Karlsruher Institut für Technologie

Algorithmische Geometrie

Überblick und Konvexe Hülle

Was ist das überhaupt?

Karlsruher Institut für Technologie

Wikipedia

- Als Algorithmische Geometrie (englisch Computational Geometry) bezeichnet man ein Teilgebiet der Informatik, das sich mit der algorithmischen Lösung geometrisch formulierter Probleme beschäftigt.

Was ist das überhaupt?

Wikipedia

- Als Algorithmische Geometrie (englisch 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.

Was ist das überhaupt?

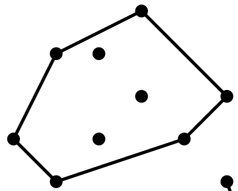
Wikipedia

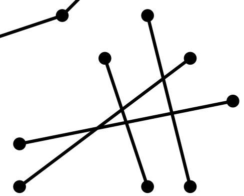
- Als Algorithmische Geometrie (englisch Computational Geometry) bezeichnet man ein Teilgebiet der Informatik, das sich mit der algorithmischen Lösung geometrisch formulierter Probleme beschäftigt.
- Ein zentrales Problem ist dabei die Speicherung und Verarbeitung geometrischer Daten.
- Im Gegensatz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

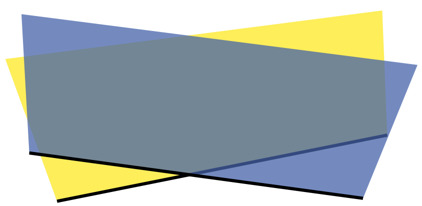
Was heißt das jetzt konkret?

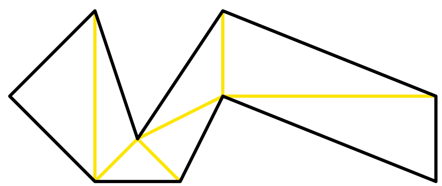
Basic Toolbox

- konvexe Hülle
- Linienschnitt
- Triangulierung
- Ebenenschnitt





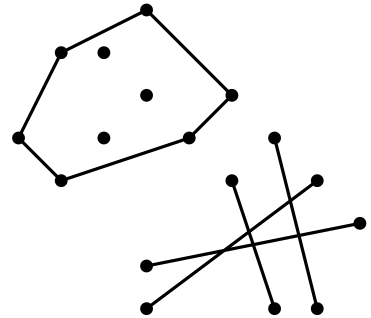


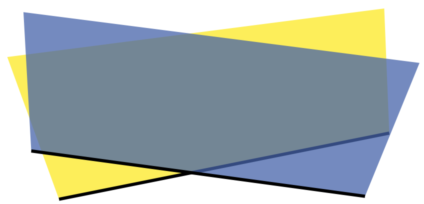


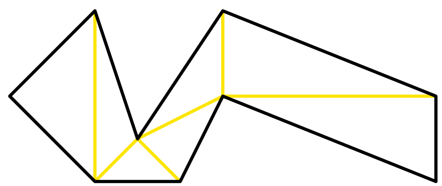
Was heißt das jetzt konkret?

Basic Toolbox

- konvexe Hülle
- Linienschnitt
- Triangulierung
- Ebenenschnitt

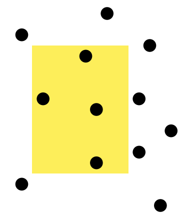


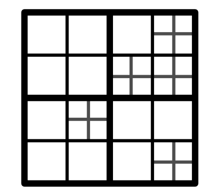


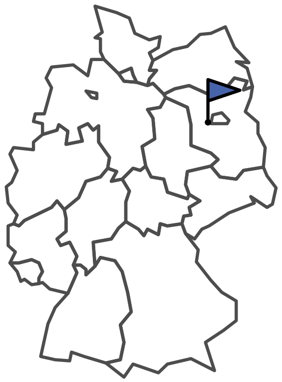


Geometrische Datenstrukturen

- orthogonal range searching
- Raum-Partitionierung
- Punkt-Lokalisierung



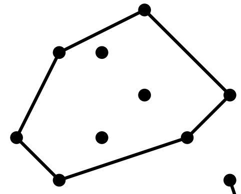


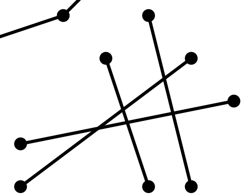


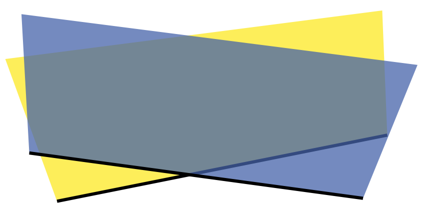
Was heißt das jetzt konkret?

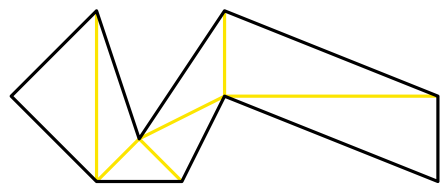
Basic Toolbox

- konvexe Hülle
- Linienschnitt
- Triangulierung
- Ebenenschnitt



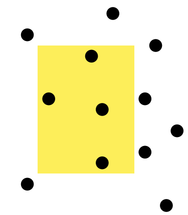


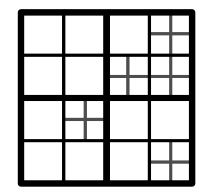


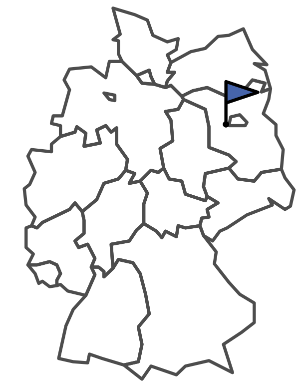


Geometrische Datenstrukturen

- orthogonal range searching
- Raum-Partitionierung
- Punkt-Lokalisierung

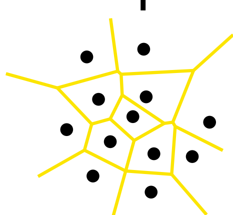


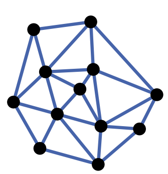


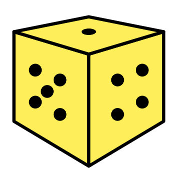


Erweiterte Toolbox

- Voronoi-Diagramme
- Delaunay-Triangulierung
- Randomisierte Algorithmen
- Komplexität

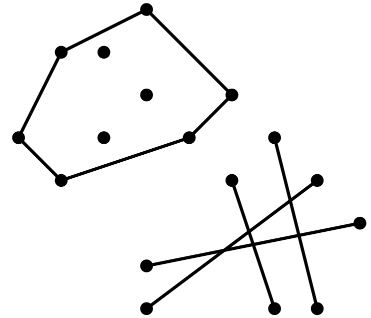
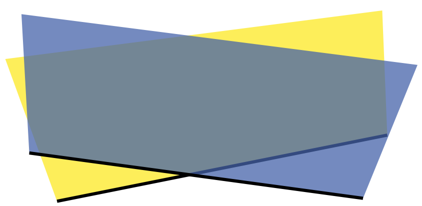
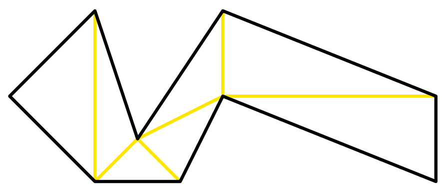




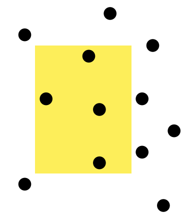
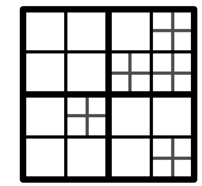
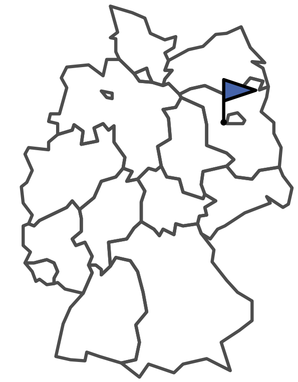


Was heißt das jetzt konkret?

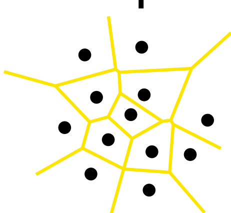
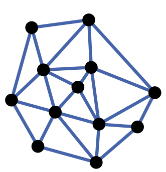
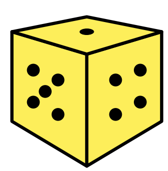
Basic Toolbox
- konvexe Hülle
- Linienschnitt
- Triangulierung
- Ebenenschnitt

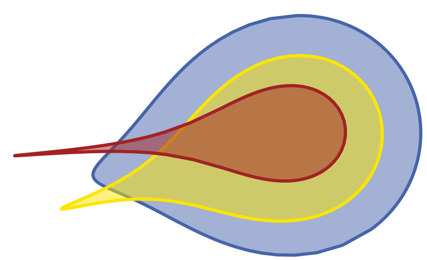
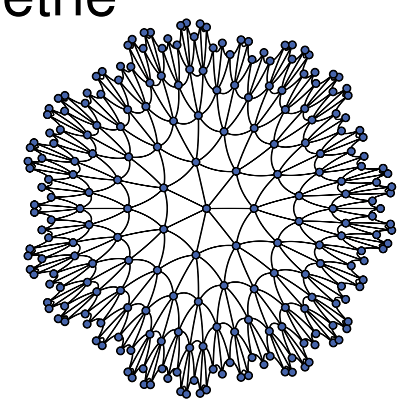
Geometrische Datenstrukturen
- orthogonal range searching
- Raum-Partitionierung
- Punkt-Lokalisierung

Erweiterte Toolbox
- Voronoi-Diagramme
- Delaunay-Triangulierung
- Randomisierte Algorithmen
- Komplexität

Verwandte Themen
- Was ist Geometrie überhaupt?
- hyperbolische Geometrie
- Grapheinbettungen

Organisatorisches

	Woche i							Woche $i+1$							Woche $i+2$							Woche $i+3$						
	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So
(i gerade)			Übungsblatt $\frac{i}{2}$														Übungsblatt $\frac{i+2}{2}$											

Vorlesung

- Vorlesung mit Folien

Organisatorisches

	Woche i							Woche $i+1$							Woche $i+2$							Woche $i+3$						
	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So
(i gerade)			Übungsblatt $\frac{i}{2}$														Übungsblatt $\frac{i+2}{2}$											

Vorlesung
- Vorlesung mit Folien

Übung (Woche $i+1$)
- gehalten von Marcus
- Besprechung Blatt $\frac{i}{2} - 1$
- Diskussion zu Blatt $\frac{i}{2}$

Organisatorisches

	Woche i							Woche $i+1$							Woche $i+2$							Woche $i+3$						
	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So
(i gerade)			Übungsblatt $\frac{i}{2}$														Übungsblatt $\frac{i+2}{2}$											

Vorlesung
- Vorlesung mit Folien

Übung (Woche $i+1$)
- gehalten von Marcus
- Besprechung Blatt $\frac{i}{2} - 1$
- Diskussion zu Blatt $\frac{i}{2}$

Aktiv-Session
- Erarbeitung weiterführender Aspekte
- Kuriositäten
- gemeinsames Papier-Lesen

Organisatorisches

	Woche i							Woche $i+1$							Woche $i+2$							Woche $i+3$						
	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So
(i gerade)		Übungsblatt $\frac{i}{2}$																Übungsblatt $\frac{i+2}{2}$										

Vorlesung

- Vorlesung mit Folien

Übung (Woche $i+1$)

- gehalten von Marcus
- Besprechung Blatt $\frac{i}{2} - 1$
- Diskussion zu Blatt $\frac{i}{2}$

Aktiv-Session

- Erarbeitung weiterführender Aspekte
- Kuriositäten
- gemeinsames Papier-Lesen

Materialien

- Vorlesungsfolien (meist schon kurz vor der Vorlesung online)

Organisatorisches

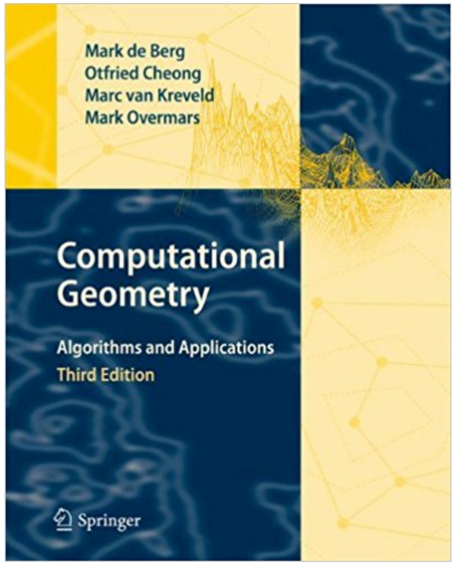
	Woche i							Woche $i+1$							Woche $i+2$							Woche $i+3$						
	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So
(i gerade)			Übungsblatt $\frac{i}{2}$														Übungsblatt $\frac{i+2}{2}$											

Vorlesung
- Vorlesung mit Folien

Übung (Woche $i+1$)
- gehalten von Marcus
- Besprechung Blatt $\frac{i}{2} - 1$
- Diskussion zu Blatt $\frac{i}{2}$

Aktiv-Session
- Erarbeitung weiterführender Aspekte
- Kuriositäten
- gemeinsames Papier-Lesen

Materialien
- Vorlesungsfolien (meist schon kurz vor der Vorlesung online)
- Buch *Computational Geometry*



Karlsruher Institut für Technologie

Organisatorisches

<table>
<tr><td colspan="7">Woche i</td><td colspan="7">Woche $i+1$</td><td colspan="7">Woche $i+2$</td><td colspan="7">Woche $i+3$</td></tr>
<tr><td>Mo</td><td>Di</td><td>Mi</td><td>Do</td><td>Fr</td><td>Sa</td><td>So</td><td>Mo</td><td>Di</td><td>Mi</td><td>Do</td><td>Fr</td><td>Sa</td><td>So</td><td>Mo</td><td>Di</td><td>Mi</td><td>Do</td><td>Fr</td><td>Sa</td><td>So</td><td>Mo</td><td>Di</td><td>Mi</td><td>Do</td><td>Fr</td><td>Sa</td><td>So</td></tr>
<tr><td colspan="2">(i gerade)</td><td colspan="14">Übungsblatt $\frac{i}{2}$</td><td colspan="12">Übungsblatt $\frac{i+2}{2}$</td></tr>
</table>

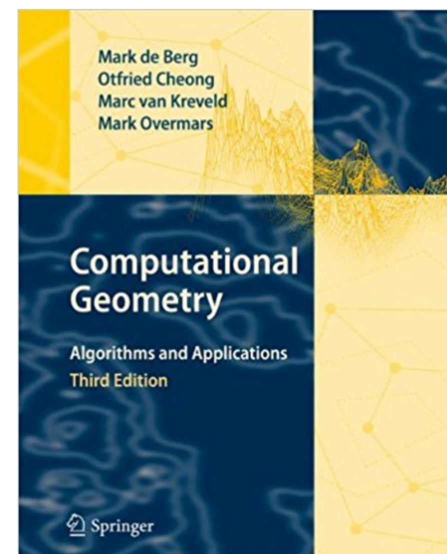
Vorlesung
- Vorlesung mit Folien

Übung (Woche $i+1$)
- gehalten von Marcus
- Besprechung Blatt $\frac{i}{2} - 1$
- Diskussion zu Blatt $\frac{i}{2}$

Aktiv-Session
- Erarbeitung weiterführender Aspekte
- Kuriositäten
- gemeinsames Papier-Lesen

Materialien
- Vorlesungsfolien (meist schon kurz vor der Vorlesung online)
- Buch *Computational Geometry*
- Musterlösungen: von euch erstellt

(mehr dazu am Freitag in der 0. Übung)



Organisatorisches

	Woche i							Woche $i+1$							Woche $i+2$							Woche $i+3$						
	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So
(i gerade)		Übungsblatt $\frac{i}{2}$															Übungsblatt $\frac{i+2}{2}$											

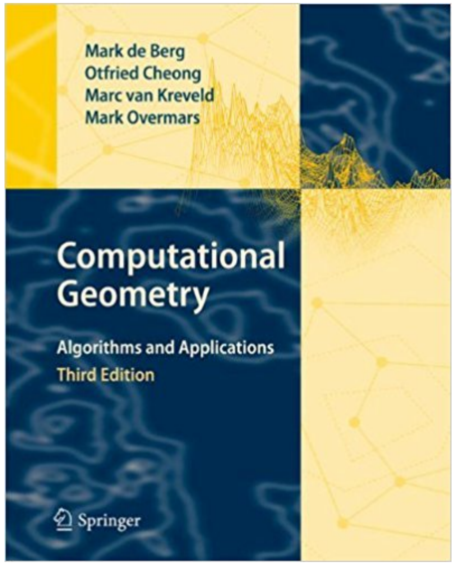
Vorlesung
- Vorlesung mit Folien

Übung (Woche $i+1$)
- gehalten von Marcus
- Besprechung Blatt $\frac{i}{2} - 1$
- Diskussion zu Blatt $\frac{i}{2}$

Aktiv-Session
- Erarbeitung weiterführender Aspekte
- Kuriositäten
- gemeinsames Papier-Lesen

Materialien
- Vorlesungsfolien (meist schon kurz vor der Vorlesung online)
- Buch *Computational Geometry*
- Musterlösungen: von euch erstellt

(mehr dazu am Freitag in der 0. Übung)



Prüfung
- mündlich
- Zulassungsvoraussetzung: 50% der Punkte bei den Übungsaufgaben

Online-Lehre

Vorteile

- gemütlich von zu Hause aus

Online-Lehre

Vorteile

- gemütlich von zu Hause aus
- ggf. einfacher zu verschieben

Online-Lehre

Vorteile

- gemütlich von zu Hause aus
- ggf. einfacher zu verschieben
- erleichtertes Namen lernen

Online-Lehre

Vorteile

- gemütlich von zu Hause aus
- ggf. einfacher zu verschieben
- erleichtertes Namen lernen

Häufige Nachteile

- 90 min Vorlesungs-Berieselung ist ermüdend

Online-Lehre

Vorteile

- gemütlich von zu Hause aus
- ggf. einfacher zu verschieben
- erleichtertes Namen lernen

Häufige Nachteile

- 90 min Vorlesungs-Berieselung ist ermüdend
- keine mündliche Interaktion: fehlende Übung für **mündliche** Prüfung

Online-Lehre

Vorteile

- gemütlich von zu Hause aus
- ggf. einfacher zu verschieben
- erleichtertes Namen lernen

Häufige Nachteile

- 90 min Vorlesungs-Berieselung ist ermüdend
- keine mündliche Interaktion: fehlende Übung für **mündliche** Prüfung
- erschwertes Kennenlernen

Online-Lehre

Vorteile

- gemütlich von zu Hause aus
- ggf. einfacher zu verschieben
- erleichtertes Namen lernen

Häufige Nachteile

- 90 min Vorlesungs-Berieselung ist ermüdend
- keine mündliche Interaktion: fehlende Übung für **mündliche** Prüfung
- erschwertes Kennenlernen
- visuelles Feedback für die Dozenten

Online-Lehre

Vorteile

- gemütlich von zu Hause aus
- ggf. einfacher zu verschieben
- erleichtertes Namen lernen

Häufige Nachteile

- 90 min Vorlesungs-Berieselung ist ermüdend
- keine mündliche Interaktion: fehlende Übung für **mündliche** Prüfung
- erschwertes Kennenlernen
- visuelles Feedback für die Dozenten

Maßnahmen

- Interaktion via mündlichem Feedback, Chat, Umfragen

Online-Lehre

Vorteile
- gemütlich von zu Hause aus
- ggf. einfacher zu verschieben
- erleichtertes Namen lernen

Häufige Nachteile
- 90 min Vorlesungs-Berieselung ist ermüdend
- keine mündliche Interaktion: fehlende Übung für **mündliche** Prüfung
- erschwertes Kennenlernen
- visuelles Feedback für die Dozenten

Maßnahmen
- Interaktion via mündlichem Feedback, Chat, Umfragen
- was ihr tun könnt:
  - Fragen stellen
  - auf unsere Fragen antworten/an der Diskussion beteiligen
  - Kamera anschalten

Online-Lehre

Vorteile
- gemütlich von zu Hause aus
- ggf. einfacher zu verschieben
- erleichtertes Namen lernen

Häufige Nachteile
- 90 min Vorlesungs-Berieselung ist ermüdend
- keine mündliche Interaktion: fehlende Übung für **mündliche** Prüfung
- erschwertes Kennenlernen
- visuelles Feedback für die Dozenten

Maßnahmen
- Interaktion via mündlichem Feedback, Chat, Umfragen
- was ihr tun könnt:
  - Fragen stellen
  - auf unsere Fragen antworten/an der Diskussion beteiligen
  - Kamera anschalten
- zum Kennenlernen 15 min vor dem Freitags-Termin: skribbl.io

Motivation

Öl in unterschiedlichen Zusammensetzungen

- die Zusammensetzung von Öl hängt von der Quelle ab, aus der es stammt
- Ziel: mische Öl aus verschiedenen Quellen, sodass die resultierende Zusammensetzung besonders gut zu verarbeiten ist

Motivation

Öl in unterschiedlichen Zusammensetzungen

- die Zusammensetzung von Öl hängt von der Quelle ab, aus der es stammt
- Ziel: mische Öl aus verschiedenen Quellen, sodass die resultierende Zusammensetzung besonders gut zu verarbeiten ist

Beispiel

- Öl enthält die Komponenten A und B
- zwei Quellen:

	A	B
Quelle 1	35%	10%
Quelle 2	20%	16%

Motivation

Öl in unterschiedlichen Zusammensetzungen

- die Zusammensetzung von Öl hängt von der Quelle ab, aus der es stammt
- Ziel: mische Öl aus verschiedenen Quellen, sodass die resultierende Zusammensetzung besonders gut zu verarbeiten ist

Beispiel

- Öl enthält die Komponenten A und B
- zwei Quellen:

	A	B
Quelle 1	35%	10%
Quelle 2	20%	16%

Können wir 30% von A und 12% von B erreichen?

