Discrete Mathematics

	topic
week 1	***Sets***
week 2	***Functions***
week 3	***Counting***
week 4	***Relations***
week 5	***Sequences***
week 6	***Modular Arithmetic***
week 7	***Asymptotic Notation***
week 8	***Orders***

Jonathan Barrett
jonathan.barrett@cs.ox.ac.uk
Material by Andrew Ker
University of Oxford
Department of Computer Science

Discrete Mathematics

Jonathan Barrett
jonathan.barrett@cs.ox.ac.uk
Material by Andrew Ker
University of Oxford
Department of Computer Science

Chapter 7: Asymptotic Notation

"Big-O" Notation

Suppose that f and g are both real-valued functions with domain $\mathbb{N}$.

We write $f(n) = O(g(n))$ if there is a real number c and an integer N with

$$|f(n)| \leq c|g(n)| \text{ for all } n \geq N.$$

and say that f is **asymptotically bounded** by g.
(Sometimes we write $f = O(g)$).

We mean that f "grows no faster than" g, for large enough values of the domain and without regard to constant multiples.

"Big-O" Notation

Suppose that f and g are both real-valued functions with domain $\mathbb{N}$.

We write $f(n) = O(g(n))$ if there is a real number c and an integer N with

$$|f(n)| \leq c|g(n)| \text{ for all } n \geq N.$$

and say that f is **asymptotically bounded** by g.
(Sometimes we write $f = O(g)$).

We mean that f "grows no faster than" g, for large enough values of the domain and without regard to constant multiples.

- The same definition applies to functions with domain $(0, \infty)$ or $\mathbb{R}$.
- When f and g are positive (as in most CS applications), we can forget the $|\cdot|$.

"Big-O" Notation

Suppose that f and g are both real-valued functions with domain $\mathbb{N}$.

We write $f(n) = O(g(n))$ if there is a real number c and an integer N with

$$|f(n)| \leq c|g(n)| \text{ for all } n \geq N.$$

and say that f is **asymptotically bounded** by g.
(Sometimes we write $f = O(g)$).

We mean that f "grows no faster than" g, for large enough values of the domain and without regard to constant multiples.

- The same definition applies to functions with domain $(0, \infty)$ or $\mathbb{R}$.
- When f and g are positive (as in most CS applications), we can forget the $|\cdot|$.

<u>Examples</u>

$$n^2 + n = O(n^2), \quad n^3 + n^2 + \log n = O(n^3),$$
$$n^5 = O(n^n), \quad n! = O(n^n)$$

Warnings

Even though $f(n) = O(g(n))$ is *written as* an equation, it is **not** an equation.

<u>Examples</u> We can have $f_1 = O(g)$ and $f_2 = O(g)$ but $f_1 \neq f_2$.
We can have $f = O(g)$ but $g \neq O(f)$.

To emphasise this, some people write $f(n) \in O(g(n))$ instead.

Warnings

Even though $f(n) = O(g(n))$ is *written as* an equation, it is **not** an equation.

<u>Examples</u> We can have $f_1 = O(g)$ and $f_2 = O(g)$ but $f_1 \neq f_2$.

We can have $f = O(g)$ but $g \neq O(f)$.

To emphasise this, some people write $f(n) \in O(g(n))$ instead.

$f(n) = O(g(n))$ behaves a bit like "$f \leq g$":

<u>Claim</u> If $f = O(g)$ and $g = O(h)$ then $f = O(h)$.

(More precisely, the relation is reflexive and transitive, but it is not antisymmetric.)

Techniques for Proving Big-O

Note that

$$|f(n)| \leq c|g(n)|$$

is equivalent to

$$|f(n)/g(n)| \leq c.$$

And when f and g are positive, this is equivalent to

$$\log \frac{f(n)}{g(n)} \leq c'.$$

When it is not simple to find N and c directly, try computing with the log quotient instead. Often, some simple calculus can be applied.

Techniques for Proving Big-O

Note that

$$|f(n)| \leq c|g(n)|$$

is equivalent to

$$|f(n)/g(n)| \leq c.$$

And when f and g are positive, this is equivalent to

$$\log \frac{f(n)}{g(n)} \leq c'.$$

When it is not simple to find N and c directly, try computing with the log quotient instead. Often, some simple calculus can be applied.

Examples $\quad n^2 = O(2^n), \quad 2^n \neq O(n^{100})$

Sentences of the Form $\exists x. \forall y. P$

To prove something like:

<u>Claim</u> There is something (x) such that, for all things (y), P is true.

We must find the correct value of x, and then prove that it works.

In practice, we often begin the proof without knowing the value of x, hoping to fill it in later.

Be careful to use the correct logical connectives. With this style of proof, it is likely that each line follows from the <u>next</u> line, not necessarily from the previous line.

Tail Behaviour

Although asymptotic behaviour describes the "tails" (large values of the domain), it applies equivalently to the whole function.

Claim As long as the domain of f and g is $\mathbb{N}$ and g is nonzero, $f(n) = O(g(n))$ is equivalent to:

There exists a real number c such that $|f(n)| \leq c|g(n)|$ for all $n \in \mathbb{N}$.

Tail Behaviour

Although asymptotic behaviour describes the "tails" (large values of the domain), it applies equivalently to the whole function.

Claim As long as the domain of f and g is $\mathbb{N}$ and g is nonzero, $f(n) = O(g(n))$ is equivalent to:

There exists a real number c such that $|f(n)| \leq c|g(n)|$ for all $n \in \mathbb{N}$.

