CS3510 – Algorithms, Fall 2022

Lecture 1: Welcome & Big-O

January 11, 2022

Lecturer: Frederic Faulkner *Scribe: Aditya Diwakar*

Welcome to CS3510! Lecture began by reviewing the syllabus, please read this syllabus and become familiar with the class structure.

1 Big O

Why do we care about Big O? It's hard to talk about performance because runtime (by seconds) varies by computer by computer. Time based performance is affected by a variety of things: code efficiency, hardware, background tasks, etc. We only care about one item: *code efficiency*.

We want to ignore multiplicative (hardware performance) and additive (i.e. loading time) constants. Hence, we introduce Big-O. We say the following:

$f(n)$ is $O(g(n))$ if $\exists c, k : f(n) \leq c \cdot g(n)$ when $n > k$ (for large enough inputs).

Another way to say this is that if $f(n) = O(g(n))$, then we can say that the growth of $f(n)$ is bounded above by $g(n)$.

The "O" can be thought of equivelantly as $\leq$. When we say $f(n)$ is $O(g(n))$, this is somewhat related to $f(n) \leq g(n)$. We are comparing the rates of growth (rather than the functions directly). We also have:

Ω: $f(n)$ is $\Omega(g(n))$ if $g(n)$ is $O(f(n))$ (reverse)

Θ: $f(n)$ is $\Theta(g(n))$ if $f(n)$ is $O(g(n))$ AND $g(n)$ is $O(f(n))$ (both ways)

1.1 Examples

1. n is $O(n^2)$ (n^2 grows faster than n)
2. n^2 is $\Omega(n)$ (n grows slower than n^2)
3. n is $\Theta(3n + 27001)$ (ignoring constants, these grow at the same rate)

Proving (3) using the definition from above:

$$\frac{n}{3n + 27001} < 1 \qquad n \geq 0 \tag{1}$$

and the reverse direction...

$$\frac{3n + 27001}{n} = \frac{3n}{n} + \frac{27001}{n} = 3 + \underbrace{\frac{27001}{n}}_{\leq 27001 \text{ when } n \geq 1} \leq 3 + 27001 = 27004 \tag{2}$$

Hence, using equation (1), we have shown that n is $O(3n + 27001)$ using (2), $3n + 27001$ is $O(n)$, hence n is $\Theta(3n + 27001)$. $\square$

1.2 Important Notes of Big O

1. Ignore multiplicative constants
2. n^a dominates n^b if $a > b$: n^2 is $O(n^4)$ and n^4 is NOT $O(n^2)$ so we can focus on what contributes to growth the most, simplifying polynomials.
3. Exponentials dominate any polynomial, for example:

$$2^{0.5n} \text{ is } \Omega(n^{16,000} + n^{10,000,000})$$
$$a^n \text{ is dominated by } b^n \text{ for } a < b$$

4. Any polynomial dominates any log

For practice, let us compare 2^n vs 2^{n+1} by using a ratio method:

$$\frac{2^n}{2^{n+1}} = \frac{1}{2} \quad \text{and} \quad \frac{2^{n+1}}{2^n} = 2$$

hence 2^n is $O\left(2^{n+1}\right)$ and 2^{n+1} is $O\left(2^n\right)$ meaning 2^{n+1} is $\Theta(2^n)$.

1.3 More Examples

1. What is the relationship between n and $3^{\log_5 n}$?

   Using logarithm rules, notice that $3 = 5^{\log_5 3}$ so we can rewrite $3^{\log_5 n}$:

   $$3^{\log_5 n} = \left(5^{\log_5 3}\right)^{\log_5 n} = 5^{(\log_5 3 \cdot \log_5 n)} = \left(5^{(\log_5 n)}\right)^{\log_5 3} = \boxed{n^{\log_5 3}}$$

   and since $3 < 5$, then $\log_5 3 < 1$, meaning n dominates $3^{\log_5 n}$.

2. What is the relationship between $\sqrt{n}$ and $\log^2 n$?

   Recall that $\sqrt{n} = n^{0.5}$ meaning $n^{0.5}$ is polynomial and using rule (4), we know that this dominates any logarithm hence $\log^2 n$ is $O\left(\sqrt{n}\right)$

3. What is the relationship between 2^n and $2^{n/2}$?

   $$2^{n/2} = 2^{0.5n} = \left(2^{0.5}\right)^n = \left(\sqrt{2}\right)^n$$

   This means that $2^{n/2}$ has a smaller base, and therefore is $O\left(2^n\right)$.