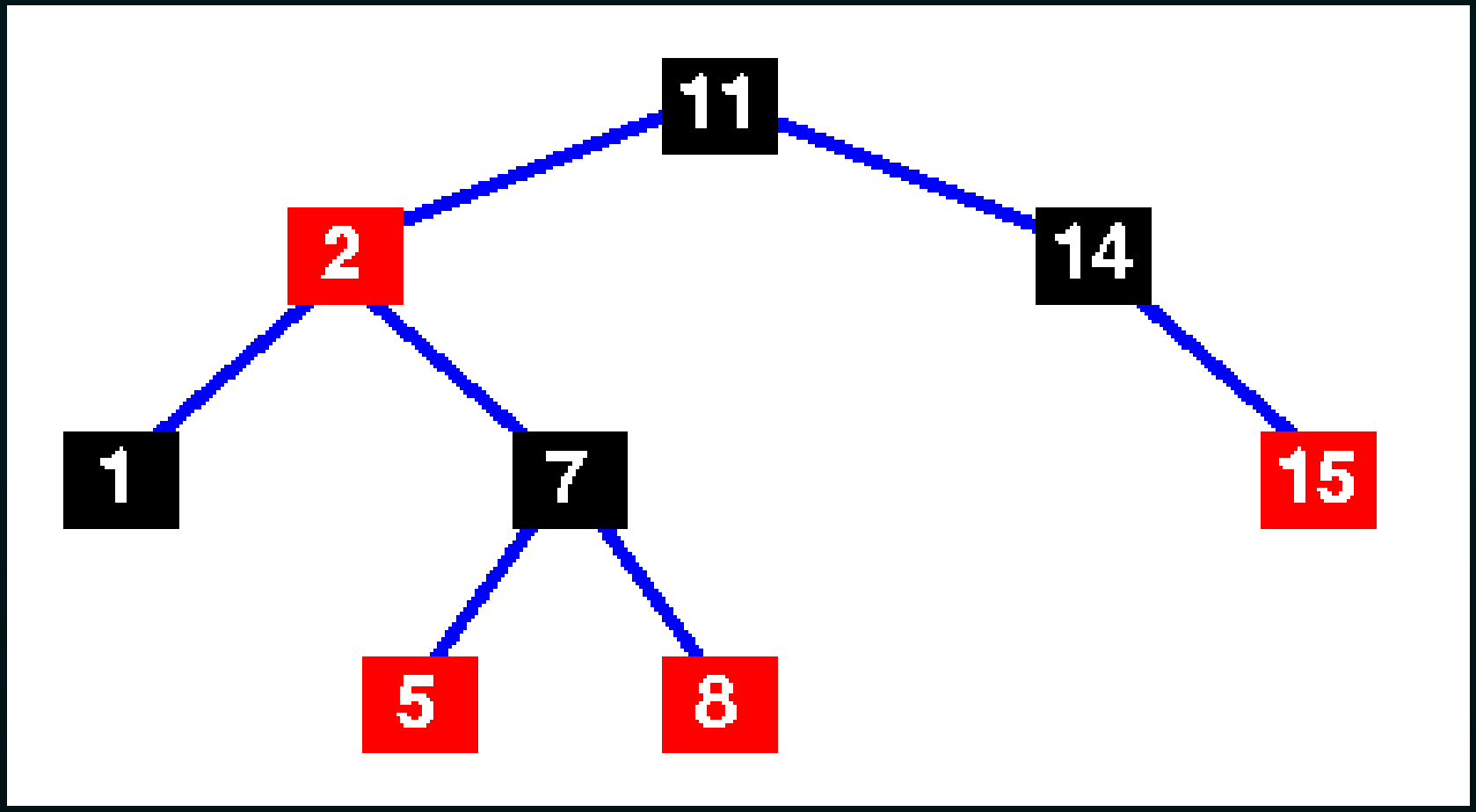
Introduction

- Red-black Tree
  - Self balancing binary search tree with one extra attribute for each node: the colour, which is either red or black.
    - colours - used to ensure that the tree remains balanced during insertions and deletions
  - Although the balance of the tree is not perfect,
    - good enough to reduce the searching time and maintain it around O(log n) time
  - Invented in 1972 by Rudolf Bayer

Introduction

- Red-black Tree Example



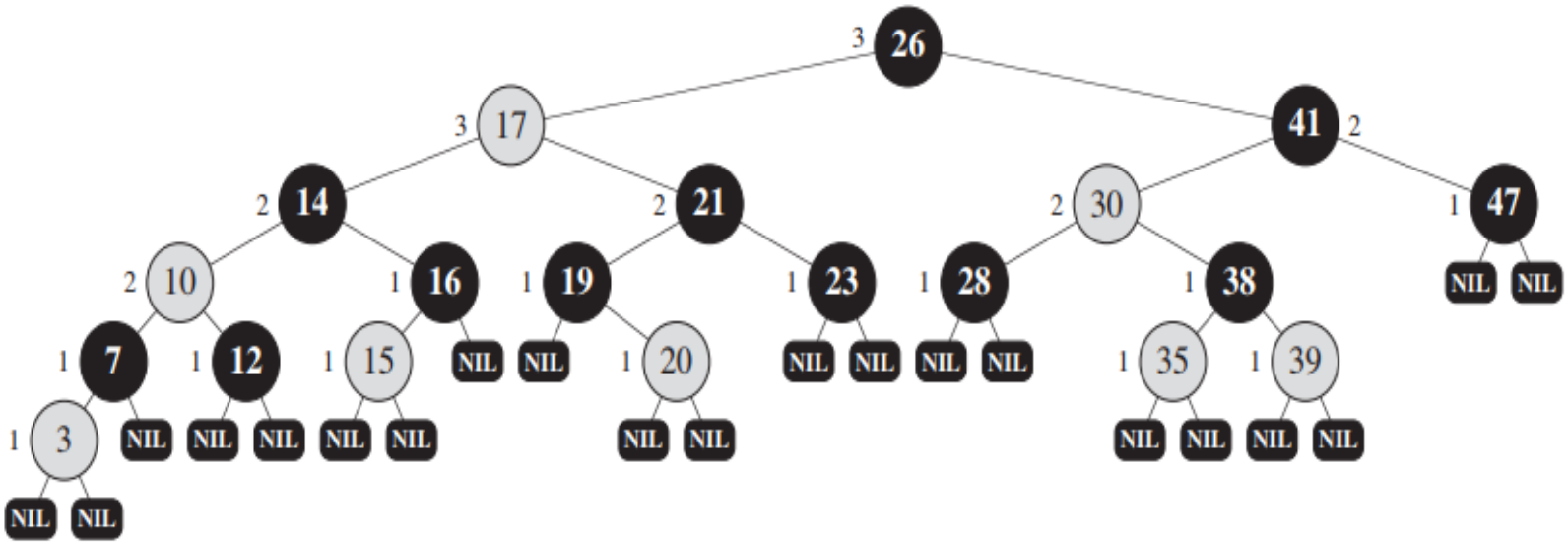
Red-Black-Trees Properties

- A red-black tree satisfies the following properties:
  1. Every node is either red or black.
  2. The root is black.
  3. Every leaf (NIL) is black.
  4. If a node is red, then both its children are black.
  5. For each node, all simple paths from the node to descendant leaves contain the same number of black nodes.

