

Asymptotic Notation, Recurrences

Mary Cryan

27th September, 2011

Asymptotic growth rates

Let $g : \mathbb{N} \rightarrow \mathbb{R}$.

O -notation: $O(g)$ is the set of all functions $f : \mathbb{N} \rightarrow \mathbb{R}$ for which there are constants $c > 0$ and $n_0 \geq 0$ such that

$$0 \leq f(n) \leq c \cdot g(n), \quad \text{for all } n \geq n_0.$$

“Rate of change of $f(n)$ is at most that of $g(n)$ ”

Ω -notation: $\Omega(g)$ is the set of all functions $f : \mathbb{N} \rightarrow \mathbb{R}$ for which there are constants $c > 0$ and $n_0 \geq 0$ such that

$$0 \leq c \cdot g(n) \leq f(n), \quad \text{for all } n \geq n_0.$$

“Rate of change of $f(n)$ is at least that of $g(n)$ ”

Θ -notation: $\Theta(g)$ is the set of all functions $f : \mathbb{N} \rightarrow \mathbb{R}$ for which there are constants $c_1, c_2 > 0$ and $n_0 \geq 0$ such that

$$0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n), \quad \text{for all } n \geq n_0.$$

“Rate of change of $f(n)$ and $g(n)$ are about the same”

Examples

- ▶ Let $f(n) = 0.01 \cdot n^2$ and $g(n) = n$. Then $g = O(f)$.
- ▶ Let $f(n) = \ln(n)$ and $g(n) = n$. Then $g = \Omega(f)$.
- ▶ Let $f(n) = 10n + \ln(n)$ and $g(n) = n$. Then $g = \Theta(n)$.

Sometimes $O(\dots)$ appears within a formula, rather than simply forming the right hand side of an equation. We make sense of this by thinking of $O(\dots)$ as standing for some anonymous (but fixed) function from the set of the same name.

For example, $h(n) = 2^{O(n)}$ means $\exists c > 0, n_0 \in \mathbb{N}$ such that

$$h(n) \leq 2^{cn} \text{ for all } n > n_0.$$

Consequences

Suppose $f(n) = O(g(n))$ AND $g(n) = O(f(n))$. What can we say?

What if $f(n) = O(g(n))$ AND $f(n) = \Omega(g(n))$?

Various consequences of the above conventions:

$$\Theta(n) \times \Theta(n^2) = \Theta(n^3),$$

$$\Theta(n) + \Theta(n^2) = \Theta(n^2),$$

$$\Theta(n) + \Theta(n) = \Theta(n).$$

Reminder of INSERTIONSORT

Algorithm INSERTION-SORT(A)

1. **for** $j \leftarrow 2$ **to** $length[A]$
2. **do** $key \leftarrow A[j]$
3. Insert $A[j]$ into the sorted sequence $A[1 \dots j - 1]$
4. $i \leftarrow j - 1$
5. **while** $i > 0$ and $A[i] > key$
6. **do** $A[i + 1] \leftarrow A[i]$
7. $i \leftarrow i - 1$
8. $A[i + 1] \leftarrow key$

running-time of INSERTIONSORT

- ▶ The for-loop on line 1 is iterated $n - 1$ times
- ▶ For each execution of the for, the while does $\leq j$ iterations;
- ▶ Each of the comparisons/assignments requires only $O(1)$ basic steps;
- ▶ Therefore the total number of steps (=time) is at most

$$O(1) \sum_{j=1}^n j = O(1) \frac{n(n+1)}{2} = O(n^2).$$

- ▶ This is essentially tight - sorting the list $n, n - 1, n - 1, \dots, 3, 2, 1$ takes $\Omega(n^2)$ time.

reminder of MERGESORT

Input: A list A of natural numbers, $p, q : 1 \leq p \leq r \leq n$.

Output: A sorted (increasing order) permutation of $A[p \dots q]$.

Algorithm MERGE-SORT(A, p, r)

1. **if** $p < r$ **then**
2. $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
3. MERGE-SORT(A, p, q)
4. MERGE-SORT($A, q + 1, r$)
5. MERGE(A, p, q, r)

reminder of MERGE

Input: A list A of natural numbers, three indices $p, q, r : 1 \leq p \leq r \leq n$, where $A[p \dots q]$ and $A[q + 1 \dots r]$ are both individually sorted.

