# 2. Το πρόβλημα του Κυρτού Περιβλήματος στις Δύο Διαστάσεις

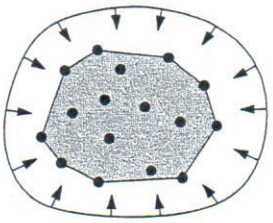
Μια από τις πιο χρήσιμες και ενδιαφέρουσες δομές στην υπολογιστική γεωμετρία είναι το κυρτό περίβλημα *(convex hull)*. Είναι ενδιαφέρον τόσο από μόνο του όσο και ως εργαλείο για την κατασκευή άλλων δομών σε ένα πλήθος διαφορετικών περιπτώσεων. Επίσης, έχει ιδιαίτερη σημασία για την υπολογιστική γεωμετρία γιατί μια από τις πρώτες δημοσιεύσεις στον τομέα αφορούσε στον υπολογισμό του κυρτού περιβλήματος. Από τότε, η έρευνα έχει αποδώσει μια μεγάλη ποικιλία αποτελεσμάτων, επιτυγχάνοντας τελικά βέλτιστους αλγορίθμους.

Προτού ασχοληθούμε με τη γεωμετρική υφή του προβλήματος, αναφέρουμε μερικές εφαρμογές:

1. *Αποφυγή εμποδίων.* Εάν το κυρτό περίβλημα ενός ρομπότ μπορεί να κινηθεί από μια αρχική θέση σε μια τελική θέση χωρίς να συγκρουστεί με εμπόδια που υπάρχουν στον χώρο, τότε το ίδιο μπορεί να κάνει και το ίδιο το ρομπότ. Καθώς ο υπολογισμός διαδρομών που αποφεύγουν εμπόδια είναι πιο εύκολος για ένα κυρτό (παρά για ένα μη κυρτό) ρομπότ, αυτή είναι η μέθοδος που χρησιμοποιείται συχνά στην εύρεση τέτοιων διαδρομών.
2. *Μικρότερο ορθογώνιο.* Το ορθογώνιο ελαχίστου εμβαδού το οποίο καλύπτει πλήρως ένα πολύγωνο έχει τουλάχιστον μία πλευρά του επάνω στον φορέα μιας πλευράς του κυρτού περιβλήματος του πολυγώνου. Γι' αυτό ο υπολογισμός του κυρτού περιβλήματος είναι το πρώτο βήμα σε αλγορίθμους υπολογισμού αυτού του ορθογωνίου. Όμοια, ο υπολογισμός του μικρότερου ορθογωνίου παραλληλεπιπέδου που περικλείει ένα τριδιάστατο αντικείμενο βασίζεται καθοριστικά στο κυρτό περίβλημα του αντικειμένου.
3. *Διάγραμμα Voronoi.* Το διάγραμμα Voronoi είναι μια άλλη πολύ χρήσιμη δομή σε προβλήματα υπολογιστικής γεωμετρίας και θα την εξετάσουμε σε επόμενο κεφάλαιο, οπότε και θα δούμε τη στενή σχέση που έχει με το κυρτό περίβλημα.

## 2.1 Ορισμοί

Το κυρτό περίβλημα ενός συνόλου σημείων στο επίπεδο μπορεί να περιγραφεί πρόχειρα ως εξής: εάν θεωρήσουμε ότι τα σημεία είναι καρφιά καρφωμένα σε ένα επίπεδο κομμάτι ξύλο, τότε το κυρτό περίβλημά τους είναι το πολύγωνο που σχηματίζεται όταν τεντώσουμε ένα λάστιχο γύρω από όλα τα καρφιά (Σχήμα 2.1). Αντίστοιχα, το κυρτό περίβλημα ενός συνόλου σημείων στις τρεις διαστάσεις είναι το πολύεδρο που σχηματίζεται όταν περιτυλίξουμε τα σημεία με μια καλά τεντωμένη ελαστική μεμβράνη.



Σχήμα 2.1

Αλλά ας δούμε κάποιους πιο αυστηρούς ορισμούς του κυρτού περιβλήματος. Υπενθυμίζουμε ότι ένα σύνολο $M$ είναι κυρτό εάν για κάθε ζεύγος σημείων $x$, $y$ του $M$ το ευθύγραμμο τμήμα $xy$ που συνδέει τα $x$ και $y$ ανήκει πλήρως στο $M$. (Σημειώνεται ότι ο ορισμός αυτός δεν θέτει κανένα περιορισμό στη διάσταση του συνόλου $M$.) Οι παρακάτω ορισμοί είναι όλοι ισοδύναμοι.

1. Το κυρτό περίβλημα ενός συνόλου σημείων $S$ στο επίπεδο είναι το μικρότερο κυρτό πολύγωνο $P$ που περικλείει το $S$ (το $P$ είναι το μικρότερο πολύγωνο με την έννοια ότι δεν υπάρχει άλλο κυρτό πολύγωνο $P'$ τέτοιο ώστε $S \subseteq P' \subset P$).
2. Το κυρτό περίβλημα ενός συνόλου σημείων $S$ στο επίπεδο είναι το κυρτό πολύγωνο με το μικρότερο εμβαδόν μεταξύ των κυρτών πολυγώνων που περικλείουν το $S$.
3. Το κυρτό περίβλημα ενός συνόλου σημείων $S$ στο επίπεδο είναι το κυρτό πολύγωνο με τη μικρότερη περίμετρο μεταξύ των κυρτών πολυγώνων που περικλείουν το $S$.
4. Το κυρτό περίβλημα ενός συνόλου σημείων $S$ στο επίπεδο είναι η ένωση όλων των κυρτών πολυγώνων με κορυφές από το σύνολο $S$. (Ο ορισμός αυτός γενικεύεται στις τρεις και παραπάνω διαστάσεις.)
5. Το κυρτό περίβλημα ενός συνόλου σημείων $S$ στο επίπεδο είναι η ένωση όλων των τριγώνων που ορίζονται από τριάδες σημείων από το σύνολο $S$. Ο ορισμός αυτός απλοποιεί τον προηγούμενο ορισμό και μας δίνει μια πρώτη ιδέα για έναν τρόπο υπολογισμού του κυρτού περιβλήματος.
6. Το κυρτό περίβλημα ενός συνόλου σημείων $S$ στο επίπεδο είναι η τομή όλων των ημιεπιπέδων που περιέχουν όλα τα σημεία του $S$. (Και ο ορισμός αυτός γενικεύεται στις τρεις και παραπάνω διαστάσεις.)

Στις επόμενες παραγράφους επικεντρώνουμε την προσοχή μας σε αλγορίθμους για το κυρτό περίβλημα, ξεκινώντας από πιο απλούς αλλά χρονικά δαπανηρούς, και καταλήγοντας τελικά σε βέλτιστους αλγορίθμους και αλγορίθμους που επεκτείνονται στις τρεις διαστάσεις.

Πριν να μελετήσουμε τους αλγορίθμους θα πρέπει να ασχοληθούμε με τη μορφή της εξόδου που επιθυμούμε να έχουν οι αλγόριθμοι. Πιο συγκεκριμένα, υπάρχουν δύο (ίσως διαφορετικά) προβλήματα: πρώτον, ο υπολογισμός των *ακραίων* σημείων, δηλαδή, των κορυφών του κυρτού περιβλήματος, και δεύτερον, ο υπολογισμός του συνόρου του. Η διαφορά έγκειται στο ότι η παράσταση του συνόρου του κυρτού περιβλήματος εμπεριέχει πολύ περισσότερη πληροφορία επιπλέον από τον καθορισμό των κορυφών του περιβλήματος. Για παράδειγμα, στις δύο διαστάσεις η παράσταση του συνόρου απαιτεί τον προσδιορισμό των κορυφών αλλά και της σειράς τους κατά μήκος του συνόρου, ενώ τα ακραία σημεία είναι απλά οι κορυφές ανεξάρτητα από τη σειρά τους. Αν και διαισθητικά ο υπολογισμός των ακραίων σημείων φαίνεται πιο εύκολος, στην πραγματικότητα δεν είναι (ως προς τον συμβολισμό-$O$).

Πιο αυστηρά, τα **ακραία σημεία** *(extreme points)* ενός συνόλου σημείων στο επίπεδο είναι οι κορυφές του κυρτού περιβλήματος στις οποίες η εσωτερική γωνία είναι μικρότερη από 180°. Δηλαδή, θεωρούμε μόνο τις "πραγματικές" κορυφές ως ακραία σημεία: σημεία που ανήκουν σε ακμές του κυρτού περιβλήματος δεν θεωρούνται ακραία.

Δοθέντος ενός συνόλου σημείων $S$ στο επίπεδο, πως μπορούμε να βρούμε ποια από αυτά είναι ακραία; Πρώτα-πρώτα, παρατηρούμε ότι το σημείο του $S$ με τη μεγαλύτερη $y$-συντεταγμένη είναι ακραίο, εάν είναι μοναδικό ή εάν υπάρχουν ακριβώς δύο τέτοια σημεία (και τα δύο είναι ακραία). Η ίδια παρατήρηση προφανώς ισχύει για τα σημεία με τη μικρότερη $y$-συντεταγμένη, και τη μεγαλύτερη και τη μικρότερη $x$-συντεταγμένη. Δεν είναι δύσκολο να δεί κανείς ότι ένα σημείο είναι ακραίο εάν και μόνο εάν υπάρχει μια ευθεία γραμμή η οποία περνά από το σημείο και δεν τέμνει πουθενά αλλού το κυρτό περίβλημα. Δυστυχώς όμως αυτός ο ορισμός δεν μεταφέρεται σε κάποιον αποδοτικό αλγόριθμο (όπως άλλωστε και αρκετοί ορισμοί που όπως αυτός βασίζονται στην ύπαρξη κάποιων "στοιχείων μαρτυρίας"). Γι' αυτό, θα προσπαθήσουμε να εντοπίσουμε τα μη ακραία σημεία.

