Introduction to Complexity

Week 1

From nano-second to eternity

Alexandra Silva & Henk Barendregt

Introduction

Our processors are fast: 4 GHz

This means that in one second 4.000.000.000 actions are preformed

That is each action takes only $\frac{1}{4{,}000{,}000{,}000} = \frac{1}{4}10^{-9} = 2.5 \times 10^{-10}$s

Assume that in every nano-second (10^{-9}s) a useful action is done

In one second there are more nano-seconds (10^{9}) than ordinary seconds in a year (31.536×10^{6})!

But speed also has its disadvantages

In $\frac{1}{4{,}000{,}000{,}000}$s light crawls only only 7.5cm!

Electrical signals travel more slowly

Therefore we should minimize access to disc/flash memory

The 'modest' game inventor

A game inventor presented his new creation to the king: chess

The king liked it very much and offered a reward

1	2	4	8	16	32	64	2^7
2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}
2^{24}							
2^{32}							
2^{40}							
2^{48}							
2^{56}							2^{63}

"This is what I'd like to ask:
a grain of rice on the first square
two on the second
four on the third, etcetera,
each time doubling
the previous amount
until the last square"

The king thought it was a modest inventor, but he was wrong ...

Let $S_n = \Sigma_{k=0}^{n-1} 2^k$. How much is S_{64}?

Analysis

We have

$$\begin{array}{rcl} S_n & = & 2^0 + 2^1 + \quad \ldots + 2^{n-2} + 2^{n-1} \\ 2S_n & = & \qquad 2^1 + 2^2 + \quad \ldots \quad + 2^{n-1} + 2^n \end{array}$$

