Κεφάλαιο 7

Διαίρει και Κυρίευε

Το «Διαίρει και Κυρίευε» αποτελεί μία από τις θεμελιώδεις προγραμματιστικές τεχνικές. Βασική ιδέα είναι η αποσύνθεση του στιγμιότυπου ενός προβλήματος σε στιγμιότυπα μικρότερου μεγέθους, η ανεξάρτητη επίλυση τους και ο συνδυασμός των επιμέρους λύσεων προκειμένου να προκύψει η λύση του αρχικού στιγμιότυπου. Στο παρόν κεφάλαιο θα παρουσιαστούν αλγόριθμοι που υλοποιούν την ιδέα αυτή.

7.1 Βασικές αρχές

Η αναδρομή αποτέλεσε το κύριο θέμα προηγούμενου κεφαλαίου. Συνοπτικά, ένας αναδρομικός αλγόριθμος επιλύει το στιγμιότυπο ενός προβλήματος λύνοντας αναδρομικά μικρότερα στιγμιότυπα του ίδιου προβλήματος. Όταν σε κάθε αναδρομική κλήση συνδυάζονται λύσεις μικρότερων στιγμιότυπων *(υποπροβλημάτων)* τότε ο αλγόριθμος εμπίπτει στην κατηγορία «Διαίρει και Κυρίευε». Συγκεκριμένα, το υπόδειγμα «Διαίρει και Κυρίευε» περιλαμβάνει τρεις φάσεις σε κάθε επίπεδο αναδρομής (αναδρομικής κλήσης):

Διαίρει: το παρόν στιγμιότυπο διαιρείται (διασπάται) σε στιγμιότυπα μικρότερου μεγέθους του ιδίου προβλήματος,

Κυρίευε: τα μικρότερα στιγμιότυπα (υποπροβλήματα) επιλύονται αναδρομικά - αν είναι αρκετά μικρά επιλύονται απευθείας,

Συνδύασε: οι λύσεις των μικρότερων στιγμιοτύπων συνδυάζονται προκειμένου να προκύψει η λύση στο παρόν στιγμιότυπο.

Για να είναι επιτυχημένη η εφαρμογή της στρατηγικής αυτής θα πρέπει τα στιγμιότυπα που δημιουργούνται σε κάθε αναδρομική κλήση να είναι περίπου ιδίου μεγέθους.

Στις επόμενες ενότητες θα παρουσιαστούν αλγόριθμοι που ακολουθούν το παραπάνω πρότυπο.

7.2 Ταξινόμηση

Από τα πρώτα υπολογιστικά προβλήματα που συναντήσαμε είναι το πρόβλημα της ταξινόμησης (Ενότητα 3.2.1). Στην παρούσα ενότητα θα παρουσιάσουμε δυο αλγόριθμους που επιλύουν το πρόβλημα αυτό κάνοντας χρήση της μεθοδολογίας «Διαίρει και Βασίλευε». Για συμβατότητα με τους υπόλοιπους αλγόριθμους ταξινόμησης που έχουμε παρουσιάσει, θα θεωρήσουμε ότι η αταξινόμητη σειρά αφορά ακέραιους οι οποίοι βρίσκονται αποθηκευμένοι σε μία δομή πίνακα, ονομαστικά A, στις θέσεις από lo ως hi. Όμως αυτό γίνεται χωρίς βλάβη της γενικότητας αφού οι αλγόριθμοι αυτοί μπορούν πολύ εύκολα να μεταγραφούν και για άλλες δομές δεδομένων.

7.2.1 Ταχυταξινόμηση

Στην Ενότητα 4.2.4 παρουσιάσαμε τον Αλγόριθμο 10 οποίος δεδομένης μίας θέσης $pvt(lo \leq pvt \leq hi)$ τοποθετεί το στοιχείο $A[pvt]$ στη *σωστή θέση* - στη θέση που θα πρέπει να βρίσκεται όταν ο πίνακας ταξινομηθεί. Ο αλγόριθμος επιστρέφει τη θέση αυτή.
Μπορούμε να εκμεταλλευτούμε την παραπάνω αλγοριθμική διαδικασία ως εξής. Αφού τοποθετηθεί το στοιχείο στη σωστή θέση, έστω k, έχουμε αταξινόμητα δυο μέρη (τμήματα) του πίνακα: αυτό που περιέχει τα στοιχεία $A[lo], \cdots, A[k-1]$ και $A[k+1], \cdots, A[hi]$. Προφανώς αν με κάποιο τρόπο μπορέσουμε να ταξινομήσουμε τα στοιχεία των δύο αυτών μερών θα έχουμε την αρχική ακολουθία ταξινομημένη. Πως όμως θα ταξινομηθούν τα δύο αυτά μέρη; Η ιδέα είναι να χρησιμοποιήσουμε αναδρομικά την ίδια διαδικασία για καθένα από τα δύο μέρη. Σε κάθε επίπεδο αναδρομής παρατηρούμε ότι το πλήθος των στοιχείων είναι κατά ένα μεγαλύτερο από το άθροισμα του πλήθους των στοιχείων των δύο μερών (αφού ένα από τα στοιχεία έχει ήδη μπει στη σωστή θέση από τον Αλγόριθμο 10). Επομένως σε κάθε επίπεδο έχουμε να λύσουμε δύο στιγμιότυπα προβλήματος ταξινόμησης (υποπροβλήματα) με συνολικά μικρότερο αριθμό στοιχείων από το παρόν. Η βάση της αναδρομής προκύπτει όταν έχουμε να επιλύσουμε στιγμιότυπο ταξινόμησης με ένα (ή κανένα) στοιχείο. Στην περίπτωση αυτή δεν εκτελείται καμία εντολή και ο έλεγχος επιστρέφει στο προηγούμενο επίπεδο της αναδρομικής διαδικασίας. Η κωδικοποίηση αποτυπώνεται στον Αλγόριθμο 44. Το στοιχείο που τοποθετείται στη σωστή θέση από τη διαδικασία Pivot_Partition (Αλγόριθμος 10) είναι αυτό που αρχικώς βρίσκεται στη μεγαλύτερη θέση - θέση hi.

Αλγόριθμος 44 Ταχυταξινόμηση

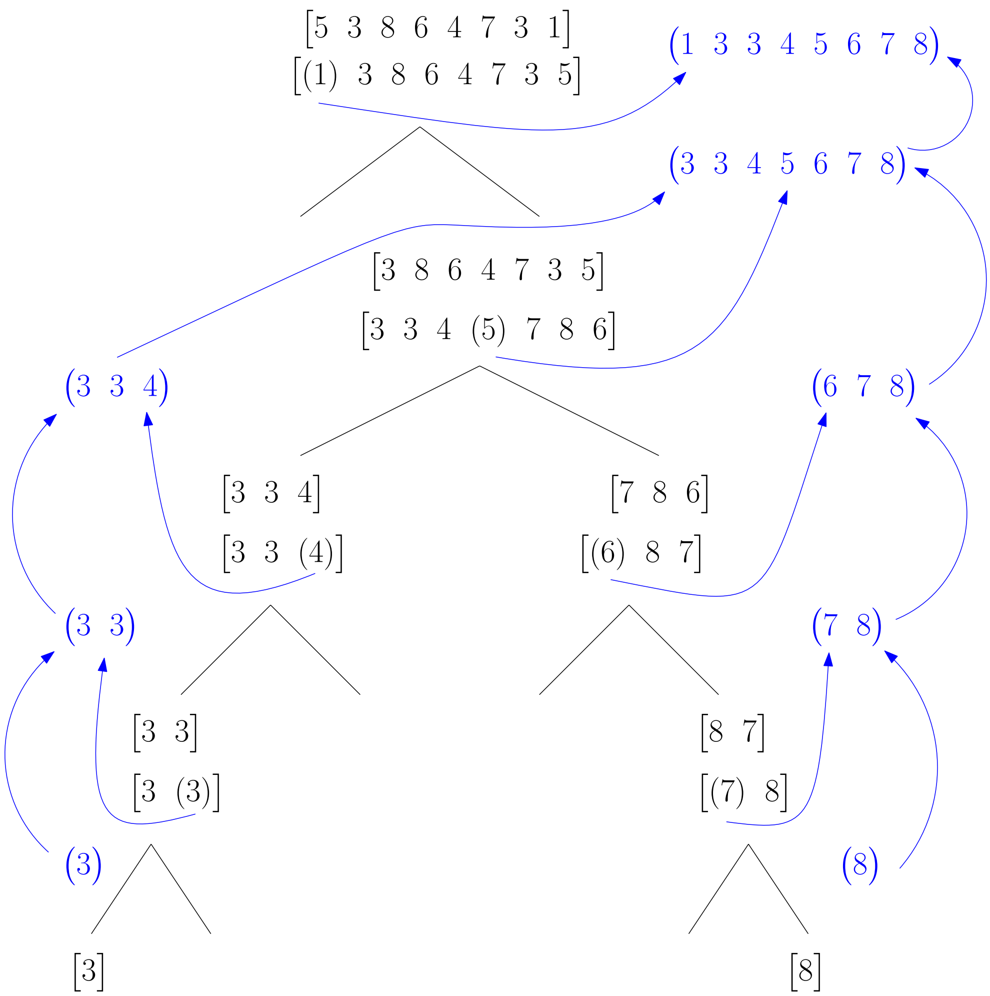
Απαιτείται: πίνακας A με στοιχεία στις θέσεις από lo έως hi
Επιστρέφεται: Ταξινομημένος πίνακας A

```
function Quicksort(int A[], int lo,int hi)
    if lb < ub then
        i ← Pivot_Partition(A, lo, hi, hi);
        Quicksort(A, lb, i − 1);
        Quicksort(A, i + 1, ub);
    end if
end function
```

