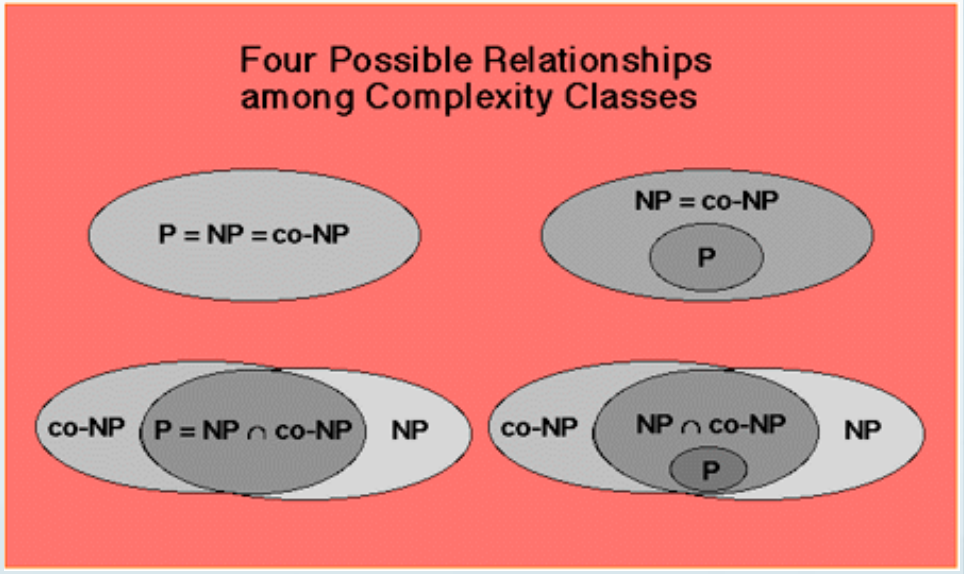
Θεωρία Υπολογισμού



Κυρίες και Κύριοι,
Ήρθε η ώρα της ...

Υπολογιστικής Πολυπλοκότητας

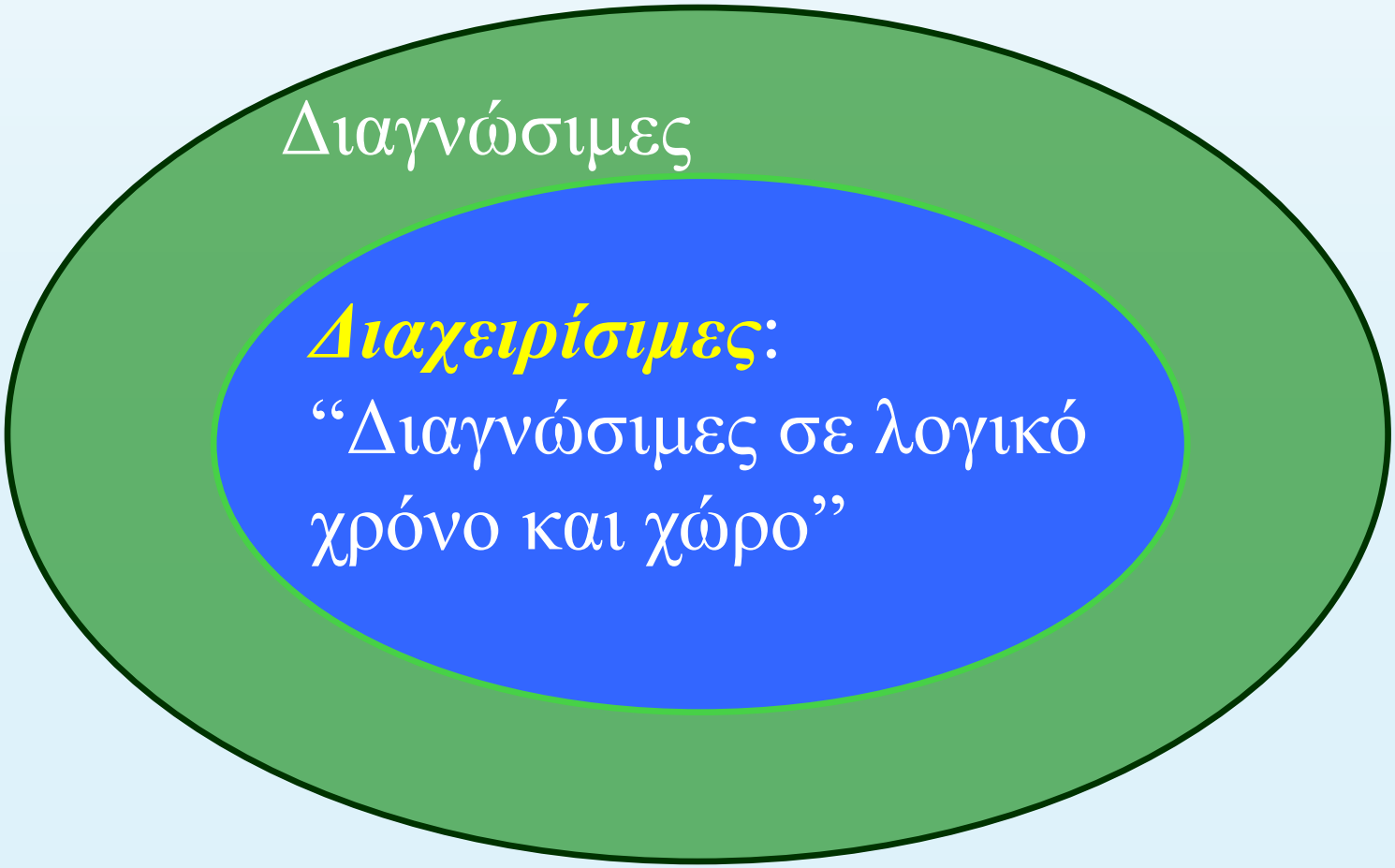
Αναπλ. Καθηγητής Τσίχλας Κωνσταντίνος

Μέχρι σήμερα

Μη-Αναγνωρίσιμες

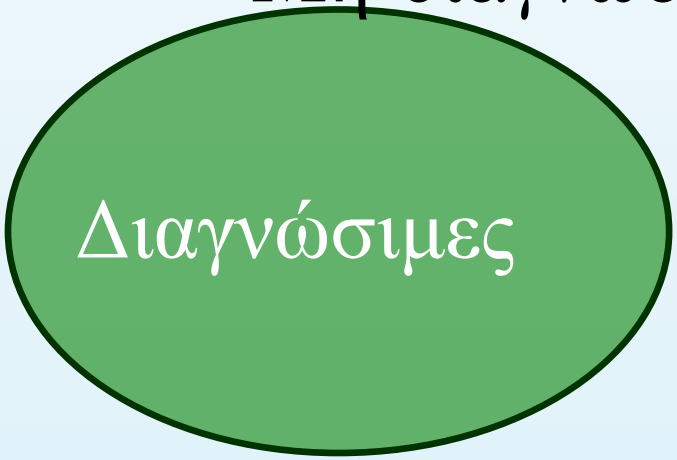
Από Σήμερα

Μη-διαγνώσιμες



Υπολογισιμότητα Πολυπλοκότητα

Μη-διαγνώσιμες



~1930 – … (;;;)

1900: Προβλήματα του Hilbert
1936: Οι *Υπολογίσιμοι Αριθμοί* του Turing
1957: Γλώσσες Chomsky

Ανοικτό πεδίο έρευνας

Μη-διαχειρίσιμες

~1960 – … (;;;)

1960: Κλάσεις Πολυπλοκότητας
1971: Cook/Levin, Karp: P=NP;
1976: Knuth O, Ω, Θ, o, ω

Περισσότερο ανοικτό και πολύ ζωντανό πεδίο έρευνας

Κλάσεις Πολυπλοκότητας

- *Κλάσεις Υπολογισιμότητας*: σύνολα προβλημάτων που μπορούν να λυθούν (διάγνωση/αναγνώριση) από μία ΤΜ
- *Κλάσεις Πολυπλοκότητας*: σύνολα προβλημάτων που μπορούν να λυθούν (διάγνωση) από μία ΤΜ σε ***περιορισμένο χρόνο και χώρο***

Πόσες κλάσεις πολυπλοκότητας υπάρχουν;

Άπειρες! «Οι γλώσσες που μπορούν να διαγνωσθούν από μία ΤΜ σε λιγότερα από 37 βήματα» είναι μία κλάση πολυπλοκότητας

Ενδιαφέρουσες Κλάσεις Πολυπλοκότητας

https://complexityzoo.net/Complexity_Zoo

Complexity Zoo

Introduction

Welcome to the **Complexity Zoo**... There are now 547 classes and counting!

Complexity classes by letter: Symbols - A - B - C - D - E - F - G - H - I - J - K - L - M - N - O - P - Q - R - S - T - U - V - W - X - Y - Z

Lists of related classes: Communication Complexity - Hierarchies - Nonuniform

Zookeeper

Scott Aaronson

Veterinarian

Greg Kuperberg

Zoo Conservationist

Oliver Habryka on behalf of the LessWrong community

what's your problem?

The Zoo first opened in 2002. It was made into a wiki in 2005, and hosted at the University of Waterloo from 2012 to 2020.

Errors? Omissions? Misattributions? Your favorite class not here? Then please contribute to the zoo as you see fit by signing up and clicking on the edit links. Please include references, or better yet links to papers if available.

