Introduction to Algorithms and Data Structures

Tutorial 1

your tutor

School of Informatics
University of Edinburgh

1st-4th October, 2024

Q1: equivalence with $\Theta(.)$

These are the target functions to "match" (with respect to Θ) to :

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

- Note the increasing order.
- Where are the biggest jumps?

Q1: sketch of the target functions

Q1(a)

(a) $g(n) = n(n+1)(2n+1)/6$

Q1(a)

(a) $g(n) = n(n+1)(2n+1)/6$

We want to decide which of the f_i will satisfy $g = \Theta(f_i)$:

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

Q1(a)

(a) $g(n) = n(n+1)(2n+1)/6$

We want to decide which of the f_i will satisfy $g = \Theta(f_i)$:

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

WHY?: We can expand $g(n)$ to $\frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$.

Q1(a)

(a) $g(n) = n(n+1)(2n+1)/6$

We want to decide which of the f_i will satisfy $g = \Theta(f_i)$:

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

WHY?: We can expand $g(n)$ to $\frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$.

All positive coefficients, and therefore the term of highest degree drives the $\Theta(\cdot)$.

(By the way, this $g(n)$ is the closed form for $\Sigma_{k=1}^{n} k^2$)

Q1(b), (c)

(b) $g(n) = n \; div \; 57$ *(integer division, rounding down).*

Q1(b), (c)

(b) $g(n) = n \; div \; 57$ *(integer division, rounding down).*

This is $\Theta(\mathbf{n})$. Indeed, $g(n)$ differs from $n/57$ (exact division) by at most 1.

Q1(b), (c)

(b) $g(n) = n$ *div* 57 *(integer division, rounding down).*
This is $\Theta(\mathbf{n})$. Indeed, $g(n)$ differs from $n/57$ (exact division) by at most 1.

Formal argument: Which constants $c > 0$ for $\Omega(n)$, $C > 0$ for $O(n)$ will give

$$c \cdot n \ \leq n \text{ div } 57 \ \leq C \cdot n$$

for sufficiently large n?

Q1(b), (c)

(b) $g(n) = n \; div \; 57$ *(integer division, rounding down).*
This is $\Theta(\mathbf{n})$. Indeed, $g(n)$ differs from $n/57$ (exact division) by at most 1.

Formal argument: Which constants $c > 0$ for $\Omega(n)$, $C > 0$ for $O(n)$ will give

$$c \cdot n \; \leq n \operatorname{div} 57 \; \leq C \cdot n$$

for sufficiently large n?

(c) $g(n) = n \; mod \; 57 + 1$.

Q1(b), (c)

(b) $g(n) = n \; div \; 57$ *(integer division, rounding down).*

This is $\Theta(\mathbf{n})$. Indeed, $g(n)$ differs from $n/57$ (exact division) by at most 1.

Formal argument: Which constants $c > 0$ for $\Omega(n)$, $C > 0$ for $O(n)$ will give

$$c \cdot n \; \leq n \operatorname{div} 57 \; \leq C \cdot n$$

for sufficiently large n?

(c) $g(n) = n \; mod \; 57 + 1$.

Values that $g(n)$ could ever possibly take are $1, 2, \ldots, 57$.

Q1(b), (c)

(b) $g(n) = n \; div \; 57$ *(integer division, rounding down).*
This is $\Theta(\mathbf{n})$. Indeed, $g(n)$ differs from $n/57$ (exact division) by at most 1.

Formal argument: Which constants $c > 0$ for $\Omega(n)$, $C > 0$ for $O(n)$ will give

$$c \cdot n \; \leq n \operatorname{div} 57 \; \leq C \cdot n$$

for sufficiently large n?

(c) $g(n) = n \; mod \; 57 + 1$.

Values that $g(n)$ could ever possibly take are $1, 2, \ldots, 57$.
$g(n)$ is $\Theta(\mathbf{1})$, because we have $1 \leq g(n) \leq 57$ for all n.

Q1(b), (c)

(b) $g(n) = n \; div \; 57$ *(integer division, rounding down).*

This is $\Theta(\mathbf{n})$. Indeed, $g(n)$ differs from $n/57$ (exact division) by at most 1.

Formal argument: Which constants $c > 0$ for $\Omega(n)$, $C > 0$ for $O(n)$ will give

$$c \cdot n \; \leq n \operatorname{div} 57 \; \leq C \cdot n$$

for sufficiently large n?

