DMA: Asymptotic growth of functions

Laura Mančinska,
Institut for Matematiske Fag

UNIVERSITY OF COPENHAGEN

Plan for today

- Asymptotic growth of functions
  - g grows at least as fast as f (big-O)
  - g grows faster than f (little-o)
  - g and f have the same order of growth (big-Θ)

- Reading for today
  - Section 2.1. in Notes for Week 3
  - [CLRS] Chapter 3
  - Warning: The asymptotic notation is tricky. Expect this to be challenging. But we can do this! Read at least twice!

Asymptotic analysis: Motivation

- Functions represent running times of algorithms
- The faster a function grows, the worse our algorithm

How should we measure running time?

```
ADDALL(A,n)
    sum ← 0
    For i = 0 thru n-1
        sum ← sum + A[i]
    return sum
```

- $T(n)$ = # of "steps" performed by ADDALL(A,n)
- $T(n) = 2n + 2$ (linear time algorithm)

Actual constants will depend on the software and hardware we execute this on!

How should we measure running time?

```
FIND2(A,n)
    For i = 0 to n-1
        if A[i]=2 then
            return TRUE
    return FALSE
```

- $T(n) =$ # of "steps" taken by `FIND2(A,n)`
- $T(n) = ?$

Running time can depend on the specific input (and not only its size)
So we often focus on the **worst-case running time.**

Worst-case running time

```
FIND2(A,n)
    For i = 0 to n-1
        if A[i]=2 then
            return TRUE
    return FALSE
```

- $T(n)$ = **max** # of "steps" taken by `FIND2(A,n)` over all lists A of size n
- $T(n) = 2n + 1$

Worst-case running time is a function of (only) n

Depending on the algorithm, running-time might or might not be a function of just the input size n.

Analysis of algorithms

Starting point: identity a function $T(n)$ describing

- **The (worst-case) running time** of your algorithm

Goals of asymptotic notation

- Analyze the growth of $T(n)$ in a hardware and software independent manner
- Focus on the large n regime
- Simplify analysis by dropping inessential information (e.g. say that $3n + 5$ and $2n + 100$ are the same asymptotically)

Asymptotic analysis allows to **classify and compare the efficiency** of algorithms more easily

(e.g. linear is better than quadratic; quadratic is better than exponential etc.)

Asymptotic analysis: Formalism

Asymptotically positive functions

Def. We say that a function $f: \mathbb{R}^+ \to \mathbb{R}$ is asymptotically positive if there exists $x_0 \in \mathbb{R}^+$ such that

$$0 < f(x)$$

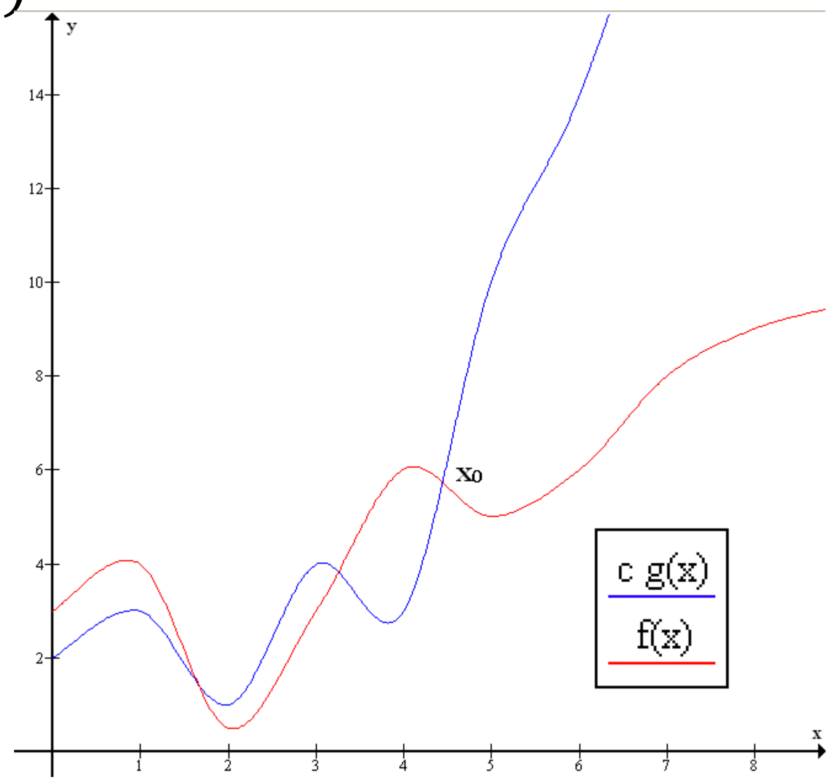
for all $x \geq x_0$.

