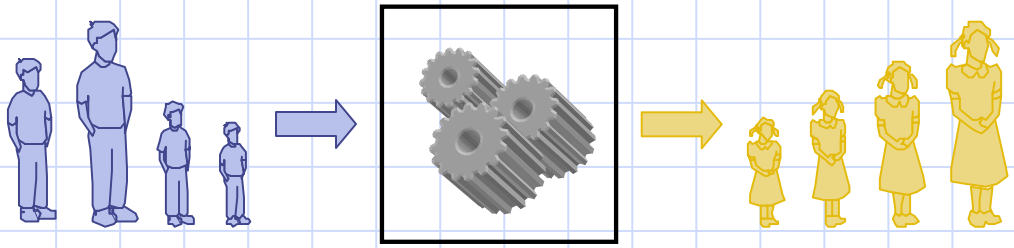
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Αλγόριθμοι και πολυπλοκότητα

Ανάλυση αλγορίθμων

Ιωάννης Τόλλης
Τμήμα Επιστήμης Υπολογιστών

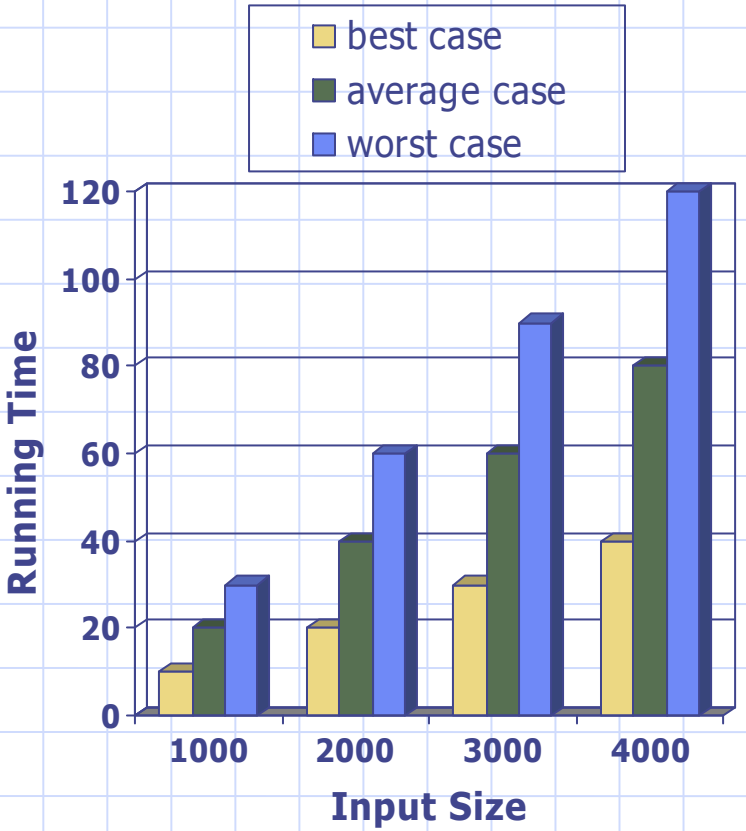
Ανάλυση Αλγορίθμων



Ένας αλγόριθμος είναι μια βήμα προς βήμα διαδικασία επίλυσης ενός προβλήματος

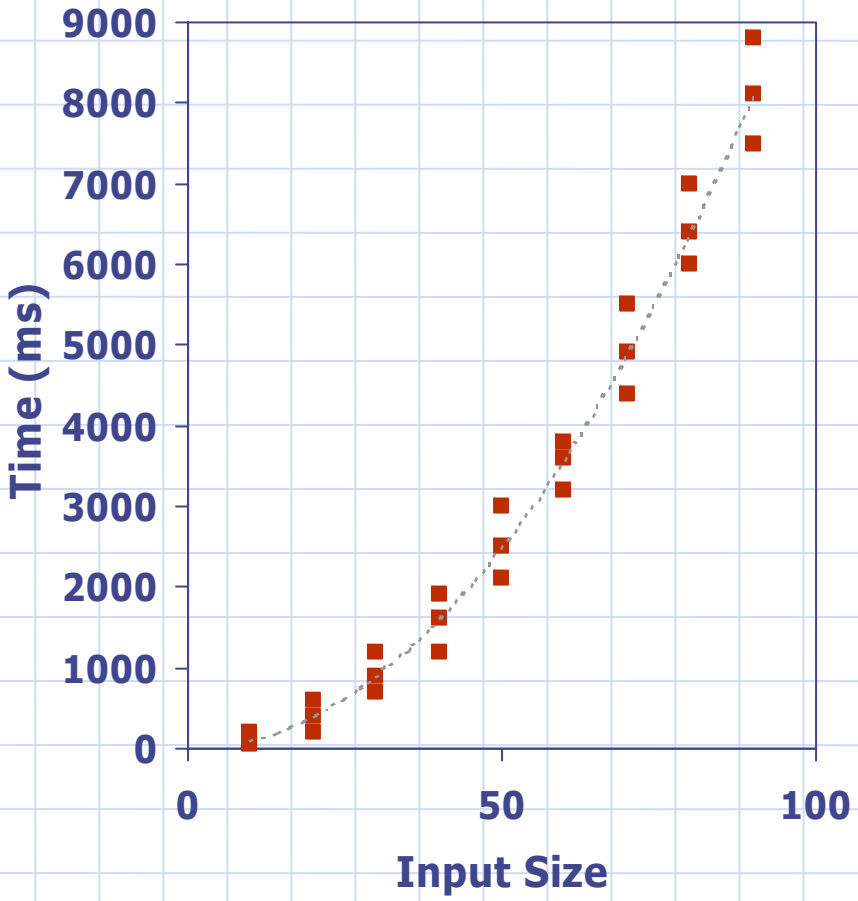
Χρόνος τρεξίματος (§1.1)

