Αλγόριθμοι και Πολυπλοκότητα

Ν. Μ. Μισυρλής

Τμήμα Πληροφορικής και Τηλεπικοινωνιών,
Πανεπιστήμιο Αθηνών

Περιεχόμενα

Περιεχόμενα

Περιεχόμενα

Εισαγωγή

Η έννοια του αλγορίθμου είναι θεμελιώδης στον χώρο της επιστήμης των υπολογιστών.
Wirth: Δομές Δεδομένων + Αλγόριθμοι = Προγράμματα

Προβλήματα

- Η αποκωδικοποίηση του ανθρωπίνου DNA
- Στο διαδίκτυο, του οποίου το ολοένα και αυξανόμενο μέγεθος δημιουργεί προβλήματα στον εντοπισμό της χρήσιμης πληροφορίας.
- Στο ηλεκτρονικό εμπόριο και γενικότερα στην επικοινωνία μέσω διαδικτύου.
- Στον εμπορικό κόσμο, η επιλογή στρατηγικών αποτελεί ένα περίπλοκο πρόβλημα.
  a. δρομολόγησης σε μια αεροπορική εταιρεία,
  β. του σχεδιασμού οδικών αξόνων,

Εισαγωγή

- Τι είναι όμως ένας αλγόριθμος;
- Πως ξεχωρίζουμε έναν καλό από έναν κακό αλγόριθμο;
- Πότε μπορούμε να κρίνουμε ότι ένας αλγόριθμος είναι ικανοποιητικός;
- Πως κατασκευάζονται αποδοτικοί αλγόριθμοι;
- Πως μπορούμε να εφαρμόζουμε έξυπνες αλγοριθμικές ιδέες στα προβλήματα που αντιμετωπίζουμε;

Ανάλυση Αλγορίθμων

Ένας αλγόριθμος είναι μια υπολογιστική διαδικασία που παίρνει μία ή περισσότερες τιμές σαν είσοδο και παράγει μία ή περισσότερες τιμές σαν έξοδο. Έτσι ένας αλγόριθμος είναι μία ακολουθία υπολογιστικών βημάτων που μετασχηματίζει την είσοδο σε έξοδο.

1. Να είναι πεπερασμένος.
2. Να είναι επακριβώς ορισμένος.
3. Να είναι ορθός.

Μία διαδικασία που έχει όλα τα χαρακτηριστικά που αναπτύξαμε αλλά δεν είναι σίγουρο αν είναι πεπερασμένη, ονομάζεται υπολογιστική μέθοδος.

Ανάλυση Αλγορίθμων

- Επιπρόσθετα δεν σημαίνει ότι κάθε αλγόριθμος μπορεί να προγραμματιστεί, δηλαδή μια καλά ορισμένη αλγοριθμική διαδικασία δεν είναι απαραίτητο ότι είναι προγραμματίσιμη.
- Χάριν απλότητας θα εννοήσουμε ταυτόσημες τις έννοιες αλγόριθμος και πρόγραμμα.

Ένα πρόγραμμα θα είναι χρήσιμο, αν εκτελείται σε

- ‘λογικό’ χρόνο
- δεσμεύει ‘λογικό’ χώρο μνήμης

- Πολυπλοκότητα χρόνου
- Πολυπλοκότητα χώρου

Ανάλυση Αλγορίθμων

Η πολυπλοκότητα ενός αλγορίθμου εκφράζεται σαν συνάρτηση της διάστασης του υπό μελέτη προβλήματος. Η διάσταση του προβλήματος είναι το πλήθος των ατομικών δεδομένων για επεξεργασία.

παράδειγμα

- το πλήθος των στοιχείων μιας ακολουθίας
- το πλήθος των στοιχείων ενός πίνακα
- το πλήθος των bits ενός αριθμού

Το πρόβλημα της Αναζήτησης

Δίνεται μια ακολουθία n στοιχείων $S = (a_1, a_2, \ldots, a_n)$ και ένα στοιχείο x. Είναι το x μέλος του S και αν ναι σε ποια θέση;

ΣΕΙΡΙΑΚΗ ΑΝΑΖΗΤΗΣΗ (S, x)

1. $i = 1$
2. while $i \neq n + 1$ and $a_i \neq x$ do
3. $\quad i = i + 1$
4. end while
5. if $i > n$
6. $\quad$ return -1
7. else
8. $\quad$ return i
9. end if

Σχήμα: Αλγόριθμος αναζήτησης ενός στοιχείου x σε μονοδιάστατο πίνακα S.

Το πρόβλημα της Αναζήτησης

- Αν το x είναι το πρώτο στοιχείο της ακολουθίας, τότε θα κάνει 3 συγκρίσεις και μία καταχώρηση
- Αν το x είναι το δεύτερο στοιχείο της ακολουθίας, τότε θα κάνει 5 συγκρίσεις και 3 καταχωρήσεις
- Αν είναι το i-οστό στοιχείο θα κάνει $2i + 1$ συγκρίσεις και $i + 1$ καταχωρήσεις.