(c) $g(n) = n \; mod \; 57 + 1$.

Values that $g(n)$ could ever possibly take are $1, 2, \ldots, 57$.

$g(n)$ is $\Theta(\mathbf{1})$, because we have $1 \leq g(n) \leq 57$ for all n.

(Without the '+1', it would be $O(1)$ but not $\Omega(1)$ (i.e. not eventually bounded below by a positive constant), because n mod 57 would be zero infinitely often.)

Q1(d)

$g(n) = n \lg n + (\lg n)^3 + e^{-n}$.

Q1(d)

$g(n) = n \lg n + (\lg n)^3 + e^{-n}$.

Allowed to assume that $\lg n = o(\sqrt{n})$.

Important - this gives $(\lg n)^3 = (\lg n) \cdot o(n)$, so $(\lg n)^3$ is $o(n \lg n)$.

Q1(d)

$g(n) = n \lg n + (\lg n)^3 + e^{-n}$.

Allowed to assume that $\lg n = o(\sqrt{n})$.

Important - this gives $(\lg n)^3 = (\lg n) \cdot o(n)$, so $(\lg n)^3$ is $o(n \lg n)$.

Also e^{-n} is less than 1 always.

So $n \lg(n)$ is the dominating component of $g(n)$, and $g(n \lg n) = O(\mathbf{n \lg n})$.

Q1(d)

$g(n) = n \lg n + (\lg n)^3 + e^{-n}$.

Allowed to assume that $\lg n = o(\sqrt{n})$.

Important - this gives $(\lg n)^3 = (\lg n) \cdot o(n)$, so $(\lg n)^3$ is $o(n \lg n)$.

Also e^{-n} is less than 1 always.

So $n \lg(n)$ is the dominating component of $g(n)$, and $g(n \lg n) = O(\mathbf{n \lg n})$.

Constants for the formal argument?

- $c = 1$, $N = 1$ for the $\Omega(n \lg n)$
- $C = 3$, $N = 2$ for the $O(n \lg n)$ (but the "truth" is $C > 1$)

Q1(e) - tricky one!

Where would the $n!$ function fit ? Does $n!$ have the same growth rate as one of the above functions f_i? Or does it fall between f_i and f_{i+1} for some i?

Q1(e) - tricky one!

Where would the $n!$ function fit ? Does $n!$ have the same growth rate as one of the above functions f_i? Or does it fall between f_i and f_{i+1} for some i?

Answer: The growth rate of $n!$ falls strictly between that of 2^n and 2^{2^n}.

Q1(e) - tricky one!

Where would the $n!$ function fit ? Does $n!$ have the same growth rate as one of the above functions f_i? Or does it fall between f_i and f_{i+1} for some i?

Answer: The growth rate of $n!$ falls strictly between that of 2^n and 2^{2^n}.

To see that $2^n = o(n!)$, let's look at the ration $n!/2^n$, which is

$$\frac{1 \times 2 \times 3 \times \cdots \times n}{2 \times 2 \times 2 \times \cdots \times 2}$$

This is at least $n/2$ (once $n \geq 2$), which tends to infinity as n does.

Q1(e) - tricky one!

Where would the $n!$ function fit ? Does $n!$ have the same growth rate as one of the above functions f_i? Or does it fall between f_i and f_{i+1} for some i?

Answer: The growth rate of $n!$ falls strictly between that of 2^n and 2^{2^n}.

To see that $2^n = o(n!)$, let's look at the ration $n!/2^n$, which is

$$\frac{1 \times 2 \times 3 \times \cdots \times n}{2 \times 2 \times 2 \times \cdots \times 2}$$

This is at least $n/2$ (once $n \geq 2$), which tends to infinity as n does.

To see that $n! = o(2^{2^n})$, we can note that for $n \geq 2$,

$$n! \;<\; n^n \;<\; (2^n)^n \;=\; 2^{n^2} \;\leq\; 2^{2^n}$$

Q2 rigorous proofs

Let's remind ourselves of the assumptions we are working under. These really matter!

- Polynomial functions grow more slowly than exponential ones: for any $k > 0$ and any $r > 1$, we have $n^k = o(r^n)$.