Παράδειγμα 18. Για να ταξινομήσουμε τα στοιχεία του πίνακα A που βρίσκονται στις θέσεις από 1 ως 8, ήτοι

$$A = \begin{bmatrix} 5 & 3 & 8 & 6 & 4 & 7 & 3 & 1 \end{bmatrix},$$

με τον Αλγόριθμο 44 καλούμε τη συνάρτηση QuickSort$(A, 1, 8)$. Η διαδικασία υπολογισμού απεικονίζεται στο Σχήμα 7.1. Το σχήμα αναπαριστά το ιδεατό δένδρο (virtual tree) της αναδρομής. Σε κάθε κόμβο του δένδρου παρουσιάζεται η εικόνα του πίνακα



Σχήμα 7.1: Αλγόριθμος ταχυταξινόμησης, Παράδειγμα 18

A πριν και μετά την εκτέλεση της διαδικασίας τοποθέτησης ενός στοιχείου στη σωστή θέση· το στοιχείο αυτό απεικονίζεται σε παρένθεση. Οι κλάδοι που δεν καταλήγουν σε κάποιο κόμβο αντιστοιχούν σε στιγμιότυπα όπου δεν υπάρχει κανένα στοιχείο προς ταξινόμηση (περίπτωση κατά την οποία $lo > hi$). Στην περίπτωση αυτή όπως και στην περίπτωση που το στιγμιότυπο έχει μόνο ένα στοιχείο ο αντίστοιχος κόμβος είναι τερματικός και δεν εκτελείται καμία πράξη - εκτός από την πράξη της σύγκρισης των παραμέτρων lo, hi. Η σειρά των αριθμών σε παρενθέσεις[1] είναι η εικόνα του διανύσματος A - κατά το τμήμα που αφορά το παρόν στιγμιότυπο - που επιστρέφει η παρούσα αναδρομική κλήση κατά την ολοκλήρωση της.

Πολυπλοκότητα

Υπολογιστικά η συμπεριφορά της ταχυταξινόμησης δεν εξαρτάται μόνο από τον αριθμό των στοιχείων αλλά και από τον τρόπο που είναι αυτά αρχικώς διατεταγμένα στον πίνακα A. Για να υπολογίσουμε την πολυπλοκότητα της ταχυταξινόμησης αρχικώς παρατηρούμε ότι η διαδικασία Pivot_Partition (Αλγόριθμος 10) είναι τάξης $\Theta(n)$. Εφόσον ένα στοιχείο τοποθετείται από τον αλγόριθμο αυτό στη σωστή θέση παραμένουν προς ταξινόμηση $n-1$ στοιχεία εκ' των οποίων τα q $(0 \leq q \leq n-1)$ συνθέτουν το στιγμιότυπο της Γραμμής 5 και τα $n-q-1$ της Γραμμής 6. Επομένως, ο αριθμός των ΣΥΒ της χειρότερης περίπτωσης, ονομαστικά $T(n)$ ικανοποιεί τη σχέση

$$T(n) = \max\{T(q) + T(n-q-1) : 0 \leq q \leq n-1\} + \Theta(n). \tag{7.1}$$

Εικάζουμε ότι $T(n)$ είναι $O(n^2)$. Εφαρμόζοντας την εικασία αυτή στην παραπάνω σχέση, στους όρους $T(q), T(n-q-1)$, έχουμε , για κάποιο $c > 0$,

$$\begin{aligned} T(n) &\leq \max\{cq^2 + c(n-q-1)^2 : 0 \leq q \leq n-1\} + \Theta(n) \\ &= c\max\{q^2 + (n-q-1)^2 : 0 \leq q \leq n-1\} + \Theta(n) \end{aligned}$$

Η συνάρτηση

$$f(q) = q^2 + (n-q-1)^2 \tag{7.2}$$

είναι κυρτή· η δεύτερη παράγωγος της είναι ίση με μία θετική σταθερά. Επομένως παίρνει τη μεγαλύτερη τιμή της στα άκρα του πεδίου ορισμού της, ήτοι για $q = 0$ ή $q = n-1$. Για τις τιμές αυτές, έχουμε $f(0) = f(n-1) = (n-1)^2$. Αντικαθιστώντας στην προηγούμενη σχέση έχουμε

$$T(n) \leq c(n-1)^2 + \Theta(n) = cn^2 - c(2n-1) + \Theta(n) \leq cn^2$$

με την τελευταία ανισότητα να προκύπτει επιλέγοντας c αρκετά μεγάλο ώστε ο όρος $c(2n-1)$ να κυριαρχεί πάνω στον όρο $\Theta(n)$.

Η παραπάνω ανάλυση δείχνει ότι η χειρότερη περίπτωση για τον αλγόριθμο της ταχυταξινόμησης προκύπτει όταν το ένα από τα δύο στιγμιότυπα είναι κενό· η αντίστοιχη σειρά δεν περιέχει καθόλου στοιχεία. Είναι εύκολο να δει κανείς ότι αυτό προκύπτει όταν η σειρά περιέχει διαφορετικά στοιχεία και είναι εξ' αρχής ταξινομημένη ή ταξινομημένη σε φθίνουσα διάταξη. Όμως σε αυτή την περίπτωση σε κάθε κλήση μετά την διαδικασία Pivot_Partition μένει να ταξινομήσουμε μία σειρά με ένα στοιχείο λιγότερο και άρα ο αριθμός των ΣΥΒ εκφράζεται από την αναδρομική σχέση

$$T(n) = T(n-1) + \Theta(n).$$

[1] Εγχρωμη απεικόνιση.

Με τη μέθοδο της διαδοχικής αντικατάστασης δεν είναι δύσκολο να δείξει κάποιος ότι $T(n) = \Theta(n^2)$.

Όπως είδαμε προηγουμένως η χειρότερη περίπτωση στην ταχυταξινόμηση προκύπτει όταν το δένδρο αναδρομής είναι μη-ισορροπημένο. Διαισθητικά η καλύτερη περίπτωση πρέπει να προκύπτει όταν το δένδρο είναι πλήρως ισορροπημένο. Δηλαδή, όταν το στοιχείο που μπαίνει κάθε φορά στη σωστή θέση (από τη διαδικασία Pivot_Partition) διαμορφώνει δυο στιγμιότυπα με (περίπου) τον ίδιο αριθμό στοιχείων· στο ένα στιγμιότυπο που πρέπει να λυθεί αναδρομικά έχουμε $\lfloor \frac{n-1}{2} \rfloor$ στοιχεία ενώ στο άλλο $\lceil \frac{n-1}{2} \rceil$. Αν $T(n)$ είναι η συνάρτηση που δίνει τον αριθμό των ΣΥΒ στην περίπτωση αυτή, ισχύει ότι

$$T(n) = \begin{cases} 1, & n \leq 1, \\ T(\lfloor \frac{n-1}{2} \rfloor) + T(\lceil \frac{n-1}{2} \rceil) + \Theta(n), & n \geq 2. \end{cases} \tag{7.3}$$

Παρατηρούμε ότι $T(n) \geq \bar{T}(n)$ όπου

$$\bar{T}(n) = \begin{cases} 2\bar{T}(\lfloor \frac{n-1}{2} \rfloor) + \Theta(n), & n \geq 2, \\ 1, & n \leq 1. \end{cases}$$

Θα δείξουμε ότι $\bar{T}(n) = \Omega(n \lg n)$. Χρησιμοποιώντας τη μέθοδο της επαγωγής, Θα υποθέσουμε ότι αυτό ισχύει για $\bar{T}(\lfloor \frac{n-1}{2} \rfloor)$. Δηλαδή, έστω ότι υπάρχει $a > 0$ τέτοιο ώστε

$$\begin{aligned} \bar{T}(\lfloor \frac{n-1}{2} \rfloor) &\geq a\lfloor \frac{n-1}{2} \rfloor \lg\lfloor \frac{n-1}{2} \rfloor \Rightarrow \\ \bar{T}(n) &\geq 2a\lfloor \frac{n-1}{2} \rfloor \lg\lfloor \frac{n-1}{2} \rfloor + cn \\ &\geq 2a \left(\frac{n-1}{2} - 1 \right) \lg \left(\frac{n-1}{2} - 1 \right) + cn \\ &= 2a\frac{n-3}{2} \lg \frac{n-3}{2} + cn \Rightarrow \\ \bar{T}(\lfloor \frac{n-1}{2} \rfloor) &\geq a(n-3)(\lg(n-3) - 1) + cn. \end{aligned} \tag{7.4}$$

Παρατηρούμε ότι για $n \geq 4$, ισχύει[2]

$$\lg(n-3) - 1 \geq \lg n - 3. \tag{7.5}$$

Από (7.4), (7.5) έχουμε

$$\begin{aligned} \bar{T}(n) &\geq a(n-3)(\lg n - 3) + cn = an\lg n - 3a\lg n - 3an + 9a + cn \\ &\geq an\lg n - 3a\lg n - 3an + cn \\ &\geq an\lg n - 3an - 3an + cn = an\lg n - (6a-c)n \Rightarrow \\ \bar{T}(n) &\geq an\lg n, \end{aligned}$$

για $6a - c \leq 0 \Rightarrow \boxed{a \leq \frac{c}{6}.}$

Άρα $\bar{T}(n) = \Omega(n \lg n)$ και συνεπώς $T(n) = \Omega(n \lg n)$

---

[2]

$\lg(n-3) - 1 \geq \lg n - 3 \Rightarrow \lg(n-3) \geq \lg n - 2 \Rightarrow \lg(n-3) \geq \lg \frac{n}{4} \Rightarrow n - 3 \geq n/4 \Rightarrow \boxed{n \geq 4.}$

Πόρισμα 6. Ο αριθμός των ΣΥΒ που εκτελεί ο αλγόριθμος της ταχυταξινόμησης, για οποιοδήποτε στιγμιότυπο μεγέθους n, είναι τάξης $O(n^2)$ και $\Omega(n \lg n)$.

