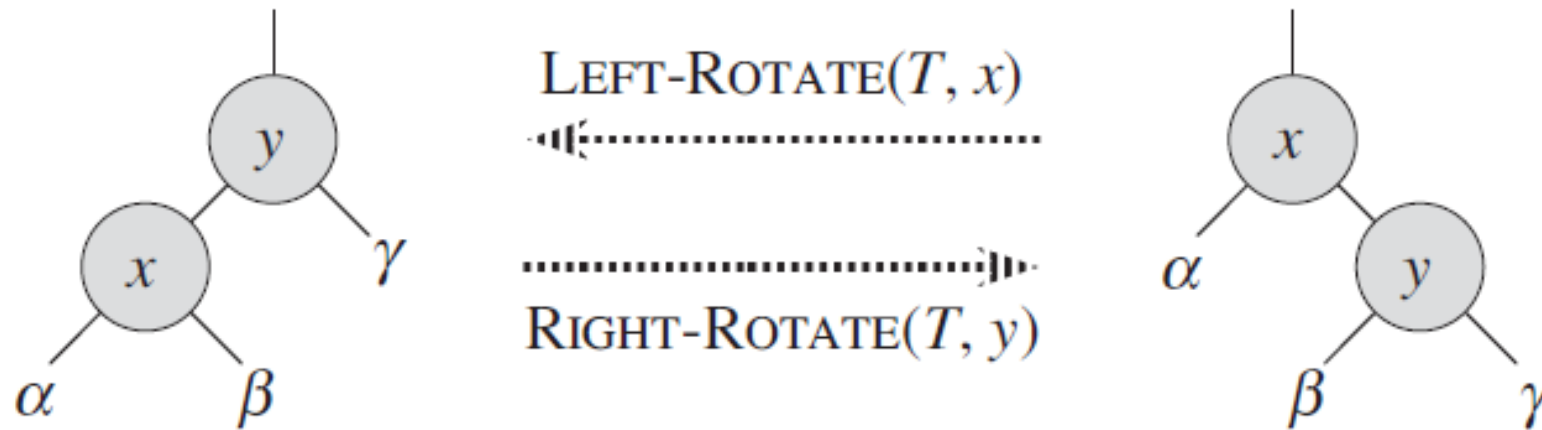


Lecture 20: Red-Black Trees II

2023/10/11

詹博华 (中国科学院软件研究所)