Motivation

Karlsruher Institut für Technologie

Öl in unterschiedlichen Zusammensetzungen

- die Zusammensetzung von Öl hängt von der Quelle ab, aus der es stammt
- Ziel: mische Öl aus verschiedenen Quellen, sodass die resultierende Zusammensetzung besonders gut zu verarbeiten ist

Beispiel

- Öl enthält die Komponenten A und B
- zwei Quellen:

	A	B
Quelle 1	35%	10%
Quelle 2	20%	16%

Können wir 30% von A und 12% von B erreichen?

$2 : 1$

Motivation

Öl in unterschiedlichen Zusammensetzungen

- die Zusammensetzung von Öl hängt von der Quelle ab, aus der es stammt
- Ziel: mische Öl aus verschiedenen Quellen, sodass die resultierende Zusammensetzung besonders gut zu verarbeiten ist

Beispiel

- Öl enthält die Komponenten A und B
- zwei Quellen:

	A	B
Quelle 1	35%	10%
Quelle 2	20%	16%

Können wir 30% von A und 12% von B erreichen?

$2 : 1$

22% von A und 13% von B?

Motivation

Öl in unterschiedlichen Zusammensetzungen

- die Zusammensetzung von Öl hängt von der Quelle ab, aus der es stammt
- Ziel: mische Öl aus verschiedenen Quellen, sodass die resultierende Zusammensetzung besonders gut zu verarbeiten ist

Beispiel

- Öl enthält die Komponenten A und B
- zwei Quellen:
- dritte Quelle:

	A	B
Quelle 1	35%	10%
Quelle 2	20%	16%
Quelle 3	15%	7%

Können wir 30% von A und 12% von B erreichen?

2 : 1

22% von A und 13% von B?

Motivation

Öl in unterschiedlichen Zusammensetzungen

- die Zusammensetzung von Öl hängt von der Quelle ab, aus der es stammt
- Ziel: mische Öl aus verschiedenen Quellen, sodass die resultierende Zusammensetzung besonders gut zu verarbeiten ist

Beispiel

- Öl enthält die Komponenten A und B
- zwei Quellen:
- dritte Quelle:

	A	B
Quelle 1	35%	10%
Quelle 2	20%	16%
Quelle 3	15%	7%

Können wir 30% von A und 12% von B erreichen?

$2 : 1$

22% von A und 13% von B?

$1 : 3 : 1$

Motivation

Öl in unterschiedlichen Zusammensetzungen

- die Zusammensetzung von Öl hängt von der Quelle ab, aus der es stammt
- Ziel: mische Öl aus verschiedenen Quellen, sodass die resultierende Zusammensetzung besonders gut zu verarbeiten ist

Beispiel

- Öl enthält die Komponenten A und B
- zwei Quellen:
- dritte Quelle:

	A	B
Quelle 1	35%	10%
Quelle 2	20%	16%
Quelle 3	15%	7%

Können wir 30% von A und 12% von B erreichen? 2 : 1

22% von A und 13% von B? 1 : 3 : 1

Was hat das mit Geometrie zu tun?

Motivation

Öl in unterschiedlichen Zusammensetzungen

- die Zusammensetzung von Öl hängt von der Quelle ab, aus der es stammt
- Ziel: mische Öl aus verschiedenen Quellen, sodass die resultierende Zusammensetzung besonders gut zu verarbeiten ist

Beispiel

- Öl enthält die Komponenten A und B
- zwei Quellen:

	A	B
Quelle 1	35%	10%
Quelle 2	20%	16%
Quelle 3	15%	7%

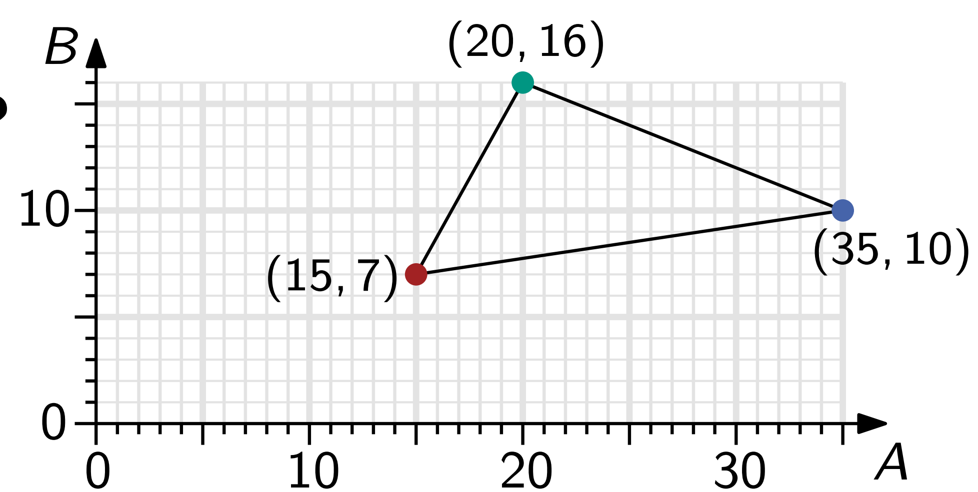
- dritte Quelle: Quelle 3

Können wir 30% von A und 12% von B erreichen? 2 : 1

22% von A und 13% von B? 1 : 3 : 1

Was hat das mit Geometrie zu tun?

- Mischverhältnis kann als Punkt interpretiert werden



Motivation

Öl in unterschiedlichen Zusammensetzungen

- die Zusammensetzung von Öl hängt von der Quelle ab, aus der es stammt
- Ziel: mische Öl aus verschiedenen Quellen, sodass die resultierende Zusammensetzung besonders gut zu verarbeiten ist

Beispiel

- Öl enthält die Komponenten A und B
- zwei Quellen:
- dritte Quelle:

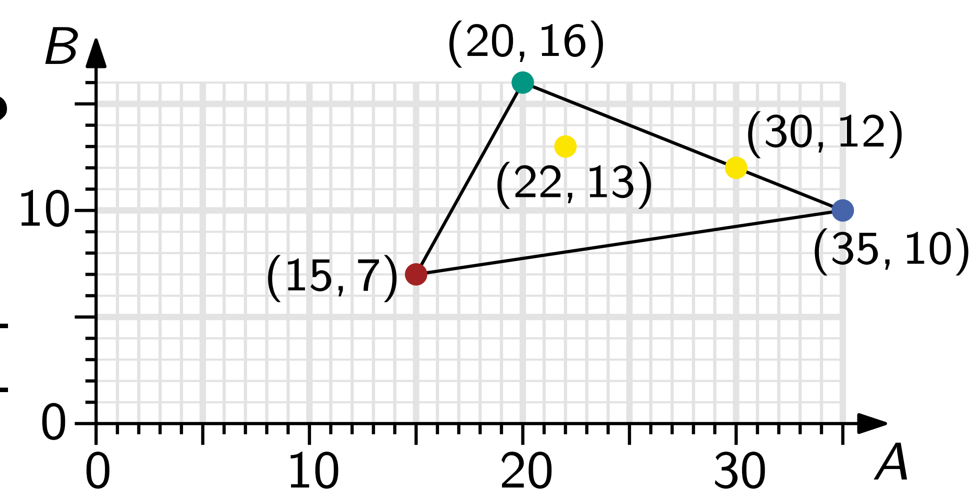
	A	B
Quelle 1	35%	10%
Quelle 2	20%	16%
Quelle 3	15%	7%

Können wir 30% von A und 12% von B erreichen? 2 : 1

22% von A und 13% von B? 1 : 3 : 1

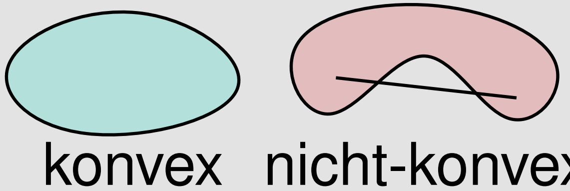
Was hat das mit Geometrie zu tun?

- Mischverhältnis kann als Punkt interpretiert werden
- Mischung ist möglich $\Leftrightarrow$ zugehöriger Punkt liegt „zwischen“ den verfügbaren Punkten



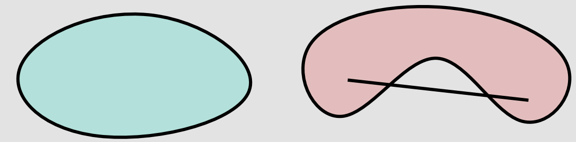
Konvexe Hülle

Definition
Eine Punktemenge $P \subseteq \mathbb{R}^d$ ist **konvex** wenn für je zwei Punkte $p, q \in P$ auch die Strecke pq in P liegt.



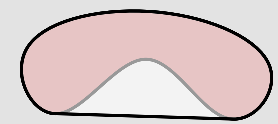
Konvexe Hülle

Definition
Eine Punktemenge $P \subseteq \mathbb{R}^d$ ist **konvex** wenn für je zwei Punkte $p, q \in P$ auch die Strecke pq in P liegt.



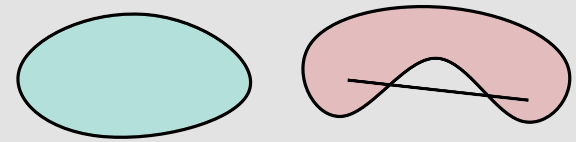
konvex nicht-konvex

Definition
Für $P \subseteq \mathbb{R}^d$ ist die **Konvexe Hülle $\boldsymbol{\mathcal{CH}(P)}$** die inklusionsminimale Teilmenge von $\mathbb{R}^d$, sodass $\mathcal{CH}(P)$ konvex ist und $P \subseteq \mathcal{CH}(H)$.



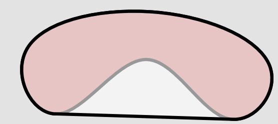
Konvexe Hülle

Definition
Eine Punktemenge $P \subseteq \mathbb{R}^d$ ist **konvex** wenn für je zwei Punkte $p, q \in P$ auch die Strecke pq in P liegt.



konvex nicht-konvex

Definition
Für $P \subseteq \mathbb{R}^d$ ist die **Konvexe Hülle $\mathcal{CH}(P)$** die inklusionsminimale Teilmenge von $\mathbb{R}^d$, sodass $\mathcal{CH}(P)$ konvex ist und $P \subseteq \mathcal{CH}(H)$.

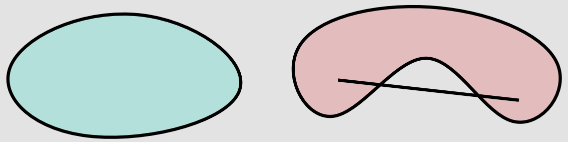


Äquivalente Definitionen

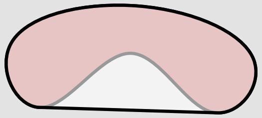
- der Schnitt aller konvexer Mengen aus $\mathbb{R}^d$, die P enthalten

Konvexe Hülle

Definition
Eine Punktemenge $P \subseteq \mathbb{R}^d$ ist **konvex** wenn für je zwei Punkte $p, q \in P$ auch die Strecke pq in P liegt.

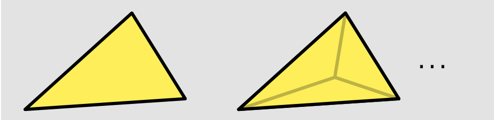


Definition
Für $P \subseteq \mathbb{R}^d$ ist die **Konvexe Hülle $\mathcal{CH}(P)$** die inklusionsminimale Teilmenge von $\mathbb{R}^d$, sodass $\mathcal{CH}(P)$ konvex ist und $P \subseteq \mathcal{CH}(H)$.



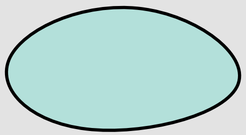
Äquivalente Definitionen

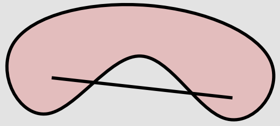
- der Schnitt aller konvexer Mengen aus $\mathbb{R}^d$, die P enthalten
- die Vereinigung aller Simplexe mit Eckpunkten in P



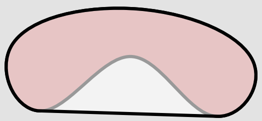
Konvex Hülle

Definition
Eine Punktemenge $P \subseteq \mathbb{R}^d$ ist **konvex** wenn für je zwei Punkte $p, q \in P$ auch die Strecke pq in P liegt.


konvex

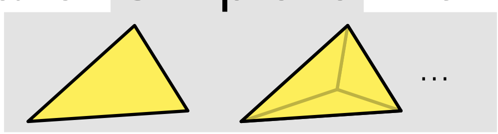

nicht-konvex

Definition
Für $P \subseteq \mathbb{R}^d$ ist die **Konvexe Hülle $\mathcal{CH}(P)$** die inklusionsminimale Teilmenge von $\mathbb{R}^d$, sodass $\mathcal{CH}(P)$ konvex ist und $P \subseteq \mathcal{CH}(H)$.



Äquivalente Definitionen

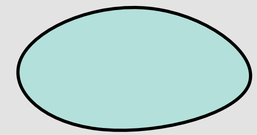
- der Schnitt aller konvexer Mengen aus $\mathbb{R}^d$, die P enthalten
- die Vereinigung aller Simplexe mit Eckpunkten in P

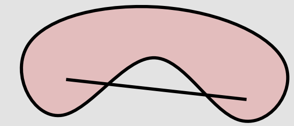


- Menge aller Punkte, die Konvexkombinationen von Punkten aus P sind

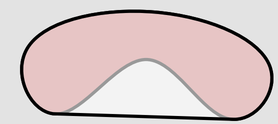
Konvexe Hülle

Definition
Eine Punktemenge $P \subseteq \mathbb{R}^d$ ist **konvex** wenn für je zwei Punkte $p, q \in P$ auch die Strecke pq in P liegt.


konvex

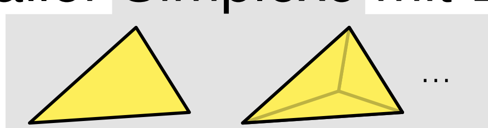

nicht-konvex

Definition
Für $P \subseteq \mathbb{R}^d$ ist die **Konvexe Hülle $\mathcal{CH}(P)$** die inklusionsminimale Teilmenge von $\mathbb{R}^d$, sodass $\mathcal{CH}(P)$ konvex ist und $P \subseteq \mathcal{CH}(H)$.



Äquivalente Definitionen

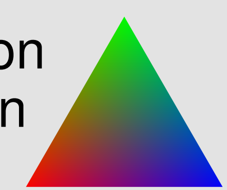
- der Schnitt aller konvexer Mengen aus $\mathbb{R}^d$, die P enthalten
- die Vereinigung aller Simplexe mit Eckpunkten in P



- Menge aller Punkte, die Konvexkombinationen von Punkten aus P sind

$$\sum_{i=1}^{n} a_i \cdot p_i \text{ mit } p_i \in P, 0 \leq a_i \in \mathbb{R} \text{ und } \sum_{i=1}^{n} a_i = 1$$

kennt ihr ggf. von baryzentrischen Koordinaten



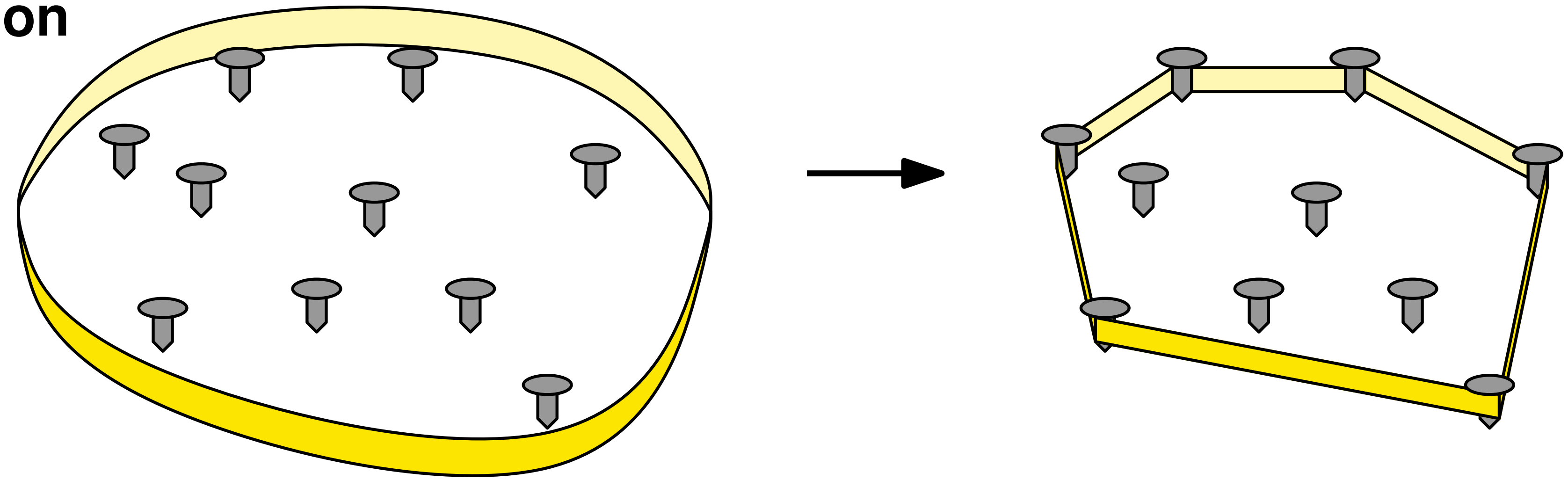
Konvexe Hülle – Trivialer Algorithmus

Problem: KONVEXE HÜLLE (2D)
Gegeben n Punkte $P \subseteq \mathbb{R}^2$, berechne die konvexe Hülle $\mathcal{CH}(P)$.

Konvexe Hülle – Trivialer Algorithmus

Problem: Konvexe Hülle (2D)
Gegeben n Punkte $P \subseteq \mathbb{R}^2$, berechne die konvexe Hülle $\mathcal{CH}(P)$.

Intuition

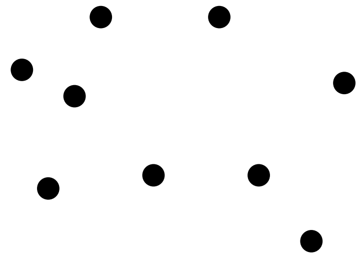


Konvexe Hülle – Trivialer Algorithmus

Problem: KONVEXE HÜLLE (2D)
Gegeben n Punkte $P \subseteq \mathbb{R}^2$, berechne die konvexe Hülle $\mathcal{CH}(P)$.

Anmerkung & Beobachtung

- Annahme: die Punkte sind in allgemeiner Lage

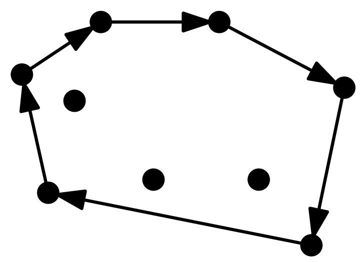


Konvexe Hülle – Trivialer Algorithmus

Problem: KONVEXE HÜLLE (2D)
Gegeben n Punkte $P \subseteq \mathbb{R}^2$, berechne die konvexe Hülle $\mathcal{CH}(P)$.

Anmerkung & Beobachtung

- Annahme: die Punkte sind in allgemeiner Lage
- $\mathcal{CH}(P)$ ist durch ein Polygon begrenzt
  $\rightarrow$ gewünschte Ausgabe ist eine Folge von Punkten

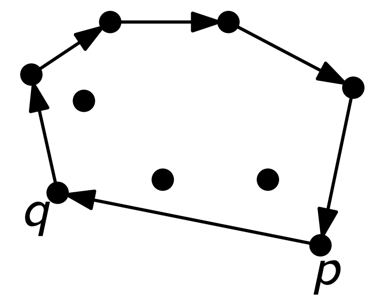


Konvexe Hülle – Trivialer Algorithmus

Problem: KONVEXE HÜLLE (2D)
Gegeben n Punkte $P \subseteq \mathbb{R}^2$, berechne die konvexe Hülle $\mathcal{CH}(P)$.

Anmerkung & Beobachtung

- Annahme: die Punkte sind in allgemeiner Lage
- $\mathcal{CH}(P)$ ist durch ein Polygon begrenzt
  $\rightarrow$ gewünschte Ausgabe ist eine Folge von Punkten
- wann ist pq eine Kante von $\mathcal{CH}(P)$?

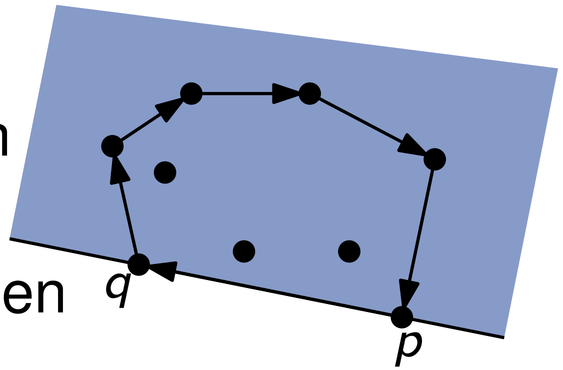


Konvexe Hülle – Trivialer Algorithmus

Problem: KONVEXE HÜLLE (2D)
Gegeben n Punkte $P \subseteq \mathbb{R}^2$, berechne die konvexe Hülle $\mathcal{CH}(P)$.

Anmerkung & Beobachtung

- Annahme: die Punkte sind in allgemeiner Lage
- $\mathcal{CH}(P)$ ist durch ein Polygon begrenzt
  $\rightarrow$ gewünschte Ausgabe ist eine Folge von Punkten
- wann ist pq eine Kante von $\mathcal{CH}(P)$?
  - wenn alle Punkte im Halbraum rechts von pq liegen

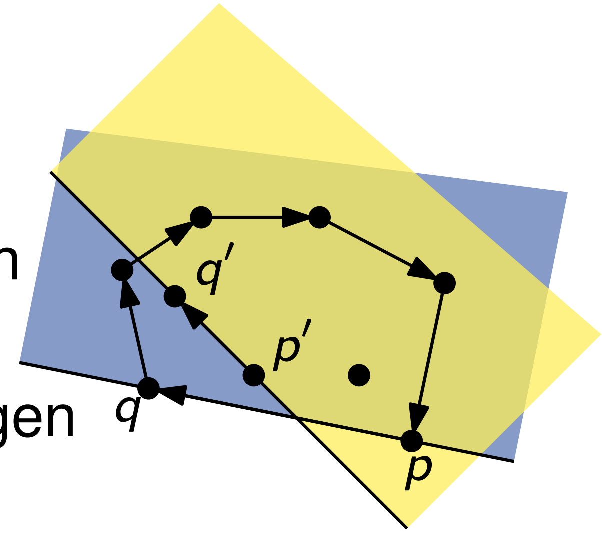


Konvexe Hülle – Trivialer Algorithmus

Problem: KONVEXE HÜLLE (2D)
Gegeben n Punkte $P \subseteq \mathbb{R}^2$, berechne die konvexe Hülle $\mathcal{CH}(P)$.

