

## Problem Set 4

Due: **9:59pm, Monday, 20 February**

This problem set focuses on understanding the asymptotic notations, syntactic sugar (Chapter 4), and the circuit complexity and the size hierarchy theorem (Chapter 5 of TCS).

You should complete the assignment by writing your answers in the `ps4.tex` LaTeX template. There is no Jupyter part for this assignment.

**Collaboration Policy:** You may discuss the problems with anyone you want. You are permitted to use any resources you find for this assignment **other than solutions from previous cs3102/cs3120 courses**. You should write up your own solutions and understand everything in them, and submit only your own work. You should note in the *Collaborators and Resources* box below the people you collaborated with and any external resources you used (you do not need to list resources you used for help with LaTeX).

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

To do the LaTeX part of this assignment, follow the same directions as previous problem sets. The URL for this template is: <https://www.overleaf.com/read/hsxfrxnsvyfw>

Before submitting your `ps4.pdf` file, 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 `ps4.tex`, `\usepackage{uvatoc}` with `\usepackage[response]{uvatoc}` so the directions do not appear in your final PDF.

## Asymptotic Operators

For all of these questions (and throughout cs3120), you should use the formal definitions of  $O$ ,  $\Theta$ , and  $\Omega$  as we defined in class this week. These are similar to the book's definitions in Section 1.4.8, but unlike the book's definitions and usage where the notations are used with "=" symbols, ideally we want you to think of them precisely as ways to describe *sets* of functions. Using those definitions carefully will help you solve these problems well.

### Soft- $O$

Logarithms grow so slowly, they are practically “constants” —  $\log_2 1\,000\,000\,000 < 40$ . 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,  $\tilde{O}$ :

**Definition 1** ( $\tilde{O}$ ) A function  $f(n) : \mathbb{N} \rightarrow \mathbb{R}$  is in  $\tilde{O}(g(n))$  for any function  $g(n) : \mathbb{N} \rightarrow \mathbb{R}$  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.)

**Problem 1** For each sub-problem, indicate if the statement is *true* or *false* and support your answer with a convincing argument.

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

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

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

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

**Answer:**

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

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

**Problem 2** *Goldilocks and the Three Os*

- (a) Prove that for any function  $f$ ,  $f \notin o(f)$ .
- (b) Prove that  $n \in o(n \log n)$ .

**Answer:**

## Counting Circuits and Functions

### Problem 3 *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  $EQUALS_{x'} : \{0, 1\}^k \rightarrow \{0, 1\}$  that on input  $x \in \{0, 1\}^k$  outputs 1 if and only if  $x = x'$ .

**Answer:**

**Problem 4** *Counting lower bound for multibit functions* (TCS Exercise 5.4)

Prove that there exists a number  $\delta > 0$  such that for every  $n, m$  there exists a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  that requires at least  $\delta m \cdot 2^n / n$  NAND gates to compute. (If you are stuck, see this exercise in the book for a hint.)

**Answer:**

**Problem 5** *Random Functions are Hard* (TCS Exercise 5.8)

Suppose  $n > 1000$  and that we choose a function  $F : \{0, 1\}^n \rightarrow \{0, 1\}$  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.)

**Answer:**

**Problem 6** *Understanding the Size Hierarchy Proof*

The proof of the Size Hierarchy Theorem (Theorem 5.5 in the book, and in class this week) defined a sequence of functions,  $f_0, f_1, \dots$ :

$$f_i(x) = \begin{cases} f^*(x) & \text{lex}(x) < i \\ 0 & \text{otherwise} \end{cases}$$

where  $f^*$  is some hard function, which we don't need to define but know must exist for sufficiently large  $n$  because of the number of functions in  $SIZE(s)$ .

- (a) Prove that when  $f^*(x_i) = 0$ ,  $f_{i+1} = f_i$  where  $\text{lex}(x_i) = i$ . That is, the two functions denoted by  $f_{i+1}$  and  $f_i$  are actually the same function.
- (b) Explain why this is not a problem for the proof.

**Answer:**

**Problem 7** *Finer Hierarchy*

The Size Hierarchy Theorem says that for every sufficiently large  $n$  and  $10n < s < 0.1 \cdot 2^n/n$ ,  $SIZE_n(s) \subsetneq SIZE_n(s + 10n)$  (where  $SIZE_n(s)$  was the set of all  $n$ -input boolean functions that can be implemented with  $s$  or fewer *NAND* gates). The need for the  $10n$ , means this does not tell us if, for a given  $s$ ,  $SIZE_n(s) \subsetneq SIZE_n(s+1)$ . As computer scientists, we should be a bit bothered by this. (Make sure you understand what the proof method used to show the theorem resulted in this  $10n$  term.) Prove that  $SIZE_2(1) \subsetneq SIZE_2(2)$ .



This is the end of the LaTeX problems for PS4. Remember to follow the last step in the directions on the first page to prepare your PDF for submission.