Algorithmische Geometrie, SoSe 2005
Skriptmitschrift vom 27. April 2005

Britta Weber
Steffen Galan

Polygraph tests are 20th-century witchcraft.

Sam Ervin

Konvexe Hüllen

Gegeben ist eine endliche Punktmenge S, berechne die konvexe Hülle $\mathrm{CH}(S)$, angegeben durch in Reihenfolge sortierte Kanten auf dem Rand (z.B. im Uhrzeigersinn).

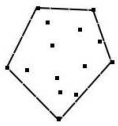

Abbildung 1: Beispiel einer konvexen Hülle

Definition. *Ein Punkt p einer konvexen Menge K heißt* Extrempunkt von K*, falls eine Gerade g existiert mit $g \cap K = \{p\}$.*

Satz. *Der Rand der konvexen Hülle einer endlichen Punktmenge S ist ein konvexes Polygon, dessen Ecken die Extrempunkte von $CH(S)$ sind.*

Beweis. *Siehe Übung! ("... nicht ganz so leicht, wenn man es in allen Einzelheiten zeigen möchte." [Herr Alt]).*

□

Satz. *Gegeben sei ein einfaches Polygon mit den Ecken $p_1, \ldots, p_n$ in dieser Reihenfolge. Dann kann $CH(S)$, wobei $S = \{p_1, \ldots, p_n\}$, in $\mathcal{O}(n)$ Zeit bestimmt werden.*

Beweis. *p_l, p_r seien Punkte mit minimaler/maximaler x-Koordinate. Dann teilt $\overline{p_l p_r}$ die konvexe Hülle $CH(S)$ in 2 Hälften.*

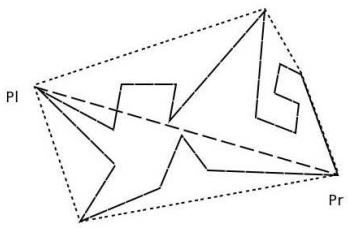

Abbildung 2: $\overline{p_l p_r}$ teilt $\mathrm{CH}(S)$

Es reicht, einen Algorithmus anzugeben, welcher den oberen Teil der konvexen Hülle berechnet, der untere Teil geht analog. Wir nummerieren um, so daß $p_l = p_1, p_2, \ldots, p_n = p_r$ in entsprechender Reihenfolge. Die Datenstruktur ist ein Stack, welcher die bisher gefundenen Kandidaten $q_0, q_1, \ldots, q_t$ (letztes Element oben auf) für die Ecken der konvexen Hülle enthält.

```
1  Initialisierung:
2  q_0 := p_r, q_1 := p_l;
3  s:=2;
4  while (q_0, q_1, p_s) Linkskurve do s:= s+1;
5  push p_s;
6  while s<n do
7          repeat s:= s+1 until p_s links von oder auf q_t q_0 oder p_s links von q_{t-1} q_t
8          while (q_{t-1}, q_t, p_s) keine Rechtskurve do pop;
9          push p_s;
```

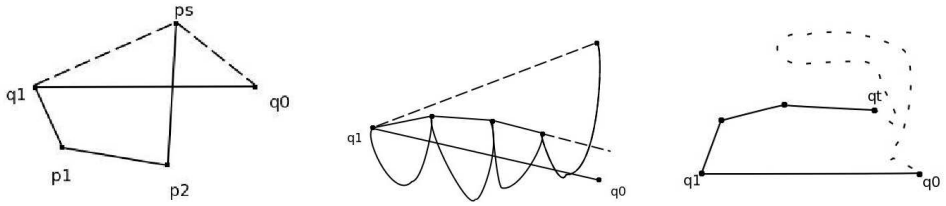

Abbildung 3: nach ersten **while**-Schleifenende und nach einigen Iterationen

Korrektheit: *Wir zeigen folgende Invarianten in der zweiten* ***while****-Schleife:* $q_0, \ldots, q_t$ *ist Teilfolge von* $p_1, \ldots, p_n$ *mit*

1. $q_0 = p_r, q_1 = p_l (= p_1), t \geq 2, q_t = p_s$
2. $q_0, q_1, \ldots, q_t$ *ist ein konvexes Polygon*
3. *obere konvexe Hülle von* S *ist Teilfolge von* $q_0, \ldots, q_t, p_{s+1}, \ldots, p_n$

