Εισαγωγή στην Επιστήμη των Υπολογιστών

4ο εξάμηνο ΣΗΜΜΥ

4η ενότητα:

Αλγοριθμικές τεχνικές, αριθμητικοί υπολογισμοί

Επιμέλεια διαφανειών: Στάθης Ζάχος, Άρης Παγουρτζής

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

Αλγόριθμος

- Webster's 50 χρόνια πριν: ανύπαρκτος όρος
- Oxford's, 1971: «erroneous refashioning of *algorism:* calculation with Arabic numerals»
- Abu Jaffar Mohammed Ibn Musa Al-Khowarizmi, , 9ος αι. μ.Χ.
- 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, 13ος αι. μ.Χ.)

Αλγόριθμος (συν.)

- 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).

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 διαίρεσης `a div b`

- *Ο Ευκλείδειος αλγόριθμος είναι ο καλύτερος γνωστός αλγόριθμος για ΜΚΔ!*

- Ανοιχτό ερώτημα: είναι βέλτιστος;

Παράδειγμα εκτέλεσης του Ευκλείδειου αλγόριθμου

ΜΚΔ των 172 και 54

10	54
10	4
2	4
2	0

ΜΚΔ = 2

Αριθμοί Fibonacci

- 0, 1, 1, 2, 3, 5, 8, 13, 21,…

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

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

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

Τρίγωνο Pascal (Yang Hui)

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

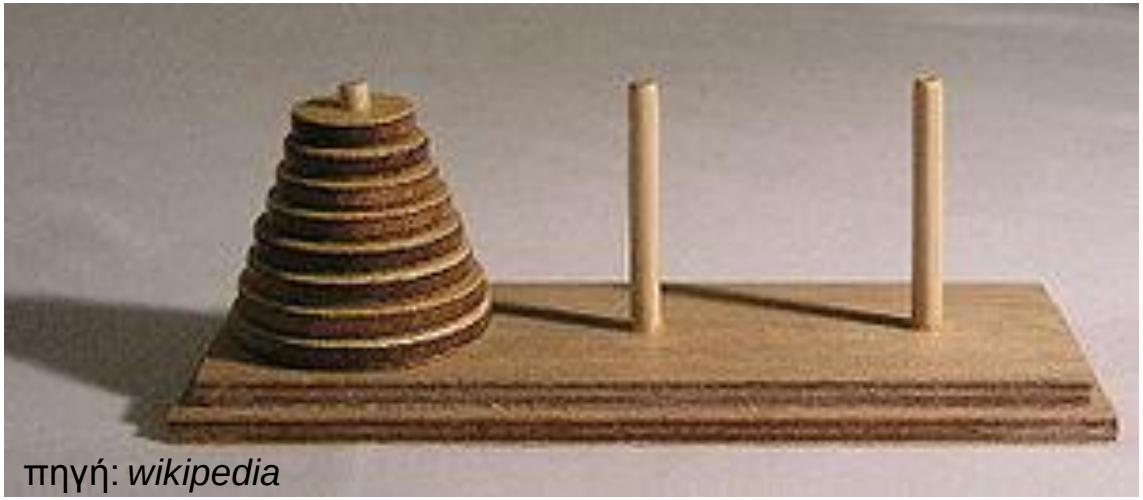
Διωνυμικοί συντελεστές / συνδυασμοί:

$$(a+b)^4 = a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4$$

Αλγοριθμικές τεχνικές

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

Πύργοι Ανόι (Hanoi Towers)



πηγή: *wikipedia*

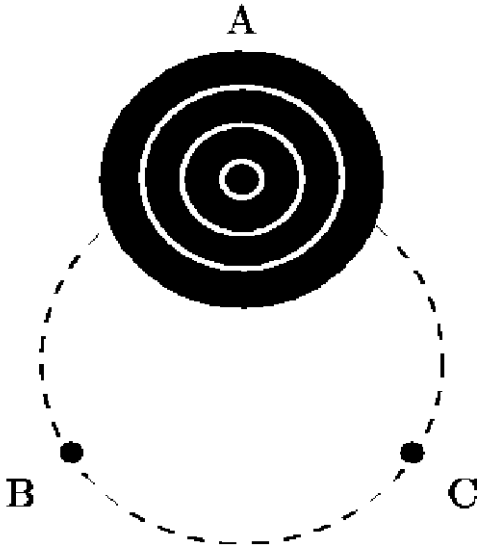
Πύργοι Ανόι (Hanoi Towers)

πηγή: *wikipedia*

Πύργοι Ανόι (Hanoi Towers): αναδρομή

```
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
```

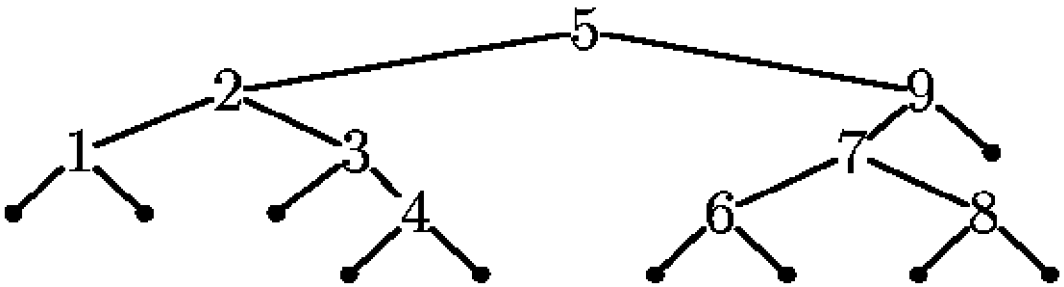
Πύργοι Ανόι (Hanoi Towers): επανάληψη



Επανάλαβε (μέχρι να επιτευχθεί η μετακίνηση):

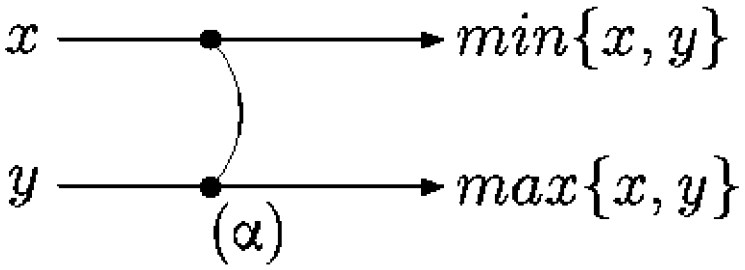
- Μετακίνησε κατά τη θετική φορά τον μικρότερο δίσκο
- Κάνε την μοναδική επιτρεπτή κίνηση που δεν αφορά τον μικρότερο δίσκο

