1 Section 1.2: What is Big-O Notation Really?

We did big-O notation last semester, but now that everyone in this class has taken calculus, we will do it again, and more precisely.

Definition 1.1. Let $f(n)$ and $g(n)$ be two functions. The function $f(n)$ is $O(g(n))$ if and only if

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty.$$

Let's compare this to an example that we did last semester: You have a book with n words in it. Describe the number of operations it takes for you to read the book twice (assuming that reading 1 word = 1 operation).

- The actual number of operations is $2n$. We will use $f(n)$ to describe this quantity.
- We describe this as $O(n)$. The function $g(n)$ is the function inside the parenthesis after the O.

Let's see how this fits with the definition of big-O: Let $f(n) = 2n$ and $g(n) = n$. Then the number of operations, $f(n)$, is $O(g(n))$ because

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2n}{n} = 2 < \infty.$$

Quick review of taking limits:

1. The limit of a constant is just equal to that constant.
2. If you are taking the limit of a rational function (a fraction) it is usually sufficient to look at the dominant (i.e. fastest growing) term in each of the numerator and denominator.
$$\lim_{n\to\infty} \frac{2n^3 + n^2 + 5}{5n^3 - 100n} = \lim_{n\to\infty} \frac{2n^3}{5n^3} = \frac{2}{5}.$$
3. The limit will evaluate to infinity if the the numerator has larger asymptotic growth, and will evaluate to 0 if the denominator has larger asymptotic growth.
4. When in doubt, and you are running into a limit of the form $\frac{0}{0}$ or $\frac{\infty}{\infty}$, you can always apply L'Hôpital's rule:
$$\lim_{n\to\infty} \frac{e^n}{n^2}$$

The previous example is technically $O(5n + 7)$ and $O(n^5)$ as well as being $O(n)$. **Show the different limits** But we use $O(n)$ because we want to describe it in the most concise and simplest form possible.

Also, the example canNOT be described as $O(1)$: show why

Definition 1.2. *The computer scientist's definition of big-O notation:* Let $f(n)$ describe the maximum number of operations which an algorithm uses on n data points. Then the algorithm is $O(g(n))$ if and only if

- $g(n)$ is a function with exactly one non-zero term,
- the coefficient of the one term in $g(n)$ is 1,
- $0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$.

For each of the following, identify $f(n)$, $g(n)$ and take the limit of the quotient to check that your answer is correct:

- Book with n words: you read it twice, then re-read the first 100 words.
- You simply read the first 50 words of a book.
- You read the first word. Then go back to the beginning and read the first and second word. Then start over and read the first three words. Repeat this until you have read the whole book. We need the fact that
$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}.$$
**This is important because people would probably be tempted to describe it as $O(n)$.

Alternative definition of big-O notation (with upper bounds rather than limits):

Definition 1.3. The function $f(n)$ is $O(g(n))$ if and only if there exists a number c such that $f(n) < c \cdot g(n)$ for some number c and all values of $n \geq n_0$ where n_0 is some number.
Note that c and n_0 will almost always be positive numbers

For example, let's assume we have an algorithm which takes $f(n) = n^3 + 20n + 1$ operations. We already know that $f(n)$ is $O(n^3)$ (so $g(n) = n^3$), but let's check that it satisfies the alternative definition!

I'm going to pick $c = 22$. Now notice that if $n = 0$ the inequality $f(n) < c \cdot g(n)$ does not hold. But asypmtotically, this inequality will hold since the n^3 term dominates. In this case we can pick $n_0 = 1$.

We see that this works by doing some algebra and isolating $c = 22$ in the inequality:
$$n^3 + 20n + 1 < 22n^3.$$
Dividing both sides by n^3 we find that
$$1 + \frac{20}{n^2} + \frac{1}{n^3} < 22.$$
This works for $n = 1$, and for any larger value of n, the LHS will be smaller, so it will still hold.

Show that if we choose $n_0 = 10$, then $c = 1.201$.

For this next example, still assume that we have an algorithm which takes $f(n) = n^3 + 20n + 1$ operations. Using both definitions of big-O notation, prove that this is NOT $O(n^2)$.

This is pretty straightforward when using the limit definition.

How do we do this for the inequality definition? It's something mathematicians call proof by contradiction (you'll do more of this in discrete math). For a contradiction proof, you pretend that something is true and then you do math on your assumption until you figure out that it just can't be true.

Here, if $f(n)$ is $O(n^2)$ then using the inequality definition of big-O, we know that for some constant value c the inequality
$$n^3 + 20n + 1 \leq c \cdot n^2$$
must be true for all values of $n \geq n_0$. Dividing both sides of the inequality by n^2 we find that
$$n + \frac{20}{n} + \frac{1}{n^2} \leq c.$$

However, no matter what value we choose for c, we can always find a value for n which will make this inequality untrue. For example, if we use $c = 1,000$ then we can also just pick $n = 1,000$ and the inequality will not hold. There is no combination of values c and n_0 which will work to make this inequality always true.