

# 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: Dienstags 12:00-14.00 Uhr, A7a  
Donnerstags 16:00-18.00, A301



# Organisatorisches

- Übungen: Dienstags 12:00-14.00 Uhr, A7a  
Donnerstags 16:00-18.00, A301
- Wöchentlich, Beginn: 1.11

# 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)

# 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

# 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

# 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)

# 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](http://www.geometrylab.de))

# 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](http://www.geometrylab.de))
- Folien/Skript online

# 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](http://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 Adaptionfähigkeiten, Entwicklung eigener Lösungsansätze

# Einordnung: Bewegungsplanung

# Einordnung: Bewegungsplanung

- Maschinen/Agenten

# Einordnung: Bewegungsplanung

- Maschinen/Agenten
- Elektronik

# Einordnung: Bewegungsplanung

- Maschinen/Agenten
- Elektronik
- Mechanik

# Einordnung: Bewegungsplanung

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



# Einordnung: Bewegungsplanung

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

# Einordnung: Bewegungsplanung

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

# Einordnung: Bewegungsplanung

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

# Einordnung: Bewegungsplanung

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

# Einordnung: Bewegungsplanung

- Maschinen/Agenten
- Elektronik
- Mechanik
- Kontroll- und Regelungstechnik
- KI
- Softwareengineering
- ⋮
- 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)

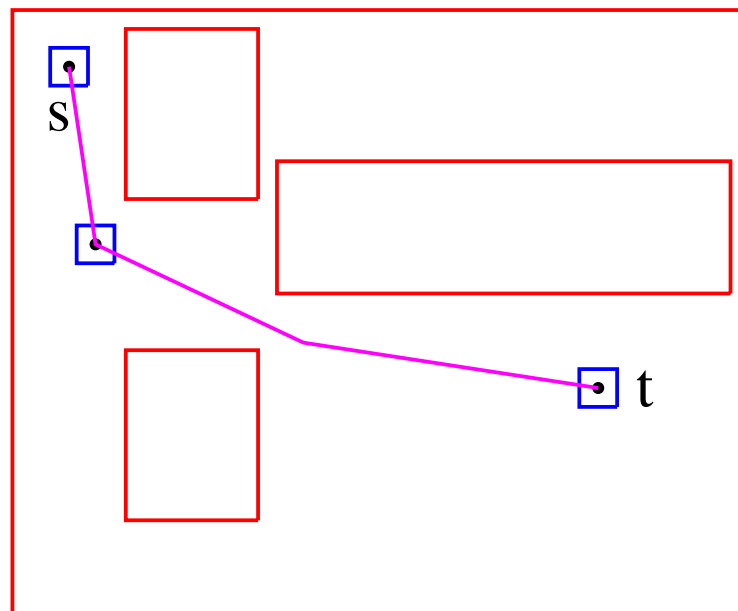
# Geometrische Algorithmen

- 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

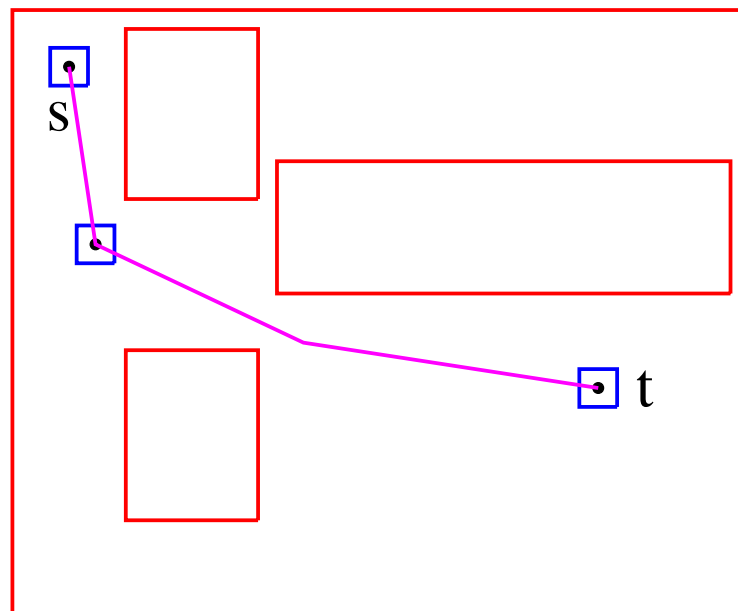
# Bsp: Translation Quadrat unter Rechtecken





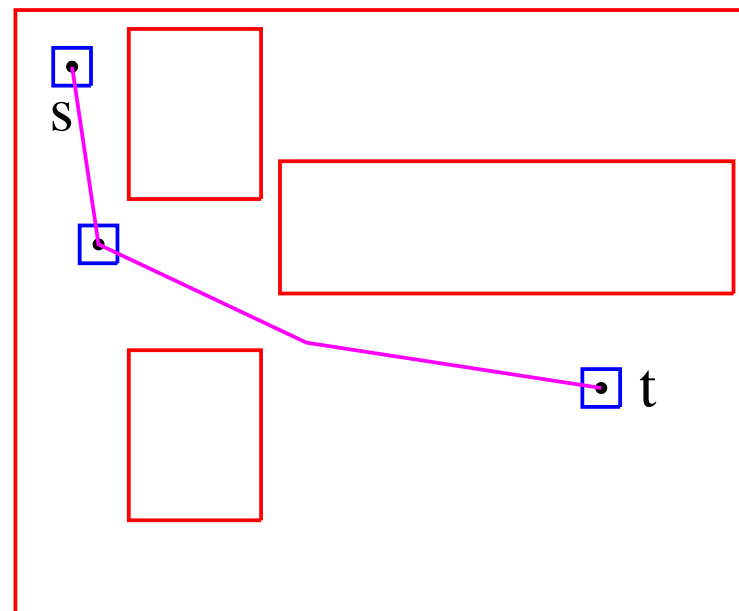
# Bsp: Translation Quadrat unter Rechtecken

- Achsenparalleles Rechteck von  $s$  nach  $t$  bewegen



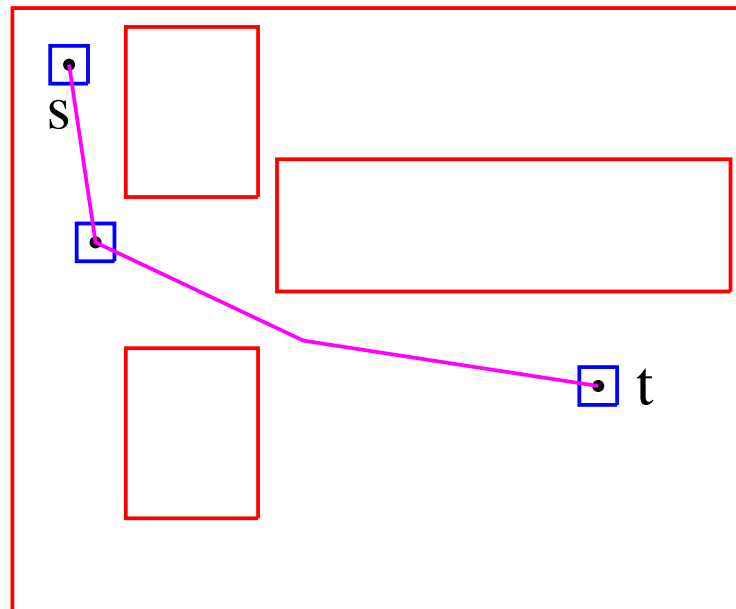
# Bsp: Translation Quadrat unter Rechtecken

- Achsenparalleles Rechteck von  $s$  nach  $t$  bewegen
- Achsenparallele Hindernisse



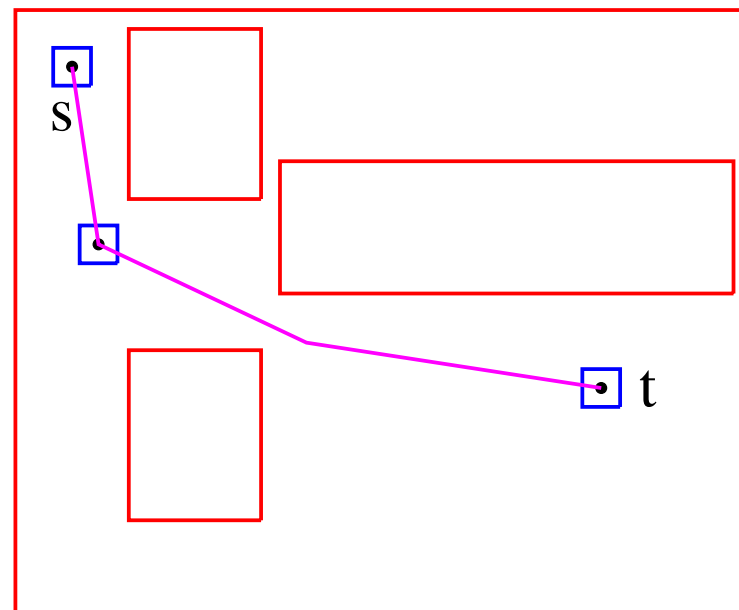
# Bsp: Translation Quadrat unter Rechtecken

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



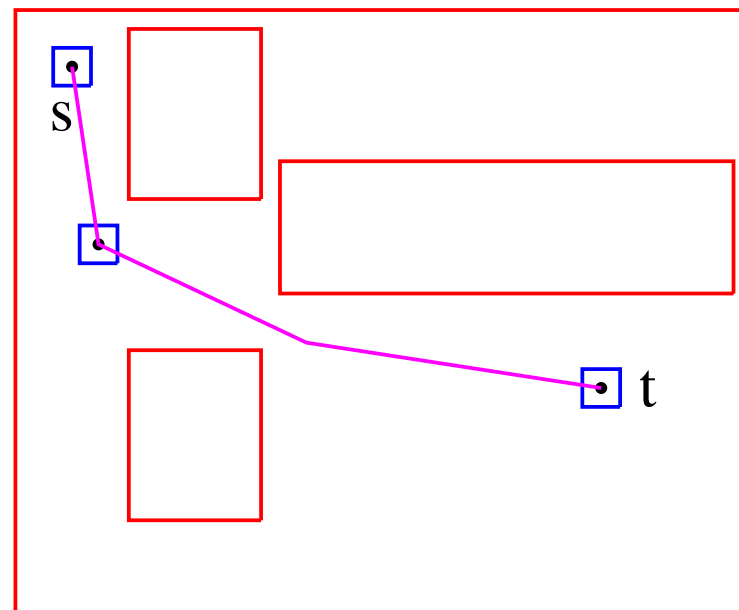
# Bsp: Translation Quadrat unter Rechtecken

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



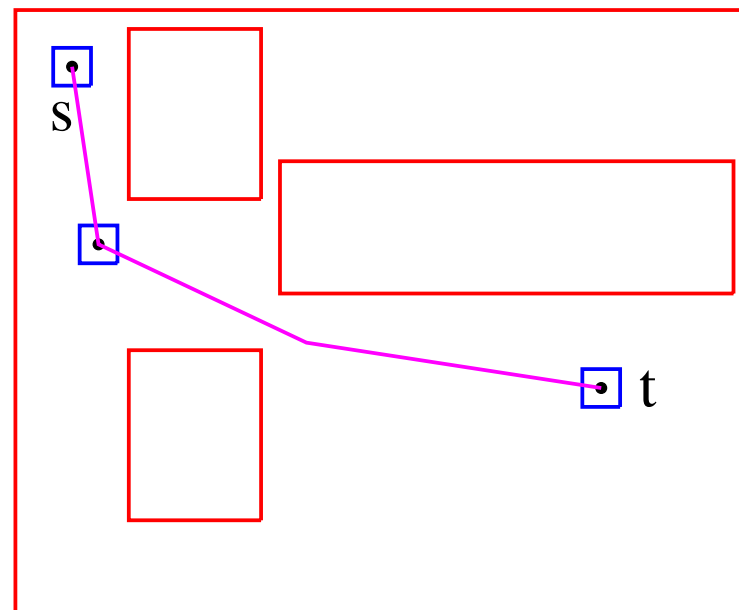
# Bsp: Translation Quadrat unter Rechtecken

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



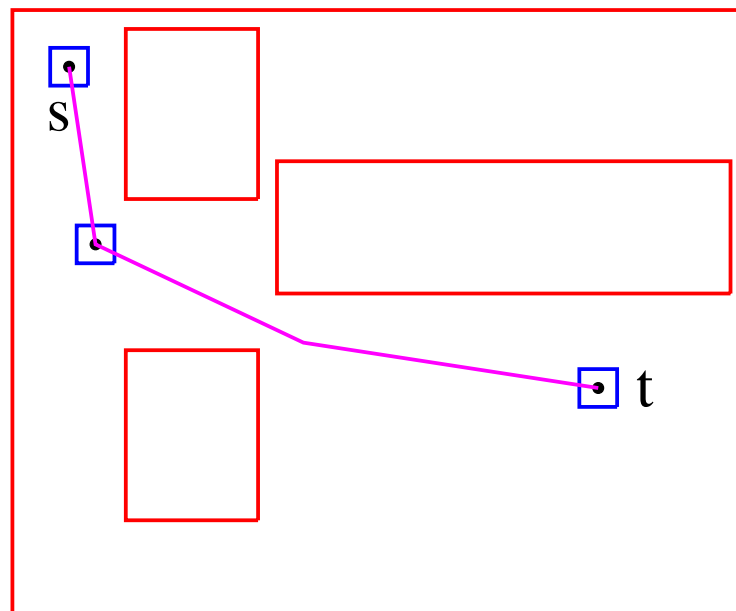
# Bsp: Translation Quadrat unter Rechtecken

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



# Bsp: Translation Quadrat unter Rechtecken

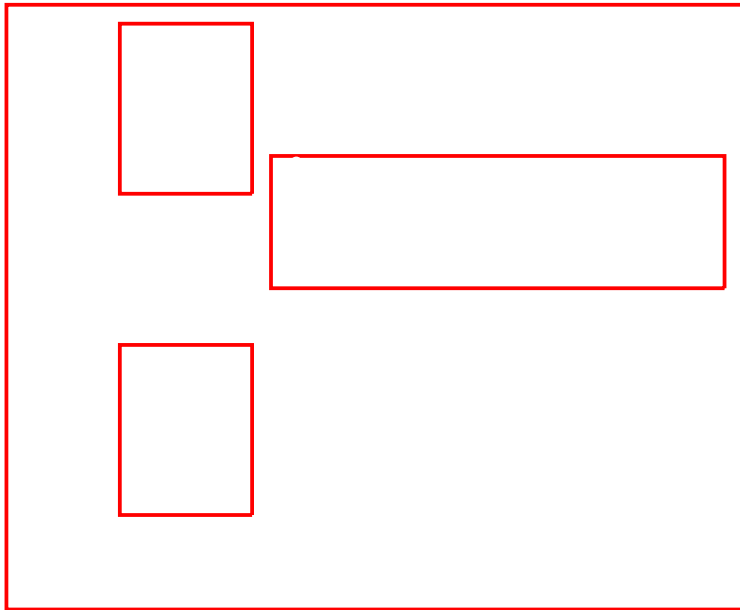
- Achsenparalleles Rechteck von  $s$  nach  $t$  bewegen
- Achsenparallele Hindernisse
- Fragen: Geht das? Geringe Kosten? Berechnung?
- Idee: Referenzpunkt kollisionsfrei setzen



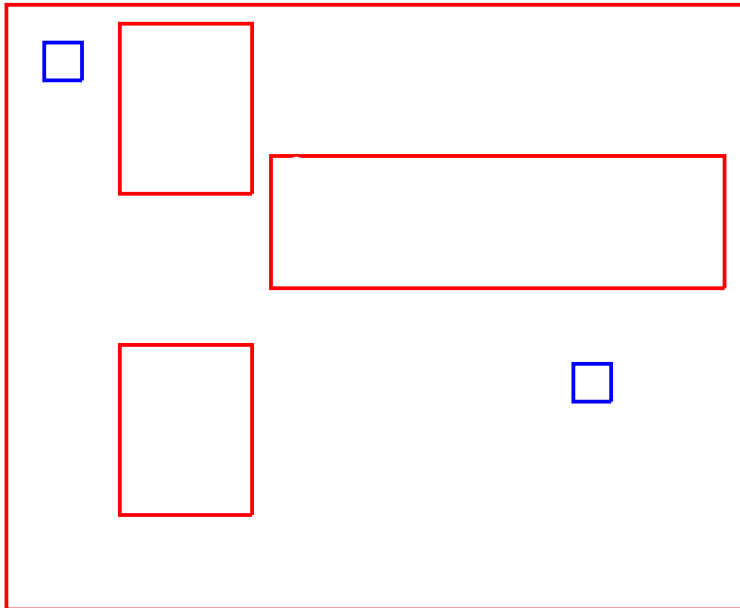
# Idee: Translation Quadrat unter Rechtecken



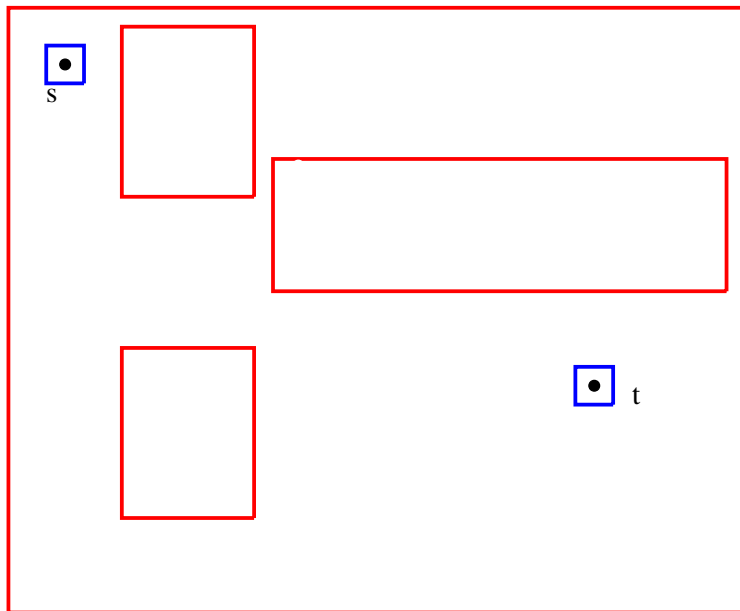
# Idee: Translation Quadrat unter Rechtecken



# Idee: Translation Quadrat unter Rechtecken

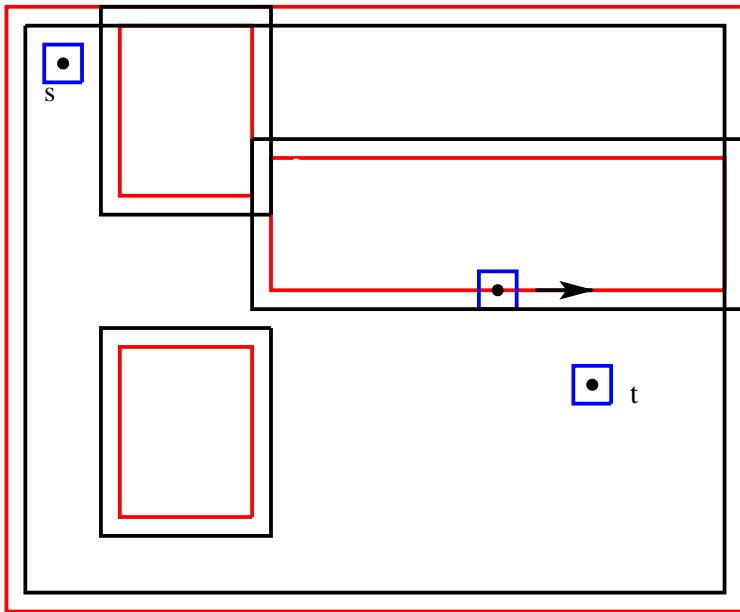


# Idee: Translation Quadrat unter Rechtecken



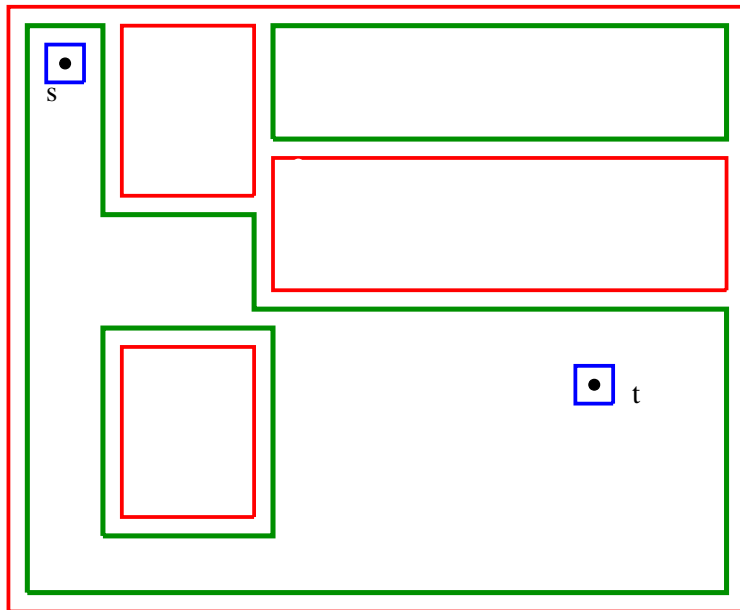
Translation: Verschieben Referenzpunkt

# Idee: Translation Quadrat unter Rechtecken



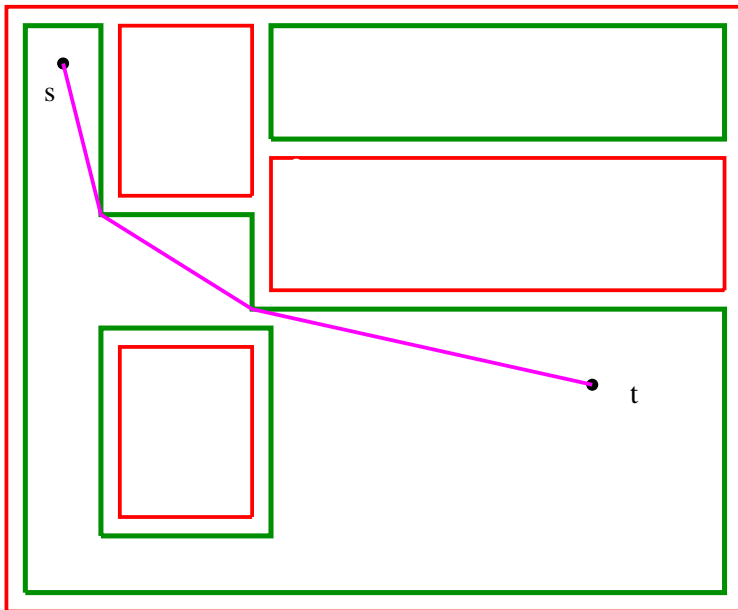
Translation: Verschieben Referenzpunkt  
Aufblasen der Hindernisse