Treesort με χρήση Binary Search Tree

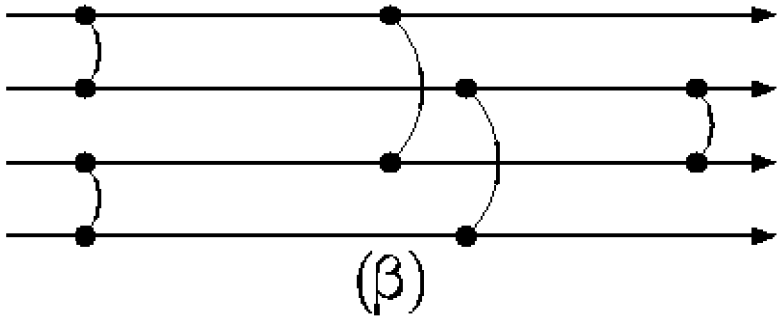



```
procedure inorder(t: treenode)
begin
    if t is not empty then
    begin
        inorder(left branch of t);
        write(element at t);
        inorder(right branch of t)
    end
end
```

Δίκτυα Ταξινόμησης (Sorting Networks)



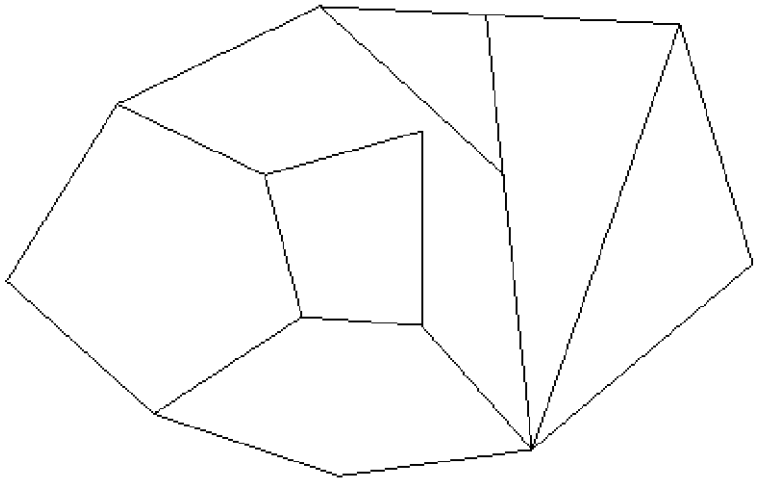
Συγκριτής



Δίκτυο ταξινόμησης 4 εισόδων

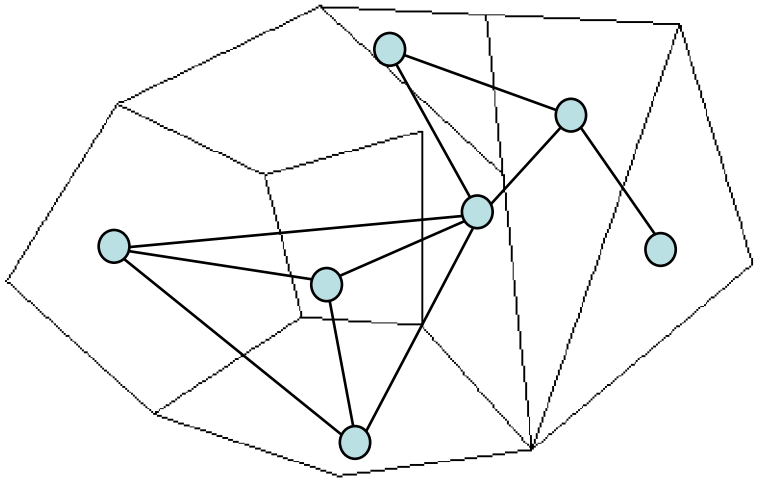
Four Color Theorem (1852-1977)

Πόσα χρώματα απαιτούνται για τον χρωματισμό όλων των χωρών, ώστε χώρες που συνορεύουν (με γραμμή για σύνορο) να έχουν διαφορετικό χρώμα;



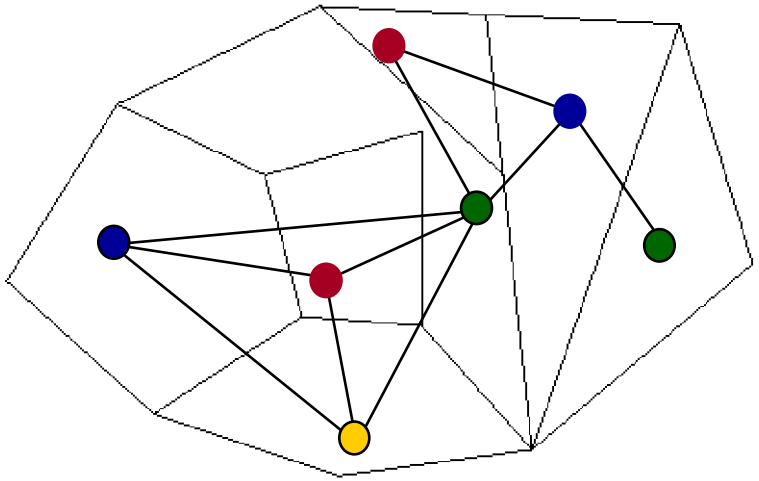
Four Color Theorem (1852-1977)

Πόσα χρώματα απαιτούνται για τον χρωματισμό όλων των χωρών, ώστε χώρες που συνορεύουν (με γραμμή για σύνορο) να έχουν διαφορετικό χρώμα;



Four Color Theorem (1852-1977)

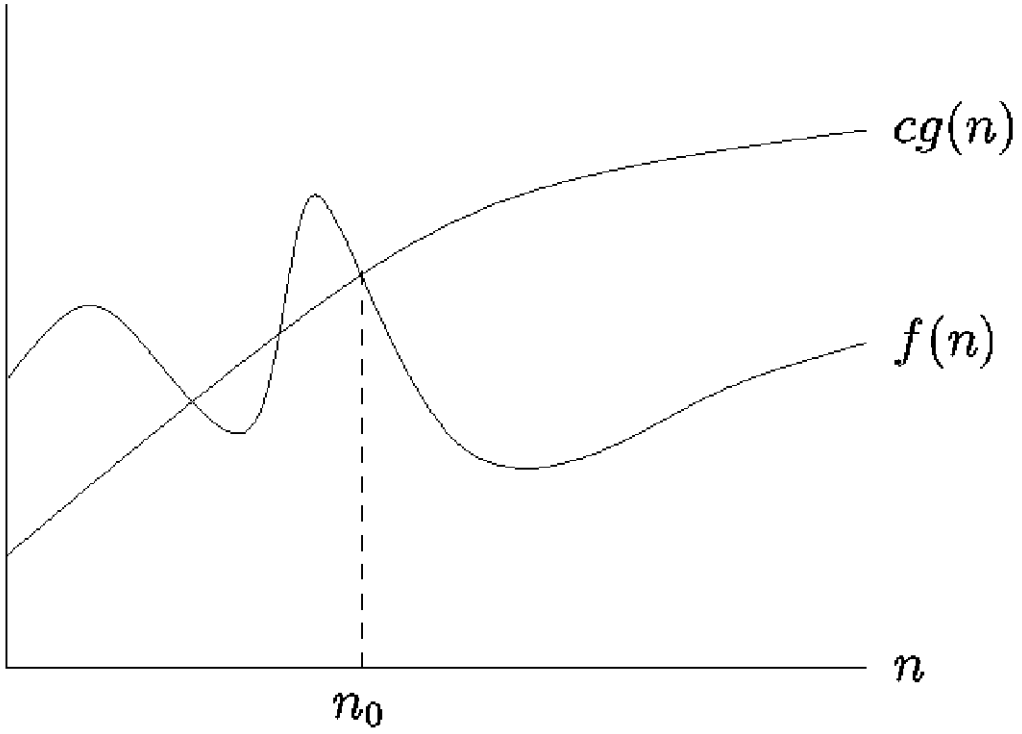
Πόσα χρώματα απαιτούνται για τον χρωματισμό όλων των χωρών, ώστε χώρες που συνορεύουν (με γραμμή για σύνορο) να έχουν διαφορετικό χρώμα;



Appel - Haken

(απόδειξη με πρόγραμμα!)

