Karlsruhe Institute of Technology

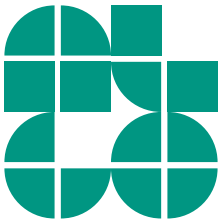
Vorlesung Algorithmische Geometrie

Einführung & Konvexe Hülle

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Martin Nöllenburg

14.04.2014



AlgoGeom-Team

Dozent

- Martin Nöllenburg
- noellenburg@kit.edu
- Raum 319
- Sprechzeiten: nach Vereinbarung

AlgoGeom-Team

Dozent

- Martin Nöllenburg
- noellenburg@kit.edu
- Raum 319
- Sprechzeiten: nach Vereinbarung

Übungsleiter

- Benjamin Niedermann
- benjamin.niedermann@kit.edu
- Raum 322
- Sprechzeiten: nach Vereinbarung

AlgoGeom-Team

Dozent

- Martin Nöllenburg
- noellenburg@kit.edu
- Raum 319
- Sprechzeiten: nach Vereinbarung

Übungsleiter

- Benjamin Niedermann
- benjamin.niedermann@kit.edu
- Raum 322
- Sprechzeiten: nach Vereinbarung

Termine

- Vorlesung: Di 9:45 – 11:15 Uhr, Raum 301
- Übung: Mi 15:45 – 17:15 Uhr, Raum 301 (ab 16.04.)

Organisatorisches

Webseite

`i11www.iti.kit.edu/teaching/sommer2014/compgeom`

- aktuelle Informationen
- Vorlesungsfolien
- Übungsblätter
- Zusatzmaterial

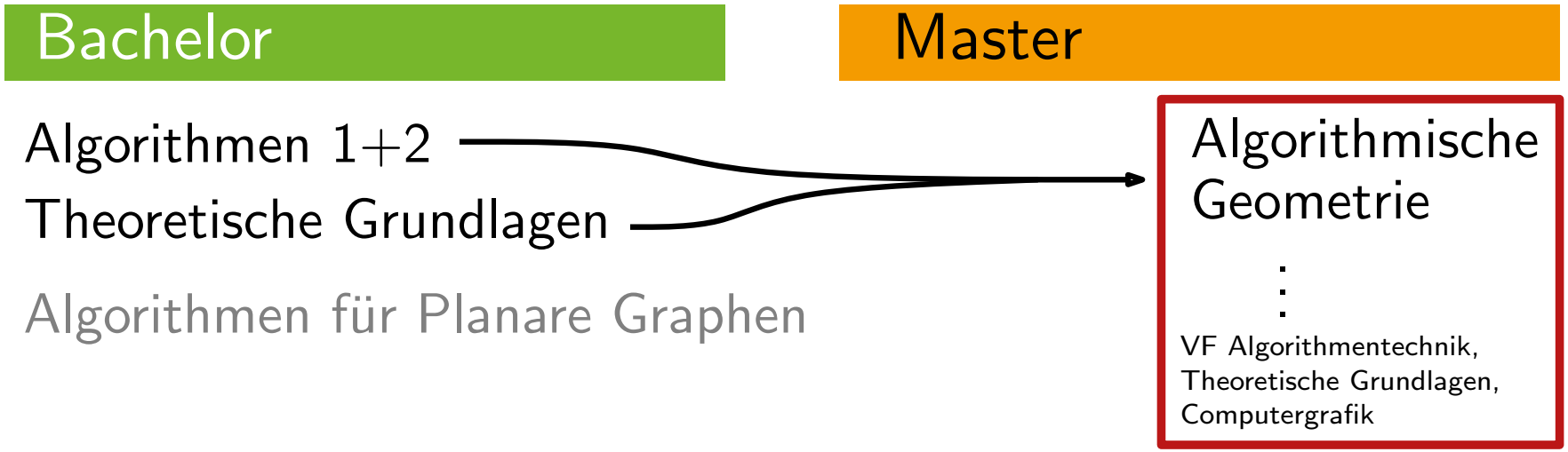
Organisatorisches

Webseite

`i11www.iti.kit.edu/teaching/sommer2014/compgeom`

- aktuelle Informationen
- Vorlesungsfolien
- Übungsblätter
- Zusatzmaterial

Algorithmische Geometrie im Master-Studium Informatik



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: 5LP = 150h

aktive Teilnahme an Vorlesung und Übung	ca. 35h
Vor- und Nachbereitung	ca. 25h
Bearbeiten der Übungsblätter	ca. 20h
Projektarbeit	ca. 40h
Prüfungsvorbereitung	ca. 30h

Prüfbarkeit

Master Informatik

- Algorithmische Geometrie (IN4INAG) [5 LP]
- Algorithm Engineering & Anwendungen (IN4INAEA) [5 LP]
- Design und Analyse von Algorithmen (IN4INDAA) [5 LP]
- Algorithmen der Computergrafik (IN4INACG) [3 LP]

Prüfbarkeit

Master Informatik

- Algorithmische Geometrie (IN4INAG) [5 LP]
- Algorithm Engineering & Anwendungen (IN4INAEA) [5 LP]
- Design und Analyse von Algorithmen (IN4INDAA) [5 LP]
- Algorithmen der Computergrafik (IN4INACG) [3 LP]

Prüfungsmodalitäten

- semesterbegleitende Projektarbeit in kleinen Teams (Visualisierung von geometrischen Algorithmen)
  $\rightarrow$ 20% der Note
- mündliche Einzelprüfung (ca. 20 Minuten)
  $\rightarrow$ 80% der Note

Prüfbarkeit

Master Informatik

- Algorithmische Geometrie (IN4INAG) [5 LP]
- Algorithm Engineering & Anwendungen (IN4INAEA) [5 LP]
- Design und Analyse von Algorithmen (IN4INDAA) [5 LP]
- Algorithmen der Computergrafik (IN4INACG) [3 LP]

Prüfungsmodalitäten

mehr dazu in 1–2 Wochen

- semesterbegleitende Projektarbeit in kleinen Teams (Visualisierung von geometrischen Algorithmen)
  $\rightarrow$ 20% der Note
- mündliche Einzelprüfung (ca. 20 Minuten)
  $\rightarrow$ 80% der Note

Übungsbetrieb

Ausgabe Blatt 1 morgen
for Woche $i = 2 \ldots 14$ **do**
  if Tag = Dienstag **then**
    Ausgabe Blatt i
    Abgabe Blatt $i - 1$
  if Tag = Mittwoch **then**
    Besprechung Blatt $i - 1$

- Bearbeitung der Aufgaben und Abgabe der Lösungen in Zweiergruppen erwünscht
- aktive Übungsteilnahme wird erwartet
- Übung wöchentlich ca. 45–60 Minuten
- Abweichungen werden rechtzeitig bekannt gegeben

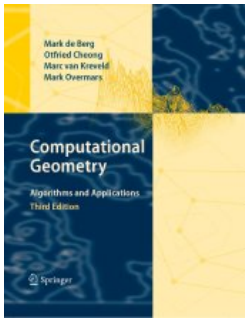
Übungsbetrieb

Ausgabe Blatt 1 morgen
for Woche $i = 2 \ldots 14$ **do**
  if Tag $=$ Dienstag **then**
    Ausgabe Blatt i
    Abgabe Blatt $i - 1$
  if Tag $=$ Mittwoch **then**
    Besprechung Blatt $i - 1$

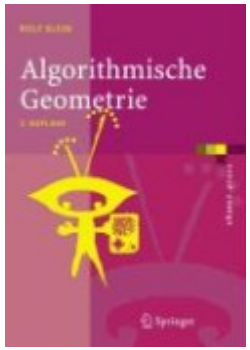
- Bearbeitung der Aufgaben und Abgabe der Lösungen in Zweiergruppen erwünscht
- aktive Übungsteilnahme wird erwartet
- Übung wöchentlich ca. 45–60 Minuten
- Abweichungen werden rechtzeitig bekannt gegeben

nächste Woche (23.04.) keine Übung!

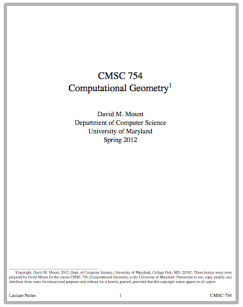
Literatur



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



David Mount:
Computational Geometry
Lecture Notes CMSC 754, U. Maryland, 2012

`http://www.cs.umd.edu/class/spring2012/cmsc754/Lects/cmsc754-lects.pdf`

Beide Bücher 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 und Bildverarbeitung
- Visualisierung
- Geoinformationssysteme (GIS)
- Robotik
- ...

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?

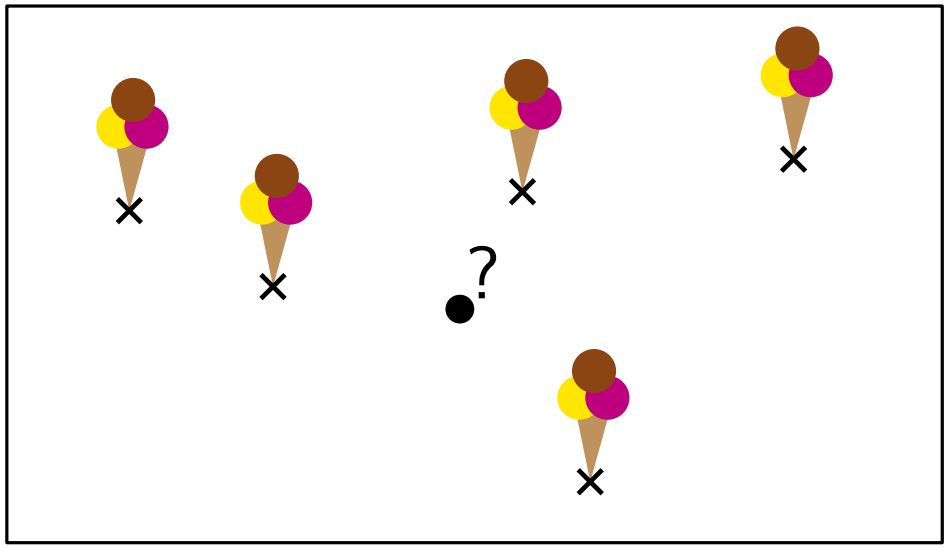
- Computergrafik und Bildverarbeitung
- Visualisierung
- Geoinformationssysteme (GIS)
- Robotik
- ...

Zentrale Themen

- geometrische Algorithmen und Datenstrukturen

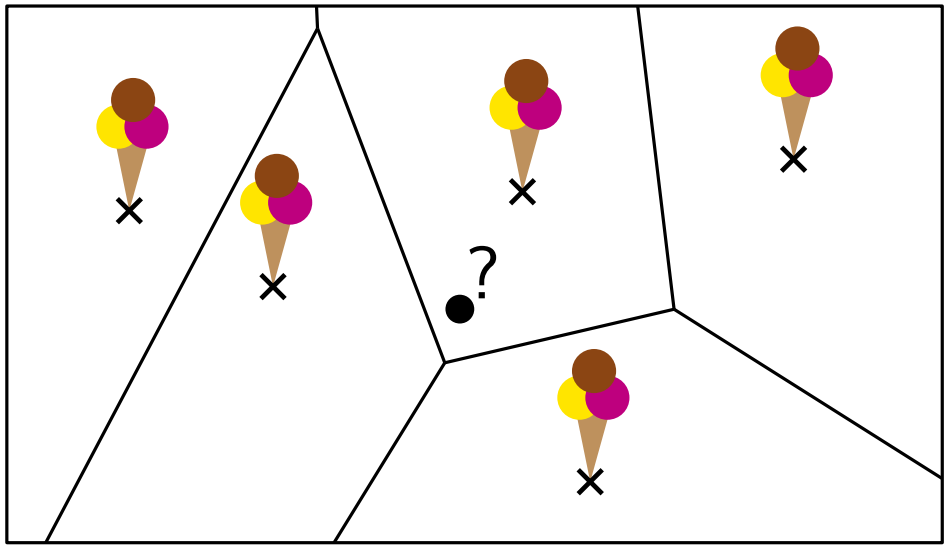
Beispiel 1

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?



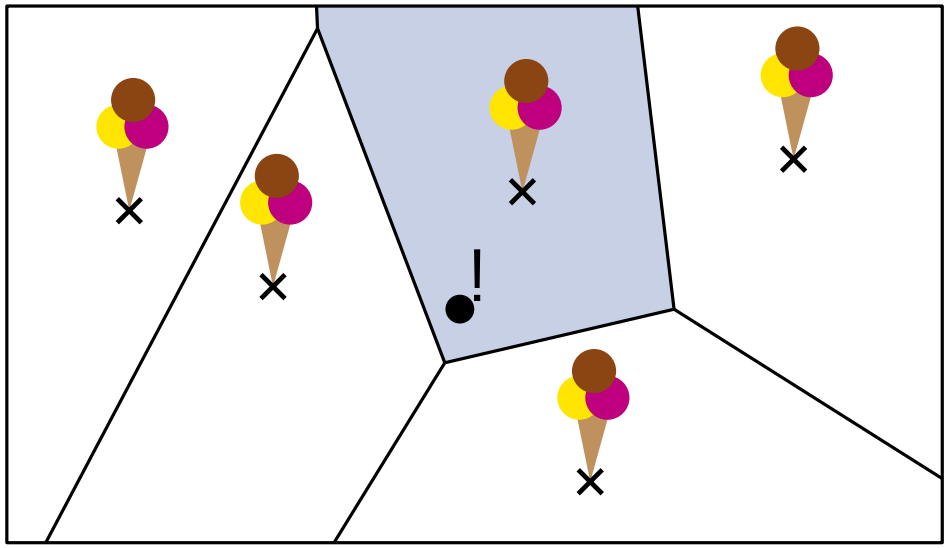
Beispiel 1

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?



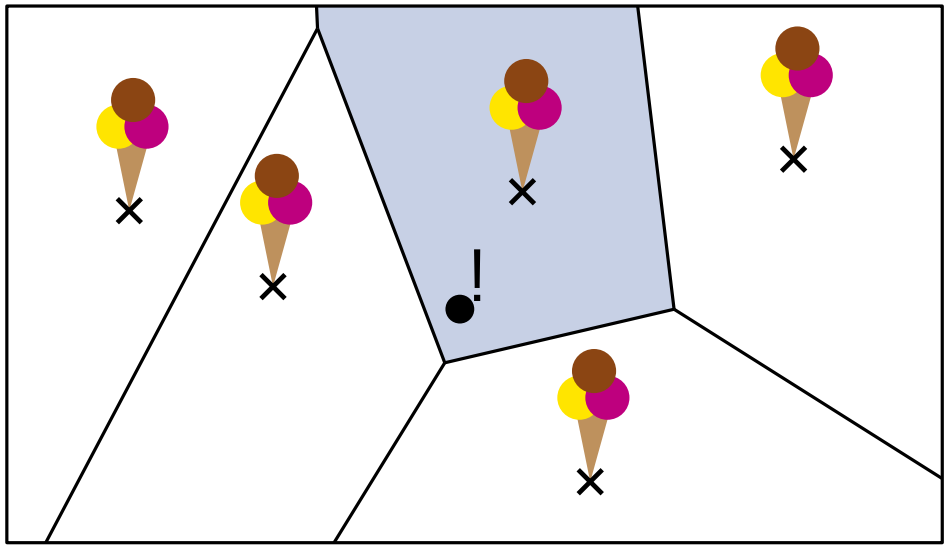
Beispiel 1

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

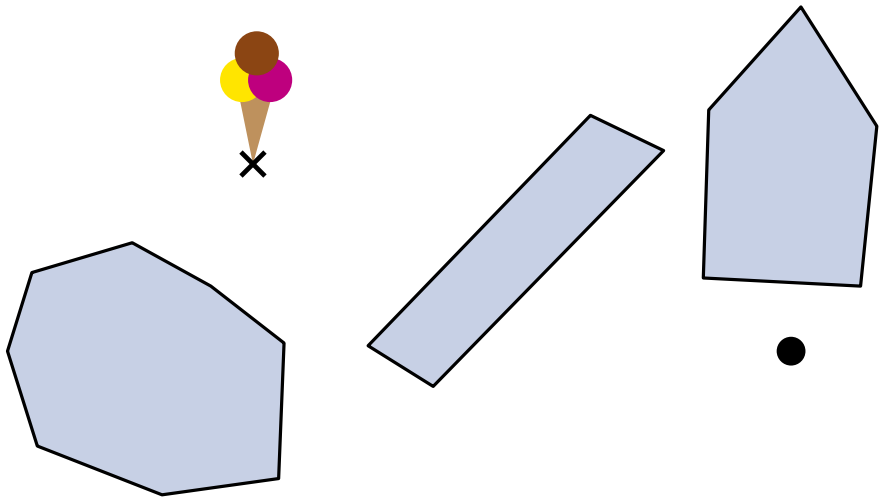
Ein sonniger Frühlingstag in Karlsruhe. Man kennt die Eisdielen in der Stadt und sucht die nächstgelegene. Wie kann man die nächste Eisdiele für jeden beliebigen Standort in einer Landkarte bestimmen?



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

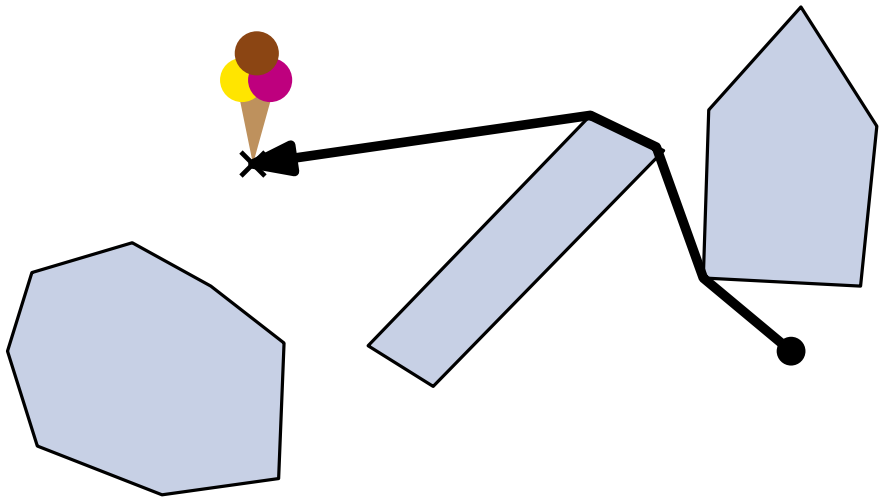
Beispiel 2

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



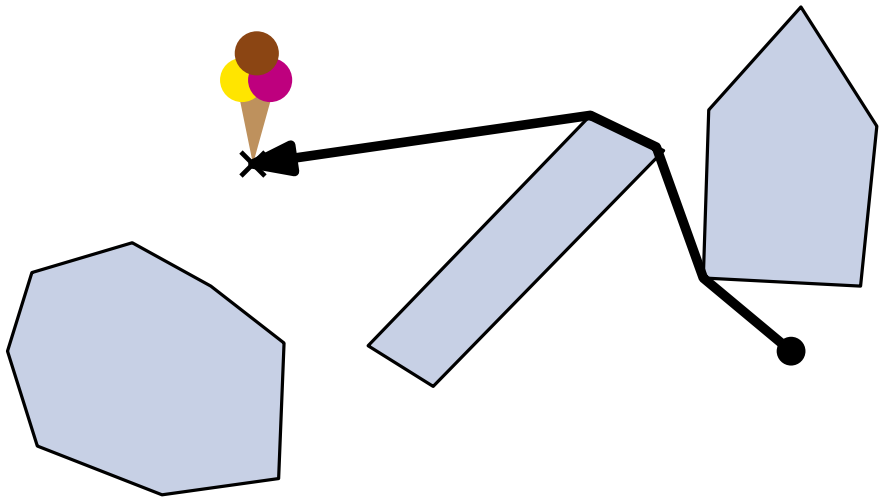
Beispiel 2

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?

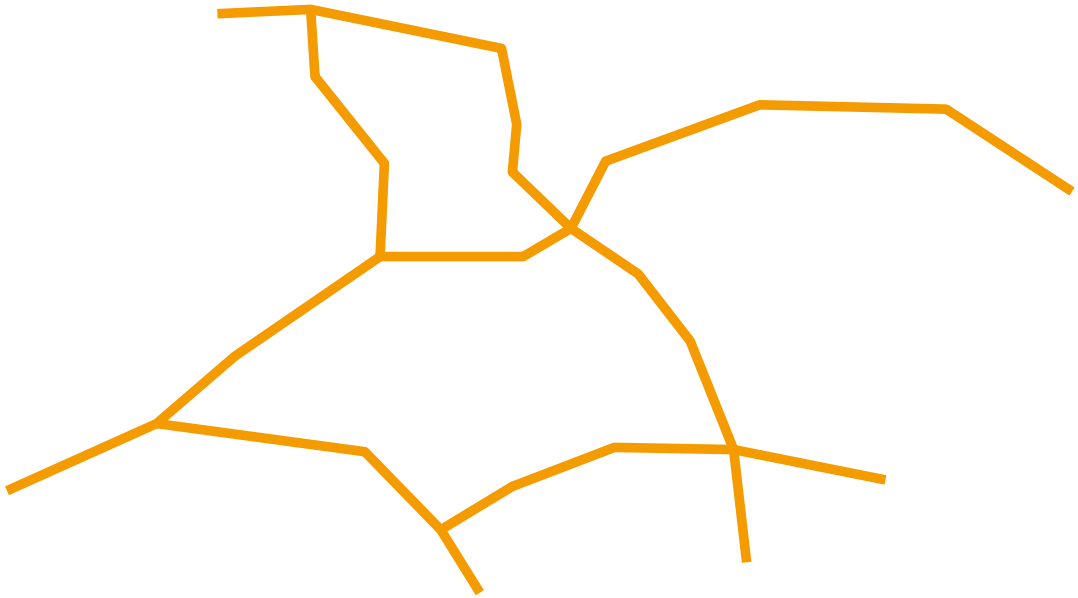


Bewegungsplanungsproblem in der Robotik:
Gegeben eine Menge von Hindernissen, einen Start- und einen Zielpunkt, finde einen kollisionsfreien kürzesten Weg zum Ziel.

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.

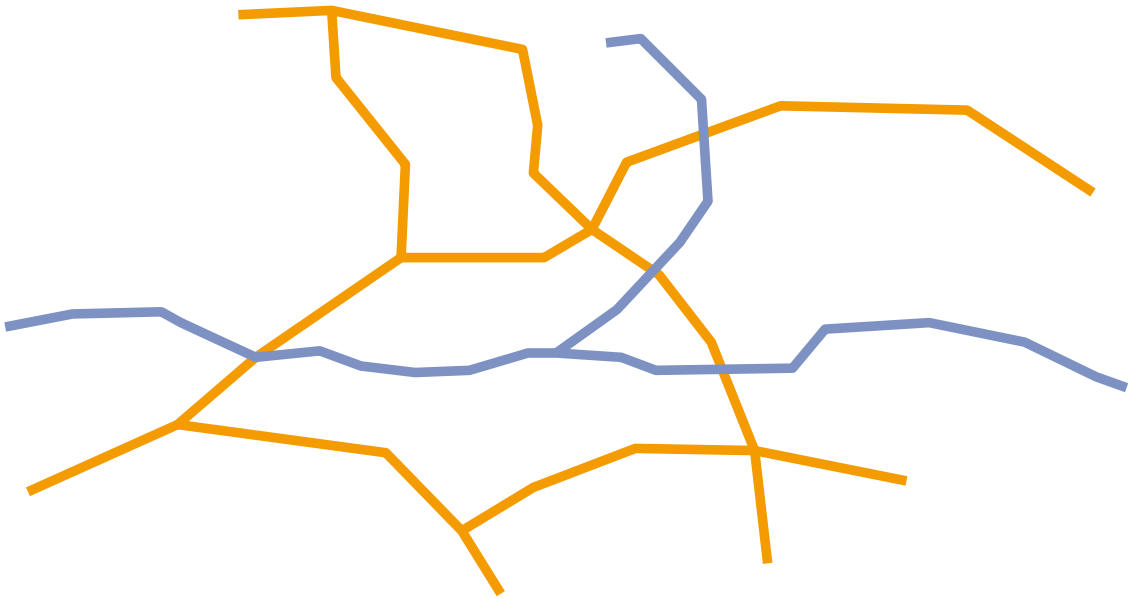
Modelliert man z.B. Straßen und Flüsse als Menge von Strecken und fragt nach den Brücken, muss man alle Schnittpunkte der beiden Streckenmengen finden.



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.

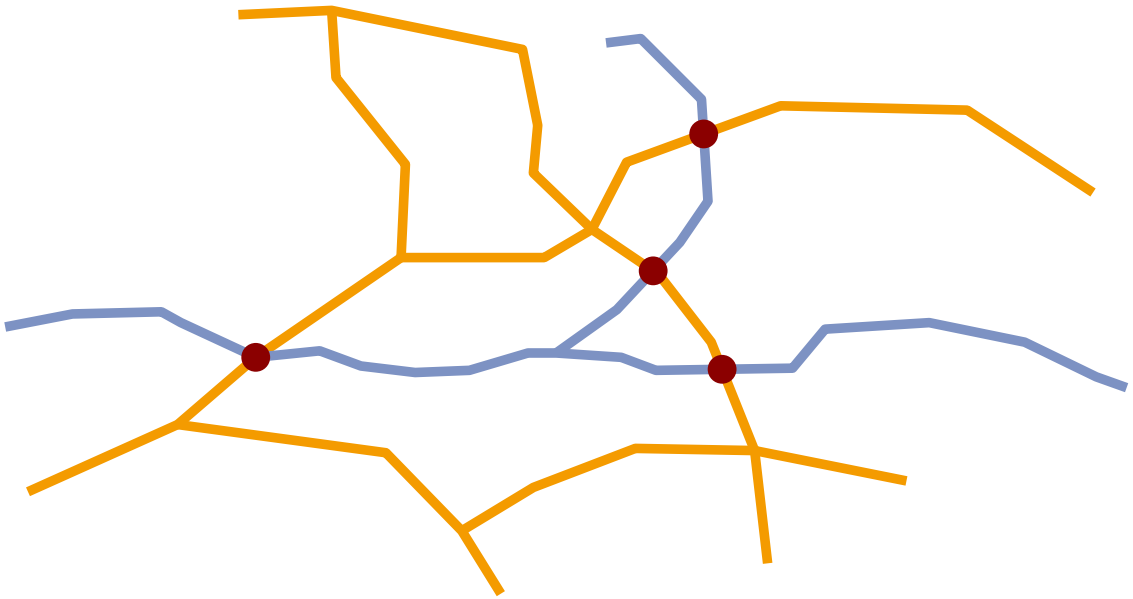
Modelliert man z.B. Straßen und Flüsse als Menge von Strecken und fragt nach den Brücken, muss man alle Schnittpunkte der beiden Streckenmengen finden.



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.

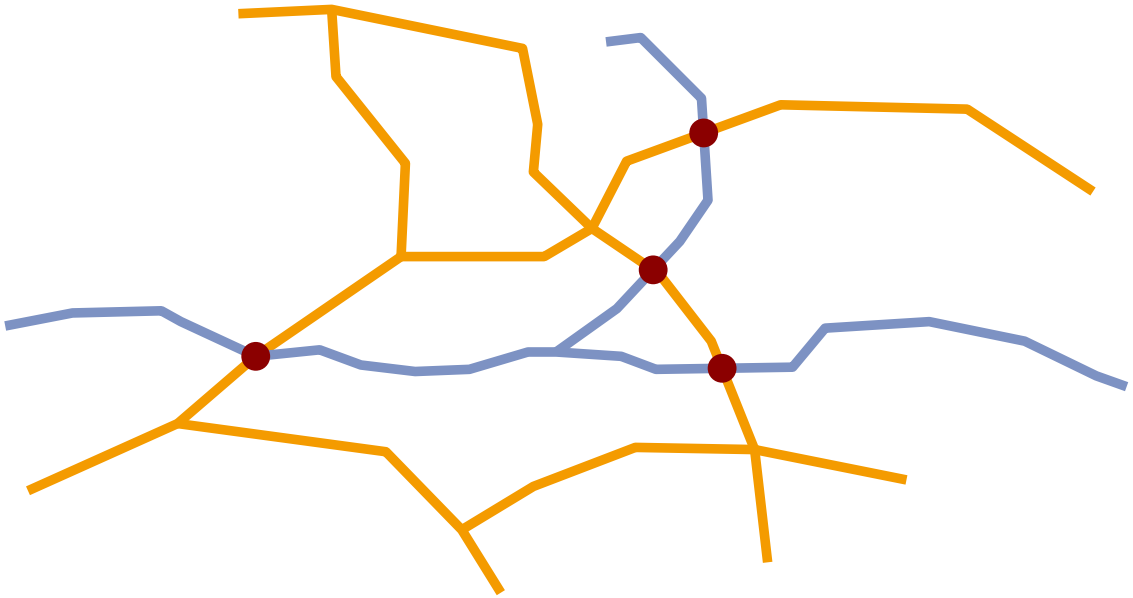
Modelliert man z.B. Straßen und Flüsse als Menge von Strecken und fragt nach den Brücken, muss man alle Schnittpunkte der beiden Streckenmengen finden.



