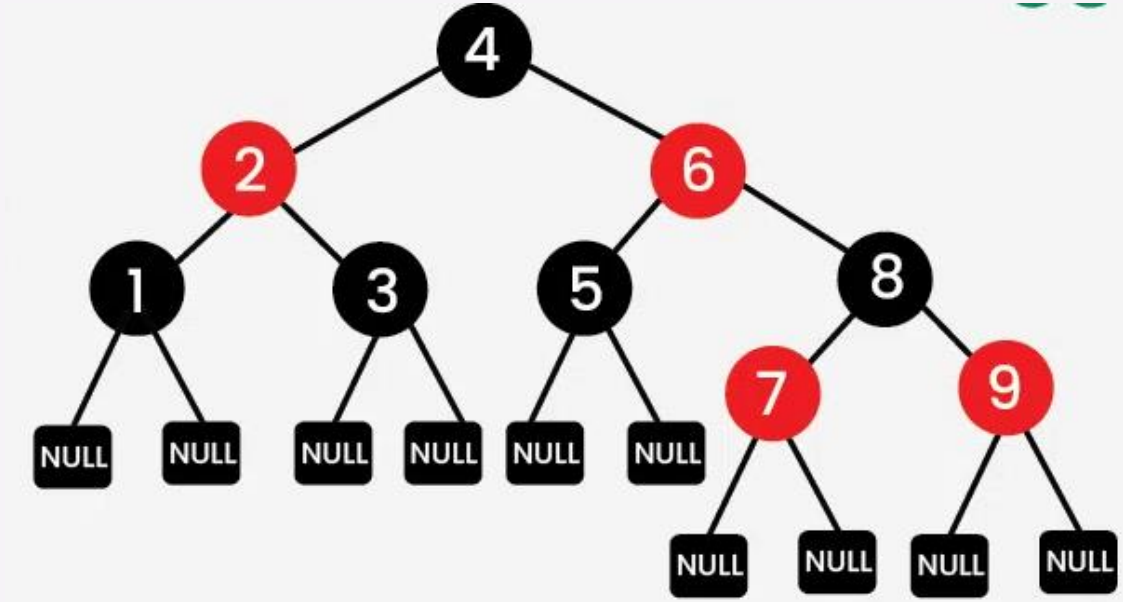
SCS 1308 – Foundation of Algorithms

Tutorial 13

What is Red Black Tree

"**Red Black Trees** are a type of **balanced binary search tree** that use a set of rules to maintain balance, ensuring **logarithmic time complexity** for operations like **insertion**, **deletion**, and **searching**, regardless of the initial shape of the tree. Red Black Trees are **self-balancing**, using a simple **color-coding scheme** to adjust the tree after each modification."

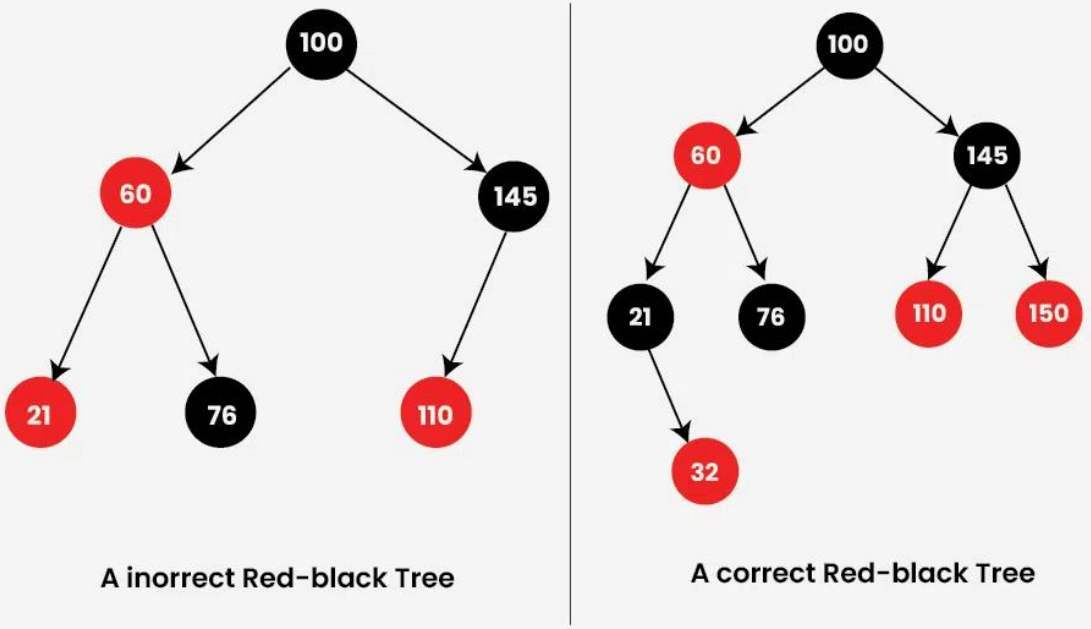


Properties of Red Black Tree

A red-black tree satisfies the following properties:

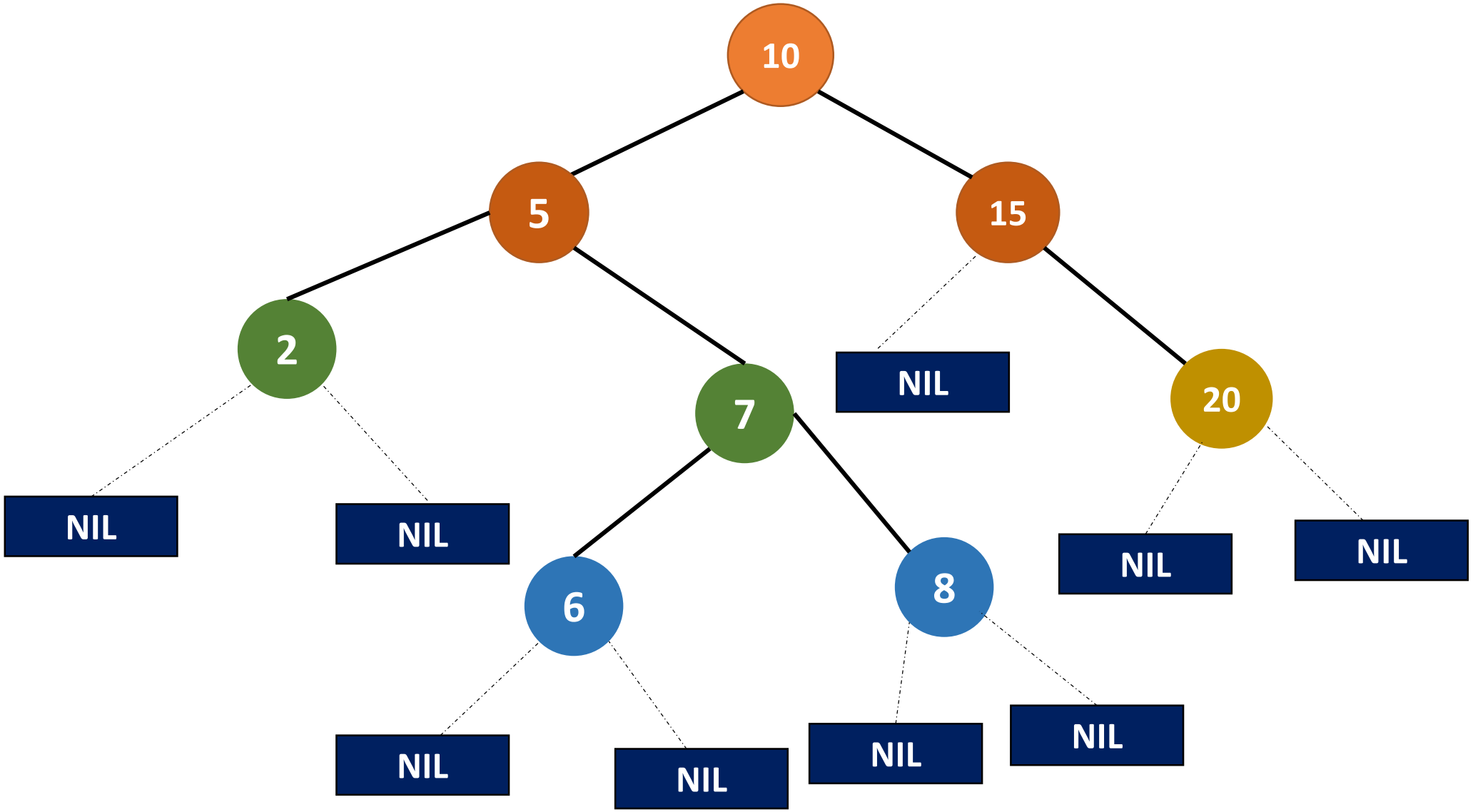
- **Red/Black Property:** Every node is coloured, either **red** or **black**.
- **Root Property:** The root is **black**.
- **Leaf Property:** Every **leaf** (NIL) is **black**.
- **Red Property:** If a **red** node has children then, the children are always **black**.
- **Depth Property:** For each node, any simple path from this **node** to any of its **descendant leaf** has the same **black-depth** (the number of black nodes).

