Karlsruhe Institute of Technology

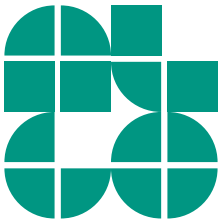
Vorlesung Algorithmische Geometrie

Einführung & Konvexe Hülle

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

Martin Nöllenburg

12.04.2011



AlgoGeom-Team

Dozent

- Martin Nöllenburg
- noellenburg@kit.edu
- Raum 319
- Sprechzeiten: Termin per Mail vereinbaren

AlgoGeom-Team

Dozent

- Martin Nöllenburg
- noellenburg@kit.edu
- Raum 319
- Sprechzeiten: Termin per Mail vereinbaren

Übungsleiter

- Andreas Gemsa
- gemsa@kit.edu
- Raum 322
- Sprechzeiten: Termin per Mail vereinbaren

AlgoGeom-Team

Dozent

- Martin Nöllenburg
- noellenburg@kit.edu
- Raum 319
- Sprechzeiten: Termin per Mail vereinbaren

Übungsleiter

- Andreas Gemsa
- gemsa@kit.edu
- Raum 322
- Sprechzeiten: Termin per Mail vereinbaren

Termine

- Vorlesung: Di 9:45 – 11:15 Uhr, Raum 131
- Übung: Do 10:15 – 11:00 Uhr, Raum 131 (ab 21.04.)

Organisatorisches

Webseite

`i11www.iti.kit.edu/teaching/sommer2011/compgeom`

- aktuelle Informationen
- Vorlesungsfolien
- Übungsblätter

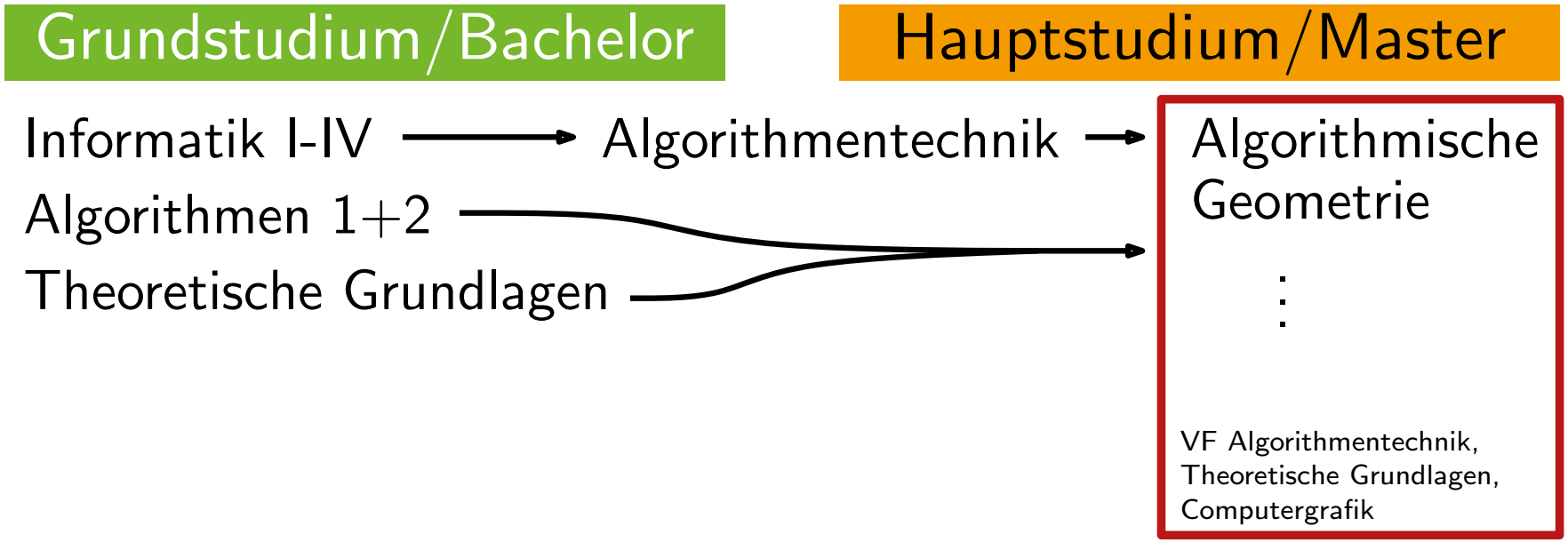
Organisatorisches

Webseite

`i11www.iti.kit.edu/teaching/sommer2011/compgeom`

- aktuelle Informationen
- Vorlesungsfolien
- Übungsblätter

Algorithmische Geometrie im Master-/Diplom-Studium



Algorithmische Geometrie

Karlsruhe Institute of Technology

Lernziele: Am Ende der Vorlesung können Sie

- Begriffe, Strukturen und Problemdefinitionen erklären
- behandelte Algorithmen ausführen, erklären und analysieren
- geeignete Algorithmen und Datenstrukturen auswählen und anpassen
- neue geometrische Probleme analysieren und eigene effiziente Lösungen entwerfen

Algorithmische Geometrie

Karlsruhe Institute of Technology

Lernziele: Am Ende der Vorlesung können Sie

- Begriffe, Strukturen und Problemdefinitionen erklären
- behandelte Algorithmen ausführen, erklären und analysieren
- geeignete Algorithmen und Datenstrukturen auswählen und anpassen
- neue geometrische Probleme analysieren und eigene effiziente Lösungen entwerfen

Vorkenntnisse: Algorithmik und elementare Geometrie

Algorithmische Geometrie

Karlsruhe Institute of Technology

Lernziele: Am Ende der Vorlesung können Sie

- Begriffe, Strukturen und Problemdefinitionen erklären
- behandelte Algorithmen ausführen, erklären und analysieren
- geeignete Algorithmen und Datenstrukturen auswählen und anpassen
- neue geometrische Probleme analysieren und eigene effiziente Lösungen entwerfen

Vorkenntnisse: Algorithmik und elementare Geometrie

Anforderungen/Prüfung:

- aktive Teilnahme an Vorlesung und Übung
- Bearbeiten der Übungsblätter
- mündliche Prüfung (einzeln oder im Vertiefungsfach)

Algorithmische Geometrie

Karlsruhe Institute of Technology

Lernziele: Am Ende der Vorlesung können Sie

- Begriffe, Strukturen und Problemdefinitionen erklären
- behandelte Algorithmen ausführen, erklären und analysieren
- geeignete Algorithmen und Datenstrukturen auswählen und anpassen
- neue geometrische Probleme analysieren und eigene effiziente Lösungen entwerfen

Vorkenntnisse: Algorithmik und elementare Geometrie

Anforderungen/Prüfung: 5LP = 150h

- aktive Teilnahme an Vorlesung und Übung — ca. 35h
- Bearbeiten der Übungsblätter — ca. 50h
- mündliche Prüfung (einzeln oder im Vertiefungsfach) — ca. 50h

Übungsbetrieb

Ausgabe Blatt 1
for Woche $i = 2 \ldots 14$ **do**
    if Tag == Dienstag **then**
        Ausgabe Blatt i
        Abgabe Blatt $(i-1)$
    if Tag == Donnerstag & !isFeiertag(Tag) **then**
        Besprechung Blatt $(i-1)$

- Bearbeitung der Aufgaben und Abgabe der Lösungen in Zweiergruppen erwünscht
- Übung wöchentlich 45 Minuten
- abweichende Regelung an Feiertagen
- Achtung: erste Übung am 21.04. in Raum −118

Werbung: Lehre am Lehrstuhl Algorithmik I

Karlsruhe Institute of Technology

Vertiefungsvorlesungen

- Algorithmen für planare Graphen
  (Di 13:00 Uhr, Do 14:00 Uhr)
- Algorithmen zur Routenplanung
  (Mo 14:00 Uhr, Do 11:30 Uhr)

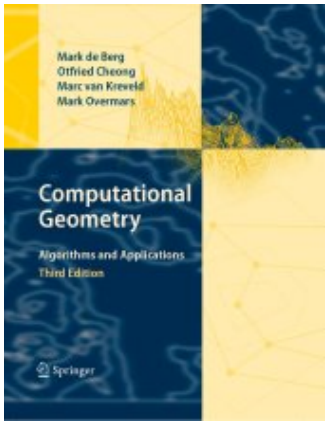
Seminare

- Geometrische Algorithmen in der Computergrafik
- Expandergraphen

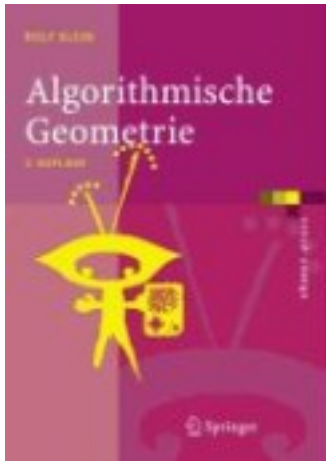
Praktikum

- Graphengeneratoren

Literatur



M. de Berg, O. Cheong, M. van Kreveld, M. Overmars:
Computational Geometry: Algorithms and Applications
Springer, 3. Auflage, 2008



Rolf Klein:
Algorithmische Geometrie
Springer, 2. Auflage, 2005

Beide Bücher (auch elektronisch) in der Bibliothek erhältlich!

Was ist algorithmische Geometrie?

Algorithmische Geometrie

Als **Algorithmische Geometrie** (engl. *Computational Geometry*) bezeichnet man ein Teilgebiet der Informatik, das sich mit der algorithmischen Lösung geometrisch formulierter Probleme beschäftigt. Ein zentrales Problem ist dabei die Speicherung und Verarbeitung geometrischer Daten. Im Gegensatz zur Bildbearbeitung, deren Grundelemente Bildpunkte (Pixel) sind, arbeitet die algorithmische Geometrie mit geometrischen Strukturelementen wie Punkten, Linien, Kreisen, Polygonen und Körpern.

Was ist algorithmische Geometrie?

Algorithmische Geometrie

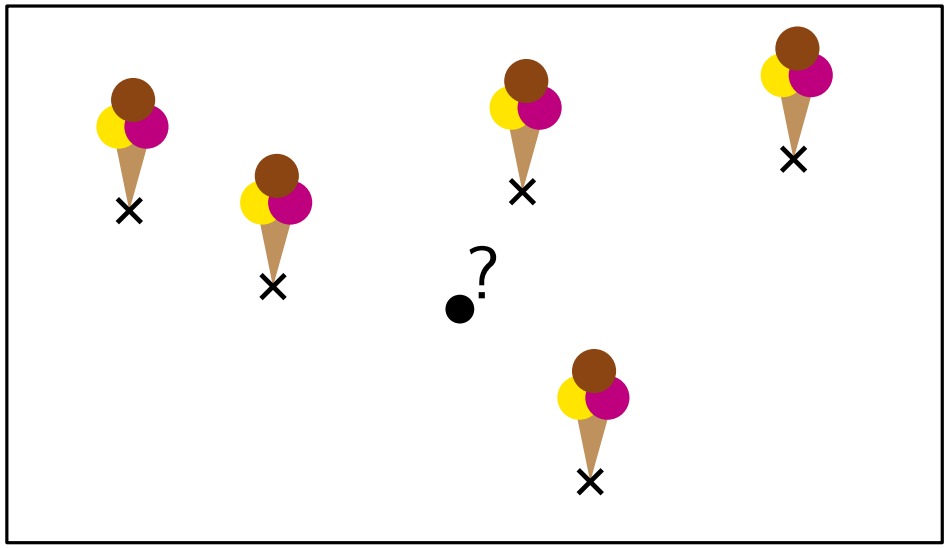
Als **Algorithmische Geometrie** (engl. *Computational Geometry*) bezeichnet man ein Teilgebiet der Informatik, das sich mit der algorithmischen Lösung geometrisch formulierter Probleme beschäftigt. Ein zentrales Problem ist dabei die Speicherung und Verarbeitung geometrischer Daten. Im Gegensatz zur Bildbearbeitung, deren Grundelemente Bildpunkte (Pixel) sind, arbeitet die algorithmische Geometrie mit geometrischen Strukturelementen wie Punkten, Linien, Kreisen, Polygonen und Körpern.

Wo treten Probleme der algorithmischen Geometrie auf?

- Computergrafik
- Visualisierung
- Geoinformationssysteme (GIS)
- Robotik
- ...

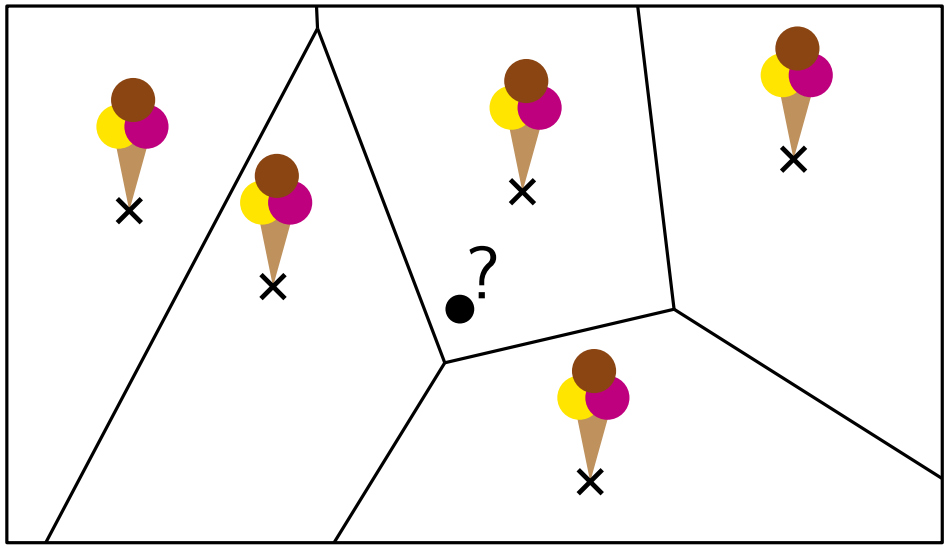
Beispiel 1

Ein sonniger Frühlingstag in Karlsruhe. Man kennt die Eisdielen in der Stadt und sucht die nächstgelegene. Wie kann man die nächste Eisdiele für jeden beliebigen Standort in einer Landkarte bestimmen?



