Offline Bewegungsplanung: Der simultane Sweep

Elmar Langetepe
University of Bonn

Geht das besser?

Geht das besser?

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

Geht das besser?

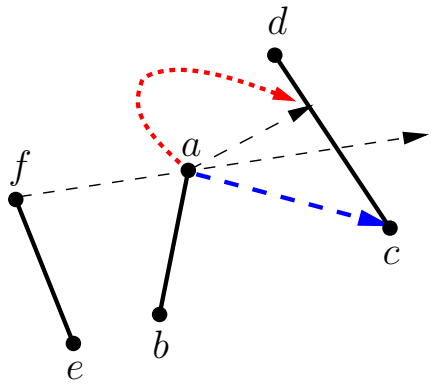
- Laufzeit $O(n^2 \log n)$, Komplexität $\Theta(n^2)$, Laufzeit $O(n^2)$?

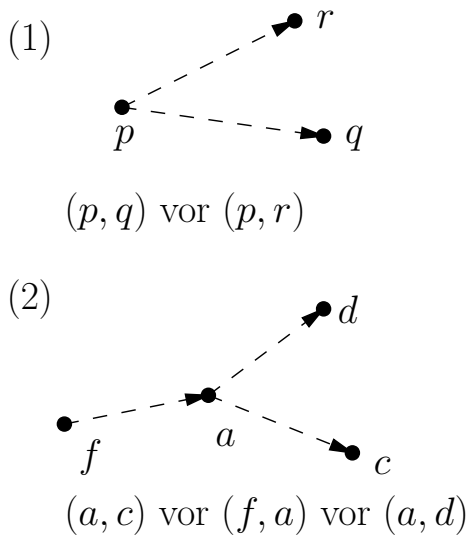
Geht das besser?

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

Geht das besser?

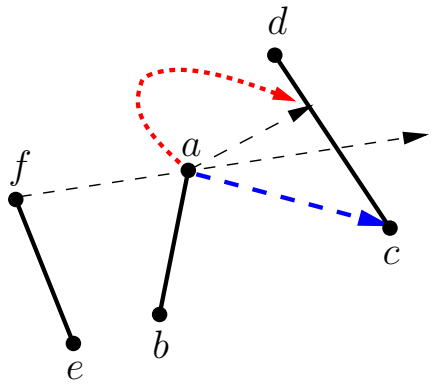
- Laufzeit $O(n^2 \log n)$, Komplexität $\Theta(n^2)$, Laufzeit $O(n^2)$?
- Simultaner Sweep, alle Segmente gleichzeitig drehen, Sicht-Informationen weiter geben

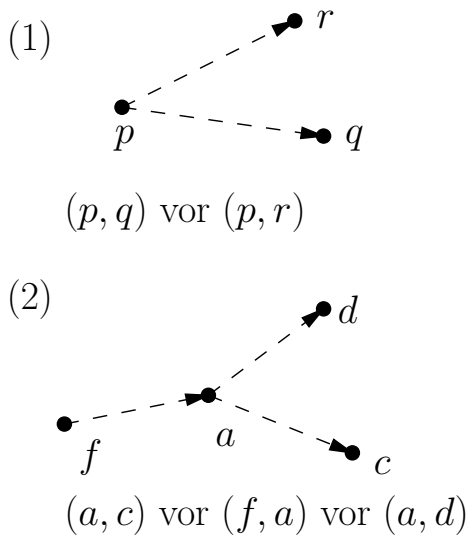




Geht das besser?

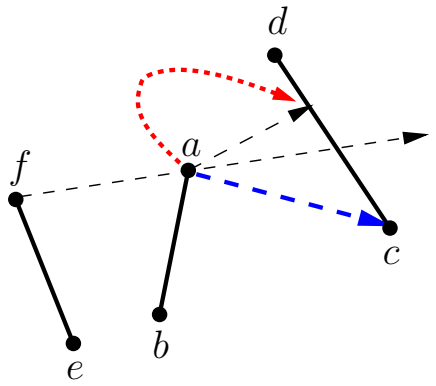
- Laufzeit $O(n^2 \log n)$, Komplexität $\Theta(n^2)$, Laufzeit $O(n^2)$?
- Simultaner Sweep, alle Segmente gleichzeitig drehen, Sicht-Informationen weiter geben
- Bearbeitungsreihenfolge: Paare von Endpunkten gemäß Steigung

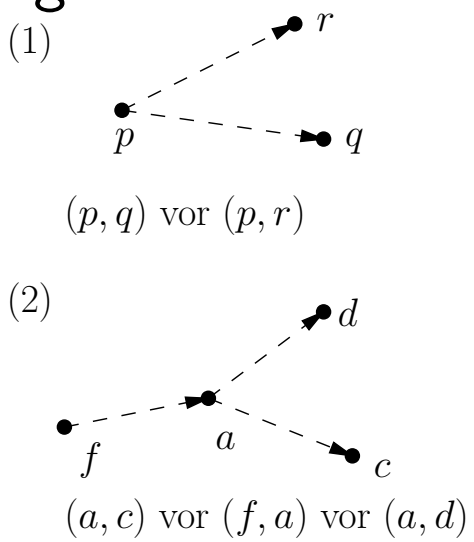




Geht das besser?

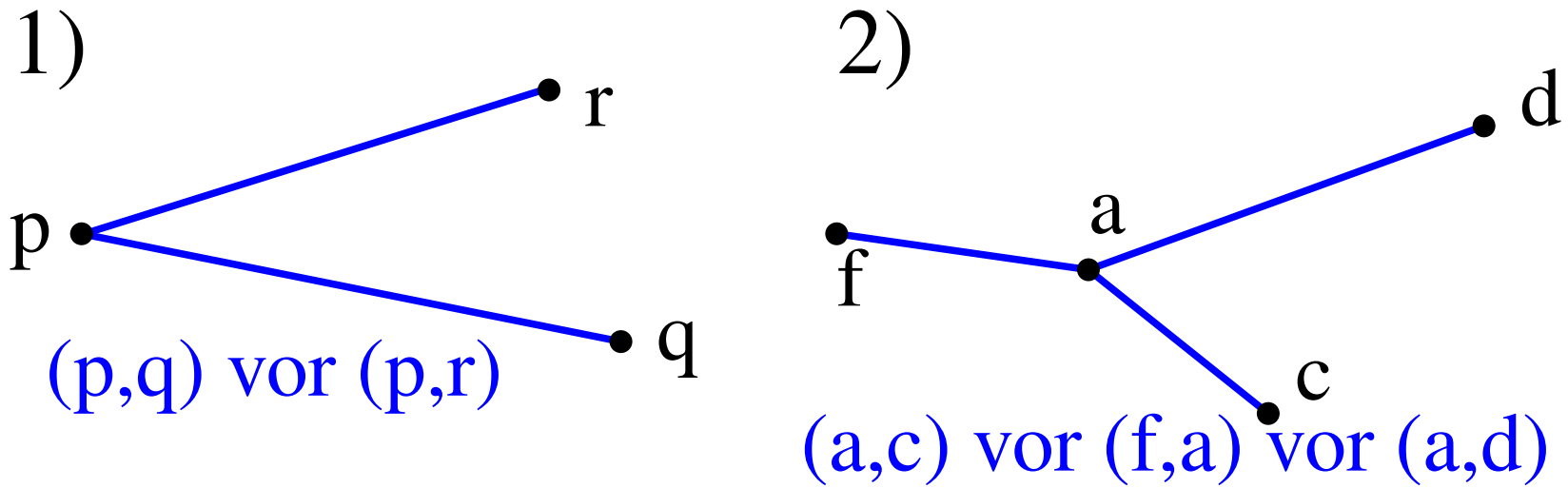
- Laufzeit $O(n^2 \log n)$, Komplexität $\Theta(n^2)$, Laufzeit $O(n^2)$?
- Simultaner Sweep, alle Segmente gleichzeitig drehen, Sicht-Informationen weiter geben
- Bearbeitungsreihenfolge: Paare von Endpunkten gemäß Steigung
- Zeiger auf momentan sichtbares Segment



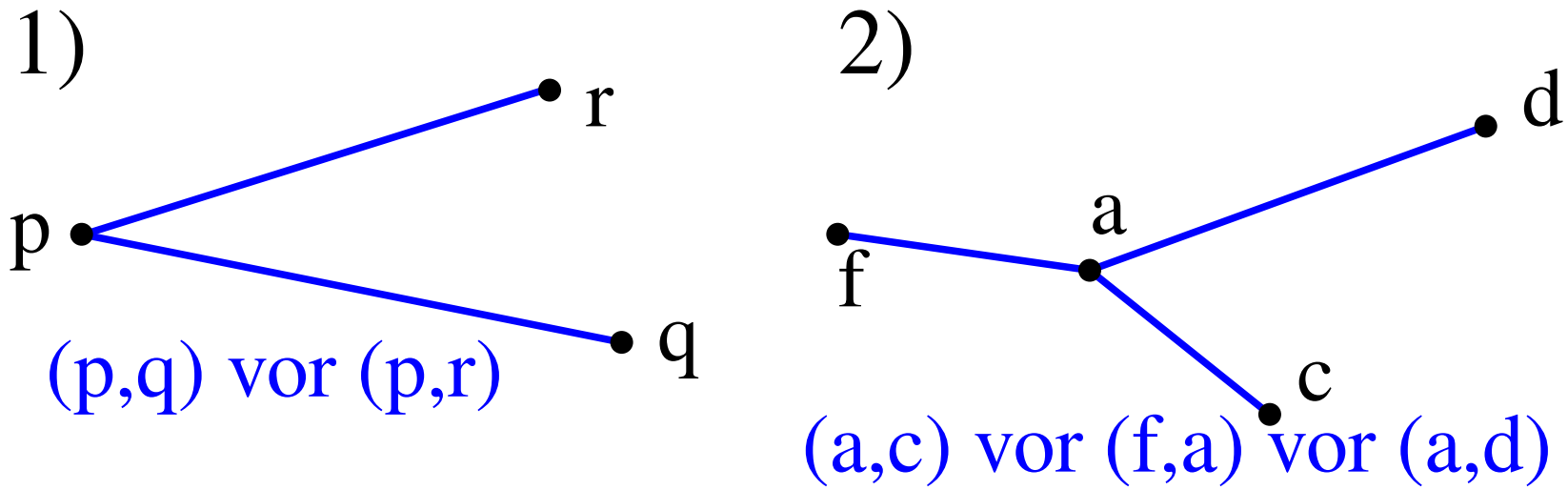


Bearbeitungsreihenfolge in $O(n^2)$

Bearbeitungsreihenfolge in $O(n^2)$

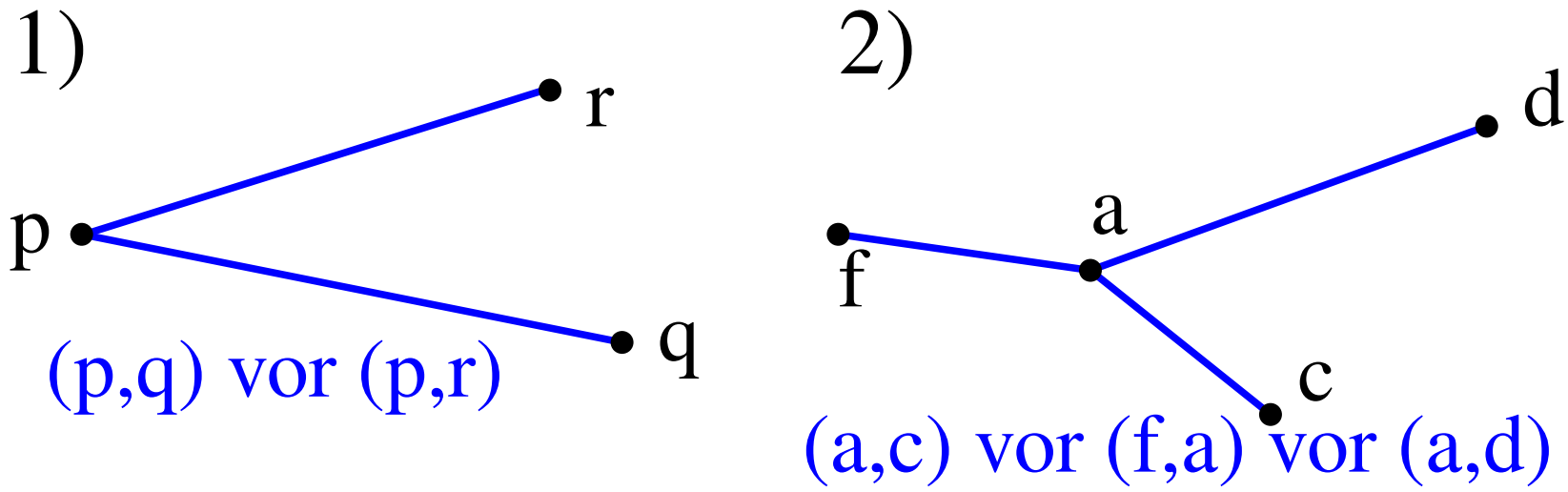


Bearbeitungsreihenfolge in $O(n^2)$



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

Bearbeitungsreihenfolge in $O(n^2)$

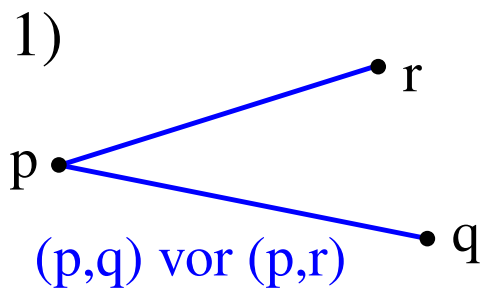


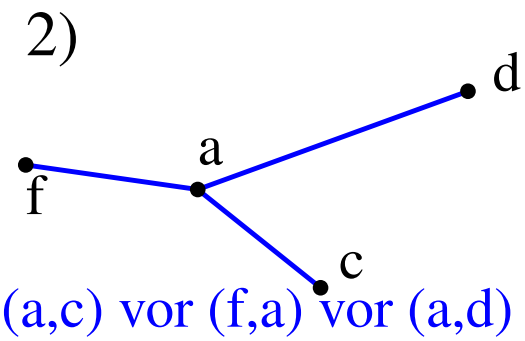
Liste von Punkten (p, q) mit $p_x \leq q_x$

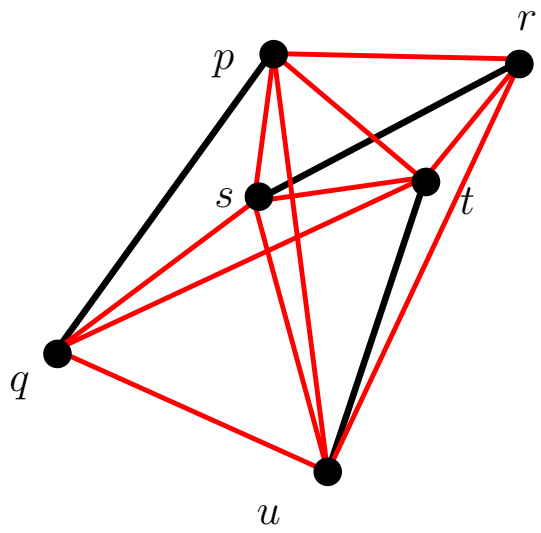
Zeigen wir später! Annahme: Ist schon gegeben für die Punktpaare!

Beispiel Bearbeitungsreihenfolge

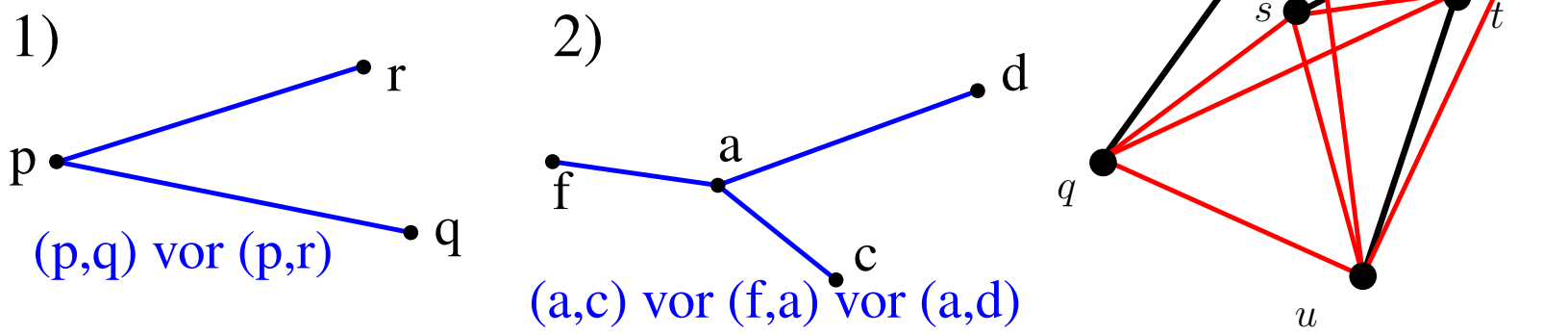
Beispiel Bearbeitungsreihenfolge







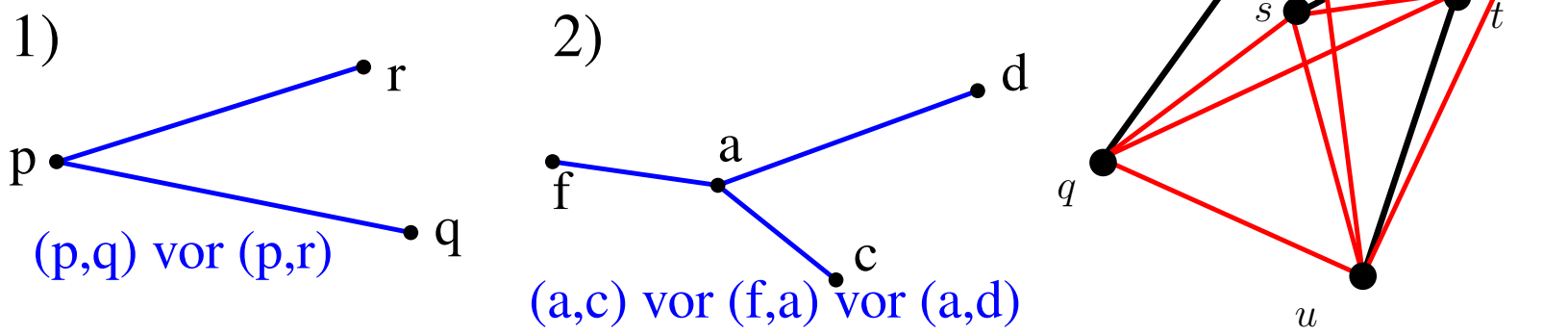
Beispiel Bearbeitungsreihenfolge



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:

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

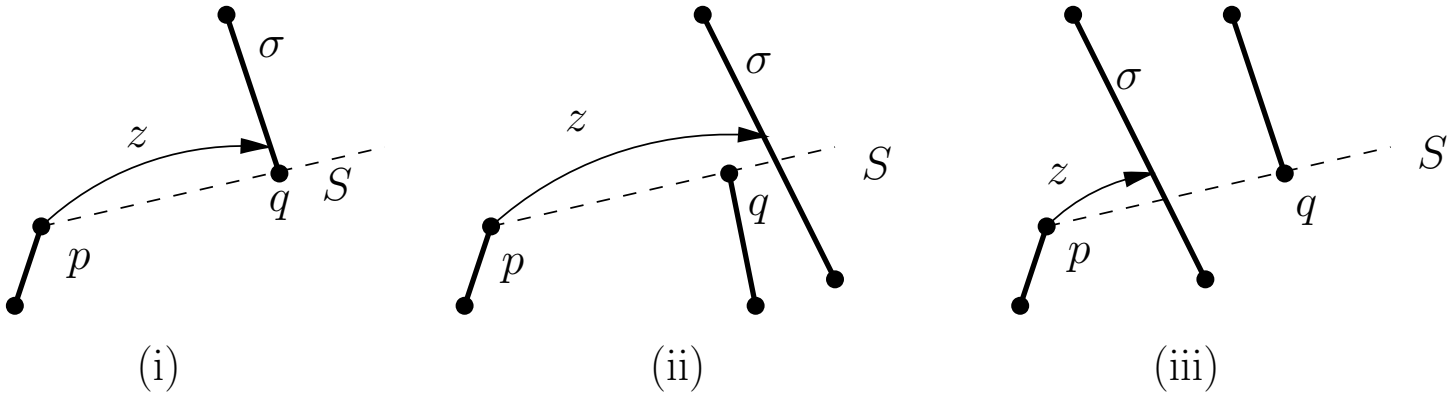
Sweep mit Bearbeitungsreihenfolge

Sweep mit Bearbeitungsreihenfolge

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

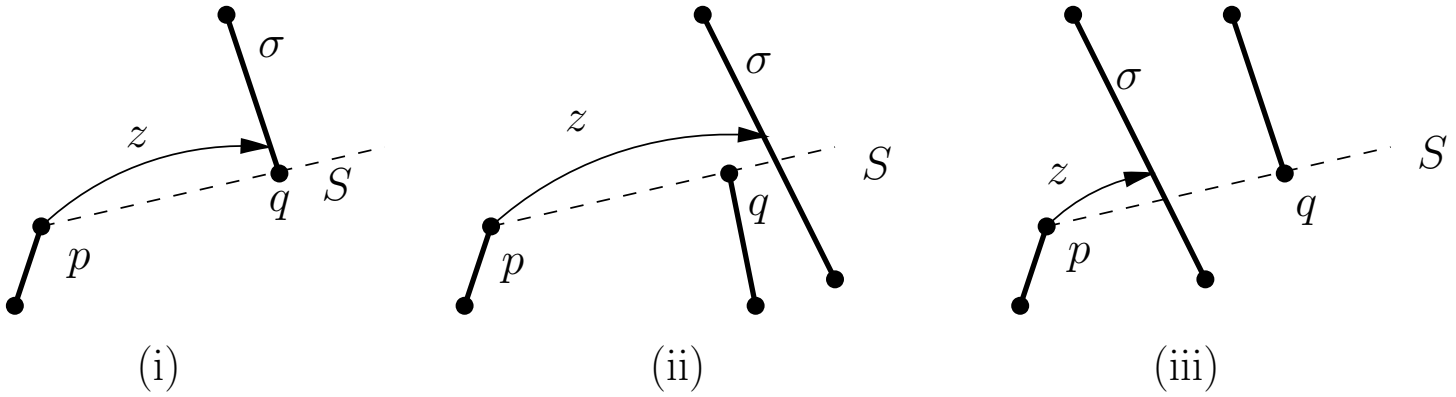
Sweep mit Bearbeitungsreihenfolge

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



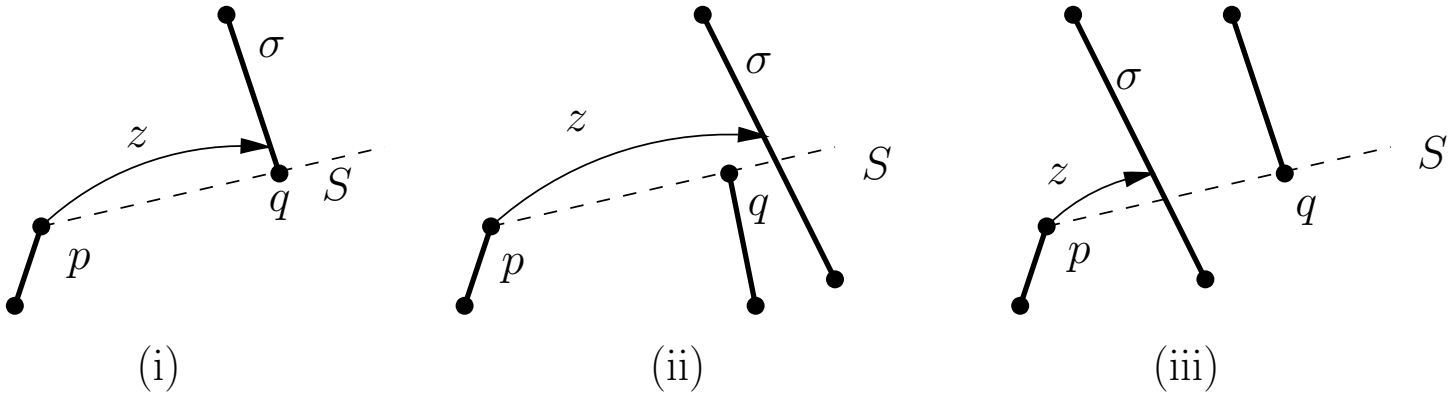
Sweep mit Bearbeitungsreihenfolge

- 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



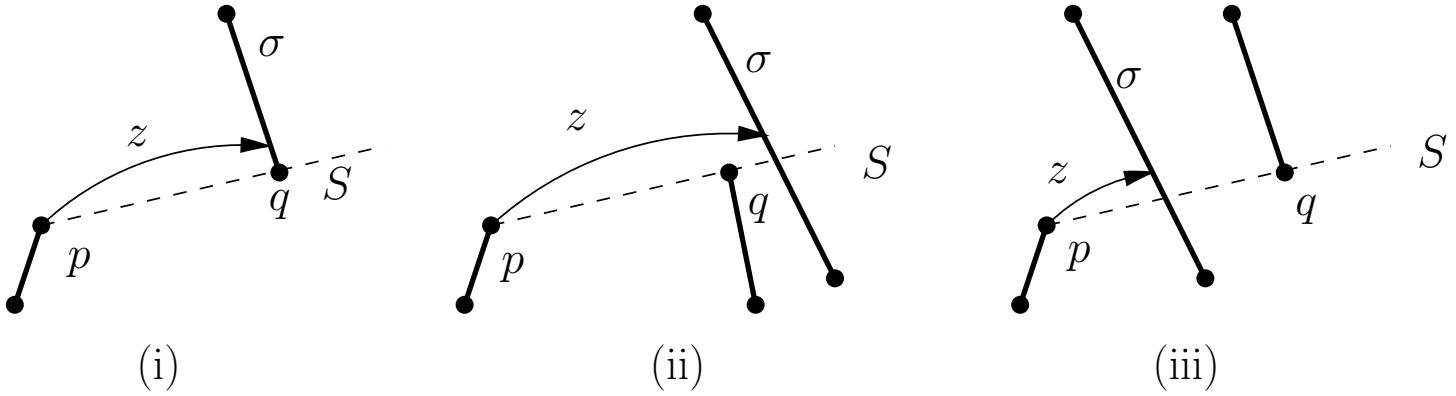
Sweep mit Bearbeitungsreihenfolge

- 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: (i), (ii) und (iii)

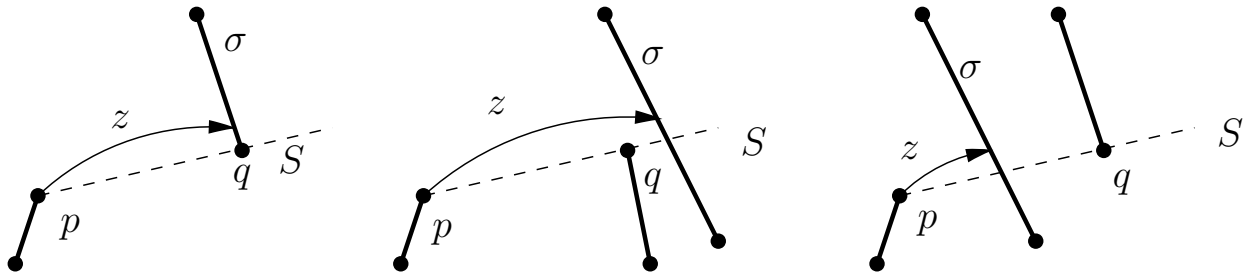


Sweep mit Bearbeitungsreihenfolge

- 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: (i), (ii) und (iii)
- Ausgabe zwischendurch

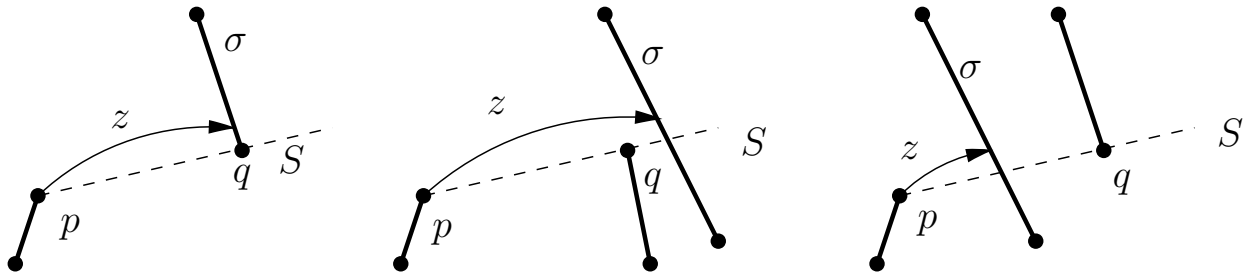


Sweep mit Bearbeitungsreihenfolge



Sweep mit Bearbeitungsreihenfolge

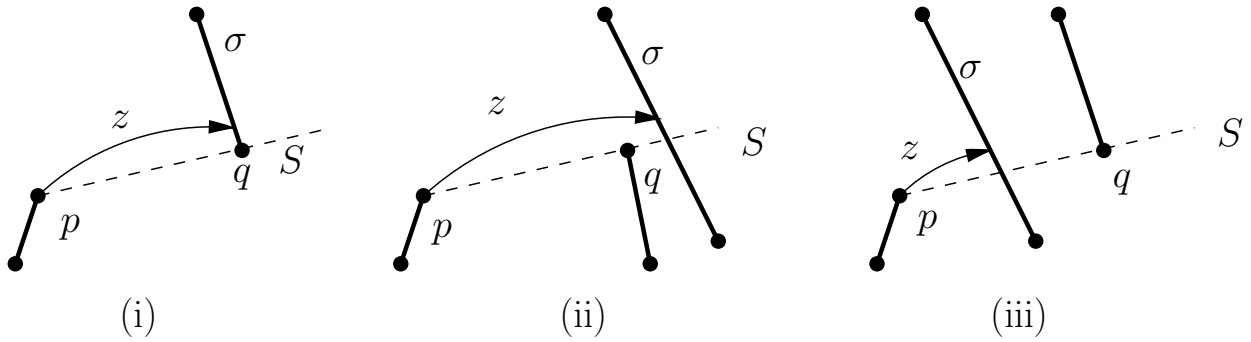
SSS: Für jedes p, (p, q) zuletzt dran:
Zeiger z auf kurz hinter q *sichtbare* Segment σ



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

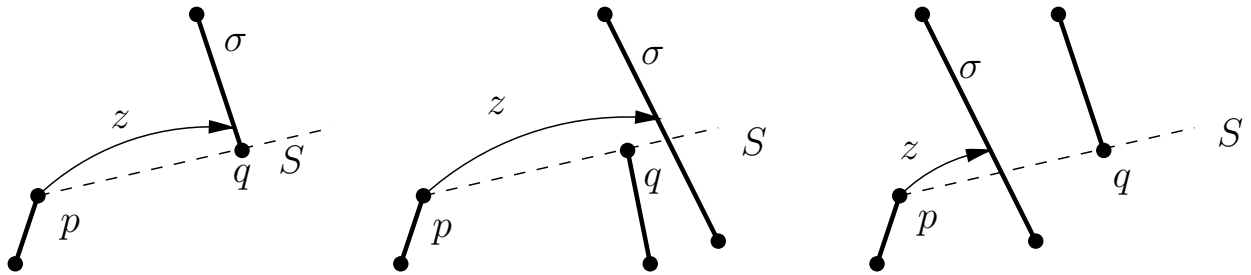


Sweep mit Bearbeitungsreihenfolge

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.



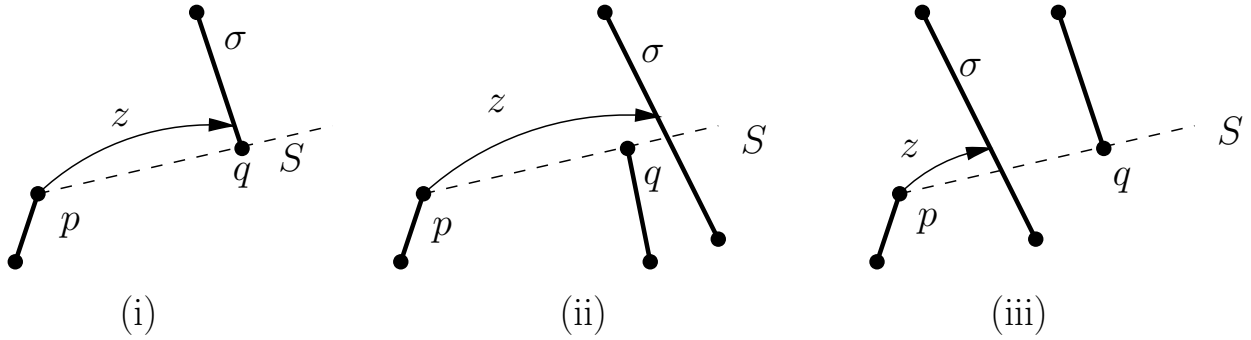
Sweep mit Bearbeitungsreihenfolge

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

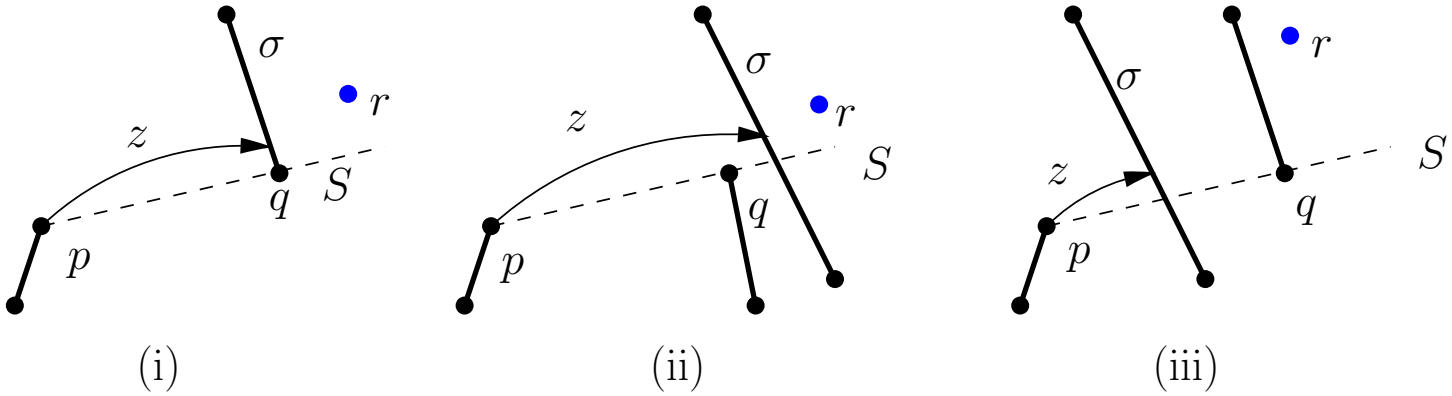
(i) q ist von p aus sichtbar und Anfangspunkt von σ

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

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



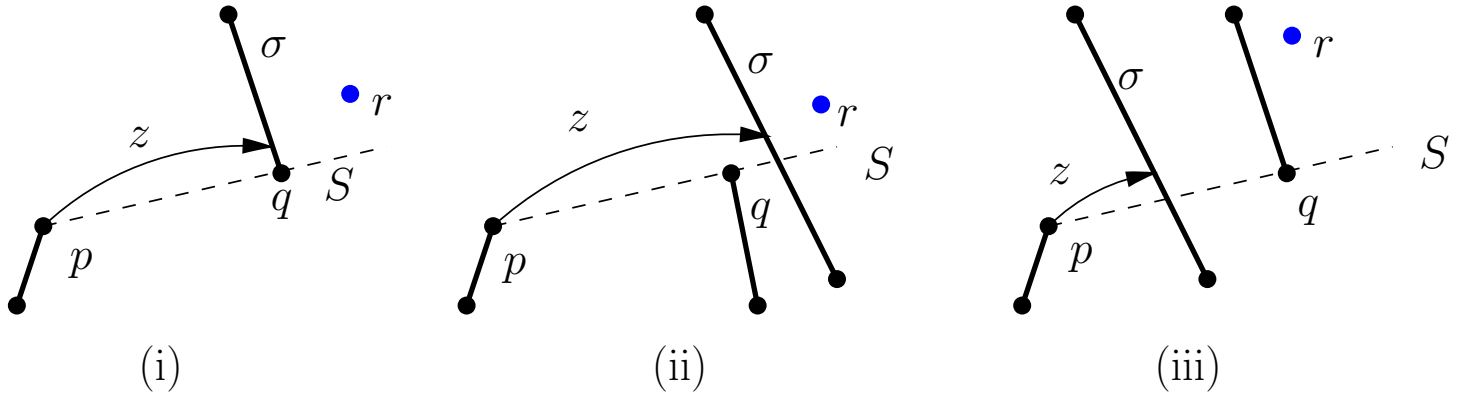
Sweep mit Bearbeitungsreihenfolge



Sweep mit Bearbeitungsreihenfolge

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

F. 1) r liegt hinter σ

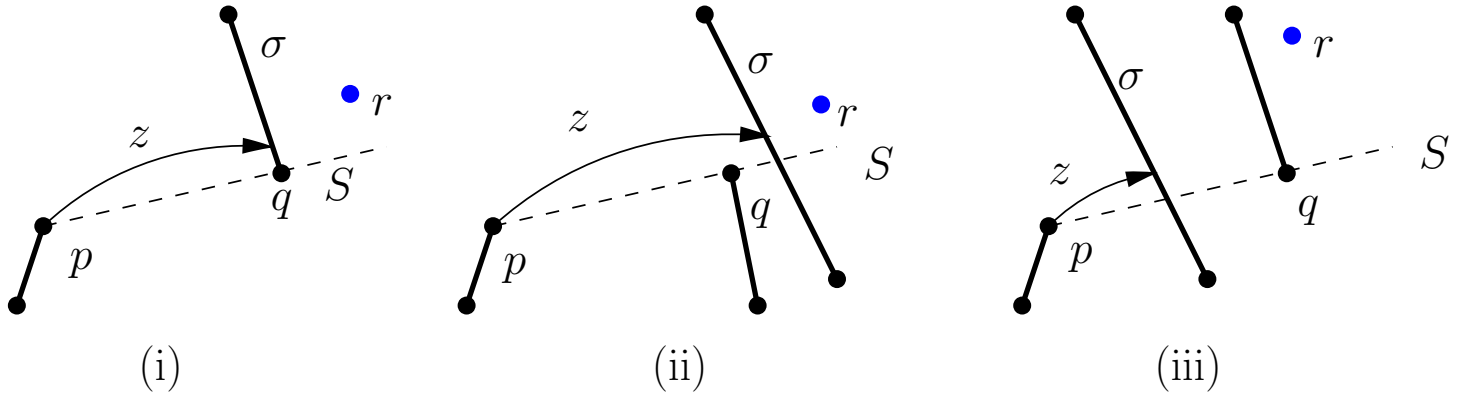


Sweep mit Bearbeitungsreihenfolge

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

F. 1) r liegt hinter σ

- Zeiger bleibt auf σ

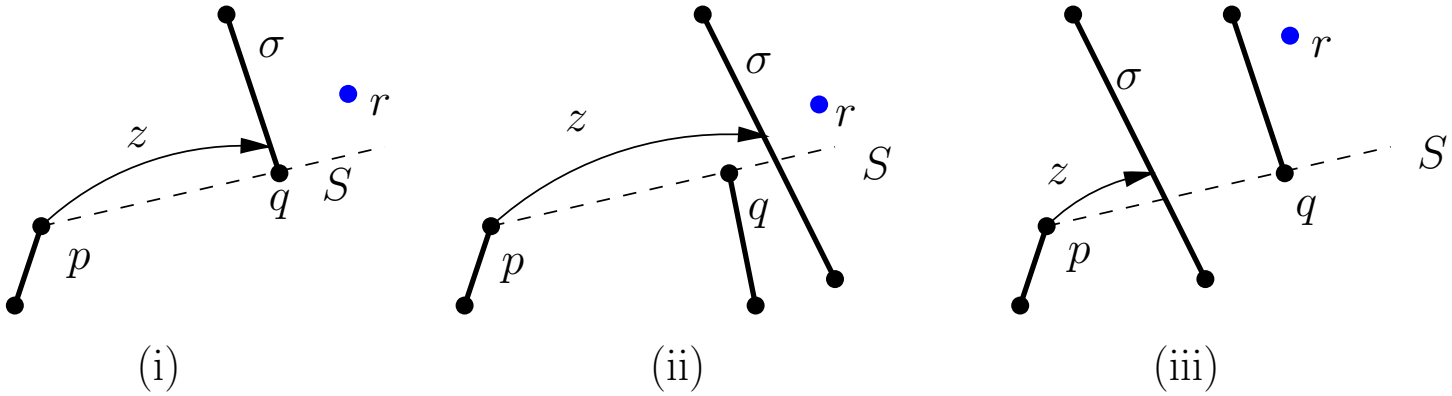


Sweep mit Bearbeitungsreihenfolge

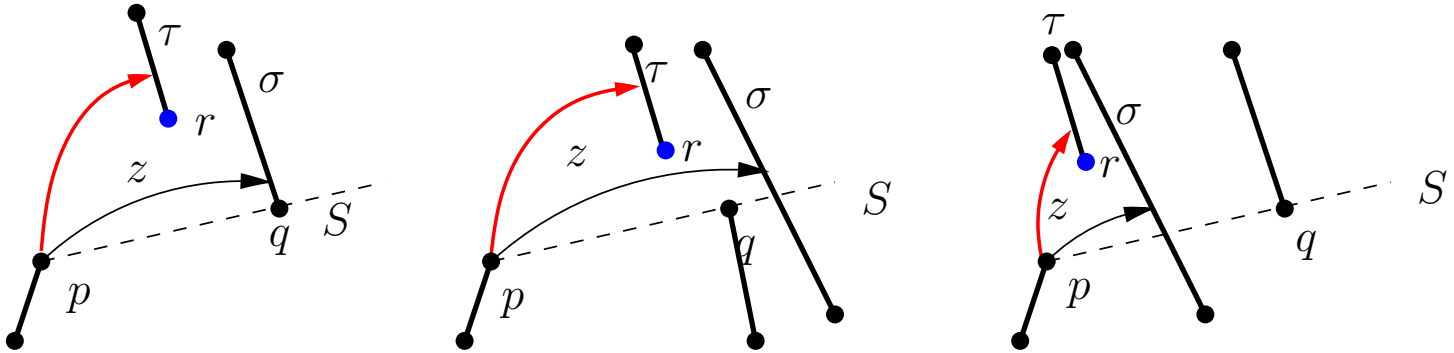
Bearbeitung des nächsten Paares (p, r) nach (p, q)

