[TAP:JBFKC] Binary Search Tree

```
public boolean contains(E value){
   if (root.isEmpty()) return false;//1

   BinaryTree<E> possibleLocation = locate(root,value);//2
   return value.equals(possibleLocation.value());//3
}
```

- Here’s an implementation of contains(). Are there any errors in the code?
  A. Line 1
  B. Line 2
  C. Line 3
  [✓ D.] None
  E. Whatever

Today's Outline

- Binary Search Tree
  - Basics
  - Operations
  - Implementation
- Balanced Binary Search Trees
  - AVL Tree
  - RB Tree

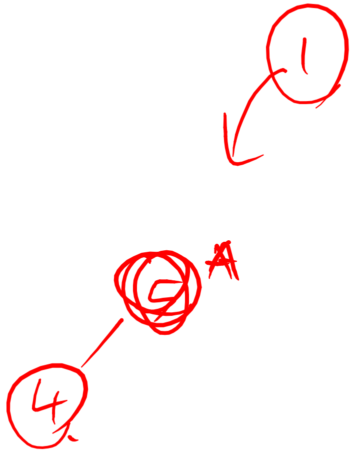
add(E value)

```
public void add(E value) {
    BinaryTree<E> node = new BinaryTree<E>(value, EMPTY, EMPTY);

    if(root.isEmpty())
        root = node;
    else {
        BinaryTree<E> loc = locate(root, value);

        E locValue = loc.value();

        if (ordering.compare(locValue, value) < 0)
            loc.setRight(node);
        else{
            if (loc.left().isEmpty())
                loc.setLeft(node);
        }   else
    }
                predecessor(loc).setRight(node);
}
    count++;
```



Predecessor

```
// return node with largest value in root’s left subtree
protected BinaryTree<E> predecessor(BinaryTree<E> root) {
    BinaryTree<E> result = root.left();
    while (!result.right().isEmpty())
        result = result.right();

    return result;
}
```

Today's Outline

- Binary Search Tree
  - Basics
  - Operations
  - Implementation
- Balanced Binary Search Trees
  - AVL Tree
  - Red-Black Tree

BST Observations

- The same data can be represented by many BST shapes
- Observations:
  - Additions to a BST happen at nodes missing at least one child
  - Removing from a BST can involve *any* node
  - Searching for a value in a BST takes time proportional to the height h of the tree

$\log n \le h \le n$

Shallow Binary Search Trees

- Strategy: Define a notion of “balance” and enforce balance via rotation.
- There are many strategies for tree balancing to preserve O(log n) height, including
  - AVL Trees: guaranteed O(log n) height
  - Red-black trees: guaranteed O(log n) height
  - B-trees (not binary): guaranteed O(log n) height
    - 2-3 trees, 2-3-4 trees, red-black 2-3-4 trees, ...
  - Splay trees: *Amortized* O(log n) time operations
  - Randomized trees: O(log n) expected height

AVL Trees

- An *AVL* Tree is a binary search tree in which every node is balanced (balance factor = 1, 0, or -1)