# Idee: Translation Quadrat unter Rechtecken



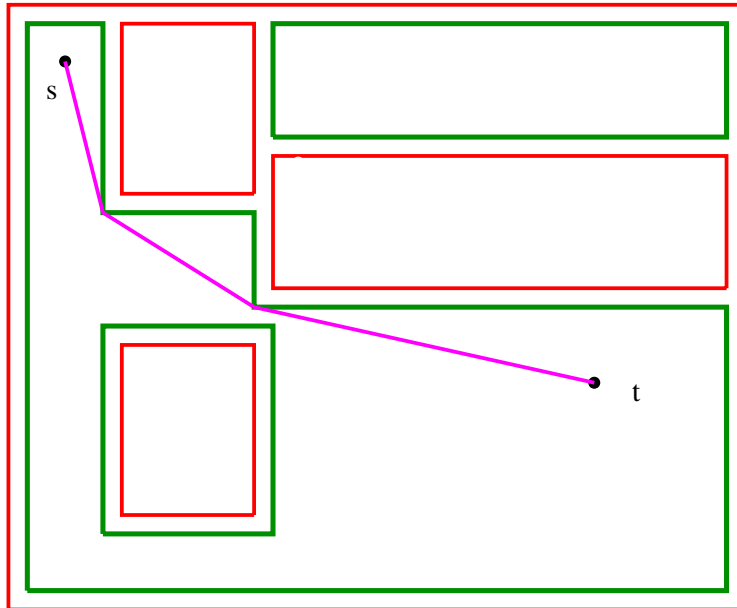
Translation: Verschieben Referenzpunkt  
Aufblasen der Hindernisse  
Vereinigung Sweep

# 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

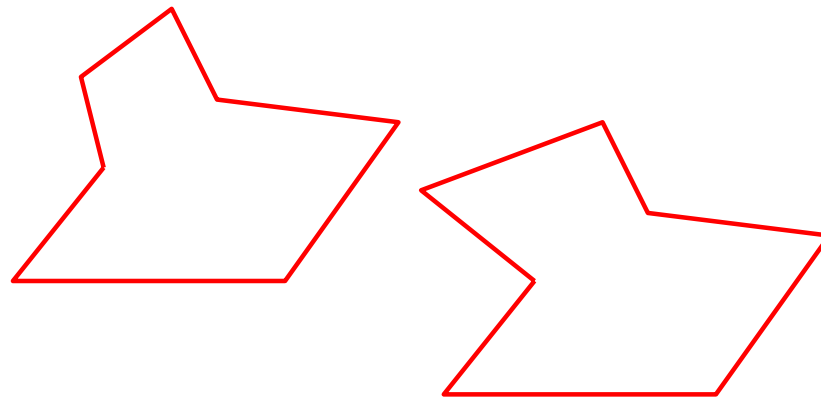


# Kap 1.1: Kürzeste Wege, polygonale Hindernisse

- *Polygonale Hindernisse*: Definition

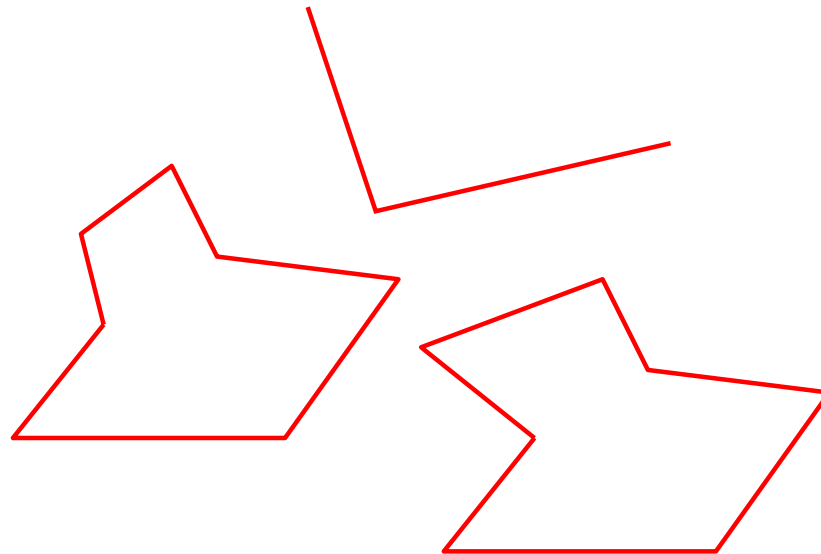
# Kap 1.1: Kürzeste Wege, polygonale Hindernisse

- *Polygonale Hindernisse*: Definition
- Geschlossene Kantenzüge,



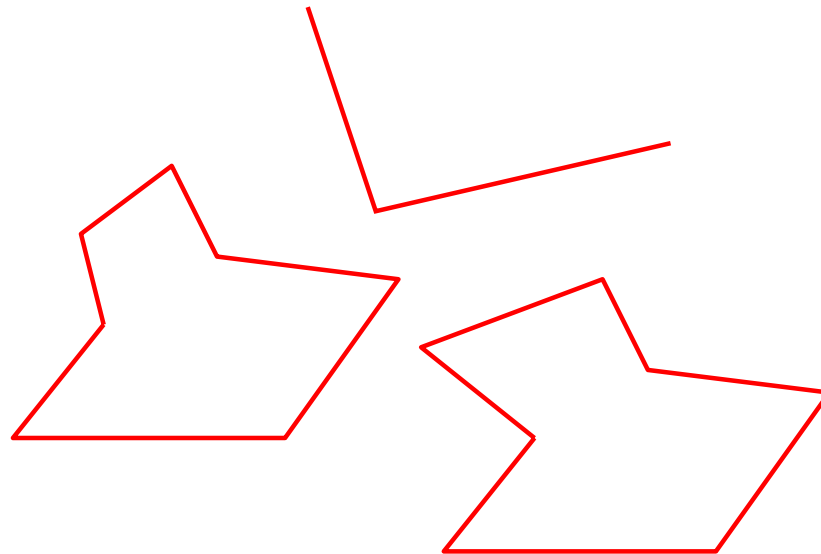
# Kap 1.1: Kürzeste Wege, polygonale Hindernisse

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



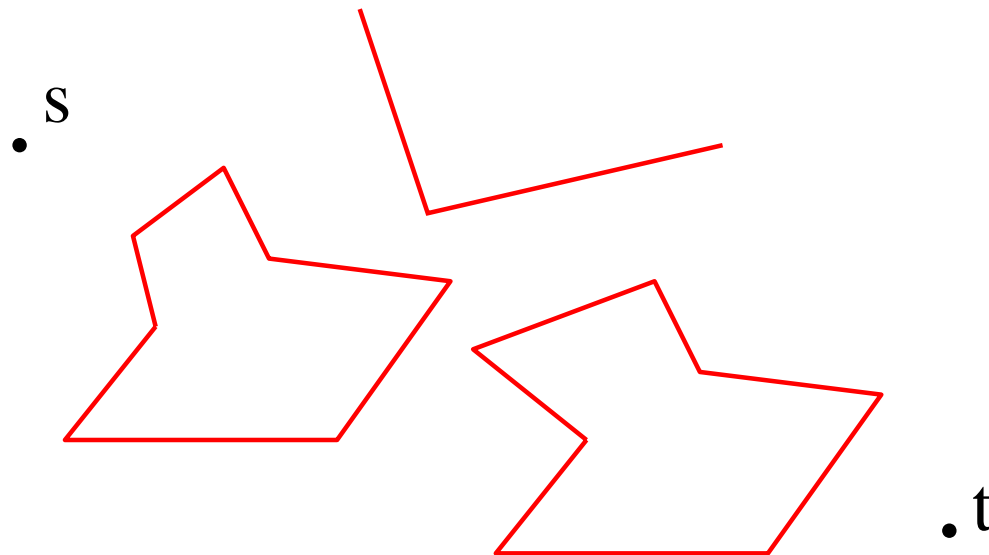
# Kap 1.1: Kürzeste Wege, polygonale Hindernisse

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



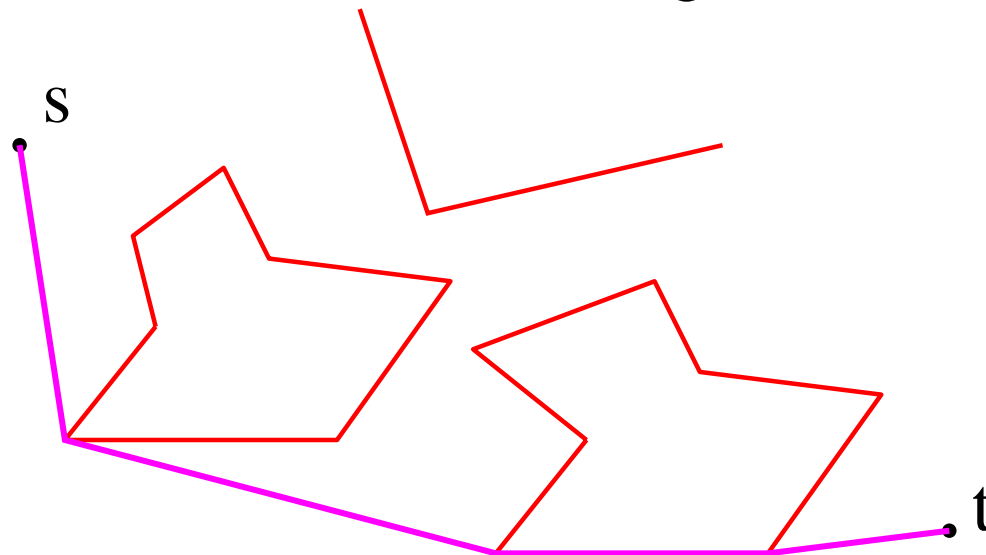
# Kap 1.1: Kürzeste Wege, polygonale Hindernisse

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

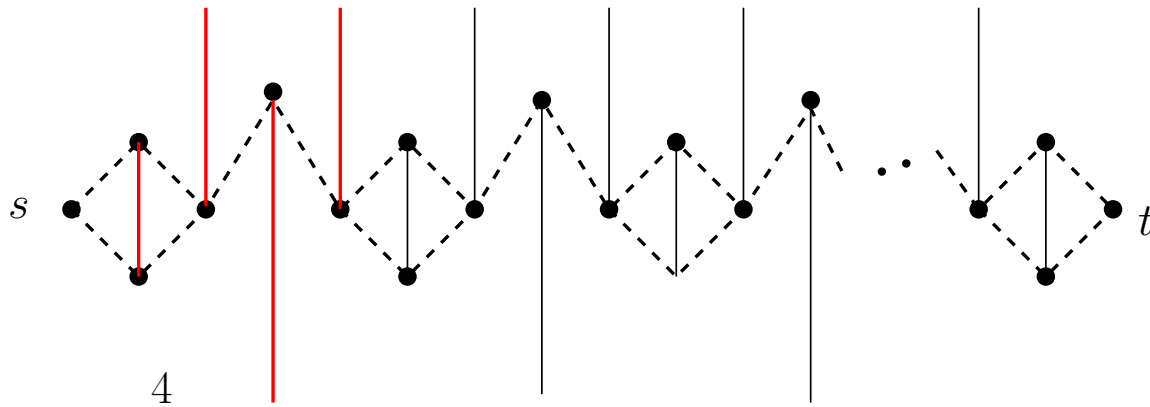


# Kap 1.1: Kürzeste Wege, polygonale Hindernisse

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

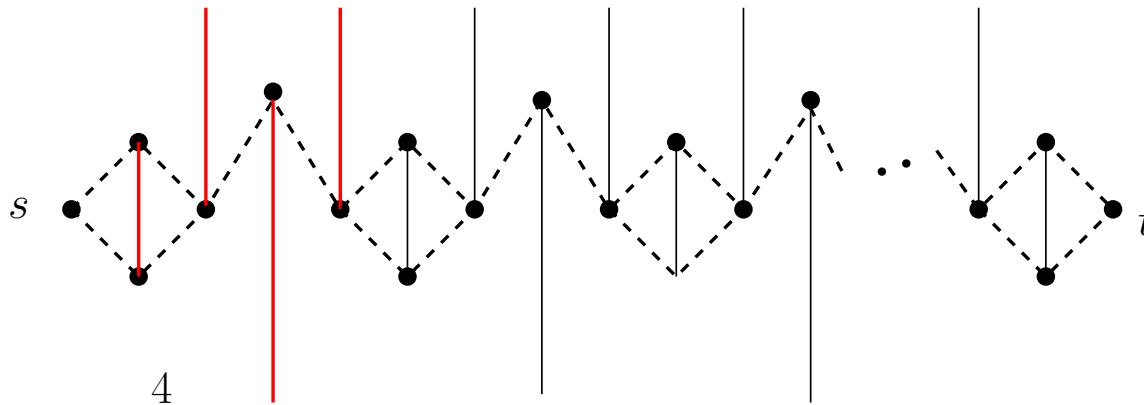


# Eigenschaften Kürzester Wege: Anzahl



# Eigenschaften Kürzester Wege: Anzahl

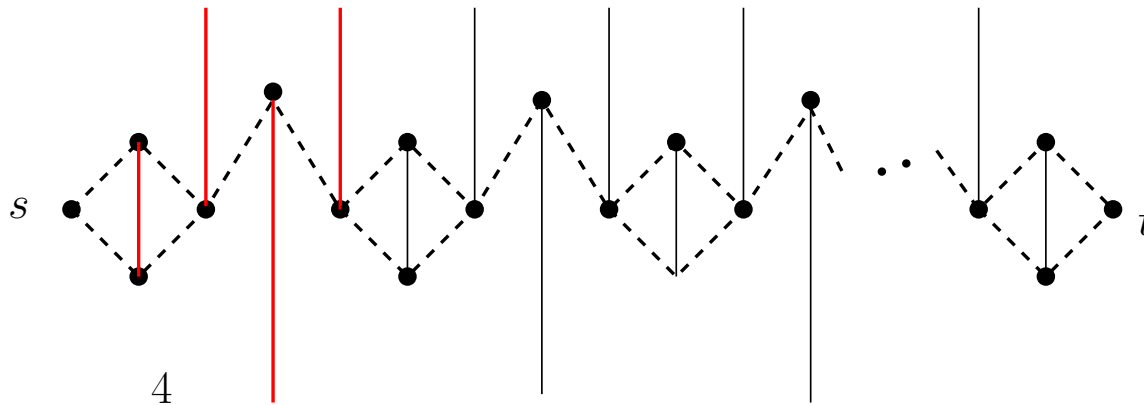
- $m$  Kanten, exponentiell viele (in  $m$ ) kürzeste Wege





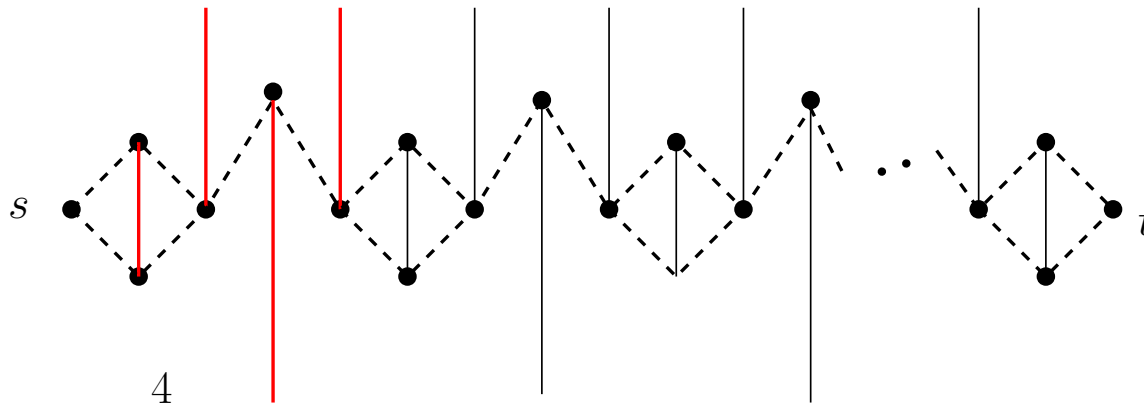
# Eigenschaften Kürzester Wege: Anzahl

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



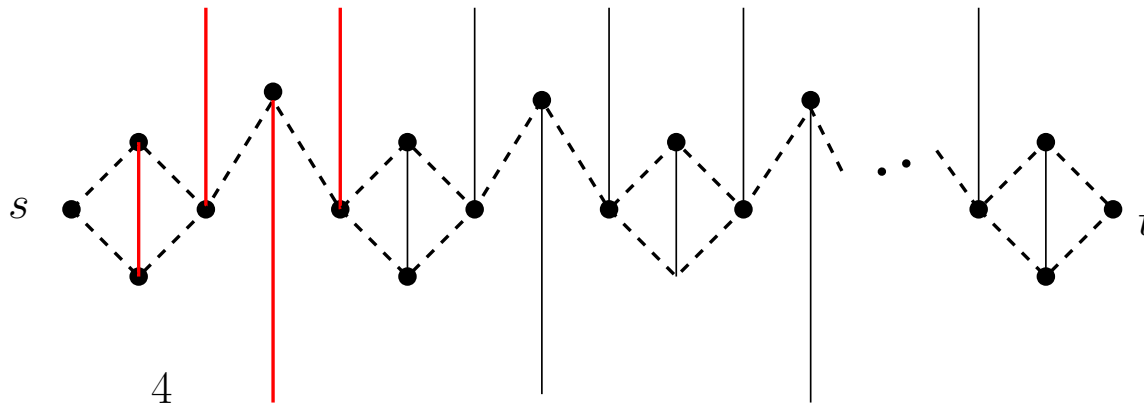
# Eigenschaften Kürzester Wege: Anzahl

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



# Eigenschaften Kürzester Wege: Anzahl

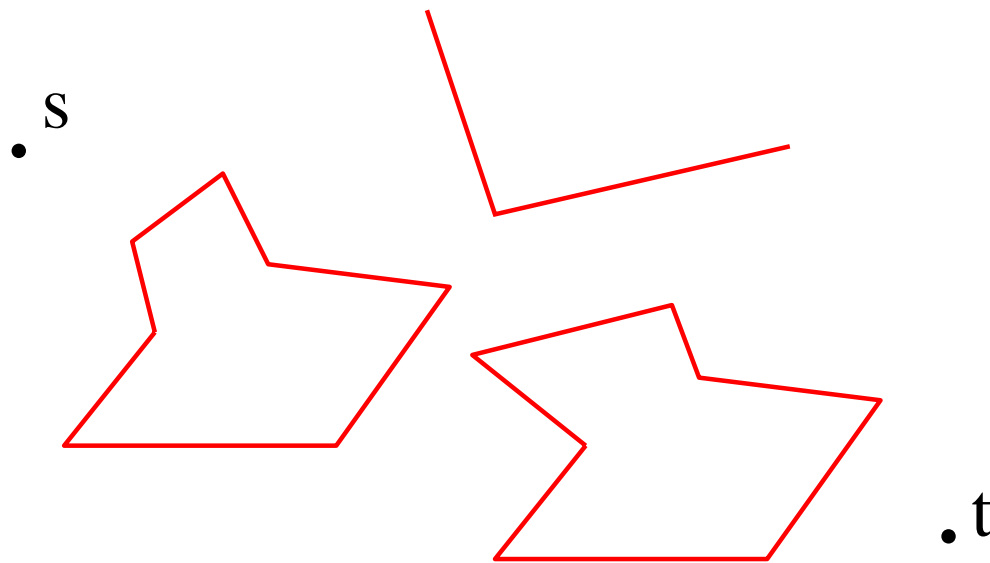
- $m$  Kanten, exponentiell viele (in  $m$ ) kürzeste Wege
- Genauer  $h = 4m + 1$  Kanten,  $2^{m+1}$  kürzeste Wege
- Pro Block verdoppeln, einmal verdoppeln am Ende