F. 1) r liegt hinter σ

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



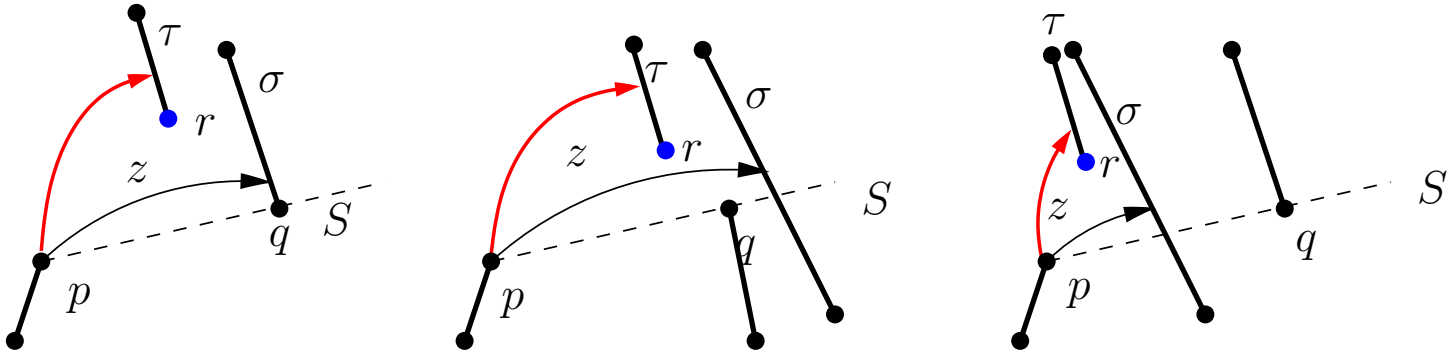
Sweep mit Bearbeitungsreihenfolge



Sweep mit Bearbeitungsreihenfolge

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

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

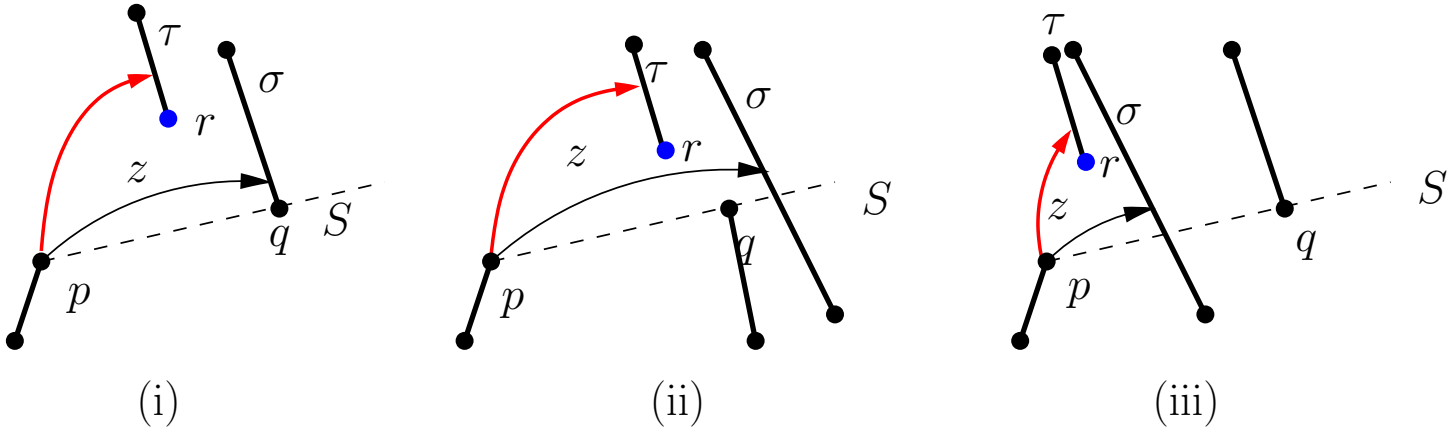


Sweep mit Bearbeitungsreihenfolge

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

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

- Zeiger auf τ setzen

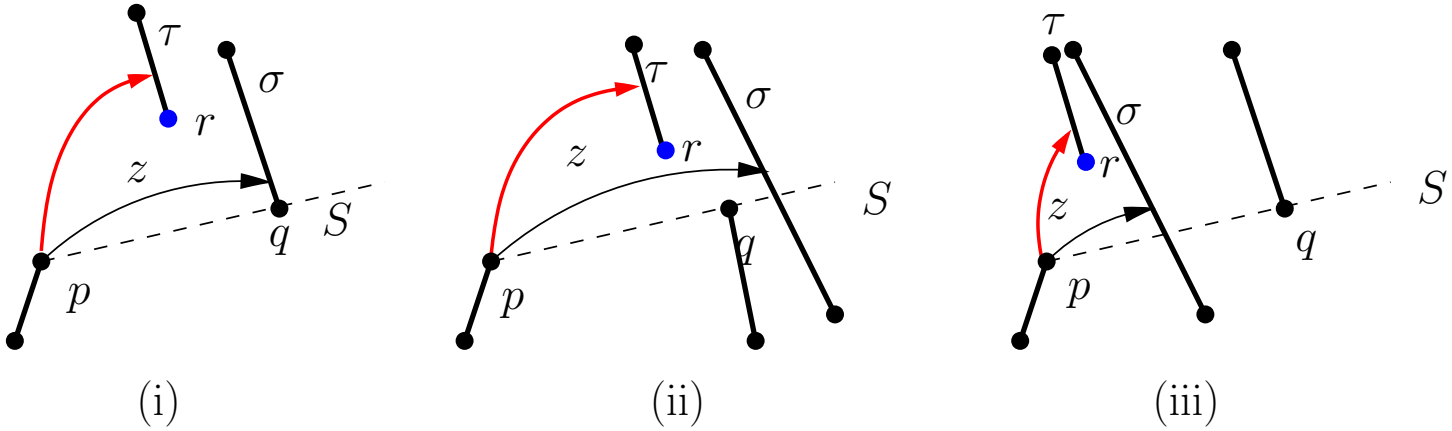


Sweep mit Bearbeitungsreihenfolge

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)!

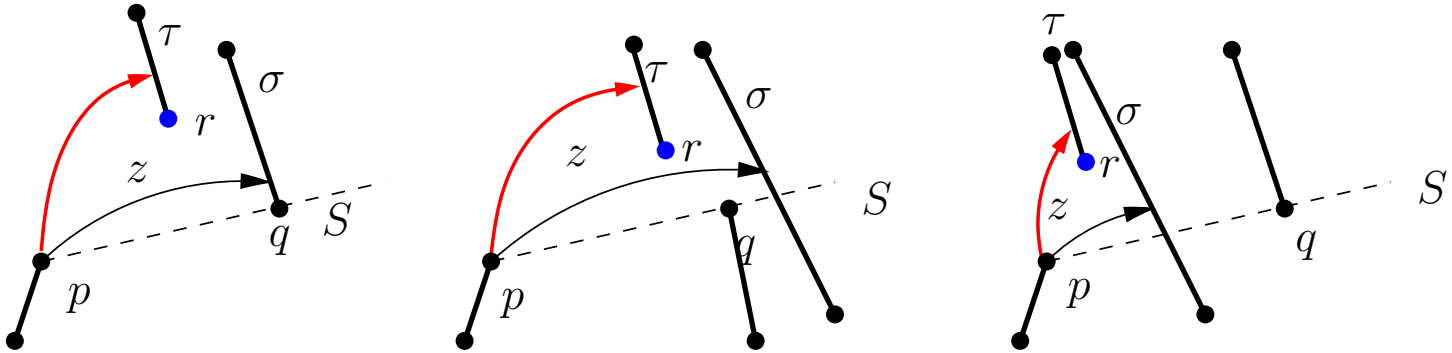


Sweep mit Bearbeitungsreihenfolge

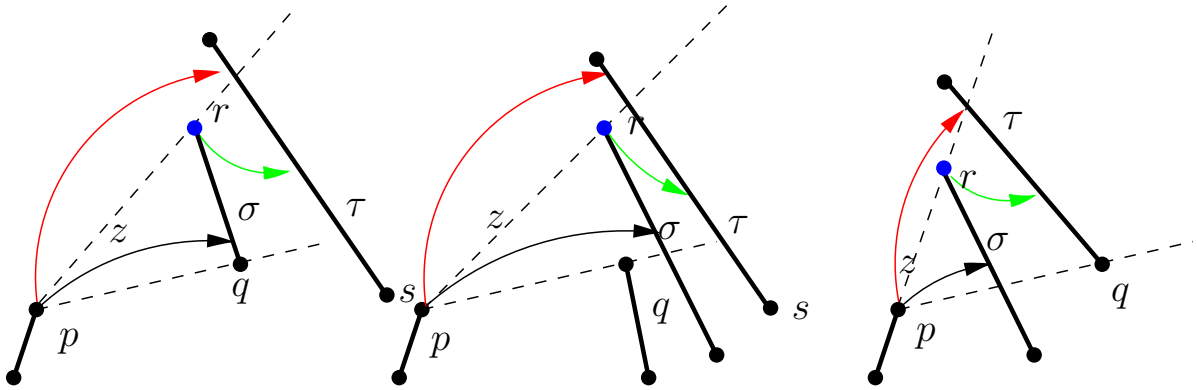
Bearbeitung des nächsten Paares (p, r) nach (p, q)

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

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



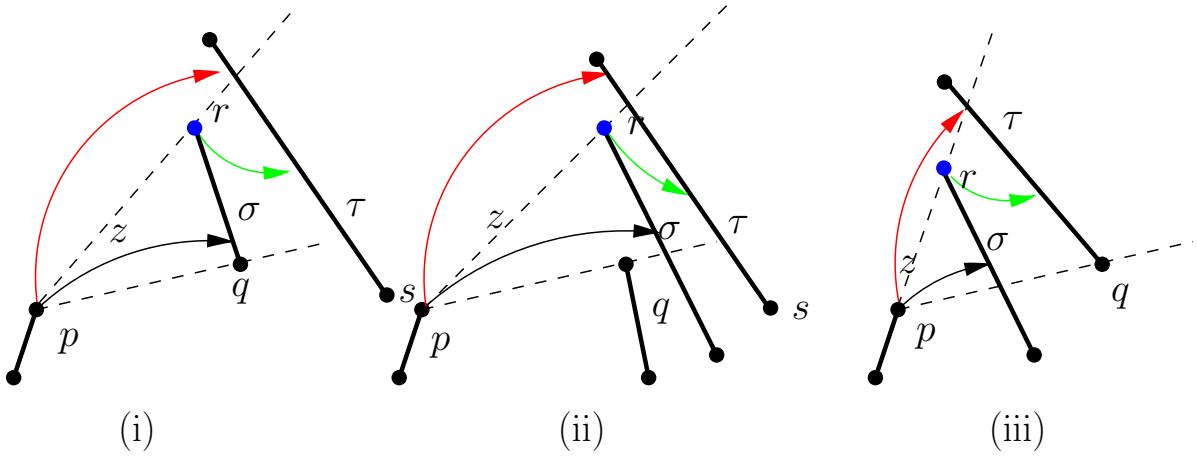
Sweep mit Bearbeitungsreihenfolge



Sweep mit Bearbeitungsreihenfolge

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

F. 3) r ist Endpunkt von σ

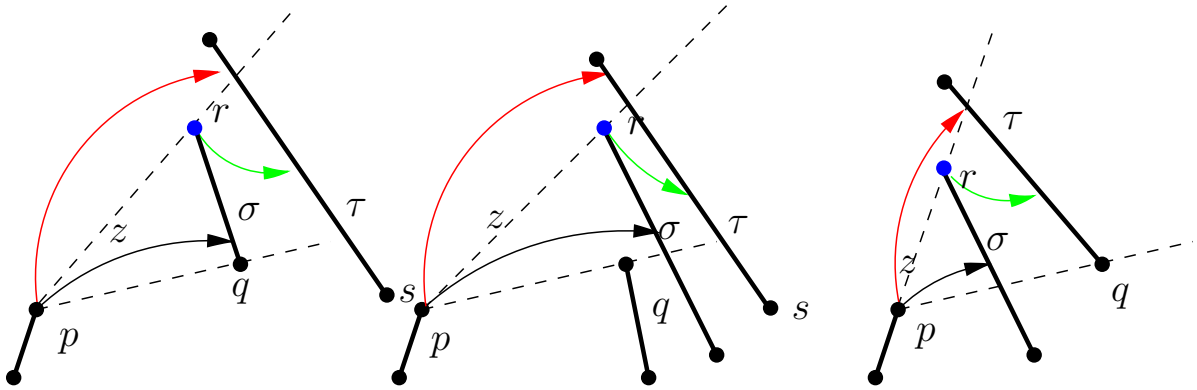


Sweep mit Bearbeitungsreihenfolge

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

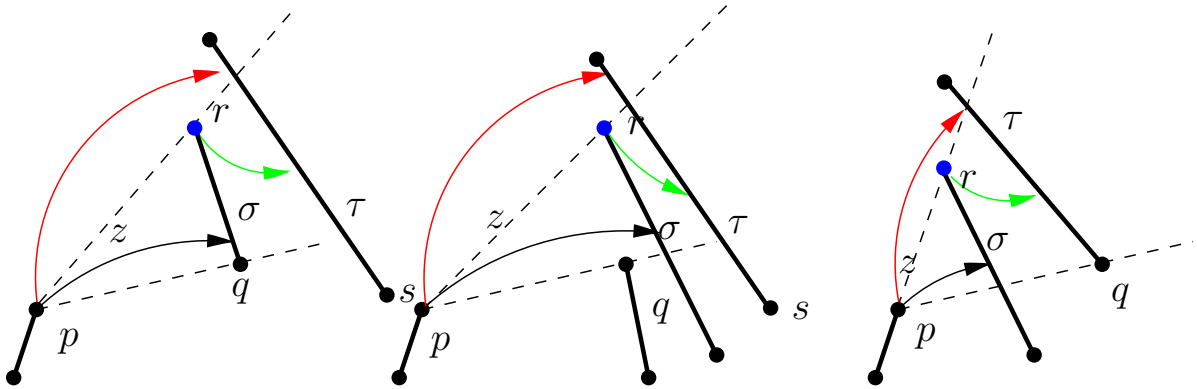


Sweep mit Bearbeitungsreihenfolge

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!!

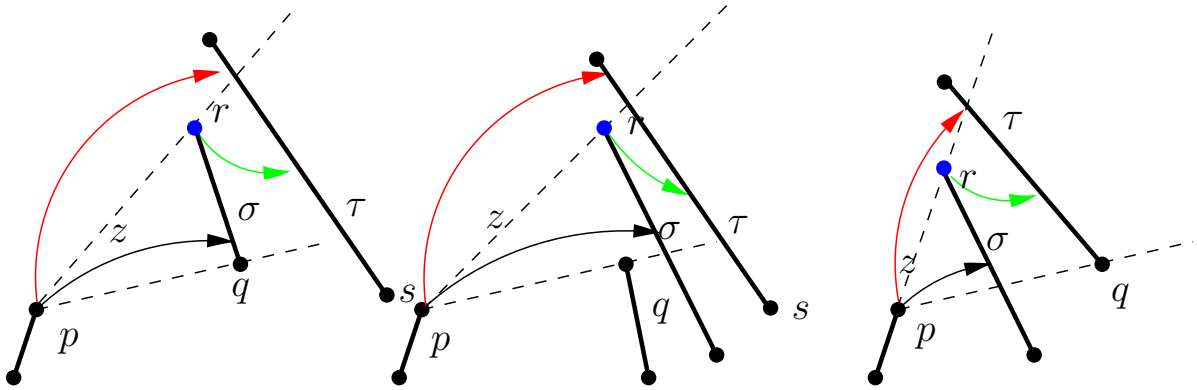


Sweep mit Bearbeitungsreihenfolge

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)!

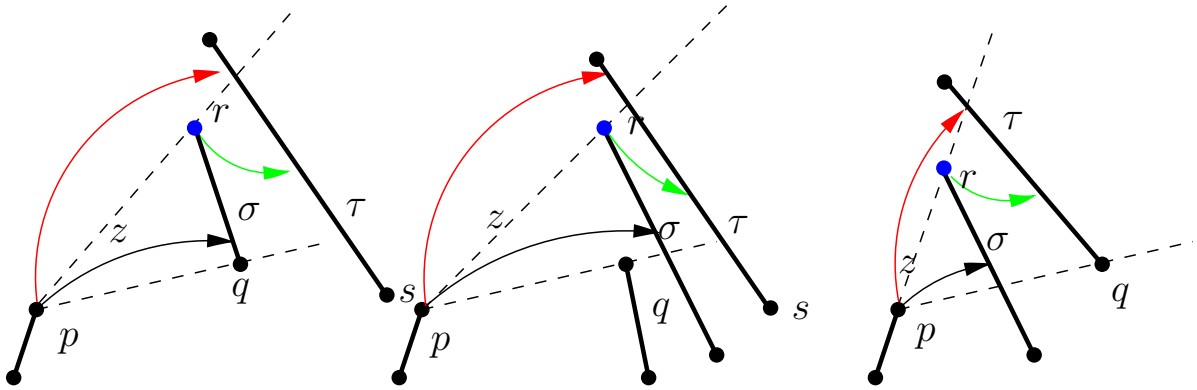


Sweep mit Bearbeitungsreihenfolge

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)

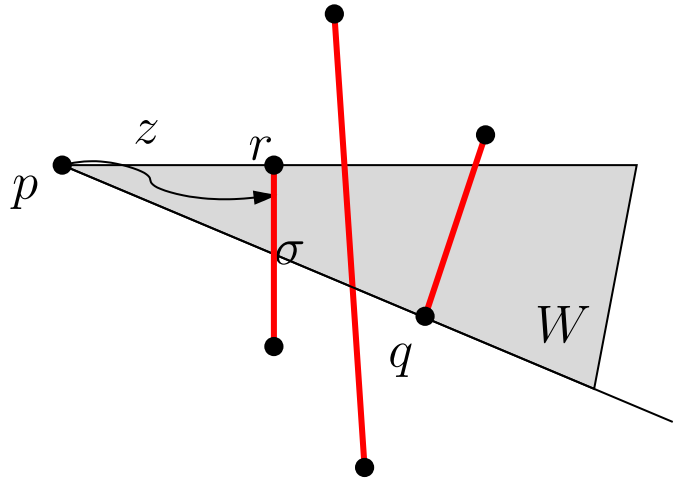
- r Endpunkt von σ

Simultaner Sweep: Ereignisverarbeitung Fall 3(iii)

- r Endpunkt von σ
- Winkelbereich W: Punktfrei

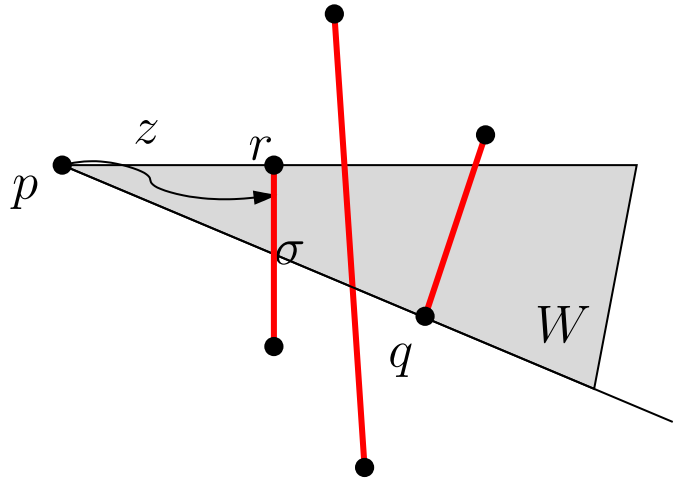
Simultaner Sweep: Ereignisverarbeitung Fall 3(iii)

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



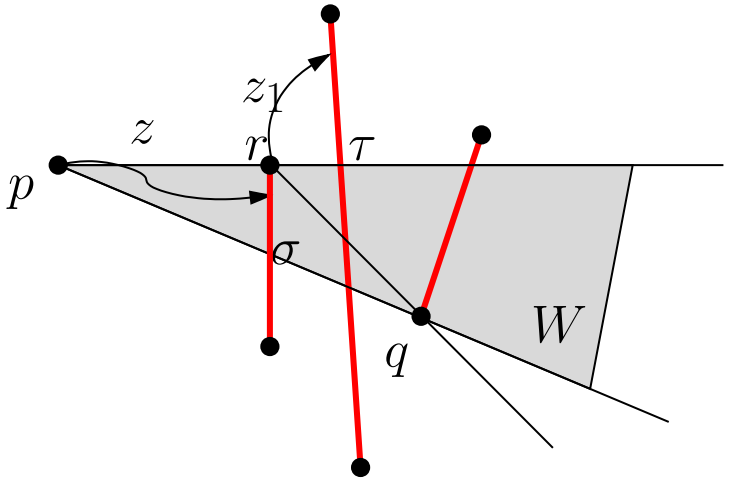
Simultaner Sweep: Ereignisverarbeitung Fall 3(iii)

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



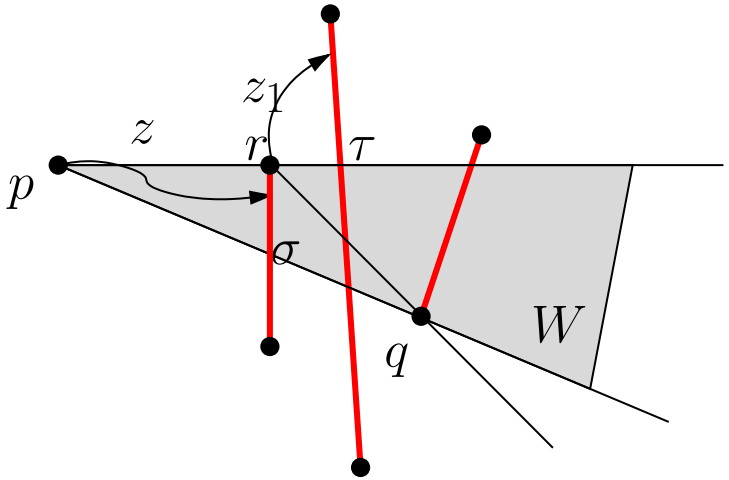
Simultaner Sweep: Ereignisverarbeitung Fall 3(iii)

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

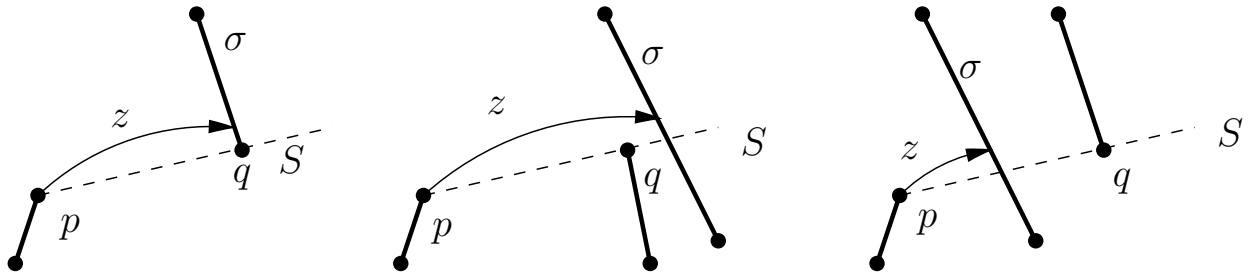


Simultaner Sweep: Ereignisverarbeitung Fall 3(iii)

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

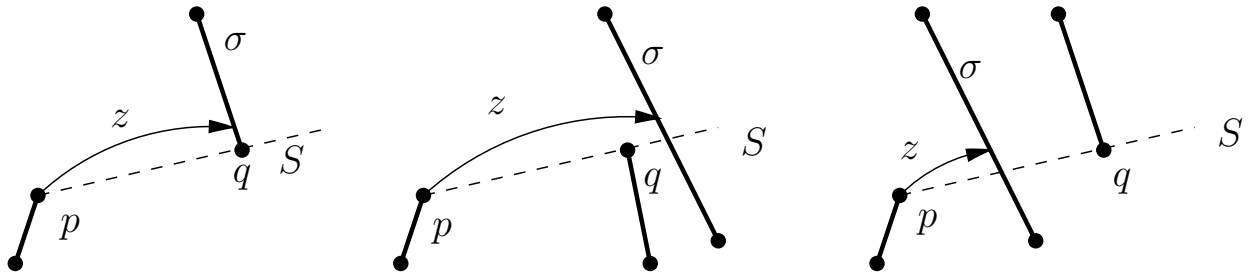


Simultaner Sweep: Laufzeit und Korrektheit



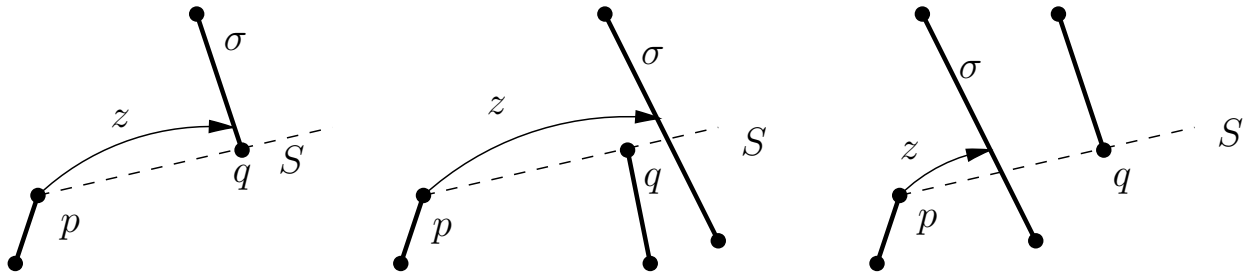
Simultaner Sweep: Laufzeit und Korrektheit

- Invariante ist stets erfüllt! Zeiger auf σ korrekt



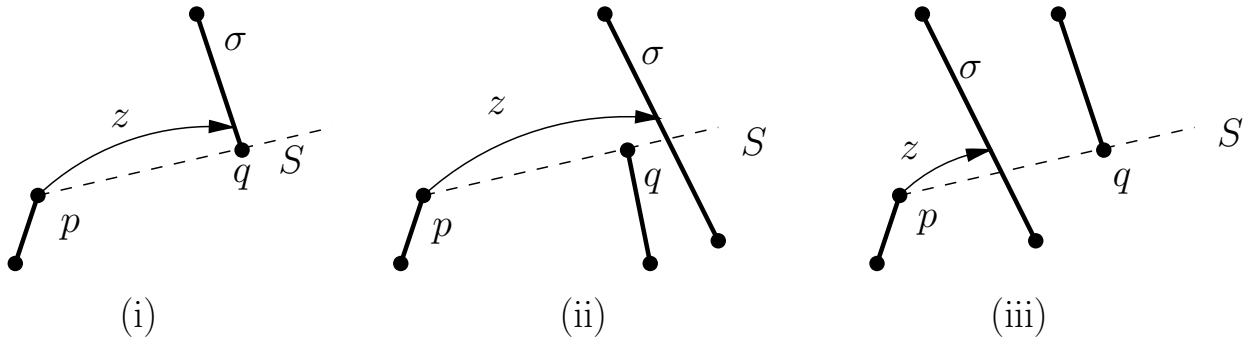
Simultaner Sweep: Laufzeit und Korrektheit

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



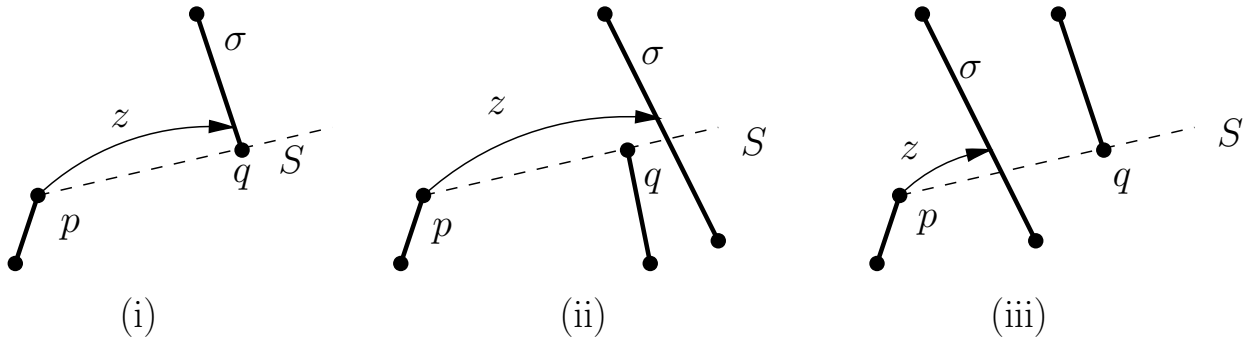
Simultaner Sweep: Laufzeit und Korrektheit

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



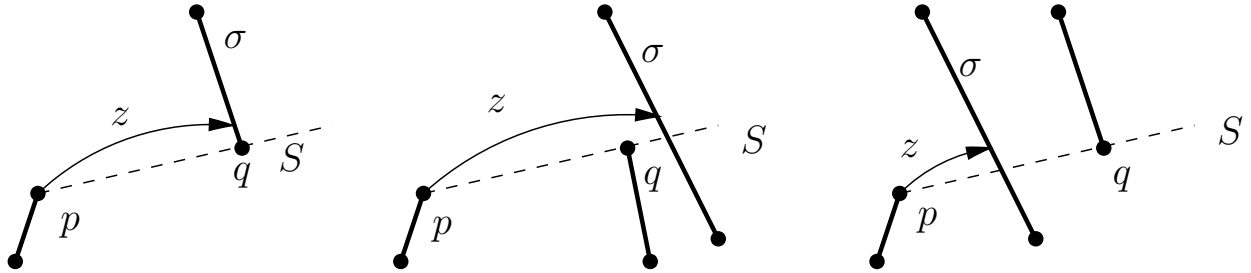
Simultaner Sweep: Laufzeit und Korrektheit

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



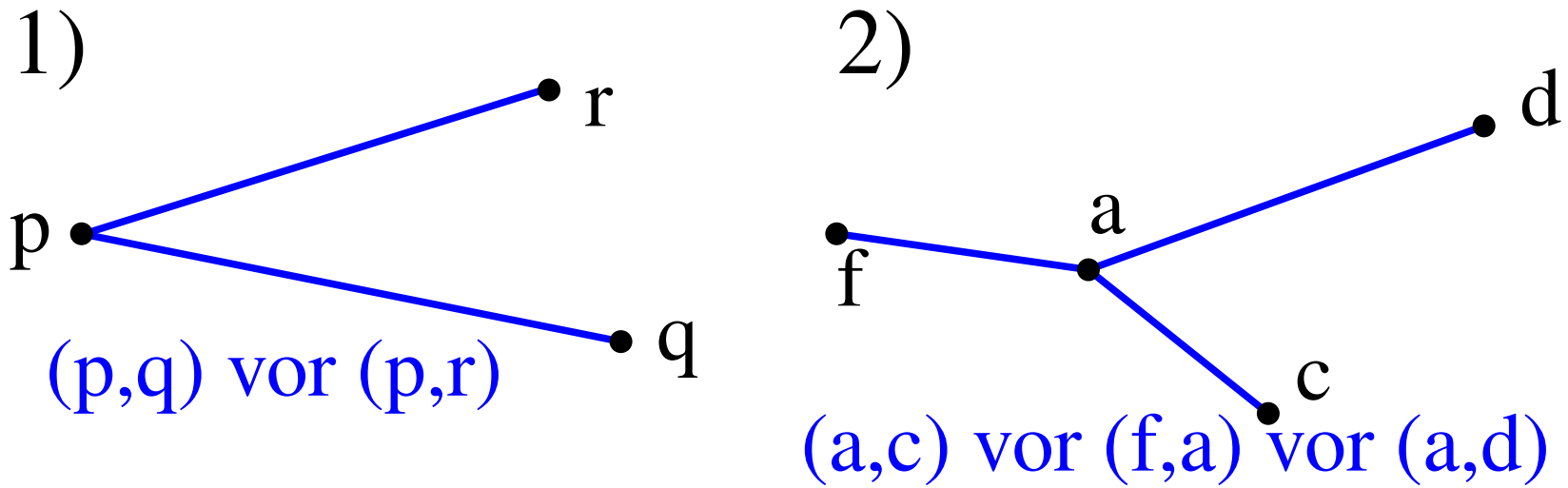
Simultaner Sweep: Laufzeit und Korrektheit

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

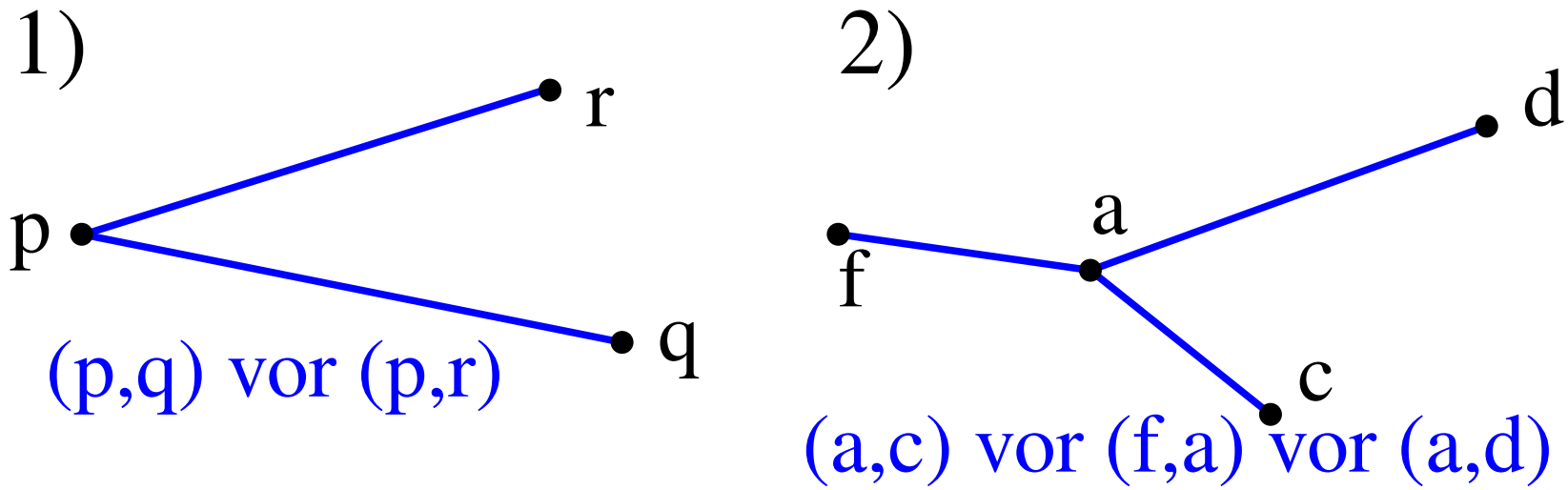


Bearbeitungsreihenfolge in $O(n^2)$

Bearbeitungsreihenfolge in $O(n^2)$

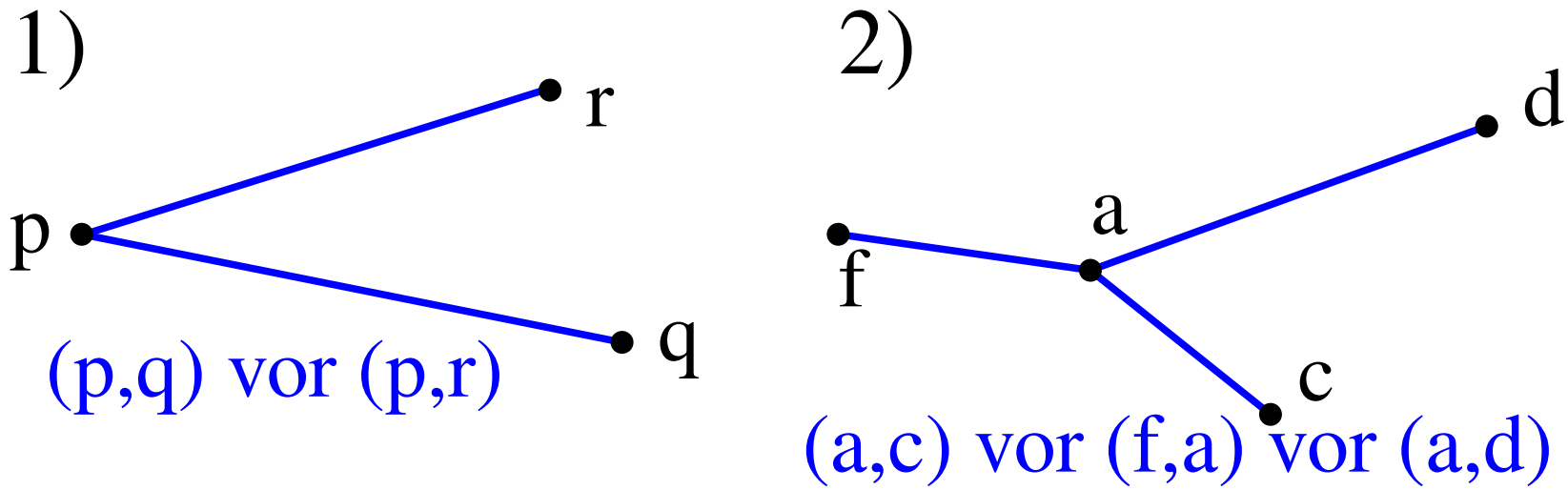


Bearbeitungsreihenfolge in $O(n^2)$



Offensichtlich wird nicht mehr gebraucht!

Bearbeitungsreihenfolge in $O(n^2)$



Offensichtlich wird nicht mehr gebraucht!

Diese muss nun berechnet werden!

Dualitätsprinzip

Dualitätsprinzip

1. Ordnungserhaltende Abbildung zwischen geometrischen Objekten

Dualitätsprinzip

1. Ordnungserhaltende Abbildung zwischen geometrischen Objekten
2. Transformiere Punkt $p = (p_x, p_y)$ zu $p^* : \{Y = p_x X - p_y\}$

Dualitätsprinzip

1. Ordnungserhaltende Abbildung zwischen geometrischen Objekten
2. Transformiere Punkt $p = (p_x, p_y)$ zu $p^* : \{Y = p_x X - p_y\}$
3. Steigung zw. zwei Punkten $p = (p_x, p_y)$ und $q = (q_x, q_y)$, $p_x < q_x$

Dualitätsprinzip

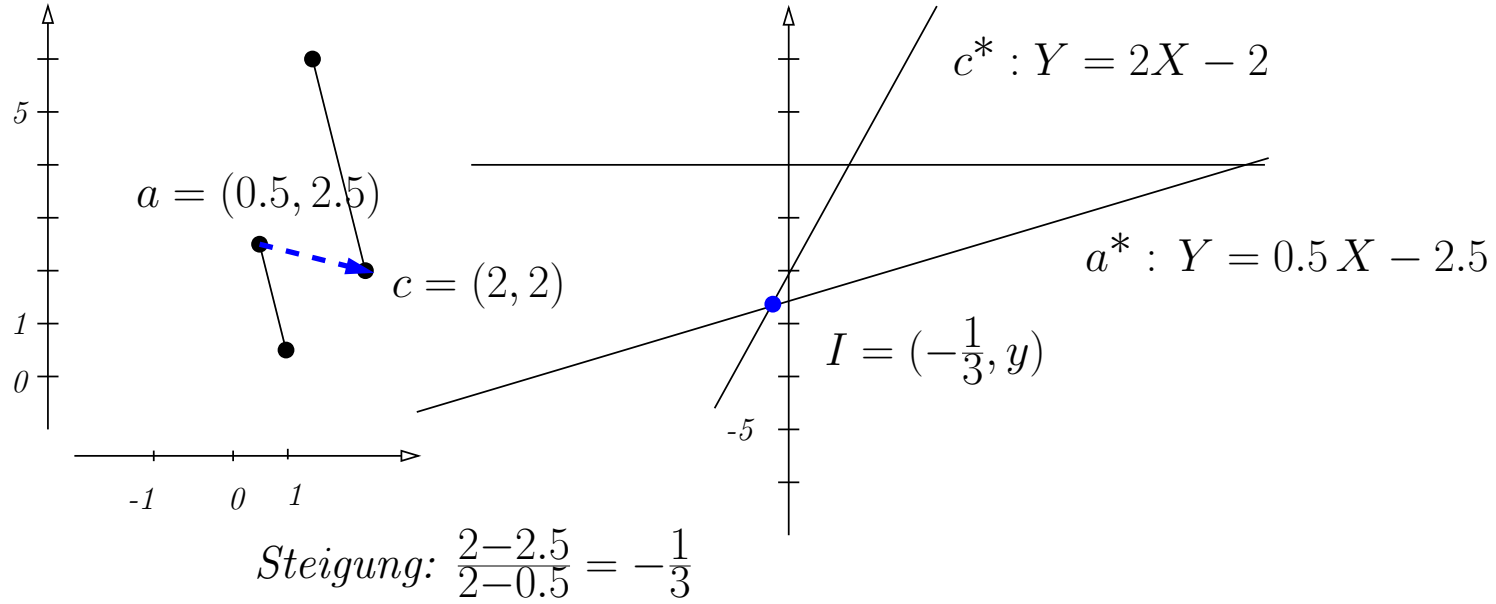
1. Ordnungserhaltende Abbildung zwischen geometrischen Objekten
2. Transformiere Punkt $p = (p_x, p_y)$ zu $p^* : \{Y = p_x X - p_y\}$
3. Steigung zw. zwei Punkten $p = (p_x, p_y)$ und $q = (q_x, q_y)$, $p_x < q_x$
4. Ergebnis $\frac{q_y - p_y}{q_x - p_x}$!

