Karlsruhe Institute of Technology

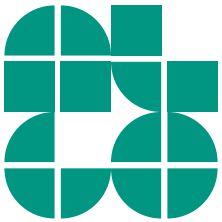
Vorlesung Algorithmische Geometrie

Voronoi-Diagramme

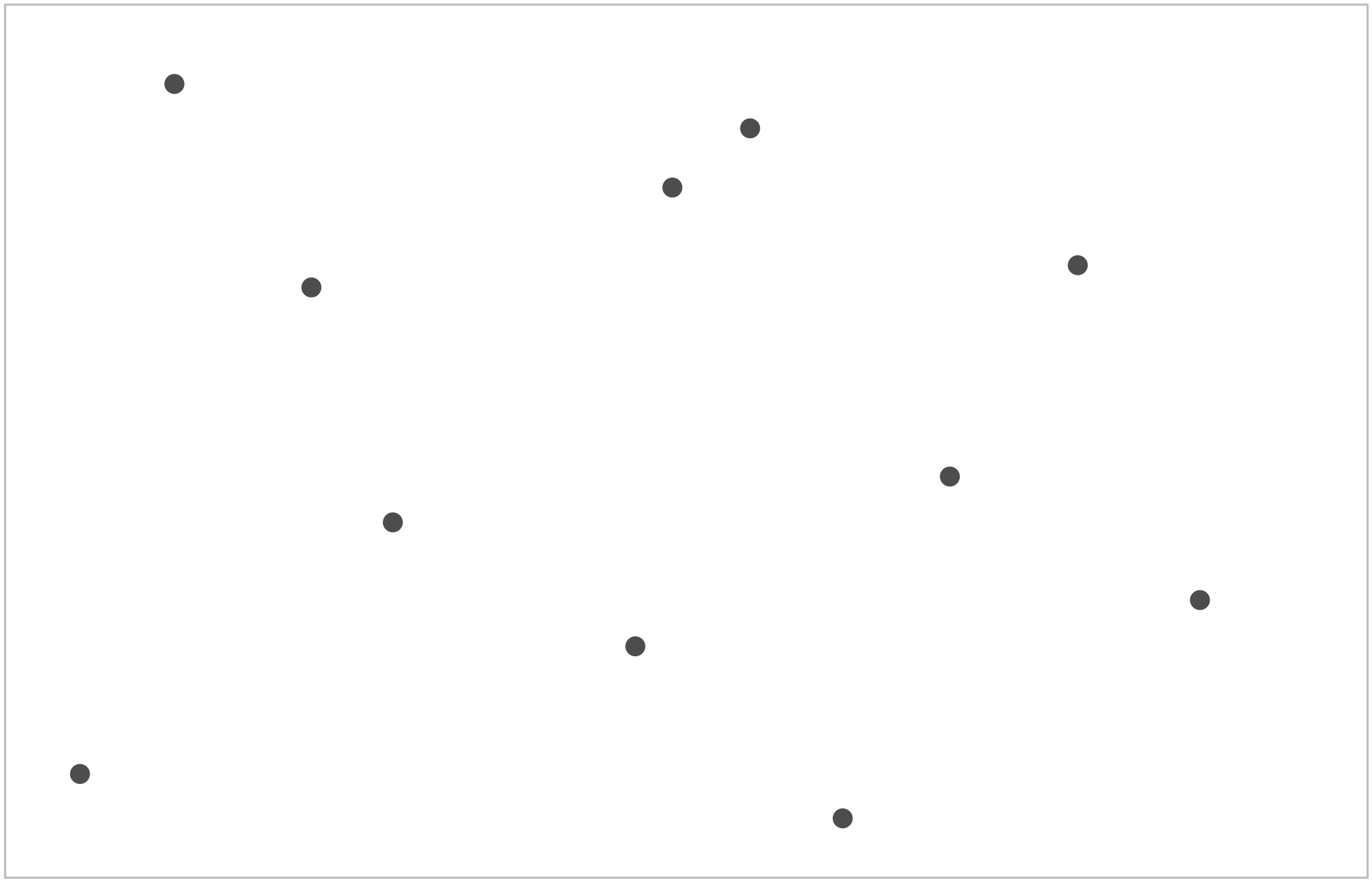
INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Martin Nöllenburg

