NAME: Haider Ali

SAP: 53109

COURSE: ANALYSIS OF ALGORITHM

Table of Contents

---

1. Introduction

The Red-Black Tree (RBT) is a self-balancing binary search tree, ensuring logarithmic time complexity for insertion, deletion, and search operations. It is widely used in real-time systems, memory management, and library implementations (e.g., TreeMap in Java). Its balancing mechanism makes it reliable for maintaining sorted data under frequent updates.

---

2. Methodology

2.1 Pseudocode Overview

1. Create a new red node.
2. Insert it as in a standard BST.
3. Rebalance the tree using rotations and recoloring based on Red-Black properties.

2.2 Python Implementation

The algorithm was implemented in Python with the following capabilities:

- Insert operation with automatic balancing
- In-order traversal for testing
- Rotations (left and right)
- Fix-up function to enforce tree properties

---

3. Complexity Analysis

- **Time Complexity**
  • Insertion: O(log n)
  • Search: O(log n)
  • Deletion: O(log n)
- **Space Complexity**
  • O(n) for storing nodes
  • O(log n) auxiliary stack in recursion (traversal)

---

4. Real-World Application

Red-Black Trees are used in:

- **Java TreeMap** and **TreeSet** implementations
- **Linux Virtual Memory Area (VMA) Trees**
- **Databases** to maintain ordered indices with efficient access
- **Multithreaded systems** needing consistent, fast lookups

---

5. Limitations

- Slightly more complex than AVL trees
- More rotations and comparisons than simpler BSTs
- Harder to implement and debug due to strict color rules

---

6. Addressing CLOs

CLO	Description	Addressed In
2.1	Explain NP, NPC, approximation	Introduction
3.1	Implement algorithm, solve problem	Python Code, Output
4.1	Analyze time/space complexity	Complexity Section
4.2	Asymptotic notations	Complexity Section
5.1	Evaluate real-world use	Application Section
6.1	Design solution creatively	Methodology (Rotation Fix)

---

7. Conclusion

The Red-Black Tree is an efficient and widely adopted structure for maintaining balanced, ordered data. Its implementation in Python demonstrated the preservation of balance through rotations and color adjustments. With guaranteed O(log n) operations, RBTs provide a robust foundation for performance-critical systems.

---

8. GitHub Repository