Κεφάλαιο 3

Δομές Δεδομένων και Αλγόριθμοι

3.1 Γενικός διδακτικός σκοπός

Ο γενικός σκοπός του κεφαλαίου είναι να γίνει κατανοητό στο μαθητή ότι τα δεδομένα αποθηκεύονται στη μνήμη (κύρια ή δευτερεύουσα) του υπολογιστή με κάποια συγκεκριμένη μέθοδο, δηλαδή με τη βοήθεια κάποιας δομής. Επίσης, κάθε δομή δεδομένων συνοδεύεται απαραιτήτως από κάποιες συγκεκριμένες λειτουργίες (δηλαδή πράξεις).

3.2 Ειδικοί διδακτικοί σκοποί

Μετά την ολοκλήρωση του παρόντος κεφαλαίου, οι μαθητές θα πρέπει να είναι σε θέση:

- να διακρίνουν τις διάφορες δομές δεδομένων, όπως πίνακας, στοίβα, ουρά κοκ.
- να αντιλαμβάνονται ποιές είναι οι πράξεις που συνοδεύουν κάθε δομή
- να μπορούν να αναπτύξουν αλγοριθμικά απλές αναδρομικές συναρτήσεις
- να εφαρμόσουν μεθόδους ταξινόμησης σε τυχαία σύνολα δεδομένων
- να αντιληφθούν την ιδιαιτερότητα των δυναμικών δομών της λίστας και του δένδρου.

3.3 Οδηγίες – επισημάνσεις

Είναι απαραίτητο να εξηγηθεί ότι δεν υπάρχει μία αδιαφιλονίκητη δομή που να χρησιμοποιείται σε κάθε εφαρμογή, αλλά η δομή επιλέγεται με βάση τις λειτουργίες που πρέπει να εκτελούνται. Θα πρέπει να δοθεί ιδιαίτερη έμφαση στη διάκριση μεταξύ των διαφόρων αλγορίθμων ταξινόμησης, επειδή σε επόμενο κεφάλαιο θα γίνει αναφορά στην ανάλυση της πολυπλοκότητάς τους. Επίσης προσοχή χρειάζεται στην παράγραφο των λιστών και των δένδρων που στηρίζονται στην έννοια της δυναμικότητας και υλοποιούνται με τη βοήθεια δεικτών.

3.4 Προγραμματισμός μαθημάτων κεφαλαίου

Προτεινόμενος αριθμός μαθημάτων

τρία (3) δίωρα μαθήματα

Σχέδιο 1ου μαθήματος

Διδακτικοί στόχοι

- Εμβάθυνση στις έννοιες «δεδομένο», «δομή δεδομένων» και «λειτουργία» επί μίας δομής
- Επεξήγηση της διαλεκτικής σχέσης μεταξύ δομών και αλγορίθμων
- Κατηγοριοποίηση των δομών και έμφαση στη δομή των πινάκων

Χώρος υλοποίησης μαθήματος

τάξη

Προτεινόμενα υλικά και εποπτικά μέσα διδασκαλίας

πίνακας, διαφάνειες

Περιεχόμενα θεωρητικής παρουσίασης

- Κάλυψη παραγράφων 3.1 μέχρι και 3.3
- Διαφορά μεταξύ δεδομένου και πληροφορίας
- Ορισμός της Πληροφορικής επιστήμης σε συνάρτηση με την έννοια των δεδομένων
- Ορισμός δομής δεδομένων
- Απαρίθμηση λειτουργιών επί των δομών
- Συσχέτιση δομών και αλγορίθμων
- Κατηγοριοποίηση δομών
- Χαρακτηριστικά δομής πίνακα

Περιεχόμενα πρακτικής εφαρμογής

Εφαρμογές, παραδείγματα από το βιβλίο του μαθητή

Θα πρέπει να διδαχθούν όλα τα παραδείγματα των παραγράφων 3.1 μέχρι 3.3 από το βιβλίο του μαθητή.

Δραστηριότητες από το τετράδιο του μαθητή

Θα πρέπει να διδαχθούν δύο από τα παραδείγματα 1, 2, 3, 4 από το τετράδιο του μαθητή. Να δοθεί στους μαθητές προς λύση η δραστηριότητα ΔΤ1 στην τάξη και μία από τις δραστηριότητες ΔΣ1 ή ΔΣ2 για το σπίτι.

Τεστ αξιολόγησης επίδοσης

Συμπληρώστε με σωστό ή λάθος

1. Αποτελεί δεδομένο ότι το ύψος ενός ατόμου είναι 1,90. Πληροφορία είναι ότι το άτομο αυτό είναι ψηλό.
2. Κάθε δομή μπορεί να χρησιμοποιηθεί σε οποιοδήποτε πρόβλημα ή εφαρμογή
3. Δυναμικές είναι οι δομές που αποθηκεύονται σε συνεχόμενες θέσεις μνήμης
4. Ένας πίνακας έχει σταθερό μέγεθος αλλά μεταβαλλόμενο περιεχόμενο
5. Ένας πίνακας μπορεί να αποθηκεύσει και ακεραίους αλλά και πραγματικούς αρθμούς

Απαντήσεις τεστ αξιολόγησης επίδοσης

Συμπληρώστε με σωστό ή λάθος

1: σωστό	2: λάθος	3: λάθος
4: σωστό	5: λάθος	

Σχέδιο 2ου μαθήματος

Διδακτικοί στόχοι

- Εμβάθυνση στη δομή της στοίβας και των λειτουργιών της
- Εμβάθυνση στη δομή της ουράς και των λειτουργιών της
- Εξέταση της σειριακής αναζήτησης

Χώρος υλοποίησης μαθήματος

τάξη

Προτεινόμενα υλικά και εποπτικά μέσα διδασκαλίας

πίνακας, διαφάνειες

Περιεχόμενα θεωρητικής παρουσίασης

- Κάλυψη παραγράφων 3.4 μέχρι και 3.6
- Επεξήγηση της δομής της στοίβας και της λογικής LIFO
- Συσχέτιση λογικής ουράς και στοίβας
- Αναφορά στη σειριακή αναζήτηση εισάγοντας ακροθιγώς την έννοια της αποτελεσματικότητας

Περιεχόμενα πρακτικής εφαρμογής

- Εφαρμογές, παραδείγματα από το βιβλίο του μαθητή

Θα πρέπει να διδαχθούν όλα τα παραδείγματα των αντίστοιχων παραγράφων από το βιβλίο του μαθητή.

- Δραστηριότητες από το τετράδιο του μαθητή

Θα πρέπει να δοθεί στους μαθητές προς λύση η δραστηριότητα ΔΤ6 στην τάξη και η δραστηριότητα ΔΣ4 στο σπίτι.

Τεστ αξιολόγησης επίδοσης

Συμπληρώστε με σωστό ή λάθος

1. Μία ουρά διατηρεί τα δεδομένα ταξινομημένα ως προς τη σειρά άφιξής τους.
2. Η υλοποίηση της ουράς χρησιμοποιεί μία μόνο μεταβλητή-δείκτη για τη διαχείριση των εισαγωγών/διαγραφών, όπως και η περίπτωση της στοίβας.
3. Όταν ψάχνουμε σε ένα τηλεφωνικό κατάλογο χρησιμοποιούμε τη σειριακή μέθοδο αναζήτησης.

Άσκηση

Έστω ότι σε μία στοίβα εισάγονται τρία στοιχεία με τη σειρά: πρώτα το Α, μετά το Β και τέλος το Γ, τα οποία αργότερα εξάγονται. Όμως, οι εξαγωγές αυτές μπορούν να συμβούν οποτεδήποτε μεταξύ ή μετά τις εισαγωγές. Για παράδειγμα, μπορεί να συμβεί η αλληλουχία: εισαγωγή Α, εξαγωγή Α, εισαγωγή Β, εξαγωγή Β, εισαγωγή Γ, εξαγωγή Γ. Έτσι στην έξοδο θα λάβουμε με τη σειρά τη διάταξη των στοιχείων Α, Β, Γ. Αν όμως, εκτελεσθούν με τη σειρά οι πράξεις εισαγωγή Α, εξαγωγή Α, εισαγωγή Β, εισαγωγή Γ, εξαγωγή Γ, εξαγωγή Β, τότε στην έξοδο θα λάβουμε με τη σειρά τη διάταξη των στοιχείων Α, Γ, Β. Η ερώτηση είναι αν θεωρήσουμε όλους τους τρόπους αλληλοδιαδοχής πράξεων εισαγωγής/εξαγωγής, πόσες και ποιές είναι οι δυνατές διατάξεις των στοιχείων Α, Β, Γ στην έξοδο;

Απαντήσεις τεστ αξιολόγησης επίδοσης

Συμπληρώστε με σωστό ή λάθος

1: σωστό	2: λάθος	3: λάθος

Άσκηση

Ο αριθμός των διατάξεων 3 στοιχείων είναι 3!=6. Οι διατάξεις αυτές είναι ΑΒΓ, ΑΓΒ, ΒΑΓ, ΒΓΑ, ΓΑΒ και ΓΒΑ. Από την εκφώνηση είναι γνωστό ότι οι δύο πρώτες διατάξεις είναι δυνατόν να παραχθούν. Υπάρχουν άλλες τρεις διατάξεις που μπορούν να παραχθούν στην έξοδο

- εισαγωγή Α, εισαγωγή Β, εξαγωγή Β, εξαγωγή Α, εισαγωγή Γ, εξαγωγή Γ, οπότε στην έξοδο λαμβάνουμε τη διάταξη ΒΑΓ
- εισαγωγή Α, εισαγωγή Β, εξαγωγή Β, εισαγωγή Γ, εξαγωγή Γ, εξαγωγή Α, οπότε λαμβάνουμε τη διάταξη ΒΓΑ
- εισαγωγή Α, εισαγωγή Β, εισαγωγή Γ, εξαγωγή Γ, εξαγωγή Β, εξαγωγή Α, οπότε λαμβάνουμε τη διάταξη ΓΒΑ

αλλά δεν υπάρχει αλληλουχία εισαγωγών/εξαγωγών που να οδηγεί στη διάταξη ΓΑΒ.

