Red-Black trees

D.E ZEGOUR

Ecole Supérieure d'Informatique

ESI

Red-Black Trees

Introduction

Very popular data structure

Implemented and integrated into several programming languages (JAVA, C...)

Used to implement dictionaries and associative arrays

Red-Black Trees

Definition

A red-black tree is a binary search tree where each node is either red or black.

Moreover, all branches emanating from any node:
- Do not have two consecutive red nodes.
- Have the same number of black nodes.

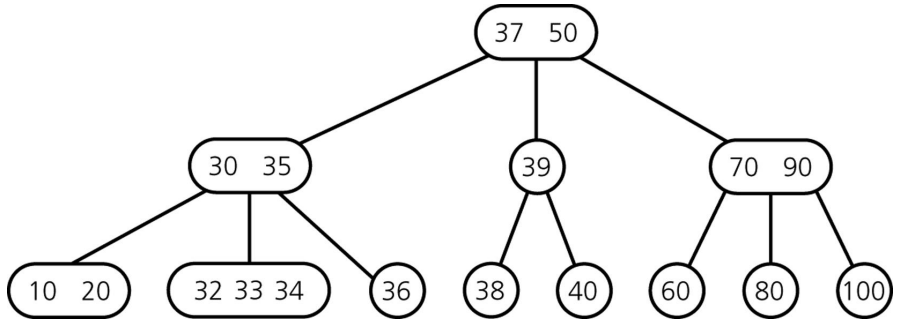
Nil considered as Black Noir

- Black nodes: perfect balance
- Red nodes: tolerate slight imbalance

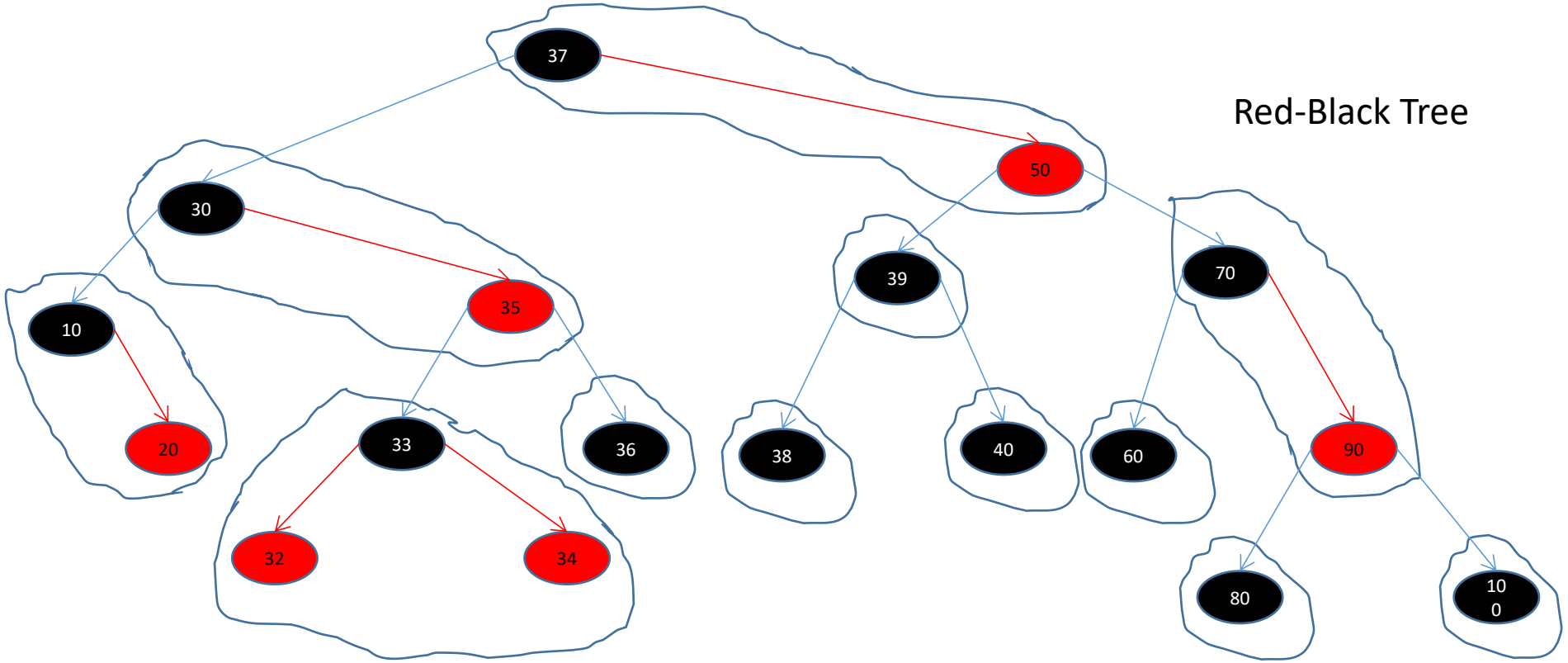
Worst-case scenario:
Alternation between red and black nodes.