# Kürzester Weg: Strukturelle Eigenschaften

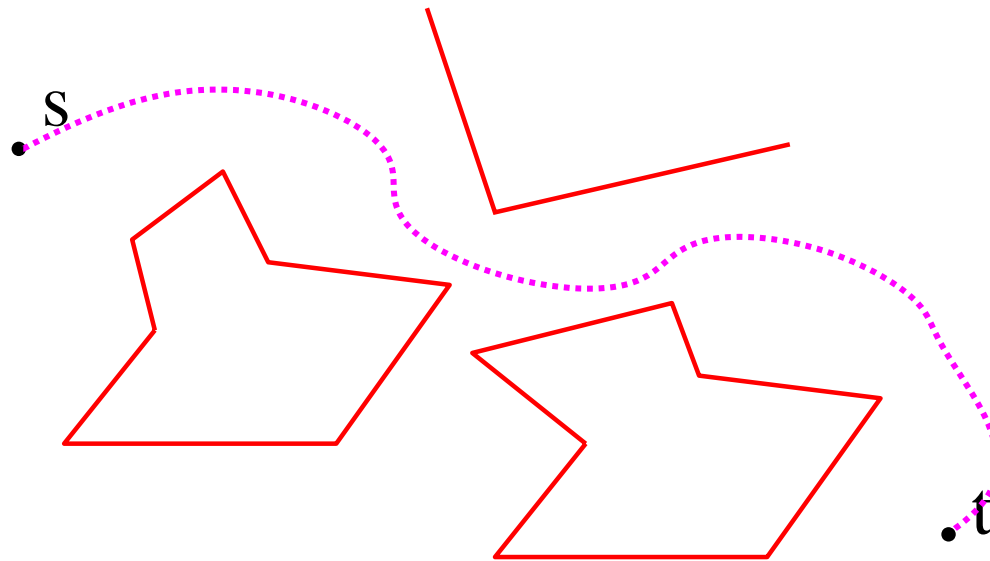
# Kürzester Weg: Strukturelle Eigenschaften

- Polygonale Kette über Ecken,



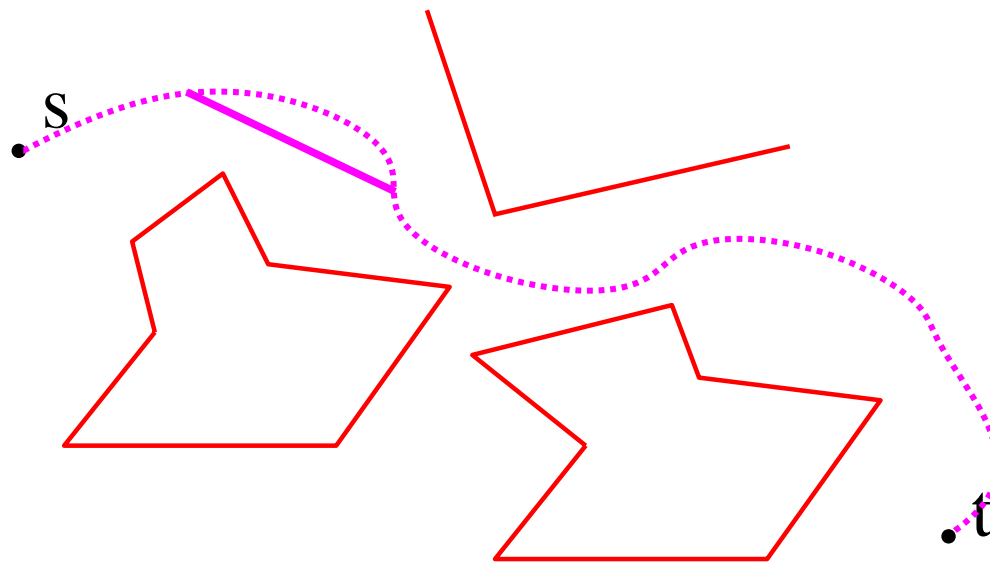
# Kürzester Weg: Strukturelle Eigenschaften

- Polygonale Kette über Ecken,



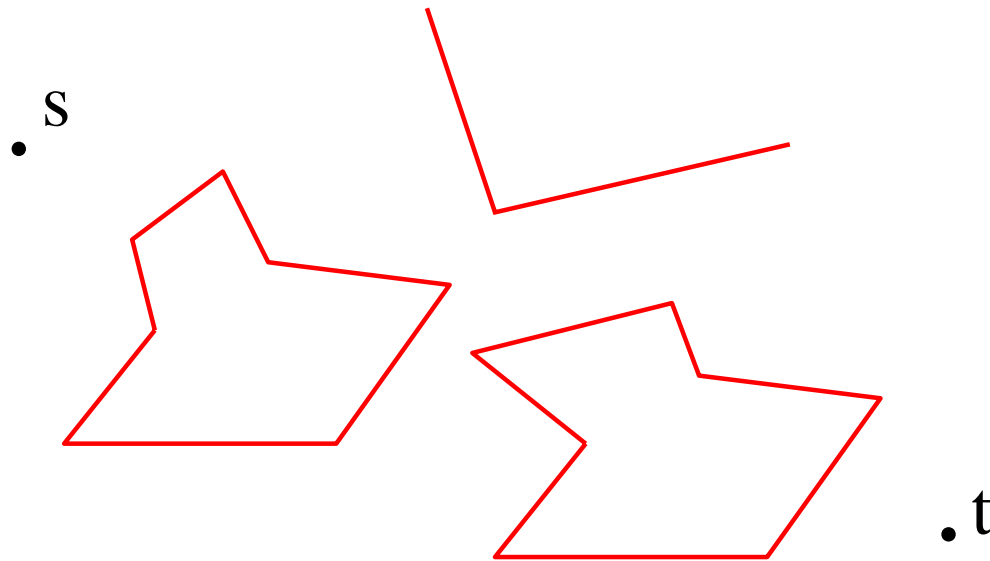
# Kürzester Weg: Strukturelle Eigenschaften

- Polygonale Kette über Ecken, lokal abkürzen



# Kürzester Weg: Strukturelle Eigenschaften

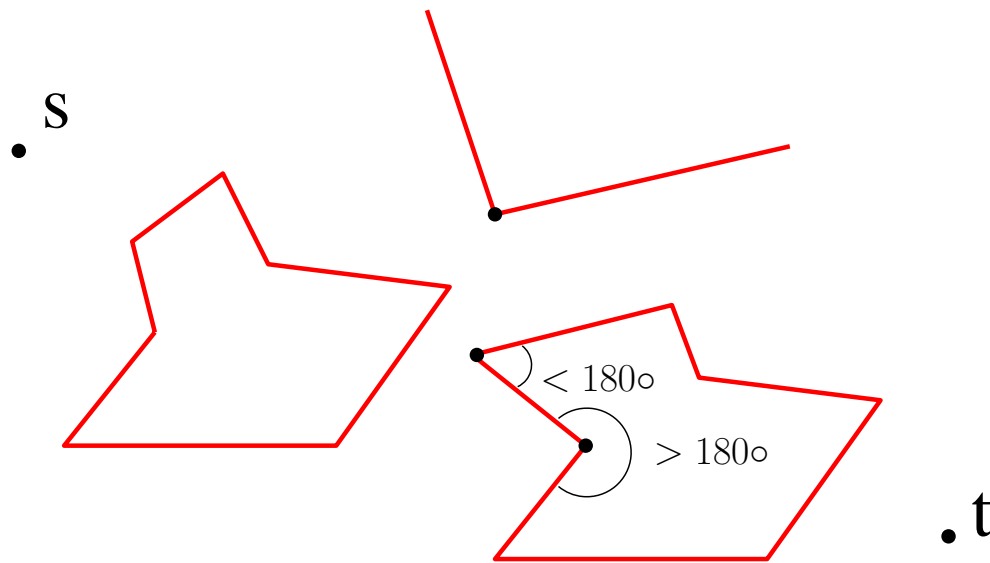
- Polygonale Kette über Ecken, lokal abkürzen
- Nur über konvexe Ecken der Hindernisse





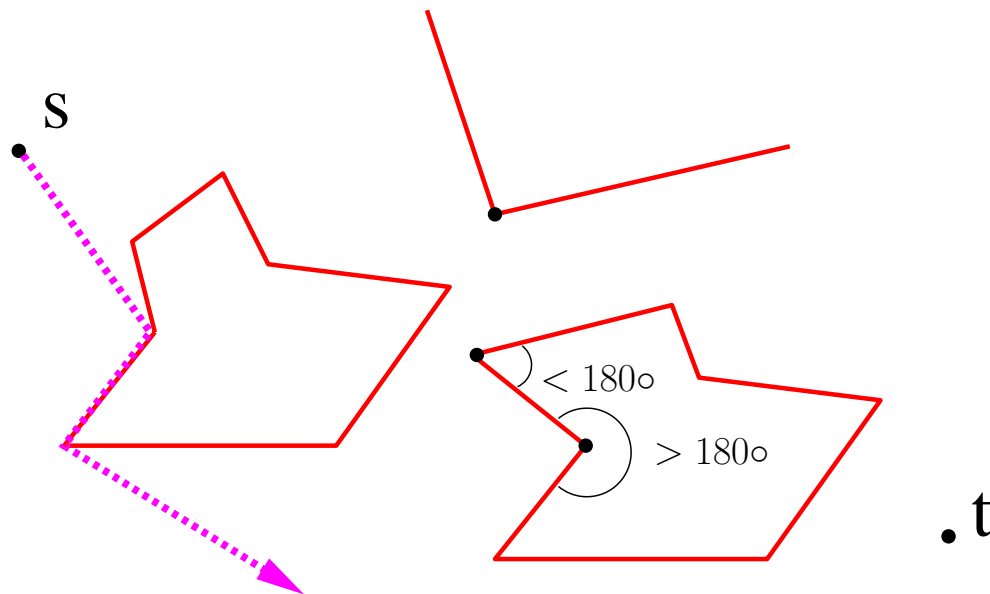
# Kürzester Weg: Strukturelle Eigenschaften

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



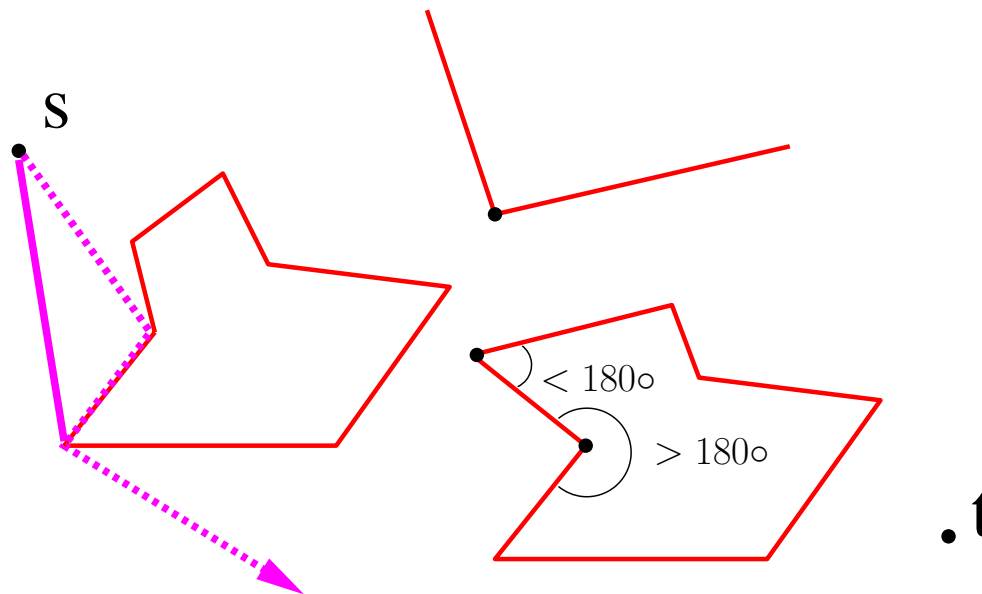
# Kürzester Weg: Strukturelle Eigenschaften

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



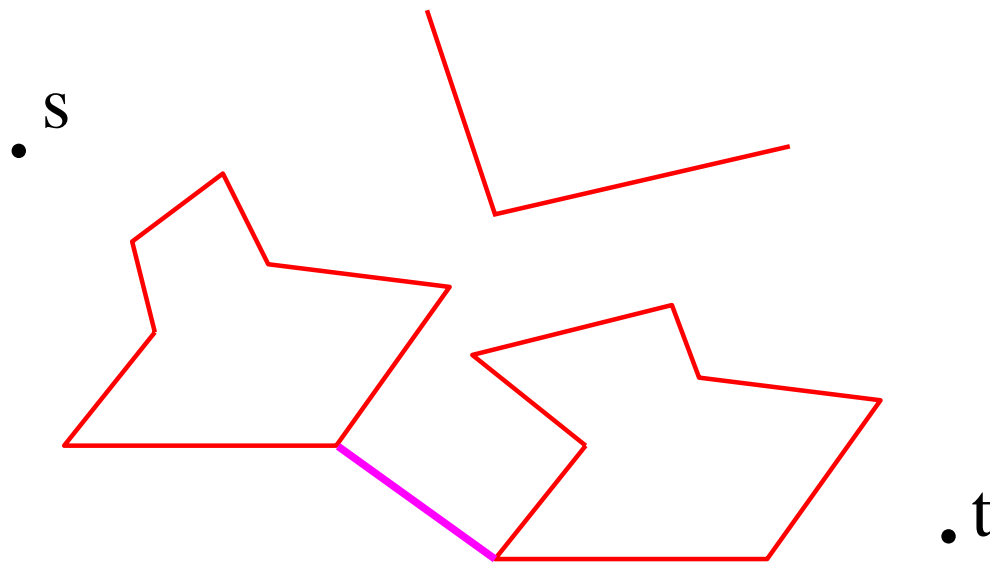
# Kürzester Weg: Strukturelle Eigenschaften

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



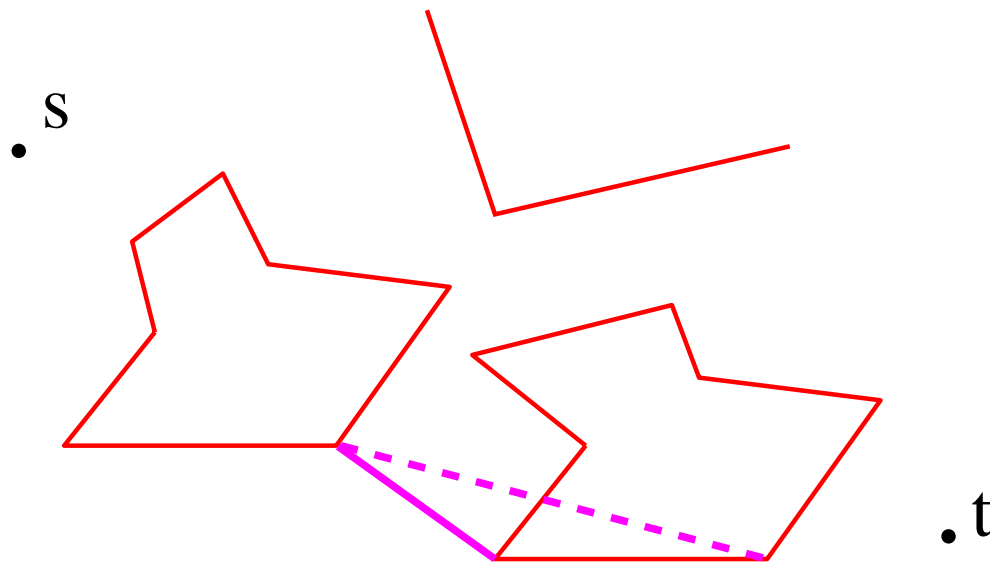
# Kürzester Weg: Strukturelle Eigenschaften

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



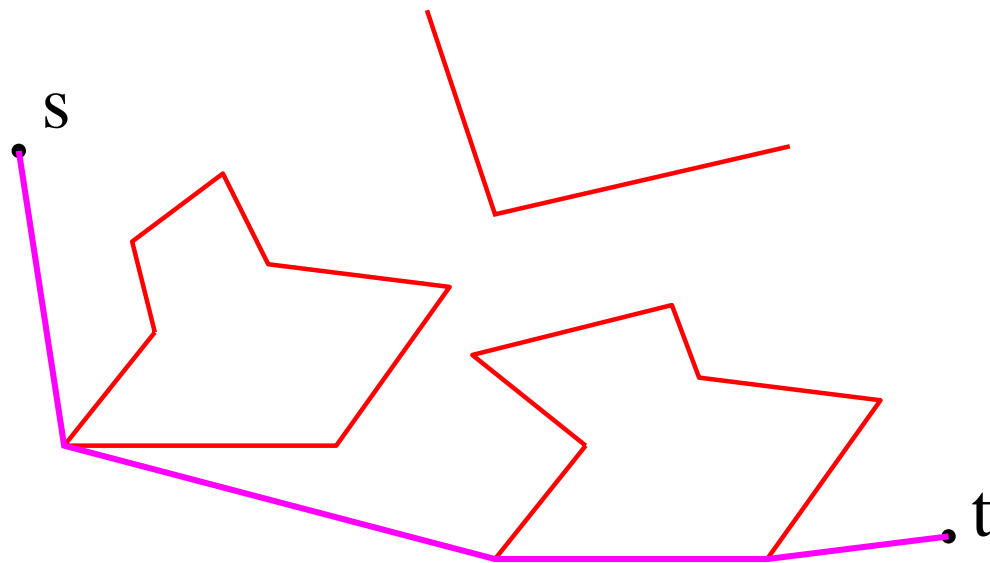
# Kürzester Weg: Strukturelle Eigenschaften

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



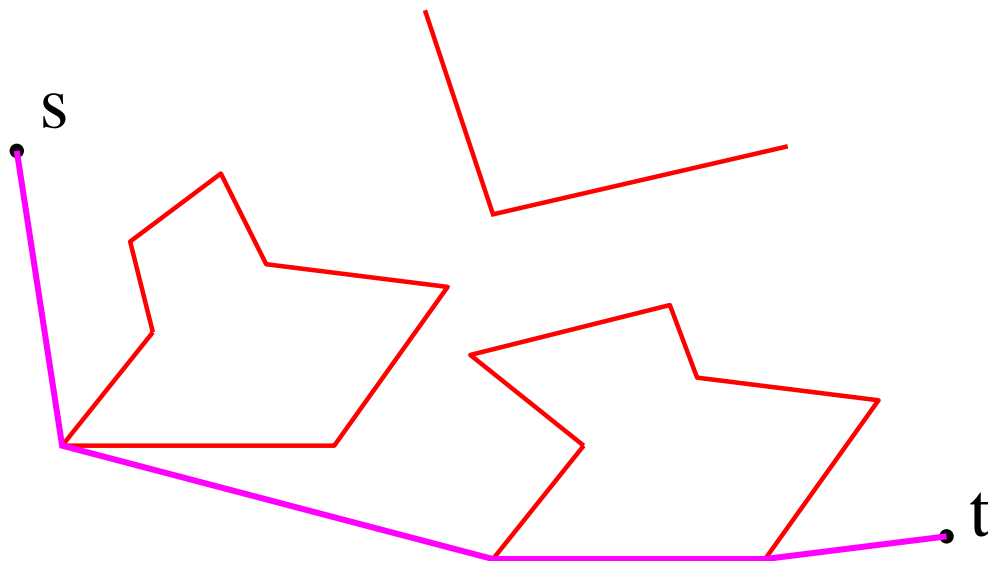
# Kürzester Weg: Strukturelle Eigenschaften

