EC330 Applied Algorithms and Data Structures for Engineers
Fall 2018

Homework 6

Out: November 8, 2018
Due: November 19, 2018

This homework has a written part and a programming part. Both are due at 8:59 am on November 19. You should submit both parts on Blackboard. For the written part, you should submit a single PDF file containing either typeset answers or scanned copies of hand-written answers. Make sure you write your answers clearly. For the programming part, your code should be easy to read and understand, and demonstrate good code design and style. Your program must compile and run on the lab computers.

1. **AVL Tree and B-Tree [20 pt]**
   For each of the following determine if the claim is correct. It so, explain why. If not, give a counter example.
   a. The order in which elements are inserted into an AVL tree does not matter, since the same AVL tree will be established following rotations.
   b. The order in which elements are inserted into a B-tree does not matter, since the same B-tree will be established.

2. **Red-Black Tree [20 pt]**
   a. Prove by induction on the black height of x (bh(x)) that a subtree rooted at x has at least $2^{bh(x)}$-1 internal nodes for a red-black tree.
   b. A post-order traversal of a red-black tree on the numbers 1 to 15 gives the following colors: R R B B B B B R B R R B R B B. Produce the tree.

3. **Binary Search Tree [60 pt]**
   For the programming part (part b), make sure to write your name and BU ID in a comment at the top of the program, along with your collaborator's name and BU ID, if any. To use the compiler on the lab computers, run "*module load gcc*" first.

   Consider two binary search trees that contain the same set of *unique* keys, possibly in different orders.
   a. Devise and analyze an efficient algorithm that will transform any given binary search tree into any other binary search tree (with the same keys) using only ZIG and ZAG rotations.
      Bonus [10 pt]: Prove the correctness of your algorithm.
   b. The provided BST.h, and BST.cpp files contain a BST class, implementing a binary search tree, and a Rotation class, which stores a rotation.
      Implement a new derived class of BST, MyBST, which extends the binary search tree with a transform method, which implements your algorithm from part (a).

You should also implement a main function, which receives two files, T1.txt and T2.txt (you may assume that these will be the names of the input files), containing one integer per line, to be inserted in order into two binary search trees. The transform method should receive the BSTs generated from T1 and T2, and return a vector of rotations required to transform T1 into T2, which is then printed out.

You may *not* modify BST.h and BST.cpp, but you may add methods to MyBST.cpp as you see fit (and add a MyBST.h file).

Assuming that the required rotations are a ZIG rotation on pivot=3, followed by a ZAG rotation on pivot=8, your output should be as follows:

```
ZIG on 3
ZAG on 8
```

To get full credit your solution must work on any two binary search trees that contain exactly the same set of unique keys.

Submit your solution as a single zip file consisting of your **MyBST.cpp, MyBST.h,** and **main.cpp**, along with a **modified makefile** that was used to compile it on the lab computers. No additional files should be submitted.