



## Datenstrukturen, Algorithmen und Programmierung 2 (DAP2)

## Teile & Herrsche

### *Teile & Herrsche (Divide & Conquer)*

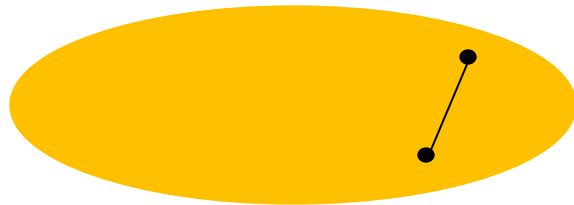
- Teile Eingabe in mehrere Teile auf
- Löse das Problem rekursiv auf den Teilen
- Füge die Teillösungen zu einer Gesamtlösung zusammen

## Teile & Herrsche

### Definition (konvex)

- Eine Menge  $M \subseteq \mathbb{R}^2$  heißt konvex, wenn für alle Punkte  $p, q \in M$  gilt, dass jeder Punkte auf der Strecke  $pq$  ebenfalls in  $M$  ist.
- Formal: Sind  $p, q \in M$ , dann ist für jedes  $\lambda$  mit  $0 \leq \lambda \leq 1$  der Punkt  $r = \lambda p + (1 - \lambda) \cdot q$  ebenfalls in  $M$ .

### Beispiel



Die Menge ist konvex



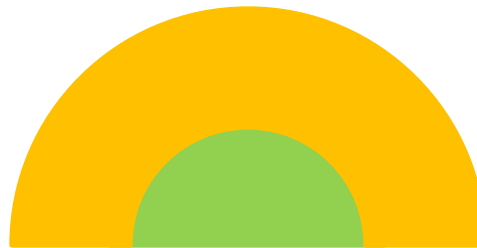
Die Menge ist nicht konvex

## Teile & Herrsche

### *Definition (konvexe Hülle)*

- Die konvexe Hülle einer Menge  $M \subseteq \mathbb{R}^2$  ist der Schnitt aller konvexen Mengen, die  $M$  enthalten.
- Intuitiv: Konvexe Hülle ist die kleinste konvexe Menge, die  $M$  enthält

### *Beispiel*

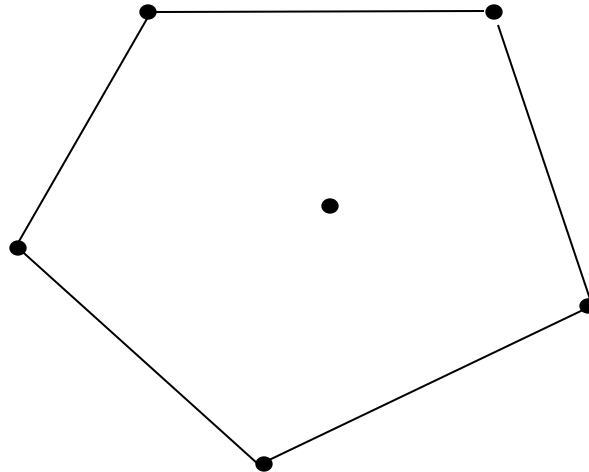


Die konvexe Hülle ist die Vereinigung  
Der orangenen und der grünen Menge

## Teile & Herrsche

### *Konvexe Hülle einer Punktmenge*

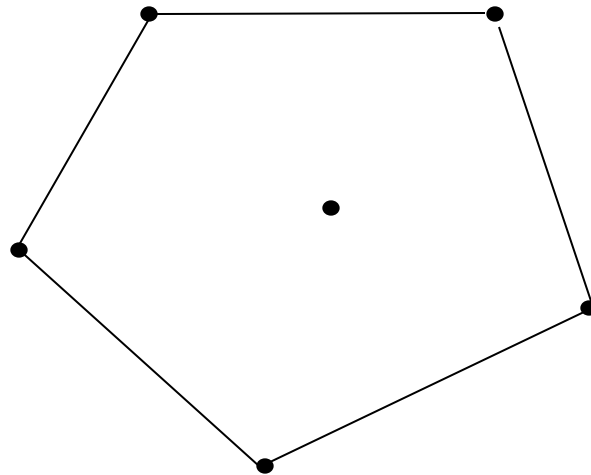
- Intuition: Punkte sind Nägel und die Hülle wird durch Gummiband um die Nägel eingeschlossen



## Teile & Herrsche

### *Problem: Berechnung der konvexen Hülle einer Punktmenge*

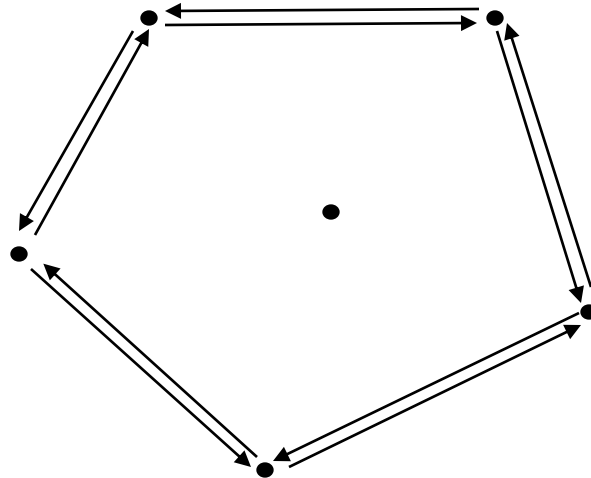
- Eingabe: Menge P von n Punkten in der Ebene  $\mathbb{R}^2$
- Ausgabe: Beschreibung der konvexen Hülle der Punktmenge



