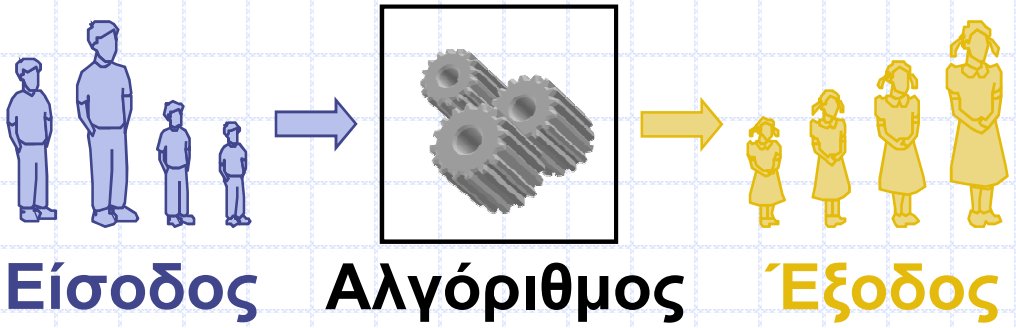
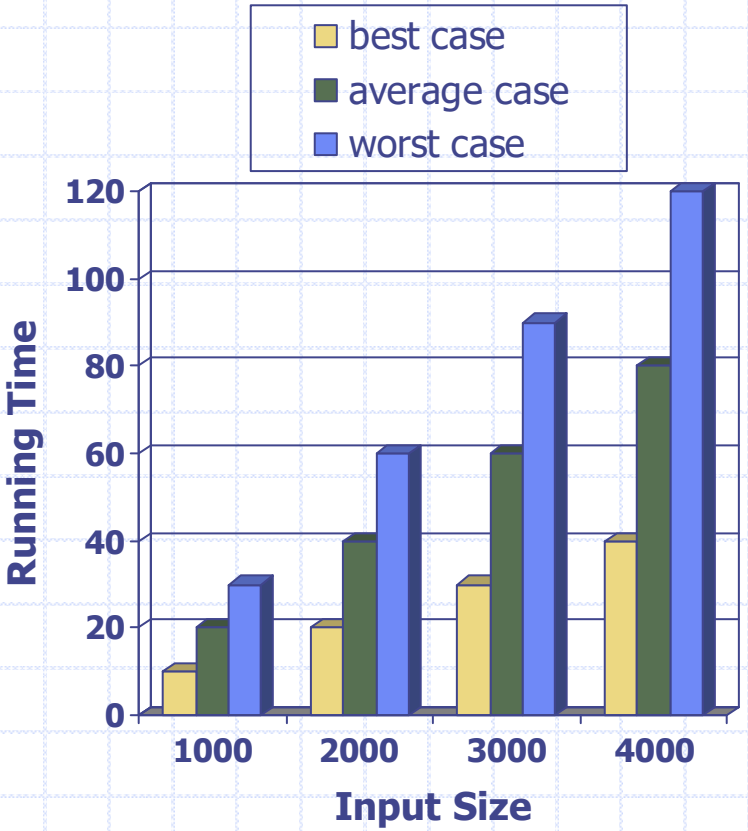
Ανάλυση Αλγορίθμων



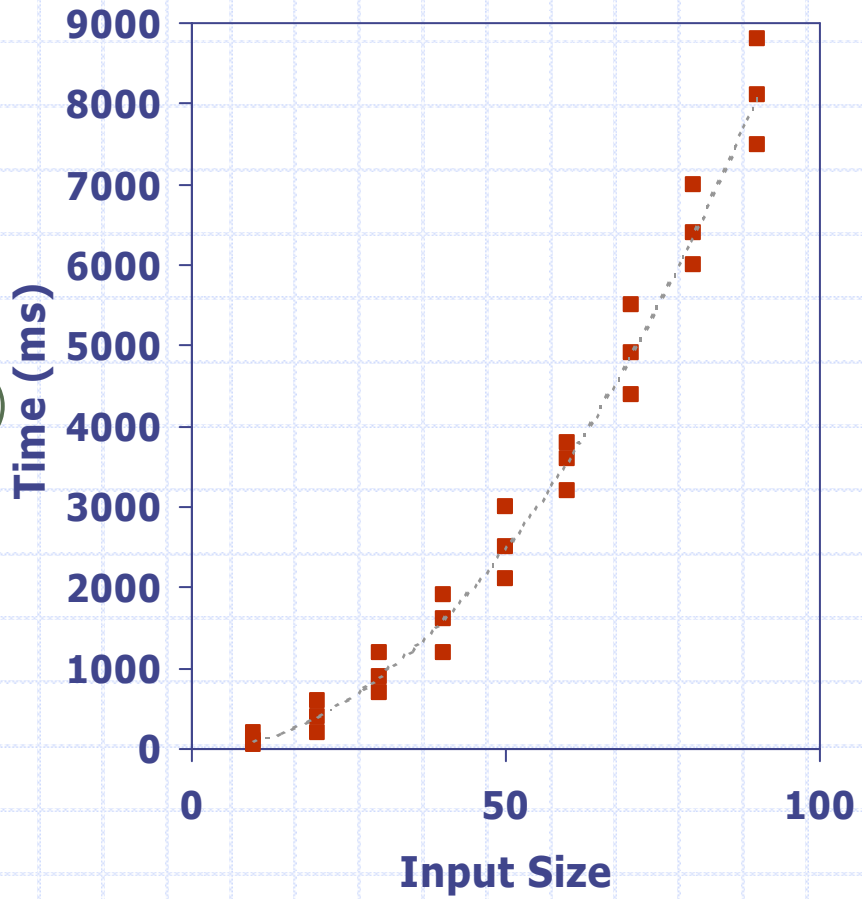
Χρόνος Τρεξίματος

- Οι περισσότεροι αλγόριθμοι μετασχηματίζουν αντικείμενα εισόδου σε αντικείμενα εξόδου
- Τυπικά ο απαιτούμενος χρόνος αυξάνει με το μέγεθος της εισόδου
- Συνήθως είναι δύσκολο να υπολογισθεί μέσος χρόνος τρεξίματος.
- Εστιάζουμε στη χειρότερη περίπτωση.
  - Ευκολότερη η Ανάλυση
  - Σημαντική για εφαρμογές όπως οικονομικές, παίγνια, και ρομποτική



