ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΤΟΠΟΓΡΑΦΙΑΣ ΚΑΙ ΓΕΩΠΛΗΡΟΦΟΡΙΚΗΣ

Πληροφορική και Προγραμματισμός

Διδάσκων
Αναστάσιος Κεσίδης

Αλγόριθμοι και πολυπλοκότητα

Επίλυση προβλημάτων

- Επίλυση προβλημάτων με χρήση υπολογιστικών συστημάτων
  - Δυσκολία στην επίλυση **αφηρημένων εννοιών** (π.χ. απόδειξη θεωρημάτων, γεωμετρικές και μαθηματικές έννοιες)
  - Προτερήματα: **ακρίβεια** και **ταχύτητα** υπολογισμών

    - Απαιτείται η διατύπωση της μεθόδου επίλυσης του προβλήματος σε μορφή απλών βημάτων

- Βασικά στάδια
  - **Ορισμός** του προβλήματος
  - Ανάπτυξη ενός **αλγόριθμου** για την επίλυση του προβλήματος
  - **Προγραμματισμός** του αλγόριθμου χρησιμοποιώντας κάποια από τις γλώσσες προγραμματισμού

Στάδια επίλυσης προβλήματος

Ορισμός του προβλήματος

- Μαθηματική διατύπωση για να μπορέσουμε να το χειριστούμε αναλυτικά (**μαθηματικό μοντέλο**)

Παράδειγμα προβλήματος

*Υπολογισμός του **μέγιστου** στοιχείου από ένα σύνολο ακεραίων αριθμών*

Πχ. Για το σύνολο $\{8, 6, \mathbf{12}, -3, 5, 9, -10, -4, 7\}$ να βρεθεί η τιμή $\mathbf{12}$

Το πρόβλημα μπορεί να διατυπωθεί και μαθηματικά ως εξής:

Δεδομένου ενός συνόλου ακεραίων αριθμών

$$A = \{a_1, a_2, \dots, a_n\} \in Z$$

να υπολογιστεί το

$$x = \max\{y : y \in A\}$$

Στάδια επίλυσης προβλήματος

Αλγόριθμος

Αλγόριθμος είναι ένα πεπερασμένο σύνολο οδηγιών για την πραγματοποίηση των υπολογισμών που απαιτούνται με σκοπό την επίλυση ενός προβλήματος

Χαρακτηριστικά

- **Ορισμός** δεδομένων εισόδου/εξόδου
- **Σαφήνεια** για κάθε βήμα του αλγορίθμου
- **Πεπερασμένα** βήματα υπολογισμού
- **Υπολογιστική πολυπλοκότητα** κάθε βήματος υπολογισμού
- **Γενίκευση** για ένα σύνολο προβλημάτων παρόμοιου τύπου

Στάδια επίλυσης προβλήματος

- Αλγόριθμος - Παράδειγμα

Δεδομένου ενός συνόλου ακεραίων αριθμών

$$A = \{a_1, a_2, \dots, a_n\} \in Z$$

να υπολογιστεί το

$$x = \max\{y: y \in A\}$$

1ο Βήμα

Όρισε σαν μέγιστο αριθμό x τον πρώτο ακέραιο του συνόλου A

2ο Βήμα

Σύγκρινε τον επόμενο αριθμό του A με τον x και εάν είναι μεγαλύτερος από τον x τότε αντικατέστησε την τιμή του x με αυτόν.

3ο Βήμα

Εάν υπάρχουν και άλλοι αριθμοί στο σύνολο A τότε επανέλαβε το Βήμα 2 και για τους υπόλοιπους αριθμούς, αλλιώς το x είναι ο μεγαλύτερος ακέραιος αριθμός του συνόλου.

Στάδια επίλυσης προβλήματος

- Ψευδοκώδικας

  Διατηρεί την **γενική δομή** και **σύνταξη** που ακολουθούν οι περισσότερες γλώσσες προγραμματισμού

  Δεν περιλαμβάνει τεχνικές λεπτομέρειες που δεν είναι απαραίτητες για την κατανόηση του αλγόριθμου.

  Περιγράφει τον αλγόριθμο με αρκετά ακριβή τρόπο ώστε να μην υπάρχει αμφιβολία για την λειτουργία του **χωρίς** να μπαίνει σε προγραμματιστικές λεπτομέρειες που αφορούν κάποια **συγκεκριμένη** γλώσσα προγραμματισμού.

Στάδια επίλυσης προβλήματος

Ψευδοκώδικας - Παράδειγμα

Δεδομένου ενός συνόλου ακεραίων αριθμών

$$A = \{a_1, a_2, \dots, a_n\} \in Z$$

να υπολογιστεί το

$$x = \max\{y: y \in A\}$$

Ψευδοκώδικας

Είσοδος: Το σύνολο $A = \{a_1, a_2, \dots, a_n\}$
Έξοδος: Το μέγιστο στοιχείο x του A

$x = a_1$
for $i = 2, \dots, n$
  if $a_i > x$
    $x = a_i$
  end if
end for
return x

Πολυπλοκότητα

Συμβολισμοί O, Ω και Θ

Για κάποιο υπολογιστικό πρόβλημα μπορεί να υπάρχουν διάφοροι αλγόριθμοι που να το επιλύουν επιτυχώς. Όμως, οι **απαιτήσεις χρόνου ή/και μνήμης** μπορεί να διαφέρουν σημαντικά.