Output: A sorted (increasing order) permutation of $A[p \dots q]$.

- (i) Define an output array B of length $r - p + 1$ ($r - p + 1$ steps).
- (ii) Perform the merge, consisting of
 - ▶ between $\min\{q - p + 1, r - q\}$ and $r - p + 1$ comparisons, and
 - ▶ exactly $r - p + 1$ copy operations (copying the lesser value to the next slot of B), and
 - ▶ $2(r - p + 1)$ updates of the indices for running through $A[p \dots q]$, $A[q + 1 \dots r]$ and $B[p \dots r]$.

The running-time of MERGE is linear in the length of the array, specifically:

$$4(r - p + 1) + \min\{q - p + 1, r - q\} \leq T_{\text{MERGE}}(p, q, r) \leq 5(r - p + 1)$$

Taking $n = r - p + 1$, we have

$$4n \leq T_{\text{MERGE}}(n) \leq 5n$$

Running-time of MERGESORT

Put

$$n = r - p + 1.$$

Running time $T_{\text{MS}}(n)$ satisfies:

$$T_{\text{MS}}(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ T_{\text{MS}}(\lceil n/2 \rceil) + T_{\text{MS}}(\lfloor n/2 \rfloor) + \Theta(n) & \text{if } n > 1. \end{cases}$$

The $\Theta(n)$ is from analysis of MERGE on the previous slide. Note that MERGESORT only gives $\lfloor \frac{n+1}{2} \rfloor$ and $\lfloor \frac{n-1}{2} \rfloor$ as the subarray sizes (BOARD)

Solving recurrences

Methods for deriving/verifying solutions to recurrences:

Induction Guess the solution and verify by induction on n .

Unfold and sum “Unfold” the recurrence by iterated substitution on the “neat” values of n (often power of 2 case). At some point a pattern emerges. The “solution” is obtained by evaluating a sum that arises from the pattern.

Since the pattern is just a *guess* for special n , the method is rigorous only if we verify the solution (e.g., by a direct induction proof, or by the Master Theorem).

“Master Theorem” Match the recurrence against a template. Read off the solution from the Master Theorem.

Upper bound for MERGESORT

First principles: No Master Theorem.

When working from first principles, need to replace “extra work” terms ($\Theta(n)$ for MERGESORT) by terms with explicit constants.

So check slide 8.

$$T_{\text{MS}}(n) \leq \begin{cases} 1 & \text{if } n = 1, \\ T_{\text{MS}}(\lceil n/2 \rceil) + T_{\text{MS}}(\lfloor n/2 \rfloor) + 5n & \text{if } n > 1. \end{cases}$$

Unfold-and-sum(BOARD) gives a “guess” for the upper bound:

$$T_{\text{MS}}(n) \leq 5n \lg(n) + n.$$

Proof of upper bound for MERGESORT

$$T_{\text{MS}}(n) \leq \begin{cases} 1 & \text{if } n = 1, \\ T_{\text{MS}}(\lceil n/2 \rceil) + T_{\text{MS}}(\lfloor n/2 \rfloor) + 5n & \text{if } n > 1. \end{cases}$$

(Finally) corrected the claim and proof on 12th Dec.

Not true that $T_{\text{MS}}(n) \leq 5n \lg(n) + n$ for all \mathbb{N} (as claimed).

new claim: $T_{\text{MS}}(n) \leq 5n \lg(n) + n$ if $n = 2^k$, for some $k \in \mathbb{N}$

Proof Base case: holds direct from the recurrence.

Induction Hypothesis (IH): Upper bound holds *for* $n = 2^{k-1}$.