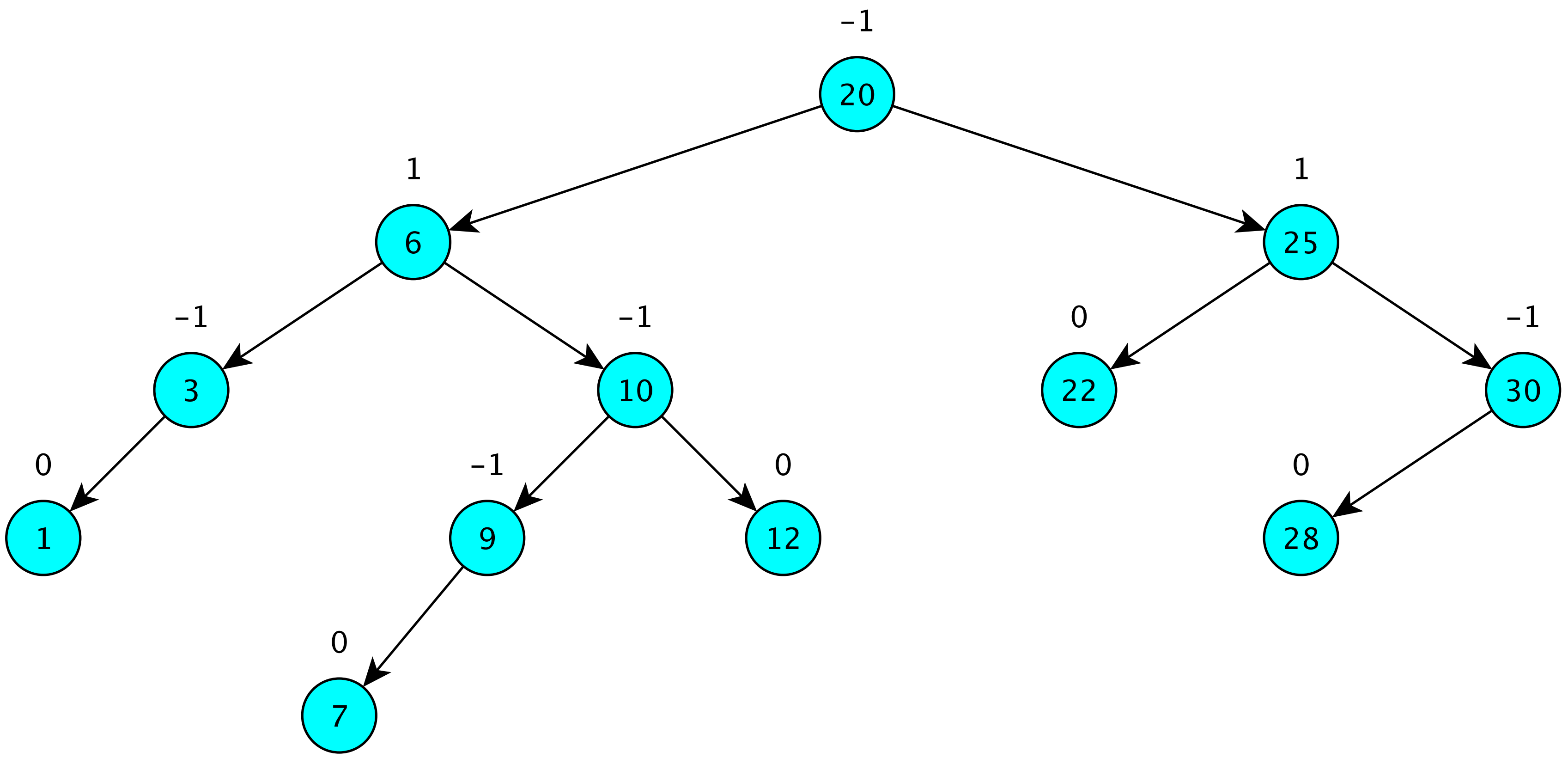
↑

height of right subtree

− left

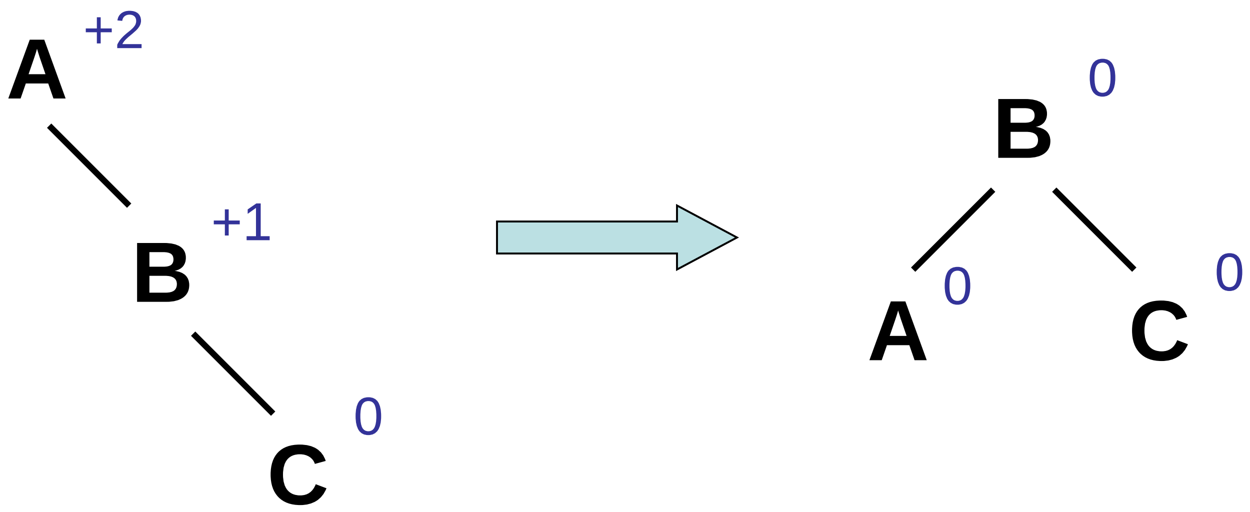
"subtrees of every node differ in height by at most 1"

AVL Trees



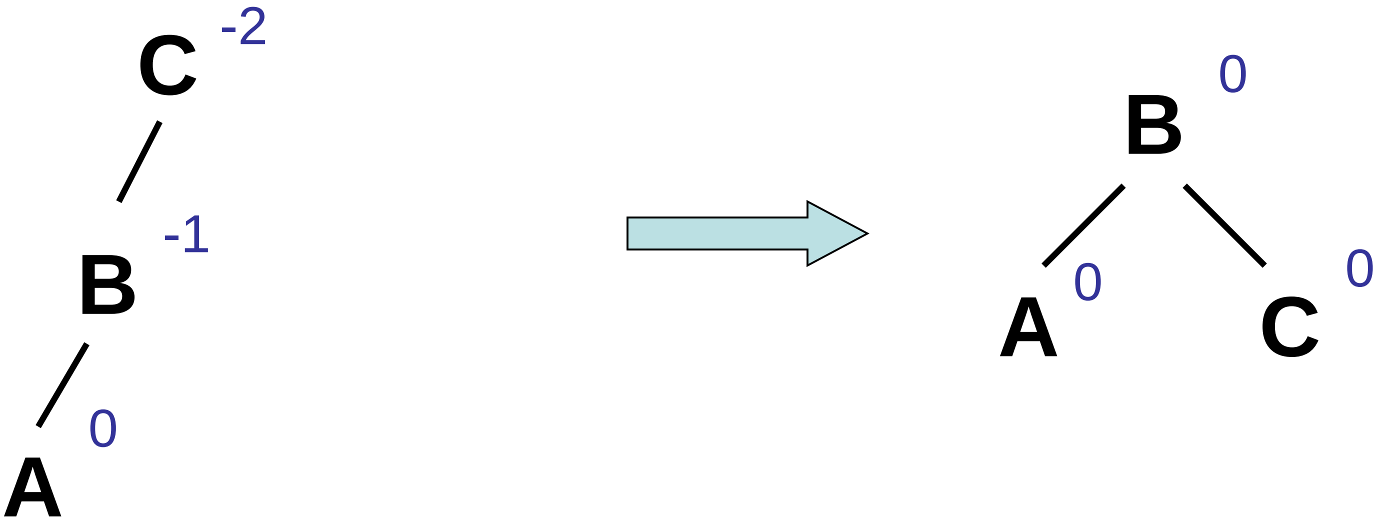
Single Rotation (Left)