## 2.2. Απλοί Αλγόριθμοι για Ακραία Σημεία

Οι αλγόριθμοι που θα περιγράψουμε σ' αυτήν την παράγραφο είναι μάλλον χρονοβόροι,

αλλά θα βοηθήσουν στη μελέτη των γρηγορότερων αλγορίθμων που θα επακολουθήσουν.

### 2.2.1. Μη Ακραία Σημεία

Προφανώς, ο υπολογισμός των σημείων που δεν είναι ακραία αρκεί για τον υπολογισμό των ακραίων σημείων.

**Λήμμα 2.2.1**
Ένα σημείο δεν είναι ακραίο εάν και μόνο εάν ανήκει σε κάποιο τρίγωνο με κορυφές σημεία του δοθέντος συνόλου και δεν είναι κάποια από τις τρεις κορυφές του τριγώνου.

**Απόδειξη** Η απόδειξη αυτού του λήμματος βασίζεται στον ορισμό 5 του κυρτού περιβλήματος που δώσαμε στην Παράγραφο 2.1. Τότε, είναι προφανές ότι ένα σημείο δεν είναι ακραίο εάν ανήκει στο εσωτερικό ενός τριγώνου, ενώ μπορεί να είναι ακραίο εάν είναι κορυφή του τριγώνου. Σημεία που ανήκουν στην περίμετρο του τριγώνου αλλά δεν είναι κορυφές, επίσης δεν είναι ακραία σημεία. ∎

Το παραπάνω λήμμα οδηγεί αμέσως στον Αλγόριθμο 2.1. Έστω $S = \{p_1, p_2, \ldots, p_n\}$ όπου όλα τα σημεία είναι διαφορετικά. Παρατηρήστε ότι δεν είναι απαραίτητος ο έλεγχος της δεύτερης συνθήκης του λήμματος, δηλαδή, ότι το $p_\ell$ δεν είναι κορυφή του τριγώνου: αφού τα σημεία στο $S$ είναι διαφορετικά και το $\ell$ είναι διαφορετικό από τα $i$, $j$ και $k$ στον βρόχο, αυτή η συνθήκη εξασφαλίζεται.

Εξ αιτίας των τεσσάρων nested βρόχων, ο αλγόριθμος απαιτεί $O(n^4)$ χρόνο: για καθένα από τα $O(n^3)$ τρίγωνα, ο έλεγχος για ακραία σημεία κοστίζει $O(n)$. Πρόκειται

```
/*  αλγόριθμος προσδιορισμού εσωτερικών σημείων  */
for (κάθε i) do
    for (κάθε j ≠ i) do
        for (κάθε k ≠ i, j) do
            for (κάθε ℓ ≠ i, j, k) do
                if (το p_ℓ είναι αριστερά της p_i p_j ή p_ℓ ∈ p_i p_j and
                    το p_ℓ είναι αριστερά της p_j p_k ή p_ℓ ∈ p_j p_k and
                    το p_ℓ είναι αριστερά της p_k p_i ή p_ℓ ∈ p_k p_i)
                then το σημείο p_ℓ δεν είναι ακραίο
```

**Αλγόριθμος 2.1** Εσωτερικά σημεία.

ίσως για τον πιο χρονοβόρο αλγόριθμο που θα μπορούσε να σκεφτεί κανείς για αυτό το πρόβλημα!

### 2.2.2. Ακραίες Ακμές

Είναι κάπως πιο εύκολο να προσδιορίσουμε τις *ακραίες ακμές*, τις ακμές του κυρτού περιβλήματος. Μια ακμή είναι ακραία εάν κάθε σημείο του $S$ είτε βρίσκεται από τη μία μόνο πλευρά της ευθείας γραμμής που ορίζει η ακμή είτε ανήκει σ' αυτήν. Για ευκολία, ας θεωρήσουμε ότι για κάθε ακμή $e$ έχουμε δύο κατευθυνόμενες ακμές με αντίθετες κατευθύνσεις και σε καθεμία από αυτές αντιστοιχίζουμε τη μία από τις δύο πλευρές που ορίζονται από την $e$, π.χ., την πλευρά στα αριστερά της κατευθυνόμενης ακμής. Τότε, ο προηγούμενος έλεγχος μπορεί να διατυπωθεί ισοδύναμα: μια κατευθυνόμενη ακμή δεν είναι ακραία εάν υπάρχει σημείο του $S$ που δεν είναι στα αριστερά της ακμής ούτε και ανήκει στην ακμή. Ο Αλγόριθμος 2.2 βασίζεται σ' αυτή τη διατύπωση.

Θα πρέπει να σημειώσουμε ωστόσο μια ανακρίβεια στον καθορισμό των ακραίων ακμών. Έστω ότι $xy$ είναι μια ακραία ακμή και το σημείο $z$ ανήκει στην $xy$. Τότε, επιπρόσθετα από την $xy$, και οι $xz$ και $zy$ θα χαρακτηρισθούν ως ακραίες ακμές με βάση τον παραπάνω έλεγχο, παρόλο που είναι λογικό ότι δεν θα θέλαμε κάτι τέτοιο. Δεν θα σταθούμε σ' αυτό το σημείο, όμως. Απλά, ας θεωρήσουμε ότι ο Αλγόριθμος 2.2 εκτελείται σε σύνολα σημείων *σε γενική θέση* (in general position): καμία τριάδα σημείων του συνόλου δεν ορίζει συνευθειακά σημεία.

Ο αλγόριθμος απαιτεί $O(n^3)$ χρόνο καθώς υπάρχουν τρεις nested βρόχοι καθένας από τους οποίους διατρέχει $O(n)$ τιμές: για καθένα από τα $n^2$ ζεύγη σημείων, ο έλεγχος για να διαπιστωθεί εάν ορίζουν μια ακραία ακμή απαιτεί $O(n)$ χρόνο. Έχοντας τώρα τις ακραίες ακμές (και με την υπόθεση ότι τα σημεία είναι σε γενική θέση), το σύνολο των ακραίων σημείων δεν είναι άλλο από το σύνολο των άκρων των ακραίων ακμών.

---

```
/*  αλγόριθμος προσδιορισμού ακραίων ακμών  */
for (κάθε i) do
    for (κάθε j ≠ i) do
        for (κάθε k ≠ i, j) do
            if (το p_k δεν είναι στα αριστερά της p_i p_j → και p_k ∉ p_i p_j)
            then το ευθ. τμήμα p_i p_j δεν είναι ακραία ακμή
```

---

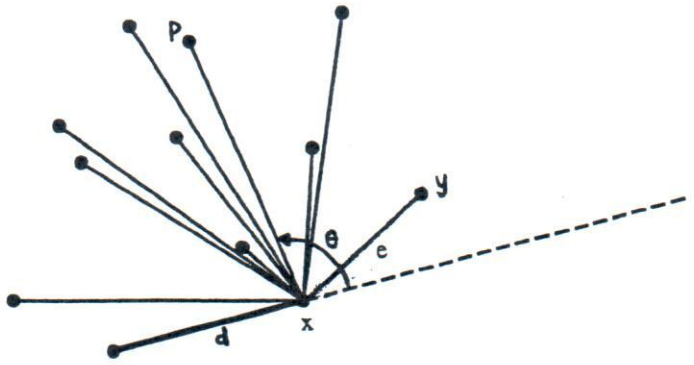
**Αλγόριθμος 2.2** Ακραίες ακμές.

## 2.3. Ο Αλγόριθμος Περιτυλίγματος (Gift Wrapping)

Μια μικρή τροποποίηση του αλγορίθμου για τον προσδιορισμό των ακραίων ακμών (Αλγόριθμος 2.2) μας επιτρέπει τόσο να τον επιταχύνουμε κατά ένα παράγοντα $n$ όσο και να παράγουμε τα ακραία σημεία με τη σειρά που εμφανίζονται κατά μήκος του συνόρου του κυρτού περιβλήματος. Η βασική ιδέα είναι να χρησιμοποιήσουμε την ακραία ακμή που μόλις βρήκαμε για να βρούμε την επόμενη. Η ιδέα αυτή δουλεύει γιατί οι ακραίες ακμές σχηματίζουν μια κλειστή πολυγωνική γραμμή. Δεδομένου ότι το κυρτό περίβλημα έχει το πολύ $O(n)$ κορυφές, θα έχει το πολύ $O(n)$ ακμές. Συνεπώς, θα ελέγξουμε $O(n)$ υποψήφιες ακραίες ακμές αντί για $O(n^2)$ όπως στον Αλγόριθμο 2.2. Έτσι, η πολυπλοκότητα χρόνου του αλγορίθμου ελαττώνεται σε $O(n^2)$.