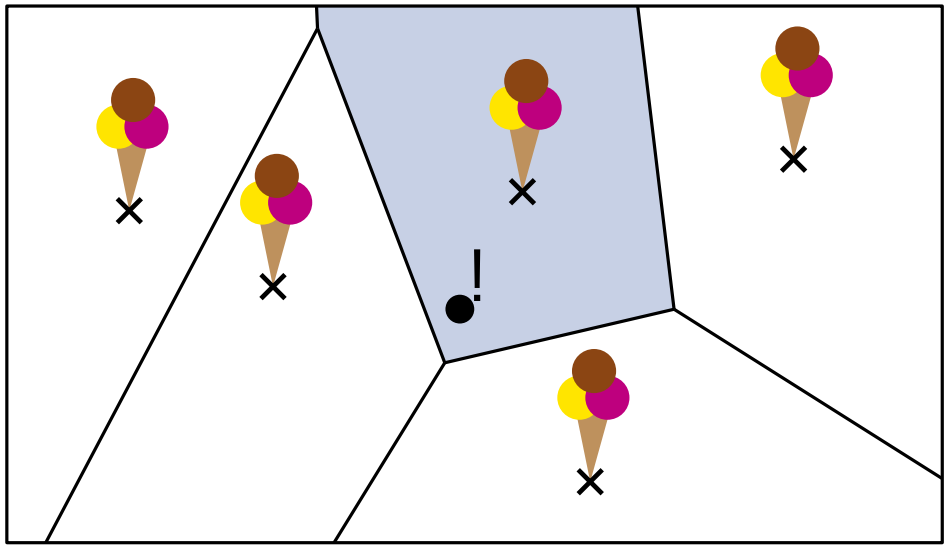
Beispiel 1

Ein sonniger Frühlingstag in Karlsruhe. Man kennt die Eisdielen in der Stadt und sucht die nächstgelegene. Wie kann man die nächste Eisdiele für jeden beliebigen Standort in einer Landkarte bestimmen?



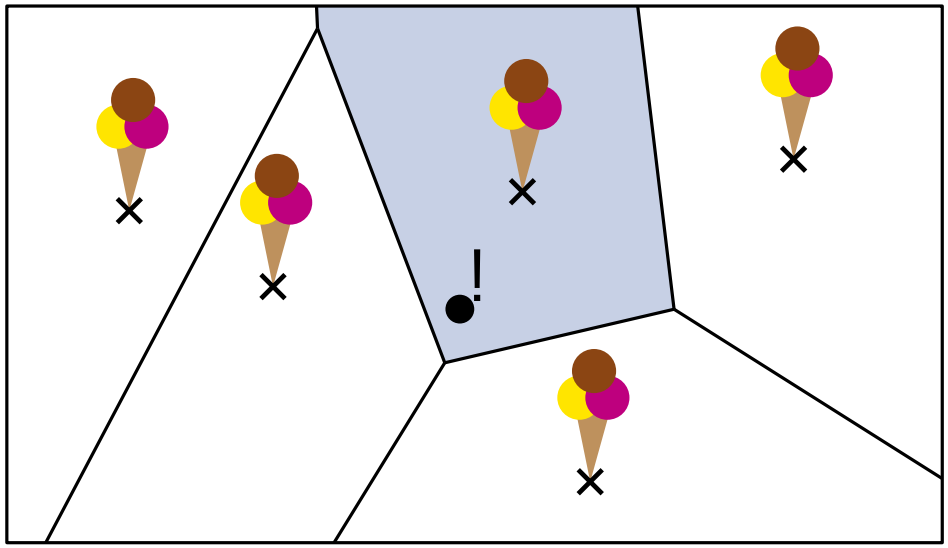
Beispiel 1

Ein sonniger Frühlingstag in Karlsruhe. Man kennt die Eisdielen in der Stadt und sucht die nächstgelegene. Wie kann man die nächste Eisdiele für jeden beliebigen Standort in einer Landkarte bestimmen?



Beispiel 1

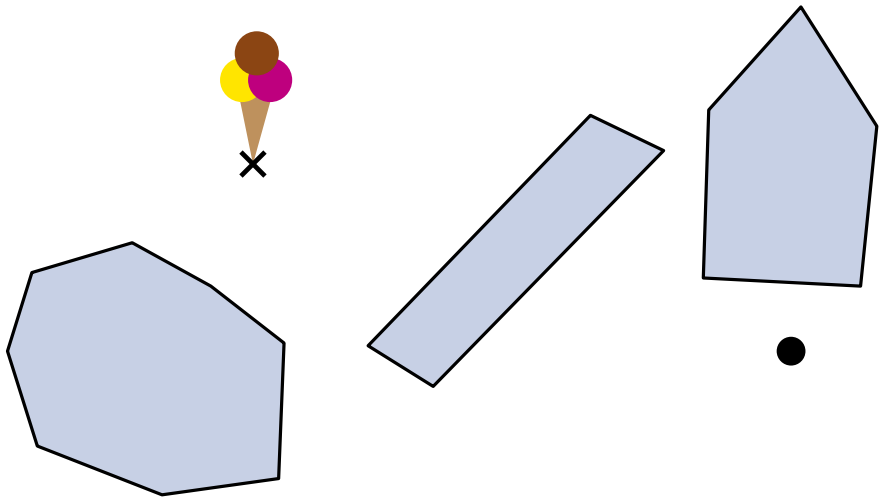
Ein sonniger Frühlingstag in Karlsruhe. Man kennt die Eisdielen in der Stadt und sucht die nächstgelegene. Wie kann man die nächste Eisdiele für jeden beliebigen Standort in einer Landkarte bestimmen?



Die Lösung ist eine *Unterteilung* der Ebene, die sich **Voronoi-Diagramm** nennt.
Anwendungen z.B. in der Standortplanung, Berechnung von Einzugsgebieten etc.

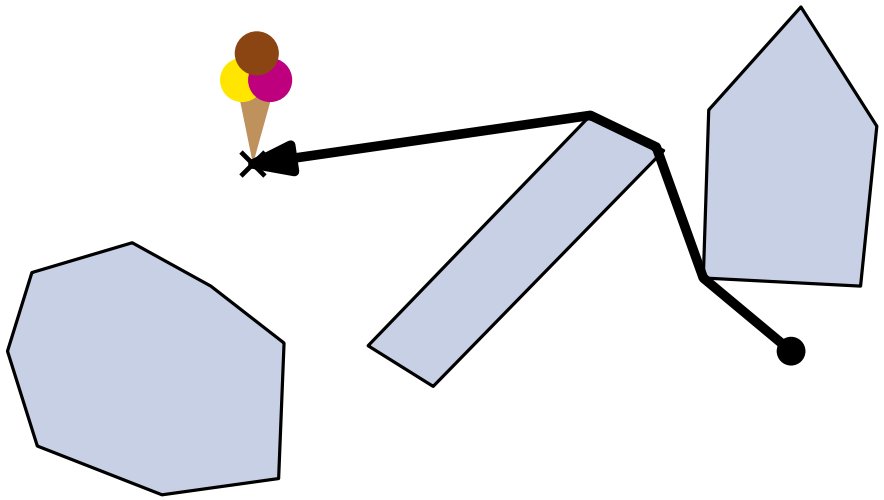
Beispiel 2

Wir wollen unseren Roboter losschicken um ein Eis zu kaufen. Wie kommt der Roboter zum Ziel ohne gegen Häuser, Parkbänke und Bäume zu laufen?



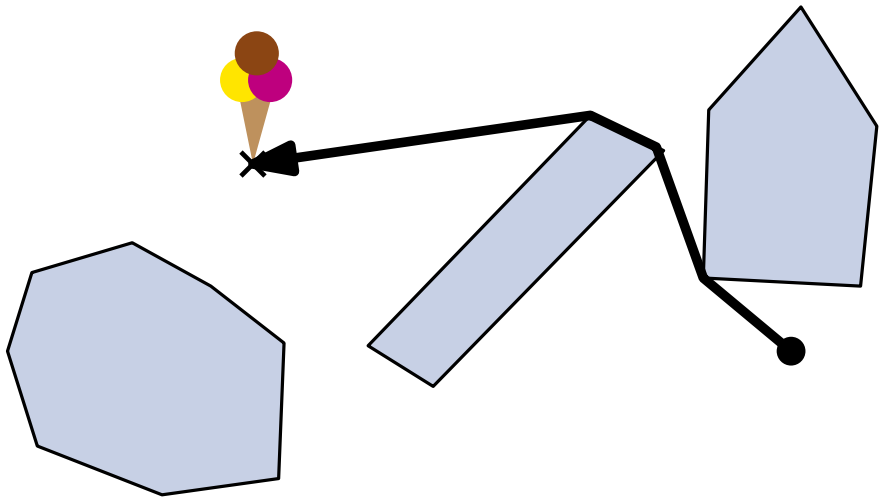
Beispiel 2

Wir wollen unseren Roboter losschicken um ein Eis zu kaufen. Wie kommt der Roboter zum Ziel ohne gegen Häuser, Parkbänke und Bäume zu laufen?



Beispiel 2

Wir wollen unseren Roboter losschicken um ein Eis zu kaufen. Wie kommt der Roboter zum Ziel ohne gegen Häuser, Parkbänke und Bäume zu laufen?

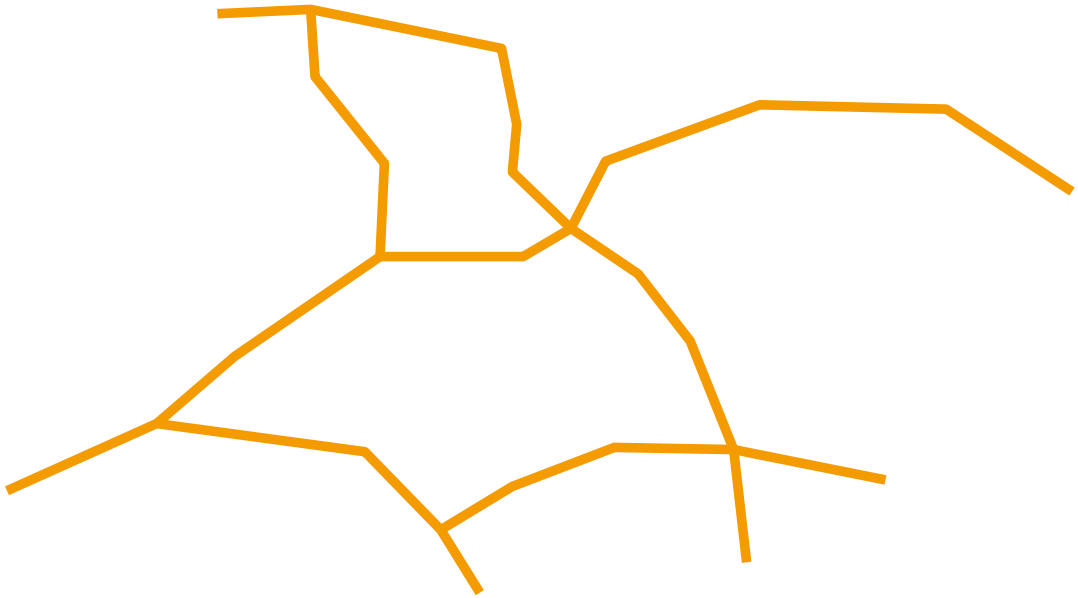


Bewegungsplanungsproblem in der Robotik:
Gegeben eine Menge von Hindernissen, einen Start- und einen Zielpunkt, finde einen kollisionsfreien kürzesten Weg zum Ziel.

Beispiel 3

Karten in Geoinformationssystemen bestehen aus mehreren Ebenen, z.B. Straßen, Gewässer, Grenzen usw. Überlagert man mehrere Ebenen, stellt sich die Frage nach den Schnittpunkten.

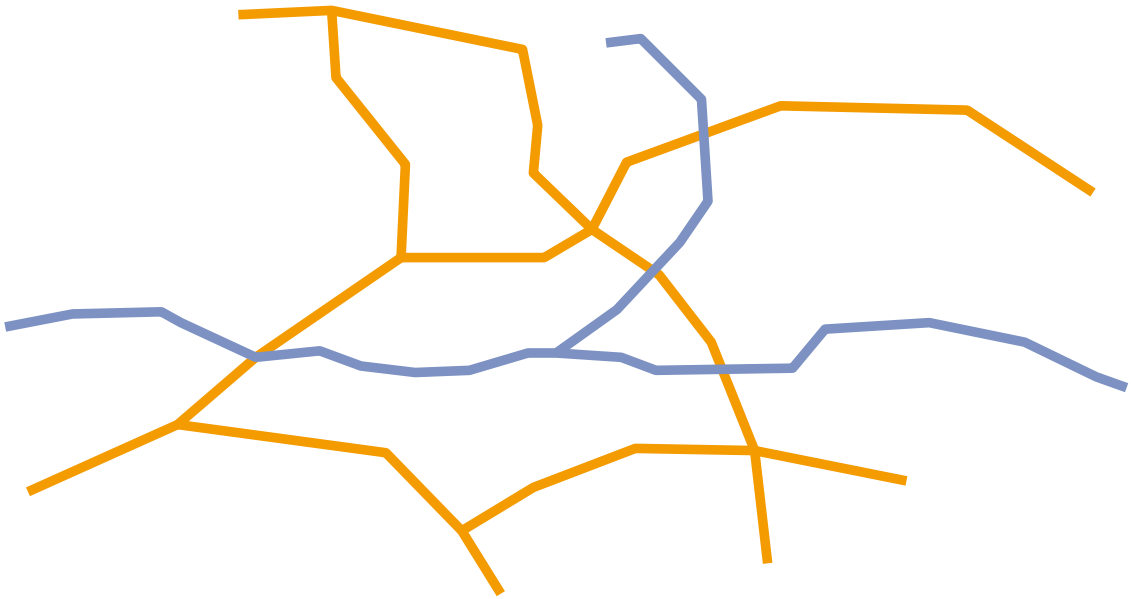
Modelliert man z.B. Straßen und Flüsse als Menge von Strecken und fragt nach den Brücken, muss man alle Schnittpunkte der beiden Streckenmengen finden.



Beispiel 3

Karten in Geoinformationssystemen bestehen aus mehreren Ebenen, z.B. Straßen, Gewässer, Grenzen usw. Überlagert man mehrere Ebenen, stellt sich die Frage nach den Schnittpunkten.

