Introduction to Algorithms and Data Structures

Lecture 4: More asymptotics: O, Ω and Θ

John Longley

School of Informatics
University of Edinburgh

29 September 2022

Where we're heading . . .

Recall our runtime functions T_I, T_M for **InsertSort**, **MergeSort**.
We've seen that T_M grows slowly relative to T_I: $T_M = o(T_I)$.

Can we place growth rates of T_I, T_M on some absolute scale?

E.g. consider the following hierarchy of 'simple' functions:

$$\begin{array}{lll} f_0(n) = 1 & f_1(n) = \lg n & f_2(n) = \sqrt{n} \\ f_3(n) = n & f_4(n) = n \lg n & f_5(n) = n^2 \\ f_6(n) = n^3 & f_7(n) = 2^n & f_8(n) = 2^{2^n} \quad \cdots \end{array}$$

Here $f_0 \in o(f_1)$, $f_1 \in o(f_2)$, . . .

Which of the above functions do T_I and T_M most closely 'resemble' in their essential growth rate?

The big guys: O, Ω, Θ

We're going to define a relation

$$f \text{ is } \Theta(g)$$

Read as 'f has same essential growth rate as g'.

Often used to classify 'complicated' functions via 'simple' ones.

E.g. it will turn out that T_I is $\Theta(n^2)$, and T_M is $\Theta(n \lg n)$.

Approach: First define

f is $O(g)$	'f grows no faster than g'
f is $\Omega(g)$	'f grows no slower than g'

Then say:

$$f \text{ is } \Theta(g) \iff f \text{ is } O(g) \text{ and } f \text{ is } \Omega(g).$$

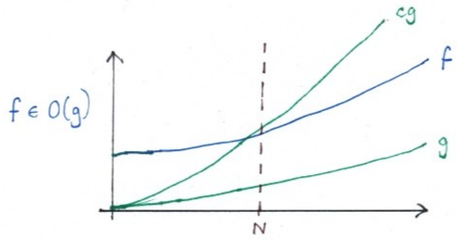
Big O

The spirit of asymptotics is that:

- we only care about behaviour 'in the limit' — can discard 'small' values of n,
- constant scaling factors are washed out.

So let's say f grows no faster than g, if f is eventually bounded above by some (sufficiently large) multiple Cg of g:

$$\exists C > 0.\ \exists N.\ \forall n \geq N.\ f(n) \leq Cg(n)$$



Write as f is $O(g)$, and call g an asymptotic upper bound for f.

Big O: an example

Suppose $f(n) = 3n + \sqrt{n}$ and $g(n) = n$.

Claim: f is $O(g)$. Or more simply, f is $O(n)$.

Proof: Need to show

$$\exists C.\ \exists N.\ \forall n \geq N.\ 3n + \sqrt{n} \leq Cn$$

Take $C = 4$, $N = 1$.

Then for all $n \geq N = 1$, we have $\sqrt{n} \leq n$, so

$$3n + \sqrt{n} \ \leq\ 4n \ =\ Cn$$

Intuition: $3n$ is the ‘dominant’ term; $\sqrt{n}$ is ‘small change’.

Comparing o and O

We've defined:

$$
\begin{array}{lll}
f \text{ is } o(g) & \text{means} & \forall c > 0.\ \exists N.\ \forall n \geq N.\ f(n) < cg(n) \\
f \text{ is } O(g) & \text{means} & \exists C > 0.\ \exists N.\ \forall n \geq N.\ f(n) \leq Cg(n)
\end{array}
$$

- For o we require that *any* multiple of g eventually overtakes f.
- For O it's enough that *some* multiple of g does.

So $f = o(g)$ implies $f = O(g)$.
But not conversely: e.g. $f = O(f)$ for any f, but f is never $o(f)$.

Loosely, can think of o as like $<$, O as like $\leq$.

Notation: Again, $O(g)$ is officially a set:

$$O(g) \ = \ \{f \ \mid \ \exists C \geq 0.\ \exists N.\ \forall n \geq N.\ f(n) \leq Cg(n)\}$$

But common to write e.g. $f = O(g)$ for $f \in O(g)$.

Big O: more examples

Example 1: Let $f(n) = (5n + 4)(7n + 100)$. Is $f = O(n^2)$?
YES!

