Important Remarks Regarding Big O Notation

IMPORTANT REMARKS REGARDING BIG O NOTATION

In how big O notation is defined, it looks at an upper bound on how fast a function grows. For example, a function that grows linearly in n is $\mathcal{O}(n^2)$ and also $\mathcal{O}(2^n)$, both of which grow much faster than linear. Typically, when using big O notation such as $f(n) = \mathcal{O}(g(n))$, we will pick g that grows at a rate that matches or closely matches the actual growth rate of f.

An example of a mathematical operation for which people often use a function g that doesn't actually match f in order of growth is multiplying two n-by-n matrices, for which the fastest known algorithm takes time $\mathcal{O}(n^{2.373})$ but since the exponent 2.373 is peculiar and the algorithm that achieves it is quite complicated and not what's typically used in practice, often times for convenience people just say that multiplying two n-by-n matrices takes time $\mathcal{O}(n^3)$. Note that when they say such a statement, they are not explicitly saying which matrix multiplication algorithm is used either.

Finally, some terminology:

- If $f(n) = \mathcal{O}(1)$, then we say that f is constant with respect to input n.

- If $f(n) = \mathcal{O}(\log n)$, then we say that f grows logarithmically respect to input n. For example, if f is the running time of a computer program, then we would say that f has logarithmic running time in n .
- If $f(n) = \mathcal{O}(n)$, then we say that f grows linearly in n. For example, if f is the running time of a computer program, then we would say that f has linear running time in n.
- If $f(n) = \mathcal{O}(n^2)$, then we say that f grows quadratically in n. For example, if f is the running time of a computer program, then we would say that f has quadratic running time in n.
- If $f(n) = \mathcal{O}(b^n)$ for some constant $b > 1$, then we say that f grows exponentially in n. For example, if f is the running time of a computer program, then we would say that f has exponential running time in n.

The above are of course just a few examples. There are many other “categories" of functions such as $\mathcal{O}(n^3)$ corresponding to cubic growth in n.

In much of our discussion to follow in 6.008.1x, we will analyze either the “time complexity" (i.e., how long it takes code to run in terms of number of basic operations, in big O) or “space complexity" (i.e., how much space a code uses for storing variables, also in big O).