- 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

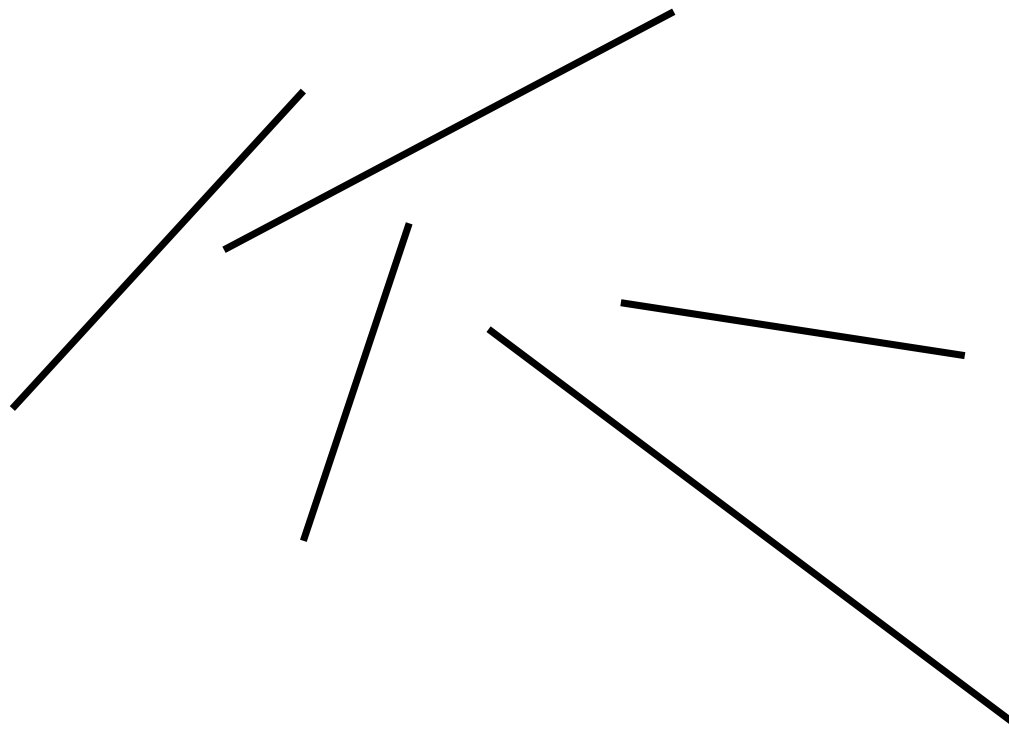


# Allg.: Sichtbarkeitsgraph für Liniensegmente



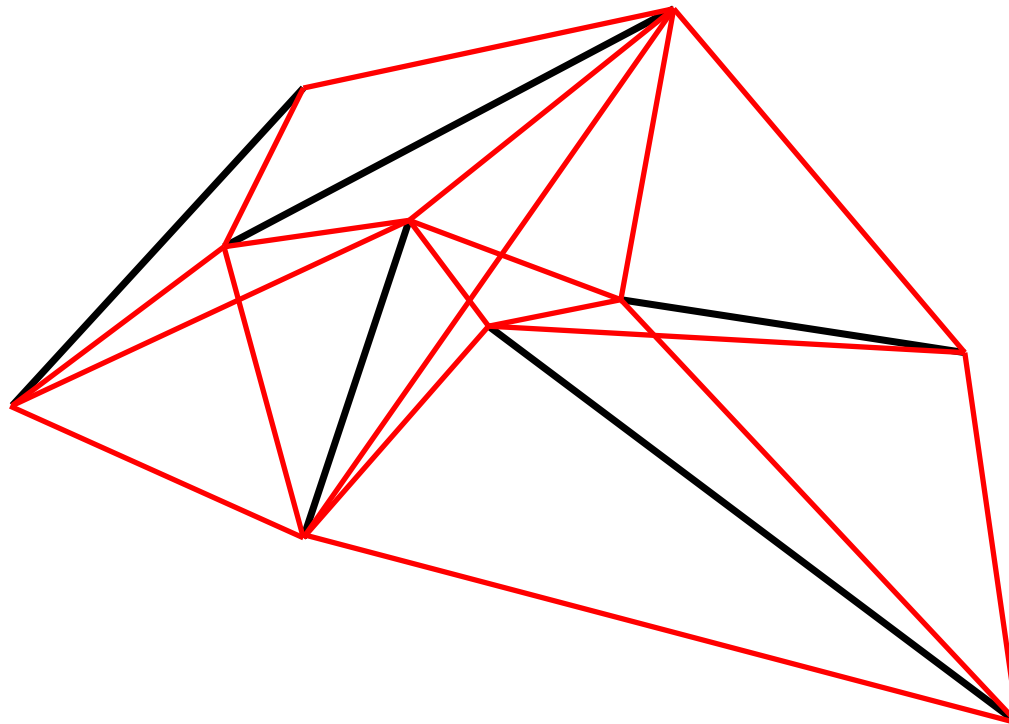
# Allg.: Sichtbarkeitsgraph für Liniensegmente

- Menge  $L$  von nicht-schneidenden Segmenten gegeben



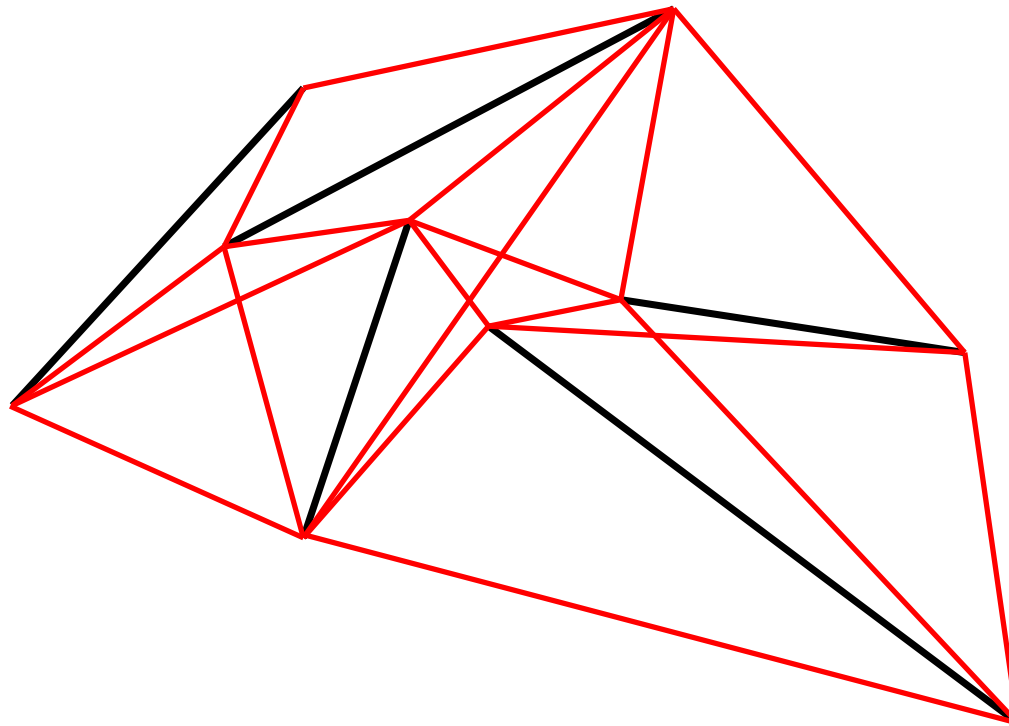
# Allg.: Sichtbarkeitsgraph für Liniensegmente

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



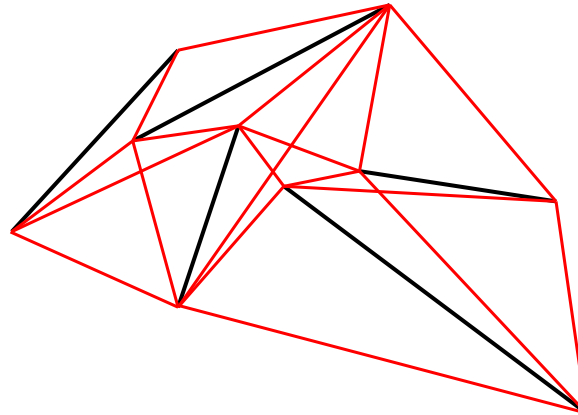
# Allg.: Sichtbarkeitsgraph für Liniensegmente

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

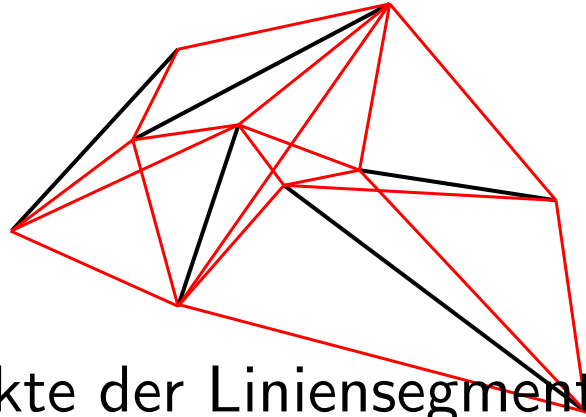


**Def. 1.1: Sichtbarkeitsgraph**  $VisG(L) = (V, E)$

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



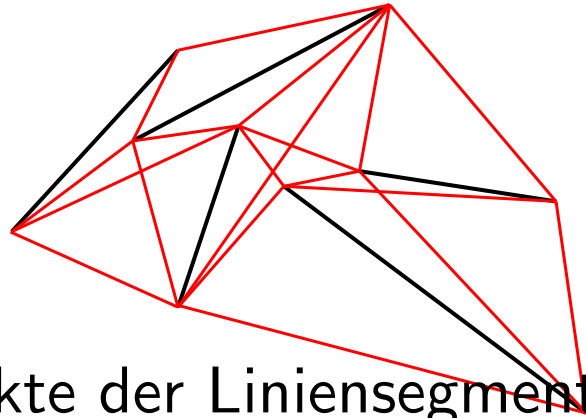
**Def. 1.1: Sichtbarkeitsgraph**  $\text{VisG}(L) = (V, E)$



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

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

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

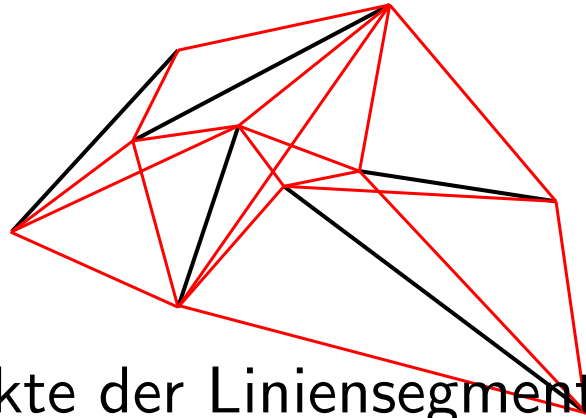


$V = \{ \text{alle Endpunkte der Liniensegmente in } L \}$

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

- Sichtbarkeitsgraph einer Menge von Polygonen  $P_i$

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



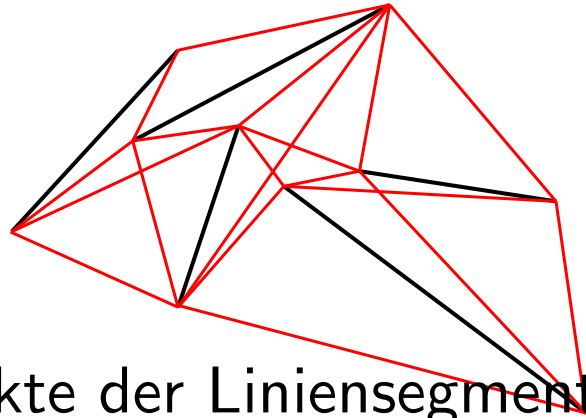
$V = \{ \text{alle Endpunkte der Liniensegmente in } L \}$

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

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



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

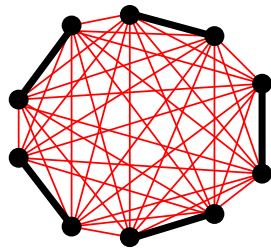


$V = \{ \text{alle Endpunkte der Liniensegmente in } L \}$

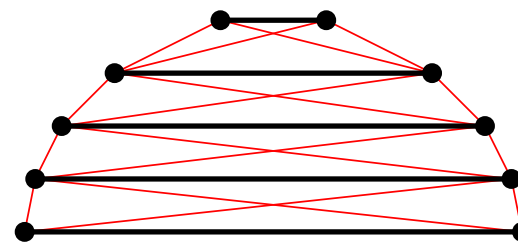
$E = \{ (p, q) \mid p, q \in V, \overline{pq} \text{ kreuzt kein Liniensegment aus } L \}$ .

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

# Komplexität Sichtbarkeitsgraph $VisG(L) = (V, E)$



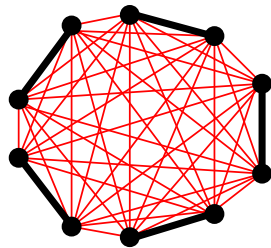
(i)



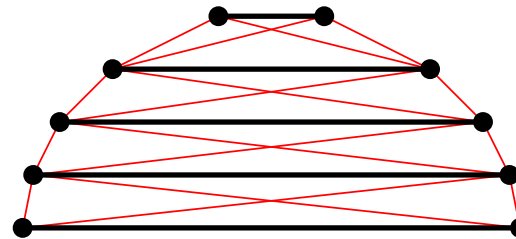
(ii)

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

- $n$  Liniensegmente



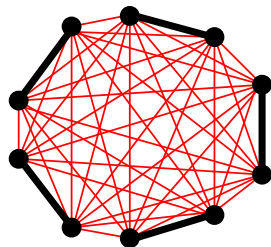
(i)



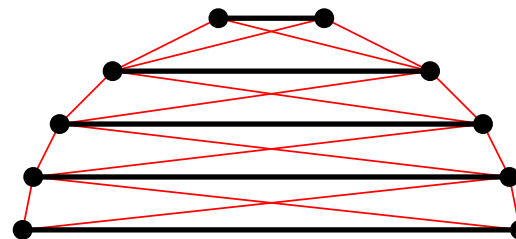
(ii)

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

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



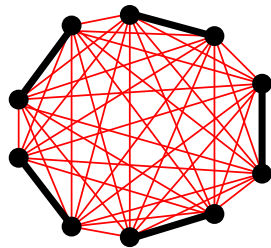
(i)



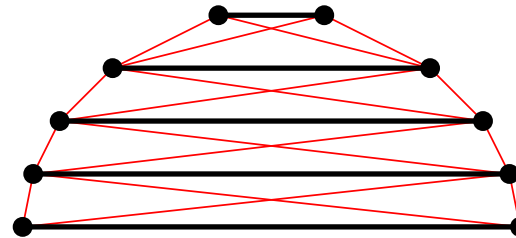
(ii)

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

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



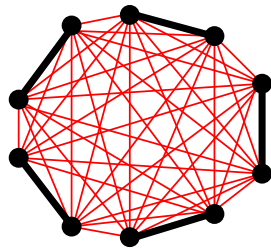
(i)



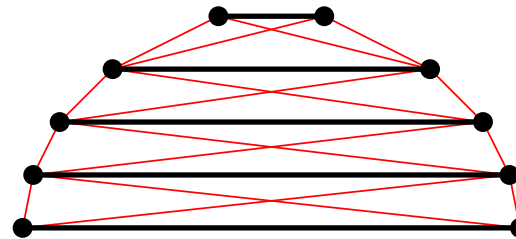
(ii)

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

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



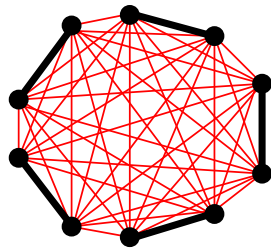
(i)



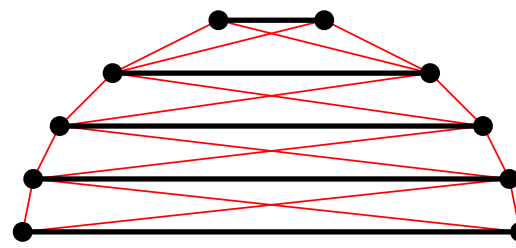
(ii)

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

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



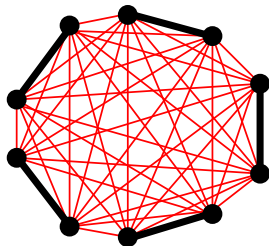
(i)



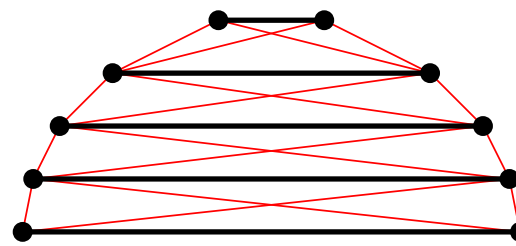
(ii)

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

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



(i)

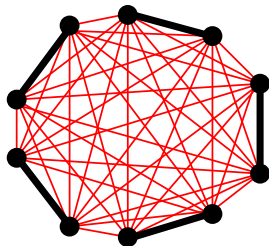


(ii)

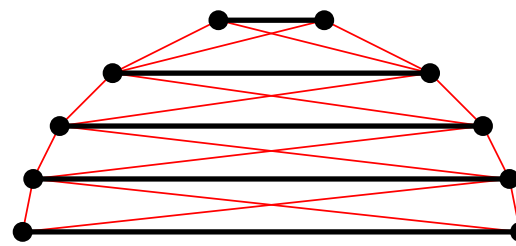


# Komplexität Sichtbarkeitsgraph $\text{VisG}(L) = (V, E)$

- $n$  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



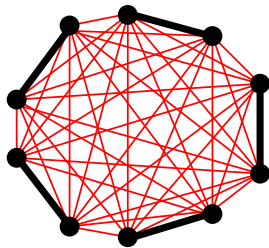
(i)



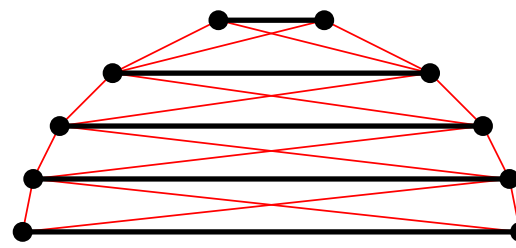
(ii)

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

- $n$  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



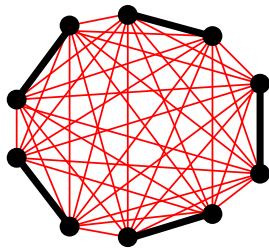
(i)



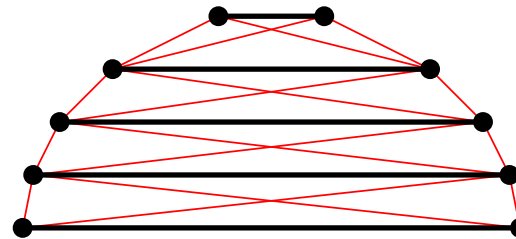
(ii)

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

- $n$  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

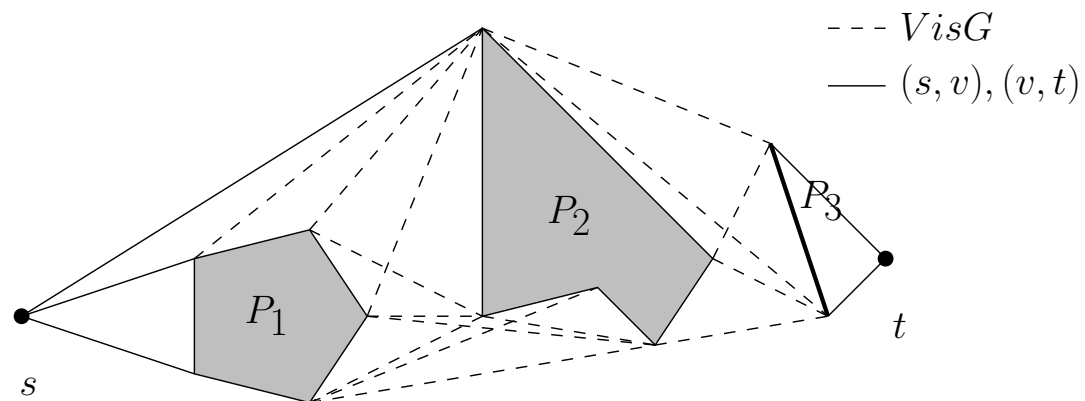


(i)



