Gesamtzusammenfassung

Elmar Langetepe
University of Bonn

Kapitel 2 Sweep Technik

- Ausfegen der Ebene mit Sweepline
- Komplexitätsreduktion: n-Dim nach (n-1)-Dim
- Sortieren und gem. Sort. fegen
- SSS (Sweep-Status-Struktur): Invariante in der Nähe der Sweepline
- ES (Ereignisstruktur): Haltepunkte der Sweepline
- Ereignisverarbeitung: Aktualisierung SSS
- Laufzeitanalyse: (# Ereignisse) x Kosten(Verab)
- Korrektheit Ergebnis: Invariante erfüllt
- Einfache Beispiele, komplexe Beispiele

Eindimensionaler Sweep

- Korollar 2.1: Die Berechnung des Closest Pairs von n reellen Zahlen hat Zeitkomplexität $O(n \log n)$.
  (Einfach, Eindimensionaler Sweep!)
- Theorem 2.2: Die Berechnung des Maximum Subvektors n konsekutiver reeller Zahlen hat Zeitkomplexitätzt $\Theta(n)$.
  (Invariante SSS: Max-Ending-Here, Aktualisierung, Beweis dazu! Unterschied: Divide and Conquer!)

-1	2	-3	5	-2	-1	6	-2	0	-1	-9	3	-1	3	?

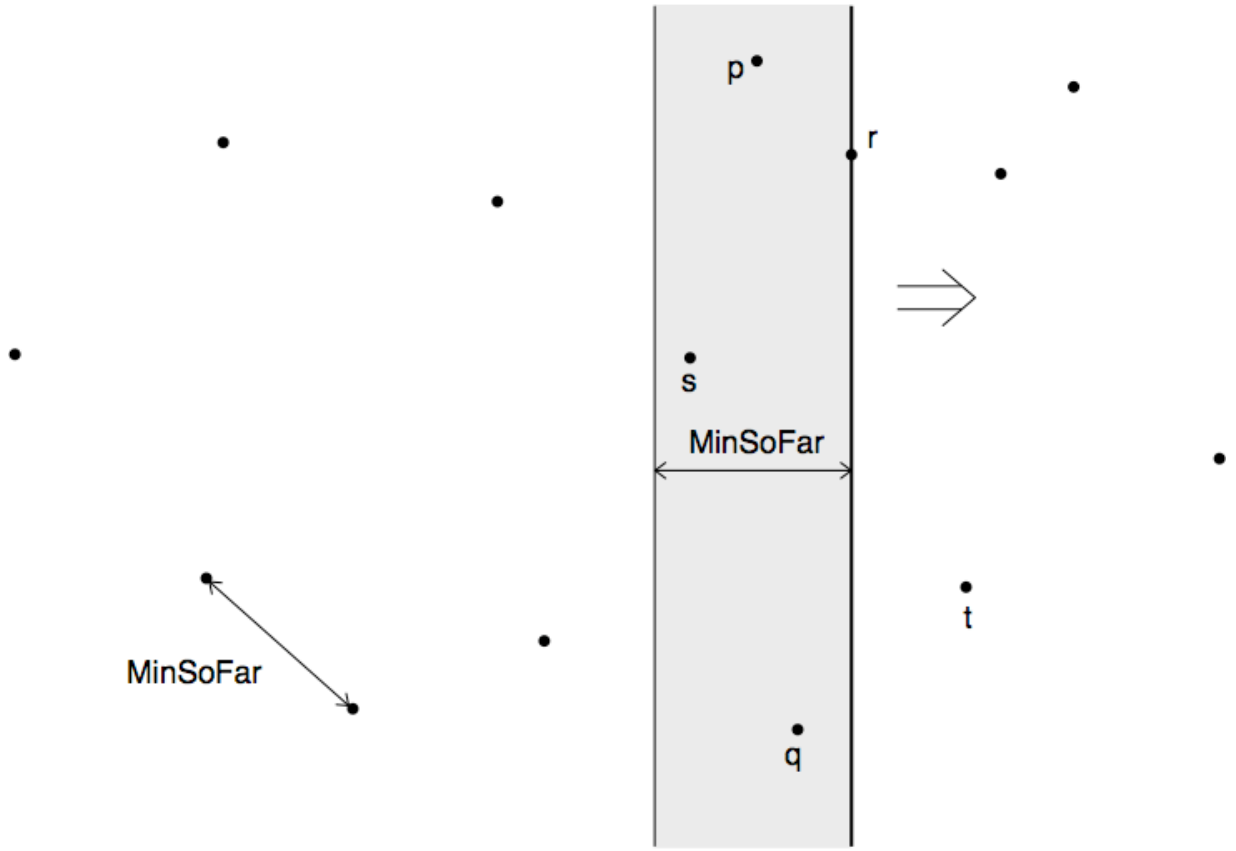
Closest-Pair Ebene

- Theorem 2.4: Der minimale Abstand aller Paare einer n-elementigen Punktmenge in der Ebene läßt sich in Zeit $O(n \log n)$ bestimmen.

  (SSS: Streifen mit Breite MinSoFar, Ereignisverarbeitung, Datenstruktur Y-Koordinate! Hilfslemma!)
- Lemma 2.3: Sei $M > 0$ und P eine Menge von Punkten im $\mathbb{R}^2$ von denen je zwei den Abstand $\geq M$ haben. Ein Rechteck mit Kantenlängen M und $2M$ enthält höchstens 10 Punkte.

  (Fläche und leere Kreise, $\log n + 10$ für Test)

Closest-Pair Ebene



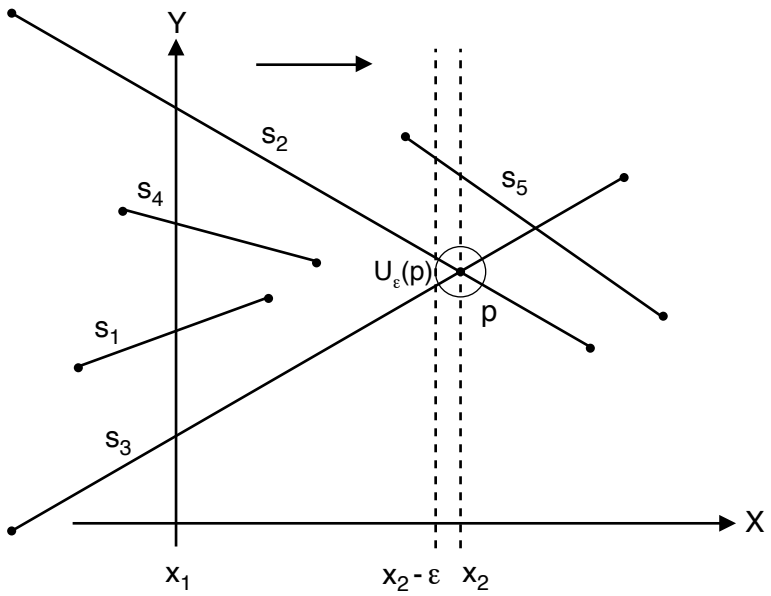
Liniensegment-Schnitt Ebene

- Theorem 2.10: Das Sweep-Verfahren berechnet die k echten
- Schnittpunkte von n Liniensegmenten in Zeit $O((n+k)\log n)$ mit Platz $O(n)$.

  (Zwischenschritte zu diesem Ergebnis
  Sparrregel Platzbedarf)
- Lemma 2.7: Haben zwei Liniensegmente einen echten Schnittpunkt zum Zeitpunkt t, dann sind die Segmente unmittelbar vor t Nachbarn in der SSS gewesen.

  (Wesentliche strukturelle Eigenschaft)

SSS: Schnittpunkte von Liniensegmenten



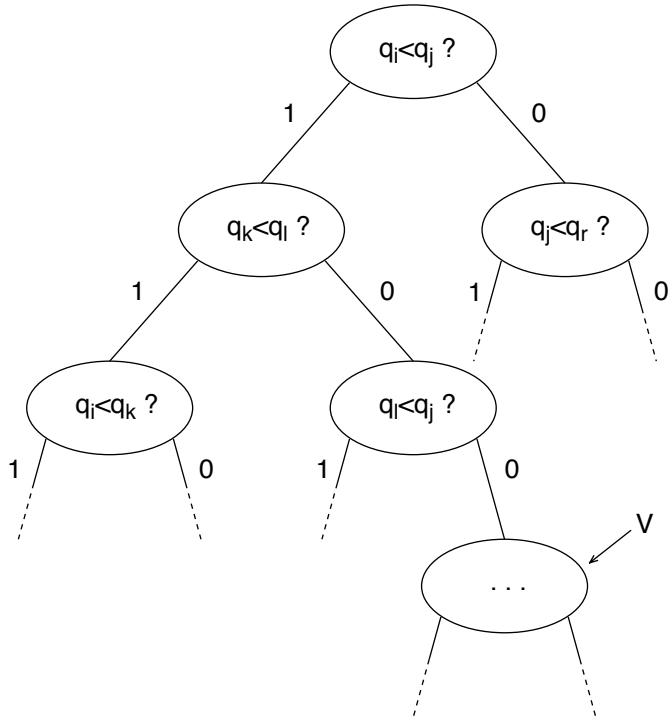
SSS: Segmente gemäß Ordnung, Ereignisse:

1. Neuer linker Endpunkt
2. Neuer rechter Endpunkt
3. Schnittpunkt

Kap. 1: Untere Schranken

- Theorem 1.4: Sortieren durch Schlüsselvergleiche hat die Zeitkomplexität $\Omega(n \log n)$.
  (Entscheidungsbaum, Lineares Modell, Min. Höhe!)
- Theorem 1.5: Sei $W \subseteq \mathbb{R}^n$ eine Menge mit m Zusammenhangskomponenten. Dann benötigt jeder Algorithmus für den Elementtest von W mindestens $\log m$ viele Schritte.
  ($W \subseteq \mathbb{R}^n$, liegt ein $x \in W$? Hyperebenentests!)
- Korollar 1.6/Korollar 1.7: ϵ-Closeness/Element-Uniqueness hat im linearen Modell eine Zeitkomplexität von $\Theta(n \log n)$.
  (Elementt. $W = \{(x_1, x_2, \ldots, x_n) | |x_i - x_j| \geq \epsilon$ für alle $i \neq j\}$)

Kap. 1: Untere Schranken



Operationen $K(i,j) = \begin{cases} 1 & \text{falls } q_i < q_j \\ 0 & \text{falls } q_i > q_j \end{cases}$

Kap. 1: Untere Schranken

- Korollar 1.8: Sortieren hat auch im linearen Modell eine Zeitkomplexität von $\Theta(n \log n)$.

  (ϵ-Closeness/Element Uniqeness $\leq_P$ Sortieren, $P \in O(n)$!)
- Korollar 1.9: Für jeden Punkt einer n elementigen Menge $S \subseteq \mathbb{R}^d$ seinen nächsten Nachbarn zu finden, hat Zeitkomplexität $\Omega(n \log n)$.

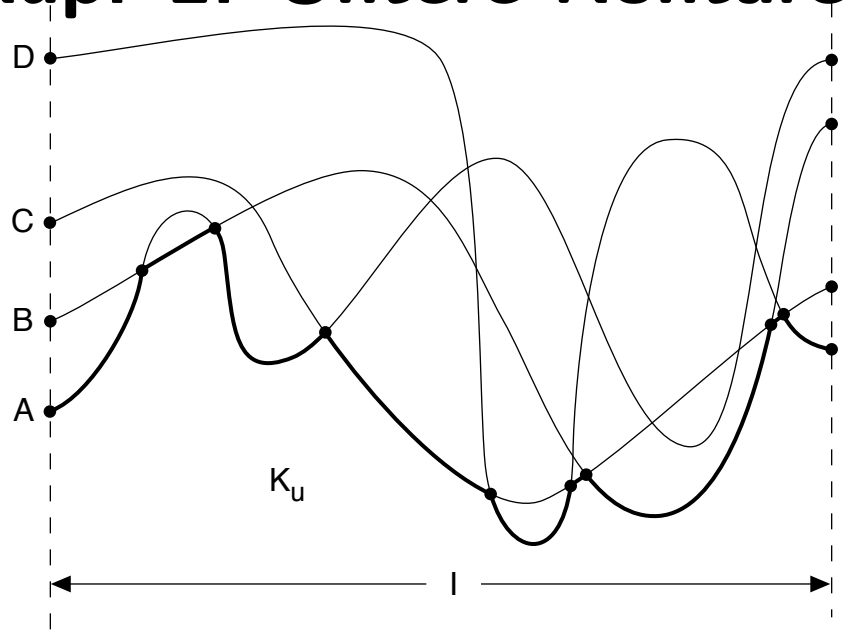
  (ϵ-Closeness $\leq_P$ All-Nearest-Neighbors, $P \in O(n)$!)
- Korollar 1.10: Das dichteste Punktepaar (closest-pair) einer n elementigen Menge $S \subseteq \mathbb{R}^d$ zu finden, hat Zeitkomplexität $\Omega(n \log n)$.

Kap. 1: Untere Schranken

- Lemma 2.6: Das Existenzproblem für den Schnitt von n Liniensegmenten hat Zeitkomplexität $\Omega(n \log n)$. Das Aufzählungsproblem für k Schnittpunkte hat Zeitkomplexität $\Omega(n \log n + k)$.