  (yes, these are allowed to be different k and r. The constant which is the "base" needs to satisfy the > 1)

- Logs grow more slowly than square roots, cube roots etc.: for any $k \geq 1$ we have $\lg n = o(n^{1/k})$.

Q2 (a) - rigorous proofs

Show directly from the definition that $100n^3 = o(n^4)$.

Q2 (a) - rigorous proofs

Show directly from the definition that $100n^3 = o(n^4)$.

Need to show that for *every* $c > 0$, we can find $N \in \mathbb{N}$ such that

$$100n^3 \leq c \cdot n^4$$

for all $n \geq N$.

Q2 (a) - rigorous proofs

Show directly from the definition that $100n^3 = o(n^4)$.

Need to show that for *every* $c > 0$, we can find $N \in \mathbb{N}$ such that

$$100n^3 \leq c \cdot n^4$$

for all $n \geq N$.

Looks promising!, given the extra n on rhs.
The 100 constant is just a little technical distraction.

Q2 (a) - rigorous proofs

Show directly from the definition that $100n^3 = o(n^4)$.

Need to show that for *every* $c > 0$, we can find $N \in \mathbb{N}$ such that

$$100n^3 \leq c \cdot n^4$$

for all $n \geq N$.

Looks promising!, given the extra n on rhs.
The 100 constant is just a little technical distraction.

Working backwards ... how do we ensure $100n^3 < cn^4$?

Q2 (a) - rigorous proofs

Show directly from the definition that $100n^3 = o(n^4)$.

Need to show that for *every* $c > 0$, we can find $N \in \mathbb{N}$ such that

$$100n^3 \leq c \cdot n^4$$

for all $n \geq N$.

Looks promising!, given the extra n on rhs.
The 100 constant is just a little technical distraction.

Working backwards ... how do we ensure $100n^3 < cn^4$?
This is equivalent to $100 < cn$ (for positive n), equivalent to $n > 100/c$.

Q2 (a) - rigorous proofs

Show directly from the definition that $100n^3 = o(n^4)$.

Need to show that for *every* $c > 0$, we can find $N \in \mathbb{N}$ such that

$$100n^3 \ \leq c \cdot n^4$$

for all $n \geq N$.

Looks promising!, given the extra n on rhs.
The 100 constant is just a little technical distraction.

Working backwards ... how do we ensure $100n^3 < cn^4$?
This is equivalent to $100 < cn$ (for positive n), equivalent to $n > 100/c$.

Now we reverse this to give our polished argument:
Given $c > 0$, consider any $N > 100/c$. Then for any $n \geq N$ we have

$$100n^3 \ = \ c(100/c)n^3 \ < \ c.n.n^3 \ = \ cn^4.$$

(The working-backwards is the stuff we really need to learn)

Q2(b)

If r, s are any real numbers with $0 \leq r < s$, then $n^r = o(n^s)$.

Q2(b)

If r, s are any real numbers with $0 \leq r < s$, then $n^r = o(n^s)$.

Notice the ratio n^s/n^r is n^{s-r}, and $s - r > 0$.

Q2(b)

If r, s are any real numbers with $0 \leq r < s$, then $n^r = o(n^s)$.

Notice the ratio n^s/n^r is n^{s-r}, and $s - r > 0$.

Can make n^{s-r} bigger than any hypothetical $C > 0$ by requiring $n > C^{1/(s-r)}$.

Q2(b)

If r, s are any real numbers with $0 \leq r < s$, then $n^r = o(n^s)$.

Notice the ratio n^s/n^r is n^{s-r}, and $s - r > 0$.

Can make n^{s-r} bigger than any hypothetical $C > 0$ by requiring $n > C^{1/(s-r)}$.

This shows that $n^s = \omega(n^r)$, which is equivalent to $n^r = o(n^s)$.

Q2(b)

If r, s are any real numbers with $0 \leq r < s$, then $n^r = o(n^s)$.

Notice the ratio n^s/n^r is n^{s-r}, and $s - r > 0$.

Can make n^{s-r} bigger than any hypothetical $C > 0$ by requiring $n > C^{1/(s-r)}$.

This shows that $n^s = \omega(n^r)$, which is equivalent to $n^r = o(n^s)$.

Tip: Looking at how the ratio of the two functions behaves is often a good start.

