Εισαγωγή στην Επιστήμη των Υπολογιστών

4ο εξάμηνο ΣΕΜΦΕ

http://www.corelab.ece.ntua.gr/courses/

1η ενότητα: Εισαγωγή, Αλγόριθμοι

Διδάσκοντες

Στάθης Ζάχος, Άρης Παγουρτζής, Κλειώ Σγουροπούλου

Βοηθός διδασκαλίας: Βαγγέλης Μπαμπάς

Ευχαριστίες: Ηλίας Κουτσουπιάς (ΕΚΠΑ)

Περιεχόμενα

Εισαγωγή

- Κλάδοι της Επιστήμης των Υπολογιστών
- Τι είναι Επιστήμη των Υπολογιστών;
- Πρόγραμμα και γλώσσα προγραμματισμού
- Αλγόριθμος
- Υπολογισιμότητα
- Πολυπλοκότητα
- Επανάληψη, επαγωγή, αναδρομή, ορθότητα
- Δομημένος προγραμματισμός
- Παράλληλες, ταυτόχρονες, κατανεμημένες διεργασίες
- Παραδείγματα: πύργοι Hanoi, ταξινόμηση treesort με binary search tree
- Το Θεώρημα τεσσάρων χρωμάτων (four color theorem)

Εισαγωγή

- Ο 21ος αιώνας έχει ονομαστεί «ο αιώνας των υπολογιστών». Οι υπολογιστές δεν είναι πλέον μόνο επαγγελματικά εργαλεία αλλά και μέσα επικοινωνίας και ψυχαγωγίας.
- Οι νέοι σπουδαστές έχουν ήδη μεγάλη εμπειρία στη χρήση υπολογιστών και του Internet. Θα μπορούσε κανείς να χρησιμοποιήσει τη φυσική γοητεία των υπολογιστών για να τονώσει το ενδιαφέρον των σπουδαστών για τα μαθηματικά και τις επιστήμες.
- Η **Αλγοριθμική** (μεθοδολογία) είναι ένας νέος τρόπος επίλυσης προβλημάτων.
- Μια πρόταση για τη μέση εκπαίδευση (και όχι μόνο): αντί να μαθαίνουμε τεχνολογικές δεξιότητες σχετικά με τους υπολογιστές, ας μαθαίνουμε μια υποτυπώδη γλώσσα προγραμματισμού πράγμα που θα ενθαρρύνει την επικέντρωση στο σχεδιασμό και στην ακριβή διατύπωση λύσεων για προβλήματα.

Επιτυχίες της Αλγοριθμικής

- Fast Fourier Transform
  [Cooley-Tukey, 1965 (αλλά και Gauss, 1805)].

- Κρυπτογραφία δημοσίου κλειδιού [Diffie-Hellman, Rivest-Shamir-Adleman, 1976-77].

- Pagerank (Google)
  [Page-Brin-Motwani-Winograd, 1995-99].

Κεντρικό ερώτημα Επιστήμης Υπολογιστών

Τι μπορεί να μηχανοποιηθεί και μάλιστα αποδοτικά ;

Ποια προβλήματα μπορούμε να λύσουμε με υπολογιστή και πόσο καλά ;

Υπολογιστές: ταχύτητα – ακρίβεια

Τι είναι η Επιστήμη των Υπολογιστών;

- Ο επιστημονικός και τεχνολογικός κλάδος που μελετάει την αναπαράσταση, αποθήκευση, επεξεργασία και μετάδοση πληροφοριών μέσω υπολογιστών και δικτύων.
- Επίσης μελετάει αλγόριθμους (για επίλυση προβλημάτων), την αποδοτικότητά τους, και γενικά υπολογιστική πολυπλοκότητα.

Computer Science

Informatics

Computing Science

Dijkstra (astronomy, telescope)

Κλάδοι Επιστήμης Υπολογιστών (i)

- Αλγόριθμοι και δομές δεδομένων
- Γλώσσες προγραμματισμού και μεταγλωττιστές
- Μεθοδολογία - Τεχνολογία Λογισμικού (software)
- Αρχιτεκτονική υπολογιστών και δικτύων (hardware)
- Αριθμητικοί και συμβολικοί υπολογισμοί
- Λειτουργικά - Παράλληλα - Κατανεμημένα Συστήματα

Κλάδοι Επιστήμης Υπολογιστών (ii)

- Βάσεις Δεδομένων και Διαχείριση Πληροφοριών
- Τεχνητή Νοημοσύνη και Ρομποτική
- Επικοινωνία ανθρώπου - υπολογιστή. Πολυμέσα
- Δίκτυα Επικοινωνιών - Ευφυή Δίκτυα - Διαδίκτυο
- Κρυπτογραφία
- Υπολογιστική βιολογία, βιολογικοί υπολογισμοί
- Κβαντικοί υπολογισμοί

Άλλες όψεις της Επιστήμης των Υπολογιστών

- Networks and Internet.
- Information flow (ordering Chinese food, movie theaters, holidays in Togo, poets of Sakhalin, Pursuit of trivia).
- Business applications (online financial transactions, e-commerce, e-banking, e-anything).
- Modern technology (AI, multimedia, ...).
- Administration (big brother, encryption, security, e-voting, digital warfare).
- Change in every aspect of everyday life.
- Futurology ....

Μια νέα μαθηματική μεθοδολογία: αλγοριθμική επίλυση προβλημάτων

Αλγοριθμικά προβλήματα εμφανίζονται σε:

- Internet (δρομολόγηση, συμφόρηση, θεωρία παιγνίων, ανάθεση κόστους)
- Βιολογία (αναδίπλωση πρωτεϊνών, γονιδίωμα, εξέλιξη)
- Κρυπτογραφία (ασφάλεια, μυστικότητα, ηλεκτρονικές υπογραφές, ψηφοφορίες)

Είπαν...

Dijkstra:

- Το ερώτημα εάν ο **υπολογιστής σκέφτεται** δεν είναι πιο ενδιαφέρον από το ερώτημα εάν το **υποβρύχιο κολυμπάει**.

- Η επιστήμη των υπολογιστών έχει αντικείμενο τους υπολογιστές όσο και η αστρονομία τα τηλεσκόπια.

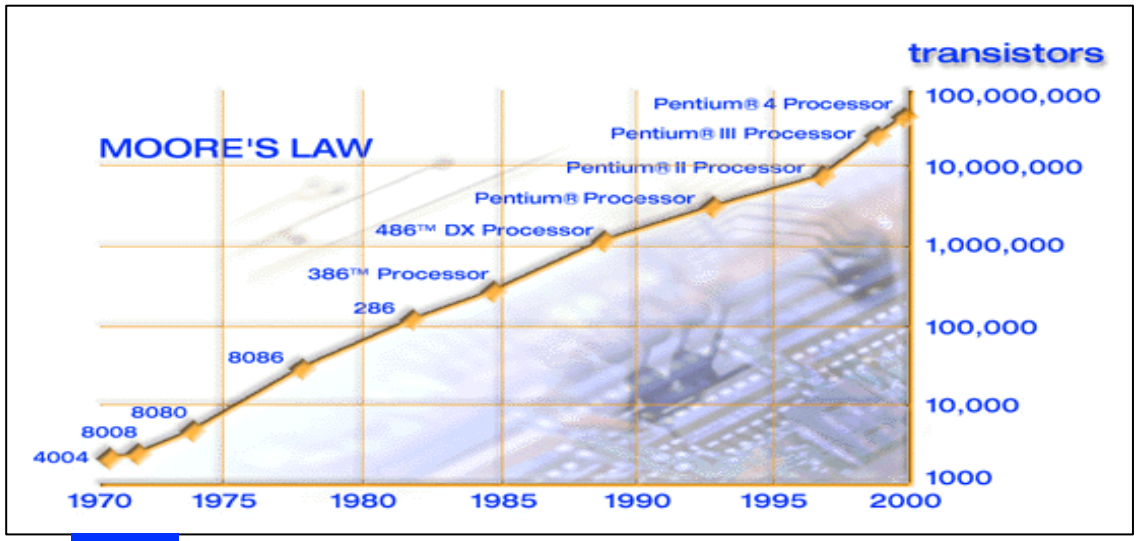
Tom Watson, IBM, 1945:

- *Ο κόσμος δεν χρειάζεται περισσότερους από πέντε υπολογιστές*

Hardware exponential growth

Gordon Moore, Intel, 1965:

- *Hardware density in integrated circuits doubles every 18 months*



© intel. http://www.intel.com/research/silicon/mooreslaw.htm

Αλλά ας μην ξεχνάμε ότι:

To err is human.

To mess everything up, you need a computer!

Μαθηματική διατύπωση της μεθοδολογίας των μηχανικών στην επιστήμη υπολογιστών

- Απαιτήσεις προβλήματος (Requirements)
- Προδιαγραφές (Specifications)
- Σχεδιασμός (Design)
- Υλοποίηση (Implementation)
- Δοκιμές – Επαλήθευση (Testing – Verification)
- Βελτιστοποίηση (Optimization)
- Πολυπλοκότητα [κόστος πόρων] (Complexity)
- Τεκμηρίωση (Documentation)
- Συντήρηση (Maintenance)

Έννοιες που χρησιμοποιούνταν από μηχανικούς απέκτησαν μαθηματική διατύπωση και αυστηρές αποδείξεις!

Πρόγραμμα

- Ακριβής περιγραφή ενός αλγορίθμου σε μία τυπική γλώσσα που ονομάζεται γλώσσα προγραμματισμού.
- Οι ενέργειες εφαρμόζονται σε αντικείμενα που λέγονται δεδομένα (data).