Example of Red-Black Tree:

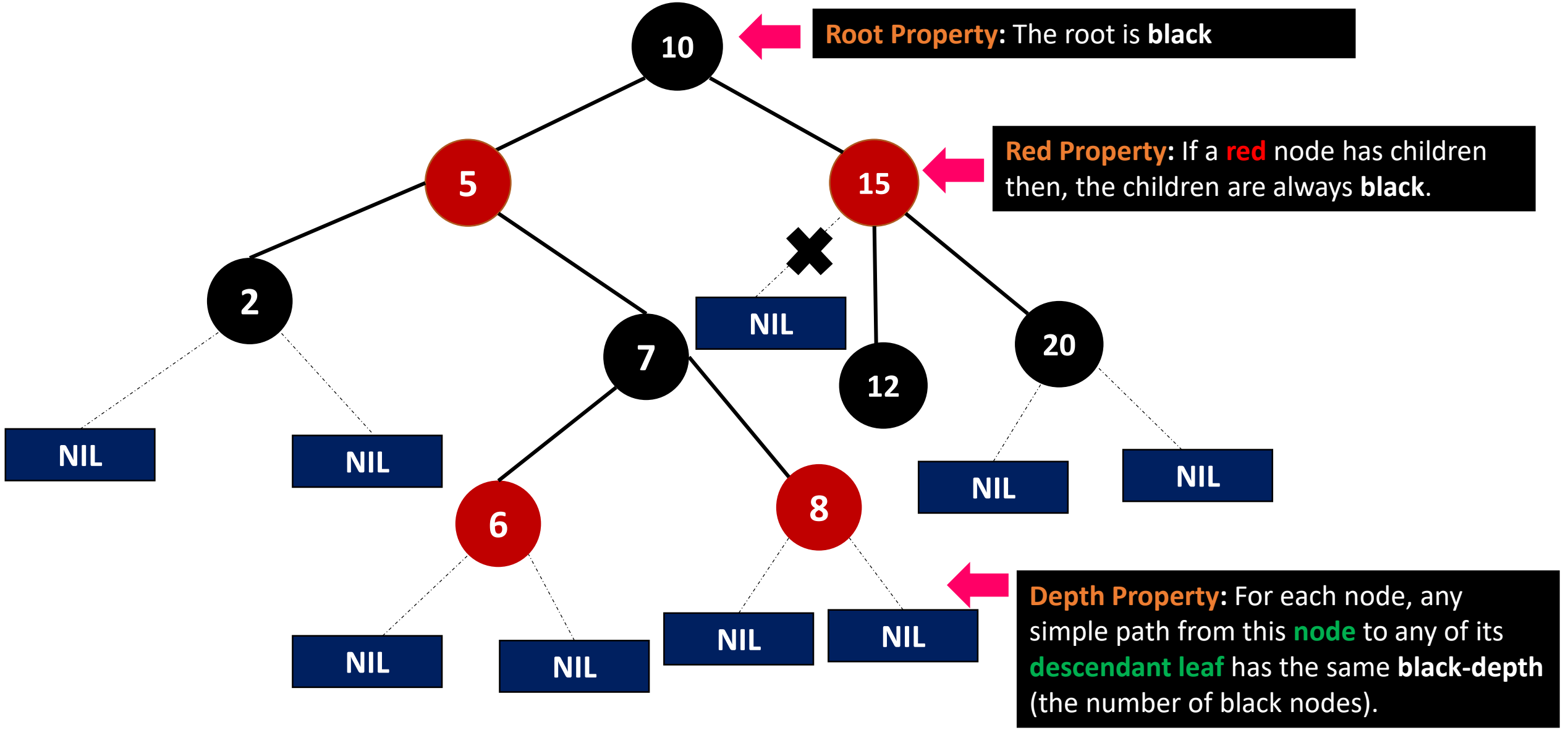


- The **Correct Red-Black Tree** in above image ensures that every path from the root to a leaf node has the **same number of black nodes**. In this case, there is one (excluding the root node).
- The **Incorrect Red Black Tree** does not follow the red-black properties as **two red nodes** are adjacent to each other. Another problem is that one of the paths to a **leaf node** has **zero black nodes**, whereas the other two contain a black node.

How to color a Red Black Tree



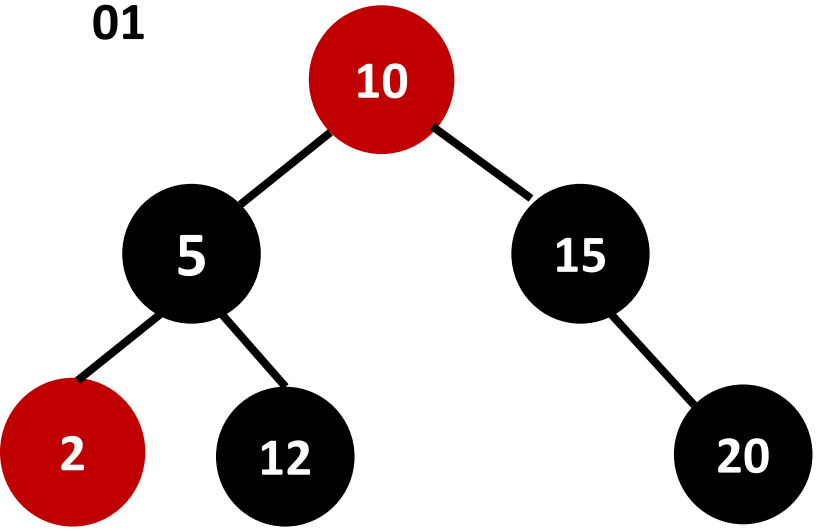
How to color a Red Black Tree



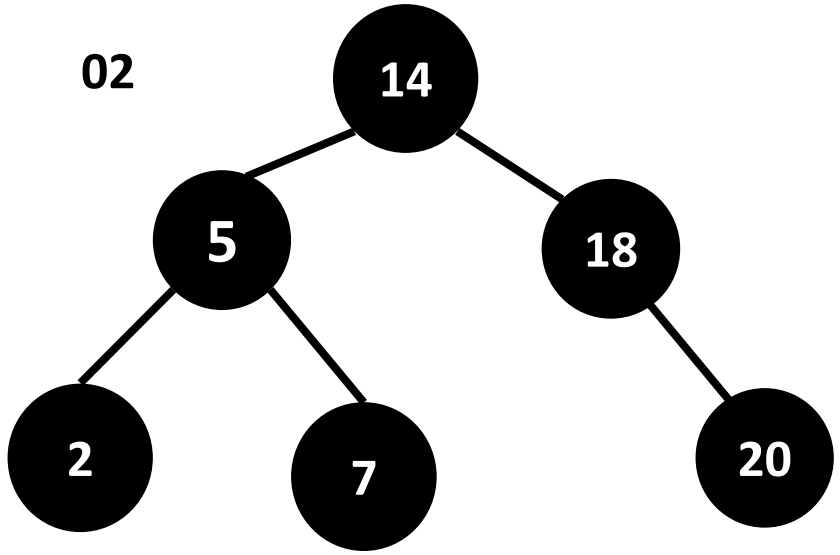
Exercise 01:

1.Are these trees are Red Black trees or not?

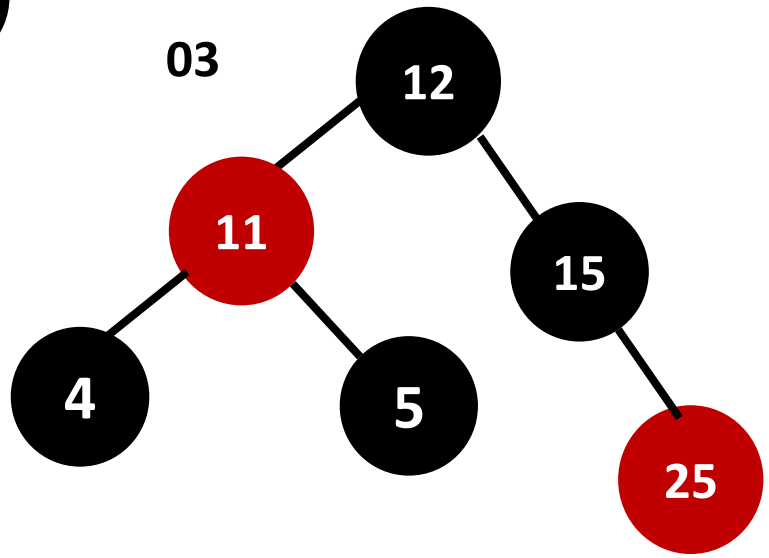
01



02



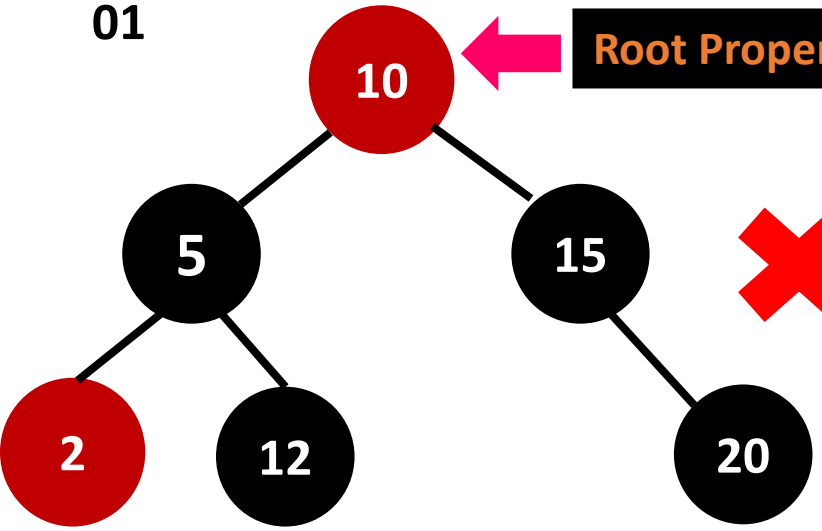
03



Exercise 01:

1.Are these trees are Red Black trees or not?