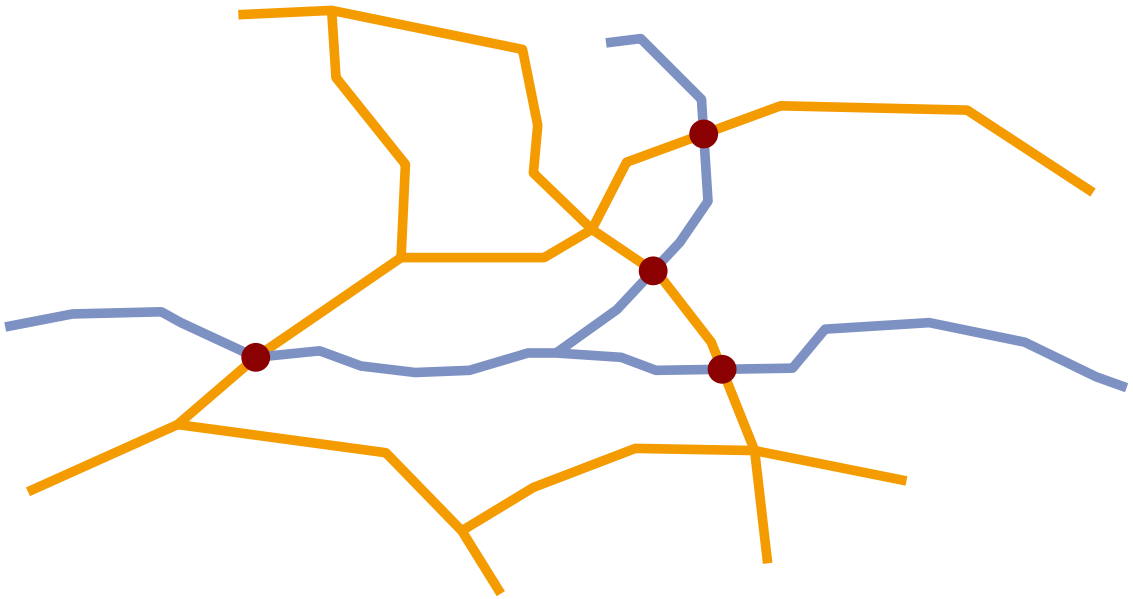
Modelliert man z.B. Straßen und Flüsse als Menge von Strecken und fragt nach den Brücken, muss man alle Schnittpunkte der beiden Streckenmengen finden.



Beispiel 3

Karten in Geoinformationssystemen bestehen aus mehreren Ebenen, z.B. Straßen, Gewässer, Grenzen usw. Überlagert man mehrere Ebenen, stellt sich die Frage nach den Schnittpunkten.

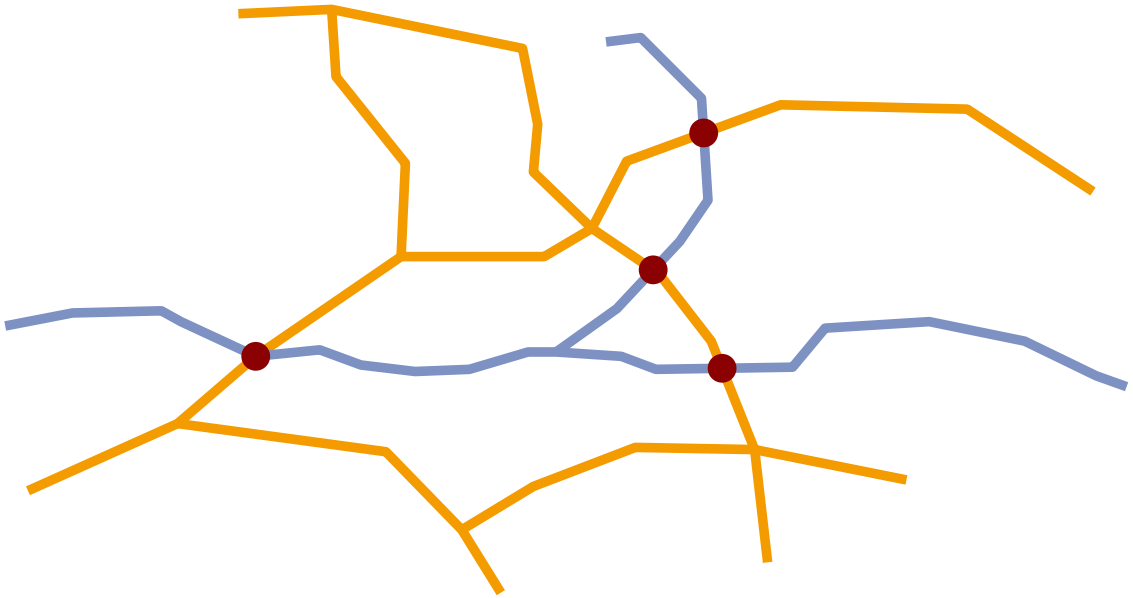
Modelliert man z.B. Straßen und Flüsse als Menge von Strecken und fragt nach den Brücken, muss man alle Schnittpunkte der beiden Streckenmengen finden.



Beispiel 3

Karten in Geoinformationssystemen bestehen aus mehreren Ebenen, z.B. Straßen, Gewässer, Grenzen usw. Überlagert man mehrere Ebenen, stellt sich die Frage nach den Schnittpunkten.

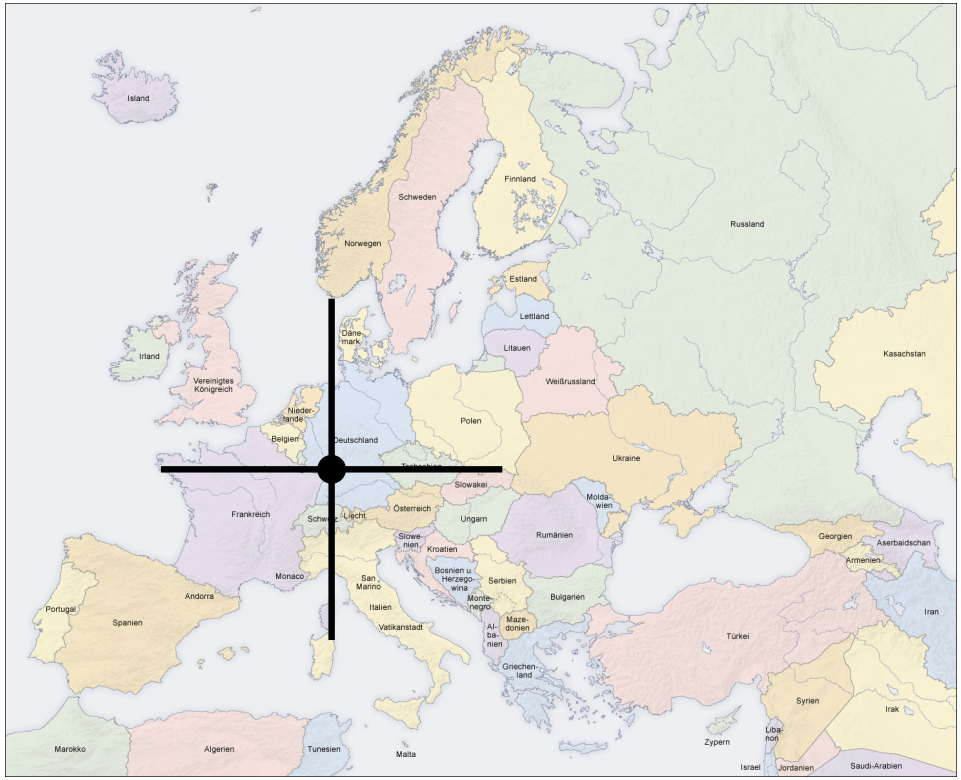
Modelliert man z.B. Straßen und Flüsse als Menge von Strecken und fragt nach den Brücken, muss man alle Schnittpunkte der beiden Streckenmengen finden.



Alle Kantenpaare testen ist zu langsam. Wie findet man schnell alle Schnittpunkte?

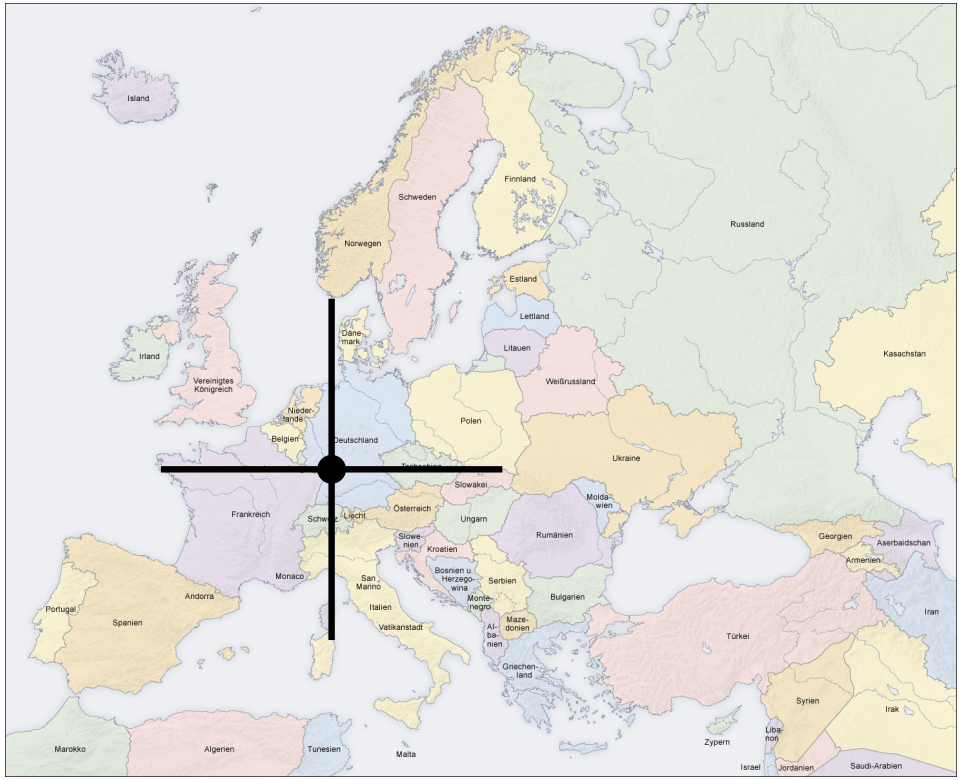
Beispiel 4

Gegeben eine Landkarte und einen Anfragepunkt q (z.B. Mausklick). Bestimme das Land, in dem q liegt.



Beispiel 4

Gegeben eine Landkarte und einen Anfragepunkt q (z.B. Mausklick). Bestimme das Land, in dem q liegt.



Ziel:

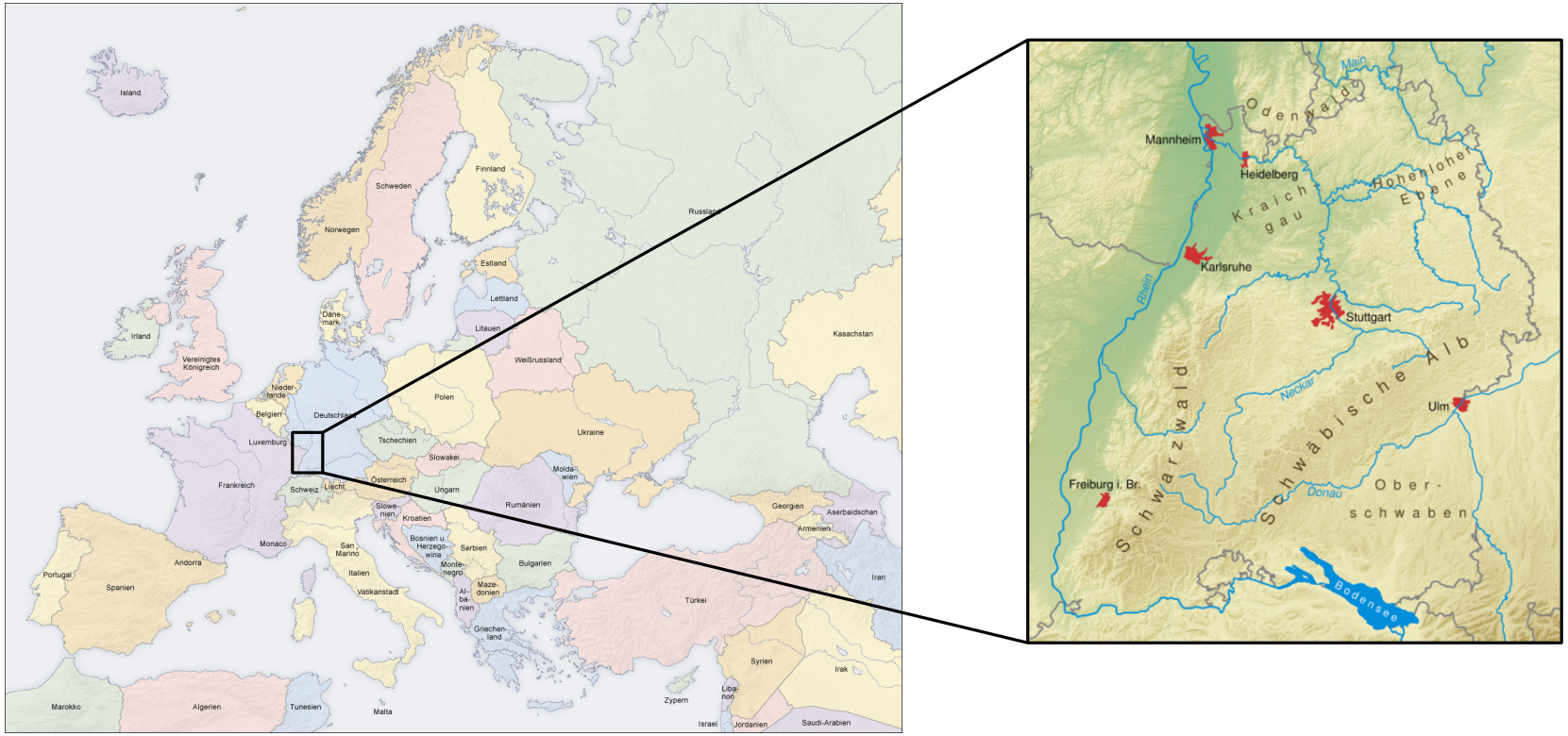
Datenstruktur zur schnellen Beantwortung solcher Anfragen

Beispiel 5

Ein Navigationssystem soll einen aktuellen Kartenausschnitt anzeigen. Wie wählt man effizient die benötigten Daten aus?