- Examples: $5, \;\; 2^x, \;\; x^2 - 6x$ are all asymptotically positive
- $100 - x^3$ is not asymptotically positive

Big-O notation: f is $O(g)$

Def. (Big-O) Let $f, g: \mathbb{R}^+ \to \mathbb{R}$ be asymptotically positive. We say that ***f* is *$O(g)$*** if there exists $c > 0$ and $x_0 > 0$ such that for all $x \geq x_0$ we have:

$$f(x) \leq cg(x)$$



Intuition:

g grows at least as fast as f

(informally think: "$f \leq g$")

Big-O notation: f is $O(g)$

Def. (Big-O) Let $f, g \colon \mathbb{R}^+ \to \mathbb{R}$ be asymptotically positive. We say that ***f* is *$O(g)$*** if there exists $c > 0$ and $x_0 > 0$ such that

$$f(x) \leq c g(x)$$

for all $x \geq x_0$.

Notes

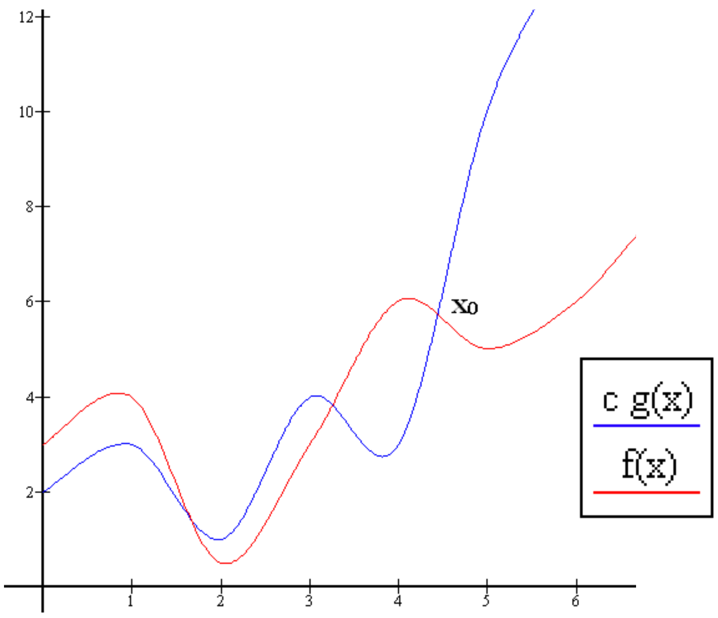
- We write "$f \in O(g)$" or "$f = O(g)$"
- Same definition applies for $f, g \colon \mathbb{Z}^+ \to \mathbb{R}$
- Different texts define big-O slightly differently

Big-O notation: f is $O(g)$

Def. (Big-O) Let $f, g\colon \mathbb{R}^+ \to \mathbb{R}$ be asymptotically positive. We say that f **is** $O(g)$ if there exists $c > 0$ and $x_0 > 0$ such that

$$f(x) \leq cg(x)$$

for all $x \geq x_0$.



Intuition:

g grows at least as fast as f

(informally think: "$f \leq g$")

Q: Suppose f is $O(g)$ for some functions f and g.
Does it follow that $f(x) \leq g(x)$ for large values of x?

g grows faster than f

- Recall: Functions represent running times
- So the algorithm whose running time is f is better than the one described by g

Asymptotic growth: **little-o notation**

Def. (Little-o) Let $f, g: \mathbb{R}^+ \to \mathbb{R}$ be asymptotically positive. We say that ***f* is *$o(g)$*** if for any $c > 0$ there exists $x_0 > 0$ such that

$$f(x) < cg(x)$$

for all $x \geq x_0$.

Intuition: g grows faster than f (think: "$f < g$")

Notes

- Write "$f \in o(g)$" or "$f = o(g)$"

g and f grow at the same rate asymptotically

Asymptotic growth: **big-Θ**

Def. (Big-Θ) Let $f, g: \mathbb{R}^+ \to \mathbb{R}$ be asymptotically positive.
We say that ***f* is $\Theta(g)$** if $f = O(g)$ and $g = O(f)$.