Cont…

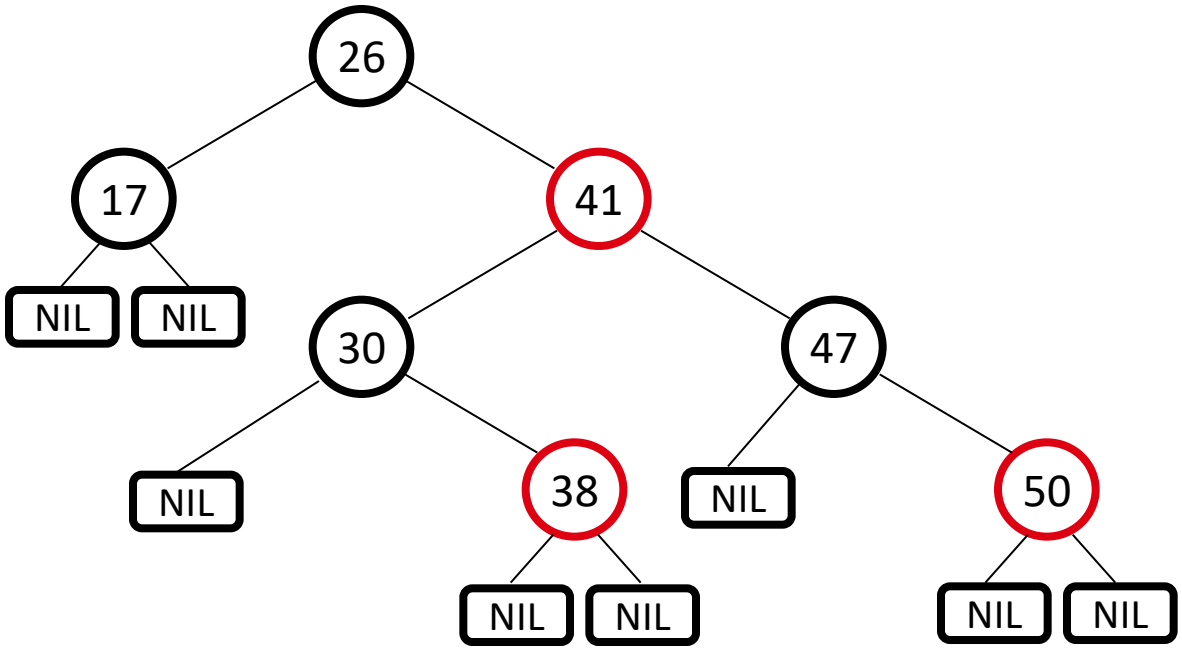
- Important Note:
  - The longest path from the root node to any leaf node
    - no more than twice as long as the shortest path from the root to any other leaf in that tree
  - Black-height of a node x: bh(x)
    - the number of black nodes (including NIL) on the path from x to a leaf, not including x