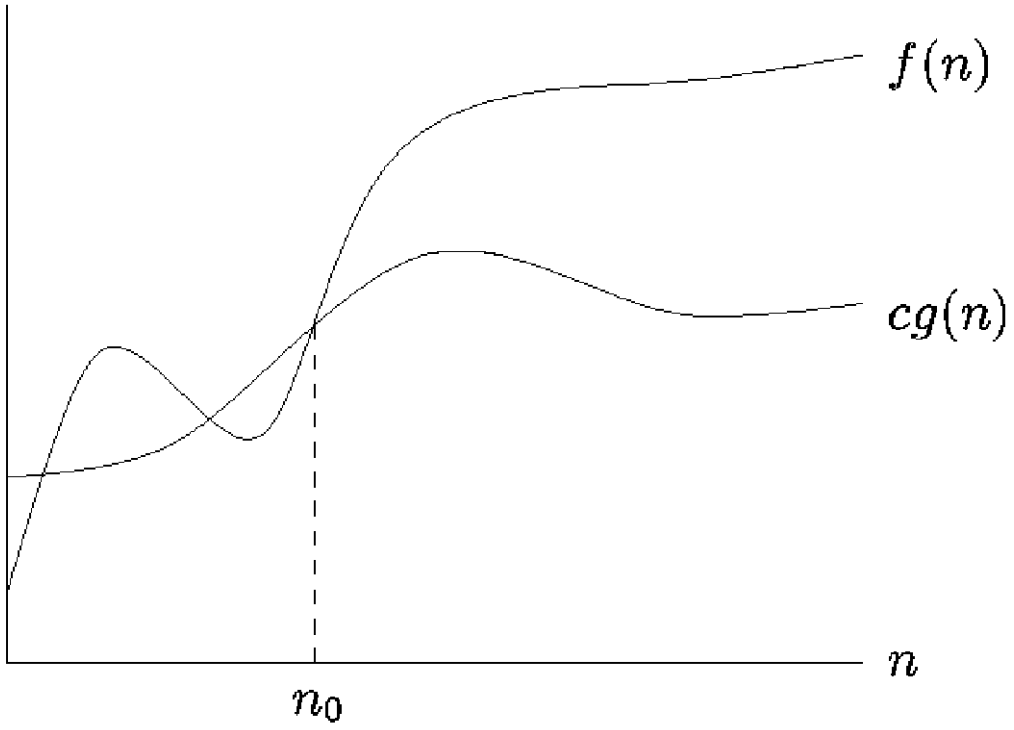
Μαθηματικοί συμβολισμοί (i)



Σχήμα 2.3: $f = O(g)$

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

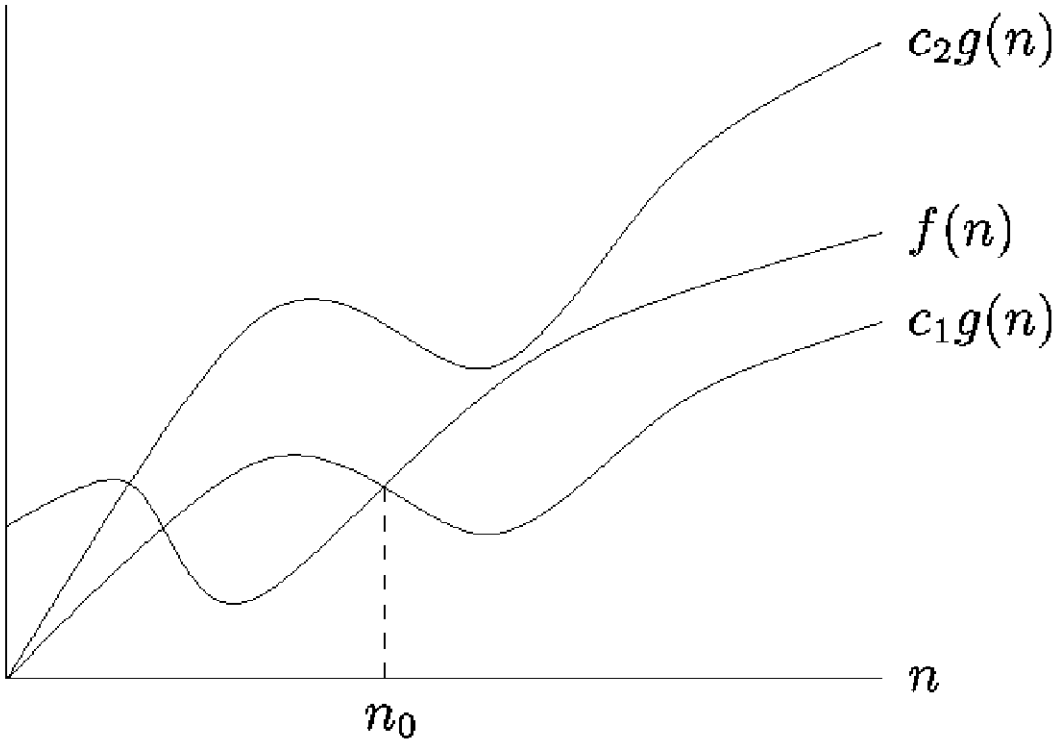
Μαθηματικοί συμβολισμοί (ii)



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

Συχνά γράφουμε (καταχρηστικά)
$g(n)=O(f(n))$ αντί για $g(n) \in O(f(n))$

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

$p(n) = \Theta(n^k)$, για κάθε πολυώνυμο p

$O(poly) = \bigcup O(n^k)$ (για όλα τα $k \in N$)

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.

