CS 441: Growth Rates

PhD. Nils Murrugarra-Llerena
nem177@pitt.edu

University of Pittsburgh

Today's topics

- Growth rates of functions
  - Big-O notation and its relation to CS
  - Growth rates of combined functions
  - Big-Omega and Big-Theta notations

Let's define this "big-O notation" that you've probably heard of (and maybe used)

Definition: Let f and g be functions from the set of integers (or real numbers) to the set of real numbers. We say that $f(x)$ is $O\big(g(x)\big)$ if there are constants C and k such that $|f(x)| \leq C|g(x)|$ whenever $x \geq k$.

- C and k are referred to as witnesses which prove the relationship

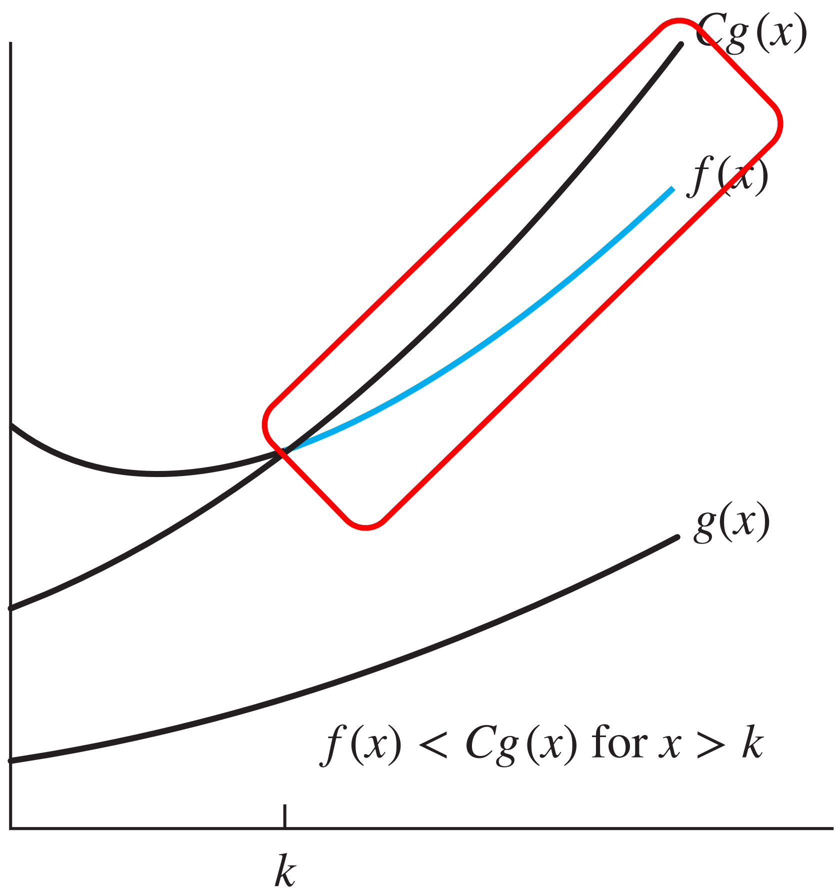
Formally, $O\big(g(x)\big)$ is a set of functions:

$$O\big(g(x)\big) = \{f \mid \exists k, C\ \forall x(x > k \rightarrow |f(x)| \leq C|g(x)|)\}$$

When considering positive values only, we will often drop the absolute value

Examples:

- $2x^2$ is $O(x^2)$ because of witnesses $C = 3$ and $k = 1$:
  $2x^2 \leq 3x^2$ whenever $x \geq 1$
- $3x + 5$ is $O(x)$ because of witnesses $C = 4$ and $k = 5$:
  $3x + 5 \leq 4x$ when $x \geq 5$



$$f(x) \text{ is } O(g(x))$$

The part of the graph f(x) that satisfies f(x)< C g(x) is shown in color.

Why does this matter to computing?

This notation predates its use in computer science by ~70 years!

Consider the intuition behind the math: “The growth of $f(x)$ is bounded above by some multiple of $g(x)$.”

- What does this tell us, if $f(x)$ describes an algorithm’s cost to solve an instance of size x?

