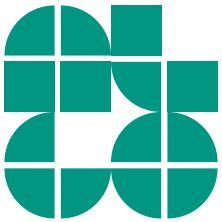
Karlsruhe Institute of Technology

Übung Algorithmische Geometrie
Polygontriangulierung

LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Andreas Gemsa

05.05.2011

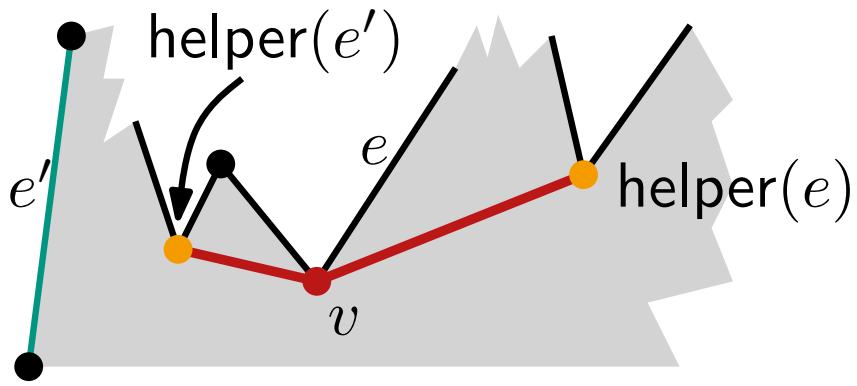


Ablauf

- Besprechung ÜB3
  - Korrektheitsbeweis
  - Lineare Laufzeit?
  - Randabdeckung genug?
  - Polygon-Splitting

Aufgabe 1

Bestimme die Korrektheit von **handleMergeVertex**.



handleMergeVertex(vertex v)

$e \leftarrow$ rechte Kante
if isMergeVertex(helper(e)) **then**
  $\mathcal{D} \leftarrow$ füge (helper(e), v) ein
lösche e aus $\mathcal{T}$
$e' \leftarrow$ Kante links von v in $\mathcal{T}$
if isMergeVertex(helper(e')) **then**
  $\mathcal{D} \leftarrow$ füge (helper(e'), v) ein
helper(e') $\leftarrow v$

Aufgabe 2

Aus Vorlesung bekannt: Algorithmus zur Partitionierung eines Polygons in y-monotone Teilpolygone in $\mathcal{O}(n \log n)$.
Angenommen die Anzahl der Wendeknoten ist in $O(1)$. Reduziere die Laufzeit zu $\mathcal{O}(n)$.

Aufgabe 2

Aus Vorlesung bekannt: Algorithmus zur Partitionierung eines Polygons in y-monotone Teilpolygone in $\mathcal{O}(n \log n)$.
Angenommen die Anzahl der Wendeknoten ist in $O(1)$. Reduziere die Laufzeit zu $\mathcal{O}(n)$.

MakeMonotone(Polygon P)

$\mathcal{D} \leftarrow$ doppelt-verkettete Kantenliste für $(V(P), E(P))$
$\mathcal{Q} \leftarrow$ priority queue für $V(P)$ lexikographisch sortiert
$\mathcal{T} \leftarrow \emptyset$ (binärer Suchbaum für Sweep-Line Status)
while $\mathcal{Q} \neq \emptyset$ **do**
    $v \leftarrow \mathcal{Q}$.nextVertex()
    $\mathcal{Q}$.deleteVertex(v)
    handleVertex(v)
return $\mathcal{D}$

Algorithmus MakeMonotone(P)

Karlsruhe Institute of Technology

MakeMonotone(Polygon P)

$\mathcal{D} \leftarrow$ doppelt-verkettete Kantenliste für $(V(P), E(P))$
$\mathcal{Q} \leftarrow$ priority queue für $V(P)$ lexikographisch sortiert
$\mathcal{T} \leftarrow \emptyset$ (binärer Suchbaum für Sweep-Line Status)
while $\mathcal{Q} \neq \emptyset$ **do**
    $v \leftarrow \mathcal{Q}$.nextVertex()
    $\mathcal{Q}$.deleteVertex(v)
    handleVertex(v)
return $\mathcal{D}$

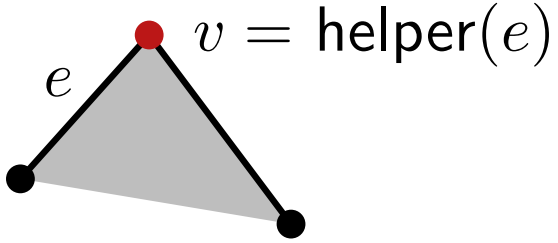
Algorithmus MakeMonotone(P)

MakeMonotone(Polygon P)

$\mathcal{D} \leftarrow$ doppelt-verkettete Kantenliste für $(V(P), E(P))$
$\mathcal{Q} \leftarrow$ priority queue für $V(P)$ lexikographisch sortiert
$\mathcal{T} \leftarrow \emptyset$ (binärer Suchbaum für Sweep-Line Status)
while $\mathcal{Q} \neq \emptyset$ **do**
    $v \leftarrow \mathcal{Q}$.nextVertex()
    $\mathcal{Q}$.deleteVertex(v)
    handleVertex(v)
return $\mathcal{D}$

handleStartVertex(vertex v)

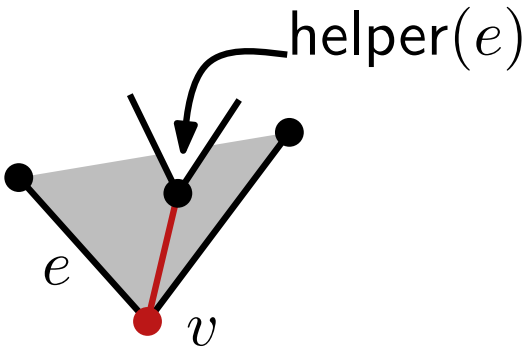
$\mathcal{T} \leftarrow$ füge linke Kante e ein
helper(e) $\leftarrow v$



Algorithmus MakeMonotone(P)

