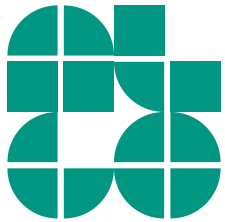
Übung Algorithmische Geometrie
Polygontriangulierung

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

Benjamin Niedermann

07.05.2014



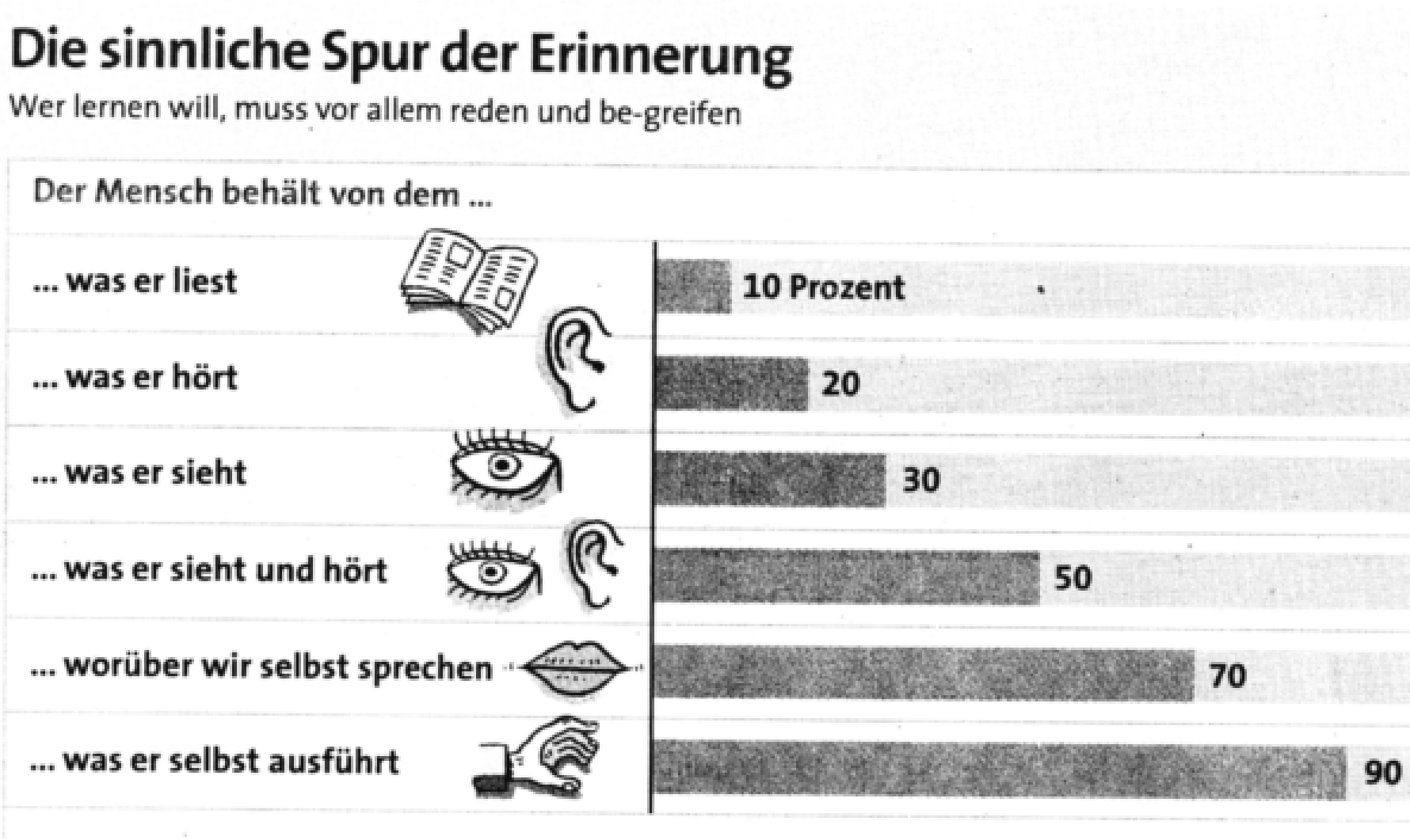
Karlsruhe Institute of Technology

Vergabe der Projekte

Übungsblatt 3

Projekte

Ziel: Visualisierung von in der Vorlesung vorgestellten Algorithmen
→ tiefes eigenes Verständnis & Lernmaterial für Kommilitonen.



Projekte

Ziel: Visualisierung von in der Vorlesung vorgestellten Algorithmen
→ tiefes eigenes Verständnis & Lernmaterial für Kommilitonen.

Ablauf: Jedes Team (Größe 2) bearbeitet ein eigenes Thema.

Projekte

Ziel: Visualisierung von in der Vorlesung vorgestellten Algorithmen
→ tiefes eigenes Verständnis & Lernmaterial für Kommilitonen.

