CIS 263 Introduction to Data Structures and Algorithms

Red Black Tree (RBT)

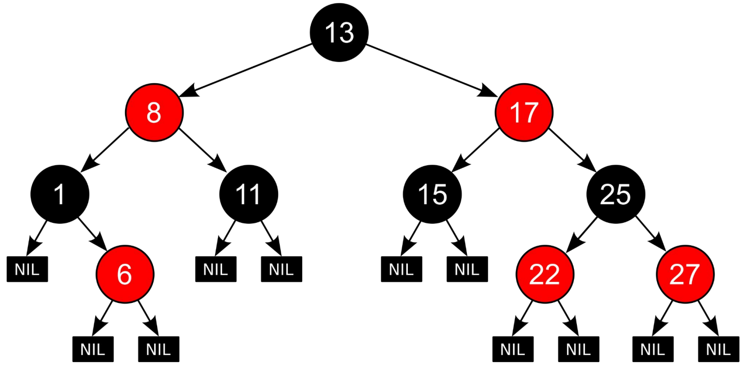
Red Black Tree (RBT)

Red Black Tree

- Red-Black tree is a BST in which every node is colored with either Red or Black.
- It is a type of self balancing BST.
- It has a good efficient worst case running time complexity.

https://www.geeksforgeeks.org/introduction-to-red-black-tree/

With explicit NIL leaves



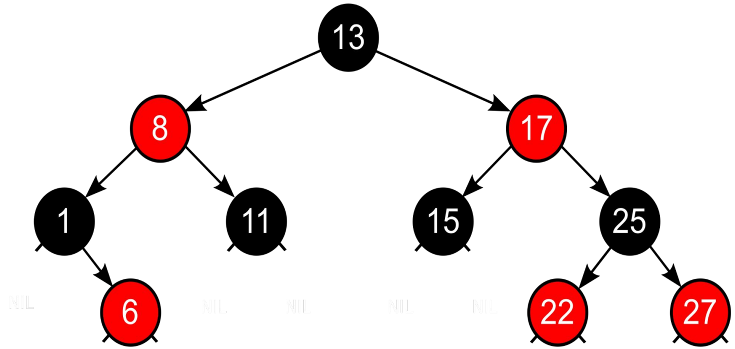
https://en.wikipedia.org/wiki/Red%E2%80%93black_tree

Red Black Tree (RBT)

Red Black Tree

- Red-Black tree is a BST in which every node is colored with either Red or Black.
- It is a type of self balancing BST.
- It has a good efficient worst case running time complexity.

With implicit left and right docking points



https://www.geeksforgeeks.org/introduction-to-red-black-tree/

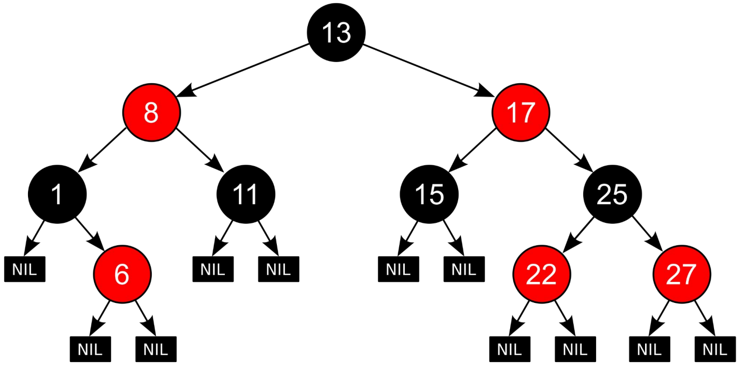
https://en.wikipedia.org/wiki/Red%E2%80%93black_tree

Red Black Tree (RBT)

Properties of Red Black Tree:

The Red-Black tree satisfies all the properties of binary search tree in addition to that it satisfies following additional properties –

1. **Root property:** The root is black.
2. **External property:** Every leaf (Leaf is a NULL child of a node) is black in Red-Black tree.
3. **Internal property:** The children of a Red node are Black. Hence possible parent of a red node is a black node.



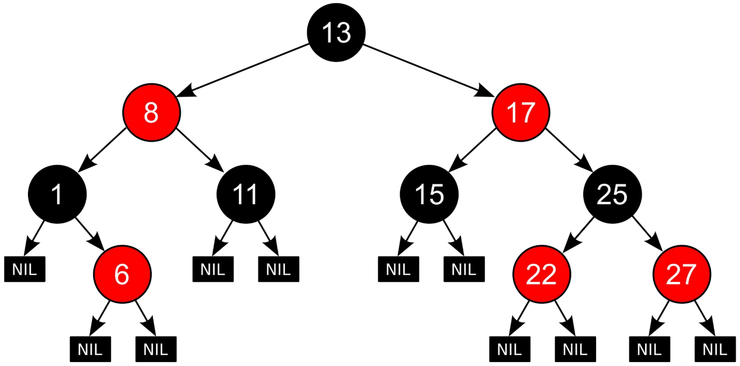
https://www.geeksforgeeks.org/introduction-to-red-black-tree/

