Unit 12 – Red-Black Tree

CS 201 - Data Structures II
Spring 2023
Habib University

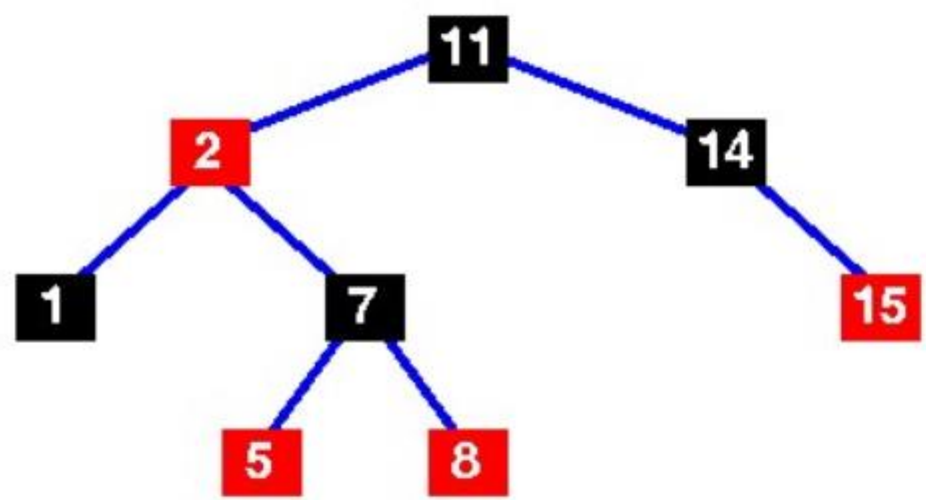
Syeda Saleha Raza

Motivation

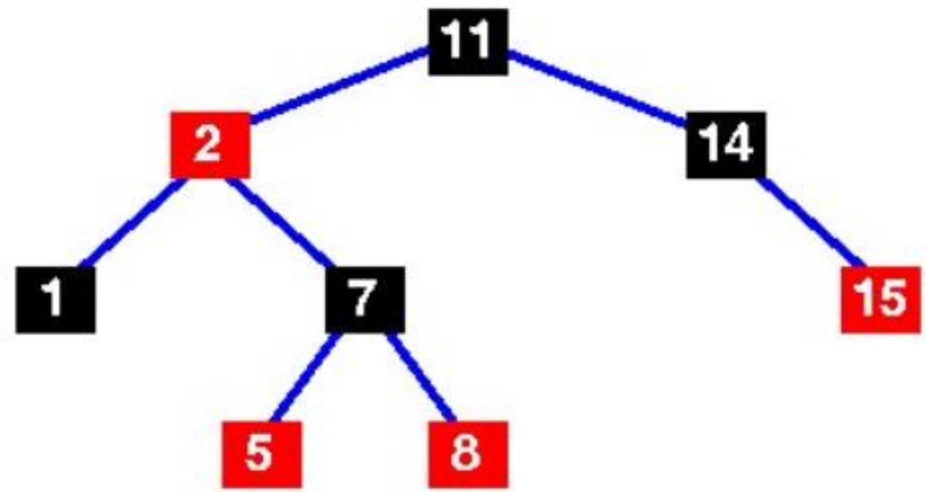
- The AVL trees are more balanced compared to Red-Black Trees, but they may cause more rotations during insertion and deletion.
- So if your application involves frequent insertions and deletions, then Red-Black trees should be preferred.
- And if the insertions and deletions are less frequent and search is a more frequent operation, then AVL tree should be preferred over Red-Black Tree.

Red-Black Tree

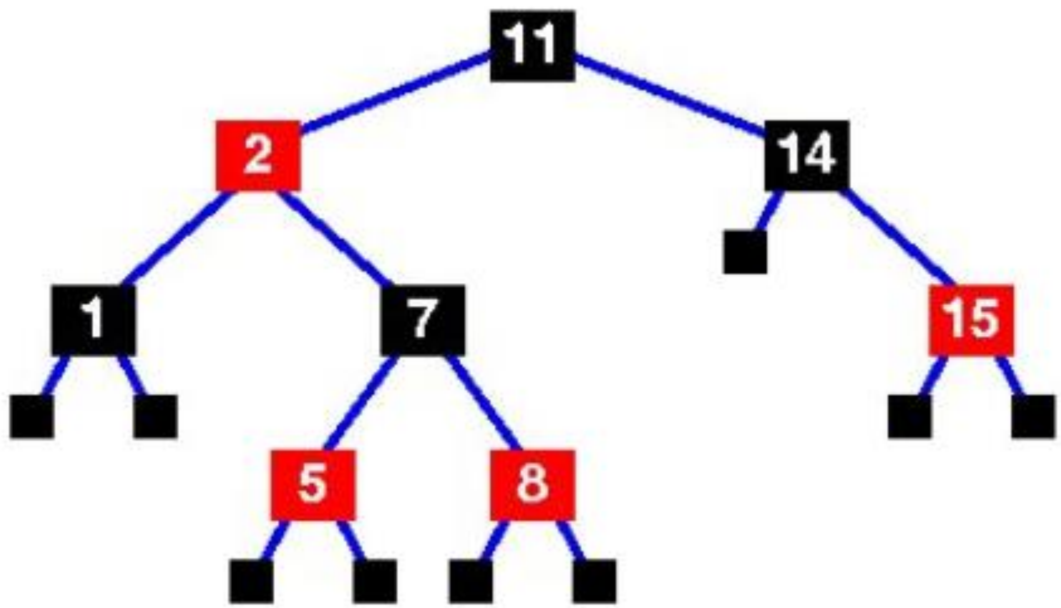
- A *red-black tree is a binary search tree with nodes* colored red and black in a way that satisfies the following properties:
  - *Root Property: The root is black.*
  - *Red Property: The children of a red node (if any) are black.*
  - *Depth Property: Every path from a node (including root) to any of its descendants NULL nodes has the same number of black nodes.*



Example



https://bloggingwithgrp27.blogspot.com/2022/01/avl-tree-vs-rb-tree-with-applications.html