Red-Black Trees

From 2-4 Trees to Red-Black trees



2-4 Tree



Red-Black Tree

Red-Black Trees

Insertion

Insertion as in a binary search tree

The inserted node is always a leaf

We attribute it the Red color

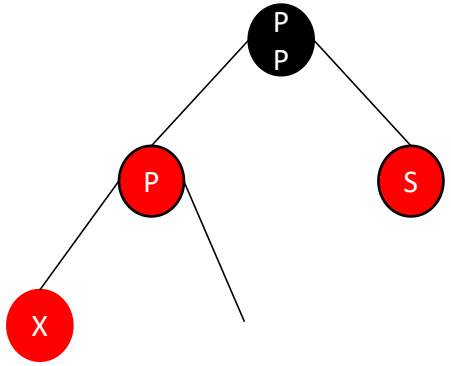
If its parent is also Red, a maintenance algorithm is applied

In the 2-4 tree : Splitting node PP

Red-Black Trees

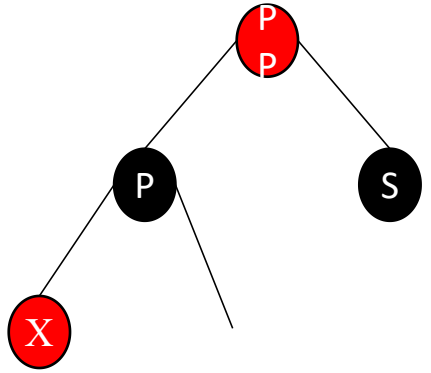
Insertion / Maintenance operation

CASE 1: The sibling S of P is Red



X : added or propagated node

Nodes P and S become Black and their parent PP becomes Red

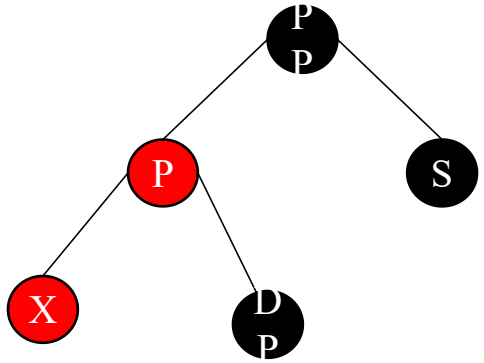


The process continues in cascade.

Red-Black Trees

Insertion / Maintenance operation

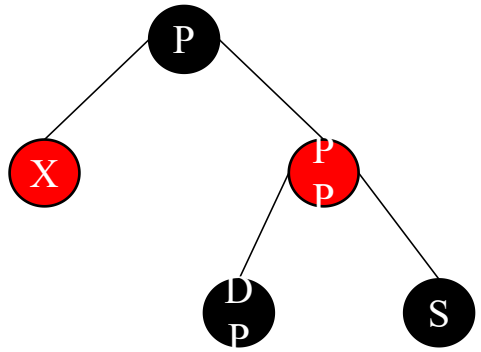
CASE 2: the sibling S of P is Black and X is the left child of P



X : added or
propagated node

Right Rotation of node PP.

P becomes Black and PP Red

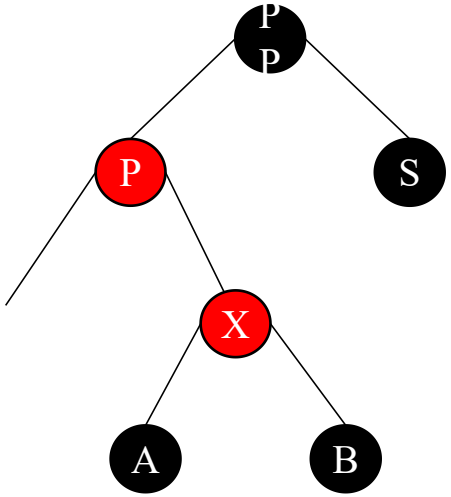


The process ends

Red-Black Trees

Insertion / Maintenance operation

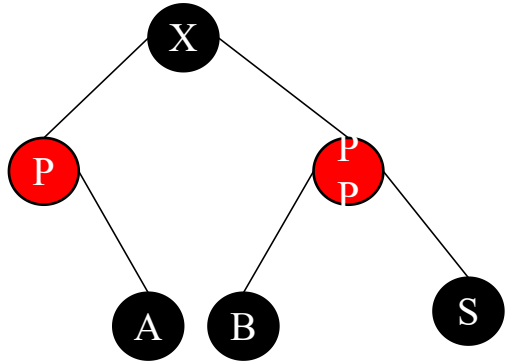
CASE 3: The sibling S of P is Black and X is the right child of P



