ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

Δομές Δεδομένων

Ενότητα 2: Θέματα Απόδοσης

Απόστολος Παπαδόπουλος
Τμήμα Πληροφορικής

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο την 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.

Θέματα Απόδοσης

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

Οργανώνοντας τα Δεδομένα

- Η επιλογή της δομής δεδομένων και του αλγορίθμου επηρεάζουν το χρόνο εκτέλεσης ενός προγράμματος, ο οποίος μπορεί να κυμαίνεται από λίγα δευτερόλεπτα έως μερικές ώρες/μέρες/...

- Μια λύση λέγεται **αποδοτική**, εάν επιλύει το πρόβλημα εντός των περιορισμών που υπάρχουν ως προς τους πόρους:
  - Χώρος
  - Χρόνος

- Το **κόστος** μιας λύσης είναι το ποσό των πόρων που αυτή η λύση δαπανά.

Επιλέγοντας μια Δομή Δεδομένων

1. Επιλέγουμε μια δομή δεδομένων ως ακολούθως:
2. Αναλύουμε το πρόβλημα για να καθορίσουμε τους περιορισμούς πόρων, στους οποίους πρέπει να υπόκειται μια λύση.
   - Καθορίζουμε τις βασικές λειτουργίες που πρέπει να υποστηρίζονται.
   - Όλα τα δεδομένα εισάγονται από την αρχή ή υπάρχουν εισαγωγές οι οποίες παρεμβάλλονται ανάμεσα σε άλλες λειτουργίες;
   - Μπορούν τα δεδομένα να διαγραφούν;
   - Όλα τα δεδομένα υφίστανται επεξεργασία σε μια καθορισμένη σειρά ή επιτρέπεται τυχαία προσπέλαση;
3. Επιλέγουμε τη δομή δεδομένων που ικανοποιεί καλύτερα αυτούς τους περιορισμούς.

Η Φιλοσοφία των Δομών Δεδομένων

- Κάθε δομή δεδομένων έχει υπέρ και κατά. Σπάνια μια δομή δεδομένων είναι καλύτερη από μια άλλη σε όλες τις περιπτώσεις.

- Μια δομή δεδομένων απαιτεί:
  - Χώρο για κάθε αντικείμενο δεδομένων που αποθηκεύει.
  - Χρόνο για να εκτελέσει κάθε βασική λειτουργία.
  - Προγραμματιστική προσπάθεια.

- Κάθε πρόβλημα έχει περιορισμούς ως προς το διαθέσιμο χώρο και χρόνο.

Αλγόριθμοι και Προγράμματα

- **Αλγόριθμος**: μια μέθοδος ή μια διαδικασία που ακολουθείται για την επίλυση ενός προβλήματος.
  - Μια συνταγή

- Ένας αλγόριθμος παίρνει την είσοδο ενός προβλήματος (συνάρτηση) και τη μετασχηματίζει στην έξοδο.
  - Μια αντιστοίχιση της εισόδου στην έξοδο..

- Ένα πρόβλημα μπορεί να επιλύεται με πολλούς εναλλακτικούς αλγορίθμους.

Ιδιότητες Αλγορίθμου

- Ένας αλγόριθμος έχει τις ακόλουθες ιδιότητες:
  - Πρέπει να είναι **ορθός.**
  - Πρέπει να αποτελείται από μια σειρά από **σαφή βήματα.**
  - Πρέπει να αποτελείται από ένα **πεπερασμένο** αριθμό βημάτων.
  - Πρέπει να είναι **σαφές** ποιο βήμα θα εκτελεστεί κάθε φορά.
  - Πρέπει να **τερματίζει.**
- Ένα πρόγραμμα υπολογιστή είναι ένα στιγμιότυπο ή μια διακριτή αναπαράσταση για έναν αλγόριθμο σε μια προγραμματιστική γλώσσα.
  - **Μαθηματικό υπόβαθρο**: Σύνολα, Αναδρομή, Επαγωγικές αποδείξεις, Λογάριθμοι, Αθροίσματα, Επαναληπτικές σχέσεις.