Hence $S_n = 2S_n - S_n = 2^n - 1$ and therefore

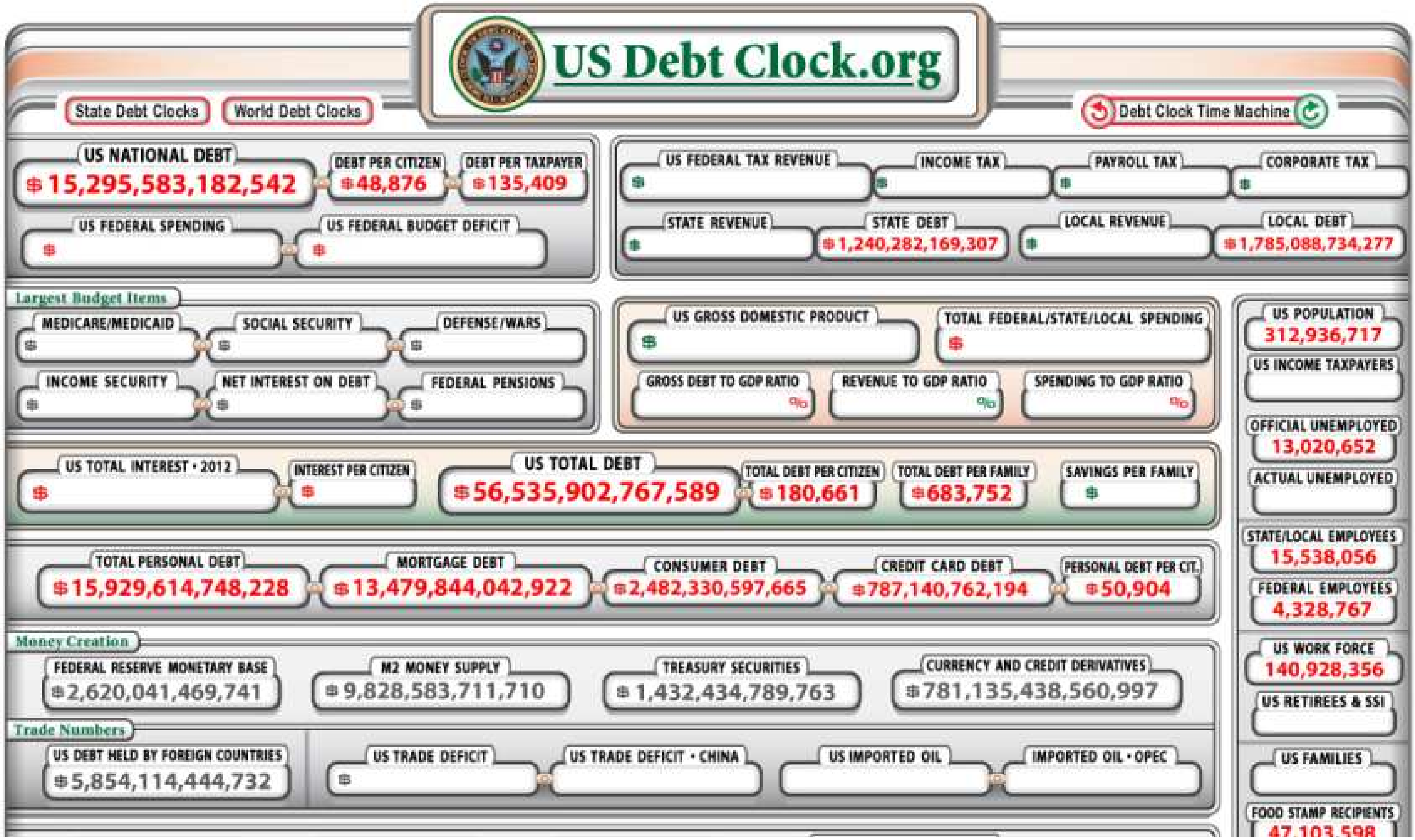
$$S_{64} = 2^{64} - 1$$

A simple experiment in my kitchen (and a lookup in wikipedia) showed that this covers the total area of the Netherlands with about 2m rice! Note that $2^{10} = 1024 \sim 10^3$, hence

$$2^{64} = 16 \times 2^{60} = 16 \times (20^{10})^6 \sim 16 \times (10^3)^6 \sim 10^{19}$$

Supra astronomical debt

Compare this with the US-debt which is $\sim 10^{13}\$$



Fast growing functions

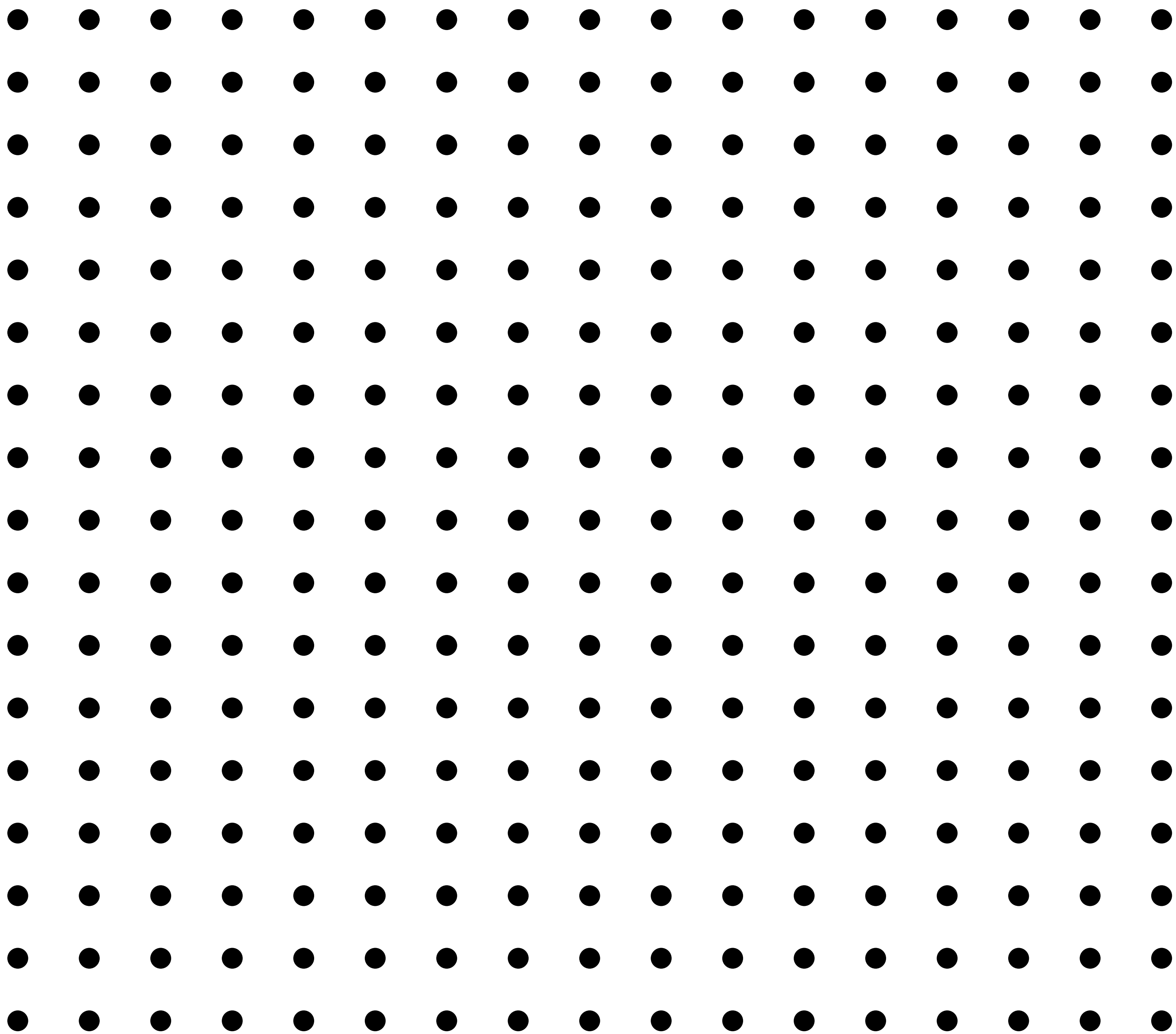
$a_n = \left. a^{a^{a^{\cdots^{a}}}} \right\} (n-1)$-times that is $a_0 = 1$; $a_{n+1} = a^{a_n}$