## Teile & Herrsche

### *Darstellung der konvexen Hülle im Rechner*

- Wir speichern den Rand der Hülle als doppelt verkettete Liste



## Teile & Herrsche

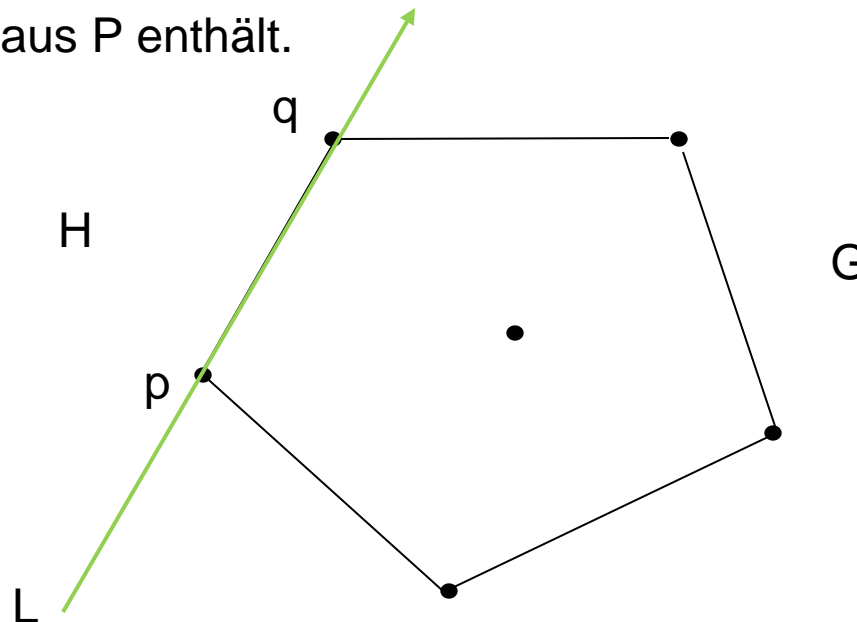
### *Allgemeine Lage*

- Zur Vereinfachung nehmen wir an, dass keine 3 Punkte auf einer Linie liegen und dass keine 2 Punkte dieselbe x-Koordinate haben.

## Teile & Herrsche

### Beobachtung

- Sei  $P$  eine Punktmenge in der Ebene. Eine Strecke  $pq$  mit  $p, q \in P$  liegt auf dem Rand der konvexen Hülle, genau dann wenn die gerichtete Linie  $L$  durch  $p$  und  $q$  die Ebene in die (offenen) Halbebenen  $H$  und  $G$  partitioniert, so dass der Halbraum  $H$  links von  $L$  keinen Punkt aus  $P$  enthält und  $G \cup L$  alle Punkte aus  $P$  enthält.



## Teile & Herrsche

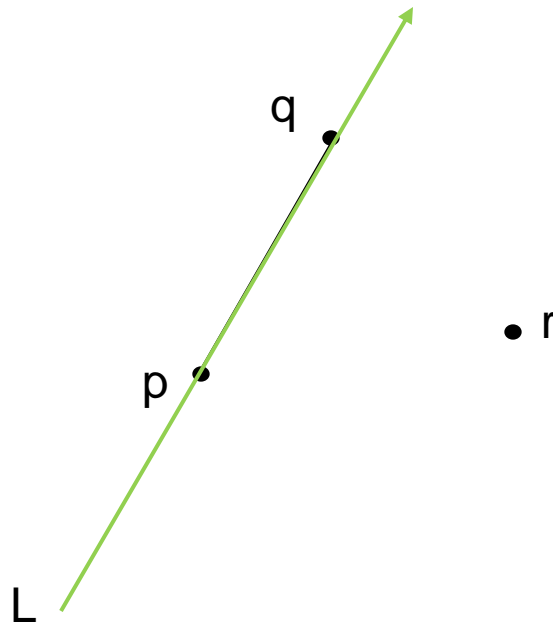
### SimpleConvexHull(P)

1. **for each**  $(p,q) \in P \times P$  **do**
2.     valid  $\leftarrow$  true
3.     **for all**  $r \in P - \{p,q\}$  **do**
4.         **if** r liegt links von der gerichteten Linie durch p und q **then** valid  $\leftarrow$  false
5.     **if** valid = true **then** füge die gerichtete Kante pq zu E hinzu
6. Aus der Menge E konstruiere eine doppelt verkettete Liste der Eckknoten der konvexen Hülle

## Teile & Herrsche

### *Geometrische Primitive*

- Grundlegende geometrische Funktionen, die von einer konstanten Anzahl von Objekten abhängen, können in konstanter Zeit berechnet werden
- Z.B.: Liegt r links von der gerichteten Linie durch p und q



## Teile & Herrsche

### *Schritt 6 des Algorithmus*

- Entferne eine beliebige Kante  $pq$  aus  $E$
- Wähle  $q$  als ersten Knoten der Liste
- Es muss eine gerichtete Kante geben, die von  $q$  ausgeht
- Diese führt zum nächsten Knoten  $r$
- Füge  $r$  in die Liste als Nachfolger von  $q$  ein
- Auf diese Weise können wir Schritt für Schritt den Rand der Hülle als (doppelt verkettete) Liste konstruieren

## Teile & Herrsche

### SimpleConvexHull(P)

1. **for each**  $(p,q) \in P \times P$  **do**
2.     valid  $\leftarrow$  true
3.     **for all**  $r \in P - \{p,q\}$  **do**
4.         **if** r liegt links von der gerichteten Linie durch p und q **then** valid  $\leftarrow$  false
5.     **if** valid = true **then** füge die gerichtete Kante pq zu E hinzu
6. Aus der Menge E konstruiere eine doppelt verkettete Liste der Eckknoten der konvexen Hülle

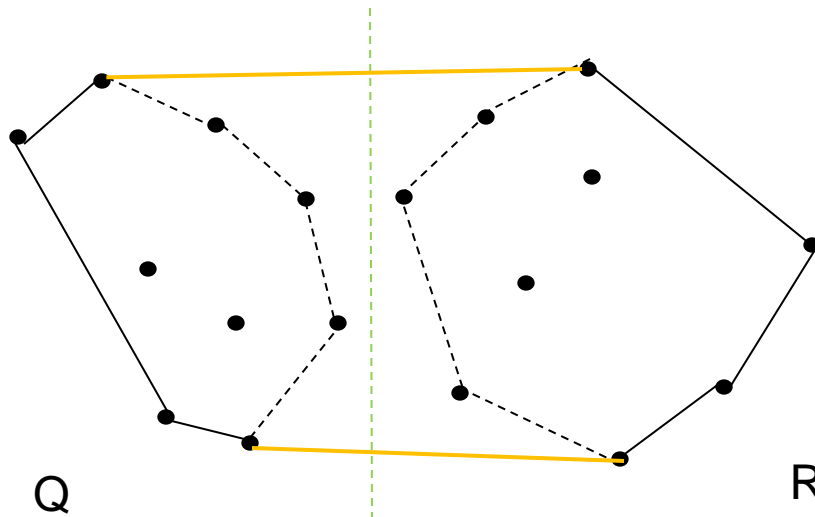
### *Laufzeit des Algorithm*

- $O(n^3)$

## Teile & Herrsche

### Grundidee (siehe Vollversion)

- Sortiere Punkte nach x-Koordinate
- Teile in zwei Hälften Q und R
- Berechne Hüllen der linken und rechten Punktmenge rekursiv
- Setze die Hüllen zusammen