- Οι περισσότεροι αλγόριθμοι μετασχηματίζουν αντικείμενα εισόδου σε αντικείμενα εξόδου
- Ο χρόνος τρεξίματος ενός αλγορίθμοι συνήθως μεγαλώνει με το μέγεθος της εισόδου
- Ο χρόνος μέσης περίπτωσης είναι συνήθως δύσκολος να υπολογιστεί
- Επικεντρωνόμαστε στο χρόνο χείριστης περίπτωσης.
  - Ευκολότερο στην ανάλυση
  - Κρίσιμο σε εφαρμογές όπως παιχνίδια, ρομποτική, οικονομικά



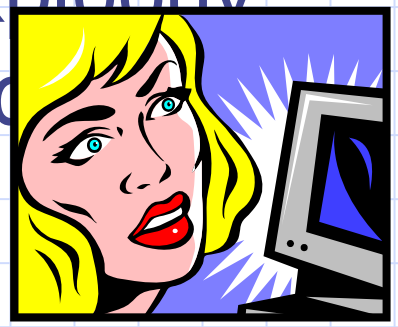
Πειραματική μελέτη (§ 1.6)

- Γράψε ένα πρόγραμμα που να υλοποιεί έαν αλγόριθμο
- Τρέξε το πρόγραμμα με εισόδους διαφόρων μεγεθών και συνθέσεων
- Χρησιμοποίησε μια μέθοδο όπως η System.currentTimeMillis() για μία ακριβή μέτρηση του χρόνου τρεξίματος
- Κάνε το γράφημα των αποτελεσμάτων



Περιορισμοί των πειραμάτων

- Πρέπει να υλοποιηθεί ο αλγόριθμος, κάτι που συχνά είναι δύσκολο
- Τα αποτελέσματα μπορεί να μην είναι ενδεικτικά του χρόνου τρεξίματος σε άλλες εισόδους που δεν συμπεριελήφθησαν στο πείραμα
- Δύο αλγόριθμοι μπορούν να συγκριθούν μόνο σε ίδιο περιβάλλον υλικού κα λογισμικού (hardware/software)



Θεωρητική Ανάλυση

- Χρησιμοποιεί μια high-level περιγραφή του αλγορίθμου αντί για υλοποίηση
- Εκφράζει το χρόνο τρεξίματος σαν συνάρτηση του μεγέθους της εισόδου
- Συνυπολογίζει όλες τις πιθανές εισόδους
- Μας επιτρέπει να υπολογίσουμε την ταχύτητα ενός αλγορίθμου ανεξάρτητα από το περιβάλλον υλικού/λογισμικού

Ψευδοκώδικας(§1.1)

- High-level περιγραφή αλγορίθμου
- Πιο δομημένος απο ένα Αγγλικό ή Ελληνικό κείμενο.
- Λιγότερο λεπτομερής από το πρόγραμμα
- Προτιμώμενη σημειογραφία για την περιγραφή αλγορίθμων
- Κρύβει προβλήματα σχεδιασμού προγραμμάτων

Παράδειγμα: Εύρεση του μεγίστου στοιχείου 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** …
  - Indentation replaces braces
- Δήλωση μεθόδου