Σχέδιο 3ου μαθήματος

Διδακτικοί στόχοι

- Εμβάθυνση στην έννοια της αναδρομής
- Παρουσίαση των δυναμικών δομών της λίστας και του δένδρου

Χώρος υλοποίησης μαθήματος

τάξη

Προτεινόμενα υλικά και εποπτικά μέσα διδασκαλίας

πίνακας, διαφάνειες

Περιεχόμενα θεωρητικής παρουσίασης

- Κάλυψη παραγράφων 3.7 μέχρι και 3.9
- Επεξήγηση της ταξινόμησης ως μίας βασικής λειτουργίας, που μπορεί να εφαρμοσθεί σε πίνακες
- Ανάπτυξη των μειονεκτημάτων και των πλεονεκτημάτων της αναδρομής σε σχέση με το χρόνο εκτέλεσης του προγράμματος και το χρόνο ανάπτυξης από τον προαγραμματιστή.

- Σύντομη παρουσίαση των δυναμικών δομών που απαιτούν χρήση μεταβλητών δεικτών που προσφέρονται από τις σύγχρονες γλώσσες προγραμματισμού.

Περιεχόμενα πρακτικής εφαρμογής

- Εφαρμογές, παραδείγματα από το βιβλίο του μαθητή

Θα πρέπει να διδαχθούν όλα τα παραδείγματα των αντίστοιχων παραγράφων από το βιβλίο του μαθητή

- Δραστηριότητες από το τετράδιο του μαθητή

Θα πρέπει να δοθεί στους μαθητές προς λύση η δραστηριότητα ΔΤ6 στην τάξη και μία από τις δραστηριότητες ΔΣ3 και ΔΣ4 για το σπίτι.

Τεστ αξιολόγησης επίδοσης

Συμπληρώστε με σωστό ή λάθος

1. Η ταξινόμηση είναι χρήσιμη διαδικασία γιατί έτσι εκτελείται γρηγορότερα η αναζήτηση.
2. Η ταξινόμηση ευθείας ανταλλαγής είναι πολύ αποτελεσματική σε πίνακες που είναι ταξινομημένοι κατά την αντίστροφη φορά σε σχέση με την επιθυμητή.
3. Η ταξινόμηση ευθείας ανταλλαγής είναι πολύ αποτελεσματική αν ο πίνακας περιέχει ίσα κλειδιά.
4. Όταν δεν είναι ιδιαίτερα κρίσιμος ο χρόνος απόκρισης μίας εφαρμογής, τότε μπορεί ο αντίστοιχος αλγόριθμος να είναι αναδρομικός.
5. Η αναδρομή στηρίζεται στη δομή της ουράς.
6. Σε ένα πίνακα δεν υπάρχει τρόπος διάκρισης της φυσικής σειράς αποθήκευσης από τη λογική αλληλουχία των στοιχείων.
7. Βασική λειτουργία σε μία δομή λίστας είναι η προσπέλαση σε τυχαίο κόμβο της δομής.

8. Η ρίζα ενός δένδρου δεν είναι παιδί κάποιου άλλου κόμβου.

Απαντήσεις τεστ αξιολόγησης επίδοσης

Συμπληρώστε με σωστό ή λάθος

1: σωστό	2: λάθος	3: σωστό
4: σωστό	5: λάθος	6: λάθος (χρήση δεικτών)
7: λάθος	8: σωστό	

3.5 Προβληματισμοί και θέματα προς συζήτηση

Θέματα προβληματισμού και συζήτησης μπορούν να προκύψουν από όλες τις δραστηριότητες που προτείνονται στο συγκεκριμένο κεφάλαιο. Συγκεκριμένα μπορούν να προταθούν συζητήσεις σχετικά με:

- Υπάρχει κάποια δομή μεταξύ των δομών που εξετάσθηκαν, η οποία να βρίσκει εύκολα το στοιχείο με τη μικρότερη τιμή; Η απάντηση είναι αρνητική. Για μία τέτοια περίπτωση υπάρχει μία ιδιαίτερη δενδρική δομή, που ονομάζεται σωρός (heap) και έχει αρκετά πολύπλοκους αλγορίθμους εισαγωγής και διαγραφής.
- Οι αλγόριθμοι αναζήτησης που εξετάσθηκαν είναι αρκετά αποτελεσματικοί ή μπορούν να σχεδιασθούν ακόμη καλύτεροι; Η απάντηση σχετίζεται με το περιεχόμενο του επομένου κεφαλαίου.

3.6 Προτεινόμενες πηγές πληροφόρησης

Όλη η προτεινόμενη βιβλιογραφία του κεφαλαίου, όπως καταγράφεται στο βιβλίο του μαθητή.

3.7 Απαντήσεις ερωτήσεων βιβλίου μαθητή

1: Δες παράγραφο 3.1

2: Δες παράγραφο 3.1

3: Δες παράγραφο 3.2

4: Δες παράγραφο 3.2

5: Δες παράγραφο 3.2

6: Δες παράγραφο 3.2

7: Δες παράγραφο 3.3

8: Δες παράγραφο 3.4

9: Δες παράγραφο 3.4

10: Δες παράγραφο 3.5

11: Δες παράγραφο 3.5

12: Δες παράγραφο 3.8

13: Ακολουθεί ένας αναδρομικός αλγόριθμος για τον υπολογισμό του a^b, όπου το a είναι πραγματικός και το b ακέραιος.

```
Αλγόριθμος Δύναμη
Δεδομένα \\ a, b \\
Αν b = 0 τότε
    power ← 1
αλλιώς
    Αν (b MOD 2) = 0 τότε
        power ← Δύναμη(a*a, b DIV 2)
    αλλιώς
        power ← Δύναμη(a*a,b DIV 2)*a
    Τέλος_Αν
Τέλος_Αν
Αποτελέσματα \\ power \\
Τέλος Δύναμη
```

14: Δες παράγραφο 3.6

15: Δες παράγραφο 3.6

16: Δες παράγραφο 3.7

17: Δες παράγραφο 3.7

3.8 Απαντήσεις δραστηριοτήτων τετραδίου μαθητή

➤ *Στην τάξη*

ΔΤ 1

Θεωρούμε τον πίνακα table με ακεραίους που αντιστοιχούν στις ηλικίες των παιδιών. Οι μεταβλητές min_age και max_age αρχικοποιούνται με την τιμή της πρώτης θέσης του πίνακα, ενώ οι μεταβλητές min και max χρησιμοποιούνται για τη εύρεση της ταυτότητας των δύο παιδιών αντίστοιχα.

```
Αλγόριθμος Κατασκήνωση
Δεδομένα // table//
min_age ← table[1]
max_age ← table[1]
min ← 1
max ← 1
Για i από 2 μέχρι 300
    Αν table[i]<min_age τότε
        min_age ← table[i]
        min ← i
    Τέλος_αν
    Αν table[i]>max_age τότε
        max_age ← table[i]
        max ← i
    Τέλος_αν
Τέλος_επανάληψης
Εκτύπωσε min, min_age, max, max_age
Τέλος Κατασκήνωση
```

ΔΤ 2

Η εκφώνηση ουσιαστικά αναφέρεται στην παραλλαγή της ταξινόμησης φυσαλίδας που αντιλαμβάνεται πότε ο πίνακας έχει ουσιαστικά ταξινομηθεί και αποφεύγει τα περιττά περάσματα (που σε συνολικό αριθμό είναι n-1). Αυτό επιτυγχάνεται με τη βοήθεια μίας λογικής μεταβλητής, (μίας σημαίας flag), που πριν κάθε πέρασμα αρχικοποιείται ως ψευδής και αλλάζει ως αληθής αν σε κάποιο πέρα-

σμα γίνει έστω και μία ανταλλαγή. Έτσι, αν σε κάποιο πέρασμα δεν εκτελεσθεί καμία ανταλλαγή, τότε η σημαία παραμένει ψευδής και αυτομάτως τελειώνει ο αλγόριθμος.

```
Αλγόριθμος Φυσσαλίδα2
Δεδομένα // table //
Αρχή_επανάληψης
    flag ← Ψευδής
    Για i από 1 μέχρι n-1
        Αν table[i+1] < table[i] τότε
            αντιμετάθεσε(table[i+1], table[i])
            flag ← Αληθής
        Τέλος_αν
    Τέλος_επανάληψης
Μέχρις_ότου flag=Ψευδής
Αποτελέσματα // table //
Τέλος Φυσσαλίδα2
```

