Algorithmen und Wahrscheinlichkeit

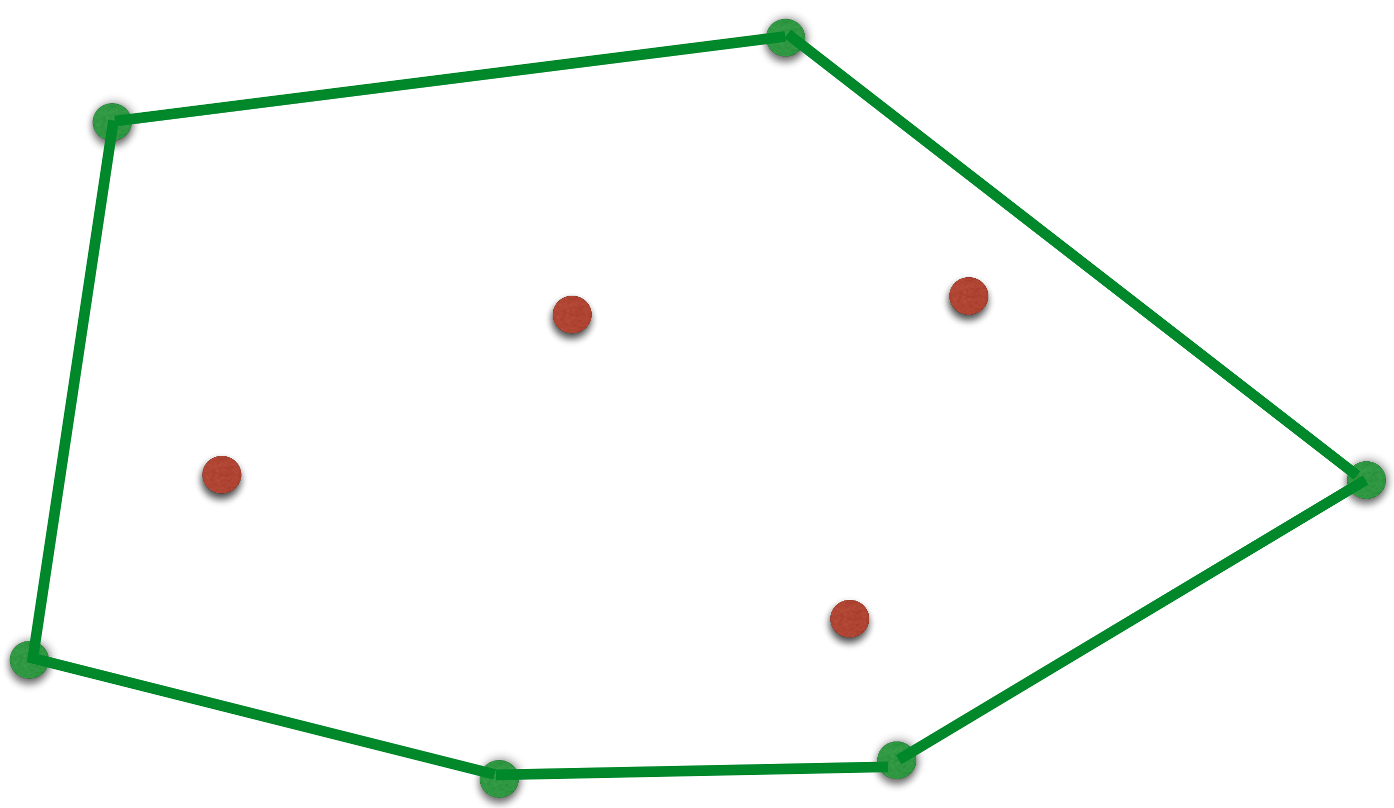
Angelika Steger

Institut für Theoretische Informatik

Kapitel 3.2.2

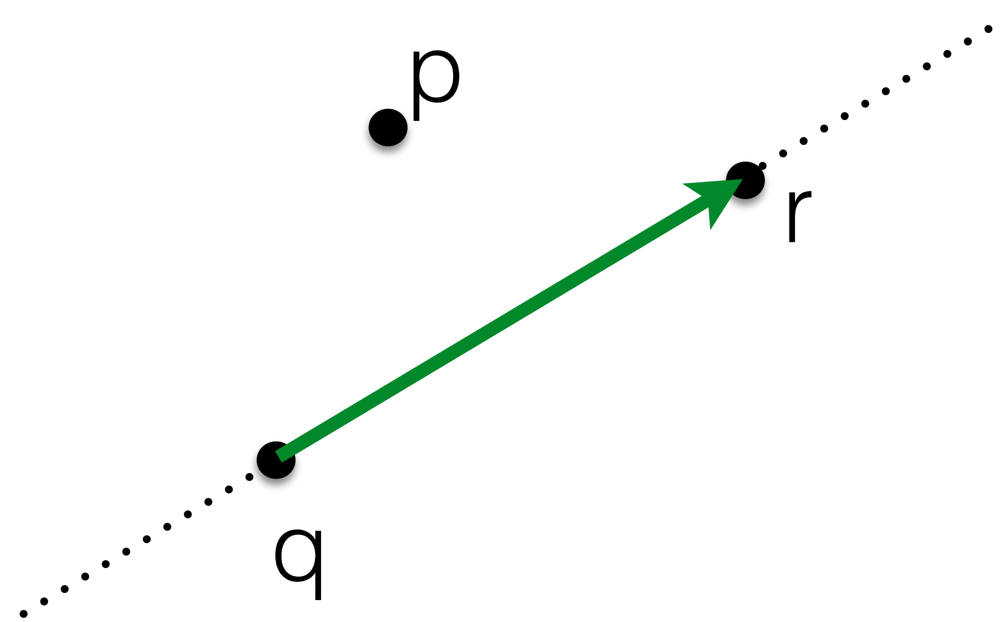
Konvexe Hülle

Gegeben: Punkte $x_1,\ldots,x_n$ in R^2



Annahme Vorlesung: keine drei Punkte auf einer Gerade,
keine zwei Punkte mit gleicher x-Koordinate

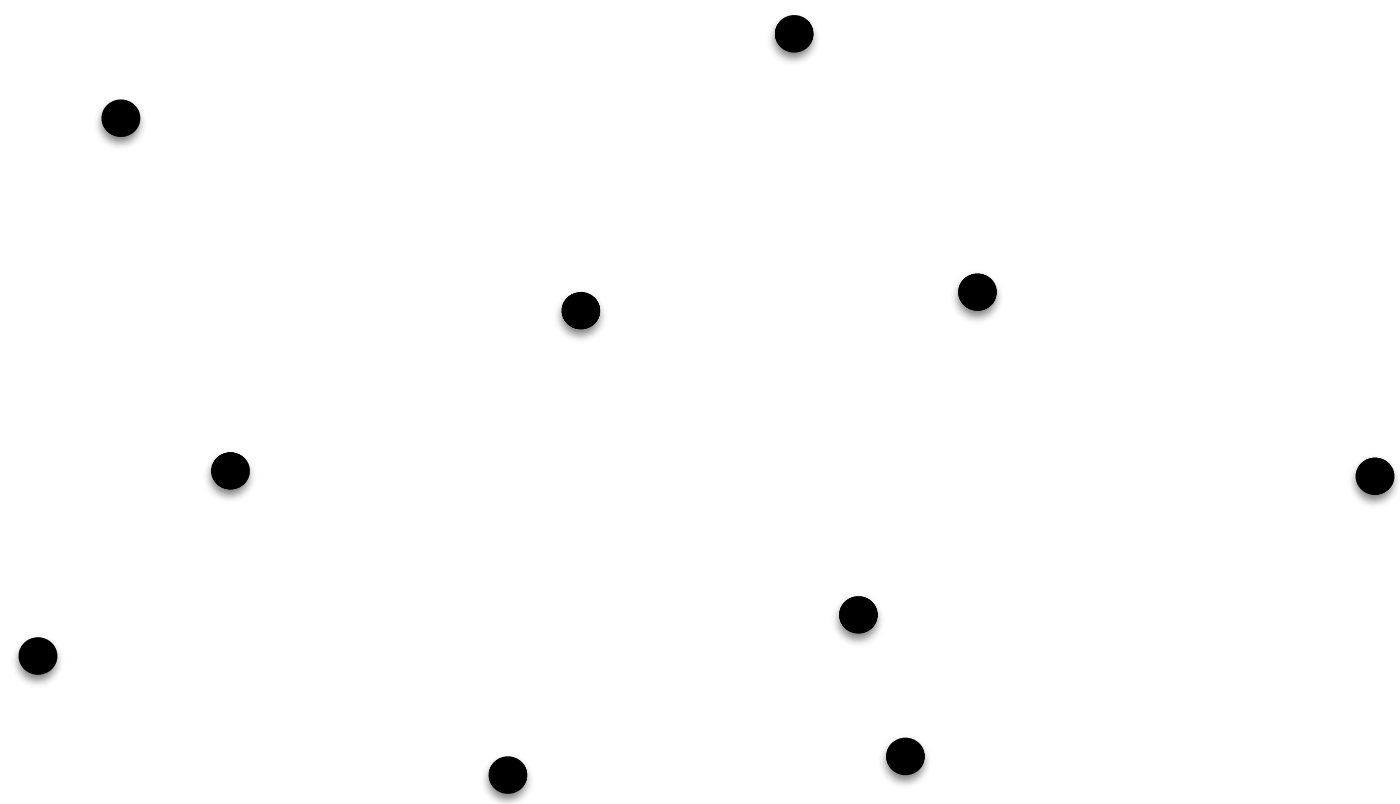
(allg. Fall: siehe Skript)



Lemma 3.35. Seien $p = (p_x, p_y)$, $q = (q_x, q_y)$, und $r = (r_x, r_y)$ Punkte in $\mathbb{R}^2$. Es gilt $q \neq r$ und p liegt links von qr genau dann wenn

$$\begin{aligned} \det(p, q, r) &:= \begin{vmatrix} p_x & p_y & 1 \\ q_x & q_y & 1 \\ r_x & r_y & 1 \end{vmatrix} = \begin{vmatrix} q_x - p_x & q_y - p_y \\ r_x - p_x & r_y - p_y \end{vmatrix} > 0 \\ \Leftrightarrow \quad & (q_x - p_x)(r_y - p_y) > (q_y - p_y)(r_x - p_x) \end{aligned}$$

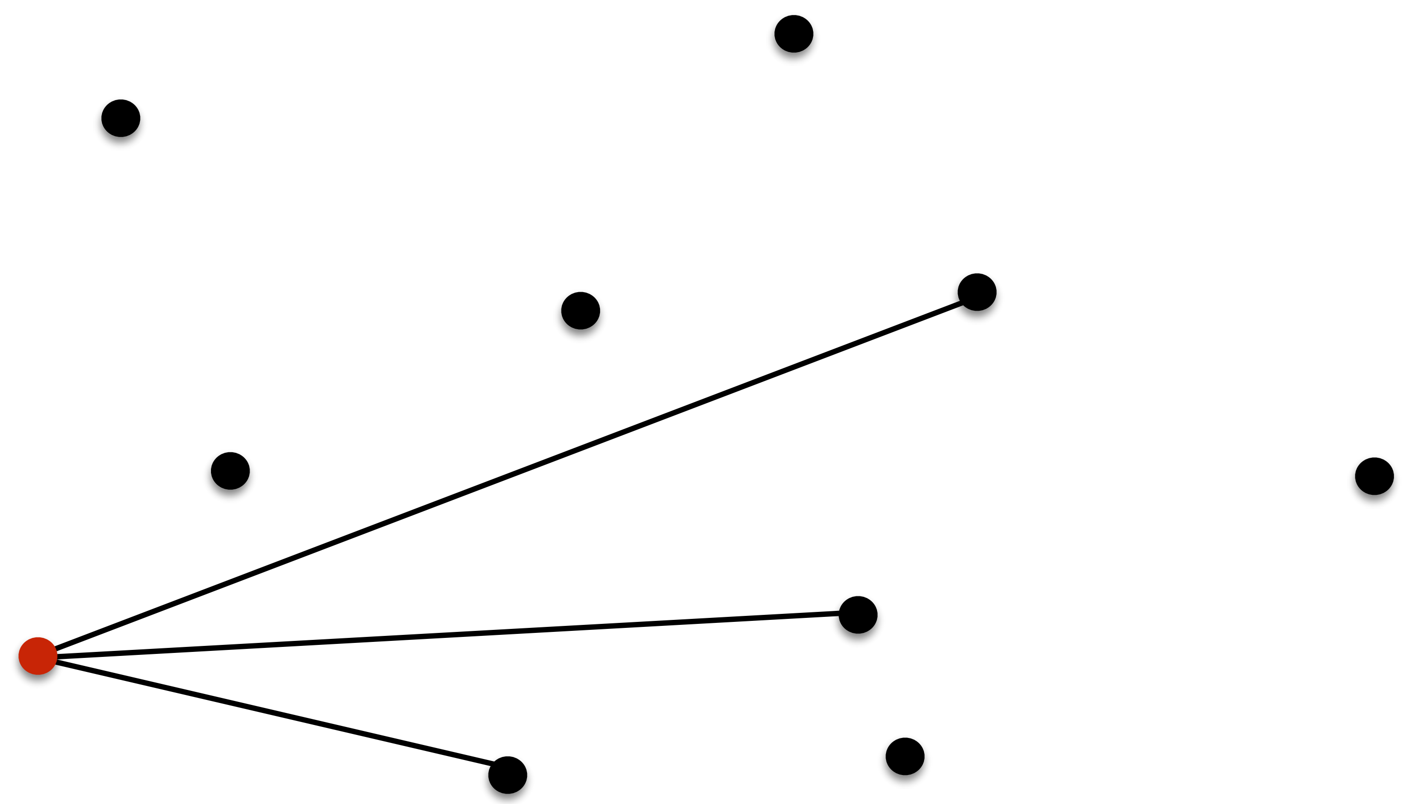
Frage: Wie finden wir die konvexe Hülle?



