Karlsruhe Institute of Technology

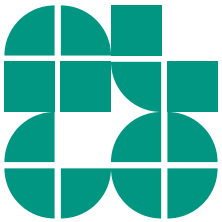
Vorlesung Algorithmische Geometrie

Punktlokalisierung

LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Martin Nöllenburg

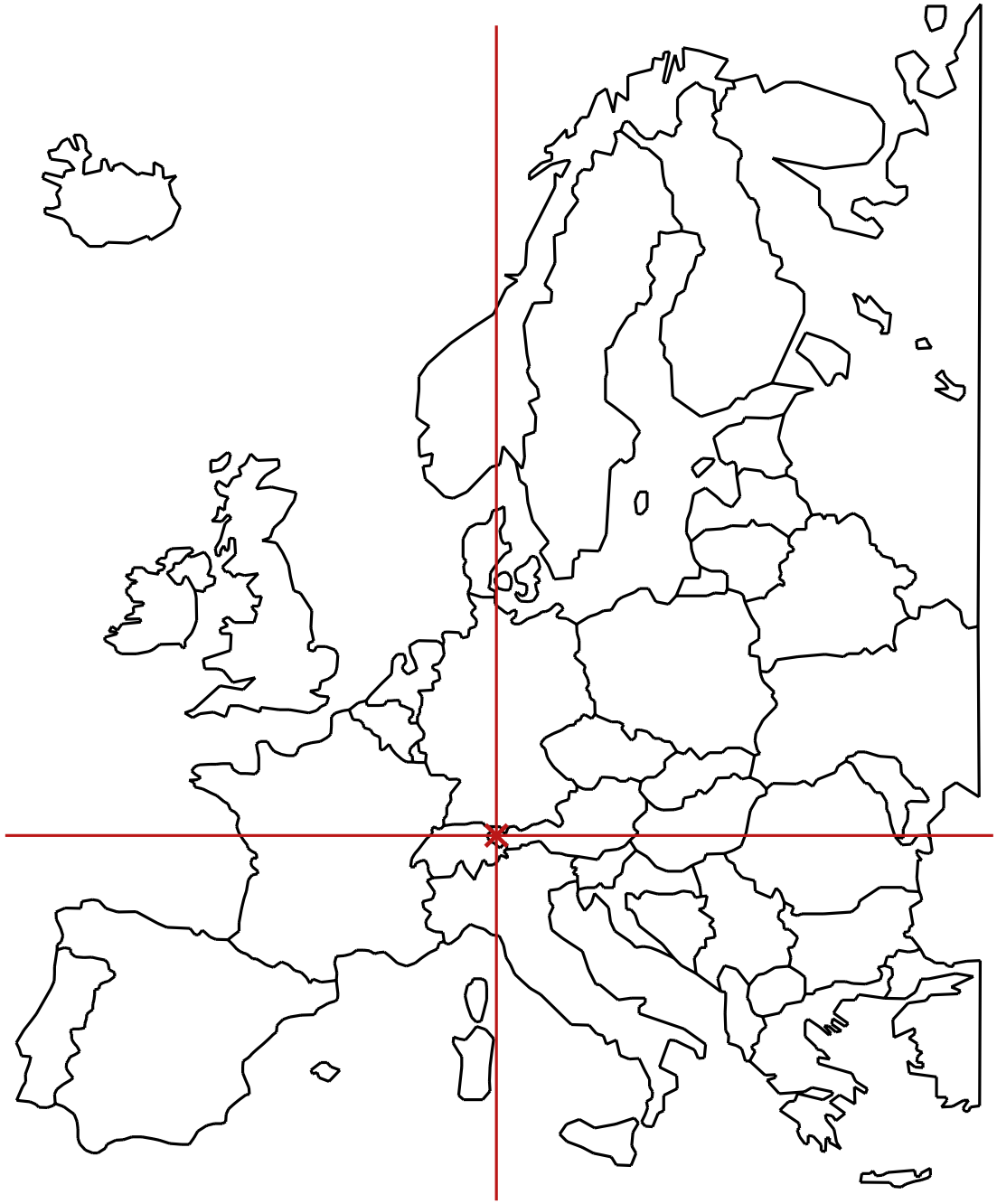
24.05.2011



Motivation

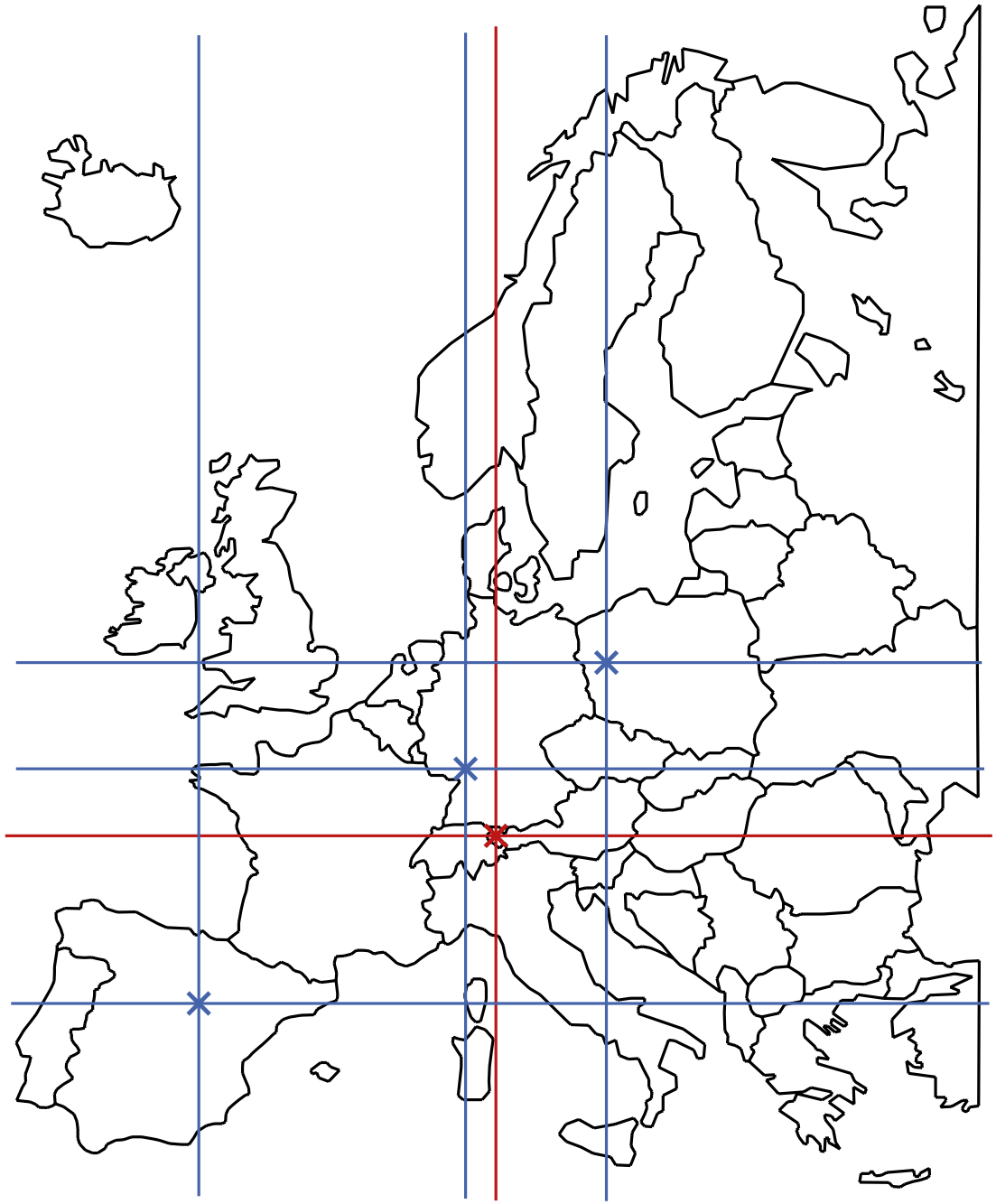
Gegeben eine Position $p = (p_x, p_y)$ in einer Landkarte, bestimme in welchem Land p liegt.

Motivation



Gegeben eine Position $p = (p_x, p_y)$ in einer Landkarte, bestimme in welchem Land p liegt.

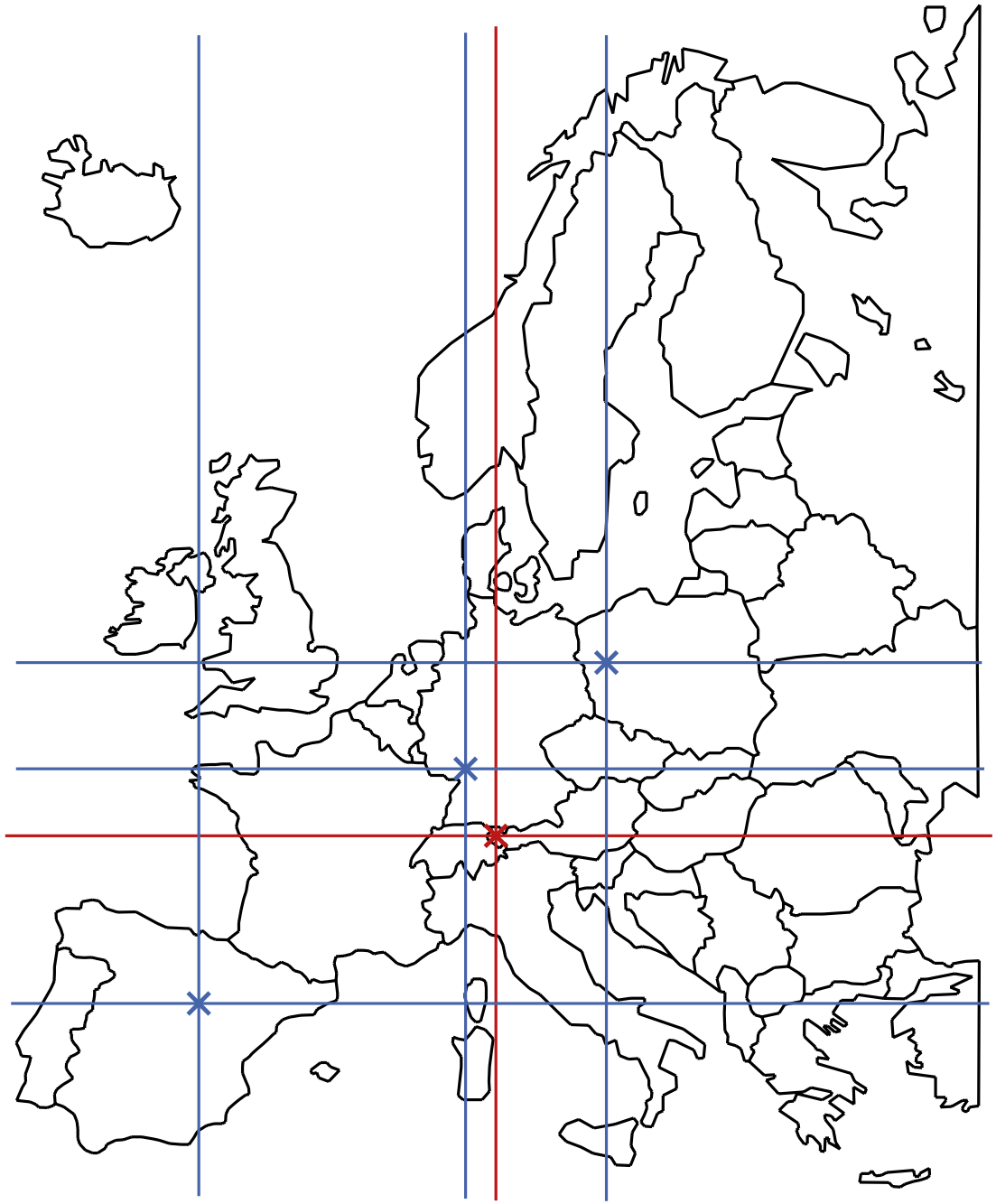
Motivation



Gegeben eine Position $p = (p_x, p_y)$ in einer Landkarte, bestimme in welchem Land p liegt.

Genauer: Finde eine Datenstruktur, die solche Lokalisierungsanfragen effizient beantworten kann.

Motivation

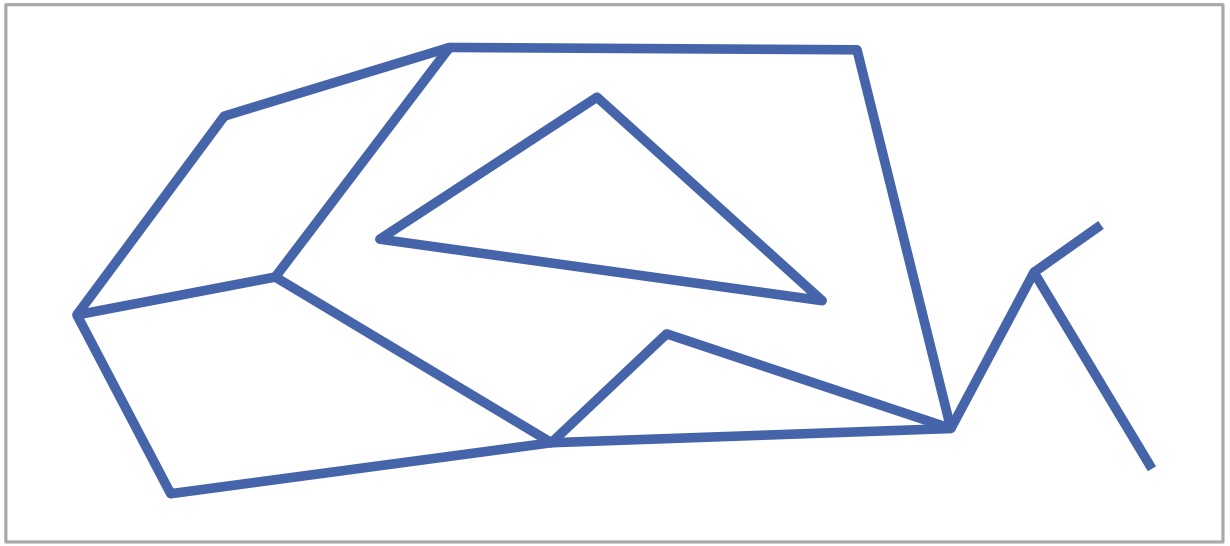


Gegeben eine Position $p = (p_x, p_y)$ in einer Landkarte, bestimme in welchem Land p liegt.

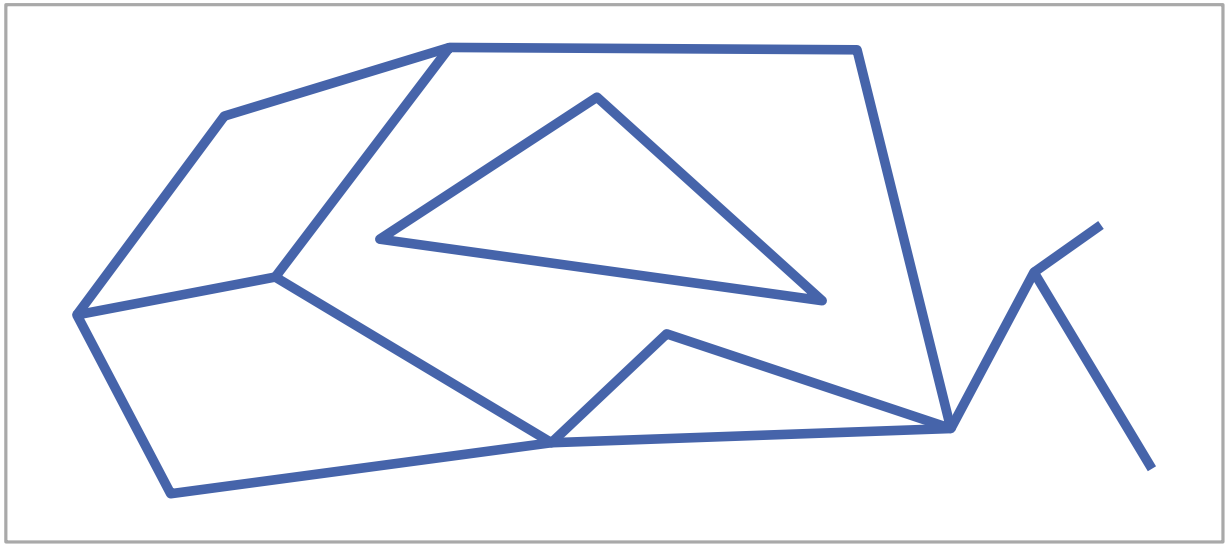
Genauer: Finde eine Datenstruktur, die solche Lokalisierungsanfragen effizient beantworten kann.

Dabei ist die Landkarte eine Unterteilung der Ebene in disjunkte Polygone.

Problemstellung

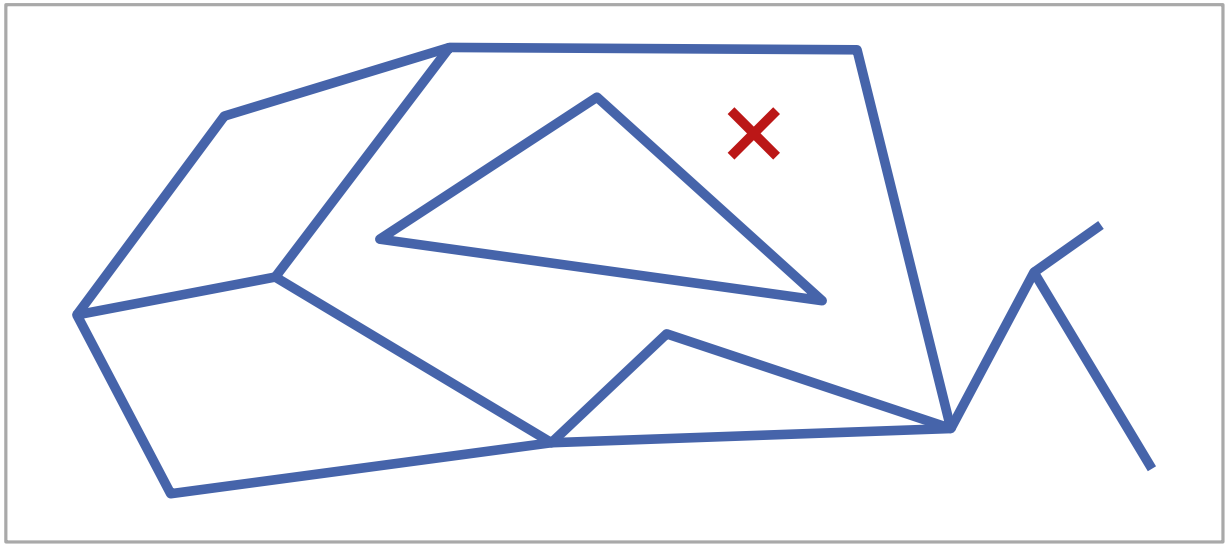


Problemstellung



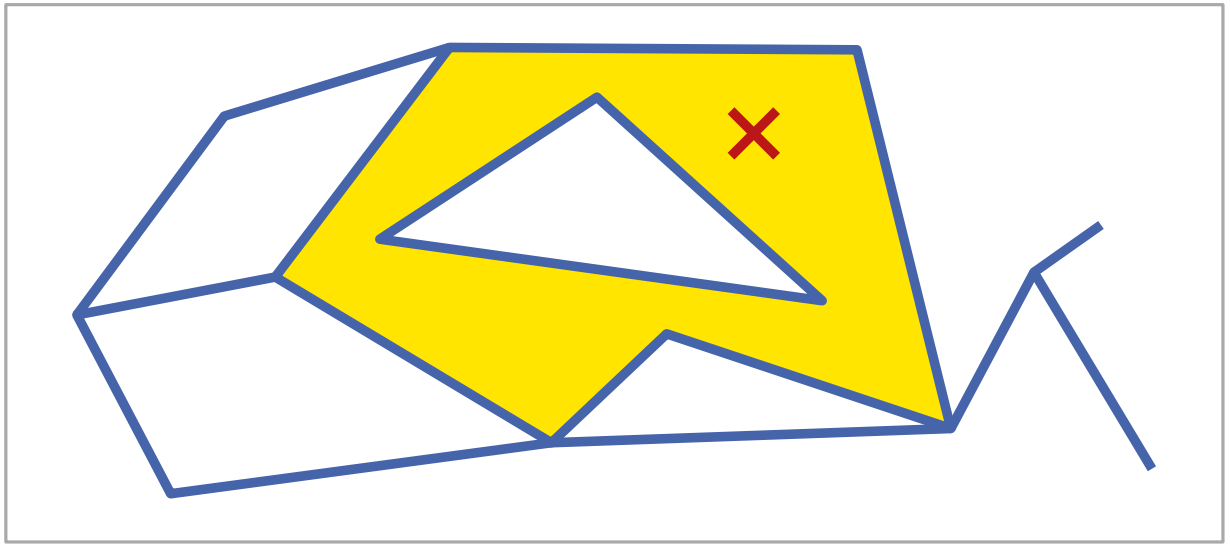
Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Problemstellung



