

2.7 Geometrische Algorithmen

2.7.1 Inside-Test

2.7.2 Konvexe Hülle

2.7.2.1 Gift Wrapping

2.7.2.2 Graham's Scan

2.7.2.3 Divide & Conquer

2.7.2.4 Optimaler Algorithmus

2.7.3 Nachbarschaften

2.7.4 Schnittprobleme



2.7 Geometrische Algorithmen

2.7.1 Inside-Test

2.7.2 Konvexe Hülle

2.7.2.1 Gift Wrapping

2.7.2.2 Graham's Scan

2.7.2.3 Divide & Conquer

2.7.2.4 Optimaler Algorithmus

2.7.3 Nachbarschaften

2.7.4 Schnittprobleme

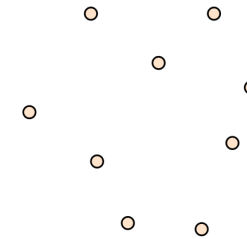


Generierung von Polygonen

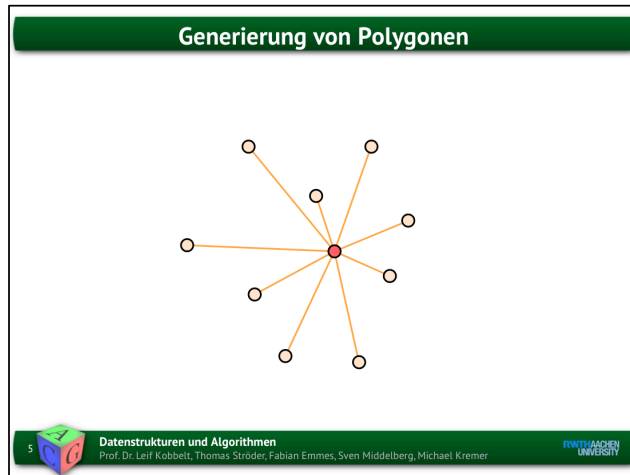
- Gegeben sei eine ungeordnete Menge von Eckpunkten p_i
- Generiere ein einfaches, geschlossenes Polygon (durch "Sortieren" der p_i)



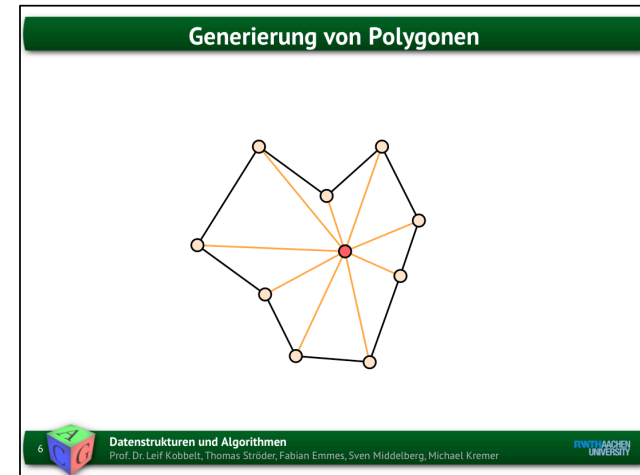
Generierung von Polygonen



Gegeben sei eine Punktwolke.



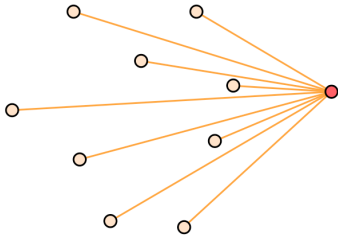
Wir suchen uns willkürlich einen Punkt heraus und verbinden alle anderen Punkte mit diesem Punkt.



Und verbinden die Punkte so, dass keine der (hier orangenen Geraden) geschnitten wird.

Generierung von Polygonen

aber...



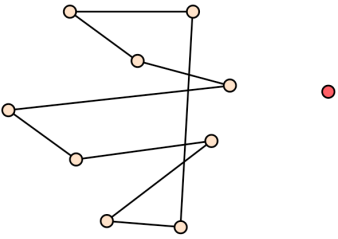
7

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FWTH AACHEN
UNIVERSITY

Generierung von Polygonen

aber...



8

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FWTH AACHEN
UNIVERSITY

Dies führt in manchen Fällen auch zu Problemen.

Konvexität

- Eine Teilmenge S des \mathbb{R}^2 heißt konvex, wenn mit jedem Paar p, q von Punkten aus S auch alle Zwischenpunkte $(1-t)p+ tq$ mit $0 \leq t \leq 1$ in S liegen.
- $(1-t)p+ tq = p-tp+ tq = p+t(q-p)$

$p \circ \xrightarrow{q-p} \circ q$

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Wir parametrisieren eine Gerade zwischen zwei Punkten p und q . Wenn t zwischen 0 und 1 liegt, können wir jeden Punkt auf diesem Geradensegment berechnen.

Konvexität

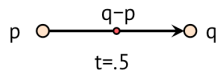
- Eine Teilmenge S des \mathbb{R}^2 heißt konvex, wenn mit jedem Paar p, q von Punkten aus S auch alle Zwischenpunkte $(1-t)p+ tq$ mit $0 \leq t \leq 1$ in S liegen.
- $(1-t)p+ tq = p-tp+ tq = p+t(q-p)$

$p \circ \xrightarrow{q-p} \circ q$
 $t=0$

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

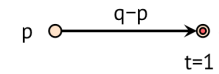
Konvexität

- Eine Teilmenge S des \mathbb{R}^2 heißt konvex, wenn mit jedem Paar p, q von Punkten aus S auch alle Zwischenpunkte $(1-t)p + tq$ mit $0 \leq t \leq 1$ in S liegen.
- $(1-t)p + tq = p - tp + tq = p + t(q-p)$

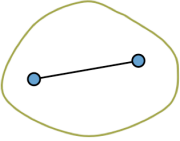


Konvexität

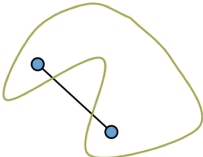
- Eine Teilmenge S des \mathbb{R}^2 heißt konvex, wenn mit jedem Paar p, q von Punkten aus S auch alle Zwischenpunkte $(1-t)p + tq$ mit $0 \leq t \leq 1$ in S liegen.
- $(1-t)p + tq = p - tp + tq = p + t(q-p)$



Konvexität



konvex



nicht konvex

13
Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FWTH AACHEN
UNIVERSITY

Konvex ist ein geometrisches Gebilde, wenn es keine Gerade zwischen zwei in dem Gebilde liegenden Punkten gibt, die den Rand des Gebildes schneidet.