Πειραματικές Μελέτες

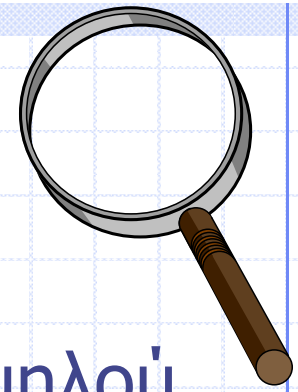
- Γράψτε ένα πρόγραμμα που να υλοποιεί τον αλγόριθμο
- Τρέξτε το πρόγραμμα με εισόδους μεταβλητού μεγέθους και σύνθεσης
- Χρησιμοποιήστε μια μέθοδο όπως την System.currentTimeMillis() για να έχετε ακριβή στοιχεία του χρόνου τρεξίματος
- Πάρτε γραφική παράσταση των αποτελεσμάτων



Περιορισμοί των Πειραμάτων

- Η υλοποίηση του αλγόριθμου μπορεί να είναι δύσκολη
- Τα αποτελέσματα μπορεί να μην είναι ενδεικτικά του χρόνου τρεξίματος για άλλες εισόδους που δεν συμπεριλαμβάνονται στο πείραμα
- Για τη σύγκριση δύο αλγορίθμων πρέπει να χρησιμοποιηθεί το ίδιο περιβάλλον λογισμικού και υλικού.

Θεωρητική Ανάλυση



- Αντί για υλοποίηση χρησιμοποιεί μια υψηλού επιπέδου περιγραφή του αλγόριθμου
- Εκφράζει το χρόνο τρεξίματος σαν συνάρτηση του μεγέθους της εισόδου n.
- Λαμβάνει υπόψη όλες τις πιθανές εισόδους
- Μας επιτρέπει να αξιολογούμε την ταχύτητα ενός αλγορίθμου ανεξάρτητα από το περιβάλλον υλικού/λογισμικού

Ψευδοκώδικας

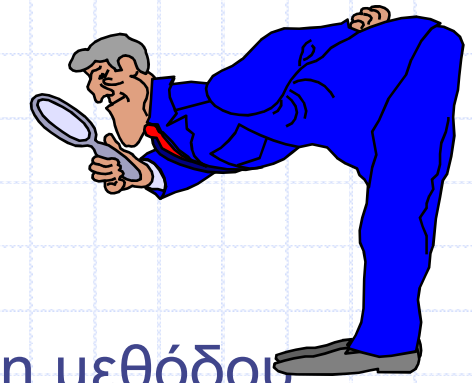
- Περιγραφή ενός αλγόριθμου σε υψηλό επίπεδο
- Πιο δομημένος τρόπος από τη φυσική γλώσσα
- Λιγότερο λεπτομερές από ένα πρόγραμμα
- Συμβολισμός που προτιμείται για την περιγραφή αλγορίθμων
- Αποκρύπτει θέματα σχεδιασμού προγραμμάτων

Παράδειγμα: να βρεθεί το μέγαλύτερο στοιχείο ενός πίνακα

```
Algorithm arrayMax(A, n)
    Input array A of n integers
    Output maximum element of A

    currentMax ← A[0]
    for i ← 1 to n – 1 do
        if A[i] > currentMax then
            currentMax ← A[i]
    return currentMax
```

Λεπτομέρειες Ψευδοκώδικα



- Έλεγχος Ροής
  - **if** … **then** … [**else** …]
  - **while** … **do** …
  - **repeat** … **until** …
  - **for** … **do** …
  - Αντί για αγκύλες έχει χρησιμοποιηθεί στοίχιση
- Δήλωση Μεθόδου

  Algorithm ***method*** (***arg*** [, ***arg***…])

  Input …

  Output …
- Κλήση μεθόδου

  var.method (***arg*** [, ***arg***…])
- Επιστρεφόμενη τιμή

  return ***expression***
- Εκφράσεις

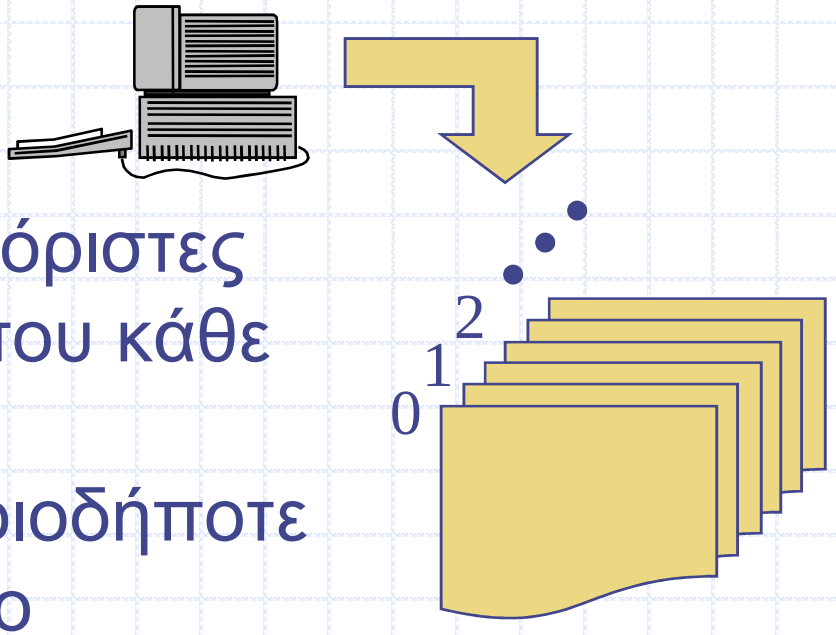
  ← Assignment (όπως το = στην Java)

  = Έλεγχος ισότητας (όπως το == στην Java)

  n^2 Επιτρέπονται δυνάμεις και άλλοι μαθηματικοί συμβολισμοί

Το μοντέλο μηχανής τυχαίας προσπέλασης (RAM)

