Red-Black Tree: Deletion

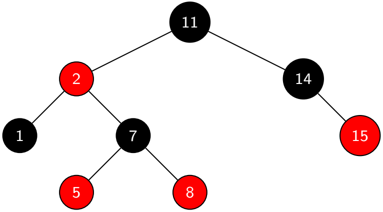
Subhabrata Samajder

IIIT, Delhi

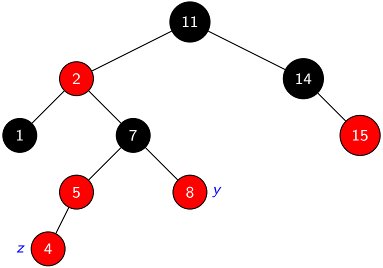
Summer Semester,
27$^{\text{th}}$ June, 2022

RBT: Insertion

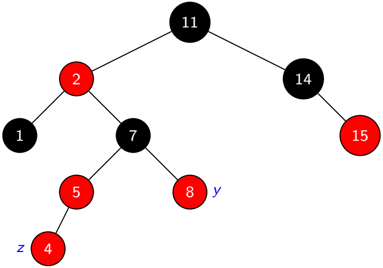
RB-INSERT-FIXUP(T, z) (Recap)



RB-INSERT-FIXUP(T, z) (Recap)



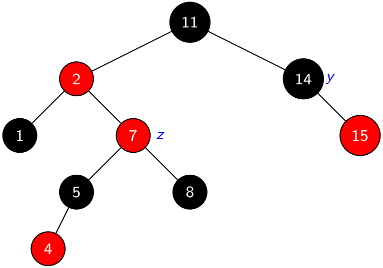
RB-INSERT-FIXUP(T, z) (Recap)



The following cases may arise:

- **Case 1:** z's uncle is red.

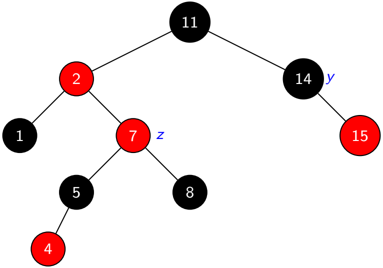
RB-INSERT-FIXUP(T, z) (Recap)



The following cases may arise:

- **Case 1:** z's uncle is red.

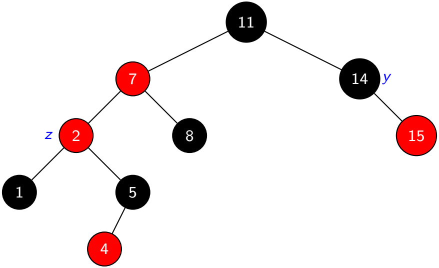
RB-INSERT-FIXUP(T, z) (Recap)



The following cases may arise:

- **Case 1:** z's uncle is red.
- **Case 2:** z's uncle is black and z is a right child.

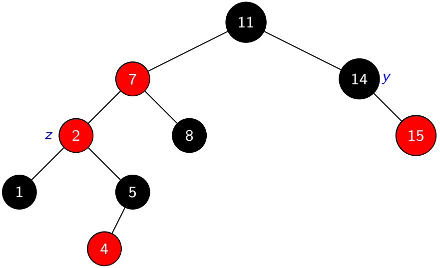
RB-INSERT-FIXUP(T, z) (Recap)



The following cases may arise:

- **Case 1:** z's uncle is red.
- **Case 2:** z's uncle is black and z is a right child.

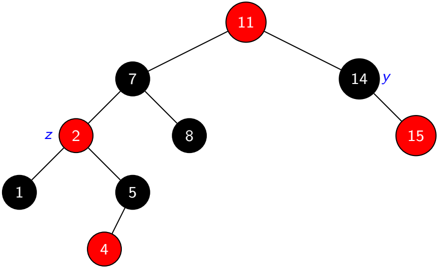
RB-INSERT-FIXUP(T, z) (Recap)



The following cases may arise:

- **Case 1:** z's uncle is red.
- **Case 2:** z's uncle is black and z is a right child.
- **Case 3:** z's uncle is black and z is a left child.

RB-INSERT-FIXUP(T, z) (Recap)

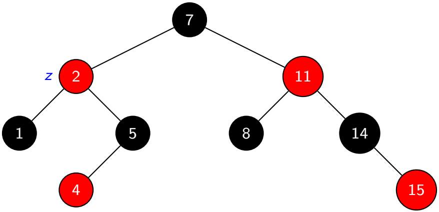


The following cases may arise:

- **Case 1:** z's uncle is red.
- **Case 2:** z's uncle is black and z is a right child.
- **Case 3:** z's uncle is black and z is a left child.

  Change Colours:
  - Properties 2 is violated.
  - Properties 4 is violated for the node 11.

RB-INSERT-FIXUP(T, z) (Recap)



The following cases may arise:

- **Case 1:** z's uncle is red.
- **Case 2:** z's uncle is black and z is a right child.
- **Case 3:** z's uncle is black and z is a left child.

Change Colours:

- Properties 2 is violated.
- Properties 4 is violated for the node 11.

Proof of Correctness

Follows from the following three part **loop-invariant**.

1. Node z is red.

2. If $p[z] = root[T]$, then $color[p[z]] = \text{BLACK}$.

3. There is at most one violation of either property 2 or 4.
   - Property 2 violation: z is the root and is red.
   - Property 4 violation: $colour[z] = colour[p[z]] = \text{RED}$.

Proof of Correctness

Initialization:

- Recall that z is the red node that was added.

- If $p[z] = root[T]$, then
  - $p[z]$ started out as black and
  - did not change prior to the call of RB-INSERT-FIXUP.

- Recall that properties 1, 3, and 5 holds when RB-INSERT-FIXUP is called.

Proof of Correctness

Initialization:

- **Property 2 violation:**
  - Root must be the newly added node.
  - Since the parent and both the children of z is black $\Rightarrow$ Property 4 is not simultaneously violated.
  - Implies property 2 is the only violation.

Proof of Correctness

Initialization:

- **Property 2 violation:**
  - Root must be the newly added node.
  - Since the parent and both the children of z is black $\Rightarrow$ Property 4 is not simultaneously violated.
  - Implies property 2 is the only violation.

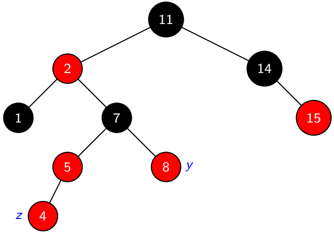
- **Property 4 violation:**
  - Note that the children of z are black.
  - Also, the tree had no prior violations before z was added.
  - Implies that $colour[z] = colour[p[z]] = \mathrm{RED}$.
  - Also, $root \neq p[z]$ as $colour[p[z]] = \mathrm{RED}$.