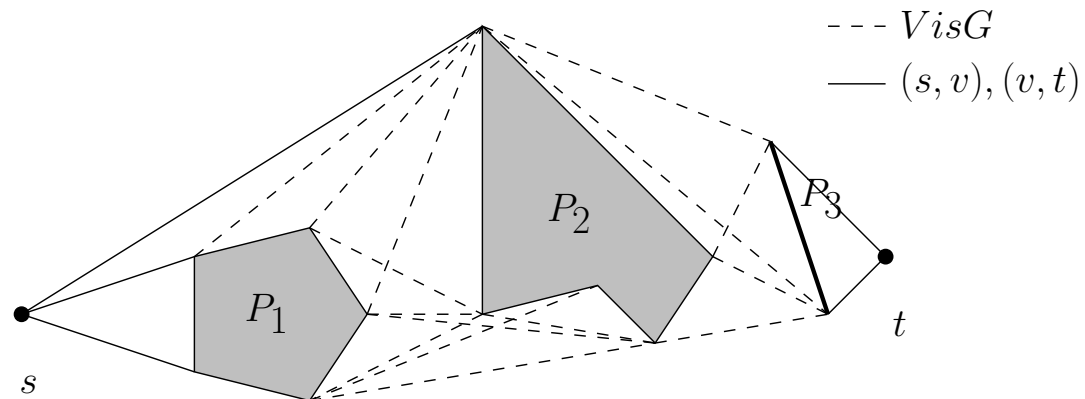
(ii)

# Was bringt der Sichtbarkeitsgraph?



# Was bringt der Sichtbarkeitsgraph?

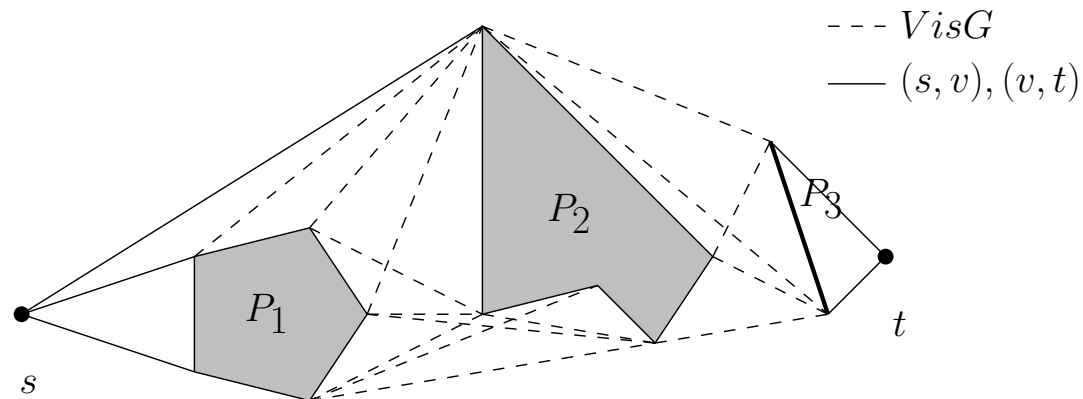
**Alg. 1.1:** Berechnung des kürzesten Weges mittels  $\text{VisG}(P)$



# Was bringt der Sichtbarkeitsgraph?

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

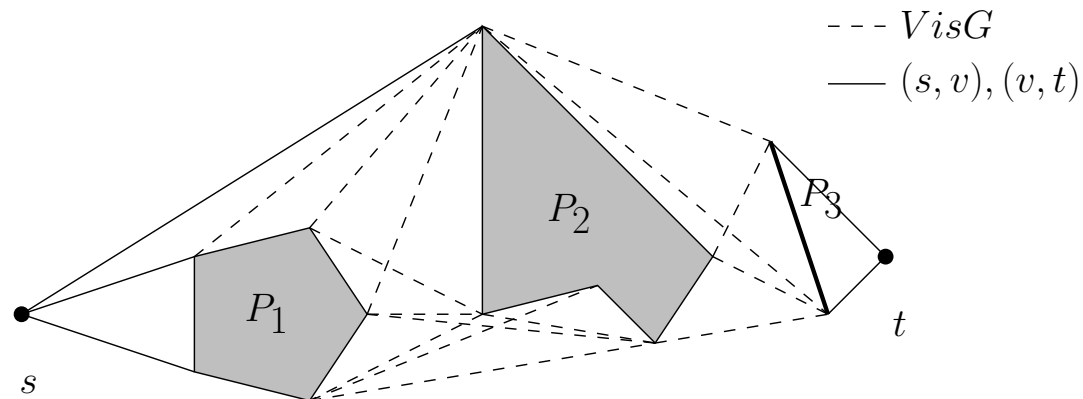
1. Sichtbarkeitsgraph  $(V, E)$  für Hindernisse:



# Was bringt der Sichtbarkeitsgraph?

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

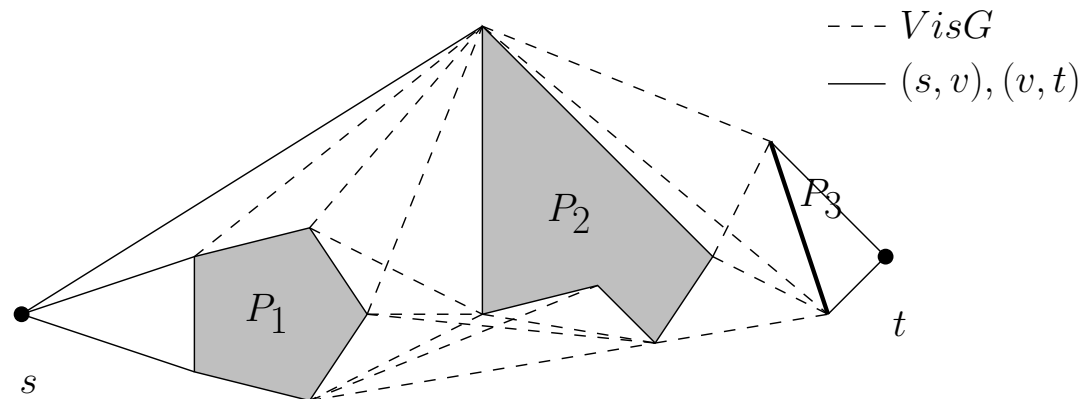
1. Sichtbarkeitsgraph  $(V, E)$  für Hindernisse:  $O(n^2)$



# Was bringt der Sichtbarkeitsgraph?

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

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

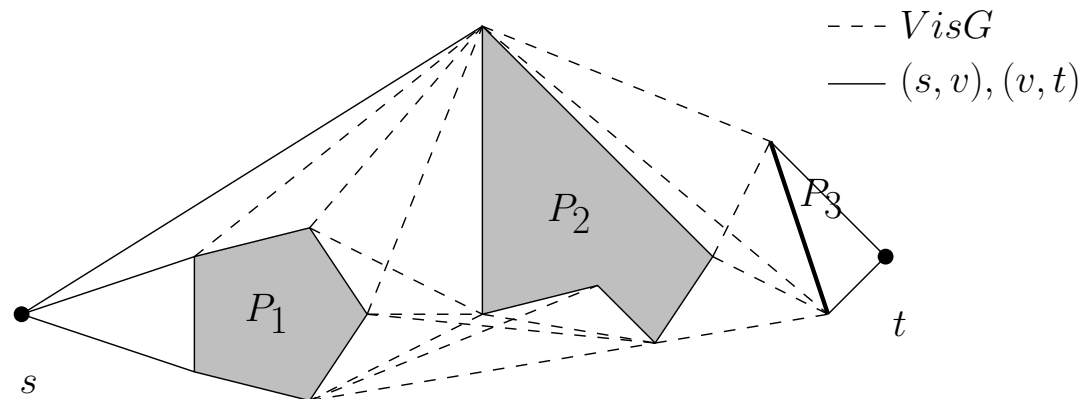




# Was bringt der Sichtbarkeitsgraph?

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

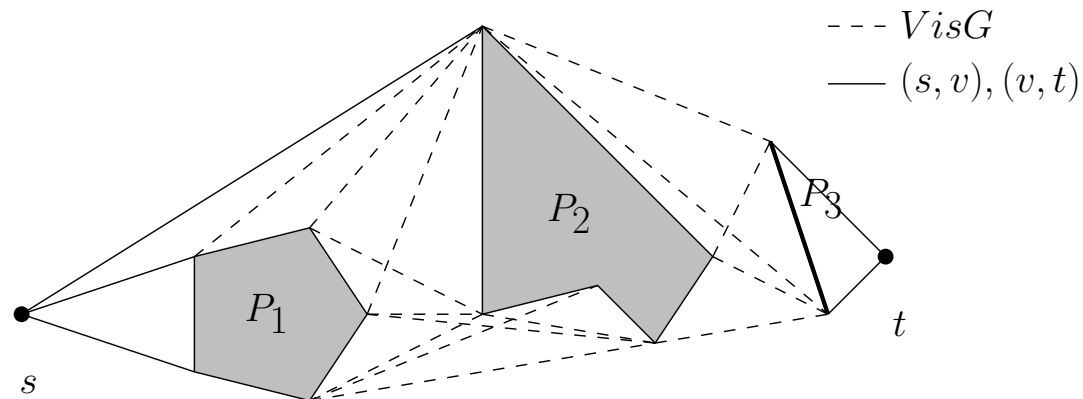
1. Sichtbarkeitsgraph  $(V, E)$  für Hindernisse:  $O(n^2)$
2. Sichtbare Kanten von  $s$  aus:  $E_1$ ,  $O(n^2)$



# Was bringt der Sichtbarkeitsgraph?

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

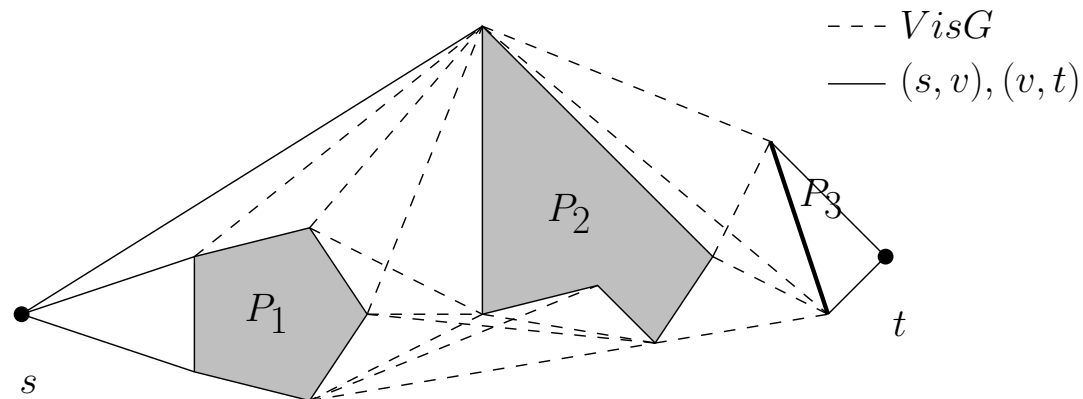
1. Sichtbarkeitsgraph  $(V, E)$  für Hindernisse:  $O(n^2)$
2. Sichtbare Kanten von  $s$  aus:  $E_1$ ,  $O(n^2)$
3. Sichtbare Kanten von  $t$  aus:  $E_2$ ,



# Was bringt der Sichtbarkeitsgraph?

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

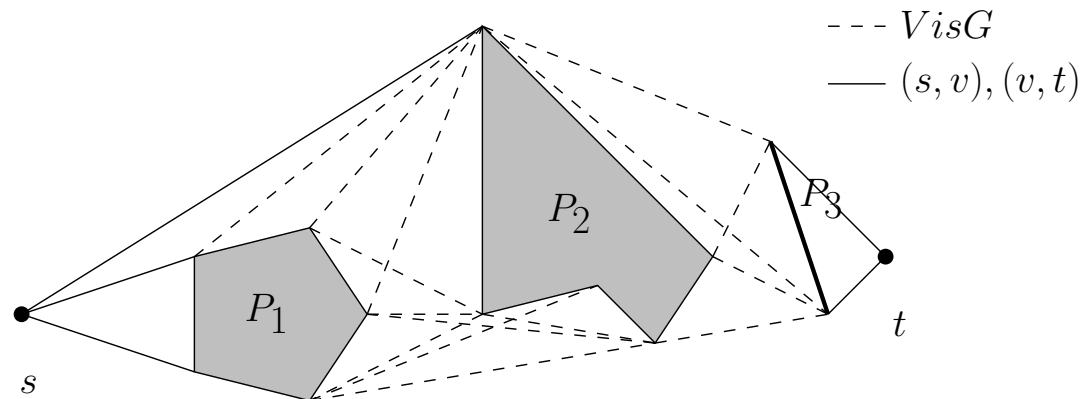
1. Sichtbarkeitsgraph  $(V, E)$  für Hindernisse:  $O(n^2)$
2. Sichtbare Kanten von  $s$  aus:  $E_1, O(n^2)$
3. Sichtbare Kanten von  $t$  aus:  $E_2, O(n^2)$



# Was bringt der Sichtbarkeitsgraph?

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

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

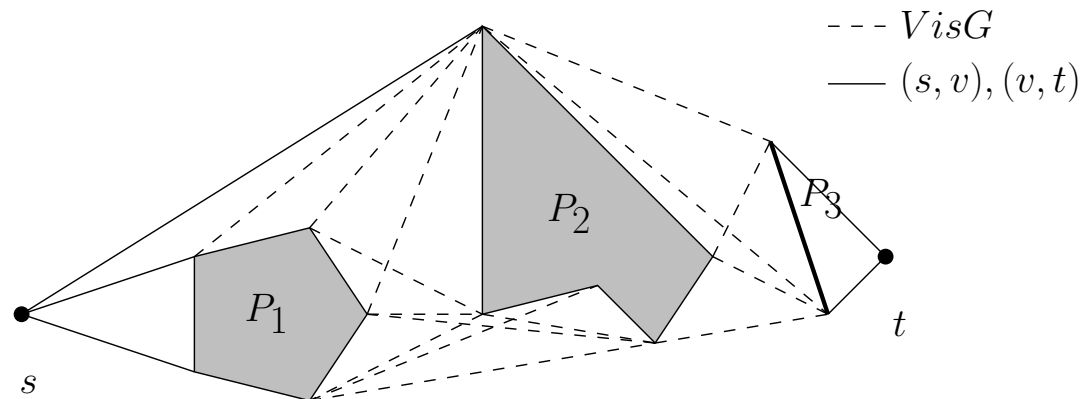


# Was bringt der Sichtbarkeitsgraph?

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

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

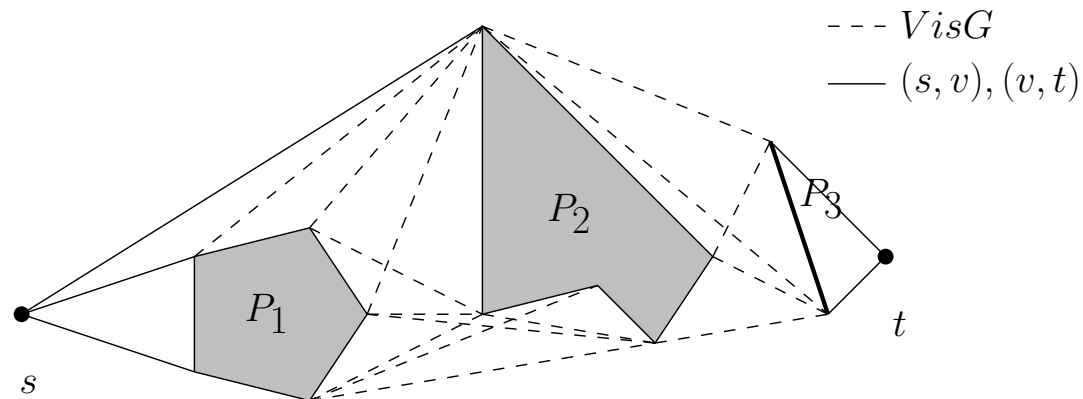
2. und 3.:



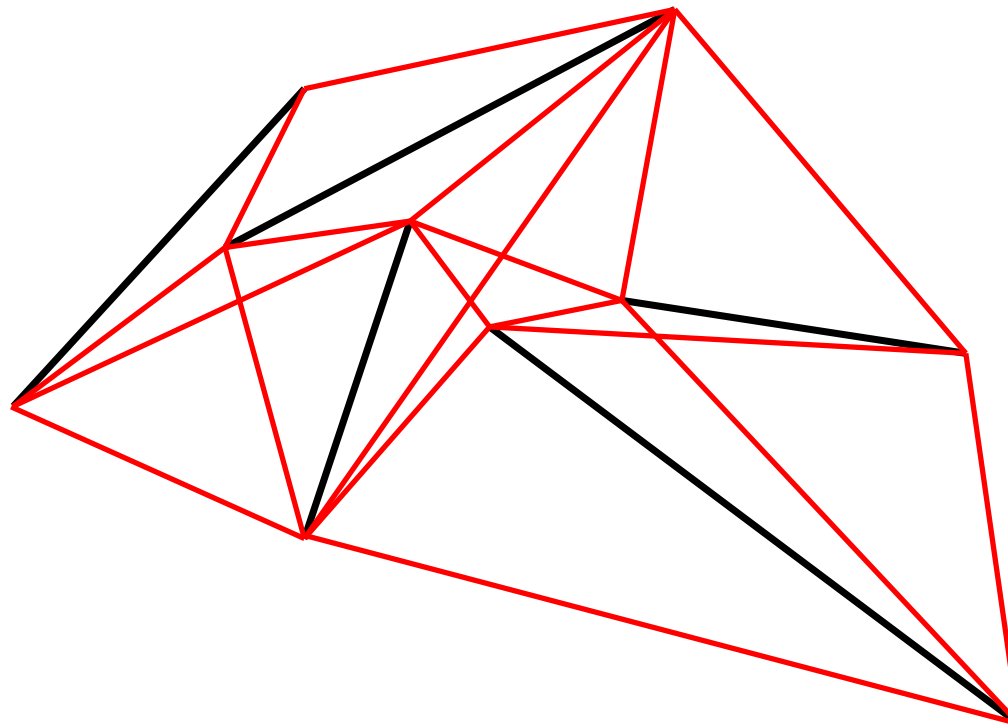
# Was bringt der Sichtbarkeitsgraph?