7.2.2 Συγχωνευτική ταξινόμηση

Μία από τις πλέον χαρακτηριστικές περιπτώσεις αλγορίθμου που εκφράζει το πνεύμα της μεθοδολογίας «Διαίρει και Κυρίευε» είναι η *συγχωνευτική ταξινόμηση (merge sort)*. Ο αλγόριθμος επιλύει το πρόβλημα της ταξινόμησης (Ενότητα 3.2.1) με τον εξής τρόπο. Αποσυνθέτει την αταξινόμητη σειρά στα δύο - επομένως από ένα στιγμιότυπο προβλήματος ταξινόμησης παράγονται δύο στιγμιότυπα με μισά στοιχεία στο καθένα (φάση «Διαίρει»). Τα στιγμιότυπα αυτά αναδρομικά αποσυντίθενται με τον ίδιο τρόπο (φάση «Κυρίευε») μέχρι να προκύψουν στιγμιότυπα με ένα (ή κανένα) στοιχείο - τέτοια στιγμιότυπα είναι εξ' ορισμού ταξινομημένα. Όταν σε μία κλήση τα δύο στιγμιότυπα που αποτελούνται από τα μισά στοιχεία είναι ταξινομημένα τότε μέσω της διαδικασίας της συγχώνευσης (Αλγόριθμος 12) μπορεί να επιλυθεί το πρόβλημα της ταξινόμησης στο παρόν στιγμιότυπο (φάση «Συνδύασε»).

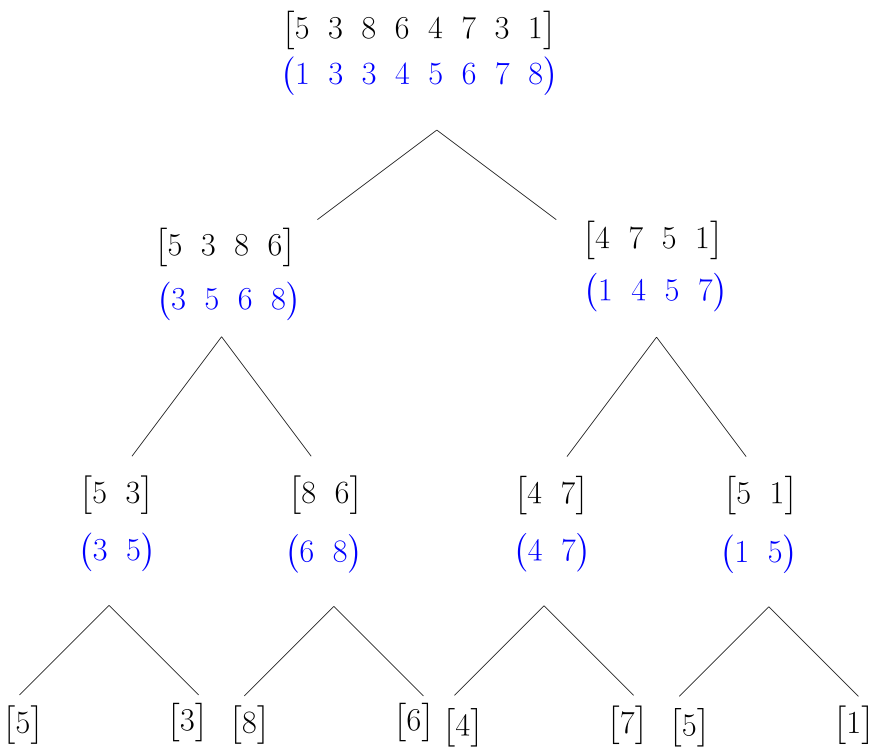
Αλγόριθμος 45 Συγχωνευτική Ταξινόμηση

Απαιτείται: πίνακας A με στοιχεία στις θέσεις από lo έως hi
Επιστρέφεται: Ταξινομημένος πίνακας A

function MergeSort(`int` $A[\,]$, `int` lo, `int` hi)
  if $lo < hi$ **then**
    $mid \leftarrow \lfloor \frac{ub+lb}{2} \rfloor$;
    Mergesort(A, lo, mid);
    Mergesort($A, mid+1, hi$);
    for $i \leftarrow lo; i \leq mid; i++$ **do**
      $A1[i] \leftarrow A[i]$;
    end for
    for $i \leftarrow mid+1; i \leq hi; i++$ **do**
      $A2[i] \leftarrow A[i]$;
    end for
    Merge($A1, lo, mid, A2, mid+1, hi, A, lo, hi$);
  end if
end function

Ο Αλγόριθμος 45 υλοποιεί την ιδέα με τη χρήση του Αλγόριθμου 12 για τη λειτουργία της συγχώνευσης. Οι φάσεις «Διαίρει» και «Κυρίευε» είναι σχετικά απλές και υλοποιούνται όπως περιγράφηκαν πρωτύτερα. Η φάση «Συνδύασε» περιγράφεται αναλυτικότερα ως εξής. Μόλις επιλυθούν τα δύο μικρότερα στιγμιότυπα, δηλαδή ταξινομηθούν οι σειρές $A[lo], \cdots, A[mid]$ και $A[mid+1], \cdots, A[hi]$ αντιγράφονται στους τοπικούς πίνακες $A1$ και $A2$ και στη συνέχεια με τη διαδικασία merge (Αλγόριθμος 12) τα στοιχεία τους συγχωνεύονται στον τοπικό πίνακα $A3$. Τα στοιχεία του πίνακα αυτού είναι τα στοιχεία του πίνακα A ταξινομημένα κατά αύξουσα σειρά· όμως τα στοιχεία αυτά καταλαμβάνουν τις θέσεις 1 ως $hi - lo + 1$ του πίνακα $A3$. Στη συνέχεια, τα στοιχεία αυτά αντιγράφονται στον πίνακα A στις θέσεις από lo ως hi.

Το δένδρο αναδρομής του Αλγόριθμου 45 για το στιγμιότυπο του Παραδείγματος 18 απεικονίζεται στο Σχήμα 7.2. Σε κάθε κόμβο του δένδρου σε τετράγωνες παρενθέσεις παρουσιάζεται η σειρά που πρέπει να ταξινομηθεί ενώ σε παρενθέσεις το ταξινομημένο διάνυσμα όπως επιστρέφεται από την συνάρτηση Merge (Αλγόριθμος 12)
.



Σχήμα 7.2: Αλγόριθμος συγχωνευτικής ταξινόμησης, Παράδειγμα 18

7.2.3 Πολυπλοκότητα

Όπως είναι εμφανές (και από το Σχήμα 7.2), ο αριθμός των ΣΥΒ που εκτελεί ο Αλγόριθμος 45 εξαρτάται μόνο από τον αριθμό των προς ταξινόμηση στοιχείων και όχι από τη διάταξη τους στην αταξινόμητη σειρά. Συγκεκριμένα, αν ο αριθμός των στοιχείων σε κάποια κλήση είναι $n = hi - lo + 1$, ζητείται η επίλυση ενός στιγμιότυπου ταξινόμησης με

$$\lfloor \frac{hi + lo}{2} \rfloor - lo + 1 \tag{7.6}$$

στοιχεία (Γραμμή 4) και ενός στιγμιότυπου με

$$hi - (\lfloor \frac{hi + lo}{2} \rfloor + 1) + 1 \tag{7.7}$$

στοιχεία (Γραμμή 5). Στη συνέχεια και αφού επιλυθούν τα δύο αυτά στιγμιότυπα (δηλαδή ταξινομηθούν οι αντίστοιχες σειρές), εκτελούνται $\Theta(n)$ ΣΥΒ για τη φάση της συγχώνευσης (Ενότητα 3.2.2).

Θα δείξουμε ότι η (7.6) είναι ίση με $\lceil \frac{n}{2} \rceil$. Εξ' ορισμού,

$$n - 1 = hi - lo \Rightarrow n - 1 + 2lo = hi + lo.$$

Διαιρώντας με 2 τα δύο μέλη, θεωρώντας το πάτωμα τους, μεταφέροντας τον όρο lo στο δεξί μέλος και προσθέτοντας σε αμφότερα το μέλη τη μονάδα, παίρνουμε την (7.6) ως

$$\lfloor \frac{hi + lo}{2} \rfloor - lo + 1 = \lfloor \frac{n-1}{2} \rfloor + 1 = \lfloor \frac{n+1}{2} \rfloor = \lceil \frac{n}{2} \rceil. \tag{7.8}$$

Συνεπώς για την (7.7) ισχύει ότι

$$hi - (\lfloor \frac{hi + lo}{2} \rfloor + 1) + 1 = \lfloor \frac{n}{2} \rfloor \tag{7.9}$$

Από την παραπάνω ανάλυση προκύπτει ότι ο αριθμός των ΣΥΒ του Αλγόριθμου 45 ικανοποιεί την αναδρομική σχέση

$$T(n) = \begin{cases} 1, & n \leq 1, \\ T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + \Theta(n), & n \geq 2. \end{cases}$$

Από το Θεώρημα 2 έχουμε ότι $T(n) = \Theta(n \lg n)$.

7.3 Πολλαπλασιασμός πινάκων

Το πρόβλημα του πολλαπλασιασμού πινάκων μπορεί να διατυπωθεί ως εξής: δοθέντων δύο πινάκων P, R θέλουμε να υπολογίσουμε το γινόμενο $Q = PR$.[3] Χωρίς βλάβη της γενικότητας θα θεωρήσουμε ότι οι πίνακες P, R είναι διάστασης $n \times n$, όπου $n = 2^k$.

[3]Εννοείται ότι οι πίνακες έχουν συμβατές διαστάσεις προκειμένου να μπορεί να πραγματοποιηθεί ο πολλαπλασιασμός.