Konvexe Hülle

- **Definition**
Die Konvexe Hülle $CH(S)$ einer Teilmenge S des \mathbb{R}^2 ist die kleinste konvexe Menge (bzgl. \subseteq), die S enthält.
- **Lemma**

$$CH(S) = \bigcap_{\substack{K \supseteq S \\ K \text{ ist konvex}}} K$$

14
Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FWTH AACHEN
UNIVERSITY

Konvexe Hülle

- **Lemma**
 Es sei eine Menge von (Eck-)Punkten p_i gegeben. Die Konvexe Hülle $CH(p_i)$ ist die Menge aller Punkte q mit

$$q = \sum_{i=1}^n \alpha_i p_i \text{ mit } \sum_{i=1}^n \alpha_i = 1 \text{ und } \alpha_i \geq 0$$

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Wenn die Koeffizienten einer Linearkombination alle positiv sind, wird dies auch „Konvexkombination“ genannt.
 Wenn sich die Koeffizienten zu 1 aufsummieren, ist es eine „Affinkombination“.

Konvexe Hülle

- **Beweis**
 Es sei $C = \left\{ \sum_{i=1}^n \alpha_i p_i : \sum_{i=1}^n \alpha_i = 1, \alpha_i \geq 0 \right\}$
 Zu zeigen: $C = CH(p_i)$
- Wegen

$$\sum_{i=1}^n \alpha_i p_i = \alpha_1 p_1 + (1 - \alpha_1) \sum_{i=2}^n \frac{\alpha_i}{1 - \alpha_1} p_i$$
 gilt offensichtlich $C \subseteq CH(p_i)$
- Noch zu zeigen: C ist konvex

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Konvexe Hülle

$$q = \sum_{i=1}^n \alpha_i p_i, \quad \sum_{i=1}^n \alpha_i = 1, \quad \alpha_i \geq 0$$
$$r = \sum_{i=1}^n \beta_i p_i, \quad \sum_{i=1}^n \beta_i = 1, \quad \beta_i \geq 0$$

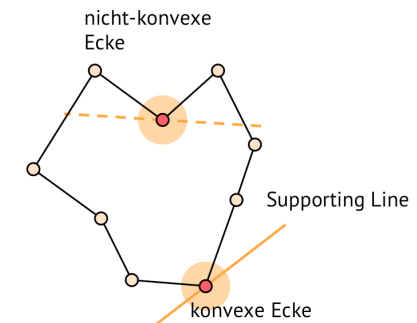
$$(1-t)q + tr = (1-t) \sum_{i=1}^n \alpha_i p_i + t \sum_{i=1}^n \beta_i p_i$$
$$= \sum_{i=1}^n (1-t)\alpha_i p_i + t\beta_i p_i$$
$$= \sum_{i=1}^n ((1-t)\alpha_i + t\beta_i) p_i$$

$$\sum_{i=1}^n ((1-t)\alpha_i + t\beta_i) = (1-t) \sum_{i=1}^n \alpha_i + t \sum_{i=1}^n \beta_i$$
$$= (1-t) + t$$
$$= 1$$

$$(1-t)\alpha_i + t\beta_i \geq 0$$



Lokale Konvexität

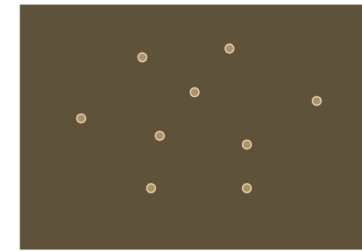


Konvexe Hülle

- Jede Supporting Line definiert einen Halbraum, in dem sich alle Eckpunkte befinden.
- Die konvexe Hülle ergibt sich aus dem Schnitt aller dieser Halbräume (ohne Beweis).

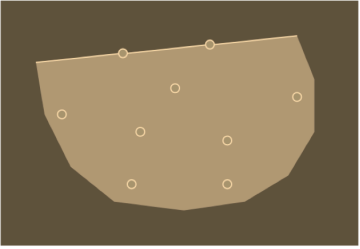


Konvexe Hülle





Wähle „Supporting Lines“ so knapp wie möglich. Sobald diese Gerade zwei Punkte schneiden, sind sie so knapp wie möglich und können nicht mehr verbessert werden.

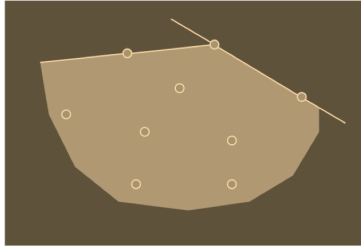
Konvexe Hülle





A diagram illustrating the concept of a convex hull. It shows a set of points (represented by small circles) and a convex polygon (the hull) that encloses all the points. A line segment connects two points on the upper boundary of the hull.

21  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

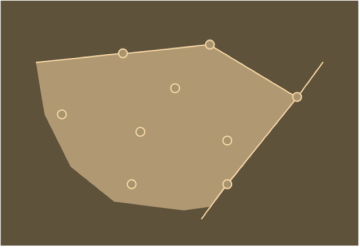
Konvexe Hülle





A diagram illustrating the concept of a convex hull. It shows a set of points (represented by small circles) and a convex polygon (the hull) that encloses all the points. A line segment connects two points on the upper boundary of the hull.