Proof of Correctness

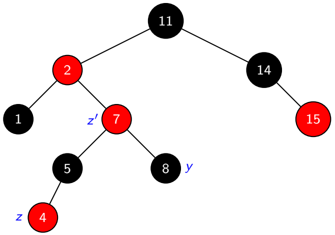
Termination:

- At termination $p[z] = \text{BLACK}$.
- Thus property 4 is not violated.
- Only violation can be to property 2.
- But Line 16 rectifies it.

Proof of Correctness: Maintenance



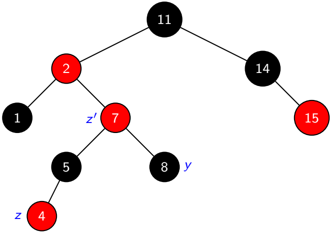
Proof of Correctness: Maintenance



Case 1: z's uncle is red.

- Let $z' = p[p[z]]$ denote the value of z in the next iteration.
- $colour[z'] = \text{RED}$ at the start of next iteration.
- Colour of $p[z'] = p[p[p[z]]]$ does not change.
- If $p[z'] = root$, then the colour of $root$ was black and remains so.

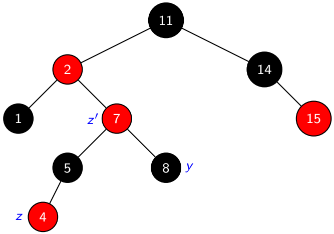
Proof of Correctness: Maintenance



Case 1: z's uncle is red.

- If $z' = root$, then Case 1 corrected the only violation of property 4.
- $colour[z'] = \text{RED}$ and $z' = root \Rightarrow$ property 2 is the only violation.

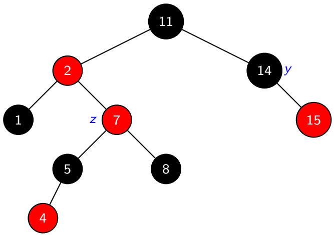
Proof of Correctness: Maintenance



Case 1: z's uncle is red.

- If $z' \neq root$, then Case 1 has not corrected any violation of property 2.
- It corrected the lone violation of property 4.
- It then made $colour[z'] = \text{RED}$ and left $p[z']$ alone.
- If $colour[p[z']] = \text{BLACK}$, there is no violation of property 4.
- If $colour[p[z']] = \text{RED}$, coloring z' red creates one violation of property 4 between z' and $p[z']$.

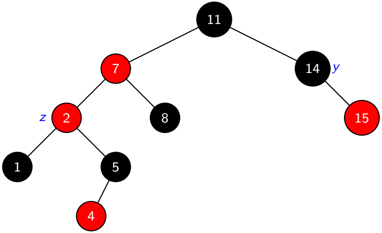
Proof of Correctness: Maintenance



Case 2: z's uncle is black and z is a right child.

- Makes z point to $p[z]$, which is red.
- No further change to z or its color occurs in cases 2 and 3.

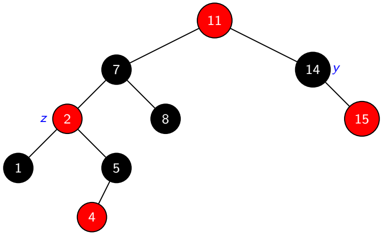
Proof of Correctness: Maintenance



Case 2: z's uncle is black and z is a right child.

- Makes z point to $p[z]$, which is red.
- No further change to z or its color occurs in cases 2 and 3.

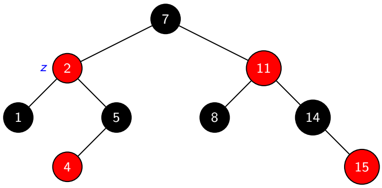
Proof of Correctness: Maintenance



Case 2: z's uncle is black and z is a right child.
Case 3: z's uncle is black and z is a left child.

- Makes $p[z]$ black.
- If $p[z] = root$ at the start of the next iteration, then it is black.

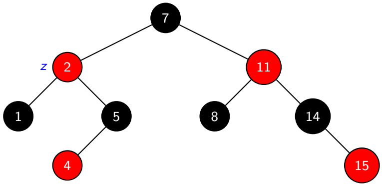
Proof of Correctness: Maintenance



Case 2: z's uncle is black and z is a right child.
Case 3: z's uncle is black and z is a left child.

- Makes $p[z]$ black.
- If $p[z] = root$ at the start of the next iteration, then it is black.

Proof of Correctness: Maintenance

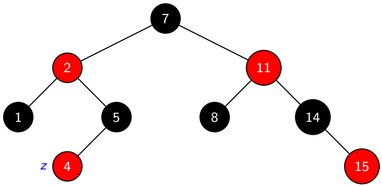


Case 2: z's uncle is black and z is a right child.
Case 3: z's uncle is black and z is a left child.

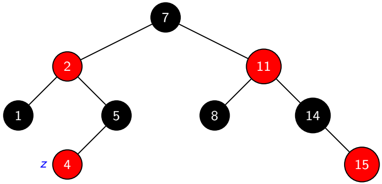
- As in Case 1, properties 1, 3, and 5 are maintained in Cases 2 and 3.
- Since node $z \neq root$ in cases 2 and 3 $\Rightarrow$ no violation of property 2.
- Also, Cases 2 and 3 do not introduce a violation of property 2, since the only node that is made red becomes a child of a black node by the rotation in case 3.
- Cases 2 and 3 correct the lone violation of property 4, and they do not introduce another violation.

RBT: Deletion

RB-DELETE(T, z)

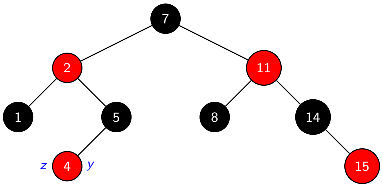


RB-DELETE(T, z)



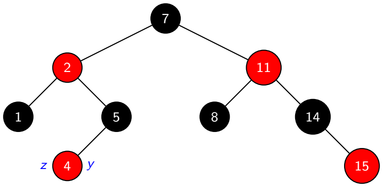
1. **if** $left[z] = nil[T]$ or $right[z] = nil[T]$

RB-DELETE(T, z)



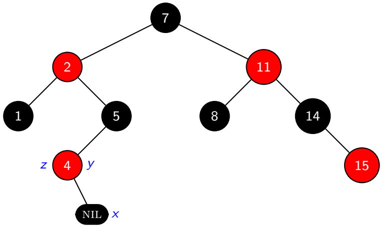
1. **if** $left[z] = nil[T]$ or $right[z] = nil[T]$
2. $\quad y \leftarrow z$

RB-DELETE(T, z)



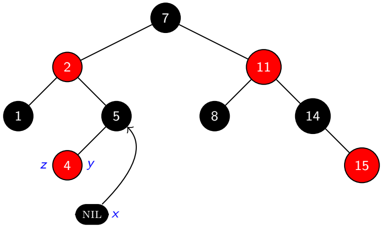
1. **if** $left[z] = nil[T]$ or $right[z] = nil[T]$
2. $\quad y \leftarrow z$
3. **if** $left[y] \neq nil[T]$

