ΑΕΠΠ 2015-2016

Α. Μπασούνας

basounas@sch.gr

Καθορισμός εξεταστέας – διδακτέας ύλης των πανελλαδικά εξεταζόμενων μαθημάτων της Γ΄ τάξης του Γενικού Λυκείου για το σχολικό έτος 2015–2016.

Αριθμ. ΥΑ 96080/Δ2-19-6-2015 (ΦΕΚ Β 1186)

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΣΕ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ (ΑΕΠΠ)

Ομάδας Προσανατολισμού Σπουδών Οικονομίας & Πληροφορικής

Από το βιβλίο «Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον» της Γ΄ τάξης Γενικού Λυκείου των Α. Βακάλη, Η. Γιαννόπουλου, Ν. Ιωαννίδη, Χ. Κοίλια, Κ. Μάλαμα, Ι. Μανωλόπουλου, Π. Πολίτη, έκδοση (Ι.Τ.Υ.Ε.) "Διόφαντος".

2. Βασικές Έννοιες Αλγορίθμων

2.1 Τι είναι αλγόριθμος.

2.3 Περιγραφή και αναπαράσταση αλγορίθμων.

2.4 Βασικές συνιστώσες/ εντολές ενός αλγορίθμου.

2.4.1 Δομή ακολουθίας.

2.4.2 Δομή Επιλογής.

2.4.3 Διαδικασίες πολλαπλών επιλογών (αφαιρείται η εντολή πολλαπλής επιλογής "Επίλεξε")

2.4.4 Εμφωλευμένες Διαδικασίες.

2.4.5 Δομή Επανάληψης.

ΠΑΡΑΤΗΡΗΣΕΙΣ (Κεφ. 2)

- Οι Αλγόριθμοι να υλοποιούνται σε αμιγώς προγραμματιστικό περιβάλλον και συγκεκριμένα αυτό της ΓΛΩΣΣΑΣ.
- Επισκόπηση της έννοιας του αλγορίθμου, των χαρακτηριστικών του και των τρόπων αναπαράστασής του, και εισαγωγή στα χαρακτηριστικά των γλωσσών προγραμματισμού και ειδικά της ΓΛΩΣΣΑΣ.
- Οι βασικές αλγοριθμικές δομές του κεφαλαίου 2 (ακολουθίας, επιλογής και επανάληψης) να διδαχθούν συνοπτικά και παράλληλα με το κεφάλαιο 7 και 8 στην κατεύθυνση της κάλυψης τυχόν γνωσιακών κενών από την προηγούμενη τάξη, με τις ασκήσεις να υλοποιούνται απ’ ευθείας σε ΓΛΩΣΣΑ.

3. Δομές Δεδομένων και Αλγόριθμοι

3.2 Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

3.3 Πίνακες

3.4 Στοίβα

3.5 Ουρά

3.6 Αναζήτηση

3.7 Ταξινόμηση

3.9 Άλλες δομές δεδομένων

ΠΑΡΑΤΗΡΗΣΕΙΣ (στο κεφάλαιο 3)

- Να προστεθούν ασκήσεις στη στοίβα και ουρά που επίσης θα υλοποιηθούν απ’ ευθείας σε ΓΛΩΣΣΑ. και με την **πρόσθεση της ενότητας 3.9 που θα διδαχθεί.**
- Οι δυναμικές δομές της ενότητας 3.9 (λίστες, δένδρα, γράφοι) να διδαχθούν αποκλειστικά ως θεωρία.
- Οι πίνακες να διδαχθούν παράλληλα με το κεφάλαιο 9 με τις ασκήσεις να υλοποιούνται απ’ ευθείας σε ΓΛΩΣΣΑ.
- Εισάγονται νέοι αλγόριθμοι αναζήτησης και ταξινόμησης σε πίνακες.

5. Ανάλυση Αλγορίθμων

5.1 Επίδοση αλγορίθμων

5.1.1 Χειρότερη περίπτωση ενός αλγορίθμου

5.1.2 Μέγεθος εισόδου ενός αλγορίθμου

5.1.3 Χρόνος εκτέλεσης προγράμματος ενός αλγορίθμου

5.1.4 Αποδοτικότητα αλγορίθμων

5.3 Πολυπλοκότητα αλγορίθμων

ΠΑΡΑΤΗΡΗΣΕΙΣ

- Στο κεφάλαιο 5 να διδαχθούν οι ενότητες 5.1 (επίδοση αλγορίθμων), και 5.3 (πολυπλοκότητα αλγορίθμων). Η έννοια της επίδοσης να εξεταστεί με αναφορά στους αλγορίθμους αναζήτησης και ταξινόμησης.
- Η πολυπλοκότητα αλγορίθμων θα διδαχθεί θεωρητικά με παραδείγματα και σε σύνδεση με την επίδοση χωρίς οι μαθητές να εμπλακούν σε ασκήσεις υπολογισμού της τάξης Ο ενός αλγορίθμου.

6. Εισαγωγή στον προγραμματισμό

6.3 Φυσικές και τεχνητές γλώσσες.

6.4 Τεχνικές σχεδίασης προγραμμάτων.

6.4.1 Ιεραρχική σχεδίαση προγράμματος.

6.4.2 Τμηματικός προγραμματισμός.

6.4.3 Δομημένος προγραμματισμός.

6.7 Προγραμματιστικά περιβάλλοντα.