Anmerkung & Beobachtung

- Annahme: die Punkte sind in allgemeiner Lage
- $\mathcal{CH}(P)$ ist durch ein Polygon begrenzt
  $\rightarrow$ gewünschte Ausgabe ist eine Folge von Punkten
- wann ist pq eine Kante von $\mathcal{CH}(P)$?
  - wenn alle Punkte im Halbraum rechts von pq liegen

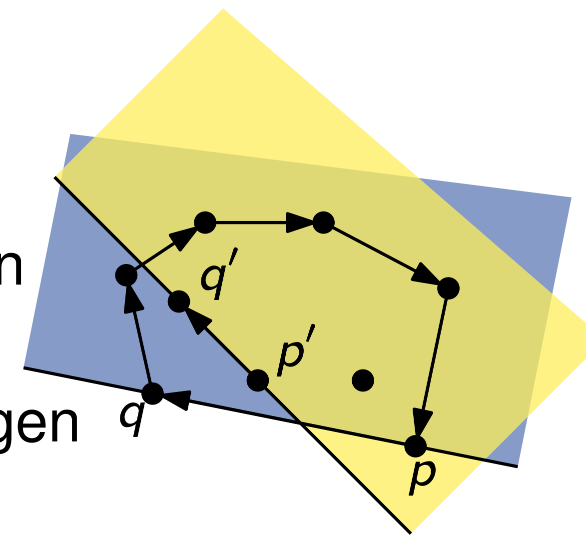


Konvexe Hülle – Trivialer Algorithmus

Problem: KONVEXE HÜLLE (2D)
Gegeben n Punkte $P \subseteq \mathbb{R}^2$, berechne die konvexe Hülle $\mathcal{CH}(P)$.

Anmerkung & Beobachtung

- Annahme: die Punkte sind in allgemeiner Lage
- $\mathcal{CH}(P)$ ist durch ein Polygon begrenzt
  $\rightarrow$ gewünschte Ausgabe ist eine Folge von Punkten
- wann ist pq eine Kante von $\mathcal{CH}(P)$?
  - wenn alle Punkte im Halbraum rechts von pq liegen



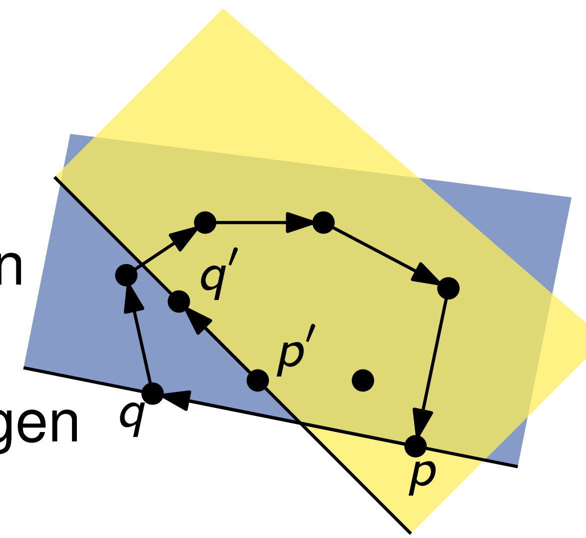
Trivialer Algorithmus

Konvexe Hülle – Trivialer Algorithmus

Problem: KONVEXE HÜLLE (2D)
Gegeben n Punkte $P \subseteq \mathbb{R}^2$, berechne die konvexe Hülle $\mathcal{CH}(P)$.

Anmerkung & Beobachtung

- Annahme: die Punkte sind in allgemeiner Lage
- $\mathcal{CH}(P)$ ist durch ein Polygon begrenzt
  $\rightarrow$ gewünschte Ausgabe ist eine Folge von Punkten
- wann ist pq eine Kante von $\mathcal{CH}(P)$?
  - wenn alle Punkte im Halbraum rechts von pq liegen



Trivialer Algorithmus

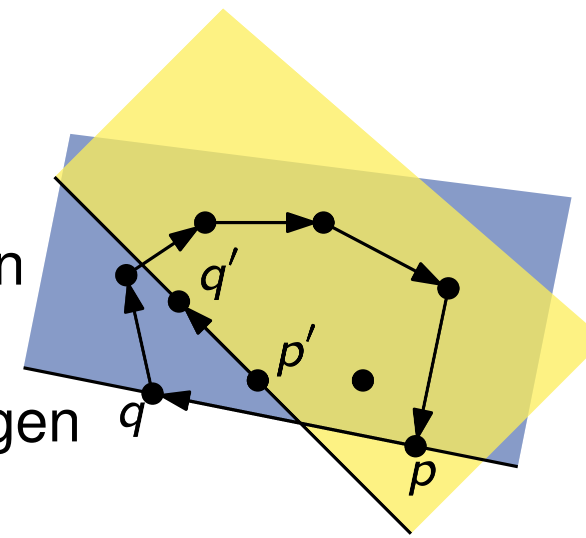
- iteriere über alle Knotenpaare $p, q \in P$ (gerichtet)
  - teste ob alle Punkte rechts von pq liegen
  - falls ja: speicher pq

Konvexe Hülle – Triviales Algorithmus

Problem: KONVEXE HÜLLE (2D)
Gegeben n Punkte $P \subseteq \mathbb{R}^2$, berechne die konvexe Hülle $\mathcal{CH}(P)$.

Anmerkung & Beobachtung

- Annahme: die Punkte sind in allgemeiner Lage
- $\mathcal{CH}(P)$ ist durch ein Polygon begrenzt
  $\rightarrow$ gewünschte Ausgabe ist eine Folge von Punkten
- wann ist pq eine Kante von $\mathcal{CH}(P)$?
  - wenn alle Punkte im Halbraum rechts von pq liegen



Trivialer Algorithmus

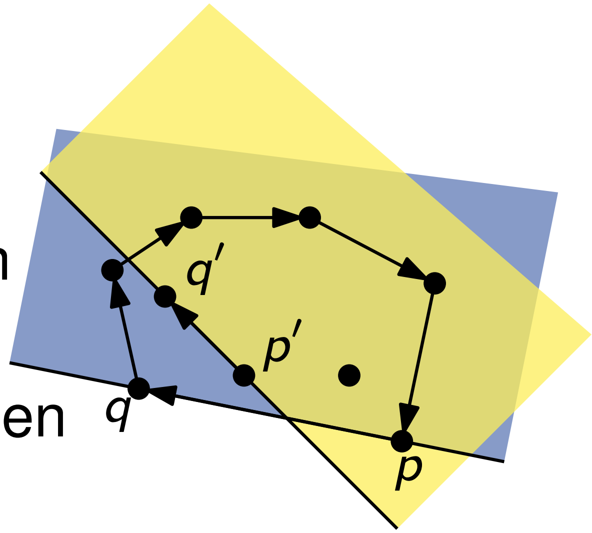
- iteriere über alle Knotenpaare $p, q \in P$ (gerichtet)
  - teste ob alle Punkte rechts von pq liegen
  - falls ja: speicher pq
- konstruiere das Polygon (als Punktfolge) aus den gespeicherten Kanten

Konvexe Hülle – Trivialer Algorithmus

Problem: KONVEXE HÜLLE (2D)
Gegeben n Punkte $P \subseteq \mathbb{R}^2$, berechne die konvexe Hülle $\mathcal{CH}(P)$.

Anmerkung & Beobachtung

- Annahme: die Punkte sind in allgemeiner Lage
- $\mathcal{CH}(P)$ ist durch ein Polygon begrenzt
  $\rightarrow$ gewünschte Ausgabe ist eine Folge von Punkten
- wann ist pq eine Kante von $\mathcal{CH}(P)$?
  - wenn alle Punkte im Halbraum rechts von pq liegen



Trivialer Algorithmus

- iteriere über alle Knotenpaare $p, q \in P$ (gerichtet)
  - teste ob alle Punkte rechts von pq liegen
  - falls ja: speicher pq
- konstruiere das Polygon (als Punktfolge) aus den gespeicherten Kanten

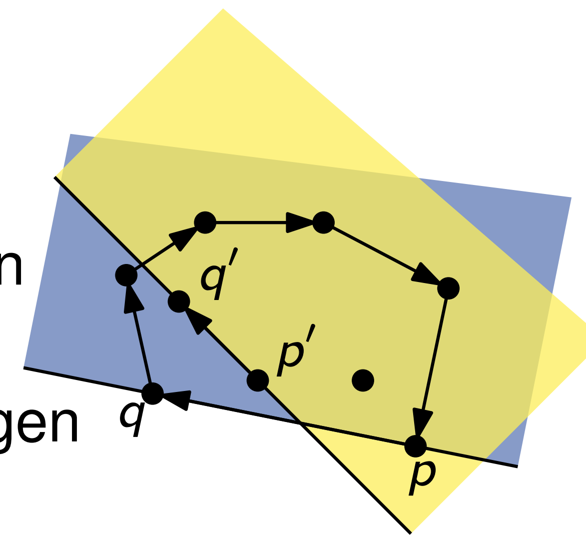
Laufzeit:

Konvexe Hülle – Trivialer Algorithmus

Problem: KONVEXE HÜLLE (2D)
Gegeben n Punkte $P \subseteq \mathbb{R}^2$, berechne die konvexe Hülle $\mathcal{CH}(P)$.

Anmerkung & Beobachtung

- Annahme: die Punkte sind in allgemeiner Lage
- $\mathcal{CH}(P)$ ist durch ein Polygon begrenzt
  $\rightarrow$ gewünschte Ausgabe ist eine Folge von Punkten
- wann ist pq eine Kante von $\mathcal{CH}(P)$?
  - wenn alle Punkte im Halbraum rechts von pq liegen



Trivialer Algorithmus

- iteriere über alle Knotenpaare $p, q \in P$ (gerichtet)
  - teste ob alle Punkte rechts von pq liegen
  - falls ja: speicher pq
- konstruiere das Polygon (als Punktfolge) aus den gespeicherten Kanten

Laufzeit: $O\left(n^3\right)$

Konvexe Hülle – Trivialer Algorithmus

Trivialer Algorithmus

- iteriere über alle Knotenpaare $p, q \in P$ (gerichtet)
  - teste ob alle Punkte rechts von pq liegen
  - falls ja: speicher pq
- konstruiere das Polygon (als Punktfolge) aus den gespeicherten Kanten

Probleme

- der Algorithmus ist langsam

Konvexe Hülle – Trivialer Algorithmus

Trivialer Algorithmus

- iteriere über alle Knotenpaare $p, q \in P$ (gerichtet)
  - teste ob alle Punkte rechts von pq liegen
  - falls ja: speicher pq
- konstruiere das Polygon (als Punktfolge) aus den gespeicherten Kanten

Probleme

- der Algorithmus ist langsam
- der Algorithmus ist nicht robust

Konvexe Hülle – Trivialer Algorithmus

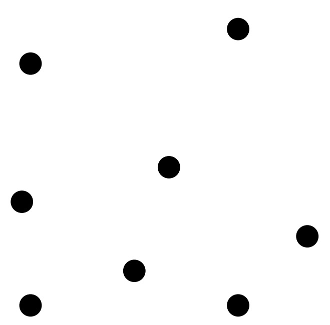
Trivialer Algorithmus

- iteriere über alle Knotenpaare $p, q \in P$ (gerichtet)
  - teste ob alle Punkte rechts von pq liegen
  - falls ja: speicher pq
- konstruiere das Polygon (als Punktfolge) aus den gespeicherten Kanten

Probleme

- der Algorithmus ist langsam
- der Algorithmus ist nicht robust

Beispiel (Robustheit)



Konvexe Hülle – Trivialer Algorithmus

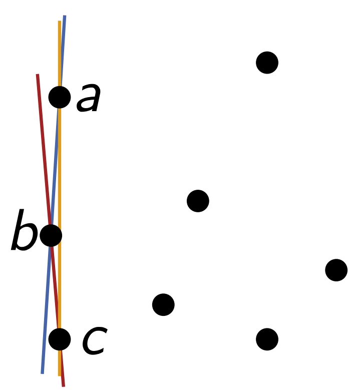
Trivialer Algorithmus

- iteriere über alle Knotenpaare $p, q \in P$ (gerichtet)
  - teste ob alle Punkte rechts von pq liegen
  - falls ja: speicher pq
- konstruiere das Polygon (als Punktfolge) aus den gespeicherten Kanten

Probleme

- der Algorithmus ist langsam
- der Algorithmus ist nicht robust

Beispiel (Robustheit)



- drei der Entscheidungen „liegt rechts von“ sind knapp

Konvexe Hülle – Trivialer Algorithmus

Trivialer Algorithmus

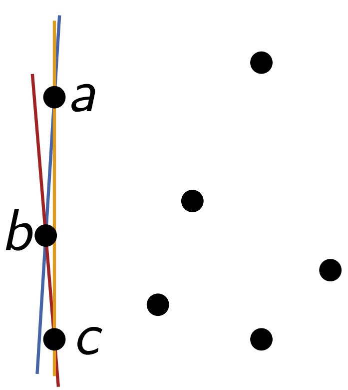
- iteriere über alle Knotenpaare $p, q \in P$ (gerichtet)
  - teste ob alle Punkte rechts von pq liegen
  - falls ja: speicher pq
- konstruiere das Polygon (als Punktfolge) aus den gespeicherten Kanten

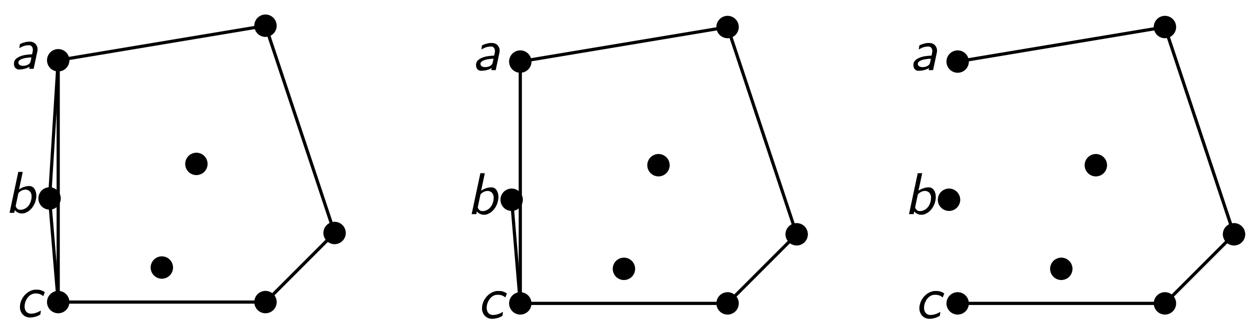
Probleme

- der Algorithmus ist langsam
- der Algorithmus ist nicht robust

Beispiel (Robustheit)

- drei der Entscheidungen „liegt rechts von“ sind knapp
- falsche Entscheidungen liefern ggf. kein Polygon

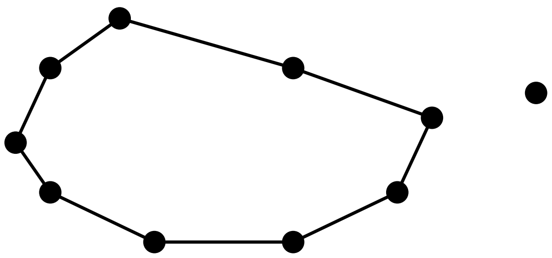




Graham Scan

Idee: iteratives Vorgehen

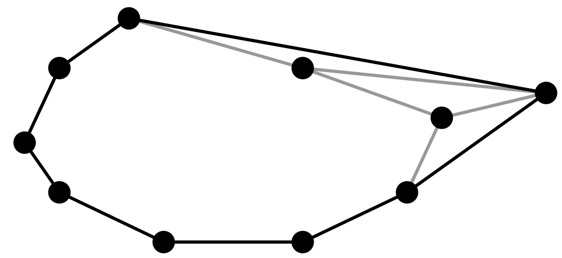
- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt



Graham Scan

Idee: iteratives Vorgehen

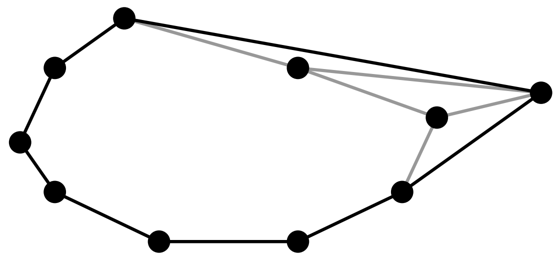
- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt



Graham Scan

Idee: iteratives Vorgehen

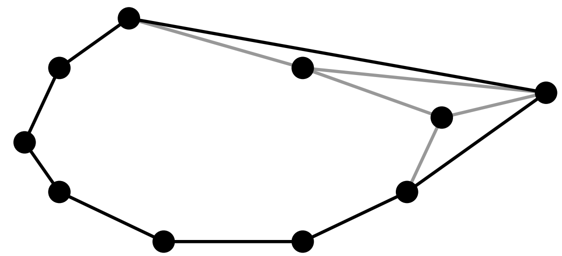
- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke

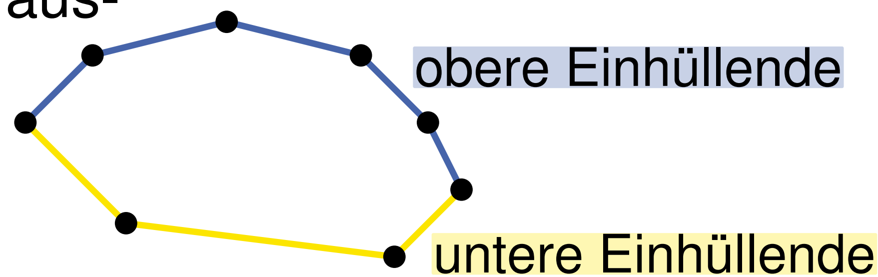


Graham Scan

Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende

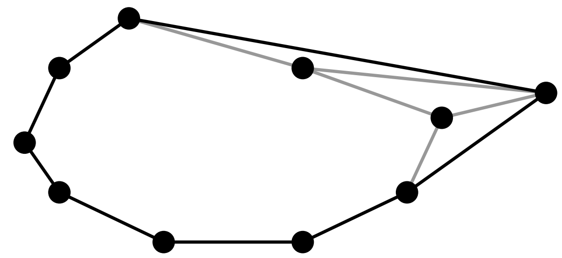


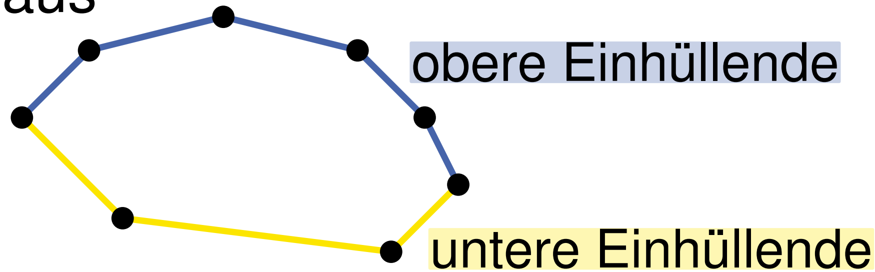


Graham Scan

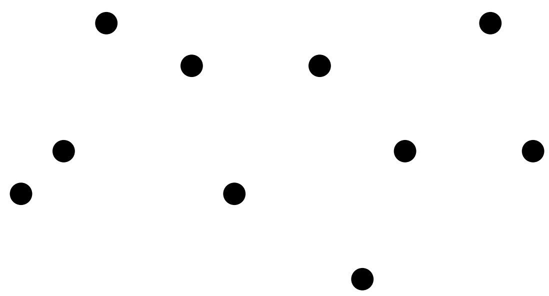
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel

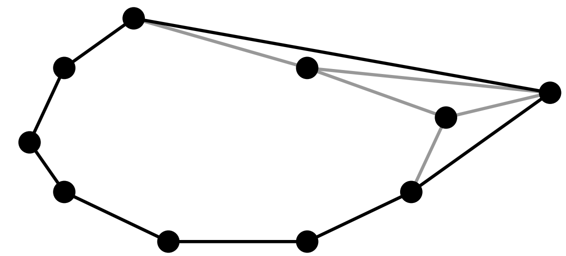


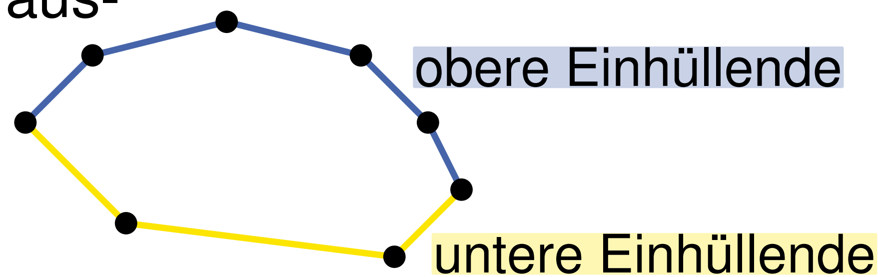
Graham Scan

Graham Scan

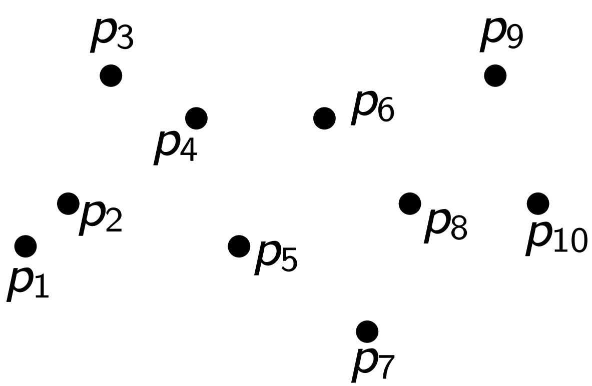
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