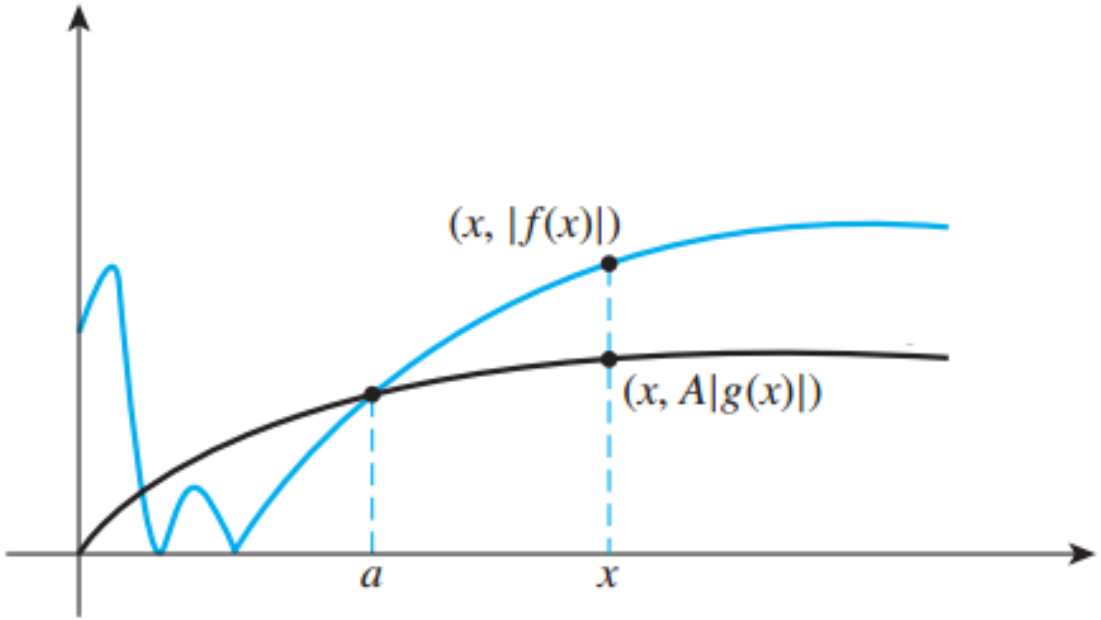
Οι συμβολισμοί O, Ω και Θ παρέχουν προσεγγίσεις οι οποίες διευκολύνουν τον υπολογισμό της αποτελεσματικότητας κατά την σύγκριση αλγορίθμων σε δεδομένα μεγάλης κλίμακας.

- Αγνοούν διαφορές που οφείλονται σε μια σταθερή ποσότητα
- Αγνοούν διαφορές που προκύπτουν σε μικρής κλίμακας δεδομένα (μικρά σύνολα δεδομένων εισόδου)

Πολυπλοκότητα

➢ Τάξη $\Omega(n)$: Ασυμπτωτικό κάτω όριο

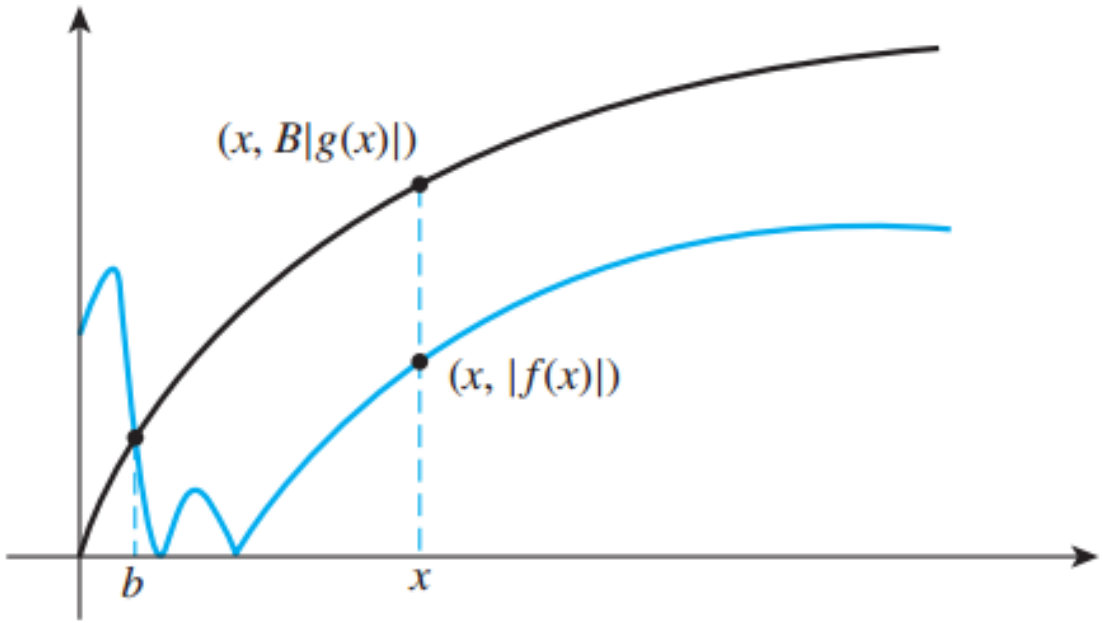
Το ασυμπτωτικό κάτω όριο καθορίζεται από μια **συνάρτηση** $\mathbf{g}(\boldsymbol{x})$ που συνήθως είναι πολύ **απλούστερη** της $f(x)$ και οι τιμές της $f(x)$ είναι **μεγαλύτερες** από κάποιο πολλαπλάσιο A της $g(x)$ για x μεγαλύτερο από κάποιο a.