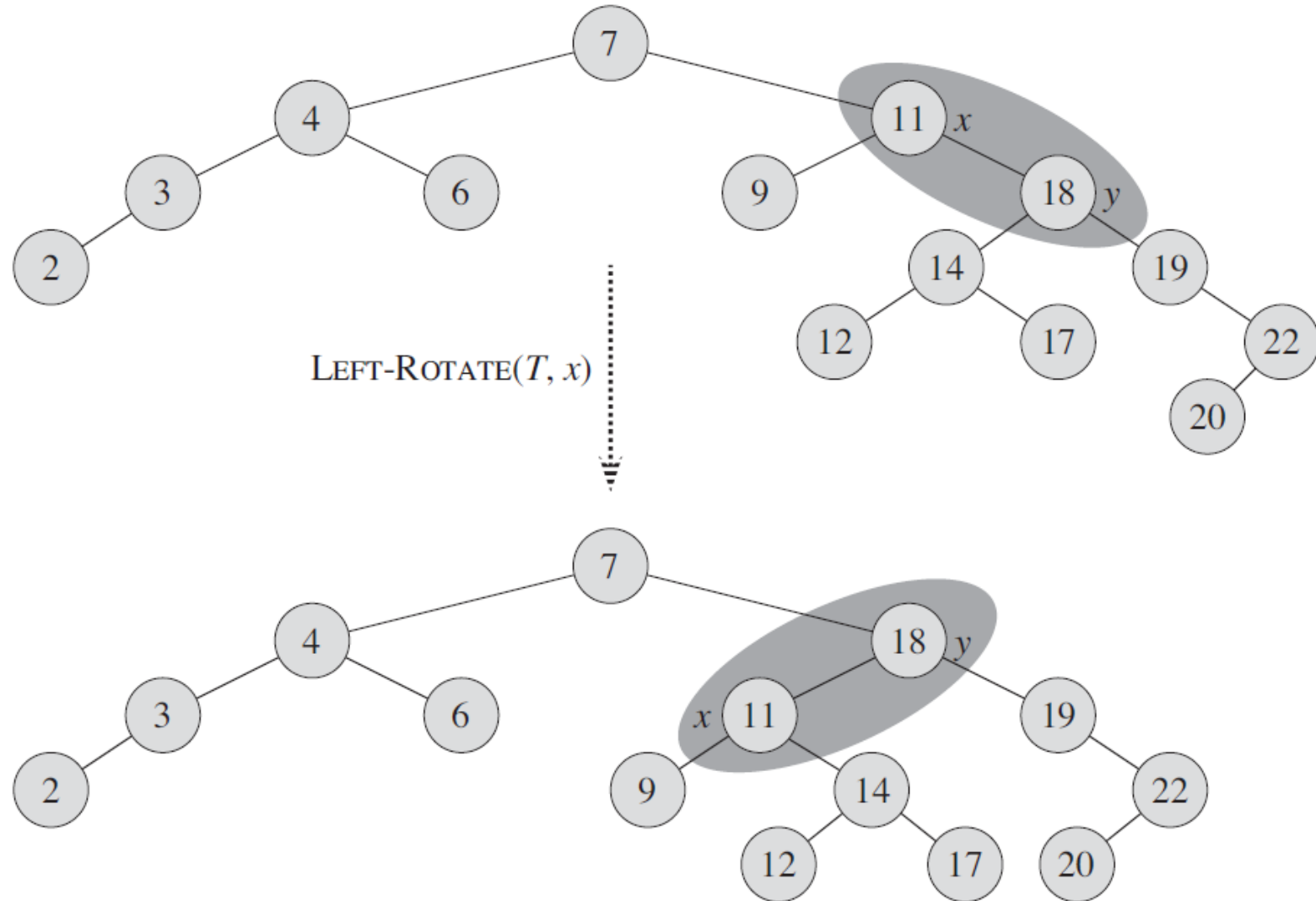
Rotations



Note:

- Invariants of binary search trees are maintained.
- What is really maintained is the **inorder traversal** of the tree.

Rotations in Practice



Rotation: implementation

LEFT-ROTATE(T, x)

```
1   $y = x.right$            // set y
2   $x.right = y.left$        // turn y's left subtree into x's right subtree
3  if  $y.left \neq T.nil$ 
4       $y.left.p = x$ 
5   $y.p = x.p$              // link x's parent to y
6  if  $x.p == T.nil$ 
7       $T.root = y$ 
8  elseif  $x == x.p.left$ 
9       $x.p.left = y$ 
10 else  $x.p.right = y$ 
11  $y.left = x$            // put x on y's left
12  $x.p = y$ 
```

Red-Black Trees: insertion

- Initial steps same as insertion for usual BST.
- The newly inserted node is colored **red**.
- Then, fixup is performed to recover the invariant.

Implementation

- Line 1-13: same as insertion in usual BST.
- Line 14-16: set fields of z .
- Line 17: fixup operations (to be expanded).