RB-DELETE(T, z)



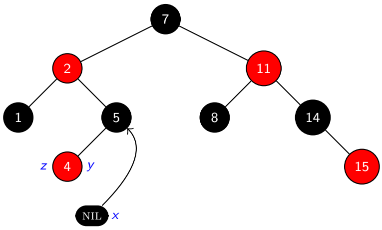
1. **if** $left[z] = nil[T]$ or $right[z] = nil[T]$
2. $\quad y \leftarrow z$
3. **if** $left[y] \neq nil[T]$
4. **else**
5. $\quad x \leftarrow right[y]$

RB-DELETE(T, z)



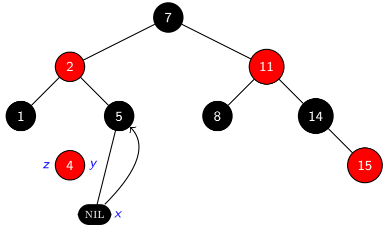
1. **if** $left[z] = nil[T]$ or $right[z] = nil[T]$
2. $\quad y \leftarrow z$
3. **if** $left[y] \neq nil[T]$
4. **else**
5. $\quad x \leftarrow right[y]$
6. $p[x] \leftarrow p[y]$

RB-DELETE(T, z)



1. **if** $left[z] = nil[T]$ or $right[z] = nil[T]$
2. $\quad y \leftarrow z$
3. **if** $left[y] \neq nil[T]$
4. **else**
5. $\quad x \leftarrow right[y]$
6. $p[x] \leftarrow p[y]$
7. **if** $p[y] = nil[T]$

RB-DELETE(T, z)



if $left[z] = nil[T]$ or $right[z] = nil[T]$

    $y \leftarrow z$

if $left[y] \neq nil[T]$

else

    $x \leftarrow right[y]$

$p[x] \leftarrow p[y]$

if $p[y] = nil[T]$

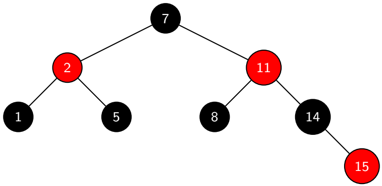
else if $y = left[p[y]]$

    $left[p[y]] \leftarrow x$

else

    $right[p[y]] \leftarrow x$

RB-DELETE(T, z)



if $\mathit{left}[z] = \mathit{nil}[T]$ or $\mathit{right}[z] = \mathit{nil}[T]$

  $y \leftarrow z$

if $\mathit{left}[y] \neq \mathit{nil}[T]$

else

  $x \leftarrow \mathit{right}[y]$

$p[x] \leftarrow p[y]$

if $p[y] = \mathit{nil}[T]$

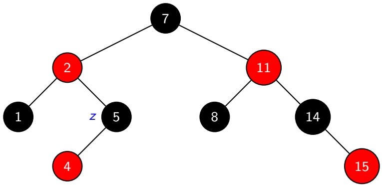
else if $y = \mathit{left}[p[y]]$

  $\mathit{left}[p[y]] \leftarrow x$

else

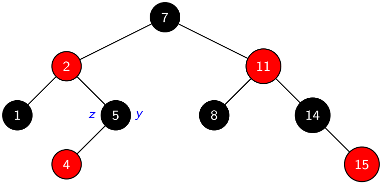
  $\mathit{right}[p[y]] \leftarrow x$

RB-DELETE(T, z)



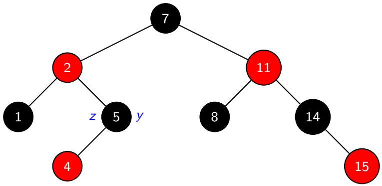

```
if left[z] = nil[T] or right[z] = nil[T]
    y ← z
if left[y] ≠ nil[T]
else
    x ← right[y]
p[x] ← p[y]
if p[y] = nil[T]
else if y = left[p[y]]
    left[p[y]] ← x
else
    right[p[y]] ← x
```

RB-DELETE(T, z)



if $left[z] = nil[T]$ or $right[z] = nil[T]$
    $y \leftarrow z$
if $left[y] \neq nil[T]$
else
    $x \leftarrow right[y]$
$p[x] \leftarrow p[y]$
if $p[y] = nil[T]$
else if $y = left[p[y]]$
    $left[p[y]] \leftarrow x$
else
    $right[p[y]] \leftarrow x$

RB-DELETE(T, z)



if $left[z] = nil[T]$ or $right[z] = nil[T]$

$\quad y \leftarrow z$

if $left[y] \neq nil[T]$

else

$\quad x \leftarrow right[y]$

$p[x] \leftarrow p[y]$

if $p[y] = nil[T]$

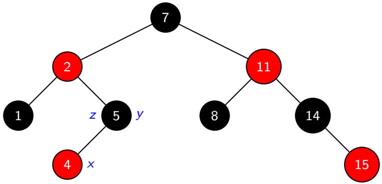
else if $y = left[p[y]]$

$\quad left[p[y]] \leftarrow x$

else

$\quad right[p[y]] \leftarrow x$

RB-DELETE(T, z)



if $left[z] = nil[T]$ or $right[z] = nil[T]$

    $y \leftarrow z$

if $left[y] \neq nil[T]$

    $x \leftarrow left[y]$

else

    $x \leftarrow right[y]$

$p[x] \leftarrow p[y]$

if $p[y] = nil[T]$

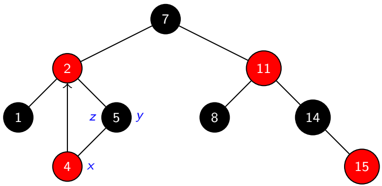
else if $y = left[p[y]]$

    $left[p[y]] \leftarrow x$

else

    $right[p[y]] \leftarrow x$

RB-DELETE(T, z)



if $left[z] = nil[T]$ or $right[z] = nil[T]$

  $y \leftarrow z$

if $left[y] \neq nil[T]$

  $x \leftarrow left[y]$

else

  $x \leftarrow right[y]$

$p[x] \leftarrow p[y]$

if $p[y] = nil[T]$

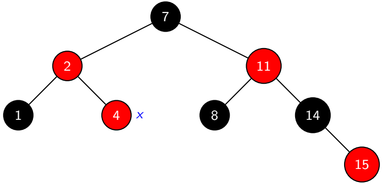
else if $y = left[p[y]]$

  $left[p[y]] \leftarrow x$

else

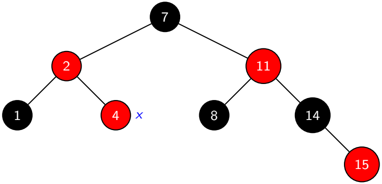
  $right[p[y]] \leftarrow x$

RB-Delete(T, z)