LL Rotation



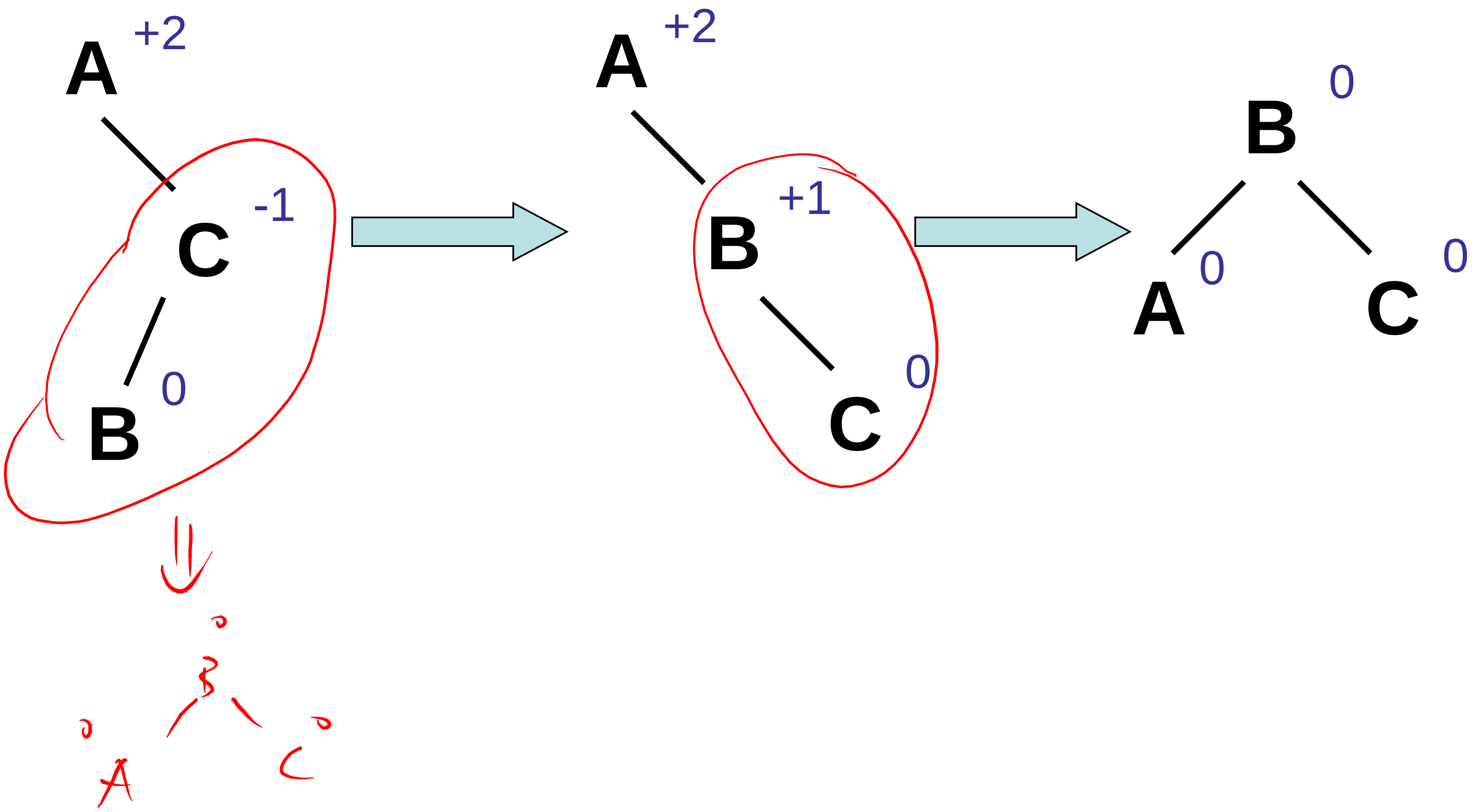
Single Rotation (Right)

RR rotation



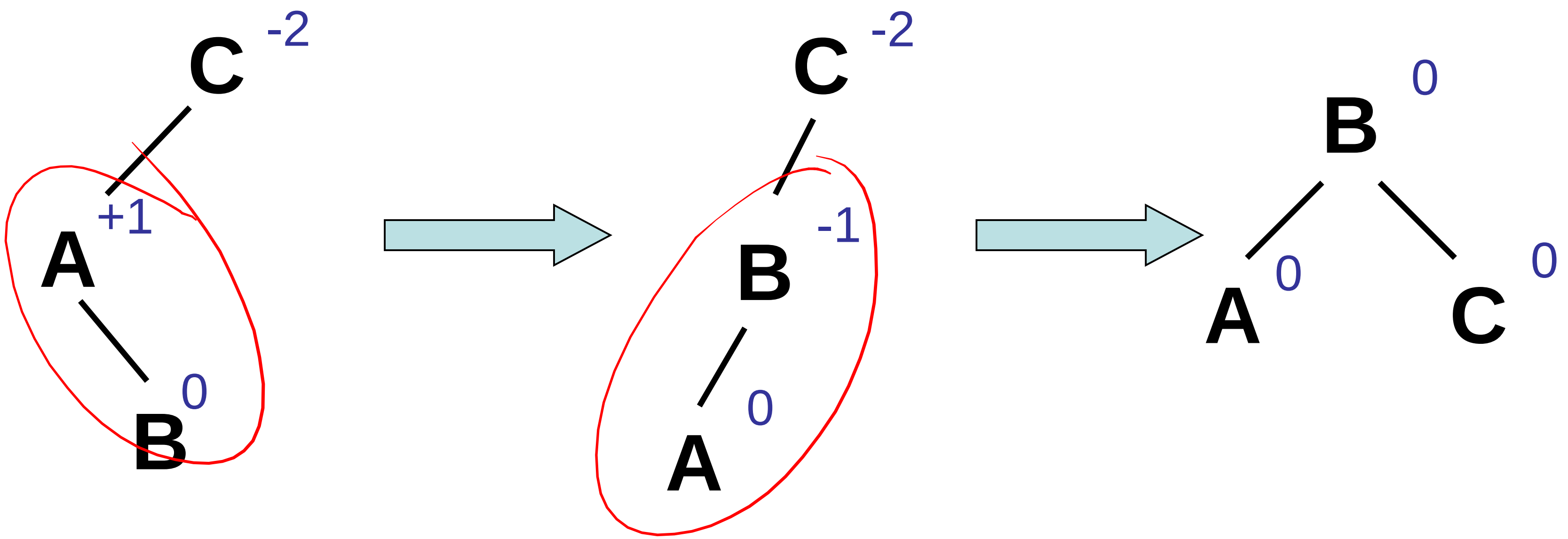
Double Rotation (Right-Left)

