Methoden der Offline Bewegungsplanung
Einführung

Elmar Langetepe
University of Bonn

Organisatorisches

- Bachelor BA-INF 124 Wahlpflicht, 4-6 Semester
  - Vorlesung 4 SWS, 5.5 LP, Übung 2 SWS, 3.5 LP
  - Prüfungen: vorauss. mündlich
  - Studienleistungen: Erfolgreiche Übungsteilnahme

Organisatorisches

- Übungen: Dienstags 12:00-14.00 Uhr, A7a
  Donnerstags 16:00-18.00, A301
- Wöchentlich, Beginn: 1.11
- Aufgabenblatt, montags! Blatt I: 24.10 (3 Aufgaben)
- Abgabe montags, Kasten Voyer
- Aufgaben, Lösungen schriftlich, Abgabe, Korrektur, Besprechung
- Bachelorarbeiten (Themenhinweise)
- Projektgruppe (Java,`www.geometrylab.de`)
- Folien/Skript online
- Mailingliste `https://lists.iai.uni-bonn.de/mailman/listinfo.cgi/vl-offline`

Ziele der Vorlesung

- Fachlich: Verständnis und Anwendung der typischen algorithmischen Ansätze und Modelle zur Beantwortung geometrischer Komplexitäts- und Optimierungsfragen in der Bewegungsplanung von Agenten unter vollständiger Information

- Integrative Schlüsselkompetenzen: Analysefähigkeiten und Adaptionsfähigkeiten, Entwicklung eigener Lösungsansätze

Einordung: Bewegungsplanung

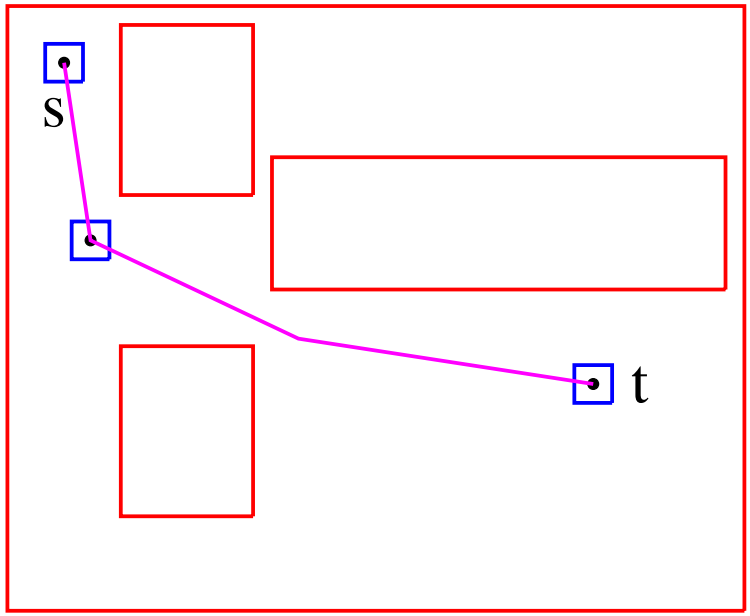
- Maschinen/Agenten
- Elektronik
- Mechanik
- Kontroll- und Regelungstechnik
- KI
- Softwareengeneering
- ⋮
- Lösungspläne: Algorithmik
- Geometrische Algorithmen

Geometrische Algorithmen

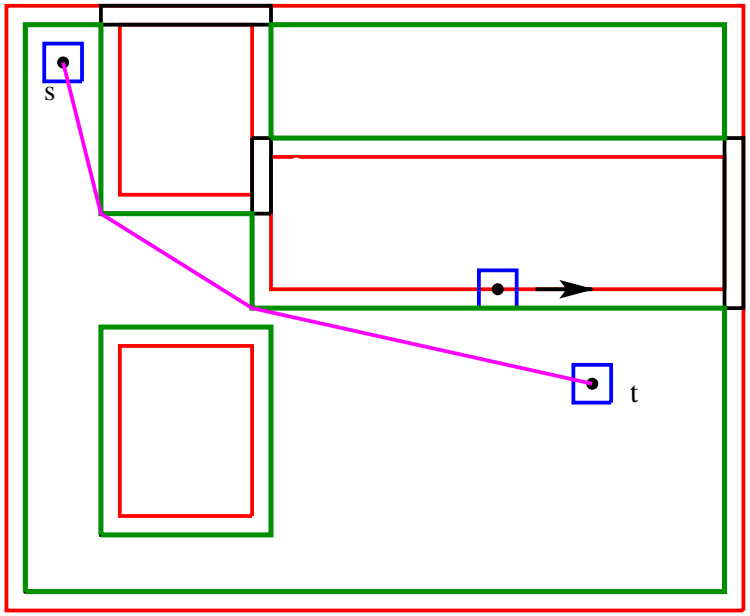
- Geometrie des Agenten und der Szene sind gegeben
- Löse Bahnplanungsproblem
- Unterscheidung: Online/Offline
- Fragen: Komplexität der Lösungen/Laufzeitkomplexität/Datenstrukturen
- Untere Schranke/Obere Schranke
- Strukturelle Eigenschaften
- Modularer Aufbau (Black Box)
- Viele Grundprobleme lösen
- Geometrische Algorithmen

Bsp: Translation Quadrat unter Rechtecken

- Achsenparalleles Rechteck von s nach t bewegen
- Achsenparallele Hindernisse
- Fragen: Geht das? Geringe Kosten? Berechnung?
- Idee: Referenzpunkt kollisionsfrei setzen



Idee: Translation Quadrat unter Rechtecken



Translation: Verschieben Referenzpunkt

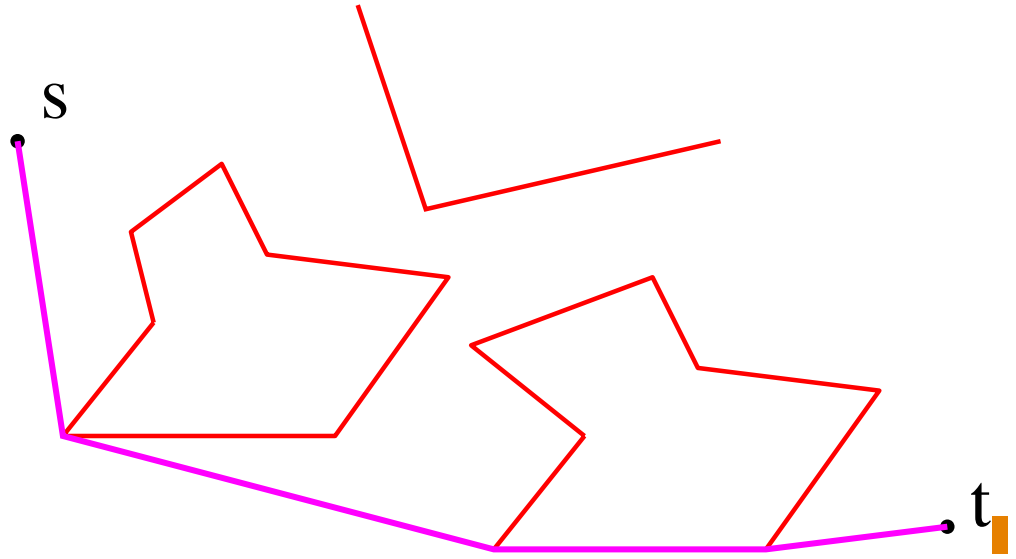
Aufblasen der Hindernisse

Vereinigung Sweep

Kürzester Weg????

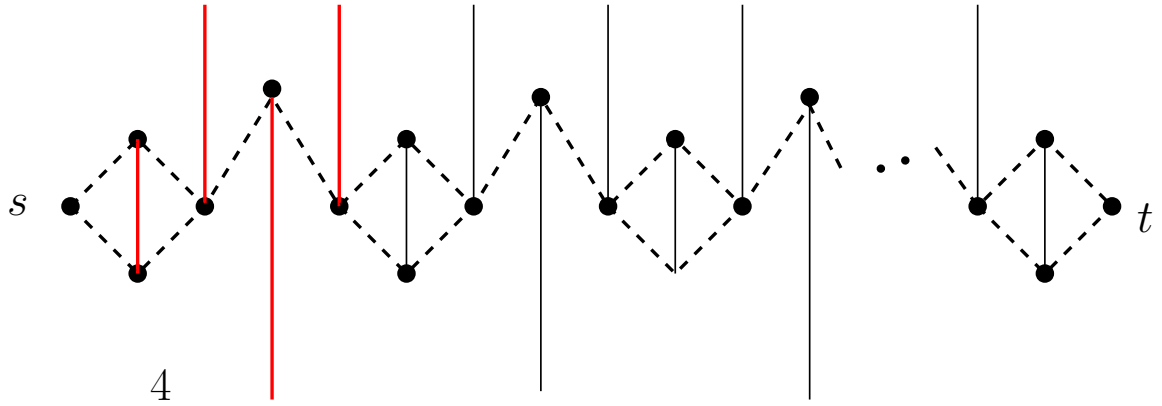
Kap 1.1: Kürzeste Wege, polygonale Hindernisse

- *Polygonale Hindernisse*: Definition
- Geschlossene Kantenzüge, offene Kantenüge
- Kantenfolge, Kanten schneiden sich nur an den Eckpunkten, nur aufeinanderfolgende
- *Polygonale Szene*: Hindernisse berühren sich nicht
- Wie berechnet man den kürzesten Weg?



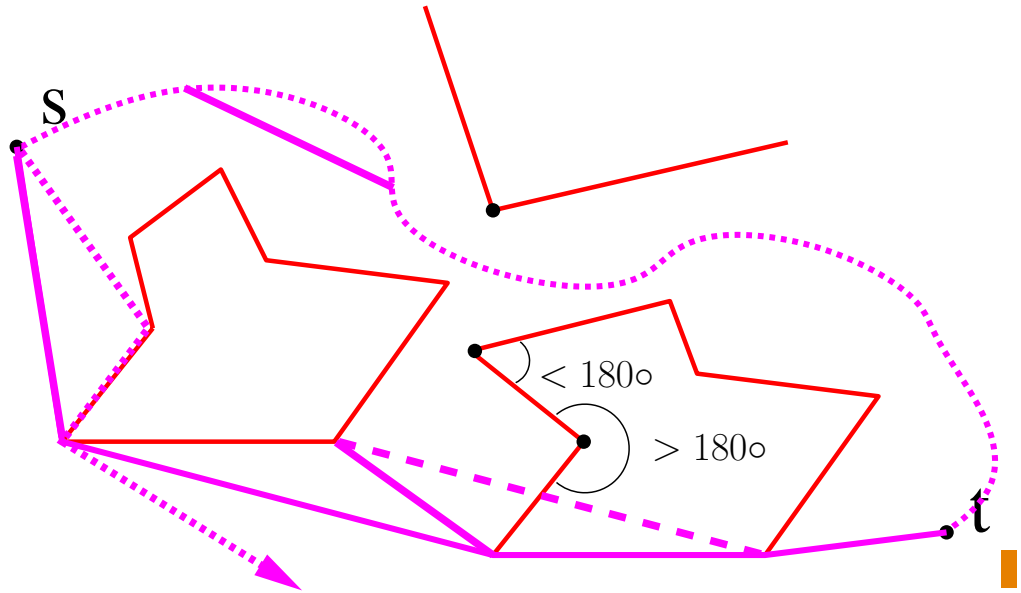
Eigenschaften Kürzester Wege: Anzahl

- m Kanten, exponentiell viele (in m) kürzeste Wege
- Genauer $h = 4m + 1$ Kanten, 2^{m+1} kürzeste Wege
- Pro Block verdoppeln, einmal verdoppeln am Ende



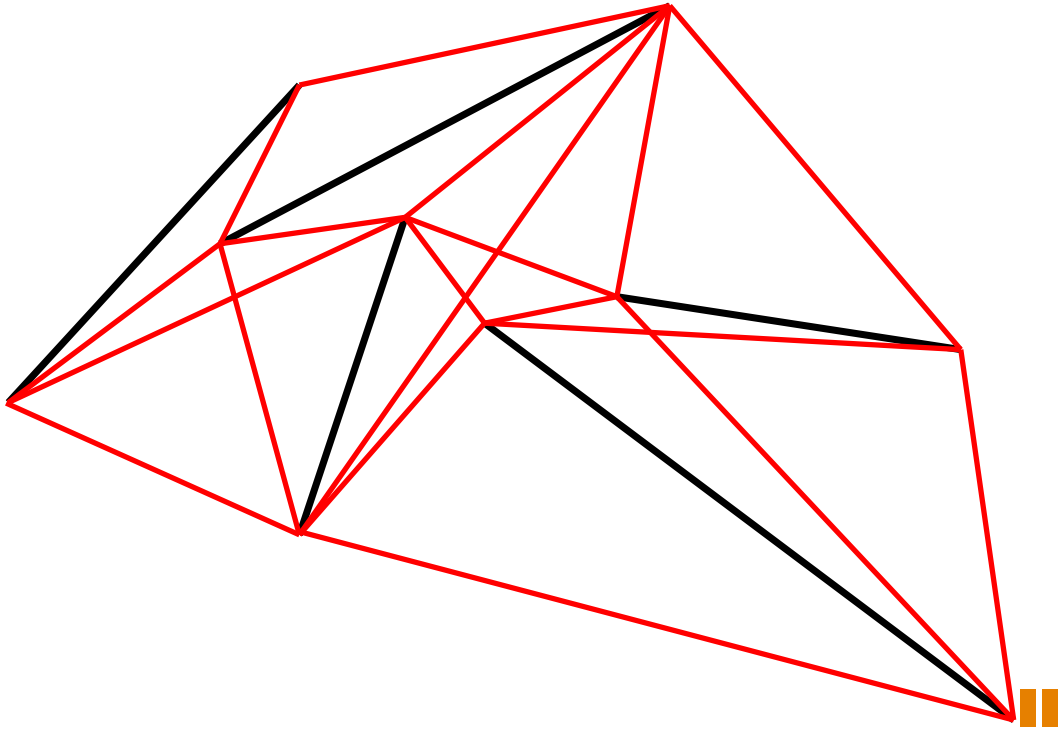
Kürzester Weg: Strukturelle Eigenschaften

- Polygonale Kette über Ecken, lokal abkürzen
- Nur über konvexe Ecken der Hindernisse
- Innenwinkel kleiner 180 Grad, lokal abkürzen
- Über gegenseitig *sichtbare* Ecken
- Verwendet nur solche Kanten, sonst abkürzen

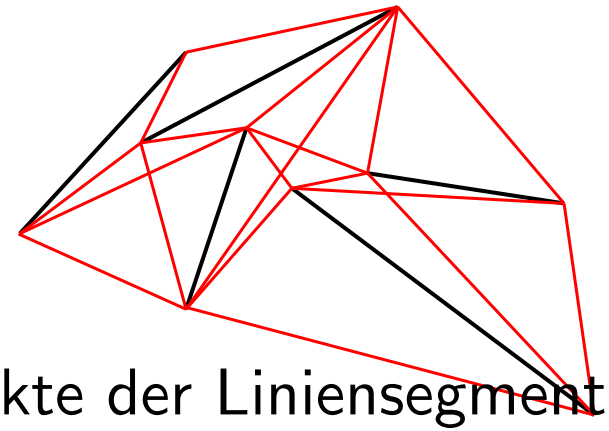


Allg.: Sichtbarkeitsgraph für Liniensegmente

- Menge L von nicht-schneidenden Segmenten gegeben
- Berechne die gegenseitig *sichtbaren* Segmente
- **Def. 1.1:** Sichtbarkeitsgraph von L: $\texttt{VisG}(L) = (V, E)$



Def. 1.1: Sichtbarkeitsgraph $\texttt{VisG}(L) = (V, E)$



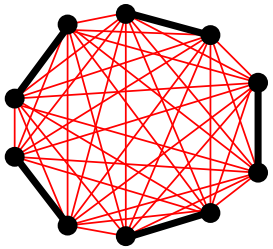
$V = \{$ alle Endpunkte der Liniensegmente in $L \}$

$E = \{ (p, q) | p, q \in V,\ \overline{pq}$ kreuzt kein Liniensegment aus $L \}$.

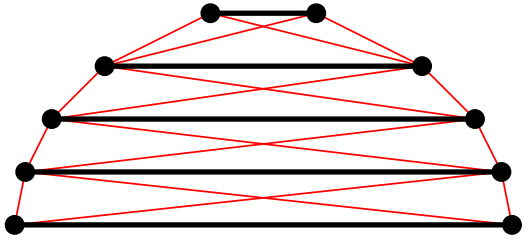
- Sichtbarkeitsgraph einer Menge von Polygonen P_i
- $\texttt{VisG}(L)$ mit $L = \{$ Kanten der $P_i \}$
- Entfernen der im Inneren der Polygone liegenden Sichtbarkeitskanten

Komplexität Sichtbarkeitsgraph $\texttt{VisG}(L) = (V, E)$

- n Liniensegmente
- $\Omega(n^2)$, $O(n^2)$, $\Theta(n^2)$
- Manchmal aber auch in $O(n)$
- Laufzeit Berechnung bestenfalls $O(n^2)$, oder Case-sensitiv
- Untere Schranke durch konkretes Beispiel
- Obere Schranke durch Nachdenken



(i)



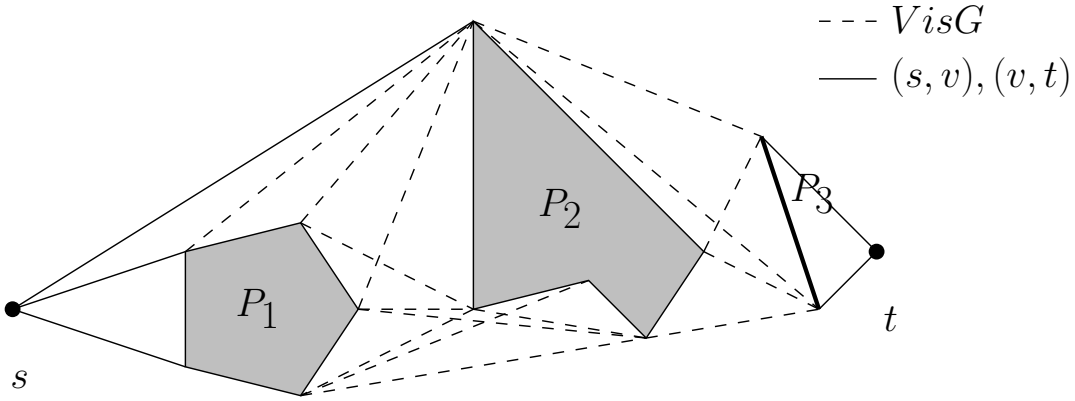
(ii)

Was bringt der Sichtbarkeitsgraph?

Alg. 1.1: Berechnung des kürzesten Weges mittels $\texttt{VisG}(P)$

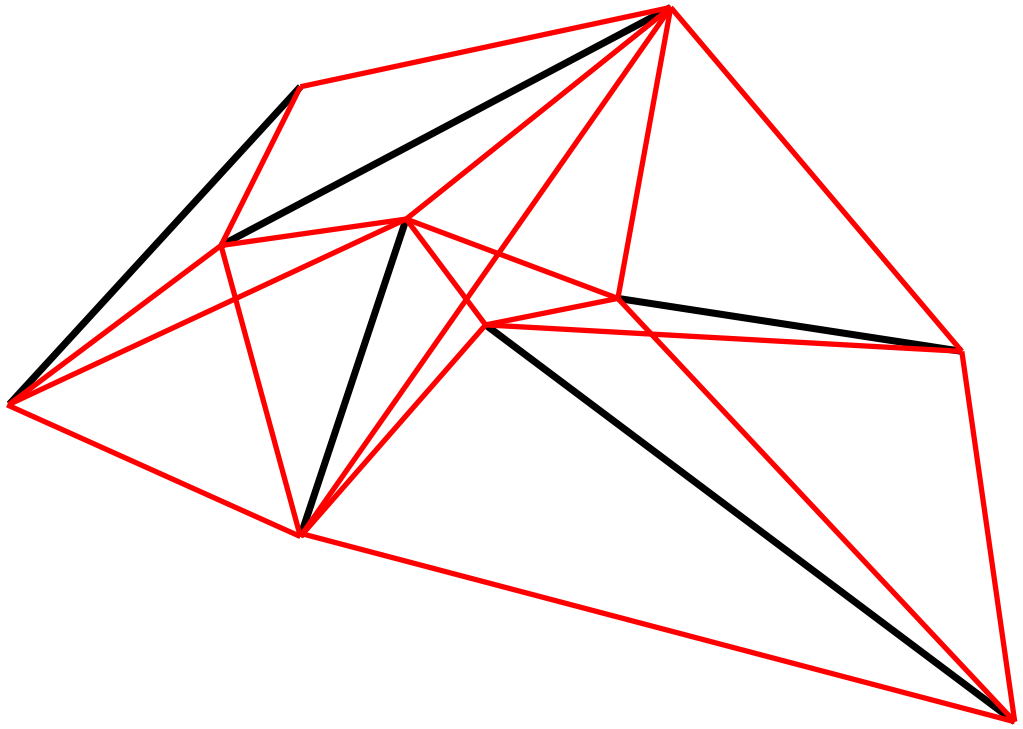
1. Sichtbarkeitsgraph (V, E) für Hindernisse: $O(n^2)$
2. Sichtbare Kanten von s aus: E_1, $O(n^2)$
3. Sichtbare Kanten von t aus: E_2, $O(n^2)$
4. All shortest path: Dijkstra auf $(V \cup \{s, t\}, E \cup E_1 \cup E_2)$

2. und 3.: Kanten mit den Ecken, Test auf Schnitt



Sichtbarkeitsgraph für Menge von Liniensegmenten

- Naiv: $O(n^3)$
- Single-Source Sweep: $O(n^2 \log n)$
- Simultaner Sweep: $O(n^2)$

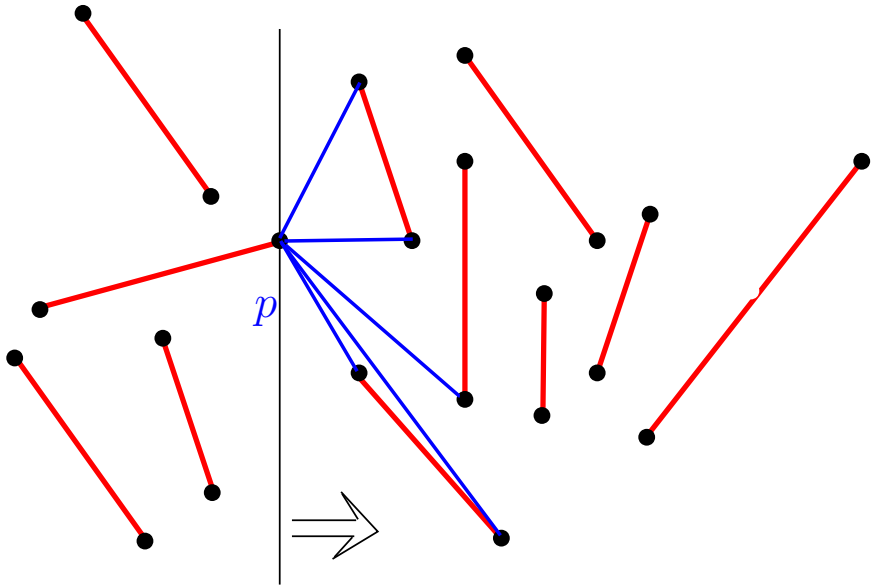


Sweep Technik

- Ausfegen der Ebene mit Sweepline
- Komplexitätsreduktion: 2-Dim nach 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
- Einfaches Beispiel: Punkte auf Linie, Closest pair

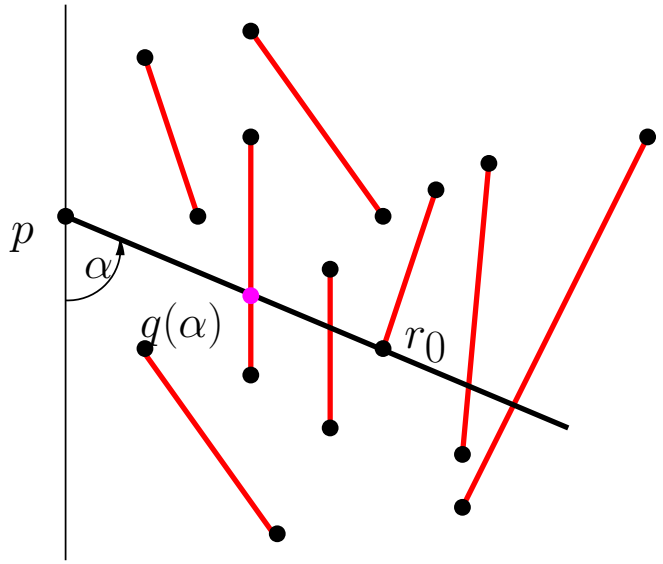
Single-Source Sweep: Punkt p

- Für jeden Endpunkt p die sichtbaren Segmente bestimmen
- Nur zu einer Seite reicht aus
- Vereinigung dieser Kanten ergibt Sichtbarkeitsgraph
- Radialer Sweep für jeden einzelnen Punkt



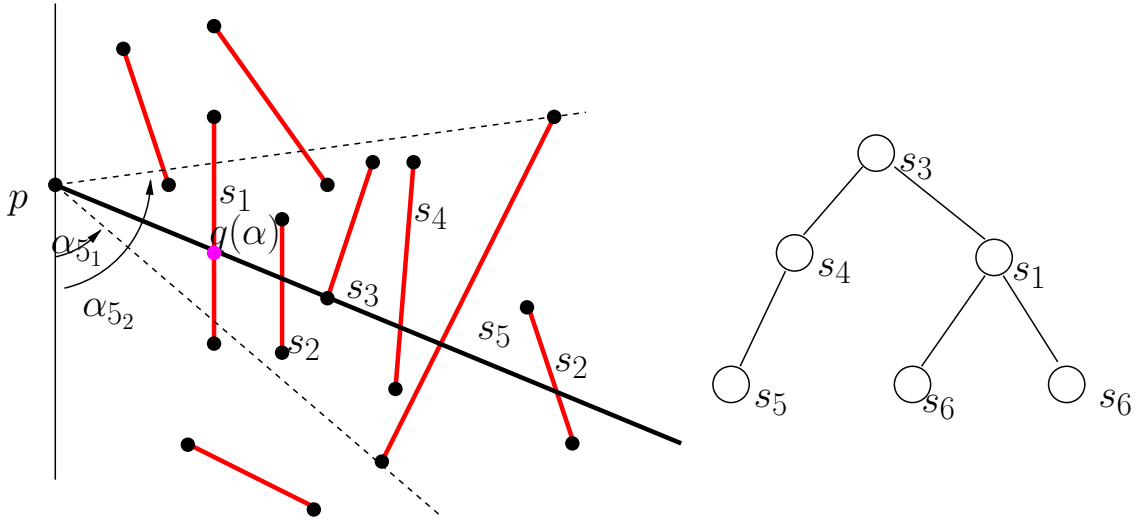
Alg. 1.2: Single-Source Sweep: Punkt p

- ES: Punkte rechts von p sortiert nach Polarkoordinaten
- SSS: Balanc. Baum (Liste der Segmente)/dynam. Schlüssel (Entfernung)
  Vorderster Punkt $q(\alpha)$
- Initialisierung: Blick nach unten



Single-Source Sweep SSS: Balancierter Baum

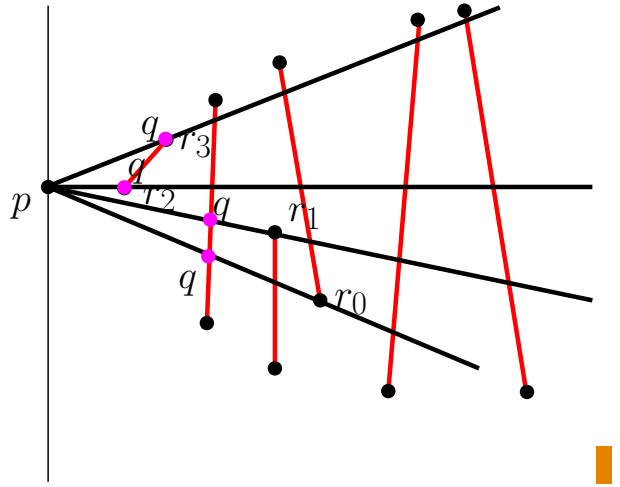
- Dynamische Schlüssel, Liniensegmente
- Abhängig vom Winkel α, Distanz zum Liniensegment
- Zeiger auf kleinstes Element des Baumes: $q(\alpha)$
- AVL Bäume: Grundstudium



Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

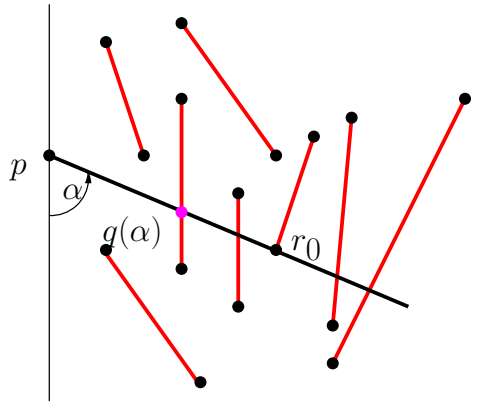
Nächster Punkt r_i: Immer in folgender Reihenfolge

1. Falls r_i Startpunkt $\Rightarrow$ Füge Segment ein: $O(\log n)$
2. Falls $r_i = q(\alpha) \Rightarrow$ Ausgabe (p, r_i): $O(1)$
3. Falls r_i Endpunkt $\Rightarrow$ Lösche Segment: $O(\log n)$



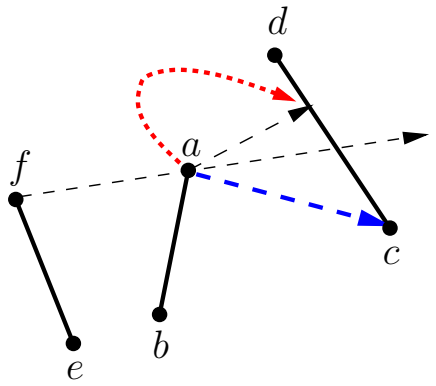
Single-Source Sweep: Korrektheit und Laufzeit

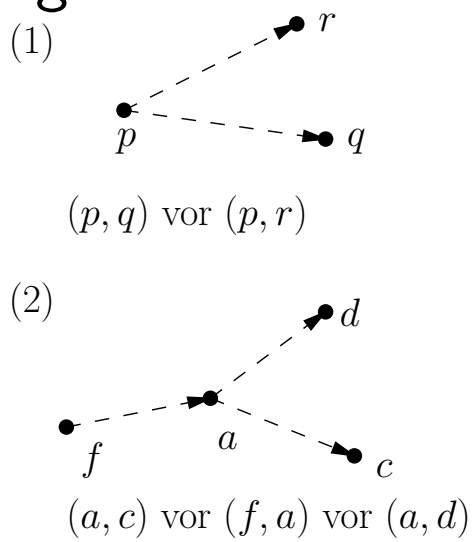
- SSS: Zu jedem Zeitpunkt ist $q(\alpha)$ sichtbarer Punkt
- Test mit aktuellem Knoten r_i
- Falls $r_i = q(\alpha) \Rightarrow$ Ausgabe (p, r_i)
- Änderung des bal. Baumes $O(\log n)$, n mal
- Sortieren der n Knoten nach Polarkoordinaten: $O(n \log n)$
- Laufzeit für alle Knoten $O(n^2 \log n)$