To create a new class, click on the edit link of the class before or after the one that you want to add and copy the format of that class. (The classes are alphabetized by their tag names.) Then add the class to the table of contents and increment the total number of classes. After this, you can use the side edit links to edit the individual sections. For more on using the wiki language, see the Mediawiki Help page.

If you would like to contribute but feel unable to make the updates yourself, email the zookeeper at scott at scottaaronson.com.

547 (πέρσι 546, πριν 9 χρόνια 495) «ενδιαφέρουσες» κλάσεις πολυπλοκότητας (και συνεχίζεται)!

The “Petting Zoo”

Petting Zoo

Main Zoo - Complexity Garden - Zoo Glossary - Zoo References

Under construction! Once finished, the Petting Zoo will introduce complexity theory to newcomers unready for the terrifying and complex beasts lurking in the main zoo. It will be a gentler source of information about major complexity classes, problems, and reductions. It is meant to be self-contained and does not require previous knowledge of complexity theory, aside from some basic terminology that can be found in the glossary. In particular, one should be familiar with the terms alphabet, word and language.

Contents [hide]

1 Modeling Computation
- 1.1 Deterministic Turing Machine
  - 1.1.1 The Tape
  - 1.1.2 The States
  - 1.1.3 Computing and Deciding Problems
  - 1.1.4 Power of a Turing Machine
- 1.2 Non-deterministic Turing Machine

2 Most Important Classes
- 2.1 P
- 2.2 NP and coNP
  - 2.2.1 NP-complete
- 2.3 PH
- 2.4 PSPACE
- 2.5 EXP
- 2.6 AC^0
- 2.7 NC
- 2.8 L
- 2.9 P/poly
- 2.10 BPP
- 2.11 MA
- 2.12 AM
- 2.13 SZK
- 2.14 BQP
- 2.15 #P
- 2.16 PP

3 What Next?

Modeling Computation

Υπολογιστική Πολυπλοκότητα

Αποτελεί εισαγωγή στην θεωρία πολυπλοκότητας για *ψάρακες* που δεν είναι έτοιμοι να αντιμετωπίσουν τα θηρία του μεγάλου ζωολογικού κήπου.

Απλά θα εισέρθουμε στο «Petting Zoo». Ακόμα και εδώ όμως θα βρείτε τρομερά θηρία!!!

Το πιο άγριο θηρίο (με την έννοια του πιο γνωστού):

Περίπτωση 1: Υπάρχουν προβλήματα στην κλάση *NP* που δεν είναι διαχειρίσιμα

Περίπτωση 2: Όλα τα προβλήματα στην κλάση *NP* είναι διαχειρίσιμα

P = NP ?

- Θα χρειαστούμε 2 μαθήματα ακόμα για να το εξηγήσουμε
- Ανοικτό πρόβλημα: κανείς δεν ξέρει την απάντηση
  - Αν το απαντήσετε, έχετε δόξα, χρήματα και ένα 10 στο μάθημα!
  - Θα πρέπει πρώτα να έχετε μία διαίσθηση σε σχέση με το τι θα είναι η απάντηση και τι θα σημαίνει

Μέτρηση Πολυπλοκότητας

- Τί μετράμε;
- Πολυπλοκότητα Χειρότερης Περίπτωσης
- Πολυπλοκότητα Μέσης Περίπτωσης
- …

Ασυμπτωτική Ανάλυση

Σημειογραφία

$O(f)$, $\Omega(f)$, $o(f)$, $\Theta(f)$

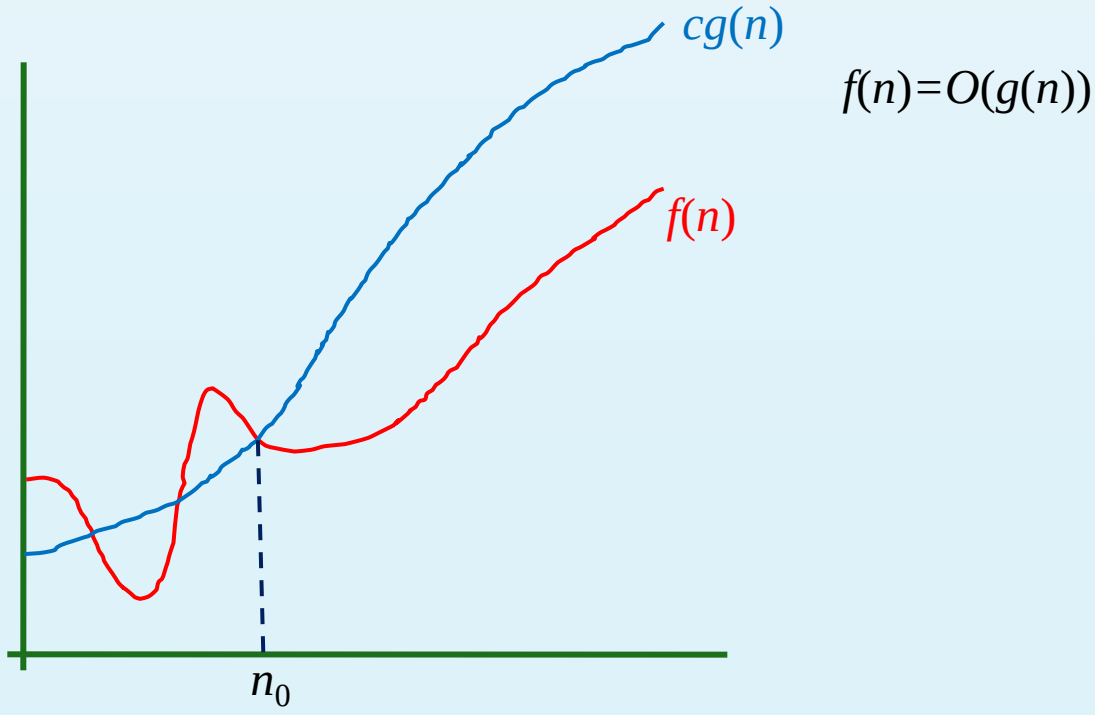
- Ορίζουμε **σύνολα συναρτήσεων**

- Ενδιαφερόμαστε για το πώς το μέγεθος της εξόδου σχετίζεται με το μέγεθος της εισόδου

Big *O*

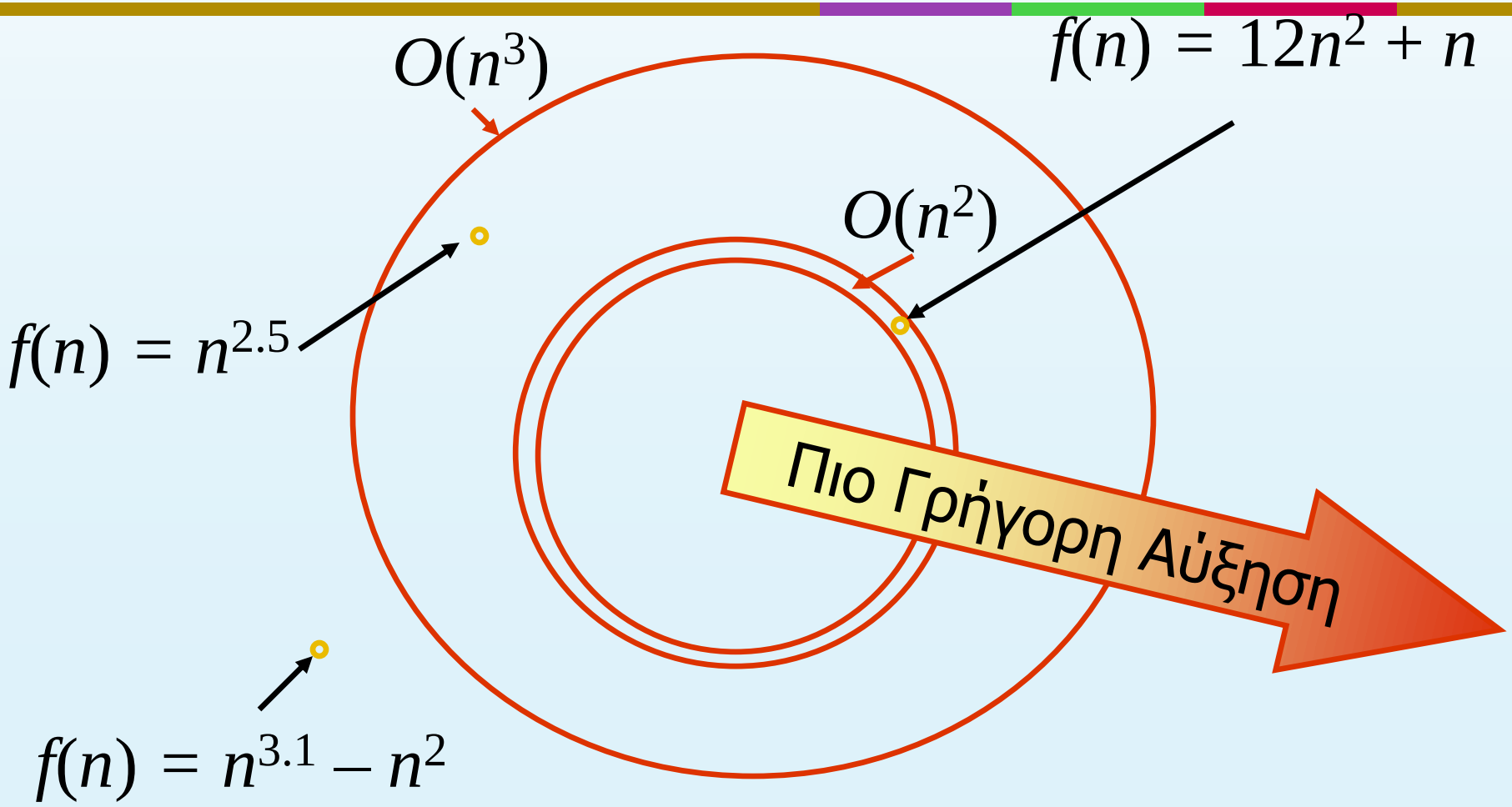
$f \in O(g)$ σημαίνει:
Υπάρχουν θετικές σταθερές c και n_0 έτσι ώστε
$f(n) \leq cg(n)$
για κάθε $n \geq n_0$.