Ο πλέον άμεσος τρόπος είναι να υπολογιστεί ο πίνακας Q είναι κάθε στοιχείο του $q_{ij}, i, j \in \{1, \ldots, n\}$ να προκύψει από την παράσταση

$$q_{i,j} = \sum_{t=1}^{n} p_{it} r_{tj}, \tag{7.10}$$

όπου p_{it} είναι το στοιχείο της γραμμής i και στήλης t του πίνακα P και r_{tj} είναι το στοιχείο της γραμμής t και στήλης j του πίνακα R. Ο αριθμός των ΣΥΒ που εκτελούνται προκειμένου να υπολογιστεί η τιμή του q_{ij} είναι τάξης $\Theta(n)$ αφού απαιτούνται n πολλαπλασιασμοί και $n-1$ προσθέσεις. Άρα ο υπολογισμός του πίνακα Q πραγματοποιείται σε $\Theta(n^3)$ ΣΥΒ. Θα δούμε στη συνέχεια ότι με την τεχνική «Διαίρει και Βασίλευε» μπορούμε να επιτύχουμε πολυπλοκότητα $o(n^3)$.

Παρατηρούμε ότι οι πίνακες P, R μπορούν να διαμεριστούν σε πίνακες διάστασης $\frac{n}{2} \times \frac{n}{2}$ με τον ακόλουθο τρόπο

$$P = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \; R = \begin{bmatrix} E & F \\ G & H \end{bmatrix}.$$

Χρησιμοποιώντας το παραπάνω σχήμα, μπορούμε να υπολογίσουμε τον πίνακα Q ως

$$Q = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}. \tag{7.11}$$

Η παραπάνω παράσταση παραπέμπει σε αλγοριθμική υλοποίηση τύπου «Διαίρει και Κυρίευε» αφού ο υπολογισμός του γινομένου δύο πινάκων διάστασης $n \times n$ μπορεί να γίνει με βάση τα γινόμενα οκτώ πινάκων διάστασης $\frac{n}{2} \times \frac{n}{2}$ τα οποία συνδυάζονται με τέσσερις πράξεις πρόσθεσης πινάκων αντίστοιχης διάστασης. Η αναδρομική σχέση που ικανοποιεί ο αριθμός των ΣΥΒ είναι

$$T(n) = \begin{cases} \Theta(1), & n = 1, \\ 8T(\frac{n}{2}) + cn^2, & n > 1 \end{cases}$$

Όμως και πάλι ο αριθμός των ΣΥΒ για τον υπολογισμό του πίνακα Q, όπως προκύπτει από το Θεώρημα 1 (πρώτη περίπτωση) είναι τάξης $\Theta(n^3)$. Στην εργασία [18] χρησιμοποιούνται οι υποπίνακες $A, \ldots, H$ που ορίστηκαν προηγουμένως προκειμένου να επιτευχθεί ο υπολογισμός σε ασυμπτωτικά καλύτερο χρόνο. Συγκεκριμένα, αποδεικνύεται με στοιχειώδη άλγεβρα πινάκων ότι ο πίνακας Q προκύπτει ως εξής:

$$Q = \begin{bmatrix} Q_1 + Q_2 - Q_4 + Q_6 & Q_4 + Q_5 \\ Q_6 + Q_7 & Q_2 - Q_3 + Q_5 - Q_7 \end{bmatrix} \tag{7.12}$$

όπου

$$\begin{aligned} Q_1 &= (B - D)(G + H), \\ Q_2 &= (A + D)(E + H), \\ Q_3 &= (A - C)(E + F), \\ Q_4 &= (A + B)H, \\ Q_5 &= A(F - H), \\ Q_6 &= D(G - E), \\ Q_7 &= (C + D)E. \end{aligned}$$

Η ορθότητα της διαδικασίας αποδεικνύεται εύκολα αντικαθιστώντας τους πίνακες $Q_t, t \in \{1, \ldots, 7\}$, στην (7.12) από τις παραπάνω ισότητες και κάνοντας αλγεβρικές πράξεις· σαν αποτέλεσμα προκύπτουν οι υποπίνακες που συνθέτουν τον Q στην (7.11). Ως παράδειγμα θα δείξουμε ότι $Q_4 + Q_5 = AF + BH$. Πράγματι,

$$Q_4 + Q_5 = (A + B)H + A(F - H) = AH + BH + AF - AH = AF + BH.$$

Παρατηρούμε ότι ο υπολογισμός του πίνακα $Q_t, t \in \{1, \ldots, 7\}$, προκύπτει από την επίλυση στιγμιότυπου του ιδίου προβλήματος (πολλαπλασιασμός πινάκων) με είσοδο πίνακες με τον μισό αριθμό γραμμών και στηλών από τους πίνακες P και R. Η παρατήρηση αυτή οδηγεί στον Αλγόριθμο 46 που χρησιμοποιεί τη μεθοδολογία «Διαίρει και Κυρίευε».

Στη γενική περίπτωση που η τιμή του n δεν είναι δύναμη του δύο συμπληρώνουμε τους πίνακες με μηδενικές γραμμές και στήλες μέχρι να φτάσουμε στον μικρότερη δύναμη του 2 που είναι μεγαλύτερη του n.

7.3.1 Πολυπλοκότητα

Ο αριθμός των ΣΥΒ ικανοποιεί την αναδρομική σχέση

$$T(n) = \begin{cases} \Theta(1), & n = 1, \\ 7T(\frac{n}{2}) + cn^2, & n > 1 \end{cases}$$

Σύμφωνα με την πρώτη περίπτωση του Θεωρήματος 1, $T(n) = \Theta(n^{\lg_2 7})$ και άρα ο υπολογισμός του γινόμενου δύο πινάκων μέσω του Αλγόριθμου 46 γίνεται ασυμπτωτικά γρηγορότερα από τον υπολογισμό που γίνεται τετριμμένα μέσω της (7.10).

7.4 Διάμεσος των διαμέσων

Στην Ενότητα 4.2.4 παραθέσαμε έναν αλγόριθμο για την εύρεση του k-ιοστού μικρότερου αριθμού σε μία αταξινόμητη σειρά n ακεραίων. Ο αλγόριθμος αυτός (Αλγόριθμος 27) εντάσσεται στη μεθοδολογία «Διαίρει και Κυρίευε» αφού σε κάθε αναδρομική κλήση επιλύει ένα στιγμιότυπο που είναι μικρότερου μεγέθους από το προηγούμενο. Ο αριθμός των ΣΥΒ που εκτελεί ο αλγόριθμος είναι τάξης $O(n^2)$ αφού σε κάθε αναδρομική κλήση, στη χειρότερη περίπτωση, ο αριθμός των στοιχείων του πίνακα A ανάμεσα στα οποία γίνεται η αναζήτηση μειώνεται μόλις κατά ένα. Οπότε είναι πιο αποτελεσματικό να ταξινομήσουμε τα στοιχεία σε αύξουσα σειρά (πολυπλοκότητα $O(n \lg n)$ με τη μέθοδο της συγχωνευτικής ταξινόμησης) και στη συνέχεια να επιλέξουμε απευθείας το $k-$ιοστό μικρότερο. Όμως το γεγονός ότι για ορισμένες τιμές της παραμέτρου k μπορούμε να λύσουμε το πρόβλημα σε γραμμικό χρόνο (π.χ. $k = 1$ ή $k = n$) προκρίνει την ιδέα ότι θα μπορούσε να γίνει το ίδιο για οποιαδήποτε τιμή της παραμέτρου k. Όντως στην εργασία [4] παρουσιάζεται μία τέτοια μέθοδος.

Η μέθοδος χρησιμοποιεί τον αναδρομικό Αλγόριθμο 27 επιλέγοντας το στοιχείο οδηγό $A[pvt]$ - άρα τη θέση pvt - όχι τυχαία αλλά με συγκεκριμένο τρόπο· θυμίζουμε ότι η επιλογή του οδηγού είναι κρίσιμη για τον αριθμό των στοιχείων ανάμεσα στα οποία θα πραγματοποιηθεί η αναζήτηση στην επόμενη κλήση (Γραμμές 9 και 11). Προκύπτει ότι αριθμός των στοιχείων στη χειρότερη περίπτωση είναι μικρότερος από ένα σημαντικό κλάσμα του n. Η διαδικασία είναι ένα σχήμα «Διαίρει και Κυρίευε» το οποίο βασίζεται στην έννοια της *διαμέσου*· σε μία σειρά στοιχείων, διάμεσος είναι το στοιχείο που είναι μικρότερο-ίσο από τα μισά στοιχεία της σειράς και μεγαλύτερο-ίσο από

Αλγόριθμος 46 Πολλαπλασιασμός πινάκων με τη μέθοδο Strassen

Απαιτείται: ακέραιος $n = 2^m$, για $m \geq 0$, πίνακες P, R διάστασης $n \times n$.