Το πρόβλημα της Αναζήτησης

Αν D_n το σύνολο των διαφορετικών εισόδων στο πρόβλημα και $d \in D_n$ μια είσοδό του

- Πολυπλοκότητα στην βέλτιστη περίπτωση:
  $C_{\beta\pi}(n) = \min\{\text{κόστος}(d), d \in D_n\}$
- Πολυπλοκότητα στην χείριστη περίπτωση:
  $C_{\chi\pi}(n) = \max\{\text{κόστος}(d), d \in D_n\}$
- Πολυπλοκότητα κατά μέσο όρο:
  $C_{\mu o}(n) = \sum_{d \in D_n} p(d)\ \text{κόστος}(d)$

Για κάθε δυνατή είσοδο $d \in D_n$ που δίνεται με πιθανότητα $p(d)$

Το πρόβλημα της Αναζήτησης

Παράδειγμα

Δίνεται η ταξινομημένη ακολουθία $1, \ldots, n$ και σαν στοιχείο προς εξερεύνηση μας δίνεται ισοπίθανα ένα από τα $\{1, 2, \ldots, 2n\}$.

$$\begin{aligned} C_{\mu o}(n) &= \frac{1}{2n}3 + \frac{1}{2n}5 + \ldots + \frac{1}{2n}(2n+1) + \frac{n}{2n}(2n+3) \\ &= \frac{1}{2n}\sum_{i=1}^{n}(2i+1) + \frac{2n+3}{2} = \frac{3n+5}{2} \end{aligned}$$

Τότε όσον αφορά τις συγκρίσεις είναι στοιχειώδης πράξη,

- ο αλγόριθμος έχει 3 στην βέλτιστη
- $2n+3$ στην χείριστη
- και περίπου $1.5n$ κατά μέσο όρο.

Εσωτερικό γινόμενο δύο διανυσμάτων

Έστω $a = (a_i), b = (b_i)$ διανύσματα του $\mathbb{R}^n$. Ζητείται το εσωτερικό γινόμενο αυτών.

ΕΣΩΤΕΡΙΚΟ ΓΙΝΟΜΕΝΟ (a, b)

1. $sp = 0$
2. for $i = 1$ to n do
3. $\quad sp = sp + a_i b_i$
4. end for
5. return sp

Σχήμα: Υπολογισμός του εσωτερικού γινομένου δύο ίσης διάστασης διανυσμάτων a και b.

Εσωτερικό γινόμενο δύο διανυσμάτων

Ανεξάρτητα από τους αριθμούς των ακολουθιών διανυσμάτων

$$C_{\beta\pi}(n) = C_{\chi\pi}(n) = C_{\mu o}(n) = n$$

- Δεν απαιτείται η καταμέτρηση όλων των διαφορετικών τύπων πράξεων
- Αρκεί η υπολογιστικά βαρύτερη πράξη

Ασυμπτωτική Πολυπλοκότητα

Έστω ένα πρόβλημα με διάσταση n (π.χ. η αναζήτηση σε μια ακολουθία) και έστω και δύο αλγόριθμοι που το λύνουν, ο A με πολυπλοκότητα $100n$ και ο B με πολυπλοκότητα n^2. Θεωρώντας σαν καλύτερο αυτόν που κάνει λιγότερες πράξεις για την ίδια είσοδο μπορούμε να διακρίνουμε τρεις περιπτώσεις ανάλογα με το n

$$n : \begin{cases} = 100 & \text{Οι A,B έχουν ίδια πολυπλοκότητα} \\ < 100 & \text{Ο B είναι αποδοτικότερος από τον A} \\ > 100 & \text{Ο A είναι αποδοτικότερός από τον B} \end{cases}$$

Ασυμπτωτική Πολυπλοκότητα

Παρατηρήσεις

- Παρατηρούμε ότι μετά από ένα κατώφλι, όσο το n γίνεται μεγαλύτερο τόσο ο αλγόριθμος A γίνεται καλύτερος του B.
- ο ρυθμός με τον οποίο γίνεται ο A καλύτερος του B αυξάνει γραμμικά σε σχέση με το n
- Για $n = 1000$ ο αλγόριθμος A εκτελείται σε $100n \times 10^{-6} = 0.1$ δευτερόλεπτα και ο B σε $n^2 \times 10^{-6} = 1$ δευτερόλεπτο, δηλαδή είναι 10 φορές πιο αργός. Για $n = 10000$ ο B είναι 100 φορές πιο αργός, κ.ο.κ.