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.

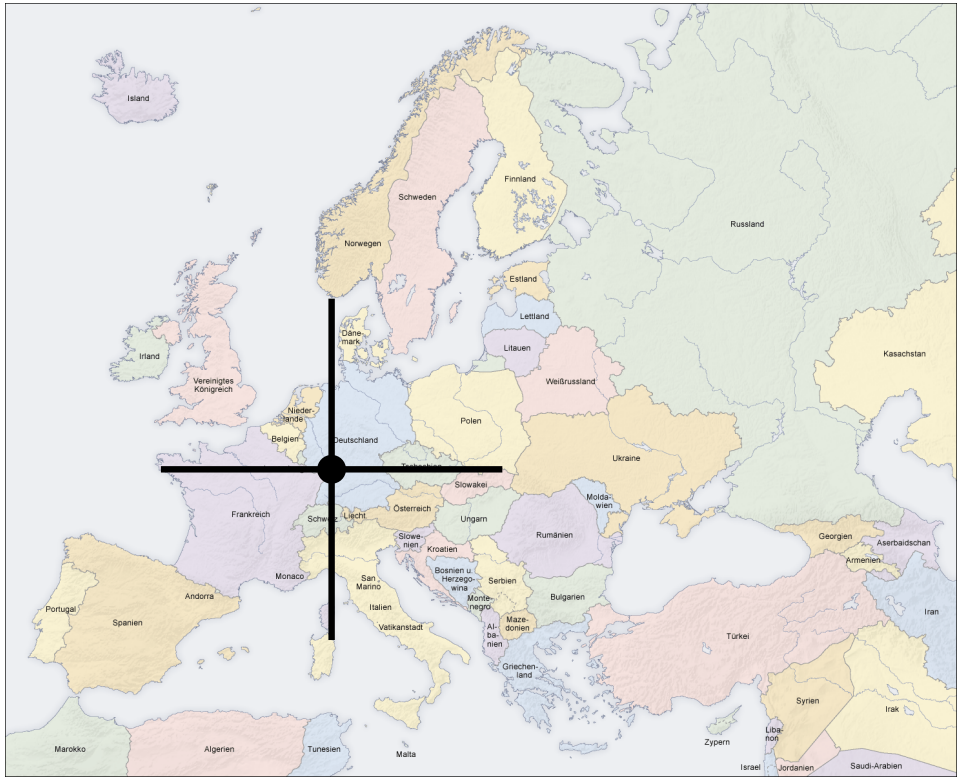
Modelliert man z.B. Straßen und Flüsse als Menge von Strecken und fragt nach den Brücken, muss man alle Schnittpunkte der beiden Streckenmengen finden.



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

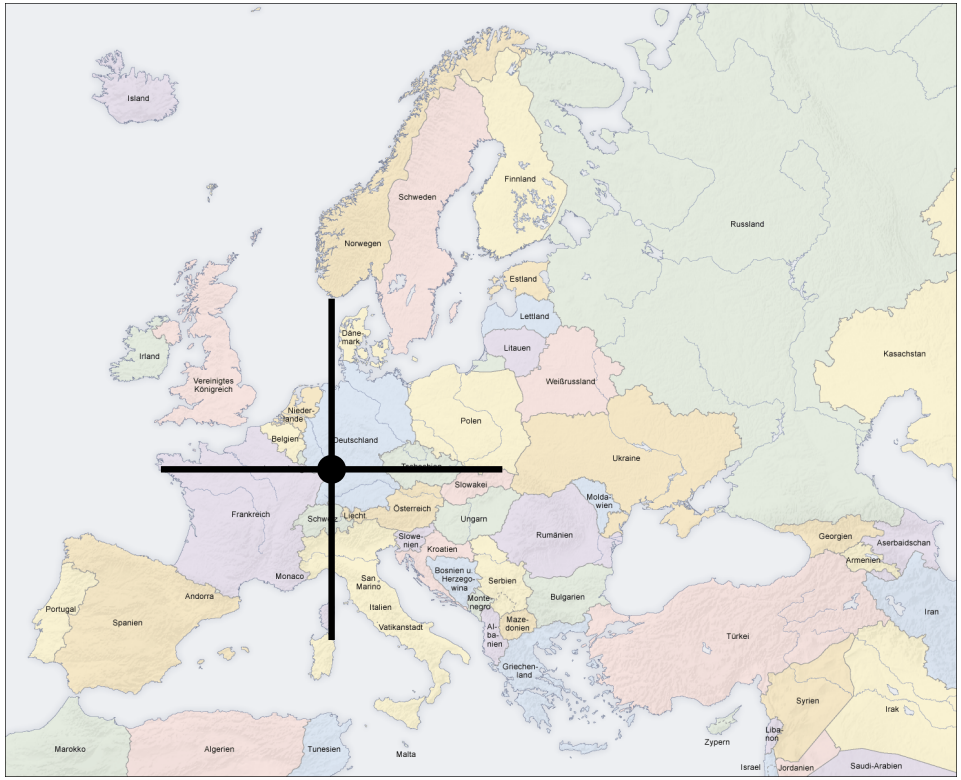
Beispiel 4

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:

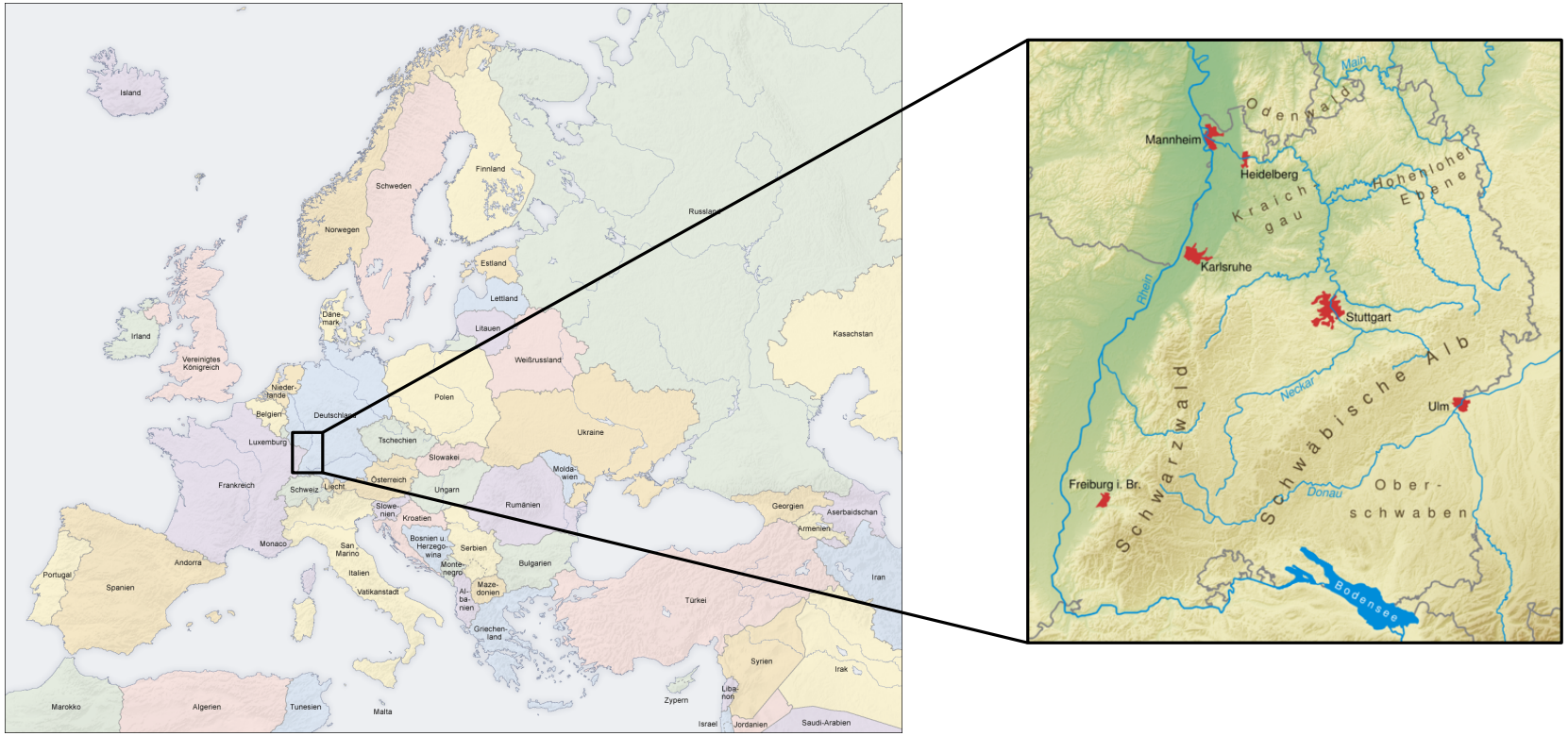
Datenstruktur zur schnellen Beantwortung von Punktanfragen

Beispiel 5

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

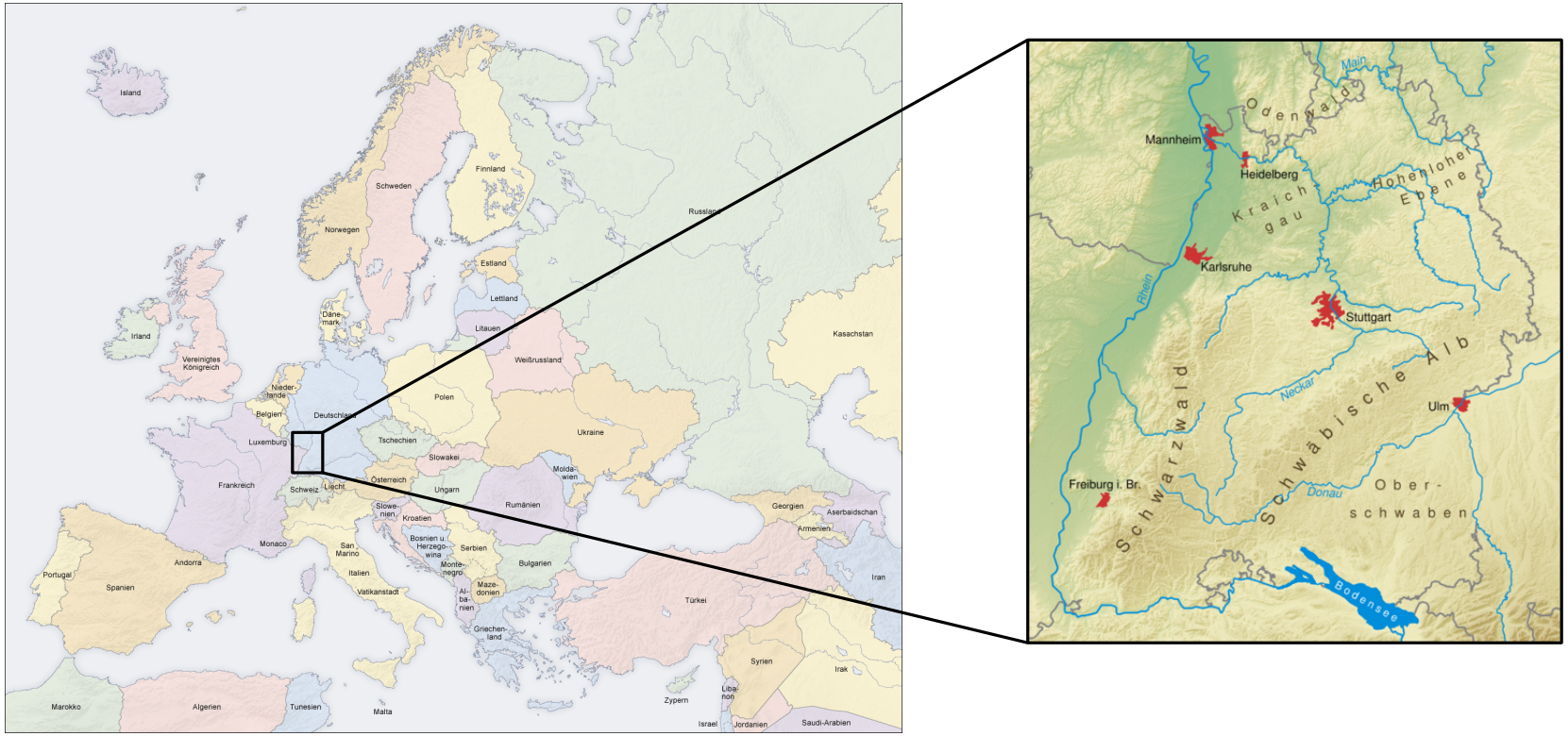
Beispiel 5

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?



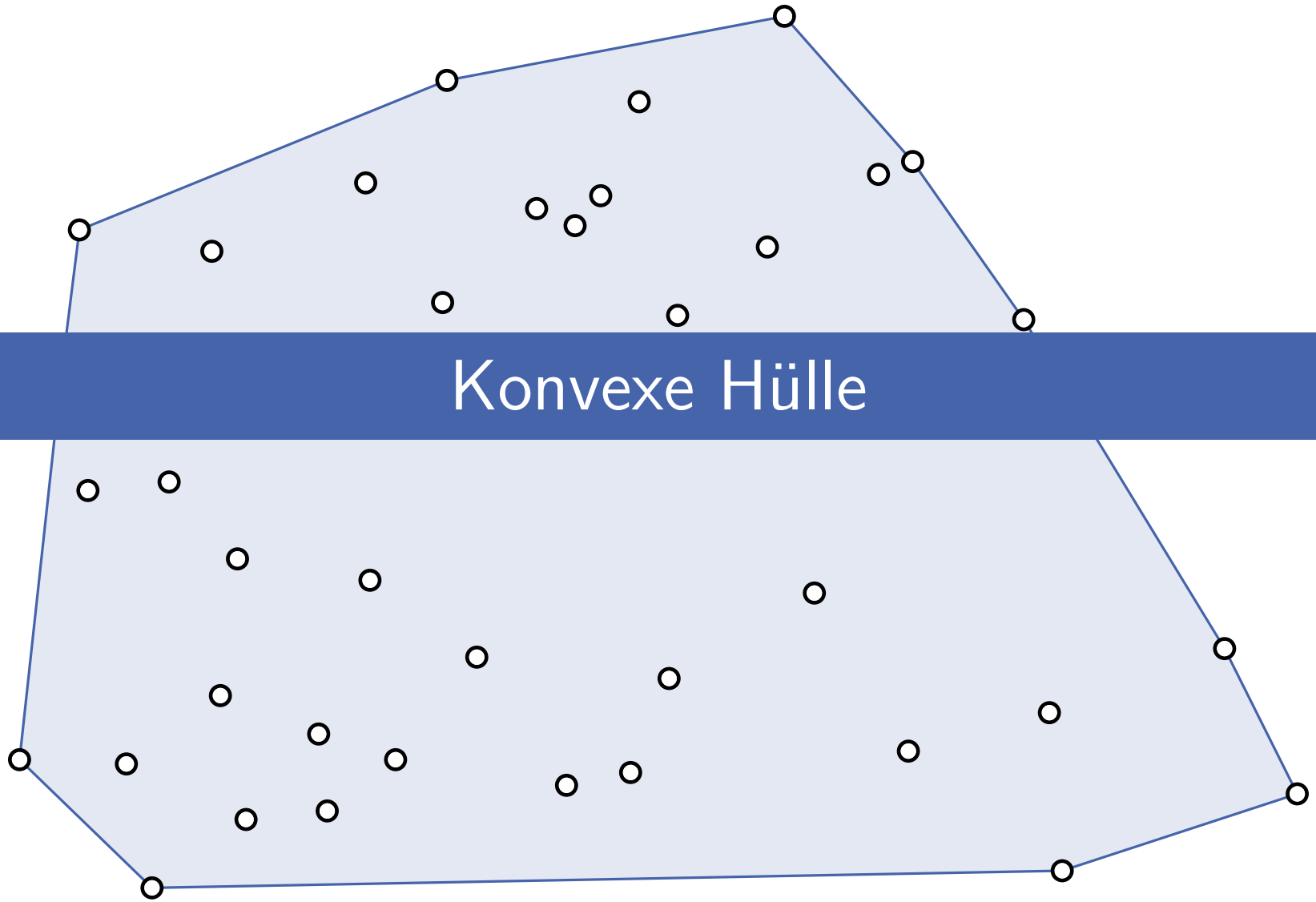
Prüfung für jedes einzelne Kartenobjekt ist unrealistisch.

Ziel: Datenstruktur zur schnellen Beantwortung von Bereichsabfragen

Vorlesungsthemen

Folgende Themen werden wir voraussichtlich behandeln:

- konvexe Hüllen
- Streckenschnitte
- Polygone triangulieren
- lineare Programmierung geometrisch
- Datenstrukturen für Bereichsanfragen
- Datenstrukturen zur Punktlokalisierung
- Voronoi-Diagramm und Delaunay-Triangulierung
- Dualität von Punkten und Geraden
- Quadtrees
- Well-Separated Pair Decompositions
- Sichtbarkeitsgraphen
- ...



Konvexe Hülle

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...

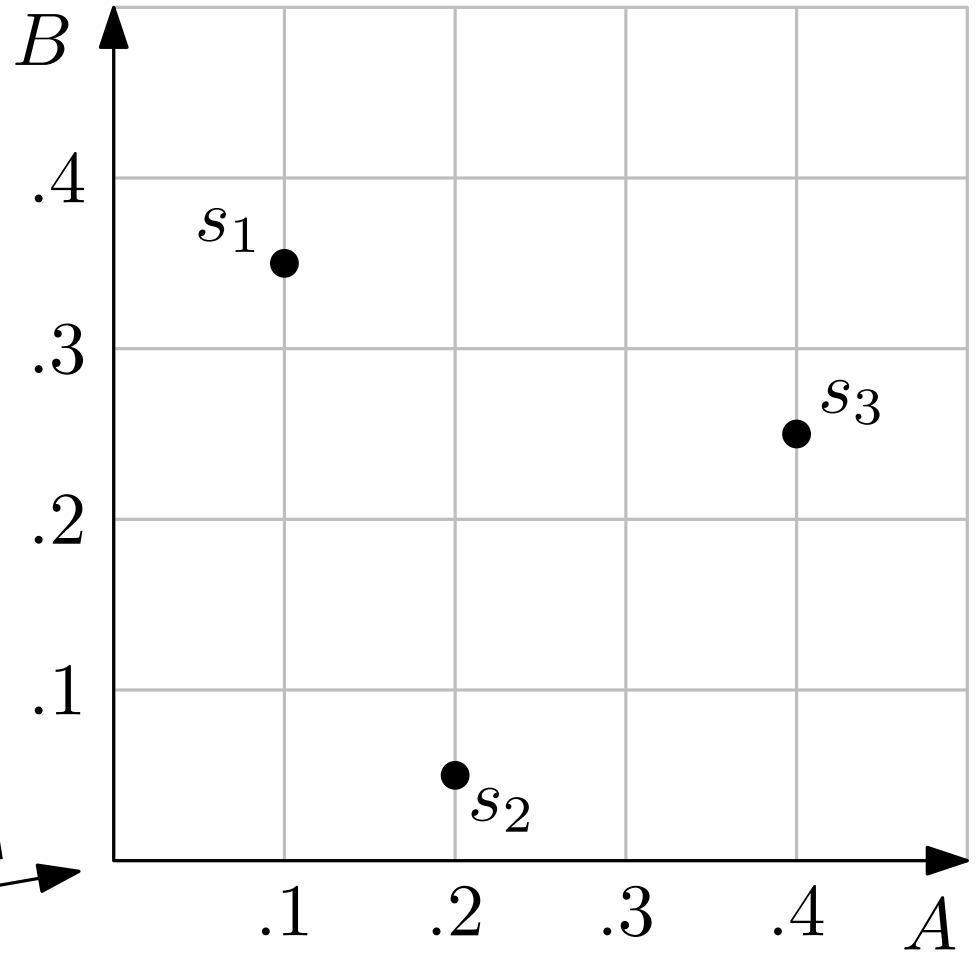
Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

mische folgende Mixturen

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

aus s_1, s_2, s_3?

geom. Interpretation



Mischungsverhältnisse

Gegeben...

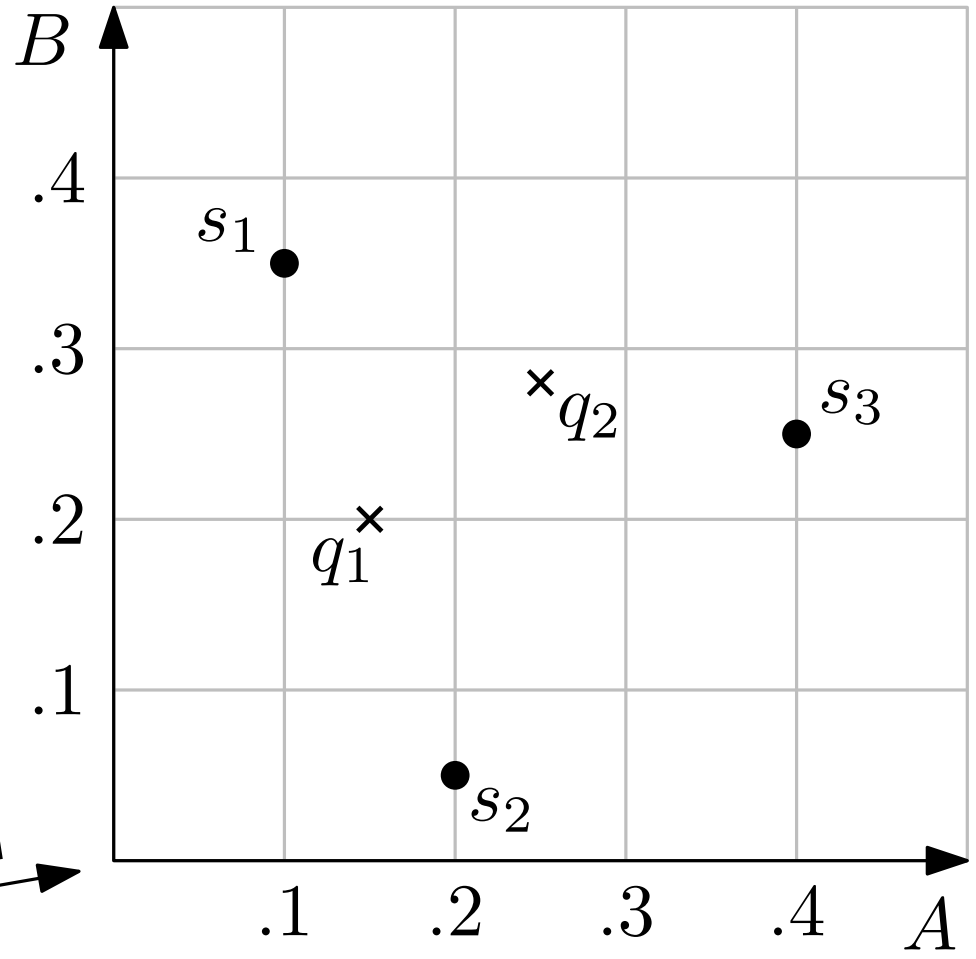
Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

mische folgende Mixturen

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

aus s_1, s_2, s_3?

geom. Interpretation



Mischungsverhältnisse

Gegeben...

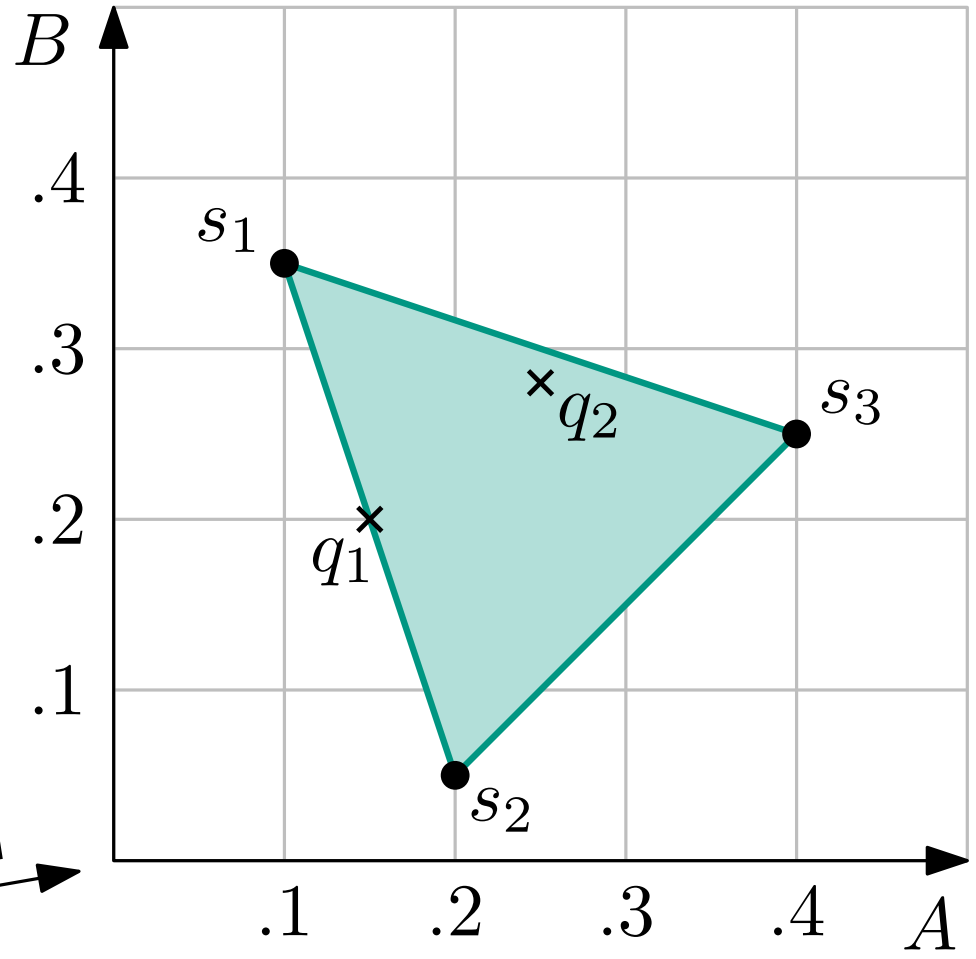
Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

mische folgende Mixturen

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

aus s_1, s_2, s_3?

geom. Interpretation



Mischungsverhältnisse

Gegeben...