Note that

$$\begin{aligned} \text{mygoogle} &:= 10^{10} &= 10_2 \\ \text{mygoogleplex} &:= 10^{10^{10}} &= 10_3 \end{aligned}$$

If we keep iterating this idea, then we get the Ackermann function

Supra astronomical numbers at your finger tips



Each bullet (17^2) represents a binary switch {on, off}

The number of possible states is larger than the amount of elementary particles in the universe according to the present theory

Impact of complexity

Suppose that in order to solve a problem of size n it takes $T(n)$ μs. Then depending on $T(n)$ one can solve size n in different times.

$T(n)$	1sec	1min	1hr	1day	1month	1yr	1century
$\lg n$	$10^{3.10^5}$						$10^{3.10^5+9}$
$\sqrt{n}$	10^{12}	10^{15}	10^{19}	10^{23}	10^{24}	10^{27}	10^{29}
n	10^6	10^8	10^9	10^{11}	10^{12}	10^{13}	10^{15}
$n \lg n$	10^4						10^{14}
n^2	10^3	7.10^3	6.10^4	3.10^5	10^6	6.10^6	5.10^7
n^3	10^2	4.10^2	10^3	4.10^3	10^4	3.10^4	10^5
2^n	20	26	32	36	41	44	51
$n!$	10						17
2_n	4						4

A problem with hyper-exponential complexity

FACT In the simply typed lambda calculus the problem whether $M = N$ is hyper-exponential
PROOF-SKETCH Let $\mathbf{c}_n := \lambda f \lambda x. f^n x := f, x \mapsto f^n x$
Church's numerals,

where $\begin{array}{rcl} f^0 x & := & x \\ f^{n+1} x & := & f(f^n x) \end{array}$ e.g. $\mathbf{c}_2 := \lambda f x. f(fx)$, $\mathbf{c}_3 := \lambda f x. f(f(fx))$

Then one has

$$\begin{array}{l} \mathbf{c}_n : (0 \to 0) \to 0 \to 0 \\ \mathbf{c}_n \mathbf{c}_m = \mathbf{c}_{m^n} \end{array}$$

and hence $\underbrace{\mathbf{c}_n \ldots \mathbf{c}_n}_{k \text{ times}} = \mathbf{c}_{n_k}$ E.g.