Ας εξετάσουμε τώρα πως βρίσκουμε μια ακραία ακμή από την προηγούμενή της. Υποθέτουμε, για απλότητα, ότι τα σημεία του συνόλου εισόδου $S$ είναι σε γενική θέση: καμία τριάδα σημείων του $S$ δεν ορίζει συνευθειακά σημεία. Έστω ότι ο αλγόριθμος έχει μόλις βρει μια ακραία ακμή $d$ με ελεύθερο άκρο το $x$ (Σχήμα 2.2) και θέλουμε να βρούμε την άλλη ακραία ακμή $e$ με άκρο το $x$. Από το $x$ φέρουμε ημιευθείες $L_i$ προς όλα τα άλλα σημεία του συνόλου. Είναι καθοριστικό να παρατηρήσουμε ότι η νέα ακραία ακμή $e$ ορίζεται από το $x$ και το σημείο, έστω $y$, τέτοιο ώστε η αντίστοιχη ημιευθεία $L_y$ σχηματίζει τη μικρότερη γωνία (κατά την ανθωρολογιακή φορά) με την προηγούμενη ακραία ακμή $d$. Συνεπώς, για κάθε σημείο $p$ του $S$ υπολογίζουμε αυτή τη γωνία και έστω ότι είναι $\theta$. Τότε το σημείο για το οποίο η $\theta$ είναι ελάχιστη είναι το σημείο που ορίζει την ακραία ακμή (υπό την προϋπόθεση ότι τα δοθέντα σημεία είναι σε γενική θέση).

Ο λόγος που ο αλγόριθμος λέγεται *αλγόριθμος περιτυλίγματος* θα πρέπει να είναι τώρα προφανής: ο αλγόριθμος μπορεί να ερμηνευθεί ως μια διαδικασία περιτυλίγματος




Σχήμα 2.2

**Αλγόριθμος:** Gift Wrapping

Βρες το σημείο με την ελάχιστη $y$-συντεταγμένη και έστω $i_0$ ο δείκτης του

$i \leftarrow i_0$

`repeat`

  $k \leftarrow -1, \quad min\_\theta \leftarrow MAXNUM$

  `for` (κάθε $j \neq i$) `do`

    Υπολόγισε την αντίστοιχη γωνία $\theta$

    `if` $\theta < min\_\theta$

    `then` $k \leftarrow j, \quad min\_\theta \leftarrow \theta$

  η $\overline{p_i p_k}$ είναι ακμή του κυρτού περιβλήματος

  $i \leftarrow k$

`until` $i = i_0$

**Αλγόριθμος 2.3** Ο Αλγόριθμος Περιτυλίγματος.

του συνόλου των σημείων με μια ταινία που κάθε φορά κάμπτεται κατά την ελάχιστη γωνία μέχρι να συναντήσει κάποιο σημείο του συνόλου. Αυτός ο αλγόριθμος περιγράφηκε για πρώτη φορά από τους Chand και Kapur (1970) ως μια μέθοδος για τον υπολογισμό κυρτών περιβλημάτων σε οποιαδήποτε διάσταση. Θα δούμε ότι υπάρχουν καλύτεροι αλγόριθμοι για κυρτά περιβλήματα στις δύο διαστάσεις, αλλά για πολλά χρόνια ο αλγόριθμος περιτυλίγματος ήταν η κύρια μέθοδος υπολογισμού κυρτών περιβλημάτων στις τρεις και παραπάνω διαστάσεις.

Μια μικρή λεπτομέρεια που απομένει να διευκρινιστεί είναι πώς ξεκινά ο αλγόριθμος. Μπορούμε να θεωρήσουμε το σημείο με τη μικρότερη $y$-συντεταγμένη ως το πρώτο σημείο γύρω από το οποίο εκτελούμε τη λειτουργία του περιτυλίγματος παίρνοντας ως "προηγούμενη" ακμή του κυρτού περιβλήματος μια οριζόντια ακμή (Αλγόριθμος 2.3).

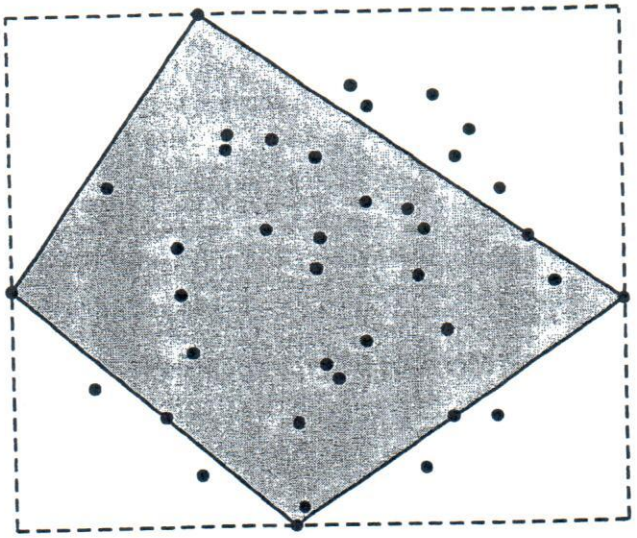
Ένα πολύ σημαντικό χαρακτηριστικό του αλγορίθμου περιτυλίγματος είναι ότι είναι *output-size sensitive*, δηλαδή, η πολυπλοκότητά του μπορεί να εκφραστεί ως συνάρτηση του μεγέθους της εξόδου: εάν $h$ είναι το πλήθος κορυφών του κυρτού περιβλήματος, η πολυπλοκότητα χρόνου είναι $O(nh)$. Με άλλα λόγια, απαιτεί λιγότερο χρόνο εάν το κυρτό περίβλημα έχει λίγες κορυφές. Στη χειρότερη περίπτωση βέβαια $h = O(n)$ και η πολυπλοκότητα χρόνου είναι $O(n^2)$: $O(n)$ χρόνος για κάθε ακμή του κυρτού περιβλήματος. Τέλος, σημειώνεται ότι ο αλγόριθμος (όπως τον περιγράψαμε παραπάνω), όπως και ο αλγόριθμος για τις ακραίες ακμές, θα πρέπει να τροποποιηθεί εάν θέλουμε να χειρίζεται και σύνολα σημείων που δεν είναι σε γενική θέση.

## 2.4. QuickHull

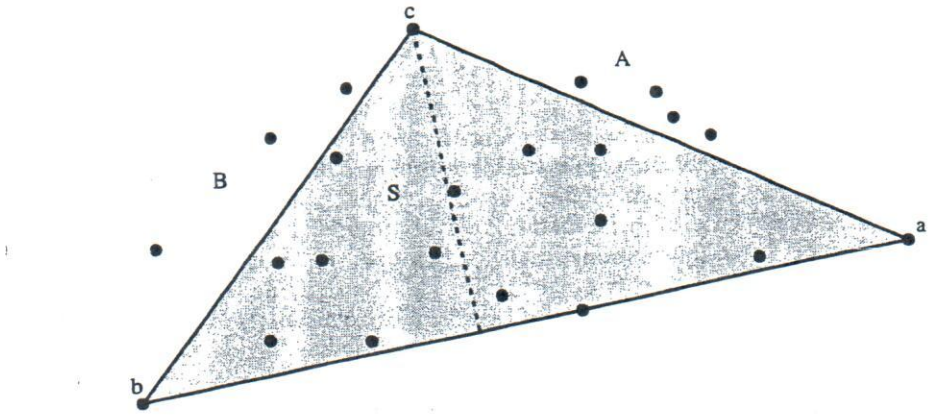
Ο αλγόριθμος αυτός προτάθηκε ανεξάρτητα από αρκετούς ερευνητές προς το τέλος της δεκαετίας του 1970. Ονομάστηκε *QuickHull* λόγω της ομοιότητάς του με τον αλγόριθμο ταξινόμησης QuickSort.

Η βασική ιδέα είναι απλή και έξυπνη: για τα "πιο πολλά" σύνολα σημείων, είναι εύκολο να απορρίψουμε αρκετά σημεία, επειδή βρίσκονται στο εσωτερικό του κυρτού περιβλήματος. Το πρώτο βήμα του αλγορίθμου QuickHull είναι η εύρεση των τεσσάρων ακραίων σημείων προς τα επάνω, κάτω, αριστερά και δεξιά. Ας υποθέσουμε για ευκολία ότι τα τέσσερα αυτά σημεία είναι διαφορετικά μεταξύ τους. Εάν τα συνδέσουμε έτσι ώστε να σχηματίσουν ένα τετράπλευρο, τότε κάθε άλλο σημείο στο εσωτερικό ή το σύνορο αυτού του τετραπλεύρου μπορεί να αγνοηθεί καθώς σίγουρα δεν είναι ακραίο σημείο (Σχήμα 2.3). Τώρα, το αρχικό πρόβλημα έχει αναχθεί σε τέσσερα ξεχωριστά προβλήματα υπολογισμού των κυρτών περιβλημάτων σε καθεμία από τις τέσσερις τριγωνικές περιοχές έξω από το τετράπλευρο. Ο αλγόριθμος τα επιλύει βρίσκοντας ένα ακραίο σημείο σε κάθε τρίγωνο, αγνοώντας κάποια σημεία και εκτελώντας αναδρομικές κλήσεις σε δύο μικρότερα σύνολα σημείων όπως περιγράφεται παρακάτω.