RL rotation

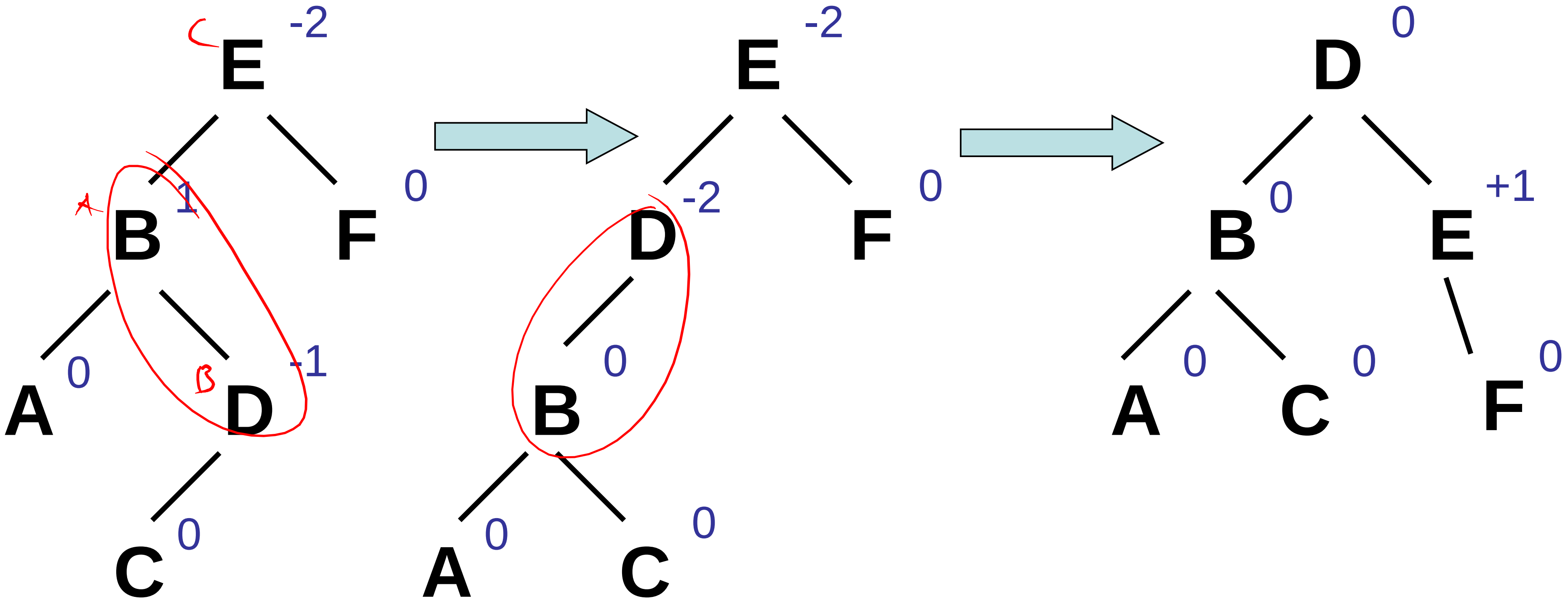


LR rotation

Double Rotation (Left-Right)



Double Rotation (Left-Right)



Today’s Outline

- Binary Search Tree
    - Basics
    - Operations
    - Implementation
- Balanced Binary Search Trees
    - AVL Tree
    - Red-Black Tree

