Your name
CS 261, Spring 2021
Homework 7
Due Monday, May 17

1. (a) For the unrank pseudocode given in the talk slides, there is no "if node == None" base case, so the algorithm will crash if node is None. Assuming that this subroutine is called with the node being the root of a non-empty binary search tree with n nodes, which values of the argument i will cause it to crash in this way?

(b) For a binary search tree with n nodes, which values of i will cause rank(unrank(i,root),root) to return the same number i?

(c) For a binary search tree with n nodes, which values of x will cause unrank(rank(x,root),root) to return the same number x?

■

2. The average of two values is not an associative operation, so range averaging is not directly a decomposable range search operation. Suppose that, nevertheless, we wish to perform range averaging for dynamic sets of numbers. Describe a way of associating a value with each number, and defining an associative binary operation on these values, so that the average can be computed by using a range search data structure.

(The value associated with a number can be something more complex than just a single number, like an array or tuple, but it should have a total size that is constant, independent of the size of the data set. Hint: Use a sketch.)

■

3. One of the steps of the lower bound proof for range sums shows that they can be simplified to an equivalent problem, prefix sums. Intuitively, this does not work for range minima and prefix minima, because the operation of taking a minimum of two elements does not have an inverse. Prove that this step does not work for range minima, by finding two sequences of numbers for which all prefix minima are the same (the prefix minimum of the first i elements of one sequence equals the prefix minimum of the first i elements of the other sequence, for all i) but for which not all range minima are the same.

■

4. The lecture notes describe fractional cascading for successor queries in a collection of sorted lists without any repetitions (each item can only appear in a single list). Suppose that you want to apply the same technique to sorted lists that might have repetitions: each item can only appear once in a single list, but it might appear in multiple lists. Describe a way of modifying the structure so that it works in this more general case. (The O-notation for space and for query times should be the same as in the no-repetition case. There are multiple different ways of doing this; you only need to describe one.)

■