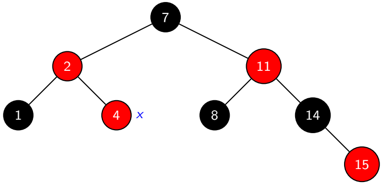

```
if left[z] = nil[T] or right[z] = nil[T]
    y ← z
if left[y] ≠ nil[T]
    x ← left[y]
else
    x ← right[y]
p[x] ← p[y]
if p[y] = nil[T]
else if y = left[p[y]]
    left[p[y]] ← x
else
    right[p[y]] ← x
```

RB-DELETE(T, z)




```
if left[z] = nil[T] or right[z] = nil[T]
    y ← z
if left[y] ≠ nil[T]
    x ← left[y]
else
    x ← right[y]
p[x] ← p[y]
if p[y] = nil[T]
else if y = left[p[y]]
    left[p[y]] ← x
else
    right[p[y]] ← x
if colour[y] = BLACK
```

RB-DELETE(T, z)



if $left[z] = nil[T]$ or $right[z] = nil[T]$

   $y \leftarrow z$

if $left[y] \neq nil[T]$

   $x \leftarrow left[y]$

else

   $x \leftarrow right[y]$

$p[x] \leftarrow p[y]$

if $p[y] = nil[T]$

else if $y = left[p[y]]$

   $left[p[y]] \leftarrow x$

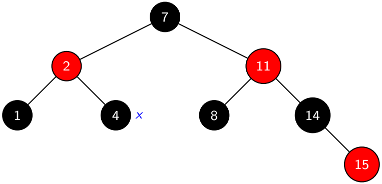
else

   $right[p[y]] \leftarrow x$

if $colour[y] = $ BLACK

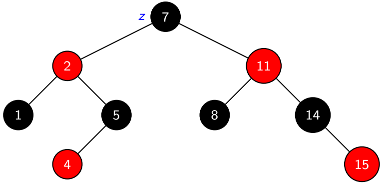
   RB-DELETE-FIXUP(T, x)

RB-DELETE-FIXUP(T, x)



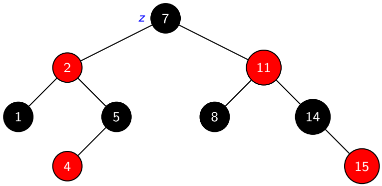
1. $colour[x] \leftarrow$ BLACK

RB-DELETE(T, z)



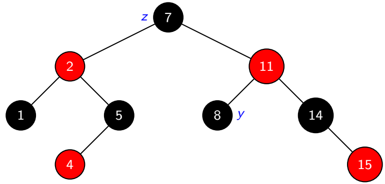

```
if left[z] = nil[T] or right[z] = nil[T]
    y ← z
if left[y] ≠ nil[T]
    x ← left[y]
else
    x ← right[y]
p[x] ← p[y]
if p[y] = nil[T]
else if y = left[p[y]]
    left[p[y]] ← x
else
    right[p[y]] ← x
if colour[y] = BLACK
    RB-DELETE-FIXUP(T, x)
```

RB-DELETE(T, z)



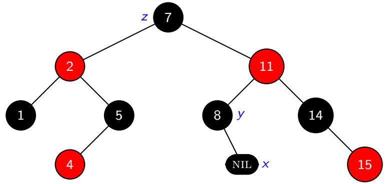

```
if left[z] = nil[T] or right[z] = nil[T]
    y ← z
if left[y] ≠ nil[T]
    x ← left[y]
else
    x ← right[y]
p[x] ← p[y]
if p[y] = nil[T]
else if y = left[p[y]]
    left[p[y]] ← x
else
    right[p[y]] ← x
if colour[y] = BLACK
    RB-DELETE-FIXUP(T, x)
```

RB-DELETE(T, z)



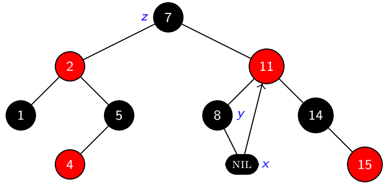
if $left[z] = nil[T]$ or $right[z] = nil[T]$
    $y \leftarrow z$
else
    $y \leftarrow$ TREE-SUCCESSOR(z)
if $left[y] \neq nil[T]$
    $x \leftarrow left[y]$
else
    $x \leftarrow right[y]$
$p[x] \leftarrow p[y]$
if $p[y] = nil[T]$
else if $y = left[p[y]]$
    $left[p[y]] \leftarrow x$
else
    $right[p[y]] \leftarrow x$
if $colour[y] =$ BLACK
    RB-DELETE-FIXUP(T, x)

RB-DELETE(T, z)



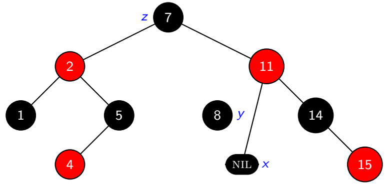
if $left[z] = nil[T]$ or $right[z] = nil[T]$
    $y \leftarrow z$
else
    $y \leftarrow$ TREE-SUCCESSOR(z)
if $left[y] \neq nil[T]$
    $x \leftarrow left[y]$
else
    $x \leftarrow right[y]$
$p[x] \leftarrow p[y]$
if $p[y] = nil[T]$
else if $y = left[p[y]]$
    $left[p[y]] \leftarrow x$
else
    $right[p[y]] \leftarrow x$
if $colour[y] =$ BLACK
    RB-DELETE-FIXUP(T, x)

RB-DELETE(T, z)



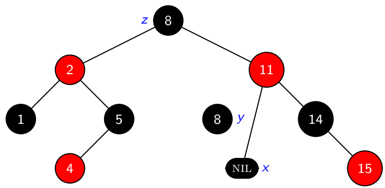

```
if left[z] = nil[T] or right[z] = nil[T]
    y ← z
else
    y ← TREE-SUCCESSOR(z)
if left[y] ≠ nil[T]
    x ← left[y]
else
    x ← right[y]
p[x] ← p[y]
if p[y] = nil[T]
else if y = left[p[y]]
    left[p[y]] ← x
else
    right[p[y]] ← x
if colour[y] = BLACK
    RB-DELETE-FIXUP(T, x)
```

RB-DELETE(T, z)



if $\mathit{left}[z] = \mathit{nil}[T]$ or $\mathit{right}[z] = \mathit{nil}[T]$
    $y \leftarrow z$
else
    $y \leftarrow$ TREE-SUCCESSOR(z)
if $\mathit{left}[y] \neq \mathit{nil}[T]$
    $x \leftarrow \mathit{left}[y]$
else
    $x \leftarrow \mathit{right}[y]$
$p[x] \leftarrow p[y]$
if $p[y] = \mathit{nil}[T]$
else if $y = \mathit{left}[p[y]]$
    $\mathit{left}[p[y]] \leftarrow x$
else
    $\mathit{right}[p[y]] \leftarrow x$
if $\mathit{colour}[y] =$ BLACK
    RB-DELETE-FIXUP(T, x)

