University of Waterloo
CS240 Winter 2024
Assignment 3

Due Date: Tuesday, March 5 at 5:00pm

Please read `https://student.cs.uwaterloo.ca/~cs240/w24/assignments.phtml#guidelines` for guidelines on submission. **Each question must be submitted individually to MarkUs as a PDF** with the corresponding file names: a3q1.pdf, a3q2.pdf, ... , a3q6.pdf . It is a good idea to submit questions as you go so you aren't trying to create several PDF files at the last minute.

Late Policy: Assignments are due at **5:00pm**, with the grace period until 11:59pm.

Notes: Logarithms are in base 2, if not mentioned otherwise.

Question 1 [2+3+3=8 marks]

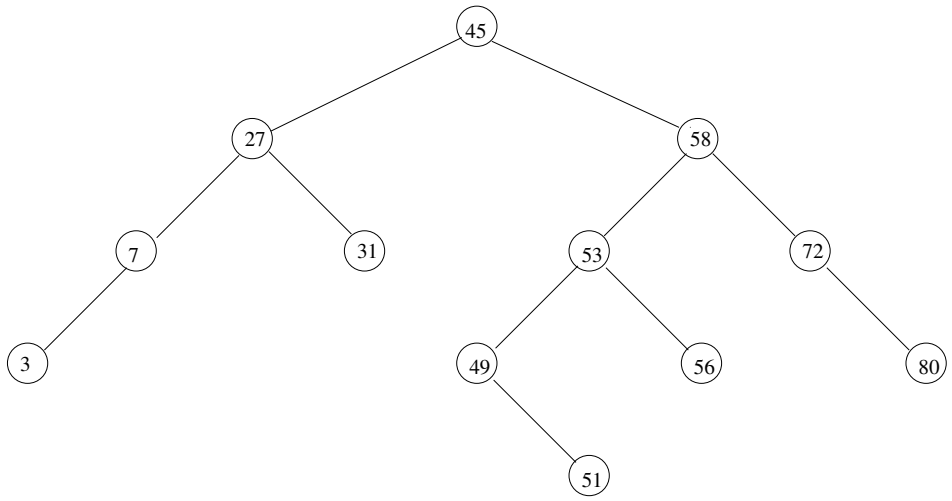

Figure 1: AVL tree of problem 1.

a) Consider the AVL tree shown in Figure 1. Draw the tree again by adding the balance factors to all nodes.

b) Consider the AVL tree shown in Figure 1. Draw the tree after performing operation insert(5). Draw the intermediate trees.

c) Consider the AVL tree shown in Figure 1. Draw the tree after performing operation delete(27). Swap with the successor. Draw the intermediate trees.

Question 2 [2+2+2+5=11 marks]

Define a family $(T_h)_{h\geq -1}$ recursively in the following manner: T_{-1} is empty and T_0 is a single node; to form T_h, we start with a single node and take a copy of T_{h-2} and a copy of T_{h-1} as the left and the right children of the root, respectively. Note that in this question, we do not distinguish between single and double rotations, i.e. when we say a 'rotation' we mean any rotation (single or double).

a) For $h \geq 0$, what is the height of T_h? Prove your claim by induction.

b) Prove that for $h \geq 0$, T_h is an AVL tree. Prove your claim by induction.

c) Which leaves of T_3 require $\lfloor 3/2 \rfloor = 1$ rotation upon their deletion? Pick one and show the resulting tree.

d) For $h \geq 0$, prove by induction that there is a node in T_h s.t. deletion of that node requires $\lfloor h/2 \rfloor$ rotations and upon deletion the resulting tree has height $h-1$.

Question 3 [4+2=6 marks]

a) Draw a diagram of a skip list starting with an empty one and inserting the seven keys $67, 28, 64, 66, 60, 81, 49$. Use the following coin tosses to determine the heights of towers (note, not every toss is necessarily used):

$$T, T, H, H, T, H, T, H, H, T, H, H, T, T, H, T, H, H, T, T, H, H, H, T, \ldots$$

It is sufficient to show the final skip list.

b) Suppose we use a biased coin to build a skip list, and probability of H (heads) is $1/3$. What is the probability that every tower in a skip list storing n entries will have height strictly larger than 4?