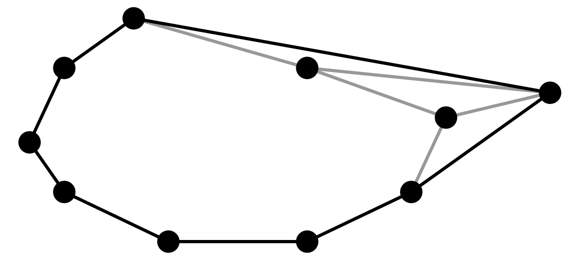
Graham Scan

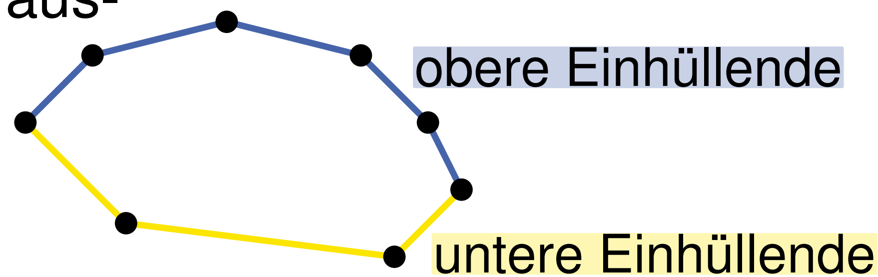
- sortiere P (bzgl. x): $p_1, \ldots, p_n$

Graham Scan

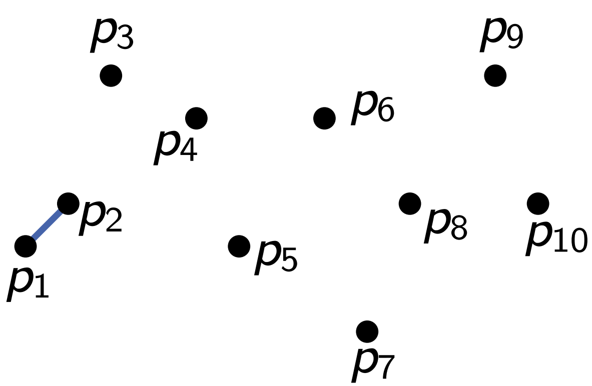
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



L: p_1 p_2

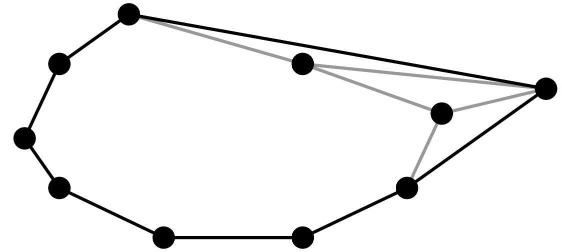
Graham Scan

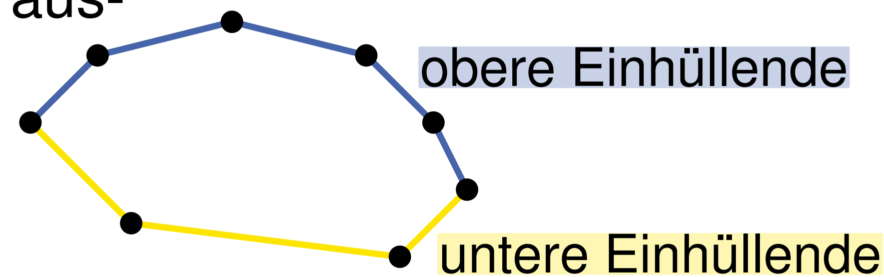
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein

Graham Scan

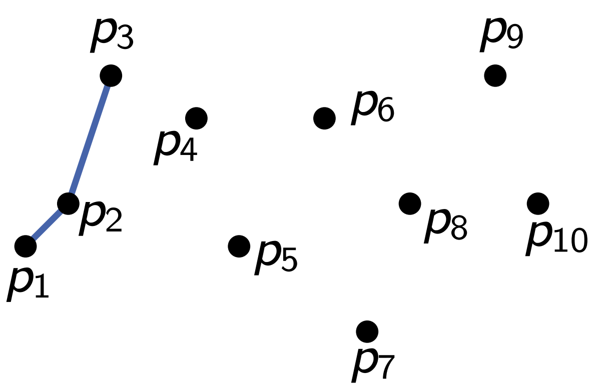
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



L: p_1 p_2 p_3

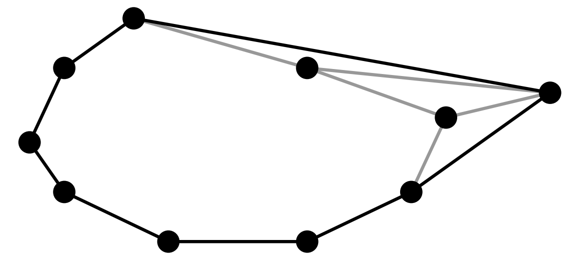
Graham Scan

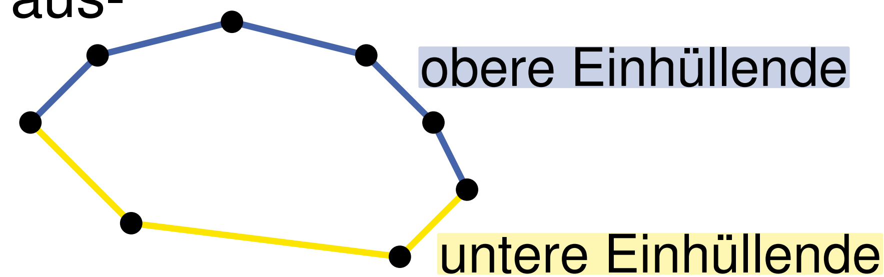
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein

Graham Scan

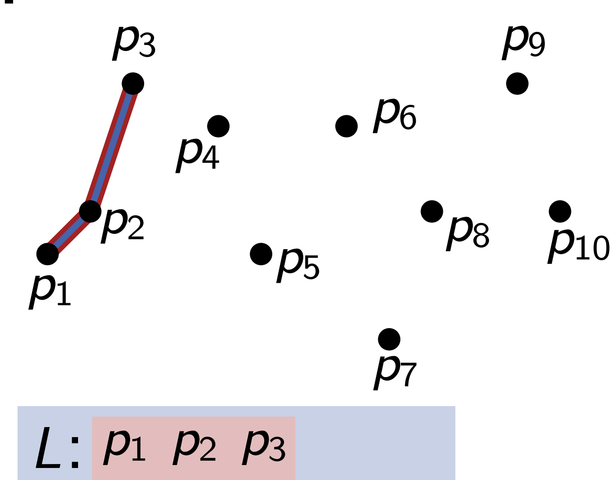
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



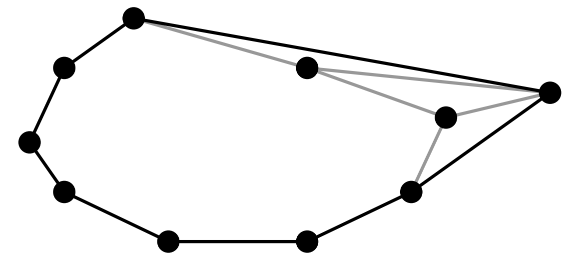
Graham Scan

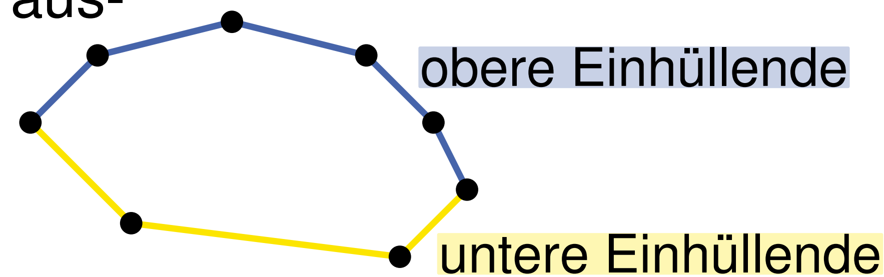
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

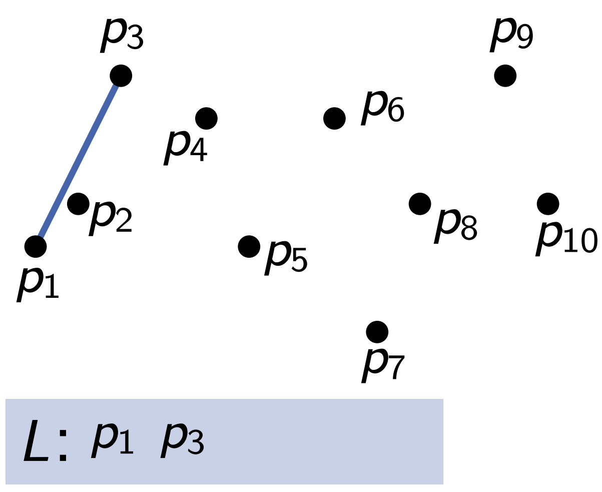
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



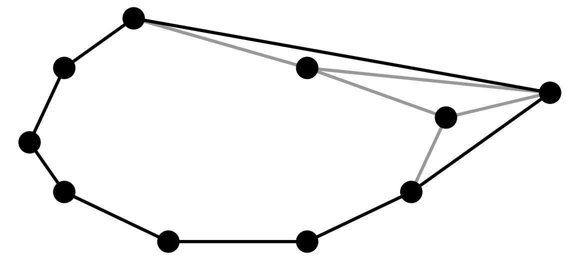
Graham Scan

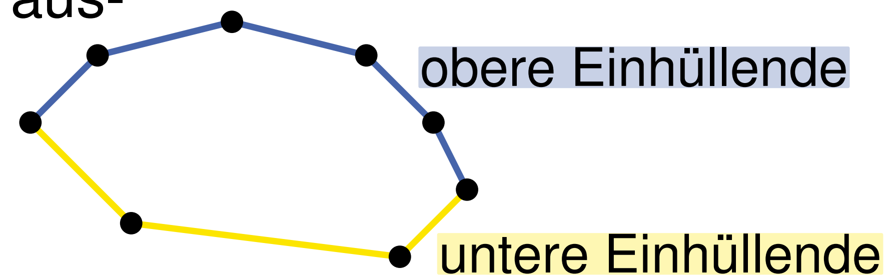
- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

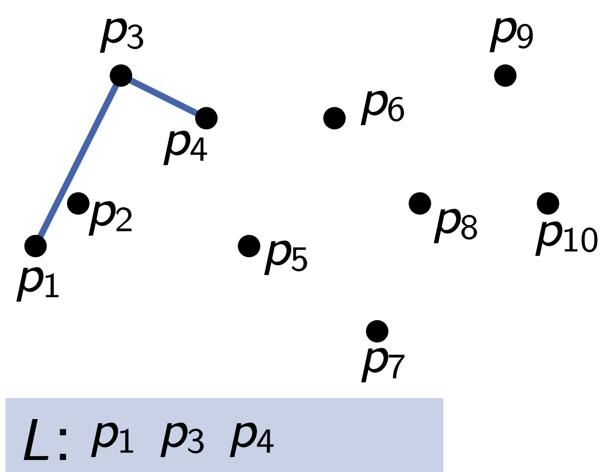
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



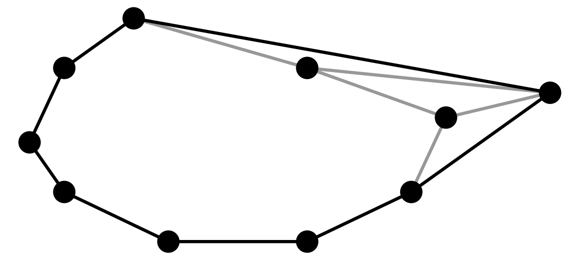
Graham Scan

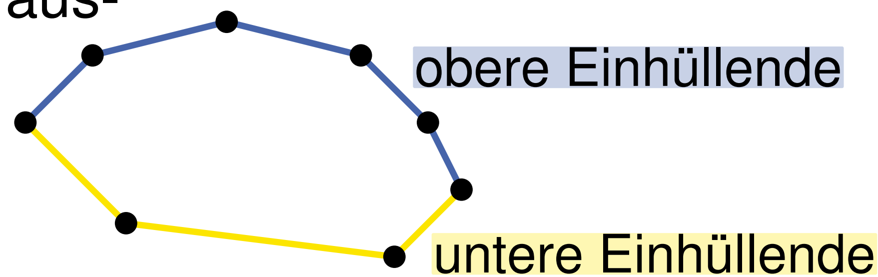
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

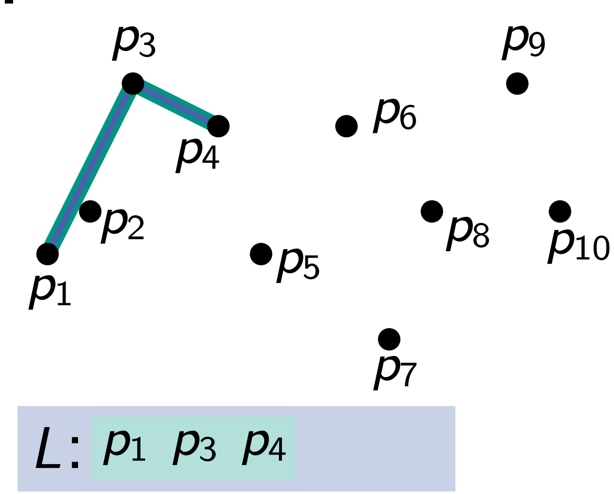
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



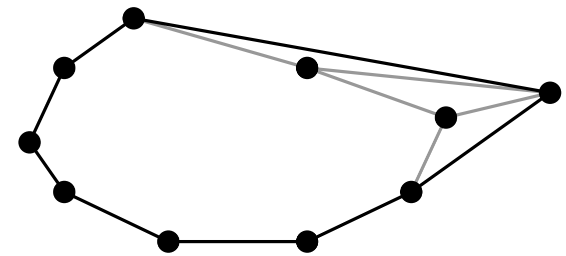
Graham Scan

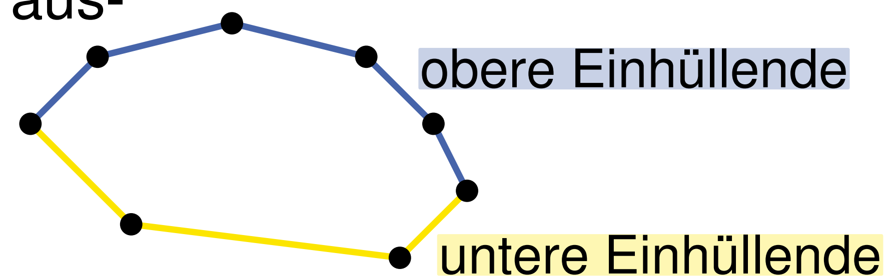
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

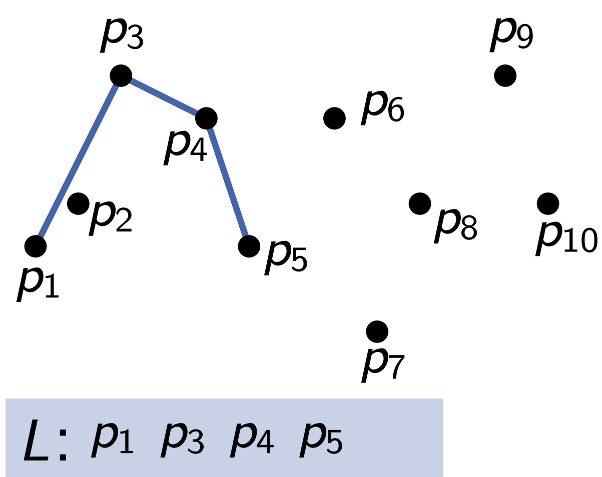
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



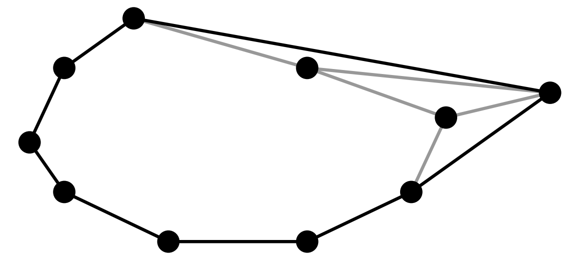
Graham Scan

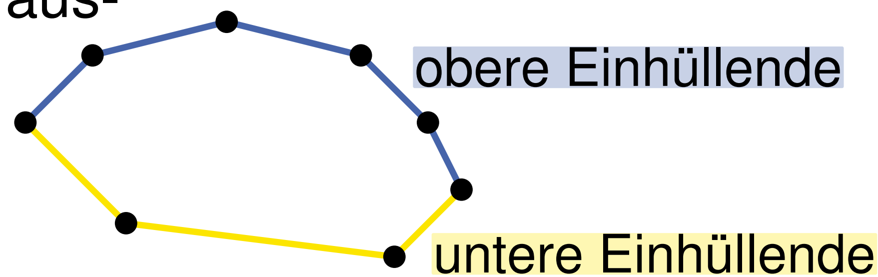
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

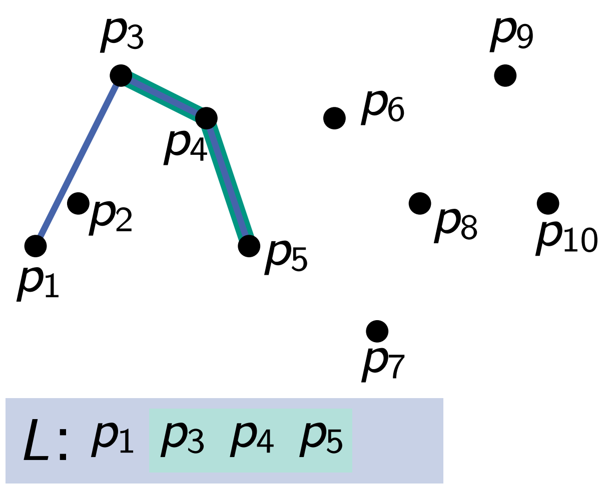
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



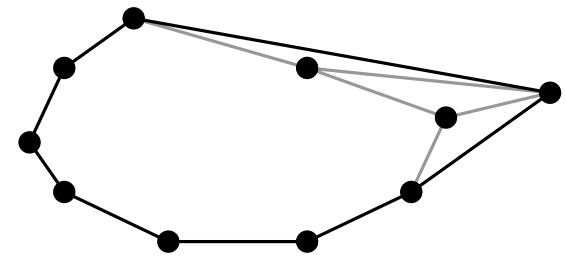
Graham Scan

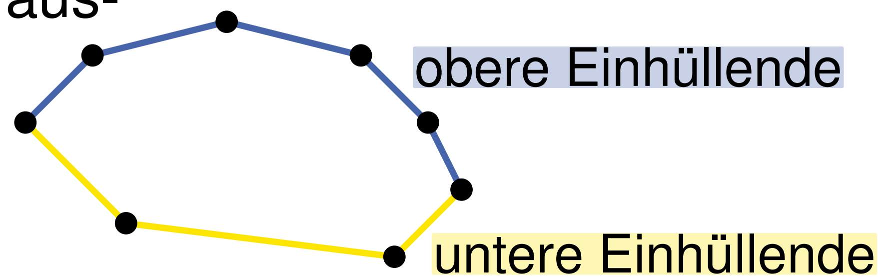
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

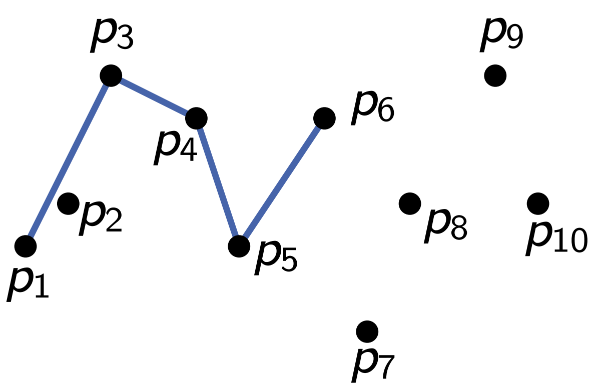
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



L: p_1 p_3 p_4 p_5 p_6

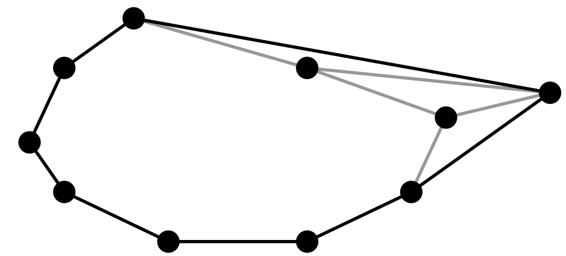
Graham Scan

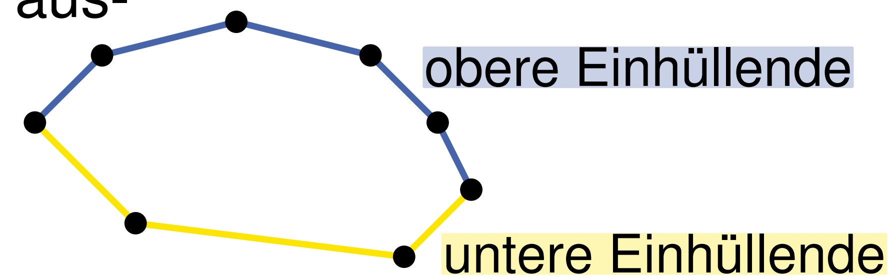
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