Ablauf: Jedes Team (Größe 2) bearbeitet ein eigenes Thema.

Anforderungen an Implementierung:

- Ablauf der Algorithmen soll visualisiert werden, z.B. schrittweise
- Benutzer soll Eingabe über graphische Oberfläche machen können.
- Interaktion mit Benutzer, z.B. schrittweises *vor* und *zurück*.
- Benutzer soll bereits vorhandene Beispieleingaben laden können.
- Bedienungsanleitung soll in Programm enthalten sein.

Projekte

Ziel: Visualisierung von in der Vorlesung vorgestellten Algorithmen
→ tiefes eigenes Verständnis & Lernmaterial für Kommilitonen.

Ablauf: Jedes Team (Größe 2) bearbeitet ein eigenes Thema.

Anforderungen an Implementierung:

- Ablauf der Algorithmen soll visualisiert werden, z.B. schrittweise
- Benutzer soll Eingabe über graphische Oberfläche machen können.
- Interaktion mit Benutzer, z.B. schrittweises *vor* und *zurück*.
- Benutzer soll bereits vorhandene Beispieleingaben laden können.
- Bedienungsanleitung soll in Programm enthalten sein.

Programmiersprache:

- Java (empfohlen)
- geeignete andere Sprachen sind auch erlaubt.

Projekte

Ziel: Visualisierung von in der Vorlesung vorgestellten Algorithmen
→ tiefes eigenes Verständnis & Lernmaterial für Kommilitonen.

Ablauf: Jedes Team (Größe 2) bearbeitet ein eigenes Thema.

Anforderungen an Implementierung:

- Ablauf der Algorithmen soll visualisiert werden, z.B. schrittweise
- Benutzer soll Eingabe über graphische Oberfläche machen können.
- Interaktion mit Benutzer, z.B. schrittweises *vor* und *zurück*.
- Benutzer soll bereits vorhandene Beispieleingaben laden können.
- Bedienungsanleitung soll in Programm enthalten sein.

Programmiersprache:

- Java (empfohlen)
- geeignete andere Sprachen sind auch erlaubt.

Anforderungen an Abgabe:

- Abgabe enthält plattformunabhängige **ausführbare Datei**, z.B. jar-Datei
- gerne auch direkt im Browser ausführbar, z.B. als Java-Applet

Projekte

Ziel: Visualisierung von in der Vorlesung vorgestellten Algorithmen
→ tiefes eigenes Verständnis & Lernmaterial für Kommilitonen.

Ablauf: Jedes Team (Größe 2) bearbeitet ein eigenes Thema.

Anforderungen an Implementierung:

- Ablauf der Algorithmen soll visualisiert werden, z.B. schrittweise
- Benutzer soll Eingabe über graphische Oberfläche machen können.

Wichtig:
Programm soll leicht auszuführen und zu bedienen sein!