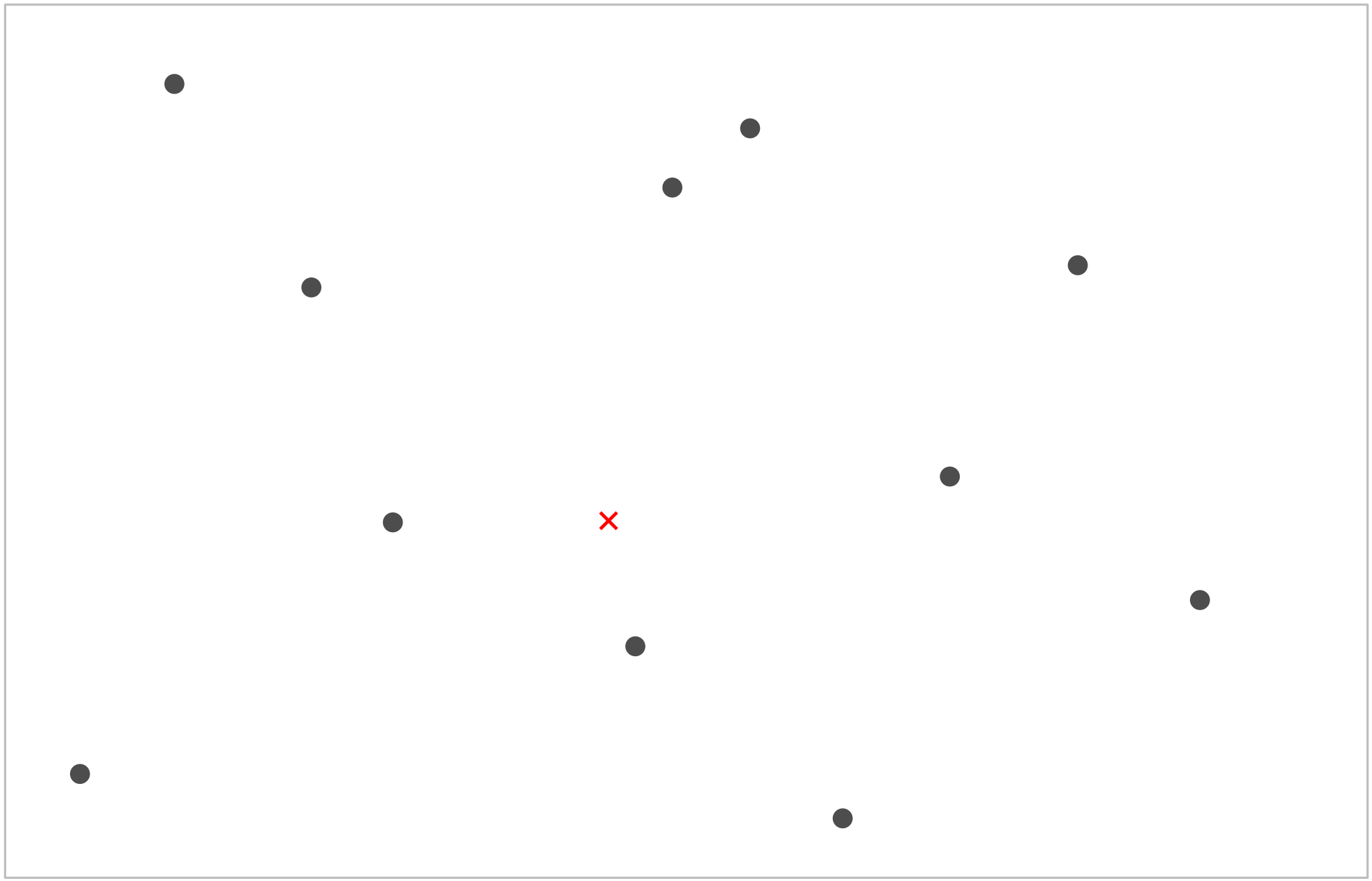
29.05.2011



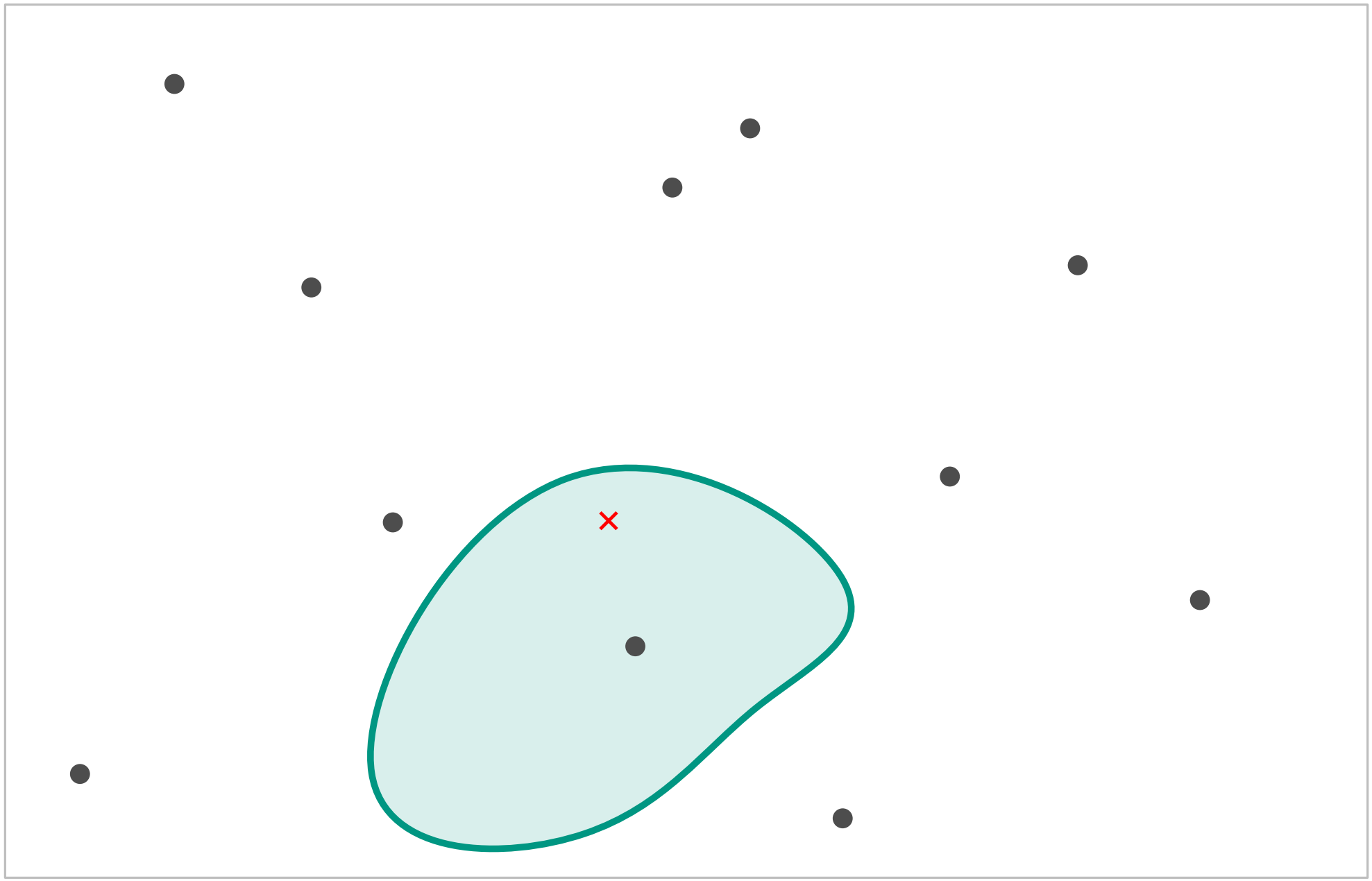
Das Postamt-Problem



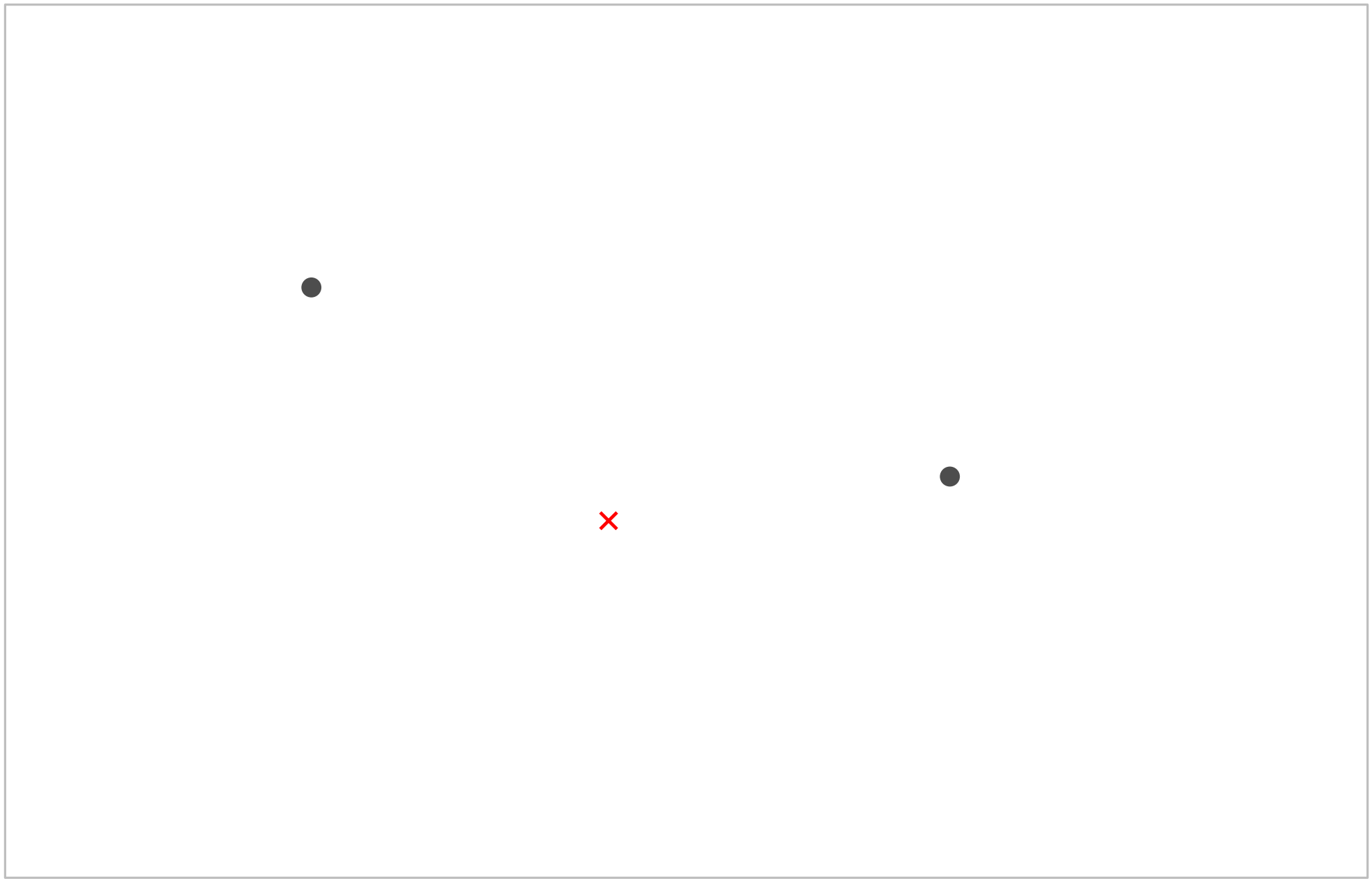
Das Postamt-Problem



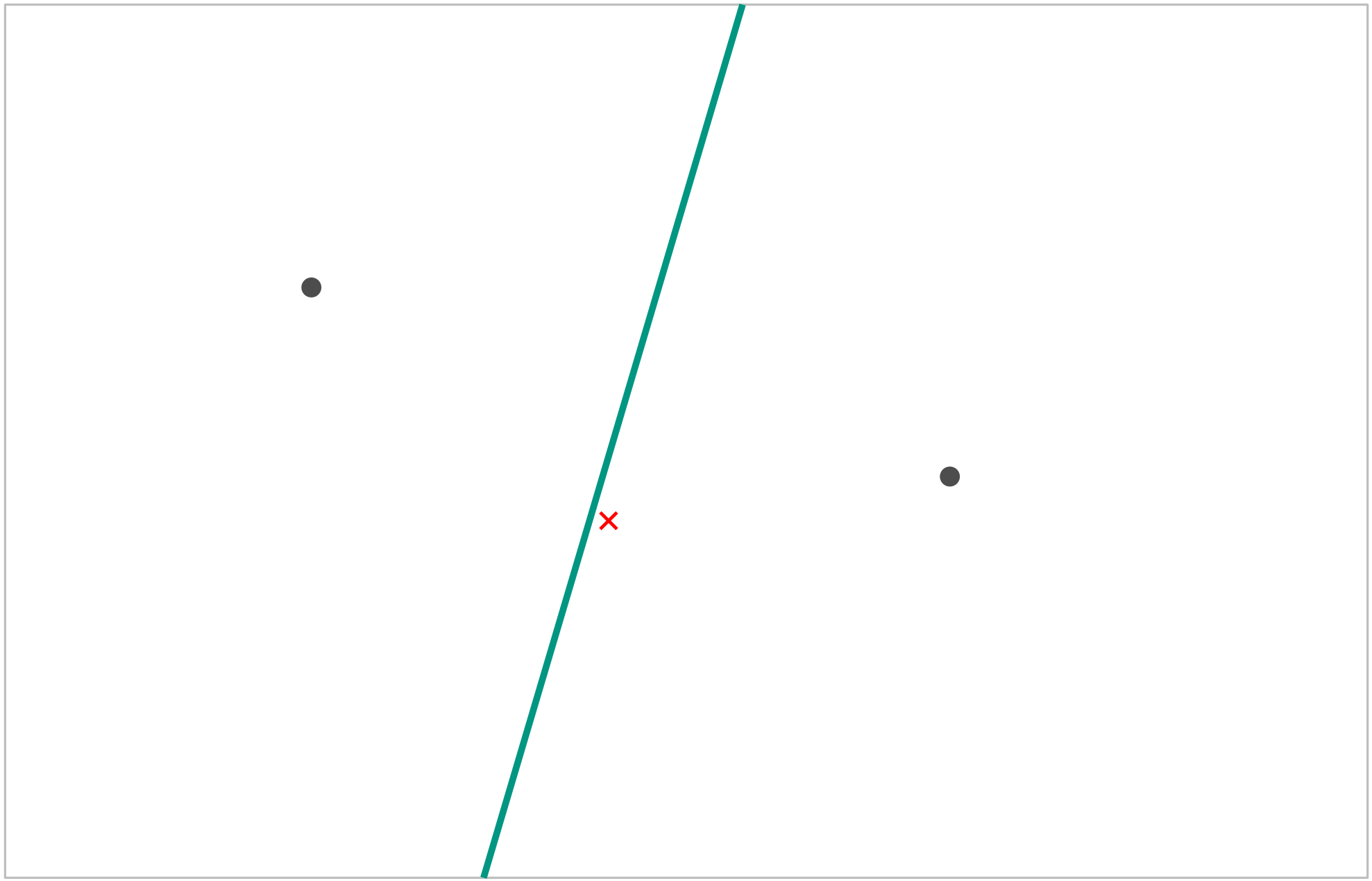
Das Postamt-Problem



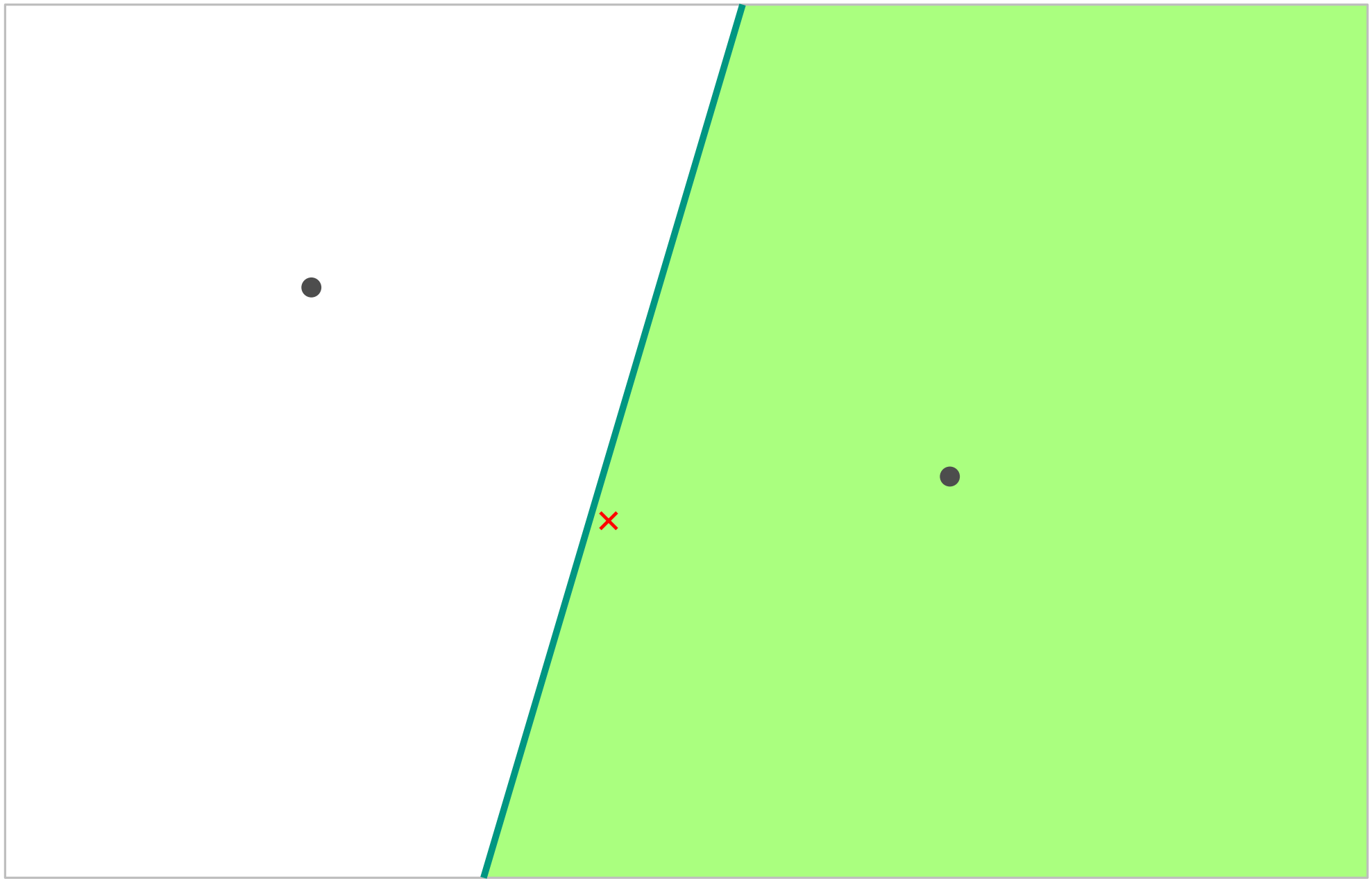
Das Postamt-Problem



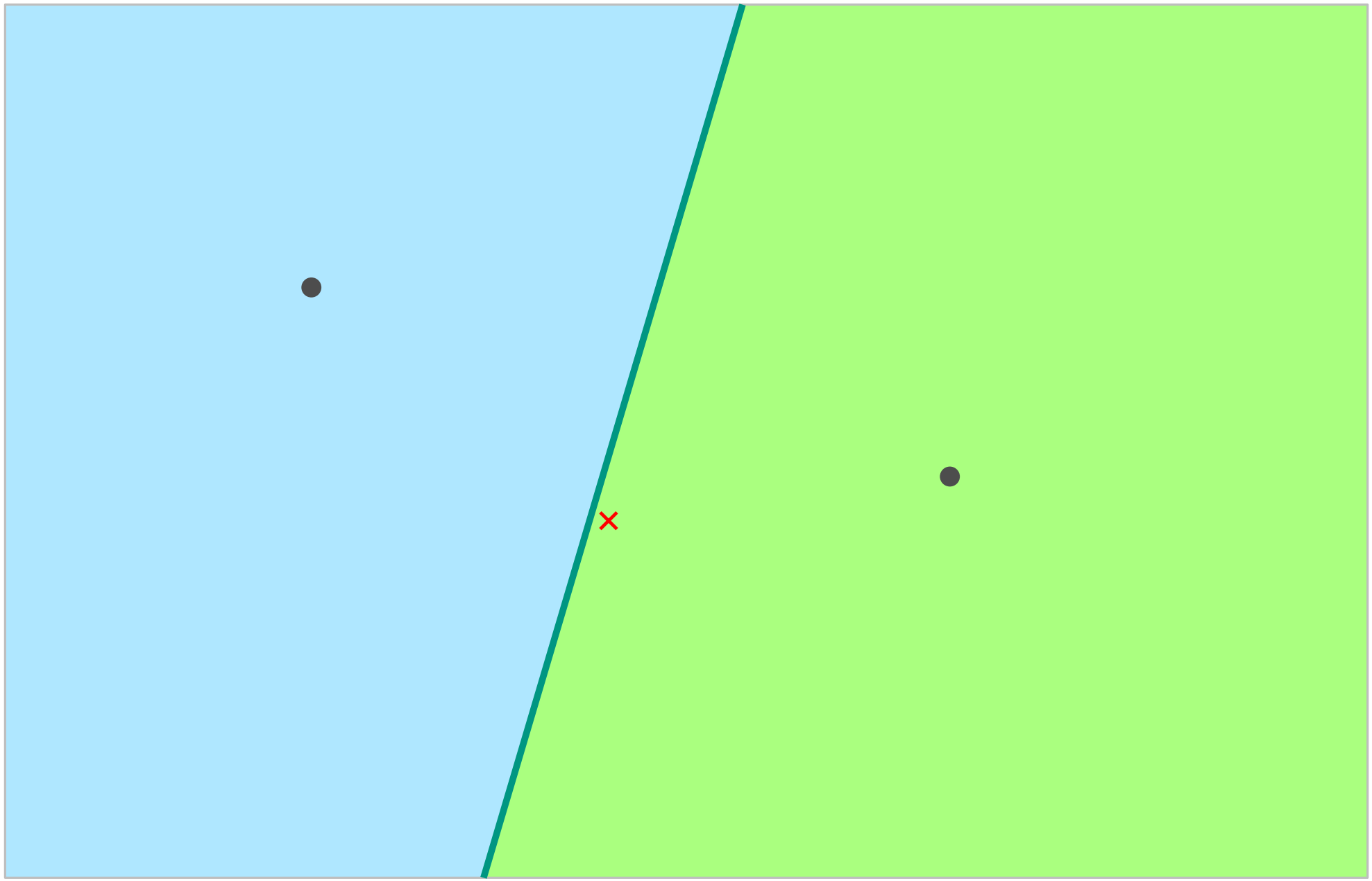
Das Postamt-Problem



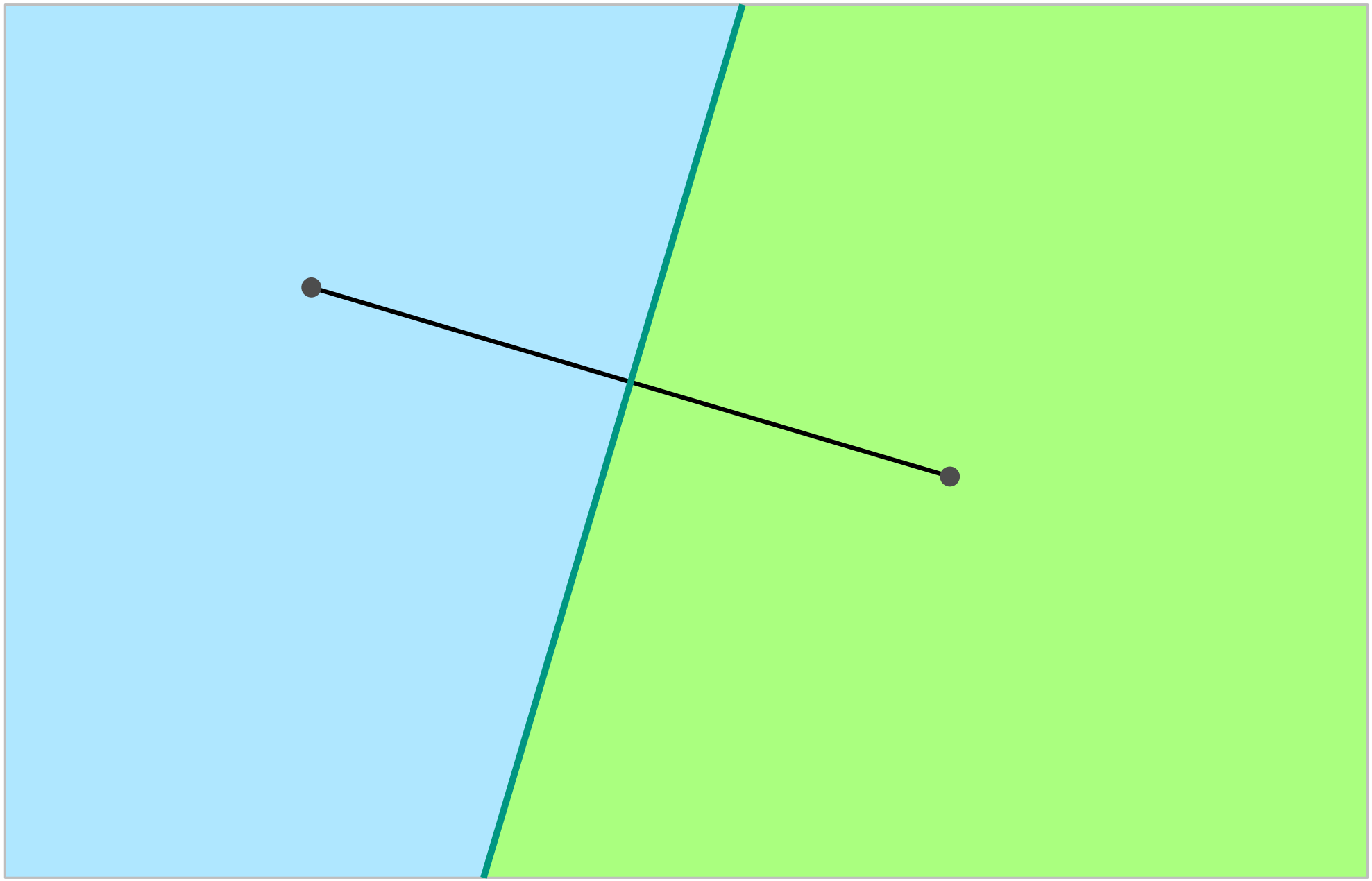
Das Postamt-Problem



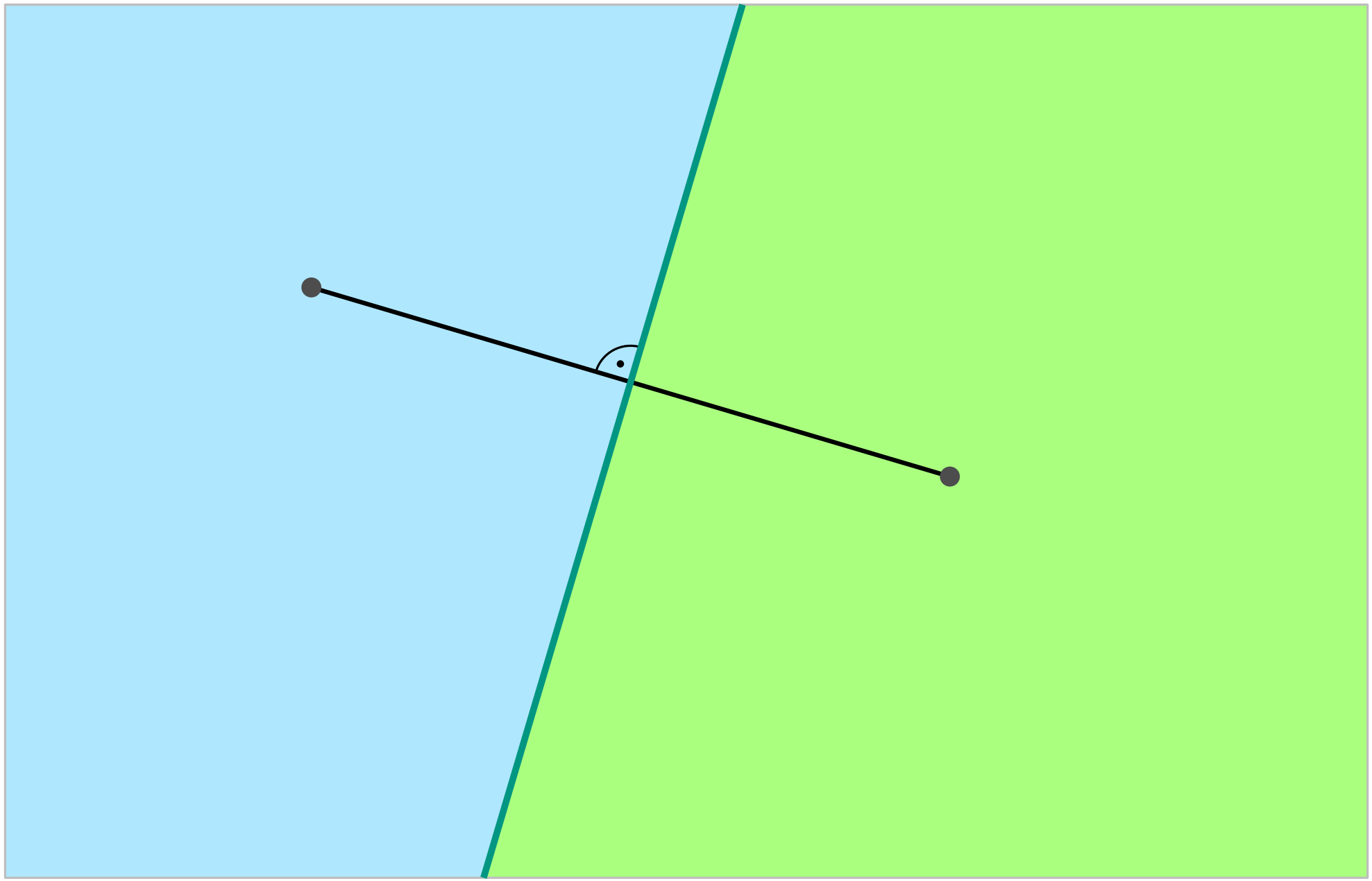
Das Postamt-Problem



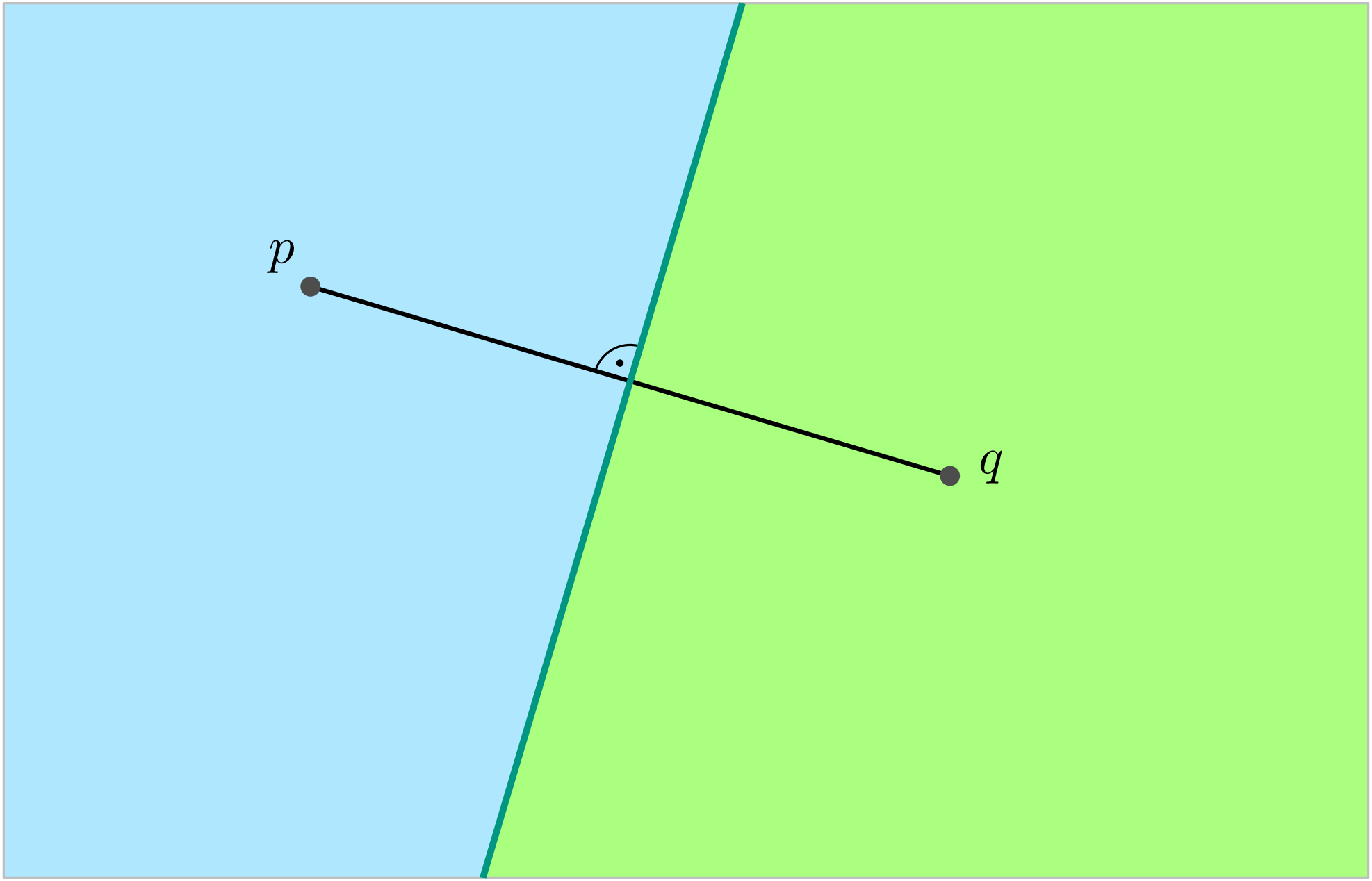
Das Postamt-Problem



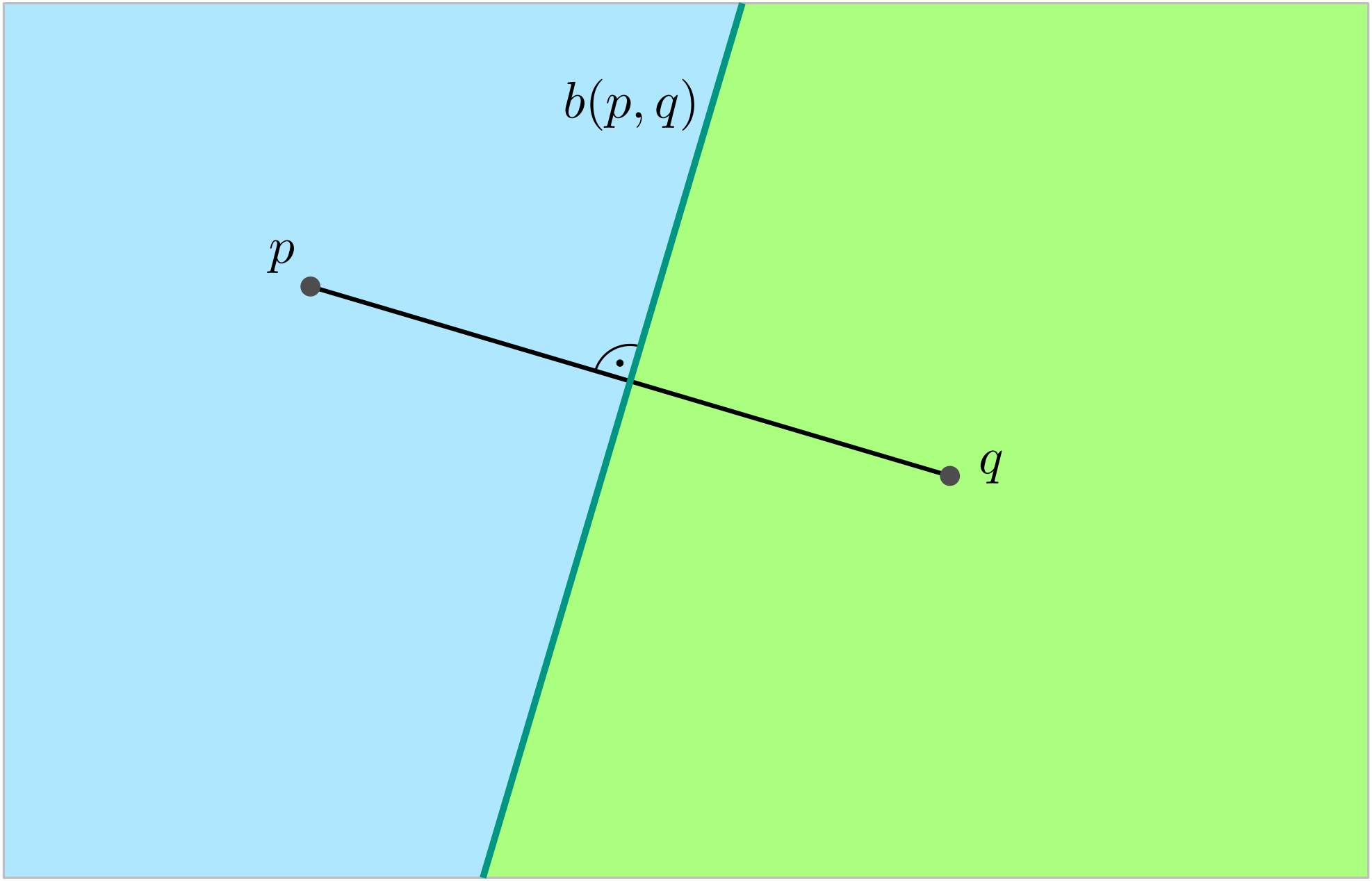
Das Postamt-Problem



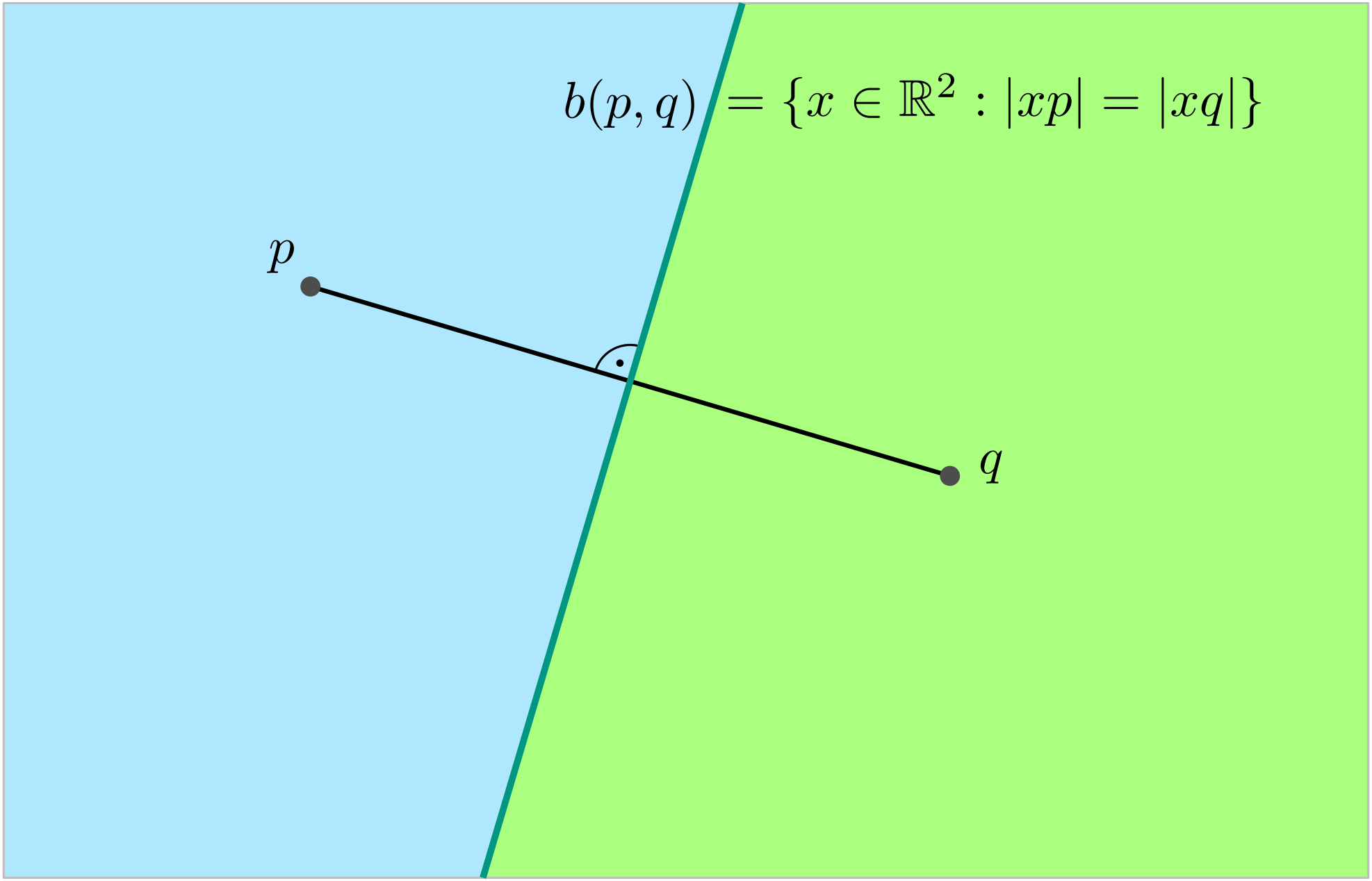
Das Postamt-Problem



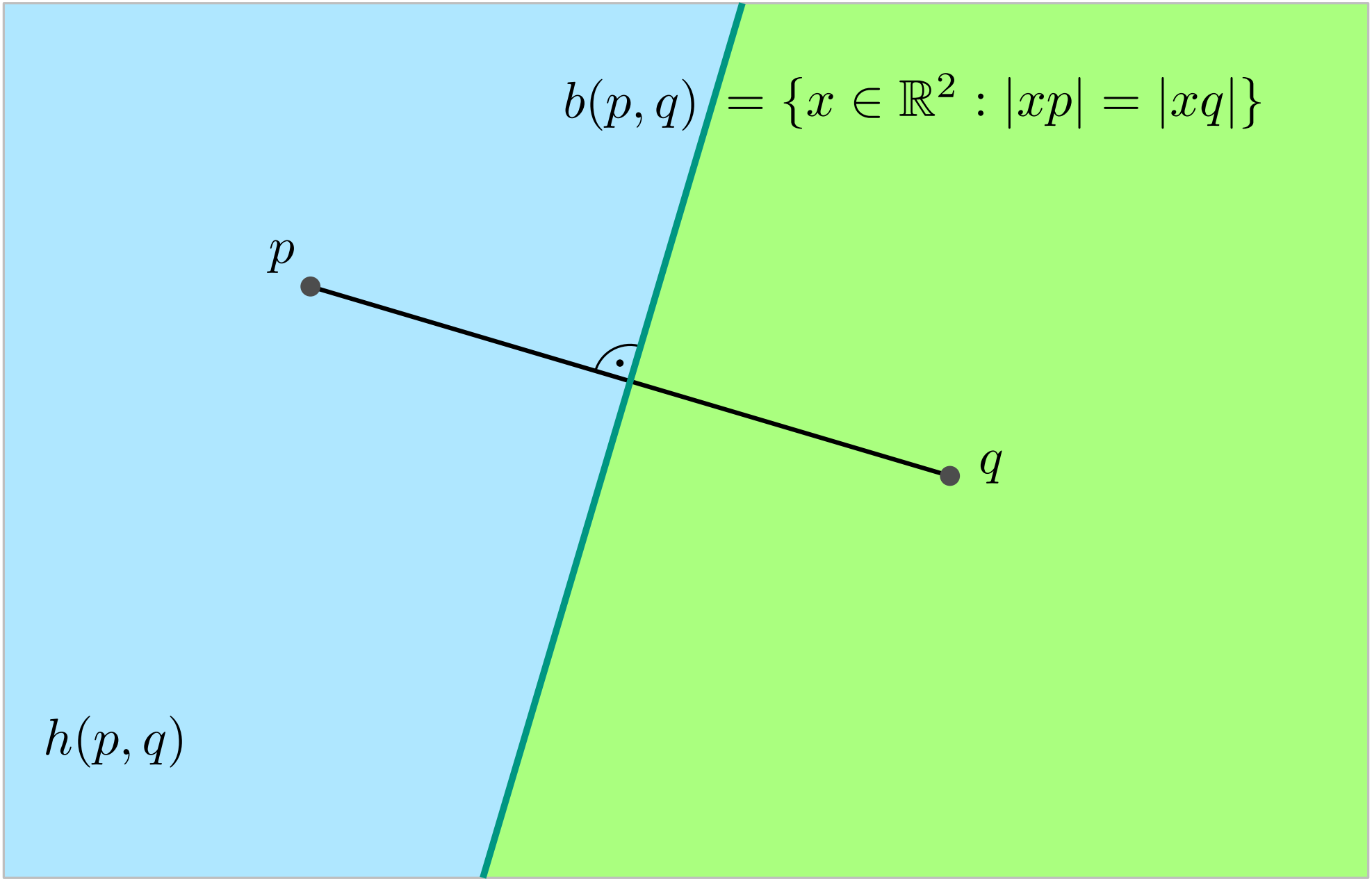
Das Postamt-Problem



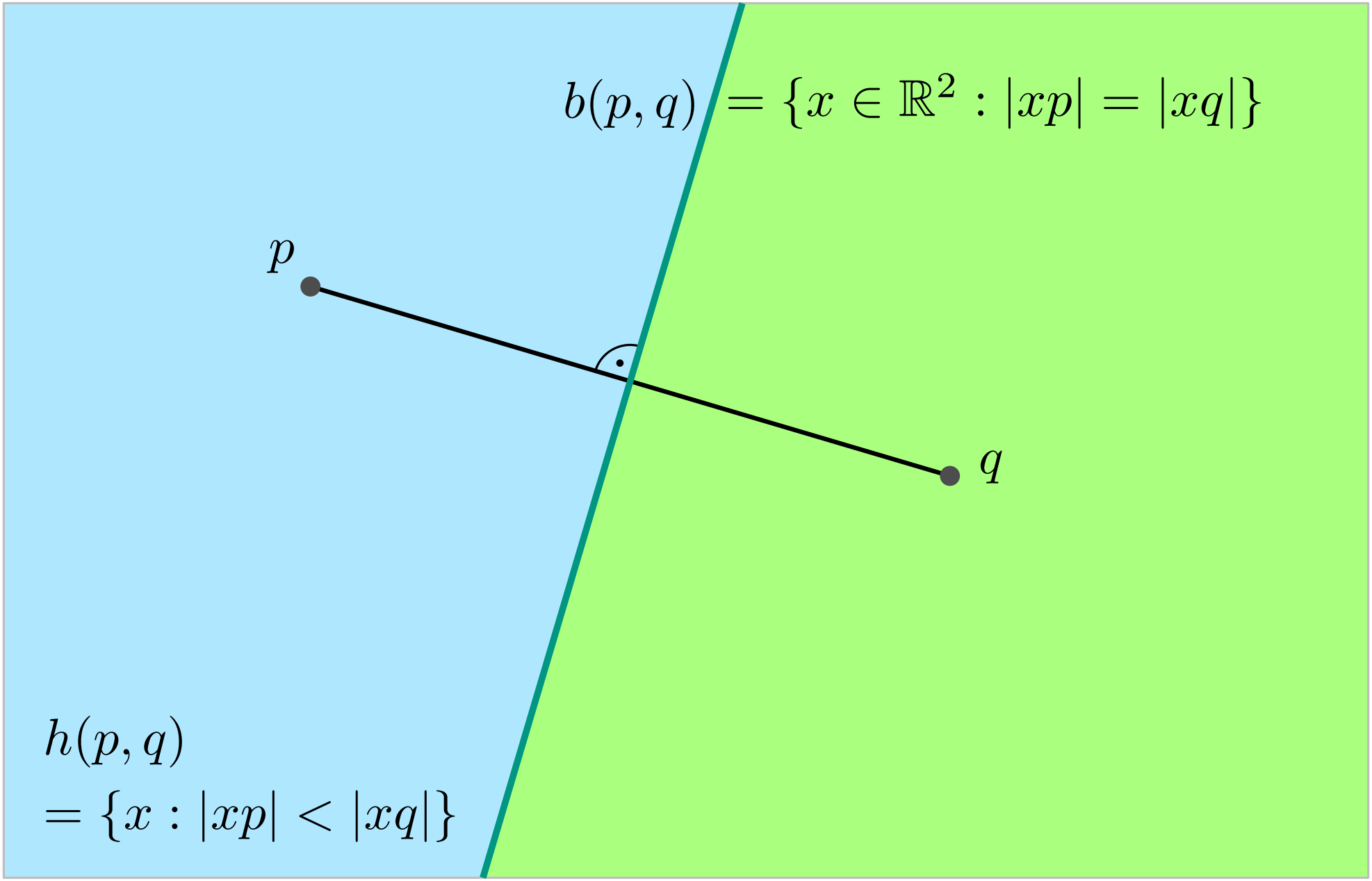
Das Postamt-Problem



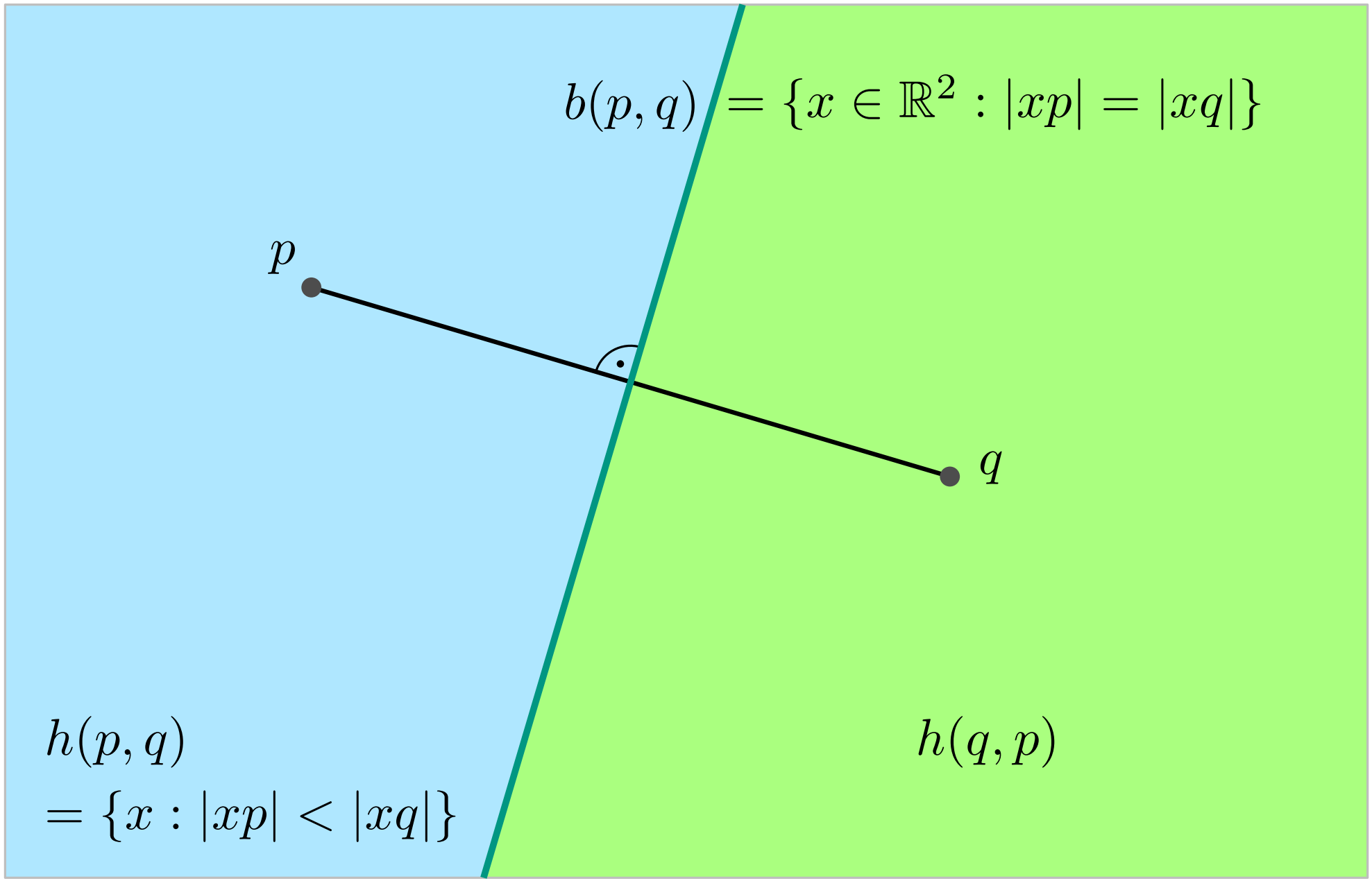
Das Postamt-Problem



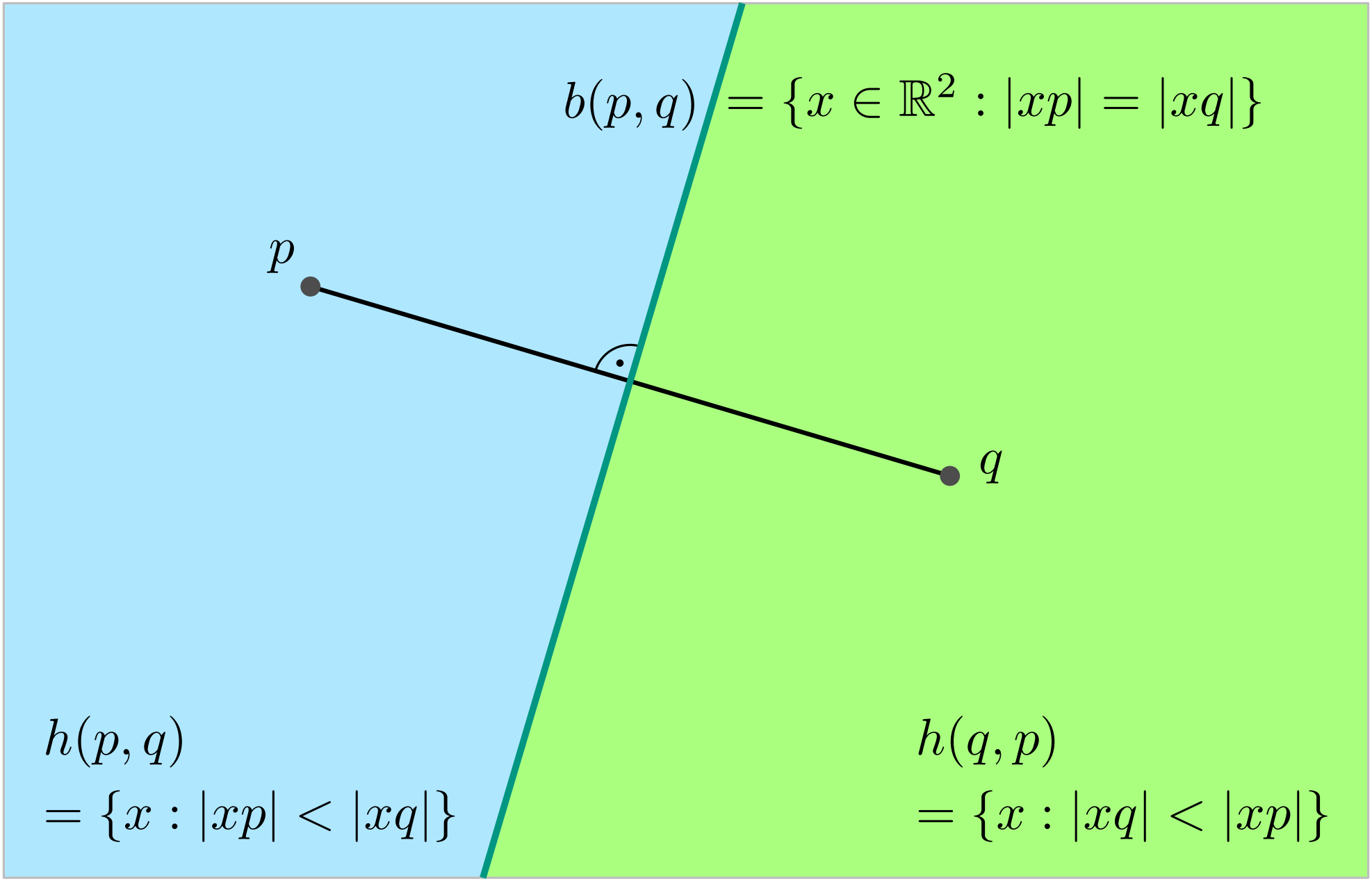
Das Postamt-Problem



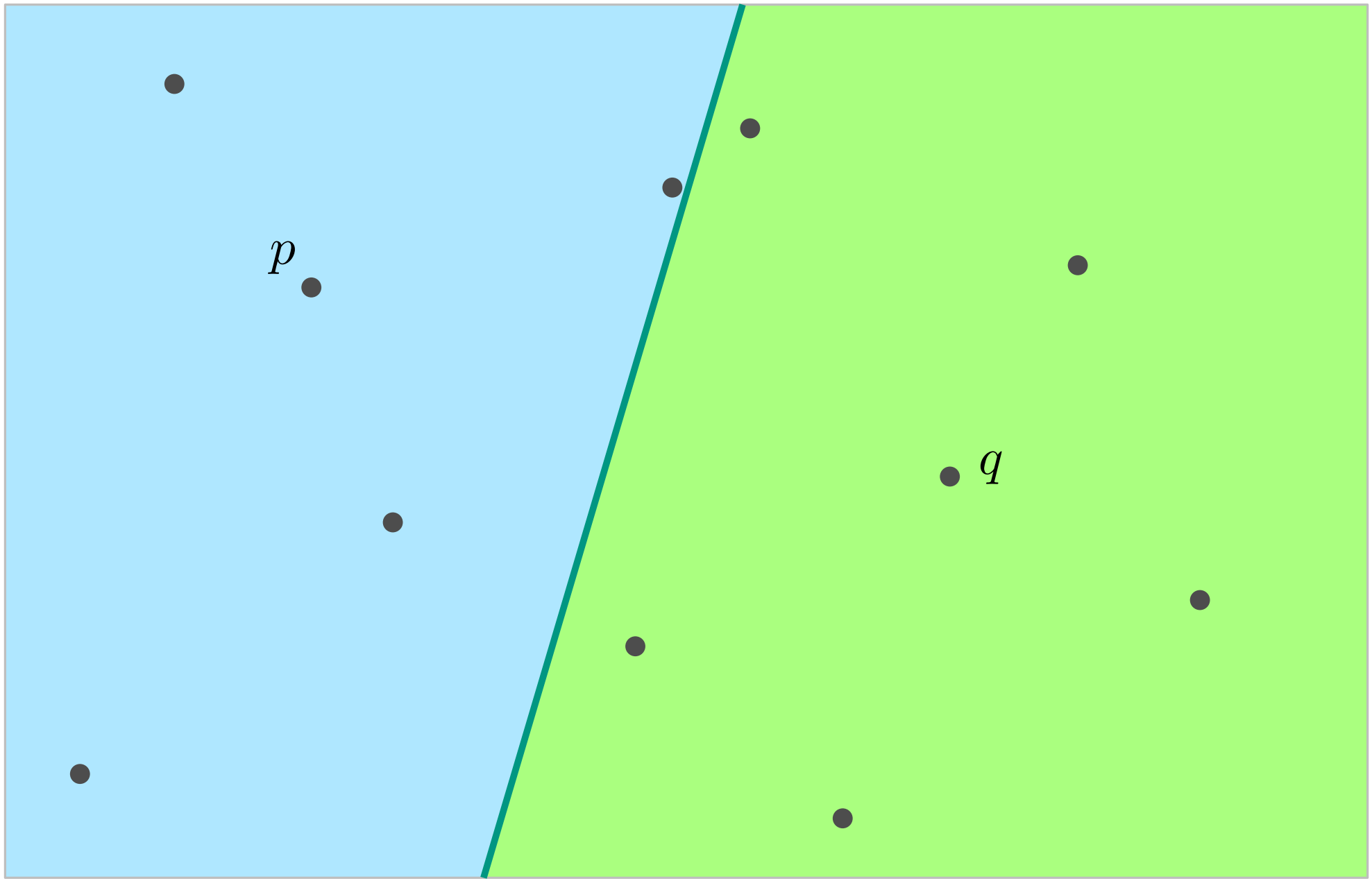
Das Postamt-Problem



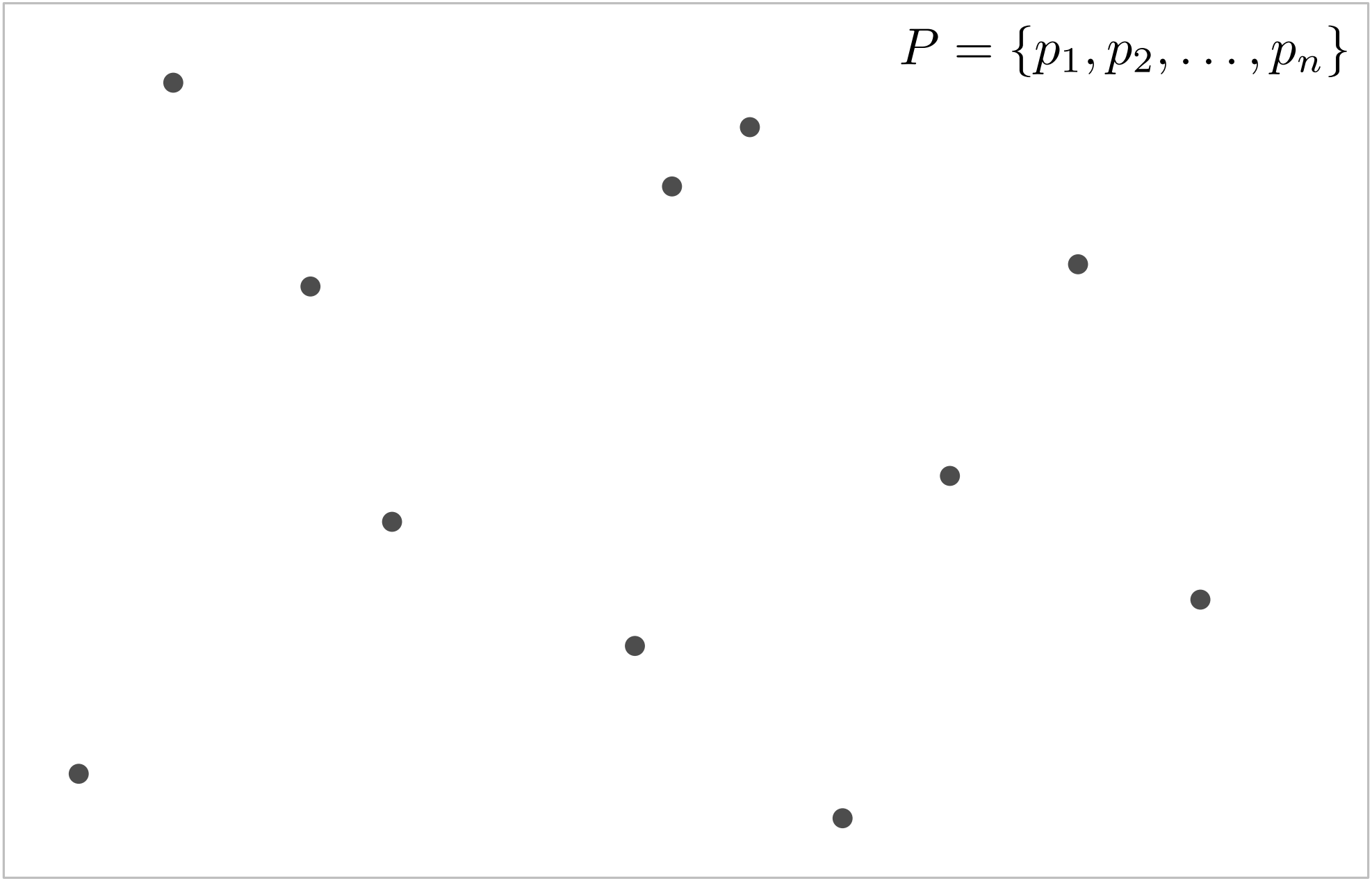
Das Postamt-Problem



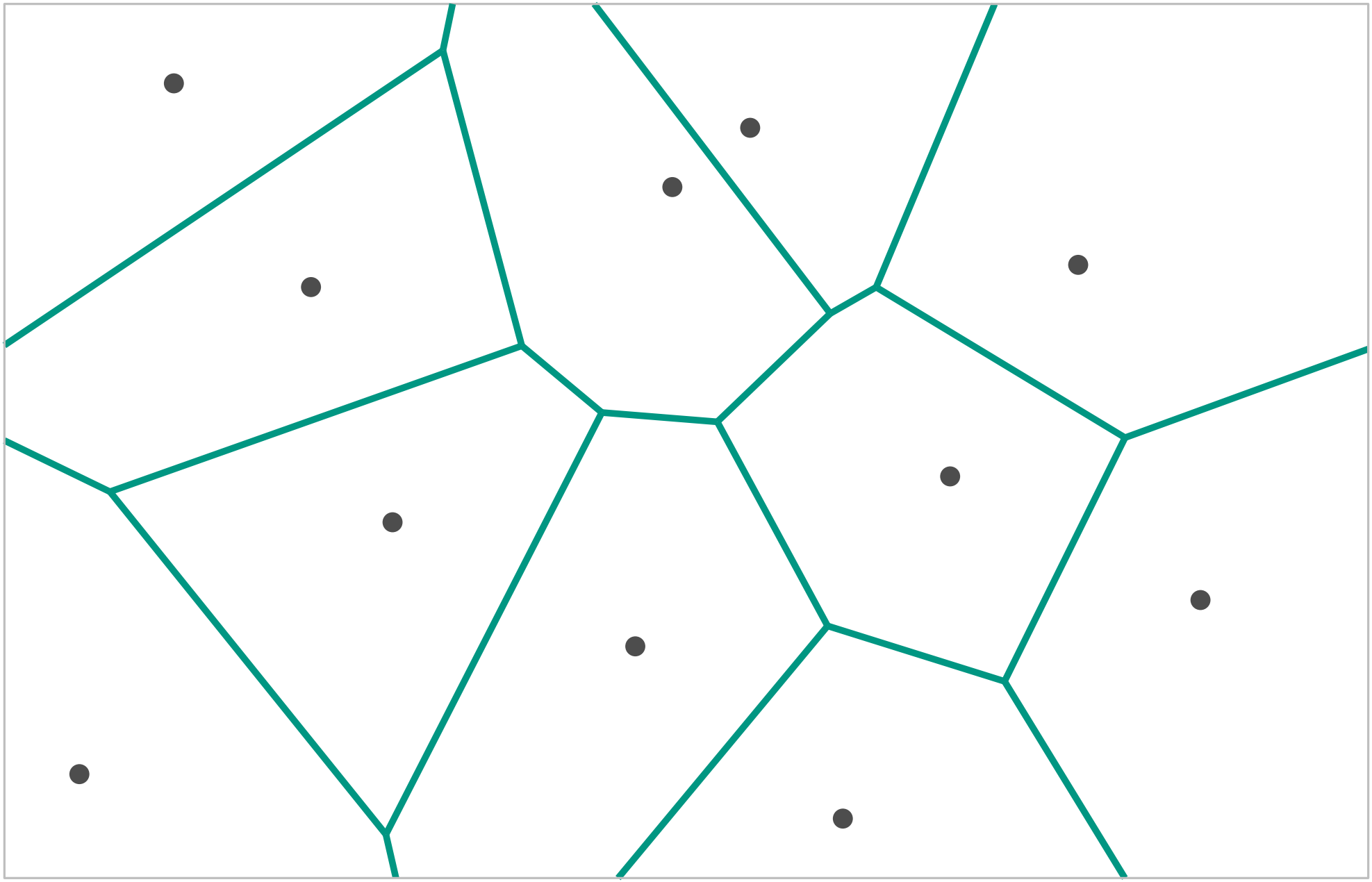
Das Postamt-Problem



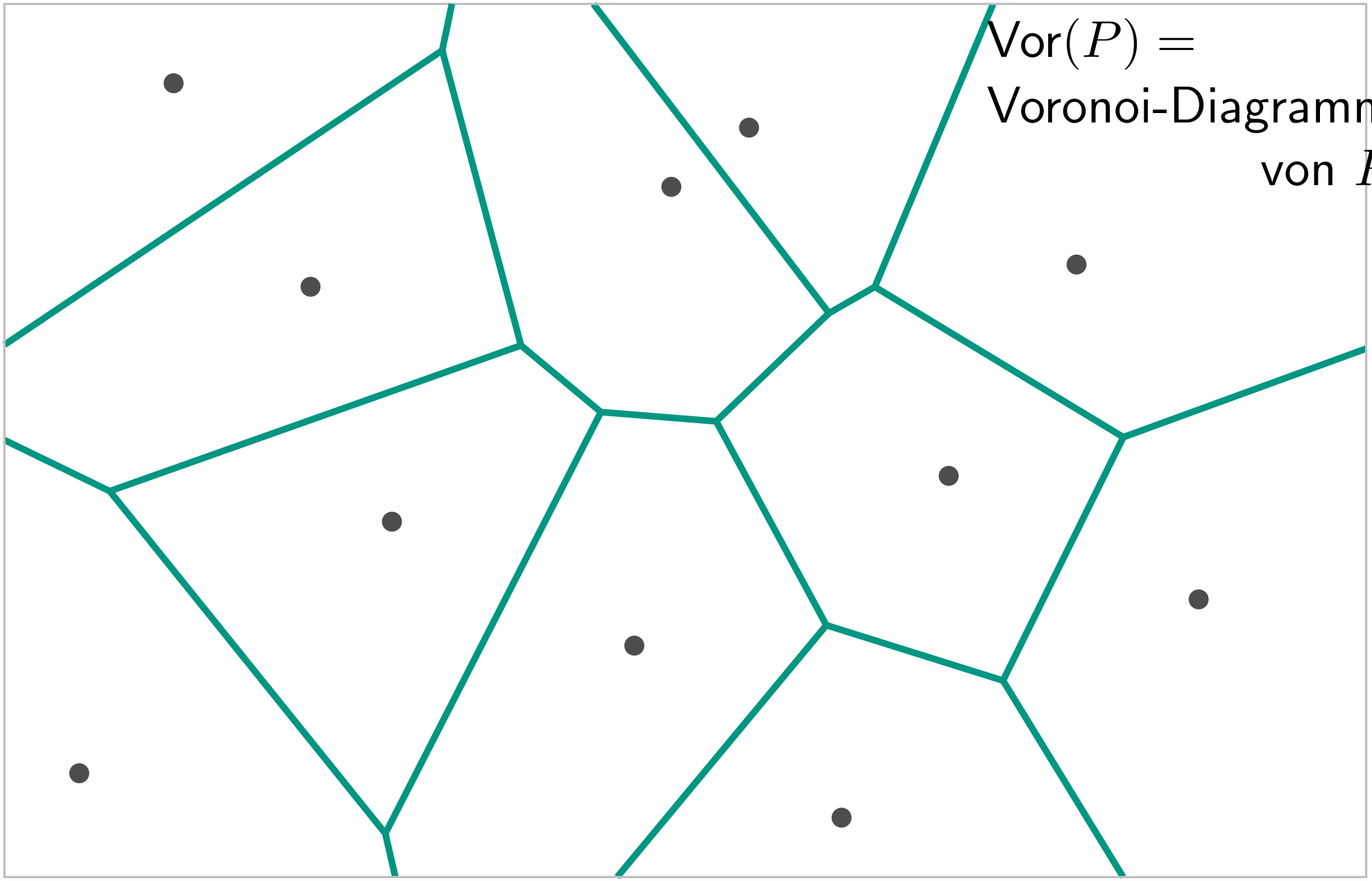
Das Postamt-Problem



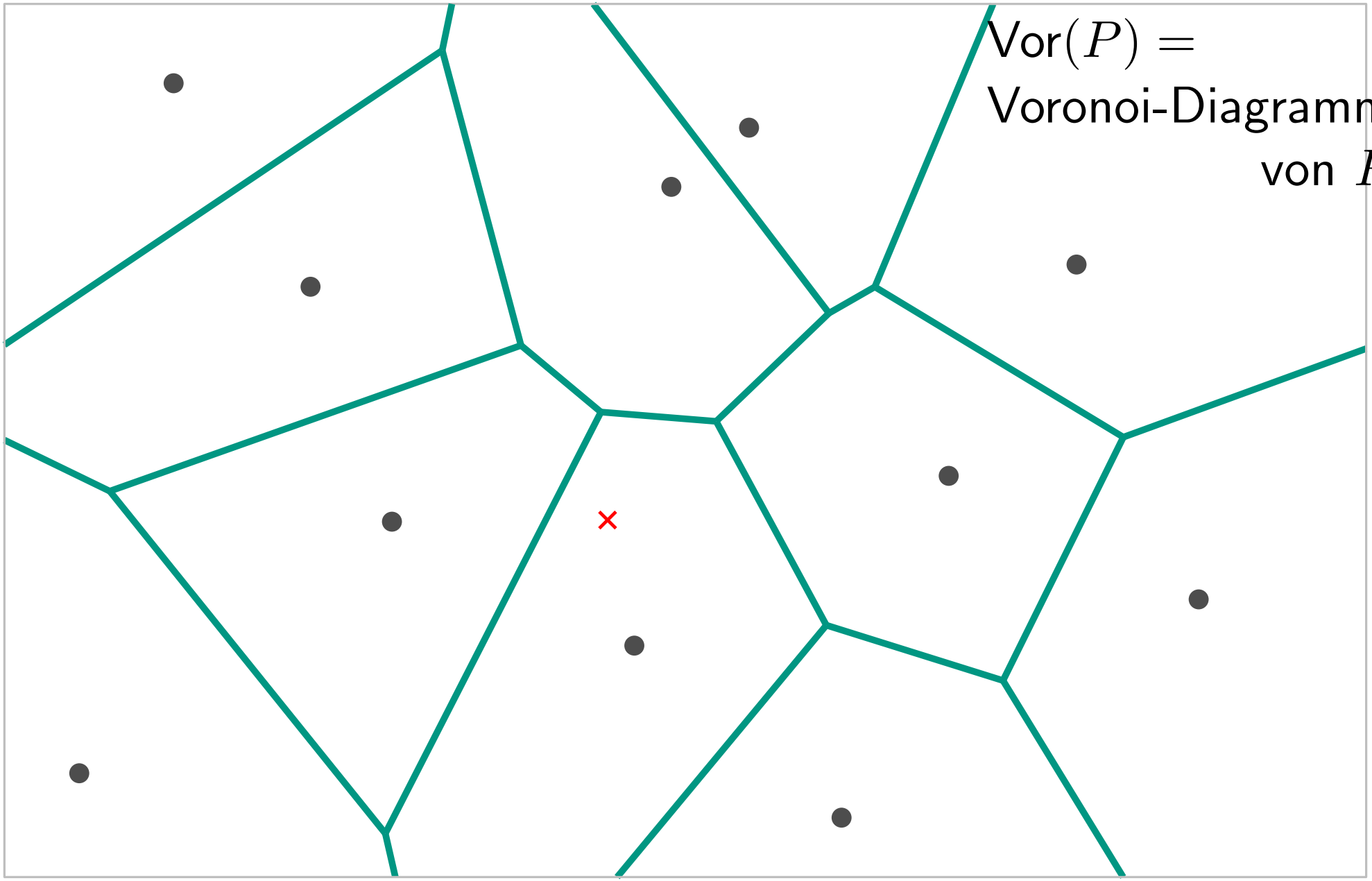
Das Postamt-Problem



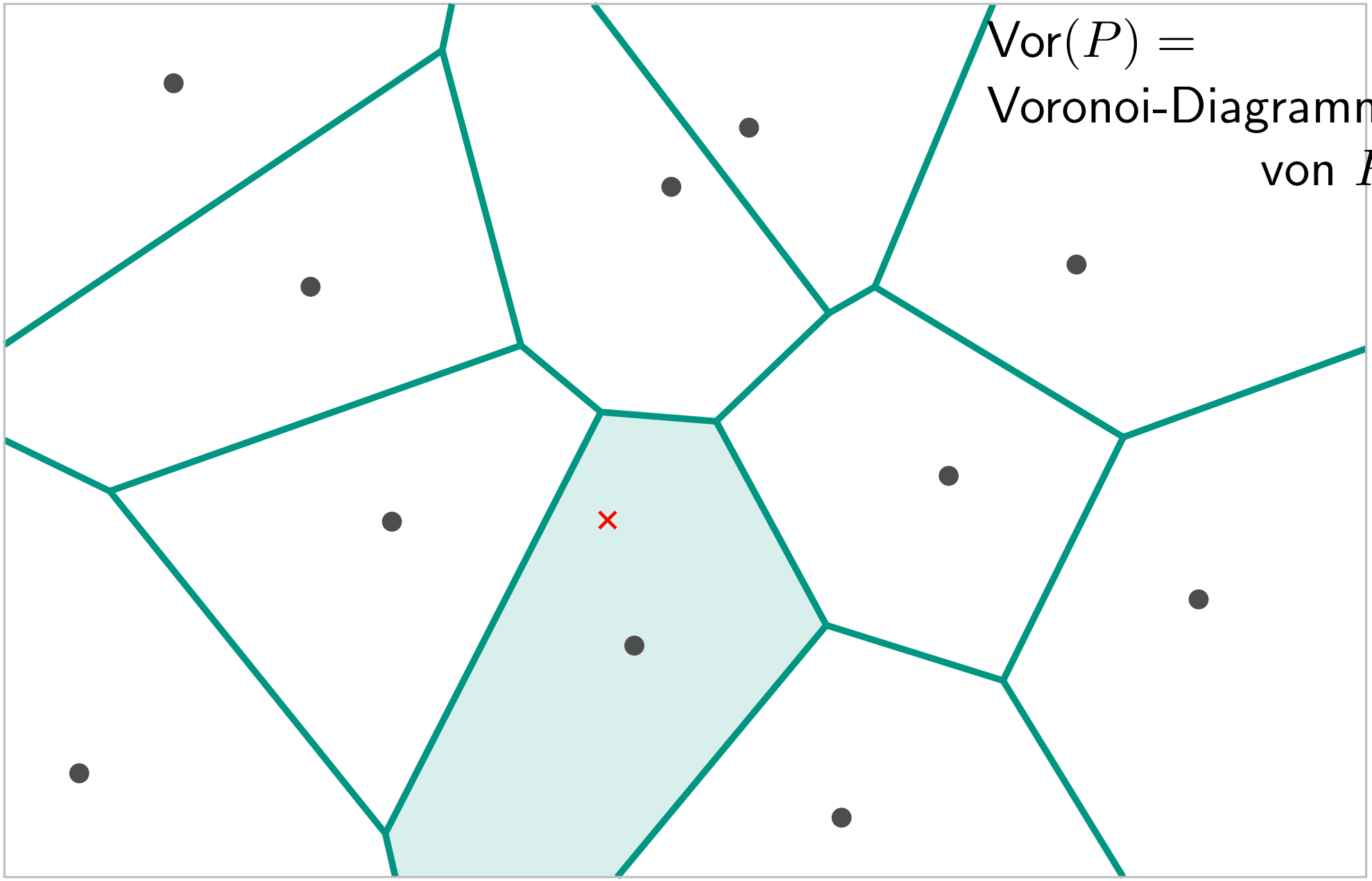
Das Postamt-Problem



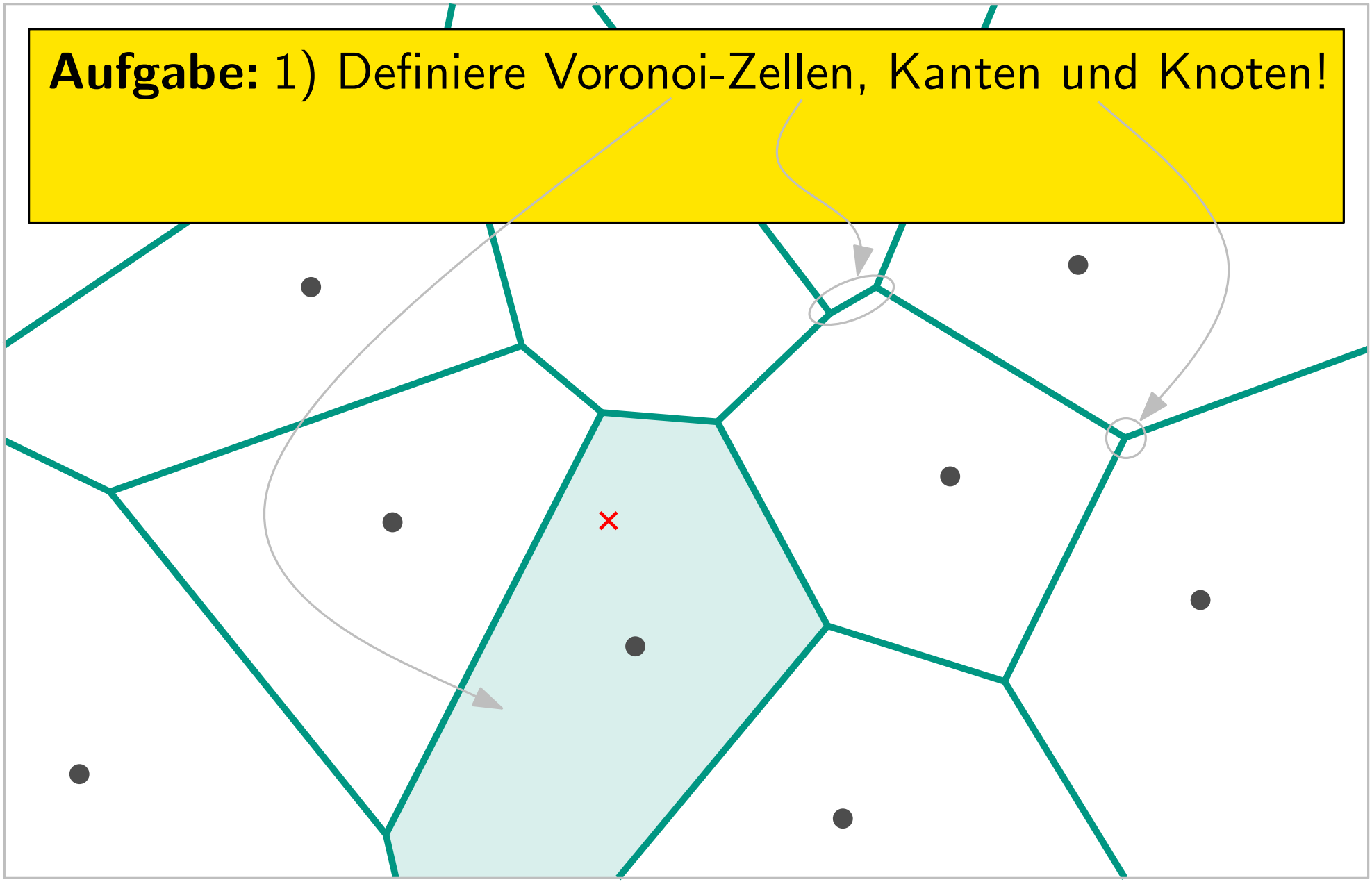
Das Postamt-Problem



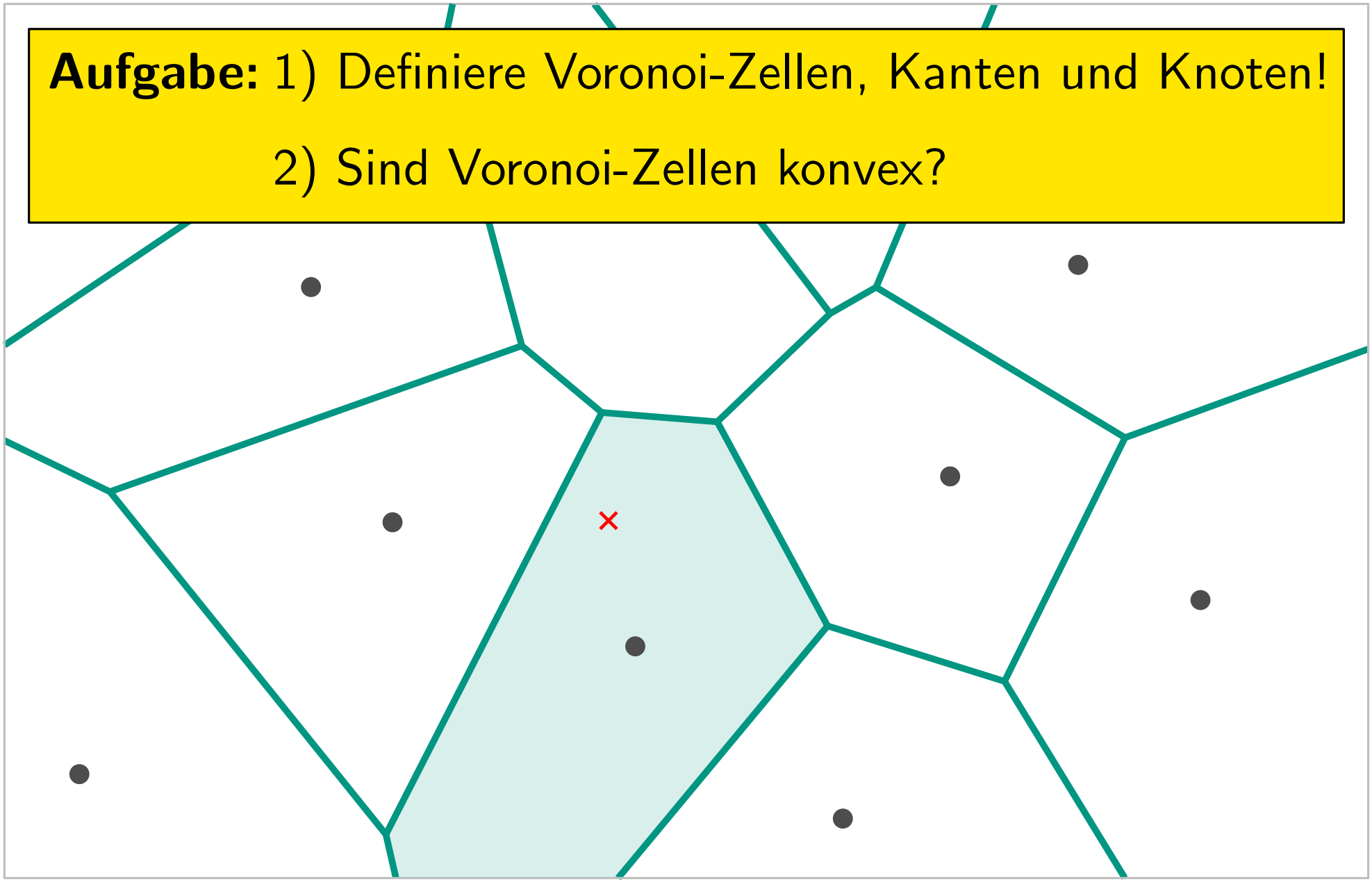
Das Postamt-Problem



Das Postamt-Problem



Das Postamt-Problem

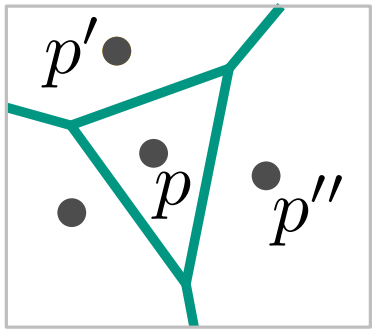


Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Das Voronoi-Diagramm

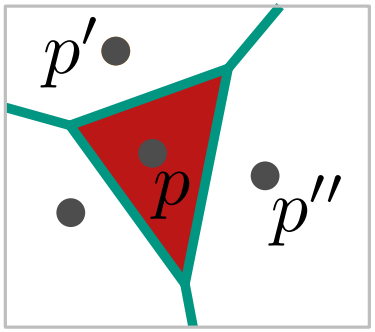
Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

Vor(P)

Das Voronoi-Diagramm

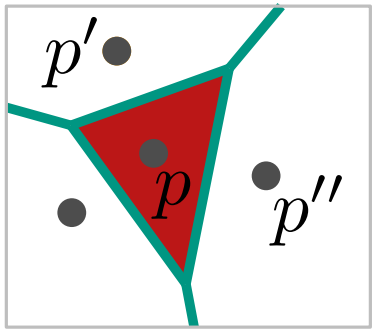
Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

Vor(P)

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

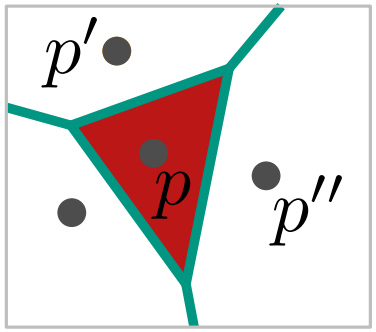
Vor(P)

- Voronoi-Zelle

$\mathcal{V}(\{p\}) =$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

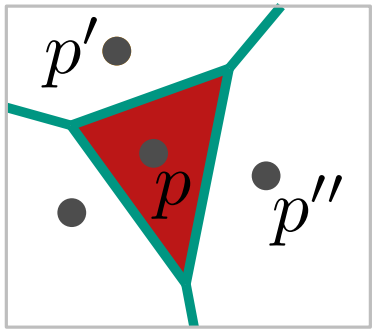
Vor(P)

- Voronoi-Zelle

$\mathcal{V}(\{p\}) = \mathcal{V}(p) =$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

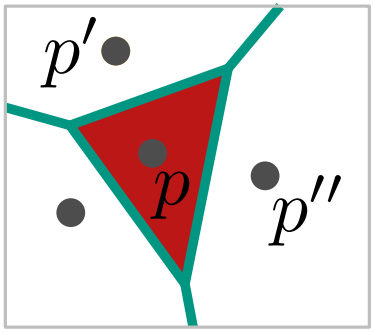
Vor(P)

- Voronoi-Zelle

$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \, \forall q \in P \setminus \{p\}\}$$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

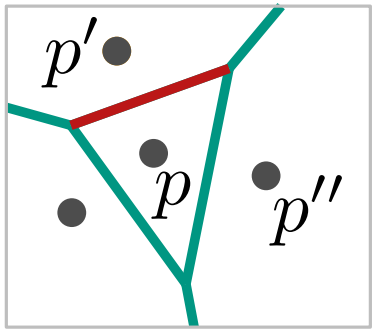
Vor(P)

- Voronoi-Zelle

$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \, \forall q \in P \setminus \{p\}\}$$
$$= \bigcap_{q \neq p} h(p, q)$$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

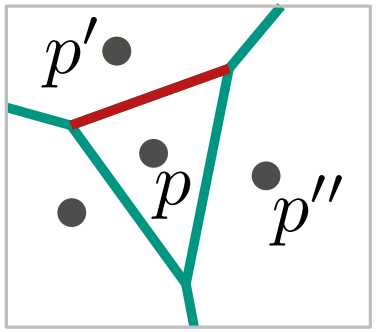
Vor(P)

- Voronoi-Zelle

$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \, \forall q \in P \setminus \{p\}\}$$
$$= \bigcap_{q \neq p} h(p, q)$$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

$\mathrm{Vor}(P)$

- Voronoi-Zelle

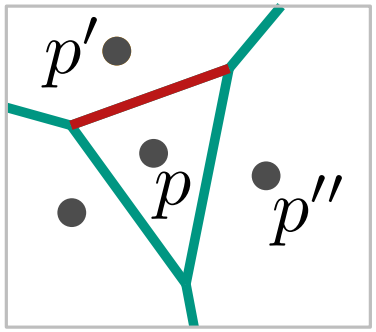
$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \, \forall q \in P \setminus \{p\}\}$$
$$= \bigcap_{q \neq p} h(p, q)$$

- Voronoi-Kante

$$\mathcal{V}(\{p, p'\}) \quad =$$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

Vor(P)

- Voronoi-Zelle

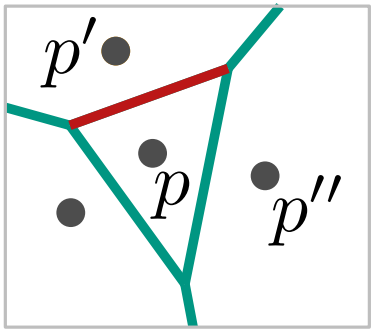
$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \, \forall q \in P \setminus \{p\}\}$$
$$= \bigcap_{q \neq p} h(p, q)$$

- Voronoi-Kante

$$\mathcal{V}(\{p, p'\}) = \{x : |xp| = |xp'| \text{ and } |xp| < |xq| \;\; \forall q \neq p, p'\}$$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

Vor(P)