11
2
14
1
7
15
5
8

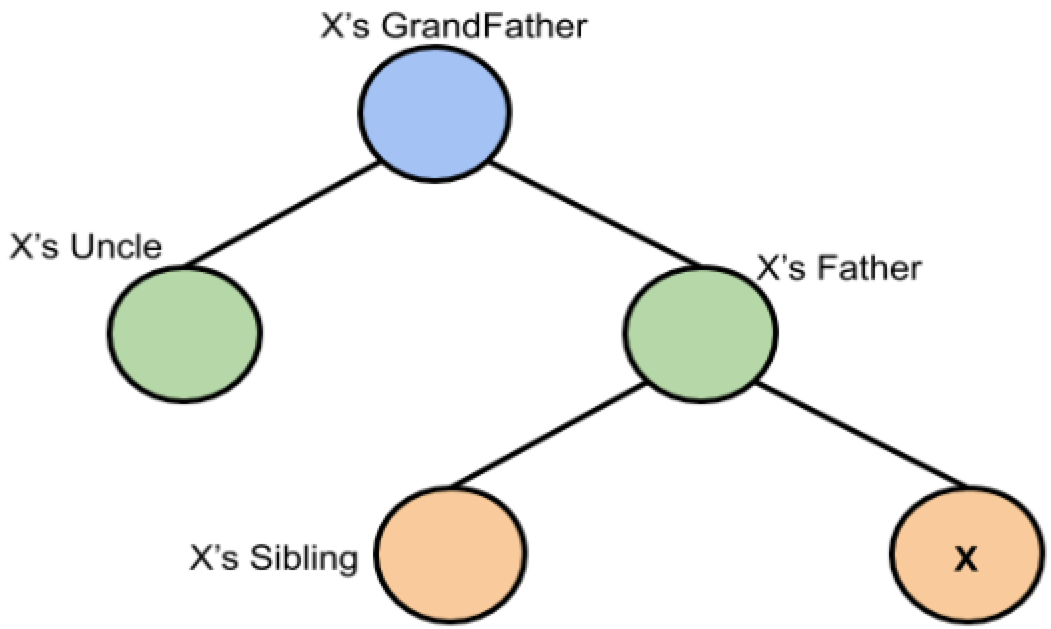
RB vs AVL Trees

- RB Trees maintain color for each node as compared to AVL trees that keep track of balance factor at each node.
- AVL trees are strictly balanced whereas RB trees are almost balanced.

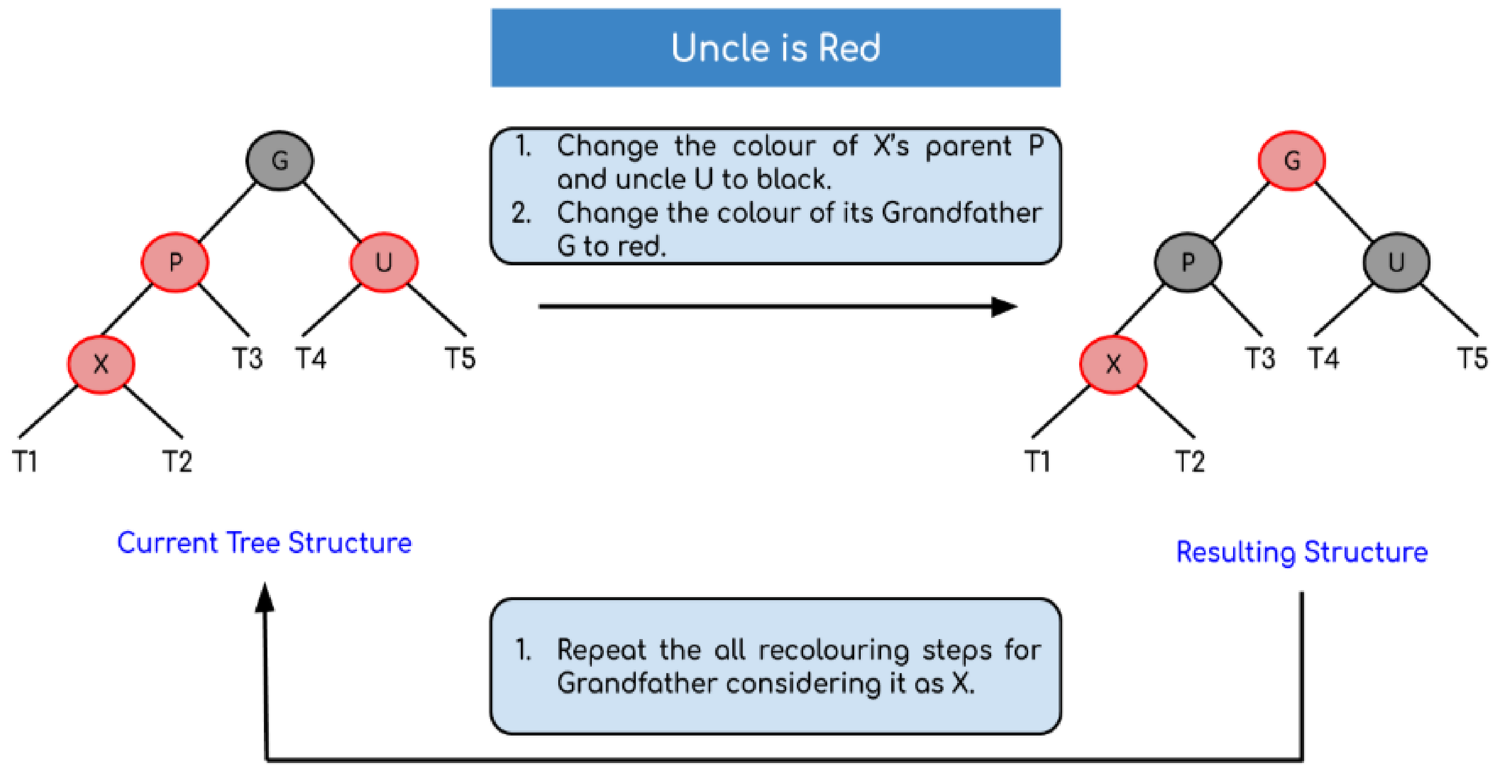
Insertion

- A new node is always inserted as a red leaf node.
- If the tree starts violating any of its property then the following operations are performed:
  – Recoloring
  – Rotation