RB-INSERT(T, z)

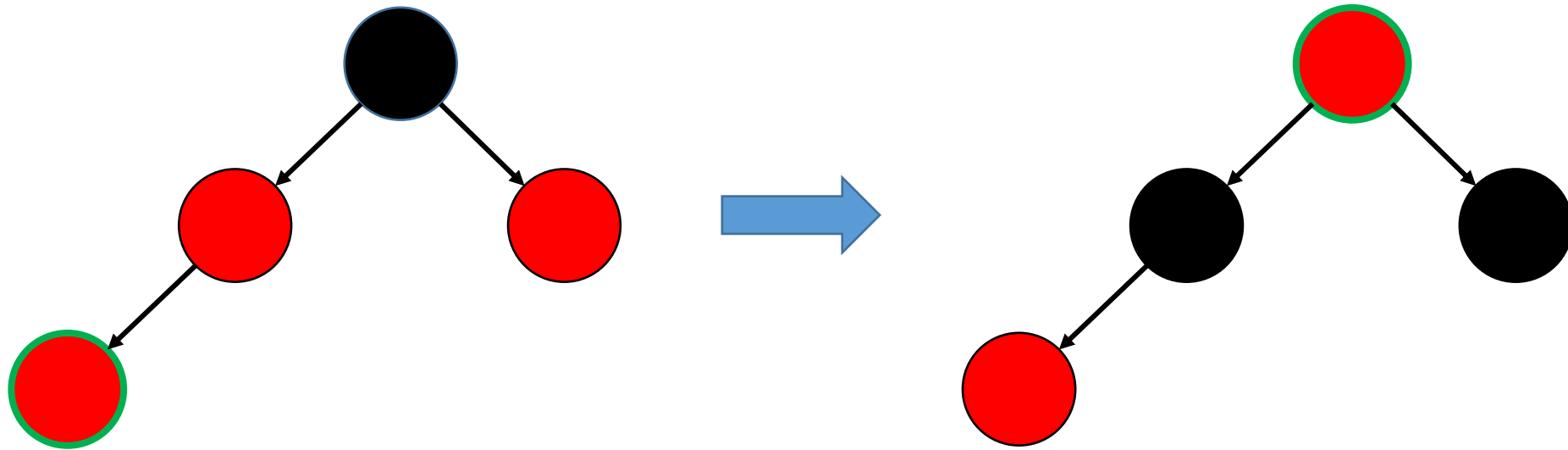
```
1   $y = T.nil$ 
2   $x = T.root$ 
3  while  $x \neq T.nil$ 
4       $y = x$ 
5      if  $z.key < x.key$ 
6           $x = x.left$ 
7      else  $x = x.right$ 
8   $z.p = y$ 
9  if  $y == T.nil$ 
10      $T.root = z$ 
11  elseif  $z.key < y.key$ 
12      $y.left = z$ 
13  else  $y.right = z$ 
14   $z.left = T.nil$ 
15   $z.right = T.nil$ 
16   $z.color = RED$ 
17  RB-INSERT-FIXUP( $T, z$ )
```

Fixup operation

- Starting from the bottom, where the new node is added.
- Proceed as long as there are two consecutive red nodes.
- Perform fixup so all invariant of the tree is maintained:
 1. No two consecutive reds.
 2. All paths have same number of black nodes.
 3. Order in binary search tree.
- Finally, if the root becomes red, recolor it black.