- Voronoi-Zelle

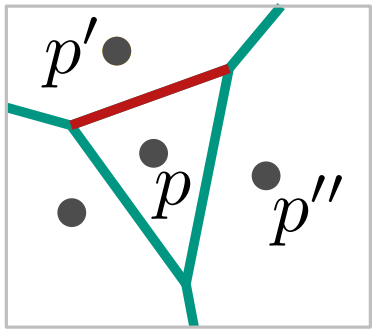
$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \, \forall q \in P \setminus \{p\}\}$$
$$= \bigcap_{q \neq p} h(p, q)$$

- Voronoi-Kante

$$\mathcal{V}(\{p, p'\}) = \{x : |xp| = |xp'| \text{ and } |xp| < |xq| \;\; \forall q \neq p, p'\}$$
$$= \partial\mathcal{V}(p) \cap \partial\mathcal{V}(p')$$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

Vor(P)

- Voronoi-Zelle

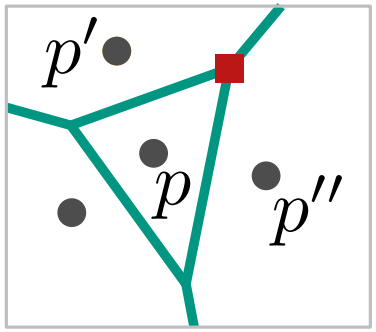
$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \, \forall q \in P \setminus \{p\}\}$$
$$= \bigcap_{q \neq p} h(p, q)$$

- Voronoi-Kante

$$\mathcal{V}(\{p, p'\}) = \{x : |xp| = |xp'| \text{ and } |xp| < |xq| \;\; \forall q \neq p, p'\}$$
$$= \text{rel-int}\big(\partial\mathcal{V}(p) \cap \partial\mathcal{V}(p')\big), \text{ d.h. ohne Endpunkte}$$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

Vor(P)

- Voronoi-Zelle

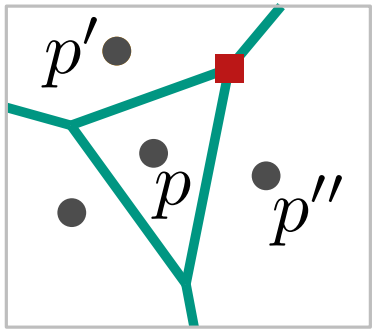
$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \, \forall q \in P \setminus \{p\}\}$$
$$= \bigcap_{q \neq p} h(p, q)$$

- Voronoi-Kante

$$\mathcal{V}(\{p, p'\}) = \{x : |xp| = |xp'| \text{ and } |xp| < |xq| \;\; \forall q \neq p, p'\}$$
$$= \text{rel-int}\big(\partial\mathcal{V}(p) \cap \partial\mathcal{V}(p')\big), \text{ d.h. ohne Endpunkte}$$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

Vor(P)

- Voronoi-Zelle

$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \, \forall q \in P \setminus \{p\}\}$$
$$= \bigcap_{q \neq p} h(p, q)$$

- Voronoi-Kante

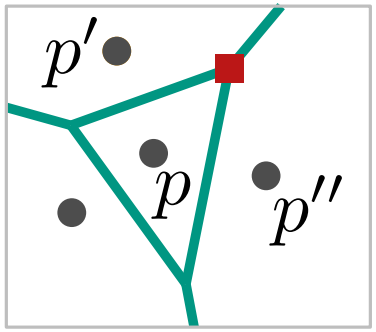
$$\mathcal{V}(\{p, p'\}) = \{x : |xp| = |xp'| \text{ and } |xp| < |xq| \;\; \forall q \neq p, p'\}$$
$$= \text{rel-int}\big(\partial\mathcal{V}(p) \cap \partial\mathcal{V}(p')\big), \text{ d.h. ohne Endpunkte}$$

- Voronoi-Knoten

$$\mathcal{V}(\{p, p', p''\})$$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm 

Vor(P)

- Voronoi-Zelle

$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \, \forall q \in P \setminus \{p\}\}$$
$$= \bigcap_{q \neq p} h(p, q)$$

- Voronoi-Kante

$$\mathcal{V}(\{p, p'\}) = \{x : |xp| = |xp'| \text{ and } |xp| < |xq| \;\; \forall q \neq p, p'\}$$
$$= \text{rel-int}\big(\partial\mathcal{V}(p) \cap \partial\mathcal{V}(p')\big), \text{ d.h. ohne Endpunkte}$$

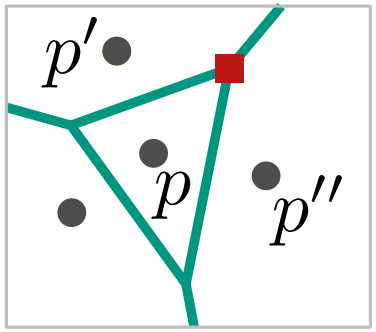
- Voronoi-Knoten

$$\mathcal{V}(\{p, p', p''\}) = \partial\mathcal{V}(p) \cap \partial\mathcal{V}(p') \cap \partial\mathcal{V}(p'')$$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm



$\mathrm{Vor}(P)$

- Voronoi-Zelle

$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \, \forall q \in P \setminus \{p\}\}$$
$$= \bigcap_{q \neq p} h(p, q)$$

- Voronoi-Kante

$$\mathcal{V}(\{p, p'\}) = \{x : |xp| = |xp'| \text{ and } |xp| < |xq| \;\; \forall q \neq p, p'\}$$
$$= \text{rel-int}\big(\partial\mathcal{V}(p) \cap \partial\mathcal{V}(p')\big), \text{ d.h. ohne Endpunkte}$$

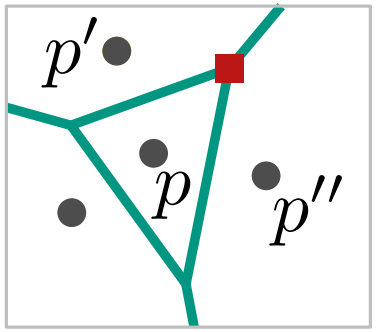
- Voronoi-Knoten

$$\mathcal{V}(\{p, p', p''\}) = \partial\mathcal{V}(p) \cap \partial\mathcal{V}(p') \cap \partial\mathcal{V}(p'')$$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm



$\mathrm{Vor}(P)$ → Unterteilung

- Voronoi-Zelle

$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \, \forall q \in P \setminus \{p\}\}$$
$$= \bigcap_{q \neq p} h(p, q)$$

- Voronoi-Kante

$$\mathcal{V}(\{p, p'\}) = \{x : |xp| = |xp'| \text{ and } |xp| < |xq| \;\; \forall q \neq p, p'\}$$
$$= \text{rel-int}\big(\partial\mathcal{V}(p) \cap \partial\mathcal{V}(p')\big), \text{ d.h. ohne Endpunkte}$$

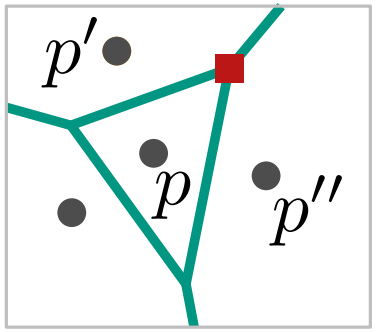
- Voronoi-Knoten