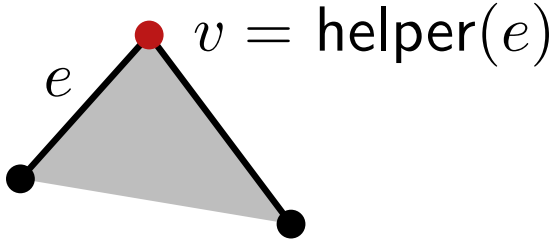
MakeMonotone(Polygon P)

$\mathcal{D} \leftarrow$ doppelt-verkettete Kantenliste für $(V(P), E(P))$
$\mathcal{Q} \leftarrow$ priority queue für $V(P)$ lexikographisch sortiert
$\mathcal{T} \leftarrow \emptyset$ (binärer Suchbaum für Sweep-Line Status)
while $\mathcal{Q} \neq \emptyset$ **do**
  $v \leftarrow \mathcal{Q}$.nextVertex()
  $\mathcal{Q}$.deleteVertex(v)
  handleVertex(v)
return $\mathcal{D}$



handleStartVertex(vertex v)

$\mathcal{T} \leftarrow$ füge linke Kante e ein
helper(e) $\leftarrow v$



handleEndVertex(vertex v)

$e \leftarrow$ linke Kante
if isMergeVertex(helper(e)) **then**
  $\mathcal{D} \leftarrow$ füge (helper(e), v) ein
lösche e aus $\mathcal{T}$

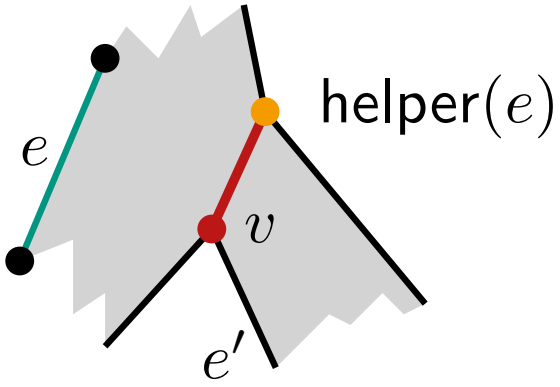
Algorithmus MakeMonotone(P)

MakeMonotone(Polygon P)

$\mathcal{D} \leftarrow$ doppelt-verkettete Kantenliste für $(V(P), E(P))$
$\mathcal{Q} \leftarrow$ priority queue für $V(P)$ lexikographisch sortiert
$\mathcal{T} \leftarrow \emptyset$ (binärer Suchbaum für Sweep-Line Status)
while $\mathcal{Q} \neq \emptyset$ **do**
  $v \leftarrow \mathcal{Q}$.nextVertex()
  $\mathcal{Q}$.deleteVertex(v)
  handleVertex(v)
return $\mathcal{D}$

handleSplitVertex(vertex v)

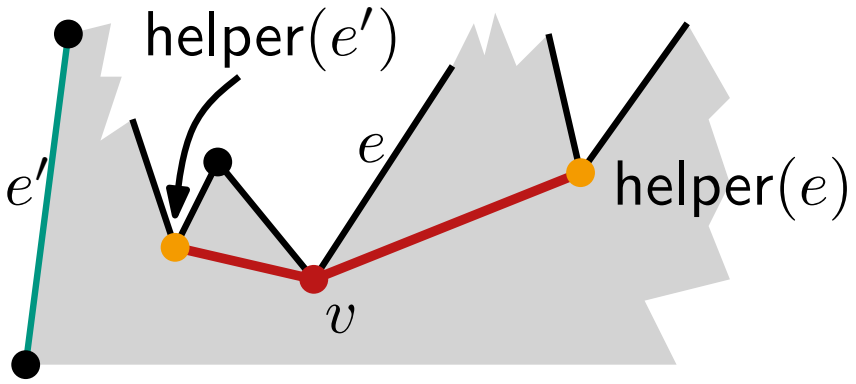
$e \leftarrow$ Kante links von v in $\mathcal{T}$
$\mathcal{D} \leftarrow$ füge (helper(e), v) ein
helper(e) $\leftarrow v$
$\mathcal{T} \leftarrow$ füge rechte Kante e' von v ein
helper(e') $\leftarrow v$



Algorithmus MakeMonotone(P)

MakeMonotone(Polygon P)

$\mathcal{D} \leftarrow$ doppelt-verkettete Kantenliste für $(V(P), E(P))$
$\mathcal{Q} \leftarrow$ priority queue für $V(P)$ lexikographisch sortiert
$\mathcal{T} \leftarrow \emptyset$ (binärer Suchbaum für Sweep-Line Status)
while $\mathcal{Q} \neq \emptyset$ **do**
  $v \leftarrow \mathcal{Q}$.nextVertex()
  $\mathcal{Q}$.deleteVertex(v)
  handleVertex(v)
return $\mathcal{D}$



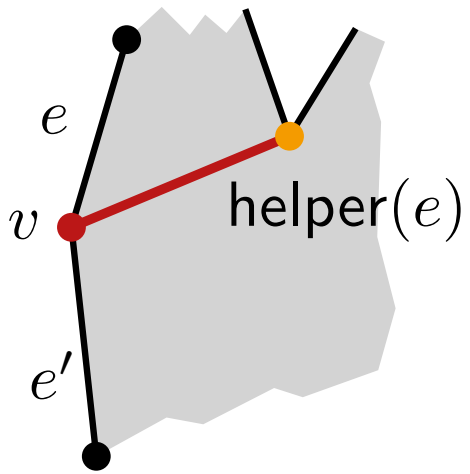
handleMergeVertex(vertex v)