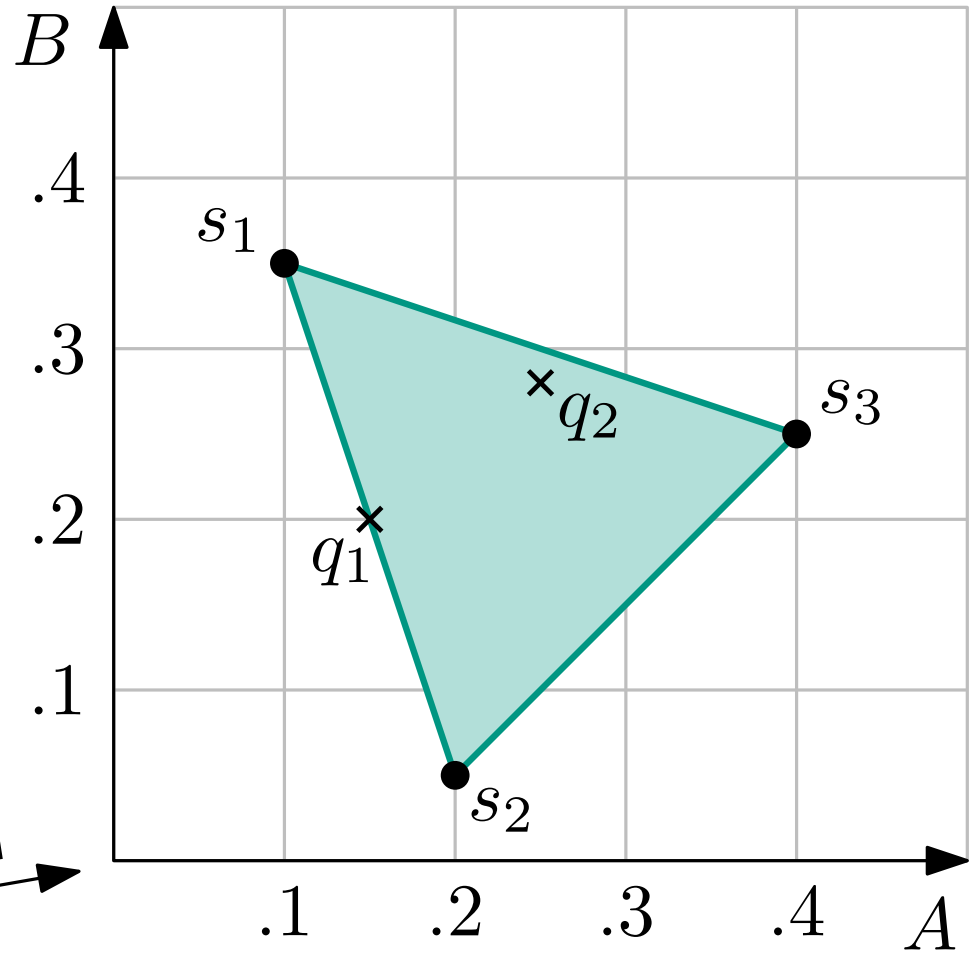
Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

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...

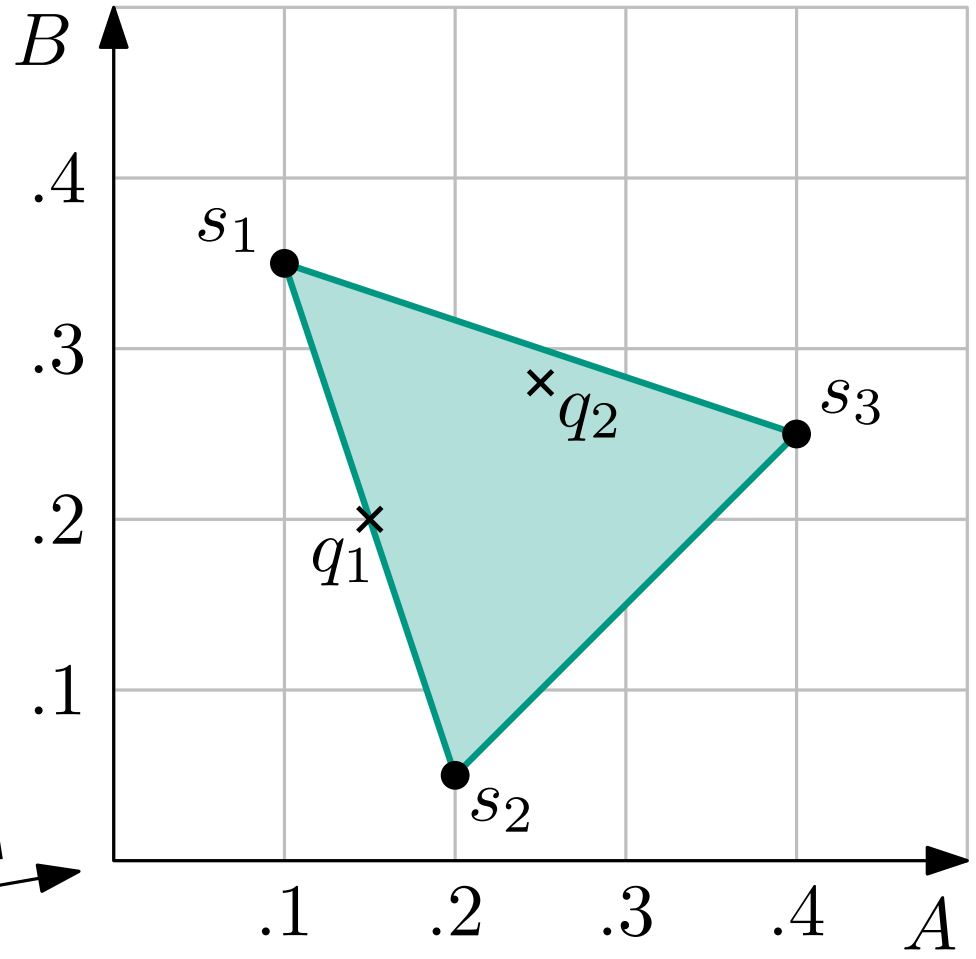
Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

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 $\Leftrightarrow$ $q \in$ konvexer Hülle $CH(S)$.

Mischungsverhältnisse

Gegeben...

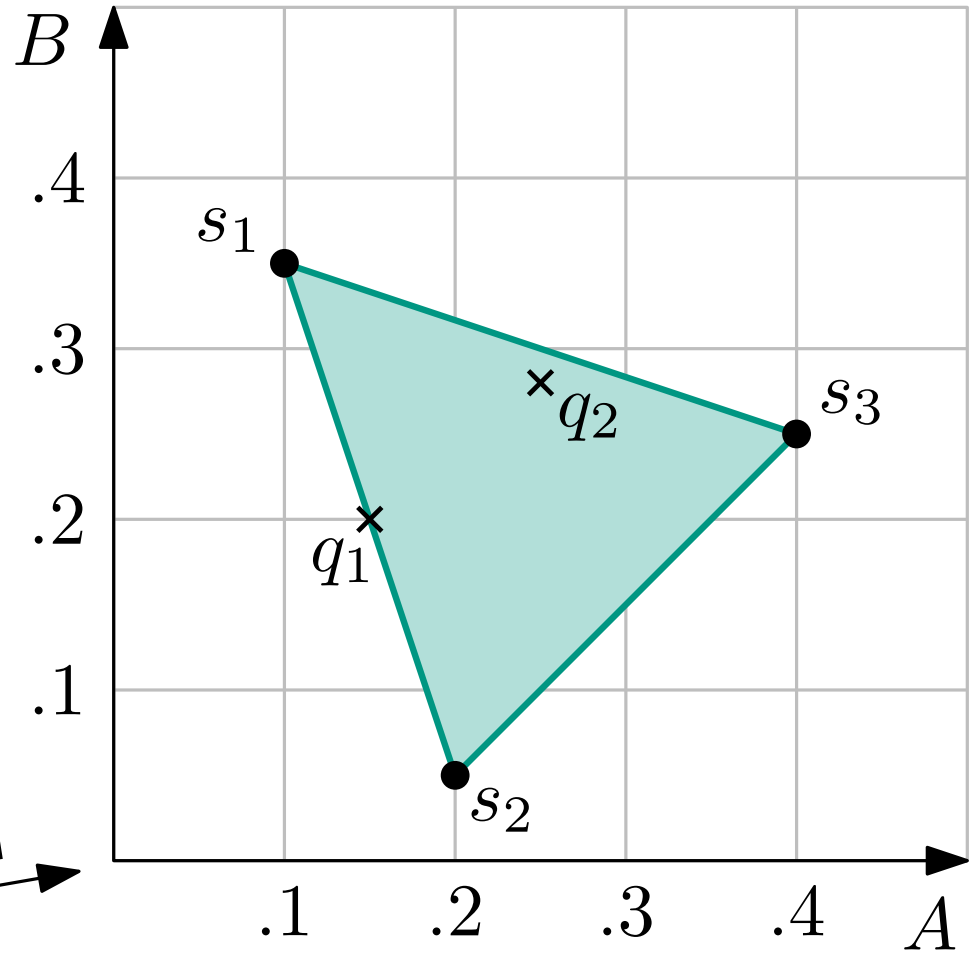
Mixtur	Anteil A	Anteil B
s_1	$10\,\%$	$35\,\%$
s_2	$20\,\%$	$5\,\%$
s_3	$40\,\%$	$25\,\%$

mische folgende Mixturen

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

aus s_1, s_2, s_3?

geom. Interpretation



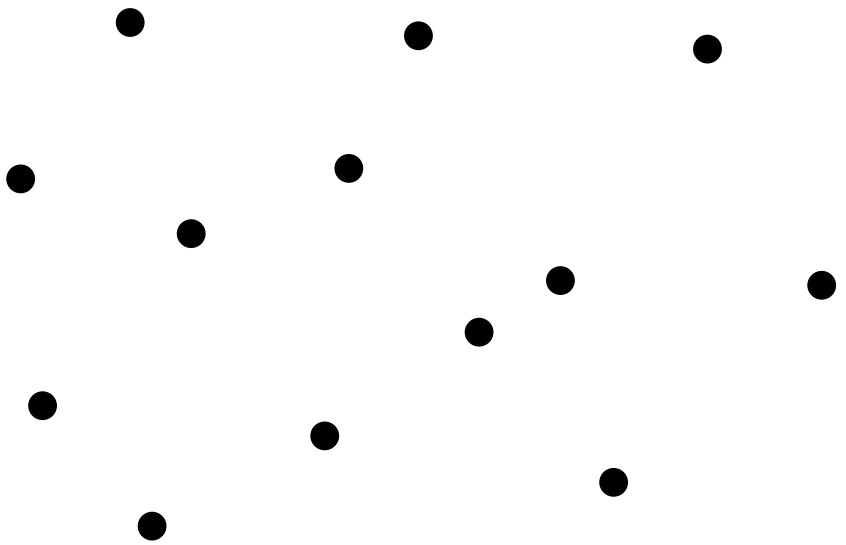
Beob: Gegeben eine Menge $S \subset \mathbb{R}^{2d}$ von Mixturen, kann man eine weitere Mixtur $q \in \mathbb{R}^{2d}$ aus S mischen $\Leftrightarrow$ $q \in$ konvexer Hülle $CH(S)$.

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

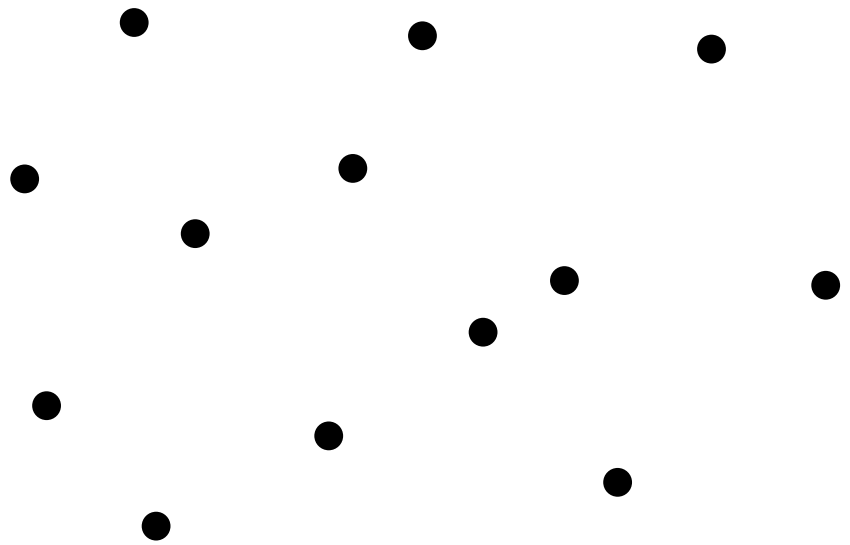
Def: Eine Menge $S \subseteq \mathbb{R}^2$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.



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:

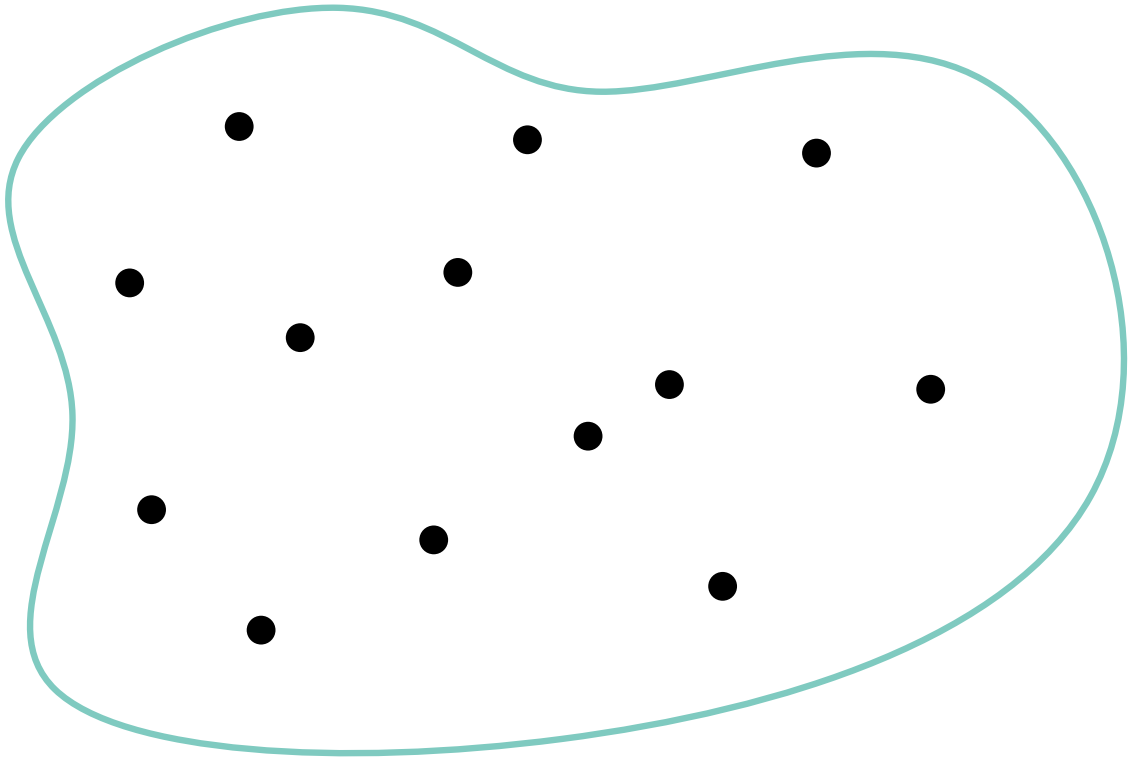


Definition Konvexe Hülle

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

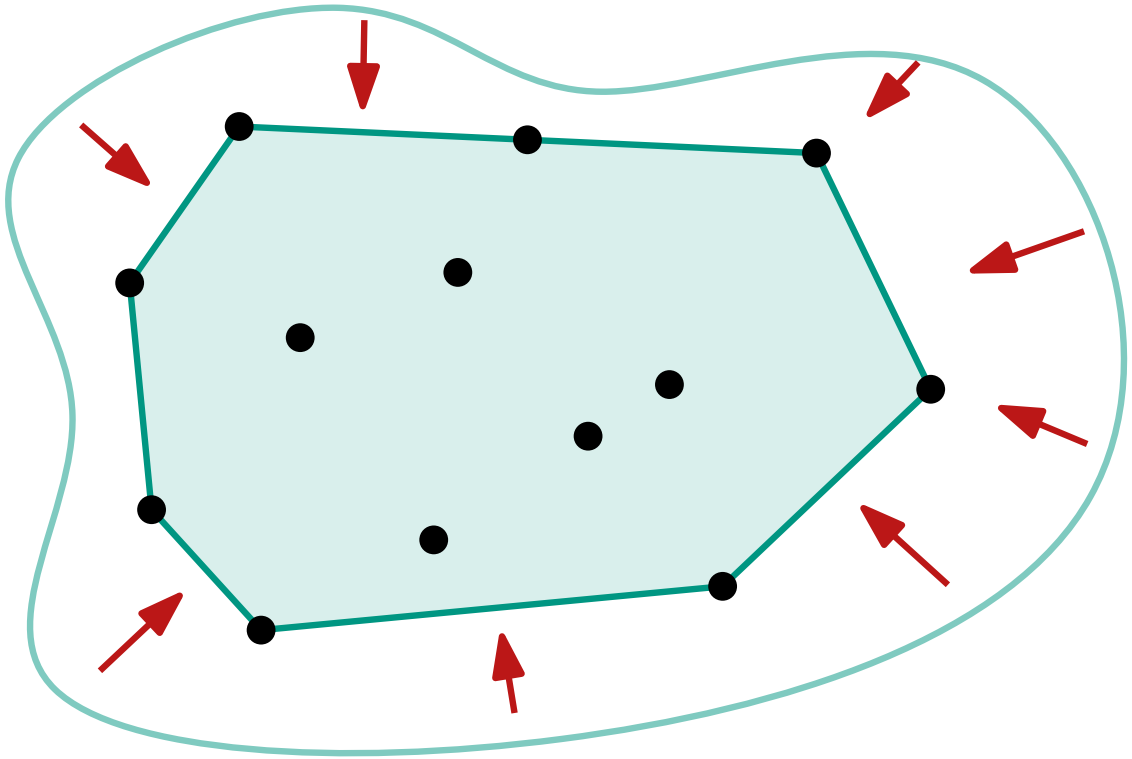


Definition Konvexe Hülle

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!

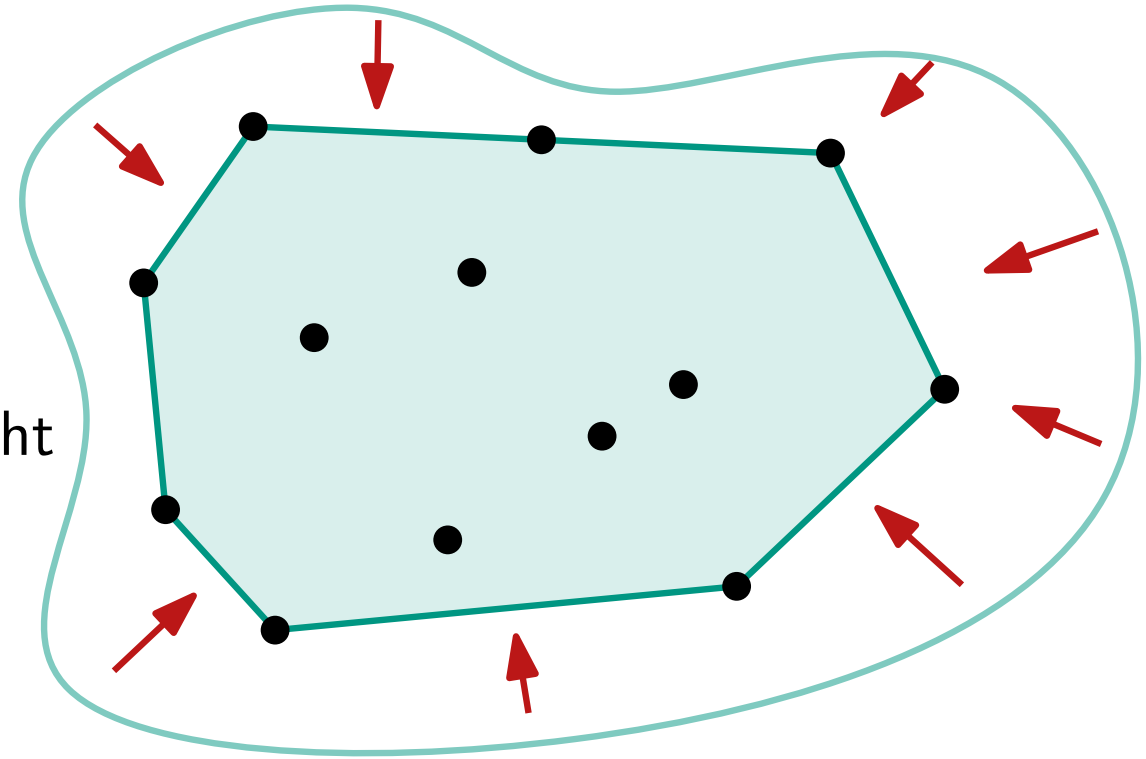


Definition Konvexe Hülle

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

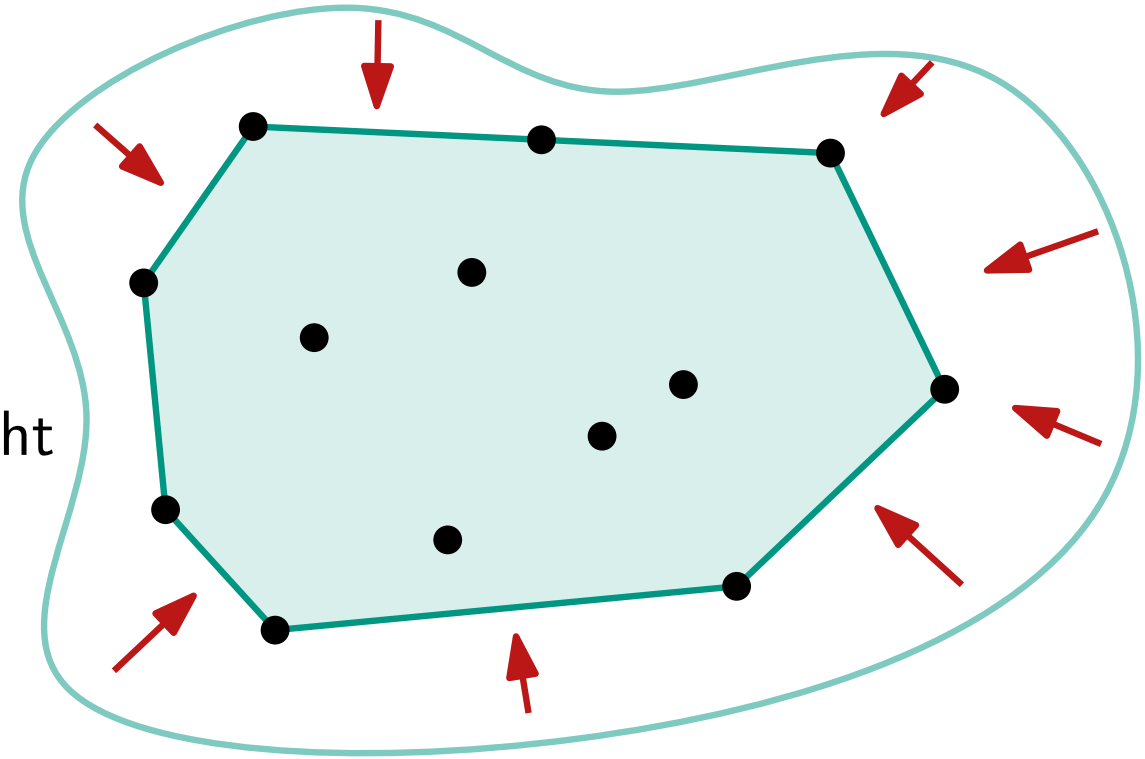


Definition Konvexe Hülle

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



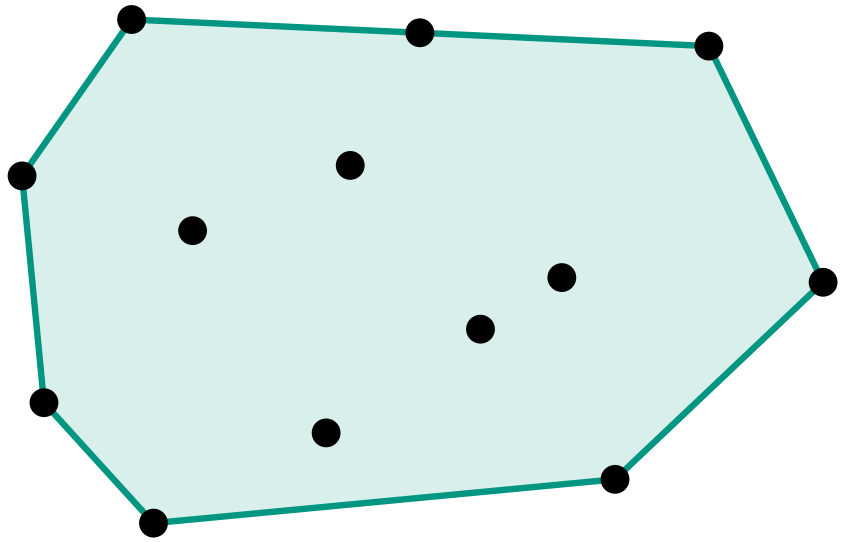
In der Mathematik:

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

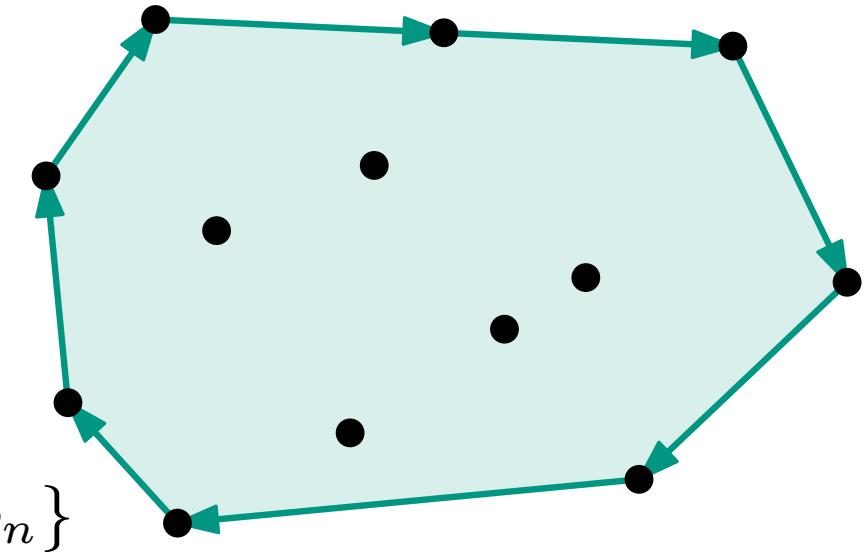
Algorithmischer Ansatz

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



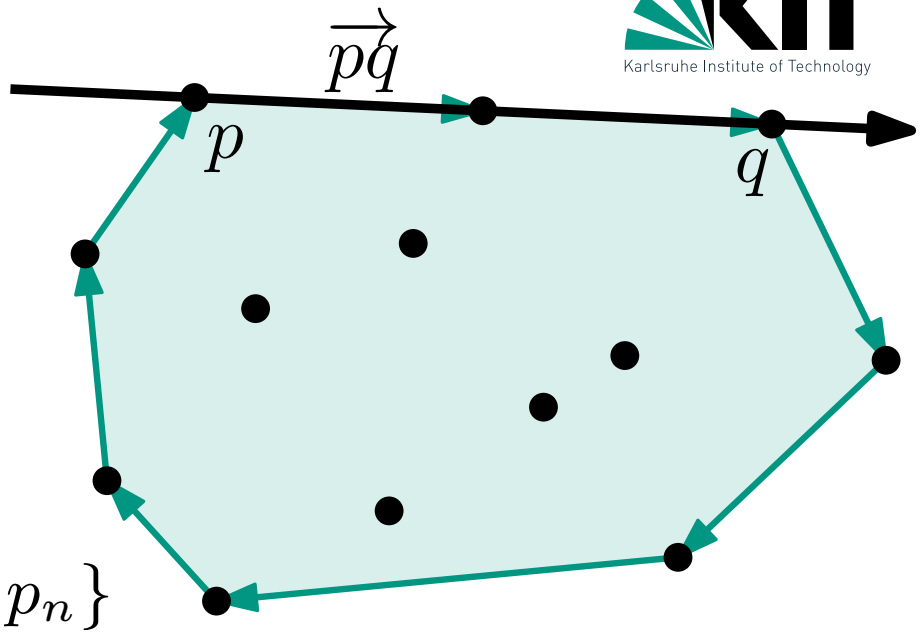
Lemma:

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 $O(n^2)$

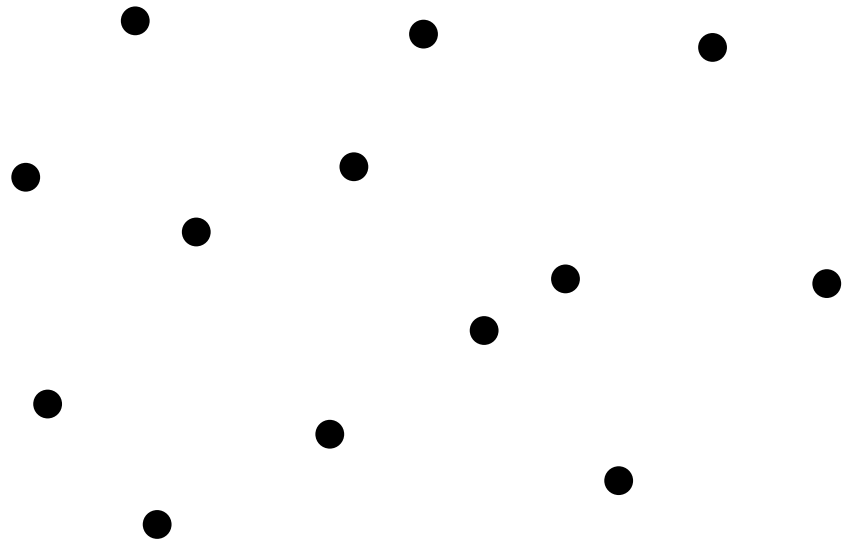
return L

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?