## Teile & Herrsche

### *Notwendige Eigenschaften der oberen fehlenden Strecke*

- Sortierung im Uhrzeigersinn um B ist A, E, C
- Sortierung im Uhrzeigersinn um E ist D, B, F
- Winkel ABE ist größer als 180 Grad
- Winkel BEF ist größer als 180 Grad
- Winkel im Uhrzeigersinn



## Teile & Herrsche

### *Notwendige Eigenschaften der oberen fehlenden Strecke*

- Sortierung im Uhrzeigersinn um B ist A, E, C

### *Beweis*

- Die einzige andere Sortierung ist A, C, E
- Dann liegen A und C auf unterschiedlichen Seiten der Linie durch B und E
- Somit kann BE nicht zum Rand der konvexen Hülle gehören



## Teile & Herrsche

### *Notwendige Eigenschaften der oberen fehlenden Strecke*

- Sortierung im Uhrzeigersinn um B ist A, E, C
- Sortierung im Uhrzeigersinn um E ist D, B, F

### *Beweis*

- Analog

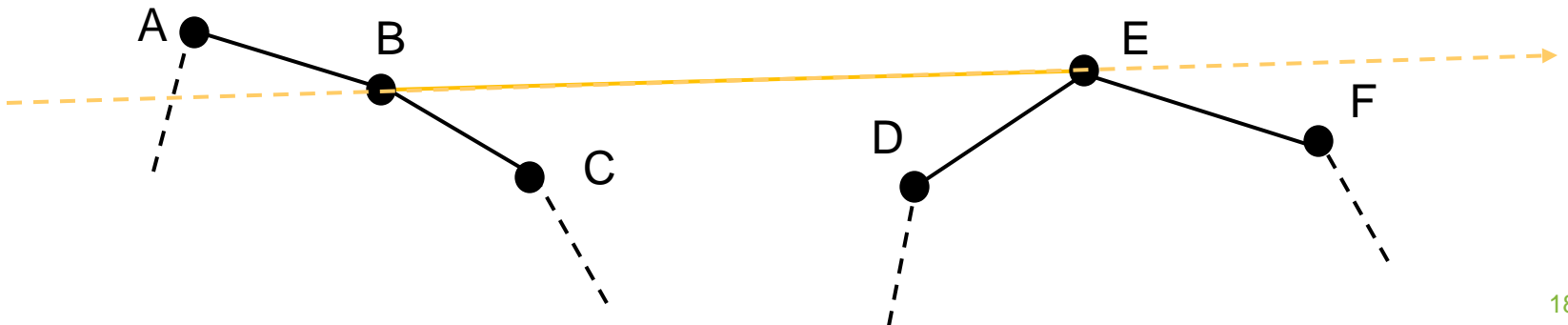
## Teile & Herrsche

### *Notwendige Eigenschaften der oberen fehlenden Strecke*

- Winkel ABE ist größer als 180 Grad

### *Beweis*

- Ist der Winkel nicht größer als 180 Grad, so ist er aufgrund der allgemeinen Lage kleiner als 180 Grad
- Dann liegt A aber links der gerichteten Linie durch B und E und somit liegt BE nicht auf dem Rand der konvexen Hülle



## Teile & Herrsche

### *Notwendige Eigenschaften der oberen fehlenden Strecke*

- Winkel ABE ist größer als 180 Grad
- Winkel BEF ist größer als 180 Grad

### *Beweis*

- Analog

## Teile & Herrsche

### *Hinreichende Eigenschaften der oberen fehlenden Strecke*

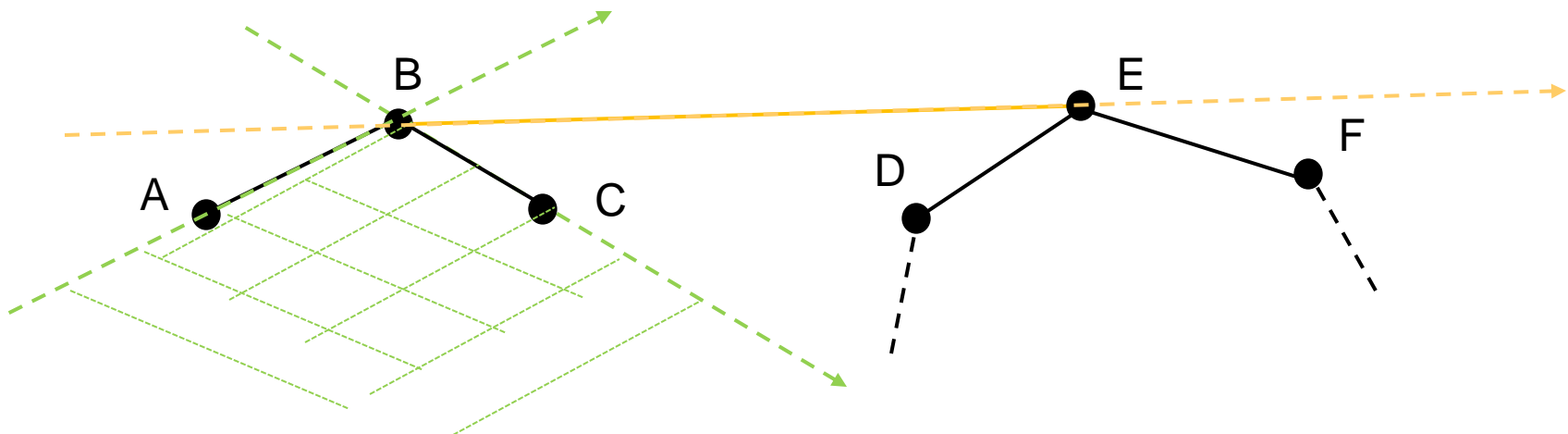
- Sortierung im Uhrzeigersinn um B ist A, E, C
- Sortierung im Uhrzeigersinn um E ist D, B, F
- Winkel ABE ist größer als 180 Grad
- Winkel BEF ist größer als 180 Grad