Red-Black Trees

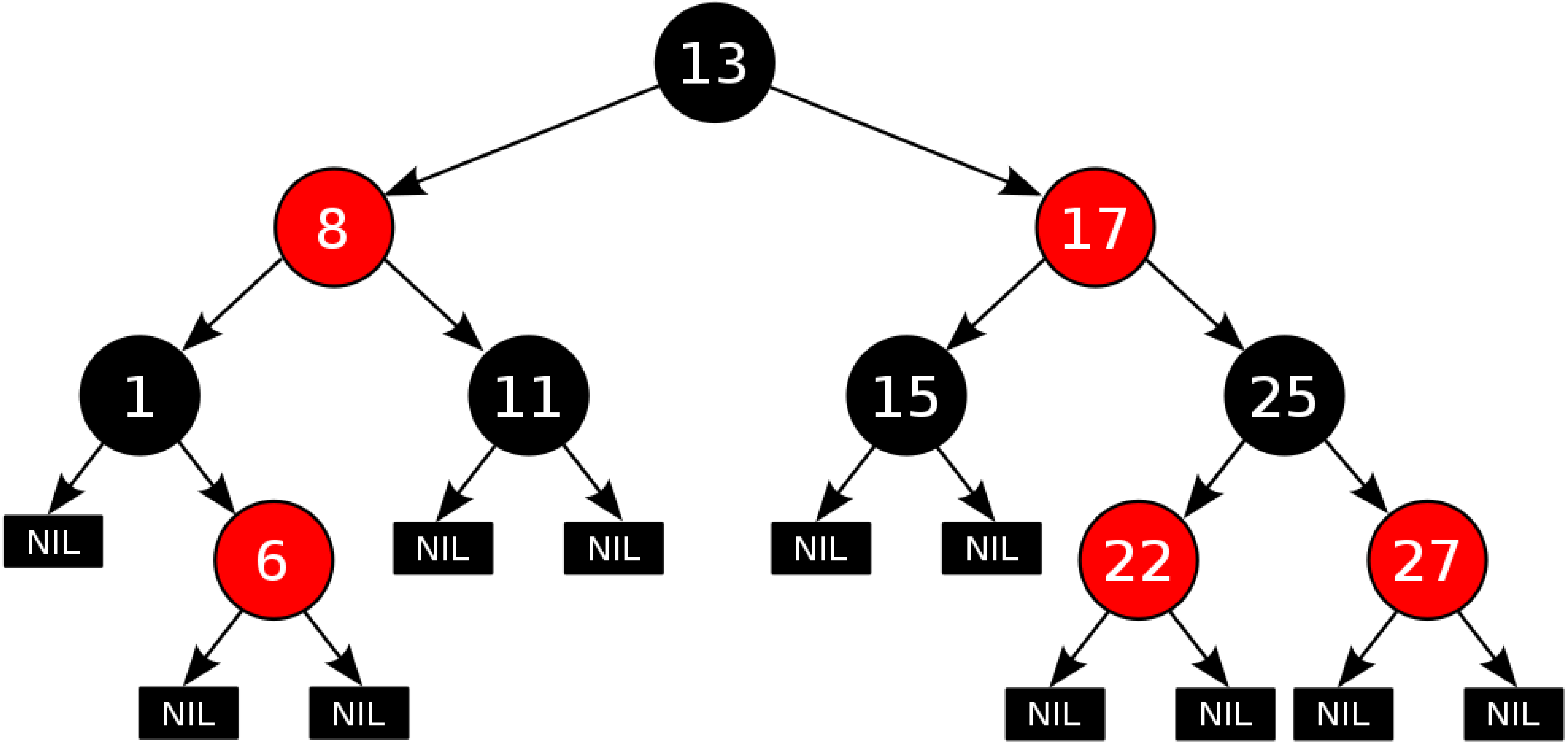
Red-Black tree is a binary search tree with the following characteristics

- Each node is colored *red* or *black*
- The following properties hold:
  - The root is black
  - The leaves (EMPTY) are black.
  - The children of red nodes are black
  - All paths from a given node to it's descendent leaves have the *same number* of black nodes

black height

A Red-Black Tree

(from Wikipedia.org)



Red-Black Tree Insertion

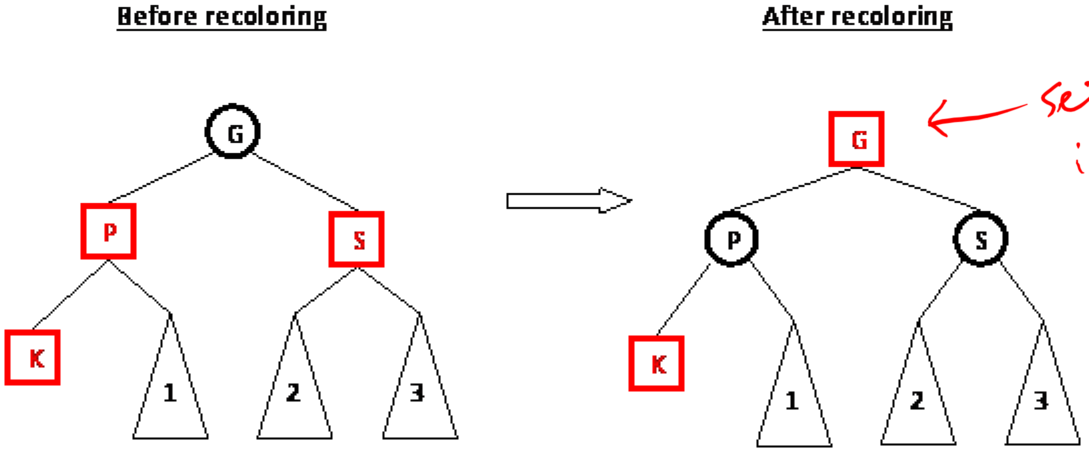
- Steps
  - Add node *k* to the tree
  - Color *k* red
  - Enforce Red-Black tree property
    - If *k*'s parent *p* is black

