All problems are written problems; submit your solutions electronically as a PDF files via UMlearn. There are 73 marks available. The assignment will be marked out of 70. Please read [http://www.cs.umanitoba.ca/~kamalis/comp3170/info.pdf](http://www.cs.umanitoba.ca/~kamalis/comp3170/info.pdf) for guidelines on academic integrity.
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.

We showed in the class that for the average cost of selection algorithm on an array of size $n$, we have

$$T(n) \leq cn + \frac{1}{n} \left( \sum_{j=0}^{i-1} T(n - j - 1) + \sum_{j=i+1}^{n-1} T(j) \right)$$

Assuming $n/5 \leq j < 4n/5$ (when pivot is good), we will have $n - j - 1 < T(4n/5)$ and $j - 1 < 4n/5$. Consequently, $T(n - j - 1) < T(4n/5)$ and $T(j - 1) < T(4n/5)$. Note that $4n/5 - n/5 = 3n/5$, i.e., with probability $3/5$, the pivot is good and with probability $2/5$, it is bad. From the above equation, we get:

$$T(n) < cn + \frac{3}{5} \cdot T(4n/5) + \frac{2}{5} \cdot T(n)$$

$$T(n) < \frac{5}{3}cn + T(4n/5)$$

$$T(n) < \frac{5}{3}cn + \frac{5}{3} \times \frac{4}{5}cn + \frac{5}{3} \times \frac{16}{25}cn + \ldots + d$$

$$T(n) < d + \frac{5}{3}cn \sum_{i=0}^{\infty} \left( \frac{4}{5} \right)^i \in O(n)$$

**Marking Scheme:** 4 marks for showing the correct recursion and 2 marks for solving the recursion correctly.
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.

Assume \( x \) is the selected median-of-medians. So, half of blocks have their medians smaller than \( x \); each of these blocks have one element smaller than their median (and hence smaller than \( x \)). So, in total, there are at least \( \frac{1}{2} \cdot \frac{n}{3} \times 2 = \frac{n}{3} \) items smaller than \( x \). Similarly, there are at least \( \frac{n}{3} \) items larger than \( x \). Consequently, the size of recursion can be at most \( 2 \frac{n}{3} \). Assume \( T(1) = d \). For \( n > 1 \), we can write

\[
T(n) \leq T(n/3) + cn + T(2n/3)
\]

Marking Scheme: 3 marks for correct reasoning and inclusion of \( T(2n/3) \); 3 mark for other elements of recursion.

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 \( T(n) \in O(n) \).

Let’s guess \( T(n) \in O(n) \) and use strong induction to prove it. We should prove there is a value \( M \) s.t. \( T(n) \leq Mn \) for all \( n \geq 1 \). For the base we have \( T(1) = d \leq M \) as long as \( M \geq d \). For any value of \( n \) we can state:

\[
T(n) \leq T(n/3) + T(2n/3) + cn \quad \text{(from above recursion)}
\]

\[
\leq M \cdot n/3 + M \cdot 2n/3 + cn \quad \text{(induction hypothesis)}
\]

\[
= (M + c)n
\]

Note that we cannot sow that \( (M + c) \leq M \) for any value of \( M \). So, following the same steps does Not give us the same result, i.e., we could Not prove that \( T(n) \in O(n) \).

Marking Scheme: At most 2 mark if a proof was claim. Otherwise, 6 marks for getting to \( (M + c) \leq M \); partial marks \( \leq 4 \) if failing in the proof on some other part.

---

1This algorithm’s complexity is in fact \( O(n \log n) \); remember to use a value \( \geq 5 \) for median of median algorithms.
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. See Figure 1

**Marking Scheme:** 2 marks if there is no mistakes; 1 mark if there is less than 3 mistakes; 0 marks otherwise.

![Figure 1](image)

Figure 1: The answer to problem 1-a

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

- **Left-left scenario; a right rotation is applied.** See Figure 2

**Marking Scheme:** 2 marks for insertion to the correct spot; 2 marks for correct final tree; 2 mark for mentioning the type of applied rotation.

![Figure 2](image)

(a) Where the node 33 is inserted.  
(b) The final tree.

Figure 2: The answer to problem 1-b.
c) Draw the original tree after performing operation \texttt{delete(10)}. Swap with its predecessor. It suffices to just draw the final tree.

\textbf{Marking Scheme:} 2 marks for replacing with the correct node; 4 marks for the correct final tree (no need to mention the type of applied rotation or the intermediate tree).

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{tree1.png}
\caption{The answer to problem 1-c.}
\end{figure}

\begin{enumerate}
\item Node 10 is replaced by 8 (its predecessor).
\item Right-left scenario.
\item Final tree.
\end{enumerate}

\textbf{Figure 3:} The answer to problem 1-c.