22  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

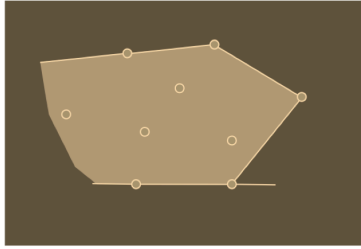
Konvexe Hülle





The diagram shows a set of points on a dark background. A light brown convex hull is formed by connecting the outermost points. A line segment is drawn from the top-right vertex of the hull to the bottom-right vertex, with an arrow pointing to the right, indicating the next step in the algorithm.

25  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

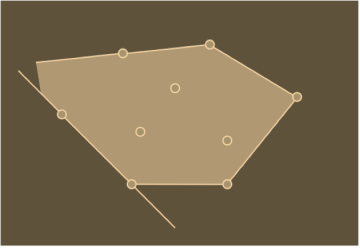
Konvexe Hülle





The diagram shows the same set of points and convex hull as the previous slide. A line segment is drawn from the top-right vertex of the hull to the bottom-left vertex, with an arrow pointing to the left, indicating the next step in the algorithm.

24  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

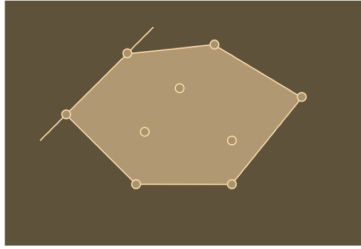
Konvexe Hülle





The diagram shows a set of points on a dark background. A light brown convex polygon is drawn around the points. Two lines extend from the left side of the polygon, one from the top-left vertex and one from the bottom-left vertex, representing the process of rotating a line to find the next vertex of the convex hull.

25  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

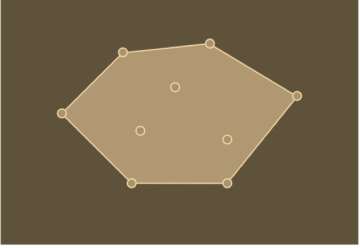
Konvexe Hülle





The diagram shows the same set of points as slide 25. The convex hull is now a more complete polygon. A line is drawn from the top-left vertex to the top-right vertex, and another line is drawn from the top-right vertex to the rightmost vertex, indicating the next steps in the hull construction.

26  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Konvexe Hülle





27  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer  FRIEDRICH-ALEXANDER
UNIVERSITÄT

„Gift Wrapping“ = Man wickelt die Punkte so „platzsparend“ wie möglich ein.

Lokale Konvexität

- **Lemma**
Der Eckpunkt mit der kleinsten x-Koordinate (und ggfs. mit der kleinsten y-Koordinate) ist immer konvex.
- **Beweis:** Supporting Line parallel zur y-Achse

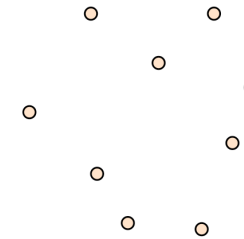
28  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer  FRIEDRICH-ALEXANDER
UNIVERSITÄT

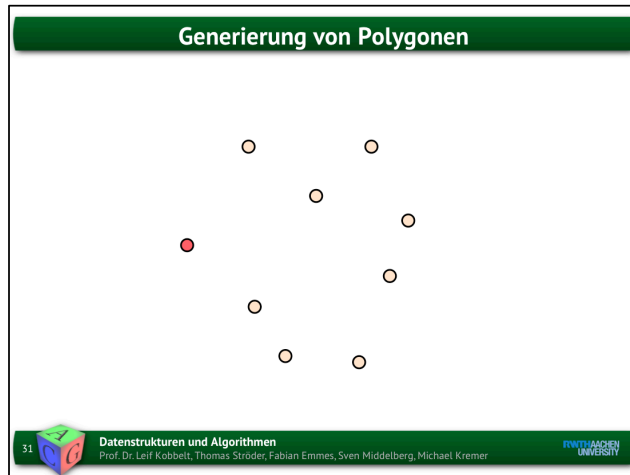
Generierung von Polygonen

- Suche eine konvexe Ecke
- Sortiere die übrigen Ecken nach dem Winkel zur x-Achse
- Verbinde aufeinanderfolgende Ecken

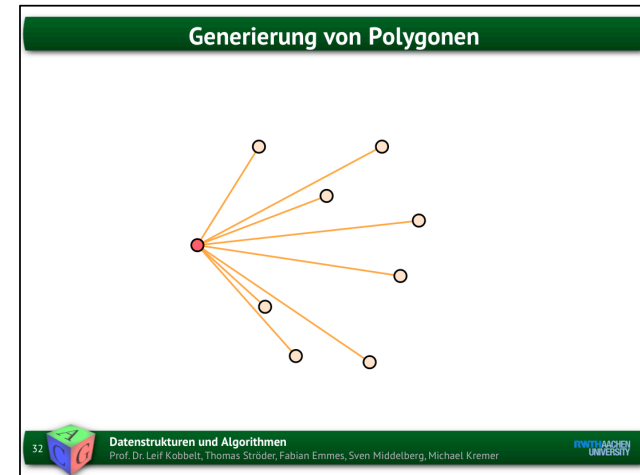


Generierung von Polygonen





Wir nehmen uns wieder einen beliebigen Punkt heraus.



Und verbinden alle anderen Punkte mit ihm. Nun sortiert man alle Verbindungslinien nach Winkel und verbindet sie.

Generierung von Polygonen

33
Datenstrukturen und Algorithmen
FRIEDRICH-SCHILLER-UNIVERSITÄT

Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Dies ist zwar kein schönes Polygon, aber ein gültiges.

