8 Αλγόριθμοι

Εισαγωγή στην Επιστήμη των Υπολογιστών © Εκδόσεις Κλειδάριθμος

8.1

Στόχοι

Μετά την ολοκλήρωση αυτού του κεφαλαίου, ο σπουδαστής θα είναι σε θέση:

- Να ορίζει έναν αλγόριθμο και να τον συσχετίζει με τη λύση ενός προβλήματος.
- Να ορίζει τρεις δομές και να περιγράφει τη χρήση τους σε αλγορίθμους.
- Να περιγράφει τα διαγράμματα UML και τον ψευδοκώδικα, καθώς και τον τρόπο χρήσης τους σε αλγορίθμους.
- Να περιγράφει βασικούς αλγορίθμους και τις εφαρμογές τους.
- Να περιγράφει την έννοια της ταξινόμησης και να κατανοεί τους μηχανισμούς των τριών στοιχειωδών αλγορίθμων ταξινόμησης.
- Να περιγράφει την έννοια της αναζήτησης και να κατανοεί τους μηχανισμούς των δύο συνηθισμένων αλγορίθμων αναζήτησης.
- Να ορίζει τους υποαλγορίθμους και τις σχέσεις τους με τους αλγορίθμους.
- Να ξεχωρίζει τους επαναληπτικούς και τους αναδρομικούς αλγορίθμους

8.2

8-1 ΕΝΝΟΙΑ

Σε αυτή την ενότητα θα δώσουμε έναν ανεπίσημο ορισμό του **αλγορίθμου** και θα αναπτύξουμε την έννοια παραθέτοντας ένα παράδειγμα.

8.3

Ανεπίσημος ορισμός

Ένας ανεπίσημος ορισμός του αλγορίθμου είναι ο εξής:

i

Αλγόριθμος: μια βήμα προς βήμα μέθοδος για την επίλυση ενός προβλήματος ή τη διεκπεραίωση μιας εργασίας.

Εικόνα 8.1 Ανεπίσημος ορισμός αλγορίθμου που χρησιμοποιείται σε υπολογιστή

8.4

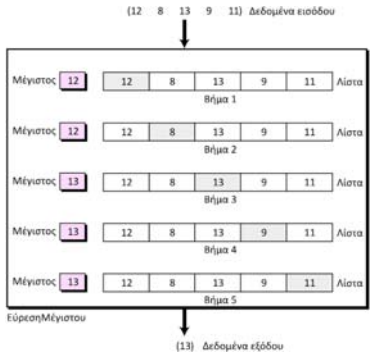
Παράδειγμα

Θέλουμε να κατασκευάσουμε έναν αλγόριθμο για την εύρεση του μεγαλύτερου ακεραίου από μια λίστα θετικών ακεραίων. Ο αλγόριθμος θα πρέπει να μπορεί να βρίσκει τον μεγαλύτερο ακέραιο από μια λίστα ακεραίων οποιουδήποτε μεγέθους (5, 1.000, 10.000, 1.000.000 κ.λπ.). Επίσης, πρέπει να είναι γενικός και να μην εξαρτάται από το πλήθος των ακεραίων.

Για τη λύση αυτού του προβλήματος απαιτείται μια διαισθητική προσέγγιση. Πρώτα θα χρησιμοποιήσουμε ένα μικρό πλήθος ακεραίων (για παράδειγμα, πέντε), και κατόπιν θα γενικεύσουμε τη λύση για οποιοδήποτε πλήθος ακεραίων.

Στην Εικόνα 8.2 παρουσιάζεται ένας τρόπος λύσης αυτού του προβλήματος. Θα ονομάσουμε τον αλγόριθμο *ΕύρεσηΜέγιστου*. Κάθε αλγόριθμος έχει ένα όνομα που τον διακρίνει από τους υπόλοιπους. Ο αλγόριθμος δέχεται ως είσοδο μια λίστα από πέντε ακεραίους και δίνει ως έξοδο τον μεγαλύτερο από αυτούς.

8.5

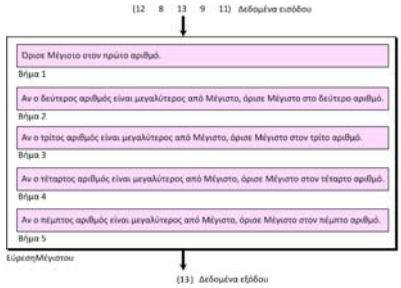


Εικόνα 8.2 Εύρεση του μέγιστου ακεραίου μεταξύ πέντε ακεραίων

8.6

Ορισμός ενεργειών

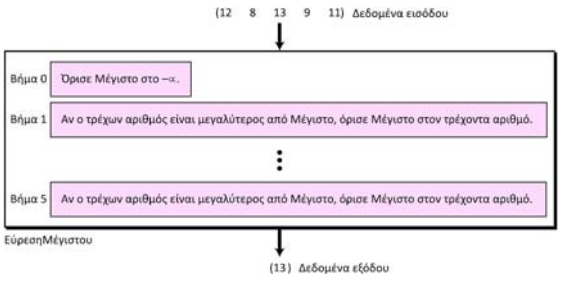
Η Εικόνα 8.2 δεν εξηγεί τι πρέπει να γίνει σε κάθε βήμα, οπότε μπορούμε να την τροποποιήσουμε ώστε να δείχνει περισσότερες λεπτομέρειες.