  (ϵ-Closeness $\leq_P$ Existenz Schnitt Liniensegmente, $P \in O(n)$!)

Kap. 2: Untere Konturen



- Theorem 2.11: Mit dem Sweep-Verfahren aus Abschnitt 2.3.2 lassen sich die k Schnittpunkte von n verschiedenen X-monotonen Wegen in Zeit $O((n+k)\log n)$ berechnen.
  (Genauso! Schnitt in O(1)!)

Kap. 2: Untere Konturen

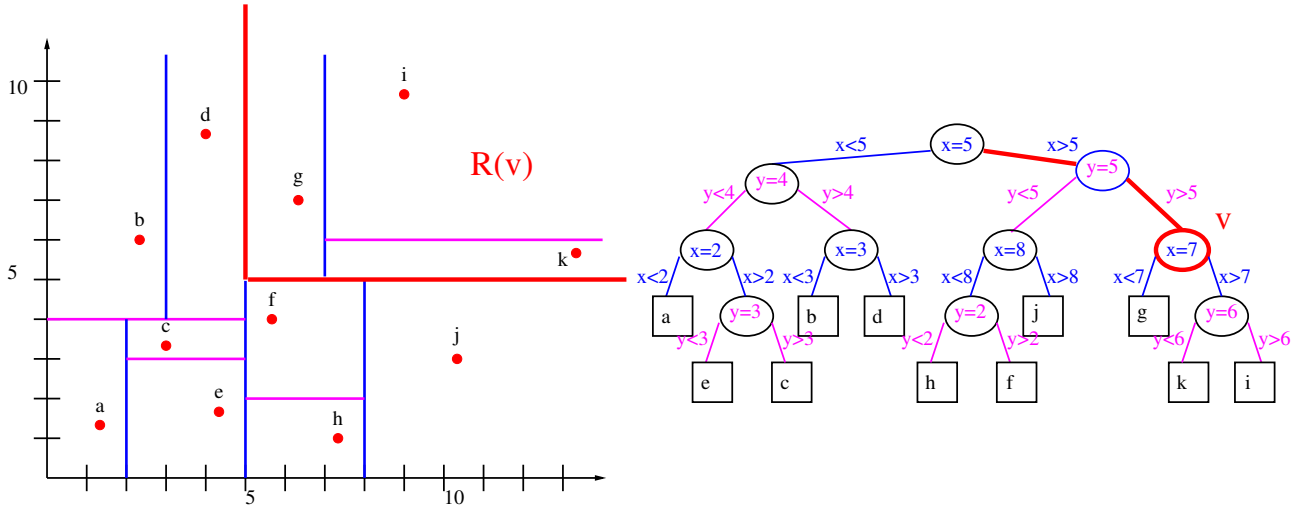
- Theorem 2.13: Die untere Kontur von n verschiedenen X monotonen Wegen über einem gemeinsamen Intervall, von denen sich je zwei höchstens s-mal schneiden, kann in Zeit $O(\lambda_s(n) \log n)$ berechnet werden.
  (Geringere Komplexität/nicht alle Schnitte!, Div. and Conq.)
- Lemma 2.12: Für alle $s, n \geq 1$ gilt $2\lambda_s(n) \leq \lambda_s(2n)$
- Definition: $\lambda_s(n)$: Maximale Komplexität der Kontur bei obigen Bedingungen.
- $\lambda_s(n)$, kombinatorische Definition:
  DSS der Ordnung s von n Buchstaben. Maximale Länge!
- Theorem 2.14: Die maximale Länge einer DSS der Ordnung s über n ist $\lambda_s(n)$. (Jeweils übertragen!)

Kap. 2: DSS und $\lambda_s(n)$

- Kompl. Kontur von Funktionen:
  Für ganzes Intervall: $\lambda_s(n)$
  Für Teilintervalle: $\lambda_{s+2}(n)$
  (Mit Strahlen verlängern!)
- Korollar 2.15: Die untere Kontur von n Liniensegmenten über einem gemeinsamen Intervall kann in $O(n \log n)$ mit Platz $O(n)$ berechnet werden.
- Korollar 2.16: Die untere Kontur von n Liniensegmenten beliebiger Länge enthält $\lambda_3(n)$ viele Segmente und kann in Zeit $O(\lambda_3(n) \log n)$ berechnet werden.
  (Anwendung der Ergebnisse)

Kap. 3: Geometrische Datenstrukturen

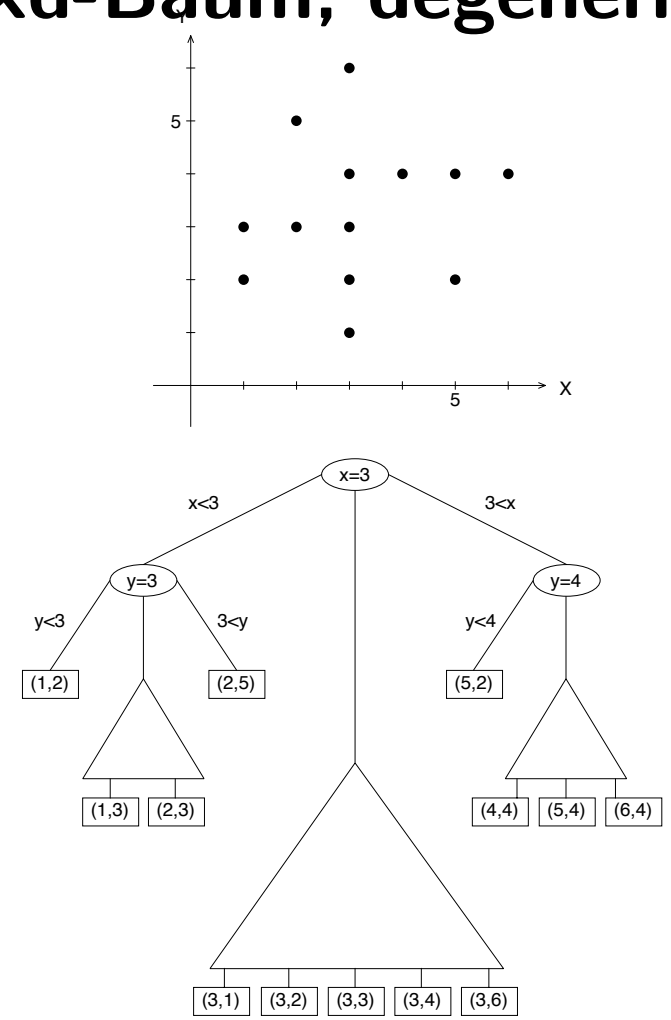
- Datenobjekte Punkte im $\mathbb{R}^k$, andere Objekte übertragen
- Bislang: Eindimensionaler balancierter Suchbaum
- Bereichsanfrage: Inklusionsanfrage, Schnittanfrage, Anfrageobj. q
- kd-Baum, Rechteckige Bereichsanfrage im $\mathbb{R}^k$



Kap. 3: kd-Baum

- Lemma 3.6: Sei T ein 2d-Baum der Höhe h mit n Punkten. Eine Bereichsanfrage mit achsenparallelen Rechteck Q läßt sich in Zeit $O\left(2^{\frac{h}{2}} + a\right)$ beantworten, wobei a die Größe der Antwort ist.
  (Knoten v mit $R(v) \subseteq q$ und
  Knoten v mit $R(v) \not\subseteq q$ (aber $R(v) \cap q \neq \emptyset$))
- Theorem 3.7: Ein ausgeglichener 2d-Baum für n Punkte in der Ebene läßt sich in Zeit $O(n \log n)$ konstruieren. Er benötigt $O(n)$ Speicherplatz . Eine Bereichsanfrage mit achsenparallelen Rechteck q kann in Zeit $O\left(\sqrt{n} + a\right)$ beantwortet werden, wobei a die Größe der Antwort ist.
  (Sortieren X und Y rekursiv in gleichgroße Teilmengen)

Kap. 3: kd-Baum, degenerierte Fälle



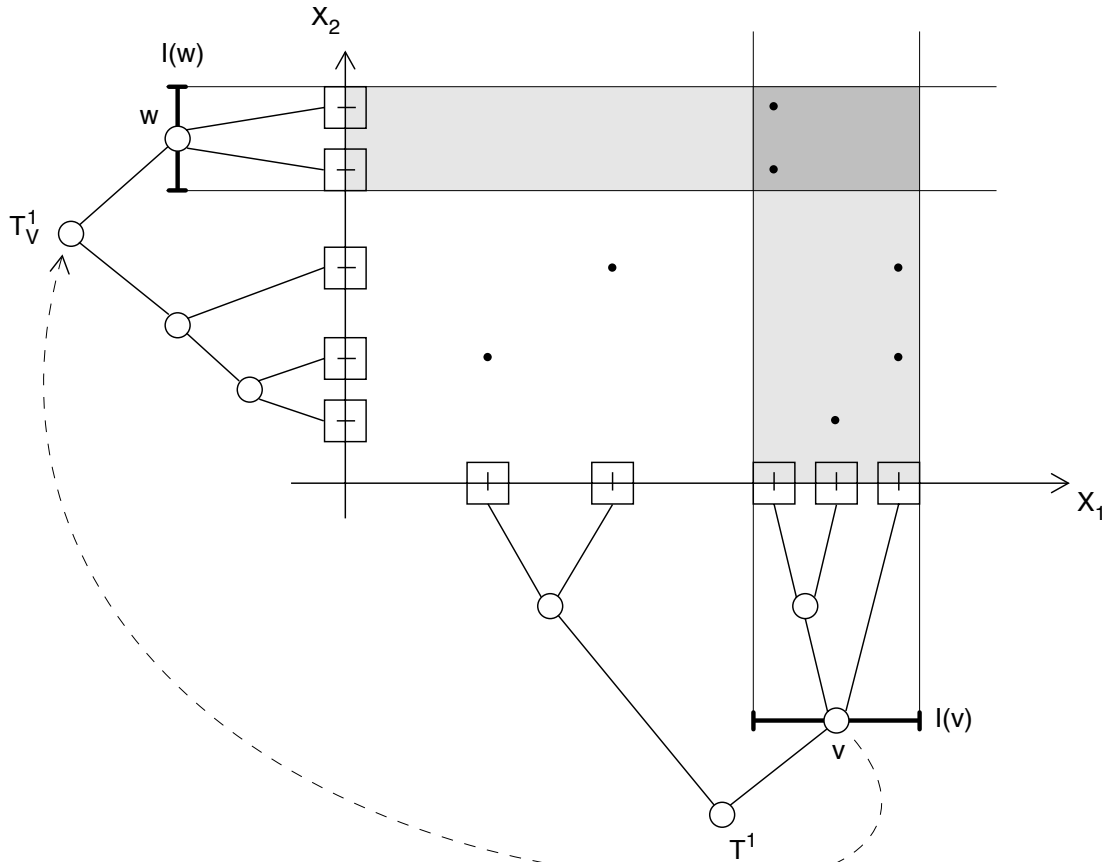
Kap. 3: kd-Baum

- Theorem 3.7: (entartet): Ein ausgeglichener kd-Baum für n Punkte im $\mathbb{R}^k$ läßt sich in Zeit $O(n \log n)$ konstruieren. Er benötigt $O(n)$ Speicherplatz. Eine orthogonale Bereichsanfrage kann in Zeit $O\left(n^{1-\frac{1}{k}} + a\right)$ beantwortet werden, wobei a die Größe der Antwort ist.