Διατύπωση αλγορίθμου σε γλώσσα

- Φυσικές γλώσσες
  - Χωρίς αυστηρούς συντακτικούς κανόνες
  - Μεγάλη πυκνότητα και σημασιολογική ικανότητα
- Τυπικές γλώσσες
  - Αυστηρό συντακτικό και σημασιολογία
- Γλώσσες προγραμματισμού
  - Τυπικές γλώσσες περιγραφής υπολογισμών
  - Εντολές εκτελέσιμες από ηλεκτρονικό υπολογιστή

Θεωρητική θεμελίωση

- Αναπαράσταση γνώσης
- Μοντελοποίηση
- Αφαίρεση:
  - (Κατευθυνόμενοι) Γράφοι
  - (Τυπική) Λογική
- Εργαλεία:
  - Μοντέλα δεδομένων
  - Δομές δεδομένων
  - Αλγόριθμοι

Αλγόριθμοι

(not in Webster's 50 years ago, 1971 in Oxford's: erroneous refashioning of algorism: calculation with Arabic numerals)

- Abu Jaffar Mohammed Ibn Musa Al-Khowarizmi, محمد بن موسى الخوارزمي, 9ος αι. μ.Χ.
  - Αυστηρά καθορισμένη (πεπερασμένη) ακολουθία ενεργειών που περιγράφει μέθοδο επίλυσης ενός προβλήματος
  - Οι ενέργειες εφαρμόζονται σε δεδομένα
- Παραδείγματα:
  - Ευκλείδειος αλγόριθμος (Ευκλείδης, 3ος αι. π.Χ.) για εύρεση ΜΚΔ
  - Αριθμοί Fibonacci (Leonardo Pisano Filius Bonacci, 13ος αι. μ.Χ.)
  - Τρίγωνο Pascal (Yang Hui, 楊輝, 13ος αι. μ.Χ.)

Αριθμοί Fibonacci

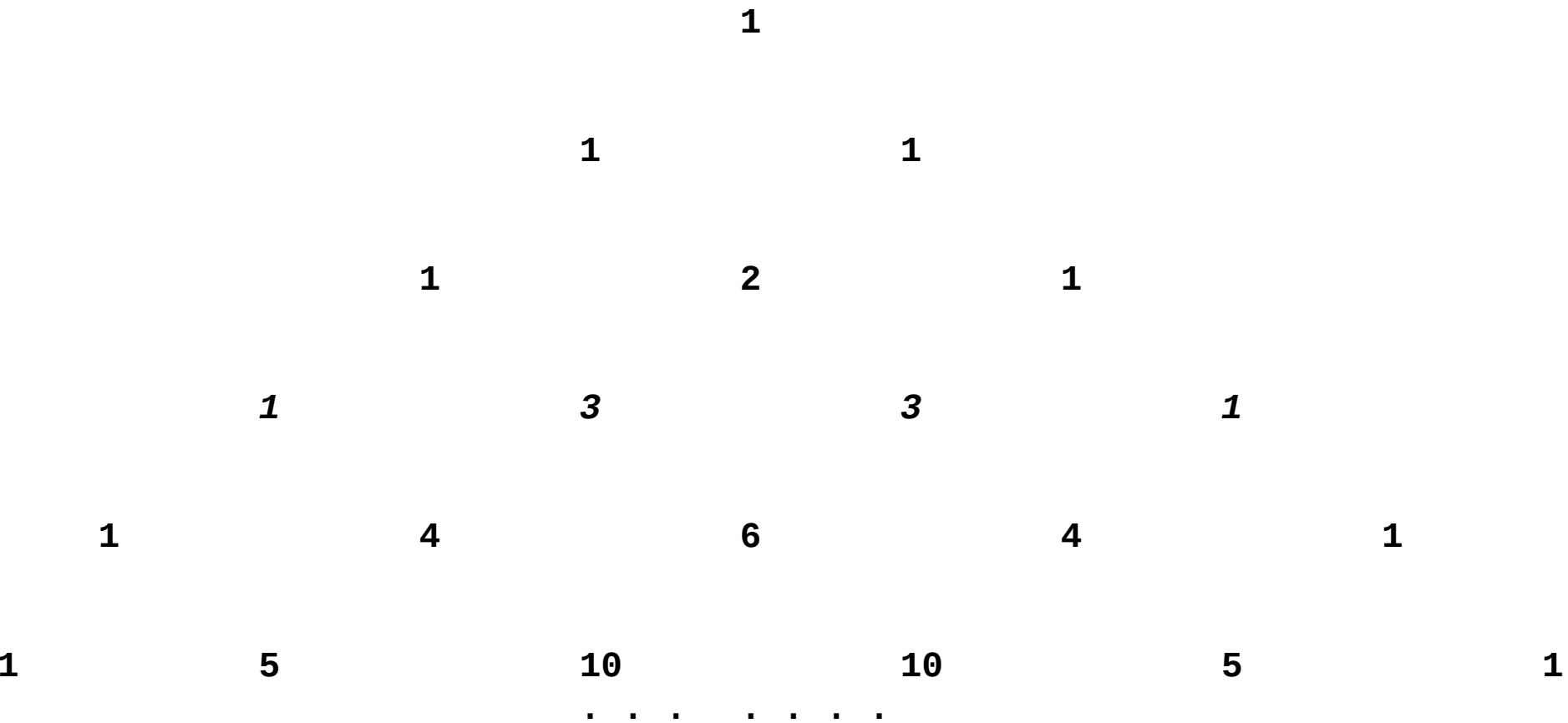
- 1, 1, 2, 3, 5, 8, 13, 21,…

$$F_n = F_{n-1} + F_{n-2}$$

- Πρόβλημα: δίνεται n, υπολόγισε τον F_n.
- Recursion (αναδρομή), iteration (επανάληψη), …
- Πόσο γρήγορα μπορεί να υπολογιστεί ο F_n;

  $O(1.618^n)$, $O(n)$, $O(\log n)$

Τρίγωνο Pascal (Yang Hui)



Διωνυμικοί συντελεστές / Συνδυασμοί :

$$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$

Η Αλγοριθμική ως κλάδος των Μαθηματικών

- "Τα μαθηματικά είναι μια μεθοδολογία επίλυσης προβλημάτων". G. Polya: How to solve it.
- Στην κλασσική προσέγγιση η λύση ενός προβλήματος περιγράφεται στατικά, π.χ. σαν λύση μιας εξίσωσης.
- Στα μοντέρνα μαθηματικά του 20ου αιώνα η λύση μπορεί να είναι δυναμική, π.χ. η περιγραφή ενός αλγορίθμου που υπολογίζει την απάντηση.
- Επιπλέον, το κόστος της αλγοριθμικής επίλυσης ενός προβλήματος ενδιαφέρει: Υπολογιστική Πολυπλοκότητα.
- D. Harel, W. Feldman. Algorithmics: the spirit of computing

Υπολογισιμότητα - Πολυπλοκότητα

- Υπολογισιμότητα (Computability)
  - Τι μπορεί να υπολογιστεί και τι όχι;
- Υπολογιστική πολυπλοκότητα (Computational Complexity)
  - Τι μπορεί να υπολογιστεί γρήγορα (ή σε λίγο χώρο) και τι όχι;
  - Πόσο αποδοτικά μπορεί να υπολογιστεί;

Μη επιλύσιμα προβλήματα: Πρόβλημα τερματισμού

- Το πρόβλημα του Collatz (Ulam):

```
while x!=1 do
  if (x is even) then x=x/2
  else x=3*x+1
```

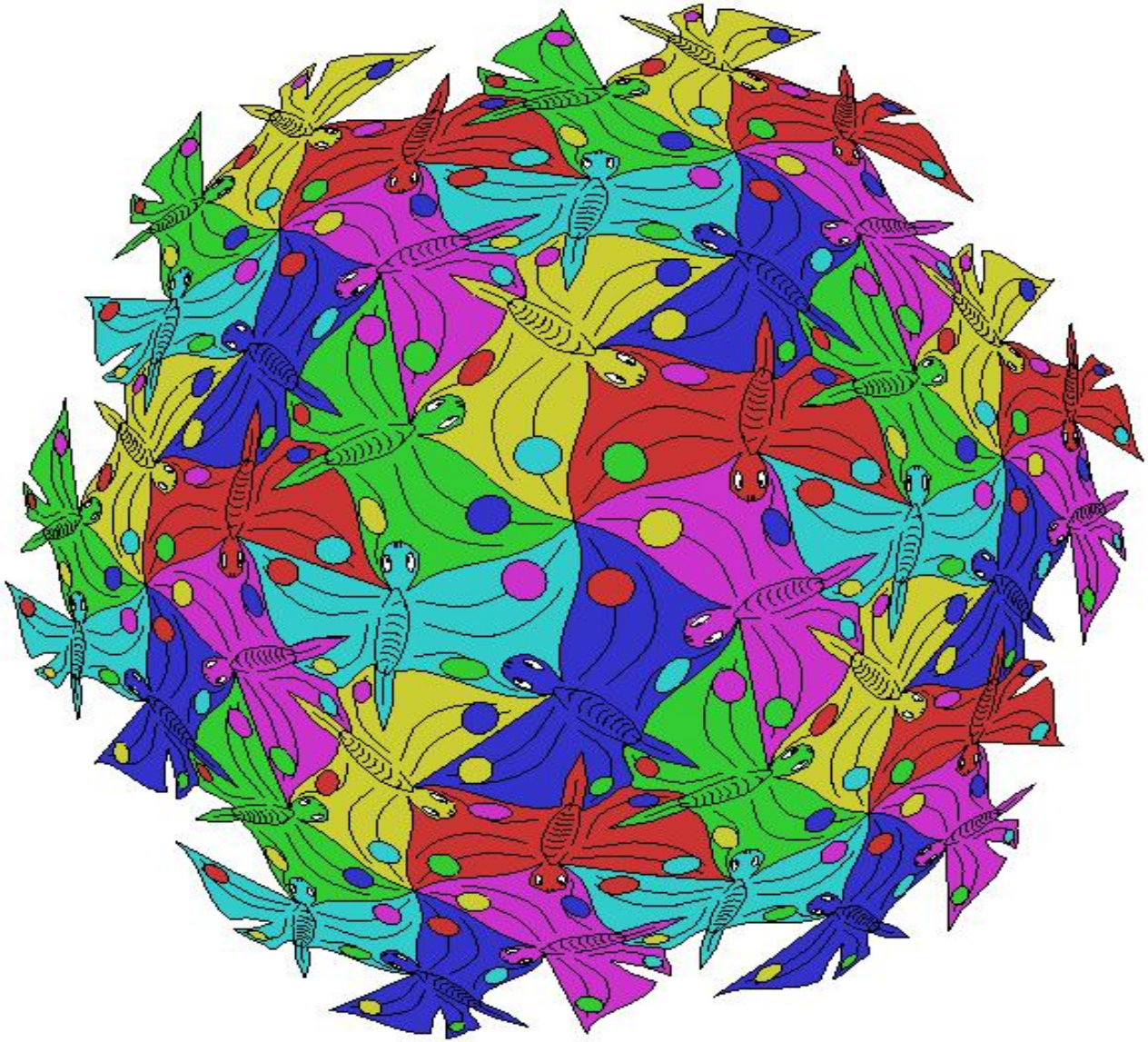
- Παράδειγμα: 7 → 22 → 11 → 34 → 17 → 52 → 26 → 13 → 40 → 20 → 10 → 5 → 16 → 8 → 4 → 2 → 1
- Πρόβλημα: Δίνεται x. Τερματίζει το πρόγραμμα;
- Πρόβλημα: Τερματίζει το πρόγραμμα για κάθε φυσικό αριθμό x;
- Δεν γνωρίζουμε την απάντηση (είναι δηλαδή ανοικτά προβλήματα).