Εικόνα 8.3 Ορισμός ενεργειών στον αλγόριθμο ΕύρεσηΜέγιστου

Βελτίωση

Για να γίνει δεκτός ο συγκεκριμένος αλγόριθμος από την κοινότητα των προγραμματιστών χρειάζεται βελτίωση. Υπάρχουν δύο προβλήματα. Πρώτον, η ενέργεια στο πρώτο βήμα είναι διαφορετική από αυτές των υπόλοιπων βημάτων. Δεύτερον, η διατύπωση διαφέρει στα βήματα 2 έως 5. Μπορούμε πολύ εύκολα να βελτιώσουμε τον αλγόριθμο ώστε να διορθώσουμε αυτά τα δύο προβλήματα, αλλάζοντας τη διατύπωση στα βήματα 2 έως 5 σε "Αν ο τρέχων αριθμός είναι μεγαλύτερος από τον Μέγιστο, όρισε τον Μέγιστο στον τρέχοντα αριθμό". Ο λόγος που το πρώτο βήμα διαφέρει από τα υπόλοιπα είναι ότι ο Μέγιστος δεν έχει αρχική τιμή. Αν δώσουμε στον Μέγιστο αρχική τιμή ίση με $-\infty$ (αρνητικό άπειρο), τότε το πρώτο βήμα μπορεί να γίνει ίδιο με τα υπόλοιπα. Έτσι, εισάγουμε ένα νέο βήμα το οποίο ονομάζουμε βήμα 0, για να δείξουμε ότι πρέπει να πραγματοποιηθεί πριν από την επεξεργασία των ακεραίων.



Εικόνα 8.4 Ο βελτιωμένος αλγόριθμος ΕύρεσηΜέγιστου

Γενίκευση

Μπορούμε άραγε να γενικεύσουμε αυτόν τον αλγόριθμο; Θέλουμε να βρούμε τον μεγαλύτερο από n θετικούς ακεραίους, όπου το n μπορεί να είναι 1.000, 1.000.000, ή ακόμα μεγαλύτερο. Βέβαια, μπορούμε να ακολουθήσουμε την Εικόνα 8.4 και να επαναλάβουμε κάθε βήμα της. Αλλά αν θέλουμε να μετατρέψουμε τον αλγόριθμο σε πρόγραμμα, τότε πρέπει να πληκτρολογήσουμε τις ενέργειες για n βήματα!

Υπάρχει ένας καλύτερος τρόπος για να γίνει αυτό. Μπορούμε να πούμε στον υπολογιστή να επαναλάβει τα βήματα n φορές. Θα συμπεριλάβουμε τώρα αυτή τη λειτουργία στη σχηματική αναπαράσταση του αλγορίθμου μας (Εικόνα 8.5).

Εικόνα 8.5 Γενίκευση του αλγορίθμου ΕύρεσηΜέγιστου

8-2 ΤΡΕΙΣ ΔΟΜΕΣ

Οι επιστήμονες των υπολογιστών έχουν καθορίσει **τρεις δομές** για τα δομημένα προγράμματα ή τους αλγορίθμους. Η ιδέα είναι ότι ένα πρόγραμμα πρέπει να αποτελεί συνδυασμό μόνο αυτών των τριών δομών: ***ακολουθία***, ***απόφαση*** (επιλογή), και ***επανάληψη*** (Εικόνα 8.6). Έχει αποδειχθεί ότι δεν χρειάζεται καμία επιπλέον δομή. Η χρήση αυτών των τριών δομών διευκολύνει την κατανόηση, τη διόρθωση, και τη μετατροπή των προγραμμάτων και των αλγορίθμων.

Εικόνα 8.6 Οι τρεις δομές

Ακολουθία

Η πρώτη δομή ονομάζεται ακολουθία. Ένας αλγόριθμος, και τελικά ένα πρόγραμμα, είναι μια ακολουθία εντολών, οι οποίες μπορεί να είναι απλές εντολές ή κάποια από τις άλλες δύο δομές.

Απόφαση

Ορισμένα προβλήματα δεν μπορούν να επιλυθούν μόνο με μια ακολουθία απλών εντολών. Σε ορισμένες περιπτώσεις πρέπει να ελεγχθεί κάποια συνθήκη. Αν το αποτέλεσμα του ελέγχου είναι η τιμή "αληθής", τότε εκτελείται μια ακολουθία εντολών, ενώ αν η τιμή είναι "ψευδής", τότε εκτελείται μια διαφορετική ακολουθία εντολών. Αυτή η δομή ονομάζεται δομή απόφασης (ή επιλογής).

