ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Δομές Δεδομένων

Ιωάννης Γ. Τόλλης
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

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.

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».

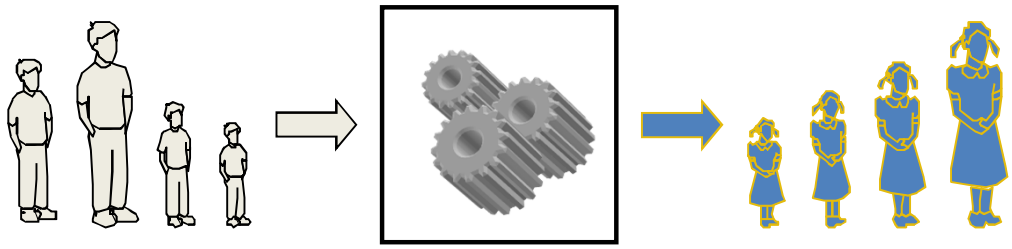
[1] http://creativecommons.org/licenses/by-nc-nd/4.0/

- 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

- Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

.

Ανάλυση Αλγορίθμων

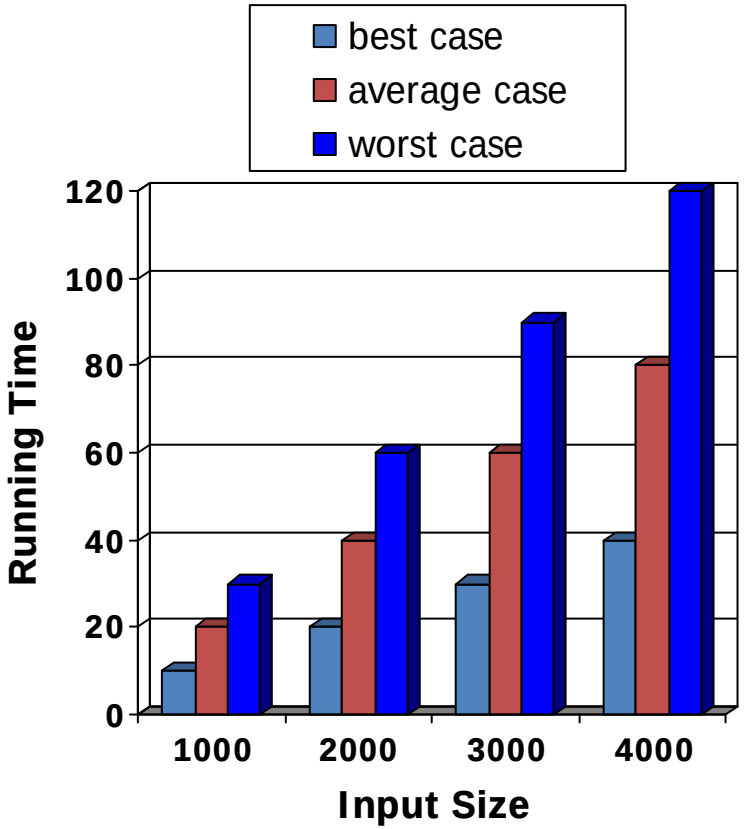


Περιγραφή και Υλικό Ανάγνωσης

- Χρόνος εκτέλεσης (§1.1)
- Ψευδοκώδικας (§1.1)
- Μέτρηση των στοιχειωδών πράξεων (§1.1)
- Ασυμπτωτική σημειογραφία (§1.2)
- Ασυμπτωτική ανάλυση (§1.2)
- Μελέτη περιπτώσεων (§1.3.1, §1.4)

