University of Waterloo
CS240 - Fall 2018
Assignment 3

Due Date: Wednesday October 31 at 5pm

Please read `http://www.student.cs.uwaterloo.ca/~cs240/f18/guidelines.pdf` for guidelines on submission. All problems are written problems; submit your solutions electronically as a PDF with file name `a03wp.pdf` using MarkUs. We will also accept individual question files named `a03q1w.pdf`, `a03q2w.pdf`, `a03q3.pdf`, `a03q4w.pdf`, `a03q5w.pdf`, `a03q6w.pdf`. if you wish to submit questions as you complete them.

There are 75 possible marks available. The assignment will be marked out of 70.

Problem 1 [5+6+6=17 marks]

You have found a treasure chest filled with n coins, some of which are genuine and some of which are counterfeit. There is at least one genuine coin and at least one counterfeit coin in the chest. All genuine coins weigh the same and all counterfeit coins weigh the same, but the counterfeit coins weigh less than the genuine coins. Your task is to separate the genuine coins from the counterfeit coins. To accomplish this you will compare the weight of pairs of subsets of the coins using a balance scale. The outcome of one weighing will determine that each subset of coins weighs the same, or that one or the other subset of coins weighs more.

a) Give a precise (not big-Omega) lower bound for the number of weighings required in the worst case to determine which coins are genuine and which are counterfeit.

b) Describe an algorithm called `FindGenuine` to determine the genuine coins when n = 4. Use the names C1, C2, C3, C4 for the four coins, and the function

$$CompareWeight\left(\{first\_subset; second\_subset\}\right),$$

which returns either "first weighs more", "second weighs more", or "both weigh the same". Your function should return the set of genuine coins.

Give an exact worst-case analysis of the number of weighings required by your algorithm. For full marks, this should match exactly the lower bound from Part (a) when $n = 4$.

c) Describe an algorithm to determine the genuine coins, for any n. Use the names C1, C2, ... , Cn and the *CompareWeight* subroutine from Part (b). Show that your algorithm is asymptotically optimal, meaning that the big-O cost should match the lower bound from Part (a).

Problem 2 AVL Trees [5+5+5=15 marks]

As discussed in class, it is possible to implement AVL trees if the nodes store only the balance factor $\{-1, 0, 1\}$ at each node instead of the height of the subtree rooted at the node.

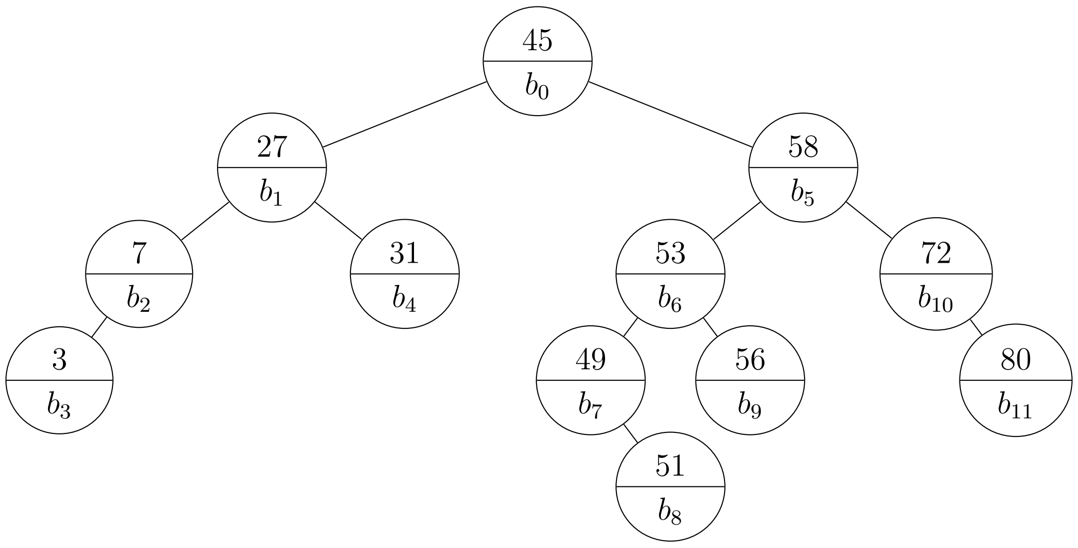

Figure 1: AVL tree of problem 1.

a) Consider the AVL tree shown in Figure ??. Draw the tree again, replacing the b_* with the actual balance factors.