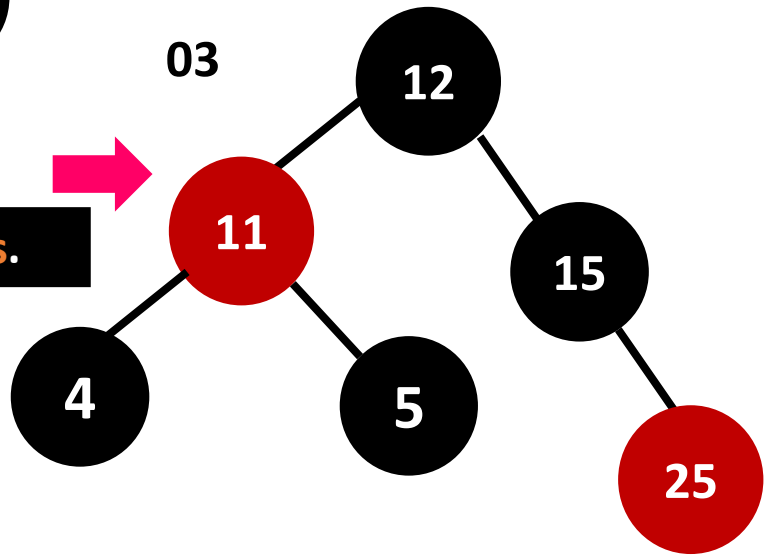
01



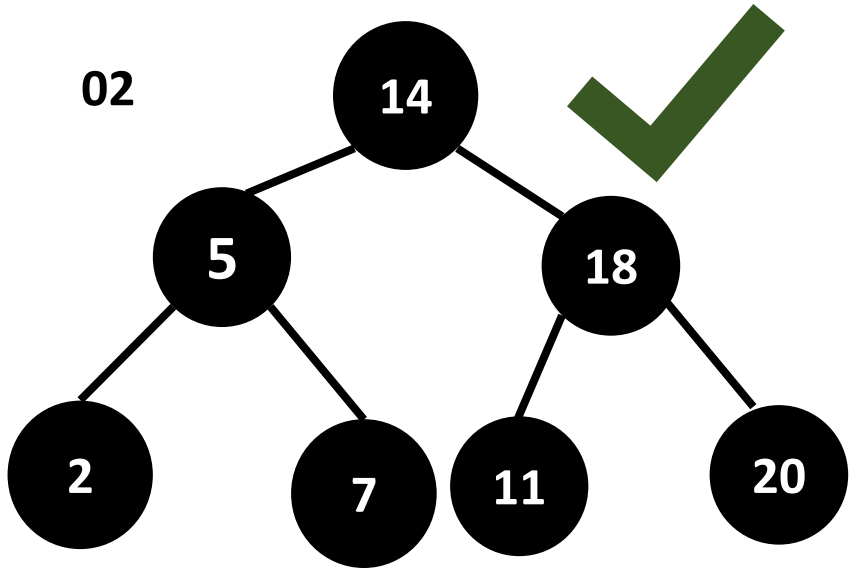
Root Property: The root is **black**

This is not following the BST Properties.

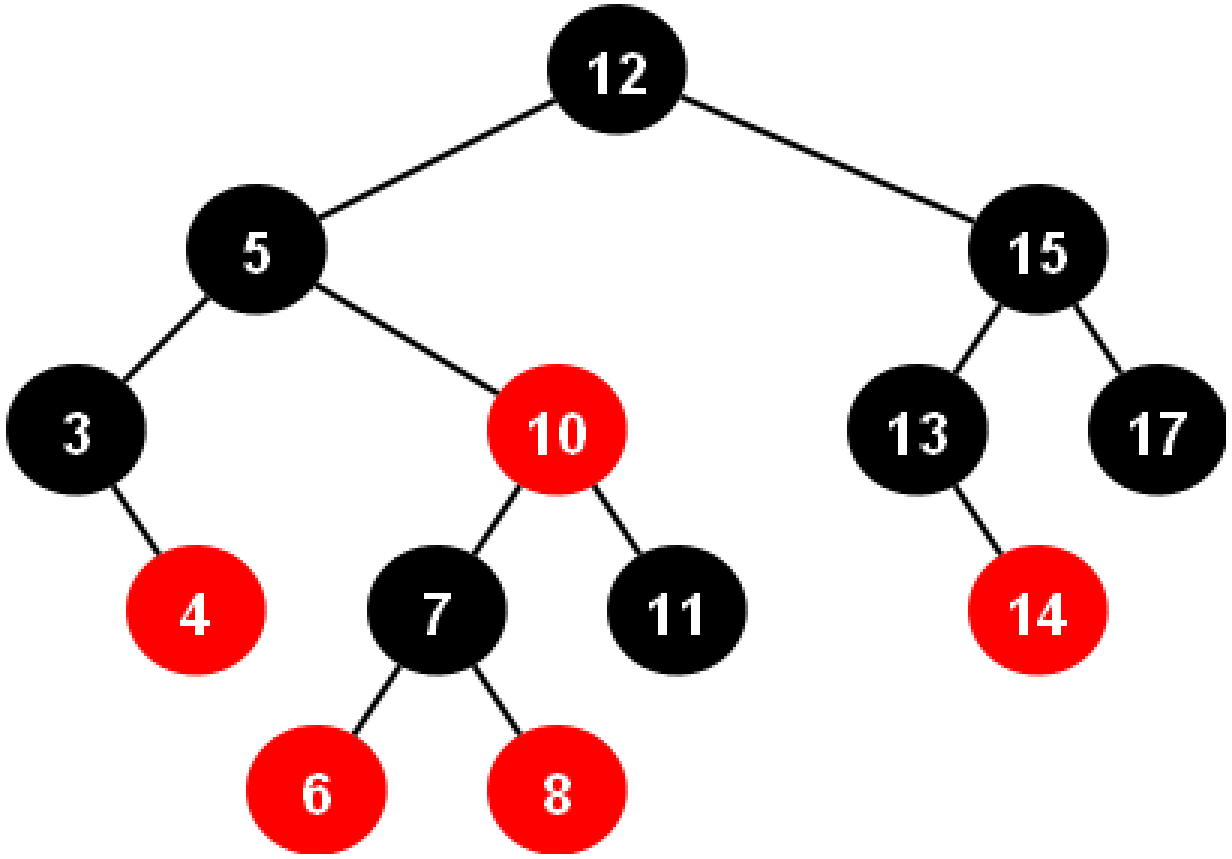
03



02



A perfect Binary tree which contains only black nodes is also a Red Black Tree.



Insertion in Red Black Trees

How to insert a node to Red Black Tree

1. If tree is **empty**, create new node as **root node** with color **black**.
2. If tree is **not empty**, create new node as **leaf node** with color **red**.
3. If parent of new node is **black** then exit.
4. If parent of new node is **red**, then check the color of **parent's sibling(node)**
    a. if color is **black** or **null,** then do **suitable rotations** and **recolor**.
    b. if color is **red** then recolor and also check if **parent's parent** of **new node** is not **root node**, then recolor it and recheck.

** Root → **Black** **

** No Two Adjacent **Red Nodes**

** Count Number of **Black nodes** in each paths**.**

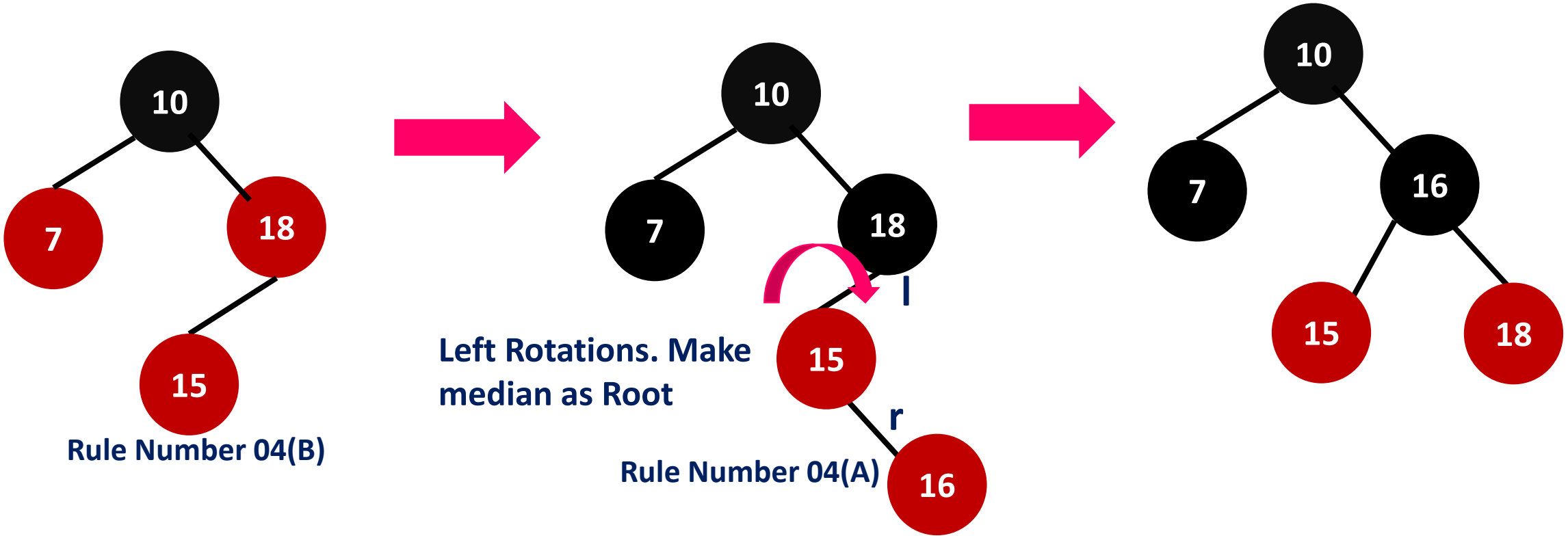
Fixing Violations During Insertion

After inserting the new node as a **red** node, we might encounter several cases depending on the colors of the **node's parent** and **uncle** (the sibling of the parent):

- **Case 1: Uncle is Red**: Recolor the parent and uncle to **black**, and the grandparent to **red**. Then move up the tree to check for further violations.
- **Case 2: Uncle is Black:**
  - **Sub-case 2.1: Node is a right child:** Perform a **left rotation** on the **parent**.
  - **Sub-case 2.2: Node is a left child:** Perform a **right rotation** on the **grandparent** and recolor appropriately.