Dies zeigt man durch Induktion über s*, woran man sich üben darf.*

Laufzeit: *Jede Ecke von* P *wird bei der Initialisierung oder in Zeile 7 und 9 durchlaufen, in Zeile 8 möglicherweise ein zweites Mal. Für die Operationen brauchen wir jeweils Konstante Zeit, also* $\mathcal{O}(n)$*.*

□

Satz. *Gegeben sei* $S \subset \mathbb{R}^2$*,* $|S| = n$*, dann kann* $CH(S)$ *in Zeit* $\mathcal{O}(n \log n)$ *berechnet werden.*

Beweis. Algorithmus:

- *sortiere* S *lexikographisch nach* x*- und* y*-Koordinaten, wir erhalten Folge* $p_1, \ldots, p_n$
- $p_1, \ldots, p_n$ *bilden einen einfachen Polygonzug, wende Algorithmus von vorher an.*

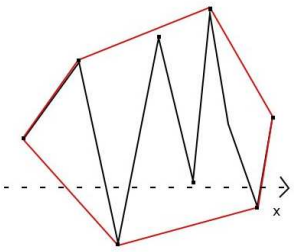

Abbildung 4: Beispiel zum Algorithmus

Das Sortieren geht in $\mathcal{O}(n \log n)$*, der vorherige Algorithmus braucht* $\mathcal{O}(n)$*, also brauchen wir insgesamt* $\mathcal{O}(n \log n)$*.*

□

Satz (untere Schranke). *Die Konstruktion der konvexen Hülle einer Menge von* n *Punkten in der Ebene erfordert* $\Omega(n \log n)$ *Zeit.*

[Modell: algebraischer Entscheidungsbaum]

(Vergleichen und arithmetische Operationen $+, \cdot, /, -$). Auch damit braucht man $\Omega(n \log n)$ Zeit zum Sortieren reeller Zahlen.

Wir zeigen: Wenn man die konvexe Hülle berechnen kann, so kann man auch mit $\mathcal{O}(n)$ zusätzlichen Aufwand n reelle Zahlen sortieren.

Gegeben sei die Folge $x_1, \ldots, x_n$ reeller Zahlen. Betrachte die Punkte $S = \{(x_i, x_i^2) \mid i = 1, \ldots, n\}$.

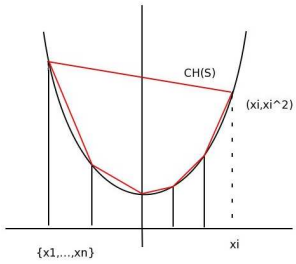

Abbildung 5: Beispiel zum Algorithmus

Berechne CH(S). Seien $p_1 = (x_{i_1}, x_{i_1}^2), \ldots, p_n = (x_{i_n}, x_{i_n}^2)$ die vom Algorithmus gelieferten Eckpunkte der konvexen Hülle in Reihenfolge, die i_k liefern dann eine Umsortierung der vorherigen x_i, so daß die Folge $x_{i_1}, \ldots, x_{i_n}$ dann die Form $x_{i_1} \geq x_{i_2} \ldots \geq x_{i_k} < x_{i_{k+1}} \geq \ldots \geq x_{i_n}$ hat. Die Stelle i_k finden geht in $\mathcal{O}(n)$, dann sind die $x_{i_{k+1}}, \ldots, x_{i_n}, x_{i_1}, \ldots, x_{i_k}$ eine absteigend sortierte Folge.

Schauen wir uns Ideen anderer, schneller konvexer-Hüllen-Algorithmen an. Im folgendem sei $S = \{p_1, \ldots, p_n\}$.

Graham-Scan

1. Bestimme Schwerpunkt $p = \frac{1}{n} \sum_{i=1}^{n} p_i$ und Strahlen von p zu allen Punkten in S in linearer Zeit. Wir sehen p als Ursprung, rechne mit Polarkoordinaten der Punkte.

2. Sortiere Punkte bezüglich der Polarkoordinaten (lexikographisch: 1. Winkel, 2. Abstand). Es sei $q_1, \ldots, q_n$ sei diese Ordnung (gegen Uhrzeigersinn). Brauchen $\mathcal{O}(n \log n)$.

3. Setze $r_1 := q_1, r_2 := q_2$. Durchlaufe die Folge, nimm q_i in CH(S) auf, solange r_{j-1}, r_j, q_i eine Linkskurve bilden. Falls dies nicht mehr der Fall ist, entferne $r_j, r_{j-1}, \ldots$, bis wieder eine Linkskurve entsteht. Brauchen $\mathcal{O}(n)$.