RB-DELETE(T, z)



if $left[z] = nil[T]$ or $right[z] = nil[T]$

$\quad y \leftarrow z$

else

$\quad y \leftarrow$ TREE-SUCCESSOR(z)

if $left[y] \neq nil[T]$

$\quad x \leftarrow left[y]$

else

$\quad x \leftarrow right[y]$

$p[x] \leftarrow p[y]$

if $p[y] = nil[T]$

else if $y = left[p[y]]$

$\quad left[p[y]] \leftarrow x$

else

$\quad right[p[y]] \leftarrow x$

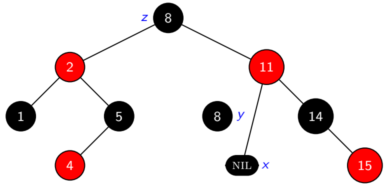
if $y \neq z$

$\quad key[z] \leftarrow key[y]$

if $colour[y] =$ BLACK

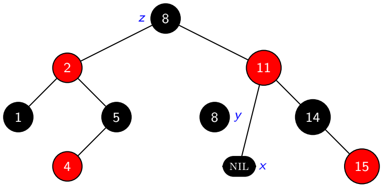
$\quad$ RB-DELETE-FIXUP(T, x)

RB-DELETE(T, z)



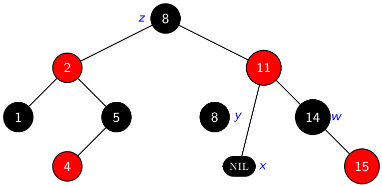
if $left[z] = nil[T]$ or $right[z] = nil[T]$
    $y \leftarrow z$
else
    $y \leftarrow$ TREE-SUCCESSOR(z)
if $left[y] \neq nil[T]$
    $x \leftarrow left[y]$
else
    $x \leftarrow right[y]$
$p[x] \leftarrow p[y]$
if $p[y] = nil[T]$
else if $y = left[p[y]]$
    $left[p[y]] \leftarrow x$
else
    $right[p[y]] \leftarrow x$
if $y \neq z$
    $key[z] \leftarrow key[y]$
if $colour[y] =$ BLACK
    RB-DELETE-FIXUP(T, x)

RB-DELETE-FIXUP(T, x)



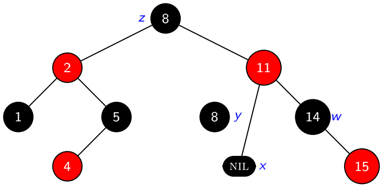
1. **while** $x \neq root[T]$ and $colour[x] =$ BLACK
2. **if** $x = left[p[x]]$
3. $colour[x] \leftarrow$ BLACK

RB-DELETE-FIXUP(T, x)



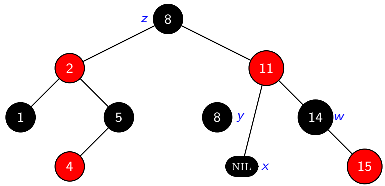
1. **while** $x \neq root[T]$ and $colour[x] =$ BLACK
2. **if** $x = left[p[x]]$
3. $w \leftarrow right[p[x]]$
4. $colour[x] \leftarrow$ BLACK

RB-DELETE-FIXUP(T, x)



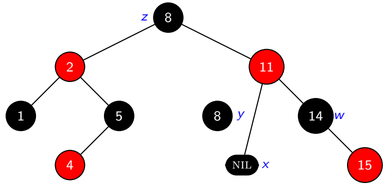
while $x \neq root[T]$ and $colour[x] = \text{BLACK}$
    if $x = left[p[x]]$
        $w \leftarrow right[p[x]]$
        if $colour[w] = \text{RED}$
            Case 1
$colour[x] \leftarrow \text{BLACK}$

RB-DELETE-FIXUP(T, x)



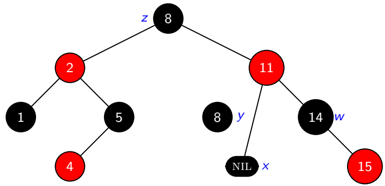
1. **while** $x \neq root[T]$ and $colour[x] = \text{BLACK}$
2. **if** $x = left[p[x]]$
3. $w \leftarrow right[p[x]]$
4. **if** $colour[w] = \text{RED}$
   Case 1
5. **if** $colour[left[w]] = \text{BLACK}$ and $colour[right[w]] = \text{BLACK}$
   Case 2
6. $colour[x] \leftarrow \text{BLACK}$

RB-DELETE-FIXUP(T, x)



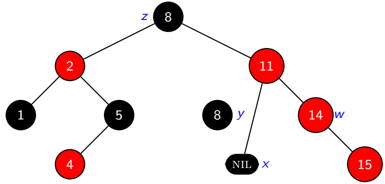
1. **while** $x \neq root[T]$ and $colour[x] =$ BLACK
2. **if** $x = left[p[x]]$
3. $w \leftarrow right[p[x]]$
4. **if** $colour[w] =$ RED
   Case 1
5. **if** $colour[left[w]] =$ BLACK and $colour[right[w]] =$ BLACK
   Case 2
6. **else if** $colour[right[w]] =$ BLACK
   Case 3
7. $colour[x] \leftarrow$ BLACK

RB-DELETE-FIXUP(T, x)



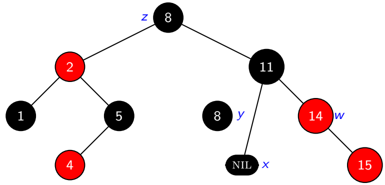
1. **while** $x \neq root[T]$ **and** $colour[x] = \text{BLACK}$
2. **if** $x = left[p[x]]$
3. $w \leftarrow right[p[x]]$
4. **if** $colour[w] = \text{RED}$
   Case 1
5. **if** $colour[left[w]] = \text{BLACK}$ **and** $colour[right[w]] = \text{BLACK}$
   Case 2
6. **else if** $colour[right[w]] = \text{BLACK}$
   Case 3
7. **else if** $colour[right[w]] = \text{RED}$
   // Case 4
8. $colour[x] \leftarrow \text{BLACK}$

RB-DELETE-FIXUP(T, x)



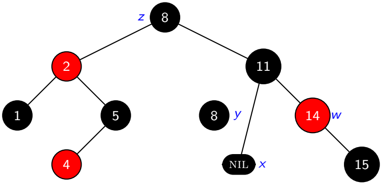
while $x \neq root[T]$ and $colour[x] =$ BLACK
  if $x = left[p[x]]$
    $w \leftarrow right[p[x]]$
    if $colour[w] =$ RED
      Case 1
    if $colour[left[w]] =$ BLACK and $colour[right[w]] =$ BLACK
      Case 2
    else if $colour[right[w]] =$ BLACK
      Case 3
    else if $colour[right[w]] =$ RED
      // Case 4
      $colour[w] \leftarrow colour[p[x]]$
$colour[x] \leftarrow$ BLACK

