1. Properties of Red-Black Tree

A red-black tree is a self-balancing binary search tree in which each node is assigned a color, either red or black. The root node is always black. Red nodes cannot have red children, which means no two red nodes can be adjacent. Every path from a node to its descendant leaf nodes contains the same number of black nodes. These properties ensure that the tree remains approximately balanced, which helps in maintaining efficient searching, insertion, and deletion operations.

2. Recolor and Rotation Process

Recoloring is a process used in red-black trees to change the color of nodes in order to maintain tree properties after insertion or deletion. Rotation is a structural adjustment where nodes are rearranged to restore balance. Recoloring changes node colors without altering structure, while rotation changes the structure of the tree. Both processes work together to keep the tree balanced.

3. Difference Between Zig and Zig-Zag Rotation

Zig rotation occurs when a node and its parent are on the same side of the tree. Zig-zag rotation occurs when a node and its parent are on opposite sides. Zig rotation involves a single rotation, while zig-zag rotation requires two rotations. These rotations are mainly used in splay trees to move frequently accessed nodes closer to the root.

4. Concept of 2-Node and 3-Node

A 2-node contains one data element and has two child pointers. A 3-node contains two data elements and has three child pointers. These concepts are used in multi-way trees to maintain balance. They help in efficient data storage and searching by reducing tree height.

5. Splay Operation in Splay Trees

The splay operation moves an accessed node to the root of the tree. This is done using a series of rotations. The goal is to make frequently accessed elements faster to reach. Splaying improves overall performance by keeping commonly used nodes near the top of the tree.

6. Amortized Analysis of Splay Tree

Amortized analysis studies the average performance of operations over a sequence of operations. In splay trees, individual operations may take more time, but the average time over many operations remains efficient. This analysis shows that splay trees perform well in practice even without strict balancing rules.

7. No Height or Balance Factor in Splay Tree

Splay trees do not store height or balance factor information like AVL trees. Instead, they rely on splay operations to keep frequently accessed nodes near the root. This makes the structure simpler and reduces memory usage. Balance is maintained dynamically through access patterns rather than strict rules.