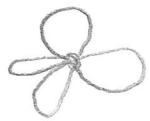
AlgGeo 7.1

Hatten gezeigt: G kreuzungsfrei im $\mathbb{R}^2$, jeder Knoten Grad ≥ 3

$$v \leq \frac{2}{3}e, \qquad v < 2f, \qquad e < 3f$$

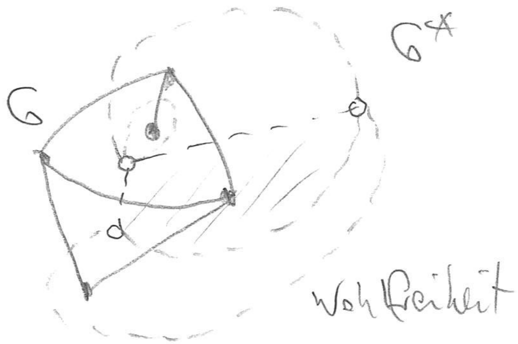
gilt ebenso für kreuzungsfreie Graphen auf Kugel

Frage: Kann man f nach oben durch v abschätzen?

I.A. nicht oben  , 

Def: G heißt schlicht, falls  und  nicht vorkommen.

Def: "Der" duale Graph G^* von G entsteht folgendermaßen:
setze einen Knoten in jede Fläche von G
für jede Kante von G verbinde die Knoten in den benachbarten Flächen



$v^* = f$ (trivial)
$e^* = e$
$f^* = v$

G zusammenhängend $\Rightarrow$ $G^{**} \cong G$

Alg Geo 7.2

G schlicht $\Longrightarrow$ G^* hat Knotengrad ≥ 3

$\underset{\text{s.o.}}{\overset{?}{\Longrightarrow}}$ $f = v^* < 2f^* = 2v$

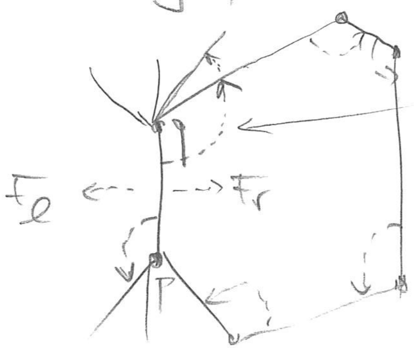
$$\sum_{i=1}^{v} \text{grad}(\text{Knoten } i) = \sum_{j=1}^{f^*} m_j^* \leq 2e^* < 6f^* = 6v$$

$\Longrightarrow$ mittlerer Knotengrad < 6.

Faustformel: Ein "ordentlicher" kreuzungsfreier Graph hat etwa gleich viele Knoten, Kanten und Flächen

Speicherung kreuzungsfreier Graphen: Doubly Connected Edge List DCEL

Kante $e = (p,q)$



hiermit kann man

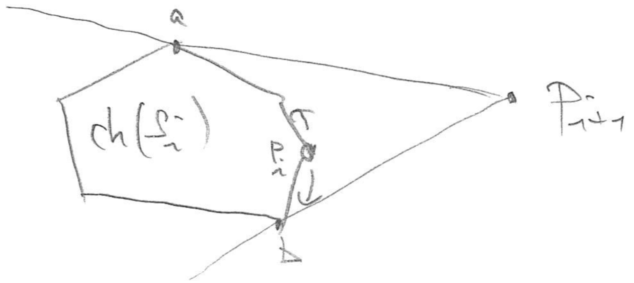
- gegen den Uhrzeiger alle Kanten mit Endpunkt q besuchen
- im Uhrzeiger alle Kanten einer gegebenen Fläche besuchen

beides linear in der Anzahl besuchter Kanten

Alg Geo 7.3

Konvexe Hülle von n Punkten S im $\mathbb{R}^3$ (und höher)

im $\mathbb{R}^2$:



$S_i = \{p_1, \dots, p_i\}$

Anm: Punkte von S in allg. Lage $\Rightarrow$ alle Flächen der konvexen Hülle sind Dreiecke

Frage: Geht das auch im $\mathbb{R}^3$? $ch(S)$ Polyeder

wichtig: Kanten / Ecken von $\partial ch(S)$ bilden einen kreuzungsfreien Graphen; (so gut wie auf S^2 eingebettet) Knotengrad ≥ 3, schlicht, zusammenhängend

Def: p_{i+1} sieht $q \in \partial ch(S_i)$ $:\Longleftrightarrow$ $\overline{p_{i+1} q}$ enthält keine inneren Punkte von $ch(S_i)$

Silhouette von $ch(S_i)$, gesehen von p_{i+1}

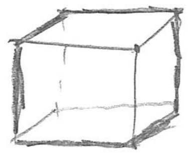
$:=$ Rand der von p_{i+1} aus sichtbaren Punkte von $\partial ch(S_i)$

$=$ Menge aller Tangentenpunkte

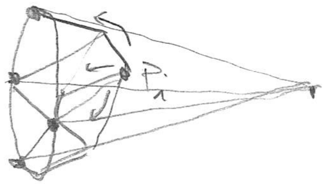
AlgGeo 7.4.

Silhouette: geschlossener Weg, bestehend aus Kanten von $\partial ch(S_i)$
liegt aber i.a. nicht in einer Ebene

Bsp:



Versuch: alter Ansatz: sortiere S nach x, füge sie in dieser Reihenfolge ein



müssen Silhouette finden, Kegel bauen

Silhouette finden: Breitensuche von p_i;
alle dabei besuchten Knoten und Kanten
(außer der Silhouette selbst)
fallen später weg.

Bau des Kegels: Füge von jedem Knoten der Silhouette Kante zu p_{i+1} ein.

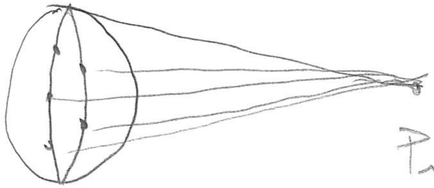
Effizienz?

$\frac{n}{2}$

$\underbrace{p_{i+1} \quad p_{i+2} \quad \dots \quad p_n}_{\frac{n}{2}}$: n^2 Laufzeit ☹

sieht nicht gut aus,
denn $\partial ch(S_n)$ hat nur Größe $O(n)$,
weil es maximal n Knoten gibt.

Alg Geo 7.5



p_{i+1}

\# der hinzuzufügenden Kanten $= \text{grad}(p_{i+1})$ in $\partial ch(S_{i+1})$

Was wäre, wenn p_{i+1} in $S_{i+1} = \{p_1, p_2, \dots, p_{i+1}\}$ zufällig gewählt wäre?

$E(\#$ neue Kanten beim Einfügen von $p_{i+1}) < 6$, denn:

$$= \underbrace{E(\# \dots \mid p_{i+1} \text{ im Inneren von } ch(S_i))}_{0} + \underbrace{E(\# \text{ neue Kanten} \mid p_{i+1} \text{ liegt außerhalb})}$$

dann: p_{i+1} Knoten von $ch(S_{i+1})$

$\Rightarrow$ Knotengrad < 6

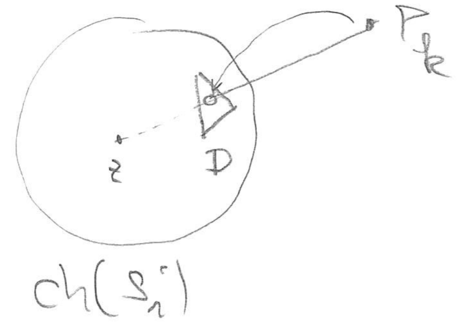
<u>Ansatz</u>: Wähle nicht die nach x sortierte, sondern eine zufällige Einfügereihenfolge $p_1, p_2, \dots, p_n$!

<u>Aber</u>: Von welchem Startpunkt aus sollen wir jetzt die Silhouette suchen? Wenn p_i auf der p_{i+1} abgewandten Seite von $ch(S_i)$ liegt, sollten wir nicht dort starten.

Idee: wählen in $ch(P_1, P_2, P_3, P_4)$ einen inneren Punkt z als Referenzpunkt

zusätzlich zu $ch(S_i)$ speichere für jedes P_k mit $i < k$ einen Zeiger

- mit Wert NIL, falls $P_k \in ch(S_i)$
- auf das von zP_k geschnittene Dreieck auf $\partial(S_i)$



- verzeigere Dreieck D mit allen diesen P_k

Bei Einfügen von P_{i+1}:

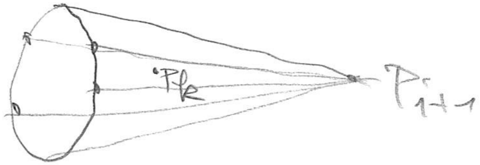
- falls Zeiger = NIL : NOP
- falls Zeiger auf Dreieck D zeigt:
  starte Breitensuche nach Silhouette in D
  (nach Konstruktion ist D von P_{i+1} sichtbar)

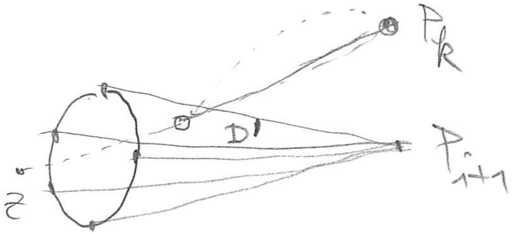
☺, aber wir müssen auch die Zeiger aktualisieren!

Betrachte festes P_k, $k \gg i$. Frage: wie oft muss der Zeiger von P_k umgehängt werden?

Alg Geo 7.7.

- auf NIL: geschieht nur 1X:
  müssen testen, ob P_k im Kegel liegt
  in Zeit $O(\text{grad } P_{i+1})$





- auf neues Dreieck:

wann immer das geschieht, ist P_{i+1} ein Eckpunkt von D'

Backwards analysis (R. Seidel):

Vorstellung: $ch(S_{i+1})$ fertig, D' ist eines der Dreiecke auf $\partial ch(S_{i+1})$

jetzt: Wähle P_{i+1} zufällig in $\{P_1, P_2, \dots, P_i, P_{i+1}\}$

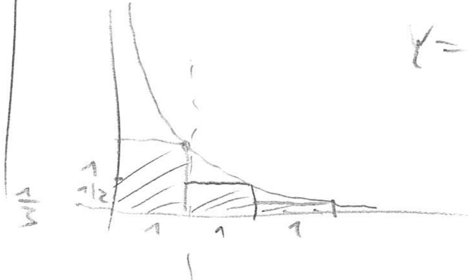
$$P(P_{i+1} \text{ ist Eckpunkt von } D') \le \frac{3}{i+1}$$

=> die zu erwartende Anzahl von Zeigeränderungen für festes P_k:

$$\le \sum_{i=1}^{n} 3 \cdot \frac{1}{i+1} \le 3 + 3\ln(n+1)$$

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n} \le \underbrace{1 + \int_1^n \frac{1}{x}\,dx}_{1+\ln n}$$

$y = \frac{1}{x}$



Alg Geo 7.8.

Theorem Konvexe Hülle von n Punkten im $\mathbb{R}^3$

hat Komplexität $O(n)$
lässt sich durch Randomized Incremental Construction (RIC)
in mittlerer Zeit $O(n \log n)$ berechnen.

(Buch: $d=2$)

Es geht auch in Worst-Case Zeit $O(n \log n)$: "Gift Wrapping"
Divide & Conquer

Ganz einfaches Beispiel für Backwards Analysis
Max in Array:

Max := A[1]
for i := 2 to n do
if $\underbrace{A[i] > max}_{\text{billig}}$ then $\underbrace{max := A[i]}_{\text{teuer}}$

Aktualisierungen: worst case: A aufsteigend sortiert: n mal teures Update

AlgGeo 7.9 Bei zufälliger Reihenfolge:

Aktualisierung bei A[i] notwendig

$\Leftrightarrow$ A[i] ist größtes Element der Menge {A[1], A[2], .., A[i]}

Aber A[i] ist ein zufälliges Element dieser Menge (Backwards analysis)

Also $P(A[i] = \max_{j=1,..,i} A[j]) = \frac{1}{i}$

$\Rightarrow$ Anzahl von Updates insgesamt im Mittel $\leq \log n$.

Randomisierte Alg: Los Vegas - Prinzip — liefert immer korrektes Ergebnis, im Mittel gute Laufzeit

Monte Carlo — liefert mit hoher Wahrscheinlichkeit korrektes Ergebnis