Πολυπλοκότητα

- Τάξη $O(n)$: Ασυμπτωτικό άνω όριο

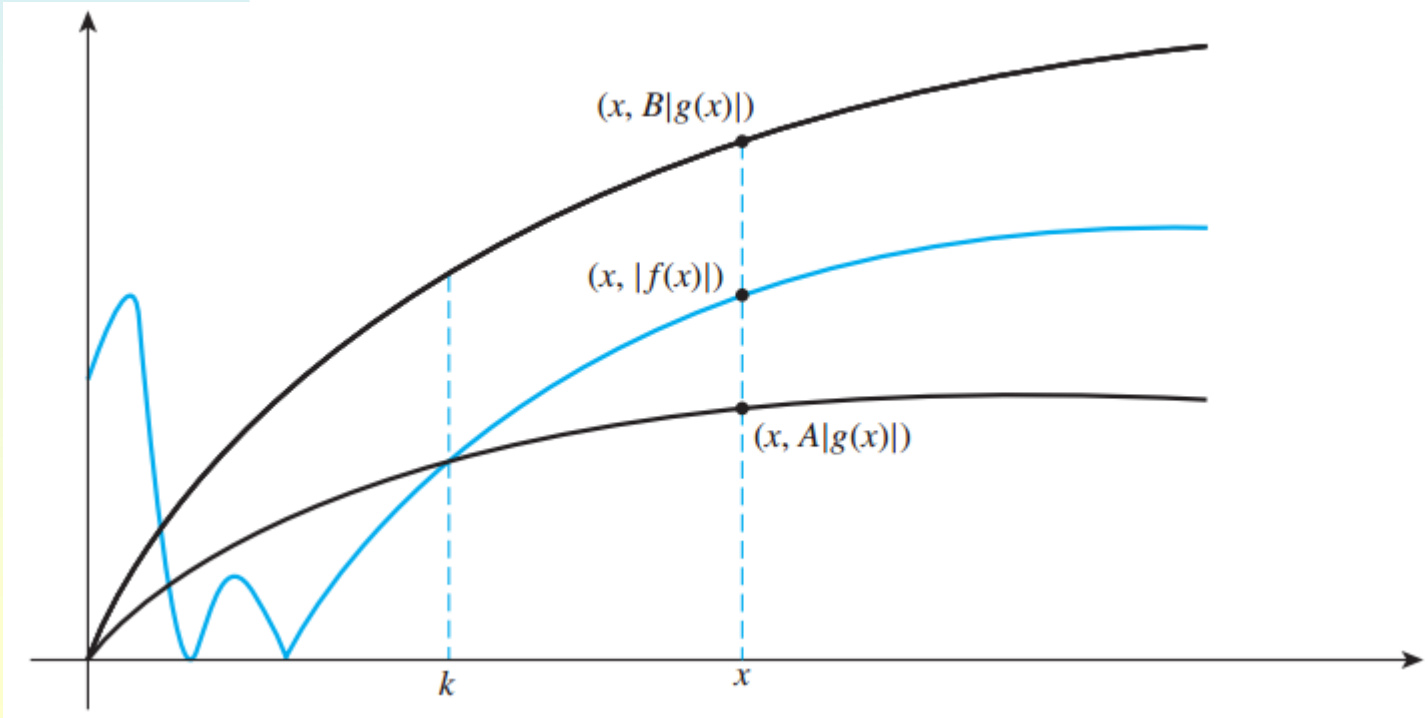
Το ασυμπτωτικό άνω όριο καθορίζεται από μια **συνάρτηση** $\mathbf{g}(\boldsymbol{x})$ που συνήθως είναι πολύ **απλούστερη** της $f(x)$ και οι τιμές της $f(x)$ είναι **μικρότερες** από κάποιο πολλαπλάσιο B της $g(x)$ για x μεγαλύτερο από κάποιο b.



Πολυπλοκότητα

- Τάξη $\Theta(n)$: Αυστηρά ασυμπτωτικά όρια

Καθορίζονται από μια **συνάρτηση** $\mathbf{g}(\boldsymbol{x})$ που συνήθως είναι πολύ **απλούστερη** της $f(x)$ και οι τιμές της $f(x)$ είναι **μεγαλύτερες** και αντίστοιχα **μικρότερες** από κάποια πολλαπλάσια της $g(x)$ για x μεγαλύτερο από κάποιο k.



Πολυπλοκότητα $O(n)$

- Έκφραση πολυπλοκότητας μέσω της τάξης $O(n)$

Ορισμός

Έστω f και g συναρτήσεις από το σύνολο των ακεραίων ή πραγματικών αριθμών προς το σύνολο των πραγματικών αριθμών.

Λέμε ότι η συνάρτηση $\boldsymbol{f(x)}$ είναι **τάξης το πολύ** $\boldsymbol{O(g(x))}$ εάν υπάρχουν σταθερές $\boldsymbol{C}$ και $\boldsymbol{k}$ για τις οποίες να ισχύει

$$|f(x)| \leq C|g(x)| \text{ για κάθε } x > k$$

Απλοποίηση Μπορούμε να θεωρήσουμε ότι οι $f(x)$ και $g(x)$ είναι πάντα θετικές οπότε ο ορισμός της τάξης $\boldsymbol{O(g(x))}$ απλοποιείται σε

$$f(x) \leq Cg(x) \text{ για κάθε } x > k$$

Συνεπώς,

Για να δείξουμε ότι η $f(x)$ είναι τάξης $O(g(x))$ αρκεί να βρούμε **ένα** ζεύγος τιμών $(\boldsymbol{C}, \boldsymbol{k})$ για τις οποίες να ισχύει η παραπάνω σχέση.