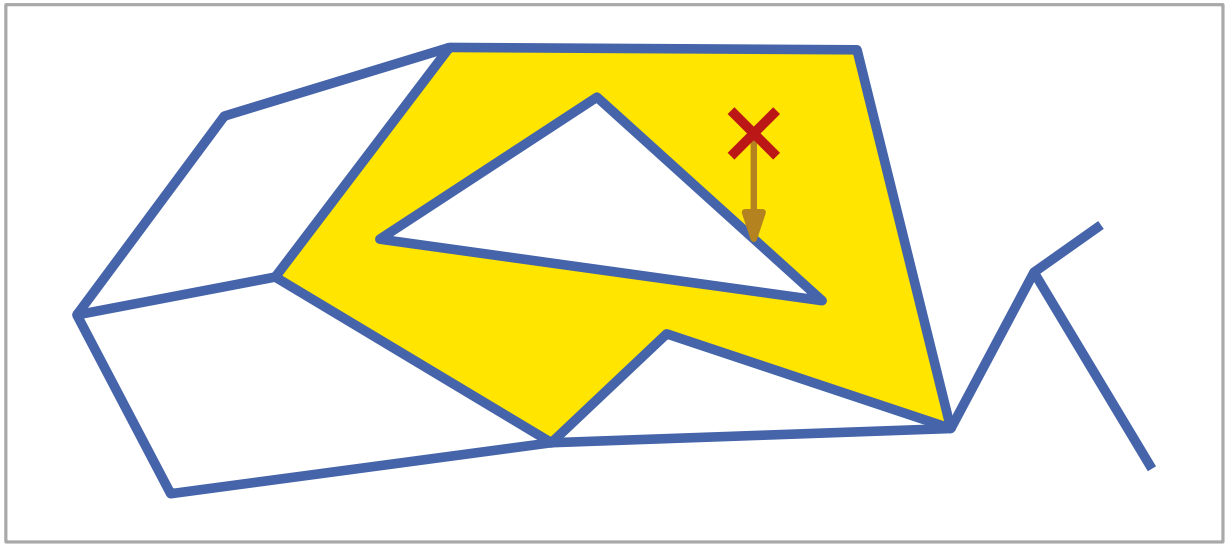
Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Problemstellung



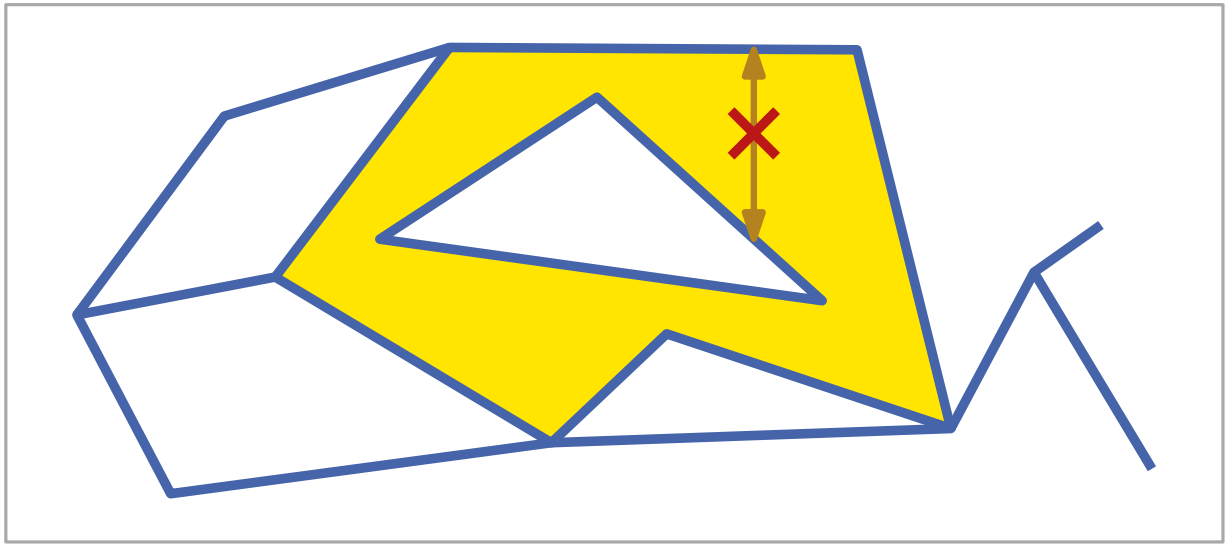
Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Problemstellung



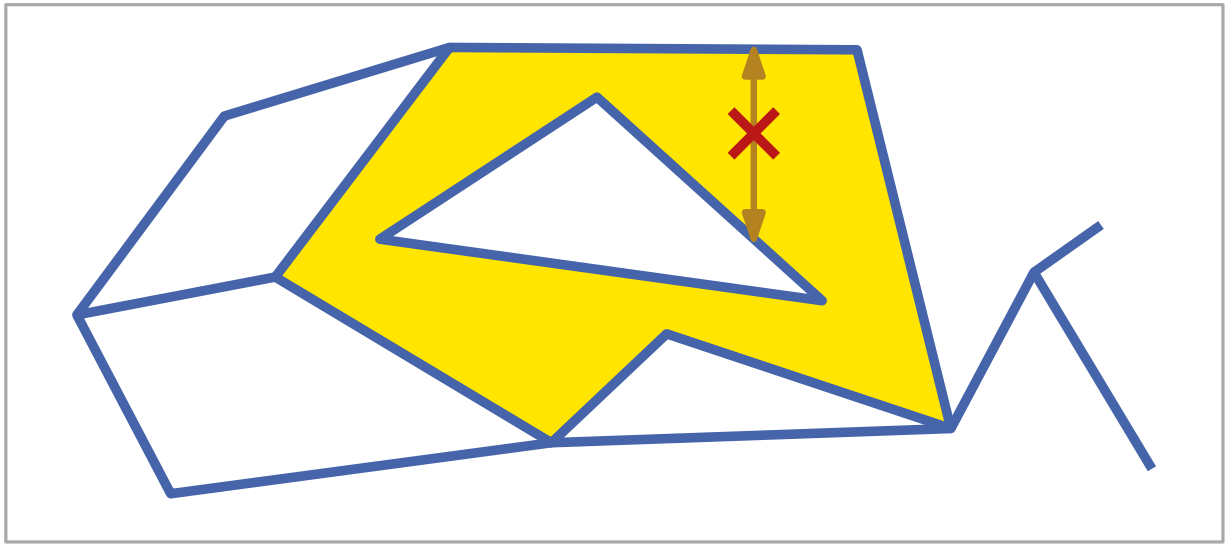
Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Problemstellung



Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

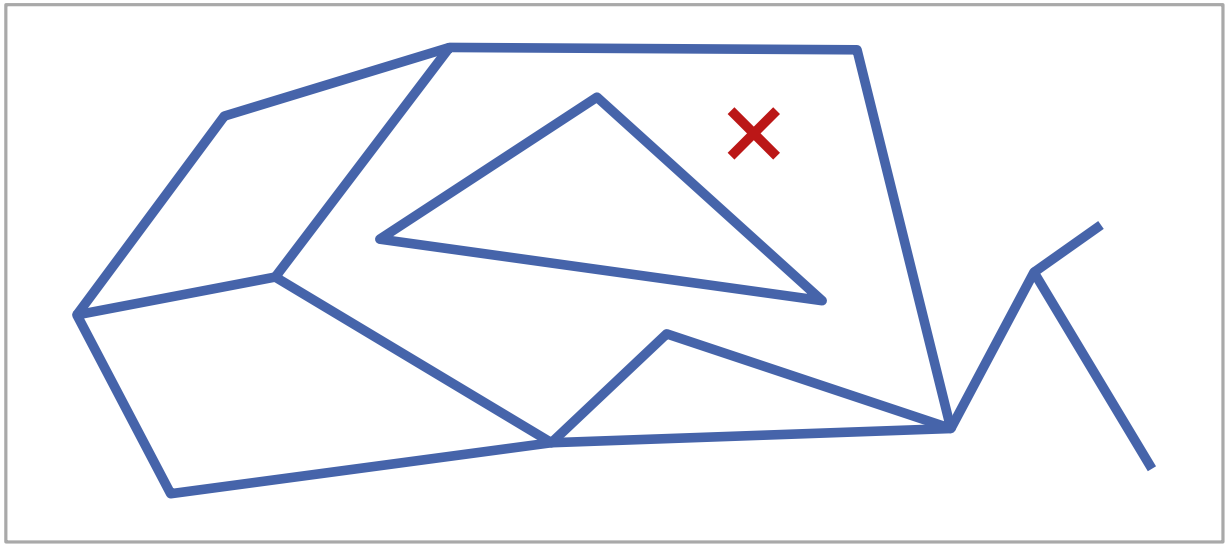
Problemstellung



Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

2 Min. nachdenken!

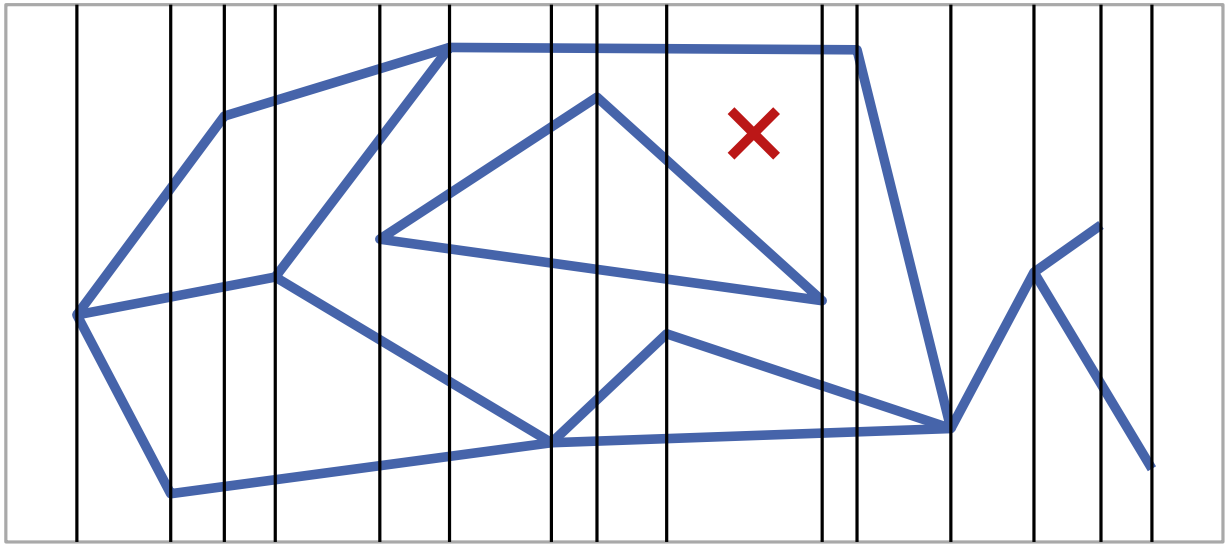
Problemstellung



Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Lösung: Zerteile $\mathcal{S}$ an Punkten in vertikale Streifen.

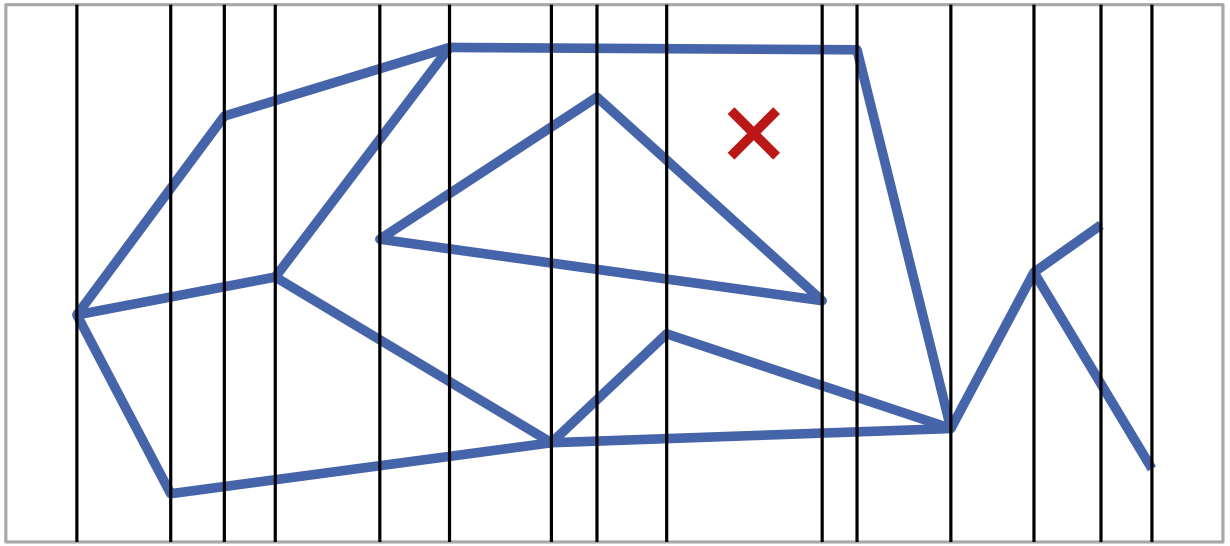
Problemstellung



Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Lösung: Zerteile $\mathcal{S}$ an Punkten in vertikale Streifen.

Problemstellung

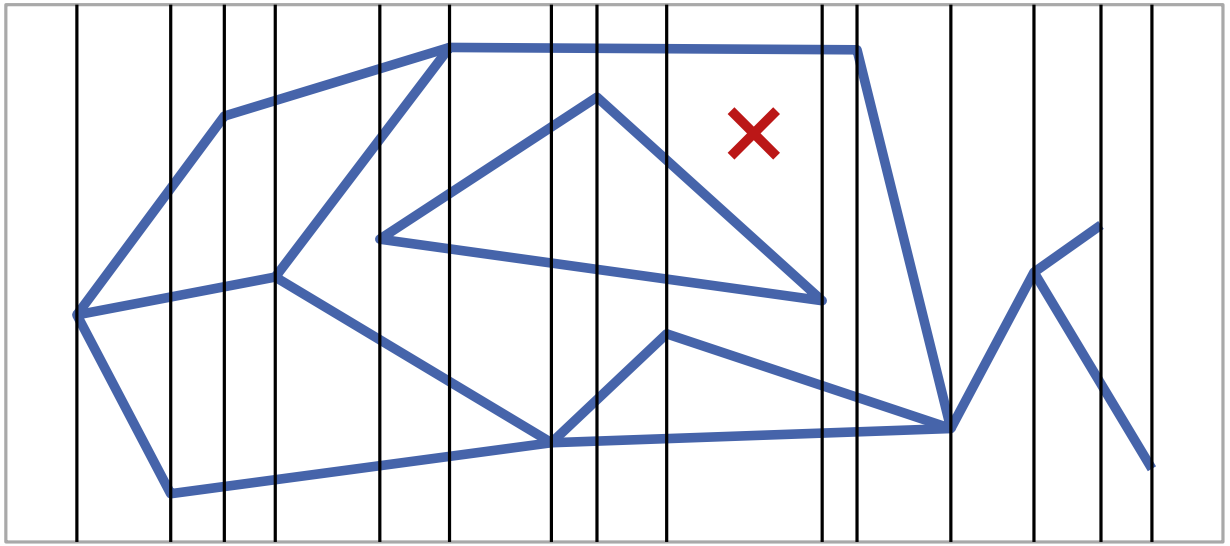


Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Lösung: Zerteile $\mathcal{S}$ an Punkten in vertikale Streifen.

Anfrage:

Problemstellung

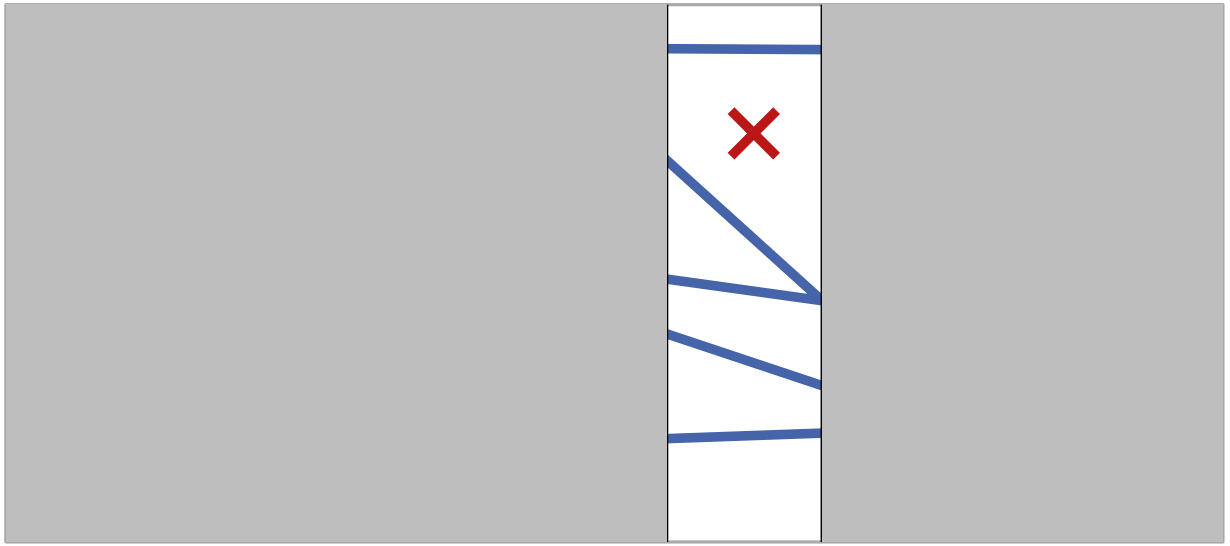


Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Lösung: Zerteile $\mathcal{S}$ an Punkten in vertikale Streifen.

Anfrage:
- finde richtigen Streifen

Problemstellung

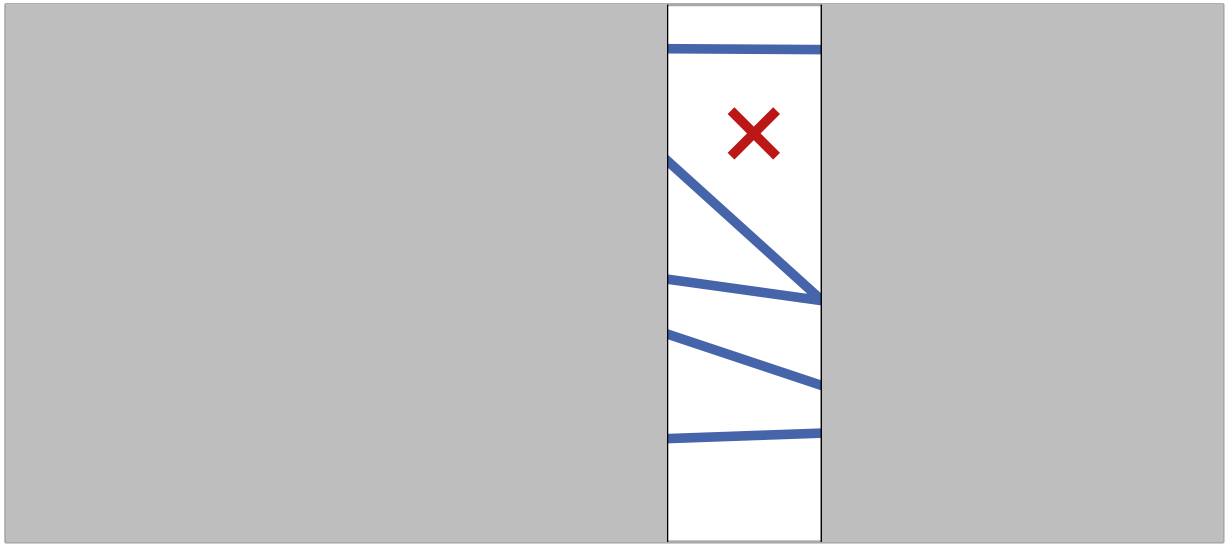


Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Lösung: Zerteile $\mathcal{S}$ an Punkten in vertikale Streifen.

Anfrage:
- finde richtigen Streifen

Problemstellung

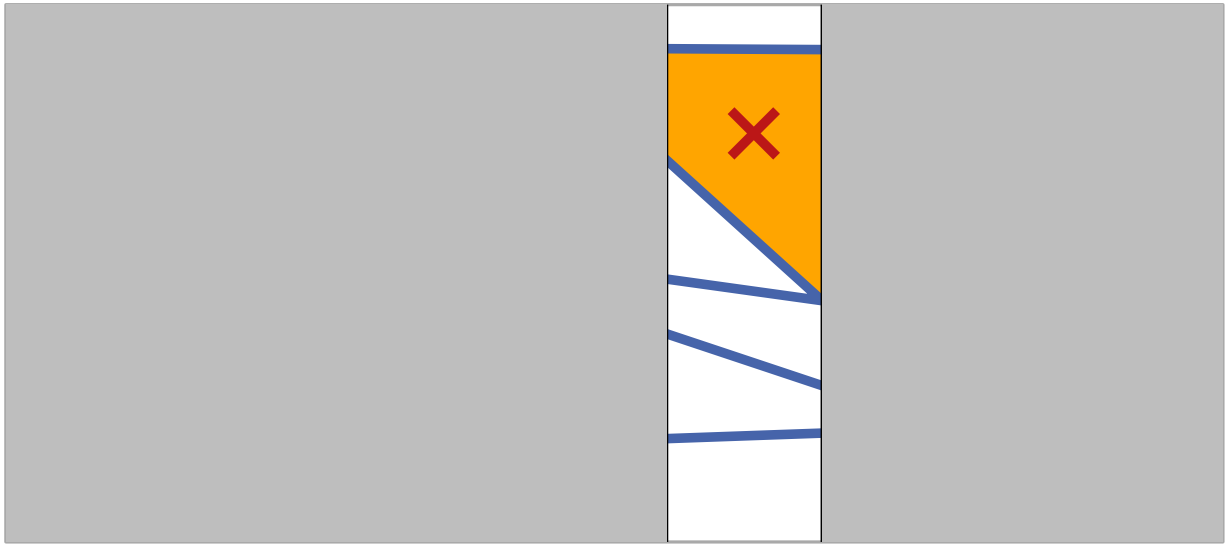


Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Lösung: Zerteile $\mathcal{S}$ an Punkten in vertikale Streifen.

