# Homework 2

Algorithms, Spring 2023

**Honor code:** *Work on this assignment alone or with one partner. Between different teams, collaboration is at level 1 [verbal collaboration only]. There are lots of resources online, such as animations, visualizations, practice problems, videos, and solutions— which you are encouraged to explore to deepen your understanding. However, you must be careful not to search for the specific problems in the assignment with the intent of getting hints for the solutions. Searching for the assignment problems on the internet violates academic honesty for this class.*

---

## Preamble

The first problem asks you to prove a certain property related to big-oh. We'll start with an example.

Claim: For any functions $f, g : N \to N$, prove that $f(n))$ is $O(g(n))$ implies that $g(n)$ is $\Omega(f(n))$.

Note that this is a *universal statement*, about *any* functions $f(n)$ and $g(n)$. To prove that it is true, we must show that it works in *all* cases. This is hard because we cannot try *every single* function $f$ and *every single* function $g$.

Rule of thumb: To prove a universal statement, you must show that it works in all cases. Note that if we want to *disprove* a universal statement, we only need to find one counterexample.

We always start by writing down what we know and what we need to show.

- What we know: Let $f(n), g(n)$ be two functions such that $f(n) = O(g(n))$.
- What we want to prove: We want to show that $g(n) = \Omega(f(n))$.

Now let's plug in the definitions of $O()$ and $\Omega()$:

- $f(n) = O(g(n))$ by definition means that there exists a constant $c > 0$ such that $f(n) \leq c \cdot g(n)$, for any $n \geq n_0$.
- $g(n) = \Omega(f(n))$ by definition means that there exists a constant $c > 0$ such that $g(n) \geq c \cdot f(n)$, for any $n \geq n_0$. Note that the constant $c$ and the $n_0$ here will be different than the ones above, so to make that clear, we can denote them with different names, for e.g. $c'$ and $n_0'$: $g(n) = \Omega(f(n))$ means that there exists a constant $c' > 0$ such that $g(n) \geq c \cdot f(n)$, for any $n \geq n_0'$.

To prove the claim, we need to show that $f = O(g))$ impliesthat $g = \Omega(f)$, which, by plugging in the definition of $O()$ and $\Omega()$, means that we need to show that:

For any functions $f, g : N \to N$: If there exists a constant $c > 0$ such that $f(n) \leq c \cdot g(n)$, for any $n \geq n_0$, this means that there exists a constant $c' > 0$ such that $g(n) \geq c' \cdot f(n)$, for any $n \geq n_0'$.

Now that we have written down in terms of the definition what we know and what we need to show, let's show it. Here is the actual proof:

Proof of the claim: Let $f(n), g(n)$ be any two functions such that $f(n) = O(g(n))$. By definition we know that there exists a constant $c > 0$ such that $f(n) \leq c \cdot g(n)$, for any $n \geq n_0$. This means that $g(n) \geq \frac{1}{c} \cdot g(n)$, for any $n \geq n_0$. Since $c$ is a constant and $c > 0$, then $1/c$ is also a constant; let $c' = 1/c$. Then it follows that $g(n) \geq c' \cdot g(n)$, for any $n \geq n_0$. By definition, this means that $g(n) = \Omega(f(n))$. This is what we had to prove.

Since this logic works for *any* functions $f(n), g(n)$ such that $f(n) = O(g(n))$, we have shown that $f(n) = O(g(n))$ implies that $g(n) = \Omega(f(n))$.

### An alternate proof

In general there can be more than one way to prove things. Here's a different way to prove the Claim above, using the definition of $O()$ and $\Omega()$ in terms of limits.

Again, we start by writing the definition of what we have and what we want to prove:

- What we have: $f(n) = O(g(n))$, By definition means that the limit $\lim \frac{f(n)}{g(n)}$ is either 0 or a constant $c > 0$.
- What we want to show: $g(n) = \Omega(f(n))$ by definition means that the limit $\lim \frac{g(n)}{f(n)}$ is either $\infty$ or a constant $c > 0$.

Let's show that the first implies the second.

Alternate proof: Let $f(n), g(n)$ be any two functions such that $f(n) = O(g(n))$. By definition we know that the limit $\lim \frac{f(n)}{g(n)}$ is either 0 or a constant $c > 0$. From here it means that $\lim \frac{g(n)}{f(n)}$ is either $\infty$ or a constant $1/c$. By definition, this means that $g(n) = \Omega(f(n))$. This is what we had to prove.

Again, since this logic works for *any* functions $f(n), g(n)$ such that $f(n) = O(g(n))$, we have shown that $f(n) = O(g(n))$ implies that $g(n) = \Omega(f(n))$.

## The problems

1. Prove that big-O is transitive, that is: For any functions $f(n), g(n), h(n)$ such that $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$, then this implies that $f(n)$ is $O(h(n))$.

   *Hint: Use (one of ) the definition(s) of $O()$.*

2. Prove or disprove: $n^2 \log^{10} n \leq O(n^{2.1})$

   *Hint: You can manipulate this expression and reduce to one of the two basic rules that were discussed in class: (1) any polylogarithm is upper bounded by a any polynomial; and (2) any polynomial is upper bounded by any exponential. Or, you can show this using the definition of $O()$ in terms of limits.*

3. Prove or disprove: $2^{2n} \leq O(2^n)$

4. Prove or disprove: $4^n = \Theta(2^n)$

5. For each of the following functions, prove whether $f = O(g), f = \Omega(g)$ or both ($f = \Theta(g)$).

   (a) $f(n) = n \lg(n^3), g(n) = n \lg n$

   (b) $f(n) = 2^{2n}, g(n) = 3^n$

   (c) $f(n) = \sum_{i=1}^{n} \lg i, g(n) = n \lg n$

6. An algorithm solves problems by dividing a problem of size $n$ into 3 sub-problems of one fourth the size and recursively solves the smaller sub-problems. It takes constant time to combine the solutions of the sub-problems. Find the asymptotic running time of the algorithm.

## Evaluation

The assignment will be evaluated along several criteria:

1. **Correctness**: Is your solution correct?
2. **Justification**: Is your answer justified?
3. **Style**: Does it look professional and neat? Is the explanation written carefully in complete sentences, and well-organized logic? Is it easily human-readable? Is it easy to understand?

   - Assignments should be typed. Feel free to annotate the pdf to add figures and formulas which are too time-consuming to type. I recommend learning LaTex, but: some problems will require a lot of formatting (e.g. recurrences) which will be a time sink. In that case, simply annotate the equations on the pdf using the iPad.
   - Write each problem on a separate page or leave plenty of space between problems so that we can write comments.
   - Try to put yourself in the position of the reader. If you hadn't been thinking of this problem for 3 hours, would your answers make sense to you?
   - Try to finish the assignment early, then step away for a day or two, and then come back to it and read it again. Chances are you'll find something you can write more clearly.
   - Look at posted solutions for style advice (if solutions are not posted, ask).