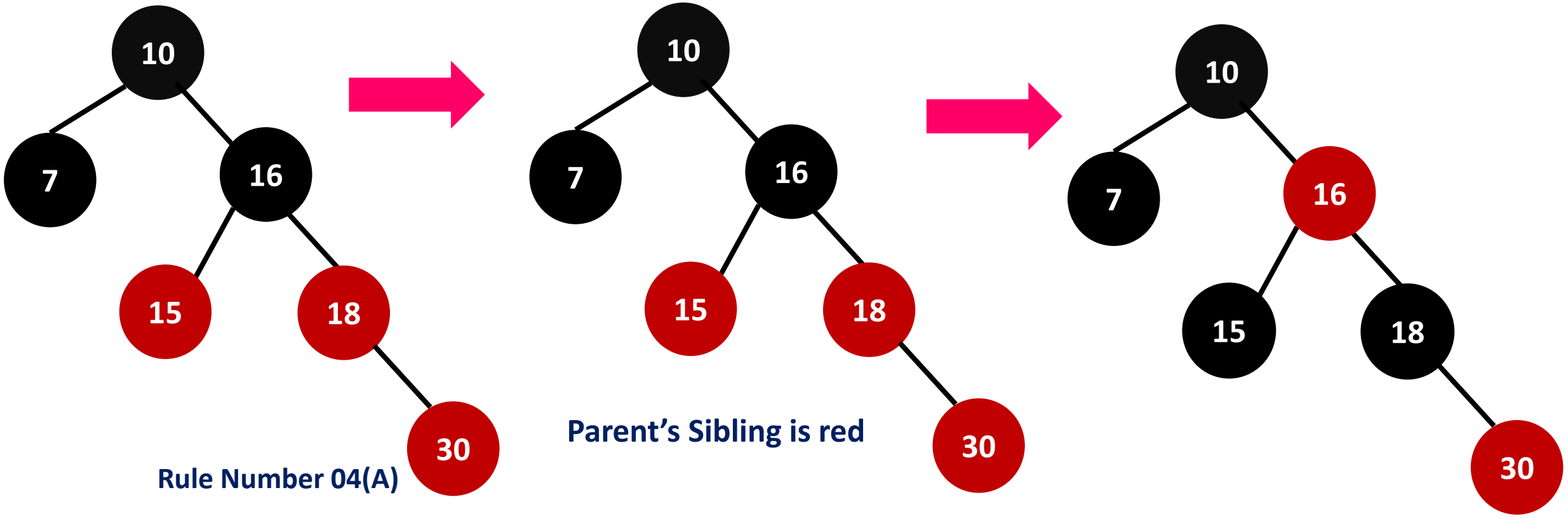
Example 02:

10	18	7	15	16	30	25	40	60	2	01	70



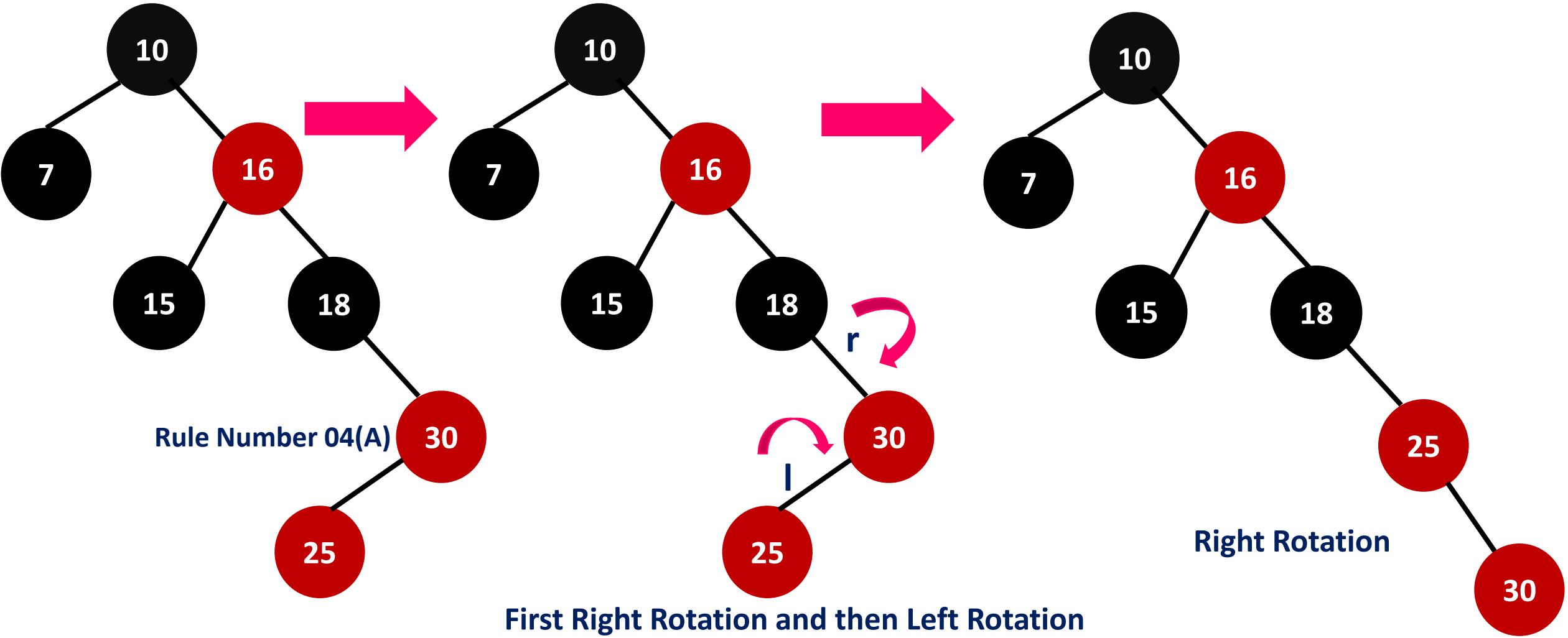
Example 02:

10	18	7	15	16	30	25	40	60	2	01	70



Example 02:

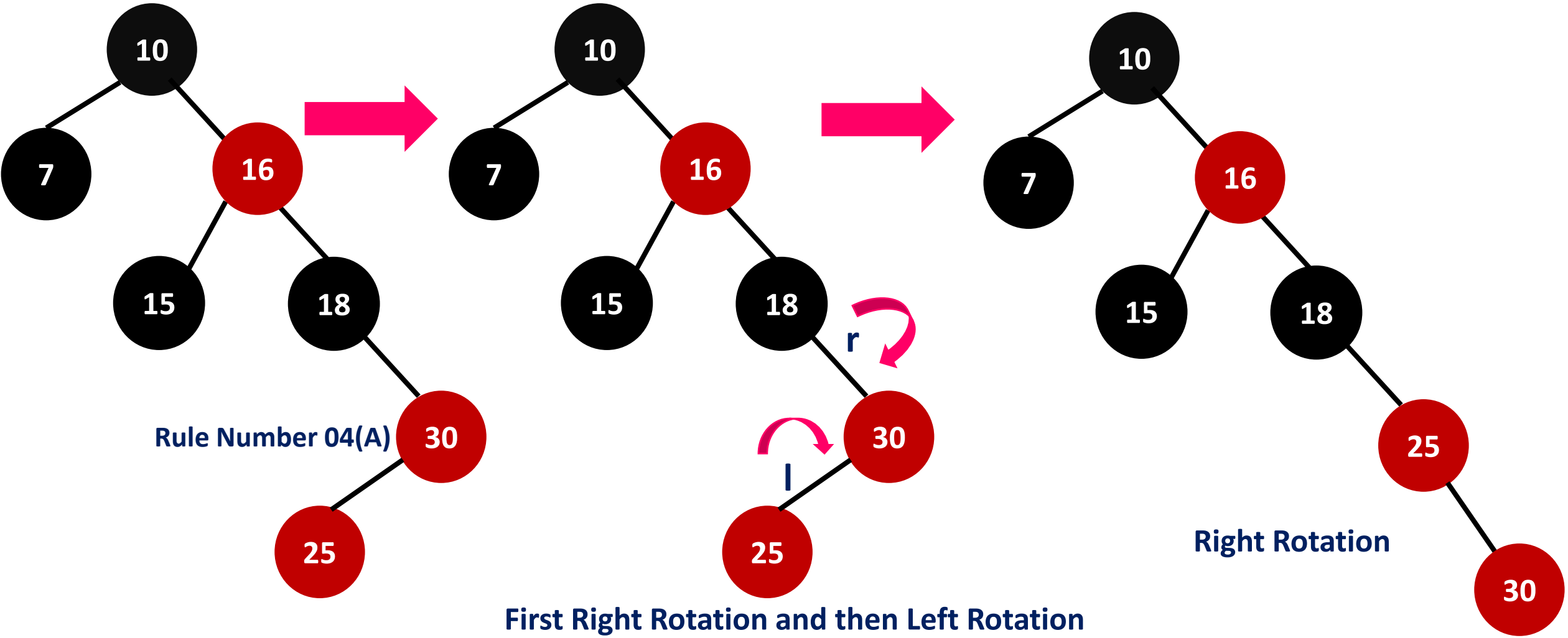
10	18	7	15	16	30	25	40	60	2	01	70



First Right Rotation and then Left Rotation needs to be done.