Anfrage:
- finde richtigen Streifen
- durchsuche diesen Streifen

Problemstellung

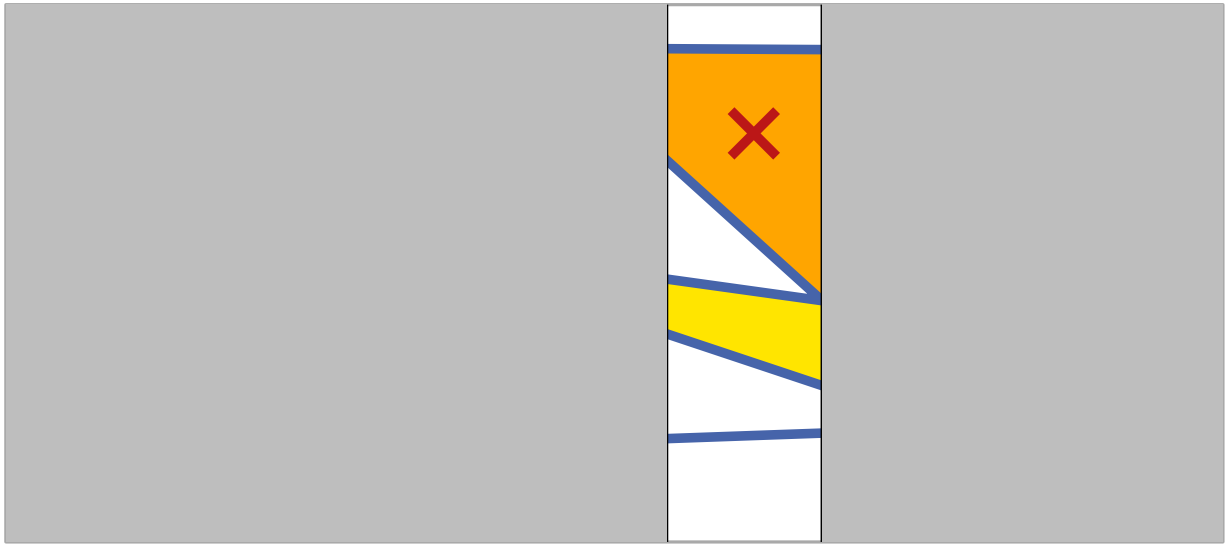


Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Lösung: Zerteile $\mathcal{S}$ an Punkten in vertikale Streifen.

Anfrage:
- finde richtigen Streifen
- durchsuche diesen Streifen

Problemstellung



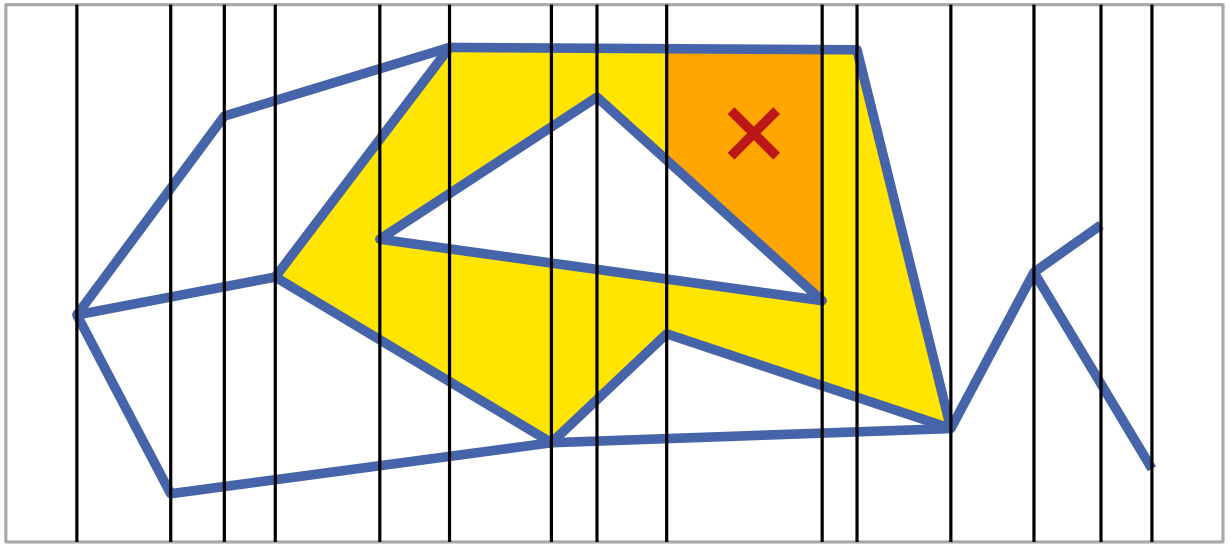
Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Lösung: Zerteile $\mathcal{S}$ an Punkten in vertikale Streifen.

Anfrage:
- finde richtigen Streifen
- durchsuche diesen Streifen

} 2 binäre Suchen

Problemstellung



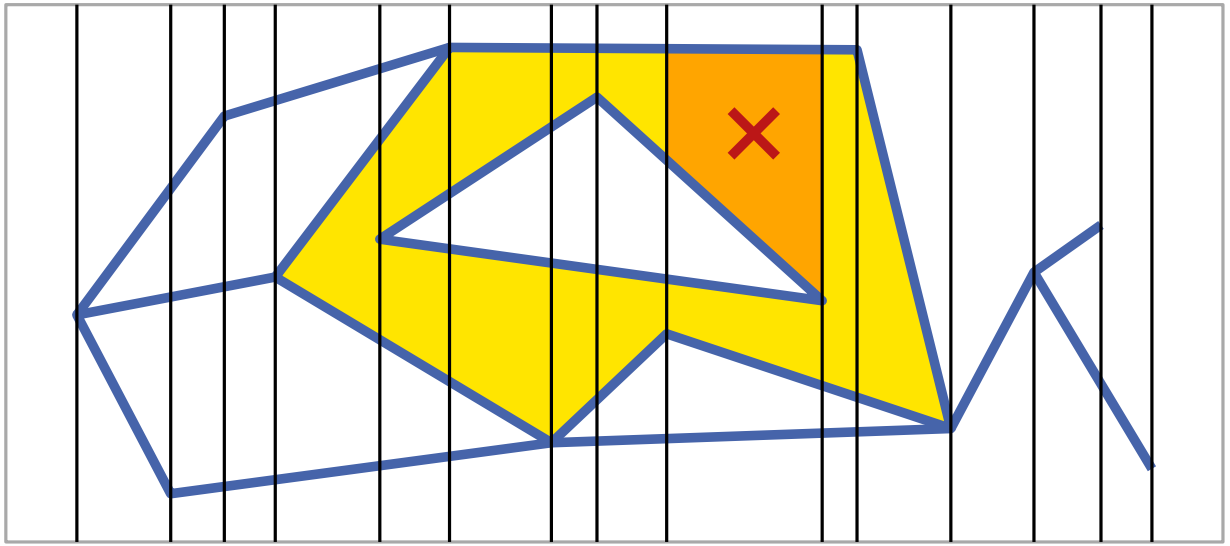
Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Lösung: Zerteile $\mathcal{S}$ an Punkten in vertikale Streifen. $O(\log n)$ Zeit

Anfrage:
- finde richtigen Streifen
- durchsuche diesen Streifen

} 2 binäre Suchen

Problemstellung



Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

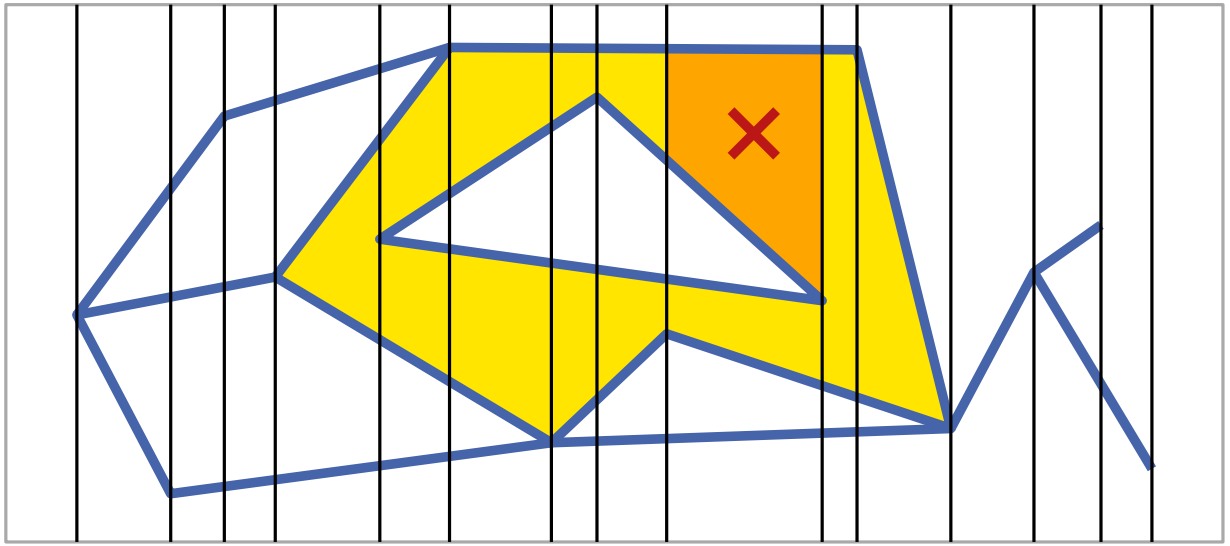
Lösung: Zerteile $\mathcal{S}$ an Punkten in vertikale Streifen. $O(\log n)$ Zeit

Anfrage:
- finde richtigen Streifen
- durchsuche diesen Streifen

} 2 binäre Suchen

Aber:

Problemstellung



Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

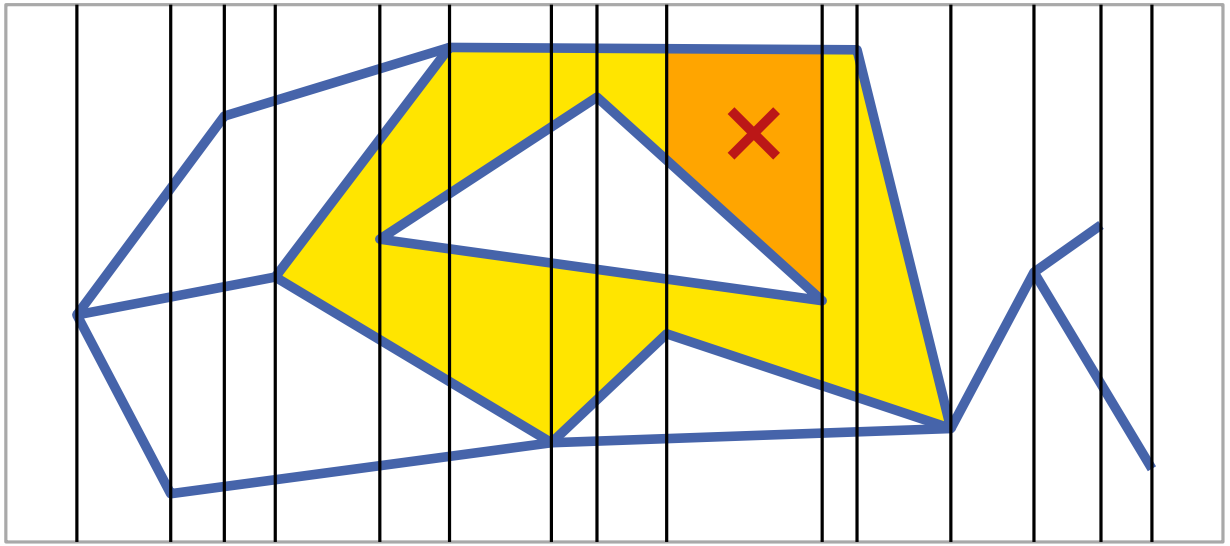
Lösung: Zerteile $\mathcal{S}$ an Punkten in vertikale Streifen. $O(\log n)$ Zeit

Anfrage:
- finde richtigen Streifen
- durchsuche diesen Streifen

} 2 binäre Suchen

Aber: Platz?

Problemstellung



Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

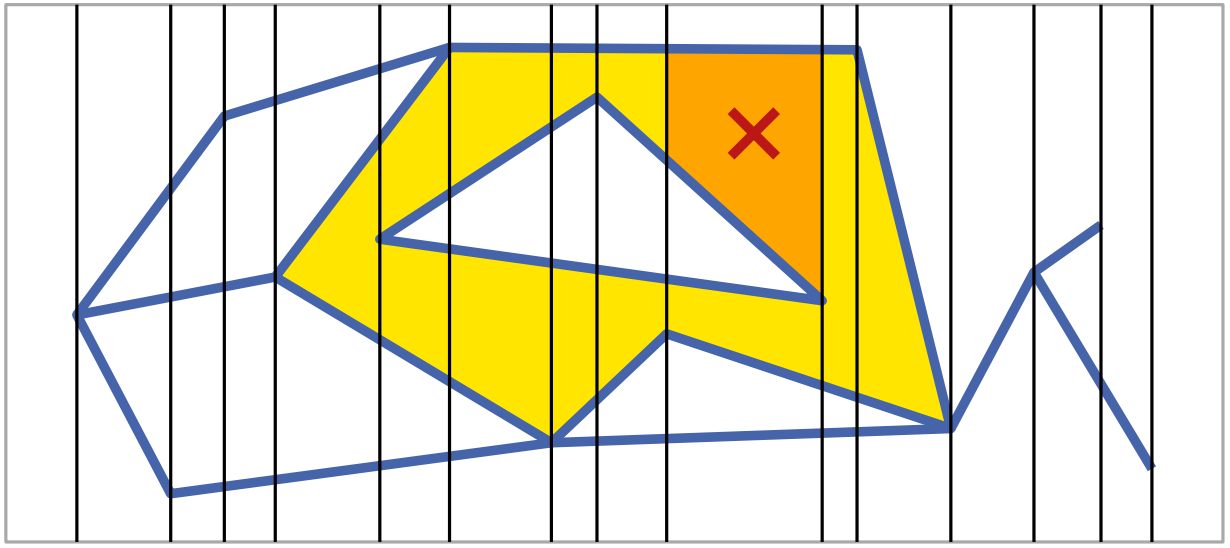
Lösung: Zerteile $\mathcal{S}$ an Punkten in vertikale Streifen. $O(\log n)$ Zeit

Anfrage:
- finde richtigen Streifen
- durchsuche diesen Streifen

} 2 binäre Suchen

Aber: Platz? $\Theta(n^2)$

Problemstellung



Ziel: Geg. eine Unterteilung $\mathcal{S}$ der Ebene mit n Strecken, erstelle Datenstruktur für schnelle Punktlokalisierung.

Lösung: Zerteile $\mathcal{S}$ an Punkten in vertikale Streifen. $O(\log n)$ Zeit

Anfrage:
- finde richtigen Streifen
- durchsuche diesen Streifen

} 2 binäre Suchen

Aber: Platz? $\Theta(n^2)$

Frage: Bsp. für untere Schranke?

Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Ziel: Finde geeignete Verfeinerung von $\mathcal{S}$ mit geringerer Komplexität!

Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Ziel: Finde geeignete Verfeinerung von $\mathcal{S}$ mit geringerer Komplexität!

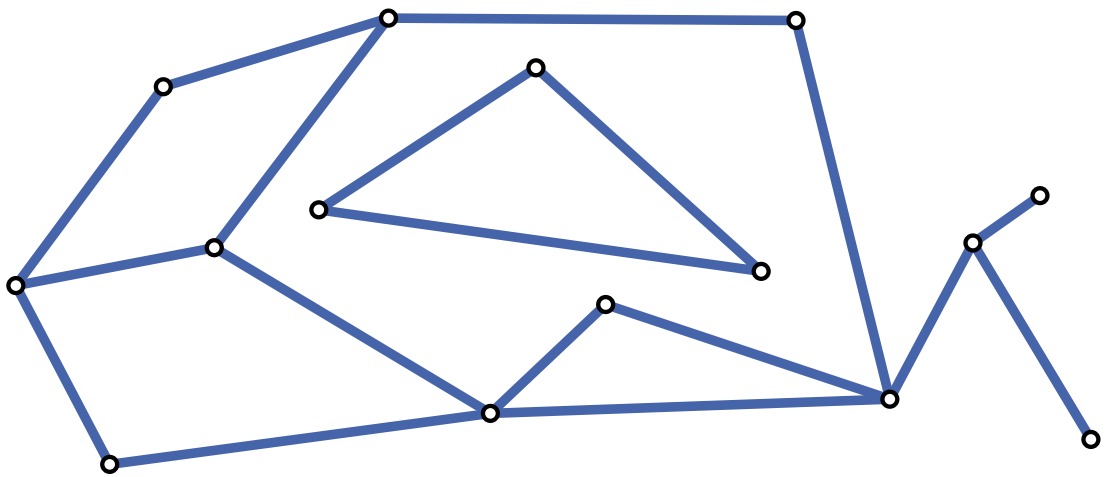
Lösung: *Trapezzerlegung* $\mathcal{T}(\mathcal{S})$

Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Ziel: Finde geeignete Verfeinerung von $\mathcal{S}$ mit geringerer Komplexität!

Lösung: *Trapezzerlegung* $\mathcal{T}(\mathcal{S})$

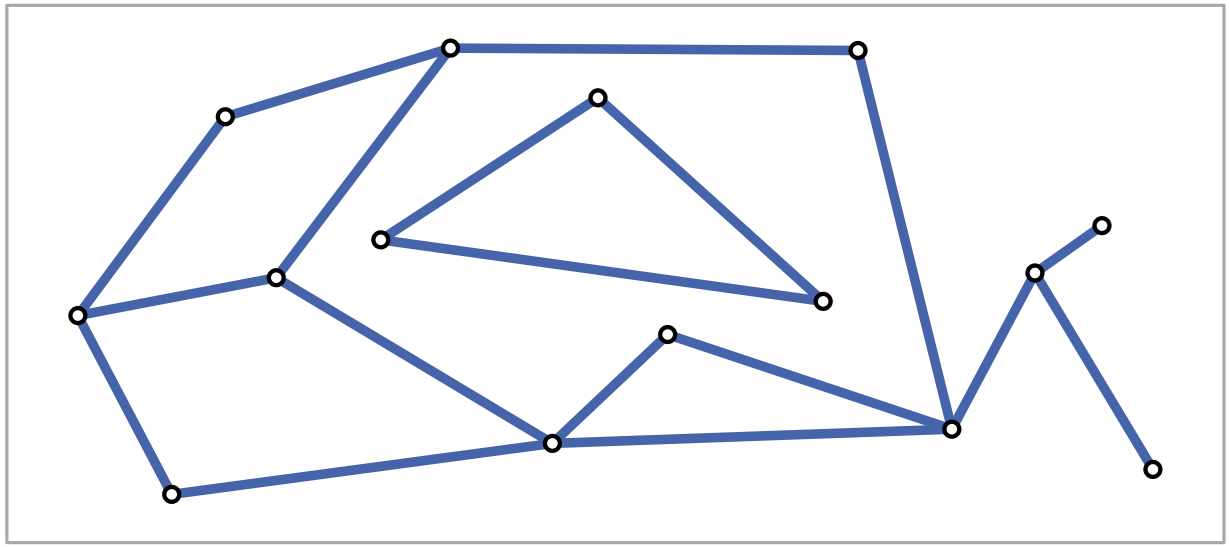


Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Ziel: Finde geeignete Verfeinerung von $\mathcal{S}$ mit geringerer Komplexität!

Lösung: *Trapezzerlegung* $\mathcal{T}(\mathcal{S})$



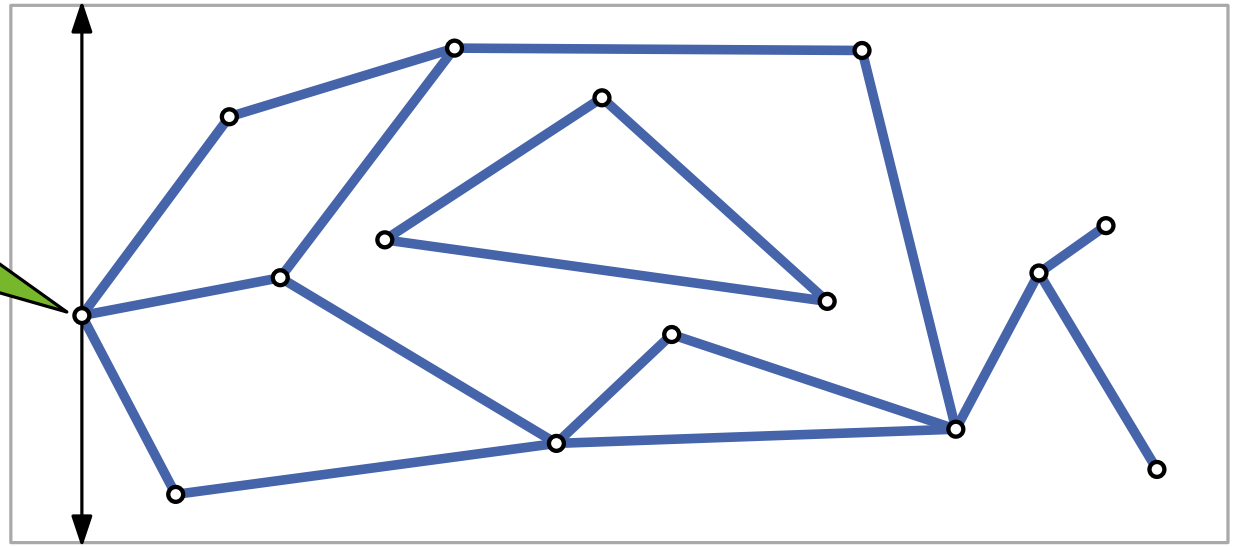
Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Ziel: Finde geeignete Verfeinerung von $\mathcal{S}$ mit geringerer Komplexität!