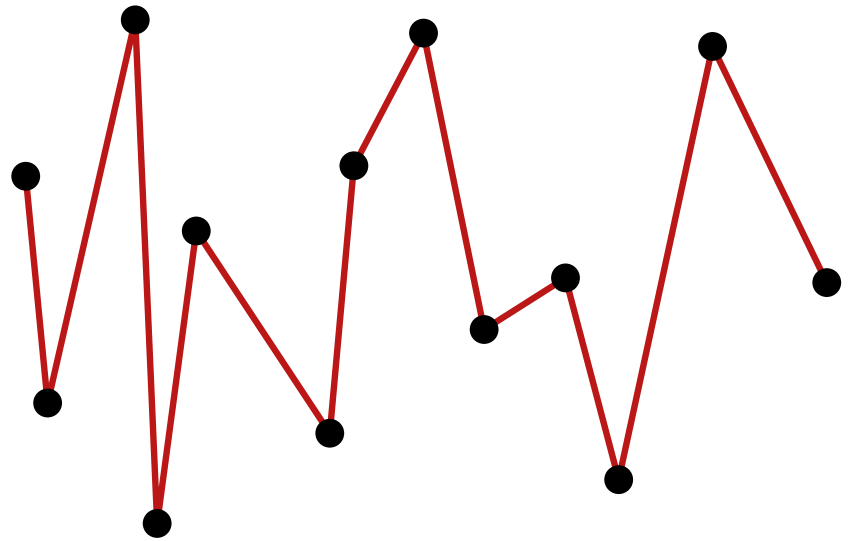


Inkrementeller Ansatz

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!



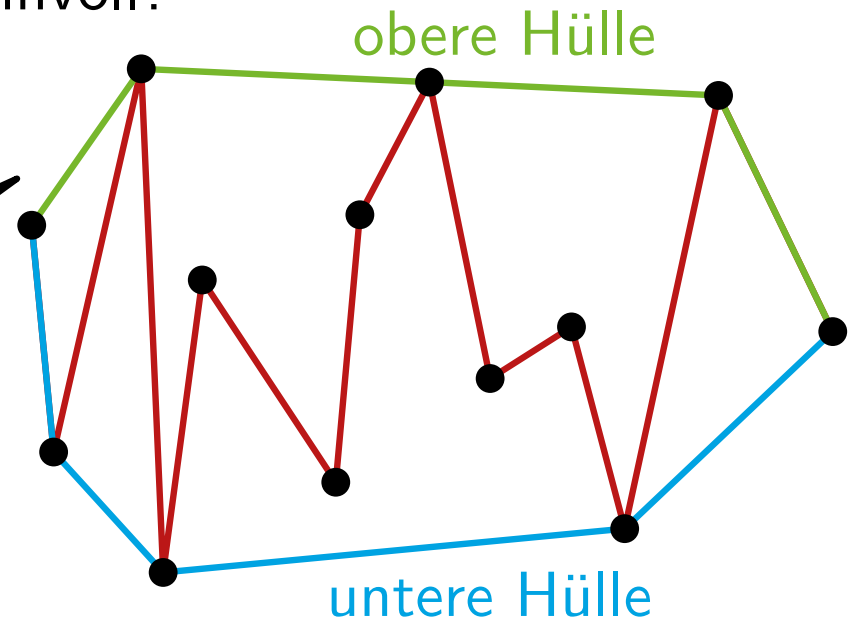
Inkrementeller Ansatz

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



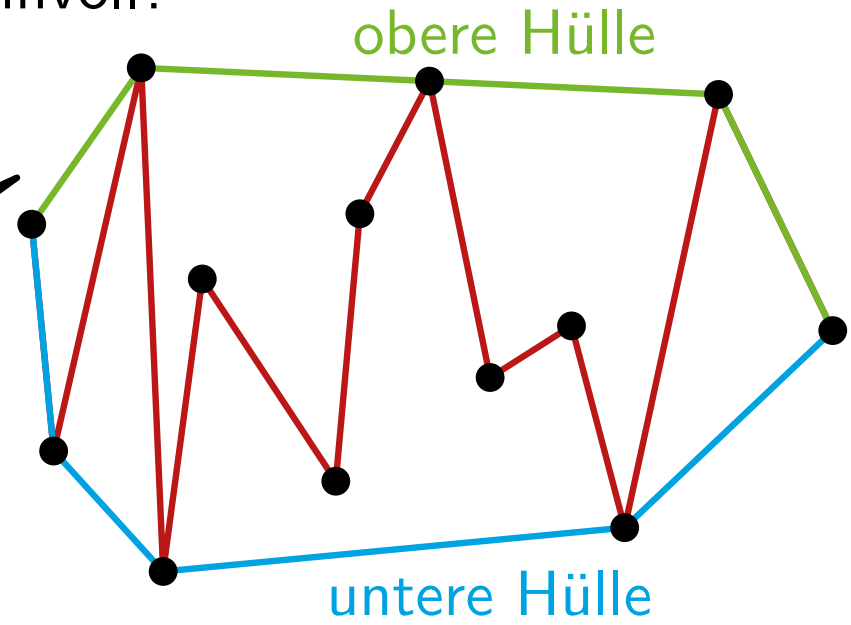
Inkrementeller Ansatz

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

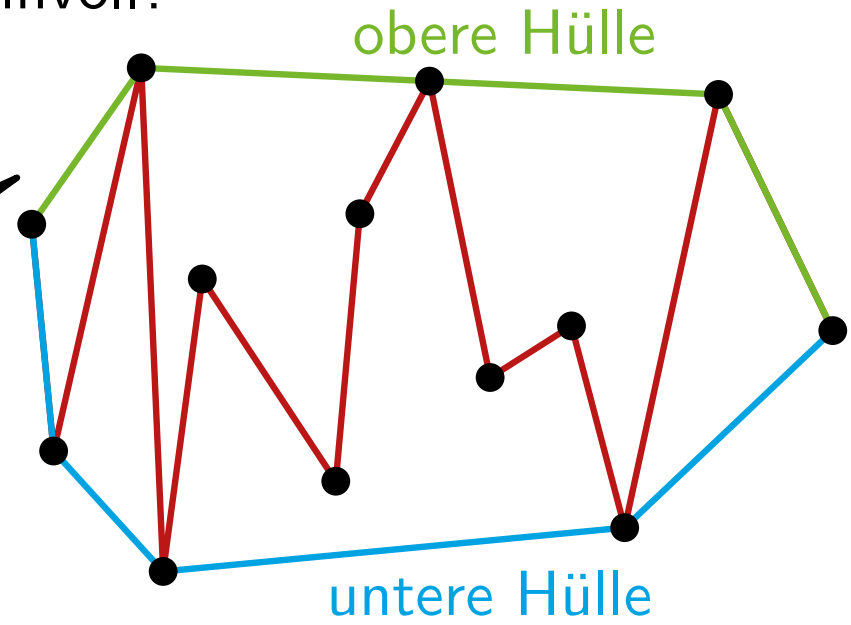
Inkrementeller Ansatz

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

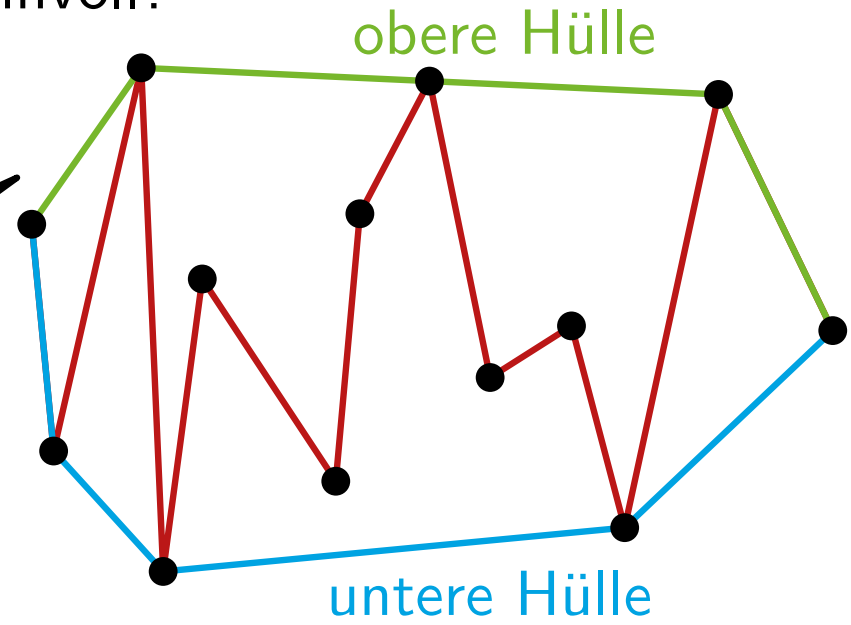
Inkrementeller Ansatz

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 $\quad O(n \log n)$

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

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**

    entferne vorletzten Punkt aus L

~~?~~ $O(n)$

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 $\quad O(n \log n)$

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

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 $O(n)$

Satz 1: Die konvexe Hülle von n Punkten in der Ebene lässt sich mit *Graham's Scan* in $O(n \log n)$ Zeit berechnen.

Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

$p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$

Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

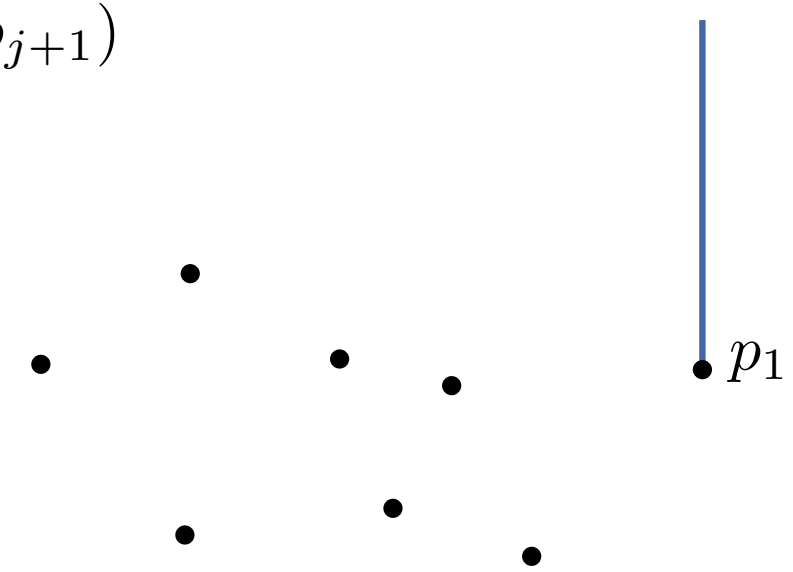
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

$p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

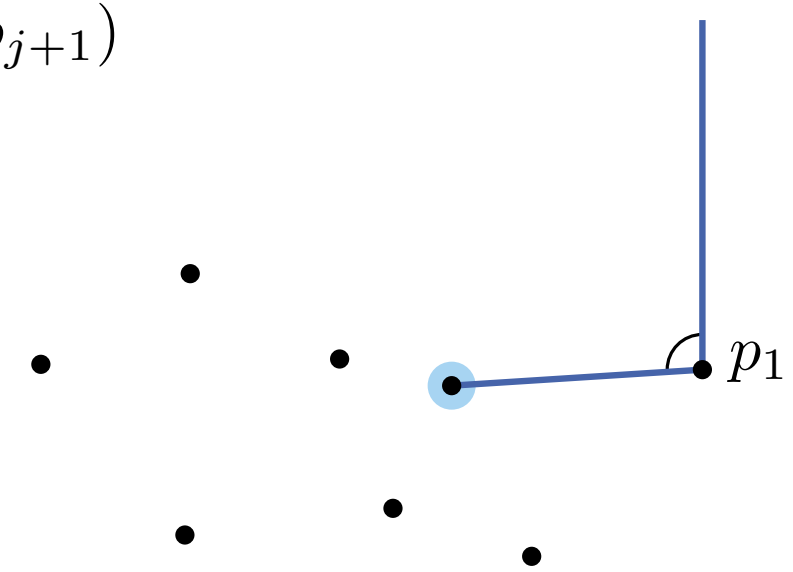
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

$p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

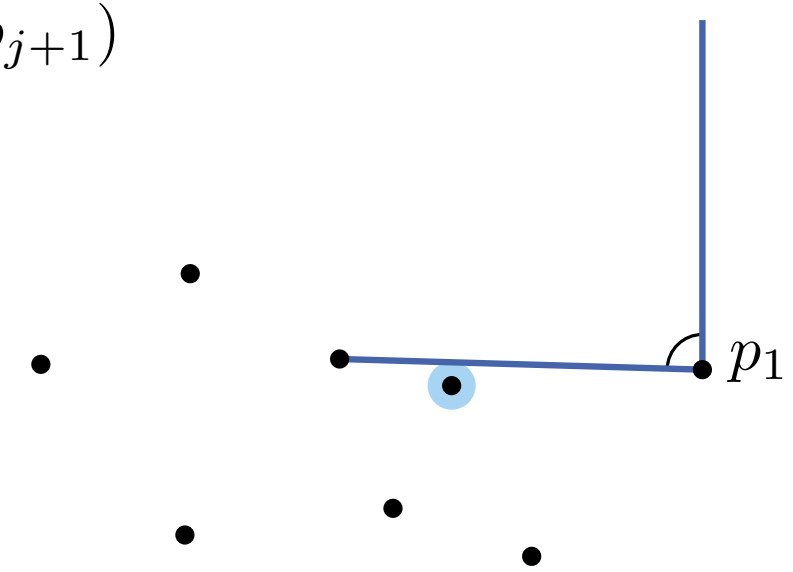
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

$p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

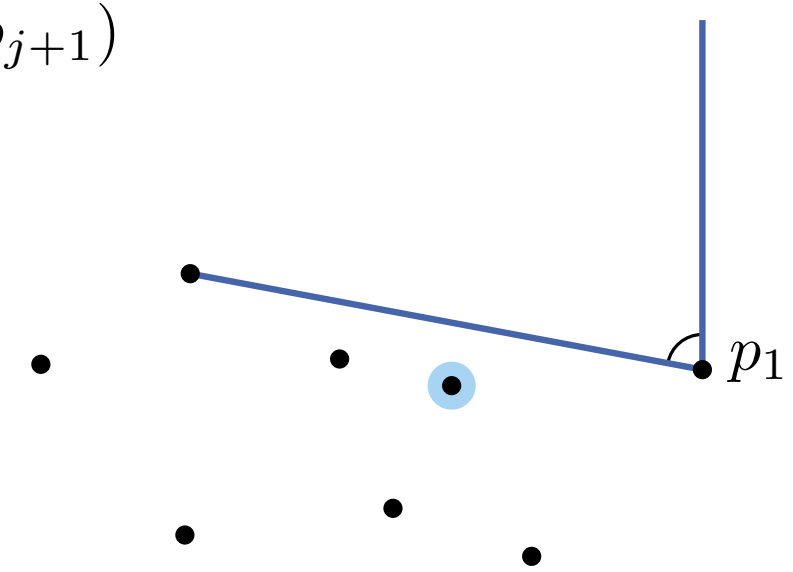
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

$p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

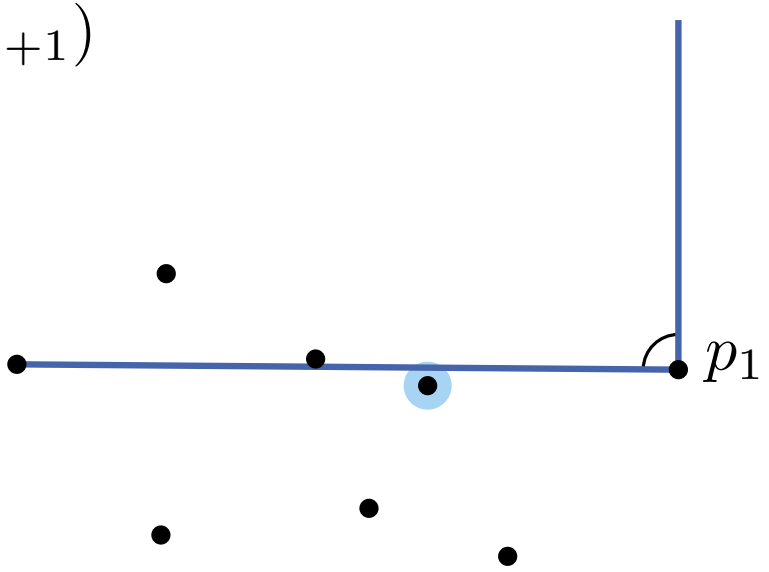
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

$p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

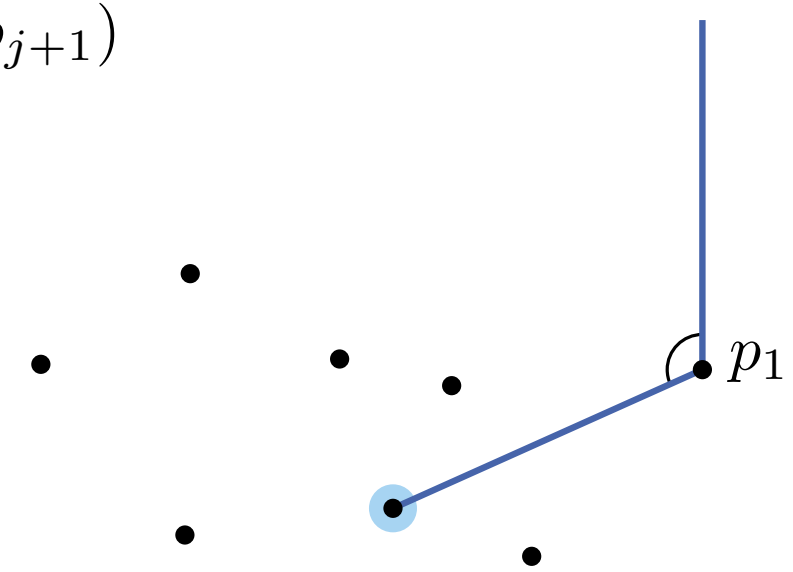
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

$p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

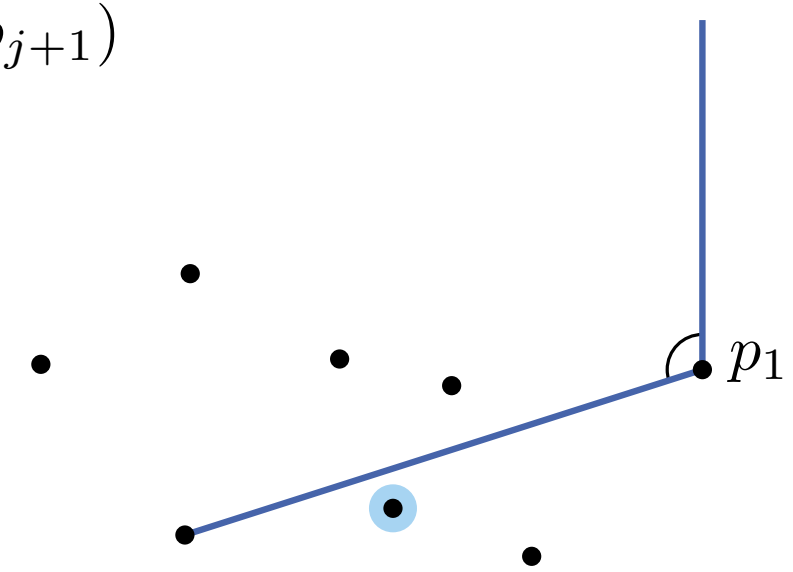
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

$p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

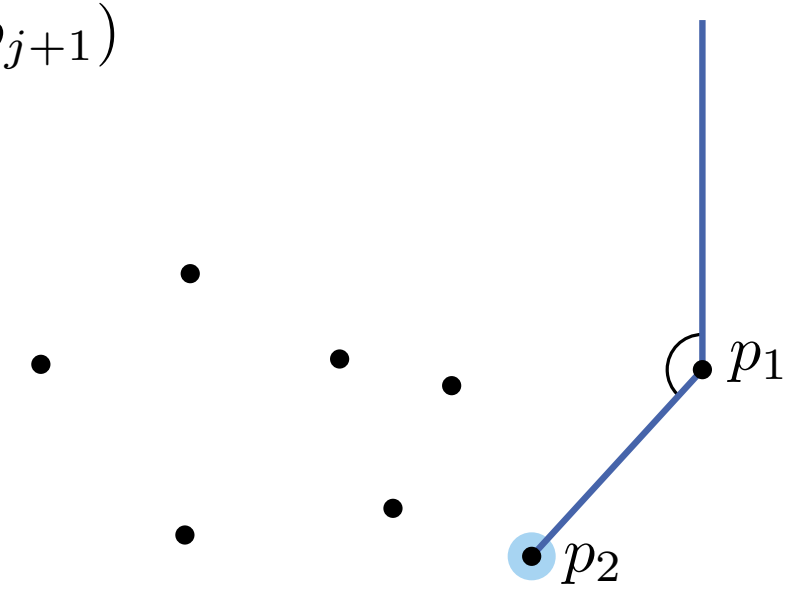
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

$p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

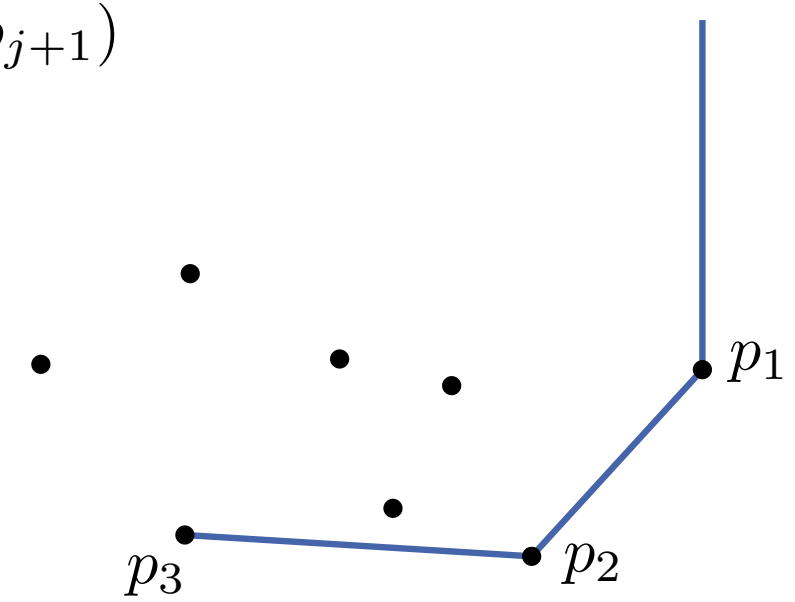
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

$p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

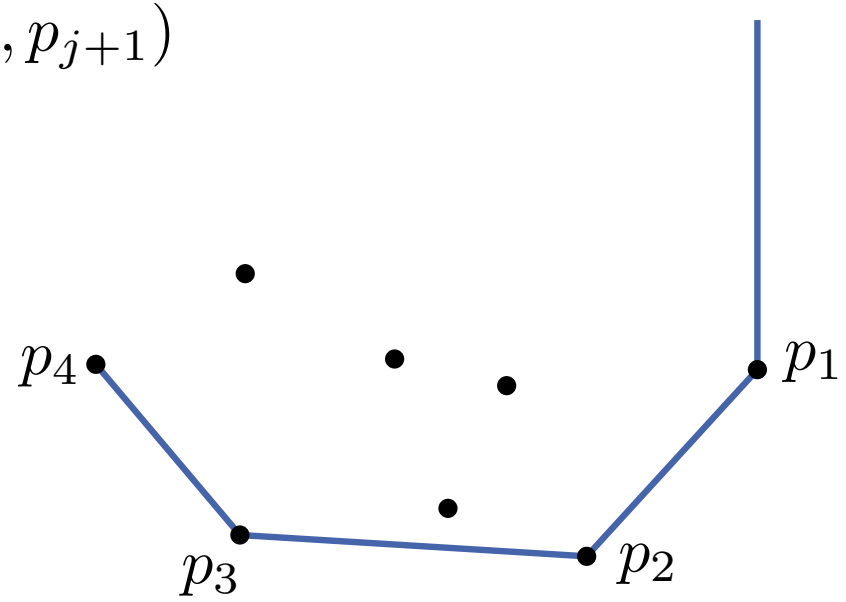
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

$p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

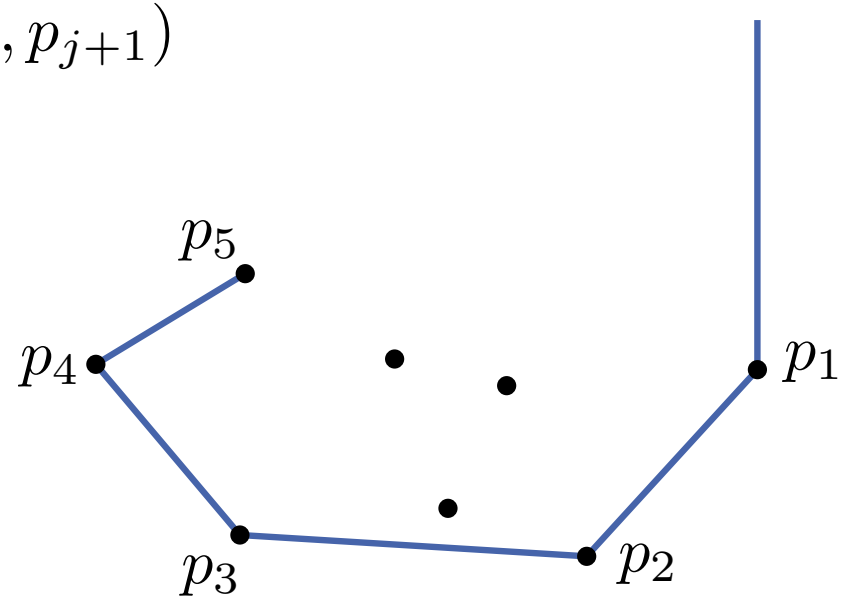
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

