# Design and Analysis of Algorithm

## Advanced Data Structure
## (Red Black Tree)

(Properties, Rotation and Insertion)

**LECTURE 28 -31**

# Overview

- A variation of binary search trees.
- Balanced: height is O(lg n), where n is the number of nodes.
- Operations will take O(lg n) time in the worst case.

# Red Black Tree

- A red-black tree is a binary search tree + 1 bit per node: an attribute color, which is either red or black.
- All leaves are empty (nil) and colored black.
  - A single sentinel, nil[T ], is used for all the leaves of red-black tree T .
  - Color of [nil[T ]](i.e. leaf node) is black.
  - The root′s parent is also nil[T ].
- All other attributes of binary search trees are inherited by red-black trees (i.e. key, left, right, and p as parent).
- We don′t care about the key in nil[T ].

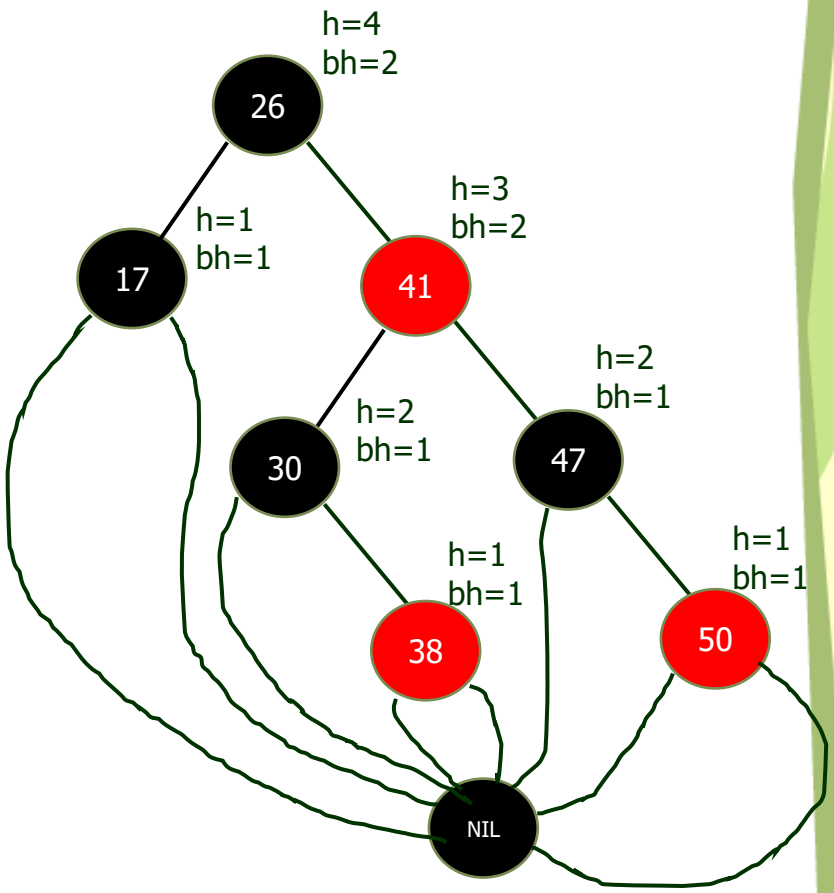
# Red Black Tree

- **Properties:**
  1. Every node is either red or black.
  2. The root is always black.
  3. Every leaf (nil[T]) is black.
  4. If a node is red, then both its children are black. (Hence no two reds in a row on a simple path from the root to a leaf is allowed .)
  5. For each node, all paths from the node to descendant leaves contain the same number of black nodes.

# Red Black Tree

## Height of a red-black tree

- Height of a node is the number of edges in a longest path to a leaf.
- Black-height of a node x: bh(x) is the number of black nodes (including nil[T ]) on the path from x to leaf, not counting x. By property 5, black-height is well defined.
- So, a red-black tree of height h has black height >= h/2. Height of a red-black tree with n nodes is $h <= 2 \log_2(n + 1)$



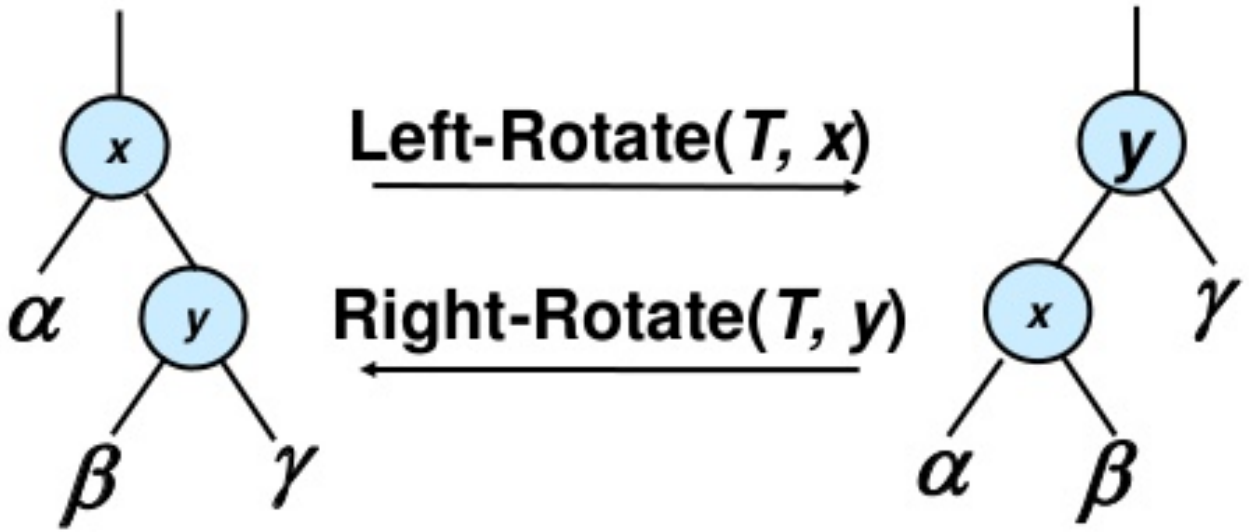
# Red Black Tree

**Rotations**

- The basic tree-restructuring operation.
- Needed to maintain red-black trees as balanced binary search trees.
- Changes the local pointer structure. (Only pointers are changed.)
- Won't upset the binary-search-tree property.
- Have both left rotation and right rotation. They are inverses of each other.
- A rotation takes a red-black-tree and a node within the tree.

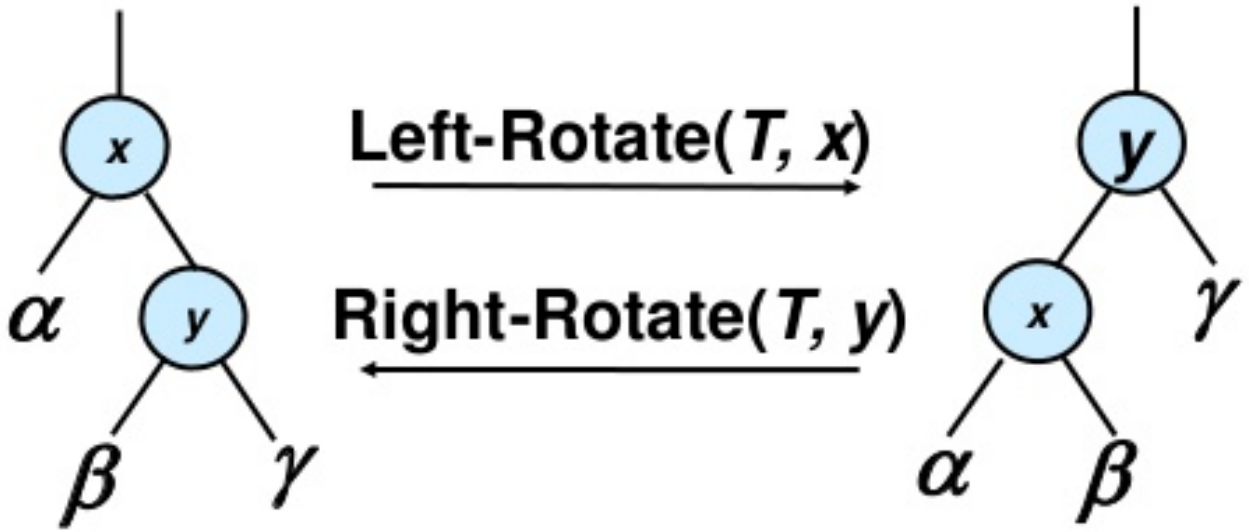
# Red Black Tree

Rotations



# Red Black Tree

Rotations



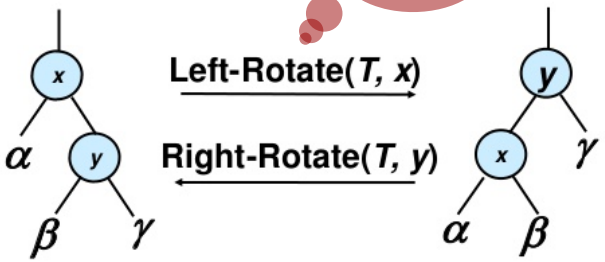

**Let's learn the game of changing pointer positions**

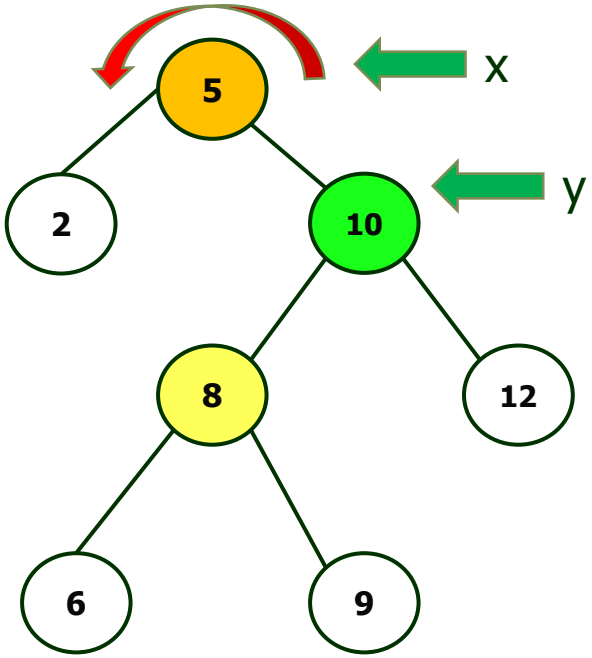
# Red Black Tree

Keep this in mind

## Rotations

Left rotate on 5



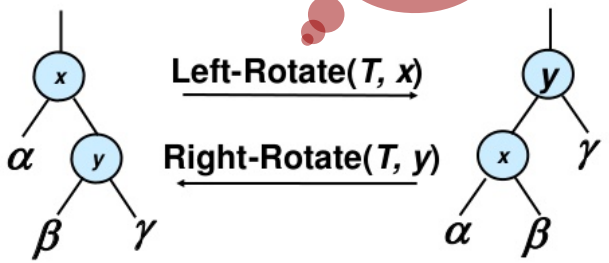



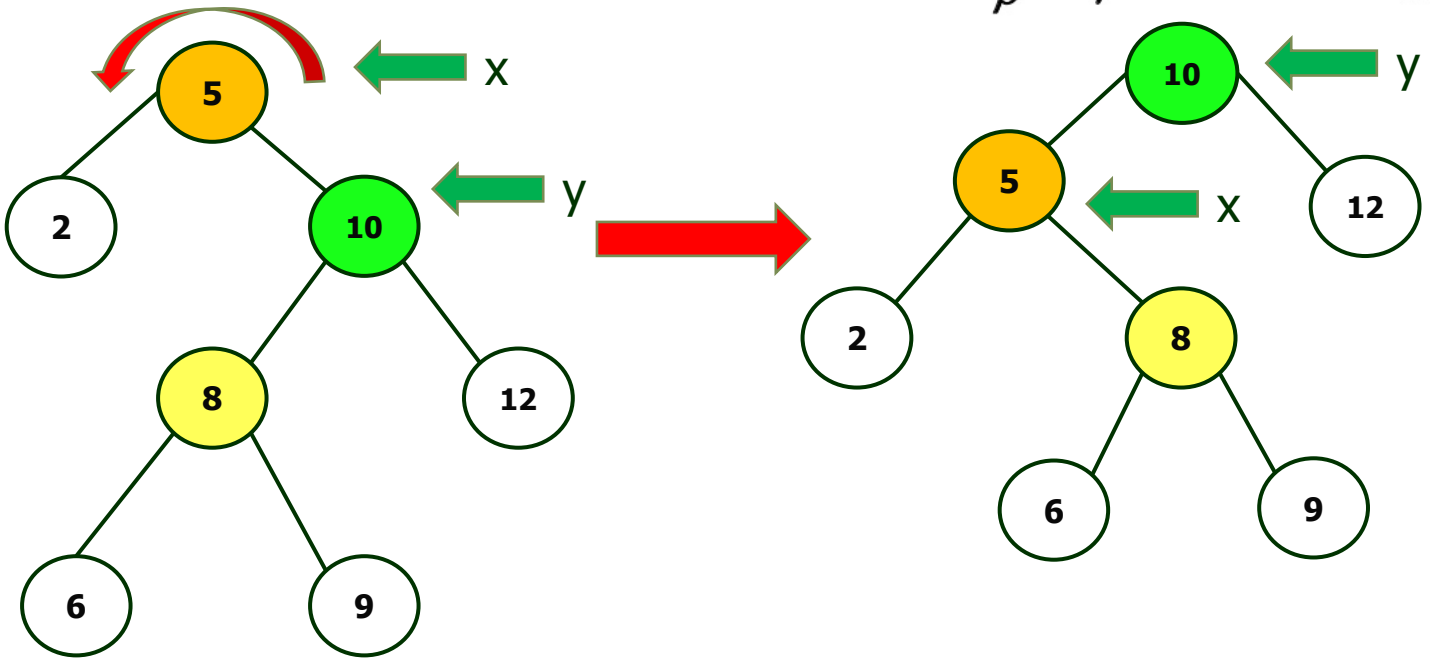
# Red Black Tree

Keep this in mind

## Rotations

Left rotate on 5



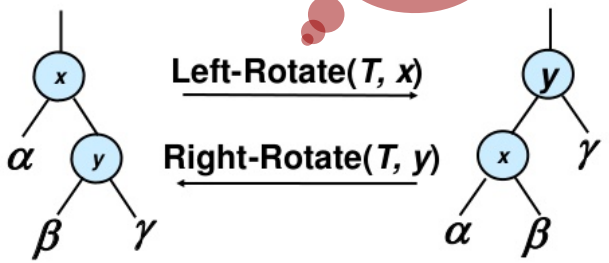



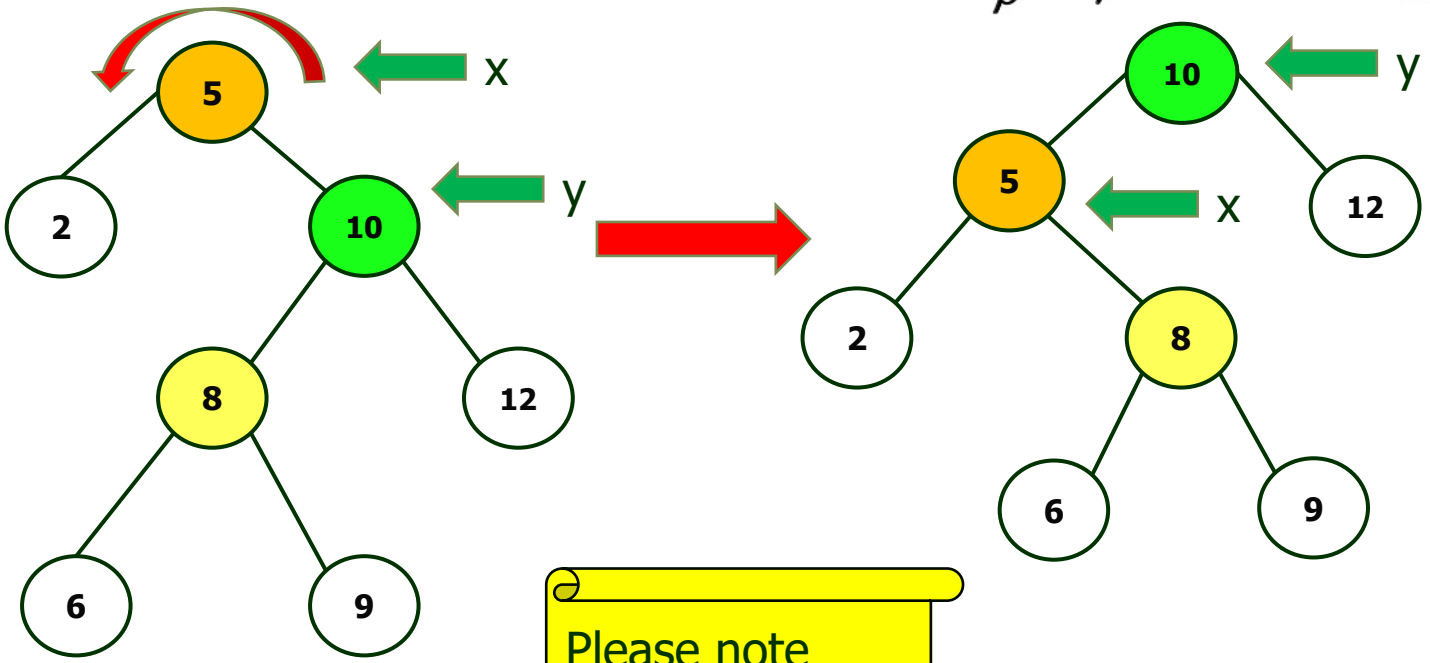
# Red Black Tree

Keep this in mind

## Rotations

Left rotate on 5



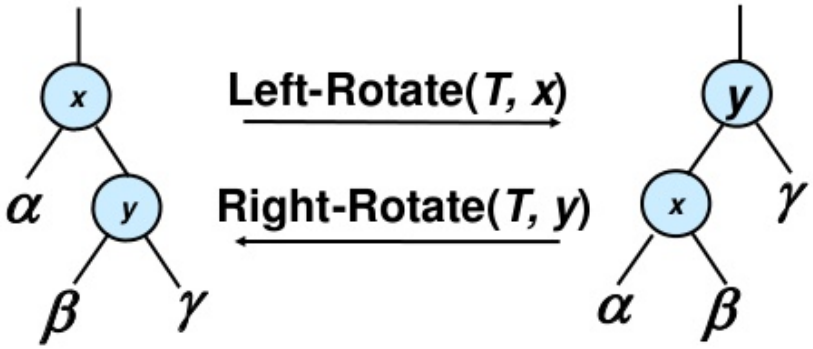

Please note down the observations

# Red Black Tree

## Rotations

**LEFT-ROTATE*(T, x)***

```
y ← right[x]            //Set y.
right[x] ← left[y]      //Turn y.s left subtree into x.s right subtree.
if left[y] ≠ nil[T ]
    then p[left[y]] ← x
p[y] ← p[x]             //Link x.s parent
if p[x] = nil[T ]
    then root[T ] ← y
    else if x = left[p[x]]
            then left[p[x]] ← y
            else right[p[x]] ← y
left[y] ← x             //Put x on y.s left.
p[x] ← y
```



# Red Black Tree

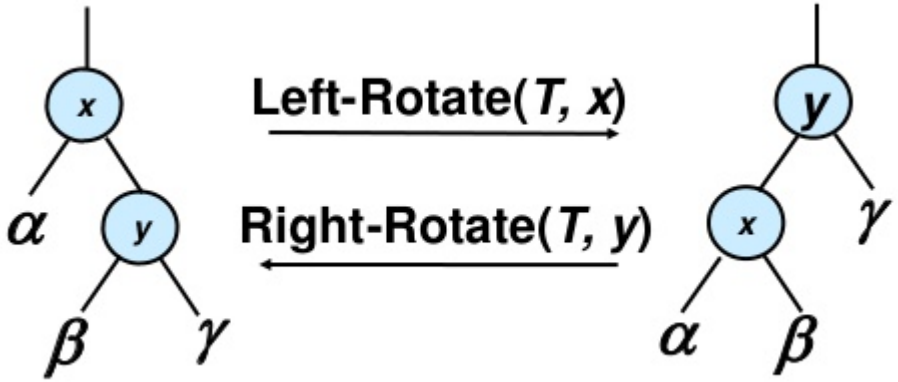
## Rotations

The pseudocode for LEFT-ROTATE assumes that

*right*[*x*] = *nil*[*T* ], and

root.s parent is *nil*[*T* ].

Pseudocode for RIGHT-ROTATE is symmetric: exchange *left* and *right* everywhere.



# Red Black Tree

## Rotations

**RIGHT-ROTATE*(T, x)***

$y \leftarrow left[x]$ //Set $y$.

$left[x] \leftarrow right[y]$ //Turn $y$.s left subtree into $x$.s right subtree.

**if** $right[y] \neq nil[T]$

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

$p[y] \leftarrow p[x]$ //Link $x$.s parent

**if** $p[x] = nil[T]$

  **then** $root[T] \leftarrow y$

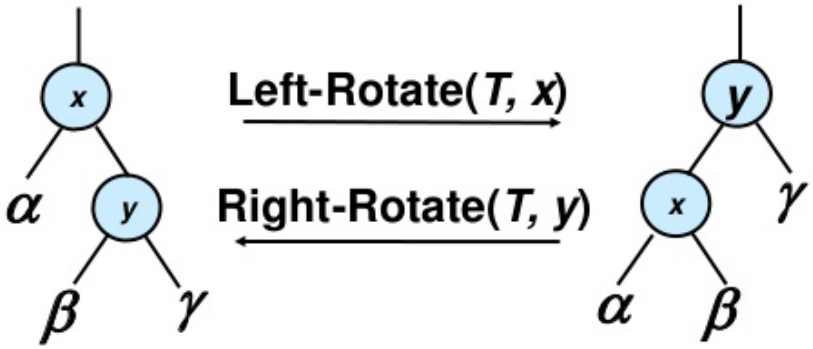
  **else if** $x = right[p[x]]$

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

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

$right[y] \leftarrow x$ //Put $x$ on $y$.s right.

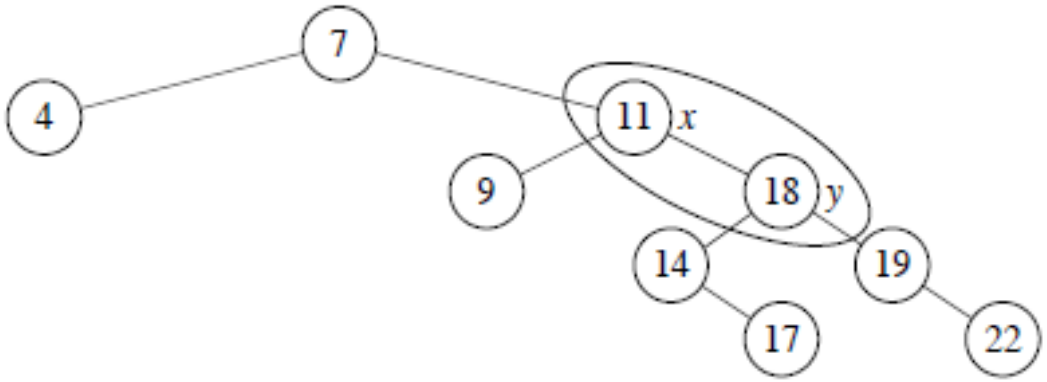
$p[x] \leftarrow y$



# Red Black Tree

## Rotations (Example)

Demonstrate of left rotation that maintains in-order ordering of keys.