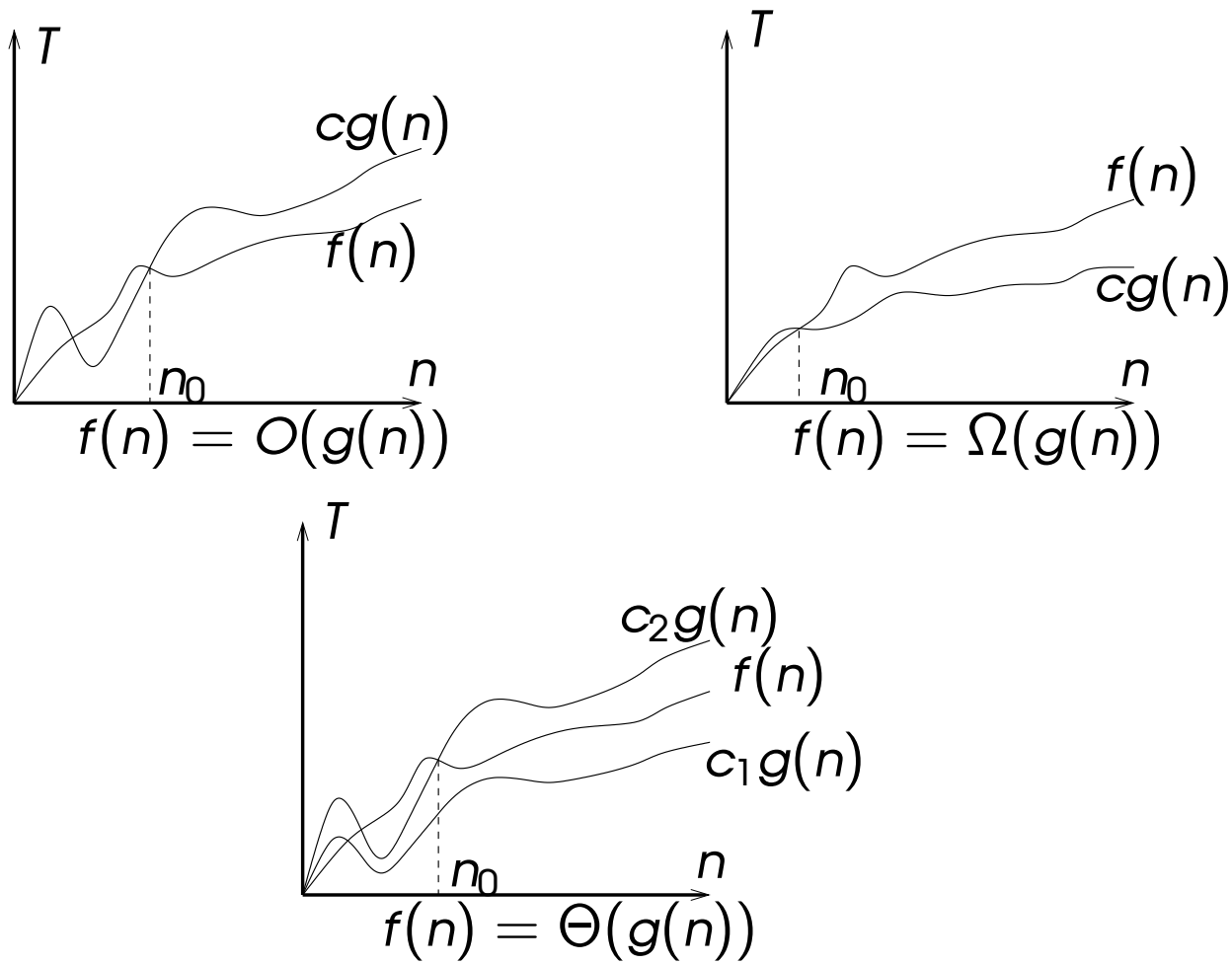
Ασυμπτωτική Πολυπλοκότητα

Συμπέρασμα

Άρα όσο αυξάνει το n τόσο η διαφορά της αποδοτικότητας γίνεται μεγάλη. Μας ενδιαφέρει τι γίνεται για μεγάλες διαστάσεις δεδομένων μετά από ένα κατώφλι η τάξη της συνάρτησης πολυπλοκότητας είναι αυτή που καθορίζει την αποδοτικότητα του αλγορίθμου

- $f(n) = O(g(n))$ αν υπάρχουν $c > 0, n_0 \geq 0$ έτσι ώστε $f(n) \leq cg(n)$ για κάθε $n \geq n_0$
- $f(n) = \Omega(g(n))$ αν υπάρχουν $c > 0, n_0 \geq 0$ έτσι ώστε $f(n) \geq cg(n)$ για κάθε $n \geq n_0$
- $f(n) = \Theta(g(n))$ αν υπάρχουν $c_1, c_2 > 0, n_0 \geq 0$ έτσι ώστε $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ για κάθε $n \geq n_0$.
- Εναλλακτικά, αν $f(n) = O(g(n))$ και $f(n) = \Omega(g(n))$, τότε $f(n) = \Theta(g(n))$

Ασυμπτωτική Πολυπλοκότητα



Σχήμα: Γραφική Απεικόνιση των συμβολισμών O, Ω, Θ

Ασυμπτωτική Πολυπλοκότητα

Τα $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$ είναι σύνολα συναρτήσεων.
Στην πραγματικότητα δηλαδή, έχουμε ότι $f(n) \in O(g(n))$, $f(n) \in \Omega(g(n))$ ή $f(n) \in \Theta(g(n))$ αλλά έχει επικρατήσει ο συμβολισμός της ισότητας. Η ισότητα είναι καθαρά συμβολισμός.
Ασυμπτωτική συμπεριφορά της συνάρτησης πολυπλοκότητας $f(n) = 3n^2 - 100n + 6$:

- $3n^2 - 100n + 6 = O(n^2)$ αφού $3n^2 - 100n + 6 \leq 3n^2$ για $n > 0$.
- $3n^2 - 100n + 6 = O(n^3)$ αφού $3n^2 - 100n + 6 \leq n^3$ για $n > 0$.
- $3n^2 - 100n + 6 \neq O(n)$ αφού $\nexists c : 3n^2 - 100n + 6 \leq cn$ για $n \geq n_0$.
- $3n^2 - 100n + 6 = \Omega(n^2)$ αφού $3n^2 - 100n + 6 \geq 2.99n^2$ για $n \geq 10^4$.
- $3n^2 - 100n + 6 = \Omega(n)$ αφού $3n^2 - 100n + 6 \geq (10^{10^{10}})n$ για $n \geq 10^{10^{10}}$.
- $3n^2 - 100n + 6 \neq \Omega(n^3)$ αφού $\nexists c : 3n^2 - 100n + 6 \geq cn^3$ για $n \geq n_0$.
- $3n^2 - 100n + 6 = \Theta(n^2)$ αφού $3n^2 - 100n + 6 = \Omega(n^2)$ και $3n^2 - 100n + 6 = O(n^2)$