Σε κάθε κλήση, γνωρίζουμε δύο σημεία, $a$ και $b$, στο κυρτό περίβλημα (αρχικά, τα $a$ και $b$ είναι δύο από τα τέσσερα ακραία σημεία) και έχουμε ένα σύνολο $S$ από σημεία τα οποία βρίσκονται στα δεξιά της κατευθυνόμενης ευθείας $\overrightarrow{ab}$. Σκοπός μας είναι να βρούμε την αλυσίδα ακμών που περιβάλλει το $S$. Είναι σημαντικό να παρατηρήσουμε



**Σχήμα 2.3** Ο αλγόριθμος QuickHull αγνοεί τα σημεία μέσα στο τετράπλευρο.




**Σχήμα 2.4** Ο αλγόριθμος QuickHull αγνοεί τα σημεία στο τρίγωνο $(a, b, c)$ και καλείται αναδρομικά στα $A$ και $B$.

ότι το σημείο $c \in S$ που είναι μακρύτερα από την ευθεία $\overline{ab}$ είναι κορυφή του περιβλήματος: είναι ακραίο ως προς την κατεύθυνση που είναι κάθετη στην $\overline{ab}$ (δες Σχήμα 2.4). Συνεπώς, μπορούμε να αγνοήσουμε όλα τα σημεία στο σύνορο ή μέσα στο τρίγωνο $abc$ (εκτός βέβαια από τα $a$, $b$ και $c$) και να επαναλάβουμε την ίδια διαδικασία για το σύνολο $A$ των σημείων δεξιά της $\overrightarrow{ac}$ και το σύνολο $B$ των σημείων δεξιά της $\overrightarrow{cb}$ (Σχήμα 2.4). Το κύριο μέρος του αλγορίθμου μπορεί να συμπεριληφθεί σε μια αναδρομική συνάρτηση που παίρνει ως ορίσματα τα $a$, $b$ και μια λίστα με τα σημεία στο $S$ (Αλγόριθμος 2.4).

Η εύρεση του αρχικού τετραπλεύρου απαιτεί $O(n)$ χρόνο. Για την αναδρομική συνάρτηση, έστω $|S| = m$. Τότε, απαιτούνται $m$ βήματα για τον προσδιορισμό του ακραίου σημείου $c$, αλλά το κόστος των αναδρομικών κλήσεων εξαρτάται από τους πληθαρίθμους των $A$ και $B$. Εάν η πολυπλοκότητα χρόνου της συνάρτησης QuickHull για $m$ σημεία είναι $T(m)$, τότε θα ισχύει η αναδρομική σχέση: $T(m) = O(m) + T(|A|) + T(|B|)$. Η καλύτερη δυνατή περίπτωση προκύπτει όταν ο χωρισμός του $S$ σε $A$ και $B$ είναι όσο το δυνατόν ισοζυγισμένος: $|A| = |B| = m/2$. Τότε, έχουμε $T(m) = 2T(m/2) + O(m)$, που έχει την λύση $T(m) = O(m \log m)$. Συνεπώς η πολυπλοκότητα χρόνου του QuickHull είναι $O(n \log n)$ στην *καλύτερη* περίπτωση, η οποία θα προέκυπτε εάν το σύνολο σημείων ήταν τυχαία κατανεμημένο.

Η χειρότερη περίπτωση προκύπτει όταν η διαίρεση του $S$ σε $A$ και $B$ είναι όσο το δυνατόν περισσότερο ανισομερής: $|A| = 1$ και $|B| = m - 1$. Σ' αυτήν την περίπτωση, $T(m) = T(m-1) + O(m)$ που έχει τη λύση $T(m) = O(m^2)$. Δηλαδή, αν και ο αλγόριθμος QuickHull είναι γενικά ταχύς στην πράξη, είναι τετραγωνικός στη χειρότερη περίπτωση.

**Αλγόριθμος:** QuickHull

```
function QuickHull(a, b, S)
    if (S = {a, b}) then return ab
    else
        c ← σημείο του S σε μέγιστη απόσταση από την ευθεία ab
        A ← σημεία του S δεξιά της κατευθυνόμενης ευθείας ac
        B ← σημεία του S δεξιά της κατευθυνόμενης ευθείας cb
        return QuickHull(a, c, A) ακολουθούμενο από QuickHull(c, b, B)
```

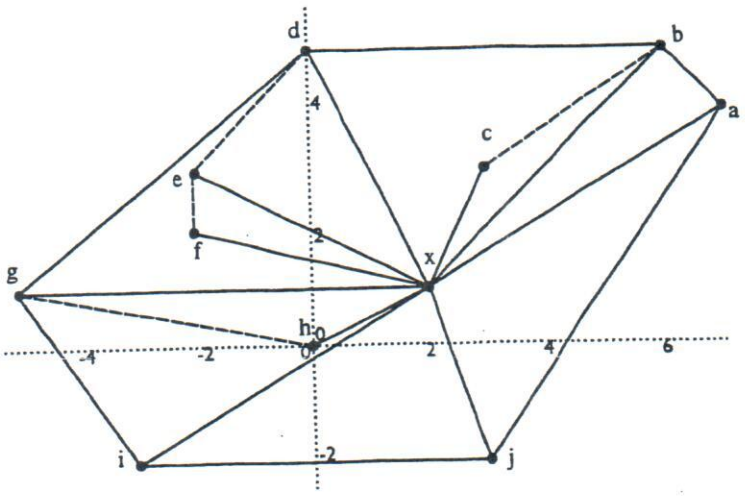
**Αλγόριθμος 2.4** Αλγόριθμος QuickHull.

## 2.5. Ο Αλγόριθμος του Graham

Ο αλγόριθμος του Graham έχει την τιμή να αντιπροσωπεύει ίσως την πρώτη εργασία που δημοσιεύτηκε σε υπολογιστική γεωμετρία. Στο τέλος της δεκαετίας του 1960, μια εφαρμογή στα Bell Labs απαιτούσε τον υπολογισμό του κυρτού περιβλήματος 10.000 σημείων στο επίπεδο και ο τετραγωνικός αλγόριθμος που ήδη υπήρχε απαιτούσε πολύ χρόνο. Με αφορμή την ανάγκη για έναν ταχύτερο αλγόριθμο, ο Graham (1972) επινόησε έναν αλγόριθμο ο οποίος υπολογίζει το κυρτό περιβλημα $n$ σημείων στο επίπεδο σε $O(n \log n)$ χρόνο.

Η βασική ιδέα του αλγορίθμου μπορεί να περιγραφεί με το ακόλουθο παράδειγμα. Έστω ότι δίνεται σημείο $x$ στο εσωτερικό του περιβλήματος και έστω επιπλέον ότι όλα τα σημεία (και το $x$) είναι σε γενική θέση. Ταξινομούμε τα σημεία με βάση τη γωνία γύρω από το $x$ με φορά αντίθετη από αυτή των δεικτών του ρολογιού (για το παράδειγμα του Σχήματος 2.5, τα ταξινομημένα σημεία είναι $a, b, \ldots, j$). Η επεξεργασία των σημείων θα γίνει με βάση την ταξινόμησή τους αυτή, και θα αρχίσουμε να σχηματίζουμε το κυρτό περίβλημα βαθμιαία γύρω από τα σημεία. Σε κάθε βήμα, το κυρτό περίβλημα που θα έχει υπολογιστεί θα είναι σωστό για τα σημεία που θα έχουμε εξετάσει μέχρι τότε, αν και σημεία που θα συναντήσουμε αργότερα μπορεί φυσικά να οδηγήσουν σε αναθεωρήσεις προηγούμενων αποφάσεων.

Κατά τη διάρκεια του υπολογισμού, το περίβλημα αποθηκεύεται σε μια στοίβα $T$ όπου τοποθετούμε σημεία. Αρχικά, η στοίβα περιέχει τα δύο πρώτα σημεία, $T = [a, b]$ στο παράδειγμά μας, με το $b$ στην κορυφή. Το σημείο $c$ ωθείται στη στοίβα γιατί τα τρία σημεία $a, b, c$ σχηματίζουν μια αριστερή στροφή στο $b$. Παρατηρήστε ότι τα περιεχόμενα της στοίβας $[a, b, c]$ σχηματίζουν μια κυρτή αλυσίδα. Αυτή είναι η σταθερή συνθήκη *(invariant)* του αλγορίθμου και θα τη διατηρήσουμε καθ' όλη τη διάρκεια της

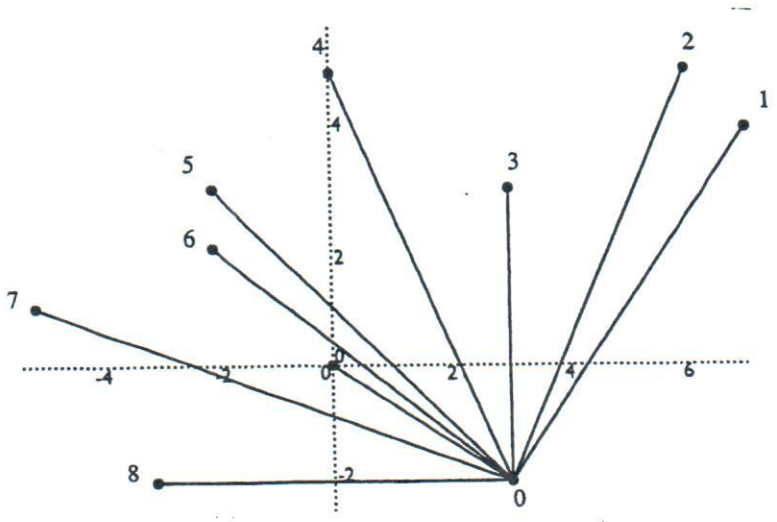



**Σχήμα 2.5** Ένα παράδειγμα εκτέλεσης του αλγορίθμου του Graham.

εκτέλεσής του. Στη συνέχεια, επεξεργαζόμαστε το σημείο $d$. Επειδή όμως τα σημεία $b, c, d$ σχηματίζουν μια δεξιά στροφή στο $c$ (που είναι στην κορυφή της στοίβας), η αλυσίδα δεν επεκτείνεται συμπεριλαμβάνοντας το $d$. Αντίθετα, η προηγούμενη απόφαση να συμπεριλάβουμε το $c$ αναθεωρείται και το $c$ απωθείται από τη στοίβα η οποία γίνεται πλέον $T = [a, b]$. Τώρα, προσθέτουμε το $d$, αφού τα $a, b, d$ σχηματίζουν μια αριστερή στροφή στο $b$.