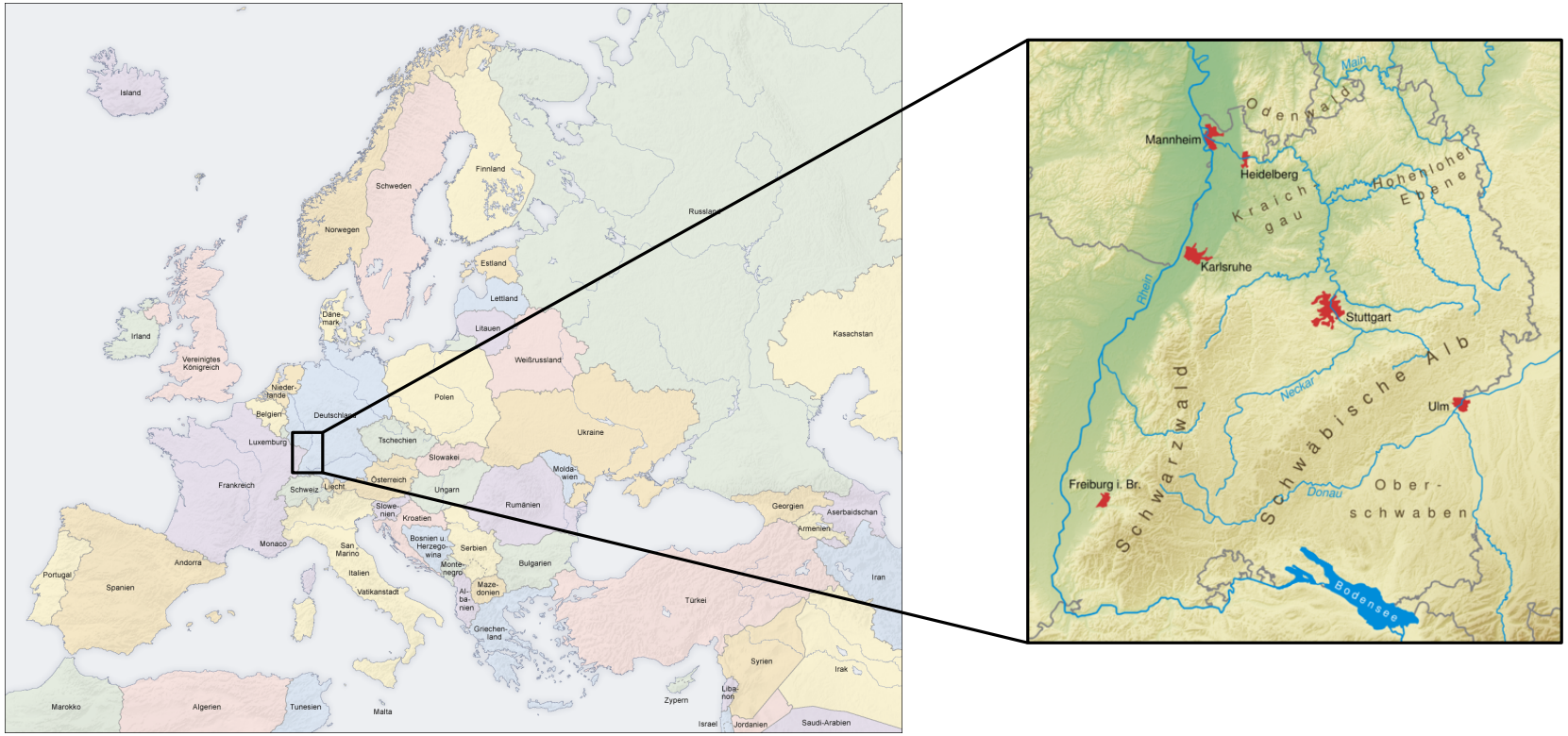
Beispiel 5

Ein Navigationssystem soll einen aktuellen Kartenausschnitt anzeigen. Wie wählt man effizient die benötigten Daten aus?



Beispiel 5

Ein Navigationssystem soll einen aktuellen Kartenausschnitt anzeigen. Wie wählt man effizient die benötigten Daten aus?



Prüfung für jedes einzelne Kartenobjekt ist unrealistisch.

Ziel: Datenstruktur zur schnellen Beantwortung von Bereichsanfragen

Mischungsverhältnisse

Karlsruhe Institute of Technology

Gegeben...

Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$

Mischungsverhältnisse

Gegeben...

Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$

mische folgende Mixturen

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

aus s_1, s_2?

Mischungsverhältnisse

Gegeben...

Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$

mische folgende Mixturen

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

aus s_1, s_2?

q_1: Ja! Verhältnis 1:1

q_2: Nein!

Mischungsverhältnisse

Gegeben...

Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

mische folgende Mixturen

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

aus s_1, s_2, s_3?

Mischungsverhältnisse

Gegeben...

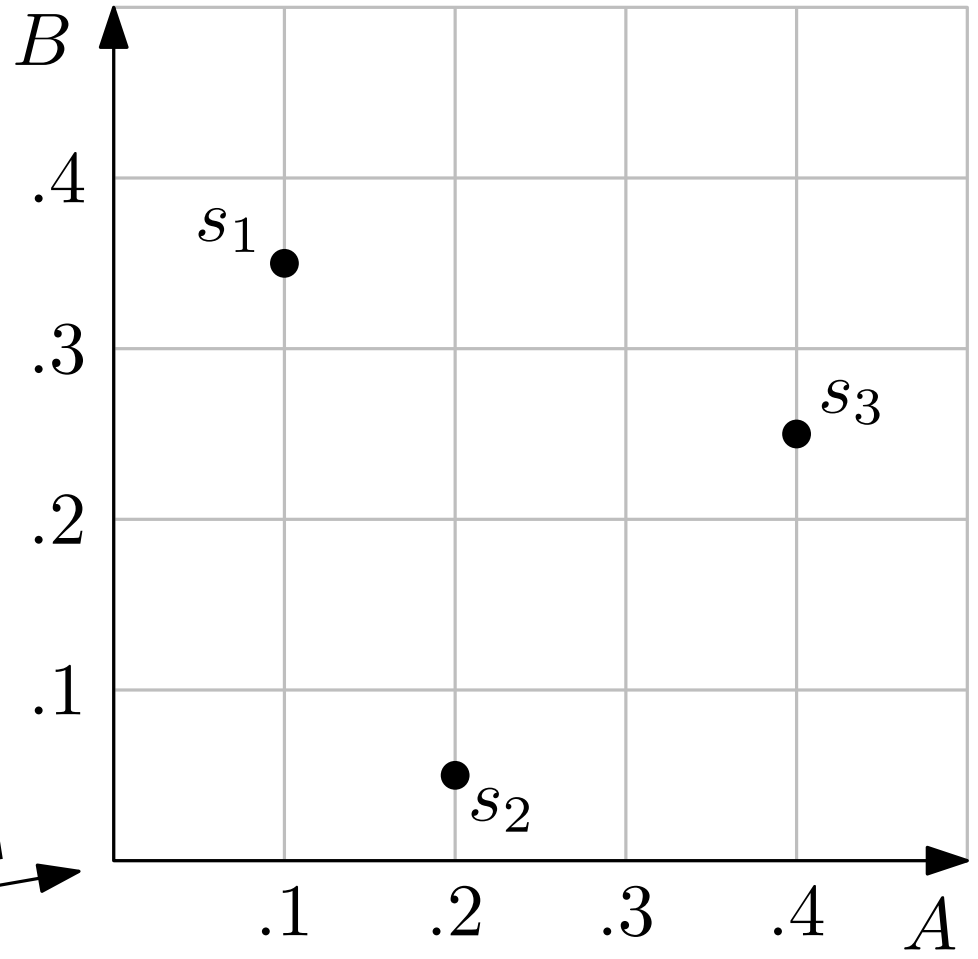
Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

mische folgende Mixturen

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

aus s_1, s_2, s_3?

geom. Interpretation



Mischungsverhältnisse

Gegeben...

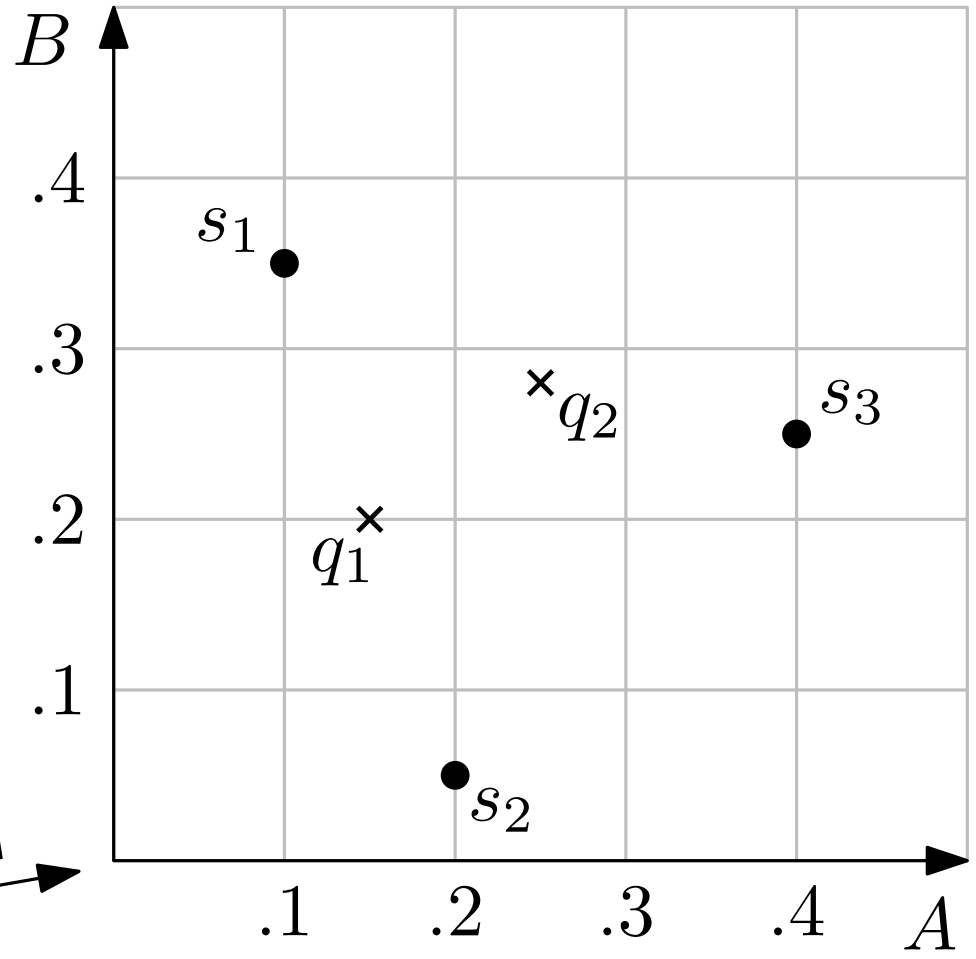
Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

mische folgende Mixturen

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

aus s_1, s_2, s_3?

geom. Interpretation



Mischungsverhältnisse

Gegeben...

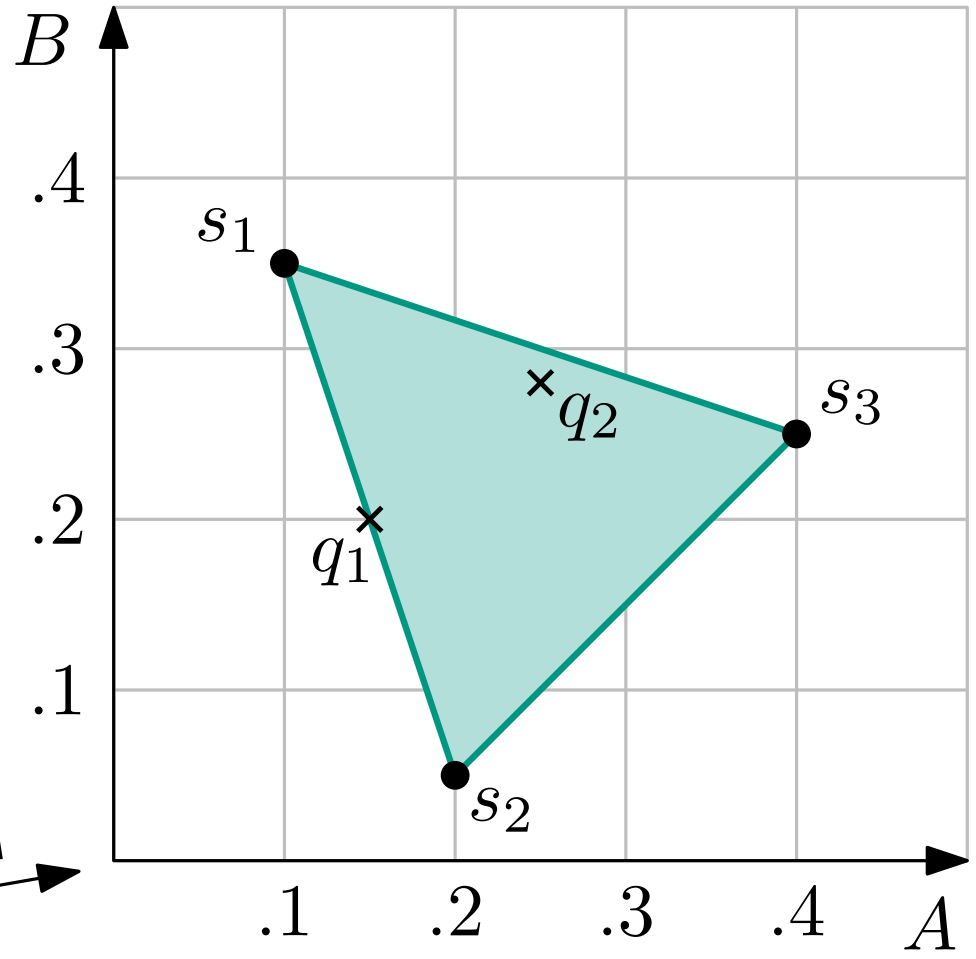
Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

mische folgende Mixturen

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

aus s_1, s_2, s_3?

geom. Interpretation



Mischungsverhältnisse

Gegeben...