Μαθηματικοί συμβολισμοί: ιδιότητες

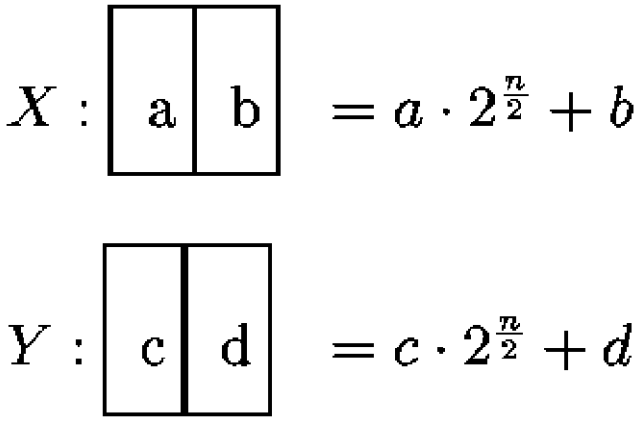
Θεώρημα. $\log(n!) = \Theta(n \log n)$

Απόδειξη: *ασυμπτωτικά* ισχύει $(n/2)^{n/2} < n! < n^n$

=> $1/2\ n\ (\log n - 1) < \log(n!) < n \log n$

=> $1/3\ n \log n < \log(n!) < n \log n$

Πολλαπλασιασμός Ακεραίων



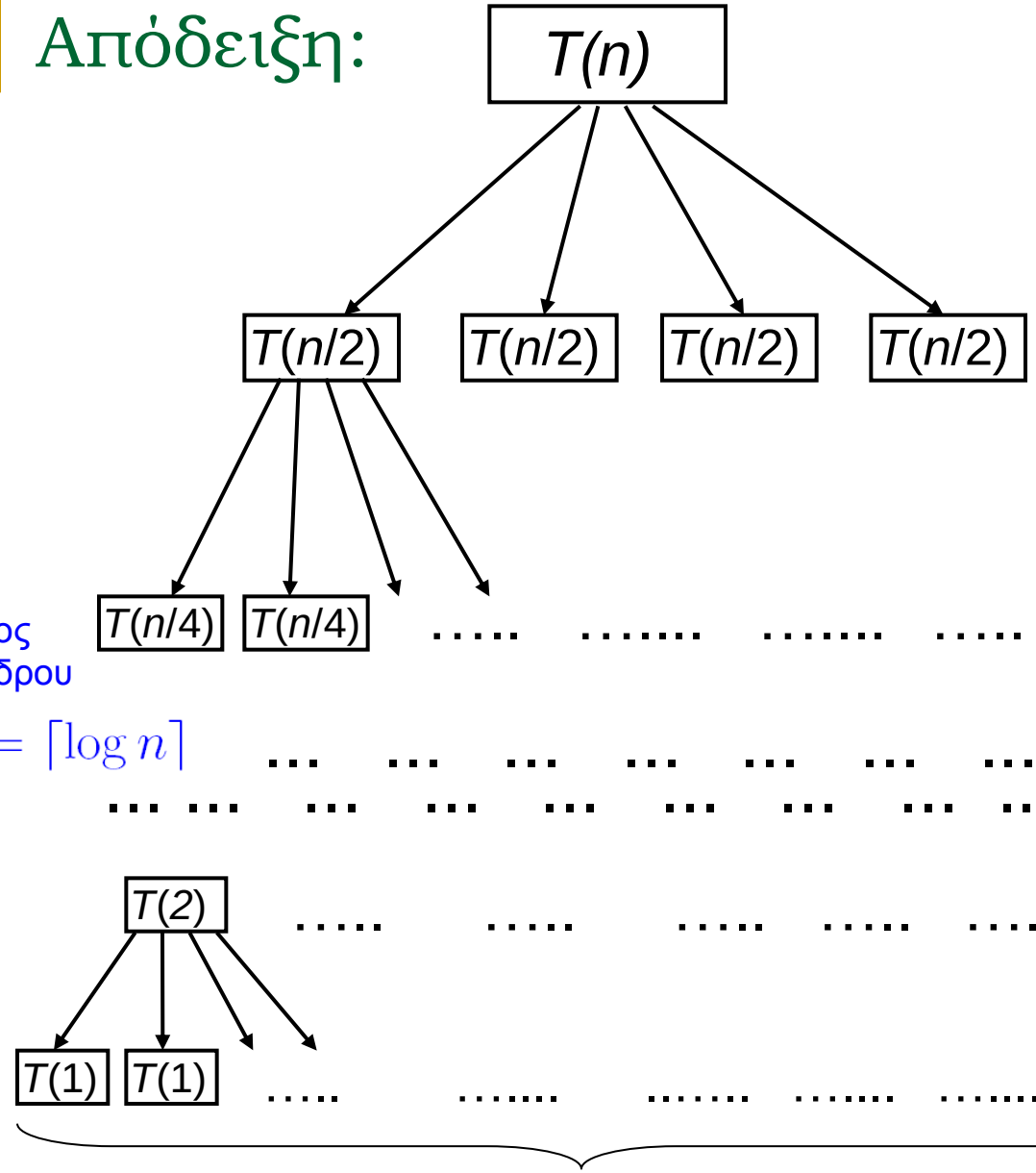
$$X\ Y = ac \cdot 2^n + (ad + bc) \cdot 2^{\frac{n}{2}} + bd$$

Πολυπλοκότητα Πολλαπλασιασμού

$$T(n) = \begin{cases} a & , \text{για } n = 1 \\ 4T(\frac{n}{2}) + cn & , \text{για } n > 1 \end{cases}$$

$$T(n) = O(n^2)$$

Απόδειξη:


T(n)
T(n/2)
T(n/2)
T(n/2)
T(n/2)
T(n/4)
T(n/4)
T(2)
T(1)
T(1)
+ cn
+ 4 c(n/2)
+ 16 c(n/4)
+ 4^(k-1) c(n/2^(k-1))
Χρον. πολ/τα
<(4/2)^k cn
≤ 2^(logn+1) cn
= O(n^2)
Ύψος δένδρου
k = ⌈log n⌉
Συνολικά: O(n^2)
4^k 'φύλλα', χρονική πολυπλ/τα α·4^k = O(n^2)

Βελτιωμένος Πολλαπλασιασμός (Gauss-Karatsuba)