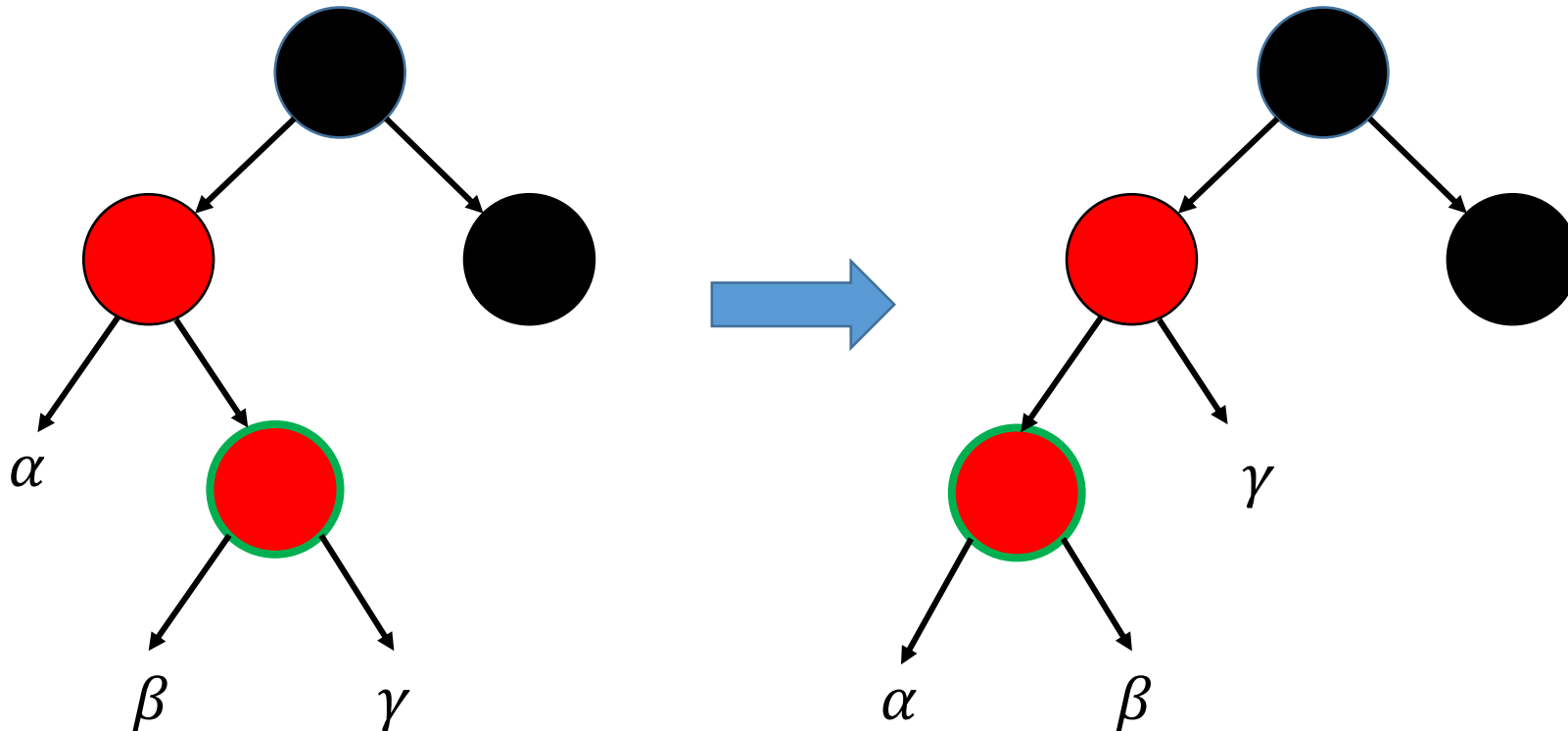
Fixup: case 1

- The uncle of the current node is also red.
 - Change parent and uncle to black, grandparent to red.



