ICS 311, Fall 2020, Problem Set 06, Topic 10B & 11

Firstname Lastname <uhumail@hawaii.edu> Section 1

Due by midnight Tuesday 10/20. There's a lot of extra credit potential here: 50 points total.

#2 Sorting larger numbers (10 pts)

Suppose we have *n* integers in the range 0 to n^4-1, and we want to sort them in *O(n)* time. (Since the font is small: that is n^4 - 1.)

a. (2 pts) Show that Counting-Sort is not an option by analyzing the runtime of Counting-Sort on this data. *Hint:* Identify the value of *k* and invoke the analysis that was presented in the textbook or lecture notes.

b. (3 pts) Show that unmodified Radix-Sort is not an option by analyzing the runtime of Counting-Sort on this data. *Hint:* Identify the value of *k* for each call to Counting-Sort. This is not the same as in the previous problem. Then identify the value of *d,* and invoke the analysis that was presented in the textbook or lecture notes.

c. (5 pts) CLRS states that "we have some flexibility in how to break each key into digits", and prove a relevant Lemma 8.4. Using this as a hint, describe a modified Radix-Sort that would sort this data in O(*n*) time and use Lemma 8.4 to show that this is the correct runtime.

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

Preliminary Comments

In this problem we delve deeply into the CLRS code for tree deletion. 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.

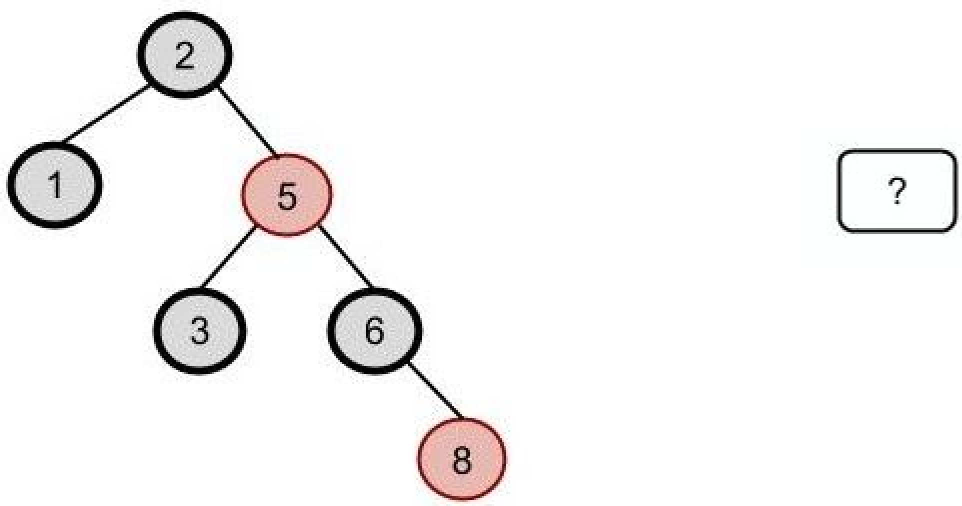
The cases for insertion are similar between G&T and CLRS, but the terminology differs (e.g., what the letters w, x, y, and z refer to). The cases for deletion differ: G&T have 3 while CLRS have 4! Be careful because there are mirror images of every situation (e.g., is the double black node a left child 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. 315) and RB-DELETE (p. 324) 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. 294) and TREE-DELETE (p. 298). 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

You may want to use the Google Drawing template provided in the document Problem-Set-06-Red-Black-Tree-Deletion-Start. Then convert the results into images to insert here.

(a) RBT as 2-4 Tree (2 pts) Draw the 2-4 tree that corresponds to the RBT shown below (replace the ? box).



(b) Deletion (6 pts): 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. As above, you'll have the RBT on the left and the 2-4 tree on the right.
- 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 (12 pts): Delete key **1** from the **initial** red/black tree shown above (NOT the tree that results from (b)). Show all steps as specified above.

#4 Red-Black Tree Height (5 pts)

(a) (4 pts) What is the largest possible number of internal nodes (those with keys) in a red-black tree with black height k? What is the height of the corresponding 2-4 tree? Prove your claims.

(b) (2 pts) What is the smallest possible number of internal nodes (those with keys) in a red-black tree with black height k? What is the height of the corresponding 2-4 tree? Prove your claims (you should use Lemma 13.1 to make this easier).

#5 Red Nodes in Red-Black Tree (5 pts)

Consider a red-black tree formed by inserting *n* nodes with RB-Insert. Prove that if *n* > 1, the tree has at least one red node. *Hint:* Nodes are red when inserted (line 16 of RB-Insert). Show that if *n* > 1 one node must be red after RB-Insert-Fixup is complete.

(It is not sufficient to say that the second node inserted will be red. You must show that some node remains red under all possible insertion sequences and the transformations that result.)

#6 O(n) sort of variable length integers (10 pts)

Suppose we have a way of representing positive integers with variable numbers of digits, for example “3”, “61” and “317” may be included. Assume that there are no leading 0s, for example, “317” not “00317”. You are given an array of positive integers under this representation where the total number of digits over all the integers in the array is n. Show how to sort the array in O(n) time. (4 points for strategy and 6 points for analysis.)

Note that it is possible for one integer to have O(n) digits and all the others to be small. Therefore unmodified radix sort won’t work: d = O(n) passes are required (and it is not viable to pad the integers with leading 0’s to make them all the same length).