

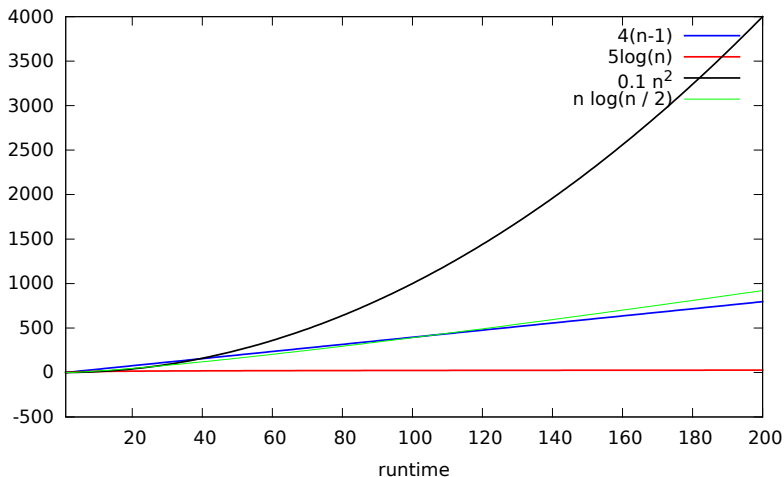
# Defining O-notation (recap)

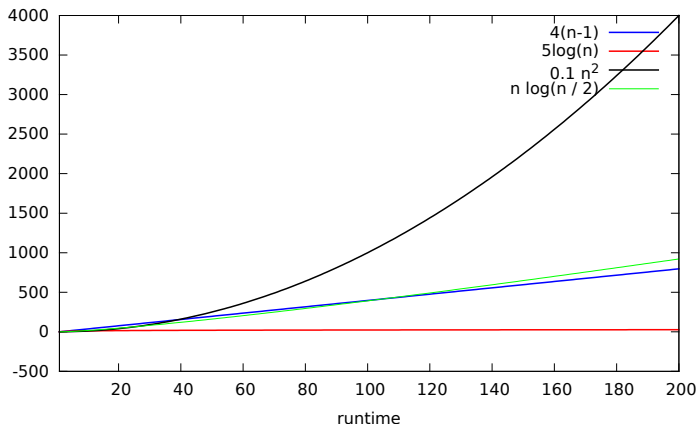
## COMS20010 (Algorithms II)

John Lapinskas, University of Bristol

# Why O-notation?

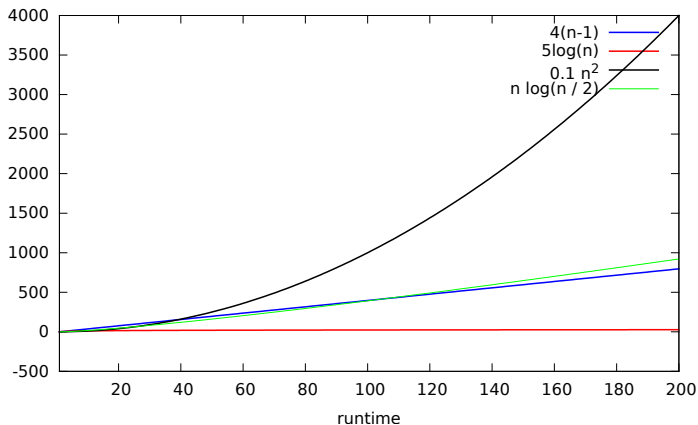
**Intuition:** As input sizes get large, asymptotic growth rate matters more than constant factors. Also, constant factors are implementation-dependent. So we focus on growth rate.





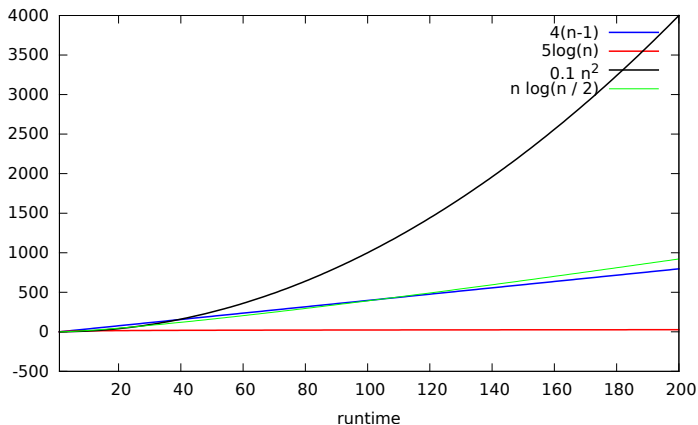
We would like  $f(n) \in O(g(n))$  to mean:

- $f(n)$  “grows no faster than”  $g(n)$ , ignoring constant factors.



