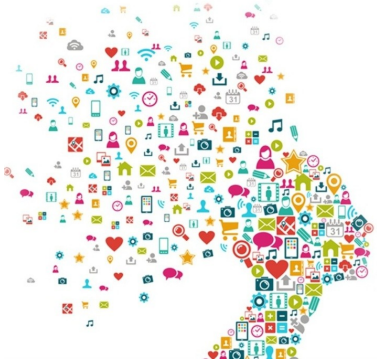


GLOBAL EDGE
Intelligence Of Things®

Red Black Tree

Sachu George

Contents

- What are Red Black trees
- RB Tree structure
- Terms used
- Rules/Properties
- Balancing in RB tree
- Tree Rotations
- Operations: Insert, Search, Delete

What are Red Black trees

- Self balancing binary search tree.
  - Invented in 1972 by Rudolf Bayer
- Each node has an extra attribute color(Red/Black).
- Color bit is used for balancing of the tree.
- Explained as extended binary search tree.
- In extended, NULL links are replaced by special nodes.

RB Tree Structure

```
typedef struct node
{
    struct node *parent;
    int data;
    char color;
    struct node *left;
    struct node *right;
}rbt;
```

Terms used

- Node - Each element of tree.
- Parent – Immediate predecessor of node.
- Grand parent – Parent’s parent node.
- Sibling – Other child of parent.
- Uncle – Sibling of parent node.
- Nephew – Children of sibling node.

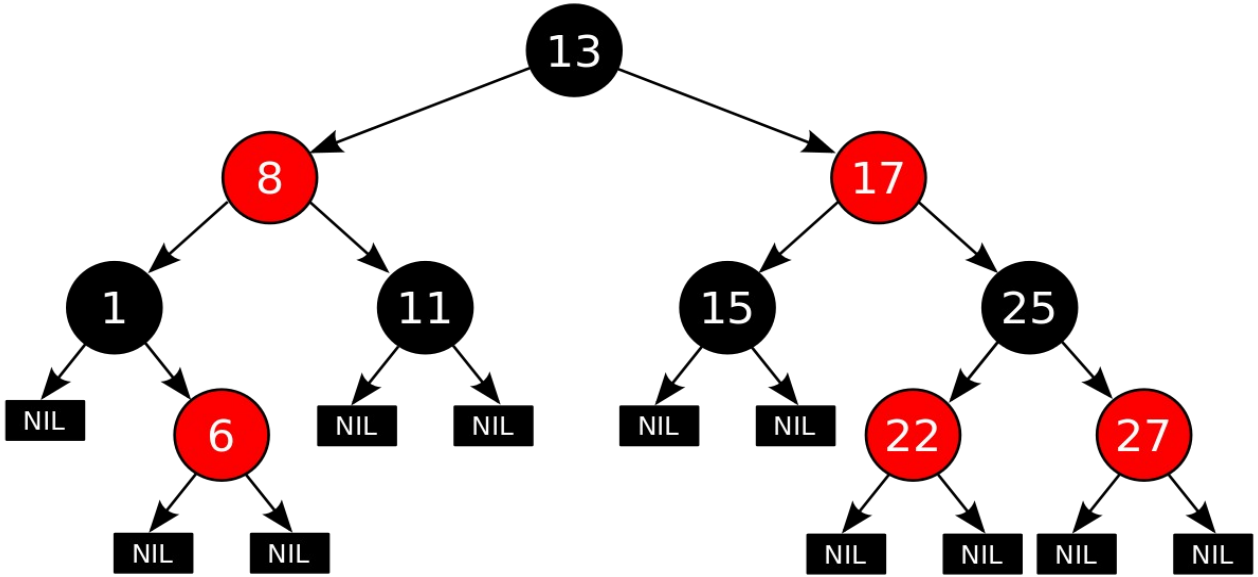
RB Tree Rules/Properties

- Root is always black.
- All extended nodes(NULL) are black.
- A red node cannot have red children, it can have only black children; A black node can have black or red children.
- For each node N, all paths from node N to external nodes contain the same number of black nodes (same black height).

Black height

- Number of black nodes on a path from a node to its leaf.
- While calculating black height of a node:
  - Leaf nodes are also counted.
  - Current node is not counted.
- Black height of tree = Black height of root.