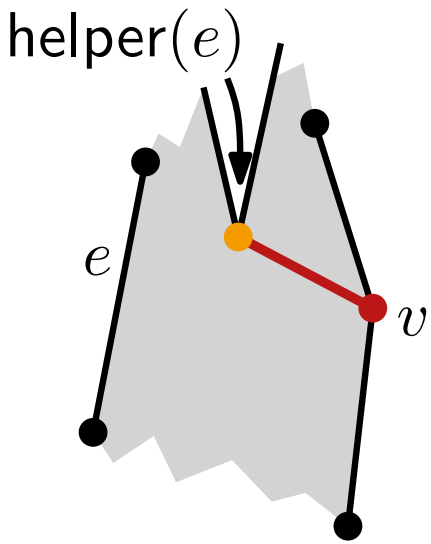
$e \leftarrow$ rechte Kante
if isMergeVertex(helper(e)) **then**
  $\mathcal{D} \leftarrow$ füge (helper(e), v) ein
lösche e aus $\mathcal{T}$
$e' \leftarrow$ Kante links von v in $\mathcal{T}$
if isMergeVertex(helper(e')) **then**
  $\mathcal{D} \leftarrow$ füge (helper(e'), v) ein
helper(e') $\leftarrow v$

Algorithmus MakeMonotone(P)

MakeMonotone(Polygon P)

$\mathcal{D} \leftarrow$ doppelt-verkettete Kantenliste für $(V(P), E(P))$
$\mathcal{Q} \leftarrow$ priority queue für $V(P)$ lexikographisch sortiert
$\mathcal{T} \leftarrow \emptyset$ (binärer Suchbaum für Sweep-Line Status)
while $\mathcal{Q} \neq \emptyset$ **do**
  $v \leftarrow \mathcal{Q}$.nextVertex()
  $\mathcal{Q}$.deleteVertex(v)
  handleVertex(v)
return $\mathcal{D}$





handleRegularVertex(vertex v)

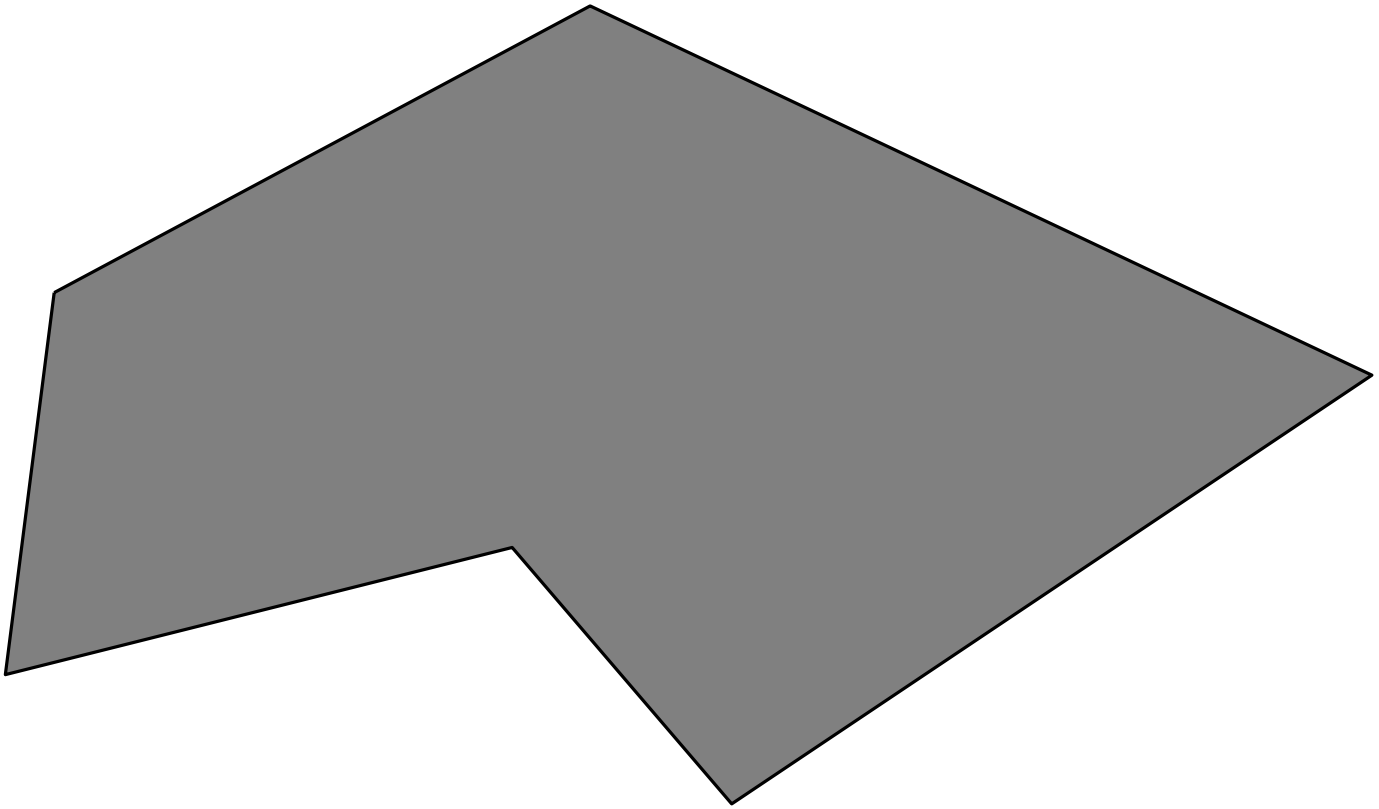
if P liegt lokal rechts von v **then**
  $e, e' \leftarrow$ obere, untere Kante
  if isMergeVertex(helper(e)) **then**
    $\mathcal{D} \leftarrow$ füge (helper(e), v) ein
  lösche e aus $\mathcal{T}$
  $\mathcal{T} \leftarrow$ füge e' ein; helper(e') $\leftarrow v$
else
  $e \leftarrow$ Kante links von v in $\mathcal{T}$
  if isMergeVertex(helper(e)) **then**
    $\mathcal{D} \leftarrow$ füge (helper(e), v) ein
  helper(e) $\leftarrow v$

Aufgabe 3

$\mathcal{P}$ sei Polygon.
Kameras sind so platziert, dass der vollständige Rand von $\mathcal{P}$ abgedeckt wird. Wird automatisch auch das Innere vollständig abgedeckt?

