

Introduction to Algorithms and Data Structures

Lecture 3: Asymptotics: o and ω

John Longley

School of Informatics
University of Edinburgh

24 September 2024

Outline

Goal of Lectures 3,4,5:

- ▶ Introduce **asymptotic analysis**, the core mathematical theory used in this course. Centres around a certain **'Gang of Five'**:

o O Θ Ω ω

- ▶ Apply this theory to **InsertSort** and **MergeSort**.

Purpose of the theory: Way of making precise, quantitative statements about efficiency properties of *algorithms themselves*. (E.g. What do *all* implementations of MergeSort have in common?)

Note: These ideas may take a while to master – don't worry!

This lecture: In what sense is MergeSort 'fundamentally faster' than InsertSort? o and ω .

Comparing runtimes for InsertSort and MergeSort

Take some specific implementations of **InsertSort** and **MergeSort**. Broadly, we want to consider . . .

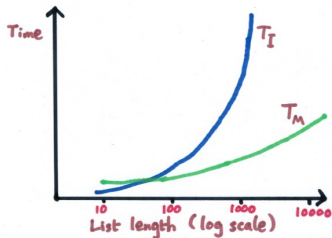
$T_I(n)$ = time taken by **InsertSort** on a list of length n (in ms)

$T_M(n)$ = time taken by **MergeSort** on a list of length n

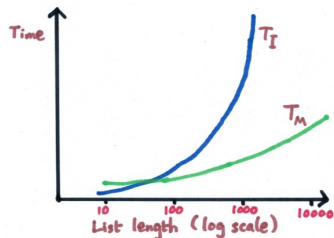
Which list of length n ? Time may vary widely between lists!

Will come back to this. For now, take $T_I(n)$, $T_M(n)$ to be the **worst-case** (i.e. maximum) times for a list of length n .

Could then plot a graph (schematic only):



Comparing T_I and T_M



How can we capture our intuition ' T_I grows much faster than T_M '?

Attempt 1: $\forall n. T_M(n) < T_I(n)$.

Not true! We've seen that for *small* n , **InsertSort** is faster. Really want to say that **MergeSort** is *eventually* faster.

Attempt 2: $\exists N. \forall n \geq N. T_M(n) < T_I(n)$.

True. E.g. $N = 100$ would do here.

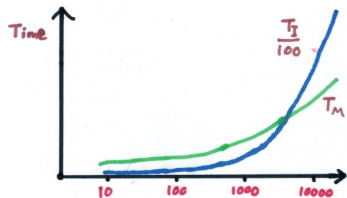
But doesn't capture the essential difference ...

Comparing growth rates

Attempt 3: Idea is that we expect that *any* impl of **MergeSort** will eventually beat *any* impl of **InsertSort**.

E.g. suppose we gave **InsertSort** an **unfair advantage** by running it on a machine 100 times faster.

Even $T_I(n)/100$ would eventually overtake $T_M(n)$:



In symbols: $\exists N. \forall n \geq N. T_M(n) < 0.01 T_I(n)$.

(E.g. $N = 100000$ would do here.)

Question: What if we replaced 0.01 by 0.0001? Or by 0.000001?

Growth rates and 'little o'

Intuition (will justify later): For *any* handicap factor c , however close to zero, $cT_I(n)$ will eventually break out and overtake T_M :

$$\forall c > 0. \exists N. \forall n \geq N. T_M(n) < cT_I(n)$$

We express this by saying T_M is $o(T_I)$. Can read this as: ' T_M is **slower-growing** than' or '**asymptotically smaller** than T_I '.

In general, we say f is $o(g)$ if

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

(Here $f, g : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$, c ranges over \mathbb{R} , and N, n range over \mathbb{N} .)

Equivalent to saying $g(n)/f(n) \rightarrow \infty$ as $n \rightarrow \infty$ (if $f : \mathbb{N} \rightarrow \mathbb{R}_{>0}$).

o -notation: Simple examples

Will come back to **InsertSort** and **MergeSort** later.
Meanwhile, some simpler examples of o .

Example 1: Is it true that n^2 is $o(n^3)$? **YES!**

Informal justification: The ratio n^3/n^2 is n , which (trivially!) tends to ∞ as n tends to ∞ .

Rigorous justification: Want to show that the o formula is satisfied:

$$\forall c > 0. \exists N. \forall n \geq N. n^2 < cn^3$$

Suppose we're given some $c > 0$. Need to pick a suitable N .

Take any $N > 1/c$. Then for all $n \geq N$, we have

$$cn^3 = cn \cdot n^2 \geq cN \cdot n^2 > c(1/c)n^2 = n^2$$

(**Idea:** If $n > 1/c$, the extra factor n will compensate for the c .)

Examples of o -notation, continued

Example 2: Is it true that $100\sqrt{n}$ is $o(n)$? **YES!**

Informal justification: The ratio $n/(100\sqrt{n})$ is $\sqrt{n}/100$, which tends to ∞ as n tends to ∞ .

