11

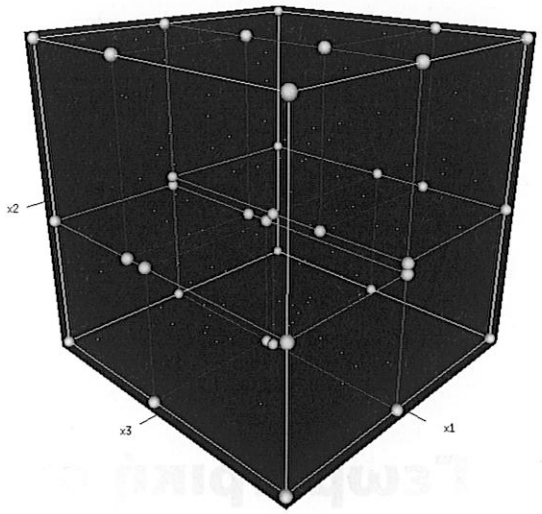
Γεωμετρική αναζήτηση

Αυτό το κεφάλαιο ασχολείται με αποδοτικές δομές γεωμετρικών δεδομένων και τους αντίστοιχους αλγορίθμους αναζήτησης και ειδικά ορθογώνιας αναζήτησης, με έμφαση στους αλγορίθμους που είναι ευαίσθητοι εξόδου. Επίσης, προσέχουμε ιδιαίτερα την κατανάλωση μνήμης, διότι οι δομές αυτές χρησιμοποιούνται με μεγάλο όγκο δεδομένων, π.χ. σε αστρονομικές, γεωγραφικές, ή βιολογικές εφαρμογές, όπου ακόμη και ένας λογαριθμικός παράγων είναι σημαντικός.

Εξετάζουμε αρχικά δομές δεδομένων για σημεία σε ευθεία ώστε να επισημάνουμε τα διαφορετικά προβλήματα αναζήτησης σε ένα απλό πλαίσιο και, ακολούθως, αντιμετωπίζουμε σημειοσύνολα στο επίπεδο και σε μεγαλύτερη διάσταση. Δίνουμε έμφαση στα *kd*–δένδρα και στα δένδρα περιοχής (range trees)· δείτε, για παράδειγμα, στο σχήμα 11.1 τη χρήση ενός *kd*–δένδρου για την αποθήκευση τριδιάστατου σημειοσυνόλου. Περιλαμβάνουμε στη συζήτηση την τεχνική της κλασματικής επαλληλίας (fractional cascading) για τη βελτίωση της πολυπλοκότητας, η οποία εφαρμόζεται και σε άλλες περιπτώσεις όπου επαναλαμβάνονται πολλά παρόμοια ερωτήματα. Το κεφάλαιο κλείνει με μια αναφορά στα δένδρα προτεραιότητας (priority trees).

Ο κυριότερος στόχος των δομών δεδομένων, περιλαμβανομένων των γεωμετρικών δομών, είναι η γρήγορη αναζήτηση. Στις τεχνικές αναζήτησης ανάμεσα σε γεωμετρικά αντικείμενα, κεντρική θέση κατέχει η ορθογώνια αναζήτηση (orthogonal search), δηλαδή η εύρεση όλων των σημείων ενός αρχικού συνόλου που βρίσκονται σε δεδομένο ορθογώνιο παραλληλόγραμμο ή παραλληλεπίπεδο.

Για τη συνέχεια, υιοθετούμε μια απλή υπόθεση γενικής θέσης των σημείων: δύο διαφορετικά σημεία διαφέρουν σε τουλάχιστον μία συντεταγμένη. Για να αναιρέσουμε την υπόθεση αυτή, μπορούμε να χρησιμοποιήσουμε μια απειροελάχιστη συμβολική διαταραχή, όπως αυτές της ενότητας 6.3. Εναλλακτικά, και με βάση την άσκηση 11.10,



Σχήμα 11.1: Τριδιάστατο σύνολο σημείων αποθηκευμένο σε *kd*–δένδρο.

όλες οι δομές του κεφαλαίου καλύπτουν την περίπτωση που τα σημεία δεν έχουν διαφορετικές συντεταγμένες.

Το παρόν κεφάλαιο αποτελεί μια εισαγωγή στον πλούσιο χώρο των γεωμετρικών δομών δεδομένων και της ορθογώνιας αναζήτησης. Δεν αναλύονται δομές εξωτερικής μνήμης, όπως τα *B*–δένδρα και τα *R*–δένδρα, που προέρχονται από τις περιοχές των βάσεων δεδομένων και της εξόρυξης δεδομένων. Για περισσότερες πληροφορίες βλ. [Aga97, BY98, dBvKOS97, PS85].

11.1 Μονοδιάστατα δεδομένα

Διακρίνουμε ορισμένες βασικές δομές για την ευθεία, οι οποίες θα γενικευτούν στη συνέχεια. Τα δεδομένα είναι, όπως και στα προηγούμενα κεφάλαια, πραγματικοί αριθμοί. Οι περιγραφές των δομών δεδομένων υπάρχουν στην ενότητα 1.4, αλλά εδώ καλύπτουμε και το ζήτημα της ορθογώνιας αναζήτησης με σκοπό τη γενίκευσή του σε δύο και περισσότερες διαστάσεις.

Σε μία διάσταση, το ερώτημα (query) ορθογώνιας περιοχής ανάγεται σε ένα ερώτημα διαστήματος $[a, a']$, για πραγματικούς αριθμούς a, a'. Ζητούνται δηλαδή όλα τα σημεία στη δομή, τα οποία βρίσκονται στο δεδομένο διάστημα. Ειδική περίπτωση αποτελεί το ερώτημα διαστήματος $(-\infty, a']$ ή, αντίστοιχα, του $[a, \infty)$, για πραγματικό αριθμό a.

Μία απλούστατη δομή είναι ο *ταξινομημένος πίνακας* (sorted array), ο οποίος υλοποιεί την αφηρημένη δομή του ευρετηρίου (βλ. ενότητα 1.4). Καταλαμβάνει μνήμη $O(n)$, όπου n είναι το πλήθος των στοιχείων στη δομή, κατασκευάζεται σε χρόνο $O(n \log n)$,

Δομή	Μνήμη	Κατασκευή	Ανανέωση	$[a, a']$	$(-\infty, a']$
ταξινομημένος πίνακας	n	$n \log n$	n	$k + \log n$	k
ΔΔΑ	n	$n \log n$	$\log n$	$k + \log n$	k
ουρά προτεραιότητας	n	n	$\log n$	n	k

Πίνακας 11.1: Συγκεντρωτικός πίνακας ασυμπτωτικής πολυπλοκότητας διαφόρων πράξεων, για δομές δεδομένων στην ευθεία.