Μη επιλύσιμα προβλήματα: Πρόβλημα τερματισμού

- Το 3x+1 είναι ειδική περίπτωση του προβλήματος τερματισμού:
  Δίνεται πρόγραμμα και είσοδος. Τερματίζει το πρόγραμμα για αυτή την είσοδο;
  Μια ισοδύναμη παραλλαγή είναι:
  Δίνεται πρόγραμμα χωρίς είσοδο. Τερματίζει;
- Θεώρημα: Το πρόβλημα τερματισμού είναι *μη επιλύσιμο*. Δηλαδή, δεν υπάρχει αλγόριθμος που να απαντάει σε αυτή την ερώτηση.

Τι είναι πολυπλοκότητα;

- Το 101101011101 είναι πιο πολύπλοκο από το 010101010101 (Kolmogorov complexity)
- Τα θηλαστικά είναι πιο πολύπλοκα από τους ιούς.
- Το σκάκι είναι πιο πολύπλοκο από την τρίλιζα.
- Οι επικαλύψεις του Escher είναι πιο πολύπλοκες από τα πλακάκια του μπάνιου.
- Οι πρώτοι αριθμοί είναι πιο πολύπλοκοι από τους περιττούς (υπολογιστική πολυπλοκότητα).



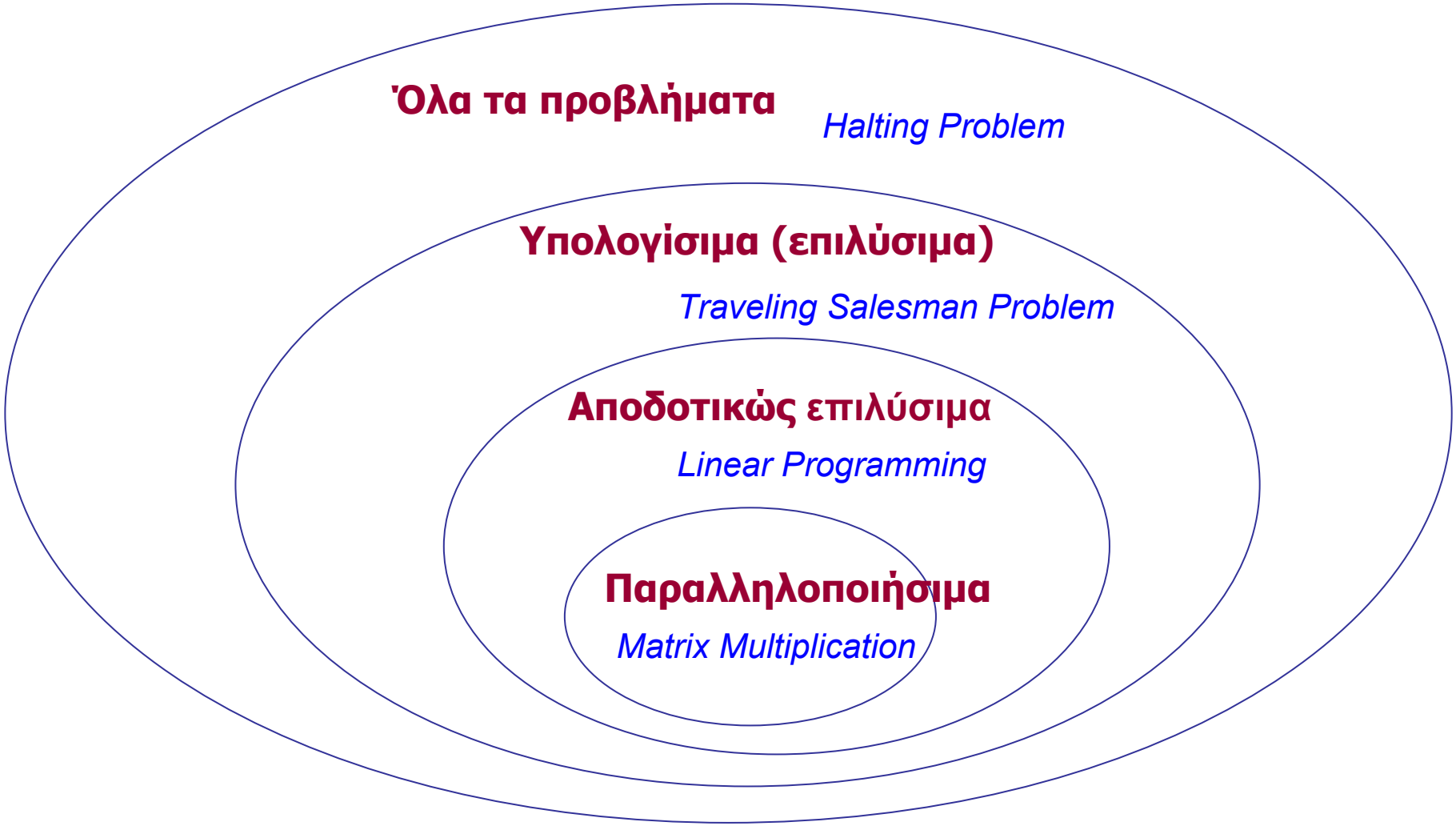
Τι είναι υπολογιστική πολυπλοκότητα;

- Ένας τρόπος για να συλλάβουμε γιατί οι πρώτοι αριθμοί είναι πιο πολύπλοκοι από τους περιττούς είναι η υπολογιστική πολυπλοκότητα.
- Το πρόβλημα «Δίνεται x. Είναι πρώτος;» είναι πιο δύσκολο από το πρόβλημα «Δίνεται x. Είναι περιττός;»

Υπολογιστική Πολυπλοκότητα

- Η έννοια της αποδοτικής (εφικτής) επίλυσης προβλημάτων.
- Σύγκριση αλγορίθμων ως προς την αποδοτικότητά τους.
- Ταξινόμηση προβλημάτων ως προς τη χρήση πόρων, π.χ. χρόνου (υπολογιστικών βημάτων), χώρου (μνήμης), πλήθους επεξεργαστών, πλήθους μηνυμάτων επικοινωνίας, χρήση τυχαίων bit, χρήση εξωτερικής βοήθειας («μαντείου»), κ.λπ.

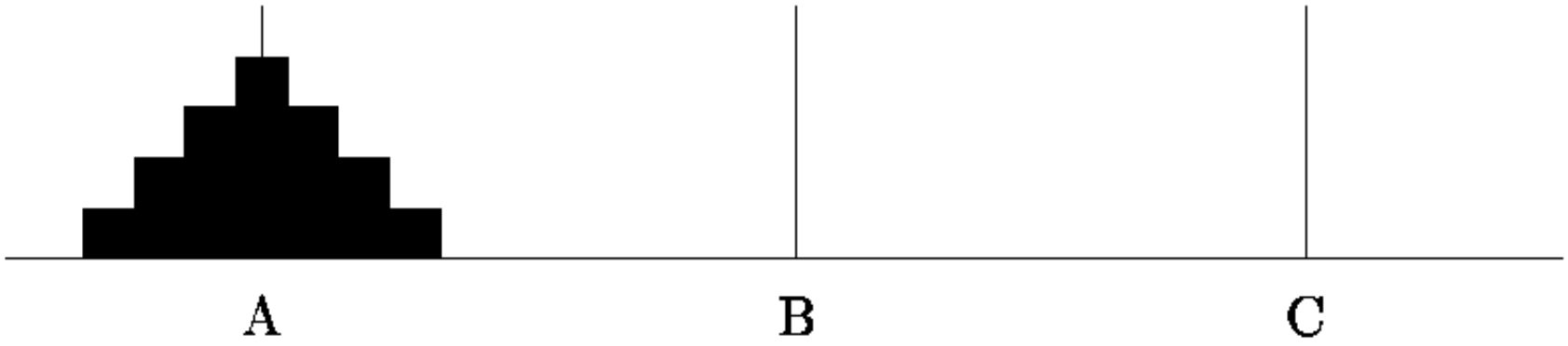
Κατηγοριοποίηση προβλημάτων



Ασυμπτωτική συμπεριφορά

$\log n$	n	n^2	2^n
3.322	10	100	1024
6.644	100	10000	1267650600228229401496703205376
9.966	1000	1000000	$(1267650600228229401496703205376)^{10}$

Επανάληψη-Αναδρομή-Επαγωγή (Iteration-Recursion-Induction)

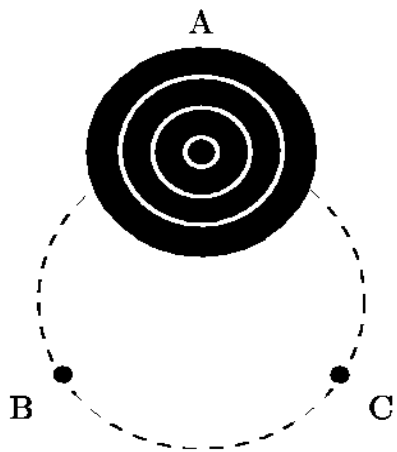


Σχήμα 1.2: Πύργοι του Ανόι ($n = 4$).

