# Übungsblatt 11

Julius Auer, Alexa Schlegel

**Aufgabe 1** (Konfliktecken):

Gegeben seien eine Menge $H_n$ von $n$ Halbebenen (bzw. Punkten im Dualen) deren Schnitt $S(H_n)$ berechnet werden soll. Zur Initialisierung wählt man 4 beliebige Halbenenen und berechnet paarweise alle Schnittgeraden zwischen diesen Ebenen sowie alle Schnittpunkte zwischen diesen Schnittgeraden. Diese Schnittpunkte sind nun die Ecken des initialen Polyeders, der den Halbraum $H_4$ beschreibt.

Der Halbraum $H_4$ muss an dieser Stelle nicht beschränkt sein, was bei der Erstellung des Konfliktgraphen problematisch sein kann. Da sich nach Annahme keine 4 Halbräume in einem Punkt schneiden, seien nun nur Ecken mit Grad 3 valide. Solange es nicht valide Ecken gibt, wird wie folgt verfahren:

(1) Wähle nicht valide Ecke $v$ vom Grad 2 und fälle das Lot auf die Gerade durch die adjazenten Ecken.

(2) Konstruiere eine Ebene $e$ deren Normalenvektor dieses Lot ist und verschiebe sie näherungsweise in's Unendliche. Wähle die Richtung der Verschiebung so, dass sie sich von $v$ entfernt.

(3) Erweitere $S(H_4)$ um $e$ (brute-force).

So entstehen zusätzliche Ebenen die den Polyeder beschränken, wobei aber garantiert ist, dass $S(H_n)$ vollständig auf einer Seite jeder dieser Ebenen liegt.

Im schlimmsten Fall wird dadurch ein Viereck mit Seiten im Unendlichen konstruiert - also höchstens 6 zusätzliche Ebenen eingefügt. Das benötigt konstante Zeit.

Gibt es nur noch valide Ecken kann wie gewohnt der Konfliktgraph konstruiert werden, indem für jede Ebene $h \in H_n$ geprüft wird, welche Ecke von $S(H_4)$ durch $h$ abgetrennt wird und die entsprechenden Zeiger angelegt werden.

**Aufgabe 2** (randomisiert inkrementelle Kontruktionen):

a) Eingabe ist eine Punktmenge $S_n \subset \mathbb{R}^2$ aus $n$ Punkten, von denen der Einfachheit halber keine 3 kolinear sind. Die konvexe Hülle $ch(S_n)$ sei im Uhrzeigersinn gegeben als zyklische, doppelt verkettete Liste von Strecken und kann folgenderweise berechnet werden:

(1) Wähle drei beliebige Punkte und konstruiere $ch(S_3)$. Wähle einen Punkt $p_m$ dieser Punkte als festen Referenzpunkt innerhalb der konvexen Hülle. (konstante Zeit)

(2) Prüfe für jeden Punkt $p \in S \setminus S_3$ welche Strecke aus $ch(S_3)$ die Strecke $\overline{p_m p}$ schneidet. Jeder Strecke der konvexen Hülle werden (als Liste) alle entsprechenden Punkte zugeordnet - jeder Punkt hat seinerseits einen Zeiger auf die entsprechende Strecke. Diese Struktur wird während des Algorithmus aufrecht erhalten. (lineare Zeit)

(3a) Im $i$-ten Schritt des Algorithmus wird ein zufällig gewählter Punkt $p_i$ zur bisher konstruierten konvexen Hülle $ch(S_{i-1})$ hinzugefügt: Die passende Strecke $\overline{p_l p_r} \in ch(S_{i-1})$ (die von $\overline{p_m p_i}$ geschnitten wird) ist direkt zu $p_i$ gespeichert und wird aus $ch(S_{i-1})$ entfernt, wofür $\overline{p_l p_i}$ sowie $\overline{p_i p_r}$ hinzugefügt werden. Anschließend wird die konvexe Hülle nach links und rechts "abgelaufen": liegt für eine so erreichte Strecke $\overline{p_l p_r}$ der Punkt $p_r$ links von $\overline{p_l p_i}$ (beim links herum laufen) bzw. $p_l$ rechts von $\overline{p_r p_i}$ (beim rechts herum laufen) wird der Punkt aus $ch(S_i)$ entfernt. (im schlimmsten Fall können also $O(n)$ Punkte angelaufen und entfernt werden)