ΔΤ 3

Στη συνέχεια δίνονται οι αλγόριθμοι ώθησης (push) και απώθησης (pop) από στοίβα. Χρησιμοποιείται μία λογική μεταβλητή, η σημαία done, που δηλώνει την επιτυχή εκτέλεση της διαδικασίας. Επίσης, η μεταβλητή top δηλώνει την επάνω θέση της στοίβας που είναι κατειλημμένη από κάποιο στοιχείο. Τα δεδομένα είναι αποθηκευμένα σε ένα μονοδιάστατο πίνακα, που ονομάζεται stack και έχει μέγεθος size. Η μεταβλητή item χρησιμεύει για την αποθήκευση του στοιχείου που εισάγεται ή εξάγεται.

```
Αλγόριθμος Ωθηση
Δεδομένα // top, item //
Αν top<size τότε
    top ← top+1
    stack[top] ← item
    done ← Αληθής
αλλιώς
    done ← Ψευδής
Τέλος_αν
Αποτελέσματα // top, done //
Τέλος Ωθηση
```

```
Αλγόριθμος Απώθηση
Δεδομένα // top //
Αν top<=1 τότε
    item ← stack[top]
    top ← top-1
    done ← Αληθής
αλλιώς
    done ← Ψευδής
Τέλος_αν
Αποτελέσματα // item, top, done //
Τέλος Απώθηση
```

Στον προηγούμενο αλγόριθμο ελέγχεται η συνθήκη «top<=1» ώστε να διαπιστωθεί αν η στοίβα έχει τουλάχιστον ένα στοιχείο. Προφανώς, αν δεν έχει τότε επιστρέφεται το μήνυμα Ψευδής.

ΔΤ 4

Οι αλγόριθμοι εισαγωγής και εξαγωγής από ουρά δίνονται στη συνέχεια. Χρησιμοποιείται μία λογική μεταβλητή, η σημαία done, που δηλώνει την επιτυχή εκτέλεση της διαδικασίας. Επίσης, οι μεταβλητές rear και front δηλώνουν τους δύο δείκτες που δείχνουν αντίστοιχα στην τελευταία θέση και στην πρώτη θέση της ουράς, που είναι ένας πίνακας queue μεγέθους size. Η μεταβλητή item χρησιμεύει για την αποθήκευση του στοιχείου που εισάγεται ή εξάγεται.

```
Αλγόριθμος Εισαγωγή_σε_Ουρά
Δεδομένα // rear, item //
Αν rear < size τότε
    rear ← rear+1
    queue[rear] ← item
    done ← Αληθής
αλλιώς
    done ← Ψευδής
Τέλος_αν
Αποτελέσματα // rear, done //
Τέλος Εισαγωγή_σε_Ουρά

Αλγόριθμος Εξαγωγή_από_Ουρά
Δεδομένα // rear, item //
```

```
Αν rear <= front τότε
    front ← front+1
    item ← queue[front]
    done ← Αληθής
αλλιώς
    done ← Ψευδής
Τέλος_αν
Αποτελέσματα // item, rear, done //
Τέλος Εξαγωγή_από_Ουρά
```

Στον προηγούμενο αλγόριθμο η συνθήκη «rear<=front» ελέγχει αν η ουρά περιέχει στοιχεία. Η ισότητα ισχύει στην περίπτωση που η ουρά περιέχει μόνο ένα στοιχείο.

ΔΤ 5

Η άσκηση αυτή είναι σχετικά σύνθετη επειδή αποτελείται από δύο σκέλη. Σε πρώτη φάση πρέπει να ταξινομηθούν αλφαβητικά τα επίθετα των μαθητών και κατόπιν πρέπει τα επίθετα αυτά να εισαχθούν σε μία ουρά. Η άσκηση επιλύεται με τη βοήθεια του αλγορίθμου ταξινόμησης της παραγράφου 3.7 ή του αλγορίθμου της προηγούμενης ΔΤ2.

ΔΤ 6

Αποτελεί απλή παραλλαγή της μεθόδου ταξινόμησης ευθείας ανταλλαγής, που περιγράφεται στην παράγραφο 3.7. Η διαφορά έγκειται στο ότι στον εξωτερικό βρόχο δεν πρέπει να γίνουν n-1 αλλά 10 επαναλήψεις, ώστε να απομονωθούν στην κορυφή του πίνακα οι 10 μικρότερες ζητούμενες τιμές.

➤ *Στο σπίτι*

ΔΣ 1

Θα θεωρήσουμε ένα δισδιάστατο πίνακα currency μεγέθους 5x2, όπου κάθε γραμμή του πίνακα αντιστοιχεί σε ένα νόμισμα (όπως τα νομίσματα αναφέρονται στην εκφώνηση), ενώ οι δύο στήλες του πίνακα αντιστοιχούν στην τιμή αγοράς ή πώλησης κάθε νομίσματος. Για λόγους ευκολίας (αποφυγή χρήσης συμβολοσειρών) χρησιμοποιούνται οι ακέραιες μεταβλητές nomisma1 και nomisma2, που

λαμβάνουν τιμές από 1 μέχρι 6. Οι τιμές αυτές αντιστοιχούν στα νομίσματα της εκφώνησης με έκτο νόμισμα τη δραχμή. Επίσης, η πραγματική μεταβλητή poso1 αντιστοιχεί στο ποσό των νομισμάτων nomisma1 που δίνει ο πελάτης, και η μεταβλητή poso2 στο ποσό των νομισμάτων nomisma2 που προκύπτουν από τη μετατροπή.

```
Αλγόριθμος Μετατροπές_νομισμάτων
Δεδομένα // currency //
Διάβασε nomisma1,nomisma2,poso1
Αν nomisma1=6 τότε
    poso2 ← poso1 * currency[nomisma2,2]
αλλιώς
    poso2 ← poso1 * currency[nomisma1,1]
Τέλος_αν
Εκτύπωσε poso2
Τέλος Μετατροπές_νομισμάτων
```

ΔΣ 2

Θα θεωρήσουμε ένα δισδιάστατο πίνακα scores μεγέθους 5x5, όπου κάθε γραμμή του πίνακα αντιστοιχεί σε ένα παίκτη, ενώ κάθε στήλη του πίνακα αντιστοιχεί σε έναν αγώνα. Τους συνολικούς πόντους που πέτυχαν οι παίκτες, τους αποθηκεύουμε σε ένα μονοδιάστατο πίνακα 5 θέσεων που ονομάζεται sum. Η μεταβλητή first δηλώνει τον αθλητή που επέτυχε συνολικά τους περισσότερους πόντους (max).

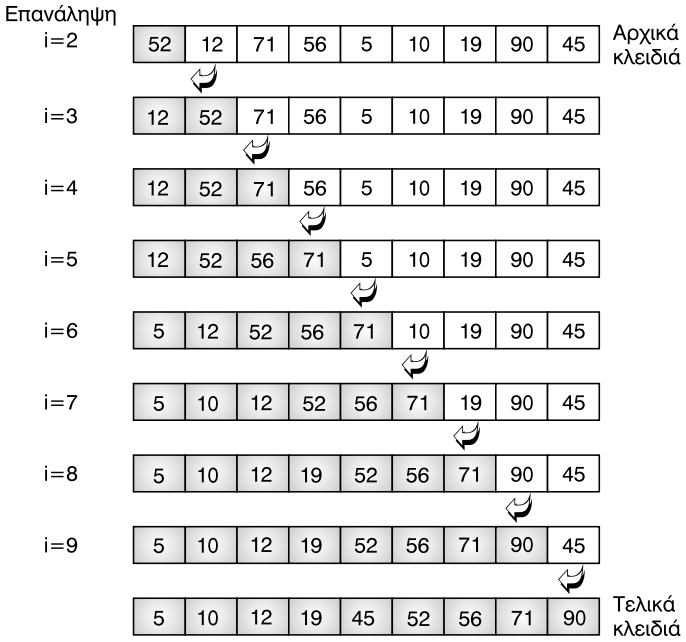
```
Αλγόριθμος Στατιστικά
Δεδομένα // sum, scores //
Για i από 1 μέχρι 5
    sum[i] ← 0
Τέλος_επανάληψης
Για i από 1 μέχρι 5
    Για j από 1 μέχρι 5
        sum[i] ← sum[i]+ scores[i,j]
    Τέλος_επανάληψης
Τέλος_επανάληψης
max ← sum[1]
first ← 1
Για i από 2 μέχρι 5
```

```
    Αν scores[i] > max τότε
        max ← scores[i]
        first ← i
    Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα // first //
Τέλος Στατιστικά
```

