TheoryPrep **(Due: 7/18/20)**

Assignment #7: Asymptotics and Recurrence

Name: Student name(s)

We've taken some problems/solutions from the course texts and CS 121/125 material. Some problems are marked as (Challenge) or (Additional Problems). Do the rest first, and if you have time, return to these problems. It's **much more important** to have a rigorous understanding of the core problems and how to prove them than to simply finish all the additional problems.

Problem 1: Understanding Asymptotics

Learning goal: The goal for this section is twofold. First, after doing these problems, you will have experience manipulating the mathematical structures of asymptotics. Second, you will build an intuitive understanding that allows you to deduce the asymptotic relationship between two functions without needing to write out all the mathematical formality.

(a) This first problem will help you make sure you understand the definition of Big-Oh. Prove that $n^2 = O(n^3)$.

(b) In the video, we said that although we define $f(n) = o(g(n))$ as $\forall \epsilon > 0 \exists N \in \mathbb{N}. \forall n > N \in \mathbb{N} f(n) < \epsilon g(n)$, we could have replaced the last predicate with $f(n) \leq \epsilon g(n)$. Why would this be a valid thing to do?

(c) Let $f, g : \mathbb{N} \to \mathbb{R}^+$. Define what it means for $f = \omega(g)$. You should aim for this relationship to capture that $f > g$ in the same way that $f = o(g)$ captures $f < g$. After you do this problem, delete the box on page 3 of the reading and comment on the similarities and differences in your definition.

(d) Fill in the table below to denote the relationship between the functions below. You do not need to prove anything since this question is more about building intuition for the relationships between functions. You should however be able to justify the table to yourself (the last line is an **Additional Problem**)

A	B	O	o	Ω	ω	Θ
n^2	$2n^2$					
n^{137}	$\pi 2^n$					
$(\log \log n)^2$	$\sqrt[3]{\log n}$					
$2^{\sqrt{\log n}}$	n^7					
n^n	$n!$					
$\log(n!)$	$\log(n^n)$					

(e) This question will help you build some understanding for properties of asymptotics and fluency with the definitions. Let $f, g : \mathbb{N} \to \mathbb{R}^+$.

1. As you may recall from the video, $f(n) = O(g(n))$ corresponds to $f \leq g$ and $g(n) = \Omega(f(n))$ corresponds to $g \geq f$, so we might expect these to be equivalent conditions. Thus, prove that $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$.
2. We associate $f(n) = O(g(n))$ with $f \leq g$ and $f(n) = o(g(n))$ with $f < g$. Thus, we might guess that $f(n) = o(g(n)) \Rightarrow f(n) = O(g(n))$. This is true–prove it (note that it's also true that $f(n) = \omega(g(n)) \Rightarrow f(n) = \Omega(g(n))$ but that proof is similar so we will not make you do both).

(f) Below is a false proof that $2^n = O(1)$. Explain why it is false.

Letting $f(n) = 2^n$, we prove by induction on n that $f(n) = O(1)$. The base case of $f(0) = 2^0 = 1 \leq 1$ holds. In the inductive step, we assume that $f(n) \leq c \cdot 1$. We know that

$$\begin{aligned} f(n+1) &= 2^{n+1} \\ &= 2f(n) \\ &\leq (2c) \cdot 1 \end{aligned} \tag{0.1}$$

Thus, $2^{n+1} = O(1)$. This completes the inductive step and therefore $f(n) = O(1)$.

(g) (Additional Problem) Since we say $f(n) = O(g(n))$ is like $f \leq g$ and $f(n) = \Omega(g(n))$ is like $f \geq g$, we might expect one or the other to hold. However, this is not true–prove that there is an $f, g : \mathbb{N} \to \mathbb{R}^+$ such that $f \neq O(g)$ and $f \neq \Omega(g)$. We can't have $n, m \in \mathbb{N}$ such that $n \not\leq m$ and $n \not\geq m$, so why is this possible for functions?

(h) (Additional Problem) Part of the motivation for our definition for asymptotics is that they are invariant with respect to constant factors–if we multiply by a factor of 2 it's still about the same runtime. In other words $2f(n) = O(f(n))$. However, this may not be true when we move the factor of 2 to the input, considering $f(n), f(2n)$. Prove that there exists some $f : \mathbb{N} \to \mathbb{R}^+$ such that $f(n)$ is $o(f(2n))$ (so $f(n)$ is much smaller than $f(2n)$ and therefore multiplying the input by 2 makes a big difference).

(i) (Additional Problem) We know that big-oh is impervious to constant factors, but we will show an even stronger fact about little-oh; that it is impervious to muliplying by functions. Let $f, g, h : \mathbb{N} \to \mathbb{R}^+$. Prove that if $f = o(g)$ then $f \cdot h = o(g \cdot h)$ (where $f \cdot h(x) = f(x)h(x)$, the product not composition).

Problem 2: Recurrence Relations

Learning goal: This section contains two proofs that will require you to use induction to deduce the asymptotic runtime of a recurrence relation. This technique will likely come up in 124 and will also help you build your induction skills (which will come up everywhere!).

(a) You may have heard of the mergesort algorithm before (if you have not and want to learn more about it you can see this video). Mergesort basically splits the problem of sorting an array of length n into the problem of sorting each half, and the problem of merging those two arrays. If we let $T(n)$ be the time for mergesort to sort a list of length n, we thus get the recurrence relation that $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n$ (mergesort the first half of the list, then the second half of the list which we give the extra element if it exists, and then a linear time algorithm to merge them–the new symbols here are the floor and ceiling function, meaning round down or up to the nearest integer). Prove that $T(n) = O(n \log(n))$. You may assume $T(1) = 1$. The log in this problem is considered to have base 2.

(b) (Additional Problem) Let $T(n) = 4T(n-2) + n$. Find an $f(n)$ such that $T(n) = \Theta(f(n))$ and prove that $T(n) = O(f(n))$. If you have time, try to prove that $T(n) = \Omega(f(n))$. You may assume $T(1) = 1$ and that n is odd (in other words, instead of proving that $\exists c, N \in \mathbb{N}. \forall n > N \in \mathbb{N} f(n) ? cg(n)$ you can prove that $\exists c, N \in \mathbb{N}. \forall n > N \in$ odd natural numbers $f(n) ? cg(n)$–where ? represents either $\leq$ or $\geq$)

(c) (Challenge Problem) Prove the master theorem, which says that if $T(n) = aT(n/b) + cn^k$ for $a \geq 1, b \geq 2$ integers and $c, k > 0$ then

$$T(n) = \begin{cases} \Theta(n^{\log_b a}) & a > b^k \\ \Theta(n^k \log n) & a = b^k \\ \Theta(n^k) & a < b^k \end{cases} \tag{0.2}$$

Problem 3: Review

Learning goal: The goal for this section is to reflect on how you can improve your proof skills!

(a) Select the problem from a previous week that you think you would learn the most from redoing. Copy the problem and your instructor's feedback below, rewrite the proof, and add a short reflection on the changes you made.

(b) (Additional Question) Choose another problem you received feedback on and copy the problem and feedback you received, rewrite the proof, and write a sentence about why you made your changes (so do Part A again) OR solve a problem you left blank from a previous week (try to pick from a week you felt least secure on).

Problem 5: Logistics

Purpose: This helps us make sure the course is going at the right speed!

(a) How long did you spend on the videos and readings this week?

(b) How long (including time in problem sessions) did you spend on this problem set?

(c) Do you have any feedback about the course in general (did the videos and readings sufficiently prepare you for the problem set)?