$p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

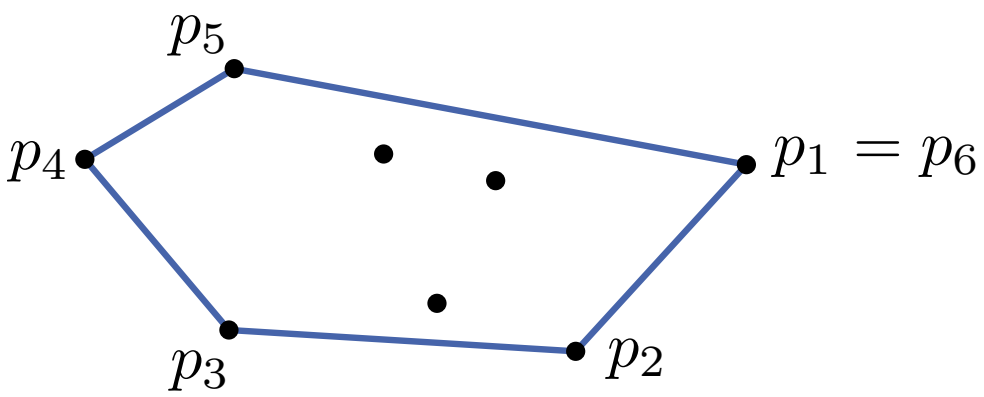
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

$p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

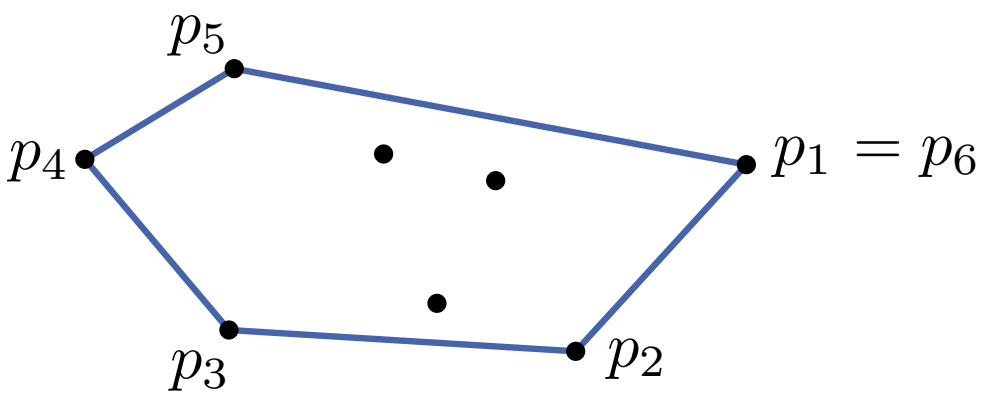
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$

  if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

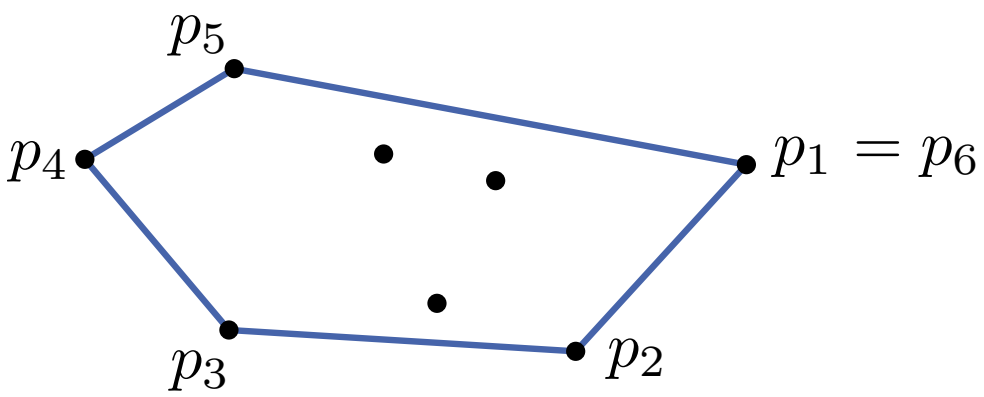
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do**

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$ $O(n)$

  if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

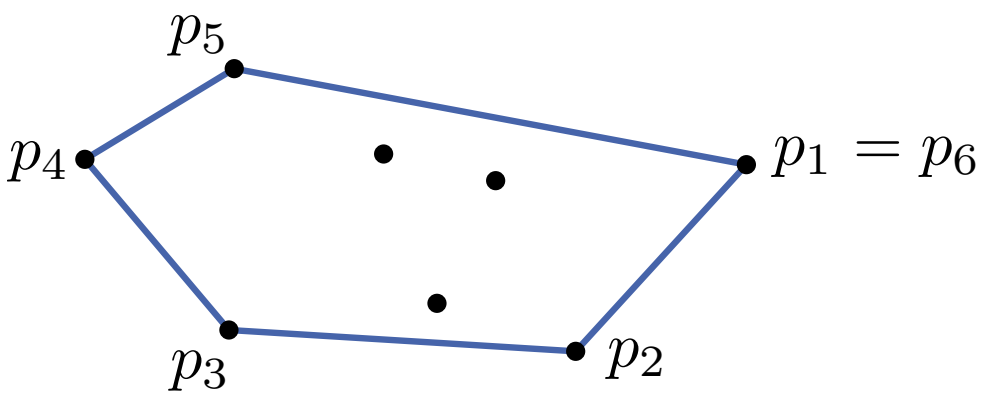
$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$

while true **do** — $O(h)$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$ — $O(n)$

  if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$ — $O(n)$

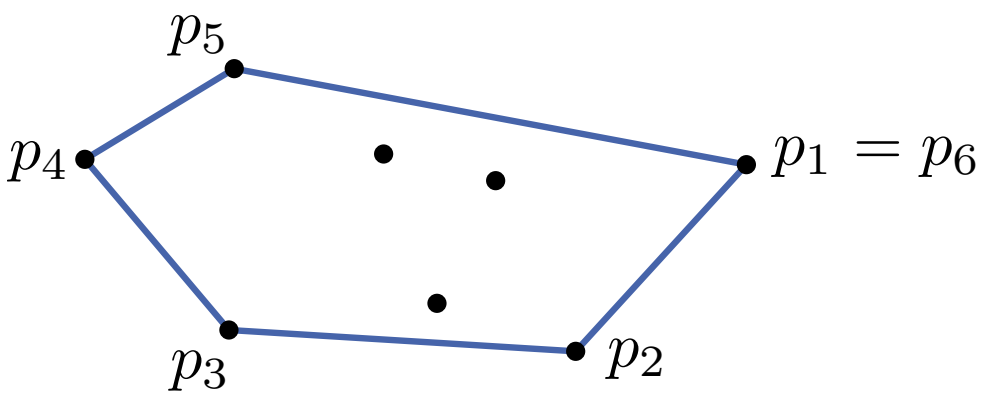
while true **do** — $O(h)$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$ — $O(n)$

  if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

return $(p_1, \ldots, p_{j+1})$

$O(n \cdot h)$



Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$ — $O(n)$

while true **do** — $O(h)$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$ — $O(n)$

  if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

— $O(n \cdot h)$

return $(p_1, \ldots, p_{j+1})$

Satz 2: Die konvexe Hülle $CH(P)$ von n Punkten P in $\mathbb{R}^2$ lässt sich mit *Gift Wrapping* (auch *Jarvis' March*) in $O(n \cdot h)$ Zeit berechnen, wobei $h = |CH(P)|$.

Alternativer Ansatz: Gift Wrapping

Idee: beginnend mit einem Punkt p_1 von $CH(P)$ finde nächste Kante von $CH(P)$ im UZS

GiftWrapping(P)

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P; $p_0 \leftarrow (x_1, \infty)$; $j \leftarrow 1$ — $O(n)$

while true **do** — $O(h)$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1}, p_j, q \mid q \in P \setminus \{p_{j-1}, p_j\}\}$ — $O(n)$

  if $p_{j+1} = p_1$ **then** break **else** $j \leftarrow j + 1$

— $O(n \cdot h)$

return $(p_1, \ldots, p_{j+1})$

Satz 2: Die konvexe Hülle $CH(P)$ von n Punkten P in $\mathbb{R}^2$ lässt sich mit *Gift Wrapping* (auch *Jarvis' March*) in $O(n \cdot h)$ Zeit berechnen, wobei $h = |CH(P)|$.

$\rightarrow$ mehr dazu in der Übung!

Bewertung

Welcher Algorithmus ist besser?

- Graham's Scan: $O(n \log n)$ Zeit
- Jarvis' March: $O(n \cdot h)$ Zeit

Bewertung

Welcher Algorithmus ist besser?

- Graham's Scan: $O(n \log n)$ Zeit
- Jarvis' March: $O(n \cdot h)$ Zeit

Es kommt darauf an wie groß $CH(P)$ ist!

Bewertung

Welcher Algorithmus ist besser?

- Graham's Scan: $O(n \log n)$ Zeit
- Jarvis' March: $O(n \cdot h)$ Zeit

Es kommt darauf an wie groß $CH(P)$ ist!

Idee: Kombiniere beide Ansätze für optimalen Algorithmus!

Chans Algorithmus

Angenommen, wir kennen h:

ChanHull(P,h)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do**

  Berechne per GrahamScan $CH(P_i)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** h **do**

  for $i = 1$ **to** $\lceil n/h \rceil$ **do**

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$

return $(p_1, \ldots, p_h)$

Chans Algorithmus

Karlsruhe Institute of Technology

Angenommen, wir kennen h:

ChanHull(P,h)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do**
  Berechne per GrahamScan $CH(P_i)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P
$p_0 \leftarrow (x_1, \infty)$
for $j = 1$ **to** h **do**
  for $i = 1$ **to** $\lceil n/h \rceil$ **do**
    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$
return $(p_1, \ldots, p_h)$

Chans Algorithmus

Karlsruhe Institute of Technology

Angenommen, wir kennen h:

ChanHull(P,h)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do**
  Berechne per GrahamScan $CH(P_i)$

GrahamScan

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

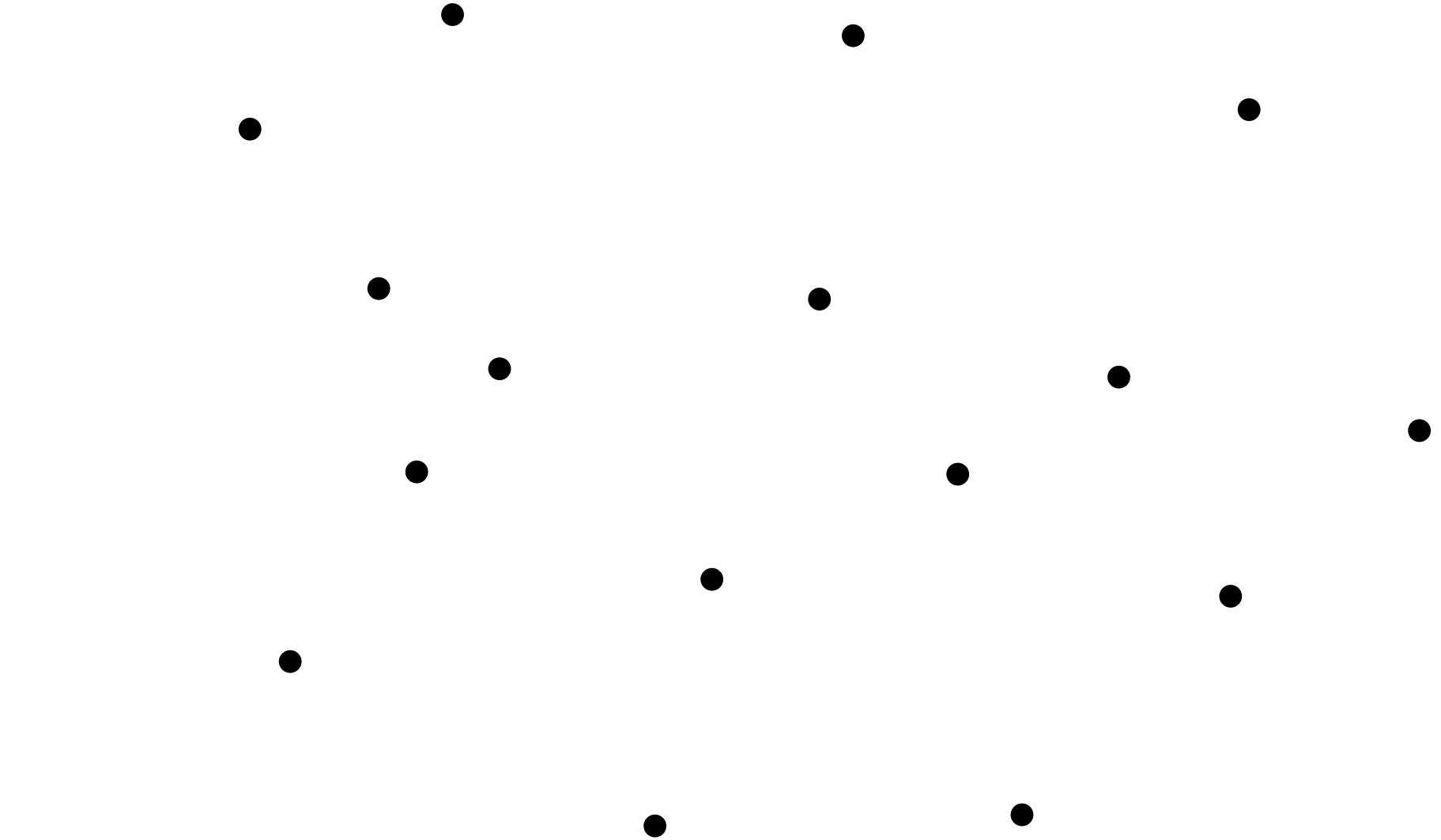
for $j = 1$ **to** h **do**
  for $i = 1$ **to** $\lceil n/h \rceil$ **do**
    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$