Επιστρέφεται: πίνακας $Q = PR$.

```
function MatrixMult(int n, int P[], int R[],int Q[])
    if n = 1 then
        Q[1,1] = P[1,1] · R[1,1];
    else
        half_n = n/2;
        for i ← 1; i ≤ half_n; i++ do
            for j ← 1; j ≤ half_n; j++ do
                A[i,j] ← P[i,j]; E[i,j] ← R[i,j];
                B[i,j] ← P[i,j+half_n]; F[i,j] ← R[i,j+half_n];
                C[i,j] ← P[i+half_n,j]; G[i,j] ← R[i+half_n,j];
                D[i,j] ← P[i+half_n,j+half_n]; H[i,j] ← R[i+half_n,j+half_n];
            end for
        end for
        for i ← 1; i ≤ half_n; i++ do
            for j ← 1; j ≤ half_n; j++ do
                BD[i,j] ← B[i,j] − D[i,j]; GH[i,j] ← G[i,j] + H[i,j];
                AD[i,j] ← A[i,j] + D[i,j]; EH[i,j] ← E[i,j] + H[i,j];
                AC[i,j] ← A[i,j] − C[i,j]; EH[i,j] ← E[i,j] + H[i,j];
                AB[i,j] ← A[i,j] + B[i,j]; FH[i,j] ← F[i,j] − H[i,j];
                GE[i,j] ← G[i,j] − E[i,j]; CD[i,j] ← C[i,j] + D[i,j];
            end for
        end for
        MatrixMult(half_n, BD, GH, Q1);
        MatrixMult(half_n, AD, EH, Q2);
        MatrixMult(half_n, AC, EF, Q3);
        MatrixMult(half_n, AB, H, Q4);
        MatrixMult(half_n, A, FH, Q5);
        MatrixMult(half_n, D, GE, Q6);
        MatrixMult(half_n, CD, E, Q7);
        for i ← 1; i ≤ half_n; i++ do
            for j ← 1; j ≤ half_n; j++ do
                Q[i,j] ← Q1[i,j] + Q2[i,j] − Q4[i,j] + Q6[i,j];
                Q[i,j+half_n] ← Q4[i,j] + Q5[i,j];
                Q[i+half_n,j] ← Q6[i,j] + Q7[i,j];
                Q[i+half_n,j+half_n] ← Q2[i,j] − Q3[i,j] + Q5[i,j] − Q7[i,j];
            end for
        end for
    end if
end function
```

τα υπόλοιπα. Αρχικώς χωρίζεται η σειρά των αταξινόμητων στοιχείων σε ομάδες των πέντε στοιχείων (πεντάδες) και υπολογίζεται η διάμεσος της κάθε ομάδας. Οι διάμεσοι των ομάδων διαμορφώνουν μία νέα αταξινόμητη σειρά. Η διάμεσος της σειράς αυτής *(διάμεσος των διαμέσων)* είναι ένα στοιχείο που πληροί την εξής ιδιότητα: τουλάχιστον $\frac{3n}{10}$ στοιχεία της αρχικής σειράς είναι μικρότερα-ίσα από αυτή. Άρα αν επιλεγεί το στοιχείο αυτό σαν οδηγός στα πλαίσια του Αλγόριθμου 27, το πλήθος των στοιχείων ανάμεσα στα οποία καλείται να γίνει η αναζήτηση στην επόμενη κλήση του αλγόριθμου (κλήση της διαδικασίας $k-$Element) (είτε στη Γραμμή 9 ή στην 11) δεν μπορεί να ξεπερνάει τα $\frac{7n}{10}$. Τέλος παρατηρούμε ότι ο υπολογισμός της διαμέσου των διαμέσων μπορεί να γίνει με την αναδρομική κλήση της παραπάνω διαδικασίας αναζητώντας τη διάμεσο στην αταξινόμητη σειρά των διαμέσων των πεντάδων.

Γιατί όμως η διάμεσος των διαμέσων έχει την προαναφερθείσα ιδιότητα; Παρατηρούμε ότι ο αριθμός των διαμέσων είναι ίσος με τον αριθμό των πεντάδων, δηλαδή $\frac{n}{5}$.[4] Η διάμεσος των διαμέσων είναι μεγαλύτερη-ίση από τις μισές από αυτές, δηλαδή από $\frac{\frac{n}{5}}{2} = \frac{n}{10}$ ενώ κάθε μία από αυτές είναι μεγαλύτερη-ίση από δύο στοιχεία (αφού κάθε ομάδα αποτελείται από πέντε στοιχεία). Επομένως η διάμεσος των διαμέσων είναι μεγαλύτερη-ίση τουλάχιστον από

$$\frac{n}{10} + 2\frac{n}{10} = \frac{3n}{10}$$

στοιχεία της αρχικής σειράς

Η διαδικασία εξειδικεύεται στον Αλγόριθμο 47 ο οποίος υπολογίζει τη θέση του $k-$ιοστού μικρότερου στοιχείου μίας αταξινόμητης σειράς ακεραίων. Ο αλγόριθμος αυτός μπορεί να θεωρηθεί μία εκτεταμένη έκδοση του Αλγόριθμου 27. Χρησιμοποιεί τη ρουτίνα Median_of_Five, ή οποία - δεν εξειδικεύεται και - αφορά τον υπολογισμό της διαμέσου σε μία ομάδα από πέντε ή λιγότερα στοιχεία. Μια άλλη σημαντική λεπτομέρεια της εξειδίκευσης είναι ότι προκειμένου να μην χρησιμοποιηθεί επιπλέον μνήμη, η σειρά των διαμέσων των πεντάδων αποθηκεύεται στις πρώτες θέσεις του πίνακα A ανταλλάσσοντας τα στοιχεία αυτά με τα στοιχεία που βρίσκονται στις αντίστοιχες θέσεις.

7.4.1 Πολυπλοκότητα

Θεωρούμε ότι ο υπολογισμός της διαμέσου μίας ομάδας που αποτελείται από πέντε (ή λιγότερα) στοιχεία πραγματοποιείται σε σταθερό χρόνο. Υπάρχουν $\lceil \frac{n}{5} \rceil$ τέτοιες ομάδες και επομένως ο αριθμός των ΣΥΒ για τον υπολογισμό των διαμέσων είναι $O(n)$. Για την επίλυση του αρχικού στιγμιότυπου απαιτείται

- η εύρεση της διαμέσου των διαμέσων, δηλαδή η επίλυση ενός στιγμιότυπου του ιδίου προβλήματος με $\lceil \frac{n}{5} \rceil$ στοιχεία,

- η τοποθέτηση της διαμέσου των διαμέσων στη σωστή θέση με την εκτέλεση της συνάρτησης Pivot_Partition (($O(n)$) ΣΥΒ) ,

- η επίλυση ενός στιγμιότυπου - πάλι του ιδίου προβλήματος - με το πολύ $\frac{7n}{10}$ στοιχεία.

[4] Η περίπτωση κατά την οποία το n δεν είναι πολλαπλάσιο του πέντε δεν διαφοροποιεί τη θεωρητική ανάλυση.

Αλγόριθμος 47 $k-$ιοστο μικρότερο στοιχείο

Απαιτείται: πίνακας A με στοιχεία στις θέσεις από lo έως hi, ακέραιος $k \in \{1, \ldots, hi - lo + 1\}$.

Επιστρέφεται: $k-$ιοστό μικρότερο στοιχείο του πίνακα A.

function Median_of_Medians(`int` $A[\,]$, `int` lo,`int` hi, `int` k)
    if $lo = hi$ **then**
        return lo
    end if
    $n \leftarrow hi - lo + 1;$
    $y \leftarrow n \bmod 5;$
    if $y = 0$ **then**
        $y \leftarrow 5;$
    end if
    $groups \leftarrow 0;$
    $first\_in\_group \leftarrow lo; last\_in\_group \leftarrow first\_in\_group + y - 1;$
    while $last\_in\_groups \leq hi$ **do**
        $group\_median \leftarrow$ Median_of_Five$(A, first_in_group, last_in_group);$
        Swap$(A[lo + groups], A[group_median]);$
        $groups++;$
        $first\_in\_group \leftarrow first\_in\_group + y;$
        $last\_in\_group \leftarrow first\_in\_group + 4;$
        $y \leftarrow 5;$
    end while
    $mid \leftarrow lo + \lfloor \frac{groups}{2} \rfloor - 1 + (groups \bmod 2);$ ▷ Αναδρομικός υπολογισμός θέσης της διαμέσου των διαμέσων
    $pvt \leftarrow$Median_of_Medians$(A, lo, lo + groups - 1, mid);$ ▷ Διαχωρισμός με βάση τη διάμεσο των διαμέσων
    $j \leftarrow$ Pivot_Partition$(A, lo, hi, pvt);$
    $i \leftarrow j - lo + 1;$
    if $i = k$ **then**
        return $A[j];$
    else if $i > k$ **then**
        return Median_of_Medians$(A, lo, j - 1, k);$
    else
        return Median_of_Medians$(A, j + 1, hi, k - i);$
    end if
end function

Συνολικά, ο αριθμός των ΣΥΒ που εκτελεί ο αλγόριθμος ικανοποιεί την αναδρομική σχέση[5]

$$T(n) = \begin{cases} \Theta(1), & n \leq 5, \\ T(\frac{n}{5}) + T(\frac{7n}{10}) + cn, & n \geq 6. \end{cases}$$

Το δεξί μέλος της παραπάνω σχέσης για $n \geq 6$ γράφεται ως

$$T(\frac{n}{5}) + T(\frac{7n}{10}) + cn = T(\frac{n}{\frac{10}{2}}) + T(\frac{n}{\frac{10}{7}}) + cn. \tag{7.13}$$

Για να υπολογίσουμε την τάξη της παράστασης αυτής με τη χρήση του Θεωρήματος 3 πρέπει να βρούμε το p για το οποίο

$$\left(\frac{2}{10}\right)^p + \left(\frac{7}{10}\right)^p = 1.$$

Δεν χρειάζεται να υπολογίσουμε το p ακριβώς· επειδή το άθροισμα των δύο λόγων είναι μικρότερο της μονάδας, έχουμε ότι $0 < p < 1$ και άρα

$$\int_1^n \frac{g(u)}{u^{p+1}} \, \mathrm{d}u = \int_1^n u^{-p} \, \mathrm{d}u = \frac{u^{1-p}}{1-p}\Big|_{u=1}^n = \frac{n^{1-p} - 1}{1-p} = \Theta(n^{1-p}).$$

Επομένως σύμφωνα με το Θεώρημα 3, η (7.13) είναι τάξης $\Theta(n^p \cdot (1 + \Theta(n^{1-p}))) = \Theta(n)$ και άρα $T(n) = O(n)$.