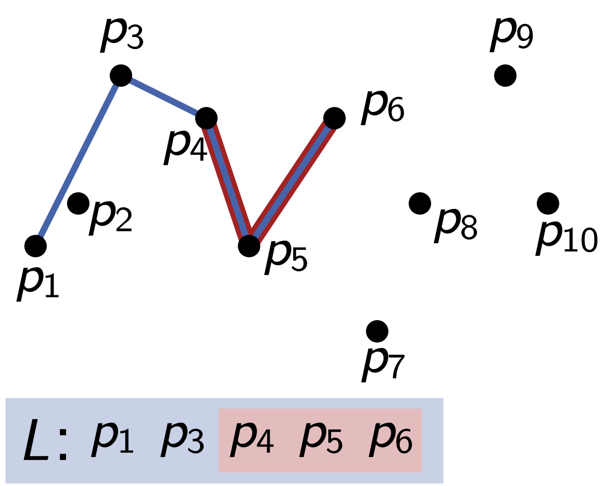
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



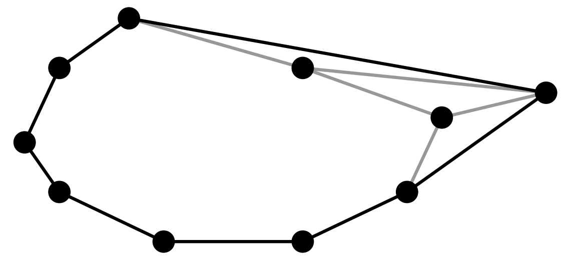
Graham Scan

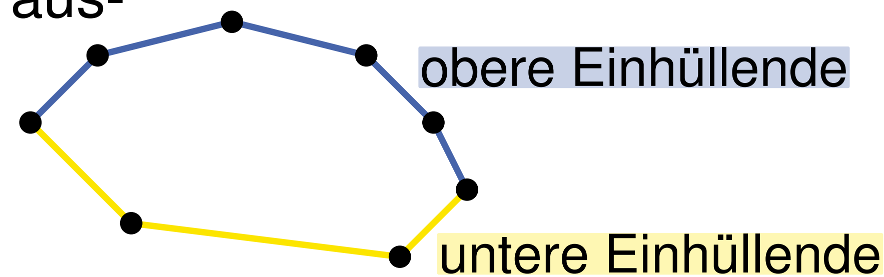
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

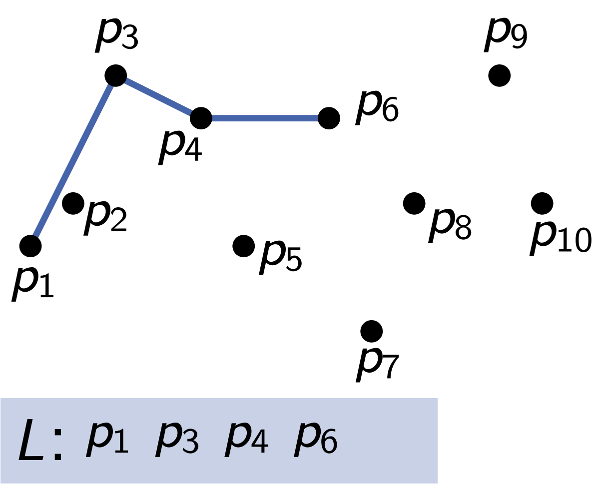
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





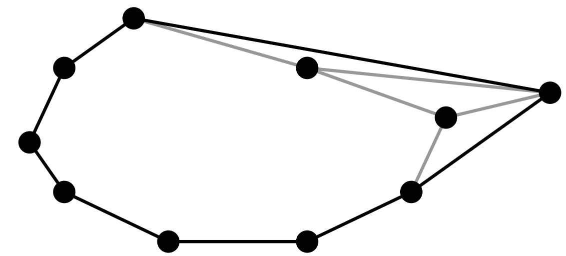
Beispiel

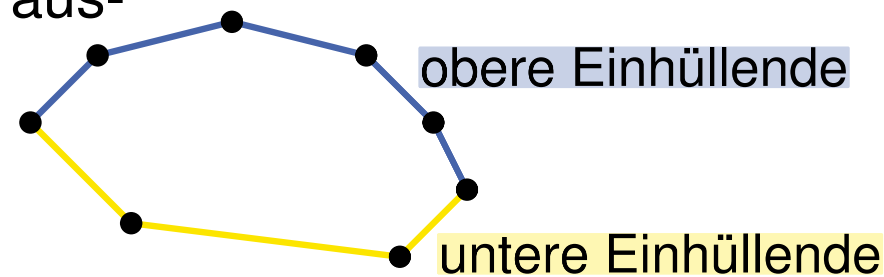


Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

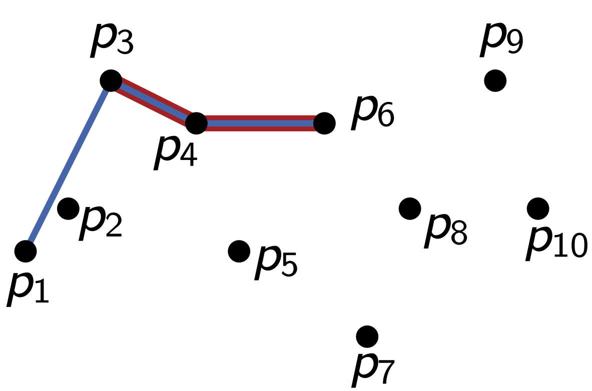




Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende

Beispiel

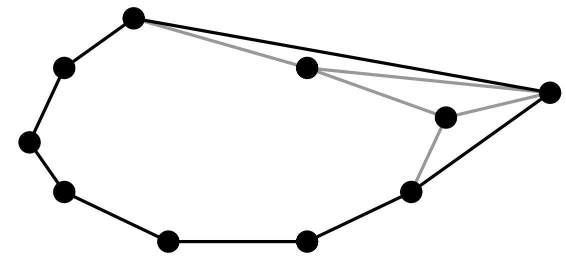


L: p_1 p_3 p_4 p_6

Graham Scan

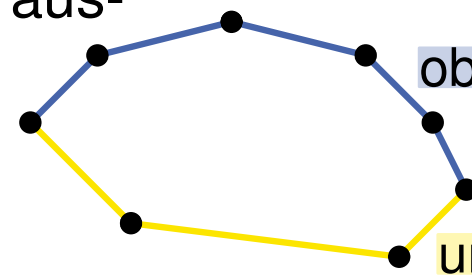
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

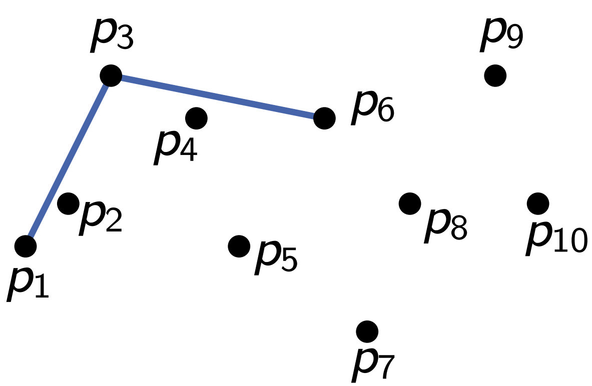


Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende



Beispiel

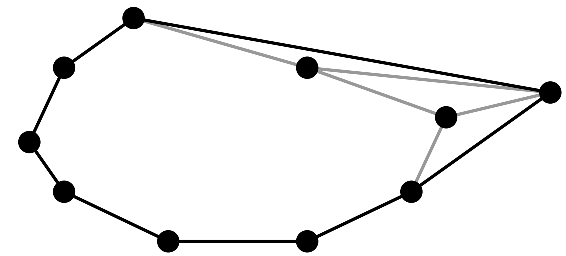


L: p_1 p_3 p_6

Graham Scan

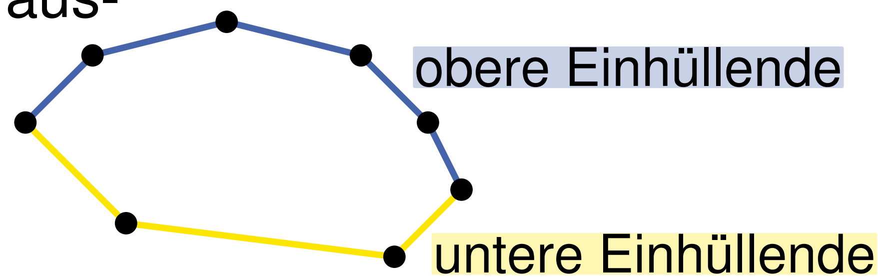
- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

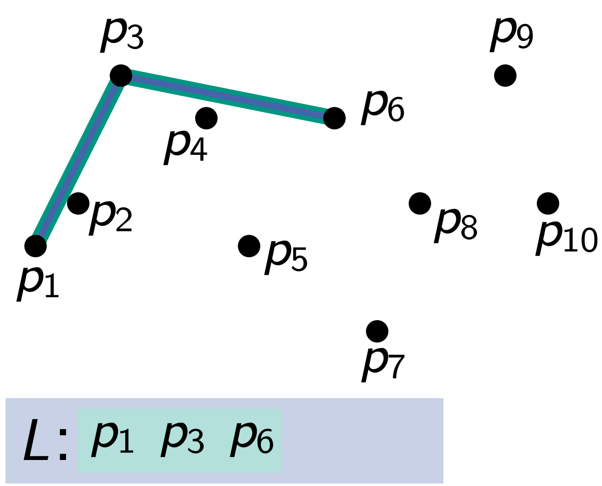


Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende



Beispiel



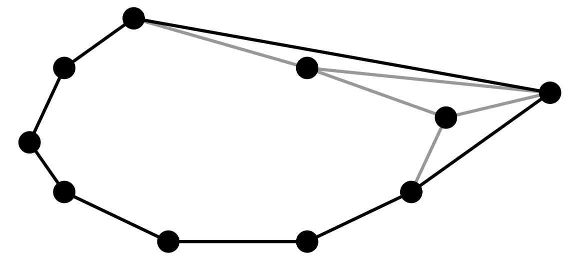
Graham Scan

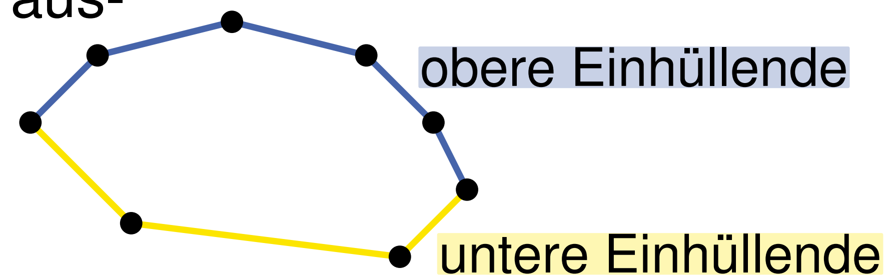
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

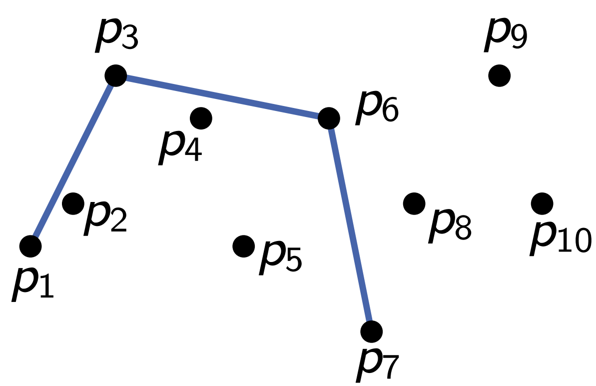
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



L: p_1 p_3 p_6 p_7

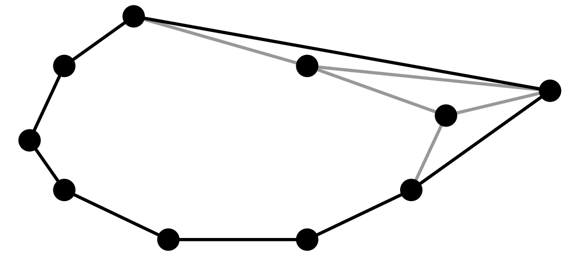
Graham Scan

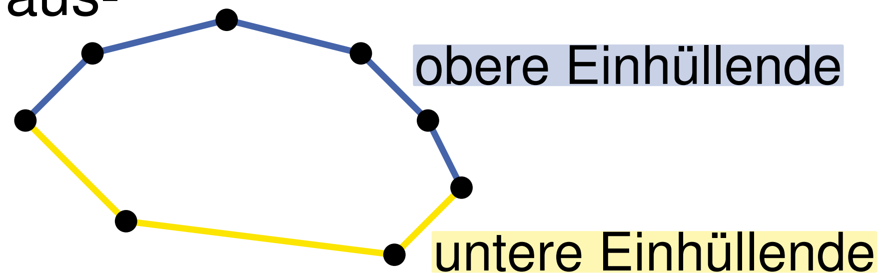
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

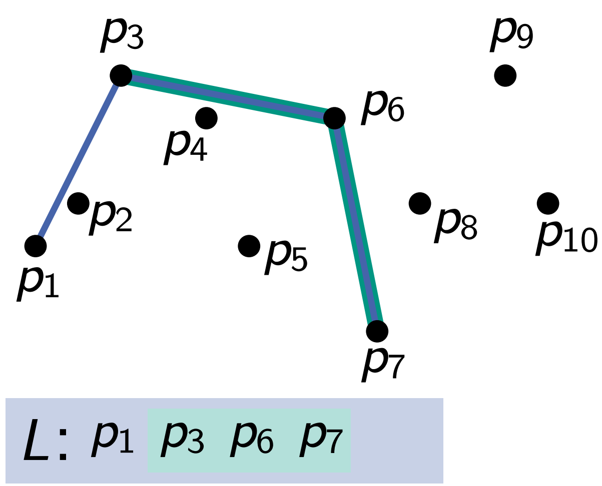
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



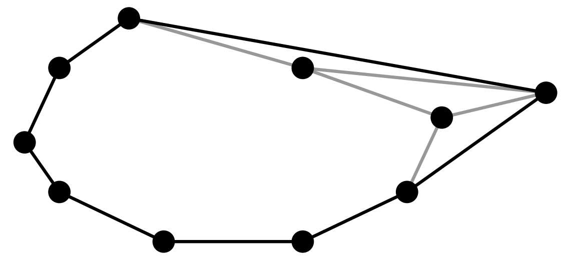
Graham Scan

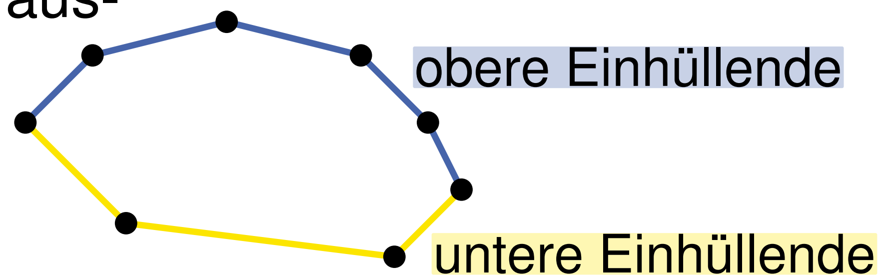
- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

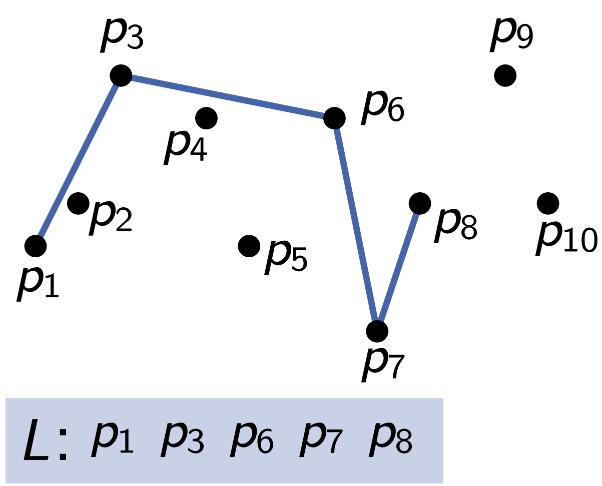
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



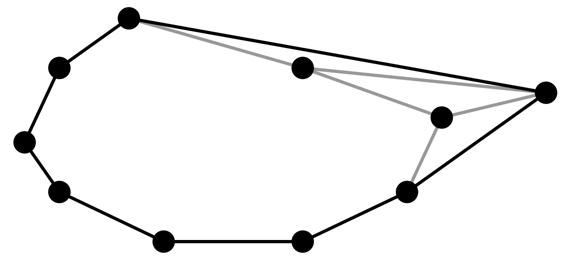
Graham Scan

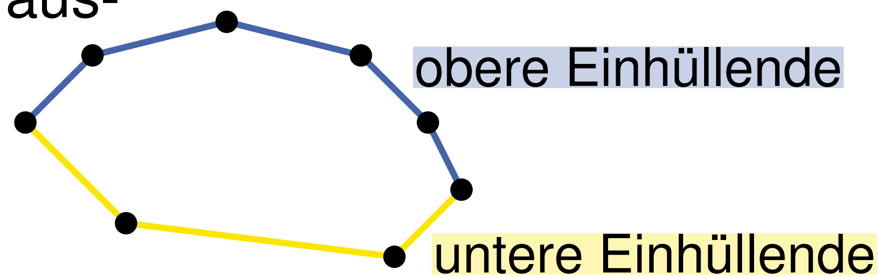
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

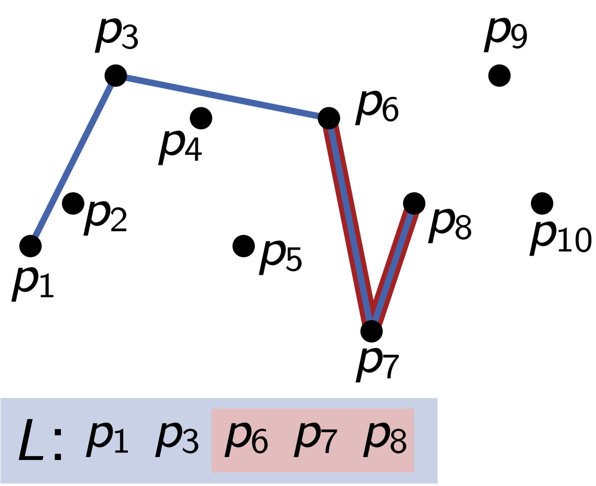
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



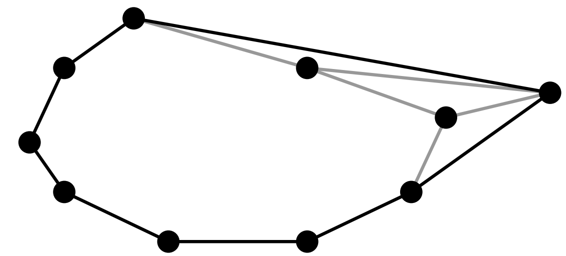
Graham Scan

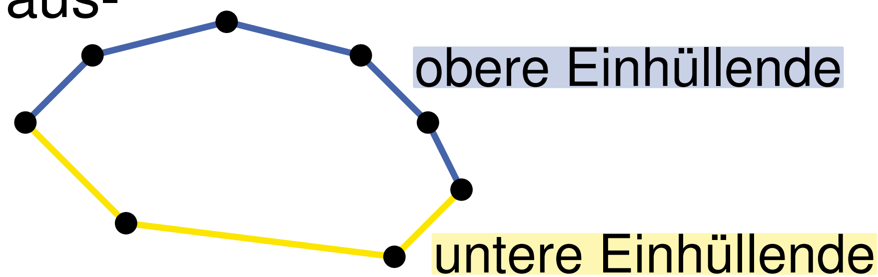
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

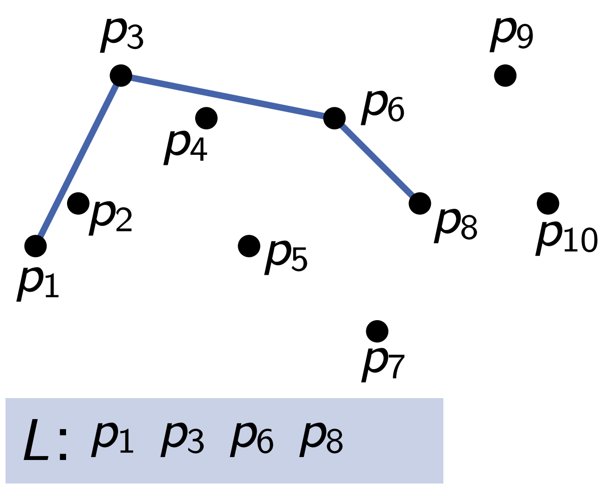
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



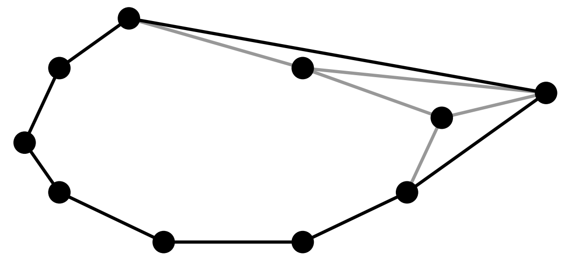
Graham Scan

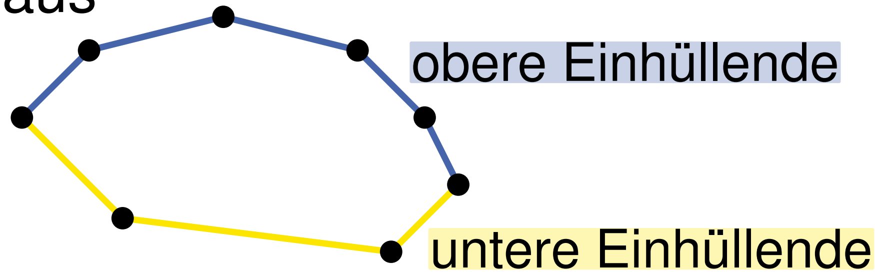
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