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

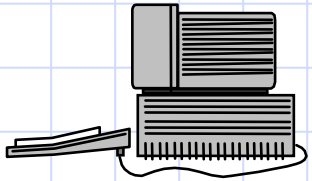
    Input …

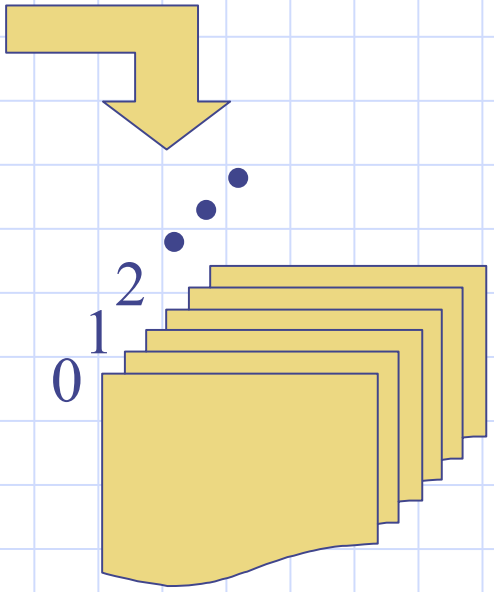
    Output …
- Κλήση μεθόδου

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

  return ***expression***
- Εκφράσεις
  - ← Ανάθεση (όπως το = στην Java)
  - = Έλεγχος ισότητας (όπως το == στην Java)
  - n^2 Εκθέτες και μαθηματική τυποποίηση επιτρέπονται

Το μοντέλο Random Access Machine (RAM)





- Α **CPU**
- Μία δυνητικά απερόριστη "τράπεζα"από κελιά μνήμης, κάθε ένα από τα οποία μπορεί να κρατήσει έναν αυθαίρετο αριθμό ή χαρακτήρα
- Τα κελιά μνήμης είναι αριθμημένα και η πρόσβαση οποιουδήποτε από αυτά παίρνει μοναδιαίο χρόνο

Βασικές λειτουργίες

- Βασικοι υπολογισμοί που εκτελεί ένας αλγόριθμος
- Αναγνωρίσιμοι σε ψευδοκώδικα
- Ανεξάρτητοι από τη γλώσσα προγραμματισμού
- Ο ακριβής ορισμός δεν είναι σημαντικός
- Θεωρείται ότι παίρουν σταθερό χρόνο στο μοντέλο RAM
- Παραδείγματα:
  - Υπολογισμός μιας έκφρασης
  - Ανάθεση μιας τιμής σε μια μεταβλητή
  - Αναφορά σε στοιχείο ενός πίνακα
  - Κλήση μιας μεθόδου
  - Επιστροφή από μια μέθοδο

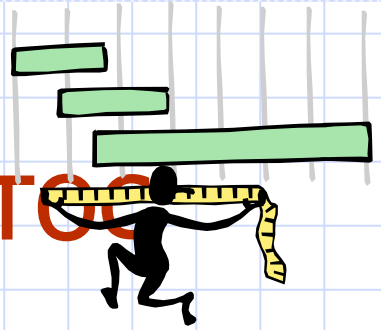
Μέτρηση Βασικών Λειτουργιών (§1.1)

- Παρατηρώντας τον ψευδοκώδικα, μπορούμε να καθορίσουμε ένα μέγιστο αριθμό βασικών λειτουργιών που εκτελούνται από έναν αλγόριθμο, σαν συνάρτηση του μεγέθους της εισόδου

Algorithm ***arrayMax***(A, n)

	# operations
currentMax $\leftarrow A[0]$	2
for $i \leftarrow 1$ **to** $n - 1$ **do**	$2 + n$
if $A[i] >$ ***currentMax*** **then**	$2(n - 1)$
currentMax $\leftarrow A[i]$	$2(n - 1)$
{ increment counter i }	$2(n - 1)$
return ***currentMax***	1
Total	$7n - 1$

Εκτίμηση χρόνου τρεξίματος



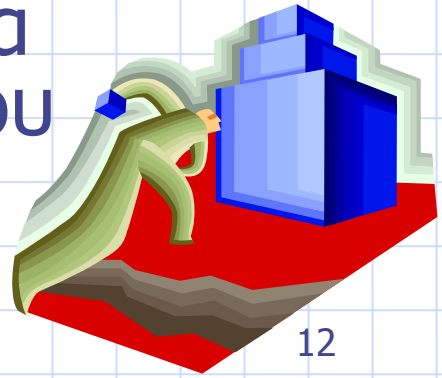
- Ο αλγόριθμος ***arrayMax*** εκτελεί $7n - 1$ βασικές λειτουργίες στη χειρότερη περίπτωση. Ορίζουμε:
  - a = Χρόνος που χρειάζεται η πιο γρήγορη βασική λειτουργία
  - b = Χρόνος που χρειάζεται η αργότερη βασική λειτουργία
- Έστω $T(n)$ ο χρόνος χείριστης περίπτωσης του ***arrayMax***. Τότε :