Gift Wrapping

return $(p_1, \ldots, p_h)$

Beispiel

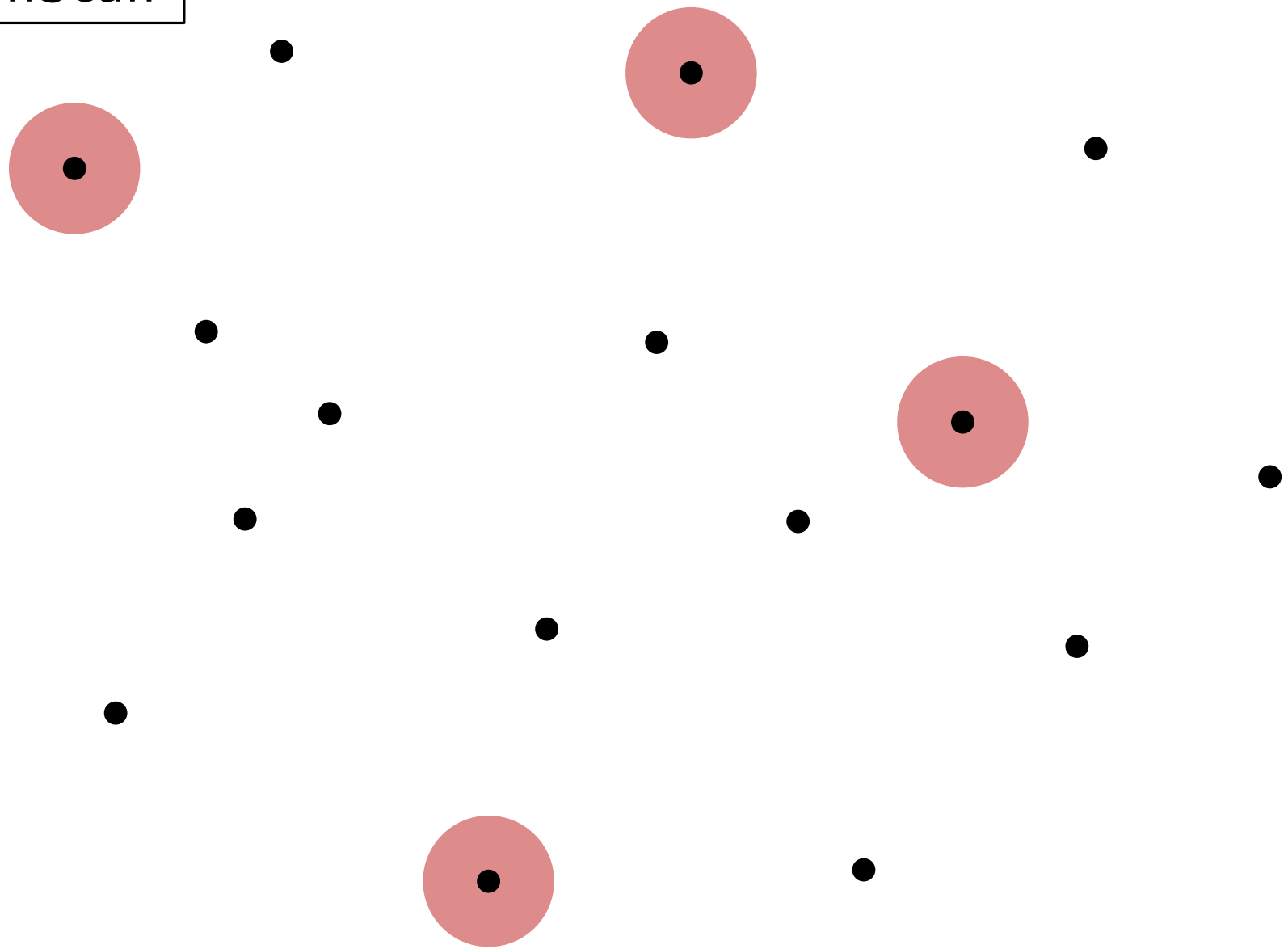
GrahamScan



$n = 16$

Beispiel

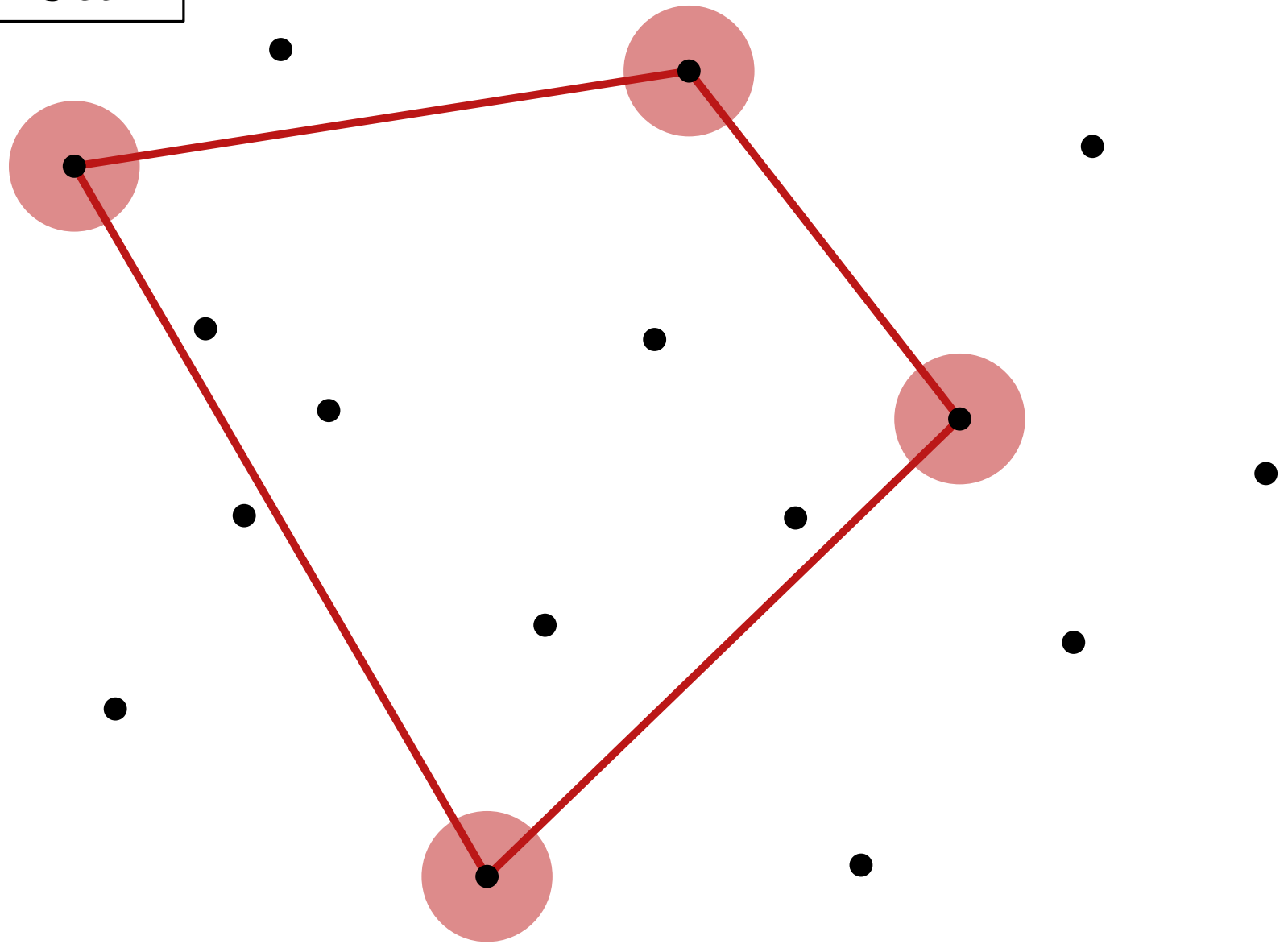
GrahamScan



$n = 16$

Beispiel

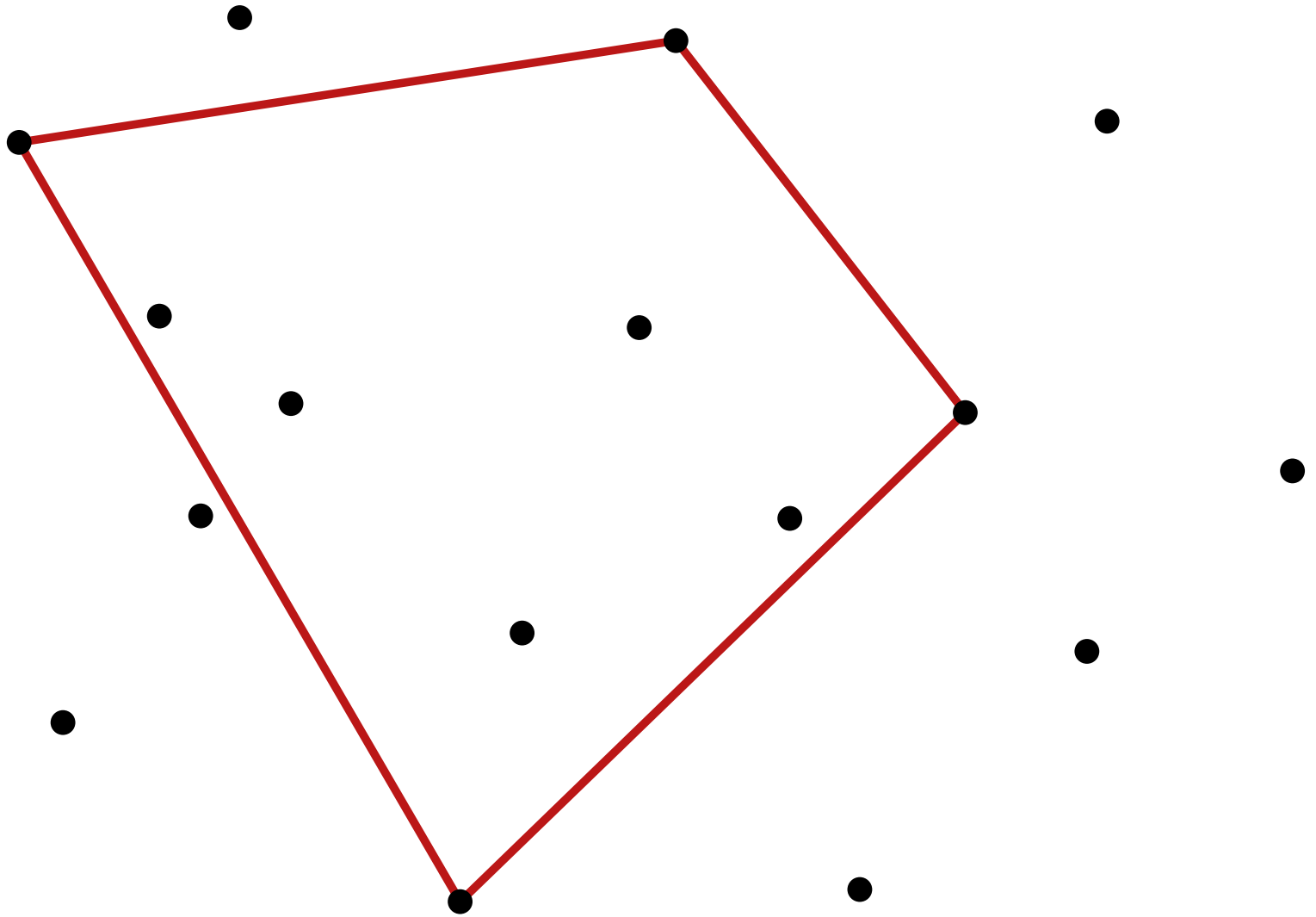
GrahamScan



$n = 16$

Beispiel

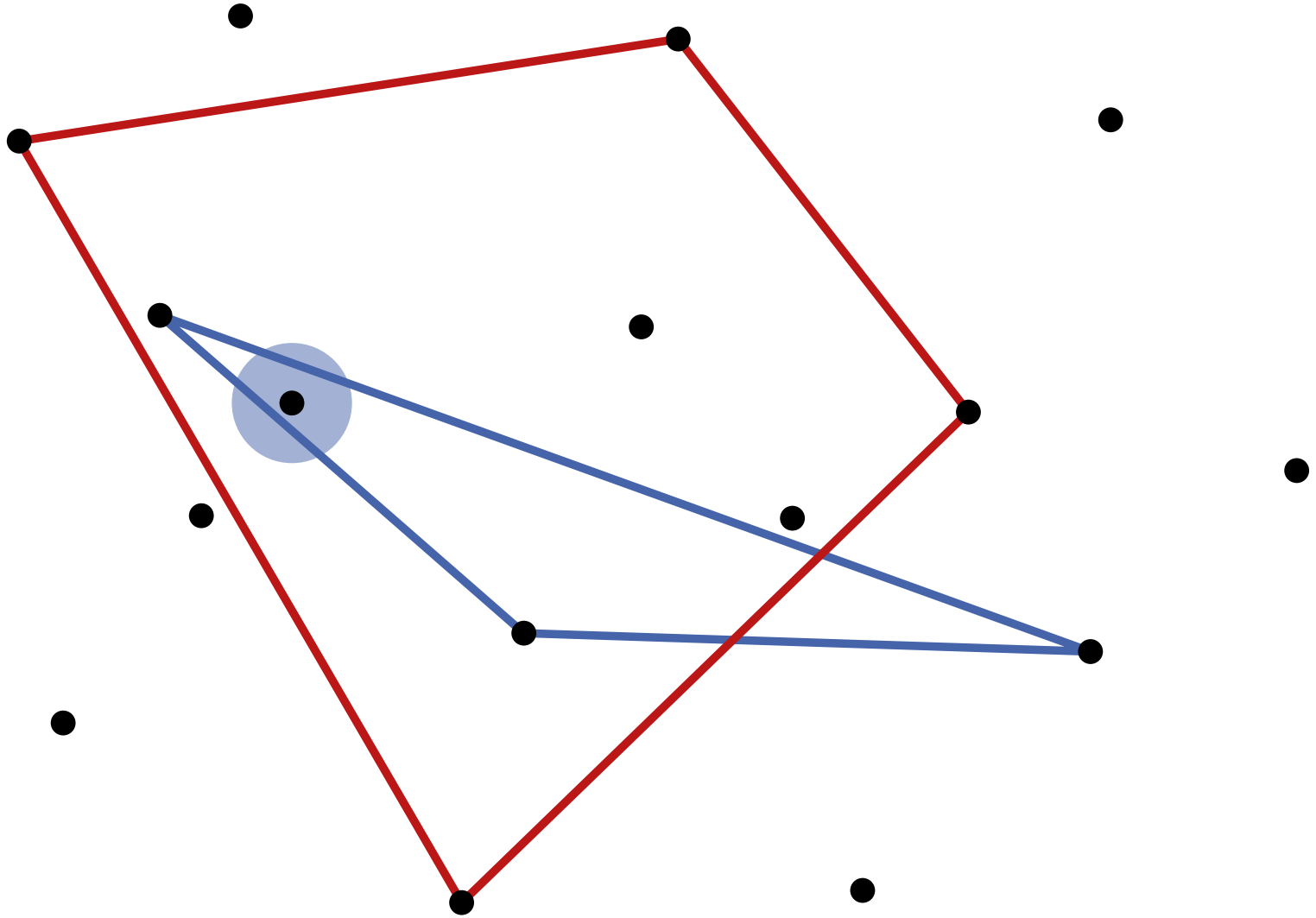
GrahamScan



$n = 16$

Beispiel

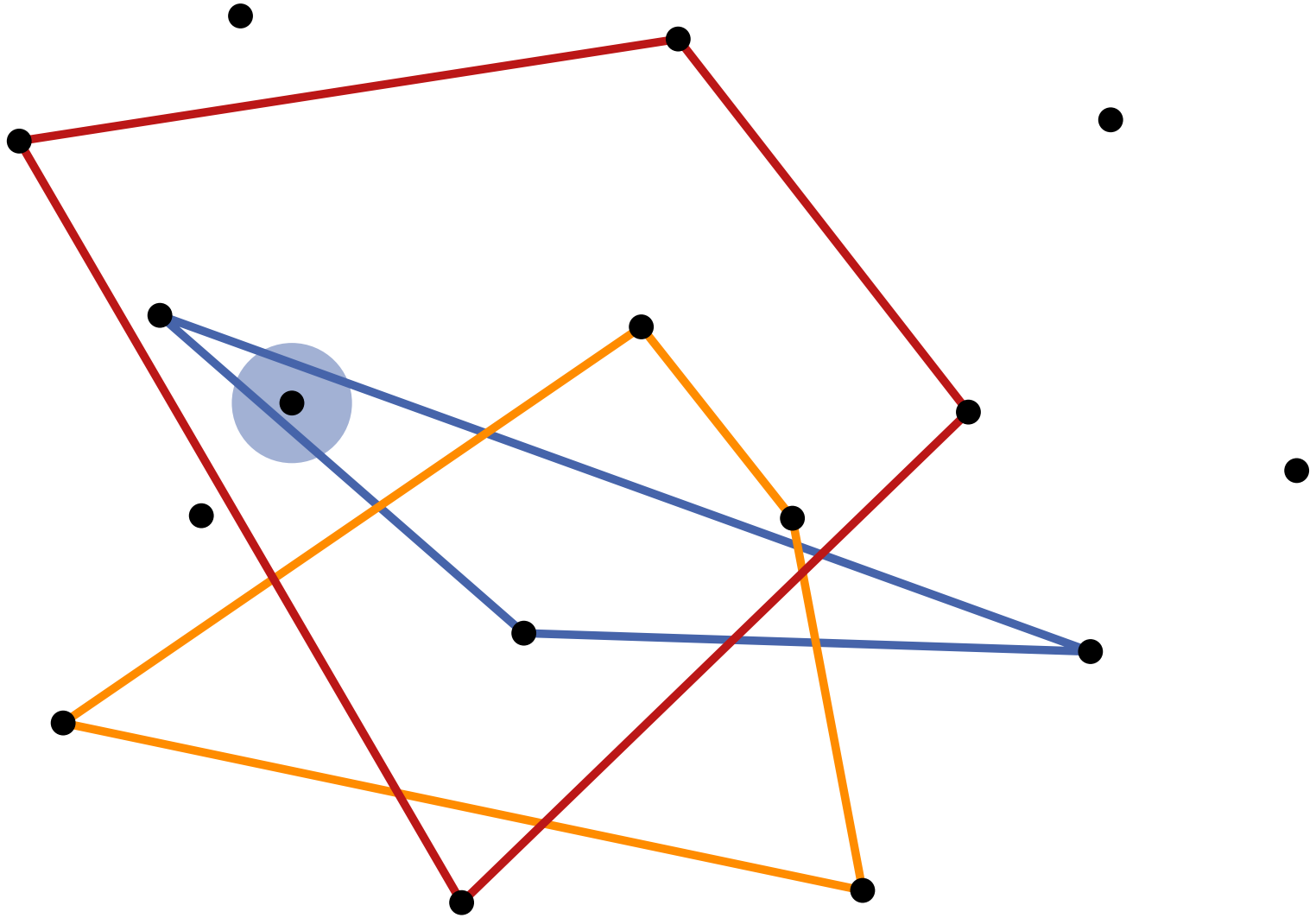
GrahamScan



$n = 16$

Beispiel

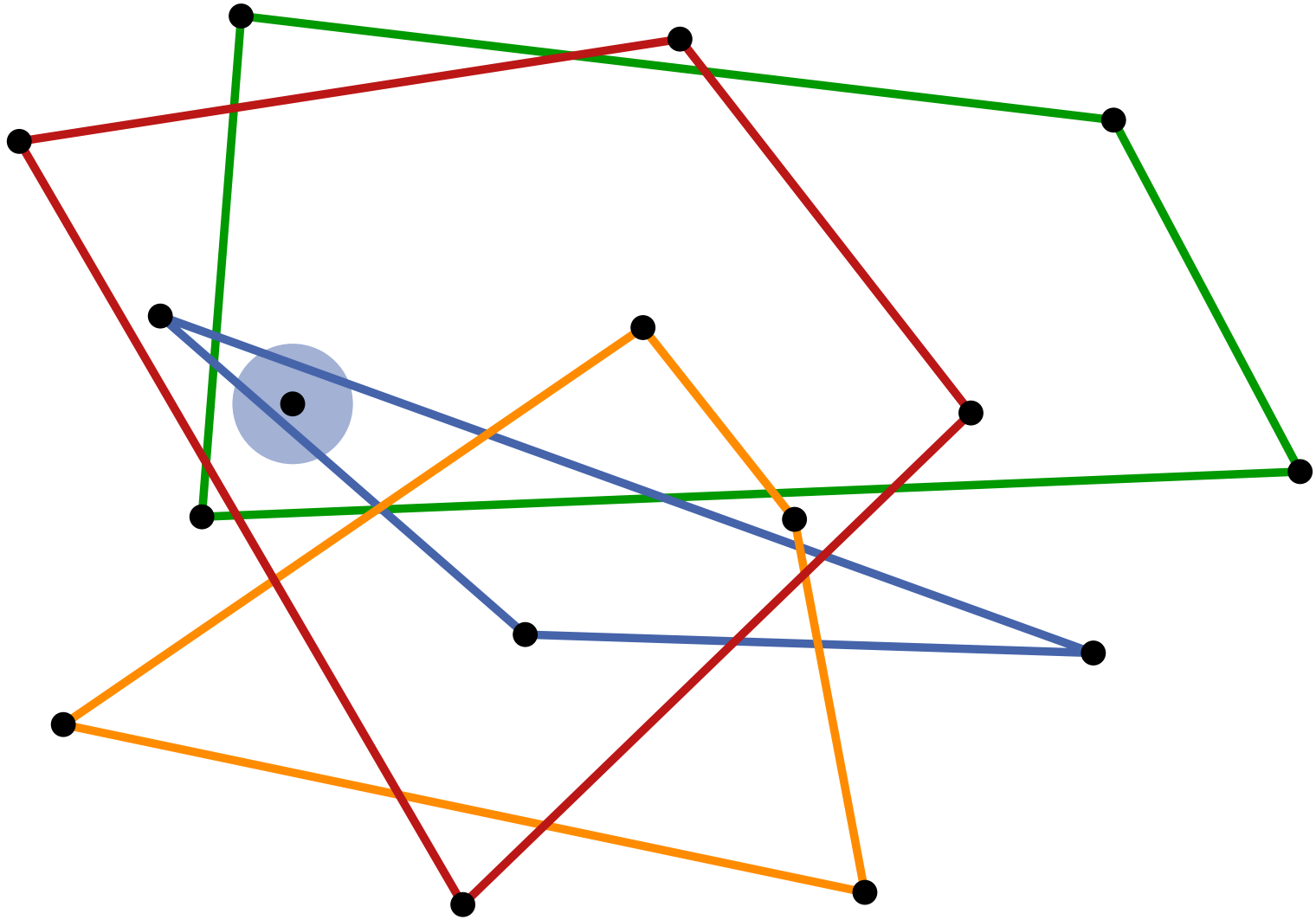
GrahamScan



$n = 16$

Beispiel

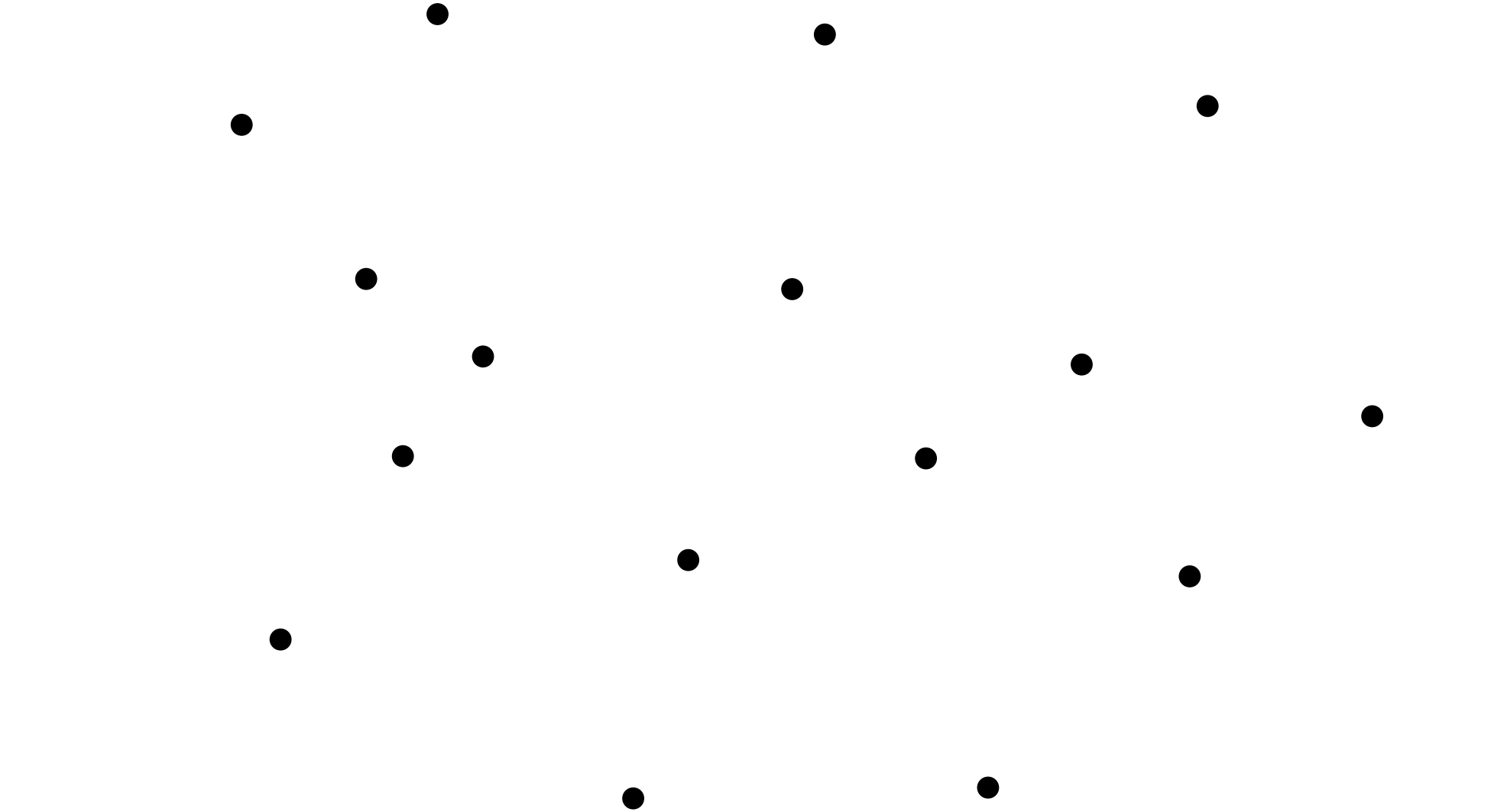
GrahamScan



$n = 16$

Beispiel

GrahamScan

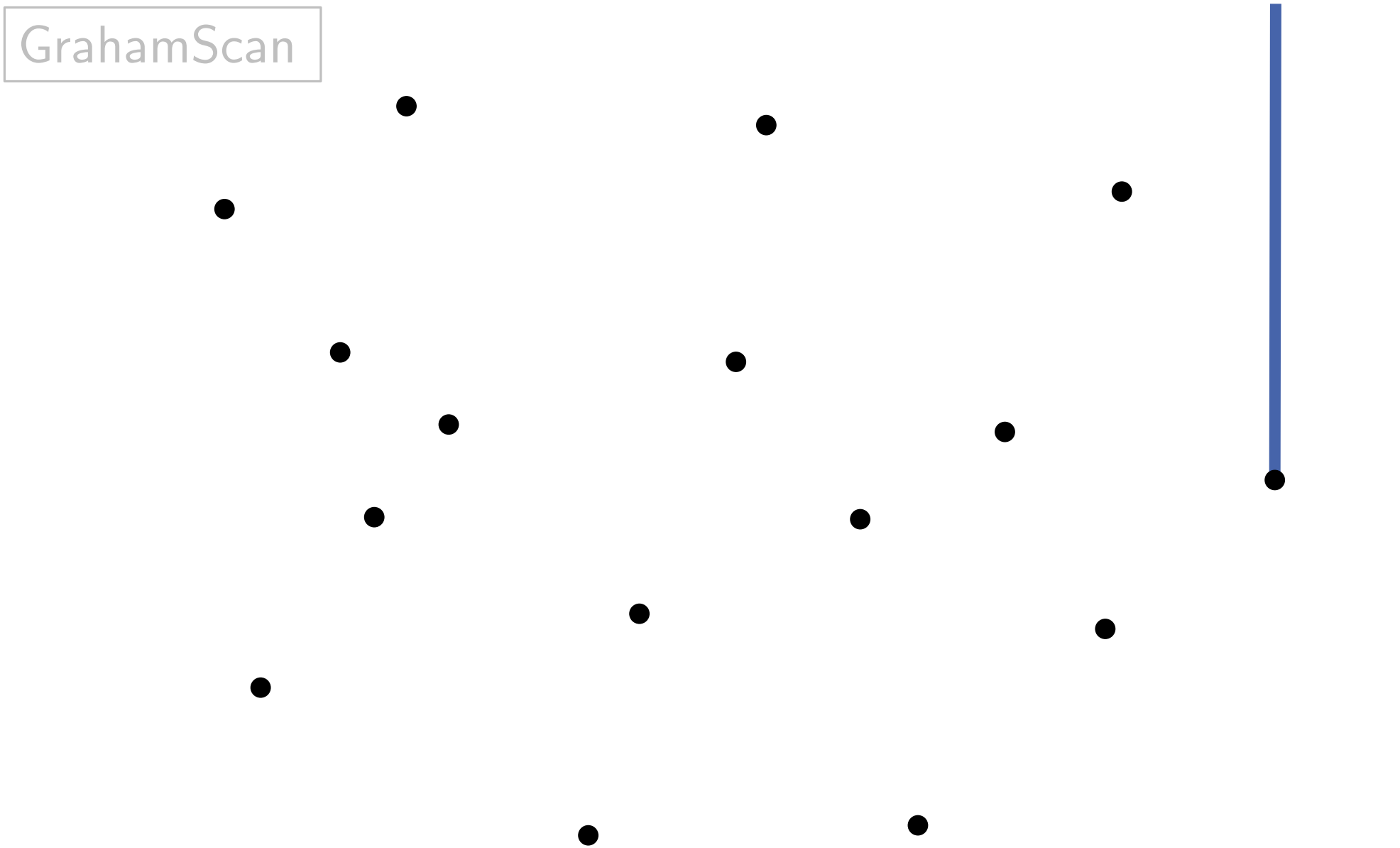


$n = 16$

Gift Wrapping

Beispiel

GrahamScan

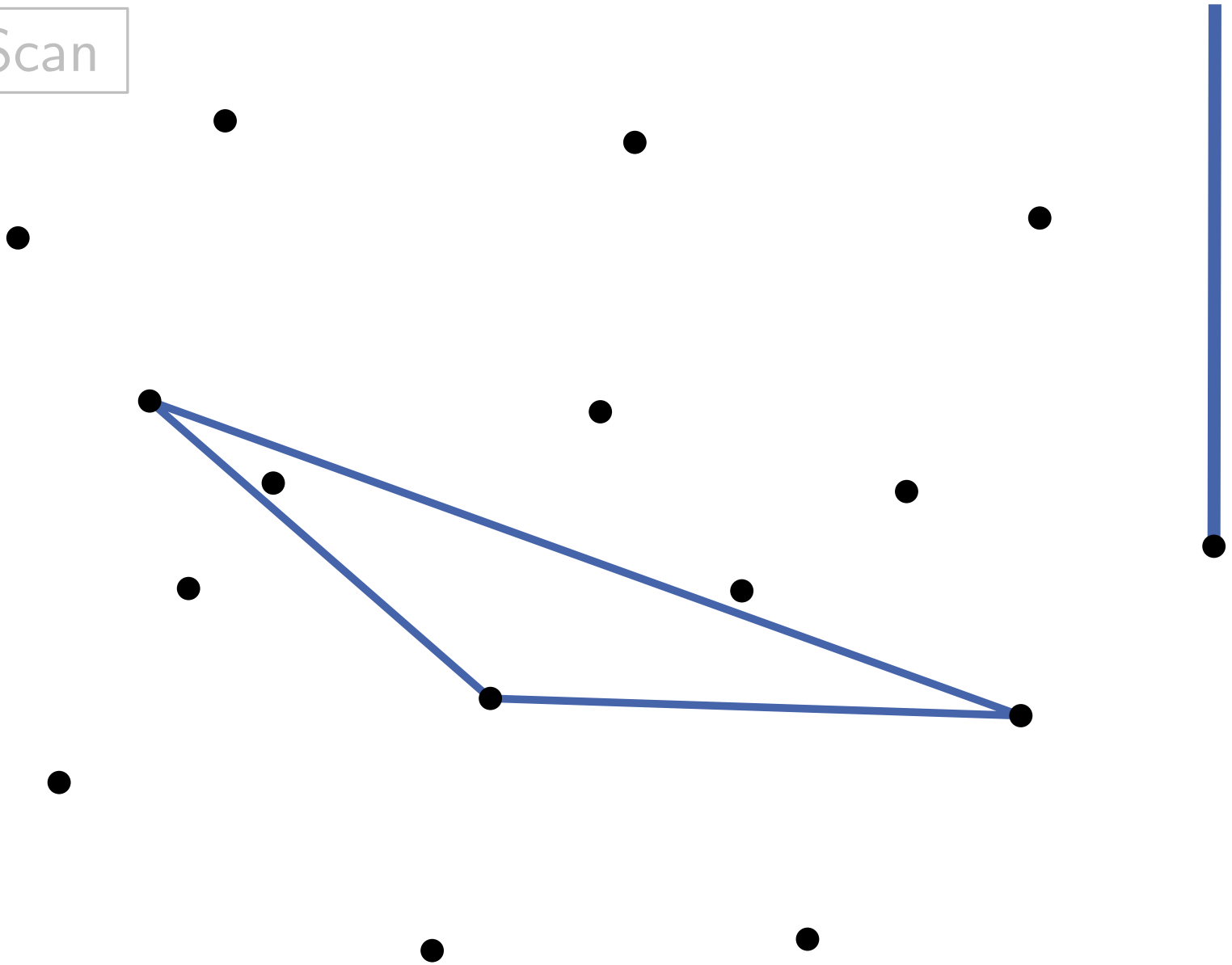


Gift Wrapping

$n = 16$

Beispiel

GrahamScan

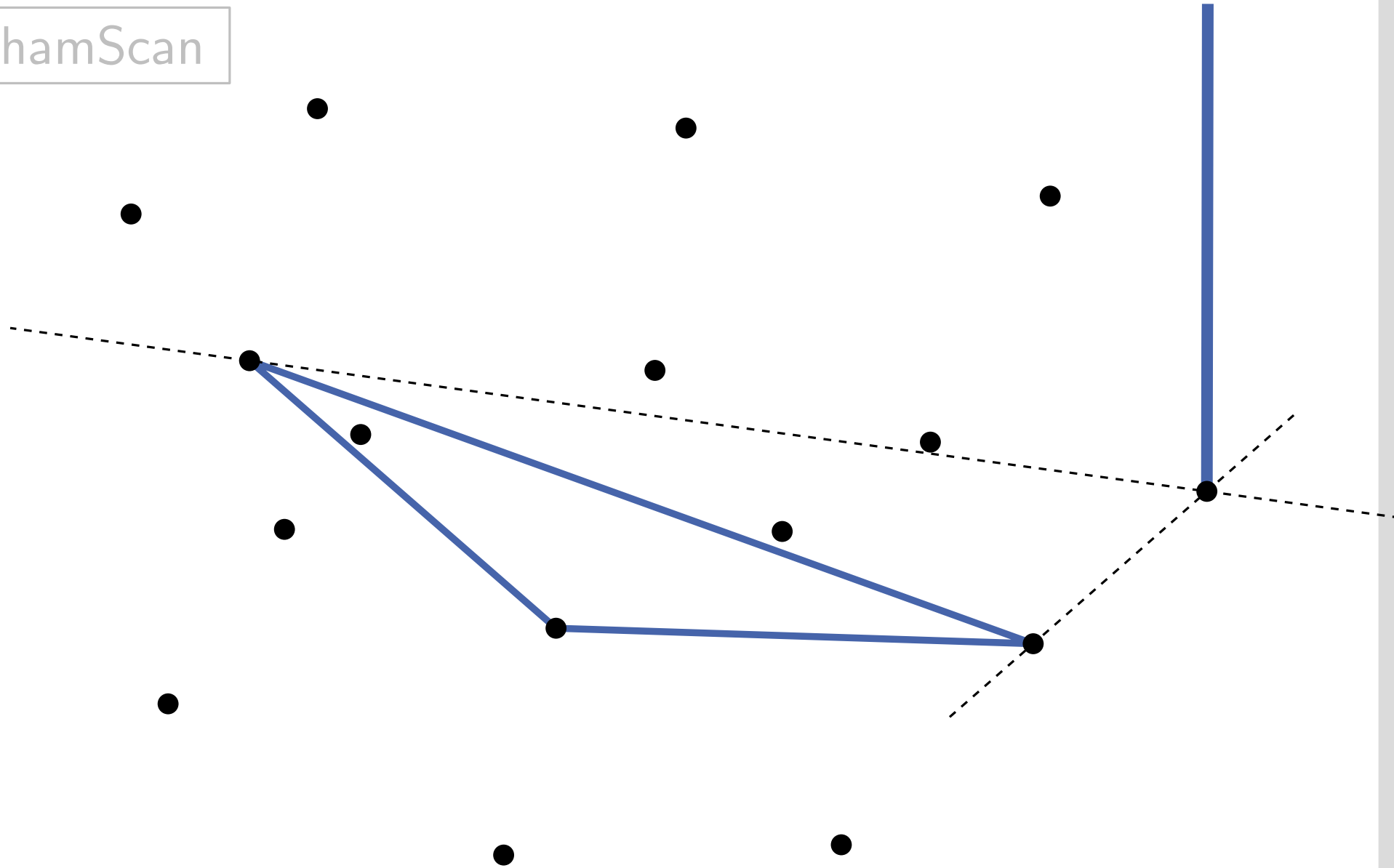


n = 16

Gift Wrapping

Beispiel

GrahamScan

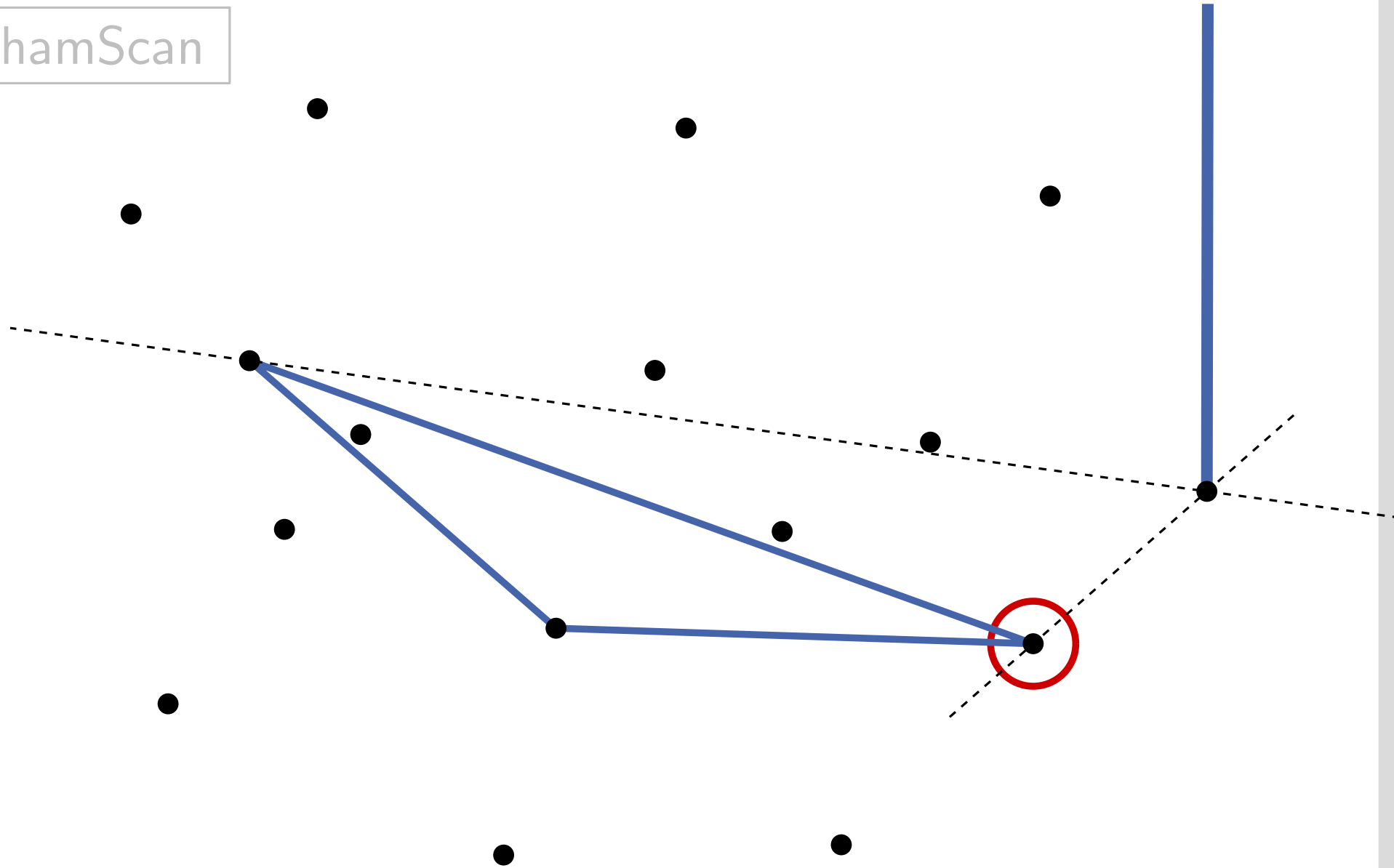


Gift Wrapping

$n = 16$

Beispiel

GrahamScan

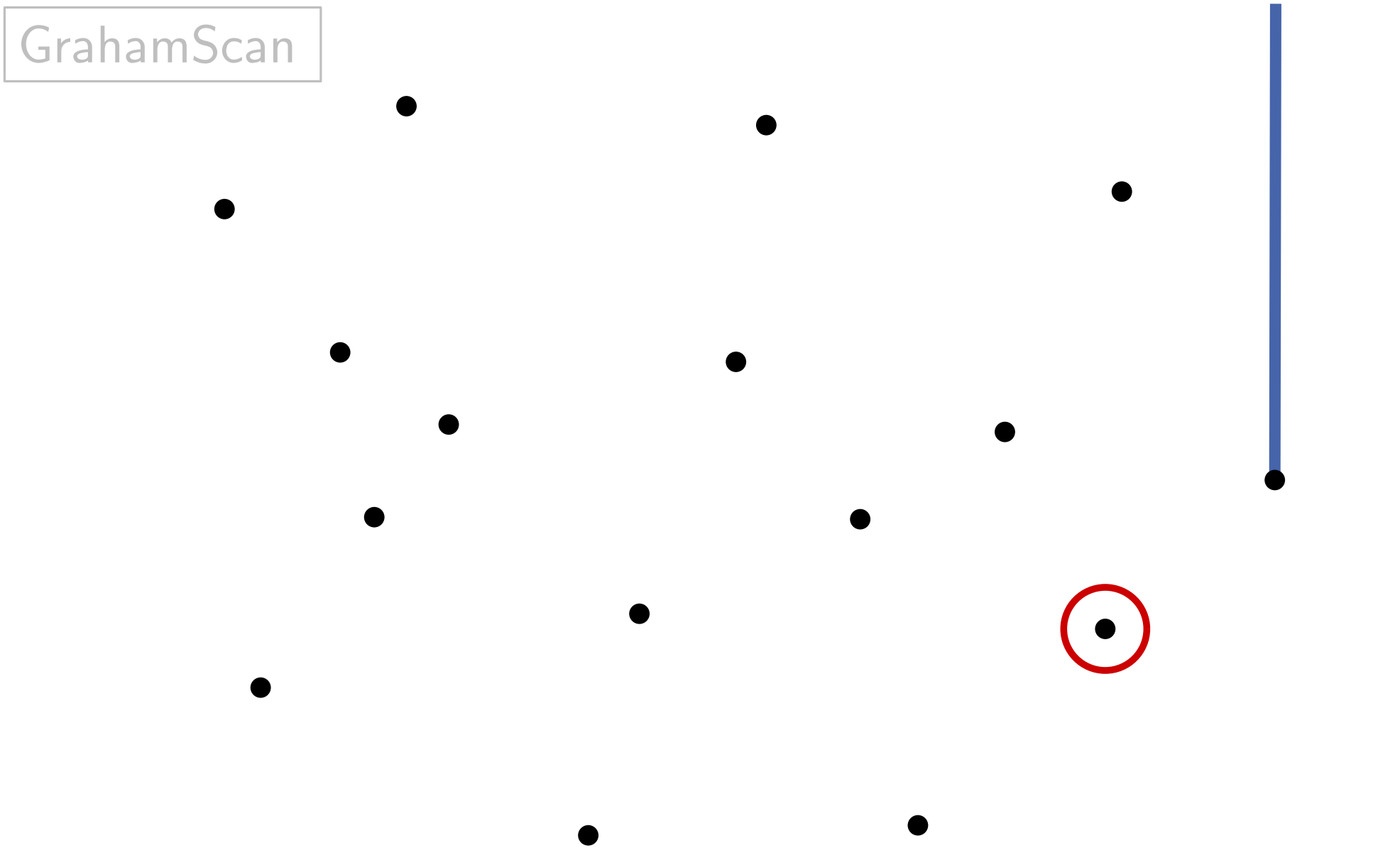


Gift Wrapping

$n = 16$

Beispiel

GrahamScan

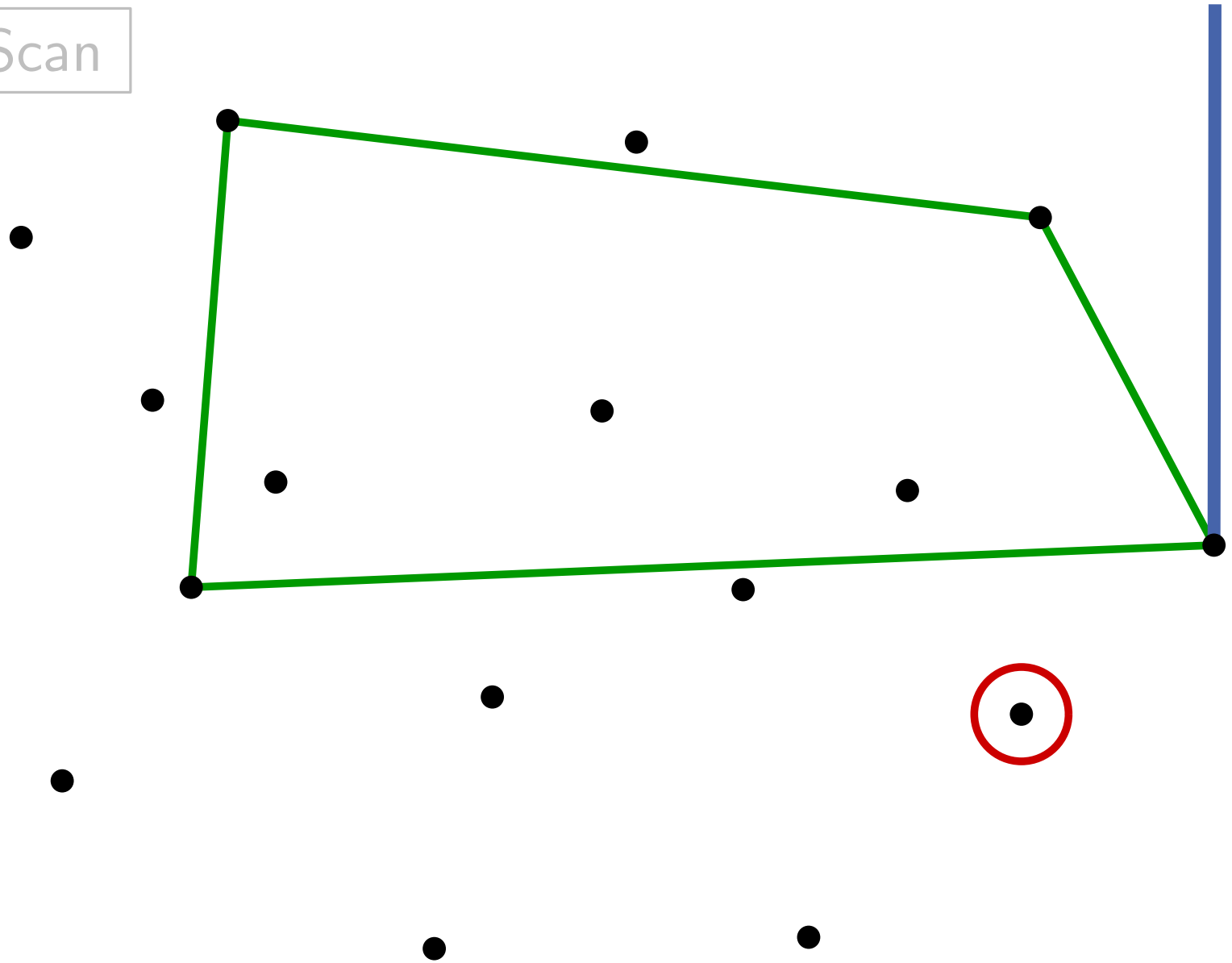


$n = 16$

Gift Wrapping

Beispiel

GrahamScan

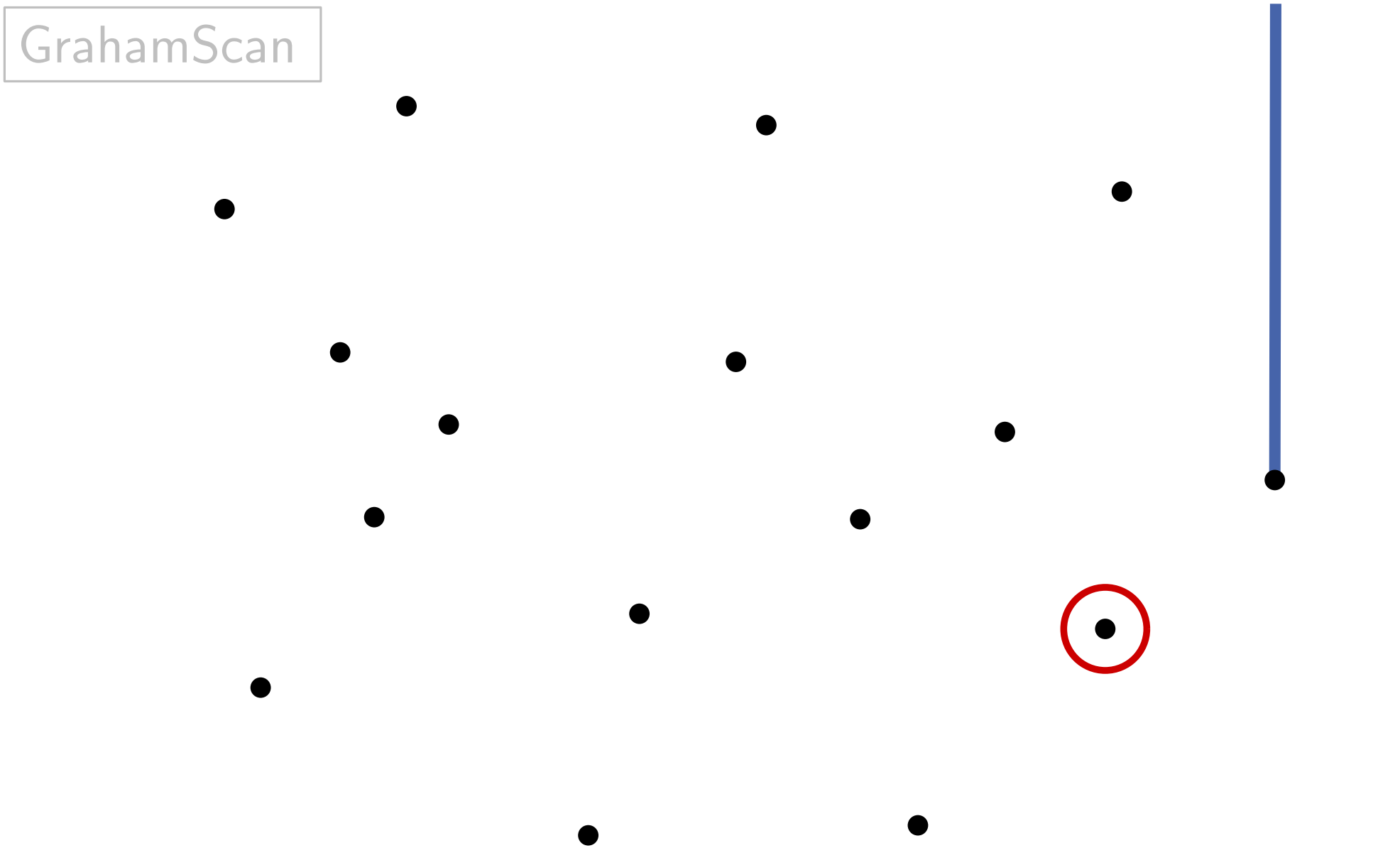


Gift Wrapping

$n = 16$

Beispiel

GrahamScan

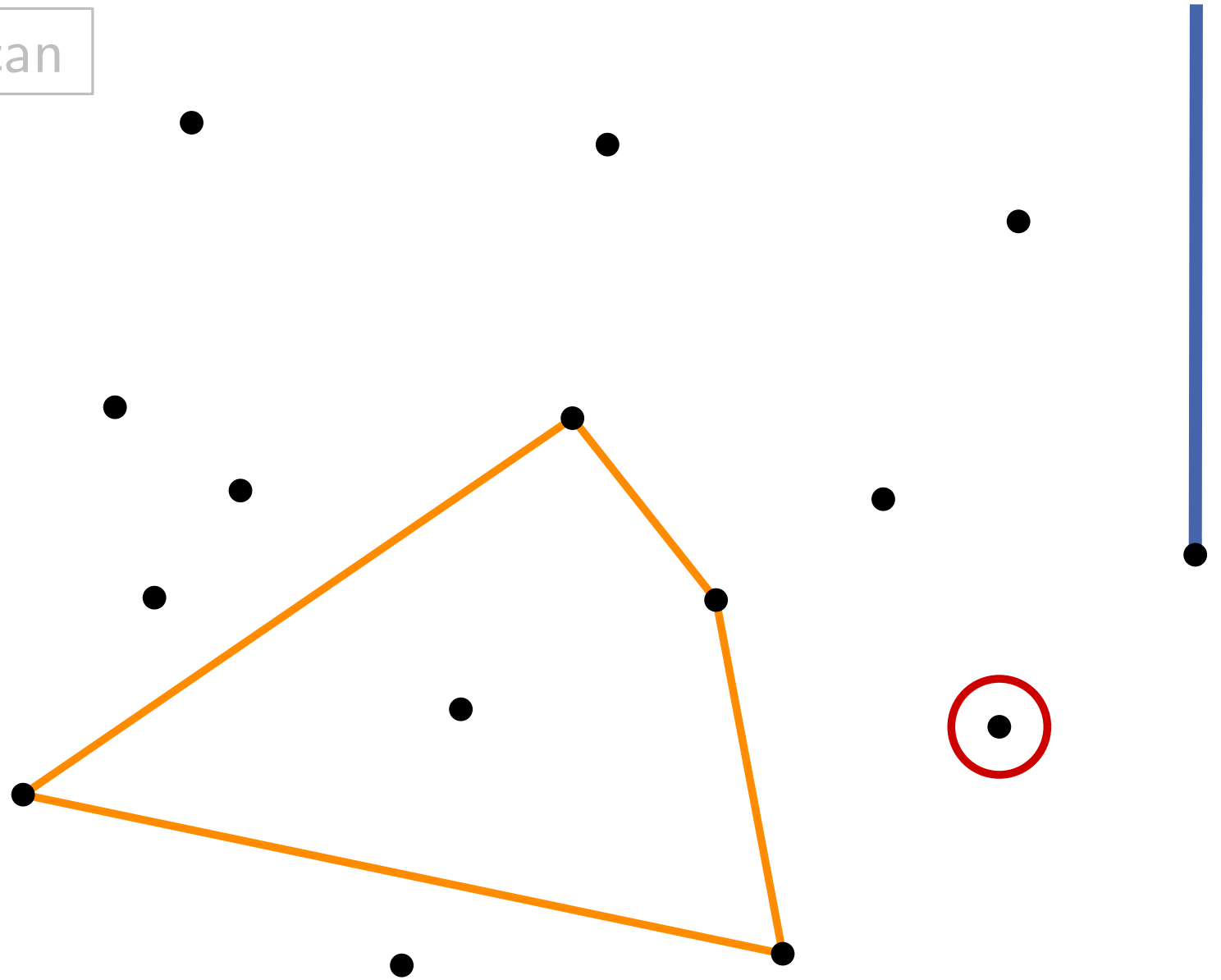


$n = 16$

Gift Wrapping

Beispiel

GrahamScan

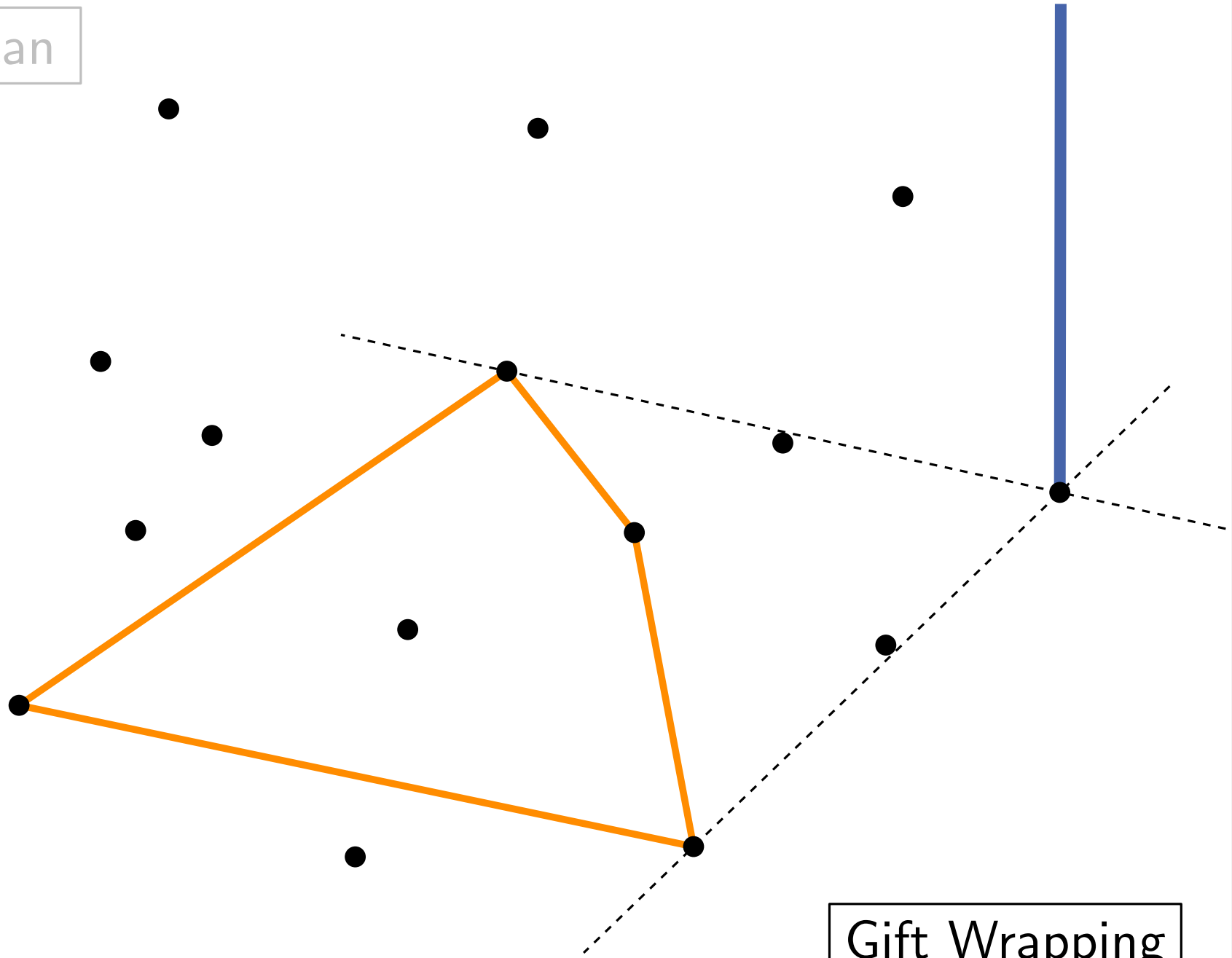


Gift Wrapping

$n = 16$

Beispiel

GrahamScan

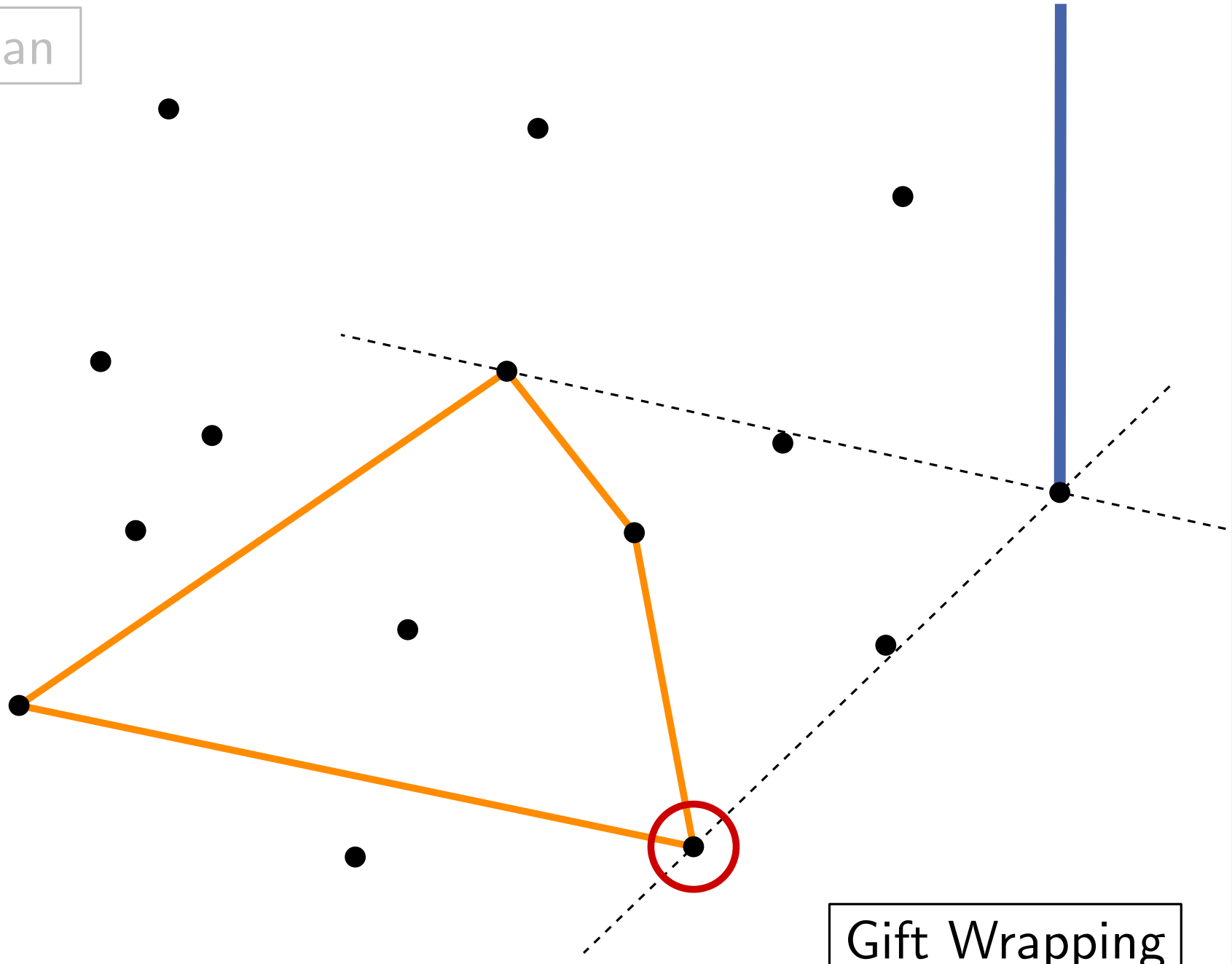


Gift Wrapping

$n = 16$

Beispiel

GrahamScan

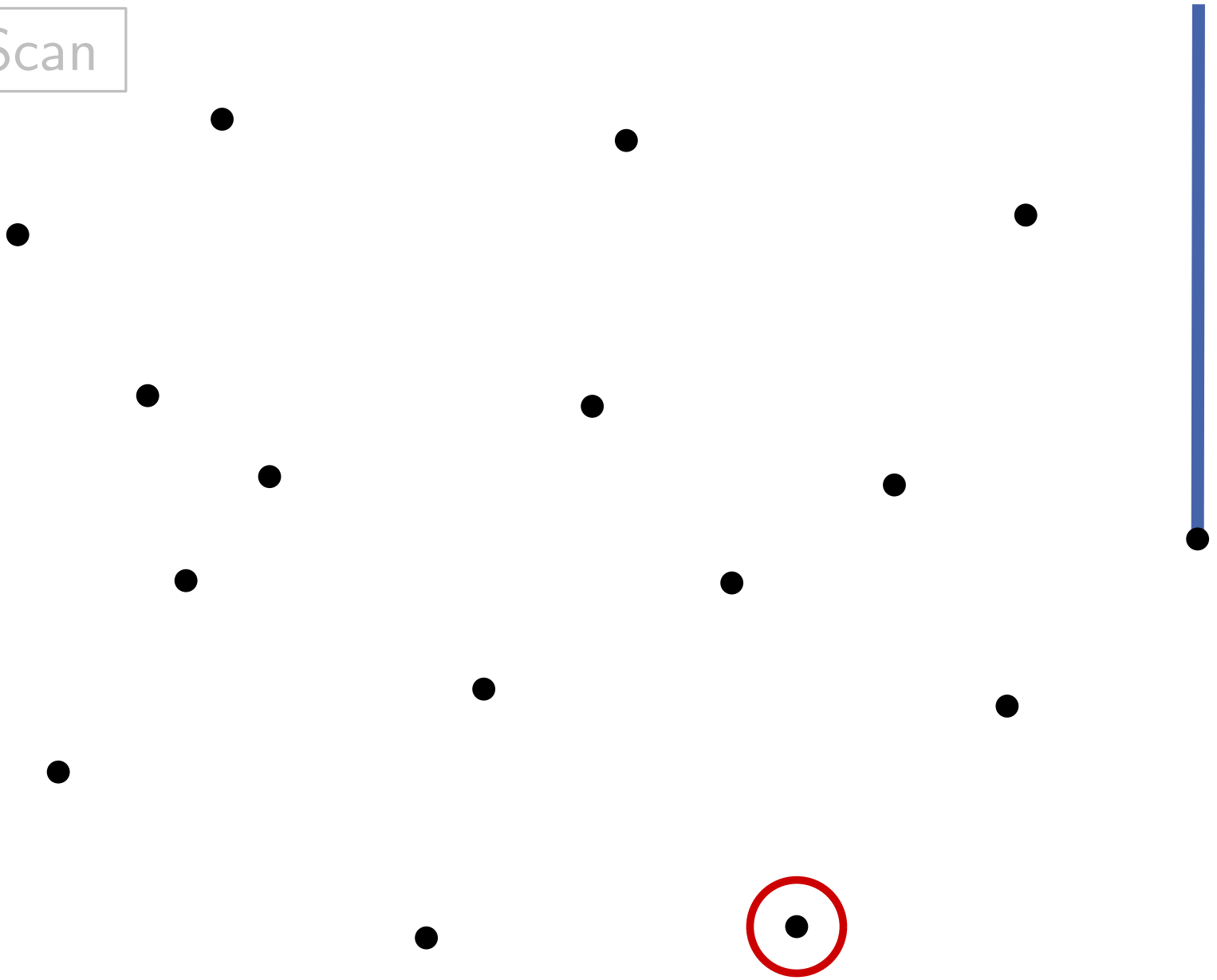


Gift Wrapping

$n = 16$

Beispiel

GrahamScan

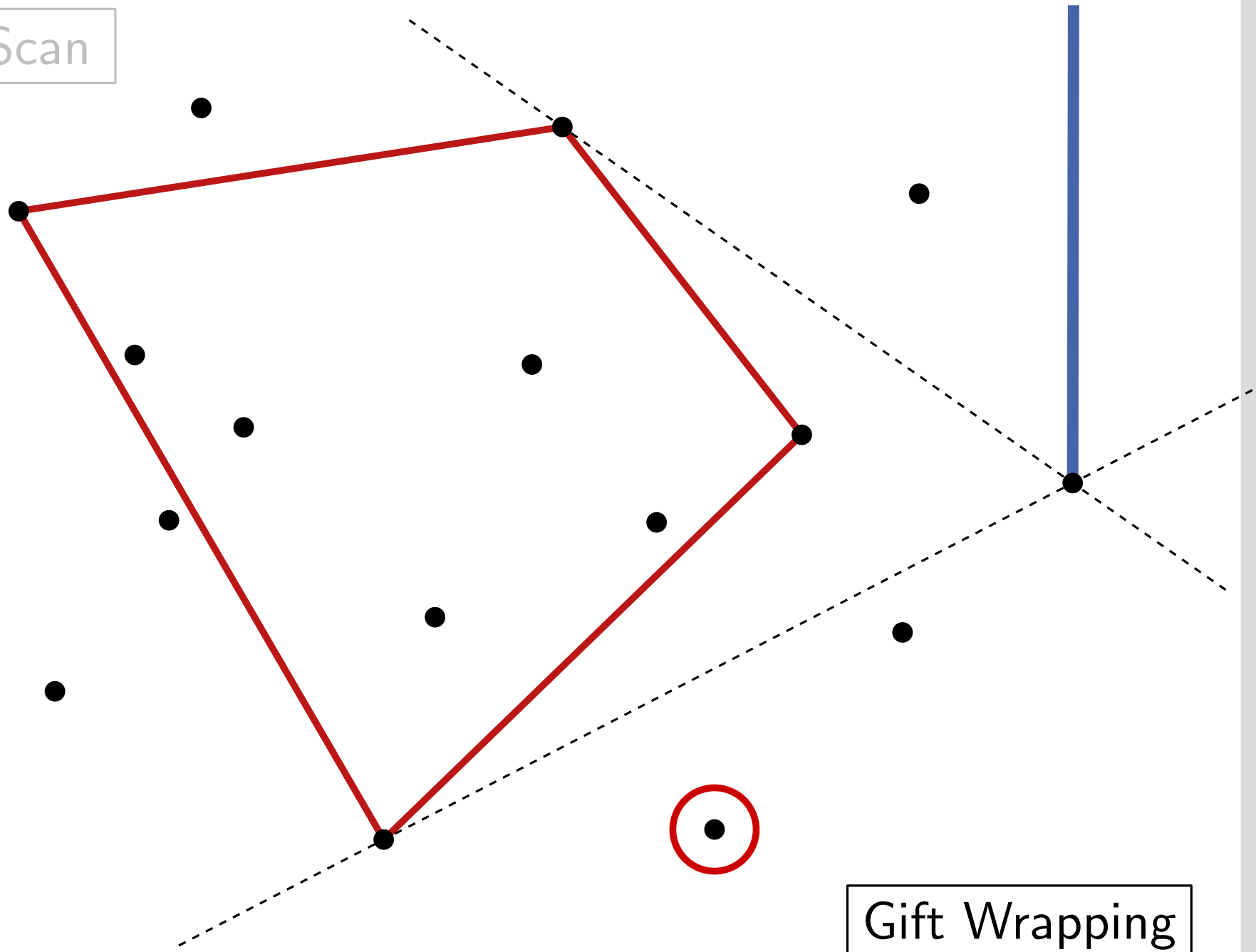


Gift Wrapping

$n = 16$

Beispiel

GrahamScan

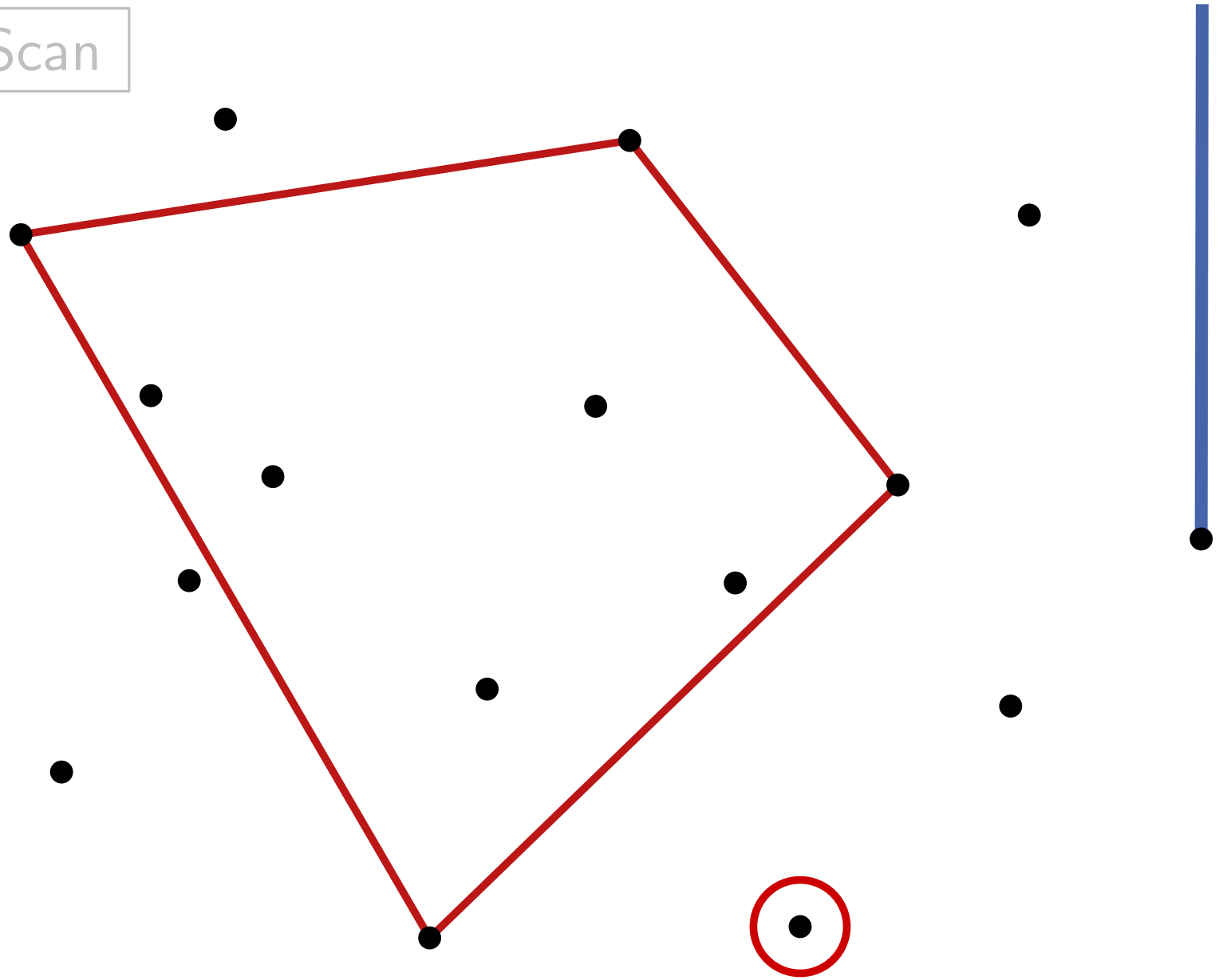


$n = 16$

Gift Wrapping

Beispiel

GrahamScan

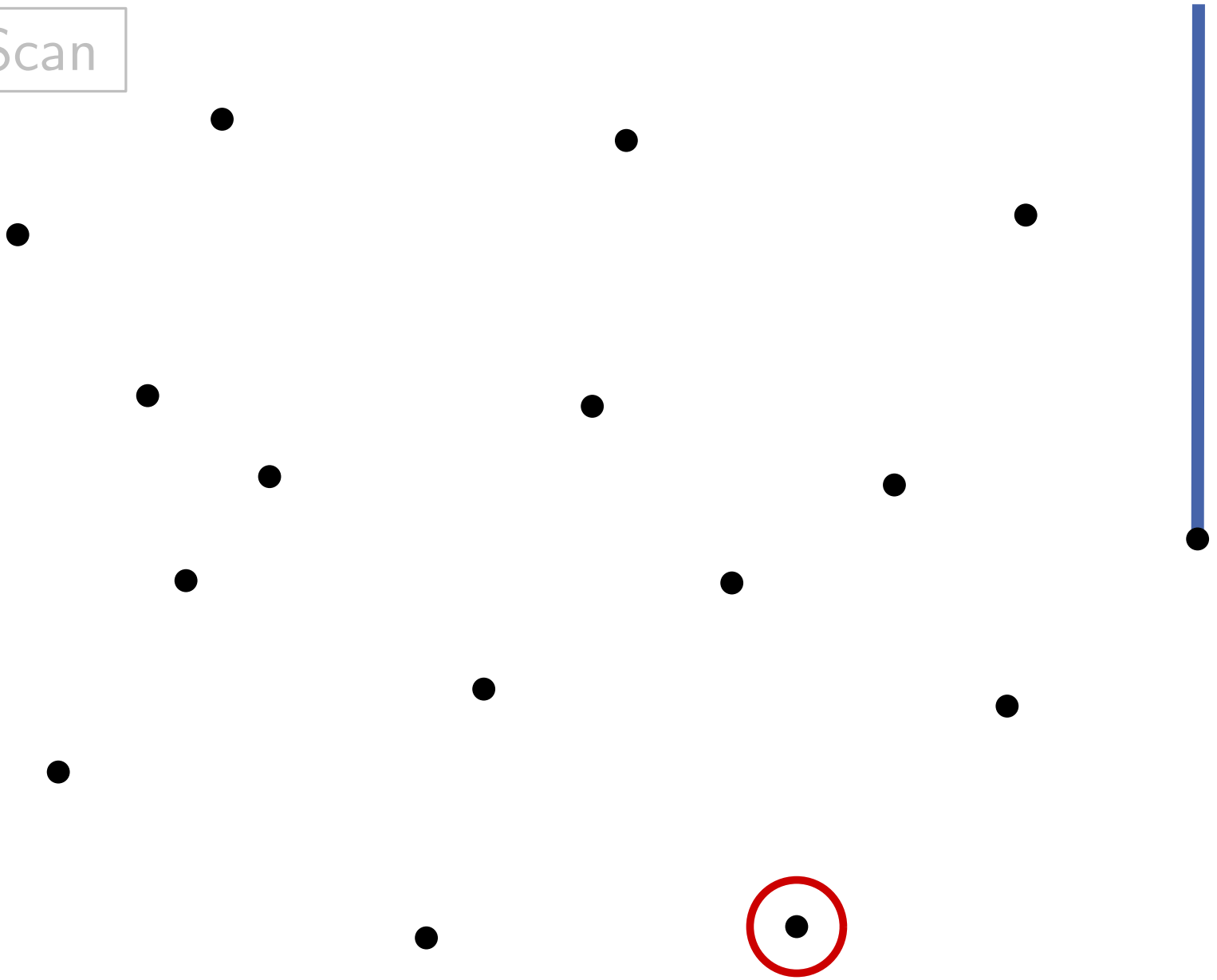


Gift Wrapping

$n = 16$

Beispiel

GrahamScan

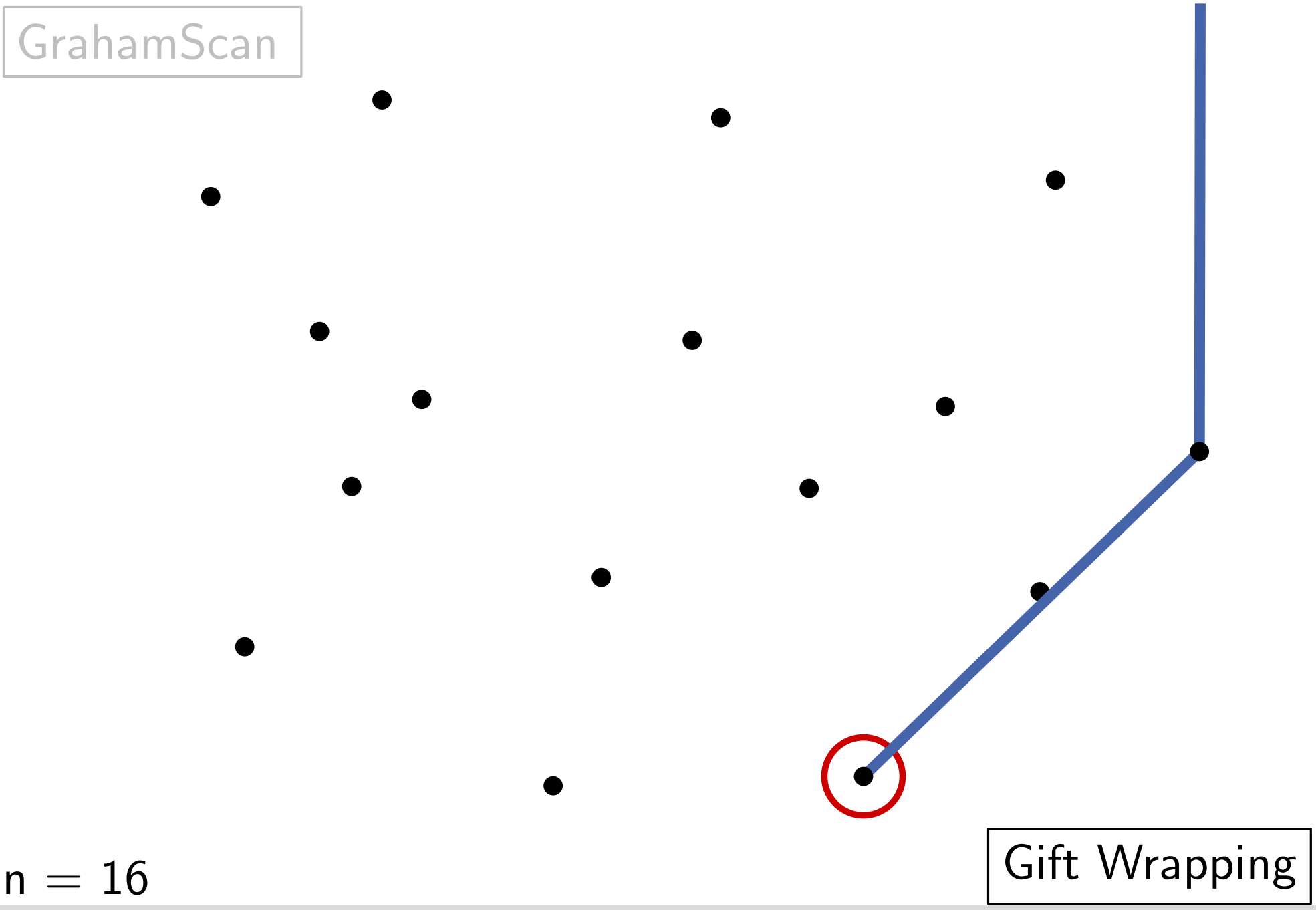


Gift Wrapping

$n = 16$

Beispiel

GrahamScan



Gift Wrapping

$n = 16$

Chans Algorithmus

Karlsruhe Institute of Technology

Angenommen, wir kennen h:

ChanHull(P,h)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do**
  Berechne per GrahamScan $CH(P_i)$

GrahamScan

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** h **do**
  for $i = 1$ **to** $\lceil n/h \rceil$ **do**
    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$
  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$

Gift Wrapping

return $(p_1, \ldots, p_h)$

Chans Algorithmus

Karlsruhe Institute of Technology

Angenommen, wir kennen h:

ChanHull(P,h)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do**

  Berechne per GrahamScan $CH(P_i)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** h **do**

  for $i = 1$ **to** $\lceil n/h \rceil$ **do**

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$

return $(p_1, \ldots, p_h)$

Chans Algorithmus

Karlsruhe Institute of Technology

Angenommen, wir kennen h:

ChanHull(P,h)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do**

  Berechne per GrahamScan $CH(P_i)$ $\mathcal{O}(h \log h)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** h **do**

  for $i = 1$ **to** $\lceil n/h \rceil$ **do**

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$

return $(p_1, \ldots, p_h)$

Chans Algorithmus

Karlsruhe Institute of Technology

Angenommen, wir kennen h:

ChanHull(P,h)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do** $\mathcal{O}(n/h)$

  Berechne per GrahamScan $CH(P_i)$ $\mathcal{O}(h \log h)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** h **do**

  for $i = 1$ **to** $\lceil n/h \rceil$ **do**

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$

return $(p_1, \ldots, p_h)$

Chans Algorithmus

Karlsruhe Institute of Technology

Angenommen, wir kennen h:

ChanHull(P,h)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do** $\mathcal{O}(n/h)$

  Berechne per GrahamScan $CH(P_i)$ $\mathcal{O}(h \log h)$

$\mathcal{O}(n \log h)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** h **do**

  for $i = 1$ **to** $\lceil n/h \rceil$ **do**

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$

return $(p_1, \ldots, p_h)$

Chans Algorithmus

Karlsruhe Institute of Technology

Angenommen, wir kennen h:

ChanHull(P,h)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do** $\mathcal{O}(n/h)$

  Berechne per GrahamScan $CH(P_i)$ $\mathcal{O}(h \log h)$

$\mathcal{O}(n \log h)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** h **do**

  for $i = 1$ **to** $\lceil n/h \rceil$ **do**

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$

$\mathcal{O}(\log h) \rightarrow$ s. Übung!

return $(p_1, \ldots, p_h)$

Chans Algorithmus

Karlsruhe Institute of Technology

Angenommen, wir kennen h:

ChanHull(P,h)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do** $\mathcal{O}(n/h)$

  Berechne per GrahamScan $CH(P_i)$ $\mathcal{O}(h \log h)$

$\mathcal{O}(n \log h)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** h **do** $\mathcal{O}(h) \cdot \mathcal{O}(n/h) = \mathcal{O}(n)$

  for $i = 1$ **to** $\lceil n/h \rceil$ **do** $\mathcal{O}(\log h) \rightarrow$ s. Übung!

