Algorithmische Geometrie

Gesellschaft für Informatik - Fachgruppe 0.1.2 Algorithmische Geometrie

Was ist Algorithmische Geometrie?

Bereits im Altertum haben sich Wissenschaftler wie Pythagoras und Euklid mit geometrischen Problemen beschäftigt. Ihr Interesse galt der Entdeckung geometrischer Sachverhalte und deren Beweis. Sie operierten ausschließlich mit geometrischen Figuren (Punkten, Geraden, Kreisen etc.). Erst die Einführung von Koordinaten durch Descartes machte es möglich, geometrische Objekte durch Zahlen zu beschreiben.

Heute gibt es in der Geometrie verschiedene Richtungen, deren unterschiedliche Ziele man vielleicht an folgendem Beispiel verdeutlichen kann. Denken wir uns eine Fläche im Raum, etwa das Paraboloid, das durch Rotation einer Parabel um seine Symmetrieachse entsteht. In der *Differentialgeometrie* werden mit analytischen Methoden Eigenschaften wie die Krümmung der Fläche an einem Punkt definiert und untersucht.

Die *Algebraische Geometrie* faßt das Paraboloid als *Nullstellenmenge* des Polynoms $p(X,Y,Z) = X^2 + Y^2 - Z$ auf; hier würde man zum Beispiel den Durchschnitt mit einer anderen algebraischen Menge, etwa dem senkrechten Zylinder $(X - x_0^2) + (Y - y_0^2) - r^2$ betrachten und sich fragen, durch welche Gleichungen der Durchschnitt beschrieben wird.

Aus der Mathematik sind uns solche Fragestellungen von einfachen Beispielen vertraut: In der Analysis werden Tangenten an Kurven betrachtet, und in der linearen Algebra immerhin Durchschnitte von Objekten, die sich durch *lineare Gleichungen* beschreiben lassen.

Die *Algorithmische Geometrie*, verfolgt andere Ziele. Ihre Aufgaben bestehen in

Existenz v. effiz. Lösungsverfahren + konkrete Angabe der Algorithmen.

- der Entwicklung von *effizienten* und *praktikablen* Algorithmen zur Lösung geometrischer Probleme, und in
  Angabe der unteren Schranke + Konstr. v. Alg., der diese Schranke nicht überschreitet.
- der Bestimmung der *algorithmischen Komplexität* geometrischer Probleme.

Die untersuchten Probleme haben meistens sehr *reale Anwendungshintergründe*. Bei der *Bahnplanung für Roboter* geht es darum, eine Bewegung von einer Anfangskonfiguration in eine Endkonfiguration zu planen, die Kollisionen mit der Umgebung vermeidet und außerdem möglichst effizient ist. Wer je eine Leiter durch verwinkelte Korridore getragen hat, kann sich ein Bild von der Schwierigkeit dieser Aufgabe machen. Sie wächst noch, wenn der Roboter seine Umgebung noch gar nicht kennt, sondern sie während der Ausführung erkunden muß.

Beim *computer aided geometric design (CAGD)* kommt es unter anderem darauf an, Durchschnitt und Vereinigung von dreidimensionalen Körpern schnell zu berechnen. Oder es sollen interpolierende Flächen durch vorgegebene Stützpunkte konstruiert werden.

Bei der Arbeit mit *geographischen Daten*, die in der Regel in Datenbanken gespeichert sind, müssen

Zusammenfassung von Algorithmische Geometrie.

Kapitel 0: Vorbemerkungen:

Inhalt der Vorlesung: Algorithmen und Datenstrukturen zur Lösung geometrischer Probleme.

Beispiele: Robotik / Bewegungsplanung, Computergrafik, CAD, VLSI

Typische Probleme:
- konvexe Hülle
- Zerlegung in einfache konvexe Teile (zB. Triangulierung).
- Schnittprobleme (zB. Segmente, Polygone)
- Suchprobleme (Memberanfrage in Pktmenge, Nearest-Neighbor, Range-Abfragen).
- Bewegungsplanung (Roboter)
- Hidden Line / Surface (Elimination).