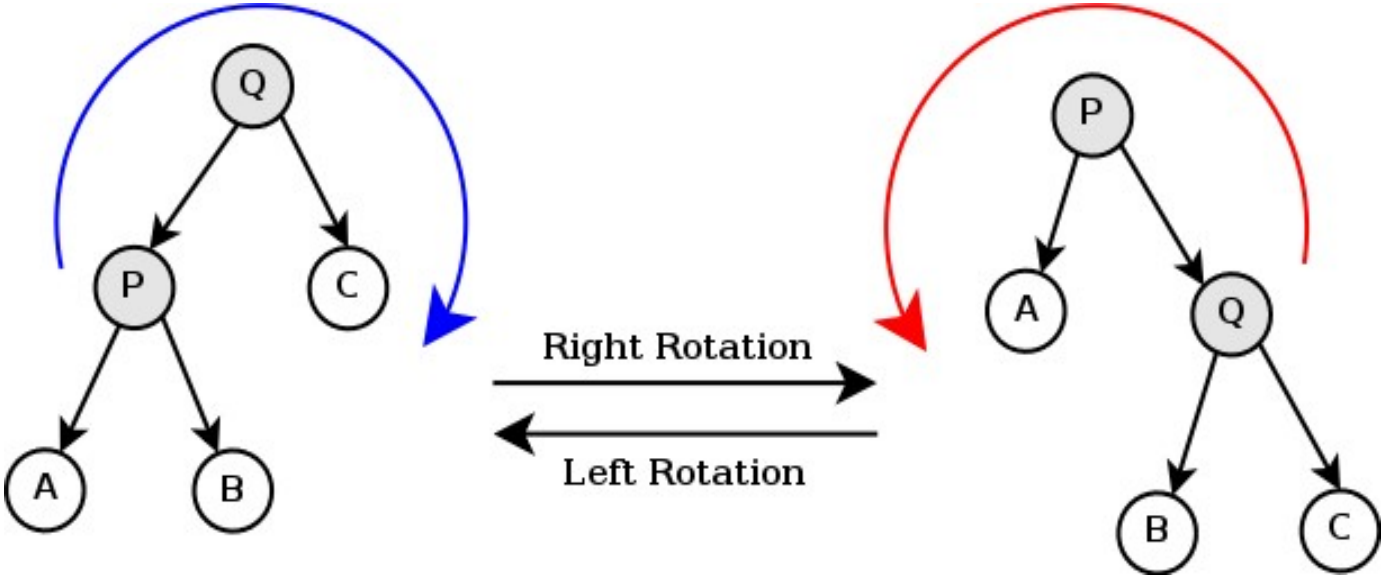
RB Tree Example



Balancing in RB Tree

- Suppose the black height of the root is k.
- Smallest path contains all black nodes i.e., k nodes
- Longest path contains alternate black and red nodes i.e., 2k nodes.
- So, no path in RBT, can be more than twice the another path.
- Hence, RBT is always balanced.

Tree Rotations



Operations

- Insertion
  - If new node is black then black height property will be violated.
  - So each new node inserted will be initially of red color.
  - Inserted red node may violate.
    i. Rule 1 if node is root.
    ii. Rule 3 if parent node is also red (double red problem).
  - Rule 1 violation is fixed by recoloring the node to black.
  - To fix Rule 3 violation there are some rules to be followed.

Insertion - Fixing of Rule 3 Violation

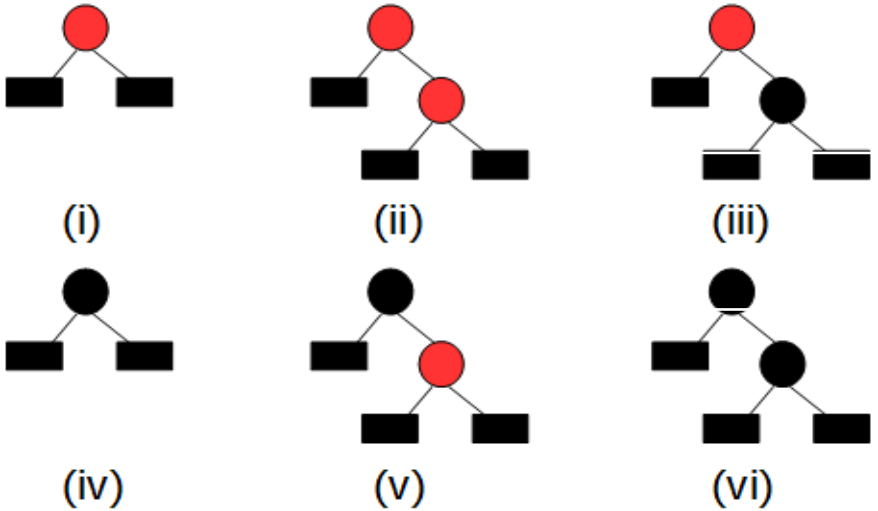
Parent Color	Parent Left/Right	Uncle Color	Node Left/Right	Case	How to balance?
Black	NA	NA	NA	NA	Already balanced
Red	Left (Case L)	Red (L_1)	NA	L_1	Recolor parent & uncle to black; Recolor grandparent to red; verify rule 3 for grandparent
		Black (L_2)	Right (L_2a)	L_2a	Rotate left about parent and Transform to case L_2b
			Left (L_2b)	L_2b	Recolor parent to black Recolor grandparent to red Right rotation is performed about the grandparent
	Right (Case R)	Red (R_1)	NA	R_1	Re-color parent & uncle to black, Re-color grandparent to red, verify rule 3 for grandparent
		Black (R_2)	Left (R_2a)	R_2a	Rotate right about parent and Transform to case R_2b
			Right (R_2b)	R_2b	Recolor parent to black, grandparent to red Left rotation is performed about the grandparent