Node's Family

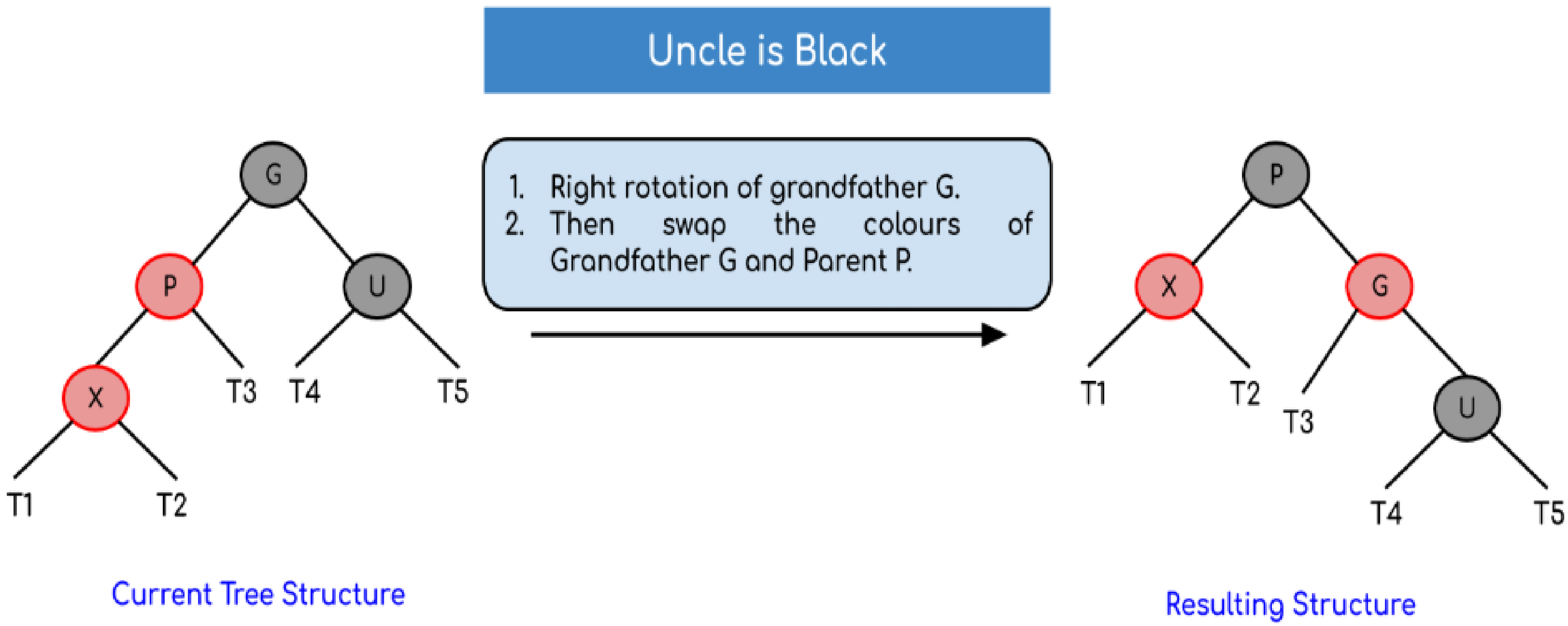


Recoloring



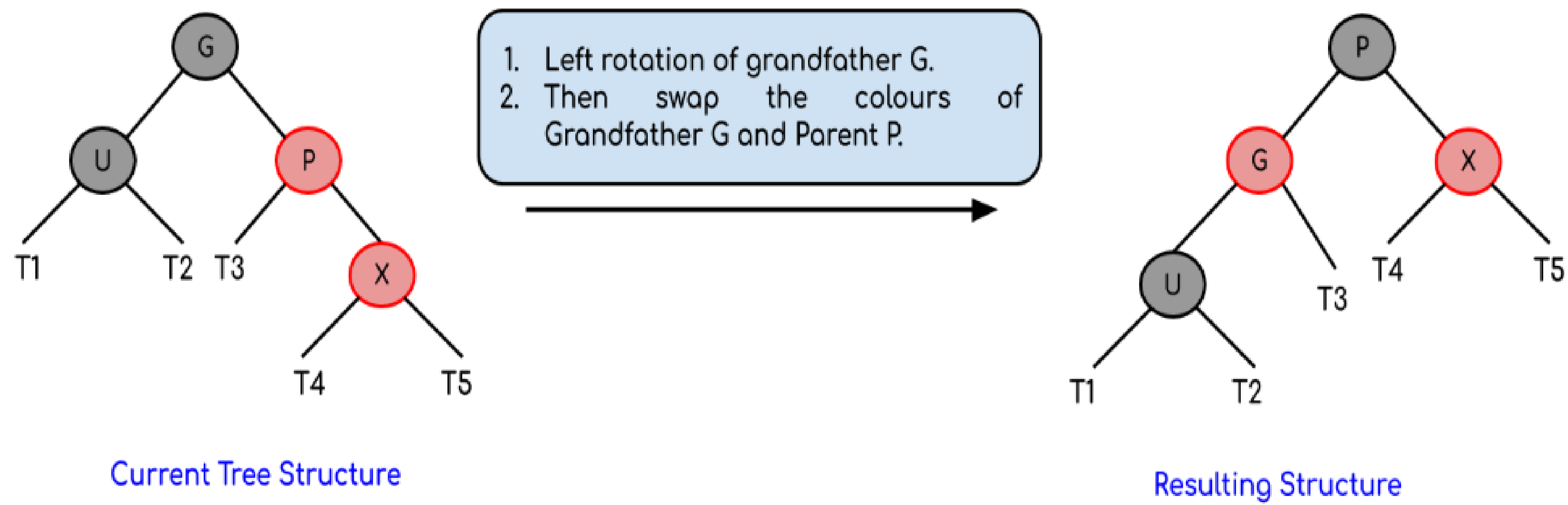
https://www.geeksforgeeks.org/red-black-tree-set-2-insert/

Rotation – Left-Left



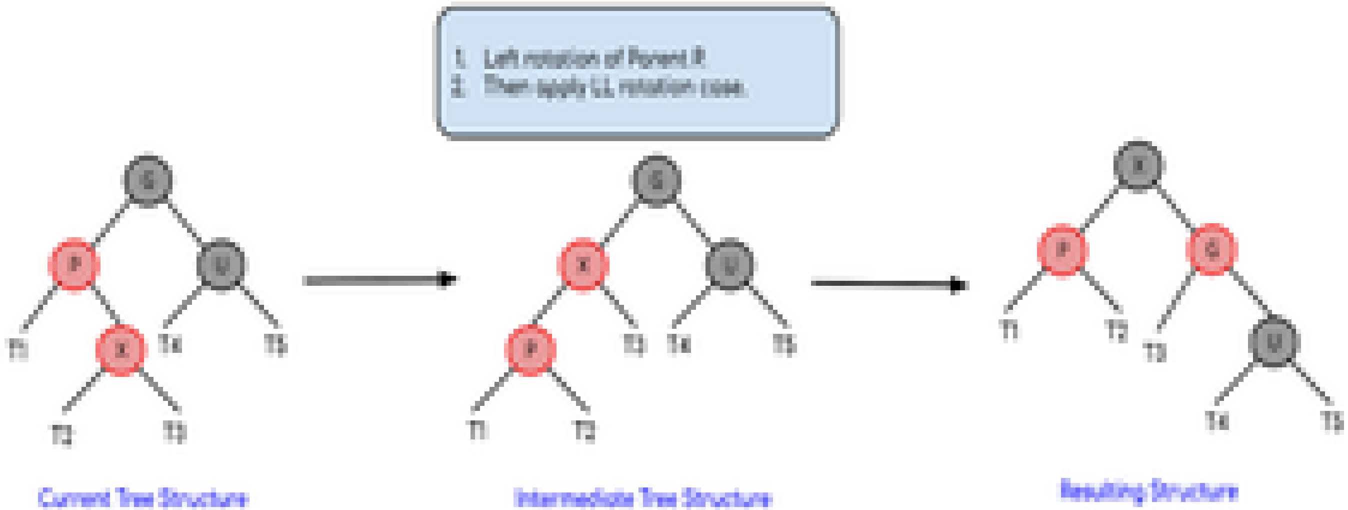
https://www.geeksforgeeks.org/red-black-tree-set-2-insert/

