Θεμελιώδη Θέματα Επιστήμης Υπολογιστών

ΣΗΜΜΥ – ΣΕΜΦΕ ΕΜΠ

3η ενότητα:

Υπολογισιμότητα και Πολυπλοκότητα

Επιμέλεια διαφανειών: Στάθης Ζάχος, Άρης Παγουρτζής

Κεντρικό ζήτημα της επιστήμης υπολογιστών

Τι μπορεί να μηχανοποιηθεί και μάλιστα αποδοτικά;

Ποια προβλήματα μπορούμε να λύσουμε με υπολογιστή και πόσο καλά;

Ποια ερωτήματα μπορούν να απαντηθούν με υπολογιστή;

- Θα φύγουμε ποτέ από το Μνημόνιο;
- Υπάρχει Θεός;
- Η πρόταση «αυτή η πρόταση είναι ψευδής» είναι αληθής;
- Ο αριθμός $2^{43112609} - 1$ είναι πρώτος;

Ποια ερωτήματα μπορούν να απαντηθούν με υπολογιστή;

Προϋποθέσεις:

- Τα δεδομένα εισόδου και εξόδου μπορούν να κωδικοποιηθούν με σύμβολα
- Υπάρχει αυστηρά καθορισμένη σχέση μεταξύ τους

Παράδειγμα: εύρεση ΜΚΔ (gcd)

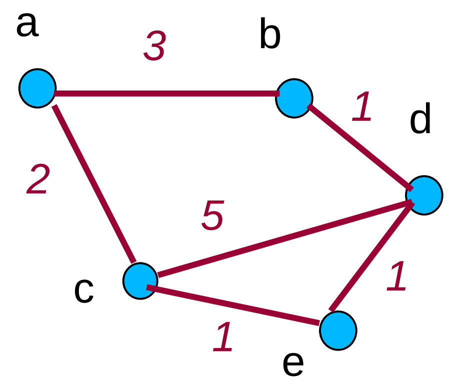
- Είσοδος: (65, 26) Έξοδος: 13
- Είσοδος: (91, 33) Έξοδος: 1

Υπολογιστικά προβλήματα

Τυπικά περιγράφονται με *διμελείς σχέσεις* (απεικονίσεις) μεταξύ συμβολοσειρών

Άλλα παραδείγματα:

- Αναγνώριση πρώτων αριθμών
  - $2^{43112609} - 1$ → «ναι»
  - 129 → «όχι»
- Συντομότερα μονοπάτια
  - (({a,b},3), ({a,c},2), ({b,d},1), ({c,d},5), ({c,e},1), ({d,e},1), a, d) → (a,c,e,d) ή (a,b,d)

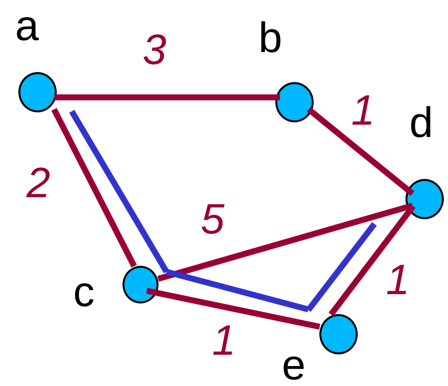


Υπολογιστικά προβλήματα

Τυπικά περιγράφονται με *διμελείς σχέσεις* (απεικονίσεις) μεταξύ συμβολοσειρών.

Άλλα παραδείγματα:

- Αναγνώριση πρώτων αριθμών
  - $2^{43112609} - 1$ → «ναι»
  - 129 → «όχι»
- Συντομότερα μονοπάτια
  - (({a,b},3), ({a,c},2), ({b,d},1), ({c,d},5), ({c,e},1), ({d,e},1), a, d) → (a,c,e,d) ή (a,b,d)



Υπολογιστικά προβλήματα (συν.)

Εμφανίζονται σε:

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

Αποτελέσματα - σταθμοί

- Gödel (1931), Church(1936), Turing (1936): δεν μπορούν να επιλυθούν όλα τα υπολογιστικά προβλήματα με υπολογιστή
  - Πρόβλημα Τερματισμού (Halting Problem)
- Cook (1971), Karp (1972): από αυτά που επιλύονται, πολλά δεν μπορούν να επιλυθούν καλά
  - Πρόβλημα Περιοδεύοντος Πωλητή (Traveling Salesperson Problem, TSP)

Υπολογισιμότητα - Πολυπλοκότητα

- Υπολογισιμότητα (Computability): *ποιά* υπολογιστικά προβλήματα μπορούμε να λύσουμε;
- Υπολογιστική πολυπλοκότητα (Computational Complexity): *πόσο καλά* μπορούμε να τα λύσουμε;
  - ως προς το χρόνο
  - ως προς το χώρο/μνήμη
  - ως προς την κατανάλωση ενέργειας
  - ως προς bandwidth
  - ...

Υπολογισιμότητα: το Πρόβλημα Τερματισμού

- Πρόβλημα Τερματισμού (Halting Problem):

  Δίνεται πρόγραμμα και είσοδος. Σταματάει το πρόγραμμα για αυτή την είσοδο (ή "τρέχει" επ' άπειρον);

- Μια ισοδύναμη παραλλαγή είναι:

  Δίνεται πρόγραμμα χωρίς είσοδο. Σταματάει;

Πρόβλημα Τερματισμού: μια ειδική περίπτωση

- Έστω το πρόγραμμα

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

- Πρόβλημα του Collatz (Ulam):

  Δίνεται φυσικός αριθμός x. Σταματάει το παραπάνω πρόγραμμα για είσοδο x;

- Παράδειγμα: **7 -> 22 -> 11 -> 34 -> 17 -> 52 -> 26 -> 13 -> 40 -> 20 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1**

Πρόβλημα Τερματισμού: μια ειδική περίπτωση (συν.)

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

- Εικασία Collatz: *το πρόγραμμα σταματάει για κάθε φυσικό αριθμό* x.
- Δεν γνωρίζουμε αν ισχύει η εικασία (ανοικτό ερώτημα) ούτε γνωρίζουμε αν το πρόβλημα Collatz είναι επιλύσιμο από υπολογιστή (αν δηλαδή μπορεί να υπάρχει πρόγραμμα που για είσοδο x να αποφαίνεται αν το πρόγραμμα Collatz σταματάει ή όχι).

Πρόβλημα Τερματισμού: μη επιλυσιμότητα

- Πρόβλημα Τερματισμού (Halting Problem):

  Δίνεται πρόγραμμα και είσοδος. Σταματάει το πρόγραμμα για αυτή την είσοδο (ή "τρέχει" επ' άπειρον);

- **Θεώρημα**. Το πρόβλημα τερματισμού είναι μη επιλύσιμο. Δηλαδή, δεν υπάρχει πρόγραμμα που να απαντάει σε αυτή την ερώτηση.

Πρόβλημα Τερματισμού: μη επιλυσιμότητα

Απόδειξη (α' τρόπος, με διαγωνιοποίηση):

- Έστω μια απαρίθμηση των προγραμμάτων Π_0, Π_1, ...
- Έστω ότι υπάρχει πρόγραμμα T, ώστε για κάθε Π_j, k, $T(\Pi_j, k)$ = "yes" αν $\Pi_j(k)$ σταματάει, "no" αλλιώς.
- Τότε υπάρχει και πρόγραμμα D που με είσοδο οποιοδήποτε k κάνει το αντίθετο από το $\Pi_k(k)$, δηλ. αν το $\Pi_k(k)$ σταματάει το $D(k)$ "τρέχει" επ' άπειρον, και αν το $\Pi_k(k)$ "τρέχει" επ' άπειρον το $D(k)$ σταματάει:

$D(k)$:
```
if T(Πk, k)="yes" then loop for ever
else stop
```

Πρόβλημα Τερματισμού: μη επιλυσιμότητα

Απόδειξη (α' τρόπος, συν.): έστω ότι το D είναι το Π_n

$\Pi_n(k)$: **if** $\texttt{T}(\Pi_k, k)$="yes" **then** loop for ever
else stop

- Τι κάνει το $\Pi_n(n)$;
  - Αν $T(\Pi_n,n)$="yes" (δηλ. $\Pi_n(n)$ *σταματάει*) τότε $\Pi_n(n)$ *τρέχει επ' άπειρον*!
  - Αν $T(\Pi_n,n)$)="no" (δηλ. $\Pi_n(n)$ *τρέχει επ' άπειρον*) τότε $\Pi_n(n)$ *σταματάει*!!
- ΑΤΟΠΟ: η μόνη υπόθεση που κάναμε είναι η ύπαρξη του προγράμματος Τ, άρα τέτοιο πρόγραμμα δεν μπορεί να υπάρχει!