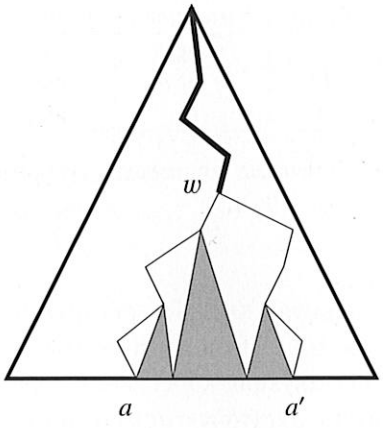
αλλά κάθε ανανέωση απαιτεί χρόνο $O(n)$. Το ερώτημα ύπαρξης συγκεκριμένου στοιχείου, καθώς και ο εντοπισμός του, απαντώνται σε λογαριθμικό χρόνο, ενώ η εύρεση του ελάχιστου ή του μέγιστου στοιχείου, καθώς και η εύρεση του προηγούμενου ή του επόμενου από δεδομένο στοιχείο, απαιτούν σταθερό χρόνο. Το ερώτημα $[a, a']$ απαντάται σε χρόνο $O(k + \log n)$, ενώ το ερώτημα $(-\infty, a']$ σε χρόνο $O(k)$, όπου k το μέγεθος της εξόδου.

Μια αποδοτικότερη δομή είναι το ισοζυγισμένο (ισορροπημένο) *δυαδικό δένδρο αναζήτησης* (balanced binary search tree) (ΔΔΑ), το οποίο μελετήθηκε στην ενότητα 1.4. Γνωστές υλοποιήσεις του περιλαμβάνουν τα κόκκινα–μαύρα δένδρα, τα δένδρα AVL και τα δένδρα 2–3 ή 2–3–4. Σε μία εκδοχή του, το ΔΔΑ αποθηκεύει τα δεδομένα σημεία στα φύλλα του ενώ οι εσωτερικοί κόμβοι περιέχουν το μεγαλύτερο στοιχείο στο αριστερό υποδένδρο του κόμβου. Δηλαδή κάθε αριστερό υποδένδρο περιέχει στοιχεία μικρότερα ή ίσα με αυτό του κόμβου, ενώ το δεξιό υποδένδρο περιέχει στοιχεία μεγαλύτερα· αυτό το σχήμα καλείται και εσωτερική ταξινόμηση (inorder). Σημειωτέον πως κάθε φύλλο, εκτός του άκρα δεξιά που περιέχει το μέγιστο στοιχείο, αντιστοιχεί σε μοναδικό κόμβο με το ίδιο στοιχείο.

Το ισοζυγισμένο ΔΔΑ καταλαμβάνει μνήμη $O(n)$ για n στοιχεία, κατασκευάζεται σε χρόνο $O(n \log n)$ και κάθε εντοπισμός στοιχείου ή ανανέωση (διαγραφή ή εισαγωγή) απαιτεί χρόνο $O(\log n)$. Για ταξινομημένα δεδομένα, η κατασκευή γίνεται σε γραμμικό χρόνο, αρχίζοντας από τα φύλλα (συνθετική κατασκευή, bottom up).

Το ερώτημα $[a, a']$ απαντάται σε χρόνο $O(k + \log n)$, όπου k το πλήθος των σημείων στο διάστημα $[a, a']$. Το δένδρο διασχίζεται όπως στον εντοπισμό του a και a' με μία μοναδική διαδρομή μέχρι κάποιο κόμβο w· στη συνέχεια, δύο διαφορετικές διαδρομές συνεχίζουν έως τα αντίστοιχα φύλλα. Μετά τον κόμβο w, για κάθε κόμβο στη διαδρομή προς το a (αντίστοιχα προς το a') όπου κινούμαστε προς τα αριστερά (αντίστοιχα, δεξιά), όλα τα στοιχεία του δεξιού (αντίστοιχα, αριστερού) υποδένδρου τυπώνονται στην έξοδο. Τα σύνολα των στοιχείων σε αυτά τα υποδένδρα διαμερίζουν το σύνολο των σημείων στο $[a, a']$· δείτε το σχ. 11.2. Τέλος, το ερώτημα $(-\infty, a']$ απαντάται σε χρόνο $O(k + \log n)$, ένα φράγμα που γίνεται $O(k)$ εφόσον τα φύλλα συνδέονται μεταξύ τους.

Η δομή αυτή μπορεί να χρησιμοποιηθεί και για d–διάστατα δεδομένα, με χρόνο αναζήτησης στοιχείου $O(d \log n)$. Όμως, σε γενική διάσταση η ορθογώνια αναζήτηση γίνεται προβληματική: θα δούμε παρακάτω μια ενδιαφέρουσα επέκταση στα δένδρα περιοχής, τα οποία είναι πιο κατάλληλα στο d–διάστατο χώρο.



Σχήμα 11.2: Αναζήτηση διαστήματος $[a, a']$ σε ΔΔΑ και ο κόμβος w όπου διαχωρίζονται οι δύο διαδρομές προς τα a, a'.

Η *ουρά προτεραιότητας* (priority queue) (η αντίστοιχη υλοποίηση λέγεται και σωρός, heap) με το ελάχιστο στοιχείο στη ρίζα, καταλαμβάνει $O(n)$ μνήμη. Η ουρά προτεραιότητας κατασκευάζεται σε χρόνο $O(n)$ αρχίζοντας από τα φύλλα, ενώ κάθε ανανέωση απαιτεί χρόνο $O(\log n)$. Το ερώτημα $[a, a']$ απαντάται σε χρόνο $O(n)$, που είναι απαγορευτικό, ενώ το ερώτημα $(-\infty, a']$ σε χρόνο $O(k)$ μόνο. Η δομή αυτή θα γενικευτεί σε δύο διαστάσεις με τα *kd*–δένδρα, χρησιμοποιώντας για την τετμημένη την προτεραιότητα και για την τεταγμένη την αποθήκευση αριστερά ή δεξιά (με ισοζυγισμένο τρόπο).

Οι πίνακες κατακερματισμού (hash table) (βλ. ενότητα 1.4) δεν εξετάζονται εδώ διότι δεν επιτρέπουν γρήγορη ορθογώνια αναζήτηση, καθώς η εύρεση ενός στοιχείου δεν βοηθά στην εύρεση του επόμενου ή του προηγούμενού του.

11.2 *kd*–δένδρα

Περνάμε τώρα σε δομές δεδομένων στο επίπεδο. Τα στοιχεία που αποθηκεύονται είναι σημεία με δύο πραγματικές συντεταγμένες. Ας υποθέσουμε πως κάθε σημείο έχει διαφορετική τετμημένη και διαφορετική τεταγμένη.

Μία αποτελεσματική δομή για προβλήματα με μεγάλη κατανάλωση μνήμης είναι τα *kd–δένδρα* [Ben75]. Το βασικό τους πλεονέκτημα είναι πως καταλαμβάνουν γραμμικό χώρο ως προς το μέγεθος του συνόλου που αποθηκεύουν. Πρόκειται, γενικά, για μη ισοζυγισμένα ΔΔΑ, τα οποία μπορούμε να τα ισοζυγίσουμε εφόσον γνωρίζουμε εκ των προτέρων το σύνολο των στοιχείων τους. Παρακάτω υποθέτουμε πως όντως γνωρίζουμε όλα τα στοιχεία.

Στηριζόμαστε στο γεγονός πως οι κόμβοι άρτιου βάθους, ξεκινώντας με τη ρίζα που έχει βάθος 0, διαχωρίζουν τα σημεία σε δύο υποσύνολα, ως προς την *τετμημένη* τους, τα οποία είναι περίπου ισάριθμα, δηλαδή διαφέρουν το πολύ κατά ένα στοιχείο. Κάθε

υποσύνολο αποθηκεύεται σε ένα υποδένδρο του κόμβου, ενώ ο ίδιος ο κόμβος διατηρεί την κεντρική τετμημένη. Το στοιχείο του κόμβου θεωρείται πως ανήκει στο αριστερό υποδένδρο.