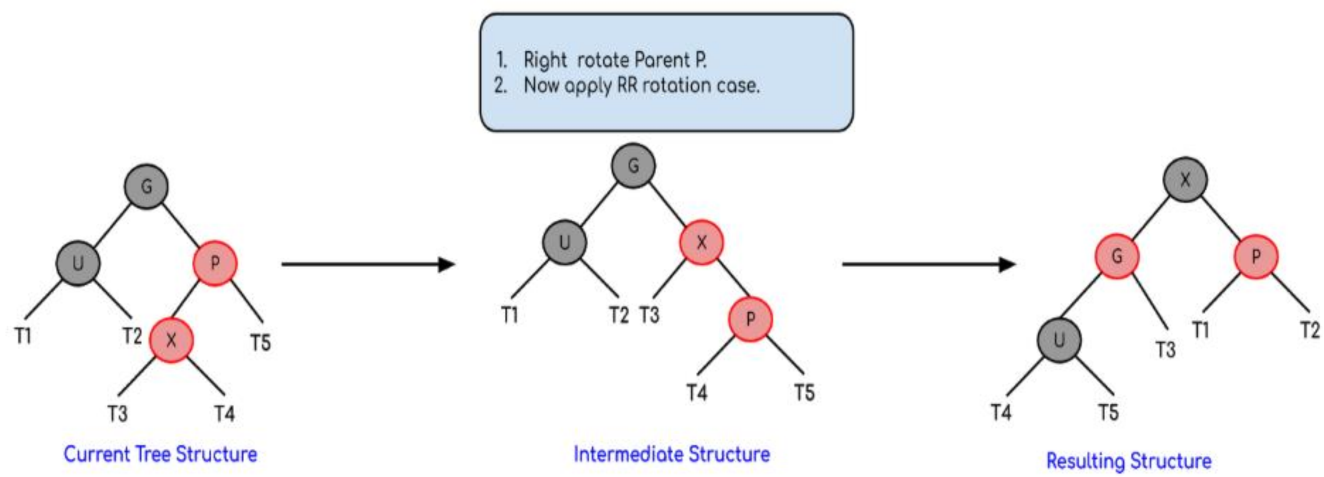
Rotation – Right Right



https://www.geeksforgeeks.org/red-black-tree-set-2-insert/

Rotation – Left/Right and Right/Left





https://www.geeksforgeeks.org/red-black-tree-set-2-insert/

Summarizing all four cases

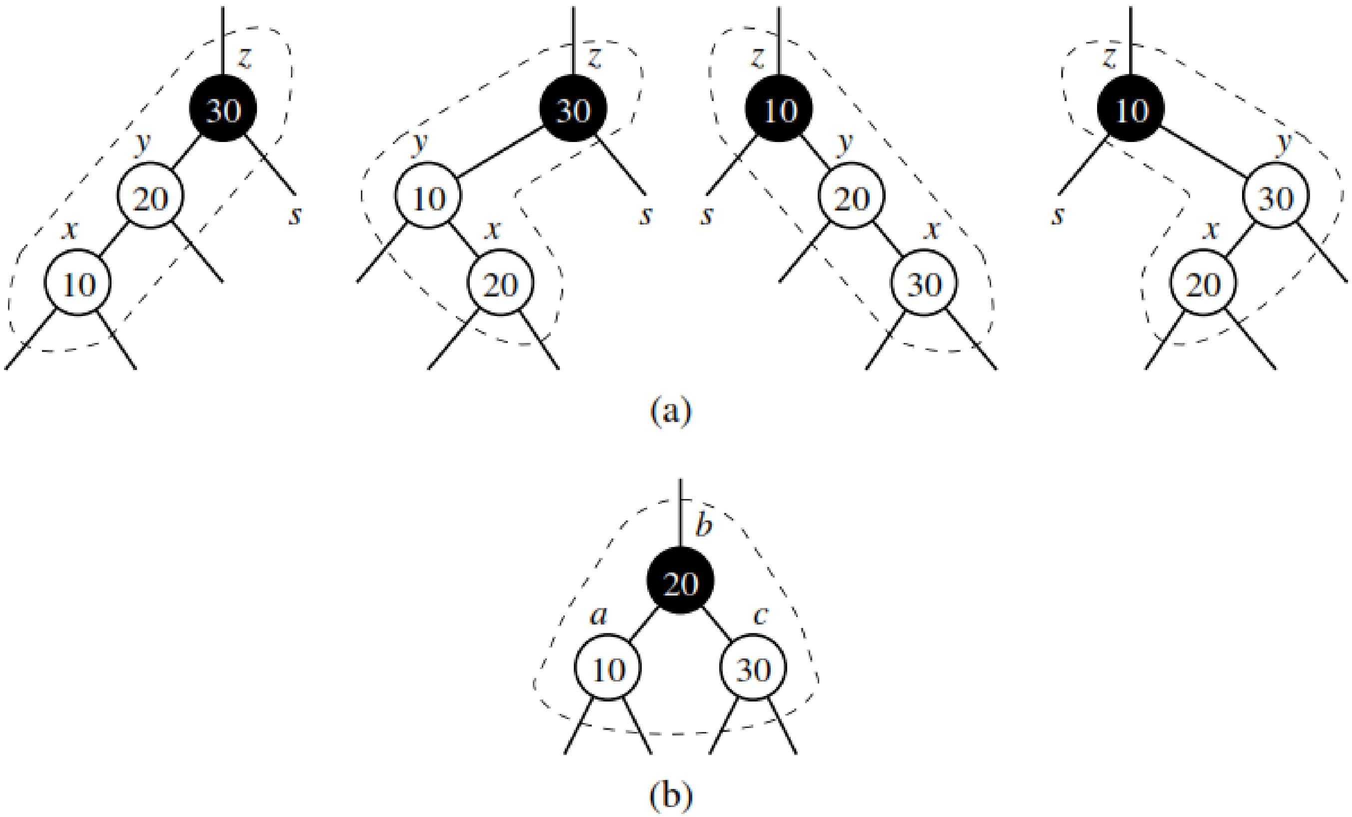
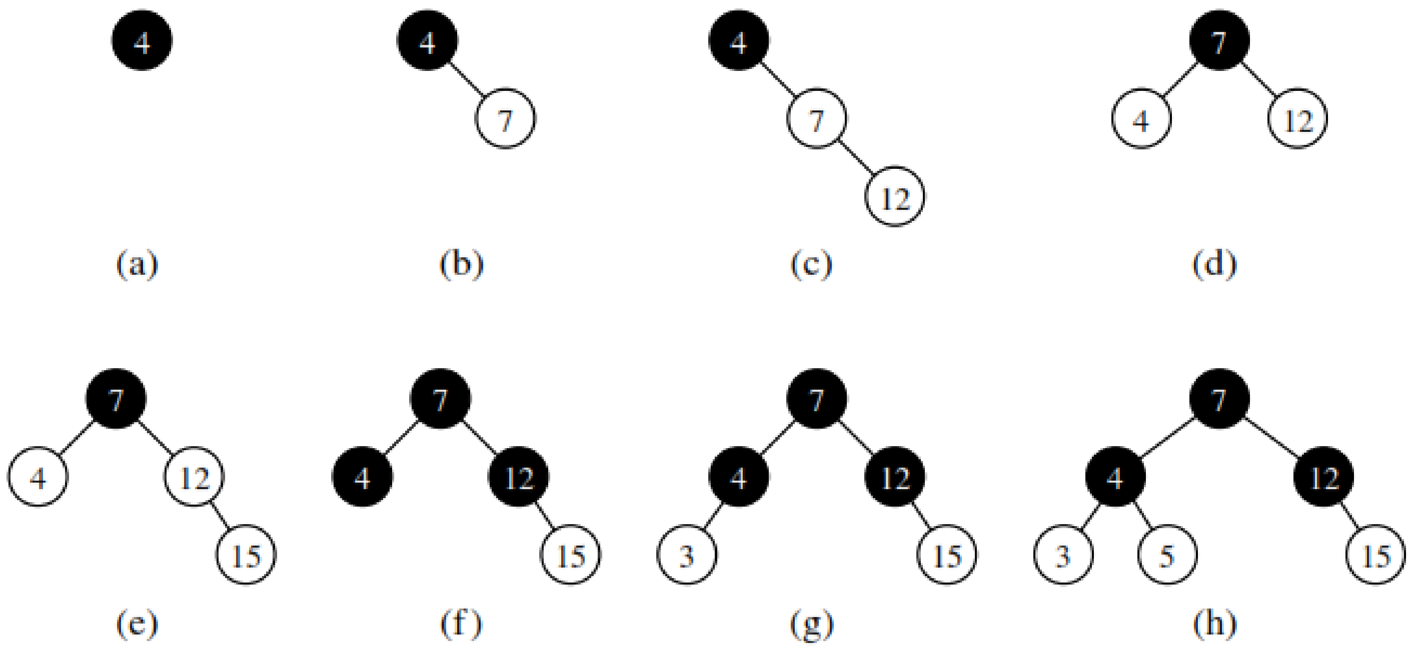