$$\mathcal{V}(\{p, p', p''\}) = \partial\mathcal{V}(p) \cap \partial\mathcal{V}(p') \cap \partial\mathcal{V}(p'')$$

Das Voronoi-Diagramm

Sei P eine Menge von Punkten in der Ebene und $p, p', p'' \in P$.

Voronoi-Diagramm



$\mathrm{Vor}(P)$ → Unterteilung

→ geometr. Graph

- Voronoi-Zelle

$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \, \forall q \in P \setminus \{p\}\}$$
$$= \bigcap_{q \neq p} h(p, q)$$

- Voronoi-Kante

$$\mathcal{V}(\{p, p'\}) = \{x : |xp| = |xp'| \text{ and } |xp| < |xq| \;\; \forall q \neq p, p'\}$$
$$= \text{rel-int}\big(\partial\mathcal{V}(p) \cap \partial\mathcal{V}(p')\big), \text{ d.h. ohne Endpunkte}$$

- Voronoi-Knoten

$$\mathcal{V}(\{p, p', p''\}) = \partial\mathcal{V}(p) \cap \partial\mathcal{V}(p') \cap \partial\mathcal{V}(p'')$$

Eigenschaften

Satz 1: Sei $P \subset \mathbb{R}^2$ eine Menge von n Punkten. Sind alle Punkte kollinear, besteht $\mathrm{Vor}(P)$ aus $n-1$ parallelen Geraden. Sonst ist $\mathrm{Vor}(P)$ zusammenhängend und die Kanten sind Strecken oder Halbgeraden.

Eigenschaften

Satz 1: Sei $P \subset \mathbb{R}^2$ eine Menge von n Punkten. Sind alle Punkte kollinear, besteht $\mathrm{Vor}(P)$ aus $n-1$ parallelen Geraden. Sonst ist $\mathrm{Vor}(P)$ zusammenhängend und die Kanten sind Strecken oder Halbgeraden.

Finde eine Menge P, so dass $\mathrm{Vor}(P)$ eine Zelle linearer Komplexität hat.

Eigenschaften

Satz 1: Sei $P \subset \mathbb{R}^2$ eine Menge von n Punkten. Sind alle Punkte kollinear, besteht $\mathrm{Vor}(P)$ aus $n-1$ parallelen Geraden. Sonst ist $\mathrm{Vor}(P)$ zusammenhängend und die Kanten sind Strecken oder Halbgeraden.

Finde eine Menge P, so dass $\mathrm{Vor}(P)$ eine Zelle linearer Komplexität hat.

Kann das für (fast) jede Zelle passieren?

Eigenschaften

Satz 1: Sei $P \subset \mathbb{R}^2$ eine Menge von n Punkten. Sind alle Punkte kollinear, besteht $\mathrm{Vor}(P)$ aus $n-1$ parallelen Geraden. Sonst ist $\mathrm{Vor}(P)$ zusammenhängend und die Kanten sind Strecken oder Halbgeraden.

Finde eine Menge P, so dass $\mathrm{Vor}(P)$ eine Zelle linearer Komplexität hat.

Kann das für (fast) jede Zelle passieren?

Satz 2: Sei $P \subset \mathbb{R}^2$ Menge von n Punkten. $\mathrm{Vor}(P)$ besteht aus

Eigenschaften

Satz 1: Sei $P \subset \mathbb{R}^2$ eine Menge von n Punkten. Sind alle Punkte kollinear, besteht $\mathrm{Vor}(P)$ aus $n-1$ parallelen Geraden. Sonst ist $\mathrm{Vor}(P)$ zusammenhängend und die Kanten sind Strecken oder Halbgeraden.

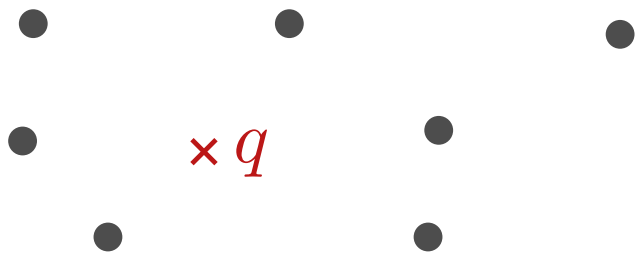
Finde eine Menge P, so dass $\mathrm{Vor}(P)$ eine Zelle linearer Komplexität hat.

Kann das für (fast) jede Zelle passieren?

Satz 2: Sei $P \subset \mathbb{R}^2$ Menge von n Punkten. $\mathrm{Vor}(P)$ besteht aus höchstens $2n-5$ Knoten und $3n-6$ Kanten.

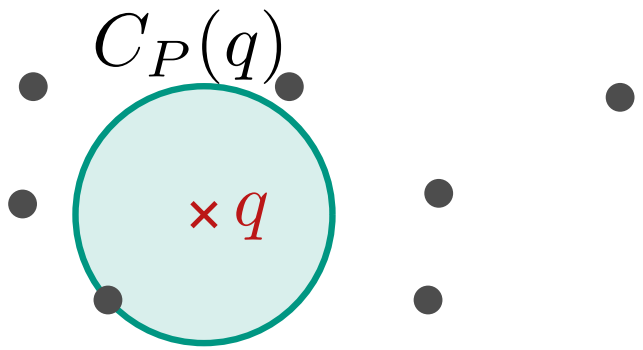
Charakterisierung

Definition: Sei q ein Punkt. Definiere $C_P(q)$ als den bzgl. P größten im Inneren leeren Kreis mit Mittelpunkt q.



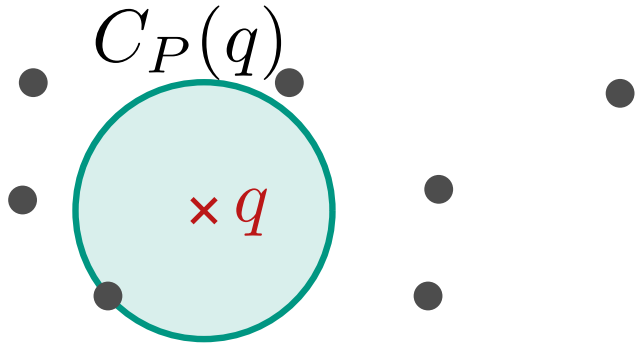
Charakterisierung

Definition: Sei q ein Punkt. Definiere $C_P(q)$ als den bzgl. P größten im Inneren leeren Kreis mit Mittelpunkt q.



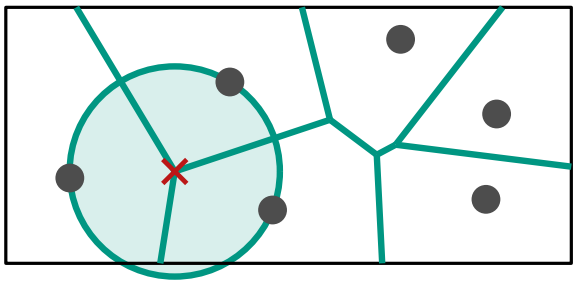
Charakterisierung

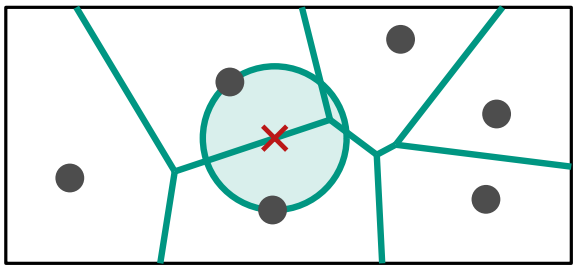
Definition: Sei q ein Punkt. Definiere $C_P(q)$ als den bzgl. P größten im Inneren leeren Kreis mit Mittelpunkt q.



Satz 3:

- Ein Punkt q ist ein Voronoi-Knoten $\Leftrightarrow |C_P(q) \cap P| \geq 3$,
- der Bisektor $b(p_i, p_j)$ definiert eine Voronoi-Kante $\Leftrightarrow \exists q \in b(p_i, p_j)$ mit $C_P(q) \cap P = \{p_i, p_j\}$.