Οι κόμβοι περιττού βάθους διαιρούν το σύνολο των σημείων που αποθηκεύονται στο υποδένδρο τους σε δύο σχεδόν ισάριθμα σύνολα, χρησιμοποιώντας την κεντρική *τεταγμένη* του συνόλου. Το αριστερό υποδένδρο αποθηκεύει τις μικρότερες τεταγμένες, δηλ. τα σημεία που βρίσκονται στο κάτω ημιεπίπεδο. Κάθε φύλλο περιέχει ένα σημείο του δεδομένου συνόλου. Τελικά, ένα *kd*–δένδρο καθορίζει μια υποδιαίρεση του επιπέδου· βλ. σχήμα 11.3. Για παράδειγμα, το σημείο $(52, 23)$ υποδιαιρεί το σύνολο σε δύο υποσύνολα των τριών σημείων.

Η κατασκευή ενός *kd*–δένδρου ξεκινά με την ταξινόμηση όλων των σημείων ως προς την τετμημένη και την τεταγμένη τους, σε $O(n \log n)$. Κάθε κόμβος, αποθηκεύει το αντίστοιχο υποσύνολο των σημείων ταξινομημένο, χρησιμοποιώντας την ταξινόμηση στον κόμβο–γονέα. Ο χρόνος υπολογισμού του αντίστοιχου υποσυνόλου είναι γραμμικός, συνεπώς το συνολικό κόστος υπολογίζεται ακριβώς όπως στην παράσταση (11.1).

Λήμμα 11.1 *Το kd–δένδρο n σημείων στο επίπεδο καταλαμβάνει συνολικό χώρο $O(n)$, ενώ η κατασκευή του ολοκληρώνεται σε χρόνο $O(n \log n)$.*

Απόδειξη. Η χωρική πολυπλοκότητα είναι προφανής. Η εύρεση του μέσου στοιχείου απαιτεί γραμμικό χρόνο, άρα όλοι οι αντίστοιχοι υπολογισμοί κοστίζουν $T(n)$, όπου

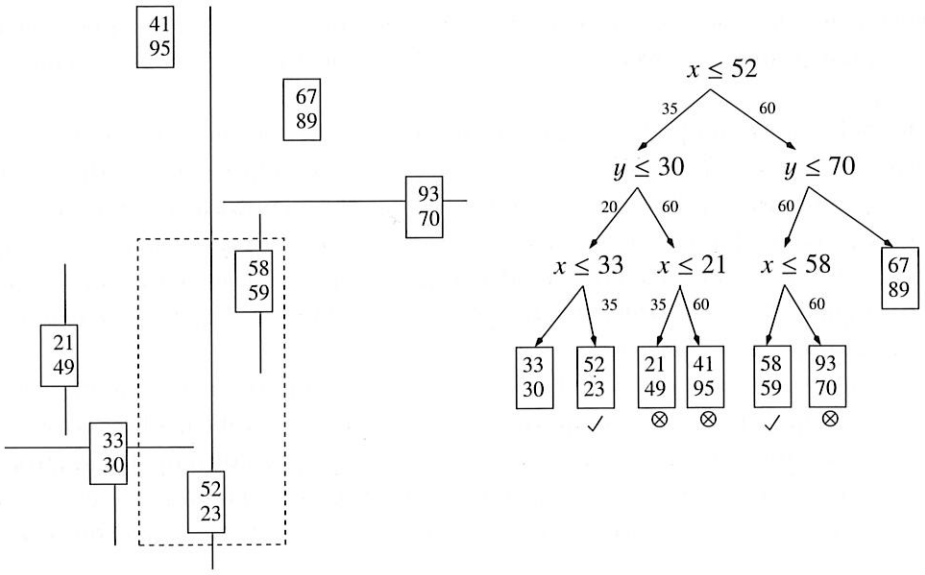
$$T(n) = O(n) + 2T(n/2) \Rightarrow T(n) = O(n \log n). \tag{11.1}$$

Αυτό αποδεικνύει το φράγμα στην πολυπλοκότητα κατασκευής. □

Για ερωτήματα που αντιστοιχούν σε ορθογώνιες περιοχές, ο αλγόριθμος διασχίζει τους κόμβους των οποίων τα αντίστοιχα σημεία ανήκουν στο ορθογώνιο του ερωτήματος. Αν τα σημεία ενός υποδένδρου ανήκουν εξ ολοκλήρου στο ορθογώνιο, το υποδένδρο διασχίζεται και όλα τα σημεία τυπώνονται. Αν τα σημεία του υποδένδρου δεν ανήκουν καθόλου στο ορθογώνιο, ο αλγόριθμος δεν εισέρχεται στο υποδένδρο. Αρκεί λοιπόν σε κάθε κόμβο να συγκρίνεται το ορθογώνιο του ερωτήματος με την ορθογώνια (φραγμένη ή μη) περιοχή στην οποία αντιστοιχεί ο κόμβος.

Λήμμα 11.2 *Θεωρούμε ένα ερώτημα που αντιστοιχεί σε ορθογώνιο με ακμές παράλληλες προς τους άξονες. Τα k από τα n αποθηκευμένα σημεία του επιπέδου, τα οποία ανήκουν στο ορθογώνιο, εντοπίζονται σε χρόνο $O(\sqrt{n} + k)$.*

Απόδειξη. Για τα υποδένδρα που ανήκουν εξ ολοκλήρου στο ορθογώνιο το κόστος είναι γραμμικό και συνεισφέρει στην ποσότητα $O(k)$. Μένει να φράξουμε το ασυμπτωτικό κόστος διάσχισης κόμβων, των οποίων το υποδένδρο περιέχει ορισμένα μόνο σημεία που ανήκουν στο ορθογώνιο του ερωτήματος. Πρέπει λοιπόν να φράξουμε το πλήθος των κόμβων των οποίων η περιοχή τέμνεται από μία ακμή του δεδομένου ορθογωνίου.



Σχήμα 11.3: Αριστερά: υποδιαίρεση του επιπέδου από 7 σημεία και το ερώτημα $[35, 60] \times [20, 60]$. Δεξιά: το *kd*–δένδρο και οι έλεγχοι για την απάντηση του ερωτήματος.

Για την ασυμπτωτική πολυπλοκότητα, αρκεί να μελετήσουμε μόνο κατακόρυφες ακμές ή, για λόγους απλότητας, κατακόρυφες ευθείες.