ΔΣ 3

Ουσιαστικά πρόκειται για τον αλγόριθμο ταξινόμησης ευθείας εισαγωγής straight insertion sort). Ο αλγόριθμος αυτός είναι ιδανικός για περιπτώσεις δεδομένων που είναι «περίπου» ταξινομημένα και χρησιμοποιείται σε πολλά υβριδικά σχήματα. Σύμφωνα με τον αλγόριθμο αυτό τα στοιχεία διακρίνονται σχηματικά σε μία *ακολουθία προορισμού* (destination sequence) table[1], table[2], ..., table[i-1] και σε μία *ακολουθία πηγής* (source sequence) table[i], ..., table[n]. Αρχικά η ακολουθία προορισμού αποτελείται από ένα στοιχείο, το πρώτο, και σταδιακά μεγαλώνει κατά ένα. Αυτό επιτυγχάνεται θεωρώντας το πρώτο στοιχείο της ακολουθίας πηγής και παρεμβάλοντάς το στην κατάλληλη θέση μεταξύ των στοιχείων της ακολουθίας προορισμού εκτελώντας διαδοχικές συγκρίσεις από τα δεξιά προς τα αριστερά των με τα στοιχεία της ακολουθίας προορισμού. Ο σχετικός αλγόριθμος δίνεται στη συνέχεια.

```
Αλγόριθμος Ευθεία_Εισαγωγή
Δεδομένα \\ table \\
Για i από 2 μέχρι n
    temp ← table[i]
    table[0] ← temp
    j ← i-1
    Οσο temp<table[j] επανάλαβε
        table[j+1] ← table[j]
        j ← j-1
    Τέλος_επανάληψης
    table[j+1] ← temp
Τέλος_επανάληψης
Αποτελέσματα \\ table \\
Τέλος Ευθεία_Εισαγωγή
```



Σχήμα 3.1. *Ταξινόμηση ευθείας εισαγωγής.*

Παράδειγμα. Έστω ότι ο αρχικός πίνακας αποτελείται από τα εννέα κλειδιά 52, 12, 71, 56, 5, 10, 19, 90 και 45. Στο προηγούμενο σχήμα παρουσιάζεται η διαδικασία ταξινόμησης θεωρώντας τον πίνακα αυτό. Κάθε φορά η ακολουθία προορισμού εμφανίζεται με σκίαση, ενώ το πρώτο στοιχείο της ακολουθίας πηγής αναδεικνύεται από το αντίστοιχο βέλος. Το στοιχείο αυτό λαμβάνει την κατάλληλη θέση μέσα στην ακολουθία προορισμού "σπρώχνοντας" μερικά στοιχεία προς τα δεξιά. Η εύρεση της κατάλληλης θέσης γίνεται εύκολα με διαδοχικές συγκρίσεις και μετακινήσεις. Λόγου χάριν, στην πέμπτη σειρά το στοιχείο 10 συγκρίνεται διαδοχικά με τα στοιχεία 71, 56, 52, 12 και 5, οπότε γίνεται αντιληπτό ότι το 10 πρέπει να παρεμβληθεί μεταξύ των 5 και 12. Για να γίνει αυτό τα στοιχεία 12 ως και 71 μετακινούνται μία θέση προς τα δεξιά για να δημιουργηθεί μία κενή θέση για το 10.

Μερικά σχόλια σχετικά με την υλοποίηση (για παράδειγμα σε Pascal) του προηγούμενου αλγορίθμου είναι αναγκαία. Κατ'αρχήν πρέπει να θεωρηθεί ότι ο πίνακας table έχει n+1 θέσεις, με πρώτη θέση, τη θέση table[0], που χρησιμο-

ποιείται για την προσωρινή αποθήκευση του στοιχείου της ακολουθίας πηγής που πρόκειται να εισαχθεί στην ακολουθία προορισμού. Αυτό το τέχνασμα υιοθετείται για να απλουστευθεί η εντολή ελέγχου της εντολής «Επανάλαβε όσο». Με απλά λόγια, έτσι δεν γίνεται κάθε φορά ένας επιπλέον έλεγχος για να διαπιστωθεί αν οι συγκρίσεις έφθασαν μέχρι το αριστερό άκρο του πίνακα.

ΔΣ 4

Η απαιτούμενη δομή είναι ένας δισδιάστατος πίνακας, όπου κάθε γραμμή αντιστοιχεί σε ένα CD. Η πρώτη στήλη αντιστοιχεί στον τίτλο του CD, ενώ η δεύτερη αντιστοιχεί στην χρονολογία έκδοσής του. Ο πίνακας αυτός πρέπει να ταξινομηθεί ως προς τη δεύτερη στήλη, και κατόπιν με σειριακή αναζήτηση να εντοπισθούν τα σχετικά CD.

ΔΣ 5

Αποτελεί απλή επέκταση των αλγορίθμων διαχείρισης ουράς που αναφέρονται στην άσκηση ΔΤ5.

3.9 Συμπληρωματικά στοιχεία

Δίνουμε εδώ μερικά επι πλέον στοιχεία θεωρίας σχετικά με την ύλη του κεφαλαίου.

3.9.1 Ταξινόμηση ευθείας επιλογής

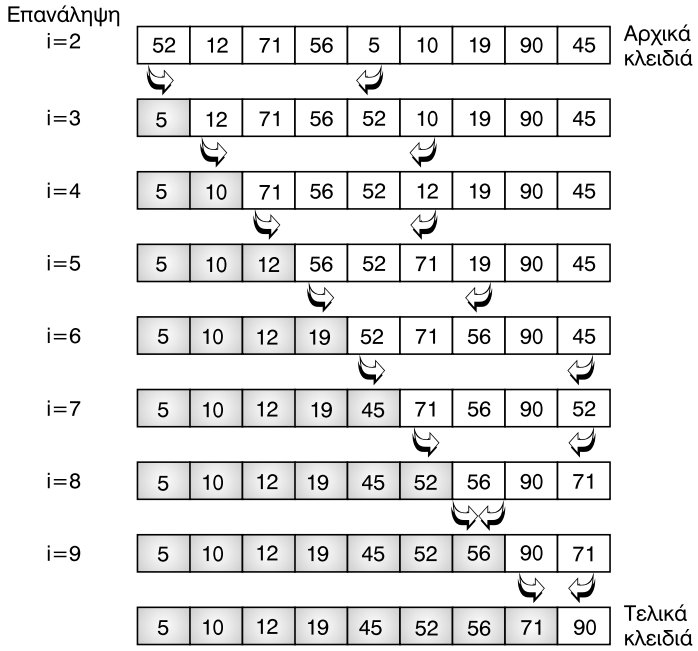
Η μέθοδος αυτή βασίζεται στις ακόλουθες δύο αρχές:

- επιλογή του στοιχείου με το ελάχιστο κλειδί και
- ανταλλαγή αυτού του στοιχείου με το πρώτο στοιχείο του πίνακα.

Αυτές οι λειτουργίες επαναλαμβάνονται για τα υπόλοιπα n-1 στοιχεία, μέχρι στο τέλος να απομείνει μόνο το μεγαλύτερο στοιχείο. Η μέθοδος που περιγράφηκε προηγουμένως ονομάζεται ***ταξινόμηση ευθείας επιλογής*** (straight selection sort) και μπορεί να θεωρηθεί ως αντίθετη της ταξινόμησης ευθείας εισαγωγής. Πιο συγκεκριμένα, η ευθεία εισαγωγή θεωρεί σε κάθε βήμα αφ' ενός το ένα και μο-

ναδικό επόμενο στοιχείο της ακολουθίας πηγής και αφ' ετέρου όλα τα στοιχεία της ακολουθίας προορισμού για να εντοπίσει το κατάλληλο σημείο εισαγωγής. Αντίθετα, η ευθεία επιλογή θεωρεί τα στοιχεία της ακολουθίας πηγής, ώστε να ανιχνεύσει το στοιχείο με το ελάχιστο κλειδί και να το τοποθετήσει ως το επόμενο στοιχείο της ακολουθίας προορισμού.

Παράδειγμα. Η μέθοδος αυτή παρουσιάζεται στο επόμενο σχήμα καθώς εφαρμόζεται στα ίδια γνωστά εννέα κλειδιά. Και πάλι, το ταξινομημένο τμήμα του πίνακα εμφανίζεται με σκίαση, ενώ με τα βέλη εμφανίζονται τα στοιχεία που ανταλλάσσονται αμοιβαία. Λόγου χάριν, στην πρώτη σειρά βρίσκουμε ότι το στοιχεί ο 5 είναι το μικρότερο και αντιμετατίθεται με το πρώτο στοιχείο του πίνακα, το 52. Έτσι προκύπτει η μορφή του πίνακα στη δεύτερη σειρά. Στη συνέχεια η διαδικασία προχωρεί με την ίδια λογική μέχρι την τελική ταξινόμηση του πίνακα. Ο αλγόριθμος για την υλοποίηση της μεθόδου ταξινόμησης ευθείας επιλογής αφήνεται στον αναγνώστη ως άσκηση.