Figure 11.33: Restructuring a red-black tree to remedy a double red: (a) the four configurations for x, y, and z before restructuring; (b) after restructuring.

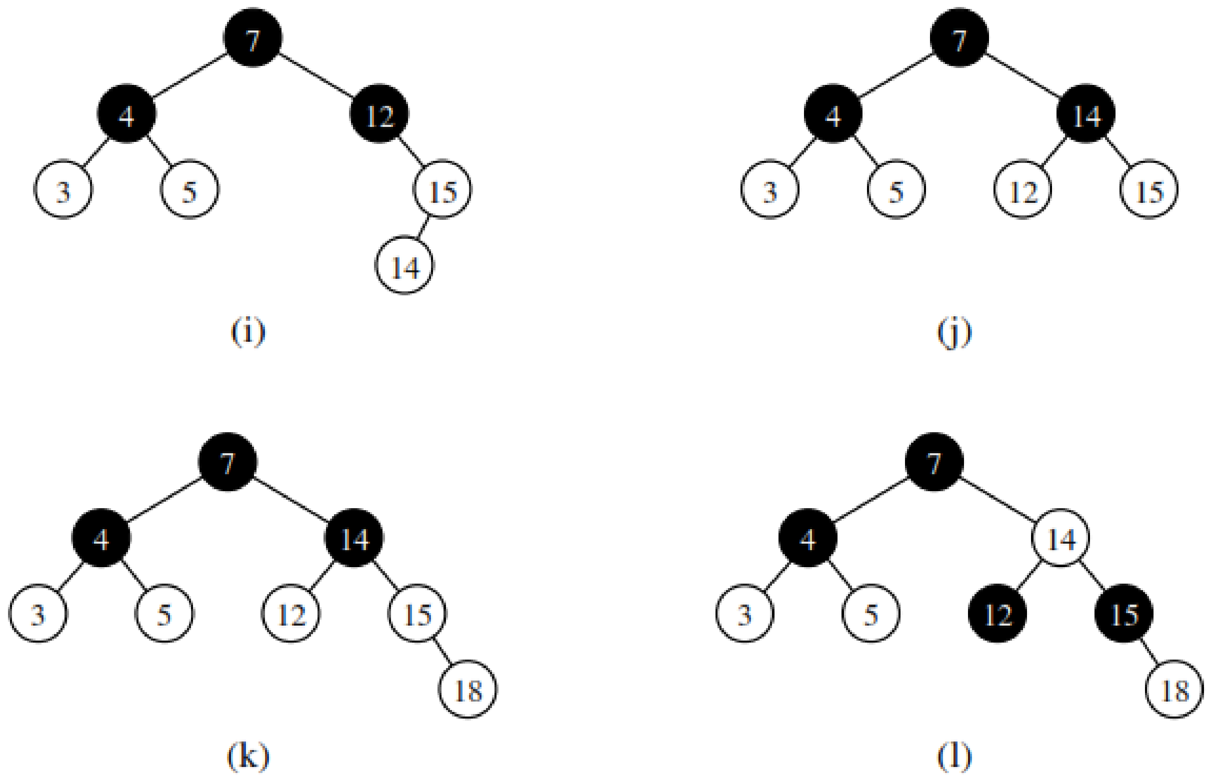
Insertion in RB Trees - Example

- Inserting the following keys:
  - 4,7,12,15,3,5,14,18

Insertion in RB Trees - Examples



Insertion in RB Trees - Examples



More Inserts

- 16, 17, 6

Exercise

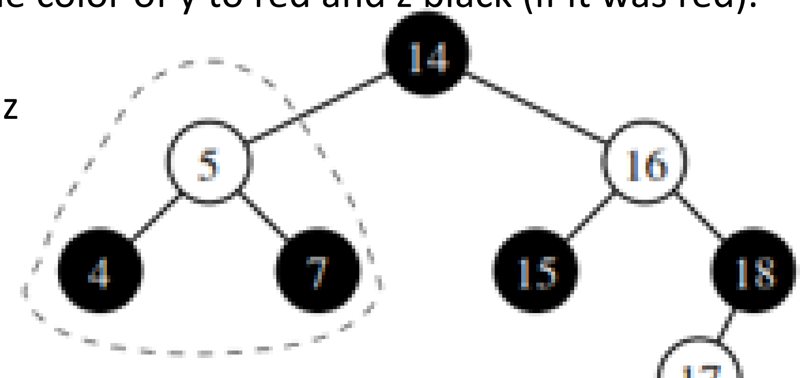
- 4,9,13,8,2,6,7,5,21

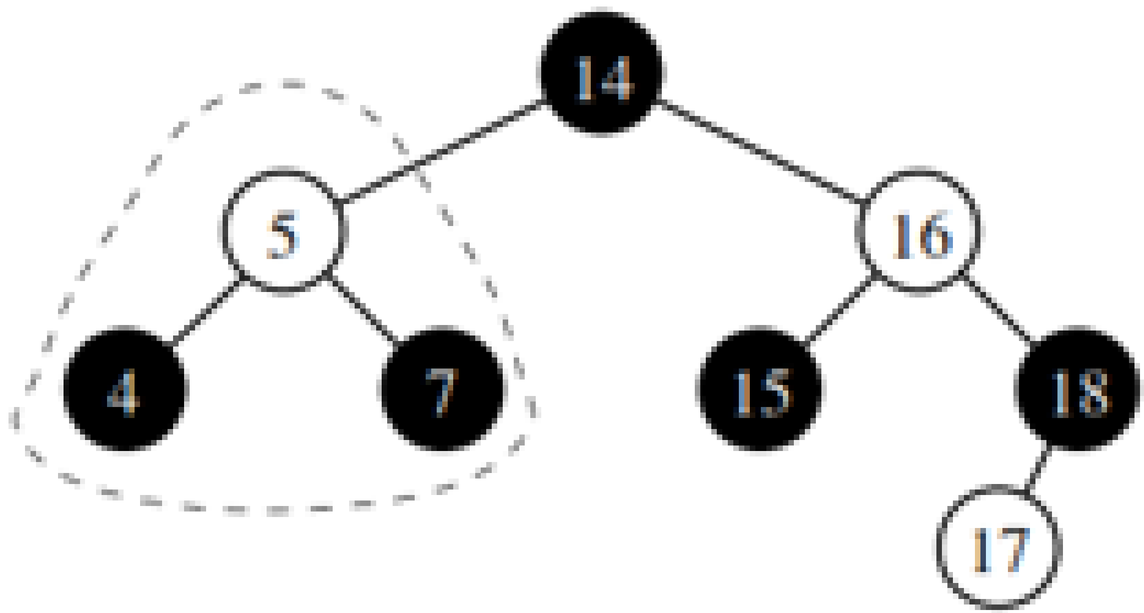
Deletion

- Perform BST Deletion
- Fix Tree
  - In the insert operation, we check the color of the uncle to decide the appropriate case. In the delete operation, ***we check the color of the sibling*** to decide the appropriate case.

RB Tree - Deletion

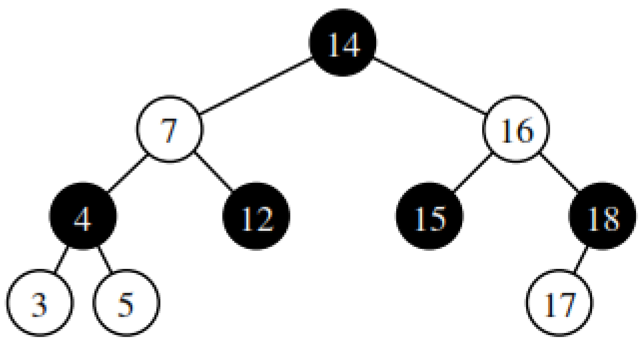
- If the node to be deleted (N) is red then delete it
