Konvexe Hülle

Hierbei handelt es sich um ein klassisches Problem aus der Algorithmischen Geometrie, dem Teilgebiet der Informatik, in dem man für geometrische Probleme effiziente Algorithmen bestimmt.
Gegeben seien n Punkte im d-dimensionalen Raum. Gesucht ist die kleinste konvexe Menge, die alle Punkte enthält. Hierbei heißt eine Menge *konvex*, wenn mit je zwei Punkten der Menge auch die Verbindungsstrecke ganz in der Menge liegt. Eine konvexe Menge enthält also weder "Inseln" noch "Halbinseln" (Abb. 1).

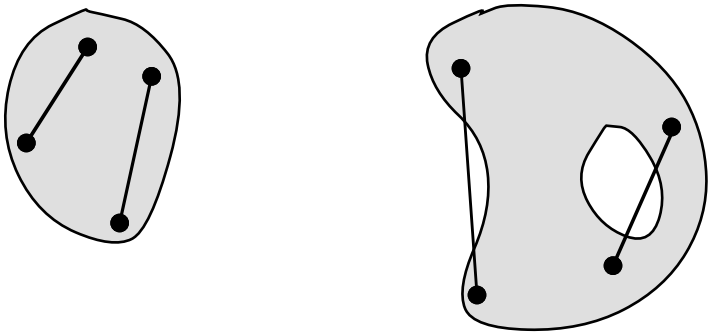

Abb. 1: Konvexe und nicht-konvexe Menge

In der Ebene kann man sich das Problem der konvexen Hülle wie folgt veranschaulichen: Gegeben sei eine Menge von Wasserstellen. Man zäune diese Wasserstellen mit einem möglichst kurzen Zaun ein. Die konvexe Hülle ist dann ein konvexes Vieleck (Abb. 2).

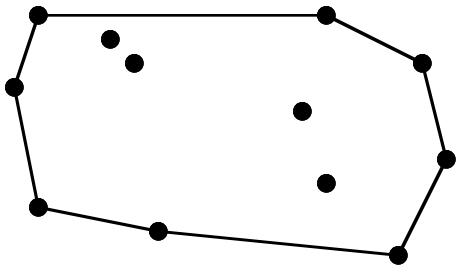

Abb. 2: Konvexe Hülle in der Ebene

In der Ebene kann man das Problem der konvexen Hülle mit einem Algorithmus lösen, der auf dem Divide-and-Conquer-Verfahren basiert. Man teilt die Punktemenge $P=\{p_1,...,p_n\}$ durch eine senkrechte Linie in zwei Hälften. Hierzu muß man die Punkte vorab entsprechend ihrer x-Koordinate aufsteigend sortieren. Für jede der beiden Hälften berechnet man rekursiv die konvexe Hülle. Anschließend verbindet man die beiden konvexen Hüllen miteinander und erhält die gesuchte konvexe Hülle für alle Punkte (Abb. 3).