Για κάποιο κόμβο σε άρτιο βάθος, ο οποίος αποθηκεύει s σημεία, έστω $q(s)$ το πλήθος των κόμβων των οποίων η περιοχή τέμνεται από μία δεδομένη κατακόρυφη ευθεία. Ο κόμβος συνεισφέρει μία μονάδα στο $q(s)$ ανν η περιοχή του τέμνεται, αλλιώς δεν συνεισφέρει. Στην περίπτωση που η περιοχή τέμνεται, η κατακόρυφη τέμνει την περιοχή είτε του αριστερού είτε του δεξιού υποδένδρου του κόμβου, αλλά όχι και τις δύο. Επίσης τέμνει το πολύ δύο εγγόνια, τα οποία βρίσκονται ξανά σε άρτιο βάθος, επομένως καταλήγουμε σε αναδρομικό ορισμό της συνάρτησης $q(s)$ ως εξής:

$$q(n) \leq 2 + 2q(n/4) \leq 2 + 2^2 + 2^2 q(n/4^2) = 2 + 2^2 + \cdots + 2^t q(n/4^t) \Rightarrow q(n) = O(\sqrt{n}),$$

όπου $t = \lfloor \log_4 n \rfloor$ και $q(n/4^t) = O(1)$. □

Τα *kd*–δένδρα επεκτείνονται σε σημεία του $\mathbb{R}^d$ για $d > 2$. Σε βάθος $0, 1, 2, \ldots, d-1$ οι κόμβοι υποδιαιρούν το σύνολο ως προς την πρώτη, δεύτερη, τρίτη κ.ο.κ συντεταγμένη. Ο χώρος αποθήκευσης παραμένει γραμμικός, η κατασκευή απαιτεί χρόνο $O(n \log n)$ αν θεωρήσουμε $d = O(1)$, ενώ το ορθογώνιο ερώτημα απαντάται σε χρόνο $O(n^{1-1/d} + k)$, όπως αποδεικνύεται στην άσκηση 11.4.

Υπάρχει μια άλλη αναζήτηση, σημαντική σε προβλήματα εξόρυξης δεδομένων. Πρόκειται για τη *μερική αντιστοίχιση* (partial correspodence), όπου δίνονται $s < d$ συντεταγμένες και πρέπει να βρεθούν τα σημεία που ικανοποιούν αυτούς τους s περιορισμούς.

Τότε το ερώτημα σε kd–δένδρο διάστασης d κοστίζει $O(k + n^{1-s/d})$ [Ben75]. Ορισμένες ενδιαφέρουσες ειδικές περιπτώσεις εξετάζονται στην άσκηση 11.5.

11.3 Δένδρα περιοχής

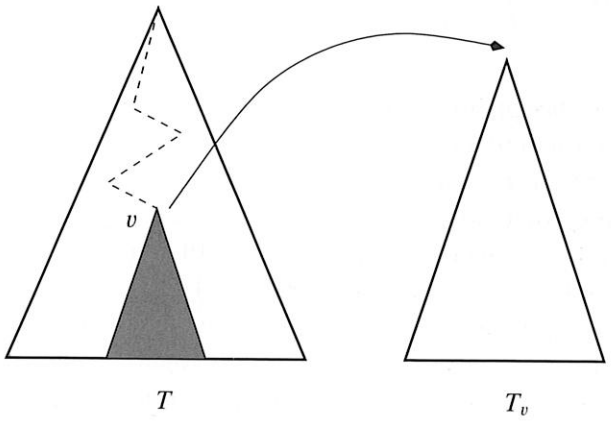
Μια ενδιαφέρουσα δομή γεωμετρικών δεδομένων είναι τα *δένδρα περιοχής* (range trees). Τα δένδρα περιοχής βασίζονται σε ένα ισοζυγισμένο ΔΔΑ, το οποίο συμβολίζεται με T και καλείται πρωτεύον (primary). Αυτό αποθηκεύει τα σημεία ως προς την τετμημένη τους, όπως αν ήταν μονοδιάστατα δεδομένα (δείτε την ενότητα 11.1). Άρα κάθε σημείο φυλάσσεται σε ένα ακριβώς φύλλο, ενώ όλα τα σημεία πλην ενός φυλάσσονται και σε έναν ακριβώς εσωτερικό κόμβο.

Τώρα όμως επιθυμούμε να αποθηκεύσουμε σημεία του επιπέδου. Κάθε κόμβος ή φύλλο v του T αντιστοιχεί σε ένα υποσύνολο σημείων, δηλαδή στα σημεία που βρίσκονται στο υποδένδρο με ρίζα το v. Αυτά αποθηκεύονται σε ένα νέο ισοζυγισμένο ΔΔΑ T_v ως προς την τεταγμένη τους, το οποίο καλείται δευτερεύον (secondary)· δείτε το σχήμα 11.4.

Λήμμα 11.3 *Τα δένδρα περιοχής καταλαμβάνουν χώρο $O(n \log n)$ για την αποθήκευση n σημείων.*

Απόδειξη. Κάθε σημείο αποθηκεύεται σε φύλλο του T, αλλά και σε όλα τα T_v για τα v που βρίσκονται στη διαδρομή του T από τη ρίζα προς το φύλλο. Άρα κάθε σημείο αποθηκεύεται $O(\log n)$ φορές. □

Μια σημαντική παρατήρηση είναι πως η ένωση όλων των δευτερευόντων δένδρων T_v, για όλους τους κόμβους v που ανήκουν σε κάποιο επίπεδο του T, περιέχει ολόκληρο το δεδομένο σημειοσύνολο.



Σχήμα 11.4: Πρωτεύον δένδρο περιοχής και δευτερεύον δένδρο για τον κόμβο v.

Λήμμα 11.4 *Η κατασκευή των δένδρων περιοχής απαιτεί χρόνο $O(n \log n)$, για ταξινομημένα ή μη δεδομένα.*

Απόδειξη. Για μη ταξινομημένα σημεία, απαιτείται πρώτα η ταξινόμησή τους ως προς x και y. Στα επόμενα στάδια θεωρούμε τα σημεία ταξινομημένα και θα δείξουμε πως η συνολική πολυπλοκότητα είναι $O(n \log n)$.

Με ταξινομημένα σημεία ως προς x, η κατασκευή του T απαιτεί γραμμικό χρόνο. Αρκεί τώρα να δείξουμε πως το ίδιο ισχύει και για τον συνολικό χρόνο κατασκευής όλων των δένδρων T_v που αντιστοιχούν στους κόμβους v ενός επιπέδου του T, διότι αυτά τα T_v περιέχουν όλα τα δεδομένα σημεία ακριβώς μία φορά το καθένα.

Υπάρχουν $O(\log n)$ επίπεδα. Άρα η συνολική πολυπλοκότητα κατασκευής των T_v φράσσεται από το $O(n \log n)$. Έμμεσα πραγματοποιήθηκε και η υποδιαίρεση του αρχικού συνόλου στα διάφορα T_v, η οποία συνολικά έχει επίσης γραμμικό κόστος. □

Πώς απαντώνται ορθογώνια ερωτήματα στα δένδρα περιοχής; Ο αλγόριθμος διασχίζει δύο διαδρομές στο T με βάση τις τετμημένες a, a' του δεδομένου ορθογωνίου, με κόστος $O(\log n)$. Έστω w ο κόμβος όπου διαχωρίζονται οι δύο διαδρομές προς τα φύλλα a, a' του T. Παρατηρήστε πως ο w είναι ο χαμηλότερος κόμβος του οποίου η τετμημένη ανήκει στο διάστημα (a, a'), εφόσον υποθέσαμε πως κάθε κόμβος αποθηκεύει μία ακριβώς τετμημένη. Από τον κόμβο w και μέχρι τα a, a', για κάθε κόμβο v σε κάθε μία από τις δύο διαδρομές, πρέπει να εξεταστεί ένα δευτερεύον δένδρο T_v στη χειρότερη περίπτωση.