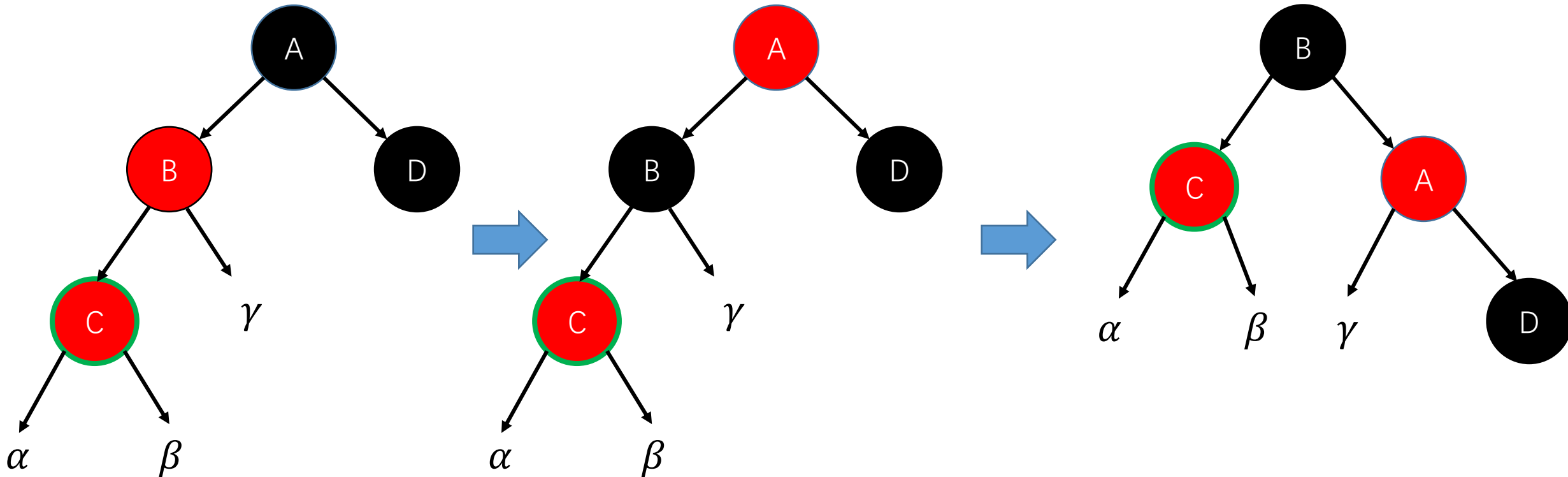
Fixup: case 2

- Uncle of current node is black. Current node is right child.
 - Perform left rotation on parent.