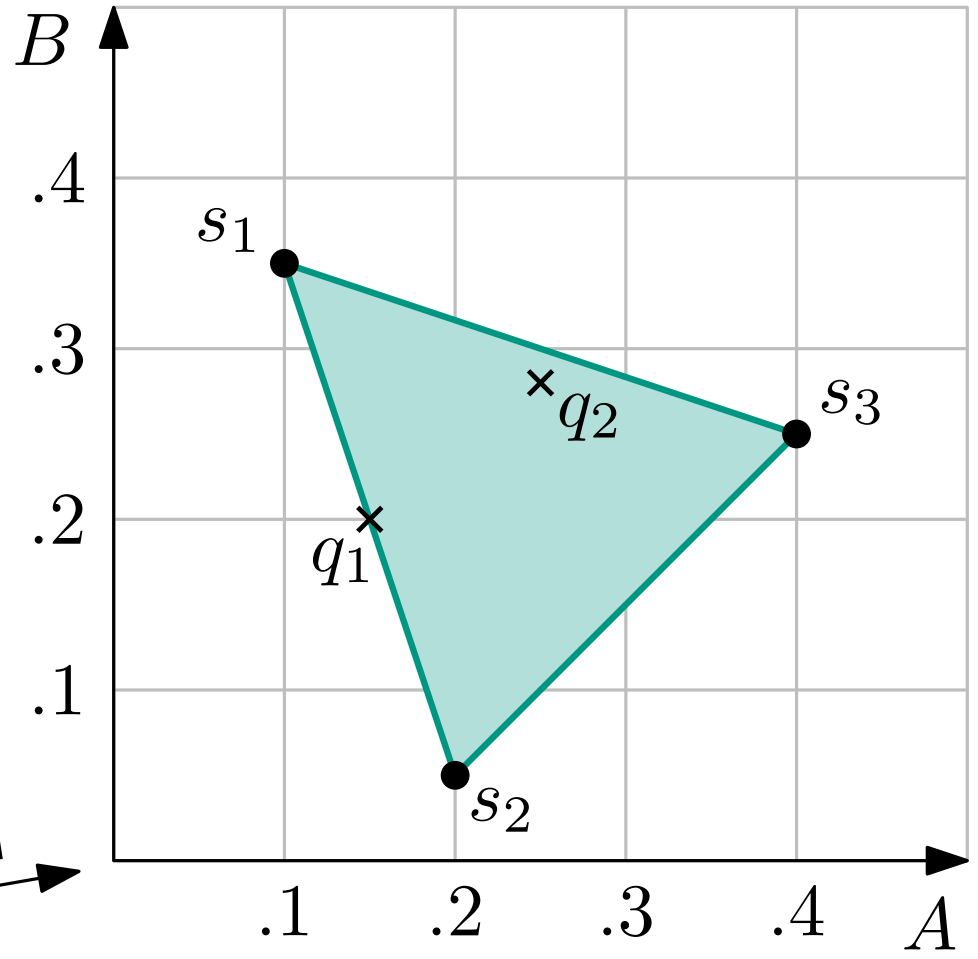
Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

mische folgende Mixturen

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

aus s_1, s_2, s_3?

geom. Interpretation



Beob: Gegeben eine Menge $S \subset \mathbb{R}^2$ von Mixturen, kann man eine weitere Mixtur $q \in \mathbb{R}^2$ aus S mischen $\Leftrightarrow$

Mischungsverhältnisse

Gegeben...

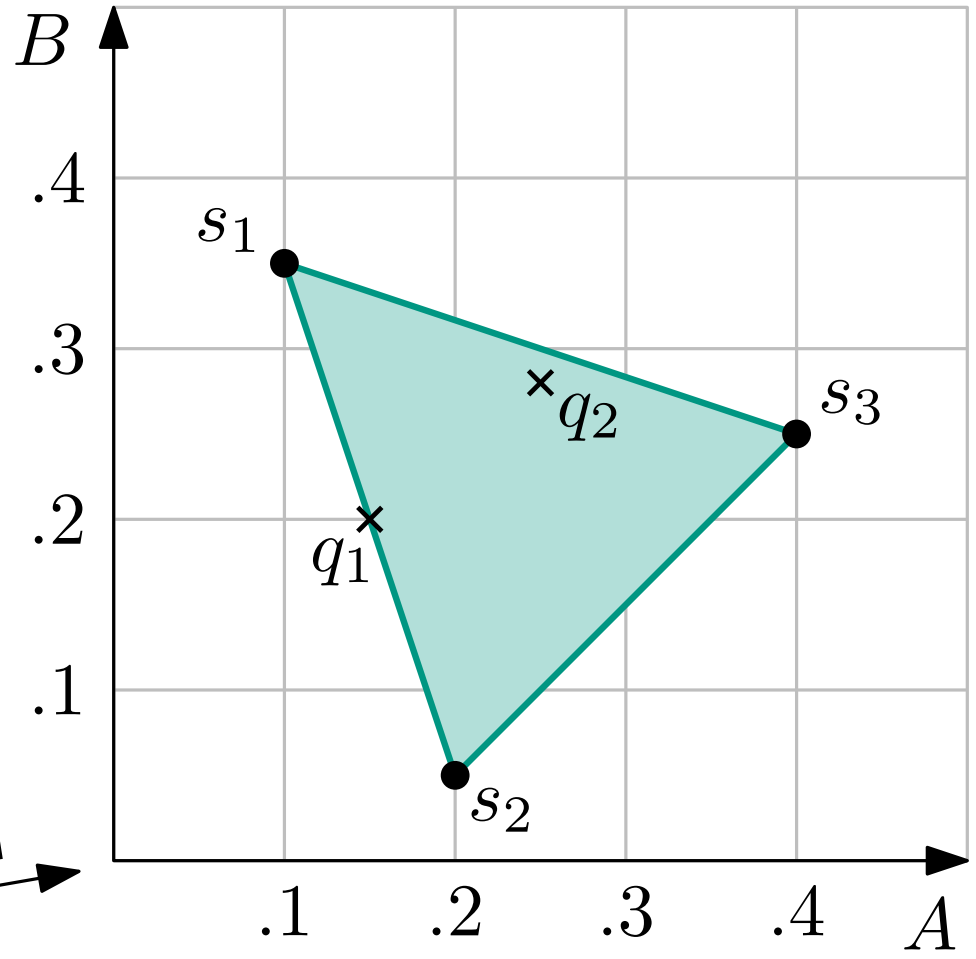
Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

mische folgende Mixturen

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

aus s_1, s_2, s_3?

geom. Interpretation



Beob: Gegeben eine Menge $S \subset \mathbb{R}^2$ von Mixturen, kann man eine weitere Mixtur $q \in \mathbb{R}^2$ aus S mischen $\Leftrightarrow$ $q \in CH(S)$.

Mischungsverhältnisse

Gegeben...

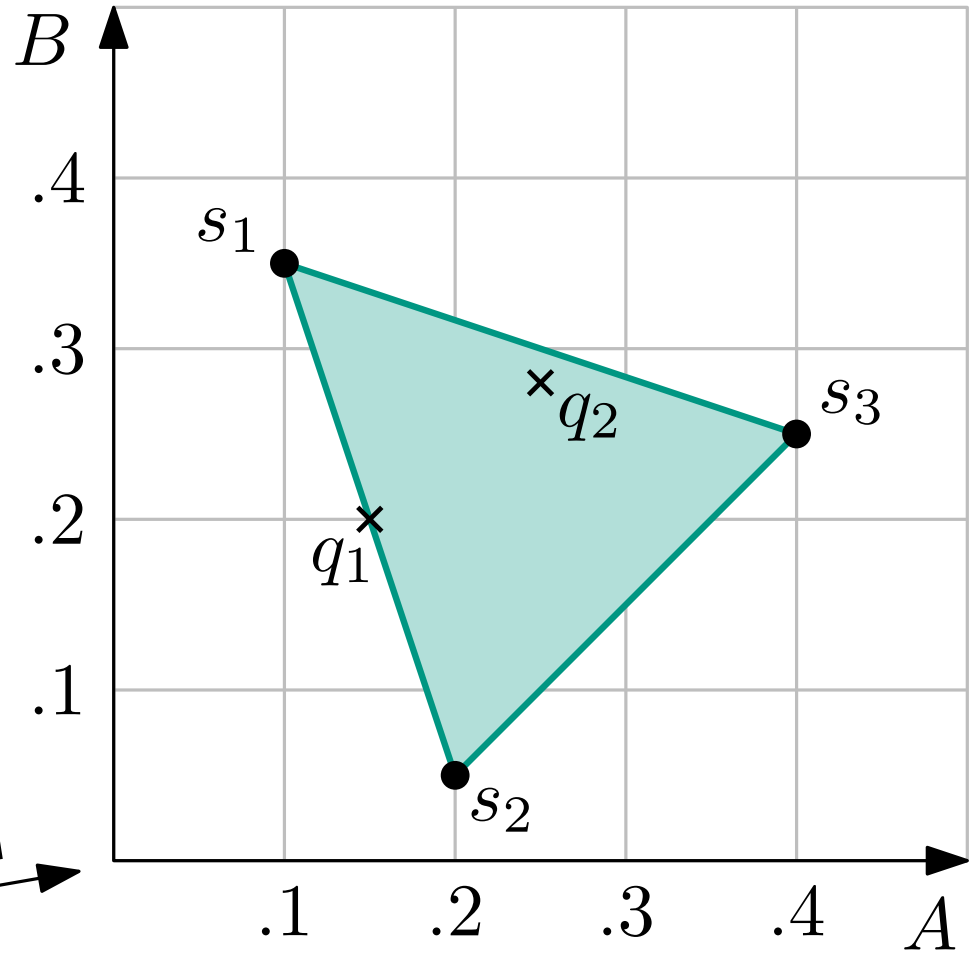
Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

mische folgende Mixturen

q_1	$15\,\%$	$20\,\%$
q_2	$25\,\%$	$28\,\%$

aus s_1, s_2, s_3?

geom. Interpretation



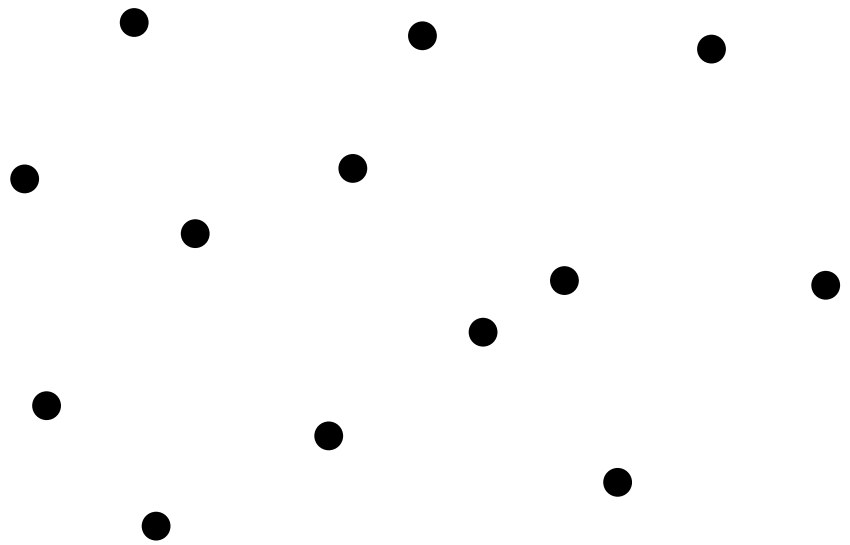
Beob: Gegeben eine Menge $S \subset \mathbb{R}^{2d}$ von Mixturen, kann man eine weitere Mixtur $q \in \mathbb{R}^{2d}$ aus S mischen $\Leftrightarrow$ $q \in CH(S)$.

Definition Konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^2$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.

Definition Konvexe Hülle

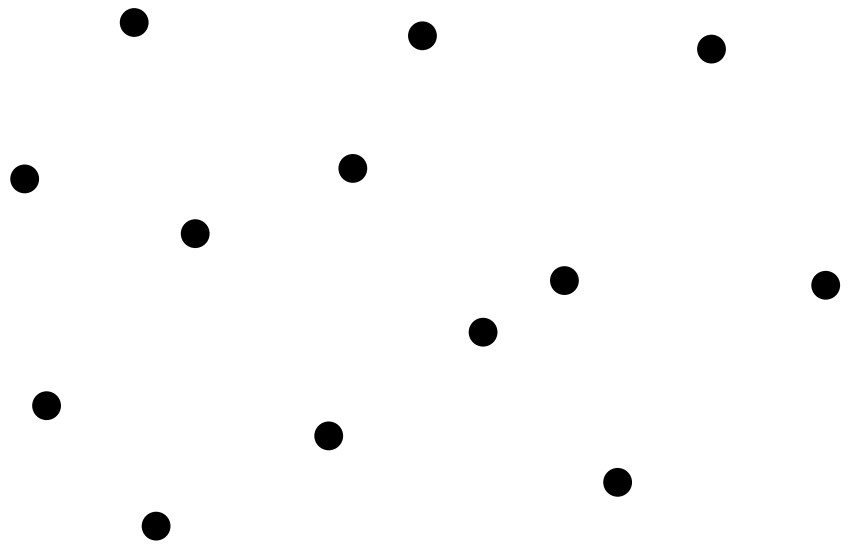
Def: Eine Menge $S \subseteq \mathbb{R}^2$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.



Definition Konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^2$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.

In der Physik:

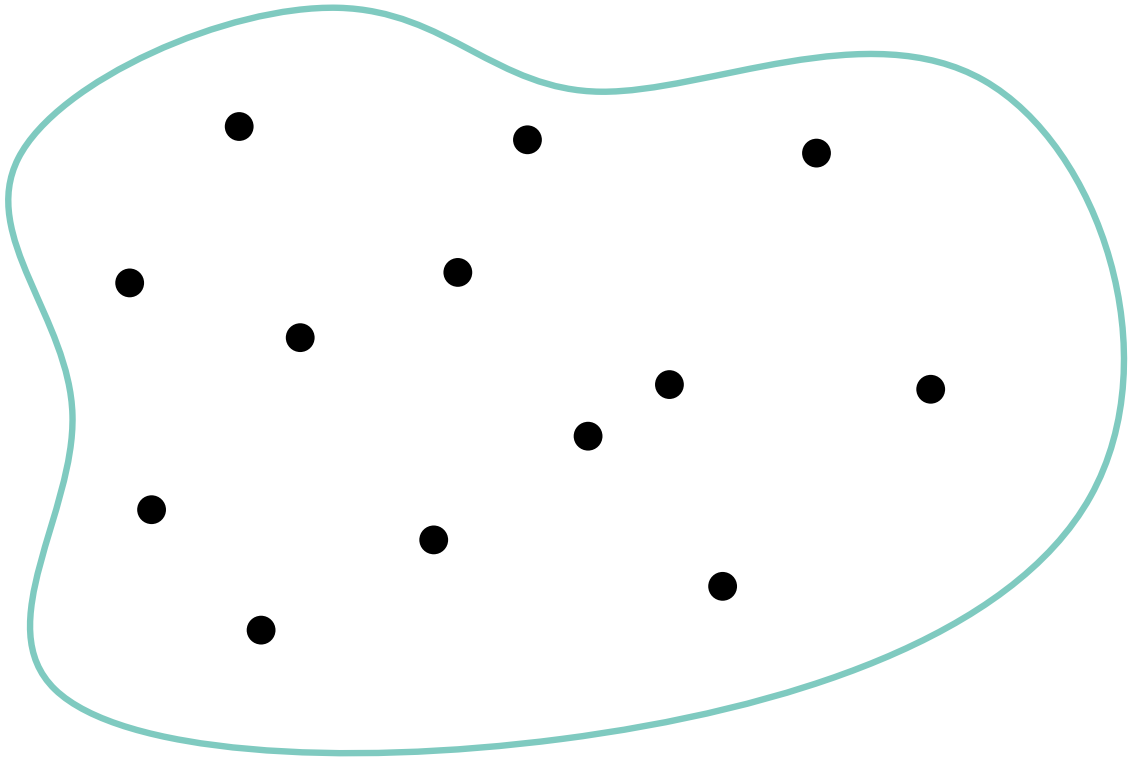


Definition Konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^2$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.