Sortierung nach Winkel

- Winkel der Kante von (x_0, y_0) nach (x_1, y_1) zur x-Achse

$$\tan \alpha = \frac{y_1 - y_0}{x_1 - x_0}$$

$$(\sin \alpha)^2 = \frac{(y_1 - y_0)^2}{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$

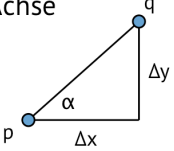
34
Datenstrukturen und Algorithmen
FRIEDRICH-SCHILLER-UNIVERSITÄT

Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Sortierung nach Winkel

- Winkel $\alpha(p,q)$ der Kante von $p=(x,y)$ nach $q=(x+\Delta x, y+\Delta y)$ zur x-Achse

$$\tan \alpha = \frac{\Delta y}{\Delta x}$$



- Nachteile
 - Berechnung des Arcustangens ist teuer.
 - Abfangen von Division durch 0.
 - Prüfung, in welchem Quadrant der Punkt liegt, notwendig.

35
Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRONTMÄCHER
UNIVERSITÄT

Sortierung nach Winkel

- Besser: Wähle einfache Funktion β mit $\beta(p,q_0) < \beta(p,q_1) \Leftrightarrow \alpha(p,q_0) < \alpha(p,q_1)$
- Wähle z.B. im ersten Quadrant

$$\beta(p,q) = \frac{\Delta y}{\Delta x + \Delta y} \in [0,1]$$

- Beta(dx,dy)


```

      beta ← dy / (abs(dx) + abs(dy))
      if dx < 0 then return 2 - beta
      if dy < 0 then return 4 + beta
      return beta
      
```

36
Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRONTMÄCHER
UNIVERSITÄT

Wenn man den Algorithmus implementieren möchte, benutzt man diese Vereinfachung, um die Benutzung des Arcustangens zu vermeiden. Bestimmung des Quadranten durch Fallunterscheidung.

Konvexe Hülle

- Wir nehmen immer an, dass die Punkte in der konvexen Hülle einer Punktmenge so sortiert sind, dass das zugehörige Polygon gegen den Uhrzeigersinn orientiert ist.



Gift Wrapping

- Beginne mit einer Supporting Line.
- Ergänze in jedem Schritt den Eckpunkt, der den geringsten Winkel zur aktuellen Supporting Line aufspannt.





Gift Wrapping

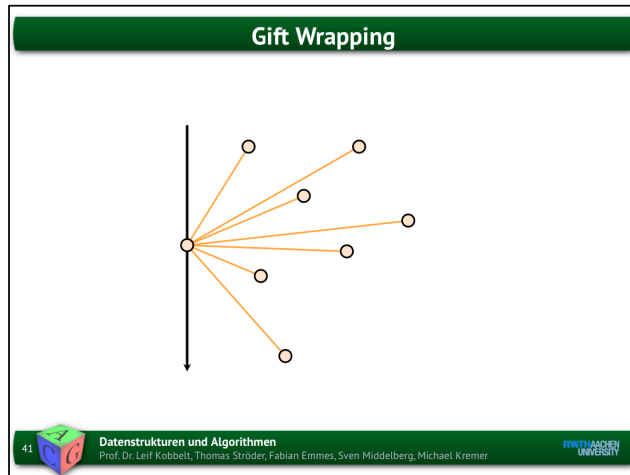
A set of 10 points in a 2D plane. The points are distributed in a roughly circular pattern. The bottom-most point is the lowest y-coordinate point.

39  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

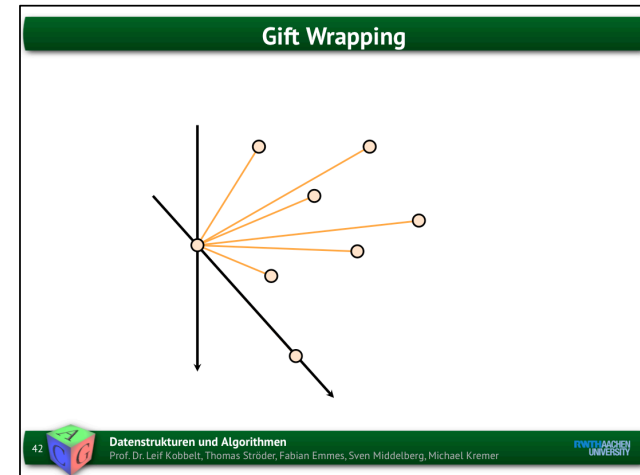
Gift Wrapping

The same set of 10 points as in slide 39. A vertical line segment with a downward-pointing arrow is drawn from the top-most point to the bottom-most point, representing the current step in the gift wrapping process.

40  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 



Vom Startpunkt wissen wir, dass er auf der konvexen Hülle liegt (Lemma). Nun nehmen wir die Verbindungsgerade mit dem geringsten Winkel zur Supporting Line.



Gift Wrapping

43 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Gift Wrapping

44 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Gift Wrapping

45 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Gift Wrapping

46 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Und so weiter...

Gift Wrapping

- GiftWrap(pts[1..n])
 - p ← LeftMostPoint(pts)
 - q ← Null
 - CH.create()
 - CH.add(p)
 - L ← negative y-axis
 - while q ≠ CH.top() do
 - q ← select point with smallest angle
 - (p,L)
 - CH.add(q)
 - L ← Line(p,q)
 - p ← q

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Min/Max aus einer Folge kann in $O(n)$ bestimmt werden.

Gift Wrapping

- Die Gift Wrapping Prozedur entspricht im Wesentlichen dem Selection Sort.
- Aufwand $O(m \times n)$, wobei m die Zahl der Eckpunkte in der konvexen Hülle ist
- Best case: $m = 3$
- Worst case: $m = n$

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

$3 \leq m \leq n$. Best-Case: Dreieck. Worst-Case: Alle Punkte liegen auf der konvexen Hülle (z.B. Kreis).