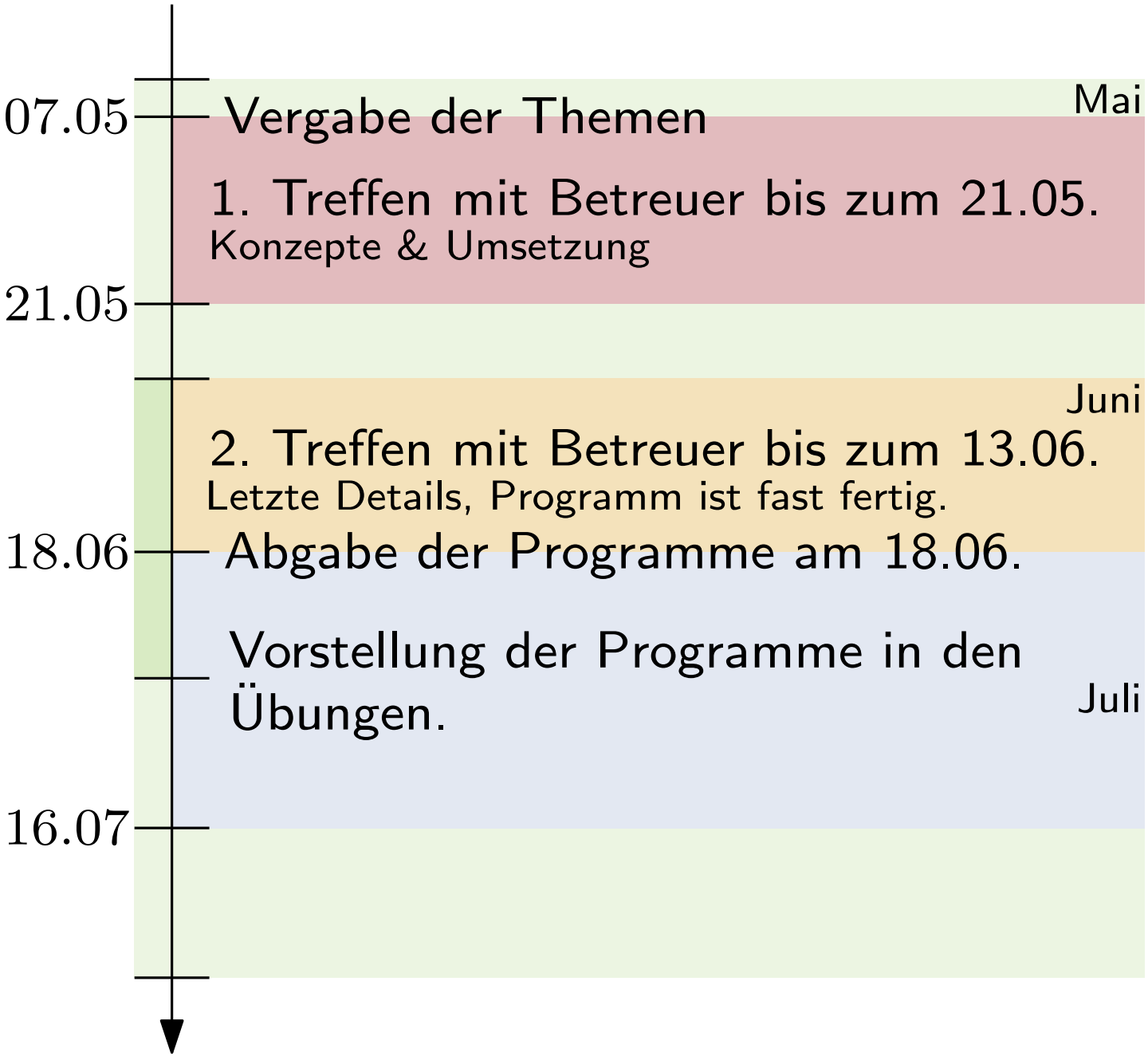
Programmiersprache:

- Java (empfohlen)
- geeignete andere Sprachen sind auch erlaubt.

Anforderungen an Abgabe:

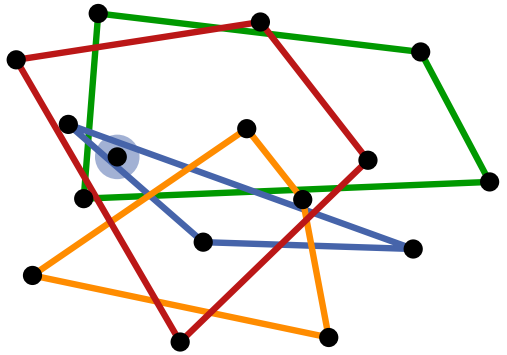
- Abgabe enthält plattformunabhängige **ausführbare Datei**, z.B. jar-Datei
- gerne auch direkt im Browser ausführbar, z.B. als Java-Applet

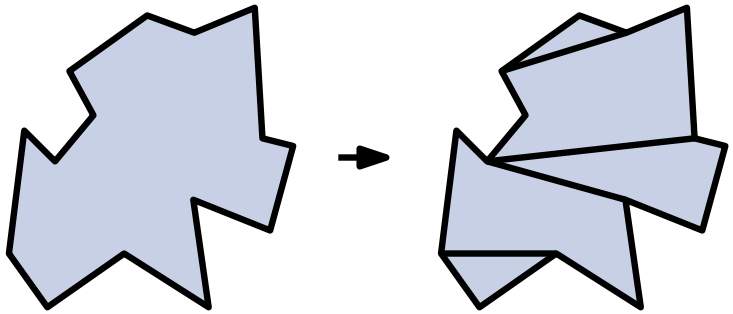
Projekte – Zeitlicher Ablauf



Themen

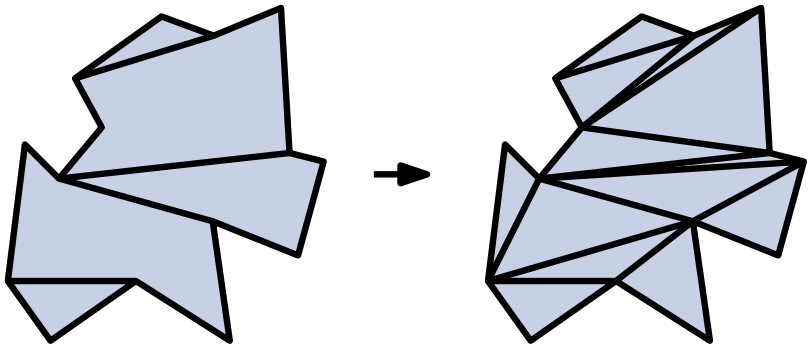
Thema 1
Konvexe Hülle – Chans Algorithmus