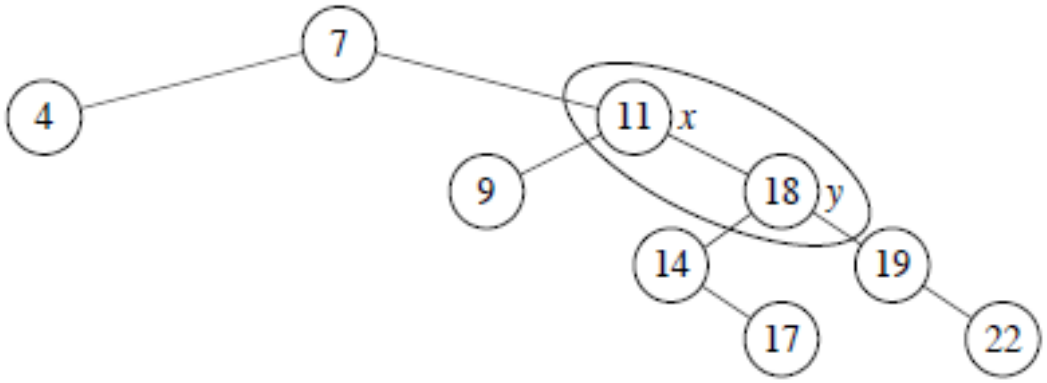


# Red Black Tree

## Rotations (Example)

Demonstrate of left rotation that maintains in-order ordering of keys.

LEFT-ROTATE($T$, $x$)

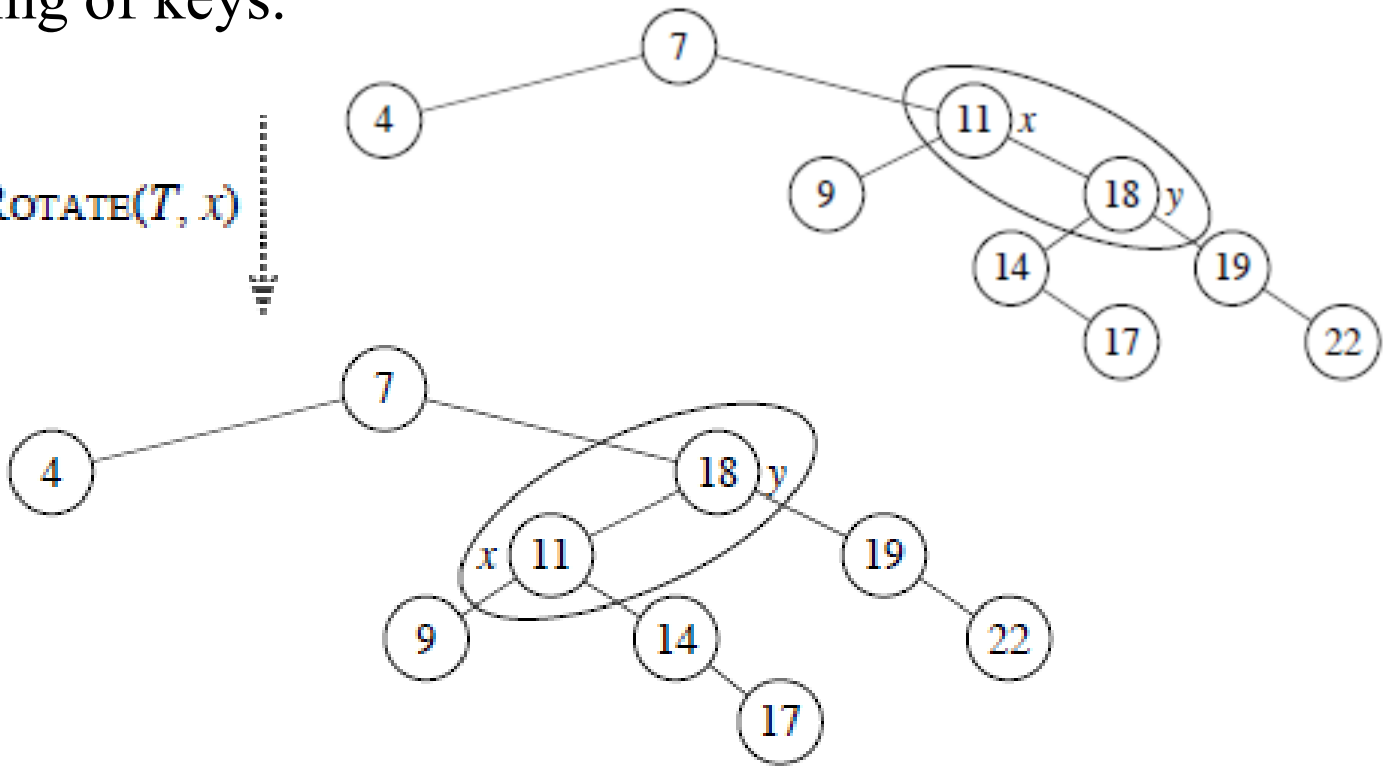


# Red Black Tree

## Rotations (Example)

Demonstrate of left rotation that maintains in-order ordering of keys.

LEFT-ROTATE($T$, $x$)

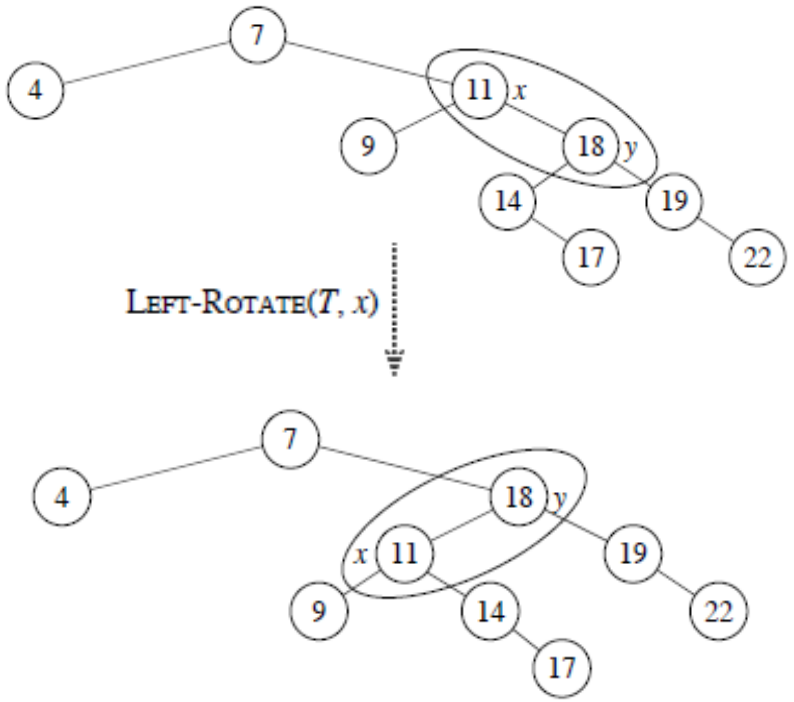


# Red Black Tree

## Rotations (Example)

Demonstrate of left rotation that maintains in-order ordering of keys.

- Before rotation: keys of $x's$ left subtree $\leq 11 \leq$ keys of $y's$ left subtree $\leq 18 \leq$ keys of $y's$ right subtree.
- Rotation makes $y's$ left subtree into $x's$ right subtree.
- After rotation: keys of $x's$ left subtree $\leq 11 \leq$ keys of $x's$ right subtree $\leq 18 \leq$ keys of $y's$ right subtree.
- Time complexity : O(1) for both LEFT-ROTATE and RIGHT-ROTATE, since a constant number of pointers are modified.



# Red Black Tree

Insertion:

Start by doing regular binary-search-tree insertion:

# Red Black Tree

## Insertion:

```
RB-INSERT(T, z)
    y ← nil[T ]
    x ← root[T ]
    while x ≠ nil[T ]
      do y ← x
        if key[z] < key[x]
        then x ← left[x]
        else x ← right[x]
    p[z] ← y
    if y = nil[T ]
      then root[T ] ← z
      else if key[z] < key[y]
              then left[y] ← z
              else right[y] ← z
    left[z] ← nil[T ]
    right[z] ← nil[T ]
    color[z] ← RED
    RB-INSERT-FIXUP(T, z)
```

# Red Black Tree

## Insertion:

```
RB-INSERT(T, z)
    y ← nil[T ]
    x ← root[T ]
    while x ≠ nil[T ]
      do y ← x
        if key[z] < key[x]
        then x ← left[x]
        else x ← right[x]
    p[z] ← y
    if y = nil[T ]
      then root[T ] ← z
      else if key[z] < key[y]
              then left[y] ← z
              else right[y] ← z
    left[z] ← nil[T ]
    right[z] ← nil[T ]
    color[z] ← RED
    RB-INSERT-FIXUP(T, z)
```

Simply Binary Search Tree Implementation

# Red Black Tree

## Insertion:

- RB-INSERT ends by coloring the new node z red.
- Then it calls RB-INSERT-FIXUP to maintain the properties of a red-black Tree.

Which property might be violated?

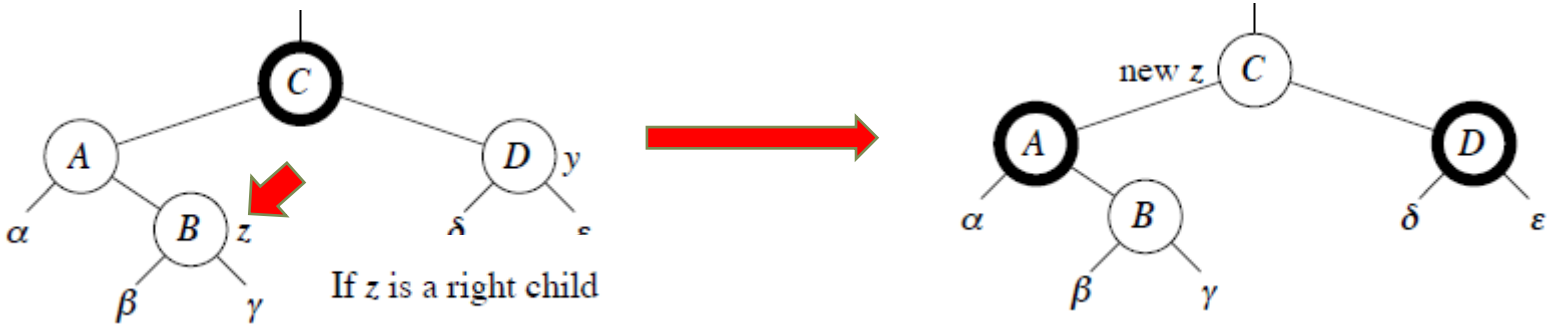
| | |
|---|---|
| 1. Every node is either red or black. | OK |
| 2. The root is always black. | If z is the root, then there′s a violation. Otherwise, OK. |
| 3. Every leaf (nil[T]) is black. | OK |
| 4. If a node is red, then both its children are black. | If p[z] is red, there′s a violation: both z and p[z] are red. |
| 5. For each node, all paths from the node to descendant leaves contain the same number of black nodes. | OK |

# Red Black Tree

## Insertion (RB-INSERT-FIXUP CASE 1) ($y$ is red and $z$ is a right child)

[Nodes with bold outline indicate black nodes and light outline indicate red nodes]

**if** *color*[*y*] = RED
  **then** *color*[*p*[*z*]] ← BLACK //Case 1
    *color*[*y*] ← BLACK //Case 1
    *color*[*p*[*p*[*z*]]] ← RED //Case 1
    *z* ← *p*[*p*[*z*]] //Case 1

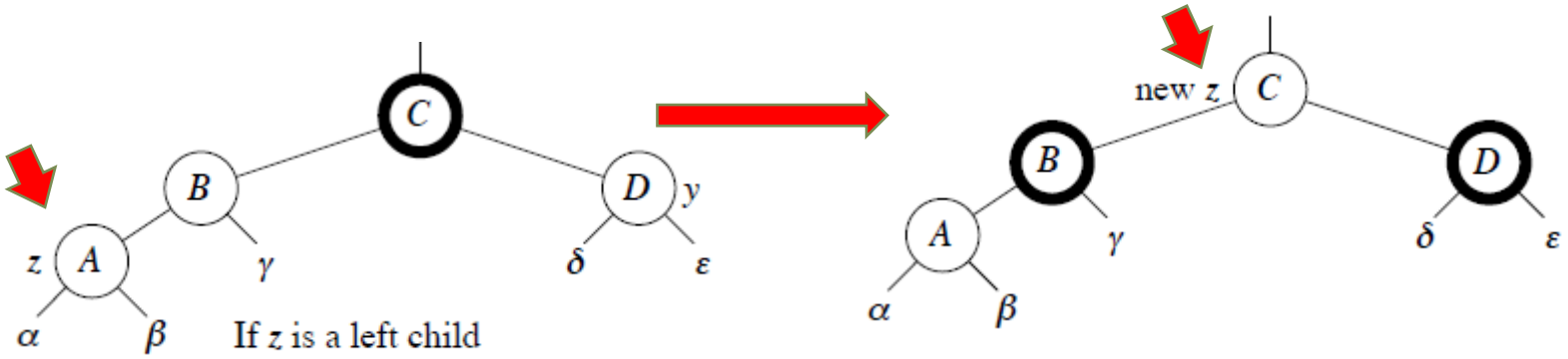


# Red Black Tree

## Insertion (RB-INSERT-FIXUP CASE 1) ($y$ is red and $z$ is a left child)

[Nodes with bold outline indicate black nodes and light outline indicate red nodes]

**if** *color*[*y*] = RED
  **then** *color*[*p*[*z*]] ← BLACK //Case 1
    *color*[*y*] ← BLACK //Case 1
    *color*[*p*[*p*[*z*]]] ← RED //Case 1
    *z* ← *p*[*p*[*z*]] //Case 1

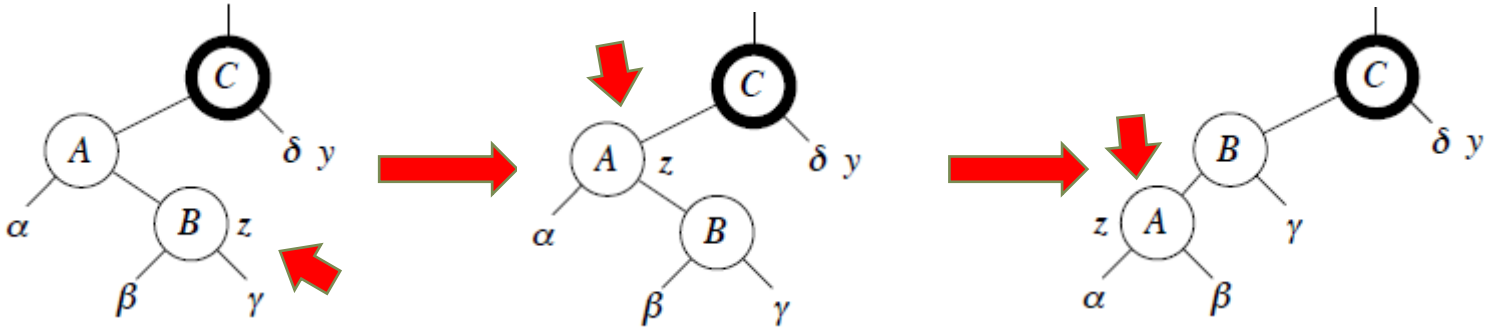


# Red Black Tree

## Insertion (RB-INSERT-FIXUP CASE 2 (*y* is black, *z* is a right child)

[Nodes with bold outline indicate black nodes and light outline indicate red nodes]

**if** $z = right[p[z]]$
  **then** $z \leftarrow p[z]$ //Case 2
    LEFT-ROTATE$(T, z)$ //Case 2

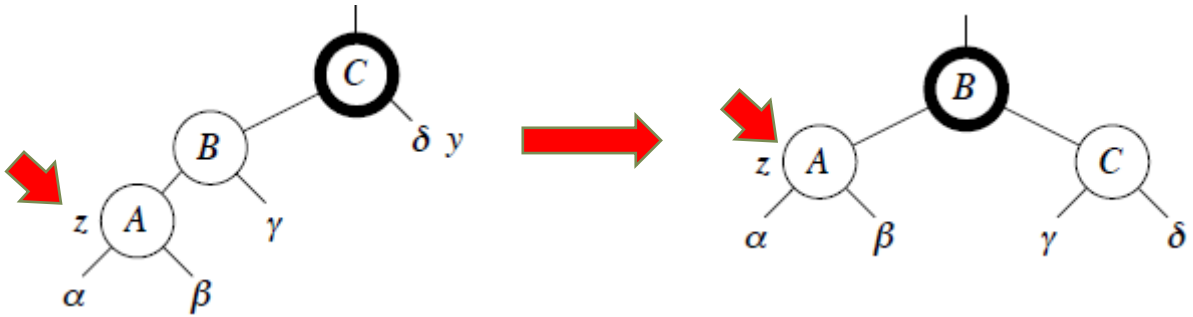


# Red Black Tree

## Insertion (RB-INSERT-FIXUP CASE 3 (*y* is black, *z* is a left child)

[Nodes with bold outline indicate black nodes and light outline indicate red nodes]

*color*[*p*[*z*]] ← BLACK // Case 3
*color*[*p*[*p*[*z*]]] ← RED // Case 3
RIGHT-ROTATE(*T*, *p*[*p*[*z*]]) // Case 3



# Red Black Tree

## Insertion:

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

**while** *color*[*p*[*z*]] = RED
**do if** *p*[*z*] = *left*[*p*[*p*[*z*]]]
**then** *y* ← *right*[*p*[*p*[*z*]]]
**if** *color*[*y*] = RED
**then**
**Apply Case 1**
**else if** *z* = *right*[*p*[*z*]]
**then**
**Apply Case 2**
**Apply Case 3**
**else** (same as **then** clause with .right. and .left. exchanged)
*color*[*root*[*T* ]] ← BLACK

# Red Black Tree

## Insertion:

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

**while** *color*[*p*[*z*]] = RED
**do if** *p*[*z*] = *left*[*p*[*p*[*z*]]]
**then** *y* ← *right*[*p*[*p*[*z*]]]
**if** *color*[*y*] = RED
**then**
**Apply Case 1**
**else if** *z* = *right*[*p*[*z*]]
**then**
**Apply Case 2**
**Apply Case 3**
**else** (same as **then** clause with .right. and .left. exchanged)
*color*[*root*[*T* ]] ← BLACK

**Once we apply Case 2 immediately, we apply Case 3**

# Red Black Tree

## Insertion:

**Once we apply Case 2 immediately, we apply Case 3**

```
RB-INSERT-FIXUP(T, z)
    while color[p[z]] = RED
      do if p[z] = left[p[p[z]]]
        then y ← right[p[p[z]]]
             if color[y] = RED
               then color[p[z]] ← BLACK                    //Case 1
                    color[y] ← BLACK                       //Case 1
                    color[p[p[z]]] ← RED                   //Case 1
                    z ← p[p[z]]                            //Case 1
               else if z = right[p[z]]
                      then z ← p[z]                        //Case 2
                           LEFT-ROTATE(T, z)               //Case 2
                    color[p[z]] ← BLACK                    // Case 3
                    color[p[p[z]]] ← RED                   // Case 3
                    RIGHT-ROTATE(T, p[p[z]])               // Case 3
        else (same as then clause with .right. and .left. exchanged)
  color[root[T ]] ← BLACK
```