- If N is black
  - Has one child (that would be red)
    - Promote child and delete N
  - Has no child
    - Let z be parent of N and y be its sibling
    - Case 1: If y is black and has a red child x
      - Perform trinode structuring
      - Color a and c black and x be the former color of z.
      - (equivalent to borrowing a key from sibling via rotation in 2-4 tree)
    - Case 2: If y is black with both children black (or None)
      - Perform recoloring (change the color of y to red and z black (if it was red).
    - Case 3: if y is red
      - perform rotation about y and z
      - Color y black and z red.



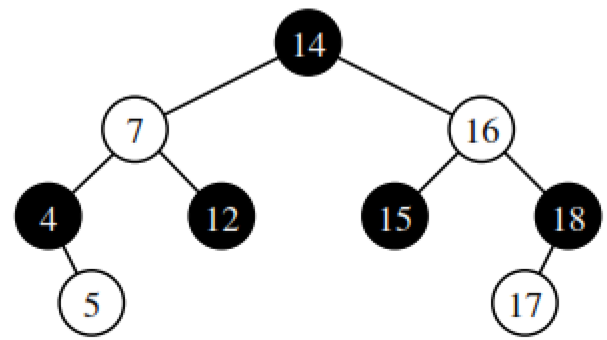

14
5
16
4
7
15
18
17

Sequence of Deletion

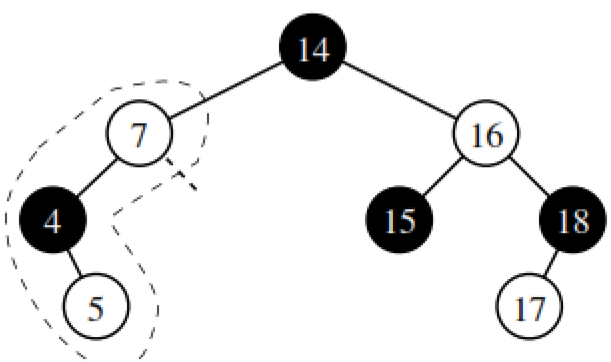
- Deleting 3,12,17,18,15, 16



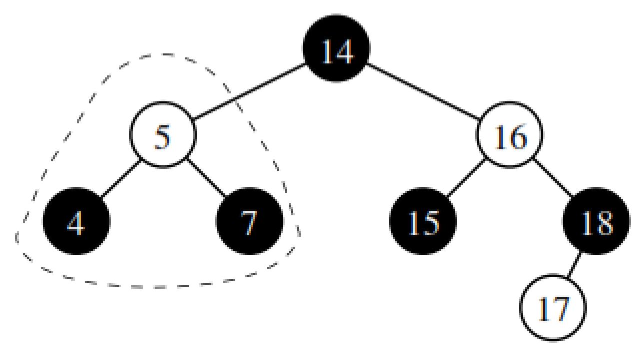
(a)