- ➤ *Es gibt n(n-1) Paare von Knoten, für jedes Paar können wir in O(n) prüfen ob das Paar eine Randkante ist.*
- ➤ *Die Menge aller Randkanten kann in $O(n^3)$ bestimmt werden.*

Konvexe Hülle: Jarvis Wrap

Idee: Betrachtete Punkt mit kleinster x Koordinate



Wie können wir (effizient) ausgehende Randkante bestimmen?

Jarvis' (Einwickel-)Algorithmus

$\textsc{JarvisWrap}(P)$

$h \leftarrow 0$
$p_{\text{now}} \leftarrow$ Punkt in P mit kleinster x-Koordinate
repeat
    $q_h \leftarrow p_{\text{now}}$
    $p_{\text{now}} \leftarrow \textsc{FindNext}(q_h)$
    $h \leftarrow h + 1$
until $p_{\text{now}} = q_0$
return $(q_0, q_1, \ldots, q_{h-1})$

$\textsc{FindNext}(q)$

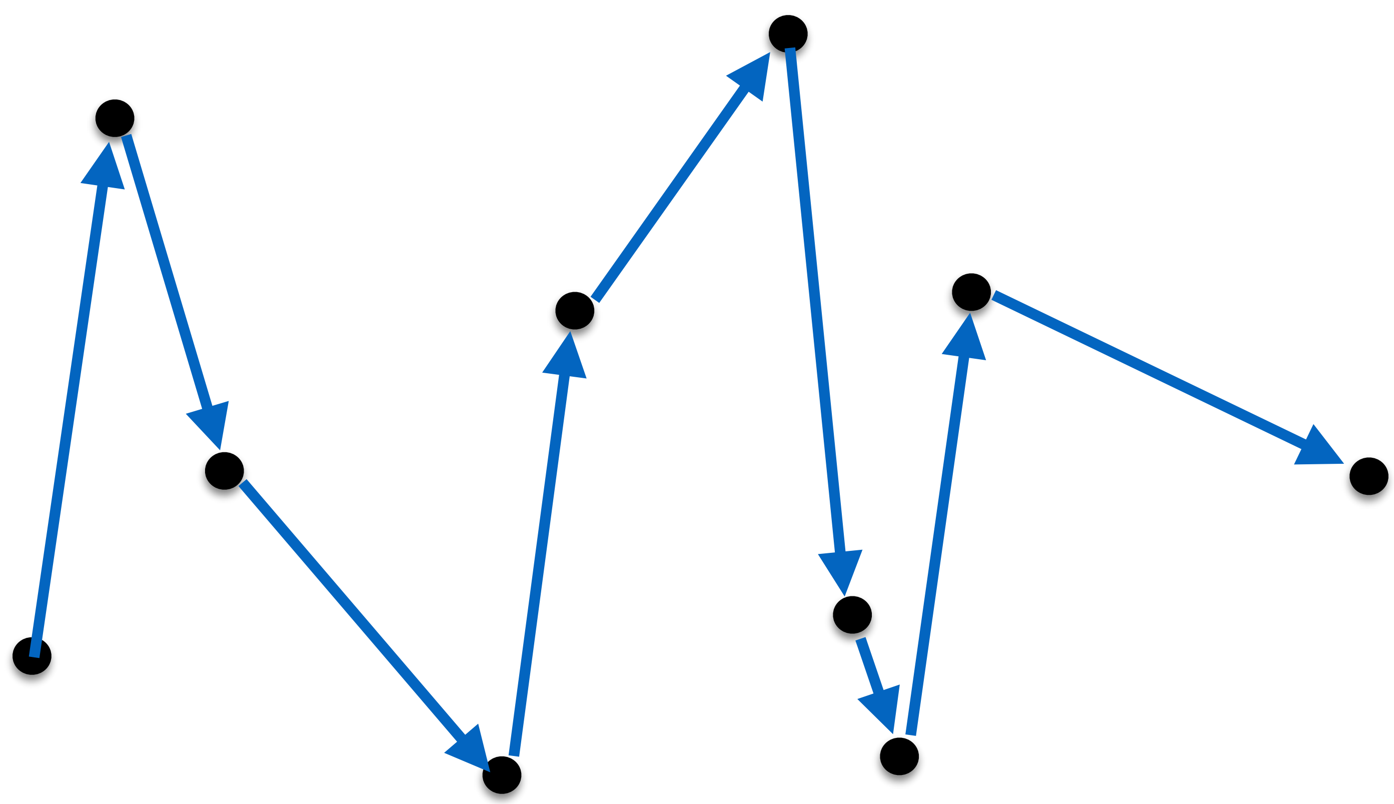
Wähle $p_0 \in P \setminus \{q\}$ beliebig
$q_{\text{next}} \leftarrow p_0$
for all $p \in P \setminus \{q, p_0\}$ **do**
    if p rechts von $q q_{\text{next}}$ **then** $q_{\text{next}} \leftarrow p$
return q_{next}

Einwickelalgorithmus

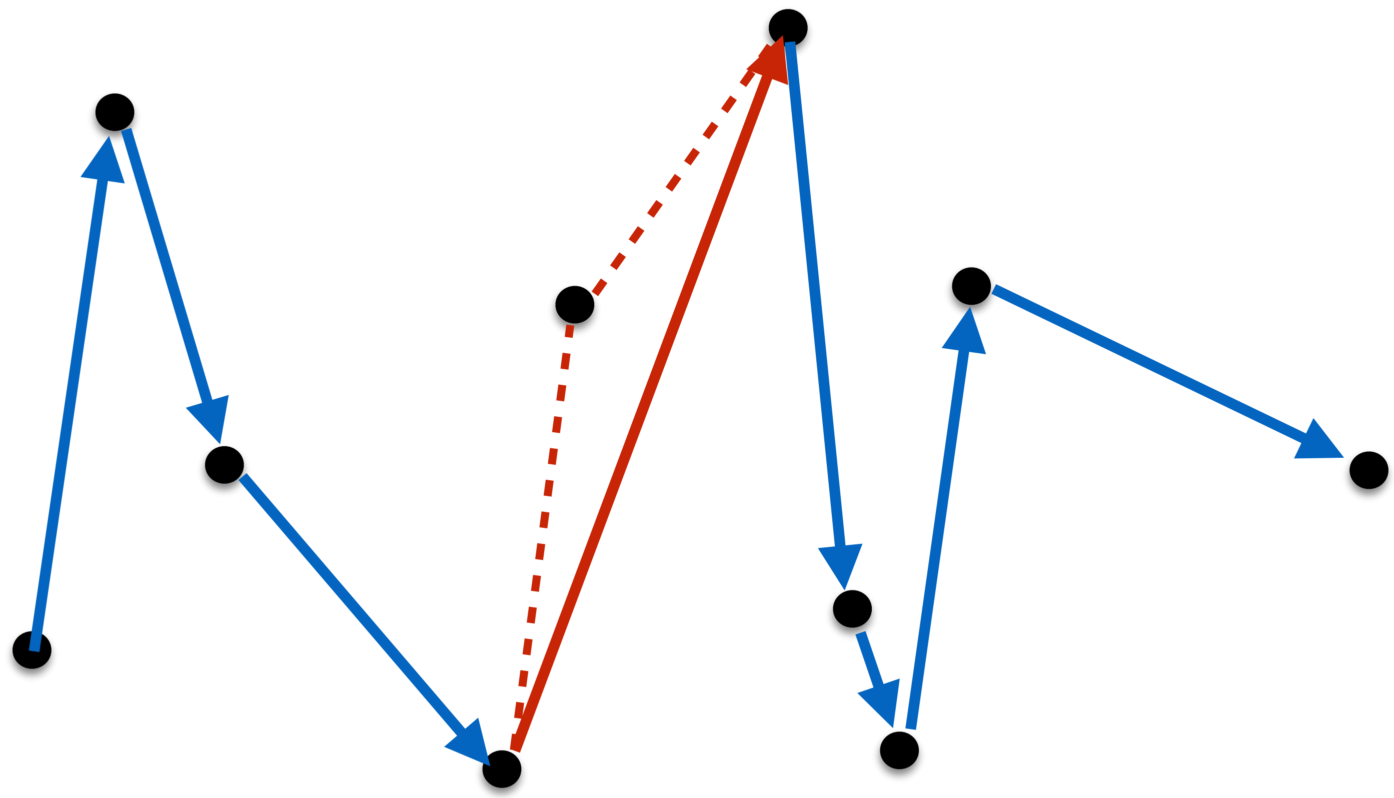
Satz Gegeben eine Menge P von n Punkten in allgemeiner Lage in $\mathbb{R}^2$, berechnet der Algorithmus JarvisWrap die konvexe Hülle in Zeit $O(nh)$, wobei h die Anzahl der Ecken der konvexen Hülle von P ist.

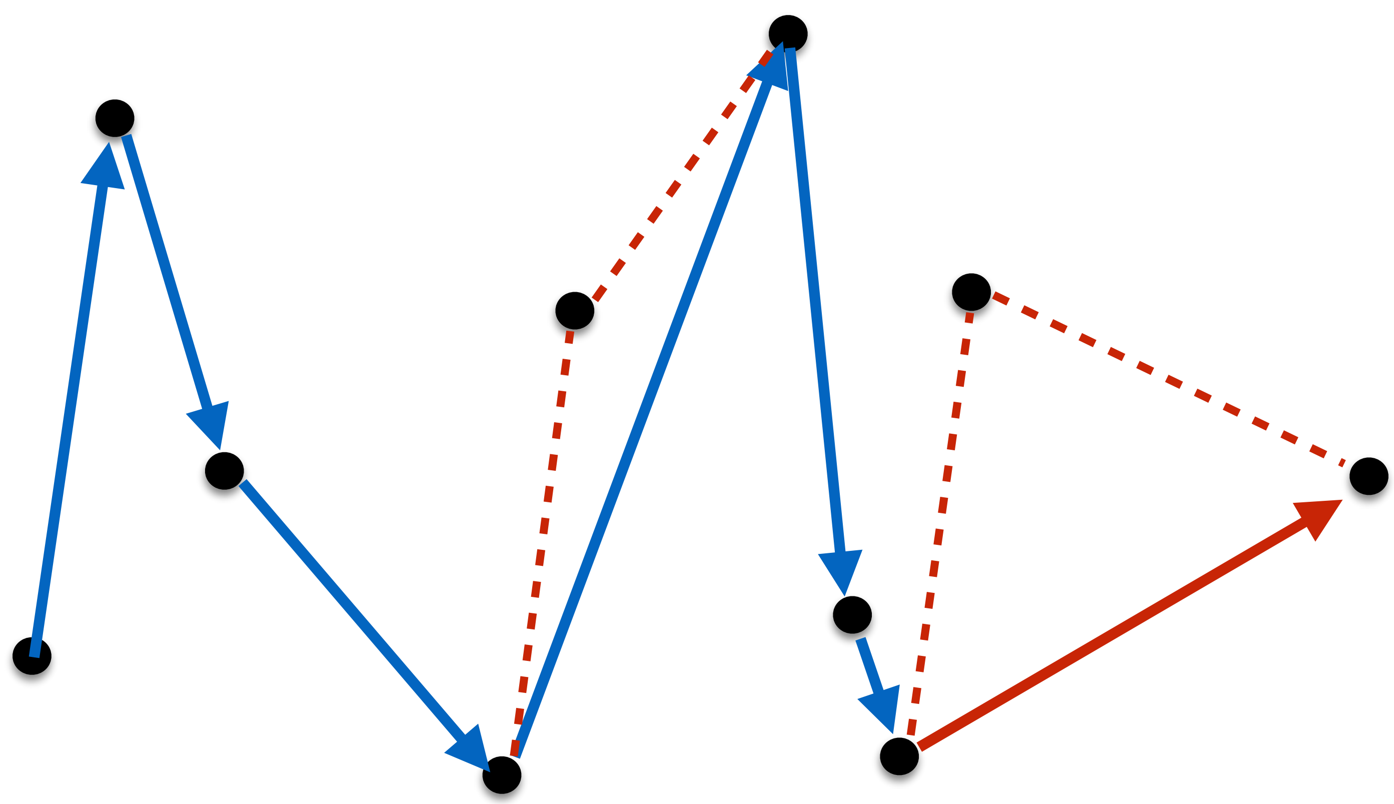
Ziel für heute: O(n log n)