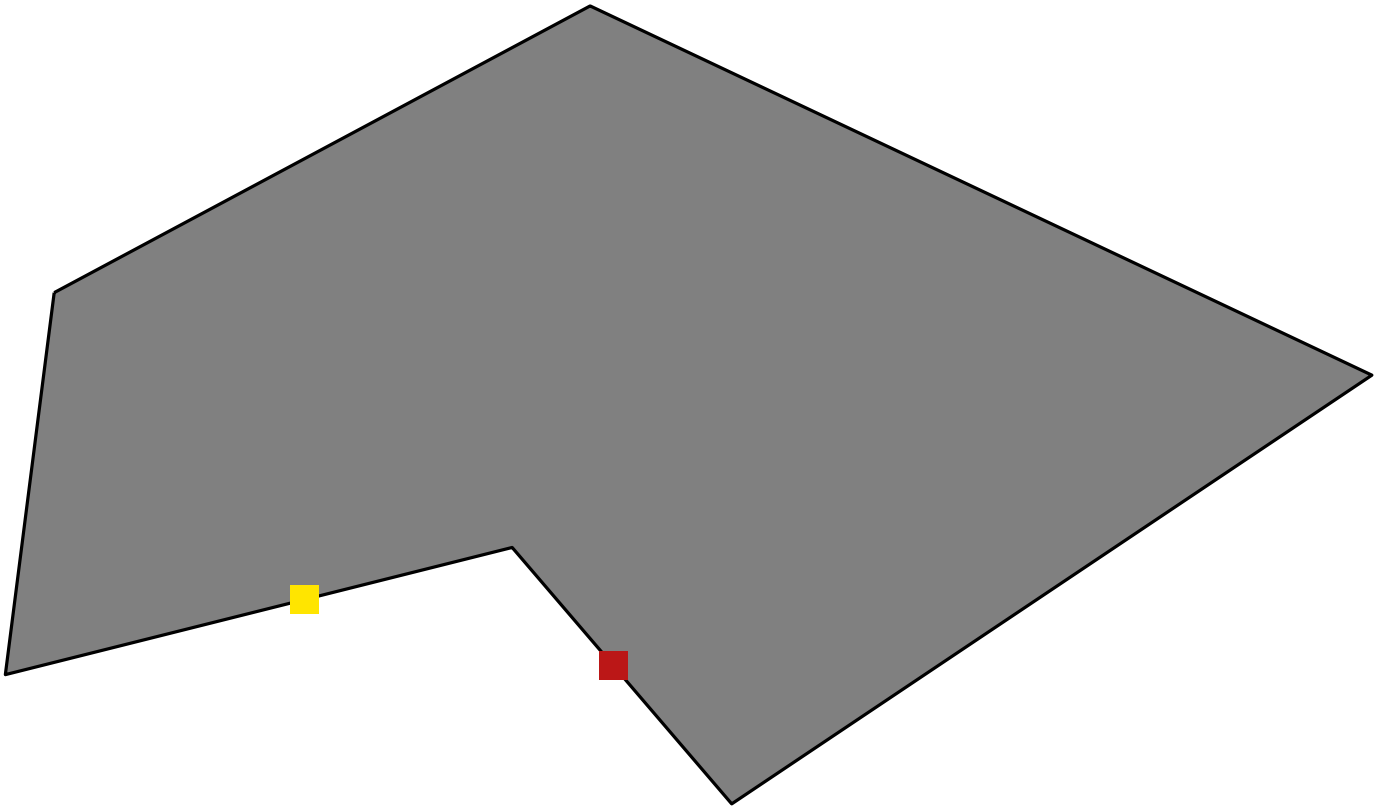
Aufgabe 3

$\mathcal{P}$ sei Polygon.
Kameras sind so platziert, dass der vollständige Rand von $\mathcal{P}$ abgedeckt wird. Wird automatisch auch das Innere vollständig abgedeckt?