## Teile & Herrsche

### *Beweis (siehe Vollversion)*

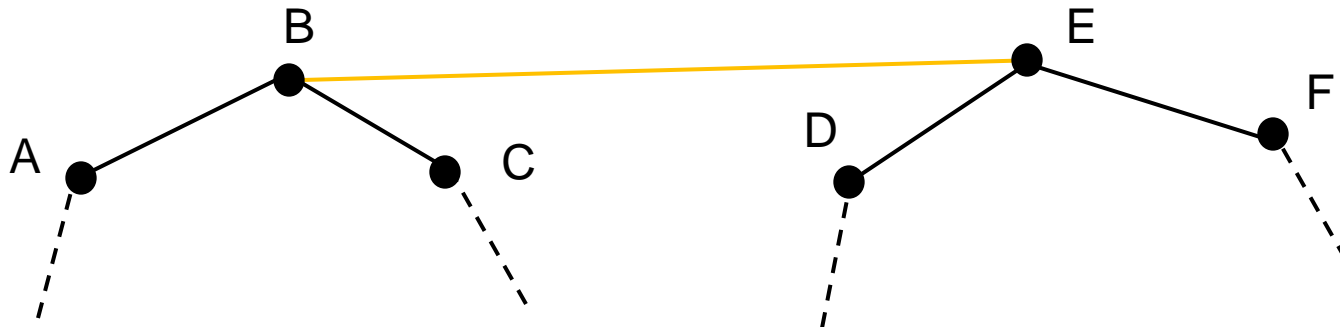
- Da AB auf dem Rand der konvexen Hülle von Q liegt, liegen alle Punkte von Q links der Linie durch A und B. Gleiches gilt für die Punkte B und C.
- Damit liegen alle Punkte aus Q rechts der gerichteten Linie durch B und E
- Analog zeigt man für R, dass alle Punkte rechts der Linie durch B und E liegen
- Damit liegt BE auf dem Rand der konvexen Hülle



## Teile & Herrsche

### *Notwendige und Hinreichende Eigenschaften der oberen fehlenden Strecke*

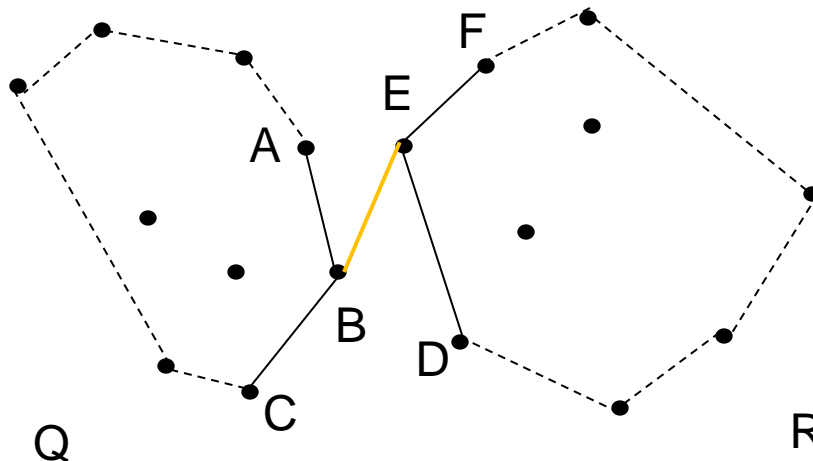
- Sortierung im Uhrzeigersinn um B ist A, E, C
- Sortierung im Uhrzeigersinn um E ist D, B, F
- Winkel ABE ist größer als 180 Grad
- Winkel BEF ist größer als 180 Grad



## Teile & Herrsche

### *Suchen der oberen Strecke*

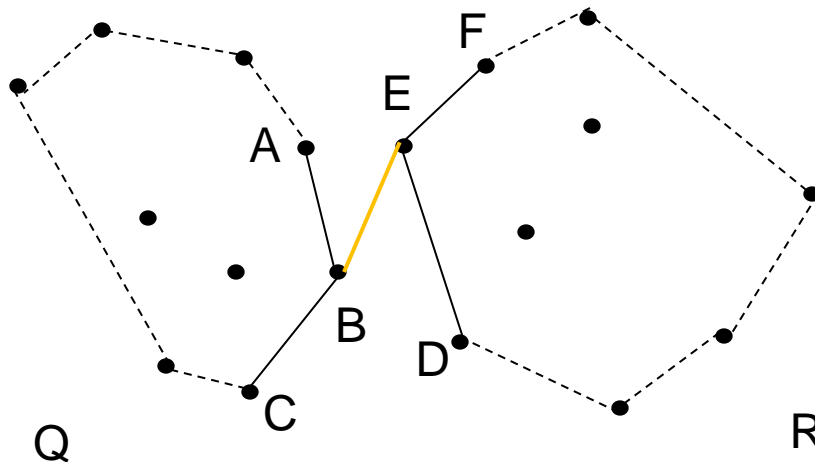
- Sei B der rechteste Knoten von Q und E der linkeste Knoten von R
- Sei A der Vorgänger und C der Nachfolger von B im Uhrzeigersinn
- Sei D der Vorgänger und F der Nachfolger von E im Uhrzeigersinn



## Teile & Herrsche

### *Suchen der oberen Strecke*

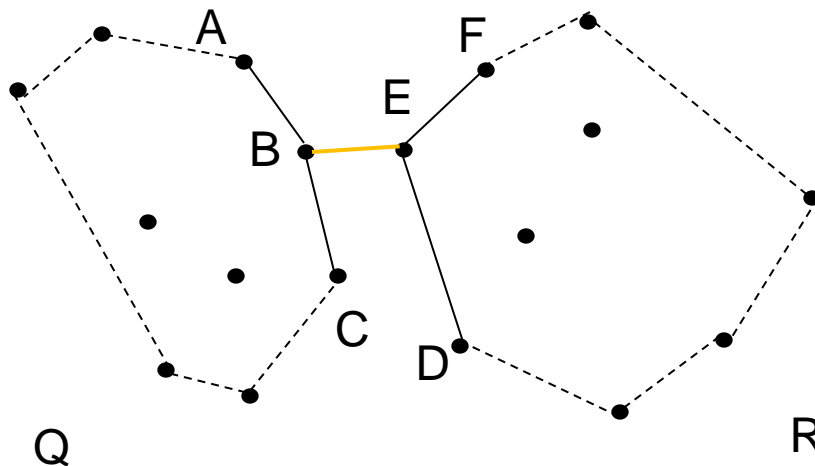
- Beobachtung: Sortierung um B ist A, E, C und Sortierung um E ist D, B, F
- Wenn Winkel ABE < 180 Grad, dann setze B=A und definiere A als den Vorgänger von B und C als den Nachfolger