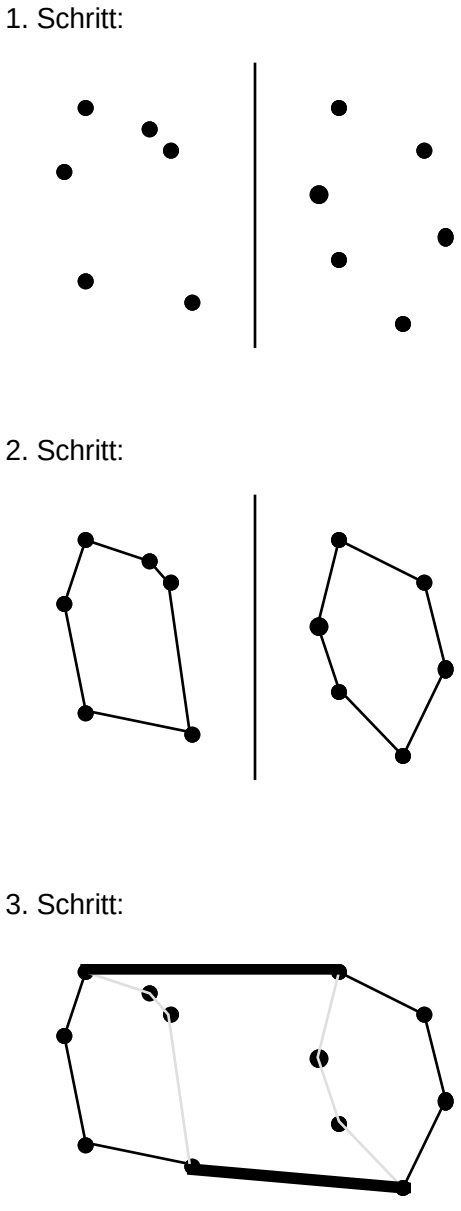

Abb. 3: Rekursive Berechnung der konvexen Hülle

Der Algorithmus:

```
var  P: "nach der x-Koordinate aufsteigend
         sortierte Folge von Punkten (p1,...,pn)";
     k: integer;
if |P|≤3 then
     "Berechne die konvexe Hülle H von P
     auf direkte Weise"
else
     k:=| |P|/2 |;
     P1:= (p1,...,pk); P2:= (pk+1,...,pn);
     "Wende das Verfahren rekursiv auf P1 und P2 an
          und erhalte zwei konvexe Hülle H1 und H2";
     "Verbinde H1 und H2 und erhalte die gesuchte
          konvexe Hülle H von P".
```

(|x| bezeichne die größte ganze Zahl kleiner oder gleich x).

Der problematische Schritt in diesem Algorithmus ist das Verbinden zweier konvexer Hüllen zu einer einzigen. Hierzu muß man die beiden Verbindungslinien (*Brücken*) b_1 und b_2 finden (Abb. 4).

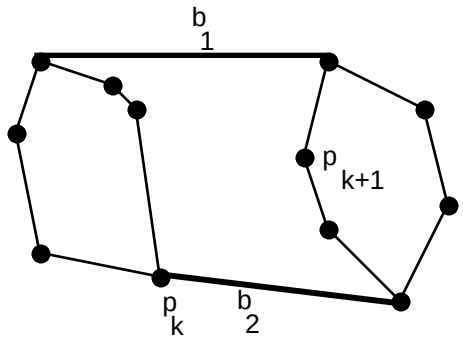

Abb. 4: Verbinden zweier konvexer Hüllen

b_1 und b_2 gewinnt man folgendermaßen: Man startet mit der Kante s zwischen dem rechtesten Punkt der linken konvexen Hülle und dem linkesten Punkt der rechten konvexen Hülle. Dies ist wegen der Sortierung der Punkte gerade die Strecke zwischen p_k und p_{k+1}. Um die obere Brücke b_1 zu finden verschiebt man schrittweise den

Endpunkt von s in der linken Hülle gegen den Uhrzeigersinn und den Endpunkt von s in der rechten Hülle mit dem Uhrzeigersinn nach folgendem Algorithmus:

"Seien u und v die Endpunkte von s";
$u:=p_k$; $v:=p_{k+1}$;
while "der auf u gegen den Uhrzeigersinn in der
linken Hülle folgende Punkt u' oder der
auf v im Uhrzeigersinn in der rechten Hülle
folgende Punkt v' liegen oberhalb von s" do
if "die while-Bedingung trifft auf u zu" then u:=u'
else v:=v'.

Die untere Brücke findet man auf analoge Weise.

Beispiel: Abb. 5 zeigt die vier Schritte, die zur Bestimmung der oberen Brücke erforderlich sind, und die jeweiligen Kanten s.

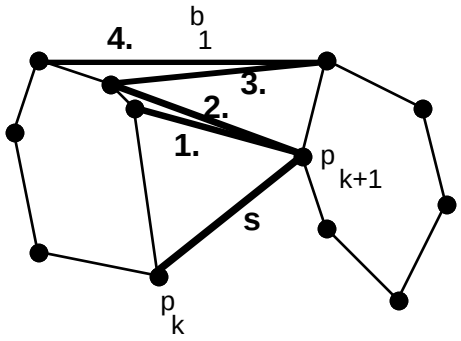

Abb. 5: Bestimmung der oberen Brücke

Bei geeigneter Implementierung benötigt der obige Algorithmus zur Bestimmung der konvexen Hülle von n Punkten $O(n \cdot \log_2 n)$ Schritte.

Man kann zeigen, daß diese Schranke optimal ist in dem Sinne, daß es keinen Algorithmus gibt, der das Problem wesentlich schneller löst. Hierzu reduziert man das Problem der konvexen Hülle auf die Sortierung von Zahlen und beweist, daß ein schnellerer Algorithmus für das Problem der konvexen Hülle einen schnelleren Algorithmus für das Sortieren liefern würde, was nicht möglich ist.
Sei dazu $x_1,...,x_n \in IR$ eine Folge von Zahlen. Diese Folge ist unter Verwendung eines Algorithmus für die konvexe Hülle als Prozedur zu sortieren. Gelingt dies, so hat man das Problem des Sortierens auf das Problem der konvexen Hülle reduziert.

