

# Algorithm Design & Analysis Basics

## A Recitation of CLRS Chapter 1-3

Wei Peng

IUPUI

04 September 2013

let us start by reviewing some of the concepts and topics

# data structure and algorithm

## ► data structure

- organize & abstract relationship between elements
- **bridge** the gap between **ideals** and **reality**
  - mathematical **ideals**: sets, mappings, graphs
  - computational **reality**: memory cells (“variables”) and addressing modes (“pointers”)
  - lists, arrays, stacks, queues, heaps, hashes, trees, graphs
- different tradeoffs

## ► algorithm

- specify **procedure** for solving **concrete** problems in **finite** steps
- procedure vs. process
  - much like in magic
  - procedure: the incantation—what is being chanted
  - process: the magic—what is going to happen
- **algorithm** describes procedure
- **execution of algorithm** produces process

## key topics

- ▶ correctness proof
  - ▶ formulate target as  $P(n)$  property; establish basic cases (e.g., “ $P(1)$  true”); show “ $P(k)$  true”  $\Rightarrow$  “ $P(k+1)$  true;” show it terminate eventually.
  - ▶ based on **mathematical induction**
- ▶ asymptotic complexity, a.k.a, “big O” & company
  - ▶ asymptotic?  $\Rightarrow$  when **input size** approaches  $\infty$
  - ▶ why? establish efficiency with relation to some **simple** growth rate like  $1, \lg n, n, n \lg n, n^k$ , and  $2^n$
  - ▶ the point is **simplification**
  - ▶ nemonics:  $o \Leftrightarrow <, O \Leftrightarrow \leq, \Theta \Leftrightarrow =, \Omega \Leftrightarrow \geq, \omega \Leftrightarrow >$
- ▶ divide-and-conquer
  - ▶ divide: divide big problem into (similar but) small ones
  - ▶ conquer: combine small solutions into a big one
  - ▶ complexity is determined by: **how many sub-cases** and **efficiency of “conquer”**
  - ▶ cf Section 4.5 “The master method for solving recurrences”

keep the following tips in mind while solving problems

## problem solving strategies

- ▶ get **elements** right
  - ▶ **vocabulary**: identifying **basic ingredients** to work on
    - ▶ numbers, characters, arrays, pointers, unions, structures
  - ▶ **combination**: making **complex things** out of basic ingredients
    - ▶ arithmetics, conditionals, iterations
  - ▶ **abstraction: naming and conquering** complex things
    - ▶ variables, functions, pointers
- ▶ **means-ends** analysis
  - ▶ ends: where are you heading?
  - ▶ means: what do you have?
  - ▶ use means to reach ends
- ▶ iterative problem solving
  - ▶ solving problems in multiple **top-down** iterations
  - ▶ get the approach right **before** messing with the details
  - ▶ “wishful thinking”
    - ▶ **if only** i can “merge” many sorted sequences into one...
  - ▶ pseudocode frees us from low-level details (for now)
  - ▶ **fail to do so is why (some of) you failed the quizzes**

the rest of this recitation session...

...is on understanding and solving two types of problems:

- ▶ asymptotic complexity: why and how
- ▶ mergesort: divide-and-conquer, recursion, merge, pseudocode

we TAs (Yuan Cao and Wei Peng) divide (and conquer) the job as follows:

- ▶ asymptotic complexity: Wei
- ▶ mergesort: Yuan

## asymptotic complexity definition

for **nonnegative** functions  $f(n)$  and  $g(n)$ ,  $f(n) = \Theta(g(n))$  is defined as:

### Definition

$\exists$  **positive** constants  $c_1, c_2$  and  $N_0$ , so that for  $\forall n \geq N_0$ :

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n).$$

- ▶  $f(n) = O(g(n))$  and  $f(n) = \Omega(g(n))$  are defined as above by omitting the  $c_1$  and  $c_2$  cases, respectively.
- ▶  $f(n) = o(g(n))$  and  $f(n) = \omega(g(n))$  are defined as  $f(n) = O(g(n))$  and  $f(n) = \Omega(g(n))$  by omitting the “=” case.



