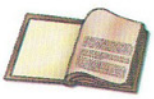
ΚΕΦΑΛΑΙΟ 4ο

ΤΕΧΝΙΚΕΣ ΣΧΕΔΙΑΣΗΣ ΑΛΓΟΡΙΘΜΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ

- Ανάλυση προβλημάτων
- Μέθοδοι σχεδίασης αλγορίθμων
- Μέθοδος διαίρει και βασίλευε
- Δυναμικός προγραμματισμός
- Άπληστη μέθοδος

Εισαγωγή



Όταν μας δίνεται ένα πρόβλημα, και πριν ασχοληθούμε με την κωδικοποίηση και τον προγραμματισμό, πρέπει προκαταρκτικά να κάνουμε μία θεωρητική μελέτη του προβλήματος. Πιο συγκεκριμένα, μία καλή ιδέα είναι να προσπαθήσουμε να διαπιστώσουμε, αν το δεδομένο πρόβλημα μπορεί να αντιμετωπισθεί με βάση κάποιες γενικές μεθοδολογίες ανάπτυξης αλγορίθμων. Σκοπός του κεφαλαίου είναι η εισαγωγή στις μεθολογίες αυτές, που αποτελούν ένα σύνολο τεχνικών και βημάτων που πρέπει να ακολουθηθούν. Για κάθε μεθοδολογία θα δούμε παραδείγματα επίλυσης γνωστών και καθημερινών προβλημάτων.

Διδακτικοί στόχοι

Στόχοι του κεφαλαίου αυτού είναι οι μαθητές:

- να τεκμηριώνουν την αναγκαιότητα ανάλυσης των προβλημάτων και σχεδίασης των κατάλληλων αλγορίθμων,
- να μπορούν να καταγράφουν την ακολουθία βημάτων για την ανάλυση των αλγορίθμων,
- να διατυπώνουν σύγχρονες τεχνικές σχεδίασης αλγορίθμων,
- να περιγράφουν τις κυριότερες προσεγγίσεις επίλυσης και ανάλυσης προβλημάτων,
- να επιλύουν προβλήματα με χρήση των κυριοτέρων προσεγγίσεων.

Προερωτήσεις

- ✓ 'Εχεις ακούσει για το πρόβλημα του περιοδεύοντα πωλητή;
- ✓ Είχες ποτέ φανταστεί ότι το «διαίρει και βασίλευε» μπορεί να έχει εφαρμογή στην Πληροφορική;
- ✓ Μπορεί να βρεθεί η ρίζα μιας μη πολυωνυμικής εξίσωσης;

4.1 Ανάλυση προβλημάτων

Σύμφωνα με όσα αναφέρθηκαν στα προηγούμενα κεφάλαια, ο αλγόριθμος αποσκοπεί στην επίλυση ενός προβλήματος. Είναι πιθανόν ένα πρόβλημα να μην επιλύεται με μία μόνο λύση αλλά με περισσότερες. Γενικά, η λύση σε ένα πρόβλημα μπορεί να προέλθει από ποικίλες και διαφορετικές προσεγγίσεις, τεχνικές και μεθόδους. Έτσι, είναι απαραίτητο να γίνεται μία καλή ανάλυση του κάθε προβλήματος και να προτείνεται συγκεκριμένη μεθοδολογία και ακολουθία βημάτων. Βασικός στόχος μας είναι η πρόταση έξυπνων και αποδοτικών λύσεων.

Η ανάλυση ενός προβλήματος σε ένα σύγχρονο υπολογιστικό περιβάλλον περιλαμβάνει:

- ✓ την καταγραφή της υπάρχουσας πληροφορίας για το πρόβλημα,
- ✓ την αναγνώριση των ιδιαιτεροτήτων του προβλήματος,
- ✓ την αποτύπωση των συνθηκών και προϋποθέσεων υλοποίησής του

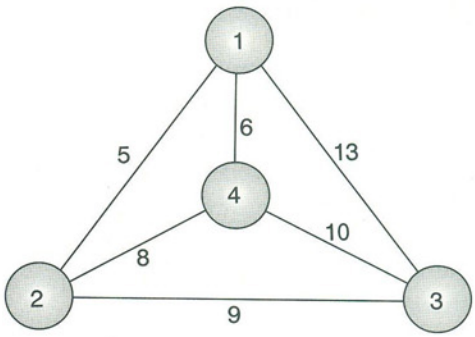
και στη συνέχεια:

- ✓ την πρόταση επίλυσης με χρήση κάποιας μεθόδου, και
- ✓ την τελική επίλυση με χρήση υπολογιστικών συστημάτων.

