ICS 311: Algorithms Spring 2025

Problem Set 4

Kyle Berney *Due: Friday, February 7, 2025 at 4pm*

You may discuss the problems with your classmates, however **you must write up the solutions on your own** and **list the names** of every person with whom you discussed each problem.

Start **every** problem on a separate sheet of paper, with the exception of Problems 1 (Peer credit assignment) – Problem 1 can be on the same page as any other problem. Any problem that starts on the same sheet of paper as some other problem will receive 0 points!

1 Peer Credit Assignment (1 point extra credit for replying)

Please list the names of the other members of your peer group for this week and the number of extra credit points you think they deserve for their participation in group work.

- You have a total of 60 points to allocate across all of your peers.
- You can distribute the points equally, give them all to one person, or do something in between.
- You need not allocate all the points available to you.
- ***You cannot allocate any points to yourself!*** Points allocated to yourself will not be recorded.

2 Understanding Binary Search Trees (20 pts)

Suppose we are searching for some key x in a BST. As we perform the search, we compare x to the keys on the path from the root to the leaf in the BST. Which sequences below **cannot** be the sequence of keys examined during the search? Explain why not. ***Do not*** assume that the searches below are performed on the same BST.

(a) **(4 pts)** 85, 76, 30, 15, 6, 1

(b) **(4 pts)** 16, 33, 77, 54, 73, 60

(c) **(4 pts)** 99, 78, 44, 87, 57, 52

(d) **(4 pts)** 45, 15, 88, 75, 52

(e) **(4 pts)** 1, 10, 2, 9, 3, 8, 4, 7, 6, 5

3 Counting Binary Trees (30 pts)

How many binary tree shapes of n nodes are there with height $n-1$? Prove your answer. *Hint: use induction. To make a good guess, try drawing all possible trees for heights 0, 1, 2,... and identify the pattern.*

4 Red-Black Tree and (2,4)-Tree Deletion (20 pts)

Preliminary Comments

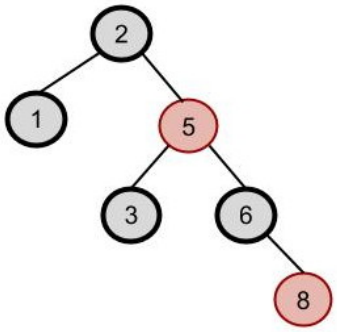
The lecture notes were based on Goodrich&Tamassia's textbook, because they show the correspondence of RBTs to (2,4)-trees, which makes the former easier to understand as balanced trees. The CLRS version differs somewhat. You will need to read the CLRS text to answer this question.

The cases for insertion are similar between G&T and CLRS, but the terminology differs (e.g., what the letters w, x, y, and z refer to). The cases for deletion differ: G&T have 3 while CLRS have 4! Be careful because there are mirror images of every situation (e.g., is the double black node a left child or a right child?): G&T and CLRS may be describing the same situation with mirror image graphs.

The top level methods in CLRS for RB-INSERT (p. 338) and RB-DELETE (p. 348) essentially do binary search tree (BST) insertion and deletion, and then call FIXUP methods to fix the red-black properties. Thus they are very similar to the BST methods TREE-INSERT (p. 321) and TREE-DELETE (p. 325). The real work specific to RBTs is in these fixup methods, so we will focus on them in these questions, but you should also study the top level methods to understand them as BST methods.

Problems

(a) **RBT as (2,4)-Tree:** Draw the (2,4)-tree that corresponds to the RBT shown below.



(b) **Deletion:** Delete key **2** from the red/black tree shown above, and show the deletion in the (2,4) representation.

Show **every** state of the RBT tree, including after the BST-style deletion and after each case applied by RB-DELETE-FIXUP. Clearly identify the colors of the nodes. Also show the state of the (2,4)-tree for each of these RBT states. If a double black node occurs (node x in CLRS), clearly identify which node it is. For each state change, identify **both** G&T case(s) from the web notes and the CLRS case(s) from the textbook that are applied in each of your steps. Your final diagram should show the RBT after RB-Delete-Fixup and the (2,4) tree representation that results.

(c) **More Deletion:** Delete key **1** from the **initial** red/black tree shown above (**NOT** the tree that results from (b)), showing all steps as specified above.

5 Sorting using Balanced BSTs (30 pts)

Explain how you would use a $(2, 4)$-tree or a red-black tree to sort n comparable elements in $O(n \log n)$ time in the worst case. Justify why the runtime of your solution is $O(n \log n)$.

6 BST Transformations (OPTIONAL - 0 pts)

In this problem you will prove that any arbitrary n-node binary search tree T_1 can be transformed into any other arbitrary n-node binary search tree T_2 using $O(n)$ rotations.

(a) Show that at most $n-1$ right rotations suffice to transform any arbitrary n-node binary search tree T_1 into a right-going chain. To do this do the following (no partial credit if you skip one of the steps below, e.g., if you present just the algorithm but do not prove its correctness or how many rotations it performs):

(i) Present an algorithm that performs the transformation

(ii) Prove that your algorithm correctly transforms *any* arbitrary n-node BST into a right-going chain, and

(iii) Prove that your algorithm performs at most $n-1$ right rotations in the process (and no left-rotations).

(b) Observe the relationship betweeen right rotations and left rotations and use it to show how to transform the right-going chain into any arbitrary BST T_2 in at most $n-1$ left rotations.

(c) Conclude that the total number of rotations to transform an arbitrary n-node BST T_1 into any other arbitrary n-node BST T_2 is $O(n)$.