Σχήμα 3.2. *Ταξινόμηση ευθείας επιλογής.*

3.9.2 Συγχώνευση

Η συγχώνευση (merging) είναι μία ακόμη βασική λειτουργία των δομών δεδομένων. Ειδικότερα, έχει μελετηθεί το πρόβλημα της συγχώνευσης ενός αριθμού (δύο, τριών ή και περισσοτέρων) ταξινομημένων πινάκων. Στη συνέχεια δίνεται ένας απλός αλγόριθμος συγχώνευσης δύο ταξινομημένων πινάκων σε ένα τρίτο ταξινομημένο πίνακα. Θεωρείται ότι στην είσοδο του αλγορίθμου συγχώνευσης δίνονται δύο ταξινομημένοι πίνακες x και y, μεγέθους n και m στοιχείων αντίστοιχα, ενώ στην έξοδο προκύπτει ένας τρίτος πίνακας z με n+m ταξινομημένα στοιχεία κατά την ίδια φορά. Πιο ειδικά, οι μεταβλητές i, j και k είναι δείκτες για την κίνηση μέσα στους πίνακες x, y και z. Η μέθοδος προχωρά ως εξής. Το μικρότερο στοιχείο από τους πίνακες x και y τοποθετείται στον πίνακα z με ταυτόχρονη αύξηση του αντίστοιχου δείκτη. Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου τελειώσουν τα στοιχεία του ενός πίνακα. Ύστερα τα υπόλοιπα στοιχεία του άλλου πίνακα μεταφέρονται στον πίνακα z.

```
Αλγόριθμος Συγχώνευση
Δεδομένα // x(), y(), n, m //
i ← 1
j ← 1
k ← 1
'Οσο i <=n και j <=m επανάλαβε
    Αν x[i] < y[j] τότε
        z[k] ← x[i]
        i ← i+1
    αλλιώς
        z[k] ← y[j]
        j ← j+1
    Τέλος_αν
    k ← k+1
Τέλος_επανάληψης
Αν i > n τότε
    Για t από j μέχρι m
        z[k+t-j] ← y[t]
    Τέλος_επανάληψης
αλλιώς
    Για t από i μέχρι n
        z[k+t-i] ← x[t]
Τέλος_αν
```

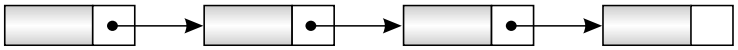
```
Αποτελέσματα  // z() //
Τέλος Συγχώνευση
```

3.9.3 Λίστες

Στις λίστες το κύριο χαρακτηριστικό είναι ότι οι κόμβοι τους συνήθως βρίσκονται σε απομακρυσμένες θέσεις μνήμης και η σύνδεσή τους γίνεται με δείκτες. Ο *δείκτης* (pointer) είναι ένα πεδίο του κόμβου, το οποίο δεν έχει κάποια συνήθη αριθμητική τιμή (όπως ακέραιος, πραγματικός, χαρακτήρας κ.λπ) αλλά δηλώνει μία διεύθυνση στην κύρια μνήμη. Στο σχήμα 3.3 παρουσιάζεται η δομή του κόμβου μίας λίστας, όπου υποτίθεται ότι το πρώτο πεδίο, που ονομάζεται info, είναι κάποια αλφαριθμητική πληροφορία, ενώ το δεύτερο πεδίο, που ονομάζεται next, είναι ο δείκτης που δείχνει στον επόμενο κόμβο της λίστας.

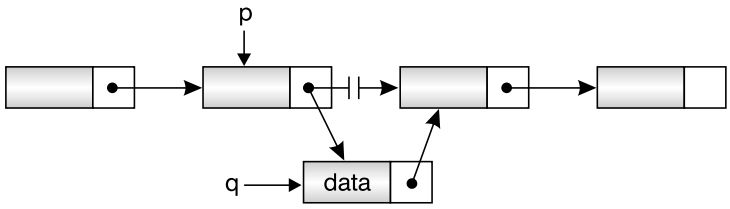
***Σχήμα 3.3**. Δομή κόμβου λίστας*

Με τη χρήση δεικτών διευκολύνονται οι λειτουργίες της εισαγωγής και της διαγραφής δεδομένων στις λίστες. Ο επόμενος αλγόριθμος περιγράφει τη λειτουργία της εισαγωγής σε μία λίστα.



***Σχήμα 3.4**. Μία λίστα με τέσσερις κόμβους*

```
Αλγόριθμος Εισαγωγή_σε_λίστα
Δεδομένα // p: δείκτης του κόμβου μετά τον οποίο θα γίνει
         η εισαγωγή data: περιεχόμενο νέου κόμβου
            q: δείκτης της θέσης νέου κόμβου //
q.info ← data
q.next ← p.next
p.next ← q
Τέλος Εισαγωγή_σε_λίστα
```



***Σχήμα 3.5.** Εισαγωγή σε λίστα*

Για την κατανόηση του προηγουμένου αλγορίθμου πρέπει να εξηγηθεί η χρησιμοποιούμενη σύμβαση: αν p είναι ένας δείκτης προς κάποιον κόμβο, τότε p.info είναι το περιεχόμενο του κόμβου και p.next είναι ο δείκτης του ίδιου κόμβου προς τον επόμενο κόμβο. Η εύρεση της θέσης μνήμης όπου θα γίνει η εισαγωγή του νέου κόμβου (στην περίπτωσή μας το q), γίνεται από το σύστημα και δεν απασχολεί τον προγραμματιστή.

***Σχήμα 3.6.** Δομή κόμβου λίστας*

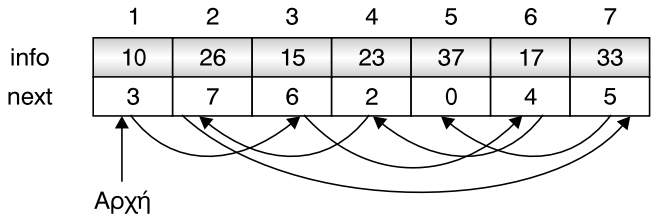
Στη συνέχεια δίνεται ένας απλός αλγόριθμος, που εκτελεί την πράξη της διαγραφής από μία λίστα. Πιο συγκεκριμένα, μας δίνεται ο δείκτης p που δείχνει προς τον κόμβο, του οποίου ο επόμενος θα διαγραφεί. Έτσι αν εφαρμοσθεί ο αλγόριθμος αυτός στη δεύτερη δομή του προηγούμενου σχήματος, θα προκύψει η πρώτη δομή (κατ'αντίστροφη φορά απ'ότι με την εισαγωγή).

```
Αλγόριθμος Διαγραφή_από_λίστα
Δεδομένα // p: δείκτης του κόμβου, του οποίου ο επόμενος
            θα διαγραφεί //
q ← p.next
p.next ← q.next
Τέλος Διαγραφή_από_λίστα
```

Οι λίστες που περιγράφηκαν λέγονται *δυναμικές* (dynamic), γιατί η υλοποίησή τους γίνεται έτσι ώστε να μην απαιτείται εκ των προτέρων καθορισμός του μέγιστου αριθμού κόμβων. Φυσικά, οι δομές αυτές είναι πιο ευέλικτες από τη στατική δομή του πίνακα, επειδή επεκτείνονται και συρρικνώνονται κατά τη διάρκεια της εκτέλεσης του προγράμματος. Οι παλαιότερες γλώσσες προγραμματισμού (ό-

πως οι Algol, Fortran, Cobol, κλπ.) δεν υποστήριζαν αυτές τις δυναμικές δομές, γιατί δεν προσέφεραν τη δυνατότητα δυναμικής παραχώρησης μνήμης κατά την εκτέλεση του προγράμματος. Παρ' όλα αυτά και στις γλώσσες αυτές είναι δυνατό να εφαρμοσθούν τεχνικές επεξεργασίας λιστών κάνοντας όμως χρήση πινάκων με σαφώς από την αρχή προκαθορισμένο μέγεθος.

Για παράδειγμα, στο σχήμα 3.7 παρουσιάζονται δύο πίνακες επτά θέσεων, οι Info και Next (εναλλακτικά μπορεί να χρησιμοποιηθεί ένας δισδιάστατος πίνακας). Στον πίνακα Info είναι αποθηκευμένα τα περιεχόμενα των κόμβων της λίστας, ενώ ο πίνακας Next χρειάζεται για την αποθήκευση των δεικτών. Στην υλοποίηση αυτή οι συγκεκριμένοι 'δείκτες' δεν εξυπηρετούν για να δείχνουν απομακρυσμένες θέσεις του πίνακα, αλλά λαμβάνουν τιμές ακέραιες για να δηλώσουν τη θέση μέσα στον πίνακα Info, όπου βρίσκεται το επόμενο λογικά στοιχείο. Έτσι αν διατρέξουμε τον πίνακα σύμφωνα με τις τιμές των δεικτών αυτών, θα προκύψουν τα δεδομένα σε αύξουσα ταξινομημένη σειρά.



Σχήμα 3.7. Στατικός πίνακας με χρήση λογικών δεικτών

3.9.4 Δένδρα

Ολες οι προηγούμενες δομές που εξετάστηκαν, λέγονται γραμμικές, διότι διαθέτουν τη σχέση της συνεχόμενης λογικής γειτονικότητας των στοιχείων τους. Τα δένδρα και οι γράφοι ανοίκουν στις μη γραμμικές δομές.

Η δομή του δένδρου σε γενικές γραμμές δεν είναι τίποτε άλλο παρά η σύνδεση κόμβων με τρόπο ανάλογο ενός πραγματικού δένδρου.

Τα δένδρα είναι ειδικές μορφές γράφων (graphs). Ενας ορισμός των δένδρων είναι ο ακόλουθος:

Δένδρο *είναι ένα σύνολο κόμβων που συνδέονται με ακμές. Υπάρχει ένας μόνο κόμβος που ονομάζεται ρίζα (root) και στον οποίο δεν καταλήγουν, αλλά μόνο ξε-*

κινούν ακμές. Από κάθε κόμβο μπορούν να ξεκινούν καμία, μία ή περισσότερες ακμές. Σε κάθε κόμβο (εκτός της ρίζας) καταλήγει μία μόνο ακμή.

