ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΠΛΗΡΟΦΟΡΙΚΗ Ι

Ενότητα 6: Αποτελεσματικότητα αλγορίθμων

Μιχάλης Δρακόπουλος

Σχολή Θετικών επιστημών

Τμήμα Μαθηματικών

Αποτελεσματικότητα αλγορίθμων

Αποτελεσματικότητα αλγορίθμων

- Ενας *σωστός* αλγόριθμος δεν είναι κατ' ανάγκη και *αποτελεσματικός*.
- Παρά τη συνεχή αύξηση των υπολογιστικών δυνατοτήτων, το θέμα της αποτελεσματικότητας παραμένει πολύ σημαντικό: ανάγκη για επίλυση ακόμα μεγαλύτερων προβλημάτων.
- Ενα πρόβλημα μπορεί στην πράξη να μην επιλύεται από ένα μη αποτελεσματικό αλγόριθμο.
- Η αποτελεσματικότητα ανάγεται στην ελαχιστοποίηση του αριθμού των πράξεων που απαιτούνται για τον υπολογισμό ενός αποτελέσματος.
- Οταν τεκμηριωθεί η αποτελεσματικότητα ενός αλγορίθμου για μια κατηγορία προβλημάτων, τότε ο αλγόριθμος μπορεί να εφαρμοστεί με *προβλέψιμη* συμπεριφορά για οποιοδήποτε πρόβλημα της ίδιας κατηγορίας.
- Το θέμα της αποτελεσματικότητας των αλγορίθμων είναι είναι από τα σημαντικότερα στην Πληροφορική. Εδώ δίνεται μόνο μια βασική εισαγωγή.

Μέτρηση της αποτελεσματικότητας

Η αποτελεσματικότητα εκφράζεται από τον ρυθμό αύξησης των απαιτήσεων του αλγορίθμου σε *χρόνο* (πράξεις) και *χώρο* (μνήμη) σαν συνάρτηση του μεγέθους του προβλήματος.

- Αν ο αλγόριθμος εφαρμοστεί σε πρόβλημα διπλάσιου μεγέθους, πόσες περισσότερες πράξεις θα εκτελεστούν και πόσα περισσότερα δεδομένα θα πρέπει να αποθηκευτούν?
- Απαντώντας στην ερώτηση αυτή μπορούμε να προβλέψουμε αλλά και να συγκρίνουμε την αποτελεσματικότητα αλγορίθμων.

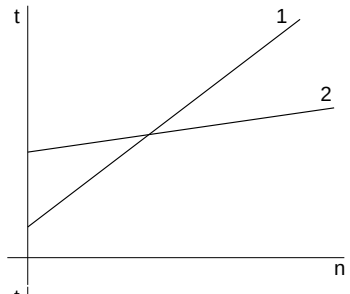
Το μέγεθος του προβλήματος n αντιπροσωπεύει το πλήθος των δεδομένων που επεξεργάζεται ένα αλγόριθμος. Π.χ. έλεγχος αν ο αριθμός n είναι πρώτος, άθροισμα n αριθμών, εύρεση πραγματικών ριζών πολυωνύμου βαθμού n.

Πολυπλοκότητα αλγορίθμου O

Γραμμικός αλγόριθμος $O(n)$

$$t = a + b \times n$$

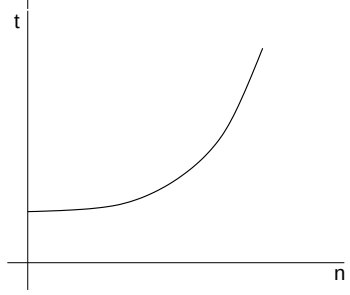
για διπλάσιο μέγεθος προβλήματος διπλασιάζεται και ο χρόνος επίλυσης.