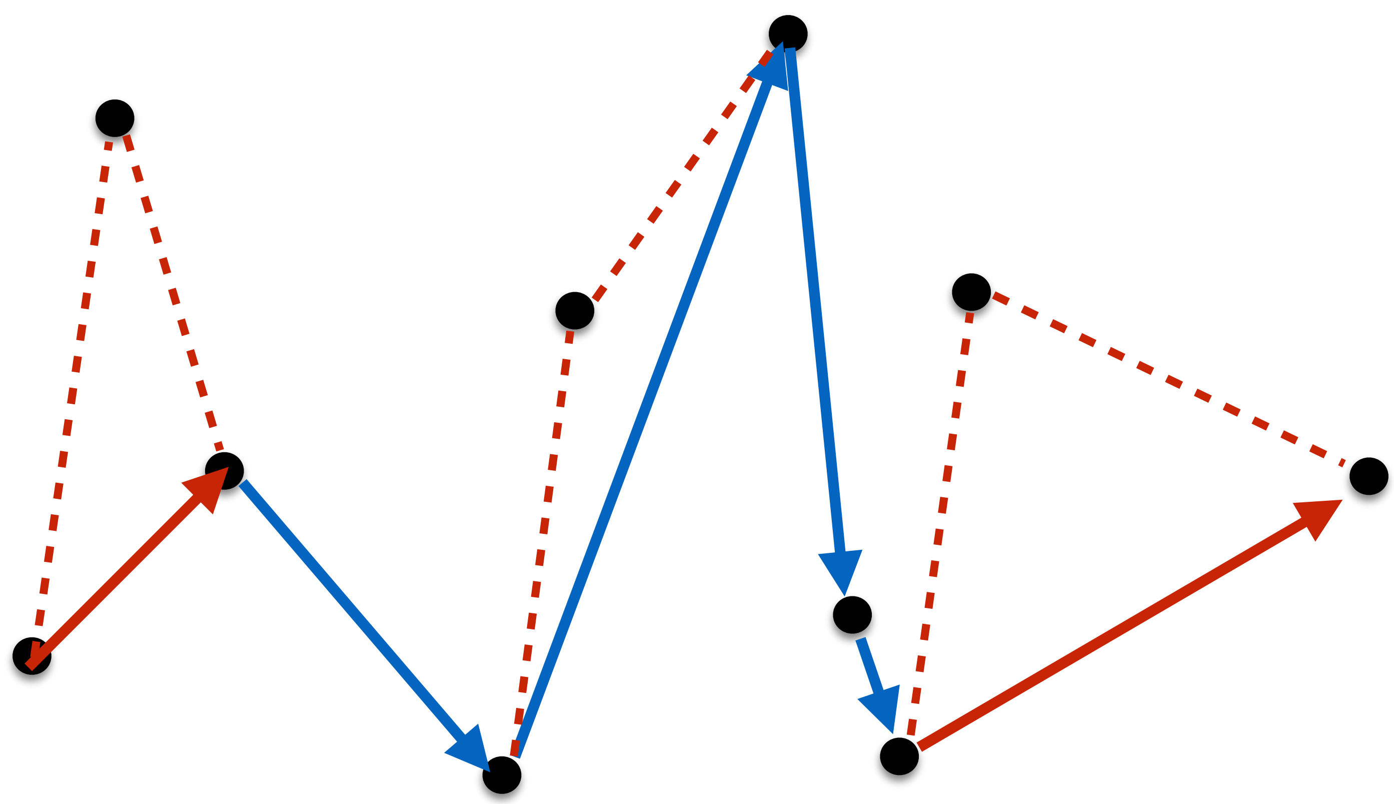
Idee: Kantenzug von „links“ nach „rechts“

