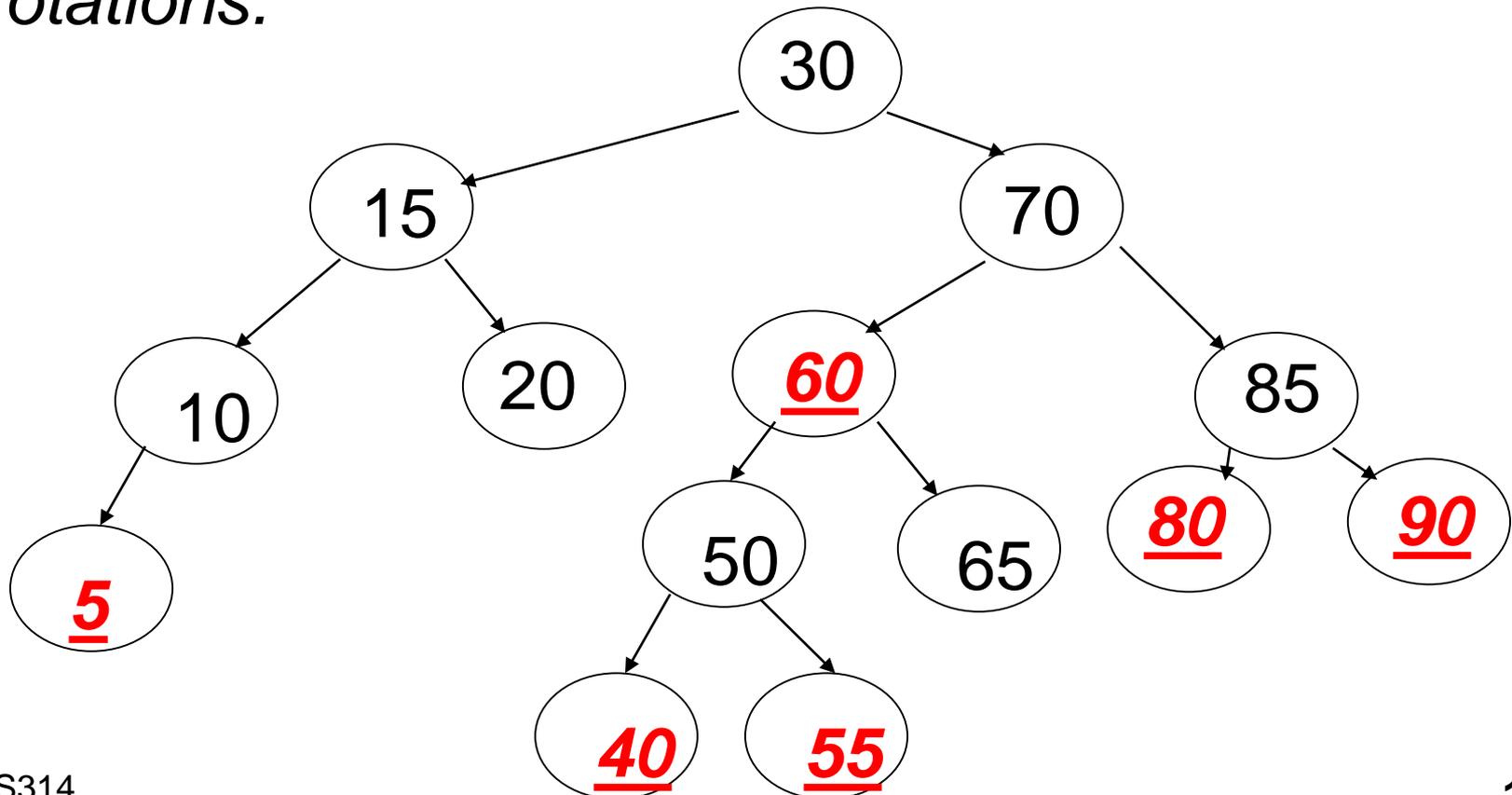
Case 1

- Suppose sibling of parent is Black.
  - by convention null nodes are black
- In the previous tree, true if we are inserting a 3 or an 8.
  - What about inserting a 99? Same case?
- Let X be the new leaf Node, P be its Red Parent, S the Black sibling and G, P's and S's parent and X's grandparent
  - What color is G?

Case 1 - The Picture
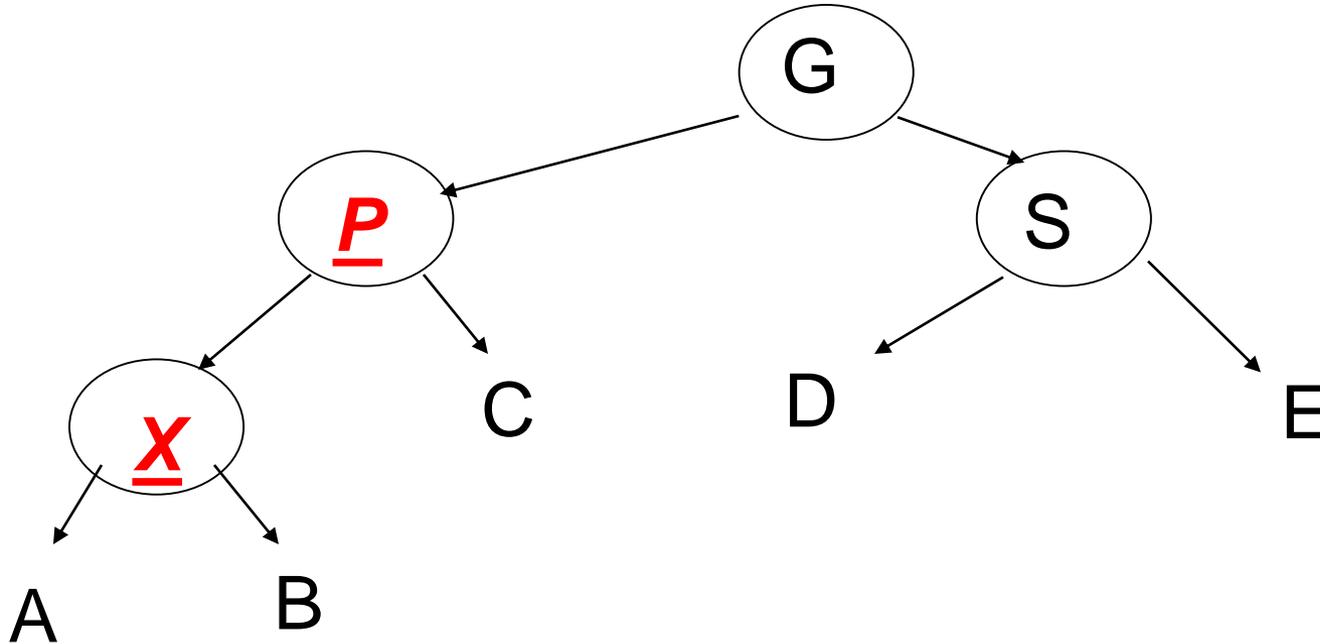
Relative to G, X could be an *inside* or *outside* node.
Outside -> left left or right right moves
Inside -> left right or right left moves

Red Black Trees

Fixing the Problem
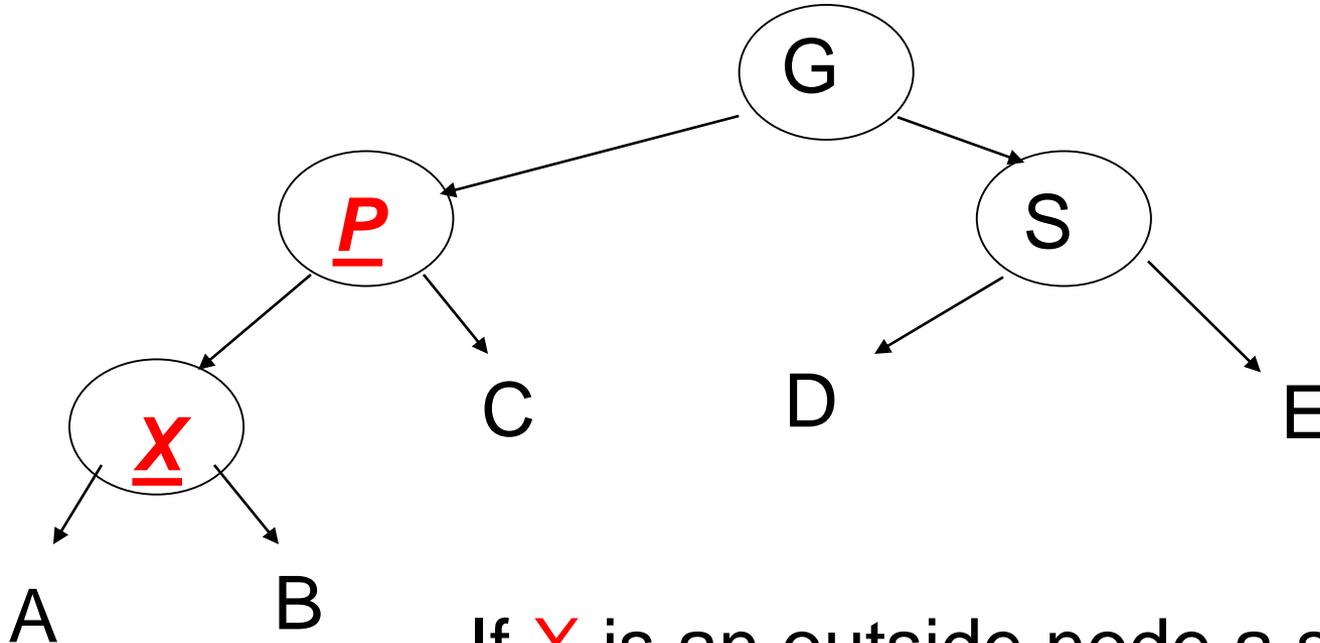
If X is an outside node a single *rotation* between P and G fixes the problem. A rotation is an exchange of roles between a parent and child node. So P becomes G's parent. Also must recolor P and G.