Example 02:

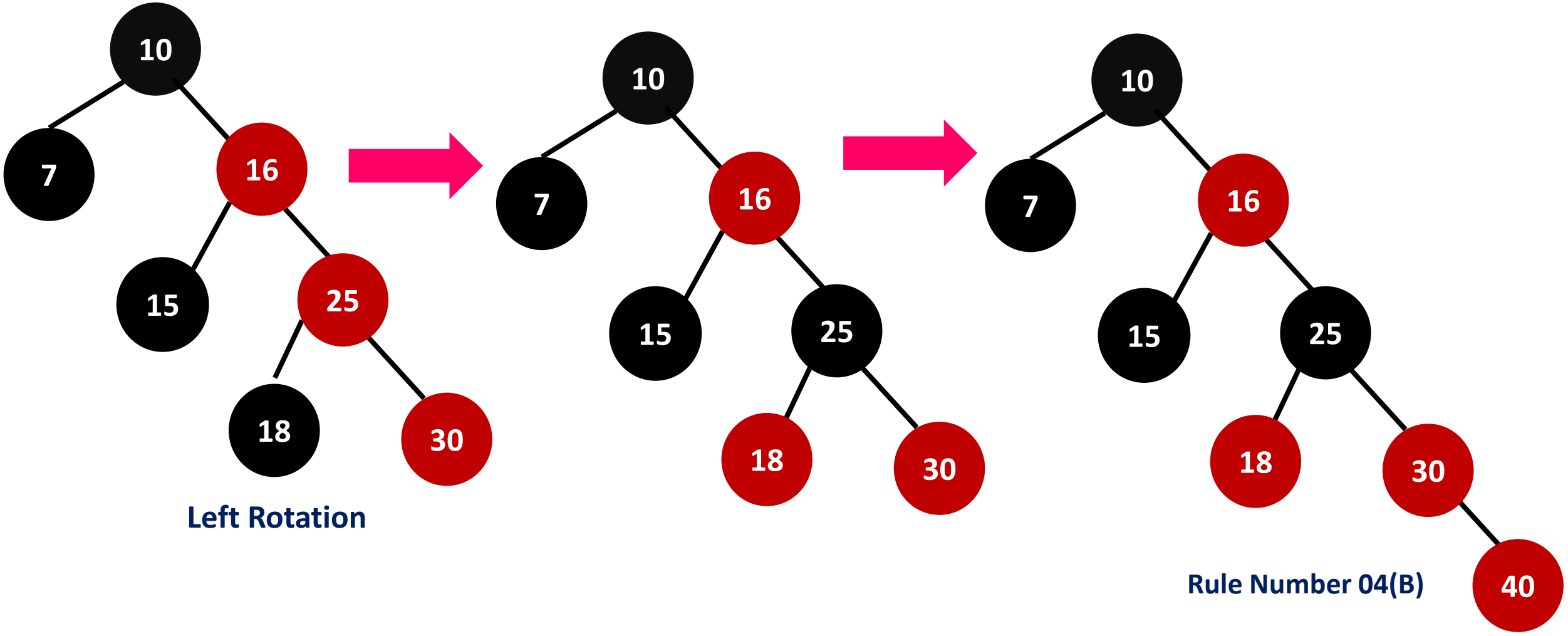
10	18	7	15	16	30	25	40	60	2	01	70



First Right Rotation and then Left Rotation needs to be done.

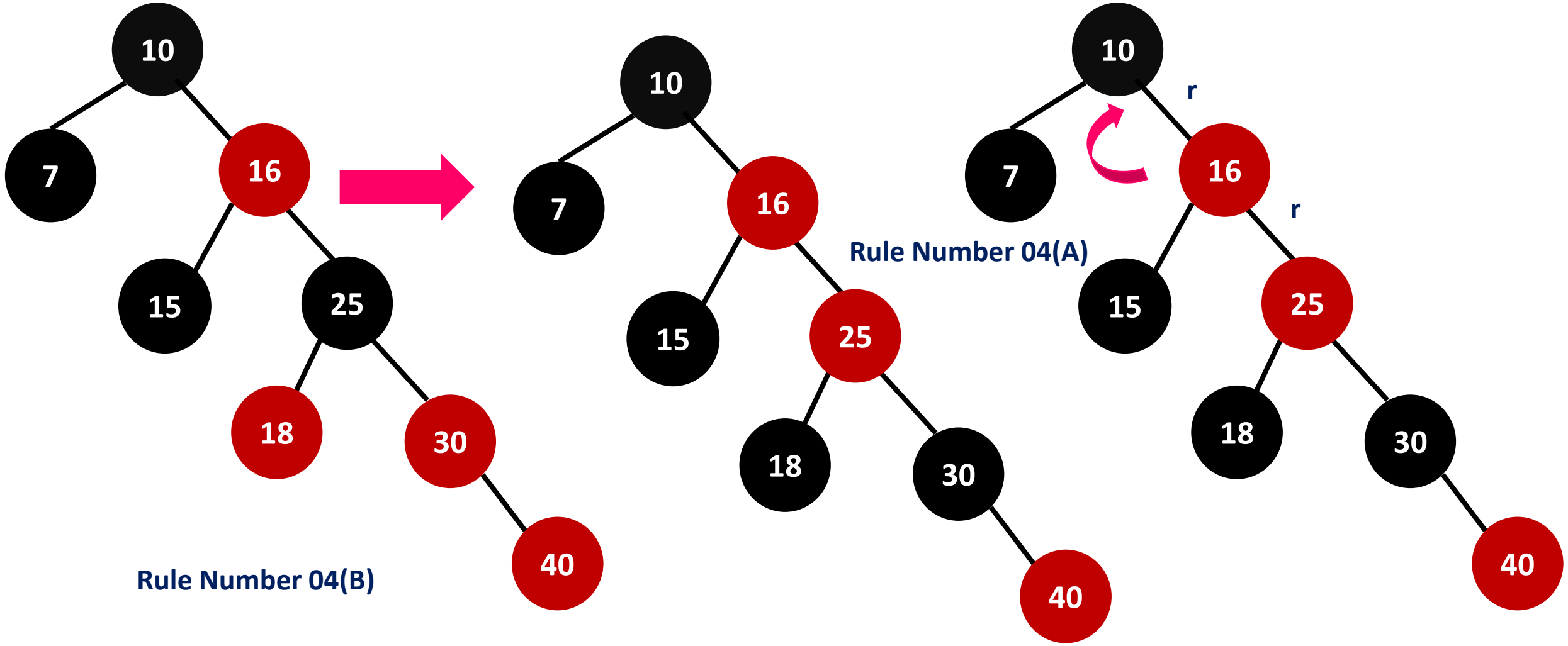
Example 02:

10	18	7	15	16	30	25	40	60	2	01	70



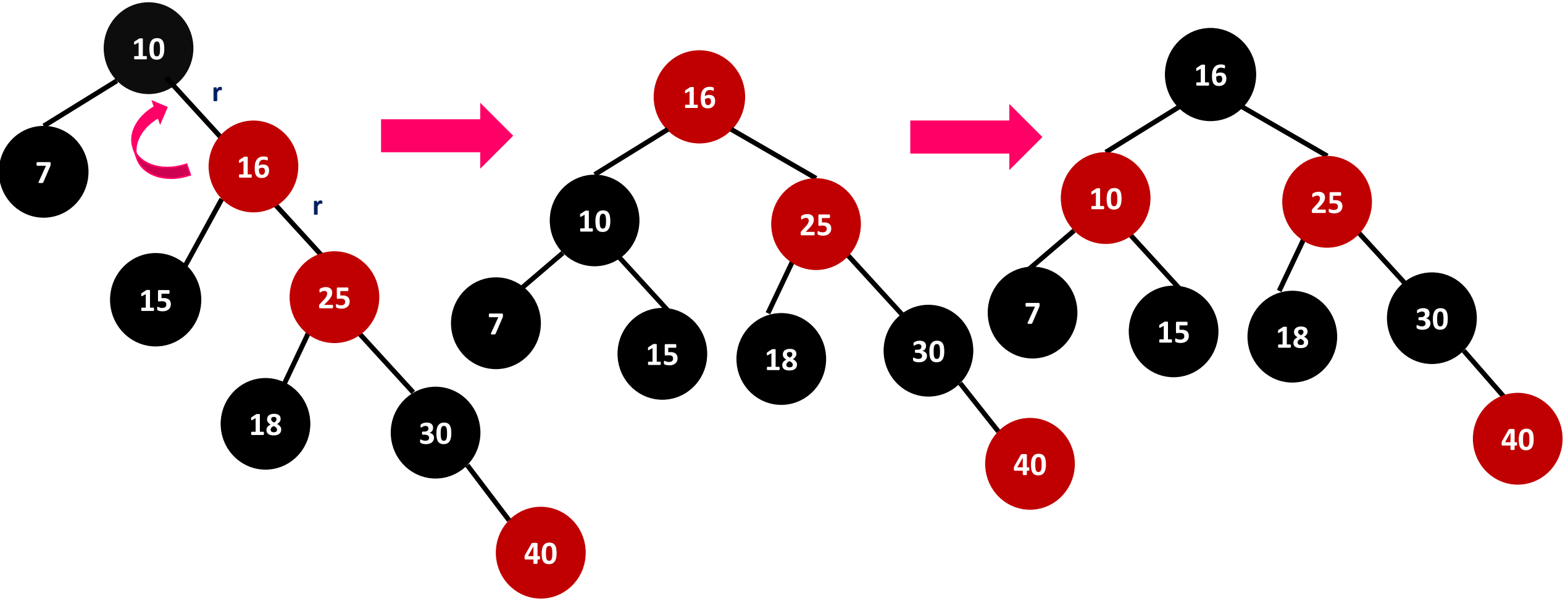
Example 02:

10	18	7	15	16	30	25	40	60	2	01	70



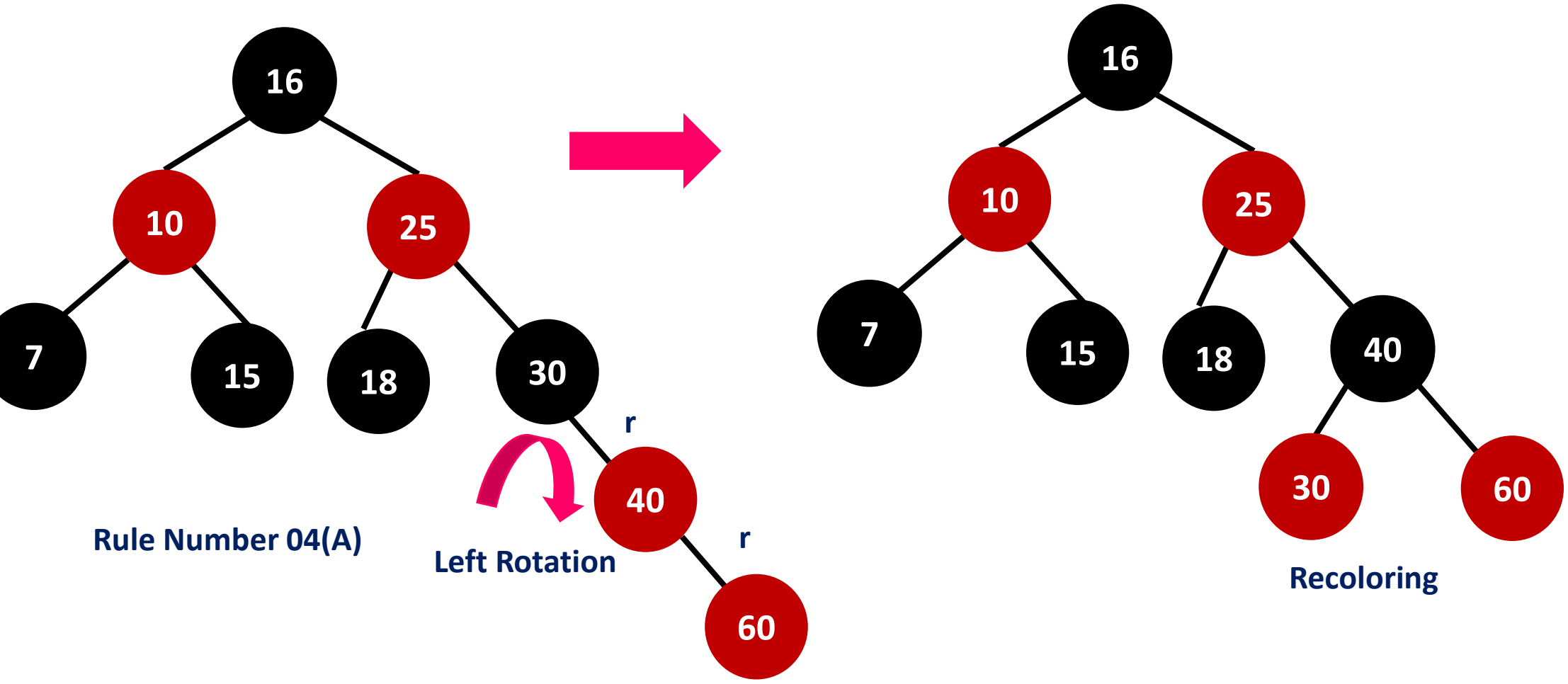
Example 02:

10	18	7	15	16	30	25	40	60	2	01	70



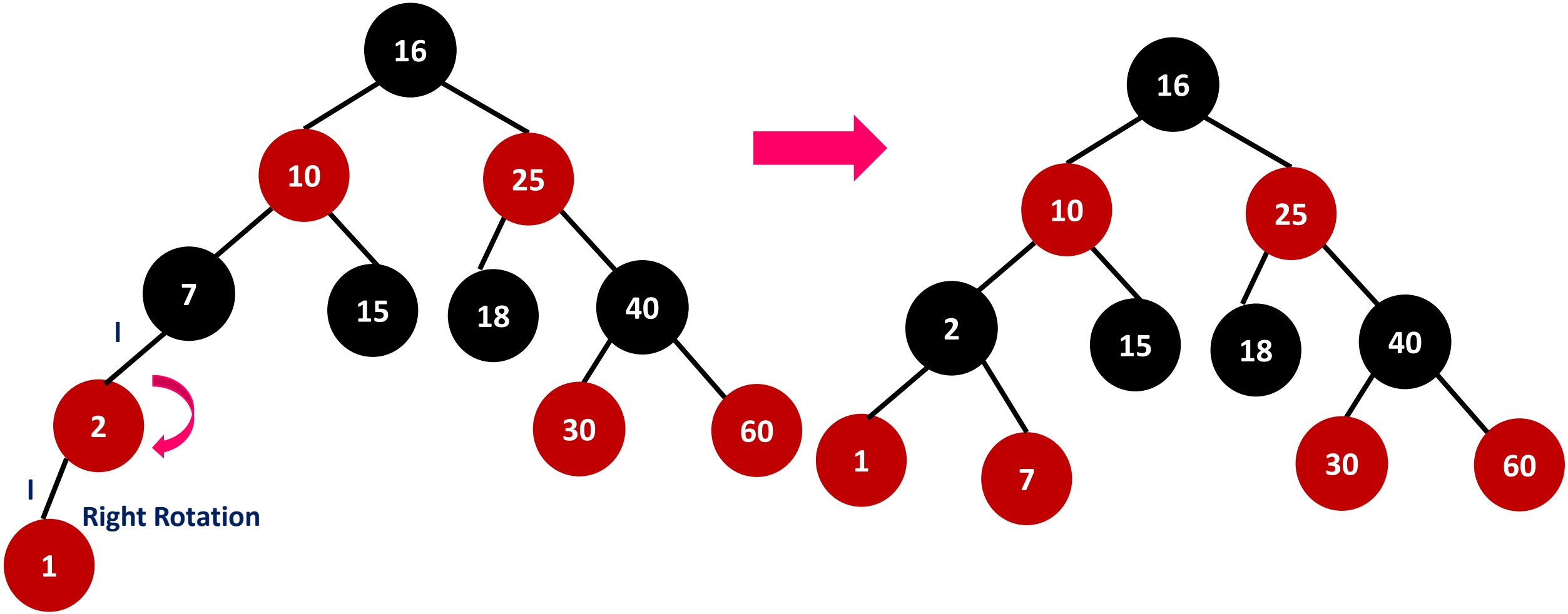
Example 02:

10	18	7	15	16	30	25	40	60	2	01	70



Example 02:

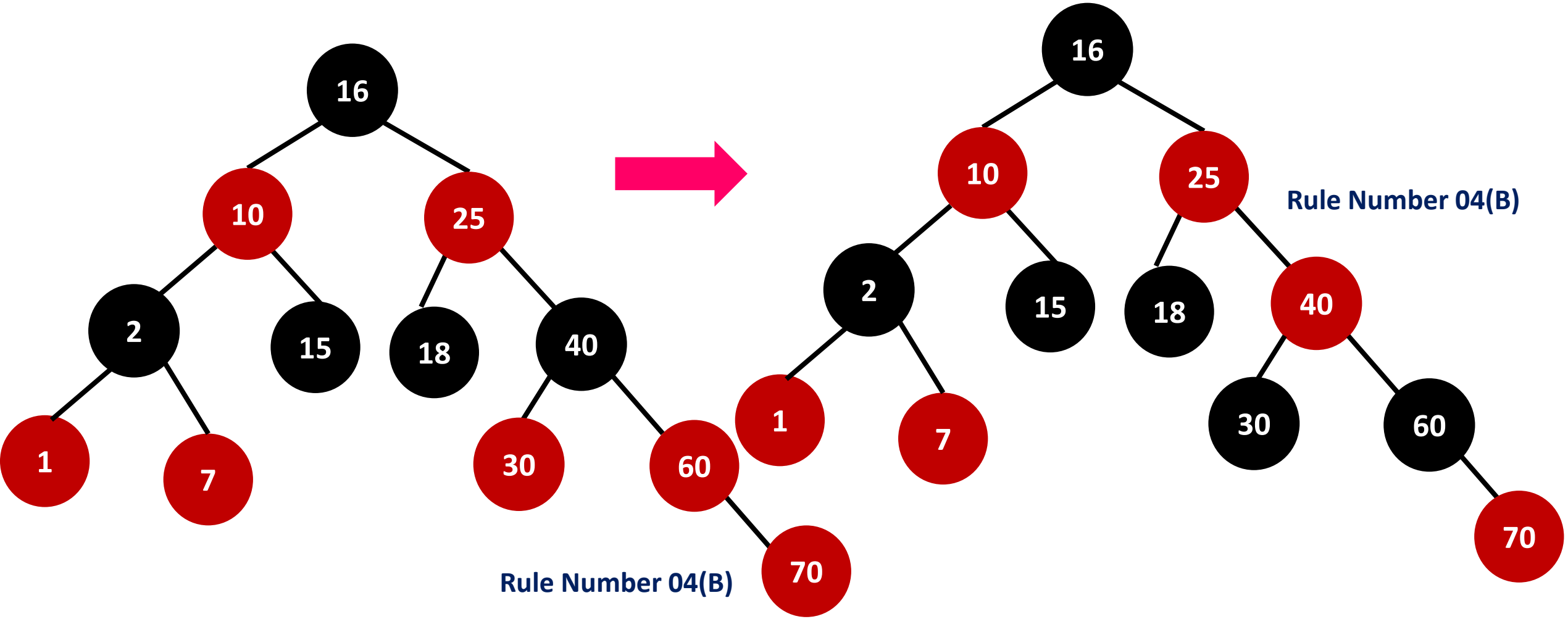
10	18	7	15	16	30	25	40	60	2	01	70



Rule Number 04(A)

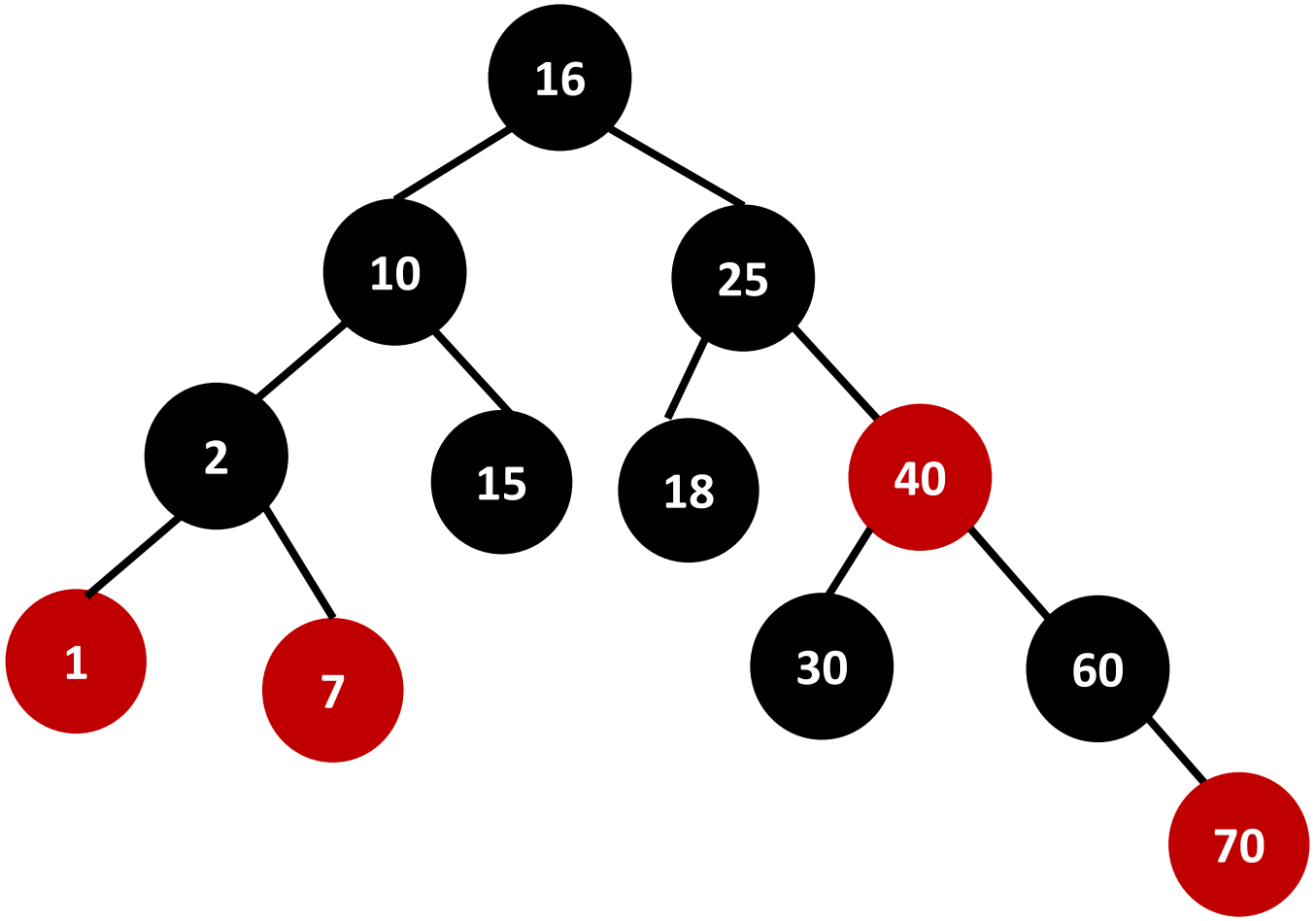
Example 02:

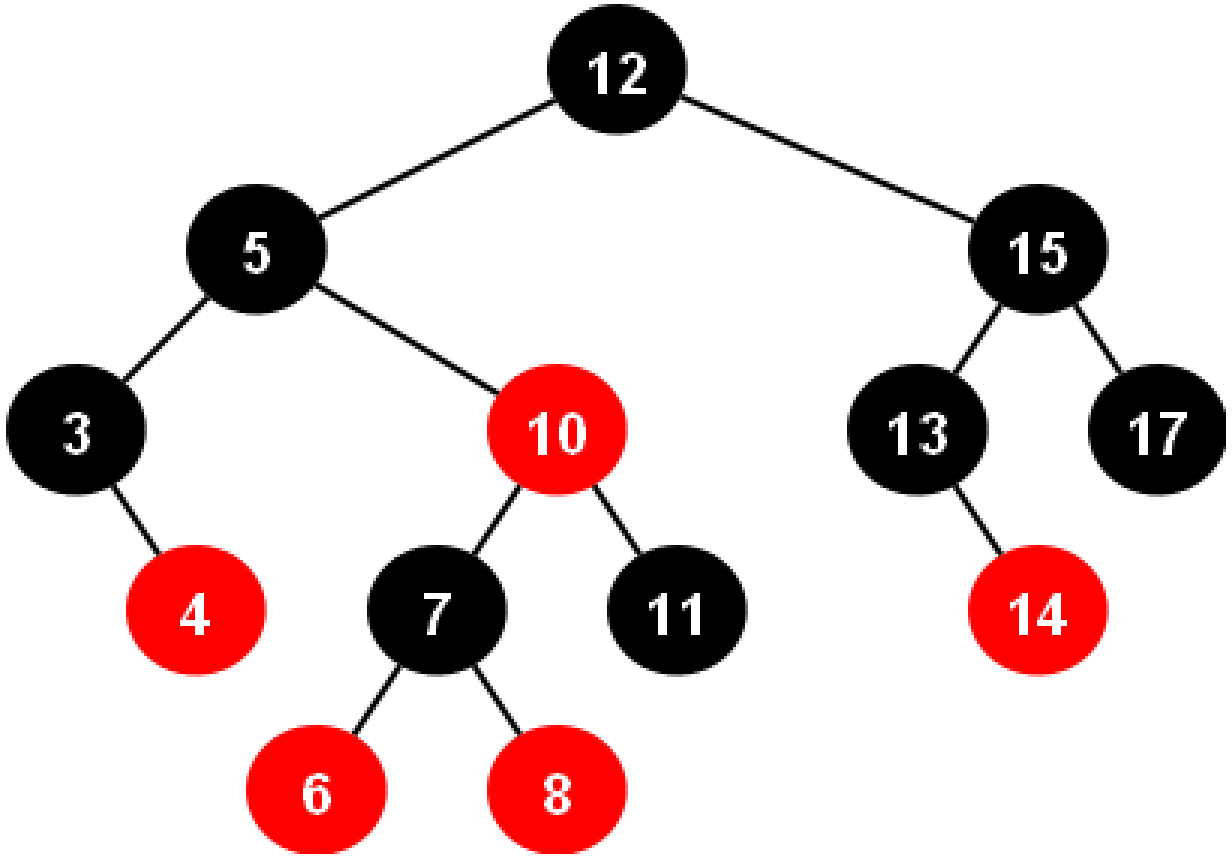
10	18	7	15	16	30	25	40	60	2	01	70



Example 02:

10	18	7	15	16	30	25	40	60	2	01	70





Deletion in Red Black Trees

How to delete a node in Red Black Tree

Deleting a node from a Red-Black Tree also involves a two-step process: performing the BST deletion, followed by fixing any violations that arise.

