# ZAMAN UNIVERSITY



Data Structures and Algorithms

Chapter 5

# Tree

# Outline



- Binary Trees
- Traversing Binary Tree
- Red-Black Trees
- Red-Black Tree Insertions
- 2-3-4 Trees

# Outline



- Binary Trees
- Traversing Binary Tree
- Red-Black Trees
- Red-Black Tree Insertions
- 2-3-4 Trees

# Red-Black Tree Characteristics



- Red-Black Tree Characteristics:
  - The nodes are colored;
  - During insertion and deletion, rules are followed.

- Colored nodes, in Red-Black tree every node is either red or black.
- Red-Black rules:

1. Every node is either red or black.
2. The root is always black.
3. If a node is red, its children must be black.
4. Every path from the root to a leaf, or to a null child, must contain the same number of black nodes.

# The Actions



- There are two actions to make Red-Black tree balancing:
  - Recolor (Change Color)
  - Rotations.

Try red-black tree on https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

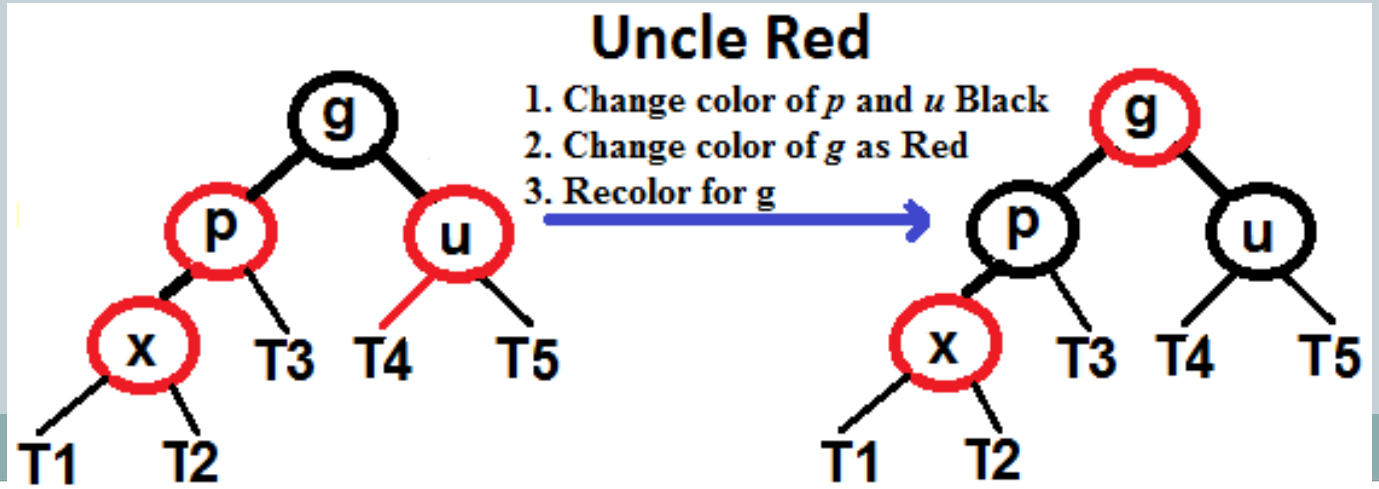
# Red-Black Insertion Procedure



Please note that color of **NULL** node is BLACK

Suppose, *x* is a new node to be inserted.

1. Perform Binary Search Tree to new inserted node, and color it RED
2. if *x* is root, change color of *x* as BLACK
3. Do following if color of *x*'s parent is RED and *x* is not root
   a. if *x*'s uncle is RED
      i. change color of parent and uncle as BLACK.
      ii. color of grand parent as RED.
      iii. Change *x* = *x*'s grandparent, repeat steps 2 and 3 for new *x*.