Επανάληψη

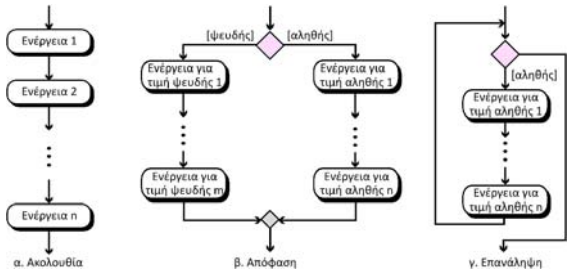
Σε μερικά προβλήματα πρέπει να επαναληφθεί η ίδια ακολουθία εντολών. Αυτό μπορούμε να το χειριστούμε με τη δομή επανάληψης ή ***βρόχου***. Μια τέτοια δομή μπορεί να χρησιμοποιηθεί και για την εύρεση του μεγαλύτερου ακεραίου από ένα σύνολο ακεραίων.

8-3 ΑΝΑΠΑΡΑΣΤΑΣΗ ΑΛΓΟΡΙΘΜΩΝ

Μέχρι τώρα χρησιμοποιήσαμε εικόνες για την απόδοση της έννοιας του αλγορίθμου. Ωστόσο, κατά τη διάρκεια των τελευταίων δεκαετιών αναπτύχθηκαν γι' αυτόν τον σκοπό διάφορα εργαλεία. Στο κεφάλαιο αυτό θα παρουσιάσουμε δύο από αυτά, τα **διαγράμματα UML** και τον **ψευδοκώδικα**.

UML

Η Ενοποιημένη Γλώσσα Μοντελοποίησης (Unified Modeling Language, UML) χρησιμοποιείται για τη σχηματική αναπαράσταση ενός αλγορίθμου. Δεν περιέχει τις λεπτομέρειες του αλγορίθμου αλλά προσπαθεί να δώσει τη γενική ιδέα του, παρουσιάζοντας τον τρόπο που ο αλγόριθμος "ρέει" από την αρχή μέχρι το τέλος του. Η UML καλύπτεται με περισσότερες λεπτομέρειες στο Παράρτημα Β. Εδώ θα δούμε μόνο το πώς αναπαρίστανται με αυτήν οι τρεις δομές (Εικόνα 8.7).



Εικόνα 8.7 Διάγραμμα UML των τριών δομών

Ψευδοκώδικας

Ο ψευδοκώδικας είναι μια αναπαράσταση ενός αλγορίθμου σε φυσική γλώσσα. Δεν υπάρχει κάποιο πρότυπο για τον ψευδοκώδικα - κάποιοι χρησιμοποιούν πολλές λεπτομέρειες, και κάποιοι άλλοι λιγότερες. Ορισμένοι χρησιμοποιούν έναν τύπο κώδικα που είναι πολύ κοντά στη φυσική γλώσσα, και κάποιοι άλλοι χρησιμοποιούν μια σύνταξη παρόμοια με αυτή της γλώσσας Pascal.

Ο ψευδοκώδικας περιγράφεται με λεπτομέρειες στο Παράρτημα Γ. Στην ενότητα αυτή θα δούμε μόνο το πώς αναπαρίστανται με αυτόν οι τρεις δομές (Εικόνα 8.8).

Εικόνα 8.8 **Ψευδοκώδικας των τριών δομών**

Παράδειγμα 8.1

Γράψτε σε ψευδοκώδικα έναν αλγόριθμο που να υπολογίζει το άθροισμα δύο ακεραίων.

Αλγόριθμος 8.1 Υπολογισμός του αθροίσματος δύο ακεραίων

```
Αλγόριθμος: ΆθροισμαΤωνΔύο (πρώτος, δεύτερος)
Σκοπός: Υπολογισμός του αθροίσματος δύο ακεραίων
Προ-συνθήκη: Δίνονται: δύο ακέραιοι (πρώτος και δεύτερος)
Μετα-συνθήκη: Τίποτα
Επιστρέφεται: Η τιμή του αθροίσματος
{
    άθροισμα ← πρώτος ← δεύτερος
    επιστροφή άθροισμα
}
```


Παράδειγμα 8.2

Γράψτε έναν αλγόριθμο που να αντικαθιστά μια αριθμητική βαθμολογία με μια αξιολόγηση "προάγεται" ή "απορρίπτεται".

Αλγόριθμος 8.2 Αξιολόγηση βαθμολογίας

```
Αλγόριθμος: ΑξιολόγησηΒαθμολογίας (βαθμός)
Σκοπός: Αξιολόγηση προαγωγής ή απόρριψης με βάση τη βαθμολογία
Προ-συνθήκη: Δίνεται η βαθμολογία που θα αξιολογηθεί
Μετα-συνθήκη: Τίποτα
Επιστρέφεται: Η αξιολόγηση
{
    αν (βαθμολογία ≥ 70)        αξιολόγηση ← "προάγεται"
    αλλιώς                      αξιολόγηση ← "απορρίπτεται"
    Επιστροφή αξιολόγηση
}
```


Παράδειγμα 8.3

Γράψτε έναν αλγόριθμο που να αντικαθιστά μια αριθμητική βαθμολογία (έναν ακέραιο) με μια αξιολόγηση βάσει γραμμάτων.