Τετραγωνικός αλγόριθμος $O(n^2)$

$$t = a + b \times n + c \times n^2$$

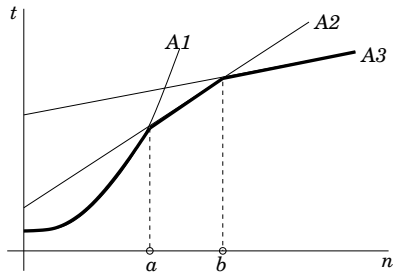
για διπλάσιο μέγεθος προβλήματος τετραπλασιάζεται και ο χρόνος επίλυσης.



Παρατηρήσεις

Ο ταχύτερος αλγόριθμος συχνά εξαρτάται από το μέγεθος του προβλήματος.

Αλλοι παράγοντες, π.χ. η είσοδος δεδομένων, μπορεί να επιβαρύνουν ένα αλγόριθμο καθώς αυξάνεται το μέγεθος του.



Η αποτελεσματικότητα ενός αλγορίθμου σε χρόνο/χώρο εξετάζεται σε 2 επίπεδα:

μαθηματικής δομής: καθορίζει την πολυπλοκότητα του αλγορίθμου.

υλοποίησης: ενας «καλός» αλγόριθμος μπορεί να υλοποιηθεί αναποτελεσματικά.

Παρατηρήσεις (συνεχ.)

Συχνά οι βελτιώσεις σε χρόνο και χώρο είναι *ασυμβίβαστες*. Π.χ. για τον έλεγχο ενός πρώτου αριθμού μπορούμε να ελέγξουμε σαν πιθανούς διαιρέτες:

1. ή *όλους* τους περιτούς αριθμούς $\leq \sqrt{n}$,
2. ή μόνο τους *πρώτους* αριθμούς $\leq \sqrt{n}$, από μια λίστα *αποθηκευμένων* πρώτων αριθμών.

Ο 2ος αλγόριθμος είναι ενδεχομένως αποτελεσματικότερος σε *χρόνο* αλλά χρειάζεται μια μεγάλη *αποθηκευμένη* λίστα πρώτων αριθμών, που μπορεί να δεσμεύει περισσότερο *χώρο* από εκείνο που διαθέτουμε.

Ανάλυση παραγόντων

Αναζήτηση των παραγόντων (όχι κατ' ανάγκη πρώτων) ακεραίου n.

```
% Αλγόριθμος A
n = input(' ? ');          % Q1
m = 2;                     % Q2
while m <= n-1,
    if rem(n, m) == 0,     % P1
        disp(m);           % P2
    end
    m = m + 1;             % P3
end
```

Για κάθε n ($n \geq 2$) εκτελούνται $n-2$ επαναλήψεις.

Αν $P = P_1 + P_2 + P_3$, $Q = Q_1 + Q_2$ είναι ο χρόνος που απαιτείται για την εκτέλεση των εντός και εκτός του **for** εντολών αντίστοιχα, τότε ο συνολικός χρόνος είναι

$$P \times (n-2) + Q = P \times n + (Q - 2 \times P)$$

και επειδή, τα P, Q ανεξάρτητα του n συμπεραίνουμε ότι πρόκειται για *γραμμικό* αλγόριθμο $O(n)$.

Ανάλυση παραγόντων (συνεχ.)

Για περαιτέρω βελτίωση, χρειάζεται μείωση των επαναλήψεων.

Για κάθε παράγοντα m του n και ο $c = n/m$ είναι επίσης παράγοντας και χρειάζονται μόνο $\approx \sqrt{n}$ επαναλήψεις.

Υπάρχει επιπλέον ένα αμελητέο κόστος Q_3, ενώ το κόστος κάθε επανάληψης αυξάνεται (στη χειρότερη περίπτωση) κατά $P_2 + P_4$ που είναι ανεξάρτητα του n.

```
% Αλγόριθμος B
n = input(' ? ');          % Q1
m = 2;                     % Q2
k = floor(sqrt(n));        % Q3
while m <= k,
    if rem(n, m) == 0,     % P1
        c = n/m;           % P4
        disp(m); disp(c);  % 2xP2
    end
    m = m + 1;             % P3
end
```

Στη χειρότερη περίπτωση:
$(P_1 + 2 \times P_2 + P_3 + P_4) \times \lfloor \sqrt{n} \rfloor + Q_1 + Q_2 + Q_3$ και επομένως ο νέος αλγόριθμος είναι $O(\sqrt{n})$ ως προς το χρόνο.