2.7 Geometrische Algorithmen

- 2.7.1 Inside-Test
- 2.7.2 Konvexe Hülle
 - 2.7.2.1 Gift Wrapping
 - 2.7.2.2 **Graham's Scan**
 - 2.7.2.3 Divide & Conquer
 - 2.7.2.4 Optimaler Algorithmus
- 2.7.3 Nachbarschaften
- 2.7.4 Schnittprobleme



Es gibt aber noch schlauere Algorithmen für diese Problemstellung. Z.B. Graham's Scan.

Graham's Scan

- Gift-Wrapping führt zu viele redundante Berechnungen durch.
- Bei der einfachen Generierung von Polygonen genügt ein einziger globaler Sortierungsschritt.
- Idee: Entferne die nicht-konvexen Ecken aus dem einfachen Polygon.



Idee: Erzeuge irgendein Polygon und entferne lokale Nichtkonvexitäten.

Graham's Scan

- GrahamScan(pts)
 - let p be the left-most point in pts
 - sort pts by angle of line (p,pts[i]) to x-axis
 - p ← head(pts); q ← next(p); r ← next(q)
 - while r ≠ p do
 - if p,q,r is convex do
 - p ← next(p); q ← next(q); r ← next(r)
 - else
 - delete q
 - q ← p; p ← prev(p)

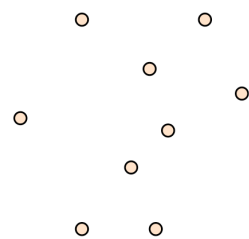
51

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

 RWTH AACHEN
UNIVERSITY

Sortieren kann bekanntermaßen in $O(n \log(n))$ bewerkstelligt werden.

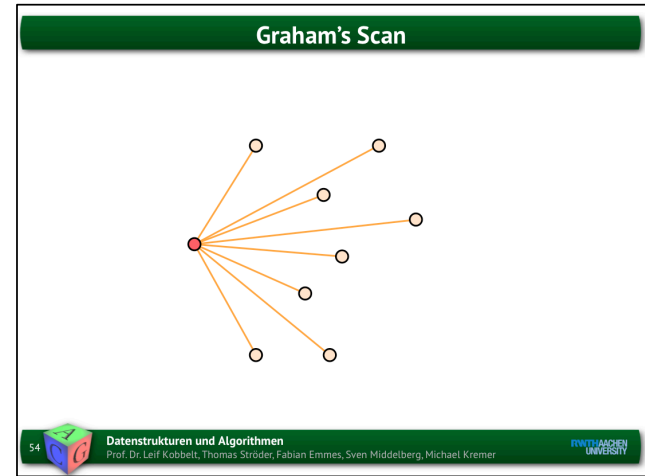
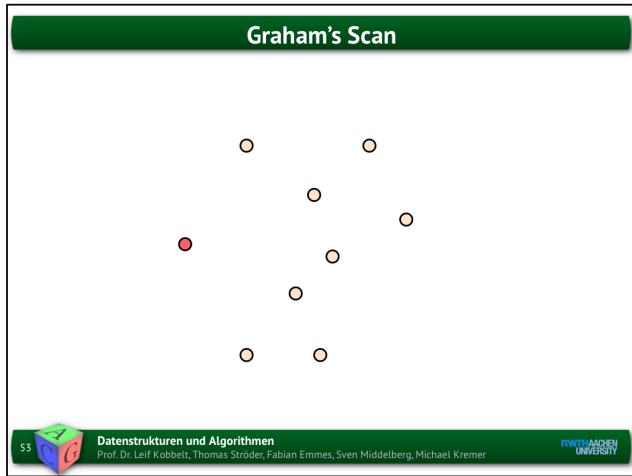
Graham's Scan



52

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

 RWTH AACHEN
UNIVERSITY



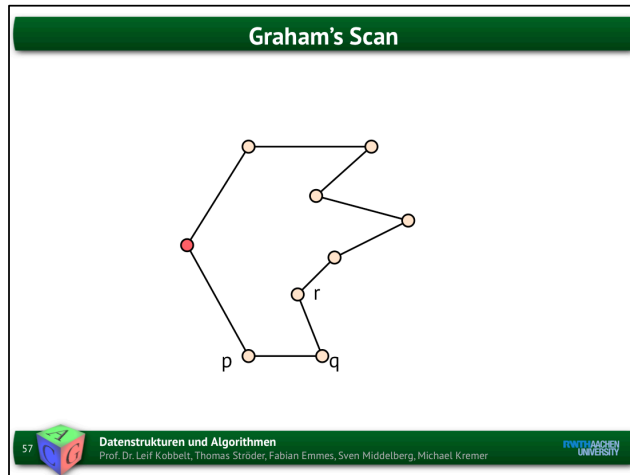
Graham's Scan

55
Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FWTH AACHEN
UNIVERSITY

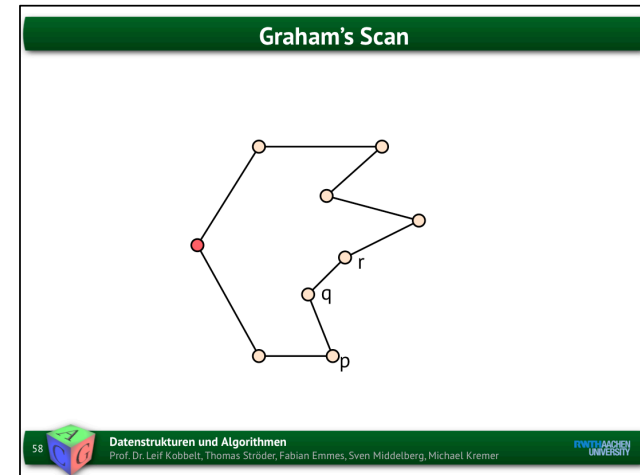
Graham's Scan

56
Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FWTH AACHEN
UNIVERSITY

Wir „biegen links“ ab => Konvex



Wir biegen wieder links ab => Konvex



Wir biegen rechts ab => Nicht-konvex, schmeiße Punkt raus.

