CSE 60111: Complexity and Algorithms
Homework 0

A note about homework 0: A large goal of this particular assignment is to force you to review some of what was covered in discrete mathematics and in data structures, both of which are important prereqs for this course. You can't design and analyze algorithms without understanding fundamental data structures, proofs, and big-O notation! So expect to pull out your old textbooks if you're rusty on those, or look in the introduction of our textbook for some suggestions, or just email to ask me for some recommendations if you aren't sure where to look.

Academic integrity policy: In this class, while you're welcome to use any reference you'd like, you are responsible for both citing your sources AND re-writing the solution in your own words - including any usage of ChatGPT or similar tools. So feel free to use the internet, but I'd recommend reading, thinking about it, adding a citation for what you read to your homework, and then putting all that away and writing it from memory, so you'll understand the answer properly. Please believe me when I say that ANY verbatim copying will get you a 0 on the homework, as well as being reported to the department. Any second offense will get you a 0 in the class, as well as be forwarded to the college for further disciplinary measures.

1. (10 points total) Sort the following 25 functions from asymptotically smallest to asymptotically largest, indicating ties if there are any. You do not need to turn in proofs (in fact, please *don't* turn in proofs), but you should do them anyway just for practice.

1	n	n^2	$\lg n$	$1 + \lg\lg n$
$\cos n + 2$	$n^{\lg n}$	$(\lg n)!$	$(\lg n)^{\lg n}$	F_n
$\lg^{1000} n$	$2^{\lg n}$	$n \lg n$	$\sum_{i=1}^{n} i$	$\sum_{i=1}^{n} i^2$
$n!$	$\lg(n^{10000})$	$\lfloor \lg\lg(n) \rfloor$	$2^{2 \log n}$	$15n^2 - 12n + 8 \lg n + 4$

To simplify notation, write $f(n) \ll g(n)$ to mean $f(n) = o(g(n))$ and $f(n) \equiv g(n)$ to mean $f(n) = \Theta(g(n))$. For example, the functions n^2, n, $\binom{n}{2}$, n^3 could be sorted either as $n \ll n^2 \equiv \binom{n}{2} \ll n^3$ or as $n \ll \binom{n}{2} \equiv n^2 \ll n^3$. *[Hint: When considering two functions $f(\cdot)$ and $g(\cdot)$ it is sometime useful to consider the functions $\ln f(\cdot)$ and $\ln g(\cdot)$.]*

[Hint: Hopefully this will become obvious, but my goal here is to make you remember: big-O, logarithms, summations, and ignoring constants! Go back and check your discrete math book and that chapter in your data structures book that talked about big-O.]

2. (10 points total)

   (a) Show how to find the k^{th} smallest value in a heap (or priority queue) in $O(k \log k)$ time. (Hint: Using another heap might help.)

   (b) Suppose you are given k sorted lists, of total length n. Using a heap, show how to merge these into a single sorted list in $O(n \log k)$ time. (Hint: Starting with the case $k = 2$ can be instructive.)

3. (10 points) Whenever a group of pigeons gathers, they instinctively establish a pecking order. For any pair of pigeons, one pigeon always packs the other, driving it away from food or potential mates. The same pair of pigeons always chooses the same pecking order, even after years of separation, no matter what other pigeons are around. Surprisingly, the overall pecking order can contain cycles - eg, pigeon A pecks pigeon B, and pigeon B pecks pigeon C, and pigeon C pecks pigeon C.

   (a) Prove that any finite set of pigeons can be arranged in a row from left to right, so that every pigeon pecks the pigeon immediately to its left. (Hint: induction is your friend.)

   (b) Suppose you are given a directed graph representing the pecking relationships among a set of n pigeons. Describe and analyze (and prove correctness for) an algorithm to compute a pecking order among the pigeons, as guaranteed in part (a).

4. (15 points total) The n^{th} Fibonacci binary tree $\mathcal{F}_n$ is defined recursively as follows:

   - $\mathcal{F}_1$ is a single root node with no children.
   - For all $n \geq 2$, $\mathcal{F}_n$ is obtained from $\mathcal{F}_{n-1}$ by adding a right child to every leaf node in $\mathcal{F}_{n-1}$ and adding a left child to every node that has only one child in $\mathcal{F}_{n-1}$.

   (a) Prove that the number of leaves in $\mathcal{F}_n$ is precisely the n^{th} Fibonacci number: $F_0 = 0, F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 2$.

   (b) How many nodes does $\mathcal{F}_n$ have? For full credit, give an *exact* closed form answer in terms of the Fibonacci numbers and prove that your answer is correct.

   (c) Prove that for $n \geq 2$, the left subtree of $\mathcal{F}_n$ is a copy of $\mathcal{F}_{n-2}$. (Hint: This is easier than it sounds!)