Due: Friday, Dec 2, 2022 at 11:59pm

Overview

Building on your knowledge of Binary Search Trees, this lab will focus on implementing a red-black tree.

Data Structures

A BST is a red-black tree if it satisfies the following rules:

- Every node is either red or black.
- Every leaf counts as black.
- If a node is red, then both of its children are black.
- Every simple path from a node to a descendant lead contains the same number of black nodes.
- The root node is always black.

You should fill out all of the methods in the provided skeleton code **rb_tree.py**. You may add additional methods but should not add any private variables to the class. Like before, you should not alter any names for any of the classes, methods, or files. Your code will be graded using an autograder on Gradescope. To start you off, some test cases have been provided in **test_lab4.py**. You may not use any data structures from the Python standard library. Some inbuilt functions in python can be used as required.

For your convenience you are given working traverse, insert, and delete methods for BST. You simply need to extend the functionality to support balanced insert and delete operations.

RB Tree

`print_with_colors(self, curr_node)` : Same as the `print_tree()` method provided in the skeleton code except it prints the color code along with the node. 'B' for black, 'R' for red. It also prints in preorder traversal format just like `print_tree()`. This is implemented for you. The function helps with the debugging process.

`left_rotate(self, curr_node)` : [see p. 328] Perform a left rotation on `curr_node`. If y =`curr_node.right` and x=`curr_node`. Then, if y is not `None`, we can perform a left rotation by doing the following:

- y becomes the new root of the subtree.
- x becomes the left child of y.
- y's left child becomes x's right child.

Note: Be careful not to set y to be the root of the entire tree. This is a local operation. Here, the root of the subtree is `curr_node`, the node that the rotation is applied to.

`right_rotate(self, curr_node)` : Perform a right rotation. This operation is much like the left rotation but with 'left' being swapped for 'right' and 'right' being swapped for left.

Note: Like the previous lab, for `left_rotate` and `right_rotate`, you should raise `KeyError` when there is nothing to rotate

`insert(self, data)` : [see p. 330] Insert a node into the red-black tree. As described in the text, this consists of performing a BST insert, checking for violations of the red-black tree rules, and performing rotations and re-colorings as needed to restore the red-black tree rules. This should be done by implementing the `__rb_insert_fixup()` method.

`delete(self, data)` : [see p. 338] Delete a node from the red-black tree. As described in the text, this consists of performing a BST delete, checking for violations of the red-black tree rules, and performing rotations as needed to restore the red-black tree rules. This should be done by implementing the `__rb_delete_fixup()` method.

Testing

Some sample test cases have been provided in **test_lab4.py** but this provided list is not exhaustive. You should provide tests for the following:

Test left rotation of intermediate node

- Insert the following elements using `bst_insert()`: $[7, 5, 9, 3, 6, 8, 10, 1, 2]$
- Left rotate node 9
- Test if the rotation was performed correctly.

Test colors after insertion

- Insert the following elements using `bst_insert()`: $[7, 5, 9, 3, 6, 8, 10, 1, 2]$
- Test the color of every node after the above insertion.

Add 3 of additional test cases of your own making.

It is strongly encouraged that you write even more tests.

Submission

Compress the **rb_tree.py** and **test_lab4.py** files and upload in Gradescope similar to the previous programming assignments. The test cases file should contain all the additional test cases that you have written to test your code.

Grading

The assignment will be graded as follows:

— AutoGrader - [70pts]

— Style - [10pts]

— Additional Test Cases - [15pts]

5pt- Test left rotation of intermediate node.

5pt- Test that node colors are correct following an insertion.

5pt- Add 3 additional test cases of your own.

— DocStrings - [5pts]

2pt- Provide docstring for `left_rotate()`

1pt- Provide docstring for `right_rotate()`

1pt- Provide docstring for`__rb_insert_fixup()`

1pt- Provide docstring for `__rb_delete_fixup()`