

Δομές Δεδομένων: Εισαγωγικά

Δημήτρης Φωτάκης

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων
Πανεπιστήμιο Αιγαίου

Δομές Δεδομένων

- 4 ώρες **Θεωρία** (Δ. Φωτάκης, **Τετάρτη 17-19** και **Πέμπτη 9-11**):
 - **Γραπτή Πρόοδος**: 10-14 Δεκεμβρίου, 20% Βαθμού (ΒΠ).
 - **Τελική Εξέταση**: Φεβρουάριος, 50% Βαθμού (ΒΤΕ).
- 2 ώρες **Εργαστήριο** (Ε. Καρύμπαλη, **Τετάρτη 11-13**, Α - Λ, και **Τετάρτη 13-15**, Μ - Ω):
 - **Υποχρεωτικό**: 30% Βαθμού (ΒΕργ), **ΒΕργ ≥ 5**
 - **8 Ασκήσεις**: Εκφώνηση 1 εβδομάδα πριν, προετοιμασία, υλοποίηση στο εργαστήριο (με καθοδήγηση), **εξέταση και παράδοση** στο εργαστήριο.
 - **Καμία παράταση** στην παράδοση των ασκήσεων!
 - **1η εργασία**: Ανακοίνωση 18 Οκτωβρίου. Παράδοση 24 Οκτωβρίου.
- **Τελικός Βαθμός**: $0.5 \times \text{ΒΤΕ} + 0.2 \times \text{ΒΠ} + 0.3 \times \text{ΒΕργ}$
εφόσον $\text{ΒΓ} \geq 5$ **και** $\text{ΒΤΕ} \geq 5$.

Δομές Δεδομένων

Εισαγωγικά — σελ. 2/14

Βιβλιογραφία - Ύλη

- Sartaj Sahni, **Δομές Δεδομένων, Αλγόριθμοι, και Εφαρμογές σε C++**, McGraw Hill, 1998. Μετάφραση Ι. Θεοδορίδης και Ι. Μανωλόπουλος.
- Cormen, Leiserson, Rivest, Stein, **Introduction to Algorithms, 2nd Edition**, MIT Press, 2002.
- **Σημειώσεις** διδάσκοντα (όπου σημαντική διαφοροποίηση από βιβλίο).
- **Ότιδήποτε** άλλο (ασκήσεις, παραδείγματα, εφαρμογές) αναφέρεται στις **διαλέξεις**.
- Σημειώσεις, ανακοινώσεις, άλλο υλικό:
http://www.icsd.aegean.gr/lecturers/fotakis/data_structures.html
- **Παρακολούθηση** διαλέξεων και ενεργή **συμμετοχή** στο εργαστήριο.
- **Μηδενική ανοχή** σε θέματα αντιγραφής, αθέμιτης συνεργασίας.

Δομές Δεδομένων

Εισαγωγικά — σελ. 3/14

Αντικείμενο

- **Δομές Δεδομένων**: **αναπαράσταση, οργάνωση**, και **διαχείριση** συνόλων αντικειμένων για αποδοτική **ενημέρωση** και **ανάκτηση** πληροφορίας.
- **Αναπαράσταση**: Γλώσσες προγραμματισμού υποστηρίζουν **βασικούς τύπους**: αριθμοί, συμβολοσειρές, σύνολα, πίνακες. Παρέχουν **μηχανισμούς** για άλλους τύπους: δομές (structures), αντικείμενα (objects).
- **Διαχείριση**: Ταξινόμηση, αναζήτηση, μεγαλύτερο / μικρότερο, (χρονικά) πρώτο / τελευταίο, τροποποίηση, εισαγωγή, διαγραφή, εξερεύνηση.
- Έτοιμες λύσεις - **συνταγές για αποδοτική** διαχείριση δεδομένων. **Ανάλυση** δείχνει **καταλληλότερη** λύση για κάθε εφαρμογή.
 - Ξέρουμε ότι υπάρχουν και τα χρησιμοποιούμε όπως και όποτε πρέπει.
 - Επιλέγουμε καταλληλότερο με βάση απαιτήσεις.
 - Τροποποίηση / προσαρμογή σε ιδιαίτερες απαιτήσεις.
 - Δομημένος σχεδιασμός και ανάπτυξη.

