Asymptotic Notation

CSCI 170 Fall 2019 Lecture 3

Sandra Batista

1.1-1.2

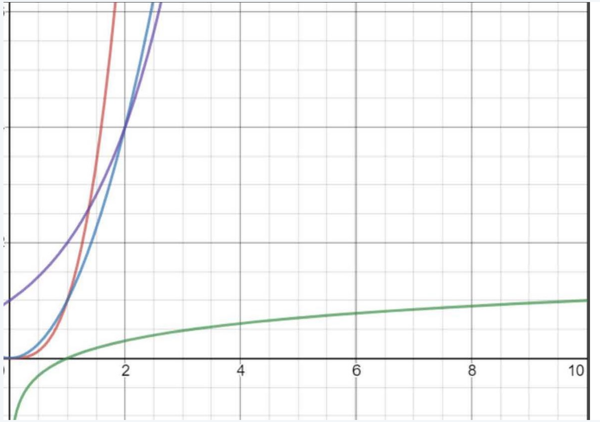
Asymptotic Notation

- **Big-O Notation**
- Big-Omega Notation
- Big-Theta Notation
- Hierarchies of Functions
- Properties of Asymptotic Notation

1.1–1.2

The Growth of Functions

Asymptotic Notation helps us to compare the growth rate of functions



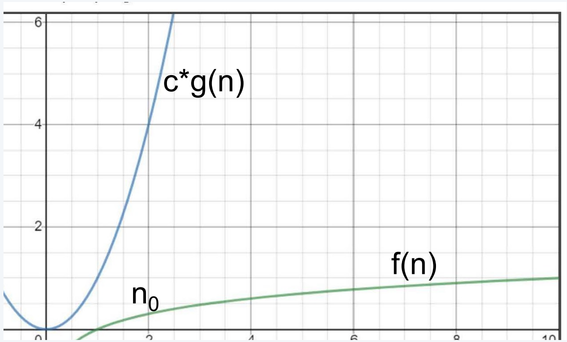
Sometimes we only need to know what happens to function of n as n grows very large

In computer science we apply asymptotic analysis to runtime analysis to determine its scalability: How many resources in terms of time and space are needed as the input size grows?

Big-O Notation

$O(g(n))$ = {f(n): there exists positive constants c and n_0 such that for any n greater than or equal to n_0, $0 \le f(n) \le c \cdot g(n)$ }

For functions, it is analogous to $\le$



Limit Rule: $f(n) \in O(g(n))$

if and only if

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

Big-O Notation Practice Problem 1

Is 2^{n+1} in $O(2^n)$?

Limit rule: $\lim_{n \to \infty} \frac{2^{n+1}}{2^n} = 2$ so $2^{n+1} \in O(2^n)$

Formal definition: Choose $n_0, c > 0$ st the definition holds. Let $n_0 = 10$ and $c = 5$

For all $n \geq 10$ $0 \leq 2^{n+1} \leq 5 * 2^n$.

Big-O Notation Practice Problem 2

Is 2^{2n} in $O(2^n)$?

limit Rule: $\lim_{n\to\infty} \frac{2^{2n}}{2^n} = \lim_{n\to\infty} 2^n = \infty$ $\quad 2^{2n} \notin O(2^n)$

Proof by contradiction:

Let's assume $2^{2n} \in O(2^n)$. There exists constants $n_0, c > 0$ st for all $n \geq n_0$ $0 \leq 2^{2n} \leq c2^n$.

$\Rightarrow 0 \leq 2^n \leq c$ for all $n \geq n_0$. (Intuition: not possible!)

$\Rightarrow$ let $\hat{n} = \max\{c, n_0\}$ $\quad 2^{\hat{n}} > c$ contradiction! $2^{2n} \notin O(2^n)$

Asymptotic Notation

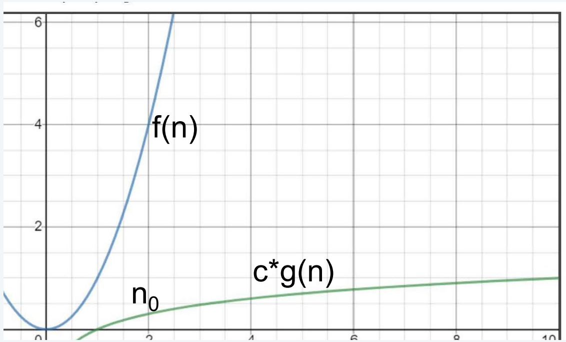
- Big-O Notation
- **Big-Omega Notation**
- Big-Theta Notation
- Hierarchies of Functions
- Properties of Asymptotic Notation

1.1–1.2

Big-Omega Notation

Ω (g(n)) = {f(n): there exists positive constants c and n_0 such that for any n greater than or equal to n_0, $0 \le c \cdot g(n) \le f(n)$ }