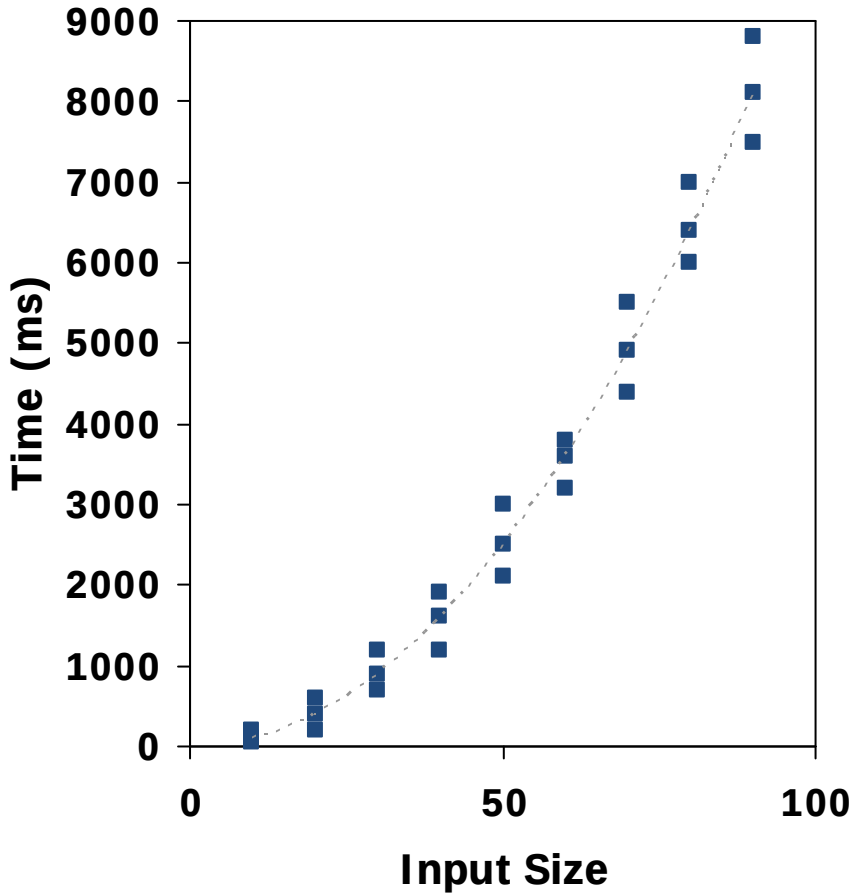
Χρόνος Εκτέλεσης

- Ο χρόνος εκτέλεσης ενός αλγορίθμου εξαρτάται από την είσοδο και συνήθως αυξάνεται με το μέγεθος της εισόδου
- Δύσκολο να καθοριστεί ένας μέσος χρόνος
- Επικεντρωνόμαστε στο χειρότερο χρόνο
  - Πιο εύκολη ανάλυση
  - Κρίσιμο για παιχνίδια, οικονομικές και ρομποτικές εφαρμογές



Πειραματικές Μελέτες

- Γράψτε ένα πρόγραμμα που να υλοποιεί τον αλγόριθμο
- Τρέξτε το πρόγραμμα με εισόδους διαφορετικού μεγέθους και περιεχομένου
- Χρησιμοποιήστε μια μέθοδο όπως η System.currentTimeMillis() για να λάβετε μια ακριβή μέτρηση του πραγματικού χρόνου εκτέλεσης
- Σχεδιάστε τα αποτελέσματα



Περιορισμοί των Πειραμάτων

- Πρέπει να υλοποιηθεί ο αλγόριθμος, πράγμα που μπορεί να είναι δύσκολο
- Τα αποτελέσματα μπορεί να μην είναι ενδεικτικά του χρόνου εκτέλεσης με άλλες εισόδους που δεν συμπεριλήφθηκαν στο πείραμα.
- Για να συγκριθούν δυο αλγόριθμοι πρέπει να χρησιμοποιηθούν τα ίδια περιβάλλοντα υλικού και λογισμικού

Θεωρητική Ανάλυση

- Χρησιμοποιείται μια περιγραφή υψηλού επιπέδου του αλγορίθμου αντί κάποιας υλοποίησης
- Λαμβάνει υπ’ όψιν κάθε πιθανή είσοδο
- Μας επιτρέπει να εκτιμήσουμε την ταχύτητα ενός αλγορίθμου ανεξάρτητα από το περιβάλλον υλικού/λογισμικού

Ψευδοκώδικας

- Υψηλού επιπέδου περιγραφή ενός αλγόριθμου
- Πιο δομημένο από την απλή γλώσσα
- Λιγότερο λεπτομερές από ένα πρόγραμμα
- Η προτιμώμενη σημειογραφία για την περιγραφή αλγορίθμων
- Αποκρύπτει τις λεπτομέρειες του σχεδιασμού των προγραμμάτων

Π.χ.: Να βρεθεί το μέγιστο 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** …
  - Εσοχές αντί για { }
- Δήλωση μεθόδων