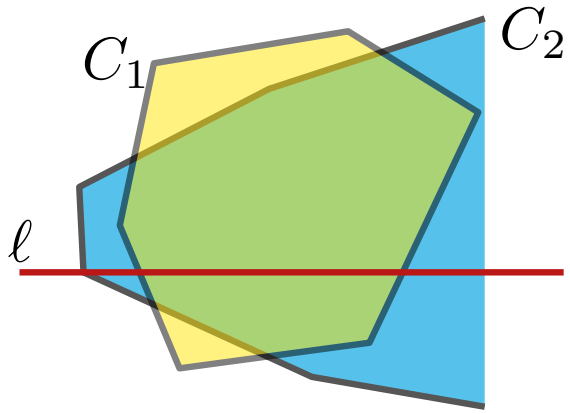
Thema 2
Einfaches Polygon $\rightarrow$ y-monotones Polygon

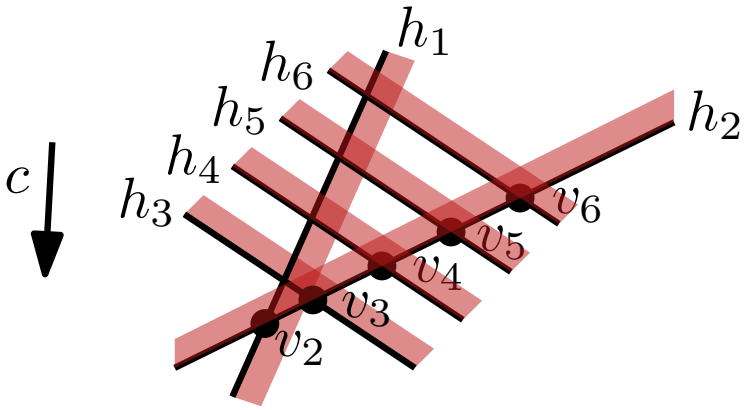
Thema 3
y-monotones Polygon $\rightarrow$ Triangulierung



Themen

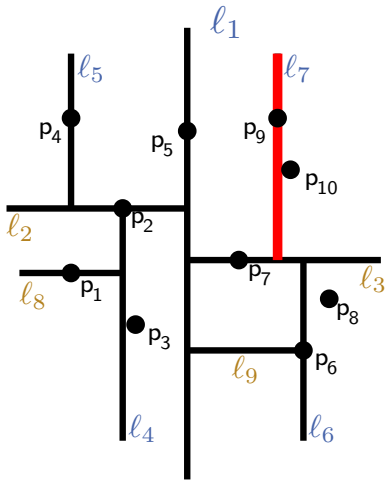
Thema 4
Lin. Prog. – Schnitt von konvexen Regionen