Zur Reduktion gehe man von den Zahlen $x_1,\ldots,x_n$ über zu den n Punkten $(x_1,x_1^2),\ldots,(x_n,x_n^2)$. Für diese Punkte bestimmt man mit dem bekannten Algorithmus eine konvexe Hülle. Offenbar gehört jeder der n Punkte (x_i,x_i^2) zur konvexen Hülle, wie man sich anhand von Abb. 6 klar macht. Der Algorithmus für die konvexe Hülle liefert bei geeigneter Implementierung die Punkte, die zur konvexen Hülle gehören, im Uhrzeigersinn beginnend beim Punkt mit der kleinsten x-Koordinate. Diese Reihenfolge liefert zugleich eine aufsteigende Sortierung der Zahlen $x_1,\ldots,x_n$, wenn man die zweite Koordinate streicht.
Folglich kann man mit jedem Algorithmus zur Bestimmung der konvexen Hülle auch Zahlen sortieren, das Problem der konvexen Hülle kann daher nicht leichter sein als das Sortierproblem. Vom Sortierproblem weiß man bereits, daß man hierzu mindestens O(nlogn) Vergleiche im schlimmsten Fall benötigt. Dasselbe muß dann auch für das Problem der konvexen Hülle gelten.

Im allgemeinen Falle kann man die konvexe Hülle von n Punkten im d-dimensionalen Raum in $O(n\cdot\log_2 n+n^{\lfloor(d+1)/2\rfloor})$ Schritten bestimmen.

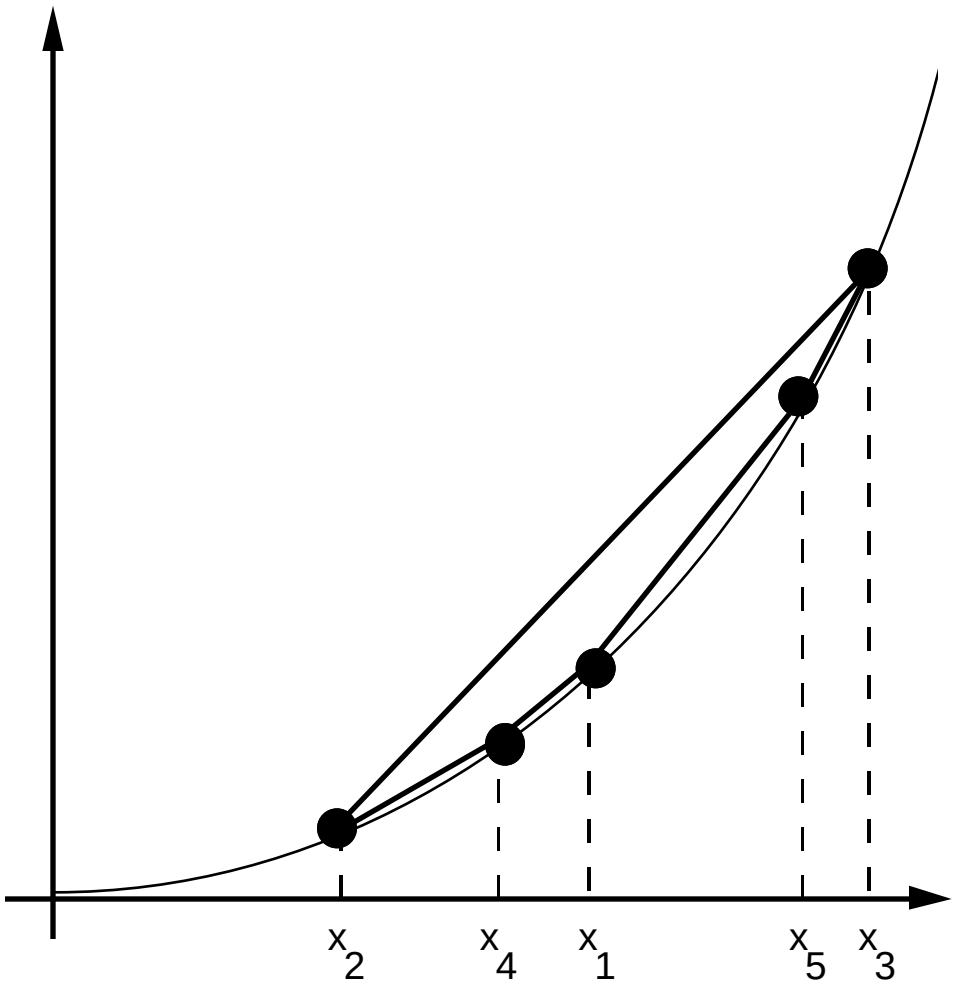

Abb. 6: Konvexe Hülle für (x_i,x_i^2)

Literatur

H. Edelsbrunner: Algorithms in Combinatorial Geometry, volume 10 of EATCS Monographs on Theoretical Computer Science. Springer-Verlag 1987.