7. Βασικά στοιχεία προγραμματισμού

7.1 Το αλφάβητο της ΓΛΩΣΣΑΣ.
7.2 Τύποι δεδομένων.
7.3 Σταθερές.
7.4 Μεταβλητές.
7.5 Αριθμητικοί τελεστές.
7.6 Συναρτήσεις.
7.7 Αριθμητικές εκφράσεις.
7.8 Εντολή εκχώρησης.
7.9 Εντολές εισόδου-εξόδου.
7.10 Δομή προγράμματος.

8. Επιλογή και επανάληψη

8.1 Εντολές Επιλογής
    8.1.1 Εντολή ΑΝ
8.2 Εντολές επανάληψης
    8.2.1 Εντολή ΟΣΟ...ΕΠΑΝΑΛΑΒΕ
    8.2.2 Εντολή ΜΕΧΡΙΣ_ΟΤΟΥ
    8.2.3 Εντολή ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ

9. Πίνακες

9.1 Μονοδιάστατοι πίνακες.

9.2 Πότε πρέπει να χρησιμοποιούνται πίνακες.

9.3 Πολυδιάστατοι πίνακες.

9.4 Τυπικές επεξεργασίες πινάκων.

10. Υποπρογράμματα

10.1 Τμηματικός προγραμματισμός.

10.2 Χαρακτηριστικά των υποπρογραμμάτων.

10.3 Πλεονεκτήματα του τμηματικού προγραμματισμού.

10.4 Παράμετροι.

10.5 Διαδικασίες και συναρτήσεις.

10.5.1 Ορισμός και κλήση συναρτήσεων.

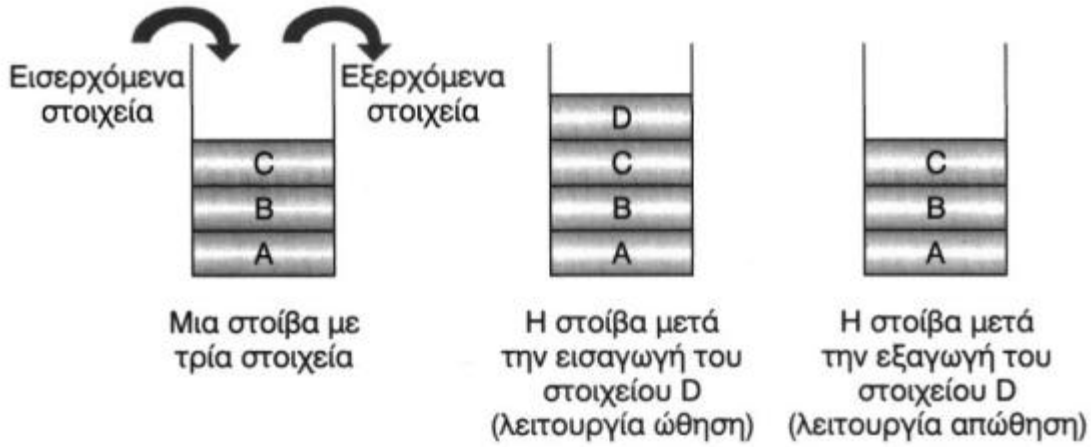
10.5.2 Ορισμός και κλήση διαδικασιών.

10.5.3 Πραγματικές και τυπικές παράμετροι.

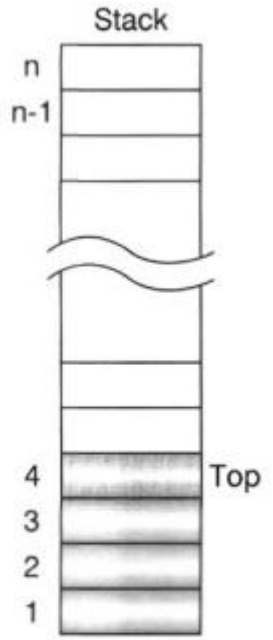
10.6 Εμβέλεια μεταβλητών – σταθερών

Ασκήσεις στη Στοίβα (Stack)

- Υλοποίηση στοίβας με πίνακα (**stack[100]**) χαρακτήρων
- Δείκτης **top**
- Ζητείται να υλοποιηθούν οι διαδικασίες **push** και **pop** οι οποίες θα επιστρέφουν (μέσω μιας λογικής μεταβλητής flag) την τιμή Αληθής αν εκτελέστηκαν σωστά ή την τιμή Ψευδής στην αντίθετη περίπτωση (Υπερχείλιση ή Υποχείλιση αντίστοιχα).
- Εισάγεται (ή εξάγεται) το στοιχείο **item**



Σχ. 3.2. Λειτουργίες στοίβας.