Πρόβλημα Τερματισμού: μη επιλυσιμότητα

Απόδειξη (β' τρόπος, έμμεση διαγωνιοποίηση):

- Έστω ότι υπάρχει πρόγραμμα T, ώστε για κάθε Π,x, T(Π,x) = "yes" αν Π(x) σταματάει, "no" αλλιώς.
- Τότε υπάρχει και πρόγραμμα D που με είσοδο οποιοδήποτε Π κάνει το αντίθετο από το Π(Π), δηλ. αν το Π(Π) σταματάει το D(Π) "τρέχει" επ' άπειρον, και αν το Π(Π) "τρέχει" επ' άπειρον το D(Π) σταματάει:

```
D(Π): if T(Π,Π)="yes" then loop for ever
       else stop
```

Πρόβλημα Τερματισμού: μη επιλυσιμότητα

Απόδειξη (β' τρόπος, συν.):

D(D): **if** T(D,D)="yes" **then** loop for ever
else stop

- Τι κάνει το D(D);
  - Αν T(D,D) = "yes" (δηλ. D(D) *σταματάει*) τότε D(D) *τρέχει επ' άπειρον*!
  - Αν T(D,D) = "no" (δηλ. D(D) *τρέχει επ' άπειρον*) τότε D(D) *σταματάει*!!
- ΑΤΟΠΟ: η μόνη υπόθεση που κάναμε είναι η ύπαρξη του προγράμματος T, άρα τέτοιο πρόγραμμα δεν μπορεί να υπάρχει!

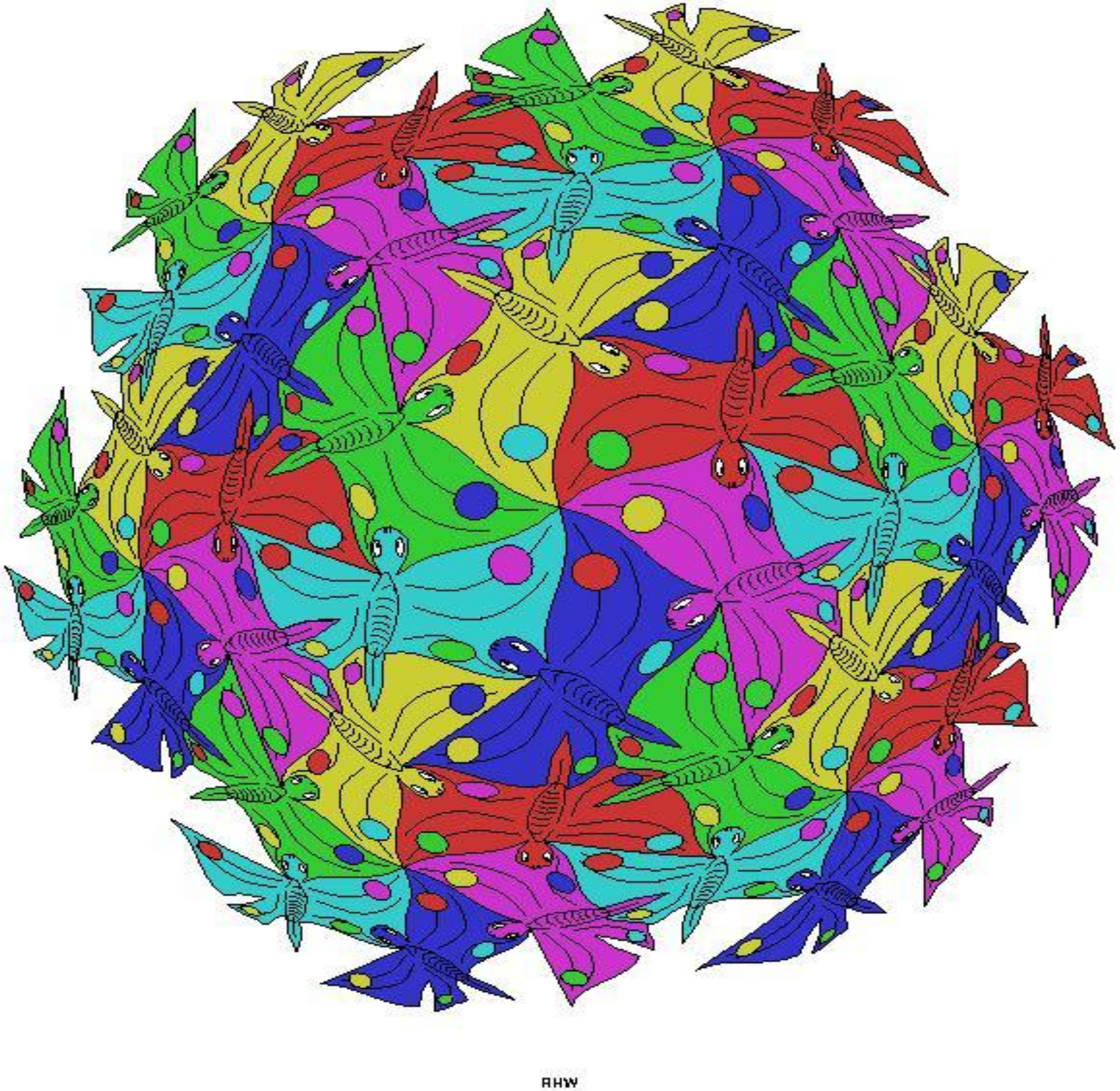
Πολυπλοκότητα υπολογιστικών προβλημάτων

- Για τα προβλήματα που επιλύονται (solvable, computable, decidable) μας ενδιαφέρει το πόσο καλά μπορεί να γίνει αυτό, δηλαδή πόσο γρήγορα, ή με πόση μνήμη, ή με πόσους επεξεργαστές (παραλληλία), ή με πόση κατανάλωση ενέργειας (sensor networks), κ.λπ.

- Αυτό λέγεται (υπολογιστική) πολυπλοκότητα.

Τι είναι πολυπλοκότητα;

- Το 101101011101 είναι πιο πολύπλοκο από το 010101010101
- 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.

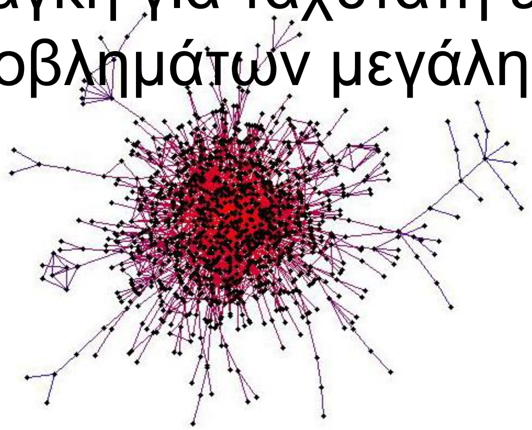

RHW

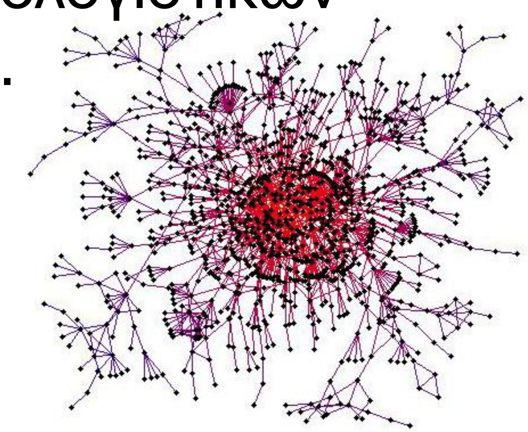
Τι είναι υπολογιστική πολυπλοκότητα;

- Η δυσκολία του να υπολογίσουμε τη λύση σε ένα πρόβλημα.
- Επιπλέον, ένας τρόπος για να εκφράσουμε μαθηματικά τη διαίσθησή μας ότι οι πρώτοι αριθμοί είναι πιο πολύπλοκοι από τους περιττούς.
- Το πρόβλημα «Δίνεται x. Είναι πρώτος;» είναι *υπολογιστικά* πιο δύσκολο από το πρόβλημα «Δίνεται x. Είναι περιττός;»

Η πρόκληση: σύγχρονα δίκτυα και συστήματα

- Πολύπλοκα, πολλές (ετερογενείς) συνιστώσες που αλληλεπιδρούν.
- Διακίνηση τεράστιου όγκου πληροφορίας.
- Ανάγκη για άμεση επεξεργασία δεδομένων και λήψη αποφάσεων.
- Ανάγκη για ταχύτατη επίλυση υπολογιστικών προβλημάτων μεγάλης κλίμακας.