## Teile & Herrsche

### *Suchen der oberen Strecke*

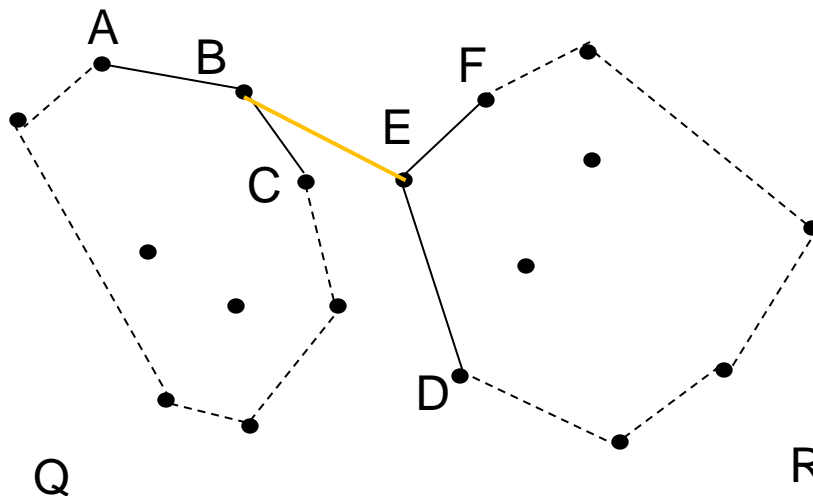
- Beobachtung: Sortierung um B ist A, E, C und Sortierung um E ist D, B, F
- Wenn Winkel ABE < 180 Grad, dann setze B=A und definiere A als den Vorgänger von B und C als den Nachfolger



## Teile & Herrsche

### *Suchen der oberen Strecke*

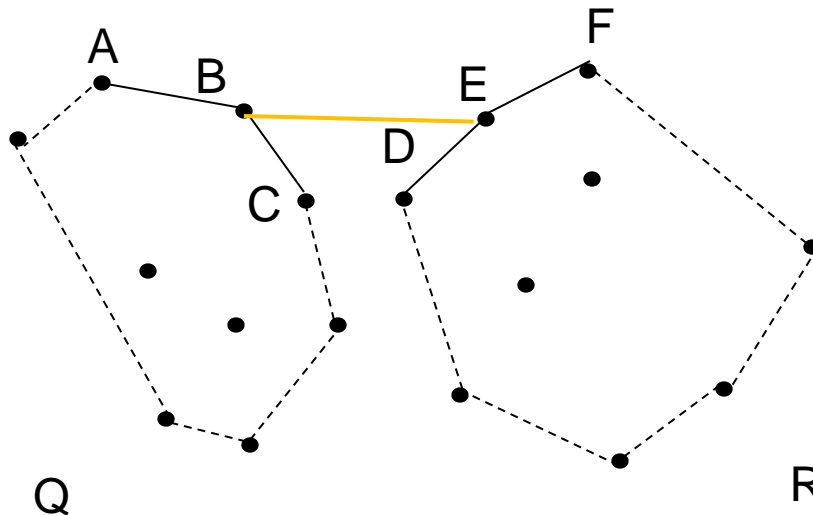
- Beobachtung: Sortierung um B ist A, E, C und Sortierung um E ist D, B, F
- Wenn Winkel ABE < 180 Grad, dann setze B=A und definiere A als den Vorgänger von B und C als den Nachfolger



## Teile & Herrsche

### *Suchen der oberen Strecke*

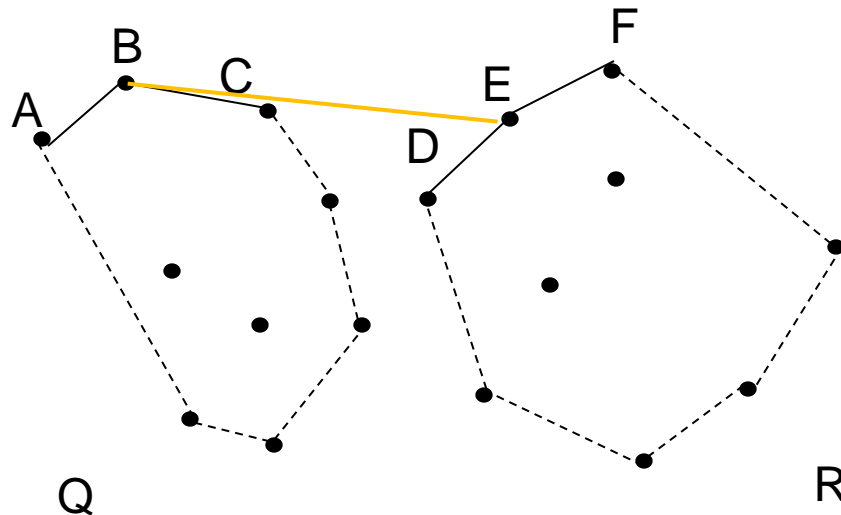
- Beobachtung: Sortierung um B ist A, E, C und Sortierung um E ist D, B, F
- Wenn Winkel BEF < 180 Grad, dann setze E=F und definiere D als den Vorgänger von E und F als den Nachfolger



## Teile & Herrsche

### *Suchen der oberen Strecke*

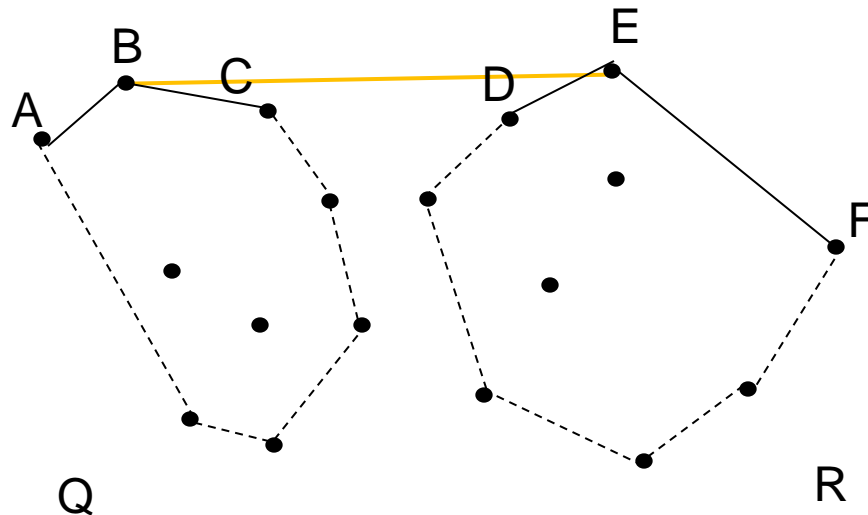
- Beobachtung: Sortierung um B ist A, E, C und Sortierung um E ist D, B, F
- Wenn Winkel ABE < 180 Grad, dann setze B=A und definiere A als den Vorgänger von B und C als den Nachfolger