19

Red Black Trees
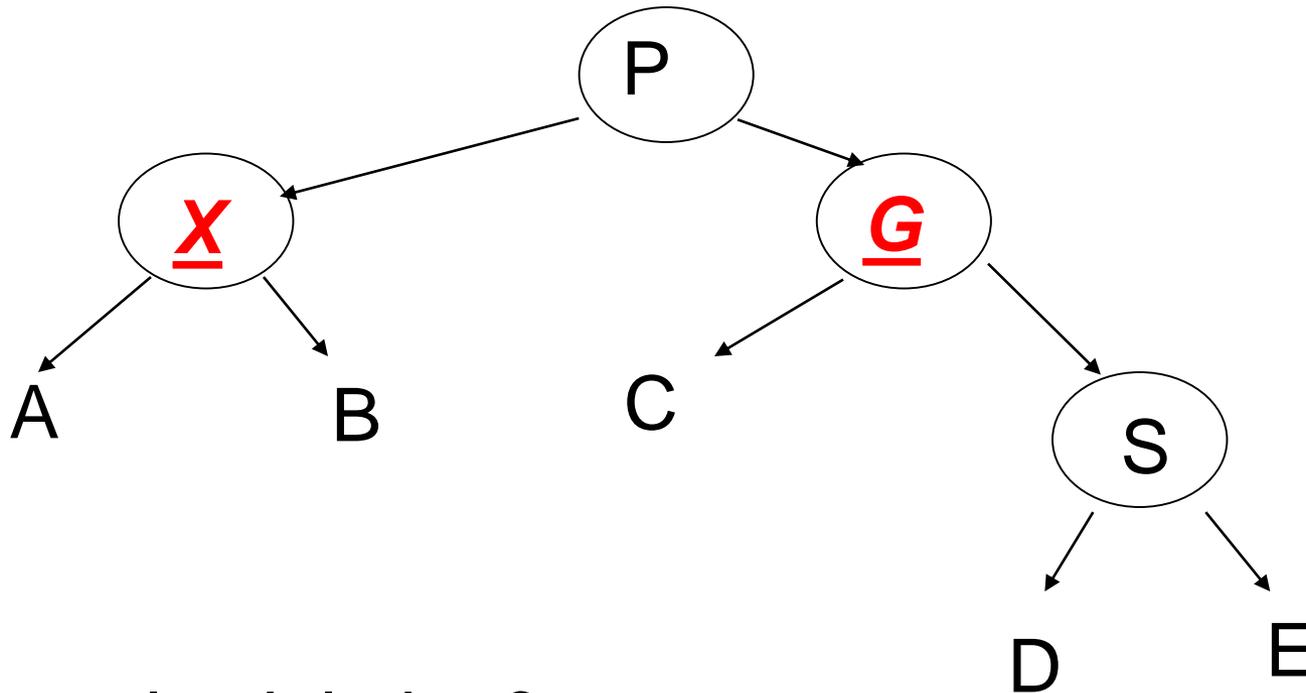
Single Rotation

Apparent rule violation?
Recall, S is null if X is a leaf, so no problem

If this occurs higher in the tree (why?) subtrees A, B, and C will have one more black node than D and E.

Red Black Trees

Case 2

‣ What if X is an inside node relative to G?
  – a single rotation will not work
‣ Must perform a double rotation
  – rotate X and P
  – rotate X and G

Red Black Trees
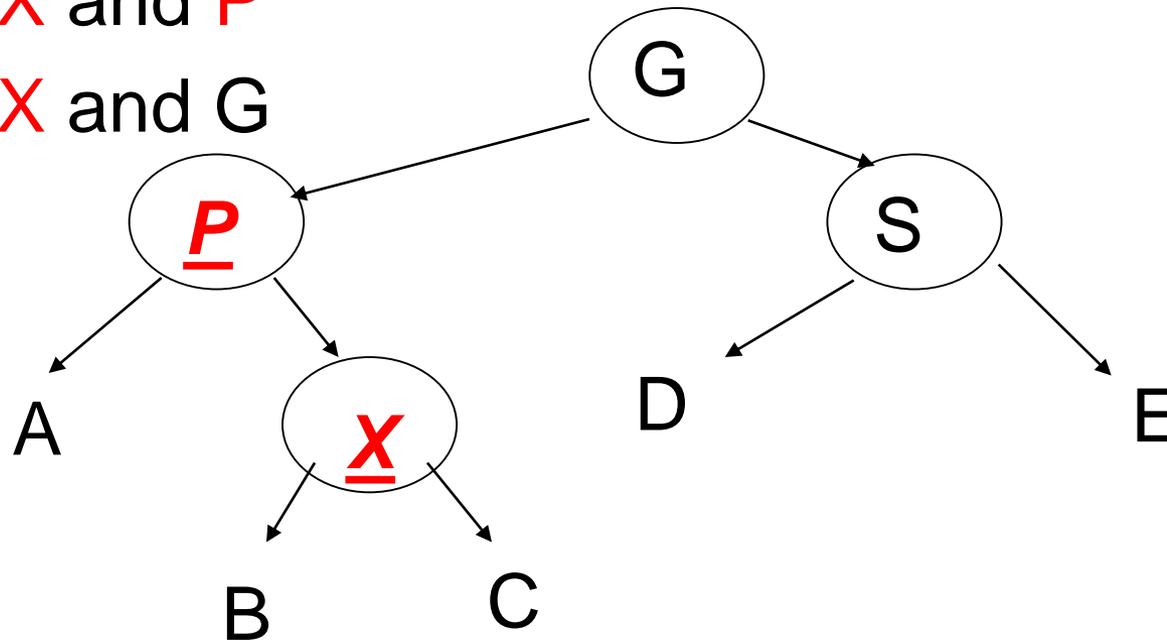
First Rotation

‣ Rotate P and X, no color change
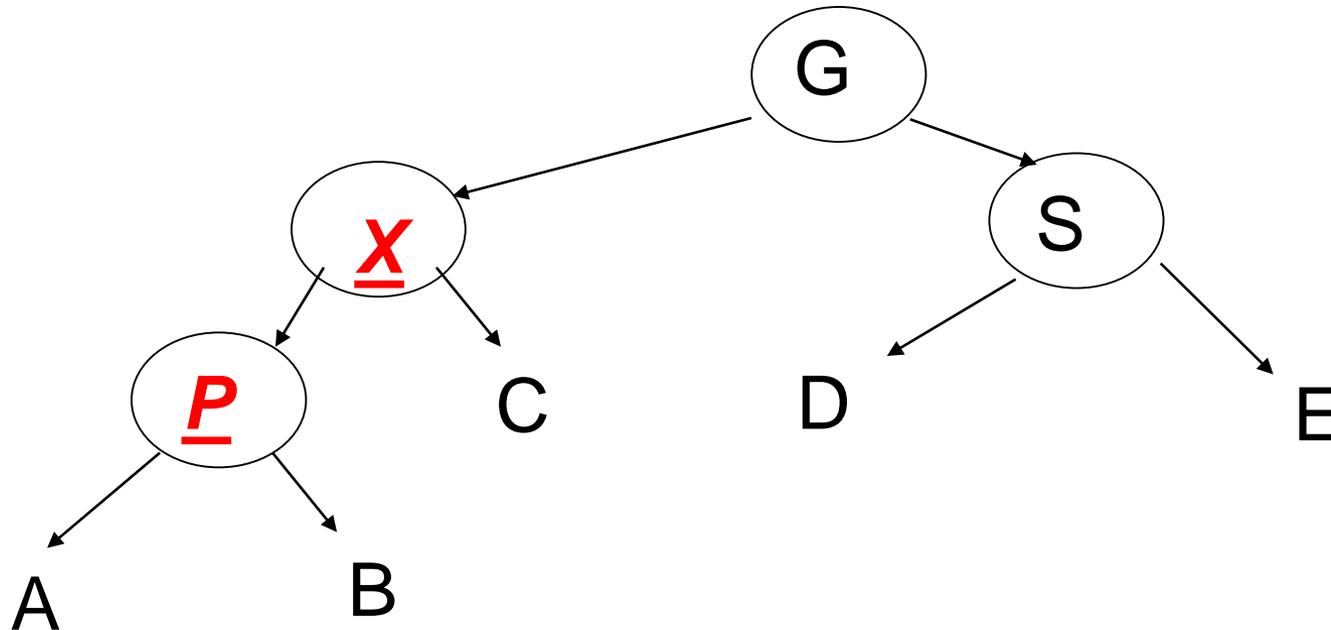
‣ What does this actually do?

