Problem 1 Quick-Select [6 marks]

When doing Quick-Select and Quick-Select, it is desired to have a good pivot which is almost in the middle of the sorted array. When doing the average-case analysis of Quick-Select, we considered a good and a bad case; the good case happened when the pivot was among the half middle items of the sorted array, i.e., we had \( n/4 \leq i < 3n/4 \) (\( i \) is the index of pivot in the partitioned array). In our analysis, we provided an upper bound for the time complexity of the algorithm in the good case and showed that \( T(n) \leq T(3n/4) + cn \) in these cases. Since the good case happened with probability \( 1/2 \), we could prove that the algorithm runs in linear time on average (see the recursion slide 10 of lecture 6).

Change the definition of the good case and assume the good case happens when we have \( n/5 \leq i < 4n/5 \). Provide an upper bound for \( T(n) \) and use that to show that Quick-Select runs in \( O(n) \). Hint: start by calculating the probability of good case and bad case happening.

Problem 2 Median-of-Three Algorithm [6+6=12 marks]

Consider a variant of Median-of-Five algorithm in which, instead of partitioning input into \( n/5 \) blocks of size 5, we partition the input into \( n/3 \) blocks of size 3.

a) Follow the same steps as slide 9 of lecture 7 to derive a recursive formula for the time complexity \( T(n) \) of this algorithm.

b) Try to solve the recursion by guessing that \( T(n) \in O(n) \). Follow the same steps as slide 9 and indicate whether we can state \( T(n) \in O(n) \).
Problem 3  AVL Trees [6+6+6+8=26 marks]

This problem will concern operations on the AVL tree $T$ shown in Figure 1.

a) Show that $T$ is an AVL tree by writing in the balance at each node.

b) Draw the tree after performing operation $\text{insert}(33)$. Indicate any rotations that are required at each step.

c) Draw the original tree after performing operation $\text{delete}(10)$. Swap with its predecessor. It suffices to just draw the final tree.

d) Draw the original tree after performing operation $\text{delete}(10)$. This time, swap with its successor. It suffices to just draw the final tree.

![Figure 1: AVL tree of Problem 3.](image)

Problem 4  More AVL Trees [8+8=16 marks]

a) We define $\textit{foo trees}$ as follows. An foo tree is a binary search tree where for every node, the heights of the left and right subtree differ by at most 2. Prove that a foo tree with $n$ nodes has height $O(\log n)$.

b) 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=$\text{nil}$ 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.
Problem 5  Skip Lists [7+6=13 marks]

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


b) Consider a skip list in which we build new towers with probability 1/4.

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 probability for tower heights described in the above quote, derive the expected height of a tower.