Απόδοση Αλγορίθμων

- Συνήθως υπάρχουν πολλοί τρόποι (αλγόριθμοι) για την επίλυση ενός προβλήματος. Πώς επιλέγουμε μεταξύ αυτών;
- Πρέπει να ικανοποιηθούν δύο (αντικρουόμενοι) στόχοι
  1. Ο αλγόριθμος να είναι εύκολα κατανοητός, ενώ παράλληλα να είναι απλός τόσο ο προγραμματισμός του όσο και η αποσφαλμάτωση (debugging).
     - Ο στόχος (1) αφορά τις Αρχές Προγραμματισμού.
  2. Ο αλγόριθμος να κάνει αποδοτική χρήση των πόρων του υπολογιστή.
     - Ο στόχος (2) αφορά τις Δομές Δεδομένων και τη Θεωρία Αλγορίθμων.
- Όταν εστιάσουμε στο 2ο στόχο πώς μετράμε το κόστος του αλγορίθμου;

Μέτρηση Απόδοσης

1. Εμπειρική σύγκριση (τρέξιμο προγραμμάτων).
2. **Ασυμπτωτική ανάλυση αλγορίθμων.**

Παράγοντες που επηρεάζουν το χρόνο εκτέλεσης ενός προγράμματος:

- Για τους περισσότερους αλγορίθμους, ο χρόνος τρεξίματος εξαρτάται από το ‘μέγεθος’ της εισόδου.
- Ο χρόνος τρεξίματος εκφράζεται ως **f(n)** για κάποια συνάρτηση **f** πάνω στο μέγεθος **n** της εισόδου.

Παραδείγματα Ρυθμού Αύξησης

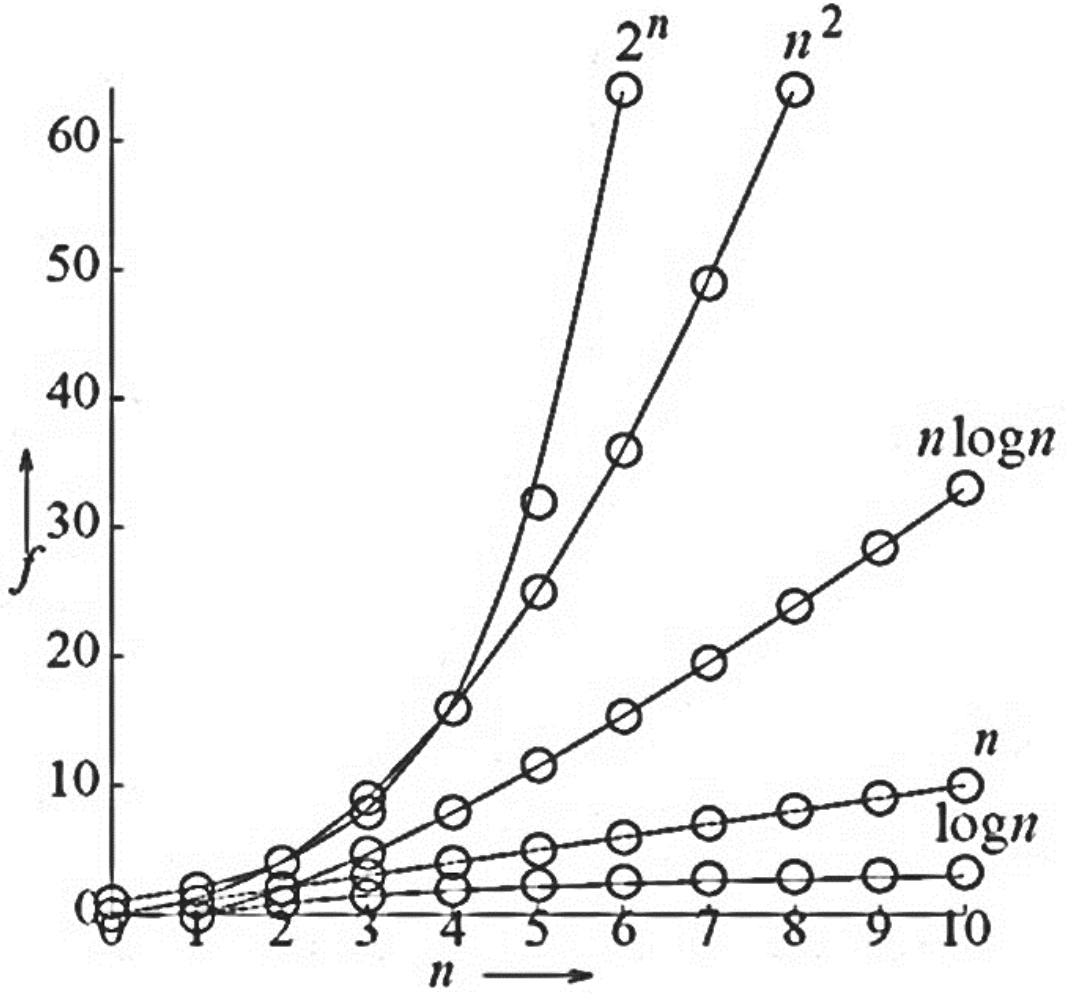
Παράδειγμα 1 (εύρεση του μέγιστου στοιχείου ενός πίνακα)

```
int largest(int array[], int n) {
int currlarge = 0;
        for (int i=1; i<n; i++)     // Για κάθε val
        if (array[currlarge] < array[i])
        currlarge = i; // αποθήκευση της θέσης
        return currlarge;          // επιστροφή μεγαλύτερου
}
```

