Κεφάλαιο 10

ΤΑΞΙΝΟΜΗΣΗ

10.1 Εισαγωγή

Στο κεφάλαιο αυτό θα εξετασθεί το πρόβλημα της τοποθέτησης ενός συνόλου κόμβων ή εγγραφών σε μία ιδιαίτερη σειρά. Η σειρά αυτή (συνήθως) είναι η **αύξουσα τάξη** (ascending sequence) μεγέθους του (πρωτεύοντος) κλειδιού της εγγραφής. Το πρόβλημα αυτό είναι γνωστό ως **ταξινόμηση** (sorting) ή **διάταξη** (ordering). Σκοπός της ταξινόμησης είναι η διευκόλυνση της αναζήτησης των εγγραφών του ταξινομηθέντος συνόλου. Για παράδειγμα, από το Κεφάλαιο 8 είναι γνωστό ότι η αναζήτηση ενός κλειδιού σε μη ταξινομημένο πίνακα με n εγγραφές μπορεί να γίνει σειριακά εκτελώντας συγκρίσεις της τάξης O(n), ενώ σε ταξινομημένο πίνακα η δυαδική αναζήτηση και η αναζήτηση παρεμβολής εκτελούν συγκρίσεις της τάξης O(log$n$) και O(loglog$n$), αντίστοιχα. Η χρησιμότητα της ταξινόμησης αποδεικνύεται στην πράξη σε περιπτώσεις αναζήτησης αριθμητικών ή αλφαβητικών δεδομένων, όπως σε βιβλιοθηκονομικά συστήματα, λεξικά, τηλεφωνικούς καταλόγους, καταλόγους φόρου εισοδήματος και γενικά παντού όπου γίνεται αναζήτηση αποθηκευμένων αντικείμενων.

Πέρα όμως από την τεράστια πρακτική χρησιμότητα της ταξινόμησης το θέμα παρουσιάζει και πολύ μεγάλο θεωρητικό ενδιαφέρον γιατί έχει αναπτυχθεί πληθώρα αλγορίθμων ταξινόμησης. Στο κεφάλαιο αυτό θα περιγραφούν αρκετοί τέτοιοι αλγόριθμοι που θα δείξουν:

- ότι η εκλογή μίας συγκεκριμένης δομής δεδομένων επηρεάζει τους αλγορίθμους που εκτελούν ένα δοσμένο έργο,

- ότι η επιλογή του κατάλληλου αλγορίθμου είναι μία δύσκολη διαδικασία που μπορεί να διευκολυνθεί με την ανάλυση της επίδοσής τους (performance analysis), και

- ότι υπάρχει μιά θεωρητικά βέλτιστη επίδοση, που κανείς αλγόριθμος ταξινόμησης δεν μπορεί να ξεπεράσει.

Στη συνέχεια δίνεται ένας τυπικός ορισμός της ταξινόμησης.

Ορισμός.

Δοθέντων των στοιχείων $a_1, a_2, ..., a_n$ η ταξινόμηση συνίσταται στη μετάθεση (permutation) της θέσης των στοιχείων, ώστε να τοποθετηθούν σε μία σειρά $a_{k_1}, a_{k_2}, ..., a_{k_n}$ έτσι ώστε, δοθείσης μίας **συνάρτησης διάταξης** (ordering function), f, να ισχύει:

$$f(a_{k_1}) \leq f(a_{k_2}) \leq ... \leq f(a_{k_n}) \qquad \square$$

Αξίζει να σημειωθεί ότι η προηγούμενη συνάρτηση διάταξης μπορεί να τροποποιηθεί, ώστε να καλύπτει και την περίπτωση που η ταξινόμηση γίνεται με **φθίνουσα τάξη** (descending sequence) μεγέθους του κλειδιού. Στην πιο γενική περίπτωση μπορεί να θεωρηθεί ότι η ταξινόμηση στηρίζεται σε δύο ή περισσότερα κλειδιά της εγγραφής. Για παράδειγμα, σε πολλά παιχνίδια της τράπουλας οι παίκτες ταξινομούν τα χαρτιά τους με πρώτο κλειδί το χρώμα του φύλλου (πχ. μπαστούνια, σπαθιά, καρά και κούπες) και με δεύτερο κλειδί την αξία του φύλλου (πχ. Ασσος, Ρήγας, Ντάμα, Βαλές, 10,..., 2).

Το κεφάλαιο αυτό έχει την εξής διάρθρωση. Στο Κεφάλαιο 10.2 γίνεται μία γενική εισαγωγή στα βασικά χαρακτηριστικά των μεθόδων ταξινόμησης πινάκων, που θα εξετασθούν με κάθε λεπτομέρεια στα επόμενα κεφάλαια. Το Κεφάλαιο 10.6 απαντά στο ενδιαφέρον ερώτημα: "Πόσο γρήγορα μπορεί να γίνει η ταξινόμηση", ενώ στα Κεφάλαια 10.7 ως 10.9 περιγράφονται οι πιο αποδοτικές μέθοδοι εσωτερικής ταξινόμησης. Στα Κεφάλαια 10.10 και 10.11 εξετάζονται συμπληρωματικές μέθοδοι ταξινόμησης που έχουν αυξημένες απαιτήσεις σε χώρο μνήμης, ενώ το Κεφάλαιο 10.12 γίνεται μία τελική συνόψιση των μεθόδων ταξινόμησης. Ο αναγνώστης επίσης παροτρύνεται να εξετάσει πολλές από τις ασκήσεις του κεφαλαίου, γιατί αφορούν μεθόδους που δεν έχουν παρουσιασθεί προηγουμένως.

10.2 Ταξινόμηση πινάκων

Οι μέθοδοι ταξινόμησης χωρίζονται στις **εσωτερικές** (internal) και στις **εξωτερικές** (external) μεθόδους. Οι κατηγορίες αυτές αντιστοιχούν στην ταξινόμηση πινάκων ή σειριακών αρχείων (sequential files) που είναι αποθηκευμένα στη γρήγορη και άμεσης προσπέλασης κύρια μνήμη ή στις αργότερες περιφερειακές συσκευές (όπως ταινίες, δίσκους), αντίστοιχα. Για παράδειγμα, η ταξινόμηση των συνδρομητών του ΟΤΕ σε μία κωμόπολη μπορεί να χρησιμοποιήσει μία από τις εσωτερικές μεθόδους, εφ' όσον φυσικά οι εγγραφές των συνδρομητών χωρούν στην κεντρική μνήμη. Δεν ισχύει το ίδιο όμως για τους συνδρομητές ενός νομού που είναι αποθηκευμένοι σε αρχείο μαγνητικού δίσκου, οπότε για

την ταξινόμησή τους θα χρησιμοποιηθεί κάποια από τις μεθόδους εξωτερικής ταξινόμησης, που εξετάζονται στο βιβλίο για Δομές Αρχείων.

Στην ταξινόμηση πινάκων υπάρχει η απαίτηση για όσο το δυνατόν περισσότερο οικονομική χρήση της μνήμης. Για όλους τους αλγορίθμους ταξινόμησης που θα παρουσιασθούν στα επόμενα κεφάλαια προϋποτίθενται οι ακόλουθες εντολές για τις δηλώσεις των τύπων και μεταβλητών.

```
TYPE index=0..n;
    item=RECORD
            key: INTEGER                    (* και άλλα δεδομένα *)
         END;
VAR table:ARRAY[1..n] OF item;
```

Στις δηλώσεις αυτές κάθε στοιχείο (item) περιγράφεται με μία εγγραφή που αποτελείται από το "κλειδί" και από "άλλα δεδομένα", που σχετίζονται με το στοιχείο. Φυσικά η επιλογή του τύπου INTEGER για το κλειδί είναι αυθαίρετη, αφού μπορεί να χρησιμοποιηθεί οποιοσδήποτε άλλος τύπος, όπου ορίζεται μία συνάρτηση διάταξης, όπως για παράδειγμα ο τύπος του χαρακτήρα. Επίσης, οι διαδικασίες χρησιμοποιούν τη μεταβλητή index ως υποσύνολο των ακεραίων αριθμών. Οι διαδικασίες που θα δοθούν στα επόμενα κεφάλαια θεωρούν ότι ως είσοδος δίνεται η μεταβλητή table με n στοιχεία που πρέπει να ταξινομηθούν. Στα Κεφάλαια 10.3 ως 10.9 θα εξετασθούν μέθοδοι που για την ταξινόμηση του πίνακα table δεν χρησιμοποιούν κάποιον άλλο βοηθητικό πίνακα όμοιου τύπου, αλλά εκτελούν την ταξινόμηση κάνοντας χρήση των **ίδιων θέσεων** (in situ, in place) του πίνακα table. Έτσι συχνότατα, όπως θα φανεί σε πολλές διαδικασίες, απαιτείται να γίνει ανταλλαγή του περιεχομένου δύο θέσεων του πίνακα. Η διαδικασία Swap που περιγράφεται στο Κεφάλαιο 5.10 εκτελεί τη λειτουργία αυτή. Στα Κεφάλαια 10.10 και 10.11 θα εξετασθούν αλγόριθμοι ταξινόμησης, που απαιτούν επιπλέον θέσεις μνήμης.

Μία μέθοδος λέγεται **ευσταθής** (stable), αν η σχετική διάταξη των στοιχείων με ίσα κλειδιά παραμένει αμετάβλητη με την εφαρμογή της μεθόδου. Ένας άλλος διαχωρισμός των μεθόδων γίνεται με βάση την αποτελεσματικότητά τους. Το πιο βασικό κριτήριο της επίδοσης μίας μεθόδου είναι ο αριθμός C, που μετρά τις απαιτούμενες **συγκρίσεις κλειδιών** (key comparisons), που εκτελούνται ώστε να γίνει η ταξινόμηση. Ένα άλλο κριτήριο είναι ο αριθμός που μετρά τις **μετακινήσεις** (moves) των στοιχείων. Οι αριθμοί C και M είναι συναρτήσεις του αριθμού n των στοιχείων, που πρέπει να ταξινομηθούν.

Οι μέθοδοι ταξινόμησης χωρίζονται με βάση την επίδοσή τους σε δύο μεγάλες κατηγορίες:

- στις μεθόδους με επίδοση τάξης O(n^2), που ονομάζονται **ευθείες** (straight) μέθοδοι, και

- στις μεθόδους με επίδοση τάξης O(n log$n$).

Παρά την ύπαρξη των γρήγορων μεθόδων, στα επόμενα τρία κεφάλαια θα εξετασθούν οι απλές και προφανείς τεχνικές ταξινόμησης, που απαιτούν χρόνο εκτέλεσης τάξης O(n^2). Οι ευθείες μέθοδοι παρουσιάζονται πριν από τους γρήγορους αλγορίθμους για τους εξής τέσσερις βασικούς λόγους:

- τα προγράμματά τους είναι μικρά και ευνόητα,
- αν και οι προηγμένες μέθοδοι απαιτούν λιγότερες λειτουργίες, αυτές οι λειτουργίες είναι συνήθως πιο πολύπλοκες στις λεπτομέρειές τους,
- οι ευθείες μέθοδοι είναι ταχύτερες για μικρά n, ενώ είναι απογοητευτικές για μεγάλα n, και
- οι ευθείες μέθοδοι προσφέρονται ιδιαίτερα για τη διευκρίνιση των βασικών λειτουργικών χαρακτηριστικών της ταξινόμησης.

Οι μέθοδοι ταξινόμησης πινάκων που χρησιμοποιούν τις ίδιες θέσεις χωρίζονται σε τρεις βασικές κατηγορίες ανάλογα με τη βασική αρχή λειτουργίας τους:

- ταξινομήσεις **εισαγωγής** (insertion),
- ταξινομήσεις **ανταλλαγής** ή **μετάθεσης** (exchanging, transposition), και
- ταξινομήσεις **επιλογής** (selection),

ενώ οι μέθοδοι ταξινόμησης που δεν χρησιμοποιούν τις ίδιες θέσεις χωρίζονται στις εξής δύο κατηγορίες:

- ταξινομήσεις **συγχώνευσης** (merging), και
- ταξινομήσεις **κατανομής** (distribution).

Η κατάταξη όλων των μεθόδων στις πέντε κατηγορίες ανάλογα με το βασικό χαρακτηριστικό τους έχει προταθεί από τον Knuth (1973) στον τρίτο τόμο του βιβλίου του. Ωστόσο πρόσφατα προτάθηκε ένα διαφορετικό σκεπτικό κατάταξης των μεθόδων ταξινόμησης που θα παρουσιασθεί στο Κεφάλαιο 10.12. Ο ενδιαφερόμενος αναγνώστης για περισσότερα στοιχεία μπορεί να ανατρέξει σε άλλες σημαντικές πηγές, όπως είναι οι επισκοπήσεις του Lorin (1971), του Martin (1971), του Bentley (1979) και του Erkio (1981), καθώς και το βιβλίο του Lorin (1975). Στη συνέχεια θα εξετασθούν οι ευθείες μέθοδοι των τριών πρώτων κατηγοριών.

10.3 Ταξινόμηση ευθείας εισαγωγής

Σύμφωνα με την **ταξινόμηση ευθείας εισαγωγής** (straight inserion sort) τα στοιχεία χωρίζονται 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]. Σε κάθε βήμα, αρχίζοντας με i=2 και αυξάνοντας διαδοχικά το i κατά ένα, το στοιχείο με δείκτη i λαμβάνεται και μεταφέρεται στην κατάλληλη θέση της ακολουθίας προορισμού. Αυτή η μέθοδος χρησιμοποιείται πολύ από τους χαρτοπαίκτες. Η δομή του αλγορίθμου της ευθείας εισαγωγής δίνεται στη συνέχεια.

```
FOR i:=2 TO n DO
    BEGIN
       x:=table[i];
       Παρέμβαλε το x στην κατάλληλη θέση
          μεταξύ των table[1],..., table[i-1]
    END;
```

Παράδειγμα.
'Εστω ότι ο αρχικός πίνακας αποτελείται από τα εννέα κλειδιά 52, 12, 71, 56, 5, 10, 19, 90 και 45. Στο Σχήμα 10.1 φαίνεται η διαδικασία ταξινόμησης θεωρώντας τον πίνακα αυτό. Η εύρεση της κατάλληλης θέσης γίνεται εύκολα με διαδοχικές συγκρίσεις και μετακινήσεις. Δηλαδή, το στοιχείο x συγκρίνεται με τα προηγούμενα στοιχεία table[i-1], table[i-2] κλπ. (προς τα αριστερά) μέχρι να βρεθεί κάποιο στοιχείο table[k] με τιμή μικρότερη από το x. Κατόπιν τα κλειδιά από table[i-1] ως table[k+1] μετακινούνται προς τα δεξιά και το x καταλαμβάνει τη σωστή θέση στην ταξινομημένη ακολουθία. □

Αρχικά κλειδιά	52	12	71	56	5	10	19	90	45
i=2	12	52	71	56	5	10	19	90	45
i=3	12	52	71	56	5	10	19	90	45
i=4	12	52	56	71	5	10	19	90	45
i=5	5	12	52	56	71	10	19	90	45
i=6	5	10	12	52	56	71	19	90	45
i=7	5	10	12	19	52	56	71	90	45
i=8	5	10	12	19	52	56	71	90	45
i=9	5	10	12	19	45	52	56	71	90

Σχήμα 10.1: Ταξινόμηση ευθείας εισαγωγής.

Η διαδικασία των διαδοχικών συγκρίσεων προς τα αριστερά μπορεί να συνεχισθεί μέχρι να γίνει η σύγκριση του στοιχείου table[1] που μπορεί να έχει τιμή μεγαλύτερη από την τιμή του x. Αυτή η χαρακτηριστική περίπτωση της

επανάληψης με δύο συνθήκες τερματισμού υπενθυμίζει τη γνωστή τεχνική της χρήσης του στοιχείου φρουρού, που επιταχύνει τη διαδικασία της σειριακής αναζήτησης (δες Κεφάλαιο 3.5.1). Αν το στοιχείο table[0] παίξει το ρόλο του στοιχείου φρουρού, τότε αυτή η δήλωση του πίνακα πρέπει να μετατραπεί σε "table:ARRAY[index] OF item;". Με την προϋπόθεση αυτή ο ακόλουθη διαδικασία εκτελεί την ταξινόμηση ευθείας εισαγωγής.

```
PROCEDURE StraightInsertion;
VAR i,j:index; x:item;
BEGIN
   FOR i:=2 TO n DO
      BEGIN
         x:=table[i]; table[0]:=x; j:=i-1;
         WHILE x.key<table[j].key DO
            BEGIN
               table[j+1]:=table[j]; j:=j-1
            END;
         table[j+1]:=x
      END
END;
```

Για την ανάλυση της ευθείας εισαγωγής ας υποτεθεί ότι όλες οι δυνατές διατάξεις των n κλειδιών είναι ισοπίθανες. Θεωρώντας τη συγκεκριμένη υλοποίηση με τον κόμβο φρουρό προκύπτει η επόμενη πρόταση. Αν η υλοποίηση είναι διαφορετική, τότε οι επόμενες σχέσεις δεν ισχύουν επακριβώς αλλά μόνο προσεγγιστικά.

Πρόταση.
Οι ελάχιστες, οι μέσες και οι μέγιστες τιμές του αριθμού των συγκρίσεων και των μετακινήσεων κατά την ταξινόμηση ευθείας εισαγωγής είναι:

$$C_{min} = n-1 \qquad C_{ave} = \frac{n^2+n-2}{4} \qquad C_{max} = \frac{n^2+n}{2} - 1$$

$$M_{min} = 2(n-1) \quad M_{ave} = \frac{n^2+9n-10}{4} \quad M_{max} = \frac{n^2+3n-4}{2}$$

Απόδειξη.
Αν C_i είναι ο αριθμός των συγκρίσεων των κλειδιών (όπου $2 \leq i \leq n$), τότε η μέγιστη τιμή του C_i είναι i-1, η ελάχιστη τιμή είναι 1 και η μέση τιμή είναι $i/2$. Ο αριθμός των $_i$ των μετακινήσεων είναι C_i+2. Αν θεωρηθεί το σύνολο των στοιχείων και γίνει άθροιση ως προς i, τότε προκύπτουν οι σχέσεις αυτές. □

Ευνόητο είναι ότι οι ελάχιστες τιμές συμβαίνουν όταν τα στοιχεία είναι ήδη ταξινομημένα. Επίσης η χειρότερη περίπτωση συμβαίνει όταν τα στοιχεία

δίνονται αρχικά στην αντίστροφη σειρά. Με την έννοια αυτή, η ταξινόμηση με εισαγωγή επιδεικνύει μία φυσιολογική συμπεριφορά. Επίσης είναι σαφές ότι ο αλγόριθμος αυτός είναι μία ευσταθής μέθοδος ταξινομήσης, δηλαδή αφήνει αμετάβλητη μία σειρά στοιχείων με ίσα κλειδιά.

Ο προηγούμενος αλγόριθμος μπορεί εύκολα να βελτιωθεί επειδή η ακολουθία προορισμού, όπου το νέο στοιχείο πρέπει να εισαχθεί, είναι ήδη ταξινομημένη. Μπορεί επομένως να χρησιμοποιηθεί, μία ταχύτερη μέθοδος για την εύρεση του σημείου εισαγωγής. Η προφανής εκλογή είναι η μέθοδος της δυαδικής αναζήτησης. Δηλαδή, η μέθοδος ελέγχει την ακολουθία προορισμού στο μέσον και συνεχίζει να διχοτομεί μέχρι να βρεθεί το σημείο εισαγωγής. Ο τροποποιημένος αλγόριθμος ονομάζεται **ταξινόμηση δυαδικής εισαγωγής** (binary insertion sort) και έχει την ακόλουθη μορφή.

```
PROCEDURE BinaryInsertion;
VAR left,right,mid,i,j:index; x:item;
BEGIN
   FOR i:=2 TO n DO
      BEGIN
         x:=table[i]; left:=1; right:=i-1;
         WHILE left<=right DO
            BEGIN
               mid:=(left+right) DIV 2;
               IF x.key<table[mid].key THEN right:=mid-1
               ELSE left:=mid+1
            END;
         FOR j:=i-1 DOWNTO left DO table[j+1]:=table[j];
         table[left]:=x
      END
END;
```

Πρόταση.
Η μέση τιμή και η χειρότερη τιμή του αριθμού των συγκρίσεων κατά την ταξινόμηση με δυαδική εισαγωγή είναι της τάξης O(n log$n$).