Lösung: *Trapezzerlegung* $\mathcal{T}(\mathcal{S})$

Kante von jedem Endpunkt vertikal nach oben und unten bis zum Nachbarsegment



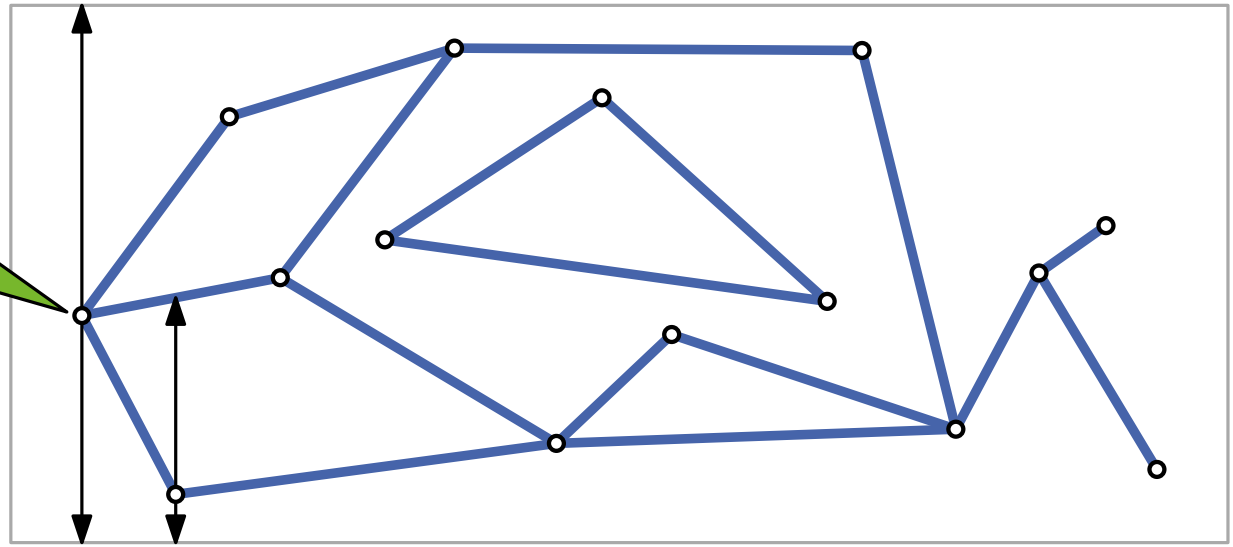
Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Ziel: Finde geeignete Verfeinerung von $\mathcal{S}$ mit geringerer Komplexität!

Lösung: *Trapezzerlegung* $\mathcal{T}(\mathcal{S})$

Kante von jedem Endpunkt vertikal nach oben und unten bis zum Nachbarsegment



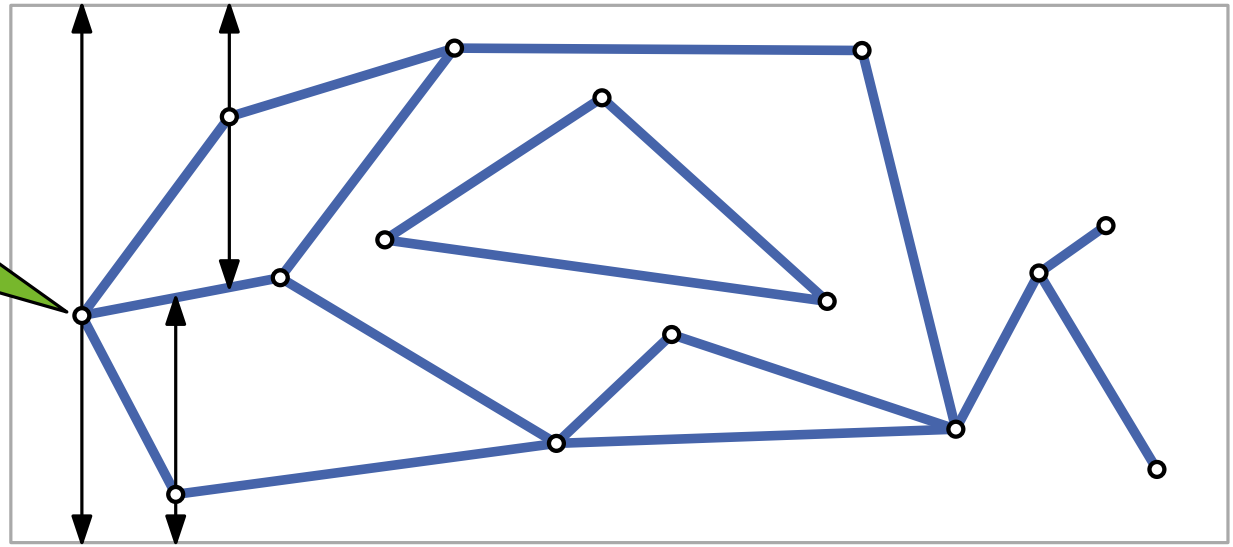
Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Ziel: Finde geeignete Verfeinerung von $\mathcal{S}$ mit geringerer Komplexität!

Lösung: *Trapezzerlegung* $\mathcal{T}(\mathcal{S})$

Kante von jedem Endpunkt vertikal nach oben und unten bis zum Nachbarsegment



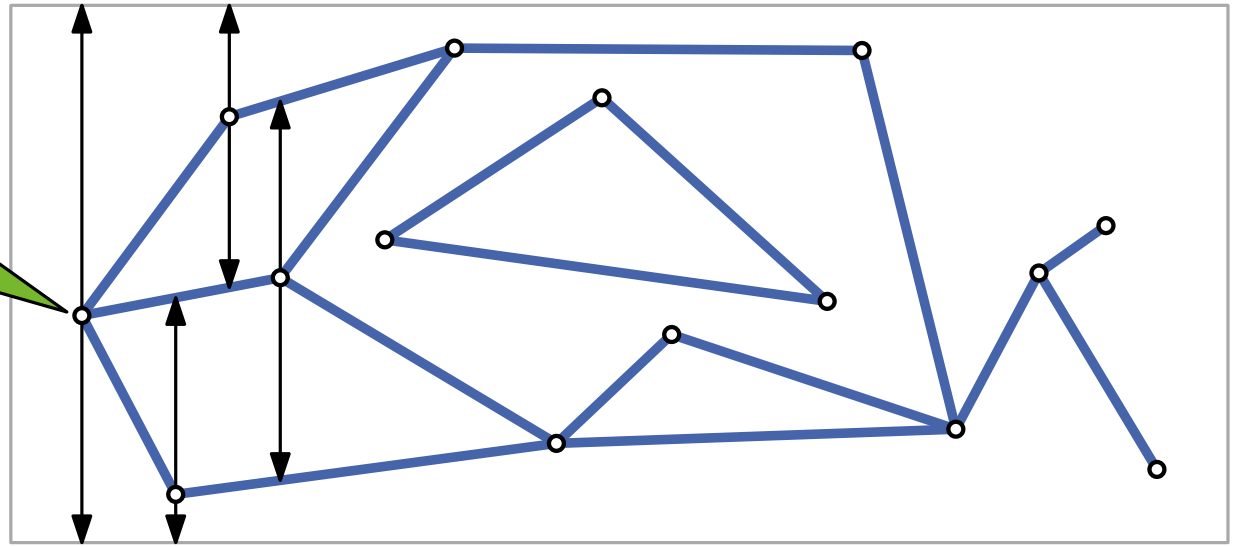
Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Ziel: Finde geeignete Verfeinerung von $\mathcal{S}$ mit geringerer Komplexität!

Lösung: *Trapezzerlegung* $\mathcal{T}(\mathcal{S})$

Kante von jedem Endpunkt vertikal nach oben und unten bis zum Nachbarsegment



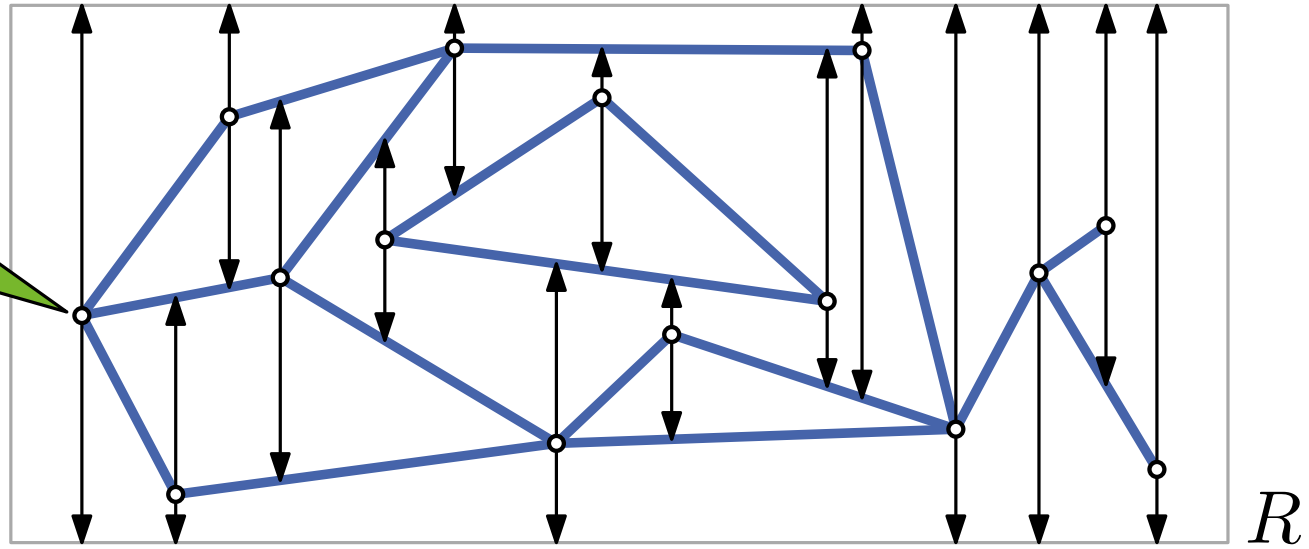
Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Ziel: Finde geeignete Verfeinerung von $\mathcal{S}$ mit geringerer Komplexität!

Lösung: *Trapezzerlegung* $\mathcal{T}(\mathcal{S})$

Kante von jedem Endpunkt vertikal nach oben und unten bis zum Nachbarsegment



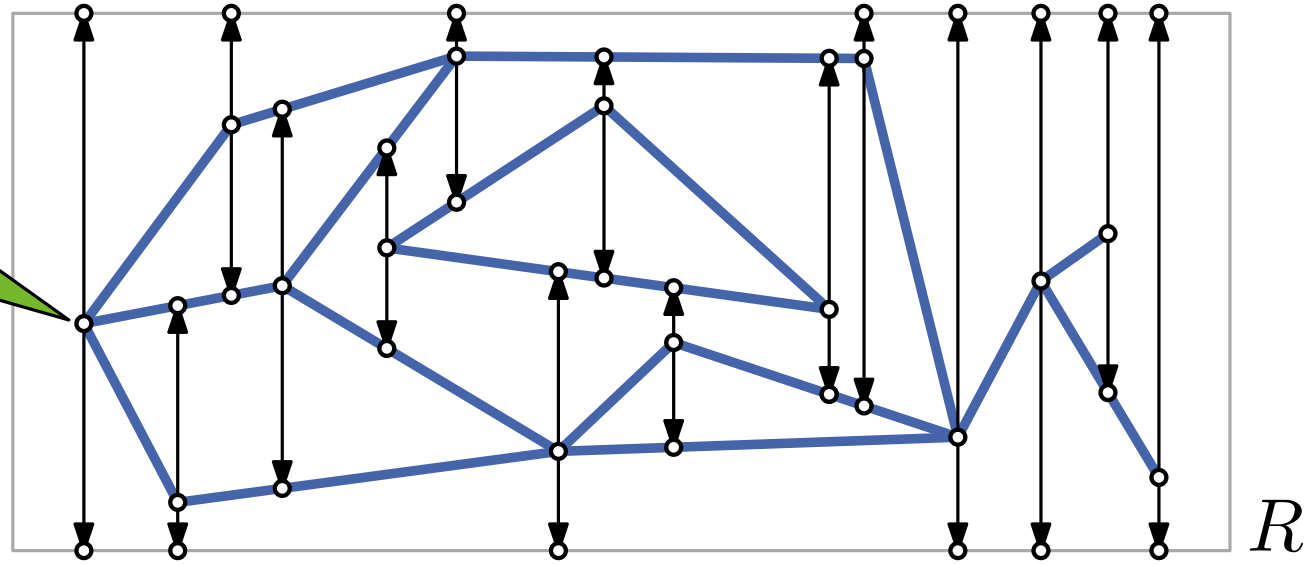
Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Ziel: Finde geeignete Verfeinerung von $\mathcal{S}$ mit geringerer Komplexität!

Lösung: *Trapezzerlegung* $\mathcal{T}(\mathcal{S})$

Kante von jedem Endpunkt vertikal nach oben und unten bis zum Nachbarsegment

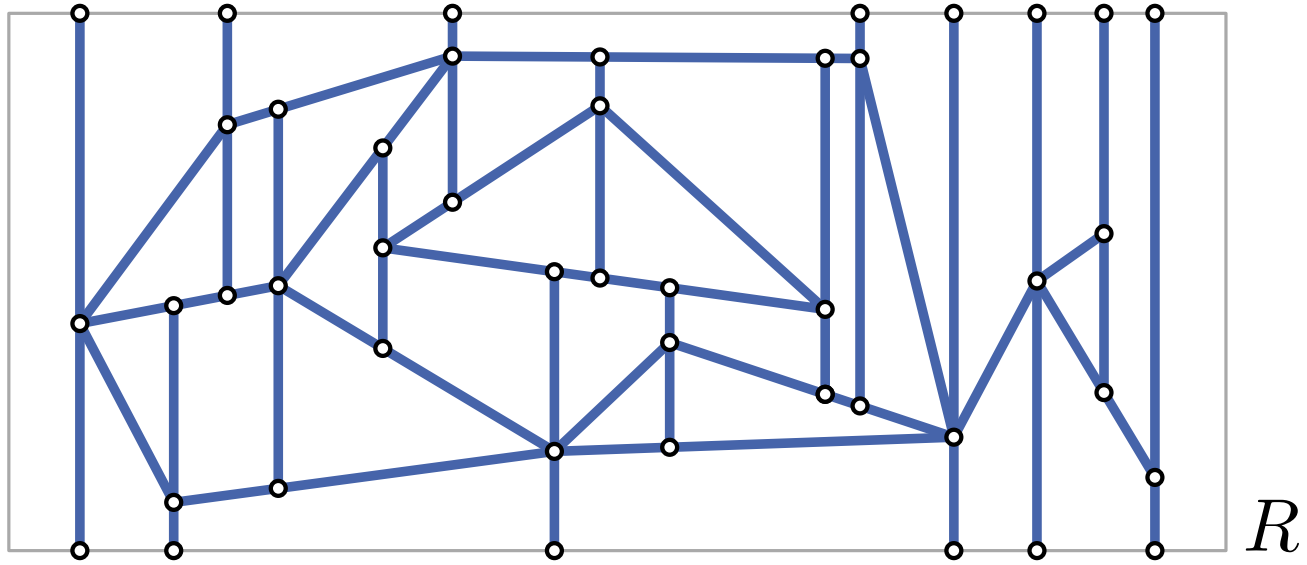


Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Ziel: Finde geeignete Verfeinerung von $\mathcal{S}$ mit geringerer Komplexität!

Lösung: *Trapezzerlegung* $\mathcal{T}(\mathcal{S})$



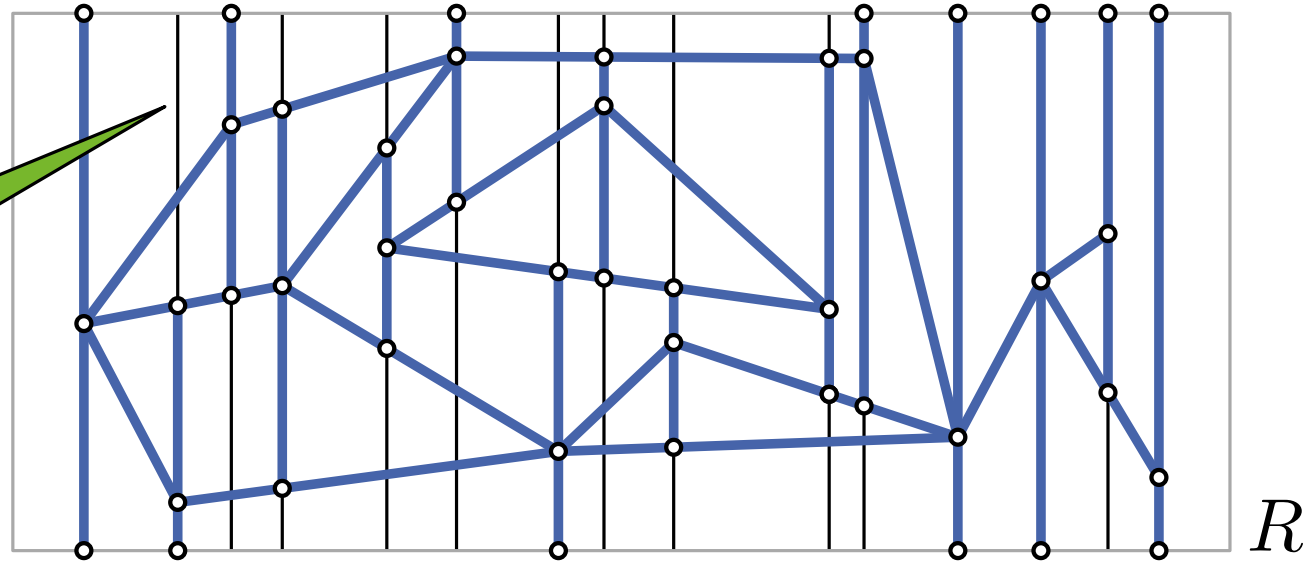
Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Ziel: Finde geeignete Verfeinerung von $\mathcal{S}$ mit geringerer Komplexität!

Lösung: *Trapezzerlegung* $\mathcal{T}(\mathcal{S})$

Vergleich mit Streifenzerlegung

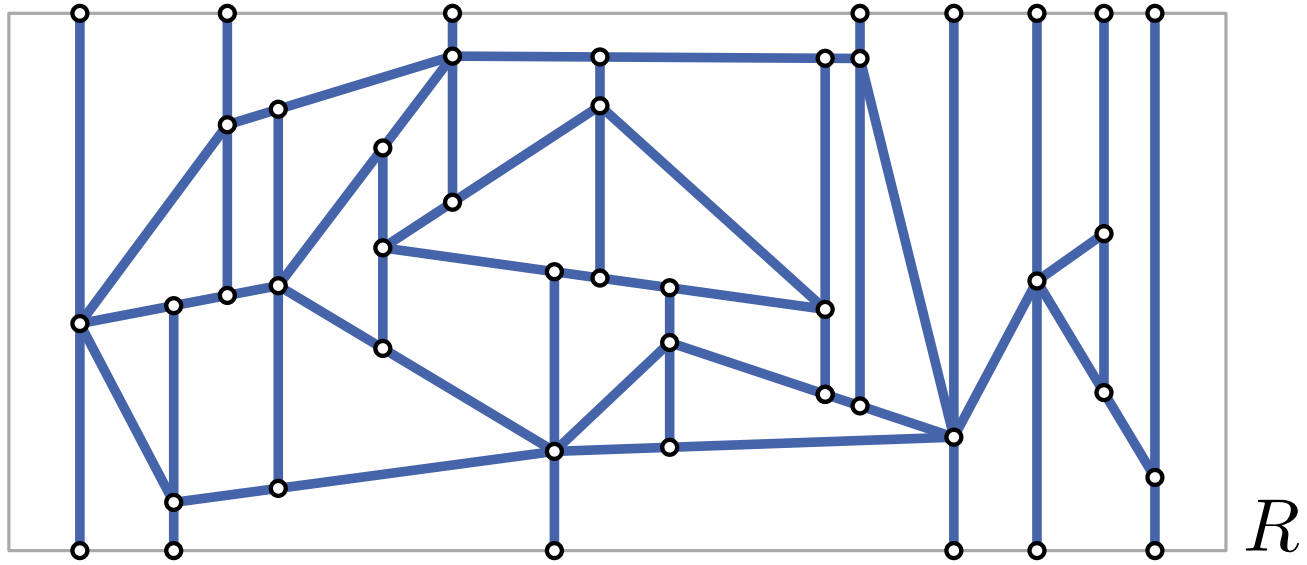


Verringern der Komplexität

Beob.: Die Streifenzerlegung ist eine Verfeinerung $\mathcal{S}'$ von $\mathcal{S}$ in (evtl. degenerierte) Trapeze.

Ziel: Finde geeignete Verfeinerung von $\mathcal{S}$ mit geringerer Komplexität!