      do nothing
    - If *k*'s parent *p* is red

      do something

Red-Black Tree Insertion

- Case 1: P's sibling S is red

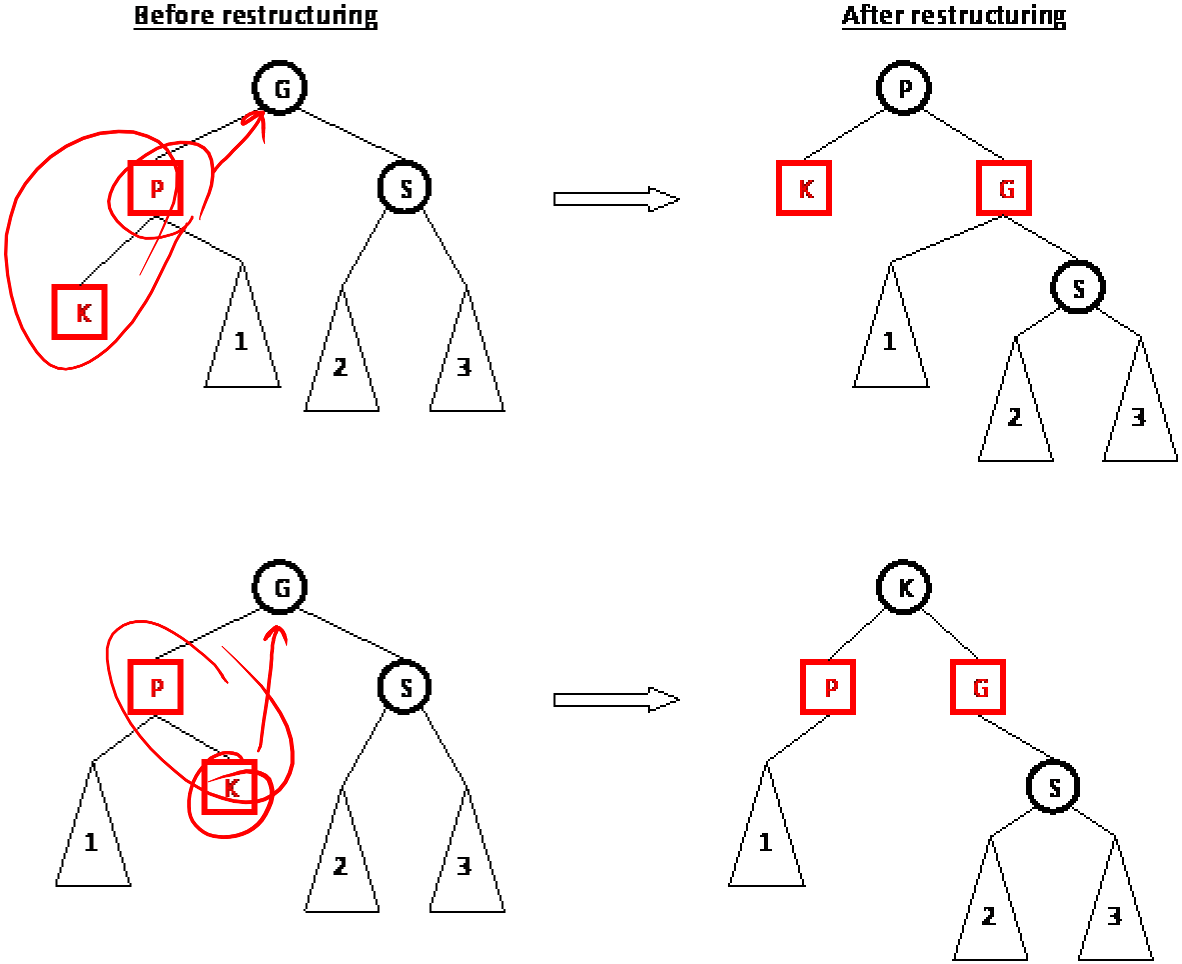


set this to black if G is the root

Credit: Paton, Wisc

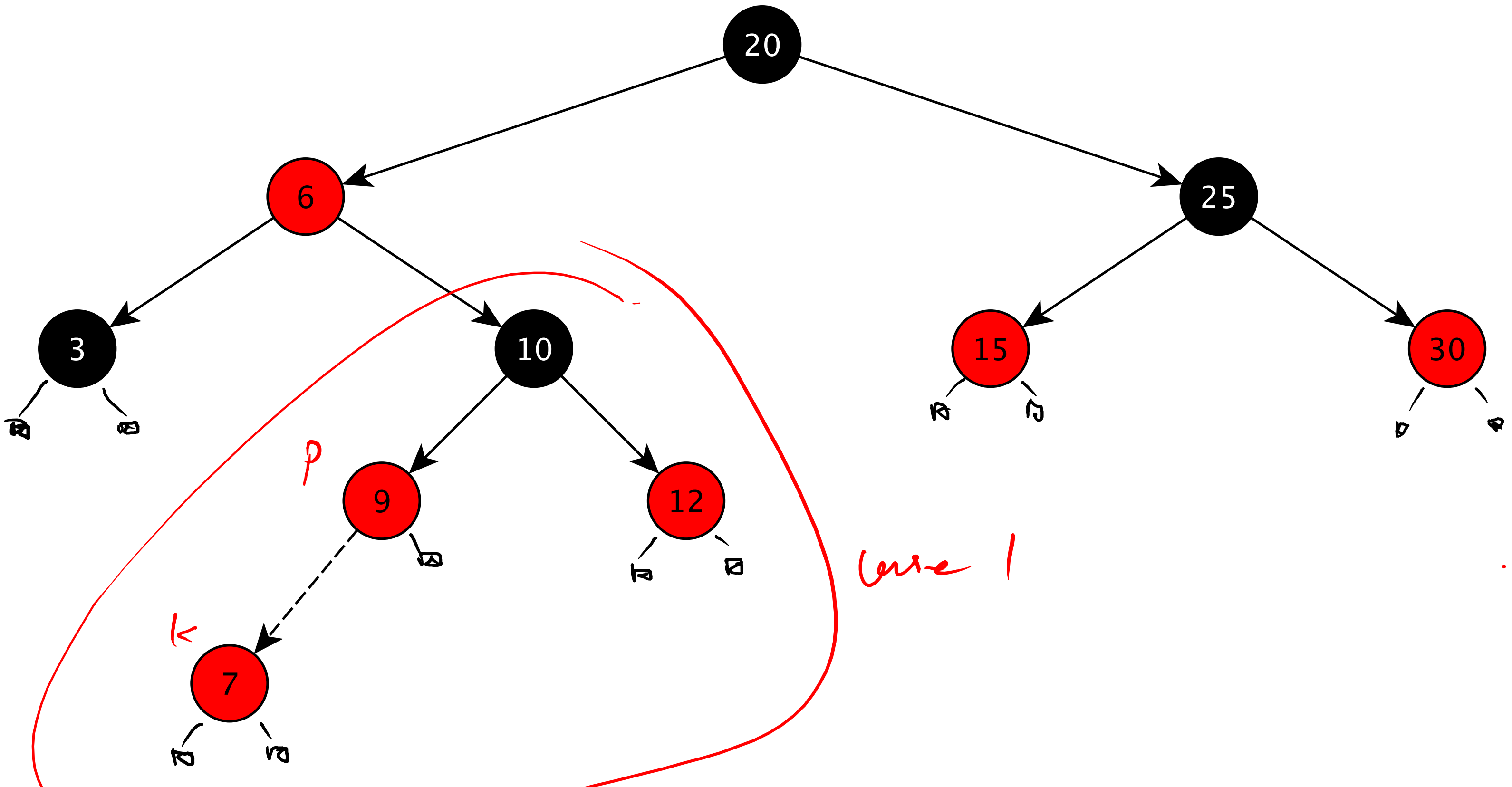