Απόδειξη.
Η εισαγωγή γίνεται στο σημείο όπου ισχύει table[left].key $\leq$ x.key $\leq$ table[right].key. Έτσι το διάστημα ανίχνευσης πρέπει στο τέλος να είναι 1. Αυτό συνεπάγεται ότι το διάστημα των i κλειδιών διχοτομείται στη χειρότερη περίπτωση $\lceil \log i \rceil$ φορές. Άρα, ισχύει:

$$C = \sum_{i=1}^{n-1} \lceil \log i \rceil$$

Το άθροισμα προσεγγίζεται με το ολοκλήρωμα:

$$C \approx \int_{1/2}^{n-1/2} \log x \mathrm{d}x = \frac{1}{\ln 2} \int_{1/2}^{n-1/2} \ln x \mathrm{d}x = \frac{1}{\ln 2} (x \ln x - x) \begin{matrix} n-1/2 \\ 1/2 \end{matrix}$$

Με επίλυση του ολοκληρώματος προκύπτει εύκολα η αλήθεια της πρότασης. Η αντιμετώπιση της μέσης περίπτωσης είναι πανομοιότυπη. □

Σε αντίθεση με την προηγούμενη μέθοδο αξίζει να σημειωθεί ότι ο ελάχιστος αριθμός συγκρίσεων αναμένεται όταν τα στοιχεία είναι αρχικά στην αντίστροφη σειρά και ο μέγιστος όταν είναι ήδη ταξινομημένα. Ο αναγνώστης καλείται να αιτιολογήσει την αφύσικη αυτή συμπεριφορά του αλγορίθμου.

Δυστυχώς, με την εφαρμογή της δυαδικής αναζήτησης ελαττώνεται μόνο ο αριθμός των συγκρίσεων και όχι ο αριθμός των μετακινήσεων. Πράγματι, αφού η μετακίνηση στοιχείων (δηλαδή των κλειδιών και των αντίστοιχων δεδομένων) είναι γενικά σημαντικά περισσότερο χρονοβόρα από τη σύγκριση δύο κλειδιών, η βελτίωση δεν είναι ικανοποιητική. Αυτός είναι ο κυριότερος λόγος που οι μέθοδοι αυτές δεν είναι αρκετά κατάλληλες για επαγγελματικές εφαρμογές. Ωστόσο, αξίζει τον κόπο να τονισθεί ότι η μέθοδος είναι από τις αποτελεσματικότερες όταν ο αριθμός των στοιχείων είναι μικρός, δηλαδή μικρότερος από 10 ως 15 κλειδιά το μέγιστο. Στη συνέχεια θα γίνει και πάλι συζήτηση για τη μέθοδο αυτή, γιατί χρησιμοποιείται σε υβριδικά σχήματα (δες Κεφάλαια 10.7, 10.8 και 10.11).

10.4 Ταξινόμηση ευθείας ανταλλαγής

Ο καθορισμός του βασικού χαρακτηριστικού μίας μεθόδου ταξινόμησης δεν είναι πάντοτε σαφής. Θα μπορούσε να υποστηριχθεί ότι η προηγούμενη μέθοδος στηρίζεται στις ανταλλαγές. Ωστόσο στη συνέχεια εξετάζεται μία μέθοδος που έχει ως βασικό χαρακτηριστικό την ανταλλαγή των στοιχείων. Η μέθοδος αυτή λέγεται **ταξινόμηση ευθείας ανταλλαγής** (straight exchange sort) και βασίζεται στην αρχή της σύγκρισης και ανταλλαγής ζευγών από γειτονικά στοιχεία, μέχρις ότου διαταχθούν όλα τα στοιχεία.

```
PROCEDURE BubbleSort;
VAR i,j:index; x:item;
BEGIN
   FOR i:=2 TO n DO
      FOR j:=n DOWNTO i DO
         IF table[j-1].key>table[j].key THEN Swap(table[j-1],table[j])
END;
```

Σύμφωνα με τη μέθοδο αυτή κάθε φορά γίνονται διαδοχικές προσπελάσεις στον πίνακα και μετακινείται το μικρότερο κλειδί της ακολουθίας προς το αρι-

στερό άκρο του πίνακα. Αν ο πίνακας θεωρηθεί σε κατακόρυφη θέση αντί σε οριζόντια και τα στοιχεία θεωρηθούν - επιστρατεύοντας αρκετή φαντασία - ως φυσσαλίδες (bubbles) σε μία δεξαμενή νερού με βάρη σύμφωνα με την τιμή του κλειδιού τους, τότε κάθε προσπέλαση στον πίνακα έχει ως αποτέλεσμα την άνοδο της φυσσαλίδας στο κατάλληλο επίπεδο βάρους. Η μέθοδος είναι γνωστή ως **ταξινόμηση φυσσαλίδας** (bubblesort) και υλοποιείται με την προηγούμενη διαδικασία.

Αρχικά κλειδιά	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9
52	5	5	5	5	5	5	5	5
12	52	10	10	10	10	10	10	10
71	12	52	12	12	12	12	12	12
56	71	12	52	19	19	19	19	19
5	56	71	19	52	45	45	45	45
10	10	56	71	45	52	52	52	52
19	19	19	56	71	56	56	56	56
90	45	45	45	56	71	71	71	71
45	90	90	90	90	90	90	90	90

Σχήμα 10.2: Ταξινόμηση φυσσαλίδας.

Παράδειγμα.
Η μέθοδος εφαρμοζόμενη στα γνωστά εννέα κλειδιά του προηγούμενου παραδείγματος εξελίσσεται όπως φαίνεται στο Σχήμα 10.2. □

Ο αλγόριθμος αυτός δέχεται αρκετές βελτιώσεις. Το παράδειγμα στο Σχήμα 10.2 δείχνει ότι τα τελευταία πέντε περάσματα δεν έχουν κανένα αποτέλεσμα στη σειρά των στοιχείων αφού τα στοιχεία είναι ήδη ταξινομημένα. Μία προφανής τεχνική για τη βελτίωση αυτού του αλγορίθμου γίνεται με τη χρήση μίας λογικής μεταβλητής, που δείχνει αν συνέβη κάποια αλλαγή κατά τη διάρκεια της τελευταίας προσπέλασης. Έτσι απαιτείται μία επιπλέον προσπέλαση χωρίς άλλες λειτουργίες ανταλλαγής μέχρι ο αλγόριθμος να τερματισθεί. Είναι δυνατόν να επιτευχθεί επιπλέον βελτίωση αν χρησιμοποιηθεί μία ακόμη μεταβλητή όπου να αποθηκεύεται η τελευταία τιμή του δείκτη, έστω k, όπου έγινε η τελευταία ανταλλαγή. Για παράδειγμα, είναι σαφές ότι όλα τα ζεύγη των γειτονικών στοιχείων επάνω από το δείκτη k είναι στην επιθυμητή σειρά και επομένως ο έλεγχος μπορεί να τερματίσει στο δείκτη k αντί να προχωρήσει στο προκαθορισμένο όριο i. Ο ενδιαφερόμενος αναγνώστης παραπέμπεται στην επισκόπηση του Maresh (1985), όπου όλες οι παραλλαγές της ταξινόμησης φυσσαλίδας παρουσιάζονται και αναλύονται.

Στο σημείο αυτό σημειώνεται μία ασυμμετρία της βελτιωμένης μεθόδου όταν πρόκειται να ταξινομηθεί ένας σχεδόν ταξινομημένος πίνακας. Για παρά-

δειγμα, με τη μέθοδο αυτή ο πίνακας:

$$\{\ 10 \quad 12 \quad 19 \quad 45 \quad 52 \quad 56 \quad 71 \quad 90 \quad 5\ \}$$

θα ταξινομηθεί με ένα μόνο πέρασμα, ενώ η ταξινόμηση του πίνακα:

$$\{\ 90 \quad 5 \quad 10 \quad 12 \quad 19 \quad 45 \quad 52 \quad 56 \quad 71\ \}$$

απαιτεί οκτώ περάσματα. Η παρατήρηση αυτή οδηγεί σε μία ακόμη περισσότερο βελτιωμένη έκδοση της μεθόδου, που στα αγγλικά ονομάζεται cocktail shakersort. Το χαρακτηριστικό αυτής της μεθόδου είναι ότι κατά τα διαδοχικά περάσματα η διεύθυνση σάρωσης της ακολουθίας εναλλάσσεται.

Παράδειγμα.
Στο Σχήμα 10.3 φαίνεται η λειτουργία της μεθόδου για τα ίδια κλειδιά. Οι δείκτες low και high δείχνουν τα όρια μέσα στον υποπίνακα, όπου θα περιορισθούν οι ανταλλαγές κατά το επόμενο πέρασμα. 'Ετσι τώρα τα απαιτούμενα περάσματα είναι τέσσερα αντί οκτώ. □

high=	2	3	3	4	4	6	6
low=	9	9	7	7	6	6	5
	52	5	5	5	5	5	5
	12	52	12	10	10	10	10
	71	12	52	12	12	12	12
	56	71	56	52	52	19	19
	5	56	10	56	19	52	45
	10	10	19	19	45	45	52
	19	19	45	45	56	56	56
	90	45	71	71	71	71	71
	45	90	90	90	90	90	90

Σχήμα 10.3: Ταξινόμηση με τη μέθοδο shakersort.

Η μέθοδος υλοποιείται με την ακόλουθη διαδικασία ShakerSort, που στηρίζεται στη διαδικασία BubbleSort. Πιο συγκεκριμένα λαμβάνεται υπ' όψη η θέση, όπου έγινε η τελευταία ανταλλαγή του προηγούμενου περάσματος χωρίς να χρησιμοποιείται μία λογική σημαία, που να δηλώνει αν στο τελευταίο πέρασμα έγιναν ανταλλαγές.

```
PROCEDURE ShakerSort;
VAR high,low,j,k:index; x:item;
BEGIN
    high:=2; low:=n; k:=n;
      REPEAT
        FOR j:=low DOWNTO high DO
```

```
            IF table[j-1].key>table[j].key THEN
               BEGIN
                  Swap(table[j-1],table[j]); k:=j
               END;
         high:=k+1;
         FOR j:=high TO low DO
            IF table[j-1].key>table[j].key THEN
               BEGIN
                  Swap(table[j-1],table[j]); k:=j
               END;
         low:=k-1;
      UNTIL high>low
END;
```

Σχετικά με τις επιδόσεις των παραλλαγών αυτών αναφέρονται οι εξής προτάσεις.

Πρόταση.
Η ελάχιστη, η μέση και η μέγιστη τιμή του αριθμού των συγκρίσεων κατά την ταξινόμηση ευθείας επιλογής είναι:

$$C = \frac{n^2 - n}{2}$$

Απόδειξη
Η ταύτιση των τριών τιμών (καλύτερη, χειρότερη και μέση) προκύπτει από το γεγονός ότι ο αριθμός των συγκρίσεων των κλειδιών είναι ανεξάρτητος της αρχικής σειράς των κλειδιών. Με την έννοια αυτή η μέθοδος συμπεριφέρεται λιγότερο φυσιολογικά από την ευθεία εισαγωγή. □

Πρόταση.
Η ελάχιστη, η μέση και η μέγιστη τιμή του αριθμού των μετακινήσεων κατά την ταξινόμηση ευθείας επιλογής είναι αντίστοιχα:

$$M_{min} = 0 \qquad M_{ave} = \frac{3(n^2 - n)}{4} \qquad M_{max} = \frac{3(n^2 - n)}{2}$$ □

Πρόταση.
Η ελάχιστη και η μέση τιμή του αριθμού των συγκρίσεων κατά την ταξινόμηση shakersort είναι:

$$C_{min} = n - 1 \qquad C_{ave} = \frac{n^2 - n(\ln n + c)}{2}$$

όπου το c είναι σταθερά. □

Από τις αναλύσεις αυτές φαίνεται ότι οι μέθοδοι αυτές δεν είναι αποτελεσματικές και για το λόγο αυτό αποφεύγεται η χρήση τους σε επαγγελματικές εφαρμογές.

10.5 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 table.

Αυτές οι λειτουργίες επαναλαμβάνονται για τα υπόλοιπα n-1 στοιχεία, μέχρι στο τέλος να απομείνει μόνο το μεγαλύτερο στοιχείο. Η δομή της διαδικασίας που αντιστοιχεί στη μέθοδο αυτή δίνεται στη συνέχεια.

```
FOR i:=1 TO n-1 DO
    BEGIN
        Αναζήτησε το δείκτη k του ελάχιστου στοιχείου μεταξύ των
            table[i],...,table[n]. Αντάλλαξε τα στοιχεία table[i] και table[k]
    END;
```

Παράδειγμα.
Η μέθοδος αυτή παρουσιάζεται στο Σχήμα 10.4 καθώς εφαρμόζεται στα ίδια γνωστά εννέα κλειδιά. □

Αρχικά κλειδιά	52	12	71	56	5	10	19	90	45
i=1	5	12	71	56	52	10	19	90	45
i=2	5	10	71	56	52	12	19	90	45
i=3	5	10	12	56	52	71	19	90	45
i=4	5	10	12	19	52	71	56	90	45
i=5	5	10	12	19	45	71	56	90	52
i=6	5	10	12	19	45	52	56	90	71
i=7	5	10	12	19	45	52	56	90	71
i=8	5	10	12	19	45	52	56	71	90

Σχήμα 10.4: Ταξινόμηση ευθείας επιλογής.

Η μέθοδος που περιγράφηκε προηγουμένως ονομάζεται **ταξινόμηση ευθείας επιλογής** (straight selection sort) και μπορεί να θεωρηθεί ως αντίθετη της ευθείας εισαγωγής. Πιο συγκεκριμένα, η ευθεία εισαγωγή θεωρεί σε κάθε βήμα αφ' ενός το ένα και μοναδικό 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. Η διαδικασία StraightSelection που ακολουθεί υλοποιεί τη μέθοδο της ευθείας επιλογής.

```
PROCEDURE StraightSelection;
VAR i,j,k:index; x:item;
BEGIN
    FOR i:=1 TO n-1 DO
       BEGIN
          k:=i; x:=table[i];
          FOR j:=i+1 TO n DO
             IF x.key>table[j].key THEN
                BEGIN
                   k:=j; x:=table[j]
                END;
          table[k]:=table[i]; table[i]:=x
       END
END;
```

Σχετικά με την επίδοση της συγκεκριμένης υλοποίησης της ταξινόμησης ευθείας επιλογής αναφέρονται οι εξής προτάσεις.

Πρόταση.

Η ελάχιστη, η μέση και η μέγιστη τιμή του αριθμού των συγκρίσεων κατά την ταξινόμηση ευθείας επιλογής είναι:

$$C = \frac{n^2 - n}{2}$$

Απόδειξη.

‘Οπως και στη μέθοδο της φυσσαλίδας η ταύτιση των τριών τιμών προκύπτει από το γεγονός ότι ο αριθμός των συγκρίσεων των κλειδιών είναι ανεξάρτητος της αρχικής σειράς των κλειδιών. □

Πρόταση.

Η ελάχιστη, η μέση και η μέγιστη τιμή του αριθμού των μετακινήσεων κατά την ταξινόμηση ευθείας επιλογής είναι αντίστοιχα:

$$_{min} = 3(n-1) \qquad _{ave} = \sum_{i=1}^{n} {}_i = n(\ln n + c) \qquad M_{max} = \frac{n^2}{4} + 3(n-1)$$

όπου το c είναι μία σταθερά.

Απόδειξη.

Στις προηγούμενες ποσότητες συμπεριλαμβάνεται τόσο ο αριθμός των μετακινήσεων του μικρότερου στοιχείου στην κατάλληλη θέση όσο και ο αριθμός των περιπτώσεων που κάθε φορά στη θέση x επιλέγεται για αποθήκευση το μικρότερο στοιχείο. Συνεπώς είναι ευνόητο ότι η ελάχιστη, η μέγιστη και η μέση τιμή αναφέρονται στην περίπτωση που τα κλειδιά είναι ταξινομημένα κατά την

σωστή, κατά την αντίστροφη φορά ή είναι τυχαία αντίστοιχα. Ο αριθμός των μετακινήσεων στη μέση περίπτωση μπορεί να προκύψει με τη εξής θεώρηση. 'Εστω ότι από i τυχαία στοιχεία πρέπει να επιλεγεί το μικρότερο. Λαμβάνεται, λοιπόν, αρχικά το πρώτο στοιχείο. Αυτό πρέπει να αντικατασταθεί από το δεύτερο με πιθανότητα 1/2, ή στη συνέχεια από το τρίτο με πιθανότητα 1/3 κοκ. 'Αρα ο αριθμός των μετακινήσεων είναι H_i-1, όπου $_i$ είναι ο γνωστός i-οστός αρμονικός αριθμός (δες Κεφάλαιο 3.5.2). Τέλος αθροίζοντας ως προς i (για $1 \leq i \leq n$-1) και ολοκληρώνοντας προκύπτει η σχέση της πρότασης ως μία πολύ καλή προσέγγιση. □

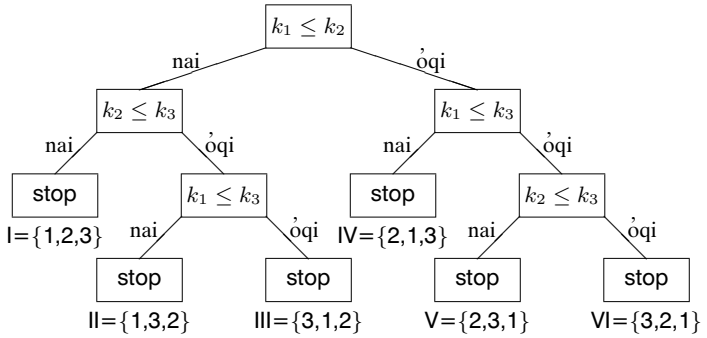
Από την προηγούμενη πρόταση συνάγεται ότι η μέθοδος είναι αρκετά ικανοποιητική ως προς τον αριθμό των μετακινήσεων. Ως γενικό συμπέρασμα προκύπτει ότι η μέθοδος ευθείας επιλογής πρέπει μάλλον να προτιμάται από τις προηγούμενες παρ' ότι για ταξινομημένα ή σχεδόν ταξινομημένα κλειδιά η ευθεία εισαγωγή είναι ταχύτερη. Στα Κεφάλαια 10.7, 10.8 και 10.9 θα παρουσιασθούν τρεις βελτιωμένες μέθοδοι για κάθε μία από τις τρεις βασικές μεθόδους ταξινόμησης: εισαγωγή, ανταλλαγή και επιλογή. Πριν από την παρουσίαση αυτή στο επόμενο κεφάλαιο θα απαντηθεί το πολύ ενδιαφέρον ερώτημα "Πόσο γρήγορα μπορεί να γίνει η ταξινόμηση;".

10.6 Κάτω όριο αλγορίθμων ταξινόμησης

Μέχρι τώρα εξετάσθηκαν τρεις μέθοδοι ταξινόμησης που στη μέση και στη χειρότερη περίπτωση απαιτούν χρόνο τάξης O(n^2). Πριν την ανάπτυξη άλλων αποτελεσματικότερων μεθόδων θα δοθεί απάντηση στο ερώτημα "Πόσο γρήγορα μπορεί μία μέθοδος να ταξινομήσει έναν πίνακα;". Θα αποδειχθεί ότι όταν μία μέθοδος ταξινόμησης στηρίζεται μόνο σε συγκρίσεις και ανταλλαγές των κλειδιών, τότε στη χειρότερη περίπτωση δεν μπορεί να ταξινομήσει σε χρόνο κατώτερο του O(n log$n$).

Πριν την απόδειξη της σχετική πρότασης είναι αναγκαίο να περιγραφεί η διαδικασία ταξινόμησης με ένα **δένδρο αποφάσεων** (decision tree). 'Ενα μονοπάτι του δένδρου δείχνει μία πιθανή διαδοχή από υπολογισμούς που ο αλγόριθμος πιθανώς να ακολουθήσει. Για παράδειγμα, ας υποτεθεί ότι πρέπει να ταξινομηθούν τρία κλειδιά: τα k_1, k_2 και k_3 αντίστοιχα. Η σειρά εισόδου των κλειδιών είναι k_1, k_2 και k_3 που φαίνεται στο δένδρο αποφάσεων ως (1,2,3). Αρχικά συγκρίνονται τα κλειδιά k_1 και k_2. Αν $k_1 < k_2$ τότε η σειρά (1,2,3) δεν αλλάζει, αλλιώς γίνεται (2,1,3).