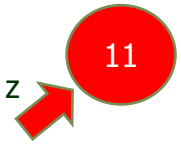
# Red Black Tree

## Insertion (Example):

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -11



# Red Black Tree

## Insertion (Example 1):

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -11

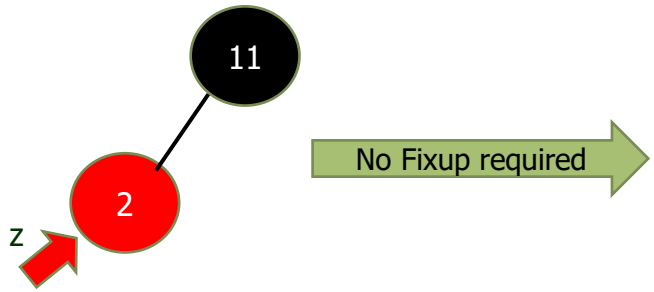
# Red Black Tree

## Insertion (Example 1):

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -2



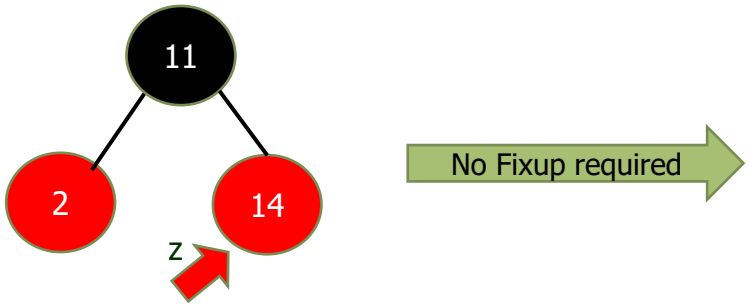
# Red Black Tree

## Insertion (Example 1):

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -14



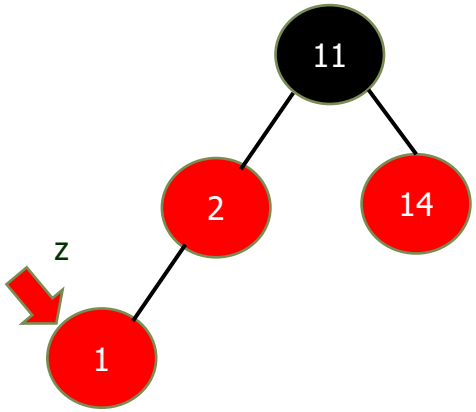
# Red Black Tree

## Insertion (Example 1):

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -1



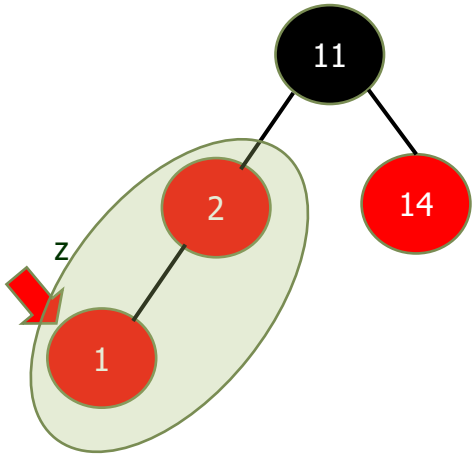
# Red Black Tree

## Insertion (Example 1):

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -1



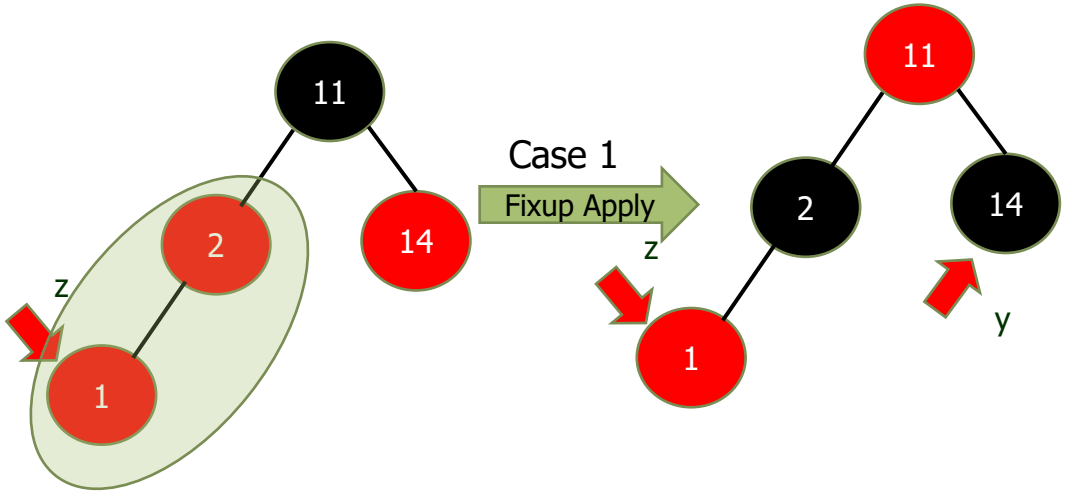
# Red Black Tree

## Insertion (Example 1):

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -1



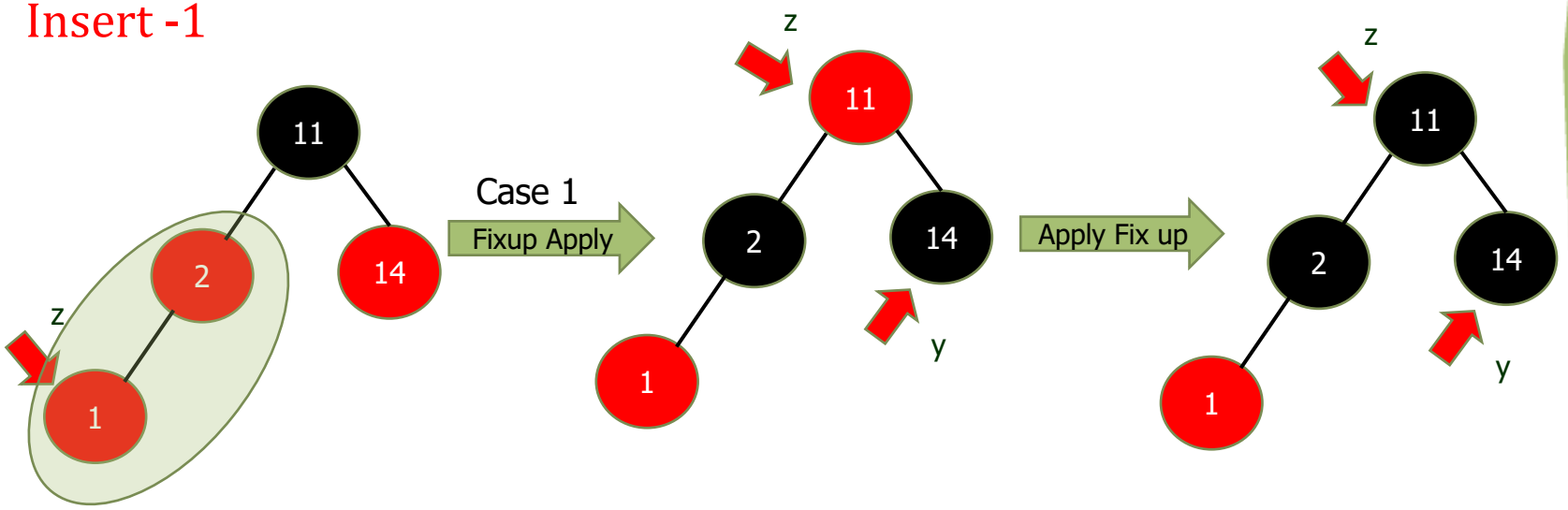
# Red Black Tree

## Insertion (Example 1):

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -1



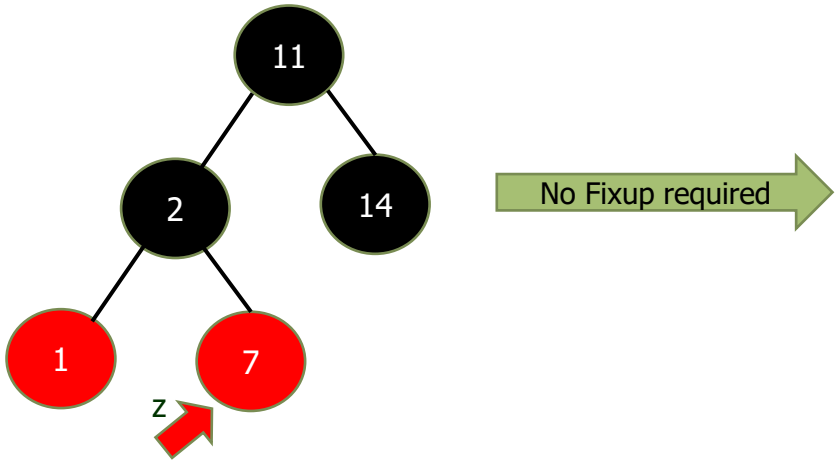
# Red Black Tree

## Insertion (Example 1):

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -7



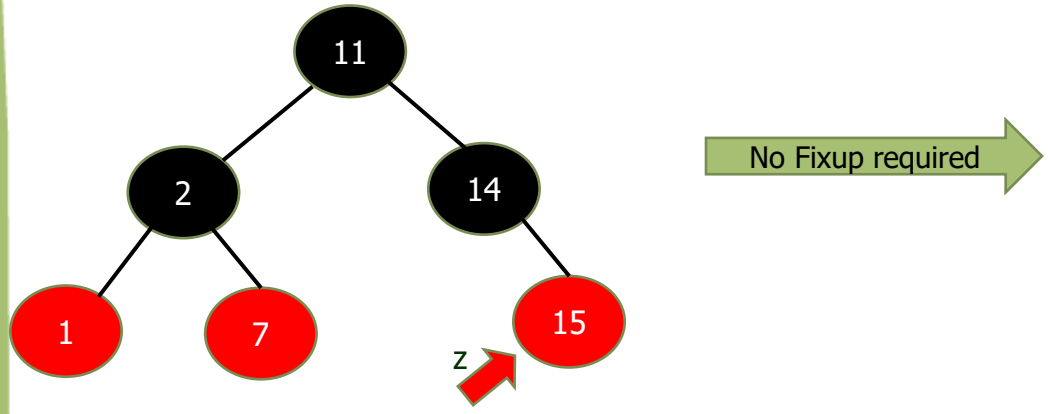
# Red Black Tree

## Insertion (Example 1):

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -15



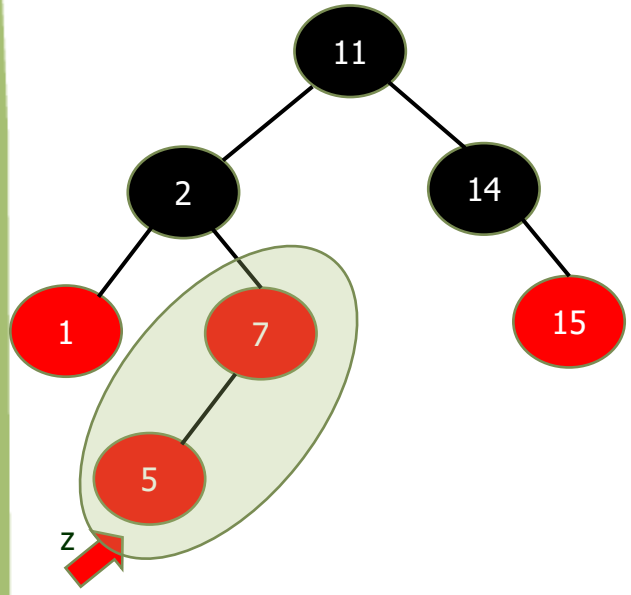
# Red Black Tree

## Insertion (Example 1):

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -5



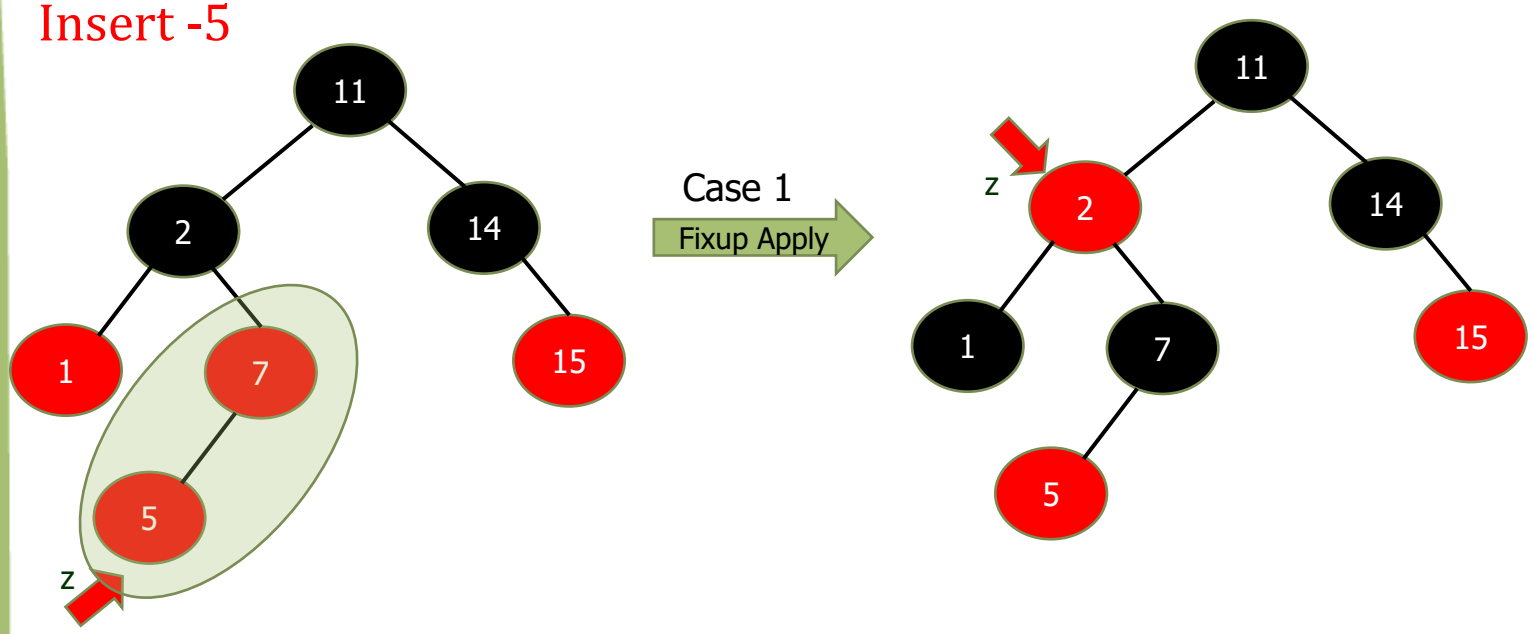
# Red Black Tree

## Insertion (Example 1):

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -5



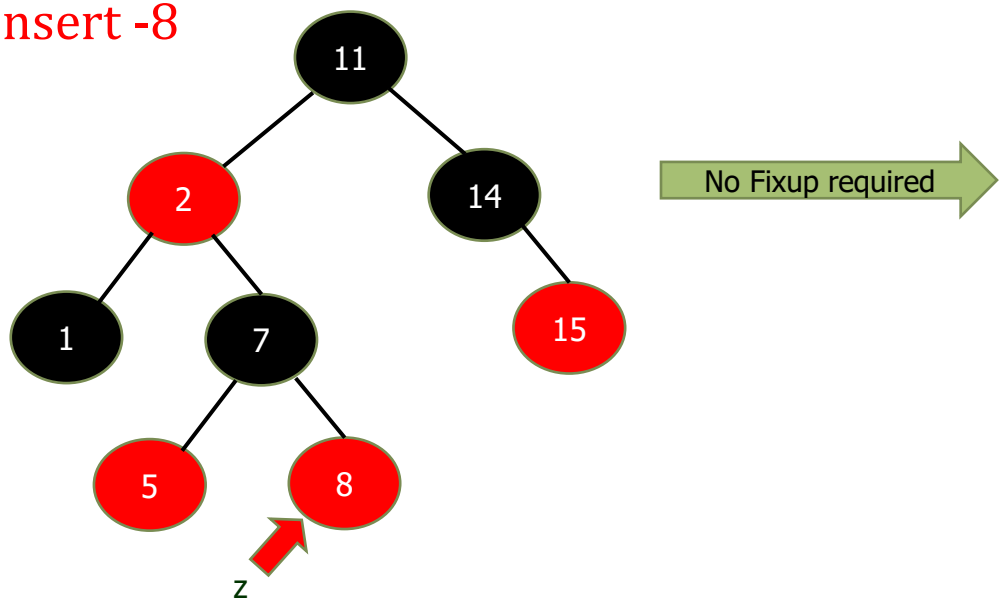
# Red Black Tree

## Insertion (Example 1):

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -8



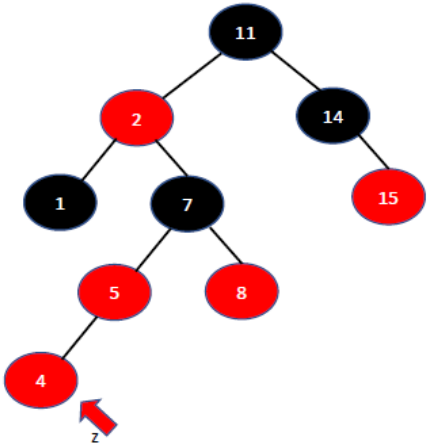
# Red Black Tree

**Insertion (Example 1):**

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -4



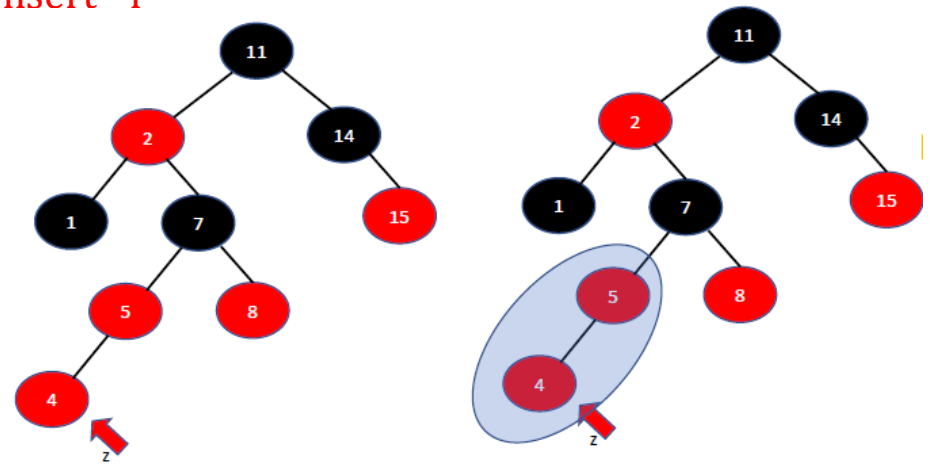
# Red Black Tree

**Insertion (Example 1):**

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -4



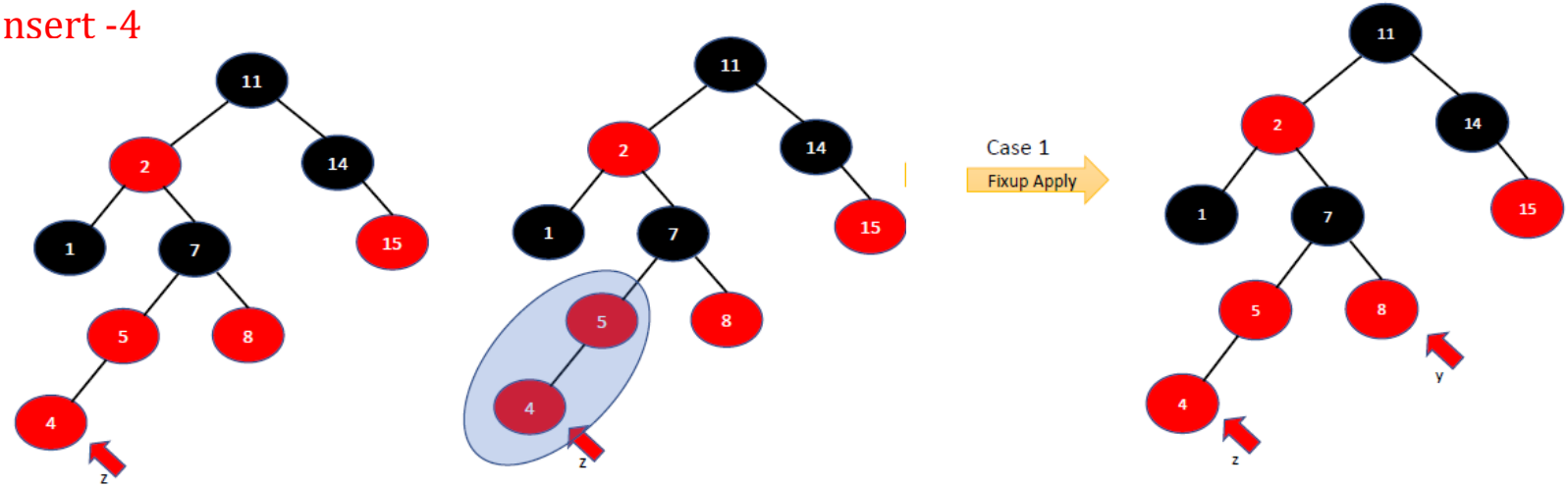
# Red Black Tree

**Insertion (Example 1):**

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -4



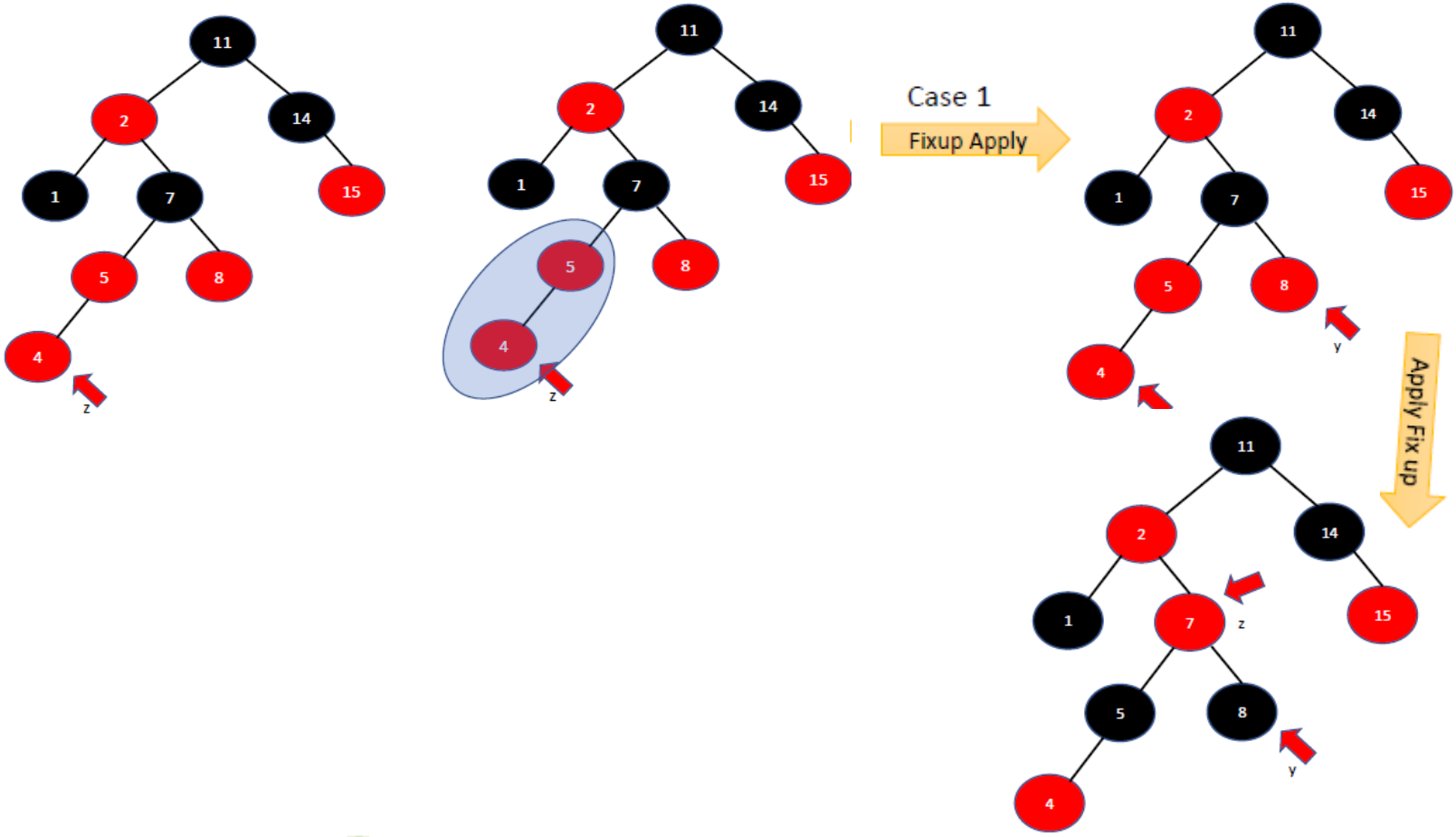