- Διαίσθηση: το σύνολο $O(f)$ είναι το σύνολο των συναρτήσεων που αυξάνονται όχι πιο γρήγορα από την f



$f(n)=O(g(n))$

Παραδείγματα



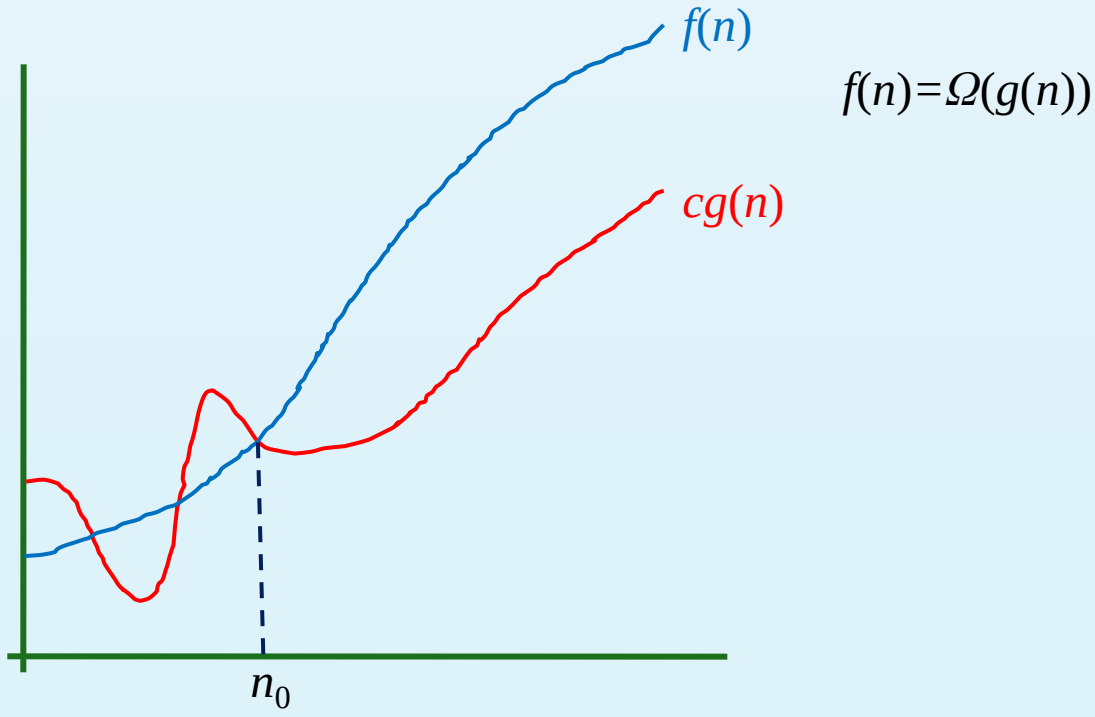
Κάτω Φράγμα: Ω

$f \in \Omega(g)$ σημαίνει:
Υπάρχουν θετικές σταθερές c και n_0 έτσι ώστε

$$f(n) \geq cg(n)$$

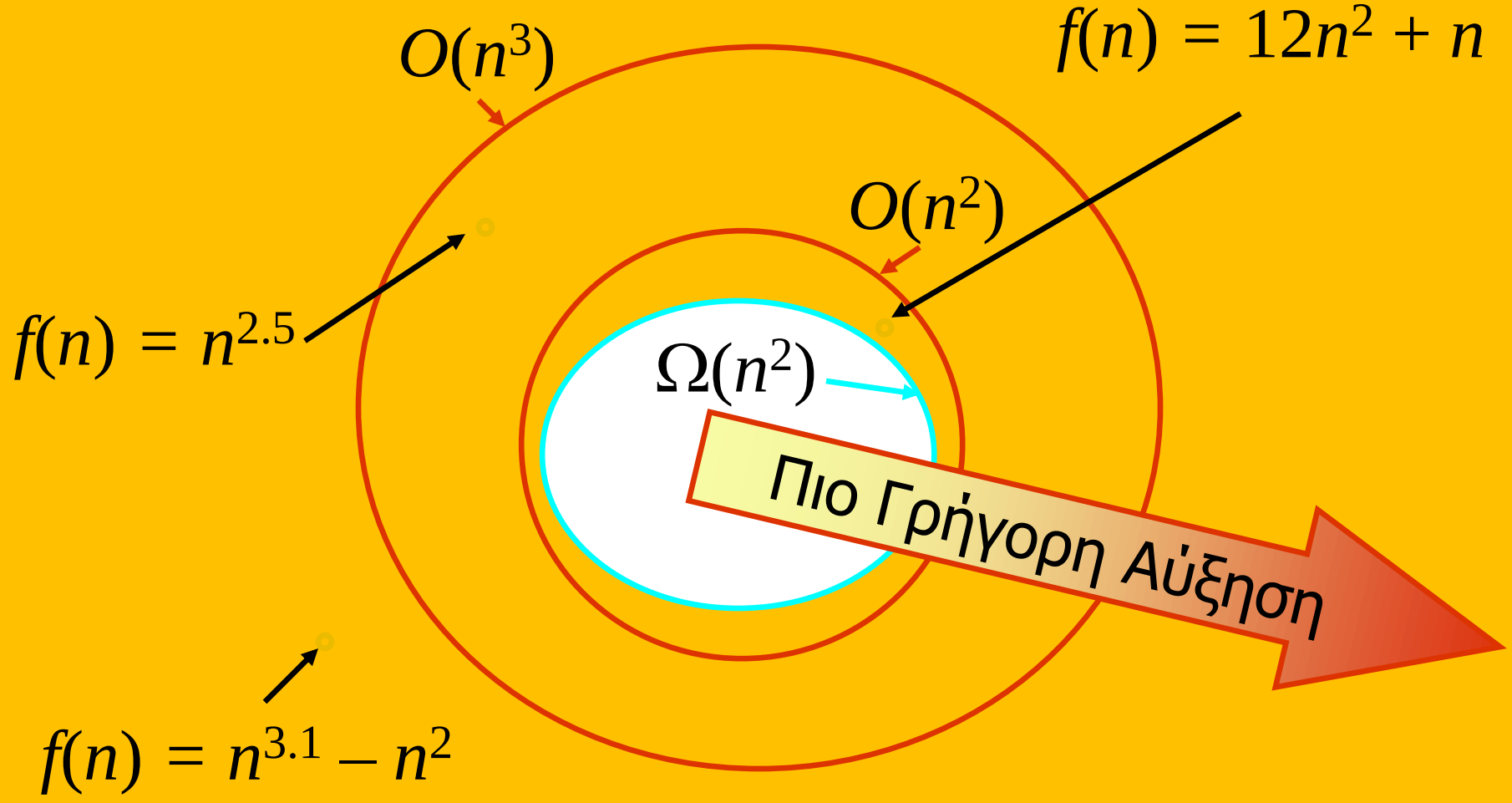
για κάθε $n \geq n_0$.

Ω: το σύνολο $\Omega(f)$ είναι το σύνολο των συναρτήσεων που αυξάνονται *όχι πιο αργά* σε σχέση με την f



$f(n)=\Omega(g(n))$

Που είναι το $\Omega(n^2)$;

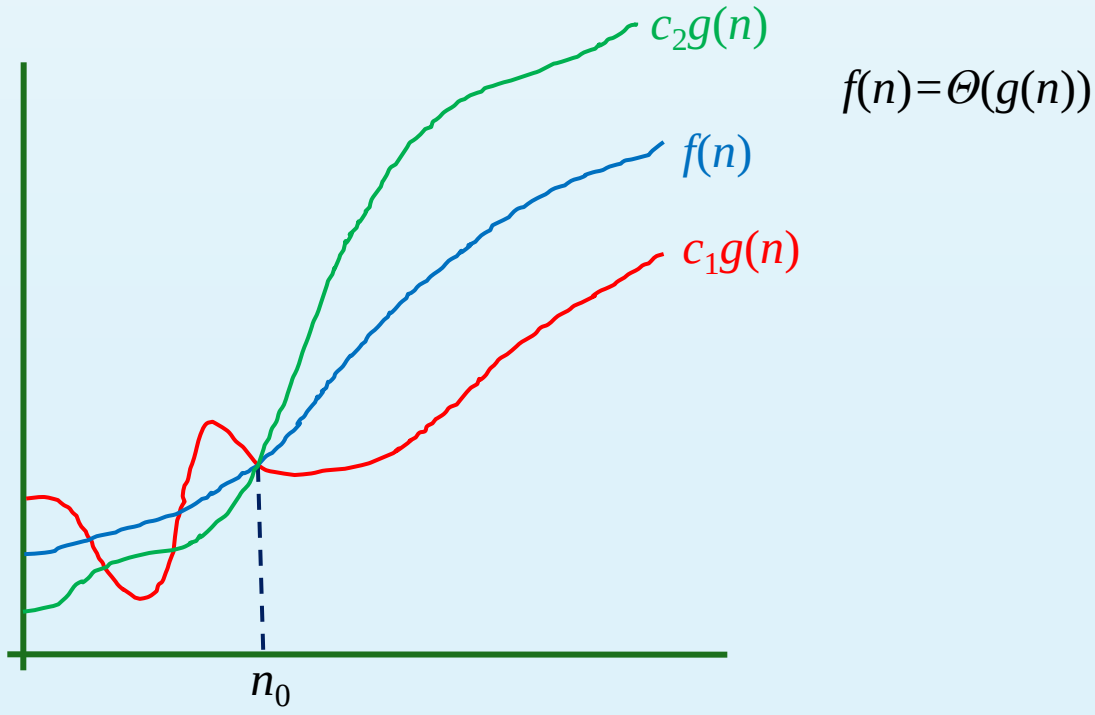


Θήτα (Θ)

