CS146

AVL Trees, Red-Black Trees (RBTs), and B-Trees are all types of self-balancing binary search trees, but they differ in their implementation and balancing strategies, as well as their typical use cases.

AVL Trees:
AVL trees are strictly balanced binary search trees.
They ensure that the heights of the left and right subtrees of any node differ by at most 1 (the AVL property). Insertions and deletions in AVL trees can require more rotations compared to Red-Black Trees, as AVL trees maintain stricter balance. As a result, AVL trees offer faster lookups compared to Red-Black Trees due to their more rigid balance properties.
However, AVL trees may require more memory due to additional balancing information stored in each node.

Red-Black Trees:
Red-Black Trees are more flexible in terms of balance compared to AVL trees.
They maintain balance by ensuring a set of properties, such as ensuring that no path from the root to any leaf is more than twice as long as any other path. Insertions and deletions in Red-Black Trees typically require fewer rotations compared to AVL trees.
Red-Black Trees are often preferred for use in environments where insertions and deletions are more frequent than lookups, as they offer better performance for these operations.
They are commonly used in language libraries like C++'s std::map and Java's TreeMap.

B-Trees:
B-Trees are balanced multiway search trees designed to work efficiently on disk storage as well as in-memory. Unlike AVL and Red-Black Trees, B-Trees are optimized for systems with large amounts of data that can't fit entirely into memory.
B-Trees have a variable number of children per node, typically more than two, allowing them to store more keys per node. They are used extensively in databases and file systems where disk access is a bottleneck, as they minimize the number of disk accesses required for insertions, deletions, and searches. B-Trees maintain balance by ensuring that all leaf nodes are at the same depth. B-Trees typically have higher fan-out (number of children per node), resulting in fewer levels of the tree compared to AVL and Red-Black Trees, which reduces the number of disk accesses required for operations.

In summary, AVL trees are rigidly balanced binary search trees offering faster lookups but requiring more rotations during insertions and deletions. Red-Black Trees offer more flexible balance properties with fewer rotations required, making them suitable for environments with frequent insertions and deletions. B-Trees are optimized for disk storage and large datasets, minimizing disk accesses by having a higher fan-out and ensuring all leaf nodes are at the same depth.