Graham's Scan

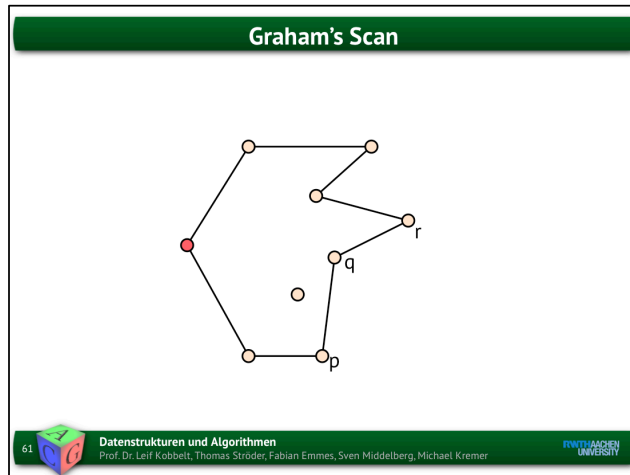
Diagram illustrating the first step of Graham's Scan. A set of 8 points is shown. One point on the left is highlighted in red. The other points are connected by lines to form a partial convex hull. Points p and q are labeled at the bottom, and r is labeled on the right side.

59 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
RWTH AACHEN UNIVERSITY

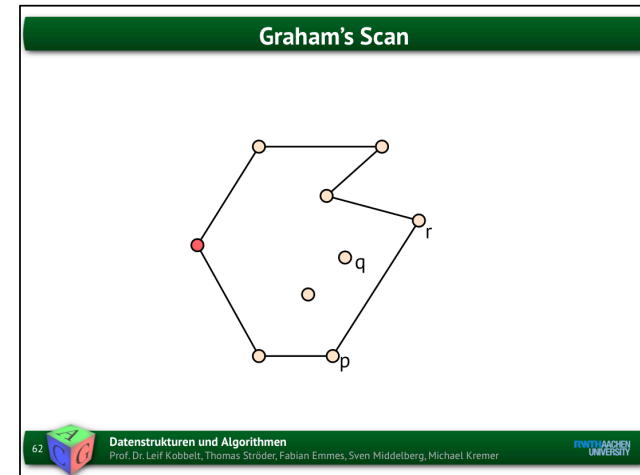
Graham's Scan

Diagram illustrating the second step of Graham's Scan. The same set of 8 points is shown. The partial convex hull is now more complete, with point q now connected to the hull. Point r is still on the right side.

60 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
RWTH AACHEN UNIVERSITY



Wieder rechts abbiegen => Nicht-konvex, schmeiße Punkt raus.



Und so weiter...

Graham's Scan

The diagram shows a set of points in a 2D plane. A point 'p' is marked in red at the bottom-left. A point 'q' is at the bottom-right. A line segment connects 'p' and 'q'. Other points are scattered above and to the right of this segment. One point 'r' is specifically labeled on the right side of the diagram.

63 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Graham's Scan

The diagram shows the same set of points as the previous slide. The line segment 'pq' is now extended upwards and to the right. Point 'r' is now to the left of this extended line, indicating that the angle at 'q' is reflex. Point 'q' is now labeled at the top-right position.

64 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Graham's Scan

Diagram illustrating the first step of Graham's Scan: A set of points is shown. The leftmost point is red. The top-right point is labeled 'r'. The rightmost point is labeled 'p'. A point 'q' is located between 'r' and 'p'. Lines connect 'r' to 'q' and 'q' to 'p'. The other points are scattered below and to the left of the line segment 'rp'.

65

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Graham's Scan

Diagram illustrating the second step of Graham's Scan: The same set of points is shown. The leftmost point is red. The top-right point is labeled 'r'. The rightmost point is labeled 'p'. Point 'q' is now inside the triangle formed by 'r' and 'p'. The other points are scattered below and to the left of the line segment 'rp'.

66

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Graham's Scan

67

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

RWTH AACHEN
UNIVERSITY

Graham's Scan

68

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

RWTH AACHEN
UNIVERSITY

Graham's Scan

p q
 r

69

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

 RWTH AACHEN
 UNIVERSITY

Halbraum-Test

- Es sei eine Supporting-Line durch die Punkte (x_0, y_0) und (x_1, y_1) gegeben
- Die Reihenfolge der Punkte definiert eine Orientierung
- Links ist innen, d.h. der Vektor $(y_0 - y_1, x_1 - x_0)$ zeigt senkrecht nach innen

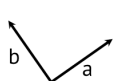
70

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

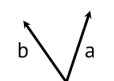
 RWTH AACHEN
 UNIVERSITY

Skalarprodukt

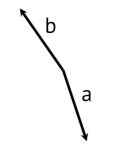
- Erinnerung: Skalarprodukt
 $(x_0, y_0) \cdot (x_1, y_1) = x_0 \cdot x_1 + y_0 \cdot y_1$
- Anschaulich:



$a \cdot b = 0$
($\Leftrightarrow a \perp b$)



$a \cdot b > 0$



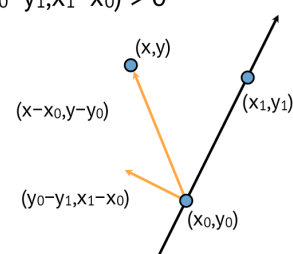
$a \cdot b < 0$

71
Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Das Skalarprodukt gibt Auskunft über den Winkel zwischen zwei Vektoren.