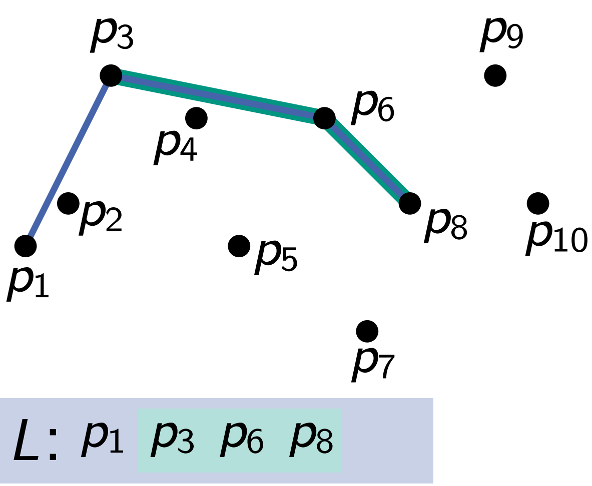
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



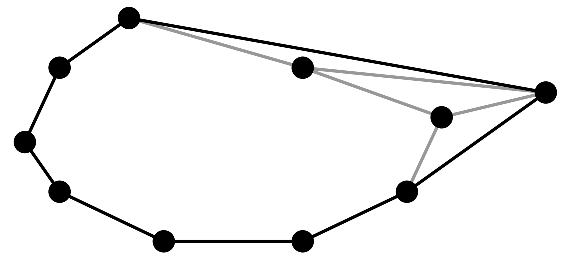
Graham Scan

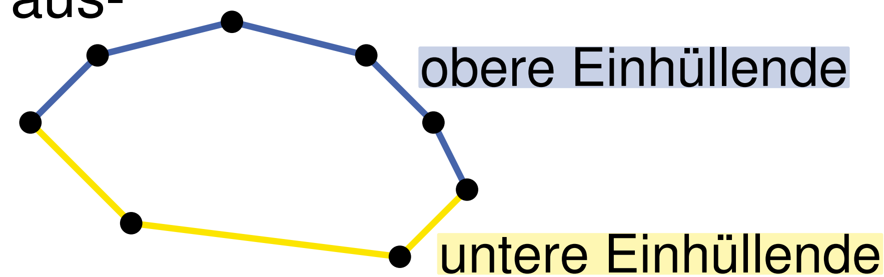
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

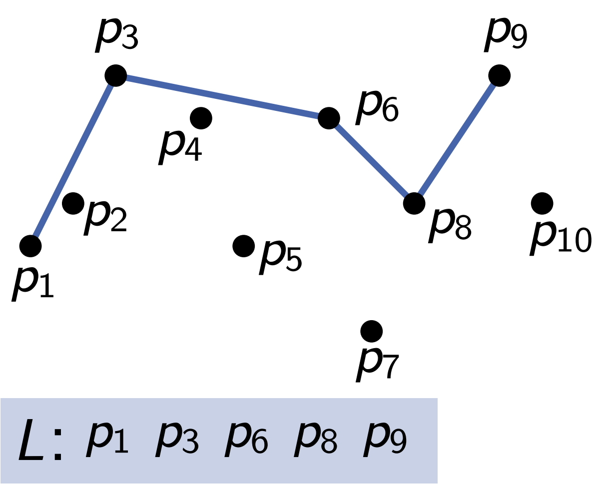
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





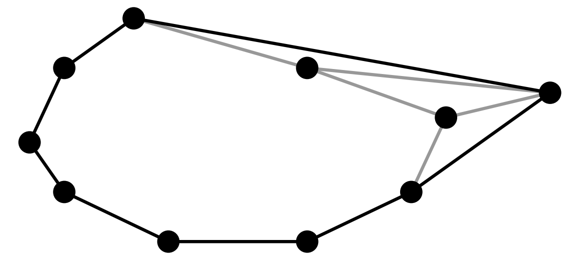
Beispiel



Graham Scan

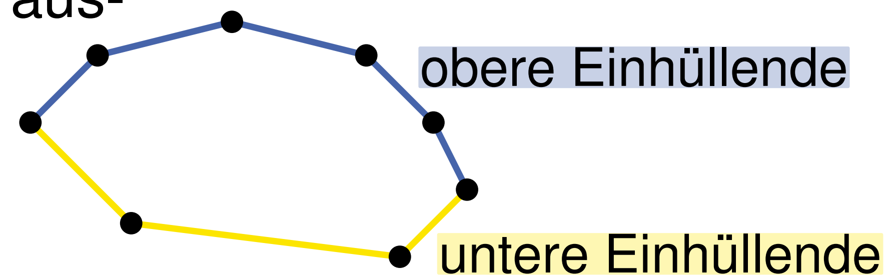
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

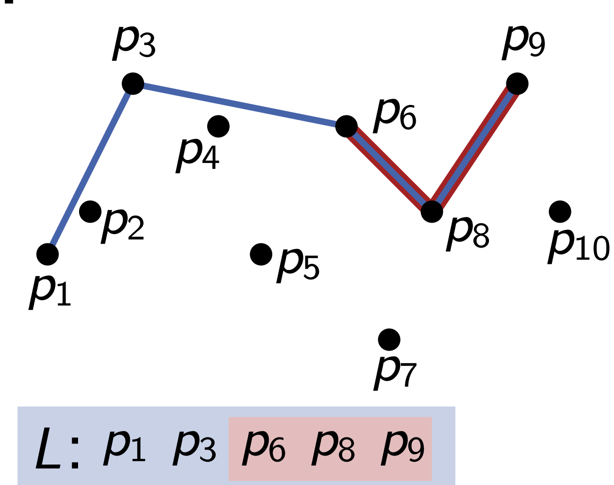


Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende



Beispiel



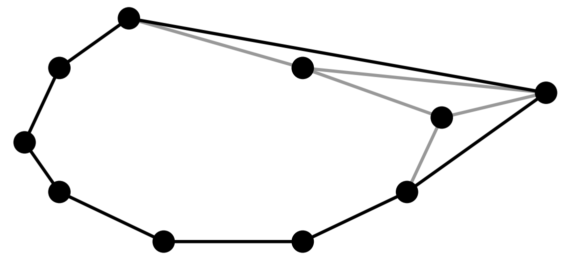
Graham Scan

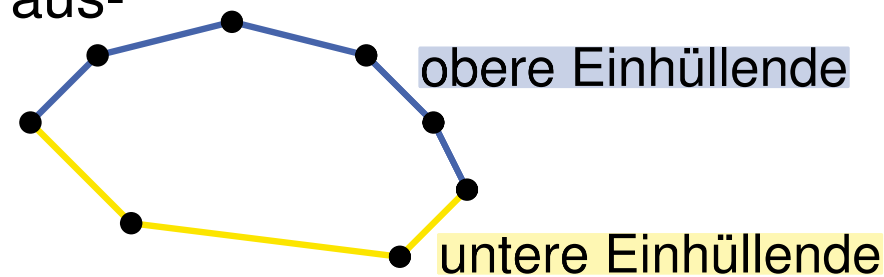
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

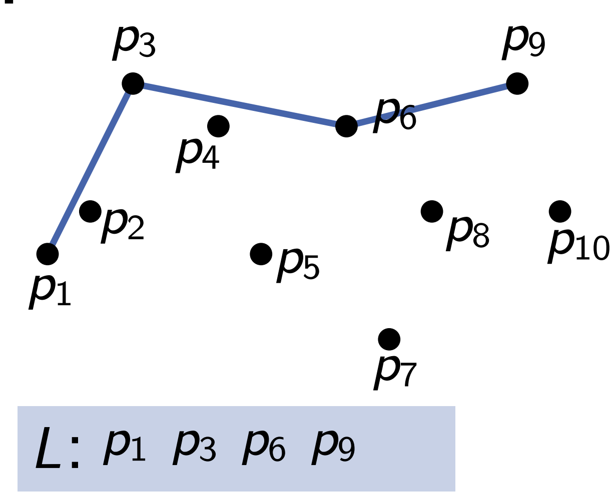
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





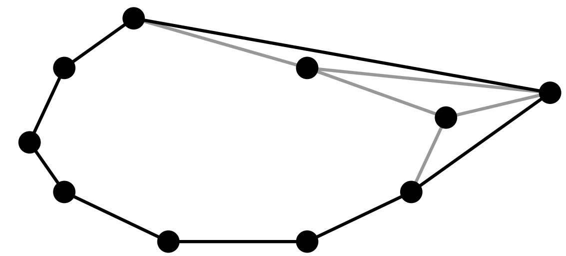
Beispiel



Graham Scan

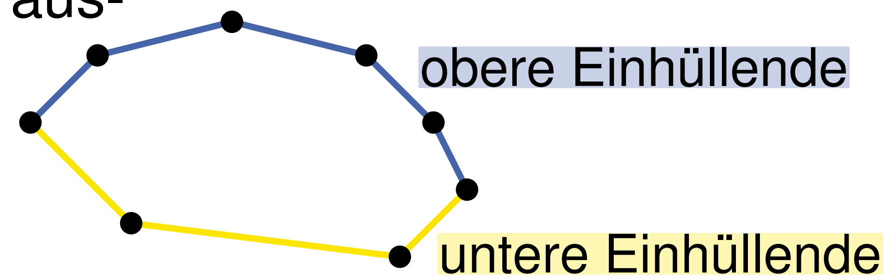
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

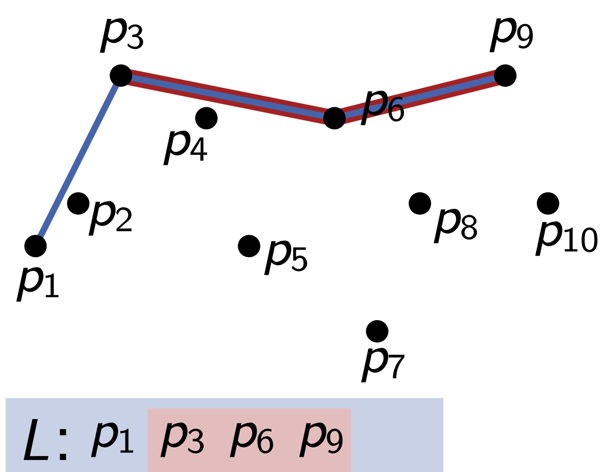


Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende



Beispiel



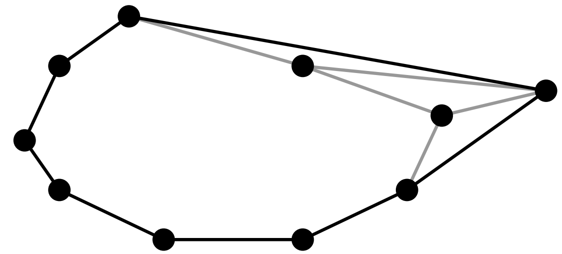
Graham Scan

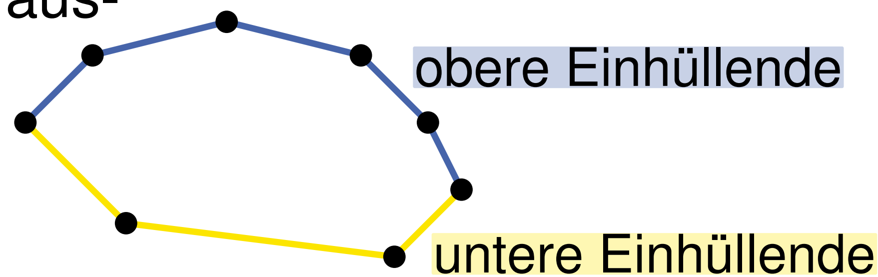
- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

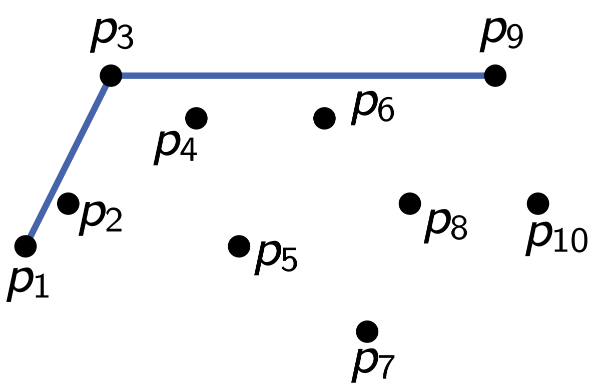
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel

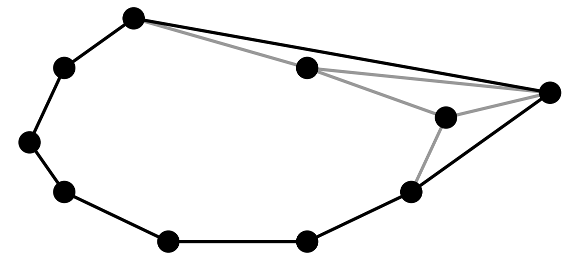


L: p_1 p_3 p_9

Graham Scan

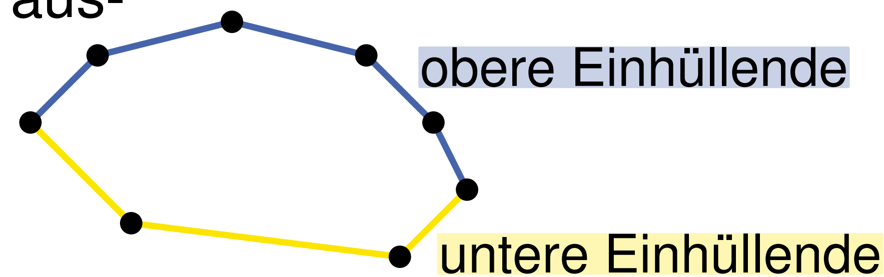
- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

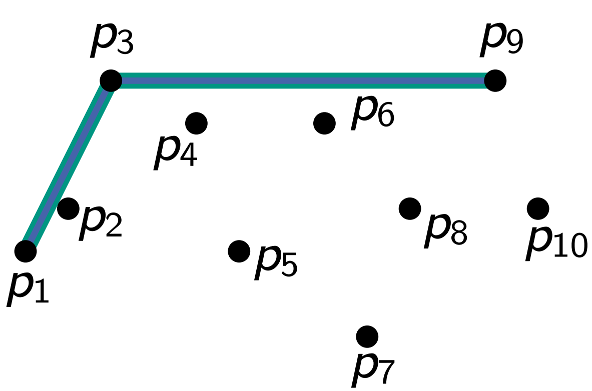


Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende



Beispiel

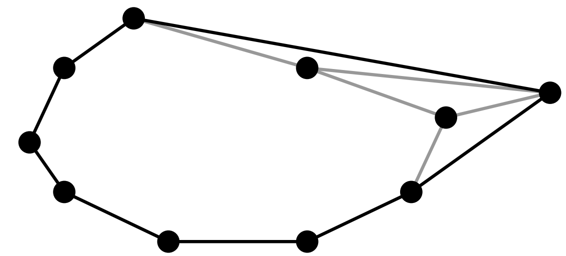


L: p_1 p_3 p_9

Graham Scan

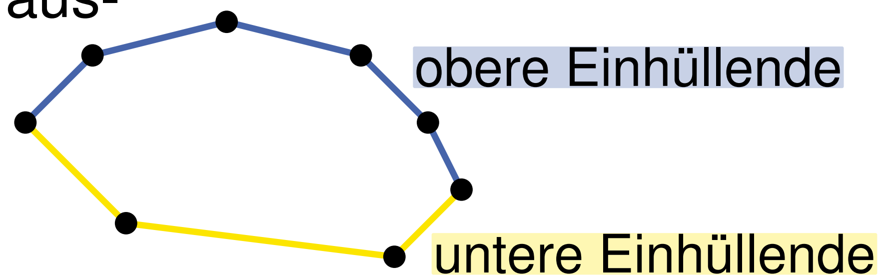
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

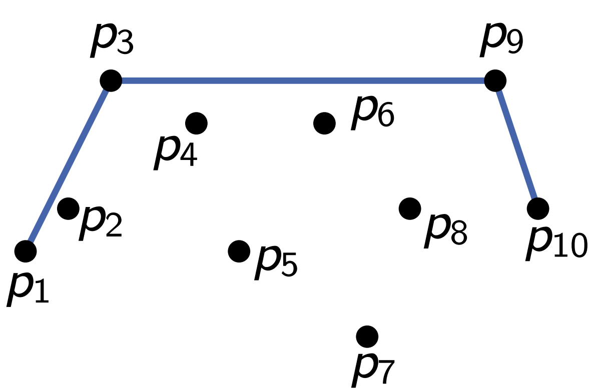


Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende



Beispiel



L: p_1 p_3 p_9 p_{10}

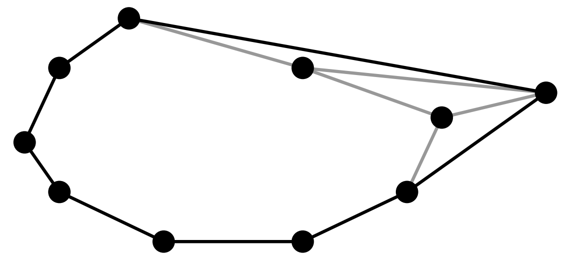
Graham Scan

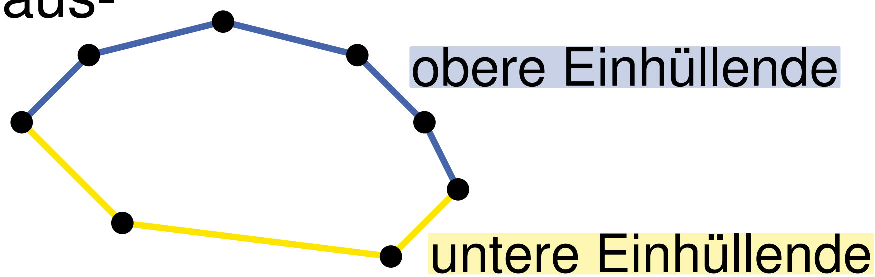
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

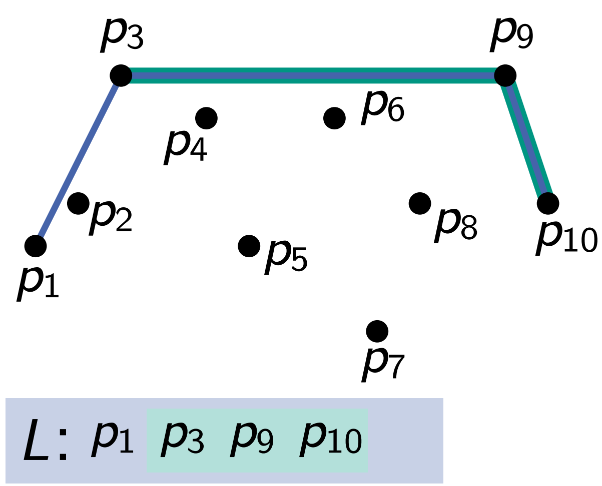
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





Beispiel



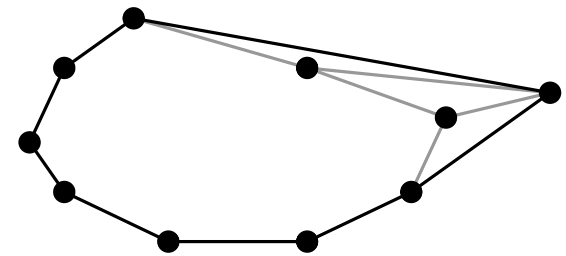
Graham Scan

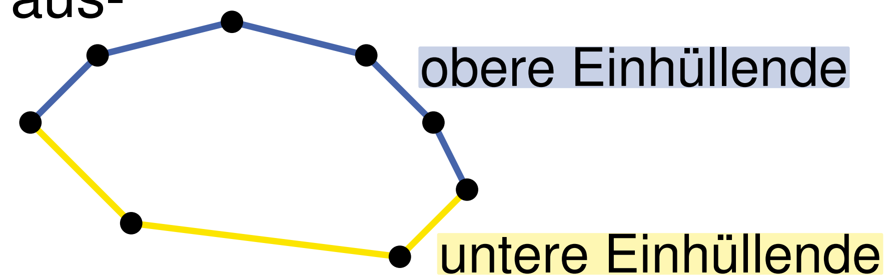
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt

Graham Scan

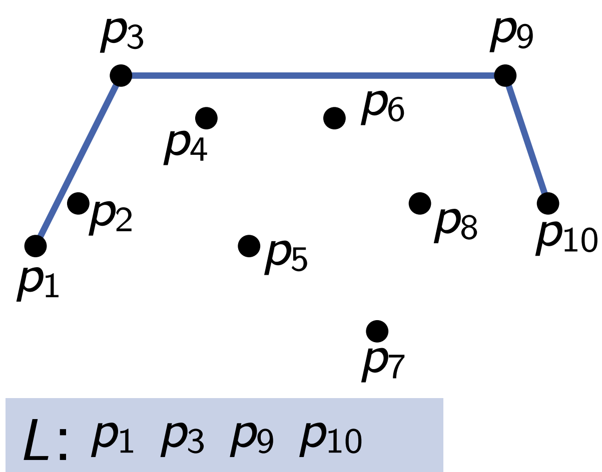
Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende





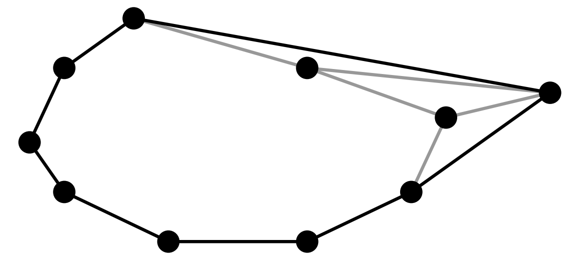
Beispiel



Graham Scan

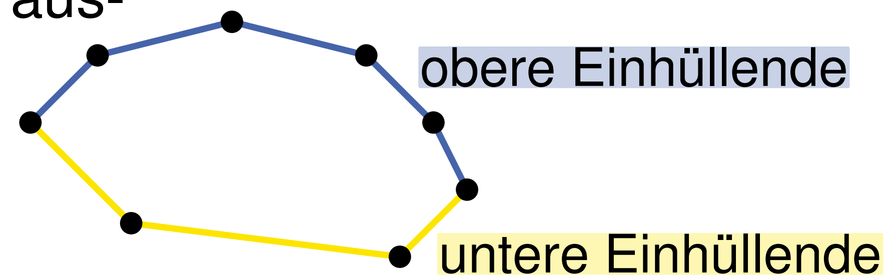
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