Στο Σχήμα 10.5 φαίνονται όλα τα πιθανά ενδεχόμενα που μπορούν να συμβούν σε ένα αλγόριθμο ταξινόμησης ανάλογα με τη σειρά εισόδου των κλειδιών στο δένδρο. Από το Σχήμα 10.5 φαίνεται ότι το δένδρο αποφάσεων είναι ένα δυαδικό δένδρο. Τα φύλλα του δένδρου είναι αριθμημένα από I ως VI και



Σχήμα 10.5: Δένδρο αποφάσεων.

αποτελούν τα μοναδικά σημεία τερματισμού κάθε αλγορίθμου ταξινόμησης. Το δένδρο αυτό έχει 3!=6 φύλλα που εξασφαλίζει ότι ο αλγόριθμος βρίσκει πάντοτε τη διάταξη εκείνη που αντιστοιχεί στην ταξινομημένη σειρά.

Πρόταση.
Κάθε δένδρο αποφάσεων που ταξινομεί n διακεκριμένα κλειδιά έχει ύψος τουλάχιστον $\log(n!)$.

Απόδειξη.
'Ενα δένδρο αποφάσεων έχει $n!$ φύλλα που αντιστοιχούν σε κάθε μία από τις $n!$ διατάξεις των n κλειδιών. Ας υποτεθεί ότι το δένδρο είναι πλήρες και ότι το ύψος του είναι h. Τότε ο αριθμός των φύλλων είναι 2^{h-1}, οπότε συνεπάγεται ότι $n! = 2^{h-1}$. Λογαριθμώντας και τα δύο μέλη της σχέσης προκύπτει:

$$\log(n!) \;=\; \log(2^{h-1}) \;=\; h-1$$

Το δένδρο δεν είναι πλήρες και άρα η πρόταση ισχύει. □

Πρόταση.
Κάθε αλγόριθμος που ταξινομεί κάνοντας μόνο συγκρίσεις και ανταλλαγές κλειδιών έχει στην χειρότερη περίπτωση χρόνο τάξης $O(n \log n)$.

Απόδειξη.
Πρέπει να αποδειχθεί ότι σε κάθε δένδρο αποφάσεων με $n!$ φύλλα υπάρχει ένα μονοπάτι μήκους $C \times n \log n$, όπου C είναι μία σταθερά. Από την προηγούμενη πρόταση προκύπτει ότι κάθε μονοπάτι είναι μήκους $\log(n!)$. Αντικαθιστώντας με βάση τον τύπο του Stirling προκύπτει:

$$h \;=\; \log(n!) \;=\; \log(\sqrt{2\pi n}) + n\log n - n\log e \;=\; O(n\log n)$$

□

10.7 Ταξινόμηση με ελαττούμενες αυξήσεις

Η μέθοδος της **ταξινόμησης με ελαττούμενες αυξήσεις** (diminishing increment sort), που προτάθηκε από τον Shell (1959), έχει και αυτή ως βασικό χαρακτη-

ριστικό την εισαγωγή. Η ακόλουθη διαδικασία ShellSort υλοποιεί τη μέθοδο, που στη συνέχεια εξηγείται με ένα παράδειγμα για τα γνωστά εννέα κλειδιά.

```
PROCEDURE ShellSort;
VAR i,j,incr:index;
BEGIN
    incr:=n DIV 2;
    WHILE incr>0 DO
       BEGIN
          FOR i:=incr+1 TO n DO
             BEGIN
                j:=i-incr;
                WHILE j>0 DO
                   IF table[j].key>table[j+incr].key THEN
                      BEGIN
                         Swap(table[j],table[j+incr]); j:=j-incr
                      END
                   ELSE j:=0
             END;
          incr:=incr DIV 2
       END
END;
```

Παράδειγμα.

Στην αρχή θεωρούνται όλα τα στοιχεία που απέχουν μεταξύ τους τέσσερις θέσεις. Στο Σχήμα 10.6 φαίνονται όλες οι ανταλλαγές στοιχείων που απέχουν τη συγκεκριμένη απόσταση. 'Οταν δεν υπάρχει ζεύγος στοιχείων που απέχει τέσσερις θέσεις και χρειάζεται ανταλλαγή, τότε λέγεται ότι έχει γίνει **ταξινόμηση με βήμα (αύξηση) 4**. Μετά αυτό το πρώτο πέρασμα, όπως φαίνεται στο σχήμα ανταλλάσονται τα στοιχεία που απέχουν δύο θέσεις μεταξύ τους. 'Ετσι ολοκληρώνεται η ταξινόμηση με βήμα (αύξηση) 2. Τελικά, στο τρίτο πέρασμα ταξινομούνται τα στοιχεία που απέχουν βήμα (αύξηση) 1. □

Με βάση το παράδειγμα μπορεί να θεωρηθεί ότι αρχικά ο πίνακας διαιρείται σε τέσσερις διαπλεκόμενους υποπίνακες, που ο καθένας ταξινομείται ανεξάρτητα από τους άλλους με την ταξινόμηση της ευθείας εισαγωγής. Στη συνέχεια κατά παρόμοιο τρόπο ο πίνακας υποδιαιρείται σε δύο υποπίνακες, ώσπου στο τέλος εφαρμόζεται μία αμιγής ταξινόμηση ευθείας εισαγωγής. Το γεγονός ότι τα στοιχεία ταξινομούνται σε κάθε πέρασμα με τη μέθοδο ευθείας εισαγωγής δεν σημαίνει τελικά ότι η μέθοδος του Shell έχει την ίδια επίδοση με την αρχική μέθοδο. Απεναντίας είναι σαφώς πιο αποδοτική γιατί σε κάθε πέρασμα γίνονται λίγες συγκρίσεις κλειδιών, ενώ στο τελευταίο πέρασμα τα κλειδιά είναι σχεδόν ταξινομημένα από τα προηγούμενα περάσματα.

Αρχικά κλειδιά	52	12	71	56	5	10	19	90	45
	5				52				
					45				52
		10				12			
			19				71		
Βήμα 4	5	10	19	56	45	12	71	90	52
							52		71
				12		56			
Βήμα 2	5	10	19	12	45	56	52	90	71
			12	19					
						52	56		
								71	90
Βήμα 1	5	10	12	19	45	52	56	71	90

Σχήμα 10.6: Ταξινόμηση με ελαττούμενες αυξήσεις.

Η διαδικασία ShellSort εκτελεί τον αλγόριθμο με τη λογική αυτή. Αξίζει όμως να σημειωθούν οι εξής παρατηρήσεις:

- δεν είναι απαραίτητο να χρησιμοποιούνται ελαττούμενες αυξήσεις που είναι δυνάμεις του 2,
- οποιαδήποτε και αν είναι τα μεγέθη των αυξήσεων, μετά το τέλος του i-οστού πέρασματος, ο πίνακας παραμένει ταξινομημένος και σε σχέση με τις αυξήσεις των προηγούμενων i-1 περασμάτων, και
- στην πράξη όλες οι αυξήσεις μπορούν να εφαρμοσθούν, αρκεί όμως η τελευταία αύξηση να ισούται με 1.

Αν η ακολουθία των διαστημάτων είναι $h_1, h_2, ..., h_t$ και ισχύουν οι συνθήκες $h_t = 1$ και $h_{i+1} < h_i$, όπου $2 \leq i \leq t$, τότε η ταξινόμηση με βήμα h_k (για $1 \leq k \leq t$) μπορεί να υλοποιηθεί με την επόμενη διαδικασία ModifiedShellSort που χρειάζεται, βέβαια, ιδιαίτερη προσοχή γιατί εφαρμόζει την τεχνική των κόμβων φρουρών. Πιο συγκεκριμένα, πρέπει για κάθε βήμα h να δημιουργηθεί ένα αντίστοιχο στοιχείο φρουρός και κατά συνέπεια ο πίνακας table δεν αρκεί να επεκταθεί κατά ένα μόνο στοιχείο table[0] αλλά κατά h_1 στοιχεία. Αυτό οδηγεί σε μία επέκταση του ορισμού του πίνακα table, όπως δίνεται στη συνέχεια. Στη διαδικασία που ακολουθεί υποτίθεται ότι η ακολουθία των αυξήσεων αποτελείται από τέσσερις όρους με τιμές: h_1=9, h_2=5, h_3=3 και h_4=1.

```
TYPE index=-h1..n
VAR table:ARRAY[index] OF item;
PROCEDURE ModifiedShellSort;
CONST t=4;
```

```
VAR i,j,k,m,s:index; x:item; h:ARRAY[1..t] OF INTEGER;
BEGIN
   h[1]:=9; h[2]:=5; h[3]:=3; h[4]:=1;
   FOR m:= 1 T t DO
      BEGIN
         k:=h[m]; s:=-k
         FOR i:=k+1 TO n DO
            BEGIN
               x:=table[i]; j:=i-k;
               IF s=0 THEN s:=-k; s:=s+1; table[s]:=x;
               WHILE x.key<table[j].key DO
                  BEGIN
                     table[j+k]:=table[j]; j:=j-k
                  END
               table[j+k]:=x
            END
      END
END;
```

Οι ευθείες μέθοδοι των προηγουμένων κεφαλαίων επιδέχονταν σχετικά εύκολα επακριβή ανάλυση της συμπεριφοράς τους για κάθε περίπτωση (χειρότερη, καλύτερη και μέση), όσον αφορά τόσο στον αριθμό των συγκρίσεων όσο και στον αριθμό των μετακινήσεων. Δεν συμβαίνει όμως το ίδιο για τη μέθοδο του Shell, ούτε και για τις μεθόδους των επομένων κεφαλαίων. 'Ενα κεφαλαιώδες πρόβλημα της μεθόδου του Shell είναι η επιλογή του αριθμού των αυξήσεων και του μεγέθους τους και εκ προοιμίου δηλώνεται ότι δεν έχει δοθεί βέλτιστη λύση στο πρόβλημα αυτό. Στη βιβλιογραφία έχουν γίνει αρκετές προσπάθειες ανάλυσης διαφόρων ακολουθιών. Ενδεικτικά αναφέρεται ότι έχουν αποδειχθεί τα εξής αναλυτικά συμπεράσματα που δεν φαίνεται να συγκλίνουν σε ένα και μοναδικό αποτέλεσμα (δες βιβλίο Gonnet-Baeza).

Πρόταση.
Η μέση τιμή του αριθμού των μετακινήσεων
για την ακολουθία $\{2^k, 2^{k-1}, ..., 4, 2, 1\}$ είναι τάξης $O(n\sqrt{n})$,
για την ακολουθία $\{2^k - 1, 2^{k-1} - 1, ..., 7, 3, 1\}$ είναι τάξης $O(n^{3/2})$,
για την ακολουθία $\{2^p 2^q, ..., 9, 8, 6, 4, 3, 2, 1\}$ είναι τάξης $O(n \log^2 n)$. □

Λόγω της αδυναμίας των αναλυτικών αποδείξεων για την επίλυση του προβλήματος, περισσότερη γνώση αποκτάται με πειραματικά αποτελέσματα. Πειραματικά, λοιπόν, έχει προκύψει ότι ο απαιτούμενος χρόνος (λαμβάνοντας υπ' όψη τόσο τον αριθμό των συγκρίσεων όσο και τον αριθμό των μετακινήσεων) είναι τάξης $a \times nb$, όπου το a είναι μεταξύ 1,1 και 1,7, ενώ το b είναι από 1,25 ως 1,28. Πιο συγκεκριμένα στο βιβλίο του Knuth αναφέρονται τα εξής

προσομοιωτικά συμπεράσματα.

Πρόταση.

Η μέση τιμή του αριθμού των μετακινήσεων
για την ακολουθία $\{2^k + 1, ..., 9, 5, 3, 1\}$ είναι της τάξης $O(n^{1,27})$,
για την ακολουθία $\{2^k - 1, ..., 15, 7, 3, 1\}$ είναι της τάξης $O(n^{1,26})$,
για την ακολουθία $\{(2^k \pm 1)/3, ..., 11, 5, 3, 1\}$ είναι της τάξης $O(n^{1,28})$,
για την ακολουθία $\{(3^k - 1)/2, ..., 40, 13, 4, 1\}$ είναι της τάξης $O(n^{1,25})$. □

Από τον Knuth τελικά εικάζεται ότι η τελευταία ακολουθία είναι μία πολύ καλή ακολουθία. Μάλιστα ο γενικός τύπος της ακολουθίας αυτής είναι:

$$h_{k-1} = 3h_k + 1 \qquad h_t = 1 \qquad t = \lfloor \log_3 n \rfloor - 1$$

Το τελικό συμπέρασμα είναι ότι η μέθοδος του Shell, αν και είναι ασταθής, έχει πολύ καλύτερη επίδοση από όλες τις προηγούμενες ευθείες μεθόδους. Μάλιστα στην προσπάθεια βελτίωσης της μεθόδου έχουν προταθεί μερικές παραλλαγές. 'Οπως είναι γνωστό, όταν στη μέθοδο του Shell εκτελείται μία ταξινόμηση με βάση κάποιο βήμα, τότε είναι δυνατόν ένα στοιχείο να συγκριθεί και να ανταλλαχθεί με όλα τα στοιχεία που απέχουν το συγκεκριμένο βήμα. Από τον Dobosiewicz (1980) προτάθηκε μία μέθοδος που υλοποιείται στη συνέχεια. Αρχικά θεωρείται μία ακολουθία h με t αυξήσεις, όπου για την τελευταία αύξηση ισχύει $h[t] > 1$. Ο πίνακας διατρέχεται με το αντίστοιχο βήμα και εκτελούνται συγκρίσεις και ανταλλαγές ζευγών διαδοχικών στοιχείων που απέχουν τη συγκεκριμένη απόσταση. Με άλλα λόγια στην παραλλαγή αυτή ένα στοιχείο συγκρίνεται και πιθανώς ανταλλάσεται μόνο με ένα γειτονικό του. Η τεχνική αυτή προέρχεται ότι την ταξινόμηση της φυσσαλίδας του Κεφαλαίου 10.4, που υπενθυμίζεται ότι πάσχει από τον αυξημένο αριθμό των μετακινήσεων. Τέλος, μετά το πέρας του πρώτου μέρους της διαδικασίας εκτελείται μία αμιγής διαδικασία φυσσαλίδας.

```
PROCEDURE ShellBubble;
VAR i,j,k,l,t:INDEX; h:ARRAY[1..t] OF INTEGER;
BEGIN
   FOR i:=1 TO t DO
      BEGIN
         inc:=h[i];              (* Ορισμός της ακολουθίας των αυξήσεων h *)
         FOR j:=1 TO n-inc DO
            IF table[j]>table[j+inc] THEN Swap(table[j],table[j+inc])
      END;
   k:=n-1;
   WHILE k>0 DO
      BEGIN
```

```
        l:=0;
        FOR i:=1 TO k DO
            IF table[i]>table[i+1] THEN
                BEGIN
                    Swap(table[j],table[j+inc]); l:=i
                END;
            k:=l-1
    END
END;
```

Η παραλλαγή αυτή έδοσε αφορμή για περαιτέρω μελέτη. Από την Incerpi (1987) προτάθηκε κάθε φορά για μία συγκεκριμένη αύξηση να μην εφαρμόζεται μία διαδικασία τύπου φυσσαλίδας, αλλά μία διαδικασία τύπου shakersort, ενώ στο τελευταίο περάσμα θα πρέπει να εφαρμοσθεί μία διαδικασία τύπου ευθείας εισαγωγής. Αν και τα πειραματικά αποτελέσματα δεν έδειξαν βελτίωση της αρχικής μεθόδου, οι παραλλαγές αυτές αναφέρονται στο σημείο αυτό για να φανούν οι επίμονες προσπάθειες που συντέλεσαν στον εμπλουτισμό του αντικειμένου. Ο αναγνώστης καλείται να εξετάσει την 'Ασκηση 10.12 που αφορά στην υλοποίηση της προηγούμενης παραλλαγής. Στα επόμενα δύο κεφάλαια θα αναπτυχθούν ακόμα καλύτερες μέθοδοι.

10.8 Ταξινόμηση με διαμερισμό και ανταλλαγή

Στο κεφάλαιο αυτό θα παρουσιασθεί μία μέθοδος που στηρίζεται στην αρχή της ανταλλαγής. Υπενθυμίζεται ότι η μέθοδος της ευθείας ανταλλαγής ήταν η χειρότερη από τους αλγορίθμους των Κεφαλαίων 10.3 ως 10.5. Απεναντίας η μέθοδος που θα εξετασθεί στο κεφάλαιο αυτό, η μέθοδος **ταξινόμησης με διαμερισμό και ανταλλαγή** (partition exchange sort), είναι ιδιαίτερα αποτελεσματική για την ταξινόμηση πινάκων με τυχαία κλειδιά. Η επίδοση της είναι θαυμάσια και για το λόγο αυτό ονομάσθηκε **γρήγορη ταξινόμηση** (quicksort) από τον εμπνευστή της, τον Hoare (1962).

Η γρήγορη ταξινόμηση στηρίζεται στην παρατήρηση ότι είναι προτιμότερο οι ανταλλαγές να γίνονται μεταξύ απομακρυσμένων μεταξύ τους στοιχείων. Ας υποτεθεί ότι ο πίνακας αποτελείται από n στοιχεία που είναι ταξινομημένα κατά φθίνουσα τάξη της τιμής των κλειδιών τους. Τα στοιχεία αυτά είναι δυνατόν να ταξινομηθούν με $n/2$ ανταλλαγές μόνο, ανταλλάσοντας αρχικά το πιο αριστερό με το πιο δεξιό στοιχείο και προχωρώντας σταδιακά και από τα δύο άκρα του πίνακα προς τα μεσαία στοιχεία. Φυσικά αυτό μπορεί να γίνει όταν είναι εκ των προτέρων γνωστό ότι τα στοιχεία είναι ταξινομημένα σε φθίνουσα τάξη. Ωστόσο το παράδειγμα κάτι διδάσκει.

```
PROCEDURE QuickSort(left,right:index);
```

```
VAR pivot:item; i,j:index;
BEGIN
   IF right>left THEN
      BEGIN
         i:=left; j:=right+1; pivot:=table[left];
         REPEAT
            REPEAT i:=i+1 UNTIL table[i].key>=pivot.key;
            REPEAT j:=j-1 UNTIL table[j].key<=pivot.key;
            IF i<j THEN Swap(table[i],table[j]);
         UNTIL i>=j;
         Swap(table[left],table[j]);
         QuickSort(left,j-1); QuickSort(j+1,right)
      END
END;
```

Η βασική ιδέα στην προηγούμενη διαδικασία QuickSort είναι η λήψη του πρώτου στοιχείου του πίνακα και η μετακίνησή του στη θέση, όπου τελικά θα αποθηκευθεί ανάλογα με την τιμή του κλειδιού του. Ταυτόχρονα με την αποθήκευση του στοιχείου αυτού στην τελική του θέση γίνεται αναδιάταξη των υπόλοιπων στοιχείων, έτσι ώστε να μην υπάρχει κανένα μικρότερο κλειδί προς τα δεξιά του και κανένα μεγαλύτερο κλειδί προς τα αριστερά του. Έτσι, το πρώτο αυτό στοιχείο παίζει το ρόλο του **άξονα** (pivot) και ο πίνακας έχει διαμερισθεί κατά τέτοιο τρόπο, ώστε το αρχικό πρόβλημα έχει αναχθεί σε δύο απλούστερα προβλήματα, δηλαδή στην ανεξάρτητη ταξινόμηση των δύο υποπινάκων. Η απόσταση μεταξύ διαμερισμού και ταξινόμησης είναι πολύ μικρή. Μετά τον διαμερισμό του πίνακα η ίδια διαδικασία εφαρμόζεται στους δύο υποπίνακες, έπειτα στους υποπίνακες των υποπινάκων, κοκ., μέχρις ότου ο κάθε υποπίνακας να αποτελείται από ένα μόνο στοιχείο. Προφανώς, στο σημείο αυτό ο αρχικός πίνακας έχει ταξινομηθεί.

Παράδειγμα.

