Discrete Structures (CSCI 246)
Homework 11

Purpose & Goals
The following problems provide practice relating to:

- asymptotic analysis (Big O),
- properties of Big O,
- algorithm analysis, and
- the problem solving process.

Submission Requirements

- **Type or clearly hand-write your solutions** into a pdf format so that they are legible and professional. Submit your pdf to Gradescope. **Illegible, non-pdf, or emailed solutions will not be graded.**
- Each problem should start on a new page of the document. When you submit to Gradescope, associate each page of your submission with the correct problem number. Please post in Discord if you are having any trouble using Gradescope.
- Try to model your formatting off of the proofs from lecture and/or the textbook.
- Submit to Gradescope early and often so that last-minute technical problems don't cause you any issues. Only the latest submission is kept. Per the syllabus, assignments submitted within 24 hours of the due date will receive a 25% penalty and assignments submitted within 48 hours will receive a 50% penalty. After that, no points are possible.

Academic Integrity

- You may work with your peers, but **you must construct your solutions in your own words on your own.**
- Do not search the web for solutions or hints, post the problem set, or otherwise violate the course collaboration policy or the MSU student code of conduct.
- Violations (regardless of intent) will be reported to the Dean of Students, per the MSU student code of conduct, and you will receive a 0 on the assignment.

Tips

- Answer each problem to the best of your ability. Partial credit is your friend!
- Read the hints for where to find relevant examples for each problem.
- Refer to the problem solving and homework tips guide.
- It is not a badge of honor to say that you spent 5 hours on a single problem or 15 hours on a single assignment. Use your time wisely and get help (see "How to Get Help" below).

How to Get Help
When you are stuck and need a little or big push, **please ask for help!**

- Timebox your effort for each problem so that you don't spend your life on the problem sets. (See the problem solving tips guide for how to do this effectively.)
- Post in Discord. If you're following the timebox guide, you can post the exact statement that you produced after spending 20 minutes being stuck.
- Come to office hours or visit the CS Student Success Center.

Problem 1 (9 points)

Hint: See the lecture on proofs about big O for examples of proving and disproving that $f(n) = O(g(n))$.

(a) (3 points) Prove that $3n^3 + 5n^2 - 2n = O(n^3)$ by constructing $c > 0, n_0 \geq 0 : \forall n \geq n_0 : 3n^3 + 5n^2 - 2n \leq c \cdot n^3$.

Grading Notes. This rubric is straightforward: give a correct c and n_0.

(b) (6 points) In class, we proved that $n^3 \neq O(n^2)$. Following that proof, copy the proof below, filling in the blanks, to show that $3n^3 + 5n^2 - 2n \neq O(n^2)$ by disproving $\exists c > 0, n_0 \geq 0 : \forall n \geq n_0 : 3n^3 + 5n^2 - 2n \leq c \cdot n^2$.

To prove $\exists c > 0, n_0 \geq 0 : \forall n \geq n_0 : 3n^3 + 5n^2 - 2n \leq c \cdot n^2$, we need to show that ______________ ____. So we show how to construct ____________________ given any ___________________________.

Let $c > 0$ and $n_0 \geq 0$. Consider $n = (c + 1)$. Then

$$\begin{aligned} 3n^3 + 5n^2 - 2n &= \textit{you fill in} \\ &= \textit{taking as many lines as you need...} \\ &= \end{aligned}$$

which is ___________________________ since $c > 0$. However, if $n_0 \geq c + 1$, then we can't set $n = (c + 1)$ since we need $n \geq n_0$. Thus, we instead choose $n = \max(n_0, c_1)$, and the inequality still holds.

We have shown how to produce an $n \geq n_0$ such that $3n^3 + 5n^2 - 2n > cn^2$ for any c, n_0, meaning that $3n^3 + 5n^2 - 2n$ is not $O(n^2)$.

Grading Notes. You get 1 point for each of the 4 blanks and 2 points for the algebra.

Problem 2 (7 points)

Hint: See the lecture on proofs about big O for examples of proving and disproving that $f(n) = O(g(n))$.

(a) (2 points) Give two functions, f and g, such that $f = O(n^2)$ and $g = O(2^n)$ but $f \neq O(g)$. Note: you may want to choose f and g to make (b) and (c) easier!

Grading Notes. This rubric is straightforward: give a correct f and g.

(b) (3 points) Prove that $f = O(n^2)$ and $g = O(2^n)$.

(c) (2 points) Prove that $f \neq O(g)$.

Grading Notes. For each of (b) and (c) above, recall that a proof that f is big O of g involves specifying a $c > 0, n_0 : \forall n \geq n_0 : f(n) \leq c \cdot g(n)$, and a proof that f is not big O of g involves demonstrating that no such c, n_0 can exist.

Problem 3 (10 points)

For the following two algorithms, derive the "best" (i.e., tightest) big O running times—technically, the big Theta running times.

Hint. See the lecture introducing algorithm analysis for examples of counting primitive operations and the lecture on properties of big O for finding the "tightest" big O bound for a function.

Grading Notes. For each of (a) and (b), we need to see:

- (3 points) A proposed function representing the number of primitive operations for the algorithm in terms of the input size, addressing each line and/or loop of the algorithm. You will not miss points for missing a few primitive operations. However, you do need to arrive at the correct number of times that a loop runs.
- (2 points) For your proposed function representing the number of primitive operations $f(n)$, find the "best" (i.e., tightest) $g(n)$ such that $f(n) = O(g(n))$—that is, you want a $g(n)$ such that $f(n) = \Theta(g(n))$. This $g(n)$ should also be in the simplest form possible, meaning it shouldn't have constant multipliers or lower-order terms.

(a)

Algorithm 1

```
for $i = 1$ to $n \cdot n$ do
    if $i$ is even then
        for $j = 1$ to $n$ do
            $x = x + 1$
```

(b)

Algorithm 2

```
for $i = 1$ to $n \cdot n$ do
    if $n | i$ then
        for $j = 1$ to $n$ do
            $x = x + 1$
```