Red Black Tree (RBT)

Properties of Red Black Tree:

The Red-Black tree satisfies all the properties of binary search tree in addition to that it satisfies following additional properties

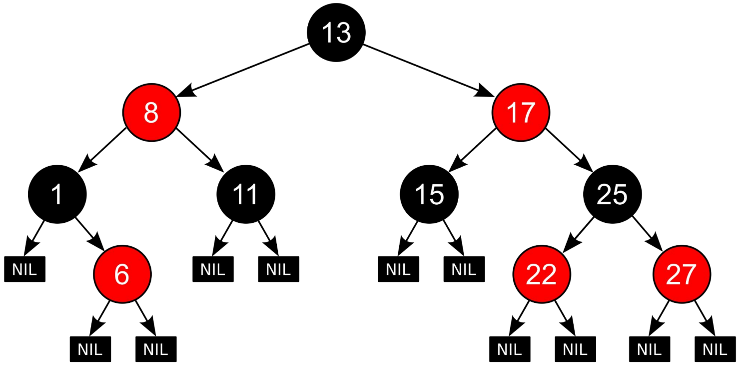
4. **Depth property:** All the leaves have the same Black depth.
5. **Path property:** Every simple path from root to descendant leaf node contains same number of Black nodes.



Red Black Tree (RBT)

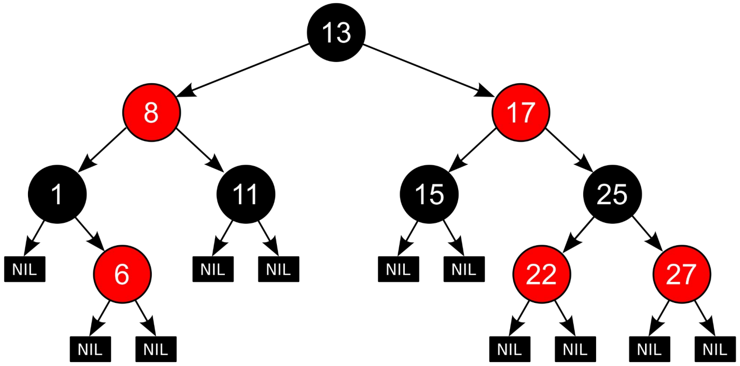
Rules That Every Red-Black Tree Follows:

1. Every node has a color either Red or Black.
2. The Root of the tree is always black.
3. There are no two adjacent Red nodes (A red node cannot have a red parent or red child).
4. Every path from a node (including root) to any of its descendants NULL nodes has the same number of black nodes.
5. Every leaf (e.i. NULL node) must be colored BLACK.



RBT - Traversal & Deletion

- RBT is a BST
- So Traversals {Pre Order, In Order, Post Order} still the same
- Deletion will also work



RBT - Insertion

Balancing is implicit (If a tree is RBT it is already balanced.

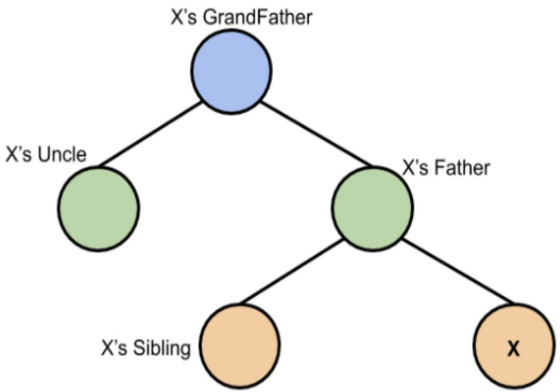
- Recoloring
- Rotation, and recoloring if required.

Try Recoloring first

- Look at the Uncle node
- If uncle node is **Red**
  - We perform **Recoloring**
- If uncle node is Black
  - We perform
    - Rotation, and Recoloring if required.

RBT - Insertion

- X is our target key (node)
- Let's define the relationships



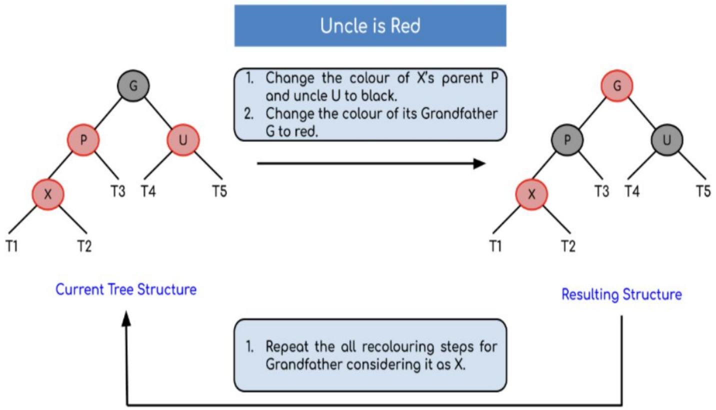
This representation is based on X

RBT - Insertion

- Uncle node is Red
- Change the color of X’s Parent(P) and Uncle(U) to be Black
- Change the color of its Grandfather G to Red

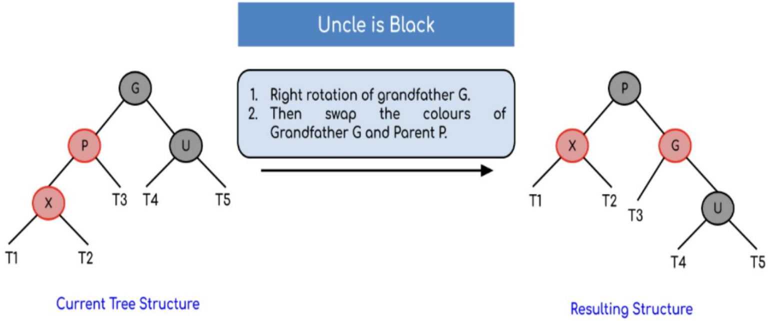
—----

- Repeat the process for G (upwords, as a child) until G is the root, then make its color Black.
- If G is root and its color is Black, keep it as it is.



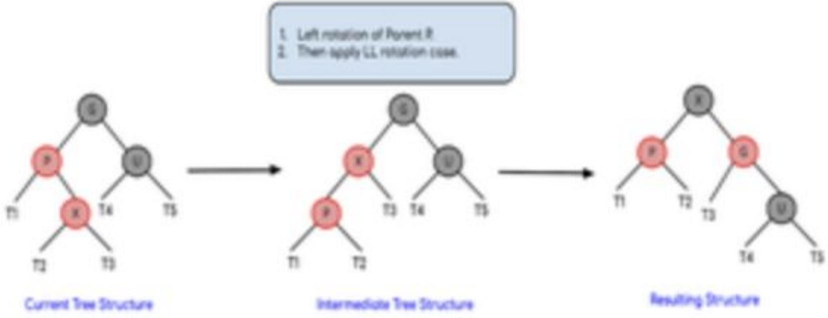
RBT - Insertion

- Uncle node is Black
- We have 4 possible cases
-
- LL rotation
  - Right rotation of grandfather G
  - Then swap the colors of G and parent P



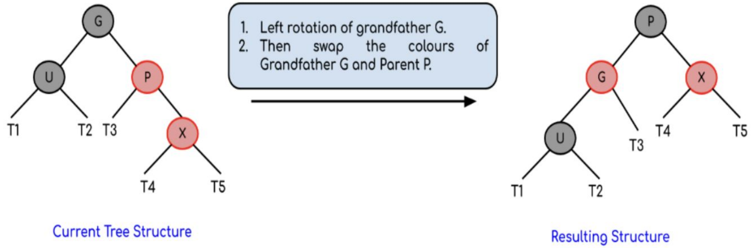
RBT - Insertion

- Uncle node is Black
- We have 4 possible cases
-
- LR rotation
    - Left rotation of parent P
    - Then LL rotation (earlier slide)



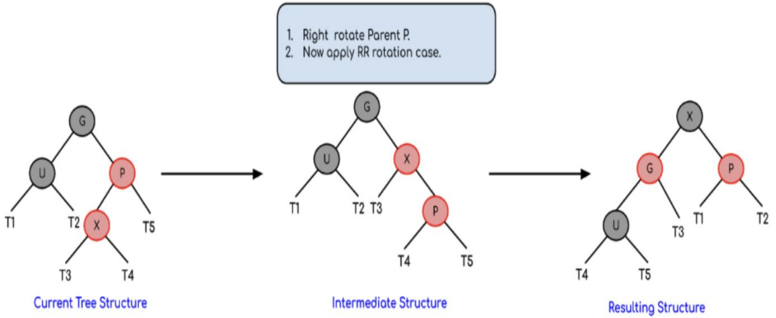
RBT - Insertion

- Uncle node is Black
- We have 4 possible cases
-
- RR rotation
    - Left rotation of grandfather G
    - Then swap the colors of G and parent P



RBT - Insertion

- Uncle node is Black
- We have 4 possible cases
-
- RL rotation
  - Right rotate parent P
  - Apply RR rotation (earlier slide)



Steps at a glance

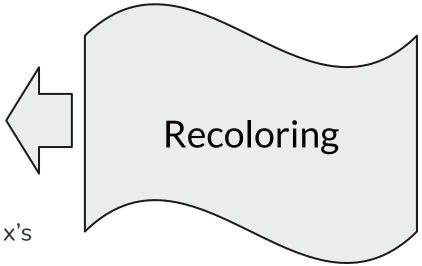
1. Perform standard BST insertion and make the colour of newly inserted nodes as RED.
2. If x is the root, change the colour of x as BLACK (Black height of complete tree increases by 1).
3. Do the following if the color of x's parent is not BLACK **and** x is not the root.

   a) If x's uncle is RED (Grandparent must have been black from property 4)

   (i) Change the colour of parent and uncle as BLACK.

   (ii) Colour 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)

   (ii) Left Right Case (p is left child of g and x is the right child of p)

   (iii) Right Right Case (Mirror of case i)

   (iv) Right Left Case (Mirror of case ii)

Steps at a glance

1. Perform standard BST insertion and make the colour of newly inserted nodes as RED.
2. If x is the root, change the colour of x as BLACK (Black height of complete tree increases by 1).
3. Do the following if the color of x's parent is not BLACK **and** x is not the root.

   a) If x's uncle is RED (Grandparent must have been black from property 4)

   (i) Change the colour of parent and uncle as BLACK.

   (ii) Colour 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)

   (ii) Left Right Case (p is left child of g and x is the right child of p)

   (iii) Right Right Case (Mirror of case i)

   (iv) Right Left Case (Mirror of case ii)

Re-coloring after rotation:

- For LL Case [3.b (i)] and RR case [3.b (iii)], swap colors of grandparent and parent after rotations
- For LR Case [3.b (ii)]and RL Case [3.b (iv)], swap colors of grandparent and inserted node after rotations

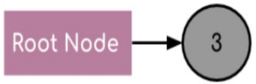
Re-coloring after rotation

Re-coloring after rotation:

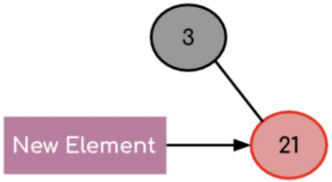
- For LL Case [3.b (i)] and RR case [3.b (iii)], swap colors of grandparent and parent after rotations
- For LR Case [3.b (ii)]and RL Case [3.b (iv)], swap colors of grandparent and inserted node after rotations

Example case

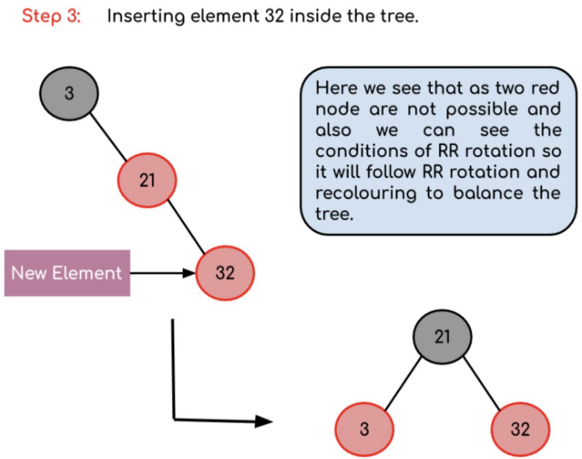
Step 1: Inserting element 3 inside the tree.



Step 2: Inserting element 21 inside the tree.

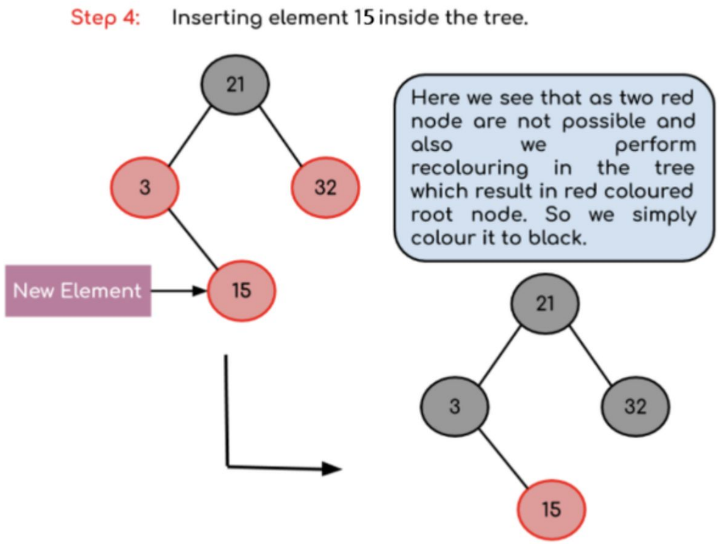


Example case



- RR rotation and recoloring turned it to be a RBT

Example case



- Recoloring operation turned it to be a RBT