CSCI 3104 Fall 2022
Prof. Grochow and Nagesh

Problem Set 5

Due Date ..... Monday October 10, 2022 8pm MT
Name ..... **Tyler Huynh**
Student ID **109603994**
Collaborators ..... **List Your Collaborators Here**

Contents

Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Useful links and references on LaTeXcan be found here on Canvas.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section Honor Code). Failure to do so will result in your assignment not being graded.

Honor Code (Make Sure to Virtually Sign the Honor Pledge)

Problem HC. On my honor, my submission reflects the following:

- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.

- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

In the specified region below, clearly indicate that you have upheld the Honor Code. Then type your name.

Honor Pledge. I, **Tyler Huynh** on my honor pledge that my submission is a reflection of my own understanding of the material, any and all collaborations/sources have been properly cited, I have not posted any material to external sources, and I have not copied other solutions as my own. □

12 Standard 12- Asymptotics I (Calculus I techniques)

Problem 12. For each part, you will be given a list of functions. Your goal is to order the functions from **slowest growing** to **fastest growing**. That is, if your answer is $f_1(n), \ldots, f_k(n)$, then it should be the case that $f_i(n) \leq O(f_{i+1}(n))$ for all i. If two adjacent functions have the same order of growth (that is, $f_i(n) = \Theta(f_{i+1}(n))$), clearly specify this. **Show all work, including Calculus details.** Plugging into WolframAlpha is not sufficient.

You may find the following helpful.

- Recall that our asymptotic relations are transitive. So if $f(n) \leq O(g(n))$ and $g(n) \leq O(h(n))$, then $f(n) \leq O(h(n))$. The same applies for little-o, Big-Theta, etc. Note that the goal is to order the growth rates, so transitivity is very helpful. We encourage you to make use of transitivity rather than comparing all possible pairs of functions, as using transitivity will make your life easier.
- You may also use the Limit Comparison Test. However, you **MUST** show all limit computations at the same level of detail as in Calculus I-II. Should you choose to use Calculus tools, whether you use them correctly will count towards your score.
- You may **NOT** use heuristic arguments, such as comparing degrees of polynomials or identifying the "high order term" in the function.
- If it is the case that $g(n) = c \cdot f(n)$ for some constant c, you may conclude that $f(n) = \Theta(g(n))$ without using Calculus tools. You must clearly identify the constant c (with any supporting work necessary to identify the constant- such as exponent or logarithm rules) and include a sentence to justify your reasoning.

You may also find it helpful to order the functions using an `itemize` block, with the work following the end of the `itemize` block.

- This function grows the slowest: $f_1(n)$
- These functions grow at the same asymptotic rate and faster than $f_1(n)$: $f_2(n), f_3(n), \ldots$
- These functions grow at the same asymptotic rate, but faster than $f_2(n)$: $f_k(n)$.

Also below is an example of an `align` block to help you organize your work.

$$\begin{aligned}\lim_{n\to\infty} \frac{n^2}{2^n} &= \lim_{n\to\infty} \frac{2n}{\ln(2) \cdot 2^n} \\ &= \lim_{n\to\infty} \frac{2}{(\ln(2))^2 \cdot 2^n} \\ &= 0.\end{aligned}$$

12.1 Problem 12??

(a) $n^2 + 200n, \qquad n + 1000000, \qquad n^3 - 20n^2, \qquad \frac{n^2}{\sqrt{n}}.$

Answer. Let us first define our values for further use:

$$\begin{aligned} f_1(n) &= n^2 + 200n \\ f_2(n) &= n + 1000000 \\ f_3(n) &= n^3 - 20n^2 \\ f_4(n) &= \tfrac{n^2}{\sqrt{n}} \end{aligned}$$

We will first start by comparing $f_1(n)$ with $f_2(n)$ to find which function will grow at a larger rate using L'hopitals rule:

$$\begin{aligned} \lim_{n\to\infty} \frac{n^2 + 200n}{n + 1000000} &= \lim_{n\to\infty} \frac{n^2 + 200n}{n + 1000000} = \frac{\infty}{\infty} \qquad \textit{Indeterminate} \\ &= \lim_{n\to\infty} \frac{2n + 200}{1} \\ &= \infty. \end{aligned}$$

We can see from the above that this will be $f_2(n) \leq O(f_1(n))$, which is representative that $f_1(n)$ will be an upper bound of $f_2(n)$. Thus, $f_1(n) \geq \Omega(f_2(n))$, which is the lower bound.

We will now compare $f_1(n)$ with $f_4(n)$ to find which function will grow at a larger rate using L'hopitals rule:

$$\begin{aligned} \lim_{n\to\infty} \frac{n^2 + 200n}{\frac{n^2}{\sqrt{n}}} &= \lim_{n\to\infty} \frac{n^2 + 200n}{n^{3/2}} = \frac{\infty}{\infty} \qquad \textit{Indeterminate} \\ &= \lim_{n\to\infty} \frac{2n + 200}{\frac{3}{2}n^{\frac{1}{2}}} = \frac{\infty}{\infty} \qquad \textit{Indeterminate} \\ &= \lim_{n\to\infty} \frac{2}{\frac{3}{4}n^{-\frac{1}{2}}} \\ &= \lim_{n\to\infty} \frac{2n^{-\frac{1}{2}}}{\frac{3}{4}} = \frac{\infty}{3} \\ &= \infty. \end{aligned}$$

We can see from the above that this will actually be $f_4(n) \leq O(f_1(n))$, which is representative that $f_1(n)$ will be an upper bound of $f_4(n)$. Thus, $f_1(n) \geq \Omega(f_4(n))$, which is the lower bound.