Έστω ότι ο αρχικός πίνακας αποτελείται από τα γνωστά εννέα κλειδιά. Στο Σχήμα 10.7 φαίνεται πως αλλάζουν οι δείκτες i και j, ώστε ο άξονας να πάρει την τελική θέση του. Στο Σχήμα 10.8 δίνεται η μορφή που παίρνει του πίνακα κατά τις διαδοχικές κλήσεις μέχρι την τελική του ταξινόμηση. Οι τρεις στήλες left, right, pivot δηλώνουν τις τιμές των αντίστοιχων παραμέτρων κατά τις διαδοχικές κλήσεις της διαδικασίας. Οι αγκύλες δείχνουν τους υποπίνακες μήκους μεγαλύτερου της μονάδας που απομένουν για ταξινόμηση. Στο σχήμα αυτό δεν φαίνονται τα στοιχεία των κλήσεων που απορρίπτονται από την πρώτη συνθήκη "left<right". Ο αναγνώστης μπορεί να επιβεβαιώσει ότι ο συνολικός αριθμός των κλήσεων είναι 13, και ποιές είναι οι τιμές των παραμέτρων κάθε φορά. □

Η ταξινόμηση με διαμερισμό και ανταλλαγή ανήκει στη μεγάλη οικογένεια

Αρχικά κλειδιά	52	12	71 $i\uparrow$	56	5	10	19	90	45 $j\uparrow$
1η ανταλλαγή	52	12	45	56 $i\uparrow$	5	10	19 $j\uparrow$	90	71
2η ανταλλαγή Διασταύρωση δεικτών	52	12	45	19	5	10 $j\uparrow$	56 $i\uparrow$	90	71
Ανταλλαγή και διαμερισμός	[10	12	45	19	5]	52	[56	90	71]

Σχήμα 10.7: Διαδικασία διαμερισμού.

κλήση	left	right	pivot								
1η	1	9	52	[52	12	71	56	5	10	19	90	45]
2η	1	5	10	[10	12	45	19	5]	52	[56	90	71]
4η	3	5	45	5	10	[45	19	12]	52	[56	90	71]
5η	3	4	12	5	10	[12	19]	45	52	[56	90	71]
9η	7	9	56	5	10	12	19	45	52	[56	90	71]
11η	8	9	90	5	10	12	19	45	52	56	[90	71]

Σχήμα 10.8: Διαδοχικοί διαμερισμοί.

αλγορίθμων που βασίζονται στη μέθοδο σχεδιασμού "διαίρει και βασίλευε". Υπενθυμίζεται ότι στην οικογένεια αυτή ανήκουν όλες οι μέθοδοι της επιλεκτικής αναζήτησης (δυαδική, Fibonacci, παρεμβολής κλπ.). Οι αλγόριθμοι αυτοί προσφέρονται για χρήση αναδρομικών κλήσεων με τη βοήθεια της γνωστής δομής της στοίβας αλλά είναι δυνατόν να προγραμματισθούν και με επαναληπτική μέθοδο. Αφήνεται για τον αναγνώστη η υλοποίηση της γρήγορης ταξινόμησης προσομοιώνοντας την αναδρομή με υλοποίηση της στοίβας (δες 'Ασκηση 10.13). Στη στοίβα θα πρέπει να εισάγεται το ζεύγος των ορίων (left,right) του υποπίνακα που θα απομένει προς επεξεργασία. Στο σημείο αυτό αναφέρεται ότι ενδιαφέρουσα είναι η τεχνική που τοποθετεί στη στοίβα τα όρια του μεγαλύτερου υποπίνακα, ώστε το μέγεθος της στοίβας στη χειρότερη περίπτωση να είναι τάξης O(n log$n$) και όχι O(n).

Η επίδοση της μεθόδου εξαρτάται από την επιλογή του άξονα. Προφανώς αν κάθε φορά ως άξονας επιλέγεται το μεσαίο στοιχείο του ταξινομημένου πίνακα, τότε η ταξινόμηση πρέπει να είναι πολύ αποτελεσματική. Πράγματι, αν οι τιμές των κλειδιών υπακούουν σε μία τυχαία στατιστική κατανομή, τότε η μέθοδος αυτή είναι πολύ γρήγορη. Ωστόσο η επίδοση της εκφυλίζεται όταν

τα κλειδιά είναι ήδη ταξινομημένα ή είναι ταξινομημένα αντίστροφα. Αυτό συμβαίνει γιατί κάθε φορά η επιλογή του άξονα είναι άτυχη, και έτσι ο πίνακας δεν επιμερίζεται σε δύο περίπου ισομήκεις υποπίνακες, όπως είναι επιθυμητό, αλλά σε ένα υποπίνακα μήκους 1 και σε ένα υποπίνακα μήκους n-1. Τίθεται, λοιπόν, η ερώτηση: "Ποιά είναι πράγματι η επίδοση του αλγορίθμου;" Στην ερώτηση απαντά η επόμενη πρόταση.

Πρόταση.

Η καλύτερη και η χειρότερη τιμή του αριθμού των συγκρίσεων για την ταξινόμηση n κλειδιών με τη μέθοδο της γρήγορης ταξινόμησης είναι τάξης O(n log$n$) και O(n^2) αντίστοιχα.

Απόδειξη.

Με το όρο **πέρασμα** (pass) εννοείται η επεξεργασία κάθε φορά όλου του πίνακα, δηλαδή η επεξεργασία όλων των υποπινάκων που δημιουργούνται σε κάθε διαμερισμό. Στο πρώτο πέρασμα απαιτούνται n-1 συγκρίσεις, στο δεύτερο απαιτούνται n-3 συγκρίσεις, κοκ. Άρα, σε κάθε πέρασμα του (υπο)πίνακα απαιτούνται συγκρίσεις της τάξης O(n). Ο αριθμός των περασμάτων είναι log$n$ και n στην καλύτερη και στη χειρότερη περίπτωση, αντίστοιχα. Άρα προκύπτει η πρόταση για την καλύτερη και τη χειρότερη περίπτωση. Όσον αφορά στη μέση επίδοση της μεθόδου προκύπτει ότι η μέση τιμή των συγκρίσεων είναι περίπου $1,386 \times n$ log$n$, δηλαδή είναι επίσης τάξης O(n log$n$). □

Για την επίλυση, λοιπόν, του προβλήματος της σωστότερης επιλογής του άξονα προτάθηκαν πολλές μέθοδοι. Μία από τις πρώτες παραλλαγές ήταν η λεγόμενη **γρηγορότερη ταξινόμηση** (quickersort), που προτάθηκε από τον Scowen (1965), και ως άξονα λαμβάνει το μεσαίο στοιχείο του υποπίνακα, ώστε να προλάβει τον εκφυλισμό της μεθόδου όταν ο πίνακας είναι ήδη ταξινομημένος. Ωστόσο είναι και πάλι δυνατόν κάθε φορά να επιλέγεται το μικρότερο ή το μεγαλύτερο στοιχείο του κάθε (υπο)πίνακα.

Ο Singleton (1969) πρότεινε να λαμβάνεται ως άξονας το μέσο κλειδί από τα εξής τρία κλειδιά: το πρώτο, το τελευταίο και το μεσαίο του υποπίνακα. Αυτός είναι ο λόγος που η ονομασία της μεθόδου αυτής είναι **ταξινόμηση με το μέσο των τριών** (median-of-three sort). Η παραλλαγή αυτή υλοποιείται με την παρεμβολή των εξής εντολών αμέσως πριν από την εντολή "pivot:=table[left]".

```
mid:=(left+right) DIV 2;
IF table[mid]>table[right] THEN Swap(table[mid],table[right]);
IF table[left]>table[right] THEN Swap(table[left],table[right]);
IF table[mid]>table[left] THEN Swap(table[mid],table[left]);
```

Με τις προηγούμενες εντολές επιτυγχάνεται η τοποθέτηση του μεσαίου κλειδιού στην πρώτη θέση, του μικρότερου στη μεσαία και του μεγαλύτερου στην τελευταία. Αν και αυτές οι συγκρίσεις και ανταλλαγές είναι μία χρονική επιβάρυνση, αποδείχθηκε ότι η τεχνική αυτή τελικά ενισχύει την απόδοση του αλγορίθμου.

Πιο συγκεκριμένα, αποδείχθηκε ότι η μέση τιμή του αριθμού των συγκρίσεων είναι περίπου $1,188 \times n \log n$, δηλαδή παραμένει και πάλι τάξης $O(n \log n)$.

Η **ταξινόμηση με το μέσο όρο** (meansort) είναι η γενίκευση της προηγούμενης μεθόδου και προτάθηκε από την Motzkin (1983). Κατά το πρώτο πέρασμα η μέθοδος αυτή λαμβάνει ως άξονα το κλειδί της πρώτης θέσης, όπως δηλαδή και η γρήγορη ταξινόμηση. 'Ομως κατά τη διάρκεια του πρώτου περάσματος υπολογίζει τις μέσες τιμές των τιμών των κλειδιών των δύο υποπινάκων και στη συνέχεια τις χρησιμοποιεί ως άξονες για τις περαιτέρω υποδιαιρέσεις των υποπινάκων. Είναι ευνόητο ότι με την τεχνική αυτή ελαχιστοποιείται ο αριθμός των περασμάτων και επομένως των συγκρίσεων αλλά η ταξινόμηση επιβαρύνεται σημαντικά με αριθμητικές πράξεις.

Οι μέθοδοι που αναφέρθηκαν βελτιώνουν τη χειρότερη περίπτωση της απλής μεθόδου της γρήγορης ταξινόμησης, που συμβαίνει όταν τα κλειδιά είναι ήδη ταξινομημένα. Ωστόσο καμία τους δεν αποφεύγει τον εκφυλισμό της επίδοσης σε τάξη $O(n^2)$, που συμβαίνει για κάποια άλλη διάταξη των κλειδιών του πίνακα. Στη συνέχεια παρουσιάζονται δύο υβριδικές μέθοδοι που βασίζονται στη γρήγορη ταξινόμηση.

```
PROCEDURE BQSort(left,right:index; midkey:INTEGER);
VAR flag,lflag,rflag:BOOLEAN; i,j,size:INDEX;
BEGIN
    IF left<right THEN
       BEGIN
          lflag:=FALSE; rflag:=FALSE; flag:=TRUE; i:=left; j:=right;
          WHILE flag DO
             BEGIN
                WHILE (table[i].key<midkey) AND (i<>j) DO
                   BEGIN
                      IF i<>left THEN IF table[i-1].key>table[i].key THEN
                         BEGIN
                            Swap(table[i-1],table[i]); lflag:=TRUE
                         END;
                      i:=i+1
                   END;
                WHILE (table[j].key>=midkey) AND (i<>j) DO
                   BEGIN
                      IF  j<>right  THEN  IF  table[j].key>table[j+1].key
THEN
                         BEGIN
                            Swap(table[j],table[j+1]); rflag:=TRUE
                         END;
```

```
                j:=j-1
              END;
            IF i<>j THEN Swap(table[j],table[i])
            ELSE
              BEGIN
                IF table[j].key>=midkey THEN
                  IF j<>right THEN
                    IF table[j].key>table[j+1].key THEN
                      BEGIN
                        Swap(table[j],table[j+1]); rflag:=TRUE
                      END
                ELSE
                  BEGIN
                    IF table[i-1].key>table[i].key THEN
                      BEGIN
                        Swap(table[i-1],table[i]); lflag:=TRUE
                      END;
                    IF table[i-2].key>table[i-1].key THEN
                      Swap(table[i-1],table[i-2])
                  END
                flag:=FALSE
              END
          END;
        size:=i-left;                (* Επεξεργασία αριστερού υποπίνακα *)
        IF size>2 THEN IF lflag THEN IF size=3 THEN
          IF table[left].key>table[left+1].key THEN
            Swap(table[left],table[left+1])
          ELSE BQSort(left,i-2,table[TRUNC((left+i-2)/2)]);
        size:=right-j+1;             (* Επεξεργασία δεξιού υποπίνακα *)
        IF size>2 THEN IF rflag THEN IF size=3 THEN
          IF table[j+1].key>table[j+2].key THEN
            Swap(table[j+2],table[j+1])
          ELSE BQSort(j+1,right,table[TRUNC((j+1+right)/2)])
      END
END;
```

Ο Wainright (1985) παρουσίασε μία υβριδική κατασκευή μεταξύ γρήγορης ταξινόμησης και ταξινόμησης φυσσαλίδας, που έχει πολύ καλή επίδοση όταν ο πίνακας είναι περίπου ταξινομημένος. Αν και η διαδικασία BQSort είναι σύνθετη, φαίνεται αμέσως ότι κατά το πρότυπο της ταξινόμησης φυσσαλίδας εκτελούνται συγκρίσεις και ανταλλαγές ζευγών γειτονικών στοιχείων καθώς οι

δείκτες i και j σαρώνουν τον πίνακα από τα δύο άκρα, καθώς και μετά από κάθε ανταλλαγή στοιχείων. Έτσι κατά το διαμερισμό οι υποπίνακες τείνουν περισσότερο προς το να είναι ήδη ταξινομημένοι και γι' αυτό ως άξονας λαμβάνεται το μεσαίο στοιχείο του υποπίνακα. Μάλιστα λόγω αυτών των ανταλλαγών που γίνονται κατά το πρότυπο της ταξινόμησης της φυσσαλίδας είναι βέβαιο ότι το δεξιότερο (αριστερότερο) κλειδί του αριστερού (αντίστοιχα, δεξιού) υποπίνακα είναι το μεγαλύτερο (αντίστοιχα, μικρότερο). Επιπλέον με τη βοήθεια των λογικών σημαιών είναι δυνατόν να αποφευχθούν αναδρομικές κλήσεις της BQSort αν πριν το διαμερισμό δεν έχουν γίνει τέτοιες ανταλλαγές, και αν ο υποπίνας αποτελείται από τρία ή λιγότερα στοιχεία.

Παράδειγμα.

Στο Σχήμα 10.9 παρουσιάζονται οι παράμετροι των κλήσεων που γίνονται από τη διαδικασία BQSort όταν εφαρμόζεται στη γνωστή ομάδα των εννέα κλειδιών. Ο αριθμός των κλήσεων είναι μόνο τρεις, επειδή οι υποπίνακες με δύο στοιχεία είναι ήδη ταξινομημένοι, ενώ οι υποπίνακες με τρία στοιχεία δεν απαιτούν κλήση αλλά τακτοποιούνται με μία το πολύ σύγκριση και ανταλλαγή στοιχείων.

κλήση	left	right	midkey
1η	1	9	5
2η	2	9	56
3η	2	5	45

Σχήμα 10.9: Διαδοχικοί διαμερισμοί με τη διαδικασία BQSort.

Στο Σχήμα 10.10 παρουσιάζονται οι ανταλλαγές που συμβαίνουν κατά τις δύο πρώτες κλήσεις. Ως άξονας λαμβάνεται κάθε φορά το μεσαίο στοιχείο. Έτσι στην πρώτη περίπτωση λαμβάνεται το κλειδί 5 (που είναι μία ατυχής επιλογή), ενώ στη δεύτερη περίπτωση μεσαίο κελιδί είναι το 56. Όπως φαίνεται, άλλοτε οι ανταλλαγές ακολουθούν τη λογική της ταξινόμησης ευθείας ανταλλαγής και άλλοτε τη λογική της ταξινόμησης με διαμερισμό και ανταλλαγή. □

Η μέθοδος αυτή όπως και οι προηγούμενες παραλλαγές δεν αποφεύγει τον εκφυλισμό της σε διαδικασία της τάξης $O(n^2)$ για μερικές διατάξεις των κλειδιών. Ωστόσο επαναλαμβάνεται και πάλι ότι η μέθοδος σχεδιάσθηκε για την περίπτωση που τα κλειδιά είναι περίπου ταξινομημένα και πράγματι με αυτή την προϋπόθεση είναι μία πολύ καλή επιλογή. Σε επόμενο κεφάλαιο θα δοθεί μία άλλη ειδική μέθοδος για την περίπτωση αυτή που είναι καλύτερη από τη μέθοδο του Wainright.

Η παρουσίαση των παραλλαγών κλείνει με μία πολύ επιτυχημένη υβριδική μέθοδο μεταξύ της γρήγορης ταξινόμησης και της ταξινόμησης ευθείας εισαγωγής, που δίνει πολύ καλά αποτελέσματα για τυχαία κλειδιά. Η μέθοδος αρχί-

12	52	71	56	5	10	19	90	45	Πίνακας πρίν 1η κλήση
							45	90	Ανταλλαγή αλά bubblesort
			5	56					Ανταλλαγή αλά bubblesort
		5	71						Ανταλλαγή αλά bubblesort
	5	52							Ανταλλαγή αλά bubblesort
5	12								Ανταλλαγή αλά quicksort
	12	52	71	56	10	19	45	90	Πίνακας πριν 2η κλήση
			45				71		Ανταλλαγή αλά quicksort
		45	52						Ανταλλαγή αλά bubblesort
				19		56			Ανταλλαγή αλά quicksort
			19	52					Ανταλλαγή αλά bubblesort

Σχήμα 10.10: Ανταλλαγές με τη διαδικασία BQSort.

ζει όπως η κλασική γρήγορη ταξινόμηση που μόλις αναπτύχθηκε. 'Οταν το μήκος κάθε υποπίνακα γίνει μικρότερο από μία συγκεκριμένη σταθερά, m, τότε σταματά η επεξεργασία του. Αυτό μπορεί να επιτευχθεί μετατρέποντας την πρώτη εντολή ελέγχου σε "right-left>m". 'Οταν δεν υπάρχει για επεξεργασία πίνακας με μήκος μεγαλύτερο από τη σταθερά m, τότε καλείται η μέθοδος ταξινόμησης με εισαγωγή που ταξινομεί τον πίνακα ως σύνολο. Από πειραματικές και αναλυτικές μελέτες έχει προκύψει ότι για τυχαία κλειδιά η τιμή της σταθεράς πρέπει να είναι 10 περίπου.

10.9 Ταξινόμηση με σωρό

Η μέθοδος της **ταξινόμησης με σωρό** (heapsort), που θα εξετασθεί στο κεφάλαιο αυτό προτάθηκε από τον Williams (1964). Στη μέθοδο αυτή επίσης εργάσθηκε και ο Floyd (1962, 1964). Η μέθοδος αυτή είναι αποτελεσματικότερη από τη μέθοδο ταξινόμησης με ελαττούμενες αυξήσεις, ωστόσο είναι πιο σύνθετη στον προγραμματισμό. Η αρχική μέθοδος, όπως διατυπώθηκε από τους Williams και Floyd, είναι λιγότερο αποτελεσματική από τη γρήγορη ταξινόμηση. Ωστόσο στο τέλος του παρόντος κεφαλαίου θα δοθεί μία πρόσφατη παραλλαγή της ταξινόμησης με σωρό, που είναι η καλύτερη μέθοδος που έχει εμφανισθεί ποτέ στη βιβλιογραφία. Για την καλύτερη κατανόηση της μεθόδου ο αναγνώστης μπορεί να ανατρέξει στο Κεφάλαιο 6.2 για περισσότερες λεπτομέρειες σχετικά με τη δομή του σωρού μεγίστων.

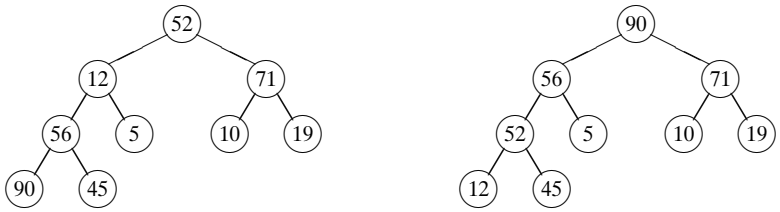
Η ταξινόμηση με σωρό αποτελείται από δύο φάσεις: τη φάση της δημιουργίας του σωρού και τη φάση της επεξεργασία του σωρού. Σε σχέση με την πρώτη φάση έχει γίνει ήδη εκτενής αναφορά στο Κεφάλαιο 6.2 όπου εξετάσθηκαν δύο μέθοδοι πολυπλοκότητας τάξης O(n log$n$) και O(n). Κατά

τη φάση της επεξεργασίας του σωρού το κλειδί της ρίζας εναλλάσσεται με το τελευταίο κλειδί του σωρού και στο εξής δεν λαμβάνεται υπ' όψη. 'Ομως με την αλλαγή αυτή τα εναπομείναντα κλειδιά δεν αποτελούν πλέον σωρό. 'Ετσι ακολουθεί μία διαδικασία αποκατάστασης των ιδιοτήτων του σωρού. Πιο συγκεκριμένα, το νέο κλειδί που βρέθηκε στη ρίζα του σωρού συγκρίνεται με το μεγαλύτερο από τα κλειδιά των κόμβων που είναι παιδιά της ρίζας. Αν το νέο κλειδί είναι μικρότερο, τότε τα κλειδιά ανταλλάσσονται. Η διαδικασία της καθόδου του νέου κλειδιού από τη ρίζα προς κατώτερα επίπεδα συνεχίζεται μέχρι να βρεθεί μικρότερο κλειδί ή να φθάσει και πάλι στο επίπεδο των φύλλων. Στη συνέχεια και πάλι το κλειδί της νέας ρίζας εναλλάσσεται με το κλειδί του (προ-)τελευταίου κόμβου του πίνακα. 'Ετσι η διαδικασία συνεχίζεται μέχρι να προκύψει σωρός μεγέθους 1.

