Red/Black Tree Implementation Report

Made by: Will Frautschy // Aasama Prabhakar

# Implementation of Red Black Tree and Test Suite

**GitHub Source**: <https://github.com/wfrautschy4/CSE-2331-Final-Project>

For this project, we used java to code it while using classes like the setup of software 2. The library includes two classes, one for binary tree, and another for the implementation of Red/Black Tree. The implementation includes functions for insertion, deletion, searching, and traversing. The test suite is also in the repo and ensures that each method works correctly.

# A screenshot of a computer AI-generated content may be incorrect.Visualization

For our project, to visualize and understand how the tree is changing given any operations are being acted on it, we use a library called TreePrinter which simply prints the tree into the command line. While it is not perfect since you cannot animate changes, it does help you understand the whole structure of the tree without manually traversing it at any given instance. It also had to ability to display red or black versions of the nodes to accurately depict the colors of the red/black tree.

# Documentation and Presentation

|  |  |
| --- | --- |
| Insert | The way that insert is implemented is by recursively climbing down the trees and picking the child whose is appropriate for the data being inserted. Once the node has been added, it checks if the node needs balanced, then traverses back upwards to the parent and checks if it needs balanced until it reaches the top of the tree. |
| Delete | Delete works similarly by recursively climbing down the tree until the node being removed is found. Once it is found it is replaced by another appropriate node nearby and then balances its way back up the tree if required. |
| Search | Recursively climbs down the tree and returns true if the node is found |
| Traverse | Prints |

# Performance Analysis

|  |  |
| --- | --- |
| Insert |  |
| Delete |  |
| Search |  |