Grundlegende Ansätze bzw. Vorgehensweisen:
- Divide & Conquer
- Plane Sweep.
- Hierarchische Darstellung
- Dualität
- Randomisierte Algorithmen.
- Rundungsfehler u. degenerierte Eingaben.

Kapitel I: Konvexe Hüllen.

1.1 Konstruktion der konvexen Hülle einer Pktmenge im $\mathbb{R}^2$ – Grundlagen.

1.1.1. Def: Sei $S \subset \mathbb{R}^2$ Pktmenge.

S heißt konvex $:\Leftrightarrow \forall\, p,q \in S$ gilt: $\overline{pq} \subset S$.

konvexe Hülle einer Menge $S \subset \mathbb{R}^2$ ($CH(S)$) ist die kleinste konvexe Menge, die S enthält.

Wir betrachten nun endliche Pktmengen, dh. $|S| = n \in \mathbb{N}$.

Anm: Der Rand von $CH(S)$ ist ein konvexes Polygon mit Ecken aus S.
Bew. Übung.

Konvexe Hülle-Problem für endl. Menge $S \subset \mathbb{R}^2$:

Berechne die Folge der Ecken von $CH(S)$ gg. den Uhrzeigersinn (pos. orientiert).
$\rightarrow q_1, \dots q_\ell \in S$. (Anordnung definiert Kanten der Fläche).

Degenerierte Fälle: $\ell = 2$ (alle Pkte sind colinear)
$\ell = 1$ (alle Pkte in S sind gleich).

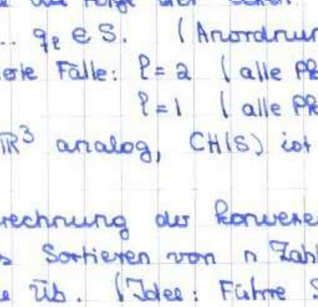
Problem im $\mathbb{R}^3$ analog, $CH(S)$ ist konvexes Polyeder mit Ecken aus S.

1.1.2. Satz: Berechnung der konvexen Hülle von n Pktn. im $\mathbb{R}^2$ ist mind. so schwer wie das Sortieren von n Zahlen.

Bew. siehe Üb. (Idee: Führe Sortieren von $x_1 .. x_n$ zurück auf $CH(\{p_1 .. p_n\})$.).

1.1.3. Korollar: Im Allgemeinen braucht die Berechnung von $CH(S)$ Zeit $\Omega(n \log n)$.

1.1.4. Bsp:



$S = \{1 .. 12\}$, $CH(S) = 8,7,3,1,4,9$
(Jeder zyklische Shift)

1.1.5. Wdh: Lexikografische Ordnung (siehe Strings) für Pkte im $\mathbb{R}^2$:

Lexikogr. Sortierung nach Kartesischen (x,y)-Koordinaten, dh. für zwei Pkte in der Ebene, also $p, q \in \mathbb{R}^2$ gilt: $p < q :\Leftrightarrow p_x < q_x \vee (p_x = q_x \wedge p_y < q_y)$

Dh. Sortierung von links nach rechts bzw. von unten nach oben (bei gleicher x-Koordinate).

Zusammenfassung von Algorithmische Geometrie.

Kapitel 0: Vorbemerkungen:

Inhalt der Vorlesung: Algorithmen und Datenstrukturen zur Lösung geometrischer Probleme.

Beispiele: Robotik / Bewegungsplanung, Computergrafik, CAD, VLSI

Typische Probleme:
- konvexe Hülle
- Zerlegung in einfache konvexe Teile (zB. Triangulierung).
- Schnittprobleme (zB. Segmente, Polygone)
- Suchprobleme (Memberanfrage in Pktmenge, Nearest-Neighbor, Range-Abfragen).
- Bewegungsplanung (Roboter)
- Hidden Line / Surface (Elimination).

Grundlegende Ansätze Bzw. Vorgehensweisen:
- Devide & Conquer
- Plane Sweep.
- Hierarchische Darstellung
- Dualität
- Randomisierte Algorithmen.
- Rundungsfehler u. degenerierte Eingaben.

Kapitel I: Konvexe Hüllen.