RB-DELETE-FIXUP(T, x)



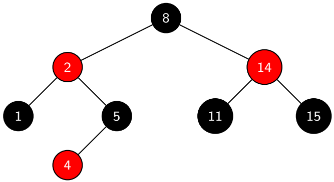

```
while x ≠ root[T] and colour[x] = BLACK
   if x = left[p[x]]
      w ← right[p[x]]
      if colour[w] = RED
         Case 1
      if colour[left[w]] = BLACK and colour[right[w]] = BLACK
         Case 2
      else if colour[right[w]] = BLACK
         Case 3
      else if colour[right[w]] = RED
         // Case 4
         colour[w] ← colour[p[x]]
         colour[p[x]] ← BLACK
colour[x] ← BLACK
```

RB-DELETE-FIXUP(T, x)



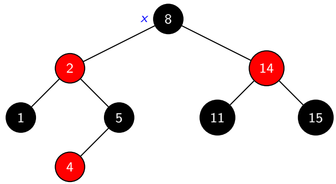
while $x \neq root[T]$ and $colour[x] = \text{BLACK}$
  if $x = left[p[x]]$
    $w \leftarrow right[p[x]]$
    if $colour[w] = \text{RED}$
      Case 1
    if $colour[left[w]] = \text{BLACK}$ and $colour[right[w]] = \text{BLACK}$
      Case 2
    else if $colour[right[w]] = \text{BLACK}$
      Case 3
    else if $colour[right[w]] = \text{RED}$
      // Case 4
      $colour[w] \leftarrow colour[p[x]]$
      $colour[p[x]] \leftarrow \text{BLACK}$
      $colour[right[w]] \leftarrow \text{BLACK}$
$colour[x] \leftarrow \text{BLACK}$

RB-DELETE-FIXUP(T, x)




```
while x ≠ root[T] and colour[x] = BLACK
   if x = left[p[x]]
      w ← right[p[x]]
      if colour[w] = RED
         Case 1
      if colour[left[w]] = BLACK and colour[right[w]] = BLACK
         Case 2
      else if colour[right[w]] = BLACK
         Case 3
      else if colour[right[w]] = RED
         // Case 4
         colour[w] ← colour[p[x]]
         colour[p[x]] ← BLACK
         colour[right[w]] ← BLACK
         LEFT-ROTATE(T, p[x])
colour[x] ← BLACK
```

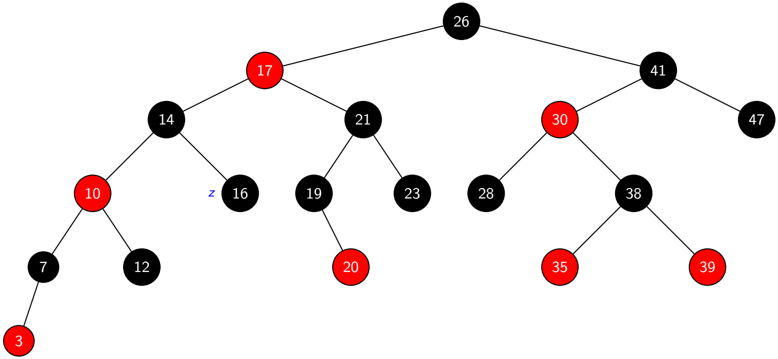
RB-DELETE-FIXUP(T, x)




```
while x ≠ root[T] and colour[x] = BLACK
   if x = left[p[x]]
      w ← right[p[x]]
      if colour[w] = RED
         Case 1
      if colour[left[w]] = BLACK and colour[right[w]] = BLACK
         Case 2
      else if colour[right[w]] = BLACK
         Case 3
      else if colour[right[w]] = RED
         // Case 4
         colour[w] ← colour[p[x]]
         colour[p[x]] ← BLACK
         colour[right[w]] ← BLACK
         LEFT-ROTATE(T, p[x])
         x ← root[T]
colour[x] ← BLACK
```

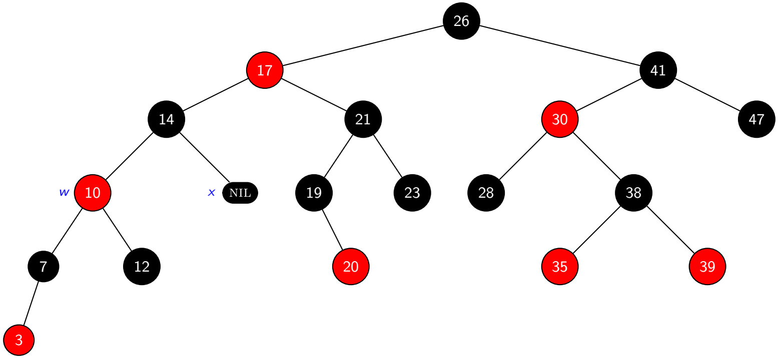
RB-DELETE-FIXUP(T, x)

Delete $z = 16$.



RB-DELETE-FIXUP(T, x)

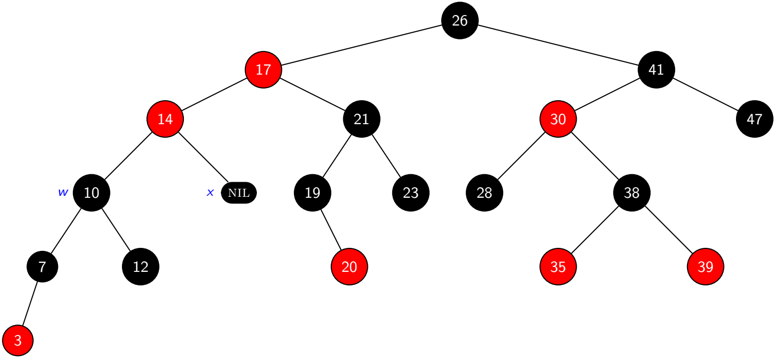
Case 1: $colour[x] = \text{BLACK}$, $x = right[p[x]]$ and $colour[w] = \text{RED}$ (Mirror case!)



RB-DELETE-FIXUP(T, x)

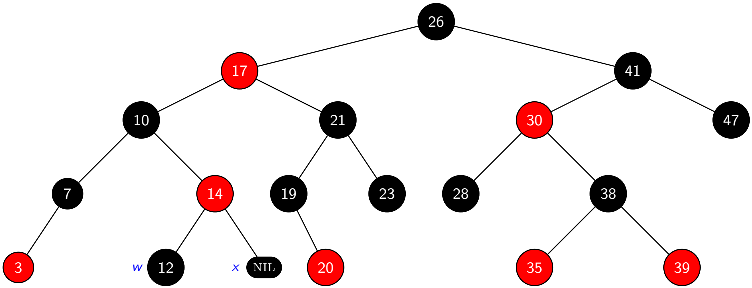
Case 1: $colour[x] = \text{BLACK}$, $x = right[p[x]]$ and $colour[w] = \text{RED}$ (Mirror case!)