Recall:

- $f = O(g)$: g grows at least as fast as f
- $g = O(f)$: f grows at least as fast as g

Intuition of "f is $\Theta(g)$": f and g grow at the same rate asymptotically (think: "$f = g$")

Q: Suppose f is $\Theta(g)$. Does this mean that g is $\Theta(f)$?

Asymptotic growth: Summary

Def. (Big-O) Let $f, g: \mathbb{R}^+ \to \mathbb{R}$ be asymptotically positive. We say that ***f* is $O(g)$** if there exists $c > 0$ and $x_0 > 0$ such that

$$f(x) \leq cg(x)$$

for all $x \geq x_0$.

Intuition

Informally (**!**)

- $f = O(g)$: g grows at least as fast as f — "$f \leq g$"
- $f = o(g)$: g grows faster than f — "$f < g$"
- $f = \Theta(g)$: g and f grow at the same rate — "$f = g$"

Careful: Analogy only goes so far. For example, there are functions such that $f \neq O(g)$ and $g \neq O(f)$.

Asymptotic growth: further remarks

little-o implies Big-O

Thm. If $f(x)$ is $o(g(x))$ then

- $f(x)$ is $O(g(x))$ and
- $g(x)$ is not $O(f(x))$

Intuition(!) behind the thm

If g grows faster than f then

- g grows at least as fast as f
- f does not grow at least as fast as g

Classes of functions

- Polynomials
- Exponentials
- Logarithms

Polynomials

Def. Polynomials are functions of the form

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

- $a_n, \dots, a_0 \in \mathbb{R}$ (or $\mathbb{Q}$ or $\mathbb{Z}$) are called coefficients.
- $n \in \{0,1,2,\dots\}$. If $a_n \neq 0$, then n is the degree of $p(x)$.
  Write: $\deg(p) = n$
- Examples: $p_1(x) = \frac{1}{2}x^2 - 5,\ p_2(x) = 20x + \pi$

More general powers: x^r where $r \in \mathbb{R}$.

Exponentials and logarithms

Def. **Exponentials** are functions of the form

$$f(y) = b^y$$

where $b > 0$ is a constant (called the base).

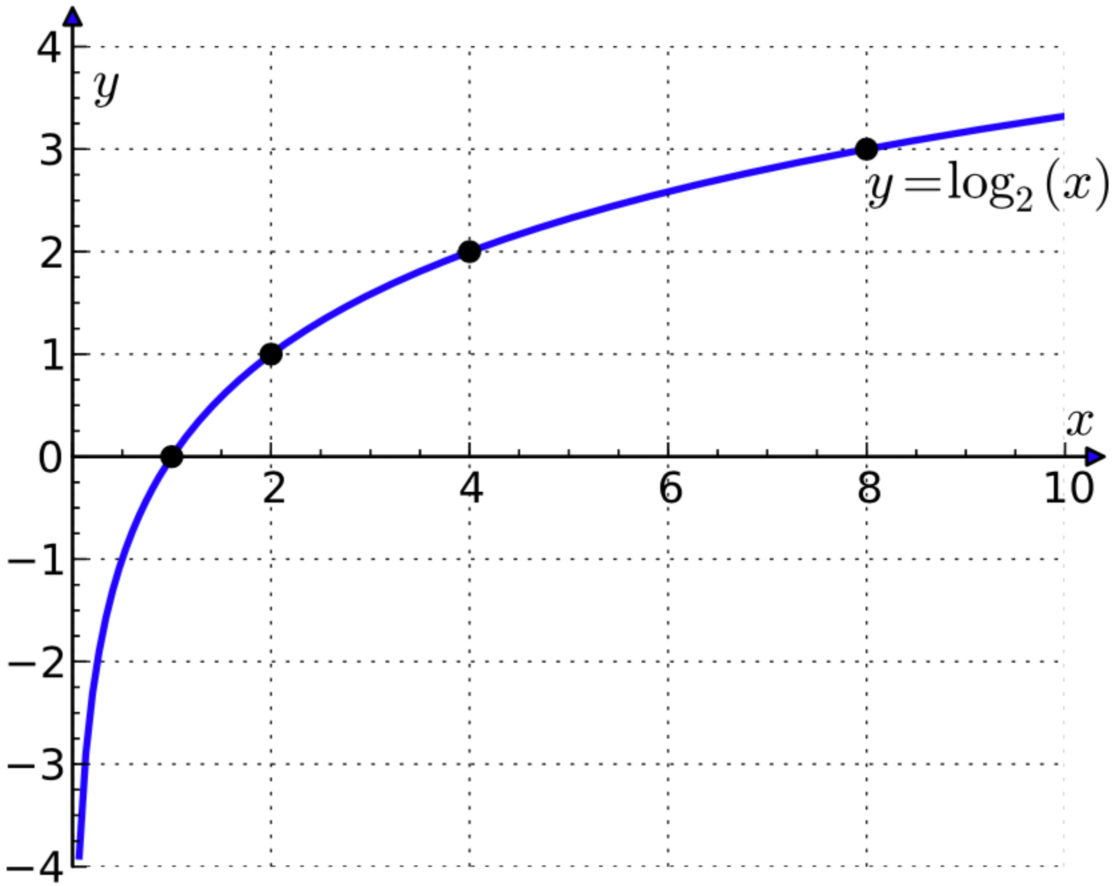
- y can be any real number
- Examples: 2^y, e^y, $\left(\frac{1}{2}\right)^y$, $(1.25)^y$
- Q: Fix some base $b > 0$. Is b^y an increasing function?