X : added or propagated node

Left Rotation of node P + Right rotation of node PP

X becomes Black and PP Red

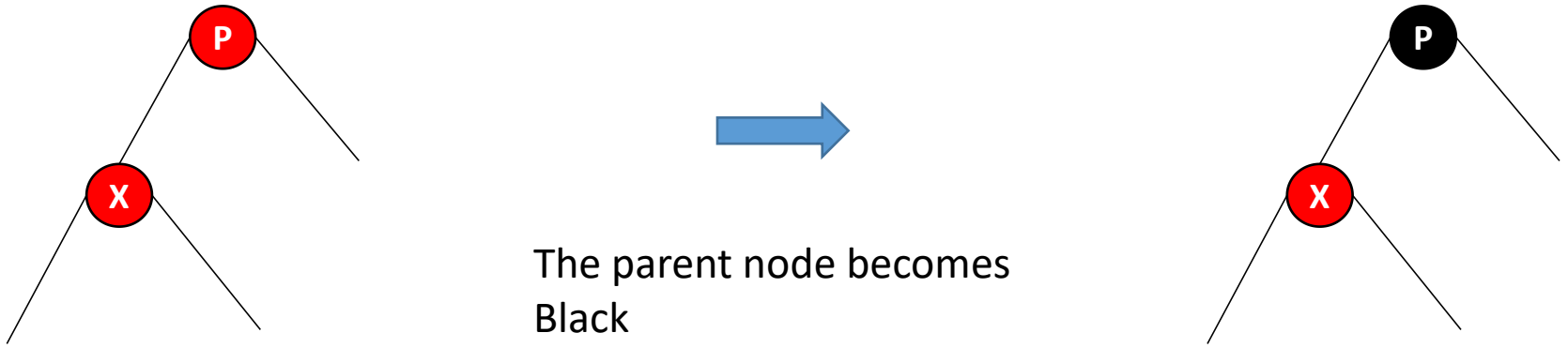


The process ends

Red-Black Trees

Insertion / Maintenance operation

CASE 4: the parent node P is the root of the tree



The parent node becomes Black

This is the only case where the Black height of the tree increases.

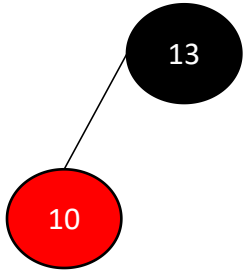
The process ends

Red-Black Trees

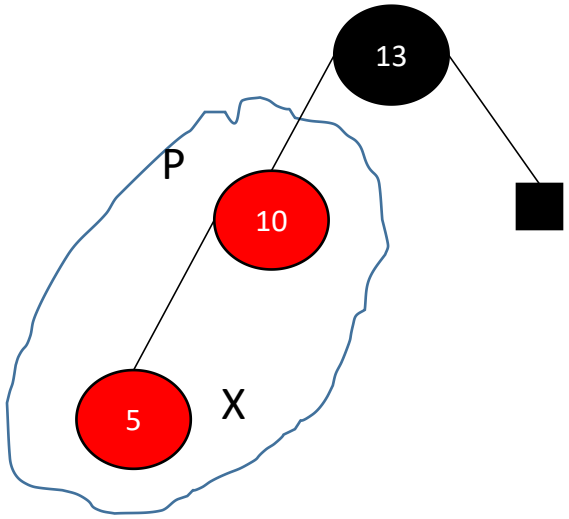
Insertion / Example

Insert **13**

Insert **10**

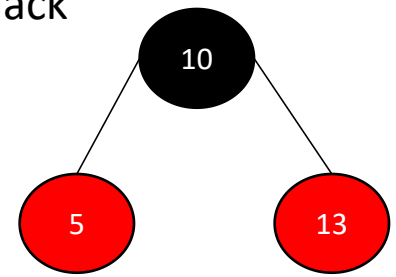


Insert **5**



Color of the sibling of P= Black
and X = Lc(P)