Σχ. 5.5 Υλοποίηση στοίβας με χρήση πίνακα

Διαδικασία push (stack, top, item, flag)

Μεταβλητές

Χαρακτήρες: stack[100], item

Ακέραιες: top

Λογικές: flag

Αρχή

Αν top < 100 **τότε** **! Έλεγχος υπερχείλισης**

top ← top+1

stack[top] ← item

flag ← Αληθής

αλλιώς

flag ← Ψευδής

Τέλος_Αν

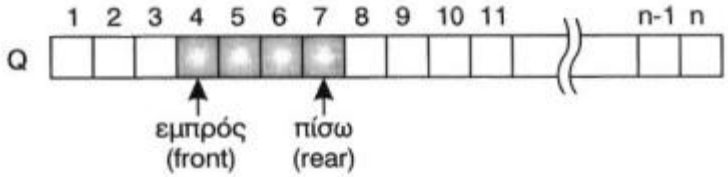
Τέλος_Διαδικασίας

```
Διαδικασία pop (stack, top, item, flag)
Μεταβλητές
    Χαρακτήρες: stack[100], item
    Ακέραιες: top                          ! Αρχική τιμή 0
    Λογικές: flag
Αρχή
      Αν top >= 1 τότε                     ! Έλεγχος υποχείλισης
          item← stack[top]
          top ← top - 1
          flag ← Αληθής
    αλλιώς
          flag ← Ψευδής
     Τέλος_Αν
Τέλος_Διαδικασίας
```

Ασκήσεις στη Ουρά (Queue)

- Υλοποίηση Ουράς με πίνακα (**queue[100]**) χαρακτήρων
- Δείκτες **front, rear**
- Ζητείται να υλοποιηθούν οι διαδικασίες **enqueue** και **dequeue** οι οποίες θα επιστρέφουν (μέσω μιας λογικής μεταβλητής flag) την τιμή Αληθής αν εκτελέστηκαν σωστά ή την τιμή Ψευδής στην αντίθετη περίπτωση
- Εισάγεται (ή εξάγεται) το στοιχείο **item**

Σχ. 3.4. Εισαγωγή και εξαγωγή από ουρά.



Σχ. 3.5 Υλοποίηση ουράς με χρήση πίνακα

Διαδικασία enqueue (queue, rear, item, flag)

Μεταβλητές

Χαρακτήρες: stack[100], item

Ακέραιες: rear

Λογικές: flag

Αρχή

Αν rear < 100 **τότε** **! Έλεγχος γεμάτης ουράς**

rear ← rear + 1

stack[rear] ← item

flag ← Αληθής

αλλιώς

flag ← Ψευδής

Τέλος_Αν

Τέλος_Διαδικασίας

Διαδικασία dequeue (queue, front, rear, item, flag)

Μεταβλητές

Χαρακτήρες: stack[100], item

Ακέραιες: front, rear ! **Αρχικά front=1, rear=0**

Λογικές: flag

Αρχή

Αν front <= rear **τότε** ! **Έλεγχος (όχι) άδειας ουράς**

item ← stack[front]

front ← front + 1

flag ← Αληθής

αλλιώς

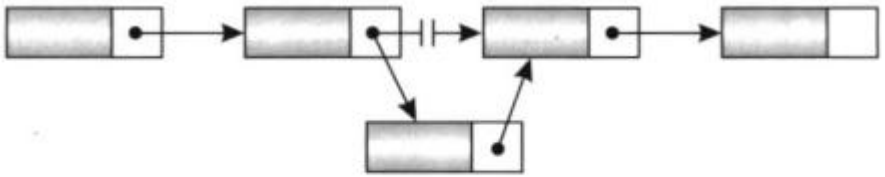
flag ← Ψευδής

Τέλος_Αν

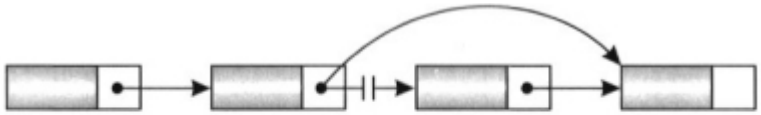
Τέλος_Διαδικασίας

3.9 Άλλες δομές δεδομένων

Σχ. 3.9. Μία λίστα με τέσσερις κόμβους

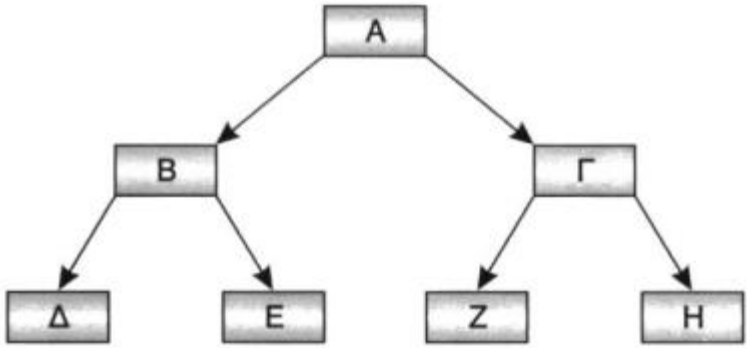


Σχ. 3.10. Εισαγωγή σε λίστα

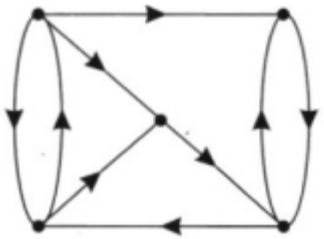


Σχ. 3.11. Διαγραφή κόμβου λίστας

3.9 Άλλες δομές δεδομένων



Σχ. 3.12. Δομή δένδρου



Σχ. 3.13. Ένας γράφος

Δυαδική αναζήτηση

- Εφαρμόζεται σε ταξινομημένο πίνακα
- Συγκρίνει την τιμή (key) που αναζητούμε με το μεσαίο στοιχείο του πίνακα.
- Αν είναι ίσα, τότε βρέθηκε.
- Αν όχι, τότε ανάλογα με την τιμή του key σε σχέση με το μεσαίο στοιχείο, αποκλείεται ο μισός πίνακας και η αναζήτηση συνεχίζεται στον άλλο μισό, όπου γίνεται με τον ίδιο τρόπο (ψάχνουμε στη μέση).