After Double Rotation

Red Black Trees
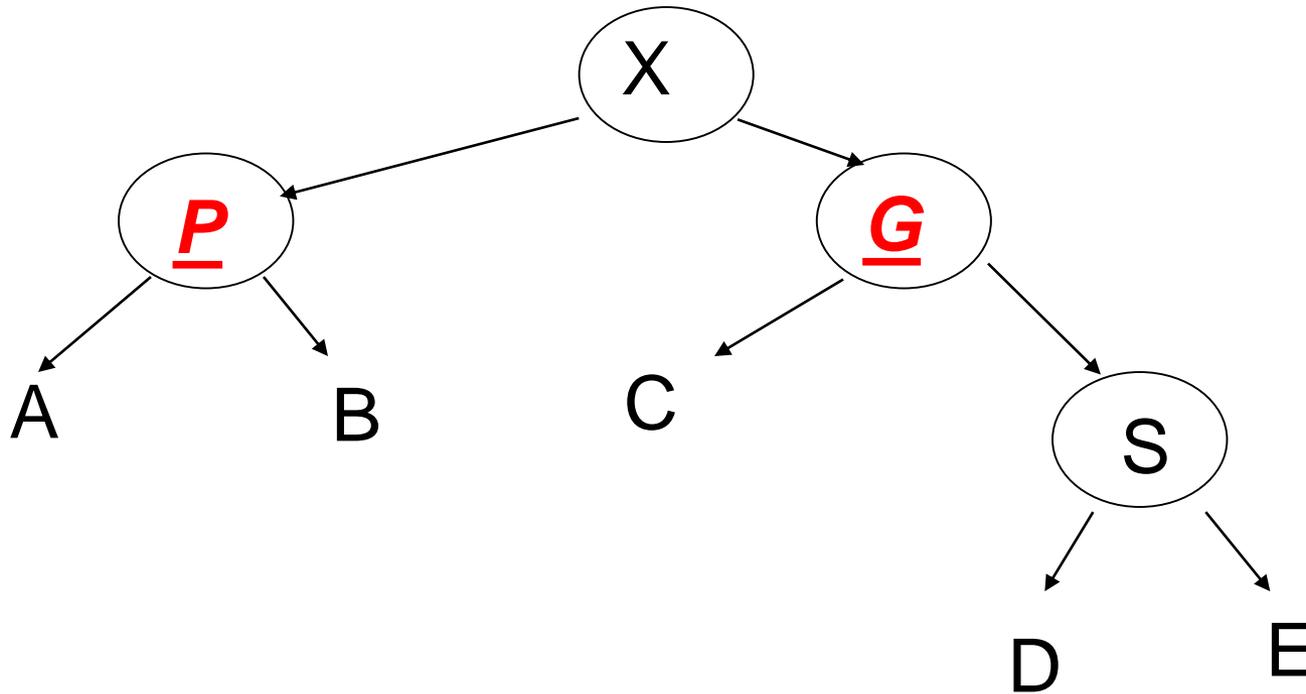
Case 3
Sibling is Red, not Black
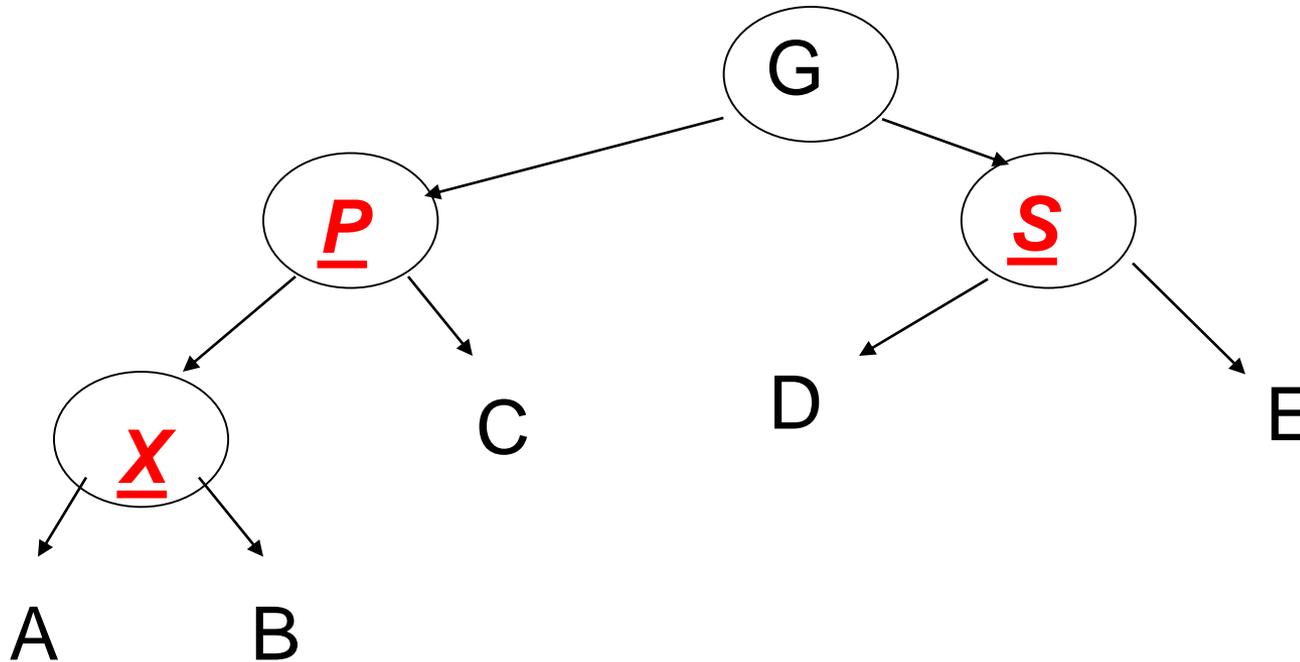
Any problems?

Fixing Tree when S is Red

‣ Must perform single rotation between parent, P and grandparent, G, and then make appropriate color changes
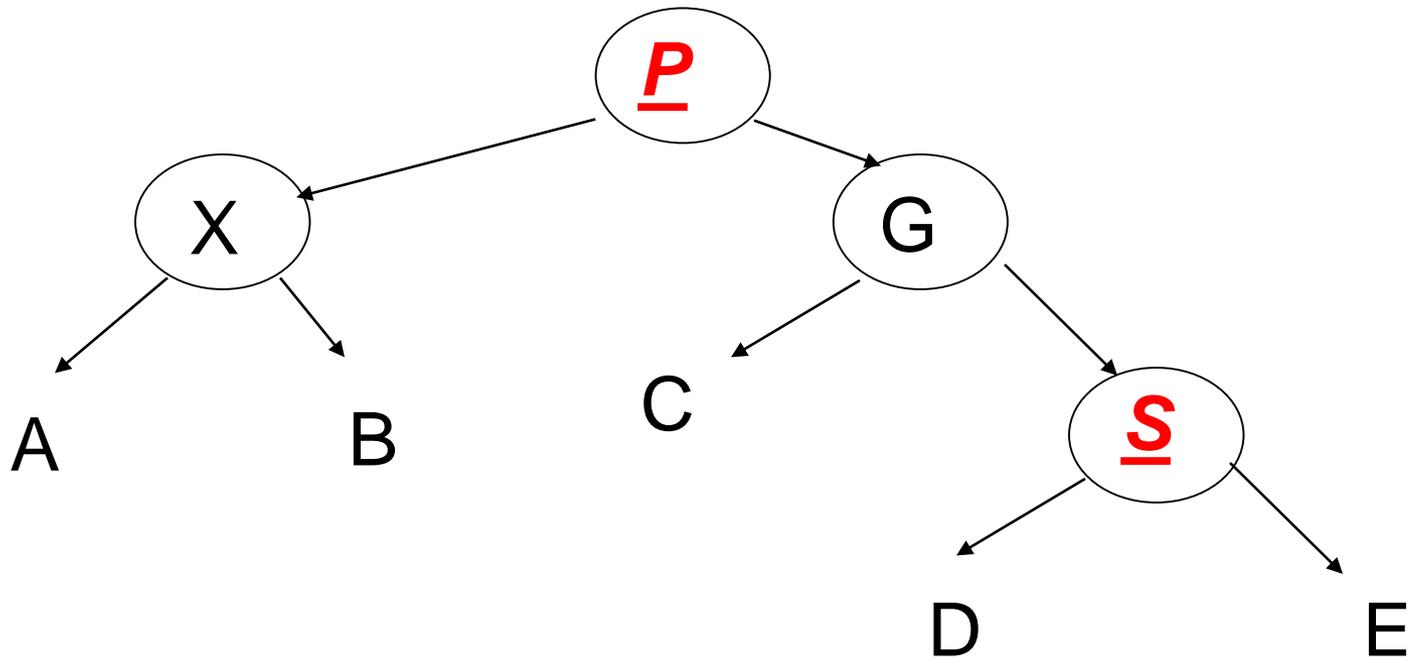
More on Insert

‣ Problem: What if on the previous example G's parent (GG!) had been red?

‣ Easier to never let Case 3 ever occur!

‣ On the way down the tree, if we see a node X that has 2 Red children, we make X Red and its two children black.