In der Physik:

- lege großes Gummiband um alle Punkte

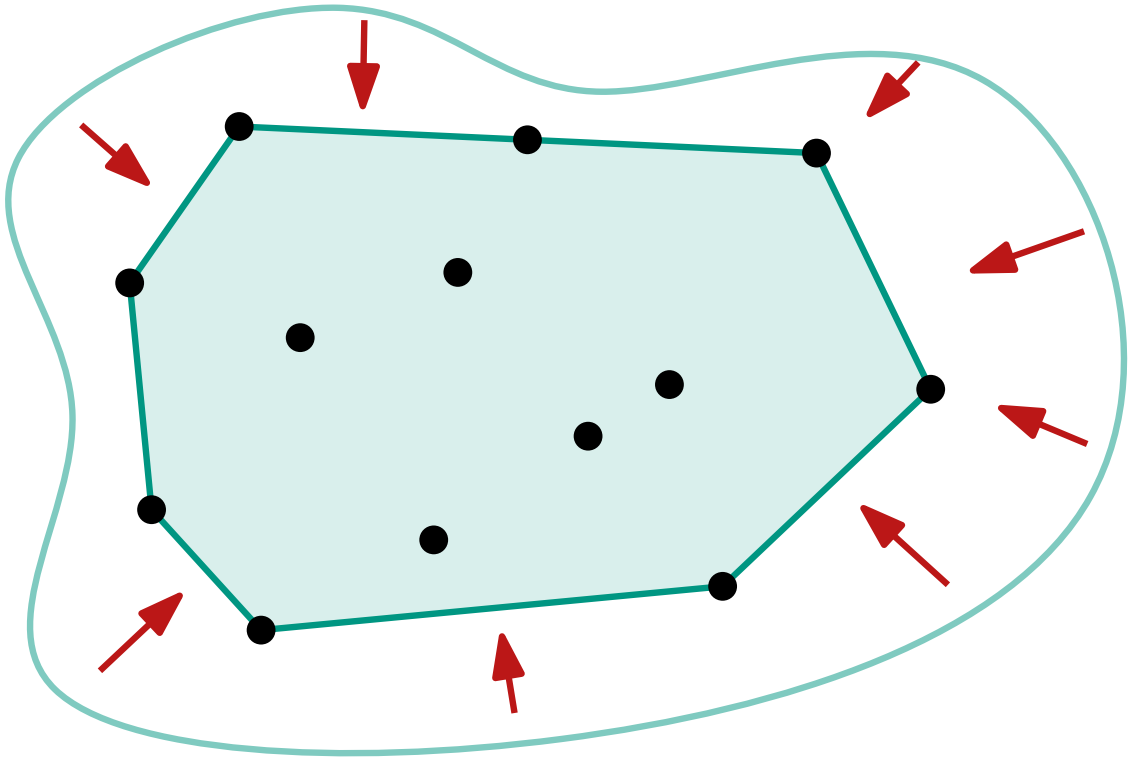


Definition Konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^2$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.

In der Physik:

- lege großes Gummiband um alle Punkte
- und lass es los!

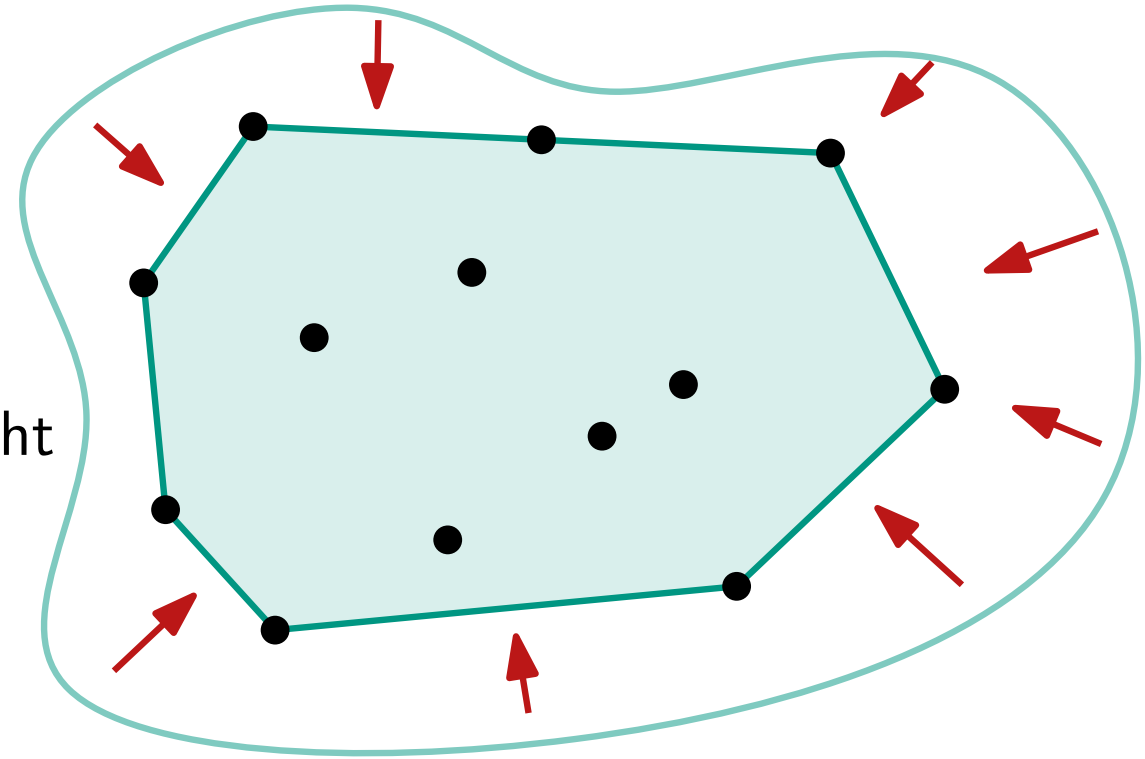


Definition Konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^2$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.

In der Physik:

- lege großes Gummiband um alle Punkte
- und lass es los!
- hilft algorithmisch leider nicht

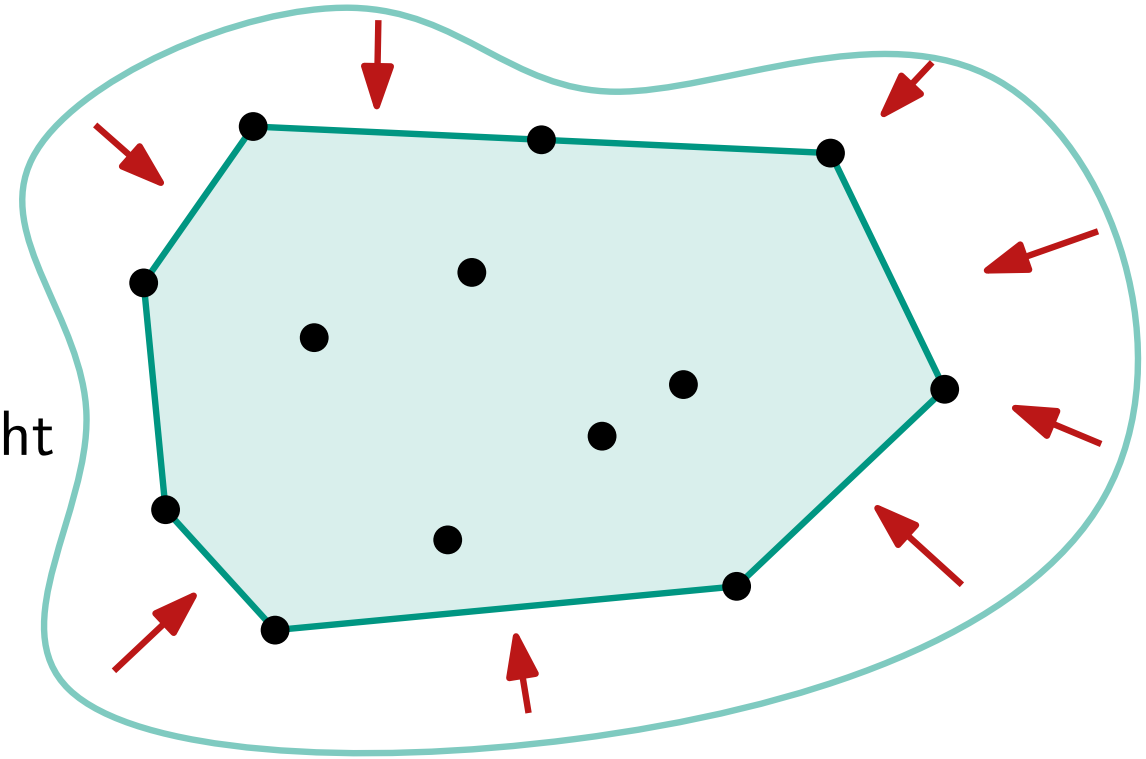


Definition Konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^2$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.

In der Physik:

- lege großes Gummiband um alle Punkte
- und lass es los!
- hilft algorithmisch leider nicht



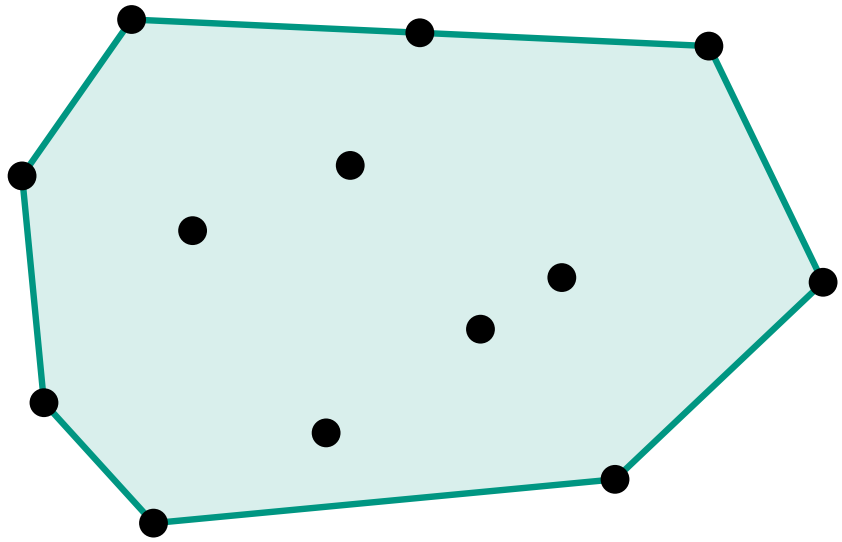
In der Mathematik:

- definiere $CH(S) = \bigcap_{C \supseteq S:\ C \text{ konvex}} C$
- hilft auch nicht :-(

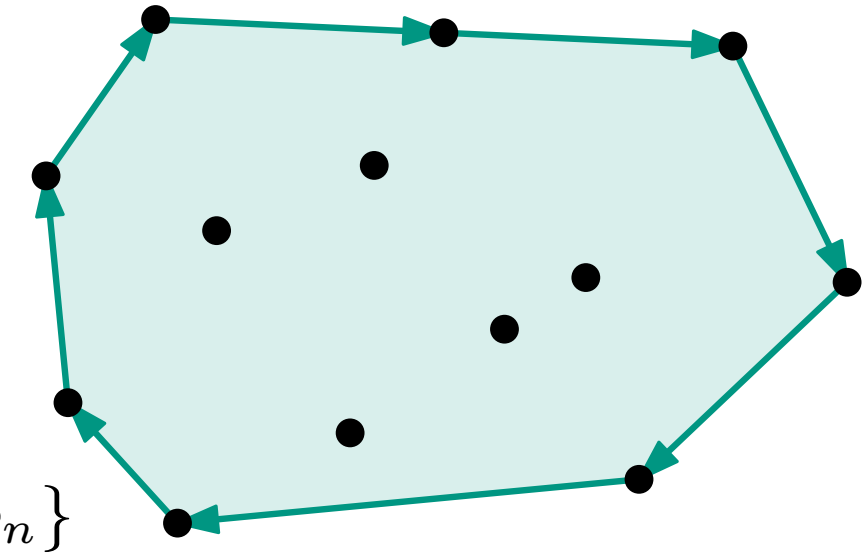
Algorithmischer Ansatz

Lemma:

Für eine Punktmenge $P \subseteq \mathbb{R}^2$ ist $CH(P)$ ein konvexes Polygon, das P enthält und dessen Ecken in P liegen.



Algorithmischer Ansatz



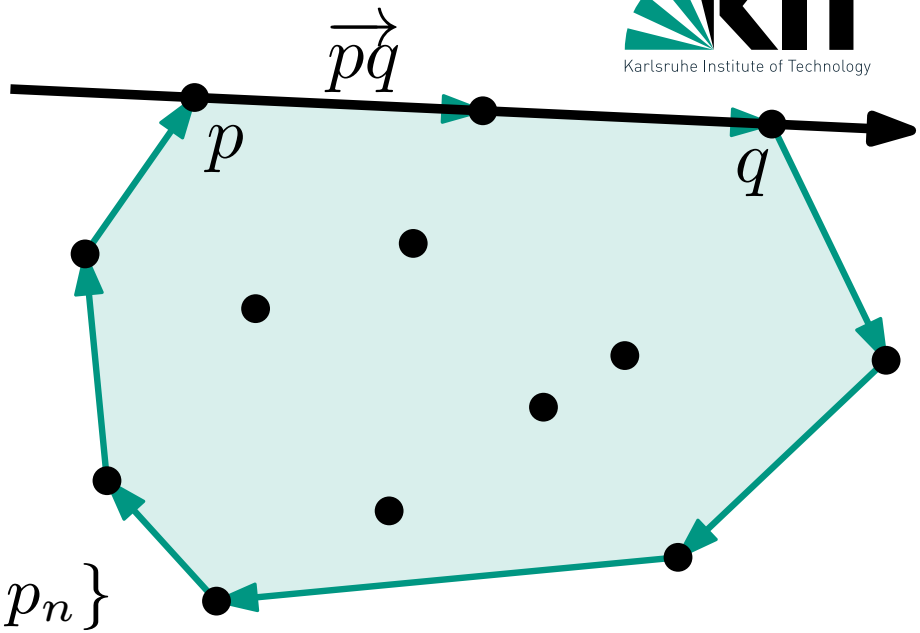
Lemma:

Für eine Punktmenge $P \subseteq \mathbb{R}^2$ ist $CH(P)$ ein konvexes Polygon, das P enthält und dessen Ecken in P liegen.