$$\mathbf{c}_2 \mathbf{c}_2 \mathbf{c}_2 = \mathbf{c}_{2^2} \mathbf{c}_2 = \mathbf{c}_{2^{2^2}} = \mathbf{c}_{2_3}$$

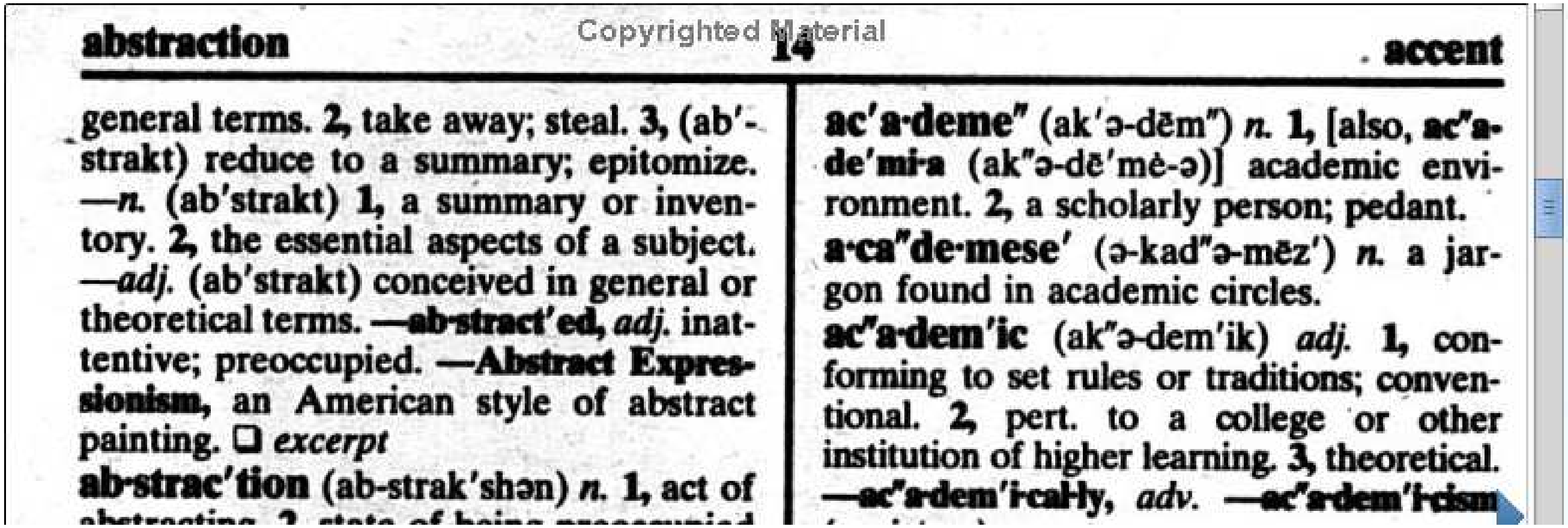
This makes it plausible that the cost of deciding $M = N$ is at least hyper-exponential

Looking up a word in a dictionary

Given a dictionary with n pages (say $n = 1000$).

How fast can we look-up a word? This is a problem of complexity $\lg n$

We want to look-up e.g. the word 'abdomen'. We open the book in the beginning



abstraction 14 **accent**

general terms. 2, take away; steal. 3, (ab'-strakt) reduce to a summary; epitomize. —*n.* (ab'strakt) 1, a summary or inventory. 2, the essential aspects of a subject. —*adj.* (ab'strakt) conceived in general or theoretical terms. **—ab·stract'ed,** *adj.* inattentive; preoccupied. **—Abstract Expressionism,** an American style of abstract painting. ❑ *excerpt*

ab·strac'tion (ab-strak'shən) *n.* 1, act of

ac'a·deme" (ak'ə-dēm") *n.* 1, [also, **ac"a·de'mi·a** (ak"ə-dē'mē-ə)] academic environment. 2, a scholarly person; pedant.

a·ca"de·mese' (ə-kad"ə-mēz') *n.* a jargon found in academic circles.

ac"a·dem'ic (ak"ə-dem'ik) *adj.* 1, conforming to set rules or traditions; conventional. 2, pert. to a college or other institution of higher learning. 3, theoretical. **—ac"a·dem'i·cal·ly,** *adv.* **—ac"a·dem'i·cism**

We have to go to the left, as 'abdomen' $<$ 'abstraction' in the lexicographical order

Remembering the page with 'abstraction',

we split the pages before in two and see if we need to go left or right.

Etcetera

A problem with logarithmic complexity

We want to compute the costs $T(n)$, the nuber of steps it takes to look up a word in a dictionary of n pages

Abstracting from the costs to open the book, to keep your fingers in it

to compare the word we look up with words in the dictionary

we say: this all has costs 1 step

Then

$$\begin{aligned} T(n) &= 1 + T(\lfloor \tfrac{n}{2} \rfloor) \\ &= 1 + 1 + T(\lfloor \tfrac{n}{4} \rfloor) \\ &= 1 + 1 + 1 + T(\lfloor \tfrac{n}{8} \rfloor) \\ &\dots \end{aligned}$$

How often can we do this maximally? Answer: $1 + \log_2 n$ times, so

$$T(n) \sim \log_2 n =: \lg n$$

Comparing functions

Let $f : \mathbb{N} \to \mathbb{N}$. The book writes $f(n) = \Theta(n^2)$ if

$$\exists c_1, c_2 \in \mathbb{R}_{>0} \forall^* n . c_1 n^2 \leq f(n) \leq c_2 n^2$$

[Here $\forall^* n.P(n)$ (*for almost all* n) means $\exists n_0 \forall n \geq n_0 . P(n)$]

This is an ugly notation; better write $f(n) \in \Theta(n^2)$

Even better, let $g : \mathbb{N} \to \mathbb{N}$. Then

$$\Theta(g) = \{ f \mid \exists c_1, c_2 \in \mathbb{R}_{>0} \forall^* n \, [\, c_1 g(n) \leq f(n) \leq c_2 g(n) \,] \}$$

Thus $f(n) \in \Theta(n^2)$ (or $f(n) = \Theta(n^2)$) should be officially written as

$$f \in \Theta(\lambda n . n^2)$$

Important is that $\Theta(g)$ is a class of functions

We still will use the notation of the book

More function classes

Definition

$$O(g) = \{f : \mathbb{N}\to\mathbb{N} \mid \exists c{\in}\mathbb{R}_{>0}\forall^* n.0 \le f(n) \le cg(n)\}$$

$$\Omega(g) = \{f : \mathbb{N}\to\mathbb{N} \mid \exists c{\in}\mathbb{R}_{>0}\forall^* n.0 \le cg(n) \le f(n)\}$$

Proposition

$$\Theta(g) = O(g) \cap \Omega(g)$$

Write $f \le_O g$ for $f{\in}O(g)$

$f \ge_\Omega g$ for $f{\in}\Omega(g)$

$f =_\Theta g$ for $f{\in}\Theta(g)$

We have

$$f \le_O g \;\&\; g \le_O h \quad\Rightarrow\quad f \le_O h$$

$$f \ge_\Omega g \;\&\; g \ge_\Omega h \quad\Rightarrow\quad f \ge_\Omega h$$

$$f =_\Theta g \;\&\; g =_\Theta h \quad\Rightarrow\quad f =_\Theta h$$

$$f =_\Theta g \quad\Leftrightarrow\quad g =_\Theta f$$

$$f \le_O g \quad\Leftrightarrow\quad g \ge_\Omega f.$$

[so we can write simply $f \le g,\ g \ge f$]

So all the notions can be defined from $\le_O$, that is from $O(g)$

Exercises. (i) $2^{n+1}{\in}O(2^n)$. (ii) $n^{10}{\in}O(2^n)$. (iii) $2^{2n} \notin O(2^n)$.

(iv) Find f, g such that $f \not\le g \;\&\; g \not\le f$.