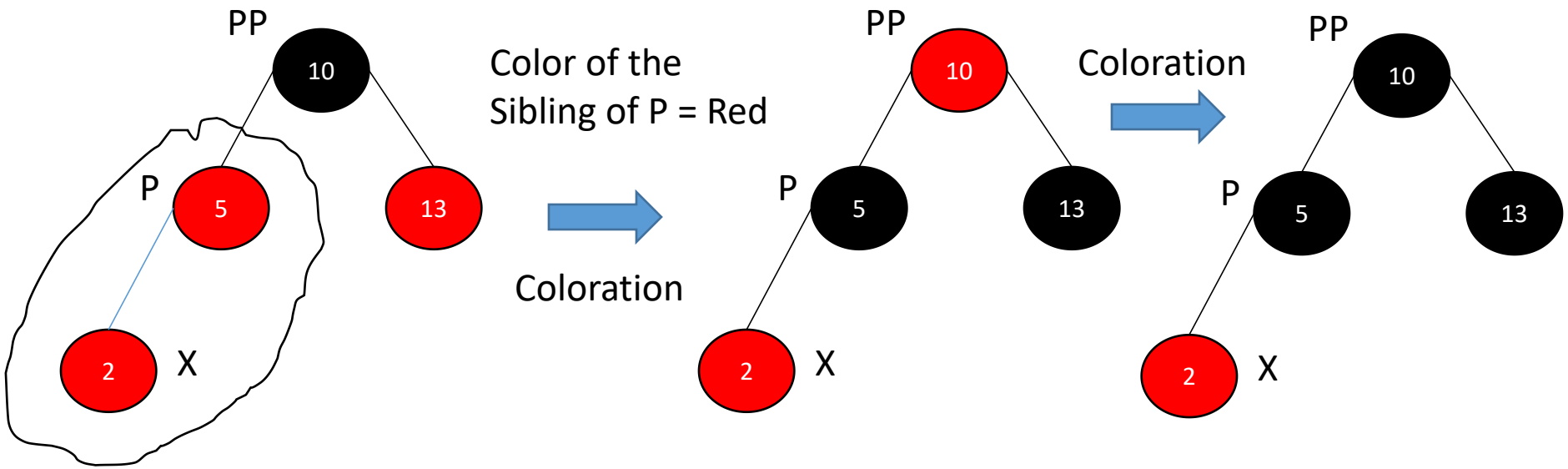
Rotation



Red-Black Trees

Insertion / Example

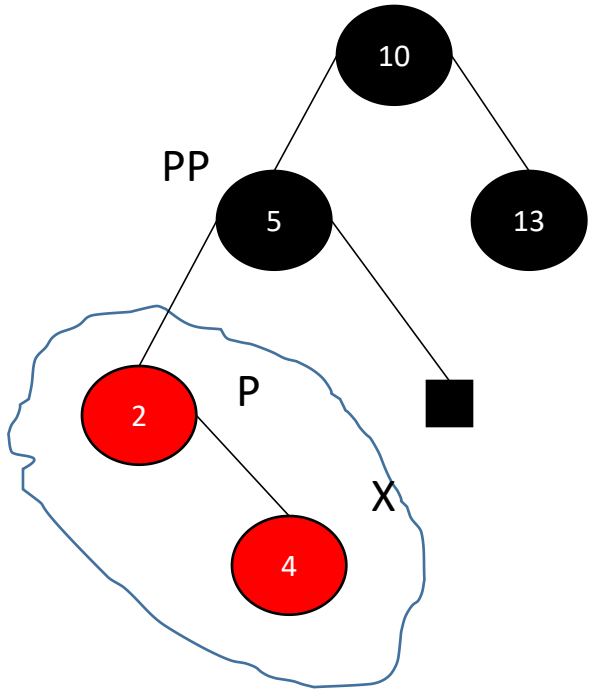
Insert **2**



Red-Black Trees

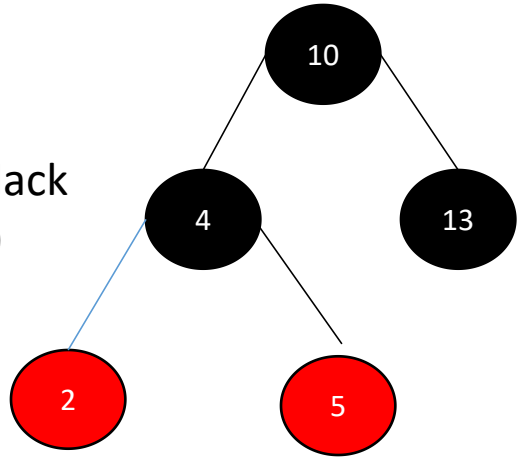
Insertion / Example

Insert **4**



Color of the Sibling of P= Black
and X = Rc(P) and P=Lc(PP)

Double rotation



Red-Black Trees

Deletion

Deletion as in a binary search tree

If the physically deleted node is black, a maintenance algorithm is applied.

It is considered that the node replacing the deleted node carries an additional black color.

This means it becomes black if it is red, and it becomes doubly black if it is already black.

The maintenance algorithm's role is to eliminate the doubly black node.

Red-Black Trees

Case 1 A :in the 2-4 tree : Merging P and S (Parent node does not give)

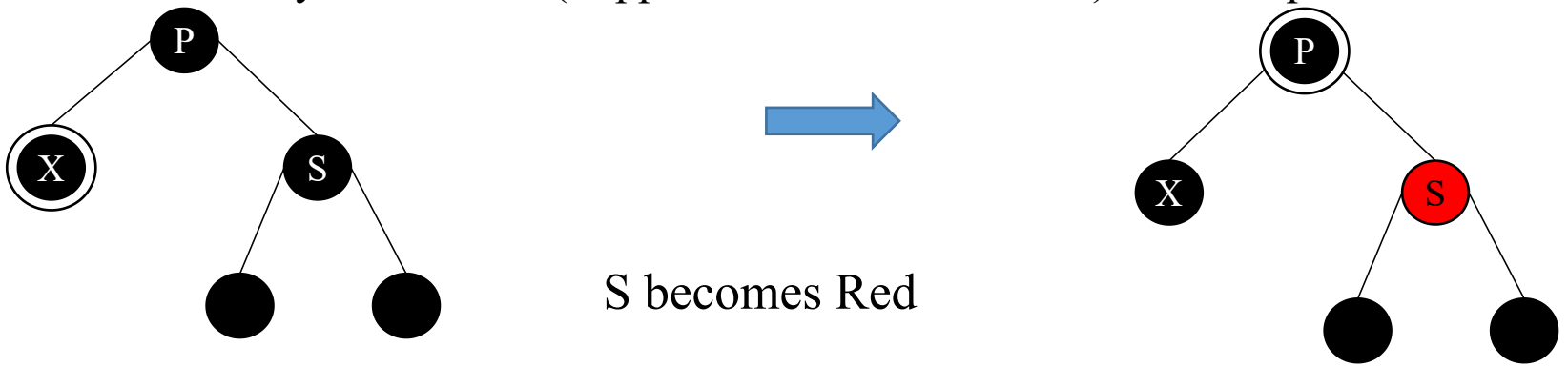
Case 1 B :in the 2-4 tree : Merging P and S (Parent node can give)

Deletion/ Maintenance operation

CASE 1: The sibling S of X is black and has two black children

Let X be the doubly black node (Supposed here as a left child) and P its parent

Case 1 A



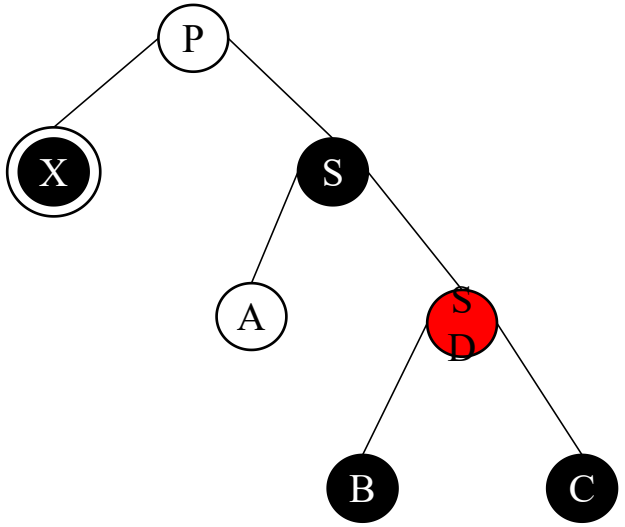
Case 1 B

Case 1 A : The parent P is Black, the process continues by ascending the tree. P becomes the new doubly black node