Halbraum-Test

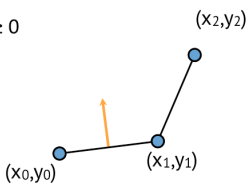
- Punkt (x, y) im Innern:
 $(x - x_0, y - y_0) \cdot (y_0 - y_1, x_1 - x_0) \geq 0$



72
Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Halbraum-Test

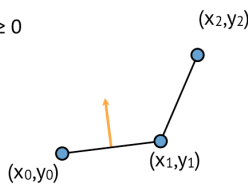
- Punkt (x_2, y_2) im Innern:
 $(x_2 - x_0, y_2 - y_0) \cdot (y_0 - y_1, x_1 - x_0) \geq 0$
 $(x_2 - x_0) \times (y_0 - y_1) + (y_2 - y_0) \times (x_1 - x_0) \geq 0$

$$\det \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix} \geq 0$$


73
Datenstrukturen und Algorithmen
FRIEDRICH-SCHILLER
UNIVERSITÄT

Halbraum-Test

- Punkt (x_2, y_2) im Innern:
 $(x_2 - x_0, y_2 - y_0) \cdot (y_0 - y_1, x_1 - x_0) \geq 0$
 $(x_2 - x_0) \times (y_0 - y_1) + (y_2 - y_0) \times (x_1 - x_0) \geq 0$

$$\det \begin{bmatrix} x_1 - x_0 & x_2 - x_1 \\ y_1 - y_0 & y_2 - y_1 \end{bmatrix} \geq 0$$


74
Datenstrukturen und Algorithmen
FRIEDRICH-SCHILLER
UNIVERSITÄT

Das Kreuzprodukt ist dasselbe wie die Determinante der o.g. Matrix.

Analyse

- GrahamScan(pts)
 - let p be the left-most point in pts
 - sort pts by angle of line (p,pts[i]) to x-axis
 - p ← head(pts); q ← next(p); r ← next(q)
 - while r ≠ p do
 - if p,q,r is convex do
 - p ← next(p); q ← next(q); r ← next(r)
 - else
 - delete q
 - q ← p; p ← prev(p)

} O(n)
} O(n log n)

} O(2n) =
O(n)



Analyse

- Grahams Scan Algorithmus benötigt zur Berechnung der konvexen Hülle von n Punkten
 $O(n \times \log n)$



2.7 Geometrische Algorithmen

- 2.7.1 Inside-Test
- 2.7.2 Konvexe Hülle
 - 2.7.2.1 Gift Wrapping
 - 2.7.2.2 Graham's Scan
 - 2.7.2.3 Divide & Conquer
 - 2.7.2.4 Optimaler Algorithmus
- 2.7.3 Nachbarschaften
- 2.7.4 Schnittprobleme



Divide & Conquer

- Die konvexe Hülle von drei Punkten (in allg. Lage) ist ein Dreieck.
- **Divide:**
Teile die Punkte in zwei Teilmengen mit disjunkter konvexer Hülle.
- **Conquer:**
Vereinige die konvexen Hüllen der Teilmengen.



Divide & Conquer

- ConvexHull(pts[1..n])
sort pts by increasing x-coordinate
if $x \geq 3$ pts have same x-coordinate then
remove interior points
ConvexHullRec(pts[1..n])



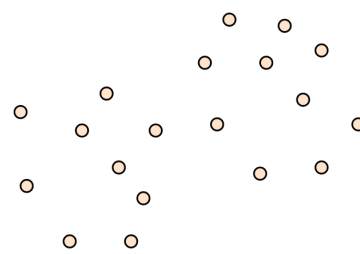
Divide & Conquer



- ConvexHullRec(pts[1..n])
 $p \leftarrow \text{pts}[1..n/2]$
 $q \leftarrow \text{pts}[n/2+1..n]$
 $[p_1..p_m] \leftarrow \text{ConvexHullRec}(p)$
 $[q_1..q_n] \leftarrow \text{ConvexHullRec}(q)$
find upper supporting line (p_i, q_j)
find lower supporting line (p_k, q_l)
return $[p_i, \dots, p_k, q_l, \dots, q_j]$



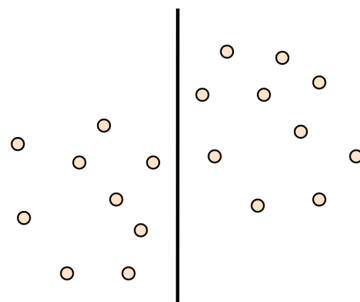
Man teilt die Menge der Punkte sukzessive in zwei Hälften.



Divide & Conquer



81  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Divide & Conquer



82  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Divide & Conquer

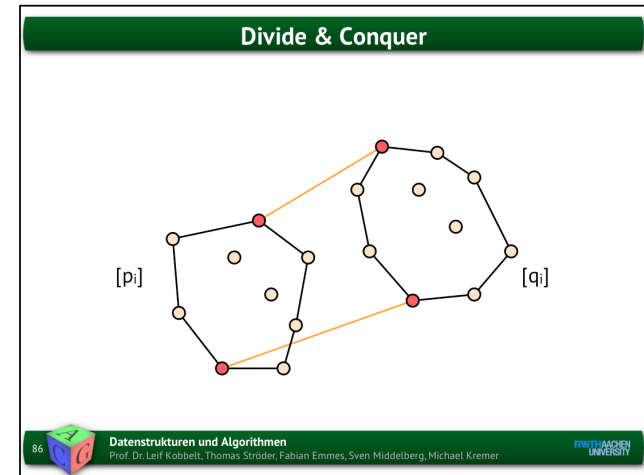
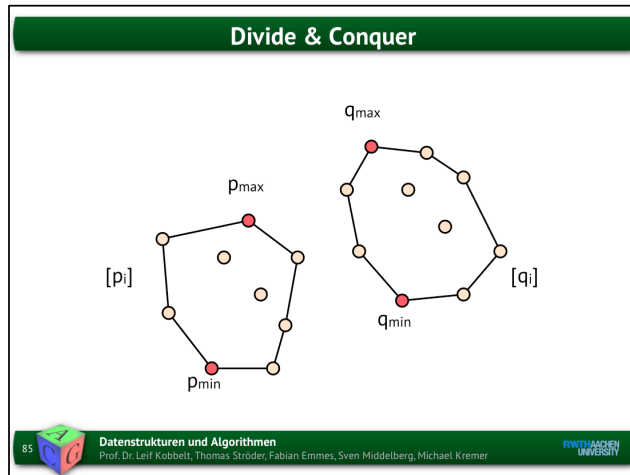
[p_i] [q_i]