## about asymptotic complexity

### ► why

- besides *correctness*, **efficiency** also matters
- example: cracking encryption keys
  - theory: brute-force can find any key eventually
  - practice: 2048-bit RSA keys are sufficiently safe<sup>1</sup>
- asymptotic complexity captures **dominating** features of efficiency...
- ...by relating to simple “complexity classes”

### ► tricks

- “asymptotic” reads as “when  $n$  (input size) approaches  $\infty$ ”
- nemonics:  $o \Leftrightarrow <$ ,  $O \Leftrightarrow \leq$ ,  $\Theta \Leftrightarrow =$ ,  $\Omega \Leftrightarrow \geq$ ,  $\omega \Leftrightarrow >$
- simple classes (in *increasing* asymptotic complexity):  $1$ ,  $\lg n$ ,  $n^k (k > 0)$ ,  $a^n (a > 1)$

### ► intuition

- $\Theta(g(n)) \Rightarrow$  the class/collection of functions (of  $n$ ) that run **as fast as** (but **not faster**) than  $g(n)$  **when  $n$  is large enough**
- $f(n) = \Theta(g(n)) \Rightarrow f(n)$  is one such aforementioned function

---

<sup>1</sup>...until 2030, according to <http://goo.gl/cc6UeQ>

## Exercise 0

### Exercise

*Let us ease into the discussion by revisiting a familiar problem.  
Show that  $\max(f(n), g(n)) = \Theta(f(n) + g(n))$ .*

► Solution

## Exercise 1

### Exercise

Shows that

$$0 < \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c < \infty$$

*implies  $f(n) = \Theta(g(n))$  (and hence trivially  $f(n) = O(g(n))$   
 $f(n) = \Omega(g(n))$  by the “=” case in the definition).*

Show these to yourself.

- ▶  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$  implies  $f(n) = o(g(n))$ .
- ▶  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$  implies  $f(n) = \omega(g(n))$ .

▶ Solution

## Exercise 2

### Exercise

Let

- ▶  $\Theta(f(n)) < \Theta(g(n))$  mean  $f(n) = O(g(n))$  and (equivalently)  $g(n) = \Omega(f(n))$
- ▶  $\Theta(f(n)) = \Theta(g(n))$  mean  $f(n) = \Theta(g(n))$  and (equivalently)  $g(n) = \Theta(f(n))$

Ordering the following asymptotic complexity  $\Theta(f(n))$  classes by the  $<$  and  $=$  relations:

$$\Theta(3^n), \Theta(2^n), \Theta(\lg n), \Theta(n^{1.1}), \Theta(n^{0.9}), \Theta(\lg \lg n), \Theta(n \lg n).$$

► Solution

## some facts to know

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} \quad \text{and} \quad e^x = \lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n.$$

- ▶ (for  $|x| < 1$ )  $1 + x \leq e^x \leq 1 + x + x^2$
- ▶ (for  $x \rightarrow 0$ )  $e^x = 1 + x + \Theta(x^2)$

Stirling's approximation:

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right) \quad \text{and} \quad n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\alpha_n}$$

where  $\frac{1}{12n+1} < \alpha_n < \frac{1}{12n}$ .

- ▶  $\lg(n!) = \Theta(n \lg n)$

Section 3.2 has a treasure of these facts.

## Exercise 3

### Exercise

*Knowing the Sterling's approximation:*

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right),$$

*Show that*

$$n! = o(n^n) \quad n! = \omega(2^n).$$

► Solution

## Exercise 4

### Exercise

Is the  $c_1/c_2/N_0$  asymptotic complexity obsolete? Not so.  
Show the following two statements are equivalent:

- ▶  $f(n)$  is **polynomially bounded**:  $f(n) = O(n^a)$  for some  $a > 0$ .
- ▶  $\lg(f(n)) = O(\lg n)$ .

▶ Solution

# Q & A



below are reference answers

## Answer to ▶ Exercise 0

Proof.

Let me show you my thought process.