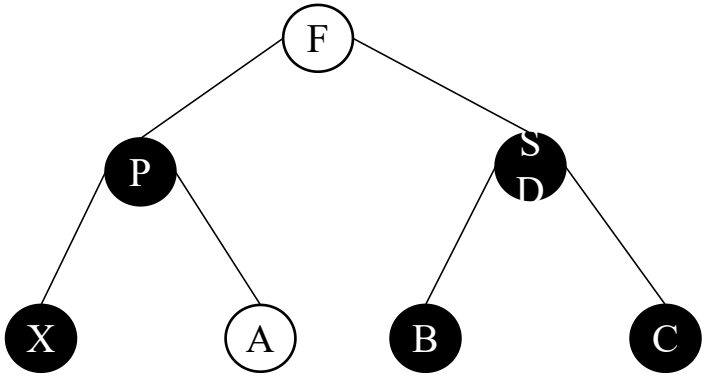
Red-Black Trees

Deletion/ Maintenance operation

CASE 2: The sibling S of node X is Black and has a Red right child (SD)



Left Rotation of node P



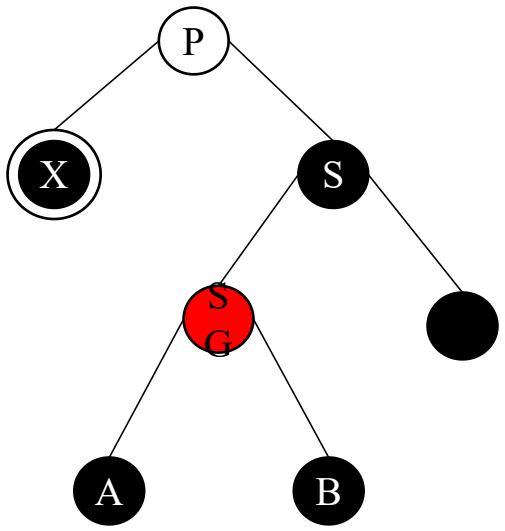
- Recolor : P and SD become Black
- The color of S is the one of P before the transformation.

The process ends

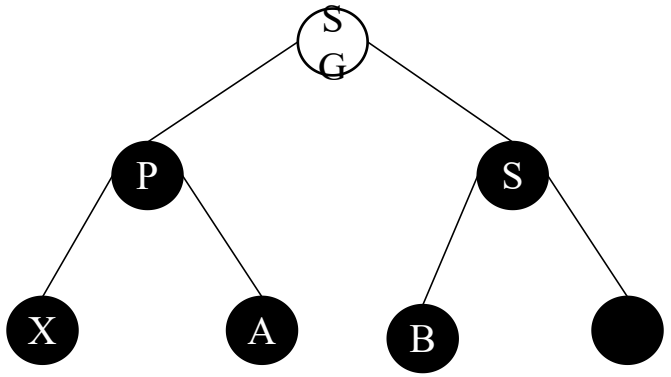
Red-Black Trees

Deletion/ Maintenance operation

CASE 3: The sibling S of node X is Black and has a Red left child(SG)



Right Rotation (S) +
Left Rotation (P)



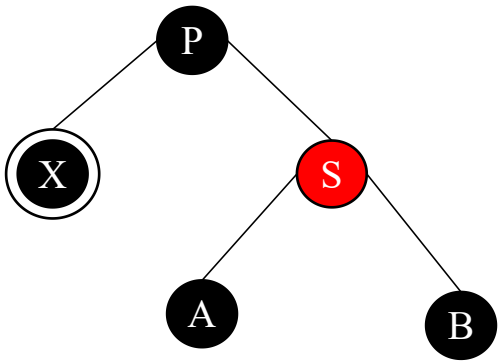
Recolor : P becomes Black
the color of SG is the one of P
before the transformation.

The process ends

Red-Black Trees

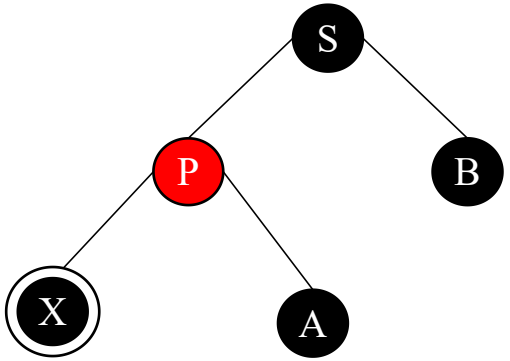
Deletion/ Maintenance operation

CASE 4: The sibling S of X is Red



Left Rotation (P)

