CPE202

Lab10: Red Black Tree

You only need to do either this lab or Lab 9. In case you do both lab 9 and lab 10, you can earn upto 100 points as extra credit toward the lab category. There is a problem at the end of this lab for which you can earn additional extra credit in the lab category, but you need to demo your solution for the extra-credit problem to your instructor or TA by 7 pm Tuesday, March 17.

BinarySearchTrees (BSTs) are great. They allow us to organize and search for items in O(log(n)) time rather than O(n) time, and to insert things in O(log(n)) time as well.

However, a binary search tree can devolve into a simple linear list, if the elements are inserted in sequential order. In that case, the time complexity of using BST becomes O(n).

Therefore, BST needs to be balanced.

There are a number of different ways of balancing BST. Two of the most well-known are AVL trees and Red-Black trees. Of the two, red-black trees are slightly newer, have more reliable insertion times, and are a bit simpler to implement for us. (Note: We need to relax the definition of a balanced tree to consider Red-Black trees to be balanced.)

The properties of a Red-Black tree are:

1. Each node is labeled as either "red" or "black."
2. Root is always black
    a. This rule is relaxed in some implementations.
3. Every path from root to leaf encounters the exact same number of black nodes.
4. The child of a red node must be a black node. In other words, there can't be more than one red node in a row along a path from root to leaf.
    a. NIL (None) leaves are considered black.

In this lab, you are going to implement a self-balancing BST using the red-black tree.

1. Create a file “bstree_rb.py” by copying your bst.py from the lab 5 and modify the code and implement the red-black tree.
2. Add a str type field “color” in BSTNode class (or whatever your class for a tree node is called) for red black tree. Your class for a tree node must have three other fields called “key”, “left”, and “right”. The left and the right should point to the left and the right

subtree respectively. You also need an additional field for storing the value associated with the key in the node.

3. Add a function “rebalance” in bst_rb.py for the red black tree.
4. Modify the insert function/method or its helper function/method in the bst_rb.py to color each node either black or red, and to rebalance the tree calling the rebalance function if the red black tree property is violated. You do not need to modify the delete function for balancing the tree after deletions is beyond the scope of this course.
5. Reuse the classmate.py from the lab 5.
6. Download the classmates.tsv from polylearn if you do not have it.
7. Create a file called “lab10.py” by copying lab5.py. Import bst_rb instead of bst. TreeMap should now use the red-black tree. i. e. the self.tree points to the root node of a red-black tree.
8. Modify the import_classmates function so that the list of Classmate objects is no longer shuffled. We do not need to shuffle it anymore because the tree is supposed to balance itself.
9. Create a test file “lab10_test.py” and write tests for functions that return some values, including import_classmates and search_classmate functions.
10. Visualized images of expected binary search tree of classmates are available on polylearn.
11. **OPTIONAL EXTRA CREDIT ASSIGNMENT (You can earn extra 50 points for the lab, but only if you demo it to me):** create a function to generate a visual representation of the red black tree using graphviz as shown in lecture and create image (png) files of binary search trees of classmates. The function needs to traverse the tree using preorder traversal, and output a text description of the tree in the dot language to a file. Then you can use “dot” command to create a png file from the text description: dot out.dot -Tpng -o out.png, where out.dot is an output file generated by your program and out.png is an image file name and both filenames are for you to choose freely. The output image should look like the one (all.png) posted on polylearn. The “dot” command is available on the unix server on campus. You can find the documentation about graphviz here: https://graphviz.gitlab.io/
https://renenyffenegger.ch/notes/tools/Graphviz/examples/index
https://codeyarns.com/2013/06/24/how-to-convert-dot-file-to-image-format/
12. **zip all files including classmate.py into one file and submit your work except for #11 above to the grader, and then submit it to polylearn.**