  – if recolor the root, recolor it to black
  – the number of black nodes on paths below X remains unchanged
  – If X's parent was Red then we have introduced 2 consecutive Red nodes.(violation of rule)
  – to fix, apply rotations to the tree, same as inserting node

# Example of Inserting Sorted Numbers

▸ 1 2 3 4 5 6 7 8 9 10

1

1
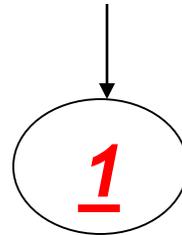
Insert 1. A leaf so red. Realize it is root so recolor to black.

Red Black Trees
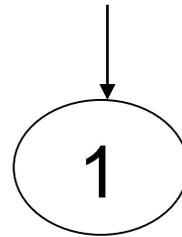
# Insert 2

make 2 red. Parent
is black so done.

Red Black Trees
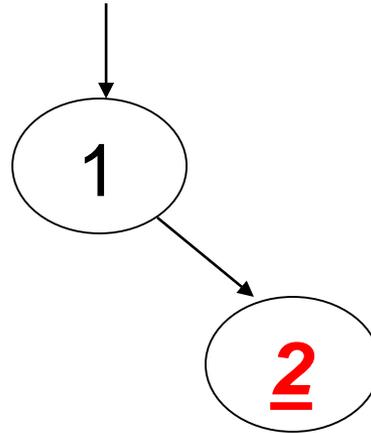
Insert 3

Insert 3. Parent is red. Parent's sibling is black (null) 3 is outside relative to grandparent. Rotate parent and grandparent

1

2

3

2

1   **3**
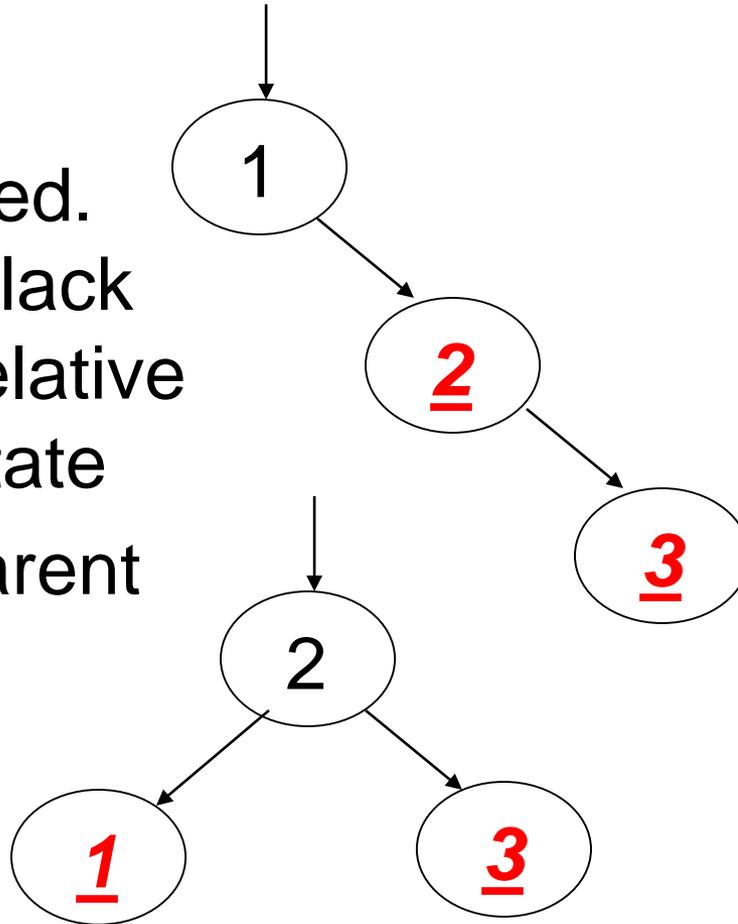
Insert 4

On way down see
2 with 2 red children.
Recolor 2 red and
children black.

2

1   *3*   *2*

1   3

2
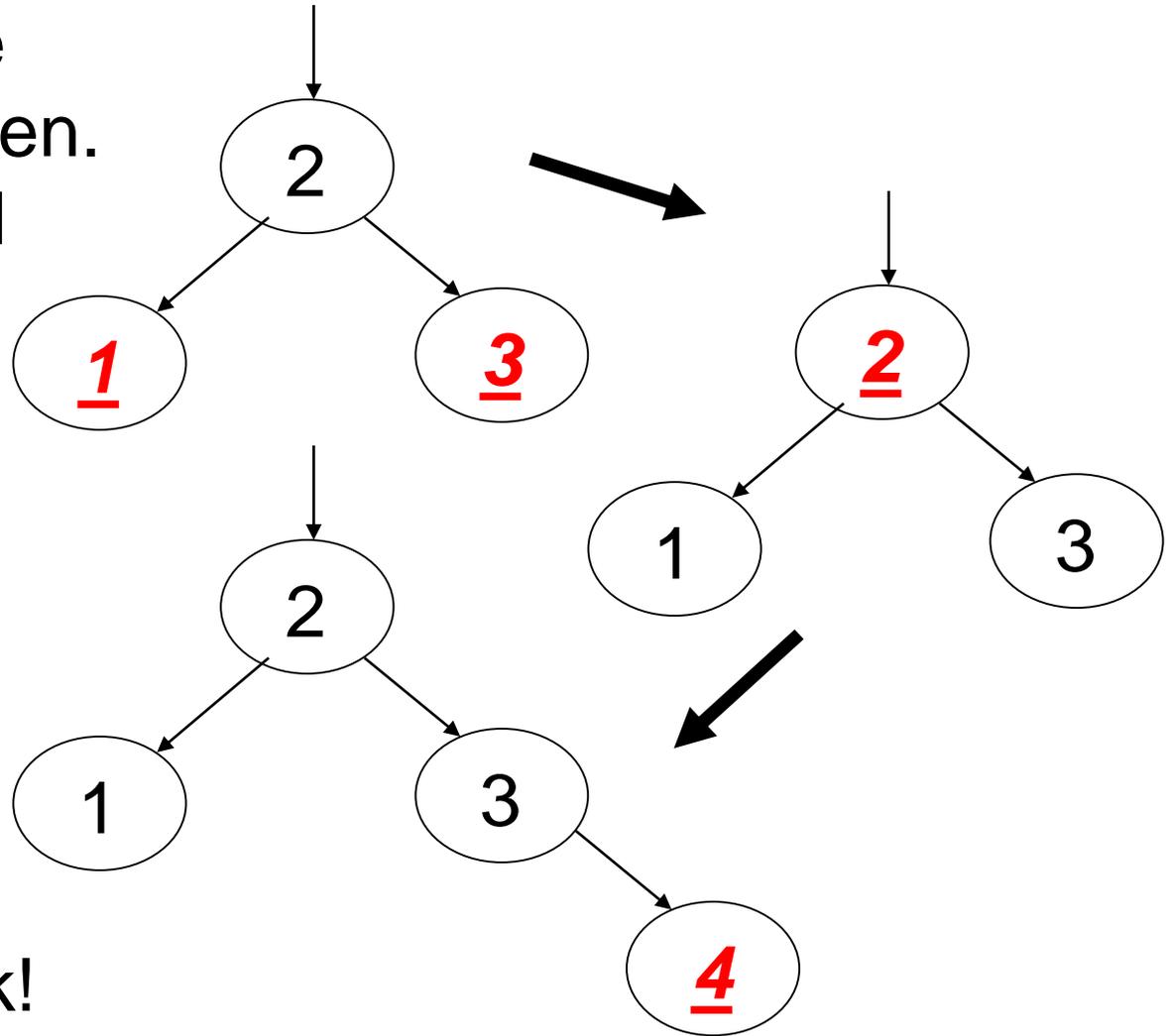
When adding 4
parent is black
so done.

1   3

Set root to black!

4

Red Black Trees

Insert 5

5's parent is red.
Parent's sibling is
black (null). 5 is
outside relative to
grandparent (3) so rotate
parent and grandparent then
recolor