Παράδειγμα 2 ('χαζός' υπολογισμός της έκφρασης n(n-1))

```
sum = 0;
for (i=1; i<=n; i++)
  for (j=1; j<n; j++)
    sum++;
}
```

Γραφική Αναπαράσταση Ρυθμού Αύξησης



Καλύτερη, Χειρότερη, Μέση Περίπτωση

- Διαφορετικές είσοδοι του ίδιου μεγέθους συνήθως απαιτούν διαφορετικό χρόνο τρεξίματος.
- Παράδειγμα: Σειριακή αναζήτηση ενός στοιχείου (του *K*) σε έναν πίνακα *n* ακεραίων:
  - Ξεκινάμε από το πρώτο στοιχείο του πίνακα και εξετάζουμε στη σειρά κάθε στοιχείο, μέχρι να βρούμε το *K*.
- Ποια είναι η καλύτερη / χειρότερη / μέση περίπτωση;
- Αν και ο μέσος χρόνος δείχνει να είναι το πιο δίκαιο μέτρο, μπορεί να είναι **δύσκολο να υπολογιστεί**.

Ταχύτερος Υπολογιστής ή Ταχύτερος Αλγόριθμος;

- Τι συμβαίνει όταν αγοράζουμε έναν υπολογιστή 10 φορές ταχύτερο;

  n (n'): το μέγεθος εισόδου που μπορούμε να επεξεργαστούμε σε 1 ώρα με τον αργό (ταχύ) υπολογιστή, χρόνος που αντιστοιχεί σε έστω 10,000 (100,000) πράξεις.

f(*n*)	*n*	*n'*	Αλλαγή	*n'*/*n*
$10n$	1,000	10,000	$n' = 10n$	10.00
$20n$	500	5,000	$n' = 10n$	10.00
$5n \log n$	250	1,842	$\sqrt{10}\, n < n' < 10n$	7.37
$2n^2$	70	223	$n' = \sqrt{10}n$	3.16
2^n	13	16	$n' = n + 3$	1.23

Συμβολισμός Όμικρον Κεφαλαίο (O)

- **Ορισμός**: f(n) = O(g(n) αν και μόνο αν υπάρχουν θετικές σταθερές c και n0 τέτοιες ώστε f(n) ≤ cg(n) για όλα τα n ≥ n0.
- **Παράδειγμα**: Ο αλγόριθμος ... είναι O(n2) [καλύτερης, μέσης, χειρότερης] περίπτωσης.
- **Ερμηνεία**: Για όλα τα αρκετά μεγάλα σύνολα δεδομένων (δηλαδή n ≥ n0), ο αλγόριθμος πάντα τερματίζει σε λιγότερα από cg(n) βήματα στην [καλύτερη, μέση, χειρότερη] περίπτωση.
- Αποτελεί **άνω όριο** κόστους.