Έτσι, κατά την ανάλυση ενός προβλήματος θα πρέπει να δοθεί απάντηση σε κάθε μία από τις επόμενες ερωτήσεις:

1. Ποια είναι τα δεδομένα και το μέγεθος του προβλήματος,
2. Ποιες είναι οι συνθήκες που πρέπει να πληρούνται για την επίλυση του προβλήματος,
3. Ποια είναι η πλέον αποδοτική μέθοδος επίλυσής τους (σχεδίαση αλγορίθμου),
4. Πώς θα καταγραφεί η λύση σε ένα πρόβλημα (π.χ. σε ψευδογλώσσα), και
5. Ποιος είναι ο τρόπος υλοποίησης στο συγκεκριμένο υπολογιστικό σύστημα (π.χ. επιλογή γλώσσας προγραμματισμού)

Παράδειγμα. Έστω ότι αντιμετωπίζουμε το πρόβλημα ενός ταχυδρομικού διανομέα, που πρέπει να ξεκινήσει από ένα χωριό, να επισκεφθεί έναν αριθμό από γειτονικά χωριά, για να μοιράσει ένα σύνολο επιστολών και να επιστρέψει στο χωριό, από όπου ξεκίνησε περνώντας μόνο μία φορά από κάθε χωριό . Το πρόβλημα έγκειται στην εύρεση της καλύτερης διαδρομής, έτσι ώστε ο διανομέας να διανύσει το μικρότερο δυνατό αριθμό χιλιομέτρων.



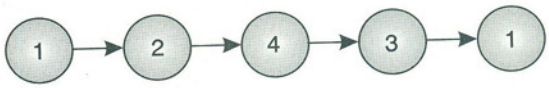
Σχ. 4.1. Χωριά και αποστάσεις μεταξύ τους.

Στο προηγούμενο σχήμα οι 4 αριθμημένοι κόμβοι αντιστοιχούν στα 4 χωριά, ενώ οι 6 συνδέσεις αντιστοιχούν στις οδικές αρτηρίας που ενώνουν τα χωριά αυτά. Τέλος, οι ακέραιοι που χαρακτηρίζουν τις συνδέσεις μεταξύ των κύκλων παρουσιάζουν τις αντίστοιχες χιλιομετρικές αποστάσεις μεταξύ των χωριών. Επιπλέον, ας υποθέσουμε ότι ο διανομέας ξεκινά από το χωριό 1, και βέβαια σε αυτό πρέπει να καταλήξει, αφού επισκεφθεί τα χωριά 2, 3 και 4. Στο πρόβλημα αυτό μπορούν να υπάρξουν διάφορες προσεγγίσεις για την ανάλυση και την επίλυσή του.

Α) Μία πρώτη ανάλυση του προβλήματος είναι:

1. να γίνει καταγραφή όλων των αποστάσεων μεταξύ των χωριών,
2. να ταξινομηθούν οι συνδέσεις των χωριών κατά αύξουσα χιλιομετρική απόσταση,
3. να επιλέγεται κάθε φορά η μετάβαση από το χωριό όπου βρίσκεται ο διανομέας προς το πλησιέστερο χωριό.

Με βάση τα παραπάνω βήματα ο διανομέας θα επέλεγε την εξής σειρά επίσκεψης των χωριών:



διανύοντας συνολικά 36 χιλιόμετρα.

Β) Μία διαφορετική ανάλυση του προβλήματος είναι:

1. να γίνει καταγραφή όλων των αποστάσεων μεταξύ των χωριών,
2. να βρεθεί μία σειρά επίσκεψης των χωριών με στόχο την ελαχιστοποίηση της συνολικής απόστασης και όχι την ελαχιστοποίηση της κάθε φορά απόστασης.

Με βάση τα παραπάνω βήματα ο διανομέας θα επέλεγε την εξής σειρά επίσκεψης των χωριών:

οπότε η συνολική διανυόμενη απόσταση είναι 30 χιλιόμετρα.

Επομένως είναι φανερό, ότι η ανάλυση κάθε προβλήματος είναι απαραίτητη, έτσι ώστε να αναζητηθεί η πλέον κατάλληλη μέθοδος που να παρέχει τη ζητούμενη λύση, όσο γίνεται ταχύτερα και με το λιγότερο δυνατό κόστος σε υπολογιστικούς πόρους. Δεν υπάρχει ένας ενιαίος κανόνας, μία γενική φόρμουλα που να αναφέρεται στην επίλυση του συνόλου των προβλημάτων. Υπάρχουν όμως "συγγενή" προβλήματα, δηλαδή προβλήματα που μπορούν να αναλυθούν με παρόμοιο τρόπο και να αντιμετωπισθούν με αντίστοιχες μεθόδους και τεχνικές. Γενικότερα, οι μέθοδοι ανάλυσης και επίλυσης των προβλημάτων παρουσιάζουν ιδιαίτερο ενδιαφέρον για τους εξής λόγους:

- παρέχουν ένα γενικό πρότυπο κατάλληλο για την επίλυση προβλημάτων ευρείας κλίμακας,
- μπορούν να αναπαρασταθούν με κοινές δομές δεδομένων και ελέγχου (που υποστηρίζονται από τις περισσότερες σύγχρονες γλώσσες προγραμματισμού),
- παρέχουν τη δυνατότητα καταγραφής των χρονικών και "χωρικών" απαιτήσεων της μεθόδου επίλυσης, έτσι ώστε να μπορεί να γίνει επακριβής εκτίμηση των αποτελεσμάτων.