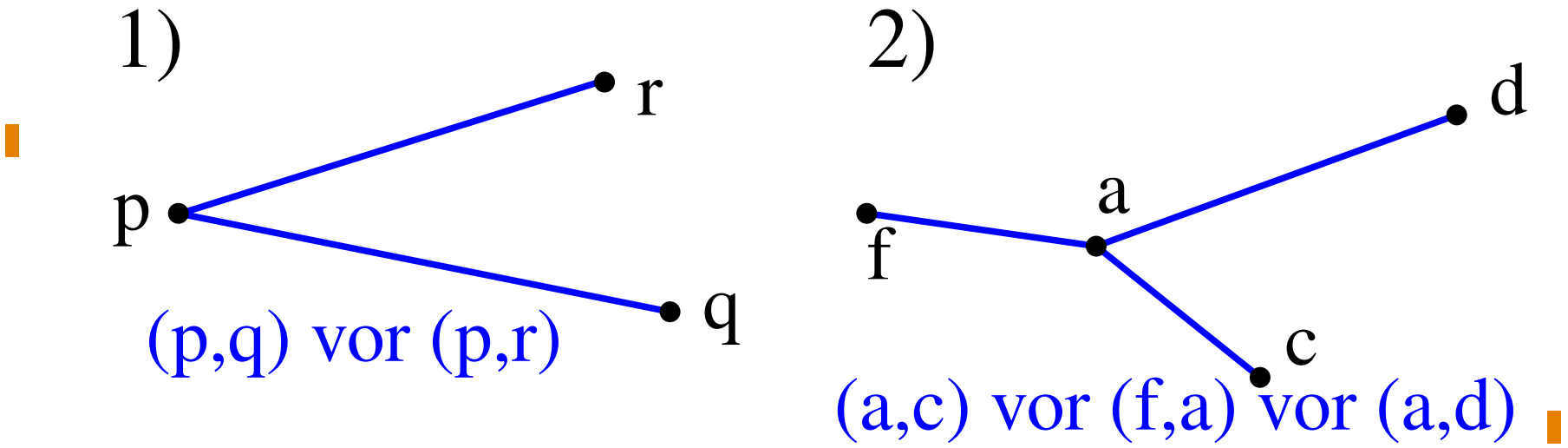
Geht das besser?

- Laufzeit $O(n^2 \log n)$, Komplexität $\Theta(n^2)$, Laufzeit $O(n^2)$?
- Simultaner Sweep, alle Segmente gleichzeitig drehen, Sicht-Informationen weiter geben
- Bearbeitungsreihenfolge: Paare von Endpunkten gemäß Steigung
- Zeiger auf momentan sichtbares Segment





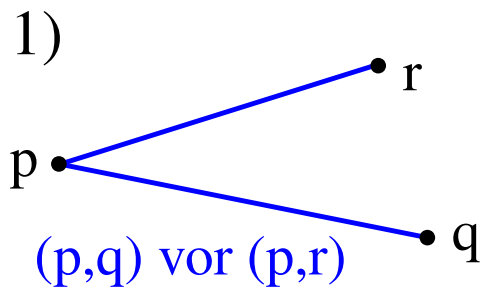
Bearbeitungsreihenfolge in $O(n^2)$

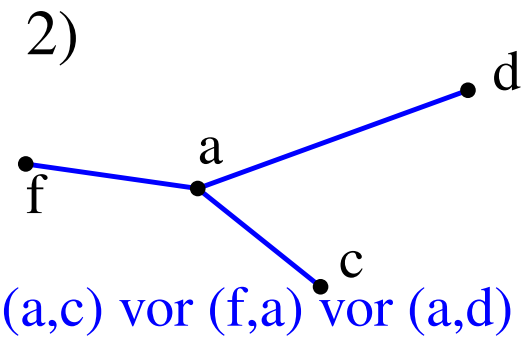


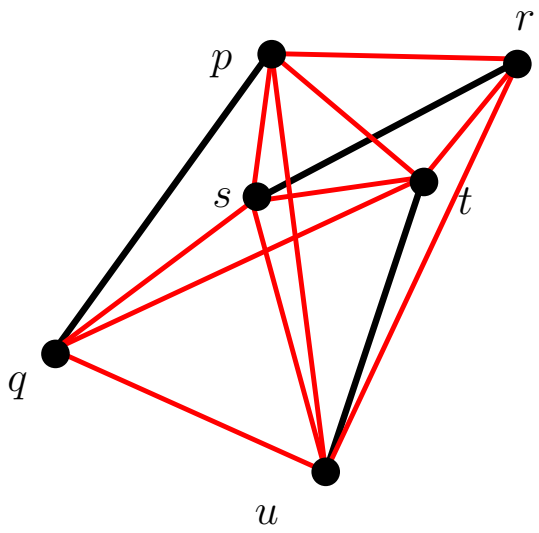
Liste von Punkten (p, q) mit $p_x \leq q_x$

Zeigen wir später! Annahme: Ist schon gegeben für die Punktpaare!

Beispiel Bearbeitungsreihenfolge







Volle Ordnung:

$(p,u),(s,u),(p,t),(q,u),(p,r),(s,t),(q,t),(q,s),(t,r),(u,r),(s,p)$

Partielle Ordung:

$(p,u),(p,t),(p,r),(q,u),(s,u),(s,t),(q,t),(q,s),\ldots$