Data Structures and Algorithms: Homework #6

Due on June 9, 2015 at 16:20

Instructors Hsuan-Tien Lin, Roger Jang

Tim Liou (b03902028)

6.1 Skip List, Binary Search Tree

(1) Do Exercise C-9.15 of the textbook.

====== Pending ======

(2) Do Exercise R-10.5 of the textbook.

====== Pending ======

(3) Do Exercise C-10.12 of the textbook.

====== Pending ======

6.2 Balanced Binary Search Trees

(1) Write a program `hw6_2` that reads 32 strings (of length at most 128 that can be compared lexicographically) line by line (each line containing one string) from `stdin` and inserts them to the AVL tree (`avl.c`), height-bounded binary search tree (`bst.c`), and Red-Black tree (`rb.c`). Please output the resulting trees (pre-order) to `stdout` with specific format.

====== Pending ======

6.3 Disjoint Set

(1) Prove that the disjoint-set forest with this heuristic yields a worst-case running time for find and union within $O(\log n)$.

====== Pending ======

(2) Suppose that you only need to output u rather than u and k for this problem. Write down the pseudo-code of an efficient algorithm based on the disjoint forest.

====== Pending ======

(3) Suppose that the prices of your friend u's games are stored in a balanced BST as keys, and you have access to the size and the sum of all keys of any subtree of the BST in an $O(1)$ time, write down the pseudo-code of an efficient algorithm for outputting k for the particular u.

====== Pending ======

(4) If we take the same heuristic as (1) and always insert the elements of the smaller BST into the bigger one, prove that processing all incidents of the first kind takes $O(n(\log n)2)$ time.

====== Pending ======

Problem 6.3 continued on next page. . .

(5) Write a program hw6 3 to solve the problem efficiently.

====== Pending (code part) ======