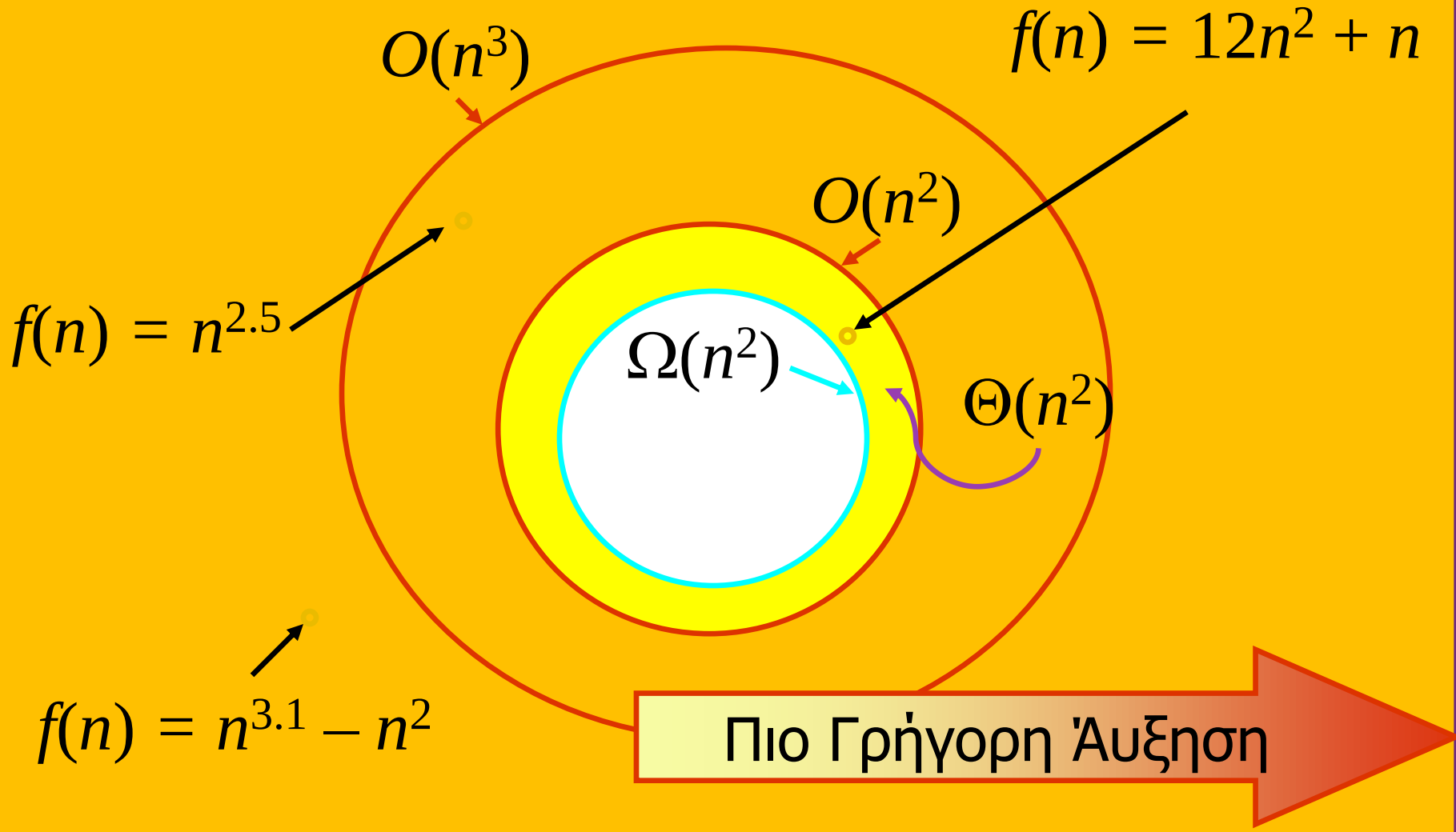
$f(n) \in \Theta(g(n))$ αν και μόνο αν ισχύουν:

1. $f(n) \in O(g(n))$
2. $f(n) \in \Omega(g(n))$

Το σύνολο $\Theta(f)$ είναι το σύνολο των συναρτήσεων που αυξάνονται *τόσο γρήγορα όσο και* η f



Αυστηρό Φράγμα Θήτα (Θ)



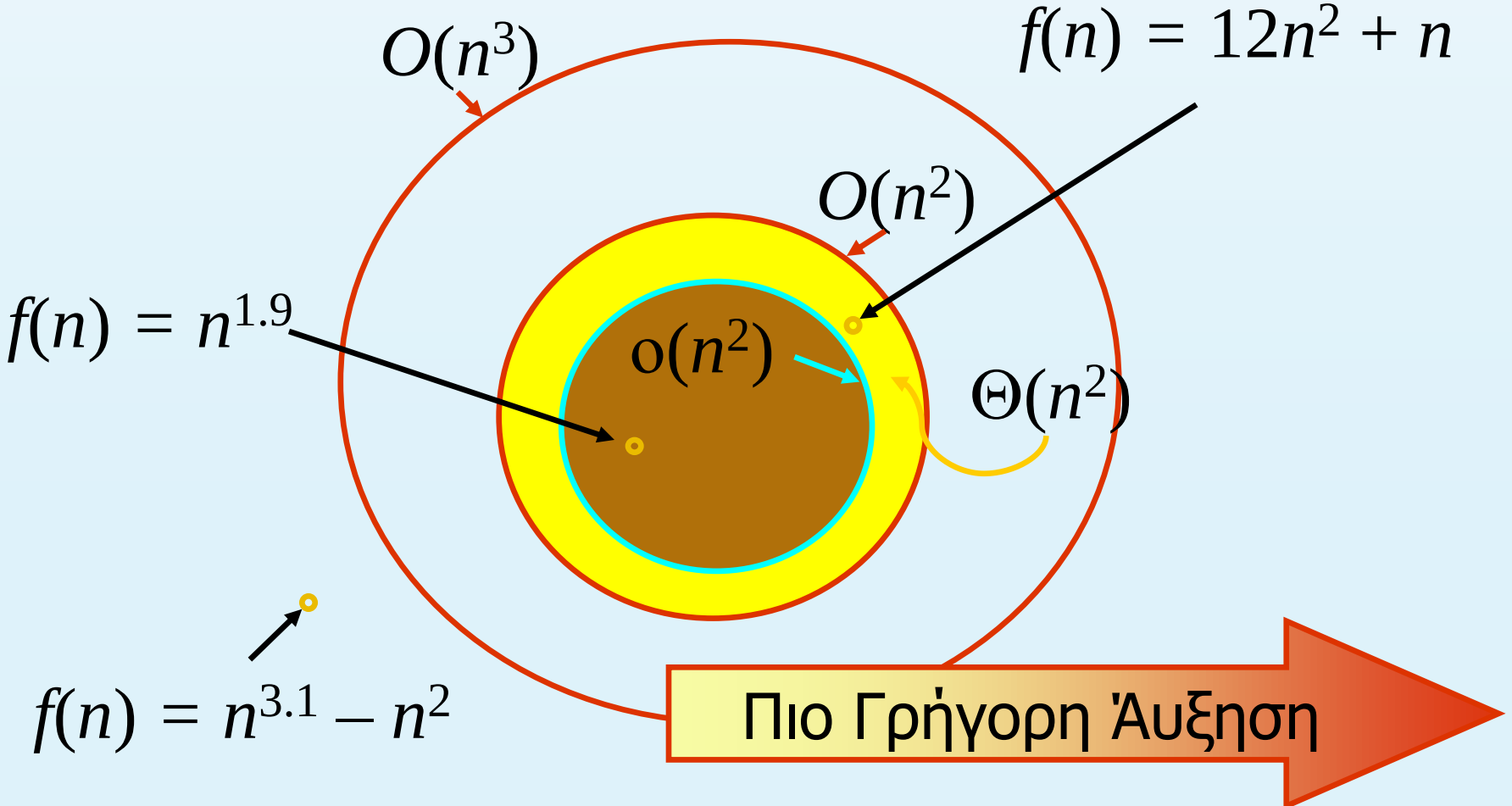
Little *o*

$f \in o(g)$ σημαίνει:
Για κάθε θετική σταθερά c και για κάποιο n_0 έχουμε:

$$f(n) < cg(n)$$

για κάθε $n \geq n_0$.