Operations (Contd..)

- Searching
  - As in BST search starts from root by comparing node value.
  - Move left or right recursively till element is found.
- Deletion
  - Node can have no child, single child or two children.
  - In single child case node is replaced with child and colored black.
  - In two child case:
    i. Identify inorder successor and replace the node value with inorder successor.
    ii. Now delete inorder successor which can have no child or right child.
    iii. This gives 6 possible cases.

Deletion (Contd..)

The 6 possible cases for deletion:



Deletion Steps

- Case 1: Red node with no child
  - Delete the node
- Case 2: Black node with right red child.
  - Delete the node and replace it with child node colored black
- Case 3 : Black node with no children
  - By deleting the node black height property is violated.
  - To fix this we need to check some cases and then follow some rules.

Case 3-Black node with no children(Right case)

- Case 3.1: Sibling Red with black children
  - Right rotate from parent(P) and make sibling black.
  - Change color of near nephew to red.
  - If near nephew has red children apply left right or left left case accordingly.
- Case 3.2: Sibling black with red parent
  - 3 sub cases.
  - 1)No children for sibling.
  - 2)Left or right red child.
  - 3)Two red children.

Case 3.2 - Black sibling with red parent

- Case 3.21:Black sibling with no children
  - Color sibling red and parent black.
- Case 3.22: Black sibling with red left or right child
  - If left child only
    - Rotate right from parent.
  - If right child only
    - First left rotate about sibling and then right rotate about parent.
    - Change color of sibling to red and new parent to black.

Case 3.2 (Contd..)

- Case 3.23: Black sibling with two red children
  - Right rotate about sibling and make new parent red.
  - Left and right children of new parent is made black.
- Case 3.3: Black sibling with black parent
  - 1)No children
  - 2)Left or right red child only for sibling.
  - 3)Two red children for sibling

Case 3.3 (Contd..)

- Case 3.31: Black sibling with no children
  - Uncle black case
    - Left rotate from grand parent and change its color to red.
    - Change sibling color to red.
  - Uncle red case
    - Make uncle color black.
    - Left rotate from grand parent and change its color to red.

Case 3.3 (Contd..)

- Case 3.32: Black sibling with left or right red child
  - If left
    - Right rotate from parent and make far nephew black.
  - If right
    - Left rotate from sibling and right rotate from parent.
    - Make new parent black.
- Case 3.33: Black sibling with two red children
  - Right rotate from parent.
  - Far nephew is made black.

Applications of RB Tree

- Completely Fair Scheduler in Linux kernel
- Computational geometry data structures.
- C++ STL implementation of Map.

Summary

- Red Black Tree
- Rules/Properties of RB Tree
- Balancing in RB Tree
- Operations performed.
- Applications.

References

- https://www.cs.usfca.edu/~galles/visualization/RedBlack.html
- https://en.wikipedia.org/wiki/Red%E2%80%93black_tree

Large enough to Deliver, Small enough to Care

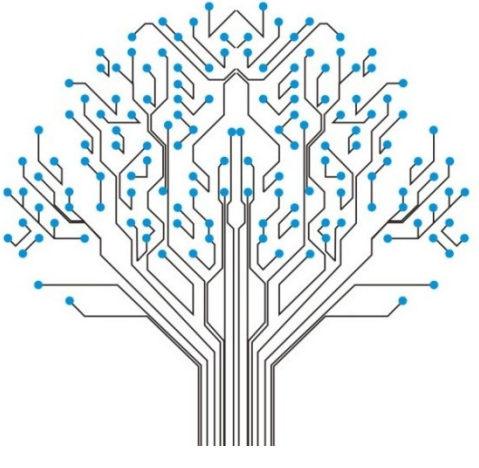
Global Village
IT SEZ
Bangalore

South Main Street
Milpitas
California

Raheja Mindspace
IT Park
Hyderabad

www.globaledgesoft.com

Thank you



Fairness

Learning

Responsibility

Innovation

Respect