Καθορισμός πολυπλοκότητας υπολογιστικών προβλημάτων

- Αλγόριθμοι: παρέχουν άνω φράγματα
  - ταξινόμηση (με bubblesort): $O(n^2)$

- Αποδείξεις δυσκολίας: παρέχουν κάτω φράγματα
  - ταξινόμηση με συγκρίσεις: $\Omega(n \log n)$
  - NP-πληρότητα: ισχυρή ένδειξη απουσίας αποδοτικού αλγορίθμου

million dollar question! (Clay Institute millennium problems)

Πολυπλοκότητα αλγορίθμου

- Μέτρηση του κόστους του σαν συνάρτηση των υπολογιστικών πόρων που απαιτούνται σε σχέση με το μέγεθος της (αναπαράστασης της) εισόδου:

  $\mathrm{cost}_A(n)$ = max {κόστος αλγορ. A για είσοδο x}

  για όλες τις εισόδους x μήκους n

- Παράδειγμα: $\text{time-cost}_{MS}(n) <= c\ n \log n$ (MS = MergeSort, c κάποια σταθερά)

Πολυπλοκότητα αλγορίθμου: απλοποιήσεις

- Συχνά θεωρούμε ως μέγεθος της εισόδου το πλήθος των δεδομένων μόνο (αγνοώντας το μέγεθός τους σε bits). Αυτό δεν δημιουργεί πρόβλημα εφ'όσον ο αλγόριθμος δεν περιέχει πράξεις ή διαδικασίες που να κοστίζουν *εκθετικά* ως προς το μέγεθος των δεδομένων σε bits.
- Επίσης θεωρούμε ότι κάθε στοιχειώδης αριθμητική πράξη (πρόσθεση, πολ/σμός, σύγκριση) έχει κόστος 1 βήματος. Αυτό λέγεται αριθμητική πολυπλοκότητα (arithmetic complexity) και είναι συνήθως αρκετά ακριβής μέτρηση. Η ανάλυση πολυπλοκότητας σε πλήθος πράξεων ψηφίων λέγεται bit complexity.

Πολυπλοκότητα προβλήματος

- Είναι η πολυπλοκότητα του βέλτιστου αλγορίθμου που λύνει το πρόβλημα.

  $\text{cost}_{\Pi}(n) = \min\ \{\text{cost}_{A}(n)\}$

  για όλους τους αλγορίθμους

  A που επιλύουν το Π

- Παράδειγμα: $\text{time-cost}_{\text{SORT}}(n) <= c\ n \log n$ [$= O(n \log n)$]
  (SORT = πρόβλημα ταξινόμησης)

- Για να δείξουμε *βελτιστότητα αλγορίθμου* χρειάζεται και *απόδειξη αντίστοιχου κάτω φράγματος*.

Πολυπλοκότητα ταξινόμησης: κάτω φράγμα

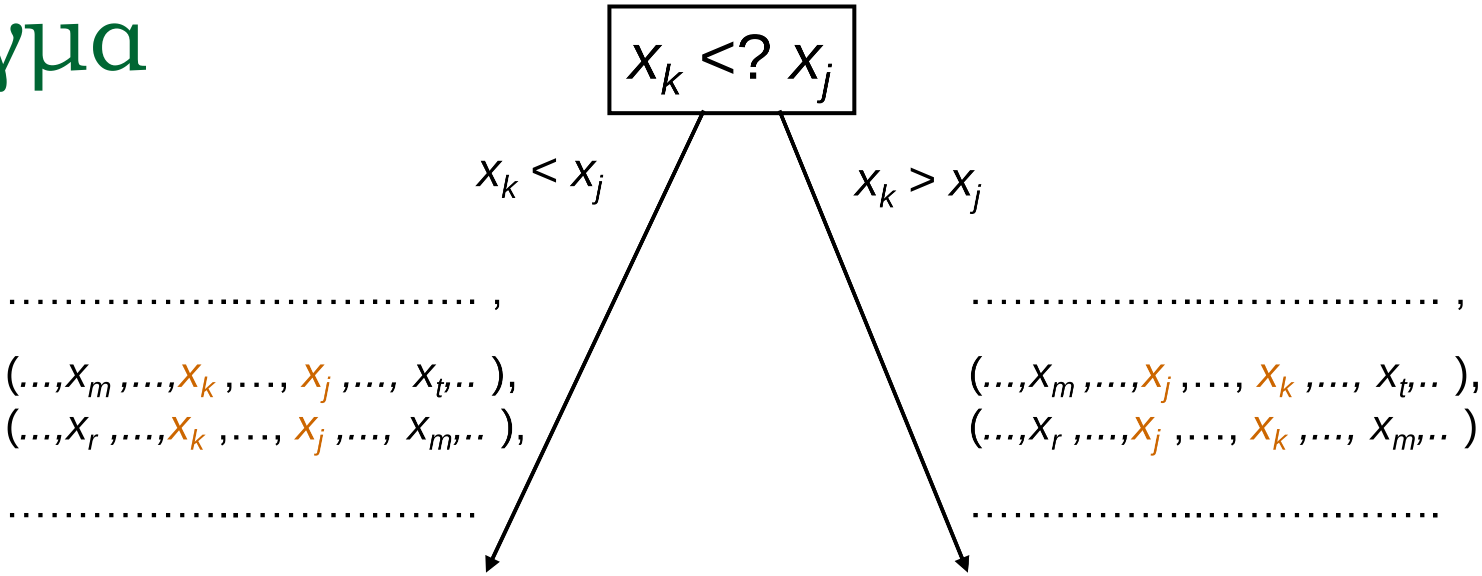
Οποιοσδήποτε αλγόριθμος ταξινόμησης n αριθμών χρειάζεται $\Omega(n \log n)$ συγκρίσεις:

- Είσοδος $(x_1, x_2, \ldots, x_n)$
- Αρχικά $n!$ περιπτώσεις:

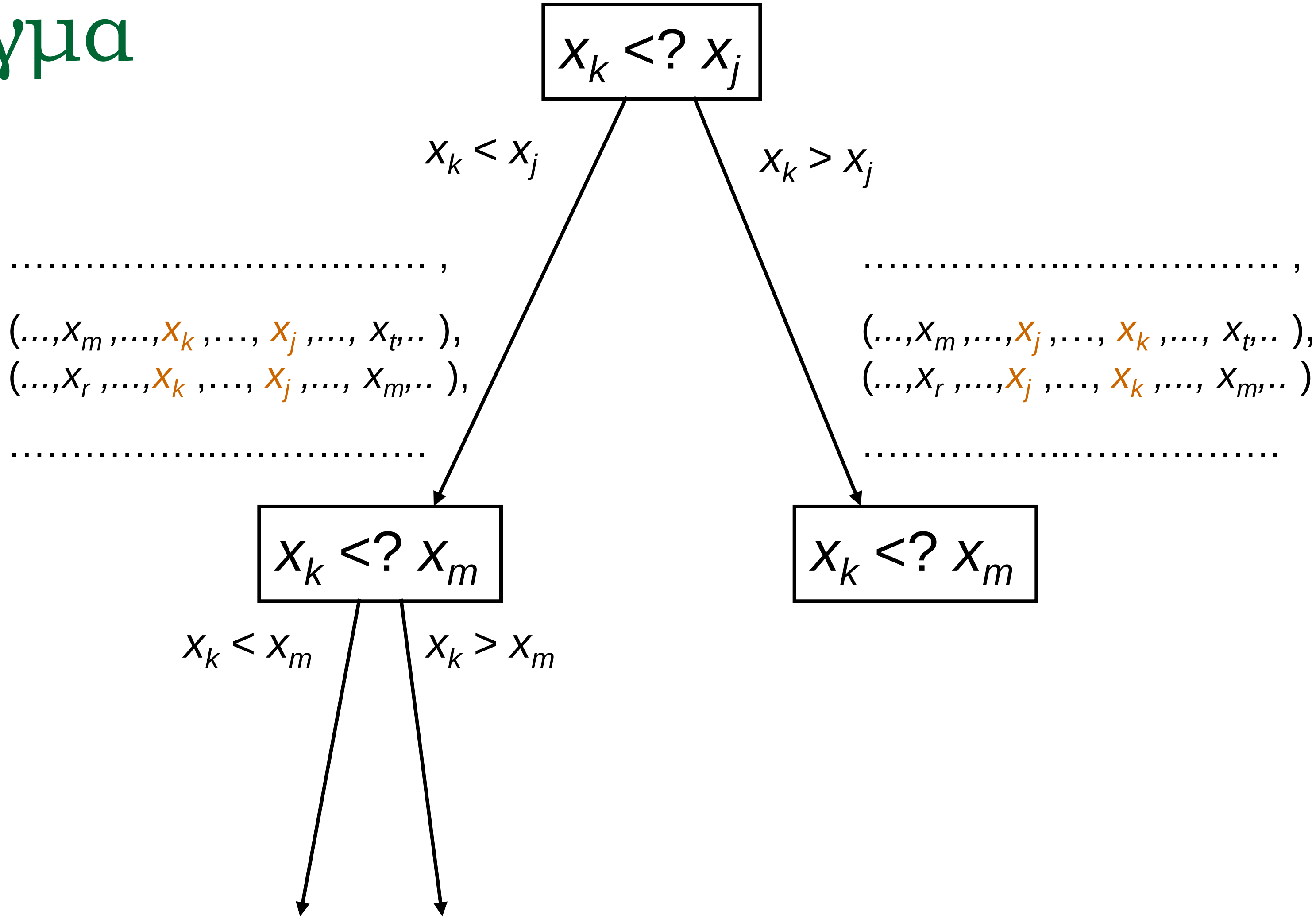
$$x_1 < x_2 < x_3 < \ldots < x_n$$
$$x_2 < x_1 < x_3 < \ldots < x_n$$
$$x_3 < x_1 < x_2 < \ldots < x_n$$
$$\ldots$$

- Σε κάθε σύγκριση το πλήθος περιπτώσεων υποδιπλ/ται (στην καλύτερη περίπτωση)
- Πλήθος συγκρίσεων: $\geq \log(n!) \geq (n \log n)/4$

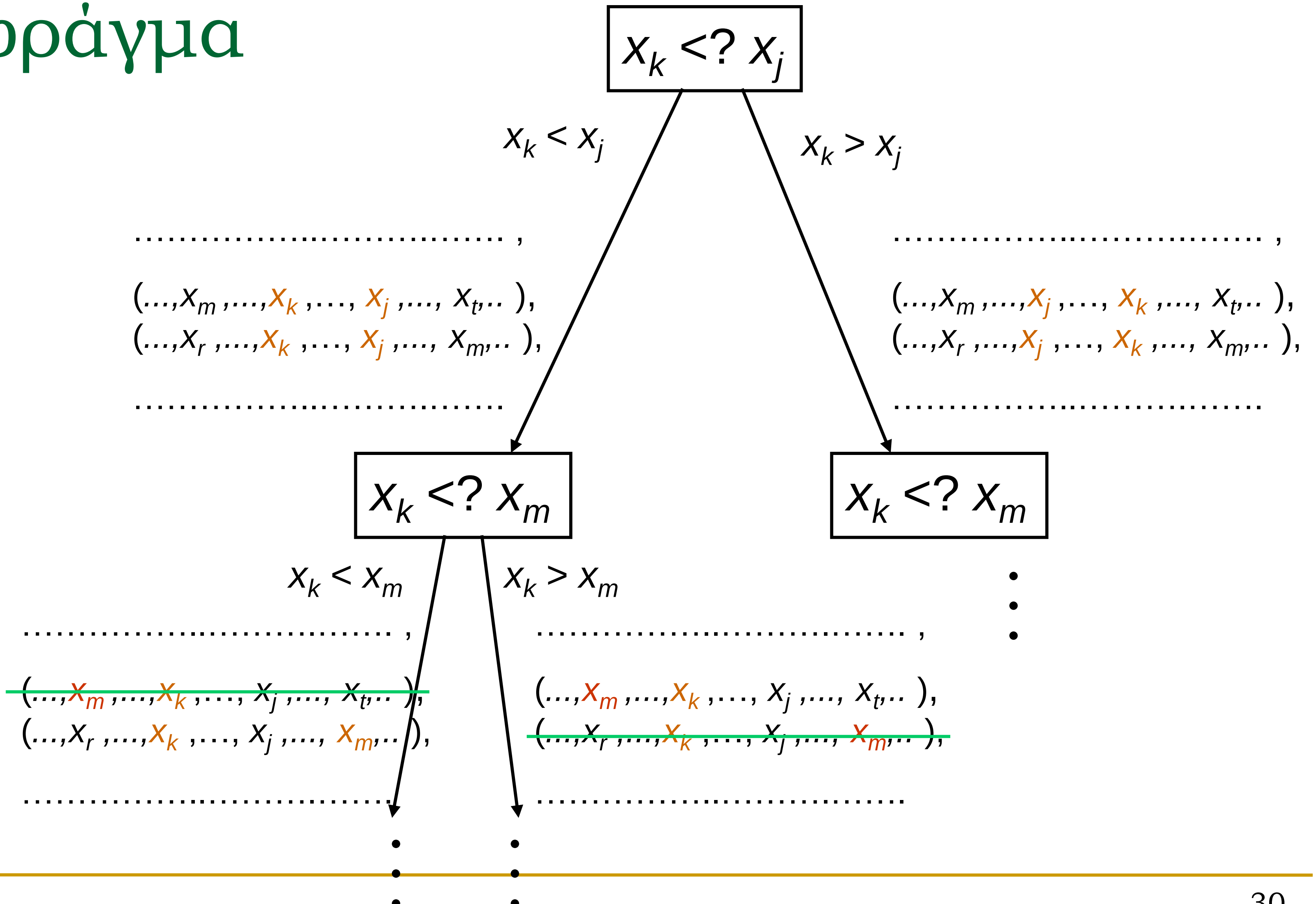
Πολυπλοκότητα ταξινόμησης: κάτω φράγμα



Πολυπλοκότητα ταξινόμησης: κάτω φράγμα



Πολυπλοκότητα ταξινόμησης: κάτω φράγμα



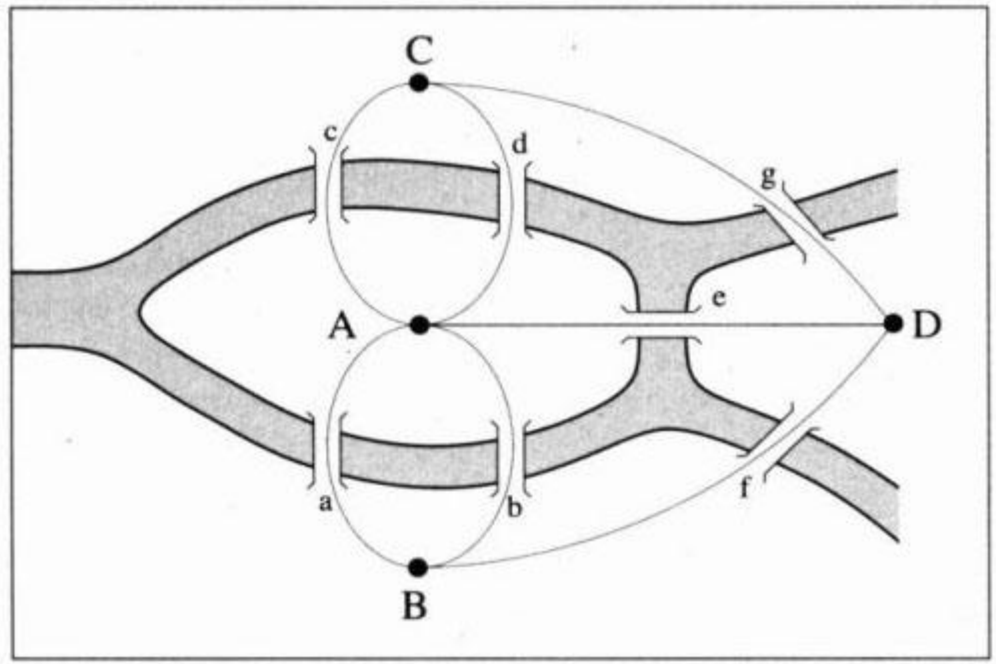
P =? NP

- 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 της Θεωρητικής Πληροφορικής 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;