Πύργοι Ανόι (Hanoi Towers)

```
procedure move_anoi(n from X to Y using Z)
begin
    if n = 1 then move top disk from X to Y
    else begin
            move_anoi(n−1 from X to Z using Y);
            move top disk from X to Y;
            move_anoi(n−1 from Z to Y using X)
         end
end
```

1. *μετακίνησε κατά την θετική φορά τον μικρότερο δίσκο·*
2. *κάνε την μοναδική επιτρεπτή κίνηση που δεν αφορά τον μικρότερο δίσκο.*



Μερική και Ολική Ορθότητα

- Λειτουργική σημασιολογία (operational semantics): περιγράφει την υπολογιστική ακολουθία που εκτελείται.
- Δηλωτική σημασιολογία (denotational semantics): ορίζει μόνο τη συνάρτηση εισόδου-εξόδου.
- Αξιωματική σημασιολογία (axiomatic semantics): περιγράφει τις σχετικές ιδιότητες που πρέπει απαραίτητα να ικανοποιούνται από την είσοδο και την έξοδο.
  - Αλγεβρική σημασιολογία (algebraic semantics): χρήση άλλων αλγεβρικών δομών αντί κατηγορηματικού λογισμού και φυσικών αριθμών.

Η ιδέα του Ευκλείδη για εύρεση ΜΚΔ δύο φυσικών αριθμών

```
if a>b then GCD(a,b):= GCD(a mod b, b)
       else GCD(a,b):= GCD(a, b mod a)
```

a mod b =
το υπόλοιπο της ακέραιας διαίρεσης a div b

Ο Ευκλείδειος αλγόριθμος είναι ο καλύτερος γνωστός αλγόριθμος για ΜΚΔ!

Ανοιχτό πρόβλημα: είναι βέλτιστος;

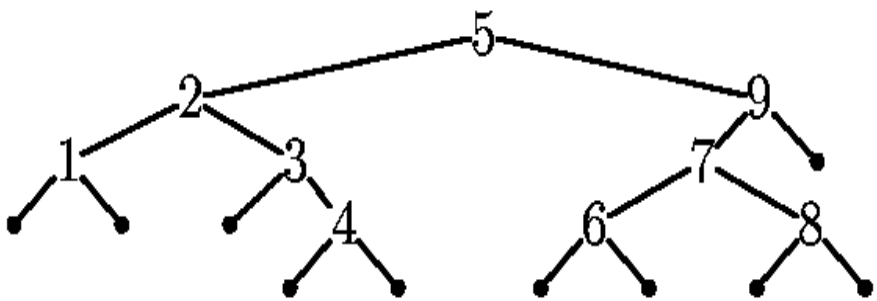
Παράδειγμα εκτέλεσης του Ευκλείδειου αλγόριθμου

ΜΚΔ των 172 και 54 (επανάληψη)

10	54
10	4
2	4
2	0

ΜΚΔ = 2

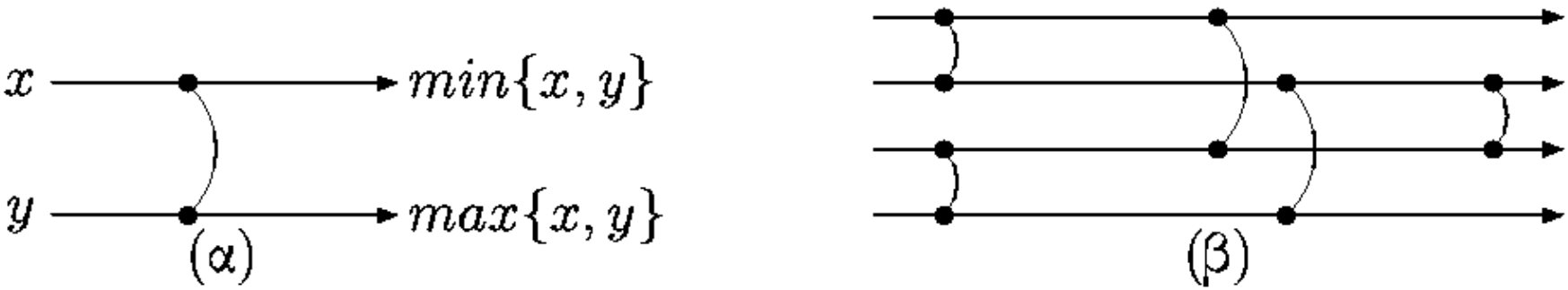
Treesort με χρήση Binary Search Tree




```
procedure inorder(t: treenode)
begin
    if t is not empty then
    begin
        inorder(left branch of t);
        write(element at t);
        inorder(right branch of t)
    end
end
```

- Δομημένος προγραμματισμός
  - Modularity
- Παραλληλία
  - Parallel Systems
  - Concurrent Systems
  - Distributed Systems

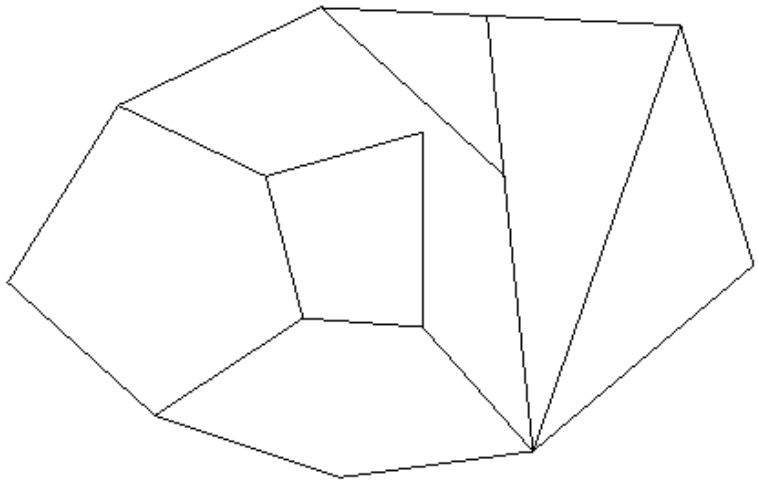
Δίκτυα Ταξινόμησης (Sorting Networks)



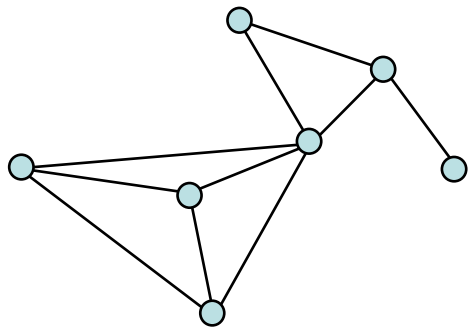
Σχήμα 1.4: (α) Συγκριτής (β) Δίκτυο ταξινόμησης 4 εισόδων

Four Color Theorem (1852-1977)

Πόσα χρώματα απαιτούνται για τον χρωματισμό όλων των χωρών, ούτως ώστε χώρες που συνορεύουν (δηλαδή έχουν γραμμή, όχι απλώς σημείο για κοινό σύνορο) να έχουν διαφορετικό χρώμα;



Σχήμα 1.6: Επίπεδος χάρτης



Appel

Haken

(απόδειξη με πρόγραμμα!)

Αλγόριθμοι

- Πρωταρχική έννοια. Μέθοδος επίλυσης προβλήματος δοσμένη ως πεπερασμένο σύνολο κανόνων (ενεργειών, διεργασιών) που επενεργούν σε δεδομένα (data).
- Πεπερασμένη εκτέλεση (finiteness).
- Κάθε κανόνας ορίζεται επακριβώς και η αντίστοιχη διεργασία είναι συγκεκριμένη (definiteness).
- Δέχεται μηδέν ή περισσότερα μεγέθη εισόδου (input).
- Δίνει τουλάχιστον ένα μέγεθος ως αποτέλεσμα (output).
- Μηχανιστικά αποτελεσματικός, εκτέλεση με "μολύβι και χαρτί" (effectiveness).

Πολυπλοκότητα

- Αλγορίθμου: μέτρηση του κόστους του ως προς τη χρήση υπολογιστικών πόρων, π.χ. χρόνου (υπολογιστικών βημάτων), χώρου (μνήμης), πλήθους επεξεργαστών, πλήθους μηνυμάτων επικοινωνίας, τυχαίων bit, κ.λπ.
- Προβλήματος: είναι η πολυπλοκότητα του βέλτιστου αλγόριθμου που λύνει το πρόβλημα. Για βελτιστότητα αλγορίθμου χρειάζεται και απόδειξη αντίστοιχου κάτω φράγματος.
- Χωρίζεται σε: συγκεκριμένη (concrete) – μη συγκεκριμένη, αφηρημένη (abstract).

Είδη πολυπλοκότητας

- Χειρότερης περίπτωσης (worst case): με αυτήν ασχολούμαστε συνήθως.
- Μέσης περίπτωσης (average case): με βάση κατανομή πιθανότητας στιγμιοτύπων (instances) του προβλήματος. Συνήθως δύσκολο να οριστεί σωστά.
- Αποσβετική (amortized): εκφράζει την μέση αποδοτικότητα σε μια σειρά επαναλήψεων του αλγορίθμου.