Αλγόριθμος 8.3 Αξιολόγηση βάσει γραμμάτων

```
Αλγόριθμος: ΑξιολόγησηΓραμμάτων (βαθμός)
Σκοπός: Αξιολόγηση βάσει γραμμάτων ανάλογα με τον βαθμό
Προ-συνθήκη: Δίνεται μια αριθμητική βαθμολογία
Μετα-συνθήκη: Τίποτα
Επιστρέφεται: Μια αξιολόγηση με γράμμα
{
    αν (100 ≥ βαθμολογία ≥ 90)    αξιολόγηση ← 'Α'
    αν (80 ≥ βαθμολογία ≥ 89)     αξιολόγηση ← 'Β'
    αν (70 ≥ βαθμολογία ≥ 79)     αξιολόγηση ← 'Γ'
    αν (60 ≥ βαθμολογία ≥ 69)     αξιολόγηση ← 'Δ'
    αν (0 ≥ βαθμολογία ≥ 59)      αξιολόγηση ← 'Ε'
    επιστροφή αξιολόγηση
}
```


Παράδειγμα 8.4

Γράψτε έναν αλγόριθμο που να βρίσκει τον μέγιστο από ένα σύνολο ακεραίων. Το πλήθος των ακεραίων δεν είναι γνωστό.

Αλγόριθμος 8.4 Εύρεση του μέγιστου ακεραίου σε ένα σύνολο ακεραίων

```
Αλγόριθμος: ΕύρεσηΜέγιστου (λίστα)
Σκοπός: Εύρεση του μέγιστου ακεραίου σε ένα σύνολο ακεραίων
Προ-συνθήκη: Δίνεται το σύνολο των ακεραίων
Μετα-συνθήκη: Τίποτα
Επιστρέφεται: Ο μέγιστος ακέραιος
{
    μέγιστος ← −∞
    όσο (υπάρχουν άλλοι ακέραιοι για έλεγχο)
    {
        τρέχων ← επόμενος ακέραιος
        αν (τρέχων > μέγιστος)    μέγιστος ← τρέχων
    }
    επιστροφή μέγιστος
}
```

Παράδειγμα 8.5

Γράψτε έναν αλγόριθμο που να βρίσκει τον μέγιστο από ένα σύνολο 1.000 ακεραίων.

Αλγόριθμος 8.5 Εύρεση του μέγιστου ακεραίου από 1000 ακέραιους

```
Αλγόριθμος: ΕύρεσηΜέγιστου2 (λίστα)
Σκοπός: Η εύρεση και η επιστροφή του μέγιστου ακεραίου από τους πρώτους 1000 ακεραίους
Προ-συνθήκη: Δίνεται ένα σύνολο με περισσότερους από 1000 ακέραιους
Μετα-συνθήκη: Τίποτα
Επιστρέφεται: Ο μέγιστος ακέραιος
{
        μέγιστος ← −∞
        μετρητής ← 1
        όσο (μετρητής ≤ 1000)
        {
                τρέχων ← επόμενος ακέραιος

                αν (τρέχων > μέγιστος)          μέγιστος ← τρέχων


                μετρητής ← μετρητής + 1

        }
        επιστροφή μέγιστος
}
```

8.25

8-4 ΕΝΑΣ ΠΙΟ ΑΥΣΤΗΡΟΣ ΟΡΙΣΜΟΣ

Τώρα που περιγράψαμε την έννοια του αλγορίθμου και έχουμε δείξει τον τρόπο αναπαράστασής του, είμαστε σε θέση να δώσουμε έναν πιο αυστηρό ορισμό.

i

Αλγόριθμος:
ένα διατεταγμένο σύνολο από σαφή βήματα το οποίο παράγει κάποιο αποτέλεσμα και τερματίζεται σε πεπερασμένο χρόνο.

8.26

Διατεταγμένο σύνολο

Ένας αλγόριθμος πρέπει να είναι ένα καλώς ορισμένο, διατεταγμένο σύνολο εντολών.

Σαφή βήματα

Κάθε βήμα ενός αλγορίθμου πρέπει να ορίζεται με σαφή και ξεκάθαρο τρόπο. Αν ένα βήμα αφορά την πρόσθεση δύο ακεραίων, πρέπει να καθορίζονται τόσο οι δύο ακέραιοι όσο και η πράξη της πρόσθεσης· για παράδειγμα, δεν μπορεί να χρησιμοποιείται σε ένα σημείο ένα σύμβολο για την πρόσθεση, και το ίδιο σύμβολο να χρησιμοποιείται για τον πολλαπλασιασμό κάπου αλλού.

8.27

Παραγωγή αποτελέσματος

Ένας αλγόριθμος πρέπει να παράγει κάποιο αποτέλεσμα, διαφορετικά είναι άχρηστος. Το αποτέλεσμα μπορεί να είναι δεδομένα που επιστρέφονται στον καλούντα αλγόριθμο ή κάποια άλλη ενέργεια (για παράδειγμα, εκτύπωση).

Τερματισμός σε πεπερασμένο χρόνο

Ένας αλγόριθμος πρέπει να τερματίζεται (να σταματά η εκτέλεσή του). Αν δεν το κάνει (για παράδειγμα, αν έχει κάποιον ατέρμονα βρόχο), τότε δεν είναι αλγόριθμος. Στο Κεφάλαιο 17 θα ασχοληθούμε με το θέμα των επιλύσιμων και μη επιλύσιμων προβλημάτων, και θα δούμε ότι ένα επιλύσιμο πρόβλημα έχει λύση με τη μορφή ενός αλγορίθμου που τερματίζεται.