Red-Black Tree Insertion

- Case 2: P's sibling S is black



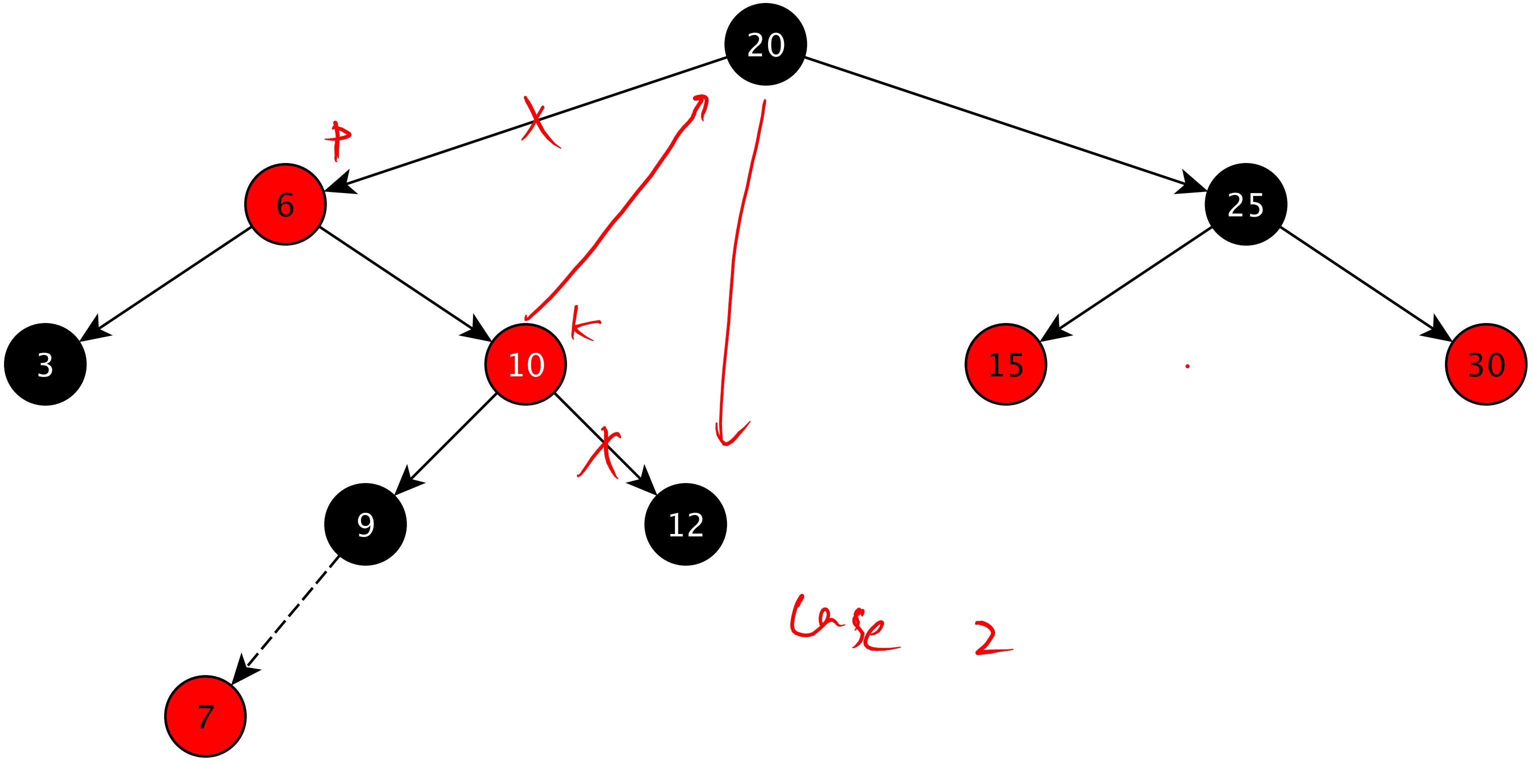
Credit: Paton, Wisc

Red-Black Tree Insertion



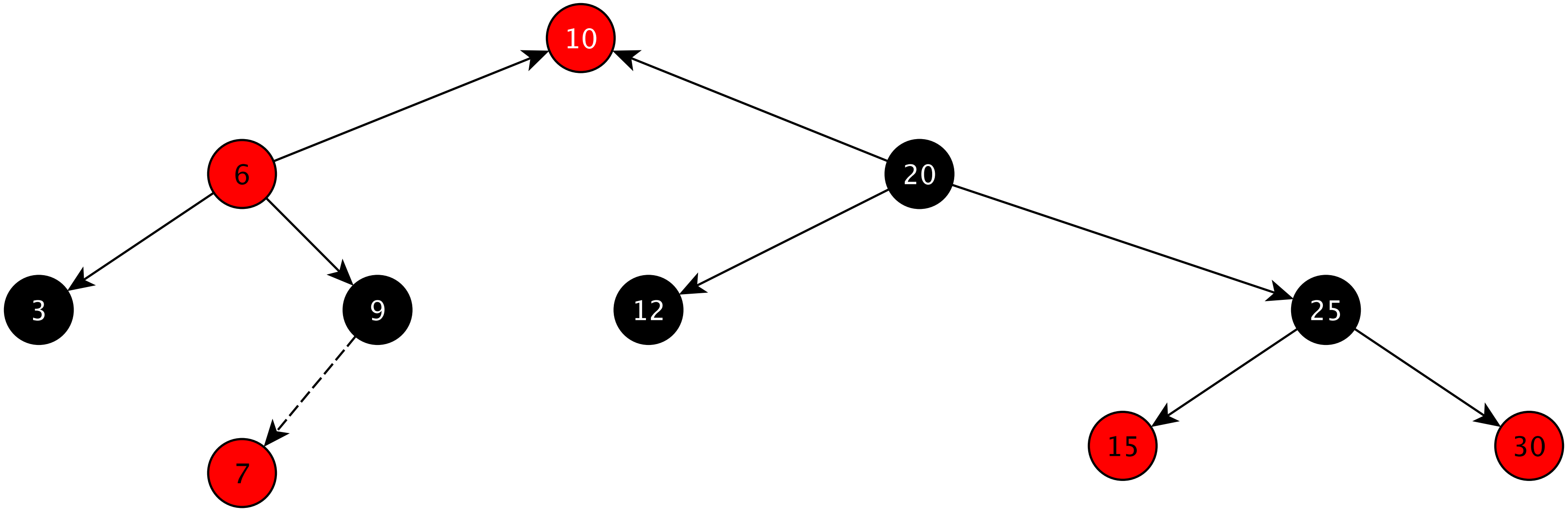
Black empty leaves not drawn. 7 just added Black-height still 2.