  (k zyklische Splitebenen, ternäre Struktur,
  Datenobjekte in Split(hyper)ebene: $(k-1)d$-Baum)
- Anwendung: Sweep 3D, Closest-Pair: Laufzeit $O(n(e(n) + b(n)))$

  (2d-Baum: $b(n) \in O(\sqrt{n} + a)$, $e(n) \in O(\log n)$)

Kap. 3: Bereichsbaum

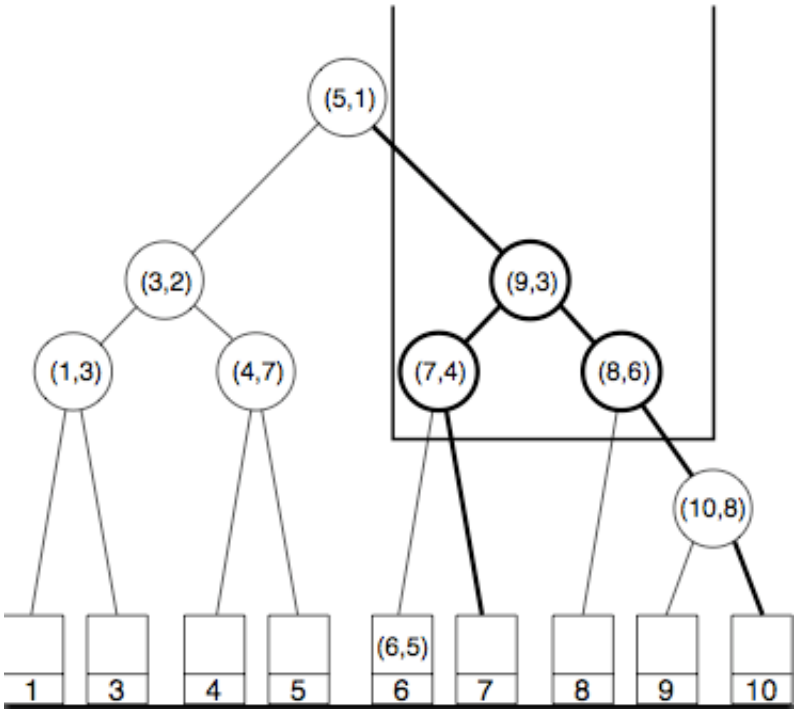


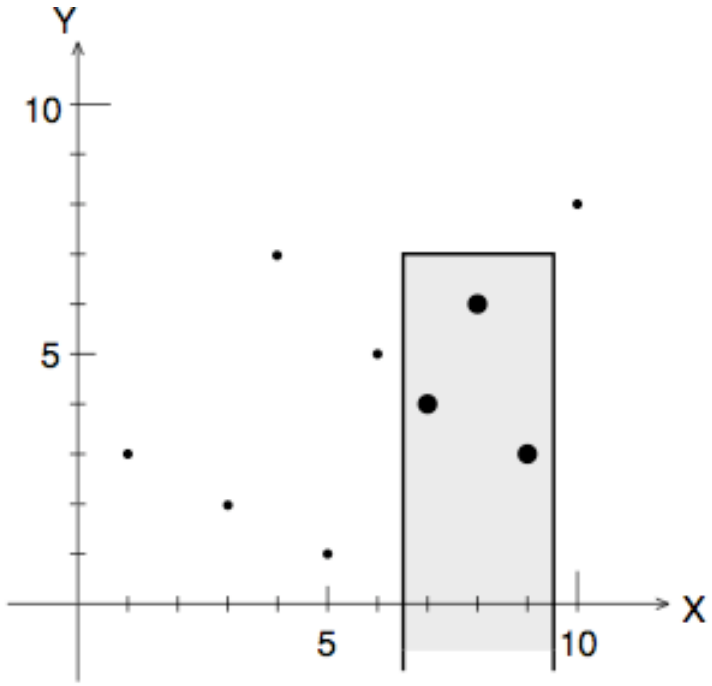
Kap. 3: Bereichsbaum

- Theorem 3.11: Ein k-dimensionaler Bereichsbaum für n Punkte im $\mathbb{R}^k$ kann in Zeit $O(n(\log n)^{k-1})$ mit Platz $O(n(\log n)^{k-1})$ aufgebaut werden. Eine Bereichsanfrage mit Hyperrechteck $q \subset \mathbb{R}^k$ kann in Zeit $O(a + (\log n)^k)$ beantwortet werden. Dabei ist a die Größe der Antwort.

  (Platz: Rekursiver Aufbau! Vorkommen eines Punktes!
  Query: Anzahl Intervalle! Induktiver Beweis!
  Aufbau: Sortieren, Aufteilungsschritt, Rekursiv!)

Prioritätssuchbaum Beispiel





- 1. Jeder Punkt auf dem Weg zu seiner X-Koordinate
- 2. Punkte entlang des Pfades nach Y-Koordinaten
- 3. So nah wie möglich an der Wurzel

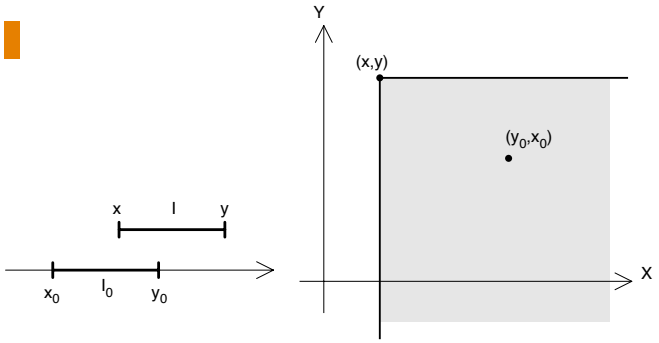
Kap. 3: Prioritätssuchbaum

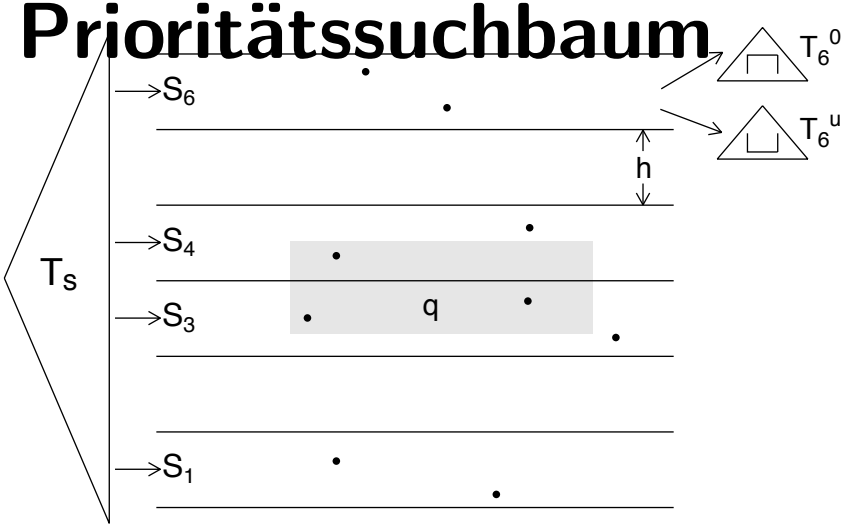
- Theorem 3.14 Ein Prioritätssuchbaum für n Punkte in der Ebene kann in Zeit $O(n \log n)$ aufgebaut werden. Er benötigt $O(n)$ viel Platz. Eine Halbstreifenanfrage kann in Zeit $O(a + \log n)$ beantwortet werden. Dabei ist a die Größe der Antwort.

  (Eindimensionaler Bereichsbaum für X-Koord.
  Heap für Y-Koord., Platz vorhanden!
  $H = [x_1, x_2] \times (-\infty, y]$: X-Grenzen, Im Innern bis Tiefe y)
- Theorem 3.17: Man kann n Intervalle mit Platz $O(n)$ so abspeichern, dass sich eine Überlappungsanfrage eines Intervalls I_0 in Zeit $O(a + \log n)$ beantworten läßt. a ist Größe der Antwort.

  (Anwendung: Intervalle in Punkte übertragen.
  Anfrage Viertelebene $[x_0, \infty) \times (-\infty, y_0]$.)

Kap. 3: Prioritätssuchbaum





- Lemma 3.16: Seien $I_0 = [x_0, y_0]$ und $I = [x, y]$ zwei Intervalle, dann gilt: I_0 überlappt mit $I \iff x \leq y_0$ und $x_0 \leq y$ (Strukturelle Eigenschaft Intervalle)
- Theorem 3.18 Sei $h > 0$ fest. n Punkte in der Ebene lassen sich so mit Platz $O(n)$ abspeichern, dass jede Rechteckanfrage mit Höhe h in Zeit $O(\log n + a)$ (a Größe der Antwort) beantwortet werden können.

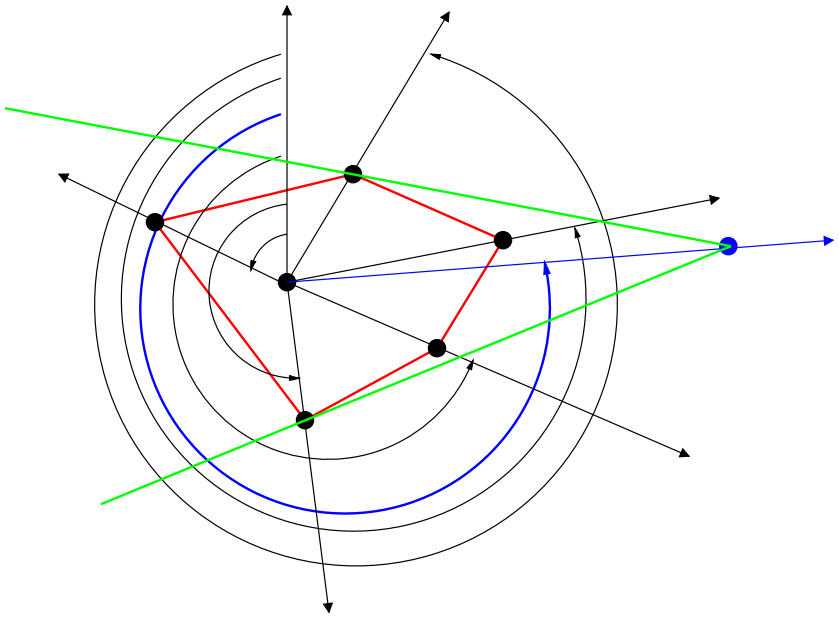
Kap. 4: Konvexe Hülle

- Lemma 4.1 Sei S eine Menge von n Punkten in der Ebene, dann ist $ch(S)$ ein konvexes Polygon mit Eckpunkten aus S.
  (Formale Definition Schnitt, induktiver Beweis!)
- Lemma 4.2: Die Konstruktion der konvexen Hülle von n Punkten in der Ebene hat Zeitkompleximtät $\Omega(n \log n)$.
  (Untere Schranke, Sortieren reduzieren auf konv. Hülle)
- Lemma 4.3: Punktmenge S gegeben. Sei W ein einfacher, geschlossener Weg, der den Rand von $ch(S)$ umschließt. Dann ist der Rand von $ch(S)$ höchstens so lang wir W.
  (Strukturelle Eigenschaft, Sektoren Winkelhalbierende)

Kap. 4: Konvexe Hülle, Inkrementelle Konstruktion

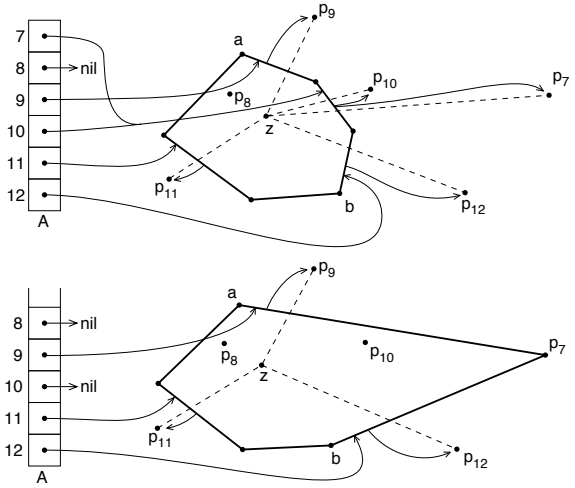
- Theorem 4.5 Die konvexe Hülle einer Menge von n Punkten in der Ebene zu konstruieren hat Zeitkomplexität $\Theta(n \log n)$ mit linearem Speicher.

  (Lokalisieren Innen/Außen, Tangentenbest., Baum mit Winkeln)



Kap. 4: Randomisierte inkrementelle Konstruktion

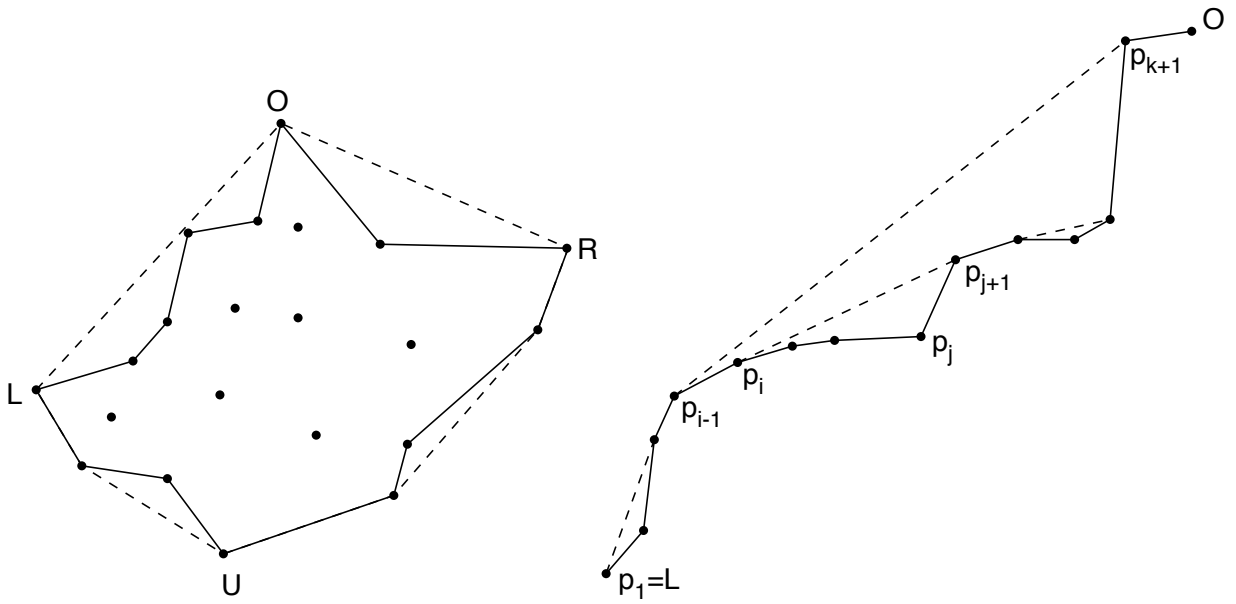
- Theorem 4.6: Wird jede der $n!$ vielen Eingabereihenfolgen einer festen Punktmenge $S = \{p_1, p_2, \ldots, p_n\}$ mit gleicher Wahrscheinlichkeit gewählt, dann ergibt sich für die inkrementelle Konstruktion eine erwartete Anzahl von $O(n \log n)$ Operationen,
  (Erwartet: $O(n \log n)$ Konfliktpaare, WC-Reihenfolge: $\Omega(n^2)$).
- (p_i, p_j) in Konflikt $\iff$ zp_j schneidet Kante von $ch(S_{i-1})$, die beim Einfügen von p_i entfernt wird, $O(\log j)$ viele!