P becomes Red and S Black



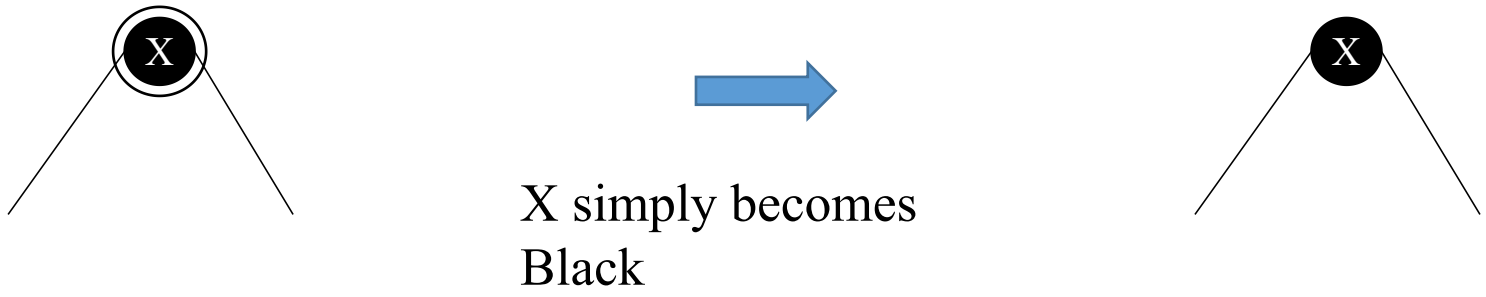
The process continues according to case 1, 2 or 3

Red-Black Trees

Deletion/ Maintenance operation

CASE 0: X is the tree root

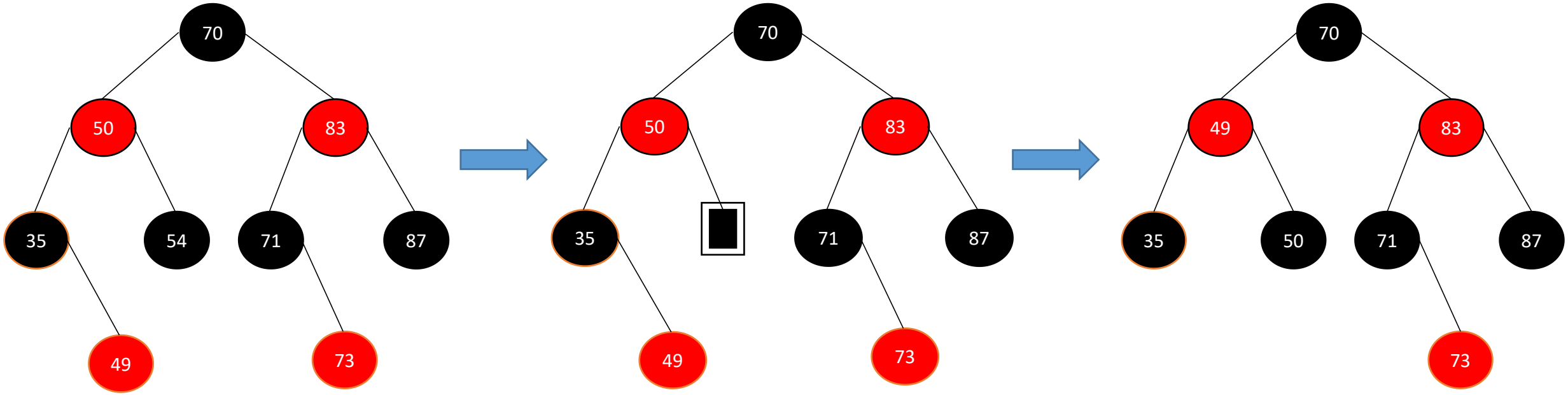
Let X be the doubly black node



It's the only case where the height of the tree decreases. The process is complete.

Red-Black Trees

Deletion/ Example



Sup 54

The node replacing the deleted node is Nil (black). Assign it an additional black color.