Informal justification: The dominant term is $35n^2$; the rest is small change that is clearly $o(n^2)$. So f is $O(n^2)$.

Rigorous justification: Want to show:

$$\exists C.\ \exists N.\ \forall n \geq N.\ (5n + 4)(7n + 100) \leq Cn^2$$

Note that

- $5n + 4 \leq 6n$ once $n \geq 4$
- $7n + 100 \leq 8n$ once $n \geq 100$.

So for all $n \geq 100$, we have $f(n) \leq 48n^2$.
In other words, $C = 48$, $N = 100$ will work.

A bit of freedom here . . .

We wanted to show

$$\exists C.\ \exists N.\ \forall n \geq N.\ (5n+4)(7n+100) \leq Cn^2$$

We did this by picking $C = 48,\ N = 100$.

There's some freedom of choice here.
By picking a larger C, can often get away with a smaller N.

E.g. once $n \geq 4$, have $5n+4 \leq 6n$ and $7n+100 \leq 32n$.
So could equally well take $C = 6 \times 32 = 192,\ N = 4$.

Advice: Make life easy for yourself!

More examples

Example 2: Let $f(n) = (5n + 4)(7n + 100)$. Is $f = O(n^3)$?
YES!

We've already shown

$$\forall n \geq 100.\, f(n) \leq 48n^2$$

So certainly

$$\forall n \geq 100.\, f(n) \leq 48n^3$$

Here we say $O(n^3)$ is an asymptotic upper bound for f, though not a tight upper bound.

We'd write $f = \Theta(n^3)$ to mean n^3 was an asymptotic upper *and* lower bound (hence tight). Not true here!

Some authors are less precise in distinguishing O and Θ (see CLRS, end of Chapter 3). But if Θ applies, it's fine only to mention O (or Ω) if that's the important bit.

More examples

Example 3: Is $2^{2n} = O(2^n)$? **NO!**

Informal justification: The ratio $2^{2n}/2^n$ is 2^n, which tends to ∞ and so will eventually exceed any given constant C. In fact, $2^{2n} = \omega(2^n)$.

Rigorous justification: Want to show:

$$\neg\,(\exists C > 0.\ \exists N.\ \forall n \geq N.\ 2^{2n} \leq C.2^n)$$

in other words

$$\forall C > 0.\ \forall N.\ \exists n \geq N.\ 2^{2n} > C.2^n$$

Given any $C > 0$ and N, take any $n > \max(N, \lg C)$. Then $2^n > C$, so $2^{2n} > C.2^n$.

Moral: Do ‘constant factors’ matter? Depends where they occur!

Big O: final example

Example 4: Is $\lg(n^7) = O(\lg n)$? **YES!**

Note that $\lg(n^7) = 7 \lg n$. So $C = 7$, $N = 1$ will do.

Big Ω

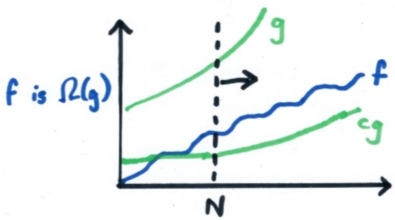
Ω is dual to O. Read f is $\Omega(g)$ as: 'f grows no slower than g', or 'g is an asymptotic lower bound for f'.

E.g. for some runtime function $T(n)$:

- $T(n) = O(g)$ says runtime is not essentially worse than $g(n)$,
- $T(n) = \Omega(g)$ says runtime is not essentially better than $g(n)$.

$f = \Omega(g)$ says f is eventually bounded below by some (sufficiently small) multiple cg of g:

$$\exists c > 0.\ \exists N.\ \forall n \geq N.\ cg(n) \leq f(n)$$



Not hard to show $f = \Omega(g) \iff g = O(f)$.

Big Ω: example

Is it true that $n - \sqrt{n}$ is $\Omega(n)$? **YES!**

Informal justification: $\sqrt{n}$ becomes negligible relative to n when n is large. So growth rate of $n - \sqrt{n}$ is essentially that of n.

Rigorous justification: Want to show:

$$\exists c.\ \exists N.\ \forall n \geq N.\ cn \leq n - \sqrt{n}$$

Take $c = 1/2,\ N = 4$.

