Homework 4*

Data Structures
Fall 2019 CS203@IITG

---

(1) Analyze the time complexity of the deletion algorithm for B-tree (closely analyze while considering recursive deletes in subtrees).

Write the pseudocode for the join operation in the BST ADT for B-trees.

(2) Draw the 2-3-4 tree corresponding to Fig. 13.1 (a) page 317 of CLRS.

Draw the 2-3-4 tree just before inserting node z in Fig. 13.4 (a). As the algorithm inserts new node z and modifies the red-black tree in Figs. 13.4 (a)-(d), modify the corresponding 2-3-4 tree accordingly.

Between any 2-3-4 tree and its corresponding red-black tree representation, argue the latter could be better in terms of space usage.

(3) For the splay trees, show the amortized time complexity of the two-level rotation $leftrotate$-$rightrotate$ to bring a node x two levels upwards is at most $3(r'(x) - r(x))$. (The r' and r denote the same as used in class.)

Show each of the split, insert, and delete operations for splay trees take $O(\lg n)$ amortized time. (Hint: Consider the amortized cost of splay and change in potential due to the operation of interest.)

(4) In case of height-biased leftist heap (HBLH), formally prove the correctness of meld algorithm.

Prove/disprove whether the binary heap is a HBLH. Determine whether the meld algorithm designed for HBLH works for the binary heap.

Significantly, determine whether meld algorithm permits representing HBLH in an array (like in the case of binary heap) while achieving $O(\lg n)$ time for the meld operation.

(5) Prove the following: While traversing upwards in both the meld and delete algorithms of HBLH, the algorithm can always stop further traversal whenever it finds a node whose s values is not required to be changed.

In deleting an arbitrary key from a HBLH with n keys, prove that at most $O(\lg n)$ ancestors of $x.parent$ are traversed.

— more problems will be added —

---

*Prepared by R. Inkulu, Department of Computer Science, IIT Guwahati, India. http://www.iitg.ac.in/rinkulu/