Big-O notation is used in algorithm analysis to group algorithms together

- Simple growth rate is more important than exact runtime
- Algorithm analysis describes how algorithms scale to larger and larger problem instances

How to find witnesses to prove a big-O relationship

When $f(x)$ is $O\big(g(x)\big)$, there are infinite witnesses

- e.g., if k works, then any $k' > k$ also works
- but, we only need to identify one pair (k, C) to prove the relationship

Simple key idea: Round up

- To prove that $2x^2 + 3x + 2$ is $O(x^2)$, “round up” each term to a multiple of x^2
- $2x^2 + 3x + 2 \leq 2x^2 + 3x^2 + 2x^2 = 7x^2$, whenever $x \geq 1$
- So, let $C = 7$ and $k = 1$

In general: Pick a threshold where it is easy to calculate an upper bound for $f(x)$ in terms of $g(x)$

But wait, doesn't this mean that any greater $g(x)$ would also work?

In fact, yes!

- $2x^2$ is $O(x^2)$, but also $O(2x^2)$, $O(10x^2)$, $O(x^3)$, and $O(x^4)$…

However, it is most useful to state the most specific or descriptive relationship that you can prove

- Multiplicative constants can be anything and are generally left out
- $O(x^2)$ is a proper subset of $O(x^3)$, so stating $f(x)$ is in the former also implies it is in the latter, but not vice-versa
- If we know it is $O(x^2)$, then stating it is $O(x^3)$ leaves out information unnecessarily
- Nobody cares that $f(x)$ is $O\left(x^{x^{x^x}}\right)$ unless you really can't prove something more specific!

How do we know when we can't use a smaller bound?

Let's see by example: Prove that x^2 is not $O(x)$

- We need to prove that there is no choice of C and k that satisfy the constraints. Let's use contradiction.
- Suppose there is a C and k where $x^2 \leq Cx$ whenever $x \geq k$
- When $x > 0$, we can divide both sides by x to see $x \leq C$
- However, we cannot pick C that satisfies this, since there is no C that is greater than any (arbitrarily large) integer
- This contradiction proves that C and k do not exist where $x^2 \leq Cx$ whenever $x \geq k$

Therefore, x^2 is not $O(x)$!

- That is, x has a strictly smaller rate of growth than x^2

A heuristic for growth rates of polynomials: Drop multiplicative constants and lower-order terms

Theorem: Let $f(x) = a_n x^n + a_{n-1} x^{n+1} + \cdots + a_1 x + a_0$, where each a_i is a real number. Then, $f(x)$ is $O(x^n)$.

- In other words, every n-degree polynomial is $O(x^n)$
- See §3.2.3 for a proof

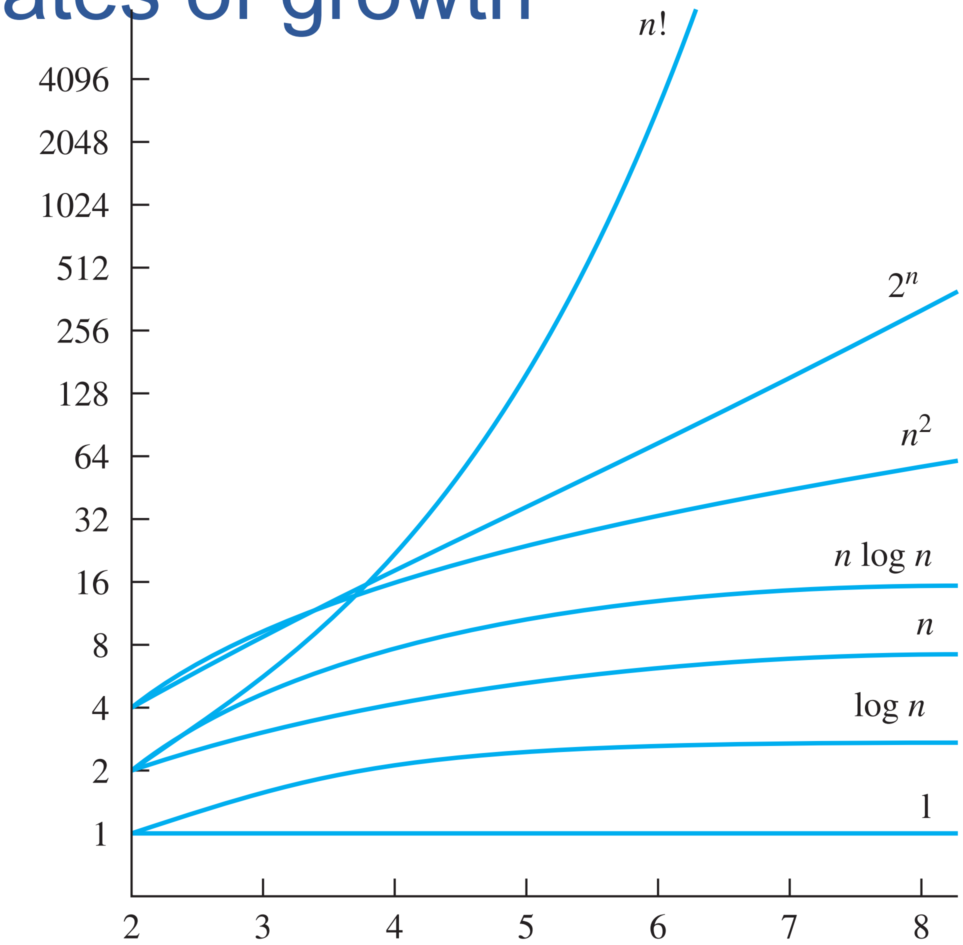
This means that we can calculate growth rates without explicitly finding witnesses