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

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

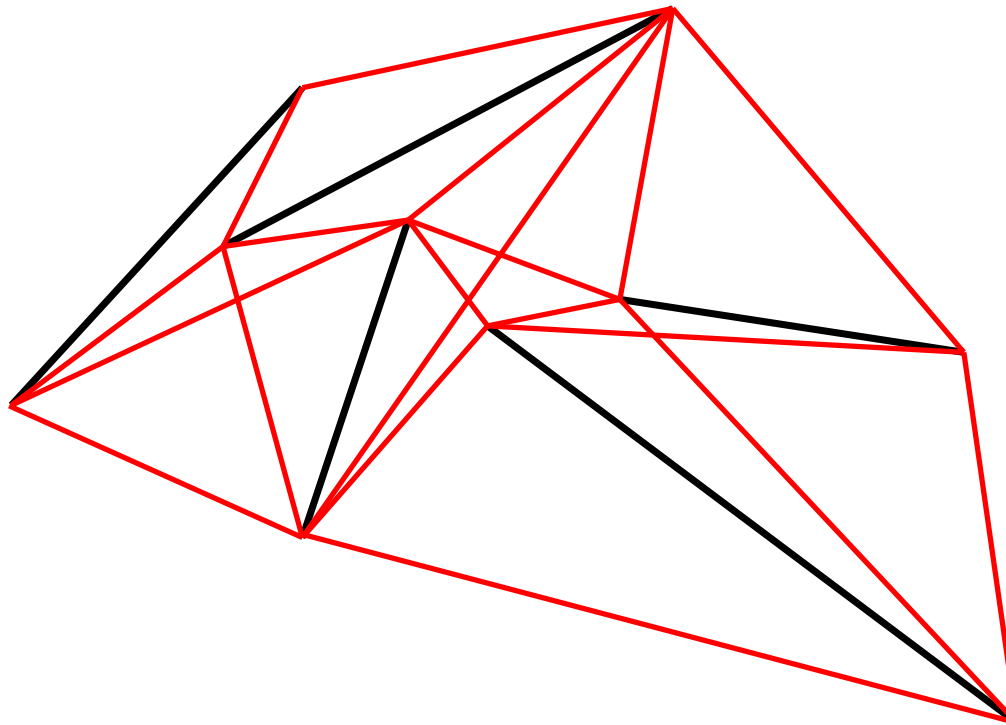


# Sichtbarkeitsgraph für Menge von Liniensegmenten



# Sichtbarkeitsgraph für Menge von Liniensegmenten

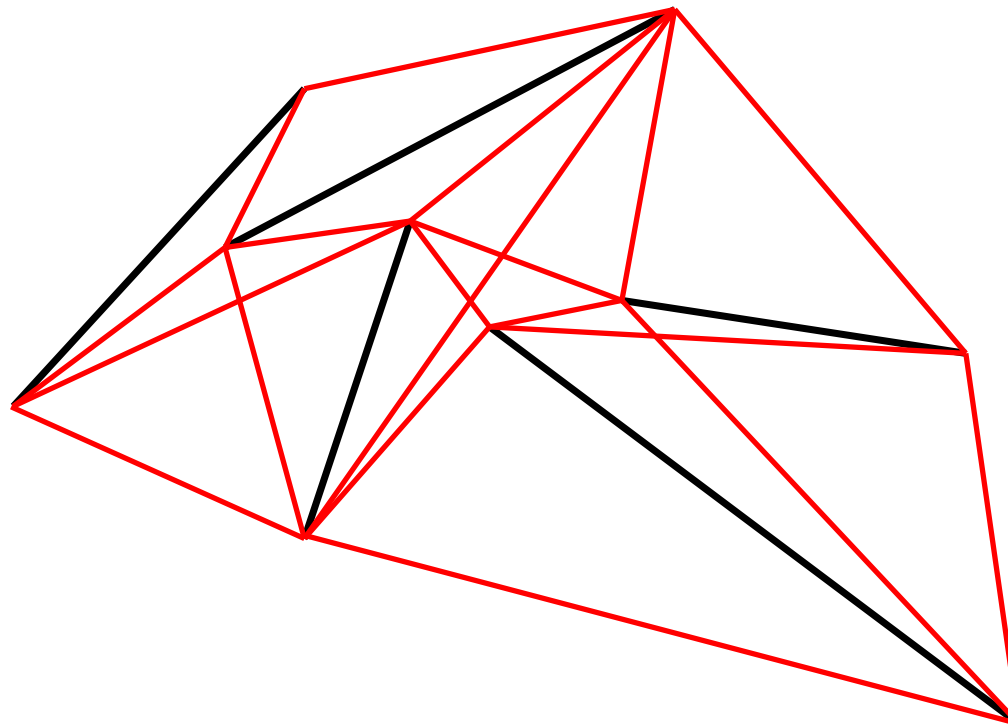
- Naiv:  $O(n^3)$





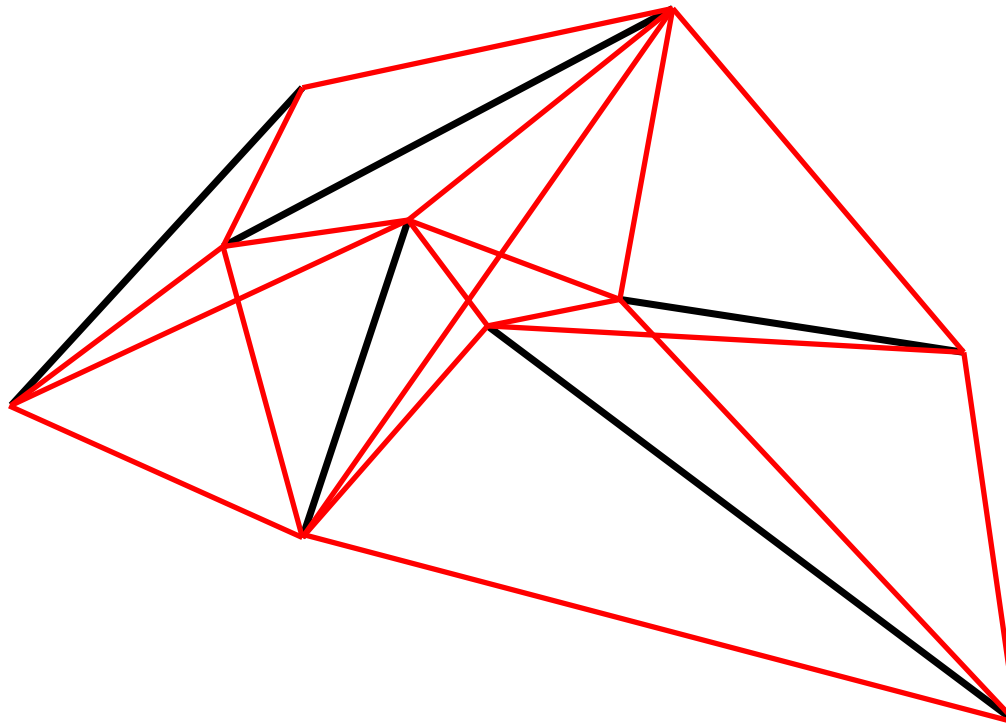
# Sichtbarkeitsgraph für Menge von Liniensegmenten

- 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)  $\times$  Kosten(Verab)

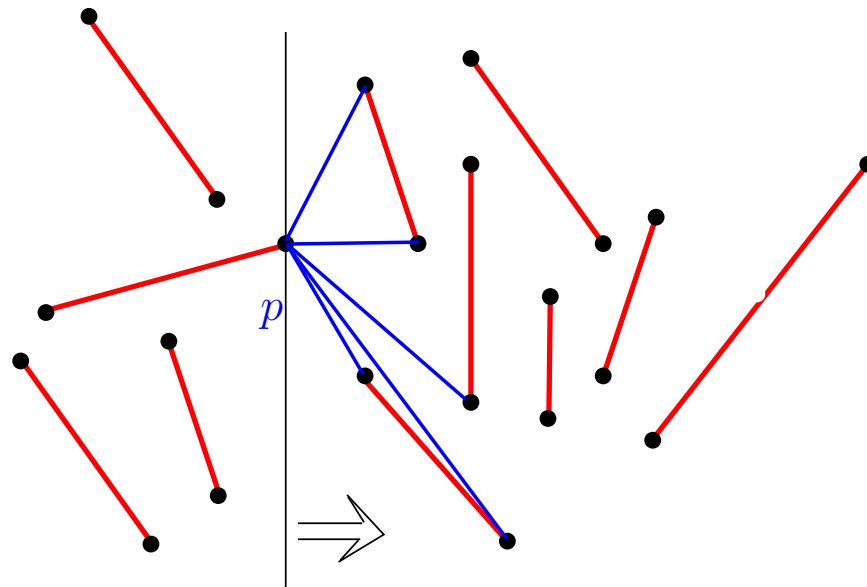
# Sweep Technik

- 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)  $\times$  Kosten(Verab)
- Korrektheit Ergebnis: Invariante erfüllt

# Sweep Technik

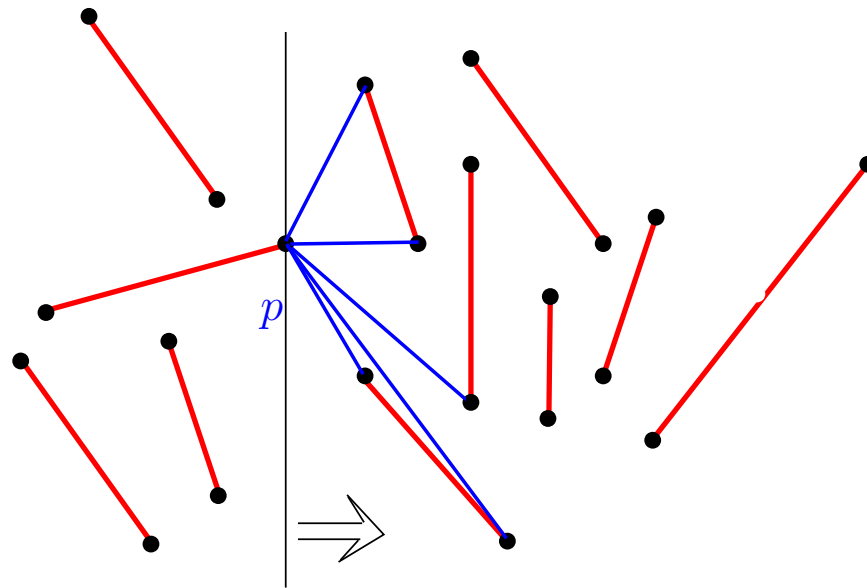
- Ausfegen der Ebene mit Sweepline
- Komplexitätsreduktion: 2-Dim nach 1-Dim
- Sortieren und gem. Sort. fegen
- SSS (Sweep-Status-Struktur): Invariante in der Nähe der Sweepline
- ES (Ereignisstruktur): Haltepunkte der Sweepline
- Ereignisverarbeitung: Aktualisierung SSS
- Laufzeitanalyse: ( $\#$  Ereignisse)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

# Single-Source Sweep: Punkt $p$



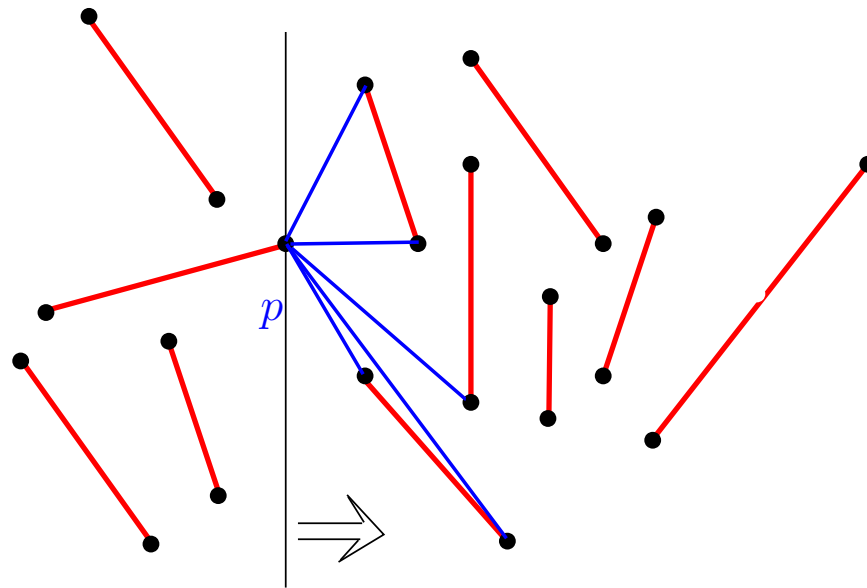
# Single-Source Sweep: Punkt $p$

- Für jeden Endpunkt  $p$  die sichtbaren Segmente bestimmen



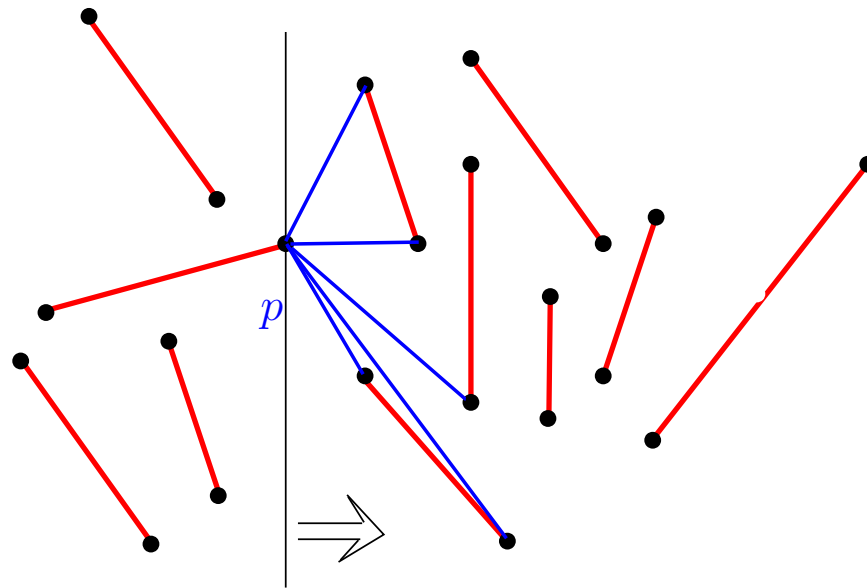
# Single-Source Sweep: Punkt $p$

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



# Single-Source Sweep: Punkt $p$

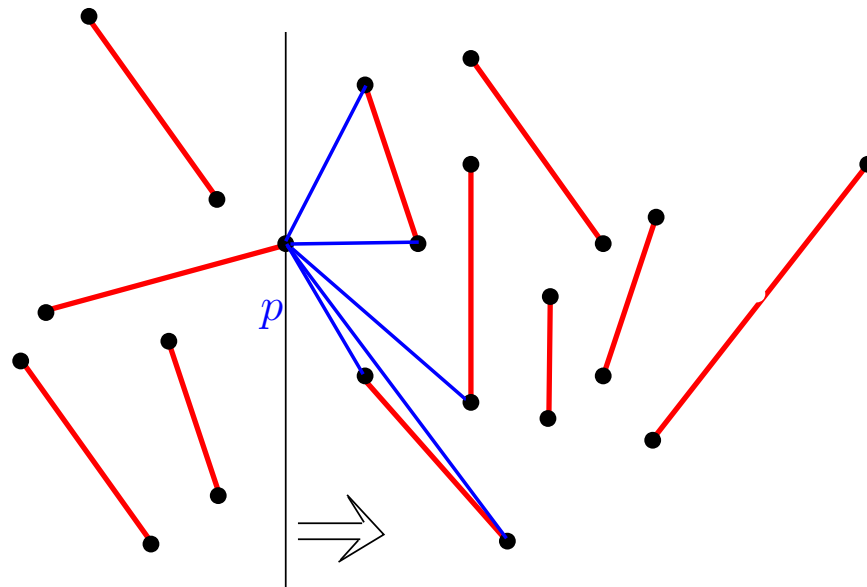
- Für jeden Endpunkt  $p$  die sichtbaren Segmente bestimmen
- Nur zu einer Seite reicht aus
- Vereinigung dieser Kanten ergibt Sichtbarkeitsgraph



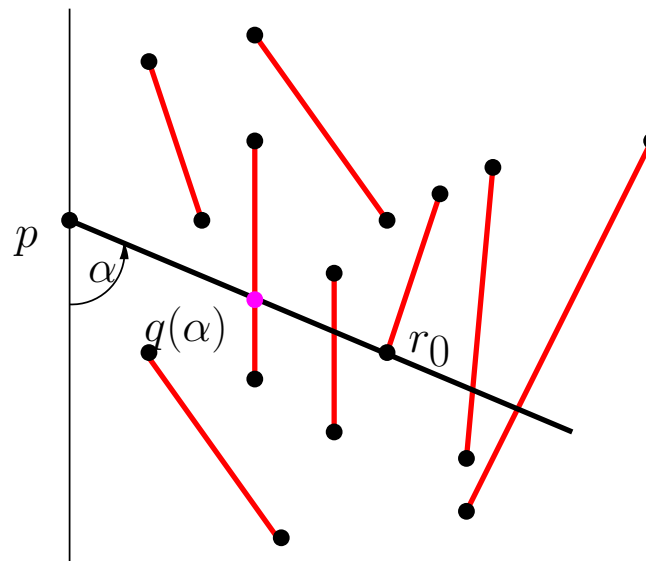


# Single-Source Sweep: Punkt $p$

- 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

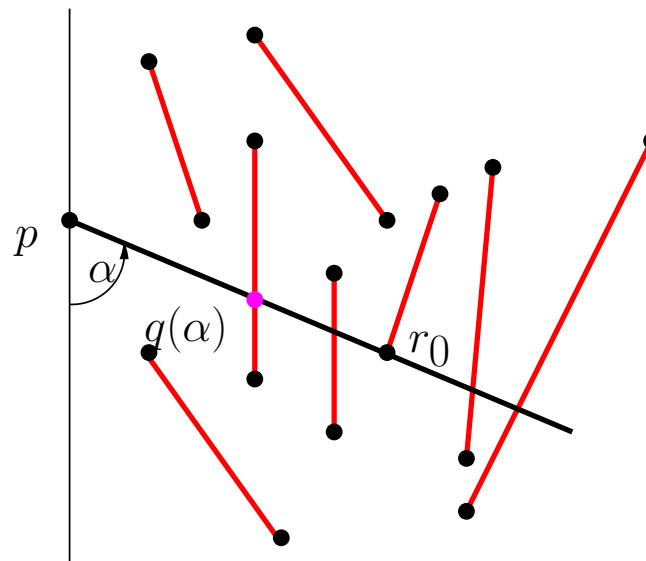


# Alg. 1.2: Single-Source Sweep: Punkt $p$



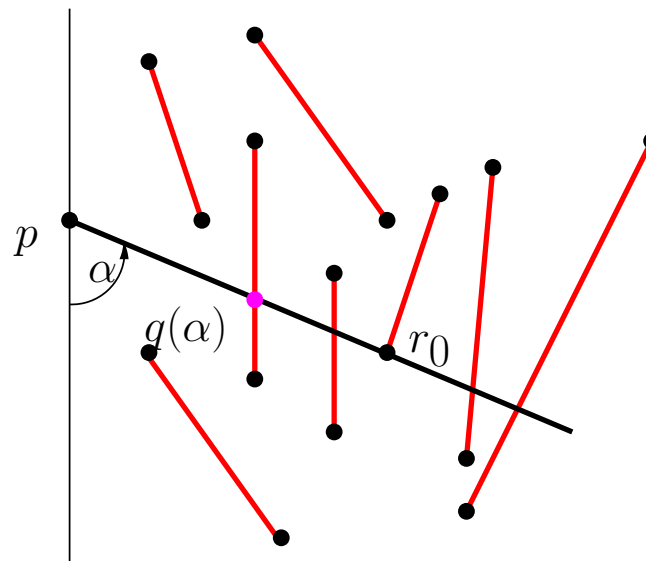
# Alg. 1.2: Single-Source Sweep: Punkt $p$

- ES: Punkte rechts von  $p$  sortiert nach Polarkoordinaten



## Alg. 1.2: Single-Source Sweep: Punkt $p$

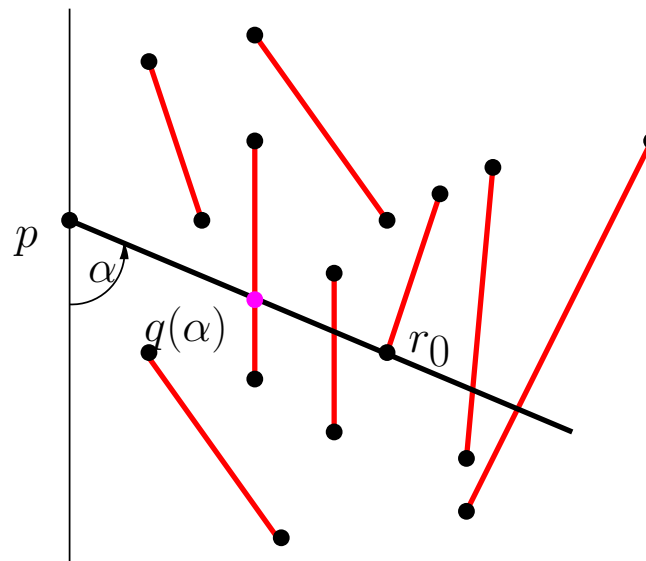
- ES: Punkte rechts von  $p$  sortiert nach Polarkoordinaten
- SSS: Balanc. Baum (Liste der Segmente)/dynam. Schlüssel (Entfernung)



## Alg. 1.2: Single-Source Sweep: Punkt $p$

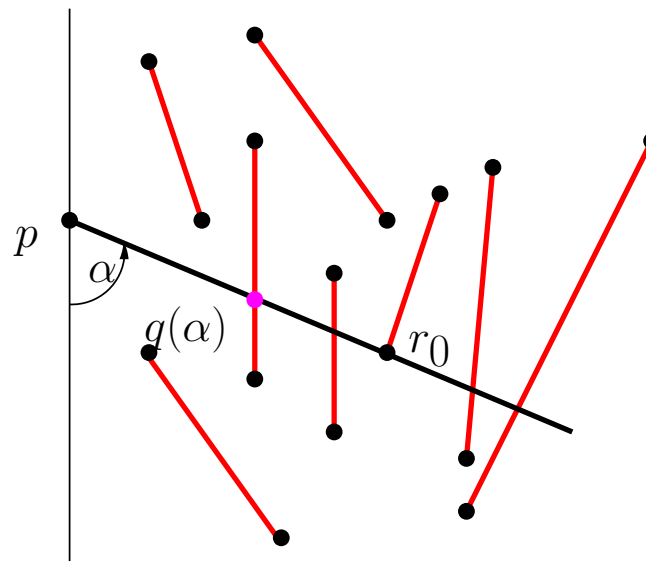
- ES: Punkte rechts von  $p$  sortiert nach Polarkoordinaten
- SSS: Balanc. Baum (Liste der Segmente)/dynam. Schlüssel (Entfernung)