- Μια **CPU**

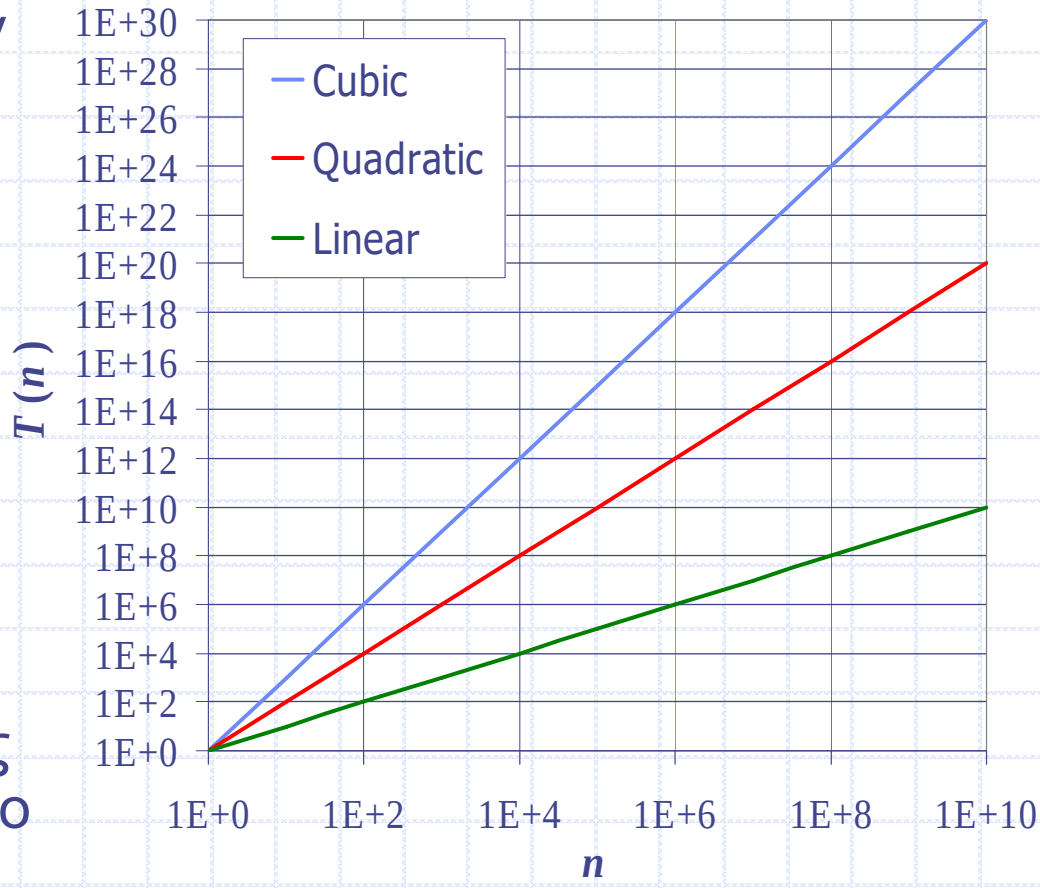


- Εν δυνάμει απεριόριστες θέσεις μνήμης όπου κάθε μια μπορεί να αποθηκεύσει οποιοδήποτε αριθμό ή σύμβολο

- Οι θέσεις μνήμης έχουν αριθμηθεί και η προσπέλαση σε μια θέση μνήμης απαιτεί μια μονάδα χρόνου.

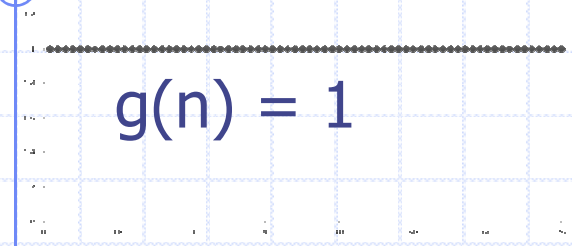
Επτά σημαντικές Συναρτήσεις

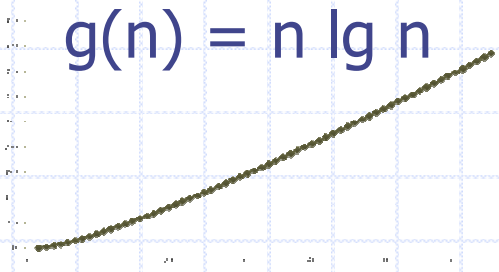
- Επτά συναρτήσεις που χρησιμοποιούνται στην ανάλυση αλγορίθμων:
  - Σταθερά ≈ 1
  - Λογαριθμική ≈ log n
  - Γραμμική ≈ n
  - N-Log-N ≈ n log n
  - Τετραγωνική ≈ n^2
  - Κυβική ≈ n^3
  - Εκθετική ≈ 2^n
- Σε ένα log-log διάγραμμα, η κλίση της γραμμής αντιστοιχεί στο ρυθμό αύξησης

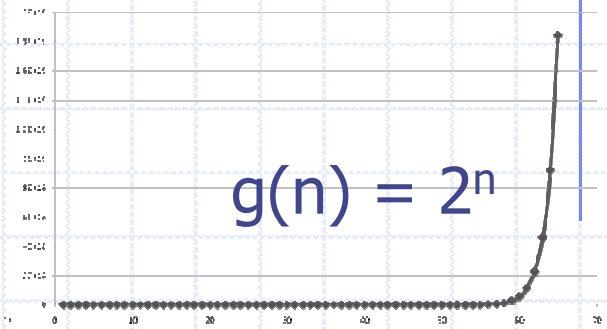


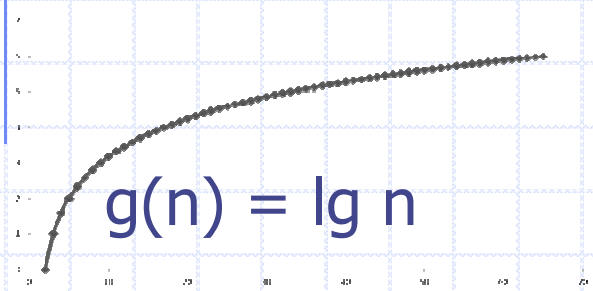
Γράφοι συναρτήσεων

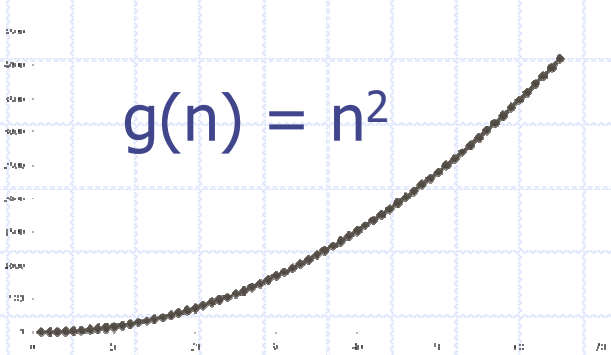
Από τον Matt Stallmann.