Διαίσθηση: το σύνολο $o(f)$ είναι το σύνολο των συναρτήσεων που αυξάνονται *αυστηρά* όχι πιο γρήγορα από την f



Ανάλυση Αλγορίθμων

- Χρησιμοποιώντας O, ποιος είναι ο χρόνος εκτέλεσης του αλγόριθμου X?

$$O(n^{n^{n}})$$

Αυτό περιέχει σίγουρα όλους τους αλγόριθμους που έχετε δει μέχρι τώρα.

Σωστή ερώτηση: Χρησιμοποιώντας Θ, ποιος είναι ο χρόνος εκτέλεσης του αλγορίθμου X?

Δοθέντος ενός αλγόριθμου, θα πρέπει πάντα να βρίσκουμε ένα αυστηρό φράγμα.

Πολυπλοκότητα Προβλημάτων

Άρα γιατί χρειαζόμαστε τα O και Ω?

Σε αυτό το μάθημα μας ενδιαφέρει η πολυπλοκότητα των ***προβλημάτων*** και όχι των ***αλγορίθμων***. Η πολυπλοκότητα ενός προβλήματος είναι ίση με την πολυπλοκότητα του καλύτερου αλγόριθμου που επιλύει το πρόβλημα.

Στο ψητό…

Έστω η γλώσσα: $A = \{0^n 1^n | n \geq 0\}$

- Αυτή η γλώσσα είναι διαγνώσιμη.
- **Ερώτηση:** Πόσο χρόνο χρειάζεται μία ΤΜ (αλγόριθμος) με μία ταινία για να την λύσει;

Ο αλγόριθμος

Συνολικός χρόνος $O(n^2)$

M_1 σε είσοδο w:

1. Διαπέρασε την ταινία και απόρριψε αν βρεις 0 δεξιά ενός 1.

$O(n)$ βήματα πρέπει να γίνουν

2. Όσο 0 και 1 εμφανίζονται στην ταινία, επανέλαβε:
   1. Διαπέρασε την ταινία, διαγράφοντας ένα 0 και ένα 1 σε κάθε πέρασμα.

$O(n^2)$ βήματα πρέπει να γίνουν

3. Αν κανένα 0 και 1 δεν έχουν παραμείνει τότε αποδέξου, αλλιώς απόρριψε.

$O(n)$ βήματα πρέπει να γίνουν

Χρόνος Εκτέλεσης

Έστω M μία αιτιοκρατική TM, και έστω:

$$t: \mathbb{N} \to \mathbb{N}$$

Λέμε ότι η M εκτελείται σε χρόνο $t(n)$ αν:

Για κάθε είσοδο x μήκους n, το πλήθος των βημάτων που η M κάνει είναι το μέγιστο $t(n)$.

Κλάση Χρονικής Πολυπλοκότητας

Έστω $t: \mathbb{N} \to \mathbb{N}$ μία συνάρτηση:

Ορισμός:

$\mathrm{DTIME}(t(n)) = \{L \mid L$ είναι μία γλώσσα που διαγιγνώσκεται από μία TM σε χρόνο $O(t(n))\}$

Έχουμε δει ότι για τη γλώσσα $A = \{0^n 1^n | n \geq 0\}$ ισχύει ότι:

$A \in \mathrm{DTIME}(n^2)$

Μπορούμε πιο γρήγορα;;;;

Home Improvement

Ένας πιο Γρήγορος Αλγόριθμος

M_2 σε είσοδο w:

Η M_2 τερματίζει;

1. Διαπέρασε την ταινία και απέρριψε αν βρεθεί 0 δεξιά ενός 1.

2. Επανέλαβε τα εξής όσο 0 και 1 εμφανίζονται στην ταινία:
   1. Διαπέρασε την ταινία, ελέγχοντας αν το συνολικός πλήθος των 0 και 1 είναι άρτιος ή περιττός. Αν είναι περιττός απέρριψε.
   2. Διαπέρασε την ταινία, σβήνοντας κάθε δεύτερο 0 (ξεκινώντας από το πρώτο), και κάθε δεύτερο 1 (ξεκινώντας από το πρώτο) σε κάθε διαπέραση.

3. Αν δεν απομένει κανένα 0 ή 1 αποδεχόμαστε, αλλιώς απορρίπτουμε.

Είναι ορθό το πρόγραμμα;

Ανάλυση

M2 σε είσοδο *w*:

1. Διαπέρασε την ταινία και απέρριψε αν βρεθεί 0 δεξιά ενός 1.
2. Επανέλαβε τα εξής όσο 0 και 1 εμφανίζονται στην ταινία:
   1. Διαπέρασε την ταινία, ελέγχοντας αν το συνολικός πλήθος των 0 και 1 είναι άρτιος ή περιττός. Αν είναι περιττός απέρριψε.
   2. Διαπέρασε την ταινία, σβήνοντας κάθε δεύτερο 0 (ξεκινώντας από το πρώτο), και κάθε δεύτερο 1 (ξεκινώντας από το πρώτο) σε κάθε διαπέραση.
3. Αν δεν απομένει κανένα 0 ή 1 αποδεχόμαστε, αλλιώς απορρίπτουμε.

Ένα πέρασμα σε κάθε βήμα (1, 2.1, 2.2, 3) απαιτεί $O(n)$ χρόνο.

Τα βήματα 1 και 3: Εκτελούνται μία φορά μόνο.

Το 2.2 διαγράφει μισά από τα 0 και μισά από τα 1: $1 + \log_2 n$ φορές

Σύνολο για το 2 είναι: $(1 + \log_2 n)O(n) = O(n\log_2 n)$.

Άρα συνολικά: $O(n) + O(n\log_2 n) = O(n\log_2 n)$

Μπορούμε Καλύτερα;

Ερώτηση: Μπορεί ο χρόνος εκτέλεσης να γίνει $o(n\log_2 n)$?

Απάντηση: Όχι σε μία ΤΜ με μία ταινία (απόδειξη αφήνεται για τον ακροατή – δύσκολη).

Ερώτηση: Πρέπει να χρησιμοποιούμε μόνο ΤΜ με μία ταινία;

Απάντηση: Και βέβαια όχι!

Μία ΤΜ με δύο Ταινίες

M3 σε είσοδο *w*:

1. Διαπέρασε την ταινία και ΑΠΟΡΡΙΨΕ αν βρεθεί 0 δεξιά ενός 1.
2. Διαπέρασε τα 0 μέχρι το πρώτο 1,αντιγράφοντάς τα στην ταινία 2.
3. Διαπέρασε τα 1 στην ταινία 1 μέχρι το τέλος. Για κάθε 1 σβήσε ένα 0 από την ταινία 2. Αν δεν υπάρχουν 0 τότε ΑΠΟΡΡΙΨΕ.
4. Αν έχουν μείνει 0, ΑΠΟΡΡΙΨΕ, αλλιώς αποδέξου.

Ερώτηση: Ποιος είναι ο χρόνος εκτέλεσης;

Πολυπλοκότητα

Διάγνωση της γλώσσας $A = \{0^n 1^n | n \geq 0\}$:

Μονοταινιακή M_1: $O(n^2)$

Μονοταινιακή M_2: $O(n\log_2 n)$ (το πιο γρήγορο!!!)

Διταινιακή M_3: $O(n)$

Διαφορά μεταξύ Υπολογισιμότητας και Πολυπλοκότητας:

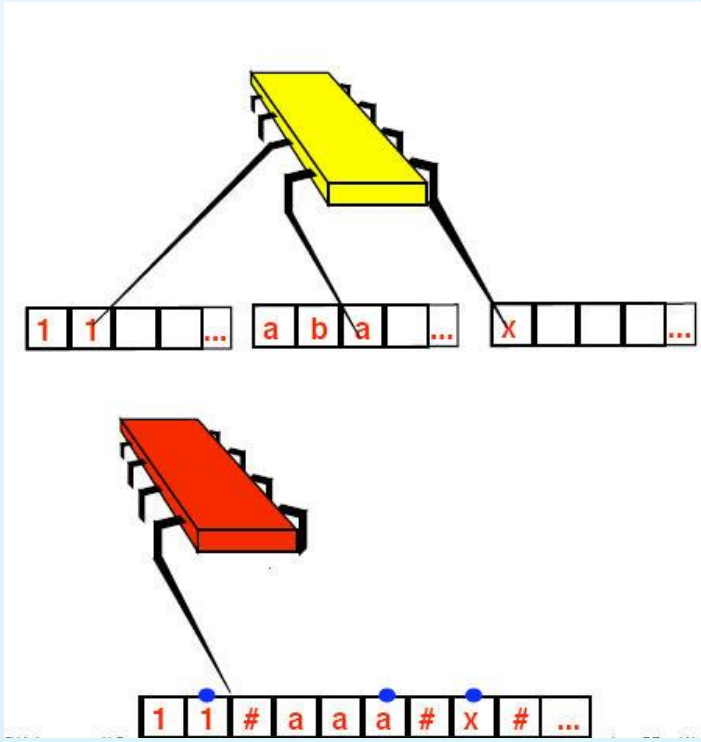
Υπολογισιμότητα: όλα τα λογικά μοντέλα είναι ισοδύναμα (Church-Turing)

Πολυπλοκότητα: επιλογή μοντέλου επηρεάζει τον χρόνο εκτέλεσης.