Eingabe: Punktmenge $P = \{p_1, \ldots, p_n\}$

Ausgabe: Knotenliste von $CH(P)$ im UZS

Algorithmischer Ansatz



Lemma:

Für eine Punktmenge $P \subseteq \mathbb{R}^2$ ist $CH(P)$ ein konvexes Polygon, das P enthält und dessen Ecken in P liegen.

Eingabe: Punktmenge $P = \{p_1, \ldots, p_n\}$

Ausgabe: Knotenliste von $CH(P)$ im UZS

Beobachtung:

(p, q) ist Kante von $CH(P) \Leftrightarrow$ jeder Punkt $r \in P \setminus \{p, q\}$ liegt

- strikt rechts der orientierten Geraden $\overrightarrow{pq}$ oder
- auf der Strecke $\overline{pq}$

Ein erster Algorithmus

FirstConvexHull(P)

$E \leftarrow \emptyset$

foreach $(p, q) \in P \times P$ with $p \neq q$ **do**

  valid $\leftarrow$ *true*

  foreach $r \in P$ **do**

    if not (r strikt rechts von $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then**

      valid $\leftarrow$ *false*

  if *valid* **then**

    $E \leftarrow E \cup \{(p, q)\}$

konstruiere sortierte Knotenliste L von $CH(P)$ aus E

return L

Ein erster Algorithmus

FirstConvexHull(P)

$E \leftarrow \emptyset$

foreach $(p, q) \in P \times P$ with $p \neq q$ **do**

Prüfe alle möglichen Kanten (p, q)

  valid $\leftarrow$ *true*

  foreach $r \in P$ **do**

    if not (r strikt rechts von $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then**

      valid $\leftarrow$ *false*

  if *valid* **then**

    $E \leftarrow E \cup \{(p, q)\}$

konstruiere sortierte Knotenliste L von $CH(P)$ aus E

return L

Ein erster Algorithmus

FirstConvexHull(P)

$E \leftarrow \emptyset$

foreach $(p, q) \in P \times P$ with $p \neq q$ **do**

Prüfe alle möglichen Kanten (p, q)

  valid $\leftarrow$ *true*

  foreach $r \in P$ **do**

    if not (r strikt rechts von $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then**

      valid $\leftarrow$ *false*

  if *valid* **then**

    $E \leftarrow E \cup \{(p, q)\}$

Lässt sich in $O(1)$ Zeit testen als

$$\begin{vmatrix} x_r & y_r & 1 \\ x_p & y_p & 1 \\ x_q & y_q & 1 \end{vmatrix} < 0$$

$\rightarrow$ Übung

konstruiere sortierte Knotenliste L von $CH(P)$ aus E

return L

Laufzeitanalyse

FirstConvexHull(P)

$E \leftarrow \emptyset$

foreach $(p, q) \in P \times P$ with $p \neq q$ **do**

  valid $\leftarrow$ *true*

  foreach $r \in P$ **do**

    if not (r strikt rechts von $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then**

      valid $\leftarrow$ *false*

  if *valid* **then**

    $E \leftarrow E \cup \{(p, q)\}$

konstruiere sortierte Knotenliste L von $CH(P)$ aus E

return L

Laufzeitanalyse

FirstConvexHull(P)

$E \leftarrow \emptyset$

foreach $(p, q) \in P \times P$ with $p \neq q$ **do**

  valid $\leftarrow$ *true*

  foreach $r \in P$ **do**

    if not (r strikt rechts von $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then**

      valid $\leftarrow$ *false*

  $\Theta(1)$

  if *valid* **then**

    $E \leftarrow E \cup \{(p, q)\}$

konstruiere sortierte Knotenliste L von $CH(P)$ aus E

return L

Laufzeitanalyse

FirstConvexHull(P)

$E \leftarrow \emptyset$

foreach $(p, q) \in P \times P$ with $p \neq q$ **do**

  valid $\leftarrow$ *true*

  foreach $r \in P$ **do**

    if not (r strikt rechts von $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then**

      valid $\leftarrow$ *false*

  $\Theta(1)$

  if *valid* **then**

    $E \leftarrow E \cup \{(p, q)\}$

$\Theta(n)$

konstruiere sortierte Knotenliste L von $CH(P)$ aus E

return L

Laufzeitanalyse

FirstConvexHull(P)

$E \leftarrow \emptyset$

foreach $(p, q) \in P \times P$ with $p \neq q$ **do** $(n^2 - n)\cdot$

  valid $\leftarrow$ *true*

  foreach $r \in P$ **do** $\Theta(n)$

    if not (r strikt rechts von $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then** $\Theta(1)$

      valid $\leftarrow$ *false*

  if *valid* **then**

    $E \leftarrow E \cup \{(p, q)\}$

konstruiere sortierte Knotenliste L von $CH(P)$ aus E

return L

Laufzeitanalyse

FirstConvexHull(P)

$E \leftarrow \emptyset$

foreach $(p, q) \in P \times P$ with $p \neq q$ **do** $(n^2 - n)\cdot$

  valid $\leftarrow$ *true*

  foreach $r \in P$ **do** $\Theta(n)$

    if not (r strikt rechts von $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then** $\Theta(1)$

      valid $\leftarrow$ *false*

  if *valid* **then**

    $E \leftarrow E \cup \{(p, q)\}$

konstruiere sortierte Knotenliste L von $CH(P)$ aus E $\Theta(n^3)$

return L

Laufzeitanalyse

FirstConvexHull(P)

$E \leftarrow \emptyset$

foreach $(p,q) \in P \times P$ with $p \neq q$ **do** $(n^2-n)\cdot$

  valid $\leftarrow$ *true*

  foreach $r \in P$ **do** $\Theta(n)$

    if not (r strikt rechts von $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then** $\Theta(1)$

      valid $\leftarrow$ *false*

  if *valid* **then**

    $E \leftarrow E \cup \{(p,q)\}$

konstruiere sortierte Knotenliste L von $CH(P)$ aus E $\Theta(n^3)$

return L

Aufgabe: Wie implementiert man das?

Laufzeitanalyse

FirstConvexHull(P)

$E \leftarrow \emptyset$

foreach $(p,q) \in P \times P$ with $p \neq q$ **do** $(n^2-n)\cdot$

  valid $\leftarrow$ *true*

  foreach $r \in P$ **do** $\Theta(n)$

    if not (r strikt rechts von $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then** $\Theta(1)$

      valid $\leftarrow$ *false*

  if *valid* **then**

    $E \leftarrow E \cup \{(p,q)\}$

konstruiere sortierte Knotenliste L von $CH(P)$ aus E $O(n^2)$

return L $\Theta(n^3)$

Laufzeitanalyse

FirstConvexHull(P)

$E \leftarrow \emptyset$

foreach $(p,q) \in P \times P$ with $p \neq q$ **do** $(n^2-n)\cdot$

  valid $\leftarrow$ *true*

  foreach $r \in P$ **do** $\Theta(1)$ $\Theta(n)$ $\Theta(n^3)$

    if not (r strikt rechts von $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then**

      valid $\leftarrow$ *false*

  if *valid* **then**

    $E \leftarrow E \cup \{(p,q)\}$

konstruiere sortierte Knotenliste L von $CH(P)$ aus E $O(n^2)$

return L

Lemma: Die konvexe Hülle von n Punkten in der Ebene lässt sich in $O(n^3)$ Zeit berechnen.

Laufzeitanalyse

FirstConvexHull(P)

$E \leftarrow \emptyset$

foreach $(p,q) \in P \times P$ with $p \neq q$ **do** $(n^2-n)\cdot$

  valid $\leftarrow$ *true*

  foreach $r \in P$ **do** $\Theta(1)$ $\Theta(n)$ $\Theta(n^3)$

    if not (r strikt rechts von $\overrightarrow{pq}$ **or** $r \in \overline{pq}$) **then**

      valid $\leftarrow$ *false*

  if *valid* **then**

    $E \leftarrow E \cup \{(p,q)\}$

Geht es auch besser?

konstruiere sortierte Knotenliste L von $CH(P)$ aus E $O(n^2)$

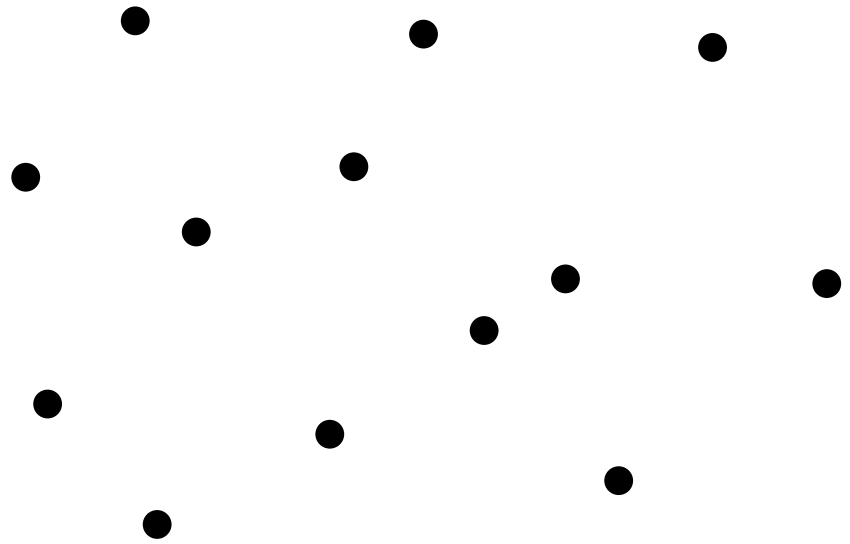
return L

Lemma: Die konvexe Hülle von n Punkten in der Ebene lässt sich in $O(n^3)$ Zeit berechnen.

Inkrementeller Ansatz

Idee: Für $i = 1, \ldots, n$ bestimme $CH(P_i)$ mit $P_i = \{p_1, \ldots, p_i\}$

Frage: Welche Ordnung der Punkte ist sinnvoll?

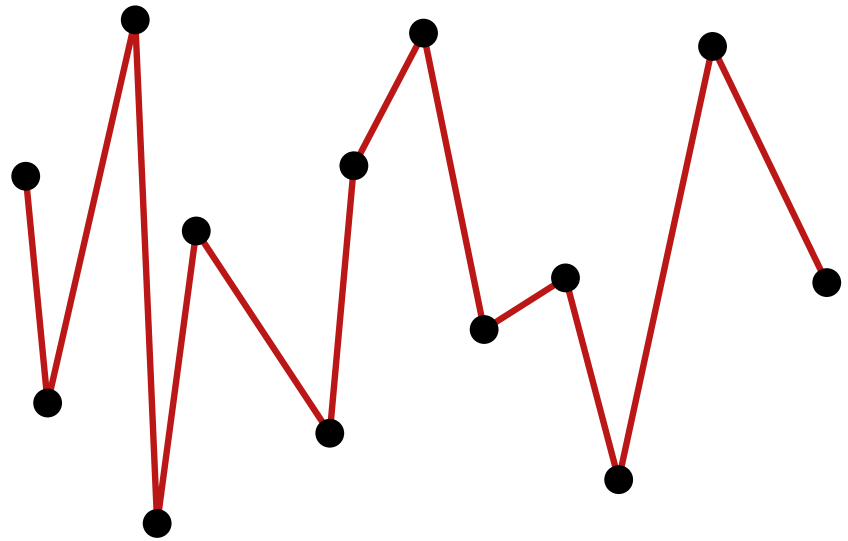


Inkrementeller Ansatz

Idee: Für $i = 1, \ldots, n$ bestimme $CH(P_i)$ mit $P_i = \{p_1, \ldots, p_i\}$

Frage: Welche Ordnung der Punkte ist sinnvoll?

Antwort: Von links nach rechts!



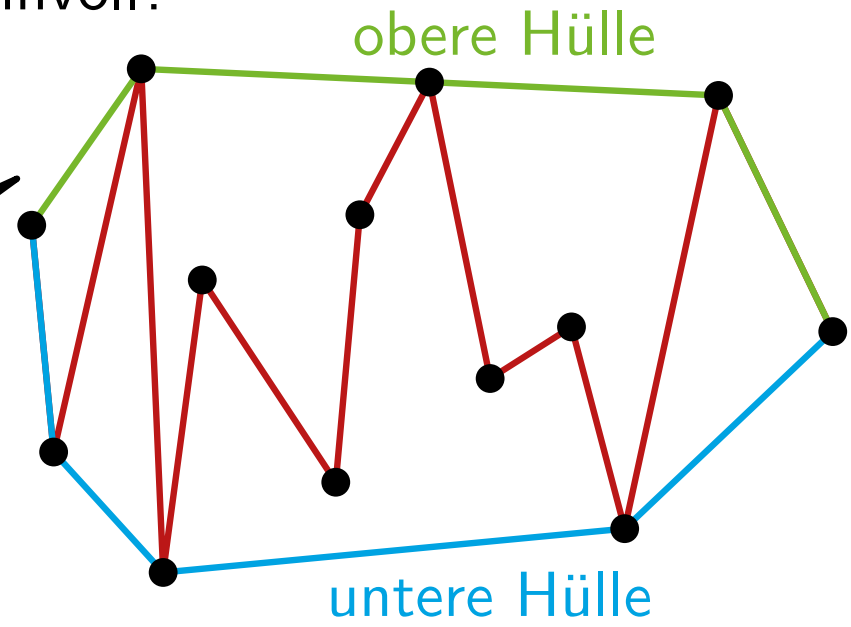
Inkrementeller Ansatz

Idee: Für $i = 1, \ldots, n$ bestimme $CH(P_i)$ mit $P_i = \{p_1, \ldots, p_i\}$

Frage: Welche Ordnung der Punkte ist sinnvoll?

Antwort: Von links nach rechts!