8.28

8-5 ΒΑΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ

Κάποιοι αλγόριθμοι χρησιμοποιούνται στην επιστήμη των υπολογιστών τόσο διαδεδομένα που θεωρούνται βασικοί. Εδώ θα περιγράψουμε τους πιο γνωστούς από αυτούς. Η περιγραφή είναι πολύ γενική, και η υλοποίηση εξαρτάται από τη γλώσσα.

8.29

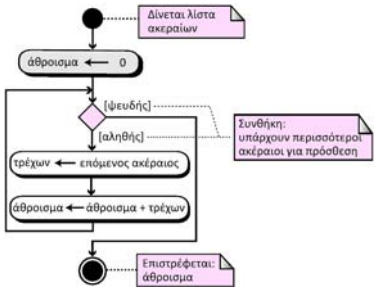
Άθροιση

Ένας από τους πιο πολυχρησιμοποιημένους αλγορίθμους στην επιστήμη των υπολογιστών είναι η **άθροιση**. Η άθροιση δύο ή τριών ακεραίων μπορεί να γίνει πολύ εύκολα, αλλά πώς μπορούμε να προσθέσουμε πολλούς ακέραιους; Η λύση είναι απλή: χρησιμοποιούμε τον τελεστή της πρόσθεσης σε έναν βρόχο (Εικόνα 8.9).

Ένας αλγόριθμος άθροισης αποτελείται από τρία λογικά μέρη:

1. Την ανάθεση αρχικής τιμής στο άθροισμα στην αρχή.
2. Τον βρόχο, ο οποίος σε κάθε επανάληψη προσθέτει στο άθροισμα ένα νέο ακέραιο.
3. Την επιστροφή του αποτελέσματος μετά την έξοδο από τον βρόχο.

8.30



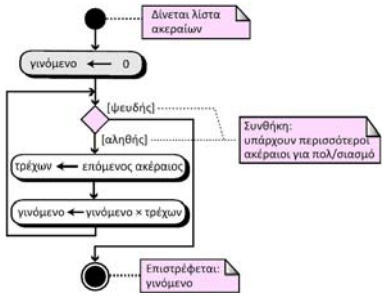
Εικόνα 8.9 Αλγόριθμος άθροισης

Γινόμενο

Άλλος ένας συνηθισμένος αλγόριθμος είναι η εύρεση του γινομένου μιας λίστας ακεραίων. Και εδώ η λύση είναι απλή: χρησιμοποιούμε τον τελεστή του πολλαπλασιασμού σε έναν βρόχο (Εικόνα 8.10).

Ένας αλγόριθμος γινομένου αποτελείται και αυτός από τρία λογικά μέρη:

1. Την ανάθεση αρχικής τιμής στο γινόμενο στην αρχή.
2. Τον βρόχο, ο οποίος σε κάθε επανάληψη πολλαπλασιάζει το γινόμενο με ένα νέο ακέραιο.
3. Την επιστροφή του αποτελέσματος μετά την έξοδο από τον βρόχο.



Εικόνα 8.10 Αλγόριθμος γινομένου

Ελάχιστο και μέγιστο

Στην αρχή του κεφαλαίου περιγράψαμε τον αλγόριθμο για την εύρεση του μέγιστου από μια λίστα ακεραίων. Η ιδέα ήταν η γραφή μιας δομής απόφασης που θα βρίσκει τον μεγαλύτερο από δύο ακεραίους. Αν τοποθετήσουμε αυτή τη δομή σε έναν βρόχο, μπορούμε να βρούμε τον μέγιστο από μια λίστα ακεραίων.

Η εύρεση του ελάχιστου από μια λίστα ακεραίων είναι παρόμοια, με δύο μικρές διαφορές. Πρώτον, χρησιμοποιούμε μια δομή απόφασης που να βρίσκει τον μικρότερο από δύο ακεραίους. Δεύτερον, χρησιμοποιούμε ως αρχική τιμή έναν πολύ μεγάλο ακέραιο αντί για έναν πολύ μικρό.

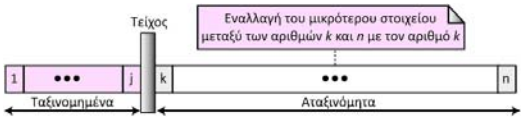
Ταξινόμηση

Μια από τις συχνές εφαρμογές στην επιστήμη των υπολογιστών είναι η ταξινόμηση, δηλαδή η διαδικασία διάταξης των δεδομένων με βάση τις τιμές τους. Οι άνθρωποι περιτριγυρίζονται από δεδομένα. Αν αυτά τα δεδομένα δεν είχαν κάποια λογική σειρά, θα χρειάζονταν ολόκληρες ώρες για την εύρεση μιας απλής πληροφορίας. Φανταστείτε πόσο δύσκολο θα ήταν να βρείτε τον αριθμό τηλεφώνου ενός ατόμου σε έναν τηλεφωνικό κατάλογο που δεν είναι ταξινομημένος.