- $colour[w] \leftarrow \text{BLACK}$
- $colour[p[x]] \leftarrow \text{RED}$



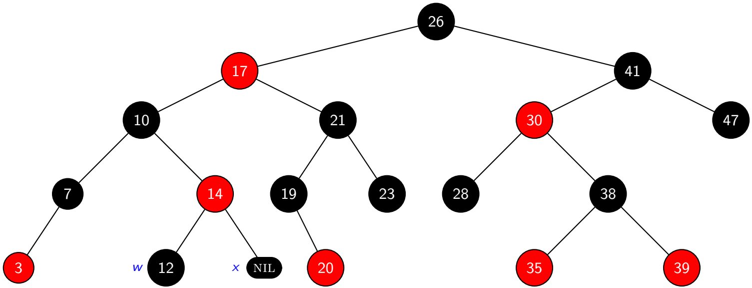
Case 1: $colour[x] = \text{BLACK}$, $x = right[p[x]]$ and $colour[w] = \text{RED}$ (Mirror case!)

- $colour[w] \leftarrow \text{BLACK}$
- $colour[p[x]] \leftarrow \text{RED}$
- RIGHT-ROTATE($T, p[x]$)
- $w \leftarrow left[p[x]]$



RB-DELETE-FIXUP(T, x)

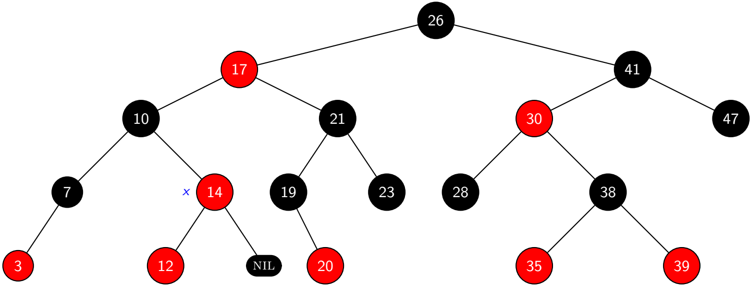
Case 2: $colour[left[w]] = \text{BLACK}$ and $colour[right[w]] = \text{BLACK}$



RB-DELETE-FIXUP(T, x)

Case 2: $colour[left[w]] = \text{BLACK}$ and $colour[right[w]] = \text{BLACK}$

- $colour[w] \leftarrow \text{RED}$
- $x \leftarrow p[x]$

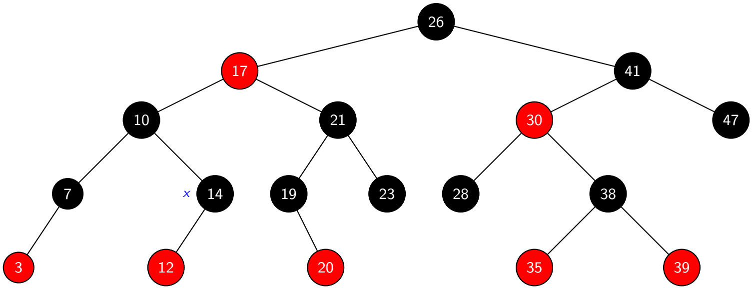


RB-DELETE-FIXUP(T, x)

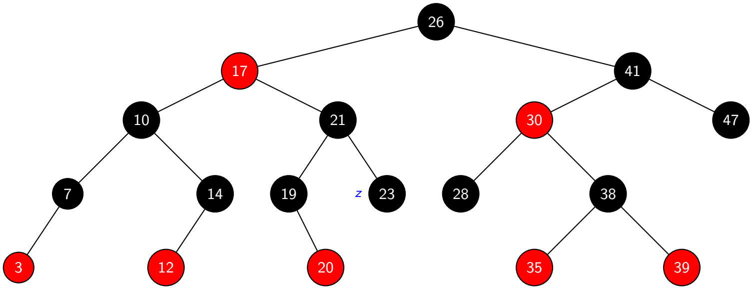
Case 2: $colour[left[w]] = \text{BLACK}$ and $colour[right[w]] = \text{BLACK}$

- $colour[w] \leftarrow \text{RED}$
- $x \leftarrow p[x]$

$colour[x] \leftarrow \text{BLACK}$ (outside the while loop)

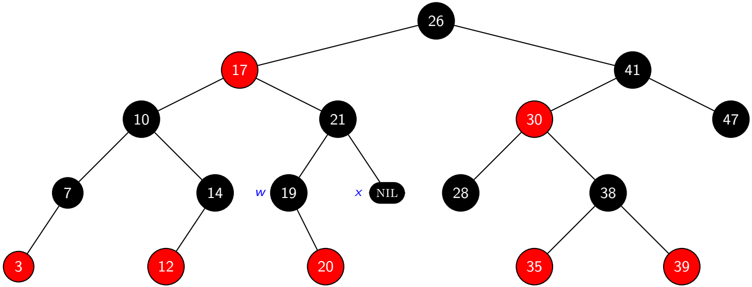


Delete $z = 23$.



RB-DELETE-FIXUP(T, x)

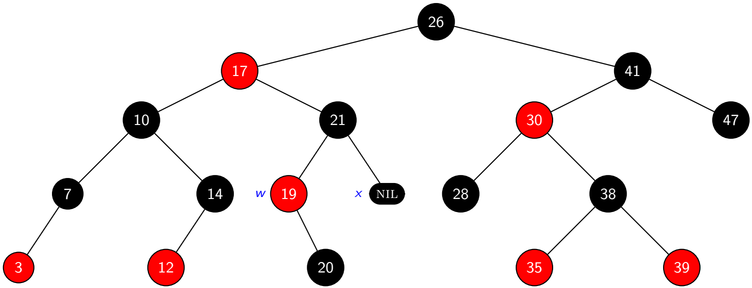
Case 3: $colour[x] = \text{BLACK}$, $x = right[p[x]]$, $colour[left[w]] = \text{BLACK}$ and $colour[right[w]] = \text{RED}$ (Mirror case!)



RB-DELETE-FIXUP(T, x)

Case 3: $colour[x] = \text{BLACK}$, $x = right[p[x]]$, $colour[left[w]] = \text{BLACK}$ and $colour[right[w]] = \text{RED}$ (Mirror case!)