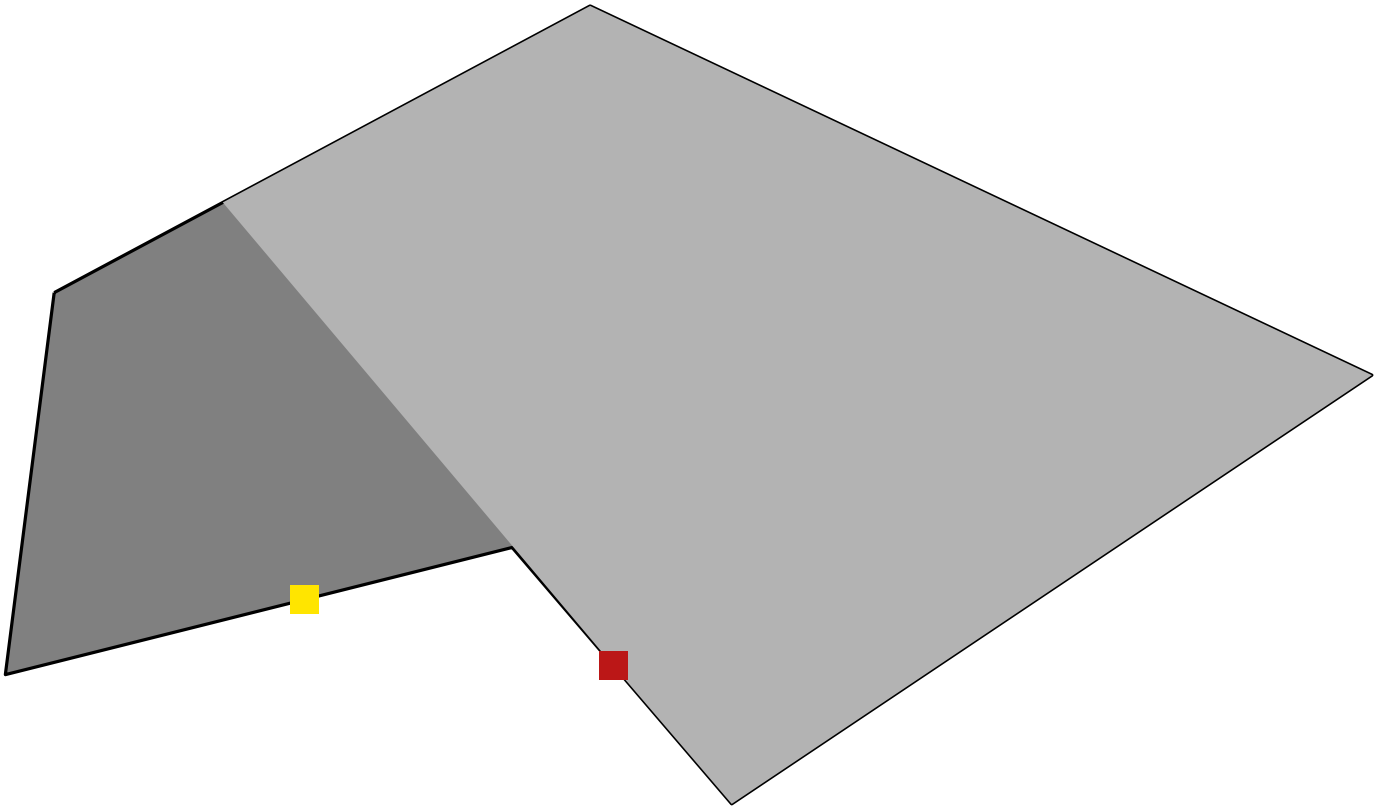
Aufgabe 3

$\mathcal{P}$ sei Polygon.
Kameras sind so platziert, dass der vollständige Rand von $\mathcal{P}$ abgedeckt wird. Wird automatisch auch das Innere vollständig abgedeckt?



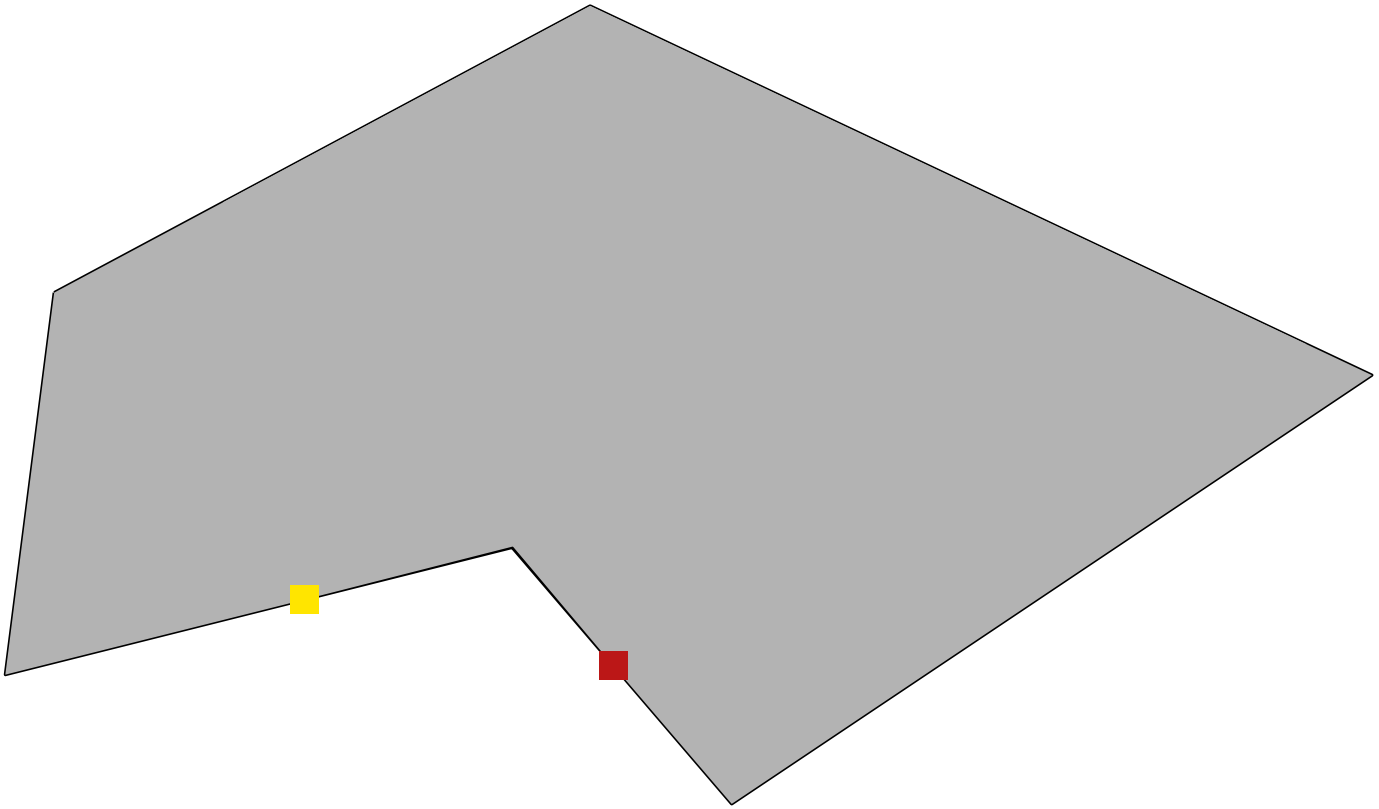
Aufgabe 3

$\mathcal{P}$ sei Polygon.
Kameras sind so platziert, dass der vollständige Rand von $\mathcal{P}$ abgedeckt wird. Wird automatisch auch das Innere vollständig abgedeckt?