For functions, it is analogous to $\geq$



Limit Rule: f(n) ∈ Ω(g(n))

if and only if

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$$

Asymptotic Notation

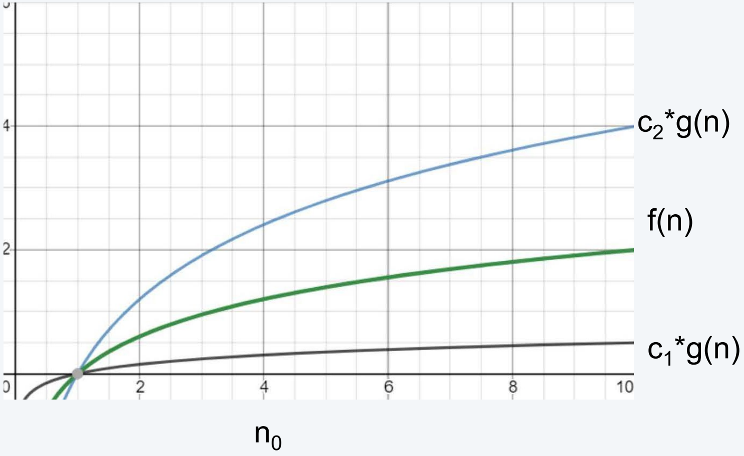
- Big-O Notation
- Big-Omega Notation
- **Big-Theta Notation**
- Hierarchies of Functions
- Properties of Asymptotic Notation

1.1–1.2

Big-Theta Notation

Θ(g(n)) = {f(n): there exists positive constants constants c_1 , c_2 and n_0 such that for any n greater than or equal to n_0, $0 \le c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$ }

For functions, it is analogous to $=$



Limit Rule: f(n) ∈ Θ(g(n))

if and only if for some positive constant k

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = k$$

Big-Theta Notation Practice Problem

Show $\log_{10}(n)$ is in $\Theta(\log_2(n))$.
(Hint: Recall how to change bases. $\log_b(n) = \log_a(n)/\log_a(b)$)

$\log_{10}(n) \in \Theta(\log_2(n)) \iff$

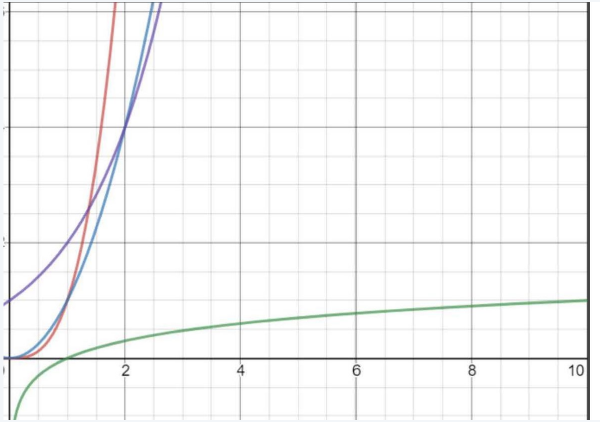
$$\lim_{n \to \infty} \frac{\log_{10}(n)}{\log_2(n)} = \frac{\log_2(n)}{\log_2 10 \cdot \log_2(n)} = \frac{1}{\log_2 10}$$

Asymptotic Notation

- Big-O Notation
- Big-Omega Notation
- Big-Theta Notation
- **Hierarchies of Functions**
- Properties of Asymptotic Notation

1.1–1.2

Hierarchies of Functions



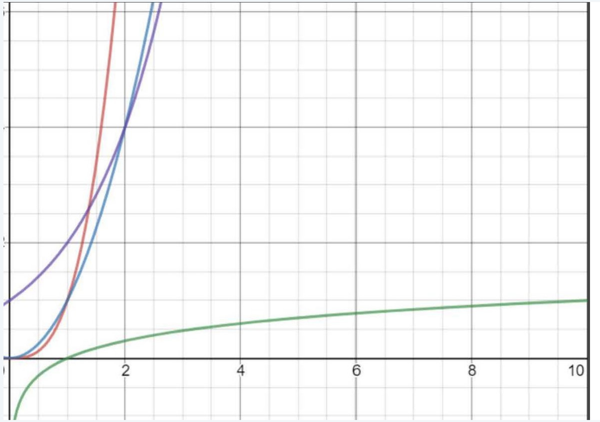
"Polynomial functions grow faster than logarithmic ones" or "Logarithmic time algorithms are better than polynomial ones"

More formally: $\log^c n$ is $O(n^d)$ for c, d > 0
[Theorem 21.5 in Lewis, Zax textbook]

"Exponential functions grow faster than polynomial ones" or "Polynomial time algorithms are better than exponential ones"

More formally: n^c is $O((1+d)^n)$ for c, d > 0
[Theorem 21.7 in Lewis, Zax textbook]

Hierarchies of Functions



Comparing the order of growth of functions is used to compare the running times of algorithms.