Finish insert of 5

Red Black Trees
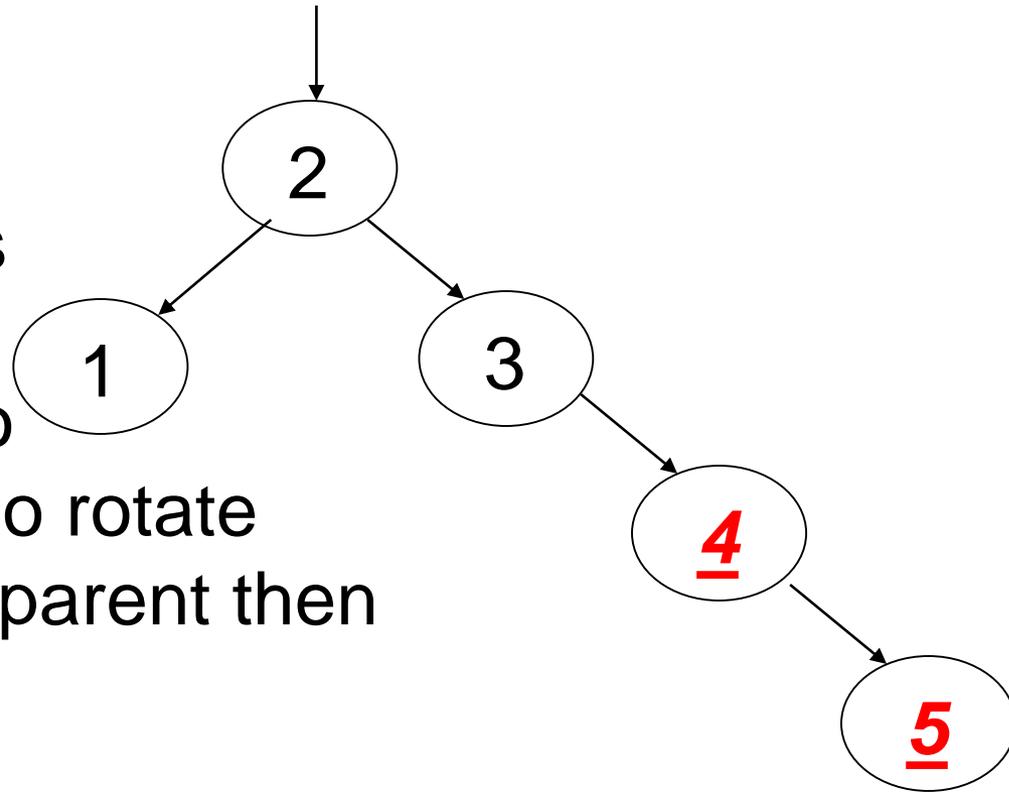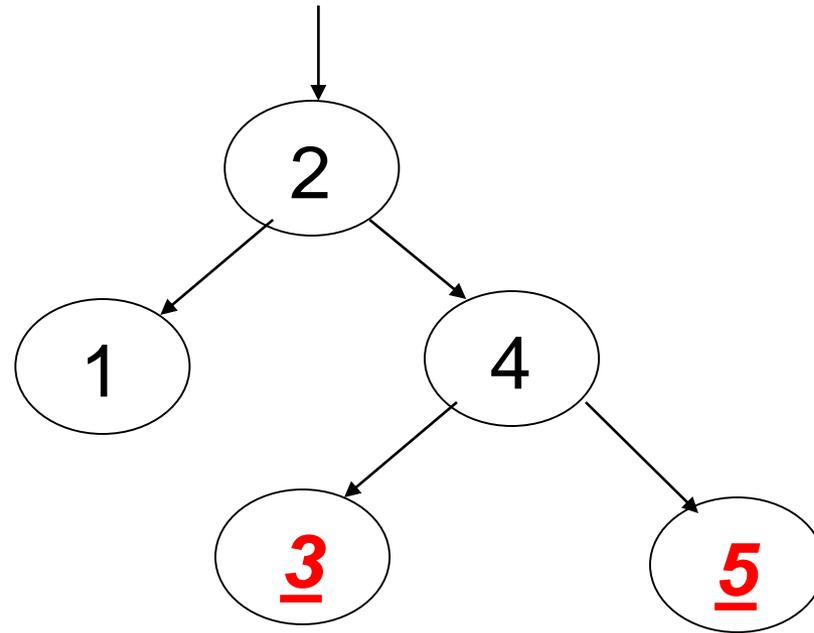
Insert 6

On way down see
4 with 2 red
children. Make
4 red and children
black. 4's parent is
black so no problem.

2

1

4

3

5
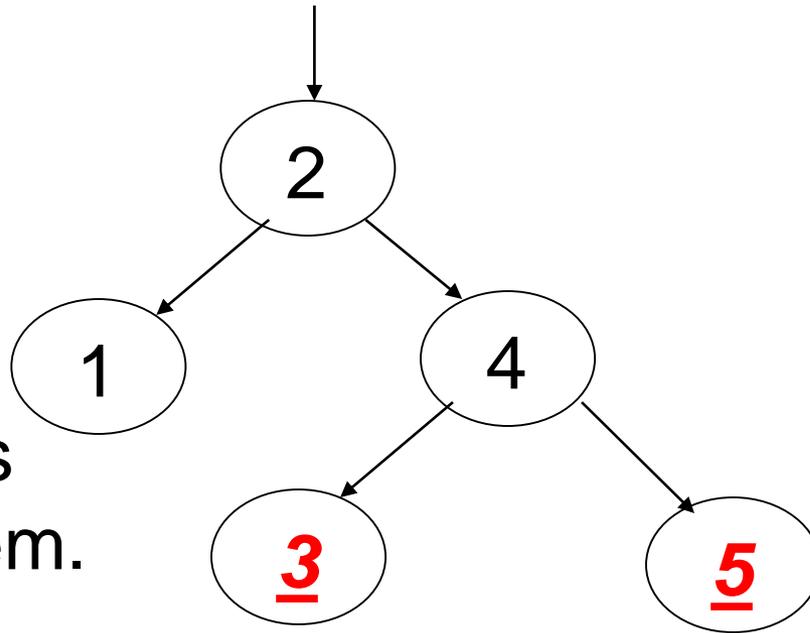
Finishing insert of 6

6's parent is black
so done.

2

1   *4*

3   5

6

Red Black Trees
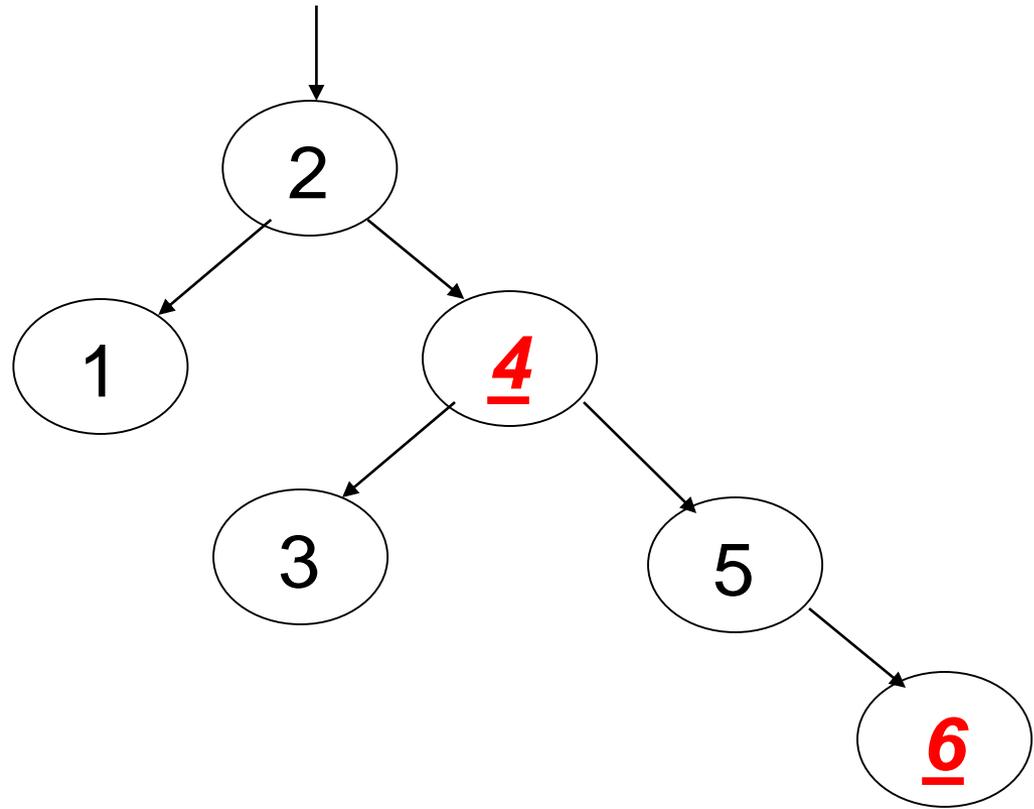
Insert 7

7's parent is red. Parent's sibling is black (null). 7 is outside relative to grandparent (5) so rotate parent and grandparent then recolor