Πολυπλοκότητα $O(n)$

- Τάξη της συνάρτησης δύναμης

Για

$$1 < x$$

είναι $x < x^2$

και $x^2 < x^3$, κλπ

Οπότε $1 < x < x^2 < x^3 \ldots$

Για οποιουδήποτε ρητούς αριθμούς r, s ισχύει

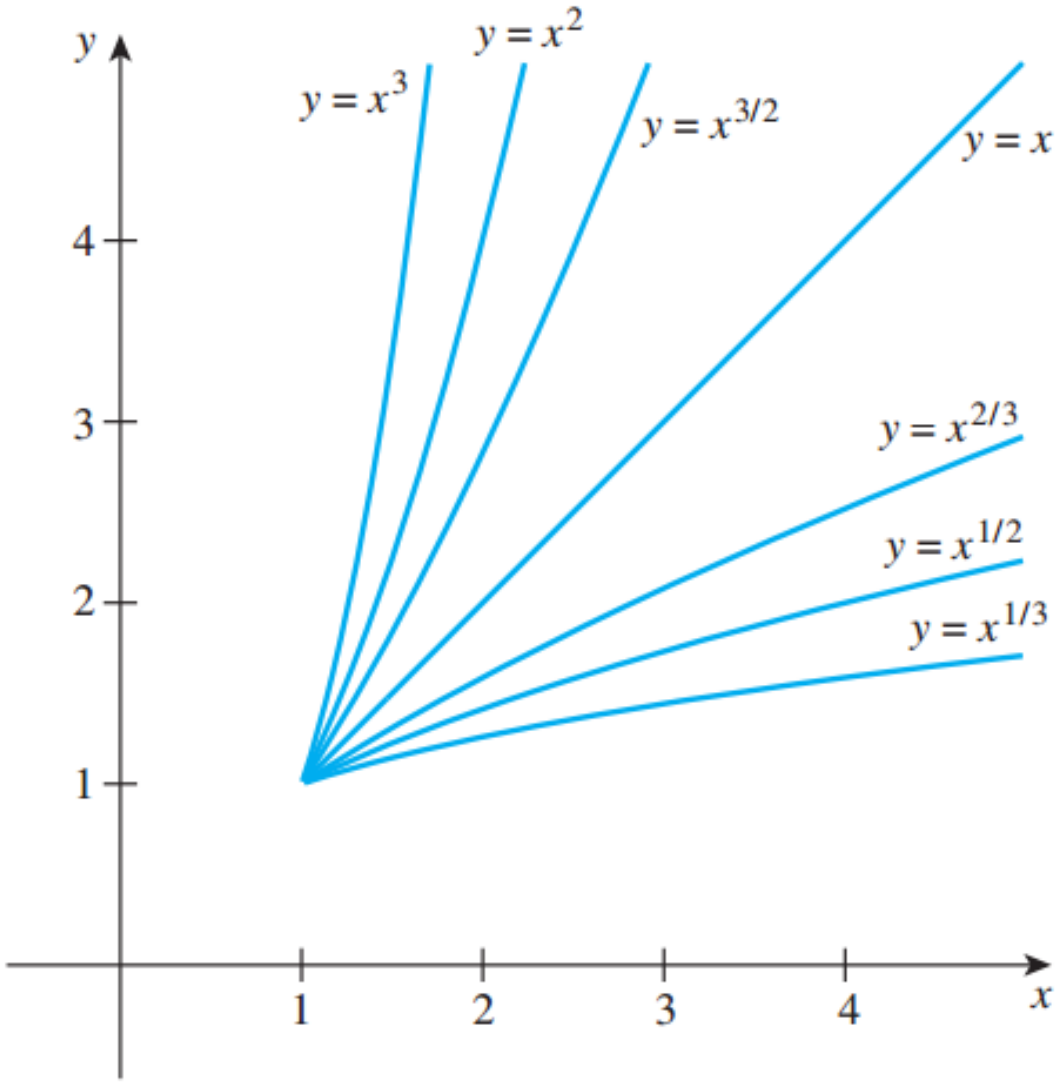
Εάν $x > 1$ και $r < s$, τότε $\boldsymbol{x^r < x^s}$

Συνέπεια αυτού είναι ότι για οποιουδήποτε ρητούς αριθμούς r, s

Εάν $r < s$ τότε και $\boldsymbol{x^r}$ **είναι τάξης** $\boldsymbol{O(x^s)}$

Πολυπλοκότητα $O(n)$

- Συνάρτηση δύναμης για διάφορες θετικές δυνάμεις



Πολυπλοκότητα $O(n)$

Πολυωνυμικές συναρτήσεις

Εάν η συνάρτηση $f(x)$ είναι **πολυώνυμο βαθμού n**, δηλαδή

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

όπου $a_0, a_1, a_2, \ldots, a_n$ πραγματικοί αριθμοί και $\boldsymbol{a_n \neq 0}$ τότε

- η $f(x)$ είναι $O(x^s)$ για κάθε ακέραιο $s \geq n$
- η $f(x)$ είναι $\Omega(x^r)$ για κάθε ακέραιο $r \leq n$
- η $f(x)$ είναι $\Theta(x^s)$

Πολυπλοκότητα $O(n)$

- Πολυωνυμικές συναρτήσεις - παράδειγμα

Δείξτε ότι η συνάρτηση $f(x) = x^2 + 2x + 1$ *είναι τάξης* $O(x^2)$

Λύση

Για $x > 1$ είναι

$$x^2 + 2x + 1 < x^2 + 2x^2 + x^2 < 4x^2$$

Αφού για $x > 1$ είναι $2x < 2x^2$ και $1 < x^2$

Συνεπώς, στον ορισμό της τάξης $O(g(x))$ ως

$f(x) \leq Cg(x)$ για κάθε $x > k$

είναι

$g(x) = x^2, C = 4, k = 1$

Συνεπώς,

η $f(x)$ είναι $O(x^2)$

Πολυπλοκότητα $O(n)$

- Συναρτήσεις ακεραίων μεταβλητών

  Συνήθως για την αναπαράσταση μιας πραγματικής μεταβλητής χρησιμοποιείται το σύμβολο $\boldsymbol{x}$ ενώ για ακέραιες μεταβλητές χρησιμοποιείται το σύμβολο $\boldsymbol{n}$.

  Συνεπώς, στον ορισμό της τάξης με την μορφή

  $$\boldsymbol{f(n)} \text{ είναι τάξης } \boldsymbol{O(g(n))}$$

  υπονοείται ότι οι συναρτήσεις f και g ορίζονται στο σύνολο των ακεραίων.

  Επίσης, εάν η $f(x)$ είναι $O(g(x))$ για όλους τους πραγματικούς αριθμούς $x > k$ τότε ισχύει ότι και η $f(n)$ είναι $O(g(n))$ για όλους τους ακεραίους $n > k$

- Παράδειγμα: το άθροισμα των πρώτων n ακεραίων

  Το άθροισμα $1 + 2 + 3 + \cdots + n$ είναι τάξης $O(n^2)$

Αποτελεσματικότητα αλγορίθμων

- Παράγοντες που καθορίζουν την αποτελεσματικότητα

Ο **χρόνος** που απαιτείται για την εκτέλεση του αλγορίθμου

Η **μνήμη** που απαιτείται για την εκτέλεση του αλγορίθμου

- Υπάρχουν αλγόριθμοι που είναι πιο αποτελεσματικοί ως προς τον χρόνο εκτέλεσης αλλά λιγότερο αποτελεσματικοί ως προς την μνήμη και αντιστρόφως.
- Η επιλογή του κατάλληλου αλγορίθμου εξαρτάται από το **πρόβλημα** και τις **απαιτήσεις** του χρήστη.