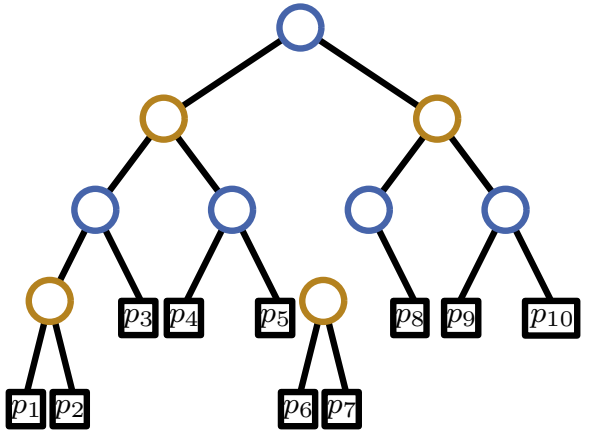




Thema 5
Lin. Prog. – Inkrementeller Ansatz

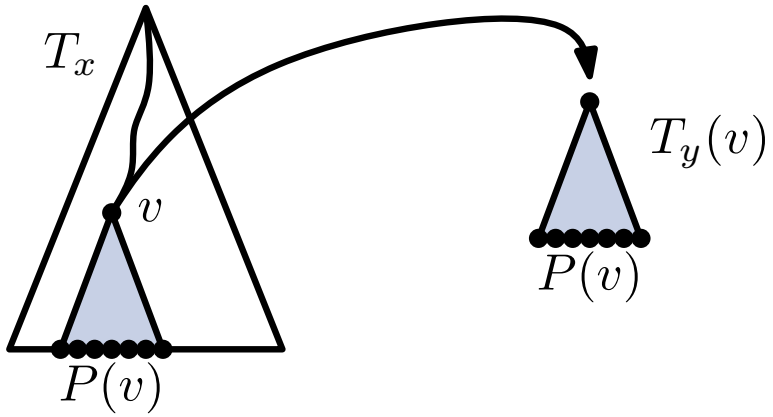
Thema 6
Bereichsanfrage – kd-Tree

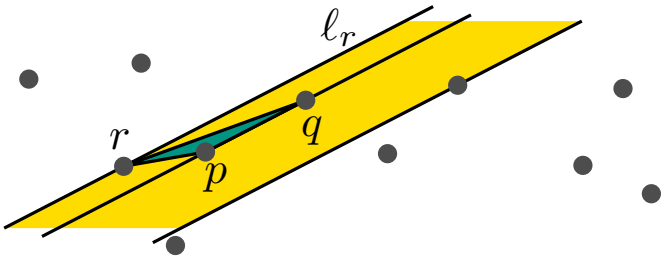




Themen

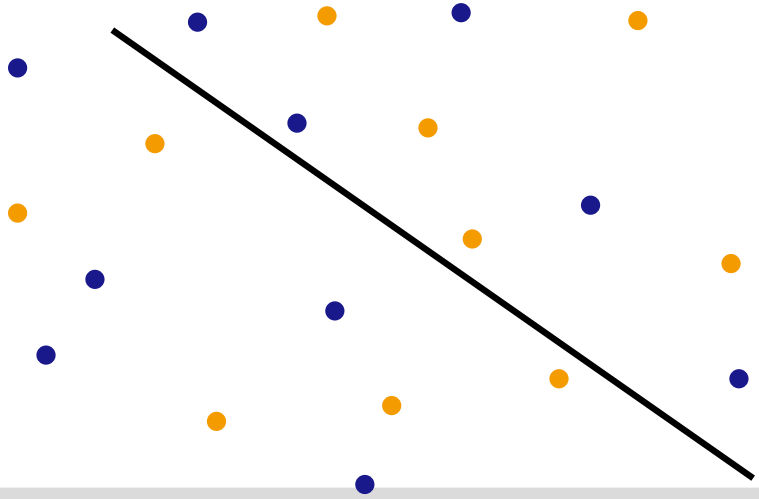
Thema 7
Bereichsanfrage – Range-Tree





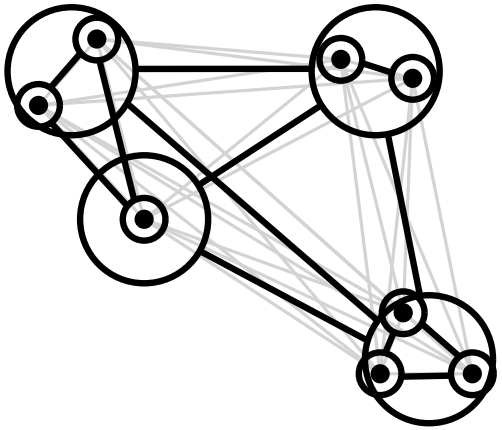
Thema 8
Dualität – Kleinstes Dreieck

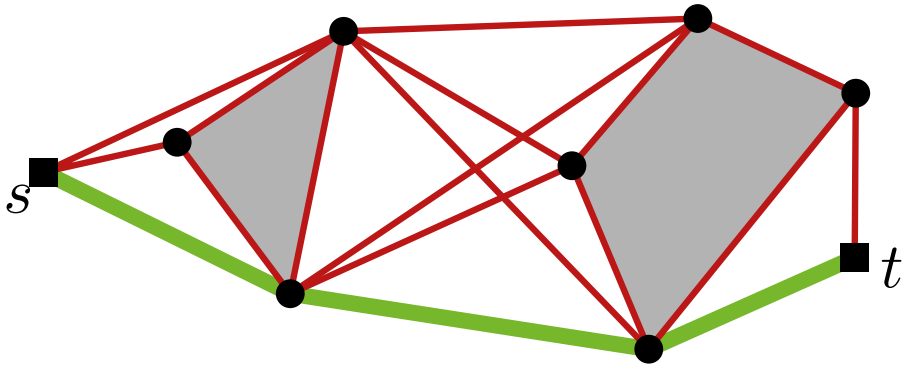
Thema 9
Dualität – Ham-Sandwich-Theorem



Themen

Thema 10
Well-Separated Pair Decomposition





Thema 11
Sichtbarkeitsgraph

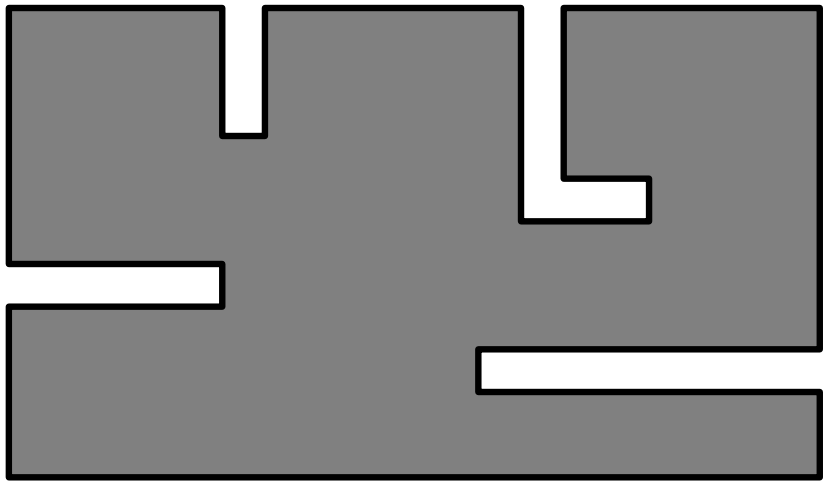
Karlsruhe Institute of Technology

Vergabe der Projekte

Übungsblatt 3

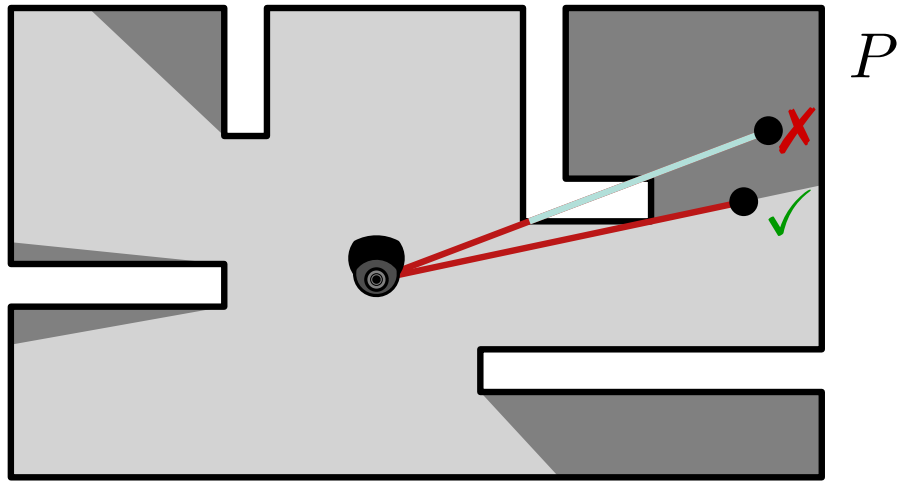
Das Kunstgalerie-Problem

Aufgabe: Installiere ein Kamerasystem zur Überwachung einer Kunstgalerie, so dass jede Stelle der Galerie gesehen wird.