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

  Input …

  Output …
- Κλήση μεθόδων

  var.method (***arg*** [, ***arg***…])
- Τιμή επιστροφής

  return ***expression***
- Εκφράσεις

  ← Καταχώρηση (όπως = στην Java)

  = Έλεγχος ισότητας (όπως == στην Java)

  n^2 Εκθέτες και άλλες μαθηματικές δομές επιτρέπονται

Στοιχειώδεις Πράξεις

- Βασικοί υπολογισμοί που εκτελεί ένας αλγόριθμος
- Αναγνωρίσιμοι στον ψευδοκώδικα
- Σε μεγάλο βαθμό ανεξάρτητοι από την γλώσσα προγραμματισμού
- Δεν είναι σημαντικός ένας ακριβής ορισμός (θα δούμε γιατί αργότερα)

- Παραδείγματα:
  - Υπολογισμός μιας έκφρασης
  - Καταχώρηση τιμής σε μεταβλητή
  - Indexing σε πίνακα
  - Κλήση μεθόδου
  - Επιστροφή από μέθοδο

Μέτρηση των Στοιχειωδών Πράξεων

- Ελέγχοντας τον ψευδοκώδικα, μπορούμε να βρούμε τον μέγιστο αριθμό των στοιχειωδών πράξεων που θα εκτελέσει ο αλγόριθμος 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
Total	7*n* – 1

Εκτίμηση του Χρόνου Εκτέλεσης

- Ο αλγόριθμος ***arrayMax*** εκτελεί $7\boldsymbol{n} - 1$ στοιχειώδεις πράξεις στην χειρότερη περίπτωση
- Ορίζουμε
  - $\boldsymbol{a}$ Ο χρόνος που απαιτείται για την γρηγορότερη στοιχειώδη πράξη
  - $\boldsymbol{b}$ Ο χρόνος που απαιτείται για την αργότερη στοιχειώδη πράξη
- Έστω ότι $\boldsymbol{T}(\boldsymbol{n})$ είναι ο χρόνος εκτέλεσης χειρότερης περίπτωσης του ***arrayMax***. Έχουμε

$$\boldsymbol{a}\,(7\boldsymbol{n} - 1) \leq \boldsymbol{T}(\boldsymbol{n}) \leq \boldsymbol{b}(7\boldsymbol{n} - 1)$$

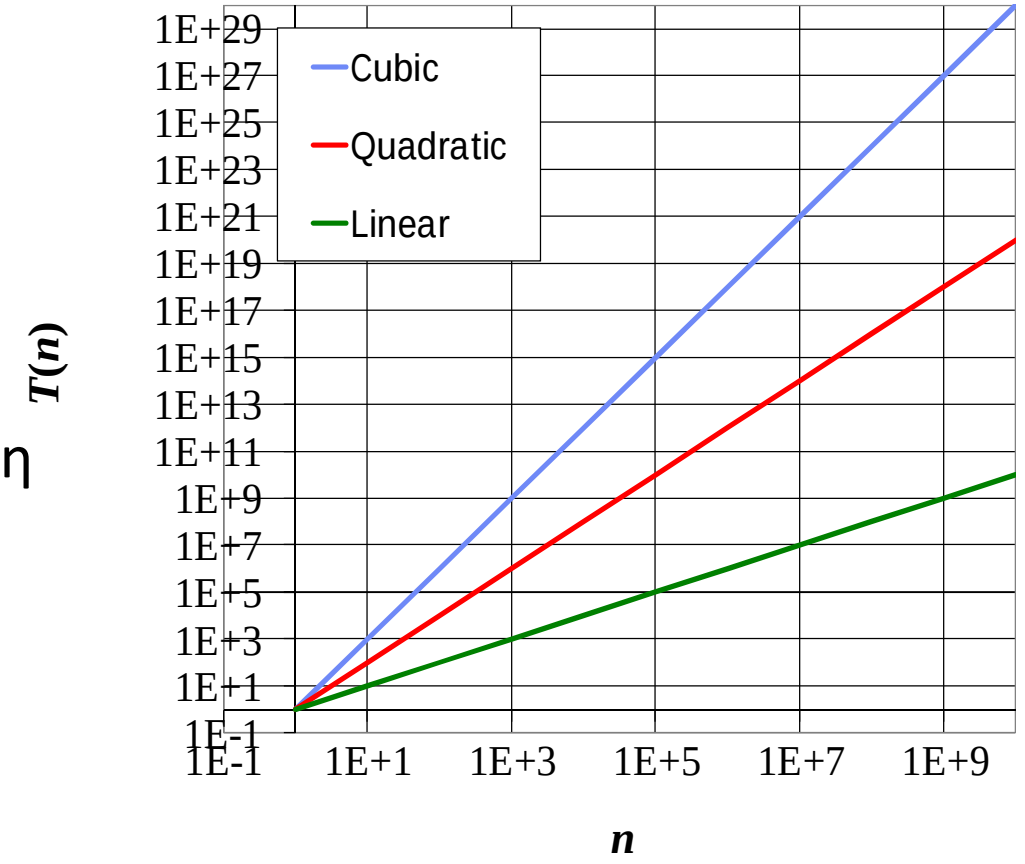
- Επομένως, ο χρόνος εκτέλεσης $\boldsymbol{T}(\boldsymbol{n})$ έχει όρια δυο γραμμικές συναρτήσεις