(b)



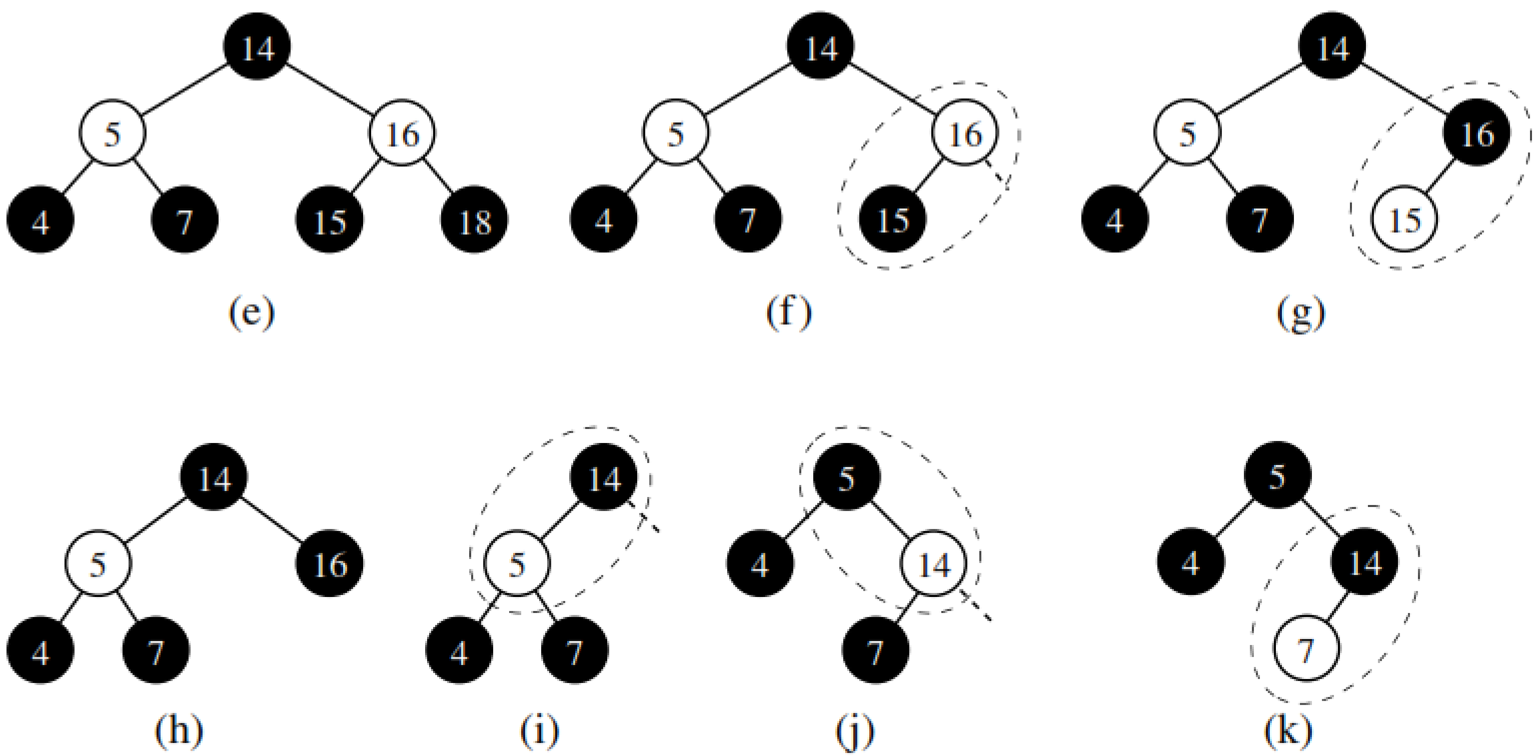
(c)



(d)

