Some logarithmic properties for b>1, a>0, c>0, where a and c are both real numbers. (There are more in the textbook, as you have seen).

$$\log_b(a * c) = \log_b a + \log_b c$$

$$\log_b\left(\frac{a}{c}\right) = \log_b a - \log_b c$$

$$\log_b a^c = c \log_b a$$

$$\log_b 1 = 0$$

$$\log_b b = 1$$

$$b^{\log_b k} = k$$

$$\log_a b = \frac{1}{\log_b a}$$

Please see Appendix C for more information on Probabilities, Expected Value, Expected Value of Random Variables, Counting, Combinations, etc.

Please see Appendix A for more information on Summations / Geometric series, etc.

Show your work for calculations you perform.

Special for calculating summation solutions: Once you have determined the appropriate summation, you are free to either calculate its solution or to reference a solution. You may reference a book, an online source such as Wikipedia, or to use Wolfram Alpha or some other calculator for this part only. You must cite your source. This is only for **solving** summations, **not** for determining what the summations should be! I would like you to be able to solve summations, but that is not what I am trying to evaluate here.

Example: If you found a complexity to be:

$$T(n) = \sum_{k=0}^{n} 3^k + \Theta(n^2)$$

Then you could say:

Citing Introduction to Algorithms (Cormen et. al), page 1147, A.5, we have that:

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1} - 1}{x - 1}$$

Therefore:

$$T(n) = \sum_{k=0}^{n} 3^k + \Theta\left(n^2\right) = \frac{3^{n+1} - 1}{3 - 1} + \Theta\left(n^2\right)$$
$$= \Theta\left(3^n\right) + \Theta\left(n^2\right) = \boldsymbol{\Theta(3^n)}$$

Homework problems are on the following pages.

Problem 1) (40 points) **(Solving Recurrences)**

Solve the following recurrences **for positive *n* values** using the stated method.

1A) $T(n) = 4\mathrm{T}\left(\frac{\mathrm{n}}{2}\right) + n$ using Master Method

$$f(n) \neq \Theta\left(n^{(\log_2 4))}\right) = \Theta\left(n^2\right)$$

$$f(n) = \mathrm{O}\left(n^{(\log_2 (4-2))}\right) = \mathrm{O}(n^1)$$

Therefore,

$$T(n) = \Theta\left(n^{(\log_2 4))}\right) = \Theta(n^2)$$

1B) $T(n) = 11\mathrm{T}\left(\frac{\mathrm{n}}{3}\right) + n^2$ using Master Method