2

1

4

3

5

6

7

Red Black Trees
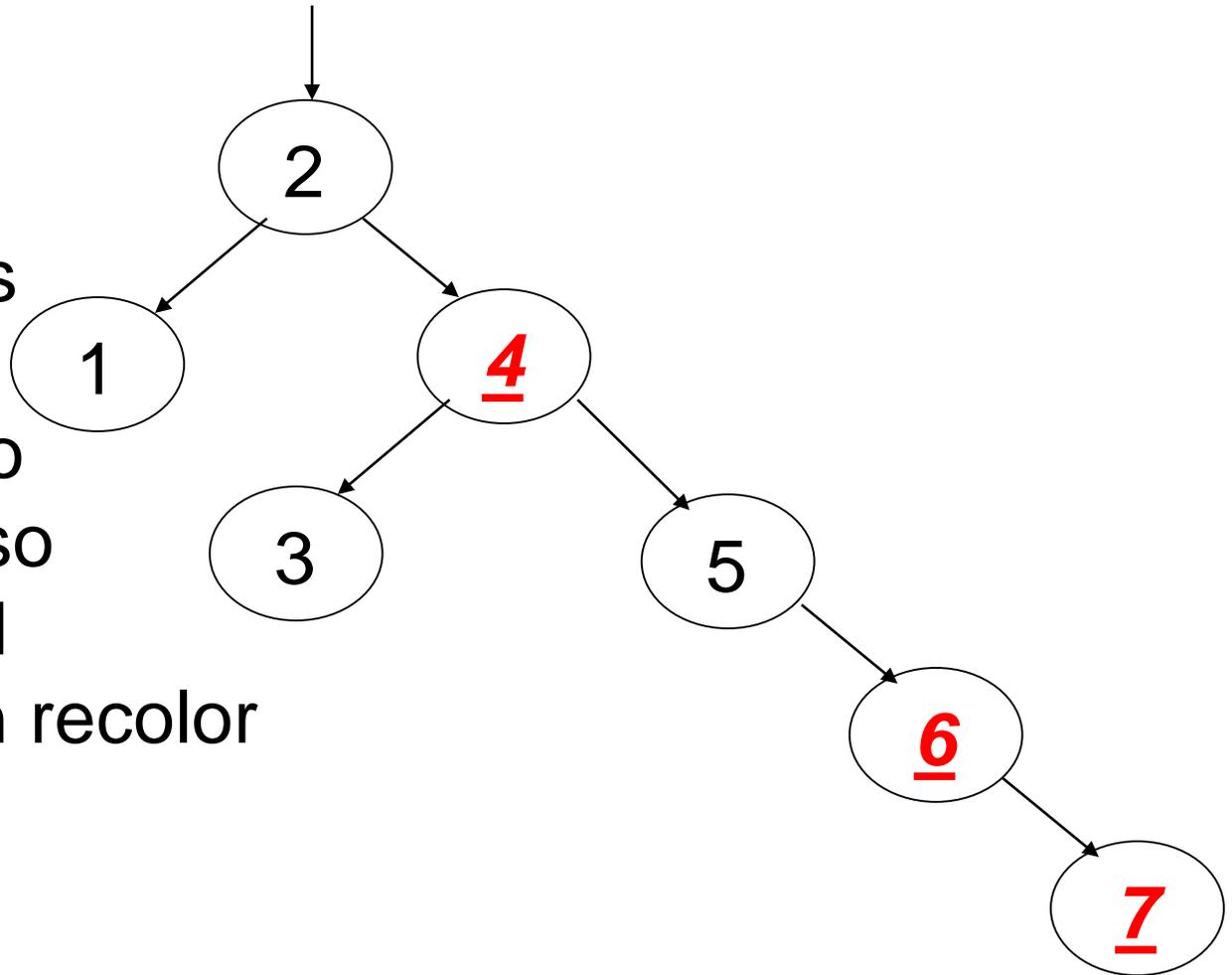
Finish insert of 7

Insert 8

The caveat!!!
Getting unbalanced
on that right subtree?!?
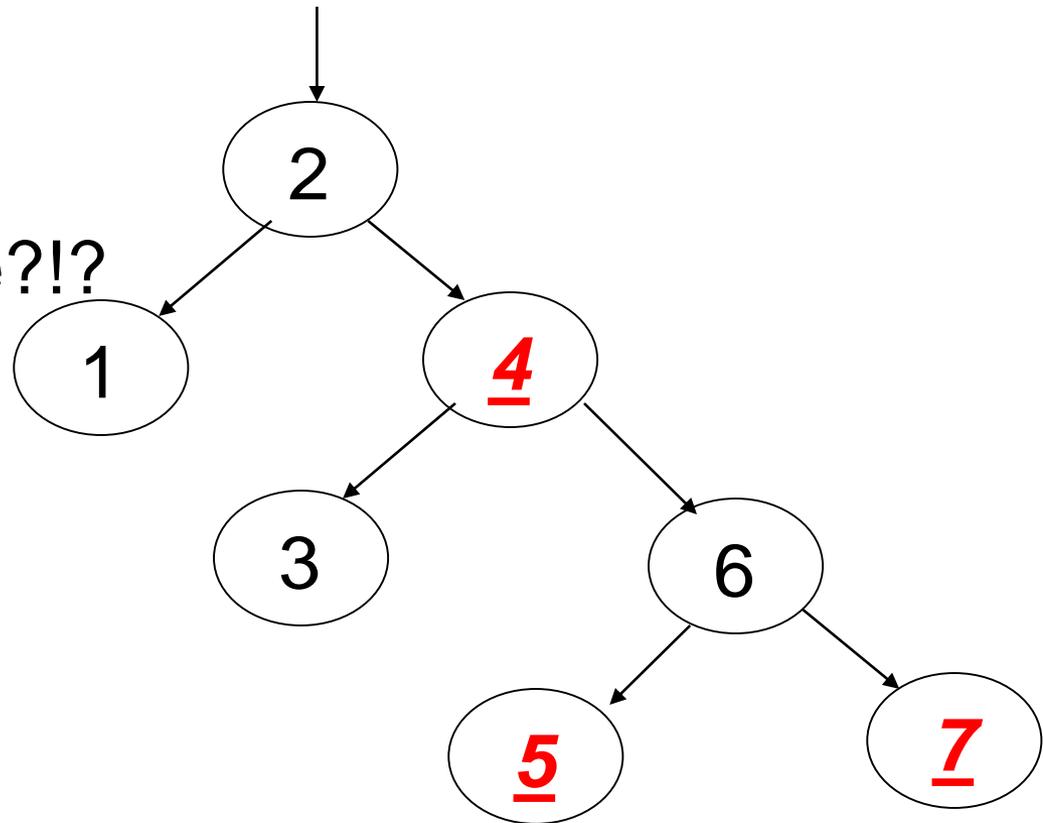
On way down see 6
with 2 red children.
Make 6 red and
children black. This
creates a problem
because 6's parent, 4, is
also red. Must perform
rotation.

2

1   *4*

3   6

5   *7*

Red Black Trees

Still Inserting 8

Recolored now
need to
rotate.

Recall, the subtrees
and the one extra
black node.

Red Black Trees
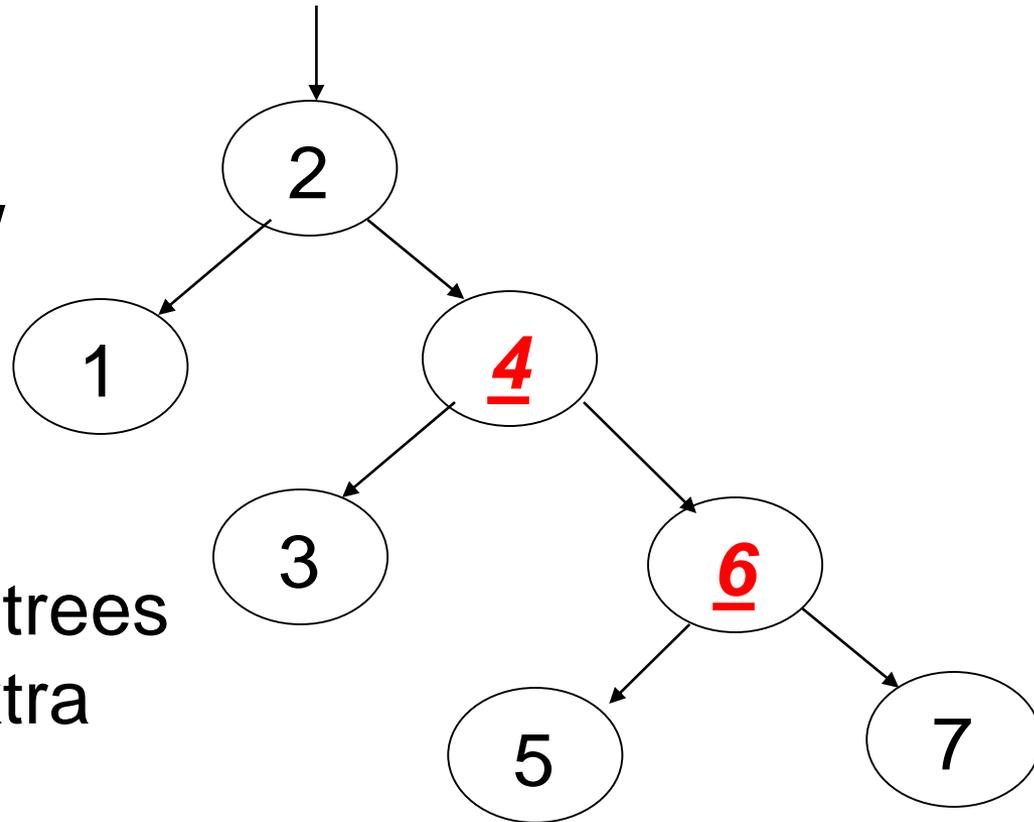
Finish inserting 8

Result of
rotation

39

Insert 9

4

2   6

1   3   5   7

8

9

On way down see 4 has two red children
so recolor 4 red and children black.
Realize 4 is the root so recolor black