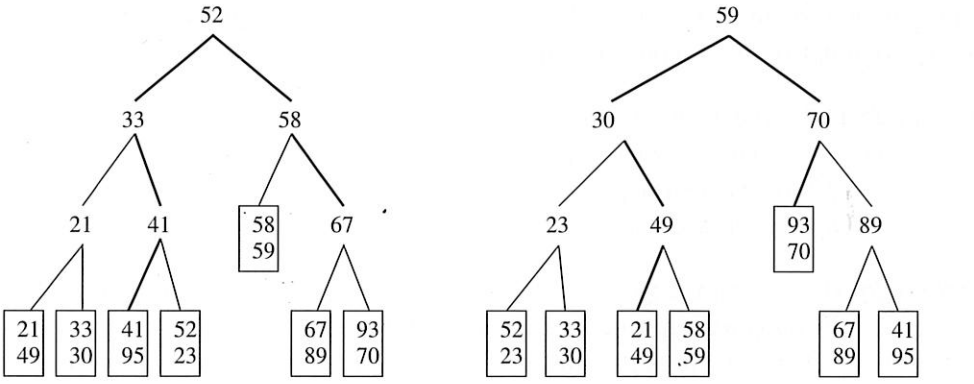
Συγκεκριμένα, εκτελείται ο αλγόριθμος 11.5 και υποθέστε πως r και l είναι, αντίστοιχα, το δεξί και αριστερό παιδί ενός κόμβου u. Παρατηρήστε πως στη διαδρομή προς a (αντίστοιχα, a') ο αλγόριθμος δεν εξετάζει ποτέ τα δευτερεύοντα δένδρα των αριστερών (αντίστοιχα, δεξιών) παιδιών. Αντίθετα, διασχίζει τα υποδένδρα στο δεξί παιδί r (αντίστοιχα, αριστερό παιδί l), εφόσον ακολουθήσει αριστερή (αντίστοιχα, δεξιά) διακλάδωση στο u. Η βασική ιδέα είναι ίδια όπως σε μονοδιάστατα δεδομένα· δείτε το σχ. 11.2.

Παράδειγμα 11.1 Ας θεωρήσουμε το παράδειγμα σημειοσυνόλου με 7 σημεία, τα οποία φαίνονται ως ζεύγη συντεταγμένων στα φύλλα του δένδρου T, ταξινομημένα ως προς την τετμημένη τους (αριστερά στο σχήμα 11.5, αλλά και στο σχήμα 11.3). Δεξιά στο σχήμα 11.5 φαίνεται το δευτερεύον δένδρο T_w, $w = 52$.

Για την αναζήτηση της περιοχής $[35, 60] \times [20, 60]$, ο αλγόριθμος διασχίζει το T με τη διαδρομή $(52, 33, 41)$, που καταλήγει στο φύλλο $(41, 95)$, και το $(52, 58, 67)$: η δεξιά διαδρομή σταματά στον κόμβο του 67 διότι βρίσκεται εκτός x–περιοχής. Ο κόμβος w, όπου διαχωρίζονται οι διαδρομές, είναι η ρίζα του T, επομένως εκεί αρχίζει η εξερεύνηση του δευτερεύοντος δένδρου που φαίνεται στο σχήμα 11.5. Στο πρώτο μονοπάτι του T, τα δεξιά υποδένδρα περιλαμβάνονται στην x–περιοχή. Αφού ανήκει στο y–διάστημα, το σημείο $(52, 23)$ τυπώνεται στην έξοδο, ενώ στο φύλλο $(41, 95)$ η τεταγμένη είναι εκτός

Αλγόριθμος 11.5 Ορθογώνια αναζήτηση σε δένδρο περιοχής.
Είσοδος: Δένδρο περιοχής και ορθογώνιο $[a, a'] \times [b, b']$.
Έξοδος: Όλα τα σημεία του δένδρου που ανήκουν στο δεδομένο ορθογώνιο.

1: Διέσχισε το πρωτεύον δένδρο T ως προς το x–διάστημα $[a, a']$· έστω πως οι δύο διαδρομές προς a, a' διαχωρίζονται στον κόμβο w.
2: Στη διαδρομή από w μέχρι το φύλλο του a, για κάθε κόμβο όπου ακολουθείται η αριστερή διακλάδωση, εξέτασε το δευτερεύον δένδρο του δεξιού παιδιού, δηλαδή το T_r. Αν πρόκειται για δεξιά διακλάδωση, δεν εξετάζεται κανένα υποδένδρο.
3: Στη διαδρομή από w μέχρι το φύλλο του a', για κάθε κόμβο όπου ακολουθείται η δεξιά διακλάδωση, εξέτασε το δευτερεύον δένδρο του αριστερού παιδιού, δηλαδή το T_l. Αν πρόκειται για αριστερή διακλάδωση, δεν εξετάζεται κανένα υποδένδρο.



Σχήμα 11.5: Αριστερά: το πρωτεύον δένδρο περιοχής T για το παράδειγμα 11.1. Ο κόμβος w, όπου οι δύο διαδρομές διαχωρίζονται, είναι η ρίζα. Δεξιά: το δευτερεύον δένδρο T_w, $w = (52, 23)$, το οποίο αποθηκεύει όλες τις τεταγμένες. Και στα δύο δένδρα, οι έντονες ακμές είναι αυτές που διασχίζει ο αλγόριθμος αναζήτησης για την περιοχή $[35, 60] \times [20, 60]$.

y–διαστήματος. Ομοίως και για τα αριστερά υποδένδρα στη δεξιά διαδρομή, δηλαδή το σημείο $(58, 59)$. ■

Λήμμα 11.6 *Θεωρούμε ερώτημα που αντιστοιχεί σε ορθογώνιο με ακμές παράλληλες προς τους άξονες. Το ερώτημα απαντάται σε χρόνο $O(\log^2 n + k)$, όπου k το μέγεθος της εξόδου και n το πλήθος των αποθηκευμένων σημείων.*

Απόδειξη. Ο αλγόριθμος ακολουθεί δύο διαδρομές στο T με βάση τις τετμημένες του δεδομένου ορθογώνιου, με κόστος $O(\log n)$. Από τον κόμβο w, όπου διαχωρίζονται οι δύο διαδρομές και μέχρι τα φύλλα του T, για κάθε κόμβο σε κάθε διαδρομή πρέπει να εξεταστεί το πολύ ένα T_v, όπως περιγράφεται στον αλγόριθμο 11.5. Αυτά τα δευτερεύο-

ντα δένδρα T_v επαρκούν για την αναζήτησή μας. Επίσης δεν περιέχουν κανένα κοινό σημείο διότι $T_u \cap T_t = \emptyset$ για οποιουσδήποτε κόμβους u, t που δεν ανήκουν στην ίδια διαδρομή από τη ρίζα προς κάποιο φύλλο. Άρα τα παραπάνω δευτερεύοντα δένδρα ορίζουν μία διαμέριση του συνόλου εξόδου.