Ασυμπτωτική Πολυπλοκότητα

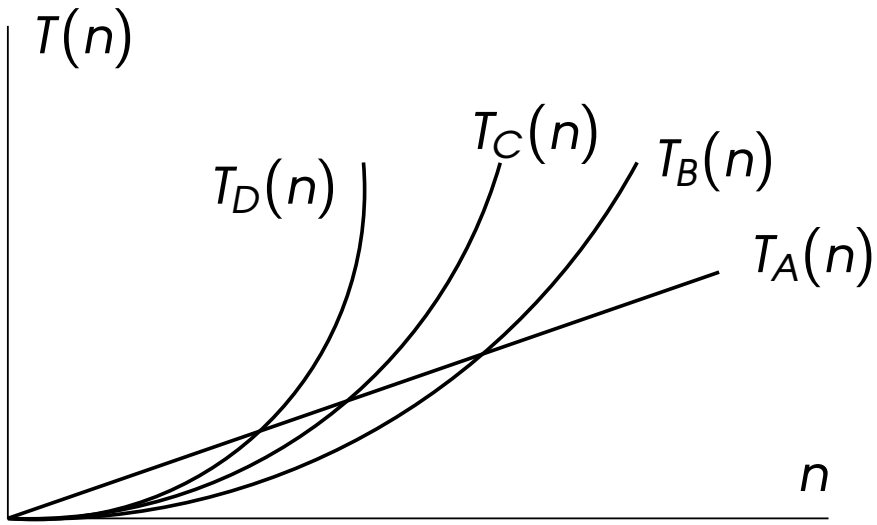
Ασυμπτωτικά σύμβολα που ορίζουν ότι τα φράγματα είναι αυστηρά:

- $f(n) = o(g(n))$ αν για κάθε $c > 0$, υπάρχει $n_0 \geq 0$ έτσι ώστε $f(n) < cg(n)$ για κάθε $n \geq n_0$
- $f(n) = \omega(g(n))$ αν για κάθε $c > 0$, υπάρχει $n_0 \geq 0$ έτσι ώστε $f(n) > cg(n)$ για κάθε $n \geq n_0$

Για παράδειγμα $n^2 + n - 1 = O(n^2) = o(n^3) \neq o(n^2)$.
Όταν $f(n) = o(g(n))$ έχουμε ότι η $f(n)$ είναι αμελητέα σε σχέση με την $g(n)$.

Κλάσεις Πολυπλοκότητας

Έστω ένα πρόβλημα Π και τέσσερις αλγόριθμοι A, B, C, D που το λύνουν με αντίστοιχες συναρτήσεις πολυπλοκότητας $T_A(n) = O(n)$, $T_B(n) = O(n^2)$, $T_C(n) = O(n^3)$ και $T_D(n) = 2^n$.



Σχήμα: Αύξηση πολυπλοκότητας με την αύξηση του μεγέθους του προβλήματος για 4 διαφορετικούς αλγορίθμους

Κλάσεις Πολυπλοκότητας

$T_A(n) < T_B(n) < T_C(n) < T_D(n)$

Αλγόριθμος	0.1 δευτ/τα	10 λεπτά	1 ημέρα
$T_A(n) = O(n)$	10^6	6×10^9	864×10^9
$T_B(n) = O(n^2)$	10^3	7.7×10^4	9.3×10^4
$T_C(n) = O(n^3)$	10^2	1800	9254
$T_D(n) = O(2^n)$	20	32	39

Πίνακας: Διάσταση προβλημάτων που εκτελούνται από τους 4 αλγόριθμους σε σταθερό χρόνο από μια μηχανή 10 MIPS

Κλάσεις Πολυπλοκότητας

Παράδειγμα: *NP*−δύσκολο πρόβλημα

Μία επιχείρηση έχει επιλέξει n τοποθεσίες για να χτίσει n εργοστάσια. Σε ποια τοποθεσία πρέπει να χτιστεί κάθε εργοστάσιο, ώστε να ελαχιστοποιείται το κόστος της μεταφοράς των υλικών;

- διάσταση 80
- μία μηχανή που εξετάζει 10^7 συνδυασμούς το δευτερόλεπτο

$\rightarrow$ περίπου 10^{112} χρόνια

Κλάσεις Πολυπλοκότητας

Παράδειγμα

Δίνεται ένα σύνολο υπαλλήλων και ένα σύνολο εργασιών οι οποίες πρέπει να ανατεθούν στους υπαλλήλους. Κάθε εργασία i στοιχίζει c_{ij} όταν ανατίθεται στον υπάλληλο j. Να βρεθεί η ανάθεση που ελαχιστοποιεί το συνολικό κόστος ανάθεσης.

- με 20 υπαλλήλους και 20 εργασίες
- θα χρειαστεί χιλιετίες για να ολοκληρωθεί
- Ο Hungarian αλγόριθμος θα χρειαστεί περίπου μόνο 1.95 δευτ/τα
- σε μία μηχανή που εξετάζει 10^6 συνδυασμούς

Συμπέρασμα

Η ανάπτυξη αποδοτικών αλγορίθμων είναι μία αναγκαιότητα παρά την ραγδαία ανάπτυξη της ταχύτητας των υπολογιστών

Κλάσεις Πολυπλοκότητας

Πολυπλοκότητα	Σημερινή Διάσταση	Μελλοντική Διάσταση
$O(n)$	n	$1000 \times n$
$O(n^2)$	n	$32 \times n$
$O(n^3)$	n	$10 \times n$
$O(n^4)$	n	$6 \times n$
$O(2^n)$	n	$n + 10$
$O(3^n)$	n	$n + 8$
$O(n!)$	n	$n + 7$

Πίνακας: Αύξηση της διάστασης του προβλήματος που επιλύεται σε σταθερό χρόνο από έναν αλγόριθμο δεδομένης πολυπλοκότητας σε μια μηχανή 1000 φορές ταχύτερη.