Δυαδική αναζήτηση

1	2	3	4	5	6	7	8	9	10
6	12	21	45	57	84	98	107	199	312

Ψάχνουμε για το στοιχείο 107
1η αναζήτηση στη θέση 5
2η αναζήτηση στη θέση 8 ➔ Βρέθηκε

Ψάχνουμε το στοιχείο 14
1η αναζήτηση στη θέση 5
2η αναζήτηση στη θέση 2
3η αναζήτηση στη θέση 3
4η αναζήτηση στη θέση 4 ➔ ΔΕΝ Βρέθηκε

Δυαδική αναζήτηση

Ψάχνουμε για την τιμή **key** στον ταξινομημένο (αύξουσα διάταξη) πίνακα **T[100]**

Η δυαδική αναζήτηση λειτουργεί ως εξής:

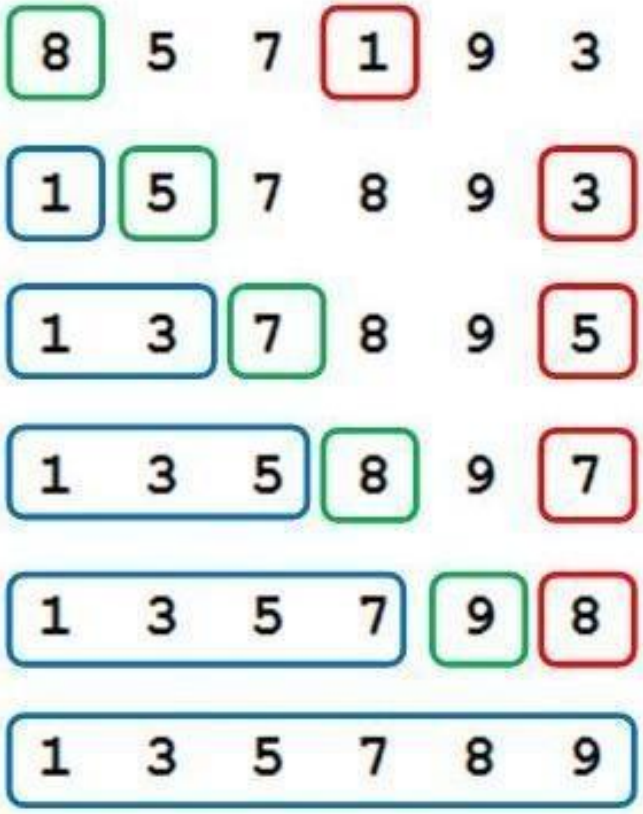
- Υπάρχουν οι δείκτες **start** και **end** που προσδιορίζουν το διάστημα μέσα στο οποίο κάνουμε την αναζήτηση (αρχικά start=1 και end=100)
- Υπολογίζουμε το μέσο **M** του διαστήματος [start, end]
- Αν T[M] = key, τότε το στοιχείο key εντοπίστηκε στη θέση M και η μέθοδος ολοκληρώθηκε.
- Αν T[M] < X, η αναζήτηση συνεχίζεται στο διάστημα [M+1, end]
- Αν T[M] > X, η αναζήτηση συνεχίζεται στο διάστημα [start, M-1]
- Αν κατά την εκτέλεση της μεθόδου, το start γίνει μεγαλύτερο του end, είμαστε σίγουροι ότι η τιμή X δεν υπάρχει μέσα στον πίνακα και η αναζήτηση σταματά.

```
. . .
start ← 1
end ← 100
found ← Ψευδής
position ← 0
ΟΣΟ (found=Ψευδής) και (start<=end) επανάλαβε
        Μ ← (start+end) DIV 2
        Αν Τ[Μ]=key τότε
                found ← Αληθής
                position ← Μ
        αλλιώς
                Αν Τ[Μ]>key τότε
                        end ← Μ -1
                Αλλιώς
                        start ← Μ + 1
                Τέλος_αν
        Τέλος_αν
Τέλος_επανάληψης
. . .
```

Άλλες μέθοδοι ταξινόμησης
1. Ταξινόμηση με επιλογή

- Στην **ταξινόμηση με επιλογή (Selection sort)** επιλέγεται το μικρότερο στοιχείο του πίνακα και τοποθετείται στην πρώτη θέση του πίνακα (αντιμετάθεση).
- Στο επόμενο βήμα εντοπίζεται το δεύτερο μικρότερο στοιχείο και τοποθετείται στη δεύτερη θέση του πίνακα.
- Η διαδικασία συνεχίζεται μέχρι να τοποθετηθούν όλα τα στοιχεία

Selection Sort.



Sorted List.
Current.
Exchange.