Συμβολισμός Όμικρον Κεφαλαίο (Ο)

Ο συμβολισμός Ο δηλώνει ένα άνω όριο.

Παράδειγμα: Εάν $f(n) = 3n^2$ τότε η $f(n)$ είναι $O(n^2)$.

Επιθυμούμε όσο το δυνατό στενότερα όρια:

Παρόλο που ισχύει ότι η συνάρτηση $f(n) = 3n^2$ είναι $O(n^3)$, προτιμούμε $O(n^2)$.

Παραδείγματα Χρήσης του O

- Παράδειγμα 1: Εύρεση της τιμής X σε έναν πίνακα (μέσο κόστος) $f(n) = c_s n/2$.

  Για όλα τα n, $c_s n/2 \leq c_s n$.

  Επομένως, $f(n)$ είναι $O(n)$ για $n_0=1$ και $c=c_s$

- Παράδειγμα 2: $f(n) = c_1 n^2 + c_2 n$ στη μέση περίπτωση.

  $c_1 n^2 + c_2 n \leq c_1 n^2 + c_2 n^2 \leq (c_1 + c_2) n^2$ για όλα τα n

  Άρα $f(n) \leq cn^2$ για $c = c_1 + c_2$ και $n_0 = 1$

  Επομένως, $f(n)$ είναι $O(n^2)$ εξ ορισμού

- Παράδειγμα 3: $f(n) = c$. Λέμε ότι είναι $O(1)$

Μια Συνηθισμένη Παρεξήγηση

- "Η καλύτερη περίπτωση για τον αλγόριθμό μου είναι n=1 επειδή αυτό είναι το γρηγορότερο."

  ΛΑΘΟΣ!

- Το Ο αναφέρεται σε ένα **αυξανόμενο ρυθμό (rate)** καθώς το n τείνει στο ∞.

- Η καλύτερη περίπτωση ορίζεται ως: **ποια** είσοδος μεγέθους n είναι φθηνότερη από όλες τις εισόδους μεγέθους n.

Συμβολισμός Ωμέγα Κεφαλαίο (Ω)

- **Ορισμός**: $f(n) = \Omega(g(n)$ αν και μόνο αν υπάρχουν θετικές σταθερές c και n_0 τέτοιες ώστε $\boldsymbol{f}(n) \geq cg(n)$ για όλα τα $n \geq n_0$
- **Παράδειγμα**: Ο αλγόριθμος ... είναι $\Omega(n^2)$ [καλύτερης, μέσης, χειρότερης] περίπτωσης.
- **Ερμηνεία**: Για όλα τα αρκετά μεγάλα σύνολα δεδομένων (δηλαδή $n \geq n_0$), ο αλγόριθμος πάντα τερματίζει σε περισσότερα από $cg(n)$ βήματα στην [καλύτερη, μέση, χειρότερη] περίπτωση.
- Αποτελεί **κάτω όριο** κόστους.

Παράδειγμα Χρήσης του Ω

$f(n) = c_1 n^2 + c_2 n$

$c_1 n^2 + c_2 n \geq c_1 n^2$ για όλα τα n

Άρα $f(n) \geq cn^2$ για $c = c_1$ και $n_0 = 1$.

Επομένως, η $f(n)$ είναι $\Omega(n^2)$ εξ ορισμού.

Θέλουμε το μεγαλύτερο κάτω όριο.

Συμβολισμός Θήτα Κεφαλαίο (Θ)

- Όταν τα Ο και Ω ταυτιστούν, το δηλώνουμε αυτό με χρήση του συμβολισμού Θ.