Red Black Trees
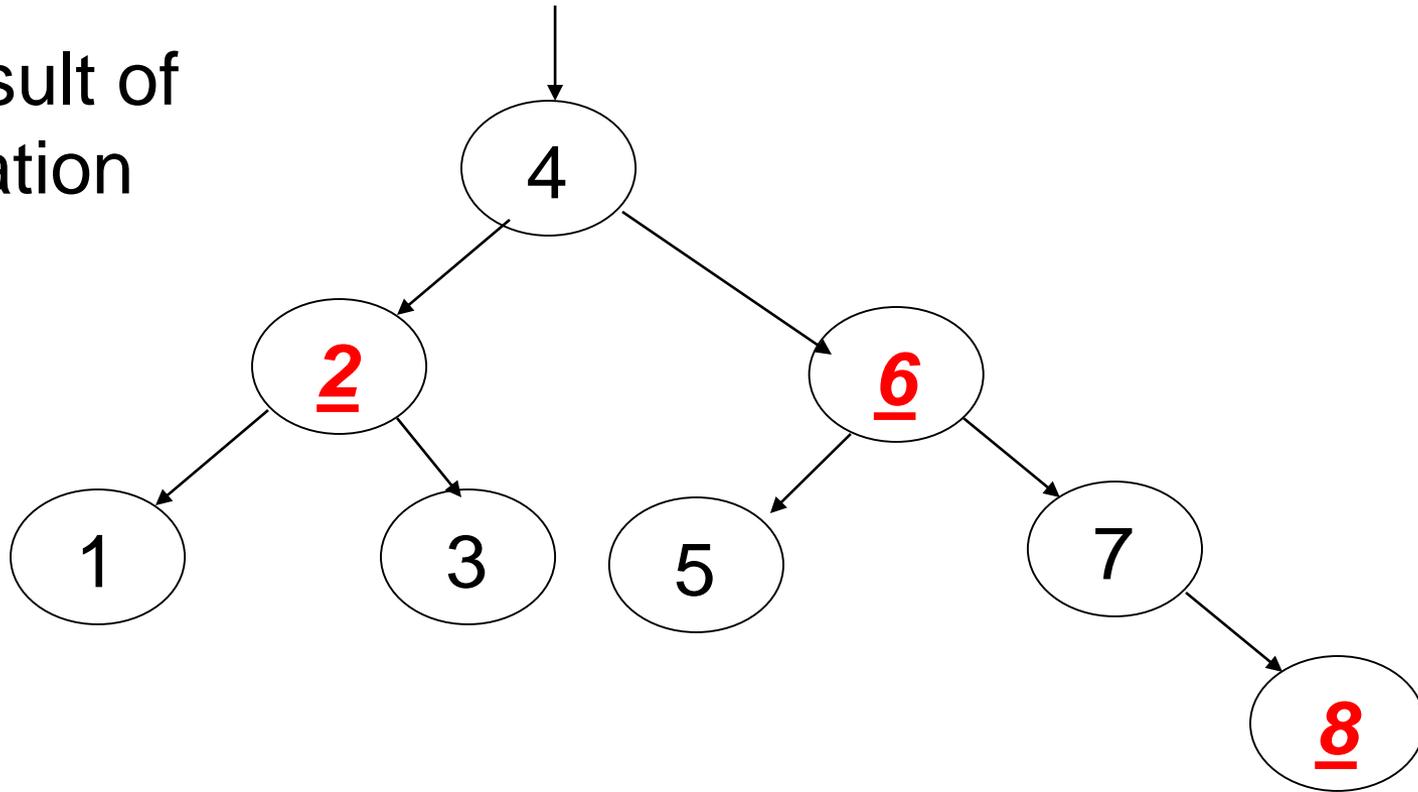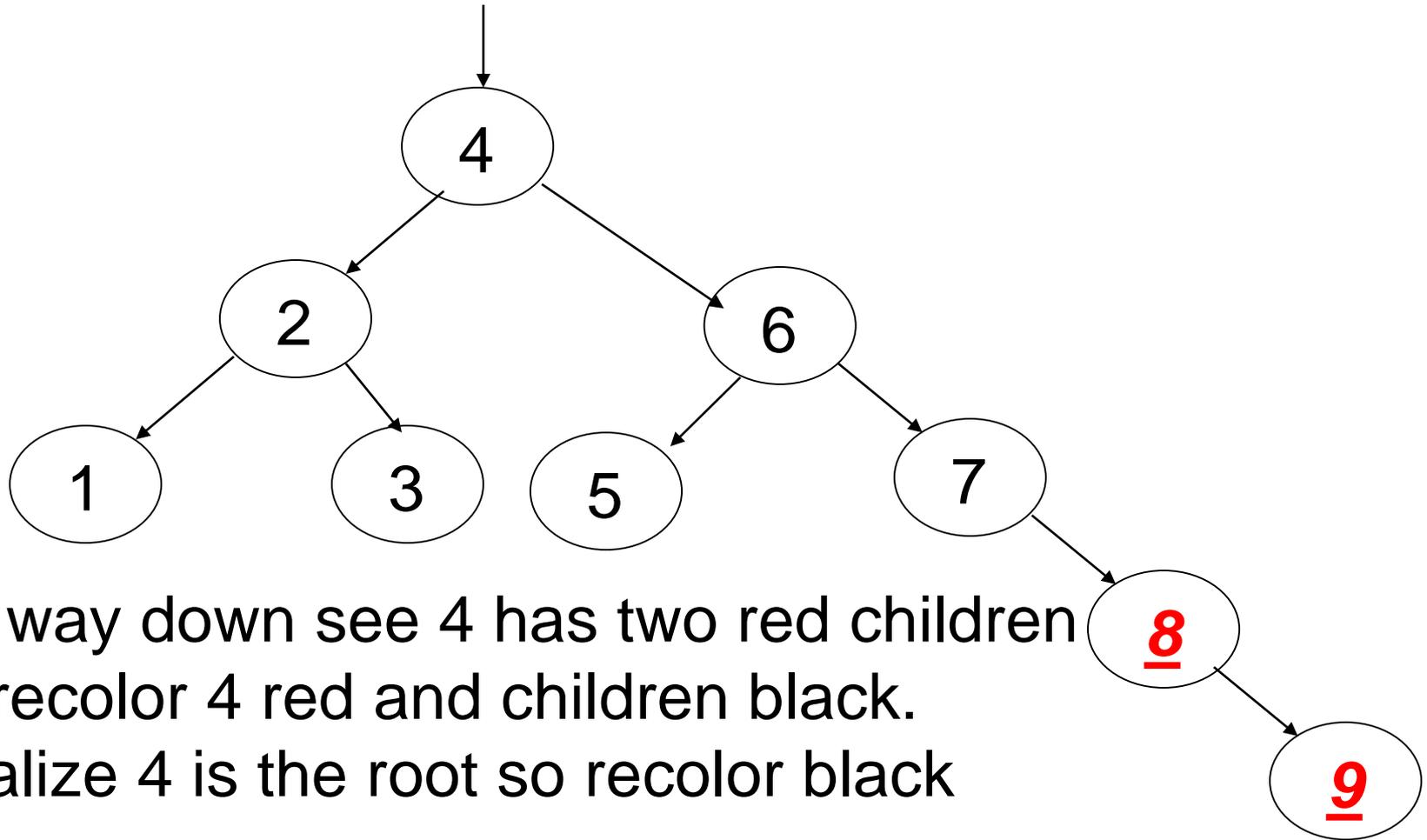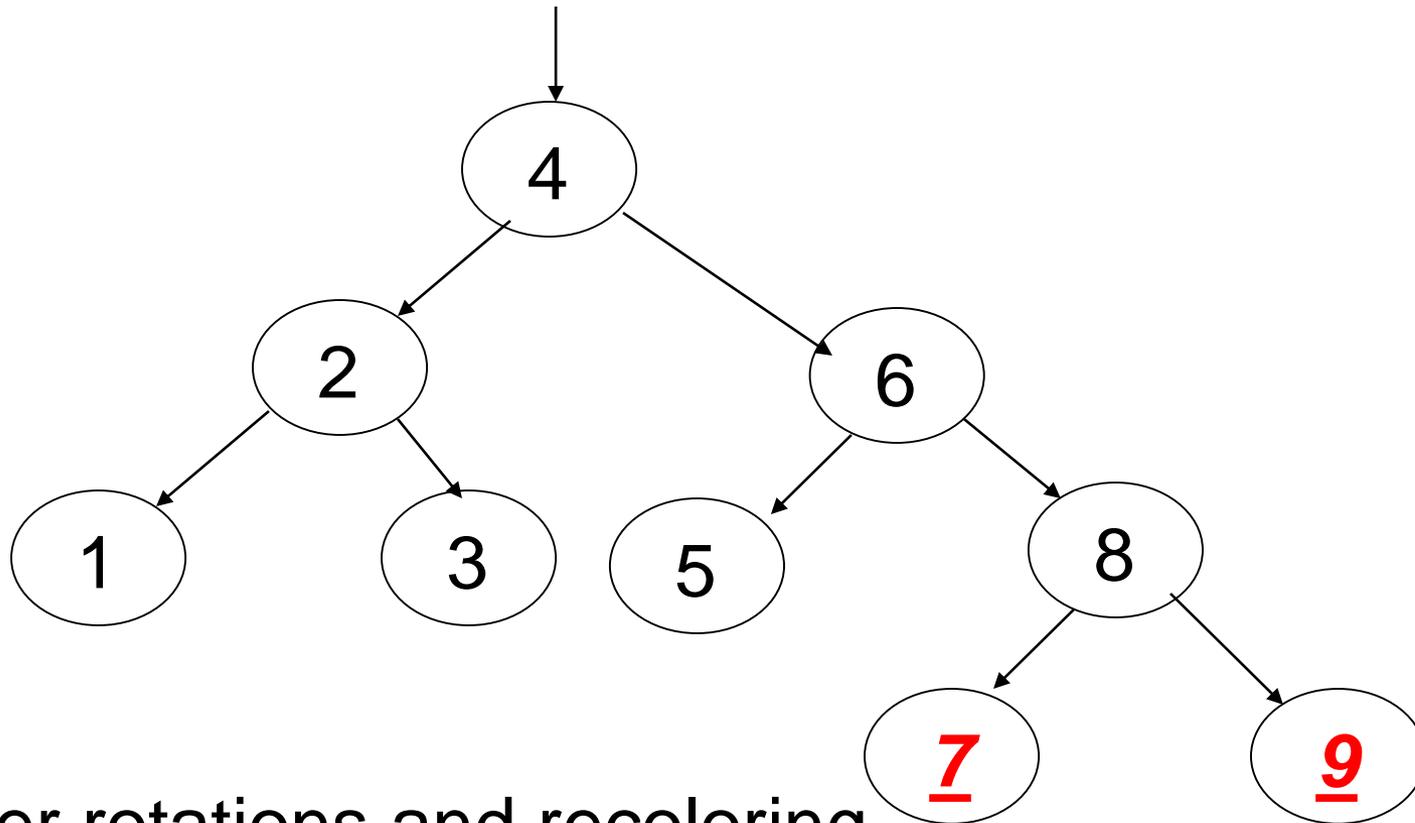
Finish Inserting 9

