University of Waterloo
CS240 - Fall 2022
Assignment 3

Due Date: Wednesday November 2 at 5pm

Please follow the guidelines for submission on the course webpage.

There are 64 possible marks available. The assignment will be marked out of 60.

Problem 1 AVL Tree Operations [2+5+5=12 marks]

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

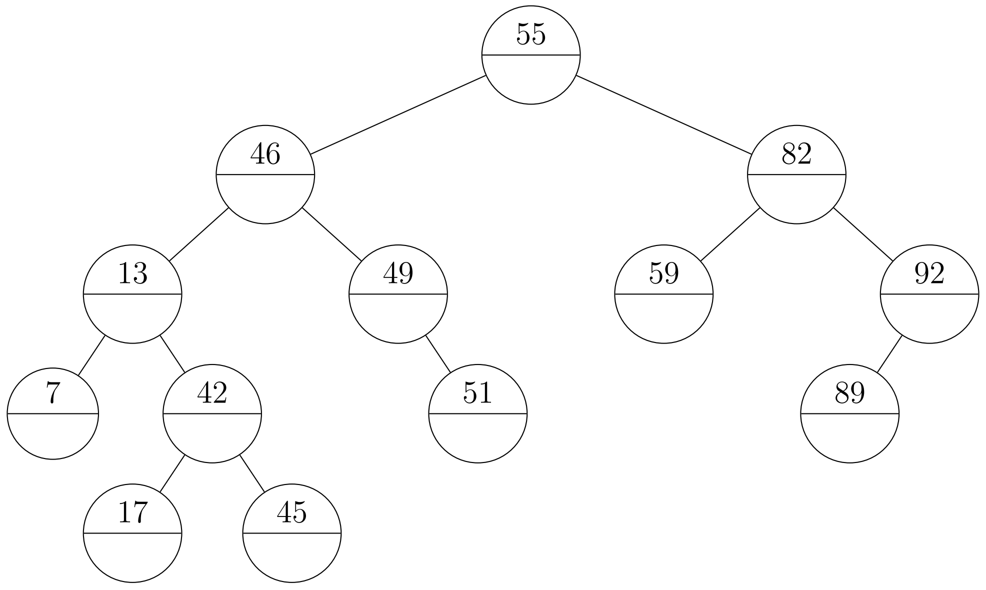

Figure 1: Binary tree T of problem 1

a) Show that the tree T in Figure 1 is an AVL tree by writing in the balance factor in the lower half of each node.

b) Show the process of inserting a KVP with key 29 into the tree T in Figure 1.. Specifically, draw the tree, with balance factors, after each call to restructure.

c) Show the process of deleting key 49 from the *original* tree T in Figure 1. Specifically, draw the tree, with balance factors, after each call to restructure.

Problem 2 AVL Trees [4+4=8]

Consider the binary tree T in Figure 2.

a) Draw the tree that results by calling restructure. Note that the balance factors are shown in the lower part of the nodes.

b) Draw the balance factors in the new tree.

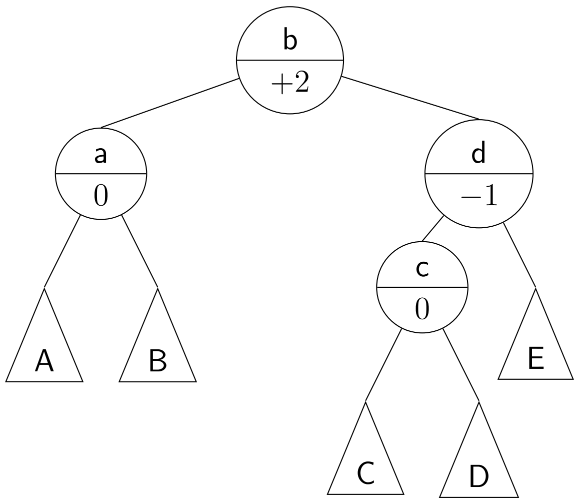

Figure 2: Binary tree T of problem 2.

Problem 3 Height of an AVL Tree [6 marks]

Describe an algorithm for computing the height of a given AVL tree in $O(\log n)$ time 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 Skip Lists [6+6=12 marks]

a) Starting with an empty skip list, insert 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):

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

Note: Should you desire to use LaTex to draw the skip list, Figure 3 gives an example of how to draw a skip list using the `tikz` package.

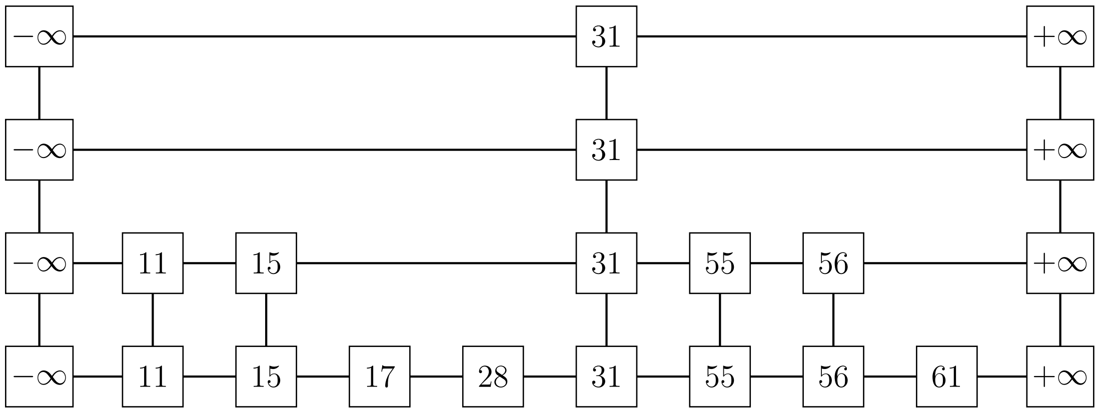

Figure 3: Example of skip list drawn using the `tikz` packge.

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.

Problem 5 Tug of War [6+7+7=20 marks]

A tug of war is a contest in which two teams of players pull on a rope in opposite directions. The team with the greater combined strength wins. (We assume that strengths are perfectly additive, and that there is no element of chance.)

For this problem, you are given n players, at least one of whom is *strong* and at least one of whom is *weak*. All strong players have exactly the same strength, and all weak players similarly have exactly the same strength. The task at hand is to determine which players are strong and which are weak. Your tool to determine this is to assign players to teams and have contests. The outcome of a single contest can either be a tie, one team winning, or the other team winning.

a) Using the decision tree approach, derive a precise expression (not big-Omega) of the lower bound for the number of contests required in the worst case to determine which players are strong and which are weak.

b) Describe an algorithm called *find-strong* to determine the strong players when $n = 4$. Use the names P_1, P_2, P_3, P_4 for the four players, and the function

$$contest\Big(\{first\_team\}, \{second\_team\}\Big),$$

which returns either "first wins", "second wins", or "tie". Your function should return a set or list of the strong players.

Give an exact worst-case analysis of the number of contests required in 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 strong and weak players, for any n. Use the names $P_1, P_2, \ldots, P_n$ and the *contest* 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 6 Tries [2+2+2=6 marks]

In this question, when drawing tries, follow the convention that \$ is always to the left of 0, and 0 is always to the left of 1.

a) Draw the standard trie on the following four strings (include edge labels for clarity): 00001\$, 0010\$, 0011\$, 001\$.

b) Draw the compressed trie on the following four strings: 00001\$, 0010\$, 0011\$, 001\$.

c) Draw the result of inserting 011\$ into your answer to part (b).