Αποτελεσματικότητα αλγορίθμων

- Εκτίμηση της αποτελεσματικότητας ενός αλγορίθμου

Υπολογισμός του **πλήθους** των βασικών πράξεων που εκτελούνται όταν ο αλγόριθμος εκτελεστεί για δεδομένα εισόδου μεγέθους $\boldsymbol{n}$

Ως **βασικές πράξεις** θεωρούνται οι:

- Πρόσθεση
- Αφαίρεση
- Πολλαπλασιασμός
- Διαίρεση
- Σύγκριση μέσω των βασικών τελεστών $=, \neq, <, \leq, >, \geq$

Αποτελεσματικότητα αλγορίθμων

➢ Παράδειγμα σύγκρισης δύο αλγορίθμων

Έστω δύο αλγόριθμοι **A** και **B** που επιλύουν το ίδιο πρόβλημα.

Για δεδομένα εισόδου μεγέθους $\boldsymbol{n}$ ο αλγόριθμος A απαιτεί $\mathbf{5000}\boldsymbol{n}$ βασικές πράξεις ενώ ο αλγόριθμος B απαιτεί $\lceil \mathbf{1.1}^{\boldsymbol{n}} \rceil$ βασικές πράξεις.

- Για $n = 10$

  ο **αλγόριθμος A** απαιτεί $5000 \cdot 10 = 50.000$ πράξεις

  ο **αλγόριθμος B** απαιτεί$\lceil 1.1^{10} \rceil = \lceil 2.5937 \rceil \cong 3$ πράξεις

- Για $n = 1000$

  ο **αλγόριθμος A** απαιτεί $5000 \cdot 1000 = 5.000.000$ πράξεις

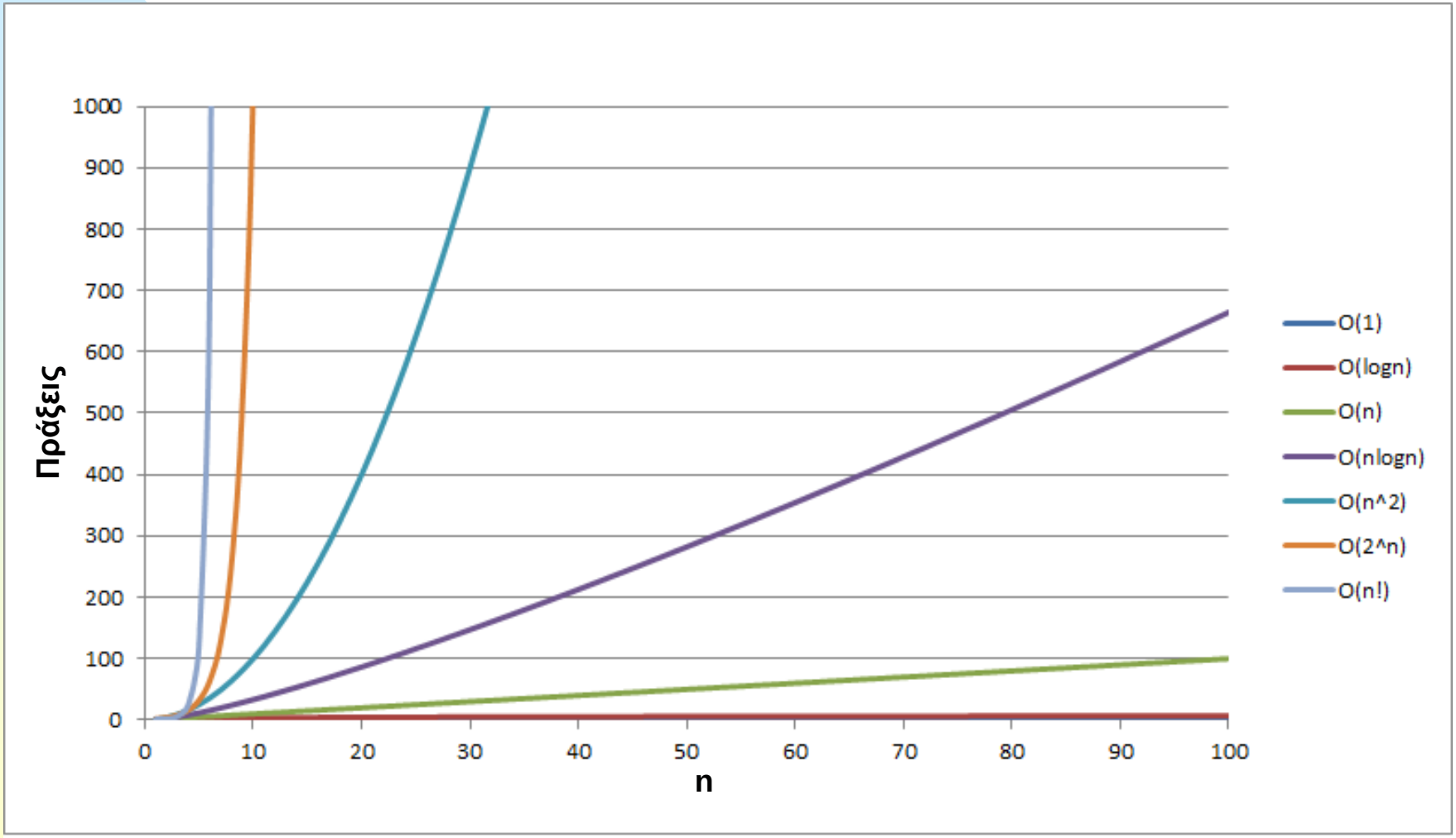
  ο **αλγόριθμος B** απαιτεί $2.5 \cdot 10^{41}$ πράξεις

Συνεπώς:

Ο **αλγόριθμος A** μπορεί να χρησιμοποιηθεί και σε μεγάλο πλήθος δεδομένων ενώ ο **αλγόριθμος B** όχι.

Αποτελεσματικότητα αλγορίθμων

- Συγκριτική πολυπλοκότητα των βασικών τάξης αλγορίθμων



Αποτελεσματικότητα αλγορίθμων

- Σύγκριση χρόνου εκτέλεσης βασικών τάξεων αλγορίθμων

Θεωρούμε ότι κάθε πράξη απαιτεί 1 nanosecond=10^{-9} sec.