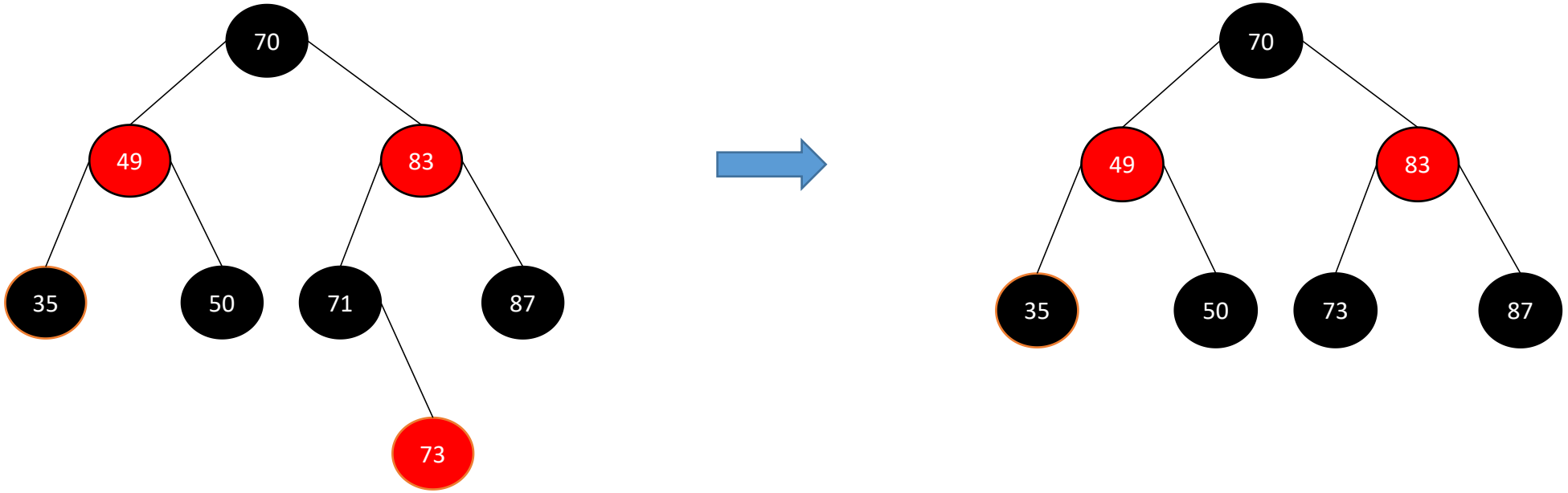
Case 3

Left Rotation(35) + Right Rotation(50)

50 becomes Black ; 49 becomes Red

Red-Black Trees

Deletion/ Example

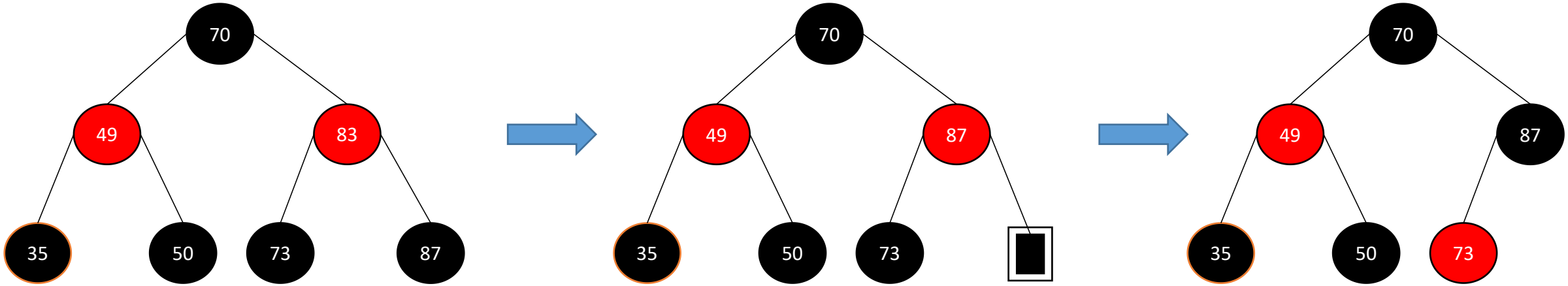


Sup 71

The node replacing the deleted node is 73 (red). Assign an additional black color to it.

Red-Black Trees

Deletion/ Example



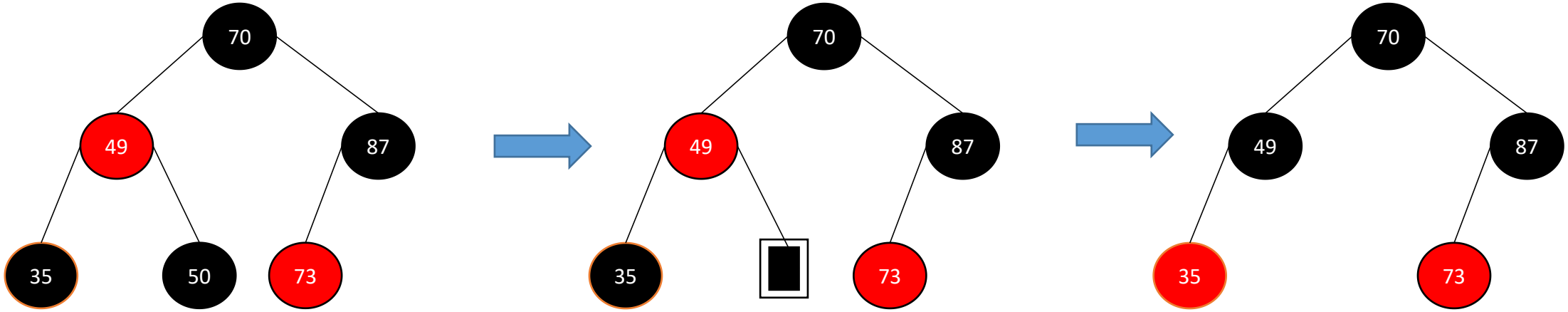
Sup 83

The node replacing the deleted node is nil (black). Assign an additional black color to it.