- **Ορισμός**: $f(n) = \Theta(g(n)$ αν και μόνο αν υπάρχουν θετικές σταθερές c_1, c_2 και n_0 τέτοιες ώστε $c_1 g(n) \leq f(n) \leq c_2 g(n)$ για όλα τα $n \geq n_0$

- **Παρατήρηση**: Ένας αλγόριθμος λέμε ότι είναι Θ(g(n) εάν είναι ταυτόχρονα O(g(n) και Ω(g(n)

Κανόνες Απλοποίησης

1. Εάν $f(n) = O(g(n))$ και $g(n) = O(h(n))$, τότε $f(n) = O(h(n))$
2. Εάν $f(n) = O(kg(n))$ για σταθερά $k > 0$, τότε $f(n) = O(g(n))$
3. Εάν $f1(n) = O(g1(n))$ και $f2(n) = O(g2(n))$, τότε $(f1 + f2)(n) = O(\max(g1(n), g2(n)))$
4. Εάν $f1(n) = O(g1(n))$ και $f2(n) = O(g2(n))$, τότε $f1(n)f2(n) = O(g1(n)g2(n))$

Ασυμπτωτικές Ταυτότητες

	$f(n)$	Asymptotic
E1	c	$\oplus(1)$
E2	$\sum_{i=0}^{k} c_i n^i$	$\oplus(n^k)$
E3	$\sum_{i=1}^{n} i$	$\oplus(n^2)$
E4	$\sum_{i=1}^{n} i^2$	$\oplus(n^3)$
E5	$\sum_{i=1}^{n} i^k, k>0$	$\oplus(n^{k+1})$
E6	$\sum_{i=0}^{n} r^i, r>1$	$\oplus(r^n)$
E7	$n!$	$\oplus(n\ (n/e)^n)$
E8	$\sum_{i=1}^{n} 1/i$	$\oplus(\log n)$

$\oplus$ can be any one of O, Ω, and Θ

Παραδείγματα

- Παράδειγμα 1: a = b;

  Αυτή η ανάθεση παίρνει σταθερό χρόνο c, οπότε είναι Θ(1)

Παράδειγμα 2: υπολογισμός n(n+1)/2

```
sum = 0;
for (i=1; i<=n; i++)
  sum += i;
```

Παράδειγμα 3: (εναλλακτικός) υπολογισμός n(n+1)/2

```
sum = 0;
for (i=1; i<=n; i++)
  for (j=1; j<=i; j++)
    sum++;
```

- Σύγκριση 2 και 3: Υπολογισμός ίδιας έκφρασης με διαφορετικό κόστος! Θ(n) έναντι Θ(n2)

Παραδείγματα (συν.)

Παράδειγμα 4:

υπολογισμός n^2

```
sum = 0;
for (i=1; i<=n; i++)
  for (j=1; j<=n; j++)
    sum++;
```

Σύγκριση 3 και 4:
Υπολογισμός διαφορετικών εκφράσεων με ίδιο κόστος!
$\Theta(n^2)$

Παράδειγμα 5:

```
sum = 0;
for (k=1; k<=n; k*=2)
  for (j=1; j<=n; j++)
    sum++;
```