7.5 Εγγύτερο ζεύγος σημείων

Το πρόβλημα αυτό είναι πολύ απλό στη διατύπωση του: δεδομένου ενός συνόλου σημείων $P \subset \mathbb{R}^2$, να βρεθεί το ζεύγος των σημείων που ελαχιστοποιεί την μεταξύ τους απόσταση.

Χωρίς βλάβη της γενικότητας μπορούμε να θεωρήσουμε ως μέτρο την Ευκλείδεια απόσταση- δεδομένων δύο σημείων $x^1 = (x_1, y_1)$ και $x^2 = (x_2, y_2)$,

$$d(x^1, x^2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{7.14}$$

Έτσι το παραπάνω πρόβλημα έγκειται στην εύρεση σημείων $x^p, x^q \in P$ τέτοιων ώστε

$$d(x^p, x^q) = \min\{d(x^r, x^s) : x^r, x^s \in P\}.$$

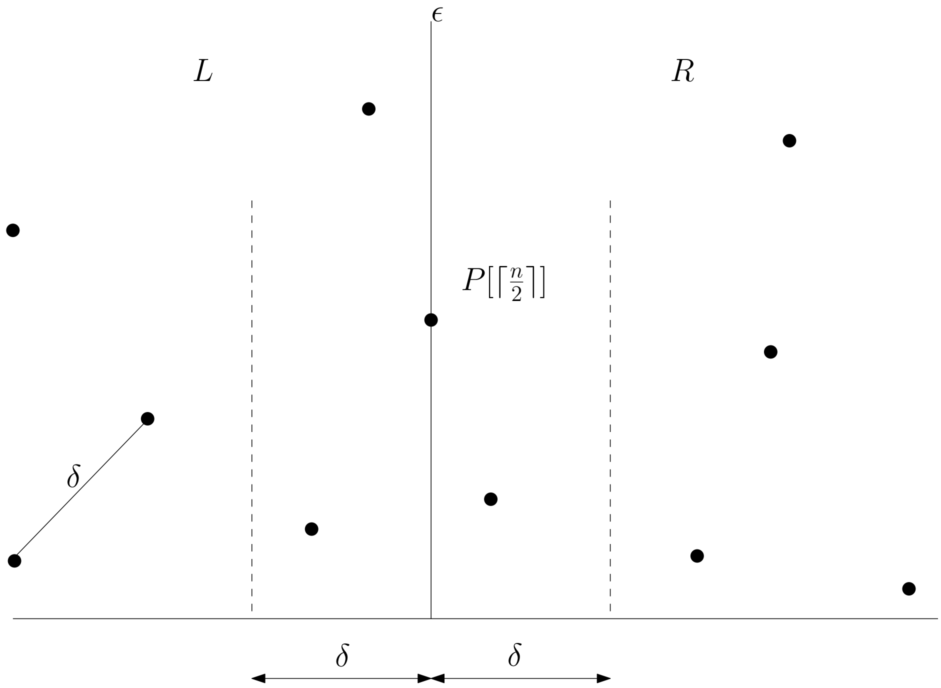
Ένας απλός τρόπος να επιλύσουμε το πρόβλημα αυτό είναι να υπολογίσουμε την ποσότητα (7.14) για κάθε ζευγάρι σημείων και να επιστρέψουμε το ζευγάρι που την ελαχιστοποιεί. Κάτι τέτοιο απαιτεί $O(n^2)$ ΣΥΒ, όπου $|P| = n$. Όπως θα δούμε στη συνέχεια μέσω ενός αναδρομικού σχήματος «Διαίρει και Κυρίευε» μπορούμε να επιτύχουμε χαμηλότερη πολυπλοκότητα.

Το αναδρομικό σχήμα λειτουργεί ως εξής. Αν το σύνολο P περιέχει το πολύ τρία σημεία $(n \leq 3)$, τότε υπολογίζουμε την (7.14) για κάθε ζευγάρι και επιστρέφεται μαζί με την ελάχιστη τιμή της το αντίστοιχο ζευγάρι (συνθήκη τερματισμού αναδρομής). Διαφορετικά (αναδρομική κλήση), έστω $x^m = (x_m, y_m)$ το σημείο με την ιδιότητα[6]

$$|L| = |R|,$$

[5] Για $n \geq 6$, το δεξί μέλος αποτελεί στην πραγματικότητα ένα άνω φράγμα στην τιμή της $T(n)$.

[6] Ο αριθμός των στοιχείων των συνόλων L, R μπορεί να διαφέρουν κατά ένα αν το n είναι περιττός.



Σχήμα 7.3: Εγγύτερο ζεύγος σημείων

όπου

$$L = \{x^j = (x_j, y_j) \in P : x_j \leq x_m\},$$
$$R = \{x^j = (x_j, y_j) \in P \setminus L : x_j \geq x_m\}.$$

Δηλαδή, η τετμημένη του σημείου x^m είναι η διάμεσος των τετμημένων των σημείων του P.
Έχουμε τρεις περιπτώσεις για το που μπορεί να βρίσκεται το ζητούμενο ζευγάρι των εγγύτερων σημείων: είτε μπορεί να βρίσκεται στο σύνολο L, είτε στο R, ή το ένα σημείο του ζεύγους στο L και το άλλο στο R Συνεπώς ορίζονται τρία στιγμιότυπα του ίδιου προβλήματος (φάση «Διαίρει»). Τα δύο πρώτα στιγμιότυπα έχουν περίπου το μισό αριθμό σημείων από το αρχικό ($\frac{n}{2}$). Για το τρίτο στιγμιότυπο μας ενδιαφέρουν μόνο τα σημεία του συνόλου

$$Q = \{x^j = (x_j, y_j) : |x_j - x_m| < \delta)\},$$

όπου $\delta = \min\{d_L, d_R\}$, όπου $d_L(d_R)$ η απόσταση του εγγύτερου ζεύγους σημείων του συνόλου $L(R)$.

Σχηματικά, φέρουμε μία κάθετη ευθεία (ϵ) από το σημείο x^m προς τον οριζόντιο άξονα και περιλαμβάνουμε στο Q όλα τα σημεία του P που απέχουν απόσταση μικρότερη από δ από την ευθεία αυτή - δες Σχήμα 7.3. Προφανώς $Q \subset P$ αφού τα σημεία που βρίσκονται σε απόσταση μεγαλύτερη-ίση από δ δεν ανήκουν στο Q.

Μας ενδιαφέρει το σύνολο Q αφού αν υπάρχει ένα ζευγάρι σημείων με απόσταση μικρότερη της δ τα σημεία αυτά θα ανήκουν στο σύνολο αυτό. Για κάθε σημείο $x^j \in Q$ ορίζουμε το σύνολο των σημείων

$$Q_j = \{x^k \in Q :, d(x^j, x^k) < \delta\}$$

και

$$d_j = \min\{d(x^j, q) : q \in Q_j\},$$

Έστω q^j το σημείο του συνόλου Q_j για το οποίο ισχύει $d(x^j, q^j) = d_j$. Σαν λύση του τρίτου στιγμιότυπου επιστρέφεται το ζευγάρι των σημείων x^j, q^j που επιτυγχάνει την μικρότερη τιμή d_j. Η τιμή αυτή είναι

$$d_Q = \min\{d_j : x^j \in Q_j\}. \tag{7.15}$$

Τέλος - φάση «Συνδύασε» - επιστρέφεται η μικρότερη από τις τιμές δ, d_Q με το αντίστοιχο ζευγάρι σημείων.

Για να λειτουργήσει αποδοτικά το παραπάνω αναδρομικό σχήμα θα πρέπει να επιλύεται με υπολογιστικά αποτελεσματικό τρόπο το τρίτο στιγμιότυπο. Προς τούτο, θεωρούμε τα σημεία του Q είναι διατεταγμένα κατά αύξουσα σειρά ως προς τις τεταγμένες τους - δηλαδή

$$\bar{Q} = (x^{i_1} = (x_{i_1}, y_{i_1}), x^{i_2} = (x_{i_2} y_{i_2}), \ldots, x^{i_r} = (x_{i_r}, y_{i_r}))$$

όπου $1 \leq r \leq n$ και για κάθε $k \geq 1$ και $k \leq r-1$ ισχύει

$$y_{i_k} \leq y_{i_{k+1}}.$$

Θα επαναορίσουμε τα σύνολα Q_j με βάση το διατεταγμένο πια σύνολο $\bar{Q}$ ως

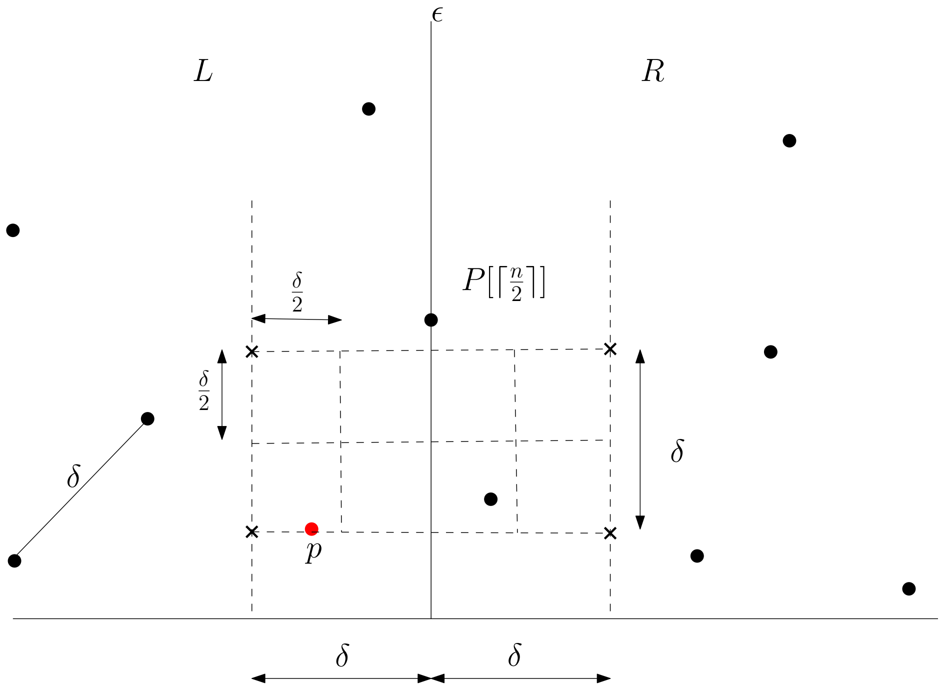
$$\bar{Q}_j = \{x^k = (x_k, y_k) \in \bar{Q} : y^k > y_j, d(x^k, x^j) < \delta\},$$

για κάθε $x^j = (x_j, y_j) \in \bar{Q}$.

