

CS 223

Laboratory Assignment #3

Due: at the end of the lab in Week 9 (Mar 13, 09)

The Problems

1. Professor Bunyan thinks he has discovered a remarkable property of binary search trees. Suppose that the search for key k in a binary search tree ends up in a leaf. Consider three sets: A , the keys to the left of the search path; B , the keys on the search path; and C , the keys to the right of the search path.

Professor Bunyan claims that any three keys $a \in A$, $b \in B$, and $c \in C$ must satisfy $a \leq b \leq c$.

Give a smallest possible counterexample to the professor's claim.

2. Let T be a binary search tree whose keys are distinct, let x be a leaf node, and let y be its parent. Show that $key[y]$ is either the smallest key in T larger than $key[x]$ or the largest key in T smaller than $key[x]$.

3. Given any n -node binary tree T , show that one can always use at most $n - 1$ right rotations to transform T into a right-going path.

4. In class, we covered Red-Black trees. In this lab assignment you need to implement it. You do not really have to start from scratch, and it is encouraged that you start from the `BinarySearchTree` class we have been using in CS223 since 2006 (<http://www.cs.montana.edu/courses/spring2006/223>). (Or you can use some red-black tree class in Sedgwick, e.g., p. 580.) In either case, you need to implement the following 8 functions:

insert — inserts a node into the red-black tree.

delete — deletes a given node from the red-black tree.

search — given a key value, returns a node from the tree with the same key, or report no such node exists.

printTree — you can use the existing method in `BinarySearchTree`, extra credit will be given if you print red nodes in *red* and black nodes in *black*.

treeHeight — returns the height of the tree.

treeBlackHeight — returns the black height of the tree.

treeMinimum — returns the smallest key value in the tree.

treeMaximum — returns the largest key value in the tree.

Test runs: While you can try your own datasets, we need the following two test results from you:

(1) Insert the following numbers into a new red black tree: 5,10,9,1,8,6,4,12,20,25,35,11,30; print out the tree. Then delete 10,20,30 and print out the tree again.

(2) Insert the numbers 25,24,...,1 into a new red black tree. Then call **treeHeight**, **treeBlackHeight**, **treeMaximum** and **treeMinimum** on this tree. Print out the results of these calls.

Finally, call **delete(search(11)),delete(search(13)),...,delete(search(21))** on the search tree. Print out the resulting tree.