Dualitätsprinzip

1. Ordnungserhaltende Abbildung zwischen geometrischen Objekten
2. Transformiere Punkt $p = (p_x, p_y)$ zu $p^* : \{Y = p_x X - p_y\}$
3. Steigung zw. zwei Punkten $p = (p_x, p_y)$ und $q = (q_x, q_y)$, $p_x < q_x$
4. Ergebnis $\frac{q_y - p_y}{q_x - p_x}$! Entspricht X-Koordinate von $p^* \cap q^*$

Dualitätsprinzip

1. Ordnungserhaltende Abbildung zwischen geometrischen Objekten
2. Transformiere Punkt $p = (p_x, p_y)$ zu $p^* : \{Y = p_x X - p_y\}$
3. Steigung zw. zwei Punkten $p = (p_x, p_y)$ und $q = (q_x, q_y)$, $p_x < q_x$
4. Ergebnis $\frac{q_y - p_y}{q_x - p_x}$! Entspricht X-Koordinate von $p^* \cap q^*$



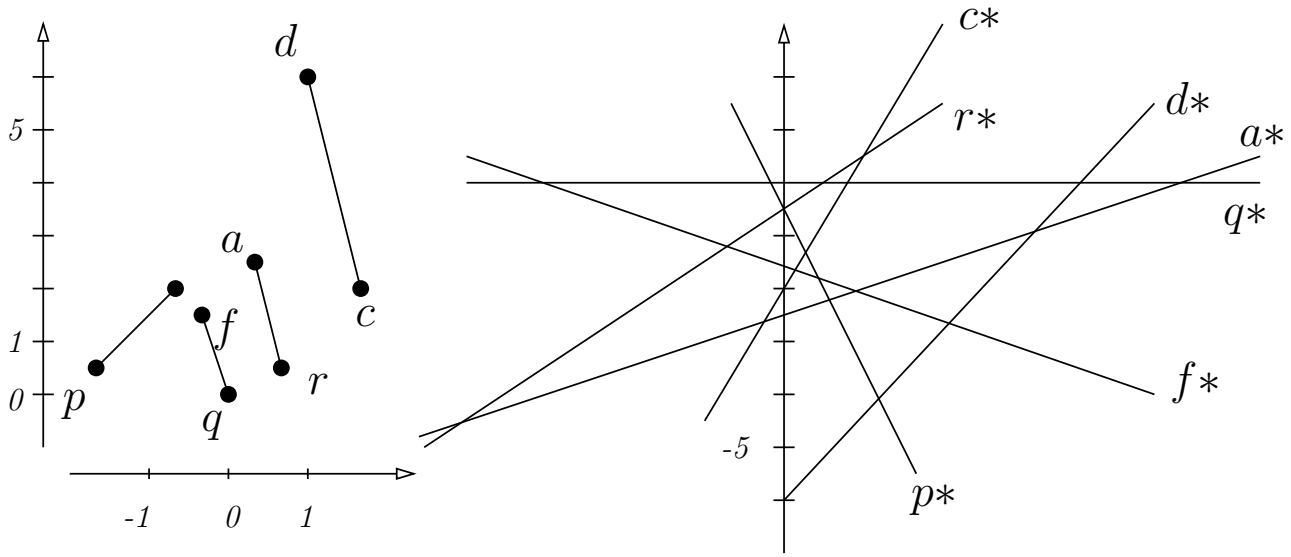
Dualitätsprinzip

Dualitätsprinzip

1. Transformiere alle Punkte $p = (p_x, p_y)$ zu $p^* : \{Y = p_x X - p_y\}$

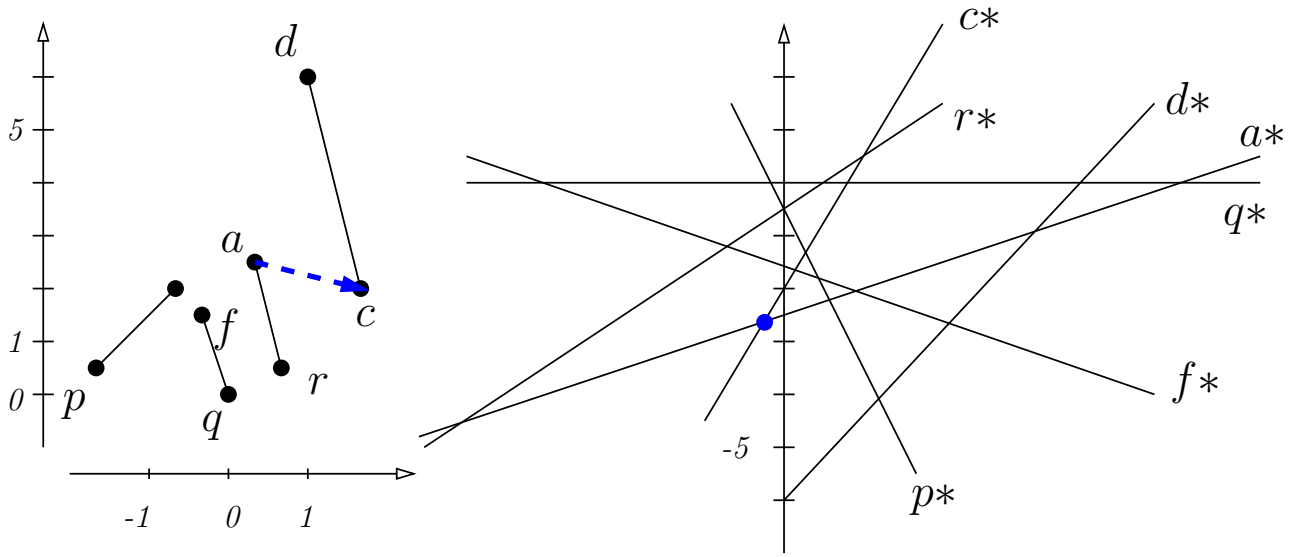
Dualitätsprinzip

1. Transformiere alle Punkte $p = (p_x, p_y)$ zu $p^* : \{Y = p_x X - p_y\}$
2. Bearbeitungsreihenfolge: 2 Bedingungen



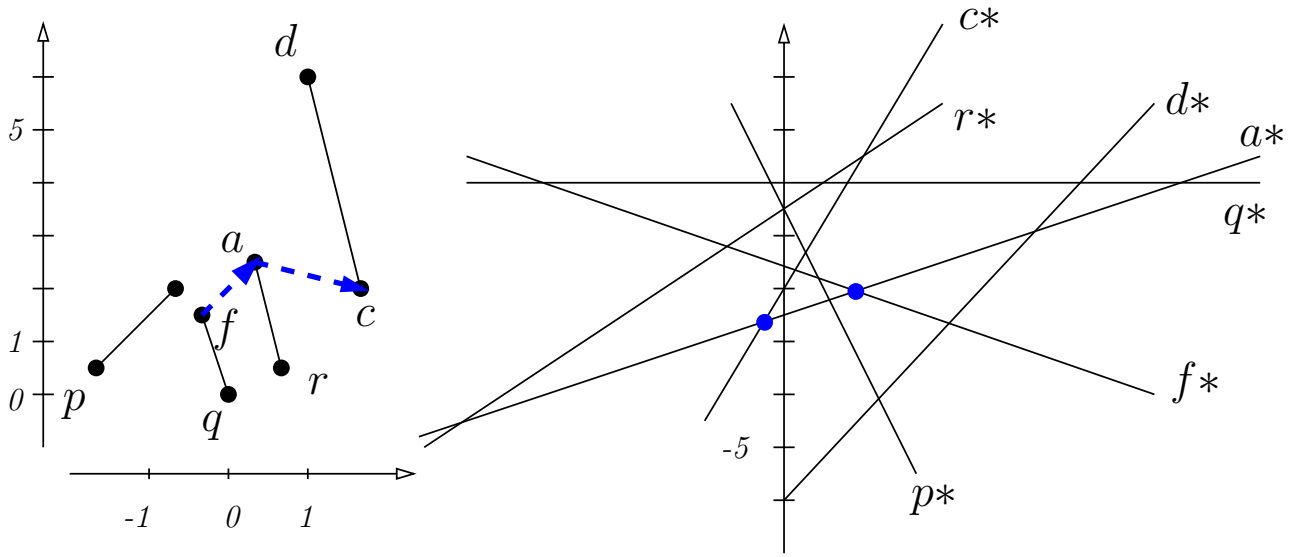
Dualitätsprinzip

1. Transformiere alle Punkte $p = (p_x, p_y)$ zu $p^* : \{Y = p_x X - p_y\}$
2. Bearbeitungsreihenfolge: 2 Bedingungen
3. Reihenfolge der Schnittpunkte entlang jeder Geraden einhalten



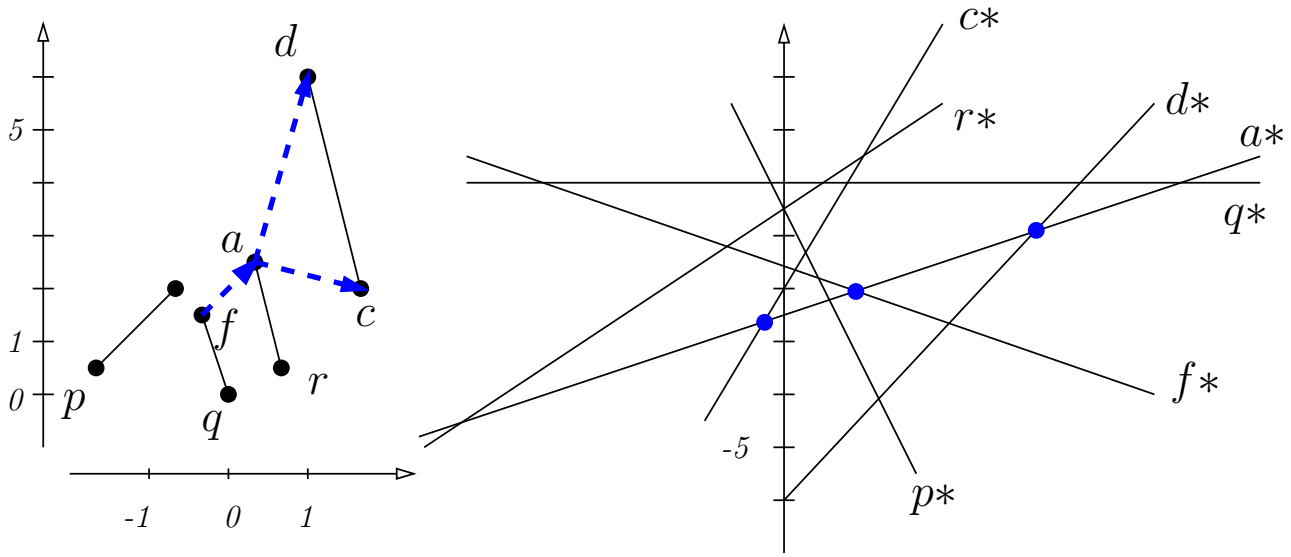
Dualitätsprinzip

1. Transformiere alle Punkte $p = (p_x, p_y)$ zu $p^* : \{Y = p_x X - p_y\}$
2. Bearbeitungsreihenfolge: 2 Bedingungen
3. Reihenfolge der Schnittpunkte entlang jeder Geraden einhalten



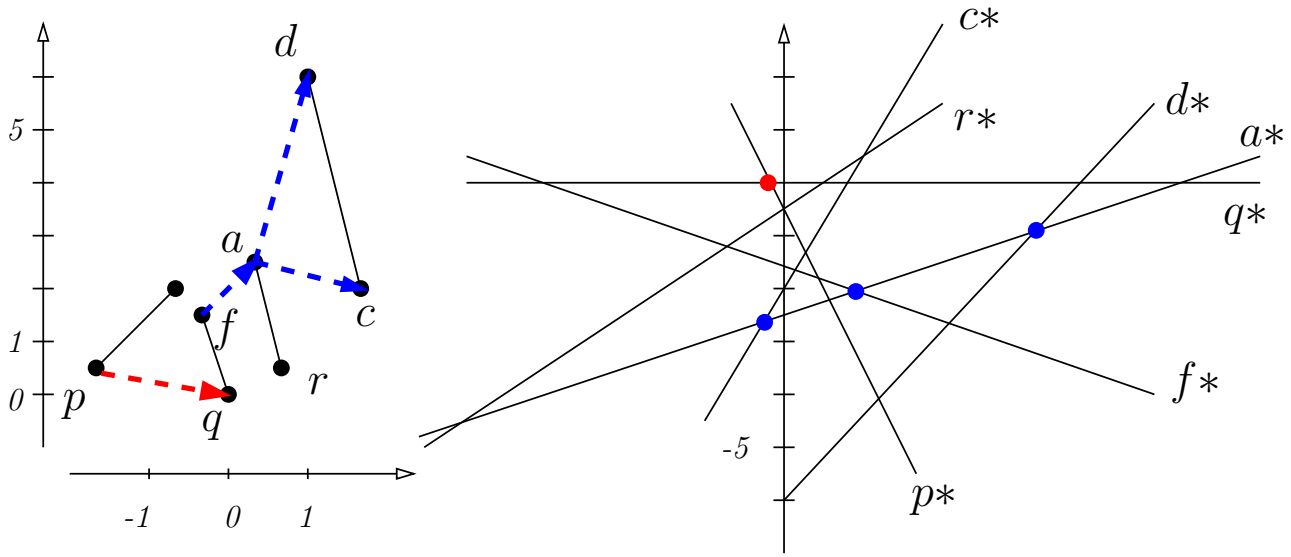
Dualitätsprinzip

1. Transformiere alle Punkte $p = (p_x, p_y)$ zu $p^* : \{Y = p_x X - p_y\}$
2. Bearbeitungsreihenfolge: 2 Bedingungen
3. Reihenfolge der Schnittpunkte entlang jeder Geraden einhalten



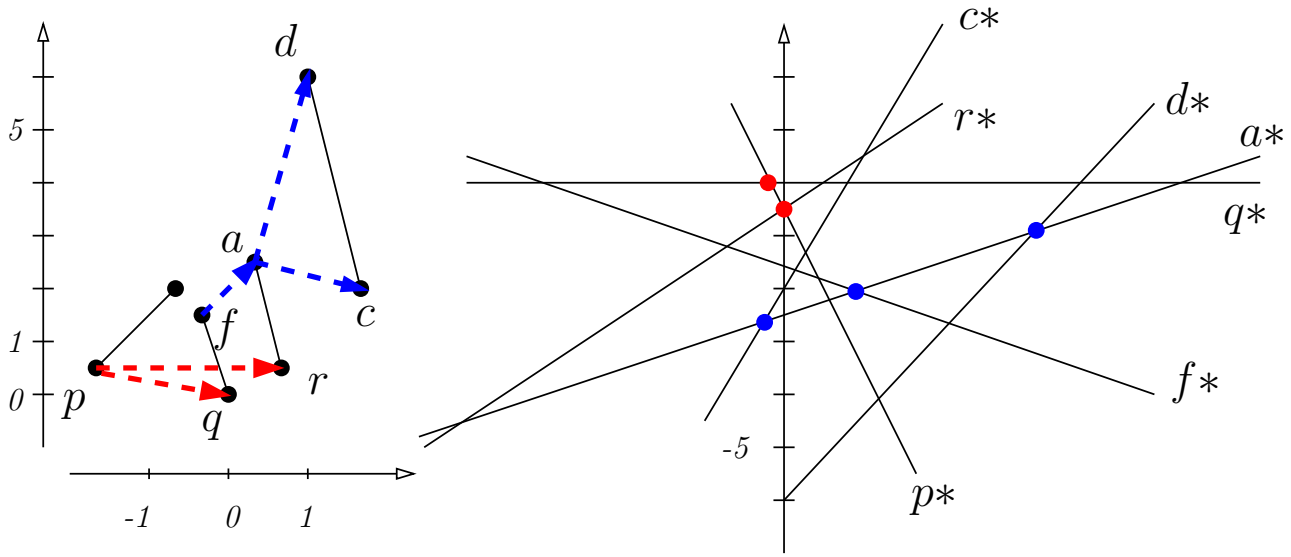
Dualitätsprinzip

1. Transformiere alle Punkte $p = (p_x, p_y)$ zu $p^* : \{Y = p_x X - p_y\}$
2. Bearbeitungsreihenfolge: 2 Bedingungen
3. Reihenfolge der Schnittpunkte entlang jeder Geraden einhalten



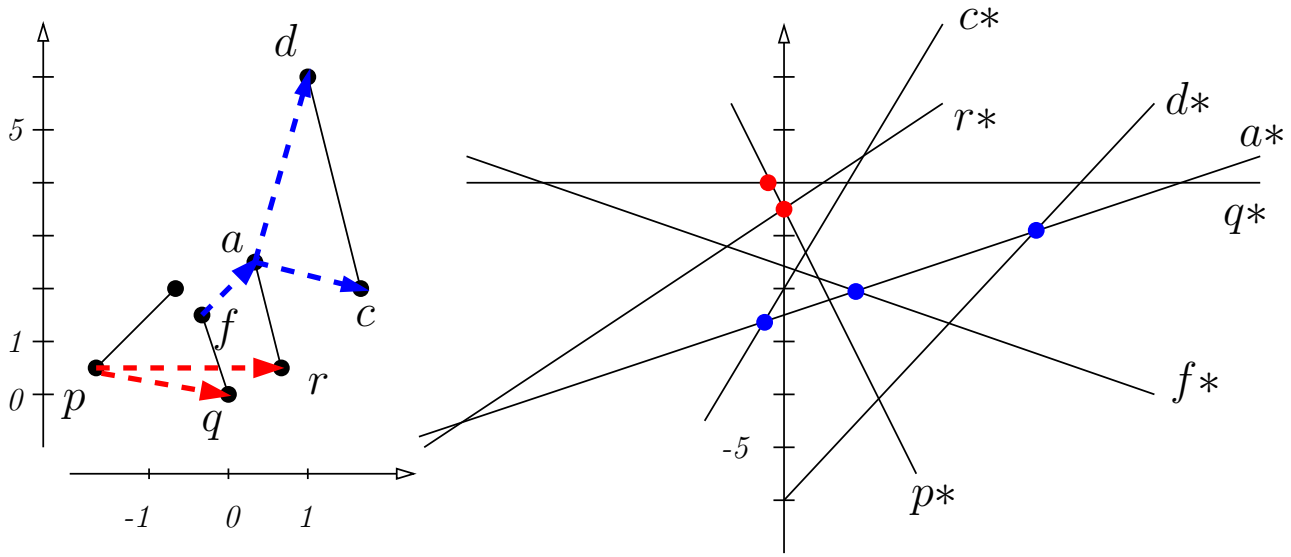
Dualitätsprinzip

1. Transformiere alle Punkte $p = (p_x, p_y)$ zu $p^* : \{Y = p_x X - p_y\}$
2. Bearbeitungsreihenfolge: 2 Bedingungen
3. Reihenfolge der Schnittpunkte entlang jeder Geraden einhalten



Dualitätsprinzip

1. Transformiere alle Punkte $p = (p_x, p_y)$ zu $p^* : \{Y = p_x X - p_y\}$
2. Bearbeitungsreihenfolge: 2 Bedingungen
3. Reihenfolge der Schnittpunkte entlang jeder Geraden einhalten



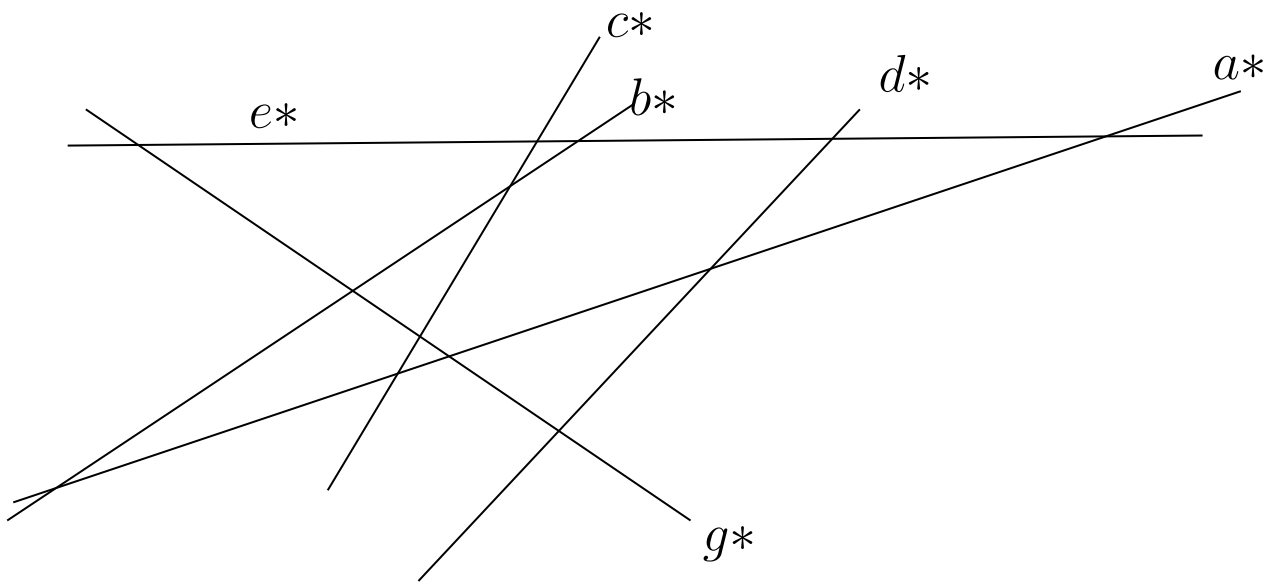
Arrangement von Geraden

Arrangement von Geraden

- Berechne die Reihenfolge der Schnittpunkte entlang jeder Geraden

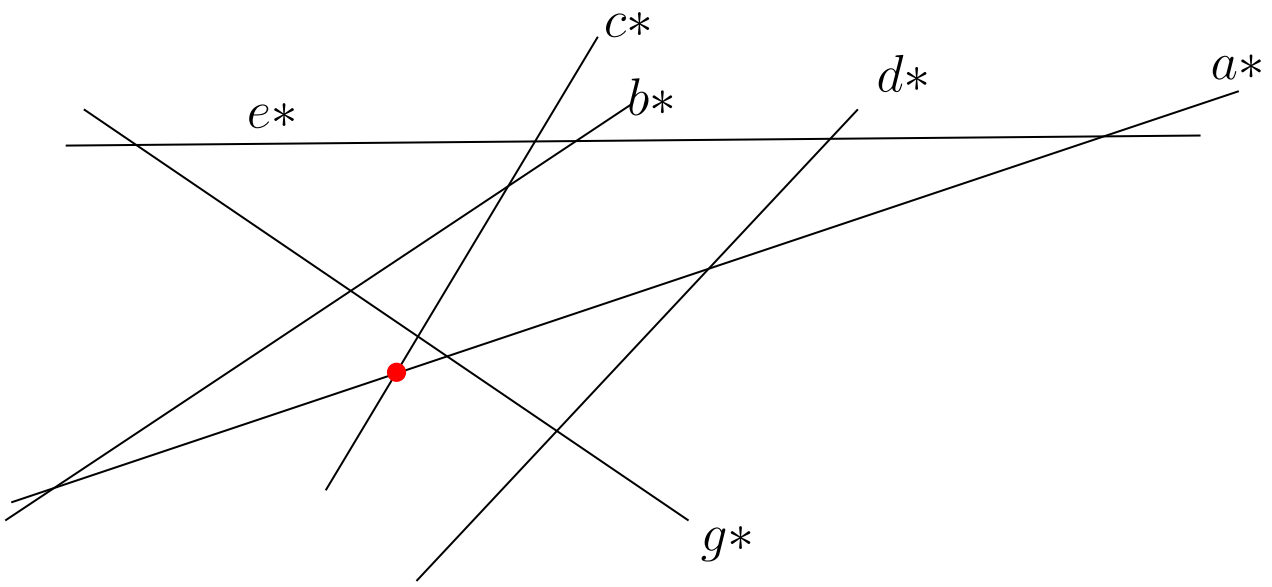
Arrangement von Geraden

- Berechne die Reihenfolge der Schnittpunkte entlang jeder Geraden
- Nicht eindeutig!!



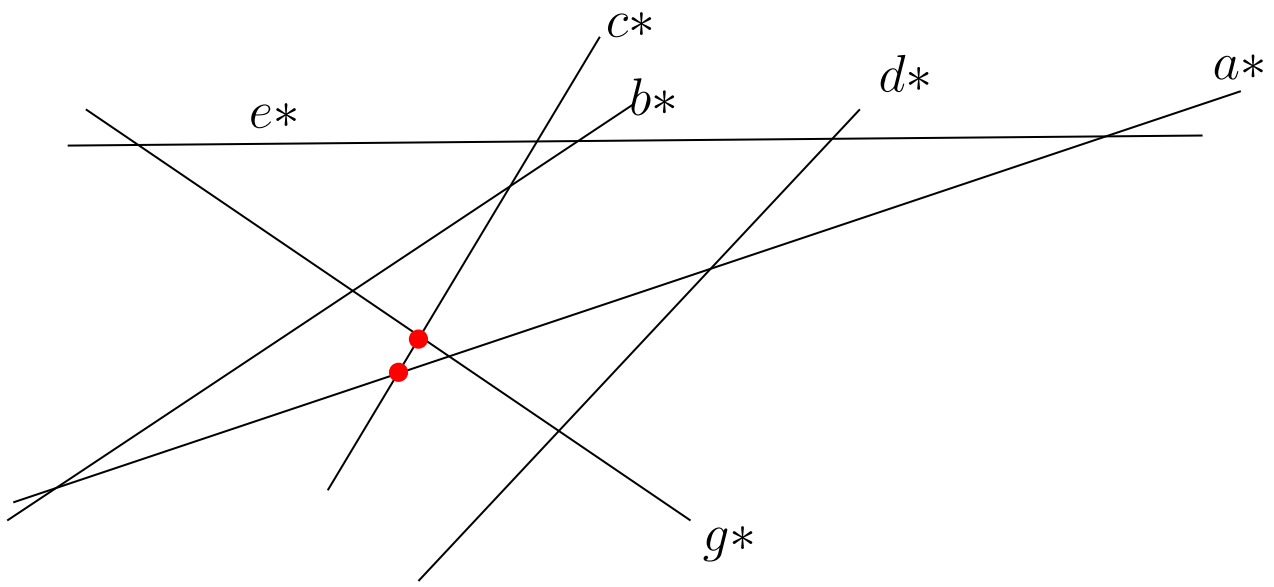
Arrangement von Geraden

- Berechne die Reihenfolge der Schnittpunkte entlang jeder Geraden
- Nicht eindeutig!!
- $a^* \cap c^*$,



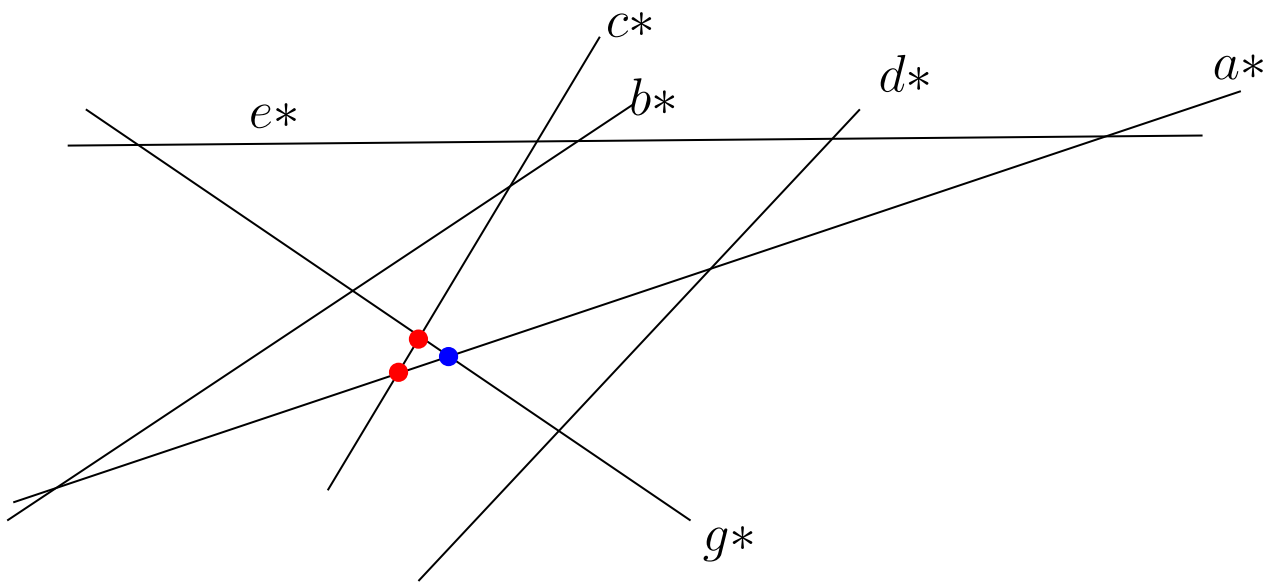
Arrangement von Geraden

- Berechne die Reihenfolge der Schnittpunkte entlang jeder Geraden
- Nicht eindeutig!!
- $a^* \cap c^*$, $g^* \cap c^*$,



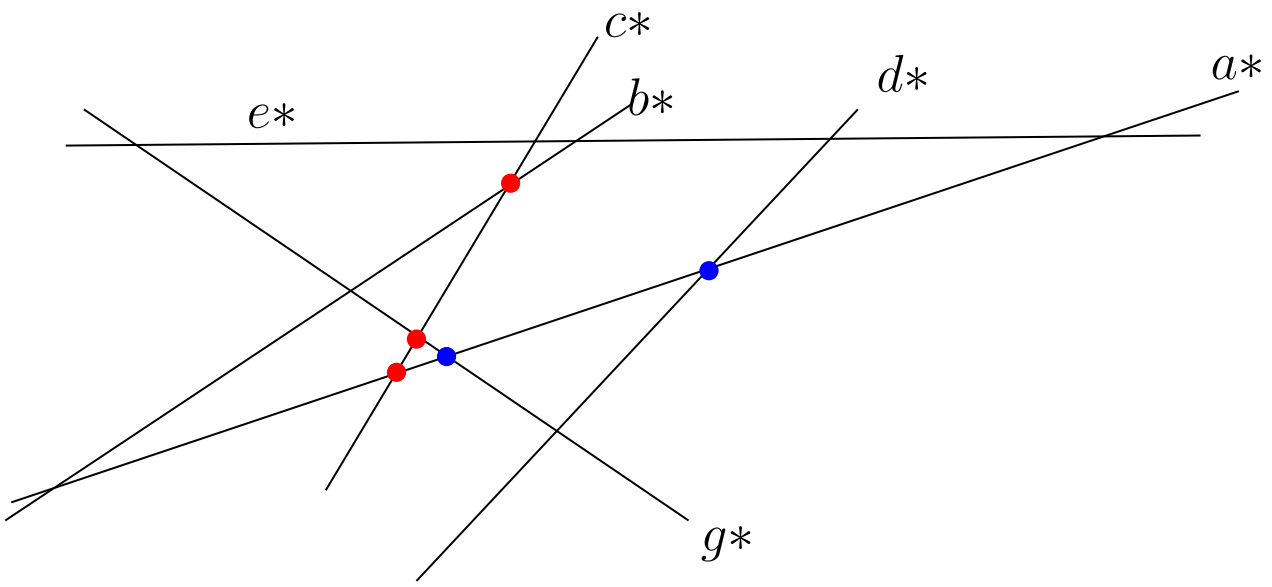
Arrangement von Geraden

- Berechne die Reihenfolge der Schnittpunkte entlang jeder Geraden
- Nicht eindeutig!!
- $a^* \cap c^*$, $g^* \cap c^*$, $g^* \cap a^*$,



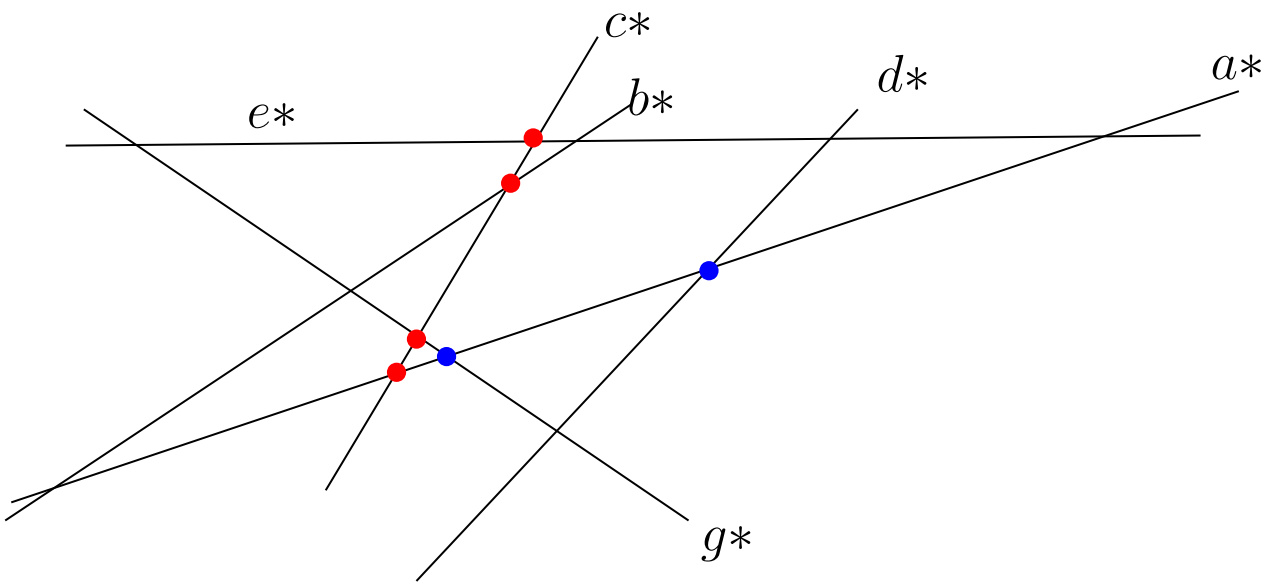
Arrangement von Geraden

- Berechne die Reihenfolge der Schnittpunkte entlang jeder Geraden
- Nicht eindeutig!!
- $a^* \cap c^*$, $g^* \cap c^*$, $g^* \cap a^*$, $c^* \cap b^*, a^* \cap d^*$,



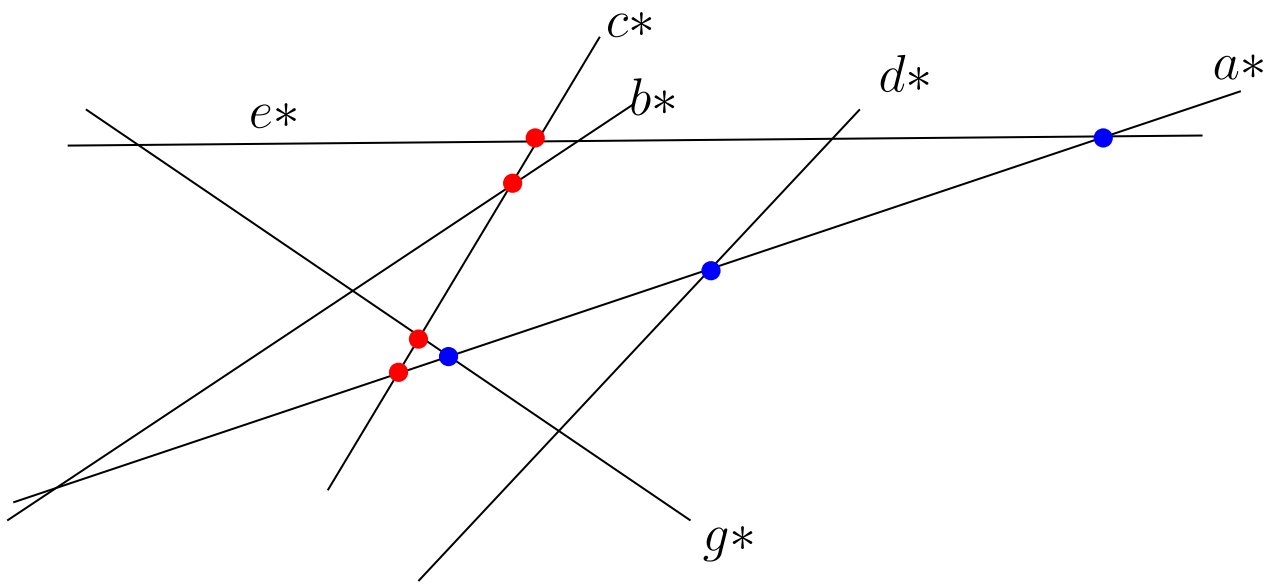
Arrangement von Geraden

- Berechne die Reihenfolge der Schnittpunkte entlang jeder Geraden
- Nicht eindeutig!!
- $a^* \cap c^*$, $g^* \cap c^*$, $g^* \cap a^*$, $c^* \cap b^*, a^* \cap d^*$, $c^* \cap e^*$,

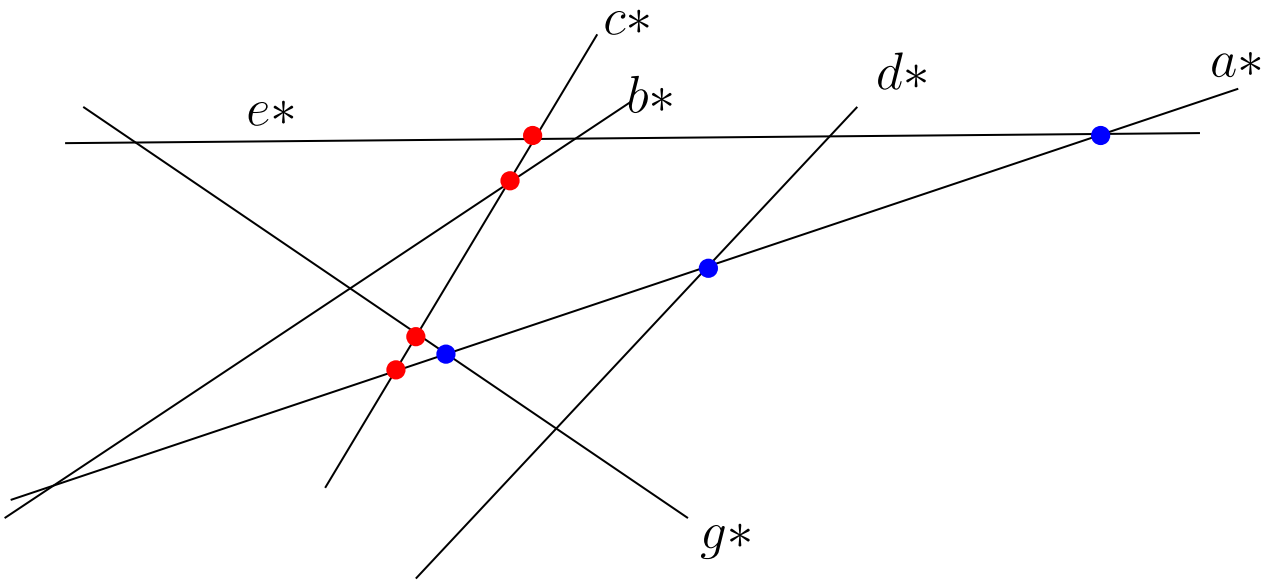


Arrangement von Geraden

- Berechne die Reihenfolge der Schnittpunkte entlang jeder Geraden
- Nicht eindeutig!!
- $a^* \cap c^*$, $g^* \cap c^*$, $g^* \cap a^*$, $c^* \cap b^*, a^* \cap d^*$, $c^* \cap e^*$, $a^* \cap e^*$

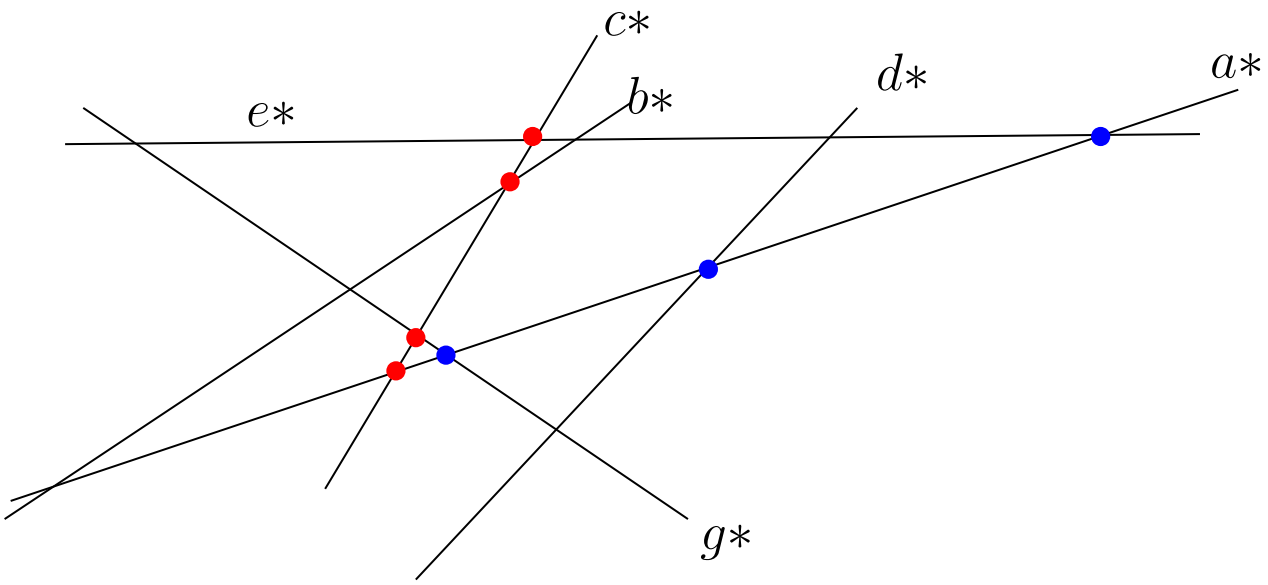


Neue Aufgabe für Arrangement von Geraden



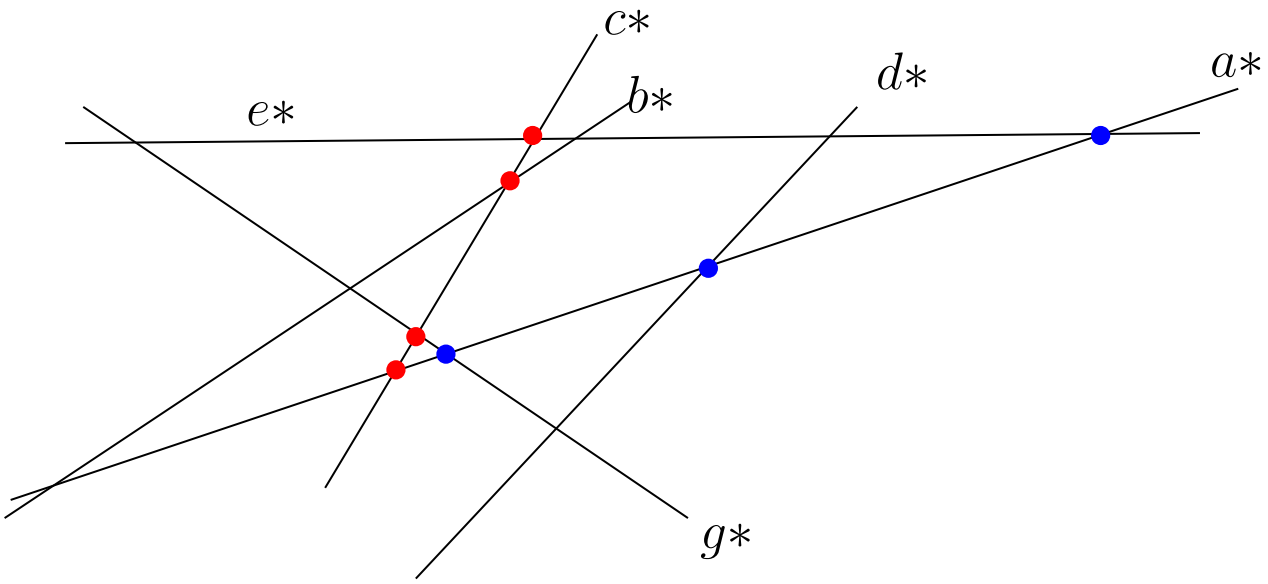
Neue Aufgabe für Arrangement von Geraden

- Berechne Schnittpunkte gemäß Ordnung entlang jeder Geraden



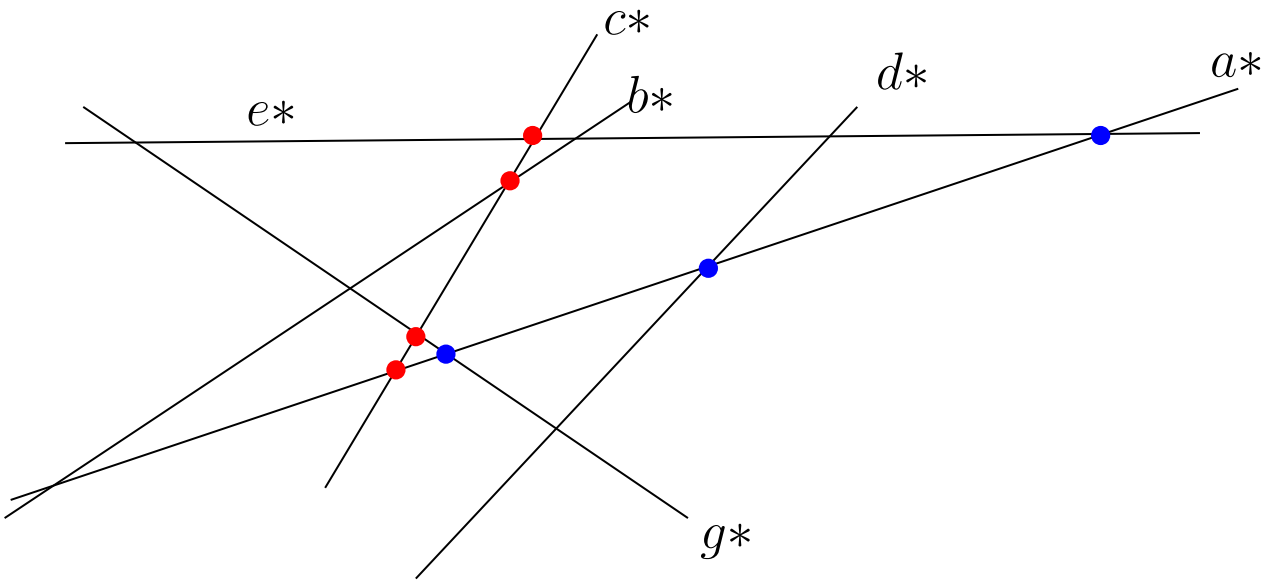
Neue Aufgabe für Arrangement von Geraden

- Berechne Schnittpunkte gemäß Ordnung entlang jeder Geraden
- Naiv!