Betrachte obere und untere Hülle getrennt



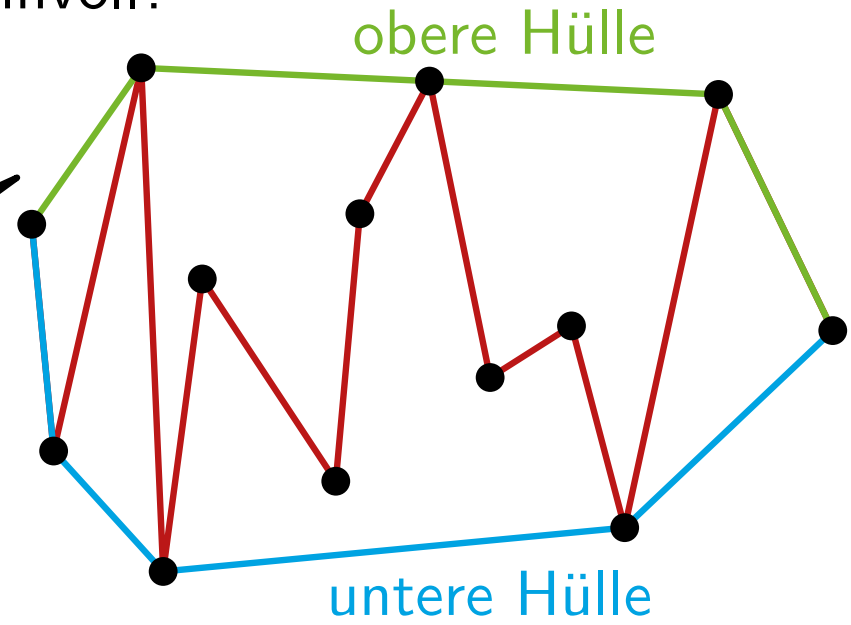
Inkrementeller Ansatz

Idee: Für $i = 1, \ldots, n$ bestimme $CH(P_i)$ mit $P_i = \{p_1, \ldots, p_i\}$

Frage: Welche Ordnung der Punkte ist sinnvoll?

Antwort: Von links nach rechts!

Betrachte obere und untere Hülle getrennt



UpperConvexHull(P)

$\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sortiere P von links nach rechts

$L \leftarrow \langle p_1, p_2 \rangle$

for $i \leftarrow 3$ **to** n **do**

L.append(p_i)

?

return L

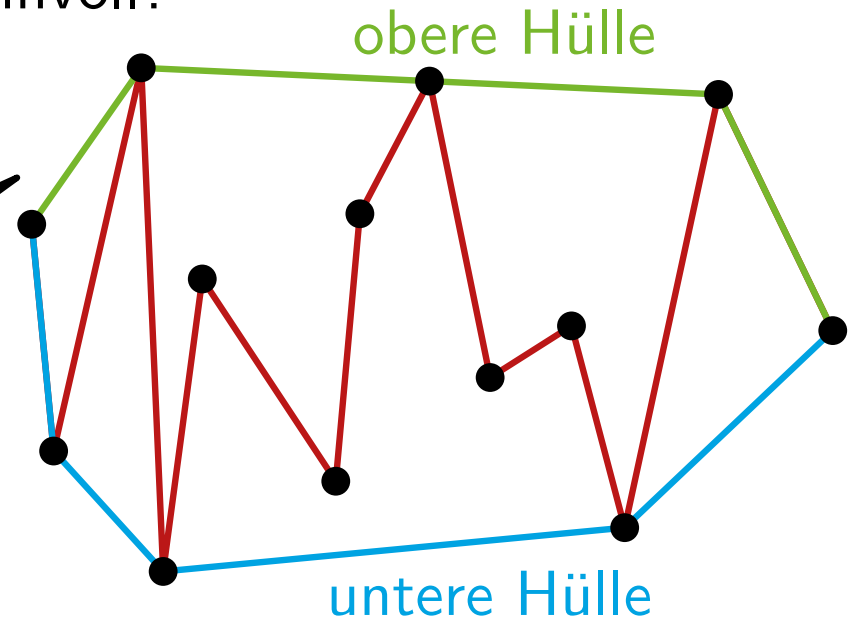
Inkrementeller Ansatz

Idee: Für $i = 1, \ldots, n$ bestimme $CH(P_i)$ mit $P_i = \{p_1, \ldots, p_i\}$

Frage: Welche Ordnung der Punkte ist sinnvoll?

Antwort: Von links nach rechts!

Betrachte obere und untere Hülle getrennt



UpperConvexHull(P)

$\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sortiere P von links nach rechts

$L \leftarrow \langle p_1, p_2 \rangle$

for $i \leftarrow 3$ **to** n **do**

  L.append(p_i)

  while $|L| > 2$ **and** letzte 3 Punkte in L kein Rechtsknick **do**

    entferne vorletzten Punkt aus L

return L

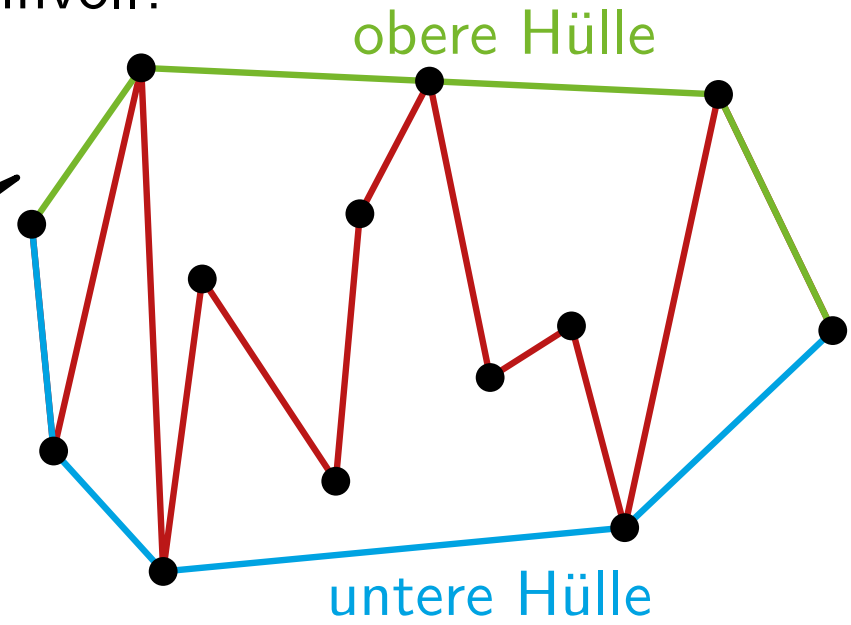
Inkrementeller Ansatz

Idee: Für $i = 1, \ldots, n$ bestimme $CH(P_i)$ mit $P_i = \{p_1, \ldots, p_i\}$

Frage: Welche Ordnung der Punkte ist sinnvoll?

Antwort: Von links nach rechts!

Betrachte obere und untere Hülle getrennt



UpperConvexHull(P)

$\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sortiere P von links nach rechts

$L \leftarrow \langle p_1, p_2 \rangle$

for $i \leftarrow 3$ **to** n **do**

  L.append(p_i)

  while $|L| > 2$ **and** letzte 3 Punkte in L kein Rechtsknick **do**

    entferne vorletzten Punkt aus L

return L

untere Hülle analog!

Laufzeitanalyse

UpperConvexHull(P)

$\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sortiere P von links nach rechts

$L \leftarrow \langle p_1, p_2 \rangle$

for $i \leftarrow 3$ **to** n **do**

  L.append(p_i)

  while $|L| > 2$ **and** letzte 3 Punkte in L kein Rechtsknick **do**

    entferne vorletzten Punkt aus L

return L

Laufzeitanalyse

UpperConvexHull(P)

$\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sortiere P von links nach rechts $\quad O(n \log n)$

$L \leftarrow \langle p_1, p_2 \rangle$

for $i \leftarrow 3$ **to** n **do**

  L.append(p_i)

  while $|L| > 2$ **and** letzte 3 Punkte in L kein Rechtsknick **do**

    entferne vorletzten Punkt aus L

return L

Laufzeitanalyse

UpperConvexHull(P)

$\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sortiere P von links nach rechts $\quad O(n \log n)$

$L \leftarrow \langle p_1, p_2 \rangle$

for $i \leftarrow 3$ **to** n **do** $\quad (n-2)\cdot$

  L.append(p_i)

  while $|L| > 2$ **and** letzte 3 Punkte in L kein Rechtsknick **do** ?

    entferne vorletzten Punkt aus L

return L

Laufzeitanalyse

UpperConvexHull(P)

$\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sortiere P von links nach rechts $\quad O(n \log n)$

$L \leftarrow \langle p_1, p_2 \rangle$

for $i \leftarrow 3$ **to** n **do** $\quad (n-2)\cdot$

  L.append(p_i)

  while $|L| > 2$ **and** letzte 3 Punkte in L kein Rechtsknick **do**

    entferne vorletzten Punkt aus L $\quad ?$

return L

Amortisierte Analyse

- jeder Punkt wird genau einmal in L eingefügt
- ein Punkt in L wird höchstens einmal aus L entfernt
- $\Rightarrow$ Laufzeit der **for**-Schleife inkl. **while**-Schleife ist $O(n)$

Laufzeitanalyse

UpperConvexHull(P)

$\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sortiere P von links nach rechts — $O(n \log n)$

$L \leftarrow \langle p_1, p_2 \rangle$

for $i \leftarrow 3$ **to** n **do** — ~~$(n-2)\cdot$~~

  L.append(p_i)

  while $|L| > 2$ **and** letzte 3 Punkte in L kein Rechtsknick **do** — ~~?~~ $O(n)$

    entferne vorletzten Punkt aus L

return L

Amortisierte Analyse

- jeder Punkt wird genau einmal in L eingefügt
- ein Punkt in L wird höchstens einmal aus L entfernt
- $\Rightarrow$ Laufzeit der **for**-Schleife inkl. **while**-Schleife ist $O(n)$

Laufzeitanalyse

UpperConvexHull(P)

$\langle p_1, p_2, \ldots, p_n \rangle \leftarrow$ sortiere P von links nach rechts — $O(n \log n)$

$L \leftarrow \langle p_1, p_2 \rangle$

for $i \leftarrow 3$ **to** n **do** — ~~$(n-2)\cdot$~~ $O(n)$

  L.append(p_i)

  while $|L| > 2$ **and** letzte 3 Punkte in L kein Rechtsknick **do** — ~~?~~

    entferne vorletzten Punkt aus L

return L

Amortisierte Analyse

- jeder Punkt wird genau einmal in L eingefügt
- ein Punkt in L wird höchstens einmal aus L entfernt
- $\Rightarrow$ Laufzeit der **for**-Schleife inkl. **while**-Schleife ist $O(n)$

Satz: Die konvexe Hülle von n Punkten in der Ebene lässt sich mit *Graham's Scan* in $O(n \log n)$ Zeit berechnen.

Diskussion

Geht es schneller als in $O(n \log n)$ Zeit?

Diskussion

Geht es schneller als in $O(n \log n)$ Zeit?

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.
Alternativ: **ausgabesensitive** Algorithmen mit Laufzeit $O(h \cdot n)$ oder $O(n \log h)$ für $|CH(P)| = h$.

Diskussion

Geht es schneller als in $O(n \log n)$ Zeit?

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.
Alternativ: **ausgabesensitive** Algorithmen mit Laufzeit $O(h \cdot n)$ oder $O(n \log h)$ für $|CH(P)| = h$.

Was passiert wenn die Sortierung von P nicht eindeutig ist?

Diskussion

Geht es schneller als in $O(n \log n)$ Zeit?

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.
Alternativ: **ausgabesensitive** Algorithmen mit Laufzeit $O(h \cdot n)$ oder $O(n \log h)$ für $|CH(P)| = h$.

Was passiert wenn die Sortierung von P nicht eindeutig ist?

Nutze lexikographische Ordnung!

Diskussion

Geht es schneller als in $O(n \log n)$ Zeit?

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.
Alternativ: **ausgabesensitive** Algorithmen mit Laufzeit $O(h \cdot n)$ oder $O(n \log h)$ für $|CH(P)| = h$.

Was passiert wenn die Sortierung von P nicht eindeutig ist?

Nutze lexikographische Ordnung!

Was passiert mit kollinearen Punkten in $CH(P)$?

Diskussion

Geht es schneller als in $O(n \log n)$ Zeit?

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.
Alternativ: **ausgabesensitive** Algorithmen mit Laufzeit $O(h \cdot n)$ oder $O(n \log h)$ für $|CH(P)| = h$.

Was passiert wenn die Sortierung von P nicht eindeutig ist?

Nutze lexikographische Ordnung!

Was passiert mit kollinearen Punkten in $CH(P)$?

Bilden keinen Rechtsknick, also wird innerer Punkt gelöscht.

Diskussion

Geht es schneller als in $O(n \log n)$ Zeit?

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.
Alternativ: **ausgabesensitive** Algorithmen mit Laufzeit $O(h \cdot n)$ oder $O(n \log h)$ für $|CH(P)| = h$.

Was passiert wenn die Sortierung von P nicht eindeutig ist?

Nutze lexikographische Ordnung!

Was passiert mit kollinearen Punkten in $CH(P)$?

Bilden keinen Rechtsknick, also wird innerer Punkt gelöscht.

Wie steht es um die Robustheit der beiden Algorithmen?

Diskussion

Geht es schneller als in $O(n \log n)$ Zeit?

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.
Alternativ: **ausgabesensitive** Algorithmen mit Laufzeit $O(h \cdot n)$ oder $O(n \log h)$ für $|CH(P)| = h$.

Was passiert wenn die Sortierung von P nicht eindeutig ist?

Nutze lexikographische Ordnung!

Was passiert mit kollinearen Punkten in $CH(P)$?

Bilden keinen Rechtsknick, also wird innerer Punkt gelöscht.

Wie steht es um die Robustheit der beiden Algorithmen?

- Robustheit bzgl. Ungenauigkeit von Fließkommaarithmetik
- FirstConvexHull liefert evtl. kein gültiges Polygon
- Upper/LowerConvexHull liefert zumindest immer ein gültiges Polygon, evtl. mit kleinen Fehlern

Phasen der Algorithmenentwicklung

1.) Degenerierte Fälle ausblenden ($\rightarrow$ *allgemeine Lage*)
- eindeutige x-Koordinaten
- keine drei kollinearen Punkte
- ...

2.) Anpassung an degenerierte Eingaben
- in bisherige Behandlung integrieren (z.B. lexikographische Ordnung)
- notfalls gesonderte Behandlung

3.) Implementierung
- primitive Operationen (verfügbar in Bibliotheken?)
- Robustheit