Πολυπλοκότητα χειρότερης περίπτωσης

- *Κόστος αλγορίθμου* A είναι η συνάρτηση

  $cost_A(n) = max \{κόστος\ του\ A\ για\ είσοδο\ χ\}$

  για όλες τις εισόδους

  χ μήκους n

- *Κόστος προβλήματος* Π είναι η συνάρτηση

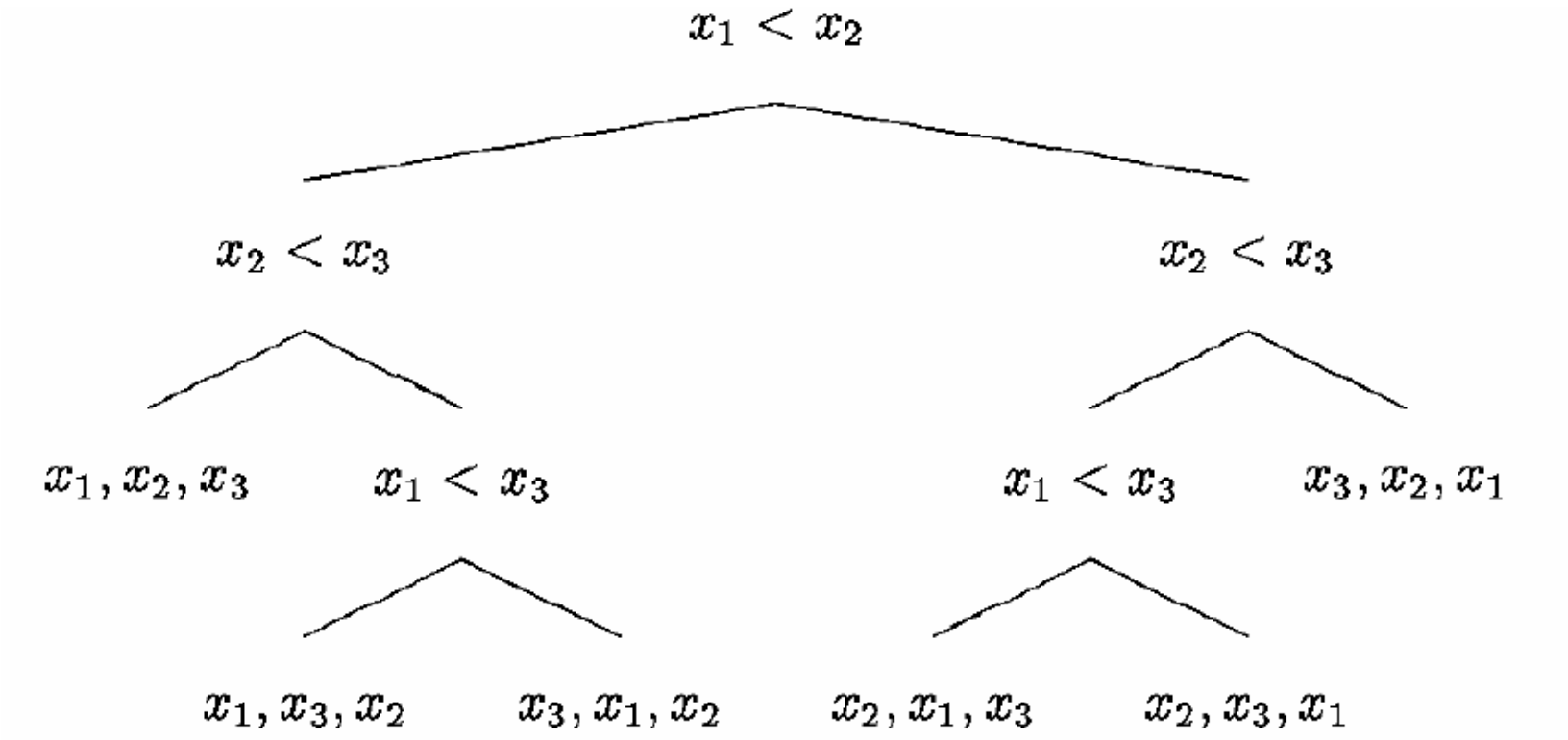
  $cost_\Pi(n) = min \{costA(n)\}$

  για όλους τους αλγορίθμους

  A που επιλύουν το Π

Πολυπλ/τα ταξινόμησης: κάτω φράγμα

Οποιοσδήποτε αλγόριθμος ταξινόμησης n αριθμών χρειάζεται $\Omega(n \log n)$ συγκρίσεις:



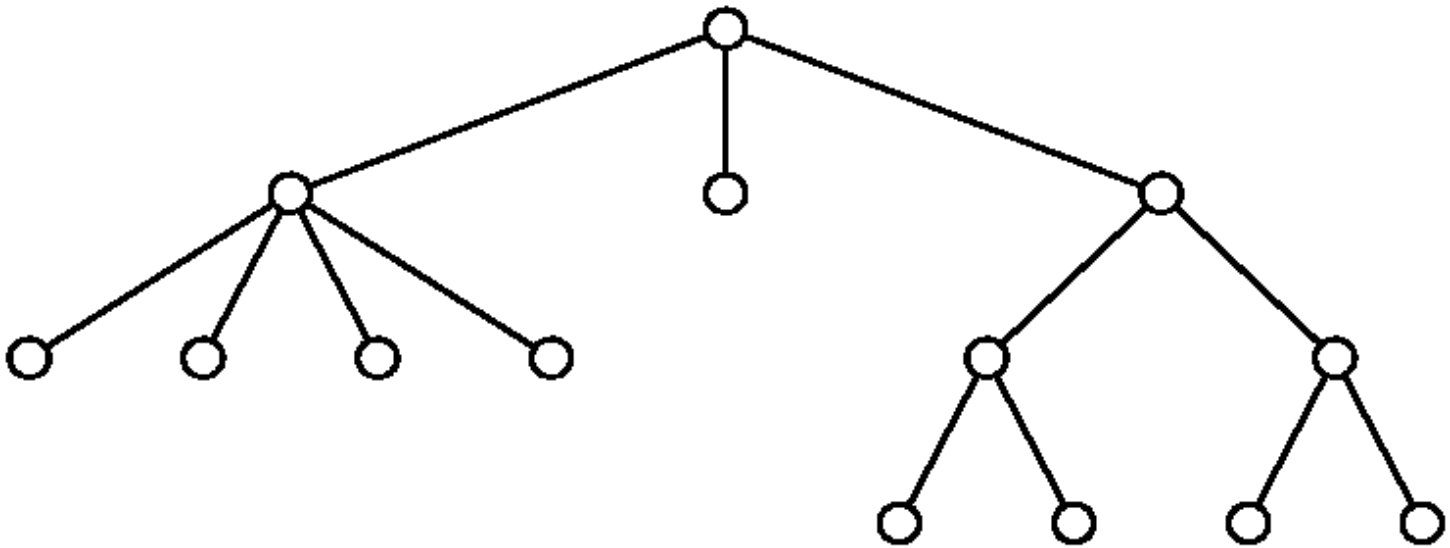
Αποσβετική πολυπλοκότητα: stacks and queues

- Πώς μπορούμε να υλοποιήσουμε ουρά με 2 στοίβες;
  - Απλοϊκός τρόπος: $O(n)$ για κάθε εξαγωγή στοιχείου, n το πλήθος των στοιχείων στην ουρά.
  - Βελτίωση: $O(1)$ κατά μέσο όρο σε μια σειρά n πράξεων. Αυτό λέγεται *αποσβετική πολυπλοκότητα*.
- Ερώτηση: υλοποίηση «άπειρης» ταινίας;

Ντετερμινιστικός αλγόριθμος (deterministic algorithm)

- Γραμμικός υπολογισμός. Για κάθε υπολογιστική διαμόρφωση (configuration) υπάρχει ακριβώς μία επόμενη.

Μη-ντετερμινιστικός αλγόριθμος (nondeterministic algorithm)



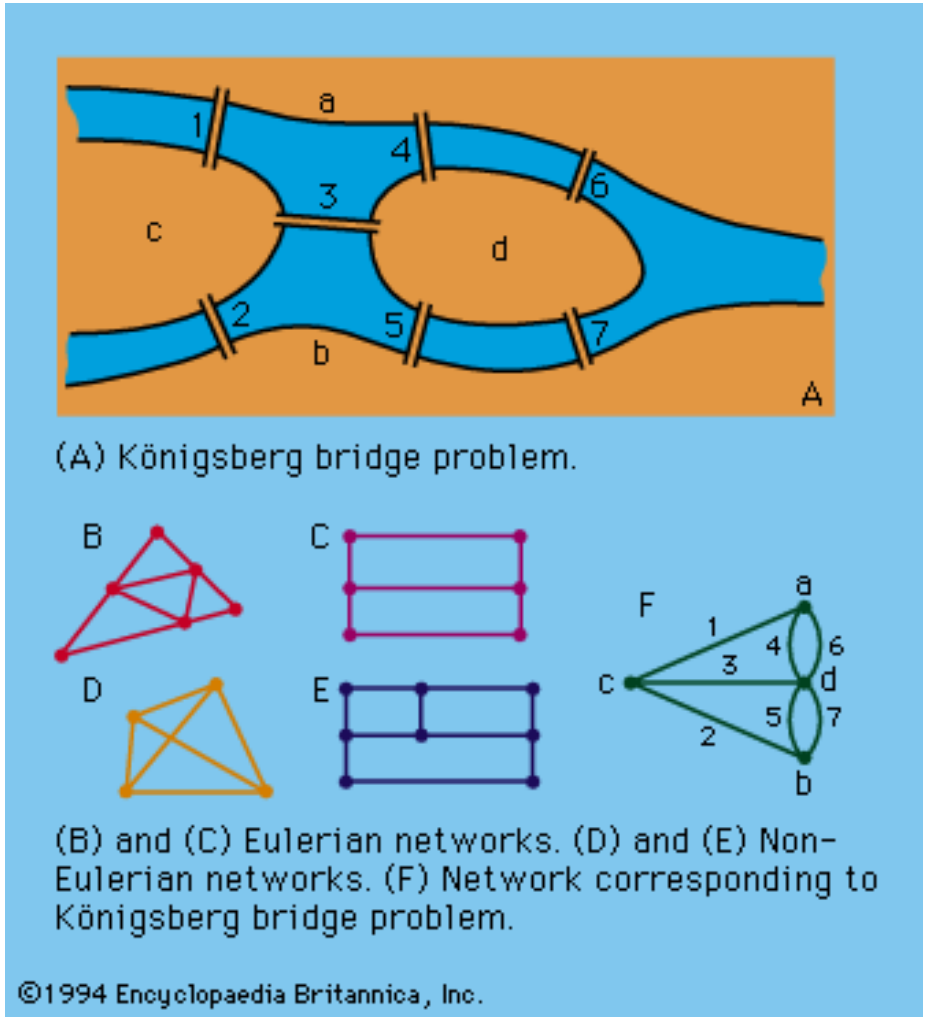
P =? NP

- Τι είναι πιο εύκολο; Να βρείτε τις λύσεις των ασκήσεων ή να τις αντιγράψετε;
- Πόσο πιο δύσκολο είναι να βρούμε κάποια λύση από το να την επαληθεύσουμε;
- Αυτό είναι ουσιαστικά το P=NP πρόβλημα, που αποτελεί το πιο σημαντικό ανοικτό πρόβλημα σήμερα.