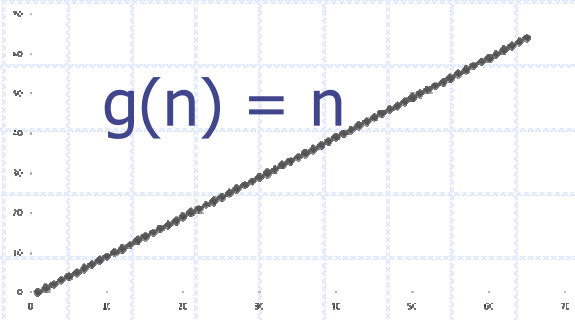


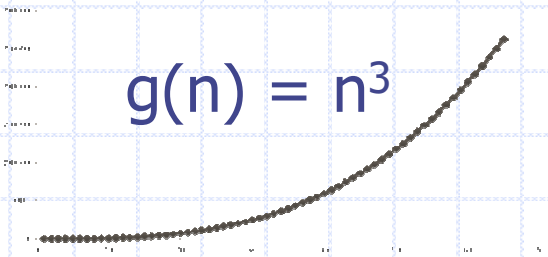












Βασικές Πράξεις

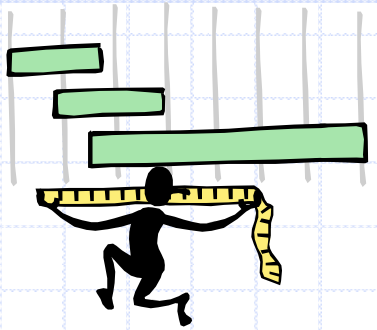
- Βασικοί υπολογισμοί που εκτελούνται από ένα αλγόριθμο
- Προσδιορίζονται σε ψευδοκώδικα
- Ανεξάρτητα από τη γλώσσα προγραμματισμού
- Δεν είναι σημαντικός ο ακριβής ορισμός
- Υποθέτουμε ότι κάθε μια απαιτεί μια μονάδα χρόνου
- Παραδείγματα:
  - Υπολογισμός μιας έκφρασης
  - Ανάθεση μιας τιμής σε μια μεταβλητή
  - Δείκτης πίνακα
  - Κλήση μιας μεθόδου
  - Επιστροφή από μια μέθοδο

Μέτρηση των Βασικών Πράξεων

- Με έλεγχο στον ψευδοκώδικα, μπορούμε να αποφανθούμε για το μέγιστο πλήθος βασικών πράξεων που εκτελούνται από ένα αλγόριθμο, σαν συνάρτηση του μεγέθους της εισόδου

Algorithm *arrayMax*(*A*, *n*)	# operations
currentMax ← *A*[0]	2
for *i* ← 1 **to** *n* – 1 **do**	2*n*
if *A*[*i*] > *currentMax* **then**	2(*n* – 1)
currentMax ← *A*[*i*]	2(*n* – 1)
{ increment counter *i* }	2(*n* – 1)
return *currentMax*	1
Σύνολο	8*n* – 2

Εκτίμηση του Χρόνου Εκτέλεσης



- Ο αλγόριθμος ***arrayMax*** εκτελεί $8n - 2$ βασικές πράξεις στη χειρότερη περίπτωση. Ορισμός:

  a = Χρόνος της γρηγορότερης βασικής πράξης

  b = Χρόνος της πιό αργής βασικής πράξης

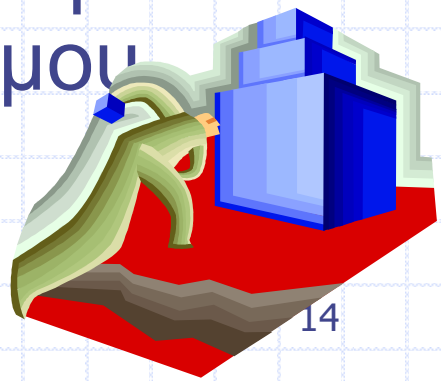
- Έστω $T(n)$ ο χειρότερος χρόνος του ***arrayMax.*** Τότε

$$a(8n - 2) \leq T(n) \leq b(8n - 2)$$

- Επομένως, ο χρόνος τρεξίματος $T(n)$ φράσεται από δύο γραμμικές συναρτήσεις

Ρυθμός Αύξησης του χρόνου τρεξίματος

- Αλλαγή του περιβάλλοντος υλικού/ λογισμικού
  - Επηρεάζει το $T(n)$ κατά σταθερό παράγοντα, αλλά
  - Δεν αλλάζει το ρυθμό αύξησης του $T(n)$
- Η γραμμική αύξηση του ρυθμού του χρόνου τρεξίματος του $T(n)$ είναι μια εσωτερική ιδιότητα του αλγορίθμου ***arrayMax***



Γιατί είναι σημαντικός ο ρυθμός αύξησης

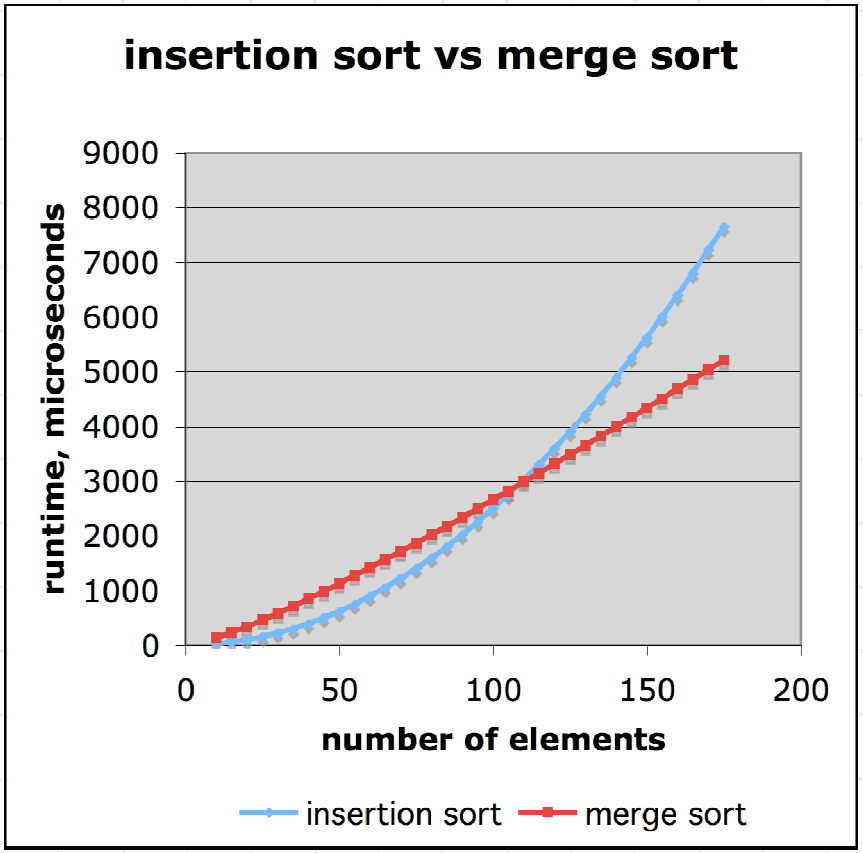
Από τον Matt Stallmann

Αν ο χρόνος είναι	για n + 1	για 2 n	για 4 n
c lg n	c lg (n + 1)	c (lg n + 1)	c(lg n + 2)
c n	c (n + 1)	2c n	4c n
c n lg n	~ c n lg n + c n	2c n lg n + 2cn	4c n lg n + 4cn
$c\ n^2$	~ $c\ n^2 + 2c\ n$	**$4c\ n^2$**	$16c\ n^2$
$c\ n^3$	~ $c\ n^3 + 3c\ n^2$	$8c\ n^3$	$64c\ n^3$
$c\ 2^n$	$c\ 2^{n+1}$	$c\ 2^{2n}$	$c\ 2^{4n}$

Ο χρόνος τετραπλασιάζεται όταν διπλασιασθεί το μέγεθος του προβλήματος

Από τον Matt Stallmann.