Graham Scan

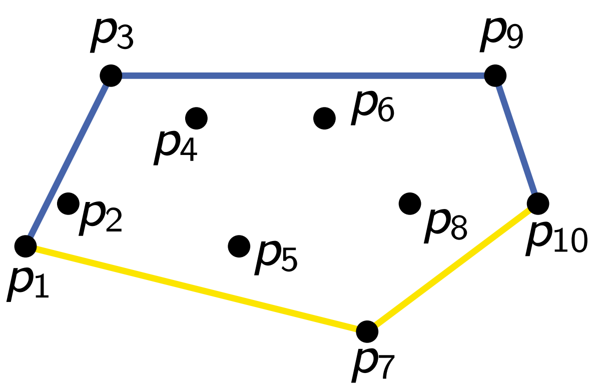


Idee: iteratives Vorgehen

- füge nach und nach Punkte ein
- aktualisiere konvexe Hülle in jedem Schritt
- Beobachtung: die konvexe Hülle macht ausschließlich Rechtsknicke
- Reihenfolge: von links nach rechts
- zunächst: nur die obere Einhüllende



Beispiel



L: p_1 p_3 p_9 p_{10}

Analog: untere Einhüllende

Graham Scan

- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

Graham Scan – Analyse

Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

Laufzeit:

Graham Scan – Analyse

Graham Scan	**Laufzeit:**
■ sortiere P (bzgl. x): $p_1, \ldots, p_n$	$O(n \log n)$
■ füge p_1 und p_2 in Liste L ein	$O(1)$
■ für jeden weiteren Punkt p_i:	$O(??)$
■ füge p_i hinten in L ein	$O(1)$
■ solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt	$O(??)$
■ L ist die obere Einhüllende	$O(n)$

Graham Scan – Analyse

Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

Laufzeit:

$O(n \log n)$

$O(1)$

$O(??)$

$O(1)$

$O(??)$

kann jedem Punkt nur einmal passieren

$O(n)$

Graham Scan – Analyse

Graham Scan | **Laufzeit:** $O(n \log n)$

- sortiere P (bzgl. x): $p_1, \ldots, p_n$ — $O(n \log n)$
- füge p_1 und p_2 in Liste L ein — $O(1)$
- für jeden weiteren Punkt p_i: — $O(n)$
  - füge p_i hinten in L ein — $O(1)$
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt — $O(1)$ (amortisiert)

    kann jedem Punkt nur einmal passieren
- L ist die obere Einhüllende — $O(n)$

Graham Scan – Analyse

Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

Laufzeit: $O(n \log n)$

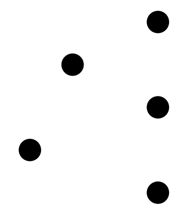
Graham Scan – Analyse

Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

Laufzeit: $O(n \log n)$

Sonderfall: gleiche x-Koordinaten



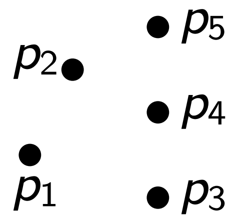
Graham Scan – Analyse

Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

Laufzeit: $O(n \log n)$

Sonderfall: gleiche x-Koordinaten



- sortiere lexikographisch (erst x, dann y)

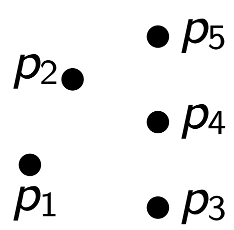
Graham Scan – Analyse

Graham Scan

- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

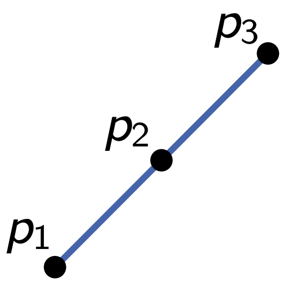
Laufzeit: $O(n \log n)$

Sonderfall: gleiche x-Koordinaten



- sortiere lexikographisch (erst x, dann y)

Sonderfall: kollineare Punkte



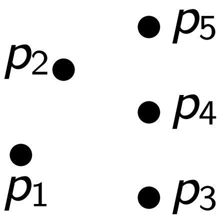
Graham Scan – Analyse

Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

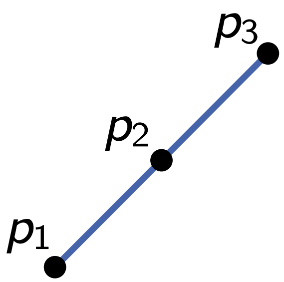
Laufzeit: $O(n \log n)$

Sonderfall: gleiche x-Koordinaten



- sortiere lexikographisch (erst x, dann y)

Sonderfall: kollineare Punkte



- $p2$ sollte nicht Teil der Ausgabe sein
- prüfe auf Rechtsknick, statt auf Linksknick

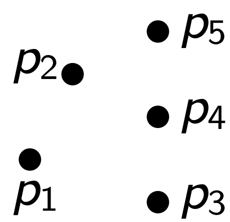
Graham Scan – Analyse

Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

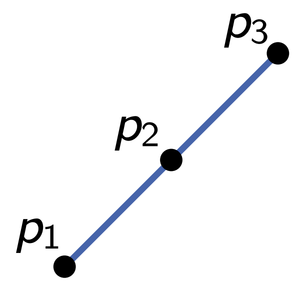
Laufzeit: $O(n \log n)$

Sonderfall: gleiche x-Koordinaten



- sortiere lexikographisch (erst x, dann y)

Sonderfall: kollineare Punkte



- $p2$ sollte nicht Teil der Ausgabe sein
- prüfe auf Rechtsknick, statt auf Linksknick

Robustheit

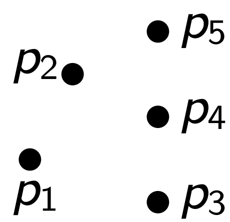
- was passiert, wenn eine Prüfung auf Links- vs. Rechtsknick schief geht?

Graham Scan – Analyse

Graham Scan
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

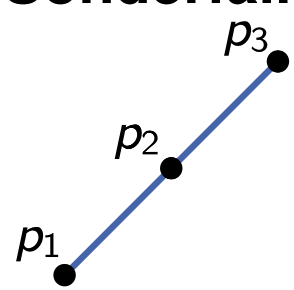
Laufzeit: $O(n \log n)$

Sonderfall: gleiche x-Koordinaten



- sortiere lexikographisch (erst x, dann y)

Sonderfall: kollineare Punkte



- $p2$ sollte nicht Teil der Ausgabe sein
- prüfe auf Rechtsknick, statt auf Linksknick

Robustheit
- was passiert, wenn eine Prüfung auf Links- vs. Rechtsknick schief geht?
- das resultierende Polygon hat ggf. einen leichten Linksknick
- ein Punkt liegt ggf. leicht außerhalb des Polygons

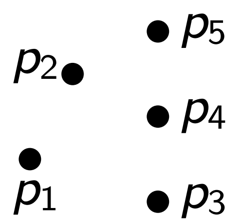
Graham Scan – Analyse

Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

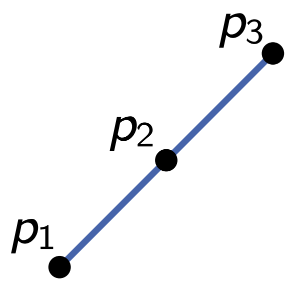
Laufzeit: $O(n \log n)$

Sonderfall: gleiche *x*-Koordinaten



- sortiere lexikographisch (erst x, dann y)

Sonderfall: kollineare Punkte



- $p2$ sollte nicht Teil der Ausgabe sein
- prüfe auf Rechtsknick, statt auf Linksknick

Robustheit

- was passiert, wenn eine Prüfung auf Links- vs. Rechtsknick schief geht?
- das resultierende Polygon hat ggf. einen leichten Linksknick
- ein Punkt liegt ggf. leicht außerhalb des Polygons
- aber: das Resultat ist in jedem Fall ein Polygon, das ähnlich zu $\mathcal{CH}(P)$ ist

Graham Scan – Korrektheit

Graham Scan

- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

Graham Scan – Korrektheit

Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

Lemma
Der Algorithmus berechnet die obere Einhüllende L von P.

Graham Scan – Korrektheit

Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

Lemma
Der Algorithmus berechnet die obere Einhüllende L von P.

Beweis

- zeige: L verbindet p_1 mit p_n, sodass
  - L macht nur Rechtsknicke
  - jeder Punkt aus $P \setminus L$ liegt unter L

Graham Scan – Korrektheit

Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

> **Lemma**
> Der Algorithmus berechnet die obere Einhüllende L von P.

Beweis

- zeige: L verbindet p_1 mit p_n, sodass
  - L macht nur Rechtsknicke
  - jeder Punkt aus $P \setminus L$ liegt unter L
- zeige diese Aussagen per Induktion über i für die Punkte $P_i = \{p_1, \ldots, p_i\}$

Graham Scan – Korrektheit

Graham Scan

- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

Lemma
Der Algorithmus berechnet die obere Einhüllende L von P.

Beweis

- zeige: L verbindet p_1 mit p_n, sodass
  - L macht nur Rechtsknicke
  - jeder Punkt aus $P \setminus L$ liegt unter L
- zeige diese Aussagen per Induktion über i für die Punkte $P_i = \{p_1, \dots, p_i\}$

Nach Schritt i startet L bei p_1 und endet bei p_i

Graham Scan – Korrektheit

Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

Lemma
Der Algorithmus berechnet die obere Einhüllende L von P.

Beweis

- zeige: L verbindet p_1 mit p_n, sodass
  - L macht nur Rechtsknicke
  - jeder Punkt aus $P \setminus L$ liegt unter L
- zeige diese Aussagen per Induktion über i für die Punkte $P_i = \{p_1, \ldots, p_i\}$

Nach Schritt *i* startet *L* bei p_1 und endet bei p_i

- offensichtlich, da erster und letzter Punkt nie gelöscht werden

Graham Scan – Korrektheit

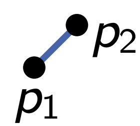
Graham Scan
- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

Lemma
Der Algorithmus berechnet die obere Einhüllende L von P.

Beweis
- zeige: L verbindet p_1 mit p_n, sodass
  - L macht nur Rechtsknicke
  - jeder Punkt aus $P \setminus L$ liegt unter L
- zeige diese Aussagen per Induktion über i für die Punkte $P_i = \{p_1, \dots, p_i\}$

Nach Schritt *i* macht *L* nur Rechtsknicke

- stimmt nach der Initialisierung ($i = 2$)



Graham Scan – Korrektheit

Graham Scan

- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

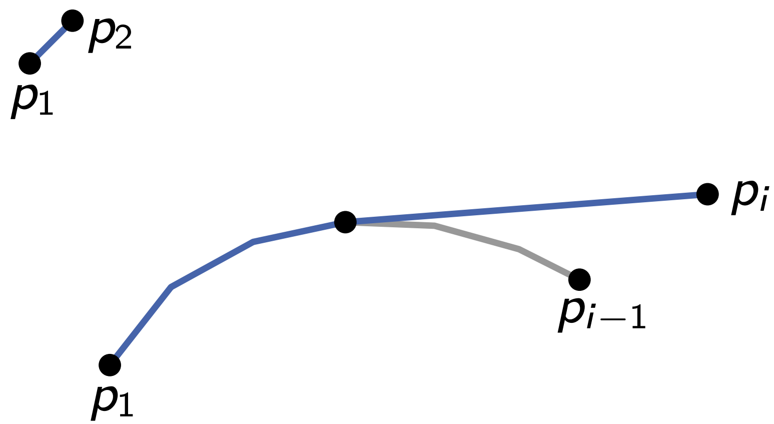
Lemma
Der Algorithmus berechnet die obere Einhüllende L von P.

Beweis

- zeige: L verbindet p_1 mit p_n, sodass
  - L macht nur Rechtsknicke
  - jeder Punkt aus $P \setminus L$ liegt unter L
- zeige diese Aussagen per Induktion über i für die Punkte $P_i = \{p_1, \dots, p_i\}$

Nach Schritt *i* macht *L* nur Rechtsknicke

- stimmt nach der Initialisierung ($i = 2$)
- nach Schritt i besteht L aus zwei Teilen
  - Präfix des Polygons L aus dem vorherigen Schritt $i - 1$
  - Kante zu p_i



Graham Scan – Korrektheit

Graham Scan

- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

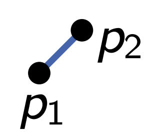
Lemma
Der Algorithmus berechnet die obere Einhüllende L von P.

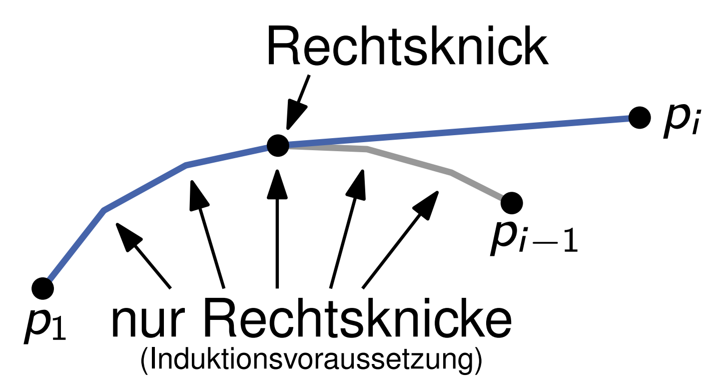
Beweis

- zeige: L verbindet p_1 mit p_n, sodass
  - L macht nur Rechtsknicke
  - jeder Punkt aus $P \setminus L$ liegt unter L
- zeige diese Aussagen per Induktion über i für die Punkte $P_i = \{p_1, \dots, p_i\}$

Nach Schritt *i* macht *L* nur Rechtsknicke

- stimmt nach der Initialisierung ($i = 2$)
- nach Schritt i besteht L aus zwei Teilen
  - Präfix des Polygons L aus dem vorherigen Schritt $i - 1$
  - Kante zu p_i





Graham Scan – Korrektheit

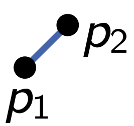
Graham Scan
- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

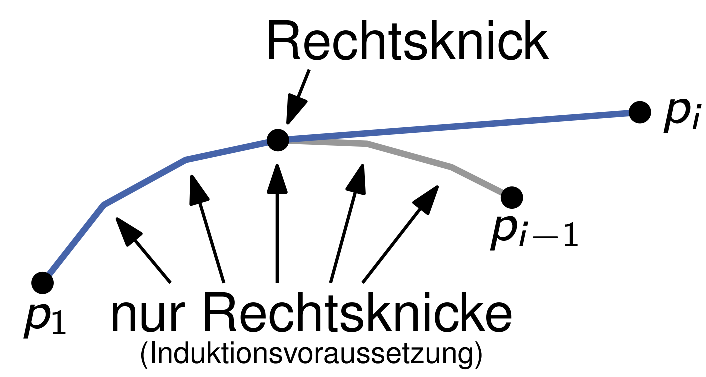
Lemma
Der Algorithmus berechnet die obere Einhüllende L von P.

Beweis
- zeige: L verbindet p_1 mit p_n, sodass
  - L macht nur Rechtsknicke
  - jeder Punkt aus $P \setminus L$ liegt unter L
- zeige diese Aussagen per Induktion über i für die Punkte $P_i = \{p_1, \ldots, p_i\}$

Nach Schritt *i* macht *L* nur Rechtsknicke
- stimmt nach der Initialisierung ($i = 2$)
- nach Schritt i besteht L aus zwei Teilen
  - Präfix des Polygons L aus dem vorherigen Schritt $i - 1$
  - Kante zu p_i

$\Rightarrow$ nur Rechtsknicke





Graham Scan – Korrektheit

Graham Scan

- sortiere P (bzgl. x): $p_1, \ldots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

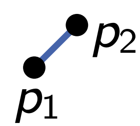
Lemma
Der Algorithmus berechnet die obere Einhüllende L von P.

Beweis

- zeige: L verbindet p_1 mit p_n, sodass
  - L macht nur Rechtsknicke
  - jeder Punkt aus $P \setminus L$ liegt unter L
- zeige diese Aussagen per Induktion über i für die Punkte $P_i = \{p_1, \ldots, p_i\}$

Nach Schritt i liegt jeder Punkt aus $P_i \setminus L$ unter L

- stimmt nach der Initialisierung ($i = 2$)



Graham Scan – Korrektheit

Graham Scan

- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

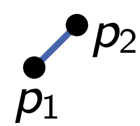
Lemma
Der Algorithmus berechnet die obere Einhüllende L von P.

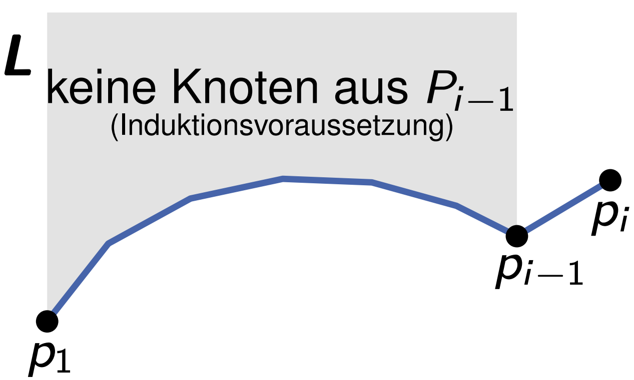
Beweis

- zeige: L verbindet p_1 mit p_n, sodass
  - L macht nur Rechtsknicke
  - jeder Punkt aus $P \setminus L$ liegt unter L
- zeige diese Aussagen per Induktion über i für die Punkte $P_i = \{p_1, \dots, p_i\}$

Nach Schritt *i* liegt jeder Punkt aus $P_i \setminus L$ unter L

- stimmt nach der Initialisierung ($i = 2$)
- stimmt auch nach dem Einfügen von p_i





Graham Scan – Korrektheit

Graham Scan

- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

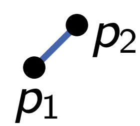
Lemma
Der Algorithmus berechnet die obere Einhüllende L von P.

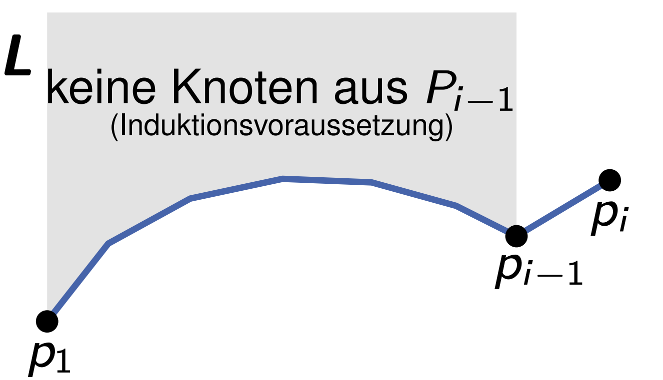
Beweis

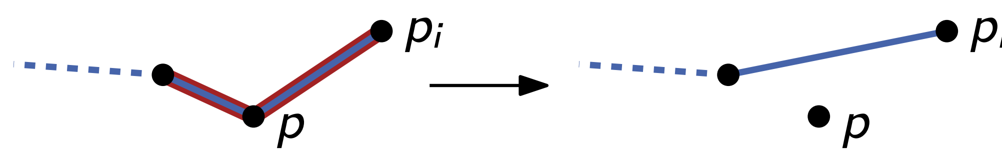
- zeige: L verbindet p_1 mit p_n, sodass
  - L macht nur Rechtsknicke
  - jeder Punkt aus $P \setminus L$ liegt unter L
- zeige diese Aussagen per Induktion über i für die Punkte $P_i = \{p_1, \dots, p_i\}$

Nach Schritt *i* liegt jeder Punkt aus $P_i \setminus L$ unter L

- stimmt nach der Initialisierung ($i = 2$)
- stimmt auch nach dem Einfügen von p_i
- Löschen eines Punkts p aus L schiebt L nur weiter nach oben
- außerdem liegt p danach unter L







Graham Scan – Korrektheit

Graham Scan

- sortiere P (bzgl. x): $p_1, \dots, p_n$
- füge p_1 und p_2 in Liste L ein
- für jeden weiteren Punkt p_i:
  - füge p_i hinten in L ein
  - solange Linksknick bei letzten drei Punkten: entferne den vorletzten Punkt
- L ist die obere Einhüllende

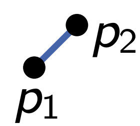
Lemma
Der Algorithmus berechnet die obere Einhüllende L von P.

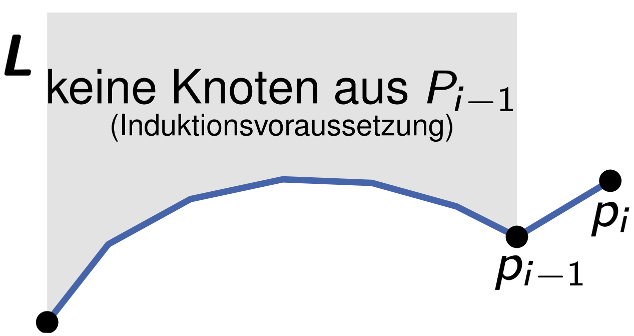
Beweis

- zeige: L verbindet p_1 mit p_n, sodass
  - L macht nur Rechtsknicke
  - jeder Punkt aus $P \setminus L$ liegt unter L
- zeige diese Aussagen per Induktion über i für die Punkte $P_i = \{p_1, \dots, p_i\}$

Nach Schritt *i* liegt jeder Punkt aus $P_i \setminus L$ unter L

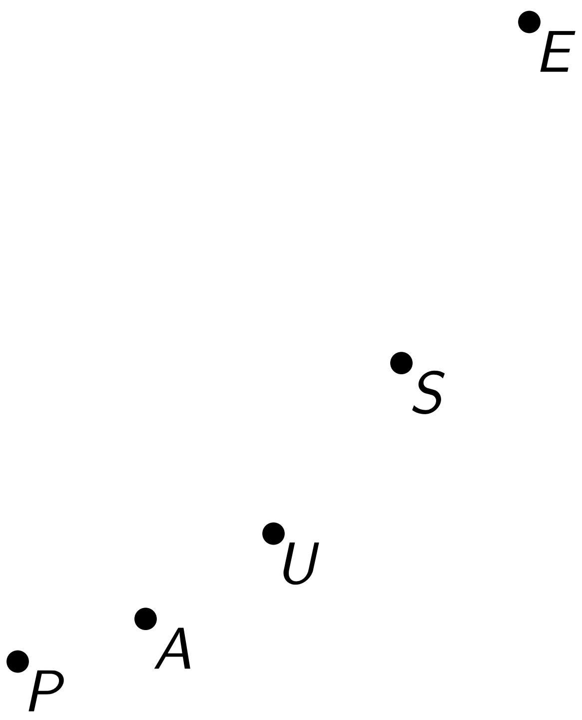
- stimmt nach der Initialisierung ($i = 2$)
- stimmt auch nach dem Einfügen von p_i
- Löschen eines Punkts p aus L schiebt L



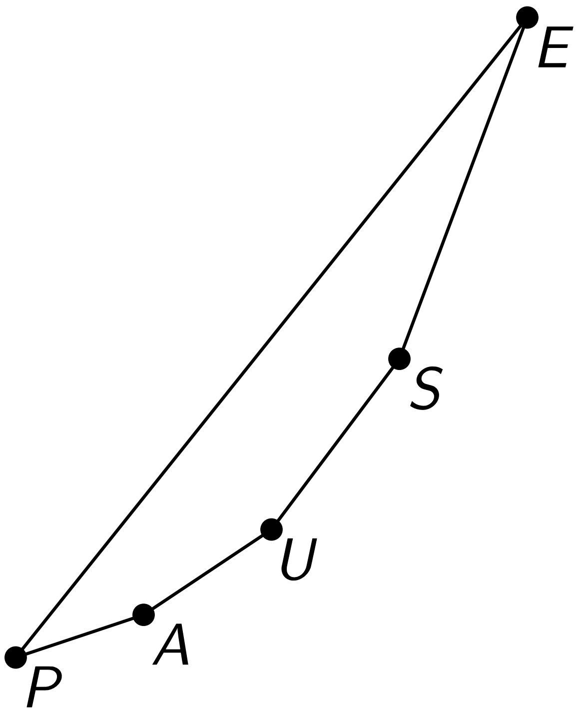


Theorem
Der Graham Scan berechnet die konvexe Hülle von n Punkten in $O(n \log n)$.