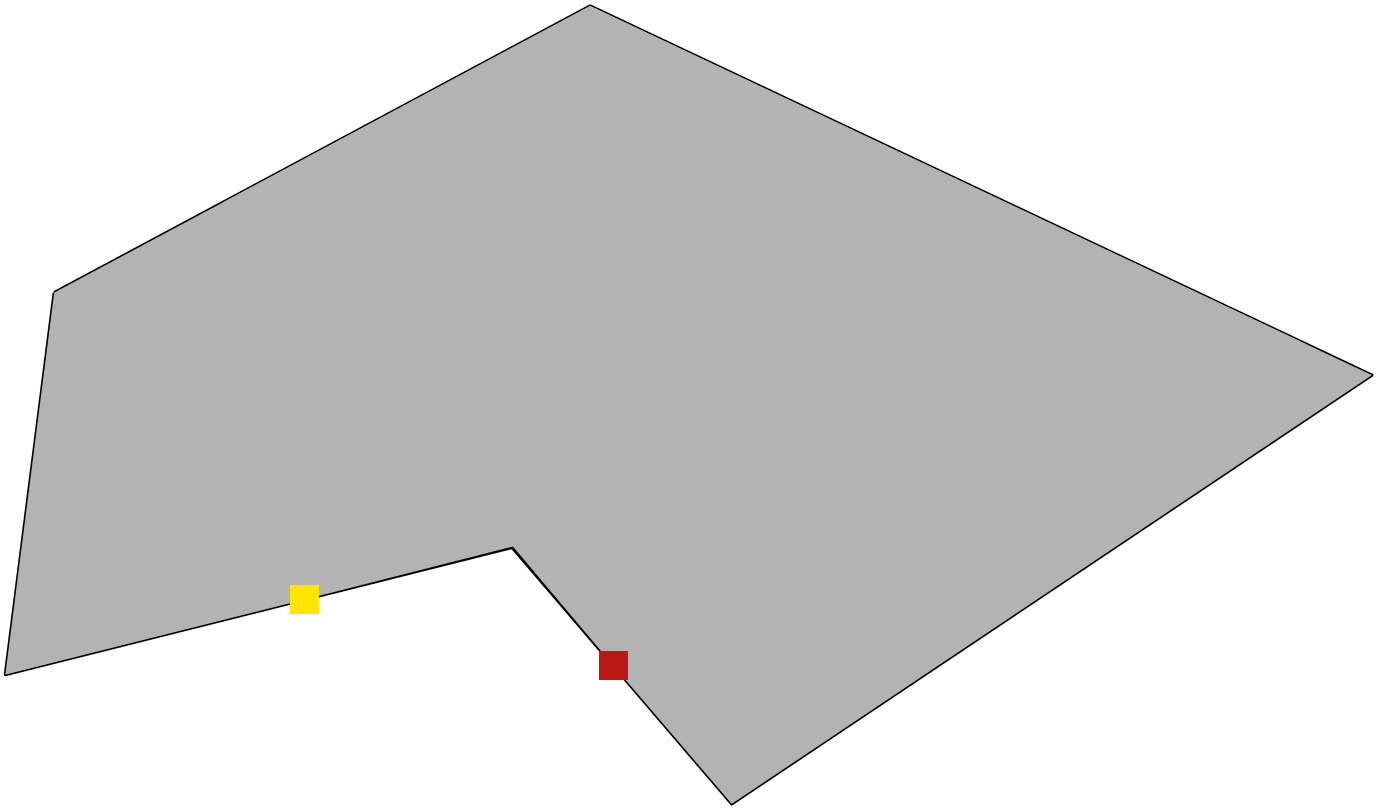
Aufgabe 3

$\mathcal{P}$ sei Polygon.
Kameras sind so platziert, dass der vollständige Rand von $\mathcal{P}$ abgedeckt wird. Wird automatisch auch das Innere vollständig abgedeckt?



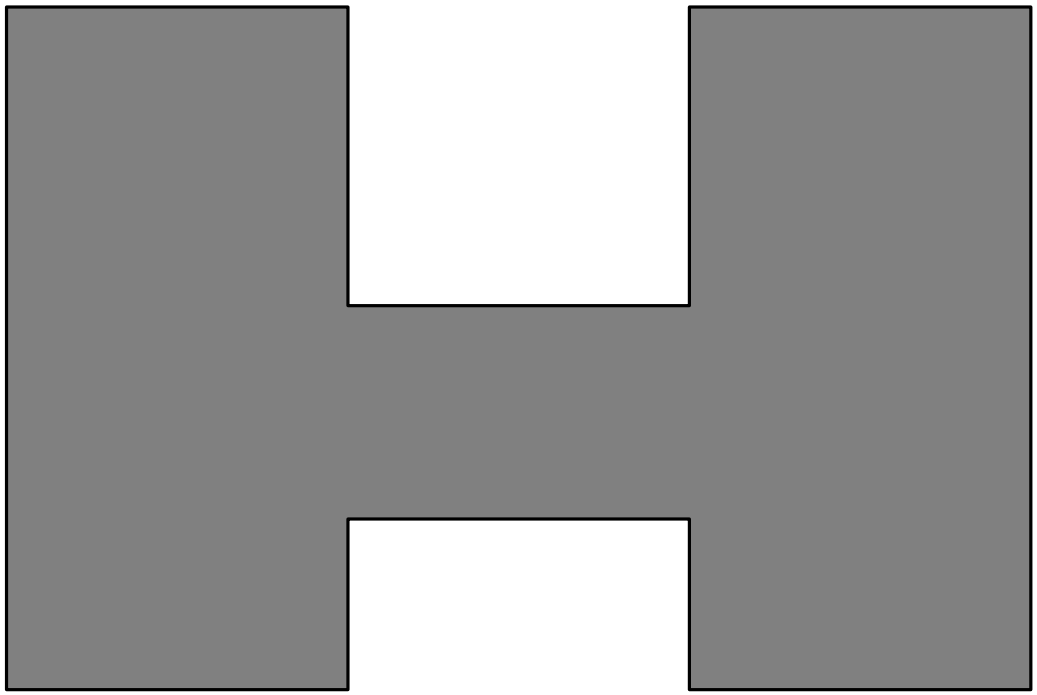
Aufgabe 3

$\mathcal{P}$ sei Polygon.
Kameras sind so platziert, dass der vollständige Rand von $\mathcal{P}$ abgedeckt wird. Wird automatisch auch das Innere vollständig abgedeckt?