Σύγκριση δύο Αλγορίθμων



η ταξινόμηση με παρεμβολή είναι
$n^2 / 4$

ταξινόμηση με συγχώνευση
$2 n \lg n$

ταξινόμηση 1000000 στοιχείων?

η παρεμβολή θέλει
περίπου 70 ώρες

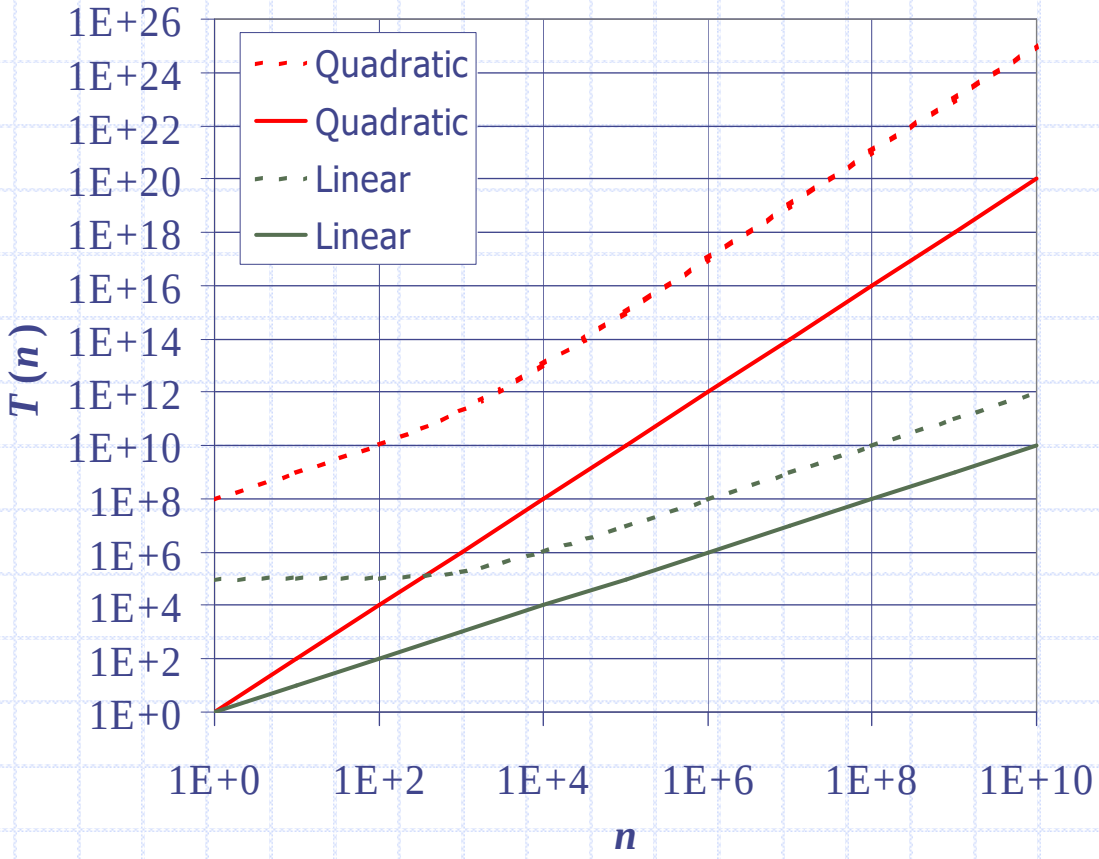
ενώ

η συγχώνευση
περίπου 40 δεύτερα

Αυτή είναι μια αργή μηχανή αλλά
100 x γρηγορότερη είναι 40 λεπτά
έναντι λιγότερο από 0.5 sec

Σταθεροί Παράγοντες

- Ο ρυθμός αύξησης δεν επηρεάζεται από
  - σταθερούς παράγοντες ή
  - όρους χαμηλότερης τάξης
- Παραδείγματα
  - $10^2n + 10^5$ είναι γραμμική συνάρτηση
  - $10^5n^2 + 10^8n$ είναι τετραγωνική συνάρτηση