Berechne die konvexe Hülle



Berechne die konvexe Hülle



Geht es schneller?

Geht es schneller?

Theorem
Wenn die konvexe Hülle von n Punkten in $f(n)$ Zeit berechnet werden kann, dann kann man auch n Zahlen in $O(f(n) + n)$ Zeit sortieren.

Beweis

Geht es schneller?

Theorem
Wenn die konvexe Hülle von n Punkten in $f(n)$ Zeit berechnet werden kann, dann kann man auch n Zahlen in $O(f(n) + n)$ Zeit sortieren.

Beweis

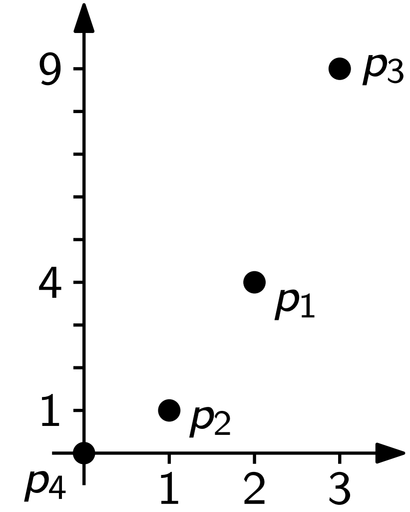
- gegeben: n Zahlen $a_1, \ldots, a_n$
- konstruiere n Punkte $P = \{p_1, \ldots, p_n\}$

Geht es schneller?

Theorem
Wenn die konvexe Hülle von n Punkten in $f(n)$ Zeit berechnet werden kann, dann kann man auch n Zahlen in $O(f(n) + n)$ Zeit sortieren.

Beweis
- gegeben: n Zahlen $a_1, \dots, a_n$
- konstruiere n Punkte $P = \{p_1, \dots, p_n\}$ mit $p_i = (a_i, a_i^2)$

Beispiel
$a_1 = 2,\ a_2 = 1,\ a_3 = 3,\ a_4 = 0$

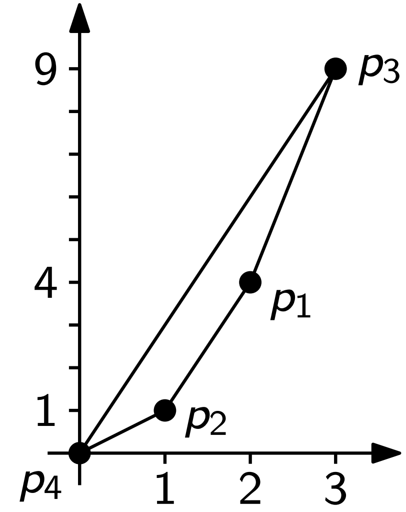


Geht es schneller?

Theorem
Wenn die konvexe Hülle von n Punkten in $f(n)$ Zeit berechnet werden kann, dann kann man auch n Zahlen in $O(f(n) + n)$ Zeit sortieren.

Beweis
- gegeben: n Zahlen $a_1, \ldots, a_n$
- konstruiere n Punkte $P = \{p_1, \ldots, p_n\}$ mit $p_i = (a_i, a_i^2)$
- $\mathcal{CH}(P)$ enthält die Punkte sortiert nach a_i

Beispiel
$a_1 = 2,\ a_2 = 1,\ a_3 = 3,\ a_4 = 0$



Geht es schneller?

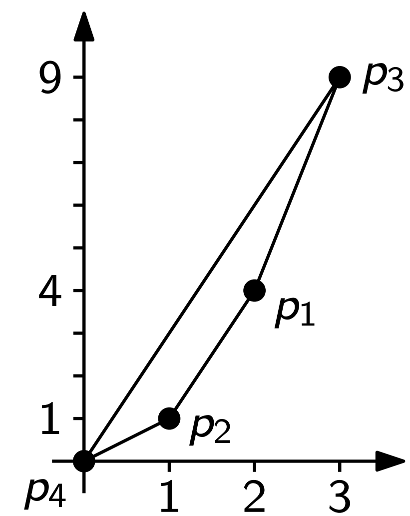
Theorem
Wenn die konvexe Hülle von n Punkten in $f(n)$ Zeit berechnet werden kann, dann kann man auch n Zahlen in $O(f(n) + n)$ Zeit sortieren.

Beweis

- gegeben: n Zahlen $a_1, \ldots, a_n$
- konstruiere n Punkte $P = \{p_1, \ldots, p_n\}$ mit $p_i = (a_i, a_i^2)$
- $\mathcal{CH}(P)$ enthält die Punkte sortiert nach a_i
- Sortierung lässt sich in $O(n)$ aus $\mathcal{CH}(P)$ ablesen

Beispiel

$a_1 = 2,\ a_2 = 1,\ a_3 = 3,\ a_4 = 0$

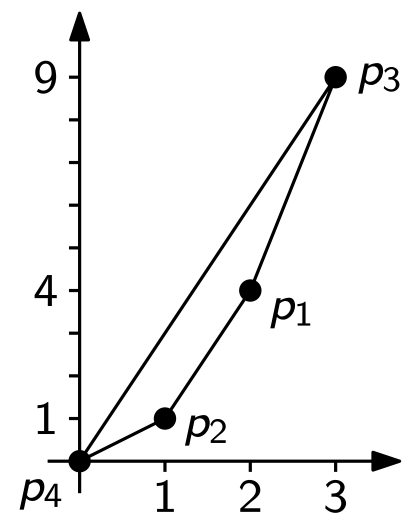


Geht es schneller?

Theorem
Wenn die konvexe Hülle von n Punkten in $f(n)$ Zeit berechnet werden kann, dann kann man auch n Zahlen in $O(f(n) + n)$ Zeit sortieren.

Beweis
- gegeben: n Zahlen $a_1, \ldots, a_n$
- konstruiere n Punkte $P = \{p_1, \ldots, p_n\}$ mit $p_i = (a_i, a_i^2)$
- $\mathcal{CH}(P)$ enthält die Punkte sortiert nach a_i
- Sortierung lässt sich in $O(n)$ aus $\mathcal{CH}(P)$ ablesen

Beispiel
$a_1 = 2,\ a_2 = 1,\ a_3 = 3,\ a_4 = 0$

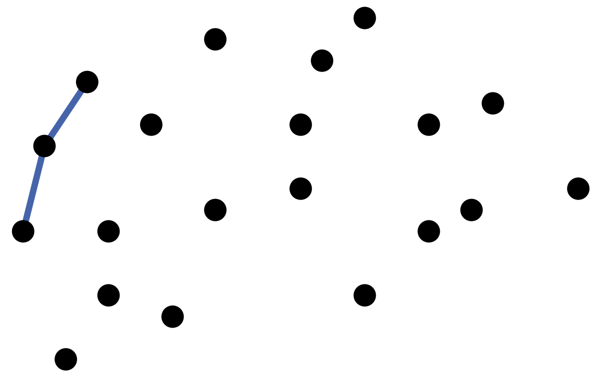


Untere Schranke
- vergleichsbasiertes Sortieren: $\Omega(n \log n)$
- Graham Scan ist also optimal

Gift Wrapping (Jarvis March)

Alternativer Ansatz

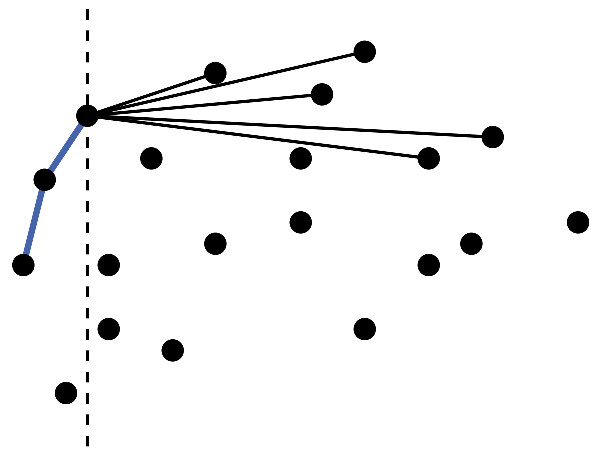
- Annahme: wir haben schon einen Teil der oberen Einhüllenden
- Ziel: finde den nächsten Punkt



Gift Wrapping (Jarvis March)

Alternativer Ansatz

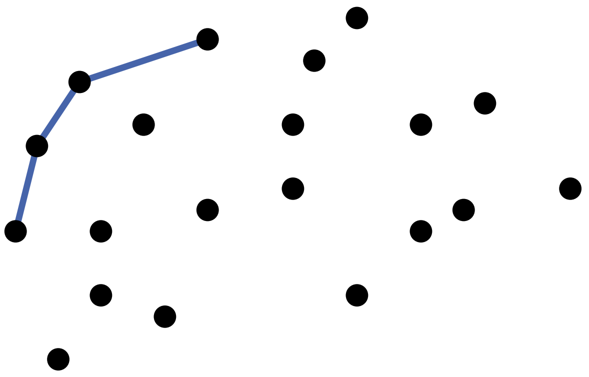
- Annahme: wir haben schon einen Teil der oberen Einhüllenden
- Ziel: finde den nächsten Punkt
- wähle den Punkt mit dem kleinsten Winkel



Gift Wrapping (Jarvis March)

Alternativer Ansatz

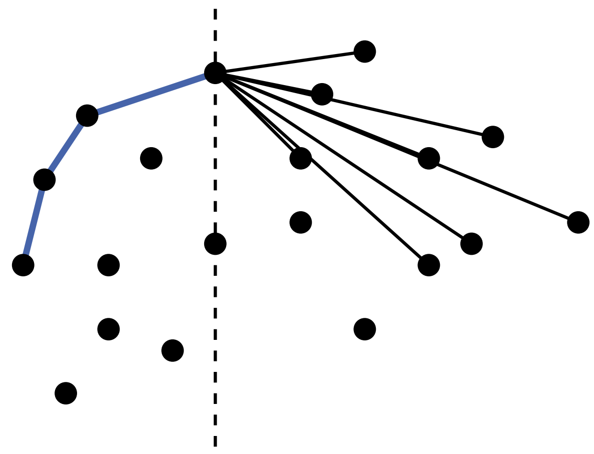
- Annahme: wir haben schon einen Teil der oberen Einhüllenden
- Ziel: finde den nächsten Punkt
- wähle den Punkt mit dem kleinsten Winkel



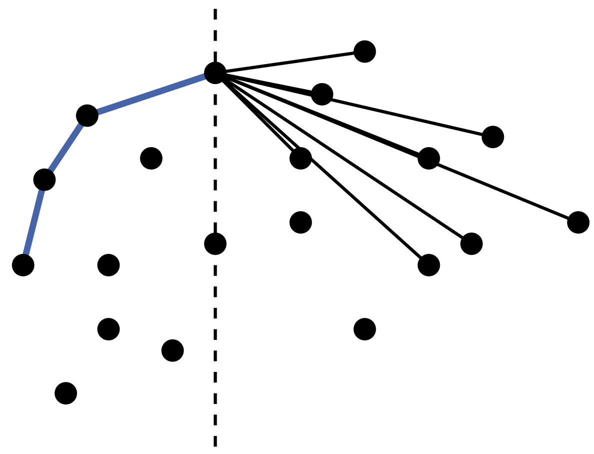
Gift Wrapping (Jarvis March)

Alternativer Ansatz

- Annahme: wir haben schon einen Teil der oberen Einhüllenden
- Ziel: finde den nächsten Punkt
- wähle den Punkt mit dem kleinsten Winkel



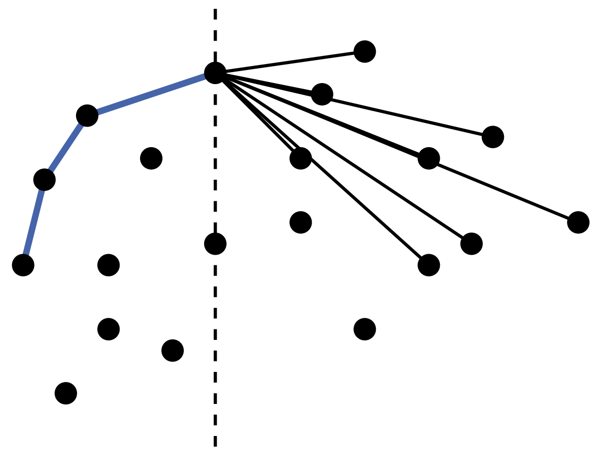
Gift Wrapping (Jarvis March)



Alternativer Ansatz
- Annahme: wir haben schon einen Teil der oberen Einhüllenden
- Ziel: finde den nächsten Punkt
- wähle den Punkt mit dem kleinsten Winkel

Laufzeit
- pro Schritt: Minimum suchen $\rightarrow O(n)$

Gift Wrapping (Jarvis March)



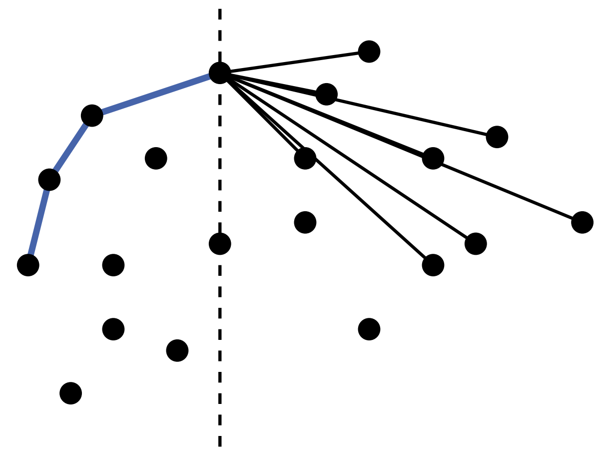
Alternativer Ansatz

- Annahme: wir haben schon einen Teil der oberen Einhüllenden
- Ziel: finde den nächsten Punkt
- wähle den Punkt mit dem kleinsten Winkel

Laufzeit

- pro Schritt: Minimum suchen $\rightarrow O(n)$
- h Schritte, für $h = |\mathcal{CH}(P)|$

Gift Wrapping (Jarvis March)



Alternativer Ansatz

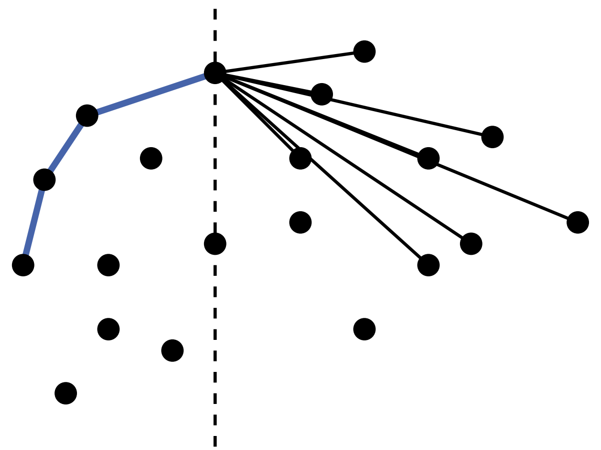
- Annahme: wir haben schon einen Teil der oberen Einhüllenden
- Ziel: finde den nächsten Punkt
- wähle den Punkt mit dem kleinsten Winkel

Laufzeit

- pro Schritt: Minimum suchen $\rightarrow O(n)$
- h Schritte, für $h = |\mathcal{CH}(P)|$

Theorem
Der Gift Wrapping Algorithmus berechnet Konvexe Hülle von n Punkten P in $O(hn)$ Zeit, wobei h die Anzahl Punkte von $\mathcal{CH}(P)$ ist.

Gift Wrapping (Jarvis March)



Alternativer Ansatz

- Annahme: wir haben schon einen Teil der oberen Einhüllenden
- Ziel: finde den nächsten Punkt
- wähle den Punkt mit dem kleinsten Winkel

Laufzeit

- pro Schritt: Minimum suchen $\rightarrow O(n)$
- h Schritte, für $h = |\mathcal{CH}(P)|$

Theorem
Der Gift Wrapping Algorithmus berechnet Konvexe Hülle von n Punkten P in $O(hn)$ Zeit, wobei h die Anzahl Punkte von $\mathcal{CH}(P)$ ist.

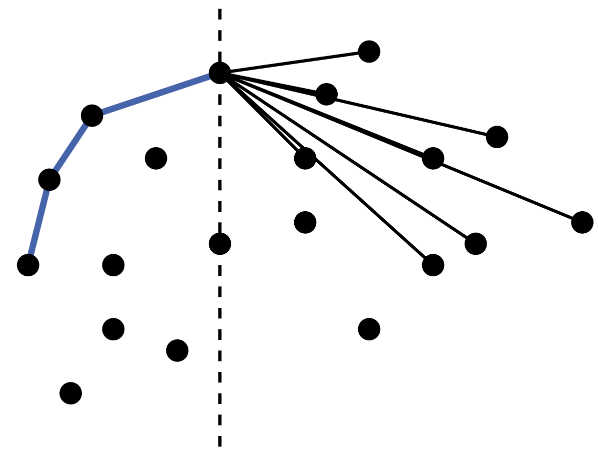
Anmerkungen

- ein solcher Algorithmus heißt **ausgabesensitiv**

Gift Wrapping (Jarvis March)

Alternativer Ansatz

- Annahme: wir haben schon einen Teil der oberen Einhüllenden
- Ziel: finde den nächsten Punkt
- wähle den Punkt mit dem kleinsten Winkel



Laufzeit

- pro Schritt: Minimum suchen $\to O(n)$
- h Schritte, für $h = |\mathcal{CH}(P)|$

> **Theorem**
> Der Gift Wrapping Algorithmus berechnet Konvexe Hülle von n Punkten P in $O(hn)$ Zeit, wobei h die Anzahl Punkte von $\mathcal{CH}(P)$ ist.

Anmerkungen

- ein solcher Algorithmus heißt **ausgabesensitiv**
- auf gewissen Instanzen (kleines h) kann er die untere Schranke schlagen

Zusammenfassung

Heute gesehen

- Algorithmus zur Berechnung der konvexen Hülle mit Laufzeit $O(n \log n)$

Zusammenfassung

Karlsruher Institut für Technologie

Heute gesehen

- Algorithmus zur Berechnung der konvexen Hülle mit Laufzeit $O(n \log n)$
- untere Schranke von $\Omega(n \log n)$

Zusammenfassung

Heute gesehen

- Algorithmus zur Berechnung der konvexen Hülle mit Laufzeit $O(n \log n)$
- untere Schranke von $\Omega(n \log n)$
- Ausgabesensitiver Algorithmus mit Laufzeit $O(hn)$

Zusammenfassung

Heute gesehen

- Algorithmus zur Berechnung der konvexen Hülle mit Laufzeit $O(n \log n)$
- untere Schranke von $\Omega(n \log n)$
- Ausgabesensitiver Algorithmus mit Laufzeit $O(hn)$
- die Robustheit ist ein wichtiger Aspekt in der algorithmischen Geometrie

Zusammenfassung

Heute gesehen

- Algorithmus zur Berechnung der konvexen Hülle mit Laufzeit $O(n \log n)$
- untere Schranke von $\Omega(n \log n)$
- Ausgabesensitiver Algorithmus mit Laufzeit $O(hn)$
- die Robustheit ist ein wichtiger Aspekt in der algorithmischen Geometrie

Was gibt es sonst noch?

- es geht sogar mit Laufzeit $O(n \log h)$

Zusammenfassung

Heute gesehen

- Algorithmus zur Berechnung der konvexen Hülle mit Laufzeit $O(n \log n)$
- untere Schranke von $\Omega(n \log n)$
- Ausgabesensitiver Algorithmus mit Laufzeit $O(hn)$
- die Robustheit ist ein wichtiger Aspekt in der algorithmischen Geometrie

Was gibt es sonst noch?

- es geht sogar mit Laufzeit $O(n \log h)$
- höhere Dimensionen

Zusammenfassung

Heute gesehen

- Algorithmus zur Berechnung der konvexen Hülle mit Laufzeit $O(n \log n)$
- untere Schranke von $\Omega(n \log n)$
- Ausgabesensitiver Algorithmus mit Laufzeit $O(hn)$
- die Robustheit ist ein wichtiger Aspekt in der algorithmischen Geometrie

Was gibt es sonst noch?

- es geht sogar mit Laufzeit $O(n \log h)$
- höhere Dimensionen
- konvexe Hülle eines einfachen Polygons geht in $O(n)$