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$

return $(p_1, \ldots, p_h)$

Chans Algorithmus

Angenommen, wir kennen h:

ChanHull(P,h)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do** $\mathcal{O}(n/h)$

  Berechne per GrahamScan $CH(P_i)$ $\mathcal{O}(h \log h)$

$\mathcal{O}(n \log h)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** h **do**

$\mathcal{O}(h) \cdot \mathcal{O}(n/h) = \mathcal{O}(n)$

  for $i = 1$ **to** $\lceil n/h \rceil$ **do**

$\mathcal{O}(\log h) \rightarrow$ s. Übung!

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$

$\mathcal{O}(n \log h)$

return $(p_1, \ldots, p_h)$

Chans Algorithmus

Karlsruhe Institute of Technology

Angenommen, wir kennen h:

ChanHull(P,h)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do** $\mathcal{O}(n/h)$

  Berechne per GrahamScan $CH(P_i)$ $\mathcal{O}(h \log h)$

$\mathcal{O}(n \log h)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** h **do**

$\mathcal{O}(h) \cdot \mathcal{O}(n/h) = \mathcal{O}(n)$

  for $i = 1$ **to** $\lceil n/h \rceil$ **do**

$\mathcal{O}(\log h) \rightarrow$ s. Übung!

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$

$\mathcal{O}(n \log h)$

return $(p_1, \ldots, p_h)$

Insgesamt: $\mathcal{O}(n \log h)$

Chans Algorithmus

Im Allgemeinen ist h aber unbekannt!

ChanHull(P,h)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do** $\mathcal{O}(n/h)$

  Berechne per GrahamScan $CH(P_i)$ $\mathcal{O}(h \log h)$

$\mathcal{O}(n \log h)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** h **do**

$\mathcal{O}(h) \cdot \mathcal{O}(n/h) = \mathcal{O}(n)$

  for $i = 1$ **to** $\lceil n/h \rceil$ **do**

$\mathcal{O}(\log h) \rightarrow$ s. Übung!

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$

$\mathcal{O}(n \log h)$

return $(p_1, \ldots, p_h)$

Chans Algorithmus

Im Allgemeinen ist h aber unbekannt!

ChanHull(P, ~~h~~ m)

unterteile P in Mengen P_i mit je $\leq h$ Knoten

for i von 1 bis $\lceil n/h \rceil$ **do** $\mathcal{O}(n/h)$

  Berechne per GrahamScan $CH(P_i)$ $\mathcal{O}(h \log h)$

$\mathcal{O}(n \log h)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** h **do**

$\mathcal{O}(h) \cdot \mathcal{O}(n/h) = \mathcal{O}(n)$

  for $i = 1$ **to** $\lceil n/h \rceil$ **do**

$\mathcal{O}(\log h) \rightarrow$ s. Übung!

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/h \rceil}\}\}$

$\mathcal{O}(n \log h)$

return $(p_1, \ldots, p_h)$

Chans Algorithmus

Im Allgemeinen ist h aber unbekannt!

ChanHull(P,m)

unterteile P in Mengen P_i mit je $\leq m$ Knoten

for i von 1 bis $\lceil n/m \rceil$ **do** $\mathcal{O}(n/m)$

  Berechne per GrahamScan $CH(P_i)$ $\mathcal{O}(m \log m)$

$\mathcal{O}(n \log m)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** m **do** $\mathcal{O}(m) \cdot \mathcal{O}(n/m) = \mathcal{O}(n)$

  for $i = 1$ **to** $\lceil n/m \rceil$ **do**

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$ $\mathcal{O}(\log m)$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/m \rceil}\}\}$

$\mathcal{O}(n \log m)$

return $(p_1, \ldots, p_m)$

Insgesamt: $\mathcal{O}(n \log m)$

Chans Algorithmus

Im Allgemeinen ist h aber unbekannt!

ChanHull(P,m)

unterteile P in Mengen P_i mit je $\leq m$ Knoten

for i von 1 bis $\lceil n/m \rceil$ **do** $\mathcal{O}(n/m)$

  Berechne per GrahamScan $CH(P_i)$ $\mathcal{O}(m \log m)$

$\mathcal{O}(n \log m)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** m **do** $\mathcal{O}(m) \cdot \mathcal{O}(n/m) = \mathcal{O}(n)$

  for $i = 1$ **to** $\lceil n/m \rceil$ **do**

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$ $\mathcal{O}(\log m)$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/m \rceil}\}\}$

$\mathcal{O}(n \log m)$

return $(p_1, \ldots, p_m)$

Insgesamt: $\mathcal{O}(n \log m)$

Chans Algorithmus

Im Allgemeinen ist h aber unbekannt!

ChanHull(P,m)

unterteile P in Mengen P_i mit je $\leq m$ Knoten

for i von 1 bis $\lceil n/m \rceil$ **do** $\mathcal{O}(n/m)$

  Berechne per GrahamScan $CH(P_i)$ $\mathcal{O}(m \log m)$

$\mathcal{O}(n \log m)$

$p_1 = (x_1, y_1) \leftarrow$ rechtester Punkt in P

$p_0 \leftarrow (x_1, \infty)$

for $j = 1$ **to** m **do** $\mathcal{O}(m) \cdot \mathcal{O}(n/m) = \mathcal{O}(n)$

  for $i = 1$ **to** $\lceil n/m \rceil$ **do**

    $q_i \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in P_i \setminus \{p_{j-1}, p_j\}\}$ $\mathcal{O}(\log m)$

  $p_{j+1} \leftarrow \arg\max\{\angle p_{j-1} p_j q \mid q \in \{q_1, \ldots, q_{\lceil n/m \rceil}\}\}$

  if $p_{j+1} = p_1$ **then return** $(p_1, \ldots, p_{j+1})$

return failure

$\mathcal{O}(n \log m)$

Insgesamt: $\mathcal{O}(n \log m)$

Was ist mit m?

Vorschläge?

Was ist mit m?

FullChanHull(P)

for $t = 0, 1, 2, \ldots$ **do**

$m =\leftarrow \min\{n, 2^{2^t}\}$

result $\leftarrow$ ChanHull(P, m)

if result $\neq$ failure **then** break

return result

Was ist mit m?

FullChanHull(P)

for $t = 0, 1, 2, \ldots$ **do**

  $m =\leftarrow \min\{n, 2^{2^t}\}$

  result $\leftarrow$ ChanHull(P, m)

  if result $\neq$ failure **then** break

return result