Kap. 4: Einfaches lineares Verfahren

- Theorem 4.7 Die konvexes Hülle von n sortierten Punkten kann in Zeit $O(n)$ berechnet werden.

  (Sortieren: 1. Konturpolygon! 2. *Geradeziehen* Ketten!)



Kap. 4: Dualität

- Dualität:

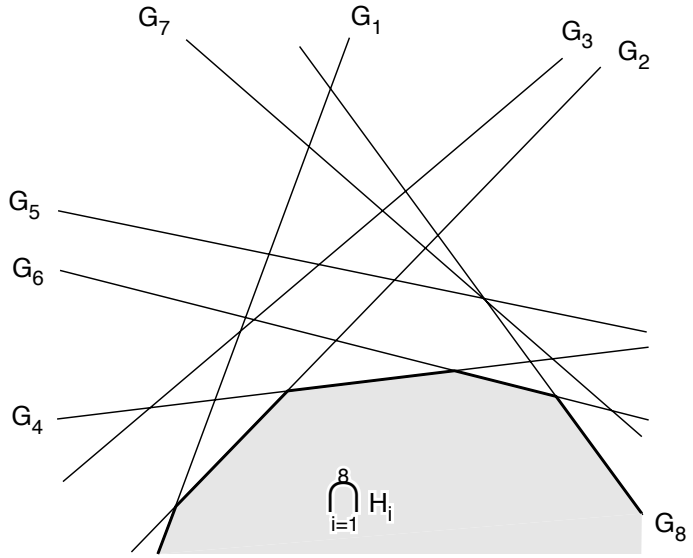
$$\begin{aligned} p = \quad (a,b) \quad &\longrightarrow \quad p^* = \{Y = aX + b\} \\ G^* = (-a,b) \quad &\longleftarrow \quad G = \{Y = aX + b\} \end{aligned}$$

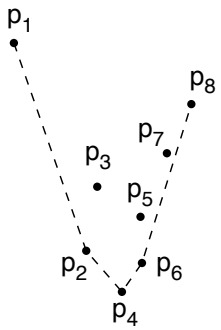
- $p \in G \Longleftrightarrow G^* \in p^*$ (relative Lage bleibt erhalten)
- Lemma 4.8: Für einen Punkt p und eine Gerade G haben wir $\mathit{gva}(p, G) = -\mathit{gva}(G^*, p^*)$. Insbesondere gilt:
  p liegt oberhalb von $G \Longleftrightarrow p^*$ verläuft oberhalb von G^*
  (direkte Übersetzung!)
- Lemma 4.9: Seien $p = (a, b)$ und $q = (c, d)$ mit $a \neq c$ und $\ell(pq)$ die Gerade durch p und q. Dann gilt: $p^* \cap q^* = (\ell(pq))^*$.
  (Ausrechnen!)

Kap. 4: Anwendung Dualität

- Theorem 4.10: Arrangement von n Geraden G_i:
- $G_i \cap G_j$ ist Eckpunkt des Durchschnitts der unteren Halbebenen $\Longleftrightarrow$ das Liniensegment $G_i^* G_j^*$ ist eine untere Kante der konvexen Hülle der Punkte G_i^*.

  (Beweis: Dualität direkt anwenden!)

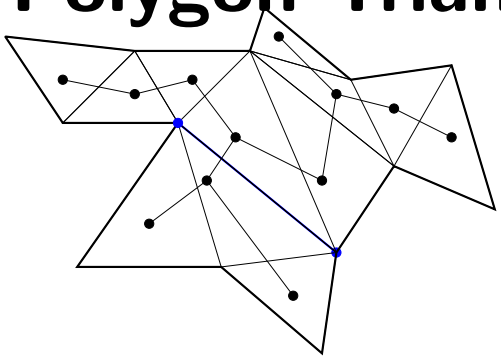




Kap. 4: Anwendung Dualität

- Korollar 4.11 Die Berechnung des Durchschnitts von n Halbebenen hat die Zeitkomplexität $\Theta(n \log n)$.
  (Sonst: Konvexe Hülle schneller!)
- Korollar 4.12 Der Schnitt von n Halbebenen, deren Geraden nach Steigung sortiert sind, kann in Zeit $O(n)$ berechnet werden.
  (Dual: Konvexe Hülle sort. eine Koordinate)

Kap. 4: Polygon-Triangulation



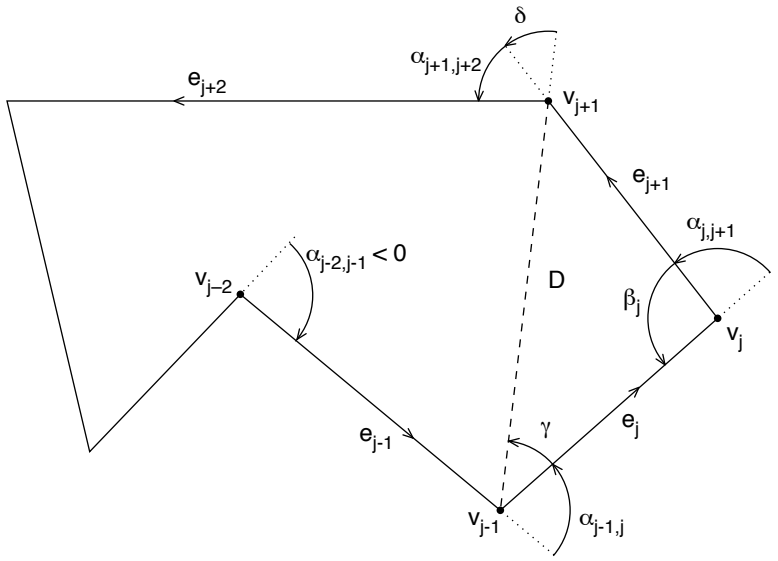
- Definition: Maximale Anzahl, nicht-kreuzender Diagonalen
- Lemma 4.13: Ist P konvex, dann bildet jedes nicht-konsekutive Paar von Ecken eine Diagonale. Ist P nicht-konvex, und v eine beliebige spitze Ecke (Innenwinkel $> 180°$), dann gibt es eine Diagonale mit Eckpunt v.
  (Konstruktiv! Nützlich, Polygon zerlegen!)
- Lemma 4.14: Jedes einfache Polygon P kann trianguliert werden.
  (Induktion: Anwendung Lemma 4.13)

Kap. 4: Polygon-Triangulation

- Theorem 4.16: In jeder Triangulation eines einfachen Polygons mit $n \geq 4$ Ecken, gibt es mindestens zwei Dreiecke, deren Rand nur von einer Diagonale begrenzt wird.
  (Zählargumente Dreiecke, Diagonalen!)
- Strukturelle Aussagen: Triangulation von P mit $|P| = n$ hat $n-2$ Dreiecke und $n-3$ Diagonalen!
  (Induktion, Ohr abspalten!)
- Lemma 4.15: Der *duale Graph* T^* ist ein Baum mit Knotengrad ≤ 3.
  (Ohrensatz!)

Kap. 4: Triangulation Anwendung Drehungen!

Lemma 4.17: Sei P ein einfaches Polygon und e_i eine beliebige Kante, dann gilt: $\alpha_{i,i} = 2\pi$.



Beweis: Induktion, ein *Ohr* abspalten, Winkel zählen, Nullsumme!

Kap. 4: Triangulation Anwendung !

- Theorem 4.21: Ein einfaches Polygon P mit n Ecken kann stets mit $\lfloor \frac{n}{3} \rfloor$ Wächtern überwacht werden. Es gibt beliebig große Beispiele, wo diese Anzahl auch benötigt wird.

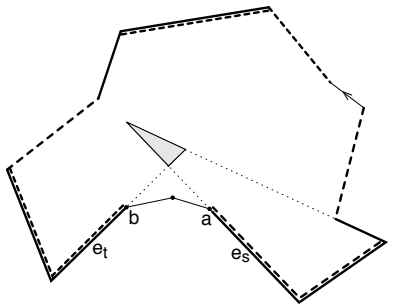
  (U. Schranke Beispiel! O. Schranke: Triang. 3-Färbbarkeit!)
- Definition: Kern Polygon, $\ker(P) := \{p \in P | \text{vis}_P(p) = P\}$
- Lemma 4.22: $\ker(P) = \bigcap_{\text{Kante } e \in P} H^+(e)$.

  (Halbebenenschnitt, Inneres, Beidseitige Inkl.)
- Theorem 4.26 Der Kern eines einfachen Polygones mit n Ecken kann in Zeit und Platz $O(n)$ berechnet werden.

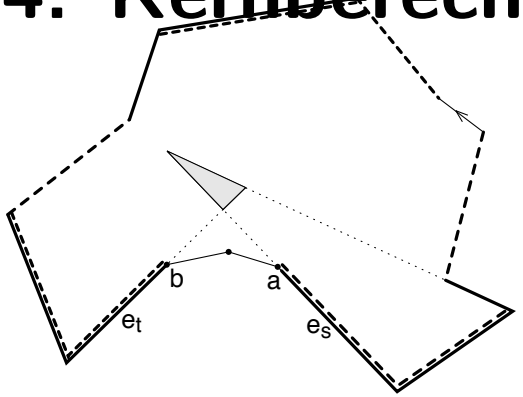
  (Max. Drehwinkelfolge, Folgen F und B, Halbebenen nach Steigung!)

Kap. 4: Kernberechnung!

- $\alpha_{\max} := \max_{i \neq j} \alpha_{i,j} = \alpha_{s,t}$
  Maximale Drehwinkelfolge, auch $\alpha(e, f)$
- Lemma 4.23: Falls der maximale Drehwinkel $\alpha_{\max}$ von P größer gleich 3π ist, ist der Kern leer.
  (Kantenfolge mit entsprechenden $H^+(e)$)
- Korollar 4.24 Sei der maximale Drehwinkel $\alpha_{\max}$ von P kleiner gleich 3π ist, dann gilt für je zwei Kanten e_i, e_j: $-\pi < \alpha(e_i, e_j)$.
  (Gesamtsumme!)



Kap. 4: Kernberechnung!



- Aus maximaler Teilfolge: $e_s, e_{s+1}, \ldots, e_t$ und $e_s, e_{s+1}, \ldots, e_t$, Kantenfolgen F und B auswählen!
- Theorem 4.25: Sei P ein einfaches Polygon mit maximalem Drehwinkel $< 3\pi$. Dann gehört jede für den Kern wesentliche Kante von P zur Folge F oder zur Folge B.
- Folgen B und F sind jeweils nach Steigung sortiert! Dualität! Danach Schnitt zweier konvexer Mengen

Kap. 1: Graphengrundlagen

- Definition, Schlingen, Mehrfachkanten, Geometrische Realisation, Planarer Graph, Äquivalente Realisation, Bezeichnungen
- Theorem 1.1: Sei G ein kreuzungfreier geometrischer Graph im $\mathbb{R}^2$. Dann gilt: v-e+f=c+1. (Strukt. Ind.! Graphen aufbauen!)
- Korollar 1.2: Sei G ein kreuzungfreier geometrischer Graph, dessen Knoten den Grad ≥ 3 haben. Es gilt:

$$\begin{aligned} v &\leq \frac{2}{3}e \\ v &\leq 2(f-c-1) < 2f \\ e &\leq 3(f-c-1) < 3f. \end{aligned}$$

Der Rand einer Fläche hat im Mittel ≤ 6 Kanten.
(Zählargumente und Eulerformel!)

Kap. 1: Graphengrundlagen

- Nichtleerer, kreuzungsfreier, zusammenhängender, geometrischer Graph $G = (V, E)$: Dualer Graph G^*
- Korollar 1.3: G kreuzungsfrei, nichtleer und schlicht, Grad der Knoten ≥ 3: Dann ist $f \leq \frac{2}{3}e$ und $f < 2v$.
  (1. Teil direkt! 2. Teil Dualer Graph und Korolar 1.2)
- K_5 ist nicht planar!
  ($v = 5$, $e = 10$, $c = 1$, dann $f =?$, Formel: $v - e + f = c + 1$, schlicht! $3f = 21 \not\leq 20 = 2e$!)