Total comparisons = $n(n - 1)/2$

$\sim O(n^2)$

Αλγόριθμος ταξινόμησης με επιλογή (Τετράδιο μαθητή 4.2.1)

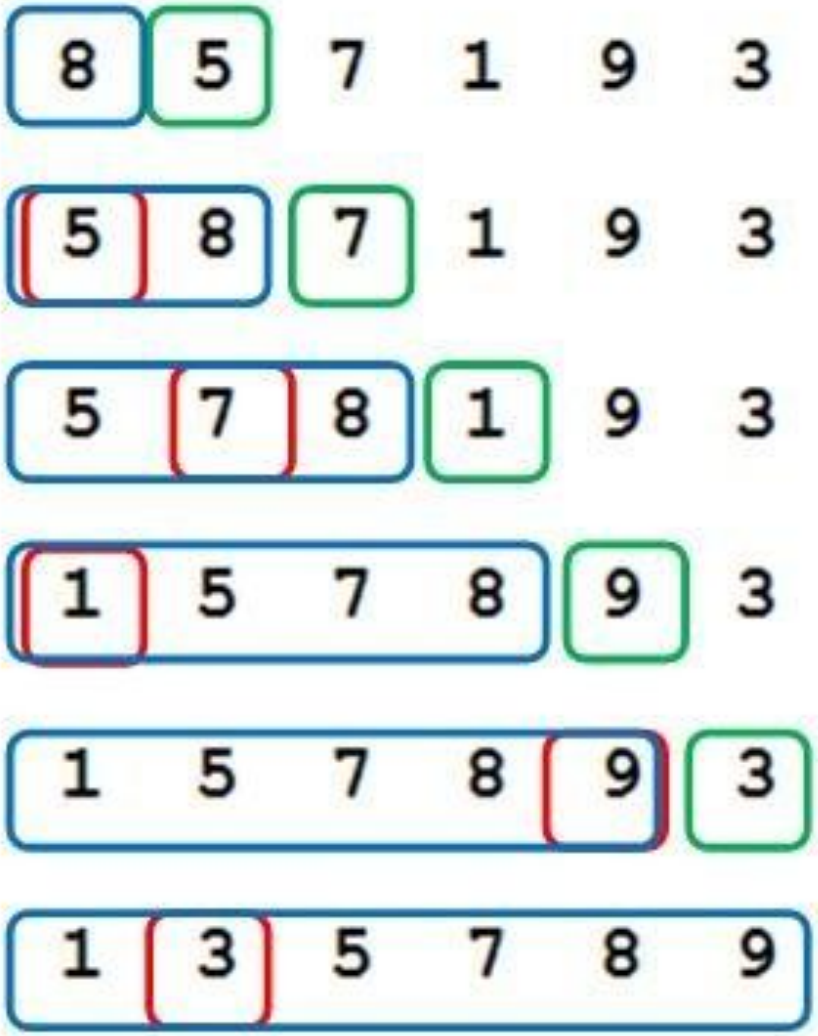
```
Αλγόριθμος Ταξινόμηση_επιλογής
Δεδομένα // Τ //
Για i από 1 μέχρι 100
    j ←i                                  ! Θέση ελάχιστου
    Για k από i+1 μέχρι 100
        Αν Τ[k] < Τ[j] τότε
            j ← k
        Τέλος_αν
    Τέλος_επανάληψης
    Αντιμετάθεσε Τ[i], Τ[j]
Τέλος_επανάληψης
Αποτελέσματα // Τ //
Τέλος
```

Αλγόριθμος ταξινόμησης με παρεμβολή (Βασίζεται στη ΔΣ3. Τετράδιο μαθητή, κεφ. 3)

Διατάξτε τα στοιχεία του πίνακα Τ[100] σε αύξουσα σειρά, με την παρακάτω μέθοδο:

- Αρχικά, συγκρίνουμε το δεύτερο με το πρώτο στοιχείο και αν χρειαστεί τα αντιμεταθέτουμε για να είναι στη σωστή σειρά.
- Στη συνέχεια ελέγχουμε το τρίτο στοιχείο και το τοποθετούμε στη σωστή σειρά σε σχέση με τα δύο πρώτα, μετακινώντας κάποια από αυτά «προς τα πάνω», αν χρειαστεί.
- Συνεχίζουμε με τον ίδιο τρόπο μέχρι και με το 100^{ο} στοιχείο του πίνακα.

Insertion sort (Card game)



Αλγόριθμος ταξινόμησης με παρεμβολή

```
Αλγόριθμος Ταξινόμηση_παρεμβολής
Δεδομένα // Τ //

Για i από 2 μέχρι 100
        j ← 1
        Όσο (Τ[j] < Τ[i]) επανάλαβε
                j ← j + 1
        Τέλος_επανάληψης
        Αν j < i τότε
                x ← Τ[i]
                Για k από i μέχρι j + 1 με_βήμα -1
                                Τ[k] ← Τ[k - 1]
                Τέλος_επανάληψης
                Τ[j] ← x
        Τέλος_αν
Τέλος_επανάληψης

Αποτελέσματα // Τ //
Τέλος
```

Αλγόριθμος ταξινόμησης με παρεμβολή -2

Έχει αναρτηθεί από τον συνάδελφο Σπ. Δουκάκη στην Πλατφόρμα Ψηφιακών Διδακτικών Σεναρίων του ΙΕΠ «ΑΙΣΩΠΟΣ» http://aesop.iep.edu.gr/