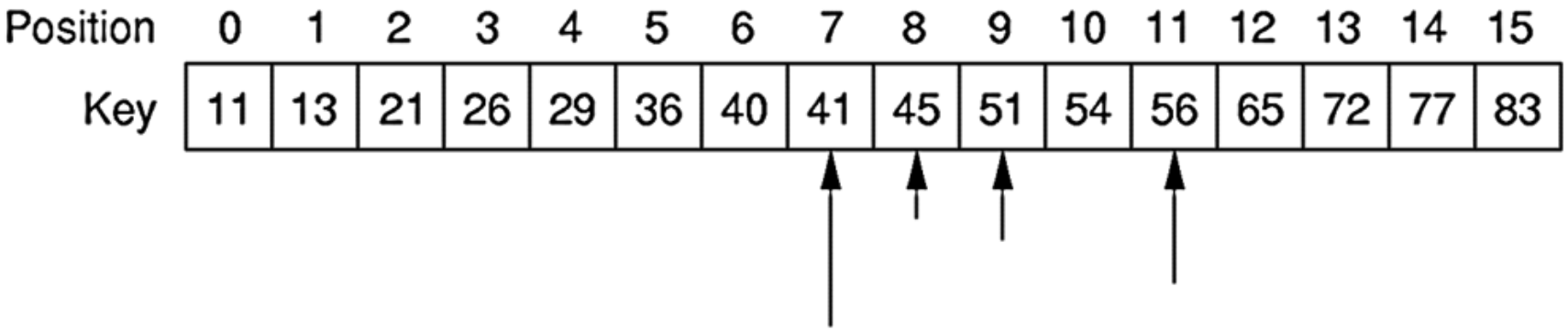
Κόστος $\Theta(n \log n)$

Παράδειγμα 6:

```
sum = 0;
for (k=1; k<=n; k*=2)
  for (j=1; j<=k; j++)
    sum++;
```

Κόστος $\Theta(n)$

Παράδειγμα: Δυαδική Αναζήτηση



- Πόσα στοιχεία εξετάζουμε στη **χειρότερη περίπτωση**?
  - Απάντηση: 4 (= $\log_2 16$)

Δυαδική Αναζήτηση (συν.)

```
// Επιστροφή της θέσης του στοιχείου με τιμή
// x στον ταξινομημένο πίνακα a μεγέθους n.

int BinarySearch(int a[], int n, int x) {
   int left = 0; int right = n-1;                  // όρια πίνακα
   while (left <= right) {                         // σταμάτα όταν τα
                                                   // όρια ταυτιστούν
      int middle = (left + right) / 2;
                                                   // έλεγχος του μεσαίου στοιχείου
      if (x == a[middle]) return middle;           // βρέθηκε
      if (x > a[middle]) left = middle + 1;
      else right = middle – 1;
      // αναζήτηση στο δεξί ή αριστερό μισό, αντίστοιχα
   }
   return -1;                                      // ένδειξη ότι δεν βρέθηκε
}
```

Χωρικά Όρια Κόστους – Ισορροπία Μεταξύ Χρονικού / Χωρικού Κόστους

- Τα χωρικά όρια μπορούν επίσης να αναλυθούν με χρήση ασυμπτωτικής ανάλυσης.
- **Χρόνος**: Αλγόριθμος που επιλύει το πρόβλημα.
- **Χώρος** της δομής δεδομένων που απαιτείται για την υλοποίηση του αλγορίθμου.
- Μπορούμε να μειώσουμε το χρόνο εάν θελήσουμε να θυσιάσουμε το χώρο (και αντίστροφα).
  - Σκεφτείτε (στην πορεία του μαθήματος) παραδείγματα ...

Παράδειγμα Ι

- Έστω η συνάρτηση $g(n) = n^2 + n + 10$

- Λέμε ότι $g(n) = O(n^2)$, διότι μπορούμε να βρούμε μία σταθερά c, και έναν αριθμό n_0 έτσι ώστε $\forall n > n_0$ $\boldsymbol{n^2 + n + 10 < c\, n^2}$

- Μπορείτε να βρείτε τιμές για τα c και n_0 ;

Π.χ. c=15, n_0=1

Παράδειγμα ΙΙ

- Δίνεται ένας σάκος που περιέχει αριθμούς.
  - Ζητείται να βρεθεί ο μεγαλύτερος αριθμός που υπάρχει στο σάκο. Κάθε φορά μπορούμε να βγάλουμε έναν αριθμό από το σάκο. Αν έχουμε n αριθμούς, ποια η πολυπλοκότητα χρόνου της μεθόδου;

Παράδειγμα III

- Αν ξέρουμε ότι ένας αλγόριθμος έχει πολυπλοκότητα O(n) σε ένα σύστημα Pentium IV στα 3 GHz, πόση είναι η πολυπλοκότητα σε ένα σύστημα Pentium III στα 500 MHz;

ΠΡΟΣΟΧΗ: Η πολυπλοκότητα ενός αλγορίθμου είναι ανεξάρτητη μηχανής.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Τέλος ενότητας

Επεξεργασία: <Μαυρίδης Απόστολος>
Θεσσαλονίκη, <Εαρινό εξάμηνο 2013-2014>

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

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.