$\mathcal{O}(n \log m) = \mathcal{O}(n \log 2^{2^t})$

Laufzeit:

Was ist mit m?

FullChanHull(P)

for $t = 0, 1, 2, \ldots$ **do**
  $m = \leftarrow \min\{n, 2^{2^t}\}$
  result $\leftarrow$ ChanHull(P, m) $\mathcal{O}(n \log m) = \mathcal{O}(n \log 2^{2^t})$
  if result $\neq$ failure **then** break
return result

Laufzeit:

$$\sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(n \log 2^{2^t})$$

Was ist mit m?

FullChanHull(P)

for $t = 0, 1, 2, \ldots$ **do**
  $m = \leftarrow \min\{n, 2^{2^t}\}$
  result $\leftarrow$ ChanHull(P, m)
  if result $\neq$ failure **then** break
return result

$\mathcal{O}(n \log m) = \mathcal{O}(n \log 2^{2^t})$

Laufzeit:

$$\sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(n \log 2^{2^t}) = \mathcal{O}(n) \sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(2^t)$$

Was ist mit m?

FullChanHull(P)

for $t = 0, 1, 2, \ldots$ **do**
  $m =\leftarrow \min\{n, 2^{2^t}\}$
  result $\leftarrow$ ChanHull(P, m)
  if result $\neq$ failure **then** break
return result

$\mathcal{O}(n \log m) = \mathcal{O}(n \log 2^{2^t})$

Laufzeit:

$$\sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(n \log 2^{2^t}) = \mathcal{O}(n) \sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(2^t)$$

$$\leq \mathcal{O}(n) \cdot \mathcal{O}(2^{\log \log h})$$

Was ist mit m?

FullChanHull(P)

for $t = 0, 1, 2, \ldots$ **do**
  $m =\leftarrow \min\{n, 2^{2^t}\}$
  result $\leftarrow$ ChanHull(P, m)
  if result $\neq$ failure **then** break
return result

$\mathcal{O}(n \log m) = \mathcal{O}(n \log 2^{2^t})$

Laufzeit:

$$\sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(n \log 2^{2^t}) = \mathcal{O}(n) \sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(2^t)$$

$$\leq \mathcal{O}(n) \cdot \mathcal{O}(2^{\log \log h}) = \mathcal{O}(n) \cdot \mathcal{O}(\log h)$$

Was ist mit m?

FullChanHull(P)

for $t = 0, 1, 2, \ldots$ **do**
  $m =\leftarrow \min\{n, 2^{2^t}\}$
  result $\leftarrow$ ChanHull(P, m)
  if result $\neq$ failure **then** break
return result

$\mathcal{O}(n \log m) = \mathcal{O}(n \log 2^{2^t})$

Laufzeit:

$$\sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(n \log 2^{2^t}) = \mathcal{O}(n) \sum_{t=0}^{\lceil \log \log h \rceil} \mathcal{O}(2^t)$$

$$\leq \mathcal{O}(n) \cdot \mathcal{O}(2^{\log \log h}) = \mathcal{O}(n) \cdot \mathcal{O}(\log h)$$

$$= \mathcal{O}(n \log h)$$

Was ist mit m?

FullChanHull(P)

for $t = 0, 1, 2, \ldots$ **do**

  $m =\leftarrow \min\{n, 2^{2^t}\}$

  result $\leftarrow$ ChanHull(P, m)

  if result $\neq$ failure **then** break

return result

Satz 3: Die konvexe Hülle $CH(P)$ von n Punkten P in $\mathbb{R}^2$ lässt sich mit Chans Algorithmus in $O(n \log h)$ Zeit berechnen, wobei $h = |CH(P)|$.

Diskussion

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

Diskussion

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

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.

Diskussion

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

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.

Was passiert in Graham's Scan, wenn die Sortierung von P nicht eindeutig ist?

Diskussion

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

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.

Was passiert in Graham's Scan, wenn die Sortierung von P nicht eindeutig ist?

Nutze lexikographische Ordnung!

Diskussion

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

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.

Was passiert in Graham's Scan, 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 $O(n \log n)$ bzw. $O(n \log h)$ Zeit?

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.

Was passiert in Graham's Scan, wenn die Sortierung von P nicht eindeutig ist?

Nutze lexikographische Ordnung!

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

Graham: Bilden keinen Rechtsknick, also wird innerer Punkt gelöscht.
Jarvis: wähle entfernteren Punkt

Diskussion

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

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.

Was passiert in Graham's Scan, wenn die Sortierung von P nicht eindeutig ist?

Nutze lexikographische Ordnung!

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

Graham: Bilden keinen Rechtsknick, also wird innerer Punkt gelöscht.
Jarvis: wähle entfernteren Punkt

Wie steht es um die Robustheit der Algorithmen?

Diskussion

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

Im allgemeinen nicht! Denn ein Algorithmus zur Berechnung der konvexen Hülle kann auch Sortieren (s. Übung) $\Rightarrow$ untere Schranke $\Omega(n \log n)$.

Was passiert in Graham's Scan, wenn die Sortierung von P nicht eindeutig ist?

Nutze lexikographische Ordnung!

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

Graham: Bilden keinen Rechtsknick, also wird innerer Punkt gelöscht.
Jarvis: wähle entfernteren Punkt

Wie steht es um die Robustheit der Algorithmen?

- Robustheit bzgl. Ungenauigkeit von Fließkommaarithmetik
- FirstConvexHull liefert evtl. kein gültiges Polygon
- Graham und Jarvis liefern 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