ESO 207 Midsem Exam

Submission Deadline: 8 pm, Oct. 22, 2020

Maximum Marks: 100

Instructions

- The exam is open book and open notes. You are however, allowed to discuss it within your team only.
- Collaborating outside your team will be considered cheating and is punishable.
- Use of online resources or internet, apart from accessing ESO207 course websites, is not permitted.
- Handwritten answers should be uploaded on mookit by the submission deadline.
- Only one submission per team is allowed. Any team member may upload file <team-name>.pdf using his/her roll number. Please include roll numbers of team members in the first page of your submission.
- If we receive more than one file from members of a team, we will grade only one randomly chosen file from those.
- In pseudo-code, try to make it understandable to those who would be reading it. Use intuitive names, as far as possible, and insert sufficient comments.
- There are marks for clarity, brevity, precision and neat work.
- One part of one question is marked as optional. You are not required to answer it. However, successfully answering it will get you some extra points. (not included in the stated maximum marks). I suggest that you try it only after you have finished rest of the paper.

- Numbers in parenthesis written at the beginning of each question are marks that question carries, broken-up parts wise.
- Figures are in a separate file.

Q1(20) For a set of n data items there may be several binary search trees (BST) of n nodes. Suppose you are given a binary tree T with n nodes with no data at any node, and a list L of n data items. Describe an algorithm $makeBST(T, L)$ which puts data items from L in nodes of T so that it is a BST. Your algorithm should not change the structure of T.

Write the pseudo-code of your algorithm. What is its time complexity? justify your answer.

Q2(7+10+10) A BST T is called left-linear if right child of all nodes in T is nil.

(a) Consider the BST in figure 1. Convert it into a left-linear BST S on the same set of nodes by applying a series of left-rotations (no other operation on tree is allowed). Show the tree obtained after each rotation.

(b) Show that any BST R, may be converted to a left-linear BST on the same set of nodes by applying a series of left-rotations only.

(c) Show that a left-linear BST R may be converted to *any* given BST U on the same set of nodes, by applying a series of right-rotations only.

(d)[Optional] Show that the transformation in (b) can be done in at most $n-1$ rotations, where n is number of nodes in the tree. Can you prove a bound for transformation in (c) also?

Q3(3+(5+10)+15) In our insertion and deletion operations on search trees, we modify the tree, thereby losing the original tree. Sometimes we need to maintain old versions of the tree also. This can be done by making a new copy of the existing tree each time an insertion or deletion is done. This however, is wasteful as we may not need to copy the entire tree, copying some nodes may be enough. As an example, figure-2 shows a tree T_1. It also shows tree T_2 obtained after inserting 75 into T_1. Only nodes with data 80, 70 and 60 from T_1 are copied into T_2.

(a) In this scheme, tree nodes do not have parent pointers. Can you say why?

(b) Try to figure out the general procedure from the example given and show result of the following insertion/deletion operations. In all cases, we preserve existing trees and create as few new nodes as possible.

(i) Suppose element 95 is inserted into tree T_2 of the example above, resulting into tree T_3. Show (representations of) T_1, T_2 and T_3 and links among them after this operation.

(ii) Starting from T_1 in the example given, first delete node with data 100 and call the resulting tree R_1. Now, from R_1 delete the node with data 80 to obtain R_2. Show (representations of) T_1, R_1 and R_2 and links among them.

(c) The scheme in this question may also be applied to RB trees. Consider RB tree, we have at the very end of the RB-insertion-example lecture ('lecture 14-Examples'). Call it S_1. Insert 55 into S_1. Show various stages of fix-up and how links change through it. In the end we should have both S_1 and S_2. Tree S_2 represents the RB tree obtained by inserting 55 into S_1 and has as few new nodes as possible.

Q4(20) In our implementation of red-black tree T, we wish to add a new field $T.bh$, which stores black height of the tree. Note that there is only one variable $T.bh$ for the whole tree, you are not allowed to add extra fields to tree nodes.

Point out the minimal changes, if any, we must do in the pseudo-codes for insertion, deletion and fix up procedures, written in the lecture notes, to correctly maintain $T.bh$. Asymptotic runtime of these procedures should not change. Give a brief justification for your answer.

—x-x-x—