## Teile & Herrsche

### *Suchen der oberen Strecke*

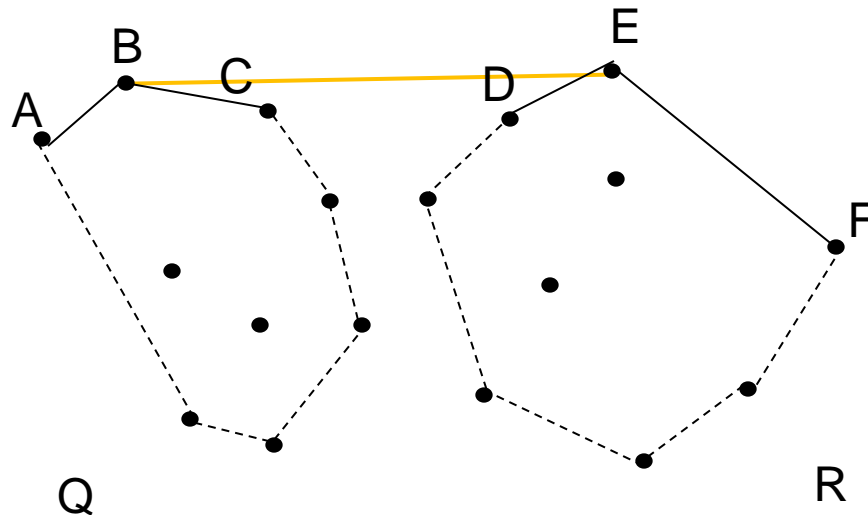
- Beobachtung: Sortierung um B ist A, E, C und Sortierung um E ist D, B, F
- Wenn Winkel BEF < 180 Grad, dann setze E=F und definiere D als den Vorgänger von E und F als den Nachfolger



## Teile & Herrsche

### *Suchen der oberen Strecke*

- Beobachtung: Sortierung um B ist A, E, C und Sortierung um E ist D, B, F
- Beide Winkel  $> 180$  Grad  $\Rightarrow$  hinreichende Bedingung erfüllt



## Teile & Herrsche

### *Suchen der oberen Strecke*

1. Sei B rechtester Punkt von Q; A Vorgänger von B und C Nachfolger von B im Uhrzeigersinn
2. Sei E linkester Punkt von R; D Vorgänger von E und F Nachfolger von E im Uhrzeigersinn
3. **while** hinreichende Bedingung nicht erfüllt **do**
4.     **if** Winkel  $ABE < 180$  **then**  $B \leftarrow A$ ;  $A \leftarrow$  Vorgänger von B;  $C \leftarrow$  Nachfolger von B
5.     **if** Winkel  $BEF < 180$  **then**  $E \leftarrow F$ ;  $D \leftarrow$  Vorgänger von E;  $F \leftarrow$  Nachfolger von E
6. **return** BE

## Teile & Herrsche

### *Suchen der oberen Strecke*

1. Sei B rechtester Punkt von Q; A Vorgänger von B und C Nachfolger von B im Uhrzeigersinn
2. Sei E linkester Punkt von R; D Vorgänger von E und F Nachfolger von E im Uhrzeigersinn
3. **while** hinreichende Bedingung nicht erfüllt **do**
4.     **if** Winkel ABE < 180 **then** B ← A; A ← Vorgänger von B; C ← Nachfolger von B
5.     **if** Winkel BEF < 180 **then** E ← F; D ← Vorgänger von E; F ← Nachfolger von E
6. **return** BE

### *Laufzeit*

- $O(n)$ , da maximal jeder Knoten in der while-Schleife einmal B sein kann

## Teile & Herrsche

### *Lemma 19*

- Die Suche der oberen Strecke hält folgende Invariante aufrecht:
- Sortierung im Uhrzeigersinn um B ist A, E, C
- Sortierung im Uhrzeigersinn um E ist D, B, F

### *Beweis*

- Zu Beginn der **while**-Schleife ist B rechtester Knoten von Q. Da E rechts von B liegt, bleibt nur die Sortierung A,E,C um B. Analog für E.

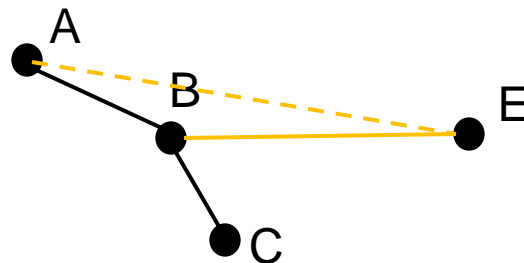
## Teile & Herrsche

### Lemma 19

- Die Suche der oberen Strecke hält folgende Invariante aufrecht:
- Sortierung im Uhrzeigersinn um B ist A, E, C
- Sortierung im Uhrzeigersinn um E ist D, B, F

### Beweis

- Ist während des Verlaufes der while-Schleife der Winkel ABE  $< 180$  Grad, so bleibt die Sortierung auf der linken Seite nach dem Umbenennen der Knoten erhalten, da der Winkel ABE  $< 180$  Grad ist



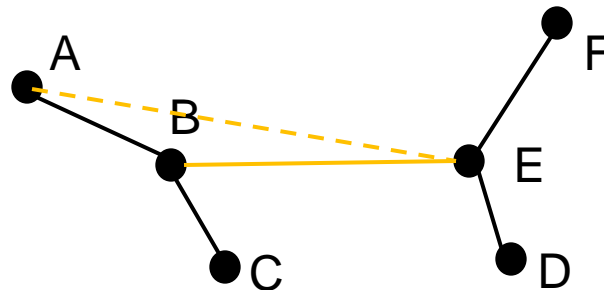
## Teile & Herrsche

### Lemma 19

- Die Suche der oberen Strecke hält folgende Invariante aufrecht:
- Sortierung im Uhrzeigersinn um B ist A, E, C
- Sortierung im Uhrzeigersinn um E ist D, B, F

### Beweis

- Auf der rechten Seite bleibt die Sortierung ebenfalls erhalten, da E weiter links als F ist. Somit kann EF nicht die Kante AE kreuzen.