$$a(7n - 1) \leq T(n) \leq b(7n - 1)$$

- Ο χρόνος τρεξίματος $T(n)$ είναι φραγμένος από δύο γραμμικές συναρτήσεις

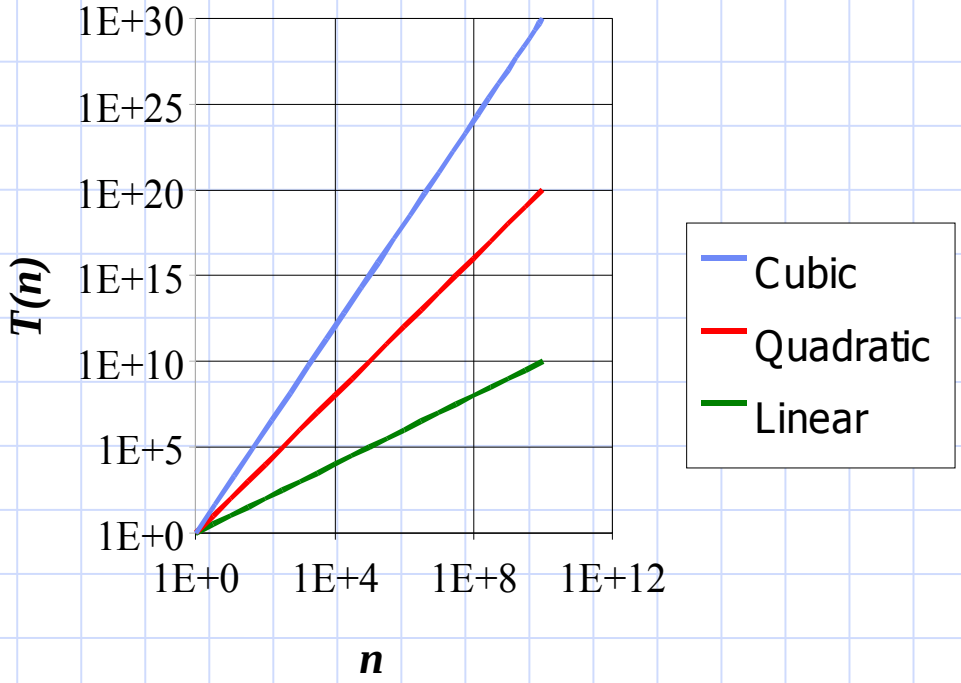
Ρυθμός Ανάπτυξης του Χρόνου Τρεξίματος

- Η αλλαγή του υλικού/λογισμικού περιβάλλοντος
  - Επηρεάζει τον $T(n)$ κατα ένα σταθερό παράγοντα,αλλά
  - Δεν αλλάζει το ρυθμό ανάπτυξηστου $T(n)$
- Ο γραμμικός ρυθμός ανάπτυξης του χρόνου τρεξίματος $T(n)$ είναι μία εγγενής ιδιότητα του αλγορίθμου ***arrayMax***



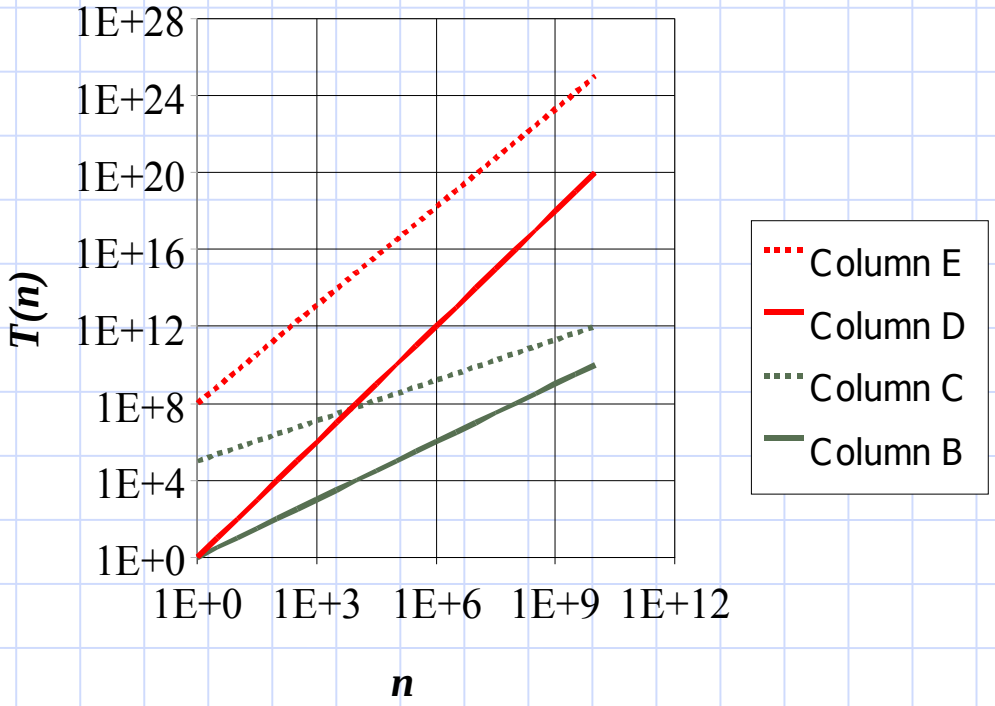
Ρυθμοί Ανάπτυξης

- Ρυθμοί Ανάπτυξης Συναρτήσεων
  - Γραμμικός ≈ n
  - Τετραγωνικός ≈ n^2
  - Κυβικός ≈ n^3
- Σε ένα log-log διάγραμμα, η κλίση της ευθείας αντιστοιχεί στο ρυθμό ανάπτυξης της συνάρτησης