Vorderster Punkt  $q(\alpha)$

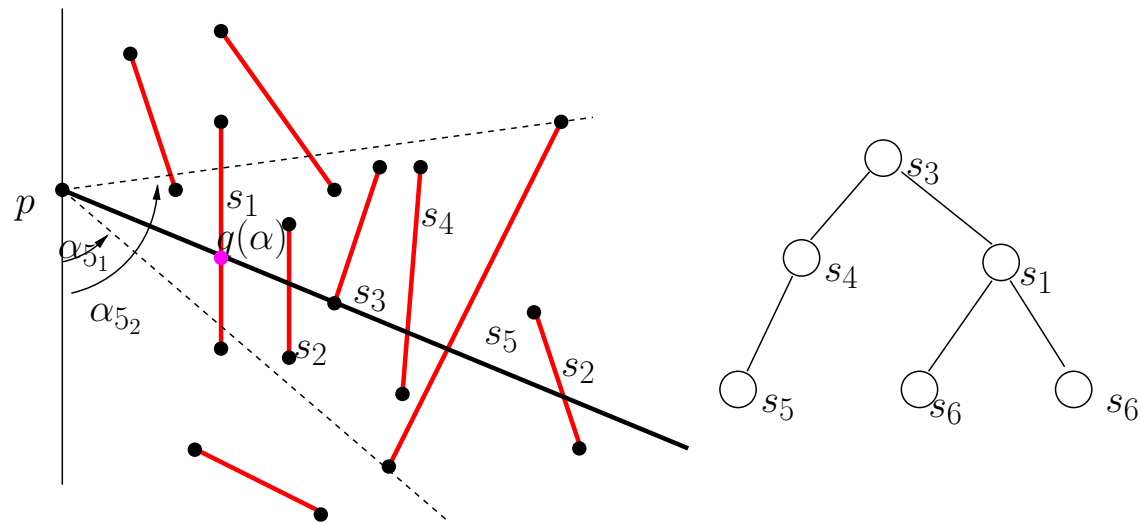


## Alg. 1.2: Single-Source Sweep: Punkt $p$

- 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

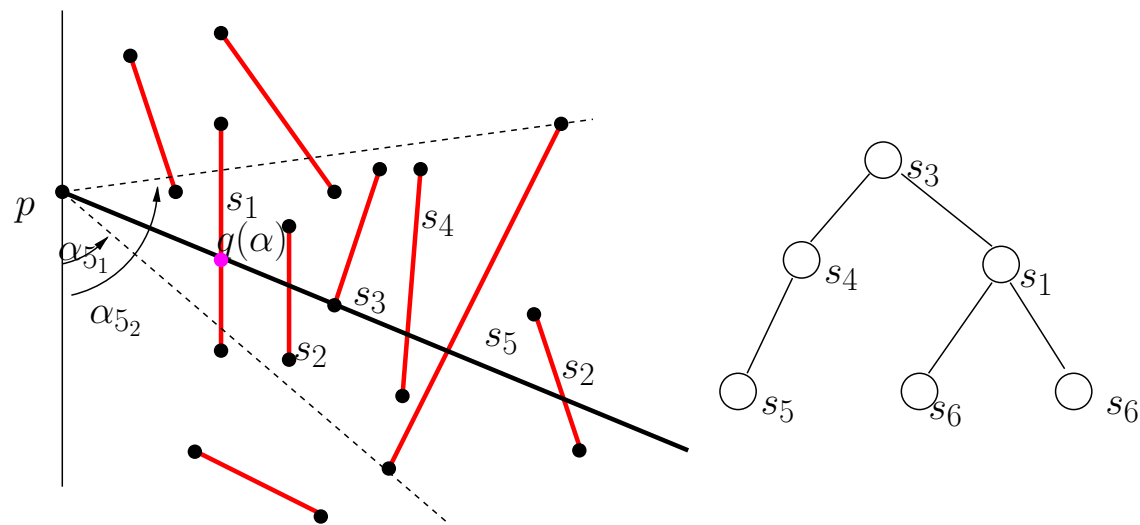


# Single-Source Sweep SSS: Balancierter Baum



# Single-Source Sweep SSS: Balancierter Baum

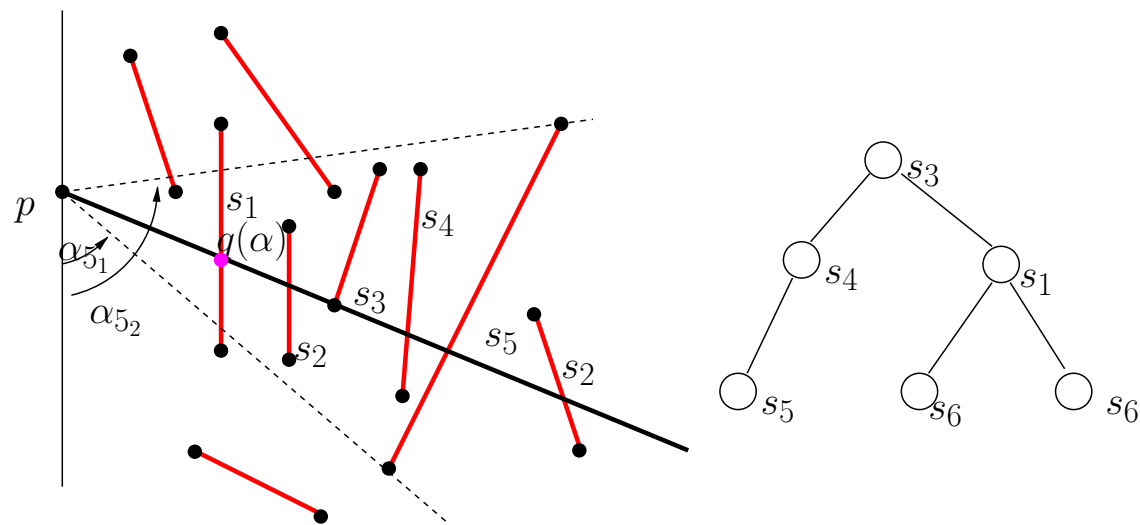
- Dynamische Schlüssel, Liniensegmente





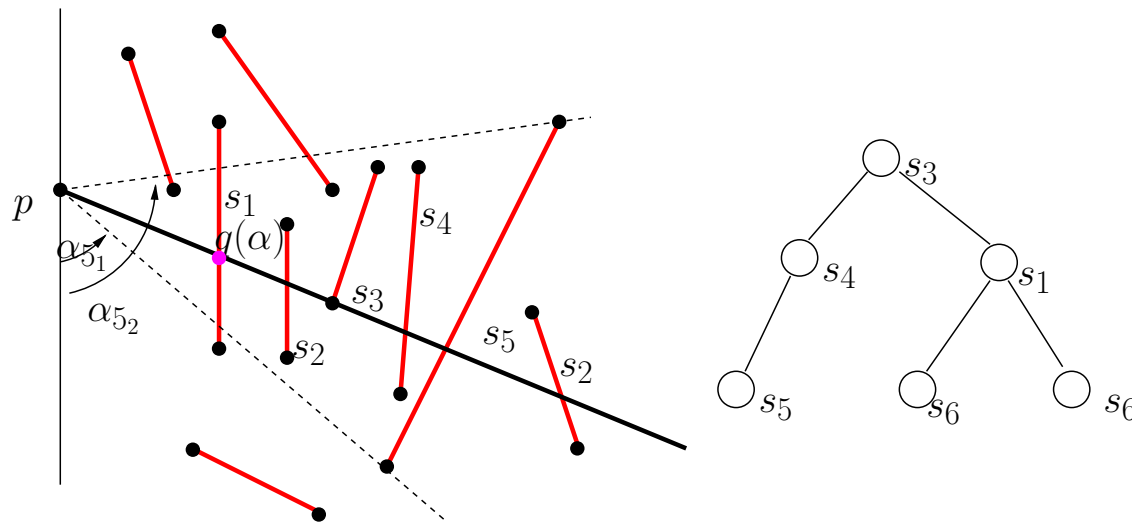
# Single-Source Sweep SSS: Balancierter Baum

- Dynamische Schlüssel, Liniensegmente
- Abhängig vom Winkel  $\alpha$ , Distanz zum Liniensegment



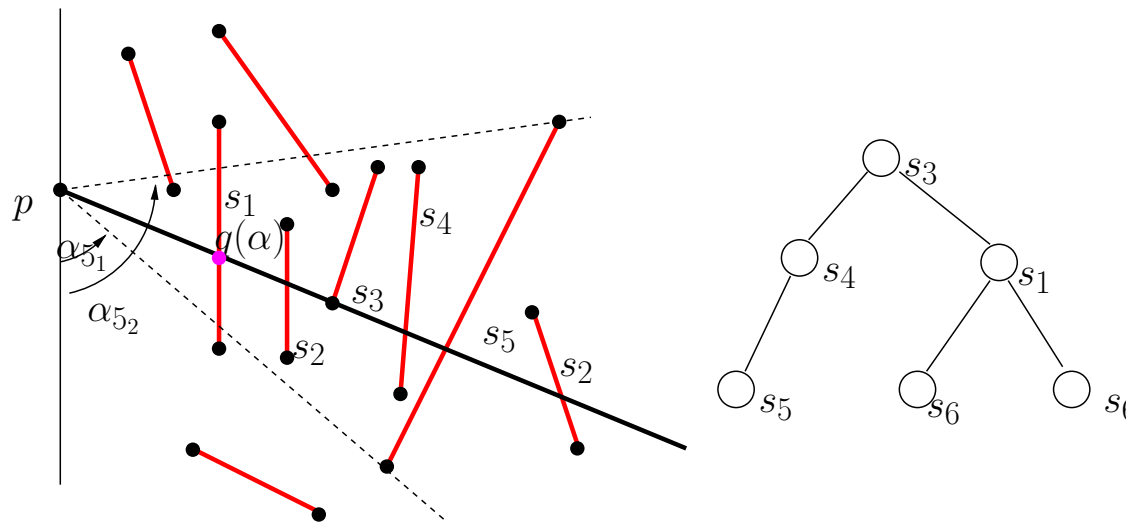
# Single-Source Sweep SSS: Balancierter Baum

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



# Single-Source Sweep SSS: Balancierter Baum

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

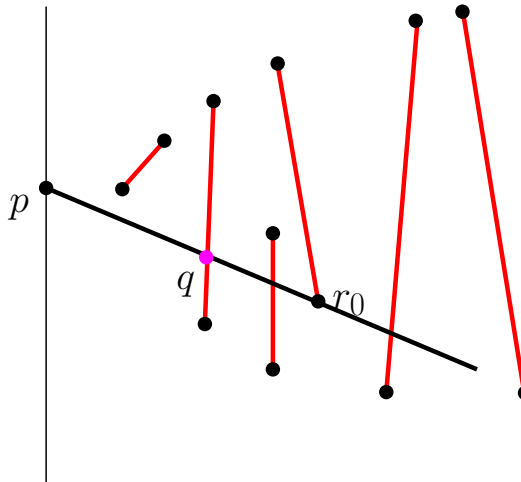


# Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

# Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

Nächster Punkt  $r_i$ : Immer in folgender Reihenfolge

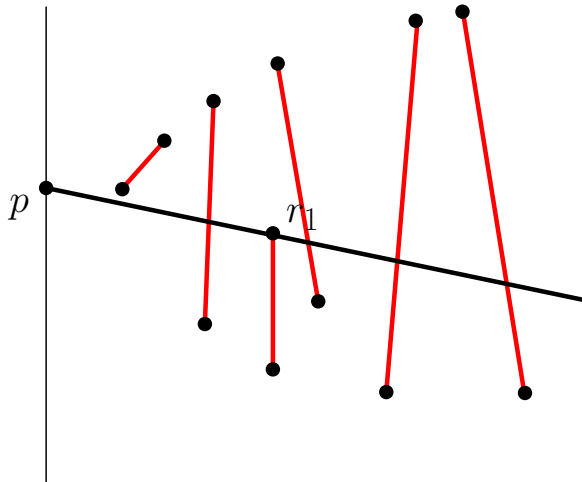
1. Falls  $r_i$  Startpunkt  $\Rightarrow$  Füge Segment ein:  $O(\log n)$
2. Falls  $r_i = q(\alpha) \Rightarrow$  Ausgabe  $(p, r_i)$ :  $O(1)$
3. Falls  $r_i$  Endpunkt  $\Rightarrow$  Lösche Segment:  $O(\log n)$



# Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

Nächster Punkt  $r_i$ : Immer in folgender Reihenfolge

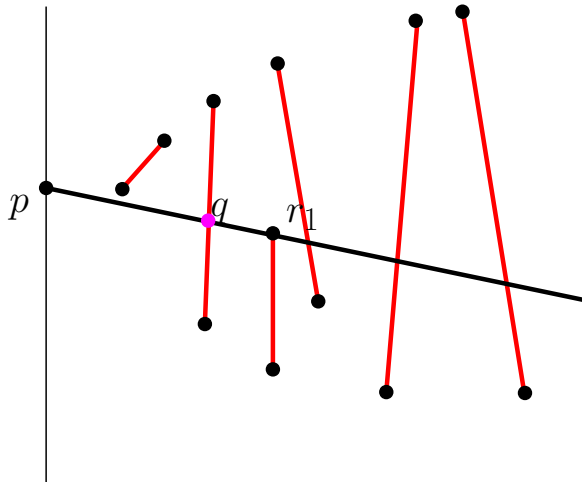
1. Falls  $r_i$  Startpunkt  $\Rightarrow$  Füge Segment ein:  $O(\log n)$
2. Falls  $r_i = q(\alpha) \Rightarrow$  Ausgabe  $(p, r_i)$ :  $O(1)$
3. Falls  $r_i$  Endpunkt  $\Rightarrow$  Lösche Segment:  $O(\log n)$



# Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

Nächster Punkt  $r_i$ : Immer in folgender Reihenfolge

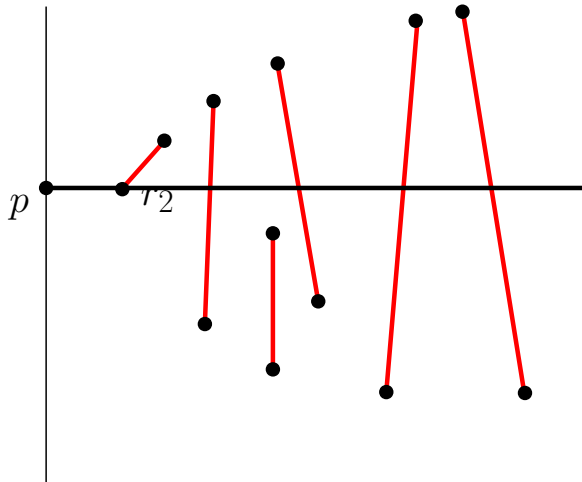
1. Falls  $r_i$  Startpunkt  $\Rightarrow$  Füge Segment ein:  $O(\log n)$
2. Falls  $r_i = q(\alpha) \Rightarrow$  Ausgabe  $(p, r_i)$ :  $O(1)$
3. Falls  $r_i$  Endpunkt  $\Rightarrow$  Lösche Segment:  $O(\log n)$



# 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:  $O(\log n)$

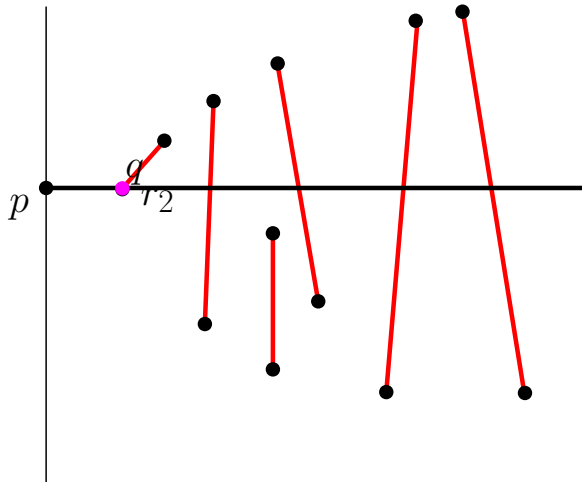




# Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

Nächster Punkt  $r_i$ : Immer in folgender Reihenfolge

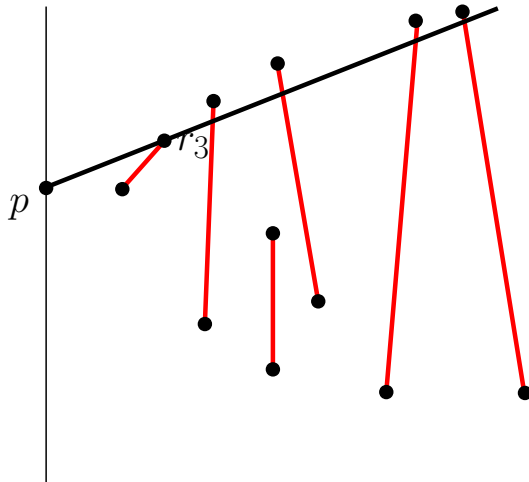
1. Falls  $r_i$  Startpunkt  $\Rightarrow$  Füge Segment ein:  $O(\log n)$
2. Falls  $r_i = q(\alpha) \Rightarrow$  Ausgabe  $(p, r_i)$ :  $O(1)$
3. Falls  $r_i$  Endpunkt  $\Rightarrow$  Lösche Segment:  $O(\log n)$



# Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

Nächster Punkt  $r_i$ : Immer in folgender Reihenfolge

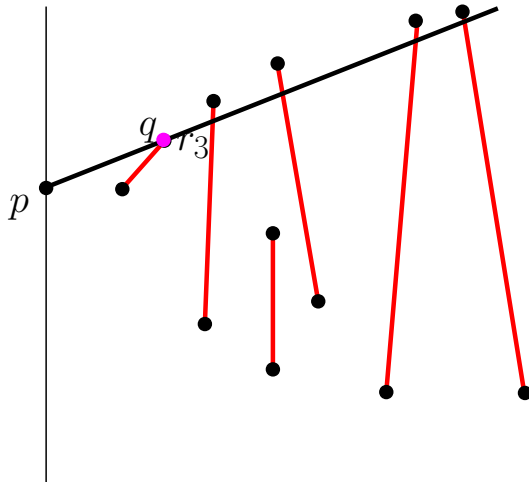
1. Falls  $r_i$  Startpunkt  $\Rightarrow$  Füge Segment ein:  $O(\log n)$
2. Falls  $r_i = q(\alpha) \Rightarrow$  Ausgabe  $(p, r_i)$ :  $O(1)$
3. Falls  $r_i$  Endpunkt  $\Rightarrow$  Lösche Segment:  $O(\log n)$



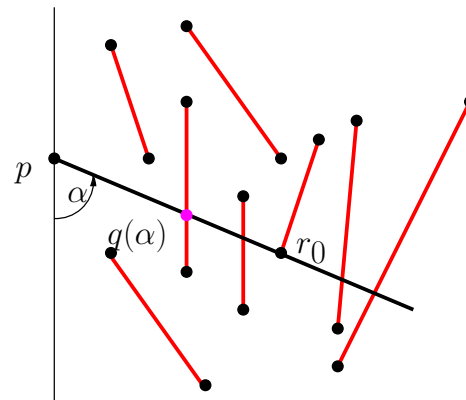
# Alg. 1.2: Single-Source Sweep: Ereignisverarbeitung

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:  $O(\log n)$

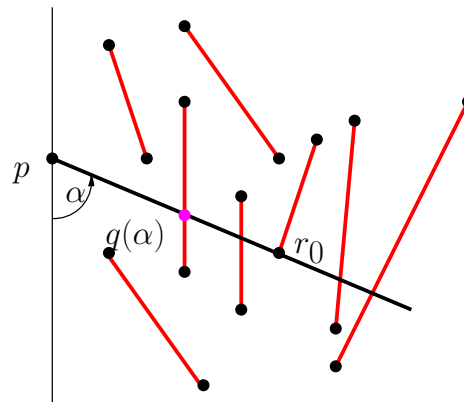


# Single-Source Sweep: Korrektheit und Laufzeit



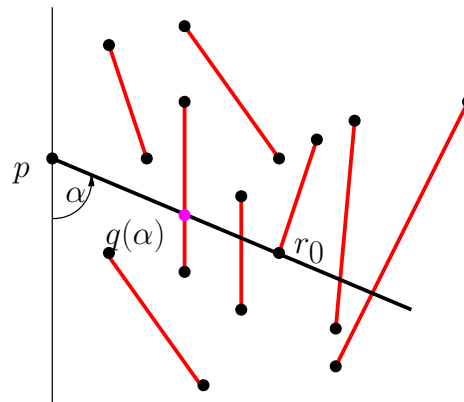
# Single-Source Sweep: Korrektheit und Laufzeit

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



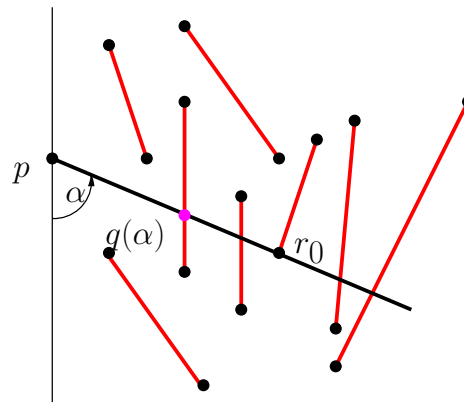
# Single-Source Sweep: Korrektheit und Laufzeit

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



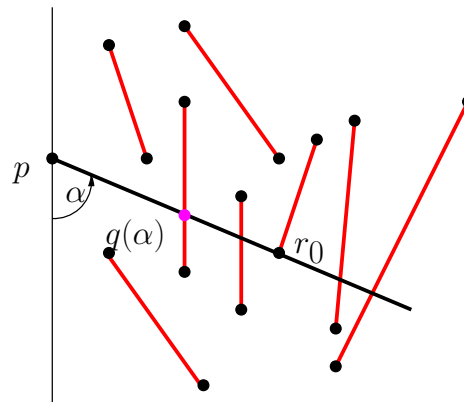
# Single-Source Sweep: Korrektheit und Laufzeit

- 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

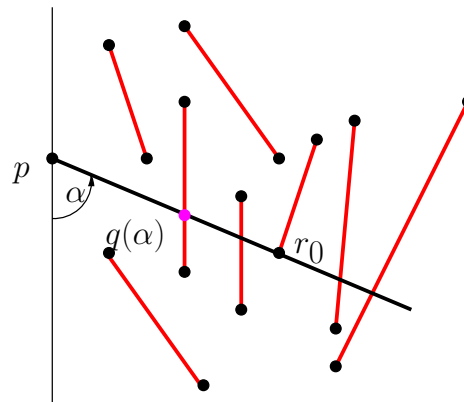
- SSS: Zu jedem Zeitpunkt ist  $q(\alpha)$  sichtbarer Punkt
- Test mit aktuellem Knoten  $r_i$
- Falls  $r_i = q(\alpha) \Rightarrow$  Ausgabe  $(p, r_i)$
- Änderung des bal. Baumes  $O(\log n)$ ,  $n$  mal





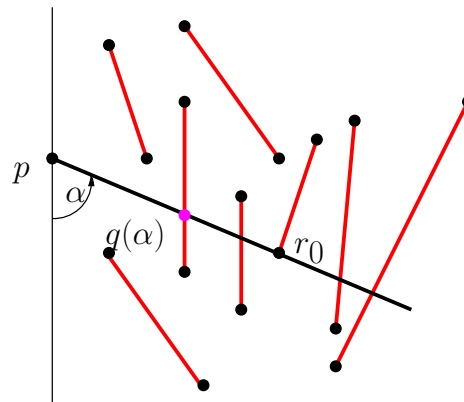
# Single-Source Sweep: Korrektheit und Laufzeit

- 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?

