MATH-UA 252/MA-UY 3204 - Fall 2022 - Worksheet #1

There are two major ways we will use "big-O notation" (Landau notation) in this class. In either case, the definition is:

$$f(x) = O(g(x)) \iff \lim_{x \to x_0} \frac{f(x)}{|g(x)|} < \infty. \tag{1}$$

One way we will use it is as a placeholder for remainders, errors, and the like. For example, if we Taylor expand $f(x+h)$ about x in the variable h, we could write:

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + O(h^3). \tag{2}$$

Here, from context, we are to interpret this as:

$$R(h) = f(x+h) - f(x) - f'(x)h - \frac{1}{2}f''(x)h^2 = O(h^3) \iff \lim_{h \to 0} \frac{R(h)}{h^3} < \infty. \tag{3}$$

That is, "$h \to 0$" in the limit is inferred from context. The other way we use big-O notation is to describe the complexity of algorithms. For example, if we have an $n \times n$ matrix A and a vector $x \in \mathbb{R}^n$, then forming the product $y = Ax$ can be done in $O(n^2)$ floating-point operations (FLOPs) using the definition of matrix multiplication:

$$y_i = \sum_{j=1}^{n} A_{ij} x_j, \qquad i = 1, \ldots, n. \tag{4}$$

We can see that we need to compute the product "$A_{ij}x_j$" for each i and j, resulting in n^2 floating-point multiplications. Once we do this, we also need to evaluating the sum $A_{i1}x_1 + \cdots + A_{in}x_n$ for each i, resulting in $n(n-1) = n^2 - n$ floating-point additions. Altogether, computing the matrix-vector product this way requires $2n^2 - n$ FLOPs. Since:

$$\lim_{n \to \infty} \frac{2n^2 - n}{n^2} = 2 < \infty, \tag{5}$$

we can conclude that $2n^2 - n = O(n^2)$—hence, the cost is $O(n^2)$. We say that this matrix-vector multiplication algorithm *has $O(n^2)$ (time) complexity*. Note that the fact that "$n \to \infty$" in this use of big-O is inferred from context, since we're interested in determining the rough, asymptotic cost of an algorithm for large problem sizes.

Problem 1. Algorithms frequently have time complexities which involve factors of n, $\log n$, and n^p for some p. Let's define a relationship "$\preceq$" such that $f(n) \preceq g(n)$ if $f(n) = O(g(n))$. Prove the following:

$$1 \preceq \log(n) \preceq n \preceq n \log n \preceq n^2. \tag{6}$$

Problem 2. Show that $\log(n)^p \preceq n$ for any fixed value of $p \in \mathbb{R}$.

Problem 3. Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$. Count the number of floating-point operations needed to compute the matrix-matrix product AB. Then, assume that $m = n = p$ and determine the big-O time complexity of this algorithm.