Μερικοί ορισμοί.

Ενας κόμβος α είναι ο **πατέρας** (father) των κόμβων β και γ όταν από τον α αρχίζουν ακμές που καταλήγουν στους β και γ. Αντίστοιχα οι β και γ είναι τα **παιδιά** (children) του α. Η ορολογία αυτή επεκτείνεται προς τα πάνω ή κάτω με παππούδες και εγγονούς και γενικότερα προγόνους και απογόνους. Στο δένδρο του σχήματος 3.12 του βιβλίου μαθητή ο κόμβος Β έχει πατέρα τον κόμβο Α και παιδιά τους κόμβους Δ και Ε.

Βαθμός (degree) ενός κόμβου είναι ο αριθμός των παιδιών του. **Βαθμός ενός δένδρου** είναι ο μέγιστος από τους βαθμούς των κόμβων του. Δένδρα βαθμού n ονομάζονται n-αδικά (n-ary) δένδρα. Τα δένδρα που χρησιμοποιούνται συνήθως είναι δυαδικά (binary), τριαδικά (ternary) ή τετραδικά (quadtree).

Οι κόμβοι που δεν έχουν παιδιά λέγονται **τερματικοί κόμβοι** ή **φύλλα** (terminal nodes ή leaves). Οι υπόλοιποι κόμβοι λέγονται εσωτερικοί ή μη τερματικοί ή κλαδιά (internal ή nonterminal nodes ή branches).

Επίπεδο ενός κόμβου είναι ο αριθμός των προγόνων του (μέχρι τη ρίζα) συν 1. Το μέγιστο επίπεδο των κόμβων ενός δένδρου λέγεται **βάθος** (depth) ή **ύψος** (height) του δένδρου. Το δένδρο του σχήματος 3.12 είναι δυαδικό και έχει ύψος 3.

Υποδένδρο ονομάζεται το δένδρο που σχηματίζεται, αν ως ρίζα ληφθεί ένας οποιοσδήποτε κόμβος.

Αναπαράσταση των δένδρων στη μνήμη του υπολογιστή.

Στη μνήμη ενός υπολογιστή χρησιμοποιείται ανάλογη τεχνική για την αναπαράστασή τους, που χρησιμοποιείται και στις λίστες. Κάθε κόμβος αποτελείται από τα δεδομένα που περιέχει και από ένα σύνολο δεικτών ίσων σε πλήθος με το βαθμό του δένδρου. Οι δείκτες δείχνουν στα παιδιά του κόμβου. Αν δεν υπάρχει κάποιο παιδί, τότε ο αντίστοιχος δείκτης είναι κενός (nil).

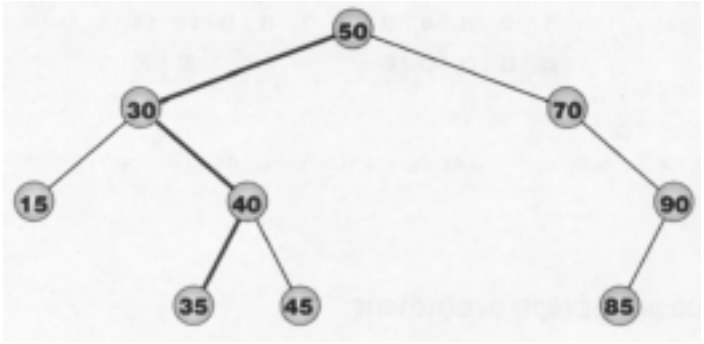
Εναλλακτικά μπορούν να χρησιμοποιηθούν n+1 πίνακες, όπου n ο βαθμός του δένδρου. Στο σχήμα 3.8 φαίνεται η αποθήκευση σε πίνακες του δυαδικού δένδρου του σχήματος 3.12 (βιβλίου μαθητή).

	Left	Info	Right
1	2	Α	3
2	4	Β	5
3	6	Γ	7
4	0	Δ	0
5	0	Ε	0
6	0	Ζ	0
7	0	Η	0

Σχήμα 3.8

➤ *Δυαδικά δένδρα αναζήτησης*

Μια πολύ ενδιαφέρουσα κατηγορία δένδρων είναι τα δυαδικά δένδρα αναζήτησης. Τα δένδρα αναζήτησης είναι διατεταγμένα δένδρα. Διατεταγμένο δένδρο λέγεται το δένδρο στο οποίο έχει σημασία η διάταξη των παιδιών κάθε κόμβου. Ετσι σε ένα δένδρο αναζήτησης η τιμή κάθε κόμβου είναι μεγαλύτερη από την τιμή όλων των κόμβων του αριστερού του υποδένδρου και μεγαλύτερη από την τιμή όλων των κόμβων του δεξιού του υποδένδρου. Ενα δυαδικό δένδρο αναζήτησης φαίνεται στο σχήμα 3.9.



Σχήμα 3.9.

Τα δένδρα αναζήτησης χρησιμοποιούνται όταν είναι σημαντική η γρήγορη προσπέλαση δεδομένων που υπακούουν σε κάποια σειρά κατάταξης.

Για την εύρεση ενός συγκεκριμένου κόμβου σε ένα δυαδικό δένδρο αναζήτησης πρώτα εξετάζεται η τιμή της ρίζας του δένδρου σε σχέση με την τιμή του κόμβου που ζητάμε. Αν ο ζητούμενος κόμβος είναι μικρότερος από τη ρίζα συνεχίζουμε με το αριστερό υποδένδρο, αν είναι μεγαλύτερος με το δεξί (φυσικά αν είναι ίσος τότε η αναζήτηση τελειώνει). Η αναζήτηση σταματά αν βρεθεί ο ζητούμενος κόμβος ή φτάσουμε σε κενό δείκτη. Στο σχήμα 3.9 φαίνεται με παχιά γραμμή η διαδρομή αναζήτησης του κόμβου με τιμή 35 και η μέθοδος αυτή παρουσιάζεται στον αλγόριθμο Search-Node.

```
Αλγόριθμος Search-Node
Δεδομένα // Root : ρίζα δένδρου, Κ : ζητούμενος κόμβος //
Β ← Root
Λ ← ψευδής
Οσο Β ≠ nil και Λ = ψευδής επανάλαβε
    Αν Β.Info = Κ τότε Λ = αληθής
    αλλιώς Αν Κ > Β.Info τότε Β ← Β.Right
                          αλλιώς Β ← Β.Left
          τέλος_αν
    Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα // Αν Λ αληθής τότε επιτυχής αναζήτηση. Αν Λ ψευδής τότε ανεπιτυχής αναζήτηση και Β η θέση στην οποία έπρεπε να βρίσκεται ο ζητούμενος κόμβος //
Τέλος Search-Node
```

Και εδώ εννοείται ότι, αν ένας κόμβος δεικτοδοτείται από το δείκτη Β, τότε Β.Info είναι το πεδίο δεδομένα και Β.Left, Β.Right οι δείκτες προς το αριστερό και δεξί παιδί αντίστοιχα.

Η εισαγωγή κόμβου σε δυαδικό δένδρο αναζήτησης γίνεται με τον εξής τρόπο: Αρχικά ακολουθείται η διαδικασία της αναζήτησης, αναζητώντας τον κόμβο που πρέπει να προστεθεί. Ο κόμβος φυσικά δεν θα υπάρχει και βρεθεί ένας κενός δείκτης. Ο κόμβος προστίθεται με τρόπο ώστε ο κενός δείκτης τώρα να δείχνει σ' αυτόν.

Η διαγραφή κόμβου είναι αρκετά πιο δύσκολη. Αν ο κόμβος που πρέπει να διαγραφεί είναι φύλλο, απλά διαγράφεται. Αν έχει ένα μόνο παιδί, τότε αντικαθίσταται από τον κόμβο παιδί. Αν όμως έχει δύο παιδιά τότε αντικαθίσταται είτε από τον πιο δεξιό κόμβο του αριστερού υποδένδρου του είτε από τον πιο αριστερό κόμβου του δεξιού υποδένδρου.

➤ *Διάσχιση δένδρων*

Μια άλλη σημαντική λειτουργία στα δένδρα είναι η διάσχιση (traverse). **Διάσχιση** λέγεται η λειτουργία κατά την οποία πρέπει να επισκεφθούμε κάθε κόμβο ακριβώς μια φορά.

Για την επεξήγηση της λειτουργίας αυτής είναι χρήσιμος ο επόμενος αναδρομικός ορισμός του δένδρου.

Δένδρο *είναι ένα σύνολο κόμβων που συνδέονται με ακμές. Υπάρχει ένας ειδικός κόμβος που ονομάζεται ρίζα (root) και στον οποίο δεν καταλήγουν αλλά μόνο ξεκινούν ακμές. Οι υπόλοιποι κόμβοι αποτελούν ξένα μεταξύ τους υποδένδρα (subtrees).*

Υπάρχουν τρεις κύριοι διαφορετικοί τρόποι επίσκεψης των κόμβων. Κάθε τρόπος απαιτεί την εκτέλεση των επόμενων τριών διαδικασιών με κάποια σειρά.

1. Επίσκεψη της ρίζας

2. Επίσκεψη του αριστερού υποδένδρου

3. Επίσκεψη του δεξιού υποδένδρου.

