CPSC 221: Theory Assignment 3

Posted: Sunday, March 19, 2017 @ 23:30 **Due**: Sunday, April 2, 2017 at 23:59

History of Non-Trivial Changes to this Document:

- Mar. 31 & Mar. 29 – clarifications or tips about 9(b)
- Mar. 20 – a brief comment about the usage of the number 8 in Q4

Submission Instructions

Submit your solutions using `handin`. You can write your solutions by hand and scan the pages or take pictures of them with your phone (but organize them according to the next paragraph); or better still: use a word processing package to typeset your solutions and produce a `.pdf` file.

Important: (1) Clearly label the question number of any file you submit. (2) Don't submit sideways or upside-down photos; these are a pain for the markers to work with. Use PDF to create a single document that can be read in the regular vertical way. Pretend that the marker is going to print your document on 8.5" x 11" paper, and will read down the page(s) in the normal way. (3) If using a photo, please don't use a high resolution because it can quickly consume a lot of space and upload/download time. (4) Be sure to include your name, CS userid, partner's name (if applicable), and partner's CS userid in your file. Two marks for this assignment will be awarded for complying with these rules.

To Submit: Copy the files that contain your solutions to the directory `~/cs221/theory3` in your home directory on an undergraduate machine. (You may have to create this directory using `mkdir ~/cs221/theory3`.) Then, run: `handin cs221 theory3` from your home directory.

We encourage you to work in pairs. **Be sure to include the names and ugrad CS login IDs of both partners on your solution, but only one partner should submit the assignment. Double-check your submission!** If your partner is submitting the assignment, confirm with him/her whether the submission was successful.

Late Submission Policy: The late penalty is 3% per hour (or portion thereof), with no late assignments being accepted after 9 hours. For example, if you hand in your program at 02:10 AM on the morning after the due date, then this is 3 hours late; so, you would lose 3 * 3% = 9% of the maximum possible mark.

Part 1: Hashing (These are deferred questions from Theory Assignment #2.)

1. Which of the following 3 choices would make the best hash function for an 11-digit account number (assuming a random digit between 0 and 9 is assigned for each of the 11 positions)? Briefly, justify your answer by analyzing and comparing the 3 hash functions given below: $h_i(k)$. We'll use open addressing with 1000 array cells, and we'll use linear probing to resolve collisions. There are 250 keys to hash.

   a) Hash function $h_1(k)$ = floor(sqrt(k)) % 1000

   b) Take the 4 digits in the positions 2, 4, 6, and 8 (offsets start at 0, like a C++ array); call them *a*, *b*, *c*, and *d*, respectively; concatenate them; and then compute $h_2(k)$ = (*abcd*) % 1000. Note: *abcd* does *not* mean *a*b*c*d*.

   c) $h_3(k)$ = ((the number of zeroes in k) + (the number of ones in k) + (the number of twos in k) + ... + (the number of nines in k)) % 1000.

2. Using a hash table with m = 11 locations and the hash function $h(k) = k$ % 11, show the hash table that results when the following integers are inserted in this order:

   26, 42, 5, 44, 92, 59, 40, 36, 12, 60, 80

   … if we assume that collisions are resolved using the following techniques. Note: If your collisions cannot be resolved within m tries, write down the failing case.

   a) Linear probing

   b) Quadratic probing

   c) Double hashing with the following secondary hash function:

      $h_2(x) = (2x)$ % 11, if this expression is non-zero
      $h_2(x) = 1$, if the expression in the previous line is zero

Part 2: Tree Traversals (This is a deferred question from Theory Assignment #2.)

3. If the preorder visitation of nodes (containing one-character variables and binary operators) in a binary tree is the following sequence: **a*+bc+*def* then:

   a) Draw its expression tree.

   b) What is its inorder sequence?

   Note that a binary operator like +, –, *, and / must have both a left child and a right child.

Part 3: Pigeonhole Principle

4. How many unique numbers must be selected from the set {1, 3, 5, 7, 8, 9, 10, 11, 13, 15} to guarantee that at least one *pair* of these numbers adds up to 16? Justify your answer. Note that if you include 8 in your list, you cannot use it twice (i.e., you cannot pick a pair of 8's).

5. How many integers from 1 to 100 must you pick in order to be sure that you get one that is divisible by 5? Justify your answer.

Part 4: AVL Trees

6. Draw an AVL tree, including nodes and pointers to children, after inserting each of the following items into an AVL tree. Remember, an AVL tree has the following properties:

   - Binary tree property: each node has ≤ 2 children
   - Balance property: for all nodes x, $-1 \leq \text{balance}(x) \leq 1$
   - Search tree property: all keys in a node's left subtree are smaller than the node's key, and all keys in a node's right subtree are larger than the node's key.

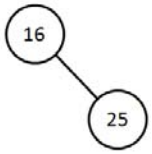
   Draw the AVL tree *after each key is inserted*. When necessary, draw the re-balanced AVL tree and declare which rotation was used for rebalancing. For a double-rotation, draw the AVL tree after the first rotation in addition to the final result.

   - Insert the following nodes (one at a time) into an initially empty AVL tree:

   16, 25, 29, 7, 12

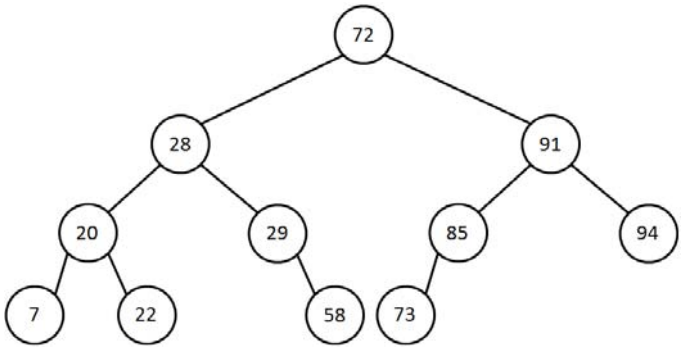
   - To get you started, the first two insertions have been completed for you.

   insert(16): 

   insert(25): 

   Continue with: insert(29).

7. Repeat the same process as in Question 6, but assume the AVL already contains the nodes shown below. Insert 89, and then 77.



Part 5: B+ Trees

8. Suppose we currently have an empty B+ tree root node that can hold up to 2 keys (and the same is true for all other nodes in the tree). We want to enter the following keys, *in this order*, into the B+ tree:

   110, 210, 121, 221, 213, 304, 320, 310

   Show the resulting structure, but not only the final result of the B+ tree, but anytime that a node splits. Anytime that you're not splitting a node, it is OK to keep adding entries to the non-full node (and keep the keys in order).

9. Suppose we want to build a B+ tree that has space for 200,000 *data entries* in its leaf pages. Each data entry is made up of a key and its corresponding data value. Let us assume the following specifications. Each page (leaf or internal) is 4096 bytes long. Each page holds three 8-byte pointers (parent, left sibling, right sibling) in addition to the bytes consumed by all of the keys and their accompanying 8-byte values. (In the case of leaf pages, these "values" are simply pointers that locate the full data record corresponding to the key (e.g., customer record for the given key.) We want to use an even number of data entries in the leaf pages. Internally, we also want an even number. The keys have unique values. For all of the following questions, show your work:

   a) Suppose each key is 56 bytes long, and suppose we fill the leaf pages to capacity (i.e., as much as possible). Compute the number of leaf pages that we need.

   b) Compute the number of internal pages, at each level, that we "need" (i.e., assume that you can fill the parents to capacity (to the maximum even number of keys)). This will result in a structure that has the fewest number of pages. Note, however, that the root can have as few as one key.