Berechnung von $\mathrm{Vor}(P)$

Für jedes $p \in P$ ist $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ der Schnitt von $n - 1$ Halbebenen.

Wie könnte man $\mathrm{Vor}(P)$ mit schon bekannten Algorithmen berechnen?

Berechnung von $\mathrm{Vor}(P)$

Für jedes $p \in P$ ist $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ der Schnitt von $n - 1$ Halbebenen.

Wie könnte man $\mathrm{Vor}(P)$ mit schon bekannten Algorithmen berechnen?

foreach $p \in P$ **do**

  berechne $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$

Berechnung von $\mathrm{Vor}(P)$

Für jedes $p \in P$ ist $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ der Schnitt von $n - 1$ Halbebenen.

Wie könnte man $\mathrm{Vor}(P)$ mit schon bekannten Algorithmen berechnen?

foreach $p \in P$ **do**

- berechne $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ $O(n \log n)$ [VL 4]

Berechnung von $\mathrm{Vor}(P)$

Für jedes $p \in P$ ist $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ der Schnitt von $n - 1$ Halbebenen.

Wie könnte man $\mathrm{Vor}(P)$ mit schon bekannten Algorithmen berechnen?

foreach $p \in P$ **do** $O(n^2 \log n)$

  berechne $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ $O(n \log n)$ [VL 4]

Berechnung von $\mathrm{Vor}(P)$

Für jedes $p \in P$ ist $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ der Schnitt von $n - 1$ Halbebenen.

Wie könnte man $\mathrm{Vor}(P)$ mit schon bekannten Algorithmen berechnen?

foreach $p \in P$ **do** $O(n^2 \log n)$

  berechne $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ $O(n \log n)$ [VL 4]

Ist $O(n^2 \log n)$ Laufzeit für ein Objekt linearer Größe nötig?

Berechnung von $\mathrm{Vor}(P)$

Für jedes $p \in P$ ist $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ der Schnitt von $n - 1$ Halbebenen.

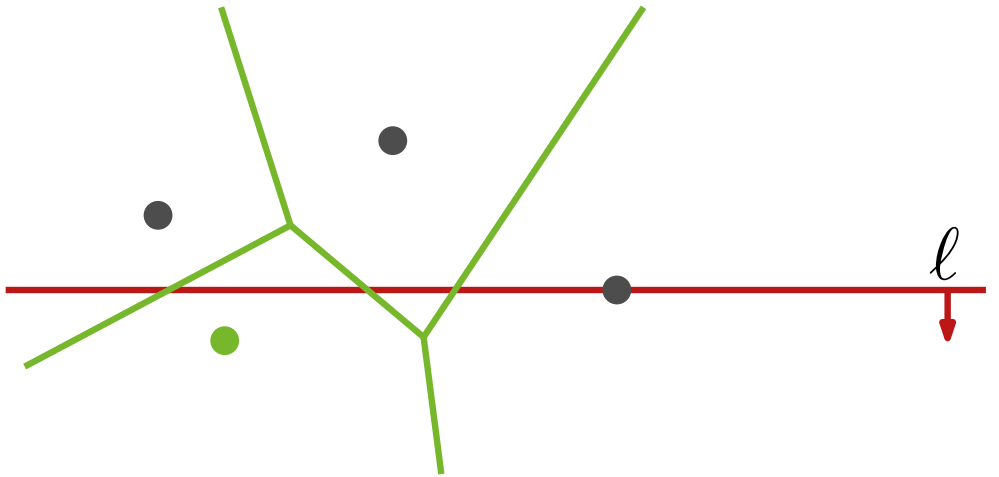
Wie könnte man $\mathrm{Vor}(P)$ mit schon bekannten Algorithmen berechnen?

foreach $p \in P$ **do** $O(n^2 \log n)$

  berechne $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ $O(n \log n)$ [VL 4]

Ist $O(n^2 \log n)$ Laufzeit für ein Objekt linearer Größe nötig?

Idee 2: Sweep-Verfahren



Berechnung von $\mathrm{Vor}(P)$

Für jedes $p \in P$ ist $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ der Schnitt von $n - 1$ Halbebenen.

Wie könnte man $\mathrm{Vor}(P)$ mit schon bekannten Algorithmen berechnen?

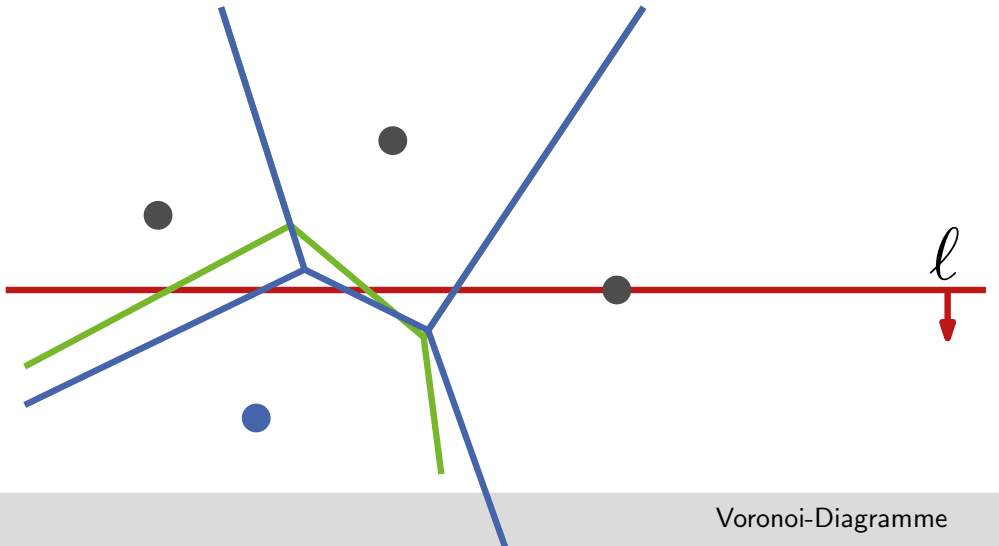
foreach $p \in P$ **do** $O(n^2 \log n)$

  berechne $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ $O(n \log n)$ [VL 4]

Ist $O(n^2 \log n)$ Laufzeit für ein Objekt linearer Größe nötig?

Idee 2: Sweep-Verfahren

Problem:
$\mathrm{Vor}(P)$ oberhalb ℓ hängt von Punkten unterhalb ℓ ab!



Berechnung von $\mathrm{Vor}(P)$

Für jedes $p \in P$ ist $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ der Schnitt von $n - 1$ Halbebenen.

Wie könnte man $\mathrm{Vor}(P)$ mit schon bekannten Algorithmen berechnen?

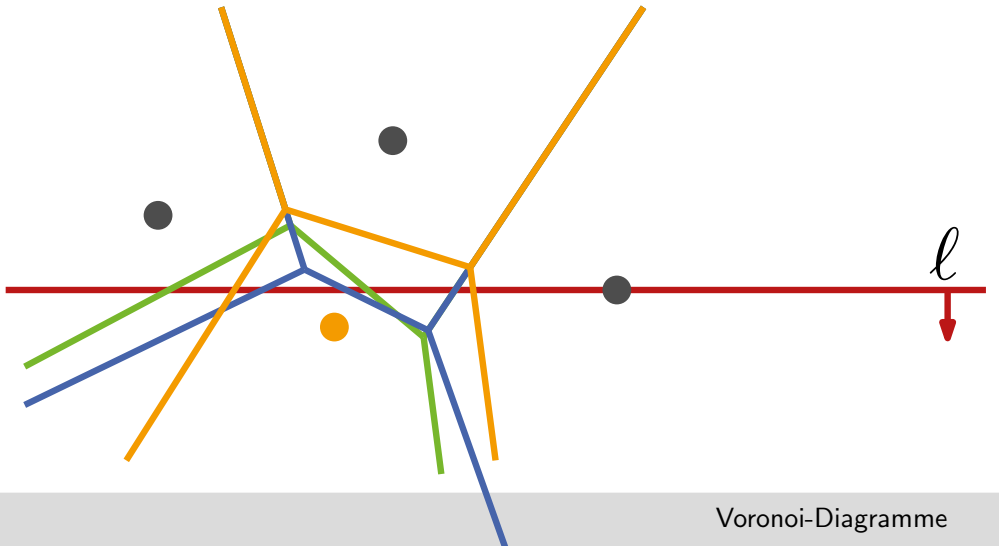
foreach $p \in P$ **do** $O(n^2 \log n)$

  berechne $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$ $O(n \log n)$ [VL 4]

Ist $O(n^2 \log n)$ Laufzeit für ein Objekt linearer Größe nötig?

Idee 2: Sweep-Verfahren

Problem:
$\mathrm{Vor}(P)$ oberhalb ℓ hängt von Punkten unterhalb ℓ ab!



In Richtung Sweep-Verfahren

Offensichtlich ist der Schnitt von $\mathrm{Vor}(P)$ und Sweep Line ℓ zum aktuellen Zeitpunkt noch nicht bekannt.

In Richtung Sweep-Verfahren

Offensichtlich ist der Schnitt von $\mathrm{Vor}(P)$ und Sweep Line ℓ zum aktuellen Zeitpunkt noch nicht bekannt.

Betrachte stattdessen den Teil oberhalb ℓ, der schon fest ist!

In Richtung Sweep-Verfahren

Offensichtlich ist der Schnitt von $\mathrm{Vor}(P)$ und Sweep Line ℓ zum aktuellen Zeitpunkt noch nicht bekannt.

Betrachte stattdessen den Teil oberhalb ℓ, der schon fest ist!

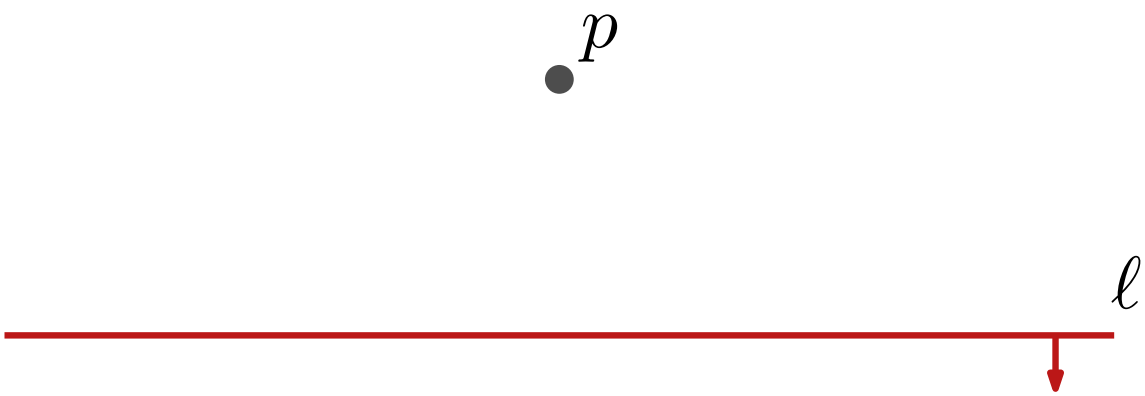
Wie sieht der aus?

In Richtung Sweep-Verfahren

Offensichtlich ist der Schnitt von $\mathrm{Vor}(P)$ und Sweep Line ℓ zum aktuellen Zeitpunkt noch nicht bekannt.

Betrachte stattdessen den Teil oberhalb ℓ, der schon fest ist!

Wie sieht der aus?

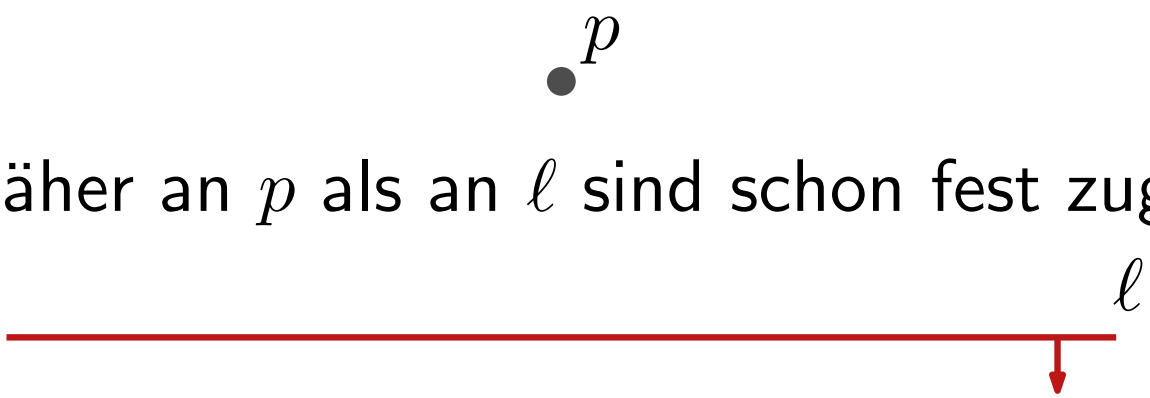


In Richtung Sweep-Verfahren

Offensichtlich ist der Schnitt von $\mathrm{Vor}(P)$ und Sweep Line ℓ zum aktuellen Zeitpunkt noch nicht bekannt.

Betrachte stattdessen den Teil oberhalb ℓ, der schon fest ist!

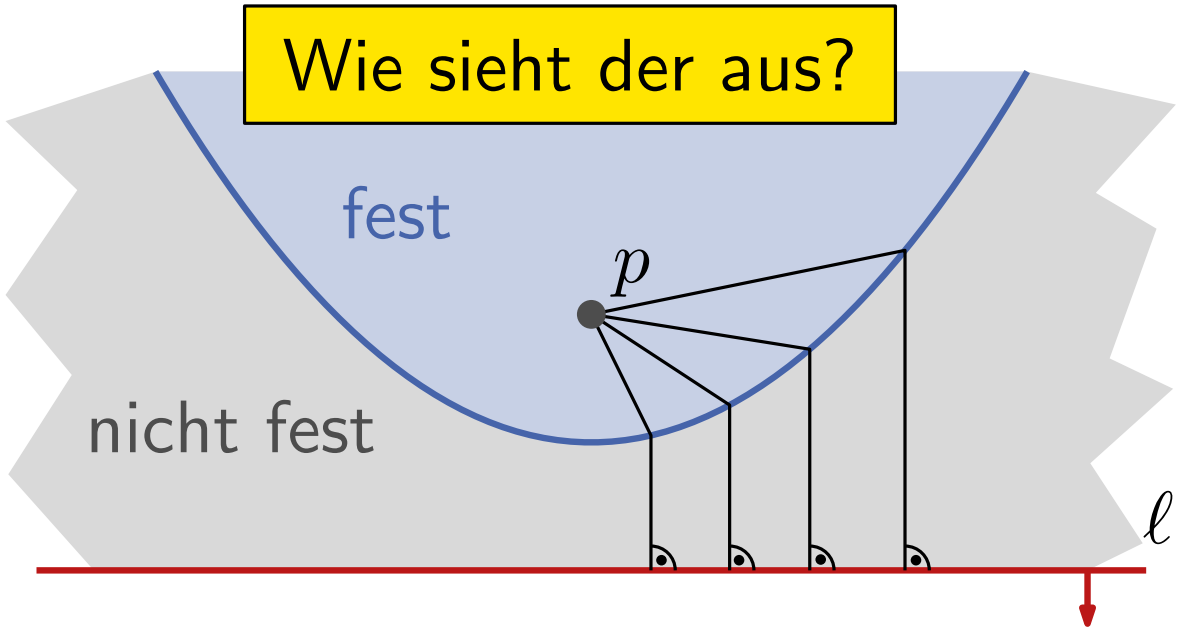
Wie sieht der aus?



In Richtung Sweep-Verfahren

Offensichtlich ist der Schnitt von $\mathrm{Vor}(P)$ und Sweep Line ℓ zum aktuellen Zeitpunkt noch nicht bekannt.

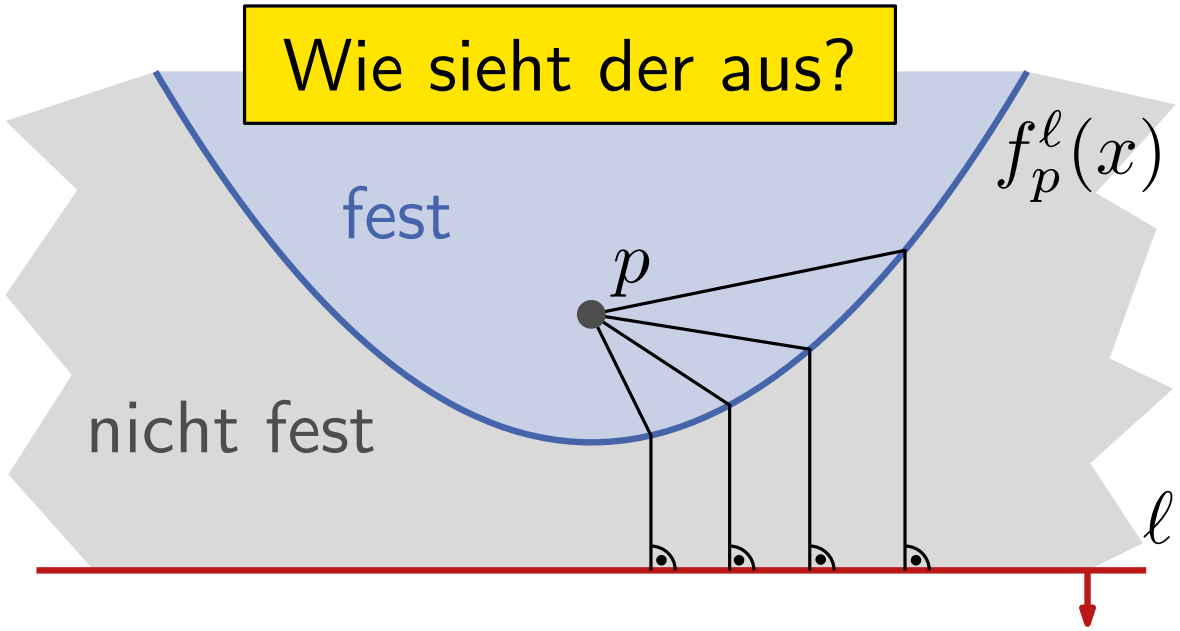
Betrachte stattdessen den Teil oberhalb ℓ, der schon fest ist!



In Richtung Sweep-Verfahren

Offensichtlich ist der Schnitt von $\mathrm{Vor}(P)$ und Sweep Line ℓ zum aktuellen Zeitpunkt noch nicht bekannt.

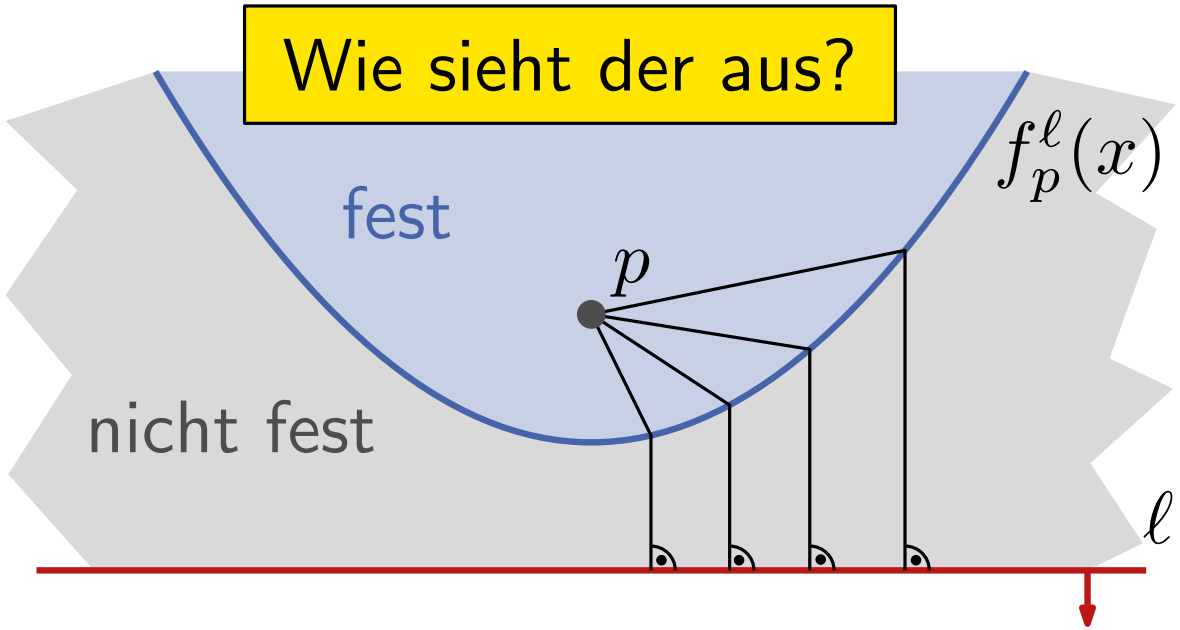
Betrachte stattdessen den Teil oberhalb ℓ, der schon fest ist!



In Richtung Sweep-Verfahren

Offensichtlich ist der Schnitt von $\mathrm{Vor}(P)$ und Sweep Line ℓ zum aktuellen Zeitpunkt noch nicht bekannt.

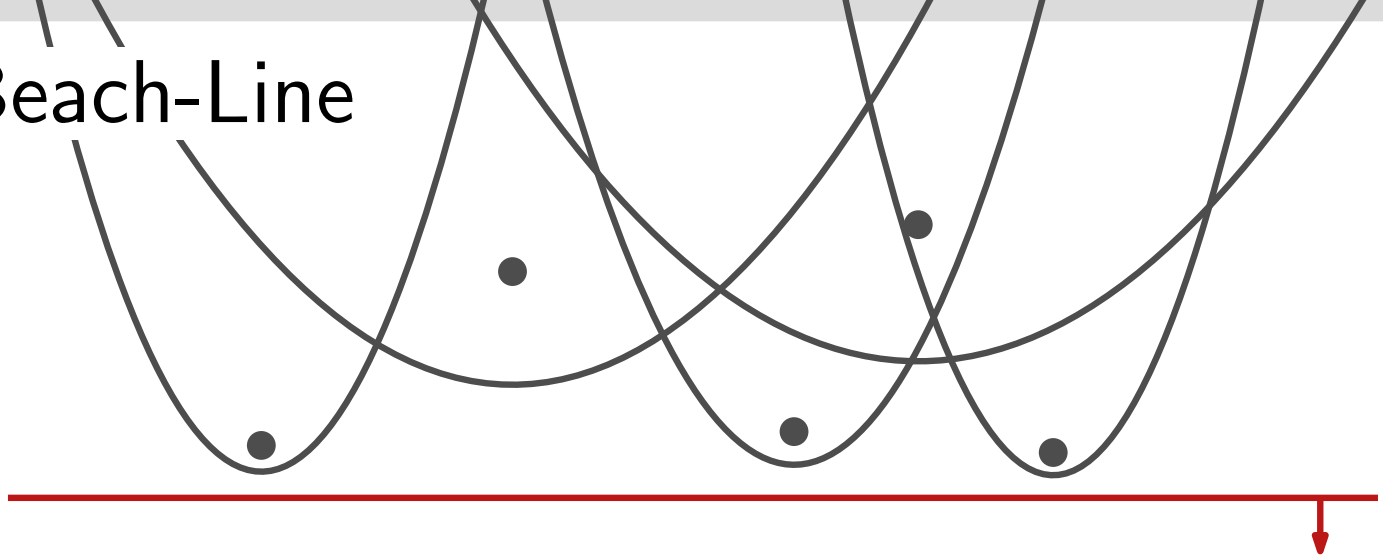
Betrachte stattdessen den Teil oberhalb ℓ, der schon fest ist!



Lösen der Gleichung $|pq| = |q\ell|$ liefert

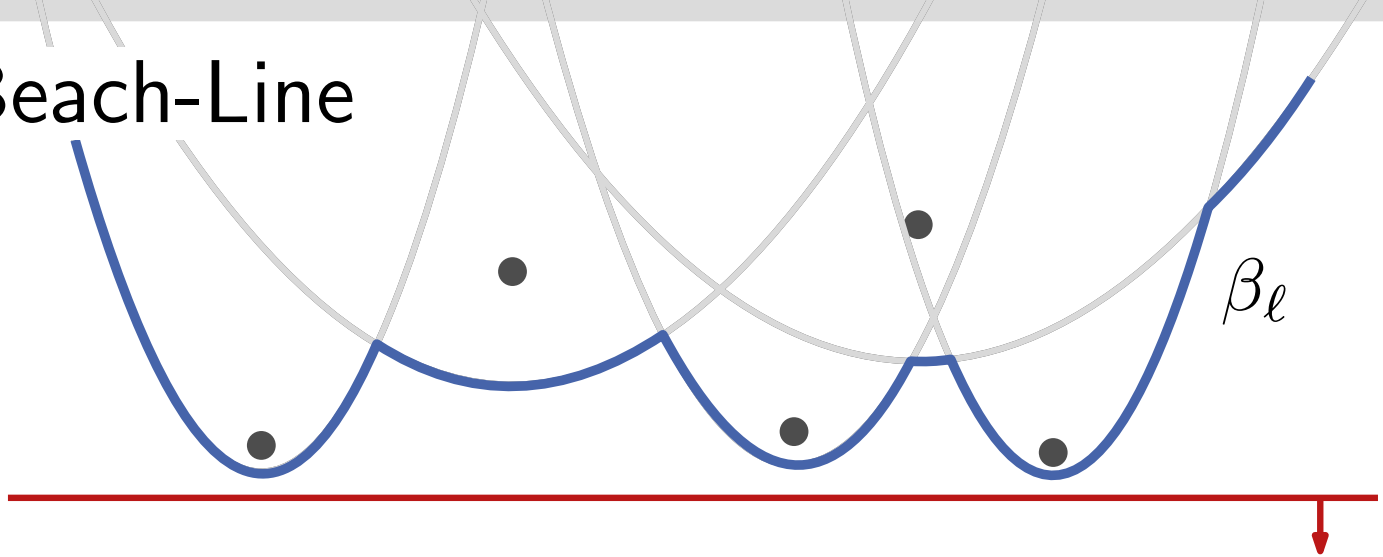
$$f_p^\ell(x) = \frac{1}{2(p_y - \ell_y)}(x - p_x)^2 + \frac{p_y + \ell_y}{2}$$

Die Beach-Line



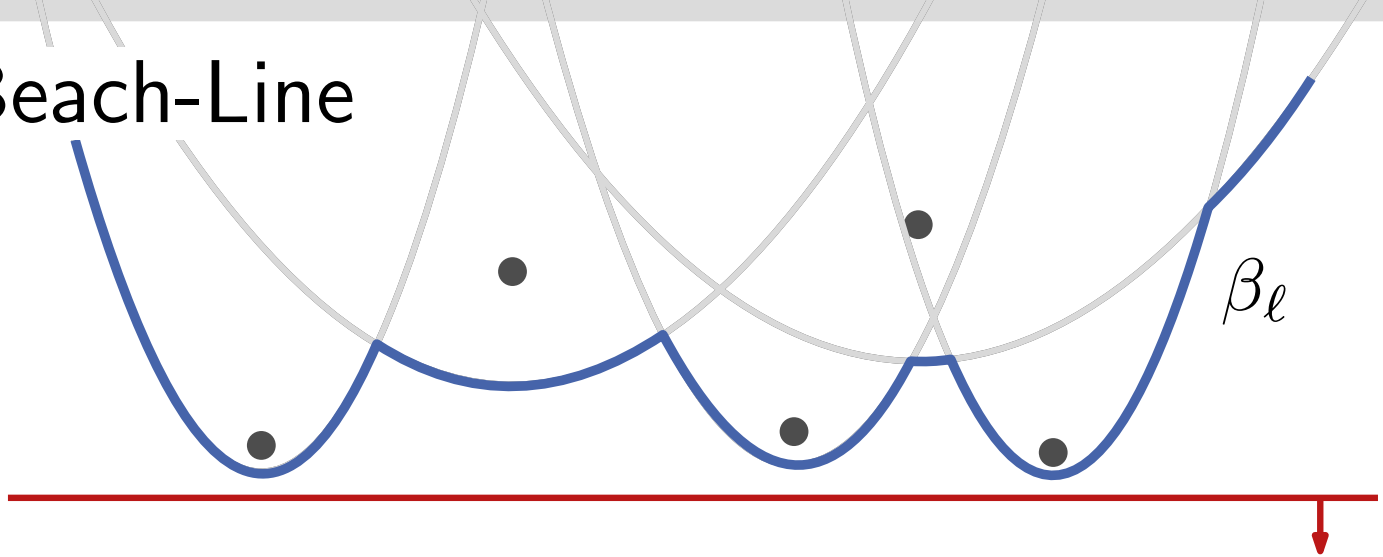
Definition: Die **Beach-Line** β_ℓ ist die untere Kontur der Parabeln f_p^ℓ für die bereits besuchten Punkte.

