Red-Black Tree and Ternary Search Tree - DSA Guide

Red-Black Tree - Introduction

A Red-Black Tree (RBT) is a self-balancing Binary Search Tree (BST). Each node contains an extra bit for color (red or black).

Properties:

- 1. Each node is either red or black.
- 2. Root is always black.
- 3. Red node can't have red child.
- 4. Every path from node to NULL has same number of black nodes.

Red-Black Tree - Insertion

Steps:

- 1. Insert node as in BST (mark RED).
- 2. If violations occur, fix using recoloring or rotations.

Cases:

- Uncle RED: Recolor.
- Uncle BLACK: Rotate & Recolor.

Red-Black Tree - Deletion

Deletion is complex. If deleted node is RED: just remove. If BLACK, handle double black issue with rotation and recoloring.

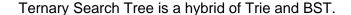
Red-Black Tree - Applications

Used in:

- C++ STL Map/Set
- Linux Process Scheduler
- DB Indexing

Ternary Search Tree - Introduction

Red-Black Tree and Ternary Search Tree - DSA Guide



Each node has 3 children:

- Left: < char

- Mid: == char

- Right: > char

Ternary Search Tree - Applications

Used in autocomplete, spell check, and storing dictionary words.

Ternary Search Tree - Deletion

Similar to Trie deletion but with 3-way branching. Carefully deallocate and re-balance if needed.

Ternary Search Tree - Autocomplete

Traverse for prefix, then do DFS from mid node to gather suggestions.

Ternary Search Tree - Longest Word

Perform DFS while tracking current path and max-length word at each terminal node.