Kap. 5: Voronoi Diagramm

- $VR(p,S) = \bigcap_{q \in S \setminus \{p\}} D(p,q)$

  Alle Punkte, die näher an p liegen als an jedem anderen Punkt aus S
- $VR(p,S)$ ist offene Menge, Rand gehört nicht dazu
- Voronoi-Diagramm: $V(S) := \mathbb{R}^2 \setminus \{VR(p,S) | p \in S\}$
- Das was stehen bleibt: Graph $G = (V,E)$
- Schnitt von Halbebenen: Jede Region ist konvex!
- Randstück-Bezeichnung: Voronoi-Kante (Bisektor), Voronoi-Knoten (Übergang)
- Insgesamt Wabenmuster: Planarer Graph, bestehend aus Bisektorstücken

Kap. 5: Voronoi Diagramm

- Lemma 5.1: Sei x ein Punkt in der Ebene, und sei $C(x)$ der sich von x ausbreitende Kreis. Dann gilt:

$C(x)$ trifft zuerst nur auf p	$\Longleftrightarrow$	x liegt in Vor.-Reg. von p
$C(x)$ trifft zuerst nur auf p, q	$\Longleftrightarrow$	x liegt auf Vor.-Kante zwischen Reg. von p u. q
$C(x)$ trifft zuerst genau auf $p_1, \ldots, p_k$ mit $k \geq 3$	$\Longleftrightarrow$	x ist Vor.-Knoten, Reg. von $p_1, \ldots, p_k$ grenzen an

Kap. 5: Voronoi Diagramm

- Lemma 5.2: Genau dann hat ein Punkt $p \in S$ eine unbeschränkte Voronoi-Region, wenn er auf dem Rand der konvexen Hülle von S liegt.

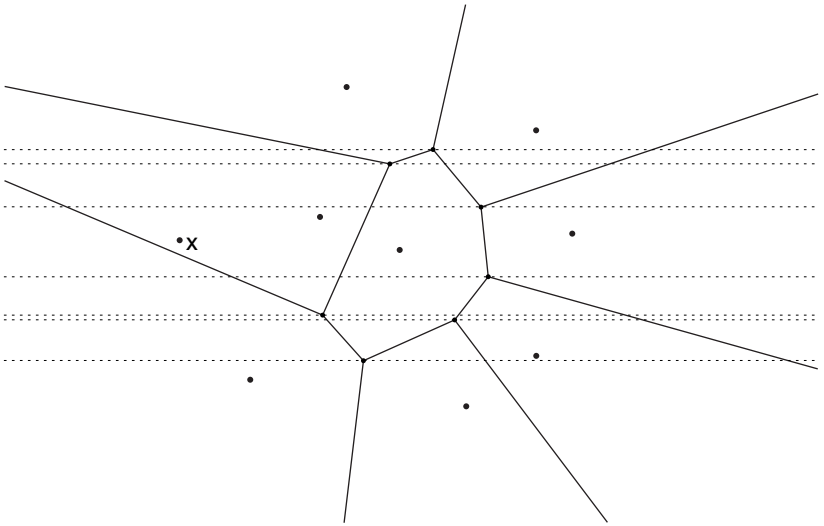
  (Zwei Richtungen! Lemma 5.1 verwenden)
- Theorem 5.3: Das Voronoi-Diagramm einer Menge von n Punkten in der Ebene hat $O(n)$ viele Knoten und Kanten. Im Mittel hat jede Fläche höchstens 6 Kanten.

  (Grad$(v) \geq 3$, Korollar 1.2, für $V_0(S)$ und $V(S)$)
- Theorem 5.4: Aus dem Voronoi-Diagramm $V(S)$ läßt sich in linearer Zeit die konvexe Hülle von S bestimmen.
- Korollar 5.5: Die Berechnung des Voronoi-Diagramms $V(S)$ von n Punkten hat Zeitkomplexität $\Omega(n \log n)$.

Kap. 5: Voronoi Diagramm

- Theorem 5.6: Zu n Orten in der Ebene kann mittels des
- Voronoi-Diagrammes eine Datenstruktur aufgebaut werden, die für jeden Anfragepunkt, den nächstgelegenen Ort in $O(\log n)$ bestimmen läßt.

Streifenmethode



Weitere Anwendungen

- Lemma 5.7: Sei $S = P \cup Q$ eine Zerlegung der endlichen Punktmenge S in zwei disjunkte, nicht-leere Teilmengen P und Q. Seien $p_0 \in P$ und $q_0 \in Q$ so gewählt, dass $|p_0q_0| = \min_{p \in P,\, q \in Q} |pq|$ gilt. Dann haben die Regionen von p_0 und q_0 im Voronoi-Diagramm $V(S)$ eine gemeinsame Kante.
  (Lokal, Dreiecksungleichung! Mittelpunkt auf Bisektor!)
- Korollar 5.8: Jeder nächste Nachbar von p in S sitzt im Voronoi-Diagramm in einer Nachbarzelle, d.h. in einer Voronoi-Region, die mit $VR(p, S)$ eine gemeinsame Kante besitzt.
  ($\{p\}$ und $S \setminus \{p\}$ als Mengen, Lemma 5.7)
- Theorem 5.9: Ist das Voronoi-Diagramm $V(S)$ vorhanden, kann in Zeit $O(n)$ für alle $p \in S$ der nächste Nachbar bestimmt werden.
  (Durchlaufen des Diagramms (z.B. DFS)!)

Weitere Anwendungen

- Minimum Spanning Tree: Kruskal: $O(|E| \log |E|)$ Laufzeit
  Vollst. Graph mit n^2 Kanten: $O(n^2 \log n)$
- Theorem 5.12: Der Rundweg um einen MST ist weniger als doppelt so lang wie eine optimale TSP Tour.
  ($|\text{MST}| \leq |\text{TSP-Opt} \setminus \{e\}| \leq |\text{TSP-Opt}|$)
- Theorem 5.11; Falls das Voronoi Diagramm $V(S)$ für n-elementige Punktmenge S gegeben ist, kann der MST in $O(n \log n)$ berechnet werden.
  (Lemma 5.7, Ablauf Kruskal, Nur *Nachbarn*! $O(n)$)

Delaunay Triangulation

- $VR(p, S)$ und $VR(q, S)$ gemeinsame Kante $\Longrightarrow$ pq heißt *Delaunay* Kante
- Menge aller Delaunay Kanten: Delaunay Zerlegung $DT(S)$
- Lemma 5.18: Drei Punkte p, q, r aus S bilden genau dann ein Dreieck $tria(p, q, r)$ von $DT(S)$, wenn der eindeutig bestimmte Kreis $UK(p, q, r)$ durch p, q, r keinen anderen Punkt aus S im Inneren enthält.

  (Lemma 5.1 Kuchenstück!)
- Theorem 5.17 Sei S eine Menge von Punkten in allgemeiner Lage. Die Delaunay Triangulation von S hat unter allen Triangulationen von S die größte Winkelfolge hat.

  (Andere Triangulation! Umkreiseigenschaft! Lokal verbessern!)

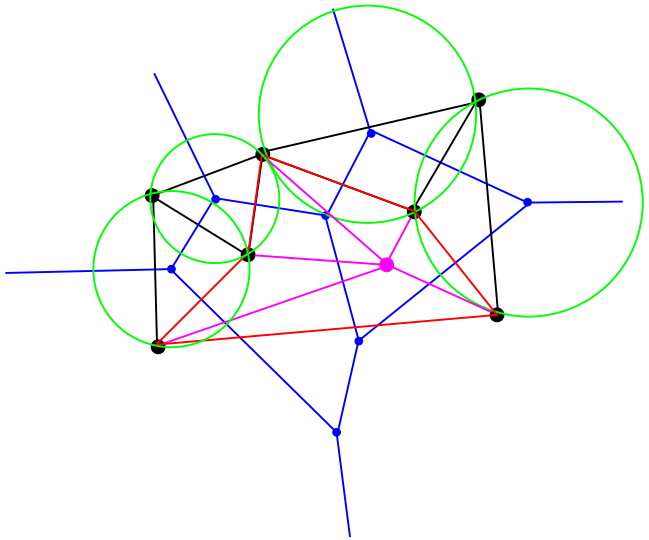
DT: Randomisierte Inkrementelle Konstruktion

- Gesamtergebnis: Die Delaunay Triangulation von n Punkten kann in erwarteter Laufzeit $O(n \log n)$ mit erwartetem linearen Speicherplatz konstruiert werden.
- Theorem 6.9: Durch die Verwendung des Delaunay DAGs kann in mittlerer Zeit $O(\log i)$ ein Punkt p_i in DT_{i-1} eingefügt werden, falls jede Reihenfolge von $p_1, p_2, \ldots, p_n$ gleich wahrscheinlich ist. Der Speicherplatzbedarf von DAG_i ist im Mittel linear.

DT: Randomisierte Inkrementelle Konstruktion

Sukzessive p_i in DT_{i-1} einfügen

1. Lokalisiere p_i in DT_{i-1}
2. Bestimme Ausgangs-Stern von p_i
3. Sukzessive Edge-Flips durchführen

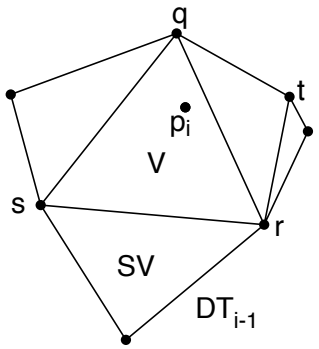


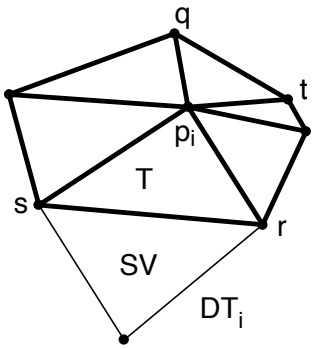
DT: Randomisierte Inkrementelle Konstruktion

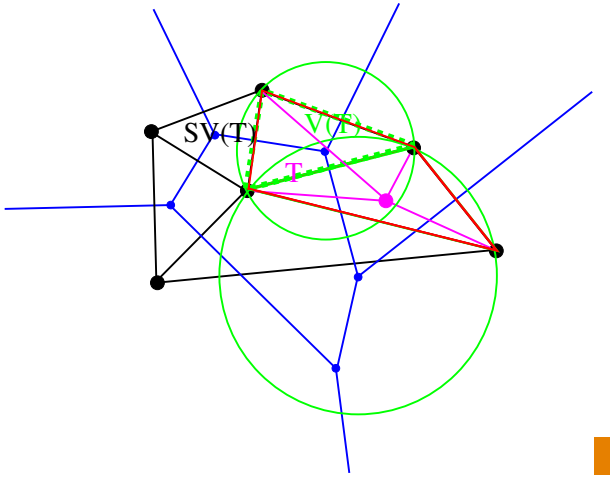
- Lemma 6.3: Sei k_i die Anzahl der Dreiecke von DT_{i-1}, die mit p_i in Konflikt stehen. Sei d_i der Grad von p_i in DT_i. Dann gilt: $k_i + 1 \leq d_i \leq k_i + 2$.
  ($p_i \notin \mathsf{ch}(S_{i-1})$: $k_i + 2 = d_i$
  $p_i \in \mathsf{ch}(S_{i-1})$: $k_i + 1 = d_i$)
- Lemma 6.4: Falls das Dreieck T aus DT_{i-1}, das p_i enthält, bekannt ist, kann DT_i in Zeit $O(d_i)$ berechnet werden.
  (Einfach: $O(i)$ und dann $O(d_i)$ Edge-Flips)
- Korollar 6.5 $\mathsf{DT}(S)$ für $|S| = n$ kann in $O(n^2)$ berechnet werden.

Inkrementelle Berechnung: Verbesserung

- Lokalisation: Finde p_i in DT_{i-1}
- Verbessern durch Datenstruktur: Delaunay-DAG
- Historie aller Dreiecke, die entstanden sind
- Beispiel: Vater/Stiefvater Beziehung, entlang Kante sr
- T neu DT_i, V Vater verantwortlich, SV Stiefvater angrenzend (Stern)

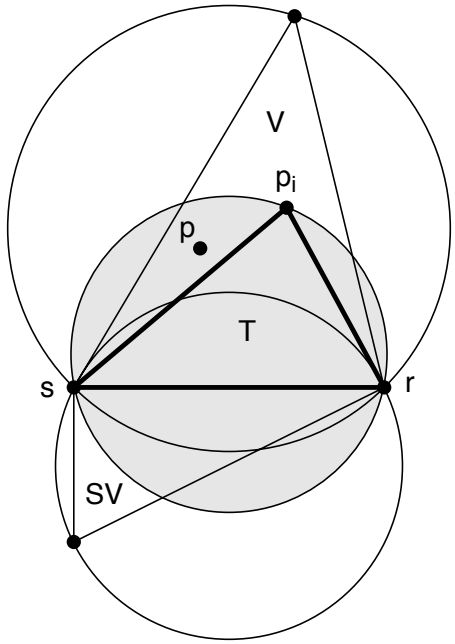






DT: Randomisierte Inkrementelle Konstruktion

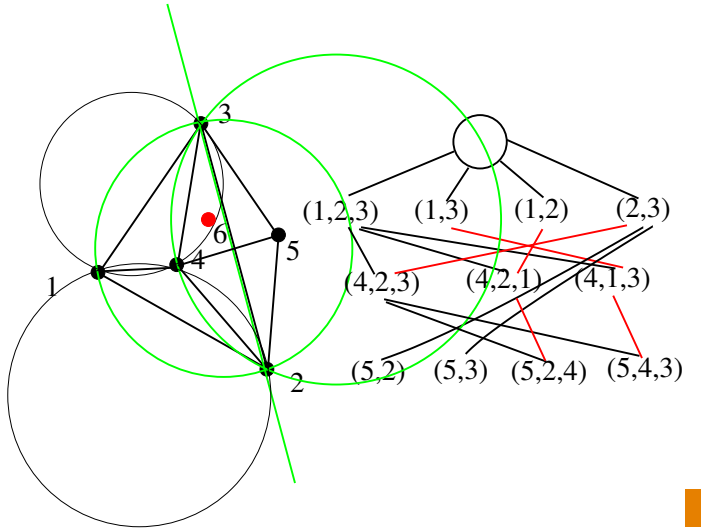
- Vater/Stiefvater Konflikte!
- Lemma 6.6: Jeder Punkt $p \notin S_i$, der mit dem Dreieck tria(s, r, p_i) in Konflikt steht, steht entweder mit dem Vater oder dem Stiefvater in Konflikt.



DT: Randomisierte Inkrementelle Konstruktion

- Korollar 6.7: Sei T ein Delaunay Dreieck in $\mathrm{DT}_3, \mathrm{DT}_3, \ldots, \mathrm{DT}_{i-1}$, das mit p_i in Konflikt steht. Dann gibt es in DAG_{i-1} einen gerichteten Weg von der Wurzel nach T, der nur Dreiecke enthält, die mit p_i in Konflikt stehen.

  (Folgerung Lemma 6.6, Konflikt Vater oder Stiefvater, sukzessive zurückverfolgen)



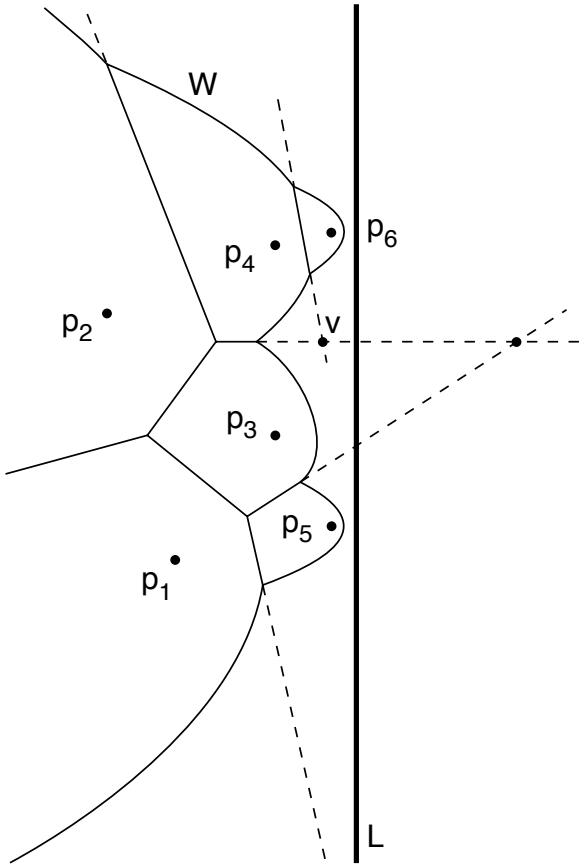
DT: Randomisierte Inkrementelle Konstruktion

- Lemma 6.8: Die Verwendung eines Delaunay-DAG macht es möglich, einen neuen Punkt p_i in Zeit $O(m_i)$ in die Delaunay Triangulation DT_{i-1} einzufügen. Dabei bezeichnet m_i die Anzahl der Dreiecke, die in $DT_3, \ldots, DT_{i-1}$ vorkommen (einfach gezählt) und deren Vater oder Stiefvater mit p_i in Konflikt steht.

  (Abschätzung m_i randomisierter Eingabefolge!
  Eingabereihenfolge gleich wahrscheinlich unter allen!
  Rückwärts: p_j mit WS $\frac{1}{j}$ aus $\{p_1, \ldots, p_j\}$)
- Theorem 6.9: Durch die Verwendung des Delaunay DAGs kann in mittlerer Zeit $O(\log i)$ einen Punkt p_i in DT_{i-1} einfügen, falls jede Reihenfolge von $p_1, p_2, \ldots, p_n$ gleich wahrscheinlich ist. Der Speicherplatzbedarf von DAG_i ist im Mittel linear.

Voronoi Konstruktion mit Sweep

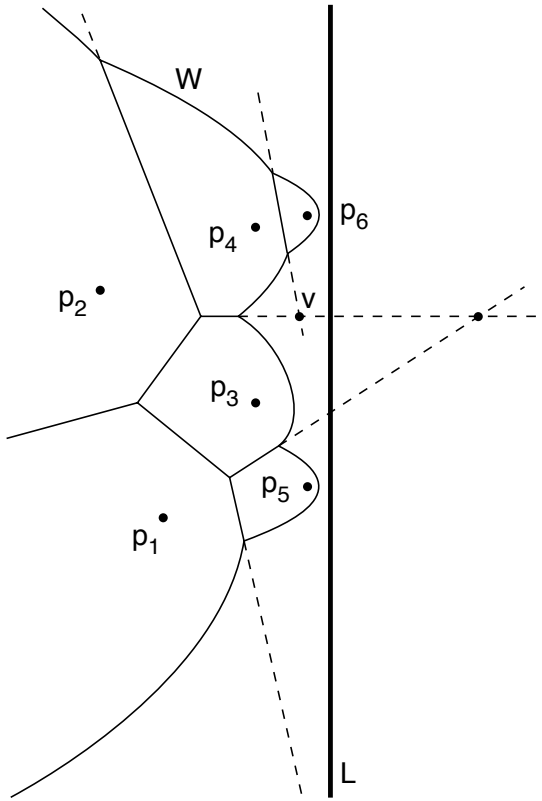
- SSS Wellenfront: Mehrere Stücke
- $B(p_j, L)$ aneinandergereiht
- Links der Wellenfront:
  $VR(\{p_1, \ldots, p_{i-1})$ fertig
- Gebiet rechts der Wellenfront gehört L
- Zwischen den Parabeln:
  Spikes, Bisektoren zwischen Punkten
- Im weiteren Verlauf: Voronoi Knoten



Ereignisse des Sweeps

Wann verändert sich die Wellenfront? Nur diskrete Zeitpunkte!

1. Parabelstück kommt hinzu: Ein neuer Punkt wird getroffen! Punkt-Ereignis, neue Parabel entsteht
2. Parabelstück verschwindet: Die Welle erreicht den Schnittpunkt zweier Spikes, Spike-Ereignis, Voronoi-Knoten entsteht



Voronoi Konstruktion mit Sweep

- Lemma 6.10 Die Wellenfront ist zusammenhängend und Y-monoton.

  (Eigenschaften der Parabeln!
  Datenstruktur: Baum mit dynamischen Schlüsseln)
- Theorem 6.11 Das Voronoi Diagramm von n Punkten läßt sich mit dem Sweep Algorithmus in Zeit $O(n \log n)$ und mit Platz $O(n)$ berechnen, das ist optimal.

  ($O(n)$ Events in je $O(\log n)$ Zeit
  Korrektheit: Algorithmus schreibt Voronoi-Diagramm links der Wellenfront *in den Sand*, Kompl. Wellenfront DSS ,
  Events: Platzsparregel)

Voronoi Liniensegmente

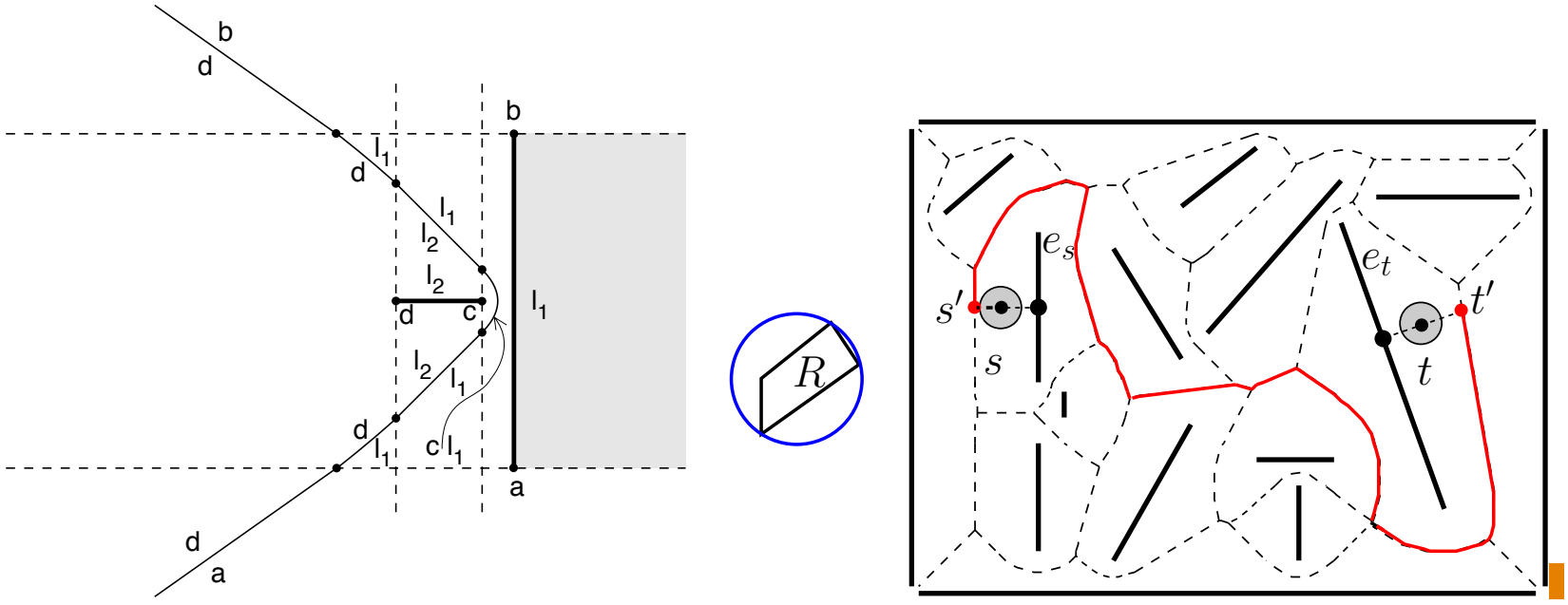
- Lemma 5.24: Der Bisektor von zwei disjunkten Liniensegmenten l_1 und l_2 ist eine Kurve aus Parabelstücken, Liniensegmenten und zwei Halbgeraden.

  (Verantwortungsbereiche)
- Lemma 5.25: Sei S eine Menge von Liniensegmenten und $l \in S$. Für jeden Punkt x in der Voronoi-Region VR(l, S) gilt: Das Liniensegment xy_x zwischen x und dem Punkt $y_x \in l$ der am nächsten zu x liegt, liegt in VR(l, S).

  (Sternförmig! Widerspruch Dreiecksungleichung!)
- Korollar 5.26: Die Voronoi-Regionen von Liniensegmenten sind zusammenhängend.

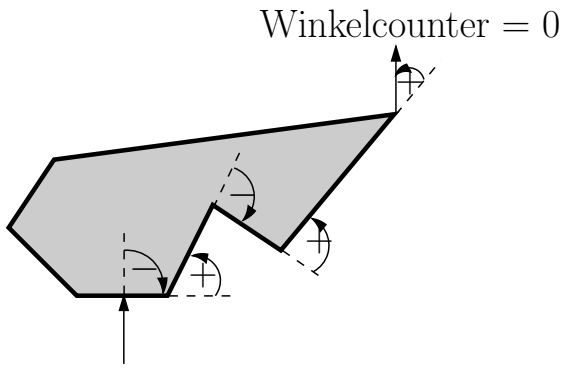
Bisektor: Maximal 7 Stücke! Anwendung!

- Lemma 5.27: Der Bisektor von zwei disjunkten Liniensegmenten l_1 und l_2 ist eine Kurve aus Parabelstücken, Liniensegmenten und zwei Halbgeraden und besteht aus maximal 7 Stücken.
- Anwendung Bahnplanung



Kap 7: Motion Planning, Pledge Algorithmus

1. Wähle Winkel φ und drehe den Roboter in diese Richtung.
2. Gehe in Richtung φ, bis der Roboter ein Hindernis erreicht.
3. Drehe nach rechts und halte den Kontakt mit der Wand an der linken Seite des Roboters.
4. Folge der Wand und addiere dabei die Drehwinkel, bis der totale Drehwinkel Null ist, dann GOTO (2).



Beweis Korrektheit!

Theorem 7.1 Der Pledge-Algorithmus findet in jedem Labyrinth und von jeder Startposition einen Ausweg, falls überhaupt einer existiert.

- Falls der Agent keinen Ausweg finden kann, ist er eingeschlossen!
- Annahme: Roboter erreicht Rand nicht.
- **Lemma 7.3:** Pfad $\Pi_\circ$ stets aufs Neue
- **Lemma 7.4:** Hat keine Kreuzungen
- Zwei Orientierungen: 1) Im UZS 2) Gegen den UZS
- 2) stets $+2\pi$ pro Runde, wird positiv, Widerspruch **Lemma 7.2**
- Also 1) stets -2π pro Runde
- Irgendwann nur noch negativ, Hindernis wird nicht verlassen
- Orientierung: Im UZS $\Rightarrow$ Innenhof! Ausweglos!

Beweis Korrektheit! Hilfsaussagen!

- Lemma 7.2: Der Winkelzähler im Pledge-Algorithmus nimmt niemals einen positiven Wert an.

  (via Konstruktion!)
- Lemma 7.3: Falls der Roboter nicht ins Freie gelangt, durchläuft er — bis auf ein endliches Anfangsstück — stets den gleichen Weg aufs Neue.