Στη συνέχεια δίνεται η διαδικασία HeapSort που υλοποιεί την προηγούμενη ανάπτυξη χρησιμοποιώντας ένα σωρό μεγίστων. Η διαδικασία αποτελείται από δύο εντολές FOR. Η πρώτη εντολή FOR από τον αταξινόμητο πίνακα δημιουργεί τον αρχικό σωρό, ενώ η δεύτερη εντολή FOR επεξεργάζεται αυτόν το σωρό. Και στις δύο φάσεις καλείται η διαδικασία Restore, κάθε φορά με τα ανάλογα ορίσματα.

```
PROCEDURE Restore(first,last:index);
VAR father,child:index;
BEGIN
    father:=first; child:=2*father;
    WHILE father<=last DO
       BEGIN
          IF child<last THEN IF table[child].key<table[child+1].key
             THEN child:=child+1;
          IF table[father].key<table[child].key THEN
             BEGIN
                table[child DIV 2]:=table[child]); child:=2*child
             END
          ELSE father:=last+1
       END;
    table[child DIV 2]:=table[father]
END;
PROCEDURE HeapSort;
VAR i:index;
BEGIN
    FOR i:=(n DIV 2) DOWNTO 1 DO Restore(i,n);
    FOR i:=n DOWNTO 2 DO
       BEGIN
```

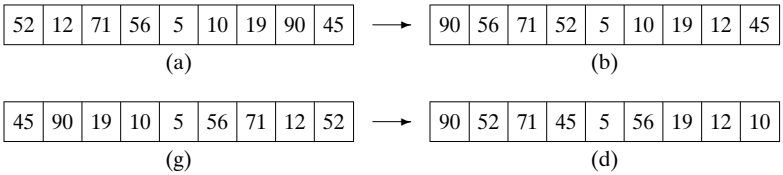
```
        Swap(table[1],table[i]); Restore(1,i-1)
    END
END;
```



Σχήμα 10.11: Δημιουργία σωρού από 9 κλειδιά.

Παράδειγμα.
Το Σχήμα 10.11 αναφέρεται στην αρχική φάση της δημιουργίας του σωρού για το γνωστό σύνολο των εννέα κλειδιών 52, 12, 71, 56, 5, 10, 19, 90 και 45, που είναι αποθηκευμένα στον πίνακα table. Στα αριστερά του σχήματος δίνεται η αρχική κατάσταση της δομής ως ένα πλήρες δυαδικό δένδρο, ενώ στα δεξιά παρουσιάζεται ο σωρός στην τελική του μορφή. Τα ενδιάμεσα στάδια απεικονίζονται στο Σχήμα 6.3. □

Στο Σχήμα 10.12α και 10.12β παρουσιάζεται το περιεχόμενο του πίνακα table που αντιστοιχεί στα Σχήματα 10.11α και 10.11ε. Αν τα ίδια κλειδιά ήταν αρχικά αποθηκευμένα στον πίνακα table με άλλη σειρά, τότε θα προέκυπτε διαφορετικός σωρός. Για παράδειγμα, ας θεωρηθεί ότι αρχικά ο πίνακας table περιέχει τα ίδια κλειδιά αλλά με την αντίστροφη σειρά 45, 90, 19, 10, 5, 56, 71, 12 και 52, όπως φαίνεται στο Σχήμα 10.12γ. Το τελικό περιεχόμενο του πίνακα table παρουσιάζεται στο Σχήμα 10.12δ. Είναι προφανές ότι τα Σχήματα 10.12β και 10.12δ είναι διαφορετικά.



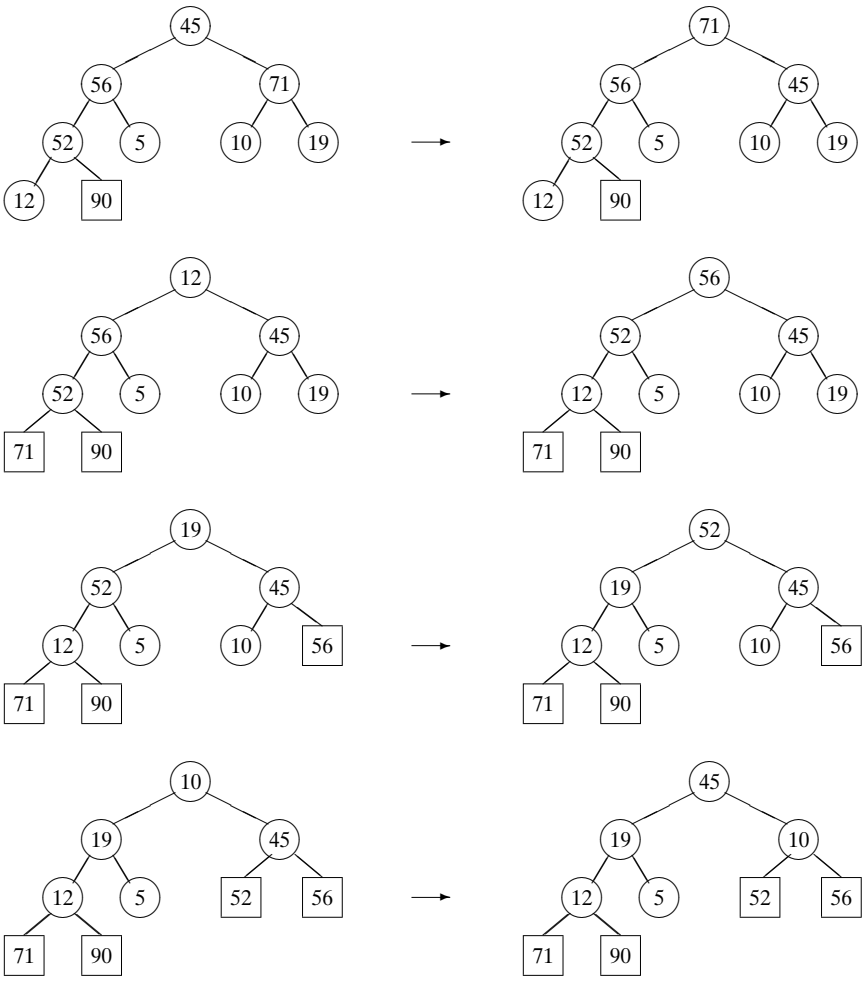
Σχήμα 10.12: Παράσταση σωρού ως πίνακα.

Παράδειγμα.
Στο Σχήμα 10.13 φαίνεται η επεξεργασία του σωρού του Σχήματος 10.11. Τα τετράγωνα πλαίσια δείχνουν ότι το κλειδί έχει καταλάβει την τελική του θέση και δεν υπόκειται σε άλλη επεξεργασία. Στη δεύτερη φάση της επεξεργασίας του σωρού η διαδικασία Restore καλείται οκτώ φορές μέσα από το δεύτερο

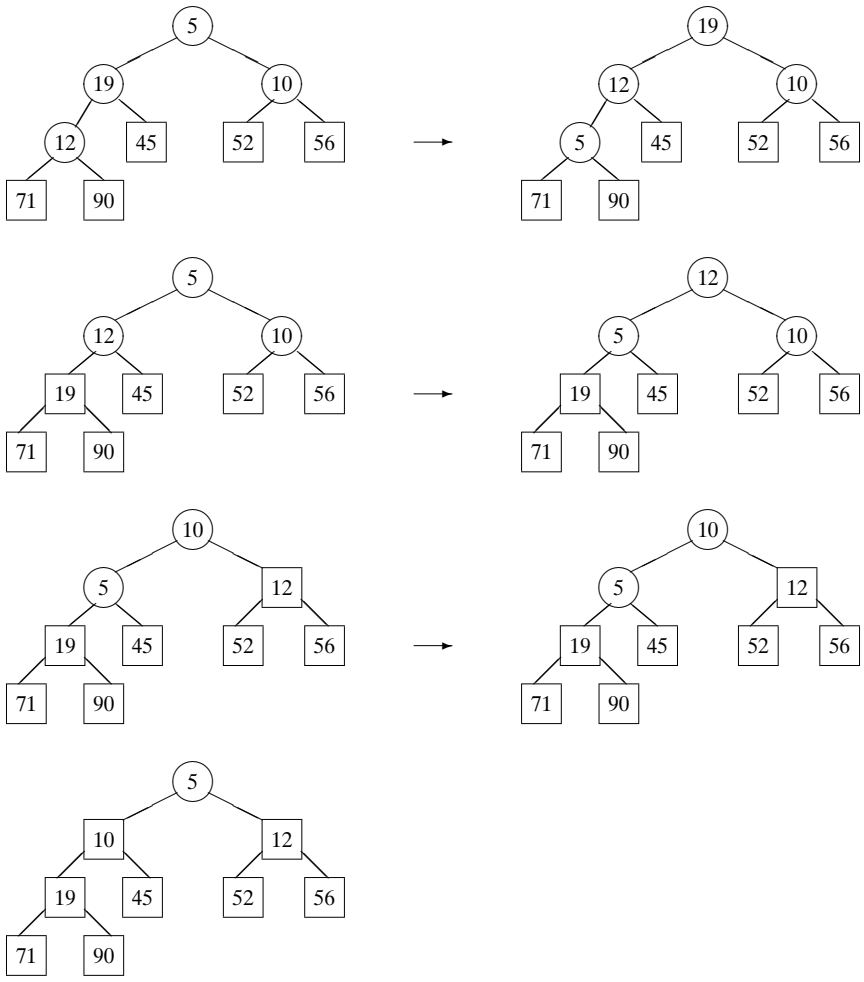
FOR της διαδικασίας HeapSort. 'Ετσι, μετά την όγδοη κλήση τα κλειδιά είναι τοποθετημένα στον πίνακα κατά αύξουσα τάξη. □

Σχετικά με τη επίδοση της μεθόδου ταξινόμησης με σωρό, κατ' αρχήν είναι γνωστό από το Κεφάλαιο 6.2, ότι η πρώτη φάση της δημιουγίας του σωρού απαιτεί γραμμικό χρόνο τάξης O(n). 'Οσον αφορά στη δεύτερη φάση της επεξεργασίας του σωρού αναφέρεται η εξής πρόταση.

Πρόταση.
Στη χειρότερη περίπτωση ο αριθμός των συγκρίσεων και ανταλλαγών κλειδιών κατά την επεξεργασία του σωρού είναι τάξης O($n \log n$).



Σχήμα 10.13: Επεξεργασία σωρού του Σχήματος 10.11.



Σχήμα 10.13: Επεξεργασία σωρού του Σχήματος 10.11.

Απόδειξη.
Στη φάση αυτή ο αριθμός των κλήσεων της διαδικασίας Restore ισούται με n-1. Σε κάθε κλήση αρχικά εκτελείται μία ανταλλαγή μέσα από τη διαδικασία HeapSort και ένας αριθμός ανταλλαγών μέσα από τη διαδικασία Restore. Σε κάθε κλήση το μέγεθος του σωρού είναι last-first, που συνεπάγεται κάθε φορά για το ύψος h του σωρού ισχύει $h = \lfloor \log(i\text{-}1) \rfloor$. Κάθε φορά, λοιπόν, η χειρότερη τιμή του αριθμού των ανταλλαγών κλειδιών ισούται με το αντίστοιχο ύψος και επομένως συνολικά η χειρότερη τιμή του αριθμού των ανταλλαγών είναι:

$$(n-1) \; + \; \sum_{i=2}^{n} \lfloor \log(i-1) \rfloor$$

από όπου με χρήση του τύπου του Strirling (δες Κεφάλαιο 6.2) προκύπτει ότι η χειρότερη τιμή του αριθμού των ανταλλαγών είναι:

$$n \times \log n \; + \; \mathrm{O}(n)$$

'Οσον αφορά στον αριθμό των συγκρίσεων με βάση το σκεπτικό της προηγούμενης πρότασης προκύπτει διπλάσια ποσότητα από τη συγκεκριμένη. Συνεπώς και πάλι στη χειρότερη περίπτωση ο αριθμός των συγκρίσεων είναι τάξης O(n log$n$). □

Η ανάλυση της μέσης περίπτωσης είναι δύσκολη υπόθεση. Προσεγγιστικά και μόνο, ας υποτεθεί ότι στην περίπτωση αυτή κάθε κόμβος μετακινείται από/προς τα φύλλα προς/από τη ρίζα το μισό της μέγιστης απόστασης. 'Αρα, η μέσες τιμές των δύο μεγεθών είναι υποδιπλάσιες των χειρότερων τιμών και επομένως προκύπτει και πάλι πολυπλοκότητα τάξης O(n) και O(n log$n$) για τις δύο φάσεις αντίστοιχα.

'Οσον αφορά στην επίδοση της μεθόδου αξιοσημείωτες είναι και οι εξής παρατηρήσεις:

- η μέθοδος αυτή είναι αποτελεσματική όταν τα δεδομένα εισάγονται ταξινομημένα κατά την αντίστροφη φορά,
- η χειρότερη περίπτωση έχει την ίδια πολυπλοκότητα με τη μέση περίπτωση, ενώ όπως είναι γνωστό δεν συμβαίνει το ίδιο με τη γρήγορη ταξινόμηση,
- στη χειρότερη περίπτωση εκτελεί περίπου $2n$ log$n$ συγκρίσεις, ενώ η γρήγορη ταξινόμηση εκτελεί στη μέση περίπτωση περίπου 1,38n log$n$ συγκρίσεις.

Πρόσφατα προτάθηκε μία πολύ ενδιαφέρουσα βελτίωση της ταξινόμησης με σωρό από τον Carlsson (1987), που βασίζεται στην εξής παρατήρηση. 'Οπως

ήδη αναλύθηκε η διαδικασία Restore σε κάθε επίπεδο εκτελεί δύο συγκρίσεις κλειδιών: μία μεταξύ των αδελφών για την εύρεση του μεγαλύτερου και μία μεταξύ παιδιού και ρίζας. Η παραλλαγή του Carlsson δεν εκτελεί αρχικά καμία σύγκριση μεταξύ ρίζας και παιδιού, αλλά βρίσκει ένα μονοπάτι από τη ρίζα προς τα φύλλα, όπου ανήκουν τα στοιχεία με τα μεγαλύτερα κλειδιά. Κατόπιν η ρίζα καταλαμβάνει τη σωστή θέση στο μονοπάτι αυτό εκτελώντας μία δυαδική εισαγωγή.

```
PROCEDURE ImprovedRestore(first,last:index);
VAR father,child,height,low,mid,high,k:index;
BEGIN
    father:=first; child:=2*father;
    WHILE child<last DO
        IF table[child].key<table[child+1].key THEN child:=2*(child+1)
        ELSE child:=2*child;
    IF child>last THEN child:=child DIV 2;
    height:=ROUND(ln(child DIV first)/ln2+0.5); low:=0; high:=height;
    WHILE low<high DO
        BEGIN
            mid:=(high+low) DIV 2;
            IF table[father].key<=table[child DIV Power(2,mid)].key THEN
                high:=mid ELSE low:=mid+1
        END;
    FOR k:=height-1 DOWNTO high DO
        table[child DIV Power(2,k+1)]:=table[child DIV Power(2,k)];
    table[child DIV Power(2,high)]:=table[father]
END;
PROCEDURE HeapSort;
VAR i:index;
BEGIN
    FOR i:=(n DIV 2) DOWNTO 1 DO ImprovedRestore(i,n)
    FOR i:=n DOWNTO 2 DO
        BEGIN
            Swap(table[1],table[i]); ImprovedRestore(1,i-1)
        END
END;
```

Η προηγούμενη διαδικασία ImprovedRestore υλοποιεί τη νέα αυτή παραλλαγή. Η δομή της διαδικασίας HeapSort δεν αλλάζει, ενώ η διαδικασία ImprovedRestore είναι παρόμοια με την προηγούμενη. Ποιά είναι όμως η επίδοση της μεθόδου αυτής; Η παραλλαγή αυτή δεν βελτιώνει την αρχική μέθοδο όσον αφορά στον αριθμό των ανταλλαγών, αλλά μόνο όσον αφορά στον αριθμό των

συγκρίσεων.

Πρόταση.

Η πολυπλοκότητα της χειρότερης τιμής του αριθμού των συγκρίσεων κλειδιών κατά την ταξινόμηση με τη μέθοδο του Carlsson είναι τάξης O(n log$n$).

Απόδειξη.

Η φάση της δημιουργίας του σωρού στη αρχική μέθοδο απαιτεί χρόνο τάξης O(n). Αν και η παραλλαγή του Carlsson βελτιώνει και αυτή τη φάση, δεν εξετάζεται περαιτέρω γιατί δεν είναι αυτό το κρίσιμο σημείο. Έστω ότι στη φάση της επεξεργασίας, ένας σωρός έχει i στοιχεία και ύψος h ίσο με log$i$. Στη χειρότερη περίπτωση η παραλλαγή αυτή εκτελεί h συγκρίσεις για την εύρεση του μονοπατιού των μεγαλύτερων παιδιών και log$h$ συγκρίσεις για την εκτέλεση της δυαδικής αναζήτησης. Συνεπώς ο συνολικός αριθμός των συγκρίσεων δίνεται από το άθροισμα:

$$\sum_{i=2}^{n}\lceil \mathsf{log}i\rceil \;+\; \sum_{i=2}^{n}\lceil \mathsf{loglog}i\rceil$$

Από το άθροισμα αυτό και πάλι προκύπτει πολυπλοκότητα τάξης O(n log$n$). □

Από την προηγούμενη πρόταση συνάγεται ότι δεν βελτιώνεται η ποιοτική αλλά η ποσοτική συμπεριφορά της μεθόδου, επειδή τώρα είναι μικρότερος ο σταθερός συντελεστής του όρου n log$n$. Πιο συγκεκριμένα όμως, έχει αποδειχθεί από τον Carlsson ότι απαιτούνται μόνο (n+1)(log(n+1) + loglog($n+1$)) συγκρίσεις στη χειρότερη περίπτωση. Η τιμή αυτή εγγίζει το θεωρητικό όριο, οπότε η τελευταία παραλλαγή καθίσταται ως η καλύτερη μέχρι τώρα μέθοδος ταξινόμησης. Ας σημειωθεί επίσης ότι η χειρότερη περίπτωση του αριθμού των συγκρίσεων της παραλλαγής αυτής είναι καλύτερη από τη μέση περίπτωση του αριθμού των συγκρίσεων της γρήγορης ταξινόμησης.

10.10 Ταξινόμηση με συγχώνευση

Στα Κεφάλαια 10.3 ως 10.9 εξετάσθηκαν μέθοδοι εσωτερικής ταξινόμησης που χρησιμοποιούν τις ίδιες θέσεις. Οι μέθοδοι αυτές για διδακτικούς λόγους κατατάχθηκαν σε τρεις κατηγορίες ανάλογα με το κυρίαρχο χαρακτηριστικό τους. Το κύριο χαρακτηριστικό των κατηγοριών αυτών ήταν η εισαγωγή, η ανταλλαγή και η επιλογή. Ο Knuth (1973) διακρίνει δύο άλλες κατηγορίες μεθόδων ταξινόμησης: τις μεθόδους που στηρίζονται στη **συγχώνευση** (merging) και τις μεθόδους που στηρίζονται στην **κατανομή** (distribution). Στις δύο τελευταίες κατηγορίες περιλαμβάνονται τεχνικές που δεν χρησιμοποιούν μόνο τις ίδιες θέσεις, αλλά απαιτούν και επιπλέον χώρο κύριας μνήμης.

Στο κεφάλαιο αυτό θα παρουσιασθούν μέθοδοι που στηρίζονται στη συγχώνευση, που είναι μία βασική λειτουργία των δομών δεδομένων όπως αναφέρεται

στο Κεφάλαιο 1.2. Το πρόβλημα της συγχώνευσης ενός αριθμού (δύο, τριών ή και περισσοτέρων) ταξινομημένων πινάκων έχει μελετηθεί από την Πληροφορική ως ένα ανεξάρτητο πρόβλημα και υπάρχει πλούσια βιβλιογραφία. Σημειώνεται μάλιστα ότι έχει ιδιαίτερη σημασία για το σχεδιασμό του αλγορίθμου η σχέση των μεγεθών των πινάκων.