Κάθε αναζήτηση σε ένα T_v έχει πολυπλοκότητα $O(\log n + k_v)$, όπου k_v το πλήθος των σημείων στο T_v που ανήκουν στο ορθογώνιο. Υπάρχουν $O(\log n)$ κόμβοι v, άρα προκύπτει ο όρος $O(\log^2 n)$ στο συνολικό κόστος. Επιπλέον, $\sum_v k_v = k$, το οποίο μας οδηγεί στο τελικό αποτέλεσμα. □

Η ανανέωση των δένδρων περιοχής κοστίζει $O(n \log n)$, διότι στη χειρότερη περίπτωση πρέπει να ανανεωθούν $O(n)$ δένδρα T_v, καθένα σε λογαριθμικό χρόνο. Η πολυπλοκότητα αυτή βελτιώνεται αν συνδυάσουμε τα δένδρα περιοχής με πιθανοθεωρητικές δομές.

Τα δένδρα περιοχής γενικεύονται. Στο $\mathbb{R}^3$ αποθηκεύεται ένα βασικό δένδρο T ως προς τη συντεταγμένη x_1 και σε κάθε κόμβο ένα δένδρο περιοχής για δεδομένα στο $\mathbb{R}^2$. Επαγωγικά ορίζονται δένδρα περιοχής για κάθε $\mathbb{R}^d$, $d > 2$.

Θεώρημα 11.7 *Τα δένδρα περιοχής στο χώρο $\mathbb{R}^d$, για διάσταση $d \geq 2$, απαιτούν χώρο και χρόνο κατασκευής στο $O(n \log^{d-1} n)$, όπου n το πλήθος των αποθηκευμένων σημείων. Τα ορθογώνια ερωτήματα σε αυτά τα δένδρα απαντώνται σε χρόνο $O(\log^d n + k)$, όπου k το μέγεθος της εξόδου.*

Απόδειξη. Η ανάλυση της χωρικής πολυπλοκότητας ανάγεται σε αυτήν του χρόνου κατασκευής. Έστω $t_d(n)$ ο χρόνος κατασκευής του δένδρου περιοχής n σημείων στο $\mathbb{R}^d$. Προφανώς, $t_d(n) \geq n$, για κάθε σταθερό δείκτη d. Θα εφαρμόσουμε επαγωγή ως προς d, όπου η επαγωγική βάση, από τα προηγούμενα, είναι $t_2(n) = O(n \log n)$.

Η θεμελιώδης παρατήρηση είναι πως, αν θεωρήσουμε όλα τα δένδρα που αντιστοιχούν σε ένα επίπεδο του T, τότε αυτά περιέχουν όλα τα δεδομένα σημεία ακριβώς μία φορά το καθένα. Θα μελετήσουμε την κατασκευή όλων των δένδρων που αντιστοιχούν σε ένα επίπεδο του T. Έστω πως το επίπεδο έχει k κόμβους, άρα το συνολικό κόστος κατασκευής είναι ασυμπτωτικά $kt_{d-1}(n/k)$, το οποίο φράσσεται από το $t_{d-1}(n)$ διότι η συνάρτηση $t_d(n)$ μεγαλώνει πιο γρήγορα από τη γραμμική συνάρτηση ως προς n. Άρα

$$t_d(n) \leq n \log n + \log n \cdot t_{d-1}(n),$$

το οποίο, από την επαγωγική υπόθεση, φράσσεται από το $O(n \log^{d-1} n)$, για $d = O(1)$.

Στο κόστος των ερωτημάτων, αρκεί να φράξουμε εκείνο το κόστος διάσχισης που είναι ανεξάρτητο του k· έστω $q_d(n)$ αυτό το κόστος. Σε δύο διαστάσεις ήταν $q_2(n) = O(\log^2 n)$, ενώ γενικότερα είναι $q_d(n) = O(\log n + \log n \cdot q_{d-1}(n))$, το οποίο δίνει το επιθυμητό φράγμα μέσω επαγωγής ως προς d. □

11.3.1 Κλασματική επαλληλία

Μια γενική τεχνική, που βελτιώνει την απόδοση των δένδρων περιοχής, είναι η *κλασματική επαλληλία* (fractional cascading), η οποία προτάθηκε από τους Chazelle–Guibas [CG86]. Η τεχνική αυτή επιτυγχάνει μείωση της πολυπλοκότητας ορθογώνιων ερωτημάτων σε $O(\log n + k)$ στο επίπεδο και για γενική διάσταση σε

$$O(\log^{d-1} n + k),$$

όπου n το πλήθος των αποθηκευμένων σημείων, k το μέγεθος εξόδου και d η διάσταση. Η κλασματική επαλληλία διατηρεί τα άλλα φράγματα αναλλοίωτα, δηλαδή η κατανάλωση μνήμης και χρόνου για την κατασκευή της δομής είναι $O(n \log^{d-1} n)$.

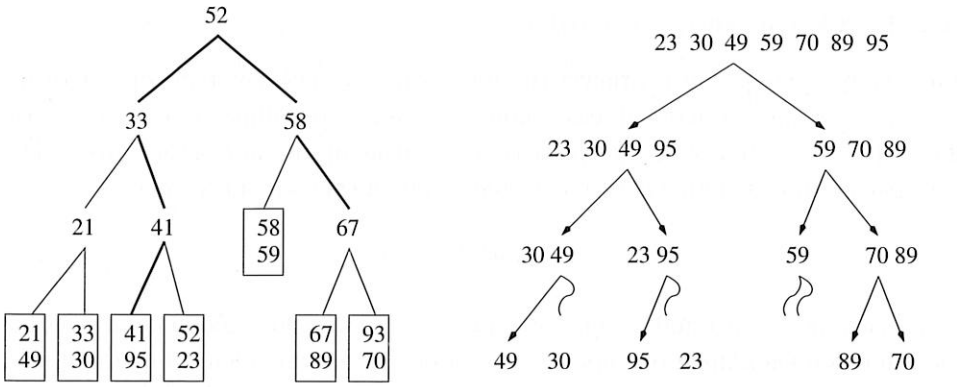
Η ιδέα είναι να συσχετίσουμε τις αναζητήσεις στα υποσύνολα των T_v, για διαφορετικούς κόμβους v, καθόσον αυτές είναι παρόμοιες. Αντί για δευτερεύοντα δένδρα, θα χρησιμοποιήσουμε ταξινομημένους πίνακες και θα εισάγουμε δείκτες μεταξύ των στοιχείων τους, οι οποίοι θα μεταφέρουν την απαραίτητη πληροφορία προς τους πίνακες των κόμβων–παιδιών. Τα στοιχεία που αποθηκεύονταν στο δένδρο T_v, για κάθε κόμβο v, αποθηκεύονται λοιπόν σε έναν πίνακα A_v, ο οποίος είναι ταξινομημένος ως προς την τεταγμένη y.

Κάθε στοιχείο $b \in A_v$ διαθέτει δείκτες προς ένα στοιχείο σε καθέναν από τους πίνακες A_l, A_r, οι οποίοι αντιστοιχούν στο αριστερό και στο δεξιό παιδί του κόμβου v. Οι δείκτες αυτοί συνδέουν το $b \in A_v$ με το ελάχιστο στοιχείο του αντίστοιχου πίνακα, που δεν είναι μικρότερο του b. Με άλλα λόγια, το σημείο με τεταγμένη b συνδέεται με το σημείο ελάχιστης τεταγμένης στον πίνακα A_l και A_r, η οποία δεν είναι μικρότερη του b.

