Homework 3
Show all work.

1. Apply the following methods to solve the recurrence, $T(n) = T(n/3) + 2$.
   **Comment**: For our purposes we can ignore the constants related to the value of the base case, although we mention them below.

   (a) Transformation with $n = 3^u$
   
   $T(n) = T\left(\frac{n}{3}\right) + 2$
   
   We substitute $n = 3^u$
   
   This gives $T(3^u) = T\left(\frac{3^u}{3}\right) + 2 = T(3^{u-1}) + 2$
   
   Also, we have $T(3^u) = T\left(\frac{3^u}{3}\right) + 2$
   
   $= T(3^{u-2}) + 2 + 2$
   
   $= T(3^{u-3}) + 2 + 2 + 2$
   
   which generalizes to $T(3^{u-k}) + 2k$
   
   The recurrence bottoms out when $n = 1$ or $3^u = 1$ or $u = 0$
   
   Since the recurrence decreases $u$ by 1 each time, it will reach $u = 0$
   
   after $u$ steps. Since $n = 3^u$, this is $u = \log_3 n$ steps,
   
   therefore $T(n) = T(1) + 2 \log_3 n$
   
   or $2 \log_3 n + c$ for some constant $c$ representing $T(1)$

(b) Iteration

$T(n) = T\left(\frac{n}{3}\right) + 2$

$= T\left(\frac{3}{3}\right) + 2 + 2$

$= T\left(\frac{3}{3}\right) + 2 + 2$

$= T\left(\frac{3}{3}\right) + 2 + 2 + 2$

$= T\left(\frac{3}{3}\right) + 2 + 2 + 2$

This generalizes to: $T(n) = T\left(\frac{n}{3^k}\right) + 2k$

The recurrence bottoms out when $T\left(\frac{n}{3^k}\right) = T(1)$ so we solve for $k$ in

$\frac{n}{3^k} = 1$, which gives $n = 3^k$ or $k = \log_3 n$

so $T(n) = T(1) + 2 \log_3 n$

or $2 \log_3 n + c$ for some constant $c$. 
(c) Tree diagram

<table>
<thead>
<tr>
<th></th>
<th>cost of level</th>
<th>height</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T(n)$</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$T(n/3)$</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$T(n/9)$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$T(n/3^k)$</td>
<td>2</td>
<td>$k$</td>
</tr>
<tr>
<td>$T(1)$</td>
<td>$c$</td>
<td>$\log_3(n) + 1$</td>
</tr>
</tbody>
</table>

The height of the recursion tree is $\log_3 n$ since $\frac{n}{3^k} = 1 \Leftrightarrow k = \log_3 n$. At every level, the contribution to the total is 2, so the total is $2 \cdot \log_3 n + c$, where $c$ is the value of the base case.

(d) Check your answer by substituting it back into the recurrence

We substitute $2 \log_3 n + c$ into the recurrence.

$$2 \log_3 n + c = 2 \log_3 \left(\frac{n}{3}\right) + c + 2$$
$$= 2(\log_3 n - \log_3 3) + c + 2$$
$$= 2 \log_3 n - 2 + c + 2$$
$$= 2 \log_3 n + c,$$

so the recurrence is satisfied.

2. Let $f(n)$ be a positive function $f(n) > 0$, and let $k$ be a constant. Prove or give a counter-example for each of the following statements.

(a) $f(n) + k$ is $\theta(f(n))$.

We take the limit of their ratio:

$$\lim_{n \to \infty} \frac{f(n) + k}{f(n)} = \lim_{n \to \infty} 1 + \frac{k}{f(n)}$$

Assuming that $\lim_{n \to \infty} f(n) > 0$, this approaches a constant as $n \to \infty$. This kind of assumption is reasonable in computer science, as we typically deal with complexity functions which are nondecreasing (since the cost of an algorithm must be nondecreasing with the size of the input). Under this assumption, $f(n) + k$ is $\theta(f(n))$. If we
allow \( f(n) \) to be a function such that \( f(n) \to 0 \) as \( n \to \infty \), then the limit of ratios can go to infinity and in such a case \( f(n) + k \) will not be \( \Theta(f(n)) \) but will instead be \( \omega(f(n)) \). \( f(n) = \frac{1}{n} \) is a good counterexample if we allow decreasing functions. So, we accepted both answers.