Red-Black Tree Insertion



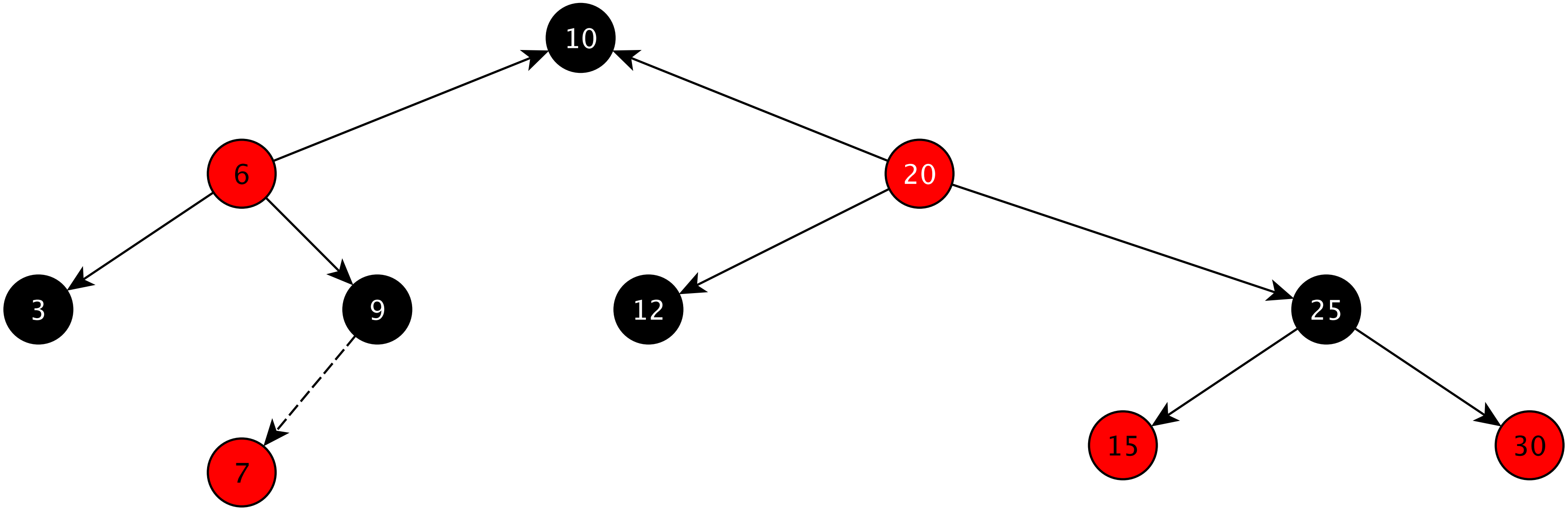
Black height still 2, color violation moved up

Red-Black Tree Insertion



Right rotation at 20, black height broken, need to recolor

Red-Black Tree Insertion



Color conditions restored, black-height restored.