Red-Black Tree

Background

- Essentially a self balancing binary search tree that has fast retrieval and storage of information (operations complete in known times)
- Invented in 1978 by Leonidas J. Guibas and Robert Sedgewick
- Nodes in a red-black tree hold a “color” that is either red or black, which is used to reorganize the tree to make sure it is balanced.

Usages

- For Java 8, the Hashmap uses a red-black tree (instead of a LinkedList) to store different elements with colliding hash codes. This improves time complexity from O(n) to O(log n) for colliding elements.
- Red-black trees are also used in cryptography algorithms and graphing algorithms (such as Dijkstra's shortest path algorithm)
- Can be used to effectively index data in database (fast search and retrieval of data)

Advantages:

- Guaranteed searching, insertion, deletion, and recoloring in O(log n) time (pretty useful)
- Dynamic in nature
- Balances trees easily, reducing the time for searching as well

Drawbacks:

- Complex to use (lots of edge cases)
- There are better alternatives in some cases (AVL trees or B-trees)
- Not suitable for massive datasets
- Extra steps for insertion and deletion operations to maintain the balance

AVL Tree

Background

- Another self balancing binary search tree (oldest self-balancing binary tree data structure to be invented).
- Invented in 1962 by Georgy Adelson-Velsky and Evgenii Landis
- More strictly balanced compared to red black trees, height-balanced, and not colored

Usages

- Index records in a database and to efficiently search in databases
- Used for some sets and dictionaries
- Applied in some corporate applications and storyline games

Advantages

- Cannot be skewed
- Faster lookups than Red-Black Trees
- Height is better balanced, so better searching and self-balancing

Disadvantages

- Difficult to implement
- More processing in order to balance the tree
- Less used/popular overall compared to other self-balancing tree counterparts
- Complicated insertion and removal operations (more rotations in general)

Arnav Doshi

B-Tree

Background

- Yet another self-balancing binary search tree
- Invented in 1970 by Rudolf Bayer and Edward M. McCreight
- Allows BSTs to have more than two children. Leaf nodes are at the same level

Usages

- Efficient for large datasets
- File systems (and storing files in them)
- Operating systems (manage memory efficiently)
- Network routers (efficiently route packets through the network)

Advantages

- Minimizes the number of disk accesses required to find a particular piece of data
- All keys are kept in sorted order, meaning the tree can be traversed sequentially
- “Partially full blocks”, meaning sped up insertion and deletion

Disadvantages

- Complex to implement and require lots of effort to create and maintain
- Lots of memory usage and processing time
- Not great for small datasets, better for larger