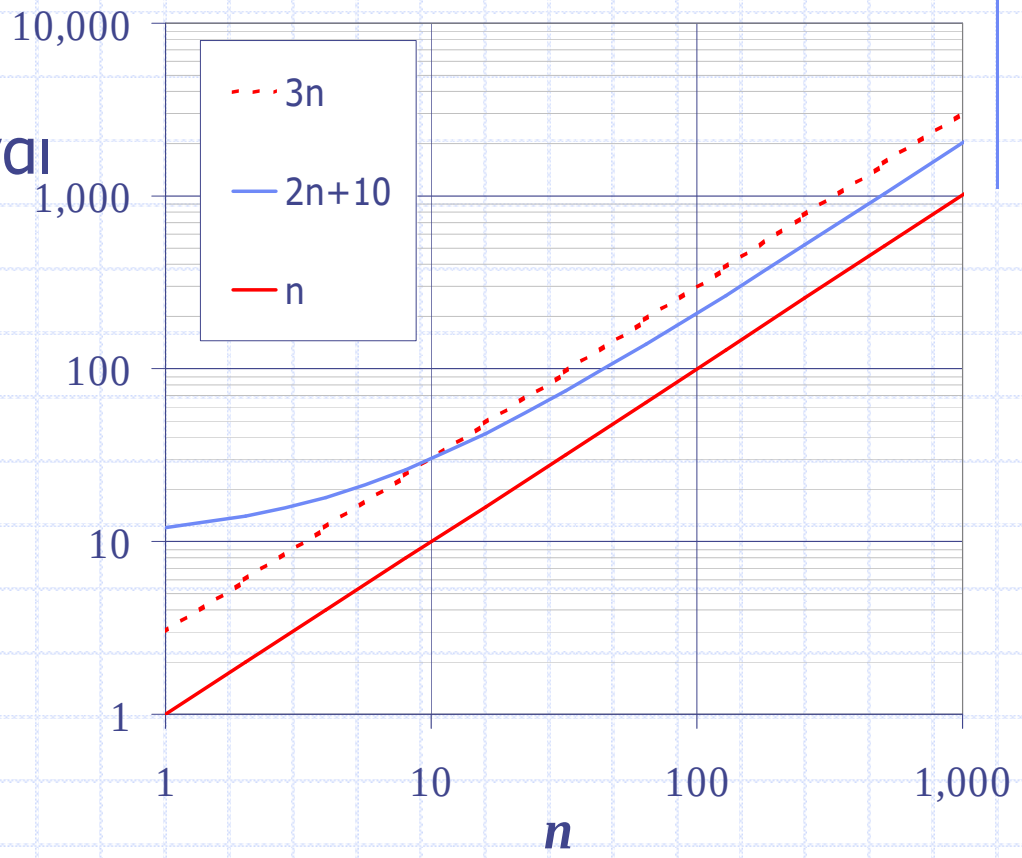


Συμβολισμός Big-Oh

- Όταν δίδονται οι συναρτήσεις $f(n)$ και $g(n)$, λέμε ότι η $f(n)$ είναι $O(g(n))$ αν υπάρχουν θετικές σταθερές c και n_0 έτσι ώστε

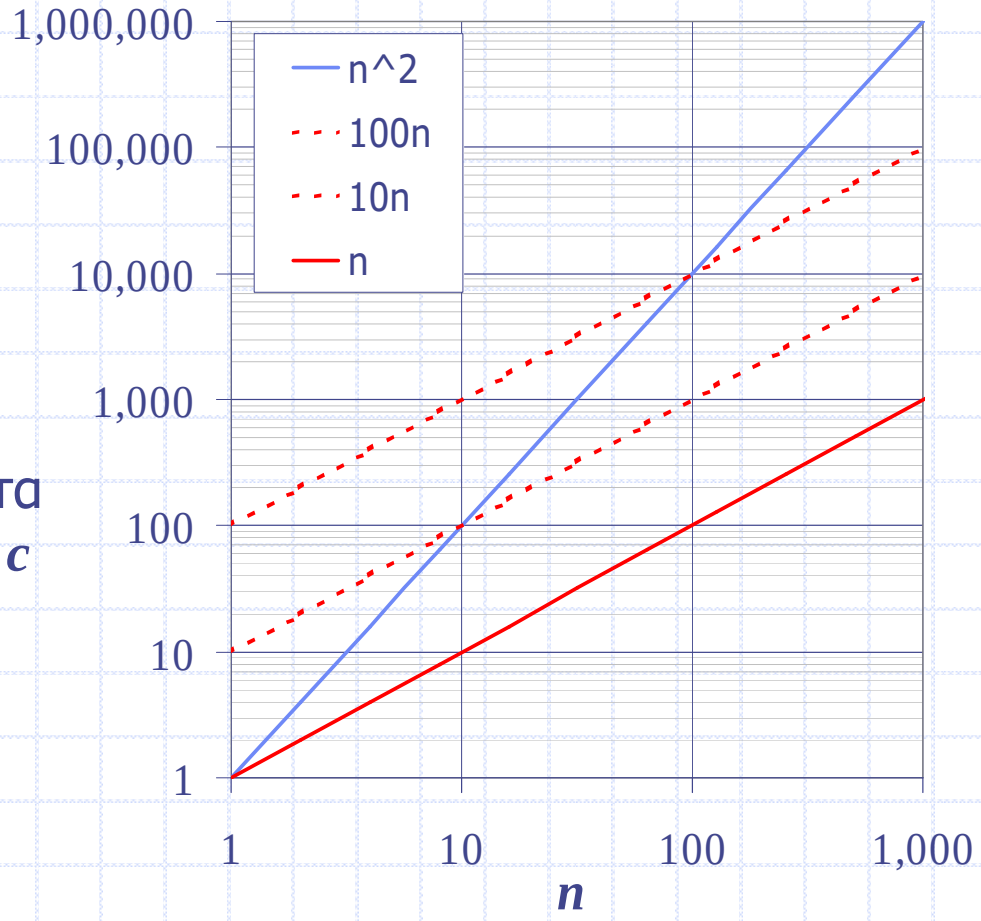
  $f(n) \leq cg(n)$ για $n \geq n_0$

- Παράδειγμα: $2n + 10$ είναι $O(n)$
  - $2n + 10 \leq cn$
  - $(c - 2)\, n \geq 10$
  - $n \geq 10/(c - 2)$
  - Έστω $c = 3$ και $n_0 = 10$

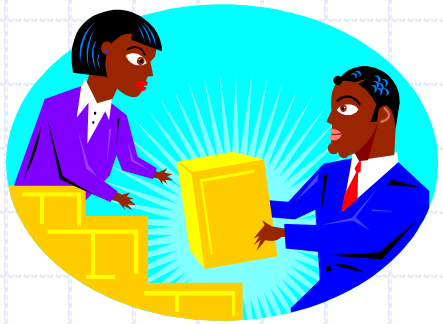


Παράδειγμα Big-Oh

- Παράδειγμα: η συνάρτηση n^2 δεν είναι $O(n)$
  - $n^2 \leq cn$
  - $n \leq c$
  - Η παραπάνω ανισότητα δεν ισχύει εφόσον το c πρέπει να είναι μια σταθερά



Παραδείγματα Big-Oh



- $7n-2$

  $7n-2$ είναι $O(n)$

  θέλουμε ένα $c > 0$ και ένα $n_0 \geq 1$ τέτοιο ώστε $7n-2 \leq c \bullet n$ για $n \geq n_0$

  αυτό ισχύει για $c = 7$ και $n_0 = 1$

- $3n^3 + 20n^2 + 5$

  $3n^3 + 20n^2 + 5$ είναι $O(n^3)$

  θέλουμε ένα $c > 0$ και ένα $n_0 \geq 1$ τέτοιο ώστε $3n^3 + 20n^2 + 5 \leq c \bullet n^3$ για $n \geq n_0$ αυτό ισχύει για $c=4$ και $n_0 = 21$

- $3 \log n + 5$

  $3 \log n + 5$ είναι $O(\log n)$

  θέλουμε ένα $c > 0$ και ένα $n_0 \geq 1$ τέτοιο ώστε $3 \log n + 5 \leq c \bullet \log n$ για $n \geq n_0$ αυτό ισχύει για $c=8$ και $n_0 = 2$