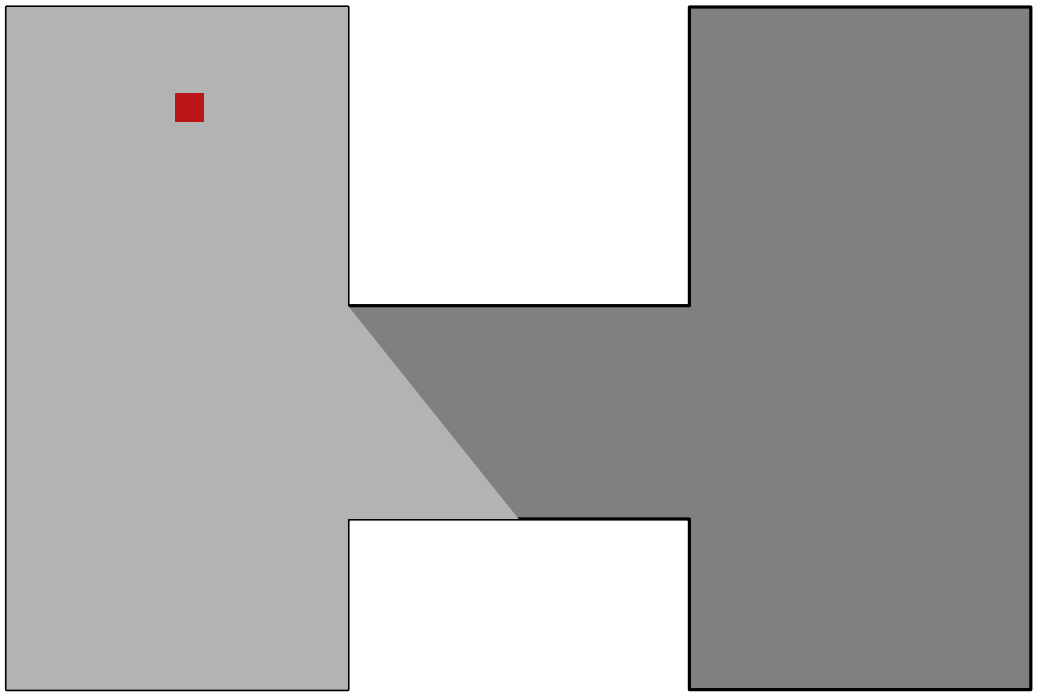
Aufgabe 3

$\mathcal{P}$ sei Polygon.
Kameras sind so platziert, dass der vollständige Rand von $\mathcal{P}$ abgedeckt wird. Wird automatisch auch das Innere vollständig abgedeckt?



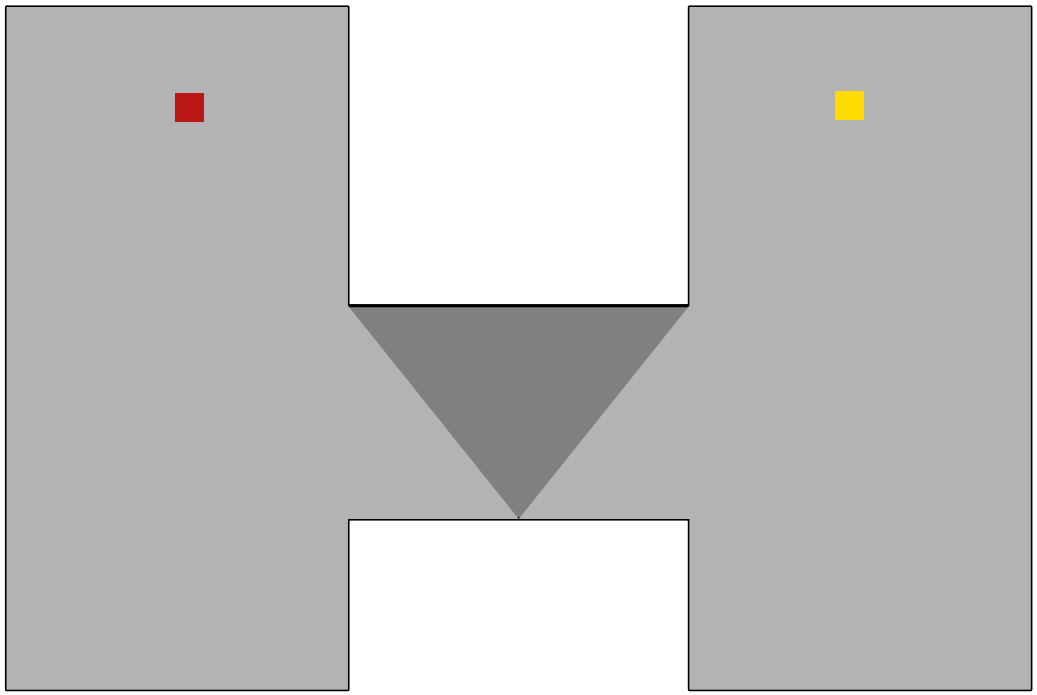
Aufgabe 3

$\mathcal{P}$ sei Polygon.
Kameras sind so platziert, dass der vollständige Rand von $\mathcal{P}$ abgedeckt wird. Wird automatisch auch das Innere vollständig abgedeckt?