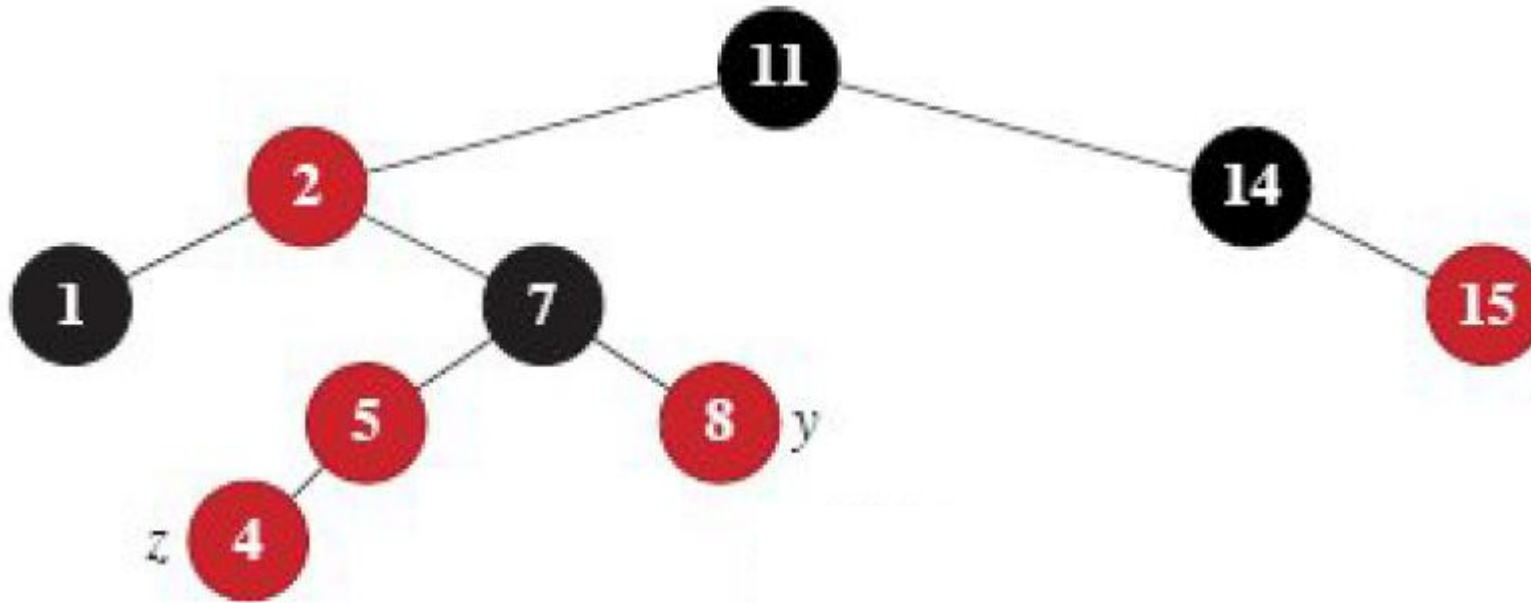
Fixup: case 3

- Uncle of current node is black. Current node is left child.
 - Recolor parent and grandparent.
 - Perform right rotation on grandparent.



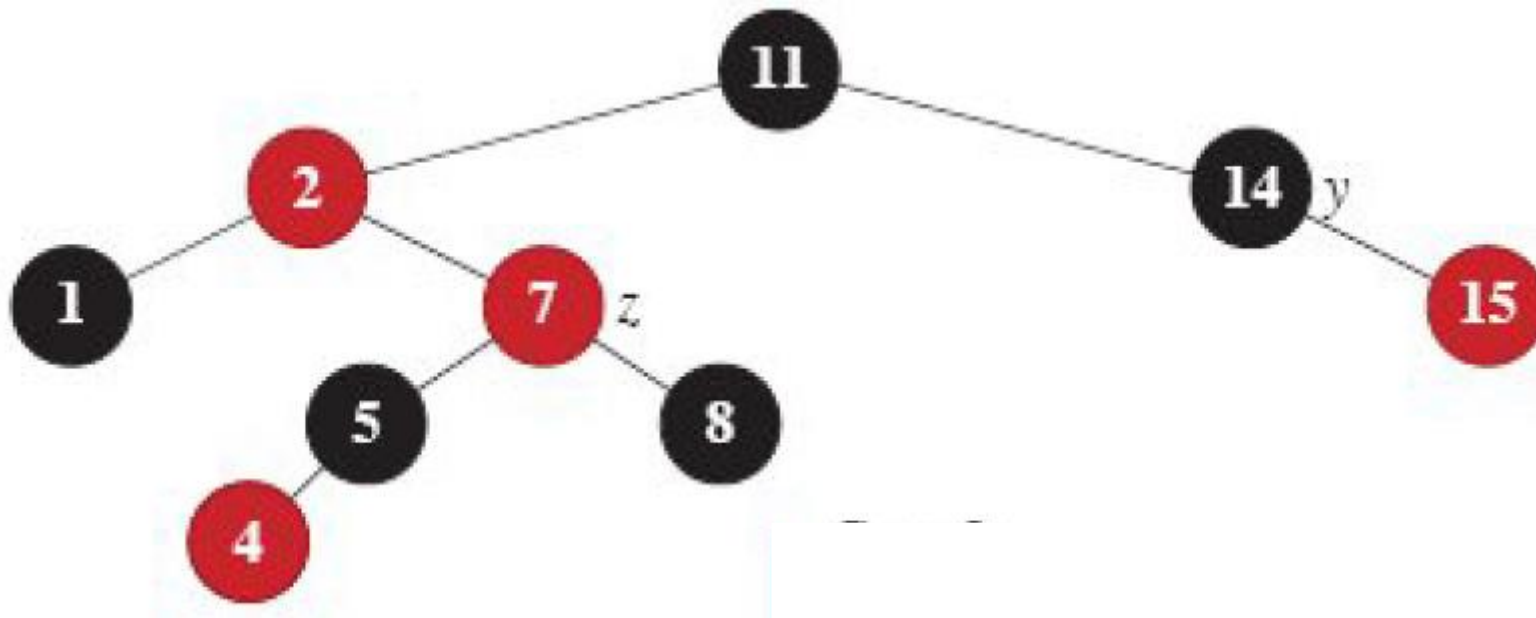
Concrete example

- Current node is z , its uncle y is red, so we are in case 1.
 - Recolor nodes 5, 7, and 8.