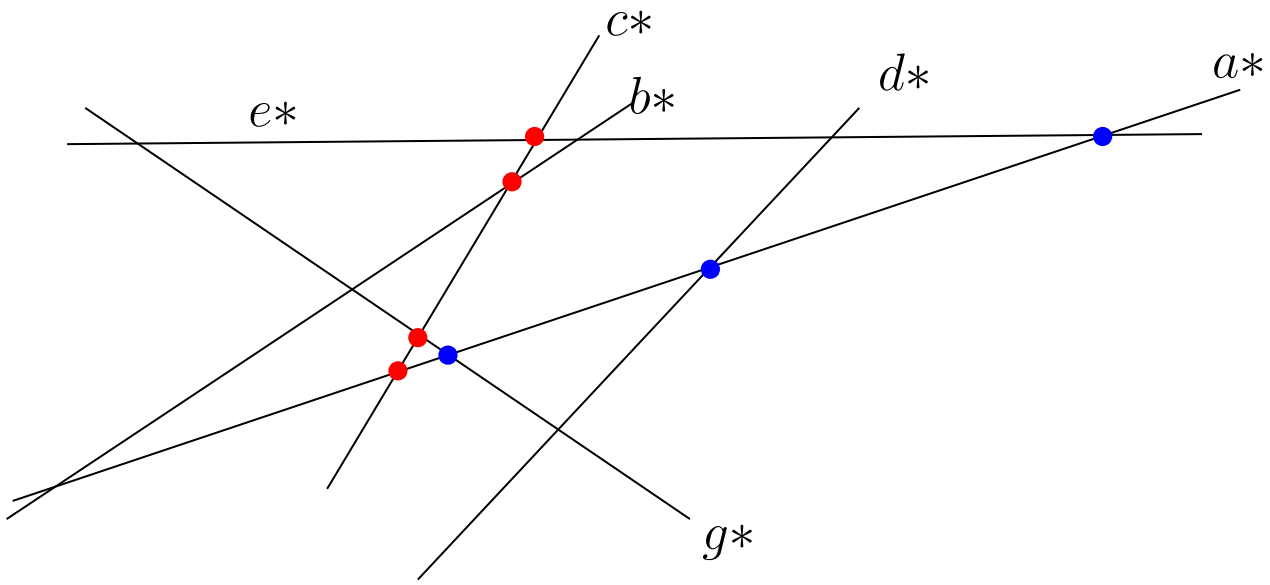
Neue Aufgabe für Arrangement von Geraden

- Berechne Schnittpunkte gemäß Ordnung entlang jeder Geraden
- Naiv! n^2 Schnittpunkte sortieren:



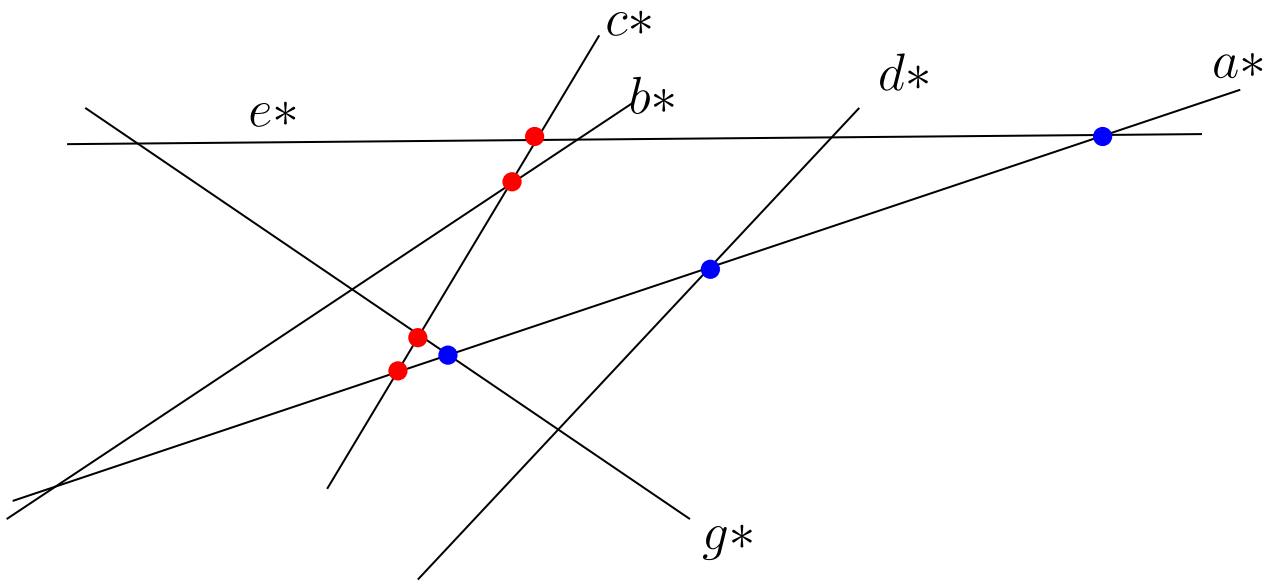
Neue Aufgabe für Arrangement von Geraden

- Berechne Schnittpunkte gemäß Ordnung entlang jeder Geraden
- Naiv! n^2 Schnittpunkte sortieren: $O(n^2 \log n)$



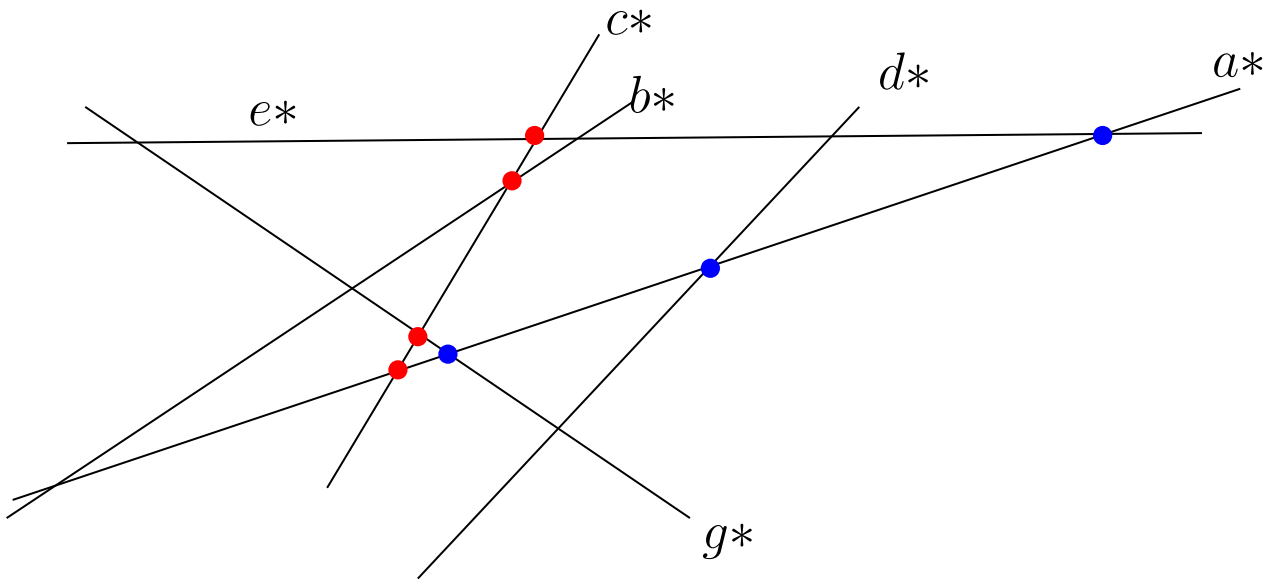
Neue Aufgabe für Arrangement von Geraden

- Berechne Schnittpunkte gemäß Ordnung entlang jeder Geraden
- Naiv! n^2 Schnittpunkte sortieren: $O(n^2 \log n)$
- Sweep mit Sweep-Gerade: $O((n+k)\log n)$, $k \in \Omega(n^2)$



Neue Aufgabe für Arrangement von Geraden

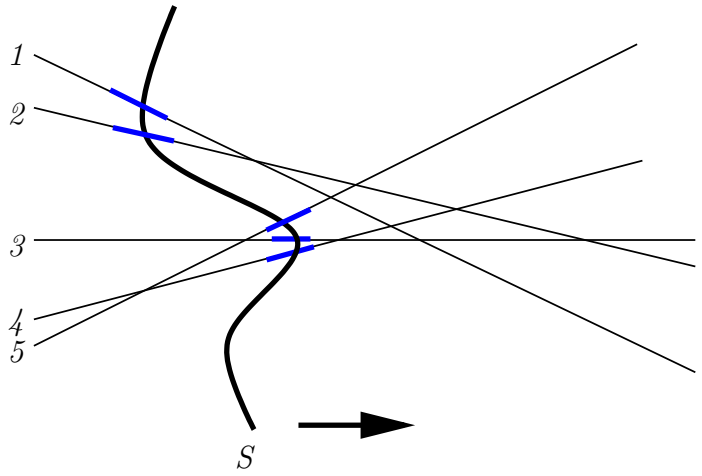
- Berechne Schnittpunkte gemäß Ordnung entlang jeder Geraden
- Naiv! n^2 Schnittpunkte sortieren: $O(n^2 \log n)$
- Sweep mit Sweep-Gerade: $O((n+k)\log n)$, $k \in \Omega(n^2)$
- Idee: Keine volle Ordnung, Sweep mit Pseudogeraden



Sweep mit Pseudogeraden

Sweep mit Pseudogeraden

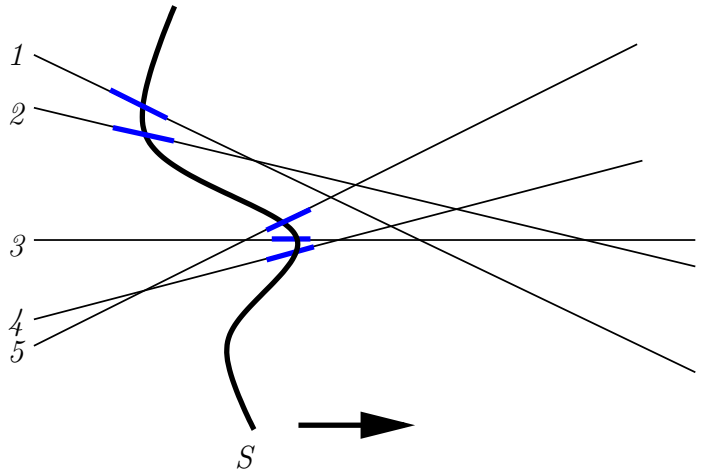
- **Def. 1.2:** Pseudogerade: Kurve schneidet jede Gerade genau einmal



$L=(1,2,5,3,4)$

Sweep mit Pseudogeraden

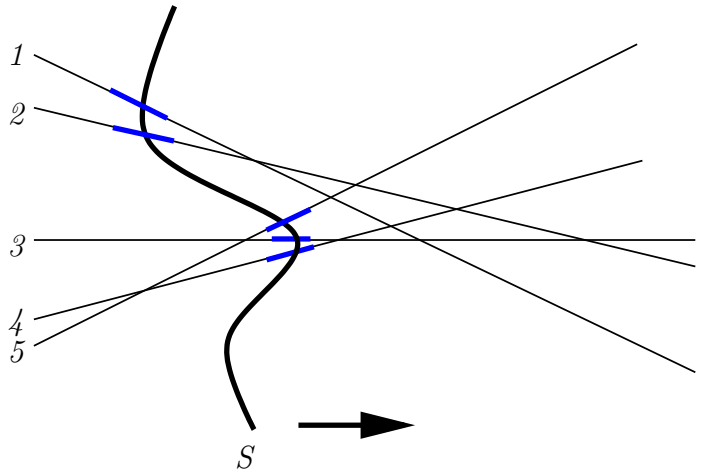
- **Def. 1.2:** Pseudogerade: Kurve schneidet jede Gerade genau einmal
- Schnittpunkte gemäß Ordnung entlang aller Geraden



$L=(1,2,5,3,4)$

Sweep mit Pseudogeraden

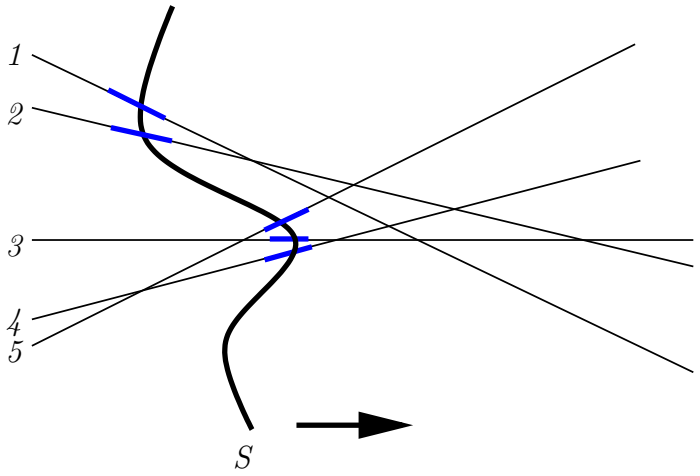
- **Def. 1.2:** Pseudogerade: Kurve schneidet jede Gerade genau einmal
- Schnittpunkte gemäß Ordnung entlang aller Geraden
- Hebe Pseudogerade über angrenzenden Schnittpunkt: Ausgabe!



$L=(1,2,5,3,4)$

Sweep mit Pseudogeraden

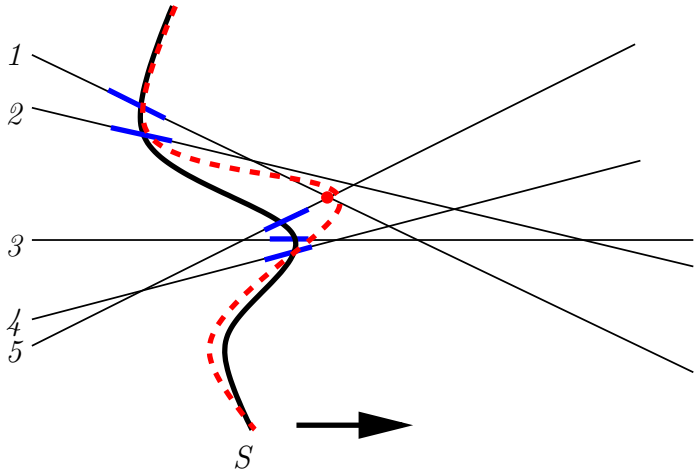
- **Def. 1.2:** Pseudogerade: Kurve schneidet jede Gerade genau einmal
- Schnittpunkte gemäß Ordnung entlang aller Geraden
- Hebe Pseudogerade über angrenzenden Schnittpunkt: Ausgabe!
- Welche Möglichkeiten?



$L=(1,2,5,3,4)$

Sweep mit Pseudogeraden

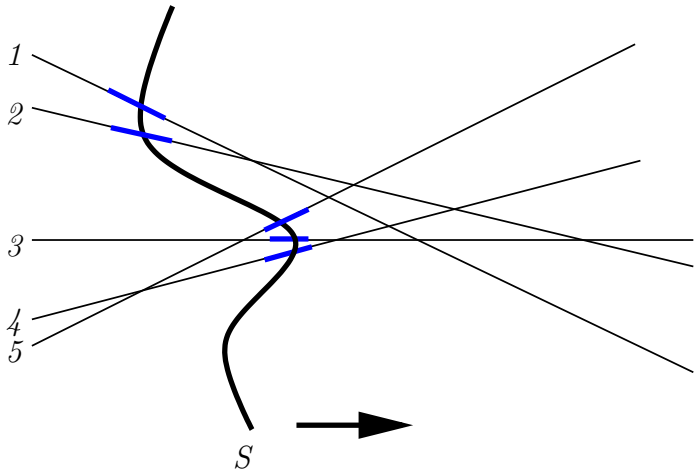
- **Def. 1.2:** Pseudogerade: Kurve schneidet jede Gerade genau einmal
- Schnittpunkte gemäß Ordnung entlang aller Geraden
- Hebe Pseudogerade über angrenzenden Schnittpunkt: Ausgabe!
- Welche Möglichkeiten?



$L=(1,2,5,3,4)$ $L=(1,2,1,5,1,3,4)$

Sweep mit Pseudogeraden

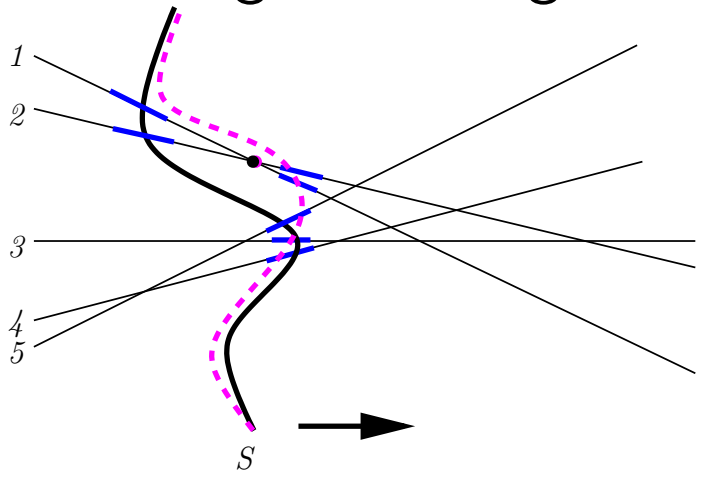
- **Def. 1.2:** Pseudogerade: Kurve schneidet jede Gerade genau einmal
- Schnittpunkte gemäß Ordnung entlang aller Geraden
- Hebe Pseudogerade über angrenzenden Schnittpunkt: Ausgabe!
- Welche Möglichkeiten?



$L=(1,2,5,3,4)$

Sweep mit Pseudogeraden

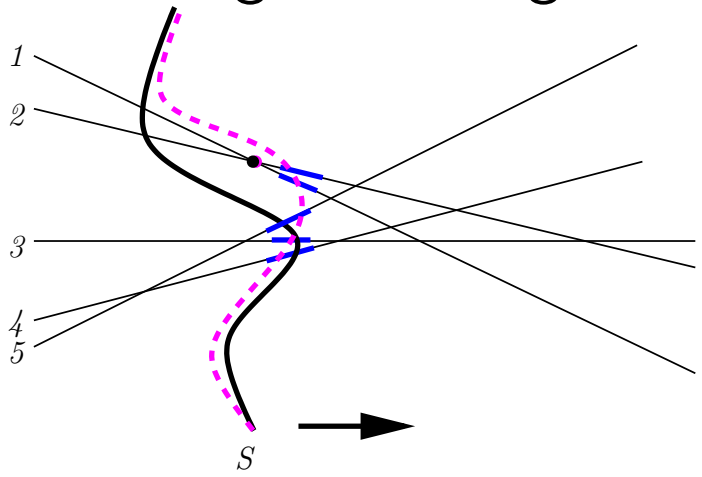
- **Def. 1.2:** Pseudogerade: Kurve schneidet jede Gerade genau einmal
- Schnittpunkte gemäß Ordnung entlang aller Geraden
- Hebe Pseudogerade über angrenzenden Schnittpunkt: Ausgabe!
- Welche Möglichkeiten?
- Pseudogerade Bed. $\Leftrightarrow$ Ausgabe richtige Reihenfolge



$L=(1,2,5,3,4)$ $L=(2,1,5,3,4)$

Sweep mit Pseudogeraden

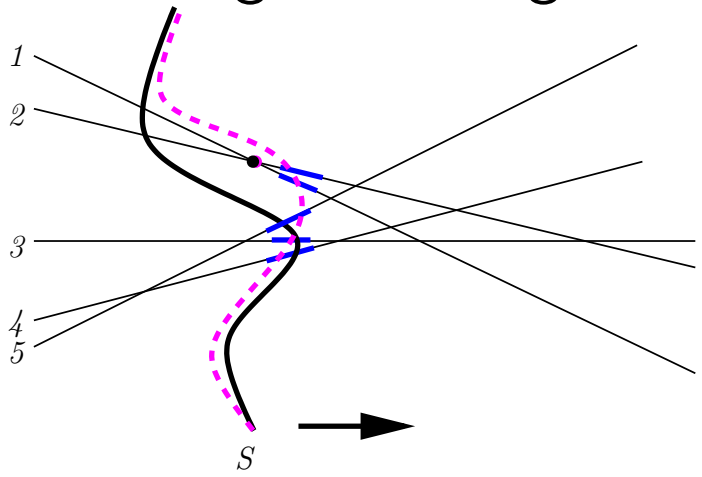
- **Def. 1.2:** Pseudogerade: Kurve schneidet jede Gerade genau einmal
- Schnittpunkte gemäß Ordnung entlang aller Geraden
- Hebe Pseudogerade über angrenzenden Schnittpunkt: Ausgabe!
- Welche Möglichkeiten?
- Pseudogerade Bed. $\Leftrightarrow$ Ausgabe richtige Reihenfolge



$L=(1,2,5,3,4)$ $L=(2,1,5,3,4)$

Sweep mit Pseudogeraden

- **Def. 1.2:** Pseudogerade: Kurve schneidet jede Gerade genau einmal
- Schnittpunkte gemäß Ordnung entlang aller Geraden
- Hebe Pseudogerade über angrenzenden Schnittpunkt: Ausgabe!
- Welche Möglichkeiten?
- Pseudogerade Bed. $\Leftrightarrow$ Ausgabe richtige Reihenfolge

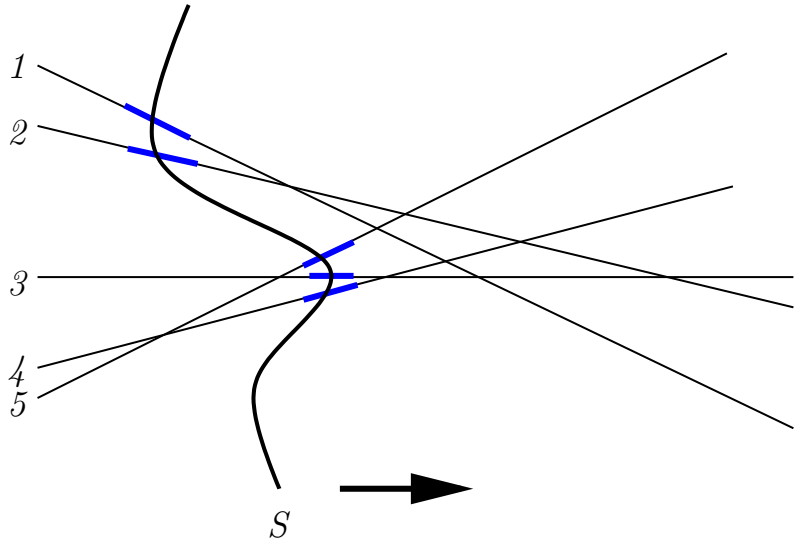


$L=(1,2,5,3,4)$ $L=(2,1,5,3,4)$

Sweep mit Pseudogeraden

Sweep mit Pseudogeraden

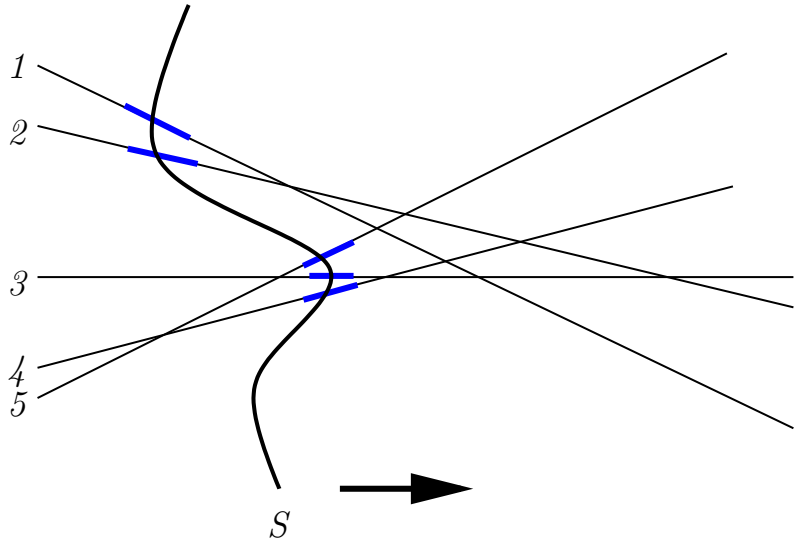
- Jede Gerade genau einmal schneiden,



$L=(1,2,5,3,4)$

Sweep mit Pseudogeraden

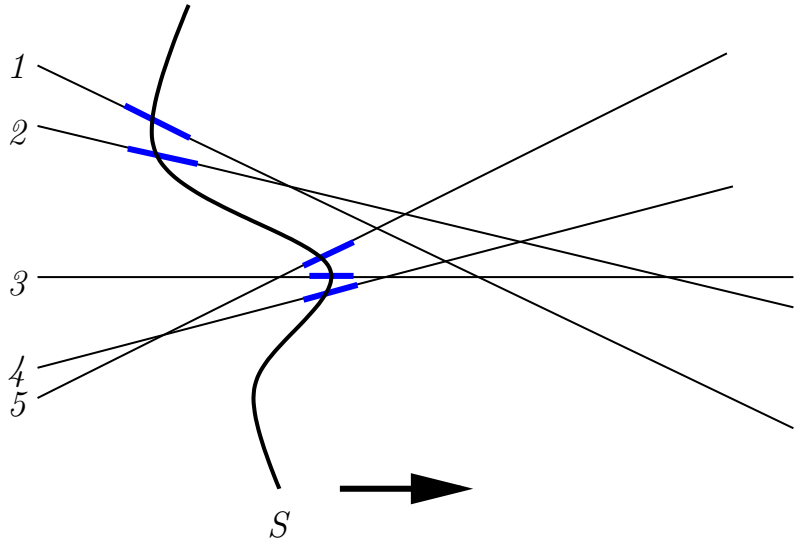
- Jede Gerade genau einmal schneiden, nächster SP



$L=(1,2,5,3,4)$

Sweep mit Pseudogeraden

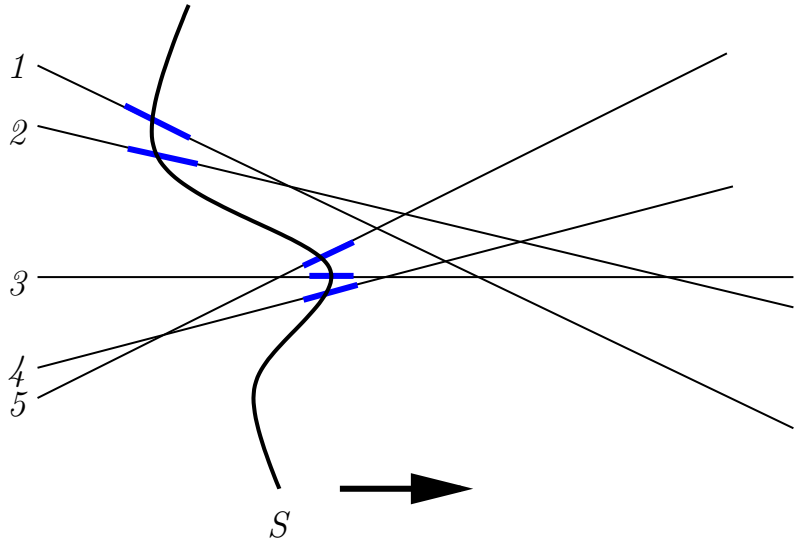
- Jede Gerade genau einmal schneiden, nächster SP
- Bedingung einhalten $\Leftrightarrow$ Reihenfolge SP entlang jeder Geraden



$L=(1,2,5,3,4)$

Sweep mit Pseudogeraden

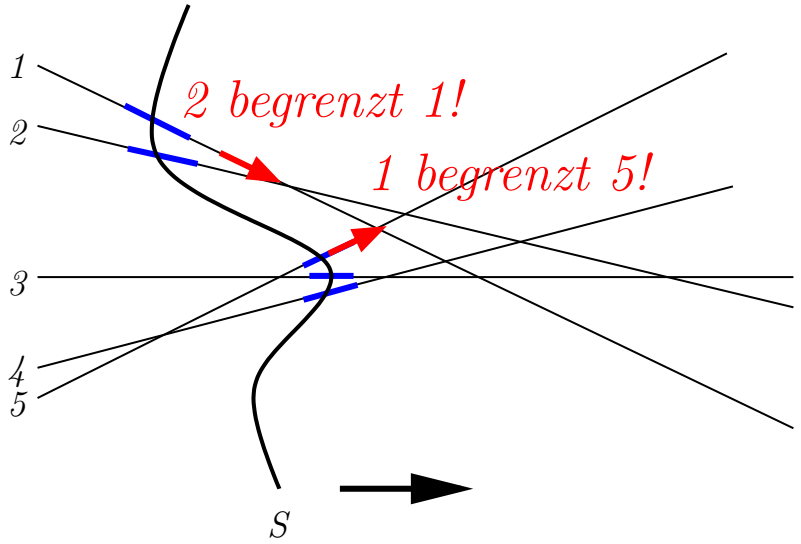
- Jede Gerade genau einmal schneiden, nächster SP
- Bedingung einhalten $\Leftrightarrow$ Reihenfolge SP entlang jeder Geraden
- Auswahl? Begrenzungen ermitteln



$L=(1,2,5,3,4)$

Sweep mit Pseudogeraden

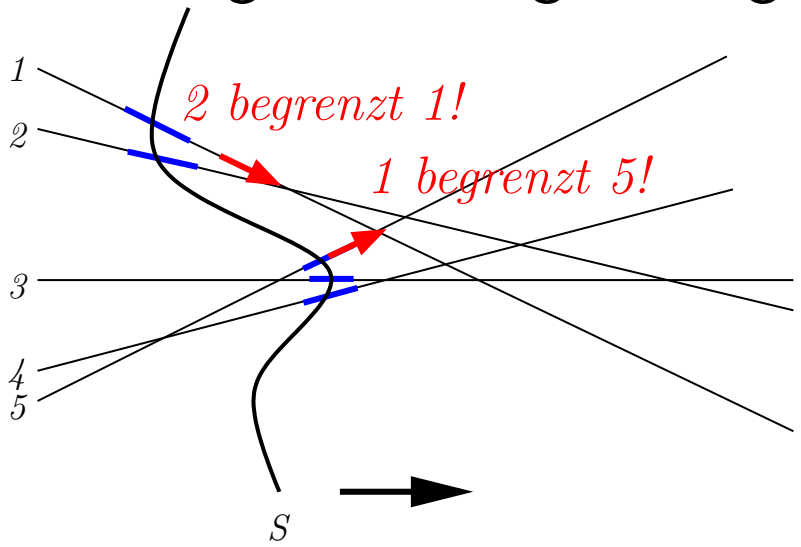
- Jede Gerade genau einmal schneiden, nächster SP
- Bedingung einhalten $\Leftrightarrow$ Reihenfolge SP entlang jeder Geraden
- Auswahl? Begrenzungen ermitteln
- Wodurch wird Gerade nach rechts begrenzt? Liste N



$L=(1,2,5,3,4)$

Sweep mit Pseudogeraden

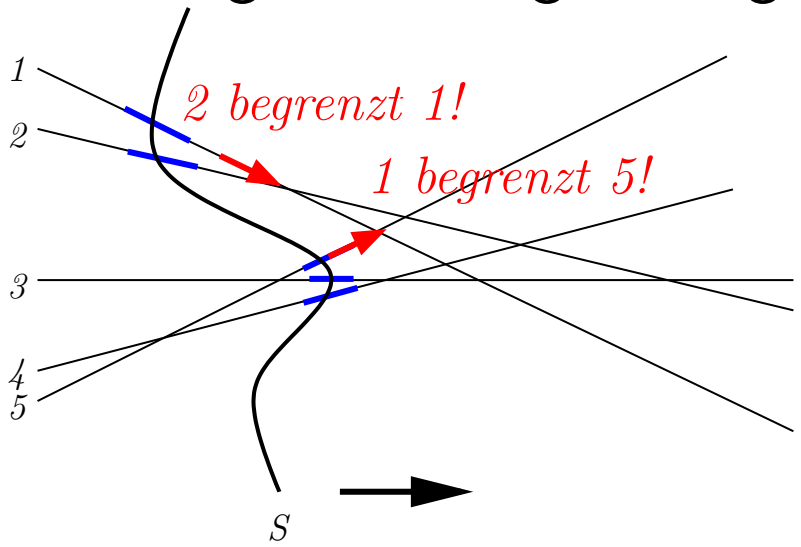
- Jede Gerade genau einmal schneiden, nächster SP
- Bedingung einhalten $\Leftrightarrow$ Reihenfolge SP entlang jeder Geraden
- Auswahl? Begrenzungen ermitteln
- Wodurch wird Gerade nach rechts begrenzt? Liste N
- Kandidaten mit gleicher Begrenzung sind genau richtig!!



$L=(1,2,5,3,4)$ $N=((1,2),(2,1),(5,1),(3,4),(4,3))$

Sweep mit Pseudogeraden

- Jede Gerade genau einmal schneiden, nächster SP
- Bedingung einhalten $\Leftrightarrow$ Reihenfolge SP entlang jeder Geraden
- Auswahl? Begrenzungen ermitteln
- Wodurch wird Gerade nach rechts begrenzt? Liste N
- Kandidaten mit gleicher Begrenzung sind genau richtig!!

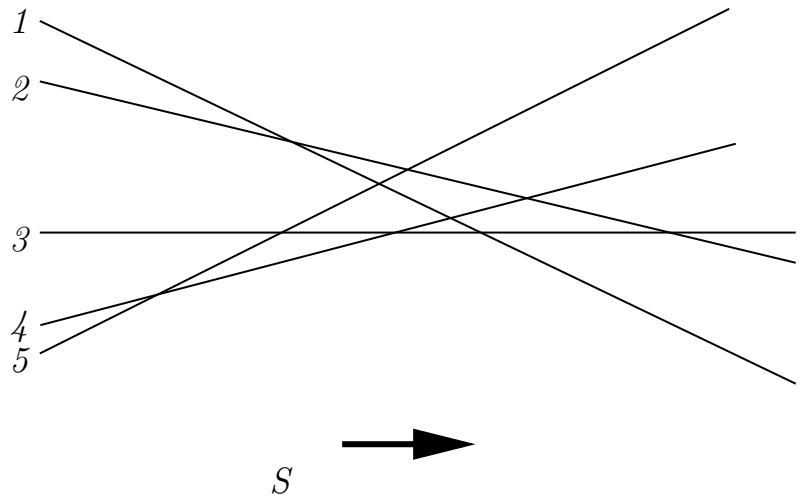


$L=(1,2,5,3,4)$ $N=((1,2),(2,1),(5,1),(3,4),(4,3))$

Neue Aufgabe: Begrenzungen ermitteln!

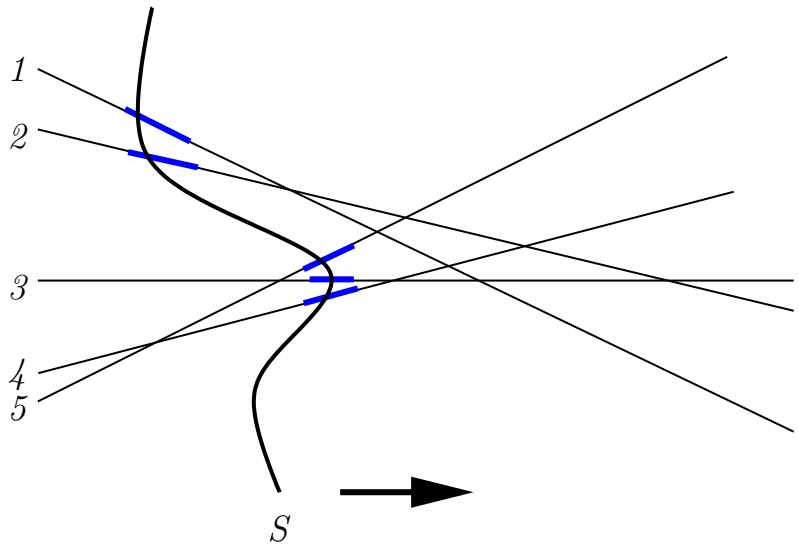
Neue Aufgabe: Begrenzungen ermitteln!

- Kandidatenliste N!



Neue Aufgabe: Begrenzungen ermitteln!

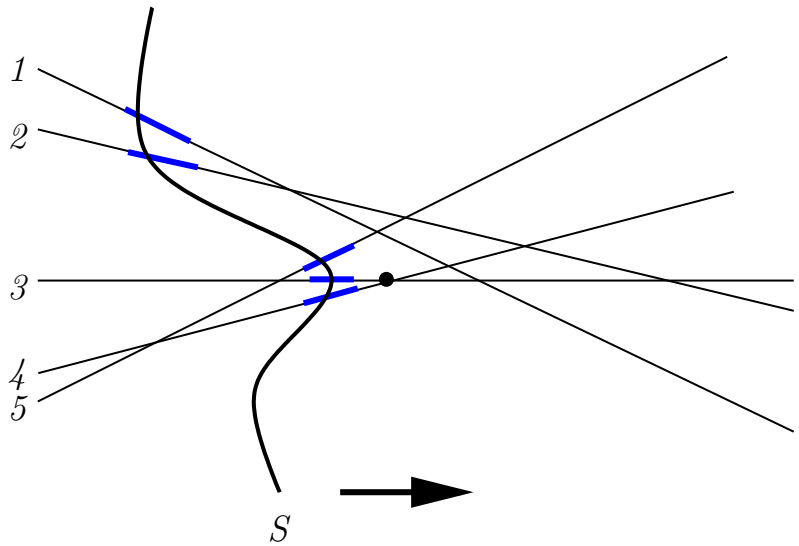
- Kandidatenliste N!
- Paare mit gleicher Beschränkung speziell markieren (Zeiger)



$N=((1,2),(2,1),(5,1),(3,4),(4,3))$

Neue Aufgabe: Begrenzungen ermitteln!