1.1 Konstruktion der konvexen Hülle einer Pktmenge im $\mathbb{R}^2$ – Grundlagen.

1.1.1. Def: Sei $S \subset \mathbb{R}^2$ Pktmenge.

S heißt konvex : $\Leftrightarrow \forall\, p, q \in S$ gilt : $\overline{pq} \subset S$.

konvexe Hülle einer Menge $S \subset \mathbb{R}^2$ ($CH(S)$) ist die kleinste konvexe Menge, die S enthält.

Wir betrachten nun endliche Pktmengen, dh. $|S| = n \in \mathbb{N}$.

Anm: Der Rand von CH(S) ist ein konvexes Polygon mit Ecken aus S.
Bew. Übung.

Konvexe Hülle-Problem für endl. Menge $S \subset \mathbb{R}^2$:

Berechne die Folge der Ecken von CH(S) gg. den Uhrzeigersinn (pos. orientiert).
$\rightarrow q_1, \dots q_\ell \in S$. (Anordnung definiert Kanten der Fläche).

Degenerierte Fälle: $\ell = 2$ (alle Pkte sind colinear)
$\ell = 1$ (alle Pkte in S sind gleich).

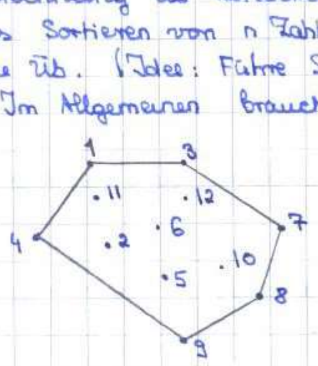
Problem im $\mathbb{R}^3$ analog, CH(S) ist konvexes Polyeder mit Ecken aus S.

1.1.2. Satz: Berechnung der konvexen Hülle von n Pktn. im $\mathbb{R}^2$ ist mind. so schwer wie das Sortieren von n Zahlen.

Bew. siehe Üb. (Idee: Führe Sortieren von $x_1 .. x_n$ zurück auf $CH(\{p_1 .. p_n\})$.).

1.1.3. Korollar: Im Allgemeinen braucht die Berechnung von CH(S) Zeit $\Omega(n \log n)$.

1.1.4. Bsp:



$S = \{1..12\}$, $CH(S) = 8,7,3,1,4,9$
(Jeder zyklische Shift)

1.1.5. Wdh: Lexikografische Ordnung (siehe Strings) für Pkte im $\mathbb{R}^2$:

Lexikogr. Sortierung nach Kartesischen (x,y)-Koordinaten, dh. für zwei Pkte in der Ebene, also $p, q \in \mathbb{R}^2$ gilt: $p < q :\Leftrightarrow p_x < q_x \vee (p_x = q_x \wedge p_y < q_y)$

Dh. Sortierung von links nach rechts Bzw. von unten nach oben (Bei gleicher x-Koordinate).

1.2 Algorithmus I: Gift Wrapping.

Intuitiv: Pktmenge wird mit Geschenkpapier eingewickelt.

Zunächst: $S \subset \mathbb{R}^2$. Voraussetzung: $|S| < \infty$

Ges: $CH(S) = p_1 \dots p_n$.

1.2.1. Idee: 1). Startpkt p_1 : Pkt mit der kleinsten y-Koordinate.

(Falls mehrere wähle den linkesten. D.h. Minimum in p_n - lexikogr. Ordnung.

(Übung: p_1 ist Ecke der konvexen Hülle).

2) Wie findet man p_2? (Nachfolger von p_1 auf konv. Hülle).

Betrachte horizontalen Strahl ℓ durch p_1 (nach rechts). Drehen den Strahl gg. den Uhrzeigersinn bis wir einen weiteren Pkt von S treffen. (Ann. $|S| > 1$).

So erhalten wir p_2.

Genauer: $\forall q \in S \setminus \{p_1\}$ sei α_q der Winkel zwischen ℓ und $\overline{p_1 q}$.

Wähle p_2 so, dass α_{p_2} minimal. Falls mehrere Pkte mit minimalen Winkel, wähle den Pkt mit maximaler Entfernung zu p_1.

$\rightsquigarrow$ lineare Suche in S nach Minimum bzgl. oben definierter Ordnung.