Cont…



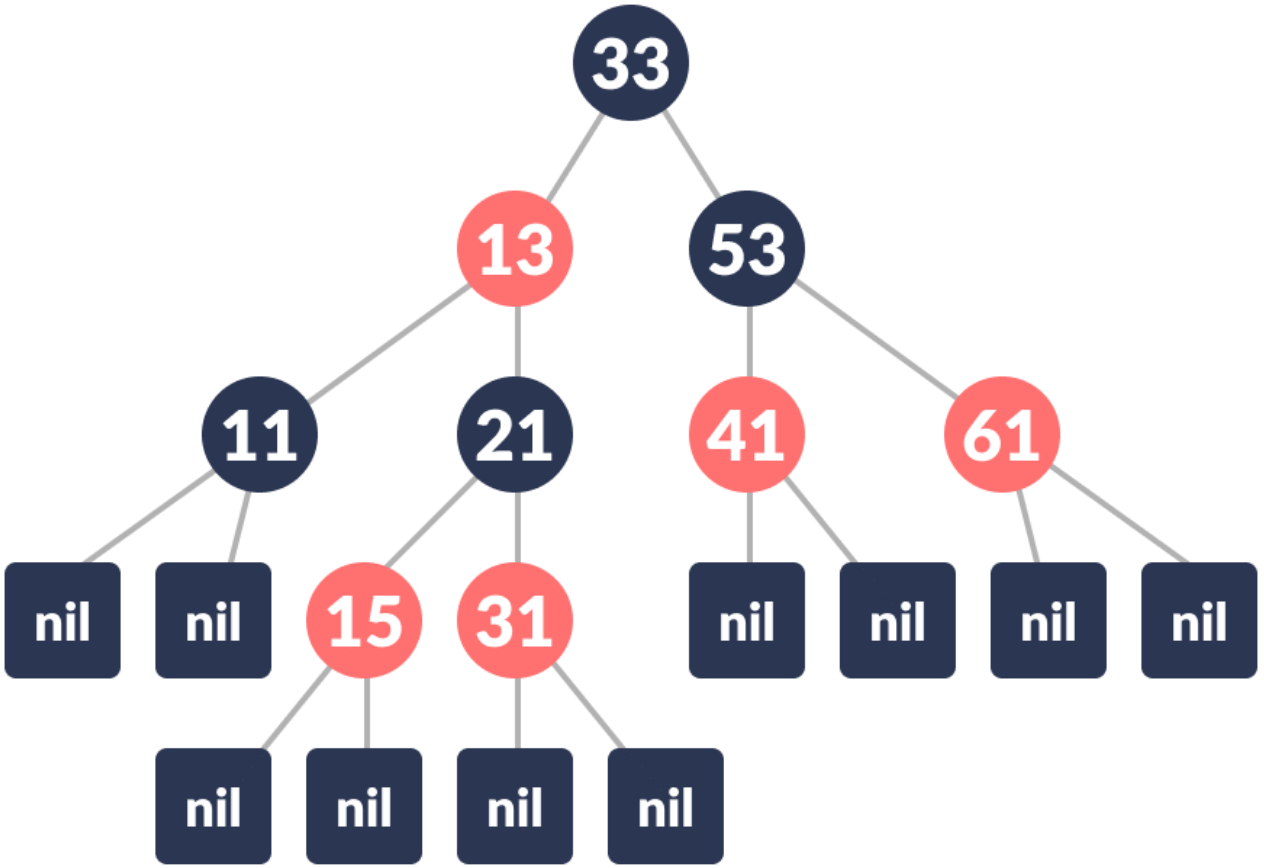
Red-Black Tree

Cont…



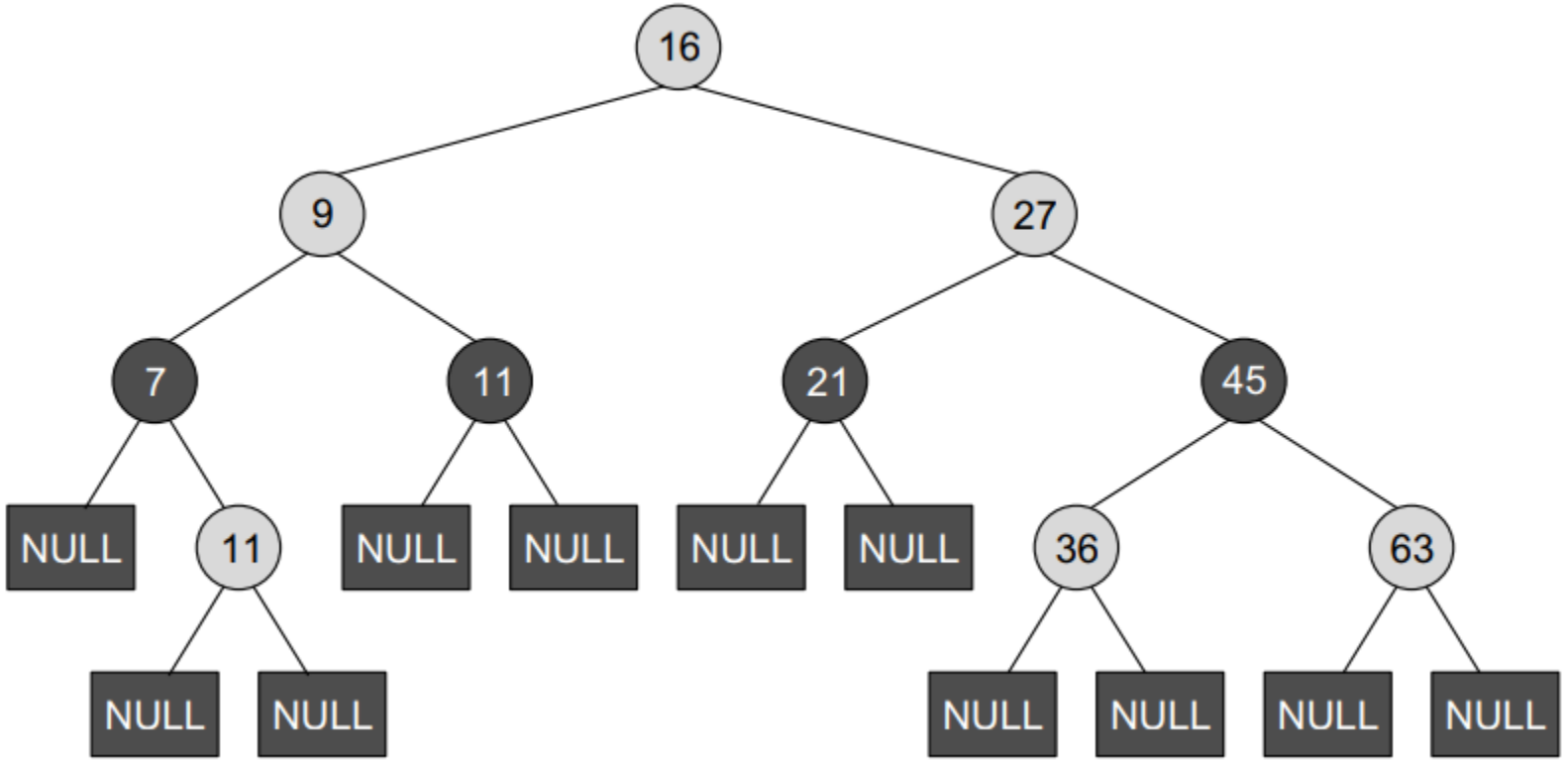
Red-Black Tree

Cont…



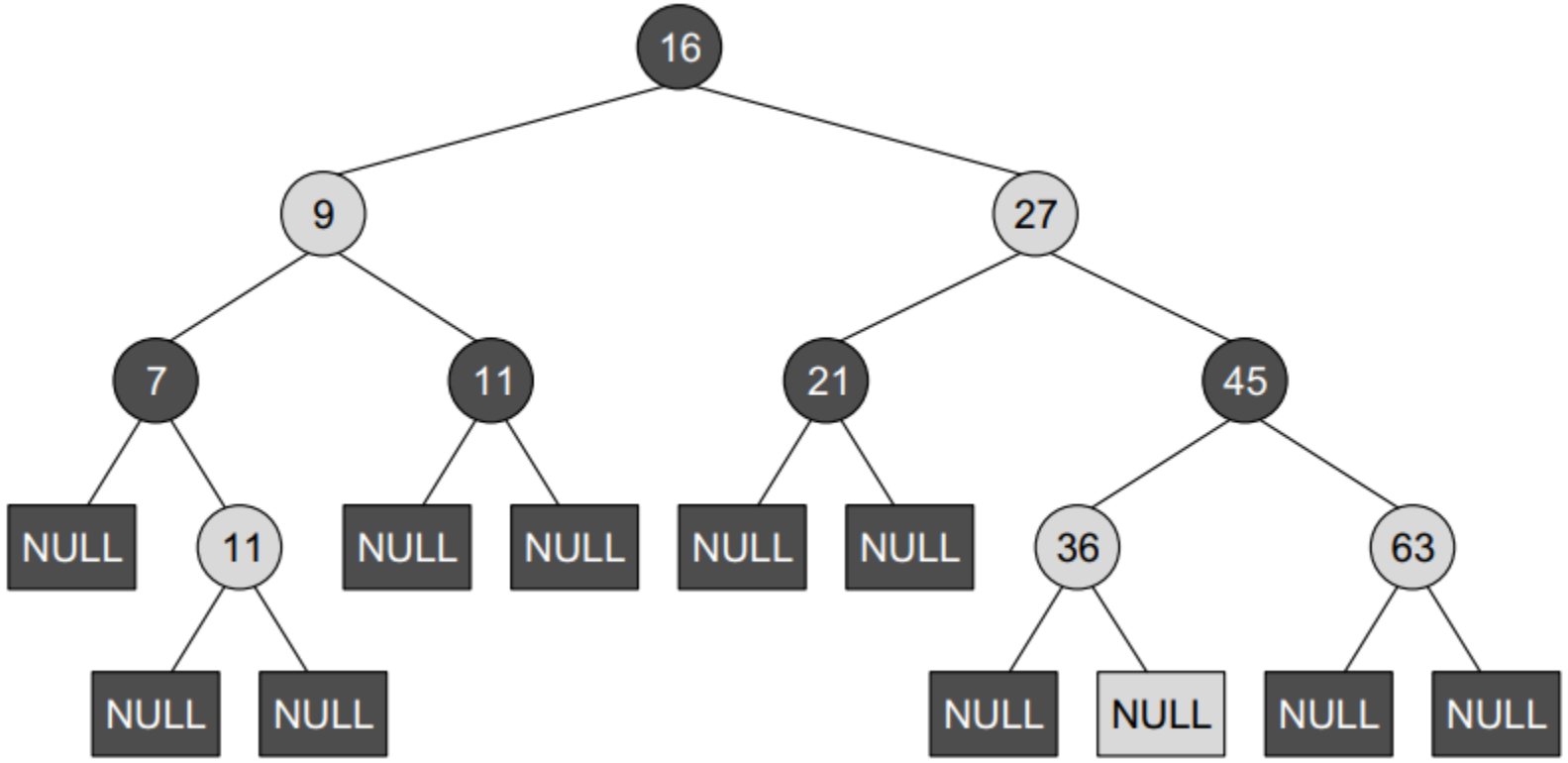
Red-Black Tree

Cont…



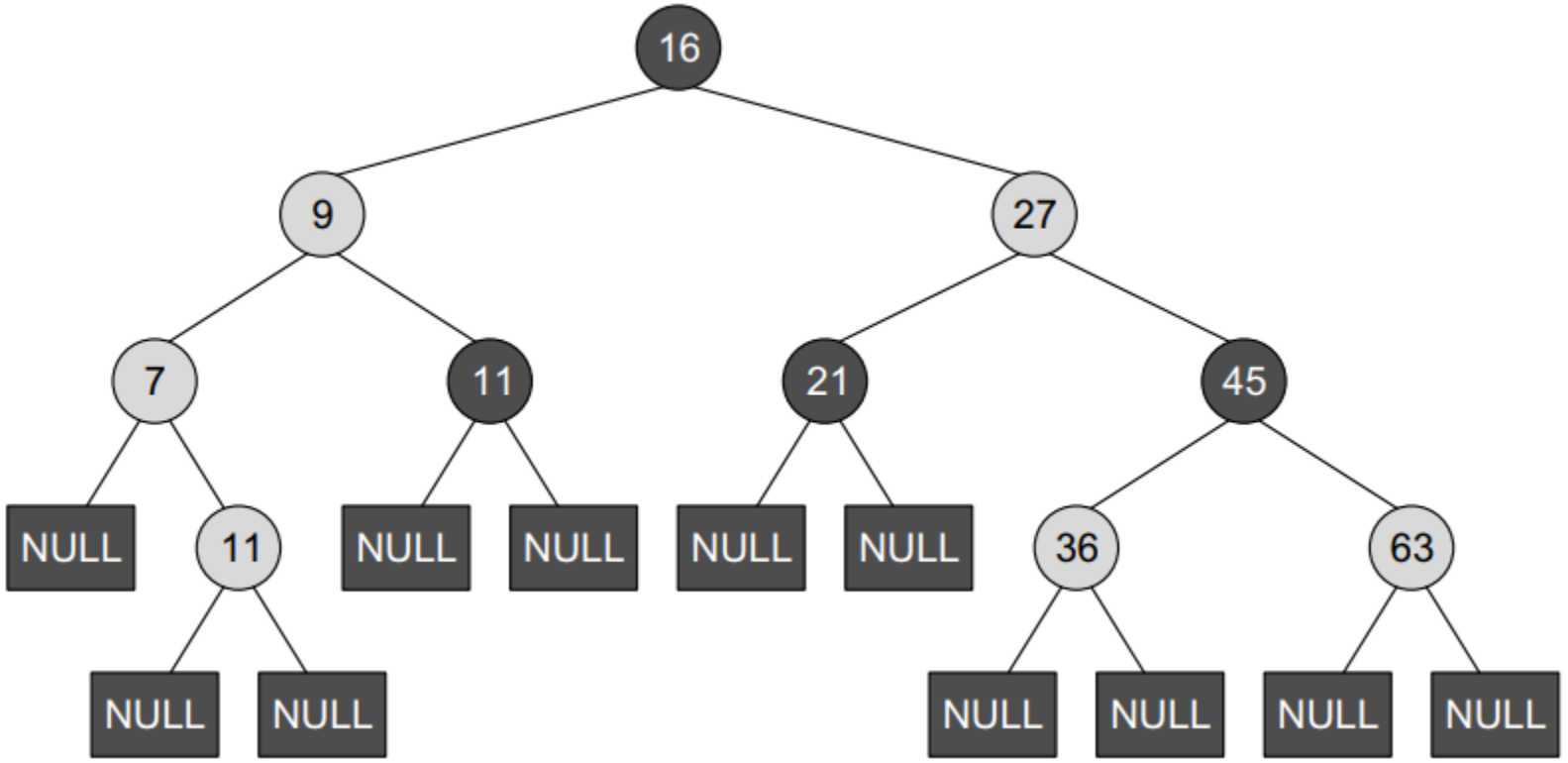
Not a Red-Black Tree

Cont…



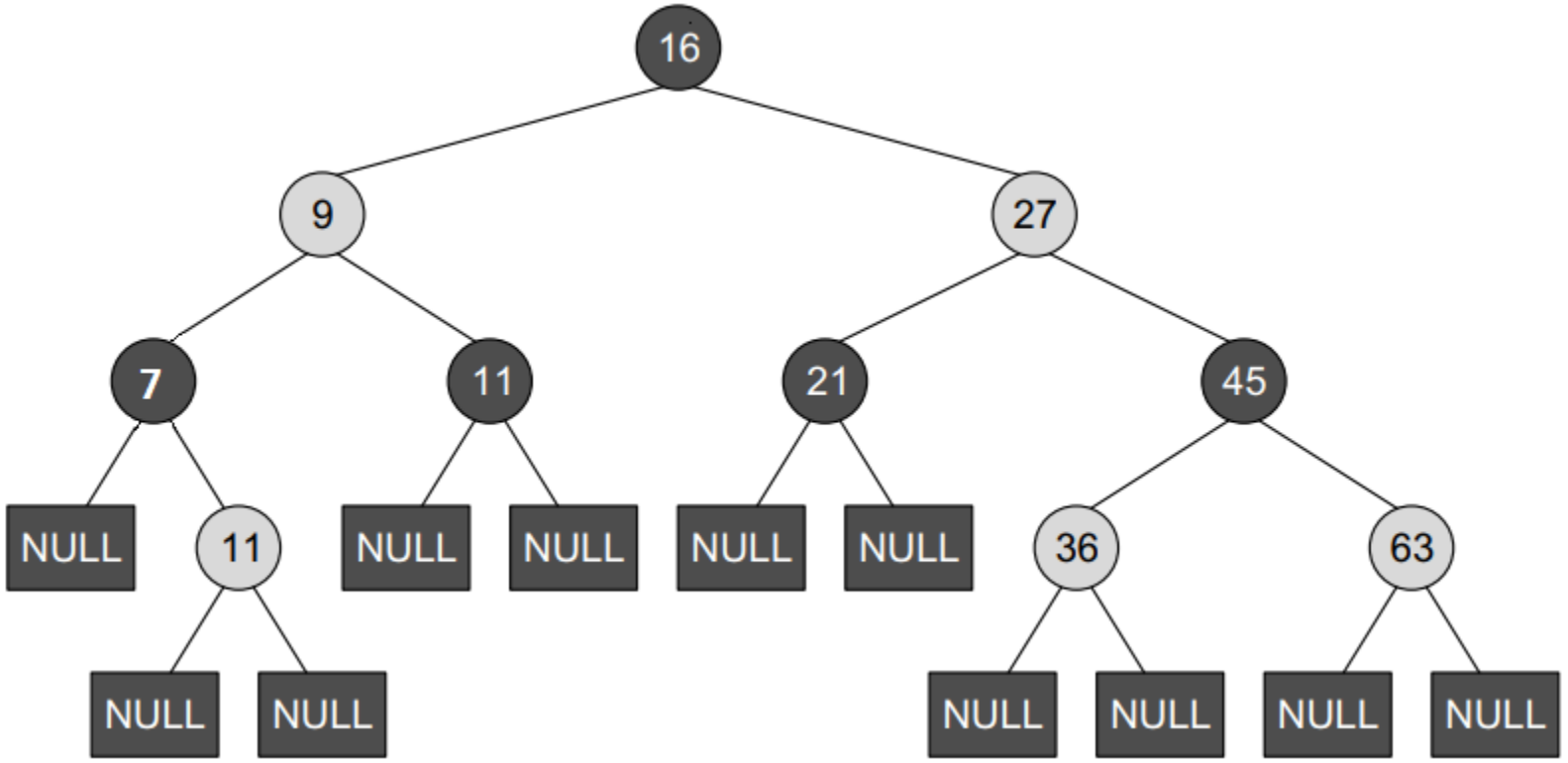
Not a Red-Black Tree

Cont…



Not a Red-Black Tree

Cont…



Red-Black Tree

Cont…

- Why Red-Black Trees?
  - In BST, most operations (e.g., search, max, min, insert, delete etc) take O(h) time, and the cost of these operations may become O(n) for a skewed Binary tree
  - If the height of the tree remains O(log n) after every insertion and deletion, then an upper bound of O(log n) for all these operations can be guaranteed