Συνεχίζουμε με τον ίδιο τρόπο, ωθώντας τα $e$ και $f$ στη στοίβα, που γίνεται $T = [a, b, d, e, f]$. Το σημείο $g$ προκαλεί τη διαγραφή του $f$ και κατόπιν του $e$ από τη στοίβα, αφού τόσο η $[e, f, g]$ όσο και η $[d, e, g]$ είναι δεξιές στροφές. Κατόπιν, το $g$ προστίθεται στη στοίβα, που γίνεται $T = [a, b, d, g]$ και ούτω καθεξής. Εάν είμαστε τόσο τυχεροί όσο στο συγκεκριμένο παράδειγμα και το πρώτο σημείο $a$ ανήκει στο κυρτό περίβλημα τότε η κυρτή πολυγωνική αλυσίδα που κατασκευάζουμε θα κλείσει από μόνη της σχηματίζοντας το σύνορο του τελικού κυρτού περιβλήματος $T = [a, b, d, g, i, j, a]$. Εάν το $a$ δεν ανήκει στο κυρτό περίβλημα, το τέλος της αλυσίδας θα αρχίσει να περικλείει την αρχή της και η ανάλυση σ' αυτήν την περίπτωση γίνεται πιο δύσκολη. Θα δούμε ότι η τελευταία περίπτωση μπορεί να αποφευχθεί.

Στην παραπάνω περιγραφή, όπως και στον αρχικό αλγόριθμο του Graham, υποθέσαμε ότι το σημείο $x$ (με βάση το οποίο γίνεται η ταξινόμηση ως προς την γωνία) ανήκει στο εσωτερικό του κυρτού περιβλήματος, κάτι που μπορούμε εύκολα να εξασφαλίσουμε σε γραμμικό χρόνο. Ωστόσο, είναι πιο απλό να κανουμε την ταξινόμηση με βάση ένα σημείο του δοσμένου συνόλου $S$ και μάλιστα ακόμη καλύτερα με βάση μια κορυφή του κυρτού περιβλήματος. Θα χρησιμοποιήσουμε το σημείο με τη μικρότερη $y$-συντεταγμένη, το οποίο σίγουρα ανήκει στο περίβλημα. Σε περίπτωση περισσότερων




**Σχήμα 2.6** Νέα γωνιακή ταξινόμηση των σημείων του Σχήματος 2.5.

από ένα σημείων με την ίδια ελάχιστη $y$-συντεταγμένη, θα χρησιμοποιήσουμε το δεξιότερο από αυτά. Για το σύνολο σημείων του Σχήματος 2.5, αυτό θα ήταν το σημείο $j$. Η ταξινόμηση με βάση τη γωνία γύρω από το $j$ δίνει την εικόνα που παρουσιάζεται στο Σχήμα 2.6. Τα σημεία (εκτός από το $j$) χαρακτηρίζονται τώρα με έναν αριθμό από 1 έως $n-1$ που δηλώνει τη σειρά ταξινόμησης, ενώ το σημείο $j$ έχει τον αριθμό 0. Χρησιμοποιώντας αυτόν τον αριθμό, θα αναφερόμαστε στα $n$ σημεία ως $p_0$, $p_1$, ..., $p_{n-1}$, δηλαδή, το σημείο $j$ είναι τώρα το $p_0$, το σημείο $a$ είναι το $p_1$, κλπ.

Δεν είναι δύσκολο να παρατηρήσουμε ότι, επιπρόσθετα από το σημείο $p_0$, και τα $p_1$ και $p_{n-1}$ ανήκουν στο κυρτό περίβλημα για οποιοδήποτε αρχικό συνολο σημείων: τα $p_1$ και $p_{n-1}$ σχηματίζουν τη μικρότερη και τη μεγαλύτερη γωνία αντίστοιχα γύρω από το $p_0$. Έτσι, εάν αρχικοποιήσουμε τη στοίβα σαν $T = [p_{n-1}, p_0]$, πετυχαίνουμε δύο πράγματα: πρώτον, η στοίβα πάντα θα περιέχει δύο σημεία (εξασφαλίζοντας ότι ο υπολογισμός της αριστερής ή δεξιάς στροφής θα μπορεί να γίνεται πάντα), και δεύτερον, η αλυσίδα θα κλείσει ομαλά από μόνη της.

Ο αλγόριθμος του Graham παρουσιάζεται στον Αλγόριθμο 2.6. Υποθέτουμε την ύπαρξη των γνωστών λειτουργιών ώθησης (push) και απώθησης (pop) για στοίβες. Ο δείκτης $t$ δείχνει στην κορυφή της στοίβας. Επίσης έχει γίνει προσπάθεια να αντιμετωπιστούν προβλήματα με σημεία που είναι συνευθειακά. Συγκεκριμένα, κατά τη διάρκεια της ταξινόμησης, σημεία που σχηματίζουν την ίδια γωνία γύρω από το $p_0$ ταξινομούνται κατά αύξουσα απόσταση από το $p_0$. Αυτό σε συνδυασμό με το γεγονός ότι το τρέχον σημείο $p_i$ προστίθεται στην αλυσίδα (ωθείται στη στοίβα) μόνον εάν είναι αυστηρά αριστερά από την ευθεία που ορίζουν τα δύο τελευταία σημεία της αλυσίδας ενώ σε αντίθετη περίπτωση διαγράφεται από την αλυσίδα (απωθείται από τη στοίβα) το τελευταίο σημείο

**Αλγόριθμος:** Graham Scan

Βρές το δεξιότερο χαμηλότερο σημείο και ονόμασέ το $p_0$

Ταξινόμησε όλα τα άλλα σημεία με βάση τη γωνία γύρω από το $p_0$ (όπου μεταξύ σημείων με την ίδια γωνία προηγείται αυτό που είναι πιο κοντά στο $p_0$) και έστω ότι τα σημεία κατά αύξουσα σειρά είναι $p_1, \ldots, p_{n-1}$

Θεωρούμε στοίβα $T = [p_{n-1}, p_0]$ όπου το σημείο $p_0$ είναι στην κορυφή της $T$

$i \leftarrow 1$

`while` $i < n$ `do`

    $y, x \leftarrow$ τα σημεία στην κορυφή και στην αμέσως επόμενη θέση της στοίβας $T$

    `if` (το $p_i$ είναι αριστερά της κατευθυνόμενης ευθείας $\overrightarrow{xy}$)

    `then` Push$(T, i)$, $i \leftarrow i + 1$

    `else` Pop$(T)$

**Αλγόριθμος 2.6** Ο αλγόριθμος του Graham.

της, εξασφαλίζει ότι η στοίβα τελικά θα περιέχει μόνο ακραία σημεία. Είναι φανερό ότι ο βρόχος εκτελείται $O(n)$ φορές: κάθε απώθηση από τη στοίβα αφαιρεί για πάντα κάποιο σημείο, οπότε το συνολικό πλήθος ωθήσεων και απωθήσεων δεν υπερβαίνει τις $2n$. Δηλαδή, ο αλγόριθμος απαιτεί γραμμικό χρόνο μετά το βήμα ταξινόμησης των σημείων, το οποίο, ως γνωστόν, απαιτεί $O(n \log n)$ χρόνο. Θα δούμε στη συνέχεια ότι αυτό είναι το καλύτερο δυνατό που μπορούμε να πετύχουμε: η πολυπλοκότητα χρόνου του αλγορίθμου είναι *βέλτιστη στη χειρότερη περίπτωση* (worst-case optimal).

## 2.6. Κάτω Φράγμα

Δεδομένου ότι υπάρχει αλγόριθμος ο οποίος μας επιτρέπει να κατασκευάσουμε το κυρτό περίβλημα $n$ σημείων σε $O(n \log n)$ χρόνο, είναι επόμενο να αναρωτηθούμε εάν μπορούμε να πετύχουμε κάτι καλύτερο. Ίσως ένας γραμμικός αλγόριθμος είναι εφικτός. Θα δείξουμε σ' αυτήν την παράγραφο ότι κάτι τέτοιο δεν είναι δυνατόν: $n \log n$ είναι ένα *κάτω φράγμα* (lower bound) στην πολυπλοκότητα οποιουδήποτε αλγορίθμου για τον υπολογισμό του κυρτού περιβλήματος.