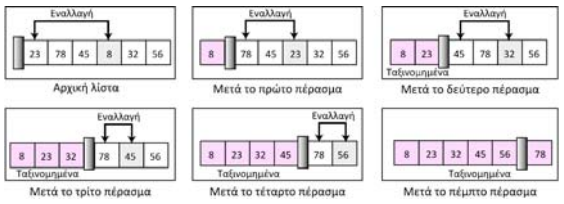
Σε αυτή την ενότητα θα παρουσιάσουμε τρεις αλγορίθμους ταξινόμησης: την **ταξινόμηση με επιλογή**, την **ταξινόμηση φυσαλίδας**, και την **ταξινόμηση με εισαγωγή**. Αυτοί οι τρεις αλγόριθμοι ταξινόμησης αποτελούν τη βάση για τους γρηγορότερους αλγορίθμους ταξινόμησης που χρησιμοποιούνται σήμερα στην επιστήμη των υπολογιστών.

Ταξινόμηση με επιλογή

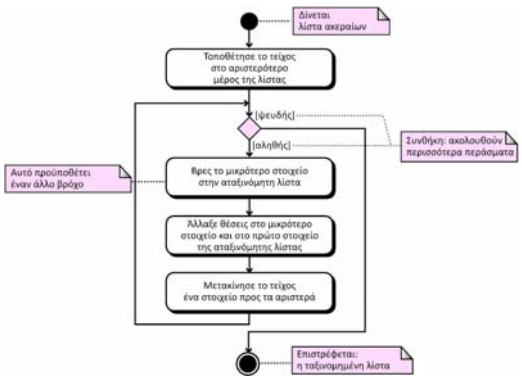
Στην **ταξινόμηση με επιλογή** (selection sort) η λίστα διαιρείται σε δύο υπολίστες — την ταξινομημένη και την αταξινόμητη — οι οποίες χωρίζονται από ένα φανταστικό τείχος. Βρίσκουμε το μικρότερο στοιχείο από την αταξινόμητη υπολίστα και το αντιμεταθέτουμε με το στοιχείο που βρίσκεται στην αρχή των ταξινομημένων δεδομένων. Μετά από κάθε επιλογή και αντιμετάθεση, το φανταστικό τείχος μεταξύ των δύο υπολιστών μετακινείται ένα στοιχείο μπροστά.



Εικόνα 8.11 Ταξινόμηση με επιλογή



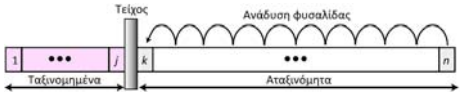
Εικόνα 8.12 **Παράδειγμα ταξινόμησης με επιλογή**



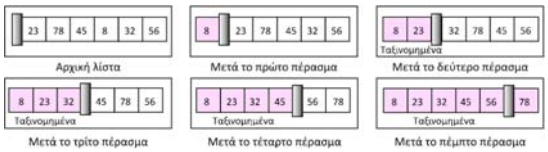
Εικόνα 8.13 **Αλγόριθμος ταξινόμησης με επιλογή**

Ταξινόμηση φυσαλίδας

Με τη μέθοδο της ταξινόμησης φυσαλίδας (bubble sort) η λίστα χωρίζεται επίσης σε δύο υπολίστες - μια ταξινομημένη και μια αταξινόμητη. Το μικρότερο στοιχείο της αταξινόμητης υπολίστας αναδύεται (σαν φυσαλίδα) και περνά στην ταξινομημένη υπολίστα. Μετά τη μετακίνηση του μικρότερου στοιχείου στην ταξινομημένη λίστα, το τείχος μετακινείται ένα στοιχείο εμπρός.



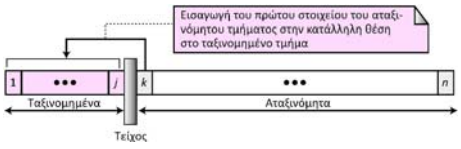
Εικόνα 8.14 **Ταξινόμηση φυσαλίδας**



Εικόνα 8.15 **Παράδειγμα ταξινόμησης φυσαλίδας**

Ταξινόμηση με εισαγωγή

Ο αλγόριθμος ταξινόμησης με εισαγωγή (insertion sort) αποτελεί μία από τις πιο συνηθισμένες τεχνικές ταξινόμησης, και συχνά χρησιμοποιείται από τους χαρτοπαίκτες. Κάθε φύλλο που σηκώνει ο παίκτης τοποθετείται στο σωστό σημείο του χεριού του ώστε να διατηρείται μια συγκεκριμένη σειρά.



Εικόνα 8.16 **Ταξινόμηση με εισαγωγή**

Εικόνα 8.17 **Παράδειγμα ταξινόμησης με εισαγωγή**

Αναζήτηση

Άλλος ένας συνηθισμένος αλγόριθμος στην επιστήμη των υπολογιστών είναι η αναζήτηση, δηλαδή η διαδικασία του εντοπισμού της θέσης ενός στόχου σε μια λίστα αντικειμένων. Στην περίπτωση μιας λίστας, αναζήτηση σημαίνει ότι, με δεδομένη μια τιμή, πρέπει να εντοπιστεί η θέση του πρώτου στοιχείου της λίστας το οποίο περιέχει τη συγκεκριμένη τιμή. Υπάρχουν δύο βασικοί τύποι αναζήτησης για λίστες: η **ακολουθιακή αναζήτηση** και η **δυαδική αναζήτηση**. Η ακολουθιακή αναζήτηση μπορεί να χρησιμοποιηθεί για τον εντοπισμό ενός στοιχείου σε οποιαδήποτε λίστα, ενώ η δυαδική αναζήτηση προϋποθέτει η λίστα να είναι ταξινομημένη.