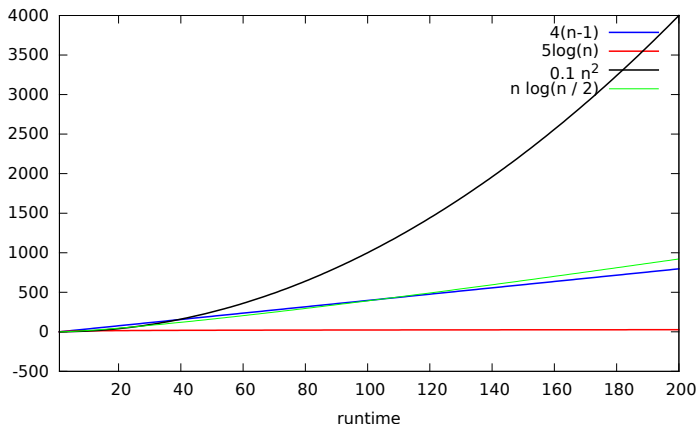
We would like  $f(n) \in O(g(n))$  to mean:

- $f(n)$  “grows no faster than”  $g(n)$ , **ignoring constant factors**.
- There exists  $C > 0$  such that  $f(n)$  “grows no faster than”  $C \cdot g(n)$ .



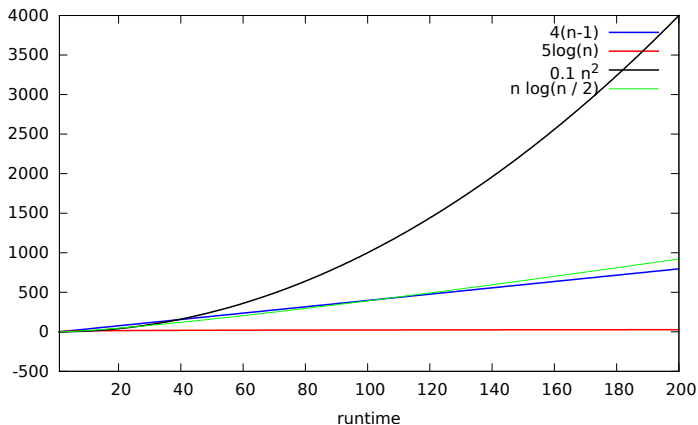
We would like  $f(n) \in O(g(n))$  to mean:

- There exists  $C > 0$  such that  $f(n)$  **“grows no faster than”**  $C \cdot g(n)$ .
- There exists  $C > 0$  such that  $f(n) \leq C \cdot g(n)$  whenever  $n$  is sufficiently large.



We would like  $f(n) \in O(g(n))$  to mean:

- There exists  $C > 0$  such that  $f(n) \leq C \cdot g(n)$  whenever  $n$  is **sufficiently large**.
- There exist  $C, n_0 > 0$  such that  $f(n) \leq C \cdot g(n)$  whenever  $n \geq n_0$ .



We would like  $f(n) \in O(g(n))$  to mean:

There exist  $C, n_0 > 0$  such that  $f(n) \leq C \cdot g(n)$  whenever  $n \geq n_0$ . ✓

This rigorous definition is “just” a more precise version of our intuition.

# Other O-notation

$f(n) \in O(g(n))$  is good notation for “ $f$  grows no faster than  $g$ , ignoring constants”. But what if we want to say “ $g$  grows no slower than  $f$ ”?

Notation	Intuitive meaning	Analogue
$f(n) \in O(g(n))$	$f$ grows at most as fast as $g$	$\leq$
$f(n) \in \Omega(g(n))$	$f$ grows at least as fast as $g$	$\geq$
$f(n) \in \Theta(g(n))$	$f$ at the same rate as $g$	$=$
$f(n) \in o(g(n))$	$f$ grows strictly less fast than $g$	$<$
$f(n) \in \omega(g(n))$	$f$ grows strictly faster than $g$	$>$

Notation	Formal definition
$f(n) \in O(g(n))$	$\exists C, n_0: \forall n \geq n_0: f(n) \leq C \cdot g(n)$
$f(n) \in \Omega(g(n))$	$\exists c, n_0: \forall n \geq n_0: f(n) \geq c \cdot g(n)$
$f(n) \in \Theta(g(n))$	$\exists c, C, n_0: \forall n \geq n_0: c \cdot g(n) \leq f(n) \leq C \cdot g(n)$
$f(n) \in o(g(n))$	$\forall C: \exists n_0: \forall n \geq n_0: f(n) \leq C \cdot g(n)$
$f(n) \in \omega(g(n))$	$\forall c: \exists n_0: \forall n \geq n_0: f(n) \geq c \cdot g(n)$



# Examples

**Example 1:** Prove  $n^2 - 5n + 12 \in \Theta(n^2)$  directly from the definition.

Remember the definition: proving  $n^2 - 5n + 12 \in \Theta(n^2)$  **means** proving there exist  $c$ ,  $C$  and  $n_0$  such that  $cn^2 \leq n^2 - 5n + 12 \leq Cn^2$  for all  $n \geq n_0$ .