Σταθεροί Παράγοντες

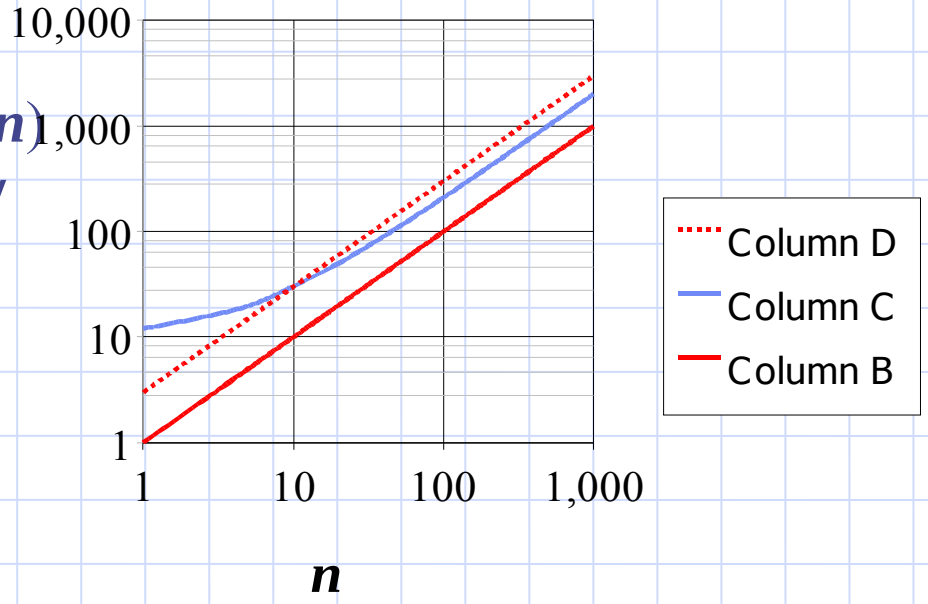
- Ο ρυθμός ανάπτυξης δεν επηρεάζεται από
  - σταθερούς παράγοντες
  - όρους κατώτερης τάξης
- Παραδείγματα
  - $10^2n + 10^5$ είναι μια γραμμική συνάρτηση
  - $10^5n^2 + 10^8n$ είναι μια τετραγωνική συνάρτηση



Συμβολισμός Big-Oh (§1.2)

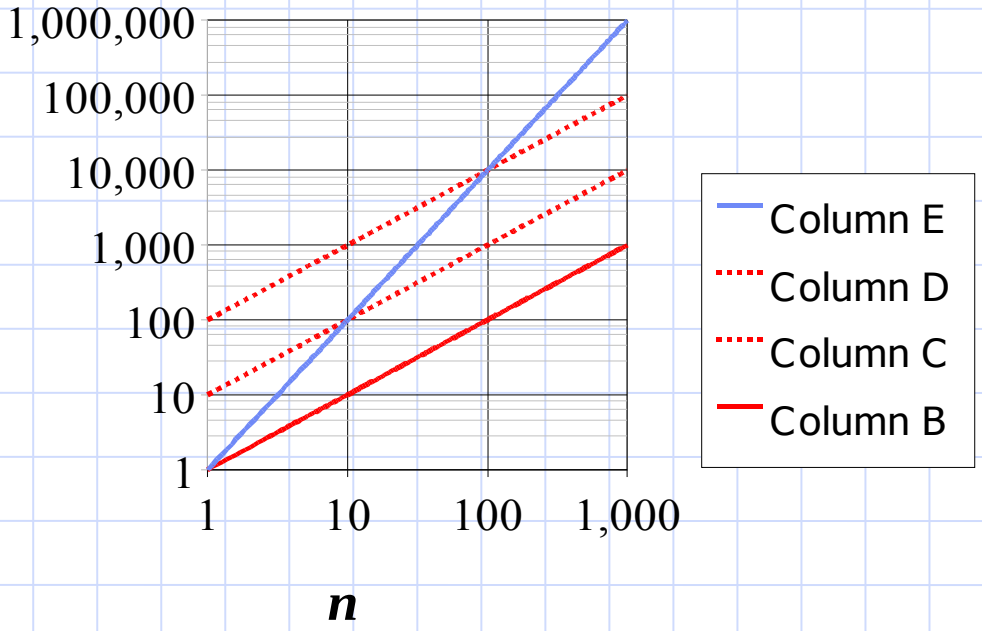
- Δοθείσων συναρτήσεων $f(n)$ και $g(n)$,λέμε ότι η $f(n)$ είναι $O(g(n))$ αν υπάρχουν θετικές σταθερές c και n_0 τέτοιες ώστε