4.2 Μέθοδοι σχεδίασης αλγορίθμων

Οι μέθοδοι λύσης ενός προβλήματος που προκύπτουν από την ανάλυσή του, οδηγούν στη σχεδίαση ενός αλγορίθμου που συνιστά την ακολουθία βημάτων, που πρέπει να ακολουθηθούν για να επιλυθεί το πρόβλημα. Όπως αναφέρθηκε, υπάρχει περίπτωση να παρουσιασθούν περισσότε-

ρες από μία τεχνικές για τη λύση ενός προβλήματος. Επομένως, για να προταθεί η καλύτερη λύση χρειάζεται να γίνουν κάποιες επιλογές και παραδοχές κατά τη διαδικασία σύνθεσης και σχεδίασης του αλγορίθμου.

Υπάρχουν μερικές τεχνικές που συχνά χρησιμοποιούνται σε πληθώρα προβλημάτων. Έτσι, οι τεχνικές αυτές έχουν τυποποιηθεί λόγω των κοινών χαρακτηριστικών τους κατά την επίλυση ενός προβλήματος. Η τυποποίηση αυτή σε κάποιο βαθμό διευκολύνει στην ένταξη κάποιου προβλήματος στην αντίστοιχη κατηγορία επίλυσής του (όπου αυτό είναι δυνατό). Γενικότερα, κάθε τεχνική χρειάζεται να υποστηρίζει τα εξής:

- ✓ να αντιμετωπίζει με τα δικό της τρόπο τα δεδομένα,
- ✓ να έχει τη δική της ακολουθία εντολών και
- ✓ να διαθέτει τη δική της αποδοτικότητα.

Επομένως κάθε τεχνική έχει τα δικά της χαρακτηριστικά και τις δικές της ιδιαιτερότητες.

Δεν υπάρχει αλγόριθμος για τη σχεδίαση αλγορίθμων.

Κατά την επίλυση ενός προβλήματος, επιχειρείται σύγκριση των χαρακτηριστικών και των ιδιοτήτων των τεχνικών που μπορούν να αποτελέσουν πρόταση λύσης του προβλήματος. Το αποτέλεσμα της σύγκρισης των διαφορετικών τεχνικών είναι η επιλογή της καταλληλότερης τεχνικής για την επίλυση του συγκεκριμένου προβλήματος. Στη βιβλιογραφία αναφέρονται αρκετές τυποποιημένες κατηγορίες τεχνικών, όμως στα πλαίσια του βιβλίου αυτού θα εξετάσουμε μερικές μόνο από αυτές:

- ✓ Μέθοδος διαίρει και βασίλευε
- ✓ Μέθοδος δυναμικού προγραμματισμού
- ✓ Άπληστη μέθοδος

Κάθε ένα από αυτά τα είδη προσεγγίσεων θα εξετασθεί ιδιαιτέρως στη συνέχεια.

Παρ' ότι στη βιβλιογραφία αναφέρονται αρκετές τεχνικές σχεδίασης αλγορίθμων, αυτό δεν σημαίνει ότι κάθε πρόβλημα μπορεί να επιλυθεί εφαρμόζοντας μία από αυτές τις γνωστές τεχνικές. Υπάρχουν πολλά προβλήματα που για την επίλυσή τους απαιτούν την εφαρμογή μίας νέας αντίληψης. Συχνά, οι μέθοδοι που εφαρμόζονται στα προβλήματα αυτά ονομάζονται *ευριστικές τεχνικές*.

4.3 Μέθοδος διαίρει και βασίλευε

Στην κατηγορία *"Διαίρει και Βασίλευε"* (divide and conquer) εντάσσονται οι τεχνικές που υποδιαιρούν ένα πρόβλημα σε μικρότερα υποπροβλήματα, που έχουν την ίδια τυποποίηση με το αρχικό πρόβλημα αλλά είναι μικρότερα σε μέγεθος. Με όμοιο τρόπο, τα υπο-προβλήματα αυτά μπορούν να διαιρεθούν σε ακόμη μικρότερα υποπροβλήματα κοκ. Έτσι η επίλυση ενός προβλήματος έγκειται στη σταδιακή επίλυση των όσο το δυνατόν μικρότερων υποπροβλημάτων, ώστε τελικά να καταλήξουμε στη συνολική λύση του αρχικού ευρύτερου προβλήματος. Αυτή η προσέγγιση ονομάζεται από *επάνω προς τα κάτω* (top-down).

Πιο τυπικά, η περιγραφή αυτής της μεθόδου σχεδίασης αλγορίθμων μπορεί να αποδοθεί με τα επόμενα βήματα:

1. Δίνεται για επίλυση ένα στιγμιότυπο ενός προβλήματος.
2. Υποδιαιρείται το στιγμιότυπο του προβλήματος σε υπο-στιγμιότυπα του ίδιου προβλήματος.
3. Δίνεται ανεξάρτητη λύση σε κάθε ένα υπο-στιγμιότυπο.
4. Συνδυάζονται όλες οι μερικές λύσεις που βρέθηκαν για τα υπο-στιγμιότυπα, έτσι ώστε να δοθεί η συνολική λύση του προβλήματος.

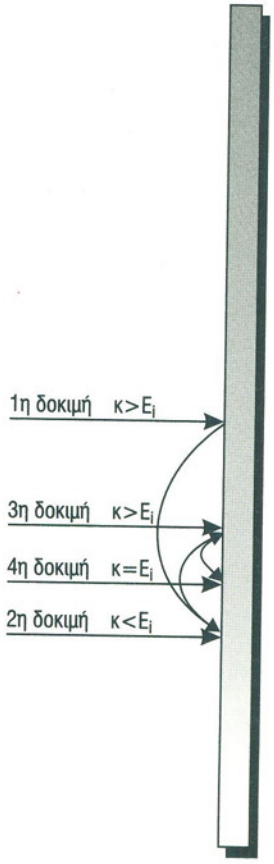
Στη συνέχεια παρουσιάζεται ένας κλασικός αλγόριθμος που ακολουθεί τη φιλοσοφία της μεθόδου Διαίρει και Βασίλευε: η Δυαδική αναζήτηση.

Δυαδική αναζήτηση

Έστω ότι δίνεται ένας ηλεκτρονικός τηλεφωνικός κατάλογος, δηλαδή ένας πίνακας με n ονόματα και έναν αριθμό τηλεφώνου για κάθε όνομα. Ο κατάλογος είναι ταξινομημένος κατά αύξουσα αλφαβητική τάξη ως προς τα ονόματα. Έστω ότι τα n ονόματα με τα αντίστοιχα τηλέφωνα είναι αποθηκευμένα σε δύο πίνακες: names[1..n] και phones[1..n] αντίστοιχα. Σε μία τέτοια περίπτωση, λοιπόν, το ζητούμενο είναι να βρεθεί το τηλέφωνο που αντιστοιχεί σε δεδομένο όνομα συνδρομητή (υποθέτοντας ότι το δεδομένο αναζητούμενο όνομα πράγματι υπάρχει στον κατάλογο).

Το πρόβλημα επιλύεται με τη μέθοδο Διαίρει και Βασίλευε με βάση την παρατήρηση ότι όταν δοθεί ένα όνομα, τότε υπάρχουν οι εξής 3 περιπτώσεις:

1. το όνομα βρίσκεται στη μεσαία θέση του πίνακα names,



Σχ. 4.2. Δυαδική αναζήτηση

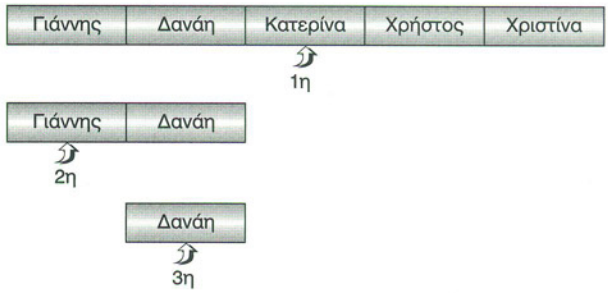
2. το όνομα βρίσκεται σε μία θέση στο πρώτο μισό κομμάτι του πίνακα names, ή
3. το όνομα βρίσκεται σε μία θέση στο δεύτερο μισό κομμάτι του πίνακα names.

Η δυαδική αναζήτηση χρησιμοποιείται μόνο στην περίπτωση ταξινομημένου πίνακα.

Η παρατήρηση αυτή οδηγεί στη σύνταξη του επόμενου αλγορίθμου. Σημειώνεται ότι ο αλγόριθμος αυτός ισχύει μόνο για την περίπτωση που το αναζητούμενο όνομα υπάρχει στον κατάλογο (δηλαδή, στον πίνακα names). Ο αλγόριθμος αυτός μπορεί εύκολα να τροποποιηθεί ώστε να αντιμετωπίζει και την περίπτωση της "ανεπιτυχούς αναζήτησης", δηλαδής της περίπτωσης όπου το αναζητούμενο όνομα δεν υπάρχει στον πίνακα.

```
Αλγόριθμος Δυαδική_αναζήτηση
Δεδομένα // names, phones, onoma, arxi, telos //
meso ← [arhi + telos]/2
Αν onoma = names[meso] τότε
    Tel ← phones[meso]
αλλιώς
    Αν onoma < names[meso] τότε
    Δυαδική_αναζήτηση(names, phones, onoma, arhi, meso-1)
    αλλιώς
    Δυαδική_αναζήτηση(names, phones, onoma, meso+1,telos)
    Τέλος_αν
Τέλος_αν
Αποτελέσματα  // Tel //
Τέλος Δυαδική_αναζήτηση
```

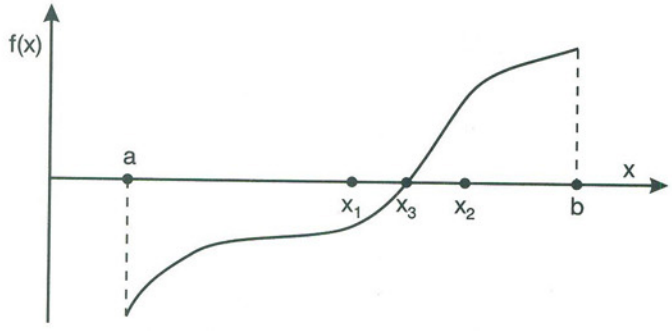
Παράδειγμα. Το παράδειγμα που ακολουθεί παρακολουθεί την αλληλουχία των βημάτων του αλγορίθμου της δυαδικής αναζήτησης ενός ονόματος σε έναν πίνακα πέντε θέσεων. Έστω ότι στον τηλεφωνικό κατάλογο αναζητείται το όνομα *Δανάη*. Κατά την πρώτη προσπάθεια από τις πέντε



***Σχ. 4.3** Διαδοχικές συγκρίσεις σε αρχικό πίνακα και υπο-πίνακες για την αναζήτηση του ονόματος Δανάη.*

θέσεις ελέγχεται η μεσαία, δηλαδή η τρίτη, όπου βρίσκεται το όνομα *Κατερίνα*. Το όνομα *Δανάη* είναι λεξικογραφικά μικρότερο από το όνομα *Κατερίνα* και επομένως η αναζήτηση περιορίζεται στο πρώτο μισό του πίνακα που αποτελείται από δύο θέσεις. Από τις δύο αυτές θέσεις, σε μία δεύτερη προσπάθεια ελέγχεται η πρώτη θέση, όπου βρίσκεται το τηλέφωνο του *Γιάννη*. Επειδή, το όνομα Δανάη είναι λεξικογραφικά μεγαλύτερο από το όνομα *Γιάννης*, η αναζήτηση περιορίζεται στον υπο-πίνακα που αποτελείται από τη δεύτερη θέση μόνο. Έτσι, στη θέση αυτή με μία τρίτη προσπάθεια βρίσκουμε το τηλέφωνο της *Δανάης*.

Ο τρόπος που λειτουργεί η δυαδική αναζήτηση είναι ανάλογος με αυτόν της μεθόδου της διχοτόμησης. Η *μέθοδος της διχοτόμησης* (ή του Bolzano) χρησιμοποιείται για την εύρεση μιας ρίζας της εξίσωσης $f(x)=0$ στο διάστημα $[a, b]$. Ως γνωστό, στο διάστημα αυτό υπάρχει μία τουλάχιστον ρίζα, αν ισχύει $f(a).f(b)<0$ (για την ακρίβεια μπορεί να υπάρχει περιττός αριθμός ριζών). Με τη μέθοδο της διχοτόμησης βρίσκεται ένα σημείο x_1 $(=(a+b)/2)$ στο μέσο του διαστήματος $[a, b]$ και εξετάζεται, αν $f(a).f(x_1)<0$. Αν ναι, τότε η ρίζα υπάρχει στο διάστημα $[a, x_1]$, αλλιώς στο $[x_1, b]$. Στη συνέχεια βρίσκεται το μέσο x_2 του υποδιαστήματος που υπάρχει η ρίζα και επαναλαμβάνονται τα ίδια. Έτσι σε κάθε επανάληψη εξαιρείται από την έρευνα το μισό διάστημα και προφανώς η ακολουθία τιμών x_1, x_2, ... συγκλίνει προς τη ρίζα της εξίσωσης. Η διαδικασία τερματίζεται, όταν η προσέγγιση της ρίζας θεωρείται ικανοποιητική.



Σχ. 4.4 Σχηματική παράσταση της μεθόδου της διχοτόμησης

4.4 Δυναμικός προγραμματισμός

Οι αλγόριθμοι της προηγούμενης παραγράφου στηρίζονται στη φιλοσοφία που συνιστά να σπάζουμε ένα πρόβλημα σε μικρότερα, ώστε να επιλύονται ευκολότερα. Έτσι η επίλυση του αρχικού προβλήματος έρχεται στη συνέχεια ως σύνθεση της επίλυσης των μικρότερων υπο - προβλημά-

των. Στην παράγραφο αυτή, θα εξετασθούν προβλήματα με την αντίστροφη φιλοσοφία δηλαδή με προσέγγιση από *κάτω προς τα επάνω* (bottom - up). Πιο συγκεκριμένα, η φιλοσοφία των προβλημάτων αυτών είναι από την αρχή να επιλύονται τα μικρότερα προβλήματα και σταδιακά να επιλύονται τα μεγαλύτερα ως σύνθεση των απλούστερων. Δηλαδή, ένας τυπικός αλγόριθμος αυτής της τεχνικής ξεκινά με τα επιμέρους μικρότερου μεγέθους υποπροβλήματα, που επιλύονται με τη χρήση κάποιου κανόνα ή τύπου. Μάλιστα συνηθέστατα τα προσωρινά αποτελέσματα αποθηκεύονται σε ένα πίνακα ώστε να χρησιμοποιηθούν αργότερα χωρίς να απαιτείται να υπολογισθούν για δεύτερη ή τρίτη φορά κ.λπ. Στη συνέχεια οι επιμέρους αυτές λύσεις συνθέτουν την κατάληξη της τελικής λύσης του αρχικού προβλήματος.

Αυτή η μέθοδος σχεδίασης αλγορίθμων χρησιμοποιείται κυρίως για την επίλυση προβλημάτων βελτιστοποίησης, δηλαδή όταν χρειάζεται να βρεθεί το ελάχιστο ή το μέγιστο κάποιου μεγέθους. Στην προσέγγιση αυτού του τύπου δεν υπάρχει η ίδια λογική επαναληπτικών εκτελέσεων που υπήρχε στη προηγούμενη προσέγγιση, αλλά υπάρχει ένας πίνακας από υπο-αποτελέσματα που καταλήγει στην τελική λύση του προβλήματος. Επομένως συμπερασματικά ακολουθείται η εξής προσέγγιση:

1. ξεκινά η λύση από το ελάχιστο στιγμιότυπο του προβλήματος,
2. υπολογίζονται σταδιακά αποτελέσματα όλο και μεγαλύτερων υποστιγμιότυπων,
3. καταλήγει στη σύνθεση.

Η τεχνική του δυναμικού προγραμματισμού παρουσιάζεται με τη βοήθεια ενός απλού σχετικά παραδείγματος.

Υπολογισμός δύναμης

Πολλές γλώσσες προγραμματισμού προσφέρουν έναν ιδιαίτερο τελεστή για τον υπολογισμό της δύναμης. Ωστόσο, υπάρχουν άλλες γλώσσες που δεν διαθέτουν τέτοιο τελεστή, οπότε ο προγραμματιστής πρέπει από μόνος του να κωδικοποιήσει αυτή τη λειτουργία και να την εντάξει σε δική του βιβλιοθήκη.

Ας θεωρήσουμε , λοιπόν, ότι πρέπει να υπολογίσουμε την ακέραια δύναμη ενός πραγματικού αριθμού, a^b. Μία πρώτη απλή προσέγγιση είναι ο επόμενος επαναληπτικός αλγόριθμος Δύναμη1, του οποίου η λειτουργία είναι ευνόητη.

```
Αλγόριθμος Δύναμη1
Δεδομένα  // a, b //
power ← 1
Για i από 1 μέχρι b
   power ← power * a
Τέλος_επανάληψης
Αποτελέσματα // power //
Τέλος Δύναμη1
```

Ο αλγόριθμος Δύναμη1 είναι μεν εύκολος στον προγραμματισμό και την κατανόηση του, αλλά δεν είναι ιδιαίτερα αποτελεσματικός, επειδή εκτελεί b πολλαπλασιασμούς για τον υπολογισμό της b-οστής δύναμης. Το ότι ο αλγόριθμος αυτός δεν είναι αποτελεσματικός, αποδεικνύεται με το απλό παράδειγμα ότι το a^{16} μπορεί να υπολογισθεί με τέσσερις πολλαπλασιασμούς ως εξής: $(((a^2)^2)^2)^2$. Στη συνέχεια ακολουθεί ο νέος τρόπος επίλυσης του προβλήματος με χρήση της μεθόδου του δυναμικού προγραμματισμού, όπου η βασική ιδέα είναι η προσωρινή αποθήκευση κάποιων προηγούμενων δυνάμεων του αριθμού μέχρι τη σταδιακή κατάληξη στη δύναμη b. Συγκεκριμένα ο πίνακας *power* θα αποθηκεύει το αποτέλεσμα της ύψωσης στο τετράγωνο της προηγούμενης θέσης του πίνακα. Για παράδειγμα, αν θα υπολογιζόταν οι δυνάμεις του 2 (δηλαδή, έστω ότι a=2), τότε ο πίνακας στη θέση 0 θα είχε την τιμή 1, στη θέση 1 την τιμή 2, στη θέση 2 την τιμή 4, στη θέση 3 την τιμή 16 κοκ, όπως φαίνεται και στο επόμενο σχήμα.

	0	1	2	3	4	
	$2^0=1$	$2^1=2$	$2^2=4$	$2^4=16$	$2^8=256$	...
Δύναμη του 2	0-κή	1η	2η	4η	8η	...

Σχ. 4.5 Αποθήκευση δυνάμεων του 2.

Για τον υπολογισμό, λοιπόν, κάποιας δύναμης απαιτούνται οι κατάλληλες θέσεις του πίνακα. Λόγου χάριν, για την ανεύρεση του 2^7, αρκούν οι αποθηκευμένες δυνάμεις μέχρι και την 4η δύναμη του 2, διότι $2^7 = 2^4 * 2^2 * 2^1$. Ο αλγόριθμος που ακολουθεί παρουσιάζει τον υπολογισμό του a^b με τη νέα αυτή ευφυέστερη τεχνική, που βασίζεται στο λεγόμενο δυναμικό προγραμματισμό.

```
Αλγόριθμος Δύναμη2
Δεδομένα // a, b //
!Σχόλιο: αποθήκευση στοιχείων πίνακα
power[1] ← a
i ← 1
pow ← 1
Όσο pow < b επανάλαβε
    i ← i+1
    pow ← 2* pow
    power[i] ← power[i-1] * power[i-1]
Τέλος_επανάληψης
!Σχόλιο: ανεύρεση της δύναμης
used ← 0
result ← 1
Όσο used < b επανάλαβε
    Αν used + pow <= b τότε
        result ← result * power[i]
        used ← used + pow
    Τέλος_αν
    pow ← pow / 2
    i ← i - 1
Τέλος_επανάληψης
Αποτελέσματα // result //
Τέλος Δύναμη2
```

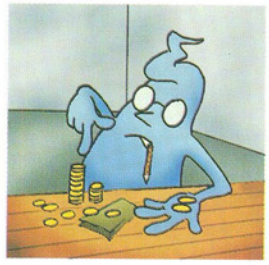
4.5 Άπληστη μέθοδος

Σε πολλά προβλήματα βελτιστοποίησης, όπου απαιτείται να βρεθεί η μέγιστη ή η ελάχιστη τιμή ενός μεγέθους, χρησιμοποιείται μία μέθοδος, που δεν βρίσκει πράγματι τη βέλτιστη λύση, αλλά σκοπός της είναι να την προσεγγίσει. Συνήθως, οι αλγόριθμοι αυτού του τύπου προχωρούν με βάση σταδιακές επιλογές που αφορούν στο βέλτιστο κάθε βήματος, χωρίς μέριμνα για το τελικό βέλτιστο. Έτσι η λειτουργία τους είναι παρόμοια με τις κινήσεις ενός παίκτη στο σκάκι που σκέφτεται και επιλέγει την επόμενη κίνηση χωρίς να τον απασχολούν οι επόμενες κινήσεις ούτε του αντιπάλου ούτε οι δικές του. Η πρώτη προσέγγιση στο παράδειγμα του ταχυδρόμου, το οποίο περιγράφεται στο Κεφάλαιο 4.1, ανήκει στην κατηγορία αυτή. Η μέθοδος αυτή που είναι γνωστή και ως "*άπληστη μέθοδος*" (greedy method), μπορεί να τυποποιηθεί σύμφωνα με τις εξής δύο παραδοχές:

- σε κάθε βήμα επιλογή της τρέχουσας βέλτιστης επιλογής
- επιβεβαίωση ότι αυτή η προσέγγιση εγγυάται τη συνολική βέλτιστη λύση (όταν ο αλγόριθμος δεν χρησιμοποιείται ως μια απλή ευριστική προσεγγιστική τακτική).

Η χρήση των αλγορίθμων αυτού του τύπου πρέπει να γίνεται με στόχο την ευφυέστερη επίλυση του προβλήματος, όπως παρουσιάζεται και στο παράδειγμα που ακολουθεί, όπου πράγματι βρίσκεται η βέλτιστη λύση.

Υπολογισμός νομισμάτων



Το νομισματικό μας σύστημα από 1/1/2002 διαθέτει κέρματα ονομαστικής αξίας των 1, 2, 5, 10, 20, 50 λεπτών, του 1 και των 2 ευρώ, καθώς και χαρτονομίσματα των 5, 10, 20, 50, 100, 200 και 500ευρώ. Το πρόβλημα είναι να βρεθεί αλγόριθμος με βάση τον οποίο, όταν δίδεται ένα ποσό, να βρίσκεται ο ελάχιστος αριθμός των απαιτούμενων κερμάτων και χαρτονομισμάτων. Για παράδειγμα για να σχηματιστεί το ποσό των 789€ απαιτούνται 1 χαρτονόμισμα των 500€, 1 των 200€, 1 των 50€, 1 των 20€, 1 των 10€, 1 των 5€ και 2 κέρματα των 2€, δηλαδή συνολικά 8 νομίσματα.

Το πρόβλημα αυτό μπορεί να επιλυθεί με "άπληστη" προσέγγιση, όπου κάθε στιγμή θα γίνεται επιλογή του νομίσματος που αντιστοιχεί στο μεγαλύτερο δυνατό ποσό. Τα βήματα επιλογής νομισμάτων περιγράφονται στον επόμενο αλγόριθμο, που παρουσιάζει μία απλοποιημένη έκδοση του προβλήματος.

```
Αλγόριθμος Νομίσματα
Δεδομένα  // C, n, poso //
find ← poso
coins ← 0
choice ← n
Όσο (choice>0) και (find > 0) επανάλαβε
    Αν C[choice]<= find τότε
        coins ← coins + 1
        find ← find - C[choice]
    αλλιώς
        choice ← choice - 1
    Τέλος_Αν
Τέλος_επανάληψης
Αποτελέσματα // coins //
Τέλος Νομίσματα
```

Στον αλγόριθμο αυτό ο πίνακας C διαθέτει όλες τις τιμές των n χρησιμοποιούμενων νομισμάτων από το μικρότερο προς το μεγαλύτερο. Ο αλγόριθμος υπολογίζει τον ελάχιστο αριθμό νομισμάτων για ένα συγκεκριμένο ποσό (μεταβλητή coins). Μπορεί να γίνει επέκτασή του, ώστε να έχει ως αποτέλεσμα τον αριθμό για κάθε είδος νομίσματος.

Ανακεφαλαίωση

Αρχικά δόθηκε ιδιαίτερη έμφαση στην έννοια και το ρόλο της ανάλυσης προβλημάτων πριν από την τελική επίλυσή τους με χρήση κάποιου αλγορίθμου. Τα μέρη της ανάλυσης ενός προβλήματος παρουσιάσθηκαν διεξοδικά και με χρήση παραδειγμάτων. Οι μέθοδοι σχεδίασης αλγορίθμων τυποποιήθηκαν και αναλύθηκαν με χρήση δύο βασικών προσεγγίσεων: της μεθόδου Διαίρει και Βασίλευε και του Δυναμικού Προγραμματισμού, που αποτελούν δύο συμπληρωματικές τεχνικές με αντίθετη φιλοσοφία. Ειδικότερα για την τεχνική Διαίρει και Βασίλευε παρουσιάσθηκε ο αναδρομικός αλγόριθμος της δυαδικής αναζήτησης. Ο αλγόριθμος αυτός είναι τυπικό παράδειγμα μιας κλάσης προβλημάτων που έχουν επιλυθεί με τέτοιου είδους προσέγγιση. Σε σχέση με τη μέθοδο του Δυναμικού Προγραμματισμού δόθηκε η λογική της λειτουργίας της και παρουσιάσθηκαν αλγόριθμοι και παραδείγματα για την επίλυση απλών προβλημάτων όπως η ύψωση σε δύναμη. Επίσης, παρουσιάσθηκε η φιλοσοφία της λεγόμενης Άπληστης μεθόδου και δόθηκε ένα πιο σύνθετο παράδειγμα, όπως είναι η επιστροφή από ρέστα με το μικρότερο αριθμό κερμάτων και χαρτονομισμάτων.

Λέξεις κλειδιά

Διαίρει και βασίλευε, Δυναμικός προγραμματισμός, Άπληστη μέθοδος, Δυαδική αναζήτηση, Μέθοδος διχοτόμησης.

Ερωτήσεις - Θέματα για συζήτηση

1. Τι περιλαμβάνει η ανάλυση ενός προβλήματος σε ένα σύγχρονο υπολογιστικό περιβάλλον;
2. Ποιοί είναι οι λόγοι για τους οποίους οι μέθοδοι ανάλυσης και επίλυσης των προβλημάτων παρουσιάζουν ιδιαίτερο ενδιαφέρον;
3. Ποιά είναι τα δύο βασικά είδη προσέγγισης της επίλυσης ενός προβλήματος;

4. Να περιγραφεί με βήματα η μέθοδος της προσέγγισης Διαίρει και Βασίλευε για τη σχεδίαση αλγορίθμων.
5. Να περιγραφεί με βήματα η μέθοδος της προσέγγισης του Δυναμικού Προγραμματισμού για τη σχεδίαση αλγορίθμων.
6. Να περιγραφεί η μέθοδος της δυαδικής αναζήτησης και να δοθεί ένα παράδειγμα επίλυσης προβλήματος με τη μέθοδο αυτή.
7. Να δοθεί αλγόριθμος για την ανεύρεση του παραγοντικού ακεραίου με χρήση της από προσέγγισης, όπου τα προσωρινά αποτελέσματα αποθηκεύονται σε έναν πίνακα.
8. Να δοθεί αλγόριθμος για την ανεύρεση της δύναμης πραγματικού σε αρνητικό ακέραιο εκθέτη με Δυναμικό Προγραμματισμό.

Βιβλιογραφία

1. Φώτω Αφράτη και Γιώργος Παπαγεωργίου: Αλγόριθμοι – Μέθοδοι Σχεδίασης και Ανάλυση Πολυπλοκότητας, Εκδόσεις Συμμετρία, Αθήνα, 1993.
2. Χρήστος Κοίλιας: Δομές Δεδομένων και Οργάνωση Αρχείων. Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1993.
3. Μανόλης Λουκάκης: Δομές Δεδομένων και Αλγόριθμοι, Έκδοσεις Θεραπευτικής Κοινότητας Ιθάκη, Θεσσαλονίκη, 1988.
4. Ιωάννης Μανωλόπουλος: Δομές Δεδομένων – μία Προσέγγιση με Pascal, Εκδόσεις Art of Text, Θεσσαλονίκη, 1998.
5. Ιωάννης Μανωλόπουλος, Μαθήματα Θεωρίας Γράφων, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1996.
6. Νίκος Μισυρλής, Δομές Δεδομένων, Αθήνα, 1995.
7. Niklaus Wirth: Αλγόριθμοι και Δομές Δεδομένων, Κλειδάριθμος, Αθήνα, 1990.
8. G. Brassard and P.Bratley: *"Fundamentals of Algorithms"*, Prentice Hall, 1996.
9. T. Cormen, C. Leiserson and R. Rivest: *"Introduction to Algorithms"* MIT Press, 1990.
10. E. Horowitz, S. Sahni and S. Rajasekaran: *"Computer Algorithms"*, Computer Science Press, 1998.

11. I. Oliver: "Programming Classics: Implementing the World's Best Algorithms", Prentice Hall, 1993.

Διευθύνσεις Διαδικτύου

- http://www.cs.pitt.edu/~kirk/algorithmcourses/index.html

Αποτελεί κόμβο με πλούσιο πληροφορικό υλικό υψηλού ακαδημαϊκού επιπέδου. Περιέχει και στοιχεία βασικού επιπέδου.