CIS 121 — Data Structures and Algorithms
Homework Assignment 4

Assigned: October 9, 2018
Due: October 15, 2018

Note: The homework is due electronically on Gradescope and Canvas on Monday, October 15 by 11:59 pm EDT. For late submissions, please refer to the Late Submission Policy on the course webpage. You may use a maximum of 2 late days on this homework.

A. Gradescope: You must select the appropriate pages on Gradescope. Gradescope makes this easy for you: before you submit, it asks you to associate pages with the homework questions. Failing to do so will get you 5% off, which cannot be argued against after the fact. Gradescope may prompt you with a warning to select your cover page, please ignore this warning.

B. Canvas: You must also submit your assignment on Canvas. Forgetting to do so will incur a 10% penalty.

C. \LaTeX: You must use the \texttt{hw121.cls} Latex template provided on the course website, or a 5% penalty will be incurred. Handwritten solutions or solutions not typeset in Latex will not be accepted.

D. Solutions: Please write concise and clear solutions; you will get only a partial credit for correct solutions that are either unnecessarily long or not clear. Please refer to the Written Homework Guidelines for all the requirements. Piazza will also contain a complete sample solution.

E. Algorithms: Whenever you present an algorithm, your answer must include 3 separate sections. Please see Piazza for an example complete solution.

1. A precise description of your algorithm in English. \textbf{No pseudocode, no code.}
2. Proof of correctness of your algorithm
3. Analysis of the running time complexity of your algorithm

F. Collaboration: You are allowed to discuss ideas for solving homework problems in groups of up to 3 people but you must write your solutions independently. Also, you must write on your homework the names of the people with whom you discussed. For more on the collaboration policy, please see the course webpage.

G. Outside Resources: Finally, you are not allowed to use any material outside of the class notes and the textbook. Any violation of this policy may seriously affect your grade in the class. If you’re unsure if something violates our policy, please ask.
Important: Many of these questions will require you to draw trees. Hand-drawn trees will result in a 0 — you must \LaTeX your solutions. Please see Piazza @933 for instructions on how to draw trees on \LaTeX. Please start early! Also, don’t leave typing this assignment up to the last minute. Type your answers up as you go.

The use of any online resources or any visualizer (including those shown in class) is prohibited. You may not check your solutions with these tools, either. You may only use the lecture slides and the CLRS textbook.

1. [25pts - Splay Tree Basics]

(a) [3] In preparation for Halloween, you created a data structure that supports the operation Boo such that a sequence of \( n \) Boo’s takes \( O(n \lg n) \) time in the worst case. With this information, we can conclude that:

(i) The amortized time of a single Boo operation is \( \Theta( ) \).

(ii) The actual time of a single Boo operation could be as low as \( \Theta( ) \) and as high as \( \Theta( ) \).

Fill in all 3 spaces above inside the \( \Theta( ) \). No justification required.

(b) [5] Prove or disprove with a counterexample: if \( T \) is a BST with at least four nodes, then any node in \( T \) can be moved to a leaf position by an appropriate sequence of splay operations.

(c) [3] Given the following splay tree, draw the result of DELETE(7). Use the algorithm from slide 14 of the lecture slides. No justification necessary.

Note: After disconnecting the root during deletion, make sure you splay the maximum node in the left subtree \( T_L \); solutions which use any of the other methods described on slide 14 will be incorrect.

\begin{center}
\begin{tikzpicture}[level distance=1.5cm, sibling distance=1.5cm, grow=up]
  \node (root) {9}
    child {node (l1) {8}
      child {node (l2) {7}
        child {node (l3) {3}
          child {node (l4) {2}}
          child {node (l5) {4}}}
      }
    }
  ;
\end{tikzpicture}
\end{center}
(d) [3] Given the following splay tree, draw the result of Insert(4). Use the algorithm from slide 13 of the lecture slides. No justification necessary.

```
6
 /   \
2     10
 / \\
1   5 7 15
```

(e) [3] Given the following splay tree, draw the result of Find(7). Use the algorithm from slide 13 of the lecture slides. No justification necessary.

```
3
 /   \n2     7
 /   \n4     8
```

(f) [8] Given the following splay tree, draw the resulting splay tree after doing Find(4), Insert(8), and Delete(3). Use algorithms from slides 13-14 of the lecture slides. Please show your work (i.e., show the resulting tree after performing each operation).

**Note:** After disconnecting the root during deletion, make sure you splay the maximum node in the left subtree $T_L$; solutions which use any of the other methods described on slide 14 will be incorrect.
2. [25pts - Red-Black Tree Basics]

(a) [3] Consider a sequence of 14 insertions to a Red-Black tree.

   (i) What is the maximum number of edges on the longest path from the root to any leaf? No justification required.

   (ii) Draw a 14 node Red-Black tree that confirms your claim in (i).

(b) [5] Prove or disprove: in a red-black tree, it is impossible to have a black node with only one black child and no red child.

(c) [4] What is the largest and smallest possible number of internal nodes in a red-black tree with black-height $k$. Give a brief (max 3 sentence) justification for each.

