CSE326 Homework #4

Due: Wednesday July 20.

Note: Each time you are asked to give an algorithm or implement a function, you should give an explanation of how the algorithm works and why it is correct, as well as pseudocode.

1. (Recommended, but don't turn in.) Show the result of inserting 10, 15, 12, 3, 1, 13, 4, 17 and 8 into the following initially empty search tree data structures.

   (a) Binary search tree (unbalanced)

   (b) AVL tree

   (c) Splay tree

   (d) B-tree with M=3, L=2

   (e) B-tree with M=3, L=3

2. In this problem we'll look at adding an additional operation to the Search ADT. The operation is `findKth( tree T, integer k )`, which returns the node with *rank* k, i.e. the kth smallest node in the tree. (So `findKth( T, 1 )` is the standard `findMin()` function.) `findKth` should run in time $O(\log n)$, where n is the number of nodes in the tree. To efficiently implement the new Search ADT with a binary search tree data structure we'll need to store additional information at each node.

   (a) The first thing you might try is to store the rank of each node along with the key. How could you implement `findKth` with this information? What is the problem with this approach?

   (b) Instead, we'll store the number of descendants of each node. Given the rank of a node v and descendant information for all nodes, how can we determine the rank of v's left and right children? How can we determine the rank of the root node? Using this, give an implementation of `findKth`.

   (c) We're not done yet, since we haven't said how to keep the descendant information up to date. Describe how to change the `insert` and `delete` functions to maintain descendant information.

3. Let T_1 and T_2 be sets such that $\forall x \in T_1, y \in T_2, x < y$. Then we call $T_1 \cup T_2$ the *concatenation* of T_1 and T_2. If T_1 and T_2 are represented as binary search trees then the concatenation is easy to implement: simply find the max of T_1 and make it the new root, with the modified T_1 as the left child and T_2 as the right child. For AVL trees things are not so simple–the heights of the two trees could be very different, so the resulting tree won't be balanced. Instead we must somehow insert one tree inside the other in a way that maintains balance.

   Design an algorithm to concatenate two AVL trees. The algorithm should run in time $O(\log n)$, and in particular, should require only one single or double rotation to restore balance. Be sure to argue why your algorithm maintains the search property and the AVL property.

4. Imagine that you have been asked to to implement a new IRS database. There are 100,000,000 records, each of which take an average of 2,000 bytes. The records are keyed with a Social Security Number which is 4 bytes. The computer that you will be using has 2,048 byte pages and pages are addressed with 4 byte integers. Assume that a B-tree node completely as possible fills a page. This may means that leaves may hold more or less keys than internal nodes depending on what data is stored on the leaves. Assume that the computer has 16 MB of memory that is usable for storing all or part of a search structure. For the B-tree, disk addresses are used for pointers, and a 2 byte integer is used to keep track of the length of the active area in the B-tree node. The leaves of the B-tree contain pointers to the actual records. We assume that the nodes, other than the root, of the B-tree are about 3/4-th full on average.

   (a) Calculate the maximum number of children (M) that an internal node of the B-tree can have. Calculate the maximum number of entries (L) a leaf of the B-tree can have.

   (b) Calculate the height of the B-tree with that order so that all the 100,000,000 records can be accessed. About how many children does the root have.

   (c) Calculate how many levels of the B-tree can fit into the memory of the computer. Several full levels and the portion of one level will fit into memory.

   (d) Calculate how many disk accesses in the worst case are necessary to find a specific record using this search structure. How many on average?