Στη συνέχεια δίνεται ένας πολύ απλός αλγόριθμος συγχώνευσης δύο μόνο 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. Οι δηλώσεις των απαραίτητων δομών δίνονται στη συνέχεια.

```
VAR x:ARRAY [1..n] OF INTEGER;
    y:ARRAY [1..m] OF INTEGER;
    z:ARRAY [1..n+m] OF INTEGER;
```

Στη διαδικασία Merge που ακολουθεί οι μεταβλητές i, j και k είναι δείκτες για την κίνηση μέσα στους πίνακες x, y και z. Η μέθοδος προχωρεί 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. Στη συγκεκριμένη υλοποίηση που ακολουθεί, γίνεται χρήση ενός κόμβου φρουρού με τιμή MAXINT και για τους δύο πίνακες x και y. Είναι φανερό ότι οι συγκρίσεις κλειδιών, οι οποίες εκτελούνται από τη διαδικασία Merge είναι τάξης $O(n+m)$ μόνο.

```
PROCEDURE Merge;
VAR i,j,k:INTEGER;
BEGIN
    i:=1; j:=1; x[n+1]:=MAXINT; y[m+1]:=MAXINT;
    FOR k:=1 TO n+m DO
        IF x[i]<y[j] THEN
            BEGIN
                z[k]:=x[i]; i:=i+1
            END
        ELSE
            BEGIN
                z[k]:=y[j]; j:=j+1
            END
END;
```

Η διαδικασία αυτή είναι μία απλή εκδοχή της συγχώνευσης δύο ταξινομημένων πινάκων και ο αναγνώστης μπορεί να τη βελτιώσει σημαντικά αν ισχύει $n \gg m$. Ως ακραίο παράδειγμα, αν θεωρηθεί ότι m=1, τότε το στοιχείο

αυτό μπορεί να καταλάβει τη σωστή θέση μεταξύ των n στοιχείων με δυαδική αναζήτηση. Έτσι, λοιπόν, έχει προκύψει μία τεχνική που ονομάζεται **δυαδική συγχώνευση** (binary merging) και ισχύει για $n \geq m$. Η δυαδική συγχώνευση περιγράφεται αμέσως στη συνέχεια.

Έστω ότι οι δύο πίνακες είναι ταξινομημένοι κατά αύξουσα τάξη. Σύμφωνα, λοιπόν, με τη μέθοδο αυτή ο πίνακας των n στοιχείων υποδιαιρείται σε $m+1$ διαδοχικούς υποπίνακες. Αν το τελευταίο (m-οστό) στοιχείο του ενός πίνακα είναι μικρότερο από το τελευταίο στοιχείο του προτελευταίου υποπίνακα του πίνακα των n στοιχείων, τότε το στοιχείο αυτό μαζί με τα στοιχεία του τελευταίου υποπίνακα αποθηκεύονται στις τελευταίες θέσεις του πίνακα εξόδου. Στην αντίθετη περίπτωση βρίσκεται η θέση του m- οστού στοιχείου μεταξύ των στοιχείων του τελευταίου υποπίνακα, οπότε και πάλι το αποτέλεσμα αποθηκεύεται στον πίνακα εξόδου. Έτσι η μέθοδος συνεχίζει αναδρομικά μέχρι την εξάντληση και των δύο αρχικών πινάκων. Η υλοποίηση αυτού του ενδιαφέροντος αλγορίθμου αφήνεται στον αναγνώστη (δες Άσκηση 10.18). Μάλιστα, το ενδιαφέρον προκύπτει από την εξής πρόταση.

Πρόταση.

Η μέση τιμή του αριθμού των συγκρίσεων κατά τη δυαδική συγχώνευση είναι:

$$C_{n,m} = m + \left\lfloor \frac{n}{2t} \right\rfloor - 1 + tm \qquad t = \left\lfloor log \frac{n}{m} \right\rfloor$$

Από τη σχέση αυτή προκύπτει ότι $C_{n,n}$=2n-1 και $C_{n,1}$ = $\lfloor \log(n+1) \rfloor$. Δηλαδή, στις δύο ακραίες περιπτώσεις η μέθοδος συμπεριφέρεται όπως η διαδικασία Merge και η διαδικασία Binary (δες Κεφάλαιο 6.3), αντίστοιχα.

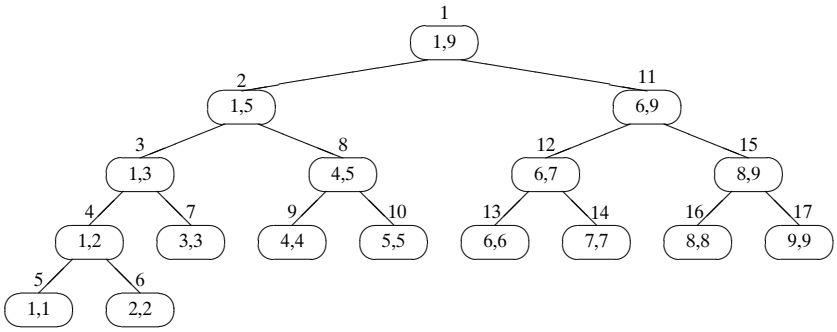
Ωστόσο, το πρόβλημα του κεφαλαίου αυτού είναι η ταξινόμηση με συγχώνευση. Στη βασική ιδέα της διαδικασίας Merge στηρίζεται η **ταξινόμηση ευθείας συγχώνευσης** (straight merge sort), που έχει μεγάλη ομοιότητα με την ταξινόμηση με διαμερισμό και ανταλλαγή, επειδή ανήκει και αυτή στην οικογένεια των αλγορίθμων που είναι σχεδιασμένες με τη μέθοδο "διαίρει και βασίλευε". Δηλαδή, υποδιαιρεί διαδοχικά τον πίνακα σε δύο υποπίνακες μέχρι να προκύψουν υποπίνακες με ένα στοιχείο. Κατόπιν, αυτοί οι υποπίνακες συγχωνεύονται διαδοχικά μέχρι να προκύψει ο τελικός ταξινομημένος πίνακας. Η επόμενη διαδικασία StraightMerge υλοποιεί αυτή τη μέθοδο χρησιμοποιώντας επιπλέον βοηθητικό χώρο, τον πίνακα aux, που έχει μεγέθος ίσο με τον αρχικό πίνακα.

```
PROCEDURE StraightMerge(low,mid,high:INTEGER);
VAR i,first,second,temp:index; aux:ARRAY [1..n] OF item;
BEGIN
    first:=low; second:=mid+1; temp:=low;
    WHILE (first<=mid) AND (second<=high) DO
```

```
        BEGIN
          IF table[first].key<=table[second].key THEN
            BEGIN
              aux[temp]:=table[first]; first:=first+1
            END
          ELSE
            BEGIN
              aux[temp]:=table[second]; second:=second+1
            END;
          temp:=temp+1
        END;
      IF first>mid THEN FOR i:=second TO high DO
        BEGIN
          aux[temp]:=table[i]; temp:=temp+1
        END
      ELSE FOR i:=first TO mid DO
        BEGIN
          aux[temp]:=table[i]; temp:=temp+1
        END;
      FOR i:=low TO high DO table[i]:=aux[i]
END;
PROCEDURE MergeSort(low,high:INTEGER);
VAR mid:INDEX;
BEGIN
    IF low<high THEN
      BEGIN
        mid:=(low+high) DIV 2;
        MergeSort(low,mid); MergeSort(mid+1,high);
        StraightMerge(low,mid,high)
    END
END;
```

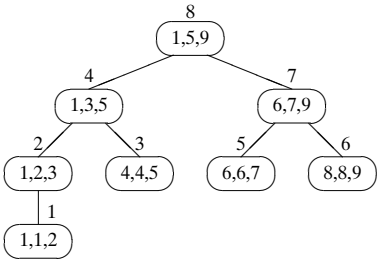
Παράδειγμα.

Έστω ότι και πάλι ο πίνακας table με τα γνωστά εννέα κλειδιά θεωρηθείται ως ολική μεταβλητή. Η διαδικασία MergeSort ακολουθώντας μία λογική διάσχισης με προτεραιότητα βάθους (λόγω της αναδρομικότητας) δημιουργεί τους υποπίνακες μεγέθους 1 εκτελούμενη 17 φορές όπως φαίνεται στο Σχήμα 10.14. Στο εσωτερικό των κόμβων του δένδρου του σχήματος δίνεται για κάθε κλήση η τιμή των παραμέτρων low και high. Στη συνέχεια στο Σχήμα 10.15 παρουσιάζεται η συγχώνευση των υποπινάκων, η οποία εκτελείται από τη διαδικασία StraightMerge μέχρι τη δημιουργία του τελικού ταξινομημένου πίνακα. Στο εσωτερικό



Σχήμα 10.14: Διαδοχικές κλήσεις της MergeSort.

των κόμβων απεικονίζονται οι τιμές των παραμέτρων low, high και mid, ενώ η διαδοχή των κλήσεων ακολουθεί μία λογική προδιατεταγμένης του δένδρου του σχήματος. □



Σχήμα 10.15: Διαδοχικές κλήσεις της StraightMerge.

Πρόταση.
Η πολυπλοκότητα για την ταξινόμηση n κλειδιών με τη μέθοδο της συγχώνευσης είναι τάξης O(n log$n$).
Απόδειξη.
'Οπως και για τον αλγόριθμο συγχώνευσης έτσι και για τη μέθοδο ταξινόμησης με συγχώνευση οι συγκρίσεις κλειδιών είναι τάξης O(n) για κάθε πέρασμα. Ευνόητο είναι επίσης ότι ο αριθμός των περασμάτων είναι log$n$. 'Αρα τελικά, σε κάθε περίπτωση (χειρότερη, μέση, καλύτερη) η πολυπλοκότητα της μεθόδου είναι τάξης O(n log$n$). □

'Οπως αναφέρεται στο βιβλίο του Knuth, η αρχική εκδοχή της προηγούμενης μεθόδου είχε επαναληπτική μορφή. Για να γίνει κατανοητή η εκδοχή αυτή, χωρίς βλάβη της γενικότητας, ας θεωρηθεί ότι το μέγεθος του πίνακα είναι δύναμη του δύο. Με τη βοήθεια δύο δεικτών ο αρχικός πίνακας εισό-

δου σαρώνεται από τα δύο άκρα με βήματα μεγέθους ένα, οπότε λαμβάνονται ζεύγη κλειδιών που συγχωνεύονται και τοποθετούνται εναλλάξ στα άκρα του δεύτερου πίνακα εξόδου. 'Οταν οι δείκτες συναντηθούν τότε ο ρόλος των πινάκων αλλάζει: η είσοδος γίνεται έξοδος και το αντίστροφο. Οι δύο δείκτες σαρώνουν το νέο πίνακα εισόδου με βήματα μεγέθους δύο, οπότε προκύπτουν τετράδες συγχωνευμένων κλειδιών που τοποθετούνται εναλλάξ στα άκρα του νέου πίνακα εξόδου. 'Ετσι η διαδικασία προχωρεί αντιστρέφοντας τους ρόλους των πινάκων και διπλασιάζοντας το μέγεθος του βήματος κάθε φορά μέχρι την πλήρη ταξινόμηση.

Μία άλλη μέθοδος της κατηγορίας αυτής είναι η **ταξινόμηση φυσικής συγχώνευσης** (natural merge sort), που μπορεί να υλοποιηθεί μόνο με επαναληπτική μορφή. Ποιά είναι η ειδοποιός διαφορά μεταξύ ευθείας και φυσικής συγχώνευσης; Στην μεν ευθεία συγχώνευση οι δείκτες προχωρούν με σταθερά και ίσα βήματα, δηλαδή 1, 2, 4 κλπ. Αντίθετα στη φυσική συγχώνευση τα βήματα δεν είναι ούτε σταθερά ούτε ίσα, αλλά το μέγεθος των δύο βημάτων είναι ανεξάρτητο και ισούται με το πλήθος των κλειδιών του πίνακα εισόδου που βρίσκονται σε μονότονη διάταξη. Στις άλλες λεπτομέρειες υπάρχει πλήρης ομοιότητα των δύο μεθόδων. 'Ετσι οι δύο ταξινομημένες υποακολουθίες από τον πίνακα εισόδου συγχωνεύονται και τοποθετούνται εναλλάξ στα άκρα του πίνακα εξόδου κοκ. Η υλοποίηση της ευθείας και της φυσικής συγχώνευσης με επαναληπτική μορφή αφήνεται στον αναγνώστη (δες 'Ασκηση 10.19).

Οι προηγούμενες τεχνικές είναι αμιγείς μέθοδοι συγχώνευσης. Στη συνέχεια περιγράφεται μία υβριδική παραλλαγή συγχώνευσης που προτάθηκε από τον Cook (1980) και είναι η ιδανικότερη μέθοδος για την ταξινόμηση πινάκων με στοιχεία που είναι περίπου ταξινομημένα. Υπενθυμίζεται ότι στην περίπτωση αυτή μπορούν να εφαρμοσθούν και δύο άλλες μέθοδοι:

- αν ο πίνακας είναι μικρός, τότε μπορεί να εφαρμοσθεί η ευθεία εισαγωγή,
- ενώ αν ο πίνακας είναι μεγάλος τότε μπορεί να εφαρμοσθεί η μέθοδος του Wainwright της οικογενείας της γρήγορης ταξινόμησης.

Η μέθοδος του Cook, που πειραματικά έχει αποδειχθεί ότι είναι πιο αποτελεσματική, λειτουργεί ως εξής. Αρχικά ελέγχονται τα στοιχεία του πίνακα εισόδου για τον εντοπισμό ζευγών διαδοχικών κλειδιών που δεν ακολουθούν τη σχέση διάταξης. Το ζεύγος αυτό διαγράφεται από τον πίνακα εισόδου και αποθηκεύεται σε έναν άλλο πίνακα εξόδου. Μάλιστα μετά την αφαίρεση τυχόντος ζεύγους η διαδικασία συνεχίζεται με τη σύγκριση του στοιχείου αμέσως πριν το πρώτο του ζεύγους με το στοιχείο αμέσως μετά το δεύτερο του ζεύγους. Μετά την απομάκρυνση κάποιων ζευγών ο πίνακας εισόδου είναι πλέον ταξινομημένος. Στη συνέχεια ο πίνακας εξόδου ταξινομείται χρησιμοποιώντας τη γρήγορη

ταξινόμηση, οπότε οι δύο ταξινομημένοι πίνακες συγχωνεύονται με μία διαδικασία παρόμοια της Merge. Η υλοποίηση της μεθόδου του Cook αφήνεται στον αναγνώστη (δες 'Ασκηση 10.20).

10.11 Ταξινόμηση με κατανομή

Οι μέθοδοι της κατηγορίας αυτής εμφανίσθηκαν πολύ πριν από τις μεθόδους όλων των άλλων κατηγοριών, όταν η εισαγωγή στοιχείων στον υπολογιστή γινόταν με διάτρητες κάρτες από ειδικές μηχανές. Με βάση τη λογική εκείνων των μεθόδων στη συνέχεια προτάθηκαν μέθοδοι, που μπορούν να εφαρμοσθούν στις σύγχρονες μηχανές. Ωστόσο, επειδή πολλές φορές είναι διαφορετική η φιλοσοφία της δομής τους, συχνά δεν μπορεί να γίνει άμεση αντιπαράθεση με τις προηγούμενες μεθόδους όσον αφορά στους παράγοντες κόστους, όπως ο αριθμός των συγκρίσεων και ο αριθμός των μετακινήσεων. 'Οπως έχει τονισθεί οι μέθοδοι της κατηγορίας αυτής απαιτούν επιπλέον χώρο για την αποθήκευση των δεδομένων και πολλές φορές είναι πιο βολικό να υλοποιούνται με χρήση συνδεδεμένων δομών και όχι σειριακών δομών, όπως οι κλασικοί πίνακες της Pascal.

Κάθε μέθοδος **ταξινόμησης με κατανομή** (distribution) αρχικά θεωρεί ένα κριτήριο κατανομής. Με βάση αυτό το κριτήριο το σύνολο των κλειδιών κατανέμεται σε m ανεξάρτητα υποσύνολα κλειδιών που λέγονται **κάδοι** ή **κουβάδες** (buckets). Στη συνέχεια τα κλειδιά κάθε κάδου κατανέμονται και πάλι με το ίδιο κριτήριο αναδρομικά σε m μικρότερους κάδους. Από τα προηγούμενα είναι προφανές ότι η τεχνική αυτή ανήκει στη μεγάλη οικογένεια των αλγορίθμων που είναι σχεδιασμένες με τη μέθοδο "διαίρει και βασίλευε", όπου ανήκουν πολλές από τις εξετασθείσες τεχνικές του βιβλίου αυτού. Πότε τερματίζεται η αναδρομή είναι ένα ζήτημα που ποικίλει από μέθοδο σε μέθοδο.

Για παράδειγμα, αν τα κλειδιά είναι ακέραιοι που ανήκουν στο διάστημα $[a,b]$, τότε μπορεί το διάστημα αυτό να υποδιαιρεθεί σε m υποδιαστήματα. 'Ετσι στο i-οστό υποδιάστημα ανήκουν κλειδιά με τιμές μεταξύ των ορίων $[a + (i-1) \times ((b-a)\mathsf{DIV}m), a + i \times ((b-a)\mathsf{DIV}m)]$. Τα κλειδιά κάθε υποδιαστήματος κατανέμονται αναδρομικά σε m μικρότερα υποσύνολα υποδιαιρώντας κάθε υποδιάστημα μέχρι τον πλήρη διαμερισμό των κλειδιών. Η μέθοδος αυτή ονομάζεται **ταξινόμηση με κατανεμητικό διαμερισμό** (distributive partition sort) και η επίδοσή της εξαρτάται από το πλήθος των κλειδιών, τον αριθμό m και την κατανομή των τιμών των κλειδιών στο διάστημα $[a,b]$. Αν η κατανομή αυτή απέχει από το να θεωρείται ομοιόμορφη, τότε πρακτικά δεν είναι χρησιμοποιήσιμη μέθοδος. Σε περίπτωση όμως που τα κλειδιά υπακούουν σε μία ομοιόμορφη κατανομή, τότε η επίδοση εξαρτάται κυρίως από τον αριθμό m.

Πρόταση.
Για την ταξινόμηση με διανεμητικό διαμερισμό, αν το m είναι σταθερά, τότε απαιτείται χρόνος τάξης O(n log$n$), ενώ αν το m ισούται με $\sqrt{n}$ ή με n, τότε απαιτείται χρόνος τάξης O(n loglog$n$) ή O(n), αντίστοιχα. □

Από τον Van der Nat (1980) προτάθηκε μία υβριδική μέθοδος, για την οποία έχει αποδειχθεί πειραματικά ότι βελτιώνει την επίδοση της προηγούμενης μεθόδου. Σύμφωνα, λοιπόν, με την υβριδική αυτή μέθοδο αρχικά λαμβάνονται τα μισά (n/2) κλειδιά και κατανέμονται στους μισούς (m/2) κάδους εφαρμόζοντας την προηγούμενη μέθοδο. Κάθε ένας από τους κάδους αυτούς ταξινομείται αναδρομικά με την ίδια μέθοδο αν το σύνολο των κλειδιών που περιέχει είναι μεγαλύτερο από δέκα. Στην αντίθετη περίπτωση καλείται η ευθεία εισαγωγή. Η ίδια διαδικασία ακολουθείται και για τα υπόλοιπα n/2 κλειδιά, οπότε προκύπτουν άλλοι m/2 κάδοι. Τέλος τα δύο ταξινομημένα υποσύνολα συγχωνεύονται σε ένα και μοναδικό πίνακα.

Είναι σαφές στον αναγνώστη ότι οι δύο αυτές μέθοδοι εκτός από τον επιπλέον βοηθητικό χώρο απαιτούν και σημαντική προσπάθεια για την παρακολούθηση των διαδοχικών κατανομών. 'Ενα άλλο χαρακτηριστικό των μεθόδων αυτών είναι ότι δεν εξαρτώνται από τη συγκεκριμένη αναπαράσταση των κλειδιών. Αντίθετα η μέθοδος που ακολουθεί χαρακτηρίζεται από αυτή την ιδιότητα.