Case 1 B
73 becomes Red
87 becomes Black

Red-Black Trees

Deletion/ Example



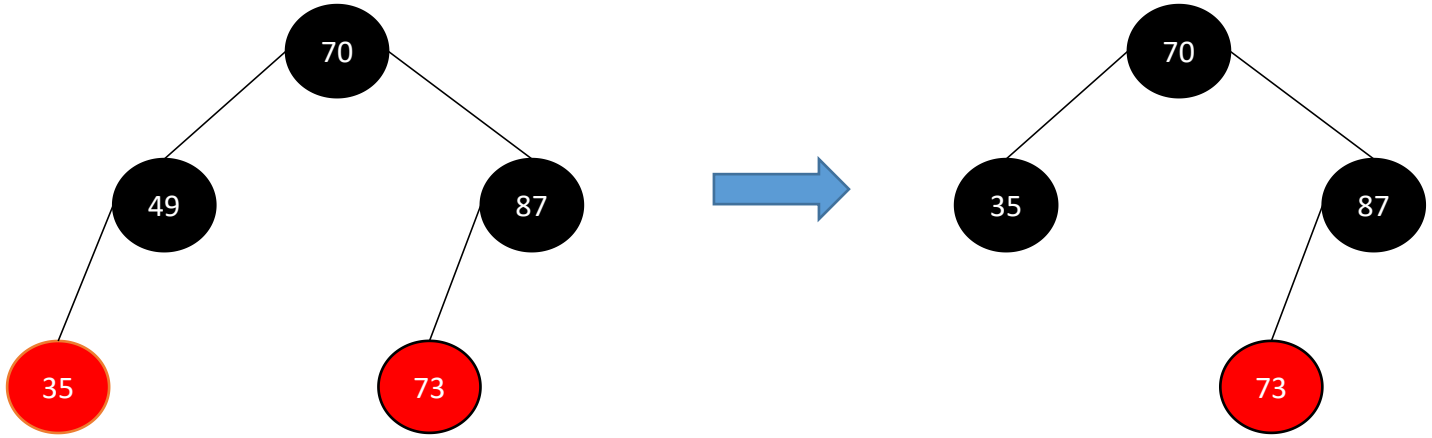
Sup 50

The node that physically replaces the deleted node is Nil (Black). Assign an additional black color to it.

Case 1 B
35 becomes Red
49 becomes Black

Red-Black Trees

Deletion/ Example

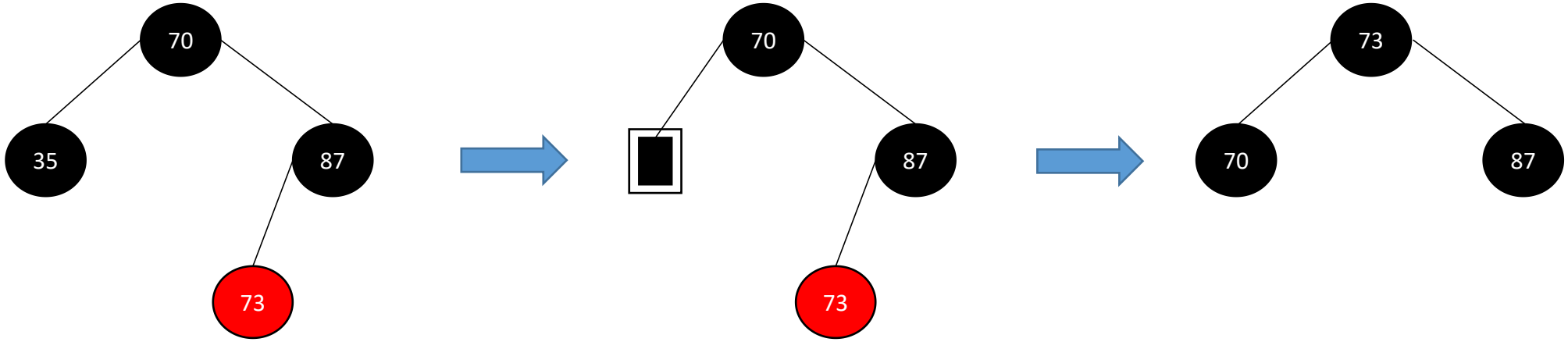


Sup 49

The node replacing the deleted node is 35 (Red). Assign an additional black color to it.

Red-Black Trees

Deletion/ Example



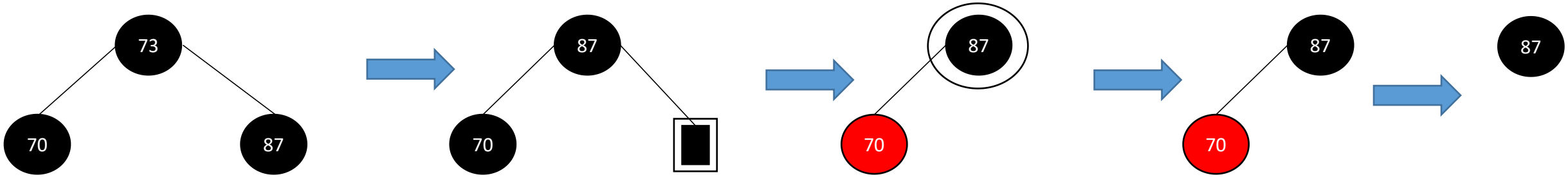
Sup 35

The node that physically replaces the deleted node is nil (Black). Assign an additional black color to it.

Case 3
Right Rotation(87) + Left Rotation(70)
70 and 73 become Black

Red-Black Trees

Deletion/ Example



Sup 73

The node that physically replaces the deleted node is nil (Black). Assign an additional black color to it.

Case 1 A
70 becomes Red
87 becomes Black

Sup 70

The node replacing the deleted node is 70 (Red). Assign an additional black color to it.

Red-Black Trees

Synthesis

The maximum depth of a balanced binary tree is 2 * log2(n).

Searching in such a tree never requires more than 100% additional comparisons than for a complete binary tree.

Maintenance Operations:
- Restructuring and coloring.
  Insertion: up to 1 restructuring and up to Log2(N) colorations.
  Deletion: up to 2 restructurings and up to Log2(N) colorations.