# Red Black Tree

**Insertion (Example 1):**

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -4



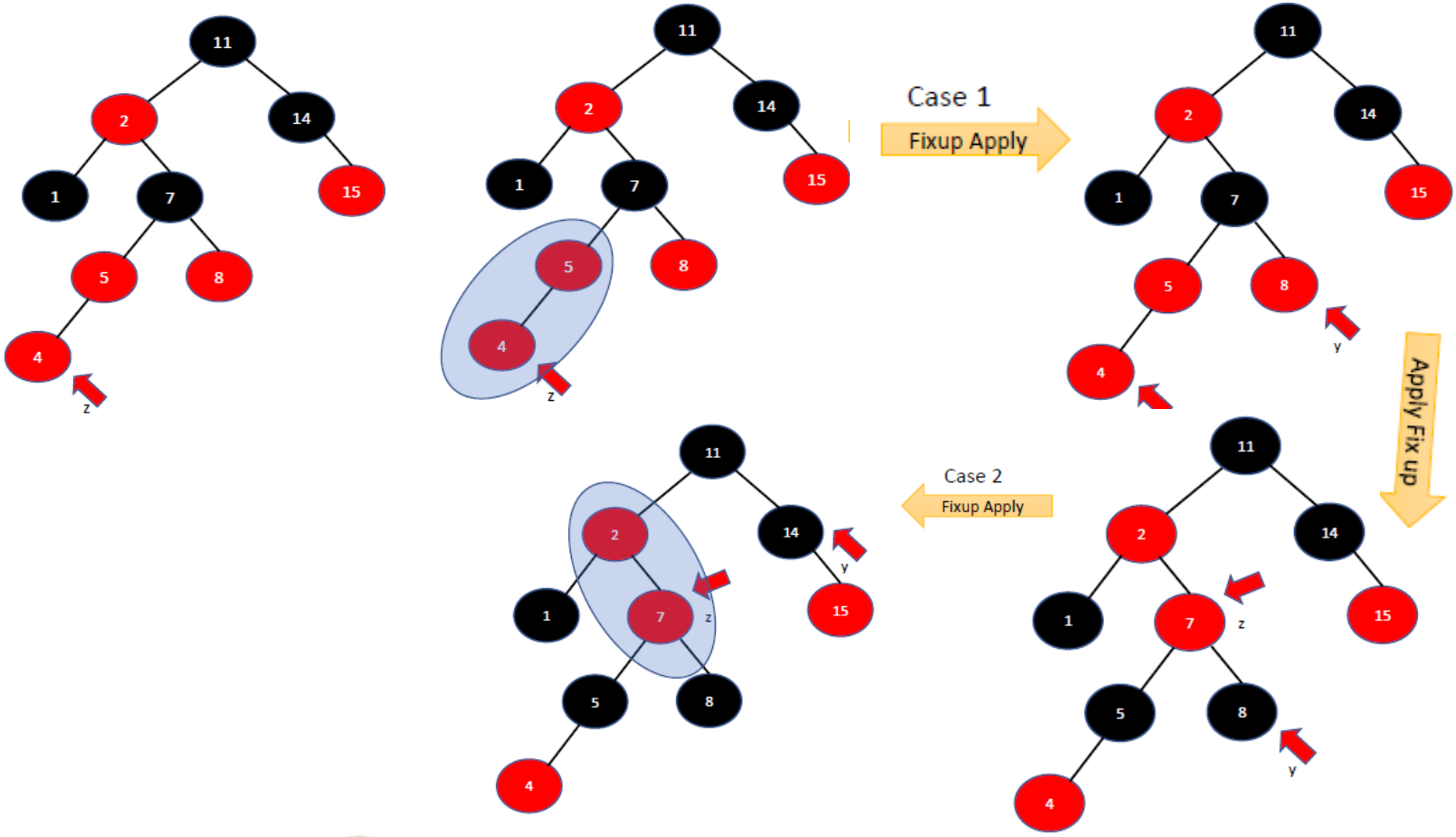
# Red Black Tree

**Insertion (Example):**

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -4



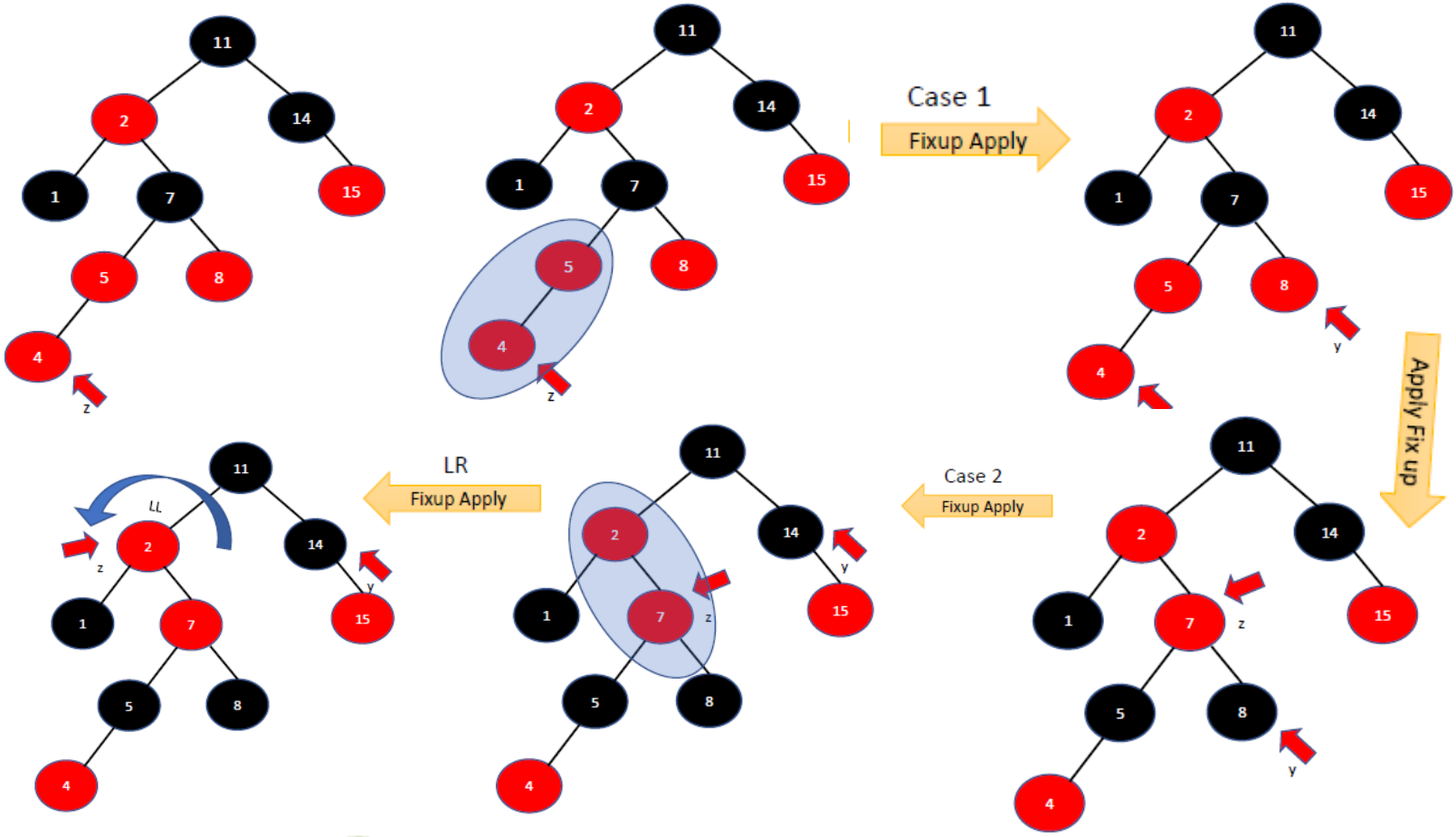
# Red Black Tree

**Insertion (Example 1):**

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -4



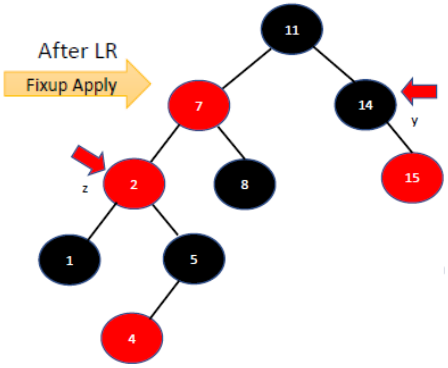
# Red Black Tree

**Insertion (Example 1):**

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -4



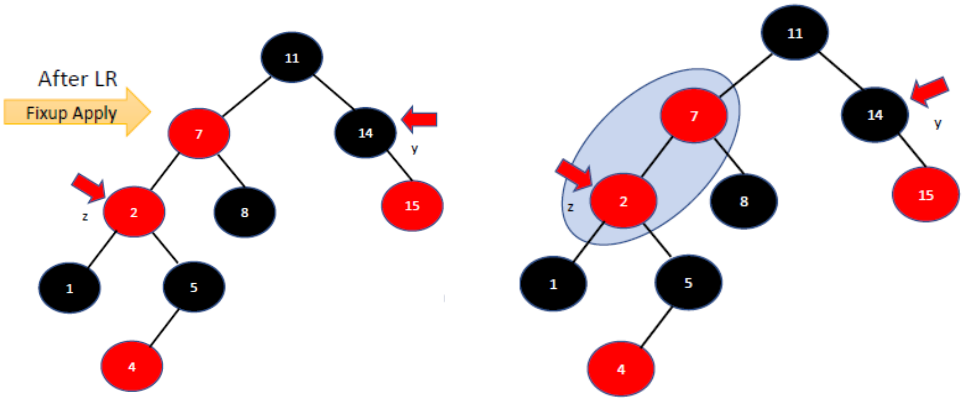
# Red Black Tree

**Insertion (Example 1):**

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -4



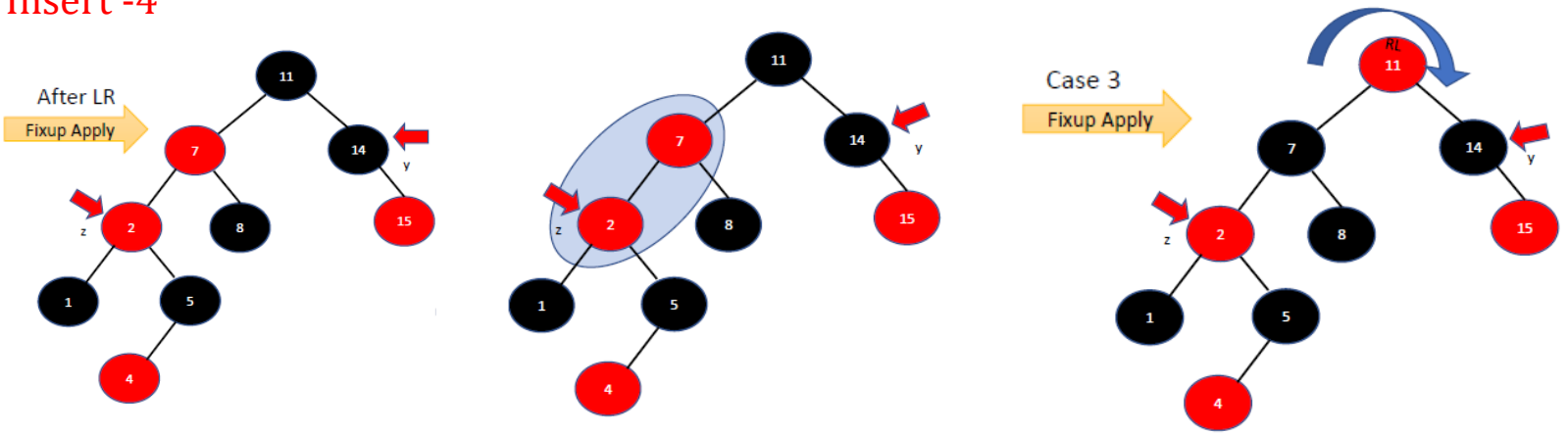
# Red Black Tree

**Insertion (Example 1):**

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -4



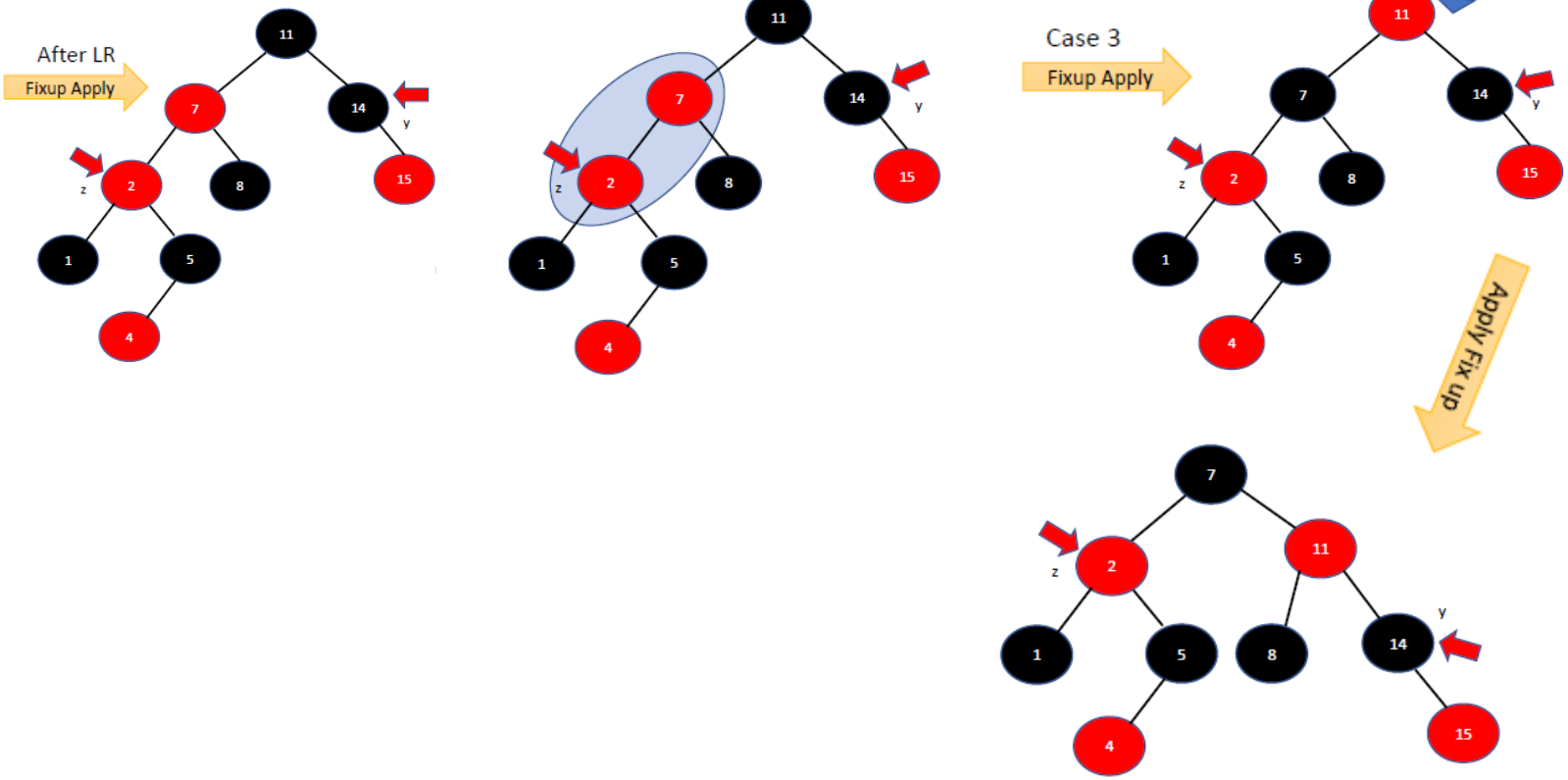
# Red Black Tree

**Insertion (Example 1):**

Insert the following elements into an empty RB-Tree.

[ 11, 2, 14, 1, 7, 15, 5, 8, 4]

Insert -4



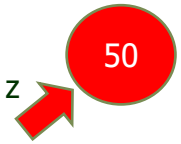
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -50



# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -50

# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -50

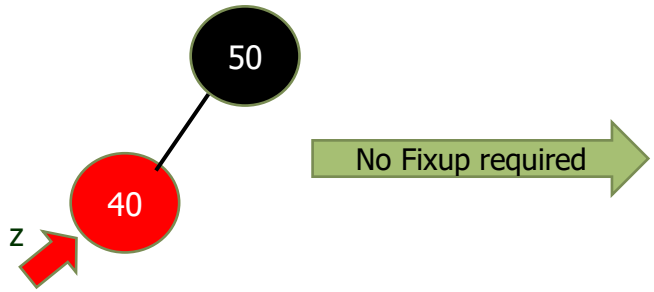
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -40



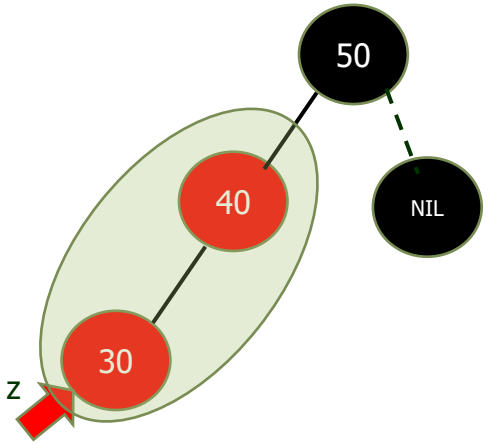
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -30



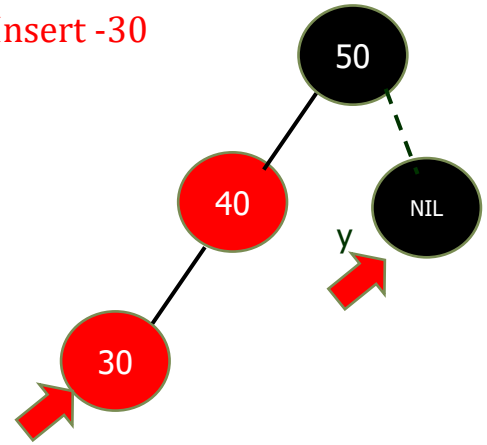
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

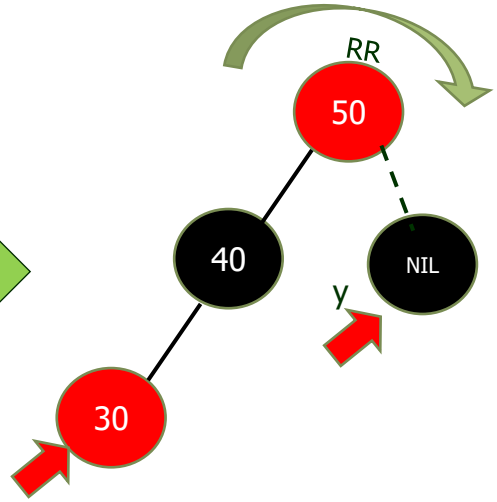
[ 50, 40, 30, 45, 20, 5]

Insert -30

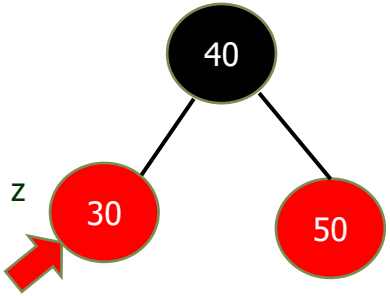



Case 3

Fixup Apply




Fixup Apply After RR



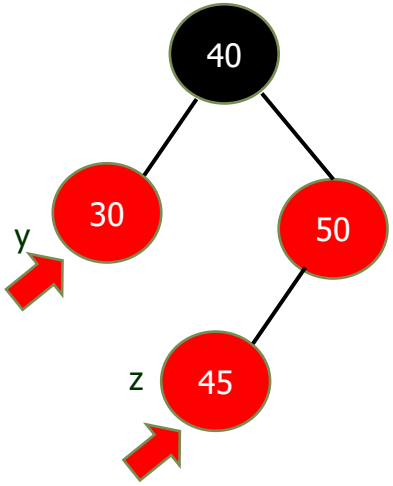
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -45



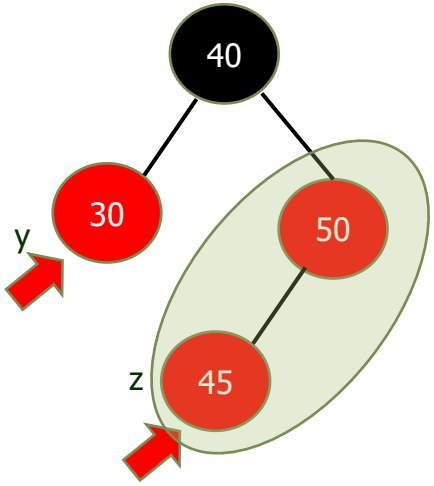
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -45



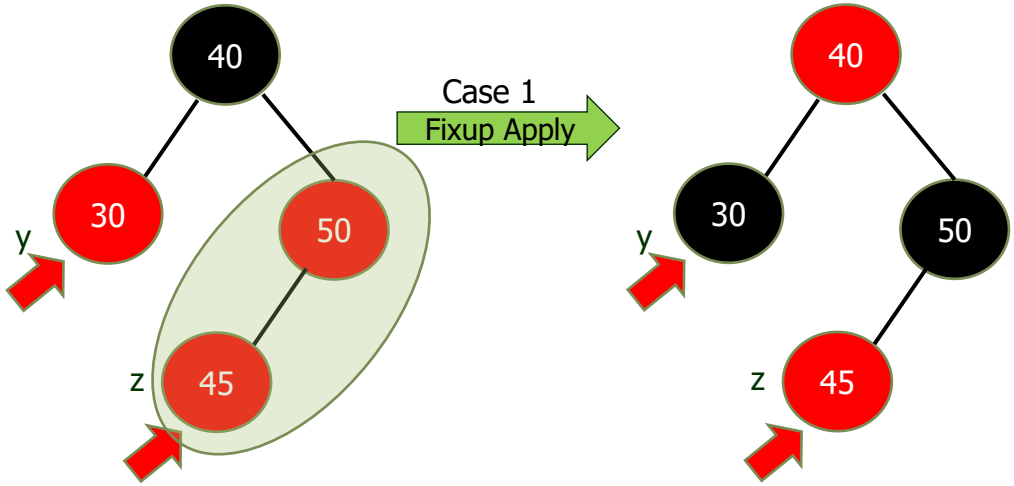
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -45



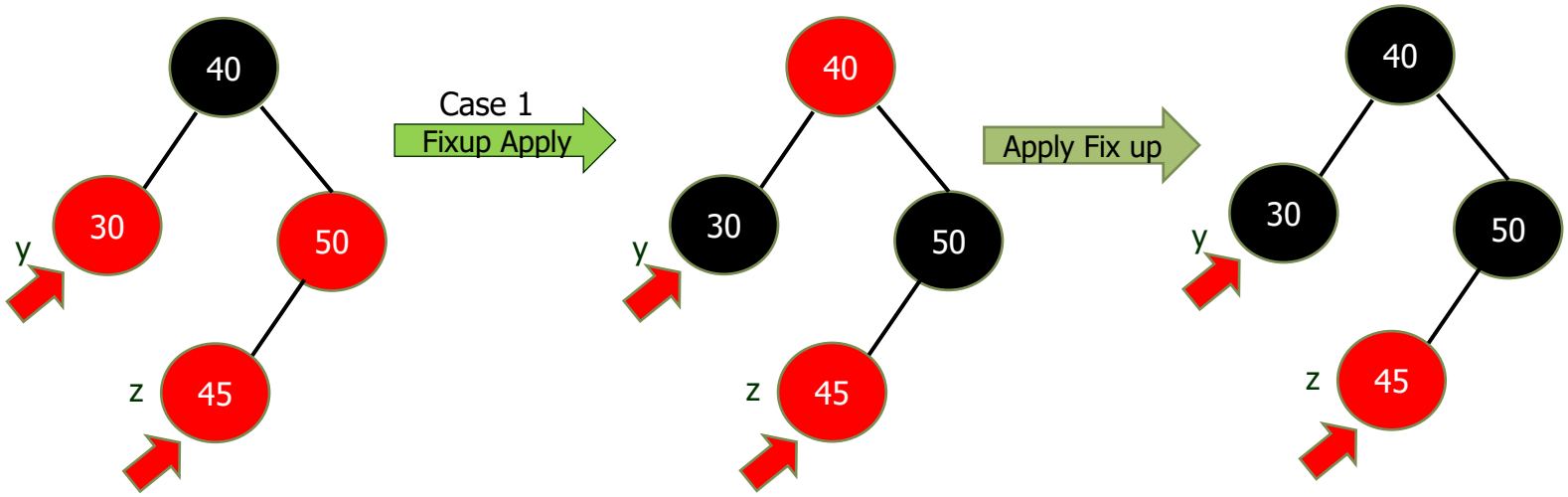
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -45



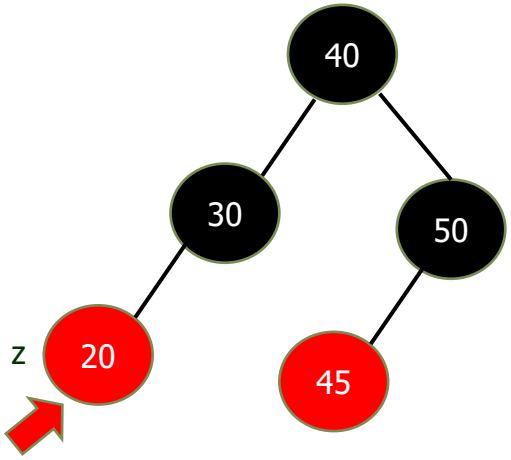
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -20



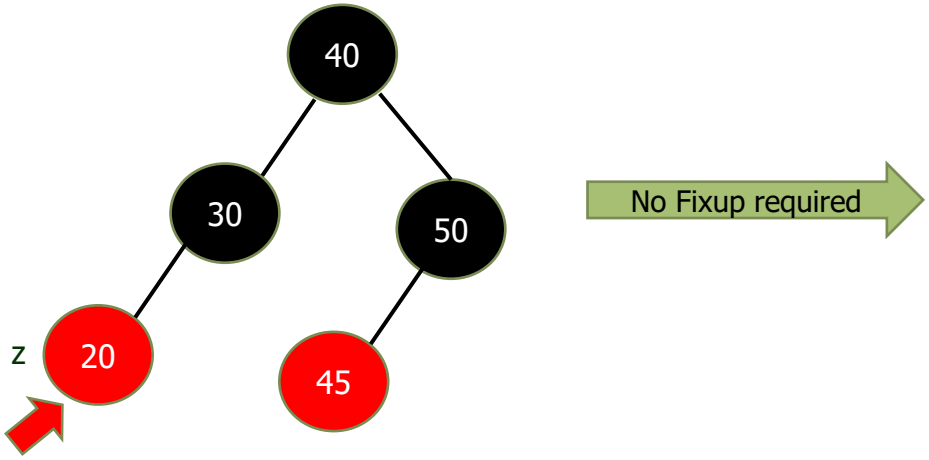
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -20