Ο αλγόριθμος αναζήτησης εξετάζει το πρωτεύον δένδρο T, ακολουθώντας δύο διαδρομές ως προς τις τετμημένες του ορθογώνιου ερωτήματος. Στον κόμβο w του T, στον οποίο διαχωρίζονται οι δύο διαδρομές, διενεργείται μια δυαδική αναζήτηση στον πίνακα A_w ως προς τις τεταγμένες. Καθώς συνεχίζεται η αναζήτηση στο T, οι δευτερεύοντες πίνακες πού πρέπει να εξεταστούν είναι αντίστοιχοι με τα δευτερεύοντα δένδρα του αρχικού αλγορίθμου. Τώρα όμως, αντί η αναζήτηση να διασχίσει τα δευτερεύοντα δένδρα, χρησιμοποιεί τους δείκτες της κλασματικής επαλληλίας, ώστε να εντοπίσει σε σταθερό χρόνο ένα σημείο που ικανοποιεί το ερώτημα. Στη συνέχεια αναφέρει όλα τα σημεία του πίνακα που ανήκουν στο σύνολο εξόδου.

Παράδειγμα 11.2 Ας θεωρήσουμε το παράδειγμα που περιγράφεται από το σχήμα 11.6, το οποίο αφορά το σημειοσύνολο 7 σημείων του παραδείγματος 11.1 (βλ. και σχήμα 11.5). Δεξιά στο σχήμα 11.6 φαίνονται όλοι οι πίνακες A_v και ορισμένες συνδέσεις που χρησιμοποιούνται για την κλασματική επαλληλία.

Για την αναζήτηση της περιοχής $[35, 60] \times [20, 60]$, ο αλγόριθμος διασχίζει το T με τη διαδρομή $(52, 33, 41)$, η οποία καταλήγει στο φύλλο $(41, 95)$, και τη διαδρομή $(52, 58, 67)$, η οποία σταματά στον κόμβο του 67 που βρίσκεται εκτός x–περιοχής. Ο κόμβος w, όπου διαχωρίζονται οι διαδρομές, είναι η ρίζα του T, επομένως εκεί αρχίζει



Σχήμα 11.6: Κλασματική επαλληλία. Αριστερά: το πρωτεύον δένδρο περιοχής T για το παράδειγμα 11.2. Ο αλγόριθμος αναζήτησης για την περιοχή $[35, 60] \times [20, 60]$ διασχίζει τις έντονες ακμές. Δεξιά: όλοι οι πίνακες A_v, $v \in T$, με τις τεταγμένες των σημείων. Φαίνονται οι δείκτες του στοιχείου 49 και των απογόνων του, ενώ οι κενοί (null) δείκτες σημειώνονται με κατσαρές γραμμές. Υπάρχουν φύλλα στα οποία δεν καταλήγει κανείς από τους δείκτες του 49 και των απογόνων του.

η εξερεύνηση των πινάκων A_v ως προς τις τεταγμένες. Στην πρώτη διαδρομή, τα δεξιά υποδένδρα περιλαμβάνονται στην x–περιοχή. Υπάρχει μόνο αυτό του κόμβου 52, δηλ. της ρίζας, επομένως αρκεί η εξερεύνηση του πίνακα A_{52}. Αφού ανήκει στο y–διάστημα, το σημείο $(52, 23)$ τυπώνεται στην έξοδο, ενώ στο φύλλο $(41, 95)$ η τεταγμένη είναι εκτός y–διαστήματος. Ομοίως και για τα αριστερά υποδένδρα στη δεξιά διαδρομή (δηλαδή το 58). ■

Θεώρημα 11.8 *Με τα δένδρα περιοχής n σημείων στο χώρο $\mathbb{R}^d$, για $d \geq 2$, τα οποία χρησιμοποιούν κλασματική επαλληλία, τα ορθογώνια ερωτήματα απαντώνται σε χρόνο $O(\log^{d-1} n + k)$, όπου n το πλήθος των αποθηκευμένων σημείων και k το μέγεθος της εξόδου.*

Απόδειξη.

Ο αλγόριθμος ακολουθεί δύο διαδρομές στο T με κόστος $O(\log n)$, οι οποίες διαχωρίζονται σε κάποιον κόμβο w. Διενεργείται μια δυαδική αναζήτηση στον πίνακα A_w, κόστους $O(\log n)$, και εντοπίζονται όλα τα στοιχεία του πίνακα που ανήκουν στο διάστημα αναζήτησης. Από αυτά, μας ενδιαφέρουν τα δύο ακραία, των οποίων θα ακολουθήσουμε τους δείκτες προς τους πίνακες των παιδιών του w.

Η αναζήτηση σε κάθε πίνακα A_v που αντιστοιχεί σε κάποιον κόμβο v των δύο διαδρομών μετά τον κόμβο w απαιτεί δύο βήματα: Πρώτο, να ακολουθηθούν οι δείκτες από τον τρέχοντα πίνακα προς τους πίνακες των παιδιών του, το οποίο εκτελείται σε χρόνο $O(1)$. Δεύτερο, να αναφερθούν τα k_v σημεία του A_v που ανήκουν στο ορθογώνιο του ερωτήματος, το οποίο γίνεται σε χρόνο $O(k_v)$, εφόσον αρκεί να εξετάσουμε σειριακά τα στοιχεία του πίνακα που είναι γειτονικά με αυτά στα οποία μας οδήγησαν οι δύο δείκτες.

Με άλλα λόγια, η αναζήτηση στο υποσύνολο που αποθηκεύεται στον πίνακα A_v, για κάποιον απόγονο v του w, κοστίζει $O(1 + k_v)$. Το άθροισμα των k_v, για όλα τα v που συναντώνται, ισούται με k, ενώ το πλήθος των πινάκων A_v που εξετάζονται είναι στο $O(\log n)$. Άρα το θεώρημα αποδείχθηκε για σημεία στο επίπεδο. Αντίστοιχο αποτέλεσμα προκύπτει στο χώρο $\mathbb{R}^d$. □

Μια επιπρόσθετη βελτίωση του Chazelle μειώνει τη χωρική πολυπλοκότητα στο $O(n(\log n/\log\log n)^{d-1})$. Η δομή ονομάζεται δένδρα περιοχής κατά επίπεδα (layered range trees).

11.4 Δένδρα προτεραιότητας

Κλείνουμε την ενότητα με μια σύντομη αναφορά στα δένδρα προτεραιότητας (priority trees). Πρόκειται για δυαδικά δένδρα, προσαρμοσμένα σε ερωτήματα μη φραγμένου ορθογωνίου στο επίπεδο, δηλαδή ερωτήματα του τύπου $[-\infty, p] \times [q, q']$ για $p, q, q' \in \mathbb{R}$.

