Analysis of Algorithms
CS 375, Fall 2022
Small Assignment 1
Due **BY THE BEGINNING OF CLASS** Monday, September 19

- For this Smaller Assignment, the standard file naming conventions apply: Please submit your typewritten answers in a PDF file named

  `CS375_SA1_<userid>.pdf`

  where `<userid>` is replaced by your Colby userid (your full userid, including class year)—for example, my file would be called `CS375_SA1_eaaron.pdf`—and submit it to your `SubmittedWork` folder in your Google drive space for this course. Please reach out to me right away with any questions or concerns about this!

- *A general note for CS375:* As always, please present answers cleanly and explain them thoroughly, giving all details needed to make your answers easy to understand. Graders may not award full credit to incomplete or illegible solutions, or answers without explanations. Clear communication *is* the point, on every assignment.

  Please feel free to ask me any questions about explanations that might come up!

Exercises

1. Familiarity with summations and $\sum$ notation is essential for algorithm complexity analysis! Here are some warm-up exercises. For these exercises, as usual for CS375, please be sure to show your work and give a short but informative explanation for every answer.

   (a) Exercise A.1.1 (pg. 1149). (A "simple formula" here means a mathematical expression with no $\sum$ notation, for which plugging in a value for n will give the numeric value of the formula. For example, a simple formula for $\sum_{i=1}^{n} i$ is $\frac{n(n+1)}{2}$.)

   (b) Find a simple formula for $\sum_{i=0}^{n} 2(3^i - 1)$.

   Hints:

   - Please re-read CLRS, Appendix A.1.
   - Often, a good way to solve these exercises is to use properties of summations (such as **Linearity**, on pg. 1146) to transform your summation into another form, and then apply some known summation formulas to get the Σ notation out. The summation formulas A.1 (page 1146) and A.5 (page 1147) are my favorites!
   - Sometimes, with summations, it can be easy to make off-by-one errors. This may or may not apply to your solutions to these exercises, depending on your approach, but in general, please be on the lookout for off-by-one errors!

2. Here are a few exercises on properties of logarithms: You'll be asked to show that a few properties of logarithms are true. The objective is to help you become more familiar with some properties of logarithms that are especially important for analysis of recursive algorithms.

As an example, here's how we might show that $\log_b xy = \log_b x + \log_b y$. In the below, recall from the definition of logarithm that a logarithm really stands for an exponent—$x = log_b n$ means that $b^x = n$, i.e., x is the exponent we raise b to, to get n.

> To make it easier to talk about raising some relevant expressions to a power, we'll introduce a few variables: let $k = \log_b xy$, $\ell = \log_b x$, and $m = \log_b y$. Then, by the definition of logarithm (above), $b^k = xy$, $b^\ell = x$, and $b^m = y$. (Note: Do you see why?) Therefore, $b^k = b^\ell b^m = b^{\ell+m}$, and thus, by standard properties of exponents, $k = \ell + m$, which is what we had set out to show.

Now, pick 2 of the following 3 properties of logarithms (your choice!) and show that they are true. (If you turn in all 3 of them, only your best 2 will be counted for your grade, so I hope you'll give all 3 of them a try!)

Please give detailed explanations, as shown in the example above! Assume that a, b, c, n are positive real numbers (but note—they are not necessarily integers).

(a) $\log_b a^n = n \log_b a$.

(b) $\log_b n = \frac{\log_c n}{\log_c b}$. (This shows that changing the base of a logarithm—say, from $log_b n$ to $log_c n$—really means multiplying by a constant factor, because this could be rewritten as $log_b n = \frac{1}{log_c b} \cdot log_c n$.)

(c) $a^{\log_b n} = n^{\log_b a}$. (This result is sometimes called the "log-switching theorem" since it says that we can "switch" a and n.)