Die Beach-Line



Definition: Die **Beach-Line** β_ℓ ist die untere Kontur der Parabeln f_p^ℓ für die bereits besuchten Punkte.

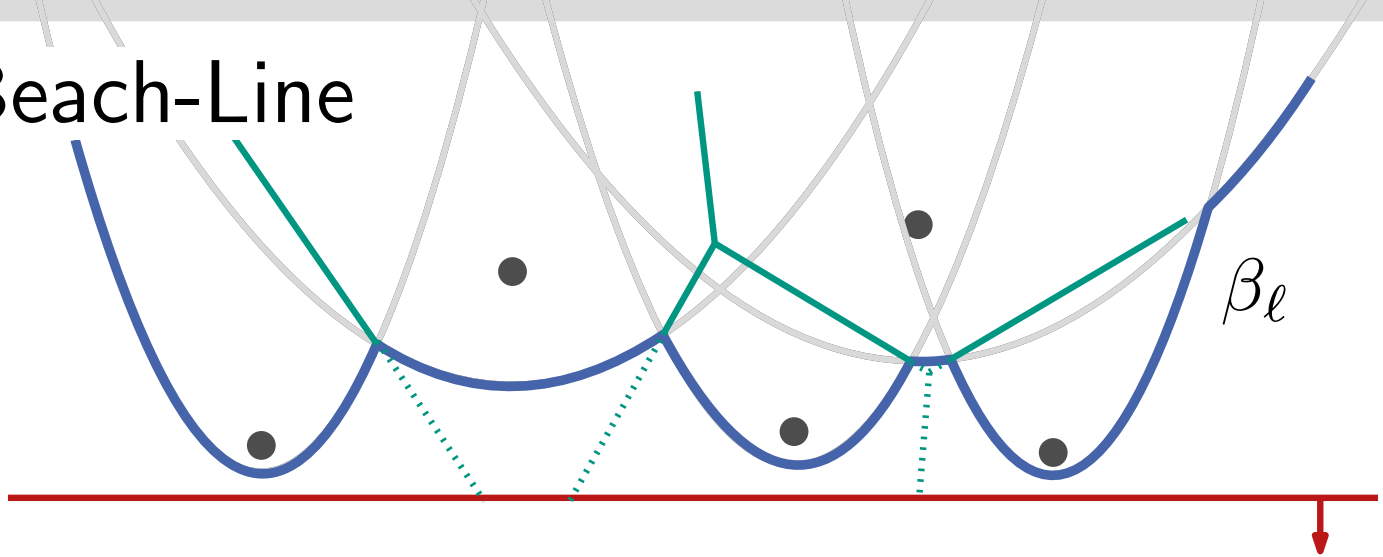
Die Beach-Line



Definition: Die **Beach-Line** β_ℓ ist die untere Kontur der Parabeln f_p^ℓ für die bereits besuchten Punkte.

Was hat das mit $\mathrm{Vor}(P)$ zu tun?

Die Beach-Line



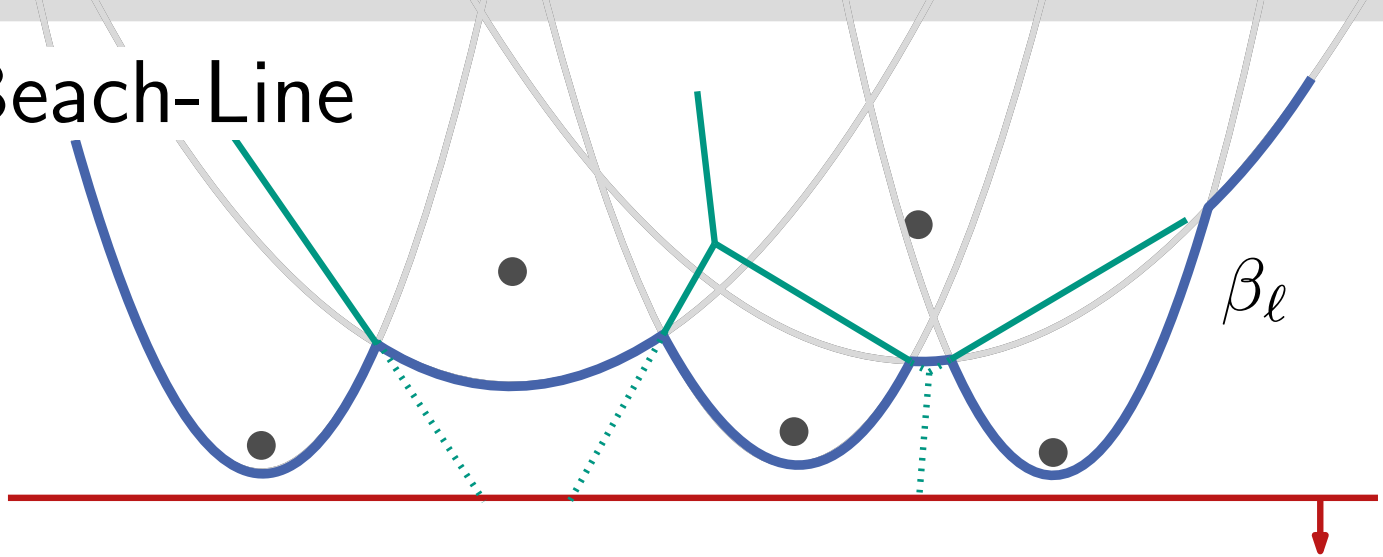
Definition: Die **Beach-Line** β_ℓ ist die untere Kontur der Parabeln f_p^ℓ für die bereits besuchten Punkte.

Was hat das mit $\mathrm{Vor}(P)$ zu tun?

Beob.:

- Beach-Line ist x-monoton
- Schnittpunkte der Beach-Line liegen auf Voronoi-Kanten

Die Beach-Line



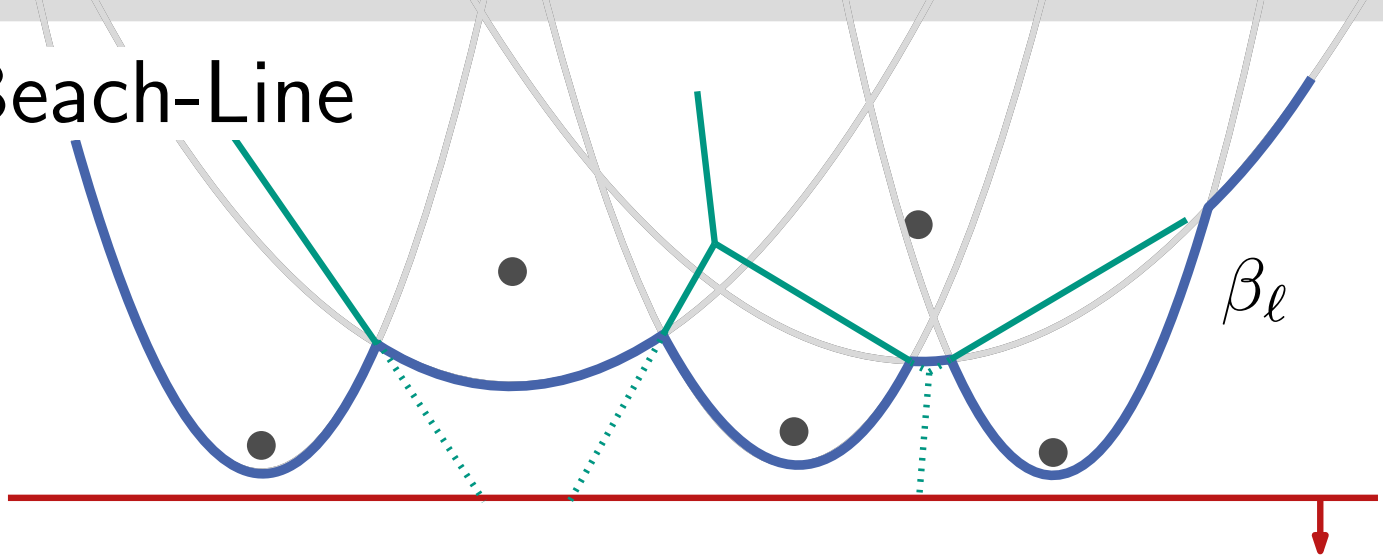
Definition: Die **Beach-Line** β_ℓ ist die untere Kontur der Parabeln f_p^ℓ für die bereits besuchten Punkte.

Was hat das mit $\mathrm{Vor}(P)$ zu tun?

Beob.:

- Beach-Line ist x-monoton
- Schnittpunkte der Beach-Line liegen auf Voronoi-Kanten
  sogar: Schnittpunkte laufen entlang $\mathrm{Vor}(P)$

Die Beach-Line



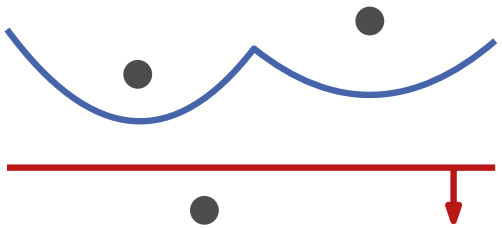
Definition: Die **Beach-Line** β_ℓ ist die untere Kontur der Parabeln f_p^ℓ für die bereits besuchten Punkte.

Was hat das mit $\mathrm{Vor}(P)$ zu tun?

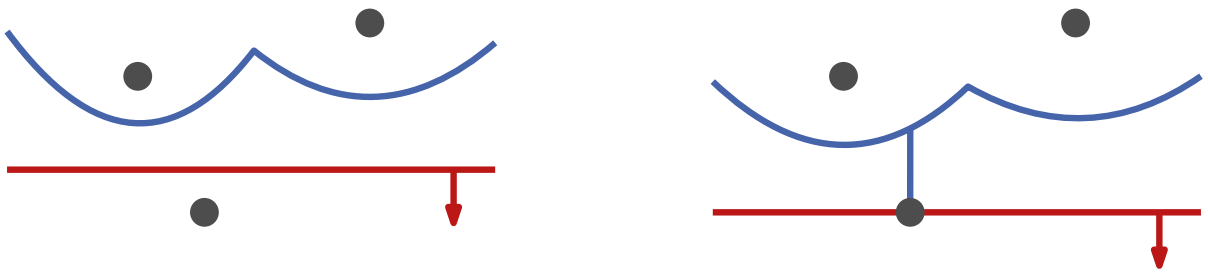
Beob.:
- Beach-Line ist x-monoton
- Schnittpunkte der Beach-Line liegen auf Voronoi-Kanten
  sogar: Schnittpunkte laufen entlang $\mathrm{Vor}(P)$

Ziel: speichere (implizit) aktuelle Kontur β_ℓ statt $\mathrm{Vor}(P) \cap \ell$

Punkt-Events

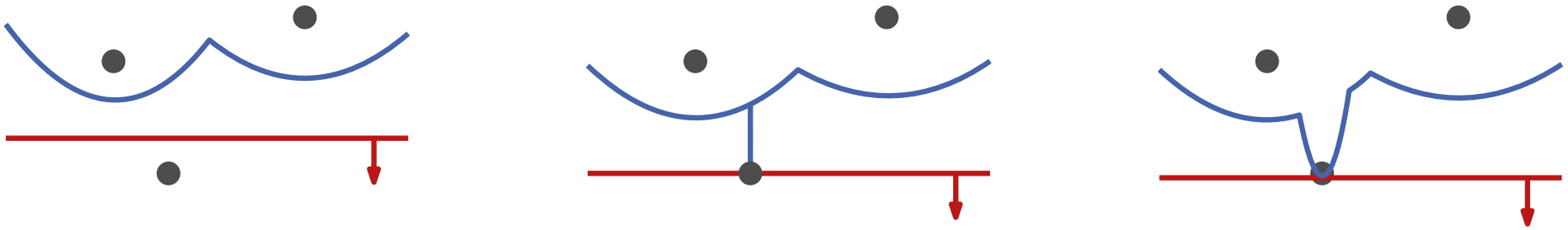


Punkt-Events



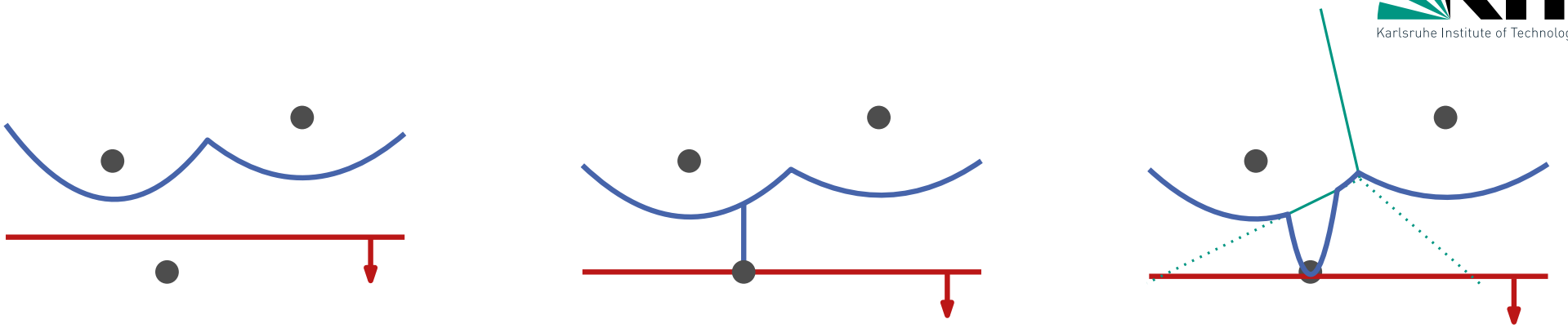
- trifft ℓ auf einen Punkt, kommt neue Parabel zu β_ℓ hinzu

Punkt-Events



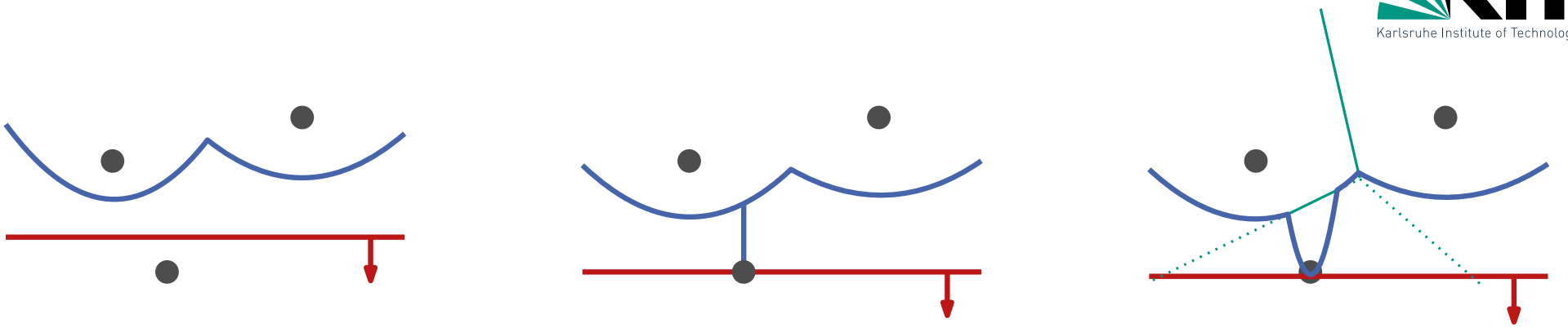
- trifft ℓ auf einen Punkt, kommt neue Parabel zu β_ℓ hinzu

Punkt-Events



- trifft ℓ auf einen Punkt, kommt neue Parabel zu β_ℓ hinzu
- die beiden Schnittpunkte erzeugen neue Teilkante

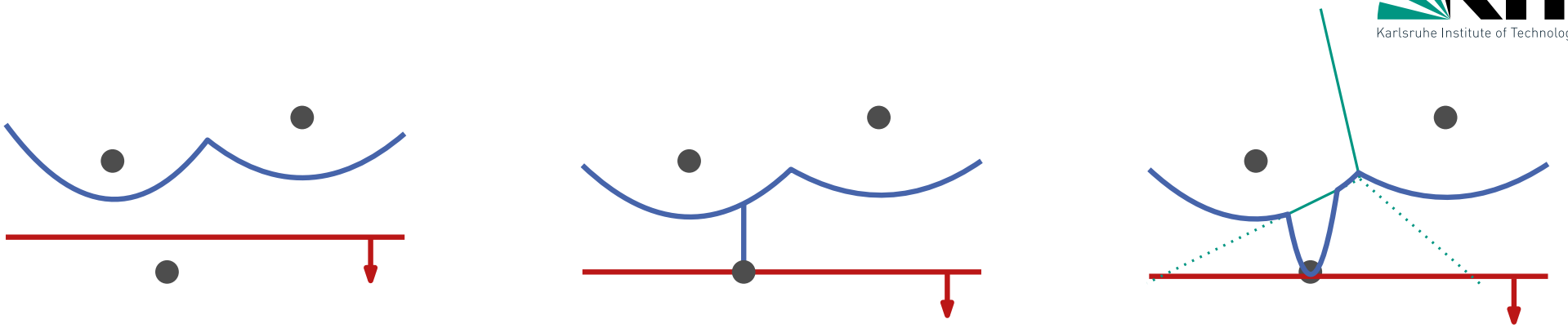
Punkt-Events



- trifft ℓ auf einen Punkt, kommt neue Parabel zu β_ℓ hinzu
- die beiden Schnittpunkte erzeugen neue Teilkante

Lemma 1: Neue Bögen auf β entstehen nur durch Punkt-Events.

Punkt-Events

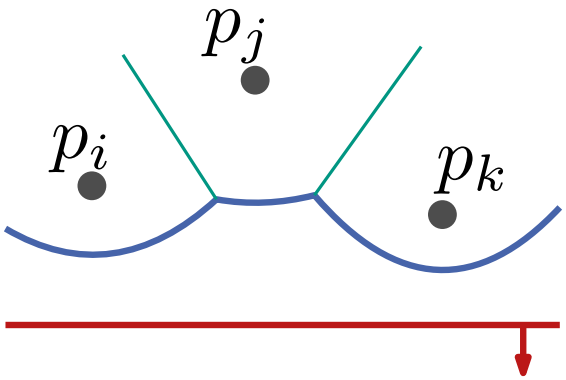


- trifft ℓ auf einen Punkt, kommt neue Parabel zu β_ℓ hinzu
- die beiden Schnittpunkte erzeugen neue Teilkante

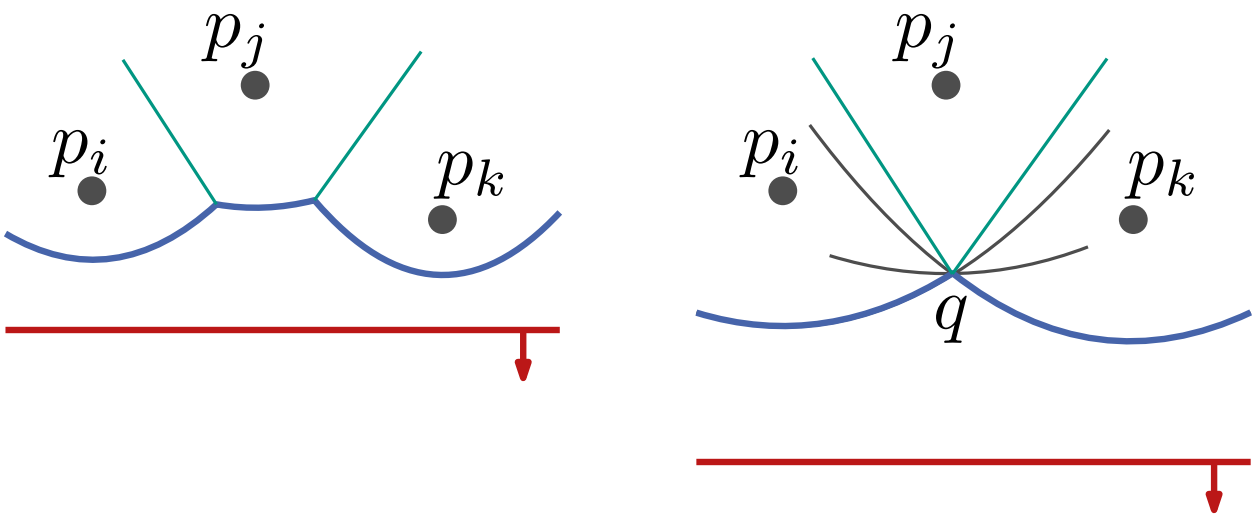
Lemma 1: Neue Bögen auf β entstehen nur durch Punkt-Events.

Korollar: β besteht aus maximal $2n - 1$ Parabelbögen

Kreis-Events

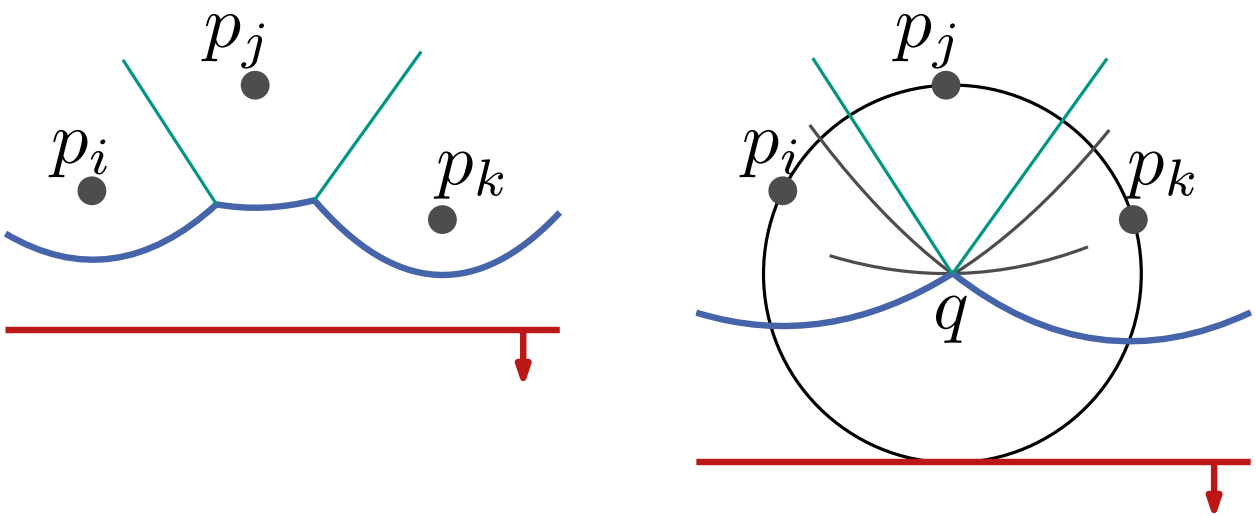


Kreis-Events



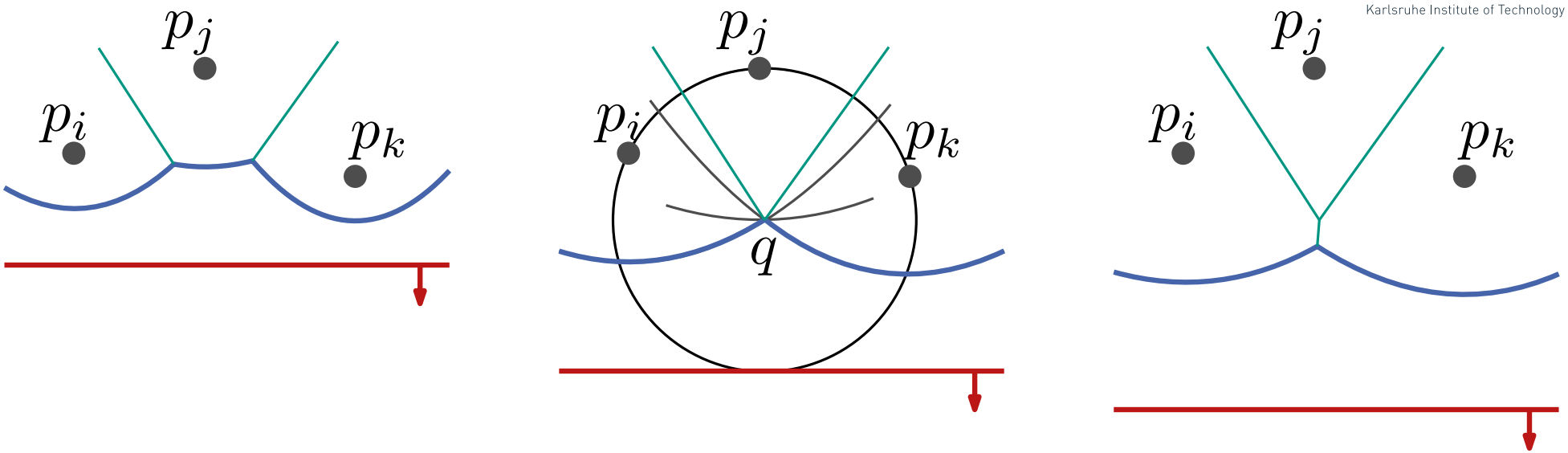
- verschwindet ein Bogen so laufen $f^{\ell}_{p_i}, f^{\ell}_{p_j}, f^{\ell}_{p_k}$ durch einen gemeinsamen Punkt q

Kreis-Events



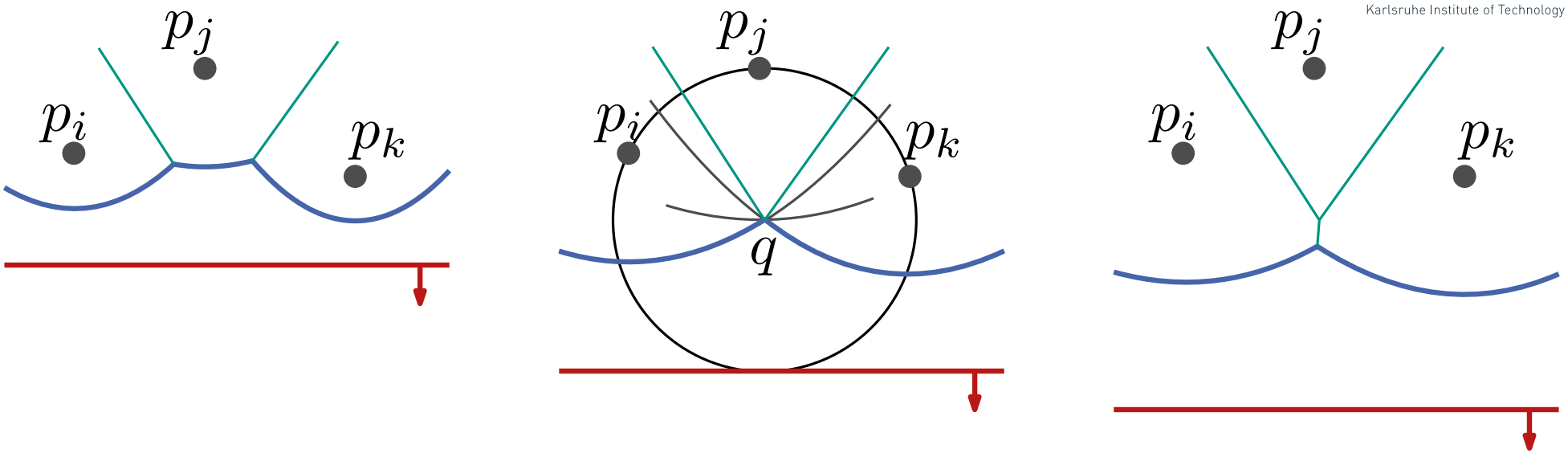
- verschwindet ein Bogen so laufen $f^{\ell}_{p_i}, f^{\ell}_{p_j}, f^{\ell}_{p_k}$ durch einen gemeinsamen Punkt q
- Kreis $C_P(q)$ geht durch p_i, p_j, p_k und berührt ℓ
  $\Rightarrow q$ ist Voronoi-Knoten

Kreis-Events



- verschwindet ein Bogen so laufen $f^{\ell}_{p_i}, f^{\ell}_{p_j}, f^{\ell}_{p_k}$ durch einen gemeinsamen Punkt q
- Kreis $C_P(q)$ geht durch p_i, p_j, p_k und berührt ℓ
  $\Rightarrow q$ ist Voronoi-Knoten

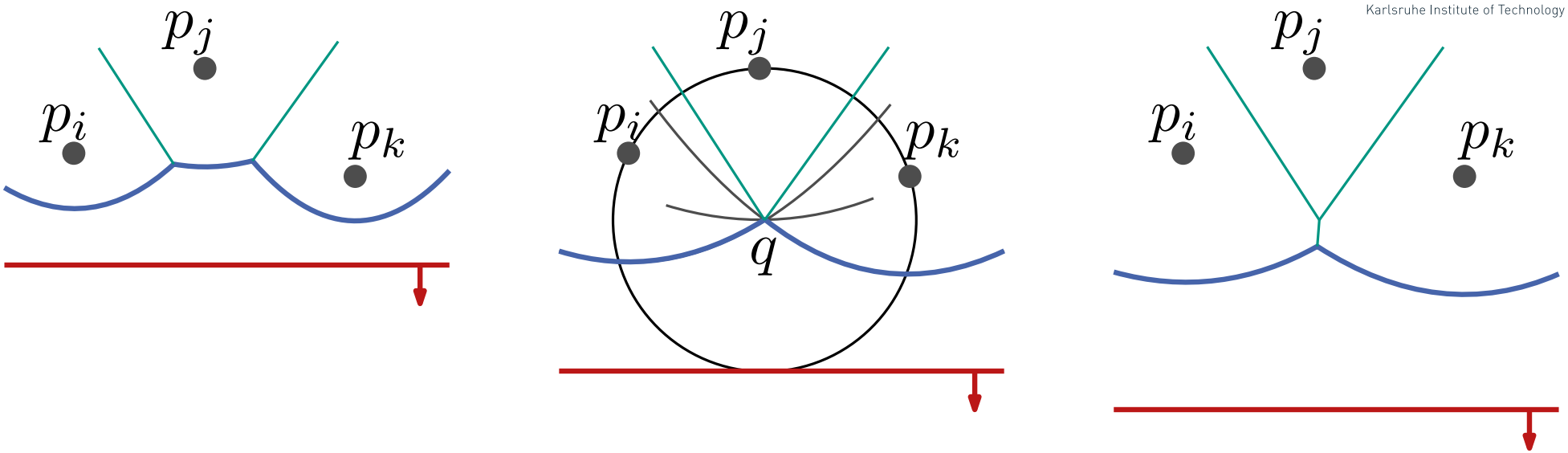
Kreis-Events



- verschwindet ein Bogen so laufen $f^{\ell}_{p_i}, f^{\ell}_{p_j}, f^{\ell}_{p_k}$ durch einen gemeinsamen Punkt q
- Kreis $C_P(q)$ geht durch p_i, p_j, p_k und berührt ℓ
  $\Rightarrow q$ ist Voronoi-Knoten

Def.: Der unterste Punkt des Kreises durch drei Punkte mit konsekutiven Bögen auf β definiert ein **Kreis-Event**.