Αν θεωρηθεί ότι τα κλειδιά είναι δεκαδικοί αριθμοί, τότε το m ισούται με 10, ενώ αν τα κλειδιά είναι συμβολοσειρές από τα κεφαλαία γράμματα του ελληνικού αλφαβήτου, τότε το m ισούται με 24. Για την πρώτη (δεύτερη) περίπτωση, λοιπόν, τα κλειδιά κατανέμονται σε δέκα (αντίστοιχα, εικοσιτέσσερις) κάδους με βάση το πιο σημαντικό ψηφίο (most significant digit), ταξινομούνται ξεχωριστά και κατόπιν συγχωνεύονται (δηλαδή, παρατίθενται) σε ένα ενιαίο σύνολο. Η διαδικασία κατανομής σε κάδους μπορεί να είναι αναδρομική, δηλαδή τα κλειδιά ενός κάδου κατανέμονται διαδοχικά σε m μικρότερους κάδους ανάλογα με το δεύτερο πιο σημαντικό ψηφίο κοκ. 'Αρα, ο μέγιστος αριθμός των αναδρομικών κλήσεων είναι ίσος με το πλήθος των ψηφίων των κλειδιών. Επειδή το κλειδί μπορεί να παρασταθεί με βάση τη ρίζα οποιοδήποτε αριθμητικού συστήματος, οπότε το m θα ισούται με τη ρίζα, για το λόγο αυτό και οι μέθοδοι αυτές λέγονται αλγόριθμοι **ταξινόμησης κατανομής ρίζας** (radix distribution sort).

Στη συνέχεια εξετάζεται με λεπτομέρεια μία μέθοδος για δυαδικά κλειδιά. Είναι γνωστό από το κεφάλαιο των ψηφιακών δένδρων (δες Κεφάλαιο 4.6) ότι τέτοιες μέθοδοι είναι ιδιαίτερα αποτελεσματικές σε γλώσσες προγραμματισμού, που επιτρέπουν πράξεις με δυαδικά ψηφία, όπως για παράδειγμα η C και οι γλώσσες assembly. Οι τελευταίες εκδόσεις της Pascal επιτρέπουν παρόμοιες πράξεις με τη βοήθεια των αριθμητικών τελεστών not, shl, shr, and, or και xor που δέχονται ως τελεσταίους ακεραίους αριθμούς των δύο χαρακτήρων.

Υπενθυμίζεται και πάλι ότι χρησιμοποιώντας τους τελεστές shr και shl από ένα δυαδικό αριθμό λαμβάνονται κάποια αριστερά ή δεξιά ψηφία, αντίστοιχα.

Η μέθοδος που θα εξετασθεί στη συνέχεια λέγεται **ταξινόμηση με ανταλλαγή ρίζας** (radix exchange sort) και εξετάζει τα ψηφία από τα αριστερά προς τα δεξιά, δηλαδή από το λεγόμενο πιο σημαντικό ψηφίο (most significant bit). Η μέθοδος θυμίζει τη γρήγορη ταξινόμηση επειδή αρχικά τα κλειδιά που έχουν ως πρώτο ψηφίο το Ο τοποθετούνται πριν τα κλειδιά που έχουν ως πρώτο ψηφίο το 1. Κατόπιν η επεξεργασία συνεχίζεται σε κάθε ένα από τα δύο αυτά υποσύνολα αναδρομικά για τα επόμενα ψηφία κατά τον ίδιο τρόπο.

Μερικά σχόλια σχετικά με τη διαδικασία RadixExchange που ακολουθεί κρίνονται αναγκαία. 'Εστω ότι ο πίνακας table αποτελείται από θετικούς ακεραίους τύπου INTEGER. Με τη βοήθεια δύο δεικτών, i και j, ο πίνακας σαρώνεται από τα αριστερά και από τα δεξιά αναζητώντας κλειδιά που να αρχίζουν από 1 και 0, αντίστοιχα. 'Οταν τέτοια κλειδιά εντοπισθούν τότε ανταλλάσσονται αμοιβαία. Η διαδικασία σάρωσης συνεχίζεται έως ότου οι δύο δείκτες συναντηθούν. Για την εκτέλεση της ταξινόμησης απαιτείται ως πρώτη εντολή η κλήση "RadixExchange(1,n,15)", οπότε στο πέρας της αναδρομικής διαδικασίας ο πίνακας είναι ταξινομημένος. Το όρισμα b της διαδικασίας χρησιμεύει για τον έλεγχο του αντίστοιχου ψηφίου ξεκινώντας από την τιμή 15 (για το αριστερότερο ψηφίο) και φθάνοντας μέχρι και την τιμή 0 (για το δεξιότερο ψηφίο). Αφήνεται στον αναγνώστη η μετατροπή της μεθόδου ώστε να ταξινομεί και αρνητικούς ακεραίους αριθμούς που δεν μπορούν να παρασταθούν με τον τύπο INTEGER (δες 'Ασκηση 10.23).

```
PROCEDURE RadixExchange(left,right,b:index);
VAR i,j:index;
BEGIN
   IF (right>left) AND (b>=0) THEN
      BEGIN
         i:=left; j:=right;
         REPEAT
            WHILE (table[i] SHR (b-1) MOD 2=0) AND (i<j) DO i:=i+1;
            WHILE (table[j] SHR (b-1) MOD 2=1) AND (i<j) DO j:=j-1;
            Swap(table[i],table[j]);
         UNTIL j:=i;
         IF (table[r] SHR (b-1) MOD 2=0) THEN j:=j+1;
         RadixExchange(left,j-1,b-1); RadixExchange(j,right,b-1)
      END
END;
```

Παράδειγμα.

Στο Σχήμα 10.16 παρουσιάζεται η εφαρμογή της μεθόδου στο γνωστό σύνολο

εννέα κλειδιών, που αντιστοιχούν σε δυαδικούς αριθμούς των επτά ψηφίων. Τα κλειδιά είναι τοποθετημένα κατακόρυφα στην πρώτη στήλη και η διαδικασία προχωρεί προς τα δεξιά. Η δεύτερη στήλη περιέχει τα κλειδιά μετά τον έλεγχο του πρώτου ψηφίου, η τρίτη στήλη περιέχει τα κλειδιά μετά τον έλεγχο του δεύτερου ψηφίου κοκ. Όταν ένα κλειδί έχει πλήρως διαφορισθεί, τότε δεν επαναλαμβάνεται η κλήση της διαδικασίας RadixExchange. □

52	0110100	52	0-110100	19	00-10011	5	000-0101	5	0000-101
12	0001100	12	0-001100	12	00-01100	12	000-1100	12	0001-100
71	1000111	45	0-101101	10	00-01010	10	000-1010	10	0001-010
56	0111000	56	0-111000	5	00-00101	19	001-0011		
5	0000101	5	0-000101	56	01-11000	45	010-1101		
10	0001010	10	0-001010	45	01-01101	56	011-1000	52	0110-100
19	0010011	19	0-010011	52	01-10100	52	011-0100	56	0111-000
90	1011010	90	1-011010	90	10-11010	71	100-0111		
45	0101101	71	1-000111	71	10-00111	90	101-1010		

10	00010-10	
12	00011-00	

Σχήμα 10.16: Έλεγχος διαδοχικών ψηφίων και ανταλλαγές κατά την ταξινόμηση με ανταλλαγή ρίζας.

Όπως φαίνεται βέβαια από το Σχήμα 10.16 υπάρχουν κάποιες διαφορές μεταξύ της ταξινόμησης με ανταλλαγή ρίζας και της ταξινόμησης με διαμερισμό και αναταλλαγή. Για παράδειγμα, σε κάθε κλήση ο άξονας είναι η τιμή 2^b, που δεν είναι υπαρκτό κλειδί, και συνεπώς δεν είναι βέβαιο ότι ένα κλειδί μετά την κλήση της διαδικασίας καταλαμβάνει την τελική του θέση. Ακόμη δεν χρησιμοποιείται κόμβος φρουρός αλλά ο έλεγχος γίνεται με τη συνθήκη "j=i".

Η ταξινόμηση με ανταλλαγή ρίζας γενικά συναγωνίζεται τη ταξινόμηση με διαμερισμό και ανταλλαγή σε αποτελεσματικότητα χωρίς, βέβαια, να είναι δυνατόν να γίνει άμεση σύγκριση λόγω των διαφορετικών λειτουργιών τους (συγκρίσεις κλειδιών έναντι συγκρίσεων ψηφίων). Πιο ειδικά ωστόσο, θεωρώντας ότι τα n κλειδιά αποτελούνται από b ψηφία, πρέπει να τονισθούν τα εξής:

- στη μέση περίπτωση, αν τα κλειδιά είναι διαφορετικά, τότε γίνεται δεκτό ότι $b \approx \log n$, και συνεπώς η πολυπλοκότητα του αλγορίθμου είναι τάξης $O(n \log n)$. Αυτή η τιμή αποτελεί ένα άνω όριο της μέσης περίπτωσης, επειδή ο αλγόριθμος μπορεί να τερματισθεί μόλις διαπιστωθεί ότι όλες οι διαφορές μεταξύ των κλειδιών έχουν τακτοποιηθεί. Για παράδειγμα, αν δοθούν για ταξινόμηση 1000 τυχαίοι ακέραιοι μήκους 32 δυαδικών ψηφίων, τότε αναμένεται ότι η ταξινόμηση θα τελειώσει μετά από την εξέταση των πιο αριστερών 11 ψηφίων (γιατί;).

- στη χειρότερη περίπτωση ο αριθμός των απαιτούμενων συγκρίσεων ψηφίων είναι της τάξης $O(nb)$. Αυτό συμβαίνει όταν τα κλειδιά:

- είναι μικροί αριθμοί, δηλαδή με πολλά μηδενικά στην αρχή, ή
- είναι γράμματα που παριστώνται με οκτώ ψηφία του πίνακα ASCII,
- είναι ίσα μεταξύ τους σε μεγάλο ποσοστό.

Στις περιπτώσεις αυτές η γρήγορη ταξινόμηση υπερτερεί σαφώς.

Μία άλλη κατηγορία αυτής της μεθόδου είναι η **ευθεία ταξινόμηση ρίζας** (straight radix sort) που κατ' εξοχήν χρησιμοποιούνταν από τις παλιές μηχανές ανάγνωσης διάτρητων καρτών. Σύμφωνα με τη μέθοδο αυτή τα δυαδικά ψηφία εξετάζονται από δεξιά προς τα αριστερά. Έτσι μετά από εξέταση του τελευταίου ψηφίου, στην αρχή (στο τέλος) του πίνακα μεταφέρονται τα κλειδιά που έχουν τιμή 0 (αντίστοιχα, τιμή 1) στη θέση αυτή διατηρώντας μεταξύ τους τις σχετικές θέσεις. Δηλαδή, στην περίπτωση αυτή δεν εφαρμόζεται η τεχνική των ανταλλαγών της προηγούμενης μεθόδου, γιατί θα μπορούσε να γίνει ανταλλαγή κλειδιών που βρίσκονται μεταξύ τους σε σωστή σειρά. Αφήνεται ως άσκηση στον αναγνώστη η υλοποίηση της μεθόδου αυτής με σειριακές και συνδεδεμένες δομές, όπως πίνακες και ουρές αντίστοιχα (δες Άσκηση 10.25).

10.12 Ταξινόμηση αλγορίθμων ταξινόμησης

Το κεφάλαιο αυτό συνοψίζει τους αλγορίθμους ταξινόμησης από την άποψη της επίδοσης και των βασικών τους χαρακτηριστικών. Στα Κεφάλαια 10.3 ως και 10.5 εξετάσθηκαν οι ευθείς αλγόριθμοι και για κάθε περίπτωση δόθηκαν επιμέρους επακριβή αναλυτικά στοιχεία. Βέβαια, τονίζεται και πάλι ότι οι επακριβείς αυτές τιμές ισχύουν για τις συγκεκριμένες υλοποιήσεις. Πιθανώς, με κάποια άλλη υλοποίηση (για παράδειγμα, με ή χωρίς κόμβο φρουρό, με συνδεδεμένες λίστες κλπ.) να προκύψουν κάποιες άλλες επακριβείς εκφράσεις, που όμως δεν θα διαφέρουν ως προς την τάξη μεγέθους. Στο Κεφάλαιο 10.6 αποδείχθηκε ότι δεν μπορεί να δημιουργηθεί αλγόριθμος ταξινόμησης ιδίων θέσεων με μέσο αριθμό συγκρίσεων μικρότερο της τάξης $O(n \log n)$. Σχετικά με τον αριθμό των συγκρίσεων και μετακινήσεων είναι πολύ δύσκολο, και μάλιστα σε πολλές περιπτώσεις μέχρι στιγμής ακατόρθωτο, να εξαχθούν επακριβείς τιμές για τις πιο κομψές μεθόδους ταξινόμησης που αναφέρονται στα Κεφάλαια 10.7 ως 10.9. Οι αλγόριθμοι αυτοί επιδεικνύουν στη μέση περίπτωση μία συμπεριφορά της τάξης $O(n \log n)$ και θεωρούνται ισοδύναμοι ποιοτικά. Στα Κεφάλαια 10.10 και 10.11 εξετάσθηκαν μέθοδοι ταξινόμησης που απαιτούν ένα συμπληρωματικό πίνακα, ίσου μεγέθους με τον αρχικό, για να εκτελεσθούν. Ο αναγνώστης παροτρύνεται να εξετάσει μερικές από τις ασκήσεις που βρίσκονται στο τέλος του κεφαλαίου επειδή αφορούν μεθόδους ταξινόμησης διαφορετικές από αυτές που παρουσιάσθηκαν στα προηγούμενα καφάλαια.

Στο κεφάλαιο αυτό γίνεται μία συνολική ποιοτική κατάταξη των αναλυτικών στοιχείων όλων των μεθόδων των προηγουμένων κεφαλαίων. Στον επόμενο πί-

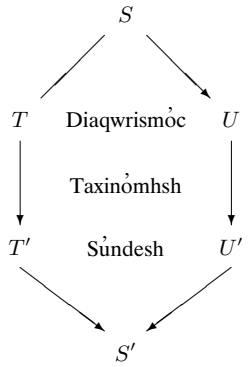
νακα παρουσιάζεται μία συνοπτική ποιοτική καταγραφή από την άποψη του αριθμού των απαιτούμενων συγκρίσεων και μετακινήσεων καθώς και του απαιτούμενου επιπρόσθετου χώρου για τις τρεις περιπτώσεις: χειρότερη, μέση και καλύτερη. Σημειώνεται ότι όπου στον πίνακα υπάρχει παύλα, αυτό σημαίνει ότι δεν υπάρχουν διαθέσιμα στοιχεία. Ο αναγνώστης καλείται να υλοποιήσει τους αλγορίθμους για να συμπεράνει πειραματικά κατά πόσο ο ένας υπερτερεί του άλλου επειδή διαφέρει κατά το σταθερό όρο της πολυπλοκότητας.

	Συγκρίσεις	Μετακινήσεις	Χώρος	Ευστάθεια
	$O(n)$	$O(n)$		
Ευθεία	$O(n^2)$	$O(n^2)$	$O(1)$	ναι
εισαγωγή	$O(n^2)$	$O(n^2)$		
	$O(n)$	$O(n)$		
Δυαδική	$O(n \log n)$	$O(n^2)$	$O(1)$	ναι
εισαγωγή	$O(n \log n)$	$O(n^2)$		
	$O(n^2)$	0		
Ευθεία	$O(n^2)$	$O(n^2)$	$O(1)$	ναι
ανταλλαγή	$O(n^2)$	$O(n^2)$		
	$O(n^2)$	$O(n)$		
Ευθεία	$O(n^2)$	$O(n \log n)$	$O(1)$	ναι
επιλογή	$O(n^2)$	$O(n^2)$		
	$O(n \log n)$	-	$O(n \log n)$	
Γρήγορη	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	όχι
ταξινόμηση	$O(n^2)$	-	$O(n^2)$	
	-	-		
Ταξινόμηση	$O(n \log n)$	$O(n \log n)$	$O(1)$	όχι
με σωρό	$O(n \log n)$	$O(n \log n)$		
	$O(n \log n)$	$O(n \log n)$	$O(n)$	
Ταξινόμηση	$O(n \log n)$	$O(n \log n)$	$O(n)$	όχι
με συγχώνευση	$O(n \log n)$	$O(n \log n)$	$O(n)$	

Πίνακας 10.1: Επιδόσεις αλγορίθμων ταξινόμησης.

Στο βιβλίο αυτό ακολουθήθηκε ο διαχωρισμός των αλγορίθμων σε πέντε κατηγορίες, ανάλογα με το βασικό τους χαρακτηριστικό, όπως είχε προταθεί από τον Knuth. Πιο ειδικά υπενθυμίζεται ότι οι αλγόριθμοι που χρησιμοποιούν τις ίδιες θέσεις ανήκουν στις τρεις πρώτες κατηγορίες. Οι κατηγορίες αυτές απο-

τελούνται από τους αλγορίθμους με βασικό χαρακτηριστικό την εισαγωγή, την ανταλλαγή και την επιλογή. Έτσι από τα κλασικά παραδείγματα των απλών μέθοδων της ευθείας εισαγωγής, της ευθείας ανταλλαγής και της ευθείας επιλογής η παρουσίαση περνά στις πιο σύνθετες μεθόδους της ταξινόμησης με ελαττούμενες αυξήσεις, της γρήγορης ταξινόμησης και της ταξινόμησης με σωρό. Έτσι στο βιβλίο αυτό, καθώς και σε όλα τα βιβλία που παρατίθενται στη βιβλιογραφία, η σειρά παρουσίασης των αλγορίθμων ήταν από τους πιο εύκολους προς τους πιο σύνθετους. Μία τέτοια σειρά παρουσίασης χαρακτηρίζεται ως προσέγγιση "από κάτω προς τα επάνω" με την έννοια ότι προηγούνται οι μέθοδοι που δίνουν έμφαση στις στοιχειώδεις λειτουργίες του αλγορίθου.



Σχήμα 10.17: Παράσταση αλγορίθμων ταξινόμησης.

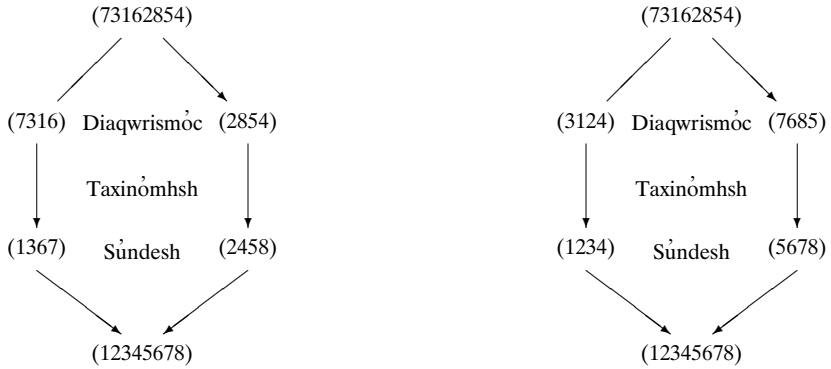
Σχετικά πρόσφατα από τη Merritt (1985) έχει προταθεί μία άλλη μεθοδολογία παρουσίασης, που χαρακτηρίζεται ως προσέγγιση "από επάνω προς τα κάτω" με την έννοια ότι προηγούνται οι μέθοδοι που δίνουν έμφαση στις αφηρημένες δομικές ιδιότητες των αλγορίθμων. Για τη νέα αυτή μεθοδολογία κάθε αλγόριθμος ταξινόμησης θεωρείται υπό το εξής γενικό πρίσμα. Υποτίθεται κατ' αρχήν ότι κάθε αλγόριθμος διαχωρίζει μία αταξινόμητη ακολουθία S σε δύο ανεξάρτητες υποακολουθίες και U. Οι δύο αυτές υποακολουθίες ταξινομούνται και έτσι προκύπτουν οι υποακολουθίες $'$ και U', αντίστοιχα. Αυτές στη συνέχεια συνδέονται και προκύπτει η ταξινομημένη ακολουθία S'. Βέβαια, αυτή η διαδικασία θυμίζει τη μέθοδο σχεδιασμού "διαίρει και βασίλευε" και μπορεί να θεωρηθεί ως αναδρομική. Στο Σχήμα 10.17 φαίνεται η σχηματική παράσταση των αλγορίθμων ταξινόμησης.