(d) [5] Insert the following sequence into an initially empty red-black tree using bottom-up insertion from slide 15-16 of the lecture slides. Draw the resulting tree (no work required). Additionally, show the resulting tree if you inserted that same sequence into a BST without balancing. INSERT: 0, 8, 1, 7, 2, 6, 3, 5 in that order.

(e) [5] Given the following Red-Black tree, draw the resulting tree after calling DELETE(0), DELETE(11). Use the algorithm from slide 26-37 of the lecture slides. No justification necessary.

(f) [3] For each below, say whether it’s true or false with a 1 sentence justification.

   (i) In the worst case, a red-black tree insertion requires $O(1)$ rotations.

   (ii) In the worst case, a red-black tree deletion requires $O(1)$ node recolorings.

   (iii) A pre-order traversal on a red-black tree with $n$ nodes takes $\Theta(n \log n)$ time.

3. [10pts - Sarah & Co. Goes Viral] Because of Sarah & Co.’s immense success since it’s grand opening, Sarah has been kicking around the idea of opening up a few more boutiques in some of the world’s hottest shopping locations – such as Paris, London, and Milan. To start building some hype for the openings, Sarah wants to work on getting the word out about Sarah & Co., and she thinks that her high-profile customers can help.
Sarah decides that she wants to have a way to easily look up her customers based on the number of Instagram followers they have, so that she can figure out who she wants to post about Sarah & Co.

Having taken 121 several years back, Sarah thinks that a good way to do this would be to store her customers in a red-black tree \( S \), sorted by the number of Instagram followers they have (we can assume that no two customers have the same number of followers). Sarah wants this tree to have all of the regular red-black tree methods – \texttt{INSERT}, \texttt{DELETE}, and \texttt{FIND} – and also wants to include a method \texttt{NumCustomers}(x), which will return the number of Sarah & Co. clients that have more followers than \( x \), so that she can get a sense of how influential her current clients are.

In other words, \( S \)'s methods are defined as follows:

- \texttt{INSERT}(S, x): Insert a customer with \( x \) followers in \( S \)
- \texttt{DELETE}(S, x): Delete a customer with \( x \) followers from \( S \)
- \texttt{FIND}(S, x): Return the customer with \( x \) followers in \( S \)
- \texttt{NumCustomers}(S, x): Return the number of customer with more than \( x \) followers in \( S \)

Describe how Sarah should modify a standard red-black tree and implement \texttt{NumCustomers} so that all of these operations run in \( O(\lg n) \) time. If your modification changes how any of the other operations are implemented, please explain why the changes don’t affect the overall runtime.

4. [15pts - JTG Trees] A few weeks ago, some of the CIS 121 TAs were interviewing for jobs, and they were asked to implement a balanced BST. Unfortunately, they completely forgot how both Splay Trees and Red Black trees worked! In an attempt to impress the interviewers, they claimed they had invented a new type of balanced BST called a JTG tree (named after their glorious leader, John T. Geyer).

In addition to a key, left child pointer, and right child pointer, each node of a JTG tree also keeps track of the height of the subtree rooted at that node (the height of a leaf is 0). A JTG tree maintains the following invariant: if \( x \) is a node, then the height of the subtrees rooted at \( x.left \) and \( x.right \) differ by at most one.

In this problem, we will prove that the height of any JTG tree is \( O(\lg n) \).

(a) [7] Prove that a JTG tree with height \( h \) has at least \( F_h \) nodes where \( F_h \) is the \( h \)-th Fibonacci number. For example, \( F_0 = 1 \), \( F_1 = 2 \), \( F_2 = 3 \), \( F_3 = 5 \), etc (note that this is slightly different from the standard Fibonacci sequence, as we are ignoring the duplicate ones).

(b) [8] Using your result from part (a), prove that the height of a JTG tree is \( O(\lg n) \).
5. **[15pts - Lonely Children]** Define a “lonely child” as a node in a BST with no siblings. That is, \( x \) is a lonely child if \( x \) is the only child of its parent.

(a) [5] Recall the definition of a JTG tree from the previous problem. Prove or provide a counterexample to the following claim: the number of “lonely children” in a JTG tree is at most \( \frac{n}{2} \).

(b) [5] Prove or provide a counterexample to the following claim: if the number of lonely children in any BST \( T \) is at most \( \frac{n}{2} \), then the height of \( T \) is \( O(\lg n) \).

(c) [5] Prove or provide a counterexample to the following claim: if a BST \( T \) has \( \Theta(n) \) lonely children all of which are leaves, then the height of \( T \) is \( O(\lg n) \).

6. **[10pts - More Traversals]** Another way to do an in-order walk of a BST \( T \) is by calling \( \text{Minimum}(T) \), then calling \( \text{Successor()} \) \( n - 1 \) times. Prove or disprove: just like the recursive in-order traversal presented in lecture/recitation, this algorithm also runs in \( O(n) \) time.

**Feedback:** How long did you spend on this assignment? What did you think of it, and how can we improve written assignments in the future? Let us know anonymously at: [https://tinyurl.com/CIS121-18fa-hw-feedback](https://tinyurl.com/CIS121-18fa-hw-feedback)

We really appreciate your feedback!