  - In Red-Black tree
    - Height of a Red-Black tree is always O(log n) where n is the number of nodes in the tree
- How does a Red-Black Tree ensure balance?
  - A chain of 3 nodes is not possible in the Red-Black tree

Cont…

- Comparison with AVL Tree
  - AVL trees are more balanced compared to Red-Black Trees
  - AVL tree may cause more rotations during insertion and deletion
  - When your application involves frequent insertions and deletions, then Red-Black trees is more preferable

Cont…

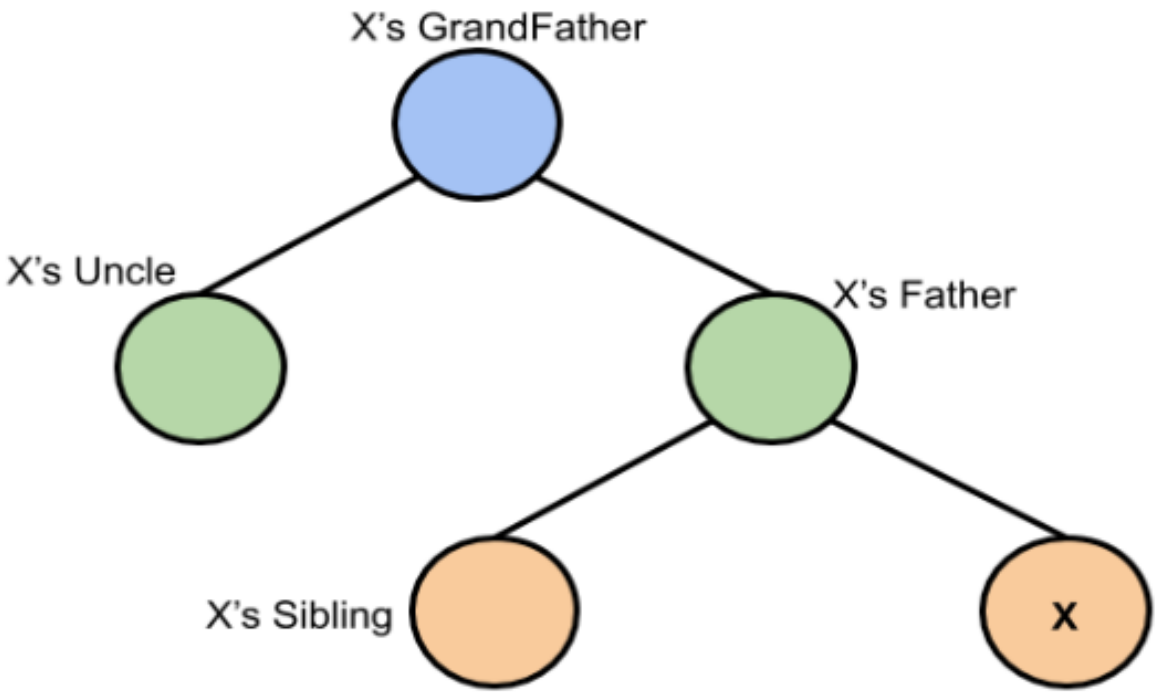
- Interesting points about Red-Black Tree
  - Black height of the red-black tree
    - No. of black nodes on a path from the root node to a leaf node (excluding the root)
    - A red-black tree of height h has black height >= h/2
  - Height of a red-black tree with n nodes is $h <= 2 \log_2(n+1)$
  - All leaves (NIL) are black
  - Black depth of a node
    - No. of black nodes from the root to that node
      - i.e the number of black ancestors
  - Every red-black tree is a special case of a binary tree

Cont…

- In the Red-Black tree, the balancing is done with the aid of
  - Recolouring
    - the change in colour of the node i.e. if it is red then change it to black and vice versa
  - Rotation
    - LL rotation, RR rotation, LR rotation and RL rotation