$$(ad + bc) \;=\; [(a - b)(d - c) + ac + bd]$$

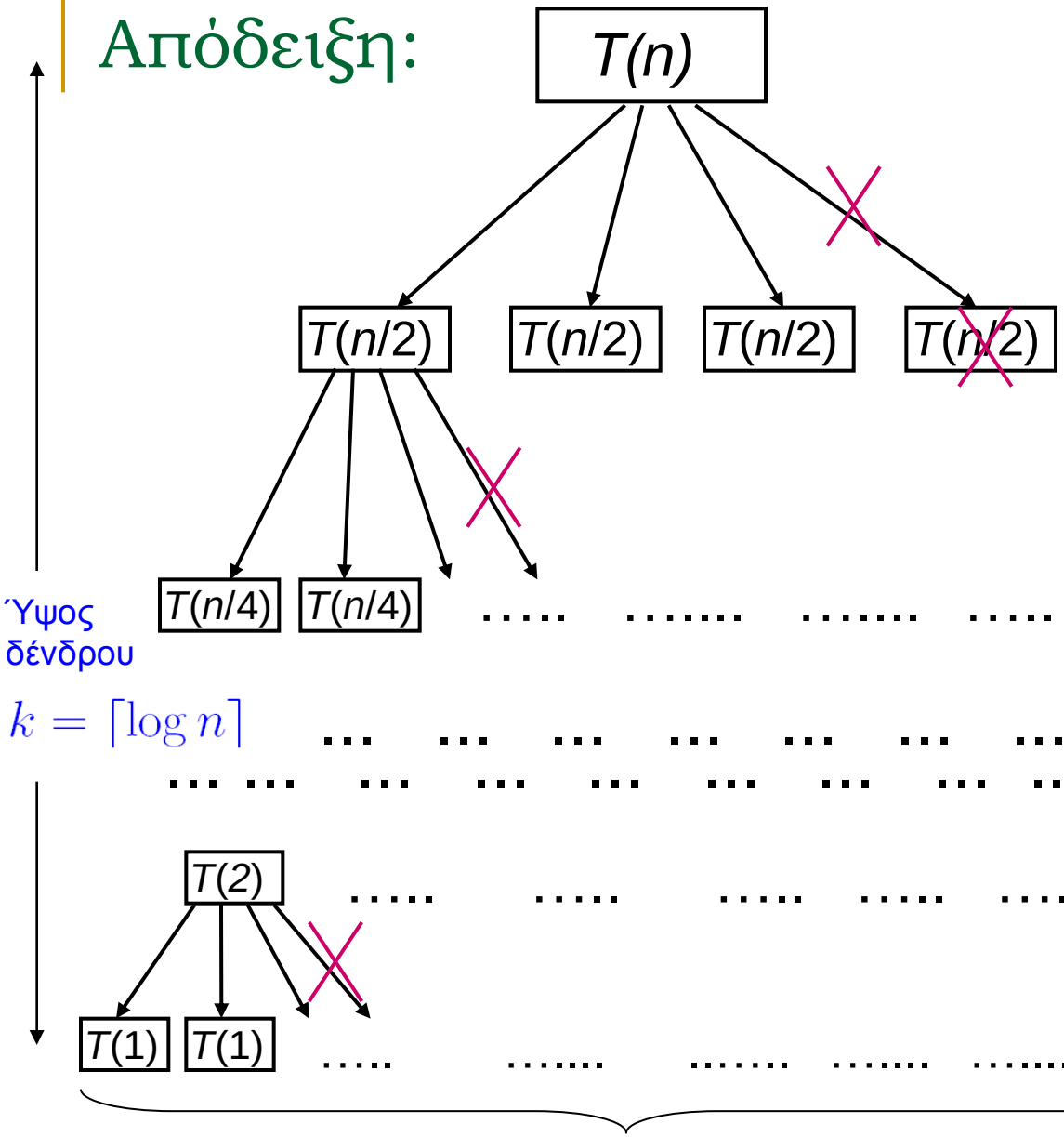
$$X\ Y = ac \cdot 2^n + [(a - b)(d - c) + ac + bd]\ 2^{\frac{n}{2}} + bd$$

Πολυπλοκότητα Βελτίωσης

$$T(n) = \begin{cases} a & , \text{για } n = 1 \\ 3T(\frac{n}{2}) + cn & , \text{για } n > 1 \end{cases}$$

$$T(n) = O(n^{\log_2 3}) = O(n^{1.59})$$

Απόδειξη:



3^k ‘φύλλα’, χρονική πολυπλ/τα $\alpha \cdot 3^k = O(3^{\log n}) = O(n^{\log 3})$

Master Theorem (απλή μορφή)

Αν $T(n) = aT(n/b) + O(n)$,

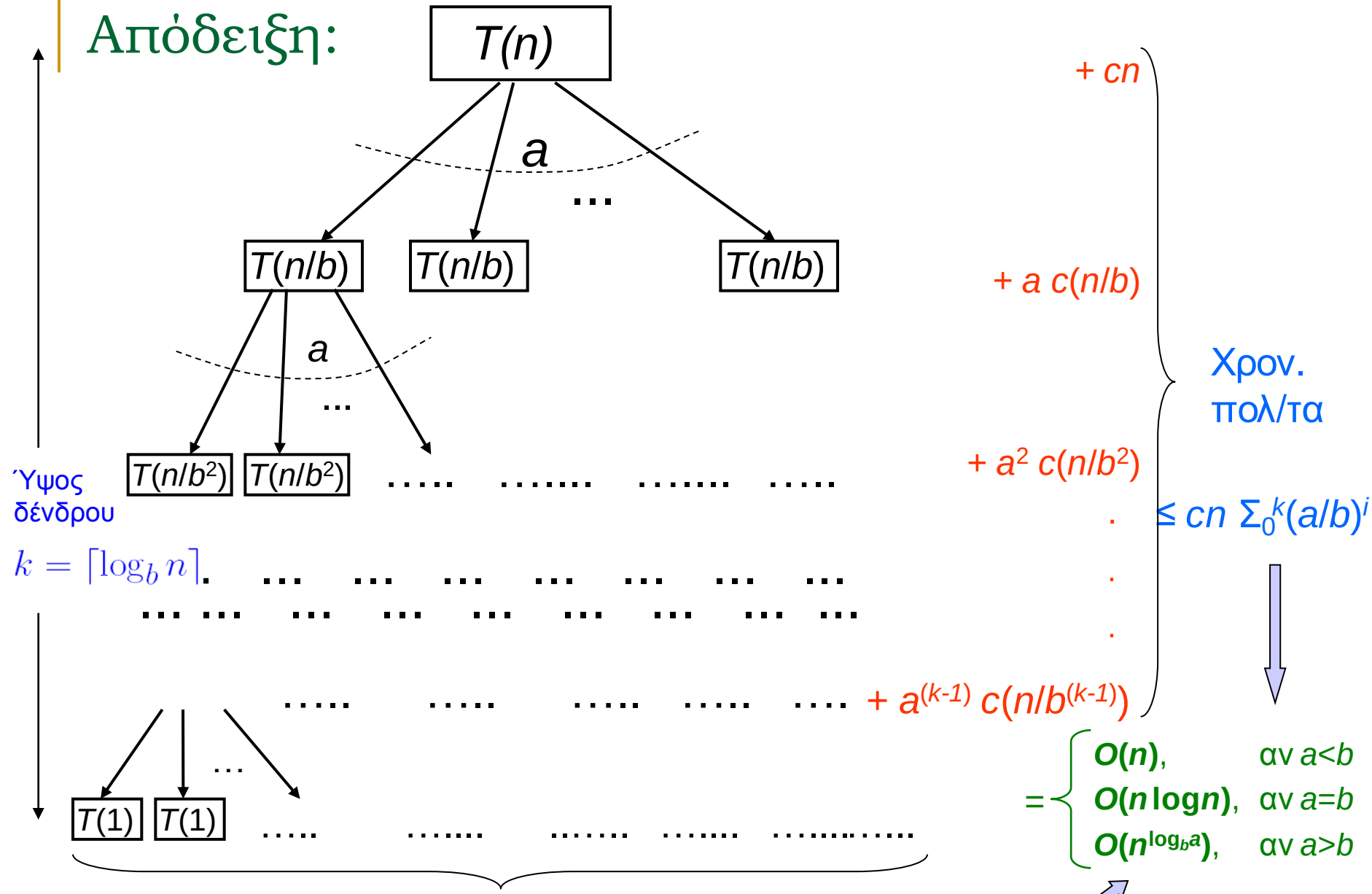
για θετικούς ακέραιους a, b

και $T(1) = O(1)$

τότε:

$$T(n) = \begin{cases} O(n), & \text{αν } a<b \\ O(n \log n), & \text{αν } a=b \\ O(n^{\log_b a}), & \text{αν } a>b \end{cases}$$