Κανονικοποιημένες μονάδες χρόνου

Το κόστος των αριθμητικών πράξεων ενός αλγορίθμου εκφράζεται παίρνοντας σαν μονάδα μέτρησης το χρόνο που χρειάζεται ο υπολογιστής για να κάνει μια πρόσθεση. Ενδεικτικές τιμές (εξαρτώνται από την αρχιτεκτονική του επεξεργαστή):

$+, -, \times$	$/$	$\sqrt{\cdot}$	floor	rem	λογικές πράξεις	input	disp
1	4	4	1	6	1	g	p

όπου το κόστος για **rem**(m,n) υπολογίζεται από $m - (m/n) \times n$.

Αναλυτική μέτρηση πράξεων

```
% Αλγόριθμος A
n = input(' ? ');          % g
m = 2;                     % 1
while m <= n-1,            % 2*(n-2)+2
    if rem(n, m) == 0,     % 7*(n-2)
        disp(m);           % p*(n-2)
    end
    m = m + 1;             % 2*(n-2)
end
```

$(11 + p) \times n - 2 \times p + g - 19$

```
% Αλγόριθμος B
n = input(' ? ');          % g
m = 2;                     % 1
k = floor(sqrt(n));        % 6
while m <= k,              % 1*(k-1)+1
    if rem(n, m) == 0,     % 7*(k-1)
        c = n/m;           % 5*(k-1)
        disp(m);           % p*(k-1)
        disp(c);           % p*(k-1)
    end
    m = m + 1;             % 2*(k-1)
end
```

$(15 + 2 \times p) \times \lfloor\sqrt{n}\rfloor - 7 - 2 \times p + g$

Παρατηρήσεις

Ο αλγόριθμος που προκύπτει από τον Β αντικαθιστώντας τις γραμμές

```
k = floor(sqrt(n));
while m <= k,
```

από την

```
while m <= floor(sqrt(n))
```

εκτελεί 6 επιπλέον μονάδες κόστους ανά επανάληψη και συνολικά εκτελεί

$$(21 + 2 \times p) \times \lfloor\sqrt{n}\rfloor - 7 - 2 \times p + g$$

«προσθέσεις» (πράξεις).

Μέτρηση πράξεων για διάφορα n

	Αλγόριθμος Α	Αλγόριθμος Β
n	$11n + p(n-2) + g - 19$	$15\lfloor\sqrt{n}\rfloor + 2p(\lfloor\sqrt{n}\rfloor - 1) + g - 7$
4	$25 + 2p + g$	$23 + 2p + g$
10	$91 + 8p + g$	$38 + 4p + g$
100	$1081 + 98p + g$	$143 + 18p + g$
500	$5481 + 498p + g$	$323 + 42p + g$
1000	$10981 + 998p + g$	$458 + 60p + g$

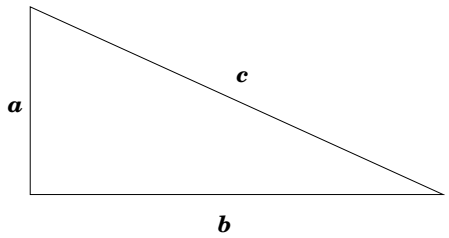
«Πυθαγόρειες τριάδες»

Να βρεθούν όλοι οι ακέραιοι a, b, c για τους οποίους ισχύει

$$a^2 + b^2 = c^2$$

και

$$a + b + c \leq p$$



ή ισοδύναμα, να βρεθούν όλα τα ορθογώνια τρίγωνα με ακέραια μήκη πλευρών, των οποίων η περίμετρος δεν υπερβαίνει μια μέγιστη τιμή p.