Kreis-Events

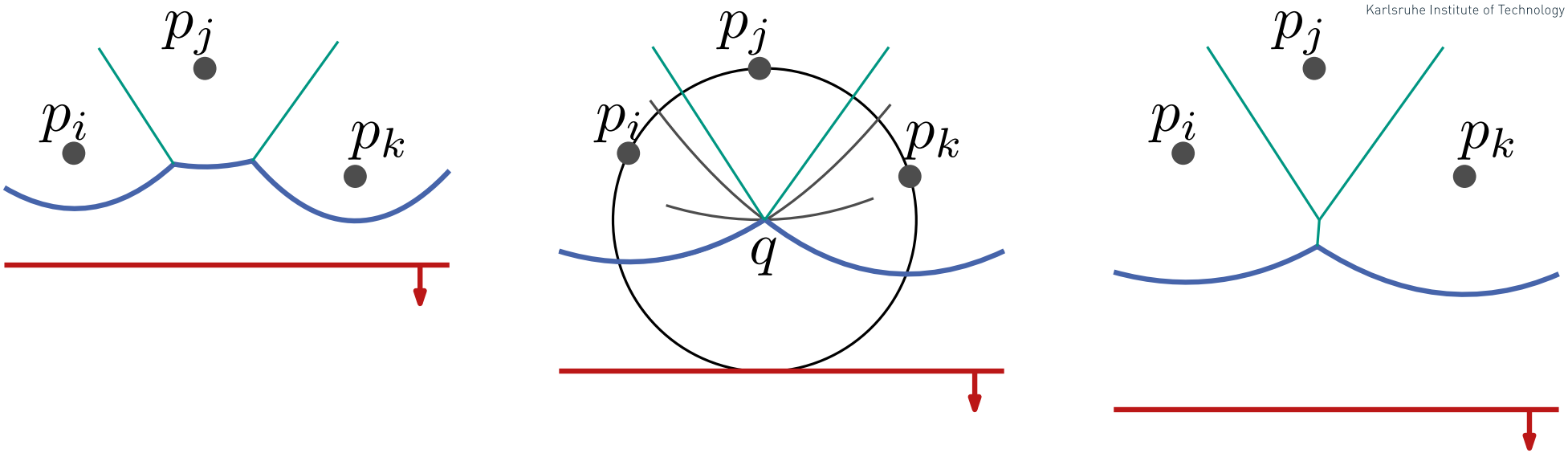


- verschwindet ein Bogen so laufen $f^{\ell}_{p_i}, f^{\ell}_{p_j}, f^{\ell}_{p_k}$ durch einen gemeinsamen Punkt q
- Kreis $C_P(q)$ geht durch p_i, p_j, p_k und berührt ℓ
  $\Rightarrow q$ ist Voronoi-Knoten

Def.: Der unterste Punkt des Kreises durch drei Punkte mit konsekutiven Bögen auf β definiert ein **Kreis-Event**.

Lemma 2: Bögen von β werden nur durch Kreis-Events entfernt.

Kreis-Events



- verschwindet ein Bogen so laufen $f^{\ell}_{p_i}, f^{\ell}_{p_j}, f^{\ell}_{p_k}$ durch einen gemeinsamen Punkt q
- Kreis $C_P(q)$ geht durch p_i, p_j, p_k und berührt ℓ
  $\Rightarrow q$ ist Voronoi-Knoten

Def.: Der unterste Punkt des Kreises durch drei Punkte mit konsekutiven Bögen auf β definiert ein **Kreis-Event**.

Lemma 2: Bögen von β werden nur durch Kreis-Events entfernt.

Lemma 3: Für jeden Voronoi-Knoten gibt es ein Kreis-Event.

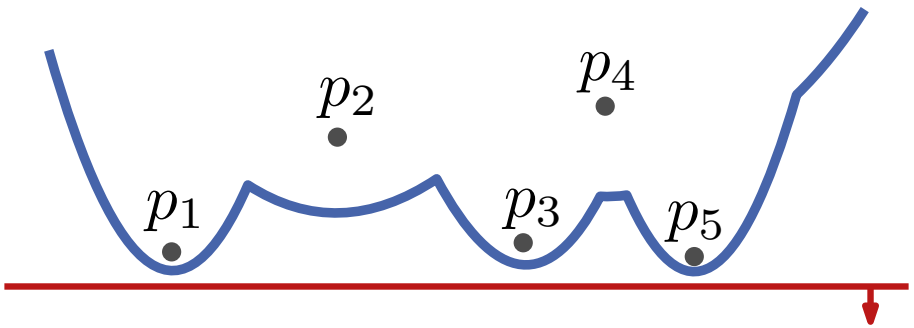
Datenstrukturen

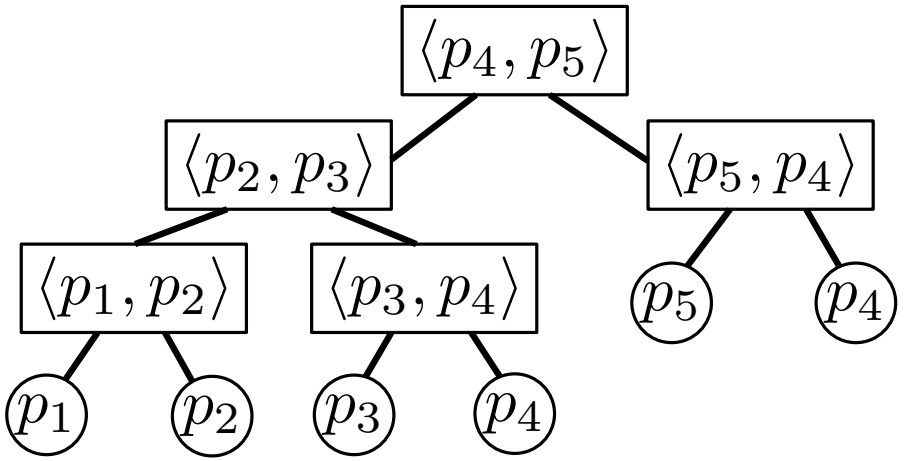
- doppelt-verkettete Kantenliste (DCEL) $\mathcal{D}$ für $\mathrm{Vor}(P)$
  Achtung: am Schluss bounding box wg. Halbgeraden einfügen

Datenstrukturen

- doppelt-verkettete Kantenliste (DCEL) $\mathcal{D}$ für $\mathrm{Vor}(P)$
  Achtung: am Schluss bounding box wg. Halbgeraden einfügen

- balancierter binärer Suchbaum $\mathcal{T}$ für implizite Beach-Line
  – Blätter entsprechen Parabelbögen von links nach rechts
  – innerer Knoten $\langle p_i, p_j \rangle$ entspricht Schnittpunkt von f_{p_i} und f_{p_j}
  – Pointer von inneren Knoten auf zugeh. Kanten in $\mathcal{D}$

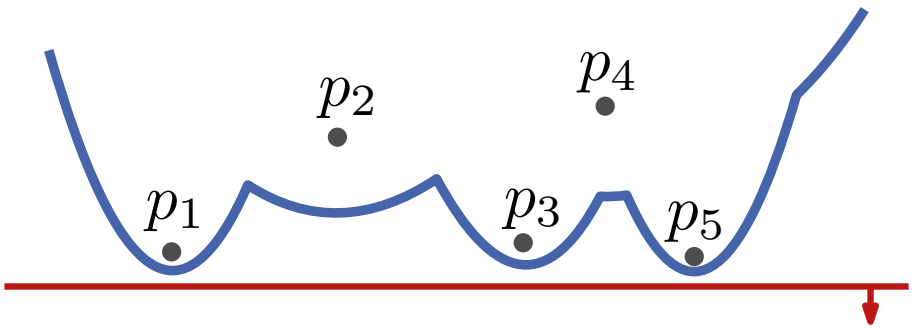


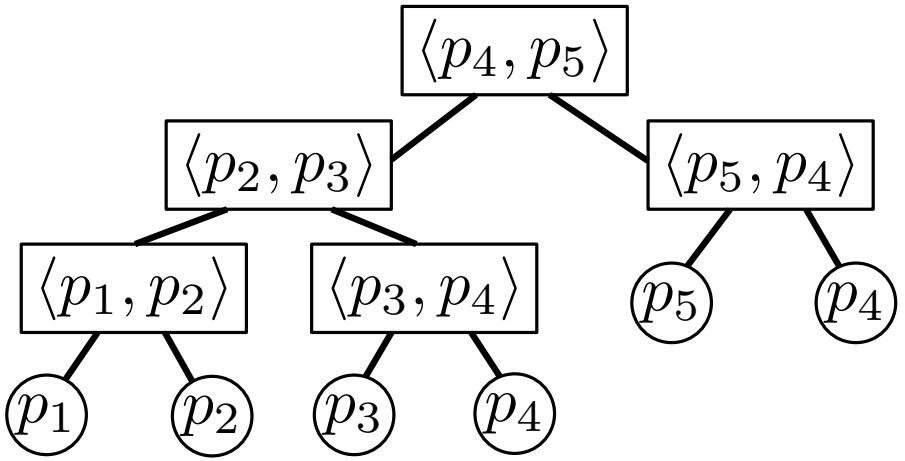


Datenstrukturen

- doppelt-verkettete Kantenliste (DCEL) $\mathcal{D}$ für $\mathrm{Vor}(P)$
  Achtung: am Schluss bounding box wg. Halbgeraden einfügen

- balancierter binärer Suchbaum $\mathcal{T}$ für implizite Beach-Line
  – Blätter entsprechen Parabelbögen von links nach rechts
  – innerer Knoten $\langle p_i, p_j \rangle$ entspricht Schnittpunkt von f_{p_i} und f_{p_j}
  – Pointer von inneren Knoten auf zugeh. Kanten in $\mathcal{D}$





- Priority Queue $\mathcal{Q}$ für die Punkt- und Kreis-Events
  – Pointer von Kreis-Events auf zugeh. Blätter in $\mathcal{T}$ und umgekehrt

Fortune's Sweep Algorithmus

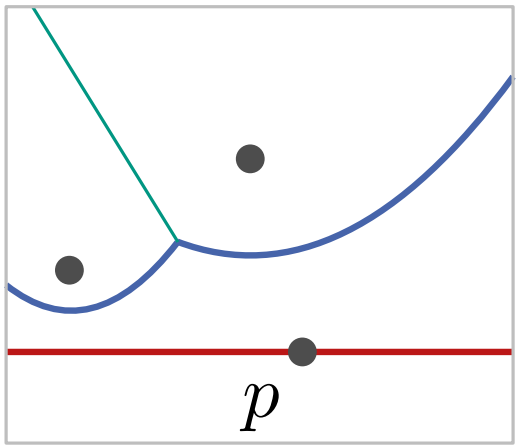
```
VoronoiDiagram(P ⊂ ℝ²)
  Q ← new PriorityQueue(P)    // Punkt-Events sortiert nach y
  T ← new BalancedBinarySearchTree()    // sweep status (β)
  D ← new DCEL()    // DS für Vor(P)
  while not Q.empty() do
      p ← Q.ExtractMax()
      if p Punkt-Event then
          HandlePointEvent(p)
      else
          α ← Bogen von β, der entfernt werden soll
          HandleCircleEvent(α)
  behandle innere Restknoten von T (Halbgeraden von Vor(P))
  return D
```

Fortune's Sweep Algorithmus

VoronoiDiagram($P \subset \mathbb{R}^2$)
 $\mathcal{Q} \leftarrow$ new PriorityQueue(P) // Punkt-Events sortiert nach y
 $\mathcal{T} \leftarrow$ new BalancedBinarySearchTree() // sweep status (β)
 $\mathcal{D} \leftarrow$ new DCEL() // DS für Vor(P)
 while not $\mathcal{Q}$.empty() **do**
  $p \leftarrow \mathcal{Q}$.ExtractMax()
  if p Punkt-Event **then**
   HandlePointEvent(p)
  else
   $\alpha \leftarrow$ Bogen von β, der entfernt werden soll
   HandleCircleEvent(α)
 behandle innere Restknoten von $\mathcal{T}$ (Halbgeraden von Vor(P))
 return $\mathcal{D}$

Punkt-Events behandeln

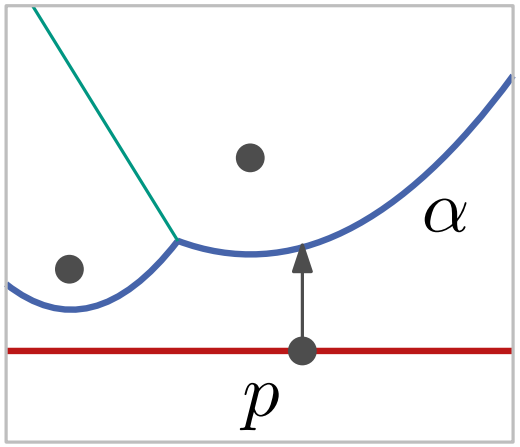
HandlePointEvent(Punkt p)



Punkt-Events behandeln

HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.

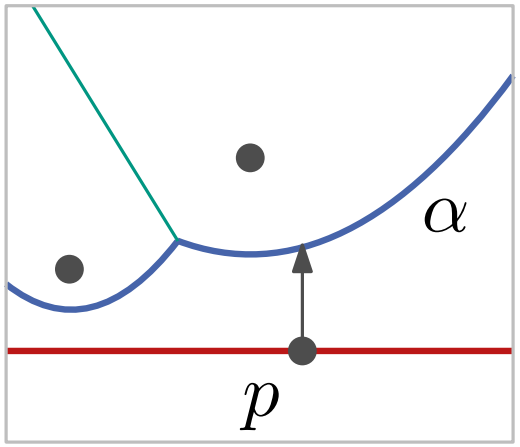


Punkt-Events behandeln

HandlePointEvent(Punkt p)

Wie?

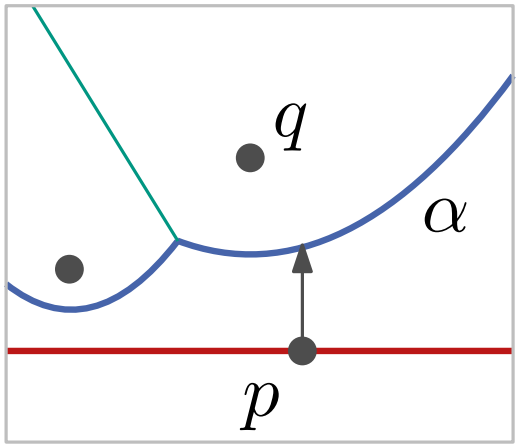
- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.



Punkt-Events behandeln

HandlePointEvent(Punkt p)

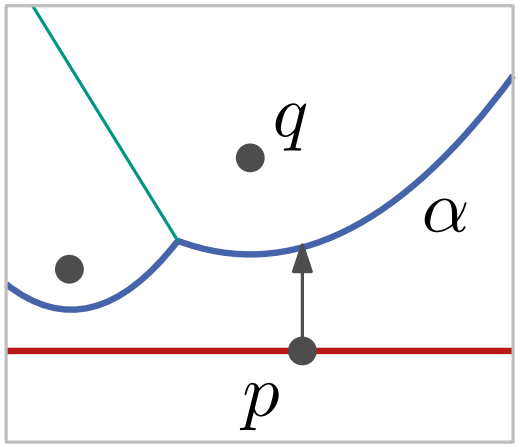
- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.



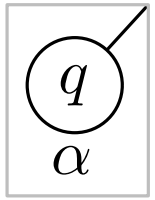
Punkt-Events behandeln

HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.



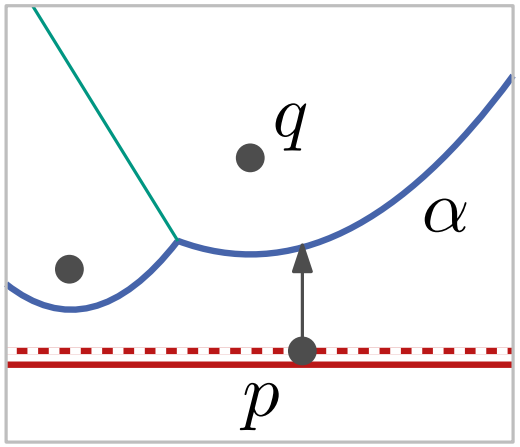
In $\mathcal{T}$:



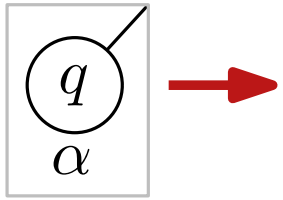
Punkt-Events behandeln

HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.



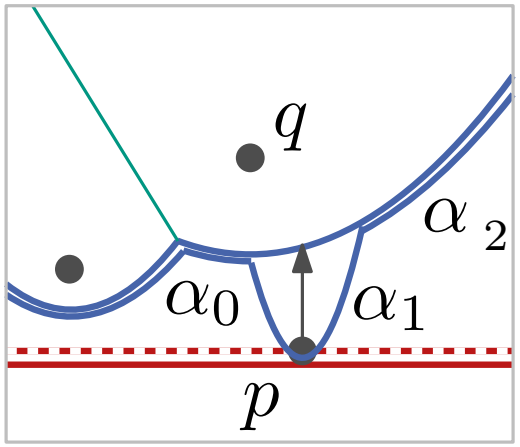
In $\mathcal{T}$:



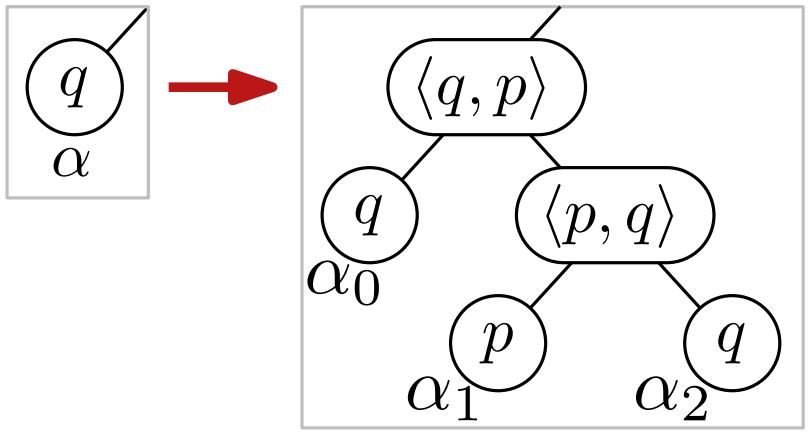
Punkt-Events behandeln

HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.



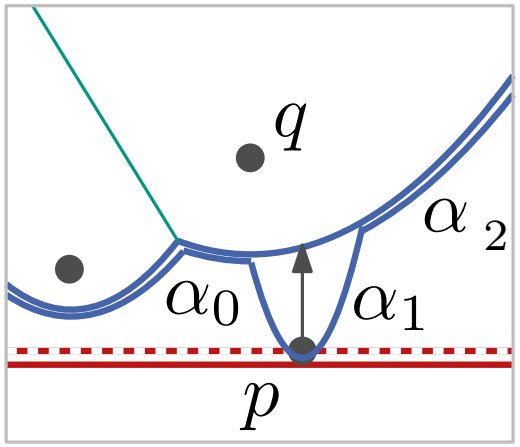
In $\mathcal{T}$:

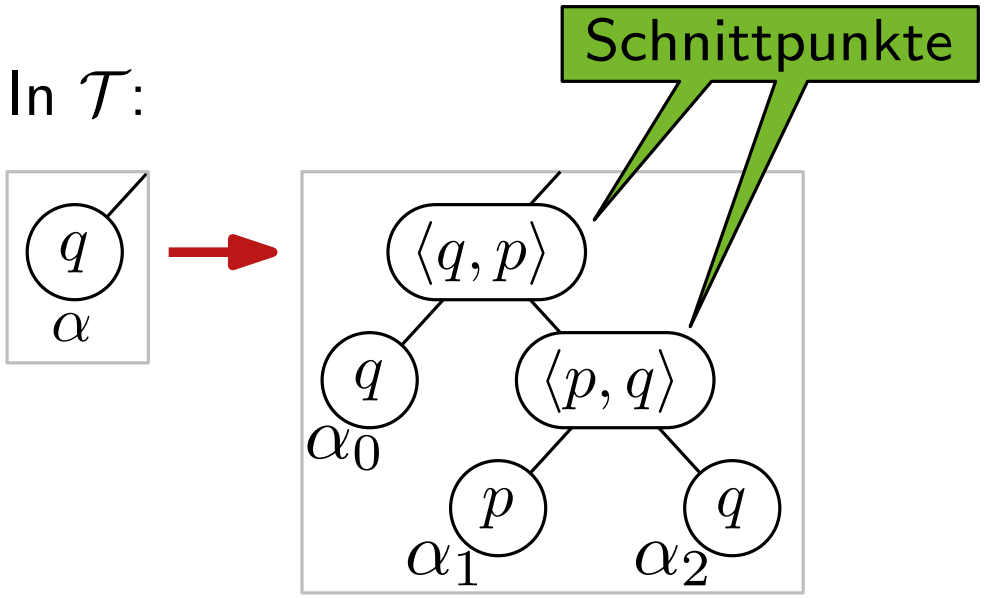


Punkt-Events behandeln

HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.

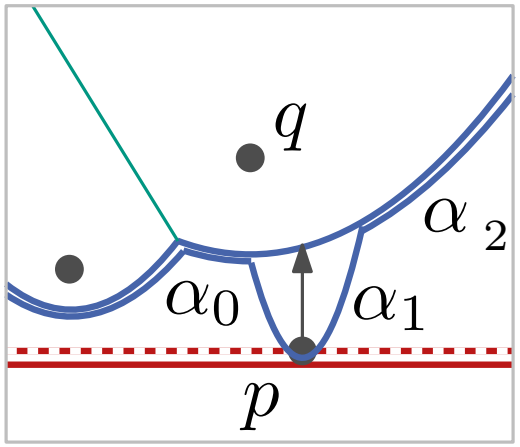


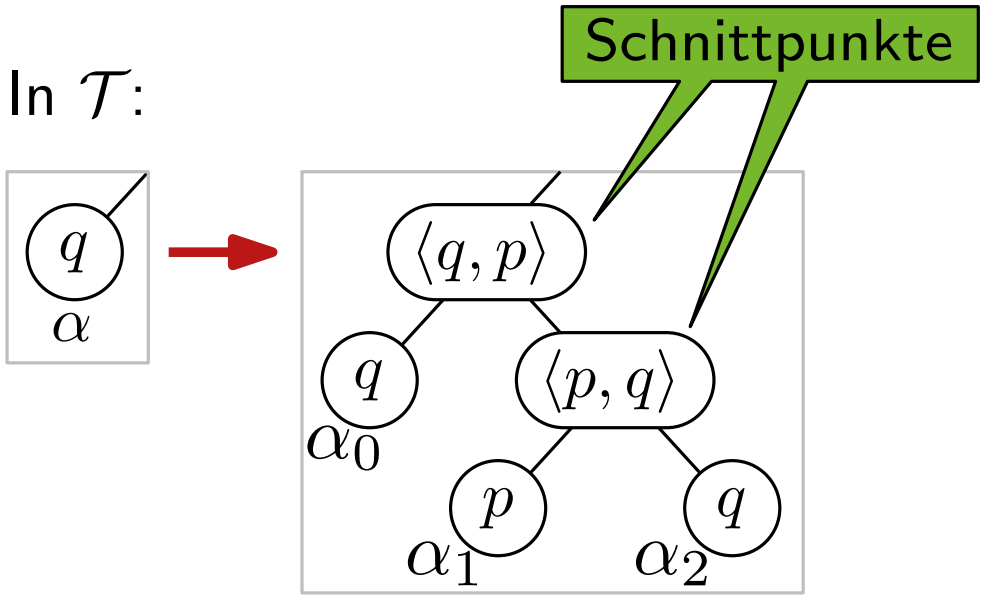


Punkt-Events behandeln

HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.
- Füge Kanten $\langle q, p \rangle$ und $\langle p, q \rangle$ in $\mathcal{D}$ ein.