  $f(n) \leq cg(n)$ για $n \geq n_0$
- Παράδειγμα: Η $2n + 10$ είναι $O(n)$
  - $2n + 10 \leq cn$
  - $(c - 2)\, n \geq 10$
  - $n \geq 10/(c - 2)$
  - Διάλεξε $c = 3$ και $n_0 = 10$

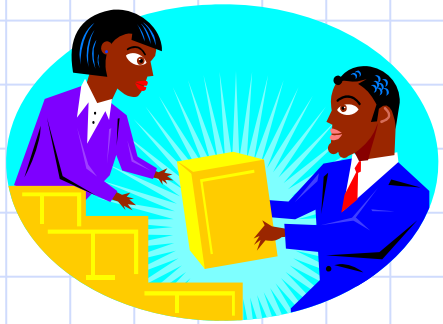


Big-Oh Παράδειγμα

- Παράδειγμα: η συνάρτηση n^2 δεν είναι $O(n)$
  - $n^2 \leq cn$
  - $n \leq c$
  - Η παραπάνω εξίσωση δεν μπορεί να ικανοποιηθεί αφού το c πρέπει να είναι σταθερά



Και άλλα Big-Oh Παραδείγματα



- 7n-2
  - 7n-2 είναι O(n)
  - Χρειάζεται $c > 0$ και $n_0 \geq 1$ τέτοια ώστε $7n-2 \leq c \bullet n$ for $n \geq n_0$
  - Αυτό ισχύει για $c = 7$ και $n_0 = 1$
- $3n^3 + 20n^2 + 5$
  - $3n^3 + 20n^2 + 5$ είναι $O(n^3)$
  - Χρειάζεται $c > 0$ και $n_0 \geq 1$ τέτοια 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 + log log n
  - 3 log n + log log n είναι O(log n)
  - Χρειάζεται $c > 0$ και $n_0 \geq 1$ τέτοια ώστε $3 \log n + \log \log n \leq c \bullet \log n$ for $n \geq n_0$
  - Αυτό ισχύει για $c = 4$ και $n_0 = 2$

Big-Oh and Growth Rate

- Ο συμβολισμός Big-Oh δίνει ένα άνω όριο για το ρυθμό ανάπτυξης μίας συνάρτησης
- Η δήλωση "η $f(n)$ είναι $O(g(n))$" σημαίνει ότι ο ρυθμός ανάπτυξης της $f(n)$ δεν είναι μεγαλύτερος από το ρυθμό ανάπτυξης της $g(n)$
- Μπορούμε ναχρησιμοποιήσουμε το συμβολισμό big-Oh για να ιεραρχήσουμε συναρτήσεις ανάλογα με το ρυθμό ανάπτυξής τους

	$f(n)$ is $O(g(n))$	$g(n)$ is $O(f(n))$
$g(n)$ grows more	Yes	No
$f(n)$ grows more	No	Yes
Same growth	Yes	Yes

Κανόνες Big-Oh

- Αν η $f(n)$ είναι πολυώνυμο βαθμού d, τότε η $f(n)$ είναι $O(n^d)$, δηλ,
  1. Βγάζουμε τους όρους κατώτερης τάξης
  2. Βγάζουμε τους σταθερούς όρους
- Χρησιμοποιούμε την μικρότερη 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 για "η $3n + 5$ είναι $O(3n)$"