Οι διαδικασίες 2 και 3 απαιτούν την επίσκεψη ενός υποδένδρου που μπορεί να γίνει επαναλαμβάνοντας τις ίδιες διαδικασίες με την ίδια σειρά.

Προδιαταγμένη διάσχιση

Αν η σειρά πραγματοποίησης των διαδικασιών αυτών είναι αυτή που σημειώνεται πιο πάνω (δηλαδή 1, 2, 3), τότε η διάσχιση των κόμβων λέγεται προδιαταγμένη (preorder). Η προδιαταγμένη διάσχιση του δένδρου του σχήματος 3.12 δίδει το εξής αποτέλεσμα:

Α Β Δ Ε Γ Ζ Η

Στην προδιαταγμένη διάσχιση επισκεφτόμαστε τη ρίζα πριν από τα παιδιά της.

Η μέθοδος της προδιαταγμένης διάσχισης περιγράφεται με τον (αναδρομικό) αλγόριθμο Preorder-Traversal.

```
Αλγόριθμος Preorder-Traversal (R)
Δεδομένα // R: δείκτης ρίζας //
```

```
Οσο R ≠ nil επανάλαβε
    'επίσκεψη ρίζας
    Τύπωσε R.Info
    ' επίσκεψη αριστερού υποδένδρου
    A = R.Left
    Preorder-Traversal (A)
    'επίσκεψη δεξιού υποδένδρου
    Δ = R.Right
    Preorder-Traversal (Δ)
Τέλος_επανάληψης
Αποτελέσματα // λίστα κόμβων //
Τέλος Preorder-Traversal
```

Στον αλγόριθμο αυτό ως επεξεργασία του κόμβου εννοείται η εκτύπωση του περιεχομένου του. Η υλοποίηση του αλγορίθμου θα δόσει στην έξοδο τη σειρά επίσκεψης των κόμβων με την προδιαταγμένη διάσχιση που αναφέρθηκε πιο πάνω.

Μεταδιαταγμένη διάσχιση

Αν η σειρά εκτέλεσης των προηγούμενων διαδικασιών γίνεται 2, 3, 1 δηλαδή

- Επίσκεψη του αριστερού υποδένδρου
- Επίσκεψη του δεξιού υποδένδρου
- Επίσκεψη της ρίζας

τότε η διάσχιση αυτή λέγεται μεταδιαταγμένη (postorder) και το αποτέλεσμα για το δένδρο του σχήματος 3.12 είναι:

Δ Ε Β Ζ Η Γ Α

Στη μεταδιαταγμένη διάσχιση επισκεφτόμαστε τη ρίζα μετά τα παιδιά της.

Ενδοδιαταγμένη διάσχιση

Τέλος αν η σειρά εκτέλεσης των διαδικασιών αυτών γίνει 2, 1, 3 δηλαδή

- Επίσκεψη αριστερού υποδένδρου
- Επίσκεψη ρίζας

- Επίσκεψη δεξιού υποδένδρου

τότε ορίζεται η ενδοδιαταγμένη (inorder) διάσχιση, με αποτέλεσμα στο ίδιο γράφημα:

Δ Β Ε Α Ζ Γ Η

Στην ενδοδιαταγμένη διάσχιση επισκεφτόμαστε τη ρίζα μεταξύ των παιδιών της.

Ας σημειωθεί ότι μπορεί να υπάρξουν τρεις ακόμη μέθοδοι διάσχισης, αν η σειρά επίσκεψης των υποδένδρων αντιστραφεί, δηλαδή η επίσκεψη του δεξιού να προηγείται αυτής του αριστερού.

➤ *Πολωνικός συμβολισμός*

Το 1951 ο Πολωνός μαθηματικός Jan Lukasiewicz πρότεινε ένα συμβολισμό για αριθμητικές παραστάσεις χωρίς να κάνει χρήση παρενθέσεων. Ο συμβολισμός εφαρμόστηκε επίσης στην άλγεβρα καθώς και σε άλλα συστήματα τελεστών-τελεστέων και έγινε γνωστός με το όνομα πολωνικός συμβολισμός (polish notation), προς τιμή του Πολωνού μαθηματικού.

Ο πολωνικός συμβολισμός συνίσταται στο να τοποθετούνται οι τελεστές πριν από τους τελεστέους. Για παράδειγμα αντί για τις παραστάσεις:

1. α + β γράφουμε + α β
2. (α + β) . γ γράφουμε . + α β γ
3. α + (β . γ) γράφουμε + α . β γ
4. α . β + γ . δ γράφουμε + . α β . γ δ

Στον πολωνικό συμβολισμό η χρήση παρενθέσεων δεν είναι απαραίτητη με την προϋπόθεση ότι κάθε τελεστής έχει συγκεκριμένο αριθμό τελεστέων. Ετσι η έκφραση + α β έχει το ίδιο νόημα με την α + β, αφού γνωρίζουμε ότι ο τελεστής + έχει δύο τελεστέους που ακολουθούν τον τελεστή. Το σημείο μείον (-) δημιουργεί κάποιο πρόβλημα, αφού χρησιμοποιείται και σαν δυαδικός τελεστής (π.χ. α-β) αλλά και σαν μονοδικός (unary) τελεστής (π.χ. -α). Το πρόβλημα λύνεται αρκεί να περιορίσουμε τη χρήση του μείον σαν αποκλειστικά μονοδικού. Αυτό σημαίνει ότι η παράσταση α - β, πρέπει να γραφεί σαν α + (-β) και επόμενα μπορεί να γραφεί στον πολωνικό συμβολισμό + α - β χωρίς πρόβλημα.

Αν αντί να τοποθετούμε τους τελεστές πριν τους τελεστέους τους τοποθετούμε μετά, έχουμε τον **ανάστροφο πολωνικό συμβολισμό** (reverse polish notation). Για παράδειγμα αντί για τις παραστάσεις:

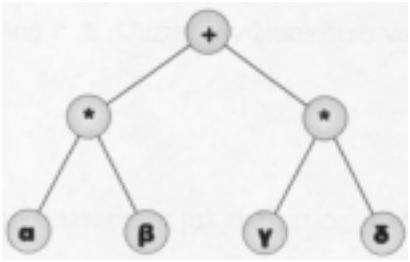
1. α + β γράφουμε α β +

2. (α + β) . γ γράφουμε α β + γ .

3. α + (β . γ) γράφουμε α β γ . +

4. α . β + γ . δ γράφουμε α β . γ δ . +

Μπορούμε να πάρουμε την πολωνική και την ανάστροφη πολωνική μορφή μιας παράστασης αν διασχίσουμε το δυαδικό δένδρο της παράστασης με προδιατεταγμένο και αντίστοιχα μεταδιατεταγμένο τρόπο. Π.χ. η παράσταση
α . β + γ . δ απεικονίζεται με το δυαδικό δένδρο του σχήματος 3.10.



Σχήμα 3.10

Η προδιαταγμένη διάσχιση του δένδρου αυτού δίδει τον πολωνικό συμβολισμό της παράστασης, που αποκαλείται **προθεματική** (prefix) μορφή :

+ . α β . γ δ

Η μεταδιαταγμένη διάσχιση του ίδιου δένδρου δίδει τον ανάστροφο πολωνικό συμβολισμό της παράστασης, που αποκαλείται **μεταθεματική** (postfix) μορφή :

α β . γ δ . +

Ας σημειωθεί τέλος, ότι η ενδοδιαταγμένη διάσχιση δίδει τη γνωστή **ένθετη** (infix) μορφή της παράστασης, δηλαδή:

α . β + γ . δ

Ο πολωνικός συμβολισμός χρησιμοποιείται σε μερικές γλώσσες προγραμματισμού αντί του γνωστού μας συμβολισμού με χρήση παρενθέσεων, π.χ. APL και FORTH. Χρήση βρίσκει επίσης και σε ορισμένους τύπους προγραμματιζόμενων αριθμομηχανών (π.χ. Hewlett- Packard). Μεγάλη χρήση βρίσκουν επίσης από τους μεταγλωττιστές και διερμηνευτές για την αποτίμηση εκφράσεων.

Παραστάσεις γραμμένες σε ανάστροφο πολωνικό συμβολισμό μπορούν να υπολογιστούν πολύ εύκολα από τα αριστερά προς τα δεξιά. Αρκεί όταν συναντάται κάποιος τελεστής να γίνεται η αντίστοιχη πράξη με τελεστέους αυτούς που υπολογίστηκαν πιο πριν.

Για να γίνει αυτό απαιτείται η χρήση μιας στοίβας και ο ορισμός μερικών πράξεων, όπως:

ADD Οι δύο τιμές στην κορυφή της στοίβας προστίθενται και το αποτέλεσμα τοποθετείται στην κορυφή της στοίβας.

MUL Οι δύο τιμές στην κορυφή της στοίβας πολλαπλασιάζονται και το αποτέλεσμα τοποθετείται στην κορυφή της στοίβας.

Ο αλγόριθμος αποτίμησης της έκφρασης διατρέχει την έκφραση από τ' αριστερά προς τα δεξιά και για κάθε τελεστέο που συναντά, εκτελεί μια λειτουργία push στη στοίβα. Οταν συναντήσει κάποιον τελεστή, τότε εκτελεί την αντίστοιχη πράξη ADD ή MUL (ή άλλη). Ετσι για να υπολογιστεί η παράσταση α β γ . + (δηλαδή η α+β.γ) πρέπει να γίνουν οι εξής πράξεις:

PUSH α

PUSH β

PUSH γ

MUL

ADD

και η στοίβα έχει στην κορυφή της την τιμή της παράστασης.

