Methoden der Offline Bewegungsplanung
Einführung

Elmar Langetepe
University of Bonn

Organisatorisches

Organisatorisches

- Bachelor BA-INF 124 Wahlpflicht, 4-6 Semester

Organisatorisches

- Bachelor BA-INF 124 Wahlpflicht, 4-6 Semester
  - Vorlesung 4 SWS, 5.5 LP, Übung 2 SWS, 3.5 LP

Organisatorisches

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

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

Organisatorisches

- Übungen: Mittwochs 16:00-18.00 Uhr, A6c

Organisatorisches

- Übungen: Mittwochs 16:00-18.00 Uhr, A6c
- Wöchentlich, Beginn: 15.10

Organisatorisches

- Übungen: Mittwochs 16:00-18.00 Uhr, A6c
- Wöchentlich, Beginn: 15.10
- Aufgabenblatt, mittwochs! Blatt I: 15.10 (3 Aufgaben)

Organisatorisches

- Übungen: Mittwochs 16:00-18.00 Uhr, A6c
- Wöchentlich, Beginn: 15.10
- Aufgabenblatt, mittwochs! Blatt I: 15.10 (3 Aufgaben)
- Abgabe mittwochs, Kasten Voyer

Organisatorisches

- Übungen: Mittwochs 16:00-18.00 Uhr, A6c
- Wöchentlich, Beginn: 15.10
- Aufgabenblatt, mittwochs! Blatt I: 15.10 (3 Aufgaben)
- Abgabe mittwochs, Kasten Voyer
- Aufgaben, Lösungen schriftlich, Abgabe, Korrektur, Besprechung

Organisatorisches

- Übungen: Mittwochs 16:00-18.00 Uhr, A6c
- Wöchentlich, Beginn: 15.10
- Aufgabenblatt, mittwochs! Blatt I: 15.10 (3 Aufgaben)
- Abgabe mittwochs, Kasten Voyer
- Aufgaben, Lösungen schriftlich, Abgabe, Korrektur, Besprechung
- Bachelorarbeiten (Themenhinweise)

Organisatorisches

- Übungen: Mittwochs 16:00-18.00 Uhr, A6c
- Wöchentlich, Beginn: 15.10
- Aufgabenblatt, mittwochs! Blatt I: 15.10 (3 Aufgaben)
- Abgabe mittwochs, Kasten Voyer
- Aufgaben, Lösungen schriftlich, Abgabe, Korrektur, Besprechung
- Bachelorarbeiten (Themenhinweise)
- Projektgruppe (Java,`www.geometrylab.de`)

Organisatorisches

- Übungen: Mittwochs 16:00-18.00 Uhr, A6c
- Wöchentlich, Beginn: 15.10
- Aufgabenblatt, mittwochs! Blatt I: 15.10 (3 Aufgaben)
- Abgabe mittwochs, Kasten Voyer
- Aufgaben, Lösungen schriftlich, Abgabe, Korrektur, Besprechung
- Bachelorarbeiten (Themenhinweise)
- Projektgruppe (Java,`www.geometrylab.de`)
- Folien/Skript online

Organisatorisches

- Übungen: Mittwochs 16:00-18.00 Uhr, A6c
- Wöchentlich, Beginn: 15.10
- Aufgabenblatt, mittwochs! Blatt I: 15.10 (3 Aufgaben)
- Abgabe mittwochs, 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

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

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

Einordung: Bewegungsplanung

- Maschinen/Agenten

Einordung: Bewegungsplanung

- Maschinen/Agenten
- Elektronik

Einordung: Bewegungsplanung

- Maschinen/Agenten
- Elektronik
- Mechanik

Einordung: Bewegungsplanung

- Maschinen/Agenten
- Elektronik
- Mechanik
- Kontroll- und Regelungstechnik

Einordung: Bewegungsplanung

- Maschinen/Agenten
- Elektronik
- Mechanik
- Kontroll- und Regelungstechnik
- KI

Einordung: Bewegungsplanung

- Maschinen/Agenten
- Elektronik
- Mechanik
- Kontroll- und Regelungstechnik
- KI
- Softwareengeneering

Einordung: Bewegungsplanung

- Maschinen/Agenten
- Elektronik
- Mechanik
- Kontroll- und Regelungstechnik
- KI
- Softwareengeneering
- ⋮

Einordung: Bewegungsplanung

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

Einordung: Bewegungsplanung

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

Geometrische Algorithmen

Geometrische Algorithmen

- Geometrie des Agenten und der Szene sind gegeben

Geometrische Algorithmen

- Geometrie des Agenten und der Szene sind gegeben
- Löse Bahnplanungsproblem

Geometrische Algorithmen

- Geometrie des Agenten und der Szene sind gegeben
- Löse Bahnplanungsproblem
- Unterscheidung: Online/Offline

Geometrische Algorithmen

- Geometrie des Agenten und der Szene sind gegeben
- Löse Bahnplanungsproblem
- Unterscheidung: Online/Offline
- Fragen: Komplexität der Lösungen/Laufzeitkomplexität/Datenstrukturen

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

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

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)

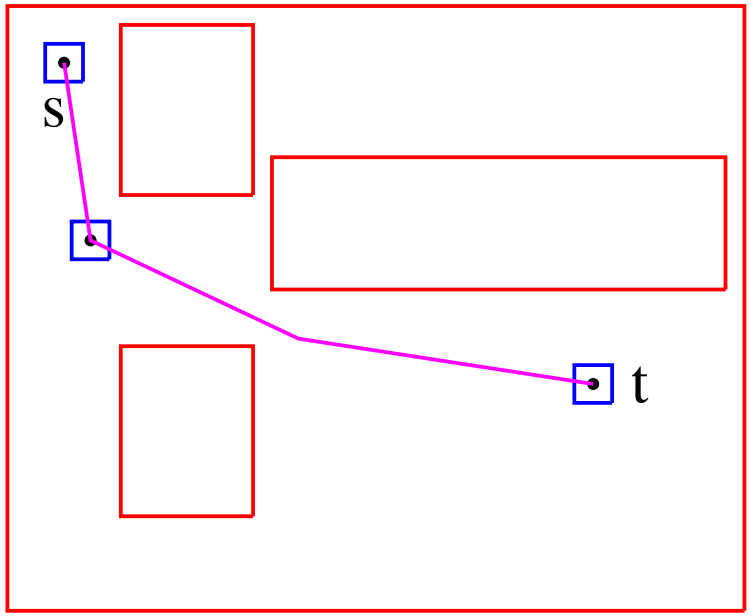
Geometrische Algorithmen

- 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

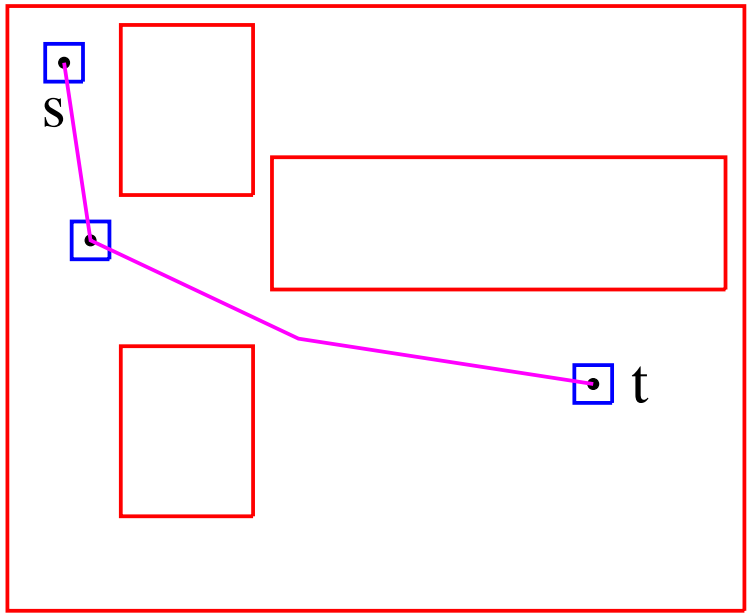
- Geometrie des Agenten und der Szene sind gegeben
- Löse Bahnplanungsproblem
- Unterscheidung: Online/Offline
- Fragen: Komplexität der Lösungen/Laufzeitkomplexität/Datenstrukturen
- Untere Schranke/Obere Schranke
- Strukturelle Eigenschaften
- Modularer Aufbau (Black Box)
- Viele Grundprobleme lösen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

Bsp: Translation Quadrat unter Rechtecken



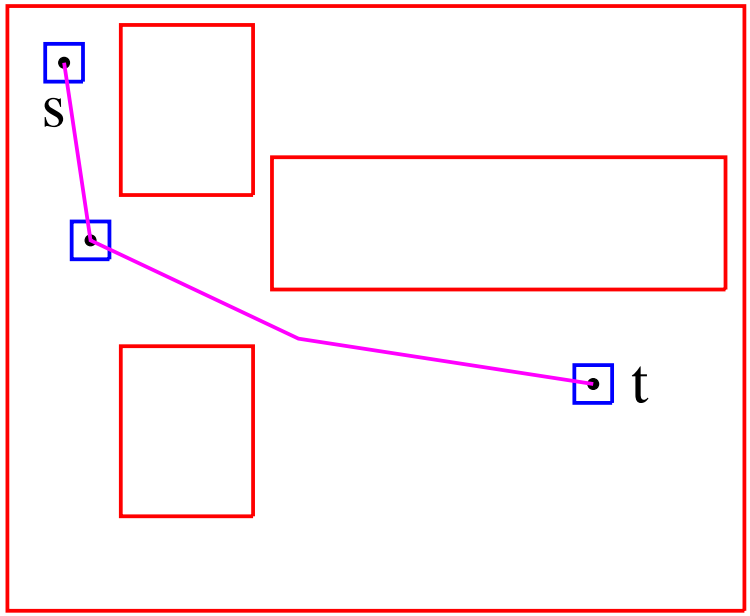
Bsp: Translation Quadrat unter Rechtecken

- Achsenparalleles Rechteck von s nach t bewegen



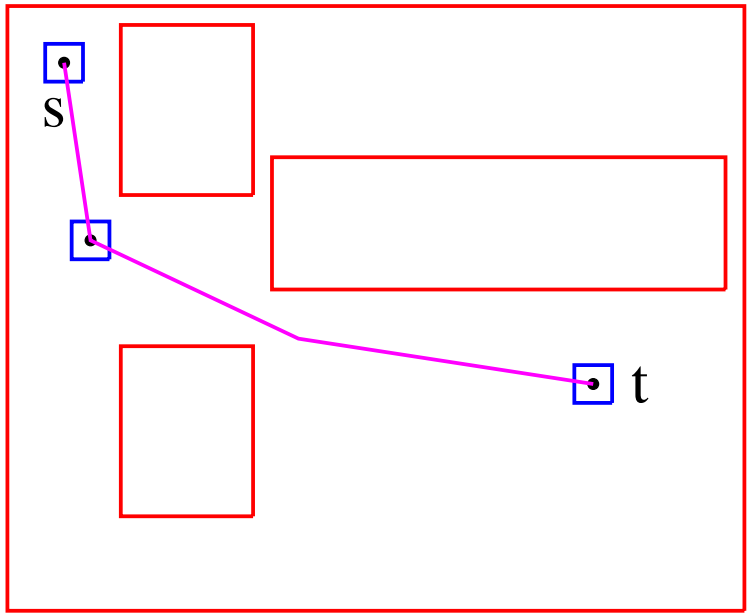
Bsp: Translation Quadrat unter Rechtecken

- Achsenparalleles Rechteck von s nach t bewegen
- Achsenparallele Hindernisse



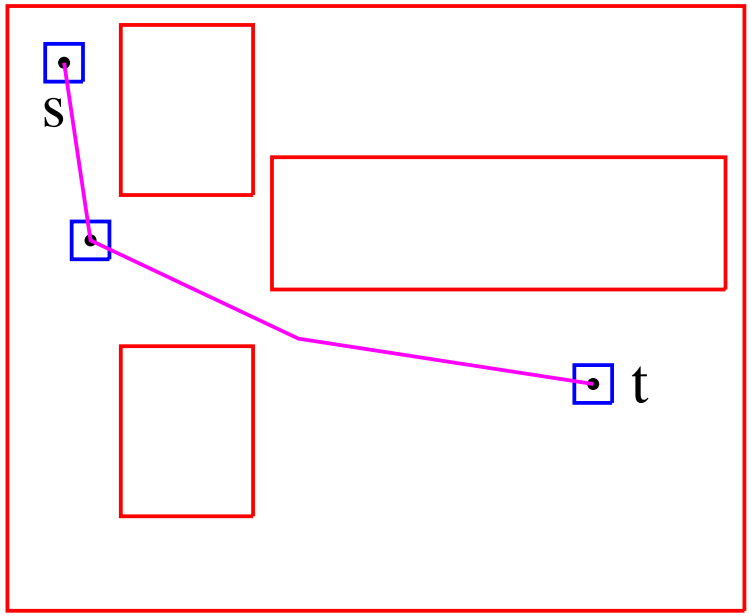
Bsp: Translation Quadrat unter Rechtecken

- Achsenparalleles Rechteck von s nach t bewegen
- Achsenparallele Hindernisse
- Fragen:



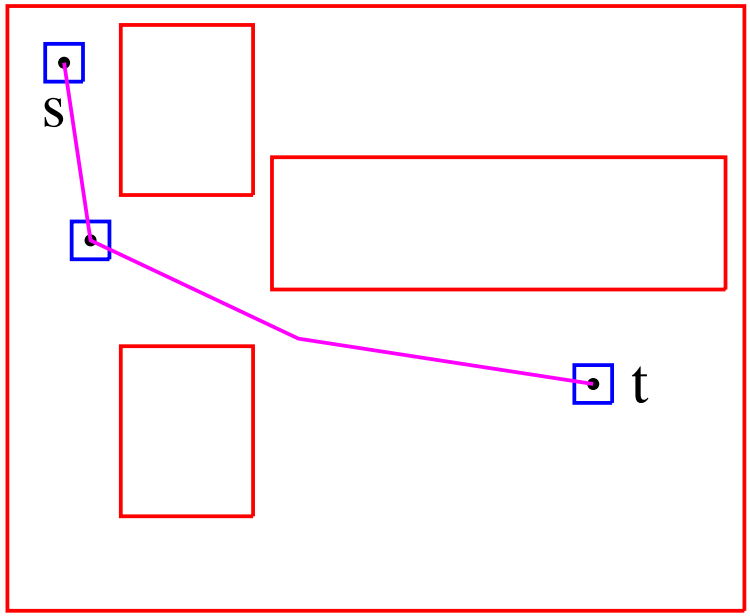
Bsp: Translation Quadrat unter Rechtecken

- Achsenparalleles Rechteck von s nach t bewegen
- Achsenparallele Hindernisse
- Fragen: Geht das?



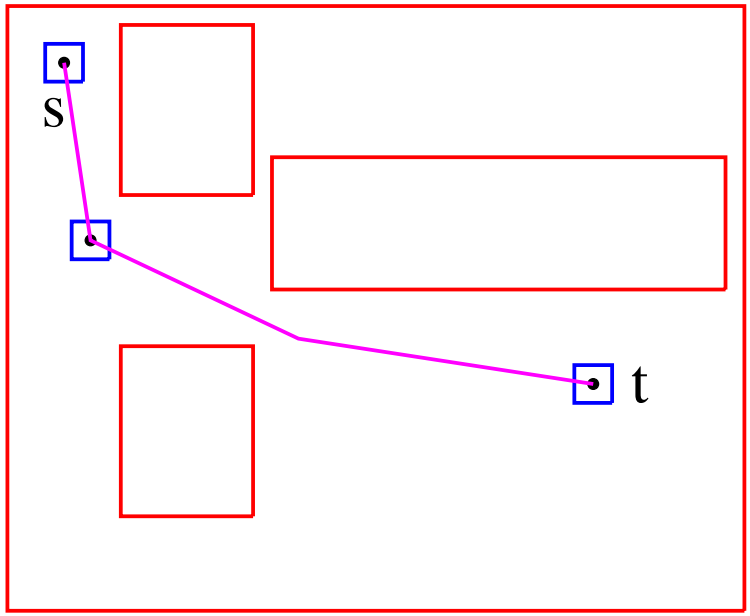
Bsp: Translation Quadrat unter Rechtecken

- Achsenparalleles Rechteck von s nach t bewegen
- Achsenparallele Hindernisse
- Fragen: Geht das? Geringe Kosten?



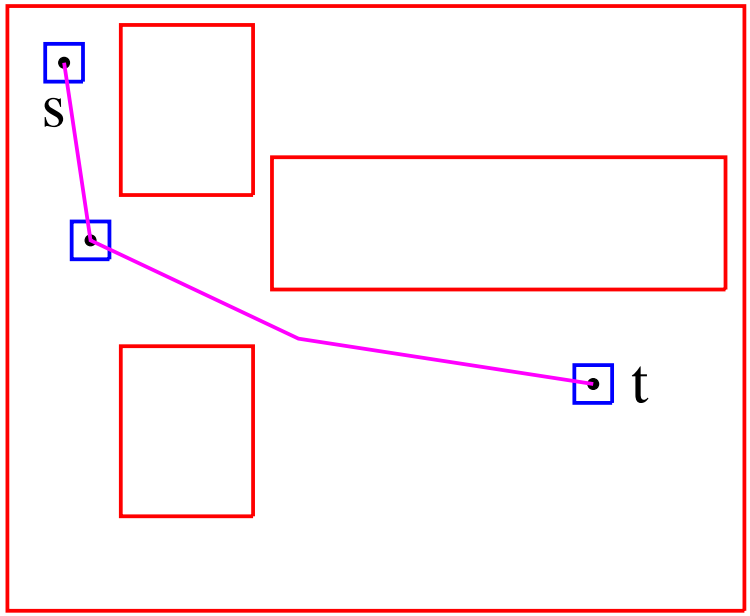
Bsp: Translation Quadrat unter Rechtecken

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



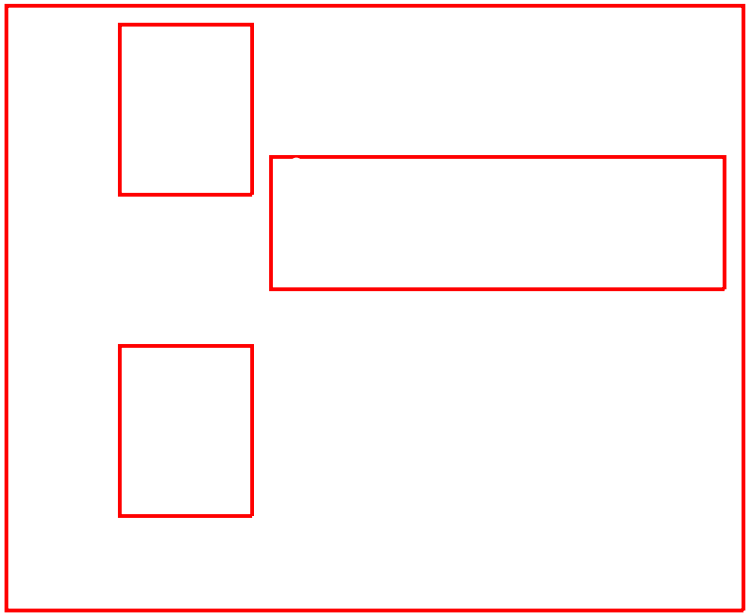
Bsp: Translation Quadrat unter Rechtecken

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

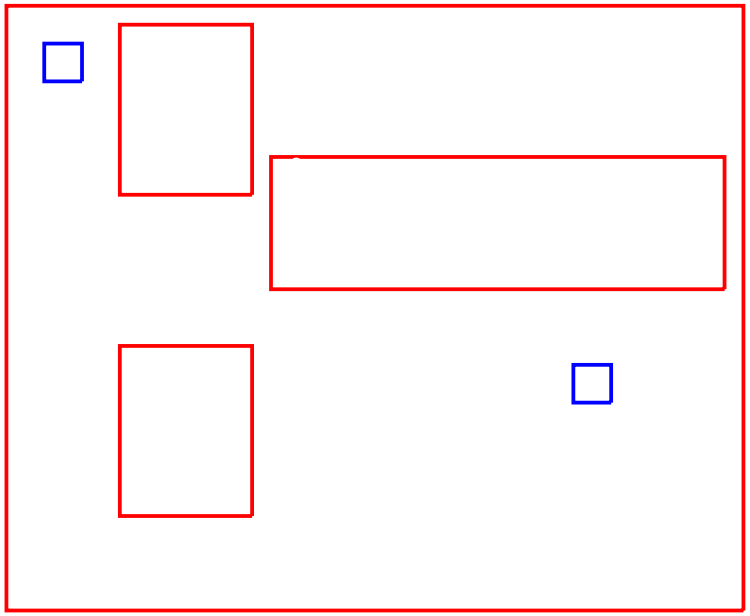


Idee: Translation Quadrat unter Rechtecken

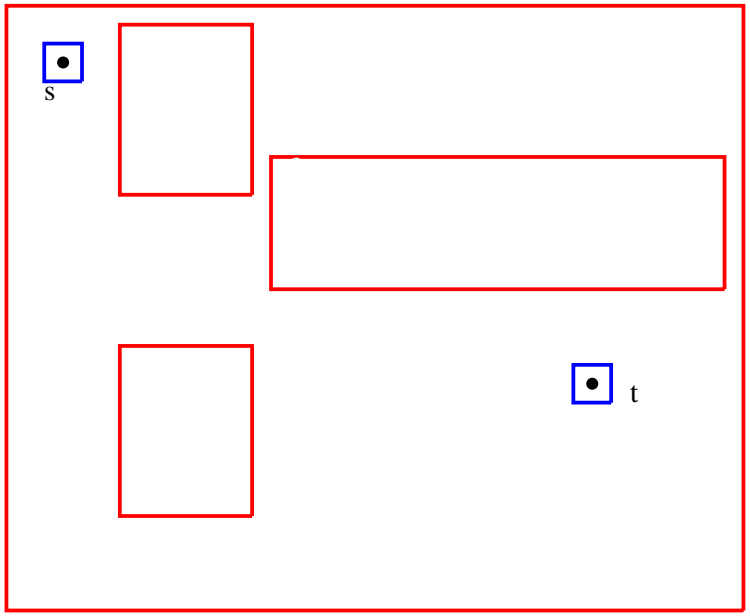
Idee: Translation Quadrat unter Rechtecken



Idee: Translation Quadrat unter Rechtecken

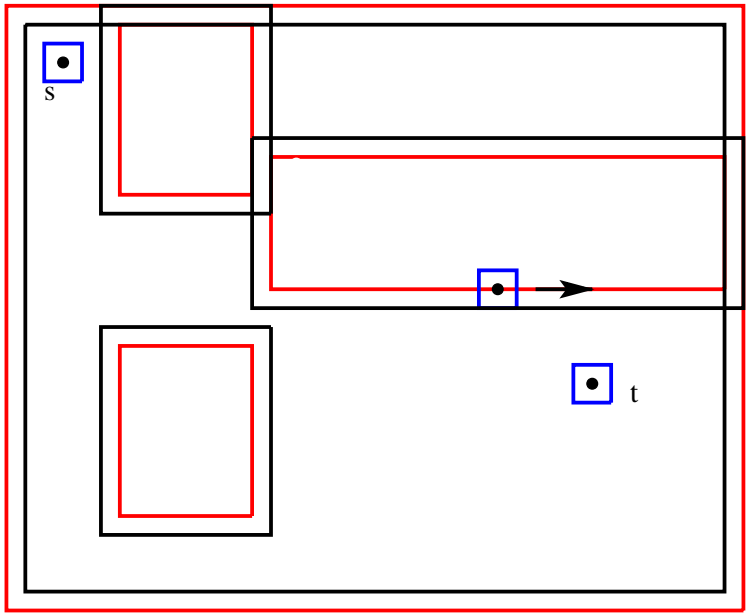


Idee: Translation Quadrat unter Rechtecken



Translation: Verschieben Referenzpunkt

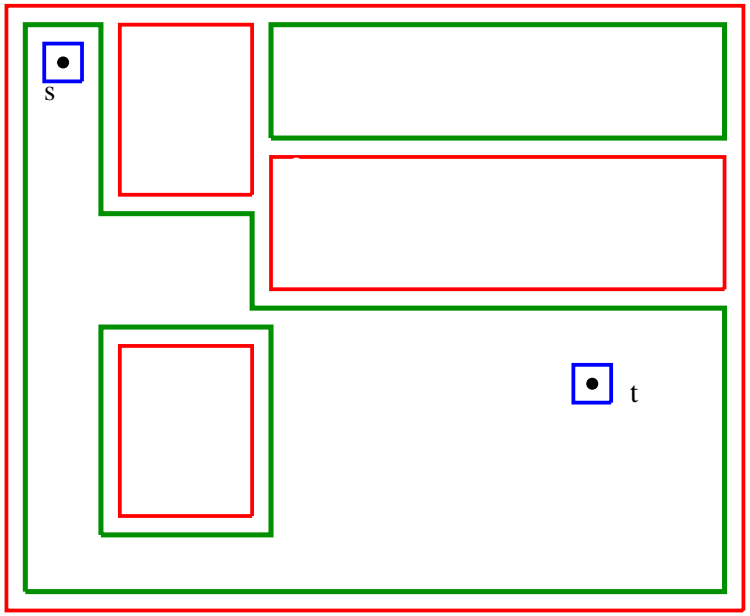
Idee: Translation Quadrat unter Rechtecken



Translation: Verschieben Referenzpunkt

Aufblasen der Hindernisse

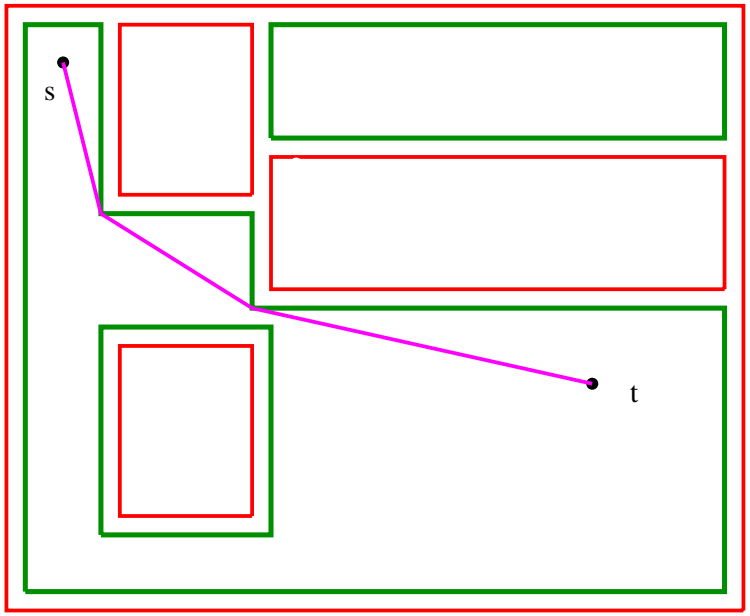
Idee: Translation Quadrat unter Rechtecken



Translation: Verschieben Referenzpunkt

Aufblasen der Hindernisse
Vereinigung Sweep

Idee: Translation Quadrat unter Rechtecken



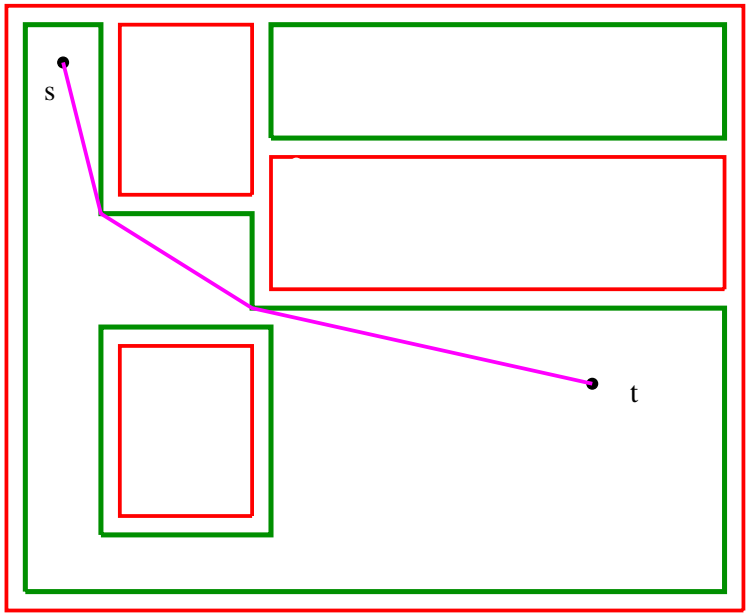
Translation: Verschieben Referenzpunkt

Aufblasen der Hindernisse

Vereinigung Sweep

Kürzester Weg????

Idee: Translation Quadrat unter Rechtecken



Translation: Verschieben Referenzpunkt

Aufblasen der Hindernisse

Vereinigung Sweep

Kürzester Weg????