  Στο http://www.claymath.org προσφέρονται 1εκ. δολάρια για τη λύση του !

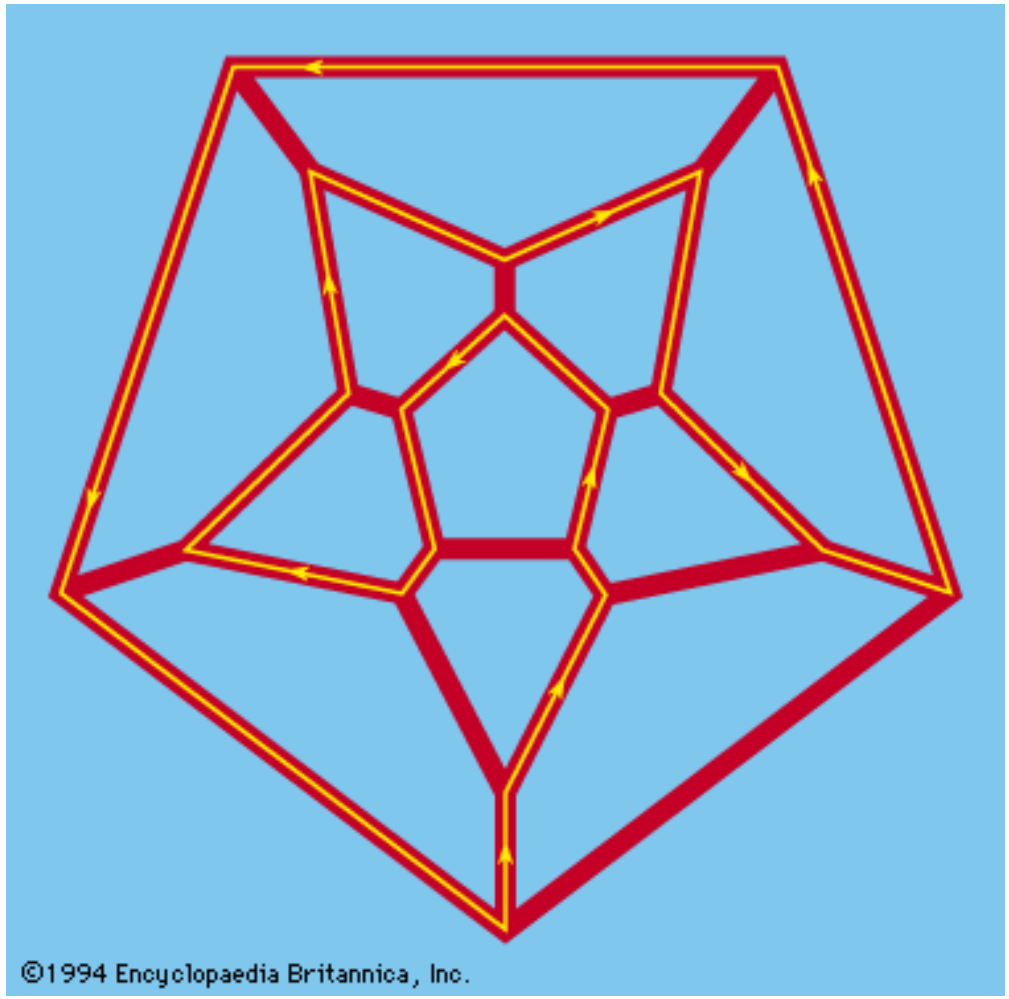
Το πρόβλημα του Euler

Δίνεται γράφος. Υπάρχει τρόπος να περάσουμε από κάθε ακμή μια ακριβώς φορά;



Το πρόβλημα του Hamilton

Δίνεται γράφος. Υπάρχει τρόπος να περάσουμε από κάθε κορυφή μια ακριβώς φορά;



Euler - Hamilton

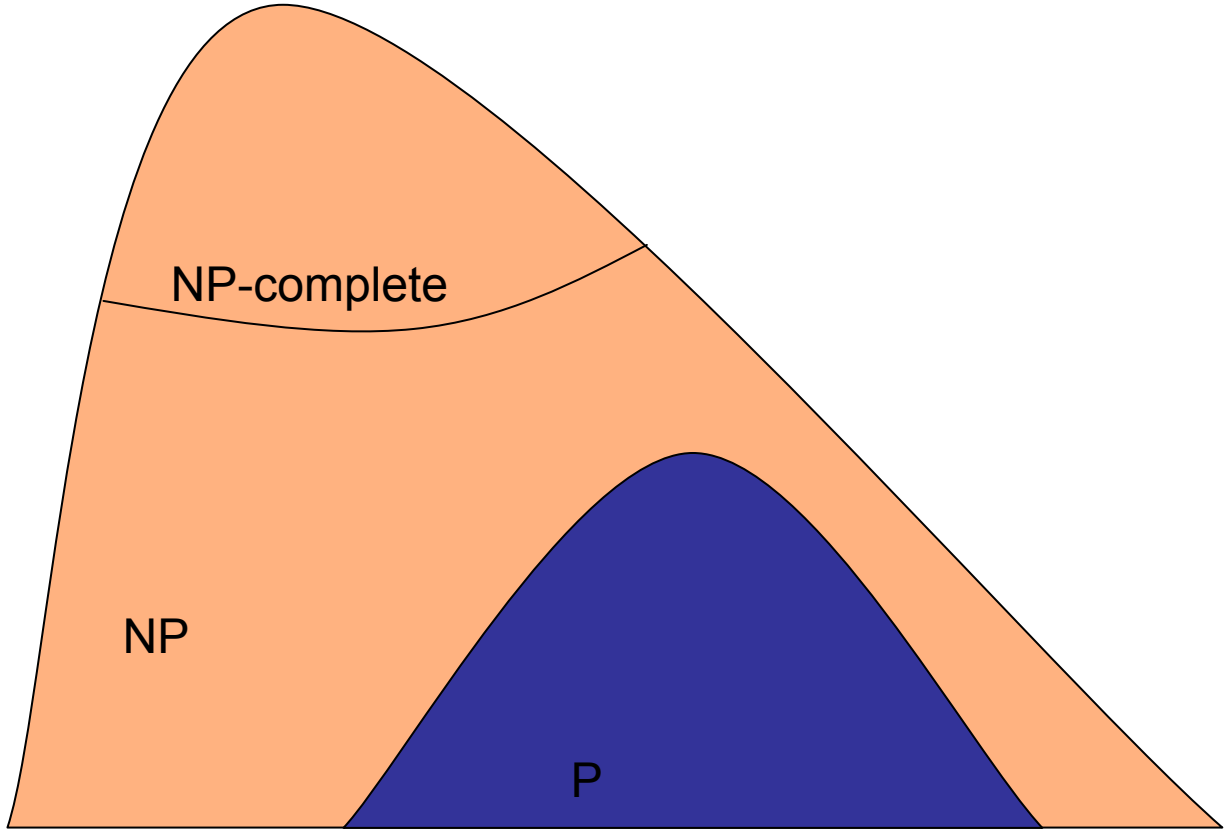
- Το πρόβλημα του Euler είναι εύκολο. Μπορούμε γρήγορα να απαντήσουμε: Ελέγχουμε αν ο αριθμός των ακμών σε κάθε κόμβο είναι άρτιος.
  - Τέτοια προβλήματα που οι αλγόριθμοι τους χρειάζονται χρόνο $O(n)$, $O(n^2)$, $O(n^3)$ … ανήκουν στην κλάση P (polynomial time).
- Το πρόβλημα του Hamilton είναι πιο δύσκολο. Δεν γνωρίζουμε κανένα γρήγορο αλγόριθμο γι' αυτό. Ο καλύτερος γνωστός αλγόριθμος δεν διαφέρει ουσιαστικά από το να δοκιμάσουμε όλους τους συνδυασμούς --- που είναι $n!=1.2.3…n$.

NP-complete προβλήματα

- Το πρόβλημα του Hamilton μπορεί να έχει γρήγορο αλγόριθμο. Δεν πιστεύουμε όμως ότι έχει. Ούτε καταφέραμε να αποδείξουμε κάτι τέτοιο.
- Το μόνο που μπορούμε να δείξουμε είναι ότι μια πλειάδα από προβλήματα που μας ενδιαφέρουν είναι της ίδιας δυσκολίας.
- Τα προβλήματα που είναι το ίδιο δύσκολα με το πρόβλημα του Hamilton τα λέμε NP-complete.

Κλάσεις πολυπλοκότητας

- **P (polynomial time)**: Το σύνολο των προβλημάτων που έχουν αλγόριθμο πολυωνυμικού χρόνου. Τα ταυτίζουμε με τα προβλήματα που μπορούμε να λύσουμε στην πράξη.
  - Το πρόβλημα του Euler ανήκει στο P
- **NP (nondeterministic polynomial time)**: Το σύνολο των προβλημάτων που μπορούμε να επιβεβαιώσουμε τη λύση τους (αν μας δοθεί) σε πολυωνυμικό χρόνο.
- **NP-complete**: Το υποσύνολο των πιο δύσκολων προβλημάτων του NP. Αν ένα από αυτά τα προβλήματα ανήκει στο P, τότε P=NP.
  - Το πρόβλημα του Hamilton είναι NP-complete.


NP-complete
NP
P

Άλλα NP-complete προβλήματα

- Satisfiability
  - Δίνεται Boolean φόρμουλα $\varphi(x_1,\ldots,x_n)$. Υπάρχουν τιμές για τα $x_1,\ldots,x_n$ που να ικανοποιούν την φ;
- Partition
  - Δίνονται ακέραιοι $a_1,\ldots,a_n$. Μπορούν να χωριστούν σε δύο μέρη με ίσα αθροίσματα;
- Πάρα πολλά άλλα προβλήματα.