Induction Step: Now consider $n = 2^k$ and apply the recurrence:

$$\begin{aligned} T_{\text{MS}}(n) &\leq T_{\text{MS}}(\lceil 2^{k-1} \rceil) + T_{\text{MS}}(\lfloor 2^{k-1} \rfloor) + 5n \\ &= 2 \cdot T_{\text{MS}}(2^{k-1}) + 5n \\ &\leq 2 \cdot 2^{k-1} (5 \lg(2^{k-1}) + 1) + 5n && \text{((IH))} \\ &= n \cdot 5 \lg(n/2) + n + 5n \\ &= 5n(\lg(n/2) + 1) + n = 5n \lg(n) + n && \text{(by lg rules),} \end{aligned}$$

AS REQUIRED.

Upper bound for MERGESORT (added 12 Dec)

So now we have proven:

$$T_{\text{MS}}(n) \leq 5n \lg(n) + n \text{ for all POWERS OF 2 } n.$$

Note: This bound is *not* true for general n (try $n = 2^k - 1$, eg $n = 7$ to see that the recurrence does not lead to this upper bound in general).

Non power-of-2 bound: First show that the upperbound of $T_{\text{MS}}(n)$ is *monotone* - this can be done *inductively*.

Second, use the monotonicity to show that $T_{\text{MS}}(n) \leq T_{\text{MS}}(\hat{n})$, for $\hat{n} = 2^{\lceil \lg(n) \rceil}$. By power-of-2 results $T(\hat{n}) = 5\hat{n} \lg(\hat{n}) + \hat{n}$. By definition of \hat{n} , $\hat{n} < 2n$, so we get

$$T_{\text{MS}}(n) \leq T_{\text{MS}}(\hat{n}) = 5\hat{n}(\lg(\hat{n}) + 1) < 5(2n) \lg(2n) + 2n = 10n \lg(n) + 12n.$$

So the bound for all $n \in \mathbb{N}$ is $T_{\text{MS}}(n) \leq 10n \lg(n) + 12n$.

The “truth” is quite a bit better, but not well-suited to an inductive proof. So my belief that this was an “easy” recurrence was wrong.

An Example recurrence

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 4T(\lfloor n/2 \rfloor) + n^2 & \text{if } n > 1. \end{cases}$$

Unfold and Sum (“guessing” the expression):

Assume that $n = 2^\ell$, that is, $\ell = \lg(n) = \log_2(n)$, for an $\ell \in \mathbb{N}$.

$$\begin{aligned} T(n) &= 4T(n/2) + n^2 &= 4^k T(n/2^k) + kn^2 \\ &= 4(4T(n/4) + n^2/4) + n^2 && \vdots \\ &= 4^2 T(n/2^2) + 2n^2 &= 4^\ell T(n/2^\ell) + \ell n^2 \\ &= 4^3 T(n/2^3) + 3n^2 &= (2^\ell)^2 T(1) + \lg(n)n^2 \\ &\vdots &= n^2 + \lg(n)n^2 \end{aligned}$$

Example (cont.)

Proposition

$$T(n) \in \Theta(n^2 \cdot \lg(n))$$

Proof: By induction on n (on the blackboard).
PROOF BY FIRST PRINCIPLES (difficult example).

Proof of Prop

UPPER BOUND: We prove that

$$T(n) \leq (\lg(n) + 1)n^2 \quad (*)$$

for all $n \in \mathbb{N}$ by induction.

$n = 1$: $T(1) = 1 = (\lg(1) + 1)1^2$.

$n > 1$: Induction hypothesis (IH): Assume (*) holds for $k < n$ (includes $k = \lfloor n/2 \rfloor$). Then

$$\begin{aligned} T(n) &= 4T(\lfloor n/2 \rfloor) + n^2 \\ &\leq 4(\lg(\lfloor n/2 \rfloor) + 1)(\lfloor n/2 \rfloor)^2 + n^2 && \text{by (IH)} \\ &\leq 4(\lg(n/2) + 1)(n/2)^2 + n^2 && \text{by } \lfloor n/2 \rfloor \leq n/2 \\ &= 4(\lg(n) - \lg(2) + 1) \left(\frac{n^2}{4} \right) + n^2 && \text{by } \lg\left(\frac{a}{b}\right) = \lg a - \lg b \\ &= \lg(n)n^2 + n^2 = (\lg(n) + 1)n^2. \end{aligned}$$

LOWER BOUND: *This is difficult* (not usually so tough)

We prove (by first principles) that

$$T(n) \geq \frac{1}{4} \lg(n)n^2 \quad (**)$$

for all $n \in \mathbb{N}$ by induction on n .