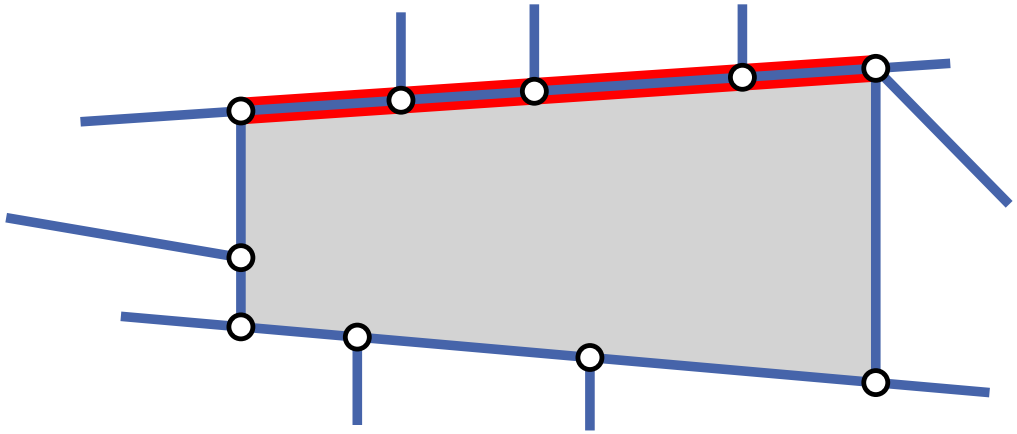
Lösung: *Trapezzerlegung* $\mathcal{T}(\mathcal{S})$



Annahme: $\mathcal{S}$ ist in *allgemeiner Lage*, d.h. keine zwei Knoten haben gleiche x-Koordinaten.

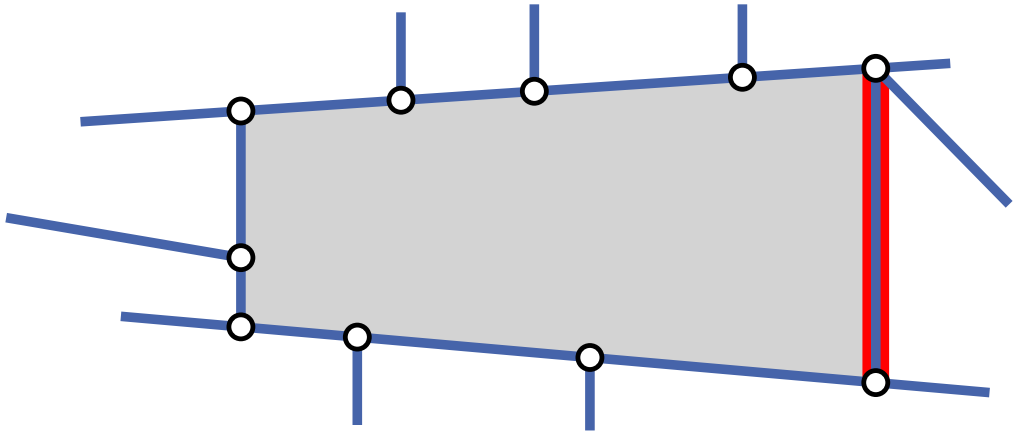
Notation

Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.



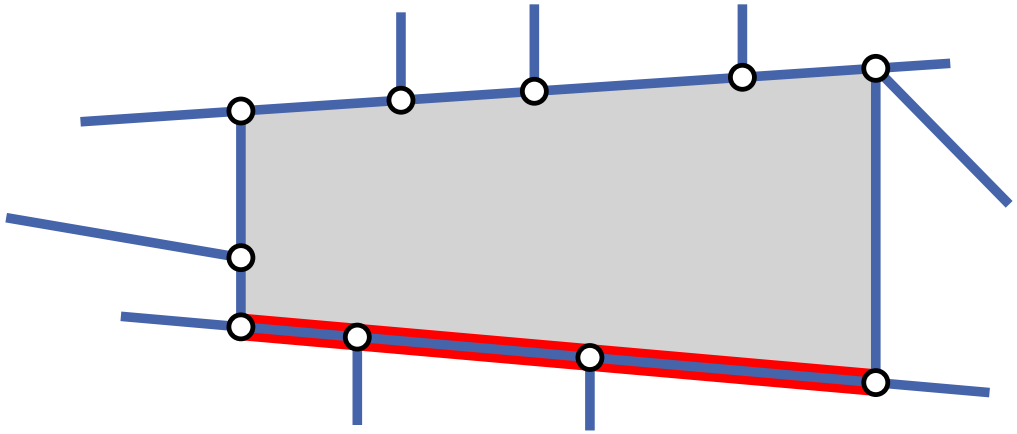
Notation

Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.



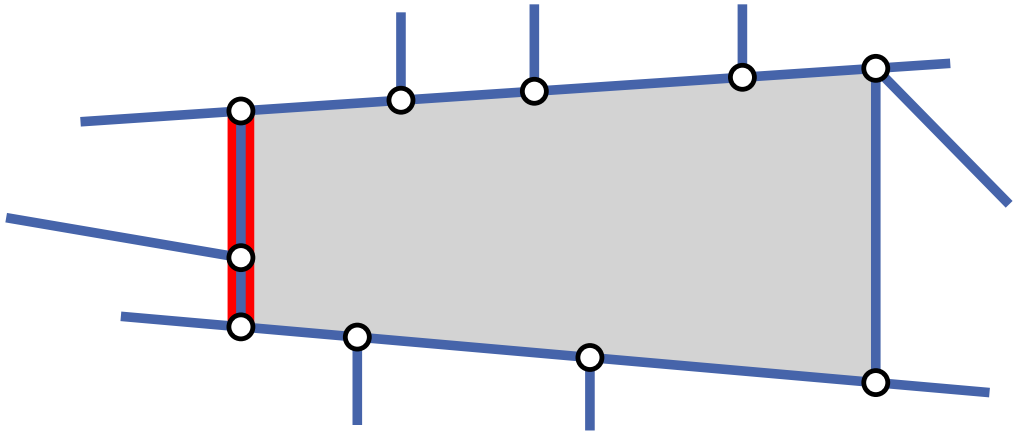
Notation

Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.



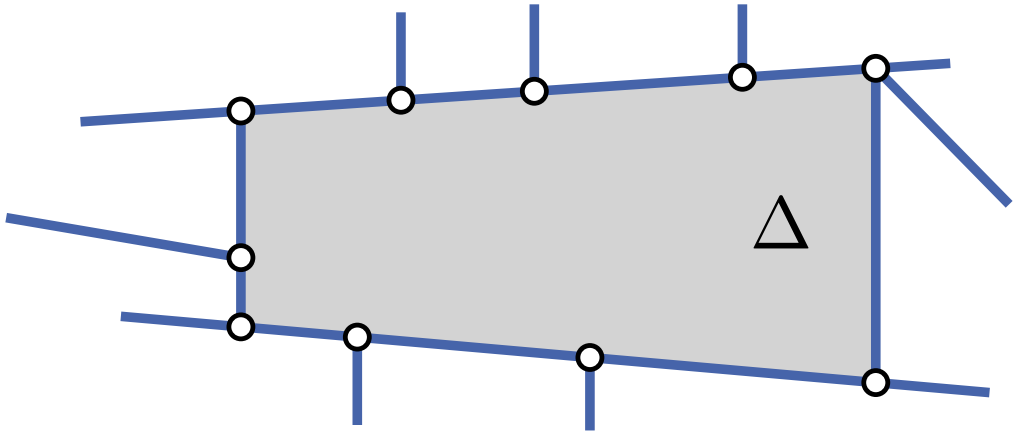
Notation

Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.



Notation

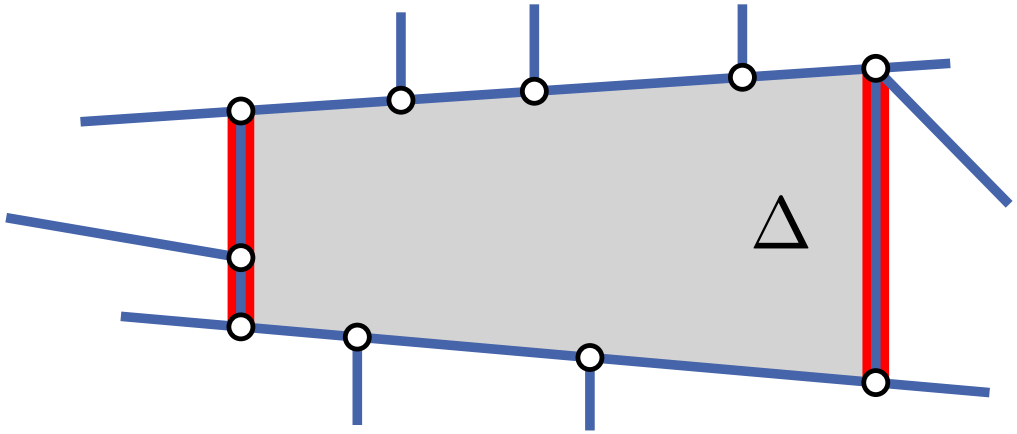
Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.



Beob.: $\mathcal{S}$ in allg. Lage $\Rightarrow$ jede Facette Δ von $\mathcal{T}(\mathcal{S})$ hat:

Notation

Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.

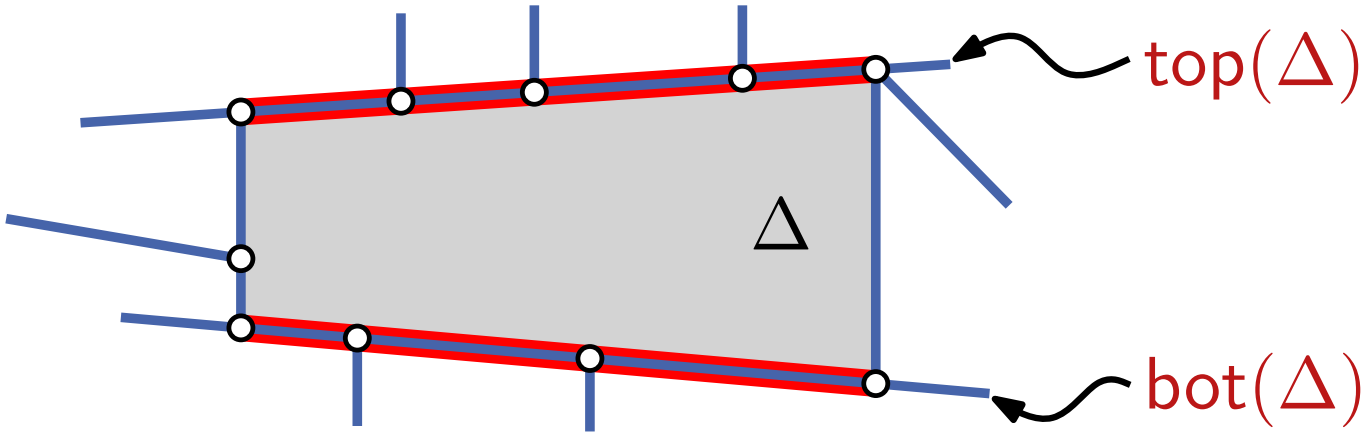


Beob.: $\mathcal{S}$ in allg. Lage $\Rightarrow$ jede Facette Δ von $\mathcal{T}(\mathcal{S})$ hat:

- ein oder zwei vertikale Seiten

Notation

Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.

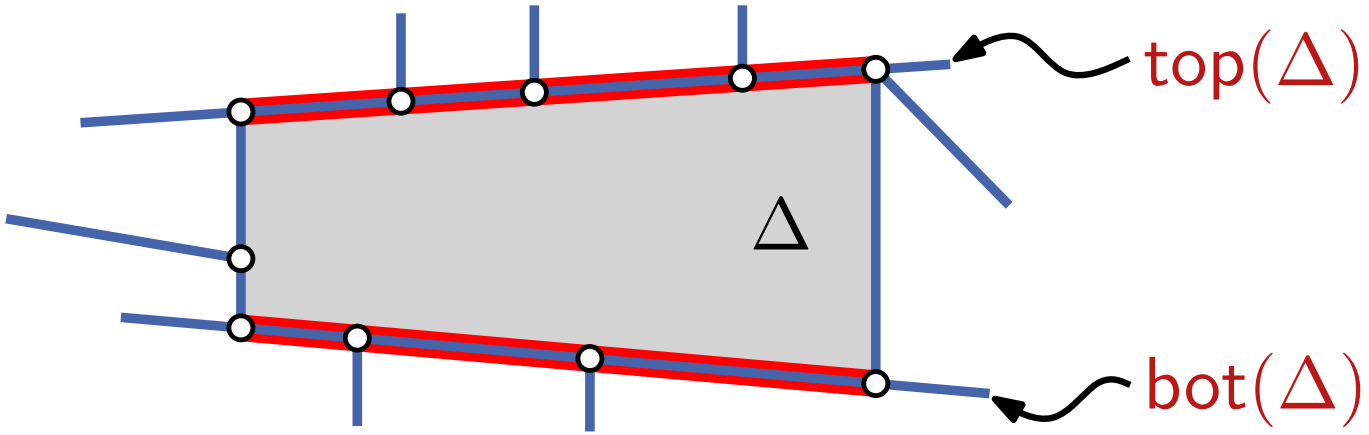


Beob.: $\mathcal{S}$ in allg. Lage $\Rightarrow$ jede Facette Δ von $\mathcal{T}(\mathcal{S})$ hat:

- ein oder zwei vertikale Seiten
- zwei nicht-vertikale Seiten

Notation

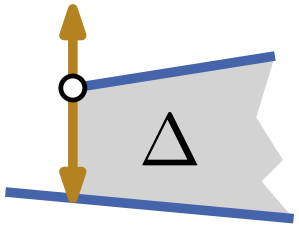
Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.



Beob.: $\mathcal{S}$ in allg. Lage $\Rightarrow$ jede Facette Δ von $\mathcal{T}(\mathcal{S})$ hat:

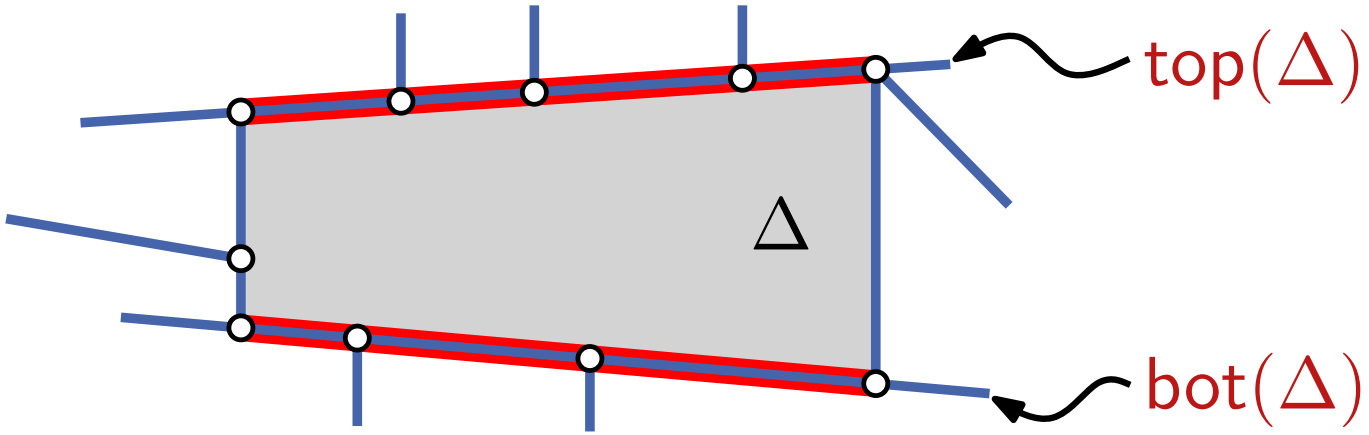
- ein oder zwei vertikale Seiten
- zwei nicht-vertikale Seiten

Linke Seite:



Notation

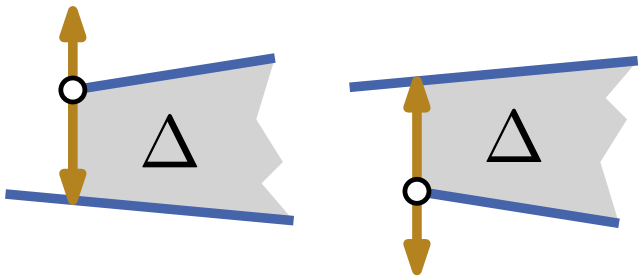
Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.



Beob.: $\mathcal{S}$ in allg. Lage $\Rightarrow$ jede Facette Δ von $\mathcal{T}(\mathcal{S})$ hat:

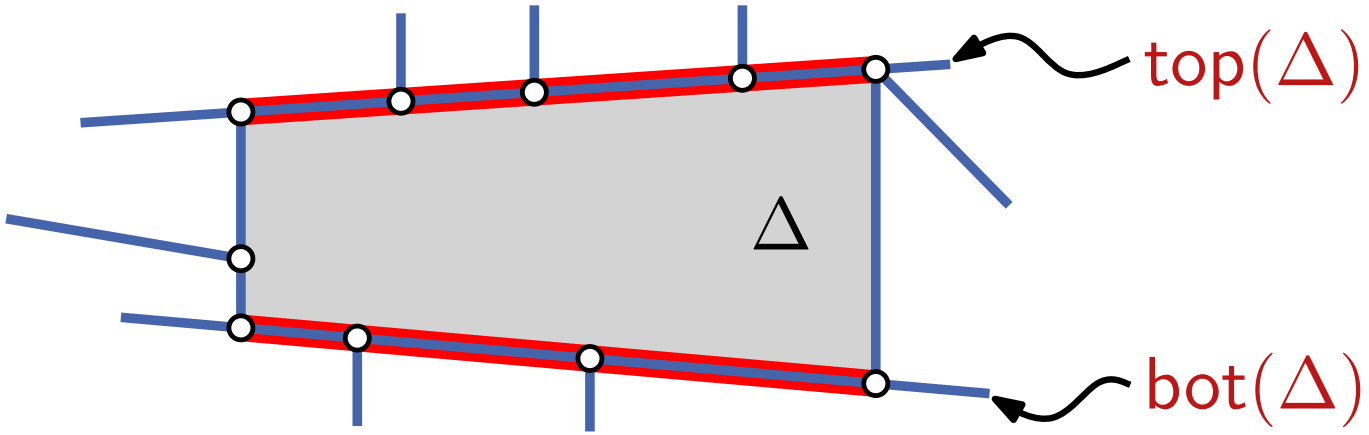
- ein oder zwei vertikale Seiten
- zwei nicht-vertikale Seiten

Linke Seite:



Notation

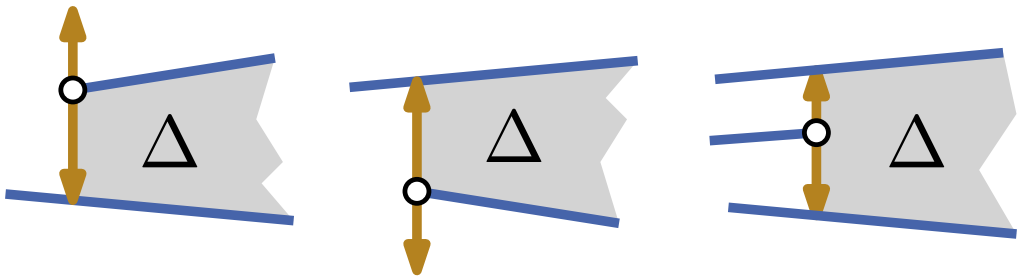
Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.



Beob.: $\mathcal{S}$ in allg. Lage $\Rightarrow$ jede Facette Δ von $\mathcal{T}(\mathcal{S})$ hat:

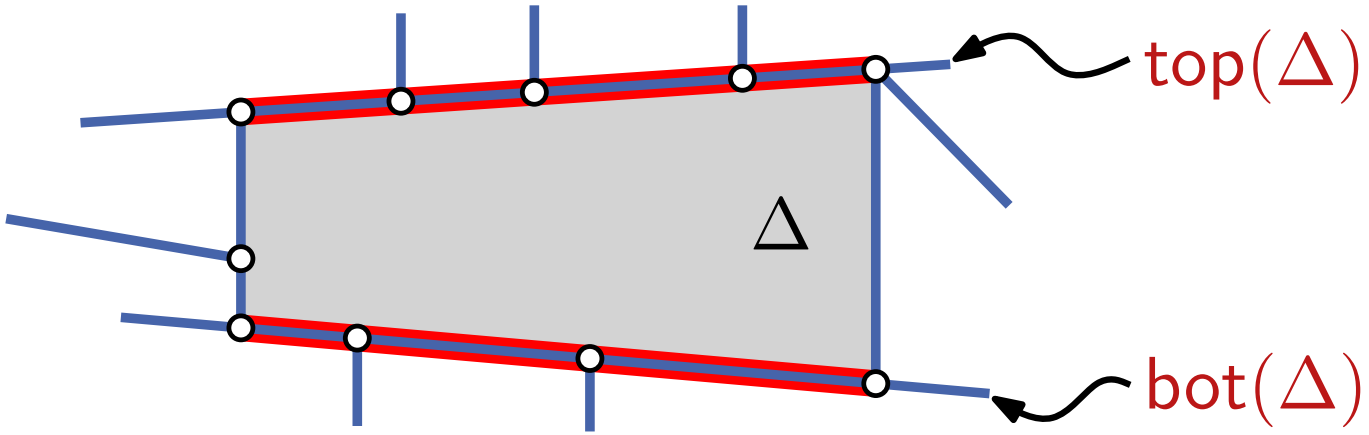
- ein oder zwei vertikale Seiten
- zwei nicht-vertikale Seiten

Linke Seite:



Notation

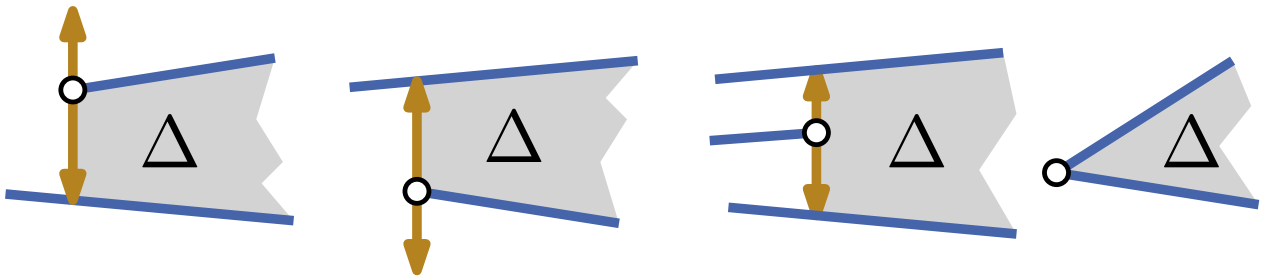
Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.