Cont…

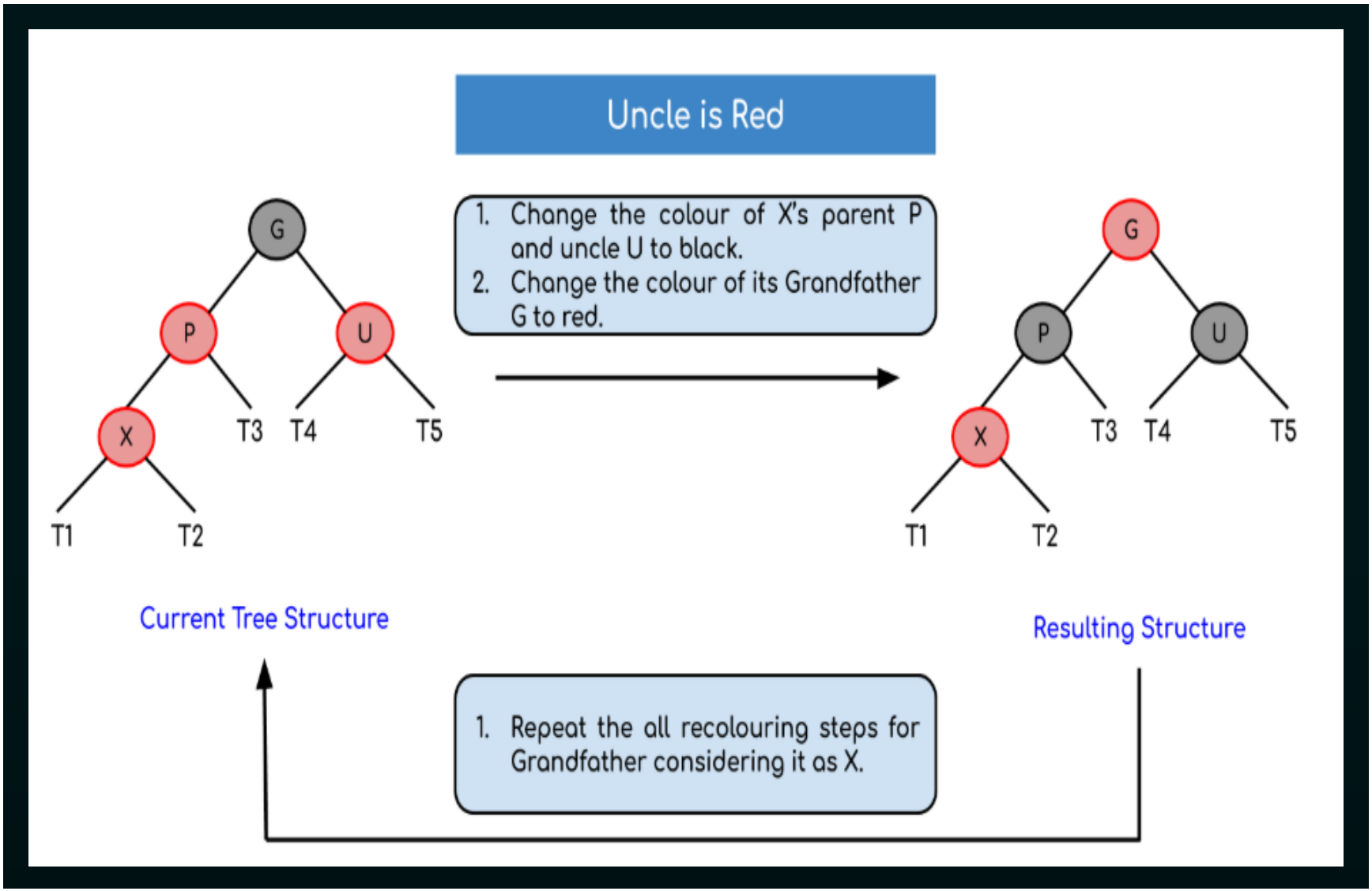
- Two cases depending upon the colour of the uncle
  - If the uncle is red, we do recolour
  - If the uncle is black, do rotations and/or recolouring



Insertion in RB Tree

- Insert the node as in a BST and assign a red colour to it
- If the inserted node is a root node
  - change its colour to black
- Else check the colour of the parent node
  - If colour is black then don't change the colour
  - Else colour is red then check colour of the node's uncle
    - If node's uncle has a red colour then change colour of
      - node's parent and uncle to black
      - grandfather to red colour
      - repeat the same process for grandfather

Cont…

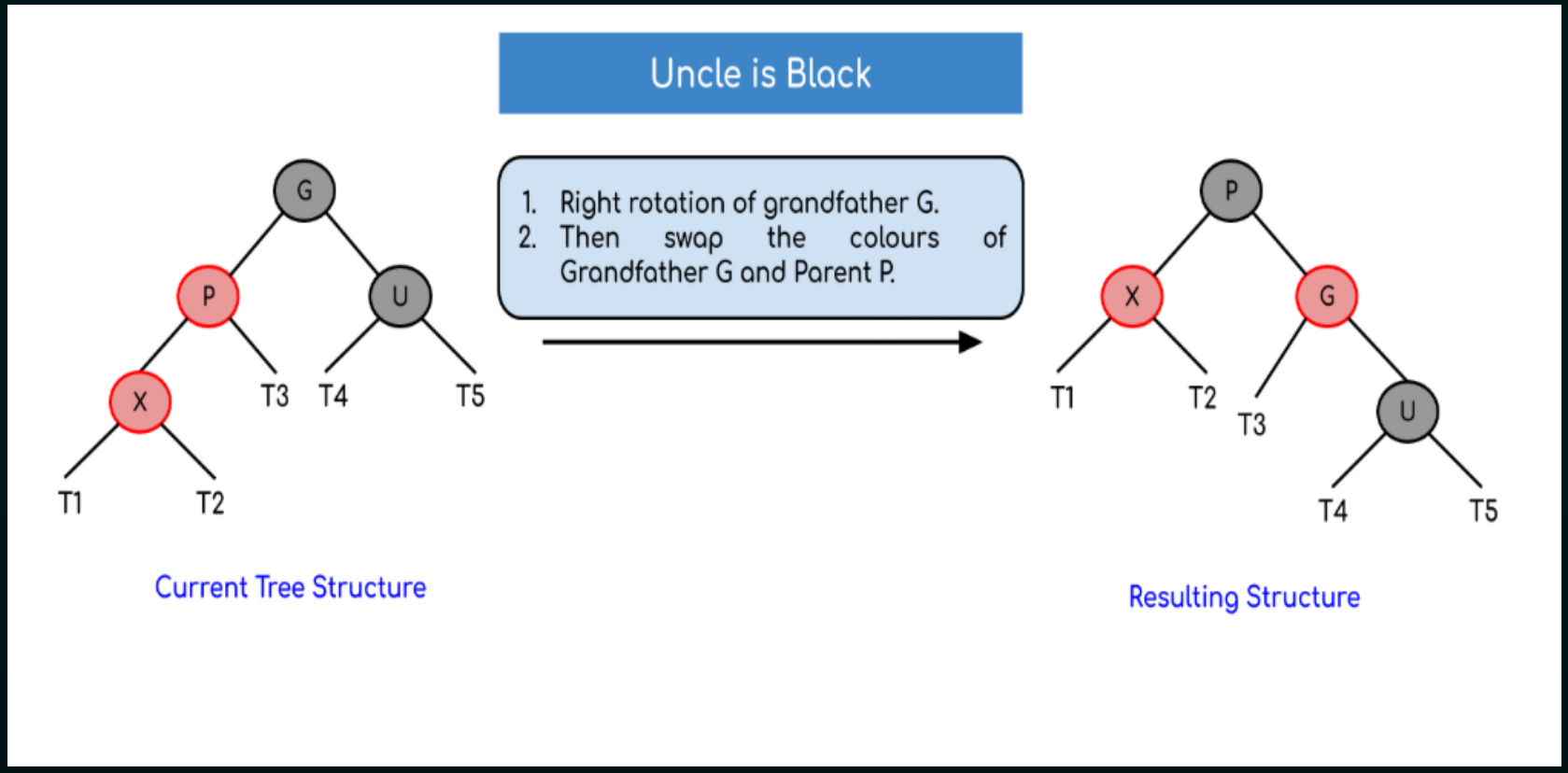

Uncle is Red
1. Change the colour of X's parent P and uncle U to black.
2. Change the colour of its Grandfather G to red.
G
P
U
X
T1
T2
T3
T4
T5
Current Tree Structure
Resulting Structure
1. Repeat the all recolouring steps for Grandfather considering it as X.