Απόδειξη:



a^k 'φύλλα', χρονική πολυπλ/τα $c' \cdot a^k = O(a^{\log_b n}) = O(n^{\log_b a}) \leq$

Master Theorem (γενική μορφή)

Αν $T(n) = aT(n/b) + O(n^d)$,

για θετικούς ακέραιους a, b, d

και $T(1) = O(1)$

τότε:

$$T(n) = \begin{cases} O(n^d), & \text{αν } a<b^d \\ O(n^d \log n), & \text{αν } a=b^d \\ O(n^{\log_b a}), & \text{αν } a>b^d \end{cases}$$

Master Theorem: εφαρμογή

Αν $T(n) = aT(n/b) + O(n^d)$, για θετικούς ακέραιους a, b, d

και $T(1) = O(1)$ τότε:

$$T(n) = \begin{cases} O(n^d), & \text{αν } a<b^d \\ O(n^d \log n), & \text{αν } a=b^d \\ O(n^{\log_b a}), & \text{αν } a>b^d \end{cases}$$

Matrix Multiplication

'Standard' divide-and-conquer: $T(n) = 8T(n/2) + O(n^2)$

$\Rightarrow T(n) = O(n^3)$

Strassen's algorithm: $T(n) = 7T(n/2) + O(n^2)$

$\Rightarrow T(n) = O(n^{\log 7})$

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 $(a$ **mod** $z \neq 0)$ **or** $(a$ **mod** $z \neq 0)$ **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** $i := i$ **mod** j **else** $j := j$ **mod** $i;$
return $(i + j)$

Εύρεση Μέγιστου Κοινού Διαιρέτη (gcd): υλοποίηση με αναδρομή

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

if a=b **then** GCD(a,b):=a
else if a>b **then** GCD(a,b):= GCD(a-b, b)
else GCD(a,b):= GCD(a, b-a)

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

if b=0 **then** GCD(a,b):= a
else GCD(a,b):= GCD(b, a **mod** b)

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*).
- Άρα η πολυπλοκότητα του Ευκλείδειου είναι $\Theta(\log \max(a,b)) = \Theta(\log (a+b))$

Επεκτεταμένος Ευκλείδειος Αλγόριθμος

- Εκφράζει τον gcd(*a*,*b*) σαν γραμμικό συνδυασμό των *a* και *b*
- Επιτρέπει την εύρεση πολλαπλασιαστικού αντιστρόφου στην αριθμητική modulo n:

  αν gcd(*a*,*n*)=1 τότε Ext. Euclid δίνει *κ*,*λ*: *κa*+*λn*=1

- Άσκηση: *σχεδιάστε και υλοποιήστε τον επεκτεταμένο Ευκλείδειο αλγόριθμο*

Ύψωση σε δύναμη

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

Πολυπλοκότητα: $O(n)$ – εκθετική! (γιατί;)

… 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

```
Fib1 (n)
    if (n<2) then return n
    else return Fib1(n-1)+Fib1(n-2)
```