Υπενθυμίζουμε ότι ο συμβολισμός $\Omega(f(n))$ παριστά το σύνολο των συναρτήσεων $g$ για τις οποίες υπάρχει σταθερά $c$ τέτοια ώστε η απόλυτη τιμή της $g(n)$ είναι φραγμένη από κάτω από το γινόμενο του $c$ επί $f(n)$. Ο ορισμός αυτός γράφεται:

$$\Omega(f(n)) = \{g(n) : \exists c, n_0 \text{ τέτοια ώστε } |g(n)| \geq cf(n)\ \forall n > n_0\}.$$

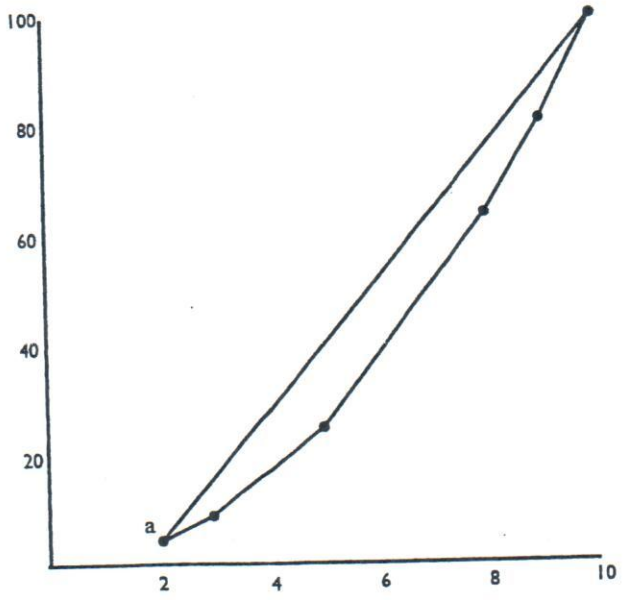
Αυτό που θα δείξουμε είναι ότι η πολυπλοκότητα κάθε αλγορίθμου για τον υπολογισμό του κυρτού περιβλήματος ανήκει στο σύνολο $\Omega(n \log n)$, ή, όπως συχνά λέγεται, ότι η κατασκευή του κυρτού περιβλήματος έχει ένα κάτω φράγμα $\Omega(n \log n)$.

Λόγω του ότι το κάτω φράγμα θα πρέπει να ισχύει για *οποιονδήποτε* αλγόριθμο, οι ερευνητές συνάντησαν δυσκολίες στο να υπολογίσουν μη τετριμμένα κάτω φράγματα για προβλήματα. Ευτυχώς, κάτω φράγματα έχουν επιτευχθεί για κάποια προβλήματα, για παράδειγμα, το πρόβλημα της ταξινόμησης: $\Omega(n \log n)$ είναι το κάτω φράγμα για την ταξινόμηση $n$ στοιχείων. Ο υπολογισμός ενός κάτω φράγματος για ένα πρόβλημα επιτρέπει τον υπολογισμό κάτω φραγμάτων για άλλα προβλήματα χάρις στη μέθοδο της *αναγωγής* (reduction).

Έστω ότι το πρόβλημα $A$ έχει κάποιο γνωστό κάτω φράγμα και έστω ότι το $A$ μπορεί να *αναχθεί* στο πρόβλημα $B$, υπό την έννοια ότι ένας αλγόριθμος για το πρόβλημα $B$ μπορεί να χρησιμοποιηθεί για την επίλυση του προβλήματος $A$ (με λίγη επιπλέον εργασία). Μια τέτοια αναγωγή συχνά οδηγεί σε ένα κάτω φράγμα για το πρόβλημα $B$, καθώς το γνωστό κάτω φράγμα για το $A$ θα παραβιάζονταν εάν μπορούσαμε να επιλύσουμε το πρόβλημα $B$ "πιο γρήγορα από όσο θα έπρεπε."

Αυτή τη μέθοδο θα χρησιμοποιήσουμε για να υπολογίσουμε ένα κάτω φράγμα για το κυρτό περίβλημα: θα δείξουμε ότι εάν είχαμε έναν αλγόριθμο για το κυρτό περίβλημα με πολυπλοκότητα χρόνου μικρότερη από $n \log n$, θα μπορούσαμε να ταξινομήσουμε $n$ στοιχεία σε λιγότερο από $n \log n$, το οποίο είναι αδύνατον.

Η ιδέα της ταξινόμησης μέσω του υπολογισμού του κυρτού περιβλήματος οφείλεται στον Shamos (1978) και είναι αρκετά απλή. Έστω ότι μας ζητείται να ταξινομήσουμε μια λίστα από αριθμούς $(x_1, x_2, \ldots, x_n)$, όπου $\forall i \; x_i \geq 0$: αυτό είναι το πρόβλημα $A$. Έστω επίσης ότι έχουμε έναν αλγόριθμο $B$ που υπολογίζει το κυρτό περίβλημα $n$ σημείων (σαν μια κλειστή κυρτή αλυσίδα) σε χρόνο $T(n)$. Θα χρησιμοποιήσουμε τον αλγόριθμο $B$ για να επιλύσουμε το πρόβλημα $A$ σε χρόνο $T(n)+O(n)$, όπου ο επιπλέον $O(n)$ χρόνος αντιστοιχεί στον χρόνο που απαιτείται για να μετατρέψουμε τη λύση που δίνει ο $B$ σε μια λύση του $A$. Από τη δοθείσα λίστα αριθμών σχηματίζουμε ένα σύνολο από διδιάστατα σημεία $(x_i, {x_i}^2)$ (Σχήμα 2.7). Τα σημεία αυτά ανήκουν σε μια παραβολή. Χρησιμοποιούμε τον αλγόριθμο $B$ για να υπολογίσουμε το κυρτό περίβλημα. Προφανώς, κάθε σημείο ανήκει στο περίβλημα. Βρίσκουμε το χαμηλότερο σημείο $a$ του κυρτού περιβλήματος σε $O(n)$ χρόνο: το σημείο αυτό αντιστοιχεί στο μικρότερο $x_i$. Επιπλέον, η σειρά με την οποία τα σημεία συναντώνται στο σύνορο του περιβλήματος (με φορά αντίθετη από αυτή των δεικτών του ρολογιού) μετά το $a$ αντιστοιχεί στην




**Σχήμα 2.7** Η κατασκευή της παραβολής για την ταξινόμηση των (5,3,2,9,8,10).

ταξινομημένη σειρά τους. Δηλαδή, μπορούμε να χρησιμοποιήσουμε έναν αλγόριθμο για το κυρτό περίβλημα για να ταξινομήσουμε σε $T(n) + O(n)$ χρόνο. Δεδομένου ότι η ταξινόμηση απαιτεί $\Omega(n \log n)$ χρόνο, τότε $T(n) = \Omega(n \log n)$.

Παρατηρήστε ότι είναι καθοριστικό για την παραπάνω αναγωγή, ο αλγόριθμος $B$ να επιστρέφει τις κορυφές του περιβλήματος με τη σειρά που συναντώνται κατά μήκος του συνόρου του. Δεν είναι καθόλου προφανές πώς μπορεί να επιτευχθεί μια παρόμοια αναγωγή από την ταξινόμηση στο πρόβλημα του υπολογισμού των ακραίων σημείων του περιβλήματος (τα οποία επιστρέφονται με τυχαία σειρά). Το ερώτημα εάν ο υπολογισμός των ακραίων σημείων είναι εφικτός σε λιγότερο από $n \log n$ χρόνο παρέμεινε αναπάντητο για αρκετά χρόνια, μέχρις ότου η εργασία αρκετών ερευνητών οδήγησε σε ένα $\Omega(n \log n)$ κάτω φράγμα και για αυτό το πρόβλημα.

## 2.7. Αυξητικός (Incremental) Αλγόριθμος

Ο αλγόριθμος αυτός είναι ένας από τους πιο απλούς αλγορίθμους που μπορεί να σκεφτεί κανείς, και έχει το πλεονέκτημα (όπως και ο επόμενος αλγόριθμος που θα αναλύσουμε) να επεκτείνεται στις τρεις (και παραπάνω) διαστάσεις[1]. Το γενικό πλάνο έχει ως εξής: πρόσθεσε τα σημεία, ένα-ένα, κατασκευάζοντας το αντίστοιχο κυρτό περίβλημα και χρησιμοποιώντας το περίβλημα αυτό κατά την προσθήκη του επόμενου σημείου. Άρα,

[1]Ο αλγόριθμος του Graham δεν φαίνεται να επεκτείνεται στις τρεις διαστάσεις καθώς δεν υπάρχει προφανές τριδιάστατο αντίστοιχο της διαδικασίας ταξινόμησης ως προς τη γωνία.

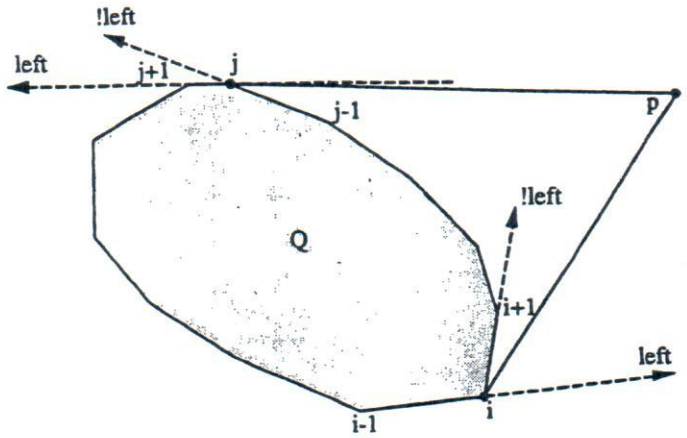
χρειάζεται να θεωρήσουμε μία μόνο περίπτωση: τη διαδικασία πρόσθεσης ενός σημείου στο υπάρχον κυρτό περίβλημα.