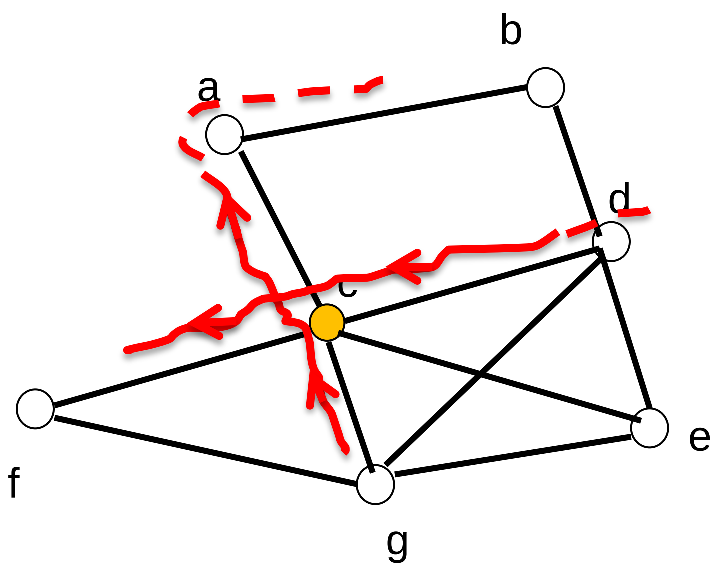


Seven Bridges of Königsberg

Source:
http://physics.weber.edu/carroll/honors_images/BarbasiBridges.jpg

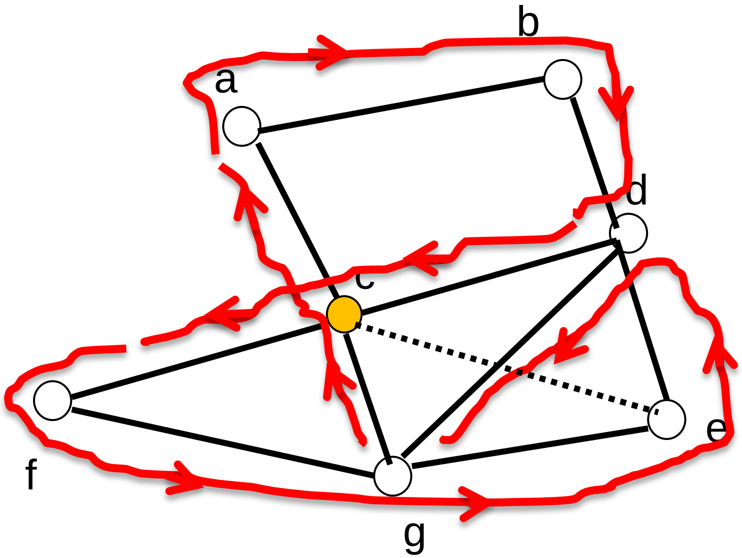
Επίλυση του προβλήματος Euler

- Το πρόβλημα του Euler είναι ευεπίλυτο. Η απάντηση είναι 'ναι' αν και μόνο αν
  κάθε κόμβος v έχει άρτιο # γειτόνων



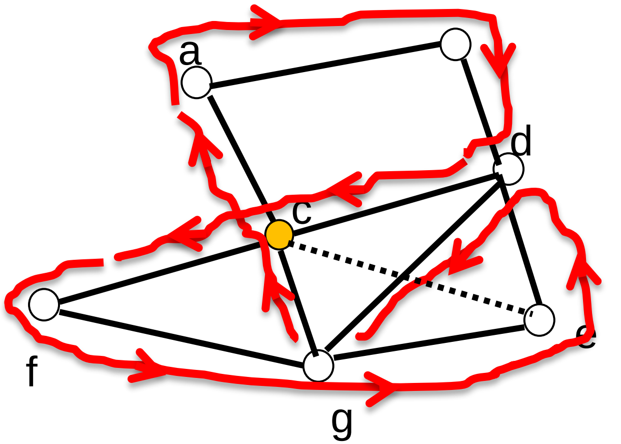
Επίλυση του προβλήματος Euler

- Το πρόβλημα του Euler είναι ευεπίλυτο. Η απάντηση είναι 'ναι' αν και μόνο αν
  κάθε κόμβος v έχει άρτιο # γειτόνων



Επίλυση του προβλήματος Euler

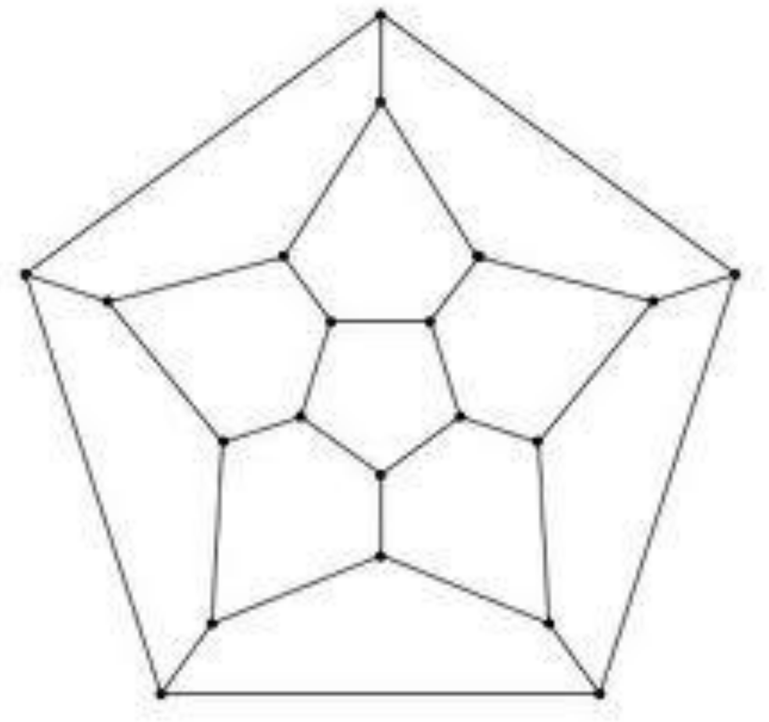
- Το πρόβλημα του Euler είναι ευεπίλυτο.
- Η απάντηση είναι 'ναι' ανν
  κάθε κόμβος v έχει
  άρτιο # γειτόνων



- Για κάθε γράφο με *n* κόμβους αρκούν n^2 έλεγχοι: χρόνος *πολυωνυμικός* ως προς το μέγεθος της εισόδου.
- Τέτοια προβλήματα που η επίλυσή τους χρειάζεται χρόνο $O(n)$, $O(n^2)$, $O(n^3)$ … λέμε ότι ανήκουν στην κλάση P (polynomial time).