We expect  $n^2 - 5n + 12 \approx n^2$  for large  $n$ , so we could e.g. set  $c = 1/2$  and  $C = 2$  and solve the quadratic. But let's be lazy! No need to optimise.

We have

$$\begin{aligned}n^2 - 5n + 12 &\leq n^2 + 12 = n^2\left(1 + \frac{12}{n^2}\right), \\n^2 - 5n + 12 &\geq n^2 - 5n = n^2\left(1 - \frac{5}{n}\right).\end{aligned}$$

Looking at it like this, it's much easier to see that

$$\begin{aligned}n^2 - 5n + 12 &\leq 13n^2 \text{ for all } n \geq 1, \\n^2 - 5n + 12 &\geq n^2/2 \text{ for all } n \geq 10 \text{ (so } \frac{5}{n} \leq \frac{1}{2}\text{)}.\end{aligned}$$

So we prove  $n^2 - 5n + 12 \in \Theta(n^2)$  by taking  $c = \frac{1}{2}$ ,  $C = 13$ , and  $n_0 = 10$ .

# Examples

**Example 2:** Prove  $n! \in \omega(2^n)$  directly from the definition.

Remember the definition: proving  $n! \in \omega(2^n)$  **means** proving that for all  $c > 0$ , there exists  $n_0$  such that for all  $n \geq n_0$ ,  $n! \geq c \cdot 2^n$ .

So we're given a constant  $c$ , and we need to show  $n! \geq c \cdot 2^n$  when  $n$  is sufficiently large. Remember we have

$$n! = \underbrace{n \cdot (n-1) \cdot \dots \cdot 1}_{n \text{ terms}}, \qquad 2^n = \underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{n \text{ terms}}.$$

So we have a lot of wiggle room to bound things term-by-term.

Let's use the fact that  $n! \geq 4^{n-3} = 2^n \cdot 2^{n-6}$ .

Thus  $n! \geq c \cdot 2^n$  whenever  $2^{n-6} \geq c$ , i.e. whenever  $n \geq \log c + 6$ .

So we prove  $n! = \omega(2^n)$  by taking  $n_0 \geq \log c + 6$ .

# Multi-variable O-notation

We will often need O-notation for functions of more than one variable.

For example, an algorithm running on an  $n$ -vertex  $m$ -edge graph will often have running time depending on both  $m$  and  $n$ .

What does it mean to say that e.g.  $f(m, n) \in O(mn)$  or  $f(m, n) \in \Theta(m^2 \log n)$ ?

The only difference is that instead of requiring  $n$  to be sufficiently large, we require **all** variables to be sufficiently large.

For example,  $f(m, n) \in O(g(m, n))$  when there exist  $C$ ,  **$m_0$**  and  **$n_0$**  such that  $f(m, n) \leq C \cdot g(m, n)$  whenever  $m \geq m_0$  **and**  $n \geq n_0$ .

All the useful properties of single-variable O-notation (see next video!) carry over to multi-variable O-notation, so e.g. if  $f(m, n) \in O(g(m, n))$  and  $f(m, n) \in \Omega(g(m, n))$  then we still have  $f(m, n) \in \Theta(g(m, n))$ .

# A pedantic clarification

O-notation can behave strangely with negative functions.

But we only care about O-notation for running times, which are positive!

So whenever you are asked to prove something general about O-notation in this course, you can assume the functions involved are non-negative.

*But* logarithms get used to bound running times all the time, and e.g.  $n \log(n/100)$  is negative for  $n = 2$ . Since it's positive for large  $n$ , we'd still like to be able to say e.g.  $n \log(n/100) \in \Theta(n \log n)$ .

So the formal requirement is that the functions involved are **eventually non-negative** — that is, before we can say  $f(n) \in O(g(n))$  or similar, we require that  $f(n), g(n) \geq 0$  for all sufficiently large  $n$ .

Any fact that holds about O-notation for non-negative functions will also hold for eventually non-negative functions, by taking  $n_0$  large enough that “eventually non-negative” becomes “non-negative”.