Deletion Steps

- **BST Deletion**: Remove the node using standard BST rules.
- **Fix Double Black:**
  - If a black node is deleted, a “double black” condition might arise, which requires specific fixes.

Fixing Violations During Deletion

When a black node is deleted, we handle the double black issue based on the **sibling's color** and the **colors of its children**:

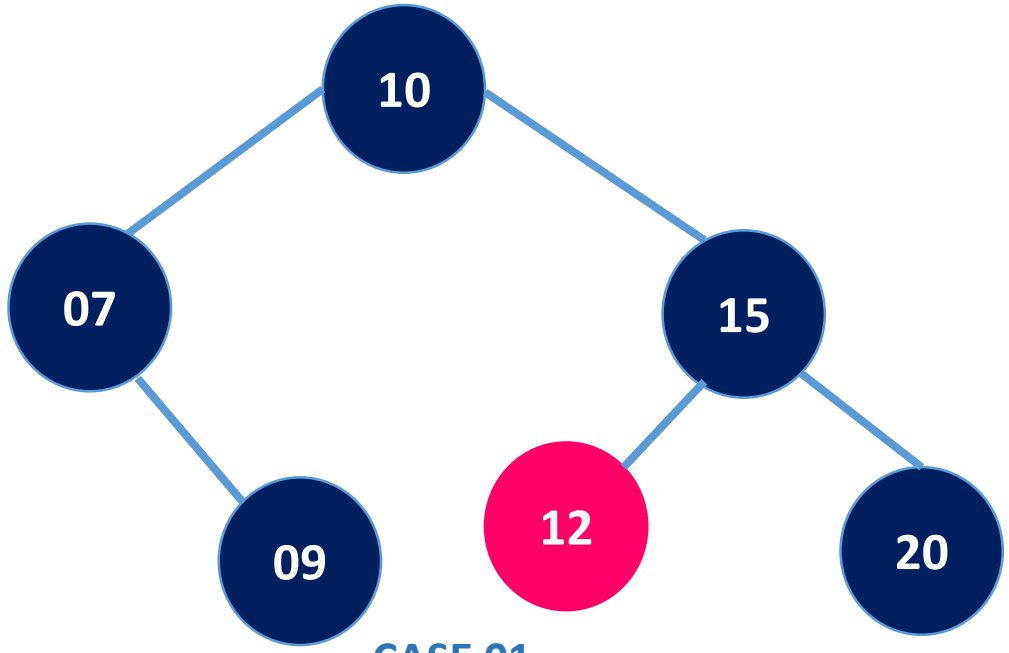
- **Case 1: Sibling is Red:** Rotate the parent and recolor the sibling and parent.
- **Case 2: Sibling is Black:**
  - **Sub-case 2.1: Sibling's children are black:** Recolor the sibling and propagate the double black upwards.
  - **Sub-case 2.2: At least one of the sibling's children is red:**
    - **If the sibling's far child is red**: Perform a rotation on the parent and sibling, and recolor appropriately.
    - **If the sibling's near child is red**: Rotate the sibling and its child, then handle as above.

Deletion in Red Black Tree

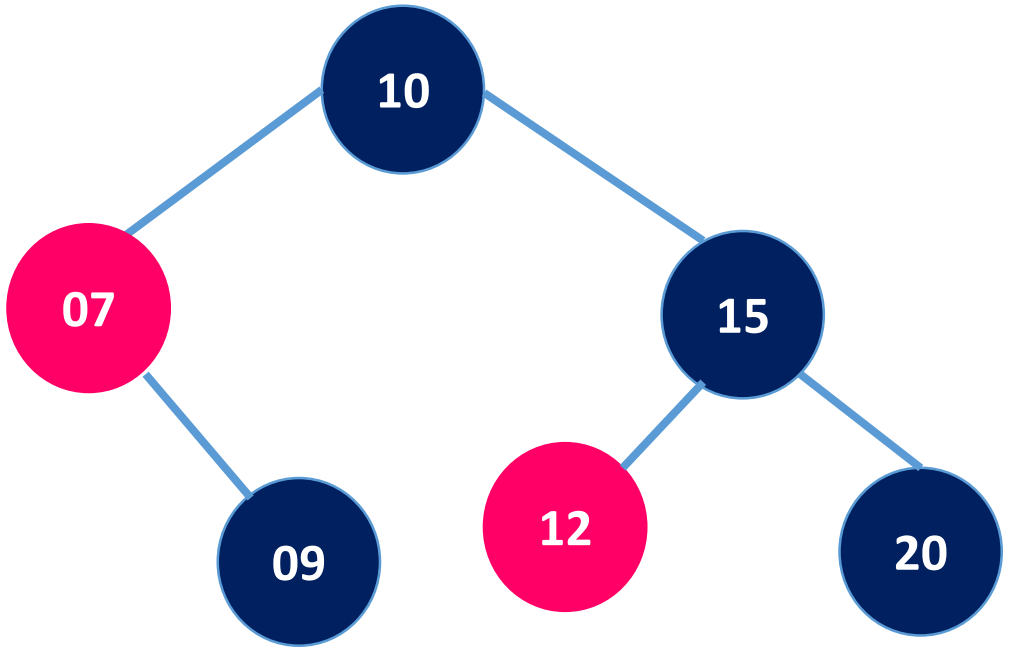
In the deletion you have to perform standard **Binary Search Tree Deletion**.

CASE 01: So if you are going to delete a node with **no children** then you can just delete that node. Ex: 12.

CASE 02: If the node that you want to delete is having only one child, You have to **replace** that node with **it's child** and delete. Because we don't delete **internal node** we only delete leaf nodes.



CASE 01



CASE 02

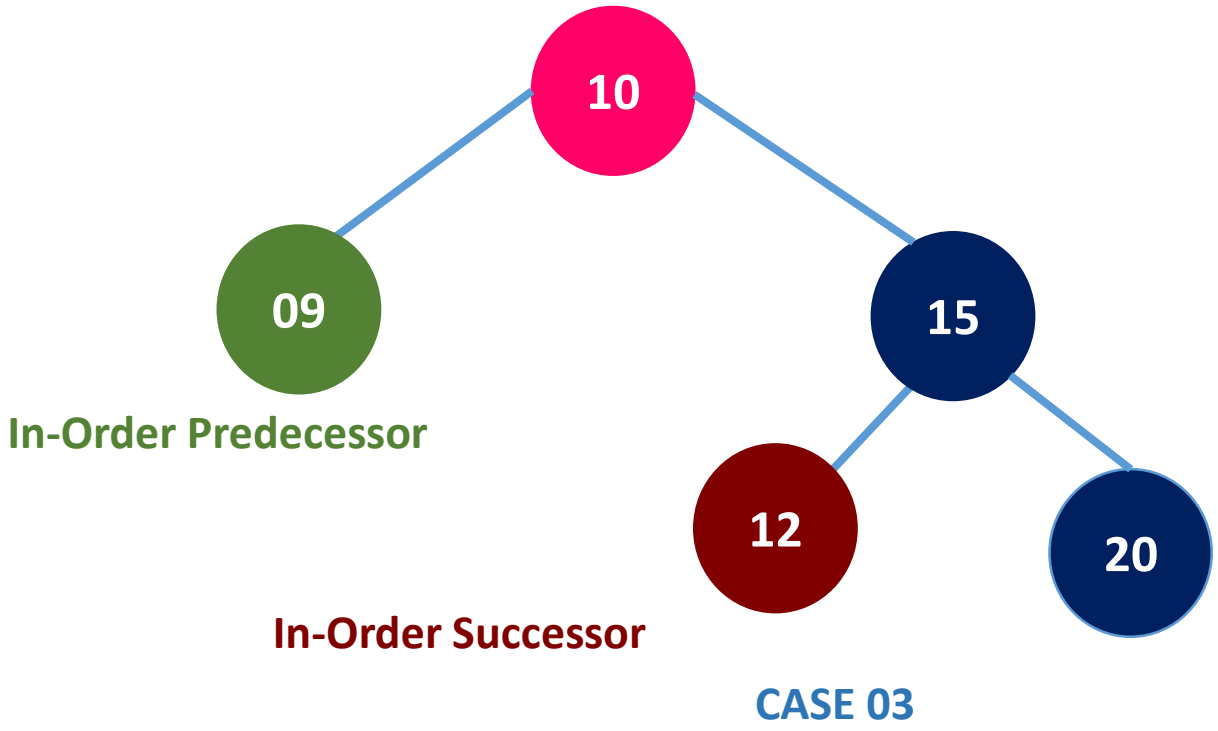
Deletion in Red Black Tree

In the deletion you have to perform standard **Binary Search Tree Deletion**.

CASE 03: So if you are going to delete a node with **two children** then you can follow two methods.

a. **In-Order Predecessor**: Replace the node with largest element of the **left Sub Tree**

b. **In-Order Successor:** Replace the node with smallest element of the **right Sub Tree**



Rules of deletion in Red Black Tree

Step 01: Perform BST Deletion.

Step 02:

Case 01: If node to be deleted is **RED** just delete it.

Case 02: If the node you want to delete is having one child you can follow BST deletion.