$f(n)$	$n = 10$	$n = 1{,}000$	$n = 100{,}000$	$n = 10{,}000{,}000$
$\log_2 n$	3.3×10^{-9} sec	10^{-8} sec	1.7×10^{-8} sec	2.3×10^{-8} sec
n	10^{-8} sec	10^{-6} sec	0.0001 sec	0.01 sec
$n \log_2 n$	3.3×10^{-8} sec	10^{-5} sec	0.0017 sec	0.23 sec
n^2	10^{-7} sec	0.001 sec	10 sec	27.8 min
n^3	10^{-6} sec	1 sec	11.6 days	31,688 yr
2^n	10^{-6} sec	3.4×10^{284} yr	3.1×10^{30086} yr	2.9×10^{3010283} yr

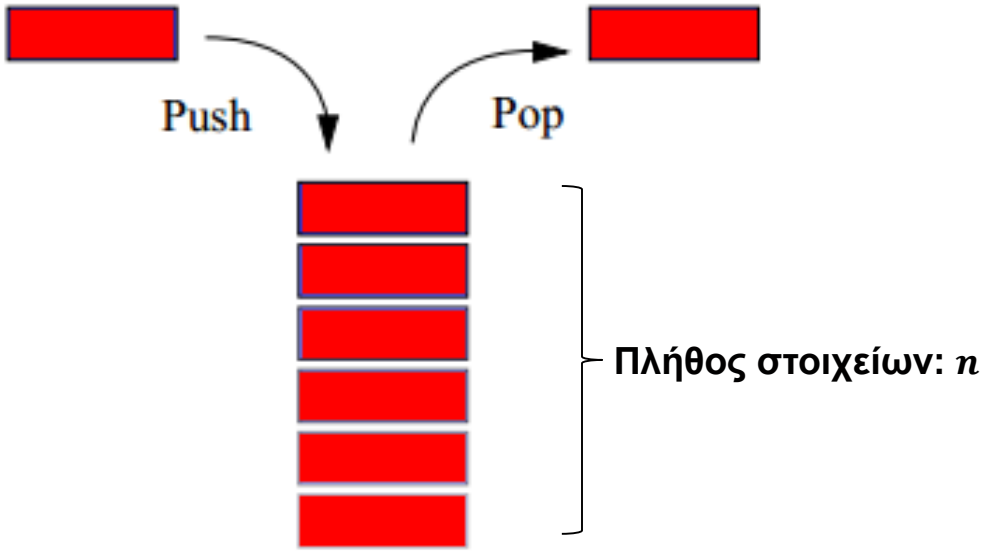
Παραδείγματα

- Σταθερός χρόνος $O(1)$

Σταθερός χρόνος είναι ο χρόνος που είναι ανεξάρτητος από το μέγεθος της εισόδου $\boldsymbol{n}$.

Λειτουργίες στοίβας

Απαιτείται σταθερός χρόνος $O(1)$ που είναι ανεξάρτητος από το μέγεθος $\boldsymbol{n}$ των δεδομένων.



Παραδείγματα

- Γραμμικός χρόνος $O(n)$

Ο χρόνος εκτέλεσης είναι ανάλογος με το μέγεθος της εισόδου $\boldsymbol{n}$.

Υπολογισμός μεγίστου

Ψευδοκώδικας

Είσοδος: Το σύνολο $A = \{a_1, a_2, \dots, a_n\}$
Έξοδος: Το μέγιστο στοιχείο x του A

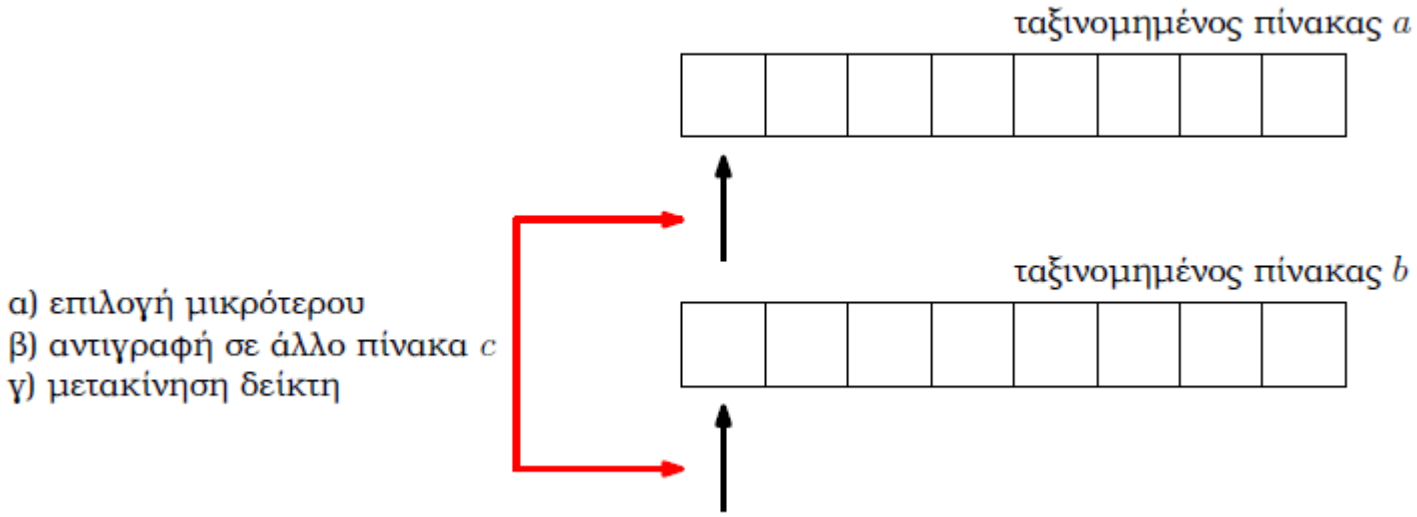
$x = a_1$
for $i = 2, \dots, n$
  if $a_i > x$
    $x = a_i$
  end if
end for
return x

Παραδείγματα

- Γραμμικός χρόνος $O(n)$

Συγχώνευση ταξινομημένων λιστών

Συνδυασμός δύο ταξινομημένων λιστών $A = \{a_1, a_2, \dots, a_n\}$ και $B = \{b_1, b_2, \dots, b_n\}$ σε μια ενιαία ταξινομημένη λίστα c.