Δομές Δεδομένων

Εισαγωγικά — σελ. 4/14

Δομή Μαθήματος

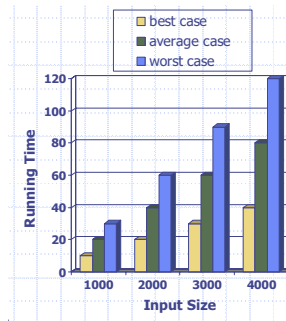
- **Βασικές έννοιες** ανάλυσης αλγορίθμων (2 διαλ.).
- **Στοιχειώδεις δομές δεδομένων** (5 διαλ: πίνακες, λίστες, ουρές, ουρές προτεραιότητας).
- **Ταξινόμηση - αναζήτηση - επιλογή - διαίρει και βασίλευε** (6 διαλ.).
- Δέντρα, (ζυγισμένα) **δέντρα αναζήτησης** (5 διαλ.) – **Πρόοδος**.
- **Ένωση και εύρεση** ξένων συνόλων (1 διαλ.).
- **Κατακερματισμός** (hashing) (2 διαλ.).
- Αναπαράσταση και εξερεύνηση **γραφημάτων** (3 διαλ.).

Αλγόριθμος - Πρόβλημα - Στιγμιότυπο

- **Αλγόριθμος**: “Συνταγή” για την επίλυση ενός **προβλήματος**.
Σαφώς ορισμένη διαδικασία για την **επίλυση προβλήματος** σε **πεπερασμένο** χρόνο από υπολογιστική **μηχανή**.
Ευκλείδης ΜΚΔ, αριθμητικές πράξεις, ταξινόμηση, αναζήτηση.
- **Πρόβλημα**: Μετασχηματισμός **εισόδου** (input) σε **εξόδο** (output).
Ορίζει **ακριβώς** μορφή δεδομένων εισόδου και εξόδου.
Πολλαπλασιασμός, ταξινόμηση, εύρεση στοιχείου, συντομότερο μονοπάτι.
- **Στιγμιότυπο**: Δεδομένα που συμφωνούν με περιορισμούς προβλήματος.
Έγκυρη είσοδος προβλήματος.
 5×20 , $(30, 10, 20, 40)$, $(15, (10, 20, 30))$.
- **Διατύπωση** αλγορίθμων: **Ψευδογλώσσα** ανάμεσα σε C και Pascal.
- **Ορθότητα** αλγορίθμου: Λύνει / απαντάει σωστά σε όλα τα στιγμιότυπα.
Λάθος: **αντι-παράδειγμα**. Ορθότητα: **μαθηματική απόδειξη**.
- Προβλήματα λύνονται από **πολλούς σωστούς** αλγόριθμους:
Ποιός είναι ο **καλύτερος** (για συγκεκριμένη εφαρμογή);