$\max(f(n), g(n))$  is clumsy to work with  $\rightarrow$  simplify it by case analysis.

$$\max(f(n), g(n)) = \begin{cases} f(n) & \text{if } f(n) \geq g(n) \geq 0 \\ g(n) & \text{if } 0 \leq f(n) < g(n) \end{cases}.$$

This is better. Now we deal with each case separately.

Take the  $f(n) \geq g(n) \geq 0$  case for a start. (Means-ends analysis) By the definition of asymptotic complexity, we are trying to establish relationship between  $f(n)$  and  $f(n) + g(n)$  under the assumption  $f(n) \geq g(n) \geq 0$ . What can we do?

$f(n) \leq f(n) + g(n)$  is obvious.

$f(n) = \frac{1}{2}(2f(n)) = \frac{1}{2}(f(n) + f(n)) \geq \frac{1}{2}(f(n) + g(n))$  is a little bit tricky, but not so much once you keep the “ends” in mind.

The other case follows the same reasoning. □

## Answer to ▶ Exercise 1

Proof.

This is an exercise of recalling and applying definition.

Recall from Calculus classes,  $0 < \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c < \infty$  implies that for any  $\epsilon > 0$ , we can find an  $N(\epsilon)$  such that  $\forall n \geq N(\epsilon)$ , we have

$$c - \epsilon < \frac{f(n)}{g(n)} < c + \epsilon.$$

Choose  $\epsilon = c/2$ , then by the aforementioned implication,  $\exists N(c/2)$  such that  $\forall n \geq N(c/2)$ , we have

$$\frac{1}{2}c = c - \frac{1}{2}c < \frac{f(n)}{g(n)} < c + \frac{1}{2}c = \frac{3}{2}c.$$

Take  $c_1 = \frac{1}{2}c$ ,  $c_2 = \frac{3}{2}c$ , and  $N_0 = N(c/2)$ , we have our proof of  $f(n) = \Theta(g(n))$ . □

## Answer to ▶ Exercise 2

Proof.

$$\Theta(\lg \lg n) < \Theta(\lg n) < \Theta(n^{0.9}) < \Theta(n \lg n) < \Theta(n^{1.1}) < \Theta(2^n) < \Theta(3^n).$$

▶ Exercise 2 gives meanings to ▶ Exercise 1: It is not easy to find  $c_1/c_2/N_0$  by hunch in these cases, but trivially to show the results by the limit form.

Example:  $\lim_{n \rightarrow \infty} 2^n/3^n = \lim_{n \rightarrow \infty} (2/3)^n = 0$  implies that  $2^n = o(3^n)$  and hence  $\Theta(2^n) < \Theta(3^n)$  by our definition.

Another example:  $\lim_{n \rightarrow \infty} (\lg \lg n)/\lg n = \lim_{k \rightarrow \infty} \lg k/k = 0$ .

▶ Exercise 2 helps us understand:

- ▶ any positive **polynomial** function ( $\Theta(n^a)(a > 0)$ ) grows faster than any **polylogarithmic** function ( $\Theta(\lg^b n)(b > 0)$ )

since  $\lim_{n \rightarrow \infty} \lg^b n/n^a = \lim_{k \rightarrow \infty} k^b/(2^k)^a = \lim_{k \rightarrow \infty} k^b/(2^a)^k = 0$ .  $\square$

## Answer to ▶ Exercise 3

Proof.

(Hint) Using the limit form from ▶ Exercise 1.



## Answer to ▶ Exercise 4

Proof.

“Show equivalence” demands us to show *bi-directional* implications.

- ▶  $f(n) = O(n^a) \rightarrow \exists c, N_0$  such that  $\forall n > N_0, f(n) \leq cn^a \rightarrow \lg(f(n)) \leq \lg(cn^a) = a \lg(cn) = a(\lg c + \lg n) = O(\lg n) \rightarrow \lg(f(n)) = O(\lg n)$ .
- ▶ Similar for the reverse direction by taking exponentials.

This exercise uses the  $c_1/c_2/N_0$  definition of asymptotic complexity. □