To do so, create bins of the functions, that are sorted by growth by hierarchies of functions, and then sort within the bins.

e.g. “Exponential functions grow faster than polynomial ones”

When in doubt, apply limit rules! Take the ratio of two functions and check whether numerator or denominator will be larger.

Hierarchies of Functions Practice Problem

Rank the following functions from smallest to largest in terms of order of growth:

$\log n^n$, n^2, $n^{\log n}$, $n \log \log n$, $2^{\log n}$, $\log^2 n$, $n^{\sqrt{2}}$, 2^n

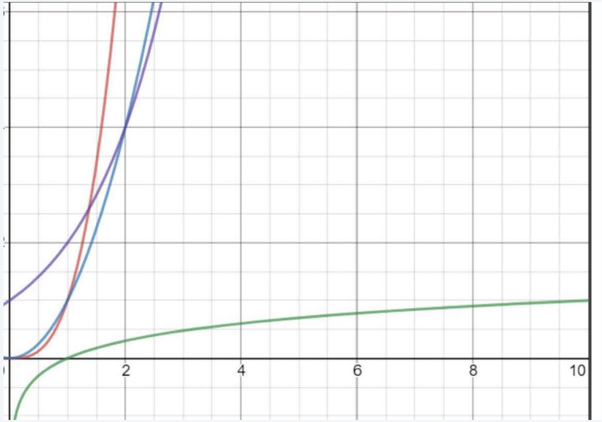
const	log ($\log^c n$)	poly (n^c)	exp (c^n)
	1 $\log^2 n$	$n \log n$ 4	$n^{\log n}$ 7 vs 2^n 8
		n^2 6	$\lim_{n\to\infty} \frac{n^{\lg n}}{2^n}$
		$n \log \log n$ 3	take logs
		$2^{\log_2 n} = n$ 2	$\log^2 n$ vs n
		$n^{\sqrt{2}}$ 5	$<$

Asymptotic Notation

- Big-O Notation
- Big-Omega Notation
- Big-Theta Notation
- Hierarchies of Functions
- **Properties of Asymptotic Notation**

1.1–1.2

Properties of Asymptotic Notation



Big-O, Big-Omega, Big-Theta are all reflexive and transitive.

e.g.
Reflexivity: f(n) is in O(f(n))

Transitivity: If f(n) is in O(g(n)) and g(n) is in O(h(n)), then f(n) is in O(h(n))

Big-Theta is also symmetric, but Big-O and Big-Omega are not.

f(n) is in Theta(g(n)) if and only if f is in O(g(n)) and Omega(g(n))

Symmetry: If f(n) is in Theta(g(n)), then g(n) is in Theta(f(n)) [Theorem 21.2 in Lewis, Zax textbook]

Let f(n) and g(n) be nonnegative functions: Show that O(f(n) +g(n)) = O(max(f(n),g(n))).

i) $O(f(n)+g(n)) \subseteq O(\max(f(n), g(n)))$

for any non-negative function if $h(n) \in O(f(n)+g(n))$

then $h(n) \in O(\max(f(n), g(n)))$

ii) $O(\max(f(n), g(n))) \subseteq O(f(n)+g(n))$

for any nonnegative function if $h(n) \in O(\max(f(n), g(n)))$

then $h(n) \in O(f(n)+g(n))$.

Properties of Asymptotic Notation Practice Problem

Let f(n) and g(n) be nonnegative functions: Show that O(f(n) +g(n)) = O(max(f(n),g(n))).

let h(n) be a nonnegative function

i) if $h(n) \in O(f(n)+g(n))$ then $h(n) \in O(\max(f(n), g(n)))$

if $h(n) \in O(f(n)+g(n))$ then there exists $c, n_0 > 0$ st for all $n \geq n_0$ $0 \leq h(n) \leq c(f(n)+g(n))$.

key observation: $f(n)+g(n) \leq 2\max(f(n), g(n))$

$\Rightarrow 0 \leq h(n) \leq c(f(n)+g(n)) \leq c \cdot 2\max(f(n), g(n))$

let $n^* = n_0$ and $c^* = c \cdot 2$, then

$h(n) \in O(\max(f(n), g(n)))$ by definition of big O.

Properties of Asymptotic Notation Practice Problem

Let f(n) and g(n) be nonnegative functions: Show that O(f(n) +g(n)) = O(max(f(n),g(n))).

Let $h(n)$ be a non-negative function

2) If $h(n) \in O(\max(f(n), g(n)))$ then $h(n) \in O(f(n)+g(n))$

if $h(n) \in O(\max(f(n), g(n)))$ then there exist constants $n_0, c > 0$ such that for any $n \geq n_0$

$0 \leq h(n) \leq c \max(f(n), g(n))$.

Key observation: $\max(f(n), g(n)) \leq f(n) + g(n)$

let $n^* = n_0$ $c^* = c$ for any $n \geq n^*$

$0 \leq h(n) \leq c \max(f(n), g(n)) \leq c^* (f(n) + g(n))$ by definition of Big O $h(n) \in O(f(n) + g(n))$