3). Setze Algor. bei p_2 fort, d.h. ℓ wird nun der Strahl, der

- in p_2 beginnt.
- in Richtung $p_1 p_2$ zeigt

$\rightsquigarrow$ Suche Minimum in $S \setminus \{p_1, p_2\}$.

4) Wiederhole bis p_1 wieder erreicht wird.

1.2.2 Korrektheit: Übung.

1.2.3. Laufzeit: $CH(S) = p_1 \dots p_h$, $|S| = n$.

p_1: lineare Suche in S kostet $O(n)$ (lin. Suche nach Min. bzgl. p_y)

$p_2 \dots p_h$: lin. Suche in S kostet $O(n)$ (lin. Suche nach Winkelordnung).

$\Rightarrow$ Gesamtlaufzeit: $O(h \cdot n)$.

1.2.4. Satz: Sei S eine Menge von n Pkten im $\mathbb{R}^2$ und h die Zahl der Ecken der konvexen Hülle CH(S). Dann kann CH(S) in Zeit $O(h \cdot n)$ berechnet werden. Bew. siehe oben.

1.2.5. Bem: $h \in \{1, \dots, n\}$

Worst Case: $h = n$ (zB. alle Pkte aus S liegen auf einem gemeinsamen Kreis, das Innere ist leer). $\Rightarrow$ Laufzeit: $O(n^2)$.

Bester Fall: $h = \text{const.} \Rightarrow$ Laufzeit: $O(h \cdot n) = O(n)$.

1.2.6. Details der Implementierung:

1). Wie findet man Pkt q so, dass α_q minimal ist?

$\rightsquigarrow$ Wir müssen Winkel vergleichen.

```
q ← undefiniert
αq ← ∞
forall p ∈ S \ {p1} do
    if αp < αq then
        q ← p
        αq ← αp
    fi
od
```

Im Allg. 3 Pkte p, q, r $\rightarrow$ vergleiche Winkel

Naiv: Berechne Winkel aus Koordinaten der Pkte mit trigonometrischen Fktnen.

$\Rightarrow$ Nachteile:
- langsam
- Präzisionsprobleme
- Robustheitsproblem (Definitionsbereich der trigonometrischen Fktnen ist eingeschränkt).

$\Rightarrow$ Besser: Andere Betrachtungsweise:

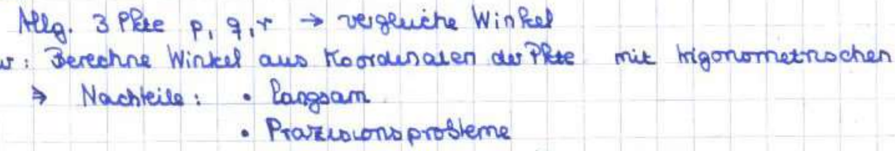
Dazu betrachte Dreieck p, q, r

Drei mögliche Orientierungen:

a) positiv orientiert (left-turn) $\Rightarrow \alpha_q < \alpha_r$

b) Orientierung 0 (collinear) $\Rightarrow \alpha_q = \alpha_r$

c) negativ orientiert (right-turn) $\Rightarrow \alpha_q > \alpha_r$



Allgemein: Seien $p=(p_x,p_y)$, $q=(q_x,q_y)$ und $r=(r_x,r_y)$ drei Pkte im $\mathbb{R}^2$.
Sei $\vec{p}$ der Vektor, der vom Nullpkt ausgeht und in Richtung p zeigt. Und seien $\vec{q}$ und $\vec{r}$ Vektoren, die von p ausgehen und in Richtung q bzw. r zeigen.
Betrachte nun das Parallelogramm, welches durch die Vektoren $\vec{q}$ und $\vec{r}$ aufgespannt wird. Sei V(P) = die Fläche des Parallelogramms.
Beh: $V(P) = |\det A|$, wobei $A = \begin{pmatrix} q_x-p_x & q_y-p_y \\ r_x-p_x & r_y-p_y \end{pmatrix}$

Bem: diese Beh. überträgt sich auch auf $\mathbb{R}^n$.
Man betrachte hierbei $u_1 \dots u_n \in \mathbb{R}^n$ und $P = \{u_1 a_1 + \dots + u_n a_n : 0 \le a_i \le 1, i \in \{1..n\}\}$.
$\Rightarrow V(P) = |\det A|$, A entsprechend.