Συμπέρασμα

Άρα ο ισχυρισμός ότι η ανάγκη ανάπτυξης αποδοτικών αλγορίθμων θα εκλείψει μπροστά στις αυξητικές αποδόσεις των αυριανών υπολογιστών, αποτελεί μία πλήρως εσφαλμένη θέση.

Κατάταξη κλάσεων πολυπλοκότητας

Κατηγοριοποίηση των συναρτήσεων πολυπλοκότητας σε κλάσεις

1	Σταθερή πολυπλοκότητα
$\log(n)$	Λογαριθμική πολυπλοκότητα
$\log^k(n)$	Πολυλογαριθμική πολυπλοκότητα όταν k μία σταθερά
n	Γραμμική πολυπλοκότητα (πχ, $f(n) = an + b$, με a, b σταθερές)
$nlogn$	
n^2	Τετραγωνική πολυπλοκότητα
n^3	Κυβική πολυπλοκότητα
n^k	Πολυωνυμικη πολυπλοκότητα, με k μία σταθερά, δηλ. $f(x) = a_k n^k + \ldots + a_0$
a^n	Εκθετική πολυπλοκότητα $1 < a < 2$
2^n	Εκθετική πολυπλοκότητα
a^n	Εκθετική πολυπλοκότητα $a > 2$
$n!$	Παραγοντική πολυπλοκότητα
n^n	Υπερεκθετική πολυπλοκότητα

Πίνακας: Χαρακτηριστικές κλάσεις πολυπλοκότητας

Η έννοια του βέλτιστου αλγορίθμου

Αν για ένα πρόβλημα, οποιοσδήποτε αλγόριθμος που το επιλύει έχει ασυμπτωτική πολυπλοκότητα $\Omega(f(n))$ και έχουμε ήδη έναν αλγόριθμο με ασυμπτωτική πολυπλοκότητα $O(f(n))$ τότε ο αλγόριθμος αυτός είναι βέλτιστος (ασυμπτωτικά).

Παράδειγμα

Ένα παράδειγμα προβλήματος στο οποίο έχει βρεθεί ο βέλτιστος, ως προς την πολυπλοκότητα, αλγόριθμος είναι το πρόβλημα της ταξινόμησης με συγκρίσεις. Αποδεικνύεται εύκολα, ότι ένας αλγόριθμος που βασίζεται σε συγκρίσεις πρέπει να κάνει το λιγότερο $n \log n$ συγκρίσεις.

- ταξινόμησης με σωρό (*heapsort*)
- γρήγορης ταξινόμησης (*quicksort*)
- $O(n \log n)$ κατά μέσο όρο
- βέλτιστοι ως προς την πολυπλοκότητα κατά μέσο όρο

Η έννοια του βέλτιστου αλγορίθμου

Αλγόριθμος ταξινόμησης με σωρό $O(n \log n)$ στην χείριστη περίπτωση

$\rightarrow$ βέλτιστος ασυμπτωτικά και στην χείριστη περίπτωση

Αλγόριθμος γρήγορης ταξινόμησης $O(n^2)$ στην χείριστη περίπτωση

$\rightarrow$ δεν είναι βέλτιστος αλγόριθμος

Η έννοια του βέλτιστου αλγορίθμου

Παράδειγμα 2: Μη βέλτιστου αλγορίθμου

Δίνονται δύο πίνακες $A_{n\times n}$ και $B_{n\times n}$ και μας ζητείται να υπολογίσουμε το γινόμενο τους $C_{n\times n}$:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, 1 \leq i, j \leq n$$

Η έννοια του βέλτιστου αλγορίθμου

ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ ΠΙΝΑΚΩΝ (A, B)

```
1.  for i = 1 to n do
2.      for j = 1 to n do
3.          c_ij = 0
4.          for k = 1 to n do
5.              c_ij = c_ij + a_ik * b_kj
6.          end for
7.      end for
8.  end for
9.  return C
```

Σχήμα: Πολλαπλασιασμός δύο τετραγωνικών πινάκων διάστασης n.

Η έννοια του βέλτιστου αλγορίθμου

- έχει πολυπλοκότητα $O(n^3)$
- το 1969 ο Strassen $O(n^{2.81})$
- Coopersmith και Winograd του 1986 με $O(n^{2.379})$.
- οι 3 παραπάνω αλγορίθμοι δεν είναι βέλτιστοι.
- Ένας βέλτιστος αλγόριθμος που λύνει το πρόβλημα αυτό πρέπει να έχει πολυπλοκότητα $\Omega(n^2)$.

Ιδιότητες ασυμπτωτικών συμβολισμών

Μεταβατικότητα

- Αν $f(n) = \Theta(g(n))$ και $g(n) = \Theta(h(n))$ τότε $f(n) = \Theta(h(n))$,
- Αν $f(n) = O(g(n))$ και $g(n) = O(h(n))$ τότε $f(n) = O(h(n))$,
- Αν $f(n) = \Omega(g(n))$ και $g(n) = \Omega(h(n))$ τότε $f(n) = \Omega(h(n))$,
- Αν $f(n) = o(g(n))$ και $g(n) = o(h(n))$ τότε $f(n) = o(h(n))$,
- Αν $f(n) = \omega(g(n))$ και $g(n) = \omega(h(n))$ τότε $f(n) = \omega(h(n))$

