University of Waterloo
CS240E, Winter 2024
Assignment 3

Due Date: Tuesday, March 5, 2024, at 5pm

Be sure to read the assignment guidelines (`https://student.cs.uwaterloo.ca/~cs240e/w24/assignments.phtml#guidelines`). Submit your solutions electronically as individual PDF files named a3q1.pdf, a3q2.pdf, ... (one per question).

Ensure you have read, signed, and submitted the **Academic Integrity Declaration** AID02.TXT.

Grace period: submissions made before 11:59PM on March 5, will be accepted without penalty. Please note that submissions made after 11:59PM **will not be graded** and may only be reviewed for feedback.

Question 1 [6+6=12 marks]

Consider the following algorithm to find the minimum in a binary search tree.

Algorithm 1: *findMin*(root r)

```
if (r is null) then return "empty tree"
while r.leftChild != null do  r ← r.leftChild
return r.key
```

Let $T^{\text{avg}}(n)$ (for $n \geq 0$) be the average-case number of executions of the while-loop in *findMin* for a tree with n nodes. Here the average is taken over all binary search trees that store $\{0, \ldots, n-1\}$, and $T^{\text{avg}}(0) = T^{\text{avg}}(1) = 0$.

a) Show that for $n \geq 2$ we have $T^{\text{avg}}(n) \leq 1 + \frac{1}{C(n)} \sum_{i=0}^{n-1} C(n-i-1)C(i)T^{\text{avg}}(i)$, where $C(n)$ is the number of binary search trees that stores $\{0, \ldots, n-1\}$. Be as precise as we were in class for *avgCaseDemo*.

b) Show that $T^{\text{avg}}(n) \in O(\log n)$. (We recommend that you show $T^{\text{avg}}(n) \leq 2 \log n$, and that you consider a 'good case' where the left subtree has size at most $n/2$.) You may use without proof that $C(n) = \sum_{i=0}^{n-1} C(i) \cdot C(n-i-1)$, and you may assume that n is divisible as needed.)

Question 2 [3+5=8 marks]

Recall that the `Selection` problem receives as input a set of n items and an integer k with $0 \leq k \leq n-1$ and it must return the item that would be at $A[k]$ if the items were put into an array A in sorted order.

1. Argue that any comparison-based algorithm for the Selection problem on n keys must have $\Omega(\log n)$ worst-case time.

2. Let T be an scapegoat($\frac{2}{3}$)-tree that stores n items. Argue that `Selection`(T, k) can be done in $O(\log n)$ time.

Question 3 [5 marks]

Let S be a skip list with $n \geq 4$ items. Assume that the lists $S_0, S_1, \ldots, S_h$ of S have the following property for all $0 \leq i < h$.

$$\text{If } |S_i| = 1 \text{ then } |S_{i+1}| = 0. \text{ If } |S_i| > 1, \text{ then } |S_{i+1}| \leq \sqrt{|S_i|}.$$

What is the maximum possible value of h, relative to n? For full marks, you should give an exact bound (no asymptotics), make no assumptions on the divisibility of n, and show that your bound is tight for infinitely many some values of n. (But part-marks may be given otherwise.) Justify your answer.

Question 4 [3 marks]

Let A be an unordered array with n distinct items $k_0, \ldots, k_{n-1}$. Give an asymptotically tight Θ-bound on the expected access-cost if you put A in the optimal static order for the following probability distribution:

$$p_i = \frac{1}{(i+1)H_n} \text{ for } 0 \leq i \leq n-1 \text{ where } H_n = \sum_{j=1}^{n} \frac{1}{j}.$$

For example, for $n = 4$ we have $H_4 = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} = \frac{25}{12}$ and the items would have access probabilities $\langle \frac{12}{25}, \frac{6}{25}, \frac{4}{25}, \frac{3}{25} \rangle$.

Question 5 [3+3+8=14 marks]

a) In lecture, we only look at the key in a van Emde Boas tree. Give a way to efficiently manage values. Explain how to support the required operations (min, max, search, predessor, successor, insert, delete), and analyze the space requirements.

b) A vEBT uses $\Theta(u)$ space regardless of the number n of items that are actually stored. We would like to use less space and keep the worst-case time of $O(\log \log u)$ for each operation. To do so, one could replace the array cluster by a structure storing only the non-empty cluster. Argue that a BST-based dictionnary is not an appropriate implementation.

c) Propose an adapted vEBT structure that uses only $O(n)$ space and supports the operations in *expected* $O(\log \log u)$ time. You should explain your changes, how they impact the vEBT operations, and the new operations you might need.