Perform the operation Insert(5) on the tree. Draw the tree before each single or double rotation is performed, with the balance factors updated up until any call to fix is required. Draw the final tree with balance factors.

b) Consider the tree in Figure ??. Perform the operation delete(27) on the tree, swapping with the inorder successor. Draw the tree before any single or double rotation is performed, with the balance factors updated up until any call to fix is required. Draw the final tree with balance factors.

Problem 3 Height of an AVL Tree [5 marks]

Describe an efficient algorithm for computing the height of a given AVL tree. Your algorithm should run in time $O(\log n)$ on an AVL tree of size n. In the pseudocode, use the following terminology: T.left, T.right, and T.parent indicate the left child, right child, and parent of a node T and T.balance indicates its *balance factor* (-1, 0, or 1). For example if T is the root we have T.parent=nil and if T is a leaf we have T.left and T.right equal to nil. The input is the root of the AVL tree. Justify correctness of the algorithm and provide a brief justification of the runtime.

Problem 4 AVL-2 trees [3+4+3+3=13 marks]

We consider a modified version of AVL trees, where the height difference between the left and right subtrees of any node is in $\{-2, -1, 0, 1, 2\}$ instead of $\{-1, 0, 1\}$. These are called AVL-2 trees. For $i \geq 0$, let m_i be the minimum number of nodes of an AVL-2 tree with height i.

a) For $i = 0, \ldots, 5$, determine m_i and give an example of an AVL-2 tree with m_i nodes.

b) Find a recurrence relation for m_i and give its initial conditions.

c) Using your recurrence, prove that $m_i \geq 2^{i/3}$ by induction on i.

d) Prove that the height of an AVL-2 tree with n nodes is $\Theta(\log(n))$.

Problem 5 Skip Lists [6+6+6=18 marks]

a) Starting with an empty skip list, insert the seven keys $54, 15, 51, 53, 47, 68, 36$. Draw your final answer as on Slide 7 in Module 5. Use the following coin tosses to determine the heights of towers (note, not every toss is necessarily used):

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

b) The worst case time for searching in a singly linked list is $\Theta(n)$. Now consider a variation of a skip list which has fixed height $h = 3$ even though n can become arbitrarily large. Level S_0 contains the keys $-\infty, k_1, k_2, \ldots, k_n, \infty$. Level S_3 contains only $-\infty$ and ∞. Describe subsets of keys that should be included in levels S_1 and S_2 so that searching in the skip list has worst case cost $\Theta(n^{1/3})$. Provide justification for the runtime of your skip list.

c) Consider a skip list in which we build new towers as follows:

> When adding an element to the skip list, we flip two coins at the same time, until we see at least one tail. The number of times we toss both coins and obtain two heads is the height of the tower.

Using the method for building heights described in the above quote, derive the expected height of any single tower (not the entire skip list).

Problem 6 Tries [2+2+2+2+2+2=12 marks]

a) Draw the trie on the following five strings (include edge labels for clarity): 0001\$, 1001\$, 1011\$, 010\$, 1000\$.

b) Draw the compressed trie on the following five string: 100\$, 0110\$, 01110011\$, 01110101\$, 01110100\$

c) Draw the result of inserting 10\$ into the compressed trie shown on Slide 17 of Module 6.

d) Draw the result of deleting 01001\$ fron the compressed trie shown on Slide 17 of Module 6.

e) Draw the result of inserting the two strings `so$` and `b$` into the compressed multi-way trie shown on Slide 22 of Module 6.

f) Draw the result of deleting `soul$` from from the compressed multi-way trie shown on Slide 22 of Module 6.