Παραδείγματα

- Γραμμικός χρόνος $O(n)$

Συγχώνευση ταξινομημένων λιστών (συν.)

Ψευδοκώδικας

Είσοδος: Ταξινομημένες λίστες A και B
Έξοδος: Ταξινομημένη λίστα C

$i = 1, j = 1$
while καμία άδεια λίστα
  if $a_i \leq b_j$
    πρόσθεσε το a_i στην λίστα C και αύξησε το i
  else
    πρόσθεσε το b_j στην λίστα C και αύξησε το j
  end if
end while
πρόσθεσε όλα τα στοιχεία της μη άδειας λίστας στην λίστα C
return C

Παραδείγματα

- Γραμμικός χρόνος $O(n)$

Σειριακή αναζήτηση

Αναζήτηση σε μια ακολουθία αριθμών σειριακά με σκοπό την εύρεση ενός συγκεκριμένου στοιχείου.

Ψευδοκώδικας

Είσοδος: Το σύνολο $A = \{a_1, a_2, \dots, a_n\}$ και ο αναζητούμενος αριθμός x
Έξοδος: η θέση όπου βρέθηκε το x ή 0 εάν δεν βρέθηκε

```
i = 1
while i ≤ n
    if x = a_i then
        return i
    end if
    i = i + 1
end while
return 0
```

Παραδείγματα

- Γραμμικός χρόνος $O(n)$

Σειριακή αναζήτηση (συν.)

Αναζήτηση σε μια ακολουθία αριθμών σειριακά με σκοπό την εύρεση ενός συγκεκριμένου στοιχείου.

Παράδειγμα

Εύρεση της **θέσης** του αριθμού $\mathbf{9}$ μέσα στο σύνολο

$$\{8, 6, 12, -3, 5, \mathbf{9}, -10, -4, 7\}$$

Καλύτερη περίπτωση

Το αναζητούμενο στοιχείο να είναι το 1ο στοιχείο του συνόλου των αριθμών. Πολυπλοκότητα $O(1)$.

Χειρότερη περίπτωση

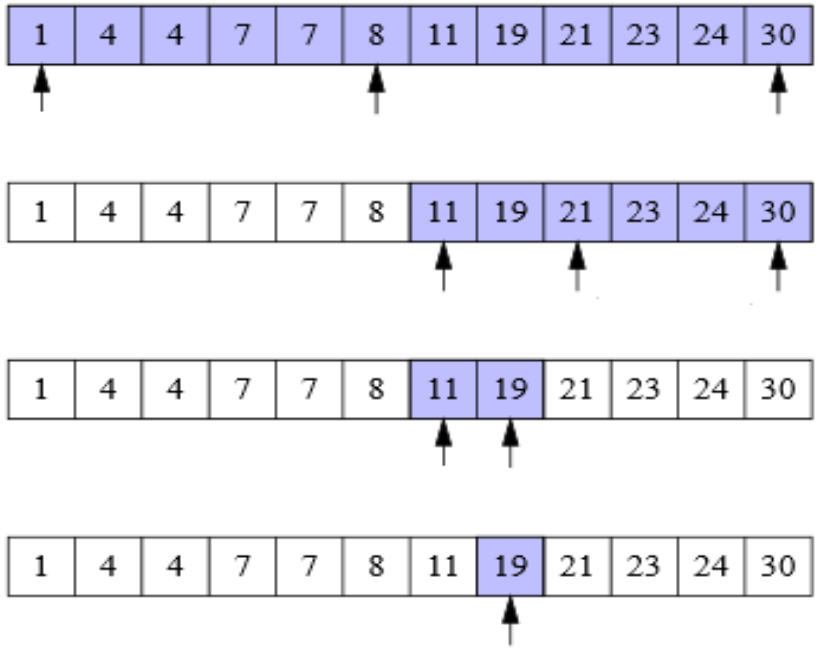
Το αναζητούμενο στοιχείο να είναι το τελευταίο στοιχείο του συνόλου των αριθμών. Πολυπλοκότητα $O(n)$.

Παραδείγματα

- Χρόνος $O(logn)$

Δυαδική αναζήτηση

Εάν τα δεδομένα είναι ταξινομημένα (π.χ. κατά αύξουσα σειρά) τότε η δυαδική αναζήτηση είναι πιο αποτελεσματική από την σειριακή.
Μέθοδος: **Διαδοχικός υποδιπλασιασμός του εύρους αναζήτησης** ώσπου να καταλήξει στην αναζητούμενη τιμή (εφόσον υπάρχει).



Παραδείγματα

- Χρόνος $O(logn)$

Δυαδική αναζήτηση (συν.)

Ψευδοκώδικας

Είσοδος: Το σύνολο $A = \{a_1, a_2, \dots, a_n\}$ και ο αναζητούμενος αριθμός x
Έξοδος: η θέση όπου βρέθηκε το x ή 0 εάν δεν βρέθηκε

```
i = 1
j = n
while ( i ≤ j)
    m = ⌊(i+j)/2⌋
    if x = a_m
        return m
    elseif x < a_m
        j = m − 1
    else
        i = m + 1
    end if
end while
return 0
```

Παραδείγματα

- Τετραγωνικός χρόνος $O(n^2)$

Απαρίθμηση όλων των συνδυασμών ενός συνόλου στοιχείων

Εύρεση κοντινότερου ζεύγους σημείων

Με είσοδο τις συντεταγμένες n σημείων στο επίπεδο $(x_1, y_1), \dots, (x_n, y_n)$, θέλουμε να βρούμε το ζεύγος σημείων που έχουν την μικρότερη απόσταση μεταξύ τους.

Ψευδοκώδικας

Είσοδος: Το σύνολο των n σημείων $(x_1, y_1), \dots, (x_n, y_n)$
Έξοδος: η μικρότερη απόσταση

for $i = 1 \dots n-1$
  for $j = 2 \dots n$
    $d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$
    if $d < min$
      $min = d$
    end if
  end for
end for

Παραδείγματα

- Τετραγωνικός χρόνος $O(n^2)$

Ταξινόμηση με εισαγωγή (Insert sort)

Ταξινόμηση των στοιχείων ενός συνόλου αριθμών $A = \{a_1, a_2, \dots, a_n\}$ σε αύξουσα σειρά.