Αλγόριθμος ταξινόμησης με παρεμβολή - 2ος τρόπος

```
Αλγόριθμος Ταξινόμηση_παρεμβολής2
Δεδομένα // Τ //

Για i από 2 μέχρι 100
        x ← T[i]
        j ← i - 1
        Βρέθηκε ← Αληθής
        Όσο (Βρέθηκε= Αληθής) και (j>=1]) επανάλαβε
        Αν x < T[j] τότε
                T[j+1] ← T[j]
                j ← j-1
        Αλλιώς
                Βρέθηκε ← Ψευδής
        Τέλος_αν
        T[j+1 ]← x
Τέλος_επανάληψης

Αποτελέσματα // Τ //
Τέλος
```

Κεφάλαιο 5
Ανάλυση Αλγορίθμων

5.1 Επίδοση αλγορίθμων

Ερωτήματα

1. Πώς υπολογίζεται ο χρόνος εκτέλεσης ενός αλγορίθμου;
2. Πώς συγκρίνονται μεταξύ τους διαφορετικοί αλγόριθμοι;
3. Πώς γνωρίζουμε αν ένας αλγόριθμος είναι βέλτιστος;

5.1.1. Χειρότερη περίπτωση ενός αλγορίθμου

- Μέγιστο κόστος εκτέλεσης σε σχέση με υπολογιστικούς πόρους
- Συνήθως εκφράζεται με μέτρηση του μέγιστου πλήθους βασικών πράξεων:
  - ανάθεση τιμής
  - αριθμητική πράξη
  - σύγκριση δύο τιμών

(Π.χ. στην εντολή Α← Β*2 εκτελούνται 2 πράξεις)

Χειρότερη περίπτωση- παράδειγμα 1

```
Αλγόριθμος Παράδειγμα 1
n←10
Αρχή_επανάληψης
    Διάβασε m
    n ← n -1
Μέχρις_ότου (m=0) ή (n=0)
Εκτύπωσε m
Τέλος Παράδειγμα1
```

Χειρότερη περίπτωση: 10 επαναλήψεις

Χειρότερη περίπτωση- παράδειγμα 2

Σύγκριση **Σειριακής** και **Δυαδικής** αναζήτησης σε πίνακα 100 στοιχείων

Χειρότερη περίπτωση: όταν το στοιχείο key που αναζητούμε **δεν** υπάρχει στον πίνακα.

1. **Σειριακή αναζήτηση**: **π=100 συγκρίσεις**

Χειρότερη περίπτωση- παράδειγμα 2

2. **Δυαδική αναζήτηση**: σε κάθε σύγκριση το μέγεθος του πίνακα αναζήτησης γίνεται ½, ¼, 1/8 …

Άρα π είναι ο μικρότερος ακέραιος για τον οποίο ισχύει $2^{\pi}>=100$, άρα $\pi>=\log_2 100$, άρα **π=7**

** Βέβαια στην δυαδική αναζήτηση εκτός από τη σύγκριση* T[M]=key *υπάρχει και ο υπολογισμός του* $M \leftarrow$(start+end) *σε κάθε επανάληψη. Και πάλι όμως η δυαδική μέθοδος παρουσιάζει πολύ καλύτερη επίδοση για μεγάλα μεγέθη πίνακα.*

5.1.3. Χρόνος εκτέλεσης προγράμματος ενός αλγορίθμου

Υπολογίστε την επίδοση του παρακάτω αλγορίθμου με βάση τον αριθμό πράξεων που θα εκτελεστούν

```
Αλγόριθμος Παράδειγμα2
x ← 123
y ← 234
Για i από 0 μέχρι 4
    Εκτύπωσε i
    z ← x * y
Τέλος_επανάληψης
Εκτύπωσε x
Εκτύπωσε y
Εκτύπωσε z
Τέλος Παράδειγμα2
```

Εντολή αλγορίθμου	Αριθμός πράξεων
ανάθεση τιμών στα x και y	2
Βρόχος επανάληψης	
αρχική τιμή i	1
έλεγχος i	6
αύξηση i	5
εκτύπωση i	5
υπολογισμός z (2X5)	10
Εκτύπωση x, y, z	3
ΣΥΝΟΛΟ	**32**

5.1.4. Αποδοτικότητα αλγορίθμων

Σύγκριση δύο αλγορίθμων που παράγουν το ίδιο αποτέλεσμα ως προς το χρόνο εκτέλεσης, τη μνήμη που απαιτούν κλπ.

Προϋποθέσεις:

- Κωδικοποίηση στην ίδια γλώσσα προγραμματισμού
- Ίδιος μεταγλωττιστής
- Ίδιο υπολογιστικό σύστημα
- Ακριβώς τα ίδια δεδομένα εισόδου

5.3. Πολυπλοκότητα αλγορίθμων

Μέτρηση της επίδοσης ενός αλγορίθμου

Α. Εμπειρικός τρόπος:

Υλοποίηση σε γλώσσα προγραμματισμού και εκτέλεση για συγκεκριμένο σύνολο δεδομένων για υπολογισμό χρόνου επεξεργασίας και χωρητικότητας μνήμης.