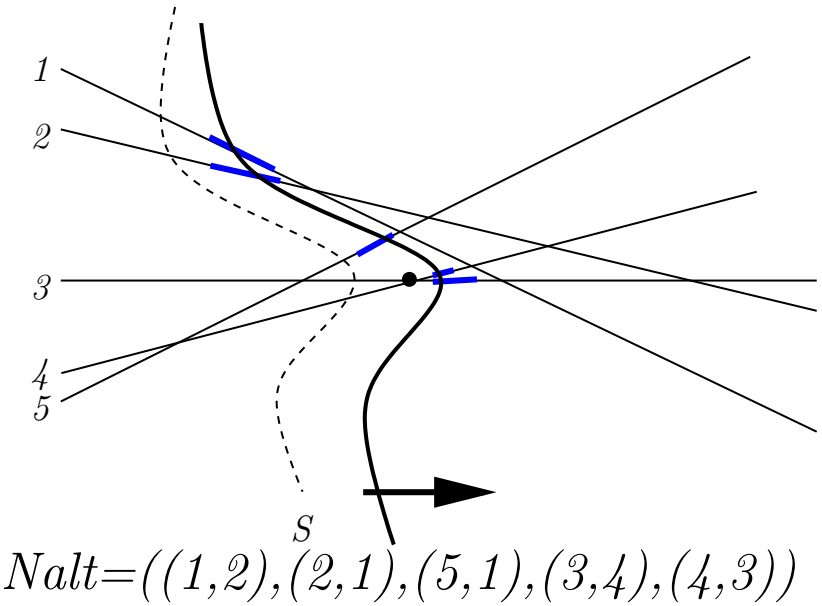
- Kandidatenliste N!
- Paare mit gleicher Beschränkung speziell markieren (Zeiger)
- Liste N aktualisieren bei nächstem Vorgang



$N=((1,2),(2,1),(5,1),(3,4),(4,3))$

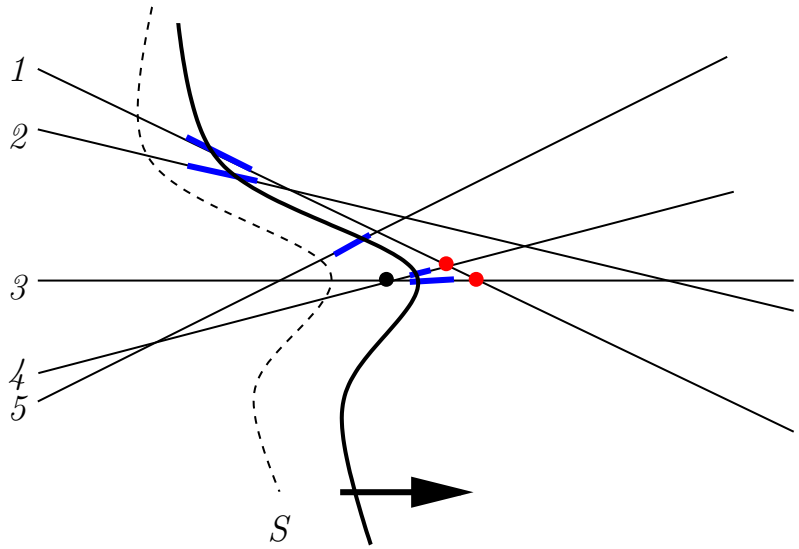
Neue Aufgabe: Begrenzungen ermitteln!

- Kandidatenliste N!
- Paare mit gleicher Beschränkung speziell markieren (Zeiger)
- Liste N aktualisieren bei nächstem Vorgang
- Neue Beschränkung zwei Geraden, mit spezieller Datenstruktur



Neue Aufgabe: Begrenzungen ermitteln!

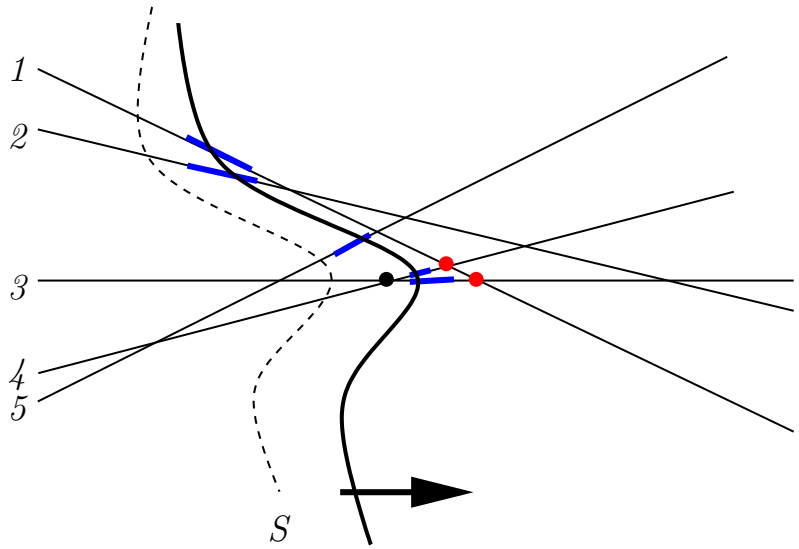
- Kandidatenliste N!
- Paare mit gleicher Beschränkung speziell markieren (Zeiger)
- Liste N aktualisieren bei nächstem Vorgang
- Neue Beschränkung zwei Geraden, mit spezieller Datenstruktur



$Nalt=((1,2),(2,1),(5,1),(3,4),(4,3))$

Neue Aufgabe: Begrenzungen ermitteln!

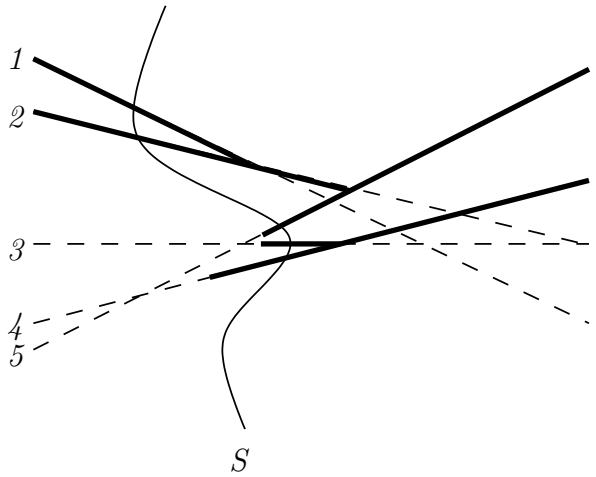
- Kandidatenliste N!
- Paare mit gleicher Beschränkung speziell markieren (Zeiger)
- Liste N aktualisieren bei nächstem Vorgang
- Neue Beschränkung zwei Geraden, mit spezieller Datenstruktur



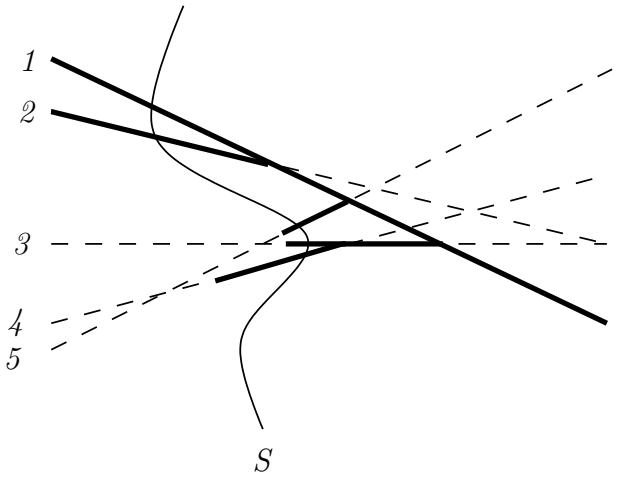
$Nalt=((1,2),(2,1),(5,1),(3,4),(4,3))$

$N=((1,2),(2,1),(5,1),(4,1),(3,1))$

Datenstruktur Horizontbäume



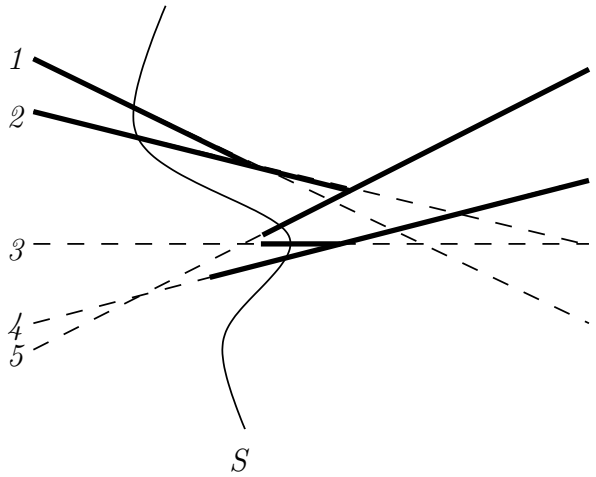
Oberer Horizontbaum



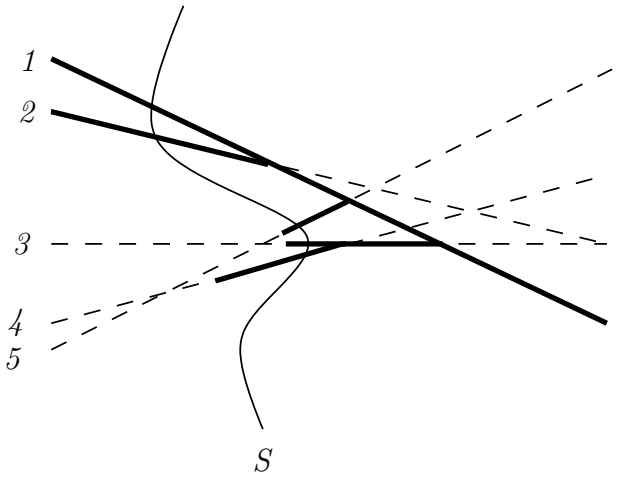
Unterer Horizontbaum

Datenstruktur Horizontbäume

- Arrangement und Pseudogerade mit Ordnung



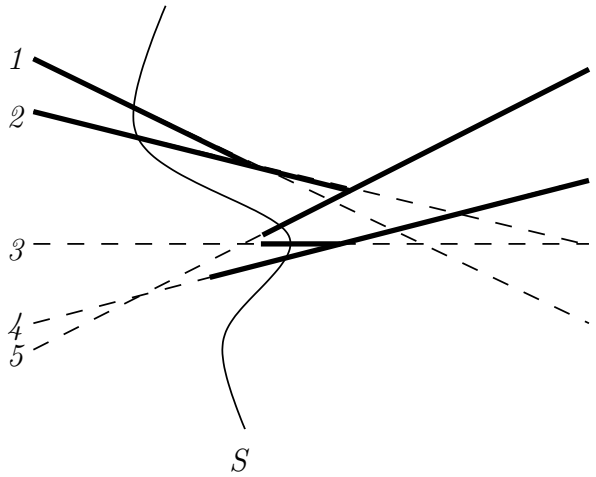
Oberer Horizontbaum



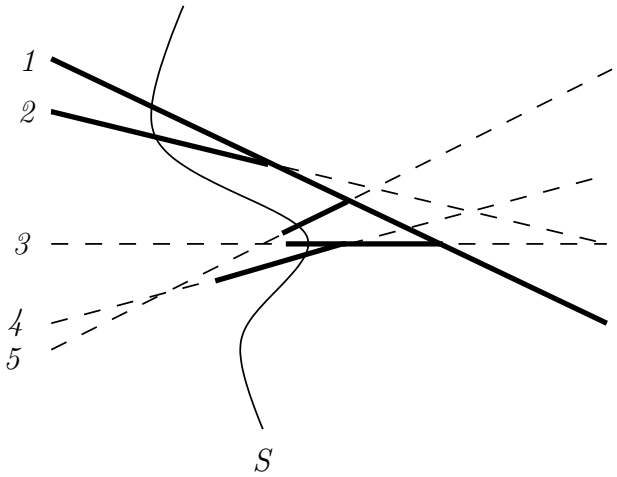
Unterer Horizontbaum

Datenstruktur Horizontbäume

- Arrangement und Pseudogerade mit Ordnung
- Verlängere die Geraden stets noch Oben/Unten



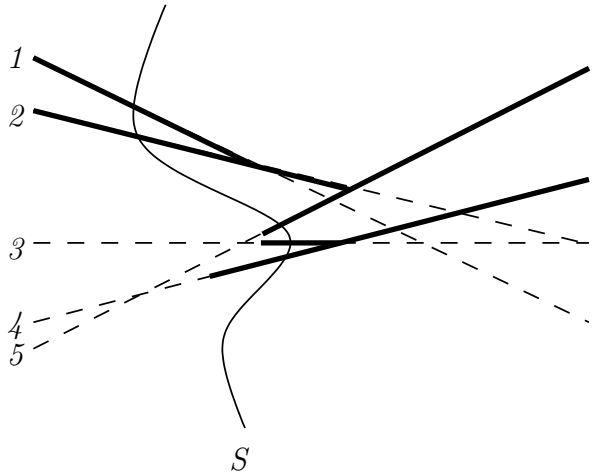
Oberer Horizontbaum



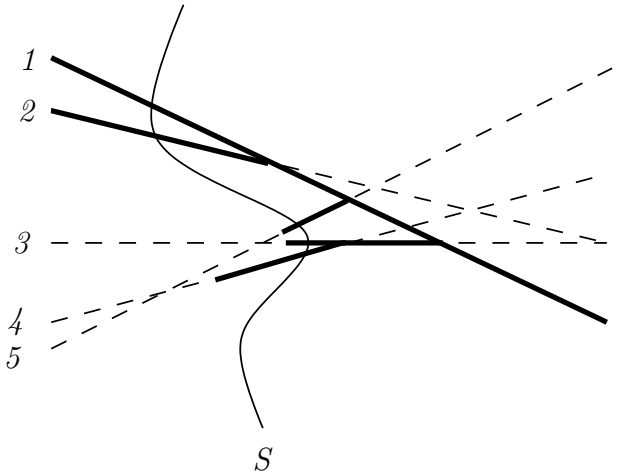
Unterer Horizontbaum

Datenstruktur Horizontbäume

- Arrangement und Pseudogerade mit Ordnung
- Verlängere die Geraden stets noch Oben/Unten
- Immer an nächstem Schnittpunkt



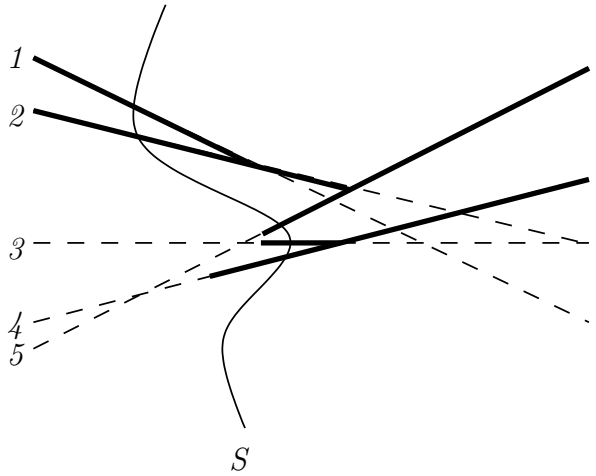
Oberer Horizontbaum



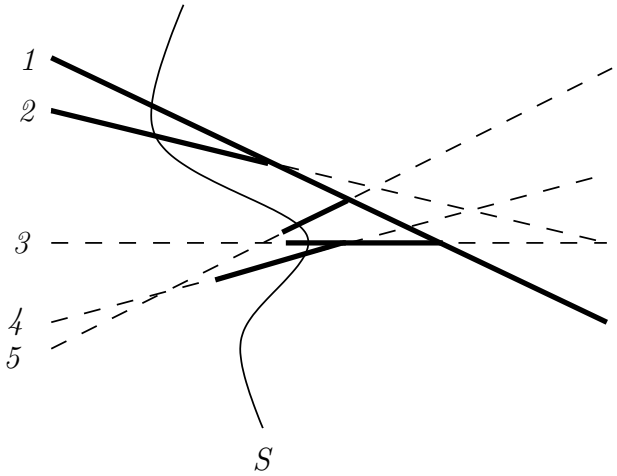
Unterer Horizontbaum

Datenstruktur Horizontbäume

- Arrangement und Pseudogerade mit Ordnung
- Verlängere die Geraden stets noch Oben/Unten
- Immer an nächstem Schnittpunkt
- Baumstruktur



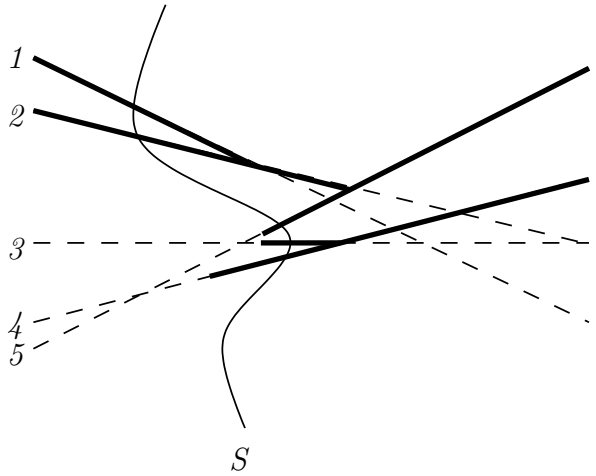
Oberer Horizontbaum



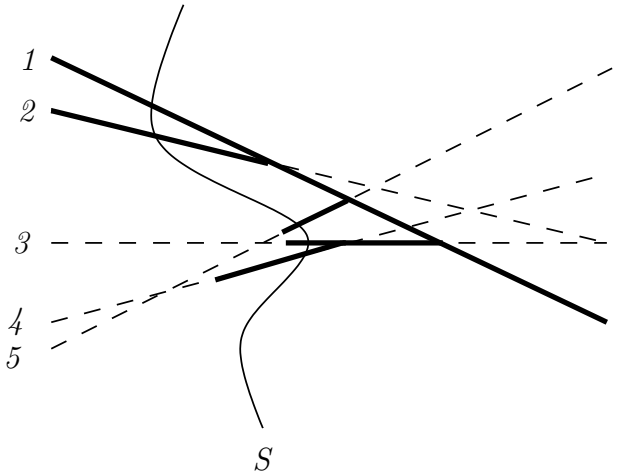
Unterer Horizontbaum

Datenstruktur Horizontbäume

- Arrangement und Pseudogerade mit Ordnung
- Verlängere die Geraden stets noch Oben/Unten
- Immer an nächstem Schnittpunkt
- Baumstruktur (kann zerfallen)



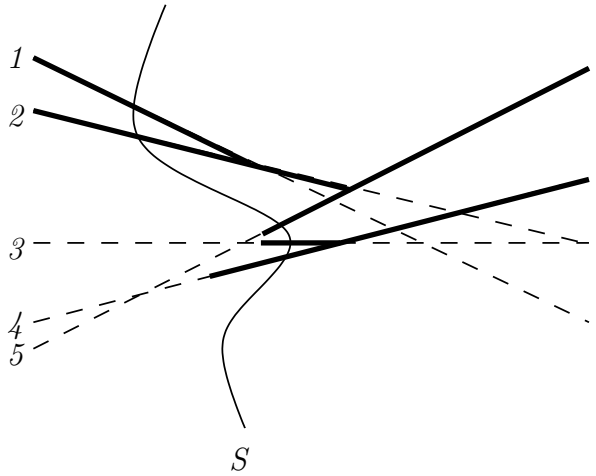
Oberer Horizontbaum



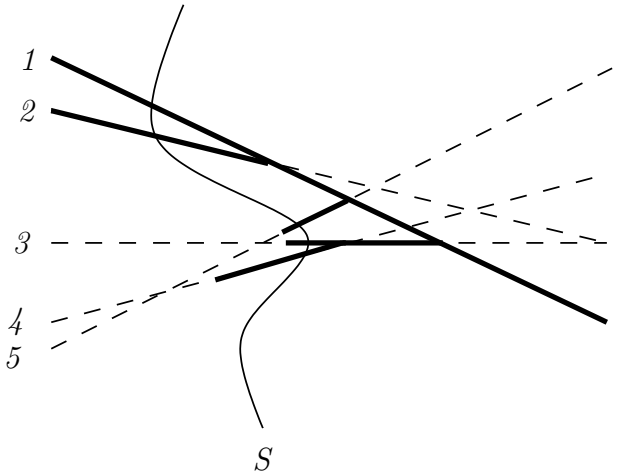
Unterer Horizontbaum

Datenstruktur Horizontbäume

- Arrangement und Pseudogerade mit Ordnung
- Verlängere die Geraden stets noch Oben/Unten
- Immer an nächstem Schnittpunkt
- Baumstruktur (kann zerfallen) Lineare Komplexität



Oberer Horizontbaum

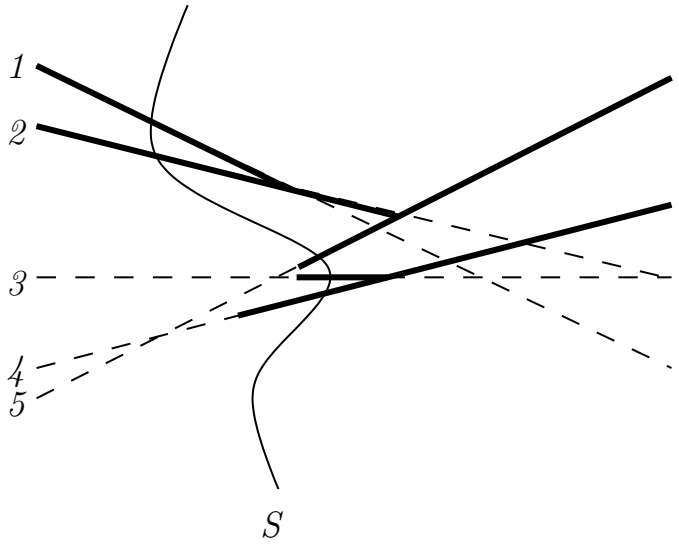


Unterer Horizontbaum

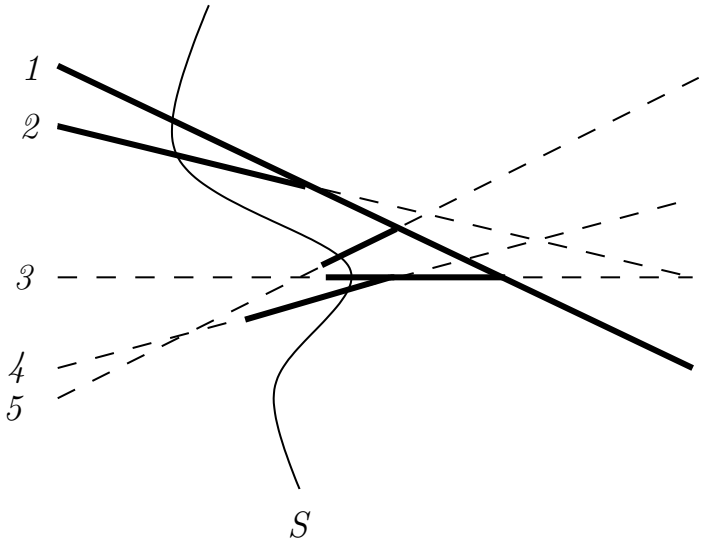
Neue Begrenzungen durch Horizontbäume ermitteln

Neue Begrenzungen durch Horizontbäume ermitteln

- Oberer Baum/Unterer Baum gegeben



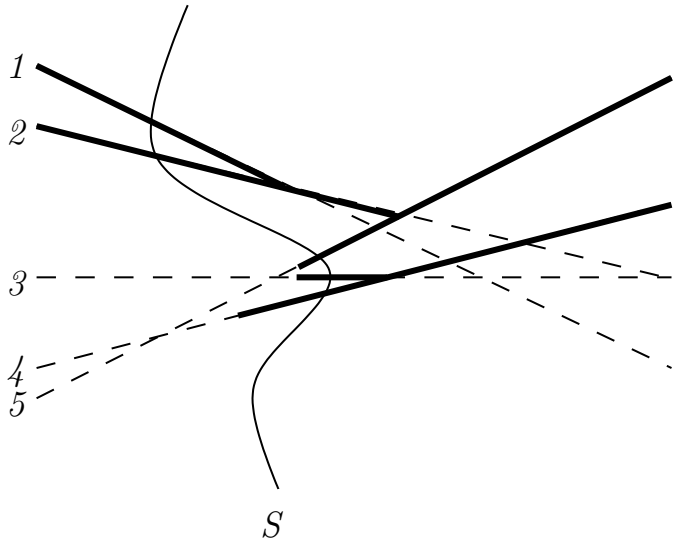
Oberer Horizontbaum



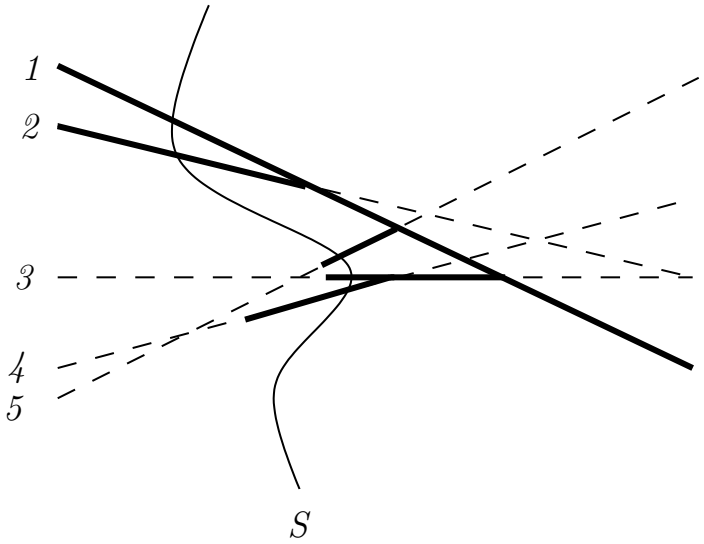
Unterer Horizontbaum

Neue Begrenzungen durch Horizontbäume ermitteln

- Oberer Baum/Unterer Baum gegeben
- *Minimum* ergibt begrenzende Gerade (Label im Baum)



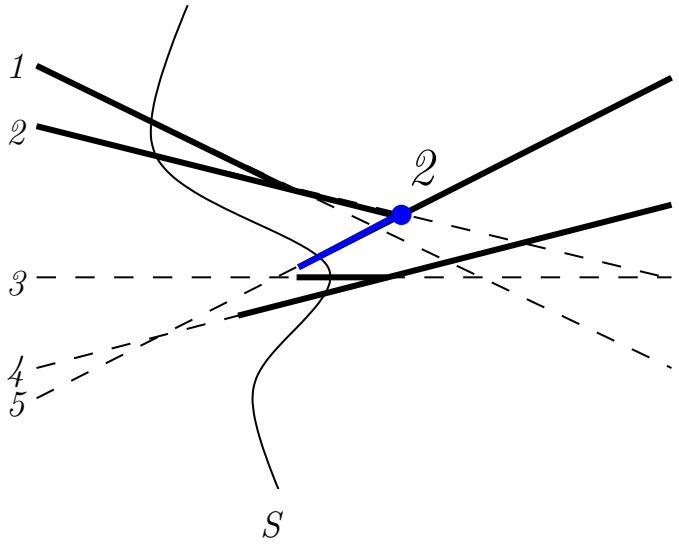
Oberer Horizontbaum



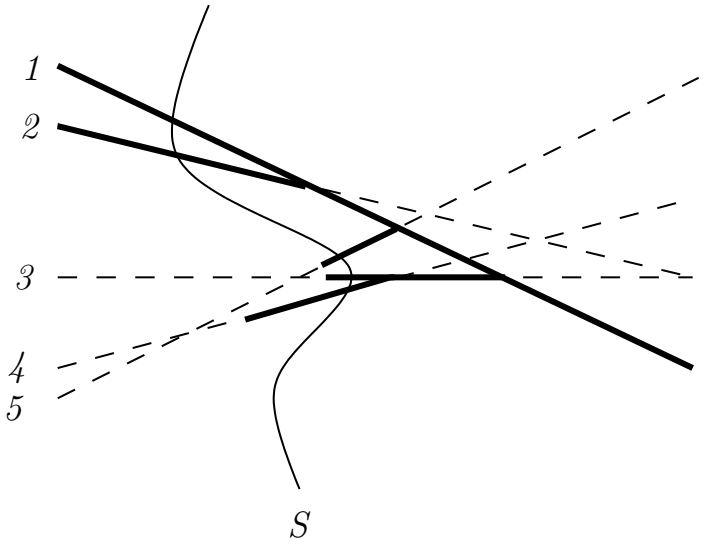
Unterer Horizontbaum

Neue Begrenzungen durch Horizontbäume ermitteln

- Oberer Baum/Unterer Baum gegeben
- *Minimum* ergibt begrenzende Gerade (Label im Baum)



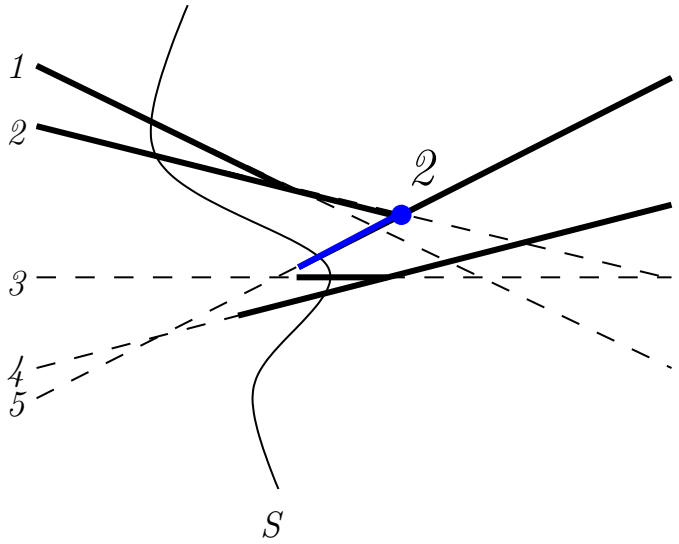
Oberer Horizontbaum



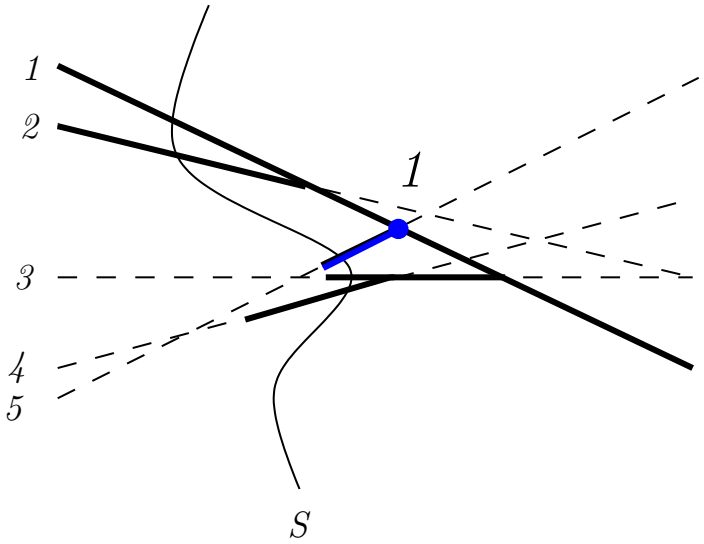
Unterer Horizontbaum

Neue Begrenzungen durch Horizontbäume ermitteln

- Oberer Baum/Unterer Baum gegeben
- *Minimum* ergibt begrenzende Gerade (Label im Baum)
- Bezüglich neuer Kanten den Bäumen folgen



Oberer Horizontbaum

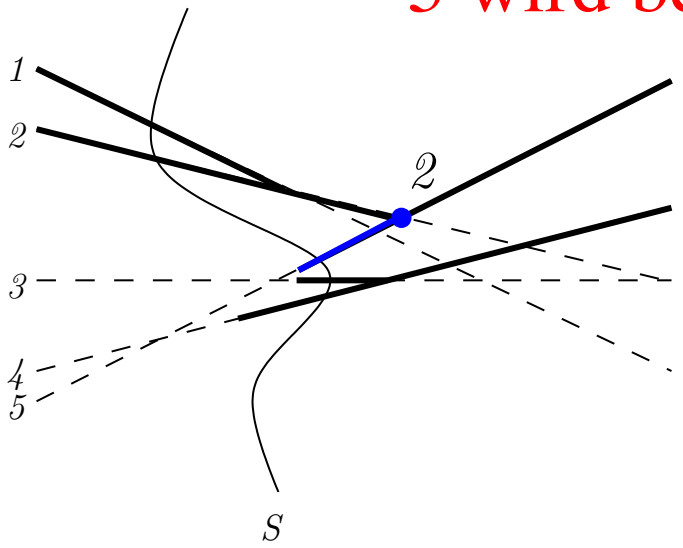


Unterer Horizontbaum

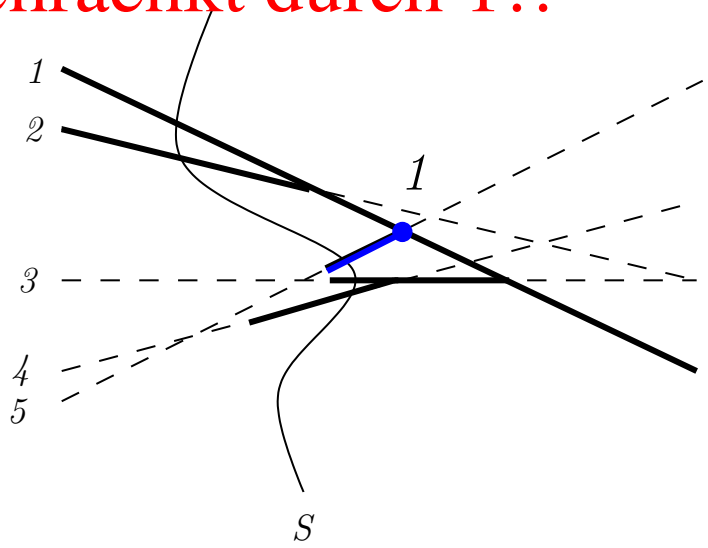
Neue Begrenzungen durch Horizontbäume ermitteln

- Oberer Baum/Unterer Baum gegeben
- *Minimum* ergibt begrenzende Gerade (Label im Baum)
- Bezüglich neuer Kanten den Bäumen folgen

5 wird beschraenkt durch 1!!



Oberer Horizontbaum

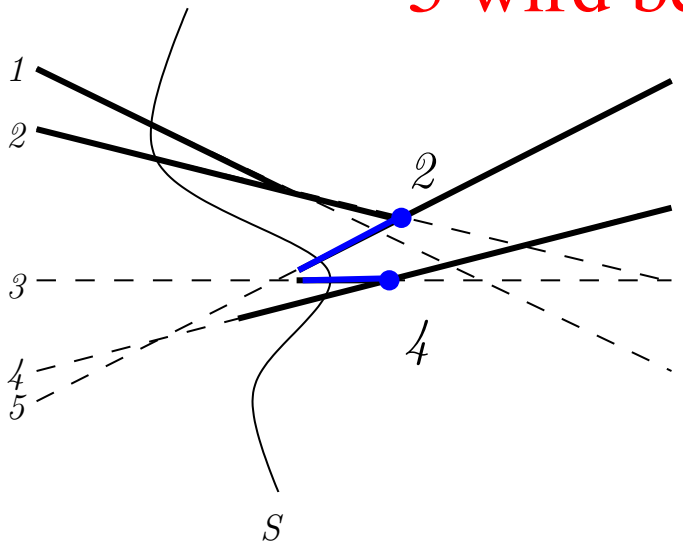


Unterer Horizontbaum

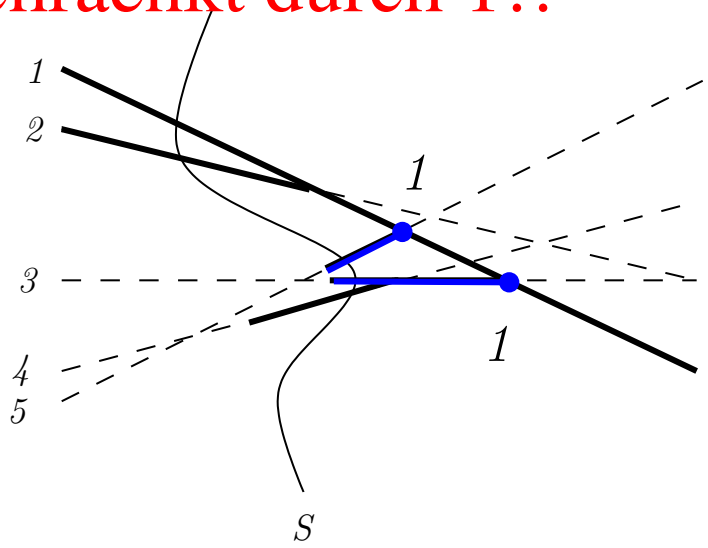
Neue Begrenzungen durch Horizontbäume ermitteln

- Oberer Baum/Unterer Baum gegeben
- *Minimum* ergibt begrenzende Gerade (Label im Baum)
- Bezüglich neuer Kanten den Bäumen folgen

5 wird beschraenkt durch 1!!



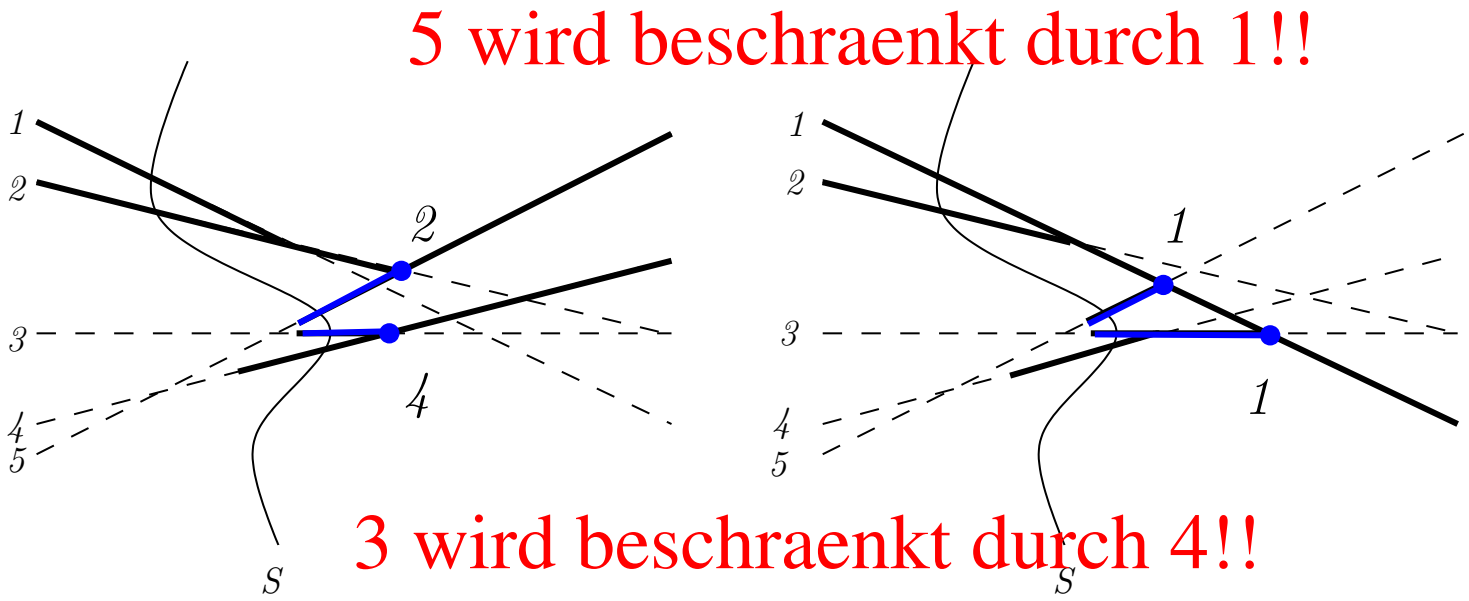
Oberer Horizontbaum



Unterer Horizontbaum

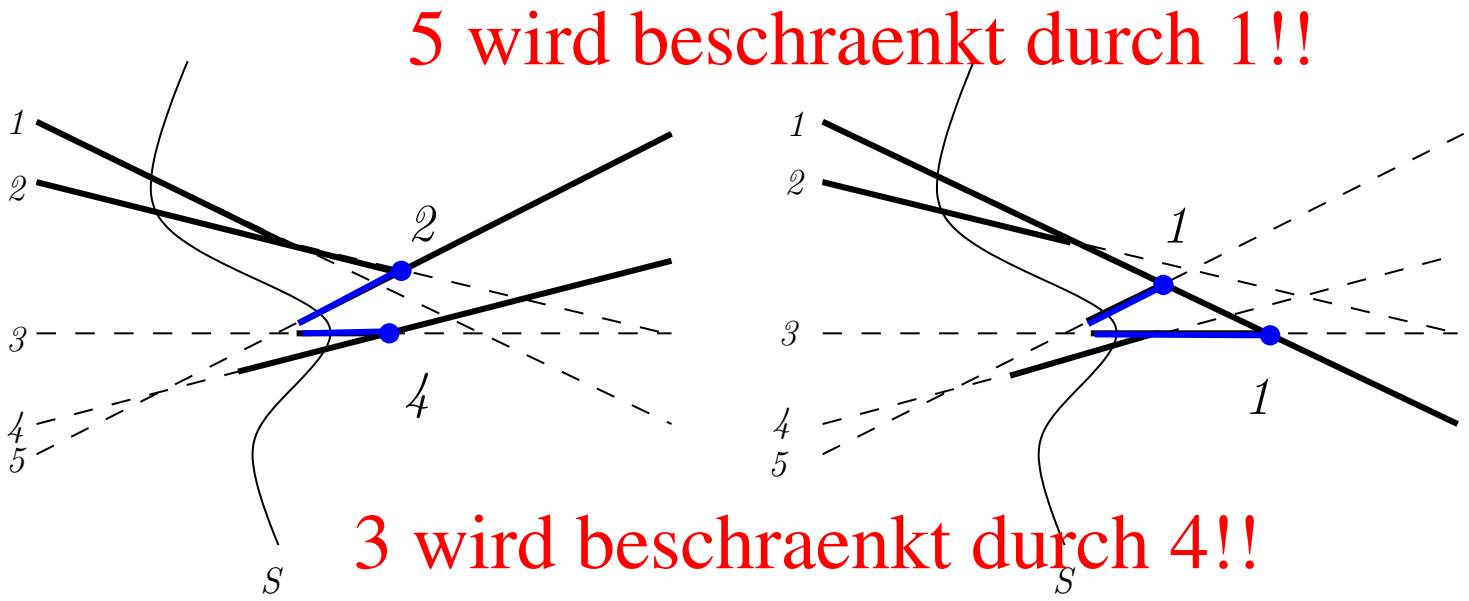
Neue Begrenzungen durch Horizontbäume ermitteln

- Oberer Baum/Unterer Baum gegeben
- *Minimum* ergibt begrenzende Gerade (Label im Baum)
- Bezüglich neuer Kanten den Bäumen folgen



Neue Begrenzungen durch Horizontbäume ermitteln

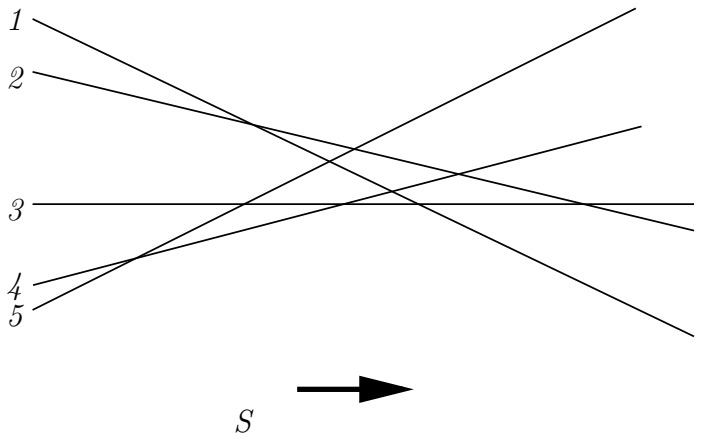
- Oberer Baum/Unterer Baum gegeben
- *Minimum* ergibt begrenzende Gerade (Label im Baum)
- Bezüglich neuer Kanten den Bäumen folgen



Neue Aufgabe: Begrenzungen ermitteln!