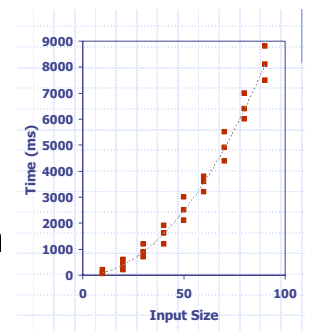
Ανάλυση Αλγορίθμων

- **Πολυπλοκότητα**: ποσότητα **υπολογιστικών πόρων** που απαιτεί ο αλγόριθμος σαν **αύξουσα συνάρτηση** μεγέθους στιγμιότυπου που επιλύει.
χρόνος, μνήμη, επεξεργαστές, επικοινωνία μέσω δικτύου
- **Μέγεθος στιγμιότυπου n** : Αριθμός bits για αναπαράσταση στη μνήμη.
Πλήθος **βασικών συνιστωσών** που αποτελούν μέτρο δυσκολίας του στιγμιότυπου και σαν συνάρτηση των οποίων **εκφράζουμε** την πολυπλοκότητα.
Πράξεις n ψηφία. Ταξινόμηση/αναζήτηση n στοιχεία.
- **Ανάλυση**: **Μαθηματική εκτίμηση** πολυπλοκότητας (χρόνος n , $n \log n$, n^2).
Διαφοροποίηση από ένα στιγμιότυπο σε άλλο (π.χ. γραμμική αναζήτηση).
Ανάλυση **χειρότερης**, **μέσης**, και **καλύτερης** περίπτωσης.
- Ανάλυση καθορίζει καταλληλότερη λύση ανάλογα με **απαιτήσεις εφαρμογής** (π.χ. στατική - δυναμική, χώρος - μνήμη).



Πειραματική Μελέτη

- Υλοποίηση αλγορίθμου σε **πρόγραμμα**.
- Δημιουργία **στιγμιοτύπων** διαφορετικού μεγέθους και **σύνθεσης**.
- **Καταγραφή πόρων** για κάθε εκτέλεση.
- **Απεικόνιση** αποτελεσμάτων σε γραφική παράσταση και **εξαγωγή συμπερασμάτων**.
- **Περιορισμοί - Δυσκολίες**
 - Υλοποίηση χρονοβόρα και ενδεχομένως δύσκολη.
 - Αποτελέσματα όχι αντιπροσωπευτικά για άλλα στιγμιότυπα.
 - Σύγκριση υποθέτει ίδια υπολογιστικά περιβάλλοντα.



Θεωρητική Ανάλυση

- Δεν απαιτεί υλοποίηση αλλά (σαφή) **περιγραφή** του αλγορίθμου.
- Λαμβάνει υπόψη **όλα** τα στιγμιότυπα (χειρότερη – μέση περίπτωση).
- Δίνει υπολογιστικούς πόρους σαν **συνάρτηση του μεγέθους n** της εισόδου.
- Υπολογίζει **στοιχειώδεις** ανάγκες υπολογιστικούς πόρους.
Ασυμπτωτική εκτίμηση: **Ανεξάρτητη** του υπολογιστικού περιβάλλοντος.
- Ανάγκες σε πόρους που εξαρτώνται από αλγόριθμο και όχι άλλους παράγοντες.
π.χ. αρχιτεκτονική, λειτουργικό σύστημα, compiler, φορτίο υπολογιστικού συστήματος.
- Αποτελέσματα **επιβεβαιώνονται** εύκολα.
- **Μαθηματικό υπόβαθρο**: Διακριτά μαθηματικά (συνδυαστική, αθροίσματα, αναδρομικές σχέσεις, γραφήματα, μαθηματική λογική), πιθανότητες, ...

Υπολογιστικό Μοντέλο

- **Μηχανή Άμεσης Προσπέλασης Μνήμης** (Random Access Machine, RAM).
Ιδεατό μονο-επεξεργαστικό σύστημα.
- Ένας επεξεργαστής, **ακολουθιακή** εκτέλεση εντολών.
- **Απεριόριστες** θέσεις μνήμης που προσπελάζονται σε **μοναδιαίο χρόνο**.
- **Στοιχειώδη** υπολογιστικά βήματα εκτελούνται σε **μοναδιαίο χρόνο**.
Ανάγνωση / εγγραφή στη μνήμη, αριθμητικές και λογικές πράξεις, συγκρίσεις, εντολές ελέγχου ροής.