Β. Θεωρητικός τρόπος

Αν η μεταβλητή **n** καθορίζει το μέγεθος του προβλήματος, η χρονική πολυπλοκότητα και η πολυπλοκότητα χώρου εκφράζονται μέσω της συνάρτησης f(n)

Ορισμός: Αν η πολυπλοκότητα ενός αλγορίθμου είναι f(n), τότε λέγεται ότι είναι τάξης O(g(n)), αν υπάρχουν δύο θετικοί ακέραιοι c και n0, έτσι ώστε για κάθε n ≥ n0 να ισχύει:

$$|f(n)| \leq c|g(n)|$$

Παράδειγμα. Έστω ότι η πολυπλοκότητα ενός αλγορίθμου δίνεται από το πολυώνυμο $f(n)=2n^3+5n^2-4n + 3$. Ο πιο σημαντικός όρος του πολυωνύμου είναι η τρίτη δύναμη, αρκεί να σκεφτούμε ότι καθώς το χ αυξάνεται (και τείνει στο άπειρο) ο όρος αυτός έχει μεγαλύτερο «ειδικό βάρος» και συνεχώς καθίσταται σημαντικότερος, ενώ η σημασία των υπολοίπων όρων σταδιακά μειώνεται. Επιπλέον, αγνοείται ο συντελεστής 2 της τρίτης δύναμης και έτσι προκύπτει ότι $g(n) = n^3$. Επομένως, τελικά η πολυπλοκότητα του αλγορίθμου είναι $O(n^3)$. Πλέον, η έκφραση αυτή εκφράζει την ποιοτική και όχι την ποσοτική συμπεριφορά του αλγορίθμου.

Παραδείγματα χρονικής πολυπλοκότητας

- O(1). Κάθε εντολή του προγράμματος εκτελείται μία φορά ή το πολύ μερικές μόνο φορές. Στην περίπτωση αυτή λέγεται ότι ο αλγόριθμος είναι σταθερής πολυπλοκότητας.
- O(logn). Ο αλγόριθμος είναι λογαριθμικής πολυπλοκότητας. Ας σημειωθεί ότι με "log" θα συμβολίζεται ο δυαδικός λογάριθμος, ενώ με "ln" θα συμβολίζεται ο φυσικός λογάριθμος. Συνήθως, οι λογάριθμοι που χρησιμοποιούνται στο βιβλίο αυτό είναι δυαδικοί.
- O(n). Η πολυπλοκότητα λέγεται γραμμική. Αυτή είναι η καλύτερη επίδοση για έναν αλγόριθμο που πρέπει να εξετάσει ή να δώσει στην έξοδο n στοιχεία.
- O(n logn). Διαβάζεται όπως ακριβώς γράφεται (n logn), δηλαδή χωρίς να χρησιμοποιείται κάποιο επίθετο (όπως για παράδειγμα γραμμολογαριθμική). Στην κατηγορία αυτή ανήκει μία πολύ σπουδαία οικογένεια αλγορίθμων ταξινόμησης.
- $O(n^2)$. Τετραγωνική πολυπλοκότητα. Πρέπει να χρησιμοποιείται μόνο για προβλήματα μικρού μεγέθους.
- $O(n^3)$. Κυβική πολυπλοκότητα. Και αυτοί οι αλγόριθμοι πρέπει να χρησιμοποιούνται μόνο για προβλήματα μικρού μεγέθους.
- $O(2^n)$. Σπάνια στην πράξη χρησιμοποιούνται αλγόριθμοι εκθετικής πολυπλοκότητας.

Πιν. 5.3. Χρόνοι εκτέλεσης αλγορίθμων ανάλογα με την πολυπλοκότητα και το μέγεθος			
Πολυπλοκότητα αλγορίθμου	Μέγεθος προβλήματος		
	n20	n40	n=60
O(n)	0.00002 δεύτερα	0.00004 δεύτερα	0.00006 δεύτερα
$O(n^2)$	0.00004 δεύτερα	0.0016 δεύτερα	0.0036 δεύτερα
$O(n^3)$	0.008 δεύτερα	0.064 δεύτερα	216 δεύτερα
$O(n^n)$	1.0 δεύτερο	2.7 ημέρες	366 αιώνες
O(n!)	771 αιώνες	$3\ 10^{32}$ αιώνες	$3\ 10^{66}$ αιώνες

Παράδειγμα: Σειριακή αναζήτηση

Ας εξετάσουμε αρχικά, πως προκύπτει η πολυπλοκότητα της επιτυχούς αναζήτησης σε μη ταξινομημένο πίνακα που αποτελείται από n στοιχεία. Αν αναζητάται το πρώτο στοιχείο, τότε η επιτυχής αναζήτηση θα κοστίσει μία σύγκριση, ενώ αν αναζητάται το δεύτερο στοιχείο, τότε το κόστος είναι δύο συγκρίσεις. Με την ίδια λογική, αν αναζητάται το η-οστό στοιχείο, τότε θα εκτελεσθούν n συγκρίσεις κλειδιών μέχρι την περάτωση του αλγορίθμου. Έτσι κατά μέσο όρο, ο απαιτούμενος αριθμός συγκρίσεων κλειδιών για την επιτυχή αναζήτηση είναι:

$$E = \frac{(1+2+\ldots+n)}{n} = \frac{n(n+1)}{2n} = \frac{n+1}{2}$$

Από τη σχέση αυτή, εύκολα καταλήγουμε στο συμπέρασμα ότι η επιτυχής αναζήτηση έχει πολυπλοκότητα της τάξης O(n), με την απαραίτητη προϋπόθεση ότι τα κλειδιά αναζητώνται ισοπίθανα.