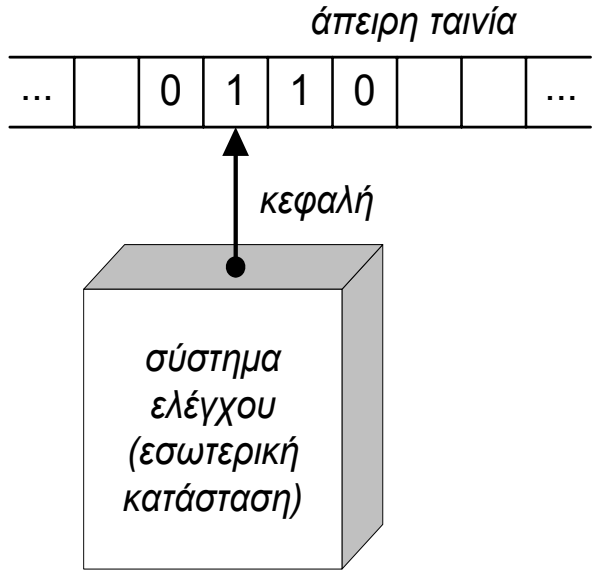
Ταξινόμηση αλγορίθμων

Ανάλογα με:

- τις δομές δεδομένων που χρησιμοποιούν
- το είδος των δεδομένων που χρησιμοποιούν (πραγματικούς αριθμούς–αριθμητική ανάλυση, γράφους, κ.τ.λ.)
- την πολυπλοκότητά τους
- την στρατηγική σχεδιασμού τους (π.χ. divide and conquer, greedy, dynamic programming, backtracking κ.τ.λ.)

Μοντέλα Υπολογισμού

- Αυστηρός ορισμός αλγορίθμων με χρήση υπολογιστικών μοντέλων: Alan Turing, Alonso Church, Stephen Kleene, Emil Post, Andrey Markov, κ.ά.
- Το πλέον «φυσικό» μοντέλο: Μηχανή Turing.



Θέση των Church-Turing

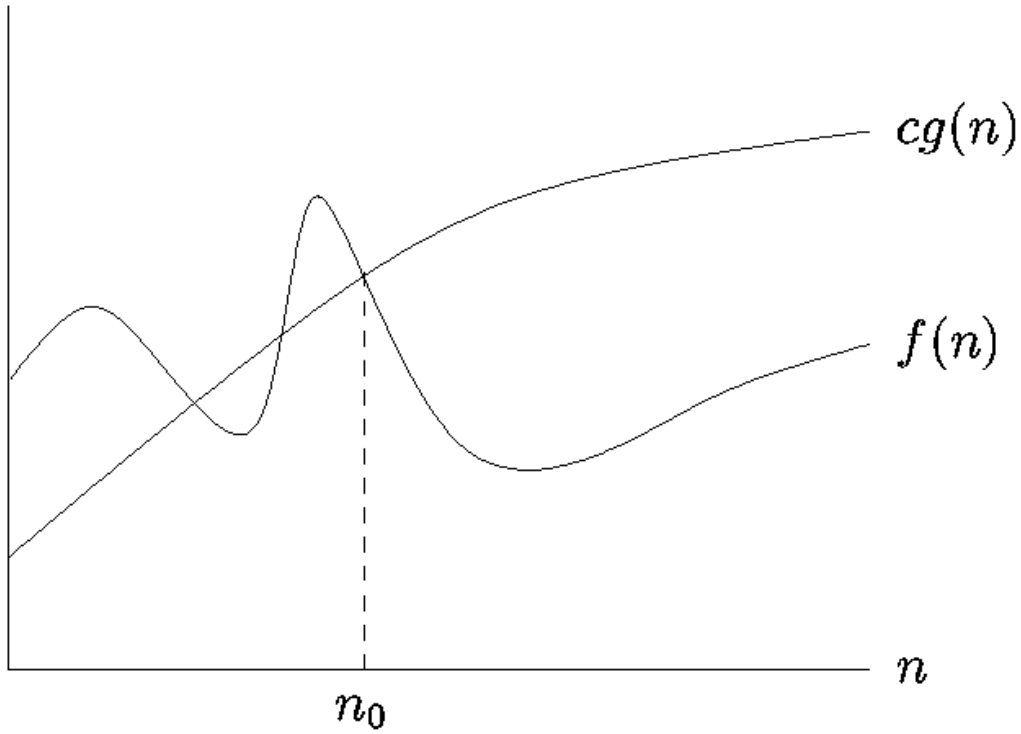
Κάθε αλγόριθμος μπορεί να περιγραφεί με τη βοήθεια μιας μηχανής Turing.

Ισοδύναμη διατύπωση:

Όλα τα γνωστά και άγνωστα υπολογιστικά μοντέλα είναι μηχανιστικά ισοδύναμα.

Δηλαδή, για κάθε ζευγάρι υπολογιστικών μοντέλων, μπορούμε με πρόγραμμα (compiler) να μεταφράζουμε αλγορίθμους από το ένα στο άλλο.

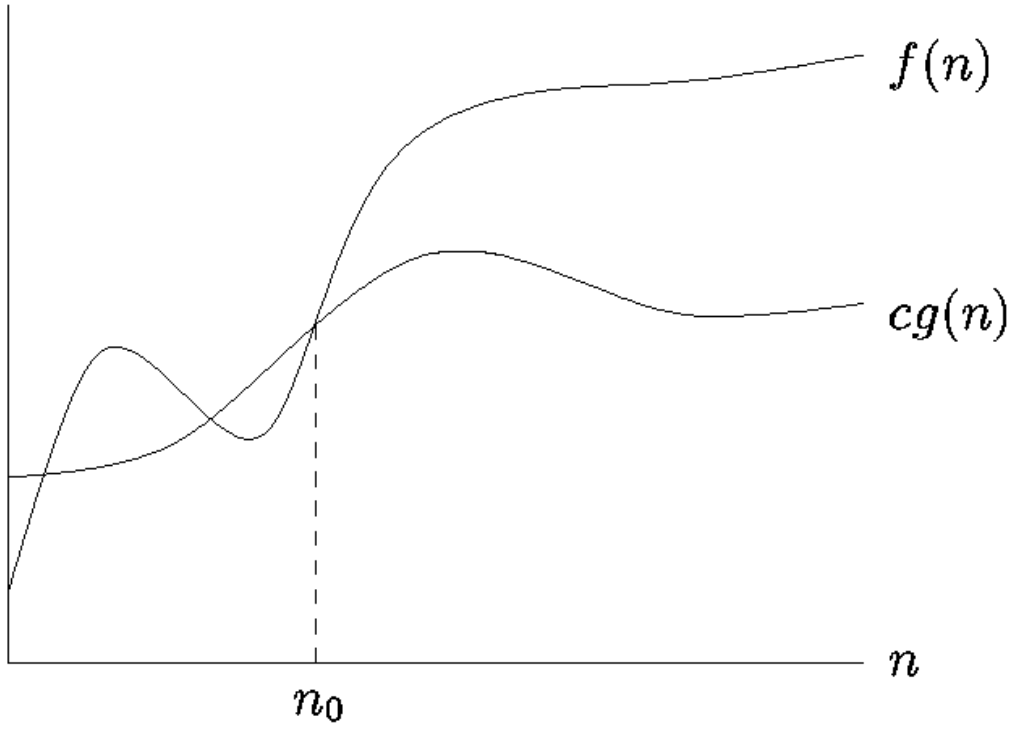
Μαθηματικοί Συμβολισμοί (i)



Σχήμα 2.3: $f = O(g)$

$$O(g) = \{f \mid \exists c > 0,\ \exists n_0 : \forall n > n_0 \quad f(n) \leq cg(n)\}$$

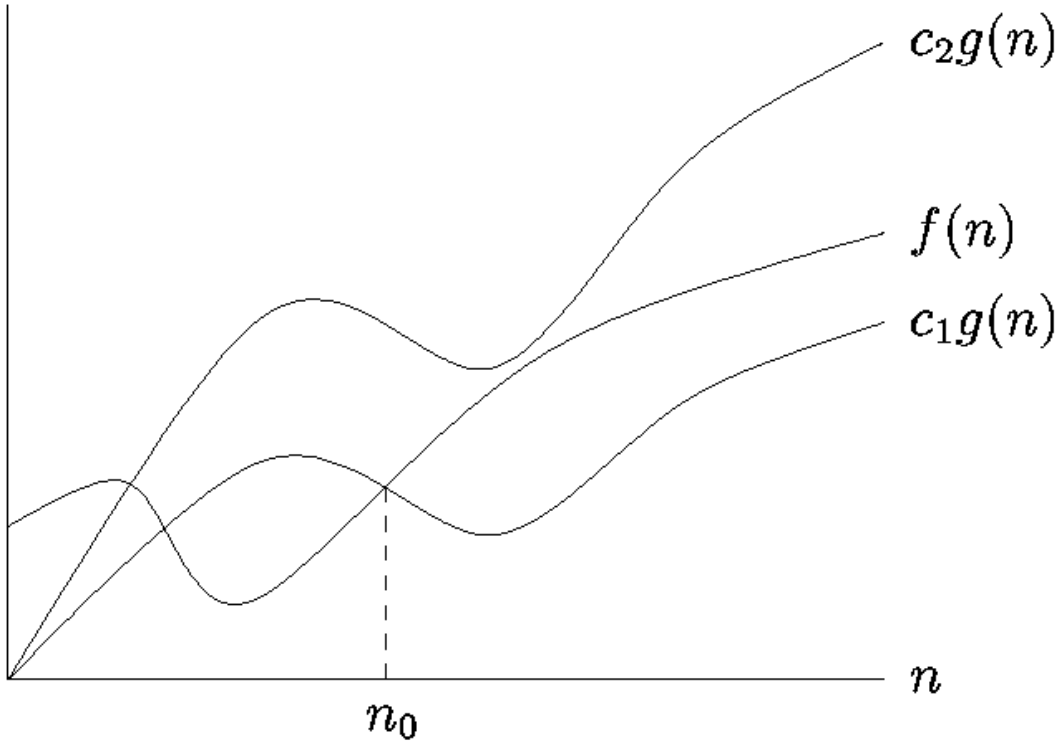
Μαθηματικοί Συμβολισμοί (ii)



Σχήμα 2.4: $f = \Omega(g)$

$$\Omega(g) = \{f \mid \exists c > 0,\ \exists n_0 : \forall n > n_0 \quad f(n) \geq cg(n)\}$$

Μαθηματικοί Συμβολισμοί (iii)



Σχήμα 2.5: $f = \Theta(g)$

$$\Theta(g) = \{f \mid \exists c_1 > 0,\ \exists c_2 > 0,\ \exists n_0 : \forall n > n_0\ \ c_1 \leq \frac{f(n)}{g(n)} \leq c_2\}$$

Μαθηματικοί Συμβολισμοί: ιδιότητες

Αν $g \in O(f)$ συνήθως γράφουμε $g(n) = O(f(n))$ και λέμε ότι συνάρτηση g είναι τάξης μεγέθους f.