Ο αλγόριθμος επαναλαμβάνεται $\boldsymbol{n}$**-1** φορές, όπου n το πλήθος των στοιχείων του συνόλου.

Για κάθε επανάληψη k το στοιχείο a_k **συγκρίνεται με όλα τα προηγούμενα** a_j με $1 \leq j \leq k-1$, με φθίνουσα σειρά για το j.

Εάν το εξεταζόμενο a_j είναι μεγαλύτερο από το a_k τότε η θέση του στοιχείου a_j αυξάνει κατά ένα. Μόλις βρεθεί κάποιο στοιχείο a_j που να είναι μικρότερο από το a_k τότε το στοιχείο a_k τοποθετείται στα δεξιά του a_j.

Παραδείγματα

- Τετραγωνικός χρόνος $O(n^2)$

Ταξινόμηση με εισαγωγή (Insert sort) (συν.)

Ψευδοκώδικας

Είσοδος: Το σύνολο $A = \{a_1, a_2, \dots, a_n\}$ των **αταξινόμητων** στοιχείων
Έξοδος: Το σύνολο $A = \{a_1, a_2, \dots, a_n\}$ με τα στοιχεία **ταξινομημένα**

for $k = 2 \dots n$
  $x = a_k$
  $j = k - 1$
  while $j > 0$ **and** $x < a_j$
    $a_{j+1} = a_j$
    $j = j - 1$
  end while
  $a_{j+1} = x$
next k
return A

Παραδείγματα

- Τετραγωνικός χρόνος $O(n^2)$

Ταξινόμηση με εισαγωγή (Insert sort) (συν.)

Ποιο είναι το **μέγιστο πλήθος των συγκρίσεων** που πραγματοποιούνται;

Κάθε στοιχείο a_k συγκρίνεται με κάποια από τα $k-1$ προηγούμενα στοιχεία και στην χειρότερη περίπτωση με όλα. Επιπλέον συγκρίνεται ο δείκτης j κατά πόσον είναι μεγαλύτερος από μηδέν.

Συνεπώς, για κάθε k υπάρχουν κατά μέγιστο $2k$ συγκρίσεις.

Με δεδομένο ότι το k λαμβάνει τιμές από 2 έως n συνεπάγεται ότι το μέγιστο πλήθος των συγκρίσεων συμβαίνει όταν τα στοιχεία στον αρχικό πίνακα A είναι ταξινομημένα σε **ανάστροφη σειρά** και είναι:

$$2 \cdot 2 + 2 \cdot 3 + \cdots + 2 \cdot n = 2(2 + 3 + \cdots + n)$$

$$= 2\big((1 + 2 + 3 + \cdots n) - 1\big) = 2\left(\frac{n(n+1)}{2} - 1\right) = n(n+1) - 2$$

$$= n^2 + n - 2$$

Ποια είναι η πολυπλοκότητα της **χειρότερης** περίπτωσης: $O(n^2)$.

Παραδείγματα

- Κυβικός χρόνος $O(n^3)$

Πολλαπλασιασμός πινάκων

Υπολογισμός του γινομένου C δύο πίνακων A και B διαστάσεων $n \times n$

$$\begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{pmatrix}$$

Παραδείγματα

- Κυβικός χρόνος $O(n^3)$

Πολλαπλασιασμός πινάκων (συν.)

Υπολογισμός του γινομένου C δύο πίνακων A και B διαστάσεων $n \times n$

Ψευδοκώδικας

Είσοδος: Οι πίνακες A και B διαστάσεων $n \times n$
Έξοδος: Ο πίνακας $C = A \times B$ διαστάσεων $n \times n$

for $i = 1 \ldots n$
  for $j = 1 \ldots n$
    $Sum = 0$
    for $k = 1 \ldots n$
      $Sum = Sum + a_{ik} b_{kj}$
    end for
    $c_{ij} = Sum$
  end for
end for
return C

Εμπειρικός προσδιορισμός

- Σύγκριση αλγορίθμου με συνήθεις συναρτήσεις

Προϋπόθεση: να υπάρχουν χρόνοι εκτέλεσης του αλγόριθμου $T(n)$ για κάποιες ενδεικτικές τιμές του n.

Υπολογίζονται οι λόγοι $\boldsymbol{T(n)/f(n)}$ για διάφορες τυπικές συναρτήσεις $f(n)$ όπως, $\boldsymbol{\log n}$, $\boldsymbol{n\log n}$, $\boldsymbol{n}$, $\boldsymbol{n^2}$ κλπ

Εάν ο λόγος $T(n)/f(n)$ για τις διάφορες τιμές του n **συγκλίνει προς μια θετική σταθερά** τότε η $f(n)$ είναι μια **καλή εκτίμηση**.

Εάν ο λόγος $T(n)/f(n)$ για τις διάφορες τιμές του n **συγκλίνει προς το μηδέν** τότε η $f(n)$ είναι μια **υπερεκτίμηση** της πραγματικής τάξης του αλγόριθμου.

Εάν ο λόγος $T(n)/f(n)$ για τις διάφορες τιμές του n **αποκλίνει** τότε η $f(n)$ είναι μια **υποτίμηση** της πραγματικής τάξης του αλγόριθμου.

Εμπειρικός προσδιορισμός

- Παράδειγμα

n	CPU time (T)	T/n^2	T/n^3	$T/n^2 \log n$
100	022	.002200	.000022000	.0004777
200	056	.001400	.000007000	.0002642
300	118	.001311	.000004370	.0002299
400	207	.001294	.000003234	.0002159
500	318	.001272	.000002544	.0002047
600	466	.001294	.000002157	.0002024
700	644	.001314	.000001877	.0002006
800	846	.001322	.000001652	.0001977
900	1,086	.001341	.000001490	.0001971
1,000	1,362	.001362	.000001362	.0001972
1,500	3,240	.001440	.000000960	.0001969
2,000	5,949	.001482	.000000740	.0001947
4,000	25,720	.001608	.000000402	.0001938