# Red-Black Insertion Procedure (cont.)



b. if x's uncle is BLACK

i. Left Left (*p* is left child of *g* and *x* is left child of *p*).

ii. Left Right (*p* is left child of *g* and *x* is right child of *p*).

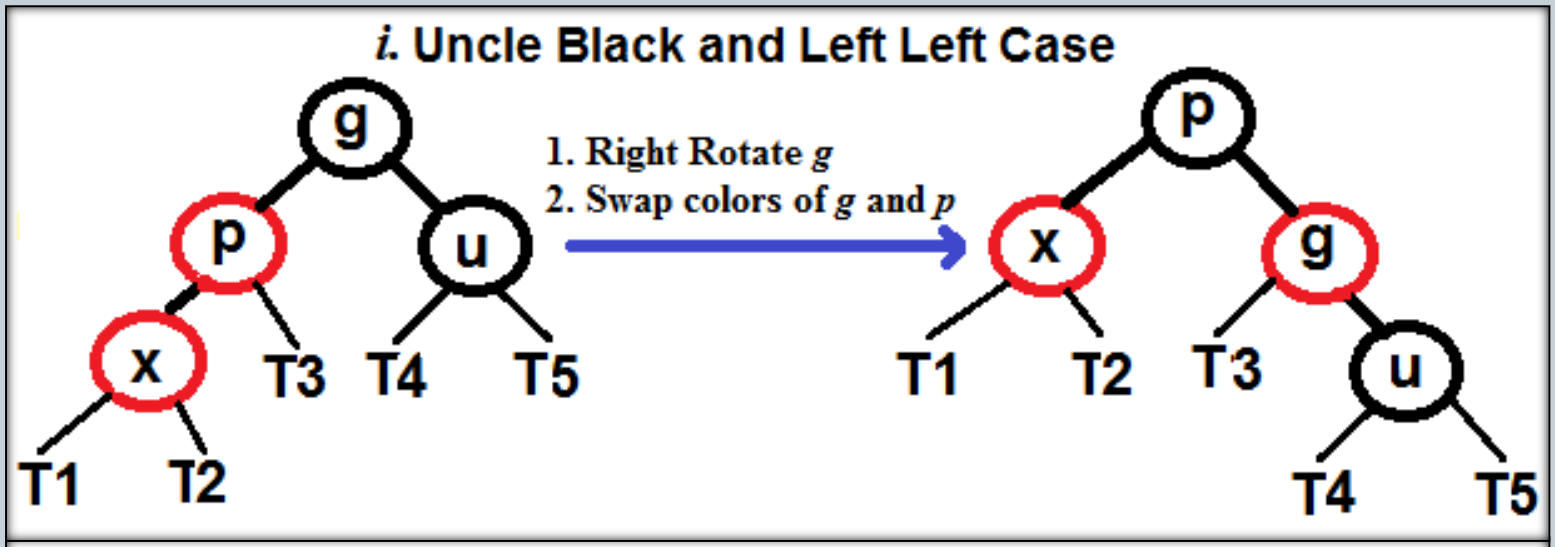
iii. Right Right (*p* is right child of *g* and *x* is right child of *p)*.