Das Kunstgalerie-Problem

Aufgabe: Installiere ein Kamerasystem zur Überwachung einer Kunstgalerie, so dass jede Stelle der Galerie gesehen wird.



Annahme: Galerie ist ein *einfaches* Polygon P mit n Ecken (keine Schnitte, keine Löcher)

Beobachtung: jede Kamera sieht sternförmiges Gebiet

Definition: Punkt $p \in P$ ist *sichtbar* von $c \in P$ wenn $\overline{cp} \in P$

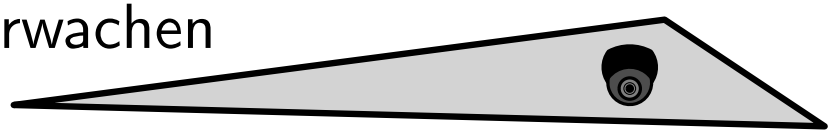
Ziel: Nutze möglichst wenige Kameras!

NP-schwer!

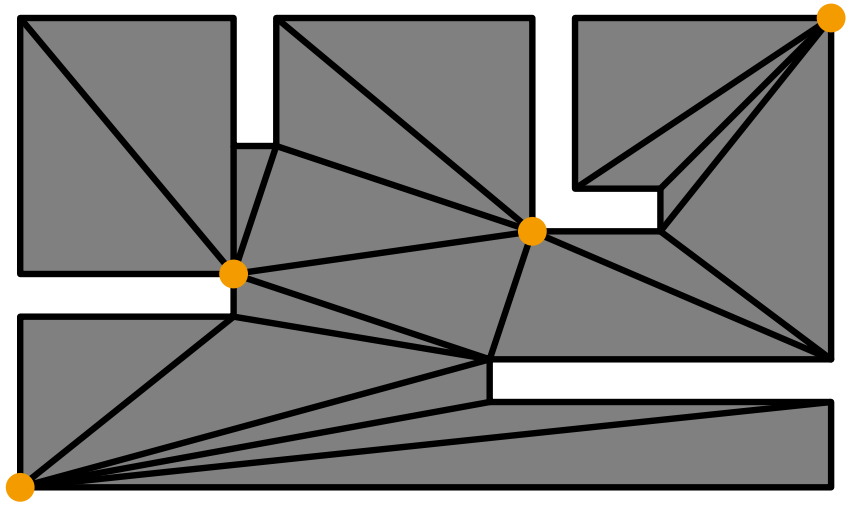
$\rightarrow$ Anzahl hängt von der Komplexität n und der Form von P ab

Vereinfachung des Problems

Beobachtung: Dreiecke sind leicht zu überwachen



Idee: zerlege P in Dreiecke und überwache die Dreiecke



Satz 1: Jedes einfache Polygon mit n Ecken besitzt eine Triangulierung; jede Triangulierung besteht aus $n-2$ Dreiecken.