Rigorous justification: Want to show that the o formula is satisfied:

$$\forall c > 0. \exists N. \forall n \geq N. 100\sqrt{n} < cn$$

Suppose we're given some $c > 0$. Need to pick a suitable N .

Take any $N > 10000/c^2$. Then for all $n \geq N$, we have

$$cn = c\sqrt{n}\sqrt{n} \geq c\sqrt{N}\sqrt{n} > c(100/c)\sqrt{n} = 100\sqrt{n}$$



How did we pick that $10000/c^2$?

E.g. by working backwards from the requirement $n/(100\sqrt{n}) > 1/c$.

Examples of o -notation, continued

Example 3: Is it true that $n + 1000000$ is $o(6n)$? **NO!**

Informal justification: Even though the ratio $6n/(n + 1000000)$ continues to increase as n tends to ∞ , it never exceeds 6, so doesn't tend to ∞ .

Rigorous justification: Want to show the *negation* of the o formula:

$$\neg (\forall c > 0. \exists N. \forall n \geq N. n + 1000000 < c.6n)$$

which is equivalent to

$$\exists c > 0. \neg (\exists N. \forall n \geq N. n + 1000000 < c.6n)$$

$$\exists c > 0. \forall N. \neg (\forall n \geq N. n + 1000000 < c.6n)$$

$$\exists c > 0. \forall N. \exists n \geq N. n + 1000000 \geq c.6n$$



We can take $c = 1/7$. It's then true for *any* $n \geq 0$ that

$$n + 1000000 > n \geq 6n/7 = c.6n$$

So it's clear that $\forall N. \exists n \geq N. n + 1000000 \geq c.6n$ (given N , can just take $n = N$).

What is 'o(g)' officially?

Officially, $o(g)$ is a **set**: namely, the set of all f that 'are $o(g)$ '.

$$o(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0} \mid \forall c > 0. \exists N. \forall n \geq N. f(n) < cg(n)\}$$

So, ' f is $o(g)$ ' technically means $f \in o(g)$.

Common convention: Write ' $o(g)$ ' to mean 'some (unspecified) function in the set $o(g)$ '. E.g.

$$f = o(g), \quad f(n) = 3n^2 + o(n)$$

Needs care: e.g. $n = o(n^2)$ and $2n = o(n^2)$ don't imply $n = 2n$!

But many useful laws are valid, e.g.

$$o(g) + o(g) = o(g)$$

which strictly means 'if $f \in o(g)$ and $f' \in o(g)$, then $f + f' \in o(g)$ '.

(Exercise if you like maths: Prove this from the definition of o .)

Reducing clutter using o

Asymptotic notation is useful when we're only interested in the **broad headlines** of how some function behaves.

E.g. Can read $3n^2 + o(n)$ as '3n² plus **small change**.'

Reduces clutter and simplifies calculations!

Example: How does the following behave for large n ?

$$(3n + 5\sqrt{n} + 17 \lg n)(4n + (\sqrt{n} / \lg n) + 12)$$

(In this course, \lg means logarithm to base 2.)

Rather than expanding this in full, can reason as follows:

$$\begin{aligned}(3n + o(n) + o(n))(4n + o(n) + o(n)) &= (3n + o(n))(4n + o(n)) \\ &= 12n^2 + o(3n^2) + o(4n^2) + o(n^2) \\ &= 12n^2 + o(n^2)\end{aligned}$$

(where every step can be rigorously justified).

Some key points

- ▶ Saying $f = o(g)$ gives just the **main headlines** of how f and g are related: 'In the limit, f is vanishingly small relative to g '. Often, this is all we care about.
- ▶ $f = o(g)$ makes a **robust** statement about f, g .
E.g. unaffected by scaling: $f = o(g) \Leftrightarrow 3f = o(0.2g)$.
- ▶ So can expect that e.g. ' $T_M = o(T_I)$ ' will remain true for *any* implementations of **MergeSort/InsertSort**.
- ▶ Use of o can **reduce clutter** and simplify calculations.
- ▶ But without sacrificing mathematical rigour: ' $f = o(g)$ ' has a precisely defined meaning.

General advice: **Sketch graphs** to understand what's going on!

And finally: ω

ω is dual to o . Recall that $f = o(g)$ means:

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

(‘ f is asymptotically smaller than / grows slower than g ’).

By contrast, read $f = \omega(g)$ as saying:

‘ f is asymptotically larger than / grows faster than g ’).

Formal definition: f is $\omega(g)$ if

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

(‘However much we scale g up by, f will eventually overtake it.’)

For purpose of comparing f and g , scaling g ‘up’ by C has same effect as scaling f ‘down’ by $c = 1/C$. So easy to show:

$$f = \omega(g) \text{ if and only if } g = o(f)$$

(Compare: $x > y$ if and only if $y < x$.)

We’ll tend to use o more than ω .

Next time: O , Ω , Θ .

(Most presentations start with these!)

Reading for Lectures 3 and 4:

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.