Beob.: $\mathcal{S}$ in allg. Lage $\Rightarrow$ jede Facette Δ von $\mathcal{T}(\mathcal{S})$ hat:

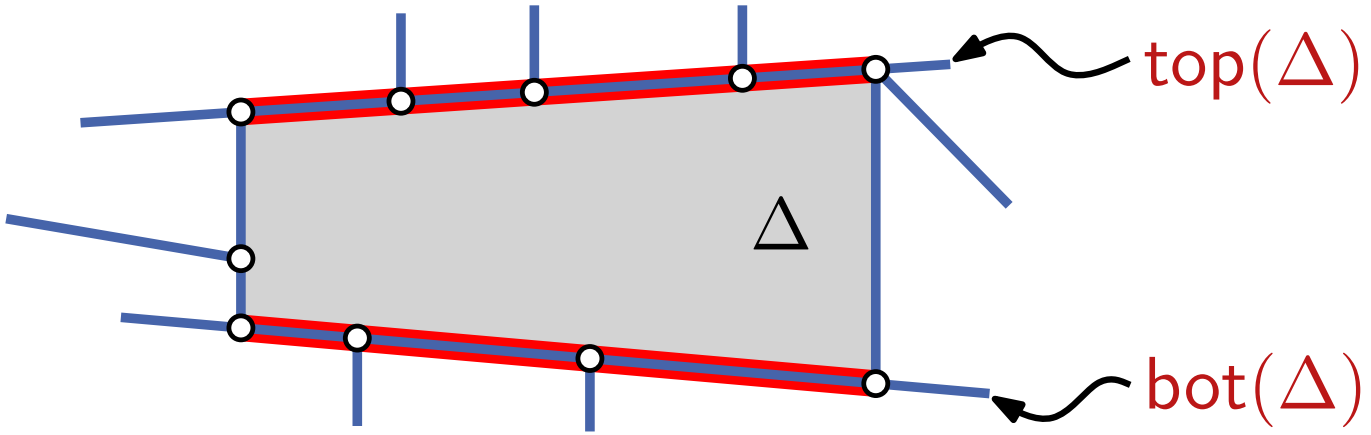
- ein oder zwei vertikale Seiten
- zwei nicht-vertikale Seiten

Linke Seite:



Notation

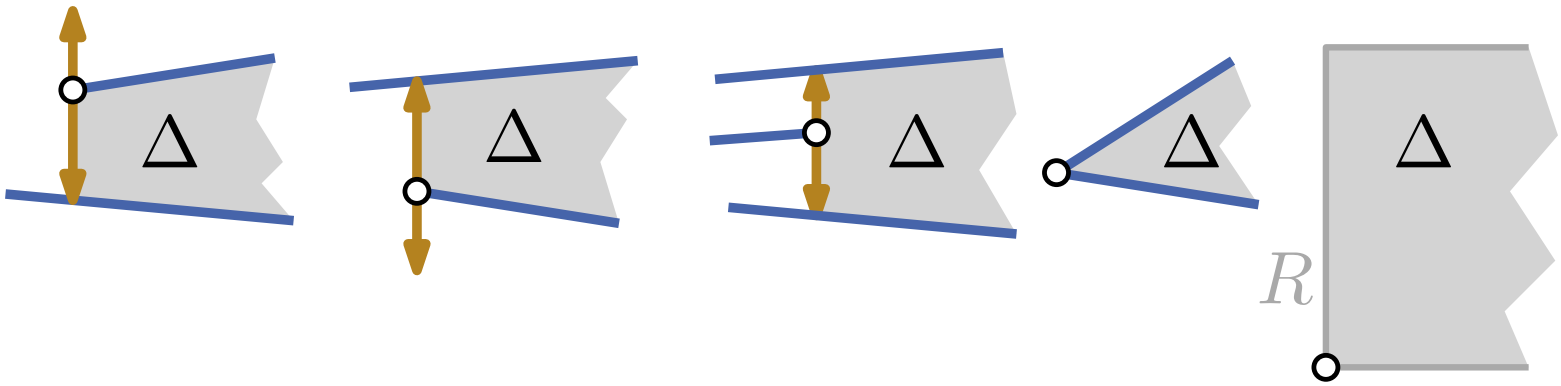
Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.



Beob.: $\mathcal{S}$ in allg. Lage $\Rightarrow$ jede Facette Δ von $\mathcal{T}(\mathcal{S})$ hat:

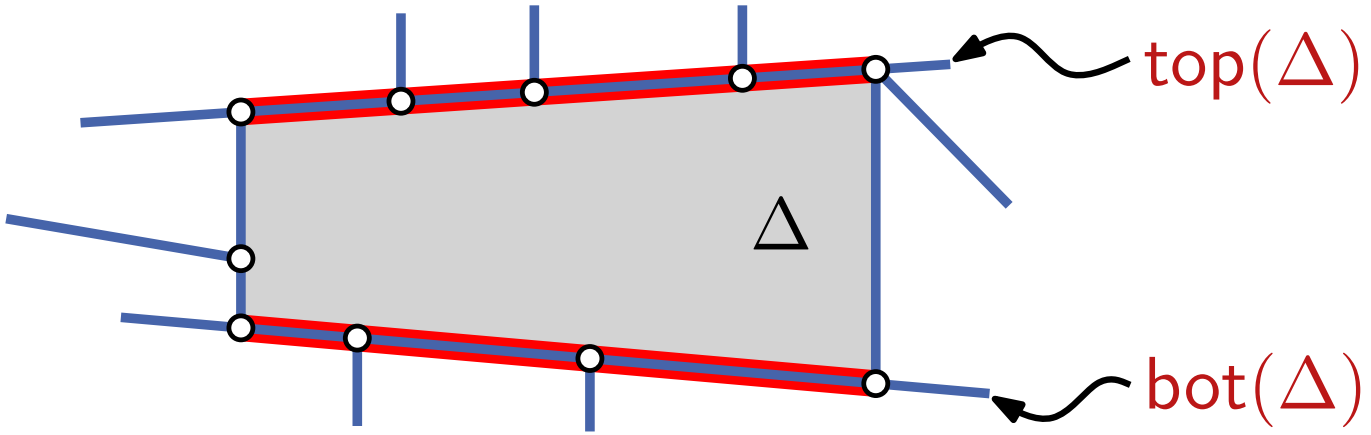
- ein oder zwei vertikale Seiten
- zwei nicht-vertikale Seiten

Linke Seite:



Notation

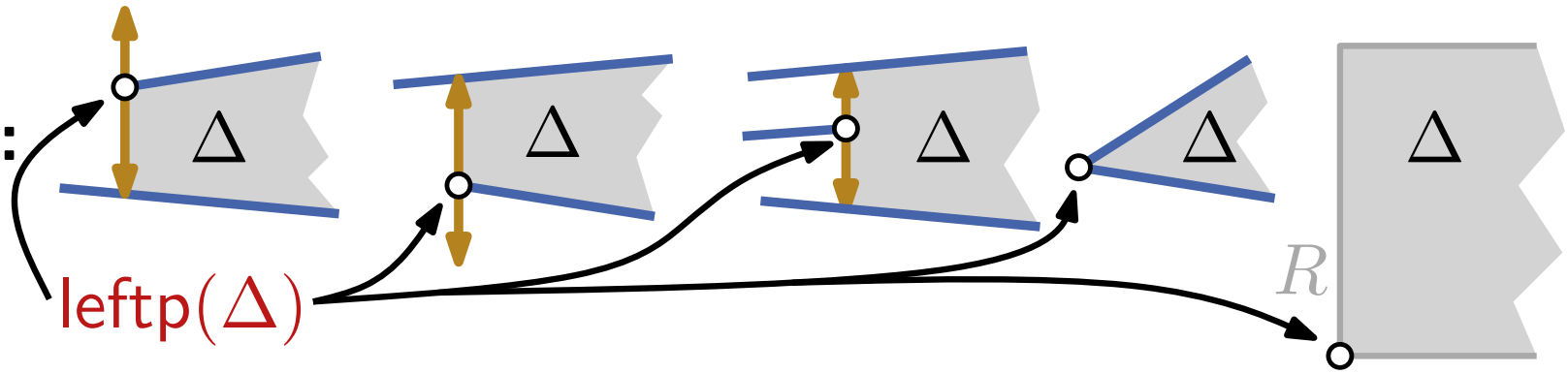
Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.



Beob.: $\mathcal{S}$ in allg. Lage $\Rightarrow$ jede Facette Δ von $\mathcal{T}(\mathcal{S})$ hat:

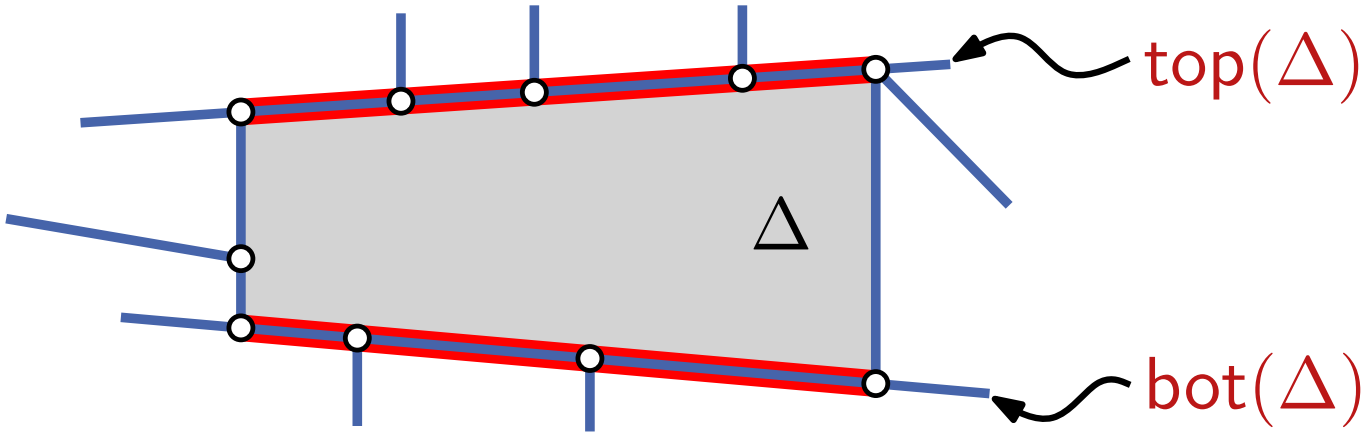
- ein oder zwei vertikale Seiten
- zwei nicht-vertikale Seiten

Linke Seite:



Notation

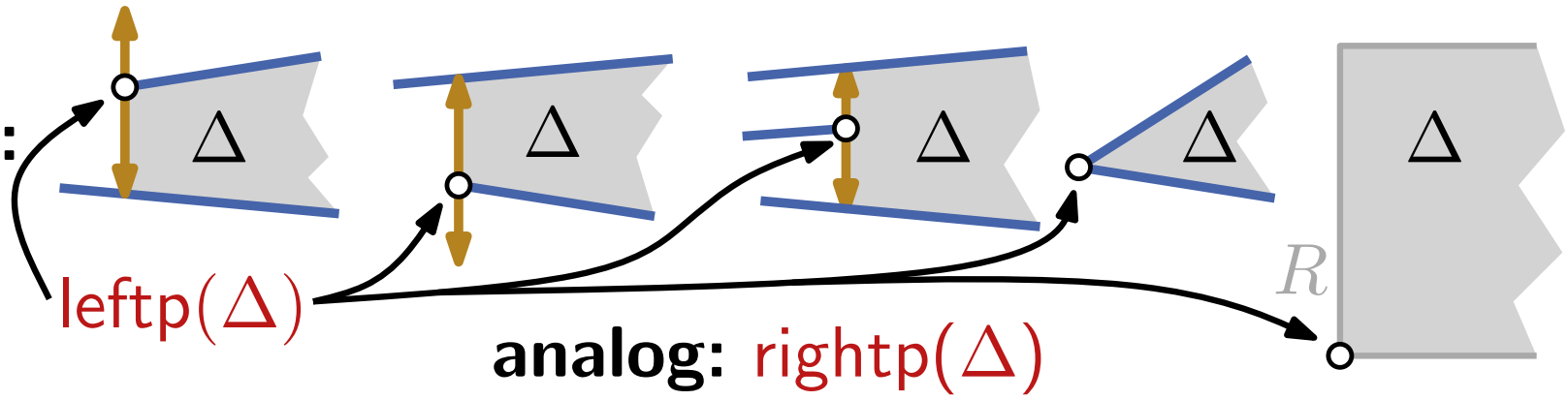
Definition: Eine *Seite* einer Facette von $\mathcal{T}(\mathcal{S})$ ist eine maximal lange Teilstrecke der Facettengrenze.



Beob.: $\mathcal{S}$ in allg. Lage $\Rightarrow$ jede Facette Δ von $\mathcal{T}(\mathcal{S})$ hat:

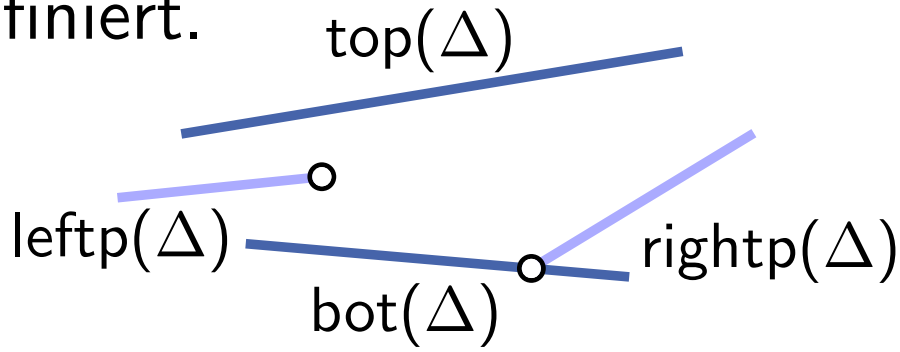
- ein oder zwei vertikale Seiten
- zwei nicht-vertikale Seiten

Linke Seite:



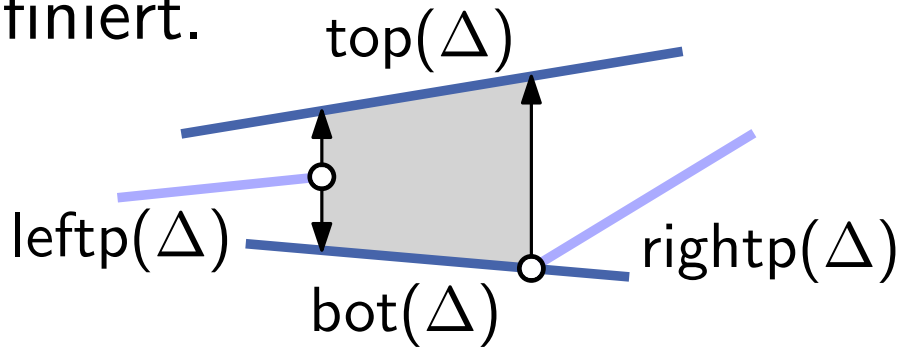
Komplexität der Trapezzerlegung

Beob.: Ein Trapez Δ wird eindeutig durch $\text{bot}(\Delta)$, $\text{top}(\Delta)$, $\text{leftp}(\Delta)$ und $\text{rightp}(\Delta)$ definiert.



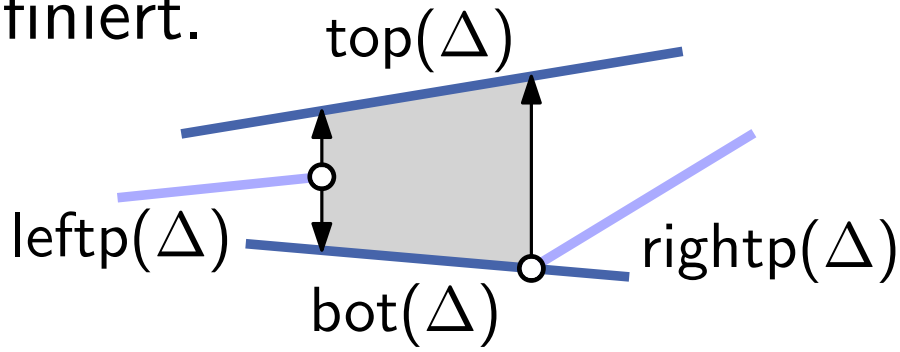
Komplexität der Trapezzerlegung

Beob.: Ein Trapez Δ wird eindeutig durch $\text{bot}(\Delta)$, $\text{top}(\Delta)$, $\text{leftp}(\Delta)$ und $\text{rightp}(\Delta)$ definiert.



Komplexität der Trapezzerlegung

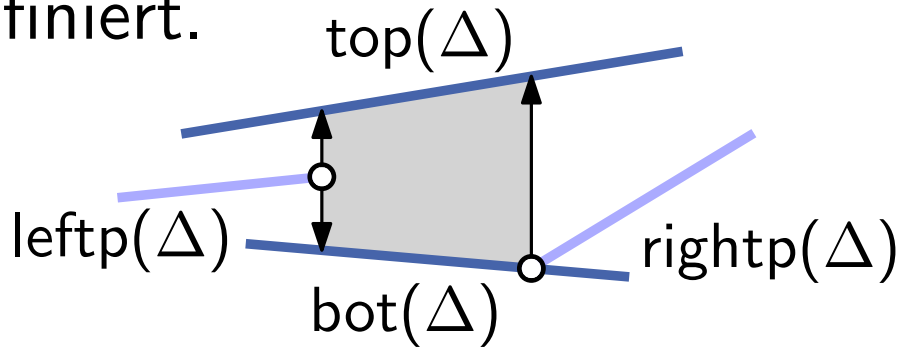
Beob.: Ein Trapez Δ wird eindeutig durch $\text{bot}(\Delta)$, $\text{top}(\Delta)$, $\text{leftp}(\Delta)$ und $\text{rightp}(\Delta)$ definiert.



Lemma: Die Trapezzerlegung $\mathcal{T}(\mathcal{S})$ einer Menge $\mathcal{S}$ von n Strecken in allgemeiner Lage enthält höchstens Knoten und höchstens Trapeze.

Komplexität der Trapezzerlegung

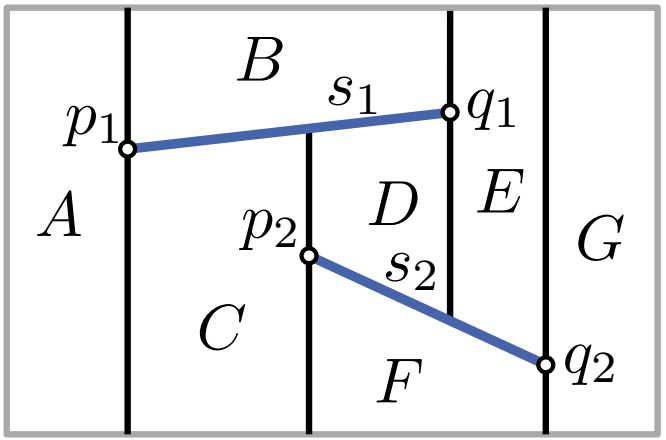
Beob.: Ein Trapez Δ wird eindeutig durch bot(Δ), top(Δ), leftp(Δ) und rightp(Δ) definiert.