## Teile & Herrsche

### *Lemma 19*

- Die Suche der oberen Strecke hält folgende Invariante aufrecht:
- Sortierung im Uhrzeigersinn um B ist A, E, C
- Sortierung im Uhrzeigersinn um E ist D, B, F

### *Beweis*

- Der Fall Winkel BEF < 180 Grad ist symmetrisch. Somit bleibt die Invariante erhalten.

## Teile & Herrsche

### *Lemma 20*

- Die Suche nach der oberen und unteren Kante ist korrekt und benötigt  $O(n)$  Zeit.

### *Beweis*

- Die Suche nach der oberen und der unteren Kante ist symmetrisch. Die Laufzeit ist  $O(n)$  wie bereits gezeigt. Es bleibt die Korrektheit zu zeigen.
- Nach Lemma 19 hält der Algorithmus die lokale Sortierung als Invariante aufrecht. Terminiert die Schleife, so sind die Winkel ABE und BEF beide  $> 180$  Grad und somit ist die hinreichende Bedingung erfüllt. Damit ist die zurückgegebene Kante die gesuchte Kante der konvexen Hülle.

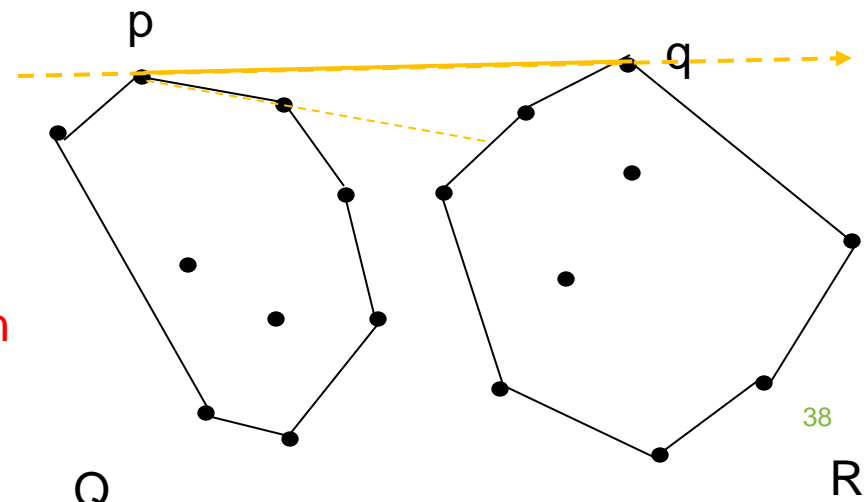
## Teile & Herrsche

### Lemma 20

- Die Suche nach der oberen und unteren Kante ist korrekt und benötigt  $O(n)$  Zeit.

### Beweis

- Es bleibt zu zeigen, dass die Schleife terminiert. Sei dazu  $p$  die obere fehlende Kante der konvexen Hülle.
- Sei o.b.d.A.  $p$  der erste Knoten von  $p$  und  $q$ , der vom Algorithmus untersucht wird (d.h.  $B$  wird auf  $p$  gesetzt).
- Dann ist für jeden Knoten  $E$  aus  $R$ , der die Invariante erfüllt, der Winkel  $ABE$  größer 180 Grad, da alle Knoten aus  $R$  links der gerichteten Geraden durch  $p$  und  $q$  liegen



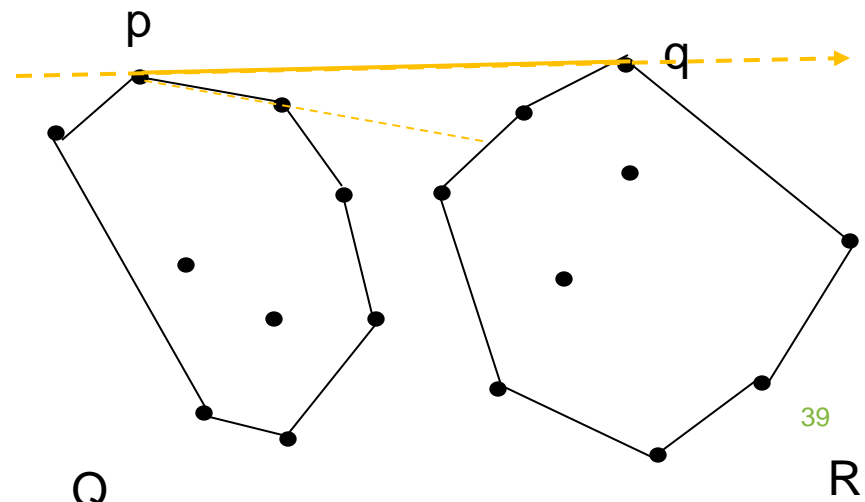
## Teile & Herrsche

### Lemma 20

- Die Suche nach der oberen und unteren Kante ist korrekt und benötigt  $O(n)$  Zeit.

### Beweis

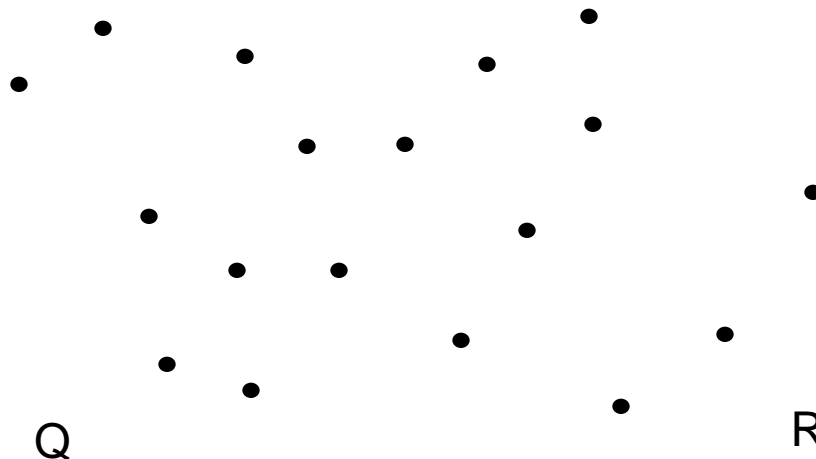
- Damit bleiben die Knoten A,B,C unverändert, bis der Algorithmus auch q gefunden hat und die Schleife terminiert.