$$f(n) \neq \Theta\left(n^{(\log_3 11)}\right)$$

$$f(n) = \mathrm{O}\left(n^{(\log_3 (11-2))}\right) = \mathrm{O}(n^2)$$

Therefore,

$$T(n) = \Theta\left(n^{(\log_3 11)}\right) = \Theta(n^{\sim 2.182658339})$$

1C) $T(n) = 8T\left(\frac{n}{2}\right) + 2n^3$ using Master Method

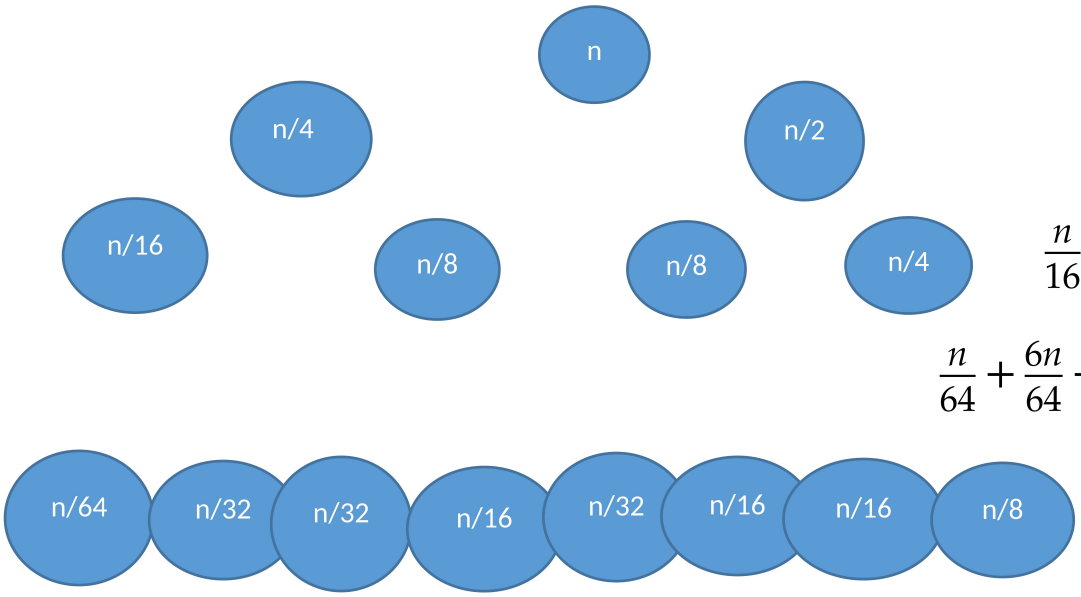
$$f(n) = 2n^3 = \Theta(n^3)$$

$$f(n) = \Theta\left(n^{\log_2 8}\right) = \Theta\left(n^3\right)$$

Therefore,

$$T(n) = \Theta\left(n^3 \lg(n)\right)$$

1D) $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n$ using Recursion Tree Method



$$n$$

$$\frac{n}{4} + \frac{2n}{4} = \frac{3^1 n}{4^1}$$

$$\frac{n}{16} + \frac{4n}{16} + \frac{4n}{16} = \frac{3^2 n}{4^2}$$

$$\frac{n}{64} + \frac{6n}{64} + \frac{12n}{64} + \frac{8n}{64} = \frac{3^3 n}{4^3}$$

$$\sum_{i=0}^{\log(n)} \frac{3^i n}{4^i} = n \sum_{i=0}^{\log(n)} \left(\frac{3}{4}\right)^i$$

The rate of change in the denominator of the summation is higher than the rate of change in the numerator, which will make the summation a constant term. Therefore $T(n) = \Theta(n)$

Problem 2) (30 points)

Given a recurrence of the form:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

Show that the height of the recursion tree is equal to $\log_b(n)$.

The base cases for the recursive function is when T(1) = 1. Therefore, when (n/b) = 1 the recursive function has reached it base case and does no further recursion calls.

Now looking at the function the variable a, it only determines the amount of children the per recursion call, it does not denote the height of the tree.

So, the height of the tree is determined by T(n/b). Knowing that the recursion call has reached its base case when T(n/b) = 1 we now must figure out how many times do you have to drop down a layer in the tree and multiply b by itself in order to for it to equal n, which will mean that it equal 1 which is the base case for the recursive function. This can be shown as $b^x = n$ were x is the number of layers you have to add and multiple b by itself. And to solve $b^x = n$ you can change the form and use logarithms to get $\log_b n = x$.

Example

Top layer $T(n) = T\left(\frac{n}{b}\right)$, Second Layer $T\left(\frac{n}{b}\right) = T\left(\frac{\frac{n}{b}}{b}\right) = T(\frac{n}{b^2})$...

and so forth until $T\left(\frac{n}{b^i}\right) = 1$ *were i is the height*

Problem 3 (30 points)

Suppose you have a polynomial. This can be represented as a summation

z$A(x) = \sum_{k=0}^{n-1} a_k x^k$

where $a = [a_0, a_1, .. a_{n-1}]$ is a vector of n coefficients.

We want an efficient function to compute $A(x)$ for any given constant x and a coefficient vector a of size n.

Naïve Approach: Represent as a coefficient function:

$$A(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

Horner's Rule Approach: Re-write the function as:

$$A(x) = a_0 + x(a_1 + x(a_2 + \ldots x(a_{n-1}))\ldots)$$

<u>Problem:</u> Write pseudocode for a recursive algorithm to compute $A(x)$ for n coefficients in the array a and a given constant x using the Horner's Rule Approach. Evaluate the time complexity as an asymptotic bound.

//Preconditons: n coefficients, a is the array of coefficients, x is the constant, and current is the beginning index (0 to n-1)

//Example: horner(5,a,2,0)

```
Float horner(n, a, x, current) {
        If (n-1 == current) {          Θ(1)
                Return a[current];     Θ(1)
        }
        Return a[current] + x * horner(n,a,x,current+1);
}
```

$$T(n) = T(n-1) + \Theta(1)$$

$$n = \Theta(1)$$
$$n-1 = \Theta(1)$$
$$n-2 = \Theta(1)$$
$$\dots$$
$$1 = \Theta(1)$$

$$T(n) = \sum_{i=0}^{n} \Theta(1)$$

$$T(n) = n * \Theta(1) = \Theta(n)$$