Στο [2] αποδεικνύεται ένα εκπληκτικό αποτέλεσμα για κάθε $\bar{Q}_j$, ήτοι

$$\left|\bar{Q}_j\right| \leq 7.$$

Για να δούμε γιατί αυτό επαρκεί ας παρατηρήσουμε το Σχήμα 7.4. Έστω ότι το παρόν σημείο του $\bar{Q}$ που εξετάζεται είναι το σημείο p. Αν υπάρχει κάποιο σημείο με μεγαλύτερη-ίση τεταγμένη με το σημείο αυτό που να έχει απόσταση μικρότερη από $\delta = \min\{d_L, d_R\}$ τότε το σημείο αυτό θα βρίσκεται αναγκαστικά μέσα στο παραλληλόγραμμο που ορίζουν τα σημεία ×. Μπορούμε να διαιρέσουμε το χώρο αυτό σε οκτώ τετράγωνα πλευράς ίσης με $\frac{\delta}{2}$, το καθένα όπως φαίνεται στο Σχήμα 7.4. Θα δείξουμε ότι σε καθένα από τα τετράγωνα αυτά μπορεί να υπάρχει μόνο ένα σημείο με τεταγμένη μεγαλύτερη-ίση με το p που να έχει απόσταση από αυτό μικρότερη από δ. Έστω ότι δεν ισχύει αυτό και υπάρχει σε κάποιο τετράγωνο παραπάνω από ένα τέτοιο σημείο. Έστω q_1, q_2 δύο τέτοια σημεία. Επειδή τα σημεία αυτά ανήκουν στο ίδιο τετράγωνο η απόσταση ανάμεσα τους πρέπει να είναι μεγαλύτερη-ίση από δ αφού το τετράγωνο ανήκει εξ'ολοκλήρου σε κάποιον από τους δύο ημιχώρους τους οποίους ορίζει η ευθεία που περνάει από το σημείο που αποτελεί διάμεσο και επομένως είτε $q_1, q_2 \in L$, ή $q_1, q_2 \in R$. Όμως αυτό αποτελεί αντίφαση με το γεγονός ότι η μεγαλύτερη απόσταση που μπορεί να χωρίζει δύο σημεία σε κάθε τετράγωνο είναι ίση με το μήκος της διαγωνίου του τετραγώνου το οποίο είναι ίσο με $\frac{\delta}{\sqrt{2}}$ το οποίο είναι μικρότερο από δ. Συνεπώς υπάρχει το πολύ ένα σημείο σε κάθε τετράγωνο και το ένα από αυτά είναι ήδη κατειλημμένο από το p. Άρα μένουν επτά τετράγωνα σε καθένα από τα οποία μπορεί να υπάρχει το πολύ ένα σημείο με τεταγμένη μεγαλύτερη-ίση με το p που να



Σχήμα 7.4: Εγγύτερο ζεύγος σημείων

έχει απόσταση από αυτό μικρότερη από δ. Επομένως επαρκεί για κάθε σημείο του $\bar{Q}$ να υπολογιστούν αποστάσεις με τα επτά σημεία του (αν υπάρχουν) που έχουν αμέσως μεγαλύτερη-ίση τεταγμένη από αυτό.

Για να κωδικοποιηθεί η παραπάνω διαδικασία απομένει να διασαφηνιστούν κάποιες λεπτομέρειες. Ο αλγόριθμος δέχεται σαν είσοδο δυο πίνακες, ονομαστικά P, Q, που περιέχουν τα σημεία ανάμεσα στα οποία αναζητείται το εγγύτερο ζεύγος στις θέσεις από 1 έως n· στον πρώτο πίνακα τα σημεία είναι ταξινομημένα σε αύξουσα σειρά ως προς τις τετμημένες και στον δεύτερο ως προς τις τεταγμένες τους. Δηλαδή, σε όρους κωδικοποίησης, το σημείο που βρίσκεται αποθηκευμένο στη θέση j του πίνακα P έχει τετμημένη $P[j].x$, τεταγμένη $P[j].y$ και ισχύει ότι

$$P[i].x \leq P[i+1].x, i \in \{1, \ldots, n-1\}.$$

Αντίστοιχα, ισχύει ότι

$$Q[j].x \leq Q[j+1].x, j \in \{1, \ldots, n-1\}.$$

Σε κάθε κλήση, με πάνω από τρία σημεία, ο πίνακας Q «χωρίζεται στα δύο» τροφοδοτώντας τους πίνακες Q_L, Q_R με τα σημεία των συνόλων L και R, αντίστοιχα. Οι πίνακες αυτοί είναι ταξινομημένοι ως προς τις τεταγμένες και αποτελούν μαζί με τον πίνακα P την είσοδο στις κλήσεις που επιλύουν τα στιγμιότυπα που αφορούν τα σημεία των συνόλων L, R. Για το τρίτο στιγμιότυπο, χρησιμοποιούνται μόνο τα σημεία του Q που έχουν απόσταση μικρότερη από δ από την ευθεία ϵ. Τα σημεία αυτά καταχωρούνται σε αύξουσα σειρά ως προς τις τεταγμένες τους σε έτερο πίνακα, ονομαστικά Q_S. Ο Αλγόριθμος 48 κωδικοποιεί την διαδικασία η οποία καλείται στα πλαίσια ενός «κύριου προγράμματος» ως

ταξινόμησε τα σημεία ως προς τις τετμημένες παράγοντας τον πίνακα P.
ταξινόμησε τα σημεία ως προς τις τετμημένες παράγοντας τον πίνακα Q.
εκτύπωσε απόσταση: Closest_Pair$(P, 1, n, Q, n, p_1, p_2)$ και ζεύγος σημείων (p_1, p_2).

Η διαδικασία Closest_Pair παρουσιάζεται ως Αλγόριθμος 48. Δέχεται σαν είσοδο τον ταξινομημένο-ως-προς-τις-τετμημένες πίνακα P (τα σημεία καταλαμβάνουν τις θέσεις από lo ως hi,) τον ταξινομημένο-ως-προς-τις-τεταγμένες πίνακα Q (τα σημεία στον πίνακα αυτόν καταλαμβάνουν τις θέσεις από 1 ως n). Σαν έξοδο ο αλγόριθμος επιστρέφει στο όνομα της συνάρτησης την εγγύτερη απόσταση και τα σημεία για τα οποία επιτυγχάνεται η απόσταση αυτή στις παραμέτρους p_1, p_2. Ο αλγόριθμος χρησιμοποιεί τη συνάρτηση dist η οποία δέχεται σαν είσοδο δύο σημεία και υπολογίζει τη μεταξύ τους απόσταση.

Μία τελευταία παρατήρηση για τον Αλγόριθμο 48 αφορά τον διαμοιρασμό των στοιχείων στους πίνακες Q_L, Q_R. Θα πρέπει το σύνολο των σημείων του πρώτου πίνακα να ταυτίζεται με το σύνολο $\{P[lo], \ldots, P[mid]\}$ ενώ το δεύτερο με το σύνολο $\{P[mid{+}1], \ldots, P[hi]\}$. Για να διασφαλιστεί αυτό στην περίπτωση που κάποια σημεία έχουν την ίδια τετμημένη (ή τεταγμένη), θα πρέπει ο πίνακας P να είναι δευτερευόντως ταξινομημένος ως προς τις τεταγμένες των σημείων που περιέχει ενώ ο πίνακας Q να είναι δευτερευόντως ταξινομημένος ως προς τις τετμημένες. Συμπληρωματικά, θα πρέπει να ελέγχεται ότι ο αριθμός των στοιχείων του πίνακα Q_L είναι ίσος με

$$mid - lo + 1 = \lfloor \frac{lo + hi}{2} \rfloor - lo + 1$$

το οποίο από την (7.8) είναι ίσο με $\lceil \frac{n}{2} \rceil$.

7.5.1 Πολυπλοκότητα

Εφόσον μας δίνονται n στοιχεία, η ταξινόμηση των πινάκων P, Q είτε με την μέθοδο της συγχωνευτικής ταξινόμησης ή με τη χρήση σωρού παίρνει χρόνο $O(n \lg n)$.