Ερώτηση: Κατά πόσο το μοντέλο επηρεάζει την πολυπλοκότητα;

Μοντέλα και Υπολογισμός: Πολυταινιακές

Έστω $t(n)$ μία συνάρτηση όπου $t(n) \geq n$, και έστω $L \subseteq \Sigma^*$ μία γλώσσα.

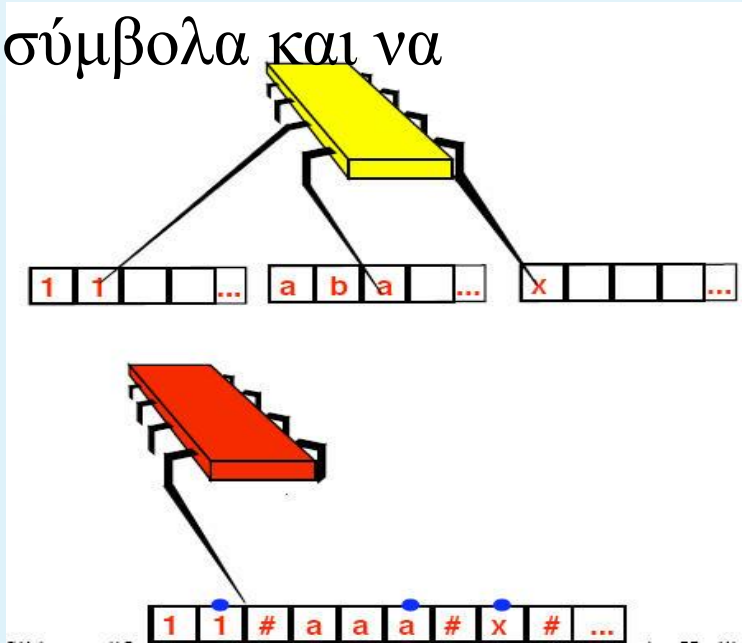


Ισχυρισμός: Αν μία πολυταινιακή ΤΜ διαγιγνώσκει την L σε χρόνο $t(n)$, τότε υπάρχει ΤΜ μίας ταινίας που να διαγιγνώσκει την L σε $O(t^2(n))$ χρόνο.

Εξομοίωση ΤΜ με πολλαπλές ταινίες

Σε είσοδο $w = w_1 \ldots w_n$, η μονοταινιακή ΤΜ S:

- Βάζουμε στην ταινία # w_1 w_2 … w_n# □ # □ #· · ·#
- Διαπερνούμε την ταινία από το πρώτο # έως το (k+1)-οστό # για να διαβάζουμε τα σύμβολα στις εικονικές κεφαλές.
- Επαναδιαπέραση για να γράψει τα νέα σύμβολα και να μετακινήσει τις κεφαλές
- Αν η S προσπαθήσει να μετακινήσει την εικονική κεφαλή εκτός #, τότε η M προσπαθεί να γράψει σε κενό (μετατόπιση).



Πολυπλοκότητα Εξομοίωσης

Για κάθε βήμα της M, η S εκτελεί:

- δύο διαπεράσεις
- Μέχρι k μετακινήσεις προς τα δεξιά (για επιπλέον χώρο)

Σε είσοδο μήκους n, η M εκτελεί $O(t(n))$ βήματα, και άρα το ενεργό μήκος κάθε ταινίας είναι το πολύ $O(t(n))$ μακρύ.

Συνολικό πλήθος βημάτων της S:

- $O(t(n))$ βήματα για να εξομοιώσουμε ένα βήμα της M.
- Εξομοίωση: $O(t(n)) \times O(t(n)) = O(t^2(n))$
- Αρχική κατασκευή της ταινίας της S σε $O(n)$
- Σύνολο: $O(n) + O(t^2(n)) = O(t^2(n))$ βήματα,

Υπό την λογική προϋπόθεση ότι $t(n) > n$.

Σχέσεις μεταξύ Χρονικών Κλάσεων

Έστω $t_1, t_2 : \mathbb{N} \to \mathbb{N}$ δύο συναρτήσεις.

- ***Ισχυρισμός:*** Αν $t_1(n) = O(t_2(n))$ τότε:

$$\text{DTIME}(t_1(n)) \subseteq \text{DTIME}(t_2(n))$$

- Με απλά λόγια, περισσότερος χρόνος δεν κάνει κακό.
- Αλλά βοηθάει πραγματικά;

Ισχυρισμός: Αν $t_1(n) = o(t_2(n)/\log_2 n)$ τότε:

$$\text{DTIME}(t_1(n)) \subset \text{DTIME}(t_2(n))$$

- Με απλά λόγια, επιπλέον διαθέσιμος χρόνος μας βοηθά.
- Αποδείξεις: Λίγο πιο πολύπλοκες διαγωνοποιήσεις – (δεν θα τις κάνουμε ☺)

Ανταιτιοκρατικός Χρόνος

Έστω N μία ανταιτιοκρατική ΤΜ και έστω:

$$t: \mathbb{N} \to \mathbb{N}$$

Θα λέμε ότι η N εκτελείται σε χρόνο $t(n)$ αν

- για κάθε είσοδο x μήκους n,
- Το μέγιστο πλήθος βημάτων που η N εκτελεί,
- σε οποιονδήποτε κλάδο του δέντρου υπολογισμού στο x,
- είναι το πολύ $t(n)$.

Παράδειγμα – Κλίκα

$$L = \{\langle G, k\rangle : \kappa\lambda\iota\kappa\alpha\ \mu\varepsilon\gamma\varepsilon\theta ους \geq k\}$$

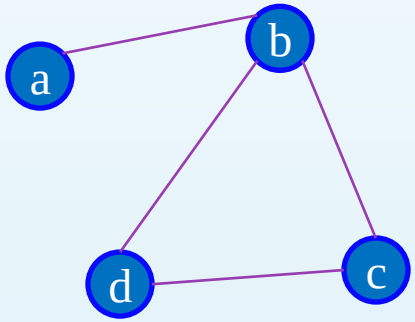
1. Τοποθέτησε $\langle G, k\rangle$ στην ταινία
2. Τοποθέτησε το σύμβολο $ στο τέλος
3. Όσο κινούμαστε προς τα αριστερά μέχρι το $

   Επέλεξε ανταιτιοκρατικά ένα σύνολο κόμβων S του G

   και τοποθέτησέ τα μετά το $
4. Έλεγξε αν $|S| \geq k$ (αν δεν είναι ΑΠΟΡΡΙΨΗ)
5. Για κάθε ζεύγος κόμβων στο S έλεγξε αν είναι γειτονικοί
6. ΑΠΟΔΟΧΗ αν όλα τα ζεύγη είναι γειτονικά

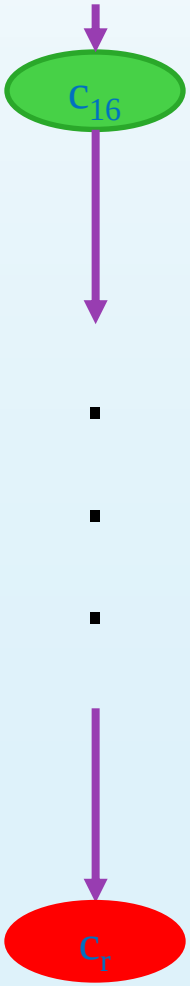
Παράδειγμα – Υπολογισμός

Διαμόρφωση μετά το βήμα 2: $c_0=q_0$a,b,c,d,ab,bd,dc,cb,3$

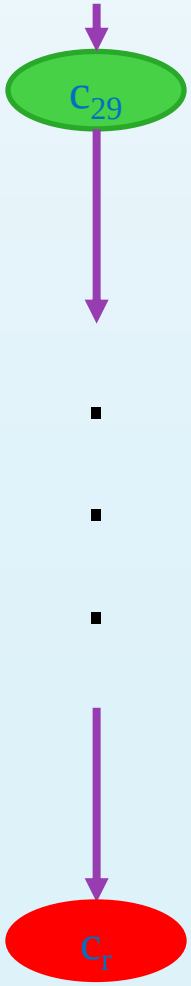


Παράδειγμα – Υπολογισμός

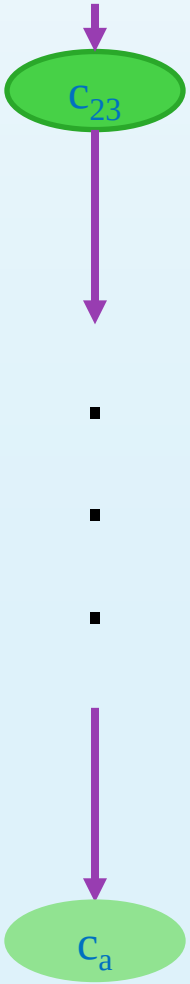
c_{16}=a,b,c,d,ab,bd,dc,cb,3,$\$q_j$a,b,c