Ρυθμός Αύξησης του Χρόνου Εκτέλεσης

- Αλλάζοντας το περιβάλλον υλικού/λογισμικού
  - Επηρεάζεται ο $T(n)$ κατά ένα σταθερό όρο, αλλά
  - Δεν αλλάζει ο ρυθμός αύξησης του $T(n)$
- Ο γραμμικός ρυθμός αύξησης του χρόνου εκτέλεσης $T(n)$ είναι μια εγγενής ιδιότητα του αλγορίθμου ***arrayMax***

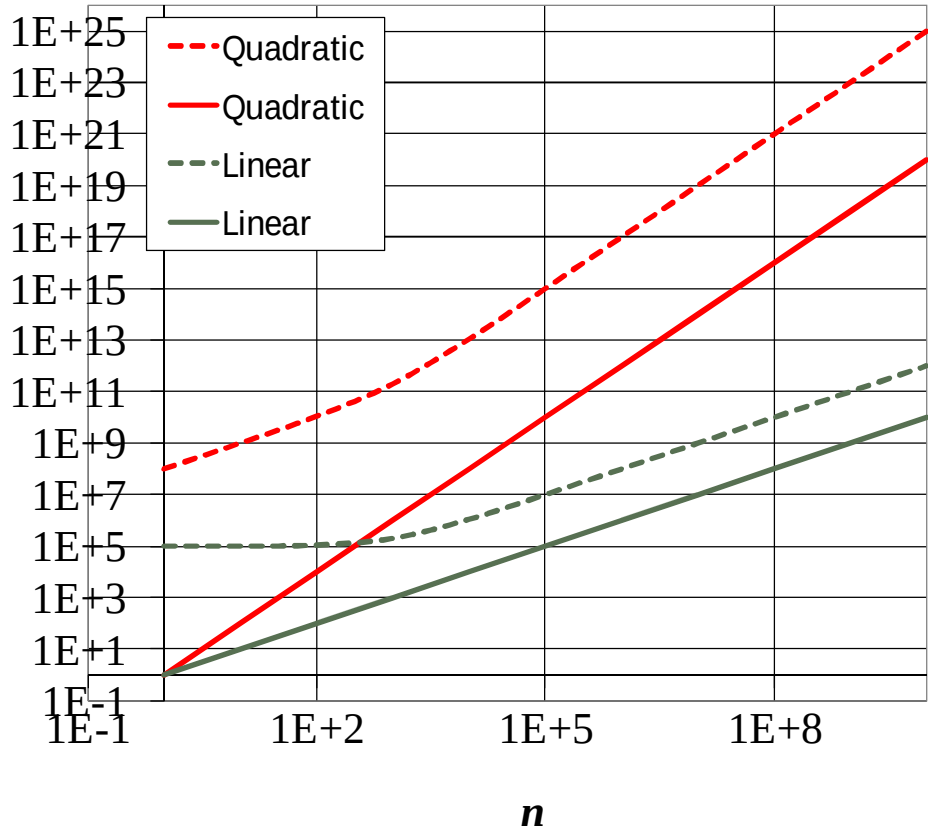
Ρυθμοί Αύξησης

- Ρυθμοί αύξησης συναρτήσεων:
  - Γραμμικός $\approx n$
  - Τετραγωνικός $\approx n^2$
  - Κυβικός $\approx n^3$
- Σε μια γραφική παράσταση log-log, η κλίση της γραμμής αντιστοιχεί στον ρυθμό αύξησης της συνάρτησης