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

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

Αριθμοί Fibonacci – καλύτερος αλγόριθμος

```
Fib2(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

Πρώτοι αριθμοί και κρυπτογραφία

Υπολογιστικά προβλήματα σημαντικά για κρυπτογραφία:

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

Κρυπτογραφία δημοσίου κλειδιού

- Συναρτήσεις μονής κατεύθυνσης (one-way functions): *εύκολο να υπολογιστούν δύσκολο να αντιστραφούν*
- Κρυπτογραφία δημοσίου κλειδιού (*κατάργησε την ανάγκη ανταλλαγής κλειδιών!*): στηρίζεται στην ύπαρξη τέτοιων συναρτήσεων.
  - Κρυπτοσύστημα RSA [Rivest-Shamir-Adleman, 1977] συνάρτηση κρυπτογράφησης: $c = m^e \bmod n$

    Ασφάλεια RSA: δεν υπάρχει (*ελπίζουμε*, χρειαζόμαστε απόδειξη!) αποδοτικός τρόπος υπολογισμού του *m* δεδομένων των *c*, *e*, και *n*, αν *n* είναι σύνθετος *(...εκτός αν γνωρίζουμε παραγοντοποίηση του n)*

    Δηλαδή *η συνάρτηση κρυπτογράφησης RSA είναι μονής κατεύθυνσης* (είναι;)

Κρυπτοσύστημα RSA (i)

- Για να στείλει η Α (Alice) στον Β (Bob) ένα μήνυμα m:
- Ο Β διαλέγει 2 μεγάλους πρώτους αριθμούς p και q, υπολογίζει το γινόμενο $n = pq$, και διαλέγει επίσης ακέραιο e σχετικά πρώτο με το $\varphi(n) = (p-1)(q-1)$.
- Ο Β στέλνει στην Α τα n και e (δημόσιο κλειδί του Β)
- Η Α στέλνει στον Β την τιμή $c = m^e \bmod n$ (κρυπτογράφημα).
- Ο Β υπολογίζει $m = c^d \bmod n$
  όπου $d = e^{-1} \pmod{\varphi(n)}$ <=> $de = 1 \pmod{\varphi(n)}$

  (ο d είναι το ιδιωτικό κλειδί του Β)

Κρυπτοσύστημα RSA (ii)

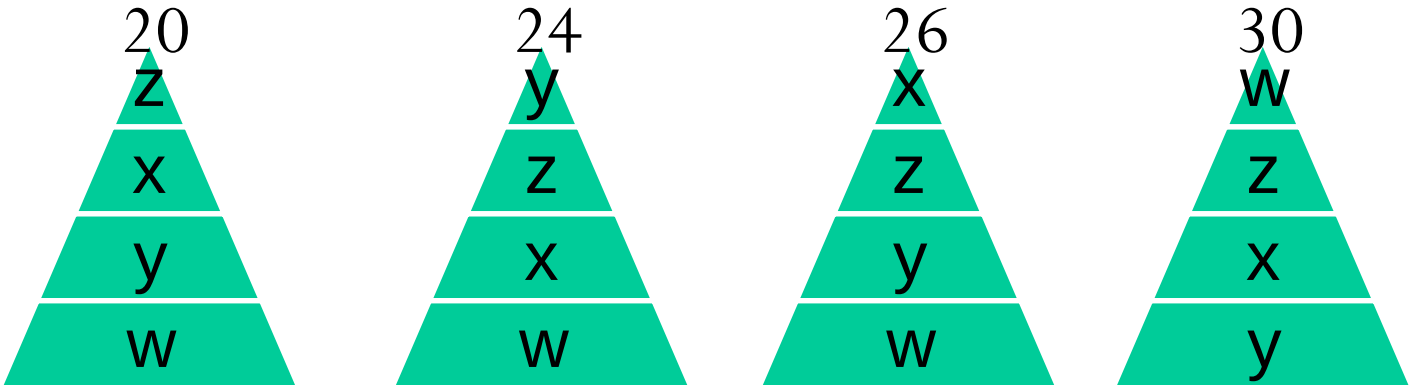
- Ορθότητα RSA: $c^d = m^{ed} = m^{k\varphi(n)+1} = m \pmod{n}$.
- Παράδειγμα συστήματος RSA: p=11, q=17, n=187, e=21, d=61, m=42, c=9
- Η ασφάλεια του RSA στηρίζεται στην (*εκτιμώμενη*, δεν υπάρχει ακόμη απόδειξη!) υπολογιστική πολυπλοκότητα της παραγοντοποίησης (factoring).
- Η λειτουργία του RSA στηρίζεται σε *αποδοτικούς αλγόριθμους* για: primality testing (Miller-Rabin), ύψωση σε δύναμη modulo n (επαναλαμβανόμενος τετραγωνισμός) και εύρεση αντιστρόφου modulo $\varphi(n)$ (επεκτεταμένος Ευκλείδειος).

Ψηφοφορίες (social choice)

- Εκλογές (βουλευτικές, πρυτανικές ;-))
- Λήψη αποφάσεων σε εταιρείες, οργανισμούς,...
- Χρήση στον παγκόσμιο ιστό:
  - ιστοσελίδες "ψηφίζουν" ιστοσελίδες δείχνοντας σε αυτές
  - χρήστες "ψηφίζουν" ιστοσελίδες ανάλογα με τον χρόνο που ξοδεύουν σε αυτές
- Κοινωνικά δίκτυα (social networks)
  - friends, followers

Ψηφοφορίες (social choice)

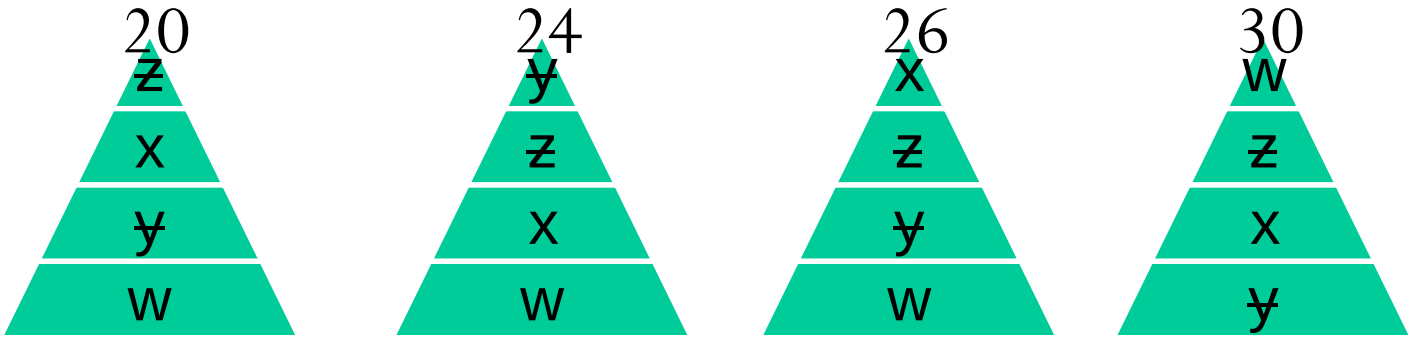
- Πληθώρα εκλογικών συστημάτων
- Το πλειοψηφικό δεν είναι πάντα δίκαιο:



  - Ο νικητής υποστηρίζεται μόνο από το 30%
  - Είναι τελευταία προτίμηση για το 70% !

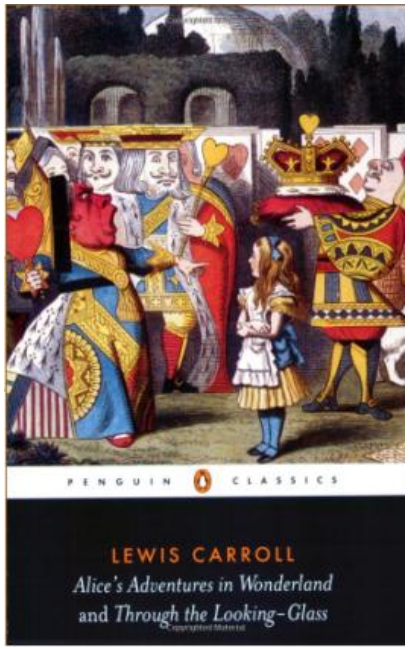
Ψηφοφορίες: κι άλλα παράδοξα

- Και το "απόλυτο" πλειοψηφικό παρουσιάζει παράδοξα:
  - οι x, w περνούν στον 2ο γύρο
  - ο x κερδίζει με 70%, παρ'όλο που 74% προτιμούν τον z από τον x (ο z έφυγε από τον 1ο γύρο!).



Ψηφοφορίες: υπολογιστικές προκλήσεις

- Δικαιότερα συστήματα μπορεί να απαιτούν πολύ μεγάλο χρόνο υπολογισμού του νικητή (υπολογιστικά απρόσιτο)