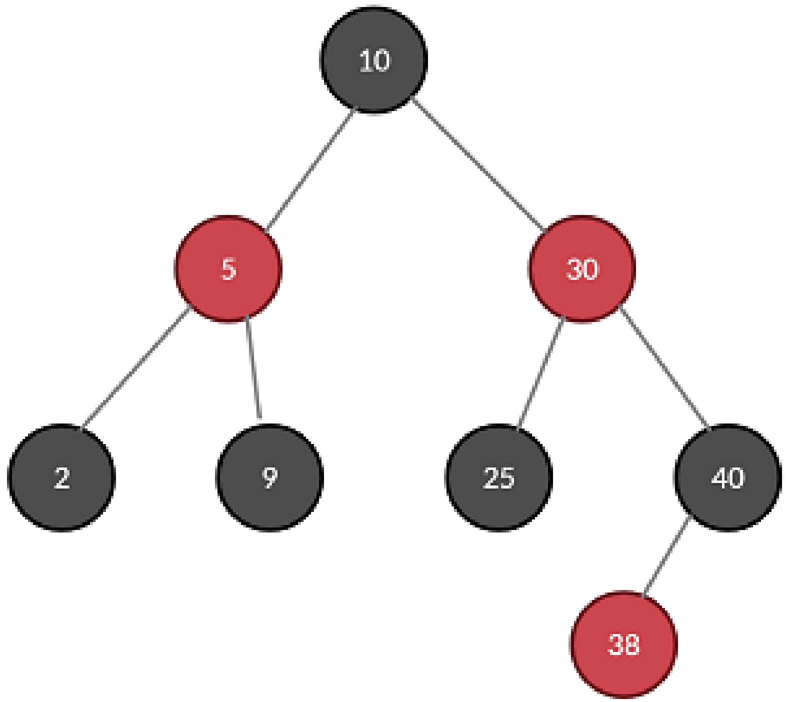
Sequence of Deletion

- Deleting 3,12,17,18,15, 16

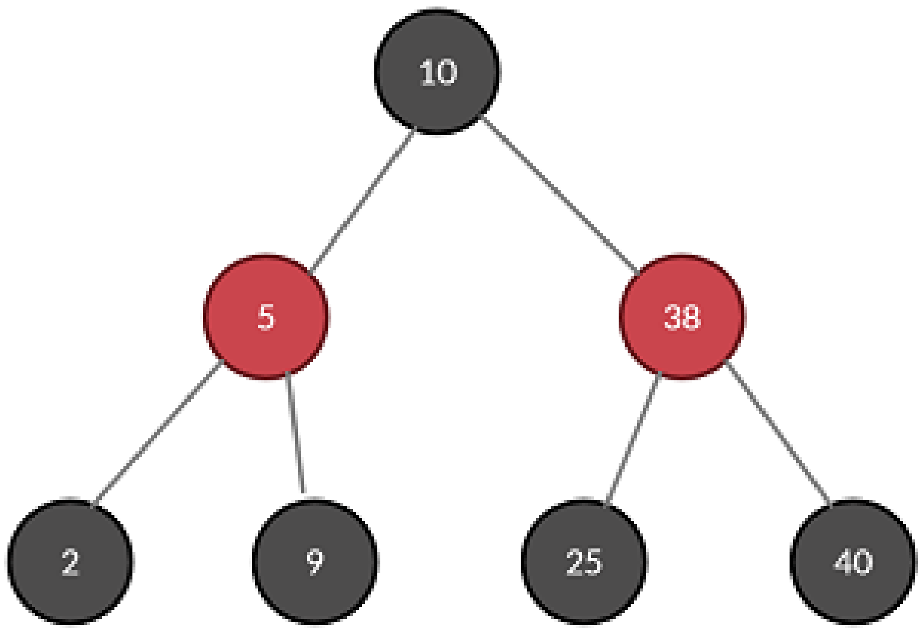


Example I

- Delete 30

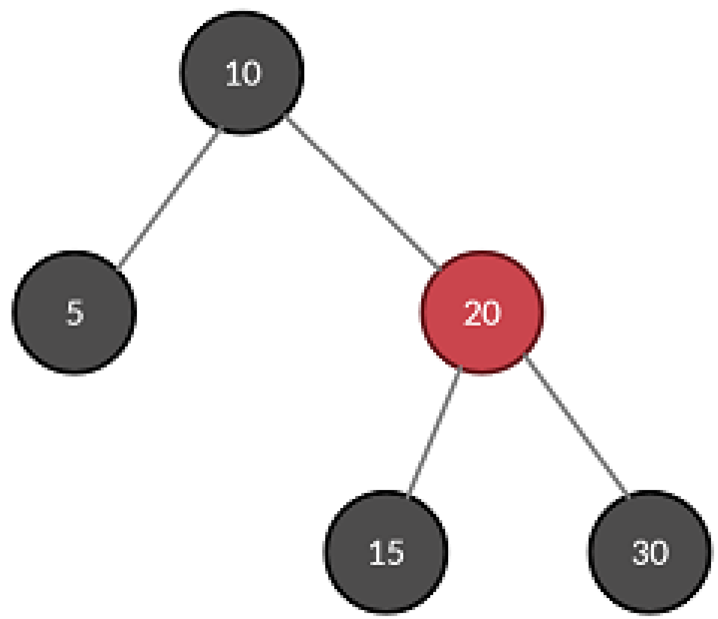


Example I – 30 deleted

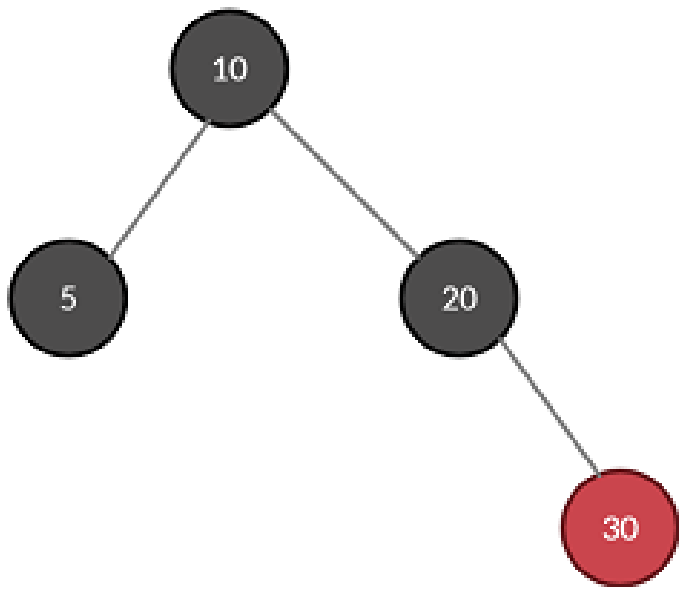


Example II

- Delete 15.



Example II – 15 deleted



More Examples

- https://medium.com/analytics-vidhya/deletion-in-red-black-rb-tree-92301e1474ea

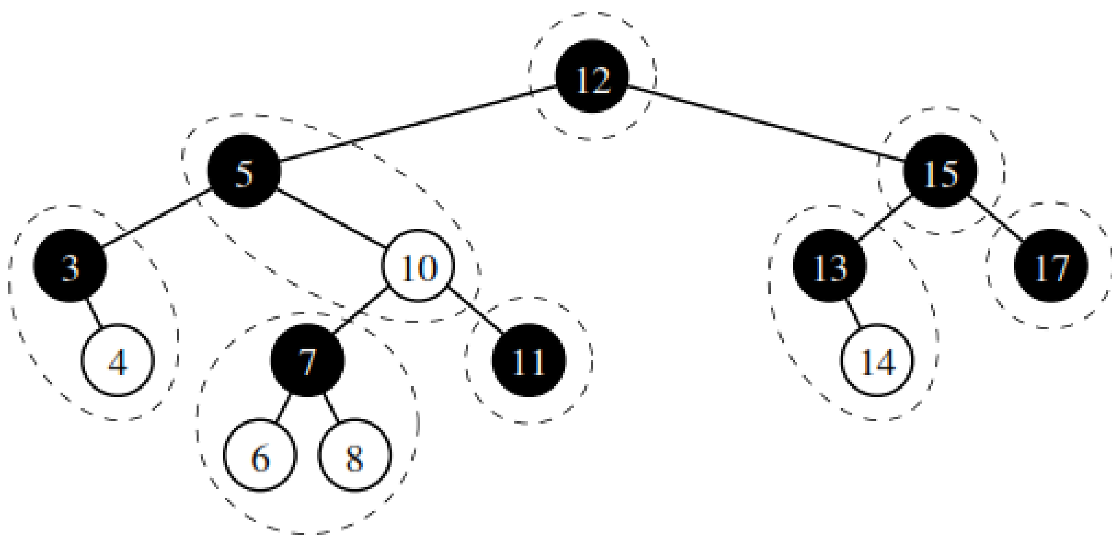
Complexity of find, insertion and deletion

Complexity

Proposition 11.10: *The insertion of an item in a red-black tree storing n items can be done in $O(\log n)$ time and requires $O(\log n)$ recolorings and at most one trinode restructuring.*

Correspondence between Red-Black and 2-4 Tree

- Ga red-black tree, we can construct a corresponding (2, 4) tree by merging every red node *w into its parent, storing the entry from w at* its parent, and with the children of *w becoming ordered children of the parent.*



Resources

- Data Structures and Algorithms in Python, by Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser. 2013. (1st. ed.). Wiley Publishing
- Open Data Structures (pseudocode edition), by Pat Morin. Available online at http://opendatastructures.org
- https://www.geeksforgeeks.org/red-black-tree-set-2-insert/

Thanks