Ασυμπτωτική Ανάλυση Αλγορίθμου

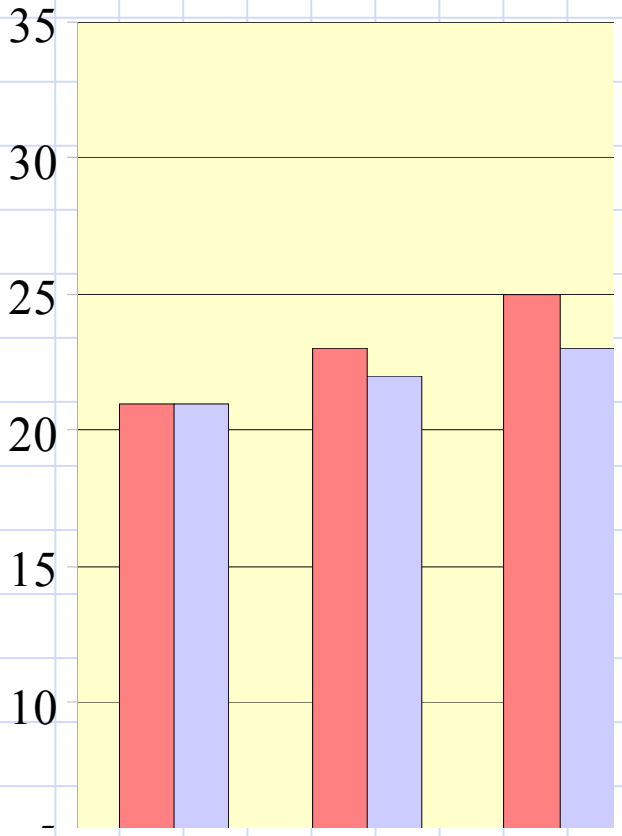
- Η ασυμπτωτική ανάλυση αλγορίθμου καθορίζει τον χρόνο τρεξίματος σε συμβολισμό Big -Oh
- Εκτέλεση της ασυμπτωτικής ανάλυσης
  - Βρίσκουμε το πλήθος βασικών λειτουργιών που εκτελούνται στη χείριστη περίπτωση σαν συνάρτηση του μεγέθους εισόδου
  - Εκφράζουμε αυτή τη συνάρτηση με συμβολισμό Big-Oh
- Παράδειγμα:
  - Προσδιορίζουμε ότι ο ***arrayMax*** εκτελεί το πολύ $7n - 1$ primitive λειτουργίες
  - Λέμε ότι ο αλγόριθμος ***arrayMax*** "τρέχει σε $O(n)$ χρόνο"
- Αφού οι σταθεροί παράγοντες και οι όροι κατώτερης τάξης δεν συνυπολογίζονται, μπορούμε να τους αφαιρέσουμε χωρίς να μετρήσουμε τις βασικές λειτουργίες τους

Υπολογισμός Μέσων Όρων Προθεμάτων

- Παρουσιάζουμε την ασυμπτωτική ανάλυση με δυο αλγορίθμους για μέσους όρους προθεμάτων
- Ο i-οστός μέσος όρος προθεμάτων ενός πίνακα X είναι ο μέσος όρος των πρώτων $(i+1)$ στοιχείων του X:

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

- Ο υπολογισμός του πίνακα A των μέσων όρων προθεμάτων ενός πίνακαγ X έχει εφαρμογές στην οικονομική ανάλυση



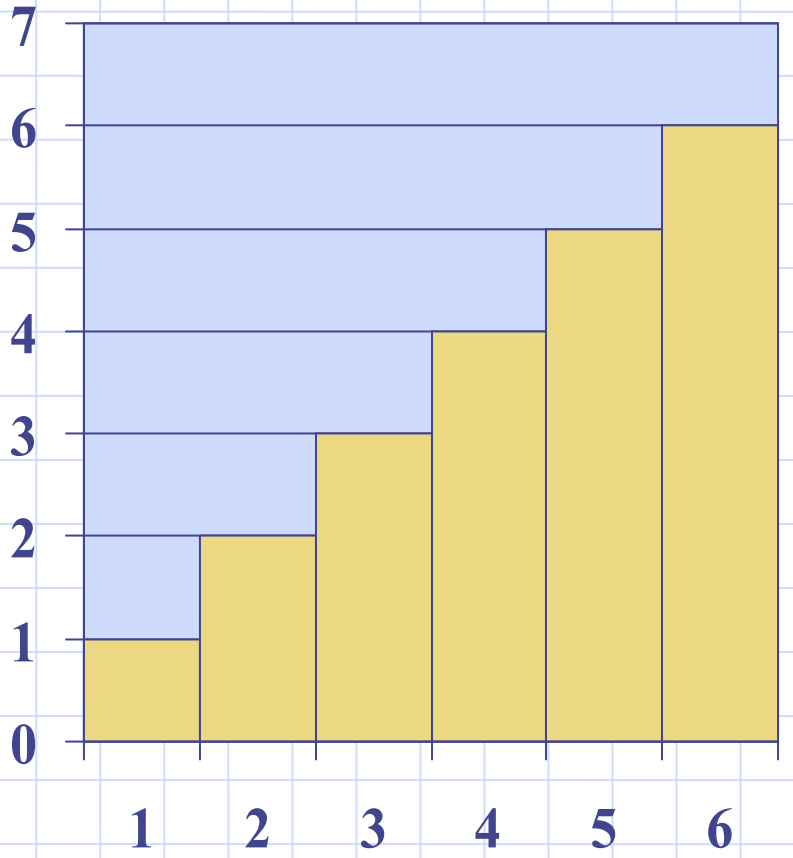
Μέσος Όρος Προθεμάτων (Τετραγωνικός)

- Ο ακόλουθος αλγόριθμος υπολογίζει τους μέσους όρους προθεμάτων σε τετραγωνικό χρόνο με εφαρμογή του ορισμού

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



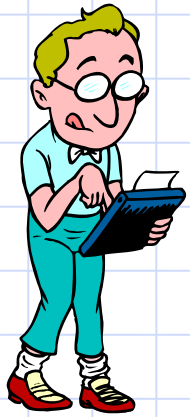
Μέσος Όρος Προθεμάτων (Γραμμικός)