  (Polygonale Kette und mögliche Knoten)
- Lemma 7.4: Annahme Roboter kann nicht entkommen. $\Pi_\circ$ stets aufs Neue. $\Pi_\circ$ ist kreuzungsfrei, darf berühren!

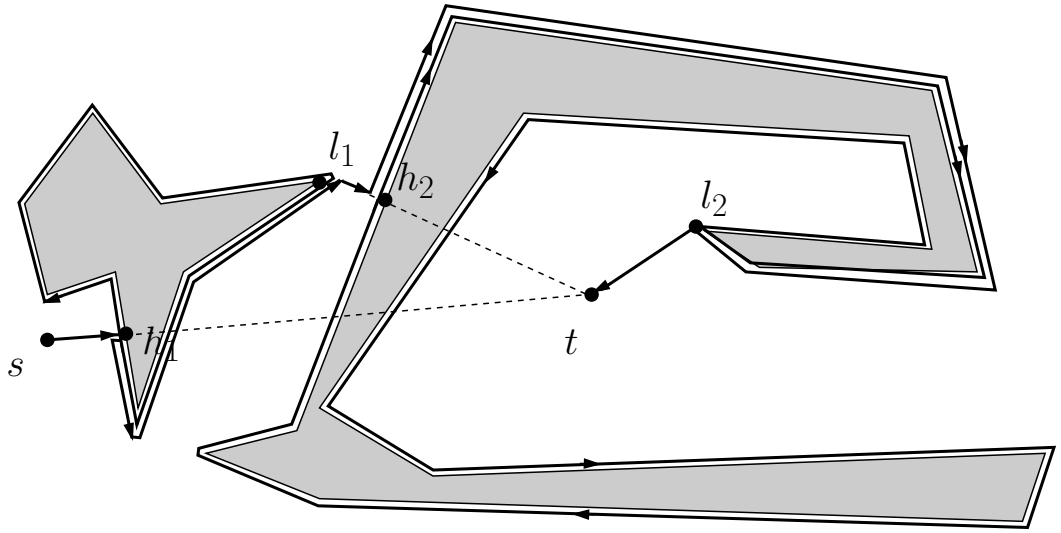
  (Spezieller Weg! Schnitt am Hindernis! Winkelvergleich!)

Zielpunktsuche Kompass: BUG Strategie

- Aktionen:

  1. Bewegung in Richtung zum Ziel
  2. Bewegung entlang des Randes

- Leave-Points l_i, Hit-Points h_i, Simpleste BUG Variante



Zielpunktsuche Kompass: BUG Strategie

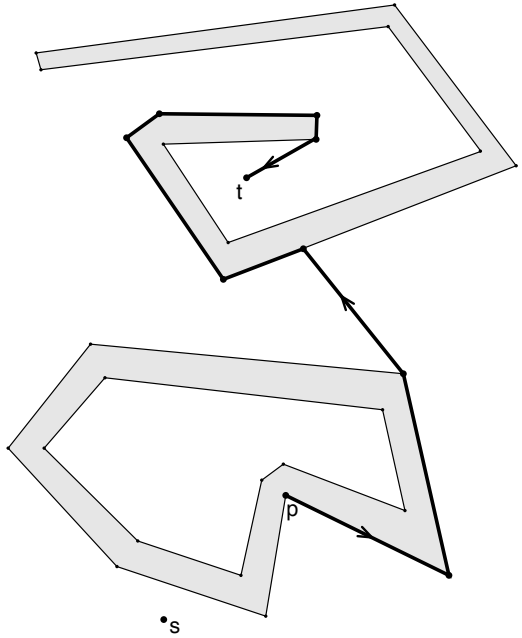
- Theorem 7.5: Die Strategie BUG findet einen Weg vom Startpunkt s zum Zielpunkt t, falls ein solcher Weg existiert.
  (Folge Leave/Hit, neues Polygon!)
- Theorem 7.6: Sei $\Pi_{\mathtt{Bug}}$ der Weg vom Start s zum Ziel t, den die Strategie Bug1 zurücklegt. Dann gilt $|\Pi_{\mathtt{Bug}}| \leq D + \frac{3}{2}\sum_i \mathtt{UP_i}$. (Umrundungen! Weg sukzessive abschätzen!)

Steuerwortkonstruktion!

Zielkompass und Tastsensor

Drei Steuerbefehle

1. T: Laufe in Richtung Ziel von Wandecke
2. L: Nächste Kante in gewohnte Richtung
3. R: Nächste Kante entg. gewohnte Richtung



Steuerwort-Beispiel:

$w(p) = L^2TR^5T$

Universelles Steuerwort!

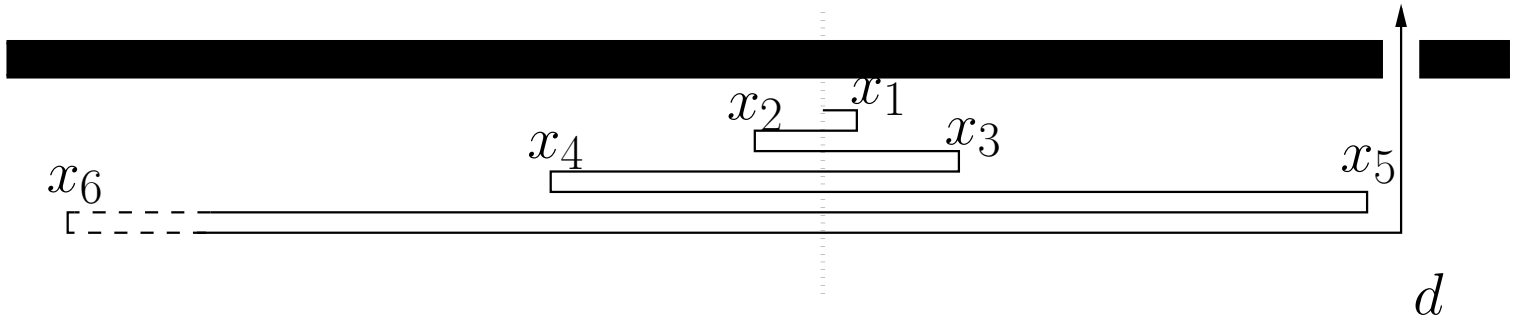
- Annahme: t gegeben, erreichbar, endliche Szene, endlich viele erreichbare Punkte $\{p_1, p_2, \ldots, p_m\}$
- Lemma 7.8: Es gibt ein universelles Steuerwort w über $\Sigma = \{T, L, R\}$, das den Agenten von jedem Startpunkt p_i zum Ziel t führt, falls es einen Weg zum Ziel t gibt.
  (Worte sukzessive erweitern für $p_1, p_2, \ldots$)
- Theorem 7.7: Im Prinzip genügen Zielkompass und Tastsensor, um in unbekannter Umgebung einen Zielpunkt zu finden.
  (Alg: Alle Wörter wachsender Länge sukzessive anwenden, irgendw. ist w_m am Zug!)

Kompetitive Strategien

- **Def. Kompetitiver Faktor**: Π Online-Problem und S Strategie, die jede Instanz $P \in \Pi$ korrekt löst. $K_S(P)$ die Kosten, die S verursacht und $K_{\text{OPT}}(P)$ die Kosten einer optimalen Offline-Lösung von P. Dann heißt S C–kompetitiv, falls es $C, A > 0$ gibt, so dass für alle $P \in \Pi$ gilt: $K_S(P) \leq C \cdot K_{\text{OPT}}(P) + A$.
  Dabei wird C als der *kompetitive Faktor* bezeichnet.
- Beispiel Bin Packing:
  - Fülle aktuelles Q_i in den ersten möglichen Behälter
  - Neuer Behälter, falls Q_i nicht passt!

Kompetitive Strategien

- Theorem 7.9: Die Strategie FirstFit verwendet maximal doppelt soviele Behälter wie die optimale Strategie und somit einen kompetitiven Faktor von 2. Es gibt beliebig große Beispiele mit einem Faktor von $\frac{5}{3}$.
  (OS: $\frac{1}{2}(m-1) < \lceil \sum_{i=1}^{n} Q_i \rceil \leq$ OPT
  US: $6n : 0.15, 6n : 0.34, 6n : 0.51$: OPT $= 6n$, FirstFit $10n$)
- 2-Wege Suche: Worst-Case, bei d verpasst, nochmal zurück! Finde Strategie, mit: $\sum_{i=1}^{k+1} 2x_i + x_k \leq Cx_k$



2-Wege Suche

- Theorem 7.10: Die Strategie der abwechselnden Verdopplung der Suchtiefe hat einen kompetitiven Faktor von 9.

  (Analysiere $\frac{\sum_{i=1}^{k+1} x_i}{x_k} = \frac{2^{k+2}-2}{2^k} = 4 - \frac{2}{2^k}$)
- Strategie $x_i = 2^{i-1}$ Doublingstrategie optimal!
- Theorem 7.11: Die Strategie der abwechselnden Verdopplung der Suchtiefe hat den kleinstmöglichen kompetitiven Faktor.

  (Beweis: Anwendung des Theorems von Gal!)
- $F_k(f_1, f_2, \ldots) := \frac{\sum_{i=1}^{k+1} f_i}{f_k}$ für alle k
  Unimodal: $F_k(A \cdot X) = F_k(X)$ and
  $F_k(X + Y) \leq \max\{F_k(X), F_k(Y)\}$!

Theorem von Gal 1980/2000

Falls F_k die folgenden Bedingungen erfüllt:

i) F_k ist stetig,

ii) F_k ist unimodal: $F_k(A \cdot X) = F_k(X)$ und
$F_k(X + Y) \leq \max\{F_k(X), F_k(Y)\}$,

iii) $\liminf_{a \mapsto \infty} F_k\left(\frac{1}{a^k}, \frac{1}{a^{k-1}}, \ldots, \frac{1}{a}, 1\right) =$
$\liminf_{\epsilon_k, \epsilon_{k-1}, \ldots, \epsilon_1 \mapsto 0} F_k\left(\epsilon_k, \epsilon_{k-1}, \ldots, \epsilon_1, 1\right)$,

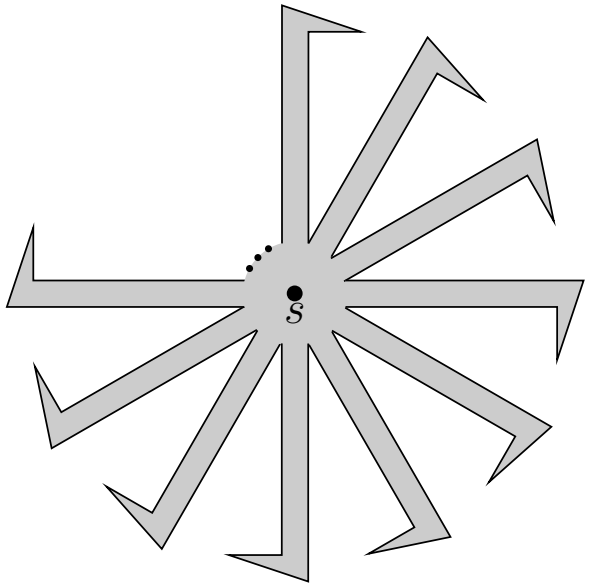
iv) $\liminf_{a \mapsto 0} F_k\left(1, a, a^2, \ldots, a^k\right) =$
$\liminf_{\epsilon_k, \epsilon_{k-1}, \ldots \epsilon_1 \mapsto 0} F_k\left(1, \epsilon_1, \epsilon_2, \ldots, \epsilon_k\right)$,

v) $F_{k+1}(f_1, \ldots, f_{k+1}) \geq F_k(f_2, \ldots, f_{k+1})$.

Dann gilt:$\sup_k F_k(X) \geq \inf_a \sup_k F_k(A_a)$ mit $A_a = a^0, a^1, a^2, \ldots$ und $a > 0$.

m-Wege Suche

- $2m-1$ gegenüber 1! Nicht komp,.
- Festes m, unendliche Strahlen!
- Tupel (f_j, J_j): Tiefe, nächster Besuch!
- Ann.: (f_j, J_j), $J_j = j+m$, $f_j \geq f_{j-1}$
- $F_k(f_1, f_2, \ldots) := \frac{f_k + 2\sum_{i=1}^{k+m-1} f_i}{f_k}$
- (Gal): optimal $a = \frac{m}{m-1}$:
  min $f_m(a) := \frac{a^m}{a-1}$
- Faktor: $C = 1 + 2m\left(\frac{m}{m-1}\right)^{m-1}$
  opt. $\frac{\sum_{i=1}^{k+m-1} a^i}{a^k} = \frac{a^{k+m}-1}{a^k(a-1)} - \frac{1}{a^k} = \frac{a^m}{a-1} - \frac{a}{a^k(a-1)} \mapsto f_m(a)$



m-Wege Suche

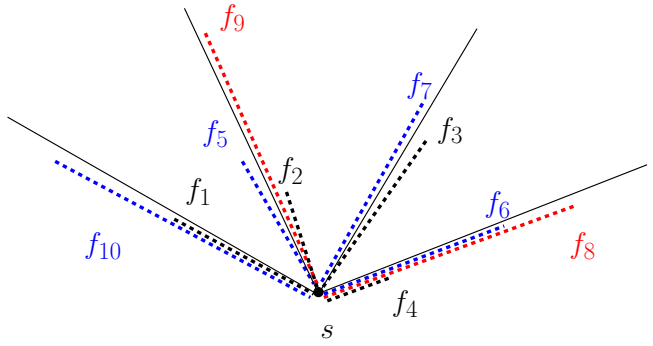
- Lemma: Es gibt eine optimale m-Wege Strategie $(f_1, f_2, \ldots)$, die die Strahlen in fester Reihenfolge und wachsender Tiefe besucht!

