Exercise Set 3: Goldilocks and the 3 O's

The first thing you should do in exercise3.tex is set up your name as the author of the submission by replacing the line, `\submitter{TODO: your name}`, with your name and UVA email id, e.g., `\submitter{Grace Hopper (gmh1a)}`.

Before submitting , also remember to:

- List your collaborators and resources, replacing the TODO in `\collaborators{TODO: replace ...}` with your collaborators and resources. (Remember to update this before submitting if you work with more people.)
- Replace the second line in exercise2.tex, `\usepackage{uvatoc}` with `\usepackage[response2]{uvatoc}`, `\usepackage[response3]{uvatoc}`, `\usepackage[response4]{uvatoc}`, `\usepackage[response5]{uvatoc}` for the appropriate problem submission.

Collaborators and Resources: TODO: replace this with your collaborators and resources (if you did not have any, replace this with *None*)

Exercise 3-2: Equal to Constant Function (TCS Exercise 5.3)

For every $k \in \mathbb{N}$ and $x' \in \{0,1\}^k$, show that there is an $O(k)$ line *NAND-CIRC* program that computes the function $EQUALS_{x'} : \{0,1\}^k \to \{0,1\}$ that on input $x \in \{0,1\}^k$ outputs 1 if and only if $x = x'$.

REMOVED: Exercise 3-3: Random Functions are Hard (TCS Exercise 5.8)

Suppose $n > 1000$ and that we choose a function $F : \{0,1\}^n \to \{0,1\}$ at random, choosing for every $x \in \{0,1\}^n$ the value $F(x)$ to be the result of tossing an independent unbiased coin. Prove that the probability that there is a $2^n/(1000n)$ line program that computes F is at most 2^{-100}. (If you are stuck, see this exercise in the book for a hint.)

Exercise 3-4: Asymptotic Operators

For each sub-problem, indicate if the statement is *true* or *false* and support your answer with a convincing argument.

(a) $17n \in O(723n + \log n)$

(b) $\min(n^n, 3012) \in O(1)$

(c) $n^2 \in \Theta(n^3)$

(d) $2.0001^n \in O(2^n)$

(e) $log_n 10 \in \Theta(log_{2n} 17)$

Exercise 3-5: Little-O

Another useful notation is "little-o" which is designed to capture the notion that a function g grows much faster than f:

Definition 1 (*o*) *A function $f(n) : \mathbb{N} \to \mathbb{R}$ is in $o(g(n))$ for any function $g(n) : \mathbb{N} \to \mathbb{R}$ if and only if for every positive constant c, there exists an $n_0 \in \mathbb{N}$ such that:*

$$\forall n > n_0 . f(n) \leq cg(n).$$

Provide a proof for each of the following sub-problems.

(a) Prove that for any function f, $f \notin o(f)$.

(b) Prove that $n \in o(n \log n)$.

Exercise 3-6: Soft-O

Logarithms grow so slowly, they are practically "constants" — $\log_2 1\text{Trillion} < 30$. So, for any size problem we could compute on a real machine, theoreticians (and students who don't like to worry about manipulating logarithms) shouldn't waste their time worrying about logarithmic factors. Indeed, even polynomials on logarithms (i.e., $a_k(\log n)^k$ for any constant k) grow so slowly to usually be irrelevant. For this reason, we often use the "Soft-O" notation, $\widetilde{O}$:

Definition 2 (*$\widetilde{O}$*) *A function $f(n) : \mathbb{N} \to \mathbb{R}$ is in $\widetilde{O}(g(n))$ for any function $g(n) : \mathbb{N} \to \mathbb{R}$ if and only if $f(n) \in O(g(n) \cdot \log^k g(n))$ for some $k \in \mathbb{N}$.*

(Note: for convenience, we write $\log^k x$ to mean $(\log x)^k$. Also, we have seen the (constant) base of a $\log$ doesn't matter within our asymptotic operators, but if it is disturbing to have a $\log$ with uncertain base, it is fine to assume it is base 2.)

For each sub-problem, indicate if the statement is *true* or *false* and support your answer with a convincing argument.

(Hint: by understanding the definition of $\widetilde{O}$ above, you should realize that one way to prove a function is in a $\widetilde{O}$ set is to choose a value for k used in the definition, but to disprove inclusion in $\widetilde{O}$ you need to show that there is no k that works.)

(a) $n^2 \log n^3 \in \widetilde{O}(n^2)$

(b) $2.0001^n \in \widetilde{O}(2^n)$

(c) maximum number of comparison operations needed to sort a list of n items $\in \widetilde{O}(n)$