This is also true if the domain is $[0, \infty)$ and f and g are continuous functions.

Asymptotics of n!

We have already seen that $n! = O(n^n)$, but this is a loose bound.

<u>Lemma</u> For all $x > 0$,

$$(\tfrac{1}{2} + \tfrac{1}{x}) \log(1 + x) > 1.$$

Asymptotics of n!

We have already seen that $n! = O(n^n)$, but this is a loose bound.

Lemma For all $x > 0$,

$$(\tfrac{1}{2} + \tfrac{1}{x}) \log(1 + x) > 1.$$

Claim $n! = O(n^{n+\frac{1}{2}} \exp(-n))$.

Asymptotics of $n!$

We have already seen that $n! = O(n^n)$, but this is a loose bound.

Lemma For all $x > 0$,

$$\left(\tfrac{1}{2} + \tfrac{1}{x}\right) \log(1+x) > 1.$$

Claim $n! = O(n^{n+\frac{1}{2}} \exp(-n))$.

This is a tight bound. In fact,

$$\frac{n!}{n^{n+\frac{1}{2}} \exp(-n)} \to \sqrt{2\pi}.$$

which leads to Stirling's formula, an approximation to $n!$ for large n:

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

Asymptotics and Recurrence Relations

Sometimes we cannot solve a recurrence relation, e.g.

$$x_1 = 0, \quad x_n = 2x_{\lfloor \frac{n}{2} \rfloor} + n \text{ for } n \geq 2$$

but we can prove something about its order of growth.

<u>Claim</u> For the sequence above,

$$x_n = O(n \log_2 n)$$

Asymptotics and Recurrence Relations

Sometimes we cannot solve a recurrence relation, e.g.

$$x_1 = 0, \quad x_n = 2x_{\lfloor \frac{n}{2} \rfloor} + n \text{ for } n \geq 2$$

but we can prove something about its order of growth.

<u>Claim</u> For the sequence above,

$$x_n = O(n \log_2 n)$$

<u>Proof</u> We prove $x_n \leq cn \log_2 n$ by induction on n, but when we start constructing the proof we do not know c, or whether it is true for all n or just for $n \geq N$.

Asymptotics and Recurrence Relations

Sometimes we cannot solve a recurrence relation, e.g.

$$x_1 = 0, \quad x_n = 2x_{\lfloor \frac{n}{2} \rfloor} + n \text{ for } n \geq 2$$

but we can prove something about its order of growth.

<u>Claim</u> For the sequence above,

$$x_n = O(n \log_2 n)$$

<u>Proof</u> We prove $x_n \leq cn \log_2 n$ by induction on n, but when we start constructing the proof we do not know c, or whether it is true for all n or just for $n \geq N$.

The same techniques work if the initial condition is altered, but the base cases of the induction become more complicated.

Diversion: The "Master Theorem"

If $f(n) = O(g(n))$ is analogous to "$f \leq g$", we also need an analogy to "$f \geq g$" and something related to "$f = g$":

We write	$f(n) = \Omega(g(n))$	if	$g(n) = O(f(n))$
and	$f(n) = \Theta(g(n))$	if	$g(n) = O(f(n)) \quad \& \quad f(n) = O(g(n))$

Diversion: The "Master Theorem"

This is a standard method for writing down the asymptotic behaviour of recurrence relations of a certain type.

<u>Theorem</u> Consider the recurrence $t_n = a t_{\frac{n}{b}} + f(n)$

...

Diversion: The “Master Theorem”

This is a standard method for writing down the asymptotic behaviour of recurrence relations of a certain type.

<u>Theorem</u> Consider the recurrence $t_n = a t_{\frac{n}{b}} + f(n)$ where $t_{\frac{n}{b}}$ means either $t_{\lfloor \frac{n}{b} \rfloor}$ or $t_{\lceil \frac{n}{b} \rceil}$, and the boundary conditions ensure that the sequence is eventually positive. Then

- If $f(n) = O(n^k)$ with $k < \log_b a$ then $t_n = \Theta(n^{\log_b a})$
- If $f(n) = \Theta(n^k)$ with $k = \log_b a$ then $t_n = \Theta(n^{\log_b a} \log n)$
- If $f(n) = \Omega(n^k)$ with $k > \log_b a$ then $t_n = \Theta(f(n))$

Diversion: The "Master Theorem"

This is a standard method for writing down the asymptotic behaviour of recurrence relations of a certain type.

<u>Theorem</u> Consider the recurrence $t_n = at_{\frac{n}{b}} + f(n)$
where $t_{\frac{n}{b}}$ means either $t_{\lfloor \frac{n}{b} \rfloor}$ or $t_{\lceil \frac{n}{b} \rceil}$, and the boundary conditions ensure that the sequence is eventually positive. Then

$at_{\frac{n}{b}}$ dominates

- If $f(n) = O(n^k)$ with $k < \log_b a$ then $t_n = \Theta(n^{\log_b a})$

asymp. equal terms

- If $f(n) = \Theta(n^k)$ with $k = \log_b a$ then $t_n = \Theta(n^{\log_b a} \log n)$
- If $f(n) = \Omega(n^k)$ with $k > \log_b a$ then $t_n = \Theta(f(n))$

$f(n)$ dominates

This doesn't solve every recurrence relation of the form given.

Discrete Mathematics

Jonathan Barrett
jonathan.barrett@cs.ox.ac.uk
Material by Andrew Ker
University of Oxford
Department of Computer Science

End of Chapter 7