Πολυπλοκότητα Χώρου Αποθήκευσης

- Πόσες θέσεις **μνήμης** δεσμεύει ο **αλγόριθμος** όταν εφαρμόζεται σε στιγμιότυπο **μεγέθους n** ;
Το αποτέλεσμα είναι μια (αύξουσα) συνάρτηση του n .
- Ενδιαφέρουν ανάγκες σε μνήμη που **αυξάνουν** όσο μεγαλώνει το n .

```
int max(int A[ ], int n)
    max ← A[0];
    for i ← 1 to n – 1 do
        if A[i] > max then
            max ← A[i];
    return(max);
```

Είσοδος: n **θέσεις** μνήμης
Αλγόριθμος: **2 θέσεις** μνήμης (i , max)!
Σταθερός χώρος, **in-place**.

```
long factorial(int n)
    if n ≤ 1 return(1);
    return(n × factorial(n – 1));
```

$c \cdot n$ **θέσεις** μνήμης (c σταθερά).
 c θέσεις για κάθε αναδρομική κλήση.
Γραμμικός χώρος.

Χώρος αποθήκευσης για **επαναληπτική υλοποίηση**;

Πολυπλοκότητα Χρόνου Εκτέλεσης

- Σε πόσο **χρόνο** ολοκληρώνεται ο **αλγόριθμος** όταν εφαρμόζεται σε στιγμιότυπο **μεγέθους n** ;
Το αποτέλεσμα είναι μια (αύξουσα) συνάρτηση του n .
- Μέτρηση **στοιχειωδών λειτουργιών** (π.χ. βασικών εντολών, αριθμητικών και λογικών πράξεων, συγκρίσεων, αναθέσεων) σαν **συνάρτηση του n** .
Ανεξάρτητες από υλοποίηση και υπάρχουν στον ψευδοκώδικα.
Μετράμε αυτές που **συνεισφέρουν περισσότερο** στο χρόνο εκτέλεσης.

```
int max(int A[ ], int n)
    max ← A[0];
    for i ← 1 to n – 1 do
        if A[i] > max then
            max ← A[i];
    return(max);
```

Μέγεθος είσοδου: n
Στοιχειώδεις λειτουργίες: $3n - 1$ (χειρ.π.)
 $2n$ (καλ.π.)
(μέτρηση με global μεταβλητή count)
Γραμμικός χρόνος.

- **Γραμμική αύξηση** χρόνου εκτέλεσης με #στοιχείων είναι **ανεξάρτητη του υπολογιστή**.

Μέτρηση Λειτουργιών

- Μέτρηση **στοιχειωδών λειτουργιών** δίνει **ακριβή εκτίμηση**.
Διάκριση μεταξύ αλγορίθμων με **συγκρίσιμο** χρόνο εκτέλεσης.
- Πολυώνυμο **βαθμού n** : $A(x) = \sum_{i=0}^n a_i x^i$ αναπαρίσταται με $[a_0, \dots, a_n]$.
Υπολογισμός **τιμής** $A(x)$ στο σημείο x :
$$A(x) = (\dots((a_n x + a_{n-1})x + a_{n-2})x + a_{n-3})x + \dots)x + a_0$$

<pre>float polyEval([a0, ..., an], x) val ← a0; y ← 1; for i ← 1 to n do y ← y × x; val ← val + a_i × y; return(val);</pre>	<pre>float Horner([a0, ..., an], x) val ← a_n; for i ← 1 to n do val ← val × x + a_{n-i}; return(val);</pre>
Λειτουργίες: $6n + 3$	Λειτουργίες: $4n + 2$
- Ενδιαφέρει **τάξη μεγέθους** χρόνου εκτέλεσης (με n) και όχι **ακριβής εκτίμηση** (συχνά δύσκολο να γίνει).

Ανταλλαγή Χώρου-Χρόνου

- Πρόβλημα λύνεται από αλγορίθμους με **διαφορετικές** πολυπλοκότητες **χώρου** και **χρόνου**.
- Επιτυγχάνουμε **μείωση χρόνου** με **αύξηση χώρου**.
- Επιτυγχάνουμε **μείωση χώρου** με **αύξηση χρόνου**.
- Ανταλλαγή χώρου - χρόνου - **time-space tradeoff**.
- Απαιτήσεις / περιορισμοί εφαρμογής καθορίζουν **προτιμότερο**.