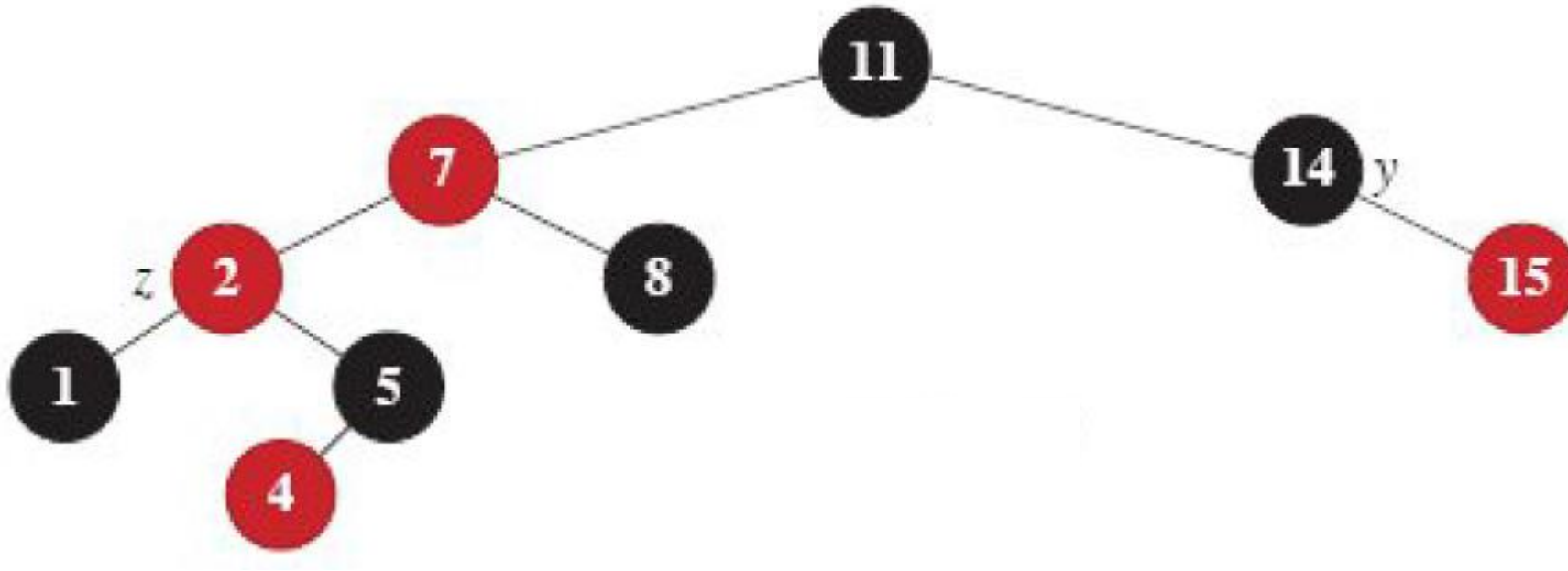
Concrete example

- Current node is z , its uncle y is black. z is right child, so case 2.
 - Perform left rotation.