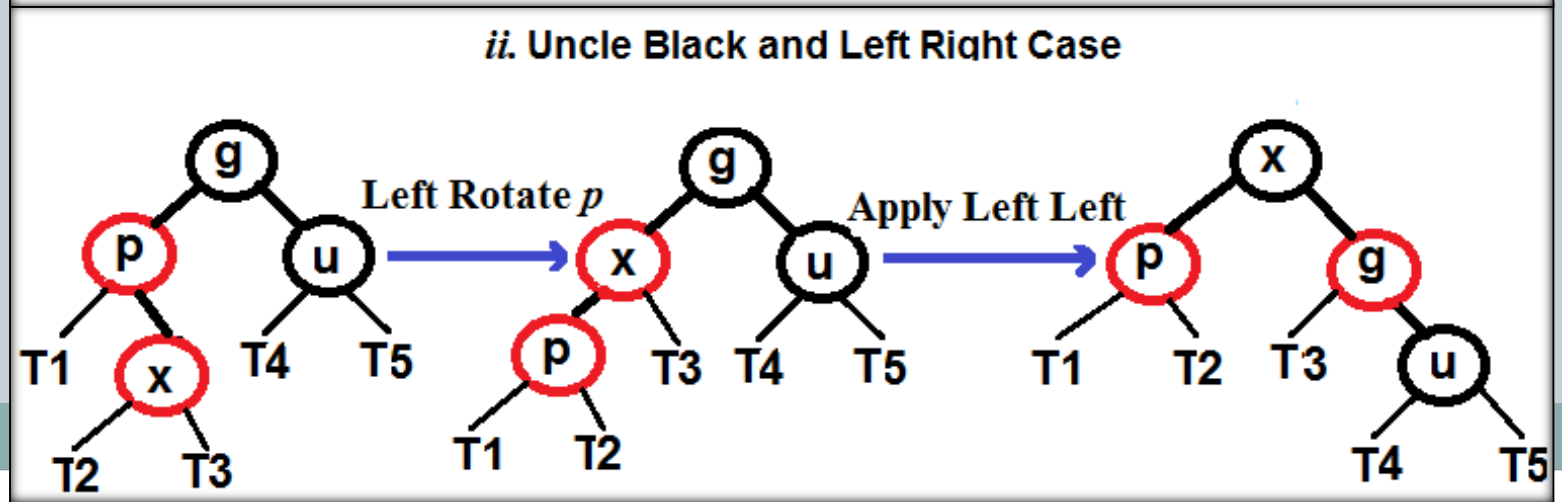
iv. Right Left (*p* is right child of *g* and *x* is left child of *p)*.

