CS583 -- Programming Assignment -- Fall 2019 – due Tuesday, Nov. 26, 11:59PM

This concerns self-organizing binary search trees. We will work from a single survey paper, “Self-organizing Data Structures” by Albers and Westbrook, available on-line. In particular sections 1, 3.0, 3.2-5. You will code four algorithms. The first is the static optimal BST algorithm from the textbook (p. 402). Second will be the splay tree algorithm. Next compute the parity of the first 4 digits of your G number; if it odd do Move-to-Root, else do Single-Rotation. Next compute the parity of the last 4 digits of your G number; if it is odd do Dynamic-Monotone trees, else do WPL trees.

You will start each time with a randomly-built BST on the distinct keys {1, … , n}, n=1000. This is done using standard insert of keys into an initially empty tree. The order of the keys will be some random permutation; use the code on page 126.

Your access sequences of keys will be non-uniformly biased as follows:

Generate n integers a_i where each is a random integer from 1 to 100, and with c_i being the sum of the first i generated integers, c_0 = 0 and A = c_n.

Obtain a random integer j from 1 to A and “access” key k if c_(k-1) < j <= c_k.

Note all accesses will be successful searches. Access sequences will be 10,000 long. You will keep track of the number of accesses to each key, so that when you reach the end you can create the static optimal BST with those frequencies (instead of probabilities).

The survey paper is very clear on how costs are defined. Under no interpretation would you report run-times.

You will compute the empirical score of the static competitive ratio, in particular the average over 10 runs. You will do this for both of your memoryless self-organizing BSTs, and compare that to the theorems in the paper. (Of course, this has to be loose since we are not doing enough experiments to estimate the hidden constants. Further make observations on the empirically observed static competitive ratio of your state-based algorithm relative to the memoryless algorithms.

You will submit your code written in C++ or Java (otherwise ask) to the TA at yyang29@gmu.edu. You will also submit a text document. It will include a short one-page write-up along the lines indicated above; it will need some tables/graphs. Further you will include any instructions the TA will need to test your code. You should mention any decisions you made when you felt the assignment was ambiguous.

[a_k is a with a subscript k.]