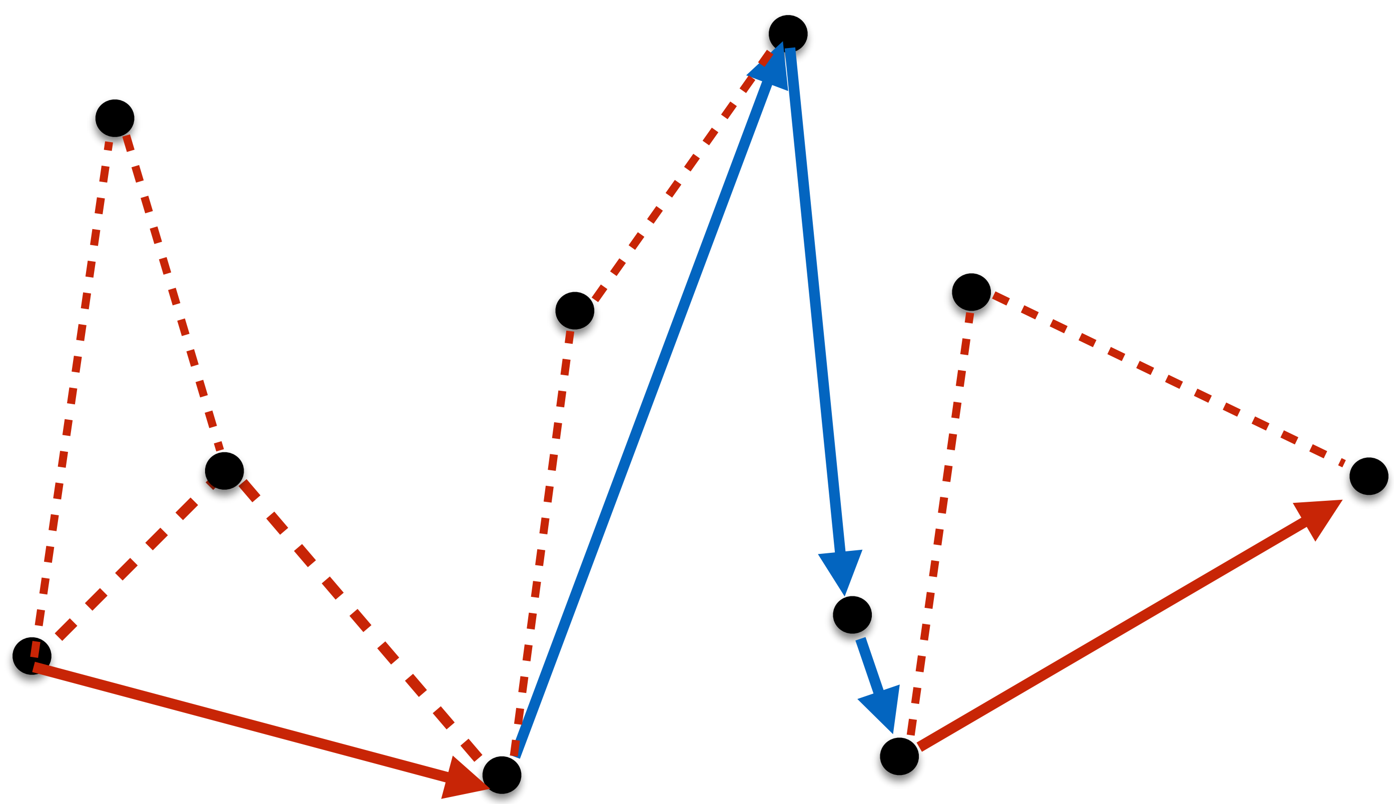


Algorithmus: sukzessives „ausbessern“

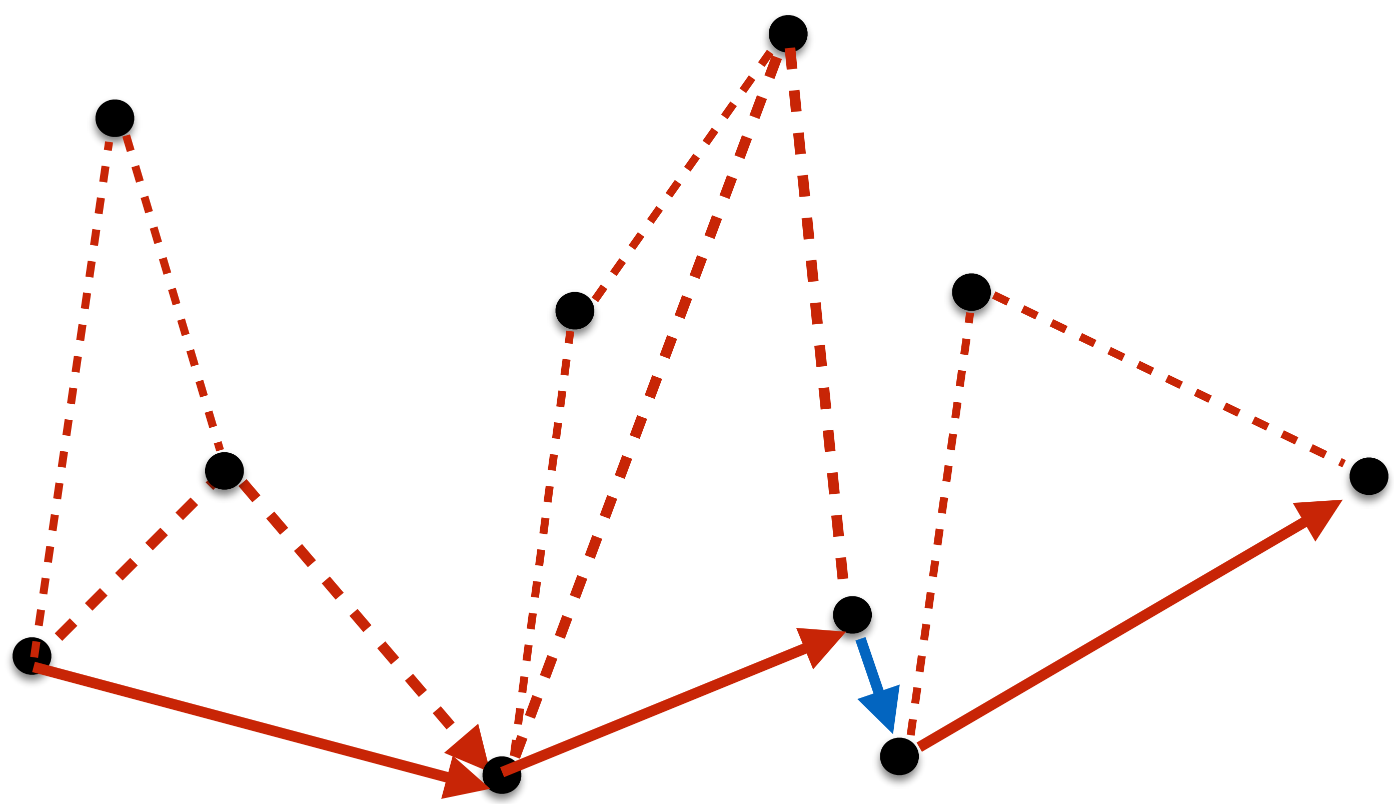


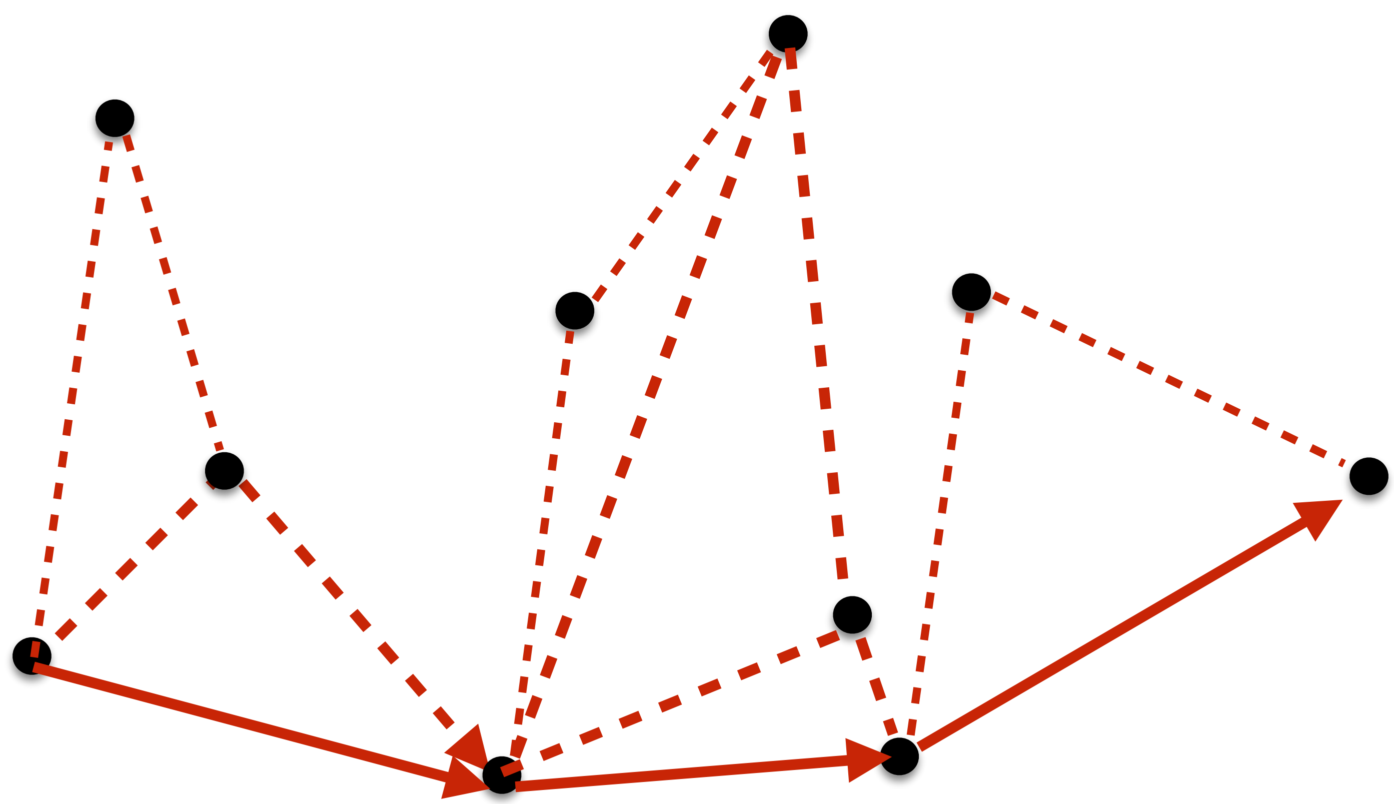




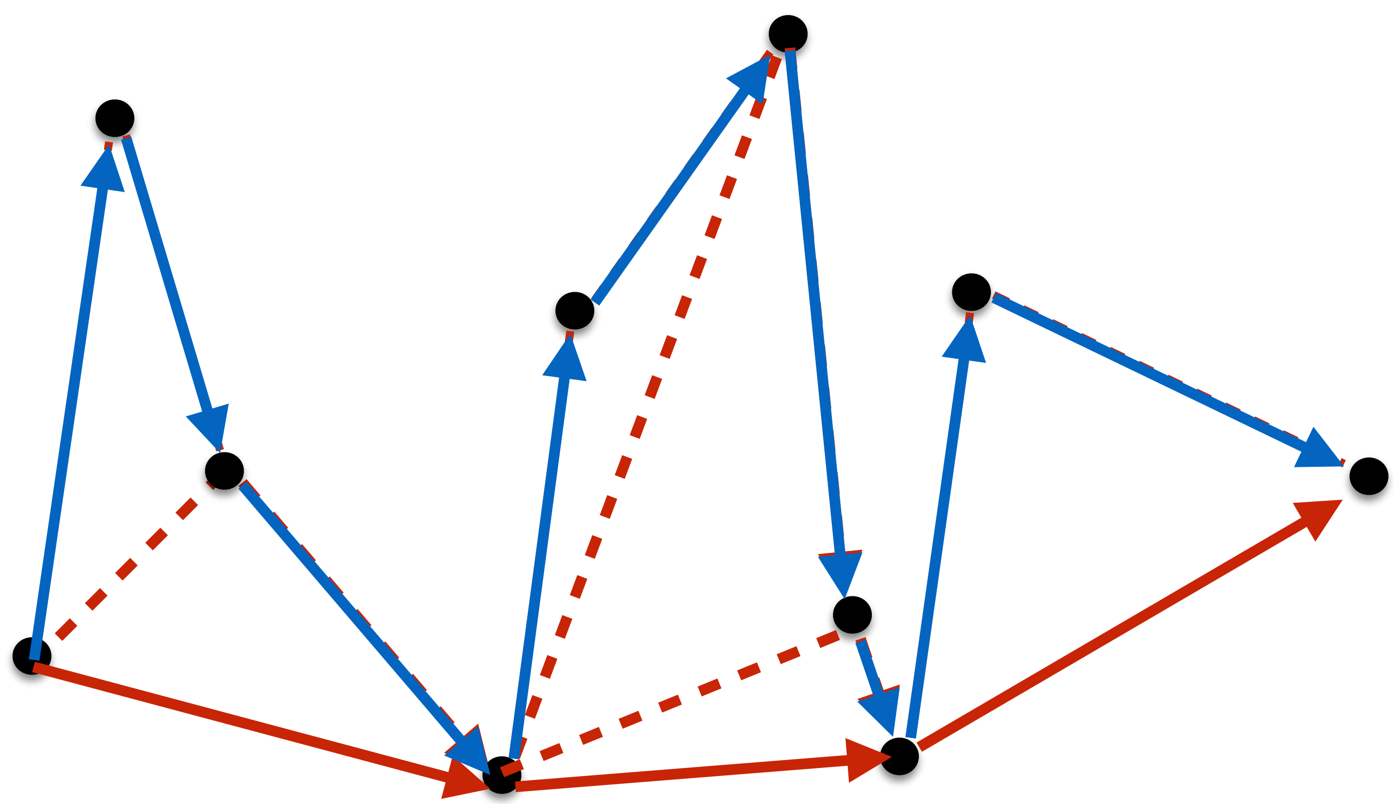


Konvexe Hülle



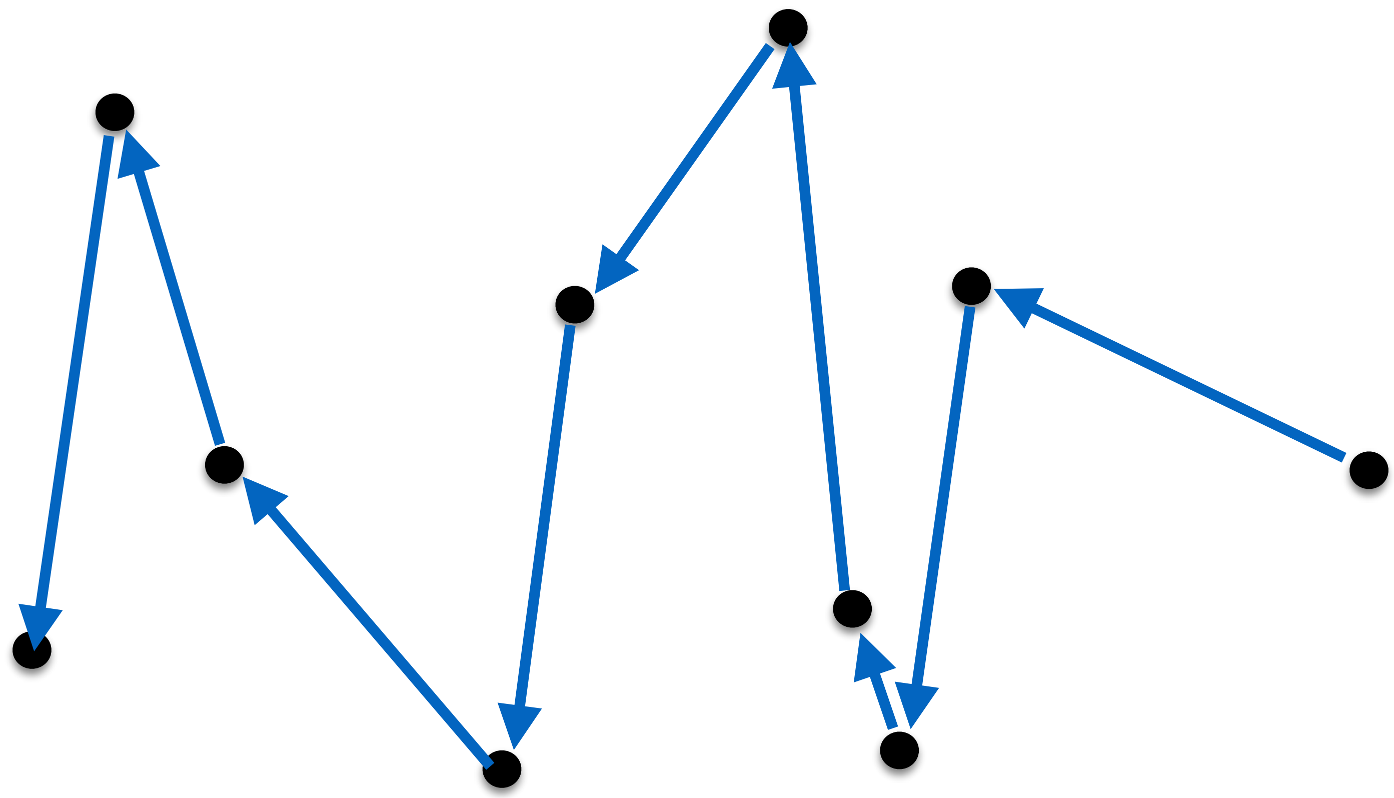


blau: Ausgangs-Kantenzug von „links“ nach „rechts“

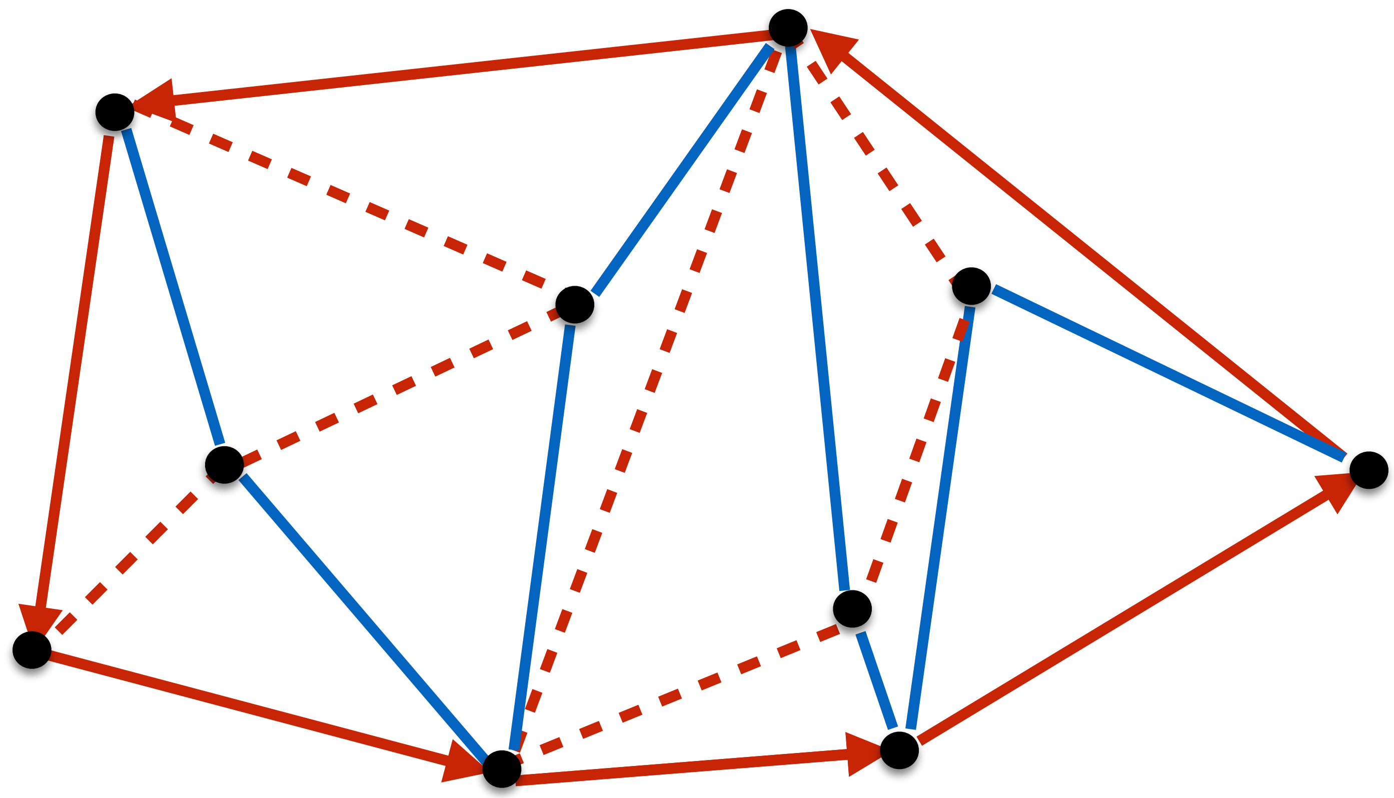


rot: neu gefundene Kanten

2.Schritt: Kantenzug von „rechts“ nach „links“



2.Schritt: Kantenzug von „rechts“ nach „links“



Idee für Laufzeitabschätzung:

Anzahl Verbesserungen = Anzahl Dreiecke

Definition Ein Graph $G = (P, E)$ auf P heisst *eben*, wenn sich die Segmente $pq := \mathrm{conv}(\{p, q\})$ der Kanten $\{p, q\} \in E$ höchstens in ihren jeweils gemeinsamen Endpunkten schneiden. Entfernen wir die Segmente pq, $\{p, q\} \in E$, aus $\mathbb{R}^2$, so nennen wir die entstehenden zusammenhängenden Gebiete die *Gebiete von* G. Die beschränkten Gebiete nennen wir *innere Gebiete*, das unbeschränkte (unendliche) Gebiet das *äussere Gebiet*.

Ein Graph $T = (P, E)$ auf P heisst *Triangulierung von* P, falls T eben ist und maximal mit dieser Eigenschaft ist, d.h. das Hinzufügen jeder Kante $\binom{P}{2} \setminus E$ zu T verletzt die Eigenschaft "eben".

Es gilt:
- Die inneren Gebiete einer Triangulierung sind alles Dreiecke.
- Das einzige äussere Gebiet ist das Komplement von conv(P) in R^2 und liegt genau an den Randkanten von P an.

Definition Ein Graph $G = (P, E)$ auf P heisst *eben*, wenn sich die Segmente $pq := \mathrm{conv}(\{p, q\})$ der Kanten $\{p, q\} \in E$ höchstens in ihren jeweils gemeinsamen Endpunkten schneiden. Entfernen wir die Segmente pq, $\{p, q\} \in E$, aus $\mathbb{R}^2$, so nennen wir die entstehenden zusammenhängenden Gebiete die *Gebiete von* G. Die beschränkten Gebiete nennen wir *innere Gebiete*, das unbeschränkte (unendliche) Gebiet das *äussere Gebiet*.

Ein Graph $T = (P, E)$ auf P heisst *Triangulierung von* P, falls T eben ist und maximal mit dieser Eigenschaft ist, d.h. das Hinzufügen jeder Kante $\binom{P}{2} \setminus E$ zu T verletzt die Eigenschaft “eben”.

Lemma Sei h die Anzahl der Ecken der konvexen Hülle von P. Der lokale Verbesserungsprozess macht genau $2n - 2 - h$ Verbesserungsschritte und erzeugt eine Triangulierung mit $2n-2-h$ Dreiecken und $3n - 3 - h$ Kanten.

