Report on Implementation of Red-Black Tree

Course Title: Data Structure

Course Code: CSE207

Section: 08

Submitted to

Md. Manowarul Islam

Adjunct Faculty

East West University

Department of Computer Science and Engineering

Submitted by:

Md. Sifatullah Sheikh

2022-1-60-029

Urmi Kirtonia

2022-1-60-184

Nuzath Tabassum Arthi

2022-1-60-185

Afsara Tasnim Biva

2022-1-60-396

Date of Submission: 27-12-2023

Introduction:

A Red-Black Tree is a special kind of binary search tree where each node is colored either red or black. What makes it unique is its ability to automatically balance itself after adding or removing nodes. This self-adjustment is crucial for maintaining efficiency in operations like inserting, deleting, and searching. The rules it follows ensure that the tree remains balanced, guaranteeing a maximum time complexity of O(log n) for these operations, no matter how the tree initially looks. While a Red-Black Tree might have a slightly larger height than some other trees, this doesn't significantly impact lookup speed. Its looser height rule makes inserting and deleting nodes faster.

Code Overview:

The code is implemented in C. It consists of the following functions:

1. `‘leftRotate’` Function:

This function rotates the given node to the left and adjusts parent and child pointers accordingly to maintain Red-Black Tree properties.

2. `‘rightRotate’` Function:

This function rotates the given node to the right and adjusts parent and child pointers accordingly to maintain Red-Black Tree properties.

3. `‘colorinsert’` Function:

It adjusts colors and performs rotations to ensure the Red-Black Tree remains balanced after inserting a new node.

4. `‘inorderTree’` and `‘postorderTree’` Functions:

These functions are used for recursive inorder and postorder tree traversal, respectively. They print the key and color of each node during traversal.

5. `‘traversal’` Function:

It prints the root of the tree and calls `‘inorderTree’` and `‘postorderTree’` to display the nodes in the specified order.

6. `‘search’` Function:

It searches for a given value in the Red-Black Tree. Returns 1 if the specified value is found in the tree; otherwise, returns 0.

7. `‘insert’` Function:

It allocates memory for a new node, sets its key and color, inserts it into the tree, and then calls `‘colorinsert’` to maintain Red-Black Tree properties.

8. `‘min’` and `‘successor’` Functions:

These functions help in finding the minimum value and the successor node, which are used in the `‘deleteNode’` function.

9. `‘colordelete’` Function:

It adjusts colors and performs rotations to ensure the Red-Black Tree remains balanced after deleting a node.

10. `‘deleteNode’` Function:

This function finds the node to be deleted, handles different cases based on the node's children, and then calls `‘colordelete’` to maintain Red-Black Tree properties.

11. `‘printTree’` and `‘print’` Functions:

These functions display the Red-Black Tree in a visually structured manner.

12. `‘main’` Function:

The main function presents a menu-driven interface, allowing the user to insert, delete, traverse, and exit the Red-Black Tree application. It uses the aforementioned functions to perform these operations.

Applications:

1. C++ and Java Libraries: Red-Black Trees are commonly used in C++ libraries (like map, multiset, and multimap) and Java packages (java.util.TreeMap and java.util.TreeSet) for efficient searching and sorting.
2. CPU Scheduling in Linux: The Linux Completely Fair Scheduler relies on Red-Black Trees to manage and organize tasks for CPU scheduling, ensuring fairness and efficiency.
3. Machine Learning (K-mean Clustering): Red-black trees play a role in the K-means clustering algorithm in machine learning, helping to reduce the time complexity of the algorithm.
4. MySQL Indexing: MySQL utilizes Red-Black Trees for indexing tables, enhancing the speed of searching and inserting data into databases.

5. Virtual Memory Management: In some operating systems, Red-Black Trees are employed in the virtual memory manager to keep track of memory pages and optimize their usage.
6. Programming Language Implementations: Programming languages like Java, C++, and Python incorporate Red-Black Trees as a built-in data structure, providing efficient methods for searching and sorting data.
7. Graph Algorithms: Red-black trees are essential in implementing graph algorithms such as Dijkstra’s shortest path and Prim’s minimum spanning tree, ensuring optimal traversal and exploration.

Conclusion:

The Red-Black Tree implementation shows that the fundamental ideas of color coding, rotation, and balance are understood, which is necessary to keep the tree functional. All in all, the code provides a useful example of Red-Black Tree functions and lays the groundwork for additional research and improvement.