Ανακλαστικότητα

- $f(n) = \Theta(f(n))$,
- $f(n) = O(f(n))$,
- $f(n) = \Omega(f(n))$

Ιδιότητες ασυμπτωτικών συμβολισμών

Συμμετρία

- $f(n) = \Theta(g(n))$ αν και μόνο αν $g(n) = \Theta(f(n))$

Ανάστροφη Συμμετρία

- $f(n) = O(g(n))$ αν και μόνο αν $g(n) = \Omega(f(n))$
- $f(n) = o(g(n))$ αν και μόνο αν $g(n) = \omega(f(n))$

Πράξεις

- $cO(f(n)) = O(f(n))$ με $c > 0$
- $O(g(n)) + O(g(n)) = O(g(n))$
- $O(g_1(n)) + O(g_2(n)) = O(\max\{g_1(n), g_2(n)\})$
- $O(g_1(n)) * O(g_2(n)) = O(g_1(n) * g_2(n))$

Ιδιότητες ασυμπτωτικών συμβολισμών

Ορισμοί

1. Αν
$$\lim_{n \to +\infty} \frac{f(n)}{g(n)} = a \neq 0$$
τότε $f(n) = \Theta(g(n))$
2. Αν
$$\lim_{n \to +\infty} \frac{f(n)}{g(n)} = 0$$
τότε $f(n) = o(g(n))$
3. Αν
$$\lim_{n \to +\infty} \frac{f(n)}{g(n)} = \infty$$
τότε $f(n) = \omega(g(n))$

Ιδιότητες ασυμπτωτικών συμβολισμών

Παράδειγμα

Να δειχθεί ότι η συνάρτηση πολυπλοκότητας

$$f(n) = \frac{(n^2 + logn)(n-1)}{n+n^2}$$

είναι $\Theta(n)$.

Έχουμε:

$$\lim_{n\to+\infty} \frac{\frac{(n^2+logn)(n-1)}{n+n^2}}{n} = \lim_{n\to+\infty} \frac{n^3 - n^2 + n\log n - \log n}{n^3 + n^2} = 1$$

Άρα $f(n) = \Theta(n)$. $\square$

Ασκήσεις

΄Ασκηση 1.

Έστω $(T_1(n) = O(f(n)))$ και $(T_2(n) = O(g(n)))$ οι πολυπλοκότητες δύο τμημάτων (P_1) και (P_2) ενός προγράμματος (Π). Αν το (P_2) εκτελείται αμέσως μετά το (P_1) ποια είναι η πολυπλοκότητα του προγράμματος (P);
Λύση:

$$T(n) = T_1(n) + T_2(n) = O(f(n)) + O(g(n)) = O(\max\{f(n), g(n)\})$$

Εφαρμογή: Αν $T_1(n) = O(n^2),\ T_2(n) = O(n^3)$ τότε

$$T(n) = T_1(n) + T_2(n) = O(\max\{n^2,\ n^3\}) = O(n^3)$$

Ασκήσεις

΄Ασκηση 2.

Να δοθεί ο καλύτερος O συμβολισμός για τις ακόλουθες συναρτήσεις:

1. $2\log n - 4n + 3n\log n$
2. $2 + 4 + \ldots + 2n$
3. $2 + 4 + 8 + \ldots + 2^n$

1. $2\log n - 4n + 3n\log n = O(n\log n)$
2. $2 + 4 + \ldots + 2n = 2(1 + 2 + \ldots + n) = 2\sum_{i=1}^{n} i = n(n+1) = O(n^2)$
3. $(2 + 4 + 8 + \ldots + 2^n = 2\frac{2^n - 1}{2-1} = 2 * 2^n - 2 = O(2^n))$

Ασκήσεις

΄Ασκηση 3.

Έστω $a > 1$ και $f(n) = O(\log_a(n))$. Δείξτε ότι $f(n) = O(\log n)$.
Αφού $f(n) = O(\log_a(n))$ με βάση τον ορισμό, υπάρχει c που για κάθε $n \geq n_0$: $f(n) \leq c \log_a(n)$. Όμως επειδη $\log_a(n) = \frac{\log n}{\log a}$ έχουμε $f(n) \leq \frac{c}{\log a} \log n$ ή $f(n) \leq c' \log n$ με $c' = \frac{c}{\log a}$ για κάθε $n \geq n_0$. ΄Αρα $f(n) = O(\log n)$.

΄Ασκηση 4.

Δείξτε ότι $\log(n!) = O(n \log n)$.
Επειδή $n! = n(n-1)\ldots 2 < n \ldots n = n^n$ και επειδή η λογαριθμική συνάρτηση είναι γνησίως αύξουσα, έχουμε $\log(n!) < n \log n$, άρα $\log(n!) = O(n \log n)$.

Ασκήσεις

΄Ασκηση 5.

Ποια είναι η πολυπλοκότητα του παρακάτω τμήματος προγράμματος:

---

1. for $i = 1$ to n do
2. for $j = 1$ to $\frac{i+1}{2}$ do
3. $x = x + 1$
4. end for
5. end for

$$T(n) = \sum_{i=1}^{n} \frac{i+1}{2} = \frac{1}{2}\left(\sum_{i=1}^{n} i + \sum_{i=1}^{n} 1\right) = \frac{1}{2}\left(\frac{n(n+1)}{2} + n\right) = O(n^2)$$

΄Ασκηση 6.

Να δειχθεί ότι η $3^n \neq O(2^n)$

΄Εχουμε $\lim_{n\to+\infty} \frac{2^n}{3^n} = 0$ ΄Αρα $3^n = \omega(2^n)$

Ασκήσεις

΄Ασκηση 7.

Να υπολογιστεί η πολυπλοκότητα του αλγόριθμου του Ευκλείδη (σχήμα 6) για τον υπολογισμό του Μέγιστου Κοινού Διαιρέτη (Μ.Κ.Δ.) δύο θετικών ακέραιων αριθμών.