- Drop lower-order terms: $3x^3 + 6x^2 - 3x + 9$ becomes $3x^3$
- Drop multiplicative constants: $3x^3$ becomes x^3
- Thus, $3x^3 + 6x^2 - 3x + 9$ is $O(x^3)$

This informal approach matches our goals with algorithm analysis

- Highest-order term will dominate at scale
- Multiplicative constants are equivalent to hardware choice

Common rates of growth



Growth rates of combined functions

Theorem: If $f_1(x)$ is $O\big(g_1(x)\big)$ and $f_2(x)$ is $O\big(g_2(x)\big)$, then $(f_1 + f_2)(x)$ is $O\big(g(x)\big)$, where $g(x) = \max\big(g_1(x), g_2(x)\big)$ for all x.

In other words, the sum of two functions has a growth rate equal to the max of their individual growth rates

Examples:

- $x^3 + x\log x$ is $O(x^3)$
- $\log x + (\log x)^2$ is $O(\log^2 x)$

Note that this is a generalization of the previous theorem regarding polynomials

Growth rates of combined functions

Theorem: If $f_1(x)$ is $O\big(g_1(x)\big)$ and $f_2(x)$ is $O\big(g_2(x)\big)$, then $(f_1 f_2)(x)$ is $O\big(g_1(x)g_2(x)\big)$.

In other words, the product of two functions has a growth rate equal to the product of their individual growth rates

Examples:

- $(x^2 + x)(x + 5)$ is $O(x^3)$
- $(\log x + \log\log x)(8 + \log x)$ is $O(\log^2 x)$

This is especially useful for analyzing nested loops in algorithm analysis

- As we'll see next time!

Related notations to big-O

Definition: Let f and g be functions from the set of integers (or real numbers) to the set of real numbers. We say that $f(x)$ is $\Omega\big(g(x)\big)$ if there are constants C and k such that $|f(x)| \geq C|g(x)|$ whenever $x \geq k$.

- If big-O represents an asymptotic upper bound, big-Omega represents an asymptotic lower bound
- (Asymptotic = at scale, as x increases toward infinity)

Examples:

- $2x^2$ is $\Omega(x^2)$, $\Omega(x)$, and $\Omega(1)$
  - In addition to being $O(x^2)$, $O(x^3)$, $O(x^4)$, …

When $f(x)$ is both $O\big(g(x)\big)$ and $\Omega\big(g(x)\big)$, we say it is $\Theta\big(g(x)\big)$, so $2x^2$ is $\Theta(x^2)$

- “Big theta”

In-class exercises

On Top Hat

Final thoughts

- Growth rates are commonly expressed using big-O and related notations
- These notations were not developed for computing, but fit well for algorithm analysis
- Next time:
  - Algorithm analysis (Section 3.3)