- Ο ακόλουθος αλγόριθμος υπολογίζει τους μέσους όρους προθεμάτων σε γραμμικό χρόνο κρατώντας ένα 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
```

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

Μαθηματικά για ανασκόπηση



- Άθροιση (Sec. 1.3.1)
- Λογάριθμοι και Εκθέτες (Sec. 1.3.2)

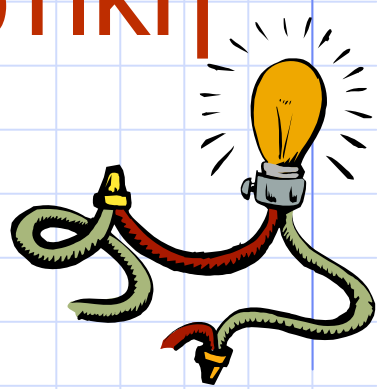
- **Ιδιότητες λογαρίθμων:**
  - $\log_b(xy) = \log_b x + \log_b y$
  - $\log_b (x/y) = \log_b x - \log_b y$
  - $\log_b xa = a\log_b x$
  - $\log_b a = \log_x a/\log_x b$
- **Ιδιότητες εκθετικών:**
  - $a^{(b+c)} = a^b a^c$
  - $a^{bc} = (a^b)^c$
  - $a^b / a^c = a^{(b-c)}$
  - $b = a^{\log_a b}$
  -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

- Τεχνικές Απόδειξης (Sec. 1.3.3)
- Βασικές Πιθανότητες (Sec. 1.3.4)

Συγγενείς του Big-Oh

- **big-Omega**
  - Η f(n) είναι $\Omega(g(n))$ αν υπάρχει σταθερά c > 0 και μία 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 $\Theta(g(n))$ αν υπάρχουν σταθερές c′ > 0 και c″ > 0 και ακέραια σταθερά $n_0 \geq 1$ τέτοια ώστε $c' \bullet g(n) \leq f(n) \leq c'' \bullet g(n)$ για $n \geq n_0$
- **little-oh**
  - f(n) είναι o(g(n)) αν, για κάθε σταθερά c > 0, υπάρχει μία ακέραια σταθερά $n_0 \geq 0$ τέτοια ώστε $f(n) \leq c \bullet g(n)$ για $n \geq n_0$
- **little-omega**
  - Η f(n) είναι $\omega(g(n))$ αν, για κάθε σταθερά c > 0, υπάρχει μία ακέραια σταθερά $n_0 \geq 0$ τέτοια ώστε $f(n) \geq c \bullet g(n)$ για $n \geq n_0$

Διαίσθηση για την Ασυμπτωτική Σημειογραφία



Big-Oh

- Η f(n) είναι O(g(n)) αν η f(n) είναι ασυμπτωτικά **less than or equal** to g(n)

big-Omega

- f(n) is Ω(g(n)) if f(n) is asymptotically **greater than or equal** to g(n)

big-Theta

- f(n) is Θ(g(n)) if f(n) is asymptotically **equal** to g(n)

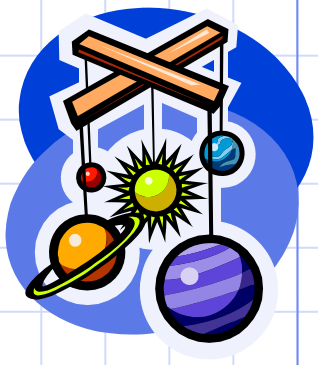
little-oh

- f(n) is o(g(n)) if f(n) is asymptotically **strictly less** than g(n)

little-omega

- f(n) is ω(g(n)) if is asymptotically **strictly greater** than g(n)

Παραδείγματα Χρήσης των Συγγενών του Big-Oh



- **$5n^2$ είναι $\Omega(n^2)$**

  Η f(n) είναι Ω(g(n)) αν υπάρχει μια σταθερά $c > 0$ και μία θετκή 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 $\Omega(n)$**

  Η f(n) είναι Ω(g(n)) αν υπάρχει μια σταθερά $c > 0$ και μία θετκή 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 $\omega(n)$**

  Η f(n) είναι ω(g(n)) αν, για κάθε σταθερά $c > 0$, υπάρχει μια θετική σταθερά $n_0 \geq 0$ τέτοια ώστε $f(n) \geq c \cdot g(n)$ για $n \geq n_0$

  Πρέπει $5n_0^2 \geq c \cdot n_0 \rightarrow$ για δοσμένο c, το n_0 που ικανοποιεί τη σχέση είναι κάποιο $n_0 \geq c/5 \geq 0$

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

Σημείωμα αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση, Όχι Παράγωγο Έργο 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/

- Ως Μη Εμπορική ορίζεται η χρήση:
  - που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
  - που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
  - που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Σημείωμα Αναφοράς

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.