Lemma Sei h die Anzahl der Ecken der konvexen Hülle von P. Der lokale Verbesserungsprozess macht genau $2n - 2 - h$ Verbesserungsschritte und erzeugt eine Triangulierung mit $2n-2-h$ Dreiecken und $3n - 3 - h$ Kanten.

Beweis: Betrachte Verbesserungsalgorithmus:

zu Beginn: 2(n-1) gerichtete Kanten, **0 Dreiecke**

jeder Verbesserungsschritt:

- zwei gerichtete Kanten werden zu ungerichteten Kanten
- eine neue gerichtete Kante wird eingefügt, **1 Dreieck entsteht**

am Ende: h gerichtete Kanten (die Kanten der konvexen Hülle)

$\Rightarrow$ # Schritte = 2(n-1) - h

… **ebenso viele Dreieck sind entstanden**

Lemma Sei h die Anzahl der Ecken der konvexen Hülle von P. Der lokale Verbesserungsprozess macht genau $2n-2-h$ Verbesserungsschritte und erzeugt eine Triangulierung mit $2n-2-h$ Dreiecken und $3n-3-h$ Kanten.

Beweis: Betrachte leicht modifizierte Version des Verbesserungsalgorithmus: (nur ungerichtete Kanten)

zu Beginn: n-1 Kanten
jeder Verbesserungsschritt:
- eine neue Kante wird eingefügt

am Ende: h Kanten (die Kanten der konvexen Hülle)

$\Rightarrow$ # Kanten = n-1 + #Schritte = 3n - 3 -h

Exkurs: Eulersche Formel

Lemma Sei h die Anzahl der Ecken der konvexen Hülle von P. Der lokale Verbesserungsprozess macht genau $2n - 2 - h$ Verbesserungsschritte und erzeugt eine Triangulierung mit $2n-2-h$ Dreiecken und $3n - 3 - h$ Kanten.

n Punkte
h Punkte auf konvexer Hülle

$\Rightarrow$

2n-2-h Dreiecke
3n-3-h Kanten

Lemma (Euler Relation) Sei $G = (P, E)$ ein ebener Graph auf P mit $v := |P|$ Knoten, $e := |E|$ Kanten, f Gebieten (inkl. des äusseren Gebiets), und c Zusammenhangskomponenten. Dann gilt

$$v - e + f = 1 + c\ .$$

$$\mathbf{n - (3n\text{-}3\text{-}h) \ + \ (2n\text{-}2\text{-}h\text{+}1) = 1\text{+}1}$$

Def: Ein Graph $G = (V,E)$ heisst **planar**, wenn man ihn in der Ebene so zeichnen kann, dass sich keine zwei Kanten kreuzen.

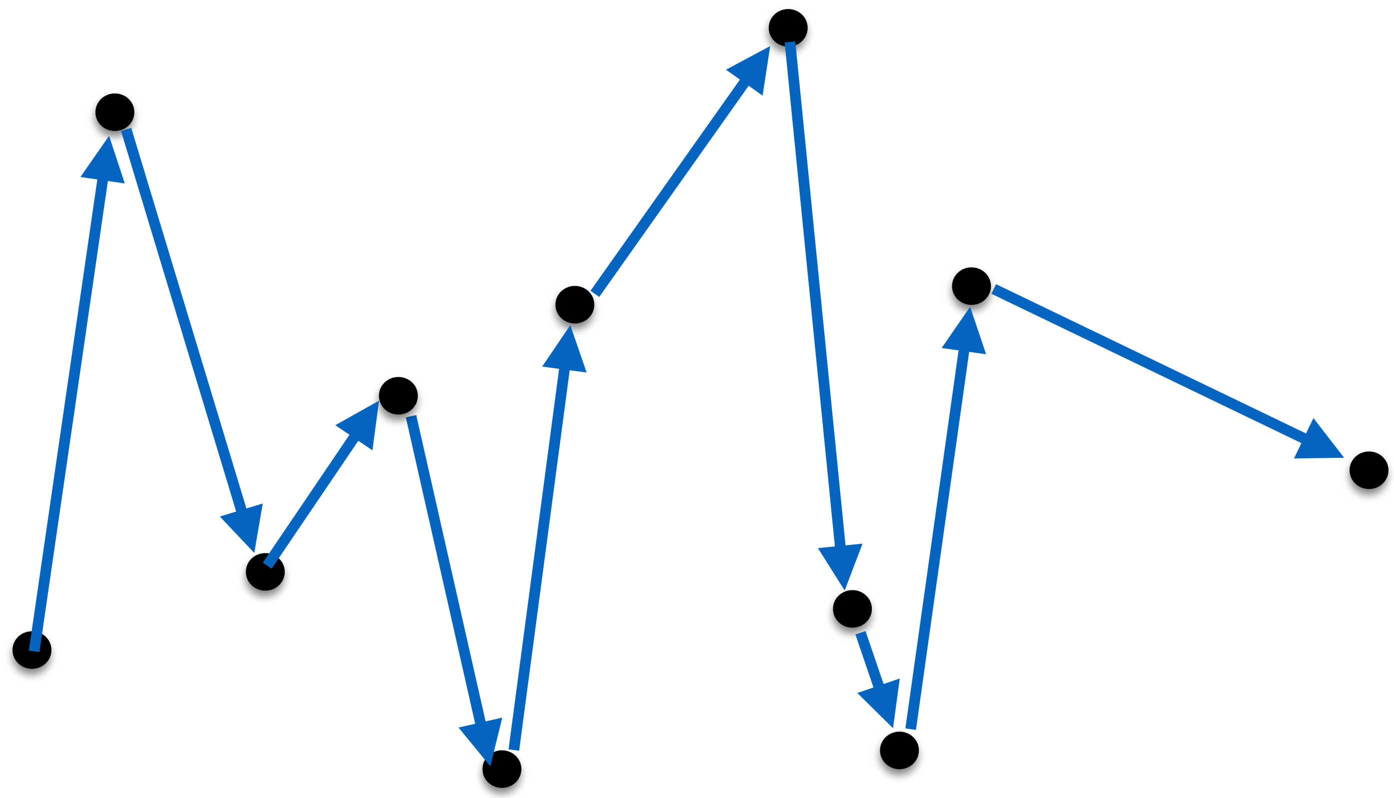
Man kann zeigen: planare Graphen kann man immer auch so zeichnen, dass alle Kanten gerade Linien sind:

Satz: Für jeden **planaren** Graph $G = (V,E)$ gibt es eine **Einbettung** $P : V \to \mathbb{R}^2$, so dass $G = (P(V), E)$ ein **ebener** Graph ist.

Kor: Für jeden **planaren** Graph $G = (V,E)$ gilt

$$|E| \leq 3|V| - 6$$

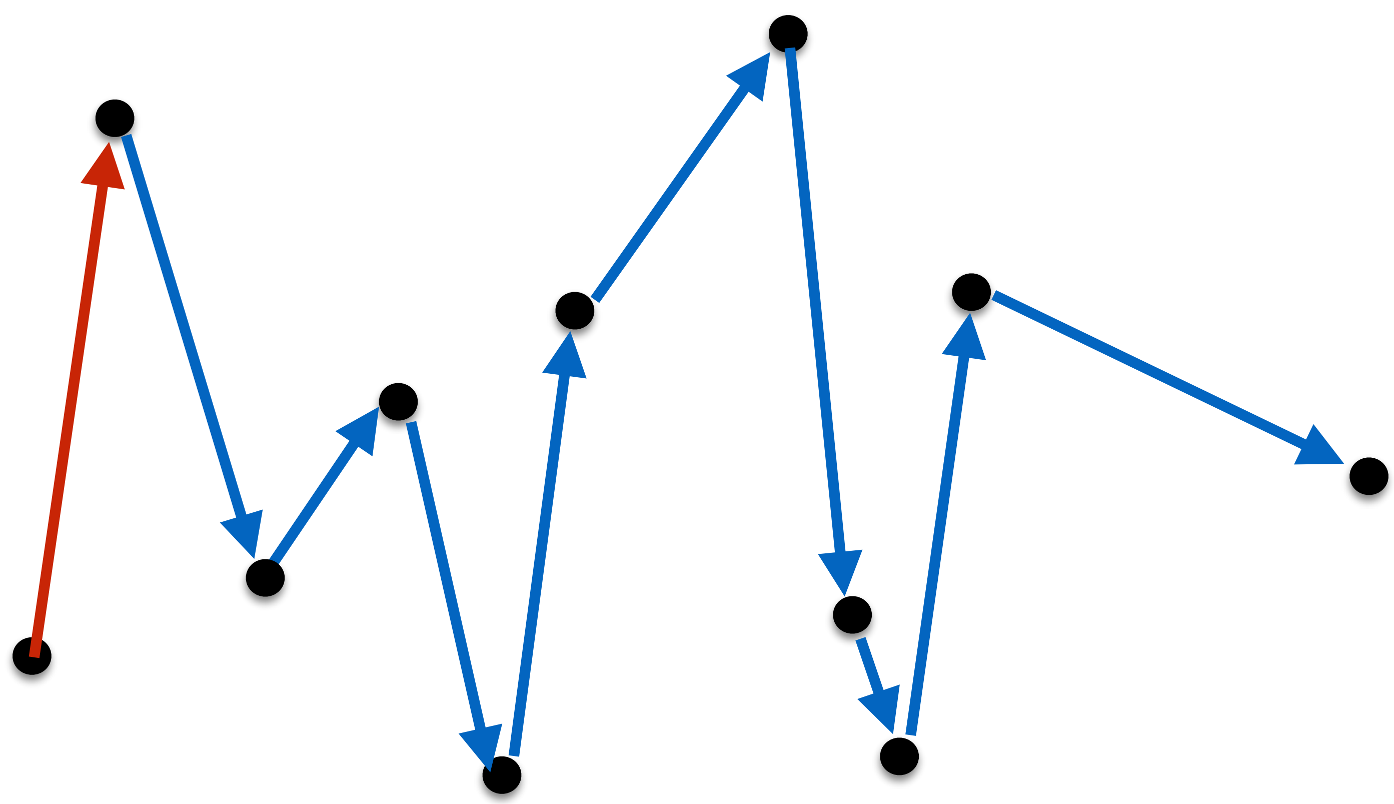
Idee: Kantenzug von „links“ nach „rechts“



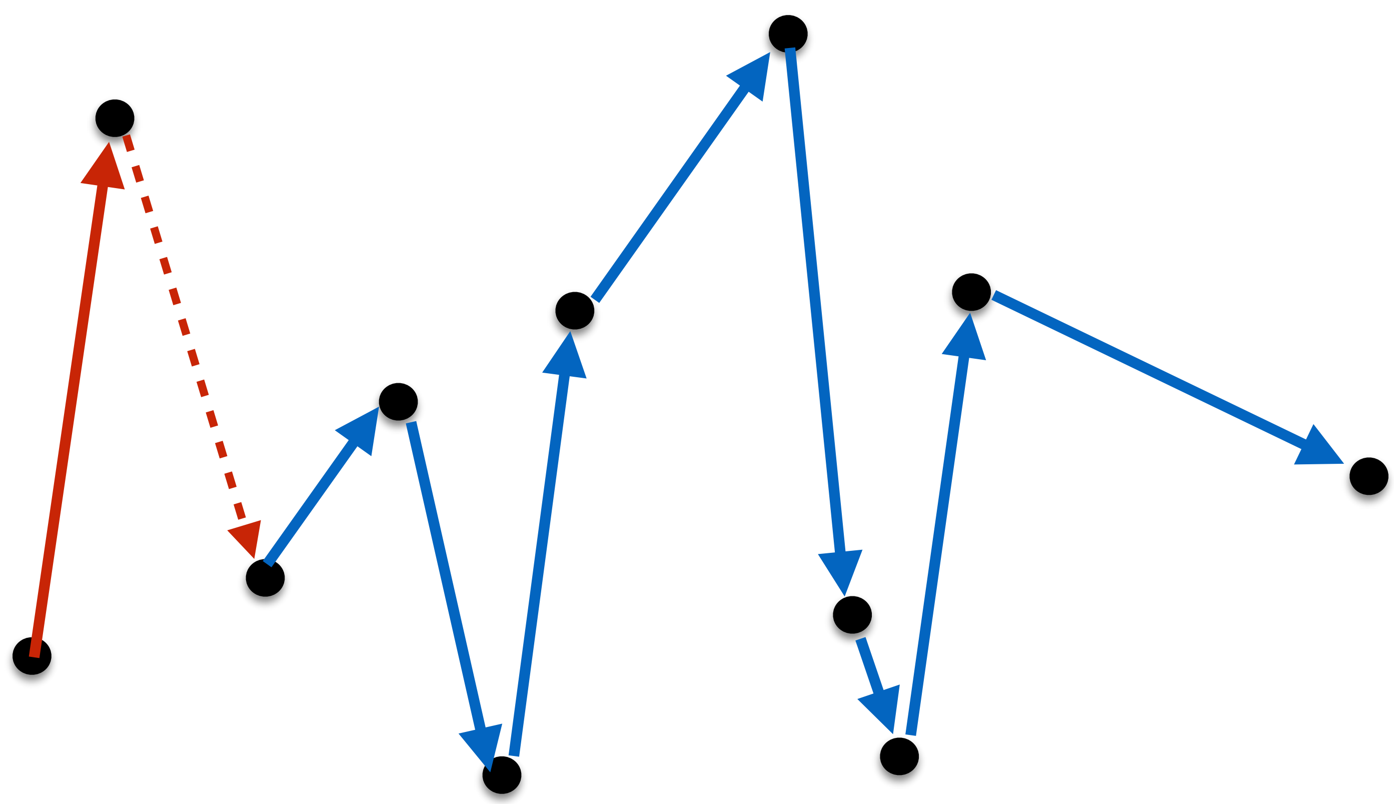
Algorithmus: *sukzessives „ausbessern“*

Frage: Wie finden wir die „Ausbesserungen“ effizient?