Έστω ότι τα σημεία που δίνονται είναι τα $p_1, p_2, \ldots, p_n$ και ας υποθέσουμε για λόγους απλότητας ότι τα σημεία είναι σε γενική θέση: καμία τριάδα σημείων δεν ορίζει συνευθειακά σημεία.

Το πρώτο κυρτό περίβλημα είναι το τρίγωνο με κορυφές $p_1$, $p_2$ και $p_3$. Έστω ότι με $H_k$ συμβολίζουμε το κυρτό περίβλημα των $k$ πρώτων σημείων. Θεωρούμε $Q = H_k$ και $p = p_{k+1}$. Τότε, για τον υπολογισμό του κυρτού περιβλήματος $H_{k+1}$ του $Q \cup p$ διακρίνουμε δύο περιπτώσεις ανάλογα με το εάν $p \in Q$ ή $p \notin Q$. Αν και υπάρχουν πολλοί τρόποι για να ελέγξουμε εάν το $p$ ανήκει στο $Q$, ίσως ο πιο ασφαλής βασίζεται στην ακόλουθη παρατήρηση: $p \in Q$ εάν και μόνο εάν για κάθε ακμή $e$ του $Q$, το $p$ είναι αριστερά του φορέα της $e$ ή ανήκει στην $e$ όταν το σύνορο του $Q$ διαγράφεται κατά την ανθωρολογιακή φορά. Προφανώς, ο έλεγχος αυτός απαιτεί χρόνο γραμμικό στο μέγεθος του $Q$.

1. $p \in Q$. Τότε το $p$ μπορεί να αγνοηθεί ακόμη και εάν ανήκει στο σύνορο του $Q$. Συνεπώς, $H_{k+1} = Q$.
2. $p \notin Q$. Σ' αυτήν την περίπτωση, πρέπει να υπολογίσουμε το κυρτό περίβλημα του $Q \cup p$. Ο υπολογισμός αυτός ανάγεται απλά στον προσδιορισμό των δύο εφαπτομένων από το $p$ στο $Q$ (Σχήμα 2.8) και την ενημέρωση του περιβλήματος αντίστοιχα. Με βάση την υπόθεση γενικής θέσης των σημείων, κάθε εφαπτομένη από το $p$ εφάπτεται στο $Q$ σε ακριβώς ένα σημείο, το οποίο πρέπει να βρούμε.

Πως μπορούμε να βρούμε τα σημεία επαφής των εφαπτομένων από το $p$ στο $Q$; Το Σχήμα 2.8 μας δείχνει ότι μπορούμε να χρησιμοποιήσουμε ελέγχους για το έαν το $p$ είναι αριστερά ή όχι από τις (κατευθυνόμενες) ακμές του $Q$. Πιο συγκεκριμένα, για




**Σχήμα 2.8**

**Αλγόριθμος:** Σημεία Επαφής
`for` $i = 1$ `to` μέγεθος($Q$) `do`
    `if` $xor$( το $p$ είναι αριστερά ή επάνω στο ευθ. τμήμα $\overline{p_{i-1}p_i}$,
        το $p$ είναι αριστερά ή επάνω στο ευθ. τμήμα $\overline{p_ip_{i+1}}$)
    `then` το $p_i$ είναι σημείο επαφής

**Αλγόριθμος 2.7** Υπολογισμός των Σημείων Επαφής.

το χαμηλότερο σημείο επαφής $p_i$, το $p$ είναι αριστερά από την $\overrightarrow{p_{i-1}p_i}$ αλλά δεξιά από την $\overrightarrow{p_ip_{i+1}}$. Για το ψηλότερο σημείο επαφής $p_j$, τα πράγματα είναι αντίστροφα: το $p$ είναι δεξιά από την $\overrightarrow{p_{j-1}p_j}$ αλλά αριστερά από την $\overrightarrow{p_jp_{j+1}}$. Και οι δύο περιπτώσεις μπορούν να συμπεριληφθούν σε μία, εάν χρησιμοποιήσουμε τη συνάρτηση exclusive-or (xor): μια κορυφή $p_i$ του $Q$ είναι σημείο επαφής εάν το $p$ βρίσκεται σε αντίθετες πλευρές ως προς τις δύο ακμές που πρόσκεινται στην κορυφή $p_i$ (Αλγόριθμος 2.7). Συνεπώς, τα δύο σημεία επαφής μπορούν να προσδιοριστούν με την ίδια σειρά ελέγχων που χρησιμοποιούνται για να προσδιοριστεί εάν $p \in Q$.

Αυτό που απομένει είναι να σχηματισθεί το καινούργιο κυρτό περίβλημα. Στην περίπτωση του Σχήματος 2.8, αυτό είναι

$$(p_1,\ p_2,\ \ldots,\ p_{i-1},\ p_i,\ p,\ p_j,\ p_{j+1},\ \ldots,\ p_n).$$

Εάν τα κυρτά περιβλήματα αποθηκεύονται σαν γραμμικές λίστες, η ενημέρωση αυτή μπορεί να γίνει εύκολα με διαγραφή της αλυσίδας $(p_{i+1},\ \ldots,\ p_{j-1})$ και εισαγωγή του $p$ στη θέση της.

Η ανάλυση της πολυπλοκότητας του αλγορίθμου είναι απλή: ο χρόνος που απαιτείται για την επεξεργασία της $k$-οστής κορυφής είναι ανάλογος του μεγέθους του κυρτού περιβλήματος $H_{k-1}$, δηλαδή, $O(k)$. Έτσι, ο συνολικός χρόνος στη χειρότερη περίπτωση θα είναι $O(3+4+\ldots+(n-1)) = O(n^2)$. Σημειώνεται ότι λίγο περισσότερη προσοχή στη σειρά επεξεργασίας των σημείων και στον τρόπο εύρεσης των σημείων επαφής μας επιτρέπει να πετύχουμε πολυπλοκότητα χρόνου $O(n \log n)$.

## 2.8. Διαίρει και Βασίλευε

Ο τελευταίος αλγόριθμος που θα περιγράψουμε επιτυγχάνει τη βέλτιστη $O(n \log n)$ πολυπλοκότητα χρόνου χρησιμοποιώντας τη μέθοδο *διαίρει-και-βασίλευε* (divide and conquer). Αυτή είναι η μόνη γνωστή τεχνική που επεκτείνεται στις τρεις διαστάσεις

όπου και επιτυγχάνει επίσης πολυπλοκότητα χρόνου $O(n \log n)$ που είναι ασυμπτωτικά βέλτιστη.

Η μέθοδος διαίρει-και-βασίλευε είναι μια γενική τεχνική επίλυσης προβλημάτων που έχει αποδειχθεί ιδιαίτερα αποτελεσματική στην πληροφορική. Βασίζεται στον χωρισμό του προβλήματος σε δύο προβλήματα με μέγεθος (σχεδόν) ίσο με το μισό του αρχικού, την αναδρομική επίλυση των δύο αυτών μικρότερων προβλημάτων και την "συγχώνευση" των λύσεων σε μια λύση του αρχικού προβλήματος. Όταν, λόγω της αναδρομής, το μεγεθος των υποπροβλημάτων (στα οποία έχει χωριστεί το αρχικό πρόβλημα) είναι αρκετά μικρό, τότε μπορούμε να τα επιλύσουμε σχετικά εύκολα. Συνεπώς, όλη η προσπάθεια συνίσταται στην αποτελεσματική εκτέλεση της συγχώνευσης των λύσεων. Εάν $T(n)$ είναι η πολυπλοκότητα χρόνου ενός αλγορίθμου που υλοποιεί τη μέθοδο διαίρει-και-βασίλευε" και η συγχώνευση των λύσεων απαιτεί γραμμικό χρόνο, τότε η $T(n)$ ικανοποιεί την αναδρομική σχέση: $T(n) = 2T(n/2) + O(n)$, η οποία έχει τη λύση $T(n) = O(n \log n)$.

Η μεθοδος διαίρει-και-βασίλευε εφαρμόστηκε στο πρόβλημα του κυρτού περιβλήματος από τους Preparata και Hong (1977) με στόχο την περιγραφή ενός αποτελεσματικού αλγορίθμου για κυρτά περιβλήματα στις τρεις διαστάσεις. Για λόγους απλότητας, υποθέτουμε ότι: (a) καμία τριάδα σημείων δεν ορίζει συνευθειακά σημεία και (b) κανένα ζεύγος σημείων δεν βρίσκεται στην ίδια κατακόρυφη γραμμή. Η γενική περιγραφή του αλγορίθμου τους έχει ως εξής:

1) Ταξινόμησε τα σημεία ως προς τη $x$-συντεταγμένη τους.
2) Χώρισε τα σημεία σε δύο σύνολα $A$ και $B$: το $A$ περιέχει τα αριστερά $\lceil n/2 \rceil$ σημεία και το $B$ τα δεξιά $\lfloor n/2 \rfloor$ σημεία.
3) Υπολόγισε αναδρομικά τα κυρτά περιβλήματα των $A$ και $B$.
4) Συνένωσε τα κυρτά περιβλήματα σε ένα κυρτό πολύγωνο, το κυρτό περίβλημα του $A \cup B$.