# Red-Black Insertion Procedure (cont.)



- Uncle is **BLACK,** Parent is **Left** Child of Grandparent

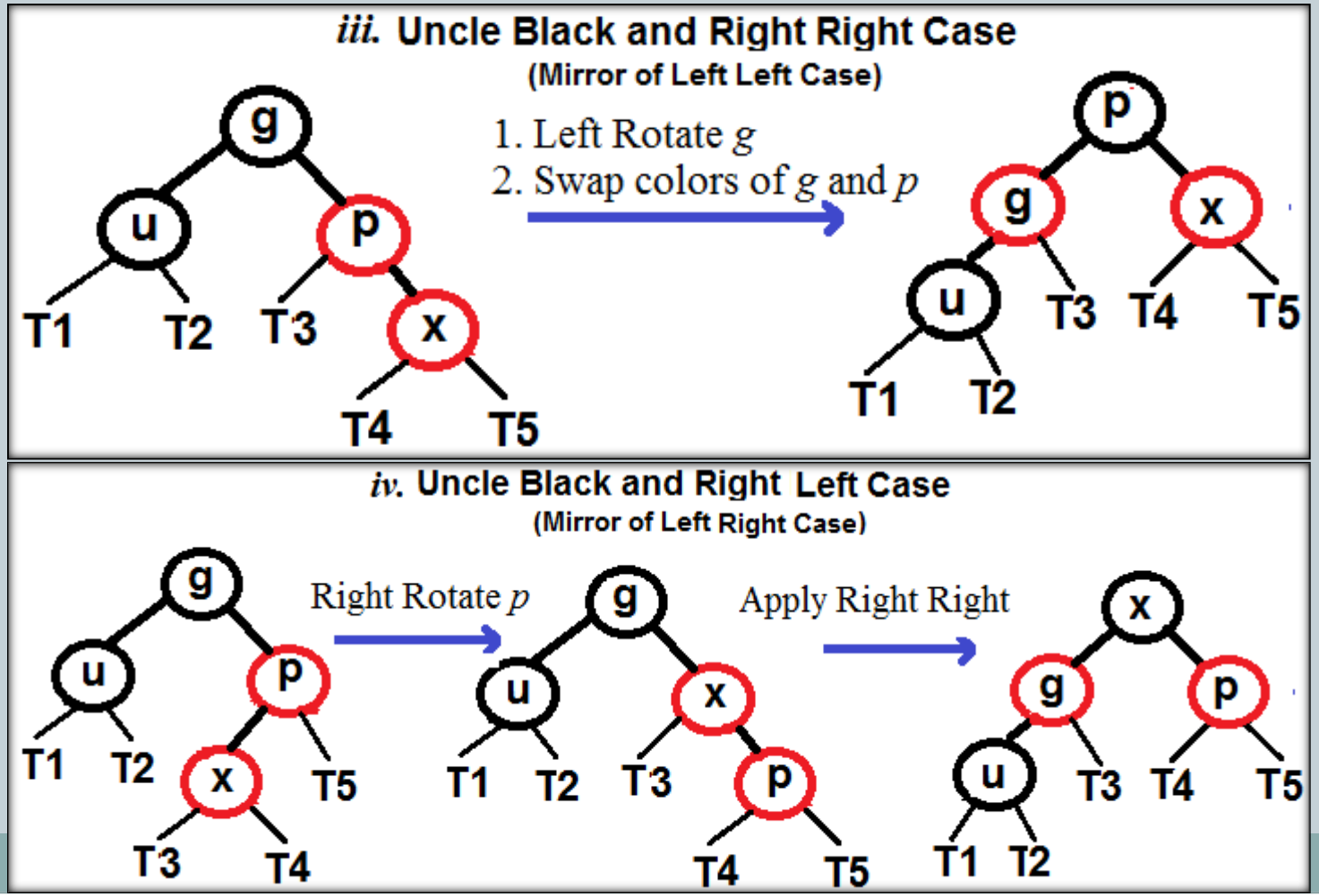


# Red-Black Insertion Procedure (cont.)



- Uncle is **BLACK,** Parent is **Right** Child of Grandparent



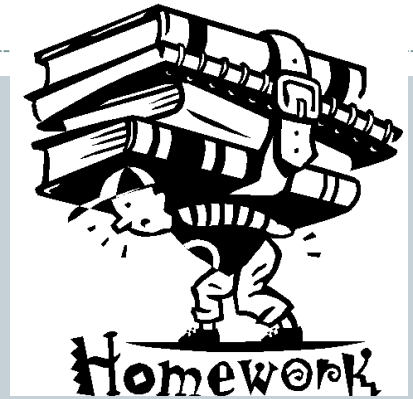
# Insert Node to RBT Pseudo Code

Binary Search Tree Insertion( *x* ):

- Search for a place to put a new node *x*.
- Insert the new node to this place.
- Fix up RB rules after insertion new node.

FixUpAfterInsertion( *x* ):

1. Set color of *x* as RED
2. If *x* is not NULL & n is not Root & Parent of *x* is RED
    2.1. Uncle of x is RED
        2.1.1. Set color of Parent and Uncle as BLACK
        2.1.2. Set color of GrandParent as RED
        2.1.3. FixUpAfterInsertion(*Grandparent of x* )
    2.2. Uncle of x is BLACK
        2.2.1. If Left Left thus
            2.2.1.1. Right Rotate GrandParent x
            2.2.1.2. Swap color Grand Parent and Parent
        2.2.2. If Left Right
            2.2.2.1. Left Rotate Parent
            2.2.2.2. Apply Left Left
        2.2.3. If Right Right
            2.2.3.1. Left Rotate Grandparent
            2.2.3.2. Swap color Grand Parent and Parent
        2.2.4. If Right Left
            2.2.4.1. Right Rotate Parent
            2.2.4.2. Apply Right Right
3. Set Color of Root as BLACK




Based on Pseudo code above, create a function for insertion new node into Red-Black Tree.

To be continued…