Το πρόβλημα του Hamilton

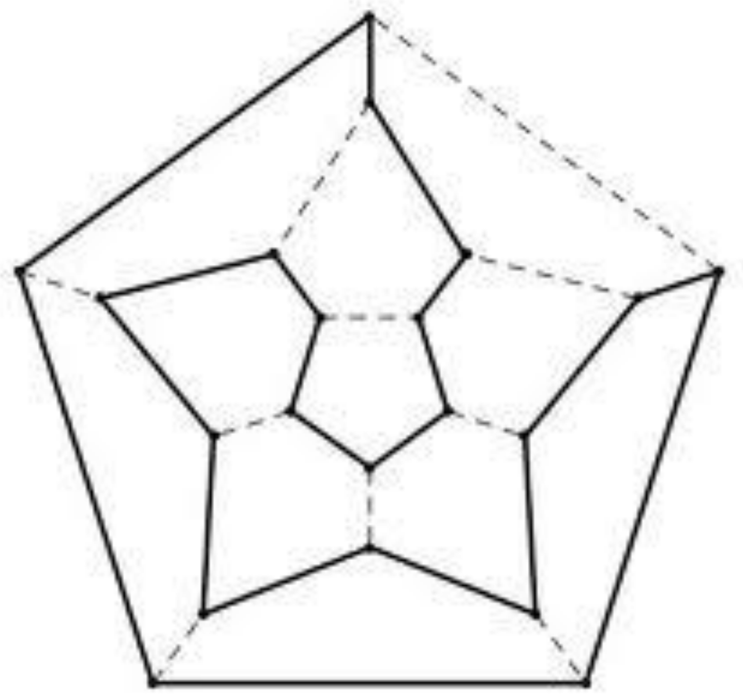
Δίνεται γράφος. Υπάρχει τρόπος να περάσουμε από κάθε κορυφή μια ακριβώς φορά;



Source: http://jwilson.coe.uga.edu/emat6680/yamaguchi/emat6690/essay1/gt.html

Το πρόβλημα του Hamilton

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



Source: http://jwilson.coe.uga.edu/emat6680/yamaguchi/emat6690/essay1/gt.html

Πολυπλοκότητα προβλήματος Hamilton

- Το πρόβλημα του Hamilton είναι πιο δύσκολο (δυσεπίλυτο). Δεν γνωρίζουμε κανέναν γρήγορο αλγόριθμο γι' αυτό. Ο καλύτερος γνωστός αλγόριθμος δεν διαφέρει ουσιαστικά από το να δοκιμάσουμε όλους τους συνδυασμούς, που είναι πολλοί ($n!$). Αν όμως μας προτείνουν μια λύση, μπορούμε να την επαληθεύσουμε πολύ γρήγορα.
- Τέτοια προβλήματα που η επαλήθευση μιας λύσης τους (αν υπάρχει και μας δοθεί) χρειάζεται χρόνο $O(n)$, $O(n^2)$, $O(n^3)$, …, λέμε ότι ανήκουν στην κλάση NP (non-deterministic polynomial time).

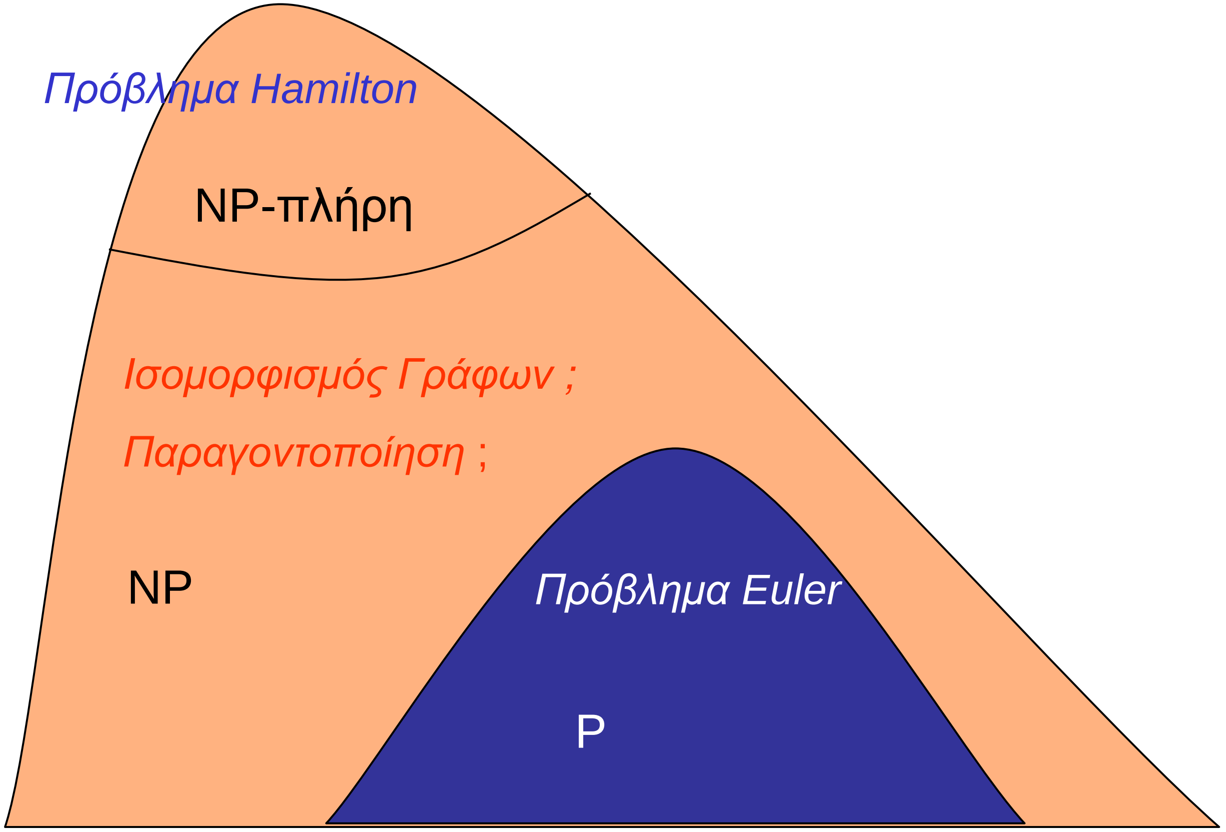
NP-πλήρη προβλήματα

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

Κλάσεις πολυπλοκότητας

- **P (πολυωνυμικός χρόνος)**: Το σύνολο των προβλημάτων που λύνονται σε πολυωνυμικό χρόνο. Θεωρούνται τα προβλήματα που μπορούμε να λύσουμε στην πράξη.
  - Το πρόβλημα του Euler ανήκει στο P
- **NP (μη ντετερμινιστικός πολυωνυμικός χρόνος)**: Το σύνολο των προβλημάτων που μπορούμε να επαληθεύσουμε τη λύση τους (αν μας δοθεί) σε πολυωνυμικό χρόνο.
- **NP-πλήρη**: Το υποσύνολο των πιο δύσκολων προβλημάτων του NP – για κανένα δεν έχει βρεθεί πολυωνυμικός αλγόριθμος. Αν οποιοδήποτε από αυτά τα προβλήματα ανήκει στο P, τότε P=NP.
  - Το πρόβλημα του Hamilton είναι NP-πλήρες.

Ο χάρτης των κλάσεων (μέχρι τώρα)