We will now compare $f_1(n)$ with $f_3(n)$ to find which function will grow at a larger rate using L'hopitals rule:

$$\begin{aligned} \lim_{n\to\infty} \frac{n^2 + 200n}{n^3 - 20n^2} &= \lim_{n\to\infty} \frac{n^2 + 200n}{n^3 - 20n^2} = \frac{\infty}{\infty} \qquad \textit{Indeterminate} \\ &= \lim_{n\to\infty} \frac{2n + 200}{3n^2 - 40n} = \frac{\infty}{\infty} \qquad \textit{Indeterminate} \\ &= \lim_{n\to\infty} \frac{2}{6n - 40} = \frac{2}{\infty} \\ &= 0. \end{aligned}$$

We can see from the above that $f_3(n)$ will actually grow faster than $f_1(n)$, such that $f_1(n) \leq O(f_3(n))$. This shows that $f_3(n)$ will be an upper bound of $f_1(n)$. Thus, $f_3(n) \geq \Omega(f_1(n))$, which is the lower bound. We

can see that since $f_3(n)$ will grow faster than $f_1(n)$ and $f_1(n)$ grows faster than both $f_2(n)$ and $f_3(n)$, than we know that it will grow the fastest out of our set of functions.

We do not know whether $f_2(n)$ or $f_4(n)$ grows faster than each other so we will compare the two functions to each other using L'hopital's rule:

$$\begin{aligned}
\lim_{n\to\infty} \frac{n+1000000}{\frac{n^2}{\sqrt{n}}} &= \lim_{n\to\infty} \frac{n+1000000}{n^{3/2}} = \frac{\infty}{\infty} \qquad \textit{Indeterminate} \\
&= \lim_{n\to\infty} \frac{1}{\frac{3}{2}n^{\frac{1}{2}}} = \frac{1}{\infty} \\
&= 0.
\end{aligned}$$

We can see from the above that $f_4(n)$ will grow faster than $f_2(n)$, such that $f_2(n) \leq O(f_4(n))$. This shows that $f_4(n)$ will be an upper bound of $f_2(n)$. Thus, $f_4(n) \geq \Omega(f_2(n))$, which is the lower bound.

□

12.2 Problem 12??

(b) $(\log_7 n)^7$, $10 \log_5 n$, $\log_7(n^7)$, $n^{1/1000}$. *Hint:* Recall change of logarithmic base formula $\log_a x = \log_b x \cdot \log_a b$

Answer. □

13 Standard 13- Asymptotics II (Calculus II techniques):

Problem 13. *For each of the following questions, put the growth rates in order, from slowest-growing to fastest. That is, if your answer is $f_1(n), f_2(n), \ldots, f_k(n)$, then $f_i(n) \leq O(f_{i+1}(n))$ for all i. If two adjacent ones are asymptotically the same (that is, $f_i(n) = \Theta(f_{i+1}(n))$), you must specify this as well. Justify your answer (show your work). You may assume transitivity: if $f(n) \leq O(g(n))$ and $g(n) \leq O(h(n))$, then $f(n) \leq O(h(n))$, and similarly for little-oh, etc. The same instructions as for Problem 1 apply.*

13.1 Problem 13??

(a) $n^{\log_5 n}$, 4, $n^{\log_3 n}$, $n^{\log_n(n^2)}$, $n^{\log_n 5}$.

Answer. □

13.2 Problem 13??

(b) $n!$, 2^n , $2^{n/3}$, n^n, 2^{n-2}, $\sqrt{n^{2n+1}}$. (*Hint:* Recall Stirling's approximation, which says that $n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$, i.e. $\lim_{n\to\infty} \frac{n!}{\left(\frac{n}{e}\right)^n \sqrt{2\pi n}} = 1$.).

Answer. □

14 Standard 14- Analyzing Code I: (Independent nested loops)

Problem 14. Analyze the *worst-case* runtime of the following algorithms. Clearly derive the runtime complexity function $T(n)$ for this algorithm, and then find a tight asymptotic bound for $T(n)$ (that is, find a function $f(n)$ such that $T(n) = \Theta(f(n))$). Avoid heuristic arguments from 2270/2824 such as multiplying the complexities of nested loops.

Algorithm 1 Nested Algorithm 1

```
1: procedure Foo2(Integer n)
2:     for i ← 1; i ≤ n; i ← i + 2 do
3:         for j ← 1; j ≤ n; j ← 3 * j do
4:             print "Hi"
```

Answer. □

15 Standard 15- Analyzing Code II: (Dependent nested loops)

Problem 15. Analyze the *worst-case* runtime of the following algorithms. Clearly derive the runtime complexity function $T(n)$ for this algorithm, and then find a tight asymptotic bound for $T(n)$ (that is, find a function $f(n)$ such that $T(n) = \Theta(f(n))$). Avoid heuristic arguments from 2270/2824 such as multiplying the complexities of nested loops.

Algorithm 2 Nested Algorithm 2

```
1: procedure Boo1(Integer n)
2:     for i ← 1; i ≤ n; i ← i + 1 do
3:         for j ← i; j ≤ n; j ← j + 1 do
4:             print "Hi"
```

Answer. □