83 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Divide & Conquer

[p_i] [q_i]

84 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer



Diese Lösung ist zwar noch nicht optimal, kann aber in wenigen Schritten „repariert“ werden, weil die Initiallösung schon recht gut ist.

Divide & Conquer

87 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
 RWTH AACHEN UNIVERSITY

Divide & Conquer

88 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
 RWTH AACHEN UNIVERSITY

Divide & Conquer

89 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
 RWTH AACHEN UNIVERSITY

Divide & Conquer

90 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
 RWTH AACHEN UNIVERSITY

Divide & Conquer

[p_i] [q_j]

p_i q_j
p_k q_t

91 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Divide & Conquer

[p_i, ..., p_k, q_j, ..., q_t]

p_i q_j
p_k q_t

92 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Supporting Lines

- Bestimme Punkte mit maximaler und minimaler y-Koordinate.
- Verschiebe die Supporting Line solange die entsprechende Ecke nicht konvex ist oder der Nachbarpunkt ausserhalb liegt.



Analyse

- ConvexHull(pts[1..n])
 sort pts by increasing x-coordinate } $O(n \log n)$
 if $x \geq 3$ pts have same x-coordinate then } $O(n)$
 remove interior points
 ConvexHullRec(pts[1..n])



Analyse

- ConvexHullRec(pts[1..n])
 $p \leftarrow \text{pts}[1..n/2]$
 $q \leftarrow \text{pts}[n/2+1..n]$
 $[p_1..p_m] \leftarrow \text{ConvexHullRec}(p)$
 $[q_1..q_n] \leftarrow \text{ConvexHullRec}(q)$
 find upper supporting line (p_i, q_j)
 find lower supporting line (p_k, q_l) $\left. \vphantom{\begin{array}{l} \text{find upper supporting line } (p_i, q_j) \\ \text{find lower supporting line } (p_k, q_l) \end{array}} \right\} O(n)$
 return $[p_i, \dots, p_k, q_l, \dots, q_j]$



Analyse

- $T(n) = 2 T(n/2) + O(n)$
- Master-Theorem: $T(n) = O(n \times \log n)$
- Vorverarbeitung: $O(n \times \log n)$
- Gesamtaufwand: $O(n \times \log n)$



2.7 Geometrische Algorithmen

- 2.7.1 Inside-Test
- 2.7.2 Konvexe Hülle
 - 2.7.2.1 Gift Wrapping
 - 2.7.2.2 Graham's Scan
 - 2.7.2.3 Divide & Conquer
 - 2.7.2.4 **Optimaler Algorithmus**
- 2.7.3 Nachbarschaften
- 2.7.4 Schnittprobleme



Optimaler Algorithmus

- Erinnerung: Ein optimaler Sortieralgorithmus hat beweisbar Aufwand $O(n \log n)$
- Alle bisher vorgestellten Algorithmen zur Berechnung der konvexen Hülle benutzen einen Sortierschritt. Sie haben daher ebenfalls **mindestens** $O(n \log n)$.
- Gibt es einen besseren Algorithmus?



Optimaler Algorithmus

- Sei A ein optimaler CH-Algorithmus
- Verwende ihn zum Sortieren ...
- Eingabe: $[v_1 \dots v_n]$, oBdA $0 \leq v_i < 2\pi$
- Konvertierung: $v_i \rightarrow p_i = (\cos v_i, \sin v_i)$
- CH-Polygon ergibt Reihenfolge

99

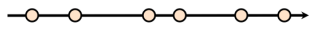
Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

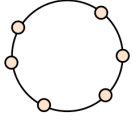
Die Konvertierung bewirkt, dass alle Sample-Punkte auf dem Einheitskreis verteilt liegen.

Optimaler Algorithmus

$[v_1 \dots v_n]$



$[p_1 \dots p_n]$

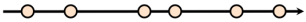


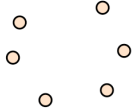
100



Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

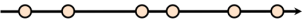
Optimaler Algorithmus

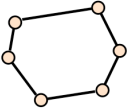
$[v_1 \dots v_n]$ 



$[p_1 \dots p_n]$ 

101  **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

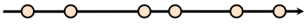
Optimaler Algorithmus

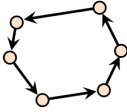
$[v_1 \dots v_n]$ 



$[p_1 \dots p_n]$ 

102  **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Optimaler Algorithmus

$[v_1 \dots v_n]$ 


$[p_1 \dots p_n]$ 

103  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Das Polygon gibt sortierte Folge zurück. Deswegen muss das Problem des Berechnens der konvexen Hülle gleich komplex sein wie das Sortieren.

Optimaler Algorithmus

- Angenommen A ist ein CH-Algorithmus, der schneller als $O(n \log n)$ ist.
- Die Konvertierung $v_i \rightarrow p_i$ liefert einen Sortieralgorithmus, der schneller als $O(n \log n)$ ist.
- Dies ist ein Widerspruch zur Lower Bound von Sortieralgorithmen.

104  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 