Σταθεροί Όροι

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

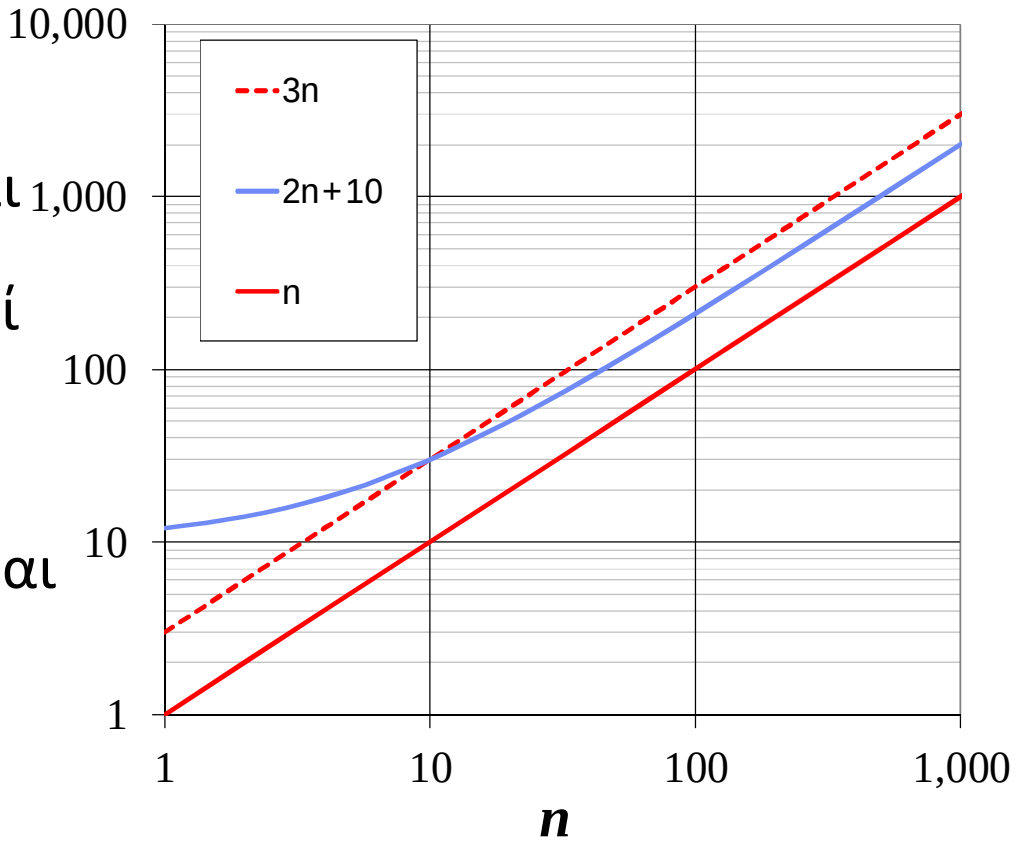


Σημειογραφία O()

- Δεδομένου των συναρτήσεων $f(n)$ και $g(n)$, λέμε ότι η $f(n)$ είναι $O(g(n))$ αν υπάρχουν θετικοί σταθεροί αριθμοί c and n_0 τέτοιοι ώστε

