CS 211 (Data Structures & Algorithms)
Instructors: Qasim Pasta & Asma Idris Mala

Left-leaning Red Black Trees

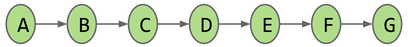
Muhammad Aaqib
18B-098-CS

Muhammad Muneeb-Ur-Rehman
18B-125-CS

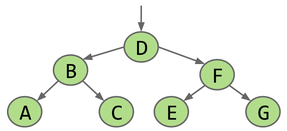
Abdullah Shah
18B-103-CS

Background

In order to overcome slow search problems in ordered data structures such as Linked Lists, we use **Binary Search Trees (BST)**.



Linked List
O(N)

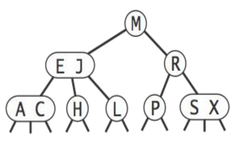


BST
O(Log N)

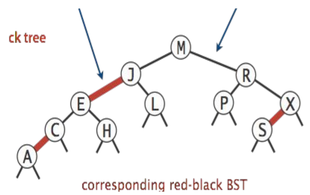
However, BSTs can become **spindly** and the performance of a spindly tree can be just as bad as a Linked List. This can be prevented by using **Self-Balancing Trees (SBT)**. A SBT is any BST that automatically keeps its height small in the face of arbitrary item insertions and deletions.
Red Black Trees and **2-3 Trees** are two of the many types of SBTs and while they're efficient at what they do, their implementations are byzantine.
This brings us to the topic of Left-Leaning Red Black Trees.

Introduction

LLRBs are structurally identical to 2-3 Trees yet meet all the original design goals set by RBTs
A BST with **left** red links that represents a 2-3 tree called a LLRB. There is a 1-1 correspondence between an LLRB and an equivalent 2-3 tree.



2-3 tree

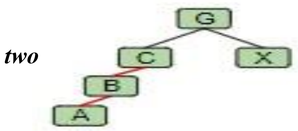


corresponding red-black BST

Properties

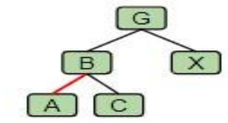
- No node has **two red links** otherwise it'd become a node with 3 items and 4 children i.e. a 4 node which are not allowed in 2-3 Trees i.e. a tree with nodes that have either 2 or 3 children.

Invalid, B has two red links.



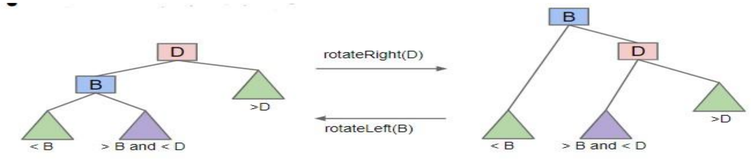
- Every path from the root to a leaf has the **same number of black links**. This is borrowed from 2-3 Trees as this is the mechanism they use to keep trees balanced.

Invalid, not black balanced.



Operations

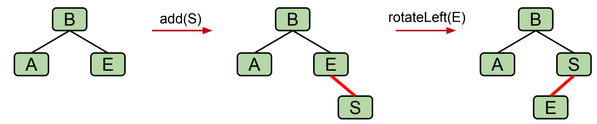
LLRBs use the following 3 operations to restore the above stated invariants after every insertion and deletion.
Rotate Right & Rotate Left:



Color Flip:

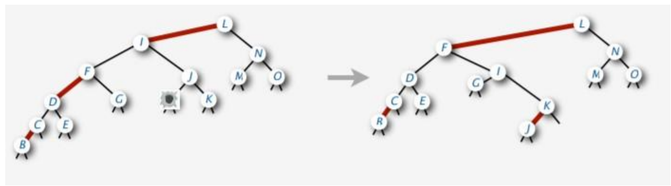
Insertion

Insertion operation is the same as that of a BST. When inserting use a red link. Then travel upwards the length of the tree and apply any of the aforementioned operations repeatedly to restore the inherent properties of LLRBs.



Deletion

When deleting a node, first travel down its left or right spine while shifting the red links downwards. Deleting a red node helps maintain 1-1 correspondence with 2-3 trees and consequently maintain the balance. Afterwards travel up the the length of the tree to correct any violations of the invariants. The searching of a node and the deletion of a leaf are the same as that in a BST.



Runtime

Since 2-3 Trees have O(Log N) runtime, searching, insertion and deletion in an LLRB are all O(Log N).

Reference: *Sedgewick, Robert. Left-Leaning Red Black Trees. (2008)*