(3b) Für alle entfernten Strecken müssen die Listen aktualisiert werden: für jeden Punkt aus der Liste einer aus $ch(S_i)$ entfernten Strecke muss geprüft werden, ob die Strecke von dort nach $p_m$ nun die links/rechts benachbarte Strecke schneidet oder sogar innerhalb von $ch(S_i)$ liegt. (in jedem Schritt des Algorithmus ggf. $O(n)$)

Insgesamt fallen somit Kosten von $O(n^2)$ an - sowohl die Kosten für (3a) als auch die für (3b) lassen sich aber günstiger abschätzen:

Da jede Strecke nur einmal hinzugefügt und somit nur einmal entfernt werden kann benötigt (3a) nur lineare Zeit über den ganzen Algorithmus.

Für (3b) kann eine Abschätzung mit der aus der Vorlesung bekannten Backward-Analyse vorgenommen werden:
Beim Rückwärtslaufen wird ein Punkt $p$ aus $ch(S_i)$ entfernt um $ch(S_{i-1})$ zu erhalten. Die Kante $e$ aus $S_i$ die von $\overline{p_m p}$ geschnitten wird, wird genau dann gelöscht, wenn ein Eckpunkt im Rückwärtsschritt gelöscht wird. Für jeden Punkt geschieht das mit einer Wahrscheinlichkeit von $O\left(\frac{1}{i}\right)$. Insgesamt müssen erwartet $O\left(\frac{n}{i}\right)$ Zeiger aktualisiert werden, womit auch für (3b) insgesamt nur linearer Aufwand erwartet wird.

Die Gesamtlaufzeit lässt sich für zufällige Punkte somit auf $O(n \cdot \log n)$ abschätzen.

b) Gegeben ist die Punktmenge $P = \{p_1, \ldots, p_n\} \in \mathbb{R}^2$. Der folgende randomisierte inkrementelle Algorithmus konstuiert das Voronoi-Diagramm $VD(P)$.

Im $i-$ten Schritt ist $VD(p_1, \ldots, p_i)$ gegeben, der Punkt $p_{i+1}$ soll eingefügt werden:

(1) Bestimme die Voronoi Region $r$ in der $p_{i+1}$ liegt und das zugehörige Zentrum $p_r$(in $O(n)$).

(2) Konstruiere eine zu $\overline{p_{i+1}, p_r}$ orthogonale Gerade $g$ durch $p_r$.

(3) Bestimme die 1-4 Schnittpunkte $S$ von $g$ mit allen Voronoi Kanten der Region $r$.

Solange $S$ nicht leer ist:

(4) Wähle und entferne $s \in S$. $s$ ist neue Voronoi Ecke.

(5) durch $s$ getroffene Voronoi Kante $e$ wird bestimmt und zu $s$ verkürzt.

(6) Orthogonale Gerade zwischen $p_{i+1}$ und dem Zentrum der anderen Region an $e$ aufstellen und die Schnittpunkte mit den Kanten der Region zu $S$ hinzufügen. Voronoi Kante zwischen dem neu gefundenen Schnittpunkten und $s$ einfügen.

Laufzeit:
(1) $O(n)$
(2) $O(1)$
(3) Auf keinen Fall schlimmer als $O(n)$
insgesamt $O(n)$

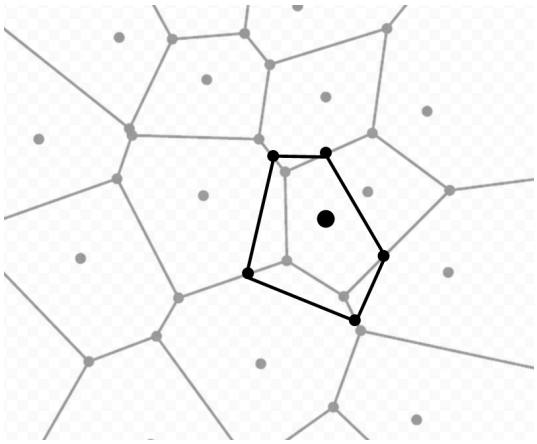

Abbildung 1: Eine neue Voronoi Region