ΜΚΔ - ΕΥΚΛΕΙΔΗΣ $(m, n \in \mathbb{Z})$

```
1.  repeat
2.      r = m mod n
3.      if r ≠ 0 then
4.          m = n
5.          n = r
6.      end if
7.  until r = 0
8.  return n
```

Σχήμα: Αλγόριθμος του Ευκλείδη για την εύρεση του Μ.Κ.Δ. δύο ακεραίων m και n.

Ασκήσεις

΄Ασκηση 7.

Ας θεωρήσουμε ότι $m \geq n$, $m = qn + r$, $q \geq 1$, $r \geq 0$. ΜΚΔ(m, n)=ΜΚΔ(n, r),

$$
\begin{aligned}
m &= q_0 n + r_0 \quad , \quad 0 \leq r_0 < n, \quad q_0 = q, \quad r_0 = r \\
n &= q_1 r_0 + r_1 \quad , \quad 0 \leq r_1 < r_0 \\
r_0 &= q_2 r_1 + r_2 \quad , \quad 0 \leq r_2 < r_1 \\
r_1 &= q_3 r_2 + r_3 \quad , \quad 0 \leq r_3 < r_2 \\
&\ldots \qquad\quad \ldots
\end{aligned}
$$

Έστω k ο αριθμός των βημάτων και μία ακολουθία από αριθμούς $m, n, r_0, r_1, \ldots r_k$ με $r_k = 0$ και r_{k-1} =ΜΚΔ(m, n). Η πολυπλοκότητα του αλγορίθμου είναι $O(k)$. Αποδεικνύται ότι $r_i < \frac{1}{2} r_{i-2}$, οπότε έχουμε ότι $r_k < \frac{1}{2} < \frac{1}{2} r_{k-2} < \frac{1}{4} r_{k-4} < \ldots < \frac{1}{2^k} r_0 < \frac{1}{2^k} n$ ΄Αρα $n > 2^{k-1}$, δηλαδή $k \leq \log n$, οπότε η πολυπλοκότητα του αλγορίθμου είναι $O(logn)$. Παρατηρούμε ότι η πολυπλοκότητα του αλγορίθμου δεν εξαρτάται από τον μεγαλύτερο αριθμό m.

Ασκήσεις

΄Ασκηση 8.

Δείξτε ότι για πραγματικούς αριθμούς a, b, $b > 0$ έχουμε $(n+a)^b = \Theta(n^b)$.
Από τον τύπο που δίνει το διώνυμο του Newton παίρνουμε:

$$(n+a)^b = n^b + \binom{b}{1} n^{b-1}a + \ldots + \binom{b}{k} n^{b-k}a^k + \ldots + a^b = \Theta(n^b)$$

΄Ασκηση 9.

- Δείξτε ότι $2^{n+1} = O(2^n)$.
  ΄Εχουμε $2^{n+1} = 2 * 2^n = O(2^n)$
- Είναι $2^{2n} = O(2^n)$
  Επειδή $2^{2n} = (2^2)^n = 4^n$, το 2^{2n} δεν είναι $O(2^n)$.
  Ισοδύναμα: $\lim_{n\to+\infty} \frac{2^{2n}}{2^n} = +\infty$ και όχι μηδέν.
- Αν $f(n) = O(n)$ τότε $2^{f(n)}$ δεν είναι $O(2^n)$.
  ΄Εστω $f(n) = 2n$. Τότε $f(n) = O(n)$, αλλά $2^{2n} \neq O(2^n)$.

Ταξινόμηση Φυσαλίδας (Bubble Sort)

Ο αλγόριθμος αυτός, σε κάθε επανάληψη ανυψώνει στο τέλος του πίνακα το ελαφρύτερο (μικρότερο) στοιχείο.

BUBBLE SORT (A)

1. for $i = n$ to 1 do
2. for $j = 1$ to $i - 1$ do
3. if $a_j < a_{j+1}$ then
4. $swap(a_j, a_{j+1})$
5. end if
6. end for
7. end for

Ταξινόμηση Φυσαλίδας (Bubble Sort)

Παρατηρούμε ότι ανεξάρτητα από την μορφή της ακολουθίας εισόδου, ο αλγόριθμος θα κάνει τις ίδιες συγκρίσεις και καταχωρήσεις.

$$C_{\beta\pi}(n) = C_{\chi\pi}(n) = C_{\mu o}(n) = \sum_{i=n}^{1}\sum_{j=1}^{i-1} O(1) = O(1)\sum_{i=1}^{n}(i-1)$$

$$= O(1)[\frac{n(n+1)}{2} - n] = O(n^2)$$

Ταξινόμηση με εισαγωγή (Insertion Sort)

Στο βήμα i, η υπακολουθία $a_1, \ldots, a_{i-1}$ είναι ταξινομημένη και η $a_{i+1}, \ldots, a_n$ αταξινόμητη. Εισάγει το στοιχείο i στην σωστή θέση της πρώτης υπακολουθίας και συνεχίζει επαναληπτικά για τα υπόλοιπα στοιχεία.