Η δομή βασίζεται στο σωρό (heap) και τοποθετεί στη ρίζα του αντίστοιχου (υπο)δένδρου το σημείο με την ελάχιστη τετμημένη. Η πρωτοτυπία έγκειται στο ότι χρησιμοποιεί τις y–συντεταγμένες για να ορίσει τα υποσύνολα των σημείων που αποθηκεύονται στα δύο υποδένδρα. Η μέση y–συντεταγμένη του αρχικού συνόλου αποθηκεύεται στη ρίζα του δένδρου, μαζί με το σημείο ελάχιστης τετμημένης. Τέλος, κάθε υποδένδρο είναι ένα δένδρο προτεραιότητας για το αντίστοιχο υποσύνολο σημείων.

Τα δένδρα προτεραιότητας μειώνουν την κατανάλωση μνήμης σε γραμμική (σε αντιδιαστολή με τα δένδρα περιοχής) δίχως να απαιτούν κλασματική επαλληλία. Κάθε ερώτημα μη φραγμένου ορθογωνίου απαντάται σε χρόνο $O(\log n + k)$, όπου k το μέγεθος της εξόδου.

Η κατασκευή των δένδρων προτεραιότητας απαιτεί χρόνο $O(n \log n)$ για n σημεία στο επίπεδο $\mathbb{R}^2$. Η απόδειξη των ιδιοτήτων των δένδρων προτεραιότητας αφήνεται ως άσκηση 11.9. Αν τα σημεία δίνονται ταξινομημένα ως προς την τεταγμένη τους, τότε η κατασκευή ολοκληρώνεται σε γραμμικό χρόνο, ακολουθώντας την κλασική κατασκευή σωρού, δηλαδή από τα φύλλα προς τη ρίζα (συνθετική, bottom up).

11.5 Ασκήσεις

Άσκηση 11.1 Χρησιμοποιήστε kd–δένδρο και δένδρο περιοχής για τις 4 παρακάτω αναζητήσεις στα δεδομένα του παραδείγματος 11.2: φραγμένη περιοχή $[1, 50] \times [30, 50]$, μη φραγμένη περιοχή $(-\infty, \infty) \times [30, 50]$, ευθεία $y = 30$, σημείο $(33, 30)$.

Άσκηση 11.2 Έστω kd–δένδρο για σύνολο n σημείων στο επίπεδο.

α) Σχεδιάστε αλγόριθμο που βρίσκει σε $O(\log n)$ τον πλησιέστερο γείτονα δεδομένου σημείου p, εφόσον τα σημεία είναι ομοιόμορφα κατανεμημένα στο περιγεγραμμένο τους παραλληλόγραμμο.

β) Δείξτε πως για αυθαίρετα σημεία, ο εντοπισμός του πλησιέστερου γείτονα απαιτεί $O(n)$ βήματα στη χειρότερη περίπτωση.

Άσκηση 11.3 Έστω kd–δένδρο για σύνολο σημείων στο επίπεδο. Σχεδιάστε αποδοτικό αλγόριθμο που εισάγει νέο σημείο στο δένδρο, χωρίς απαραίτητα να το εξισορροπεί στη συνέχεια.

Άσκηση 11.4 Έστω n σημεία στον χώρο $\mathbb{R}^d$, όπου το d είναι σταθερό. Αποδείξτε πως η κατασκευή του kd-δένδρου απαιτεί χρόνο $O(n \log n)$, ενώ το ορθογώνιο ερώτημα απαντάται σε χρόνο $O(n^{1-1/d} + k)$, όπου k το πλήθος των σημείων στο ορθογώνιο.

Άσκηση 11.5 Έστω kd–δένδρο διάστασης d με n σημεία. Μας ενδιαφέρει το πρόβλημα μερικής αντιστοίχισης, όπου δίνονται $s < d$ συντεταγμένες. Αποδείξτε πως για $d = 2, s = 1$, η πολυπλοκότητα του ερωτήματος οριζόντιας ευθείας στο επίπεδο είναι $O(\sqrt{n} + k)$. Ομοίως, για $d = 3, s = 2$, το ερώτημα ευθείας στο χώρο έχει πολυπλοκότητα $O(n^{1/3} + k)$.

Άσκηση 11.6 Έστω kd–δένδρο n σημείων στο επίπεδο. Αποδείξτε πως αν τεθεί ερώτημα, όπου το αντίστοιχο ορθογώνιο είναι απλό σημείο, τότε το ερώτημα κοστίζει $O(\log n)$. Ποια είναι η πολυπλοκότητα όταν πρόκειται για δένδρο περιοχής;

Άσκηση 11.7 Σε ορισμένες εφαρμογές, μας ενδιαφέρει μόνο ο υπολογισμός του πλήθους k των σημείων που βρίσκονται στο ορθογώνιο του ερωτήματος και όχι η εύρεση των σημείων. Στόχος μας είναι η αντίστοιχη πολυπλοκότητα να μην περιέχει τον όρο k.

α) Προσαρμόστε ένα μονοδιάστατο δένδρο περιοχής ώστε ο υπολογισμός του πλήθους να εκτελείται σε χρόνο $O(\log n)$.

β) Επεκτείνετε τη λύση στο (α) σε d–διάστατα ερωτήματα ώστε ο υπολογισμός του πλήθους να γίνεται σε χρόνο $O(\log^d n)$, για διάσταση $d \geq 2$.

γ) Χρησιμοποιήστε κλασματική επαλληλία ώστε η πολυπλοκότητα του (β) να μειωθεί κατά $O(\log n)$.

Άσκηση 11.8 Έστω σύνολο S_1 αποτελούμενο από n ξένα οριζόντια ευθύγραμμα τμήματα και σύνολο S_2 αποτελούμενο από m ξένα κατακόρυφα ευθύγραμμα τμήματα, στο επίπεδο. Σχεδιάστε αλγόριθμο σάρωσης που υπολογίζει το πλήθος των τομών στο σύνολο $S_1 \cup S_2$ με πολυπλοκότητα $O((n+m)\log(n+m))$. Οι αλγόριθμοι του κεφαλαίου 10 θα φανούν χρήσιμοι.

Άσκηση 11.9 Αποδείξτε πως τα δένδρα προτεραιότητας έχουν γραμμική κατανάλωση μνήμης, το ερώτημα μη φραγμένου ορθογωνίου απαντάται σε χρόνο $O(\log n + k)$ και η κατασκευή τους απαιτεί χρόνο $O(n \log n)$.

Άσκηση 11.10 Ορίζουμε ένα σύνθετο αριθμό $(a|b)$, για κάθε ζεύγος $a, b \in \mathbb{R}$, και τη λεξικογραφική διάταξη: $(a|b) < (a'|b')$ ανν $a < a'$, ή $a = a'$, $b < b'$. Ορίζουμε ένα σύνθετο σημείο

$$\widehat{p} = ((p_x|p_y), (p_y|p_x)) \in \mathbb{R}^2$$

για κάθε $p = (p_x, p_y) \in \mathbb{R}^2$. Η ενδιαφέρουσα ιδιότητα των σύνθετων σημείων είναι ότι καμία συντεταγμένη τους δεν επαναλαμβάνεται, εφόσον τα p είναι διαφορετικά. Για οποιοδήποτε ορθογώνιο $R = [x, x'] \times [y, y'] \subset \mathbb{R}^2$, έστω

$$\widehat{R} = [(x| - \infty), (x'| + \infty)] \times [(y| - \infty), (y'| + \infty)].$$

Αποδείξτε πως $p \in R \Leftrightarrow \widehat{p} \in \widehat{R}$.