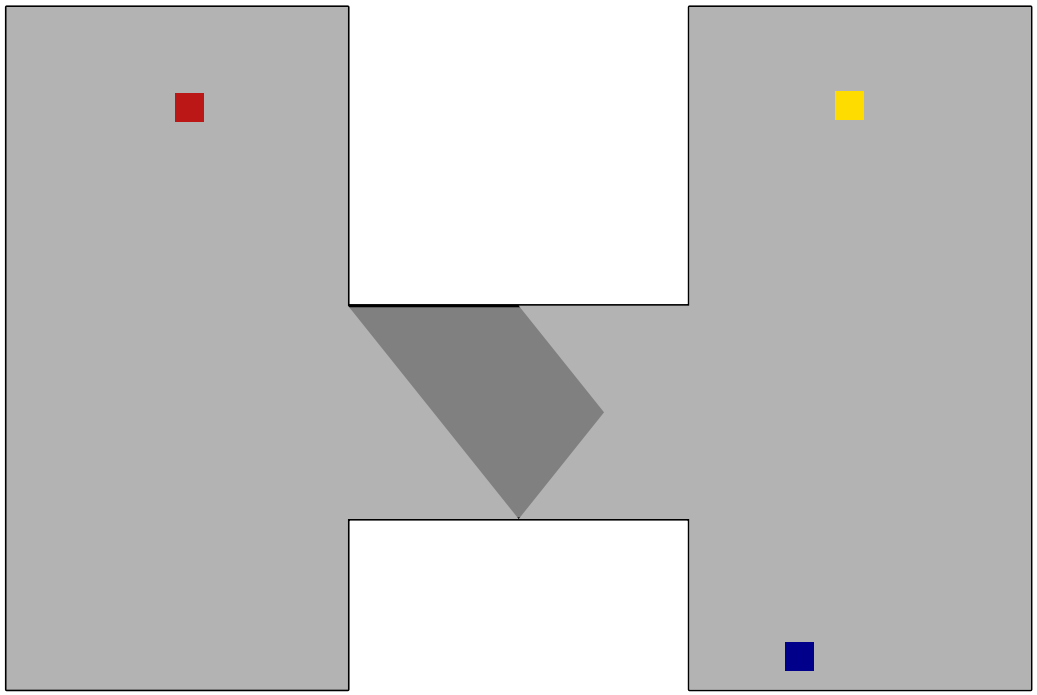
Aufgabe 3

$\mathcal{P}$ sei Polygon.
Kameras sind so platziert, dass der vollständige Rand von $\mathcal{P}$ abgedeckt wird. Wird automatisch auch das Innere vollständig abgedeckt?



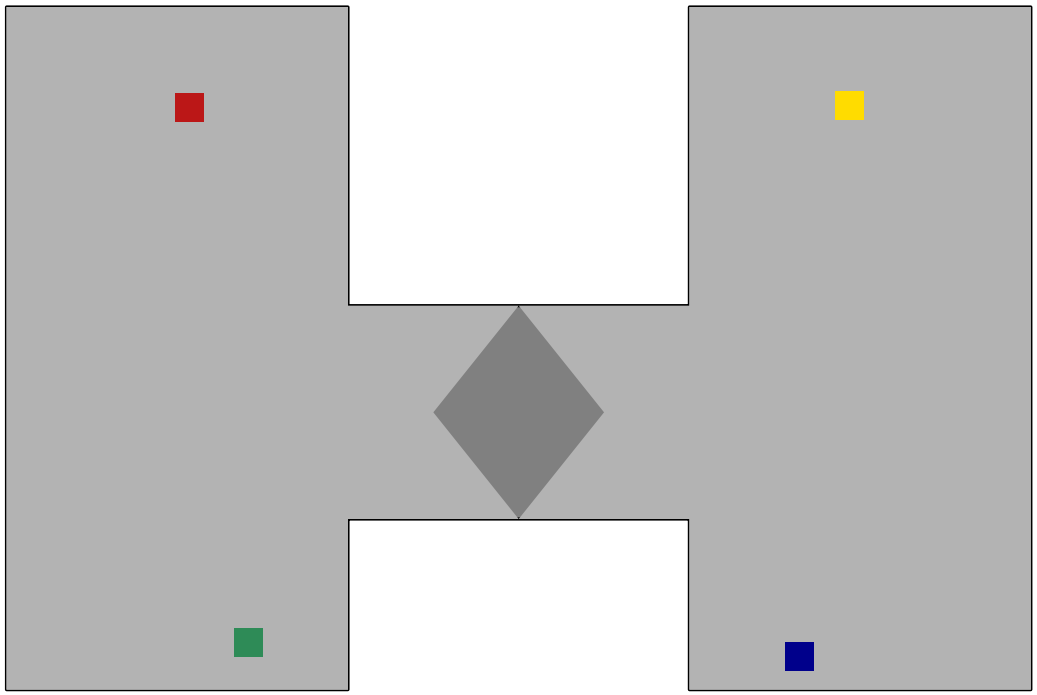
Aufgabe 3

$\mathcal{P}$ sei Polygon.
Kameras sind so platziert, dass der vollständige Rand von $\mathcal{P}$ abgedeckt wird. Wird automatisch auch das Innere vollständig abgedeckt?



Aufgabe 3

$\mathcal{P}$ sei Polygon.
Kameras sind so platziert, dass der vollständige Rand von $\mathcal{P}$ abgedeckt wird. Wird automatisch auch das Innere vollständig abgedeckt?



Aufgabe 4

Gebe einen $\mathcal{O}(n \log n)$ Algorithmus an der ein einfaches Polygon aus n Knoten in zwei einfache Polygone auftrennt. Jedes der beiden soll aus höchstens $\lfloor 2n/3 \rfloor + 2$ Knoten bestehen.

Aufgabe 4

Gebe einen $\mathcal{O}(n \log n)$ Algorithmus an der ein einfaches Polygon aus n Knoten in zwei einfache Polygone auftrennt. Jedes der beiden soll aus höchstens $\lfloor 2n/3 \rfloor + 2$ Knoten bestehen.

- Trianguliere das Polygon
- Berechne Dualgraph zum Polygon
- Bestimme alle Blätter im Dualgraphen (Dualgraph ist ein Baum)

Aufgabe 4

Gebe einen $\mathcal{O}(n \log n)$ Algorithmus an der ein einfaches Polygon aus n Knoten in zwei einfache Polygone auftrennt. Jedes der beiden soll aus höchstens $\lfloor 2n/3 \rfloor + 2$ Knoten bestehen.

- Trianguliere das Polygon
- Berechne Dualgraph zum Polygon
- Bestimme alle Blätter im Dualgraphen (Dualgraph ist ein Baum)

Geht's noch schneller?

Das war's!

Karlsruhe Institute of Technology

Fragen?

Nächster Termin:

Donnertag, 12.05, 10:15 Uhr

Raum 131, Gebäude 50.34