$n = 1$: $T(1) = 1 \geq 0 = \frac{1}{4} \lg(1)1^2$

$n \geq 2$: Induction hypothesis (IH-2): Assume (**) holds for all $k < n$ - in particular, for $k = \lfloor n/2 \rfloor$. Then

$$\begin{aligned} T(n) &= 4T(\lfloor n/2 \rfloor) + n^2 \\ &\geq \lg(\lfloor n/2 \rfloor)(\lfloor n/2 \rfloor)^2 + n^2 \quad (\text{by (IH-2)}) \\ &\geq \lg((n-1)/2)((n-1)/2)^2 + n^2 \\ &= (1/4)(\lg(n-1) - 1)(n^2 - 2n + 1) + n^2 \\ &= \frac{1}{4}n^2 \lg(n-1) - \frac{1}{2}n \lg(n-1) + \frac{1}{4}\lg(n-1) - \frac{1}{4}n^2 \\ &\quad + \frac{1}{2}n - \frac{1}{4} + n^2 \end{aligned}$$

$$\begin{aligned}
T(n) &\geq \frac{1}{4}n^2 \lg(n-1) + \frac{3}{4}n^2 - \frac{1}{2}n \lg(n-1) + \frac{1}{4}\lg(n-1) \\
&\quad + \frac{1}{2}n - \frac{1}{4} \\
&= \frac{1}{4}n^2 \lg(n-1) + \frac{1}{4}n^2 + \frac{1}{2}n^2 - \frac{1}{4}(2n-1)(\lg(n-1) - 1) \\
&\geq \frac{1}{4}n^2 \lg(n)
\end{aligned}$$

To see that the last inequality is correct, we have to prove that for all $n \geq 2$ we have:

$$(a) \quad \frac{1}{4}n^2 \lg(n-1) + \frac{1}{4}n^2 = \frac{1}{4}n^2 \lg(2(n-1)) \geq \frac{1}{4}n^2 \lg(n),$$

$$(b) \quad \frac{1}{2}n^2 = \frac{1}{4}2n^2 \geq \frac{1}{4}(2n-1)n \geq \frac{1}{4}(2n-1)(\lg(n-1) - 1).$$

Last step of (a) relies on the fact that if $n \geq 2$, then $2(n-1) \geq n$.

Last step of (b) relies on $n \geq \frac{n-1}{2} \geq \lg(\frac{n-1}{2}) = \lg(n-1) - 1$, true for $n \geq 1$.

$O(\cdot)$ and $\Omega(\cdot)$

LOWER and UPPER bounds together imply that $T(n) = \Theta(n^2 \lg(n))$.

We have shown

$$T(n) \leq (\lg(n) + 1)n^2 \quad (*)$$

for all $n \in \mathbb{N}$ by induction. Taking $c = 2$, this implies that

$$T(n) = O(n^2 \lg(n)).$$

Also, we have shown that

$$T(n) \geq \frac{1}{4} \lg(n)n^2 \quad (**)$$

for all $n \in \mathbb{N}$ by induction on n . Taking $c = 1/4$ in the definition of Ω , this implies that

$$T(n) = \Omega(n^2 \lg(n)).$$

Better Way

Our analysis for the 2nd recurrence was horrible, because we had to work with floors/ceilings. For floors/ceilings, it is better to

- ▶ First prove $T(n) \geq T(n-1)$ for all n IF TRUE (induction).
- ▶ Then work with $T(2^k)$ (LOWER) and $T(2^{k+1})$ (UPPER), for $2^k \leq n \leq 2^{k+1}$.

Reading Assignment

Inf2B ADS Lecture Notes 2 and 8.

[CLRS] Chapter 2.3 and most of Chapter 3 (pp. 28–54, omitting the bits on the little- o and little- ω notation on pp.47-48).

same material is in Chapter 2, pp. 23-35 (omitting 29-30) of [CLR]
(all this material should be familiar from Inf2B and your math classes)