Question 4 [2+(2+2)+4 =10 marks]

For this problem, we use a skip list to store key-value pairs (KVPs) with real-numbered keys and values which can be compared to each other. Keys are unique. For skip list node v, we access the node on the same level with $v.after()$ and the level below with $v.below()$. We access the key of node v with $v.key()$, and if $v \in S_0$, we access its value with $v.value()$. We call a node v of the skip-list *not-final* if $v.key() \neq +\infty$. Assume the front sentinel node in S_0 stores $-\infty$ for the value.

a) Let $i \geq 1$, let v be a not-final node in S_i and let $w = v.after()$. Show that the expected number of nodes between $v.below()$ and $w.below()$, including $v.below()$ and $w.below()$, is $O(1)$.

b) Let v be a not-final node in S_i. Define w^v to be a node in S_0 s.t. $v.key() \leq w^v.key() < v.after().key()$ and $w^v.value() \geq q.value()$ for any $q \in S_0$ with $v.key() \leq q.key() < v.after().key()$. We wish to modify skip list so that each non-final node $v \in S_i$ maintains an attribute $v.maxVal$ equal to $w^v.value()$. This is achieved with two steps below. You can use the result in part (a) even if you did not prove it.

i) Develop an algorithm $fixFromBelow(v)$ which takes not-final node $v \in S_i$ as an input and sets $v.maxVal$ correctly. You can assume that if $i > 0$, then $z.maxVal$ are correct for all nodes z in S_j with $j < i$. The expected running time of $fixFromBelow(v)$ must be $O(1)$.

ii) After each *insert* and *delete* in skip list, $v.maxValue$ of some nodes becomes incorrect and must be fixed. Explain on which nodes and in which order you have to call $fixFromBelow$ developed in part (i) to restore $v.maxValue$ attributes to correct values after insert and after delete. The expected running time of fixing $v.maxValue$ attributes must be $O(\log n)$.

c) Given key y, explain how to find KVP (k, v) in the skip list with $k < y$ and the largest v, where the skip list was modified as in part (b). In other words, out of all KVP with keys less than y, find and return the one with the largest value. Even if you did not solve (b), you can assume the result in (b) is available to you. The expected running time must be $O(\log n)$.

d) (Just for fun, not graded) Given keys $x < y$, explain how to find KVP (k, v) in the skip list with $x \leq k < y$ and the largest v, where the skip list was modified as in part (b). In other words, out of all KVP with keys in between x (inclusively) and y (exclusively), find and return the one with the largest value. Even if you did not solve (b), you can assume the result in (b) is available to you. The expected running time must be $O(\log n)$.

Question 5 [2+2+(2+2)=8 marks]

a) Consider the list of keys:

[1 2 3 4 5 6 7 8 9 10]

and assume we perform the following searches:

10, 7, 2, 2, 4*, 1, 2, 1, 2, 1*, 1, 7, 1, 9*

Using the move-to-front heuristic, give the list ordering after the starred ($\star$) searches are performed.

b) Repeat part (a), using the transpose heuristic instead of the move-to-front heuristic.

c) You are given a dictionary D implemented using the Move-to-Front heuristic, with n distinct keys, stored as an unordered linked list L. Suppose you want to perform a sequence of n searches on keys present in D where the number of different keys searched for is $\sqrt{n}$. What is the best and worst-case runtime to perform all n searches? Use Θ notation. Briefly explain.

Question 6 [2+4=6 marks]

a) Illustrate interpolation search on array $A = [0, 10, 100, 1000, 10000]$ and search key 10. To illustrate the search, it is enough to show the sequence of values computed for m.

b) Assume that A is an array of size n with $A[i] = ai^2 + b$ for $0 \leq i \leq n-1$, where a, b are some numbers. Recall that interpolation search used the formula $m = l + \left\lfloor \frac{k-A[l])}{A[r]-A[l]} \cdot (r-l) \right\rfloor$ to determine the index m. Define a different formula for m such that for any key k that is in the dictionary, and any choice of l and r, the new formula for m gives exactly the index where k is stored. You can assume the key k is in between l and r.