Then for all $n \geq N = 4$, we have $\sqrt{n} \leq n/2$, so

$$n - \sqrt{n} \ \geq\ n - n/2 \ =\ n/2 \ =\ cn$$

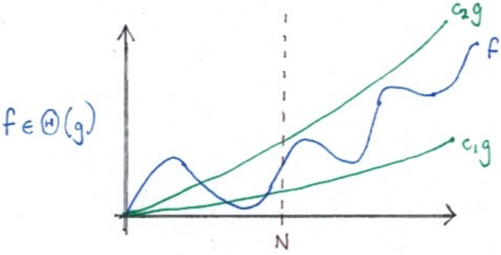
Big Θ

Can now capture the idea that f and g have ‘essentially the same growth rate’.

Say f is $\Theta(g)$ (or g is an asymptotically tight bound for f) if both $f \in O(g)$ and $f \in \Omega(g)$.

Equivalently, $f \in \Theta(g)$ if and only if

$$\exists c_1, c_2 > 0.\ \exists N.\ \forall n \geq N.\ c_1 g(n) \leq f(n) \leq c_2 g(n)$$



Note also that $f = \Theta(g) \iff g = \Theta(f)$.

Examples of Θ

For each of the following functions f, identify some 'simple' g such that $f = \Theta(g)$.

Example 1: $f(n) = 3n^2 - 2n + 19$. Answer: $f(n) = \Theta(n^2)$.

The dominant term is $3n^2$, the rest is small change.
So $f(n)$ will eventually be sandwiched between $2n^2$ and $4n^2$.
(Specifically, can take e.g. $c_1 = 2$, $c_2 = 4$, $N = 5$.)

Example 2: $f(n) = 5 - 4/n$. Answer: $f(n) = \Theta(1)$.

That is, we're taking our 'g' to be the constant function $g(n) = 1$.
Then for any $n \geq 1$, we have

$$1.g(n) \ = \ 1 \ \leq \ 5 - 4/n \ \leq \ 5 \ = \ 5.g(n)$$

So taking $c_1 = 1$, $c_2 = 5$, $N = 1$ will work.

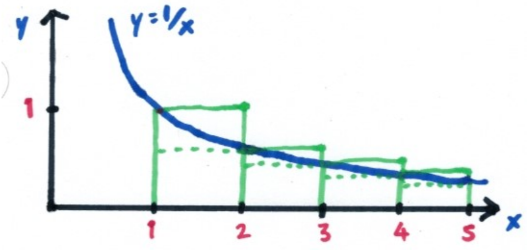
Harder example

Identify some simple g such that $f = \Theta(g)$.

Example 3: $f(n) = \Sigma_{i=1}^{n} 1/i$.
E.g. $f(4) = 1 + 1/2 + 1/3 + 1/4 = 2\frac{1}{12}$.
Answer: $f(n) = \Theta(\ln n)$.

Idea: $f(n)$ is close to $\int_1^n (1/x)dx$, which is $\ln n$.

E.g. for $n = 4$:



Actually, $\Theta(\ln n)$ is same as $\Theta(\lg n)$: see Tutorial Sheet 1.

Growth rates and algorithms

Let's return to an earlier question. Suppose each implementation J of (say) **MergeSort** yields some runtime function T_J.

Question: What do we expect all these T_J to have in common?

Answer: Same growth rate!

$$\forall J, J' \text{ implementing MergeSort. } T_J = \Theta(T_{J'})$$

Will justify this next time, and furthermore see that

$$\forall J \text{ implementing MergeSort. } T_J = \Theta(n \lg n)$$

Idea: Asymptotic notation can crisply express essential properties of algorithms, abstracting away from implementation detail.

Of the Gang of Five, we'll meet O and Θ most often.

Some common growth rates

Certain (types of) growth rates crop up frequently, and have names in common use.

- $\Theta(1)$: (within) constant time
- $\Theta(\lg n)$: logarithmic time
- $\Theta(n)$: linear time
- $\Theta(n \lg n)$: log-linear time
- $\Theta(n^2)$: quadratic time
- $\Theta(n^k)$ for some exponent k: polynomial time
- $\Theta(b^n)$ for some base b: exponential time

Reading (same as for Lecture 3):
Roughgarden Chapter 2
Kleinberg/Tardos Chapter 2, especially 2.2, 2.4
CLRS Chapter 3 (covers whole Gang of Five)
GGT Sections 3.3, 3.4.