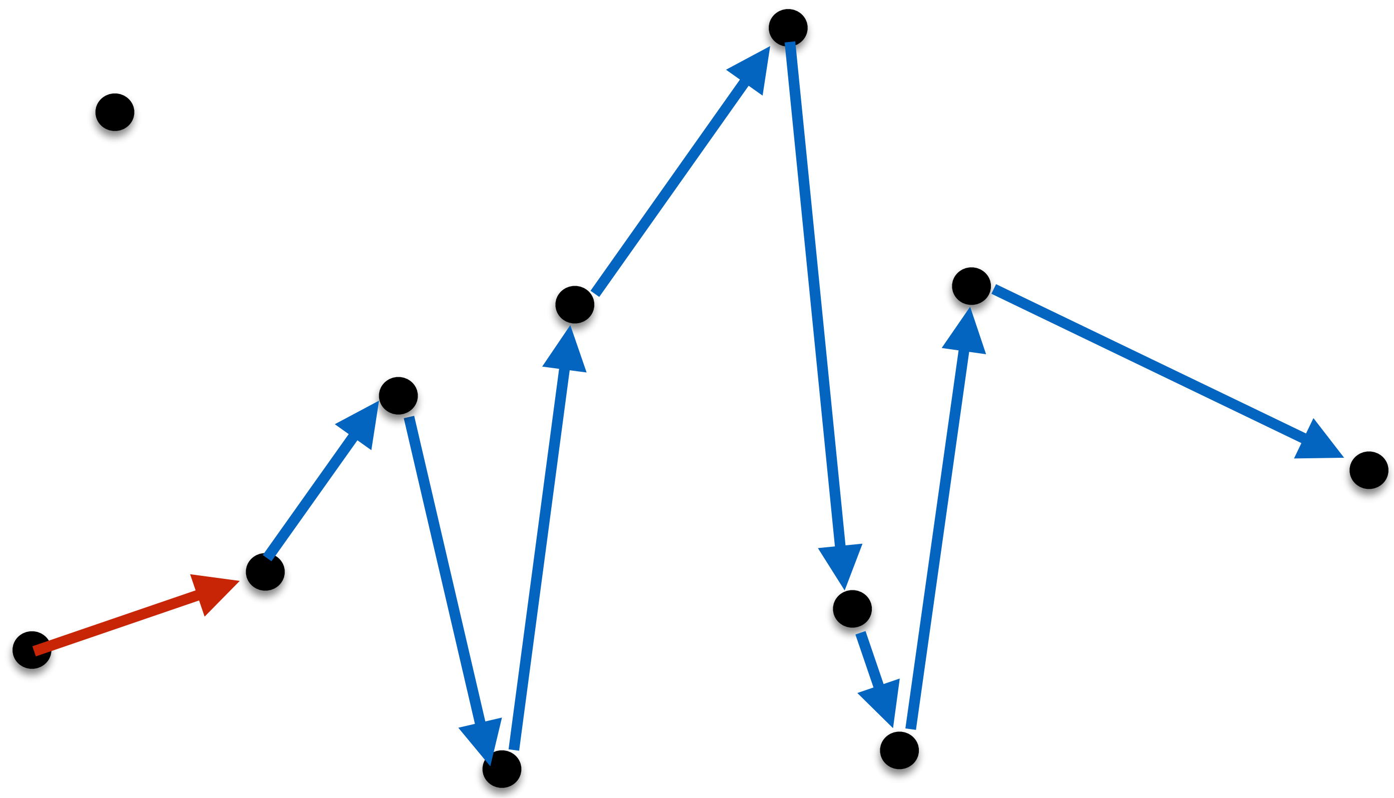
Idee: Konstruiere untere Hülle von „links“ nach „rechts“



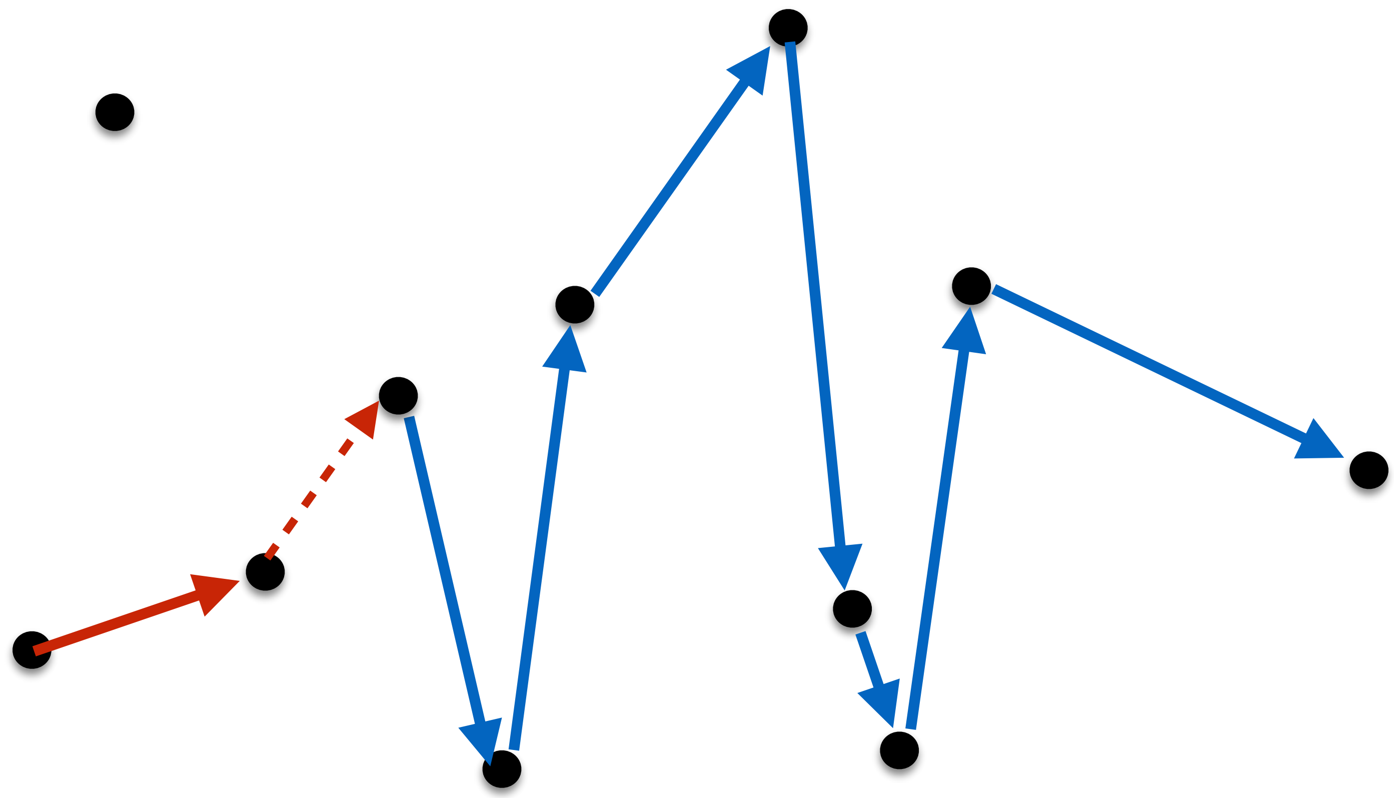
Idee: Konstruiere untere Hülle von „links“ nach „rechts“



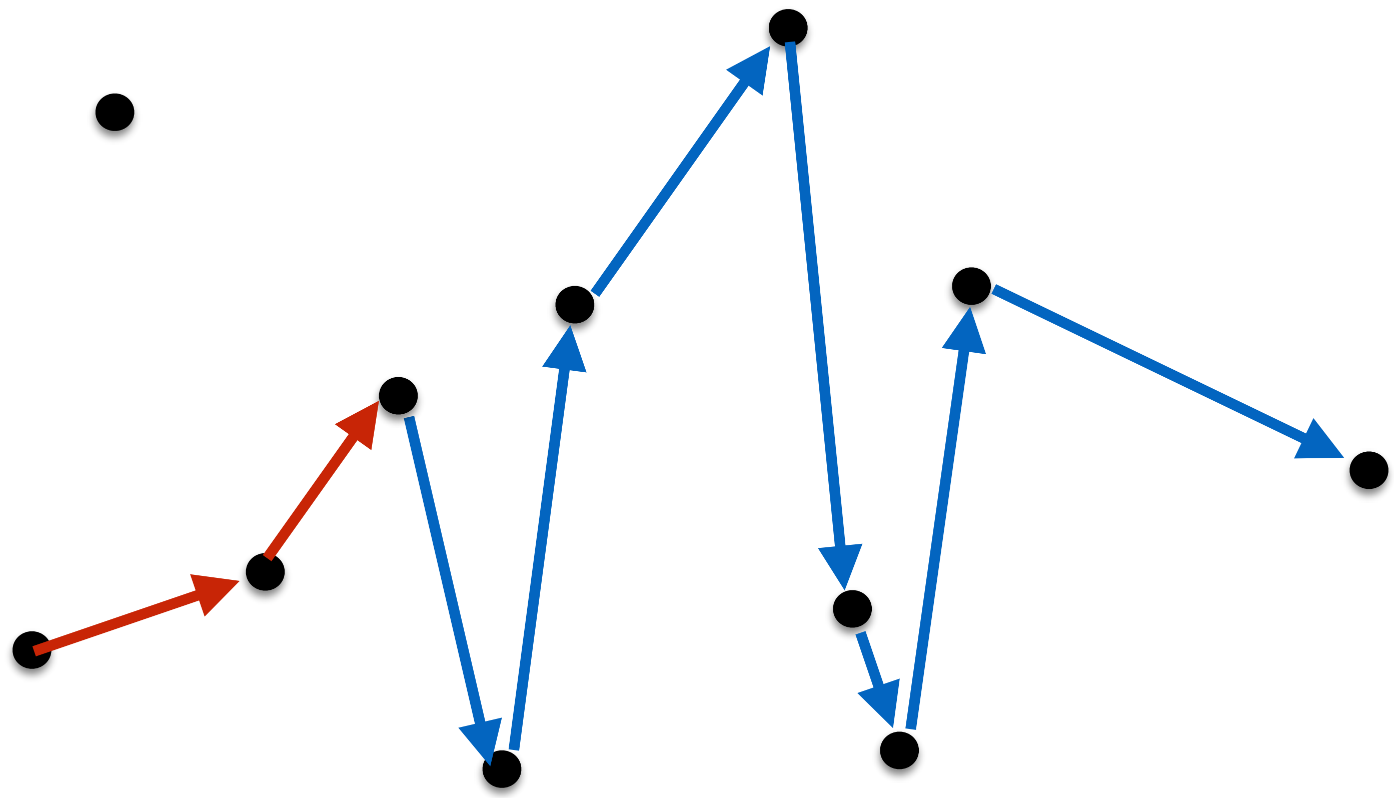
Idee: Konstruiere untere Hülle von „links“ nach „rechts“



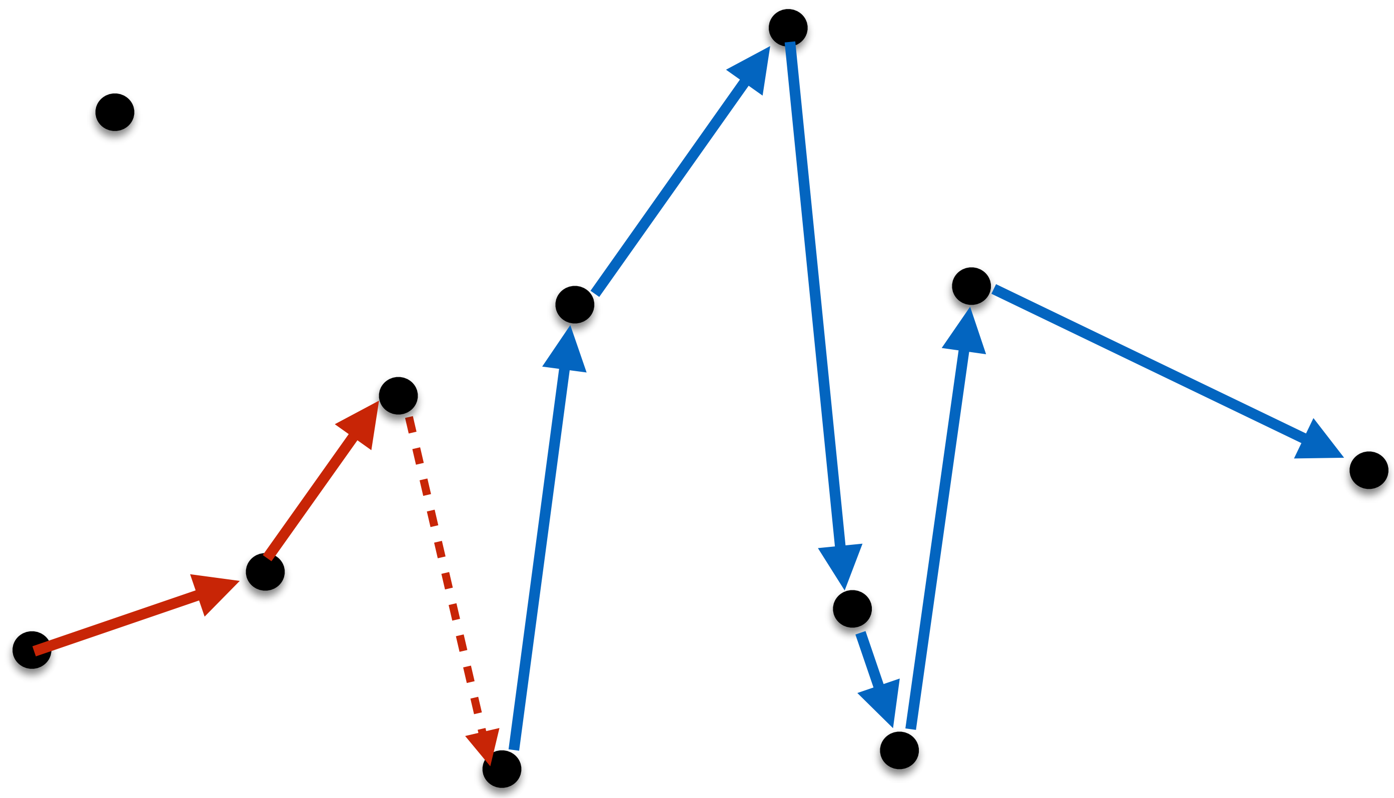
Idee: Konstruiere untere Hülle von „links“ nach „rechts“



