Data Structures (IT205) 2015-16
Second Midsemester-semester Exam
16^{th} October, 2015

Time: 2 hours **marks: 80**

This question paper consists of 4 questions printed on two back-to-back pages. Check that your question paper is complete. Each question is worth 20 marks.

1. An incompetent programmer of a firm with data stored in a massive binary search tree is asked to write code and balance the tree to improve search time. Instead of implementing the standard rotation algorithm and running it at the nodes with imbalance, the programmer instead implements a routine which exchanges the left and right child pointers at those nodes. Thus the left and right subtrees get exchanged at nodes where this wrong implementation of rotation (let us call it flipping) is performed. The firm decides it is not worth undoing the work of this incompetent programer and instead decides to adopt his adhoc structure for maintaining this database.

   They decide that the left and right subtree roles can be decided arbitrarily at each node (i.e in some nodes the left subtrees contain all smaller key values and right subtrees contain larger key values; while in other nodes this convention is reversed).

   Write efficient algorithms for the following operations on this modified type of binary search tree:

   (a) Search for an element with a particular key field.

   (b) Find the maximum element of a subtree.

   (c) Find the successor of an element in the tree.

   You need to write pseudocodes for these three routines in the modified data structure and also analyse the runing times of these routines in the worst case.

2. Consider a binary search tree on 37 nodes. Normally the height of the tree is given by the formula

$$h(T) = 1 + Max\{h(T_{Left}), h(T_{Right})\}$$

   For what values of rank of root will the height of this binary search tree on 37 nodes depend only on the height of the right subtree (in the above formula the left subtree height will have no impact on the result for these values of the rank of root). Generalise your answer for any integer of the form $(2^k - 1) + k + 1$. This function takes the value 37 for $k = 5$.

3. Consider an array A containing the elements ten elements 50,90,10,80,60,100,30,20,40,70 in that order. Show the execution sequence (simulate the program) for the following operations.

   (a) Build it into a MAX-Heap using the routine which applies heapify at each internal node starting from the last internal node.

   (b) Increase the key value of the node in position 4 by 25 in the heap built in part (a) and restore the heap property. Node indexing begins with 1 and not 0.

   (c) Decrease the key value of the node in position 5 by 35 in the heap obtained from part (b) and restore the heap property.

   (d) Delete the node in position 3 from the heap resulting from part (c) and restore the heap among the nine remaining elements.

   (e) Insert a node with key value 44 in the nine node heap resulting from part (d) and restore the heap property.

4. In the lecture the following algorithm was discussed for AVL insert.

   (a) While descending down the tree in the search phase of search-and-insert, each node is marked as positive, negative or neutral in the direction of traversal (meaning if the subtree that direction has greater, smaller or equal height as the one in the other direction- direction here refers to left/right).

   (b) After the insert, according to the standard insert algorithm for binary search trees, we travel upwards along the insert path to the root and along the way encounter nodes of type positive, negative or neutral.

   (c) For every neutral node encountered, the algorithm simply augments its height by 1 and goes further up the tree.

   (d) When a positive node is encountered, a rotation is performed in the opposite direction, the heights of nodes whose heights potentially change are recomputed and stored and the procedure terminates.

   (e) When a negative node is encountered, the algorithm simply terminates.

   Translate this algorithmic description into a precise and **formal pseudocode** (line numbers, while statements, if then else for loops etc.) as has been done in lectures for various routines during the course. You do not need to give the code for the rotate operation, nor the search-and-insert part. The code of how to restore AVL properties as described in the algorithm is all you need to give. You may pass as argument the inserted node x, the tree T and assume there is a variable called $ROOT[T]$. Addidionally, you may assume the information on heights of nodes, the balance -1 for left is taller by 1, 0 for equal heights and +1 for right is taler by one is stored at each node. The terms positive negative and neutral used in the algorithm is only for description. The detail stored in the data structure is only left/right child pointers, parent pointers and the balance factor as an integer -1, 0, +1 at each node and the heaight at each node.

   Since the detailed description of what needs to be done is given here, the pseudocode must give all the changes necessary to be performed and must be precise. It is preferrable that you try what you want and write the final version neatly. Do not scratch and overwrite in the final version of your answer.