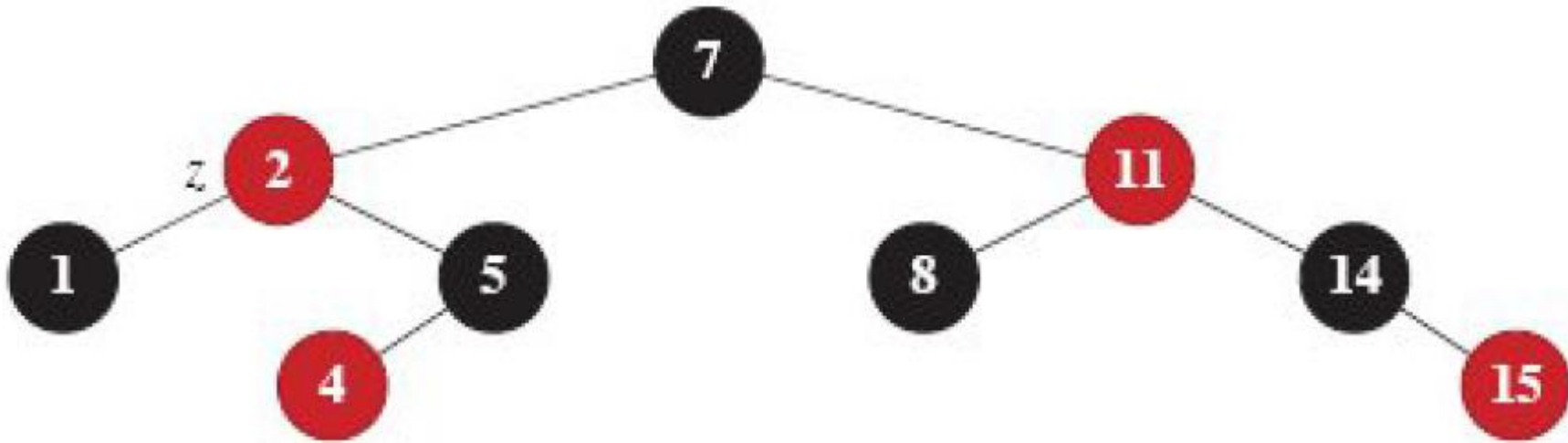
Concrete example

- Current node is z , its uncle y is black. z is left child, so case 3.
 - Recolor nodes 7 and 11, then perform right rotation.



Concrete example

- Final state.



Implementation

RB-INSERT-FIXUP(T, z)

```
1  while  $z.p.color == \text{RED}$ 
2      if  $z.p == z.p.p.left$ 
3           $y = z.p.p.right$ 
4          if  $y.color == \text{RED}$ 
5               $z.p.color = \text{BLACK}$                 // case 1
6               $y.color = \text{BLACK}$                 // case 1
7               $z.p.p.color = \text{RED}$                 // case 1
8               $z = z.p.p$                         // case 1
9          else if  $z == z.p.right$ 
10              $z = z.p$                             // case 2
11             LEFT-ROTATE( $T, z$ )                    // case 2
12              $z.p.color = \text{BLACK}$                 // case 3
13              $z.p.p.color = \text{RED}$                 // case 3
14             RIGHT-ROTATE( $T, z.p.p$ )                // case 3
15         else (same as then clause
                with “right” and “left” exchanged)
16      $T.root.color = \text{BLACK}$ 
```

Exercise (previous week)

- Consider inserting the keys 10, 22, 31, 4, 15, 28, 17, 88, 59 into a hash table of length $m = 11$ using open addressing with the auxiliary hash function $h'(k) = k$. Illustrate the result of inserting these keys using linear probing, using quadratic probing with $c_1 = 1$ and $c_2 = 3$, and using double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \bmod (m - 1))$.
- 考虑用开放寻址法将元素10, 22, 31, 4, 15, 28, 17, 88, 59 加入到长度为 $m = 11$ 的散列表中, 使用辅助散列函数 $h'(k) = k$ 。分别展示使用线性探查, 二次探查 ($c_1 = 1, c_2 = 3$) 和双重散列 ($h_1(k) = k, h_2(k) = 1 + (k \bmod (m - 1))$) 加入散列表的过程。

Exercise (this week)

- Construct red-black tree with insertion of values:
1, 2, 3, 4, 5, 6, 7, 8
- Remember:
 1. Nil nodes at leaf position (not shown) are black.
 2. Use symmetric version of rules if necessary.
 3. If root node becomes red, it is recolored black.