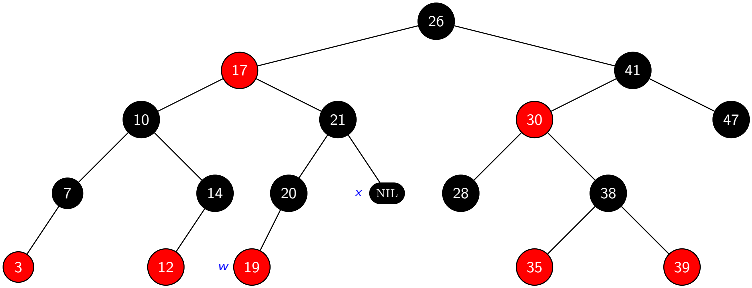
- $colour[right[w]] \leftarrow \text{BLACK}$
- $colour[w] \leftarrow \text{RED}$



RB-Delete-Fixup(T, x)

Case 3: $colour[x] = \text{BLACK}$, $x = right[p[x]]$, $colour[left[w]] = \text{BLACK}$ and $colour[right[w]] = \text{RED}$ (Mirror case!)

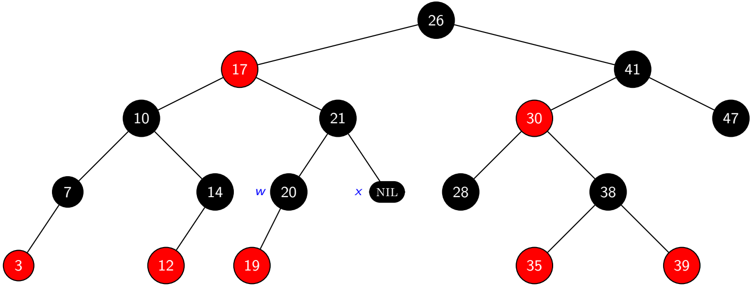
- $colour[right[w]] \leftarrow \text{BLACK}$
- $colour[w] \leftarrow \text{RED}$
- Left-Rotate(T, w)



RB-DELETE-FIXUP(T, x)

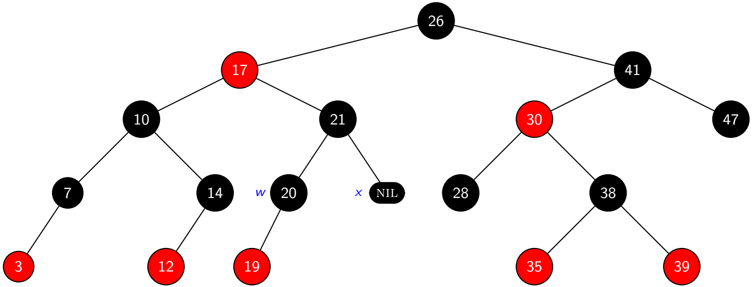
Case 3: $colour[x] = \text{BLACK}$, $x = right[p[x]]$, $colour[left[w]] = \text{BLACK}$ and $colour[right[w]] = \text{RED}$ (Mirror case!)

- $colour[right[w]] \leftarrow \text{BLACK}$
- $colour[w] \leftarrow \text{RED}$
- LEFT-ROTATE(T, w)
- $w \leftarrow left[p[x]]$



RB-DELETE-FIXUP(T, x)

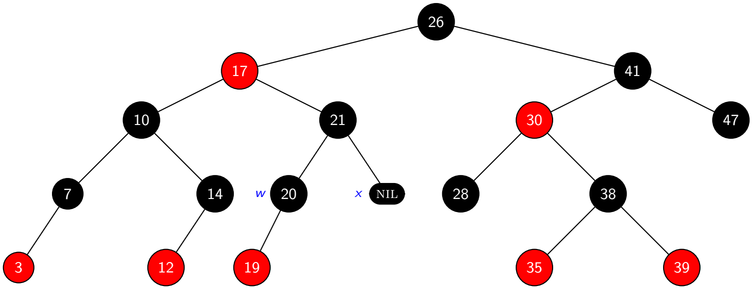
Case 4: $colour[x] = \text{BLACK}$, $x = right[p[x]]$, $colour[w] = \text{BLACK}$, $colour[left[w]] = \text{RED}$ (Mirror case!)



RB-DELETE-FIXUP(T, x)

Case 4: $colour[x] = \text{BLACK}$, $x = right[p[x]]$, $colour[w] = \text{BLACK}$, $colour[left[w]] = \text{RED}$ (Mirror case!)

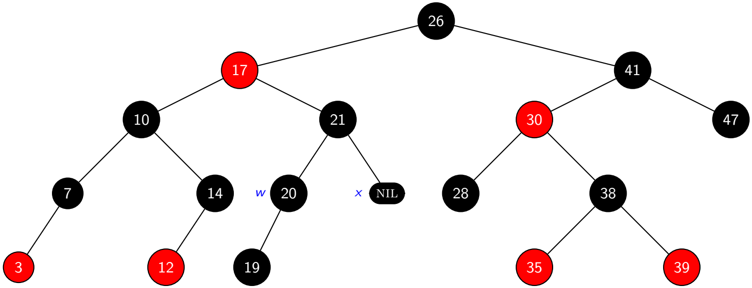
- $colour[w] \leftarrow colour[p[x]]$
- $colour[p[x]] \leftarrow \text{BLACK}$



RB-DELETE-FIXUP(T, x)

Case 4: $colour[x]$ = BLACK, $x = right[p[x]]$, $colour[w]$ = BLACK, $colour[left[w]]$ = RED (Mirror case!)

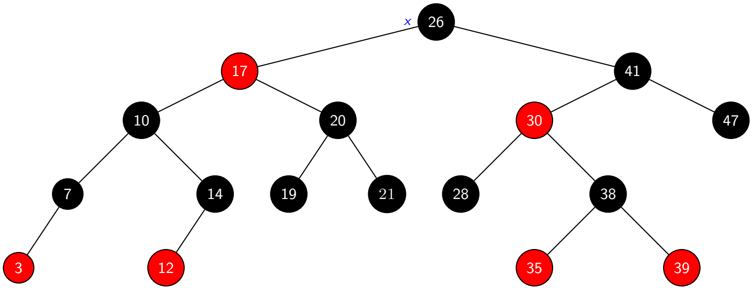
- $colour[w] \leftarrow colour[p[x]]$
- $colour[p[x]] \leftarrow$ BLACK
- $colour[left[w]] \leftarrow$ BLACK



RB-DELETE-FIXUP(T, x)

Case 4: $colour[x] = \text{BLACK}$, $x = right[p[x]]$, $colour[w] = \text{BLACK}$, $colour[left[w]] = \text{RED}$ (Mirror case!)

- $colour[w] \leftarrow colour[p[x]]$
- $colour[p[x]] \leftarrow \text{BLACK}$
- $colour[left[w]] \leftarrow \text{BLACK}$
- RIGHT-ROTATE($T, p[x]$)
- $x \leftarrow root[T]$



Thank You for your kind attention!

Books and Other Materials Consulted

1. Red-Black Tree part taken from Chapter 13 of the *Introduction to Algorithms* book by Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein.

Questions!!