Base-b logarithm:

$$\log_b x \stackrel{\text{def}}{=} y \text{ such that } b^y = x$$

- We will assume that $x > 0$ and focus on $b > 1$.

Logarithms : Example



Properties of logarithms

Thm. For all $b, b', x, x_1, x_2 > 0$ and all $r \in \mathbb{R}$ we have

$$\log_b x^r = r \log_b x$$

$$\log_b(x_1 x_2) = \log_b x_1 + \log_b x_2$$

$$\log_b(x) = \left(\frac{1}{\log_{b'} b}\right) \log_{b'} x$$

where none of the logarithm bases are 1.

Order the following functions from slowest to fastest growing

- 3^n
- $\log_2(n)$
- 200
- n^2
- n^3
- 2^n
- $\log_3(n)$

Asymptotic growth: Rules

Asymptotic notation: Classes of functions

R4: const$<$ **log**: any $c\ >\ 0$ is $o(\log_a(x))$ for all $a > 1$.

R5: log$<$**power**: $\log_a(x)$ is $o(x^b)$ for all $a > 1$, $b > 0$.

R6: power$<$**exp** x^a is $o(b^x)$ for all a and all $b > 1$.

Informally:

Constants < Logarithms < Powers (polynomials) < Exponentials

- Since $f = o(g)$ implies that $f = O(g)$, the above rules hold for big-O as well.

Classes of functions and orders (memorize)

Classes of functions arranged from slower to faster growth

Constants, Logarithms, Positive powers, Exponentials $(\boldsymbol{b} > \mathbf{1})$

1.25 $\log_5(n)$ $n^{3.7}$ 2^n

Ordering within a class:

Which of the two functions grow faster or are they of the same order?

- **Constants** 0.0001 1000
- **Logarithms** $\log_5(n)$ $\log_2(n)$
- **Powers** n^{200} n^2
- **Exponentials** 2^n 5^n

Classes of functions and orders (memorize)

Classes of functions arranged from slower to faster growth

Constants,	**Logarithms,**	**Positive powers,**	**Exponentials** $(b > 1)$
1.25	$\log_5(n)$	$n^{3.7}$	2^n

Ordering within a class:

Which of the two functions grow faster or are they of the same order?

• **Constants**	0.0001	1000	All grow at the same rate
• **Logarithms**	$\log_5(n)$	$\log_2(n)$	All grow at the same rate
• **Powers**	n^{200}	n^2	Larger power $\Longrightarrow$ faster growth
• **Exponentials**	2^n	5^n	Larger base $\Longrightarrow$ faster growth

Simplification rules

R1: Overall constants can be ignored:

$cf(x)$ is $\Theta(f(x))$ for any constant $c > 0$

R2: Only the highest-order term matters: polynomials

$p(x)$ is $\Theta(x^d)$ where $p(x)$ is a polynomial of degree d

R3: Only the highest-order term matters:

If $f(x) = \mathrm{o}(g(x))$ then $\mathrm{c}_1 g(x) + \mathrm{c}_2 f(x)$ is $\Theta(g(x))$ for any constants $c_1 > 0$ and $c_2 \in \mathbb{R}$

Intuition: think of $f(x) = \mathrm{o}(g(x))$ as

"f is negligible in comparison to g"