«Πυθαγόρειες τριάδες»- Αλγόριθμος 1

```
maxperimeter = input('Μέγιστη περίμετρος ? ');
for side1 = 1: maxperimeter,
    for side2 = 1: maxperimeter,
        for hyp = 1:maxperimeter,
            if side1^2+side2^2==hyp^2 & side1+side2+hyp<=maxperimeter,
                disp(side1 ); disp(side2 ); disp(hyp);
            end
        end
    end
end
```

Το μέγεθος του προβλήματος p είναι η τιμή της maxperimeter.

Η εντολή **if** εκτελείται maxperimeter^3 φορές και επομένως ο αλγόριθμος 1 είναι $O(p^3)$.

Παρατηρήσεις στον αλγόριθμο 1

- Εστω c ο μεγαλύτερος από τους 3 ακεραίους. Από τριγωνική ανισότητα:

$$c \leq a + b \Rightarrow 2c \leq a + b + c \Rightarrow c \leq p/2$$

- Ο αλγόριθμος 1 θα υπολογίσει π.χ. για τους αριθμούς $3:4:5$ όλους τους δυνατούς συνδυασμούς τους. Αυτό αποφεύγεται αν θεωρήσουμε ότι οι πλευρές είναι διατεταγμένες έτσι ώστε $a \leq b \leq c$.
- Η πιο εσωτερική εντολή επανάληψης μπορεί να τερματιστεί όταν το άθροισμα των πλευρών ξεπεράσει την περίμετρο.

«Πυθαγόρειες τριάδες»- Αλγόριθμος 2

```
maxperimeter = input('Μέγιστη περίμετρος ? ');
for side1 = 1: maxperimeter/2,
   for side2 = 1: maxperimeter/2,
      sumside = side1 + side2;
      hyp = side2;
      while hyp<=maxperimeter/2 & sumside+hyp<=maxperimeter,
         if side1^2+side2^2==hyp^2,
            disp(side1 ); disp(side2 ); disp(hyp);
         end
         hyp = hyp + 1;
      end
   end
end
```

Παρατηρήσεις στον αλγόριθμο 2

Μια λεπτομερής καταμέτρηση των πράξεων του αλγορίθμου 2, δίνει

$$\frac{p^3}{48} + \frac{p^2}{8} + \frac{p}{6}$$

δηλ. ο αλγόριθμος 2 είναι και πάλι $O(p^3)$.

Για περαιτέρω βελτίωση μπορούμε να εξαλείψουμε την πιο εσωτερική εντολή επανάληψεις παρατηρώντας ότι:

- Η hyp μπορεί αν υπολογιστεί από τις side1 και side2

```
hyp = fix ( sqrt (side1^2+side2^2));
```

- Αφού υπολογίσουμε την hyp θα πρέπει να ελέγξουμε αν είναι ακέραιος αριθμός

```
if hyp^2==side1^2+side2^2
```


«Πυθαγόρειες τριάδες»- Αλγόριθμος 3

```
maxperimeter = input('Μέγιστη περίμετρος ? ');
for side1 = 1: maxperimeter/2,
   side2 = side1;
   sidesq = 2*side1^2;
   while side2<=maxperimeter/2 & sidesq<=maxperimeter^2/4,
      sidesq = side1^2 + side2^2;
      hyp = fix ( sqrt (sidesq ));
      if hyp^2==sidesq,
         disp(side1 ); disp(side2 ); disp(hyp);
      end
      side2 = side2 + 1;
   end
end
```

Εκτελούνται $p^2/8 + p/4$ πράξεις, δηλ. ο αλγόριθμος 3 είναι $O(p^2)$.

Σημειώματα

Σημείωμα Αναφοράς

Copyright Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών, Μιχάλης Δρακόπουλος, 2014.
Μιχάλης Δρακόπουλος. «Πληροφορική Ι. Ενότητα 6: Αποτελεσματικότητα αλγορίθμων». Έκδοση: 1.0. Αθήνα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: http://opencourses.uoa.gr/modules/document/?course=MATH105.

Σημείωμα Αδειοδότησης

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση Παρόμοια Διανομή 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-sa/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.

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.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» 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