Η πολυπλοκότητα της ανεπιτυχούς αναζήτησης είναι επίσης τάξης O(n). Αυτό προκύπτει με βάση την απλή σκέψη ότι, όταν το αναζητούμενο κλειδί δεν υπάρχει στον πίνακα, τότε η αναζήτηση καταλήγει να εξετάσει ένα προς ένα όλα τα κλειδιά μέχρι το τέλος του πίνακα. Αν ο πίνακας είναι ταξινομημένος, τότε η διαδικασία της επιτυχούς και της ανεπιτυχούς αναζήτησης μπορεί να βελτιωθεί, ωστόσο και πάλι η πολυπλοκότητα θα είναι γραμμικής τάξης.

* Σύγκριση με δυαδική μέθοδο

Παράδειγμα 2: Ταξινόμηση φυσαλίδας

Το πιο βασικό κριτήριο της επίδοσης μίας μεθόδου ταξινόμησης είναι ο αριθμός C, που μετρά τις απαιτούμενες *συγκρίσεις κλειδιών* (key comparisons), που εκτελούνται μέχρι να τελειώσει η ταξινόμηση. Ένα άλλο κριτήριο είναι ο αριθμός M που μετρά τις *μετακινήσεις* (moves) των στοιχείων. Οι αριθμοί C και M είναι συναρτήσεις του αριθμού n των στοιχείων, που πρέπει να ταξινομηθούν.

Με βάση τον αλγόριθμο όπως περιγράφεται στην παράγραφο, εύκολα προκύπτει ότι ο αριθμός των συγκρίσεων στην καλύτερη, τη μέση και τη χειρότερη περίπτωση είναι ο ίδιος. Ο αριθμός αυτός είναι:

$$C = 1 + 2 + \ldots + (n-1)$$

Με βάση τις ιδιότητες της αριθμητικής προόδου προκύπτει ότι

$$C = \frac{n(n-1)}{2}$$

που τελικά σημαίνει, ότι η πολυπλοκότητα της ταξινόμησης αυτής είναι $O(n^2)$.

Πιν. 5.4. Πολυπλοκότητες μερικών αλγορίθμων	
Αλγόριθμος	**Πολυπλοκότητα**
Ώθηση-απώθηση σε στοίβα	O(1)
Εισαγωγή-εξαγωγή σε ουρά	O(1)
Fibonacci επαναληπτική	O(n)
Ταξινόμηση ευθείας ανταλλαγής	$O(n^2)$
Σειριακή αναζήτηση	O(n)
Δυαδική αναζήτηση	O(logn)

Ασκήσεις (Τετράδιο μαθητή)

ΔΤ1. Να συζητηθεί η επίδοση των παρακάτω κομματιών αλγορίθμων και να καταγραφεί για κάθε περίπτωση και η αντίστοιχη πολυπλοκότητα. Για να βρείτε την πολυπλοκότητα, θα πρέπει να μετρήσετε τον αριθμό των πράξεων στη χειρότερη περίπτωση:

1.
```
Για i από 1 μέχρι n -1 με βήμα 2
        a ← 2*i
Τέλος_επανάληψης
```
n/2 πράξεις **(O(n))**

2.
```
Για i από 1 μέχρι n
        Για j από 1 μέχρι n
                a ← 2*i +j
        Τέλος_επανάληψης
Τέλος_επανάληψης
```
n^2 πράξεις **($O(n^2)$)**

3.
```
Για i από 1 μέχρι n με βήμα 2
        Για j από 1 μέχρι n
                a ← 2*i +j
        Τέλος_επανάληψης
Τέλος_επανάληψης
```
$n^2/2$ πράξεις **($O(n^2)$)**

Ασκήσεις (Τετράδιο μαθητή) - 2

ΔΤ3. Να σχολιασθεί και να παρακολουθήσετε βήμα-βήμα τον ακόλουθο επαναληπτικό αλγόριθμο υπολογισμού των αριθμών Fibonacci. Ποια είναι η πολυπλοκότητα του ακόλουθου αλγορίθμου;

Αλγόριθμος Fibonacci
i ← 1
j ← 0
k ← 0
l ← 1
Επανάλαβε όσο n > 0
    Αν n MOD 2 = 1 τότε m ← j*l
    j ← i*l+j*k+m
    i ← i*k+m
    m ← **Ρίζα(l) !Ρίζα είναι η συνάρτηση τετραγωνικής ρίζας'**
    h ← 2*k*l+m
    k ← **Ρίζα(k)+m**
    n ← n **DIV 2**
Τέλος_επανάληψης
Fib ← j
Αποτελέσματα Fib

Τέλος Fibonacci

logn πράξεις (O(logn))

Ασκήσεις (Τετράδιο μαθητή) - 3

ΔΣ1. Να βρεις την πολυπλοκότητα των παρακάτω κομματιών αλγορίθμων:

1.
Όσο i<100 επανάλαβε

a ← 2*i **O(n)**

Τέλος_επανάληψης

2.
Για i από 1 μέχρι n -1

a ← 2*I **O(n)**

Τέλος_επανάληψης

3.
Για i από 1 μέχρι n -1 με βήμα 3

a ← 2*I **O(n)**

Τέλος_επανάληψης

10.6 Εμβέλεια μεταβλητών – σταθερών

- Απεριόριστη εμβέλεια
- Περιορισμένη εμβέλεια (ΓΛΩΣΣΑ – πλήρης αυτονομία υποπρογραμμάτων)
- Μερικώς περιορισμένη εμβέλεια

(μόνο θεωρία)