Όλοι, λοιπόν, οι αλγόριθμοι ταξινόμησης διαιρούνται σε δύο κατηγορίες: σε εκείνους που διακρίνονται από **εύκολο διαχωρισμό με δύσκολη σύνδεση** (easysplit/hardjoin) και σε εκείνους που διακρίνονται από **δύσκολο διαχωρισμό με εύκολη σύνδεση** (hardsplit/easyjoin). Δηλαδή, όπως δείχνει και η ονομασία,

Εύκολος διαχωρισμός Δύσκολη σύνδεση	Δύσκολος διαχωρισμός Εύκολη σύνδεση
Ευθεία εισαγωγή Με συγχώνευση Με ελαττούμενες αυξήσεις	Ευθεία ανταλλαγή Ευθεία επιλογή Με διαμερισμό και ανταλλαγή

Πίνακας 10.2: Ταξινόμηση αλγορίθμων ταξινόμησης.

οι αλγόριθμοι κατατάσσονται ανάλογα με τη δυσκολία ή ευκολία των διαδικασιών διαχωρισμού και σύνδεσης. Στον Πίνακα 10.2 φαίνεται η κατάταξη γνωστών αλγορίθμων σύμφωνα με αυτό το σκεπτικό.



Σχήμα 10.18: Παράδειγμα ταξινόμησης με συγχώνευση και γρήγορης ταξινόμησης.

Παράδειγμα.
Τα κλασικά παραδείγματα των δύο αυτών κατηγοριών κάτω από το νέο πρίσμα είναι η ταξινόμηση με συγχώνευση και η ταξινόμηση με διαχωρισμό και ανταλλαγή. Από αυτήν την άποψη τα παραδείγματα του Σχήματος 10.18 είναι ανάγλυφα. □

Ο αναγνώστης εύκολα θα διαπιστώσει πως ο αλγόριθμος ταξινόμησης με εισαγωγή μπορεί να θεωρηθεί ως μερική περίπτωση του αλγόριθμου ταξινόμησης με συγχώνευση, ενώ κατά τον ίδιο τρόπο ο αλγόριθμος ταξινόμησης με επιλογή μπορεί να θεωρηθεί ως μερική περίπτωση του αλγόριθμου ταξινόμησης

με διαμερισμό και ανταλλαγή.

10.13 Ασκήσεις

<**1**> **Ευθεία εισαγωγή με λίστα** (list insertion). Σύμφωνα με την παραλλαγή αυτή της ευθείας εισαγωγής διατηρείται ένας πίνακας link ίσου μεγέθους με τον πίνακα table που περιέχει δείκτες. Αρχικά τίθεται "link[i]:=i+1" για $1 \leq i < n$, και "link[n]:=0". Έτσι ο πίνακας table μπορεί να θεωρηθεί ως μία συνδεδεμένη λίστα που δείχνεται από έναν εξωτερικό δείκτη first. Η εισαγωγή του k-οστού στοιχείου εκτελείται με διάσχιση της λίστας και κατάλληλη ενημέρωση των δεικτών. Να υλοποιηθεί η μέθοδος και να εκτιμηθεί η επίδοσή της.
<**2**> **Ταξινόμηση με υπολογισμό διεύθυνσης** ή **ταξινόμηση με κατακερματισμό** (address calculation sorting, sorting by hashing). Η μέθοδος αυτή, που προτάθηκε από τον Isaac (1956), απαιτεί την ύπαρξη μίας συνάρτησης κατακερματισμού $f(.)$, τέτοιας ώστε αν $x < y$ τότε $f(x) < f(y)$. Η συνάρτηση αυτή κατευθύνει τα κλειδιά σε υποπίνακες με ανεξάρτητα μεταξύ τους πεδία ορισμού των τιμών κλειδιών. Τα κλειδιά κάθε υποπίνακα ταξινομούνται με οποιαδήποτε μέθοδο (για παράδειγμα, ευθεία εισαγωγή με λίστα) και ακολουθεί συγχώνευση (παράθεση) των υποπινάκων. Να υλοποιηθεί η μέθοδος και να εκτιμηθεί η επίδοσή της.
<**3**> Σε ένα πίνακα υπάρχουν αποθηκευμένα διπλά κλειδιά που πρέπει να απομακρυνθούν. Ποιά μέθοδος ταξινόμησης αν τροποποιηθεί εκτελεί αυτήν τη λειτουργία πιο αποτελεσματικά;
<**4**> Να γραφεί ένα πρόγραμμα που να εκτυπώνει όλες τις εξάδες θετικών αριθμών a_1, a_2, a_3, a_4, a_5 και a_6, έτσι ώστε να ισχύουν οι σχέσεις:

$$a_1 \leq a_2 \leq a_3 \leq 20 \qquad a_1 \leq a_4 \leq a_5 \leq a_6 \leq 20$$

ενώ επίσης το άθροισμα των τετραγώνων των αριθμών a_1, a_2 και a_3 να ισούται με το άθροισμα των τετραγώνων των αριθμών a_4, a_5 και a_6. Να χρησιμοποιηθεί μία διαδικασία ταξινόμησης για την εύρεση των διπλών κλειδιών.
<**5**> Να γραφεί πρόγραμμα που να δέχεται δύο ακεραίους m και n ($m < n$) και να δίνει m τυχαίους διακριτούς και ταξινομημένους ακέραιους αριθμούς από 1 ως n. Να υλοποιηθούν περισσότεροι από δύο αλγόριθμοι και να συγκριθούν από άποψη πολυπλοκότητας.
<**6**> Να υλοποιηθούν οι τρεις παραλλαγές της ευθείας ανταλλαγής (δες Κεφάλαιο 10.4), δηλαδή:

- με λογική σημαία που δείχνει αν στο τελευταίο πέρασμα έγιναν ανταλλαγές,
- με δείκτη της θέσης όπου έγινε η τελευταία ανταλλαγή, και

- με συνδυασμό των δύο προηγούμενων τεχνικών.

Να συγκριθεί η επίδοση των πέντε παραλλαγών της οικογένειας αυτής για τυχαία κλειδιά, ταξινομημένα κατά την ορθή φορά και ταξινομημένα κατά την αντίστροφη φορά. Με δική σας πρωτοβουλία να θεωρηθούν όλες οι παράμετροι της προσομοίωσης, όπως: το πλήθος κλειδιών, το πεδίο ορισμού των κλειδιών, το πλήθος των επαναλήψεων κλπ.

<7> **Ταξινόμηση περιττής-άρτιας μετάθεσης** (odd-even transposition sort). Σύμφωνα με αυτή τη μέθοδο στα περάσματα περιττής (άρτιας) τάξης συγκρίνονται τα στοιχεία A[i] και A[i+1], όπου το i είναι περιττός (αντίστοιχα, άρτιος) αριθμός. Αν ισχύει A[i]>A[i+1], τότε τα στοιχεία ανταλλάσσονται. Τα περάσματα επαναλαμβάνονται μέχρις ότου δεν συμβούν άλλες ανταλλαγές. Η μέθοδος αυτή να υλοποιηθεί και να εκτιμηθεί η επίδοσή της.

<8> **Ευθεία εισαγωγή δύο δρόμων** (two way straight insertion). Σύμφωνα με αυτή τη μέθοδο ένας πίνακας με n κλειδιά ταξινομείται τοποθετώντας τα κλειδιά σε έναν άλλον πίνακα μεγέθους $2n$+1 θέσεων. Η μέθοδος τοποθετεί το πρώτο κλειδί ως μεσαίο στοιχείο του πίνακα εξόδου. Τα υπόλοιπα κλειδιά τοποθετούνται αριστερά ή δεξιά του μεσαίου στοιχείου ανάλογα με το μέγεθος του κλειδιού. Κατά τη διάρκεια της τοποθέτησης γίνονται οι απαραίτητες μετακινήσεις. Να υλοποιηθεί η μέθοδος και να εκτιμηθεί η επίδοσή της. Για παράδειγμα, αν τα κλειδιά είναι στον πίνακα εισόδου με τη σειρά 52, 12, 71, 56, 5, 10, 19, 90 και 45, τότε τοποθετούνται στον πίνακα εξόδου με τη μορφή:

4	5	6	7	8	9	10	11	12	13	14
						52				
					12	52				
					12	52	71			
					12	52	56	71		
				5	12	52	56	71		
			5	10	12	52	56	71		
		5	10	12	19	52	56	71		
		5	10	12	19	52	56	71	90	
	5	10	12	19	45	52	56	71	90	

<9> **Δυαδική εισαγωγή δύο δρόμων** (two way binary insertion). Η μέθοδος είναι παρόμοια προς την ευθεία εισαγωγή δύο δρόμων της 'Ασκησης 10.8, αλλά η παρεμβολή γίνεται με δυαδικό τρόπο. Να υλοποιηθεί η μέθοδος και να εκτιμηθεί η επίδοσή της.

<10> Να συγκριθούν οι μέθοδοι των Ασκήσεων 10.8 και 10.9 σε σχέση με την ταξινόμηση της ευθείας και δυαδικής εισαγωγής αντίστοιχα του Κεφαλαίου 10.3.

<**11**> Σε ένα πίνακα Α με n στοιχεία συμβαίνει τα στοιχεία Α[1], Α[3], Α[5] κλπ. είναι τα μικρότερα στοιχεία, ενώ τα Α[2], Α[4], Α[6] κλπ. είναι τα μεγαλύτερα. Να βρεθεί ο συνολικός αριθμός των συγκρίσεων (ως συνάρτηση του n) που θα κάνει η μέθοδος της ευθείας εισαγωγής, η μέθοδος της δυαδικής εισαγωγής και η μέθοδος του Shell με δύο ελαττούμενες αυξήσεις μεγέθους 2 και 1, αντίστοιχα.
<**12**> Να υλοποιηθεί η μέθοδος της Incerpi, που είναι παραλλαγή της μεθόδου του Shell (δες Κεφάλαιο 10.7). Να συγκριθούν πειραματικά οι μέθοδοι του Shell, του Dobosiewicz και της Incerpi για κλειδιά τυχαία, κλειδιά ταξινομημένα κατά την ορθή φορά και κλειδιά ταξινομημένα κατά την αντίστροφη φορά όσον αφορά τόσο στον αριθμό των συγκρίσεων όσο και στον αριθμό των μετακινήσεων. Με δική σας πρωτοβουλία να θεωρηθούν όλες οι παράμετροι της προσομοίωσης, όπως: το πλήθος κλειδιών, το πεδίο ορισμού των κλειδιών, το πλήθος των επαναλήψεων κλπ.
<**13**> Να υλοποιηθεί με επαναληπτική μέθοδο (χρησιμοποιώντας στοίβες) ο αλγόριθμος ταξινόμησης με διαμερισμό και ανταλλαγή. Στη στοίβα να εισάγεται ο μεγαλύτερος υποπίνακας.
<**14**> **Τριαδικός σωρός μεγίστων** (ternary max heap). Η δομή αυτή είναι ένα τριαδικό σχεδόν πλήρες δένδρο, όπου η τιμή του κλειδιού του πατέρα είναι μεγαλύτερη από τις τιμές των κλειδιών των παιδιών. Με βάση τη διαδικασία Heapsort να υλοποιηθεί η μέθοδος ταξινόμησης με τριαδικό σωρό μεγίστων και να εκτιμηθεί η επίδοσή της.
<**15**> Πόσες συγκρίσεις απαιτούνται για την εύρεση των δύο μεγαλύτερων αριθμών ενός συνόλου n αριθμών;
<**16**> Δίνεται ένας πίνακας με n κλειδιά. Ζητείται να βρεθούν τα μικρότερα k κλειδιά, όπου $1 < k < n/2$, χωρίς όμως να ταξινομηθεί πρώτα ο πίνακας. Να προταθούν τουλάχιστον δύο τρόποι. Να εκτιμηθεί η επίδοση των τρόπων αυτών για διάφορες τιμές των n και k (για παράδειγμα, k=2 και n=1000, k=3 και n=8 κλπ.).
<**17**> Έστω ότι πρέπει να ταξινομηθεί ένας πίνακας με n κλειδιά, όπου τα αρχικά n_1 κλειδιά της ακολουθίας είναι ήδη ταξινομημένα, ενώ τα τελευταία n_2 δεν είναι. (Ισχύει βέβαια $n = n_1 + n_2$.) Ας θεωρηθεί ότι $n_1 \gg n_2$ και ότι $n_1 \approx n_2$. Να προταθούν μέθοδοι ικανοποιητικές για κάθε μία από τις δύο περιπτώσεις. Να γίνει ανάλυση της επίδοσής τους.
<**18**> Να υλοποιηθεί η τεχνική της δυαδικής συγχώνευσης και να διαπιστωθεί πειραματικά η ισχύς της σχετικής πρότασης (δες Κεφάλαιο 10.10). Με δική σας πρωτοβουλία να θεωρηθούν όλες οι παράμετροι της προσομοίωσης, όπως: το πλήθος κλειδιών, το πεδίο ορισμού των κλειδιών, το πλήθος των επαναλήψεων κλπ.
<**19**> Η διαδικασία MergeSort (δες Κεφάλαιο 10.10) είναι μία αναδρομική διαδικασία σχεδιασμένη από επάνω προς τα κάτω. Να υλοποιηθεί με επαναλη-

πτική μέθοδο η μέθοδος της ευθείας συγχώνευσης και της φυσικής συγχώνευσης σχεδιασμένη από κάτω προς τα επάνω χρησιμοποιώντας ένα βοηθητικό πίνακα.
<**20**> Να υλοποιηθεί η μέθοδος του Cook (δες Κεφάλαιο 10.10) με σκοπό την ταξινόμηση περίπου ταξινομημένων πινάκων.
<**21**> **Ταξινόμηση με μέτρηση της κατανομής** (distribution count sort). Η διαδικασία DistributionCount ταξινομεί n κλειδιά που είναι ακέραιοι από 1 ως m. Η διαδικασία να δοκιμασθεί για διαφορετικές τιμές του m και να υπολογισθεί η πολυπλοκότητα χρόνου και χώρου. Πότε η μέθοδος αυτή είναι αποτελεσματική;

```
PROCEDURE DistributionCount;
VAR i,j:index; temp:ARRAY[index] OF item;
    count:ARRAY[0..m-1] OF INTEGER;
BEGIN
    FOR j:=1 TO m DO count[j]:=0;
    FOR i:=1 TO n DO count[table[i]]:=count[table[i]]+1;
    FOR j:=2 TO m DO count[j]:=count[j-1]+count[j];
    FOR i:=n DOWNTO 1 DO
       BEGIN
          temp[count[table[i]]]:=table[i]; count[table[i]]:=count[table[i]]-
1
       END;
    FOR i:=1 TO n DO table[i]:=temp[i]
END;
```

<**22**> **Κυκλική ταξινόμηση** (cycle sort). Η μέθοδος αυτή που προτάθηκε από τον Haddon (1990) είναι μία παραλλαγή της μεθόδου της προηγούμενης άσκησης. Υποτίθεται ότι ένα ιστόγραμμα διατηρεί τις συχνότητες εμφάνισης όλων των τιμών των κλειδιών που βρίσκονται στο διάστημα $1..m$. Η διαδικασία CycleSort, που υλοποιεί τη μέθοδο, να δοκιμασθεί για διαφορετικές τιμές του m και να υπολογισθεί η πολυπλοκότητα χρόνου και χώρου. Πότε η μέθοδος είναι αποτελεσματική;

```
TYPE keytype=1..m; elementcount:=0..n;
    histogram=array[keytype] of elementcount;
PROCEDURE CycleSort;
VAR i,j:index; p:histogram; temp:item;
BEGIN
    p[1]:=0; FOR k:=1 TO m-1 DO p[k+1]:=p[k]+h[k];
    FOR i:=1 TO n DO
       BEGIN
          k:=table[i].key; j:=p[k]+1;
          IF i>=j THEN
             BEGIN
```

```
                IF i<>j THEN
                   BEGIN
                      temp:=table[i];
                      REPEAT
                         Swap(table[j],temp);
                         p[k]:=j; k:=temp.key; j:=p[k]+1;
                      UNTIL i=j
                      table[i]:=temp
                   END;
                p[k]:=i
             END
       END
END;
```

<**23**> Να τροποποιηθεί η διαδικασία RadixExchange ώστε να ταξινομεί και ακέραιους αριθμούς τύπου LONGINT (συμπεριλαμβανομένων των αρνητικών αριθμών). Επίσης να χρησιμοποιηθούν συνδεδεμένες δομές ως εξής. Αρχικά τα κλειδιά είναι συνδεδεμένα σε μία λίστα που σαρώνεται, οπότε τα κλειδιά εισάγονται στις αντίστοιχες ουρές με βάση το πιο σημαντικό ψηφίο. Η λίστα σχηματίζεται και πάλι με τη συγχώνευση-παράθεση των ουρών και τη σάρωση με βάση το επόμενο πιο σημαντικό ψηφίο κοκ.

<**24**> Να υλοποιηθεί η μέθοδος ευθείας ταξινόμησης ρίζας (δες Κεφάλαιο 10.11). Επειδή η μέθοδος πρέπει να είναι ευσταθής δεν συνιστάται η ανταλλαγή κλειδιών της διαδικασίας RadixExchange. Μπορεί να χρησιμοποιηθεί η διαδικασία DistributionCount της 'Ασκησης 10.21.

<**25**> Να υλοποιηθεί η μέθοδος ευθείας ταξινόμησης ρίζας χρησιμοποιώντας συνδεδεμένες δομές (όπως στην 'Ασκηση 10.23).

<**26**> **Ταξινόμηση γραμμικής εξέτασης** (linear probing sort). Η μέθοδος αυτή, που προτάθηκε από τον Gonnet (1981), συνδυάζει στοιχεία της αναζήτησης παρεμβολής και του ταξινομημένου κατακερματισμού με γραμμική εξέταση. Για να εφαρμοσθεί πρέπει να είναι γνωστά τα όρια των τιμών των κλειδιών και να θεωρηθεί ένας συμπληρωματικός πίνακας $m+w$ θέσεων, όπου $m > n$, ενώ οι w θέσεις χρησιμεύουν ως περιοχή υπερχείλισης. Κάθε στοιχείο του αρχικού πίνακα τοποθετείται σε κένη θέση του βοηθητικού πίνακα εφαρμόζοντας τη διαδικασία της αναζήτησης παρεμβολής. Αν η θέση του βοηθητικού πίνακα είναι κατειλημμένη από άλλο κλειδί, τότε η θέση αυτή καταλαμβάνεται τελικά από το μικρότερο κλειδί, ενώ το μεγαλύτερο τοποθετείται στην επόμενη θέση του βοηθητικού πίνακα. Γενικά ένα στοιχείο εκτοπίζεται από άλλα στοιχεία με μικρότερα κλειδιά. Μετά την εισαγωγή όλων των στοιχείων στο βοηθητικό πίνακα, με ένα απλό πέρασμα τα στοιχεία τοποθετούνται και πάλι στον αρχικό πίνακα. Να υλοποιηθεί η μέθοδος, να δοκιμασθεί πειραματικά και να εκτιμηθεί

η επίδοσή της. Με δική σας πρωτοβουλία να θεωρηθούν όλες οι παράμετροι της προσομοίωσης, όπως: το πλήθος κλειδιών, το πεδίο ορισμού των κλειδιών, το πλήθος των επαναλήψεων κλπ.

<27> **Το πρόβλημα της επιλογής**. Το πρόβλημα αυτό εξετάσθηκε στο Κεφάλαιο 6.8 και συνίσταται στην εύρεση του k-οστού μεγαλύτερου στοιχείου. Η επόμενη διαδικασία Find, που υλοποιεί την πράξη αυτή, προτάθηκε από τον Hoare (1970) και στηρίζεται στη γρήγορη ταξινόμηση. Η διαδικασία να δοκιμασθεί για διαφορετικές τιμές του k, και να υπολογισθεί αναλυτικά και πειραματικά η πολυπλοκότητα χρόνου και χώρου. Με δική σας πρωτοβουλία να θεωρηθούν όλες οι παράμετροι της προσομοίωσης, όπως: το πλήθος κλειδιών, το πεδίο ορισμού των κλειδιών, το πλήθος των επαναλήψεων κλπ.

```
PROCEDURE Find(k:index);
VAR left,right,i,j,index; x:item;
BEGIN
    left:=1; right:=n;
    WHILE left<right DO
        BEGIN
            x:=table[k]; i:=left; j:=right;
            REPEAT
                WHILE table[i]<x DO i:=i+1;
                WHILE x<table[j] DO j:=j-1;
                IF i<=j THEN
                    BEGIN
                        Swap(table[i],table[j]); i:=i+1; j:=j-1
                    END;
            UNTIL i>j;
            IF j<k THEN left:=i; IF k<i THEN right:=j
        END
END;
```