Big-Oh και Ρυθμός Αύξησης

- Ο συμβολισμός big-Oh αποτελεί ένα άνω φράγμα στο ρυθμό αύξησης μιας συνάρτησης
- Η πρόταση "η $f(n)$ είναι $O(g(n))$" σημαίνει ότι ο ρυθμός αύξησης της $f(n)$ δεν ξεπερνά το ρυθμό αύξησης της $g(n)$
- Μπορούμε να χρησιμοποιήσουμε τον συμβολισμό big-Oh για βαθμολόγηση συναρτήσεων με βάση το ρυθμό αύξησής τους

	$f(n)$ είναι $O(g(n))$	$g(n)$ είναι $O(f(n))$
$g(n)$ πιό γρήγορα	ναι	όχι
$f(n)$ πιό γρήγορα	όχι	ναι
Ίδια αύξηση	ναι	ναι

Κανόνες της Big-Oh

- Αν η $f(n)$ είναι πολυώνυμο βαθμού d, τότε η $f(n)$ είναι $O(n^d)$, δηλ.
  1. Απορρίπτουμε τους όρους χαμηλότερης τάξης
  2. Απορρίπτουμε τις σταθερές
- Χρησιμοποιούμε την ελάχιστη δυνατή κλάση συναρτήσεων
  - Λέμε "η $2n$ είναι $O(n)$" αντί για " η $2n$ είναι $O(n^2)$"
- Χρησιμοποιούμε την απλούστερη έκφραση της κλάσης
  - Λέμε "η $3n + 5$ είναι $O(n)$" αντί "η $3n + 5$ είναι $O(3n)$"

Ασυμπτωτική Ανάλυση Αλγορίθμων

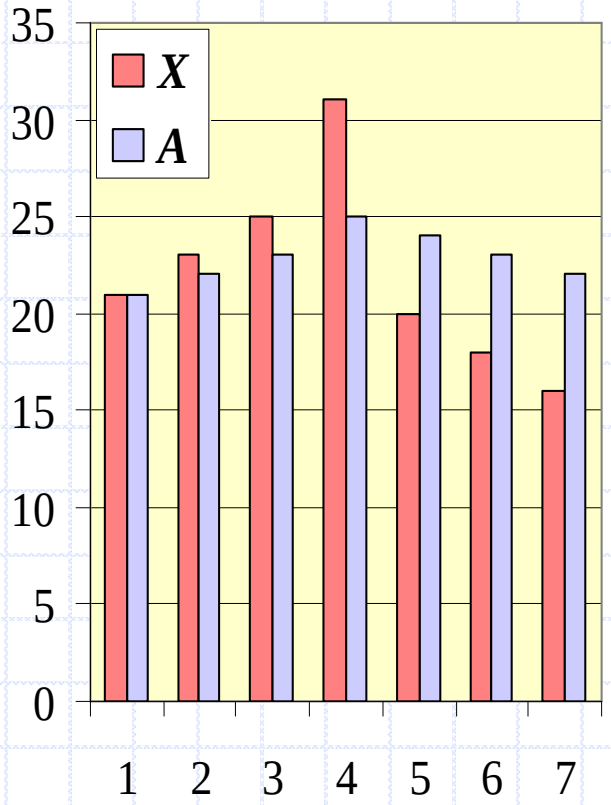
- Η ασυμπτωτική ανάλυση ενός αλγορίθμου καθορίζει το χρόνο τρεξίματος σε συμβολισμό big-Oh
- Για την ασυμπτωτική ανάλυση
  - Βρίσκουμε την χειρότερη περίπτωση βασικών πράξεων σαν συνάρτηση του μεγέθους της εισόδου
  - Εκφράζουμε αυτή τη συνάρτηση με συμβολισμό big-Oh
- Παράδειγμα:
  - Βρίσκουμε ότι ο αλγόριθμος ***arrayMax*** εκτελεί το πολύ $8n - 2$ βασικές πράξεις
  - Λέμε ότι ο αλγόριθμος ***arrayMax*** "τρέχει σε $O(n)$ χρόνο"
- Αφού οι σταθερές και οι χαμηλότερης τάξης όροι τελικά απορρίπτονται, μπορούμε να τους αγνοήσουμε όταν μετράμε τις βασικές πράξεις

Υπολογισμός μέσων προθέματος

- Δείχνουμε επιπλέον ασυμπτωτική ανάλυση με δύο αλγόριθμους για μέσους προθέματος
- Ο i-th μέσος προθέματος ενός πίνακα X είναι η μέση τιμή των πρώτων $(i+1)$ στοιχείων του X:

$$A[i] = (X[0] + X[1] + \ldots + X[i])/(i+1)$$

- Ο υπολογισμός ενός πίνακα A με τους μέσους προθέματος ενός άλλου πίνακα X έχει εφαρμογές στην οικονομική ανάλυση



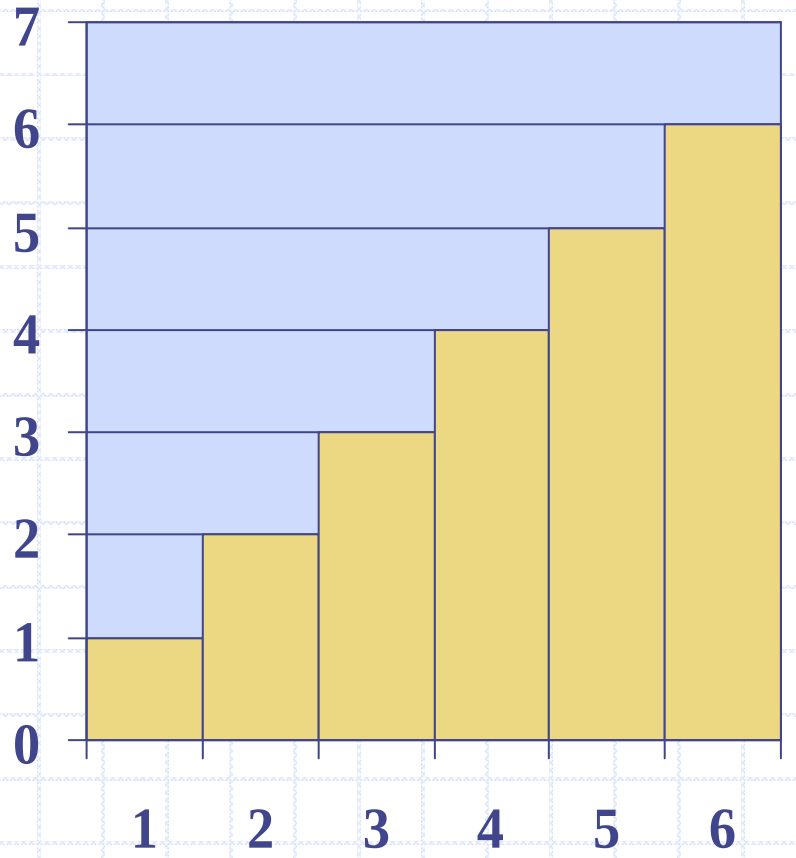
Μέσοι προθέματος (Τετραγωνικός)