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

\textbf{See Figure 4.}

\textbf{Marking Scheme:} 2 mark is for replacing with the correct node, and 4 marks for the correct final tree (no need to mention the type of applied rotation or the intermediate tree).
Problem 4  More AVL Trees [8+8=16 marks]

a) We define 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 $O(\log n)$.

Let $N(h)$ denote the number of nodes in a foo tree of height $h$. We can write $N(0) = 1$
and $N(-1) = 0$. For $h > 1$, we have $N(h) \geq N(h-1) + N(h-3)$; this is because one
of the subtrees of a tree with height $h$ has height $h-1$ and the other one can have
height at least $h - 3$ by the definition of foo trees. Following the inequality, we get
$N(h) \geq 2N(h-3)$, and iterating $h/3$ times, we get $N(g) \geq 2^{h/3}N(0) = 2^{h/3}$. In other
words, $\log(N(h)) = h/3$ or $h \leq 3\log(N(h))$. This means that a foo tree of $n$ nodes has
height $O(\log n)$.

Marking Scheme: 5 marks for deducing the right recursion (including a brief justi-
fication) and 3 marks for solving the recursion.

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=nil and if $T$ is a leaf we have $T$.left and

Figure 4: The answer to problem 1-c.
\( T.right \) equal to \( \text{nil} \). The input is the root of the AVL tree.

Here is the algorithm:

```plaintext
height(T)
T: the root of an AVL tree
1. if T.left = nil then
2. return T.balance
3. else
4. return 1 + height(T.left) + \lceil T.balance/2 \rceil
```

If \( T.left = \text{nil} \) then \( T.balance \) is either 0 or 1, the height of the tree. This shows that the correct answer is always returned at line 2. Now use induction on \( h \), the height of the tree. For the base case \( h = 0 \), we have \( T.left = \text{nil} \), and we have already noted that the correct answer will be returned at line 2. Assume, to apply the principle of strong mathematical induction, that the algorithm is correct for all trees of height up to \( h - 1 \), some \( h > 0 \), and let \( T \) be a tree of height \( h \). If \( T.balance \leq 0 \) then \( h \) is equal to 1 plus the height of \( T.left \). If \( T.balance = 1 \) then \( h \) is equal to 2 plus the height of \( T.left \). In both of these cases the correct answer is returned at line 4 since \( \lceil -1/2 \rceil = 0 \), \( \lceil 0/2 \rceil = 0 \) and \( \lceil 1/2 \rceil = 1 \).

The nonrecursive work done during each call to \( height \) is \( O(1) \), and the number of recursive calls is bounded by \( h \). Since \( h = O(\log n) \) the overall running time is \( O(\log n) \).

**Marking Scheme:** 6 marks for the right algorithm (note that the \( \lceil T.balance/2 \rceil \) can be replaced by a couple of if statements). 2 algorithms for a brief justification; that includes a brief explanation on why the algorithm is correct and why it runs in \( O(\log n) \). [no formal proof like above is necessary]
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):


See the following picture.

**Marking Scheme:** deduce two marks for each mistake with respect to height or location of an inserted node.

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.

The height of a tower is 1 if at least one of the flips is tail after the first flip of coins; that has a probability of 3/4. The height of a tower is 2 if the first two flips are both head (chance of 1/4) and in the next trial at least one of the flips is tail (chance of 3/4); so the height is 2 with a chance of 1/4 \times 3/4 = 3/16. Similarly, we have a height \( h \) with a chance of \((1/4)^{h-1}\) (not seeing a tail in previous \( h - 1 \) trials) times 3/4 (seeing a tail in \( h \)'th trial).

The expected height will be at most

\[ H = \frac{3}{4} \cdot 1 + \frac{1}{4^2} \cdot 2 + \frac{1}{4^3} \cdot 3 + \ldots + \frac{1}{4^{h-1}} \cdot \frac{3}{4} + \ldots = \frac{3}{2} \cdot (1+2/4+3/16+\ldots) \]

Note that if \( X = 1+2/4+3/16+\ldots \) then \( X/4 = 1/4+2/16+3/64+\ldots \) and hence \( X - X/4 = 1 + 1/4 + 1/16 + \ldots = 4/3, \) i.e., \( X = 16/9. \) We conclude that the expected height \( H \) will be \( \frac{3}{4} \cdot 16/9 = 4/3. \)

**Marking Scheme:** 2 marks for distinction between probabilities 3/4 and 1/4; 2 marks for writing the right formula for \( H \) and 2 marks for solving it correctly (getting the right final answer). Be gentle in giving partial marks for this question based on your judgement.