## Teile & Herrsche

### *Der konvexe Hülle Algorithmus*

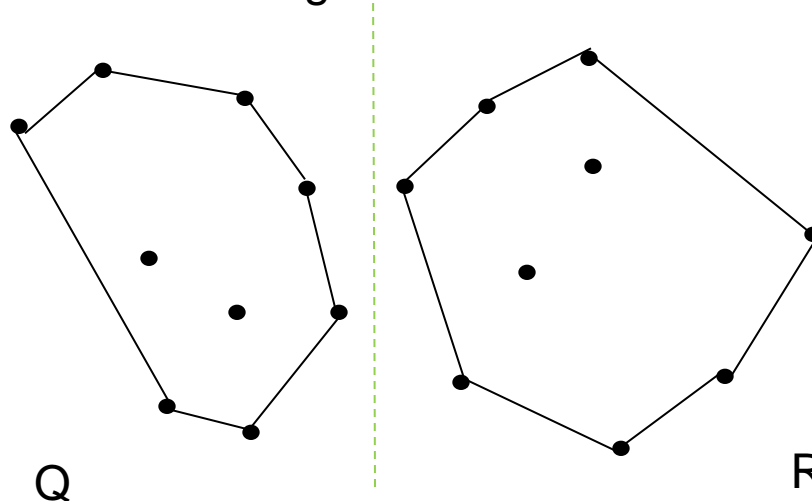
- Teile Punktmenge die Mengen Q und R der  $n/2$  Punkte mit den kleinsten bzw. größten x-Koordinaten auf
- Löse das Problem rekursiv
- Berechne die obere und untere fehlende Strecke
- Lösche die dazwischenliegenden Punkte
- Rekursionsabbruch: Erster Algorithmus



## Teile & Herrsche

### *Der konvexe Hülle Algorithmus*

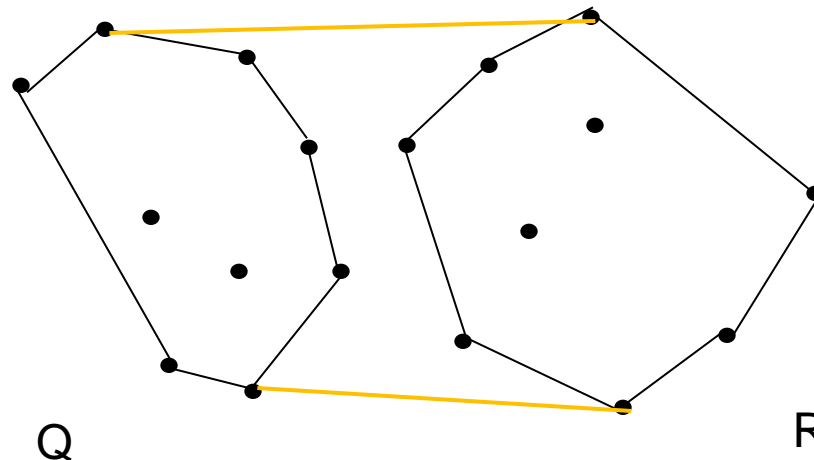
- Teile Punktmenge die Mengen Q und R der  $n/2$  Punkte mit den kleinsten bzw. größten x-Koordinaten auf
- Löse das Problem rekursiv
- Berechne die obere und untere fehlende Strecke
- Lösche die dazwischenliegenden Punkte
- Rekursionsabbruch: Erster Algorithmus



## Teile & Herrsche

### *Der konvexe Hülle Algorithmus*

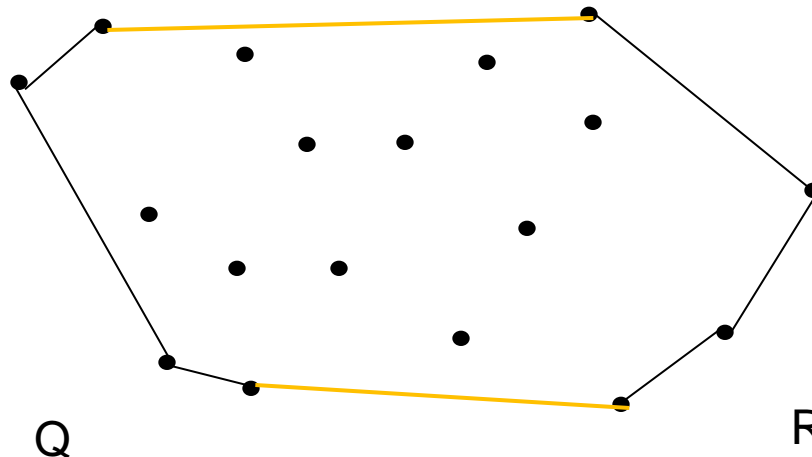
- Teile Punktmenge die Mengen Q und R der  $n/2$  Punkte mit den kleinsten bzw. größten x-Koordinaten auf
- Löse das Problem rekursiv
- Berechne die obere und untere fehlende Strecke
- Lösche die dazwischenliegenden Punkte
- Rekursionsabbruch: Erster Algorithmus



## Teile & Herrsche

### *Der konvexe Hülle Algorithmus*

- Teile Punktmenge die Mengen Q und R der  $n/2$  Punkte mit den kleinsten bzw. größten x-Koordinaten auf
- Löse das Problem rekursiv
- Berechne die obere und untere fehlende Strecke
- Lösche die dazwischenliegenden Punkte
- Rekursionsabbruch: Erster Algorithmus



## Teile & Herrsche

### Satz 21

- Die konvexe Hülle einer Punktmenge kann in  $O(n \log n)$  Zeit mit dem Teile & Herrsche Verfahren berechnet werden.

### Beweis

- Das Entfernen der überflüssigen Kanten geht in  $O(n)$  Zeit.
- Es genügt, zu Beginn des Algorithmus die Punkte einmal nach x-Koordinate zu sortieren. Dies benötigt  $O(n \log n)$  Zeit.
- Nach der Sortierung ergibt sich als Laufzeit:
- $T(n) = T(n/2) + cn$  und  $T(4) = c$
- Wie beim Mergesort ergibt dies Laufzeit  $O(n \log n)$
- Also ist die gesamte Laufzeit  $O(n \log n)$