Wir betrachten Δpqr:

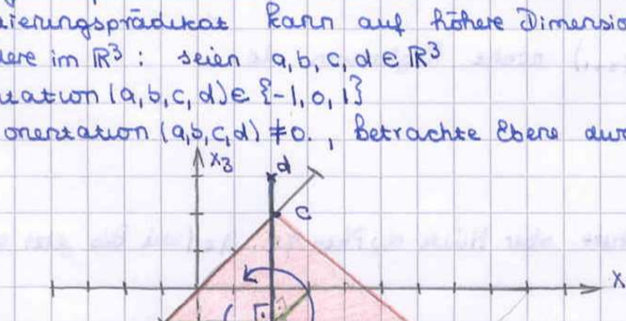
1.2.6.1 Satz: $p,q,r \in \mathbb{R}^2$, $A = \begin{pmatrix} q_x-p_x & q_y-p_y \\ r_x-p_x & r_y-p_y \end{pmatrix}$.
$\Rightarrow$ (i) $|\det A| = 2 \cdot$ Fläche von Δpqr.
(ii) sign(det A) gibt die Orientierung an:
-1: neg. orientiert (right turn)
0: collinear.
+1: pos. orientiert (left turn).

Bew. siehe Lernordner.

1.2.6.2 Def: Wir definieren das geometrische Prädikat:
$$\text{orientation}(p,q,r) = \operatorname{sign}\begin{vmatrix} p_x & p_y & 1 \\ q_x & q_y & 1 \\ r_x & r_y & 1 \end{vmatrix} \quad \left(= \operatorname{sign}\left((q_x-p_x)(r_y-p_y)-(r_x-p_x)(q_y-p_y)\right)\right).$$
Dh. orientation(p,q,r) und damit der Test $d_r < d_q$ kann mit zwei Multiplikationen und fünf Subtraktionen berechnet werden.

2) Falls $d_r = d_q$ (dh. orientation(p,q,r) = 0), müssen wir herausfinden, welcher der Pkte q oder r weiter von p entfernt liegt.
Naiv: Berechne Distanzen (mit eukl. Metrik):
$$\text{dist}(p,q) = \sqrt{(p_x-q_x)^2 + (p_y-q_y)^2}$$
$\rightsquigarrow$ Nachteil: Wurzel!
$\Rightarrow$ Besser: Vergleiche Quadrate der Distanzen:
$$(p_x-q_x)^2 + (p_y-q_y)^2 \overset{?}{\underset{>}{<}} (p_x-r_x)^2 + (p_y-r_y)^2 \;?$$
($\rightsquigarrow$ 4 Subtraktionen, 4 Multiplikationen, 2 Additionen).

1.2.7. Übertragung auf höhere Dimensionen:
Das Orientierungsprädikat kann auf höhere Dimensionen verallgemeinert werden.
Insbesondere im $\mathbb{R}^3$: seien $a,b,c,d \in \mathbb{R}^3$
orientation$(a,b,c,d) \in \{-1,0,1\}$
Sei orientation$(a,b,c,d) \neq 0$, betrachte Ebene durch a,b,c.



1.2.7.1 Satz: Für $a,b,c,d \in \mathbb{R}^3$ betrachte Ebene durch a,b,c.
Falls orientation $\neq 0$, so gilt:
(i) orientation(a,b,c,d) = -1, wenn von d aus $\Delta a,b,c$ negativ orientiert ist.
orientation(a,b,c,d) = +1, wenn von d aus $\Delta a,b,c$ positiv orientiert ist.
(ii) $$\text{orientation}(a,b,c,d) = \operatorname{sign}\begin{vmatrix} a_{x_1} & a_{x_2} & a_{x_3} & 1 \\ b_{x_1} & b_{x_2} & b_{x_3} & 1 \\ c_{x_1} & c_{x_2} & c_{x_3} & 1 \\ d_{x_1} & d_{x_2} & d_{x_3} & 1 \end{vmatrix}$$