(b) \( f(n + k) = \Theta(f(n)) \). Here is a counterexample: \( f(n) = n! \):

\[
\lim_{n \to \infty} \frac{(n + k)!}{n!} = \lim_{n \to \infty} (n + k)(n + k - 1) \ldots (n + 1) = \infty,
\]
so for \( f(n) = n! \), \( f(n) \) is not \( \Theta(f(n + k)) \).

(c) \( f(n) \) is \( O(2^{f(n)}) \). This is true:

\[
\lim_{n \to \infty} \frac{2^{f(n)}}{f(n)} = \lim_{n \to \infty} \frac{f'(n) \cdot \ln 2 \cdot 2^{f(n)}}{f'(n)} = \lim_{n \to \infty} \ln 2 \cdot 2^{f(n)} = \left\{ \begin{array}{ll} \infty & \text{if } \lim_{n \to \infty} f(n) = \infty \\ c & \text{otherwise (for some constant } c \) \end{array} \right.
\]

Thus \( f(n) \) is in \( \{o(2^{f(n)}) \cup \Theta(2^{f(n)})\} = O(2^{f(n)}) \). The above proof works if \( f(n) \) is differentiable. If \( f(n) \) is not differentiable, we simply note that \( 2^u > u \) for all nonnegative \( u \) (since \( 2^0 > 0 \) and \( 2^u \) grows faster than \( u \)), and \( f(n) \) is a nonnegative function, so \( f(n) < 2^{f(n)} \) and is therefore \( O(2^{f(n)}) \) from the basic definition.

(d) \( f(n) \) is \( o(f(2^n)) \). A simple counterexample to this is \( f(n) = k \), where \( k \) is a constant. In that case,

\[
\lim_{n \to \infty} \frac{f(n)}{f(2^n)} = \lim_{n \to \infty} \frac{k}{k} = 1.
\]

In that case, \( f(n) \) is \( \Theta(f(2^n)) \), not \( o(f(2^n)) \).

3. A complete binary tree is one in which every node except leaf nodes has two descendents. Prove that more than half the nodes in a complete binary tree are leaf nodes.

We prove this by induction on \( h \), the height of the tree.

**Base Case:** When \( h = 1 \), we have the single root node which is also a leaf. The number of leaf nodes is greater than half the number of nodes in the tree, since \( 1 > \frac{1}{2} \).

**Induction:** Suppose that more than half the leaf nodes in a complete binary tree are leaf nodes when the \( h = k \). Let \( t \) = the total number of nodes in a complete binary tree of height \( k \), and let \( L \) be the
number of leaf nodes. Then our inductive hypothesis is $L > t/2$ or $2 \cdot L > t$. Now, consider the complete binary tree of height $k + 1$. It has $2 \cdot L$ leaf nodes (because its leaf nodes consist of the two children of each of the $L$ leaf nodes in the tree of height $k$), and $t + 2 \cdot L$ total nodes (the $t$ nodes that were in the tree of height $k$ plus the new $2L$ leaf nodes). By the inductive hypothesis,

$$2 \cdot L > t$$
$$2 \cdot L + 2 \cdot L > t + 2 \cdot L$$
$$2 \cdot L > \frac{t + 2 \cdot L}{2},$$

so the number of new leaf nodes is more than half the new total number of nodes. This completes the induction.

Q.E.D

4. **Extra Credit:**

A complete $k$-ary tree is one in which every node except leaf nodes has $k$ descendants. State the strongest result you can about the ratio of the number of leaf nodes to the number of all nodes, and prove your result.

The strongest possible result is the exact formula for the ratio of the number of leaf nodes to the number of all nodes. The number of all nodes in a tree of height $h$ is $\sum_{i=0}^{h} k^i = \frac{k^{h+1}-1}{k-1}$, and the number of leaf nodes is $k^h$, so the ratio is $\frac{k^h(k-1)}{k^h+1+k^{h-1}+\ldots+k+1}$. Factoring the denominator gives

$$\frac{k^h(k-1)}{1+\frac{1}{k^1}+\frac{1}{k^2}+\ldots+\frac{1}{k^{h-1}}+\frac{1}{k^h}}$$

**Comment:** Note that as $k$ gets large, the ratio approaches $1$, which implies that almost all of the nodes are leaf nodes when there is a high branching factor.

**Comment:** Note that such a tree is a form of exponential growth. The exponential function has the property that the current value of the function is proportional to the sum of all values over the history of the function, which is also reflected in the formula $\int e^x \, dx = e^x$. 