3.9.5 Γράφοι

Πολλά πρακτικά προβλήματα και καταστάσεις της καθημερινής μας ζωής μπορούν να περιγραφούν με τη βοήθεια ενός διαγράμματος, που αποτελείται

από ένα σύνολο σημείων και ένα σύνολο γραμμών που ενώνουν ζεύγη σημείων. Για παράδειγμα, θα μπορούσε τα σημεία να συμβολίζουν πρόσωπα και ο γραμμές να δηλώνουν πρόσωπα που είναι φίλοι μεταξύ τους. Η περιγραφή αυτή είναι ένας *γράφος* (graph). Άλλωστε και η δομή του σχήματος 4.1 (βιβλίου μαθητή) είναι ένα παράδειγμα γράφου. Ο γράφος αποτελεί την πιο γενική δομή δεδομένων. Δηλαδή, μπορεί να λεχθεί ότι οι πίνακες, οι λίστες και τα δένδρα είναι υπο-περιπτώσεις των γράφων.

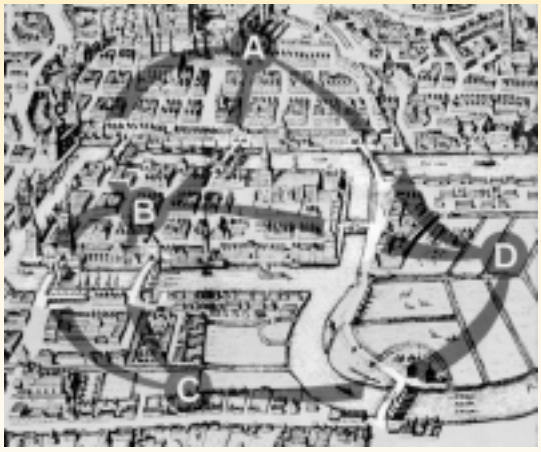
Ο γράφος αποθηκεύεται στη μνήμη του υπολογιστή με τη βοήθεια ενός δισδιάστατου πίνακα a, που ονομάζεται *πίνακας γειτνίασης* (adjacency matrix). Ο πίνακας αυτός είναι τετραγωνικός, δηλαδή nxn, όπου n είναι ο αριθμός των σημείων. Το στοιχείο a_{ij} του πίνακα είναι 0 αν τα σημεία i και j δεν ενώνονται, ενώ αν ενώνονται τότε το στοιχείο a_{ij} λαμβάνει τιμή ίση με το λεγόμενο *βάρος* (weight) της σύνδεσης, όπως είναι η χιλιομετρική απόσταση στο παράδειγμά μας. Ο επόμενος πίνακας είναι ο πίνακας γειτνίασης του προβλήματος του ταχυδρομικού διανομέα. Παρατηρούμε επίσης ότι ο πίνακας αυτός είναι *συμμετρικός* (symmetric), δηλαδή ισχύει $a_{ij} = a_{ji}$.

	0	5	13	6
	5	0	9	8
	13	9	0	10
	6	8	11	0

Στη γενική περίπτωση, ο πίνακας γειτνίασης δεν είναι συμμετρικός, δηλαδή δεν ισχύει πάντοτε η σχέση $a_{ij} = a_{ji}$. Για παράδειγμα, έστω ότι οι πλατείες και οι διασταυρώσεις ενός αστικού οδικού δικτύου αντιστοιχούν στα σημεία του γράφου, ενώ οι γραμμές του γράφου δηλώνουν τις αποστάσεις μεταξύ τους. Σε μία τέτοια περίπτωση, οι μονόδρομοι που ενώνουν δύο διαφορετικά σημεία μπορεί να έχουν διαφορετικό μήκος. Αν δεν υπάρχουν βάρη μεταξύ των σημείων, τότε όλα τα στοιχεία του πίνακα που αντιστοιχούν στις συνδέσεις αυτές λαμβάνουν την τιμή 1. Έτσι, ο πίνακας γειτνίασης είναι ένας *δυαδικός* (binary) πίνακας, επειδή αποτελείται από 0 και 1.

Θεωρία Γράφων

Η Θεωρία Γράφων είναι ένα πεδίο με μεγάλο ιστορικό βάθος. Βεβαίως, προϋπήρξε της Πληροφορικής ως ένα πεδίο των Μαθηματικών, αλλά η Πληροφορική έδωσε ώθηση στη Θεωρία Γράφων εμπλουτίζοντάς την με την έννοια του αλγορίθμου. Αναφέρεται ότι η πρώτη ιστορική διατριβή που έθεσε τις βάσεις της Θεωρίας Γράφων ήταν μία μελέτη του Leoard Euler που έλυσε το πρόβλημα των "γεφυρών του Koenigsberg", το 1736. Στο επόμενο σχήμα φαίνεται μια άποψη του Koenigsberg που διασχίζεται από τον ποταμό Pregel, ενώ μέσα στον ποταμό υπάρχουν 2 νησάκια που ενώνονται με τις όχθες και μεταξύ τους με 7 γέφυρες. Οι κάτοικοι του Koenigsberg δεν μπορούσαν να λύσουν το πρόβλημα "ποια είναι η πορεία που κάποιος πρέπει να ακολουθήσει ώστε να ξεκινήσει από ένα σημείο, να περάσει από όλες τις γέφυρες μία μόνο φορά και να επιστρέψει στο ίδιο σημείο". Ο Euler απόδειξε ότι δεν υπάρχει λύση στο πρόβλημα.



Μία άλλη ιστορική μελέτη που σημάδεψε τη Θεωρία Γράφων αποδίδεται στον Kirchoff, που το 1847 ανέπτυξε τη δομή των δένδρων και μελέτησε τις εφαρμογές τους στα ηλεκτρικά δίκτυα, ενώ λίγο αργότερα ο Cayley θεώρησε τη δομή των δένδρων για να απαριθμήσει τα ισομερή των κεκορεσμένων υδρογονανθράκων. Ένα ακόμη ιστορικό πρόβλημα ήταν ο "χρωματισμός χαρτών με 4 χρώματα", δηλαδή η δυνατότητα να χρωματισθούν οι περιοχές ενός πολιτικού χάρτη με 4 χρώματα το πολύ, έτσι ώστε δύο γειτονικές περιοχές να μην έχουν το ίδιο χρώμα. Σημειώνεται ότι το πρόβλημα αυτό τέθηκε το 1850 και επιλύθηκε το 1976.

3.10 Απάντηση δεύτερης λύσης του παραδείγματος της παραγράφου 3.2 βιβλίου μαθητή

Σχετικά με τη δεύτερη προσέγγιση του παραδείγματος του ΟΤΕ (σελίδα 51).

Η αλγοριθμική λύση της δεύτερης προσέγγισης έχει ως εξής. Κατ' αρχήν υποτίθεται ότι οι συνδρομητές είναι ταξινομημένοι αλφαβητικά και μπορούν να αποθηκευτούν σε ένα δισδιάστατο πίνακα (τον κύριο πίνακα) με τόσες γραμμές, όσοι είναι οι συνδρομητές και με δύο στήλες που αντιστοιχούν στο όνομα και το τηλέφωνο του συνδρομητή. Ομως επιπλέον χρειαζόμαστε μια ακόμη βοηθητική δομή, δηλαδή ένα δισδιάστατο πίνακα 24 γραμμών και 2 στηλών, όπου τα στοιχεία της πρώτης στήλης είναι τα γράμματα του αλφαβήτου, ενώ τα στοιχεία της δεύτερης στήλης δίνουν τη θέση στον πρώτο πίνακα, όπου είναι αποθηκευμένος ο πρώτος λεξικογραφικά συνδρομητής του οποίου το όνομα αρχίζει από το αντίστοιχο γράμμα.

Εναλλακτικά ο βοηθητικός πίνακας μπορεί να είναι μονοδιάστατος με 24 στοιχεία, όπου κάθε στοιχείο αντιστοιχεί κατά σειρά σε ένα από τα γράμματα του αλφαβήτου (δηλαδή Α, Β, κ.λπ.). Βέβαια, το περιεχόμενο κάθε θέσης αυτού του μονοδιάστατου πίνακα αντιστοιχεί στο συνδρομητή του οποίου το όνομα είναι το πρώτο λεξικογραφικά στον κύριο πίνακα και αρχίζει βέβαια από το αντίστοιχο γράμμα.

Επομένως για την εύρεση του τηλεφώνου ενός συνδρομητή, η αναζήτηση αρχίζει από τη μικρότερη βοηθητική δομή με βάση το πρώτο γράμμα του ονόματος του συνδρομητή. Ετσι εντοπίζεται εύκολα η θέση στον κύριο πίνακα, όπου είναι αποθηκευμένος ο πρώτος συνδρομητής του οποίου το όνομα αρχίζει από το αντίστοιχο γράμμα. Στη συνέχεια η αναζήτηση συνεχίζεται στον κύριο πίνακα σειριακά μέχρι την εύρεση του ονόματος του σχετικού συνδρομητή και τον εντοπισμό του αριθμού τηλεφώνου του.