Ισχύουν: $\Theta(f) = O(f) \bigcap \Omega(f)$ και $f \in \Theta(g) \iff (f \in O(g)$ και $g \in O(f))$.

Αν $p = c_k n^k + c_{k-1} n^{k-1} + \cdots + c_0$, δηλαδή πολυώνυμο βαθμού k, τότε $p \in \mathrm{O}(n^k)$ ή $p(n) = \mathrm{O}(n^k)$. Επίσης $p \in \Omega(n^k)$ ή $p(n) = \Omega(n^k)$. Συνεπώς $p(n) = \Theta(n^k)$.

Ορίζουμε $\mathrm{O}(poly) = \bigcup \mathrm{O}(n^k)$.

Μαθηματικοί Συμβολισμοί: ιδιότητες

$$O(1) < O(\alpha(n)) < O(\log^* n)$$
$$< O(\log(n)) < O(\sqrt{n}) < O(n)$$
$$< O(n\log(n)) < O(n^2) < \ldots < O(poly)$$
$$< O(2^n) < O(n!) < O(n^n) < O(A(n))$$

Σημείωση: γράφουμε "$<$" αντί "$\subset$".

log*n: πόσες φορές πρέπει να λογαριθμήσουμε το n για να φτάσουμε κάτω από το 1 (αντίστροφη υπερεκθετικής)

A: Ackermann.

α: αντίστροφη της A.

Εύρεση Μέγιστου Κοινού Διαιρέτη (gcd)

Δεν είναι λογικό να ανάγεται στο πρόβλημα εύρεσης πρώτων παραγόντων γιατί αυτό δεν λύνεται αποδοτικά.

Απλός αλγόριθμος: *O(min(α,b))*

$z := min(a, b);$
while $(a \bmod z \neq 0)$ **or** $(a \bmod z \neq 0)$ **do** $z := z - 1;$

Αλγόριθμος με αφαιρέσεις: *O(max(α,b))*

$i := a;\ j := b;$
while $i \neq j$ **do if** $i > j$ **then** $i := i - j$ **else** $j := j - i;$
return (i)

Αλγόριθμος του Ευκλείδη: *O(log(α+b))*

$i := a;\ j := b;$
while $(i > 0)$ **and** $(j > 0)$ **do**
  if $i > j$ **then** $i := i \bmod j$ **else** $j := j \bmod i;$
return $(i + j)$

Πολυπλοκότητα Ευκλείδειου Αλγορίθμου

- $O(\log \max(a,b))$: σε κάθε 2 επαναλήψεις ο μεγαλύτερος αριθμός υποδιπλασιάζεται (γιατί;)
- $\Omega(\log \max(a,b))$: για ζεύγη διαδοχικών αριθμών Fibonacci F_{k-1}, F_k, χρειάζεται k επαναλήψεις, και $k \approx \log F_k$, αφού $F_k \approx \varphi^k/\sqrt{5}$, $\varphi = (1+\sqrt{5})/2$ (φ η *χρυσή τομή*).
- Επομένως $\Theta(\log \max(a,b)) = \Theta(\log (a+b))$.

Ύψωση σε δύναμη

```
power(a, n)
    result := 1;
    for i := 1 to n do
        result := result*a;
    return result
```

Πολυπλοκότητα: $O(n)$ - εκθετική!

Ύψωση σε δύναμη με *επαναλαμβανόμενο τετραγωνισμό (Gauss)*

```
fastpower(a, n)
    result := 1;
    while n>0 do {
        if odd(n) then result:=result*a;
        n := n div 2;
        a := a*a
    }
    return result
```

Ιδέα: $a^{13} = a^{1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0}$

Πολυπλοκότητα: $O(\log n)$ - πολυωνυμική

Αριθμοί Fibonacci

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

$F_0 = 0,\ F_1 = 1,$

$F_n = F_{n-1} + F_{n-2},\ n >= 2$

Πρόβλημα: Δίνεται n, να υπολογιστεί το F_n

Πόσο γρήγορο μπορεί να είναι το πρόγραμμά μας;

Αριθμοί Fibonacci – αναδρομικός αλγόριθμος

- ```
  F(n)
      if (n<2) then return n
      else return F(n-1)+F(n-2);
  ```

- Πολυπλοκότητα: T(n) = T(n-1) + T(n-2) + c, δηλ. η T(n) ορίζεται όπως η F(n) (+ μια σταθερά), οπότε:

$$T(n) > F(n) = \Omega(1.618^n)$$
  ```

Αριθμοί Fibonacci – καλύτερος αλγόριθμος

- ```
  F(n)
      a:=0; b:=1;
      for i:=2 to n do
          c:=b; b:=a+b; a:=c;
      return b;
  ```

- Πολυπλοκότητα: $O(n)$
  ```

Αριθμοί Fibonacci – ακόμα καλύτερος αλγόριθμος

Μπορούμε να γράψουμε τον υπολογισμό σε μορφή πινάκων:

$$\begin{bmatrix} F(n) \\ F(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F(n-1) \\ F(n-2) \end{bmatrix}$$

Από αυτό συμπεραίνουμε:

$$\begin{bmatrix} F(n) \\ F(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Και ο αριθμός των **αριθμητικών πράξεων** μειώνεται σε $O(\log n)$.

Χρόνος εκτέλεσης αλγορίθμων

- Θεωρήστε 4 προγράμματα με αριθμό βημάτων $O(2^n)$, $O(n^2)$, $O(n)$, και $O(\log n)$ που το καθένα χρειάζεται 1 δευτερόλεπτο για να υπολογίσει το F(100).
- Πόσα δευτερόλεπτα θα χρειαστούν για να υπολογίσουν το F(n);

	$c \cdot 2^n$	$c \cdot n^2$	$c \cdot n$	$c \cdot \log n$
F(100)	1	1	1	1
F(101)	2	1.02	1.01	1.002
F(110)	1024	1.21	1.1	1.02
F(200)	??????	4	2	1.15

Προβλήματα πρώτων αριθμών

- Primality testing: Δίνεται ακέραιος *n*. Είναι πρώτος;
  - Σχετικά εύκολο. Ανήκει στο P όπως έδειξαν πρόσφατα (2002) προπτυχιακοί Ινδοί φοιτητές.
- Factoring: Δίνεται ακέραιος *n*. Να βρεθούν οι πρώτοι παράγοντες του.
  - Δεν ξέρουμε αν είναι εύκολο ή δύσκολο. Πιστεύουμε ότι δεν είναι στο P, αλλά ούτε ότι είναι τόσο δύσκολο όσο τα NP-complete προβλήματα.
  - Για κβαντικούς υπολογιστές (*που δεν έχουμε ακόμα καταφέρει να κατασκευάσουμε*) ανήκει στο P.

Factoring και κρυπτογραφία

- RSA: Κρυπτογραφικό σχήμα δημοσίου κλειδιού για να στείλει η A (Alice) στον B (Bob) ένα μήνυμα m.
- Ο B διαλέγει 2 μεγάλους πρώτους αριθμούς p και q, υπολογίζει το γινόμενο n=pq, και διαλέγει ακέραιο e σχετικά πρώτο με το φ(n)=(p-1)(q-1).
- Ο B στέλνει στην A τα n και e.
- Η A στέλνει στον B τον αριθμό $c=m^e \pmod n$.
- Ο B υπολογίζει το m: $m=c^d \pmod n$, όπου το $d=e^{-1} \pmod{\varphi(n)}$.
- Παράδειγμα: p=11, q=17, n=187, e=21, d=61, m=42, c=9

- *Η ασφάλεια του RSA στηρίζεται στην (εκτιμώμενη) δυσκολία του factoring.*
- *Για την υλοποίηση του RSA χρησιμοποιούνται, μεταξύ άλλων, ο αλγόριθμος επαναλαμβανόμενου τετραγωνισμού και ο* επεκτεταμένος Ευκλείδειος *αλγόριθμος (που επιπλέον εκφράζει τον gcd(a,b) σαν γραμμικό συνδυασμό των a και b).*

Πολυπλοκότητα: ανοικτά ερωτήματα

- Εκτός από κάποιες ειδικές περιπτώσεις, για κανένα πρόβλημα δεν γνωρίζουμε πόσο γρήγορα μπορεί να λυθεί.
- Ακόμα και για τον πολλαπλασιασμό αριθμών δεν γνωρίζουμε τον ταχύτερο αλγόριθμο.
- Ο σχολικός τρόπος πολλαπλασιασμού αριθμών με n ψηφία χρειάζεται $O(n^2)$ βήματα.
- Με μέθοδο «διαίρει και κυρίευε» $O(n^{\log 3}) \approx O(n^{1.58})$ βήματα αρκούν [Karatsuba 1960].
- Υπάρχουν ακόμα καλύτεροι αλγόριθμοι που χρειάζονται περίπου $O(n \log n)$ βήματα [Schönhage-Strassen 1971, Fürer 2007].
- Υπάρχει αλγόριθμος που χρειάζεται μόνο $O(n)$ βήματα; Αυτό είναι ανοικτό ερώτημα.