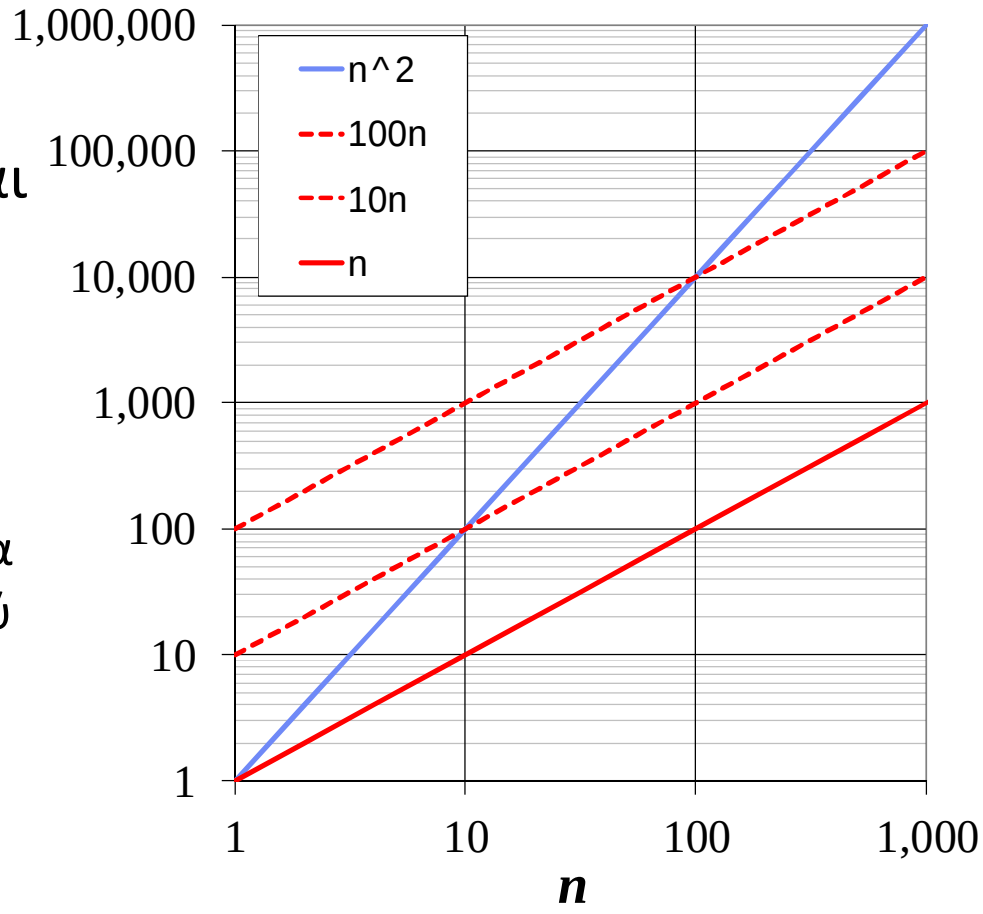
  $f(n) \leq cg(n)$ για $n \geq n_0$

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



Σημειογραφία O() (συνέχεια)

- Παράδειγμα: η συνάρτηση n^2 δεν είναι $O(n)$
  - $n^2 \leq cn$
  - $n \leq c$
  - Η παραπάνω ανισότητα δεν ικανοποιείται αφού ο c πρέπει να είναι σταθερός



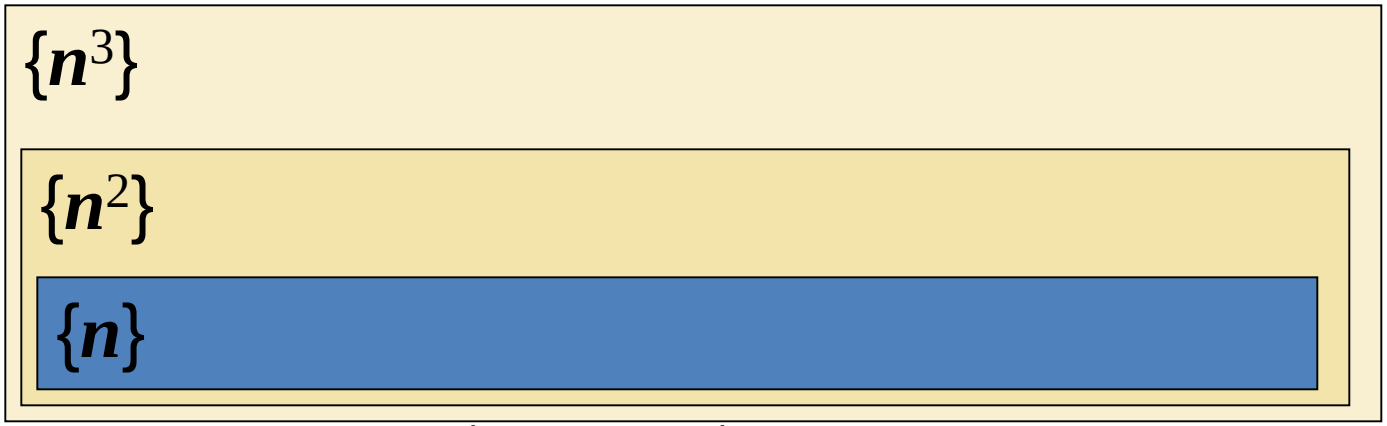
O() και Ρυθμός Αύξησης

- Η σημειογραφία O() θέτει ένα άνω όριο στο ρυθμό αύξησης μιας συνάρτησης
- Η δήλωση "η $f(n)$ είναι $O(g(n))$" σημαίνει πως ο ρυθμός αύξησης της $f(n)$ είναι το πολύ ίσος με $g(n)$
- Μπορούμε να χρησιμοποιήσουμε την σημειογραφία O() για να κατατάξουμε συναρτήσεις σύμφωνα με τον ρυθμό αύξησης τους