c_{29}=a,b,c,d,ab,bd,dc,cb,3,$\$q_j$d

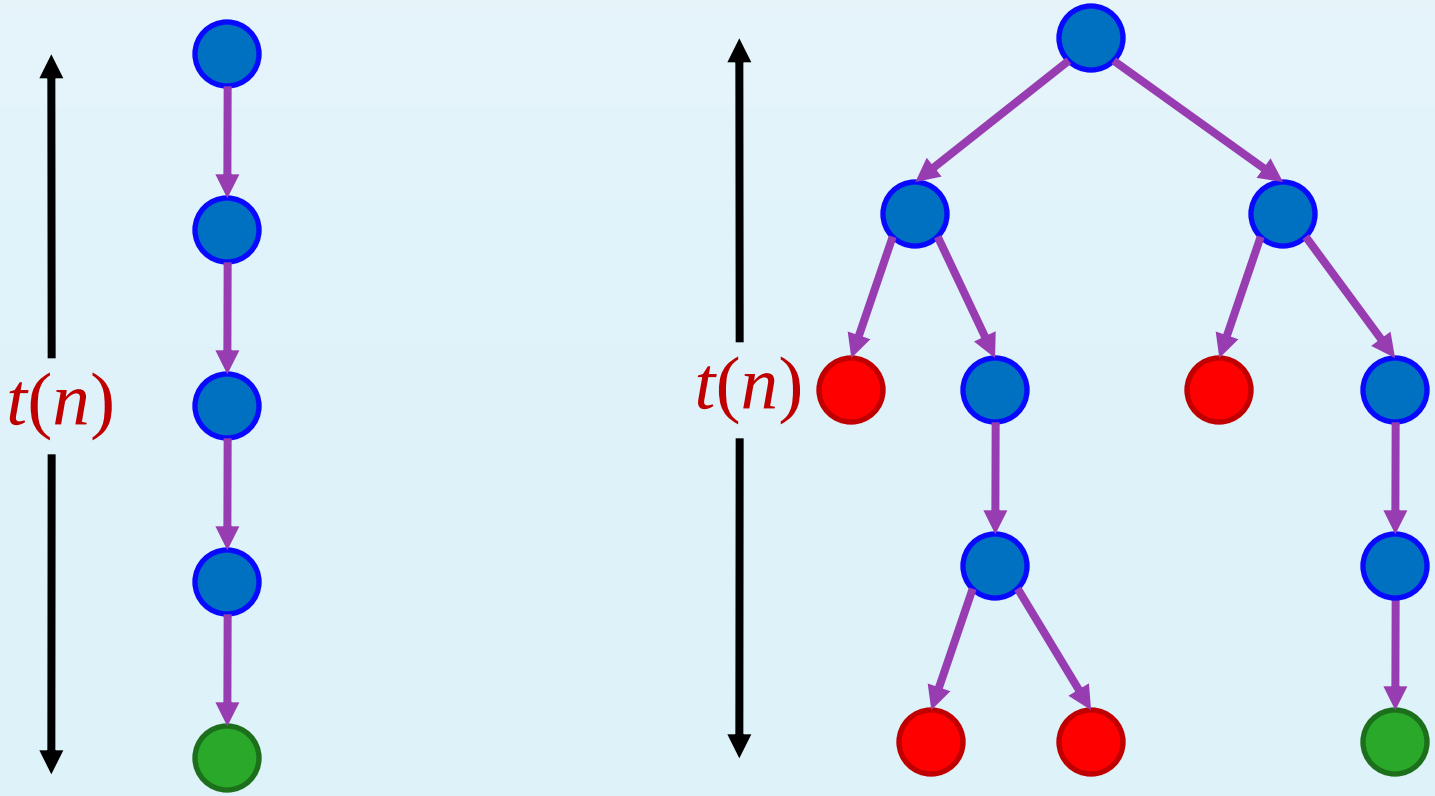


c_{23}=a,b,c,d,ab,bd,dc,cb,3,$\$q_j$b,c,d



Αιτιοκρατικός vs Ανταιτιοκρατικός

Προσέξτε ότι απορριπτικά μονοπάτια απορρίπτουν μέσα σε το πολύ $t(n)$ βήματα.



Πολυπλοκότητα Ανταιτιοκρατικού Μοντέλου

Ισχυρισμός: Έστω N μία ανταιτιοκρατική ΤΜ που τρέχει σε χρόνο $t(n)$ και διαγιγνώσκει την γλώσσα L.

Τότε: Υπάρχει μία αιτιοκρατική ΤΜ, D, που σε $2^{O(t(n))}$ βήματα διαγιγνώσκει την L.

Προσέξτε τη διαφορά με την πολυταινιακή ΤΜ.

Εξομοίωση

Έστω N η ανταιτιοκρατική ΤΜ που έχει χρόνο εκτέλεσης $t(n)$. Θέλουμε να περιγράψουμε την αιτιοκρατική ΤΜ D, που εξομοιώνει την N.

Βασική ιδέα εξομοίωσης:

- Η D δοκιμάζει όλους τους δυνατούς κλάδους.
- Αν η D βρει μία κατάσταση αποδοχής τότε αποδέχεται.
- Αν όλοι οι κλάδοι απορρίπτουν, τότε η D απορρίπτει.
- Προσοχή ότι η N ***δεν εγκλωβίζεται***, και άρα ένα από τα δύο θα συμβεί οπωσδήποτε.

Λεπτομέρειες Εξομοίωσης

Ο υπολογισμός της N είναι δένδρο:

- Η ρίζα είναι η εναρκτήρια διαμόρφωση,
- Κάθε κόμβος έχει φραγμένο πλήθος παιδιών $\leq b$ (γιατί;),
- Κάθε κλάδος έχει μήκος $\leq t(n)$,
- Συνολικό πλήθος φύλλων είναι το πολύ $b^{t(n)}$,
- Συνολικό πλήθος κόμβων στο δένδρο είναι $O(b^{t(n)})$,
- Χρόνος για να φτάσεις από τη ρίζα σε οποιονδήποτε κόμβο είναι $O(t(n))$.

Άρα ο συνολικός χρόνος για επίσκεψη σε όλους τους κόμβους είναι:

$$O(t(n)\times b^{t(n)})=O(2^{O(t(n))})$$

Παρατηρήσεις

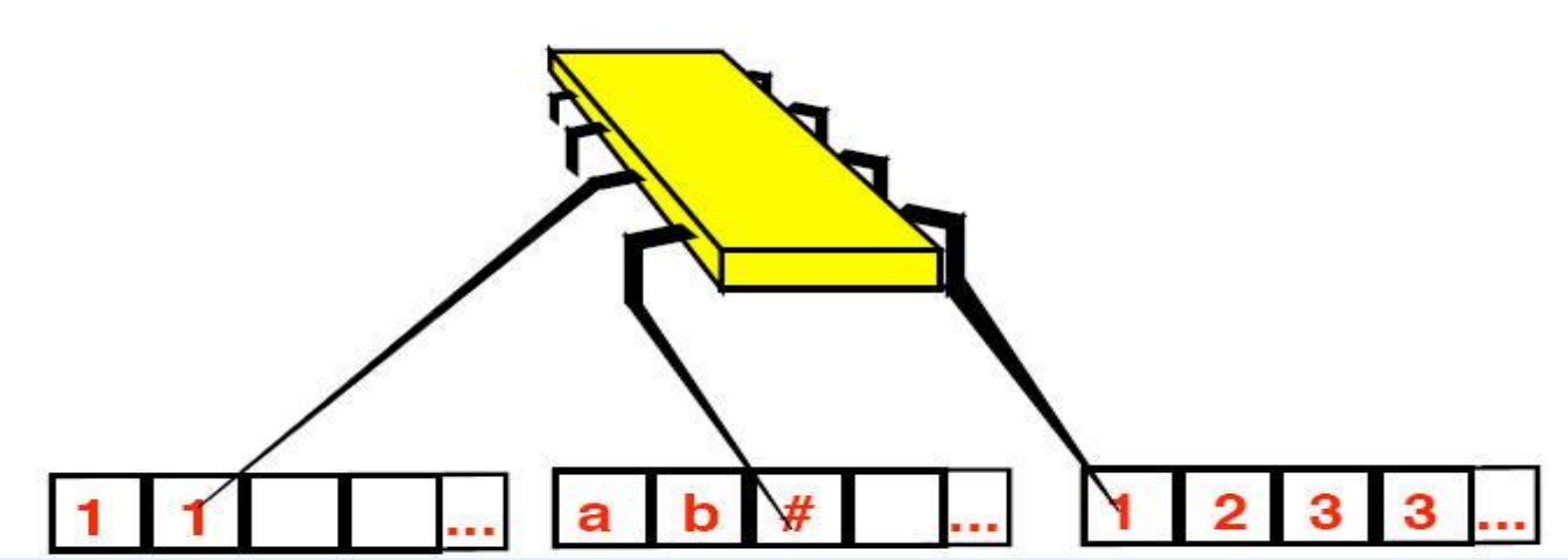
Διαπέραση κατά πλάτος χρησιμοποιείται στο δένδρο υπολογισμού της N:

- Μη αποδοτική διαπέραση από τη ρίζα προς κάθε κόμβο.
- Μπορούμε να το βελτιώσουμε χρησιμοποιώντας άλλες διαπεράσεις, όπως διαπέραση κατά βάθος (γιατί δουλεύει;)
- Όμως ακόμα και αυτό δεν μειώνει τον εκθετικό χρόνο παρά κάποιες σταθερές

Μονοταινιακή Εξομοίωση

- Η εξομοίωση χρησιμοποιεί τρεις ταινίες.
- Εξομοίωση με μία ταινία:

$$(2^{O(t(n))})^2 = 2^{O(2t(n))} = 2^{O(t(n))}$$



Θεμελιώδης Διαφορές

- Το πολυωνυμικό χάσμα στον χρόνο μεταξύ διαφορετικών αιτιοκρατικών μοντέλων (πολυταινιακές, RAM, δισδιάστατες, κτλ.)