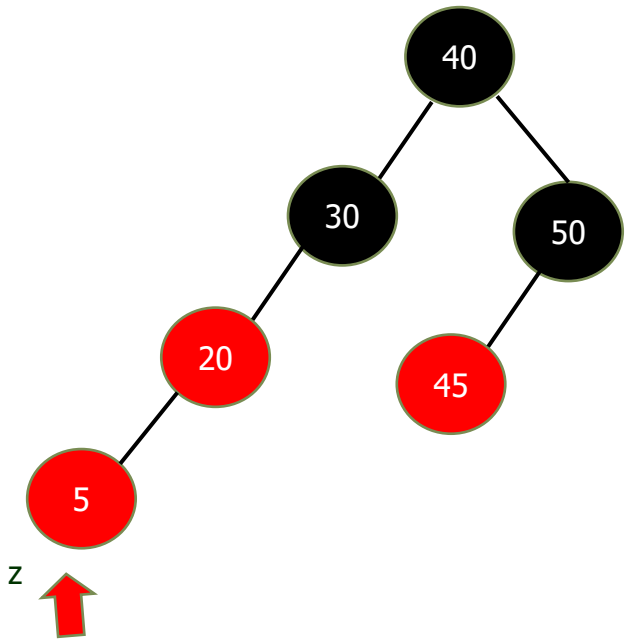
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -5



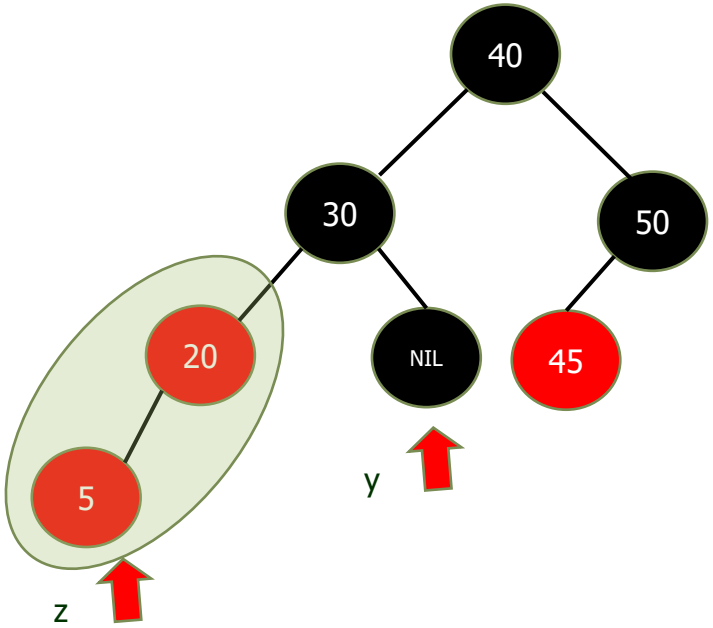
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -5



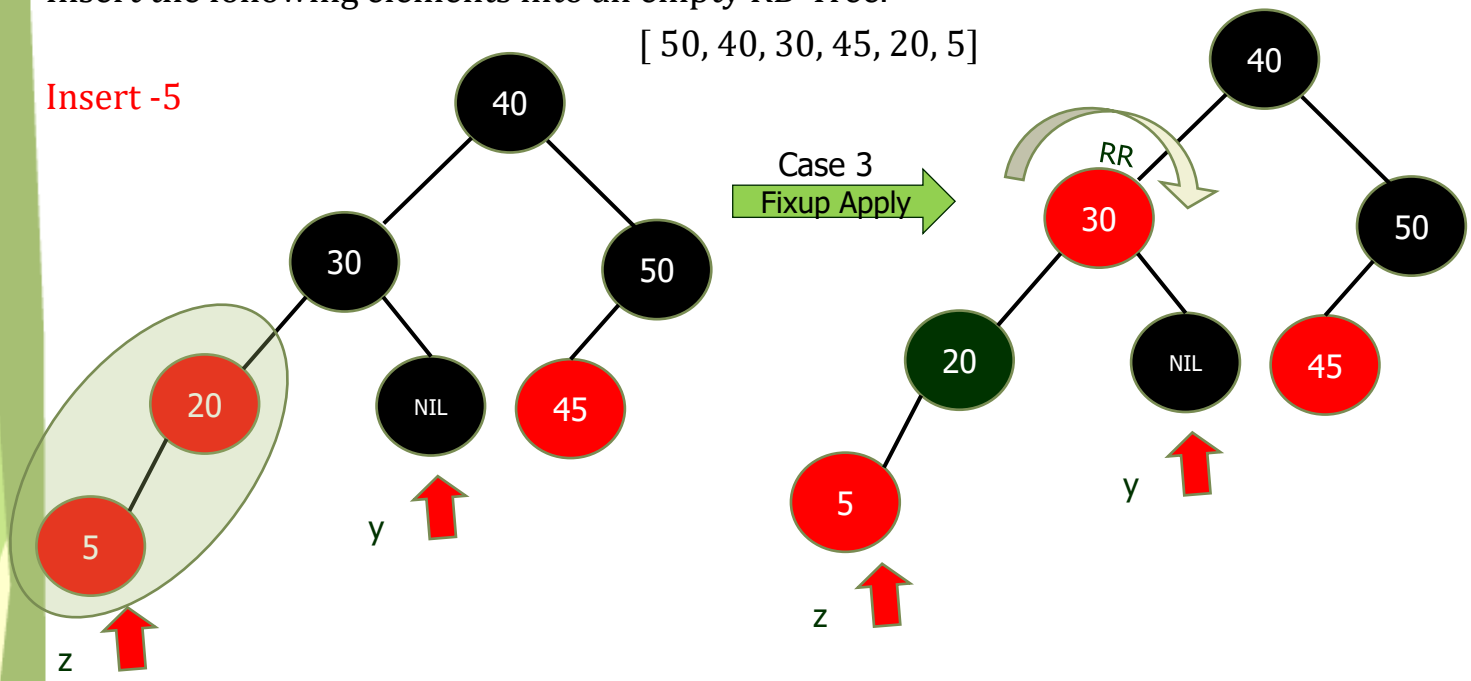
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -5



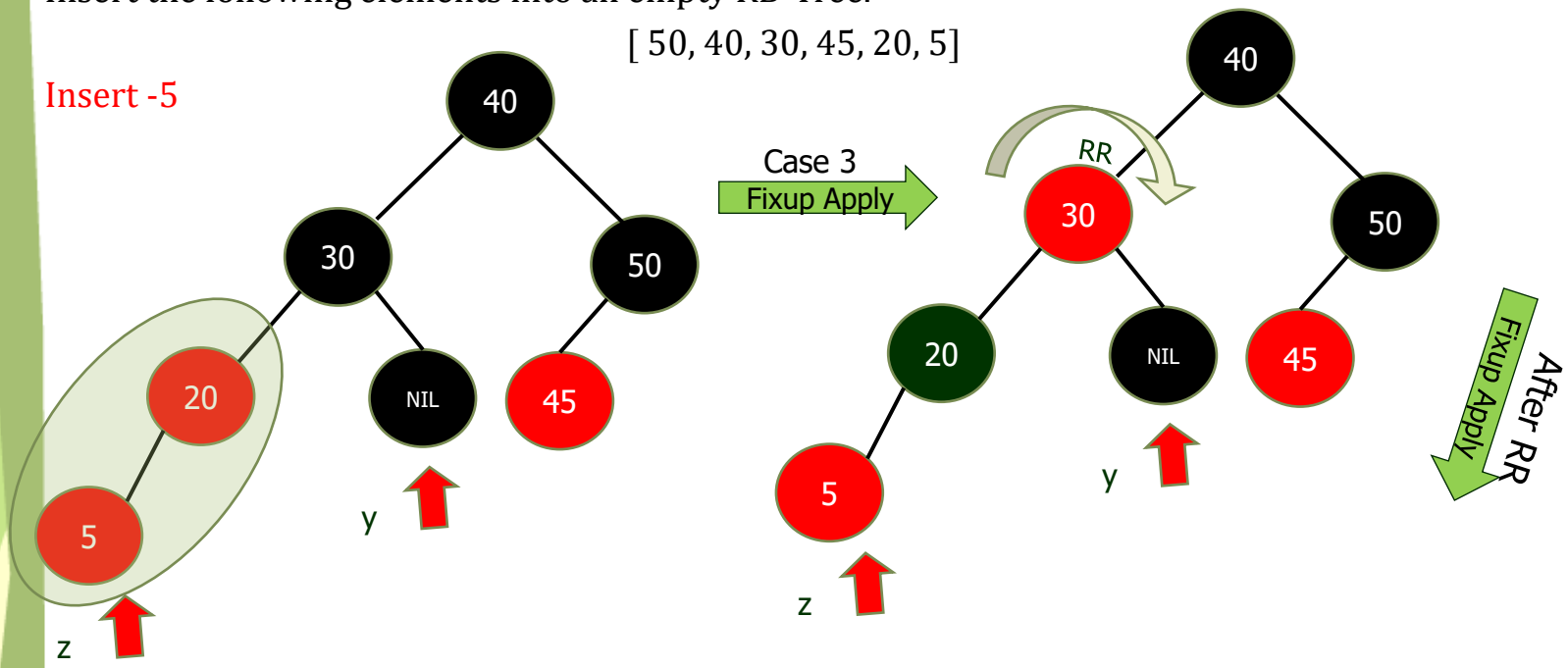
# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]

Insert -5

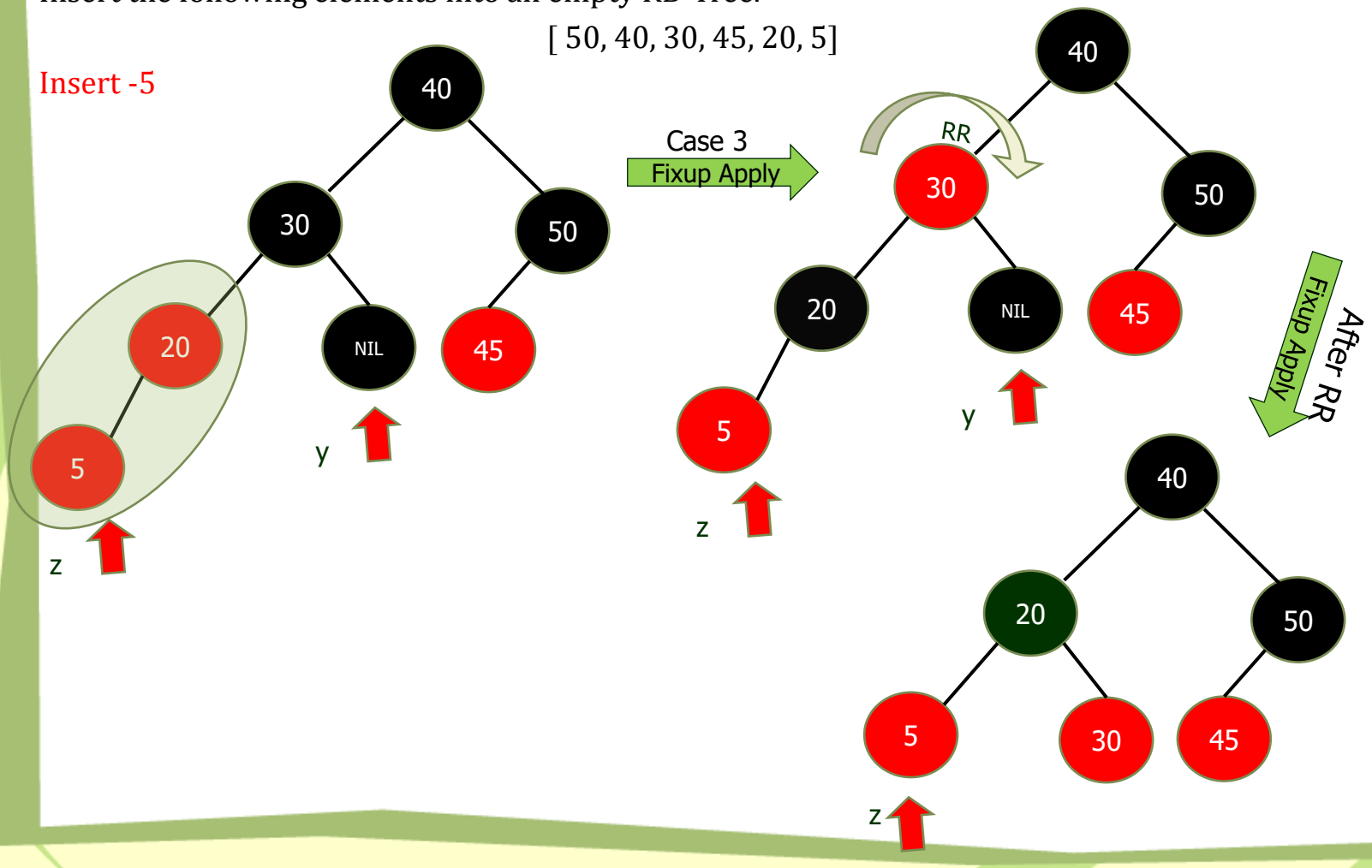


# Red Black Tree

**Insertion (Example 2):**

Insert the following elements into an empty RB-Tree.

[ 50, 40, 30, 45, 20, 5]



# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -41

# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -50

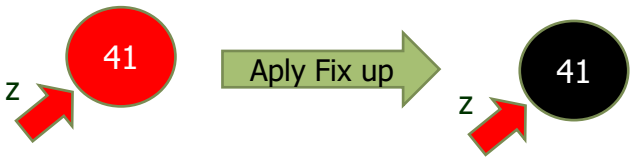
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -50



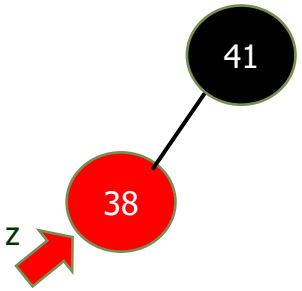
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -38



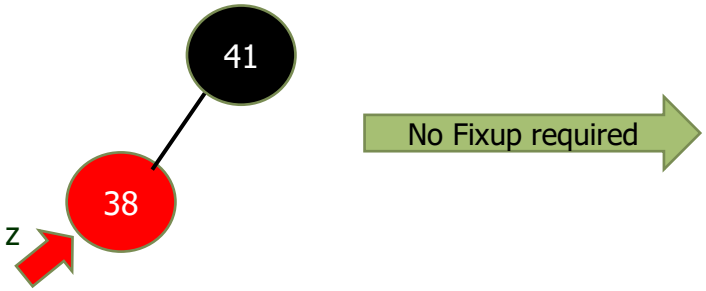
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -38



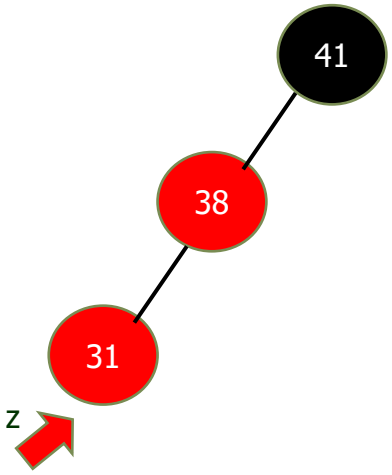
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -31



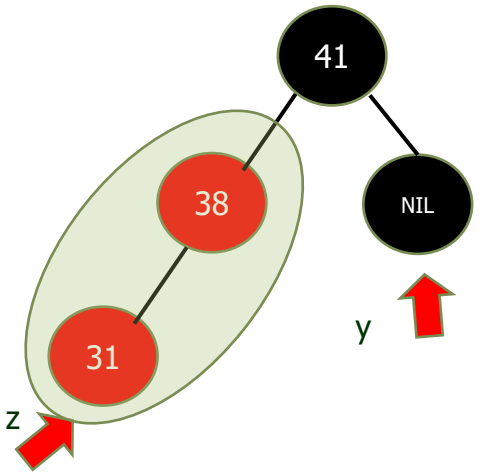
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -31



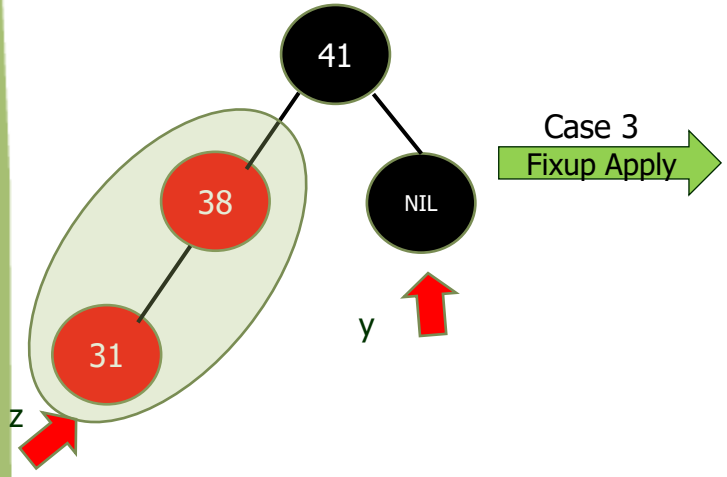
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -31



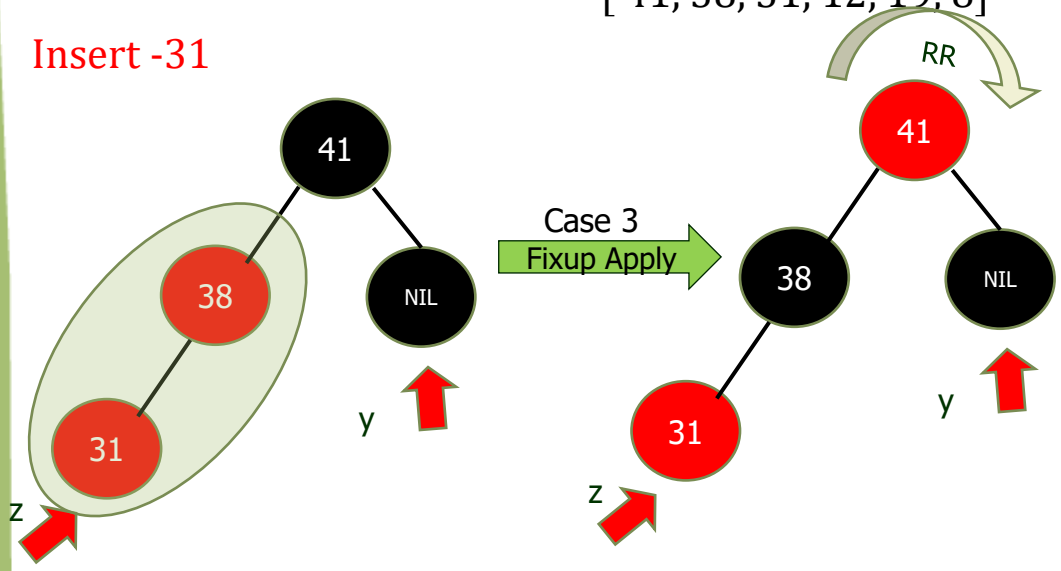
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -31



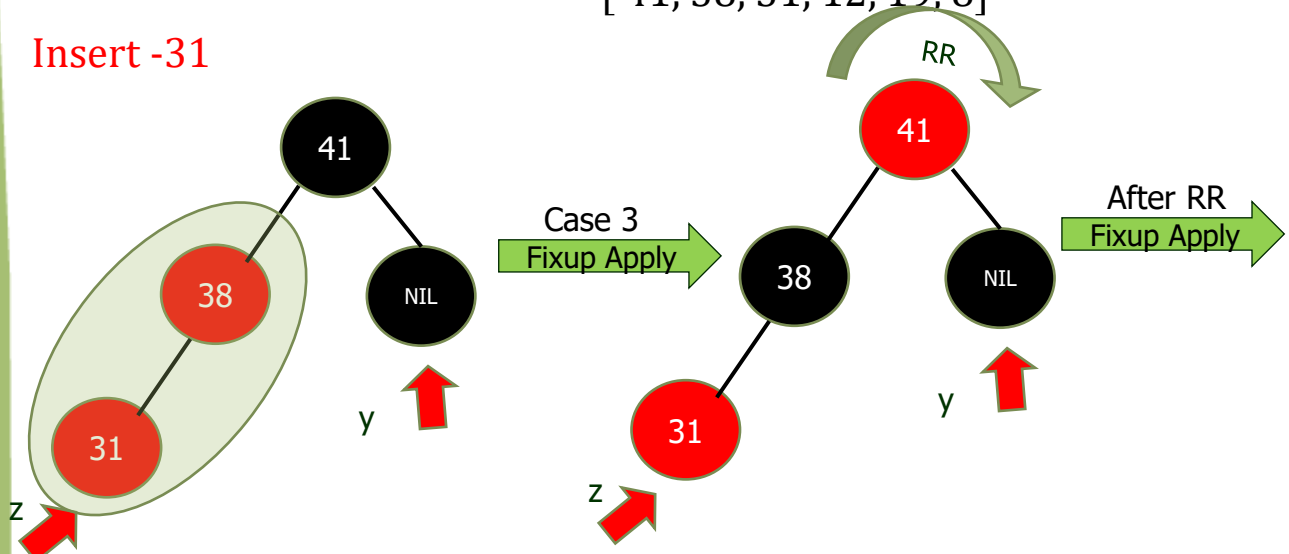
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -31



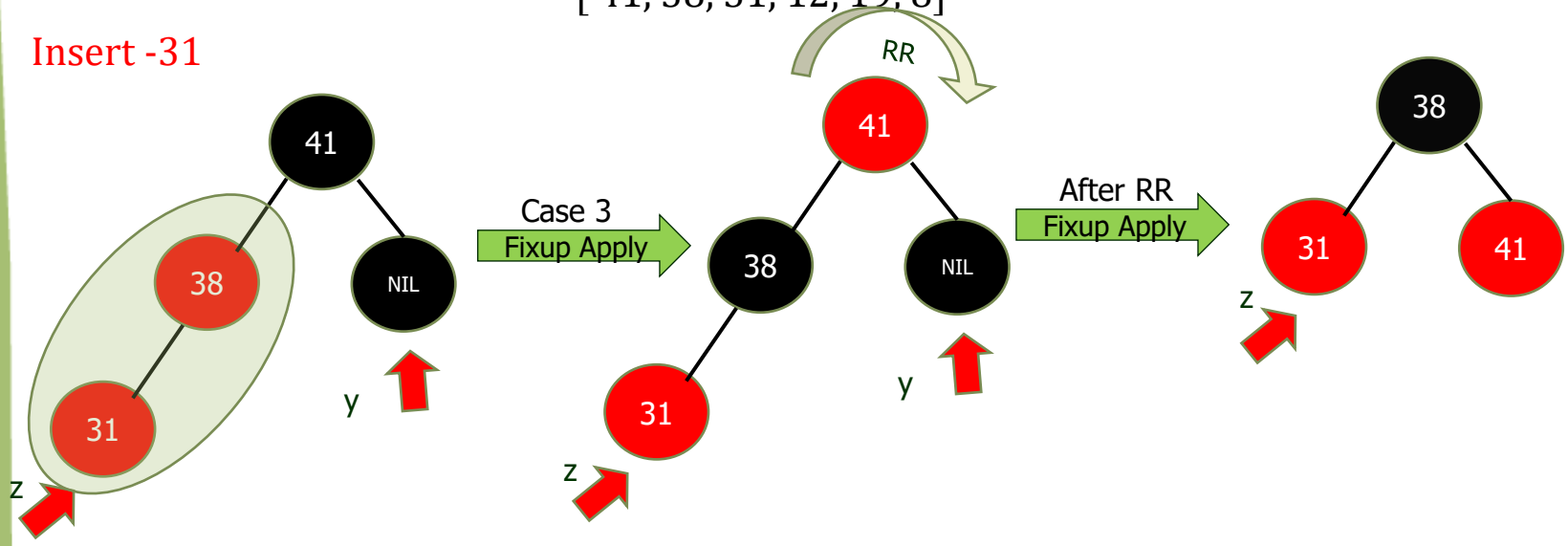
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -31



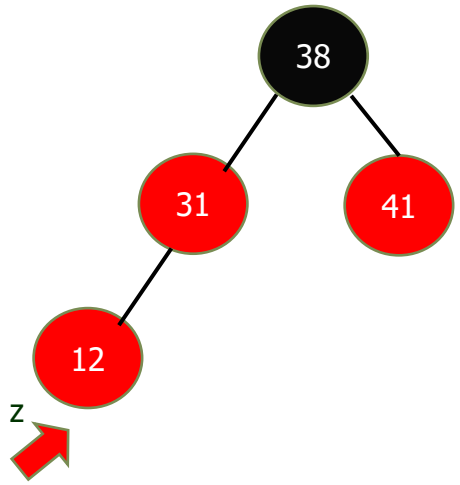
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -12



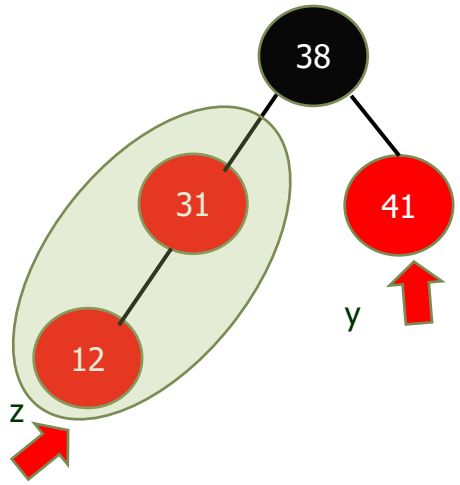
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -12



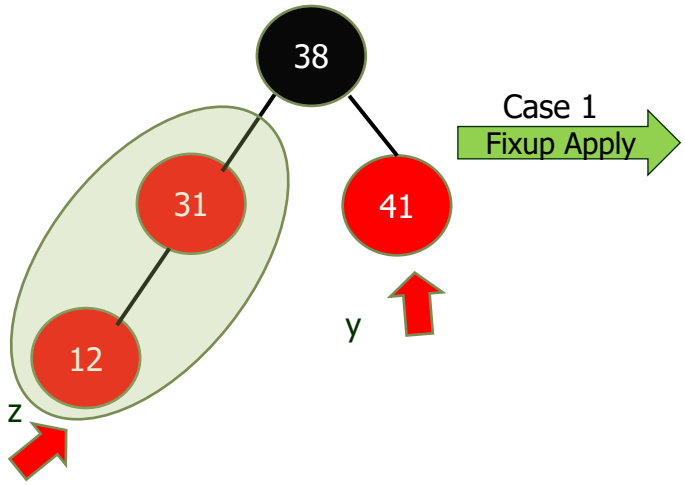
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -12



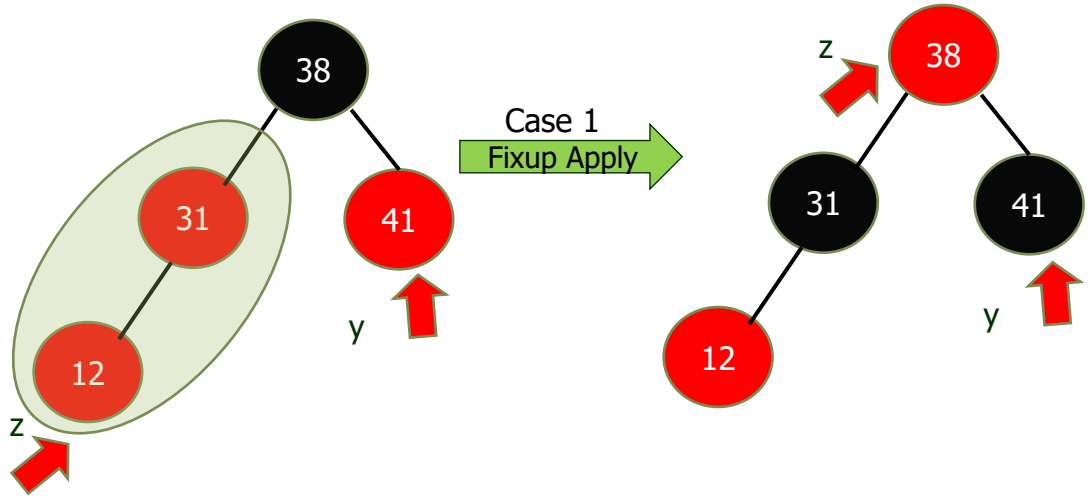
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -12



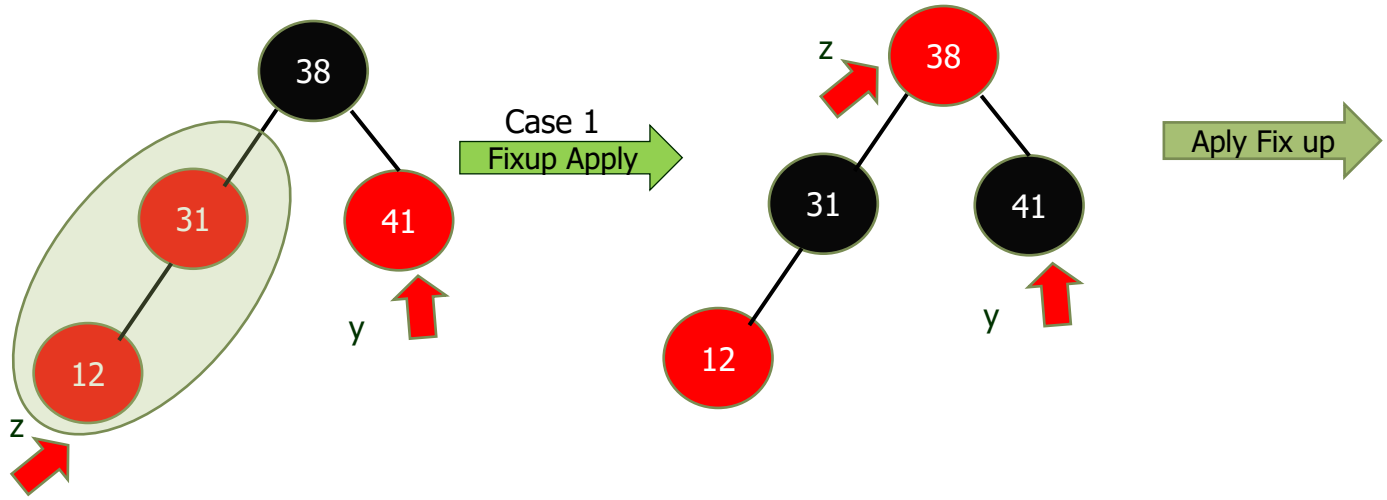
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -12



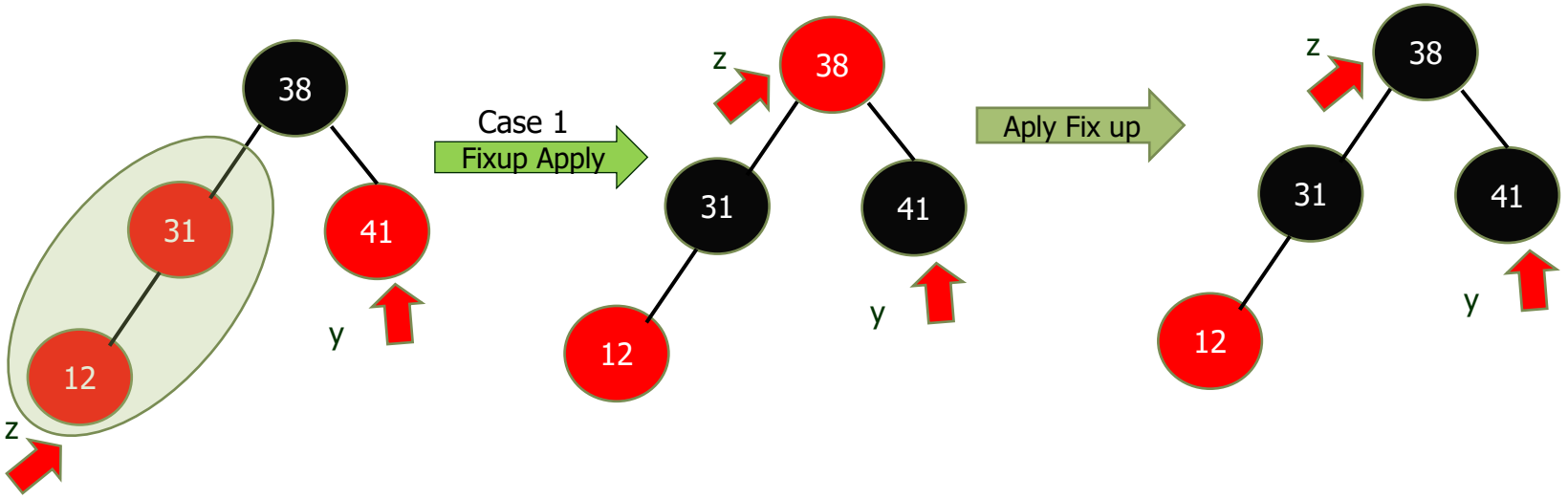
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -12



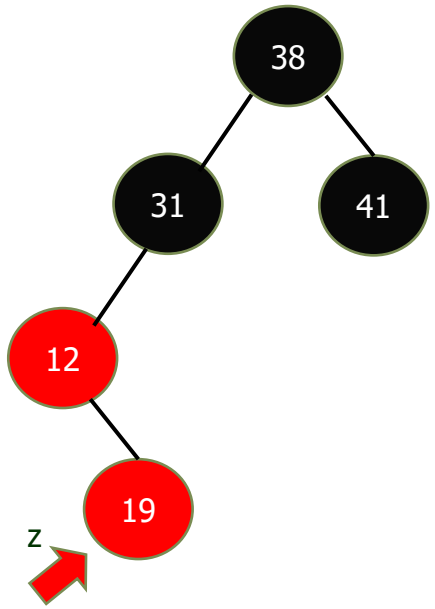
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -19



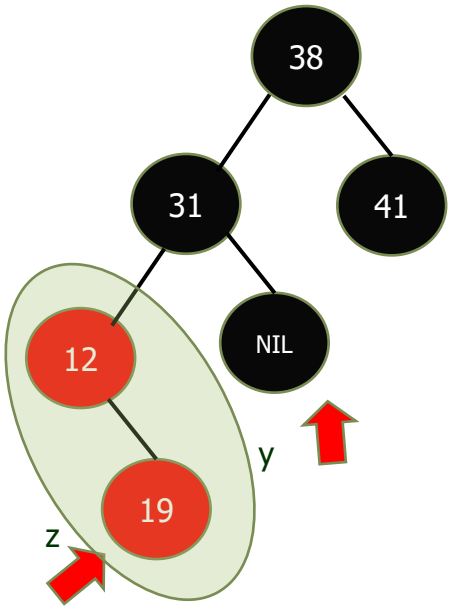
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -19



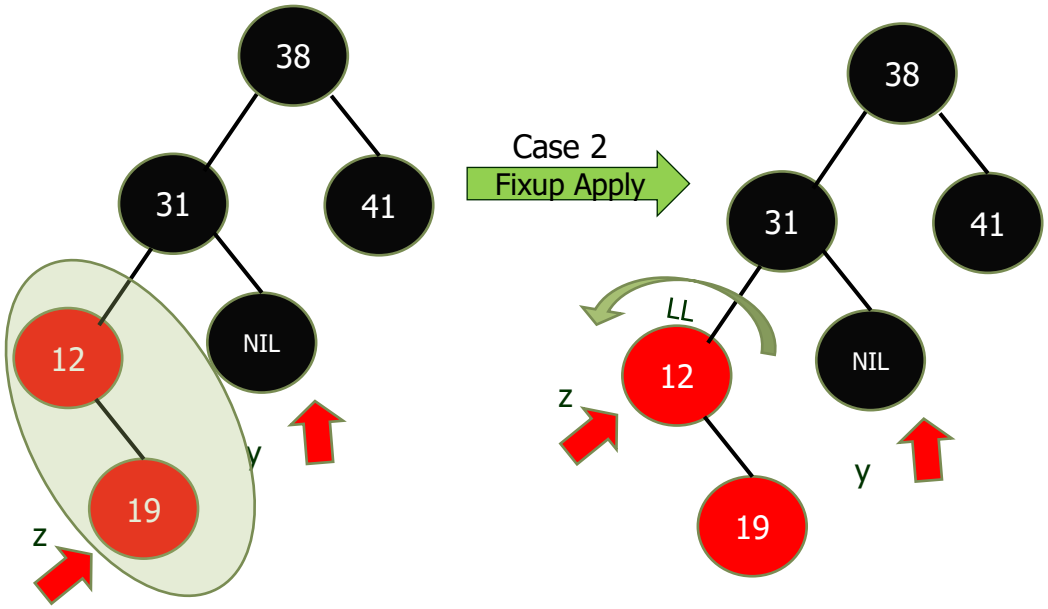
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -19



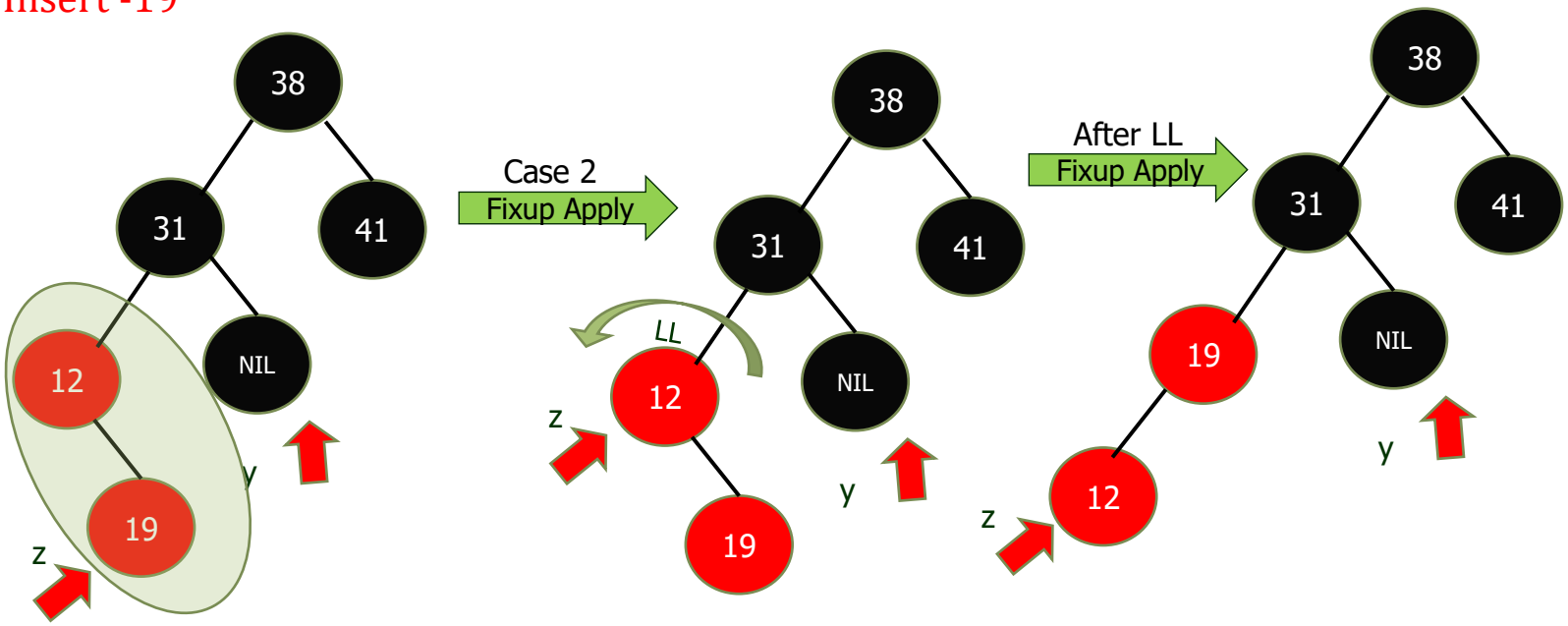
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -19



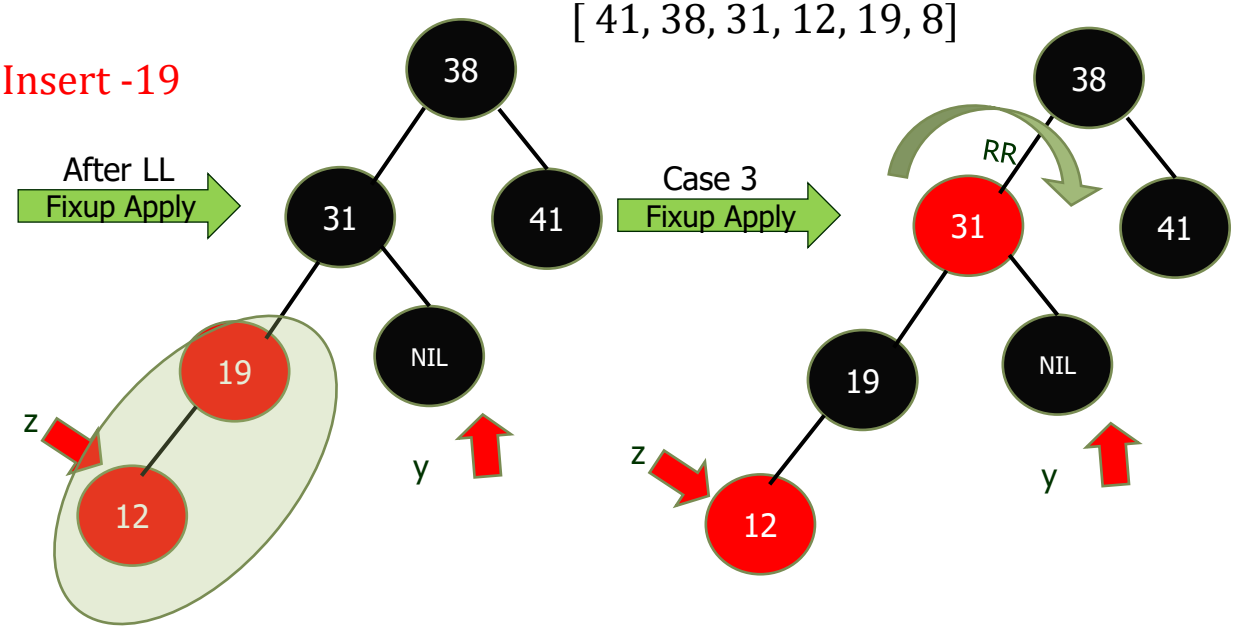
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -19

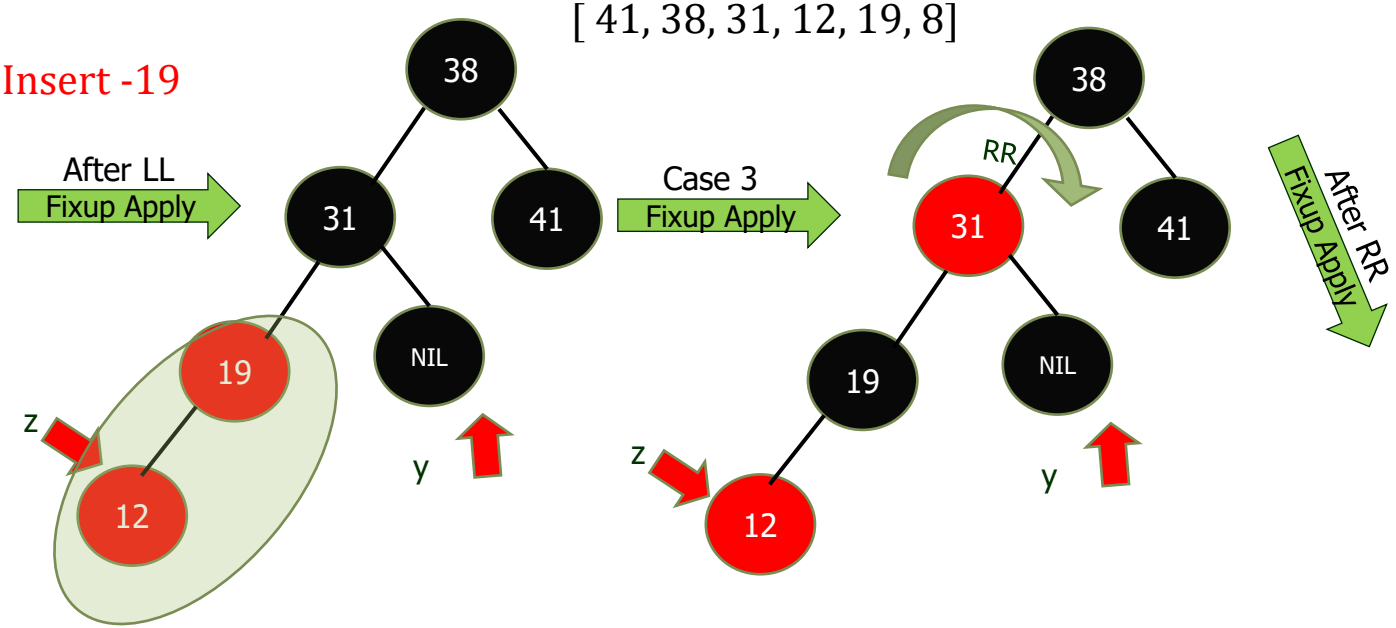


# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

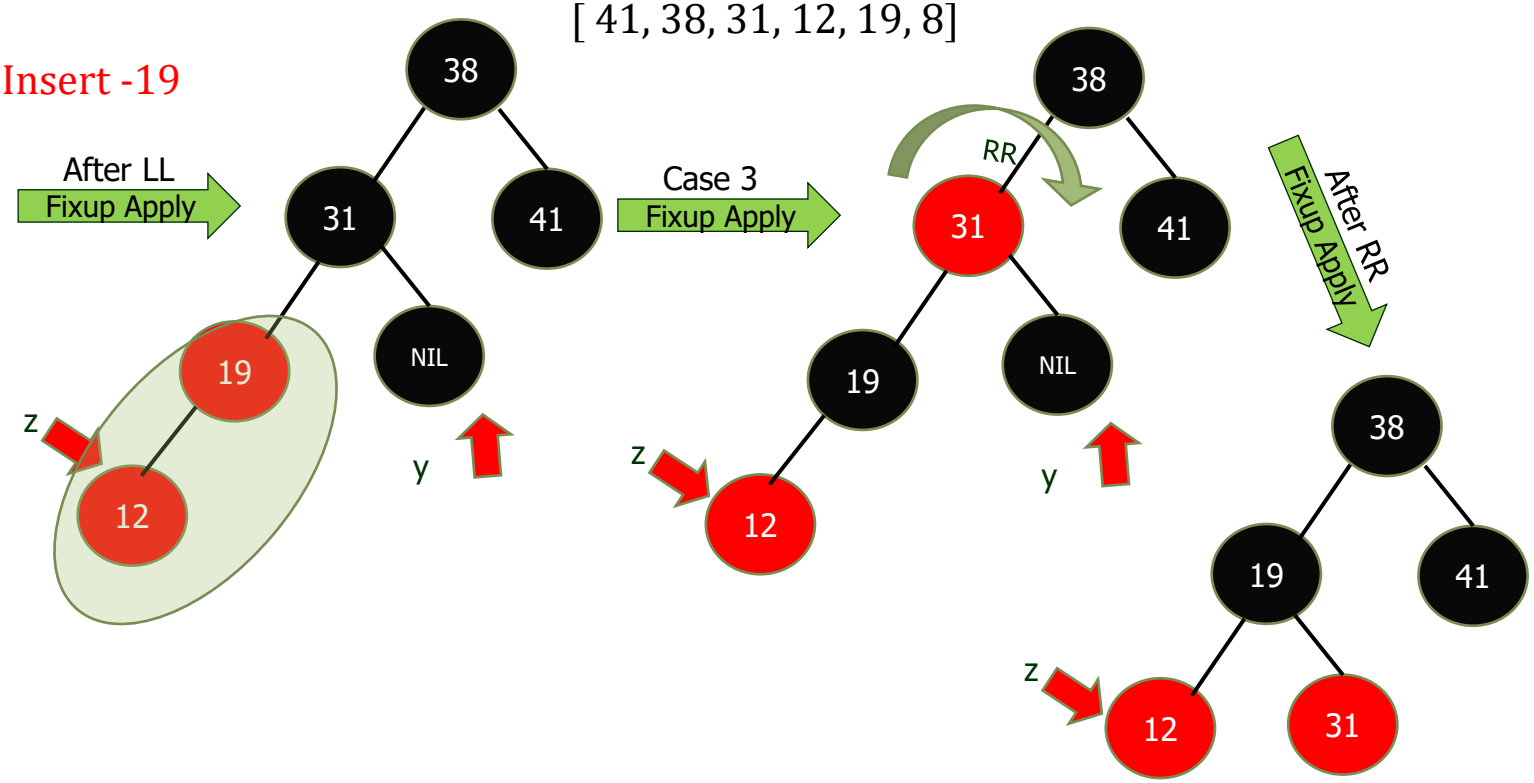


# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]



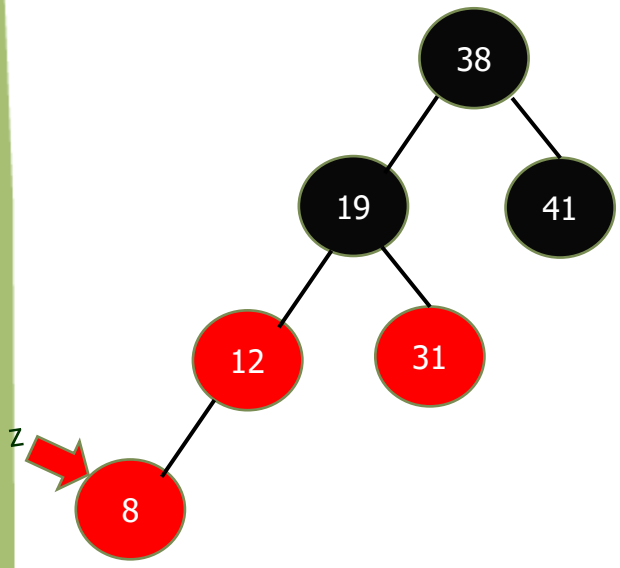
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -8



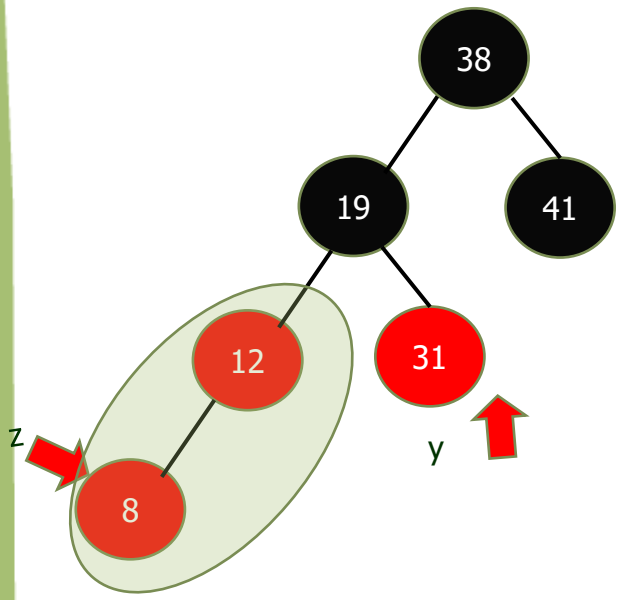
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -8



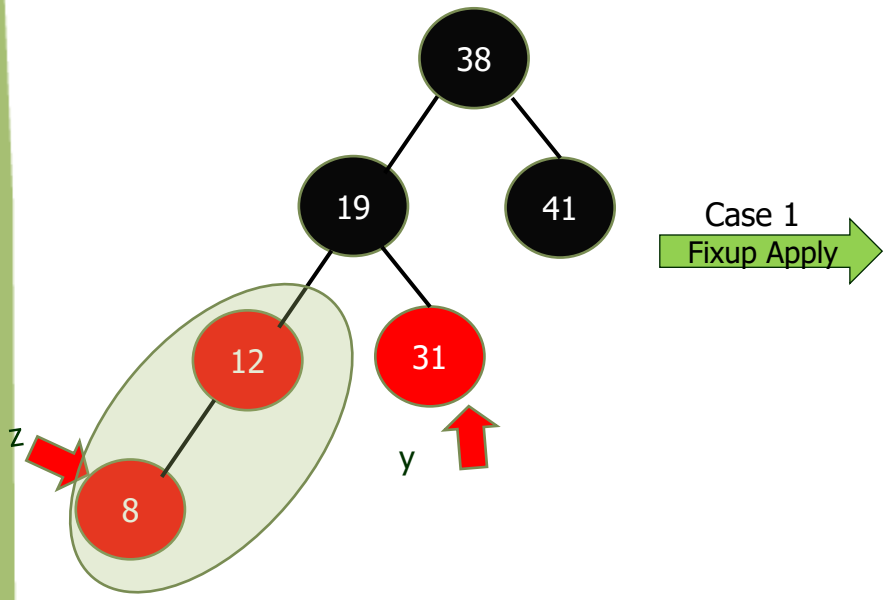
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -8



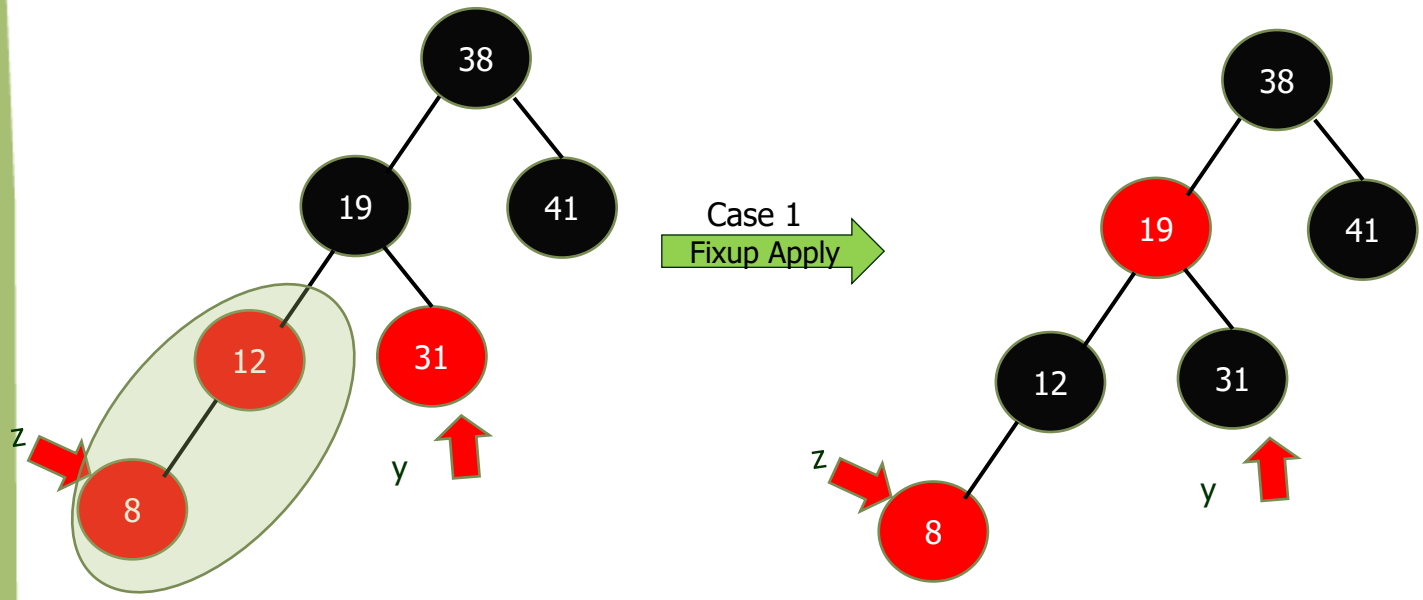
# Red Black Tree

**Insertion (Example 3):**

Insert the following elements into an empty RB-Tree.

[ 41, 38, 31, 12, 19, 8]

Insert -8



# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -5

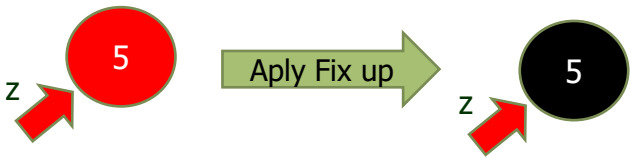
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -5



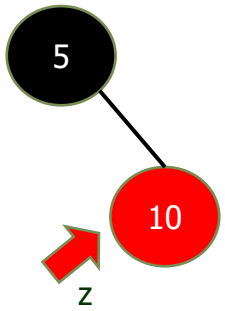
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -10



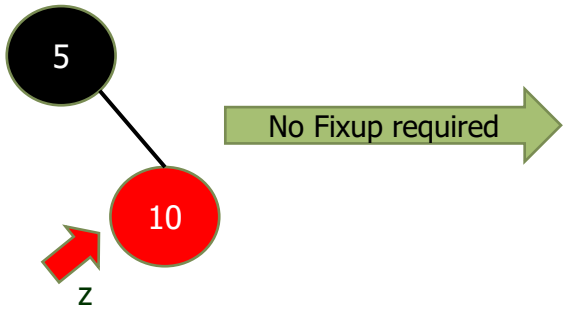
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -10



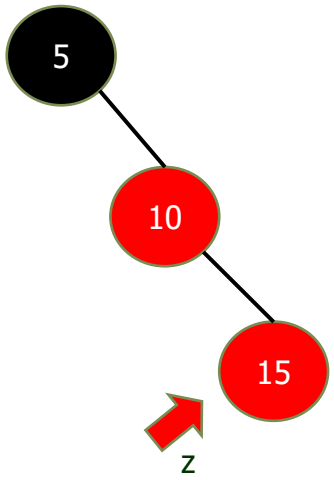
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -15



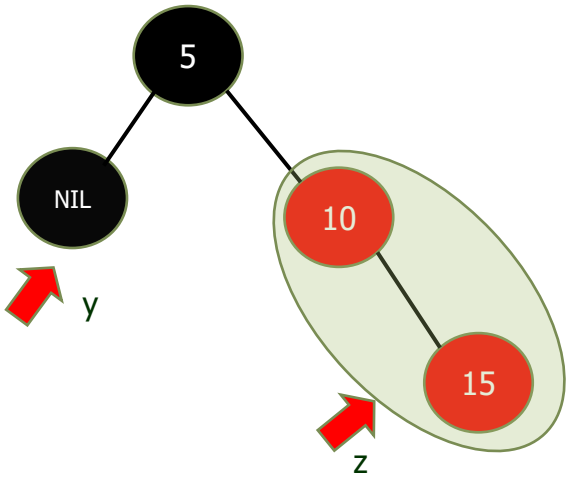
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -15



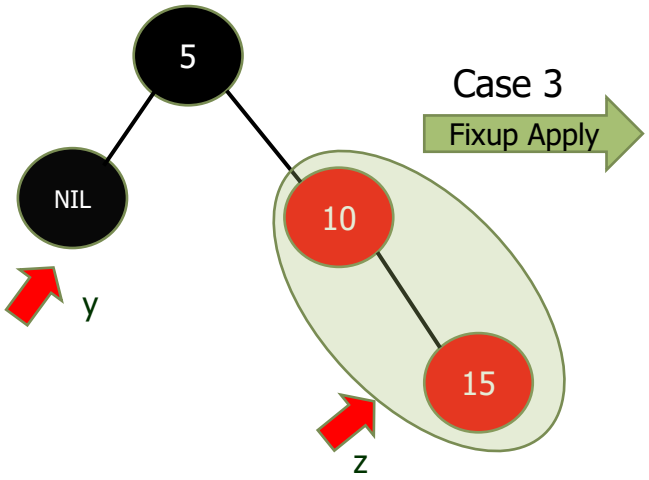
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -15



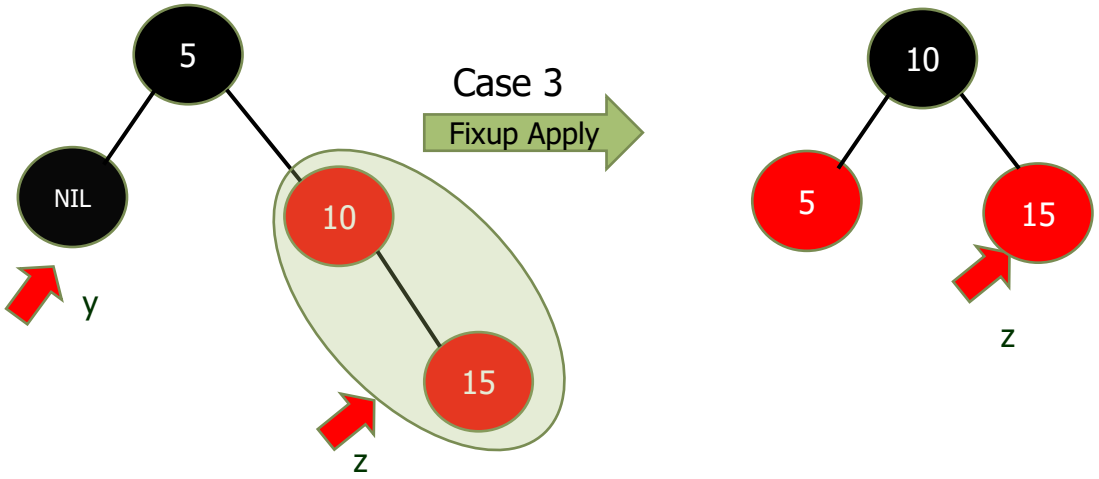
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -15



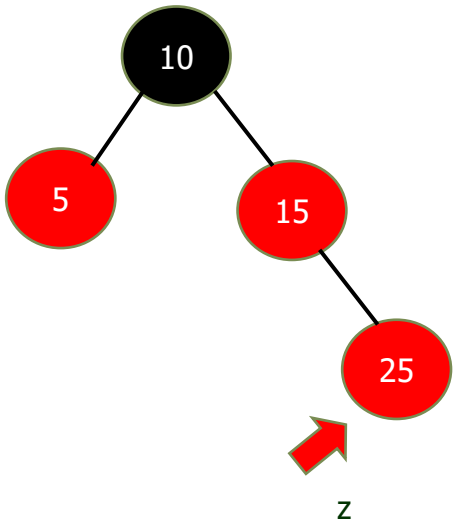
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -25



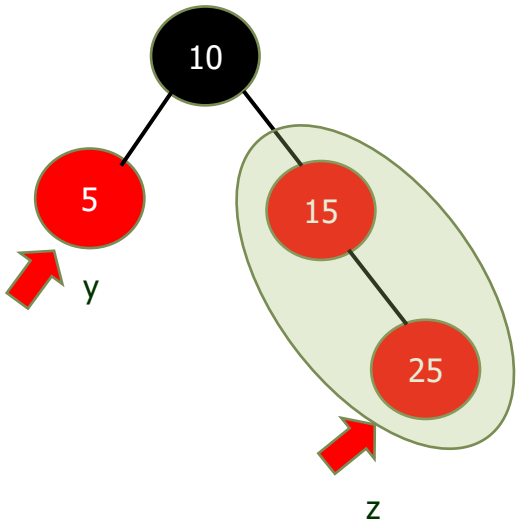
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -25



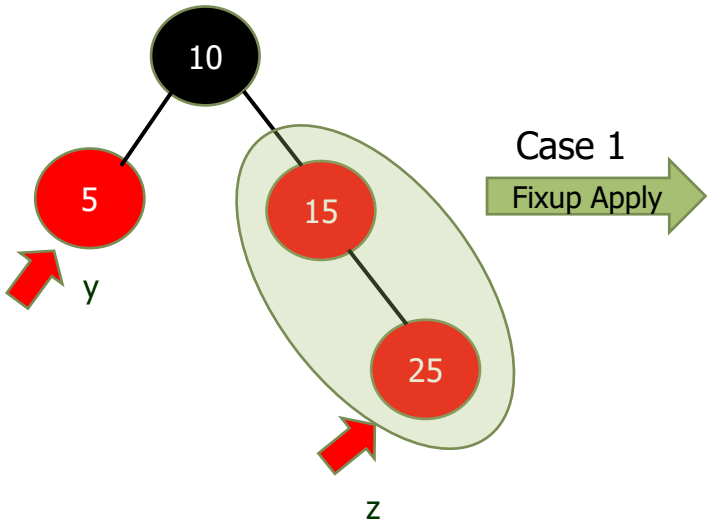
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -25



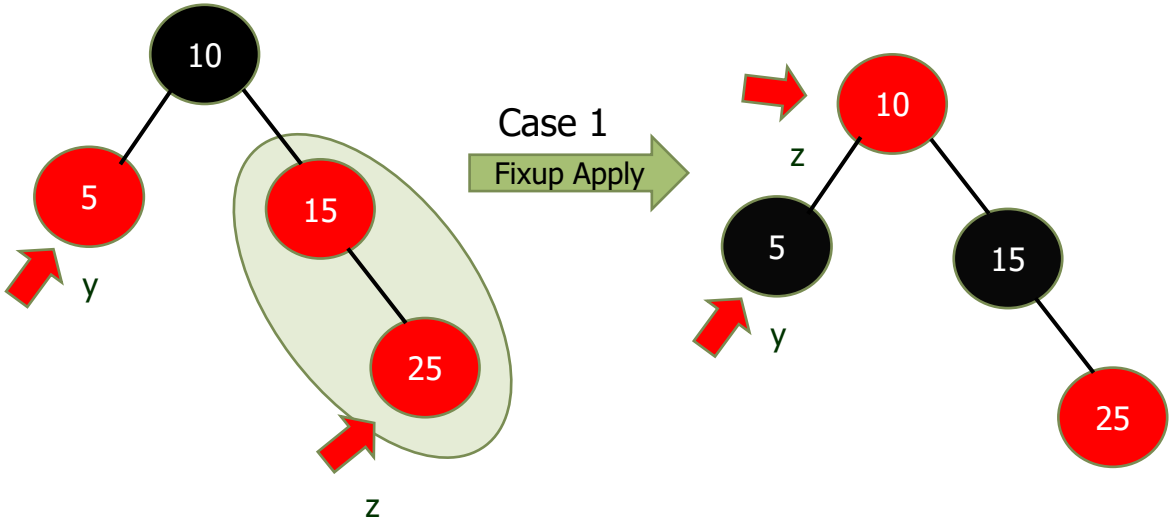
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -25



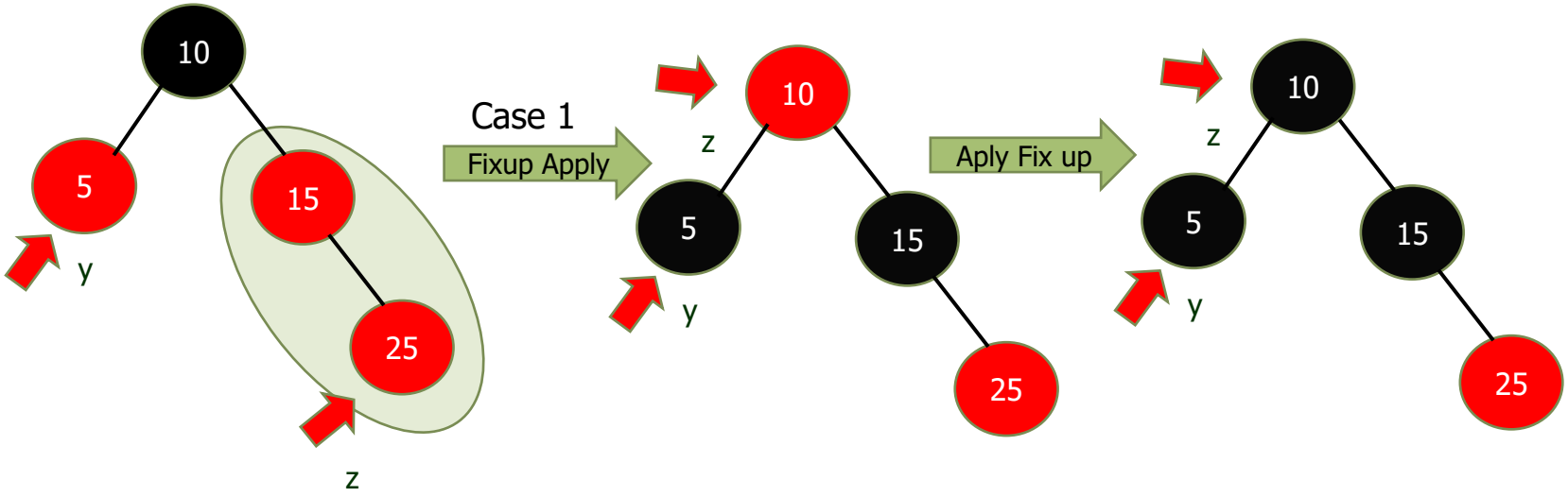
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -25



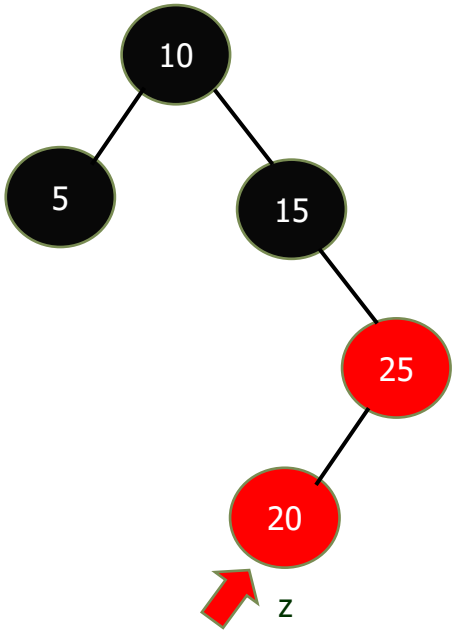
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -20



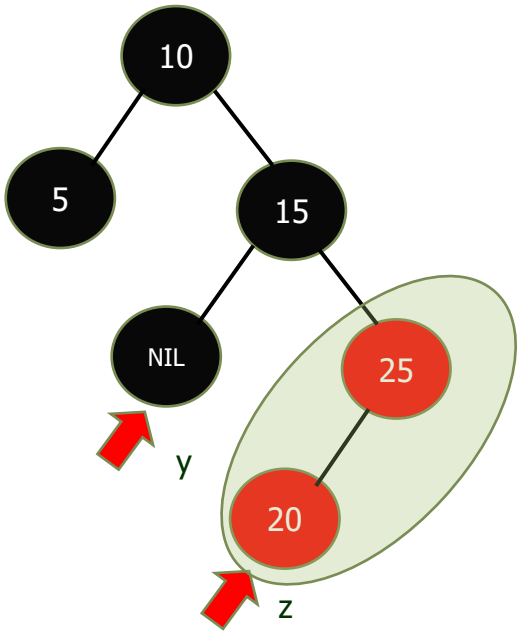
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -20



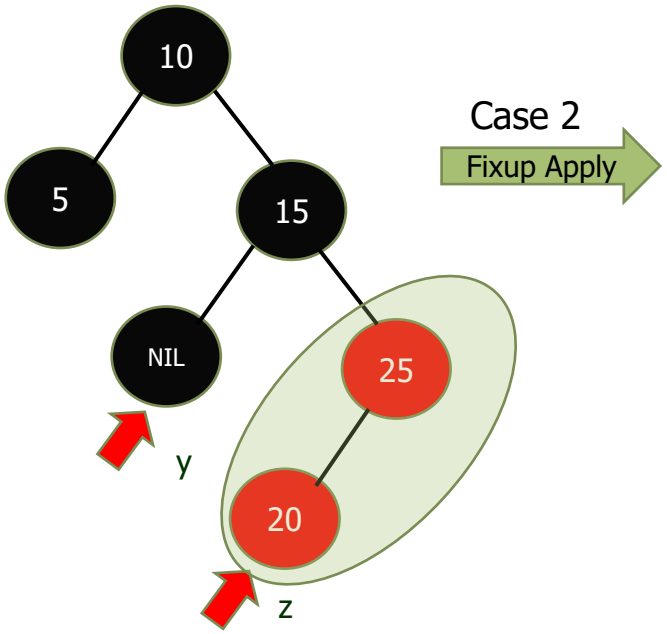
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -20



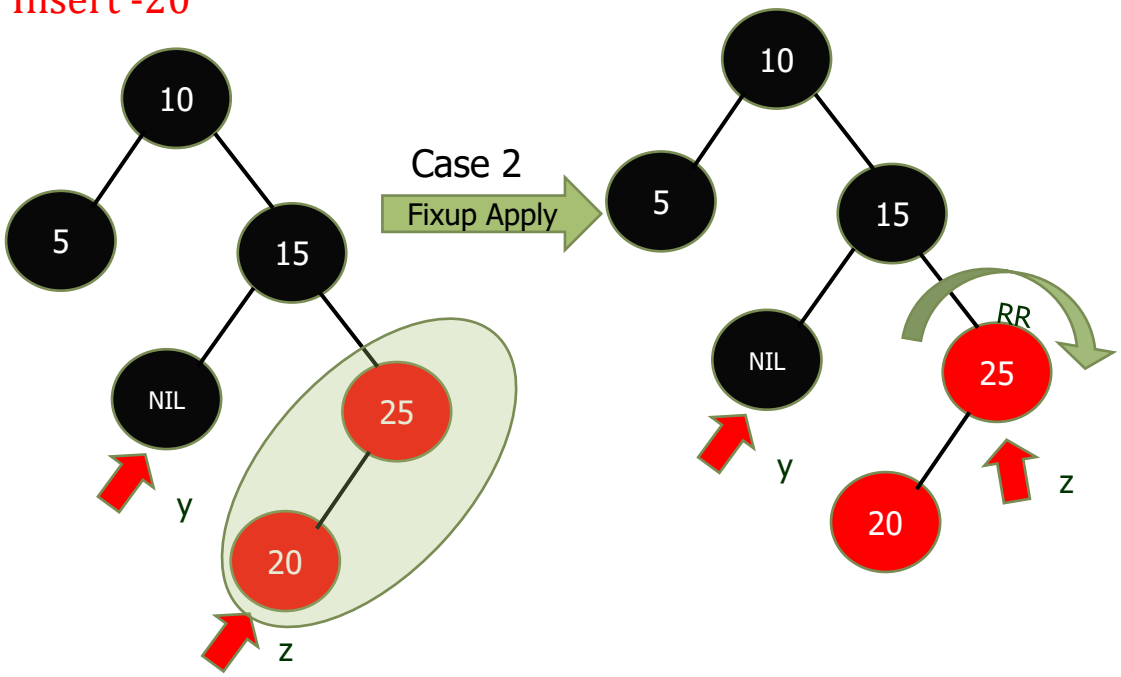
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -20



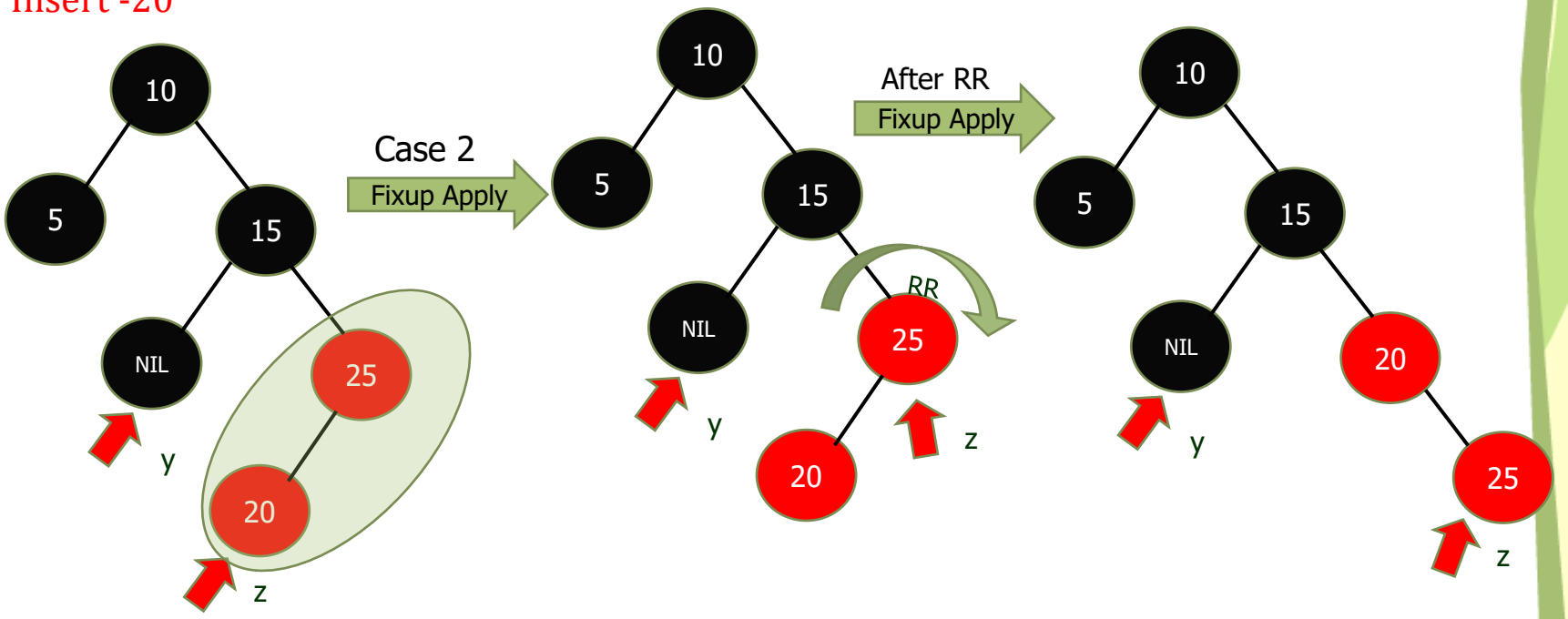
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -20



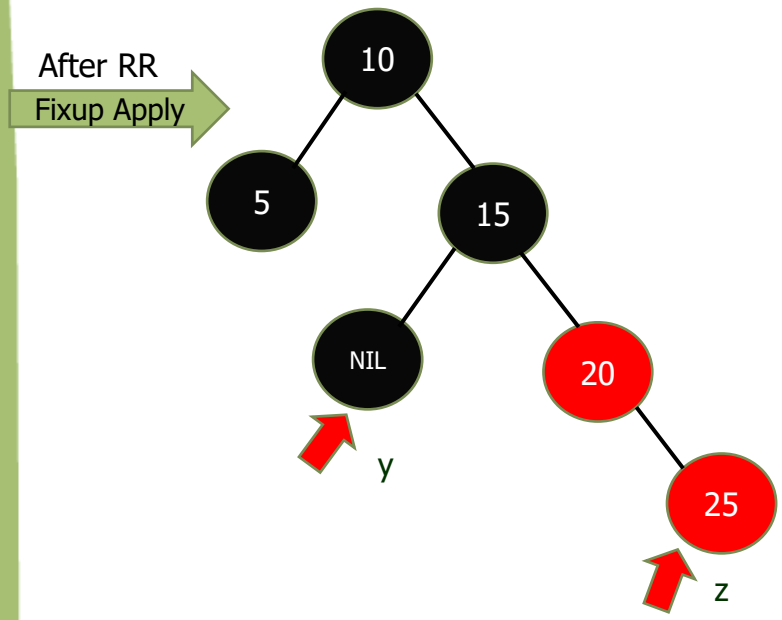
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -20



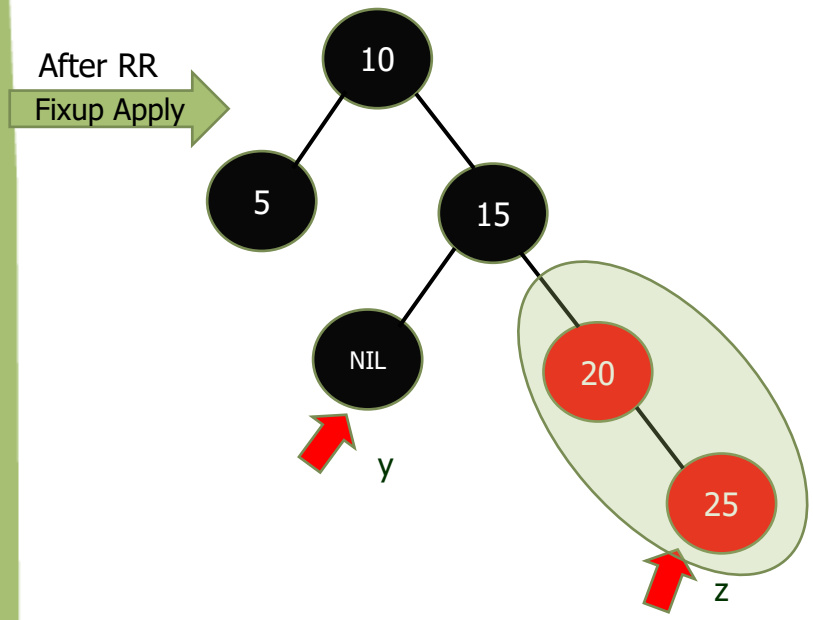
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -20



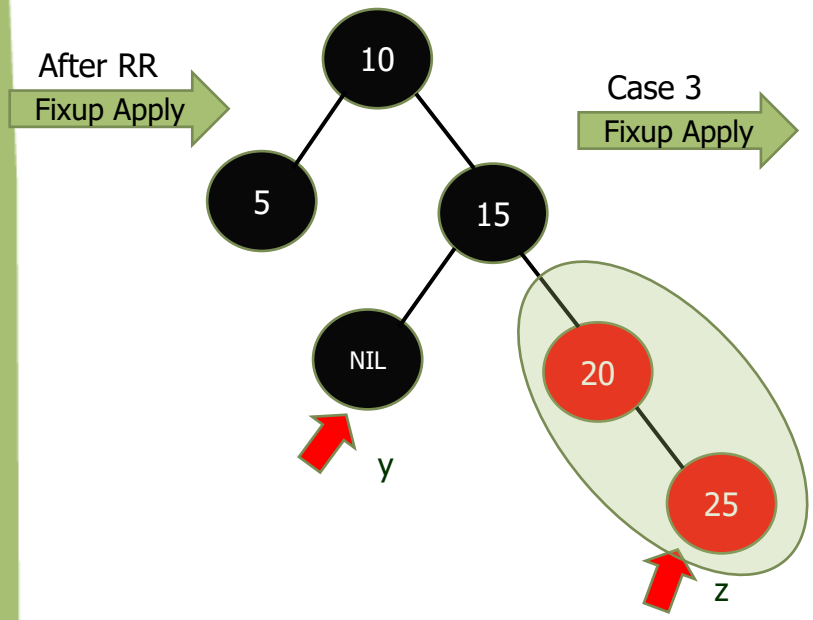
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -20



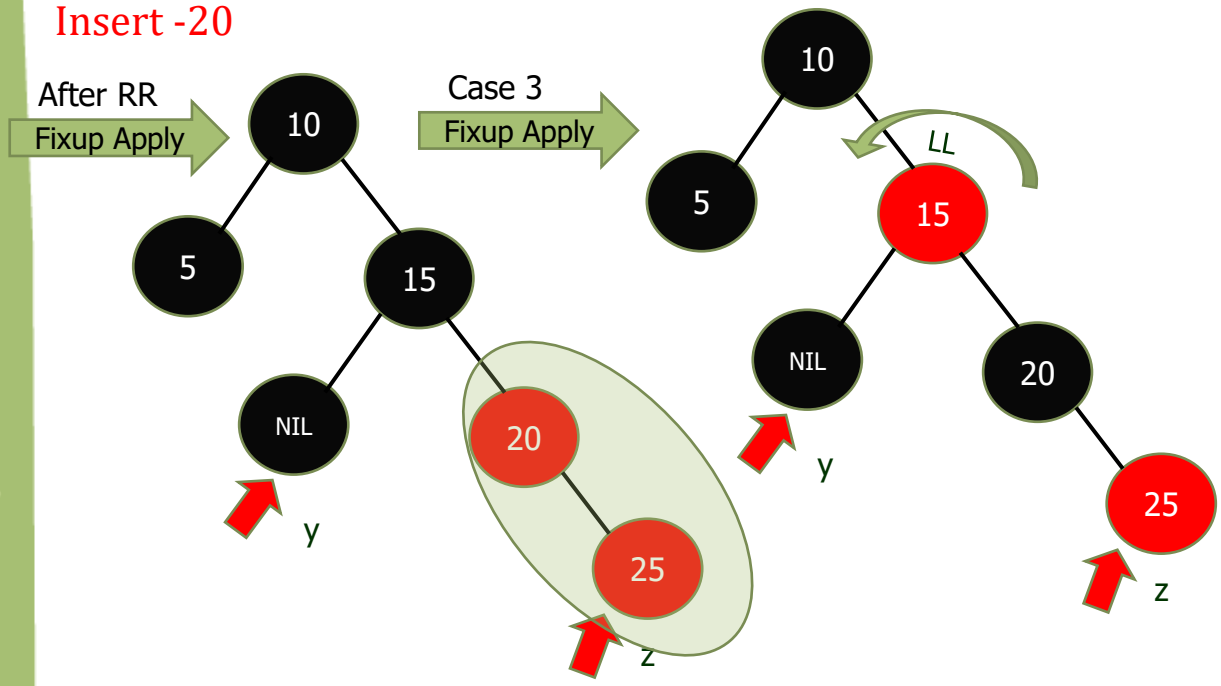
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -20



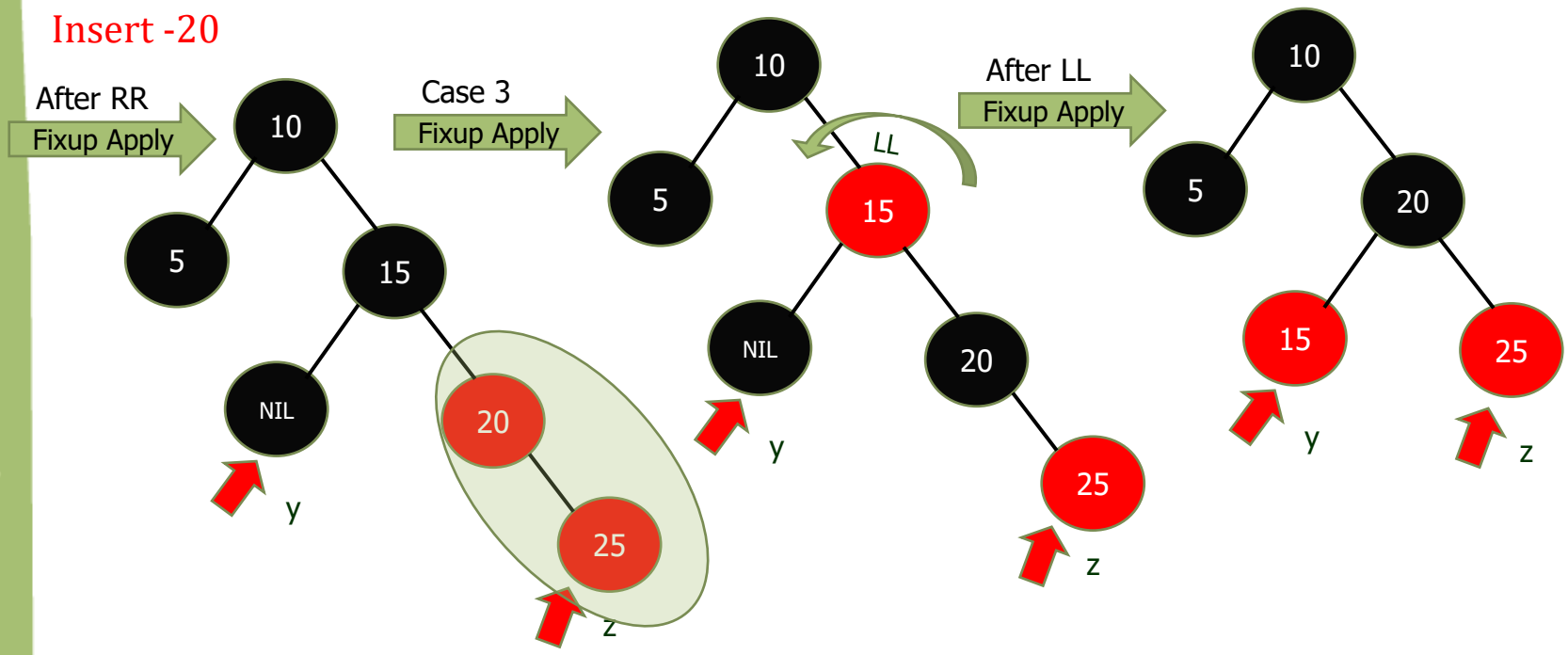
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -20



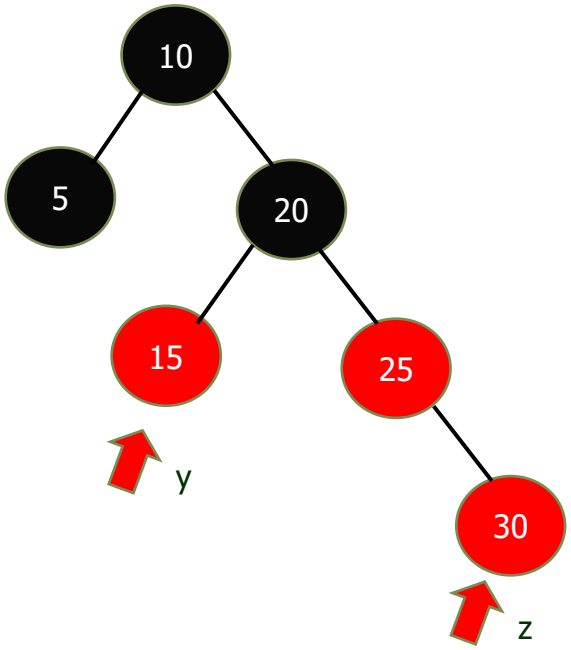
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -30



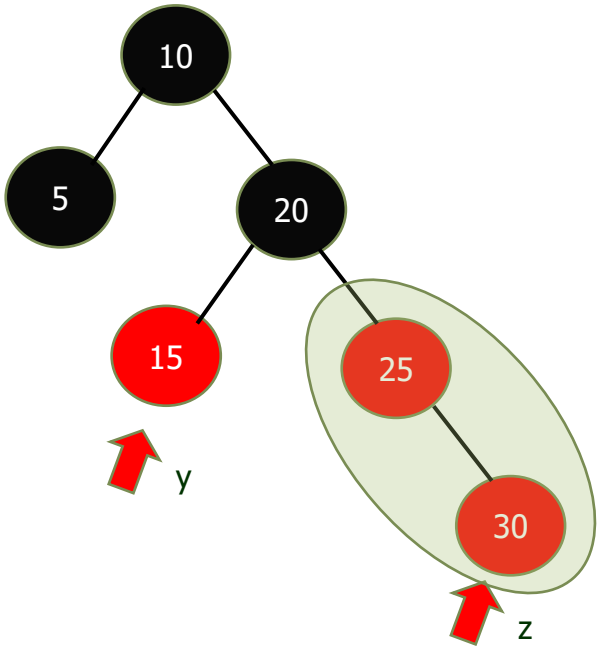
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -30



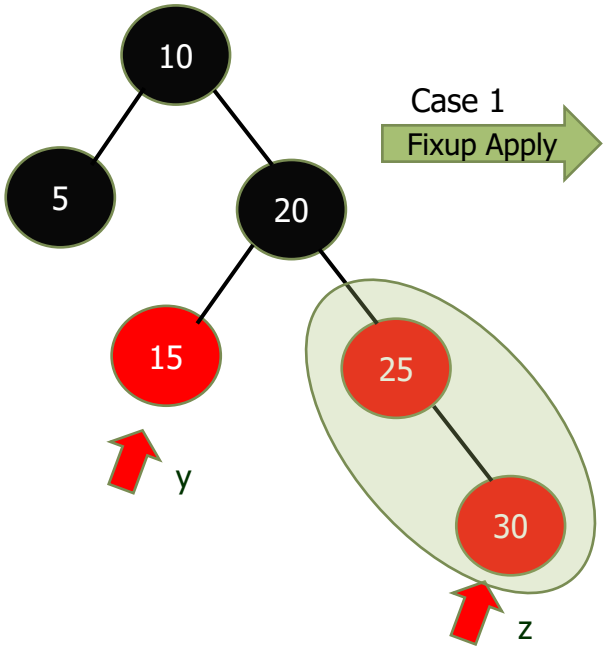
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -30



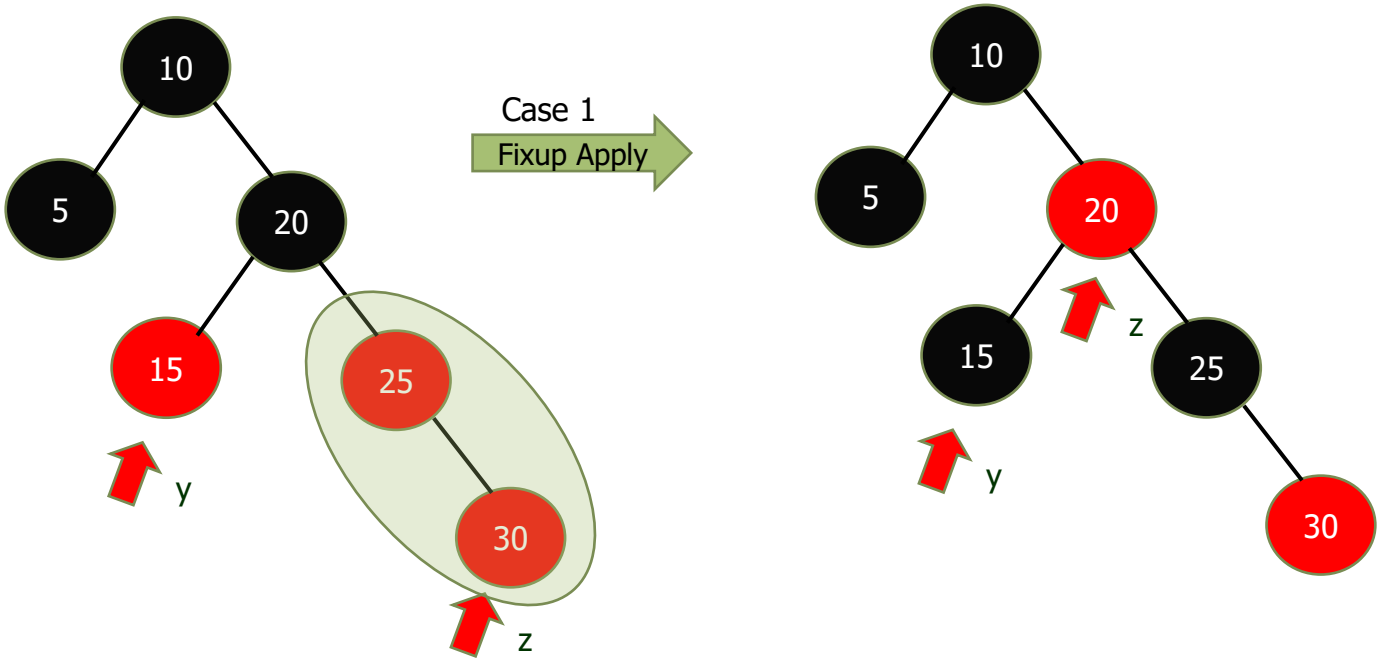
# Red Black Tree

**Insertion (Example 4):**

Insert the following elements into an empty RB-Tree.

[ 5, 10 ,15 , 25, 20 ,30]

Insert -30




Thank U