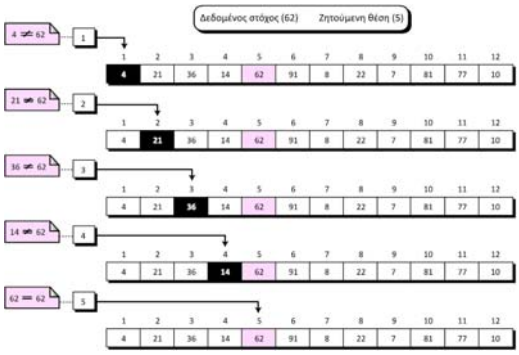
8.43

Ακολουθιακή αναζήτηση

Η ακολουθιακή αναζήτηση (sequential search) χρησιμοποιείται όταν η λίστα στην οποία θα πραγματοποιηθεί η αναζήτηση δεν είναι ταξινομημένη. Γενικά, η τεχνική αυτή είναι κατάλληλη μόνο για μικρές λίστες ή για λίστες στις οποίες δεν γίνεται συχνά αναζήτηση. Στις άλλες περιπτώσεις, η καλύτερη προσέγγιση είναι να ταξινομηθεί πρώτα η λίστα και μετά να χρησιμοποιηθεί δυαδική αναζήτηση, την οποία θα περιγράψουμε αργότερα.

Στην ακολουθιακή αναζήτηση ξεκινάμε την έρευνα για τον στόχο από την αρχή της λίστας, και συνεχίζουμε μέχρι να βρούμε τον στόχο ή να φτάσουμε στο τέλος της λίστας.

8.44



Εικόνα 8.18 Παράδειγμα ακολουθιακής αναζήτησης

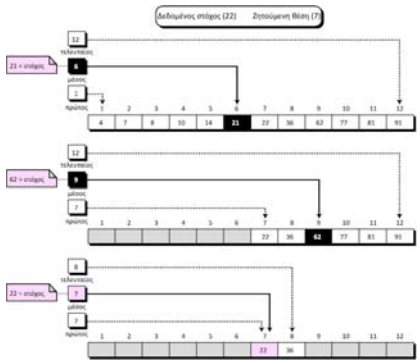
8.45

Δυαδική αναζήτηση

Ο αλγόριθμος της ακολουθιακής αναζήτησης είναι πολύ αργός. Αν η λίστα διαθέτει ένα εκατομμύριο στοιχεία, ίσως χρειαστεί να γίνουν μέχρι και ένα εκατομμύριο συγκρίσεις. Όταν η λίστα δεν είναι ταξινομημένη αυτή είναι και η μοναδική λύση. Αν όμως η λίστα είναι ταξινομημένη μπορούμε να χρησιμοποιήσουμε έναν πιο αποδοτικό αλγόριθμο ο οποίος ονομάζεται δυαδική αναζήτηση (binary search). Γενικά, οι προγραμματιστές χρησιμοποιούν δυαδική αναζήτηση όταν η λίστα είναι μεγάλη.

Η δυαδική αναζήτηση ξεκινά με τον έλεγχο του στοιχείου που βρίσκεται στο μέσο της λίστας. Με αυτή την ενέργεια προσδιορίζεται αν ο στόχος βρίσκεται στο πρώτο ή στο δεύτερο μισό της λίστας. Αν βρίσκεται στο πρώτο μισό, το δεύτερο μισό δεν χρειάζεται να ελεγχθεί περαιτέρω. Αν πάλι βρίσκεται στο δεύτερο μισό, αυτό που δεν χρειάζεται να ελεγχθεί περαιτέρω είναι το πρώτο μισό. Με άλλα λόγια έχουμε αποκλείσει τη μισή λίστα από τη συνέχεια της διαδικασίας.

8.46



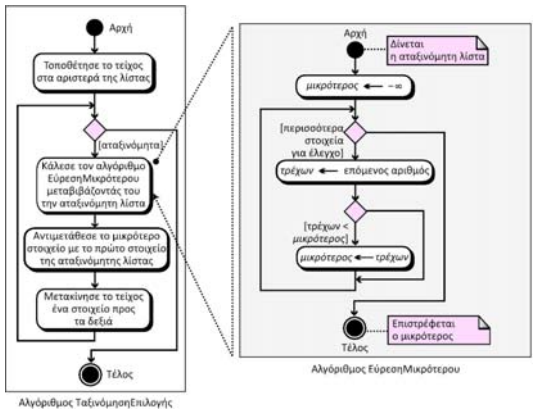
Εικόνα 8.19 Παράδειγμα δυαδικής αναζήτησης

8.47

8-6 ΥΠΟΑΛΓΟΡΙΘΜΟΙ

Με τις τρεις δομές που περιγράψαμε στην Ενότητα 8.2 μπορούμε να δημιουργήσουμε έναν αλγόριθμο για κάθε επιλύσιμο πρόβλημα. Ωστόσο, οι αρχές του δομημένου προγραμματισμού απαιτούν ένας αλγόριθμος να χωρίζεται σε μικρότερες μονάδες που ονομάζονται **υποαλγόριθμοι**. Κάθε υποαλγόριθμος με τη σειρά του διαιρείται σε μικρότερους υποαλγορίθμους. Ένα καλό παράδειγμα είναι ο αλγόριθμος για την ταξινόμηση με επιλογή που παρουσιάζεται στην Εικόνα 8.13.