Cont…

- If node's uncle has a black colour then four case are possible
  - Left Left Case (LL rotation)
  - Left Right Case (LR rotation)
  - Right Right Case (RR rotation)
  - Right Left Case (RL rotation)

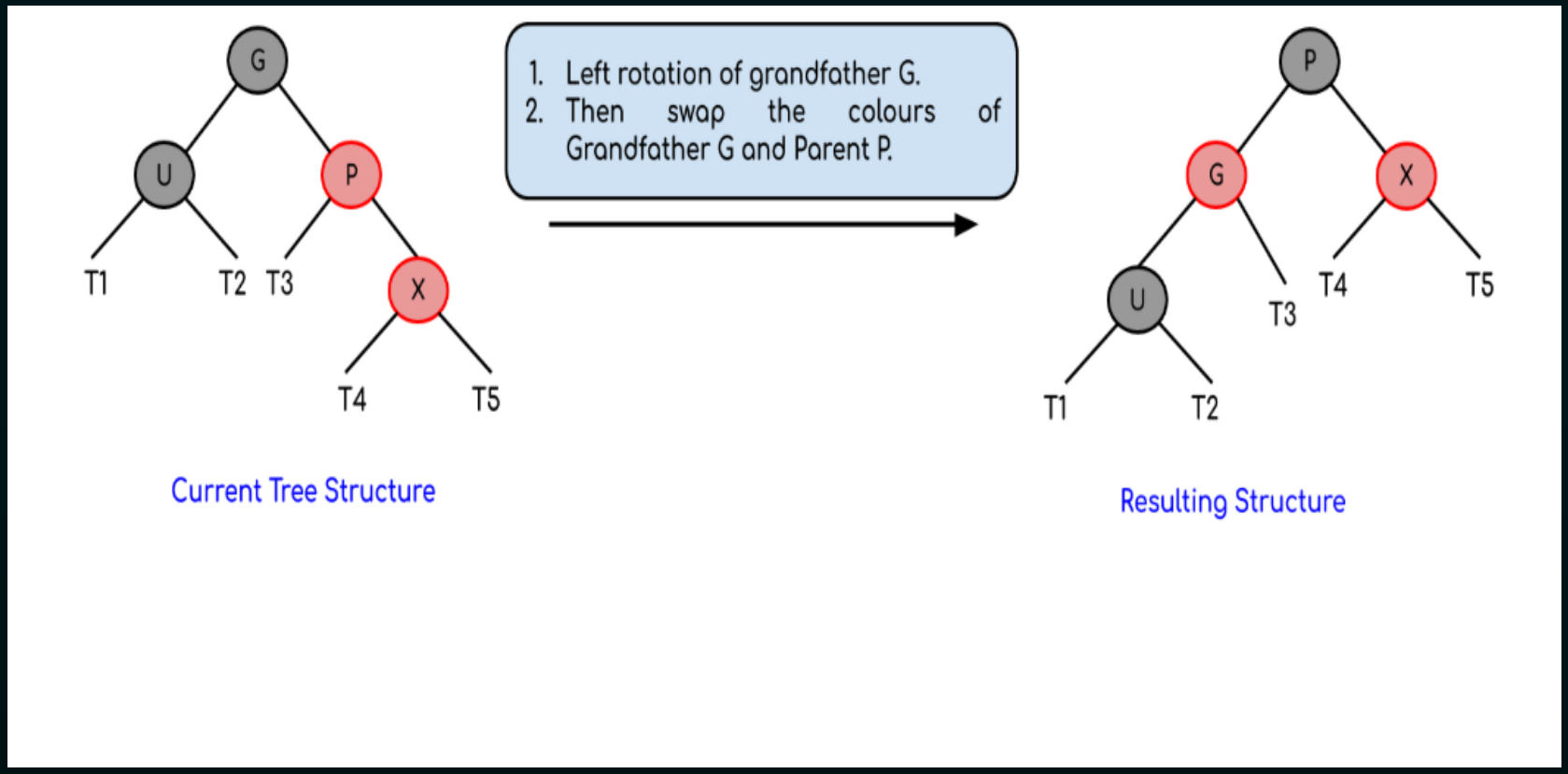
Cont…

- Left Left Case (RR rotation)