Η ταξινόμηση στο βήμα (1) εξασφαλίζει ότι τα δύο σύνολα $A$ και $B$ θα διαχωρίζονται από μια κατακόρυφη γραμμή (σύμφωνα και με την υπόθεσή μας ότι κανένα ζεύγος σημείων δεν ανήκει στην ίδια κατακόρυφη γραμμή), το οποίο συνεπάγεται ότι τα αντίστοιχα κυρτά περιβλήματα δεν τέμνονται. Αυτό απλοποιεί το βήμα (4) της συγχώνευσης. Τα βήματα (2), (3) και (4) επαναλαμβάνονται σε κάθε αναδρομική κλήση μέχρις ότου $n \leq 3$. Εάν $n = 3$, το αντίστοιχο κυρτό περίβλημα είναι ένα τρίγωνο.

Όλη η δουλειά σ' αυτόν τον αλγόριθμο εντοπίζεται στο βήμα της συγχώνευσης και δεν είναι εύκολο να εκτελέσουμε τη συγχώνευση σε γραμμικό χρόνο. Έστω $a$ και

**Αλγόριθμος:** Κάτω Εξωτερική Εφαπτομένη

$a \leftarrow$ δεξιότερο σημείο του $A$

$b \leftarrow$ αριστερότερο σημείο του $B$

`while` ($\eta$ $L = \overline{ab}$ δεν είναι κάτω εφαπτομένη σε $A$ και $B$) `do`

    `while` ($\eta$ $L$ δεν είναι κάτω εφαπτομένη στο $A$) `do`

        $a \leftarrow a - 1$

    `while` ($\eta$ $L$ δεν είναι κάτω εφαπτομένη στο $B$) `do`

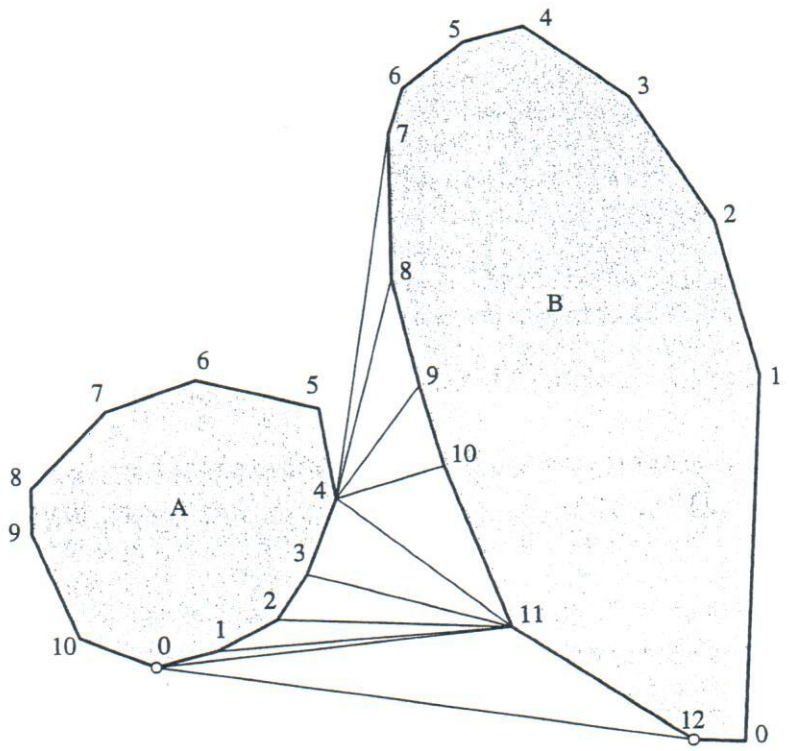
        $b \leftarrow b + 1$

**Αλγόριθμος 2.8** Προσδιορισμός Κάτω Εφαπτομένης.

$b$ δείκτες κορυφών στα κυρτά περιβλήματα των $A$ και $B$ αντίστοιχα, όπου οι δείκτες αυξάνουν καθώς κινούμαστε κατά την ανθωρολογιακή φορά κατά μήκος του συνόρου. Για απλοποίηση, θεωρούμε ότι οι εκφράσεις $a + 1$ και $a - 1$ υποδηλώνουν αντίστοιχα την επόμενη και την προηγούμενη κορυφή από την $a$ στο σύνορο του περιβλήματος του $A$. Ο στόχος είναι να βρούμε τις δύο εξωτερικές εφαπτομένες, τη μία που είναι κάτω και την άλλη που είναι επάνω και από τα δύο σύνολα $A$ και $B$. Εάν έχουμε αυτές τις εφαπτομένες, μπορούμε εύκολα να υπολογίσουμε το κυρτό περίβλημα του $A \cup B$ σε $O(n)$ χρόνο. Θα ασχοληθούμε παρακάτω μόνο με το πώς βρίσκουμε την κάτω εξωτερική εφαπτομένη. Η επάνω εξωτερική εφαπτομένη βρίσκεται με ανάλογο τρόπο.

Έστω ότι η κάτω εξωτερική εφαπτομένη είναι $L = \overline{ab}$. Η δυσκολία στον προσδιορισμό της $L$ έγκειται στο ότι κανένα από τα σημεία επαφής της δεν είναι γνωστό και έτσι η αναζήτηση θα πρέπει να προχωρά τόσο στο κυρτό περίβλημα του $A$ όσο και του $B$. (Υπενθυμίζουμε ότι στην περίπτωση του αυξητικού αλγορίθμου, τα πράγματα ήταν πιο απλά καθώς γνωρίζαμε το ένα άκρο της εφαπτομένης.) Σημειώστε ότι μια εξαντλητική αναζήτηση για όλα τα πιθανά $a$ και όλα τα πιθανά $b$ οδηγεί σε διαδικασία συγχώνευσης τετραγωνικού χρόνου που δεν είναι ικανοποιητική.

Η ιδέα των Preparata και Hong είναι να ορίσουν την $L$ αρχικά ως την ευθεία που συνδέει το δεξιότερο σημείο του $A$ με το αριστερότερο σημείο του $B$ και κατόπιν να την μετακινήσουν προς τα κάτω μετατοπίζοντας πότε το ένα της άκρο (στο ένα κυρτό περίβλημα) και πότε το άλλο (στο άλλο κυρτό περίβλημα). Η εναλλαγή στη μετατόπιση των άκρων γίνεται όταν μια "τοπική" κάτω εφαπτομένη έχει βρεθεί (Σχήμα 2.9). Ψευδοκώδικας δίνεται στον Αλγόριθμο 2.8. Σημειώστε ότι $a - 1$ δηλώνει κίνηση κατά τη φορά των δεικτών του ρολογιού, ενώ $b + 1$ αντίθετη με τη φορά των δεικτών του




**Σχήμα 2.9** Ο προσδιορισμός της κάτω εφαπτομένης: από το (4,7) στο (0,12).

ρολογιού. Και οι δύο όμως δηλώνουν κίνηση προς τα κάτω. Η ευθεία $L = \overline{ab}$ είναι μια κάτω εφαπτομένη στο $a$ εάν οι κορυφές με δείκτες $a-1$ και $a+1$ βρίσκονται επάνω από την $L$, ή ισοδύναμα, εάν και οι δύο κορυφές είναι αριστερά της $\overrightarrow{ab}$ ή ανήκουν στο τμήμα $ab$. Ένας ανάλογος ορισμός ισχύει για την εφαπτομένη στο $B$.

Ένα παράδειγμα φαίνεται στο Σχήμα 2.9. Αρχικά $L = (4, 7)$. Η $L$ εφάπτεται τόσο στο $A$ όσο και στο $B$, αλλά είναι κάτω εφαπτομένη μόνο στο $A$. Ο βρόχος για το $A$ δεν εκτελείται, αλλά ο βρόχος για το $B$ αυξάνει το $b$ σε 11, οπότε η $L = (4, 11)$ είναι κάτω εφαπτομένη στο $B$. Τότε όμως η $L$ δεν είναι πλέον κάτω εφαπτομένη στο $A$ και ο βρόχος για το $A$ μειώνει το $a$ σε 0. Τώρα, η $L = (0, 11)$ είναι κάτω εφαπτομένη στο $A$. Όχι όμως και στο $B$ και το $b$ αυξάνεται σε 12. Τώρα, η $L = (0, 12)$ είναι κάτω εφαπτομένη και στο $A$ και στο $B$ και ο αλγόριθμος τερματίζει.

Ούτε η πολυπλοκότητα χρόνου ούτε η ορθότητα του αλγορίθμου είναι προφανείς. Ωστόσο, αποδεικνύεται ότι ο αλγόριθμος υπολογίζει σωστά την κάτω εφαπτομένη και ότι τα $a$ και $b$ κινούνται πάντα προς τις τελικές τους τιμές χωρίς να τις υπερβούν. Συνεπώς, οι βρόχοι απαιτούν γραμμικό χρόνο, καθώς είτε μειώνουν το $a$ είτε αυξάνουν το $b$, και άρα το πλήθος επαναλήψεων φράσσεται από το συνολικό πλήθος κορυφών των $A$ και $B$. Δηλαδή, η συγχώνευση γίνεται σε γραμμικό χρόνο, το οποίο συνεπάγεται ότι ο συνολικός αλγόριθμος έχει πολυπλοκότητα χρόνου $O(n \log n)$.