  (periodisch und monoton, also: (f_j, J_j), $J_j = j + m$, $f_j \geq f_{j-1}$
  Strategie ändern! Bedingungen erfüllen!)

Beispiel: $J'_{j+1} < J_j$, $f_1 > f_2$, $J'_2 = 5 < 10 = J'_1$

$\frac{\sum_{i=1}^{J'_k - 1} f'_i}{f'_k} \leq C$? $k = 1$, $J'_2 < J'_1$

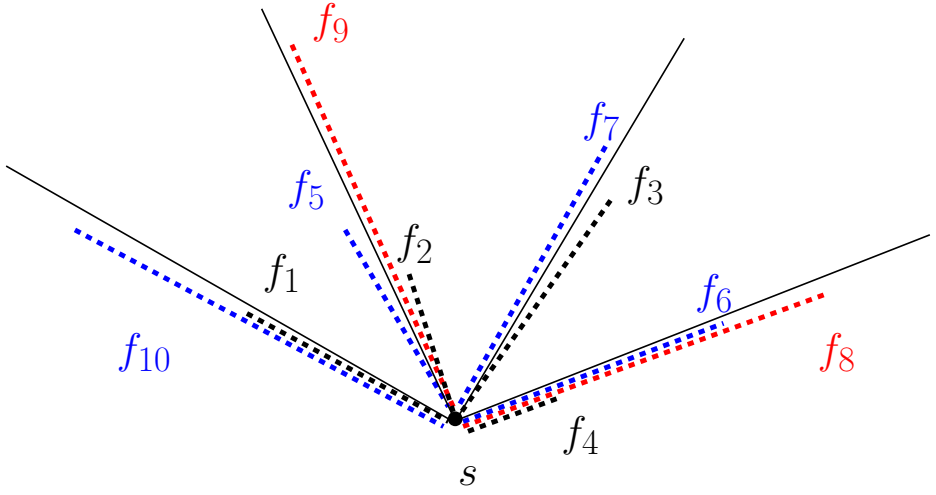
Abhilfe: Vertausche alle Besuche der beteiligten Strahlen ab Index j.



Beweis Monotonie!

$f_j > f_{j+1}$, $f'_j = f_{j+1}$, $f'_{j+1} := f_j$, $(J_k = J'_k)$: Bedingungen:

1. $\frac{\sum_{i=1}^{J'_k-1} f'_i}{f'_k} \leq C$: $k \neq j, j+1$, erfüllt! Vorher erfüllt!
2. $\frac{\sum_{i=1}^{J'_k-1} f'_i}{f'_k} \leq C$: $k = j+1$, erfüllt! $f'_{j+1} > f_{j+1}$, $J'_{j+1} = J_{j+1}$
3. $\frac{\sum_{i=1}^{J'_k-1} f'_i}{f'_k} \leq C$: $k = j$, nur erfüllt falls $J'_j < J'_{j+1}$



Beweis Periodisch!

$J_j > J_{j+1}$ auf Strahl K und L

Vertauche Besuche K und L ab $j+1$. Bedingungen:

1. $\frac{\sum_{i=1}^{J'_k-1} f'_i}{f'_k} \leq C$: $k \neq j, j+1$, erfüllt! Vorher erfüllt!
2. $\frac{\sum_{i=1}^{J'_k-1} f'_i}{f'_k} \leq C$: $k = j$, erfüllt! $J'_j < J_j$
3. $\frac{\sum_{i=1}^{J'_k-1} f'_i}{f'_k} \leq C$: $k = j+1$, erfüllt da $f_{j+1} \geq f_j$

Randomisierung

Gegenspielermodelle:

1. Kennt gewürfelte Strategie und Reihenfolge: Kein Unterschied!
2. Kennt Strategie, kennt Reihenfolge nicht! Gewürfelt!
   Analyse Doublingstrategie: Gegenspieler wählt Strahl und $x \in [2^k + \epsilon, 2^{k+1}]$! Erwartungswert des Komp. Faktors: 7!
3. Gegenspieler kennt Strategie UND Reihenfolge nicht!
   Gegenspieler wählt Strahl und Distanz!

Beweis zu 2: Gegenspielerwahl $x = 2^k + y$, $y \in \left[0, 2^k\right]$ und Seite!

$\frac{2^k+y+2\sum_{i=1}^{k} 2^i}{2^k+y} \leq 5$ und $\frac{2^k+y+2\sum_{i=1}^{k+1} 2^i}{2^k+y} \leq 9$: $y = \epsilon$ und $\frac{1}{2}(5+9) = 7$

Randomisierung: m-Wege

2. Kennt Strategie, kennt Reihenfolge der Besuche nicht! Gewürfelt!

Analyse Doublingstrategie: Gegenspieler wählt Strahl und $x = \left(\frac{m}{m-1}\right)^k + y$ mit $y \in \left[0, \left(\frac{m}{m-1}\right)^k\right]$

Faktor auf Strahl i: $\frac{\left(\frac{m}{m-1}\right)^k + y + 2\sum_{i=1}^{m+k-i}\left(\frac{m}{m-1}\right)^i}{\left(\frac{m}{m-1}\right)^k + y} \leq 1 + 2m\left(\frac{m}{m-1}\right)^{m-i}$

$$\frac{1}{m}\sum_{i=1}^{m} 1 + 2m\left(\frac{m}{m-1}\right)^{m-i} = \left(1 + 2m\left(\frac{m}{m-1}\right)^{m-1}\right) - 2(m-1)$$

$$= C_m - 2(m-1)$$

Randomisierung

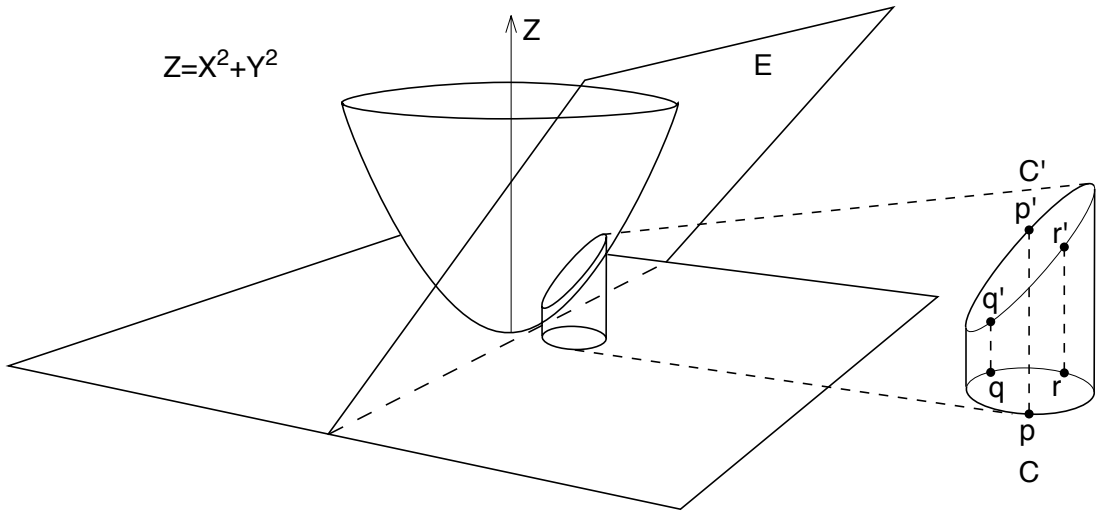
3. Gegenspieler kennt Strategie UND Reihenfolge nicht!
Gegenspieler wählt Strahl und Distanz!
Strategie m-Wege, (auch $m = 2$)

Wähle zufällige Permutation Π, wähle zufällig $\epsilon \in [0, 1)$
Verwende festes r, Setze: $d := r^{\epsilon}$
Suchtiefen: $f_i := r^{\epsilon+i}$, periodisch mit Π
$m_2 : r \approx 3.59$, $E(C) = r + 1 = 4.59\ldots$

Kao, Reif, Tate: Randomized Cow Path Problem, 1996

Projektion Punkte 2D nach 3D

- Paraboloid im $\mathbb{R}^3$: $P = \{(x, y, z) \in \mathbb{R}^3; x^2 + y^2 = z\}$
- Rotation der Parabel $\{Y^2 = Z\}$ um die Z-Achse
- Punkt $p = (x, y)$ projeziere $p' = (x, y, x^2 + y^2)$ auf P
- Auf Punktmengen S fortsetzen



Kreisprojektion

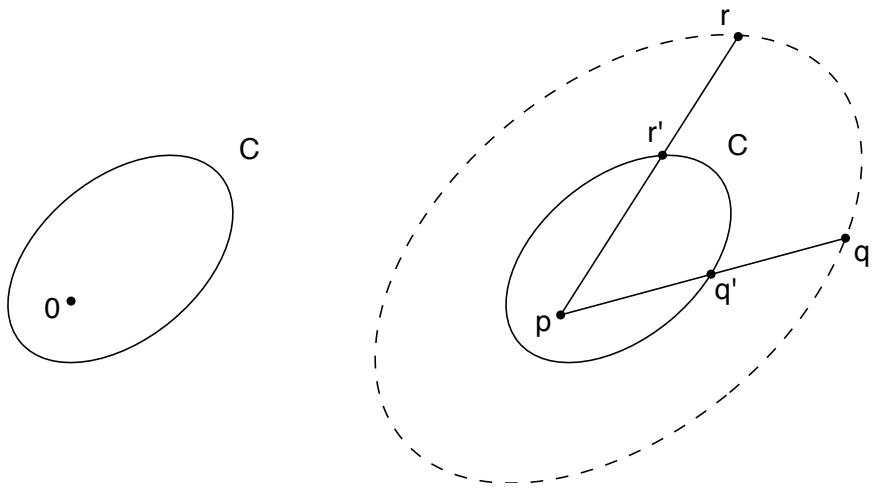
- Lemma 6.17: Sei K der Rand eines Kreises in der XY-Ebene.
- Dann ist die geschlossene Kurve K' auf dem Paraboloid P in einer Ebene des $\mathbb{R}^3$ enthalten.

  (Projektion des Kreises liegt in einer Ebene! Direkte Manipulation! Kreisprojektion!)
- Theorem 6.18: Sei S eine endliche Punktmenge in der XY-Ebene. Dann ist die Delaunay-Triangulation von S gleich der Projektion der unteren konvexen Hülle von S' auf die XY-Ebene.

  (Umkreis p, q, r entält keinen Punkt gdw. Dreieck p', q', r' liegt auf konvexer Hülle)

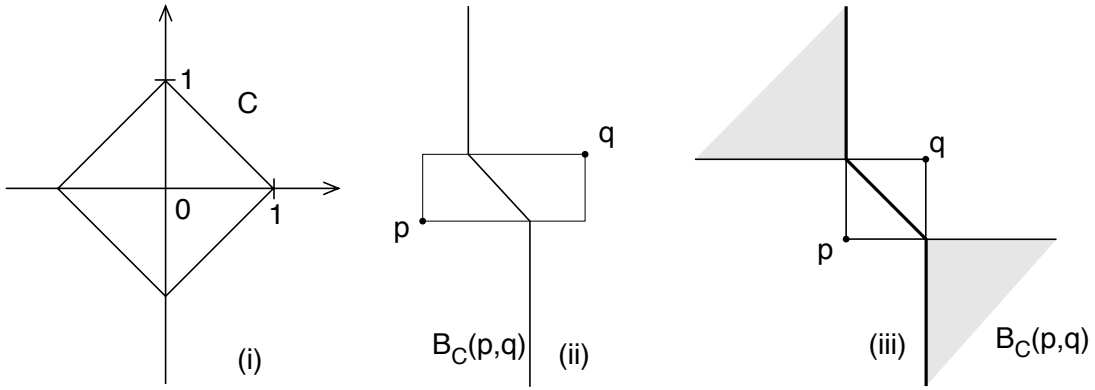
Definition: Konvexe Distanzfunktion

- *Einheitskreis* C kompakte, konvexe Menge im $\mathbb{R}^2$
- Nullpunkt in C enthalten
- Abstand von p nach q: C verschieben, Strahl von p durch q schneidet Rand von C bei q'
- $d_C(p,q) := \frac{|pq|}{|pq'|}$, Skalierungsfaktor für q



Andere Metriken

- Bisektor durch wachsende Kreise
- Nicht streng konvex $\Rightarrow$ flächige Bisektoren möglich
- L_1, L_∞



Konvexe Distanzfunktionen: Gemeinsamkeiten

- Lemma 5.20 Sei d_C eine konvexe Distanzfunktion. Dann ist jede
- Voronoi-Region $\mathrm{VR}_C(p, S)$ sternförmig und enthält p im Kern.
  (Widerspruch! Dreiecksungleichung!)
- Korollar 5.21 Jede Voronoi-Region bezüglich einer konvexen Distanzfunktion ist zusammenhängend.