INSERTION SORT (A)

1. for $i = 2$ to n do
2. $\quad val = a_i$
3. $\quad j = i - 1$
4. $\quad$ while $j \geq 1$ and $a_j < val$ do
5. $\quad\quad a_{j+1} = a_j$
6. $\quad\quad j = j - 1$
7. $\quad$ end while
8. $\quad a_{j+1} = val$
9. end for

Ταξινόμηση με εισαγωγή (Insertion Sort)

- Στη βέλτιστη περίπτωση (ήδη ταξινομημένη φθίνουσα ακολουθία)
  $\rightarrow C_{\beta\pi}(n) = O(n)$.
- Στη χείριστη περίπτωση
  $\rightarrow C_{\chi\pi}(n) = \sum_{i=2}^{n} \sum_{j=i-1}^{1} O(1) = O(1) \sum_{i=2}^{n} (i-1) = O(n^2)$
- Αποδεικνύεται $C_{\mu o} = O(n^2)$.

Ταξινόμηση με Επιλογή (Selection Sort)

Βρίσκει στο βήμα i το στοιχείο που είναι το μεγαλύτερο στην υπακολουθία $a_i, \ldots, a_n$. Θέτει στην θέση i το μέγιστο αυτό στοιχείο και συνεχίζει επαναληπτικά.

SELECTION SORT (A)

1. for $i = 1$ to $n - 1$ do
2. $\quad max = i$
3. $\quad$ for $j = i + 1$ to n do
4. $\quad\quad$ if $a_j > a_{max}$ then
5. $\quad\quad\quad max = j$
6. $\quad\quad$ end if
7. $\quad$ end for
8. $\quad swap(a_{max}, a_i)$
9. end for

Ταξινόμηση με Επιλογή (Selection Sort)

Η πολυπλοκότητα όπως και στην Ταξινόμηση Φυσαλίδας δεν εξαρτάται από την ακολουθία που έχουμε σαν είσοδο.

$$C_{\beta\pi}(n) = C_{\chi\pi}(n) = C_{\mu o}(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} O(1) = O(1) \sum_{i=1}^{n-1} (n-i) =$$

$$= O(1)\left[n(n-1) - \frac{n(n-1)}{2}\right] = O(n^2)$$

Ο καλύτερος είναι ο αλγόριθμος της Εισαγωγής γιατί στη βέλτιστη περίπτωση είναι $O(n)$ σε αντίθεση με τους άλλους δύο που είναι πάντα $O(n^2)$. Οι τρεις αλγόριθμοι παρουσιάζουν τετραγωνική πολυπλοκότητα κατά μέσο όρο.

Ορθότητα-Αναλλοίωτη Συνθήκη (Correctness-Loop invariant)

Επαγωγή

Απλή μορφή Έστω πρόταση π η οποία εξαρτάται από ένα φυσικό $n \geq n_0$.

1. Η π αληθής για $n = n_0$
2. Αν η π αληθής για $n = k$ τότε αληθής για $n = k + 1$

Επαγωγή

Ισχυρη μορφή Έστω πρόταση π η οποία εξαρτάται από ένα φυσικό $n \geq n_0$.

1. Η π αληθής για $n = n_0$
2. Αν η π αληθής για $n \leq k$ τότε αληθής για $n = k + 1$

Ορθότητα-Αναλλοίωτη Συνθήκη (Correctness-Loop invariant)

Αποδειξη ορθότητας βρόχου B

Εστω βρόχος B με μια πρόταση (αναλλοίωτη συνθήκη) π.

1. **Αρχικός έλεγχος:** Η π ισχύει πριν την πρώτη επανάληψη του B.
2. **΄Ελεγχος διατήρησης:** Αν η π ισχύει πριν την i-οστή επανάληψη εξακολουθεί να ισχύει και μετά την i-οστή επανάληψη.
3. **Επιβεβαίωση αποτελέσματος:** Η π ισχύει και μετά το τέλος του B.

Ορθότητα Ταξινόμησης με Επιλογή (Correctnes-Selection Sort)

Παράδειγμα

Να αποδειχθεί η ορθότητα του ακόλουθου αλγόριθμου για την εύρεση του μέγιστου στοιχείου ενός πίνακα $A[1..n]$.

$$
\begin{array}{l}
\text{μέγιστο} = A[1] \\
\text{για } i \leftarrow 2 \text{ έως } n \\
\quad \text{αν} \quad \text{μέγιστο} < A[i] \\
\quad \text{τότε} \quad \text{μέγιστο} = A[i]
\end{array}
$$

Ορθότητα Ταξινόμησης με Επιλογή (Correctnes-Selection Sort)

Παράδειγμα

π: στην αρχή της i-ιοστής επανάληψης η *μέγιστο περιέχει το μέγιστο στοιχείο των πρώτων* $i-1$ *στοιχείων του πίνακα.*

1. ***Αρχικός έλεγχος:*** *Η* π *είναι αληθής πριν την πρώτη επανάληψη όπου* $i = 2$, *διότι* $\mu\acute{\epsilon}\gamma\iota\sigma\tau o = A[1]$ *(τετριμμένη).*
2. ***Διατήρηση:*** *Έστω ότι η* π *είναι αληθής μετά την* $i-1$ *επανάληψη (πριν την* i*), έστω δηλαδή ότι*

$$\mu\acute{\epsilon}\gamma\iota\sigma\tau o = max A[1..i-1].$$

Θα παραμένει αληθής και μετά το πέρας της i *επανάληψης, διότι η τιμή της μέγιστο, λόγω της αν .. τότε θα είναι*

$$\mu\acute{\epsilon}\gamma\iota\sigma\tau o = \max\{\mu\acute{\epsilon}\gamma\iota\sigma\tau o, A[i]\}.$$

3. ***Επιβεβαίωση:*** *Η* π *θα είναι αληθής και μετά το τέλος των επαναλήψεων διότι θέτοντας* $i = n+1$ *στην* π

$$\mu\acute{\epsilon}\gamma\iota\sigma\tau o = max A[1..n]$$