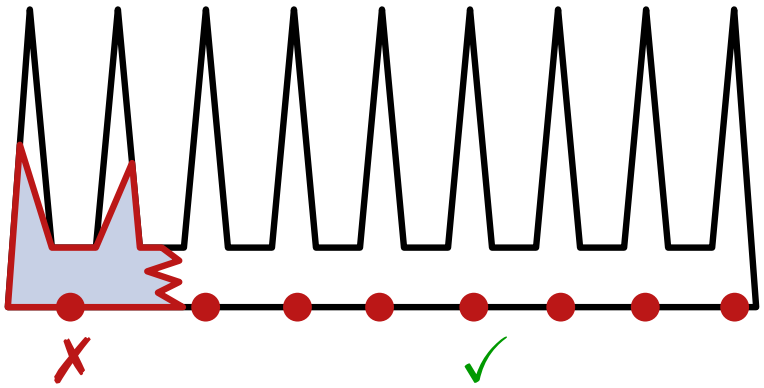
- P lässt sich mit $n-2$ Kameras in den Dreiecken überwachen
- P lässt sich mit $\approx n/2$ Kameras auf den Diagonalen überwachen
- P lässt sich mit noch weniger Kameras auf den Ecken überwachen

Das Art-Gallery-Theorem [Chvátal '75]

Satz 2: Für ein einfaches Polygon P mit n Ecken sind manchmal $\lfloor n/3 \rfloor$ Kameras nötig, aber immer ausreichend um P zu überwachen.

Beweis:

- Finde einfaches Polygon für beliebiges n, das $\approx n/3$ Kameras braucht!



- Teil 2 an der Tafel.

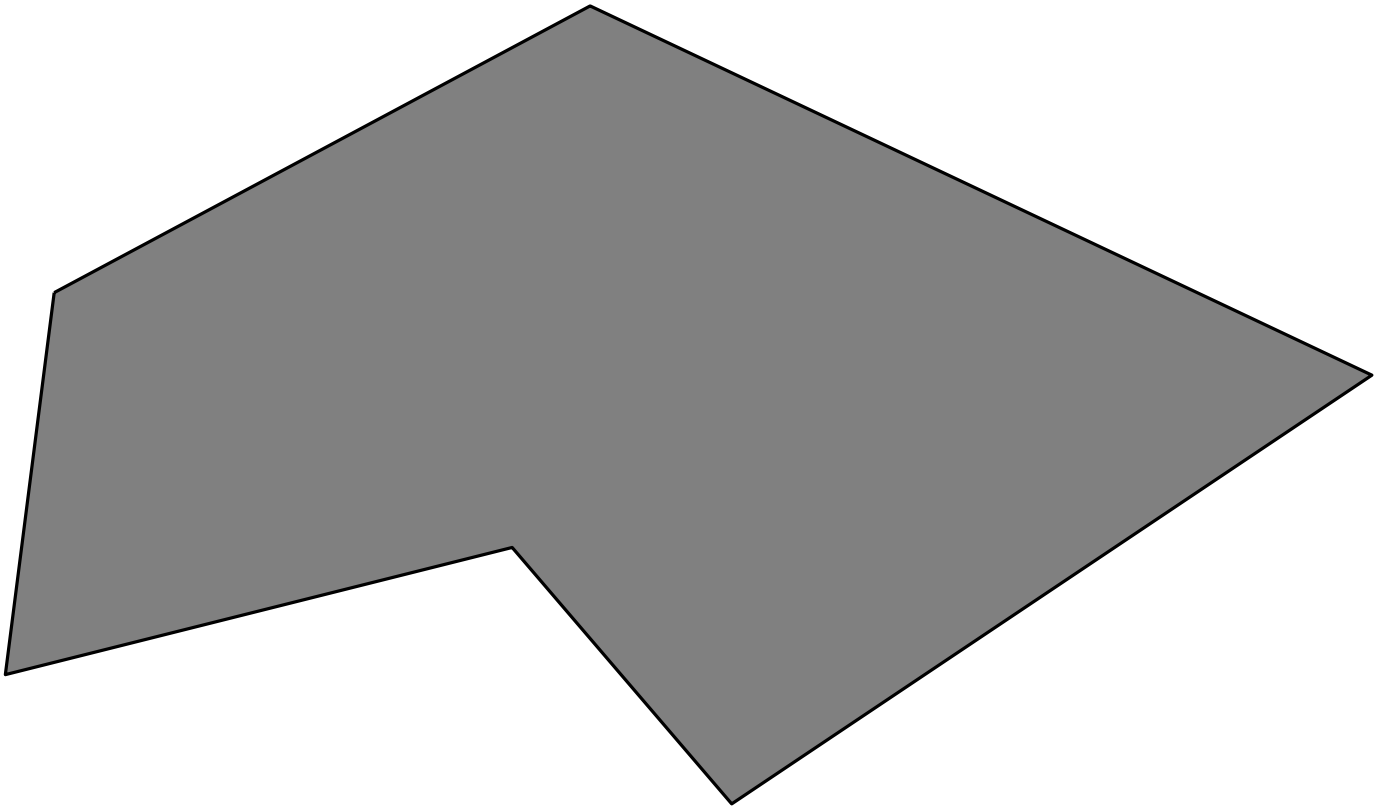
Fazit: Hat man eine Triangulierung, lassen sich $\lfloor n/3 \rfloor$ Kameras in $O(n)$ Zeit platzieren.