8.48



Εικόνα 8.20 Η έννοια του υποαλγορίθμου

Διάγραμμα δομής

Άλλο ένα εργαλείο που χρησιμοποιούν οι προγραμματιστές είναι το διάγραμμα δομής (structure chart). Το διάγραμμα δομής αποτελεί ένα εργαλείο σχεδιασμού υψηλού επιπέδου, το οποίο δείχνει τη σχέση μεταξύ των αλγορίθμων και των υποαλγορίθμων. Χρησιμοποιείται κυρίως στο στάδιο του σχεδιασμού και όχι στο στάδιο του προγραμματισμού. Το διάγραμμα δομής περιγράφεται συνοπτικά στο Παράρτημα Δ.

8-7 ΑΝΑΔΡΟΜΗ

Γενικά, υπάρχουν δύο προσεγγίσεις στη συγγραφή αλγορίθμων για τη λύση ενός προβλήματος. Στη μία προσέγγιση χρησιμοποιούνται **επαναλήψεις**, και στην άλλη **αναδρομή**. Αναδρομή (recursion) ονομάζεται η διαδικασία κατά την οποία ένας αλγόριθμος καλεί τον εαυτό του.

Επαναληπτικός ορισμός

Για να χρησιμοποιήσουμε ένα απλό παράδειγμα, φανταστείτε τον υπολογισμό ενός παραγοντικού. Το παραγοντικό ενός ακεραίου είναι το γινόμενο όλων των ακέραιων τιμών από το 1 έως τον συγκεκριμένο ακέραιο. Ο ορισμός είναι επαναληπτικός (Εικόνα 8.21). Ένας αλγόριθμος είναι επαναληπτικός όταν στον ορισμό του δεν αναφέρει τον εαυτό του.

$$\text{Παραγοντικό}\,(n) = \begin{cases} 1 & \text{αν } n = 0 \\ n \times (n-1) \times (n-2) \quad \cdots \quad 3 \times 2 \times 1 & \text{αν } n > 0 \end{cases}$$

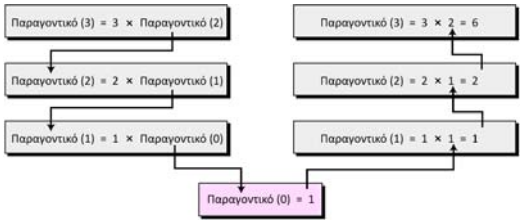
Εικόνα 8.21 Επαναληπτικός ορισμός του παραγοντικού

Αναδρομικός ορισμός

Ένας αλγόριθμος ορίζεται αναδρομικά όταν μέσα στον ορισμό του εμφανίζεται ο ίδιος ο εαυτός του. Για παράδειγμα, το παραγοντικό μπορεί να οριστεί αναδρομικά με τον τρόπο που φαίνεται στην Εικόνα 8.22.

$$\text{Παραγοντικό}\,(n) = \begin{cases} 1 & \text{αν } n = 0 \\ n \times \text{Παραγοντικό}\,(n-1) & \text{αν } n > 0 \end{cases}$$

Εικόνα 8.22 Αναδρομικός ορισμός του παραγοντικού



Εικόνα 8.23 Πορεία της αναδρομικής λύσης για το πρόβλημα του παραγοντικού

Επαναληπτική λύση

Η λύση αυτή συνήθως χρησιμοποιεί έναν βρόχο.

Αλγόριθμος 8.6 **Επαναληπτική λύση στο πρόβλημα του παραγοντικού**

```
Αλγόριθμος: Παραγοντικό (n)
Σκοπός: Εύρεση του παραγοντικού ενός αριθμού με τη χρήση ενός βρόχου
Προ-συνθήκη: Δίνεται το n
Μετα-συνθήκη: Τίποτα
Επιστρέφεται: n!
{
        F ← 1
        i ← 1
        όσο (i ≤ n)
        {
                F ← F × i
                i ← i + 1
        }
        επιστροφή F
}
```


Αναδρομική λύση

Στην αναδρομική λύση δεν χρειάζεται βρόχος επειδή η επανάληψη είναι ενσωματωμένη στην ίδια την έννοια.

Αλγόριθμος 8.7 **Ψευδοκώδικας για την αναδρομική λύση του προβλήματος παραγοντικού**

```
Αλγόριθμος: Παραγοντικό (n)
Σκοπός: Εύρεση του παραγοντικού ενός αριθμού με τη χρήση αναδρομής
Προ-συνθήκη: Δίνεται το n
Μετα-συνθήκη: Τίποτα
Επιστρέφεται: n!
{
        αν (n = 0)      επιστροφή 1
        αλλιώς          επιστροφή n × Παραγοντικό (n –1)
}
```