	η $f(n)$ είναι $O(g(n))$	η $g(n)$ είναι $O(f(n))$
ρ.α. $g(n)$ > ρ.α. $f(n)$	Ναι	Όχι
ρ.α. $f(n)$ > ρ.α. $g(n)$	Όχι	Ναι
Ίδιος ρυθμ. αύξησης	Ναι	Ναι

Κατηγορίες Συναρτήσεων

- Έστω ότι με $\{g(n)\}$ δηλώνουμε την κατηγορία (το σύνολο) των συναρτήσεων οι οποίες είναι $O(g(n))$
- Έχουμε
  $\{n\} \subset \{n^2\} \subset \{n^3\} \subset \{n^4\} \subset \{n^5\} \subset \ldots$
  όπου η σχέση περιεχομένου είναι αυστηρή



Κανόνες O()

- Αν το $f(n)$ είναι ένα πολυώνυμο βαθμού d, τότε $f(n)$ είναι $O(n^d)$, δηλαδή,
  1. Αφαιρούμε όρους χαμηλότερου βαθμού
  2. Αφαιρούμε σταθερούς όρους
- Πρέπει να χρησιμοποιήσουμε την μικρότερη δυνατή κατηγορία συναρτήσεων
  - Λέμε ότι "η $2n$ είναι $O(n)$" αντί για "η $2n$ είναι $O(n^2)$"
- Πρέπει να χρησιμοποιήσουμε την απλούστερη έκφραση της κατηγορίας
  - Λέμε ότι "η $3n + 5$ είναι $O(n)$" αντί για "η $3n + 5$ είναι $O(3n)$"

Ασυμπτωτική Ανάλυση Αλγορίθμου

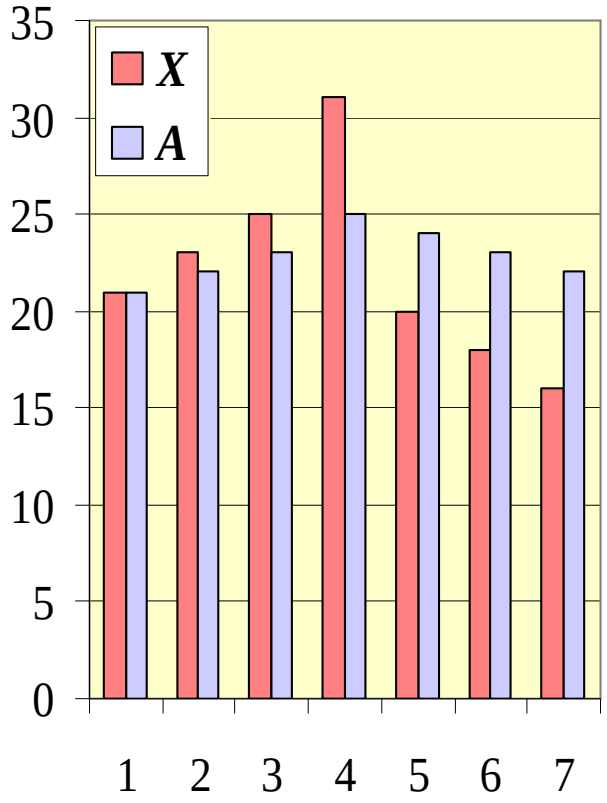
- Η ασυμπτωτική ανάλυση ενός αλγορίθμου καθορίζει τον χρόνο εκτέλεσης σε σημειογραφία O()
- Για να κάνουμε την ασυμπτωτική ανάλυση
  - Βρίσκουμε τον αριθμό των στοιχειωδών πράξεων που εκτελούνται στην χειρότερη περίπτωση σαν συνάρτηση του μεγέθους της εισόδου
  - Εκφράζουμε αυτή τη συνάρτηση σε σημειογραφία O()
- Παράδειγμα:
  - Προσδιορίσαμε ότι ο αλγόριθμος ***arrayMax*** εκτελεί το πολύ $7\boldsymbol{n} - 1$ στοιχειώδεις πράξεις
  - Λέμε ότι ο αλγόριθμος ***arrayMax*** "τρέχει σε χρόνο $\boldsymbol{O}(\boldsymbol{n})$"
- Αφού σταθεροί όροι και όροι χαμηλότερης τάξης εντέλει παραλείπονται, μπορούμε να τους παραβλέψουμε κατά τη μέτρηση των στοιχ. πράξεων