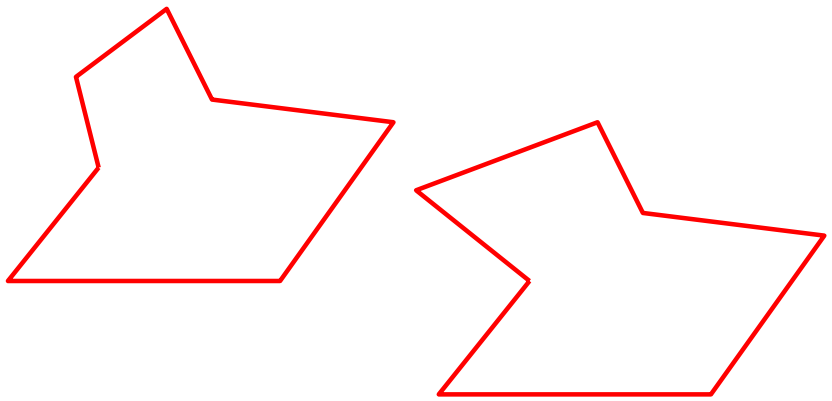
Kap 1.1: Kürzeste Wege, polygonale Hindernisse

Kap 1.1: Kürzeste Wege, polygonale Hindernisse

- *Polygonale Hindernisse*: Definition

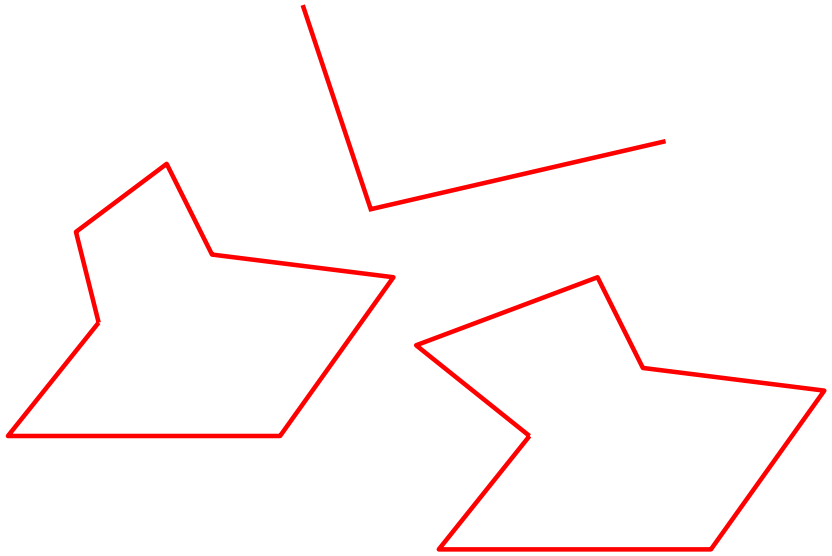
Kap 1.1: Kürzeste Wege, polygonale Hindernisse

- *Polygonale Hindernisse*: Definition
- Geschlossene Kantenzüge,



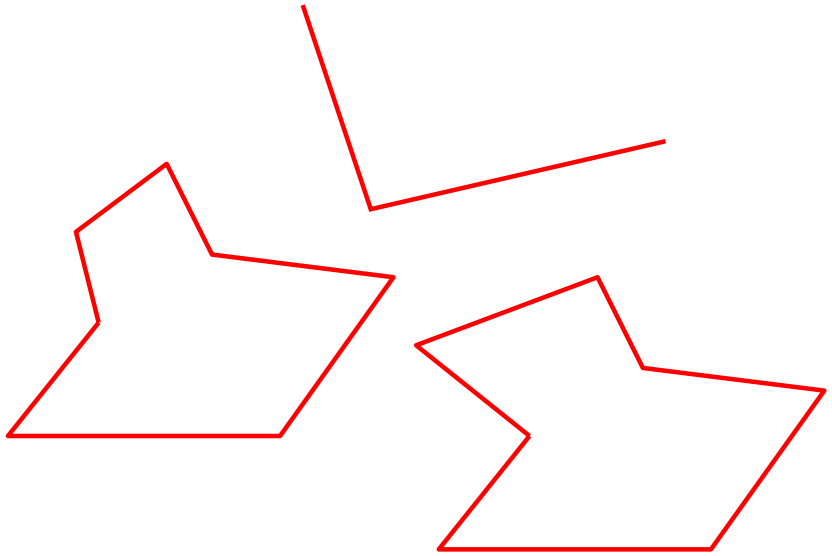
Kap 1.1: Kürzeste Wege, polygonale Hindernisse

- *Polygonale Hindernisse*: Definition
- Geschlossene Kantenzüge, offene Kantenüge



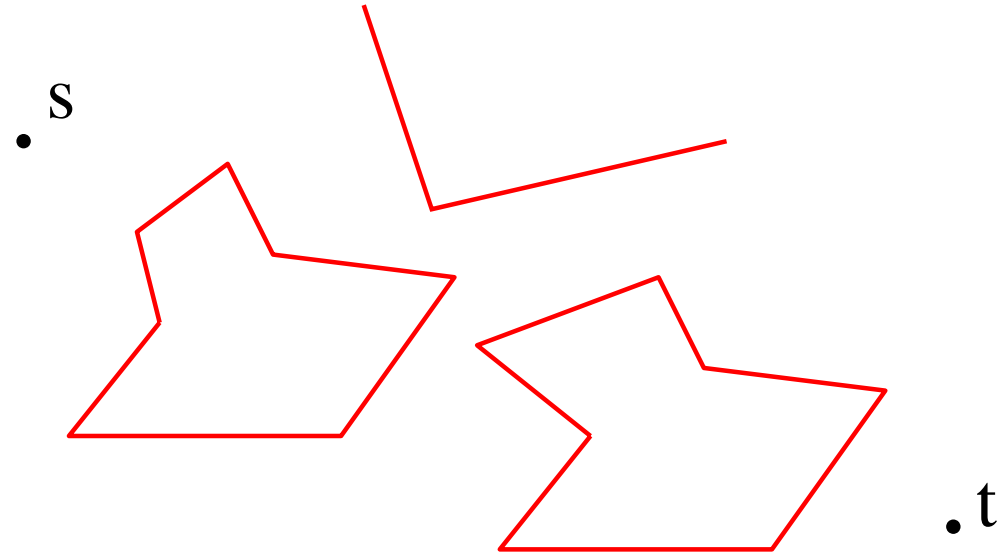
Kap 1.1: Kürzeste Wege, polygonale Hindernisse

- *Polygonale Hindernisse*: Definition
- Geschlossene Kantenzüge, offene Kantenüge
- Kantenfolge, Kanten schneiden sich nur an den Eckpunkten, nur aufeinanderfolgende



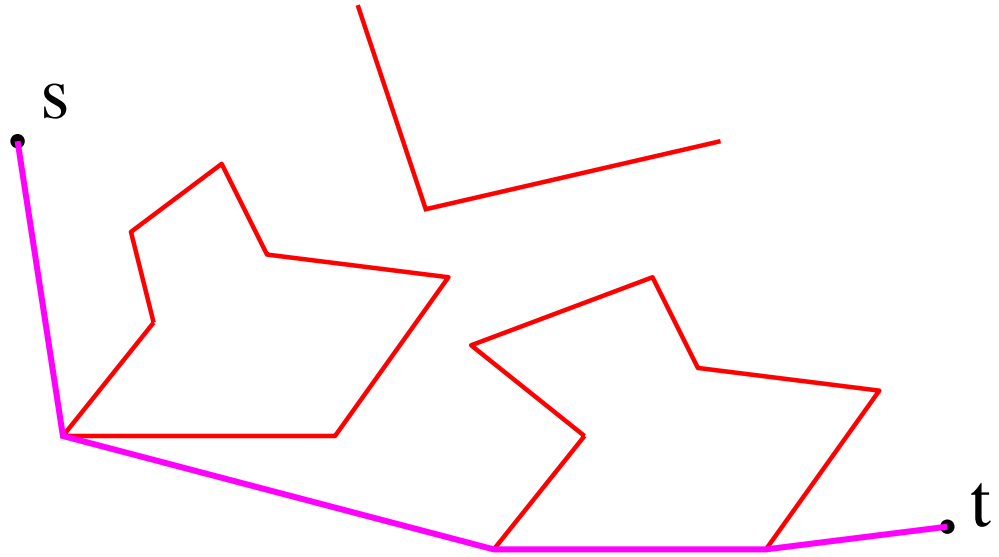
Kap 1.1: Kürzeste Wege, polygonale Hindernisse

- *Polygonale Hindernisse*: Definition
- Geschlossene Kantenzüge, offene Kantenüge
- Kantenfolge, Kanten schneiden sich nur an den Eckpunkten, nur aufeinanderfolgende
- *Polygonale Szene*: Hindernisse berühren sich nicht

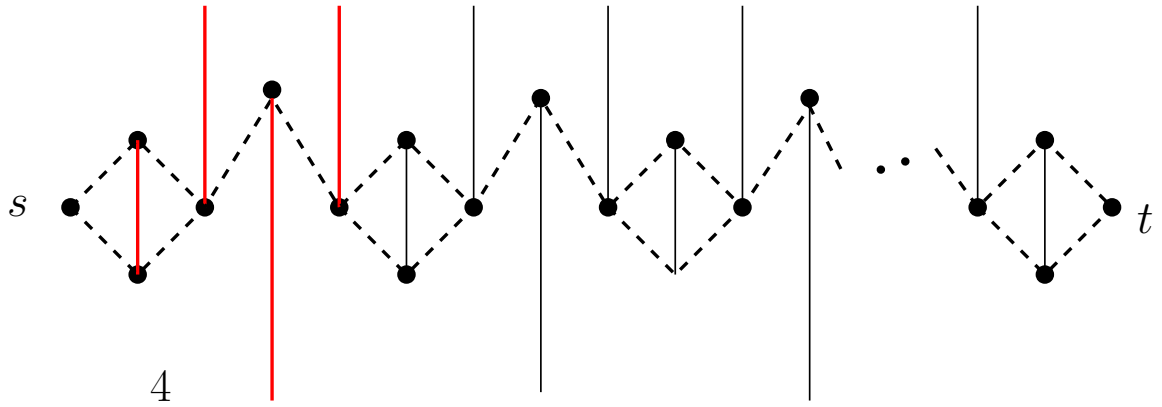


Kap 1.1: Kürzeste Wege, polygonale Hindernisse

- *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?

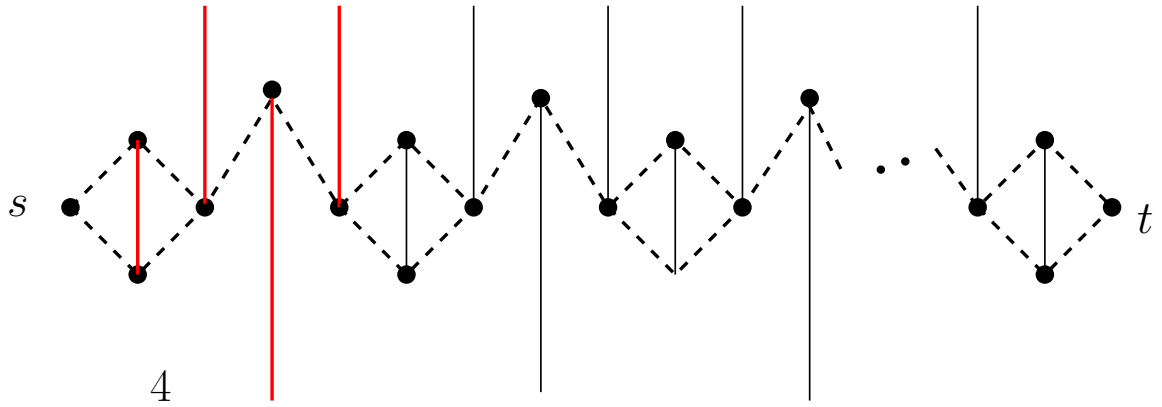


Eigenschaften Kürzester Wege: Anzahl



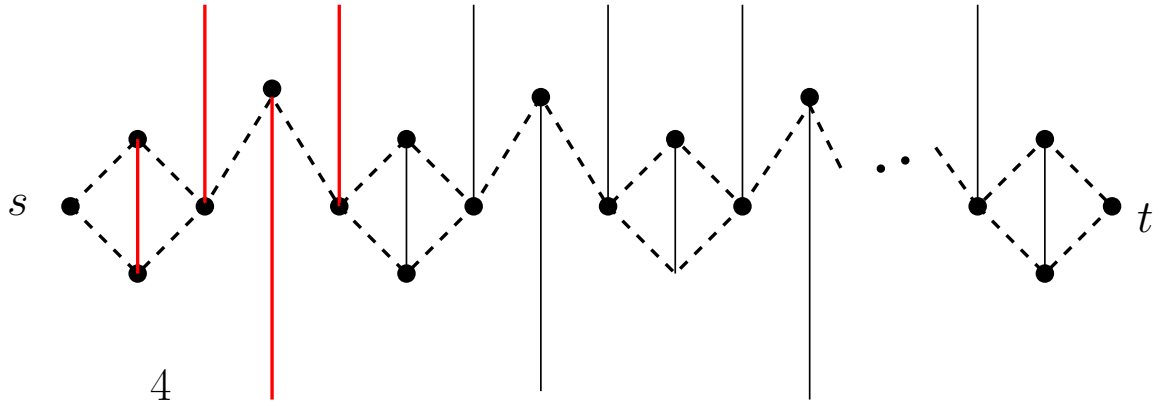
Eigenschaften Kürzester Wege: Anzahl

- m Kanten, exponentiell viele (in m) kürzeste Wege



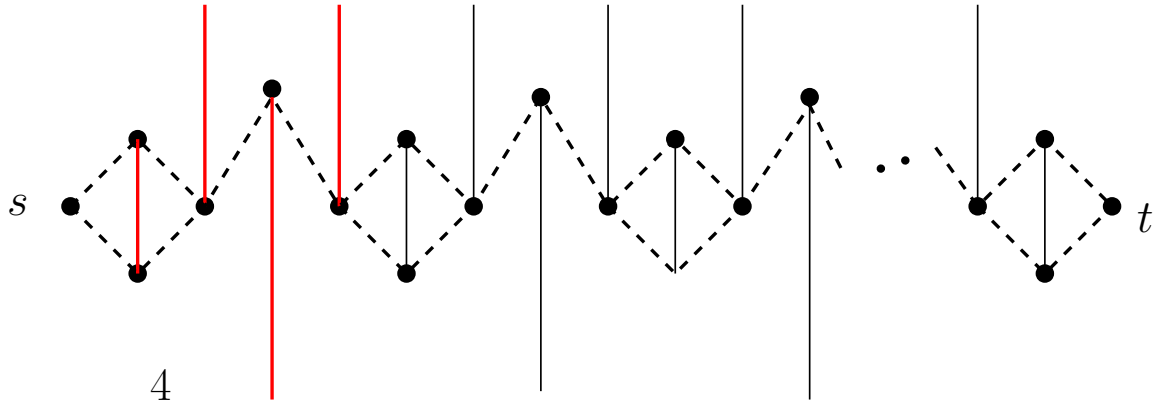
Eigenschaften Kürzester Wege: Anzahl

- m Kanten, exponentiell viele (in m) kürzeste Wege
- Genauer $h = 4m + 1$ Kanten, 2^{m+1} kürzeste Wege



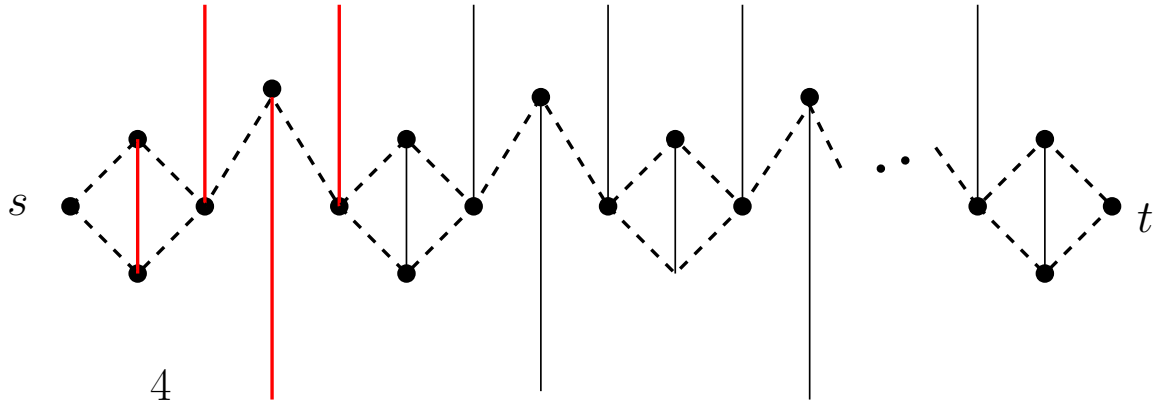
Eigenschaften Kürzester Wege: Anzahl

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



Eigenschaften Kürzester Wege: Anzahl

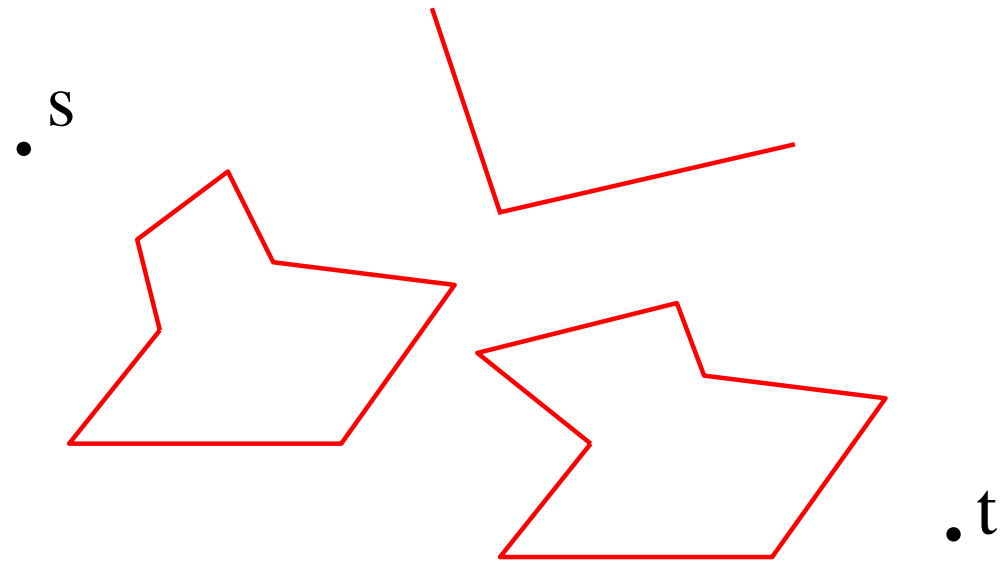
- m Kanten, exponentiell viele (in m) kürzeste Wege
- Genauer $h = 4m + 1$ Kanten, 2^{m+1} kürzeste Wege
- Pro Block verdoppeln, einmal verdoppeln am Ende



Kürzester Weg: Strukturelle Eigenschaften

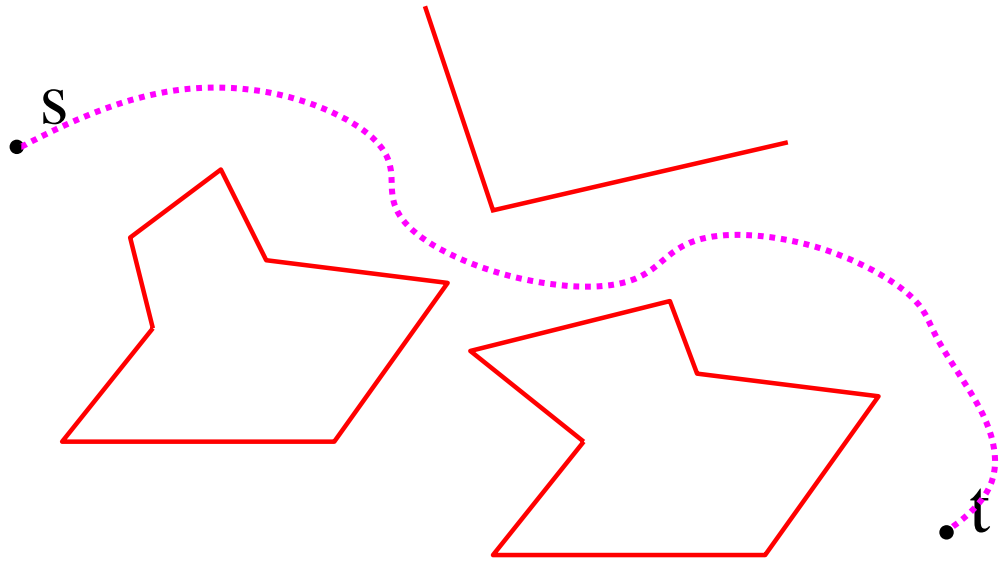
Kürzester Weg: Strukturelle Eigenschaften

- Polygonale Kette über Ecken,



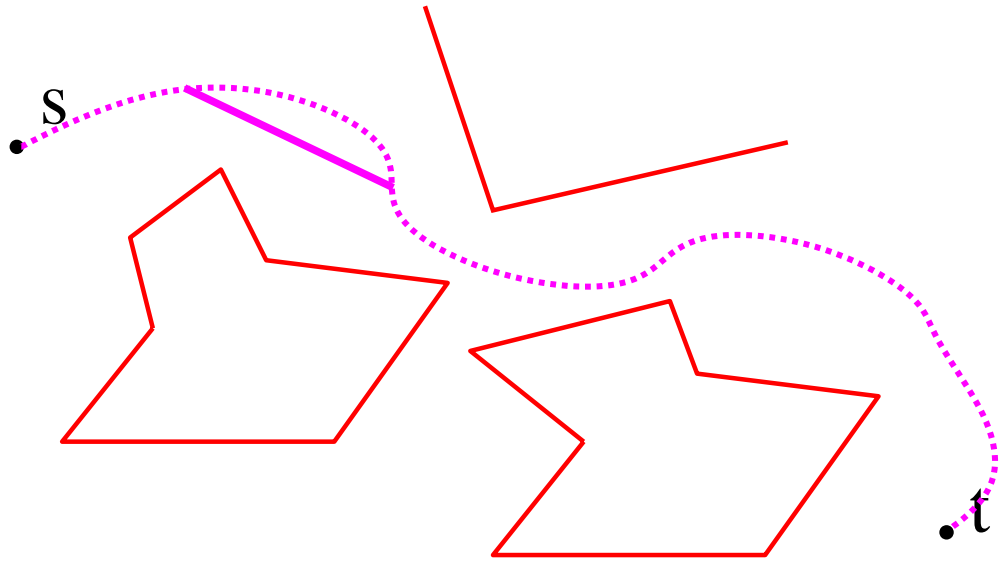
Kürzester Weg: Strukturelle Eigenschaften

- Polygonale Kette über Ecken,



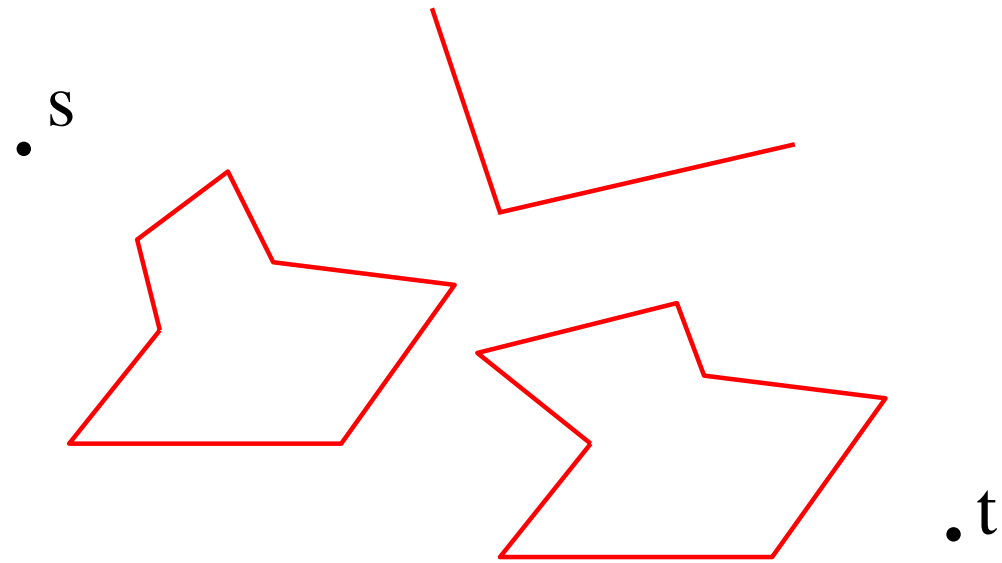
Kürzester Weg: Strukturelle Eigenschaften

- Polygonale Kette über Ecken, lokal abkürzen



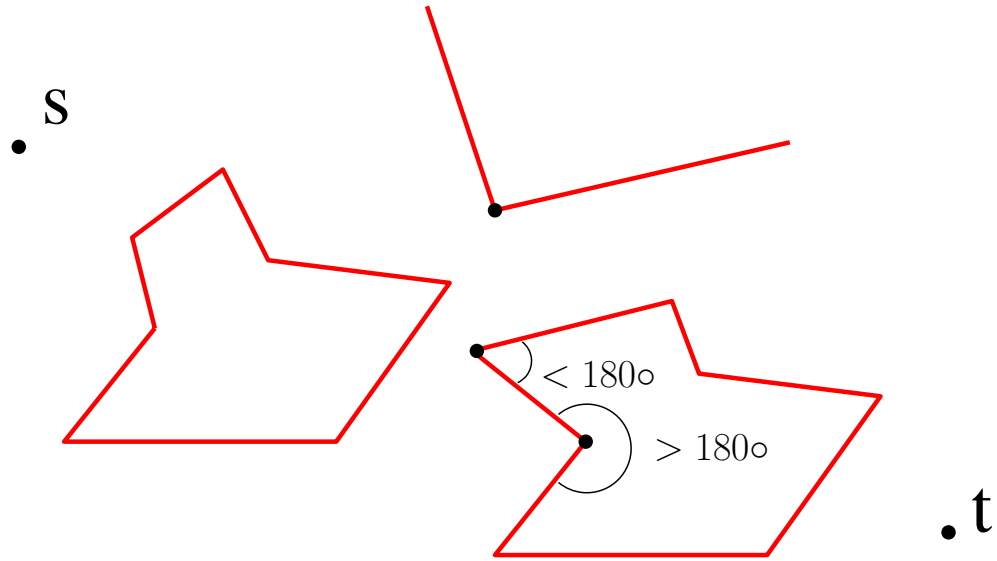
Kürzester Weg: Strukturelle Eigenschaften

- Polygonale Kette über Ecken, lokal abkürzen
- Nur über konvexe Ecken der Hindernisse



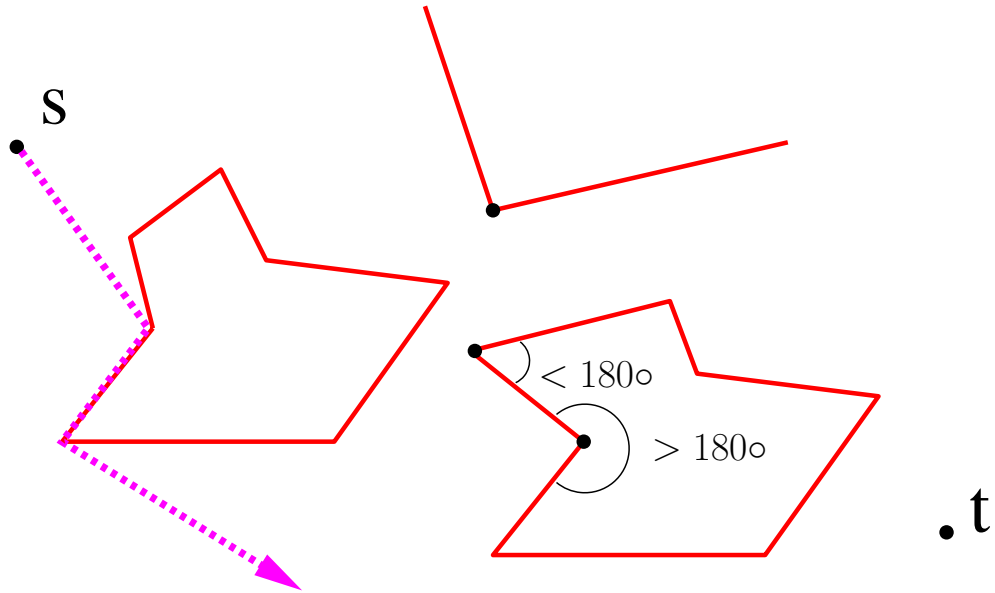
Kürzester Weg: Strukturelle Eigenschaften

- Polygonale Kette über Ecken, lokal abkürzen
- Nur über konvexe Ecken der Hindernisse
- Innenwinkel kleiner 180 Grad,



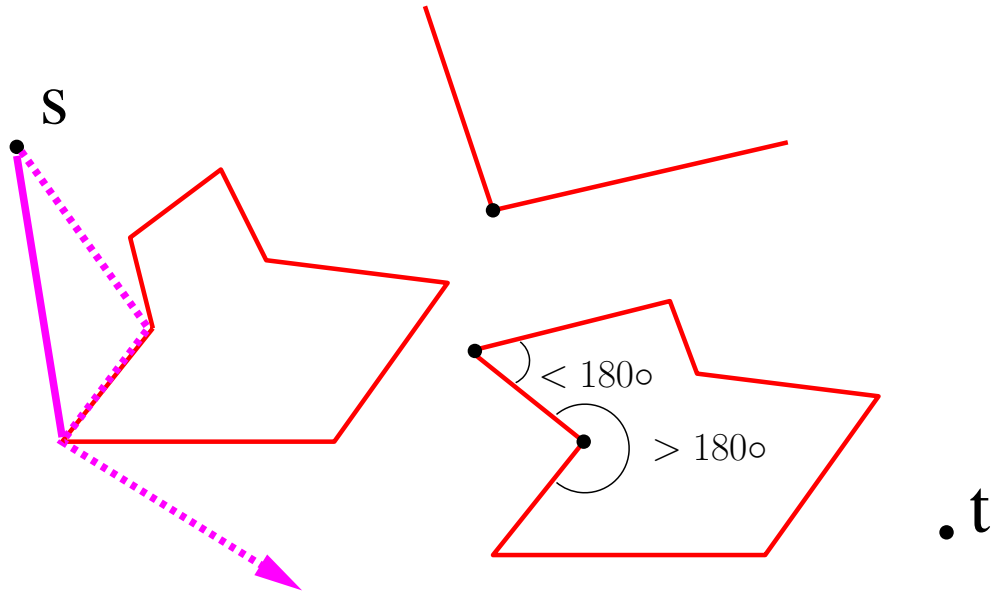
Kürzester Weg: Strukturelle Eigenschaften

- Polygonale Kette über Ecken, lokal abkürzen
- Nur über konvexe Ecken der Hindernisse
- Innenwinkel kleiner 180 Grad,



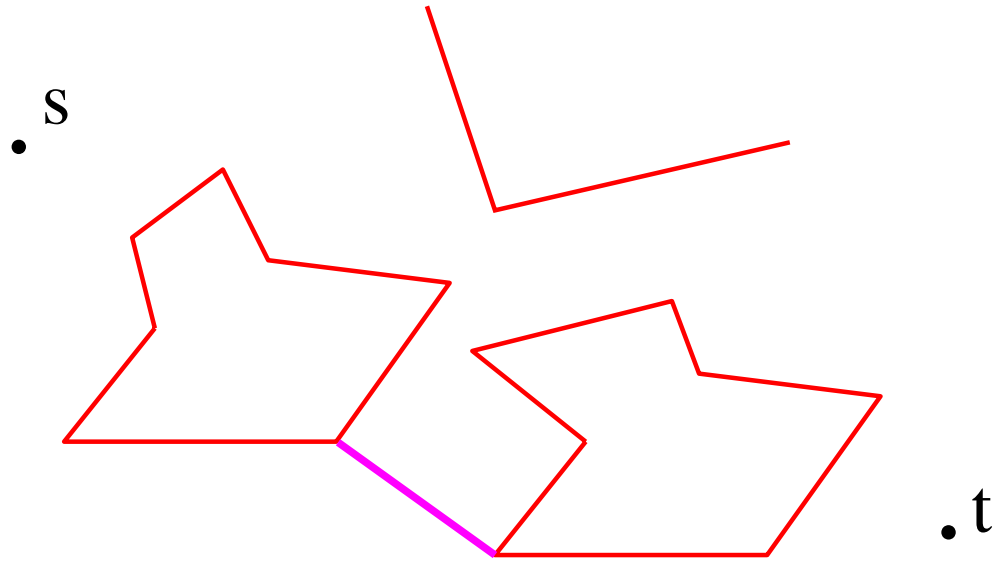
Kürzester Weg: Strukturelle Eigenschaften