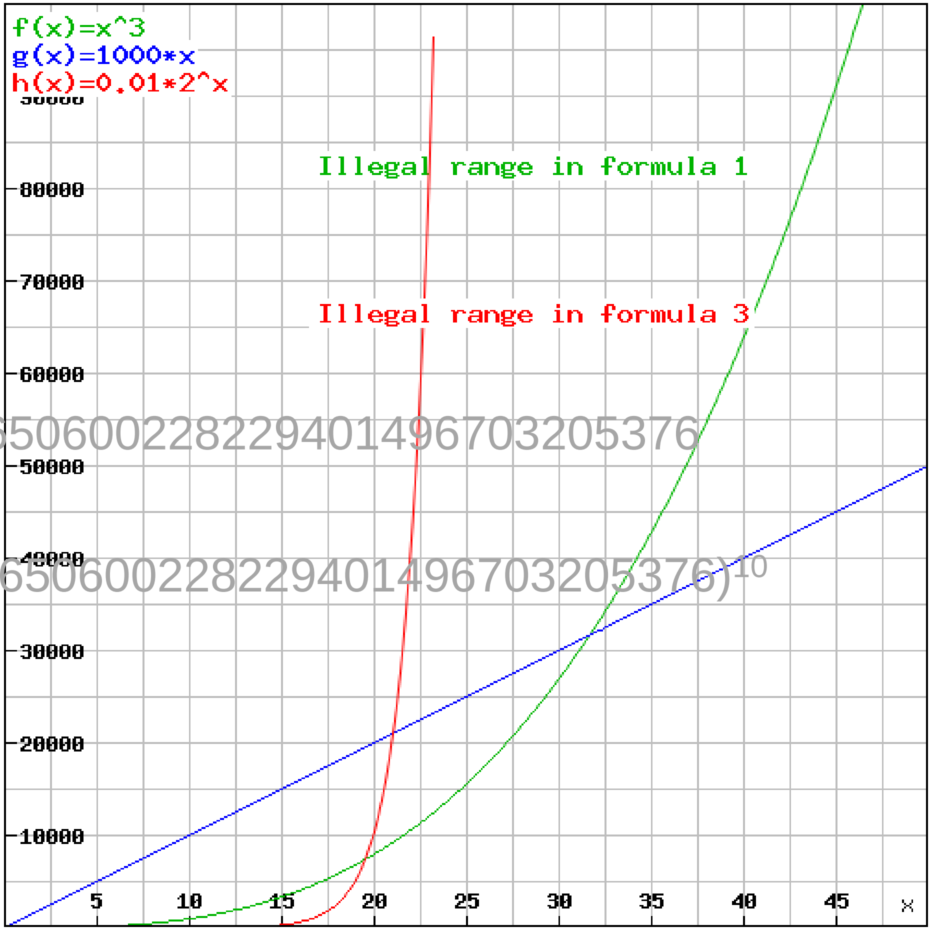
Γιατί θέλουμε πολυωνυμικό χρόνο;

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

Ο ρυθμός αύξησης των εκθετικών συναρτήσεων είναι απαγορευτικός για μεγάλα στιγμιότυπα!

Γιατί θέλουμε πολυωνυμικό χρόνο;

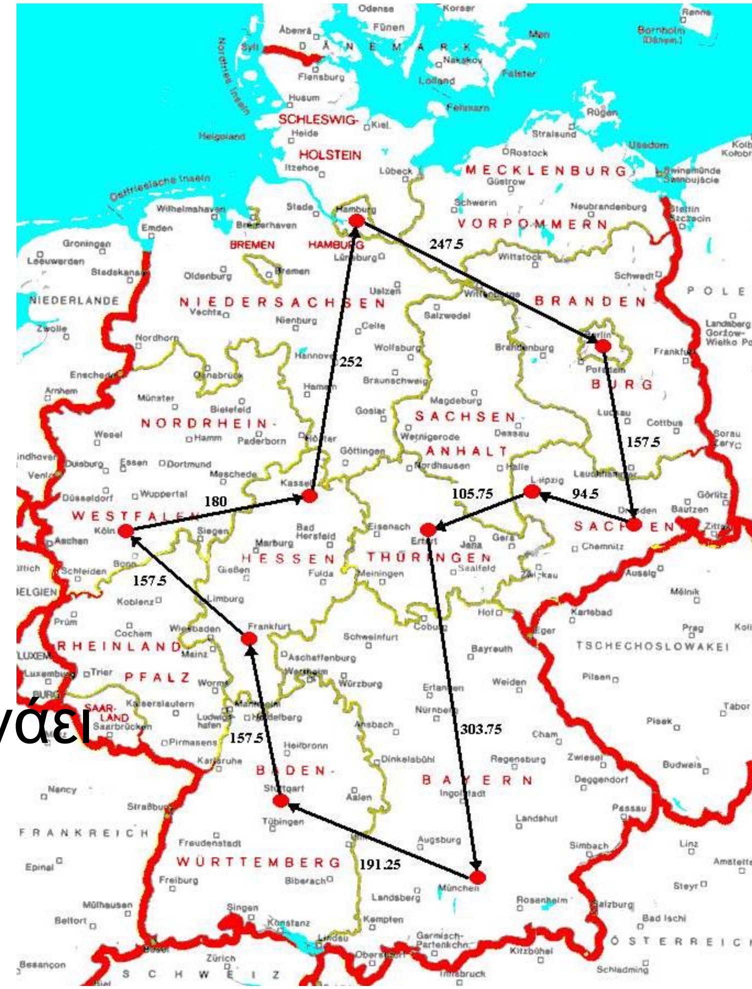
log$n$	n	n^2	2^n
3.322	10	100	1024
6.644	100	10000	1267650600228229401496703205376
9.966	1000	1000000	$(1267650600228229401496703205376)^{10}$



Ο ρυθμός αύξησης των εκθετικών συναρτήσεων είναι απαγορευτικός για μεγάλα στιγμιότυπα!

Αποδείξεις NP-πληρότητας

- Πρόβλημα Πλανόδιου Πωλητή (Traveling Salesperson Problem, TSP)
  - Δίνεται πλήρης γράφος με *n* κόμβους, οι αποστάσεις μεταξύ τους $d(v_i, v_k)$ και ένας φυσικός αριθμός *D*.
  - Υπάρχει διαδρομή που να περνάει μία φορά από κάθε κόμβο με συνολικό κόστος <= *D*;



http://myprojectsdiary.blogspot.com/2005_03_01_archive.html

Αποδείξεις NP-πληρότητας: αναγωγές

- Δοθέντος ενός στιγμιοτύπου (εισόδου) του προβλήματος Hamilton μπορούμε να το αναγάγουμε σε στιγμιότυπο του προβλήματος TSP:
  - Μετατρέπουμε τον γράφο σε πλήρη.
  - Στις υπάρχουσες ακμές θέτουμε απόσταση 1, ενώ στις νέες θέτουμε απόσταση 2.
  - Θέτουμε $D=n$.

Αποδείξεις NP-πληρότητας: αναγωγές

- Είναι εύκολο να δούμε ότι η απάντηση για το αρχικό στιγμιότυπο (του προβλήματος Hamilton) είναι «ναι», δηλαδή υπάρχει κύκλος Hamilton στον αρχικό γράφο, *αν και μόνο αν* η απάντηση για το νέο στιγμιότυπο (του προβλήματος TSP) είναι «ναι», δηλαδή υπάρχει κύκλος κόστους *n* στον νέο γράφο.

- Επομένως είναι μάλλον απίθανο το TSP να λύνεται σε πολυωνυμικό χρόνο.

Άλλα NP-πλήρη προβλήματα

- Ικανοποιησιμότητα (Satisfiability)
  - Δίνεται προτασιακός τύπος Boole $\varphi(x_1,\ldots,x_n)$. Υπάρχει ανάθεση αληθοτιμών για τα $x_1,\ldots,x_n$ που να ικανοποιεί την φ;
- Διαμέριση (Partition)
  - Δίνονται ακέραιοι $a_1,\ldots,a_n$. Μπορούν να χωριστούν σε δύο σύνολα με ίσα αθροίσματα;
- Πάρα πολλά άλλα προβλήματα.

Άλλα NP-πλήρη προβλήματα

- Partition
  - Δίνονται ακέραιοι $a_1,\ldots,a_n$ μπορούμε να τους διαμερίσουμε σε δύο σύνολα ίσου αθροίσματος;
  - Παράδειγμα [Lance Fortnow, *The Golden Ticket: P, NP, and the Search for the Impossible,* 2013]

    14175, 15055, 16616, 17495, 18072, 19390, 19731, 22161, 23320, 23717, 26343, 28725, 29127, 32257, 40020, 41867, 43155, 46298, 56734, 57176, 58306, 61848, 65825, 66042, 68634, 69189, 72936, 74287, 74537, 81942, 82027, 82623, 82802, 82988, 90467, 97042, 97507, 99564.