Υπολογίζοντας Μέσους Όρους Προθέματος (Prefix Averages)

- Θα επεξηγήσουμε περαιτέρω την ασυμπτωτική ανάλυση χρησιμοποιώντας δυο αλγορίθμους για μέσους όρους προθέματος
- Ο i-οστός μέσος όρος προθέματος ενός πίνακα X είναι ο μέσος όρος των πρώτων $(i + 1)$ στοιχείων του X

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

- Ο υπολογισμός του πίνακα A ο οποίος περιέχει τους μέσους όρους προθέματος ενός άλλου πίνακα X έχει εφαρμογές στην οικονομική ανάλυση



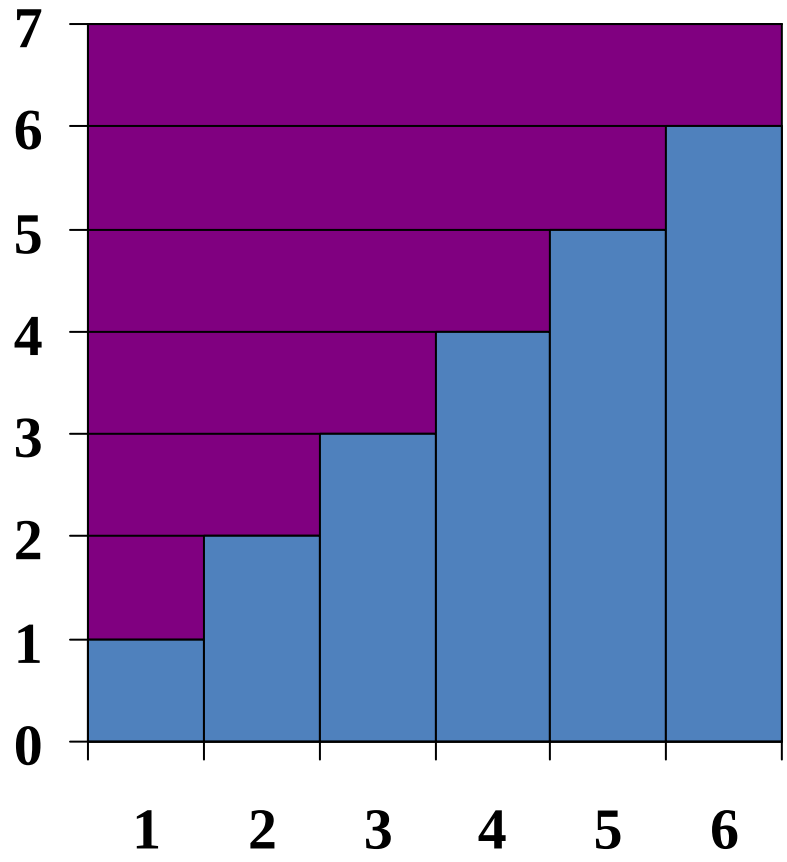
Μέσοι Όροι Προθέματος (Τετραγωνικός χρόνος)

- Ο ακόλουθος αλγόριθμος υπολογίζει τους μέσους όρους 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**	$1 + 2 + \ldots + (n - 1)$
s ← *s* + *X*[*j*]	$1 + 2 + \ldots + (n - 1)$
A[*i*] ← *s* / (*i* + 1)	*n*
return *A*	1

Αριθμητική Πρόοδος

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



Μέσοι Όροι Προθέματος (Γραμμικός χρόνος)

- Ο ακόλουθος αλγόριθμος υπολογίζει μέσους όρους προθέματος 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                                      1
```

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

Τέλος Ενότητας

BY NC ND

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

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

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης