Περιεχόμενα

ΜΕΡΟΣ ΤΡΙΑ Ταξινόμηση 291

Πρόλογος

ΑΥΤΟ ΤΟ ΒΙΒΛΙΟ, στην πρωτότυπη αμερικανική του έκδοση, είναι το πρώτο από μια σειρά τριών τόμων που έχουν σκοπό τη μελέτη των σημαντικότερων αλγορίθμων που χρησιμοποιούνται σήμερα στους υπολογιστές. Ο πρώτος τόμος (Μέρη 1-4) καλύπτει τις θεμελιώδεις έννοιες (Μέρος 1), τις δομές δεδομένων (Μέρος 2), τους αλγορίθμους ταξινόμησης (Μέρος 3), και τους αλγορίθμους αναζήτησης (Μέρος 4). Ο δεύτερος τόμος (Μέρος 5) καλύπτει τους γράφους και τους αλγορίθμους γράφων, ενώ ο τρίτος τόμος (Μέρη 6-8) — ο οποίος δεν είχε κυκλοφορήσει ακόμη όταν μεταφραζόταν αυτό το βιβλίο — καλύπτει τα αλφαριθμητικά (Μέρος 6), την υπολογιστική γεωμετρία (Μέρος 7), και προηγμένους αλγορίθμους και εφαρμογές (Μέρος 8).

Τα συγκεκριμένα βιβλία είναι χρήσιμα ως συγγράμματα σχετικά νωρίς στο πρόγραμμα μαθημάτων της επιστήμης των υπολογιστών, αφού αποκτήσουν οι σπουδαστές βασικές προγραμματιστικές δεξιότητες και οικειότητα με τα συστήματα υπολογιστών, αλλά πριν αρχίσουν να παρακολουθούν εξειδικευμένα μαθήματα σε προηγμένους τομείς της επιστήμης των υπολογιστών ή των εφαρμογών υπολογιστών. Τα βιβλία είναι επίσης χρήσιμα και ως μέσα αυτοδιδασκαλίας ή ως οδηγοί αναφοράς για τα άτομα που ασχολούνται με την ανάπτυξη συστημάτων υπολογιστών ή προγραμμάτων εφαρμογών, επειδή περιέχουν υλοποιήσεις χρήσιμων αλγορίθμων και λεπτομερείς πληροφορίες για τα χαρακτηριστικά επιδόσεων αυτών των αλγορίθμων. Η ευρεία προοπτική που ακολουθείται κάνει αυτή τη σειρά βιβλίων κατάλληλη ως εισαγωγή στο συγκεκριμένο γνωστικό πεδίο.

Στο σύνολό τους, οι τρεις τόμοι αποτελούν την *Τρίτη Αμερικανική Έκδοση* ενός βιβλίου το οποίο χρησιμοποιείται ευρέως από σπουδαστές και προγραμματιστές σε ολόκληρο τον κόσμο για πολλά χρόνια. Έχω ξαναγράψει από την αρχή το κείμενο αυτής της έκδοσης και έχω προσθέσει χιλιάδες νέες ασκήσεις, εκατοντάδες νέες εικόνες, δεκάδες νέα προγράμματα, και λεπτομερή σχόλια σε όλες τις εικόνες και τα προγράμματα. Στόχος μου με την προσθήκη αυτού του νέου υλικού ήταν να καλύψω νέα θέματα αλλά και να εξηγήσω αναλυτικότερα πολλούς από τους κλασικούς αλγορίθμους. Η έμφαση που δίνεται (σε όλα τα βιβλία) στους αφηρημένους τύπους δεδομένων, κάνει τα προγράμματα χρήσιμα σε ένα μεγαλύτερο εύρος εφαρμογών και κατάλληλα για σύγχρονα αντικειμενοστρεφή περιβάλλοντα προγραμματισμού. Οι αναγνώστες που έχουν διαβάσει τις προηγούμενες εκδόσεις θα βρουν σε αυτή τη νέα έκδοση αρκετές νέες πληροφορίες, και όλοι οι αναγνώστες θα βρουν πλούσιο διδακτικό υλικό που παρέχει αποτελεσματική πρόσβαση στις βασικές έννοιες.

Αυτά τα βιβλία δεν είναι κατάλληλα μόνο για προγραμματιστές και σπουδαστές της επιστήμης των υπολογιστών. Όλοι όσοι χρησιμοποιούν υπολογιστές θέλουν τα προγράμματα που χρησιμοποιούν να εκτελούνται ταχύτερα ή να λύνουν μεγαλύτερα προβλήματα. Οι αλγόριθμοι που παρουσιάζουμε αντιπροσωπεύουν γνώσεις οι οποίες αναπτύχθηκαν τα τελευταία

50 χρόνια και αποτελούν τη βάση για αποδοτική χρήση του υπολογιστή σε μια ευρεία κλίμακα εφαρμογών. Από τα προβλήματα προσομοίωσης του προβλήματος των N σωμάτων στη φυσική μέχρι τα προβλήματα των γενετικών ακολουθιών στη μοριακή βιολογία, οι βασικές μέθοδοι που περιγράφονται εδώ έχουν γίνει απαραίτητες στην επιστημονική έρευνα· επίσης, από τα συστήματα βάσεων δεδομένων μέχρι της μηχανές αναζήτησης στο Διαδίκτυο, έχουν γίνει απαραίτητα τμήματα των σύγχρονων συστημάτων λογισμικού. Καθώς το πεδίο των εφαρμογών υπολογιστών γίνεται όλο και μεγαλύτερο, ταυτόχρονα αυξάνεται και η επίδραση των βασικών αλγορίθμων. Στόχος του παρόντος βιβλίου είναι να λειτουργήσει ως πηγή αναφοράς, έτσι ώστε οι σπουδαστές και οι επαγγελματίες να γνωρίσουν αυτούς τους θεμελιώδεις αλγορίθμους προκειμένου να μπορούν να τους χρησιμοποιούν με έξυπνους τρόπους όποτε παρουσιάζεται ανάγκη, σε οποιαδήποτε εφαρμογή υπολογιστή.

Στόχοι του βιβλίου

Το βιβλίο *Αλγόριθμοι σε C++*, *Τρίτη αμερικανική έκδοση*, *Μέρη 1-4*, αποτελείται από 16 κεφάλαια τα οποία έχουν ομαδοποιηθεί σε τέσσερα κύρια μέρη: θεμελιώδεις έννοιες, δομές δεδομένων, ταξινόμηση, και αναζήτηση. Το βιβλίο έχει στόχο να δώσει στους αναγνώστες τη δυνατότητα να κατανοήσουν τις βασικές ιδιότητες όσο το δυνατό περισσότερων θεμελιωδών αλγορίθμων. Οι αλγόριθμοι που περιγράφονται χρησιμοποιούνται ευρέως για πολλά χρόνια, και αντιπροσωπεύουν ένα βασικό τμήμα γνώσης τόσο για τον ασκούμενο προγραμματιστή όσο και για το σπουδαστή της επιστήμης των υπολογιστών. Περιγράφονται πολλές έξυπνες μέθοδοι, από τις διωνυμικές ουρές έως τα patricia trie, οι οποίες σχετίζονται με βασικά παραδείγματα που βρίσκονται στην καρδιά της επιστήμης των υπολογιστών. Ο δεύτερος τόμος είναι αφιερωμένος σε αλγορίθμους γράφων, και ο τρίτος αποτελείται από τέσσερα μέρη που καλύπτουν τα αλφαριθμητικά, τη γεωμετρία, και προχωρημένα θέματα. Ο κύριος στόχος μου στη συγγραφή αυτών των βιβλίων ήταν να συγκεντρώσω τις θεμελιώδεις μεθόδους αυτών των τομέων της επιστήμης των υπολογιστών, έτσι ώστε να κάνω προσιτές τις καλύτερες γνωστές μεθόδους για την επίλυση προβλημάτων με υπολογιστή.

Θα εκτιμήσετε ακόμη περισσότερο το υλικό του βιβλίου αν έχετε ήδη παρακολουθήσει ένα ή δύο προηγούμενα μαθήματα της επιστήμης των υπολογιστών ή διαθέτετε αντίστοιχη εμπειρία στον προγραμματισμό: ένα μάθημα στον προγραμματισμό σε γλώσσα υψηλού επιπέδου όπως η C, η C++, ή η Java, και ίσως ένα ακόμη μάθημα στο οποίο διδάσκονται οι θεμελιώδεις έννοιες των συστημάτων προγραμματισμού. Κατά συνέπεια, το βιβλίο απευθύνεται σε όλους εκείνους που έχουν επαφή με κάποια σύγχρονη γλώσσα προγραμματισμού και σε εκείνους που ασχολούνται με τα βασικά χαρακτηριστικά των σύγχρονων συστημάτων υπολογιστών. Επίσης, στο κείμενο θα βρείτε παραπομπές σε βιβλιογραφία που ίσως σας βοηθήσει να καλύψετε κάποια κενά στα θέματα που περιγράφονται.

Επειδή το μεγαλύτερο μέρος της ύλης των μαθηματικών με το οποίο υποστηρίζονται τα αναλυτικά αποτελέσματα είναι αυτοτελές (ή επισημαίνεται ότι ξεφεύγει από τους στόχους του βιβλίου), δεν χρειάζεται πολλή προετοιμασία στα μαθηματικά για το μεγαλύτερο τμήμα του βιβλίου — αν και η μαθηματική ωριμότητα οπωσδήποτε είναι χρήσιμη.

Χρήση του βιβλίου ως διδακτέας ύλης

Υπάρχει μεγάλη ευελιξία σε ό,τι αφορά τον τρόπο με τον οποίο μπορεί να διδαχθεί η ύλη του βιβλίου· εξαρτάται από τον καθηγητή και την προετοιμασία των σπουδαστών του. Γίνεται επαρκής κάλυψη της βασικής ύλης, έτσι ώστε να μπορεί το βιβλίο να χρησιμοποιηθεί για τη διδασκαλία αρχαρίων στις δομές δεδομένων, ενώ ταυτόχρονα παρέχονται επαρκείς λεπτομέρειες και κάλυψη του προχωρημένου υλικού προκειμένου να μπορεί να χρησιμοποιηθεί και στη διδασκαλία της σχεδίασης και ανάλυσης αλγορίθμων σε σπουδαστές πιο προχωρημένου επιπέδου. Μερικοί καθηγητές μπορεί να προτιμήσουν να δώσουν έμφαση στις υλοποιήσεις και στα πρακτικά θέματα, ενώ κάποιοι άλλοι μπορεί να επιθυμούν να δώσουν έμφαση στην ανάλυση και τις θεωρητικές έννοιες.

Σε ένα εισαγωγικό μάθημα στις δομές δεδομένων και τους αλγορίθμους, θα μπορούσε να δοθεί έμφαση στις βασικές δομές δεδομένων του Μέρους 2 και τη χρήση τους στις υλοποιήσεις των Μερών 3 και 4. Σε ένα μάθημα σχεδίασης και ανάλυσης αλγορίθμων θα μπορούσε να δοθεί έμφαση στο θεμελιώδες υλικό του Μέρους 1 και στο Κεφάλαιο 5, και στη συνέχεια να μελετηθούν οι τρόποι με τους οποίους οι αλγόριθμοι των Μερών 3 και 4 επιτυγχάνουν καλές ασυμπτωτικές επιδόσεις. Σε ένα μάθημα τεχνολογίας λογισμικού, θα μπορούσε να παραλειφθεί η ύλη των μαθηματικών καθώς και η προχωρημένη ύλη των αλγορίθμων, και να δοθεί έμφαση στους τρόπους ολοκλήρωσης των υλοποιήσεων που περιλαμβάνονται στο βιβλίο σε μεγάλα προγράμματα ή συστήματα. Τέλος, σε ένα μάθημα με θέμα τους αλγορίθμους, θα μπορούσε να ακολουθηθεί μια γενική προσέγγιση και να εισαχθούν έννοιες από όλους τους παραπάνω τομείς.

Οι προηγούμενες εκδόσεις του βιβλίου, οι οποίες βασίζονται σε άλλες γλώσσες προγραμματισμού, έχουν χρησιμοποιηθεί σε πάρα πολλά κολέγια και πανεπιστήμια ως διδακτέα ύλη για το δεύτερο ή τρίτο μάθημα στην επιστήμη των υπολογιστών, και ως βοηθητικό εγχειρίδιο για άλλα μαθήματα. Στο Princeton, έχουμε διαπιστώσει ότι η ύλη που καλύπτει το βιβλίο παρέχει στους σπουδαστές μας μια εισαγωγή στην επιστήμη των υπολογιστών η οποία μπορεί να επεκταθεί σε μεταγενέστερα μαθήματα που αφορούν την ανάλυση αλγορίθμων, τον προγραμματισμό συστημάτων, και τη θεωρητική επιστήμη των υπολογιστών· από την άλλη μεριά, παρέχει στην αυξανόμενη ομάδα των σπουδαστών από άλλους επιστημονικούς κλάδους ένα ευρύ σύνολο τεχνικών τις οποίες μπορούν να χρησιμοποιήσουν αμέσως.

Οι ασκήσεις — από τις οποίες σχεδόν όλες είναι καινούργιες σε αυτή την τρίτη έκδοση — ανήκουν σε διάφορους τύπους. Μερικές έχουν στόχο να ελέγξουν αν έγινε κατανοητή η ύλη του βιβλίου, και απλώς ζητούν από τους αναγνώστες να εργαστούν σε κάποιο παράδειγμα ή να εφαρμόσουν βασικές έννοιες που περιγράφονται στο βιβλίο. Άλλες περιλαμβάνουν την υλοποίηση και τη συναρμολόγηση των αλγορίθμων, ή την εκτέλεση πειραματικών μελετών με στόχο τη σύγκριση διαφόρων παραλλαγών των αλγορίθμων και την καλύτερη κατανόηση των ιδιοτήτων τους. Κάποιες άλλες, πάλι, είναι ένας θησαυρός σημαντικών πληροφοριών, σε επίπεδο λεπτομέρειας το οποίο δεν είναι κατάλληλο για το βιβλίο. Η μελέτη και η διερεύνηση των ασκήσεων θα βοηθήσουν σε μεγάλο βαθμό τον αναγνώστη να κατανοήσει τα θέματα που περιγράφονται στο βιβλίο.

Αλγόριθμοι πρακτικής χρήσης

Οποιοσδήποτε θέλει να χρησιμοποιήσει τον υπολογιστή του αποτελεσματικότερα μπορεί να χρησιμοποιήσει αυτό το βιβλίο ως οδηγό αναφοράς ή για αυτοδιδασκαλία. Όσοι έχουν πείρα στον προγραμματισμό μπορούν να βρουν στα κεφάλαια του βιβλίου πληροφορίες που αφορούν συγκεκριμένα θέματα. Ως ένα μεγάλο βαθμό, μπορείτε να διαβάσετε κάθε κεφάλαιο του βιβλίου ανεξάρτητα από τα υπόλοιπα, παρά το γεγονός ότι, σε μερικές περιπτώσεις, ορισμένοι αλγόριθμοι ενός κεφαλαίου χρησιμοποιούν μεθόδους από κάποιο προηγούμενο κεφάλαιο.

Το βιβλίο είναι προσανατολισμένο στη μελέτη των αλγορίθμων που είναι πιθανό να έχουν πρακτική χρήση. Παρέχονται πληροφορίες σχετικά με τα εργαλεία του εμπορίου, έτσι ώστε οι αναγνώστες να μπορούν με βεβαιότητα να υλοποιήσουν, να αποσφαλματώσουν, και να χρησιμοποιήσουν στην πράξη αλγορίθμους για την επίλυση ενός προβλήματος ή την προσθήκη επιπλέον λειτουργιών σε κάποια εφαρμογή. Περιλαμβάνονται πλήρεις υλοποιήσεις των μεθόδων που εξετάζονται, όπως επίσης και περιγραφές των λειτουργιών αυτών των προγραμμάτων σε ένα συνεπές σύνολο παραδειγμάτων.

Επειδή εργαζόμαστε με πραγματικό κώδικα αντί να γράφουμε ψευδοκώδικα, μπορείτε να χρησιμοποιήσετε αυτά τα προγράμματα στην πράξη πολύ γρήγορα. Τα προγράμματα είναι διαθέσιμα και στην ηλεκτρονική σελίδα του βιβλίου στο Διαδίκτυο. Μπορείτε να χρησιμοποιήσετε αυτά τα προγράμματα με πολλούς τρόπους προκειμένου να σας βοηθήσουν στη μελέτη των αλγορίθμων. Διαβάστε τα για να ελέγξετε πόσο έχετε κατανοήσει τις λεπτομέρειες ενός αλγορίθμου, ή για να δείτε κάποιον τρόπο χειρισμού ανάθεσης αρχικών τιμών, οριακών συνθηκών, και άλλων περίεργων καταστάσεων που αποτελούν συχνά προγραμματιστικές προκλήσεις. Εκτελέστε τα για να δείτε τους αλγορίθμους σε δράση, να μελετήσετε πειραματικά τις επιδόσεις τους, και να ελέγξετε τα αποτελέσματά σας σε σχέση με τους πίνακες που περιέχονται στο βιβλίο, ή να δοκιμάσετε τις δικές σας τροποποιήσεις.

Μια από τις πρακτικές εφαρμογές των αλγορίθμων είναι και η χρήση τους για την παραγωγή των εκατοντάδων εικόνων που περιλαμβάνονται σε αυτό το βιβλίο. Πολλοί αλγόριθμοι αποκαλύπτονται σε διαισθητικό επίπεδο μέσω της οπτικοποιημένης τους διάστασης η οποία παρέχεται από αυτές τις εικόνες.

Στο βιβλίο εξετάζονται λεπτομερώς τα χαρακτηριστικά των αλγορίθμων και των περιπτώσεων στις οποίες μπορεί να είναι χρήσιμοι, ενώ παράλληλα γίνονται συνδέσεις με την ανάλυση αλγορίθμων και τη θεωρητική επιστήμη των υπολογιστών. Όποτε κρίνεται απαραίτητο, παρουσιάζονται πειραματικά και αναλυτικά αποτελέσματα προκειμένου να διασαφηνιστεί ο λόγος για τον οποίο προτιμούνται κάποιοι συγκεκριμένοι αλγόριθμοι. Όποτε παρουσιάζει ενδιαφέρον, περιγράφεται και η σχέση των πρακτικών αλγορίθμων που εξετάζονται με αμιγώς θεωρητικά αποτελέσματα. Επίσης, οι ειδικές πληροφορίες σχετικά με τα χαρακτηριστικά επιδόσεων των αλγορίθμων και των υλοποιήσεων συγκεντρώνονται, ενσωματώνονται στο κείμενο, και αναλύονται σε ολόκληρο το βιβλίο.

Γλώσσα προγραμματισμού

Η γλώσσα προγραμματισμού που χρησιμοποιείται για όλες τις υλοποιήσεις των αλγορίθμων είναι η C++. Στα προγράμματα χρησιμοποιείται μια ευρεία κλίμακα καθιερωμένων ιδιωματισμών της C++, και στο βιβλίο περιλαμβάνονται περιεκτικές περιγραφές κάθε κατασκευής.

Ο Chris Van Wyk και εγώ αναπτύξαμε ένα στυλ προγραμματισμού σε C++ το οποίο βασίζεται σε κλάσεις, πρότυπα, και υπερφορτωμένους τελεστές· νομίζουμε ότι είναι ένας αποτελεσματικός τρόπος για να παρουσιάζουμε τους αλγορίθμους και τις δομές δεδομένων ως πραγματικά προγράμματα. Κάναμε αρκετή προσπάθεια για να δώσουμε κομψές, συμπαγείς, αποδοτικές, και φορητές υλοποιήσεις. Το στυλ προγραμματισμού παρουσιάζει συνέπεια όποτε αυτό είναι δυνατό, έτσι ώστε τα προγράμματα που είναι παρόμοια να φαίνονται και παρόμοια.

Στην περίπτωση πολλών από τους αλγορίθμους του βιβλίου, οι ομοιότητες ισχύουν ανεξάρτητα από τη γλώσσα: ο αλγόριθμος ταξινόμησης quicksort είναι πάντοτε ο αλγόριθμος ταξινόμησης quicksort (για να επιλέξουμε ένα εξέχον παράδειγμα), ανεξάρτητα από το αν εκφράζεται σε Ada, Algol-60, Basic, C, C++, Fortran, Java, Mesa, Modula-3, Pascal, PostScript, Smalltalk, ή σε οποιαδήποτε από τις αμέτρητες άλλες γλώσσες προγραμματισμού και περιβάλλοντα όπου έχει αποδειχθεί ότι είναι μια αποτελεσματική μέθοδος ταξινόμησης. Από τη μια πλευρά, ο κώδικάς μας παίρνει πληροφορίες από τις εμπειρίες υλοποίησης αλγορίθμων σε αυτές και σε πολλές άλλες γλώσσες προγραμματισμού (υπάρχουν εκδόσεις του βιβλίου και για τις γλώσσες C και Java)· από την άλλη, ορισμένες από τις ιδιότητες μερικών από αυτές τις γλώσσες εμπλουτίζονται από την εμπειρία των σχεδιαστών τους σε μερικούς από τους αλγορίθμους και τις δομές δεδομένων που εξετάζουμε σε αυτό το βιβλίο.

Το Κεφάλαιο 1 αποτελεί λεπτομερές παράδειγμα αυτής της προσέγγισης στην ανάπτυξη αποδοτικών υλοποιήσεων των αλγορίθμων μας σε C++, ενώ στο Κεφάλαιο 2 περιγράφεται η προσέγγιση που ακολουθούμε για την ανάλυσή τους. Τα Κεφάλαια 3 και 4 είναι αφιερωμένα στην περιγραφή και την αιτιολόγηση των βασικών μηχανισμών που χρησιμοποιούμε στις υλοποιήσεις τύπων δεδομένων και αφηρημένων τύπων δεδομένων (ΑΤΔ). Αυτά τα τέσσερα κεφάλαια αποτελούν τα θεμέλια για τα επόμενα κεφάλαια του βιβλίου.

Ευχαριστίες

Η ανταπόκριση πολλών ανθρώπων από τις παλιότερες εκδόσεις αυτού του βιβλίου ήταν ιδιαίτερα χρήσιμη. Πιο συγκεκριμένα, εκατοντάδες φοιτητές των πανεπιστημίων Princeton και Brown υπέστησαν τις προκαταρκτικές εκδόσεις για αρκετά χρόνια. Θα ήθελα να εκφράσω ιδιαίτερες ευχαριστίες στην Trina Avery και τον Tom Freeman για τη βοήθειά τους στη δημιουργία της πρώτης έκδοσης· στην Janet Incerpi για τη δημιουργικότητα και την εξυπνάδα της που της επέτρεψαν να πειθαναγκάσει το πρωτόγονο υλικό και λογισμικό συγγραφής κειμένου που χρησιμοποιούσαμε προκειμένου να δημιουργήσουν την πρώτη έκδοση· στον Marc Brown για τη συμμετοχή του στην έρευνα οπτικοποίησης των αλγορίθμων, η οποία οδήγησε στη δημιουργία πολλών από τις εικόνες του βιβλίου· και στους Dave Hanson και Andrew Appel για την προθυμία που έδειξαν να απαντήσουν σε όλες τις ερωτήσεις μου σχετικά με τις γλώσσες προγραμματισμού. Θα ήθελα επίσης να ευχαριστήσω όλους τους αναγνώστες που έκαναν σχόλια σχετικά με τις διάφορες εκδόσεις του βιβλίου, μεταξύ των οποίων είναι και οι Guy Almes, Jon Bentley, Marc Brown, Jay Gischer, Allan Heydon, Kennedy Lemke, Udi Manber, Dana Richards, John Reif, M. Rosenfeld, Stephen Seidman, Michael Quinn, και William Ward.

Για να δημιουργήσω αυτή τη νέα έκδοση είχα την ευχαρίστηση να συνεργαστώ με τους Peter Gordon, Debbie Lafferty, και Helen Goldstein της Addison-Wesley, οι οποίοι και καθοδήγησαν με υπομονή αυτό το έργο καθώς εξελισσόταν. Επίσης, είχα την ευχαρίστηση να συνεργαστώ με πολλά άλλα μέλη του εξειδικευμένου προσωπικού της Addison-Wesley. Η φύση του έργου έκανε το βιβλίο μια, κατά κάποιον τρόπο, ασυνήθιστη πρόκληση για πολλούς από αυτούς, και εκτιμώ ιδιαίτερα την υπομονή τους.

Κέρδισα και τρεις νέους μέντορες κατά τη συγγραφή αυτού του βιβλίου, και θα ήθελα να εκφράσω ιδιαίτερα την εκτίμησή μου σε αυτούς. Ο πρώτος ήταν ο Steve Summit, ο οποίος έλεγξε προσεκτικά τις πρώτες εκδόσεις του βιβλίου σε τεχνικό επίπεδο και έκανε κυριολεκτικά χιλιάδες λεπτομερή σχόλια, ιδιαίτερα για τα προγράμματα. Ο Steve κατάλαβε ξεκάθαρα το στόχο μου να δώσω κομψές, αποδοτικές, και αποτελεσματικές υλοποιήσεις, και τα σχόλιά του, όχι μόνο με βοήθησαν να προσφέρω ένα μέτρο συνέπειας κατά τις υλοποιήσεις, αλλά και να βελτιώσω σημαντικά πολλές από αυτές. Ο δεύτερος ήταν η Lyn Dupré που έκανε και αυτή χιλιάδες λεπτομερή σχόλια, τα οποία ήταν ανεκτίμητης αξίας και με βοήθησαν, όχι μόνο να διορθώσω και να αποφύγω γραμματικά λάθη, αλλά επίσης — που είναι ακόμη σημαντικότερο — να βρω ένα περιεκτικό και συνεπές στυλ συγγραφής το οποίο με βοήθησε να συναρμολογήσω το μεγάλο όγκο του τεχνικού υλικού. Ο τρίτος ήταν ο Chris Van Wyk που υλοποίησε και αποσφαλμάτωσε όλους τους αλγορίθμους μου στη C++, απάντησε σε αναρίθμητες ερωτήσεις μου σχετικά με τη C++, βοήθησε στην ανάπτυξη ενός κατάλληλου στυλ προγραμματισμού σε C++, και διάβασε με μεγάλη προσοχή το κείμενο — και, μάλιστα, δύο φορές. Ο Chris έδειξε επίσης μεγάλη υπομονή όσο "διαμέλιζα" πολλά από τα προγράμματά του της C++ στα "εξ ων συνετέθησαν"· αργότερα, καθώς μάθαινα όλο και περισσότερα πράγματα από αυτόν για τη C++, έπρεπε να τα "συναρμολογήσω" και πάλι, με τον ίδιο σχεδόν τρόπο με τον οποίο τα είχε γράψει εκείνος. Είμαι εξαιρετικά ευγνώμων για την ευκαιρία που μου δόθηκε να μάθω πολλά πράγματα από τους Steve, Lyn, και Chris — η προσφορά τους ήταν ζωτικής σημασίας για την ανάπτυξη αυτού του βιβλίου.

Πολλά από αυτά που γράφονται στο βιβλίο τα έμαθα από τη διδασκαλία και τα γραπτά του Don Knuth, που ήταν ο επιβλέπων καθηγητής μου στο Πανεπιστήμιο Stanford. Παρά το γεγονός ότι ο Don δεν είχε άμεση επίδραση σε αυτή τη δουλειά, η παρουσία του γίνεται αισθητή σε αυτό το βιβλίο επειδή ήταν εκείνος που έθεσε τη μελέτη των αλγορίθμων στην επιστημονική βάση η οποία καθιστά δυνατή μια δουλειά σαν αυτή. Ο φίλος και συνάδελφός μου Philippe Flajolet, ο οποίος αποτέλεσε βασικό παράγοντα για την ανάπτυξη της ανάλυσης αλγορίθμων ως ενός ώριμου τομέα ερευνών, είχε και αυτός παρόμοια επίδραση σε αυτό το έργο.

Θα ήθελα να εκφράσω τις βαθιές μου ευχαριστίες για την υποστήριξη που είχα από το Πανεπιστήμιο Princeton, το Πανεπιστήμιο Brown, και το Εθνικό Ινστιτούτο Έρευνας στην Πληροφορική και τον Αυτοματισμό (Institut National de Recherche en Informatique et Automatique, INRIA), όπου έκανα το μεγαλύτερο μέρος της έρευνας για το βιβλίο· το ίδιο ισχύει και για το Ινστιτούτο Αναλύσεων Άμυνας (Institute for Defense Analyses) καθώς και για το Ερευνητικό Κέντρο της Xerox στο Palo Alto (Xerox Palo Alto Research Center), όπου έκανα ένα μέρος της δουλειάς όσο ήμουν επισκέπτης. Πολλά μέρη του βιβλίου βασίζονται σε έρευνες που υποστηρίχθηκαν γενναιόδωρα από το Εθνικό Ίδρυμα Ερευνών (National Science Foundation) και το Γραφείο Ναυτικών Ερευνών (Office of Naval Research). Τέλος, ευχαριστώ τους Bill Bowen, Aaron Lemonick, και Neil Rudenstine για την υποστήριξή τους στη δημιουργία ακαδημαϊκού περιβάλλοντος στο Princeton, στο οποίο είχα τη δυνατότητα να ετοιμάσω αυτό το βιβλίο παρά τις πολλές άλλες υποχρεώσεις μου.

Robert Sedgewick
Marly-le-Roi, Γαλλία, 1983
Princeton, New Jersey, 1990, 1992
Jamestown, Rhode Island, 1997
Princeton, New Jersey, 1998

Πρόλογος συμβούλου σε θέματα C++

Οι αλγόριθμοι ήταν η αιτία που με οδήγησε να ασχοληθώ με την επιστήμη των υπολογιστών. Για να μελετήσει κανείς τους αλγορίθμους είναι απαραίτητο να μπορεί να σκέφτεται με πολλούς τρόπους: δημιουργικά, για να ανακαλύψει μια ιδέα με την οποία θα λύσει ένα πρόβλημα· λογικά, ώστε να μπορεί να αναλύσει την ορθότητά της· μαθηματικά, για να αναλύσει την αποτελεσματικότητά της· και επιμελώς, προκειμένου να εκφράσει την ιδέα με τη μορφή μιας διαδοχικής σειράς λεπτομερών βημάτων έτσι ώστε να μπορεί να μετατραπεί σε λογισμικό. Επειδή, λοιπόν, η μεγαλύτερη ικανοποίηση που αντλώ από τη μελέτη των αλγορίθμων προέρχεται από την αντίληψή τους ως ολοκληρωμένων προγραμμάτων για υπολογιστή τα οποία να λειτουργούν και να δίνουν αποτελέσματα, άδραξα πραγματικά την ευκαιρία να συνεργαστώ με τον Bob Sedgewick σε ένα βιβλίο για τους αλγορίθμους το οποίο βασίζεται σε προγράμματα C++.

Σε συνεργασία με τον Bob, γράψαμε τα παραδείγματα προγραμμάτων αυτού του βιβλίου χρησιμοποιώντας τις κατάλληλες κάθε φορά δυνατότητες της C++ με τρόπο σαφή και ξεκάθαρο. Χρησιμοποιήσαμε κλάσεις για να διαχωρίσουμε τις προδιαγραφές ενός αφηρημένου τύπου δεδομένων από τις λεπτομέρειες της υλοποίησής του·χρησιμοποιήσαμε πρότυπα και υπερφορτωμένους τελεστές έτσι ώστε τα προγράμματά μας να μπορούν να χρησιμοποιηθούν χωρίς καμία αλλαγή για πολλούς και διάφορους τύπους δεδομένων.

Από την άλλη μεριά, πάντως, αφήσαμε να χαθεί η ευκαιρία να επιδείξουμε πολλές άλλες τεχνικές της C++. Για παράδειγμα, τις περισσότερες φορές παραλείψαμε τουλάχιστον μερικές από τις κατασκευάστριες συναρτήσεις που ήταν απαραίτητες προκειμένου να μετατραπεί μια κλάση σε "πρώτη κλάση" (αλλά στο Κεφάλαιο 4 μπορείτε να μάθετε πώς να το κάνετε αυτό)· στις περισσότερες κατασκευάστριες συναρτήσεις χρησιμοποιήσαμε απλή ανάθεση τιμών αντί για ανάθεση αρχικών τιμών για την αποθήκευση τιμών σε δεδομένα-μέλη·χρησιμοποιήσαμε αλφαριθμητικά χαρακτήρων του στυλ της C αντί να βασιστούμε στα αλφαριθμητικά της βιβλιοθήκης της C++·δεν χρησιμοποιήσαμε τις πιο "ελαφριές" κλάσεις συσκευασίας που θα μπορούσαμε· και χρησιμοποιήσαμε απλούς, αντί για "έξυπνους", δείκτες.

Κάναμε ενσυνείδητα τις περισσότερες από αυτές τις επιλογές προκειμένου να διατηρήσουμε στο επίκεντρο της μελέτης σας τους αλγορίθμους, και να μη σας αποσπάσουμε την προσοχή με λεπτομέρειες τεχνικών οι οποίες χρησιμοποιούνται αποκλειστικά στη C++. Από τη στιγμή, όμως, που θα έχετε κατανοήσει τον τρόπο λειτουργίας των προγραμμάτων αυτού του βιβλίου, θα είστε κατάλληλα καταρτισμένοι για να μάθετε οποιαδήποτε από αυτές τις τεχνικές της C++, έτσι ώστε να εκτιμήσετε τα πλεονεκτήματα που προσφέρουν σε ό,τι αφορά τις επιδόσεις και να θαυμάσετε την εφευρετικότητα των σχεδιαστών της καθιερωμένης βιβλιοθήκης προτύπων (Standard Template Library) της C++.

Είτε ασχολείστε με τη μελέτη των αλγορίθμων για πρώτη φορά είτε έχετε στόχο να "φρεσκάρετε" γνώσεις που είχατε αποκτήσει στο παρελθόν, σας εύχομαι να το απολαύσετε τουλάχιστον τόσο όσο απήλαυσα εγώ τη συνεργασία μου με τον Bob Sedgewick κατά τη συγγραφή των προγραμμάτων.

Ευχαριστώ τους Jon Bentley, Brian Kernighan, και Tom Szymanski οι οποίοι μου δίδαξαν πολλά από αυτά που γνωρίζω για τον προγραμματισμό, την Debbie Lafferty η οποία μου πρότεινε να ασχοληθώ με αυτό το έργο, καθώς και τα εργαστήρια της Bell, το Πανεπιστήμιο Drew, και το Πανεπιστήμιο Princeton για την υποστήριξη που μου παρείχαν.

Christopher Van Wyk
Chatham, New Jersey, 1998

5.7 Αναδρομικοί αλγόριθμοι δυαδικού δένδρου

Οι αλγόριθμοι διάσχισης δένδρου που εξετάσαμε στην Ενότητα 5.6 αποτελούν χαρακτηριστικό δείγμα του βασικού λόγου που μας οδηγεί να μελετήσουμε τους αναδρομικούς αλγορίθμους για δυαδικά δένδρα, λόγω της φύσης των δυαδικών δένδρων ως αναδρομικών δομών. Σε πολλές εργασίες υιοθετούνται άμεσοι αναδρομικοί αλγόριθμοι "διαίρει και βασίλευε", οι οποίοι στην ουσία γενικεύουν τους αλγορίθμους διάσχισης. Επεξεργαζόμαστε ένα δένδρο επεξεργαζόμενοι τον κόμβο της ρίζας και (αναδρομικώς) τα υποδένδρα της ρίζας, και έχουμε τη δυνατότητα να εκτελέσουμε τους υπολογισμούς πριν τις αναδρομικές κλήσεις, μεταξύ των αναδρομικών κλήσεων, ή μετά από αυτές (ή πιθανόν και με τους τρεις τρόπους).

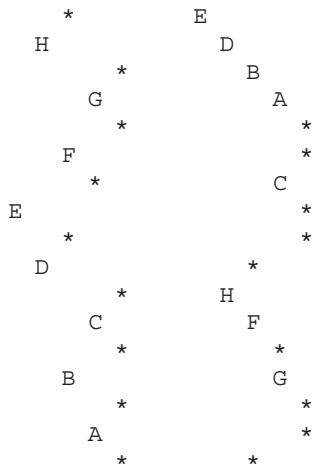
Συχνά χρειάζεται να βρούμε τις τιμές διαφόρων δομικών παραμέτρων ενός δένδρου, με δεδομένο μόνο ένα σύνδεσμο προς το δένδρο. Για παράδειγμα, το Πρόγραμμα 5.17 αποτελείται από αναδρομικές συναρτήσεις υπολογισμού του πλήθους των κόμβων τους οποίους περιέχει κάποιο δένδρο, καθώς και του ύψους του δένδρου. Οι συναρτήσεις προκύπτουν άμεσα από τον Ορισμό 5.6. Καμία από αυτές τις συναρτήσεις δεν εξαρτάται από τη σειρά με την οποία γίνεται η επεξεργασία των αναδρομικών κλήσεων — για παράδειγμα, αν εναλλάξουμε τις αναδρομικές κλήσεις, οι συναρτήσεις θα επεξεργαστούν όλους τους κόμβους του δένδρου και θα επιστρέψουν την ίδια απάντηση. Πάντως, δεν υπολογίζονται τόσο εύκολα και οι τρεις παράμετροι — για παράδειγμα, ένα πρόγραμμα για τον αποδοτικό υπολογισμό του μήκους της εσωτερικής διαδρομής ενός δυαδικού δένδρου αποτελεί σαφώς μεγαλύτερη πρόκληση (δείτε τις Ασκήσεις 5.88 έως 5.90).

Μια άλλη συνάρτηση, η οποία είναι χρήσιμη όποτε γράφουμε προγράμματα επεξεργασίας δένδρων, είναι εκείνη που τυπώνει ή σχεδιάζει τα δένδρα. Για παράδειγμα, το Πρόγραμμα 5.18 είναι μια αναδρομική διαδικασία που εμφανίζει στην οθόνη (ή τυπώνει) ένα δένδρο με τη μορφή που φαίνεται στην Εικόνα 5.29. Μπορούμε να χρησιμοποιήσουμε την ίδια βασική αναδρομική μέθοδο για να σχεδιάσουμε και πιο σύνθετες αναπαραστάσεις δένδρων, σαν αυτές που χρησιμοποιούνται στις εικόνες του βιβλίου (δείτε την Άσκηση 5.85).

Πρόγραμμα 5.17 Υπολογισμός παραμέτρων δένδρου

Μπορούμε να χρησιμοποιήσουμε απλές αναδρομικές διαδικασίες, όπως αυτές που φαίνεται εδώ, για να μάθουμε τις βασικές δομικές ιδιότητες ενός δένδρου.

```
int count(link h)
  {
    if (h == 0) return 0;
    return count(h->l) + count(h->r) + 1;
  }
int height(link h)
  {
    if (h == 0) return -1;
    int u = height(h->l), v = height(h->r);
    if (u > v) return u+1; else return v+1;
  }
```



Εικόνα 5.29
Εκτύπωση ενός δένδρου (ενδοδιατεταγμένη και προδιατεταγμένη διάσχιση)

Η έξοδος στα αριστερά είναι αποτέλεσμα της χρήσης του Προγράμματος 5.18 με το παράδειγμα δένδρου της Εικόνας 5.26 και παρουσιάζει τη δομή του δένδρου με τρόπο παρόμοιο με τη γραφική αναπαράσταση που έχουμε χρησιμοποιήσει μέχρι τώρα, αλλά περιστρεμμένη κατά 90 μοίρες. Η έξοδος στα δεξιά είναι από το ίδιο πρόγραμμα, αλλά η εντολή εκτύπωσης έχει μετακινηθεί στην αρχή· παρουσιάζει τη δομή του δένδρου σε μια οικεία μορφή διάρθρωσης.

Πρόγραμμα 5.18 Γρήγορη συνάρτηση εκτύπωσης δένδρου

Αυτό το αναδρομικό πρόγραμμα παρακολουθεί το ύψος του δένδρου και χρησιμοποιεί αυτή την πληροφορία για τη ρύθμιση των περιθωρίων κατά την εκτύπωση μιας αναπαράστασης του δένδρου, την οποία μπορούμε να χρησιμοποιήσουμε για την αποσφαλμάτωση προγραμμάτων επεξεργασίας δένδρων (δείτε την Εικόνα 5.29). Στο πρόγραμμα θεωρούμε ότι τα στοιχεία των κόμβων είναι τύπου `Item`, για τον οποίο και έχει οριστεί μέσω υπερφόρτωσης ο τελεστής `operator<<`.

```
void printnode(Item x, int h)
  { for (int i = 0; i < h; i++) cout << "  ";
    cout << x << endl;
  }
void show(link t, int h)
  {
    if (t == 0) { printnode('*', h); return; }
    show(t->r, h+1);
    printnode(t->item, h);
    show(t->l, h+1);
  }
```

Στο Πρόγραμμα 5.18 γίνεται ενδοδιατεταγμένη διάσχιση του δένδρου — αν τυπώσουμε το στοιχείο πριν από τις αναδρομικές κλήσεις, θα πάρουμε μια προδιατεταγμένη διάσχιση η οποία παρουσιάζεται επίσης στην Εικόνα 5.29. Αυτή η μορφή μάς είναι οικεία και μπορούμε να τη χρησιμοποιήσουμε, για παράδειγμα, σε κάποιο οικογενειακό δένδρο, ή για την παρουσίαση αρχείων σε ένα σύστημα αρχείων που βασίζεται σε δένδρα, ή για τη διάρθρωση ενός εγγράφου. Για παράδειγμα, με μια προδιατεταγμένη διάσχιση του δένδρου της Εικόνας 5.19 μπορούμε να πάρουμε μια έκδοση του πίνακα περιεχομένων αυτού του βιβλίου.

Το πρώτο μας παράδειγμα προγράμματος που κατασκευάζει μια ρητή δομή δυαδικού δένδρου σχετίζεται με την εφαρμογή εύρεσης του μέγιστου στοιχείου την οποία εξετάσαμε στην Ενότητα 5.2. Στόχος μας είναι να δημιουργήσουμε ένα *τουρνουά*: ένα δυαδικό δένδρο στο οποίο το στοιχείο κάθε εσωτερικού κόμβου είναι αντίγραφο του μεγαλύτερου από τα στοιχεία των δύο παιδιών αυτού του κόμβου. Πιο συγκεκριμένα, το στοιχείο στη ρίζα είναι αντίγραφο του μεγαλύτερου στοιχείου του τουρνουά. Τα στοιχεία στα φύλλα (κόμβοι χωρίς παιδιά) είναι τα δεδομένα που μας ενδιαφέρουν, ενώ το υπόλοιπο δένδρο είναι μια δομή δεδομένων που μας επιτρέπει να βρίσκουμε αποδοτικά το μεγαλύτερο από τα στοιχεία.

Το Πρόγραμμα 5.19 είναι ένα αναδρομικό πρόγραμμα που δημιουργεί ένα τουρνουά από τα στοιχεία του πίνακα. Επειδή αποτελεί επέκταση του Προγράμματος 5.6, χρησιμοποιεί και αυτό μια στρατηγική "διαίρει και βασίλευε". Για τη δημιουργία ενός τουρνουά με ένα και μοναδικό στοιχείο, δημιουργούμε (και επιστρέφουμε) ένα φύλλο που περιέχει αυτό το στοιχείο. Για τη δημιουργία ενός τουρνουά με $N > 1$ στοιχεία, χρησιμοποιούμε τη στρατηγική "διαίρει και βασίλευε" — διχοτομούμε τα στοιχεία, δημιουργούμε ένα τουρνουά για καθένα από τα μισά, και μετά δημιουργούμε ένα νέο κόμβο με συνδέσμους προς τα δύο τουρνουά και με ένα στοιχείο που είναι αντίγραφο του μεγαλύτερου από τα στοιχεία που βρίσκονται στις ρίζες των δύο τουρνουά.

Πρόγραμμα 5.19 Δημιουργία τουρνουά

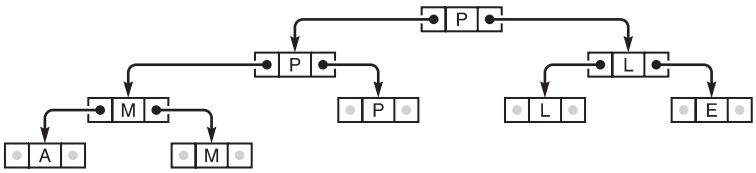
Αυτή η αναδρομική συνάρτηση διαιρεί έναν πίνακα `a[l], …, a[r]` σε δύο μέρη `a[l], …, a[m]` και `a[m+1], …, a[r]`, δημιουργεί από ένα τουρνουά για τα δύο μέρη (αναδρομικώς), και κατασκευάζει και ένα τουρνουά για ολόκληρο τον πίνακα ορίζοντας σε ένα νέο κόμβο συνδέσμους προς τα αναδρομικώς κατασκευασμένα τουρνουά και αναθέτοντάς του τη μεγαλύτερη από τις τιμές που περιέχονται στις ρίζες των δύο αναδρομικώς κατασκευασμένων τουρνουά.

```
struct node
  { Item item; node *l, *r;
    node(Item x)
      { item = x; l = 0; r = 0; }
  };
typedef node* link;
link max(Item a[], int l, int r)
  { int m = (l+r)/2;
    link x = new node(a[m]);
    if (l == r) return x;
    x->l = max(a, l, m);
    x->r = max(a, m+1, r);
    Item u = x->l->item, v = x->r->item;
    if (u > v)
      x->item = u; else x->item = v;
    return x;
  }
```

Η Εικόνα 5.30 αποτελεί παράδειγμα μιας ρητής δομής δένδρου που έχει κατασκευαστεί από το Πρόγραμμα 5.19. Η δημιουργία μιας αναδρομικής δομής δεδομένων όπως αυτή είναι ενδεχομένως προτιμότερη για την εύρεση του μέγιστου στοιχείου με σάρωση των δεδομένων, όπως στο Πρόγραμμα 5.6, επειδή η δομή δένδρου μάς παρέχει την ευελιξία να εκτελούμε άλλες πράξεις. Η βασική πράξη που χρησιμοποιούμε για να δημιουργήσουμε το τουρνουά είναι ένα σημαντικό παράδειγμα — αν έχουμε δύο τουρνουά, μπορούμε να τα συνδυάσουμε σε ένα και μοναδικό τουρνουά σε σταθερό χρόνο, δημιουργώντας ένα νέο κόμβο, κάνοντας τον αριστερό σύνδεσμο του κόμβου να δείχνει στο ένα από τα τουρνουά και το δεξιό σύνδεσμο να δείχνει στο άλλο τουρνουά, και παίρνοντας το μεγαλύτερο από τα δύο στοιχεία (στις ρίζες των δύο τουρνουά) ως το μεγαλύτερο στοιχείο του συνδυασμένου τουρνουά. Επίσης, μπορούμε να εξετάσουμε τη δυνατότητα χρήσης αλγορίθμων προσθήκης στοιχείων, αφαίρεσης στοιχείων, και εφαρμογής άλλων λειτουργιών. Δεν πρόκειται να ασχοληθούμε εδώ με τέτοιες πράξεις, επειδή θα εξετάσουμε παρόμοιες δομές δεδομένων με αυτή την ευελιξία στο Κεφάλαιο 9.

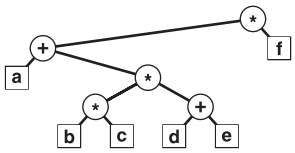
Στην πραγματικότητα, οι υλοποιήσεις πολλών από τους γενικευμένους αφηρημένους τύπους δεδομένων (ΑΤΔ) ουρών που βασίζονται σε δένδρα, τις οποίες εξετάσαμε στην Ενότητα 4.6, αποτελούν κυρίαρχο θέμα σε ένα μεγάλο μέρος του βιβλίου. Πιο συγκεκριμένα, πολλοί από τους αλγορίθμους των Κεφαλαίων 12 έως 15 βασίζονται σε *δένδρα δυαδικής αναζήτησης* (binary search trees), τα οποία είναι ρητές δομές δένδρων που αντιστοιχούν στη δυαδική αναζήτηση, έχοντας σχέση ανάλογη με τη σχέση ανάμεσα στη ρητή δομή της Εικόνας 5.30 και τον αναδρομικό αλγόριθμο εύρεσης του μέγιστου (δείτε την Εικόνα 5.6). Η δυσκολία στην υλοποίηση και τη χρήση τέτοιων δομών είναι να διασφαλίσουμε ότι οι αλγόριθμοί μας θα παραμένουν αποδοτικοί ακόμη και μετά την εκτέλεση μιας μακράς ακολουθίας λειτουργιών *εισαγωγής*, *αφαίρεσης*, και άλλων.

Το δεύτερο παράδειγμα προγράμματος που κατασκευάζει ένα δυαδικό δένδρο είναι μια τροποποίηση του προγράμματος υπολογισμού προθεματικών παραστάσεων που είδαμε στην Ενότητα 5.1 (Πρόγραμμα 5.4) για την κατασκευή ενός δένδρου αναπαράστασης μιας προθεματικής παράστασης αντί απλώς του υπολογισμού της (δείτε την Εικόνα 5.31). Το Πρόγραμμα 5.20 χρησιμοποιεί την ίδια αναδρομική διαδικασία με το Πρόγραμμα 5.4, αλλά η αναδρομική συνάρτηση επιστρέφει ένα σύνδεσμο προς ένα δένδρο, αντί για κάποια τιμή. Για κάθε



Εικόνα 5.30
Ρητό δένδρο για την εύρεση του μέγιστου (τουρνουά)

Σε αυτή την εικόνα παρουσιάζεται η ρητή δομή δένδρου που κατασκευάζεται από το Πρόγραμμα 5.19 για τα δεδομένα εισόδου A M P L E*. Τα στοιχεία δεδομένων βρίσκονται στα φύλλα. Κάθε εσωτερικός κόμβος περιέχει ένα αντίγραφο του μεγαλύτερου από τα στοιχεία των δύο παιδιών του, οπότε, επαγωγικά, το μεγαλύτερο στοιχείο βρίσκεται στη ρίζα.*



Εικόνα 5.31
Δένδρο συντακτικής ανάλυσης

Το δένδρο αυτό δημιουργήθηκε από το Πρόγραμμα 5.20 για την προθεματική παράσταση `* + a * * b c + d e f`. *Αποτελεί φυσικό τρόπο αναπαράστασης της παράστασης: κάθε όρος βρίσκεται σε ένα φύλλο (το οποίο σχεδιάζουμε εδώ ως εξωτερικό κόμβο) και κάθε τελεστής πρόκειται να εφαρμοστεί στις παραστάσεις που αναπαρίστανται από το δεξιό και το αριστερό υποδένδρο του κόμβου που τον περιέχει.*

χαρακτήρα της παράστασης δημιουργούμε ένα νέο κόμβο δένδρου — οι κόμβοι που αντιστοιχούν σε τελεστές έχουν συνδέσμους προς τους όρους τους, ενώ οι κόμβοι-φύλλα περιέχουν τις μεταβλητές (ή σταθερές) που αποτελούν τα δεδομένα εισόδου της παράστασης.

Τα προγράμματα μετάφρασης, όπως οι μεταγλωττιστές, χρησιμοποιούν συχνά τέτοιου είδους εσωτερικές αναπαραστάσεις δομών δένδρου, επειδή τα δένδρα είναι χρήσιμα σε πολλές περιπτώσεις. Για παράδειγμα, μπορούμε να φανταστούμε ότι οι τελεστέοι αντιστοιχούν σε μεταβλητές που παίρνουν τιμές, και να δημιουργήσουμε κώδικα μηχανής για τον υπολογισμό της παράστασης που αντιπροσωπεύεται από το δένδρο, με μεταδιατεταγμένη διάσχιση. Εναλλακτικά, θα μπορούσαμε να χρησιμοποιήσουμε το δένδρο για να εμφανίσουμε την παράσταση σε ενθεματική μορφή με ενδοδιατεταγμένη διάσχιση ή σε επιθεματική μορφή με μεταδιατεταγμένη διάσχιση.

Πρόγραμμα 5.20 Κατασκευή δένδρου συντακτικής ανάλυσης

Αυτό το πρόγραμμα, στο οποίο χρησιμοποιείται η ίδια στρατηγική που χρησιμοποιήσαμε και για τον υπολογισμό προθεματικών παραστάσεων (δείτε το Πρόγραμμα 5.4), δημιουργεί ένα δένδρο συντακτικής ανάλυσης από μια προθεματική παράσταση. Για λόγους απλούστευσης, υποθέτουμε ότι οι όροι είναι μεμονωμένοι χαρακτήρες. Κάθε κλήση της αναδρομικής συνάρτησης δημιουργεί ένα νέο κόμβο του οποίου το γλωσσικό σημείο είναι ο επόμενος χαρακτήρας της εισόδου . Αν το γλωσσικό σημείο είναι όρος, επιστρέφουμε το νέο κόμβο· αν είναι τελεστής, ορίζουμε τον αριστερό και το δεξιό δείκτη έτσι ώστε να δείχνουν στο δένδρο που κατασκευάζεται (αναδρομικώς) για τα δύο ορίσματα.

```
char *a; int i;
struct node
  { Item item; node *l, *r;
    node(Item x)
      { item = x; l = 0; r = 0; }
  };
typedef node* link;
link parse()
  { char t = a[i++]; link x = new node(t);
    if ((t == '+') || (t == '*'))
      { x->l = parse(); x->r = parse(); }
    return x;
  }
```

Εξετάσαμε τα λίγα παραδείγματα αυτής της ενότητας προκειμένου να εισαγάγουμε την έννοια ότι μπορούμε να δημιουργήσουμε και να επεξεργαστούμε ρητές συνδεδεμένες δομές δένδρων με αναδρομικά προγράμματα. Για να το κάνουμε αυτό αποτελεσματικά, πρέπει να εξετάσουμε την απόδοση πολλών αλγορίθμων, εναλλακτικών αναπαραστάσεων, εναλλακτικών μη αναδρομικών μεθόδων, και πολλές άλλες λεπτομέρειες. Παρόλα αυτά, θα αναβάλουμε την αναλυτικότερη εξέταση των προγραμμάτων επεξεργασίας δένδρων μέχρι το Κεφάλαιο 12, επειδή στα Κεφάλαια 7 έως 11 χρησιμοποιούμε τα δένδρα κυρίως για περιγραφικούς σκοπούς. Θα επανέλθουμε σε ρητές υλοποιήσεις δένδρων στο Κεφάλαιο 12, επειδή αποτελούν τη βάση πολλών αλγορίθμων τους οποίους θα γνωρίσουμε στα Κεφάλαια 12 έως 15.

Ασκήσεις

○ **5.85** Τροποποιήστε το Πρόγραμμα 5.18 έτσι ώστε να εξάγει ένα πρόγραμμα PostScript που να σχεδιάζει το δένδρο, με μορφή παρόμοια με αυτή που χρησιμοποιήθηκε στην Εικόνα 5.23, αλλά χωρίς τα μικρά πλαίσια που αναπαριστούν τους εξωτερικούς κόμβους. Για να σχεδιάσετε τις γραμμές χρησιμοποιήστε τις συναρτήσεις `moveto` και `lineto`, και για να σχεδιάσετε τους κόμβους χρησιμοποιήστε τον οριζόμενο από το χρήστη τελεστή

```
/node { newpath moveto currentpoint 4 0 360 arc fill} def
```

Μετά από αυτόν τον ορισμό, η κλήση στη `node` θα σχεδιάζει μια μαύρη κουκκίδα στις συντεταγμένες που περιέχονται στη στοίβα (δείτε την Ενότητα 4.3)

▷ **5.86** Γράψτε ένα πρόγραμμα που να μετρά τα φύλλα ενός δυαδικού δένδρου.

▷ **5.87** Γράψτε ένα πρόγραμμα που να μετρά το πλήθος των κόμβων ενός δυαδικού δένδρου οι οποίοι έχουν ένα εξωτερικό και ένα εσωτερικό παιδί.

▷ **5.88** Γράψτε ένα αναδρομικό πρόγραμμα που να υπολογίζει το μήκος της εσωτερικής διαδρομής ενός δυαδικού δένδρου, χρησιμοποιώντας τον Ορισμό 5.6.

5.89 Προσδιορίστε το πλήθος των κλήσεων συνάρτησης που γίνονται από το πρόγραμμά σας όταν υπολογίζει το μήκος της εσωτερικής διαδρομής ενός δυαδικού δένδρου. Αποδείξτε την απάντησή σας με επαγωγή.

● **5.90** Γράψτε ένα αναδρομικό πρόγραμμα που να υπολογίζει το μήκος της εσωτερικής διαδρομής ενός δυαδικού δένδρου, σε χρόνο ανάλογο του πλήθους των κόμβων του δένδρου.

○ **5.91** Γράψτε ένα αναδρομικό πρόγραμμα που να αφαιρεί από ένα τουρνουά όλα τα φύλλα με δεδομένο κλειδί (δείτε την Άσκηση 5.59).

5.8 Διάσχιση γράφου

Ως τελευταίο παράδειγμα αναδρομικού προγράμματος σε αυτό το κεφάλαιο, θα εξετάσουμε ένα από τα σημαντικότερα αναδρομικά προγράμματα: την αναδρομική διάσχιση γράφου ή *αναζήτηση με προτεραιότητα βάθους* (depth-first search). Αυτή η μέθοδος, η οποία έχει στόχο τη συστηματική επίσκεψη σε όλους τους κόμβους ενός γράφου, αποτελεί άμεση γενίκευση των μεθόδων διάσχισης δένδρου που εξετάσαμε στην Ενότητα 5.6 και προσφέρεται ως βάση για πολλούς βασικούς αλγορίθμους επεξεργασίας γράφων (δείτε το Μέρος 7). Είναι ένας απλός αναδρομικός αλγόριθμος. Ξεκινώντας από οποιονδήποτε κόμβο v,

- Επισκεπτόμαστε τον *v*.
- Επισκεπτόμαστε (αναδρομικώς) κάθε κόμβο (που *δεν έχουμε επισκεφθεί*) ο οποίος συνδέεται με τον *v*.

Αν ο γράφος είναι συνδεδεμένος, τελικά επισκεπτόμαστε όλους τους κόμβους. Το Πρόγραμμα 5.21 αποτελεί υλοποίηση αυτής της αναδρομικής διαδικασίας.

Για παράδειγμα, ας υποθέσουμε ότι χρησιμοποιούμε την αναπαράσταση λίστας γειτνίασης (adjacency list) που είδαμε στο παράδειγμα γράφου της Εικόνας 3.15. Στην Εικόνα 5.32 φαίνονται οι αναδρομικές κλήσεις που γίνονται κατά την αναζήτηση με προτεραιότητα βάθους σε αυτόν το γράφο, ενώ η ακολουθία στα αριστερά της Εικόνας 5.33 δείχνει τον τρόπο με τον οποίο ακολουθούμε τις ακμές του γράφου. Ακολουθούμε κάθε ακμή του γράφου με μία από δύο πιθανές εκβάσεις: αν η ακμή μάς οδηγήσει σε κάποιον κόμβο τον οποίο έχουμε επισκεφθεί ήδη, την αγνοούμε· αν μας οδηγήσει σε κάποιον κόμβο τον οποίο δεν έχουμε επισκεφθεί ακόμη, την ακολουθούμε μέχρι αυτόν τον κόμβο μέσω μιας αναδρομικής κλήσης. Το σύνολο όλων των ακμών που ακολουθούμε με αυτόν τον τρόπο σχηματίζει ένα επικαλύπτον δένδρο (spanning tree) του γράφου.

Η διαφορά μεταξύ της αναζήτησης με προτεραιότητα βάθους και της γενικής διάσχισης δένδρου (δείτε το Πρόγραμμα 5.14) είναι ότι πρέπει να προσέχουμε ιδιαίτερα ώστε να μην επισκεπτόμαστε κόμβους τους οποίους έχουμε επισκεφθεί ήδη. Σε ένα δένδρο, δεν συναντάμε ποτέ τέτοιους κόμβους. Στην πραγματικότητα, αν ο γράφος είναι δένδρο, η αναζήτηση με προτεραιότητα βάθους με αρχή τη ρίζα ισοδυναμεί με την προδιατεταγμένη διάσχιση.

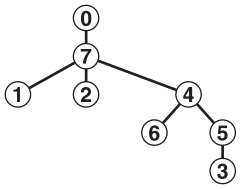
Ιδιότητα 5.10 *Η αναζήτηση με προτεραιότητα βάθους απαιτεί χρόνο ανάλογο του $V + E$ σε ένα γράφο με V κορυφές και E ακμές, όταν χρησιμοποιείται αναπαράσταση με λίστες γειτνίασης.*

Στην αναπαράσταση με λίστες γειτνίασης, υπάρχει ένας κόμβος λίστας που αντιστοιχεί σε κάθε ακμή του γράφου και ένας δείκτης κεφαλής της λίστας που αντιστοιχεί σε κάθε κορυφή του γράφου. Η αναζήτηση με προτεραιότητα βάθους περνά από όλα αυτά τα στοιχεία το πολύ μία φορά. ■

Πρόγραμμα 5.21 Αναζήτηση με προτεραιότητα βάθους

Για να επισκεφθούμε όλους τους κόμβους που συνδέονται με τον κόμβο *k* ενός γράφου, σημειώνουμε αυτόν τον κόμβο ως κόμβο *που έχουμε επισκεφθεί* και έπειτα επισκεπτόμαστε (αναδρομικώς) όλους τους κόμβους που δεν έχουμε επισκεφθεί και περιλαμβάνονται στη λίστα γειτνίασης του *k*.

```
void traverse(int k, void visit(int))
  { visit(k); visited[k] = 1;
    for (link t = adj[k]; t != 0; t = t->next)
      if (!visited[t->v]) traverse(t->v, visit);
  }
```



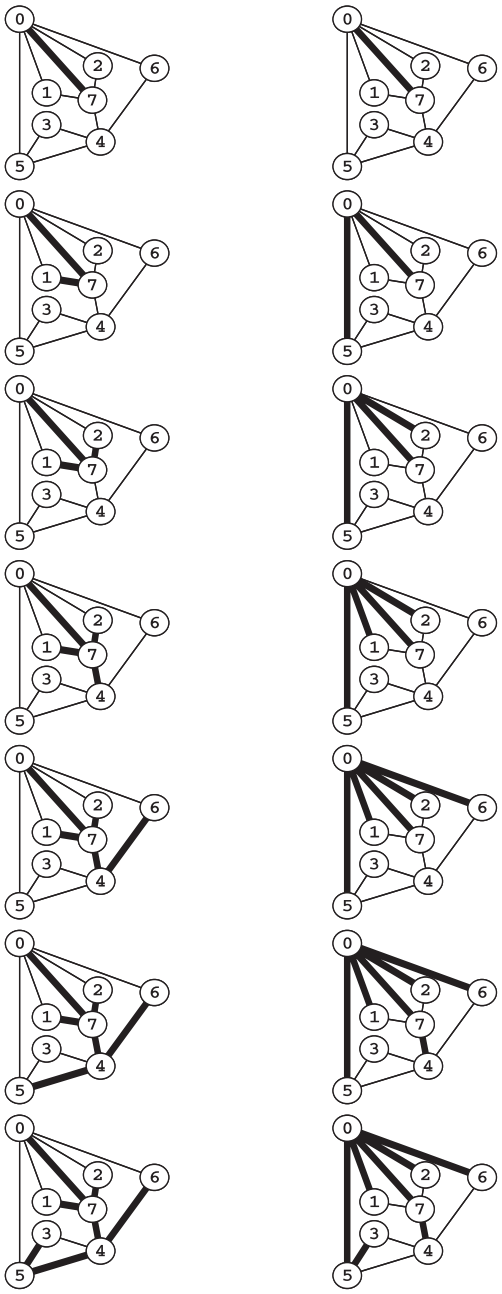

```
επίσκεψη στον 0
    επίσκεψη στον 7 (πρώτος στη λίστα του 0)
        επίσκεψη στον 1 (πρώτος στη λίστα του 7)
            έλεγχος του 7 στη λίστα του 1
            έλεγχος του 0 στη λίστα του 1
        επίσκεψη στον 2 (δεύτερος στη λίστα του 7)
            έλεγχος του 7 στη λίστα του 2
            έλεγχος του 0 στη λίστα του 2
        έλεγχος του 0 στη λίστα του 7
        επίσκεψη στον 4 (τέταρτος στη λίστα του 7)
            επίσκεψη στον 6 (πρώτος στη λίστα του 4)
                έλεγχος του 4 στη λίστα του 6
                έλεγχος του 0 στη λίστα του 6
            επίσκεψη στον 5 (δεύτερος στη λίστα του 4)
                έλεγχος του 0 στη λίστα του 5
                έλεγχος του 4 στη λίστα του 5
                επίσκεψη στον 3 (τρίτος στη λίστα του 5)
                    έλεγχος του 5 στη λίστα του 3
                    έλεγχος του 4 στη λίστα του 3
            έλεγχος του 7 στη λίστα του 4
            έλεγχος του 3 στη λίστα του 4
    έλεγχος του 5 στη λίστα του 0
    έλεγχος του 2 στη λίστα του 0
    έλεγχος του 1 στη λίστα του 0
    έλεγχος του 6 στη λίστα του 0
```

Εικόνα 5.32
Κλήσεις μεθόδου αναζήτησης με προτεραιότητα βάθους

Αυτή η ακολουθία κλήσεων μεθόδου αποτελεί την αναζήτηση με προτεραιότητα βάθους για το παράδειγμα του γράφου της Εικόνας 3.15. Το δένδρο που απεικονίζει την αναδρομική δομή (επάνω) ονομάζεται δένδρο αναζήτησης με προτεραιότητα βάθους.

Επειδή απαιτείται επίσης χρόνος ανάλογος του $V + E$ για να δημιουργήσουμε την αναπαράσταση με λίστες γειτνίασης από κάποια ακολουθία εισόδου ακμών (δείτε το Πρόγραμμα 3.19), η αναζήτηση με προτεραιότητα βάθους μάς δίνει μια λύση γραμμικού χρόνου για το πρόβλημα συνδετικότητας του Κεφαλαίου 1. Όταν, όμως, έχουμε να κάνουμε με γράφους τεραστίων διαστάσεων, οι λύσεις ένωσης-εύρεσης ενδέχεται να παραμένουν προτιμότερες, επειδή η αναπαράσταση ολόκληρου του γράφου απαιτεί χώρο ανάλογο του E, ενώ οι λύσεις ένωσης-εύρεσης απαιτούν χώρο ανάλογο του V.

Όπως και με τη διάσχιση ενός δένδρου, μπορούμε να ορίσουμε μια μέθοδο διάσχισης γράφου που να χρησιμοποιεί κάποια ρητή δομή στοίβας, όπως φαίνεται στην Εικόνα 5.34. Μπορούμε να φανταστούμε μια αφηρημένη στοίβα που να διατηρεί διπλές καταχωρίσεις: έναν κόμβο και ένα δείκτη προς τη λίστα γειτνίασης αυτού του κόμβου. Αφού ανατεθούν ως αρχικές τιμές στη στοίβα ο κόμβος εκκίνησης και σε ένα δείκτη ο πρώτος κόμβος της λίστας γειτνίασης αυτού του κόμβου, ο αλγόριθμος αναζήτησης με προτεραιότητα βάθους ισοδυναμεί με την είσοδο σε ένα βρόχο, κατά τη διάρκεια του οποίου επισκεπτόμαστε τον κόμβο που βρίσκεται στην κορυφή της στοίβας (αν δεν τον έχουμε επισκεφθεί ήδη), αποθηκεύουμε τον κόμβο στον οποίο αναφέρεται ο τρέχων δείκτης προς τη λίστα γειτνίασης, ενημερώνουμε την



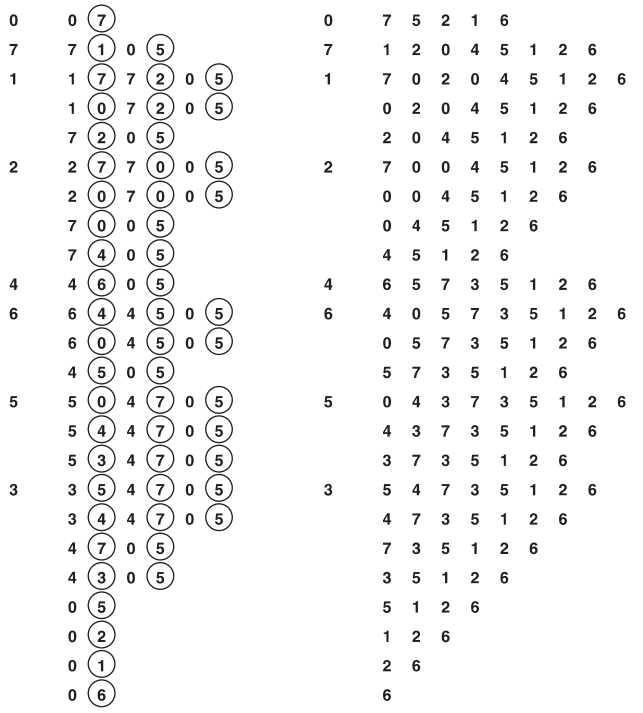
Εικόνα 5.33
Αναζήτηση με προτεραιότητα βάθους και αναζήτηση με προτεραιότητα εύρους

Κατά την αναζήτηση με προτεραιότητα βάθους (αριστερά) μετακινούμαστε από κόμβο σε κόμβο, επιστρέφοντας στον προηγούμενο κόμβο για να δοκιμάσουμε την επόμενη δυνατότητα όταν έχουμε δοκιμάσει όλες τις δυνατότητες σε κάποιο δεδομένο κόμβο. Κατά την αναζήτηση με προτεραιότητα εύρους (δεξιά) εξαντλούμε όλες τις δυνατότητες σε κάποιο κόμβο πριν μετακινηθούμε στον επόμενο.

αναφορά της λίστας γειτνίασης προς τον επόμενο κόμβο (απωθώντας από τη στοίβα την καταχώριση αν βρισκόμαστε στο τέλος της λίστας γειτνίασης), και ωθούμε στη στοίβα μια καταχώριση για τον κόμβο που αποθηκεύσαμε, με μια αναφορά στον πρώτο κόμβο της λίστας γειτνίασης αυτού του κόμβου.

Εναλλακτικά, όπως και στη διάσχιση δένδρου, μπορούμε να θεωρήσουμε ότι η στοίβα περιέχει μόνο συνδέσμους προς κόμβους. Η στοίβα δέχεται ως αρχική τιμή τον κόμβο εκκίνησης και κατόπιν μπαίνουμε σε ένα βρόχο, κατά τη διάρκεια του οποίου επισκεπτόμαστε τον κόμβο που βρίσκεται στην κορυφή της στοίβας (αν δεν τον έχουμε επισκεφθεί ήδη) και τοποθετούμε στη στοίβα όλους τους κόμβους που είναι γειτονικοί με αυτόν τον κόμβο. Στην Εικόνα 5.34 φαίνεται ότι και οι δύο αυτές μέθοδοι είναι ισοδύναμες με την αναζήτηση με προτεραιότητα βάθους για το παράδειγμα γράφου που χρησιμοποιήσαμε, και ότι αυτή η ισοδυναμία ισχύει πράγματι γενικά.

Ο αλγόριθμος επίσκεψης στον κόμβο της κορυφής και τοποθέτησης στη στοίβα όλων των γειτονικών του κόμβων είναι μια απλή μορφή της αναζήτησης με προτεραιότητα βάθους, αλλά από την Εικόνα 5.34 γίνεται σαφές ότι υποφέρει από το μειονέκτημα ότι είναι πιθανό να αφήνει στη στοίβα πολλά αντίγραφα κάθε κόμβου. Αυτό μπορεί να συμβεί ακόμη και αν ελέγξουμε μήπως έχουμε επισκεφθεί ήδη κάθε κόμβο που είμαστε έτοιμοι να τοποθετήσουμε



Εικόνα 5.34
Δυναμική στοίβας αναζήτησης με προτεραιότητα βάθους

Μπορούμε να φανταστούμε τη στοίβα ώθησης προς τα κάτω που υποστηρίζει την αναζήτηση προτεραιότητας βάθους σαν μια δομή που περιέχει έναν κόμβο και μια αναφορά προς τη λίστα γειτνίασης αυτού του κόμβου (η οποία σημειώνεται με έναν κόμβο σε κύκλο) (αριστερά). Συνεπώς, ξεκινάμε με τον κόμβο 0 της στοίβας, με αναφορά στον πρώτο κόμβο της λίστας του, τον κόμβο 7. Σε κάθε γραμμή εμφανίζεται το αποτέλεσμα της απώθησης ενός στοιχείου από τη στοίβα, της ώθησης της αναφοράς στον επόμενο κόμβο της λίστας για τους κόμβους που έχουμε επισκεφθεί ήδη, και την ώθηση μιας καταχώρισης στη στοίβα για τους κόμβους που δεν έχουμε επισκεφθεί ακόμη. Εναλλακτικά, μπορούμε να φανταστούμε τη διαδικασία σαν να ωθούμε απλώς στη στοίβα όλους τους κόμβους που είναι γειτονικοί με κάθε κόμβο που δεν έχουμε επισκεφθεί (δεξιά).

στη στοίβα, αποφεύγοντας έτσι την τοποθέτηση του κόμβου στη στοίβα αν τον έχουμε επισκεφθεί. Για να αποφύγουμε αυτό το πρόβλημα, μπορούμε να χρησιμοποιήσουμε μια υλοποίηση στοίβας που να μην επιτρέπει διπλά στοιχεία χρησιμοποιώντας μια πολιτική "παράβλεψης του παλιού στοιχείου", επειδή το αντίγραφο που βρίσκεται πλησιέστερα στην κορυφή της στοίβας είναι πάντοτε το πρώτο που επισκεπτόμαστε και, επομένως, τα υπόλοιπα απλώς απωθούνται.

Η δυναμική της στοίβας για την αναζήτηση με προτεραιότητα βάθους που παρουσιάζεται στην Εικόνα 5.34 εξαρτάται από το αν οι κόμβοι κάθε λίστας γειτνίασης καταλήγουν στη στοίβα με την ίδια σειρά που εμφανίζονται στη λίστα. Για να πάρουμε αυτή τη διάταξη για μια δεδομένη λίστα γειτνίασης όταν τοποθετούμε στη στοίβα έναν κόμβο κάθε φορά, πρέπει να τοποθετήσουμε στη στοίβα πρώτο τον τελευταίο κόμβο, στη συνέχεια τον προτελευταίο, και ούτω καθεξής. Επίσης, προκειμένου να περιορίσουμε το μέγεθος της στοίβας στο πλήθος των κορυφών, και ταυτόχρονα να επισκεπτόμαστε τους κόμβους με την ίδια σειρά όπως και στην αναζήτηση με προτεραιότητα βάθους, πρέπει να χρησιμοποιήσουμε έναν κανόνα στοίβας που να ακολουθεί την πολιτική "παράβλεψης του παλιού στοιχείου". Αν δεν μας ενδιαφέρει η επίσκεψη στους κόμβους με την ίδια σειρά όπως και στην αναζήτηση με προτεραιότητα βάθους, μπορούμε να αποφύγουμε και τις δύο αυτές συνέπειες και να σχηματίσουμε άμεσα μια μη αναδρομική μέθοδο διάσχισης γράφου που να βασίζεται σε στοίβα — η στοίβα δέχεται ως αρχική τιμή τον κόμβο εκκίνησης και μπαίνουμε σε ένα βρόχο, κατά τη διάρκεια του οποίου επισκεπτόμαστε τον κόμβο που βρίσκεται στην κορυφή της στοίβας, και στη συνέχεια προχωράμε μέσω της λίστας γειτνίασης αυτού του κόμβου, τοποθετώντας στη στοίβα κάθε κόμβο (αν δεν τον έχουμε επισκεφθεί ήδη), χρησιμοποιώντας μια υλοποίηση στοίβας που δεν επιτρέπει την ύπαρξη διπλών στοιχείων με μια πολιτική "παράβλεψης του νέου στοιχείου". Αυτός ο αλγόριθμος επισκέπτεται όλους τους κόμβους του γράφου με τρόπο παρόμοιο με αυτόν της αναζήτησης με προτεραιότητα βάθους, αλλά δεν είναι αναδρομικός.

Ο αλγόριθμος της προηγούμενης παραγράφου αξίζει την προσοχή μας επειδή θα μπορούσαμε να χρησιμοποιήσουμε οποιονδήποτε γενικευμένο ΑΤΔ ουράς χωρίς να χάσουμε τη δυνατότητα να επισκεφθούμε καθέναν από τους κόμβους του γράφου (και να δημιουργήσουμε ένα επικαλύπτον δένδρο). Για παράδειγμα, αν χρησιμοποιήσουμε ουρά αντί για στοίβα, έχουμε την περίπτωση της *αναζήτησης με προτεραιότητα εύρους* (breath-first search), η οποία είναι ανάλογη με τη διάσχιση σειράς επιπέδου ενός δένδρου. Το Πρόγραμμα 5.22 αποτελεί μια υλοποίηση αυτής της μεθόδου (με την προϋπόθεση ότι χρησιμοποιούμε υλοποίηση ουράς όπως στο Πρόγραμμα 4.16). Ένα παράδειγμα του αλγορίθμου σε λειτουργία μπορείτε να δείτε στην Εικόνα 5.35. Στο Μέρος 5 θα εξετάσουμε πολλούς αλγορίθμους γράφων που βασίζονται σε πιο εκλεπτυσμένους γενικευμένους αφηρημένους τύπους δεδομένων ουρών.

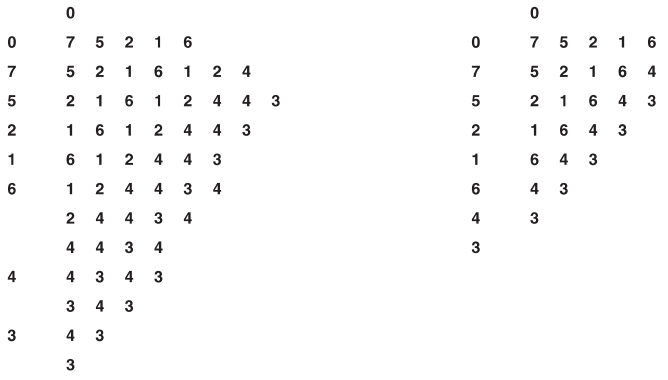
Η αναζήτηση με προτεραιότητα βάθους και η αναζήτηση με προτεραιότητα εύρους επισκέπτονται και οι δύο όλους τους κόμβους ενός γράφου, αλλά με εντελώς διαφορετικό τρόπο, όπως φαίνεται στην Εικόνα 5.36. Η αναζήτηση με προτεραιότητα εύρους ισοδυναμεί με ένα στρατό από ανιχνευτές που διασκορπίζονται προκειμένου να καλύψουν μια δεδομένη περιοχή· η αναζήτηση με προτεραιότητα βάθους αντιστοιχεί σε έναν και μοναδικό ανιχνευτή

που εξετάζει την άγνωστη περιοχή όσο πιο βαθιά μπορεί, οπισθοχωρώντας μόνο όταν καταλήγει σε αδιέξοδο. Πρόκειται για βασικά υποδείγματα επίλυσης προβλημάτων, τα οποία είναι σημαντικά για πολλούς τομείς της επιστήμης των υπολογιστών, και όχι μόνο για την αναζήτηση σε γράφους.

Πρόγραμμα 5.22 Αναζήτηση με προτεραιότητα εύρους

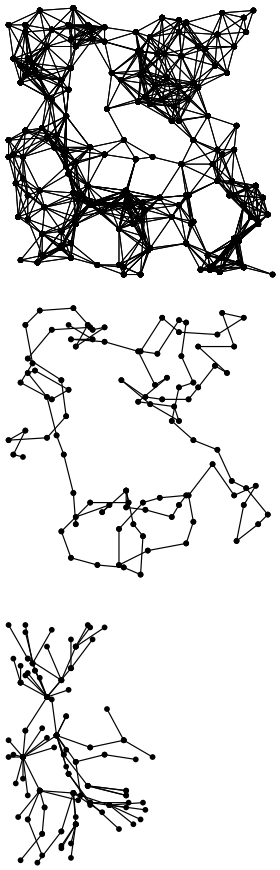
Για να επισκεφθούμε όλους τους κόμβους που συνδέονται με τον κόμβο *k* ενός γράφου, τοποθετούμε τον *k* σε μια ουρά FIFO και στη συνέχεια μπαίνουμε σε ένα βρόχο, κατά τη διάρκεια του οποίου παίρνουμε τον επόμενο κόμβο από την ουρά και, αν δεν τον έχουμε επισκεφθεί ήδη, τον επισκεπτόμαστε και ωθούμε στη στοίβα όλους τους κόμβους που ανήκουν στη λίστα γειτνίασης αυτού του κόμβου, συνεχίζοντας αυτή τη διαδικασία μέχρι να αδειάσει η ουρά.

```
void traverse(int k, void visit(int))
  {
    QUEUE<int> q(V*V);
    q.put(k);
    while (!q.empty())
      if (visited[k = q.get()] == 0)
        {
          visit(k); visited[k] = 1;
          for (link t = adj[k]; t != 0; t = t->next)
            if (visited[t->v] == 0) q.put(t->v);
        }
  }
```



Εικόνα 5.35
Δυναμική ουράς αναζήτησης με προτεραιότητα εύρους

Ξεκινάμε με το 0 *της ουράς και έπειτα παίρνουμε το* 0*, το επισκεπτόμαστε, και τοποθετούμε στην ουρά τους κόμβους που περιέχονται στη λίστα γειτνίασής του,* 7 5 2 1 6*, με την ίδια σειρά. Στη συνέχεια, παίρνουμε το* 7*, το επισκεπτόμαστε, και τοποθετούμε στην ουρά τους κόμβους που περιέχονται στη δική του λίστα γειτνίασης, και ούτω καθεξής. Με τον κανόνα απαγόρευσης διπλών στοιχείων και την πολιτική "παράβλεψης του νέου στοιχείου" (δεξιά), παίρνουμε το ίδιο αποτέλεσμα χωρίς περιττές καταχωρίσεις στην ουρά.*



Εικόνα 5.36
Δένδρα διάσχισης γράφου

Σε αυτό το διάγραμμα παρουσιάζεται η αναζήτηση με προτεραιότητα βάθους (κέντρο) και η αναζήτηση με προτεραιότητα εύρους (κάτω), ενώ βρίσκονται στη μέση της διαδικασίας αναζήτησης σε ένα μεγάλο γράφο (επάνω). Η αναζήτηση με προτεραιότητα βάθους ελίσσεται από τον έναν κόμβο στον άλλο, και έτσι οι περισσότεροι κόμβοι συνδέονται μόνο με άλλους δύο. Αντίθετα, η αναζήτηση με προτεραιότητα εύρους σαρώνει το γράφο, επισκεπτόμενη όλους τους κόμβους που συνδέονται με κάποιο δεδομένο κόμβο πριν μετακινηθεί στον επόμενο, και έτσι πολλοί κόμβοι συνδέονται με πολλούς άλλους κόμβους.

Ασκήσεις

5.92 Δείξτε τον τρόπο με τον οποίο η αναδρομική αναζήτηση με προτεραιότητα βάθους επισκέπτεται τους κόμβους του γράφου ο οποίος κατασκευάζεται από την ακολουθία ακμών `0-2`, `1-4`, `2-5`, `3-6`, `0-4`, `6-0`, και `1-3` (δείτε την Άσκηση 3.70), σχεδιάζοντας διαγράμματα αντίστοιχα με αυτά των Εικόνων 5.33 (αριστερά) και 5.34 (δεξιά).

5.93 Δείξτε τον τρόπο με τον οποίο η αναζήτηση με προτεραιότητα βάθους που βασίζεται σε στοίβα επισκέπτεται τους κόμβους του γράφο ο οποίος κατασκευάζεται από την ακολουθία ακμών `0-2`, `1-4`, `2-5`, `3-6`, `0-4`, `6-0`, και `1-3` (δείτε την Άσκηση 3.70), σχεδιάζοντας διαγράμματα αντίστοιχα με αυτά των Εικόνων 5.33 (αριστερά) και 5.34 (δεξιά).

5.94 Δείξτε τον τρόπο με τον οποίο η αναζήτηση με προτεραιότητα εύρους που βασίζεται σε ουρά επισκέπτεται τους κόμβους του γράφο ο οποίος κατασκευάζεται από την ακολουθία ακμών `0-2`, `1-4`, `2-5`, `3-6`, `0-4`, `6-0`, και `1-3` (δείτε την Άσκηση 3.70), σχεδιάζοντας διαγράμματα αντίστοιχα με αυτά των Εικόνων 5.33 (δεξιά) και 5.35 (αριστερά).

○ **5.95** Γιατί ο χρόνος εκτέλεσης που αναφέρεται στην Ιδιότητα 5.10 είναι $V + E$ και όχι απλώς E;

5.96 Δείξτε τον τρόπο με τον οποίο η αναζήτηση με προτεραιότητα βάθους που βασίζεται σε στοίβα επισκέπτεται τους κόμβους του δείγματος γράφου ο οποίος περιγράφεται στο κείμενο (Εικόνα 3.15) όταν χρησιμοποιείται η πολιτική "παράβλεψης του παλιού στοιχείου", σχεδιάζοντας διαγράμματα αντίστοιχα με αυτά των Εικόνων 5.33 (αριστερά) και 5.35 (δεξιά).

5.97 Δείξτε τον τρόπο με τον οποίο η αναζήτηση με προτεραιότητα βάθους που βασίζεται σε στοίβα επισκέπτεται τους κόμβους του δείγματος γράφου ο οποίος περιγράφεται στο κείμενο (Εικόνα 3.15) όταν χρησιμοποιείται η πολιτική "παράβλεψης του νέου στοιχείου", σχεδιάζοντας διαγράμματα αντίστοιχα με αυτά των Εικόνων 5.33 (αριστερά) και 5.35 (δεξιά).

▷ **5.98** Υλοποιήστε μια αναζήτηση με προτεραιότητα βάθους που να βασίζεται σε στοίβα για γράφους οι οποίοι αναπαρίστανται με λίστες γειτνίασης.

○ **5.99** Υλοποιήστε μια αναδρομική αναζήτηση με προτεραιότητα βάθους για γράφους που αναπαρίστανται με λίστες γειτνίασης.

5.9 Προοπτική

Η αναδρομή βρίσκεται στον πυρήνα των πρώτων θεωρητικών μελετών για τη φύση των υπολογιστικών διαδικασιών. Οι αναδρομικές συναρτήσεις και τα αναδρομικά προγράμματα παίζουν κεντρικό ρόλο στις μαθηματικές μελέτες που επιχειρούν το διαχωρισμό των προβλημάτων σε εκείνα που είναι δυνατό να λυθούν από κάποιον υπολογιστή και σε εκείνα που δεν μπορούν.

Είναι σίγουρα αδύνατο να κρίνουμε θέματα πολυσύνθετα όπως αυτά των δένδρων και της αναδρομής με τόσο συνοπτικές περιγραφές και σύντομη ανάλυση. Πολλά από τα καλύτερα παραδείγματα αναδρομικών προγραμμάτων θα βρεθούν στο επίκεντρό μας σε ολόκληρο το βιβλίο — οι αλγόριθμοι "διαίρει και βασίλευε" και οι αναδρομικές δομές δεδομένων που έχουν εφαρμοστεί με επιτυχία για την επίλυση μιας μεγάλης γκάμας προβλημάτων μεγάλου ενδιαφέροντος. Για πολλές εφαρμογές δεν υπάρχει λόγος να προχωρήσουμε πιο πέρα από μια απλή άμεση αναδρομική υλοποίηση· για κάποιες άλλες, όμως, θα εξετάσουμε και εναλλακτικές συνθετικές μη αναδρομικές υλοποιήσεις.

Σε αυτό το βιβλίο, το ενδιαφέρον μας εστιάζεται στις πρακτικές πλευρές των αναδρομικών προγραμμάτων και δομών δεδομένων. Στόχος μας είναι να αξιοποιήσουμε την αναδρομή ώστε να δίνουμε κομψές και αποδοτικές υλοποιήσεις. Για να επιτύχουμε αυτόν το στόχο, πρέπει να σεβόμαστε ιδιαίτερα τους κινδύνους που κρύβουν τα απλά προγράμματα, οι οποίοι οδηγούν μερικές φορές σε εκθετικά πλήθη κλήσεων συνάρτησης ή απαγορευτικά βαθιά ένθεση. Πάντως, παρά το γεγονός ότι υπάρχει αυτή η παγίδα, τα αναδρομικά προγράμματα και οι δομές δεδομένων είναι ιδιαίτερα ελκυστικά εργαλεία επειδή συχνά μας παρέχουν επαγωγικά επιχειρήματα τα οποία μπορούν να μας πείσουν ότι τα προγράμματά μας είναι σωστά και αποδοτικά.

Χρησιμοποιούμε δένδρα σε πολλά σημεία του βιβλίου, τόσο για να μας βοηθήσουν να κατανοήσουμε τις δυναμικές ιδιότητες των προγραμμάτων όσο και ως δυναμικές δομές δεδομένων. Ιδιαίτερα στα Κεφάλαια 12 έως 15 θα επικεντρωθούμε σε μεγάλο βαθμό στο χειρισμό ρητών δομών δένδρου. Οι ιδιότητες που περιγράψαμε σε αυτό το κεφάλαιο παρέχουν τις βασικές πληροφορίες που χρειαζόμαστε αν θέλουμε να χρησιμοποιούμε ρητές δομές δένδρων με αποτελεσματικό τρόπο.

Παρά τον κεντρικό ρόλο της στο σχεδιασμό αλγορίθμων, η αναδρομή δεν αποτελεί πανάκεια. Όπως διαπιστώσαμε στις μελέτες μας σχετικά με τους αλγορίθμους διάσχισης δένδρων και γράφων, οι αλγόριθμοι που βασίζονται σε στοίβα (εγγενώς αναδρομικοί) δεν αποτελούν τη μοναδική επιλογή όταν έχουμε να χειριστούμε πολλές υπολογιστικές εργασίες. Μια αποτελεσματική τεχνική σχεδίασης αλγορίθμων για πολλά προβλήματα είναι να χρησιμοποιήσουμε γενικευμένες υλοποιήσεις ουρών αντί για στοίβες, οι οποίες μας δίνουν την ελευθερία να επιλέγουμε την επόμενη εργασία με βάση υποκειμενικότερα κριτήρια αντί απλώς να επιλέγουμε την πιο πρόσφατη. Οι δομές δεδομένων και οι αλγόριθμοι που υποστηρίζουν αποδοτικά τέτοιες λειτουργίες αποτελούν το κύριο θέμα του Κεφαλαίου 9, και θα συναντήσουμε πολλά παραδείγματα εφαρμογών τους κατά την εξέταση των αλγορίθμων γράφων στο Μέρος 5.