  - για το σύστημα του Dodgson (γνωστός και ως Lewis Caroll, 19ος αιώνας) το πρόβλημα είναι πλήρες για μια κλάση πολυπλοκότητας ευρύτερη της NP
- Θέλουμε ο υπολογισμός του νικητή να είναι υπολογιστικά προσιτός



Ψηφοφορίες: υπολογιστικές προκλήσεις

- Θεώρημα Gibbard- Satterthwaite (1973):

  «Πέρα από κάποιες τετριμμένες περιπτώσεις, όλα τα συστήματα ψηφοφορίας είναι χειραγωγήσιμα (εκτός αν είναι δικτατορικά)!»

- Θέλουμε η χειραγώγηση να είναι δύσκολη (υπολογιστικά απρόσιτη)

Μη συνεργατικά παίγνια

- Παίκτες (agents: χρήστες, οντότητες λογισμικού, συστήματα) ανταγωνίζονται, συνήθως για διεκδίκηση πόρων
- Κάθε παίκτης αποφασίζει μόνο τη δική του στρατηγική
  - στόχος: ελαχιστοποίηση ατομικού κόστους
- Το ατομικό κόστος εξαρτάται από τις στρατηγικές όλων

Μη συνεργατικά παίγνια

- Ισορροπία Nash: κανείς δεν βελτιώνει το ατομικό του κόστος αλλάζοντας μόνο τη δική του στρατηγική.
  - Nash (1952): απέδειξε ότι πάντα υπάρχει τέτοια ισορροπία (αλλά μπορεί να είναι μεικτή – mixed).
  - Η ισορροπία Nash αποτελεί «λύση» του συστήματος:

    αν οι παίκτες συμπεριφερθούν στρατηγικά και λογικά και έχουν στη διάθεσή τους πλήρη γνώση και επαρκή χρόνο, τότε καταλήγουν σε μία ισορροπία Nash.

Ισορροπία Nash

•Δίλημμα φυλακισμένων: συλλαμβάνονται δύο διαρρήκτες, συνεργάτες σε μεγάλη κλοπή. Κρατούνται σε χωριστά κελιά χωρίς επικοινωνία

	Ομολογεί Β	Δεν ομολογεί Β
Ομολογεί Α	**5, 5**	**0, 15**
Δεν ομολογεί Α	**15, 0**	**1, 1**

- Αποτέλεσμα: αμφότεροι ομολογούν!
- Ισορροπία Nash δεν βελτιστοποιεί συνολικό αποτέλεσμα

Ισορροπία Nash: ερωτήματα

•Τίμημα αναρχίας: πόσο "άσχημα" μπορεί να συμπεριφερθεί το σύστημα; Λόγος συνολικού κόστους χειρότερης ισορροπίας προς βέλτιστη συνεργατική λύση

[Koutsoupias, Papadimitriou, 1999]

•Μπορούμε να βρούμε την "χειρότερη" ισορροπία; Οποιαδήποτε ισορροπία; σύμφωνα με ισχυρές ενδείξεις δυσεπίλυτο πρόβλημα: *πλήρες* για την κλάση PPAD.

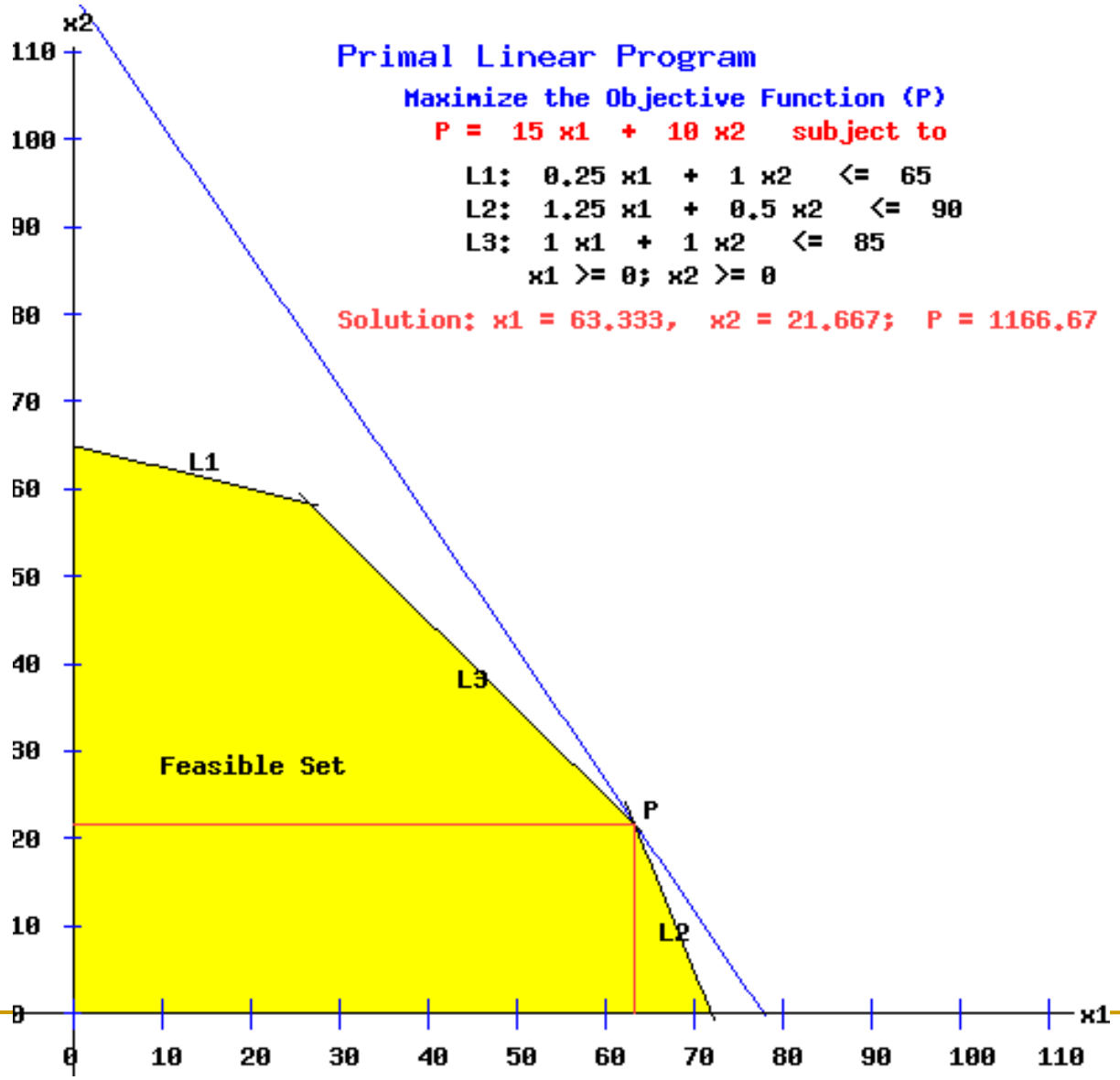
[Daskalakis, Goldberg, Papadimitriou, 2005]

[Chen, Deng, 2005]

•*«If your laptop can't find it, neither can the market!»*

- Kamal Jain (Microsoft Research)

Linear Programming



Επιτυχίες-σταθμοί Θεωρίας Αλγορίθμων και Πολυπλοκότητας

- Linear Programming
  [Dantzig - von Neumann, 1947, Khachiyan, 1979, Karmakar, 1984]
- Fast Fourier Transform
  [Cooley-Tukey, 1965 (αλλά και Gauss, 1805)]
- NP-πληρότητα [Cook-Karp, 1971-72]:
  αδυναμία αποδοτικής επίλυσης πολλών σημαντικών προβλημάτων
- Κρυπτογραφία δημοσίου κλειδιού
  [Diffie-Hellman, Rivest-Shamir-Adleman, 1976-77]

Επιτυχίες-σταθμοί Θεωρίας Αλγορίθμων και Πολυπλοκότητας

- Pagerank (Google)
  [Page-Brin-Motwani-Winograd, 1995-99]
- Κβαντικοί υπολογισμοί [Shor, 1996]:
  παραγοντοποίηση σε πολυωνυμικό χρόνο
- Θεώρημα PCP, μη-προσεγγισιμότητα
  [Arora-Feige-Goldwasser-Lund-Lovasz-Motwani-Safra-Sudan-Szegedy, 1992-98]
- Δυσκολία υπολογισμού ισορροπιών Nash
  [Goldberg-Daskalakis-Papadimitriou, Chen-Deng, 2005]

Συμπεράσματα

- Πολλά σύγχρονα συστήματα στηρίζονται στην ταχύτητα υπολογισμών που επιτυγχάνεται μέσω αποδοτικών αλγορίθμων.
- Η υπολογιστική δυσκολία ορισμένων προβλημάτων (π.χ. factoring) μπορεί να είναι επιθυμητή (κρυπτογραφία, εκλογές).
- Τα μαθηματικά είναι πάντα επίκαιρα!