- Ο παρακάτω αλγόριθμος υπολογίζει τους μέσους προθέματος σε τετραγωνικό χρόνο εφαρμόζοντας τον ορισμό

```
Algorithm prefixAverages1(X, n)
    Input array X of n integers
    Output array A of prefix averages of X          #operations
    A ← new array of n integers                          n
    for i ← 0 to n – 1 do                                n
        s ← X[0]                                         n
        for j ← 1 to i do                    1 + 2 + …+ (n – 1)
            s ← s + X[j]                     1 + 2 + …+ (n – 1)
        A[i] ← s / (i + 1)                               n
    return A                                             1
```

Αριθμητική Πρόοδος

- Ο χρόνος του ***prefixAverages1*** είναι $O(1 + 2 + \ldots + n)$
- Το άθροισμα των πρώτων n ακεραίων είναι $n(n + 1) / 2$
  - Υπάρχει μια απλή οπτική απόδειξη αυτού του γεγονότος
- Επομένως, ο αλγόριθμος ***prefixAverages1*** τρέχει σε $O(n^2)$ χρόνο



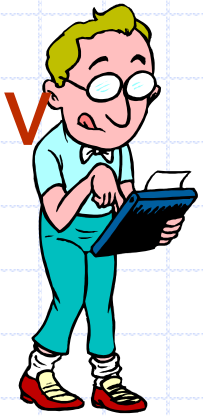
Μέσοι Προθέματος (Γραμμικός)

- Ο παρακάτω αλγόριθμος υπολογίζει τους μέσους προθέματος σε γραμμικό χρόνο αποθηκεύοντας το τρέχον άθροισμα

```
Algorithm prefixAverages2(X, n)
    Input array X of n integers
    Output array A of prefix averages of X          #operations
    A ← new array of n integers                          n
    s ← 0                                                1
    for i ← 0 to n – 1 do                                n
        s ← s + X[i]                                     n
        A[i] ← s / (i + 1)                               n
    return A                                             1
```

- Ο αλγόριθμος *prefixAverages2* τρέχει σε $O(n)$ χρόνο

Μαθηματικά που θα σας χρειασθούν



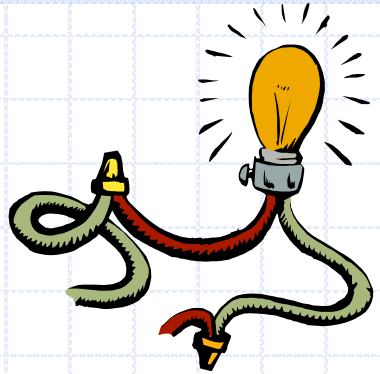
- Αθροίσματα
- Λογάριθμοι και Εκθετικές
- Τεχνικές απόδειξης
- Βασικές πιθανότητες

- **Ιδιότητες λογαρίθμων :**
  $\log_b(xy) = \log_b x + \log_b y$
  $\log_b (x/y) = \log_b x - \log_b y$
  $\log_b xa = a\log_b x$
  $\log_b a = \log_x a/\log_x b$
- **Ιδιότητες εκθετικών** :
  $a^{(b+c)} = a^b a^c$
  $a^{bc} = (a^b)^c$
  $a^b / a^c = a^{(b-c)}$
  $b = a^{\log_a b}$
  $b^c = a^{c*\log_a b}$

Συγγενείς της Big-Oh

- **big-Omega**
  - η f(n) είναι Ω(g(n)) αν υπάρχει μια σταθερά c > 0 και μια ακέραια σταθερά $n_0 \geq 1$ τέτοια ώστε $f(n) \geq c \bullet g(n)$ για $n \geq n_0$
- **big-Theta**
  - η f(n) είναι Θ(g(n)) αν υπάρχουν σταθερές c′ > 0 και c″ > 0 και μια ακέραια σταθερά $n_0 \geq 1$ τέτοια ώστε $c' \bullet g(n) \leq f(n) \leq c'' \bullet g(n)$ για $n \geq n_0$

Διαίσθηση του Ασυμπτωτικού Συμβολισμού



Big-Oh

- η f(n) είναι O(g(n)) αν η f(n) είναι ασυμπτωτικά **μικρότερη ή ίση** με την g(n)

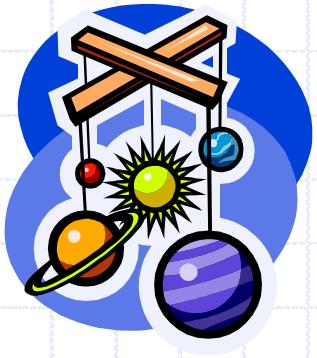
big-Omega

- η f(n) είναι Ω(g(n)) αν η f(n) είναι ασυμπτωτικά **μεγαλύτερη ή ίση** με την g(n)

big-Theta

- η f(n) είναι Θ(g(n)) αν η f(n) είναι ασυμπτωτικά **ίση** με την g(n)

Παραδείγματα Χρήσης των συγγενών της Big-Oh



- **$5n^2$ είναι $\Omega(n^2)$**

  $f(n)$ είναι $\Omega(g(n))$ αν υπάρχει μια σταθερά $c > 0$ και μια ακέραια σταθερά $n_0 \geq 1$ τέτοια ώστε $f(n) \geq c \cdot g(n)$ για $n \geq n_0$

  έστω $c = 5$ και $n_0 = 1$

- **$5n^2$ είναι $\Omega(n)$**

  $f(n)$ είναι $\Omega(g(n))$ αν υπάρχει μια σταθερά $c > 0$ και μια ακέραια σταθερά $n_0 \geq 1$ τέτοια ώστε $f(n) \geq c \cdot g(n)$ για $n \geq n_0$

  έστω $c = 1$ και $n_0 = 1$

- **$5n^2$ είναι $\Theta(n^2)$**

  $f(n)$ είναι $\Theta(g(n))$ αν είναι $\Omega(n^2)$ και $O(n^2)$. Έχουμε ήδη διαπιστώσει το πρώτο, για το δεύτερο θυμίζουμε ότι η $f(n)$ είναι $O(g(n))$ αν υπάρχει μια σταθερά $c > 0$ και μια ακέραια $n_0 \geq 1$ υέτοια ώστε $f(n) \leq c \cdot g(n)$ για $n \geq n_0$

  Έστω $c = 5$ και $n_0 = 1$