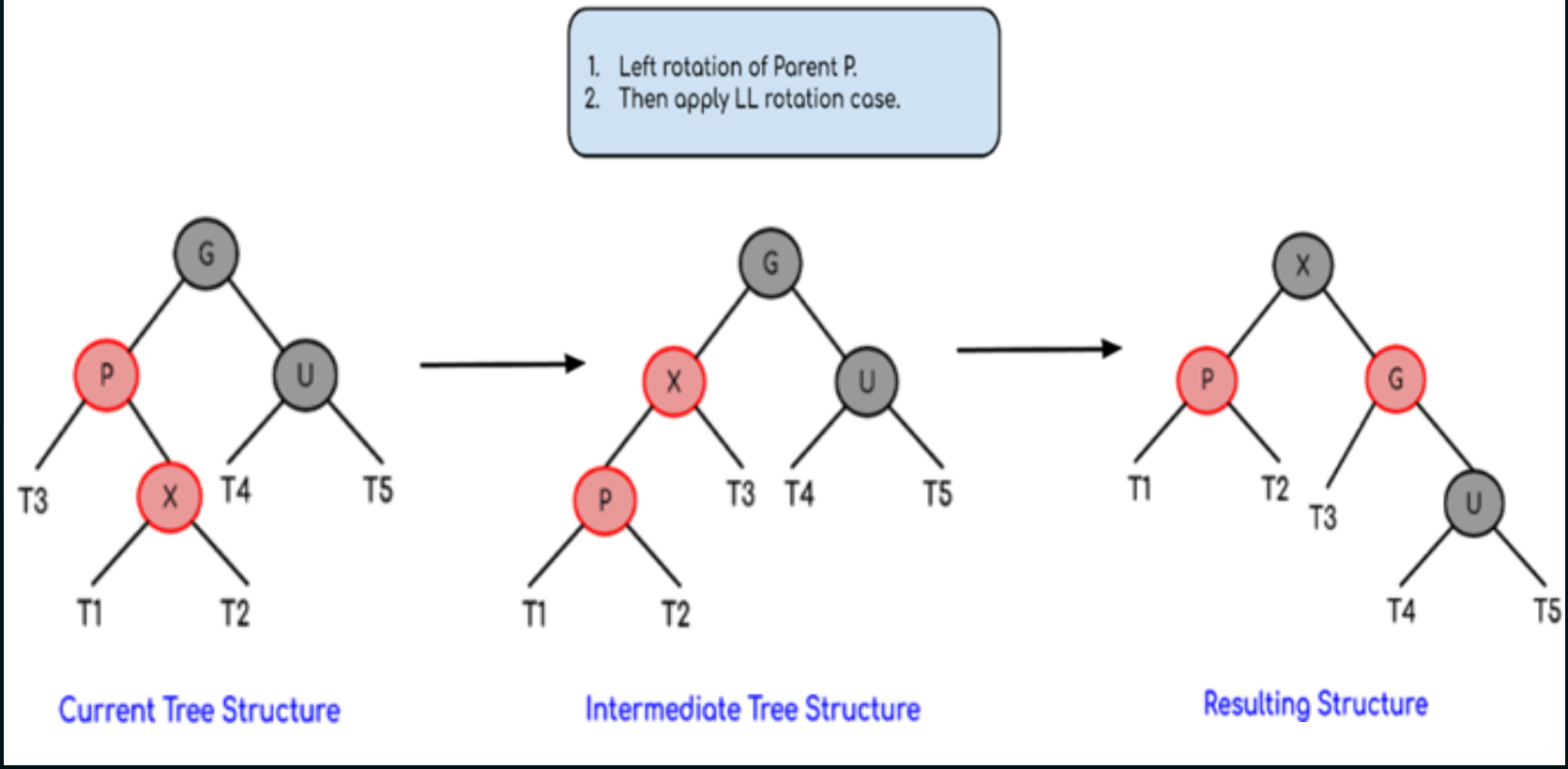
Cont…

- Right Right Case (LL rotation):



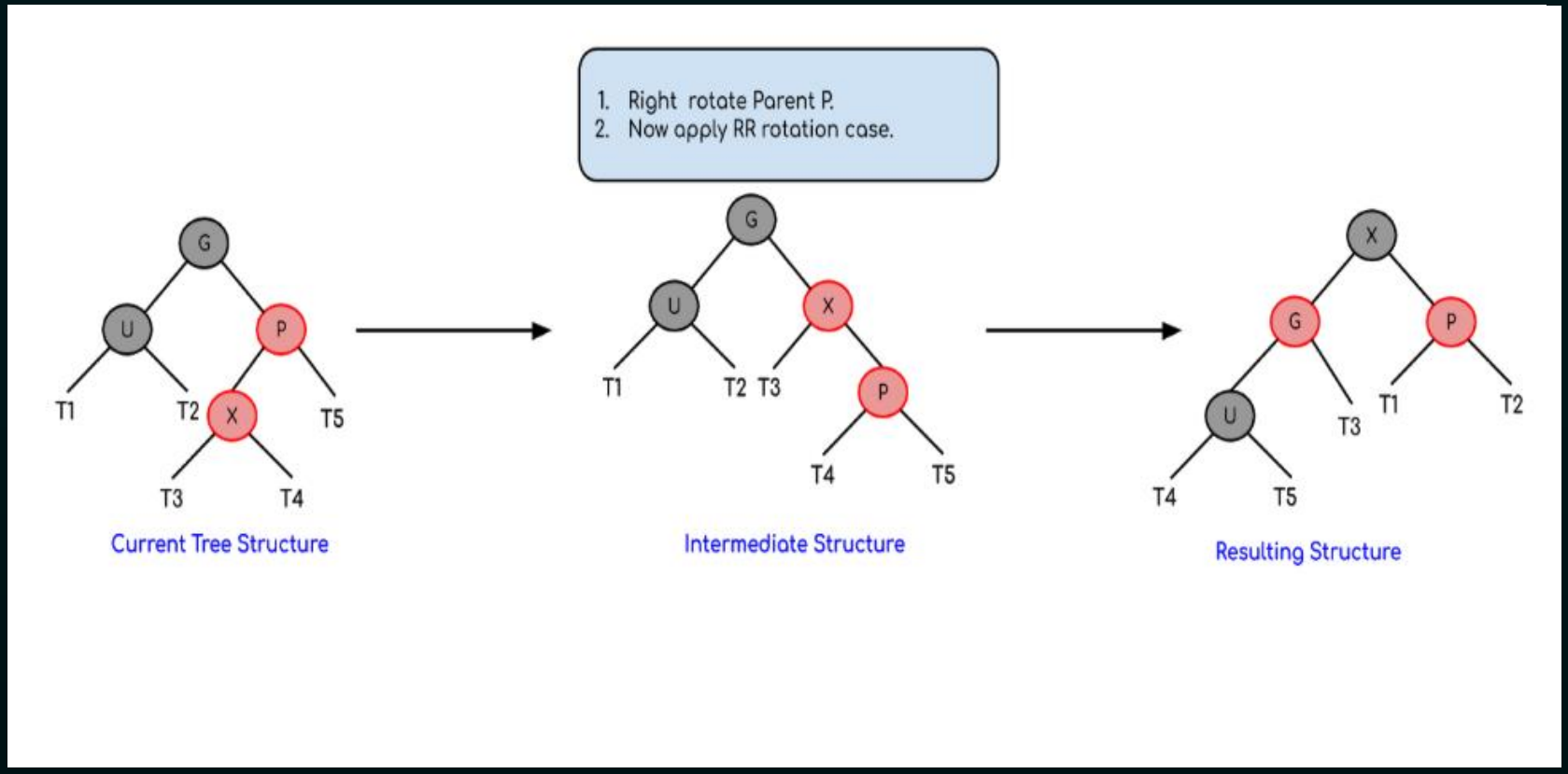
Cont…

- Left Right Case (LR rotation):



Cont…

- Right Left Case (RL rotation):



Cont…

- Insertion Algorithm: x be the newly inserted node
  1. Perform standard BST insertion and make the color of newly inserted nodes as RED
  2. If x is the root, change the color of x as BLACK
  3. x is not the root and the color of x's parent is RED
     a) If x's uncle is RED (Grandparent must have been black from property 4)
        i. Change the color of parent and uncle as BLACK.
        ii. Color of a grandparent as RED.
        iii. Change x = x's grandparent, repeat steps 2 and 3 for new x.
     b) If x's uncle is BLACK, then there can be four configurations for x, x's parent (p) and x's grandparent (g)
        i. Left Left Case (p is left child of g and x is left child of p) –RR rotation
        ii. Left Right Case (p is left child of g and x is the right child of p) – LR rotation
        iii. Right Right Case (Mirror of case i) – LL rotation
        iv. Right Left Case (Mirror of case ii) – RL rotation

Cont…

- Applications:
  - Most of the self-balancing BST library functions like map and set in C++ (OR TreeSet and TreeMap in Java) use Red-Black Tree.
  - Used to implement CPU Scheduling Linux. Completely Fair Scheduler uses it.
  - Used in the K-mean clustering algorithm for reducing time complexity.
  - MySQL also uses the Red-Black tree for indexes on tables.