Σε σχέση με τη διαδικασία Closest_Pair, όταν αληθεύει η συνθήκη τερματισμού της αναδρομής εκτελείται σταθερός αριθμός εκτελούνται - περίπτωση κατά την οποία $n \leq 3$. Για το δημιουργία των στιγμιότυπων που αφορούν τα σύνολα των σημείων L και R - τοποθέτηση σημείων στους πίνακες Q_L, Q_R, αντίστοιχα, εκτελούνται $\Theta(n)$ ΣΥΒ. Αν $T(n)$ είναι ο αριθμός των ΣΥΒ για την επίλυση του αρχικού προβλήματος, τότε $(\frac{n}{2})$ είναι ο αριθμός των ΣΥΒ που εκτελούνται για την επίλυση των δύο αυτών στιγμιοτύπων.[7] Για το τρίτο στιγμιότυπο, ο αριθμός των σημείων στον πίνακα Q_S δεν μπορεί να ξεπερνά το n και επειδή για κάθε στοιχείο ελέγχονται το πολύ τα επόμενα επτά (στον πίνακα Q_S), ο αριθμός των ΣΥΒ γαι την επίλυση του τρίτου στιγμιότυπου είναι $O(n)$.
Από τα παραπάνω προκύπτει ότι ο αριθμός των ΣΥΒ που εκτελεί ο Αλγόριθμος 48 (διαδικασία Closest_Pair) ικανοποιεί την αναδρομική σχέση

$$T(n) = \begin{cases} \Theta(1), & n \leq 3, \\ 2T(\frac{n}{2}) + O(n), & n \geq 4, \end{cases}$$

Αν στην παραπάνω αναδρομική σχέση ο όρος $O(n)$ αντικατασταθεί από τον $\Theta(n)$, μπορούμε, με τη χρήση του Θεωρήματος 2, να εκτιμήσουμε ότι η λύση της αναδρομικής σχέσης είναι τάξης $\Theta(n \lg n)$. Αποκαθιστώντας τον τελευταίο όρο της αναδρομικής σχέσης σε $O(n)$, παίρνουμε ότι η ασυμπτωτική εκτίμηση ισχύει ως προς το άνω

[7] Για την ακρίβεια της ανάλυσης, το ένα στιγμιότυπο επιλύεται σε $T(\lceil \frac{n}{2} \rceil)$ και το άλλο σε $T(\lfloor \frac{n}{2} \rfloor)$ ΣΥΒ. Όμως αυτή η λεπτομέρεια δεν επηρεάζει το αποτέλεσμα της ανάλυσης.

Αλγόριθμος 48 Εγγύτερο ζεύγος σημείων στο επίπεδο

Απαιτείται: Πίνακας P με σημεία αποθηκευμένα στις θέσεις από lo έως hi και ταξινομημένα κατά αύξουσα σειρά ως προς τις τετμημένες τους και Πίνακας Q με τα σημεία του πίνακα P αποθηκευμένα στις θέσεις από 1 έως n και ταξινομημένα κατά αύξουσα σειρά ως προς τις τεταγμένες τους.

Επιστρέφεται: Εγγύτερα σημεία p_1, p_2 που ανήκουν στον πίνακα P και η απόσταση τους.

function Closest_Pair(`point` $P[\,]$, `int` lo,`int` hi, `point` $Q[\,]$, `int` n, `point` p_1, `point` p_2)
    if $n \leq 1$ **then**
        $n \leftarrow hi - lo + 1;$
        $\delta \leftarrow \infty; p_1 \leftarrow (0,0); p_2 \leftarrow (0,0);$
    else if $2 \leq n \leq 3$ **then**
        for $(i \leftarrow lo; i \leq hi - 1; i++)$ **do**
            for $(j \leftarrow i + 1; i \leq hi; j++)$ **do**
                if $\delta >$ dist$(P[i], P[j]);$ **then**
                    $\delta \leftarrow$ dist$(P[i], P[j]); p_1 \leftarrow P[i]; p_2 \leftarrow P[j];$
                end if
            end for
        end for
    else
        $mid \leftarrow \lfloor \frac{lo+hi}{2} \rfloor;$
        $j \leftarrow 0; k \leftarrow 0;$
        for $i \leftarrow 1; i \leq n; i++$ **do**
            if $Q[i].x \leq P[mid].x$ **and** $j < mid - lo + 1$ **then**
                $j++; Q_L[j] \leftarrow Q[i]$
            else
                $k++; Q_R[k] \leftarrow Q[i]$
            end if
        end for
        $d_L \leftarrow$ Closest_Pair$(P, lo, mid, Q_L, j, q_1, q_2);$
        $d_R \leftarrow$ Closest_Pair$(P, mid + 1, hi, Q_R, k, r_1, r_2);$
        if $d_L < d_R$ **then**
            $\delta \leftarrow d_L; p_1 \leftarrow q_1; p_2 \leftarrow q_2;$
        else
            $\delta \leftarrow d_R; p_1 \leftarrow r_1; p_2 \leftarrow r_2;$
        end if
        $k \leftarrow 0;$
        for $(j \leftarrow 1; j \leq n; j++)$ **do**
            if $|Q[j].x - P[mid].x| < \delta$ **then**
                $k++; Q_S[k] \leftarrow Q[j];$
            end if
        end for
        for $(i \leftarrow 1; i \leq k - 1; i++)$ **do**
            $j \leftarrow 1;$
            while $i + j \leq k$ **and** $j \leq 7$ **do**
                if $\delta >$ dist$(Q_S[i], Q_S[i + j])$ **then**
                    $\delta \leftarrow$ dist$(Q_S[i], Q_S[i + j]);$
                    $p_1 \leftarrow Q_S[i]; p_2 \leftarrow Q_S[i + j];$
                end if
                $j++;$
            end while
        end for
    end if
    return $\delta;$
end function

φράγμα, δηλαδή $T(n) = O(n \lg n)$. Παρατηρούμε ότι ο αριθμός των ΣΥΒ που εκτελούνται αρχικώς (πριν την κλήση του Αλγόριθμου 48) για την ταξινόμηση των σημείων σε πίνακες P, Q, είναι ίδιας τάξης, επομένως συνολικός αριθμός ΣΥΒ για τον υπολογισμό εγγύτερου ζεύγους σημείων είναι τάξης $O(n \lg n)$.

7.6 Ασκήσεις

1. Ένας πίνακας A περιέχει n ταξινομημένους ακεραίους (θέσεις $lb, \ldots, ub$) διαφορετικούς μεταξύ τους. Να περιγράψετε έναν αλγόριθμο πολυπλοκότητας $\Theta(\lg n)$ που να βρίσκει αν υπάρχει στον πίνακα θέση k τέτοια ώστε $A[k] = k$.

2. Δίνεται πίνακας A ο οποίος στις θέσεις από 1 ως n περιέχει ακεραίους από το σύνολο $\{0, 1\}$. Θεωρούμε ότι ο πίνακας είναι ταξινομημένος σε αύξουσα σειρά.

   (α΄) Να περιγράψετε σε ψευδοκώδικα αλγόριθμο τύπου «Διαίρει και Κυρίευε» τάξης $\Theta(\lg n)$ ο οποίος να υπολογίζει τον πλήθος των στοιχείων του πίνακα που είναι ίσα με 1.

   (β΄) Να αποδείξετε την πολυπλοκότητα του αλγόριθμου.

3. Δοθέντος ενός συνόλου S n σημείων της γραμμής $\mathbb{R}$, θέλουμε να βρούμε το ζεύγος των σημείων $p, q \in S$ που ελαχιστοποιεί την απόσταση $|s - r|$, για $s, r \in S$. Να προτείνετε έναν αλγόριθμο για την επίλυση του προβλήματος αυτού με χρήση της τεχνικής «Διαίρει και Κυρίευε». Ποια είναι η πολυπλοκότητα του αλγόριθμου σας;

4. Έστω $S \subset \mathbb{R}^2$. Ένα σημείο $p = (p_x, p_y) \in S$ ονομάζεται *κυρίαρχο* αν δεν υπάρχει σημείο $q = (q_x, q_y) \in S$ τέτοιο ώστε $p_x \leq q_x$ και $p_y \leq q_y$. Να εκπονήσετε αλγόριθμο τάξης $o(n^2)$ που να υπολογίζει τα κυρίαρχα σημεία ενός δοθέντος συνόλου S.

5. Έστω n διαστήματα $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ επί της γραμμής $\mathbb{R}$. Σχεδιάστε και αναλύστε έναν αλγόριθμο που να υπολογίζει το μεγαλύτερο πλήθος διαστημάτων με μη-κενή τομή.

6. Δίνεται πίνακας A ο οποίος περιέχει ακέραιους στις θέσεις 1 ως n. Ένα ζευγάρι στοιχείων του πίνακα $A[i], A[j]$ βρίσκεται σε «αντιστροφή» αν $i < j$ αλλά $A[i] > A[j]$. Να εκπονήσετε έναν αλγόριθμο τύπου «Διαίρει και Κυρίευε» ο οποίος να μετράει τον αριθμό των ζευγαριών που βρίσκεται σε αντιστροφή σε $\Theta(n \lg n)$ βήματα. Να αποδείξετε την πολυπλοκότητα του ζητούμενου αλγόριθμου.

7. Έστω δύο ταξινομημένοι πίνακες A, B, μεγέθους n έκαστος. Περιγράψτε και αναλύστε έναν αλγόριθμο ο οποίος εντοπίζει τη διάμεσο των στοιχείων των δυο πινάκων.

8. Ένας πίνακας A έχει ένα *πλειοψηφούν* στοιχείο αν τα περισσότερα από τα μισά στοιχεία του είναι ίδια. Να εκπονηθεί αλγόριθμος πολυπλοκότητας $O(n \lg n)$, όπου n ο αριθμός των στοιχείων του πίνακα A, ο οποίος να προσδιορίζει αν ο πίνακας έχει πλειοψηφούν στοιχείο και αν ναι να βρίσκει το στοιχείο αυτό. Θεωρήστε ότι τα στοιχεία που βρίσκονται αποθηκευμένα στον πίνακα δεν είναι απαραίτητα αριθμοί και επομένως μπορεί να χρησιμοποιηθούν λογικές εκφράσεις τις μορφής $A[i] = A[j]$ ή $A[i] \neq A[j]$ και όχι $A[i] >= A[j]$ ή $A[i] > A[j]$.