CS41 Homework 3

This homework is due at 11:59pm on Monday, September 20. Write your solution using LaTeX. Submit this homework using as a **.tex** file on **github**. This is an individual homework. It's ok to discuss approaches at a high level. In fact, you are encouraged to discuss general strategies. However, you should not reveal specific details of a solution, nor should you show your written solution to anyone else. The only exception to this rule is work you've done with a lab partner *while in lab*. In this case, note (in your **post-homework survey**) who you've worked with and what parts were solved during lab.

The main **learning goals** of this lab are to work with asymptotic bounds, practice algorithm analysis, and remember CS35 content. There is a bit of algorithm design, too.

1. **Analysis.** Let $f(n) = 7n^{4/7}$ and $g(n) = n^{2/7}(\log n)^5$. Prove that $g(n) = O(f(n))$. You may use techniques and facts from class and the textbook; your proof should be formal and complete.

2. **Asymptotic rates of growth.** Arrange the following functions in ascending order of growth rate. That is, if g follows f in your list, then it should be the case that $f = O(g)$.

   - $f_1(n) = n^{3.6}$
   - $f_2(n) = \frac{1}{2}n\log(n) + 9$
   - $f_3(n) = 5 \cdot 10^n$
   - $f_4(n) = 7n + 3$
   - $f_5(n) = \sqrt{4n}$

   No proofs are necessary, just give an ordering.

3. **Lots of functions.** Let k be a fixed constant and suppose that $f_1, \ldots, f_k$ and h are functions such that $f_i(n)$ is $O(h(n))$ for all i. (We also write this $f_i(n) \in O(h(n))$.)

   (a) Let $g_1(n) = f_1(n) + ... + f_k(n)$. Is $g_1(n) \in O(h(n))$? Prove or give a counterexample. If you give a counterexample, give an alternate statement that best captures the relationship between the two functions.

   (b) Let $g_2(n) = f_1(n) \cdot \ldots \cdot f_k(n)$. Is $g_2(n) \in O(h(n))$? Prove or give a counterexample. If you give a counterexample, give an alternate statement that best captures the relationship between the two functions.

4. **Close to sorted.** Say that a list of numbers is "k-close-to-sorted" if each number in the list is less than k positions from its actual place in the sorted order. (So a 1-close-to-sorted list is *actually* sorted.) Give an $O(n \log k)$ algorithm for sorting a list of numbers that is k-close-to-sorted.

   In your algorithm, you may use any data structure or algorithm from CS35 by name, without describing how it works.

5. **(extra challenge)** For this problem, your example functions should have domain and range the positive integers $\mathbb{N}$.

- Find (with proof) a function f such that $f(2n)$ is $O(f(n))$.
- Find (with proof) a function g such that $g(2n)$ is not $O(g(n))$.
- Find (with proof) a function h such that $h(2^n)$ is $O(h(n))$.

6. **(extra challenge)** Give a proof or counterexample: If f is not $O(g)$, then g is $O(f)$.