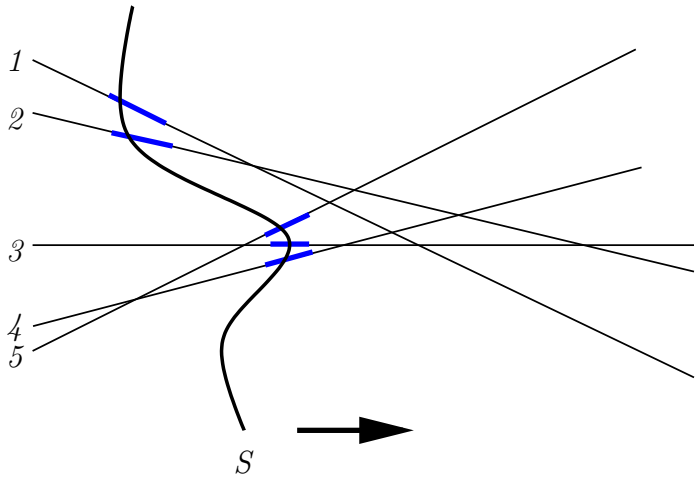
Neue Aufgabe: Begrenzungen ermitteln!

- Kandidatenliste N!



Neue Aufgabe: Begrenzungen ermitteln!

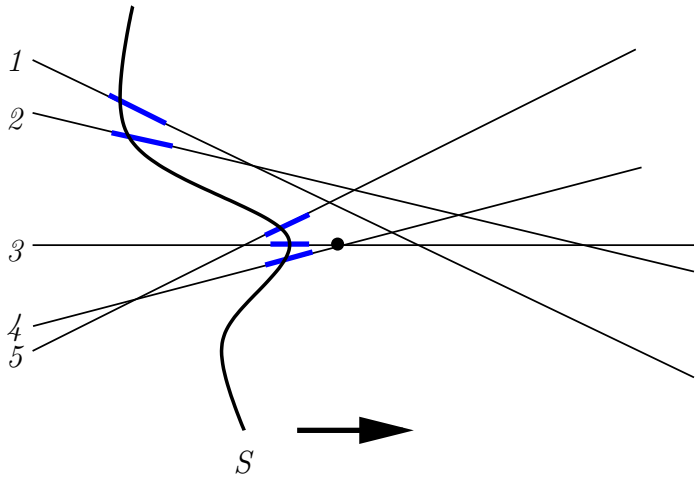
- Kandidatenliste N!
- Paare mit gleicher Beschränkung speziell markieren (Zeiger)



$N=((1,2),(2,1),(5,1),(3,4),(4,3))$

Neue Aufgabe: Begrenzungen ermitteln!

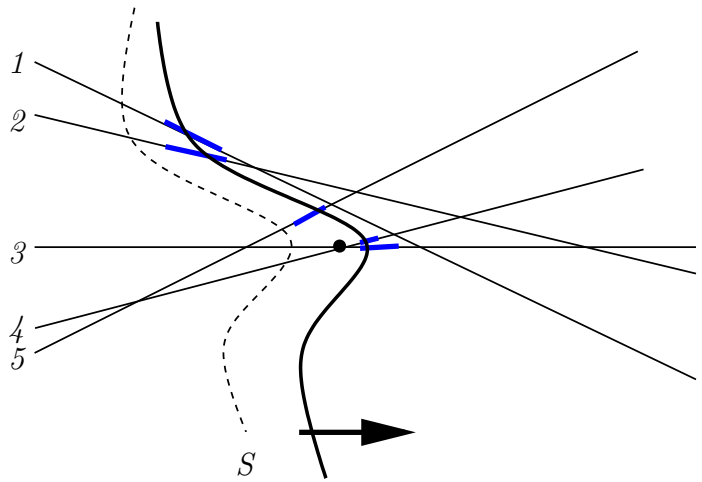
- Kandidatenliste N!
- Paare mit gleicher Beschränkung speziell markieren (Zeiger)
- Liste N aktualisieren bei nächstem Vorgang



$N=((1,2),(2,1),(5,1),(3,4),(4,3))$

Neue Aufgabe: Begrenzungen ermitteln!

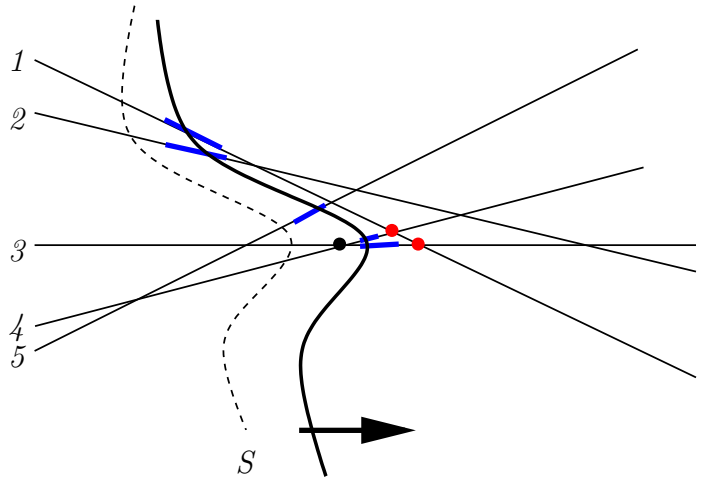
- Kandidatenliste N!
- Paare mit gleicher Beschränkung speziell markieren (Zeiger)
- Liste N aktualisieren bei nächstem Vorgang
- Neue Beschränkung zwei Geraden, mit spezieller Datenstruktur



$Nalt=((1,2),(2,1),(5,1),(3,4),(4,3))$

Neue Aufgabe: Begrenzungen ermitteln!

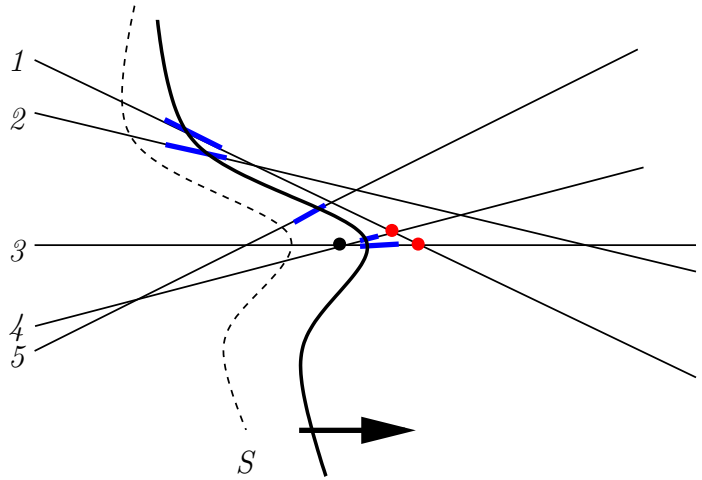
- Kandidatenliste N!
- Paare mit gleicher Beschränkung speziell markieren (Zeiger)
- Liste N aktualisieren bei nächstem Vorgang
- Neue Beschränkung zwei Geraden, mit spezieller Datenstruktur
- Durch Aktualisierung von T^o und T_u



$Nalt=((1,2),(2,1),(5,1),(3,4),(4,3))$

Neue Aufgabe: Begrenzungen ermitteln!

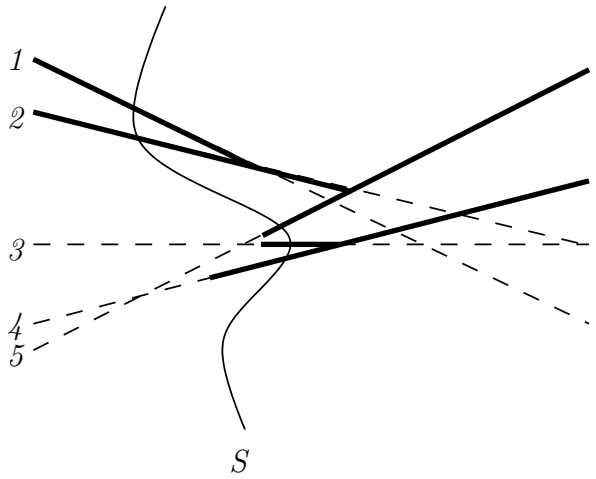
- Kandidatenliste N!
- Paare mit gleicher Beschränkung speziell markieren (Zeiger)
- Liste N aktualisieren bei nächstem Vorgang
- Neue Beschränkung zwei Geraden, mit spezieller Datenstruktur
- Durch Aktualisierung von T^o und T_u



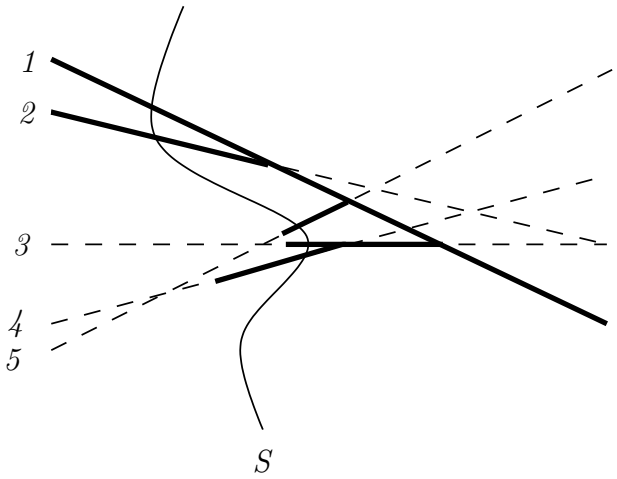
$Nalt=((1,2),(2,1),(5,1),(3,4),(4,3))$

$N=((1,2),(2,1),(5,1),(4,1),(3,1))$

Datenstruktur Horizontbäume



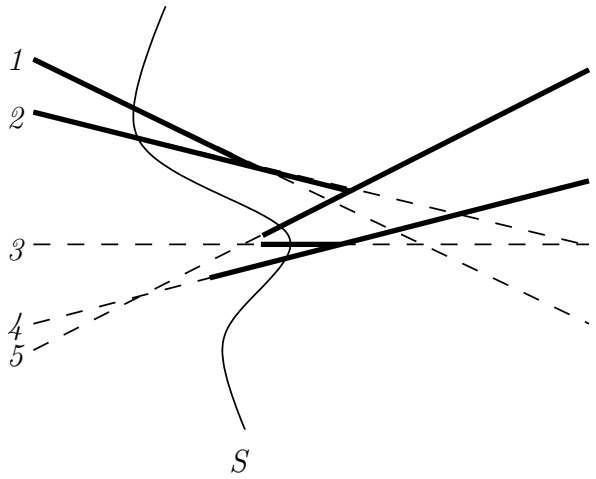
Oberer Horizontbaum



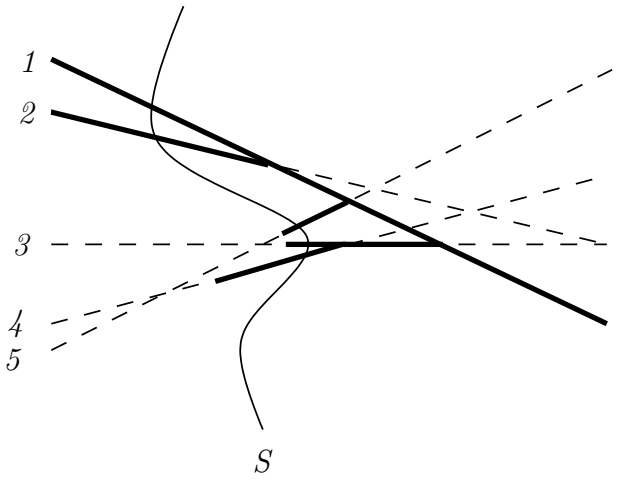
Unterer Horizontbaum

Datenstruktur Horizontbäume

- Für Pseudogerade mit Ordnung



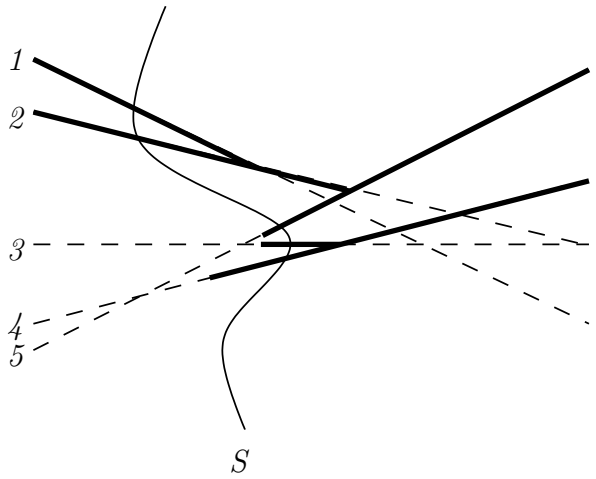
Oberer Horizontbaum



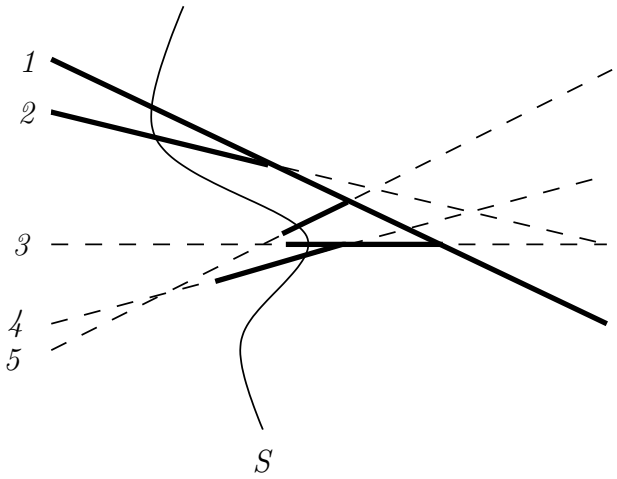
Unterer Horizontbaum

Datenstruktur Horizontbäume

- Für Pseudogerade mit Ordnung
- Oberer und Unterer Horizontbaum gegeben



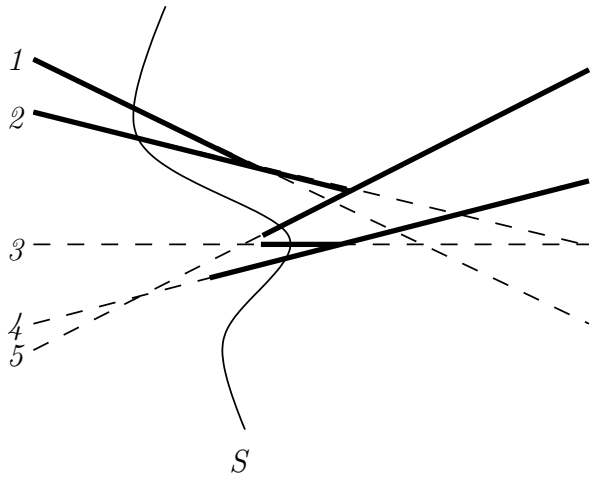
Oberer Horizontbaum



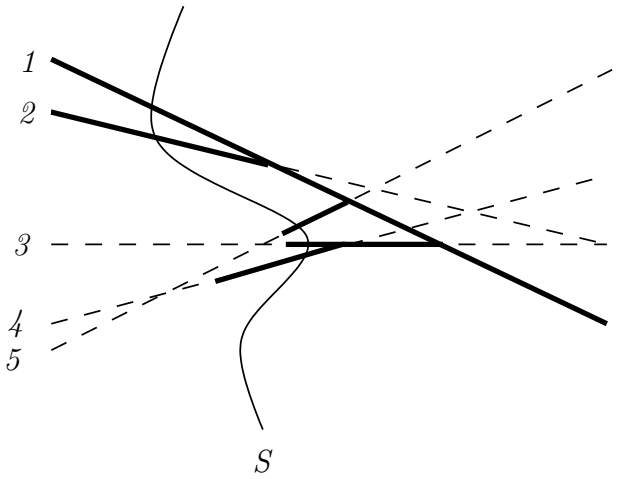
Unterer Horizontbaum

Datenstruktur Horizontbäume

- Für Pseudogerade mit Ordnung
- Oberer und Unterer Horizontbaum gegeben
- Hebe über nächsten erlaubten SP



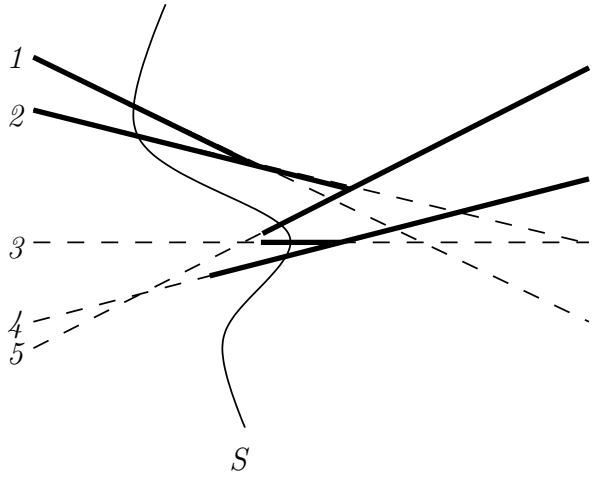
Oberer Horizontbaum



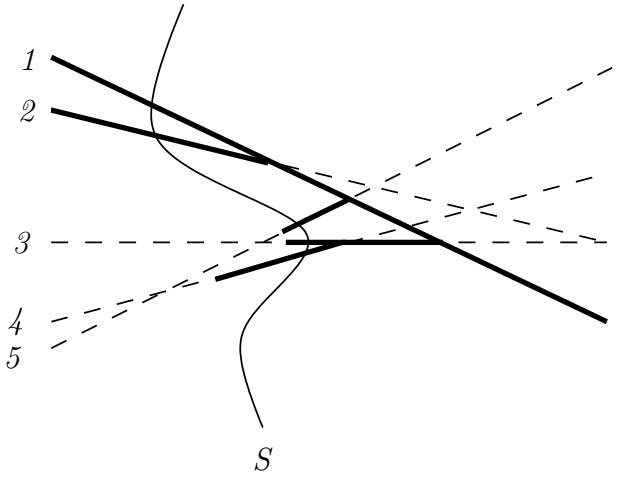
Unterer Horizontbaum

Datenstruktur Horizontbäume

- Für Pseudogerade mit Ordnung
- Oberer und Unterer Horizontbaum gegeben
- Hebe über nächsten erlaubten SP
- Aktualisiere die Bäume



Oberer Horizontbaum

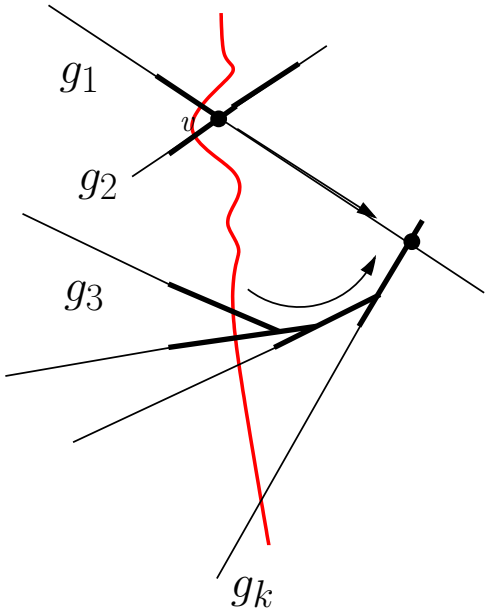


Unterer Horizontbaum

Kosten Aktualisierung T^o (T_u)

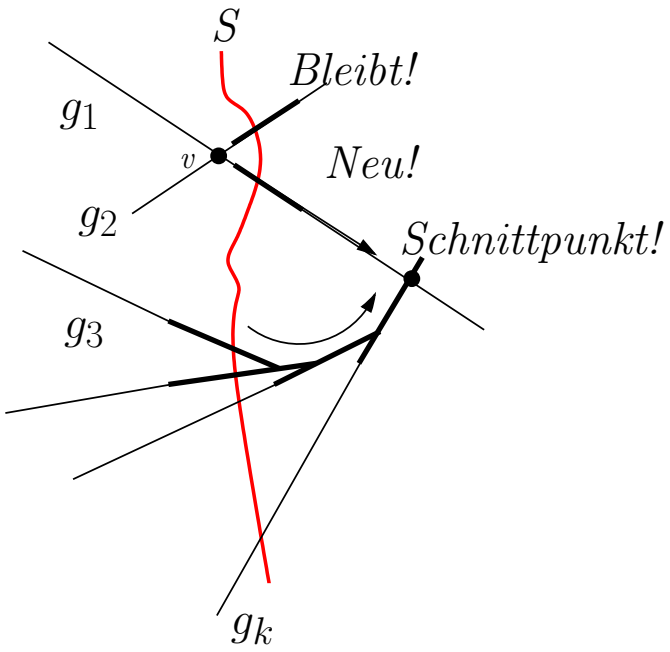
Kosten Aktualisierung T^o (T_u)

1. Nächster Knoten v!



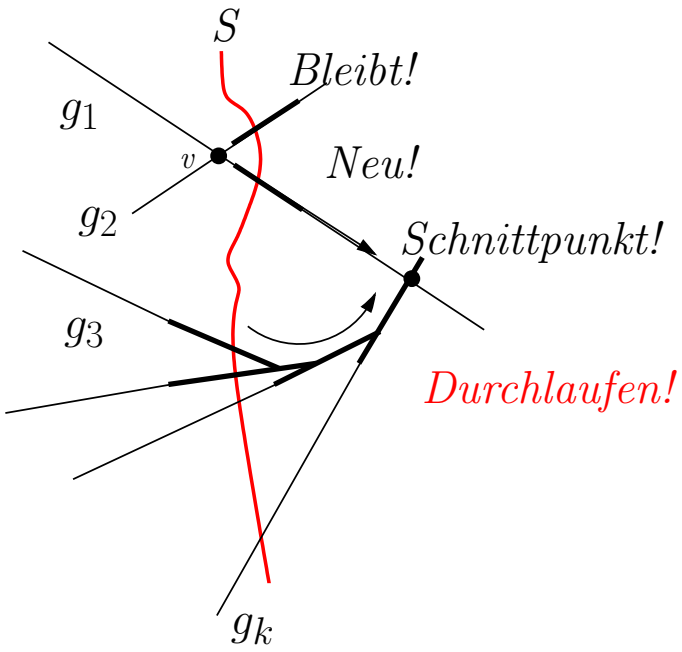
Kosten Aktualisierung T^o (T_u)

1. Nächster Knoten v!
2. Drüberheben



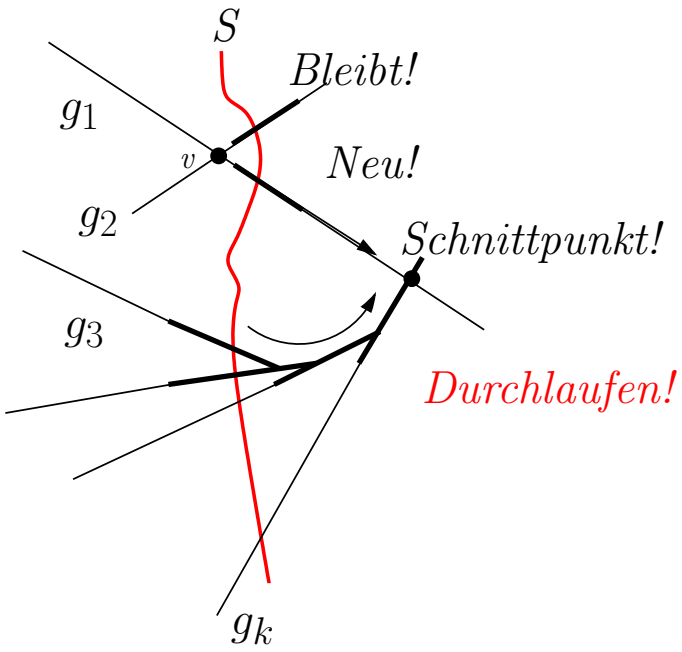
Kosten Aktualisierung T^o (T_u)

1. Nächster Knoten v!
2. Drüberheben
3. Schnittpunkt mit Kette darunter bestimmen



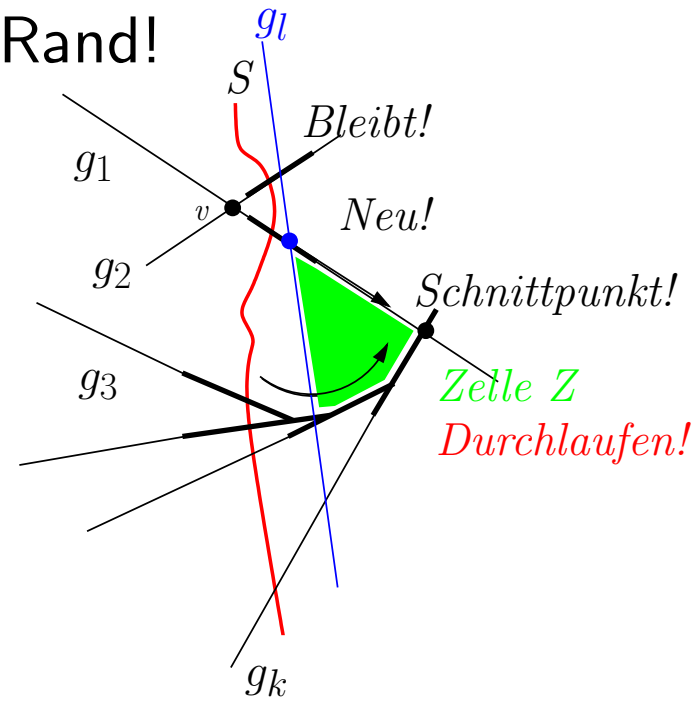
Kosten Aktualisierung T^o (T_u)

1. Nächster Knoten v!
2. Drüberheben
3. Schnittpunkt mit Kette darunter bestimmen
4. Einmal Kanten einer Zelle durchlaufen



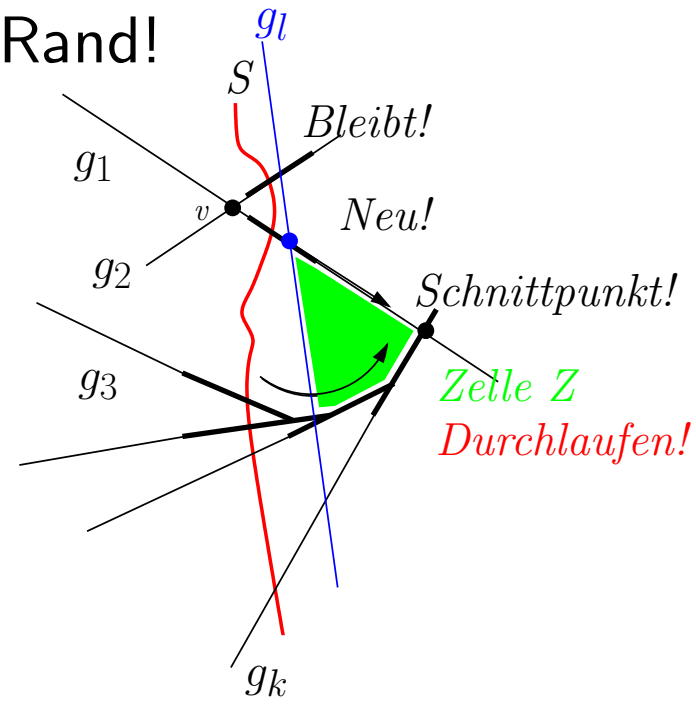
Kosten Aktualisierung T^o (T_u)

1. Nächster Knoten v!
2. Drüberheben
3. Schnittpunkt mit Kette darunter bestimmen
4. Einmal Kanten einer Zelle durchlaufen
5. Nicht alle auf dem Rand!



Kosten Aktualisierung T^o (T_u)

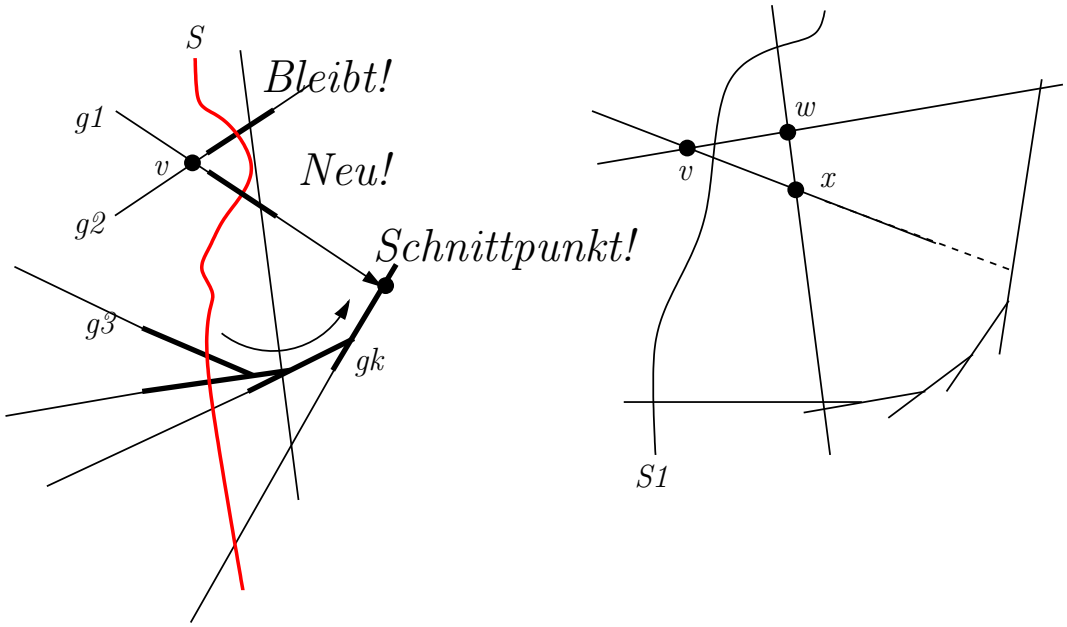
1. Nächster Knoten v!
2. Drüberheben
3. Schnittpunkt mit Kette darunter bestimmen
4. Einmal Kanten einer Zelle durchlaufen
5. Nicht alle auf dem Rand!



Gesamtkosten: Aktualisierung T^o (T_u)

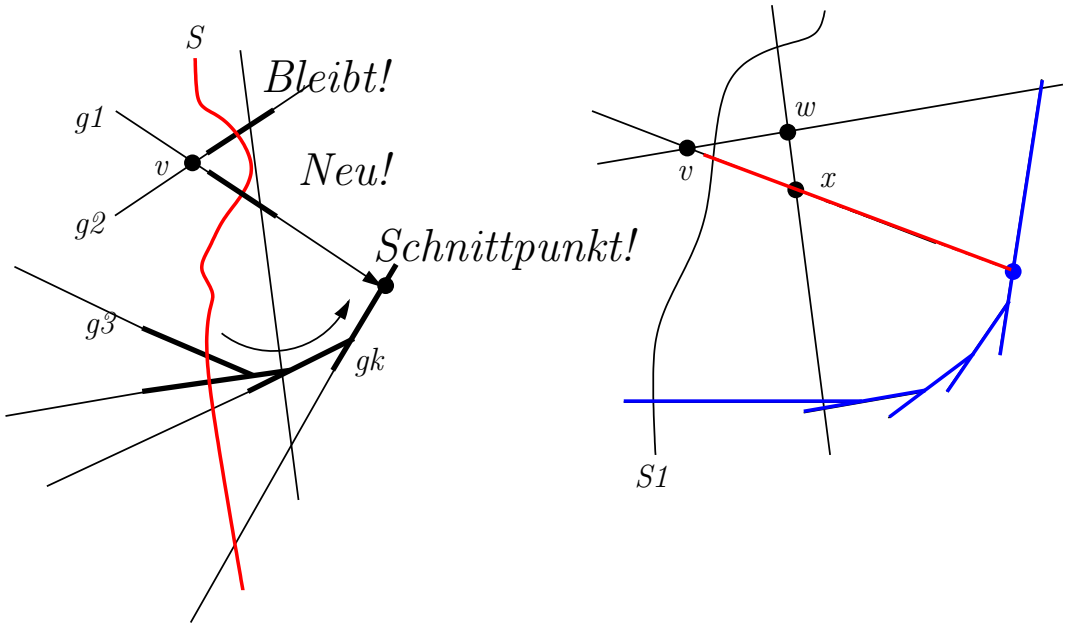
Gesamtkosten: Aktualisierung T^o (T_u)

1. Geschickte Zuordnung



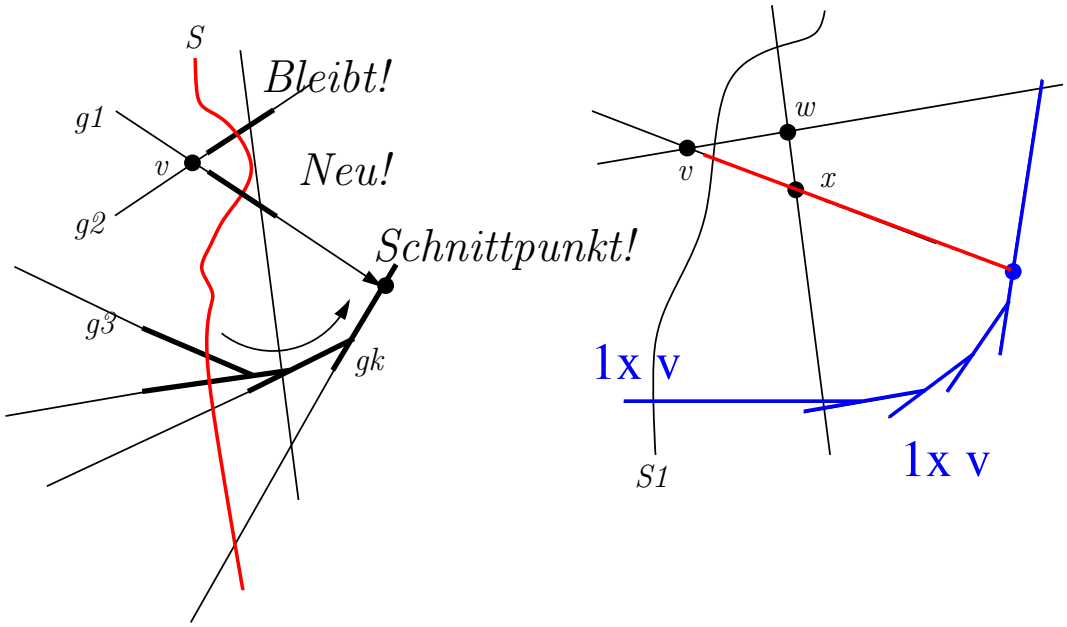
Gesamtkosten: Aktualisierung T^o (T_u)

1. Geschickte Zuordnung
2. Gesamtzählweise: Pro Knoten alle Kanten einer Zelle



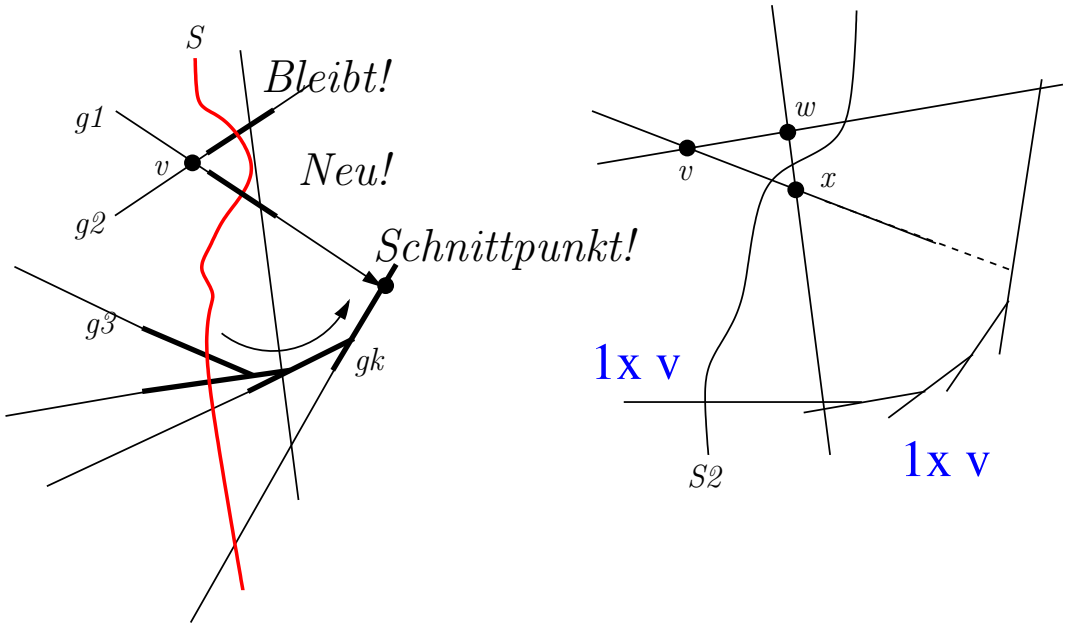
Gesamtkosten: Aktualisierung T^o (T_u)

1. Geschickte Zuordnung
2. Gesamtzählweise: Pro Knoten alle Kanten einer Zelle
3. $\sum$ Zelle $_Z$(Kanten Z) $\times$ (Knoten Z)



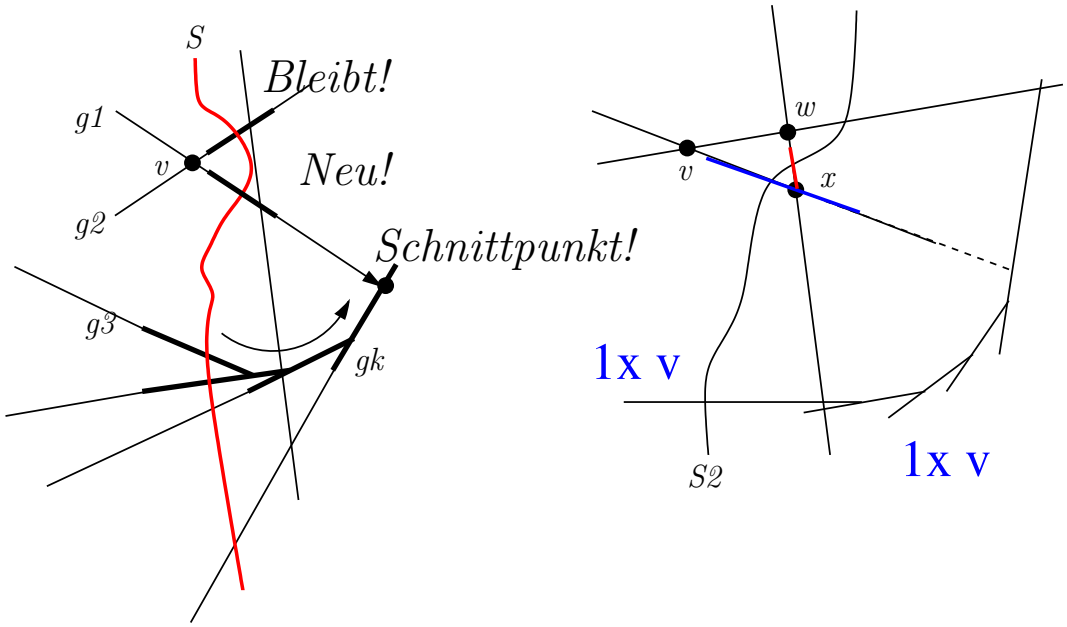
Gesamtkosten: Aktualisierung T^o (T_u)

1. Geschickte Zuordnung
2. Gesamtzählweise: Pro Knoten alle Kanten einer Zelle
3. $\sum_{\text{Zelle } Z}(\text{Kanten } Z) \times (\text{Knoten } Z)$



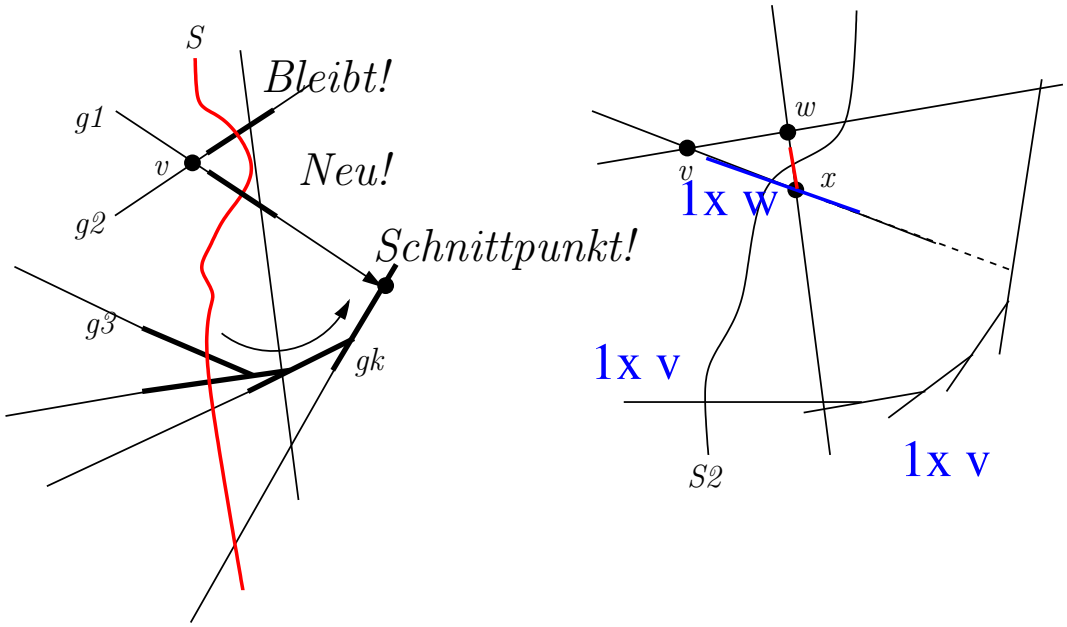
Gesamtkosten: Aktualisierung T^o (T_u)