- Polygonale Kette über Ecken, lokal abkürzen
- Nur über konvexe Ecken der Hindernisse
- Innenwinkel kleiner 180 Grad, lokal abkürzen



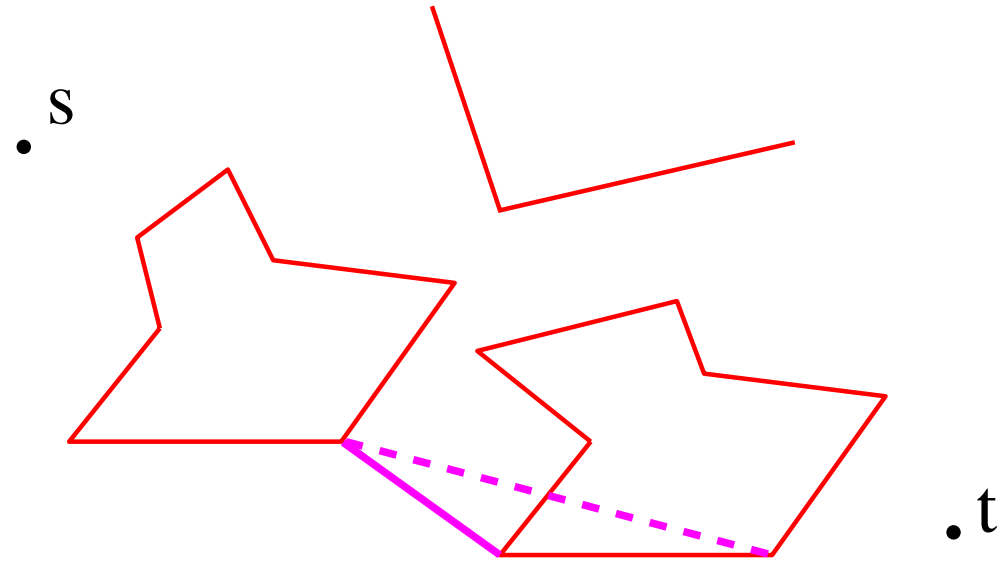
Kürzester Weg: Strukturelle Eigenschaften

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



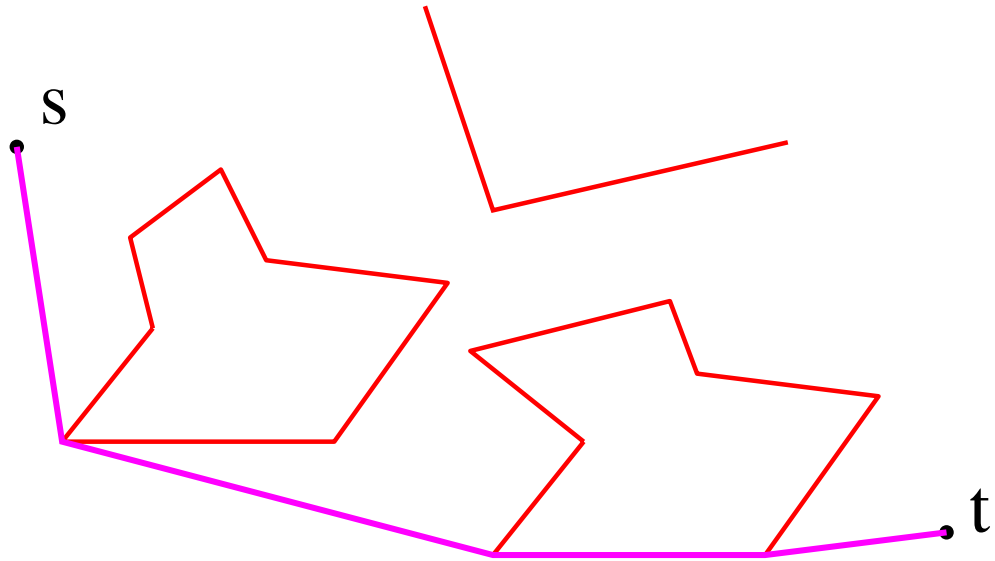
Kürzester Weg: Strukturelle Eigenschaften

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



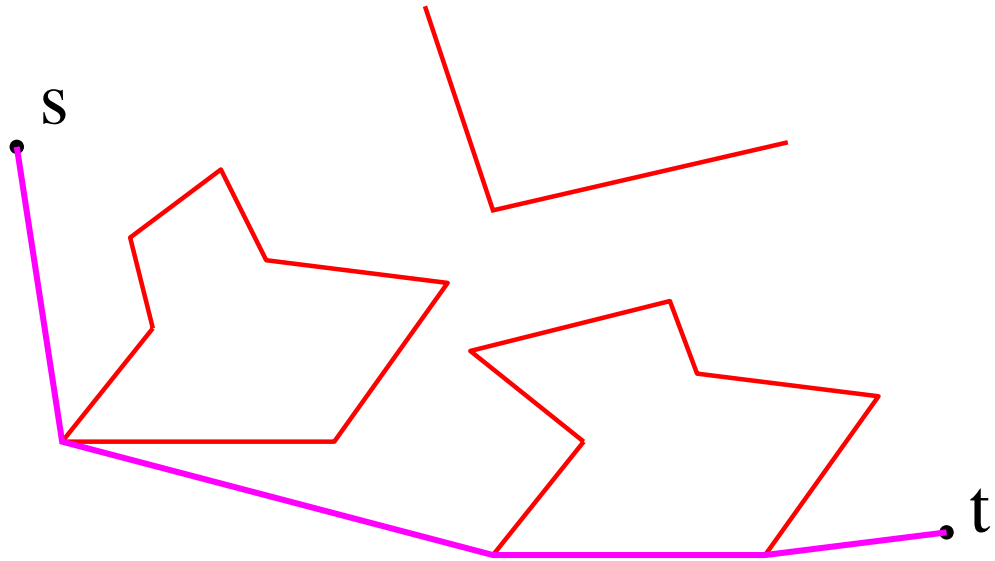
Kürzester Weg: Strukturelle Eigenschaften

- 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,



Kürzester Weg: Strukturelle Eigenschaften

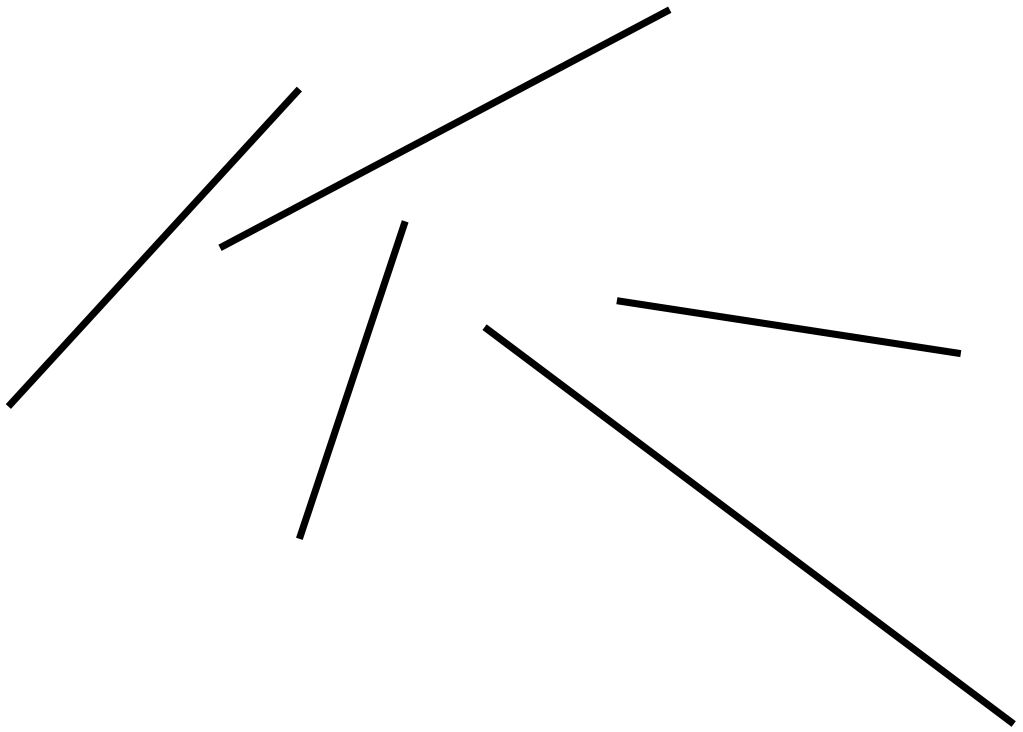
- Polygonale Kette über Ecken, lokal abkürzen
- Nur über konvexe Ecken der Hindernisse
- Innenwinkel kleiner 180 Grad, lokal abkürzen
- Über gegenseitig *sichtbare* Ecken
- Verwendet nur solche Kanten, sonst abkürzen



Allg.: Sichtbarkeitsgraph für Liniensegmente

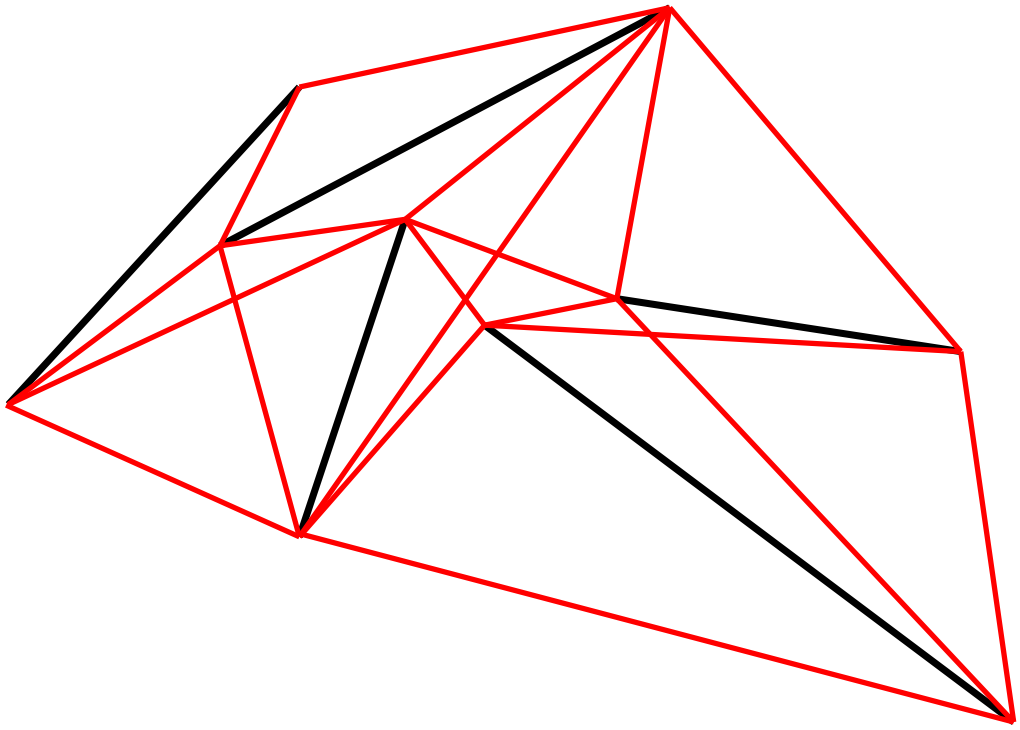
Allg.: Sichtbarkeitsgraph für Liniensegmente

- Menge L von nicht-schneidenden Segmenten gegeben



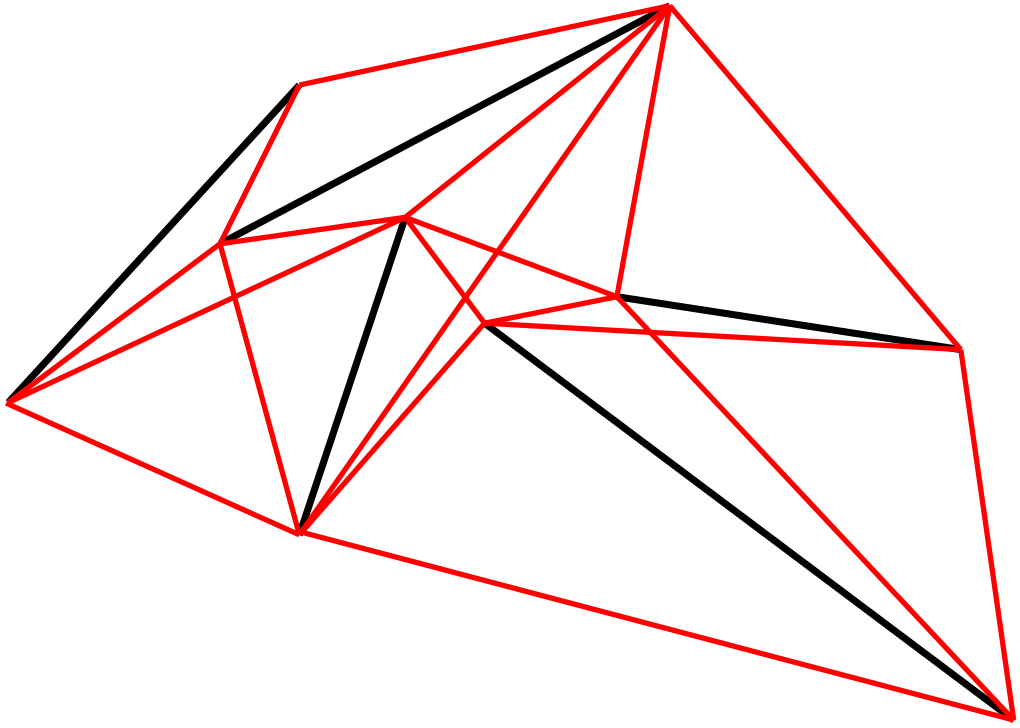
Allg.: Sichtbarkeitsgraph für Liniensegmente

- Menge L von nicht-schneidenden Segmenten gegeben
- Berechne die gegenseitig *sichtbaren* Segmente



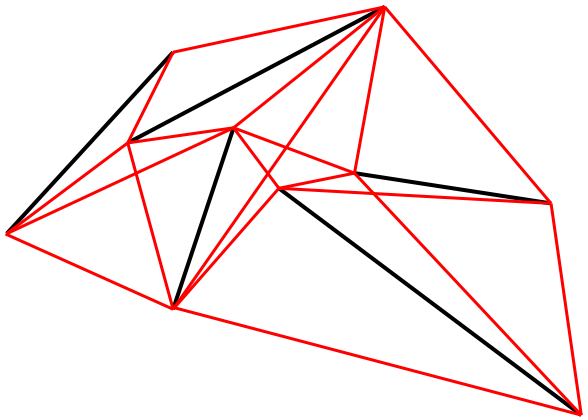
Allg.: Sichtbarkeitsgraph für Liniensegmente

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

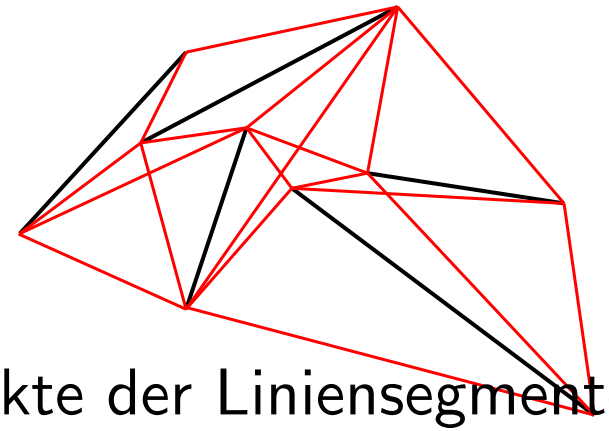


Def. 1.1: Sichtbarkeitsgraph $\texttt{VisG}(L) = (V, E)$

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



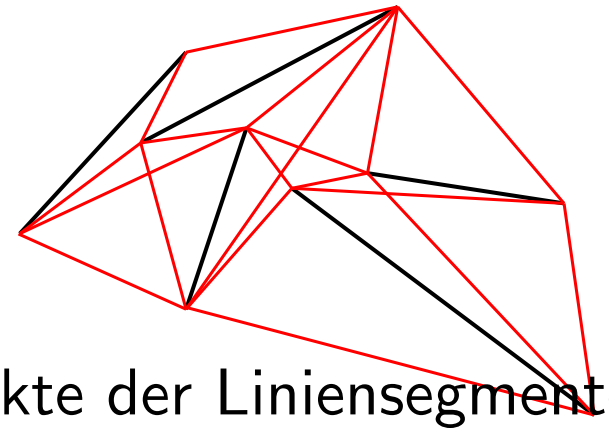
Def. 1.1: Sichtbarkeitsgraph $\texttt{VisG}(L) = (V, E)$



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

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

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

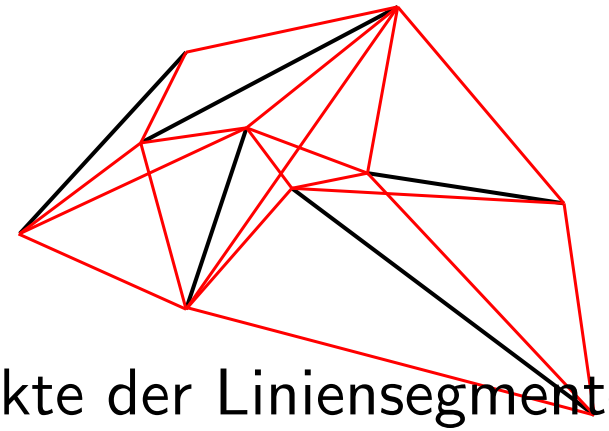


$V = \{$ alle Endpunkte der Liniensegmente in $L \}$

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

- Sichtbarkeitsgraph einer Menge von Polygonen P_i

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

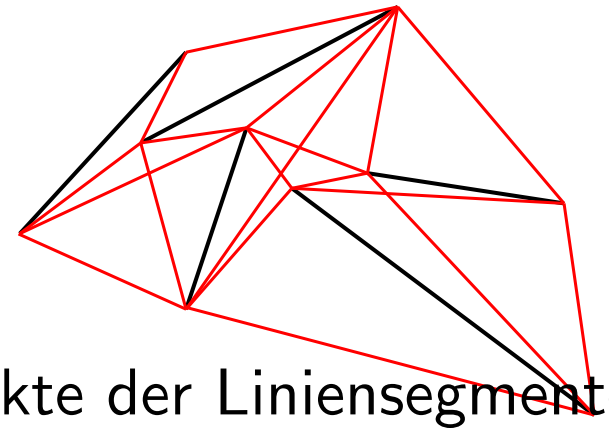


$V = \{$ alle Endpunkte der Liniensegmente in $L \}$

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

- Sichtbarkeitsgraph einer Menge von Polygonen P_i
- $\texttt{VisG}(L)$ mit $L = \{$ Kanten der $P_i \}$

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

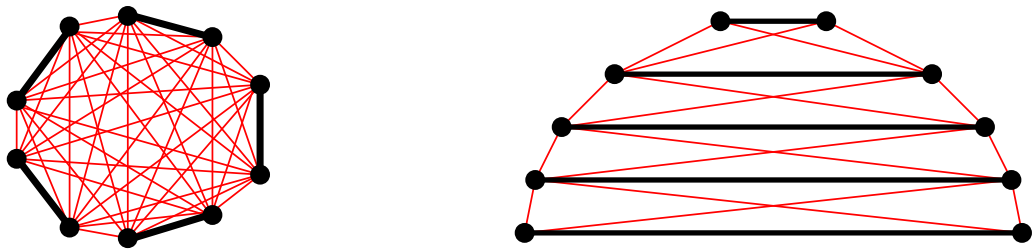


$V = \{$ alle Endpunkte der Liniensegmente in $L \}$

$E = \{ (p, q) | p, q \in V,$ $\overline{pq}$ 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

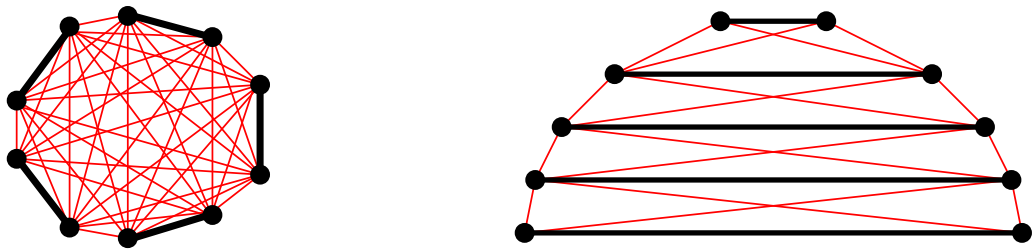
Komplexität Sichtbarkeitsgraph $\texttt{VisG}(L) = (V, E)$



(i) (ii)

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

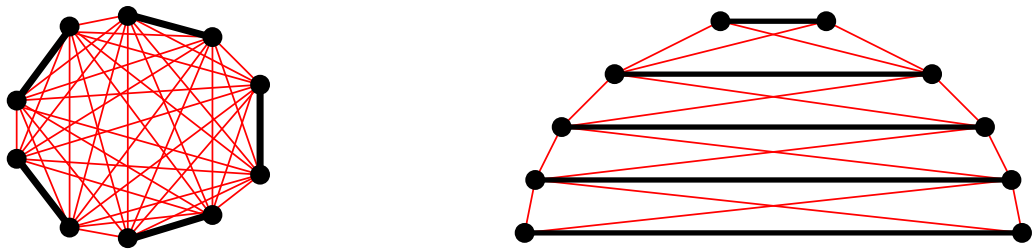
- n Liniensegmente



(i) (ii)

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

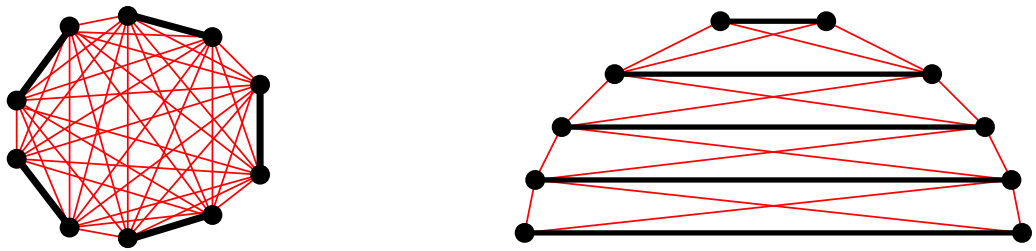
- n Liniensegmente
- $\Omega(n^2)$,



(i) (ii)

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

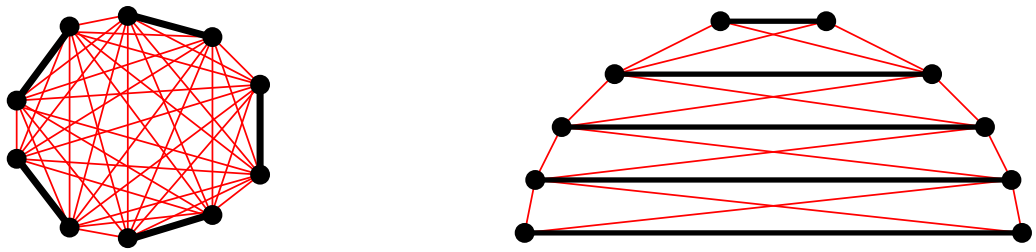
- n Liniensegmente
- $\Omega(n^2)$, $O(n^2)$,



(i) (ii)

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

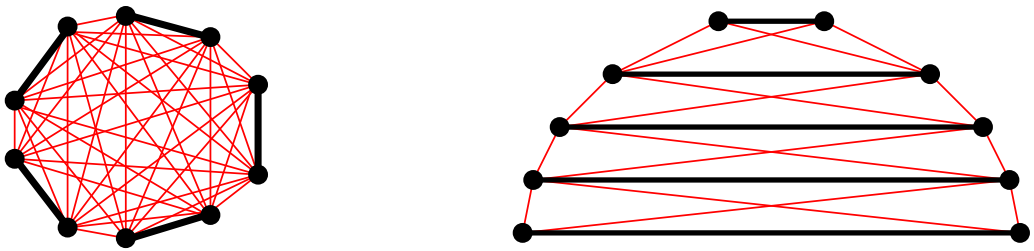
- n Liniensegmente
- $\Omega(n^2)$, $O(n^2)$, $\Theta(n^2)$



(i) (ii)

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

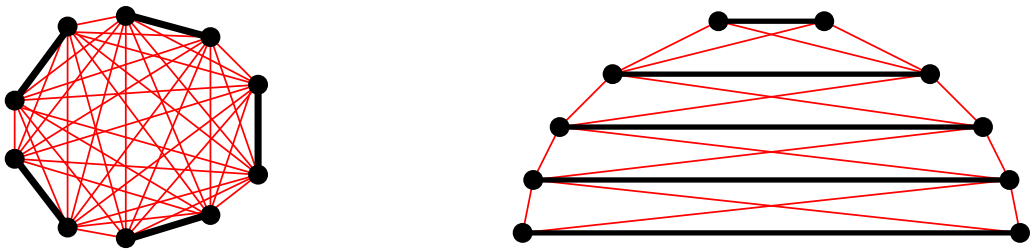
- n Liniensegmente
- $\Omega(n^2)$, $O(n^2)$, $\Theta(n^2)$
- Manchmal aber auch in $O(n)$



(i) (ii)

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

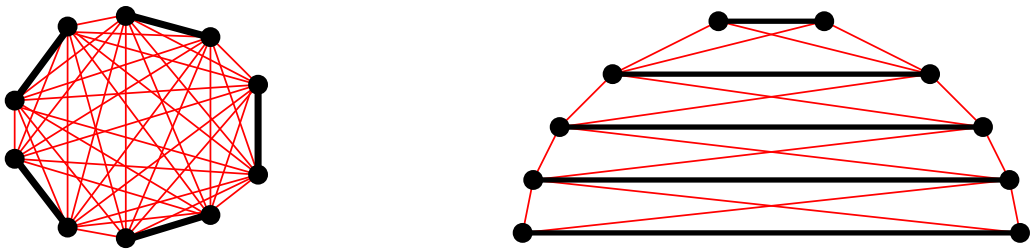
- n Liniensegmente
- $\Omega(n^2)$, $O(n^2)$, $\Theta(n^2)$
- Manchmal aber auch in $O(n)$
- Laufzeit Berechnung bestenfalls $O(n^2)$,



(i) (ii)

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

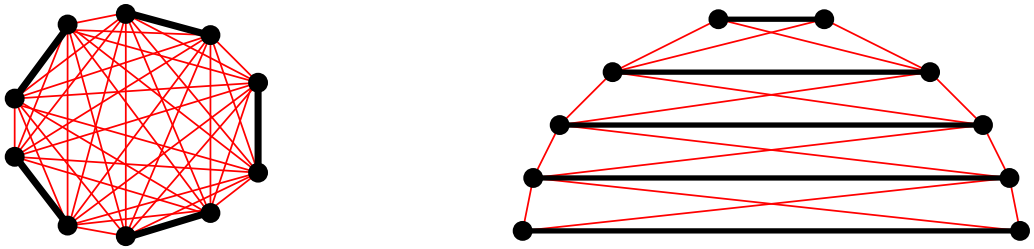
- n Liniensegmente
- $\Omega(n^2)$, $O(n^2)$, $\Theta(n^2)$
- Manchmal aber auch in $O(n)$
- Laufzeit Berechnung bestenfalls $O(n^2)$, oder Case-sensitiv



(i) (ii)

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

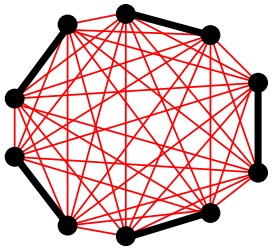
- n Liniensegmente
- $\Omega(n^2)$, $O(n^2)$, $\Theta(n^2)$
- Manchmal aber auch in $O(n)$
- Laufzeit Berechnung bestenfalls $O(n^2)$, oder Case-sensitiv
- Obere Schranke durch konkretes Beispiel



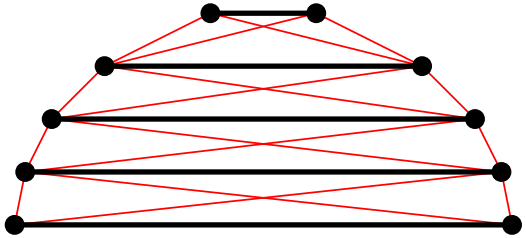
(i) (ii)

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
- Obere Schranke durch konkretes Beispiel
- Untere Schranke durch Nachdenken

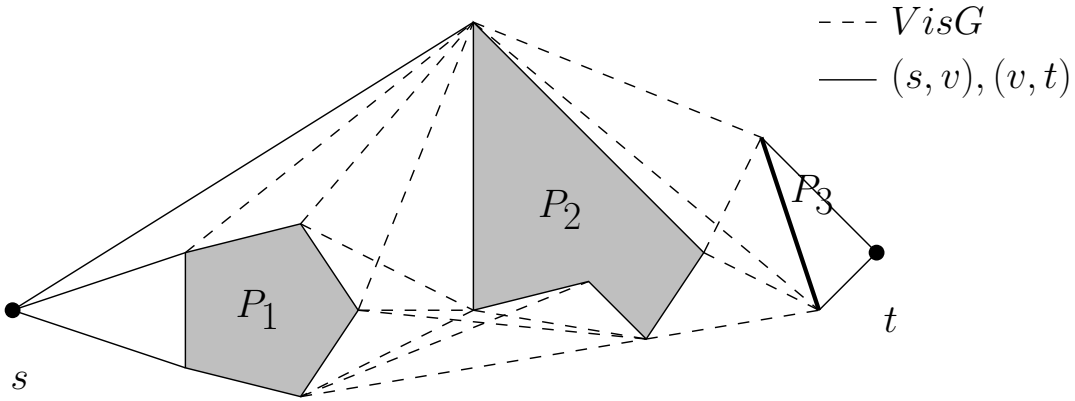


(i)



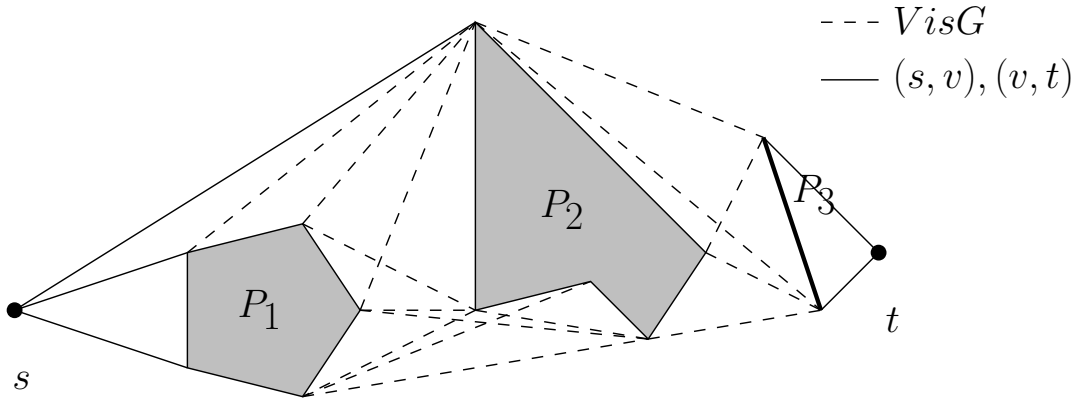
(ii)

Was bringt der Sichtbarkeitsgraph?



Was bringt der Sichtbarkeitsgraph?

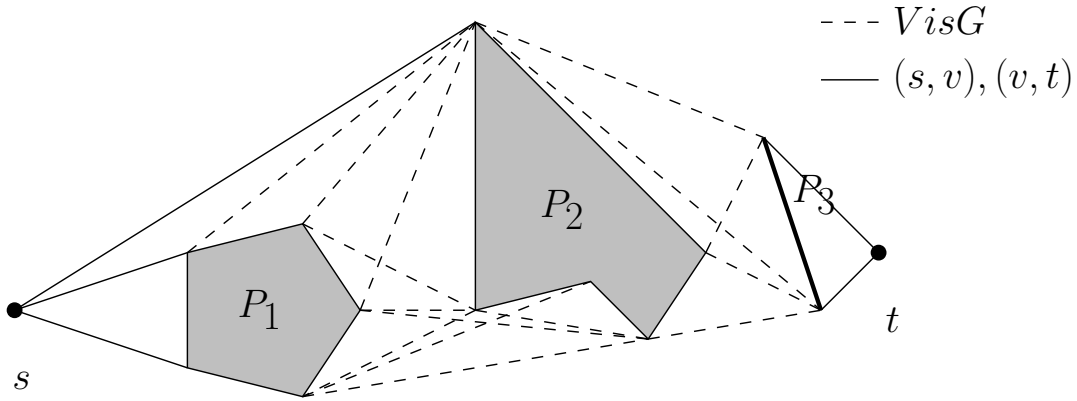
Alg. 1.1: Berechnung des kürzesten Weges mittels $\texttt{VisG}(P)$



Was bringt der Sichtbarkeitsgraph?

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

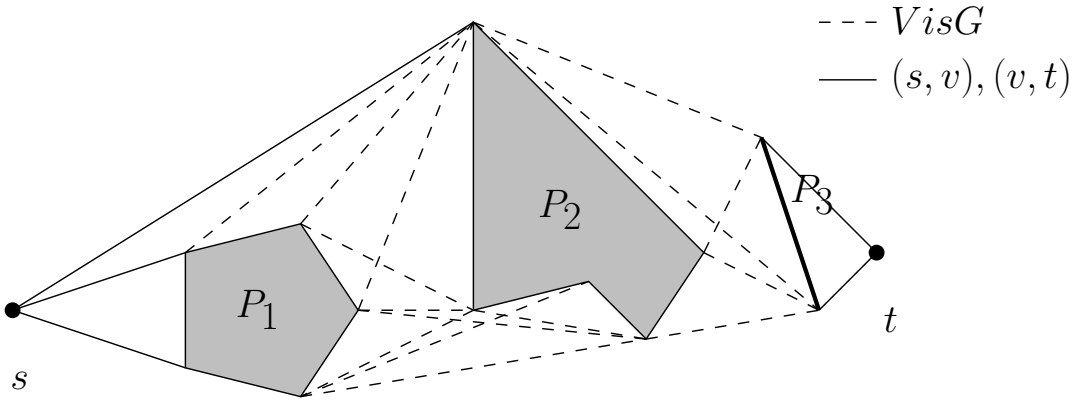
1. Sichtbarkeitsgraph (V, E) für Hindernisse:



Was bringt der Sichtbarkeitsgraph?

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

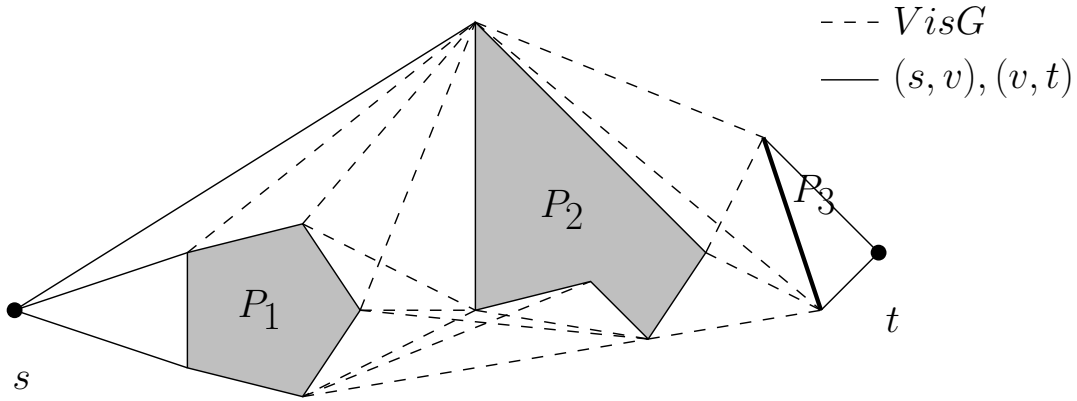
1. Sichtbarkeitsgraph (V, E) für Hindernisse: $O(n^2)$



Was bringt der Sichtbarkeitsgraph?

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

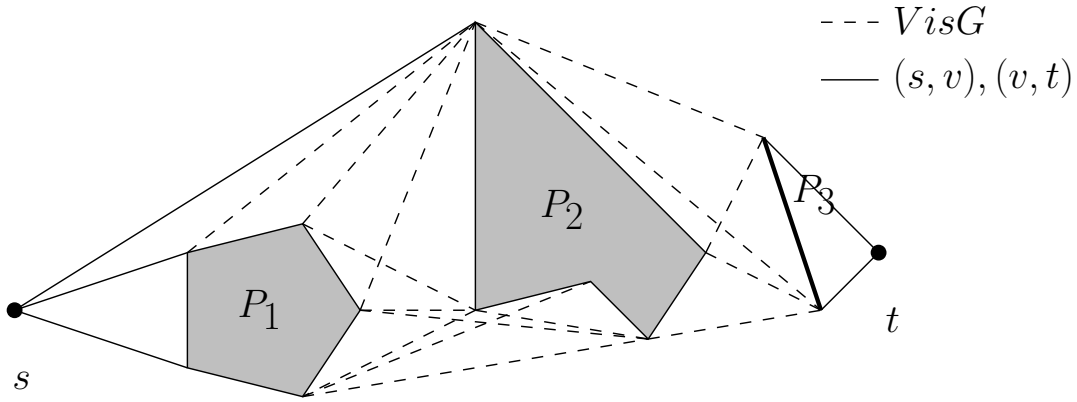
1. Sichtbarkeitsgraph (V, E) für Hindernisse: $O(n^2)$
2. Sichtbare Kanten von s aus: E_1,



Was bringt der Sichtbarkeitsgraph?

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

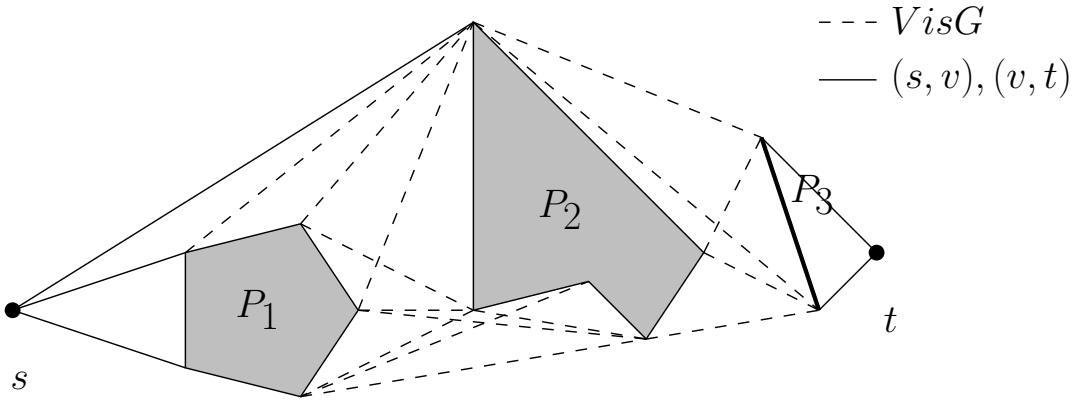
1. Sichtbarkeitsgraph (V, E) für Hindernisse: $O(n^2)$
2. Sichtbare Kanten von s aus: E_1, $O(n^2)$



Was bringt der Sichtbarkeitsgraph?

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

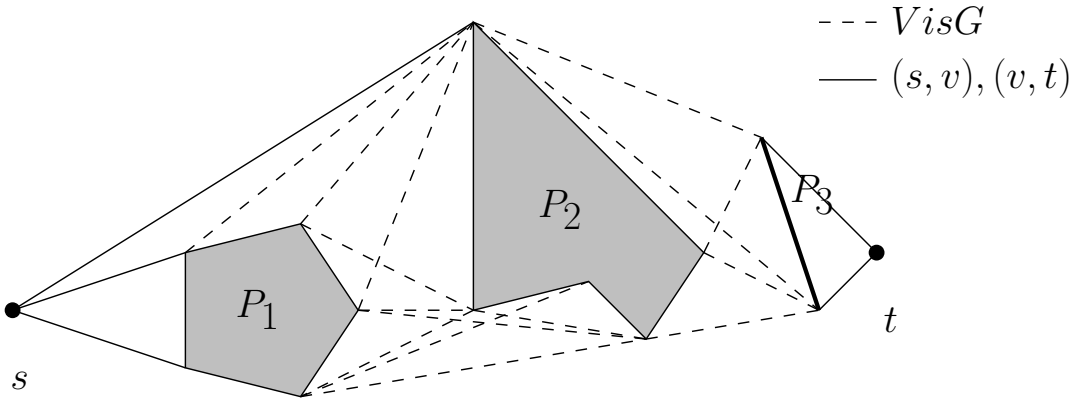
1. Sichtbarkeitsgraph (V, E) für Hindernisse: $O(n^2)$
2. Sichtbare Kanten von s aus: E_1, $O(n^2)$
3. Sichtbare Kanten von t aus: E_2,



Was bringt der Sichtbarkeitsgraph?

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

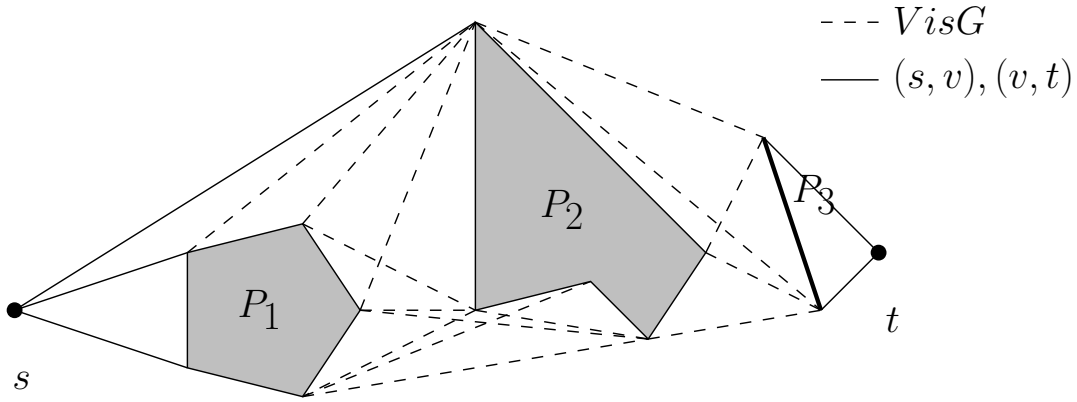
1. Sichtbarkeitsgraph (V, E) für Hindernisse: $O(n^2)$
2. Sichtbare Kanten von s aus: E_1, $O(n^2)$
3. Sichtbare Kanten von t aus: E_2, $O(n^2)$



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 $(V \cup \{s, t\}, E \cup E_1 \cup E_2)$

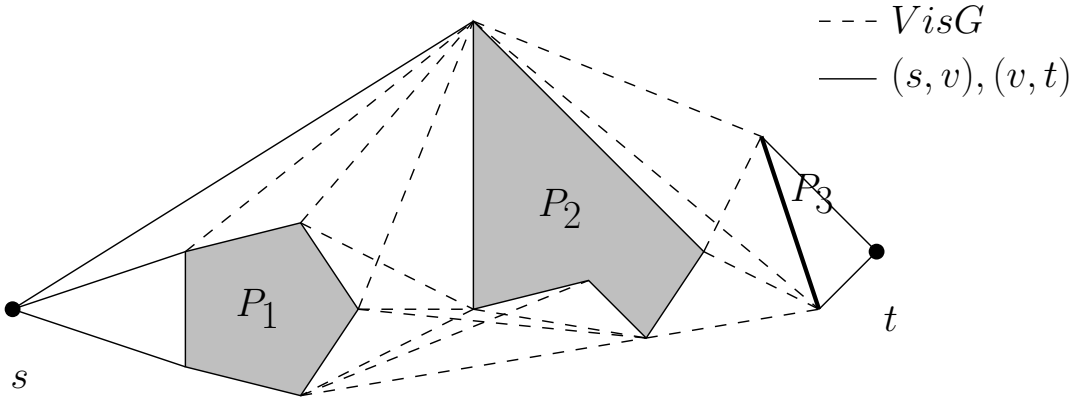


Was bringt der Sichtbarkeitsgraph?

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.:

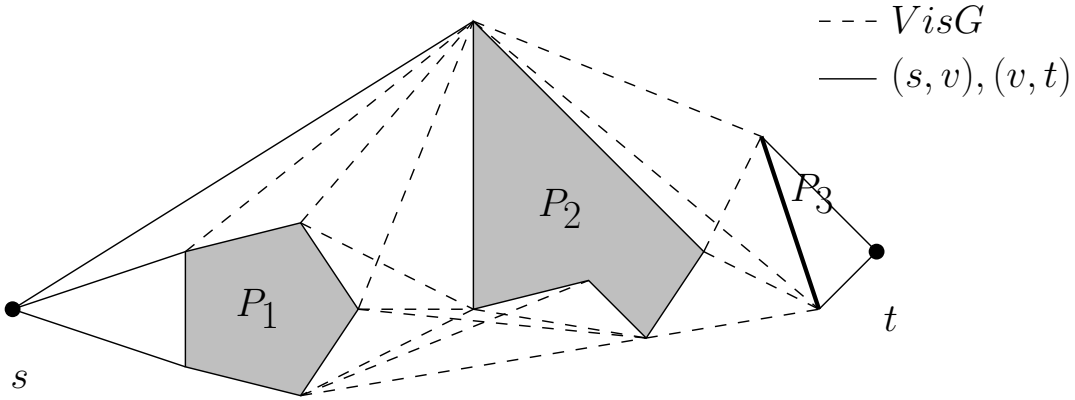


Was bringt der Sichtbarkeitsgraph?

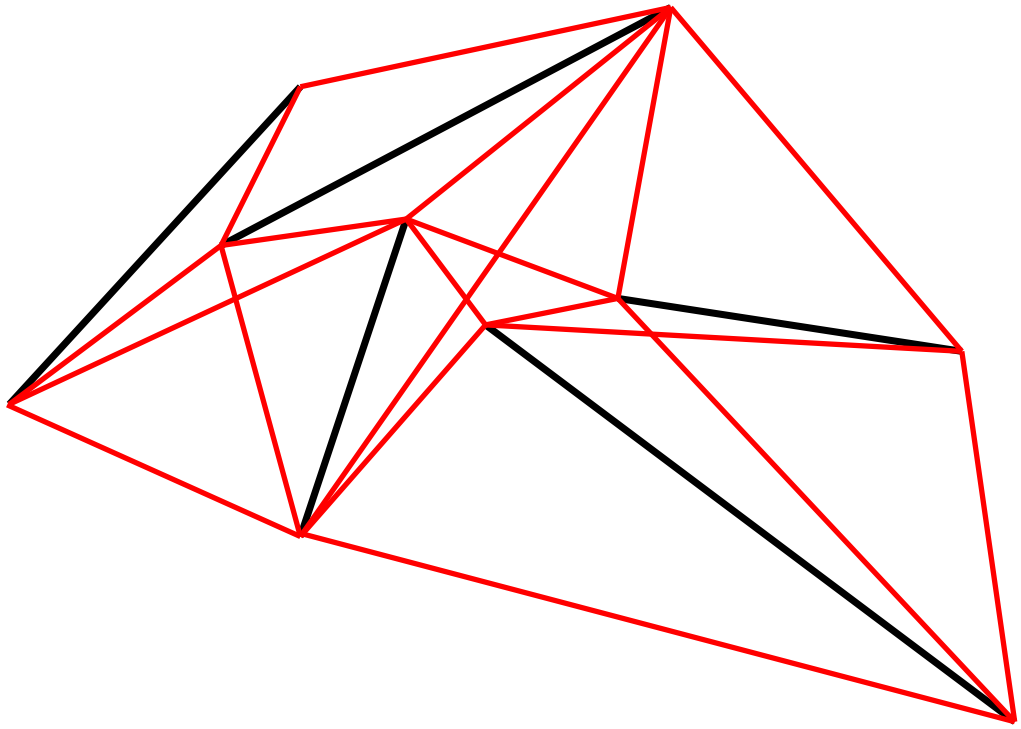
Alg. 1.1: Berechnung des kürzesten Weges mittels $\texttt{VisG}(P)$

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

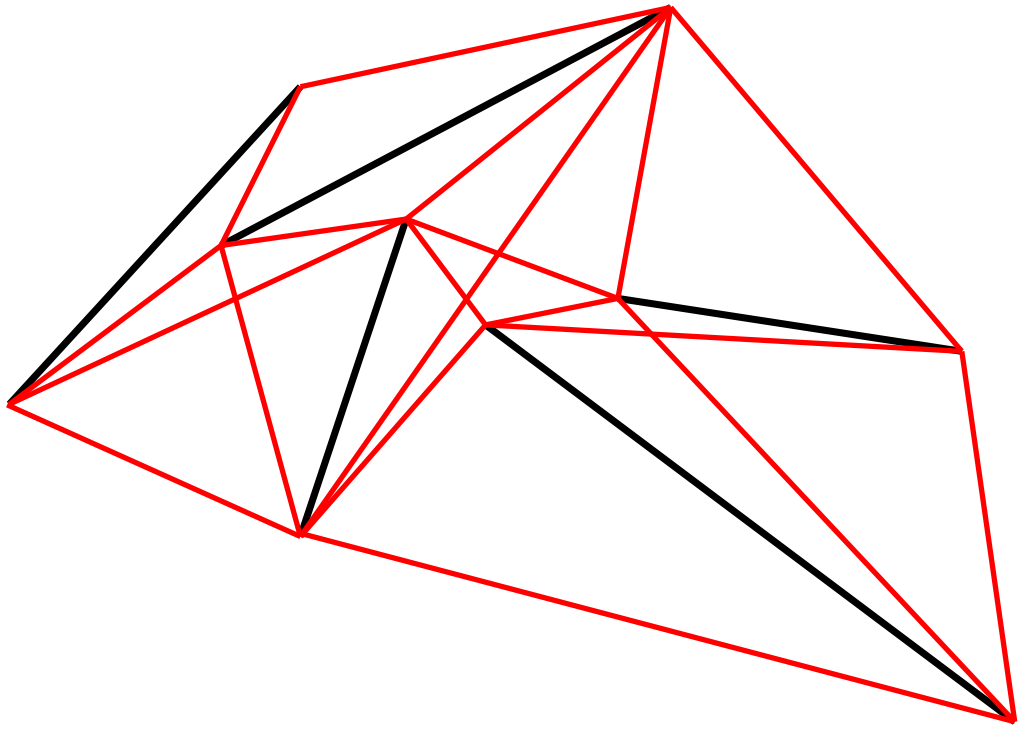


Sichtbarkeitsgraph für Menge von Liniensegmenten



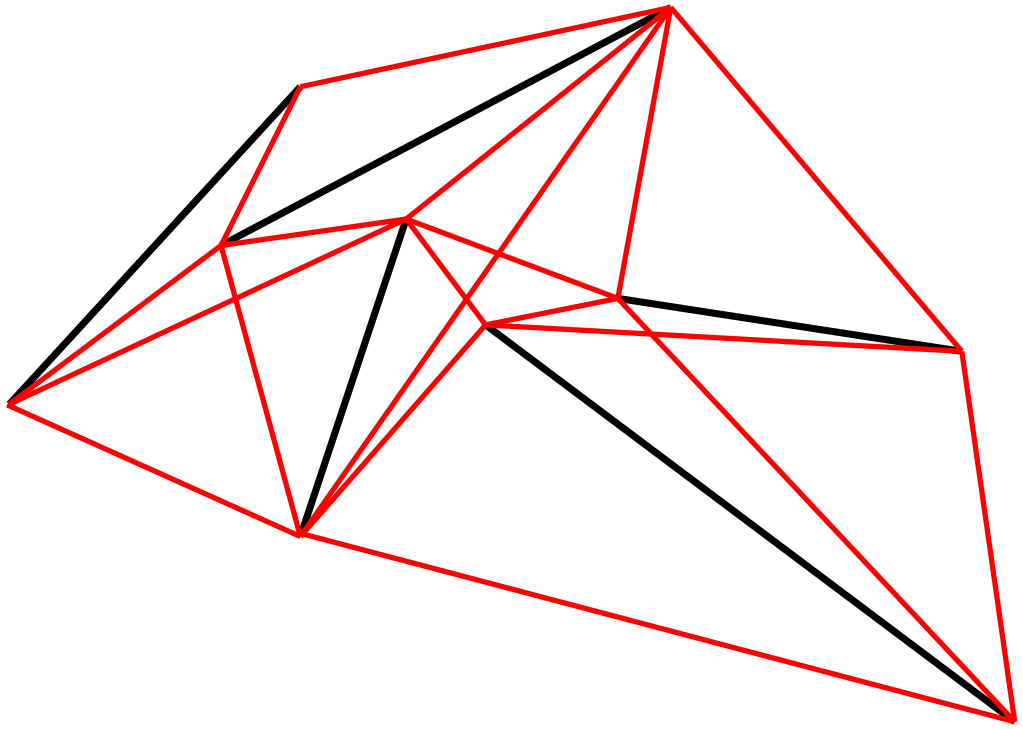
Sichtbarkeitsgraph für Menge von Liniensegmenten

- Naiv: $O(n^3)$



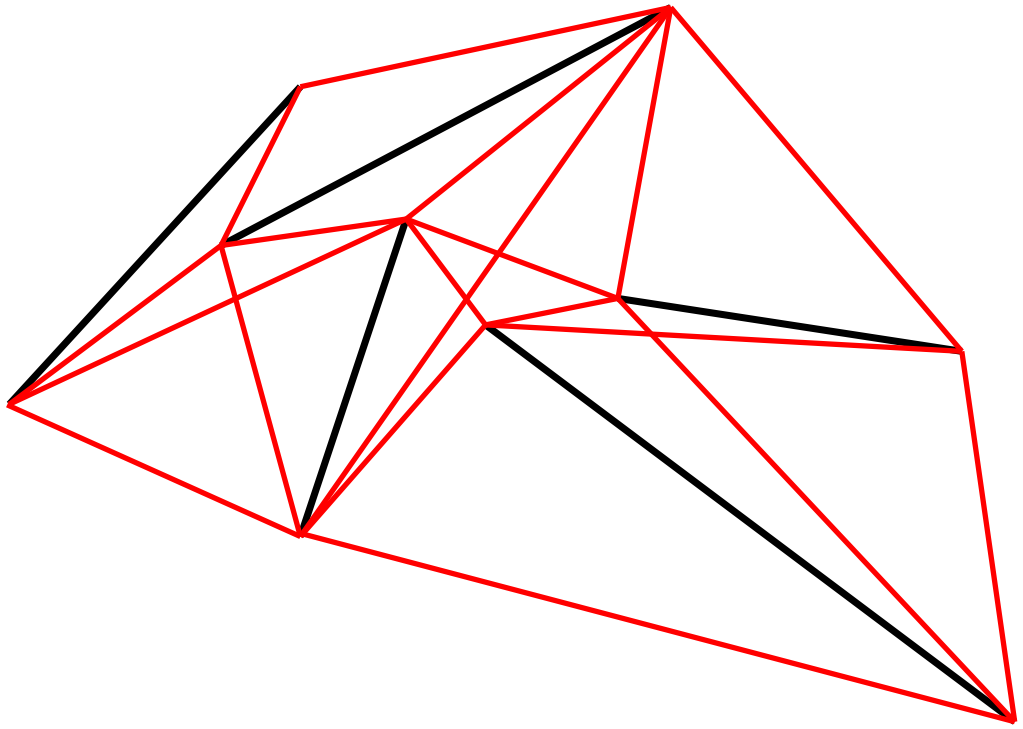
Sichtbarkeitsgraph für Menge von Liniensegmenten

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



Sichtbarkeitsgraph für Menge von Liniensegmenten

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



Sweep Technik

Sweep Technik

- Ausfegen der Ebene mit Sweepline

Sweep Technik

- Ausfegen der Ebene mit Sweepline
- Komplexitätsreduktion: 2-Dim nach 1-Dim

Sweep Technik

- Ausfegen der Ebene mit Sweepline
- Komplexitätsreduktion: 2-Dim nach 1-Dim
- Sortieren und gem. Sort. fegen

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

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

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

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)

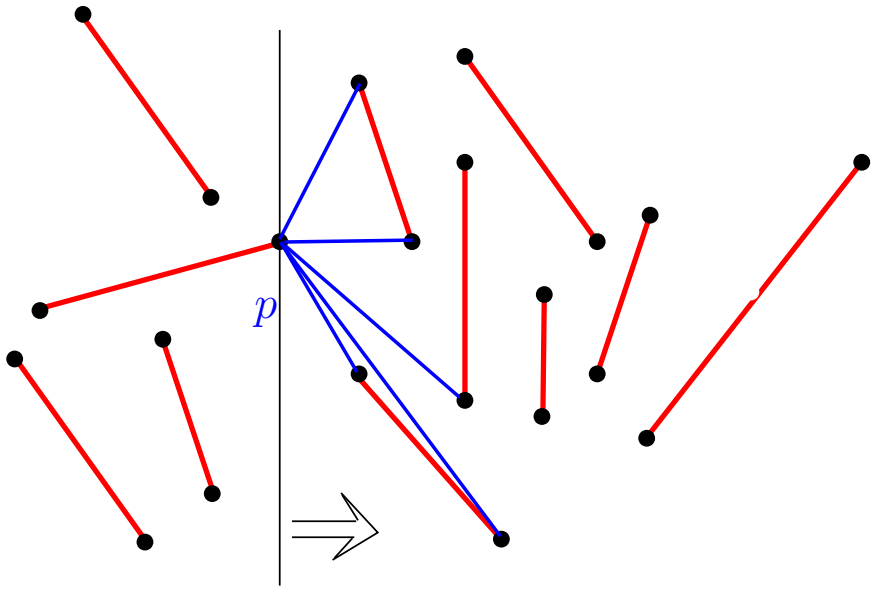
Sweep Technik

- 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

Sweep Technik

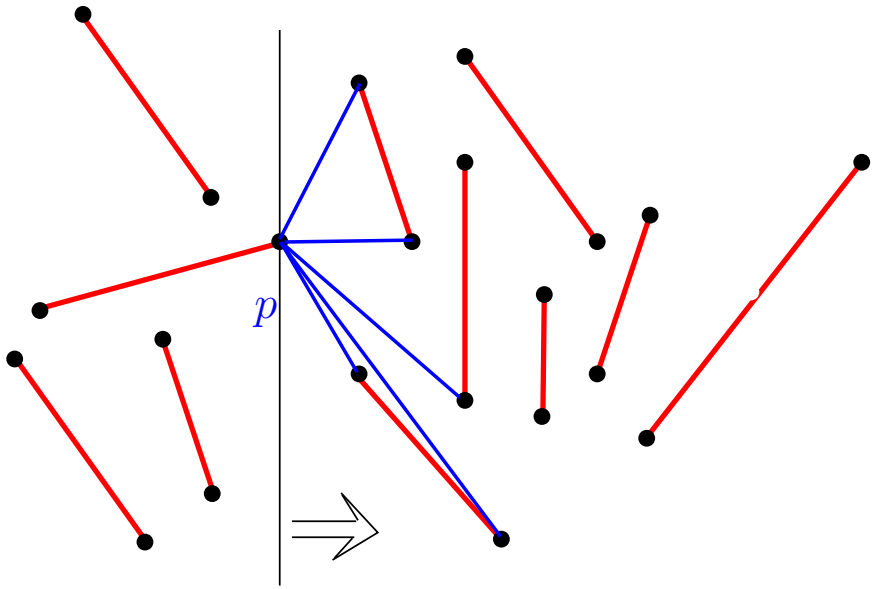
- Ausfegen der Ebene mit Sweepline
- Komplexitätsreduktion: 2-Dim nach 1-Dim
- Sortieren und gem. Sort. fegen
- SSS (Sweep-Status-Struktur): Invariante in der Nähe der Sweepline
- ES (Ereignisstruktur): Haltepunkte der Sweepline
- Ereignisverarbeitung: Aktualisierung SSS
- Laufzeitanalyse: (# Ereignisse) x Kosten(Verab)
- Korrektheit Ergebnis: Invariante erfüllt
- Einfaches Beispiel: Punkte auf Linie, Closest pair

Single-Source Sweep: Punkt p



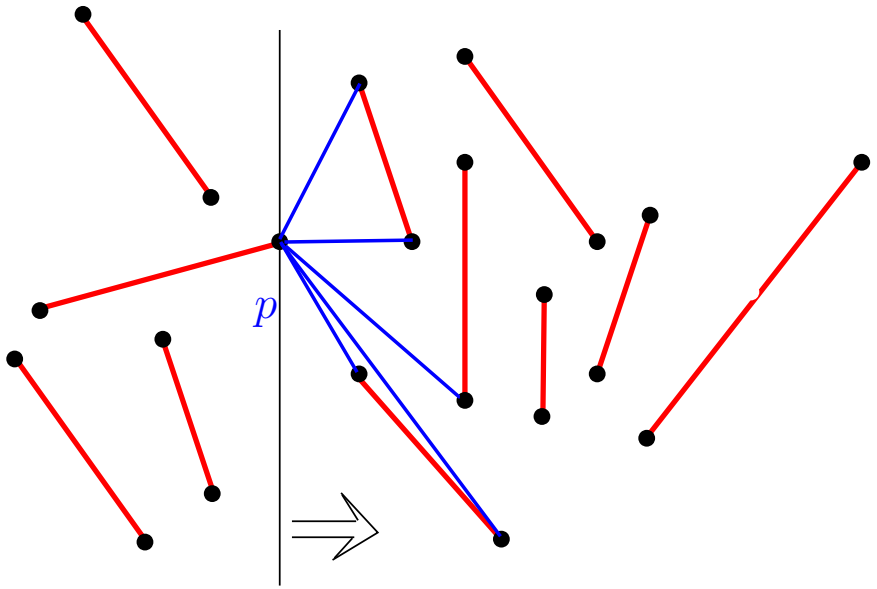
Single-Source Sweep: Punkt p

- Für jeden Endpunkt p die sichtbaren Segmente bestimmen



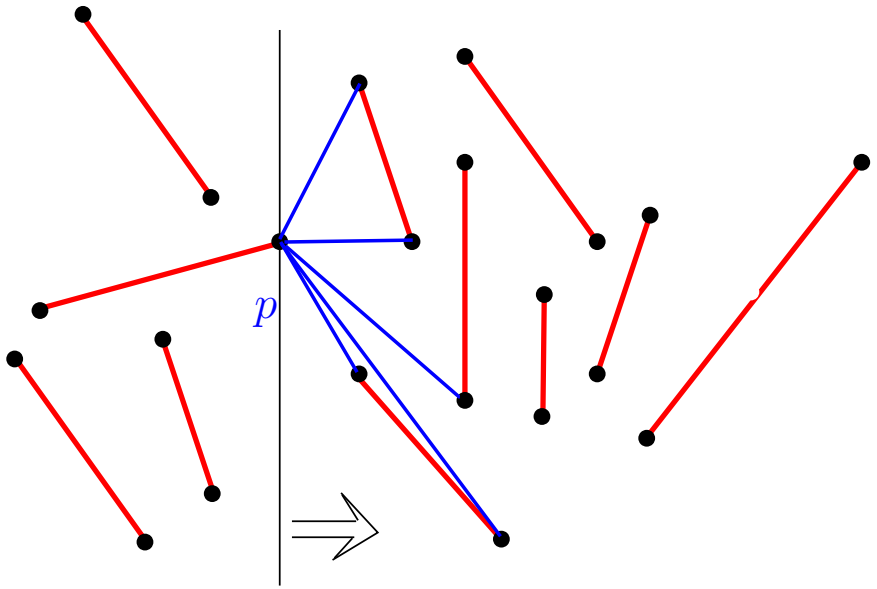
Single-Source Sweep: Punkt p

- Für jeden Endpunkt p die sichtbaren Segmente bestimmen
- Nur zu einer Seite reicht aus



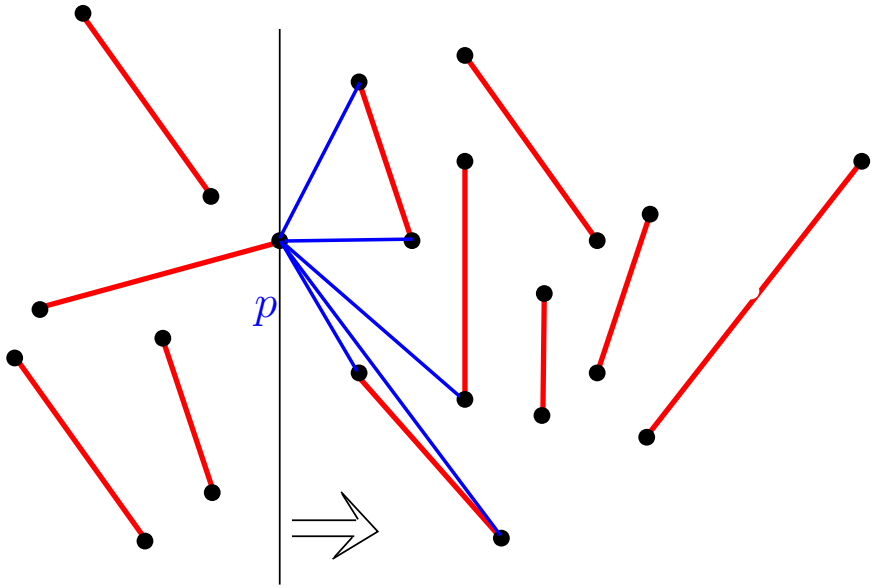
Single-Source Sweep: Punkt p

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

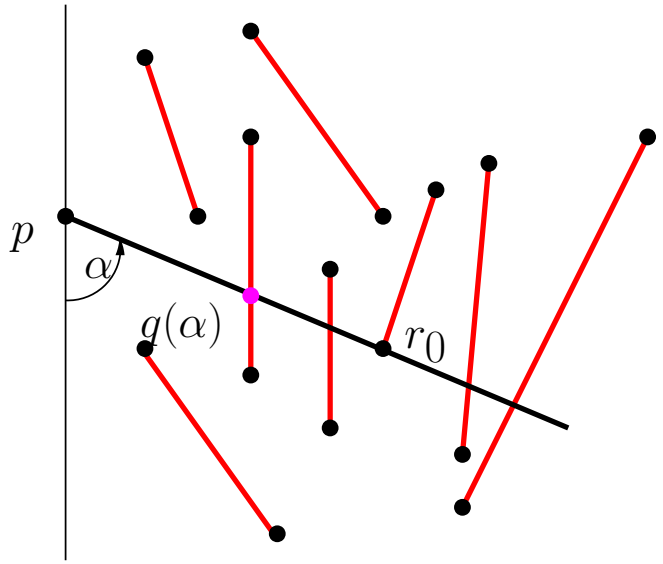


Single-Source Sweep: Punkt p

- 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

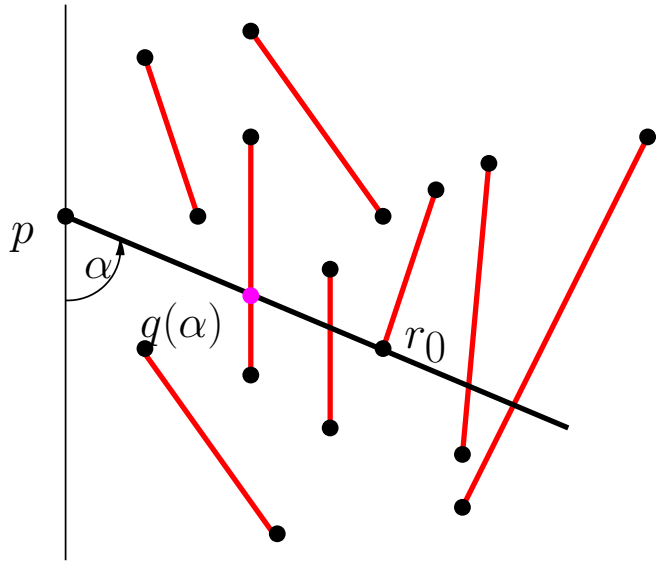


Alg. 1.2: Single-Source Sweep: Punkt p



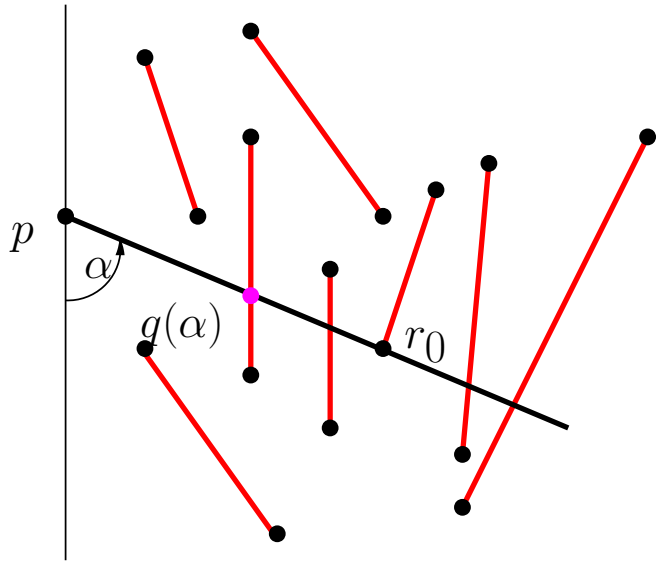
Alg. 1.2: Single-Source Sweep: Punkt p

- ES: Punkte rechts von p sortiert nach Polarkoordinaten



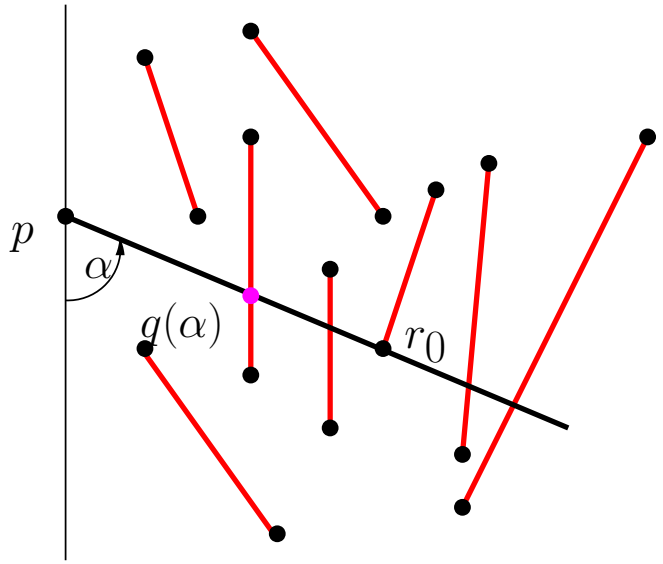
Alg. 1.2: Single-Source Sweep: Punkt p

- ES: Punkte rechts von p sortiert nach Polarkoordinaten
- SSS: Balanc. Baum (Liste der Segmente)/dynam. Schlüssel (Entfernung)



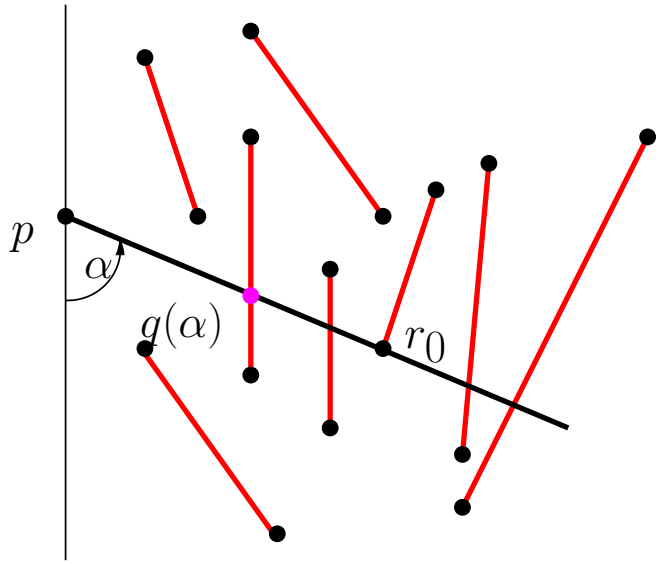
Alg. 1.2: Single-Source Sweep: Punkt p

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

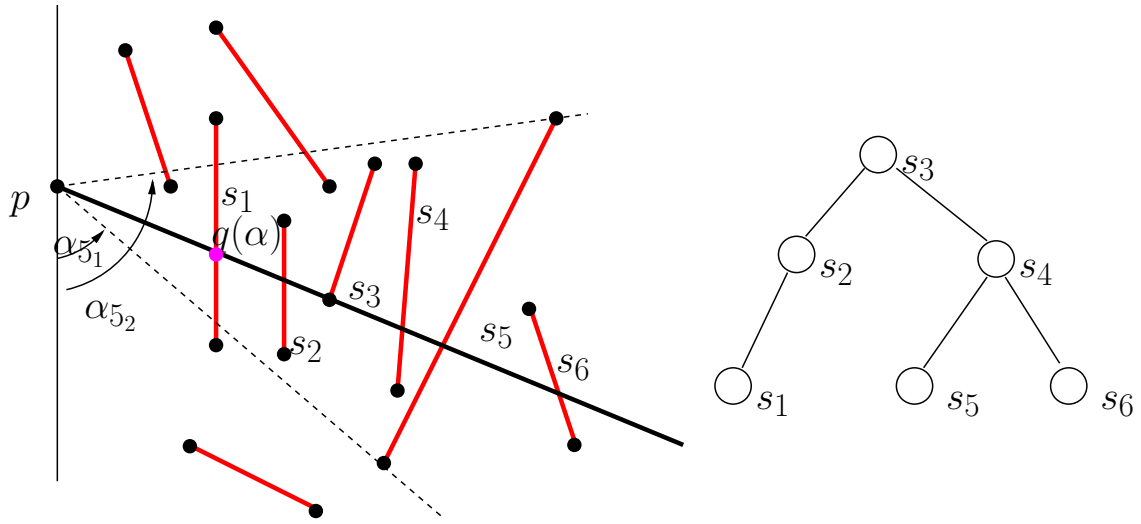


Alg. 1.2: Single-Source Sweep: Punkt p

- 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

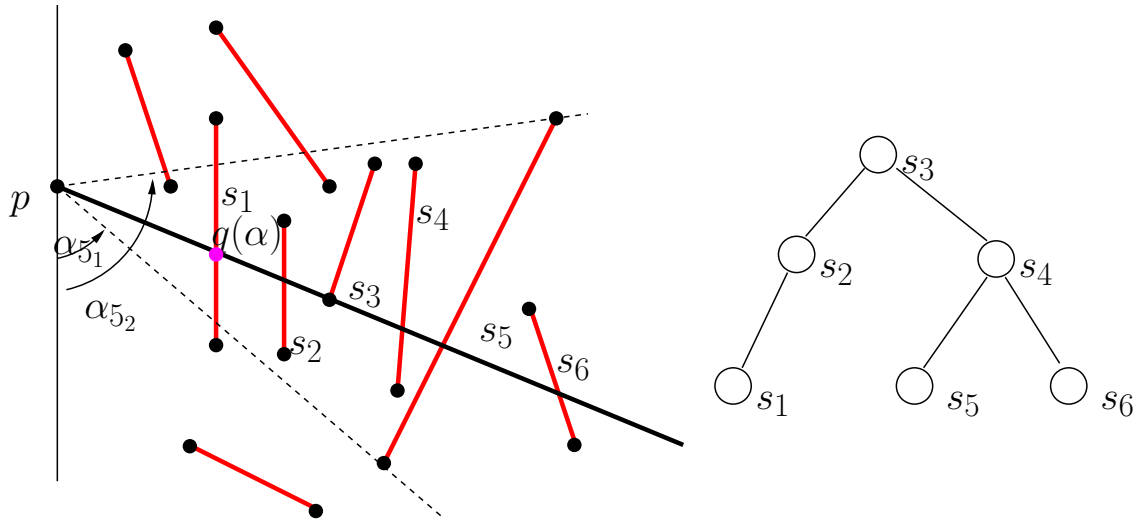


Single-Source Sweep SSS: Balancierter Baum



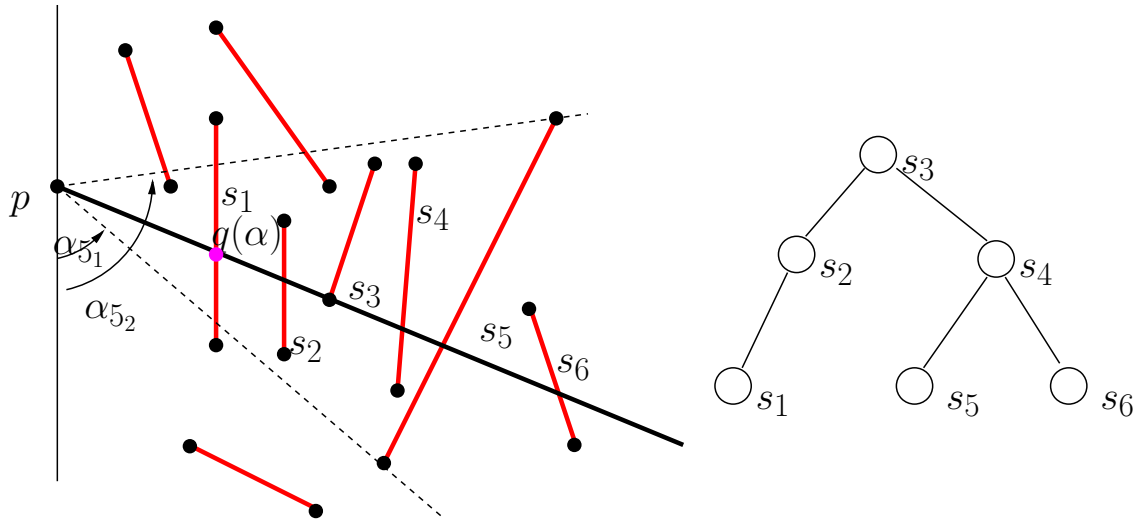
Single-Source Sweep SSS: Balancierter Baum

- Dynamische Schlüssel, Liniensegmente



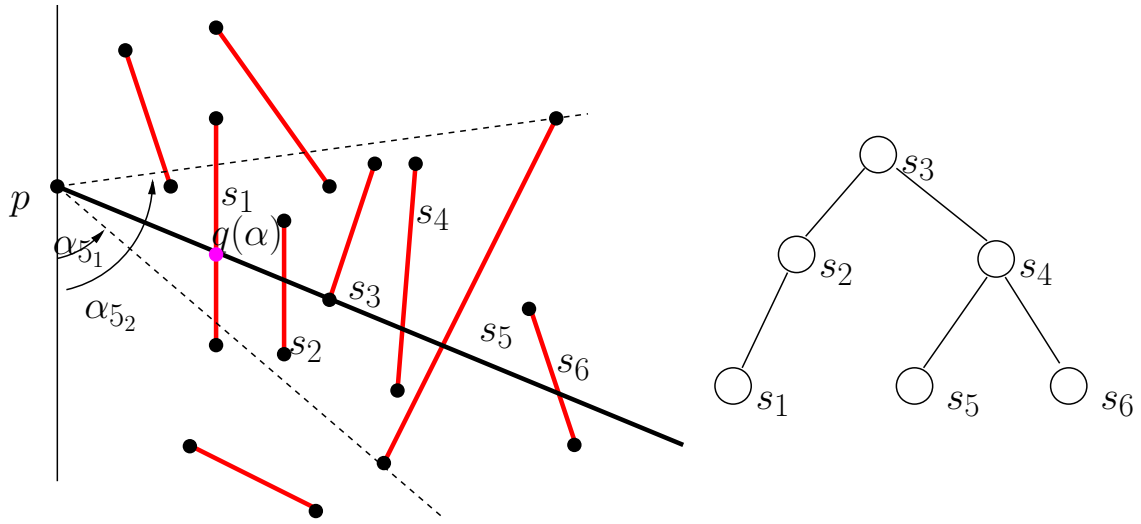
Single-Source Sweep SSS: Balancierter Baum

- Dynamische Schlüssel, Liniensegmente
- Abhängig vom Winkel α, Distanz zum Liniensegment



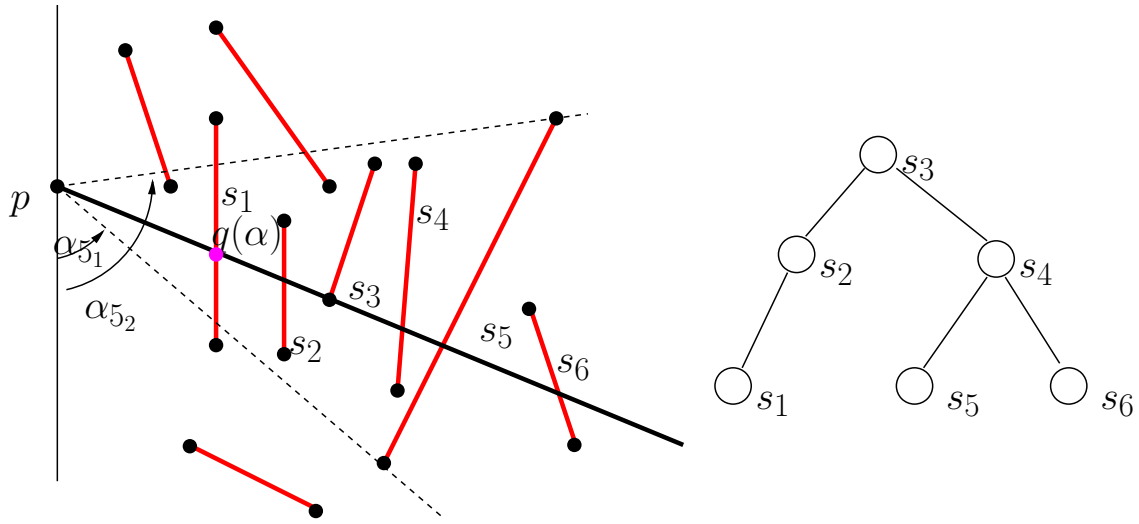
Single-Source Sweep SSS: Balancierter Baum

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



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

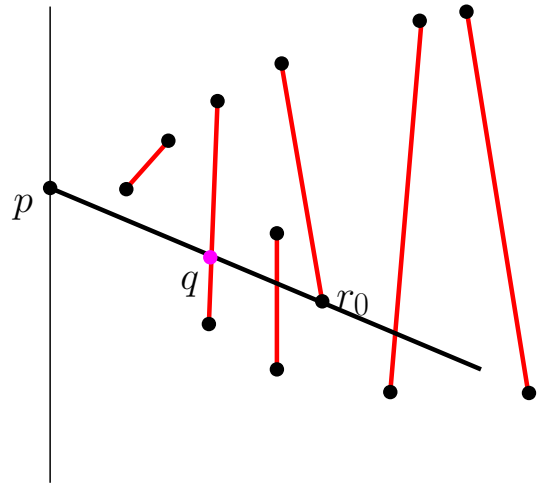


Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

Nächster Punkt r_i: Immer in folgender Reihenfolge

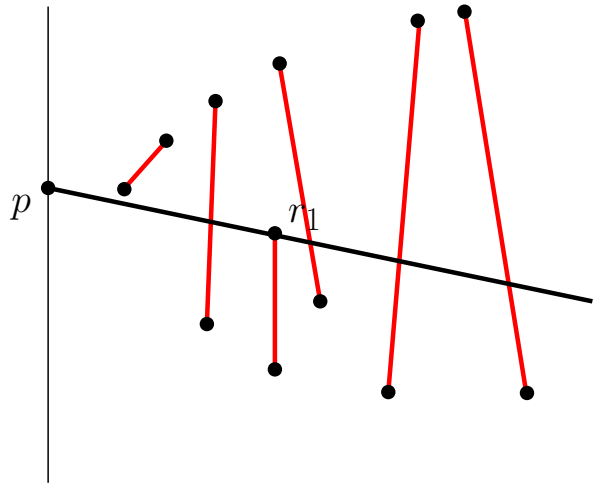
1. Falls r_i Startpunkt $\Rightarrow$ Füge Segment ein: $O(\log n)$
2. Falls $r_i = q(\alpha) \Rightarrow$ Ausgabe (p, r_i): $O(1)$
3. Falls r_i Endpunkt $\Rightarrow$ Lösche Segment: $O(\log n)$



Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

Nächster Punkt r_i: Immer in folgender Reihenfolge

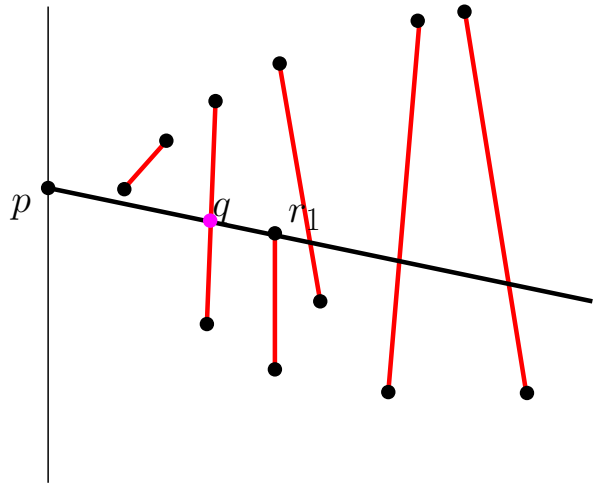
1. Falls r_i Startpunkt $\Rightarrow$ Füge Segment ein: $O(\log n)$
2. Falls $r_i = q(\alpha) \Rightarrow$ Ausgabe (p, r_i): $O(1)$
3. Falls r_i Endpunkt $\Rightarrow$ Lösche Segment: $O(\log n)$



Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

Nächster Punkt r_i: Immer in folgender Reihenfolge

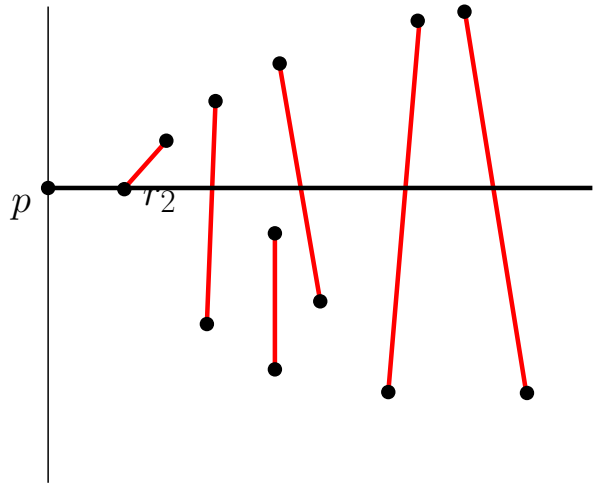
1. Falls r_i Startpunkt $\Rightarrow$ Füge Segment ein: $O(\log n)$
2. Falls $r_i = q(\alpha) \Rightarrow$ Ausgabe (p, r_i): $O(1)$
3. Falls r_i Endpunkt $\Rightarrow$ Lösche Segment: $O(\log n)$



Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

Nächster Punkt r_i: Immer in folgender Reihenfolge

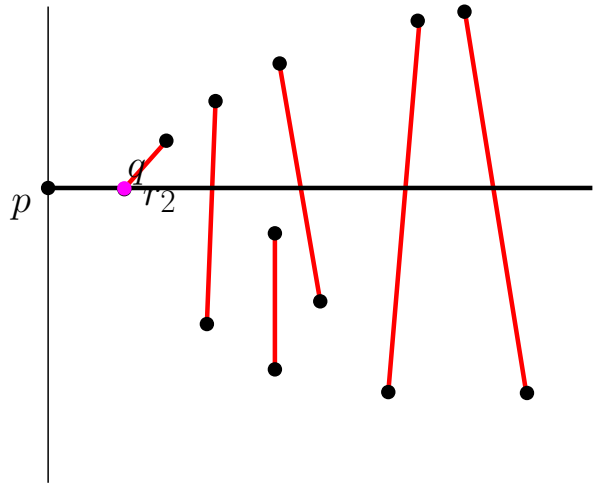
1. Falls r_i Startpunkt $\Rightarrow$ Füge Segment ein: $O(\log n)$
2. Falls $r_i = q(\alpha) \Rightarrow$ Ausgabe (p, r_i): $O(1)$
3. Falls r_i Endpunkt $\Rightarrow$ Lösche Segment: $O(\log n)$



Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

Nächster Punkt r_i: Immer in folgender Reihenfolge

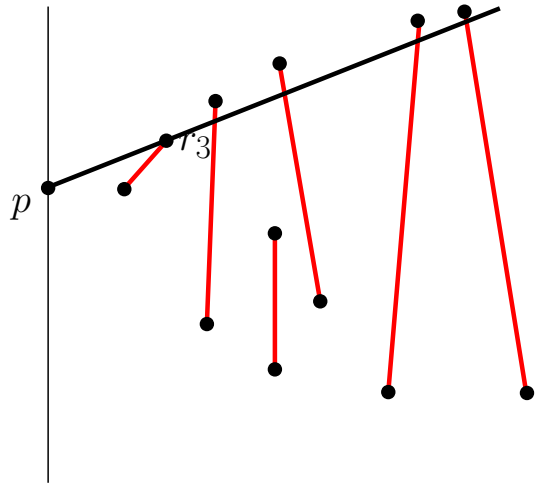
1. Falls r_i Startpunkt $\Rightarrow$ Füge Segment ein: $O(\log n)$
2. Falls $r_i = q(\alpha) \Rightarrow$ Ausgabe (p, r_i): $O(1)$
3. Falls r_i Endpunkt $\Rightarrow$ Lösche Segment: $O(\log n)$



Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

Nächster Punkt r_i: Immer in folgender Reihenfolge

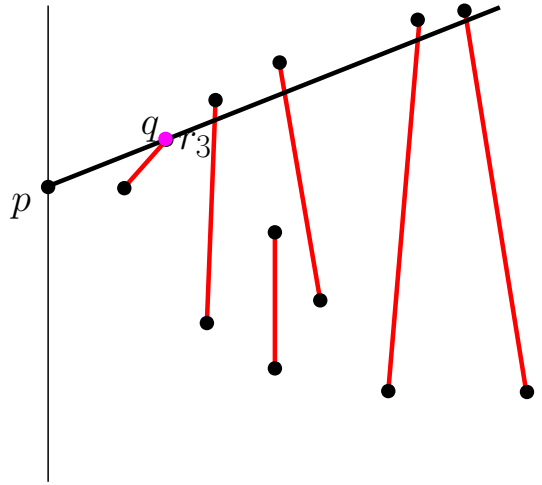
1. Falls r_i Startpunkt $\Rightarrow$ Füge Segment ein: $O(\log n)$
2. Falls $r_i = q(\alpha) \Rightarrow$ Ausgabe (p, r_i): $O(1)$
3. Falls r_i Endpunkt $\Rightarrow$ Lösche Segment: $O(\log n)$



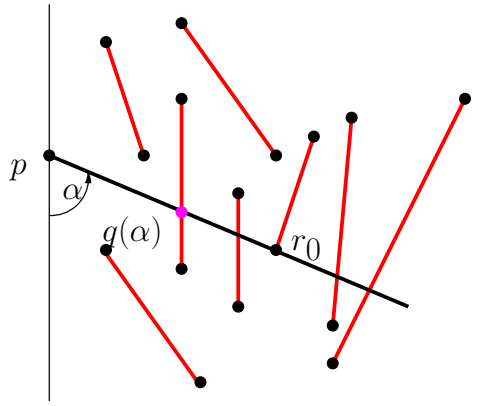
Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

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: $O(\log n)$

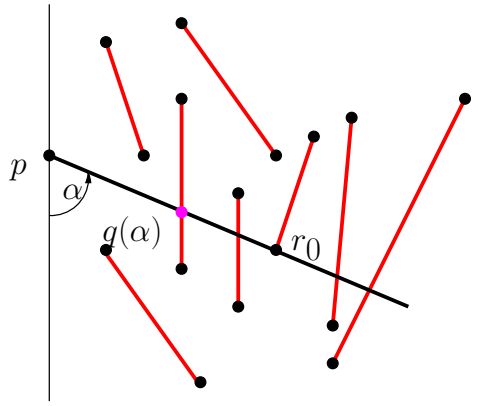


Single-Source Sweep: Korrektheit und Laufzeit



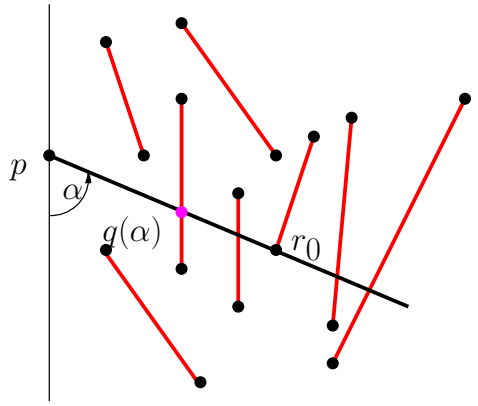
Single-Source Sweep: Korrektheit und Laufzeit

- SSS: Zu jedem Zeitpunkt ist $q(\alpha)$ sichtbarer Punkt



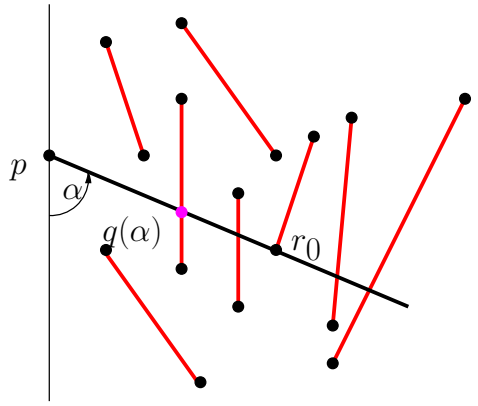
Single-Source Sweep: Korrektheit und Laufzeit

- SSS: Zu jedem Zeitpunkt ist $q(\alpha)$ sichtbarer Punkt
- Test mit aktuellem Knoten r_i



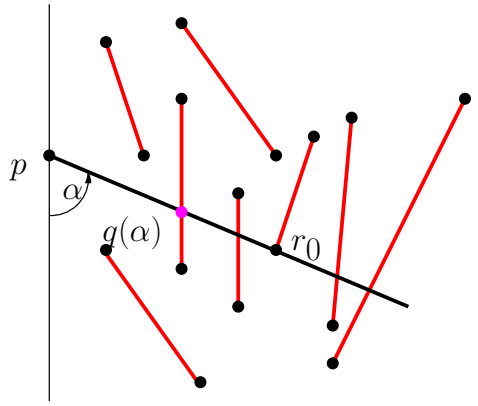
Single-Source Sweep: Korrektheit und Laufzeit

- 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)



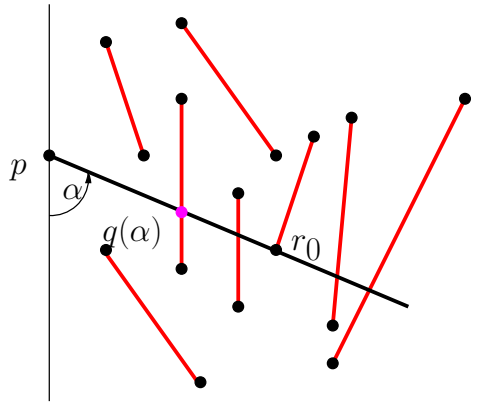
Single-Source Sweep: Korrektheit und Laufzeit

- 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



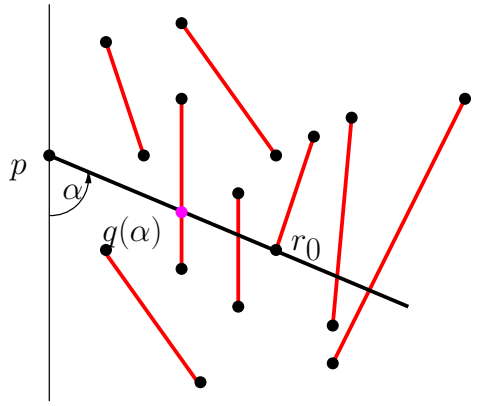
Single-Source Sweep: Korrektheit und Laufzeit

- 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: $O(n \log n)$



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?

Geht das besser?

- Laufzeit $O(n^2 \log n)$, Komplexität $\Theta(n^2)$,

Geht das besser?

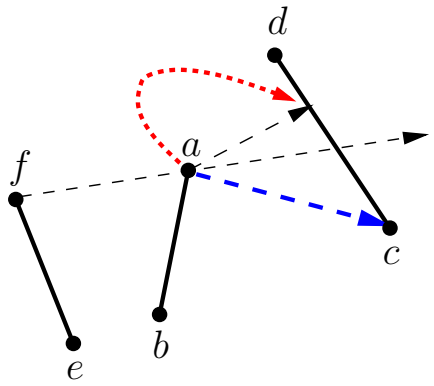
- Laufzeit $O(n^2 \log n)$, Komplexität $\Theta(n^2)$, Laufzeit $O(n^2)$?

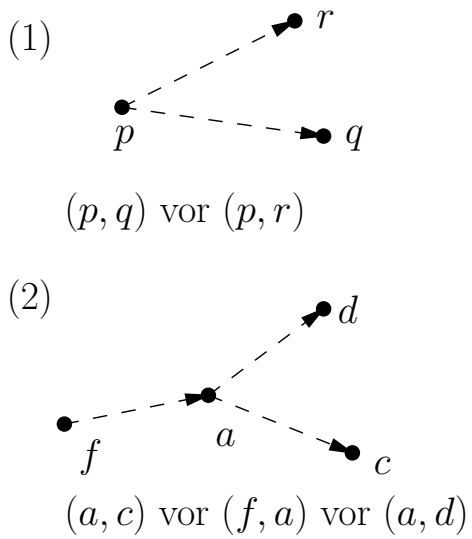
Geht das besser?

- Laufzeit $O(n^2 \log n)$, Komplexität $\Theta(n^2)$, Laufzeit $O(n^2)$?
- Simultaner Sweep, alle Segmente gleichzeitig drehen,

Geht das besser?

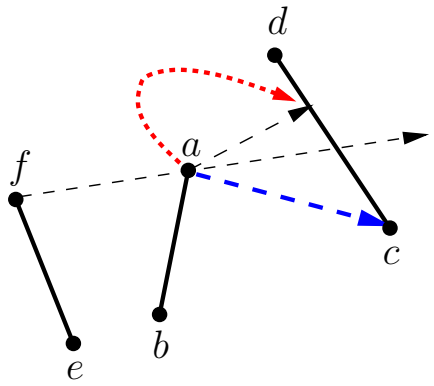
- Laufzeit $O(n^2 \log n)$, Komplexität $\Theta(n^2)$, Laufzeit $O(n^2)$?
- Simultaner Sweep, alle Segmente gleichzeitig drehen, Sicht-Informationen weiter geben

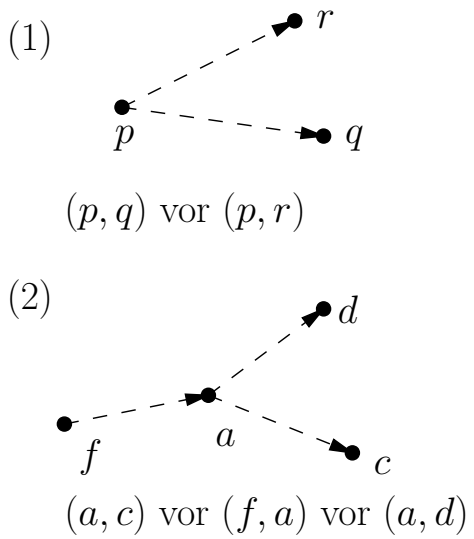




Geht das besser?

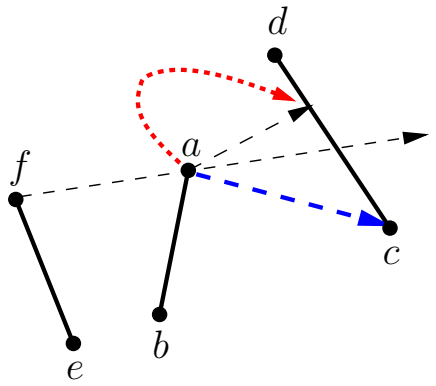
- Laufzeit $O(n^2 \log n)$, Komplexität $\Theta(n^2)$, Laufzeit $O(n^2)$?
- Simultaner Sweep, alle Segmente gleichzeitig drehen, Sicht-Informationen weiter geben
- Bearbeitungsreihenfolge: Paare von Endpunkten gemäß Steigung

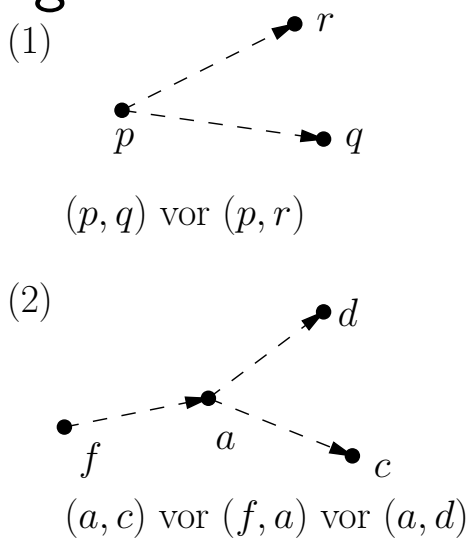




Geht das besser?

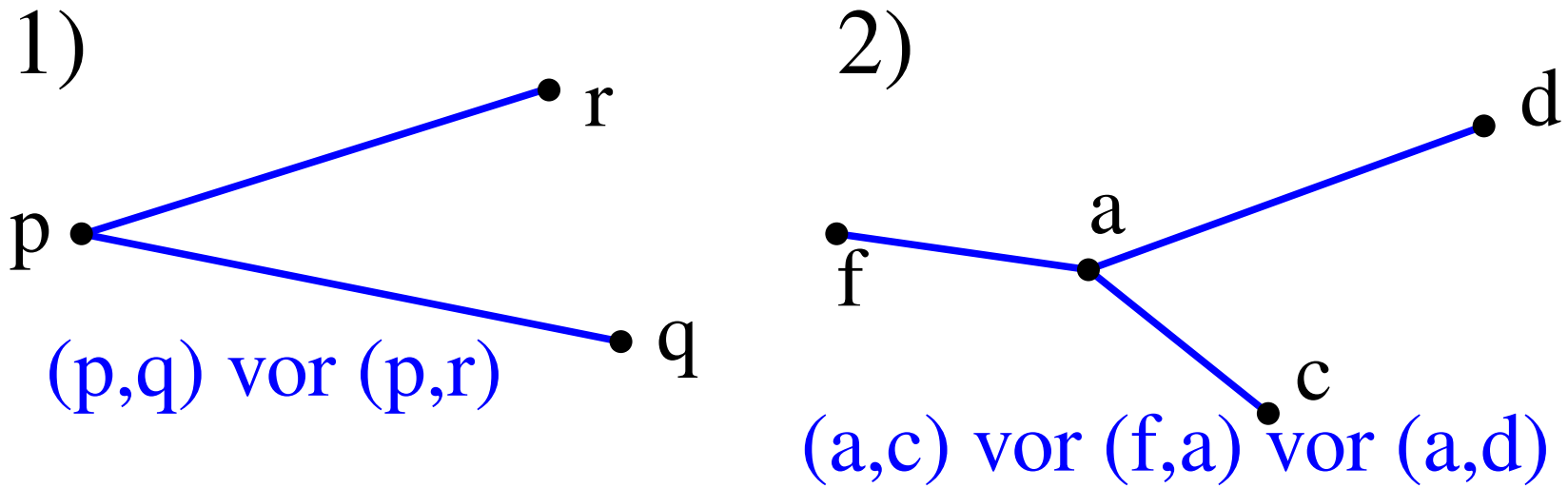
- Laufzeit $O(n^2 \log n)$, Komplexität $\Theta(n^2)$, Laufzeit $O(n^2)$?
- Simultaner Sweep, alle Segmente gleichzeitig drehen, Sicht-Informationen weiter geben
- Bearbeitungsreihenfolge: Paare von Endpunkten gemäß Steigung
- Zeiger auf momentan sichtbares Segment



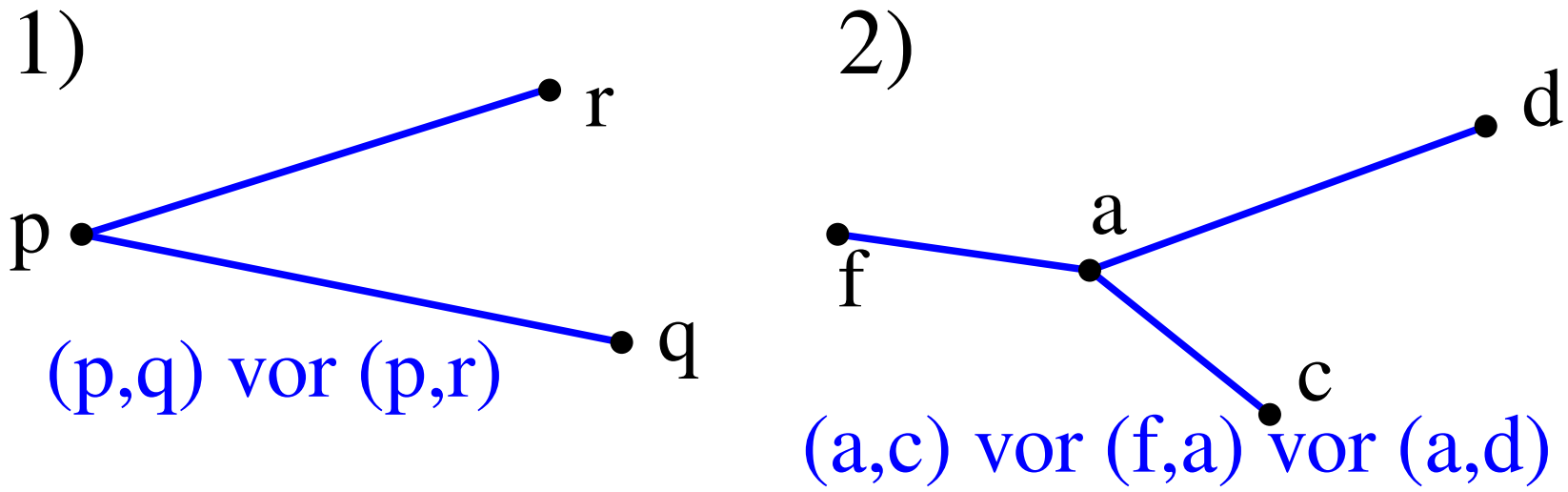


Bearbeitungsreihenfolge in $O(n^2)$

Bearbeitungsreihenfolge in $O(n^2)$

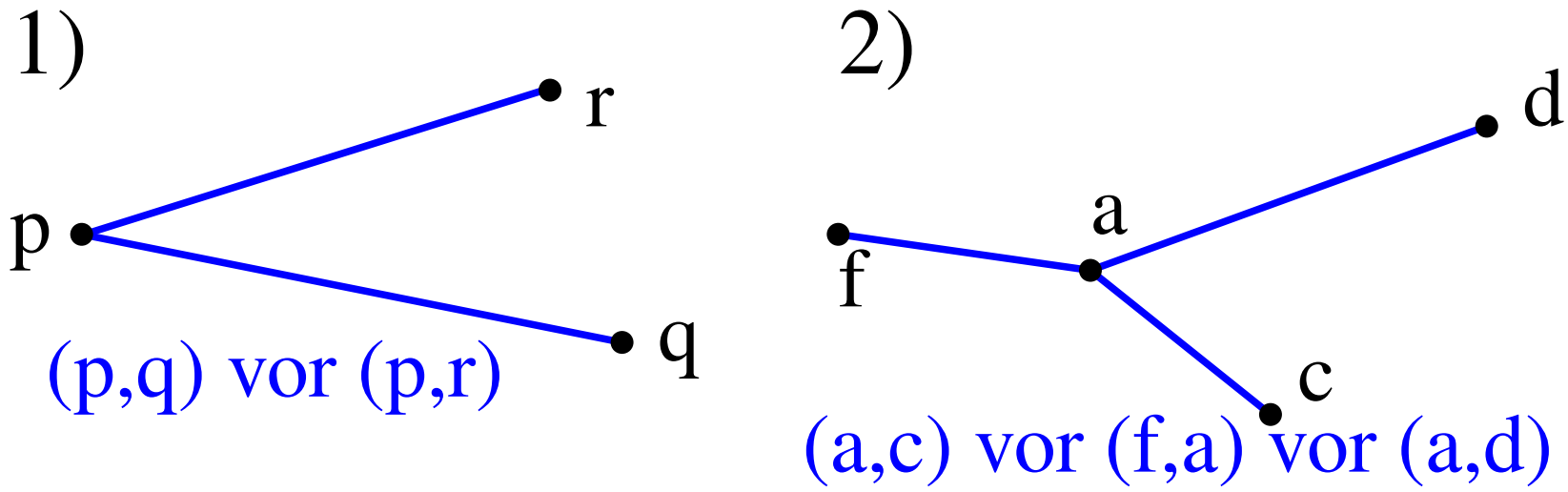


Bearbeitungsreihenfolge in $O(n^2)$



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

Bearbeitungsreihenfolge in $O(n^2)$

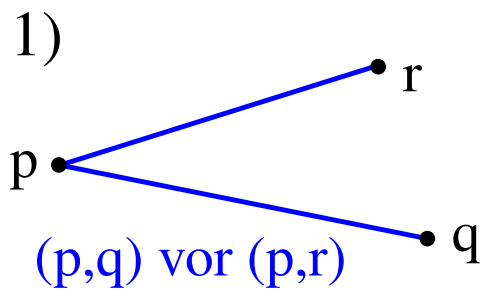


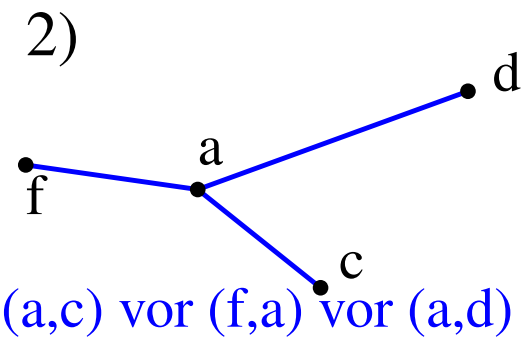
Liste von Punkten (p, q) mit $p_x \leq q_x$

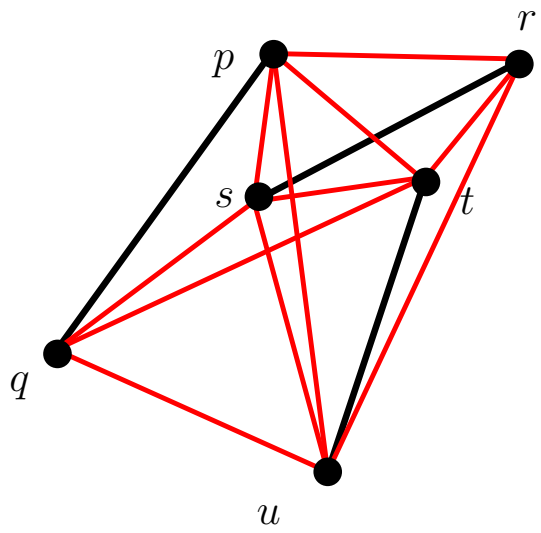
Zeigen wir später! Annahme: Ist schon gegeben für die Punktpaare!

Beispiel Bearbeitungsreihenfolge

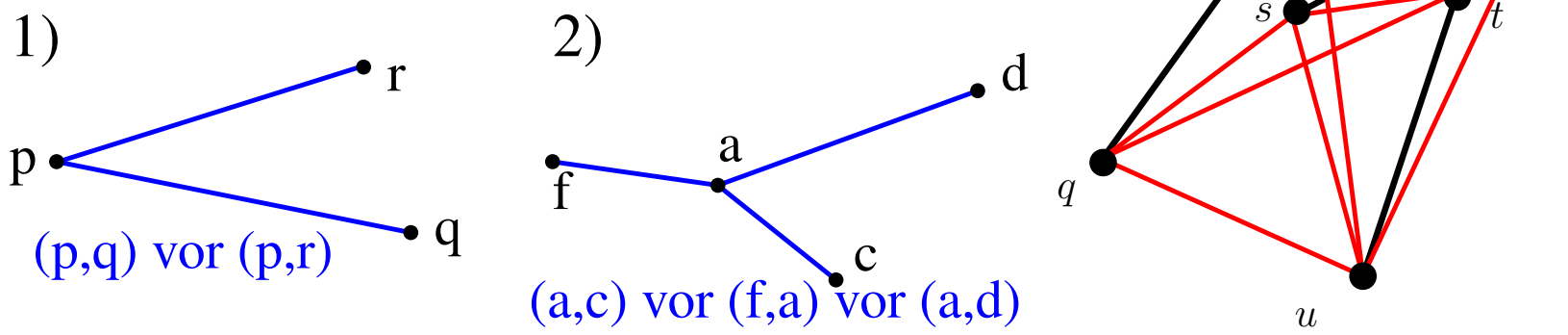
Beispiel Bearbeitungsreihenfolge







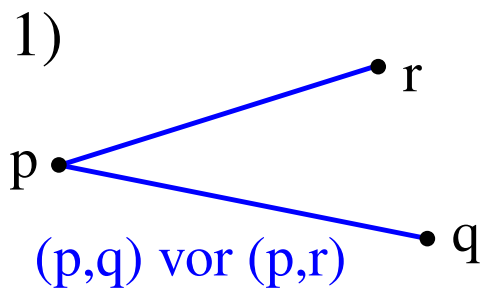
Beispiel Bearbeitungsreihenfolge

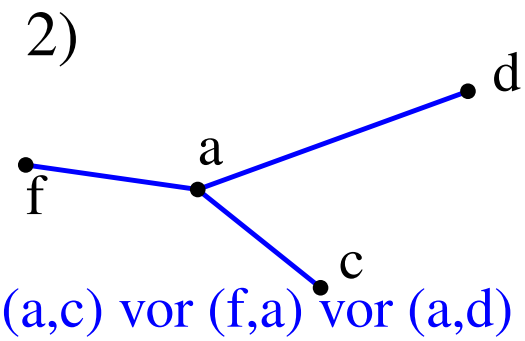


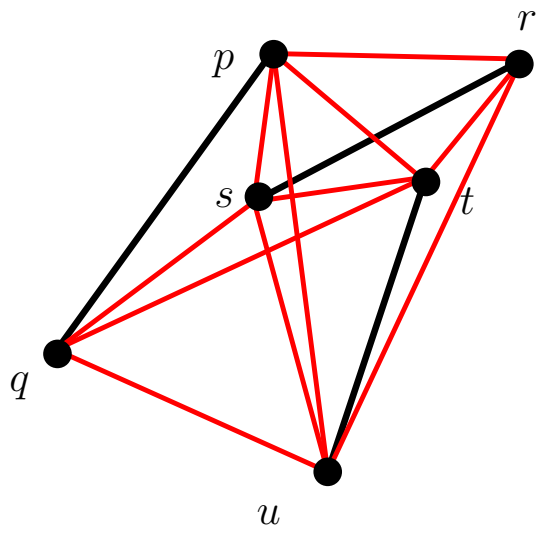
Volle Ordnung:

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

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:

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

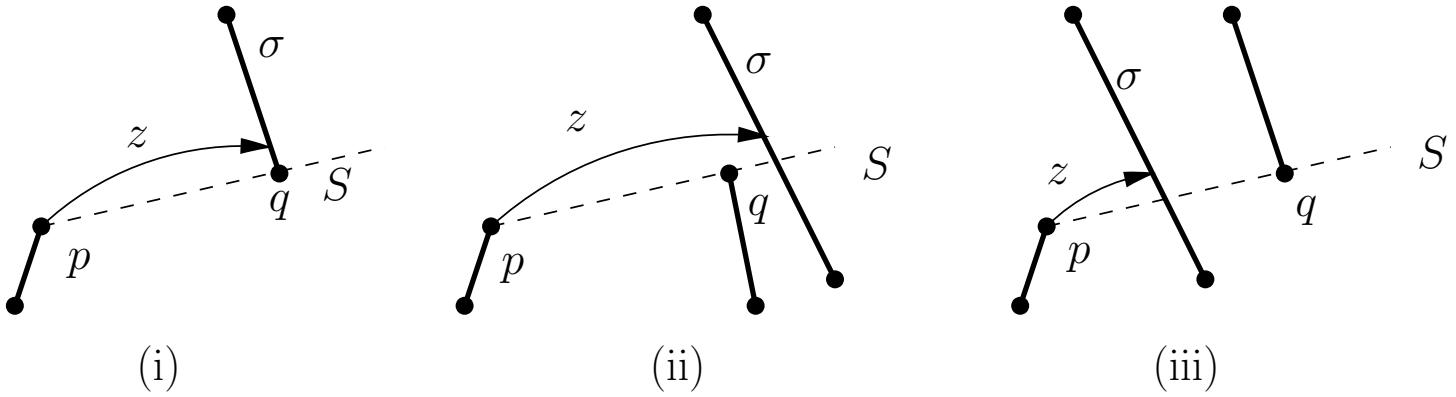
Sweep mit Bearbeitungsreihenfolge

Sweep mit Bearbeitungsreihenfolge

- ES gemäß Bearbeitungsreihenfolge: $\ldots, (p, q), \ldots, (p, r)$

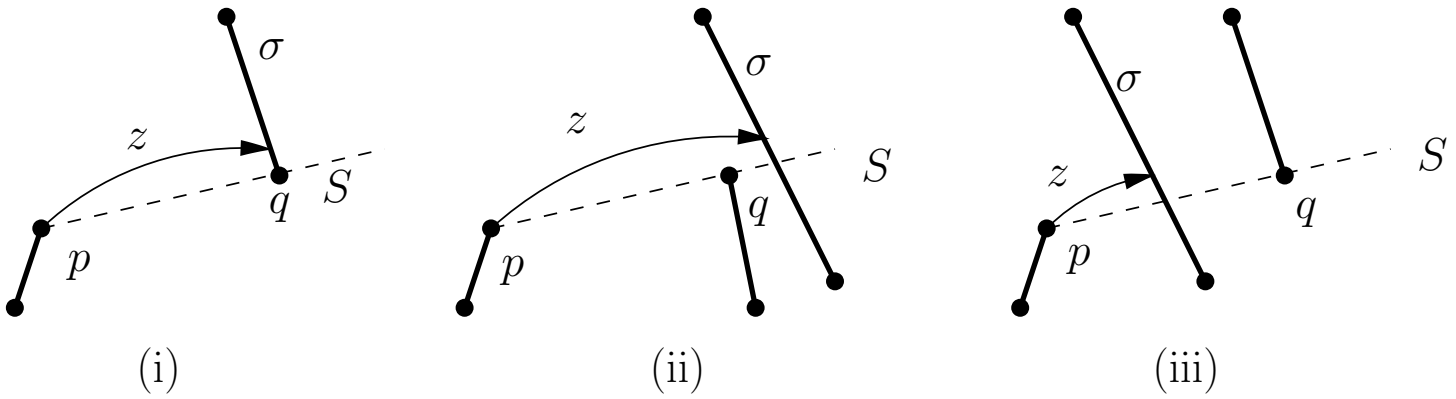
Sweep mit Bearbeitungsreihenfolge

- ES gemäß Bearbeitungsreihenfolge: $\ldots, (p,q), \ldots, (p,r)$
- SSS: Für jedes p, (p,q) zuletzt dran:



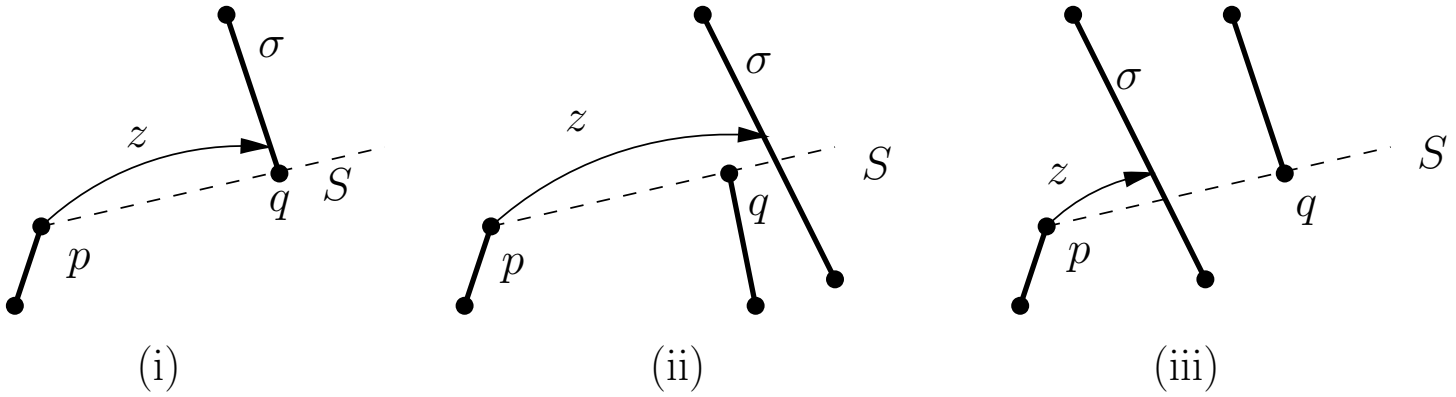
Sweep mit Bearbeitungsreihenfolge

- ES gemäß Bearbeitungsreihenfolge: $\ldots, (p,q), \ldots, (p,r)$
- SSS: Für jedes p, (p,q) zuletzt dran:
  Zeiger z auf das kurz hinter q *sichtbare* Segment σ



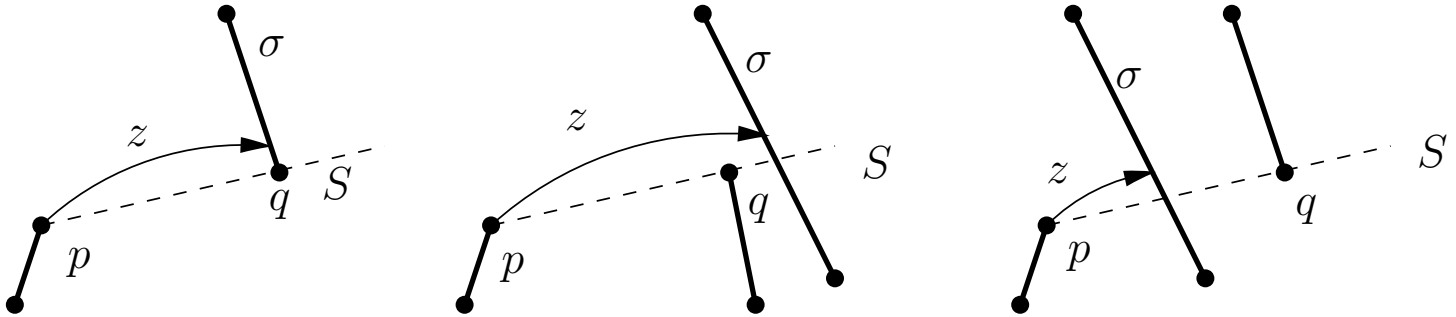
Sweep mit Bearbeitungsreihenfolge

- ES gemäß Bearbeitungsreihenfolge: $\ldots, (p,q), \ldots, (p,r)$
- SSS: Für jedes p, (p,q) zuletzt dran:
  Zeiger z auf das kurz hinter q *sichtbare* Segment σ
- Drei Möglichkeiten:

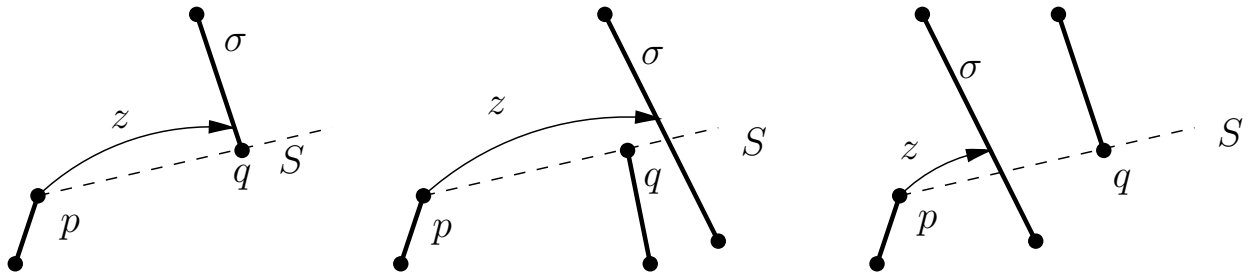


Sweep mit Bearbeitungsreihenfolge

- ES gemäß Bearbeitungsreihenfolge: $\ldots, (p,q), \ldots, (p,r)$
- SSS: Für jedes p, (p,q) zuletzt dran:
  Zeiger z auf das kurz hinter q *sichtbare* Segment σ
- Drei Möglichkeiten:
- Ausgabe zwischendurch

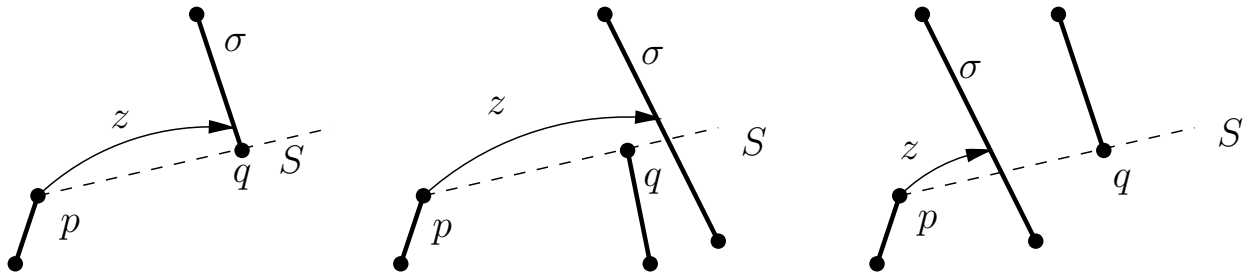


Sweep mit Bearbeitungsreihenfolge



Sweep mit Bearbeitungsreihenfolge

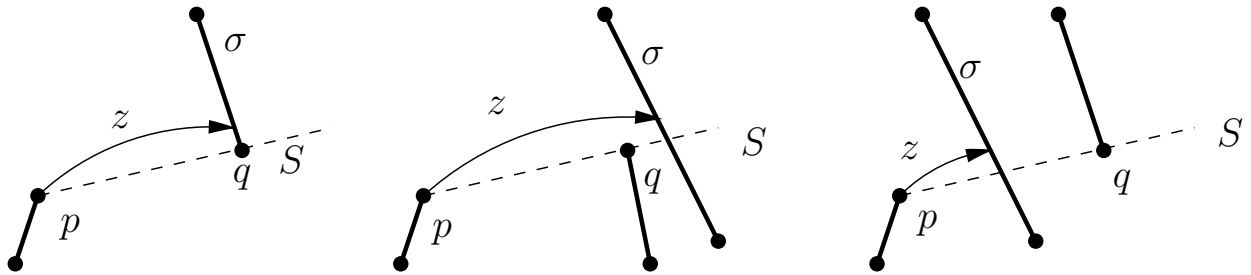
SSS: Für jedes p, (p, q) zuletzt dran:
Zeiger z auf kurz hinter q *sichtbare* Segment σ



Sweep mit Bearbeitungsreihenfolge

SSS: Für jedes p, (p, q) zuletzt dran:
Zeiger z auf kurz hinter q *sichtbare* Segment σ

i) q ist von p aus sichtbar und Anfangspunkt von σ

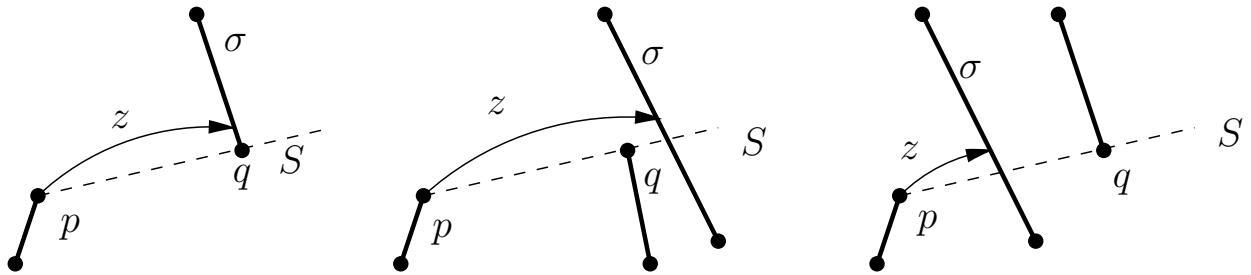


Sweep mit Bearbeitungsreihenfolge

SSS: Für jedes p, (p, q) zuletzt dran:
Zeiger z auf kurz hinter q *sichtbare* Segment σ

i) q ist von p aus sichtbar und Anfangspunkt von σ

ii) q ist von p aus sichtbar und Endpunkt eines Segments. σ ist das auf dem Strahl S von p durch q nächste Segment.



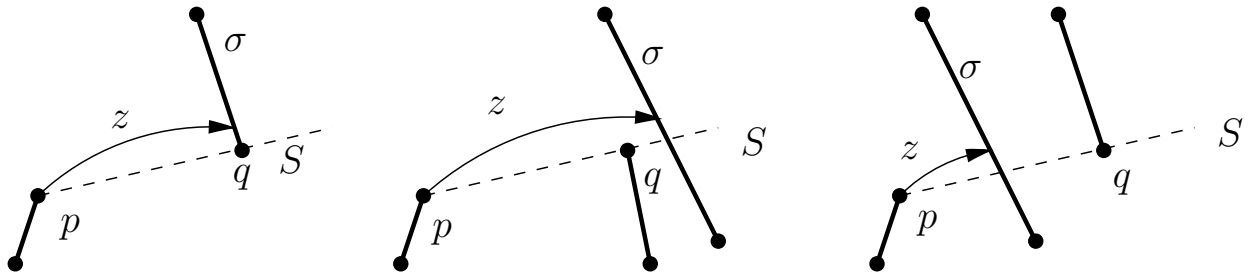
Sweep mit Bearbeitungsreihenfolge

SSS: Für jedes p, (p, q) zuletzt dran:
Zeiger z auf kurz hinter q *sichtbare* Segment σ

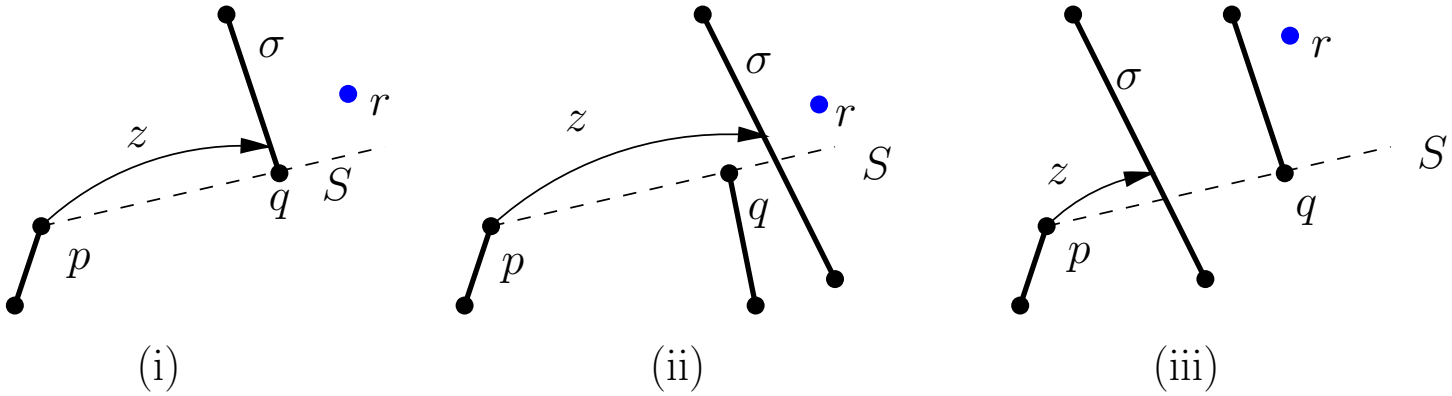
i) q ist von p aus sichtbar und Anfangspunkt von σ

ii) q ist von p aus sichtbar und Endpunkt eines Segments. σ ist das auf dem Strahl S von p durch q nächste Segment.

ii) q ist von p aus nicht sichtbar. σ ist das zu p nächste Segment auf dem Strahl S von p durch q.



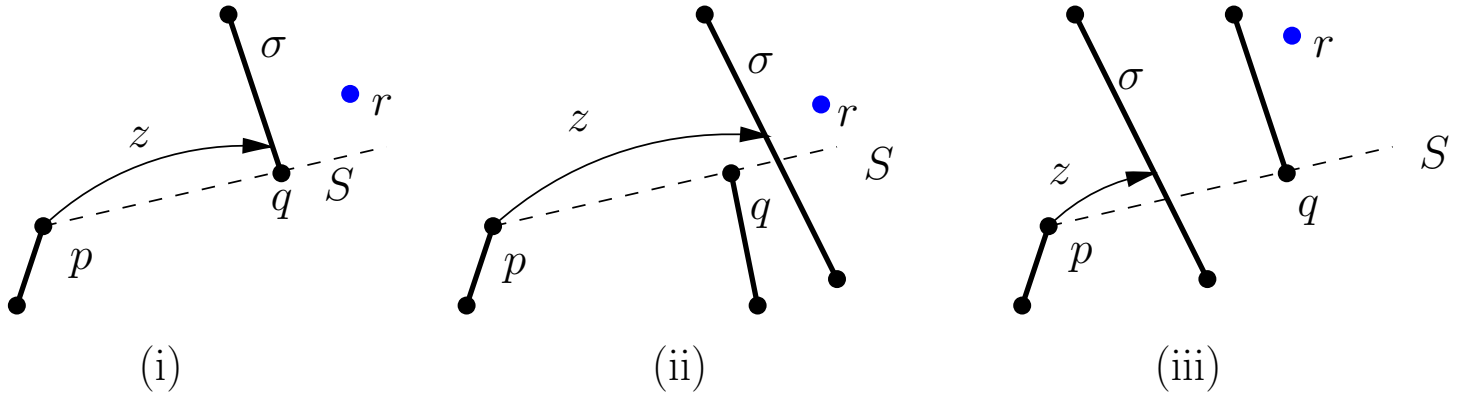
Sweep mit Bearbeitungsreihenfolge



Sweep mit Bearbeitungsreihenfolge

Bearbeitung des nächsten Paares (p, r) nach (p, q)

F. 1) r liegt hinter σ

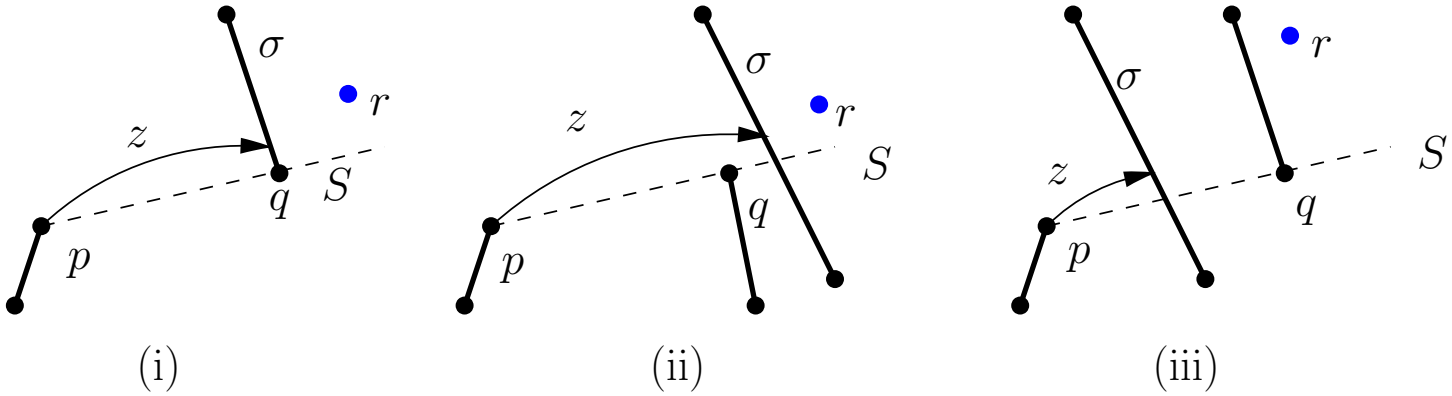


Sweep mit Bearbeitungsreihenfolge

Bearbeitung des nächsten Paares (p, r) nach (p, q)

F. 1) r liegt hinter σ

- Zeiger bleibt auf σ

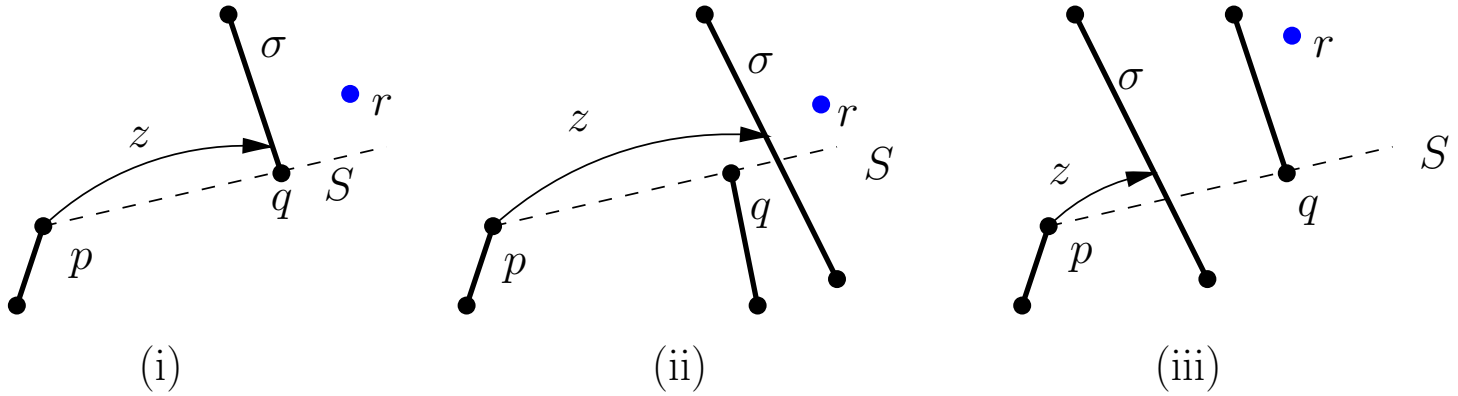


Sweep mit Bearbeitungsreihenfolge

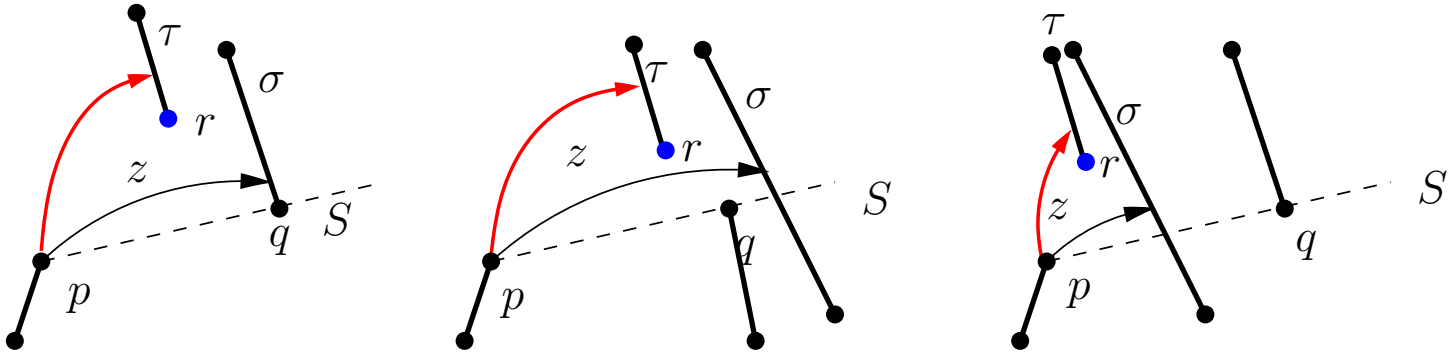
Bearbeitung des nächsten Paares (p, r) nach (p, q)

F. 1) r liegt hinter σ

- Zeiger bleibt auf σ
- Invariante gilt nun für (p, r)



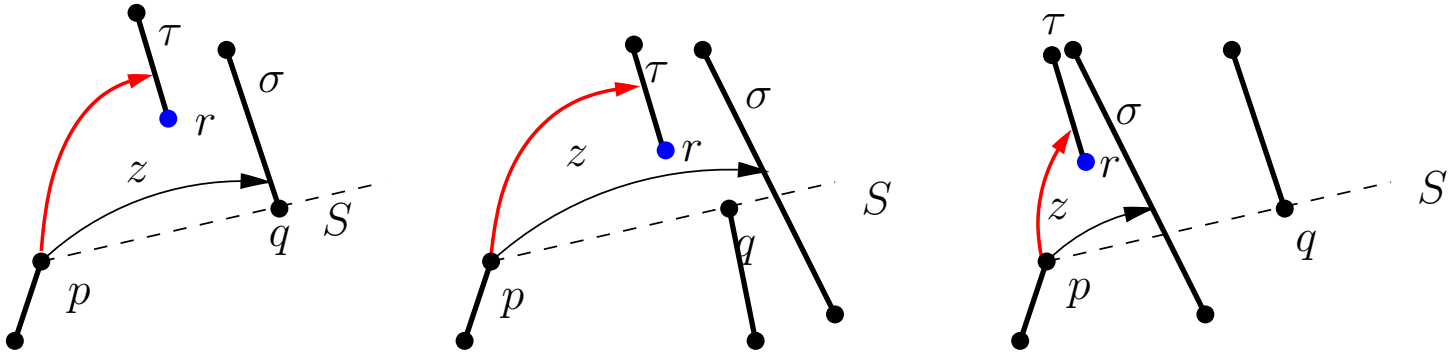
Sweep mit Bearbeitungsreihenfolge



Sweep mit Bearbeitungsreihenfolge

Bearbeitung des nächsten Paares (p, r) nach (p, q)

F. 2) r liegt vor σ, Anfangspunkt von τ

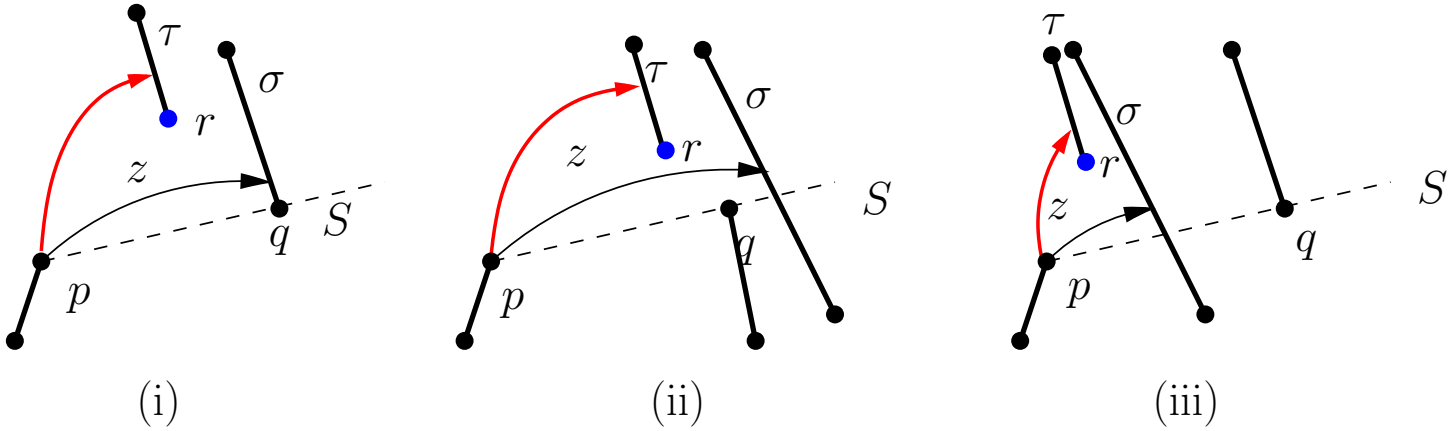


Sweep mit Bearbeitungsreihenfolge

Bearbeitung des nächsten Paares (p, r) nach (p, q)

F. 2) r liegt vor σ, Anfangspunkt von τ

- Zeiger auf τ setzen

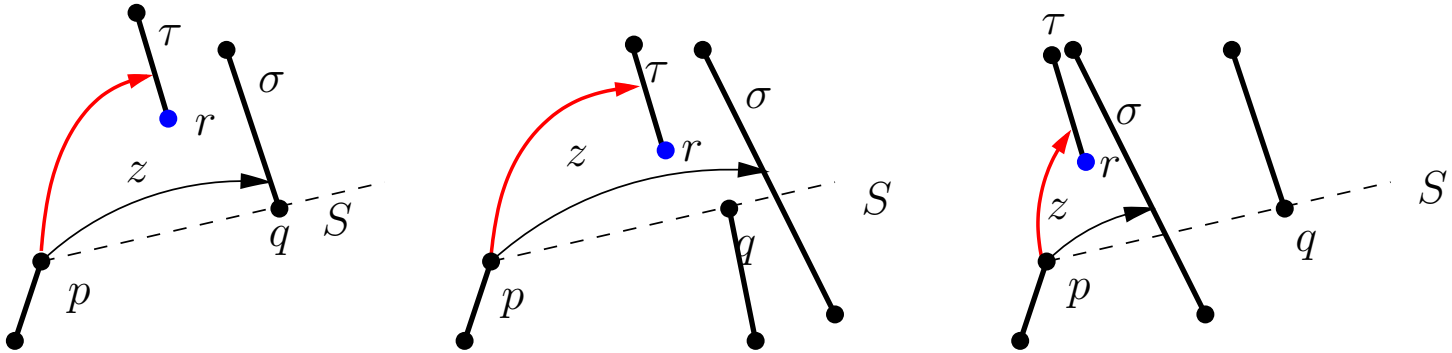


Sweep mit Bearbeitungsreihenfolge

Bearbeitung des nächsten Paares (p, r) nach (p, q)

F. 2) r liegt vor σ, Anfangspunkt von τ

- Zeiger auf τ setzen
- Ausgabe: (p, r)!

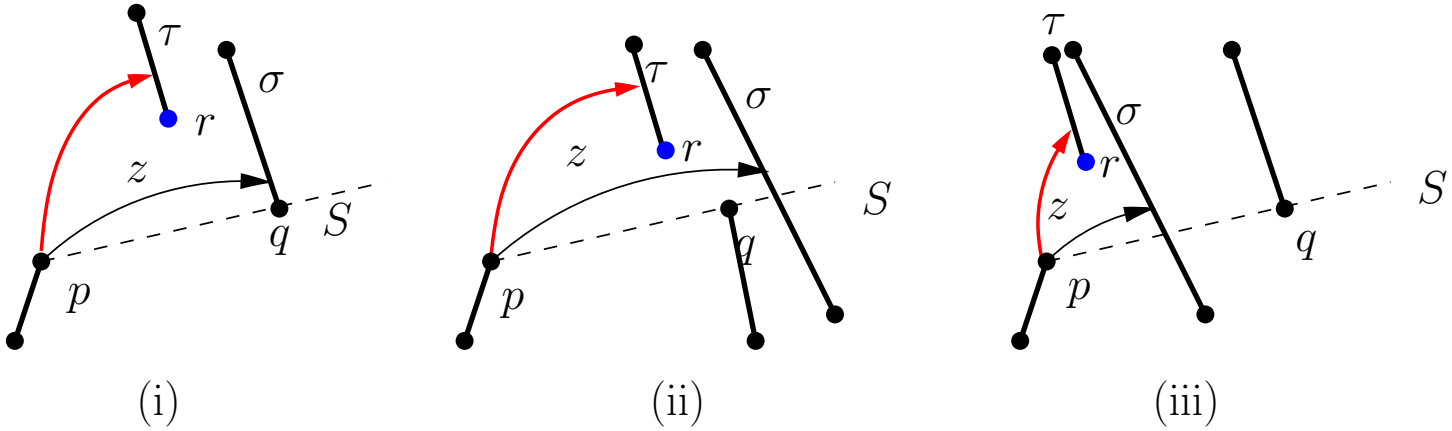


Sweep mit Bearbeitungsreihenfolge

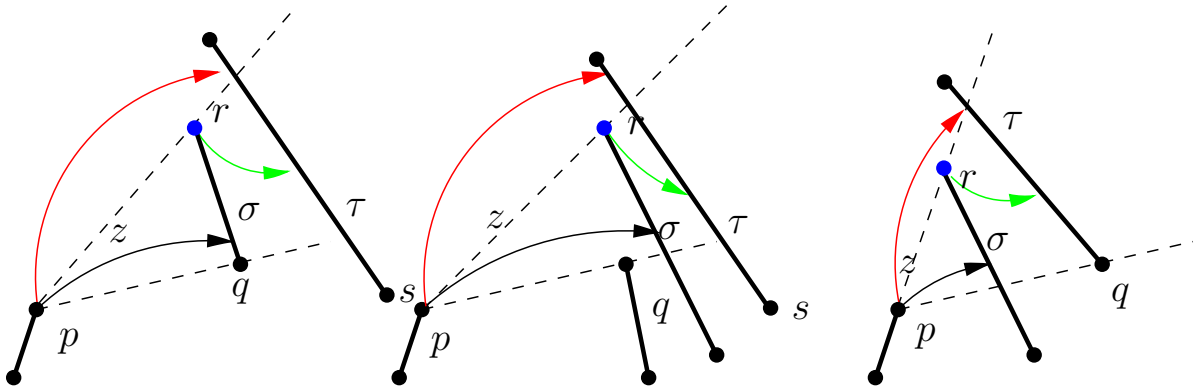
Bearbeitung des nächsten Paares (p, r) nach (p, q)

F. 2) r liegt vor σ, Anfangspunkt von τ

- Zeiger auf τ setzen
- Ausgabe: (p, r)! Invariante gilt nun für (p, r)



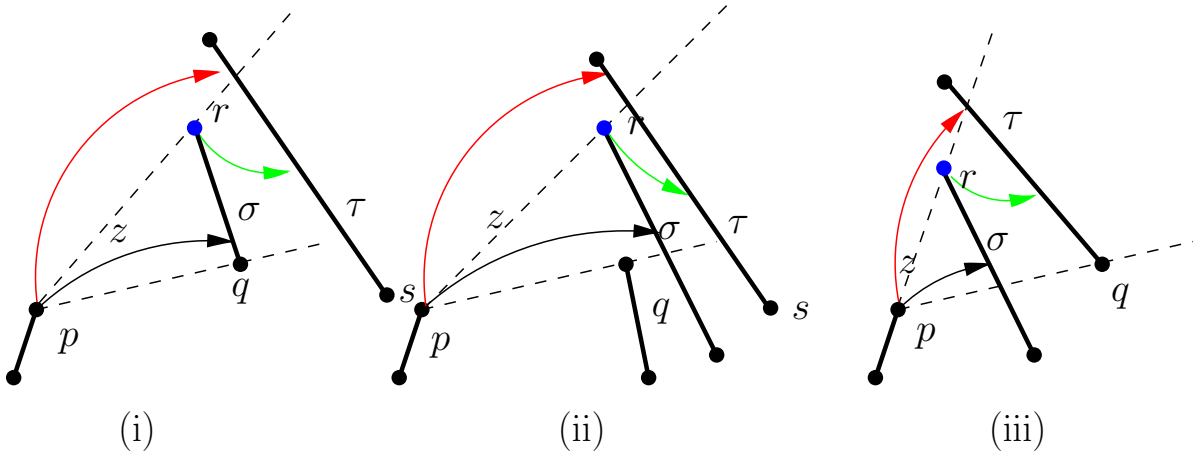
Sweep mit Bearbeitungsreihenfolge



Sweep mit Bearbeitungsreihenfolge

Bearbeitung des nächsten Paares (p, r) nach (p, q)

F. 3) r ist Endpunkt von σ

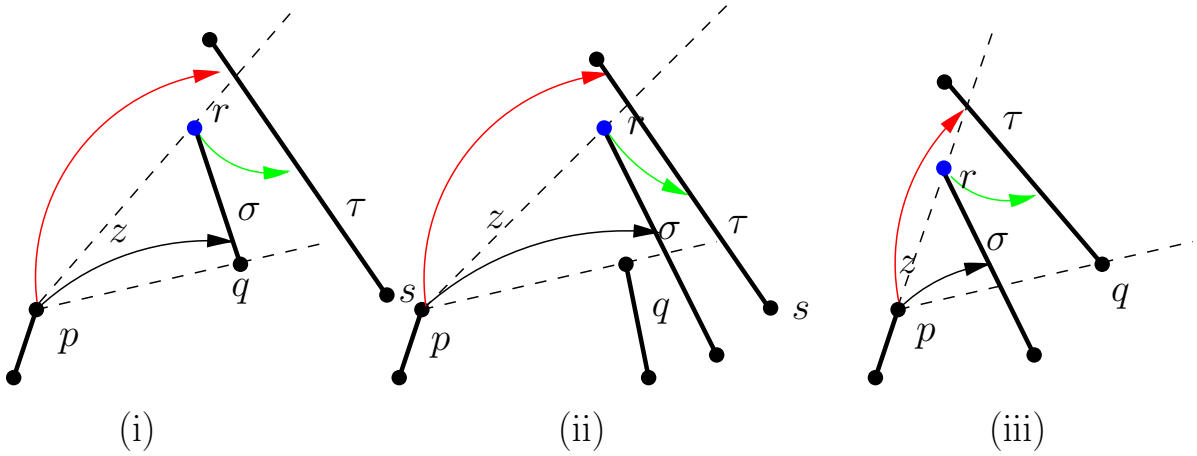


Sweep mit Bearbeitungsreihenfolge

Bearbeitung des nächsten Paares (p, r) nach (p, q)

F. 3) r ist Endpunkt von σ

- Zeiger auf τ von r: (r, s) war schon dran Reihenfolge

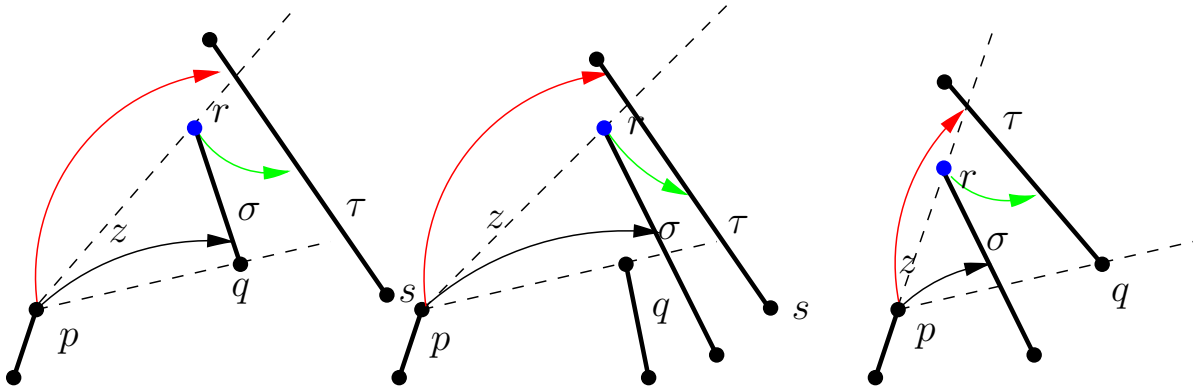


Sweep mit Bearbeitungsreihenfolge

Bearbeitung des nächsten Paares (p, r) nach (p, q)

F. 3) r ist Endpunkt von σ

- Zeiger auf τ von r: (r, s) war schon dran Reihenfolge
- Winkelbereich ist leer!!

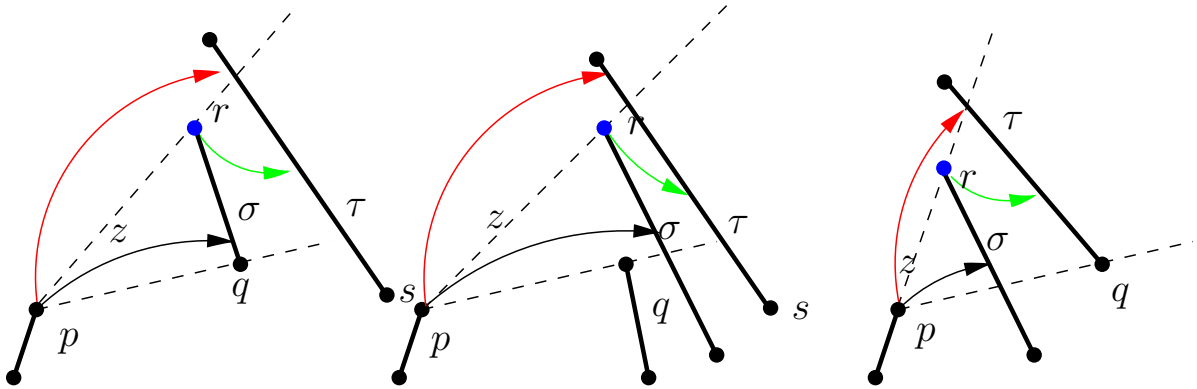


Sweep mit Bearbeitungsreihenfolge

Bearbeitung des nächsten Paares (p, r) nach (p, q)

F. 3) r ist Endpunkt von σ

- Zeiger auf τ von r: (r, s) war schon dran Reihenfolge
- Winkelbereich ist leer!!
- Ausgabe: (p, r)!

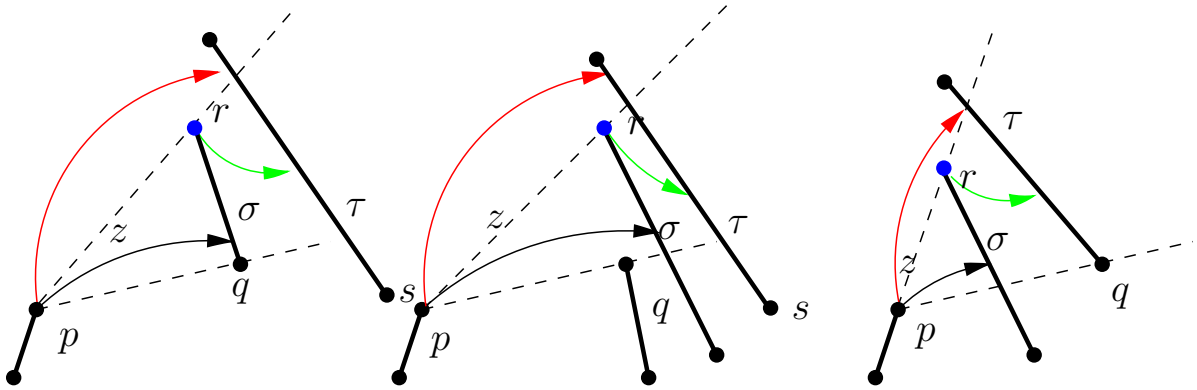


Sweep mit Bearbeitungsreihenfolge

Bearbeitung des nächsten Paares (p, r) nach (p, q)

F. 3) r ist Endpunkt von σ

- Zeiger auf τ von r: (r, s) war schon dran Reihenfolge
- Winkelbereich ist leer!!
- Ausgabe: (p, r)! Invariante gilt nun für (p, r)



Simultaner Sweep: Ereignisverarbeitung Fall 3(iii)

Simultaner Sweep: Ereignisverarbeitung Fall 3(iii)

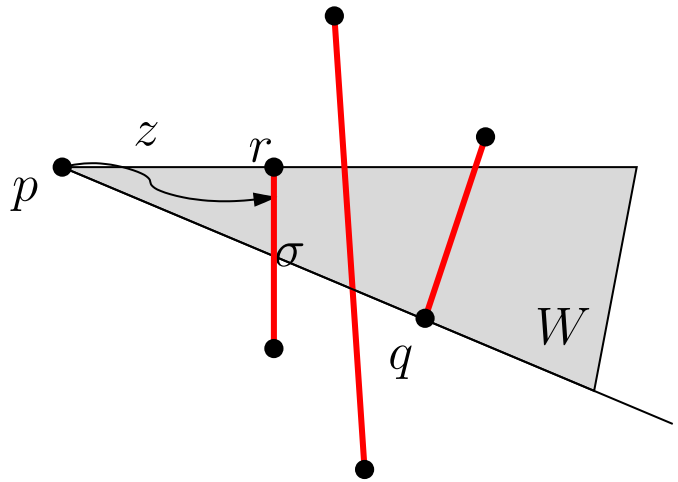
- r Endpunkt von S

Simultaner Sweep: Ereignisverarbeitung Fall 3(iii)

- r Endpunkt von S
- Winkelbereich W: Punktfrei

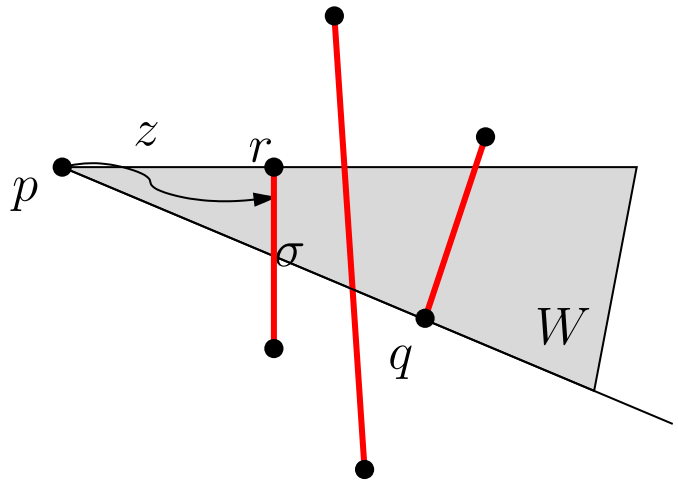
Simultaner Sweep: Ereignisverarbeitung Fall 3(iii)

- r Endpunkt von S
- Winkelbereich W: Punktfrei
- (r, q) war dran:



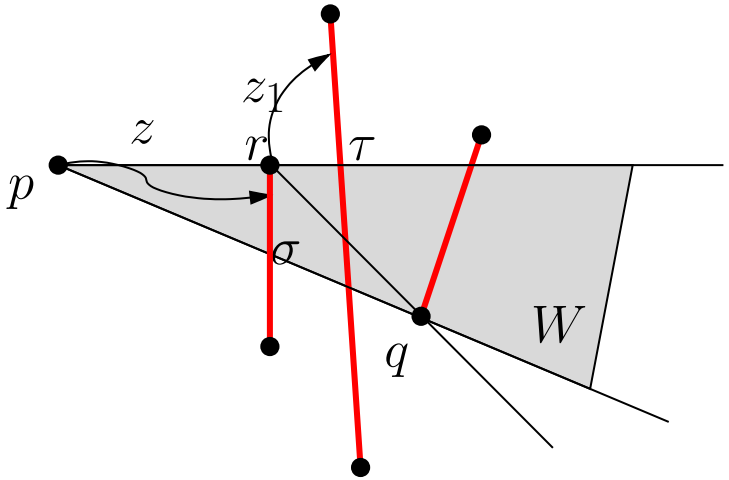
Simultaner Sweep: Ereignisverarbeitung Fall 3(iii)

- r Endpunkt von S
- Winkelbereich W: Punktfrei
- (r, q) war dran: Umbiegen d. Zeigers/Ausgabe,



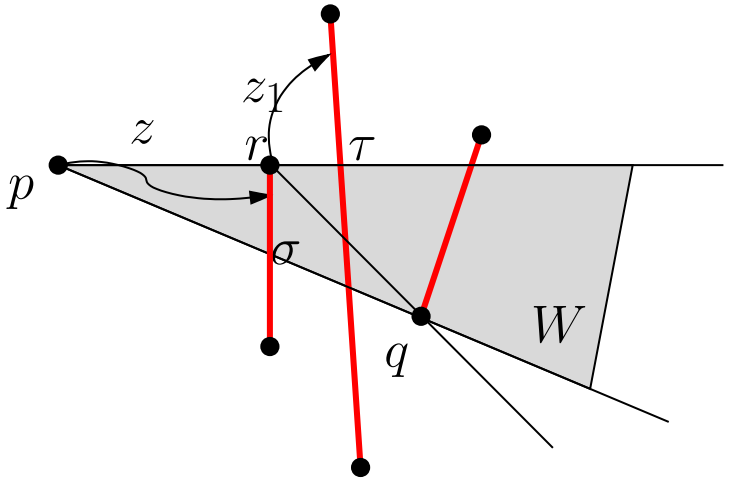
Simultaner Sweep: Ereignisverarbeitung Fall 3(iii)

- r Endpunkt von S
- Winkelbereich W: Punktfrei
- (r, q) war dran: Umbiegen d. Zeigers/Ausgabe, $O(1)$

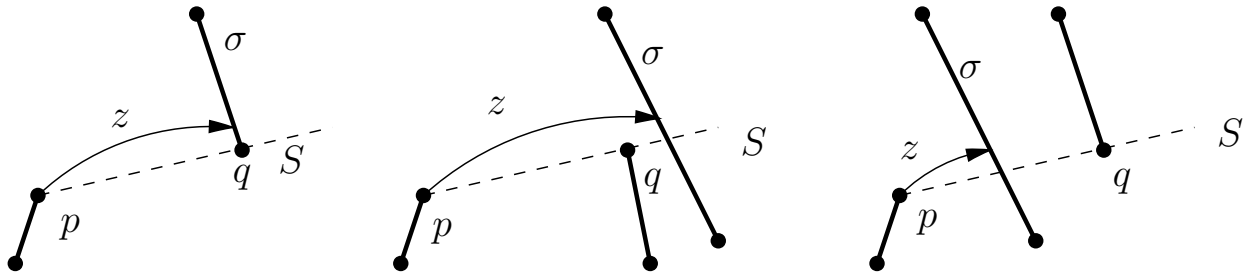


Simultaner Sweep: Ereignisverarbeitung Fall 3(iii)

- r Endpunkt von S
- Winkelbereich W: Punktfrei
- (r, q) war dran: Umbiegen d. Zeigers/Ausgabe, $O(1)$

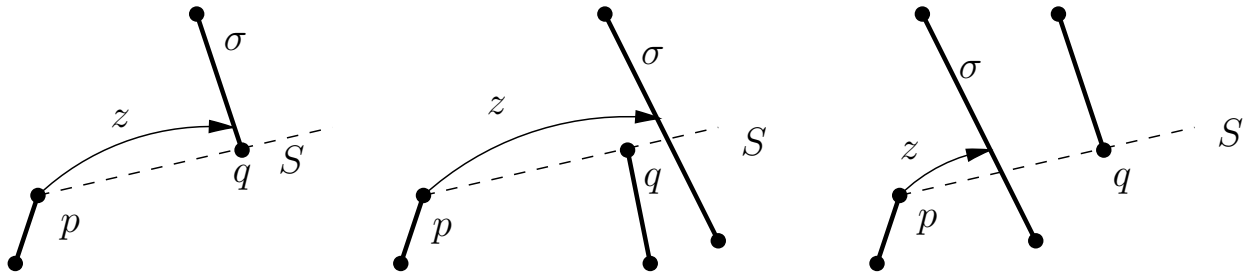


Simultaner Sweep: Laufzeit und Korrektheit



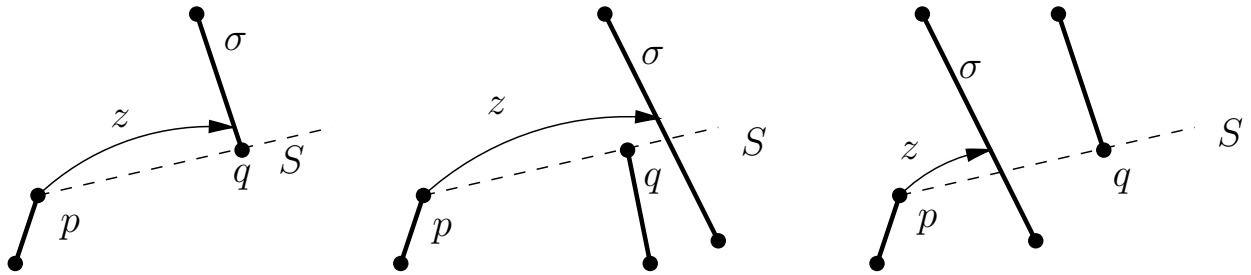
Simultaner Sweep: Laufzeit und Korrektheit

- Invariante ist stets erfüllt! Zeiger auf σ korrekt



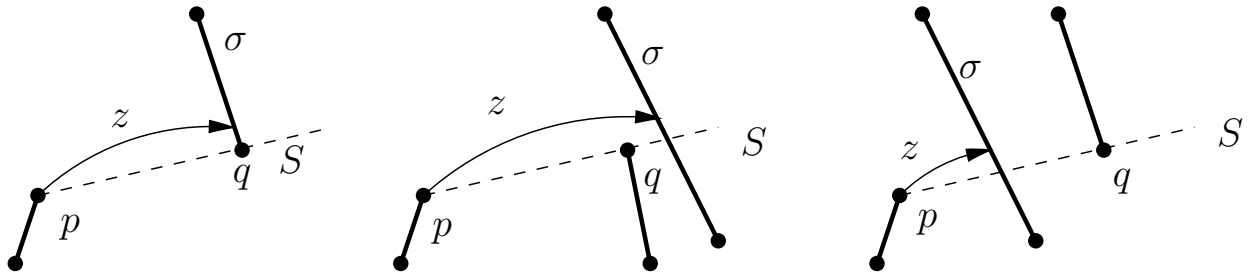
Simultaner Sweep: Laufzeit und Korrektheit

- Invariante ist stets erfüllt! Zeiger auf σ korrekt
- Bearbeitungsreihenfolge: Alle Paare (p, r) werden betrachtet!



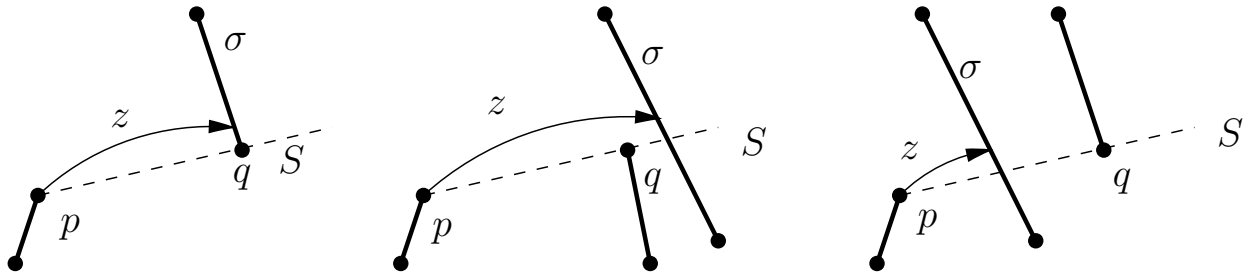
Simultaner Sweep: Laufzeit und Korrektheit

- Invariante ist stets erfüllt! Zeiger auf σ korrekt
- Bearbeitungsreihenfolge: Alle Paare (p, r) werden betrachtet!
- Immer wenn r auf σ liegt erfolgt Ausgabe



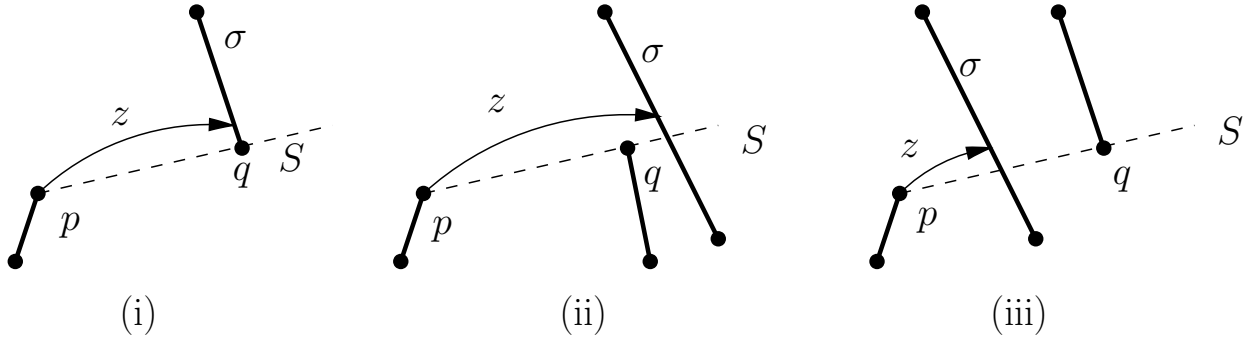
Simultaner Sweep: Laufzeit und Korrektheit

- Invariante ist stets erfüllt! Zeiger auf σ korrekt
- Bearbeitungsreihenfolge: Alle Paare (p, r) werden betrachtet!
- Immer wenn r auf σ liegt erfolgt Ausgabe
- Bearbeitungsreihenfolge: $O(n^2)$ Ereignisse, jeweils $O(1)$ Aufwand



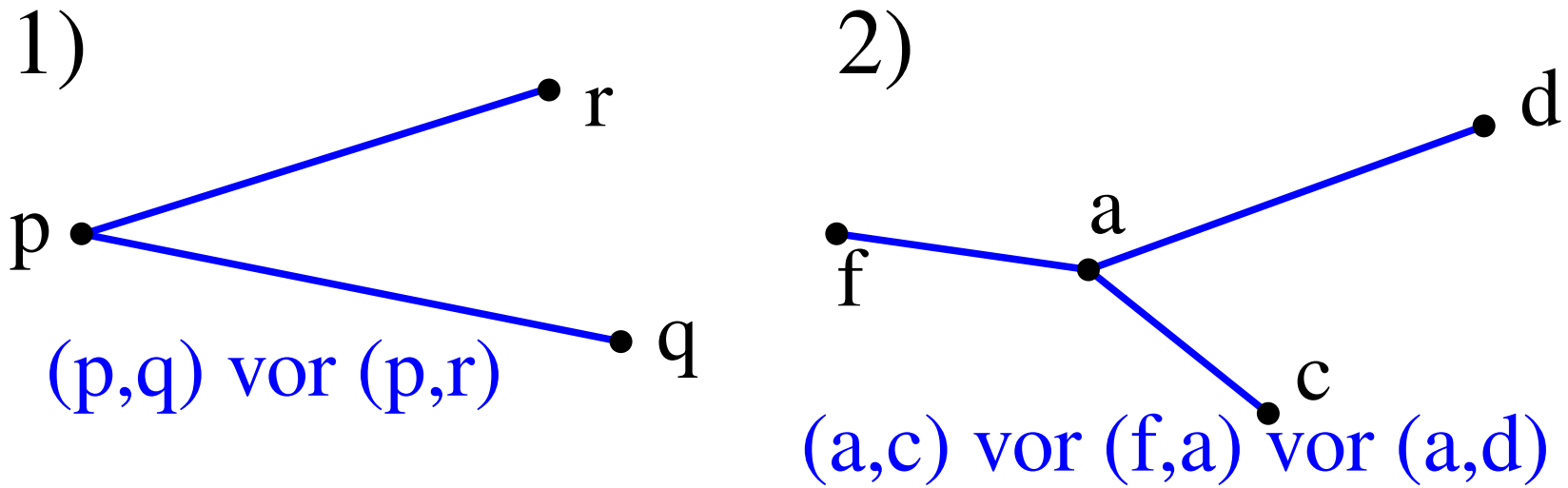
Simultaner Sweep: Laufzeit und Korrektheit

- Invariante ist stets erfüllt! Zeiger auf σ korrekt
- Bearbeitungsreihenfolge: Alle Paare (p, r) werden betrachtet!
- Immer wenn r auf σ liegt erfolgt Ausgabe
- Bearbeitungsreihenfolge: $O(n^2)$ Ereignisse, jeweils $O(1)$ Aufwand
- Korrekter Sweep in $O(n^2)$

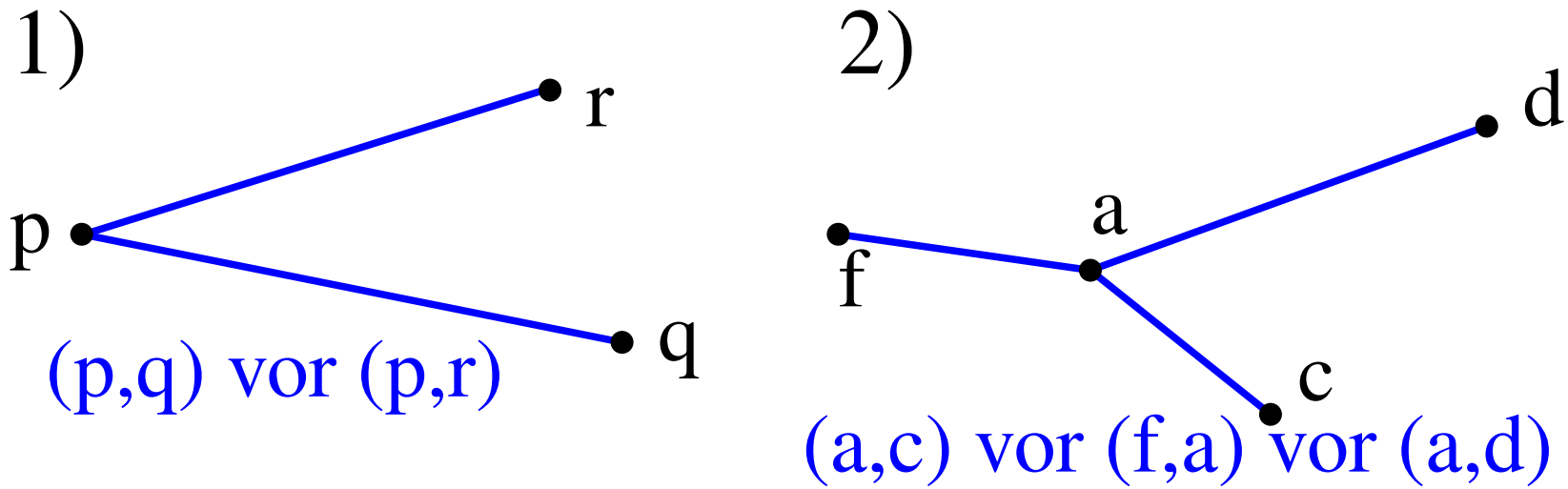


Bearbeitungsreihenfolge in $O(n^2)$

Bearbeitungsreihenfolge in $O(n^2)$

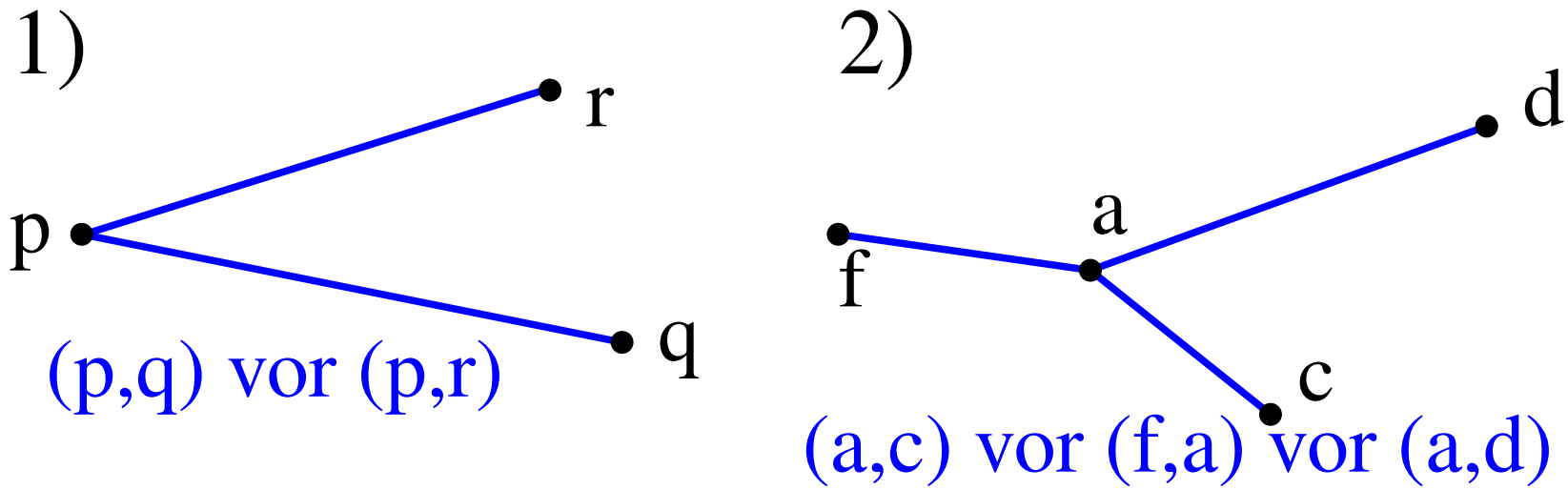


Bearbeitungsreihenfolge in $O(n^2)$



Offensichtlich wird nicht mehr gebraucht!

Bearbeitungsreihenfolge in $O(n^2)$



Offensichtlich wird nicht mehr gebraucht!

Diese muss nun berechnet werden!