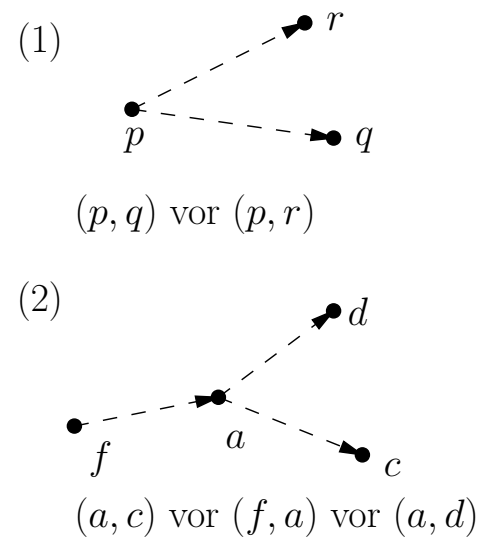
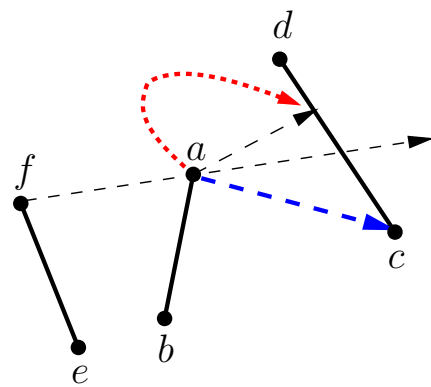
- Laufzeit  $O(n^2 \log n)$ , Komplexität  $\Theta(n^2)$ , Laufzeit  $O(n^2)$ ?

## Geht das besser?

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

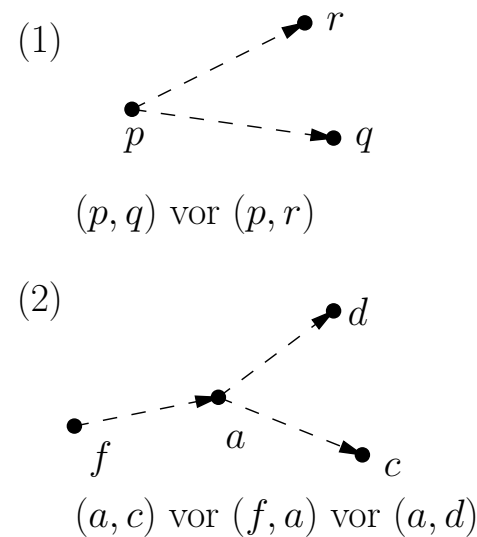
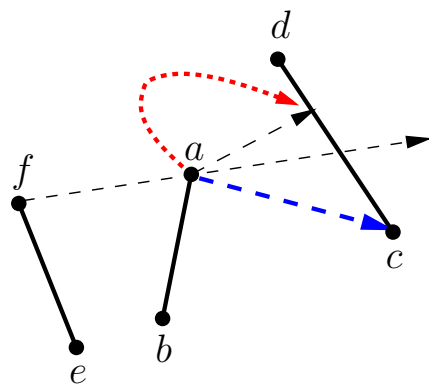
# Geht das besser?

- 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?

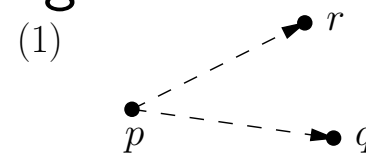
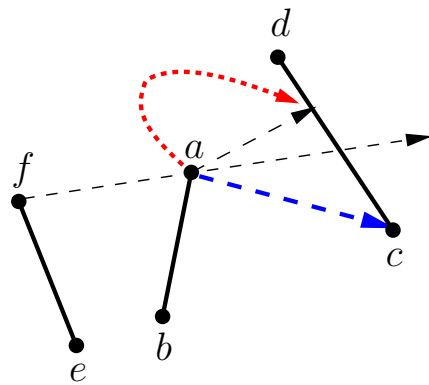
- Laufzeit  $O(n^2 \log n)$ , Komplexität  $\Theta(n^2)$ , Laufzeit  $O(n^2)$ ?
- Simultaner Sweep, alle Segmente gleichzeitig drehen, Sicht-Informationen weiter geben
- Bearbeitungsreihenfolge: Paare von Endpunkten gemäß Steigung



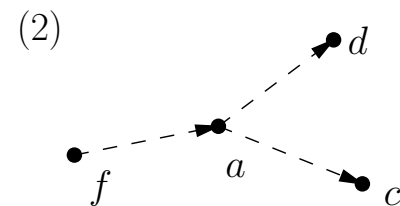


# Geht das besser?

- 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



$(p, q)$  vor  $(p, r)$

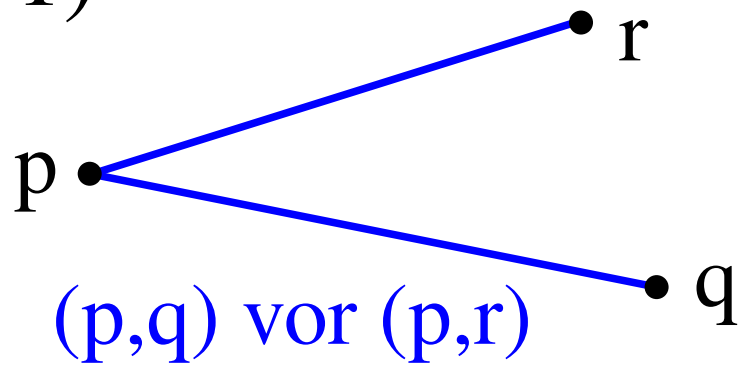


$(a, c)$  vor  $(f, a)$  vor  $(a, d)$

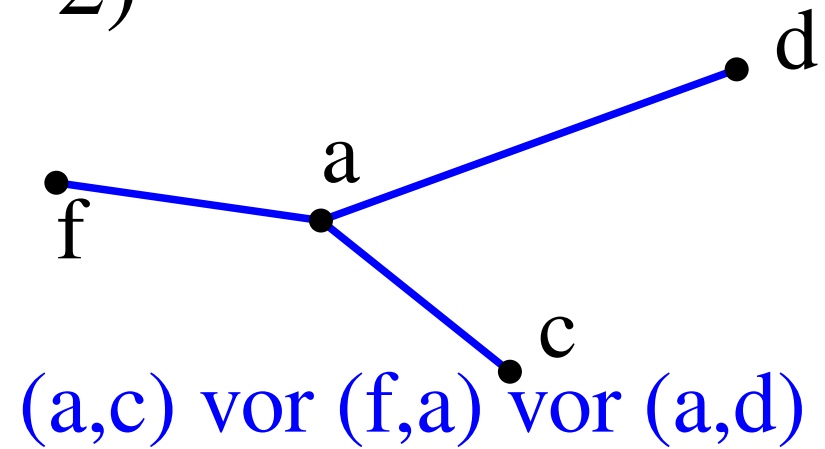
# Bearbeitungsreihenfolge in $O(n^2)$

## Bearbeitungsreihenfolge in $O(n^2)$

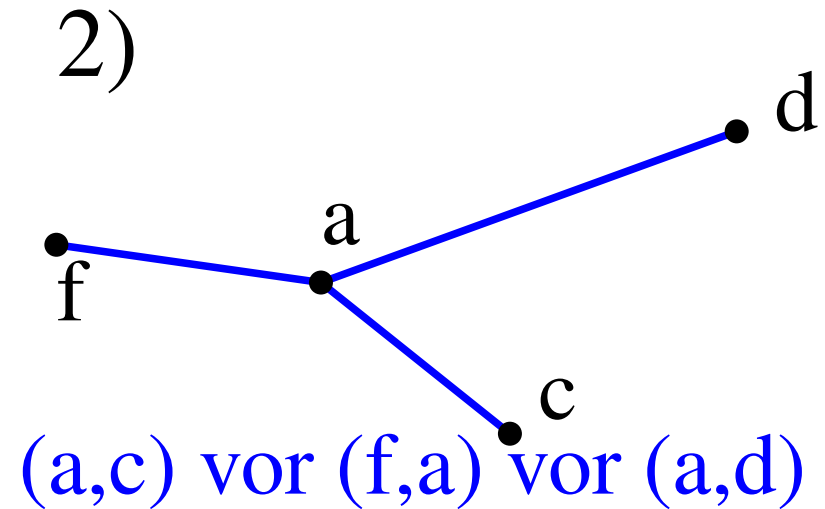
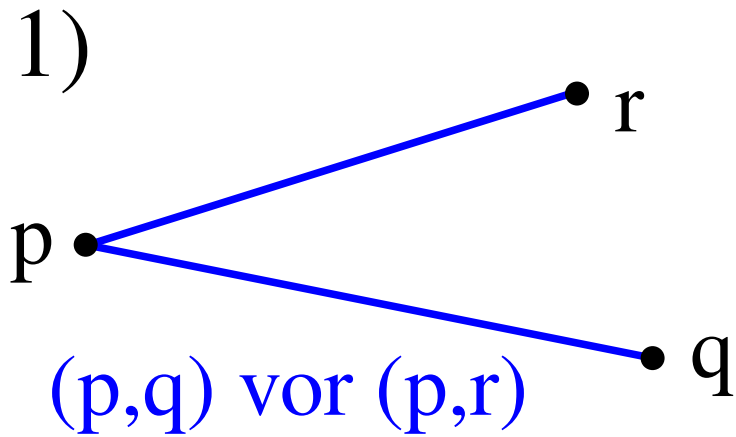
1)



2)

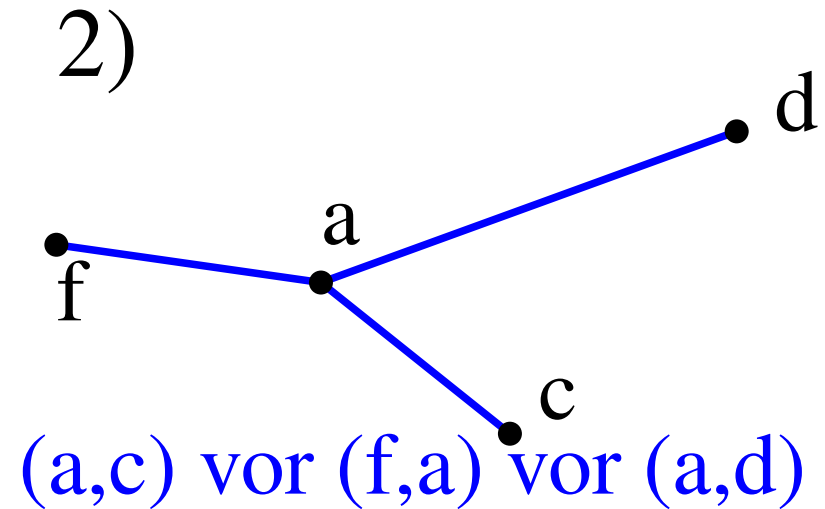
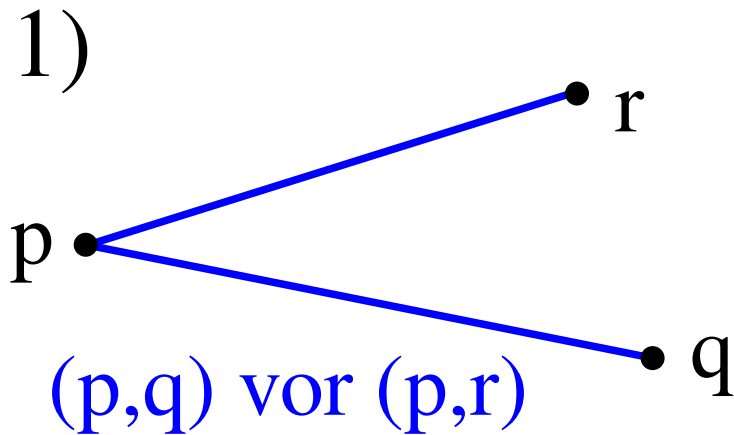


## Bearbeitungsreihenfolge in $O(n^2)$



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

## Bearbeitungsreihenfolge in $O(n^2)$

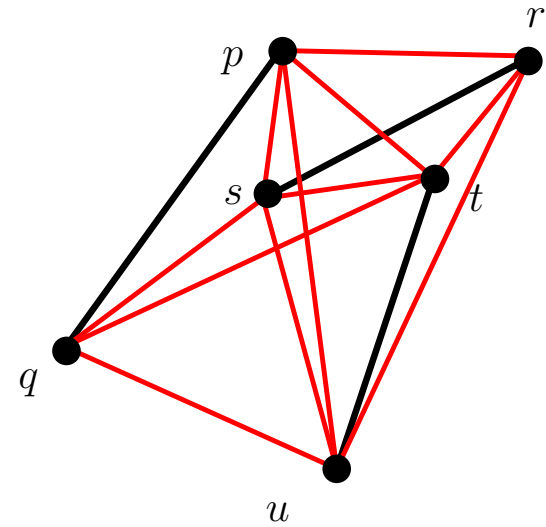
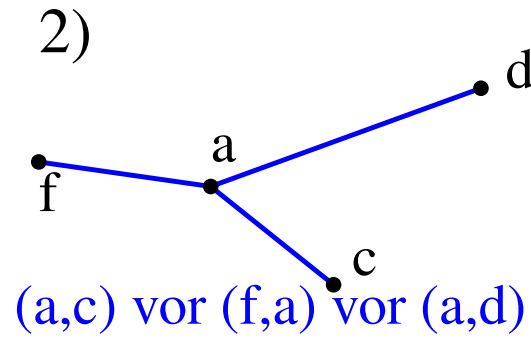
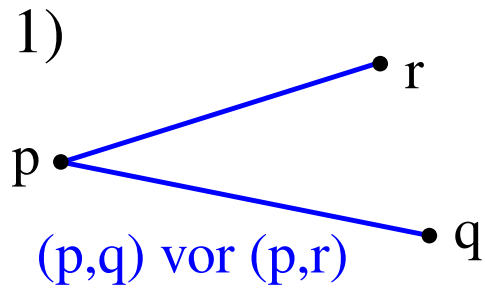


Liste von Punkten  $(p, q)$  mit  $p_x \leq q_x$

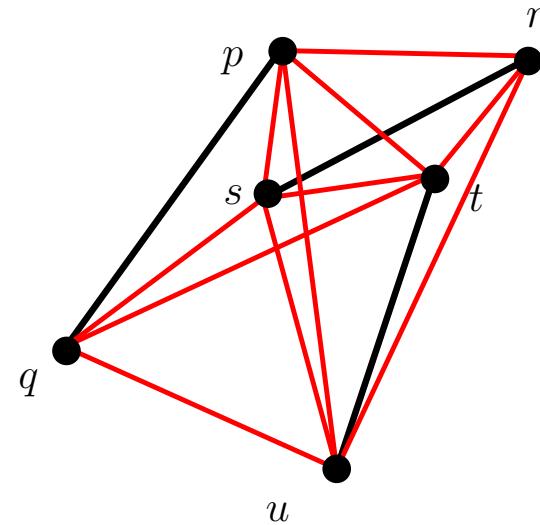
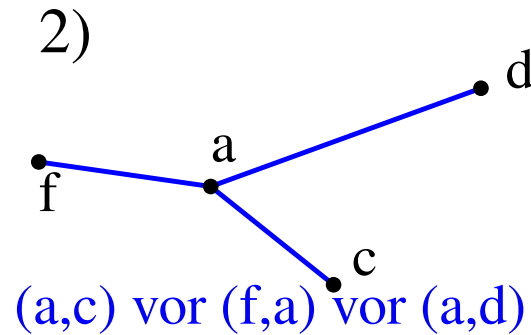
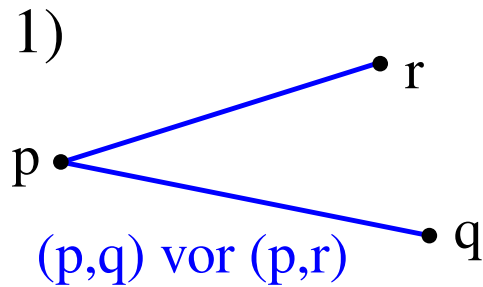
Zeigen wir später! Annahme: Ist schon gegeben für die Punktpaare!

# Beispiel Bearbeitungsreihenfolge

# Beispiel Bearbeitungsreihenfolge



# Beispiel Bearbeitungsreihenfolge

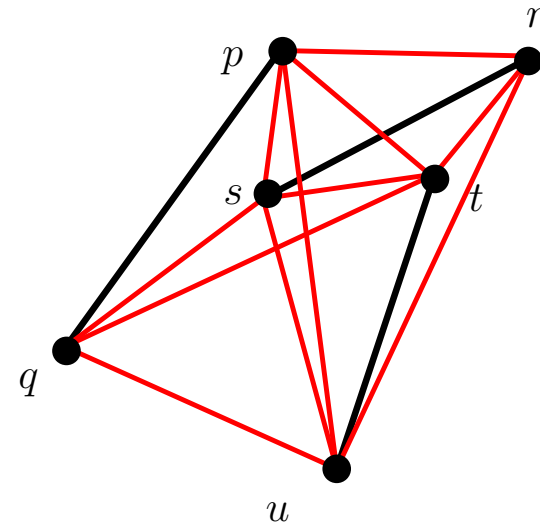
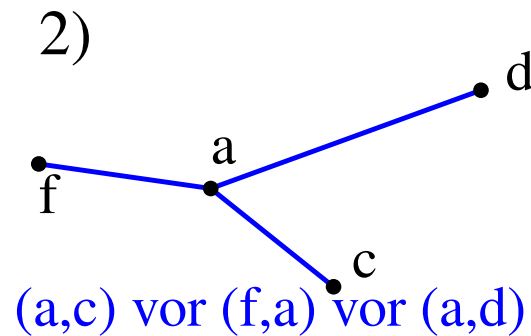
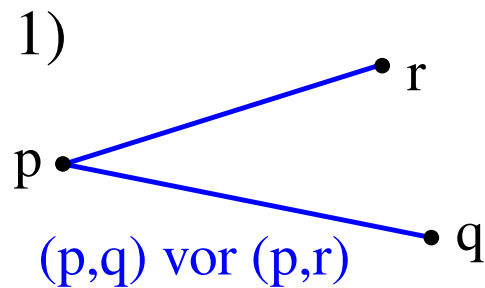


Volle Ordnung:

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



# 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 Ordnung:

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