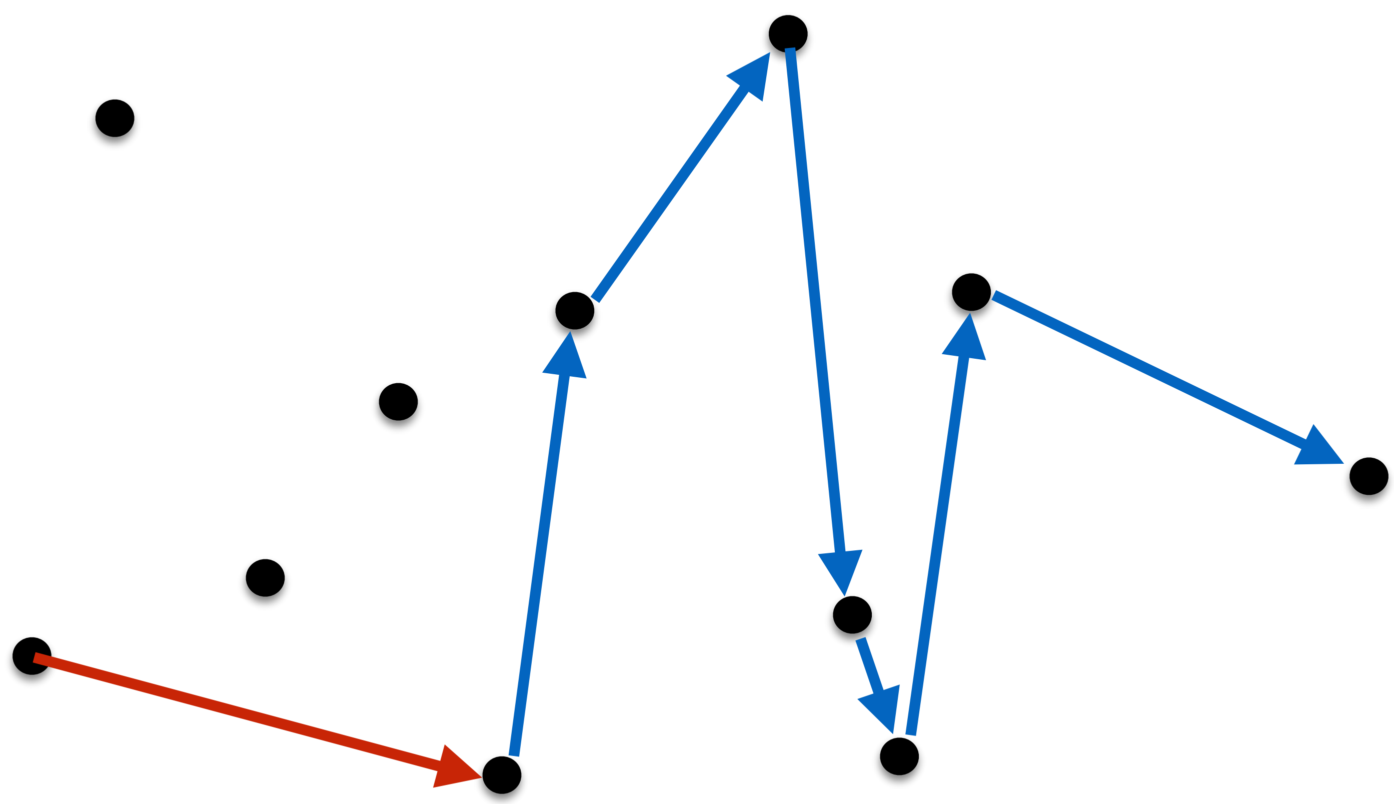
Idee: Konstruiere untere Hülle von „links“ nach „rechts“



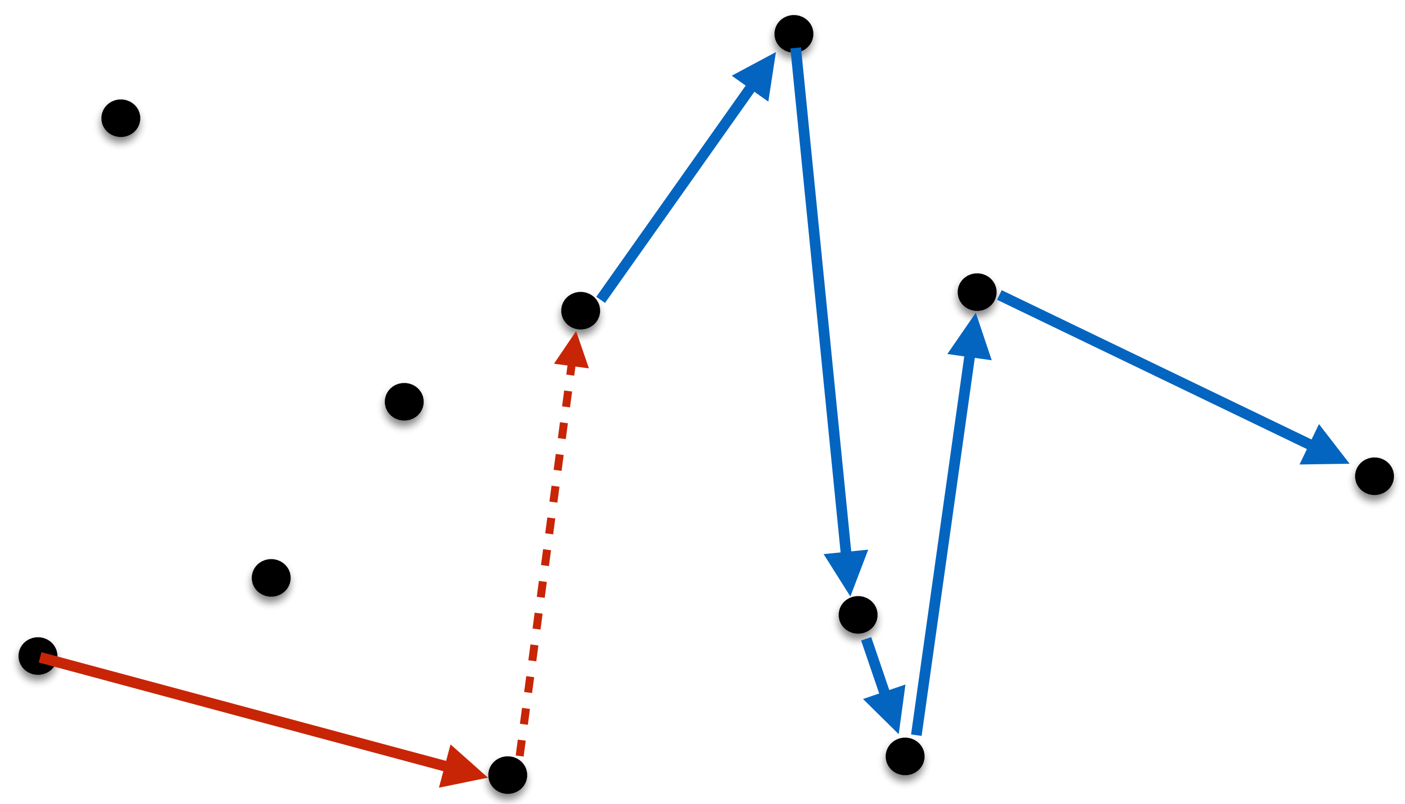
Idee: Konstruiere untere Hülle von „links“ nach „rechts“



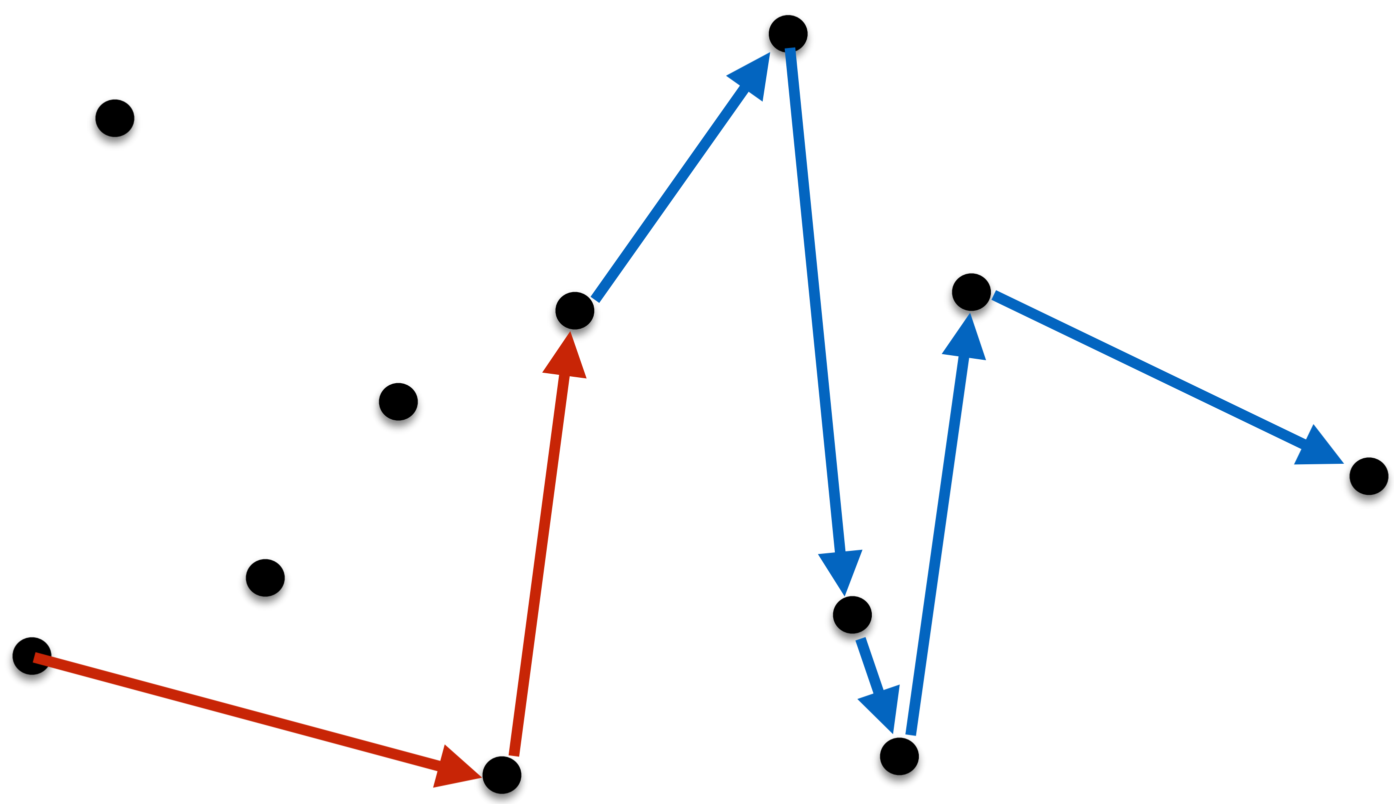
Idee: Konstruiere untere Hülle von „links“ nach „rechts“