Q2(c)

Writing '$\lg$' *for log to base 2 and* '$\ln$' *for log to base e, show that* $\ln n = O(\lg n)$. *Deduce that* $\lg n = \Theta(\ln n)$.

Q2(c)

Writing $\lg$*' for log to base 2 and* $\ln$*' for log to base* e*, show that* $\ln n = O(\lg n)$*. Deduce that* $\lg n = \Theta(\ln n)$*.*

There is a well-known formula

$$\log_b x \;=\; (\log_b a)(\log_a x)$$

Can use this to write $\lg x$ exactly as $(\lg e)(\ln x)$.

Q2(c)

Writing '$\lg$' for log to base 2 and '$\ln$' for log to base e, show that $\ln n = O(\lg n)$. *Deduce that* $\lg n = \Theta(\ln n)$.

There is a well-known formula

$$\log_b x \;=\; (\log_b a)(\log_a x)$$

Can use this to write $\lg x$ exactly as $(\lg e)(\ln x)$.

Here $\lg e$ is an absolute constant, so we immediately get $\lg n = \Theta(\ln n)$.

Tip: "rules of logs" are really important for IADS

Q2(d)

Is it likewise true that $2^n = \Theta(e^n)$*?*

Q2(d)

Is it likewise true that $2^n = \Theta(e^n)$*?* No!

The ratio $e^n/2^n$ is $(e/2)^n$, which will surpass any given $C > 0$ as n increases (specifically, once $n > \ln C / \ln(e/2)$).

Q3 - long-division

Recall the methods you learned at school for addition, long multiplication and long division. For each of these, informally analyse the asymptotic worst-case runtime on inputs of at most n decimal digits.
You may take 'time' to mean the number of times you have to write a symbol on the page.

Q3 - long-division

Recall the methods you learned at school for addition, long multiplication and long division. For each of these, informally analyse the asymptotic worst-case runtime on inputs of at most n decimal digits.
You may take 'time' to mean the number of times you have to write a symbol on the page.

This is a "discuss" (informal) question - however, it was inspired by UK primary school teaching!

It's interesting to discuss the different formatting of Long Division in different countries, as discussed in this reddit post (for example)

Q3 - some examples

Q3

- For numbers of at most n digits, addition takes 'time' $\Theta(n)$.

  We have to write the (at most $n+1$) digits of the answer, plus (at worst) a similar number of carry digits.

Q3

- For numbers of at most n digits, addition takes 'time' $\Theta(n)$.

  We have to write the (at most $n+1$) digits of the answer, plus (at worst) a similar number of carry digits.

- Long multiplication of two n-digit numbers is a bit like writing n numbers (each of at most $2n+1$ digits), then adding all these.

  Not hard to reason all of this takes time $\Theta(n^2)$.

Q3

- For numbers of at most n digits, addition takes 'time' $\Theta(n)$.

  We have to write the (at most $n+1$) digits of the answer, plus (at worst) a similar number of carry digits.

- Long multiplication of two n-digit numbers is a bit like writing n numbers (each of at most $2n+1$ digits), then adding all these.

  Not hard to reason all of this takes time $\Theta(n^2)$.

- For integer long division (e.g. resulting in a div b and a mod b).

  The division will proceed in $\leq n$ 'rounds', in each of which we perform a subtraction of size $\leq n+1$.

  (The necessary values of $b, 2b, \ldots, 9b$ can be precomputed at the start, taking just time $\Theta(n)$.)

  So the overall runtime is clearly $O(n^2)$.

Q3

- For numbers of at most n digits, addition takes 'time' $\Theta(n)$.

  We have to write the (at most $n+1$) digits of the answer, plus (at worst) a similar number of carry digits.

- Long multiplication of two n-digit numbers is a bit like writing n numbers (each of at most $2n+1$ digits), then adding all these.

  Not hard to reason all of this takes time $\Theta(n^2)$.

- For integer long division (e.g. resulting in a div b and a mod b).

  The division will proceed in $\leq n$ 'rounds', in each of which we perform a subtraction of size $\leq n+1$.

  (The necessary values of $b, 2b, \ldots, 9b$ can be precomputed at the start, taking just time $\Theta(n)$.)

  So the overall runtime is clearly $O(n^2)$.

  To see that the worst-case runtime is also $\Omega(n^2)$, consider the situation of dividing an n-digit a by an n/2-digit b. Clearly this can require around n/2 subtractions of size n/2.