- σε σύγκριση με

- το εκθετικό χάσμα στο χρόνο που υπάρχει μεταξύ αιτιοκρατικών και ανταιτιοκρατικών ΤΜ μοντέλων.

Η RAM

Η RAM μοντελοποιεί καλύτερα τους σύγχρονους Η/Υ

- Η μνήμη αποτελείται από «άπειρους» καταχωρητές x[1],x[2],…
- Το μέγεθος των αριθμών είναι άπειρο σε κάθε καταχωρήτη

Προσοχή!!!!!

- Άμεση προσπέλαση σε κάθε καταχωρητή
  - Μέσω άλλου καταχωρητή που κρατά τη διεύθυνσή του
- Η αποθήκευση του προγράμματος γίνεται σε άλλη μνήμη του παραπάνω τύπου (καταχωρητής = γραμμή).

Χρόνος Εκτέλεσης σε RAM

Τι μετράμε;

- Αν μετράμε πλήθος βημάτων τότε μπορούμε να κλέψουμε αφού έχουμε απείρως μεγάλους αριθμούς

Σύνολο εντολών:

x[*i*]=0	x[*i*]++	x[*i*]--
x[*i*]=x[*i*]+x[*j*]	x[*i*]=x[*i*]-x[*j*]	
x[*i*]=x[x[*j*]]	x[x[*i*]]=x[*j*]	

if x[*i*]≤0 ***then goto*** *p*;

Μετράμε το πλήθος των bits που λαμβάνουν μέρος σε κάθε βήμα του υπολογισμού (σε κάθε εντολή). (***Λογαριθμικού κόστους RAM***)

Παράδειγμα: Αν ένα πρόγραμμα κάνει n βήματα σε αριθμούς με το πολύ k δυαδικά ψηφία τότε η πολυπλοκότητά του είναι $O(nk)$.

Εξομοίωση RAM

Ισχυρισμός: Έστω R μία RAM που τρέχει σε χρόνο $t(n)$ και διαγιγνώσκει την γλώσσα L.

Τότε: Υπάρχει μία πολυταινιακή αιτιοκρατική ΤΜ, D, που σε $O(t^2(n))$ βήματα διαγιγνώσκει την L.

Θα χρησιμοποιήσουμε μία ΤΜ με 4 ταινίες.

Εξομοίωση

- Η 1η ταινία κρατά το περιεχόμενο των καταχωρητών x[*i*]. Κρατά μόνο αυτούς που χρησιμοποιούνται και γράφονται από την RAM. Η εγγραφή του *y* στο x[*z*] αναπαρίσταται ως ##*y*#*z*. Η ανάγνωση του x[*z*] γίνεται με ψάξιμο στην ταινία για το ##*y*#*z*.
- Κάθε μία από τις πράξεις μπορεί να εξομοιωθεί με 3 ταινίες.
- Μετά το πέρας της εξομοίωσης κάθε πράξης η κεφαλή της 1ης ταινίας πάει στο τέλος ενώ οι υπόλοιπες στην αρχή τους.
- Για κάθε μία πράξη (εκτός από ανάγνωση μνήμης) αυτή μπορεί να γίνει με τις 3 ταινίες σε χρόνο ίσο με το πλήθος των bits. Η ανάγνωση απαιτεί ψάξιμο στην 1η ταινία σε χρόνο $O(t(n))$ (Γιατί;).
- Άρα εξομοίωση σε $O(t^2(n))$.

Το Θεώρημα της Γραμμικής Επιτάχυνσης

Γιατί ο ασυμπτωτικός συμβολισμός O δεν μας πειράζει ιδιαίτερα;

Για κάθε ΤΜ T που επιλύει τη γλώσσα L σε $f(n)$ βήματα και για κάθε σταθερά $\varepsilon>0$, υπάρχει μία ΤΜ S που επιλύει την L σε $\varepsilon f(n)+n+2$.

Γραμμική Επιτάχυνση

Θεώρημα: Έστω η ΤΜ M που διαγιγνώσκει τη γλώσσα L σε χρόνο $f(n)$. Τότε για κάθε $\epsilon > 0$, υπάρχει ΤΜ M' που διαγιγνώσκει την L σε χρόνο

$$\epsilon f(n) + n + 2.$$

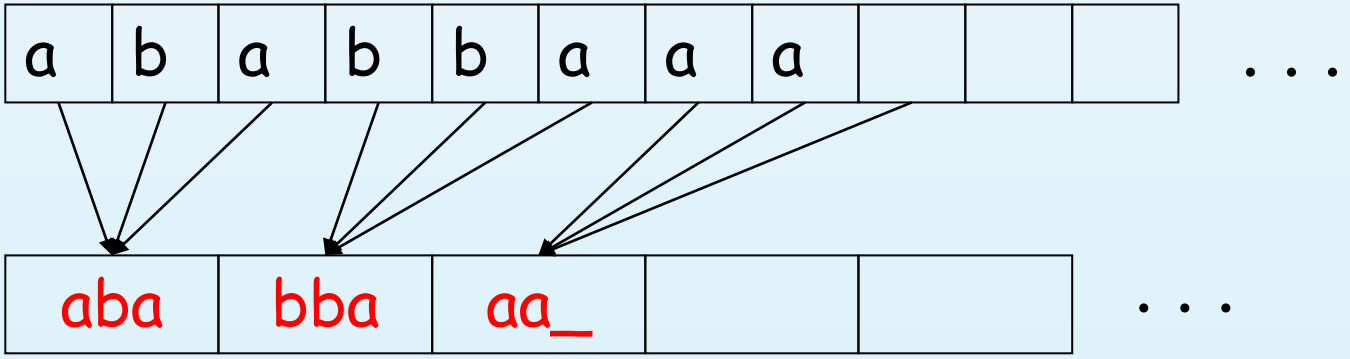
- Απόδειξη:
  - Ιδέα: αύξηση «μήκους λέξης»
  - Η M' θα έχει
    - μία επιπλέον ταινία της M
    - νέο αλφάβητο βασισμένο στο αλφάβητο του M

$$\Sigma_{new} = \Sigma_{old} \cup \Sigma_{old}^{m}$$

    - πολλές επιπλέον καταστάσεις

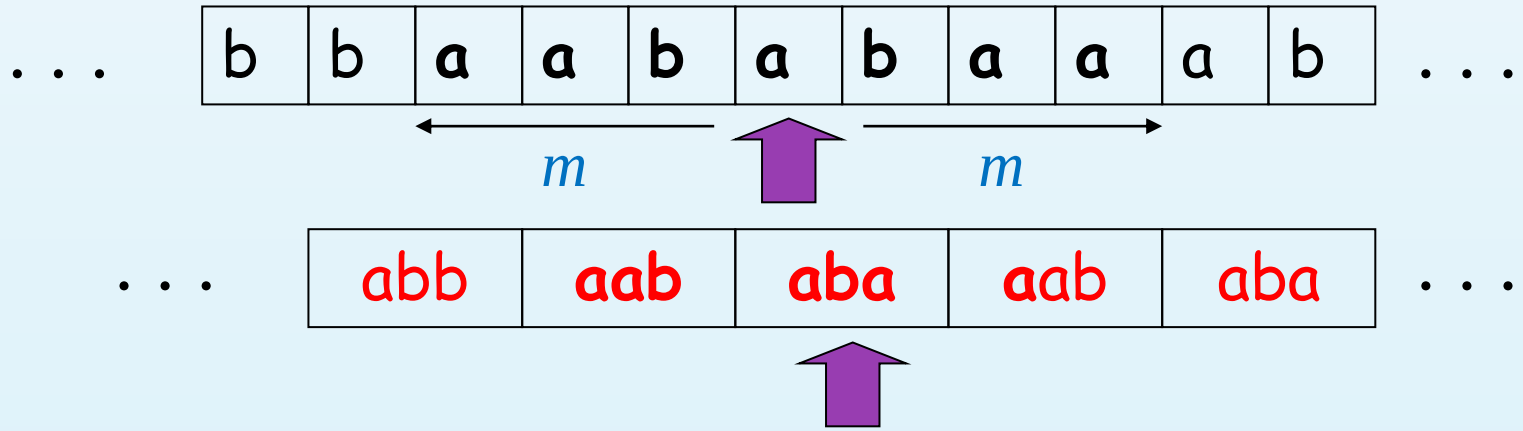
Γραμμική Επιτάχυνση

- Συμπίεση εισόδου στη νέα ταινία



Γραμμική Επιτάχυνση

- Εξομοίωση M, κατά m βήματα κάθε φορά



  - 4 (Α,Δ,Δ,Α) βήματα για ανάγνωση συμβόλων, κατάλληλη κατάσταση
  - 2 (Α,Δ ή Δ,Α) για να γίνουν οι αλλαγές στην M

Γραμμική Επιτάχυνση

- Σύνολο:
  - Αντιγραφή : $n + 2$ βήματα
  - (εξομοίωση): $6\,(f(n)/m)$
  - Θέτουμε $m = 6/\epsilon$
  - Σύνολο: $\epsilon f(n) + n + 2$

Θεώρημα: Έστω ΤΜ M που διαγιγνώσκει τη γλώσσα L σε χώρο $f(n)$. Τότε για κάθε $\epsilon > 0$, υπάρχει ΤΜ M' που διαγιγνώσκει την L σε χώρο $\epsilon f(n) + 2$.

- Απόδειξη: Αντίστοιχη.