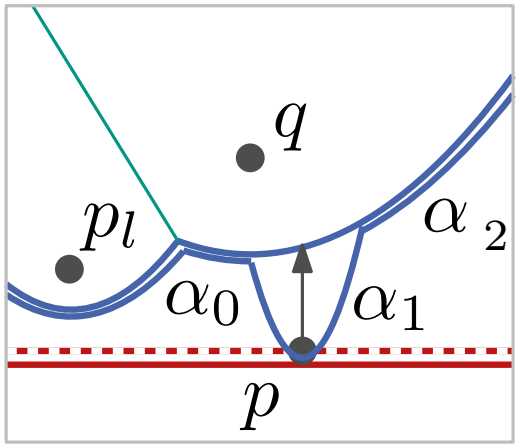


Punkt-Events behandeln

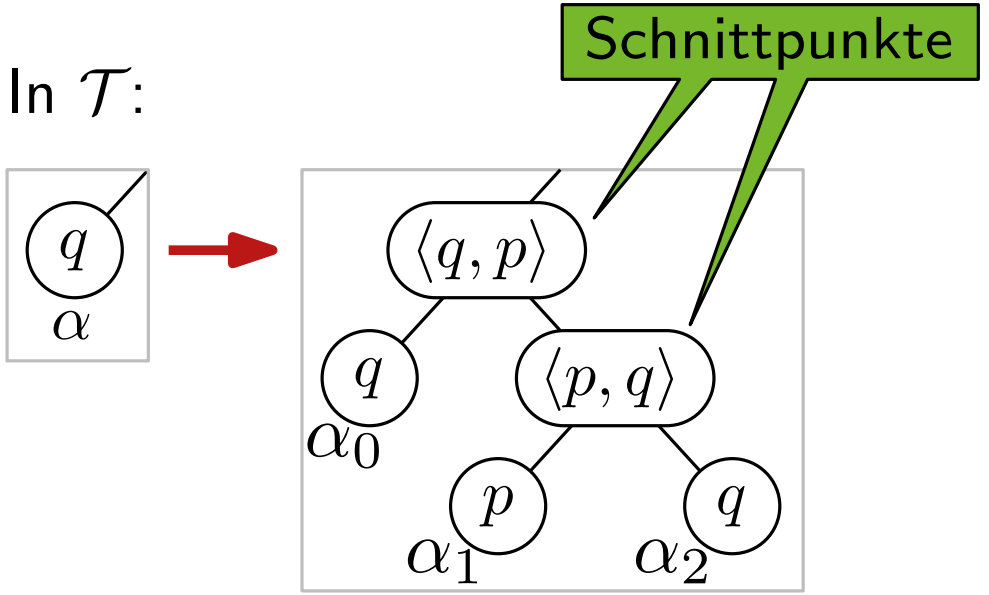
HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.
- Füge Kanten $\langle q, p\rangle$ und $\langle p, q\rangle$ in $\mathcal{D}$ ein.
- Prüfe $\langle p_l, q, p\rangle$ und $\langle q, p, p_r\rangle$ auf Kreis-Events.

Nachbarn auf β



In $\mathcal{T}$:

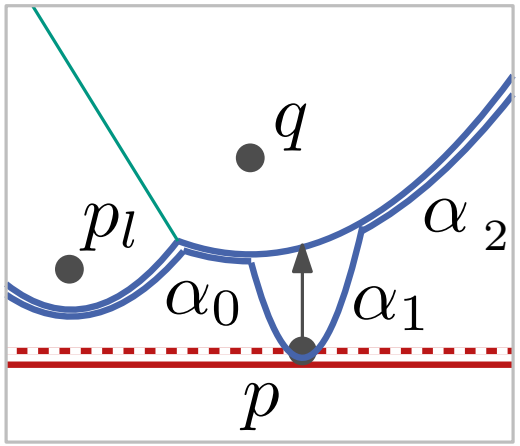


Punkt-Events behandeln

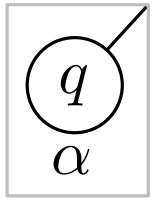
HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.
- Füge Kanten $\langle q, p\rangle$ und $\langle p, q\rangle$ in $\mathcal{D}$ ein.
- Prüfe $\langle p_l, q, p\rangle$ und $\langle q, p, p_r\rangle$ auf Kreis-Events.

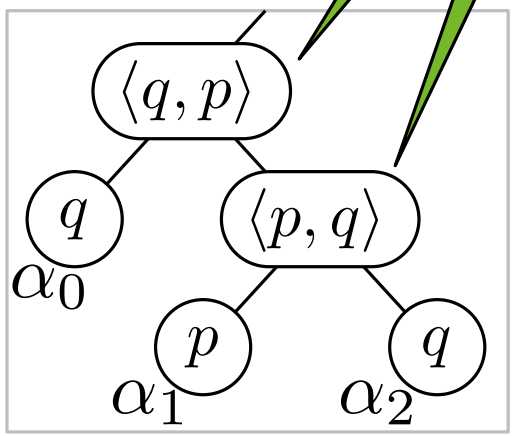
Nachbarn auf β



In $\mathcal{T}$:



Schnittpunkte



Laufzeit?

Punkt-Events behandeln

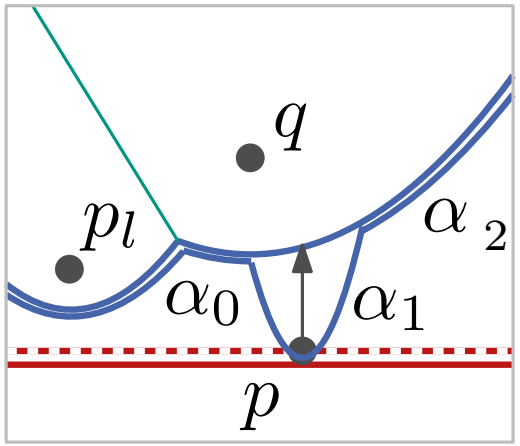
Karlsruhe Institute of Technology

$O(\log n)$

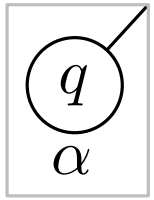
HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.
- Füge Kanten $\langle q, p\rangle$ und $\langle p, q\rangle$ in $\mathcal{D}$ ein.
- Prüfe $\langle p_l, q, p\rangle$ und $\langle q, p, p_r\rangle$ auf Kreis-Events.

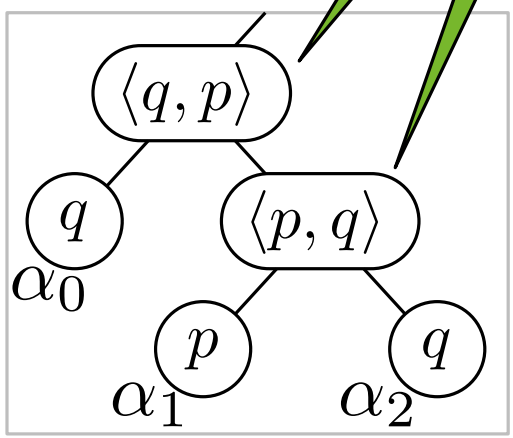
Nachbarn auf β



In $\mathcal{T}$:



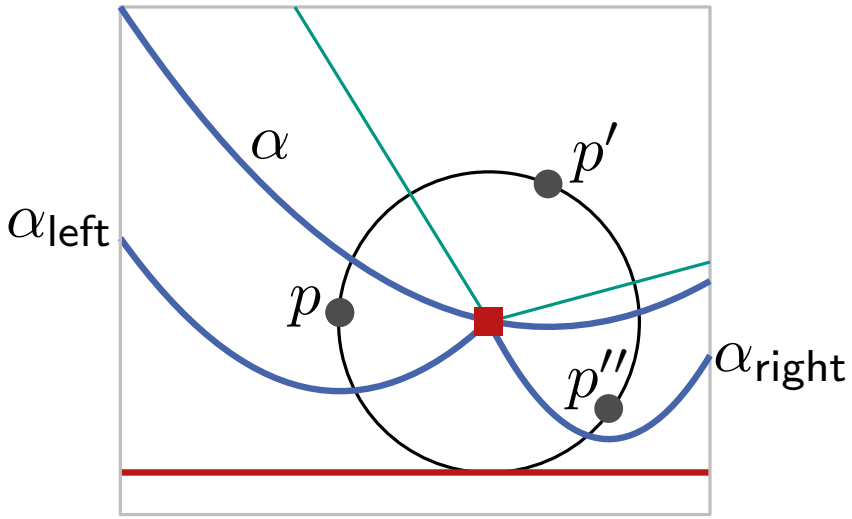
Schnittpunkte



Laufzeit?

Kreis-Events behandeln

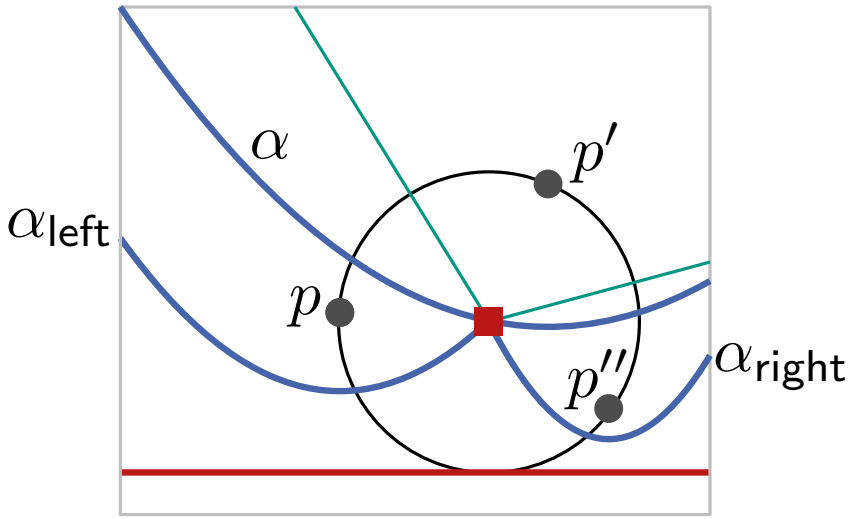
HandleCircleEvent(Bogen α)



Kreis-Events behandeln

HandleCircleEvent(Bogen α)

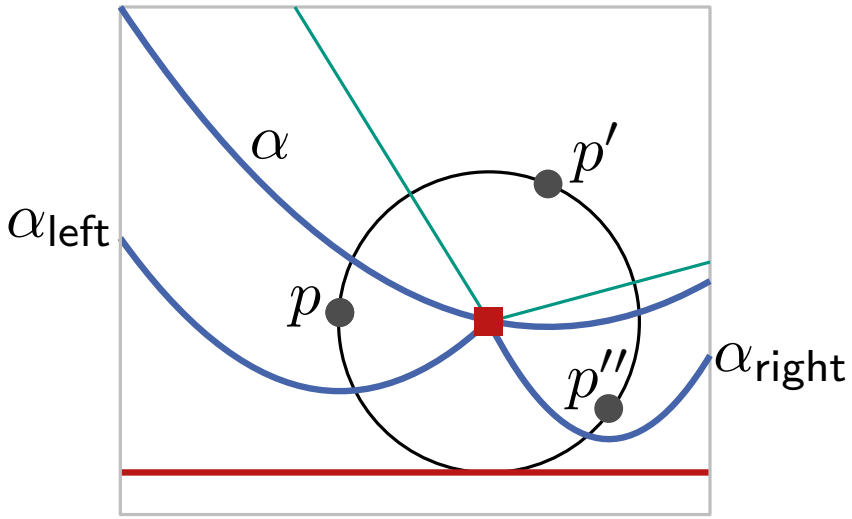
- $\mathcal{T}$.delete(α); Schnittpunkte in $\mathcal{T}$ updaten



Kreis-Events behandeln

HandleCircleEvent(Bogen α)

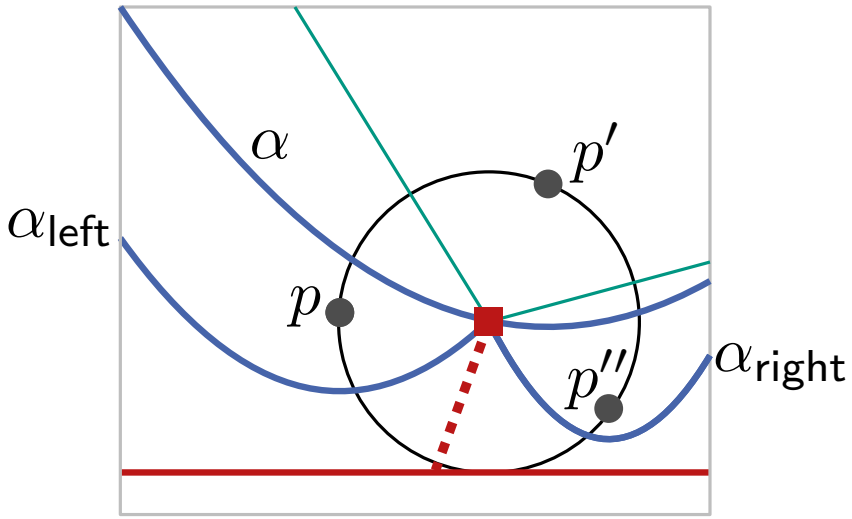
- $\mathcal{T}$.delete(α); Schnittpunkte in $\mathcal{T}$ updaten
- Entferne alle Kreis-Events zu α aus $\mathcal{Q}$.



Kreis-Events behandeln

HandleCircleEvent(Bogen α)

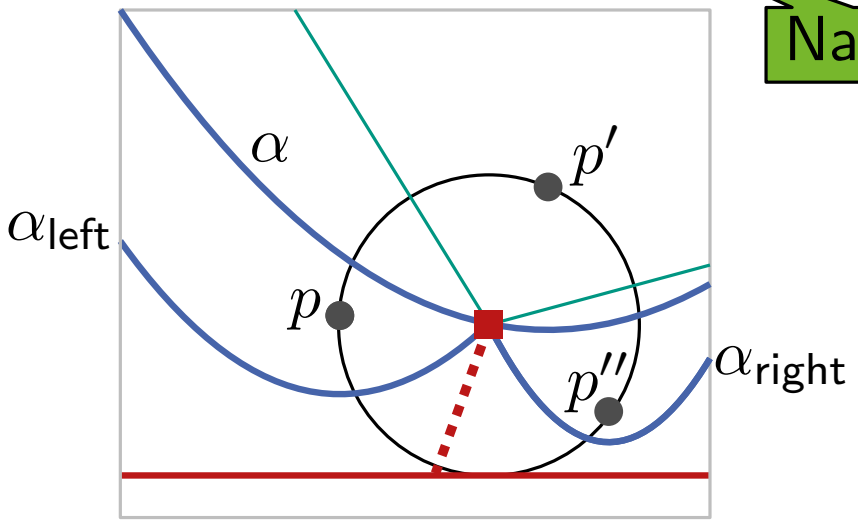
- $\mathcal{T}$.delete(α); Schnittpunkte in $\mathcal{T}$ updaten
- Entferne alle Kreis-Events zu α aus $\mathcal{Q}$.
- Füge Knoten $\mathcal{V}(\{p, p', p''\})$ und Kanten $\langle p, p'' \rangle$, $\langle p'', p \rangle$ in $\mathcal{D}$ ein.



Kreis-Events behandeln

HandleCircleEvent(Bogen α)

- $\mathcal{T}$.delete(α); Schnittpunkte in $\mathcal{T}$ updaten
- Entferne alle Kreis-Events zu α aus $\mathcal{Q}$.
- Füge Knoten $\mathcal{V}(\{p, p', p''\})$ und Kanten $\langle p, p'' \rangle$, $\langle p'', p \rangle$ in $\mathcal{D}$ ein.
- Füge potenzielle Kreis-Events $\langle p_l, p, p'' \rangle$ und $\langle p, p'', p_r \rangle$ in $\mathcal{Q}$ ein.

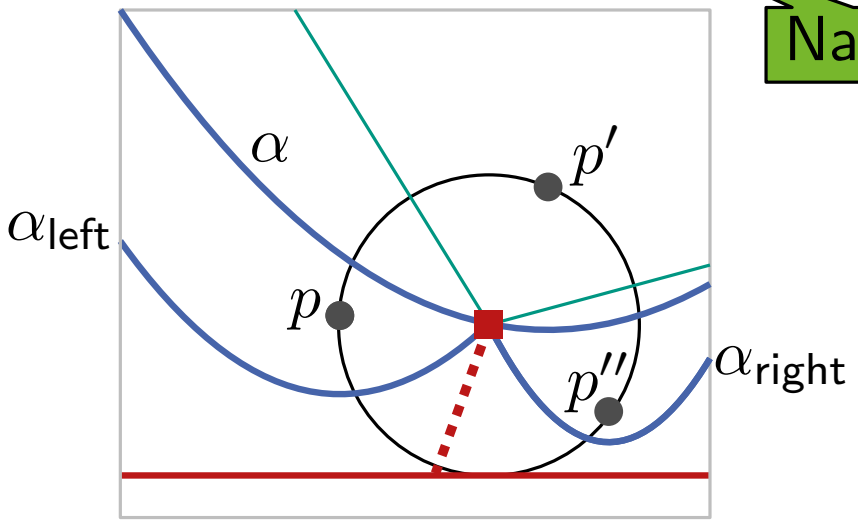


Kreis-Events behandeln

HandleCircleEvent(Bogen α)

- $\mathcal{T}$.delete(α); Schnittpunkte in $\mathcal{T}$ updaten
- Entferne alle Kreis-Events zu α aus $\mathcal{Q}$.
- Füge Knoten $\mathcal{V}(\{p, p', p''\})$ und Kanten $\langle p, p'' \rangle$, $\langle p'', p \rangle$ in $\mathcal{D}$ ein.
- Füge potenzielle Kreis-Events $\langle p_l, p, p'' \rangle$ und $\langle p, p'', p_r \rangle$ in $\mathcal{Q}$ ein.

Nachbarn auf β



Laufzeit?

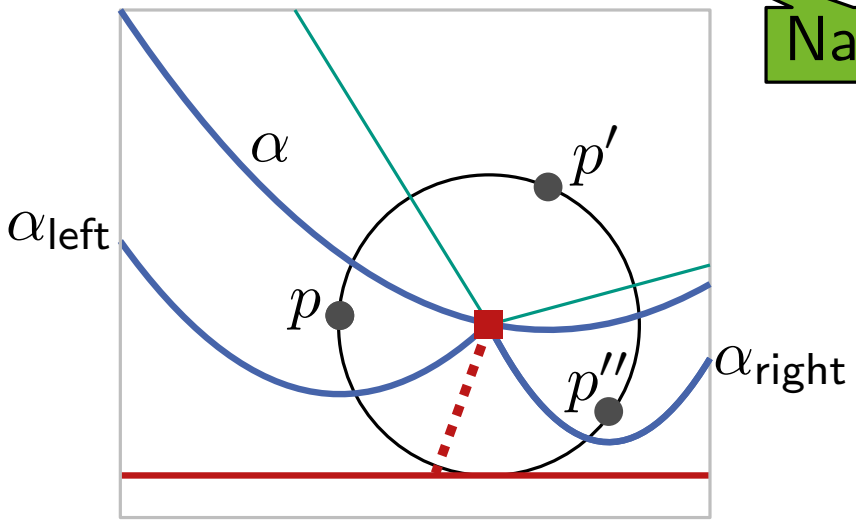
Kreis-Events behandeln

HandleCircleEvent(Bogen α)

$O(\log n)$

- $\mathcal{T}$.delete(α); Schnittpunkte in $\mathcal{T}$ updaten
- Entferne alle Kreis-Events zu α aus $\mathcal{Q}$.
- Füge Knoten $\mathcal{V}(\{p, p', p''\})$ und Kanten $\langle p, p'' \rangle$, $\langle p'', p \rangle$ in $\mathcal{D}$ ein.
- Füge potenzielle Kreis-Events $\langle p_l, p, p'' \rangle$ und $\langle p, p'', p_r \rangle$ in $\mathcal{Q}$ ein.

Nachbarn auf β



Laufzeit?

Fortune's Sweep Algorithmus

Satz 4: Für eine Menge P von n Punkten berechnet Fortune's Sweep Algorithmus das Voronoi-Diagramm $\mathrm{Vor}(P)$ in $O(n \log n)$ Zeit und $O(n)$ Platz.

Fortune's Sweep Algorithmus

Satz 4: Für eine Menge P von n Punkten berechnet Fortune's Sweep Algorithmus das Voronoi-Diagramm $\mathrm{Vor}(P)$ in $O(n \log n)$ Zeit und $O(n)$ Platz.

Beweisskizze:

- jedes Event benötigt $O(\log n)$ Zeit
- n Punkt-Events
- $\leq 2n - 5$ Kreis-Events ($=$ #Knoten von $\mathrm{Vor}(P)$)
- Fehlalarme erzeugen & löschen bereits inklusive

Fortune's Sweep Algorithmus

Satz 4: Für eine Menge P von n Punkten berechnet Fortune's Sweep Algorithmus das Voronoi-Diagramm $\mathrm{Vor}(P)$ in $O(n \log n)$ Zeit und $O(n)$ Platz.

Beweisskizze:

- jedes Event benötigt $O(\log n)$ Zeit
- n Punkt-Events
- $\leq 2n - 5$ Kreis-Events ($=$ #Knoten von $\mathrm{Vor}(P)$)
- Fehlalarme erzeugen & löschen bereits inklusive

Bemerkung:

degenerierte Eingaben diesmal unproblematisch:

- Kreis-Events vor Punkt-Events, sonst Reihenfolge beliebig
- Kreis-Events für $k \geq 4$ Punkte: Länge-0 Kanten und Knotenduplikate im Postprocessing entfernen
- zweideutiges Punktevent: beliebigen Bogen wählen

Diskussion

Karlsruhe Institute of Technology

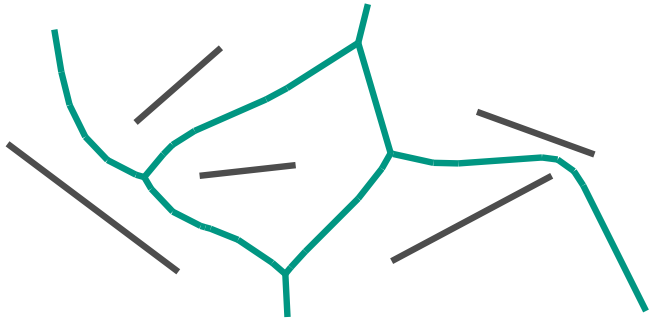
Gibt es weitere Varianten von Voronoi-Diagrammen?

Diskussion

Gibt es weitere Varianten von Voronoi-Diagrammen?

Ja! Beispielsweise kann der Algorithmus bei gleicher Laufzeit und Platzbedarf auch für Voronoi-Diagramme von Strecken modifiziert werden.

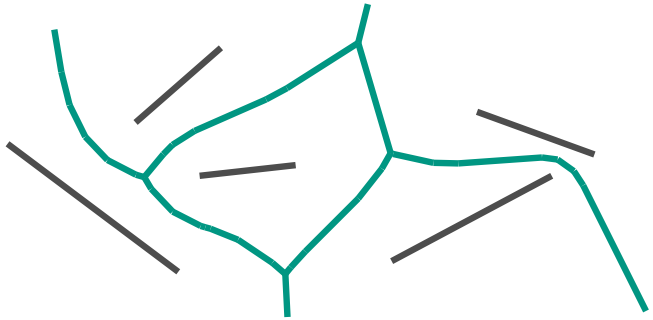
Auch andere Metriken wie L_p oder additiv/ multiplikativ gewichtete Voronoi-Diagramme sind möglich.



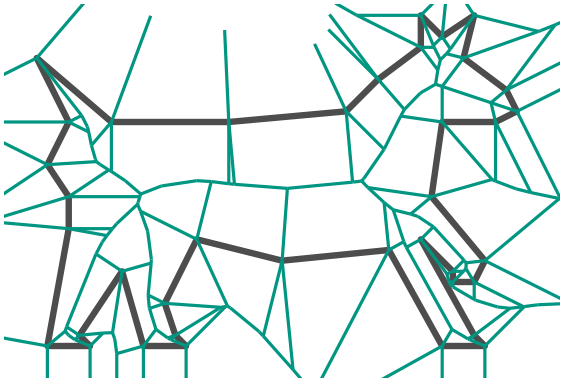
Diskussion

Gibt es weitere Varianten von Voronoi-Diagrammen?

Ja! Beispielsweise kann der Algorithmus bei gleicher Laufzeit und Platzbedarf auch für Voronoi-Diagramme von Strecken modifiziert werden.



Auch andere Metriken wie L_p oder additiv/ multiplikativ gewichtete Voronoi-Diagramme sind möglich.



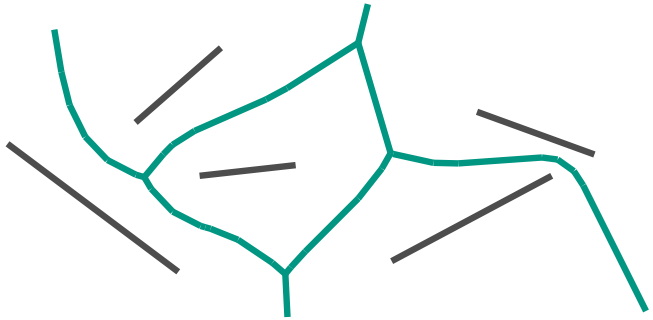
Voronoi-Diagramm für Polygone definieren die sog. Mittelachse, die z.B. in der Bildverarbeitung wichtig ist.

Auch farthest-point Voronoi-Diagramme sind möglich.

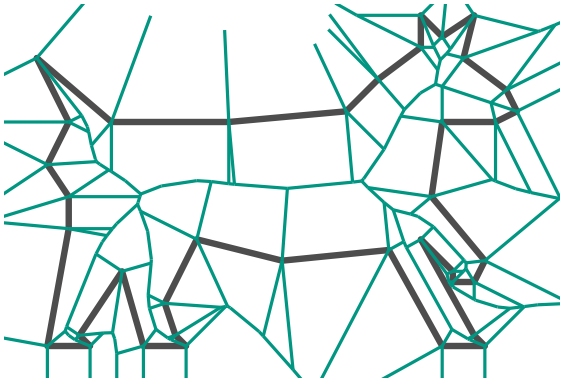
Diskussion

Gibt es weitere Varianten von Voronoi-Diagrammen?

Ja! Beispielsweise kann der Algorithmus bei gleicher Laufzeit und Platzbedarf auch für Voronoi-Diagramme von Strecken modifiziert werden.



Auch andere Metriken wie L_p oder additiv/ multiplikativ gewichtete Voronoi-Diagramme sind möglich.



Voronoi-Diagramm für Polygone definieren die sog. Mittelachse, die z.B. in der Bildverarbeitung wichtig ist.

Auch farthest-point Voronoi-Diagramme sind möglich.

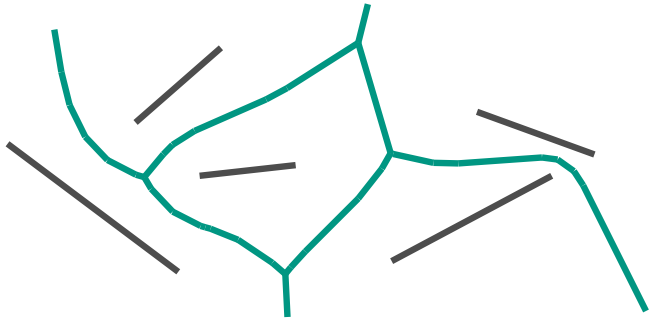
Was passiert in höheren Dimensionen?

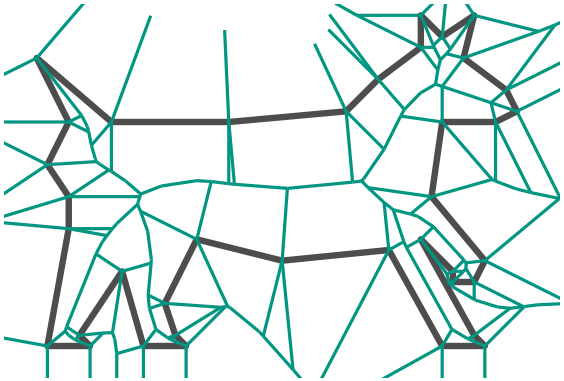
Diskussion

Gibt es weitere Varianten von Voronoi-Diagrammen?

Ja! Beispielsweise kann der Algorithmus bei gleicher Laufzeit und Platzbedarf auch für Voronoi-Diagramme von Strecken modifiziert werden.

Auch andere Metriken wie L_p oder additiv/ multiplikativ gewichtete Voronoi-Diagramme sind möglich.





Voronoi-Diagramm für Polygone definieren die sog. Mittelachse, die z.B. in der Bildverarbeitung wichtig ist.

Auch farthest-point Voronoi-Diagramme sind möglich.

Was passiert in höheren Dimensionen?

Die Komplexität von $\mathrm{Vor}(P)$ steigt auf $\Theta(n^{\lceil d/2 \rceil})$ und die Laufzeit auf $O(n \log n + n^{\lceil d/2 \rceil})$.