1. Geschickte Zuordnung
2. Gesamtzählweise: Pro Knoten alle Kanten einer Zelle
3. $\sum_{\text{Zelle } Z}(\text{Kanten } Z) \times (\text{Knoten } Z)$



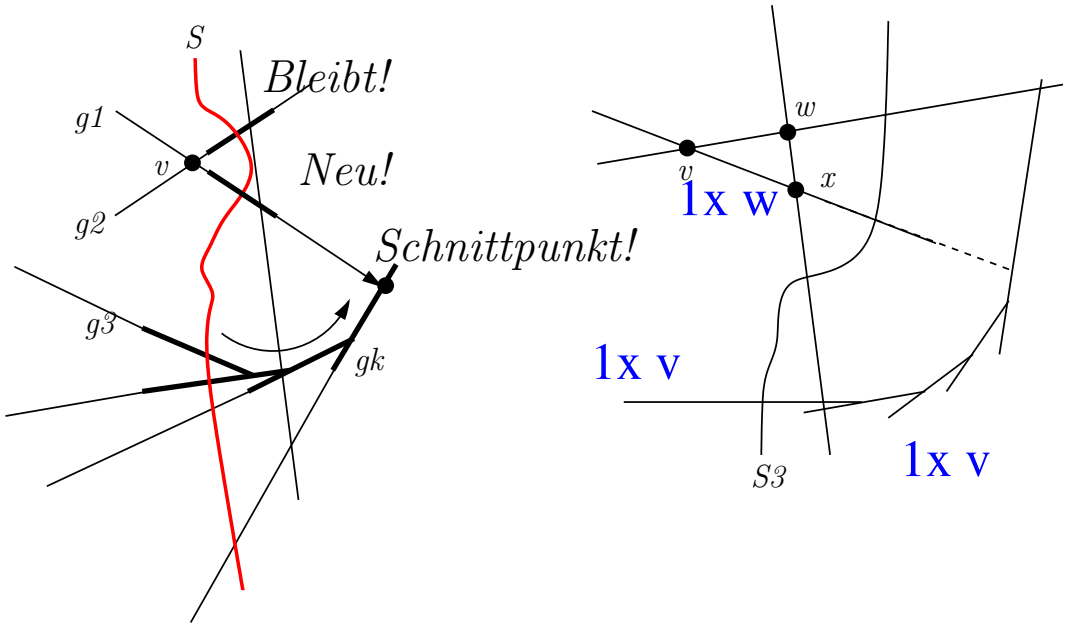
Gesamtkosten: Aktualisierung T^o (T_u)

1. Geschickte Zuordnung
2. Gesamtzählweise: Pro Knoten alle Kanten einer Zelle
3. $\sum_{\text{Zelle } Z}(\text{Kanten } Z) \times (\text{Knoten } Z)$



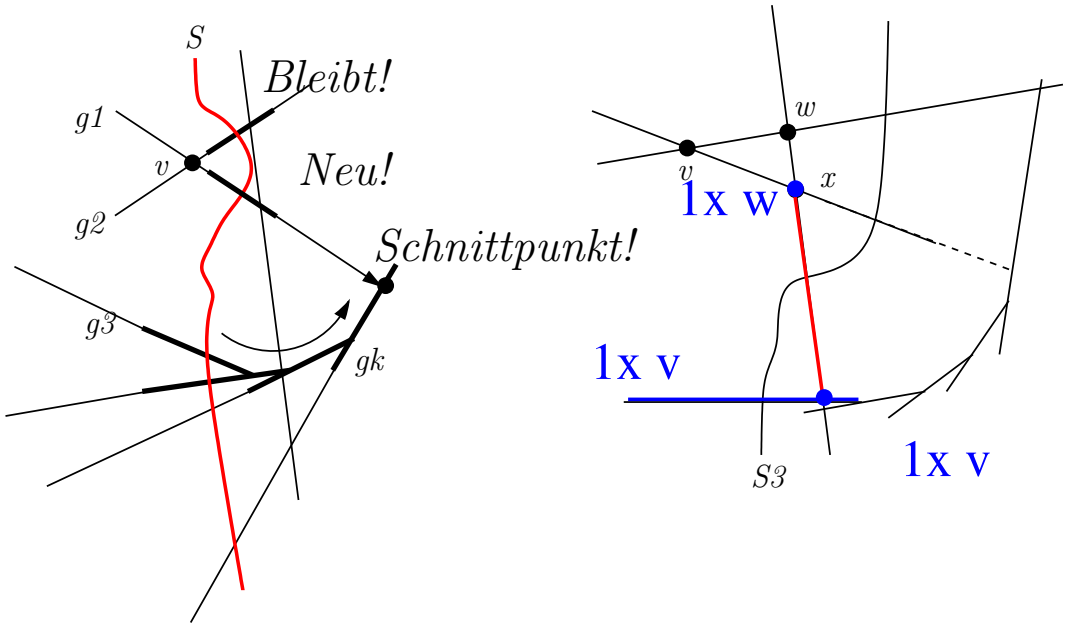
Gesamtkosten: Aktualisierung T^o (T_u)

1. Geschickte Zuordnung
2. Gesamtzählweise: Pro Knoten alle Kanten einer Zelle
3. $\sum_{\text{Zelle } Z}(\text{Kanten } Z) \times (\text{Knoten } Z)$



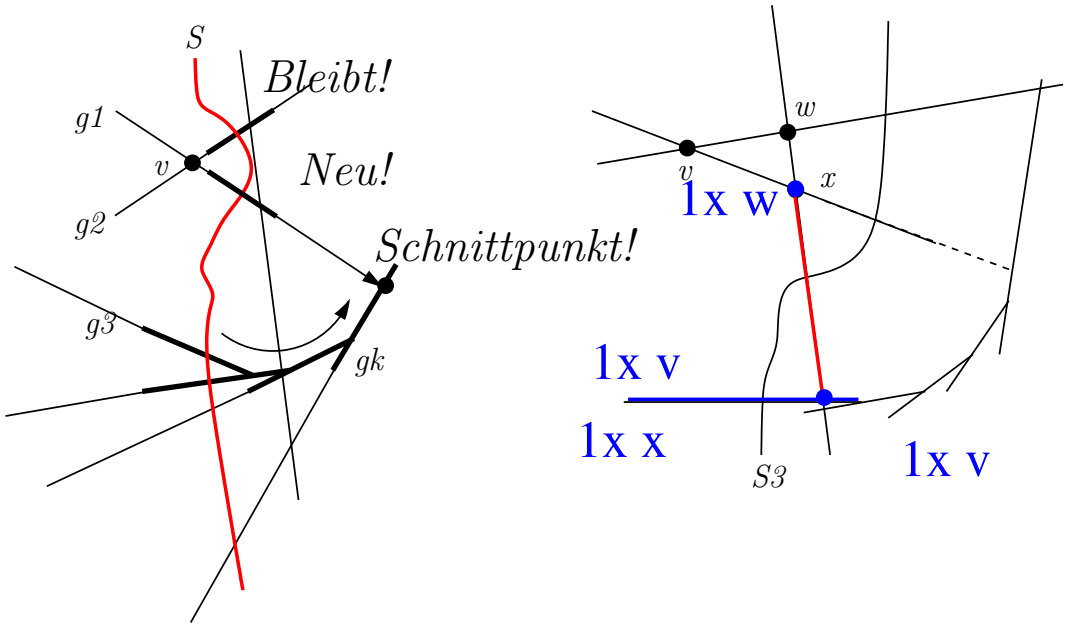
Gesamtkosten: Aktualisierung T^o (T_u)

1. Geschickte Zuordnung
2. Gesamtzählweise: Pro Knoten alle Kanten einer Zelle
3. $\sum_{\text{Zelle } Z}(\text{Kanten } Z) \times (\text{Knoten } Z)$



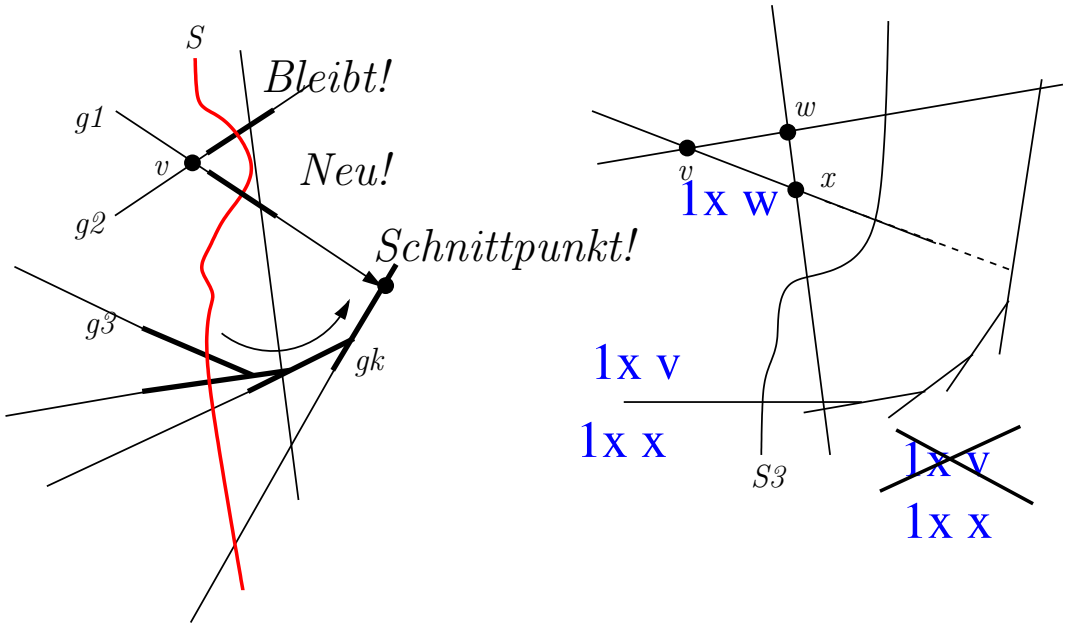
Gesamtkosten: Aktualisierung T^o (T_u)

1. Geschickte Zuordnung
2. Gesamtzählweise: Pro Knoten alle Kanten einer Zelle
3. $\sum_{\text{Zelle } Z}(\text{Kanten } Z) \times (\text{Knoten } Z)$



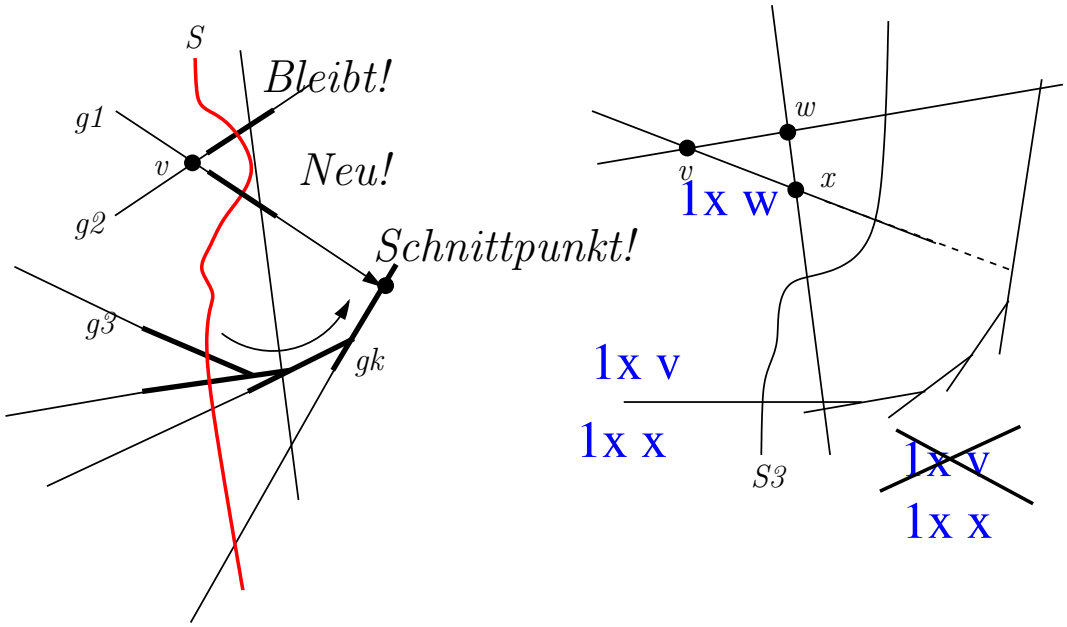
Gesamtkosten: Aktualisierung T^o (T_u)

1. Geschickte Zuordnung
2. Gesamtzählweise: Pro Knoten alle Kanten einer Zelle
3. $\sum$ Zelle $_Z$(Kanten Z) $\times$ (Knoten Z)



Gesamtkosten: Aktualisierung T^o (T_u)

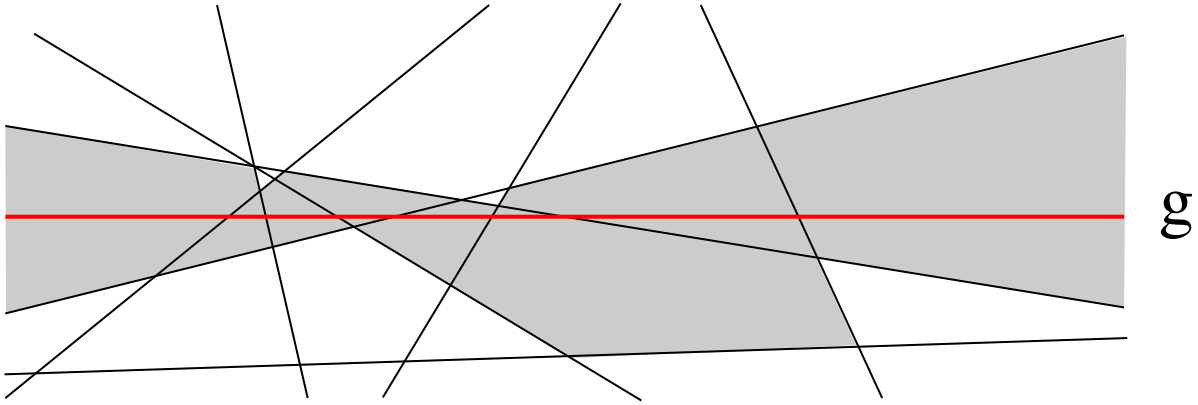
1. Geschickte Zuordnung
2. Gesamtzählweise: Pro Knoten alle Kanten einer Zelle
3. $\sum$ Zelle $_Z$(Kanten Z) $\times$ (Knoten Z)



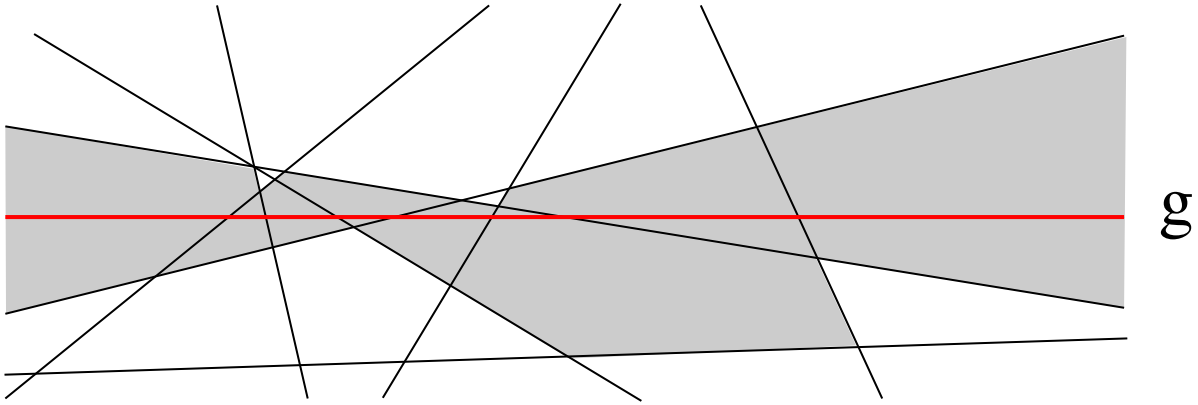
Andere Zählweise, bessere Zuordnung!

$$\sum_{\text{Zelle } Z} (\text{Kanten } Z) \times (\text{Knoten } Z) \leq$$

$$\sum_{\text{Gerade } g} \left(\sum_{g \text{ schneidet } Z} (\text{Kanten } Z) \right)$$

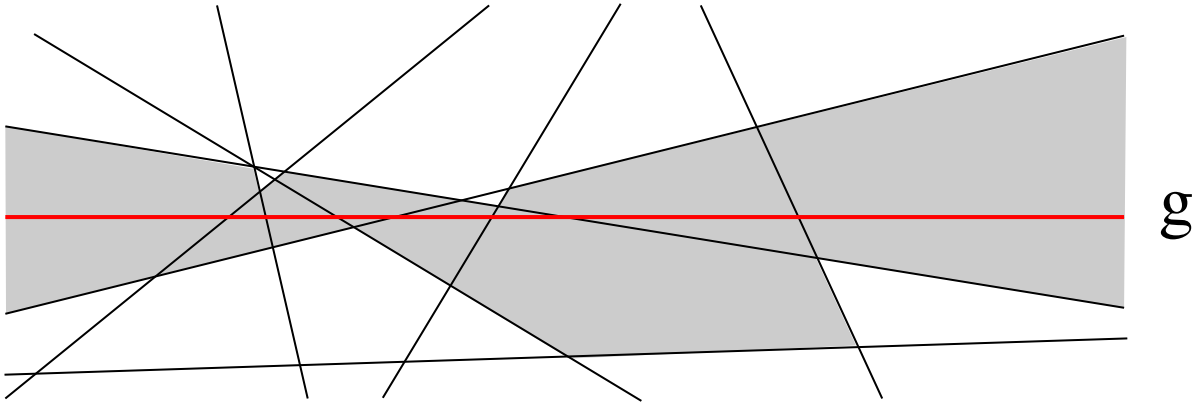


Komplexität Zone einer Geraden g



Komplexität Zone einer Geraden g

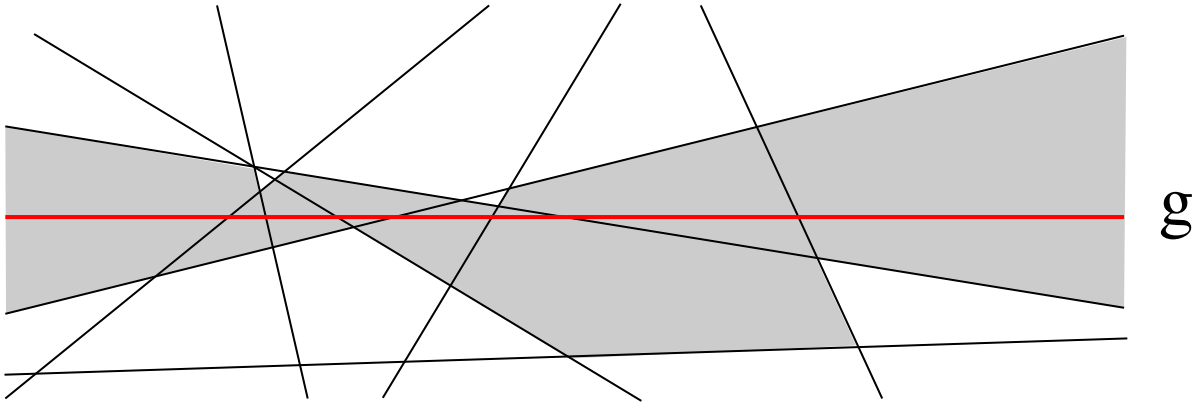
$$\left(\sum_{g \text{ schneidet } Z} (\text{Kanten } Z) \right)$$



Komplexität Zone einer Geraden g

$$\left(\sum_{g \text{ schneidet } Z} (\text{Kanten } Z) \right)$$

n-mal diese Kosten, jede Gerade



Zone(l) in O(n)

Zone(l) in O(n)

- n nicht-senkrechte Geraden, Horizontale l

Zone(l) in O(n)

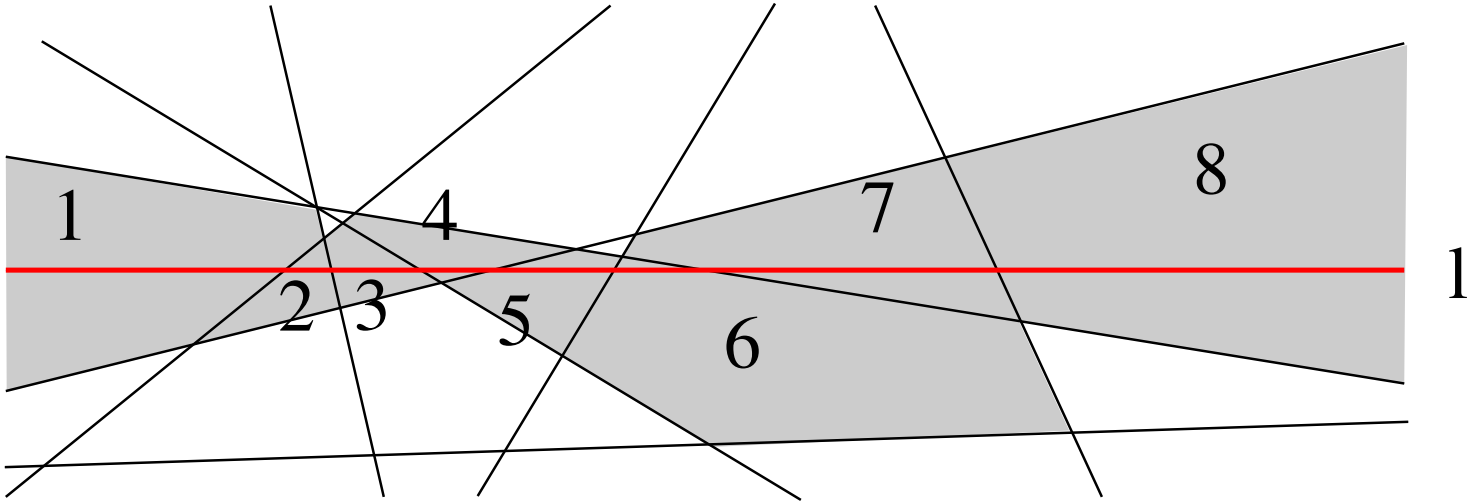
- n nicht-senkrechte Geraden, Horizontale l
- **Def. 1.3:** $\texttt{Zone}(l) := \{\text{Kanten der Zelle } Z | \overline{Z} \cap l \neq \emptyset\}$

Zone(l) in O(n)

- n nicht-senkrechte Geraden, Horizontale l
- **Def. 1.3:** $\texttt{Zone}(l) := \{\text{Kanten der Zelle } Z | \overline{Z} \cap l \neq \emptyset\}$
- **Theorem 1.4:** Komplexität von $\texttt{Zone}(l)$ liegt in $O(n)$

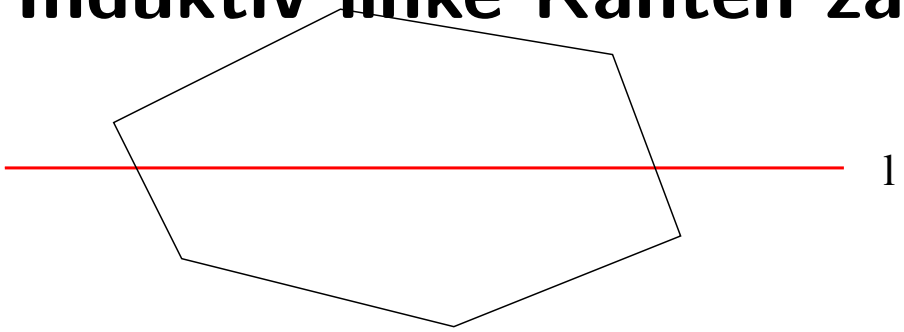
Zone(l) in O(n)

- n nicht-senkrechte Geraden, Horizontale l
- **Def. 1.3:** $\texttt{Zone}(l) := \{\text{Kanten der Zelle } Z | \overline{Z} \cap l \neq \emptyset\}$
- **Theorem 1.4:** Komplexität von $\texttt{Zone}(l)$ liegt in $O(n)$

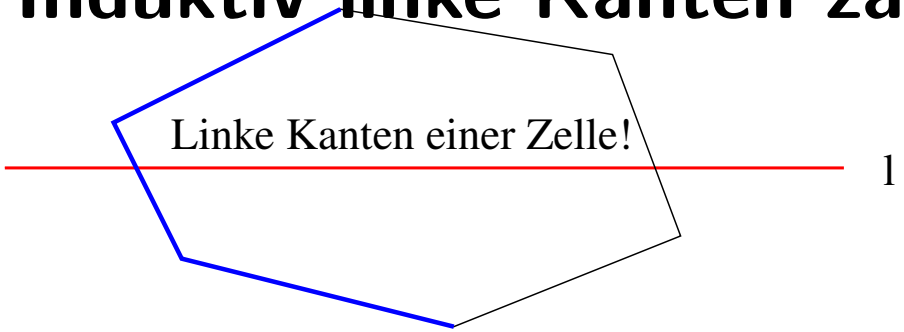


Induktiv linke Kanten zählen

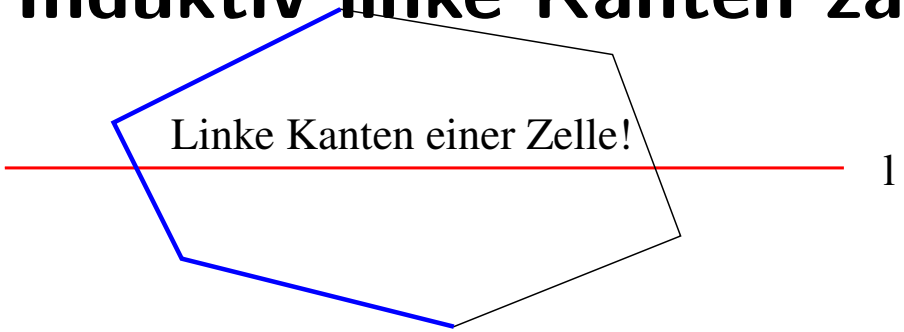
Induktiv linke Kanten zählen



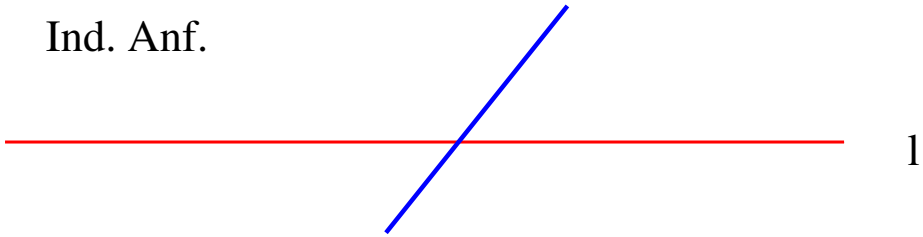
Induktiv linke Kanten zählen



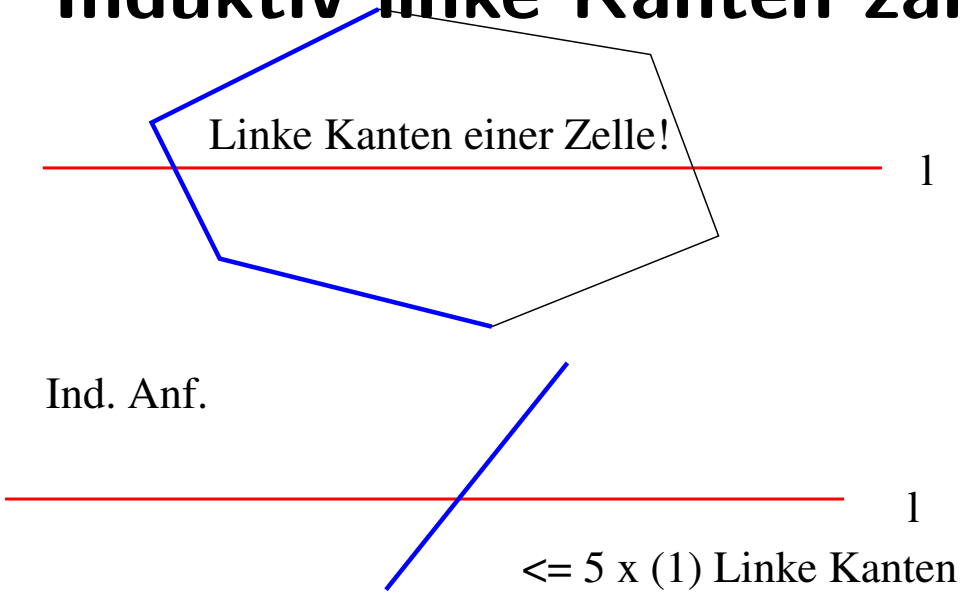
Induktiv linke Kanten zählen



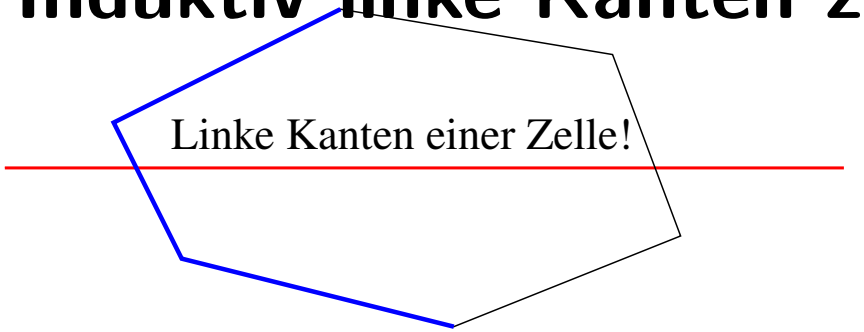
Ind. Anf.



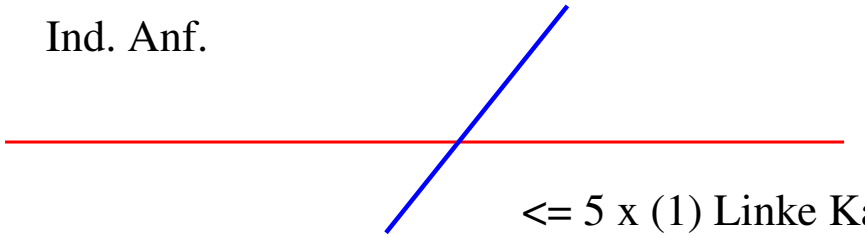
Induktiv linke Kanten zählen



Induktiv linke Kanten zählen

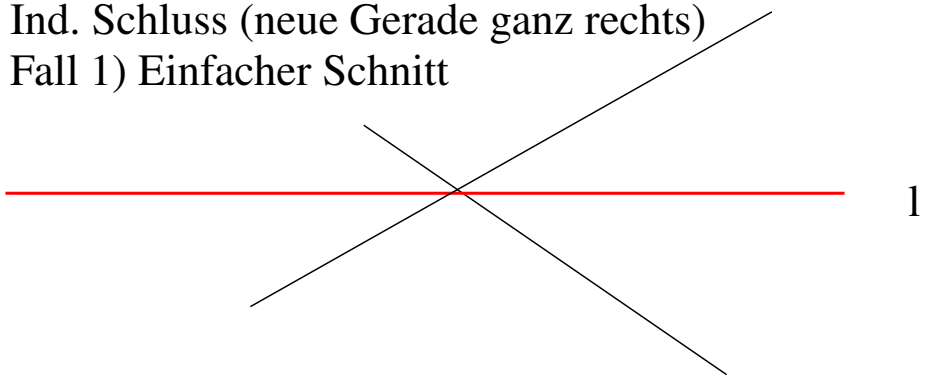


Ind. Anf.

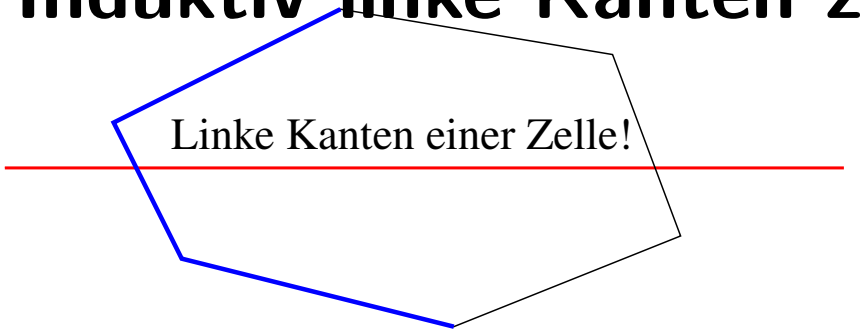


<= 5 x (1) Linke Kanten

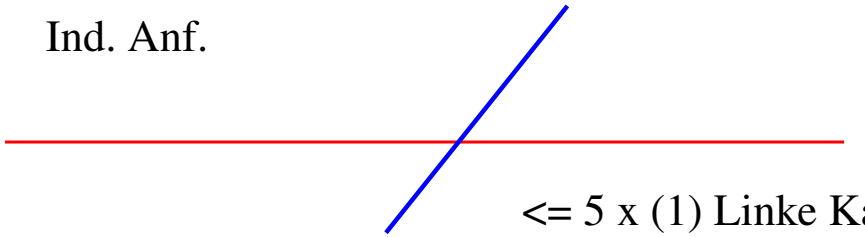
Ind. Schluss (neue Gerade ganz rechts)
Fall 1) Einfacher Schnitt



Induktiv linke Kanten zählen

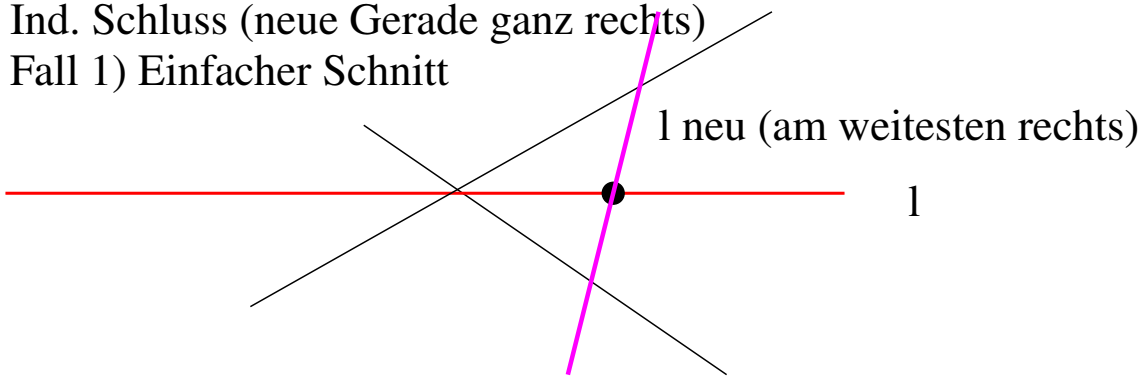


Ind. Anf.

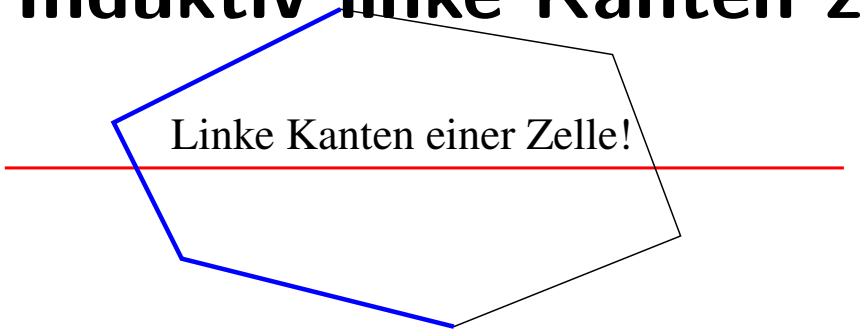


<= 5 x (1) Linke Kanten

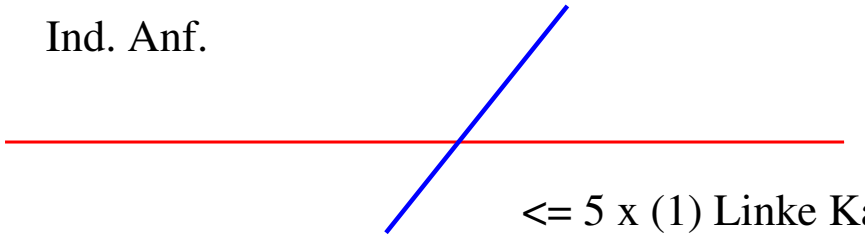
Ind. Schluss (neue Gerade ganz rechts)
Fall 1) Einfacher Schnitt



Induktiv linke Kanten zählen

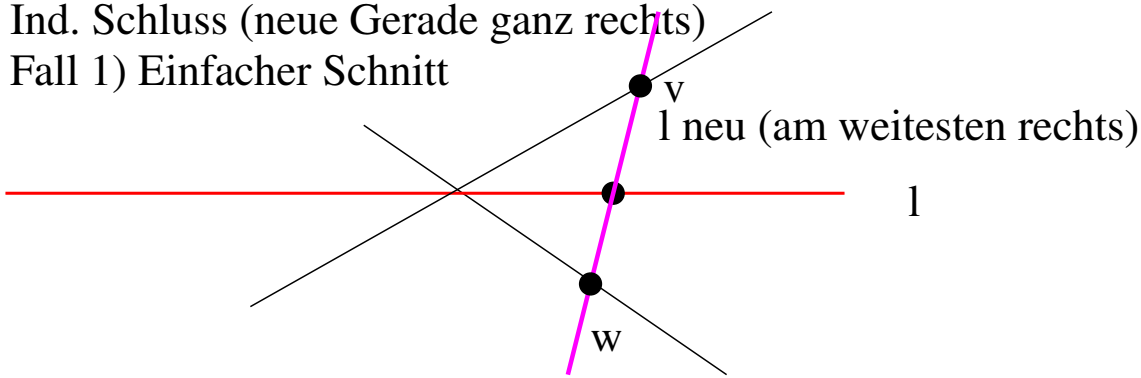


Ind. Anf.

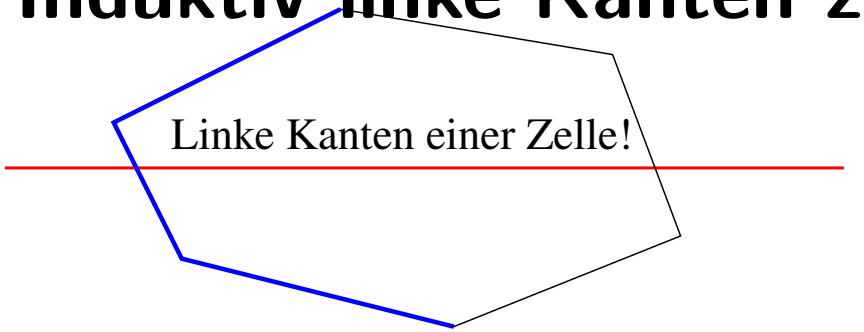


<= 5 x (1) Linke Kanten

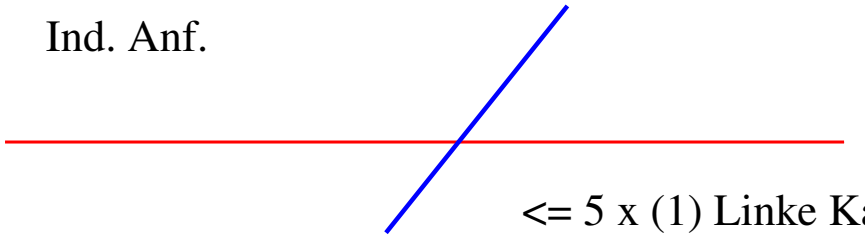
Ind. Schluss (neue Gerade ganz rechts)
Fall 1) Einfacher Schnitt



Induktiv linke Kanten zählen

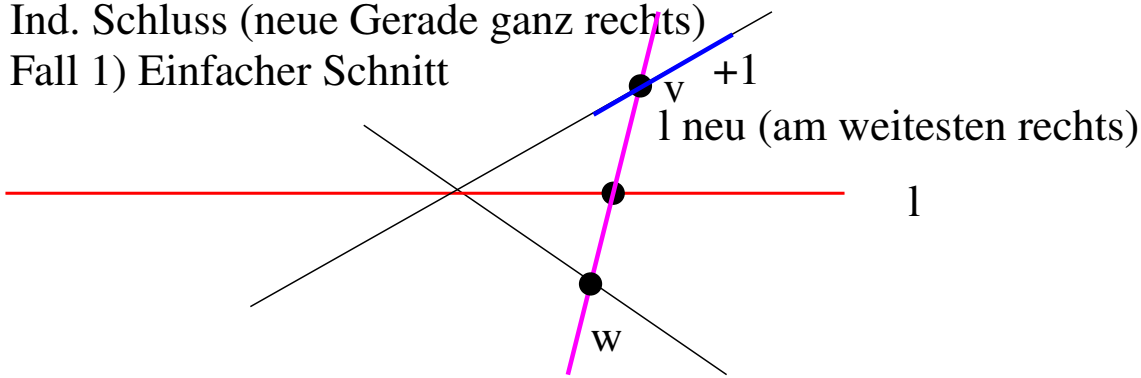


Ind. Anf.