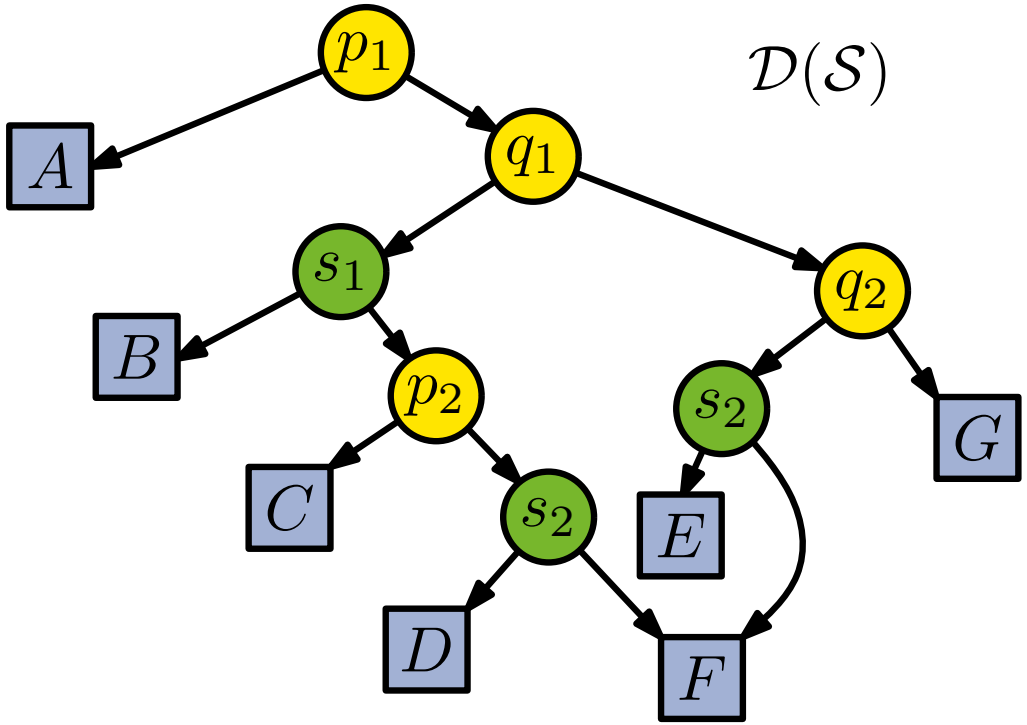
Lemma: Die Trapezzerlegung $\mathcal{T}(\mathcal{S})$ einer Menge $\mathcal{S}$ von n Strecken in allgemeiner Lage enthält höchstens $6n + 4$ Knoten und höchstens $3n + 1$ Trapeze.

Suchstruktur

Ziel: Berechne die Trapezzerlegung $\mathcal{T}(\mathcal{S})$ und gleichzeitig eine Datenstruktur $\mathcal{D}(\mathcal{S})$ zur Lokalisierung in $\mathcal{T}(\mathcal{S})$.



$\mathcal{T}(\mathcal{S})$



$\mathcal{D}(\mathcal{S})$ ist ein DAG mit:

- p — x-Knoten für Punkt p testet auf links/rechts von p
- s — y-Knoten für Strecke s testet auf oberhalb/unterhalb von s
- Δ — Blattknoten für Trapez Δ

Inkrementeller Algorithmus

Trapezzerlegung($\mathcal{S}$)

Input: Menge $\mathcal{S} = \{s_1, \ldots, s_n\}$ von kreuzungsfreien Strecken
Output: Trapezzerlegung $\mathcal{T}(\mathcal{S})$ und Suchstruktur $\mathcal{D}(\mathcal{S})$
Initialisiere $\mathcal{T}$ und $\mathcal{D}$ für $R = \text{BBox}(\mathcal{S})$

for $i \leftarrow 1$ **to** n **do**
- $H \leftarrow \{\Delta \in \mathcal{T} \mid \Delta \cap s_i \neq \emptyset\}$
- $\mathcal{T} \leftarrow \mathcal{T} \setminus H$
- $\mathcal{T} \leftarrow \mathcal{T} \cup$ neue durch s_i erzeugte Trapeze
- $\mathcal{D} \leftarrow$ ersetze Blätter für H durch Blätter für neue Trapeze

return $(\mathcal{T}, \mathcal{D})$

Inkrementeller Algorithmus

Trapezzerlegung($\mathcal{S}$)

Input: Menge $\mathcal{S} = \{s_1, \ldots, s_n\}$ von kreuzungsfreien Strecken
Output: Trapezzerlegung $\mathcal{T}(\mathcal{S})$ und Suchstruktur $\mathcal{D}(\mathcal{S})$
Initialisiere $\mathcal{T}$ und $\mathcal{D}$ für $R = \text{BBox}(\mathcal{S})$

for $i \leftarrow 1$ **to** n **do**
- $H \leftarrow \{\Delta \in \mathcal{T} \mid \Delta \cap s_i \neq \emptyset\}$
- $\mathcal{T} \leftarrow \mathcal{T} \setminus H$
- $\mathcal{T} \leftarrow \mathcal{T} \cup$ neue durch s_i erzeugte Trapeze
- $\mathcal{D} \leftarrow$ ersetze Blätter für H durch Blätter für neue Trapeze

return $(\mathcal{T}, \mathcal{D})$

Problem: Größe von $\mathcal{D}$ und Anfragezeit sind abhängig von Einfügereihenfolge

Inkrementeller Algorithmus

Trapezzerlegung($\mathcal{S}$)

Input: Menge $\mathcal{S} = \{s_1, \ldots, s_n\}$ von kreuzungsfreien Strecken
Output: Trapezzerlegung $\mathcal{T}(\mathcal{S})$ und Suchstruktur $\mathcal{D}(\mathcal{S})$
Initialisiere $\mathcal{T}$ und $\mathcal{D}$ für $R = \text{BBox}(\mathcal{S})$
$\mathcal{S} \leftarrow$ **RandomPermutation**($\mathcal{S}$)
for $i \leftarrow 1$ **to** n **do**
  $H \leftarrow \{\Delta \in \mathcal{T} \mid \Delta \cap s_i \neq \emptyset\}$
  $\mathcal{T} \leftarrow \mathcal{T} \setminus H$
  $\mathcal{T} \leftarrow \mathcal{T} \cup$ neue durch s_i erzeugte Trapeze
  $\mathcal{D} \leftarrow$ ersetze Blätter für H durch Blätter für neue Trapeze
return $(\mathcal{T}, \mathcal{D})$

Problem: Größe von $\mathcal{D}$ und Anfragezeit sind abhängig von Einfügereihenfolge

Lösung: Randomisierung!

Randomisierter Inkrementeller Algorithmus

Invariante: $\mathcal{T}$ ist Trapezzerlegung für $\mathcal{S}_i = \{s_1, \ldots, s_i\}$ und $\mathcal{D}$ passende Suchstruktur

Randomisierter Inkrementeller Algorithmus

Invariante: $\mathcal{T}$ ist Trapezzerlegung für $\mathcal{S}_i = \{s_1, \ldots, s_i\}$ und $\mathcal{D}$ passende Suchstruktur

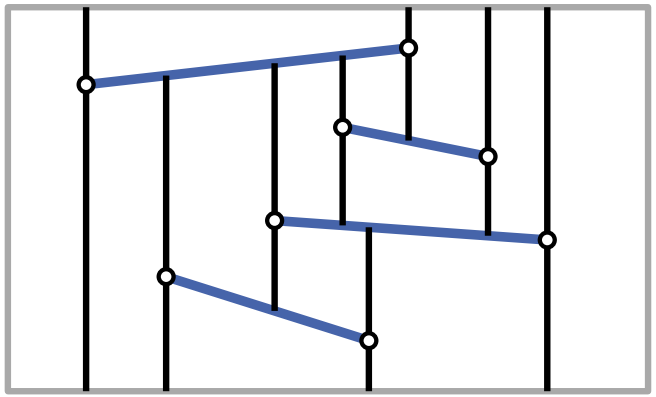
Initialisierung: $\mathcal{T} = \mathcal{T}(\emptyset) = R$ und $\mathcal{D} = (R, \emptyset)$

Randomisierter Inkrementeller Algorithmus

Invariante: $\mathcal{T}$ ist Trapezzerlegung für $\mathcal{S}_i = \{s_1, \ldots, s_i\}$ und $\mathcal{D}$ passende Suchstruktur

Initialisierung: $\mathcal{T} = \mathcal{T}(\emptyset) = R$ und $\mathcal{D} = (R, \emptyset)$

Schritt 1: $H \leftarrow \{\Delta \in \mathcal{T} \mid \Delta \cap s_i \neq \emptyset\}$

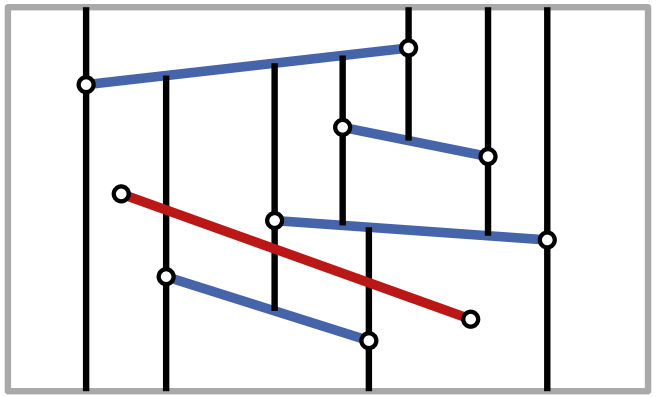


Randomisierter Inkrementeller Algorithmus

Invariante: $\mathcal{T}$ ist Trapezzerlegung für $\mathcal{S}_i = \{s_1, \ldots, s_i\}$ und $\mathcal{D}$ passende Suchstruktur

Initialisierung: $\mathcal{T} = \mathcal{T}(\emptyset) = R$ und $\mathcal{D} = (R, \emptyset)$

Schritt 1: $H \leftarrow \{\Delta \in \mathcal{T} \mid \Delta \cap s_i \neq \emptyset\}$

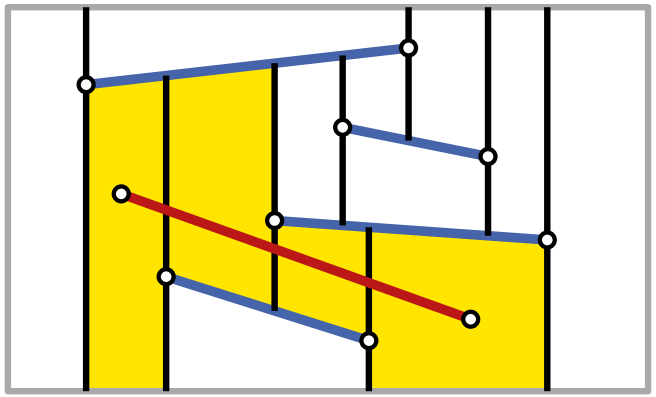


Randomisierter Inkrementeller Algorithmus

Invariante: $\mathcal{T}$ ist Trapezzerlegung für $\mathcal{S}_i = \{s_1, \ldots, s_i\}$ und $\mathcal{D}$ passende Suchstruktur

Initialisierung: $\mathcal{T} = \mathcal{T}(\emptyset) = R$ und $\mathcal{D} = (R, \emptyset)$

Schritt 1: $H \leftarrow \{\Delta \in \mathcal{T} \mid \Delta \cap s_i \neq \emptyset\}$



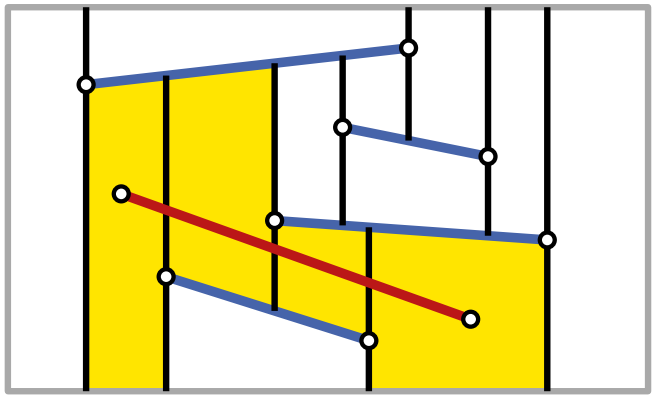
Randomisierter Inkrementeller Algorithmus

Invariante: $\mathcal{T}$ ist Trapezzerlegung für $\mathcal{S}_i = \{s_1, \ldots, s_i\}$ und $\mathcal{D}$ passende Suchstruktur

Initialisierung: $\mathcal{T} = \mathcal{T}(\emptyset) = R$ und $\mathcal{D} = (R, \emptyset)$

Schritt 1: $H \leftarrow \{\Delta \in \mathcal{T} \mid \Delta \cap s_i \neq \emptyset\}$

Aufgabe: Wie findet man die Trapezmenge H von links nach rechts?

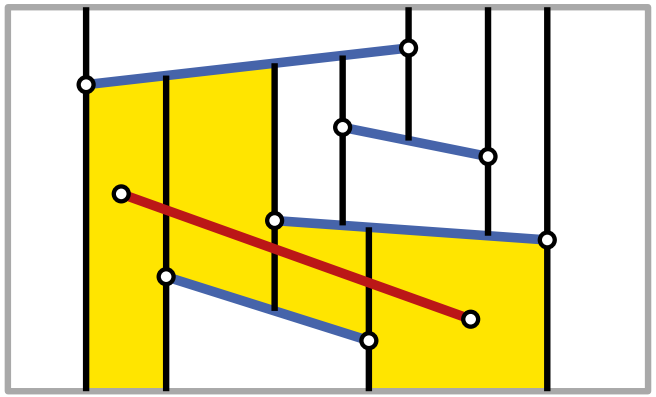


Randomisierter Inkrementeller Algorithmus

Invariante: $\mathcal{T}$ ist Trapezzerlegung für $\mathcal{S}_i = \{s_1, \ldots, s_i\}$ und $\mathcal{D}$ passende Suchstruktur

Initialisierung: $\mathcal{T} = \mathcal{T}(\emptyset) = R$ und $\mathcal{D} = (R, \emptyset)$

Schritt 1: $H \leftarrow \{\Delta \in \mathcal{T} \mid \Delta \cap s_i \neq \emptyset\}$



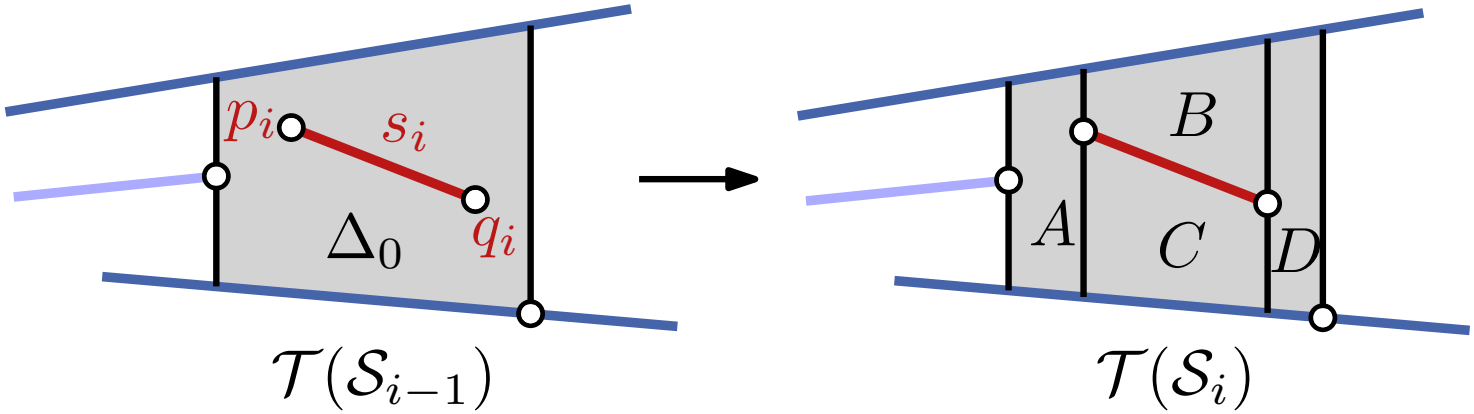
Aufgabe: Wie findet man die Trapezmenge H von links nach rechts?

$\Delta_0 \leftarrow$ FindeTrapez$(p_i, \mathcal{D})$; $j \leftarrow 0$
while rechter Endpunkt q_i rechts von rightp(Δ_j) **do**
  if rightp(Δ_j) über s_i **then**
    $\Delta_{j+1} \leftarrow$ unterer rechter Nachbar von Δ_j
  else
    $\Delta_{j+1} \leftarrow$ oberer rechter Nachbar von Δ_j
  $j \leftarrow j + 1$
return $\Delta_0, \ldots, \Delta_j$

Update von $\mathcal{T}(\mathcal{S})$ und $\mathcal{D}(\mathcal{S})$

Schritt 2: Aktualisiere $\mathcal{T}$ und $\mathcal{D}$

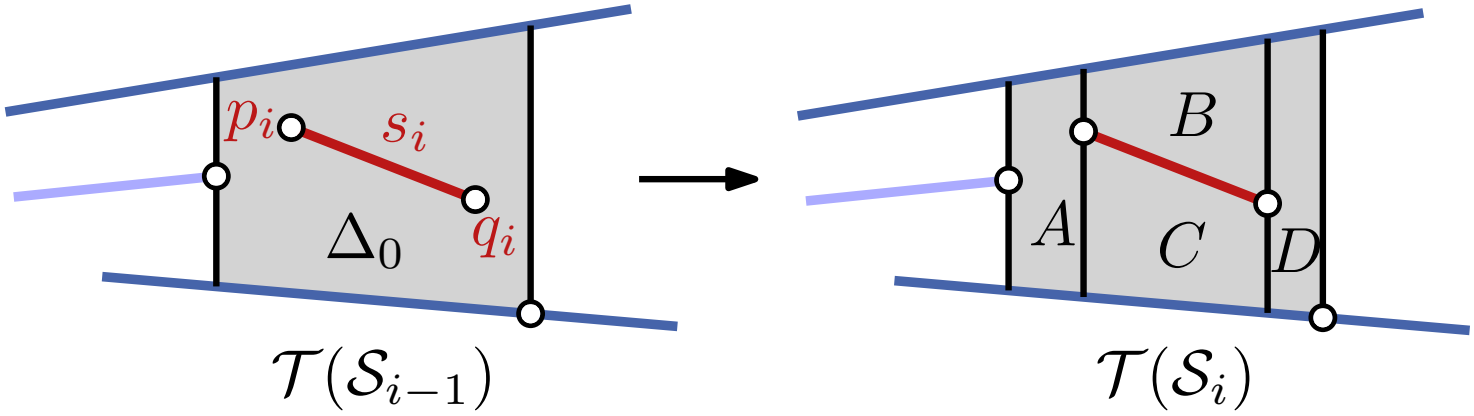
- Fall 1: $s_i \subset \Delta_0$

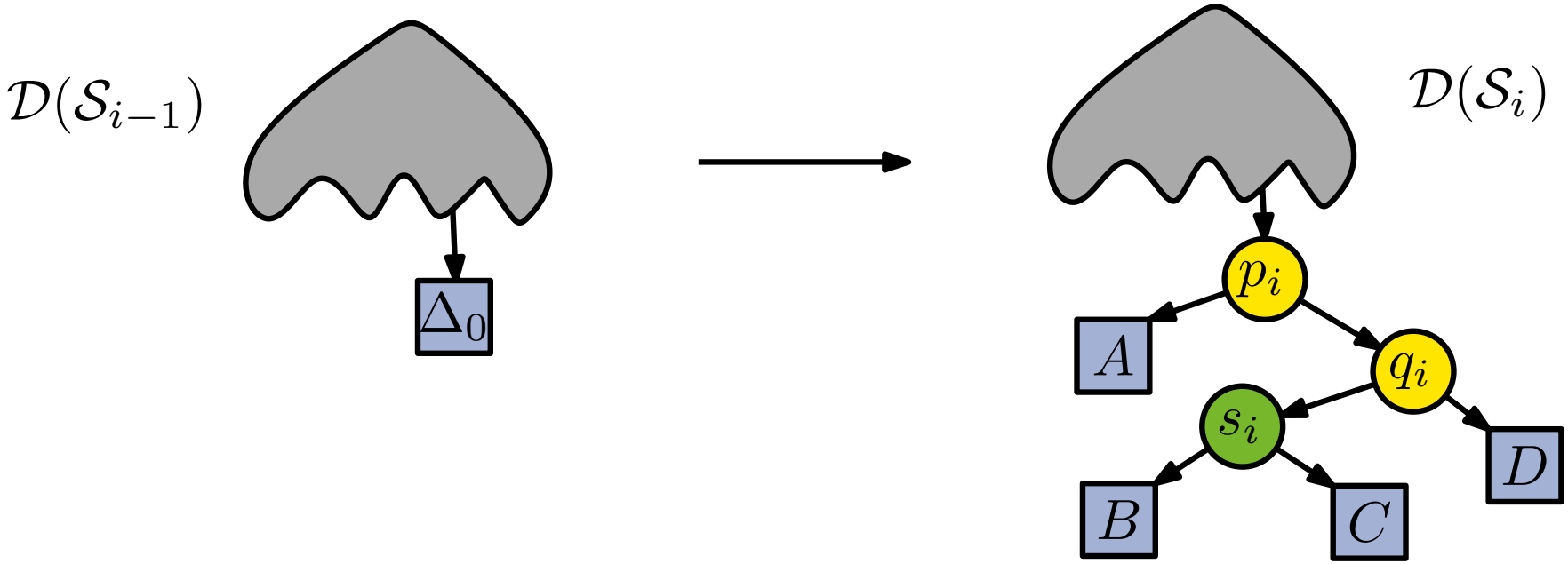


Update von $\mathcal{T}(\mathcal{S})$ und $\mathcal{D}(\mathcal{S})$

Schritt 2: Aktualisiere $\mathcal{T}$ und $\mathcal{D}$

- Fall 1: $s_i \subset \Delta_0$

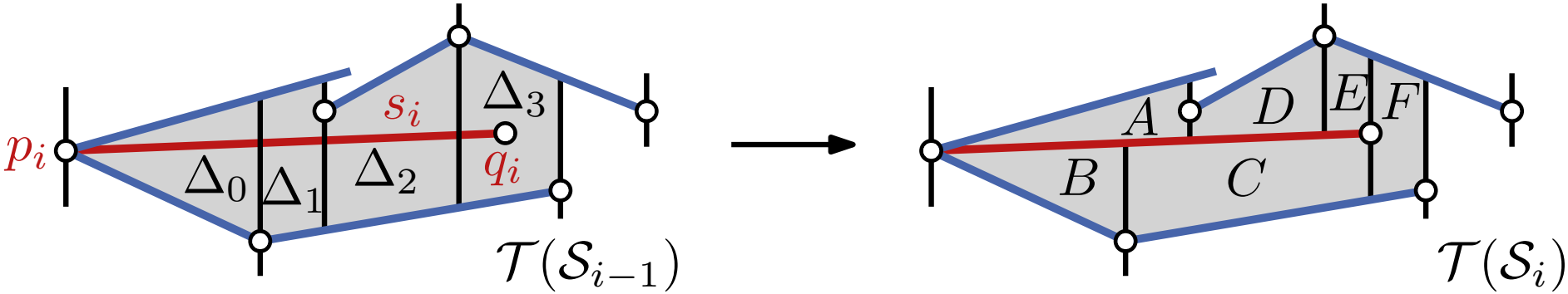




Update von $\mathcal{T}(\mathcal{S})$ und $\mathcal{D}(\mathcal{S})$

Schritt 2: Aktualisiere $\mathcal{T}$ und $\mathcal{D}$

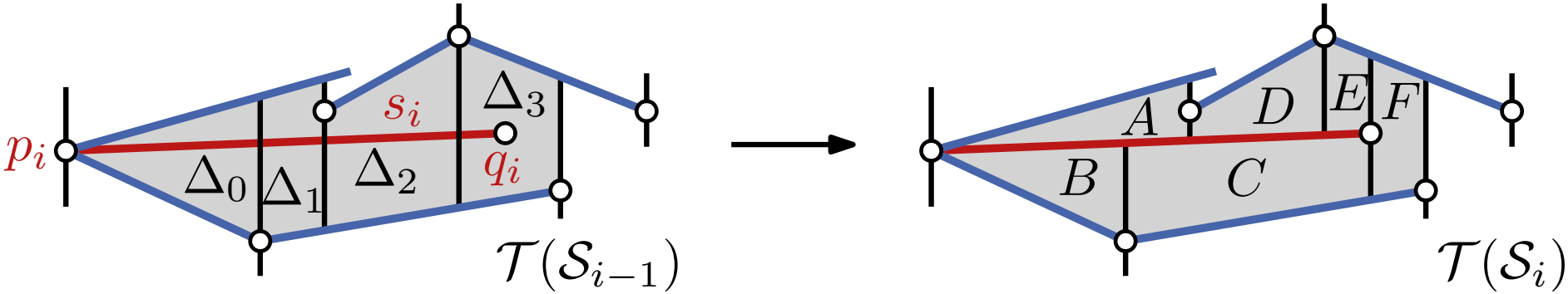
- Fall 1: $s_i \subset \Delta_0$
- Fall 2: $|\mathcal{T} \cap s_i| \geq 2$