    - Μπορείτε να βρείτε τη λύση;
    - Κάθε σύνολο θα πρέπει να έχει άθροισμα 1000000

Ενδιάμεση πολυπλοκότητα;

- Factoring

  Δίνεται σύνθετος αριθμός N, βρείτε την παραγοντοποίησή του:

  1230186684530117755130494958384962720772853569595334792197322452151726400507263657518745202199786469389956474942774063845925192557326303453731548268507917026122142913461670429214311602221240479274737794080665351419597459856902143413

  =

  3347807169895689878604416984821269081770479498371376856891
  2431388982883793878002287614711652531743087737814467999489

  x

  3674604366679959042824463379962795263227915816434308764267
  6032283815739666511279233373417143396810270092798736308917

Ενδιάμεση πολυπλοκότητα;

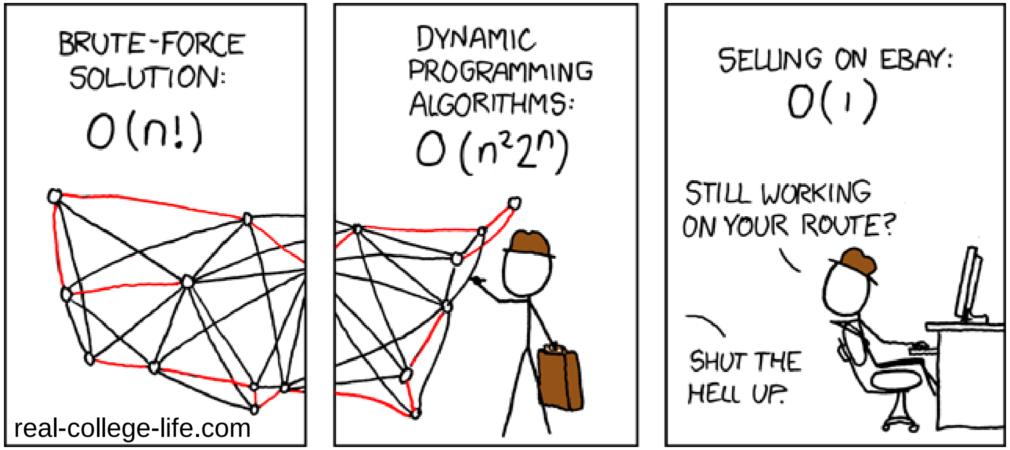
- Factoring

  Δίνεται σύνθετος αριθμός N, βρείτε την παραγοντοποίησή του.

  - Ενώ το Primality ανήκει στην κλάση **P**, το Factoring μάλλον όχι.
  - Είναι στην **NP** (γιατί;), αλλά μάλλον όχι **NP**-complete.
  - Το κρυπτοσύστημα RSA, και πολλά άλλα βασίζονται στη δυσκολία του Factoring.

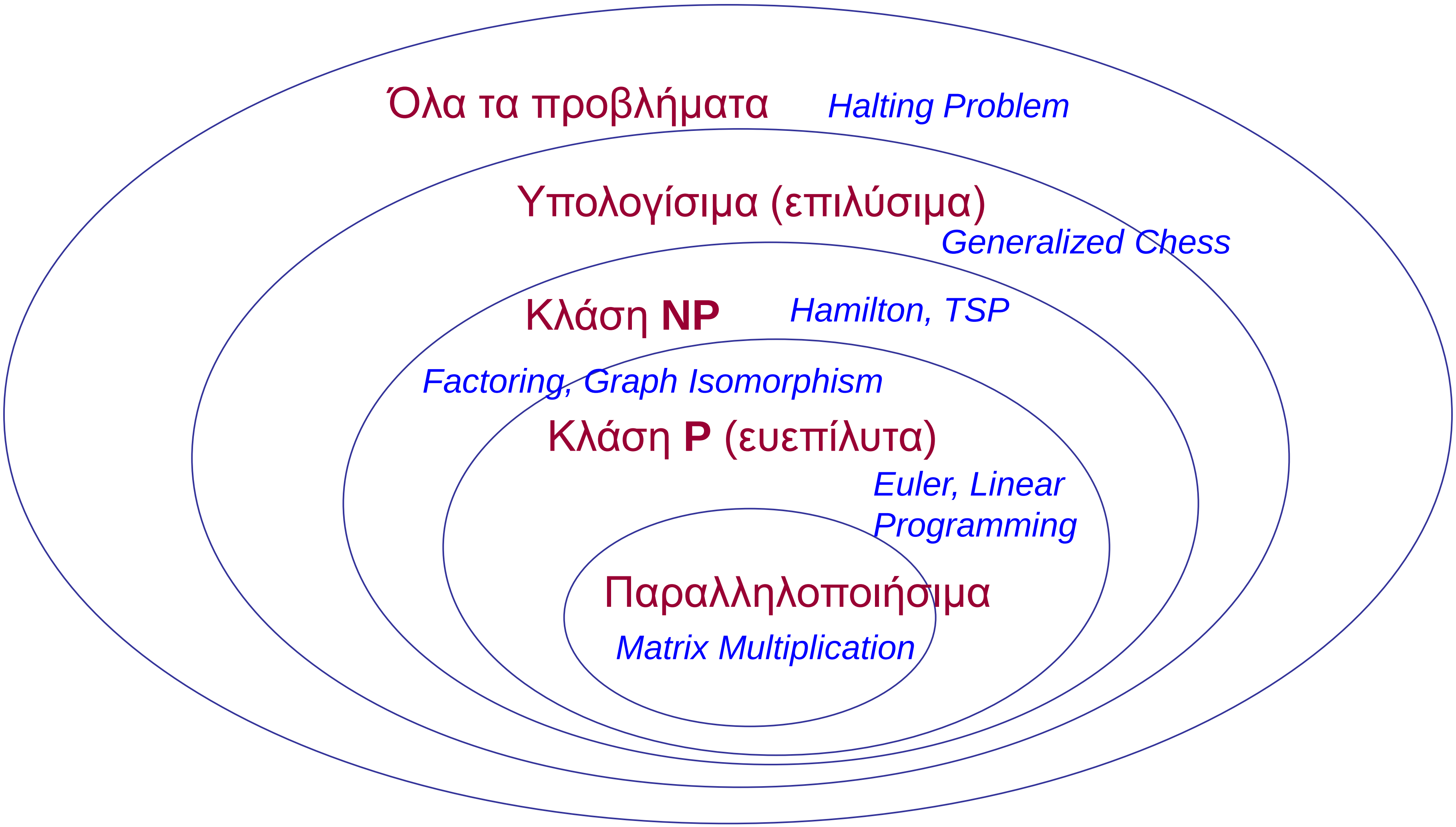
Γιατί ασχολούμαστε με την NP-πληρότητα;

- Γλιτώνουμε την απόλυση (...λέμε τώρα ☺)
- Στροφή σε πιο ρεαλιστικές λύσεις: ειδικές περιπτώσεις, προσεγγιστική επίλυση



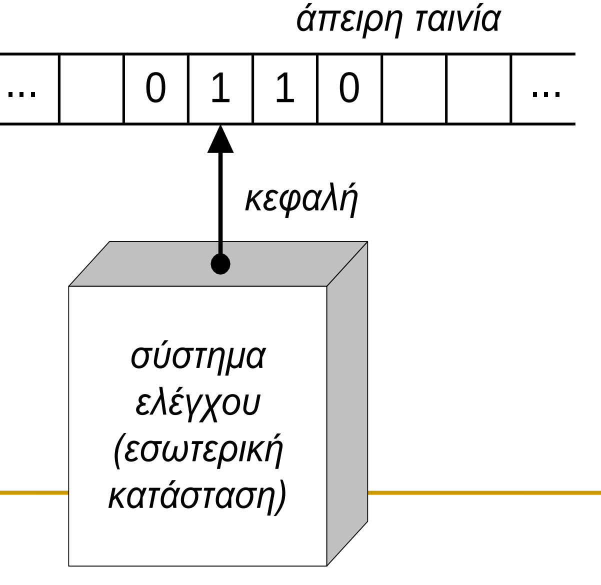
- Χρήση προς όφελός μας (κρυπτογραφία, εκλογές)

Κατηγοριοποίηση προβλημάτων



Πώς ξέρουμε ότι δεν κάνουμε λάθος;

- Μήπως υπάρχει πιο «έξυπνος» τρόπος υπολογισμού;
- Αυστηρός ορισμός αλγορίθμων με χρήση υπολογιστικών μοντέλων: Alan Turing, Alonzo 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.

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: 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, από ιδέα Gauss].
- 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.