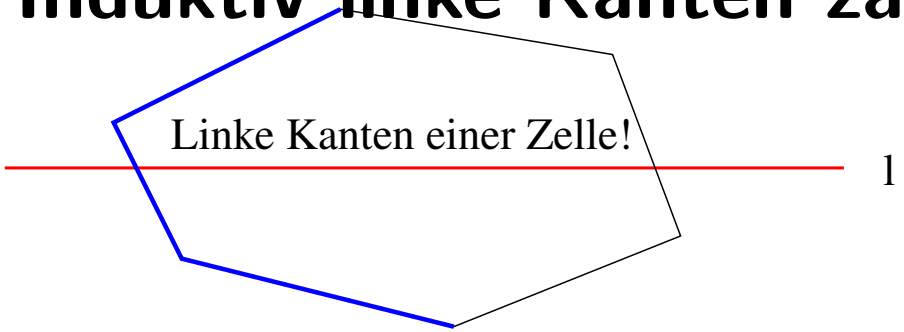


<= 5 x (1) Linke Kanten

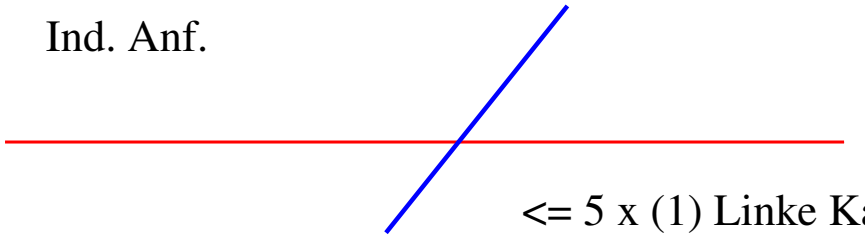
Ind. Schluss (neue Gerade ganz rechts)
Fall 1) Einfacher Schnitt



Induktiv linke Kanten zählen

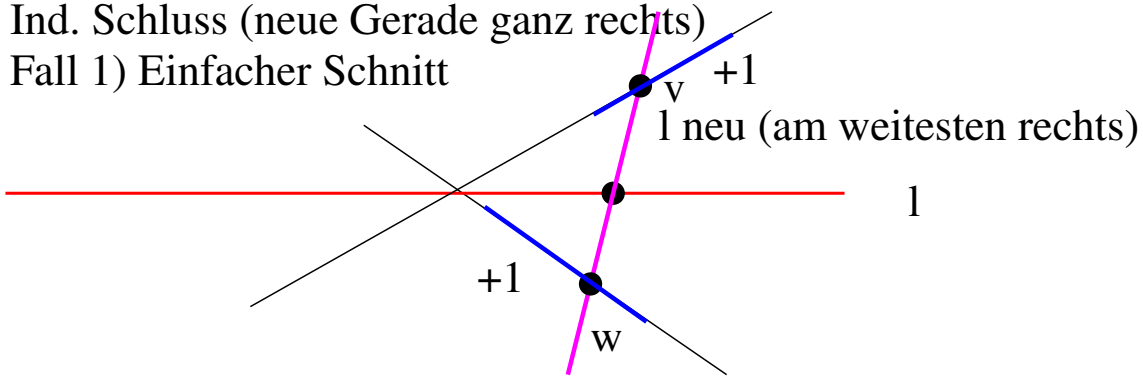


Ind. Anf.

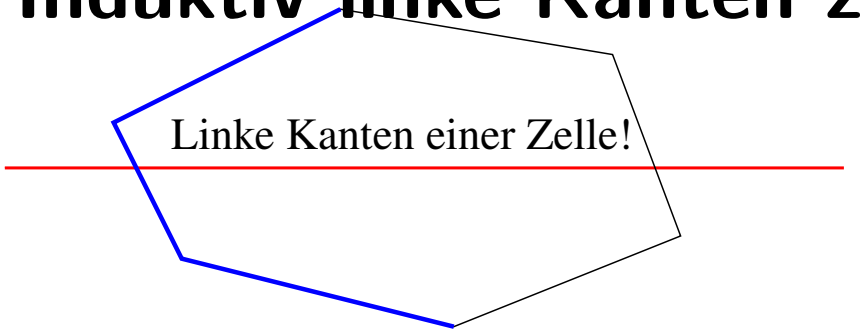


<= 5 x (1) Linke Kanten

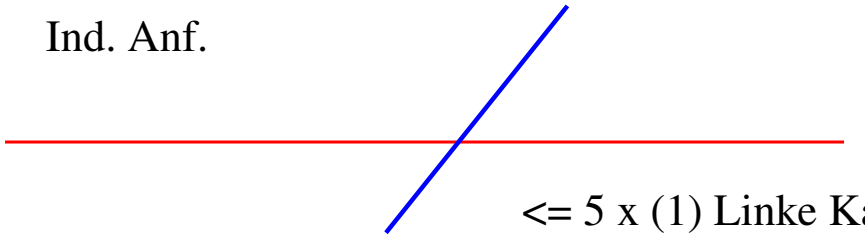
Ind. Schluss (neue Gerade ganz rechts)
Fall 1) Einfacher Schnitt



Induktiv linke Kanten zählen

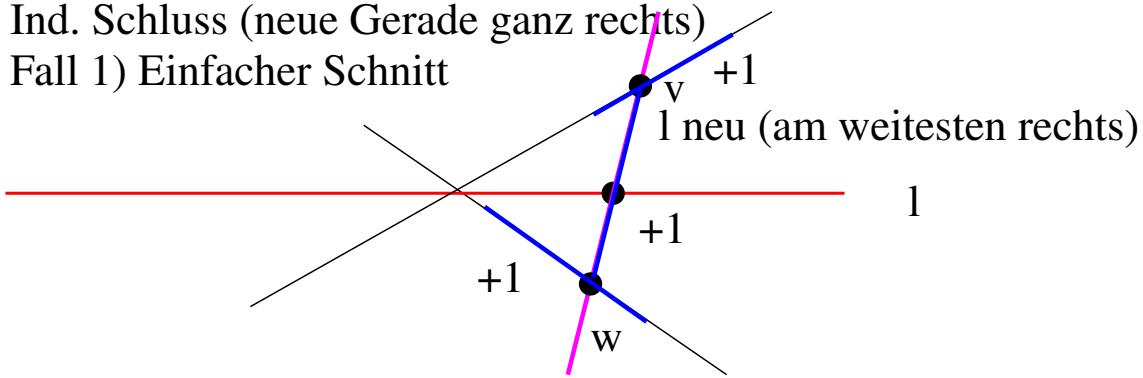


Ind. Anf.



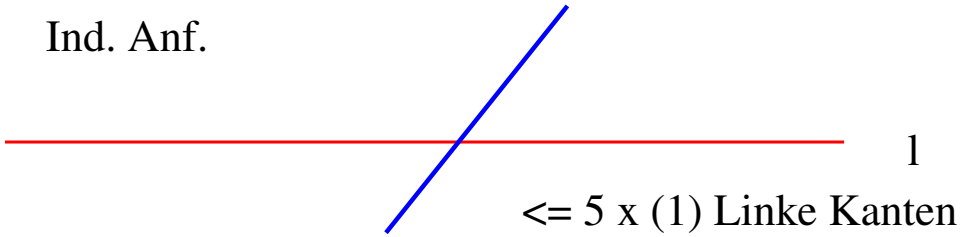
<= 5 x (1) Linke Kanten

Ind. Schluss (neue Gerade ganz rechts)
Fall 1) Einfacher Schnitt

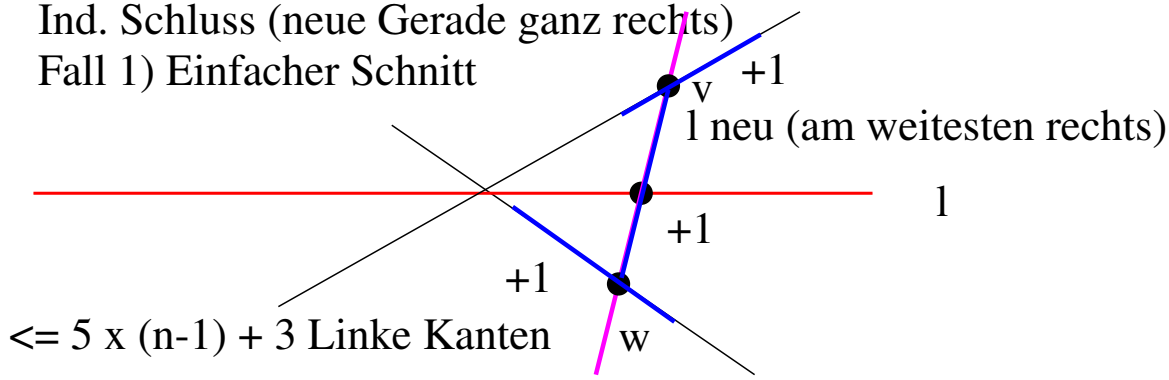


Induktiv linke Kanten zählen

Ind. Anf.

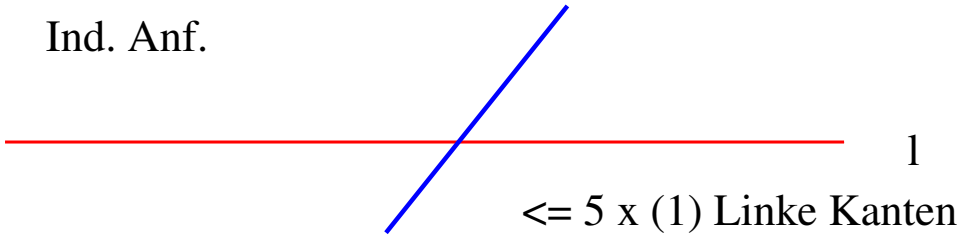


Ind. Schluss (neue Gerade ganz rechts)
Fall 1) Einfacher Schnitt

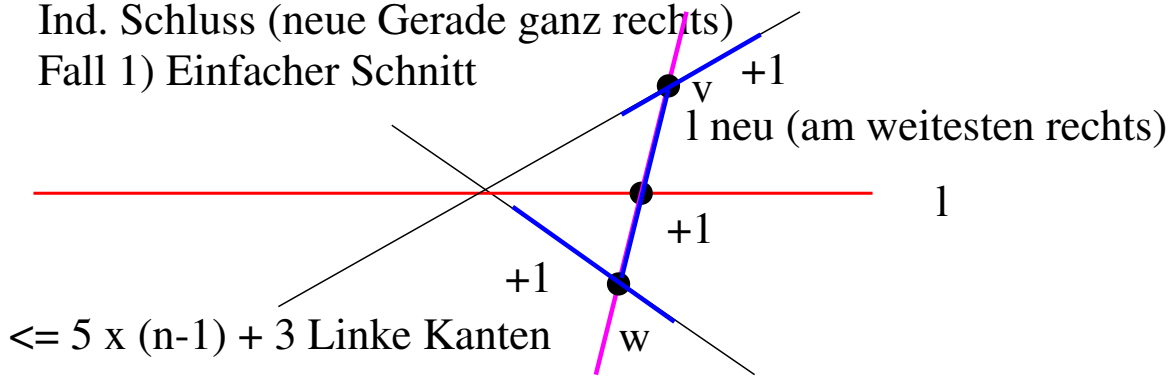


Induktiv linke Kanten zählen

Ind. Anf.



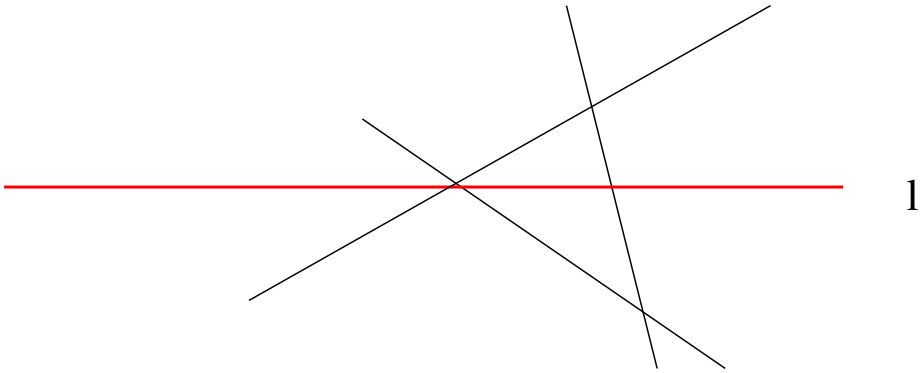
Ind. Schluss (neue Gerade ganz rechts)
Fall 1) Einfacher Schnitt



Induktiv linke Kanten zählen

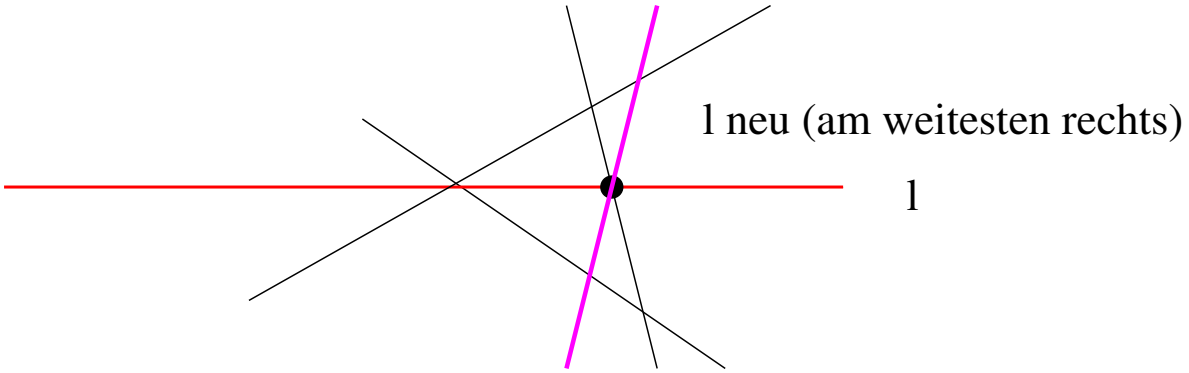
Induktiv linke Kanten zählen

Ind. Schluss (WC)
Fall 2) Schnitt mit einer Kante



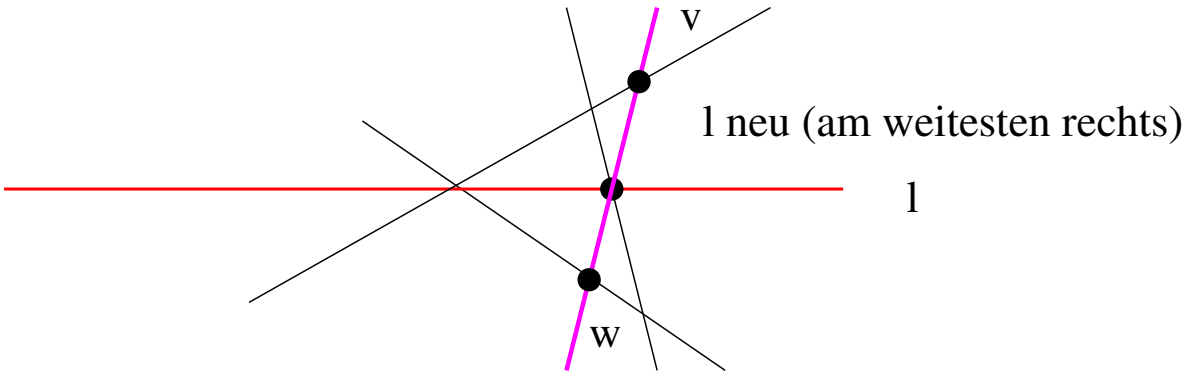
Induktiv linke Kanten zählen

Ind. Schluss (WC)
Fall 2) Schnitt mit einer Kante



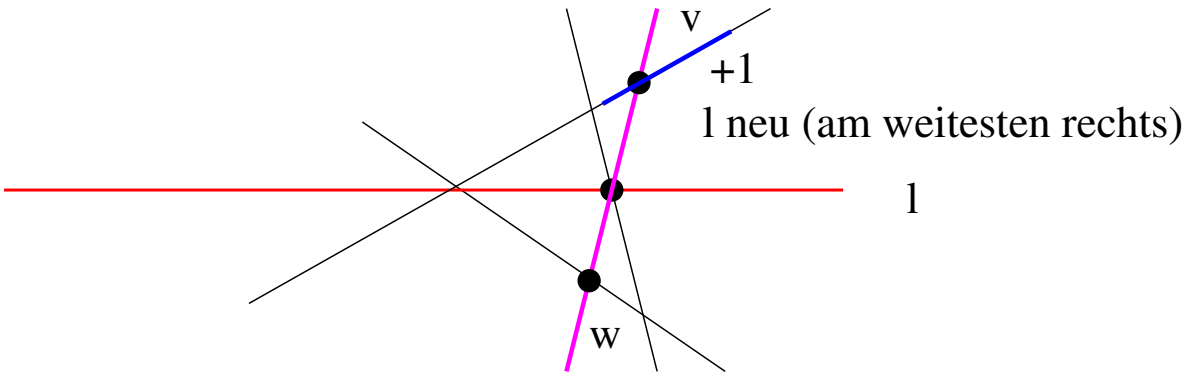
Induktiv linke Kanten zählen

Ind. Schluss (WC)
Fall 2) Schnitt mit einer Kante



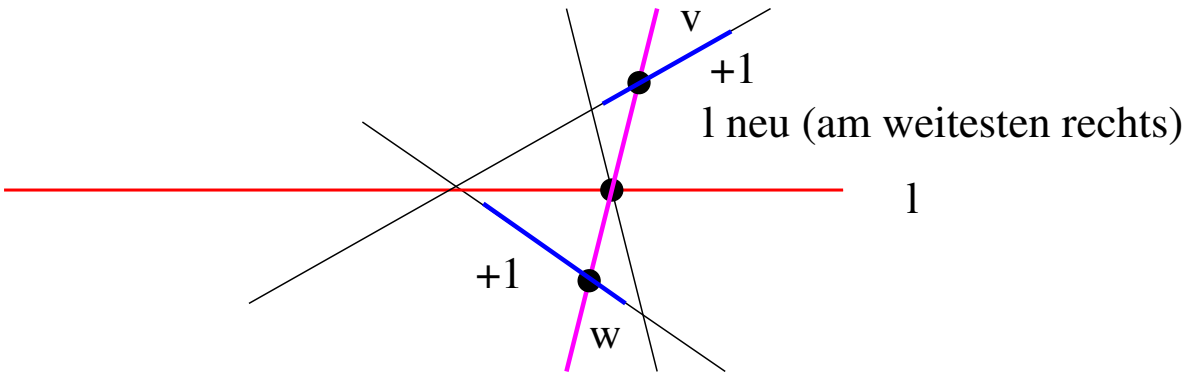
Induktiv linke Kanten zählen

Ind. Schluss (WC)
Fall 2) Schnitt mit einer Kante



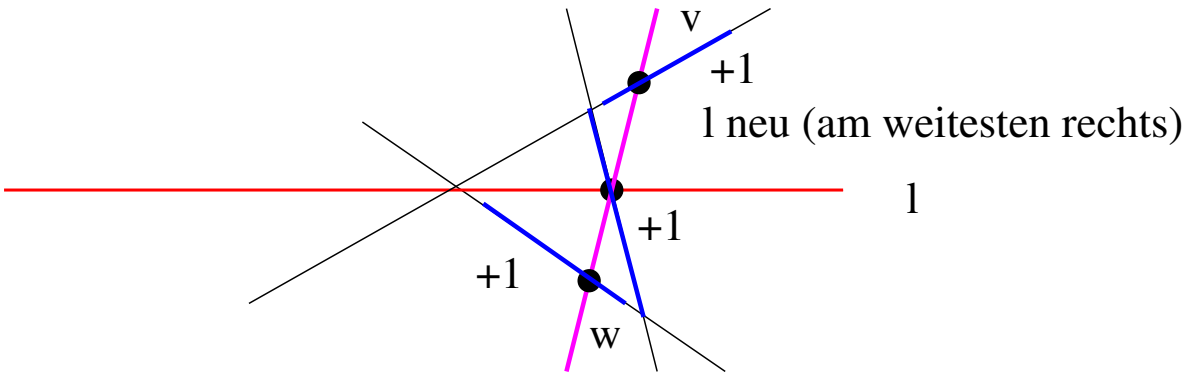
Induktiv linke Kanten zählen

Ind. Schluss (WC)
Fall 2) Schnitt mit einer Kante



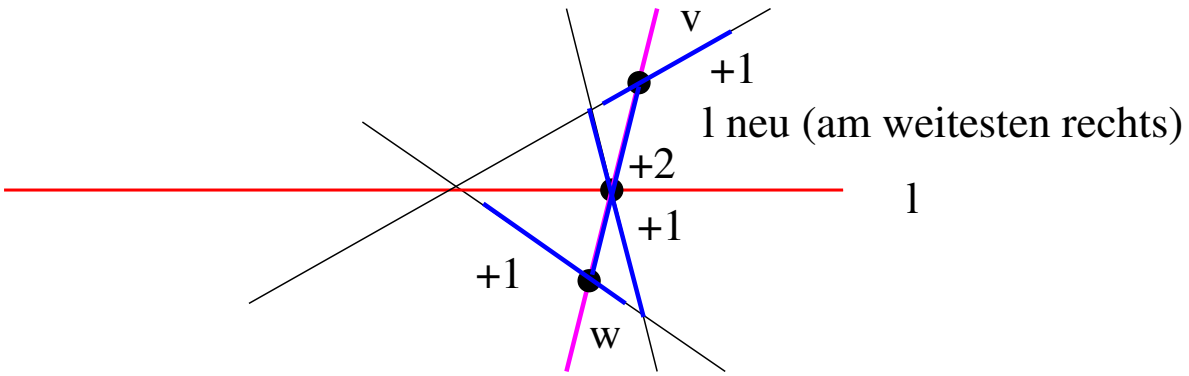
Induktiv linke Kanten zählen

Ind. Schluss (WC)
Fall 2) Schnitt mit einer Kante



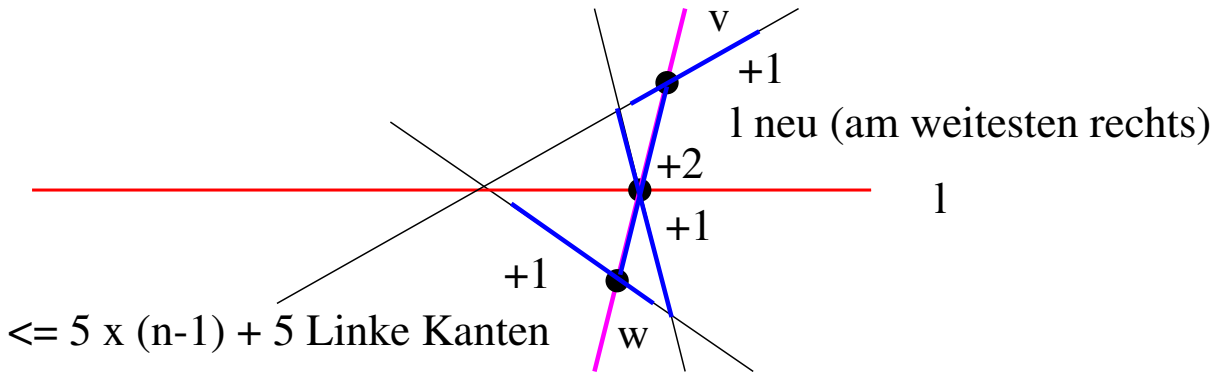
Induktiv linke Kanten zählen

Ind. Schluss (WC)
Fall 2) Schnitt mit einer Kante



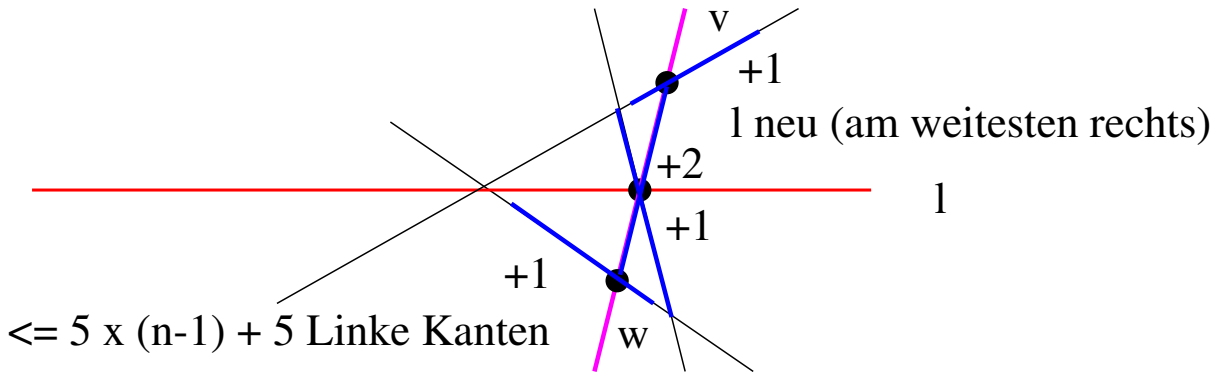
Induktiv linke Kanten zählen

Ind. Schluss (WC)
Fall 2) Schnitt mit einer Kante



Induktiv linke Kanten zählen

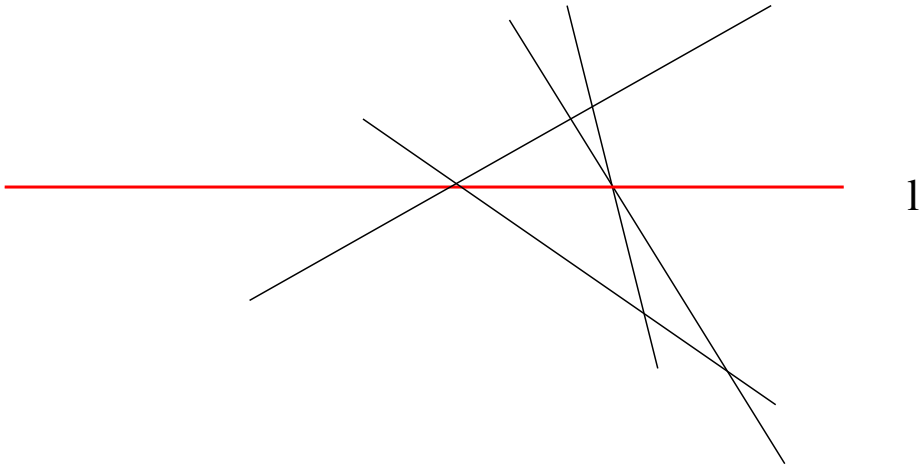
Ind. Schluss (WC)
Fall 2) Schnitt mit einer Kante



Induktiv linke Kanten zählen

Induktiv linke Kanten zählen

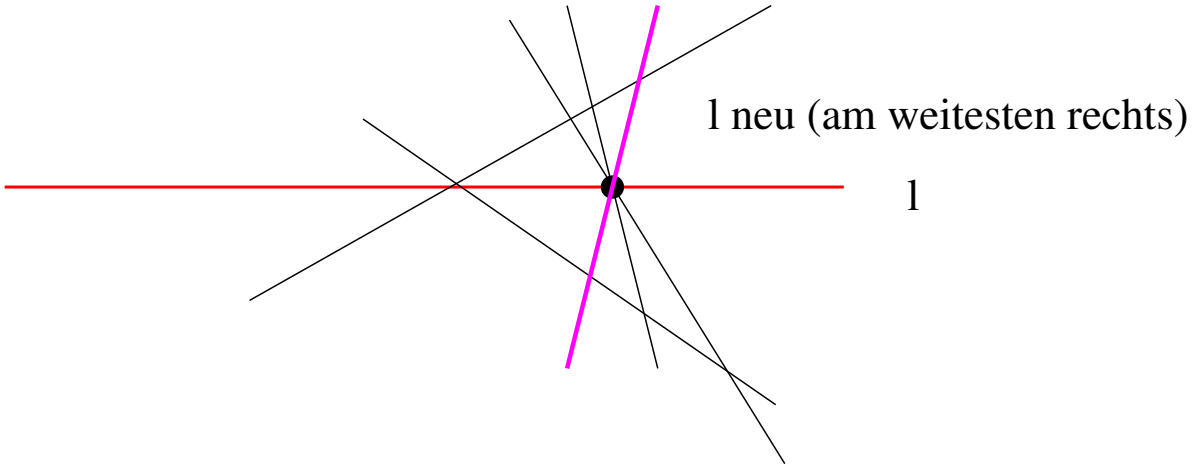
Ind. Schluss (WC)
Fall 3) Schnitt mit mahr als einer Kante



Induktiv linke Kanten zählen

Ind. Schluss (WC)

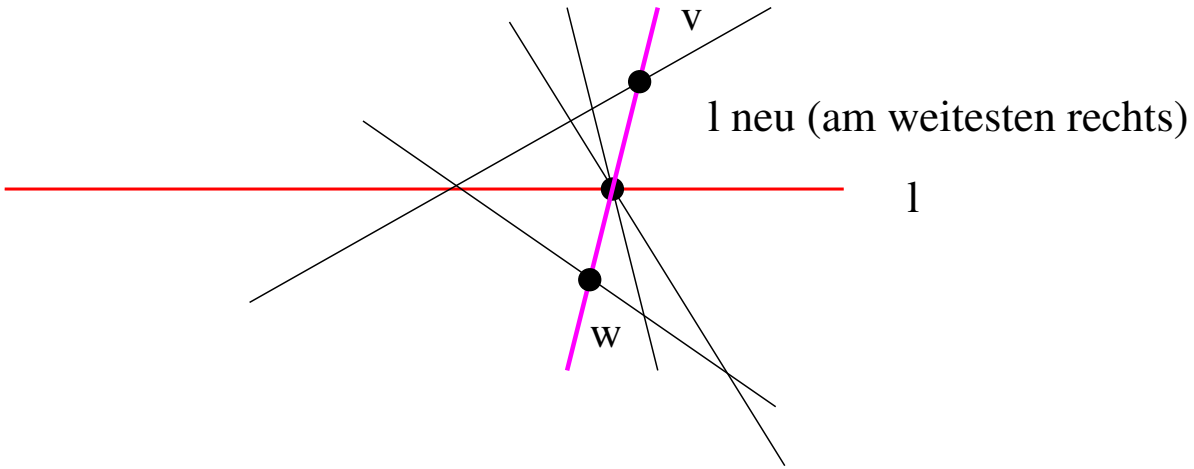
Fall 3) Schnitt mit mahr als einer Kante



Induktiv linke Kanten zählen

Ind. Schluss (WC)

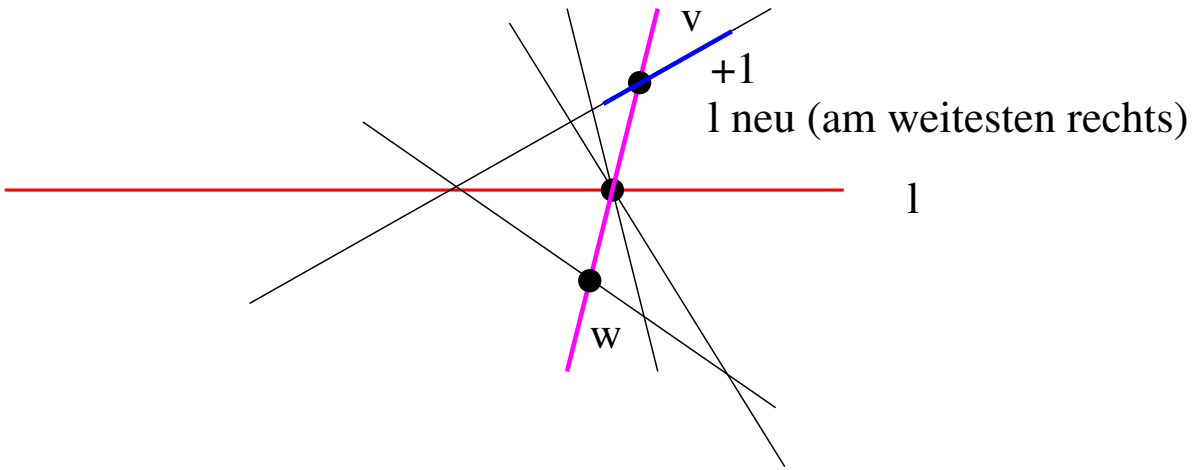
Fall 3) Schnitt mit mahr als einer Kante



Induktiv linke Kanten zählen

Ind. Schluss (WC)

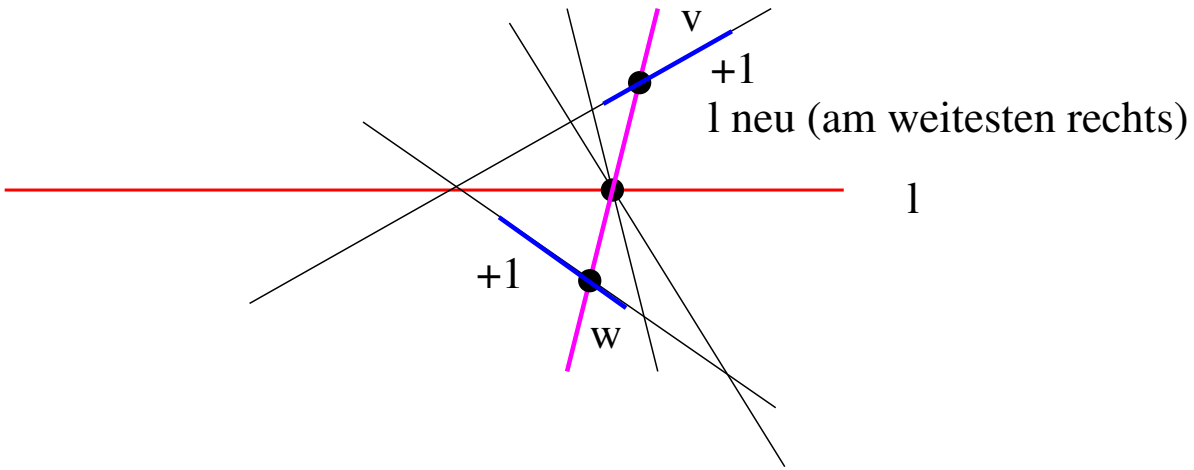
Fall 3) Schnitt mit mahr als einer Kante



Induktiv linke Kanten zählen

Ind. Schluss (WC)

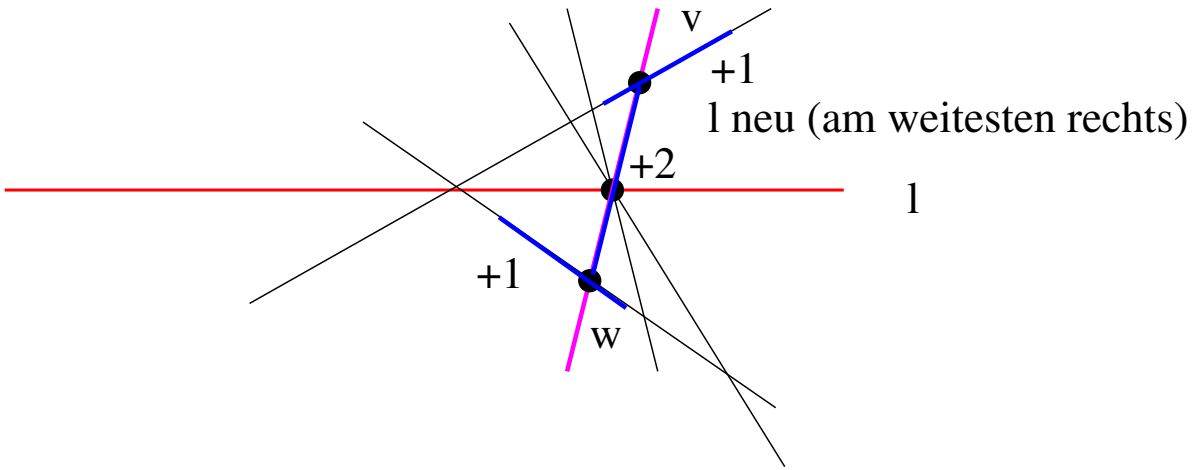
Fall 3) Schnitt mit mahr als einer Kante



Induktiv linke Kanten zählen

Ind. Schluss (WC)

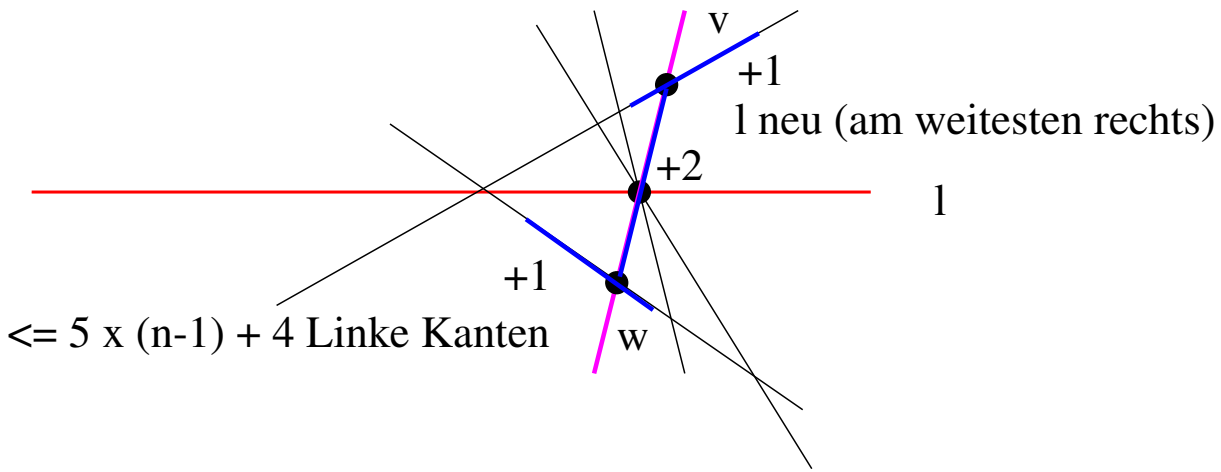
Fall 3) Schnitt mit mahr als einer Kante



Induktiv linke Kanten zählen

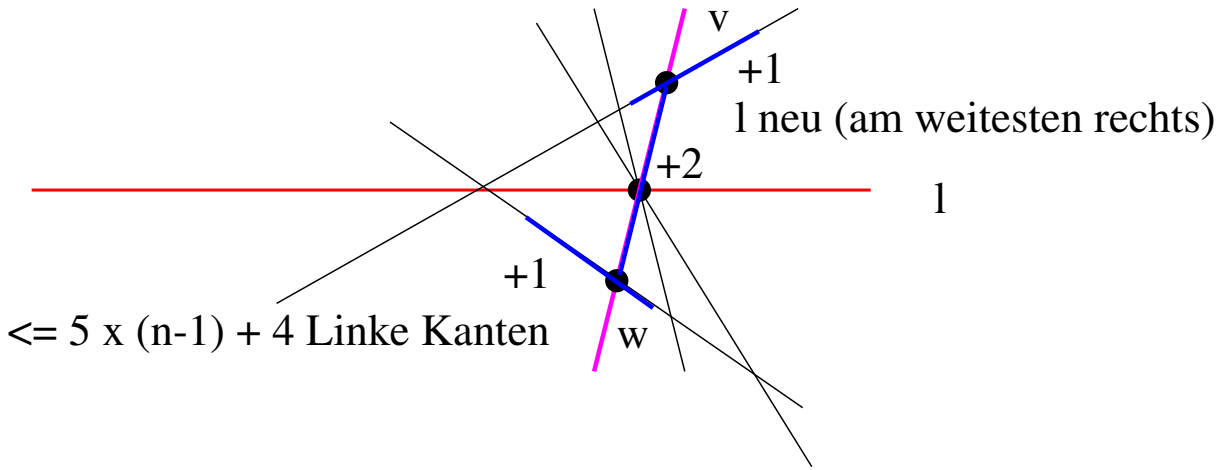
Ind. Schluss (WC)

Fall 3) Schnitt mit mahr als einer Kante



Induktiv linke Kanten zählen

Ind. Schluss (WC)
Fall 3) Schnitt mit mahr als einer Kante



Insgesamt nicht mehr als $5n$ linke Kanten!

Algorithmus Bearbeitungsreihenfolge

Algorithmus Bearbeitungsreihenfolge

- Init.: Nachbarliste, Kandidatenliste, Horizontbäume:

Algorithmus Bearbeitungsreihenfolge

- Init.: Nachbarliste, Kandidatenliste, Horizontbäume: $O(n^2)$

Algorithmus Bearbeitungsreihenfolge

- Init.: Nachbarliste, Kandidatenliste, Horizontbäume: $O(n^2)$
- Kandidaten auswählen, Liste aktualisieren (Horizontbaum):

Algorithmus Bearbeitungsreihenfolge

- Init.: Nachbarliste, Kandidatenliste, Horizontbäume: $O(n^2)$
- Kandidaten auswählen, Liste aktualisieren (Horizontbaum): $O(1)$

Algorithmus Bearbeitungsreihenfolge

- Init.: Nachbarliste, Kandidatenliste, Horizontbäume: $O(n^2)$
- Kandidaten auswählen, Liste aktualisieren (Horizontbaum): $O(1)$
- Horizontbäume aktualisieren:

Algorithmus Bearbeitungsreihenfolge

- Init.: Nachbarliste, Kandidatenliste, Horizontbäume: $O(n^2)$
- Kandidaten auswählen, Liste aktualisieren (Horizontbaum): $O(1)$
- Horizontbäume aktualisieren: $O(n^2)$

Algorithmus Bearbeitungsreihenfolge

- Init.: Nachbarliste, Kandidatenliste, Horizontbäume: $O(n^2)$
- Kandidaten auswählen, Liste aktualisieren (Horizontbaum): $O(1)$
- Horizontbäume aktualisieren: $O(n^2)$
- Reihenfolge des Überschreitens: Bearbeitungsreihenfolge,

Algorithmus Bearbeitungsreihenfolge

- Init.: Nachbarliste, Kandidatenliste, Horizontbäume: $O(n^2)$
- Kandidaten auswählen, Liste aktualisieren (Horizontbaum): $O(1)$
- Horizontbäume aktualisieren: $O(n^2)$
- Reihenfolge des Überschreitens: Bearbeitungsreihenfolge, $O(n^2)$

Zusammenfassung Gesamtalgorithmus

- Horizontbäume aktualisieren: $O(n^2)$
- Topol. Sweep: $O(n^2)$
- Bearbeitungsreihenfolge: $O(n^2)$
- Sweep Sichtbarkeitsgraph: $O(n^2)$
- **Theorem 1.5::** Sichtbarkeitsgraph von n Liniensegmenten in $O(n^2)$ berechenbar. Speicherbedarf $O(n)$