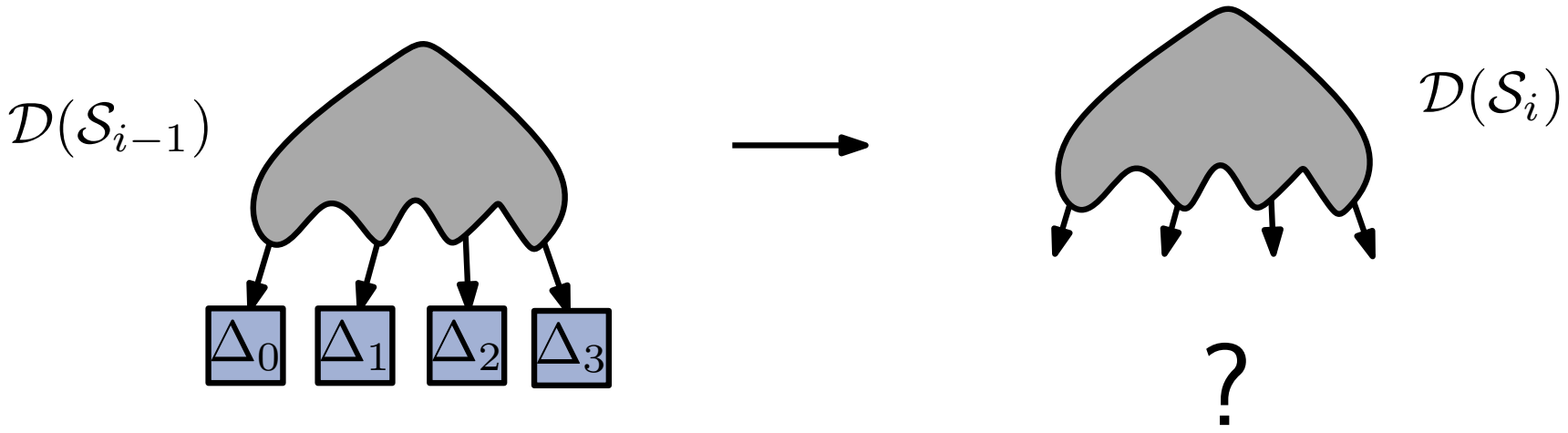


Update von $\mathcal{T}(\mathcal{S})$ und $\mathcal{D}(\mathcal{S})$

Schritt 2: Aktualisiere $\mathcal{T}$ und $\mathcal{D}$

- Fall 1: $s_i \subset \Delta_0$
- Fall 2: $|\mathcal{T} \cap s_i| \geq 2$

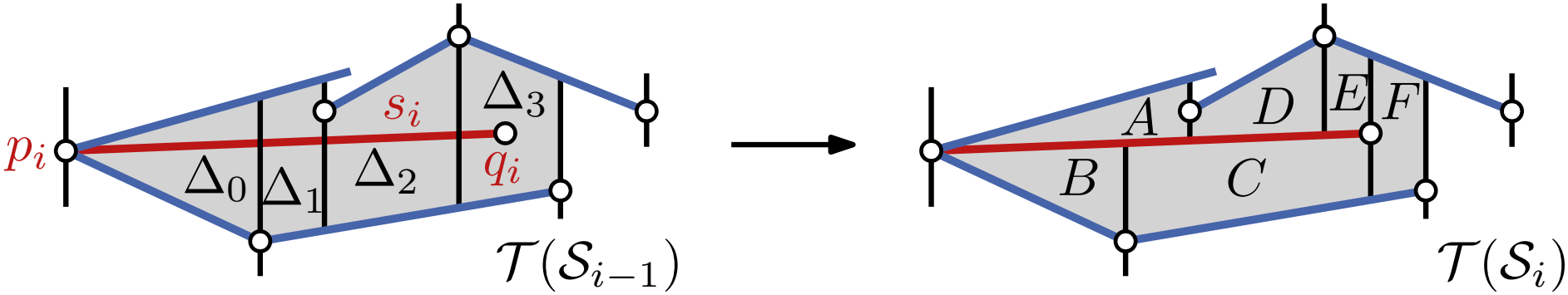


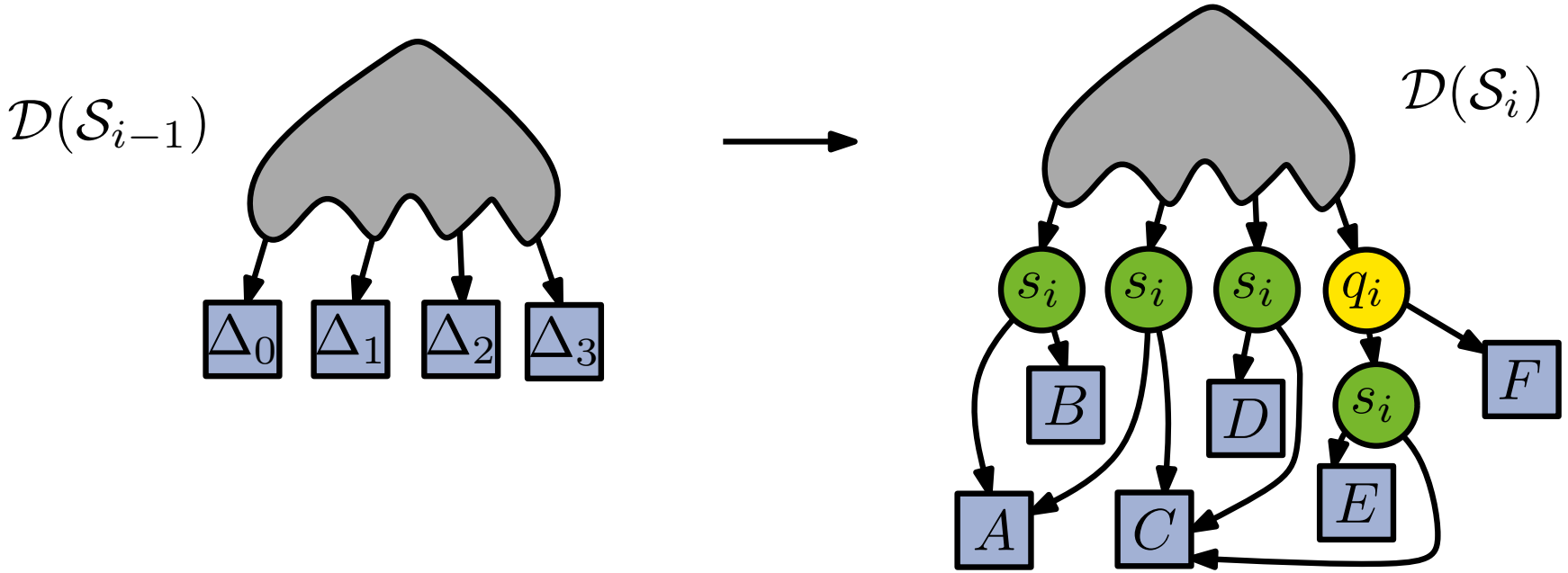


Update von $\mathcal{T}(\mathcal{S})$ und $\mathcal{D}(\mathcal{S})$

Schritt 2: Aktualisiere $\mathcal{T}$ und $\mathcal{D}$

- Fall 1: $s_i \subset \Delta_0$
- Fall 2: $|\mathcal{T} \cap s_i| \geq 2$





Analyse

Satz: Der Algorithmus berechnet die Trapezzerlegung $\mathcal{T}(\mathcal{S})$ und die Suchstruktur $\mathcal{D}$ für eine Menge $\mathcal{S}$ von n Strecken in *erwartet* $O(n \log n)$ Zeit. Die *erwartete* Größe von $\mathcal{D}$ ist $O(n)$ und die *erwartete* Anfragezeit $O(\log n)$.

Analyse

Satz: Der Algorithmus berechnet die Trapezzerlegung $\mathcal{T}(\mathcal{S})$ und die Suchstruktur $\mathcal{D}$ für eine Menge $\mathcal{S}$ von n Strecken in *erwartet* $O(n \log n)$ Zeit. Die *erwartete* Größe von $\mathcal{D}$ ist $O(n)$ und die *erwartete* Anfragezeit $O(\log n)$.

Beobachtungen:

- im schlimmsten Fall kann $\mathcal{D}$ quadratisch groß werden und eine Anfrage lineare Zeit benötigen
- Hoffnung: das passiert nur selten!
- betrachte erwartete Laufzeit und Größe über alle $n!$ möglichen Permutationen von $\mathcal{S}$
- der Satz gilt unabhängig von der Eingabemenge $\mathcal{S}$

Degenerierte Eingaben

Zwei Annahmen bislang:

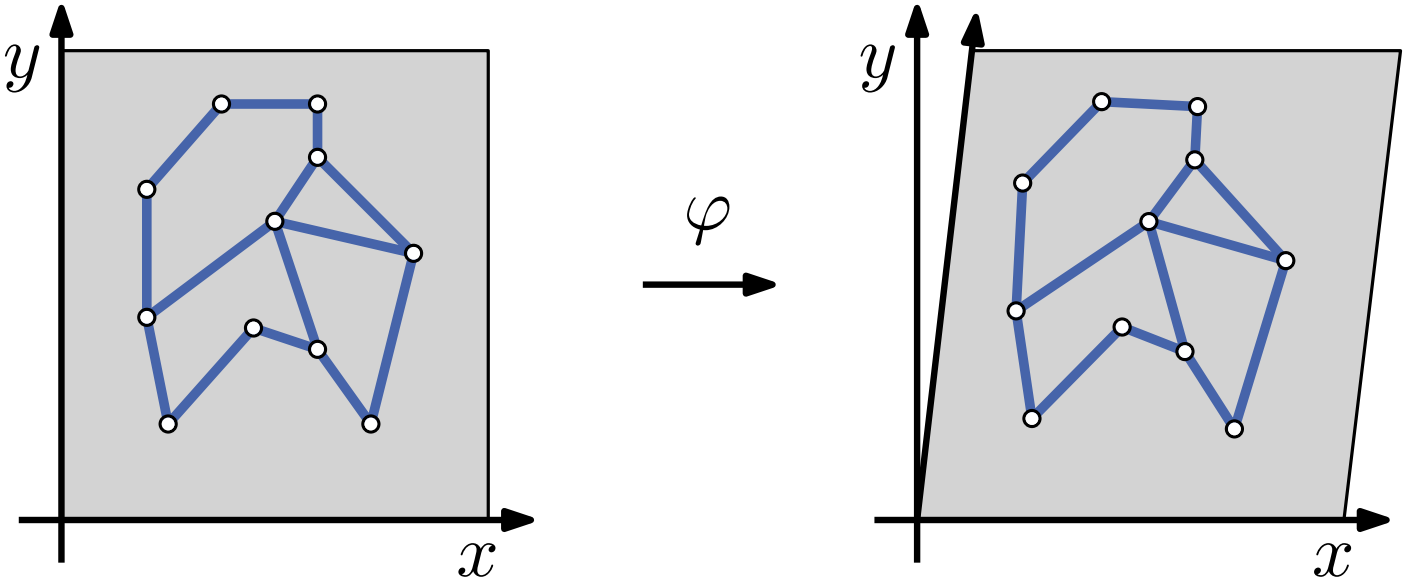
- keine zwei Streckenendpunkte habe gleiche x-Koordinate
- immer eindeutige Antworten im Suchpfad

Degenerierte Eingaben

Zwei Annahmen bislang:

- keine zwei Streckenendpunkte habe gleiche x-Koordinate
- immer eindeutige Antworten im Suchpfad

Ausweg: symbolische Scherung $\varphi : (x, y) \mapsto (x + \varepsilon y, y)$

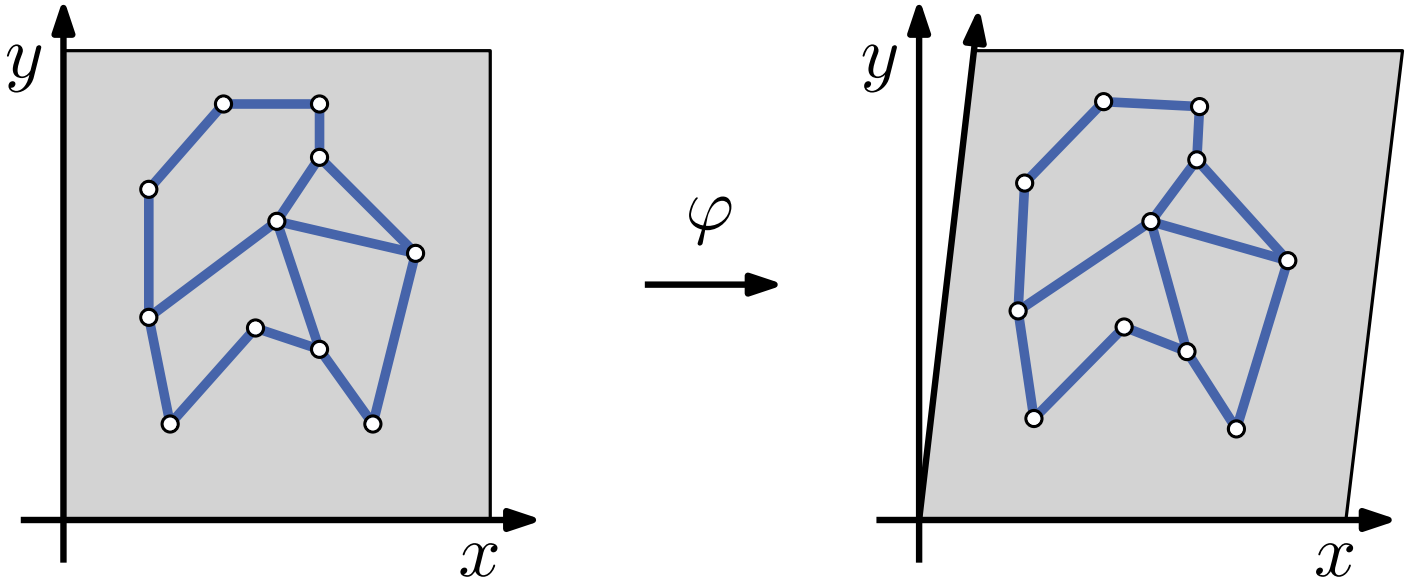


Degenerierte Eingaben

Zwei Annahmen bislang:

- keine zwei Streckenendpunkte habe gleiche x-Koordinate
- immer eindeutige Antworten im Suchpfad

Ausweg: symbolische Scherung $\varphi : (x, y) \mapsto (x + \varepsilon y, y)$



Dabei ist $\varepsilon > 0$ so gewählt, dass sich die x-Ordnung der Punkte nicht ändert.

Degenerierte Eingaben

- Führe Algorithmus für $\varphi\mathcal{S} = \{\varphi s \mid s \in \mathcal{S}\}$ und φp aus.

Degenerierte Eingaben

- Führe Algorithmus für $\varphi\mathcal{S} = \{\varphi s \mid s \in \mathcal{S}\}$ und φp aus.

- Zwei elementare Operationen beim Aufbau von $\mathcal{T}$ und $\mathcal{D}$:
  1. liegt q links oder rechts der Vertikalen durch p?
  2. liegt q oberhalb oder unterhalb der Strecke s?

Degenerierte Eingaben

- Führe Algorithmus für $\varphi\mathcal{S} = \{\varphi s \mid s \in \mathcal{S}\}$ und φp aus.

- Zwei elementare Operationen beim Aufbau von $\mathcal{T}$ und $\mathcal{D}$:
  1. liegt q links oder rechts der Vertikalen durch p?
  2. liegt q oberhalb oder unterhalb der Strecke s?

- Auch Lokalisierung eines Punktes q in $\mathcal{T}(\mathcal{S})$ geht durch Lokalisierung von φq in $\mathcal{T}(\varphi\mathcal{S})$.

Worst-Case Abschätzung

Bislang: erwartete Anfragezeit für bel. Punkt ist $O(\log n)$

Aber: jede Permutation könnte sehr schlechte Anfragepunkte haben

Worst-Case Abschätzung

Bislang: erwartete Anfragezeit für bel. Punkt ist $O(\log n)$

Aber: jede Permutation könnte sehr schlechte Anfragepunkte haben

Lemma: Sei $\mathcal{S}$ Menge von n kreuzungsfreien Strecken, q Anfragepunkt und $\lambda > 0$. Dann gilt $\Pr[\text{Suchpfad für } q \text{ länger als } 3\lambda \ln(n+1)] \leq 1/(n+1)^{\lambda \ln 1.25 - 1}$.

Worst-Case Abschätzung

Bislang: erwartete Anfragezeit für bel. Punkt ist $O(\log n)$

Aber: jede Permutation könnte sehr schlechte Anfragepunkte haben

Lemma: Sei $\mathcal{S}$ Menge von n kreuzungsfreien Strecken, q Anfragepunkt und $\lambda > 0$. Dann gilt
Pr[Suchpfad für q länger als $3\lambda \ln(n+1)] \leq 1/(n+1)^{\lambda \ln 1.25 - 1}$.

Lemma: Sei $\mathcal{S}$ Menge von n kreuzungsfreien Strecken und $\lambda > 0$. Dann gilt
Pr[max. Suchpfad in $\mathcal{D}$ länger als $3\lambda \ln(n+1)] \leq 2/(n+1)^{\lambda \ln 1.25 - 3}$.

Worst-Case Abschätzung

Bislang: erwartete Anfragezeit für bel. Punkt ist $O(\log n)$

Aber: jede Permutation könnte sehr schlechte Anfragepunkte haben

Lemma: Sei $\mathcal{S}$ Menge von n kreuzungsfreien Strecken, q Anfragepunkt und $\lambda > 0$. Dann gilt
Pr[Suchpfad für q länger als $3\lambda \ln(n+1)] \leq 1/(n+1)^{\lambda \ln 1.25 - 1}$.

Lemma: Sei $\mathcal{S}$ Menge von n kreuzungsfreien Strecken und $\lambda > 0$. Dann gilt
Pr[max. Suchpfad in $\mathcal{D}$ länger als $3\lambda \ln(n+1)] \leq 2/(n+1)^{\lambda \ln 1.25 - 3}$.

Satz: Sei $\mathcal{S}$ Unterteilung der Ebene mit n Kanten. Es gibt eine Suchstruktur zur Punktlokalisierung in $\mathcal{S}$ mit Platzbedarf $O(n)$ und Anfragezeit $O(\log n)$.

Diskussion

Karlsruhe Institute of Technology

Gibt es ähnliche Methoden auch für höhere Dimensionen?

Diskussion

Gibt es ähnliche Methoden auch für höhere Dimensionen?

Die derzeit beste Datenstruktur in drei Dimensionen benötigt $O(n \log n)$ Platz und $O(\log^2 n)$ Anfragezeit [Snoeyink '04]. Ob es mit linearem Platz und $O(\log n)$ Anfragezeit geht ist eine offene Frage. In höheren Dimensionen sind effiziente Verfahren nur für spezielle Unterteilungen durch Hyperebenen bekannt.