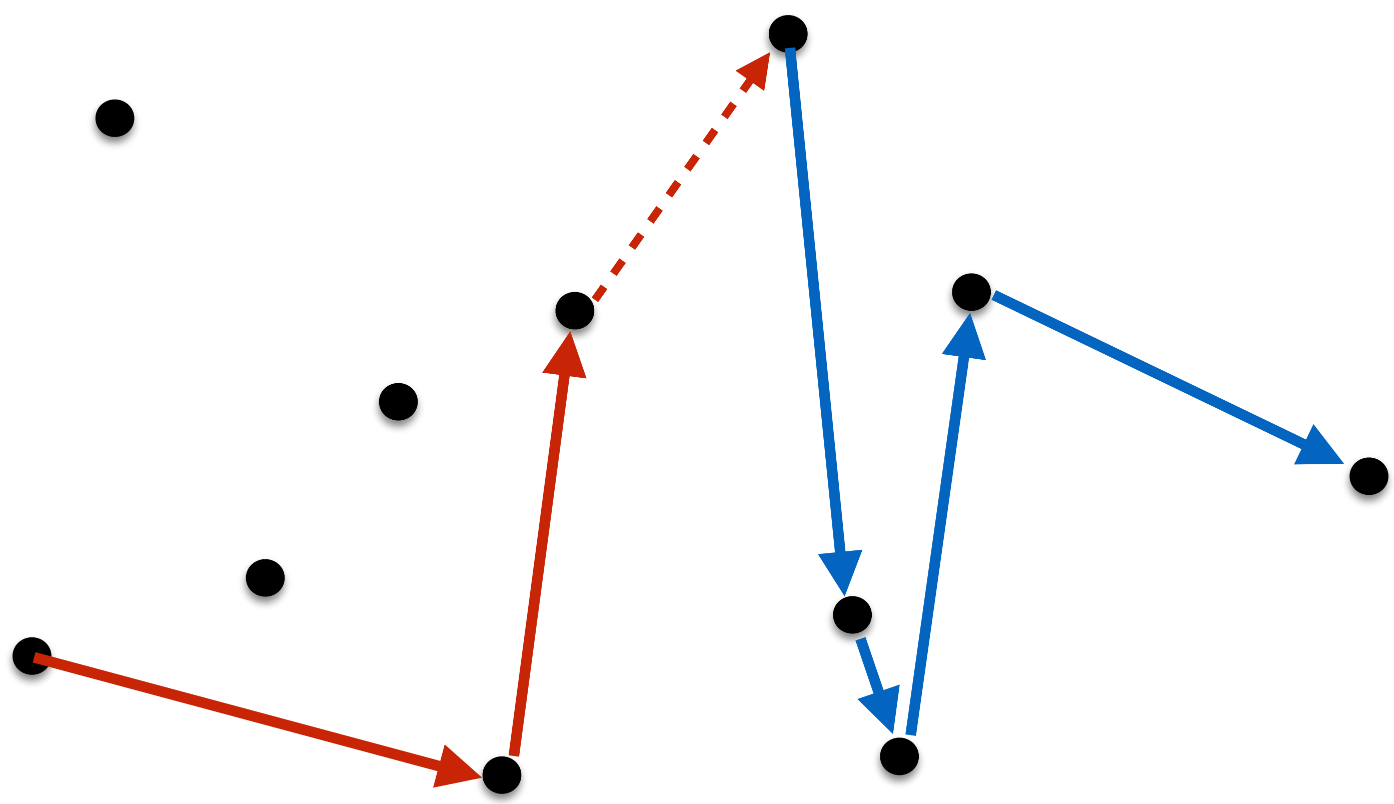
Idee: Konstruiere untere Hülle von „links“ nach „rechts“



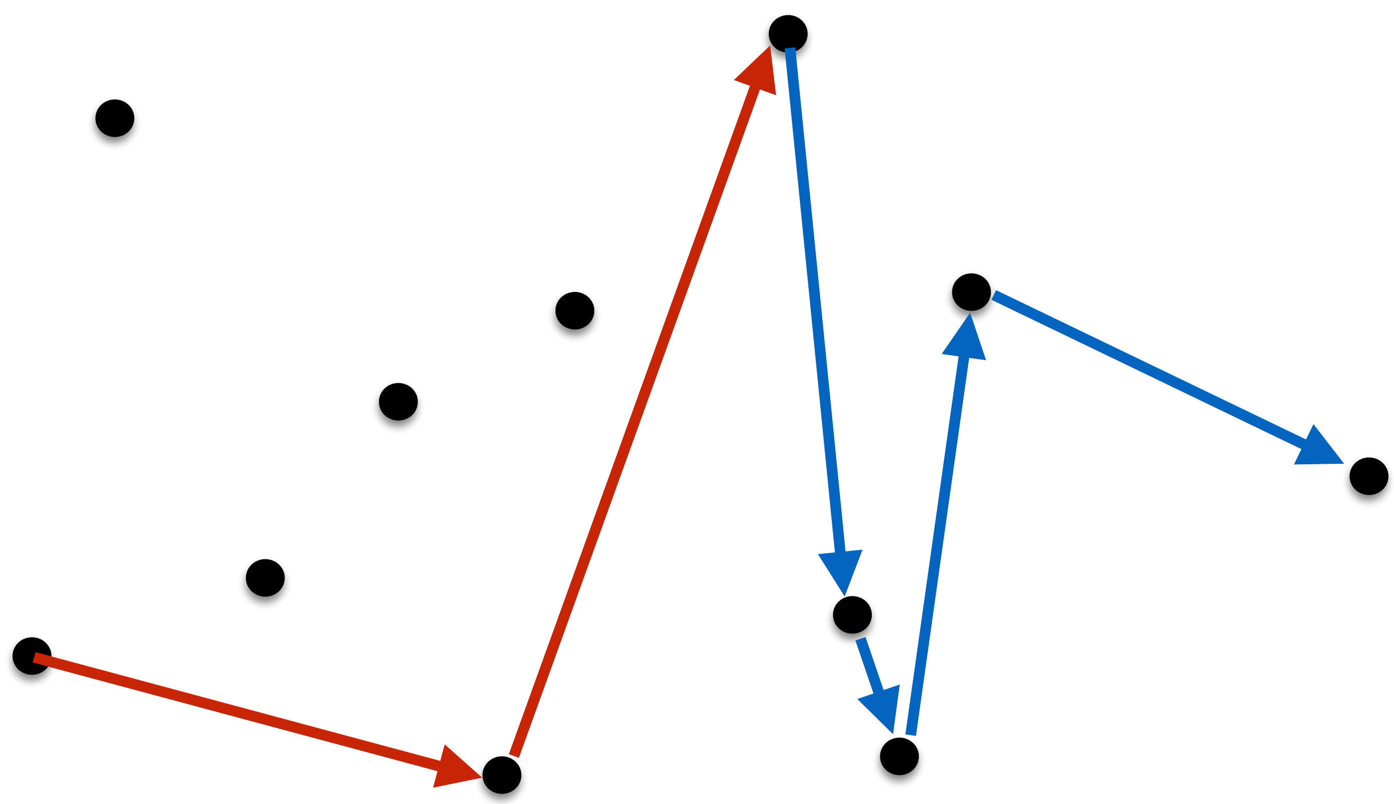
Idee: Konstruiere untere Hülle von „links“ nach „rechts“



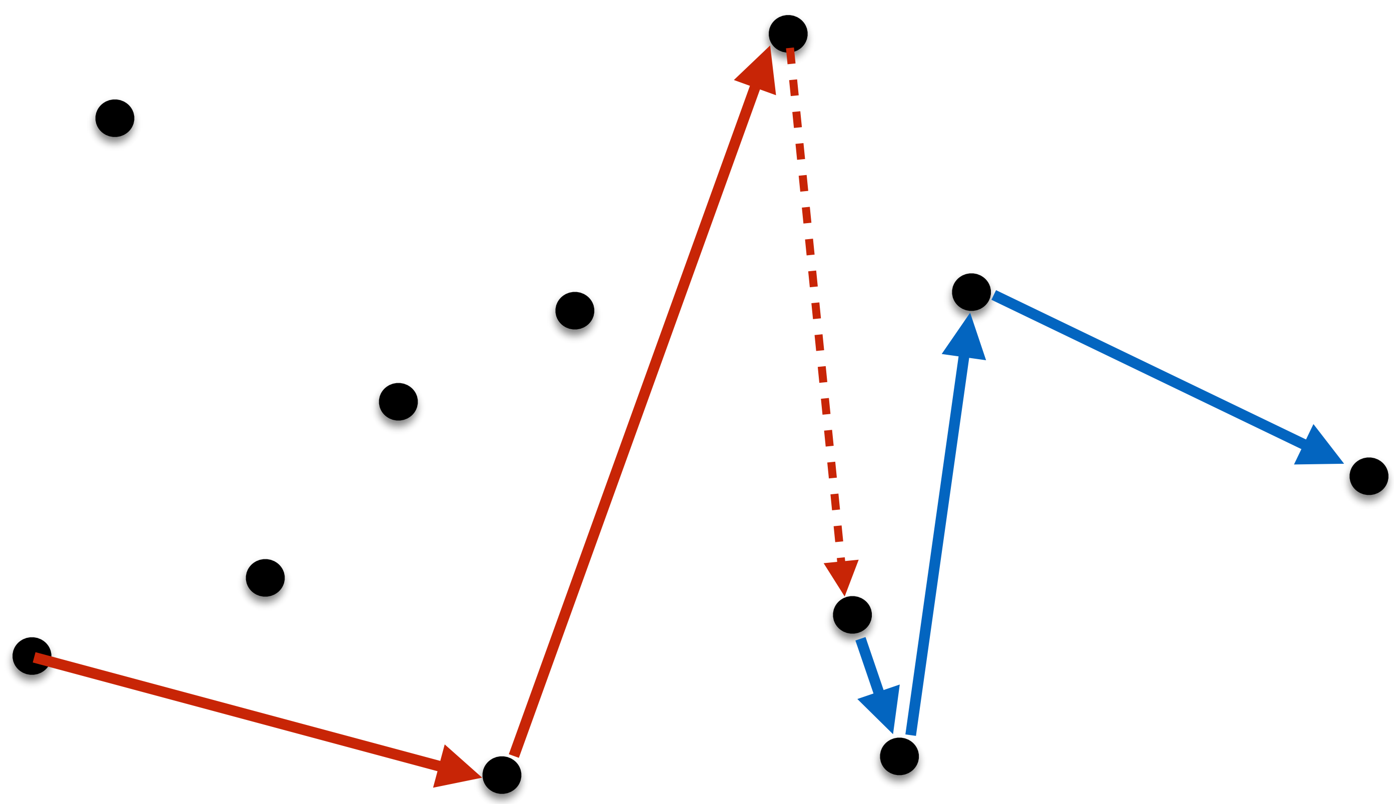
Idee: Konstruiere untere Hülle von „links“ nach „rechts“



Idee: Konstruiere untere Hülle von „links“ nach „rechts“



Idee: Konstruiere untere Hülle von „links“ nach „rechts“



etc.

LocalRepair

LocalRepair$(p_1, p_2, \ldots, p_n)$

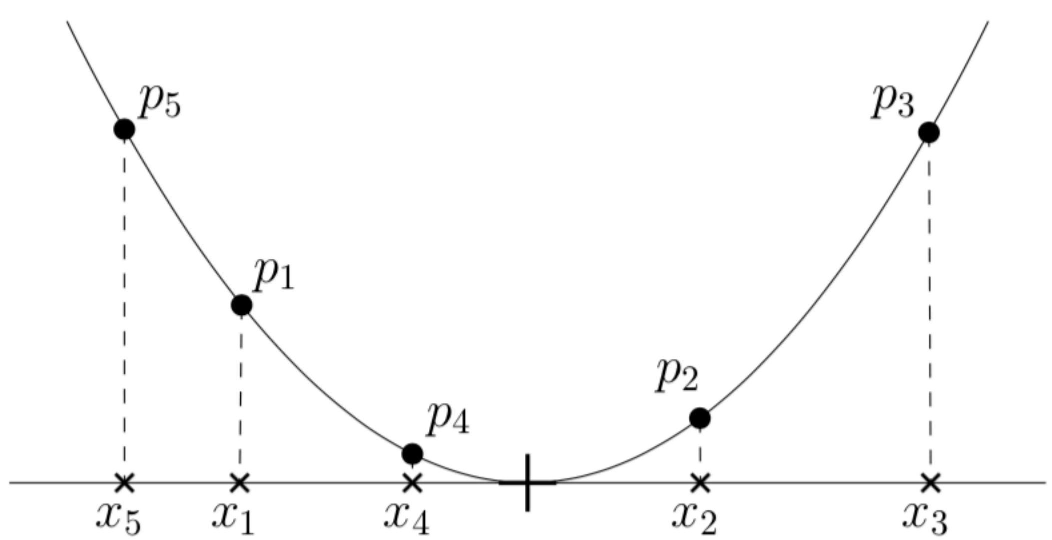
▷ setzt $(p_1, p_2, \ldots, p_n)$, $n \geq 3$, nach x-Koordinate sortiert, voraus

$q_0 \leftarrow p_1$
$h \leftarrow 0$
for $i \leftarrow 2$ **to** n **do** ▷ untere konvexe Hülle, links nach rechts
    while $h > 0$ und q_h links von $q_{h-1}p_i$ **do**
        $h \leftarrow h - 1$
    $h \leftarrow h + 1$
    $q_h \leftarrow p_i$ ▷ $(q_0, \ldots, q_h)$ untere konvexe Hülle von $\{p_1, \ldots, p_i\}$
$h' \leftarrow h$
for $i \leftarrow n - 1$ **downto** 1 **do** ▷ obere konvexe Hülle, rechts nach links
    while $h > h'$ und q_h links von $q_{h-1}p_i$ **do**
        $h \leftarrow h - 1$
    $h \leftarrow h + 1$
    $q_h \leftarrow p_i$
return $(q_0, q_1, \ldots, q_{h-1})$ ▷ Ecken der konvexen Hülle, gg. Uhrzeigersinn

Satz Gegeben eine Folge $p_1, p_2, \ldots, p_n$ nach x-Koordinate sortierter Punkte in allgemeiner Lage in $\mathbb{R}^2$, berechnet der Algorithmus LocalRepair die konvexe Hülle von $\{p_1, p_2, \ldots, p_n\}$ in Zeit $O(n)$.

Achtung: für das Sortieren der Punkte nach x-Koordinate benötigt man Zeit O(n log n)

Man kann zeigen:
Konvexe Hülle kann i.A. nicht schneller als Sortieren gelöst werden.



$$p_i = (x_i, x_i^2),$$