After rotations and recoloring

Insert 10
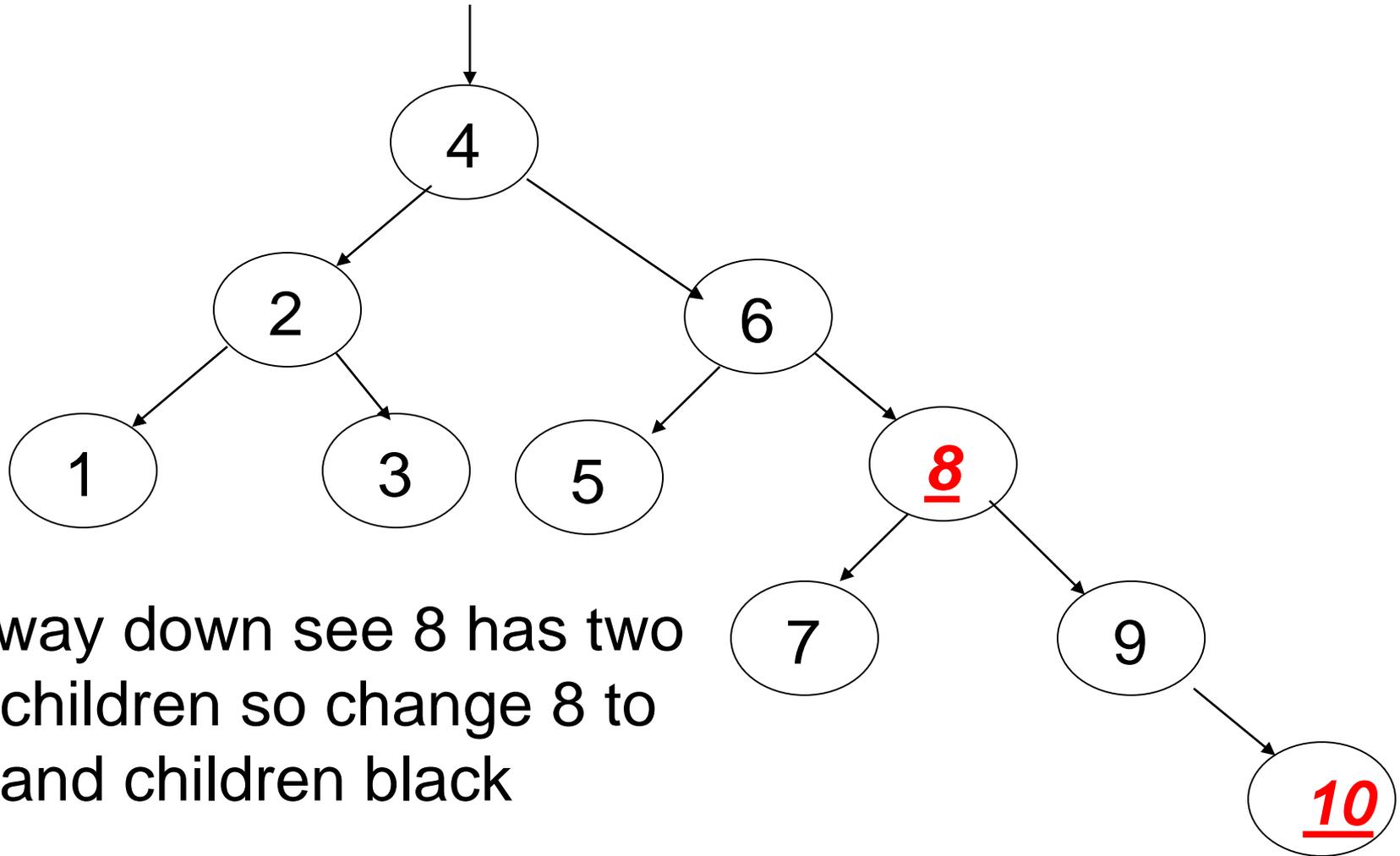
On way down see 8 has two red children so change 8 to red and children black

Insert 11
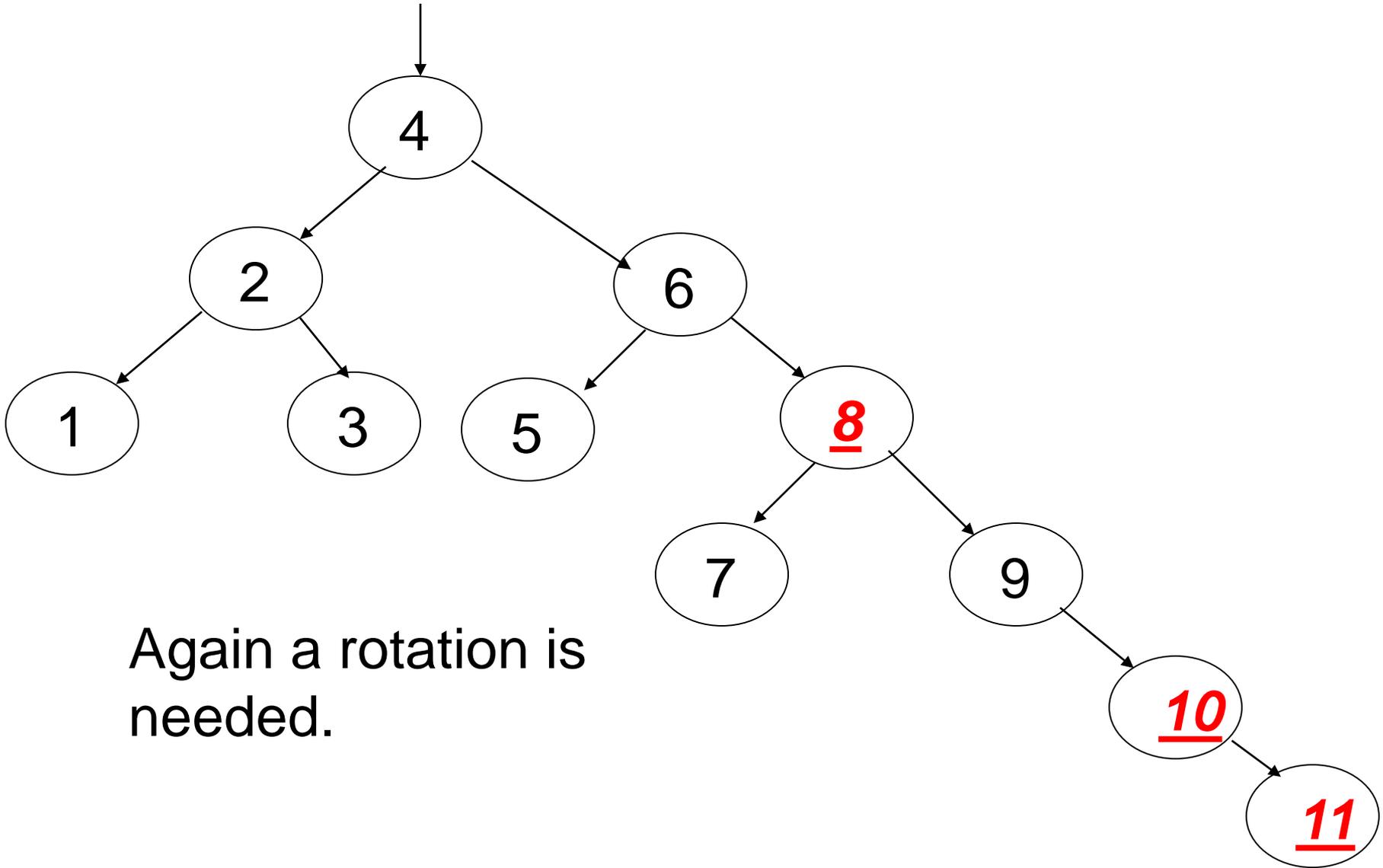
Again a rotation is needed.

Red Black Trees

Finish inserting 11


```
        4
       / \
      2   6
     / \ / \
    1  3 5  8
            / \
           7  10
              / \
             9  11
```

Red Black Trees