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?

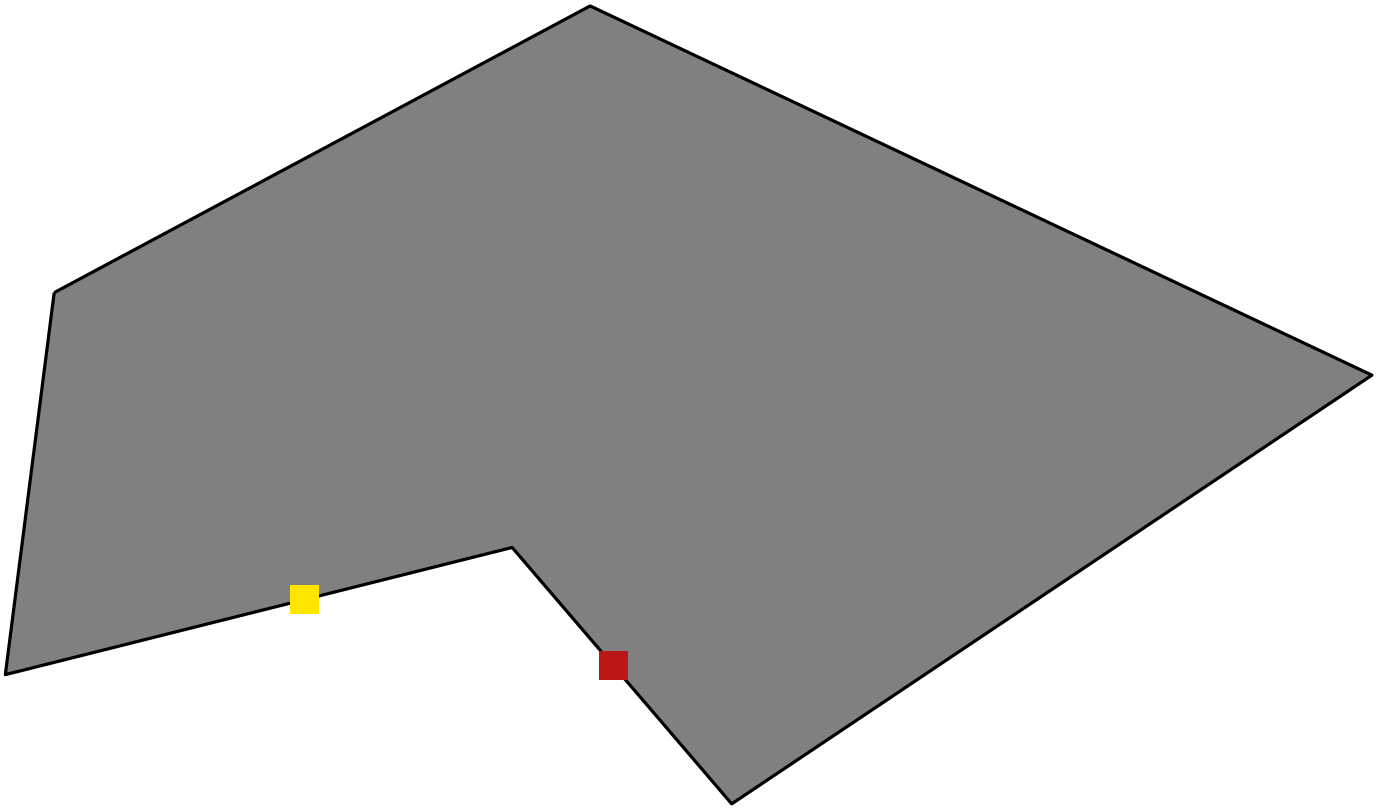
Aufgabe 3

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



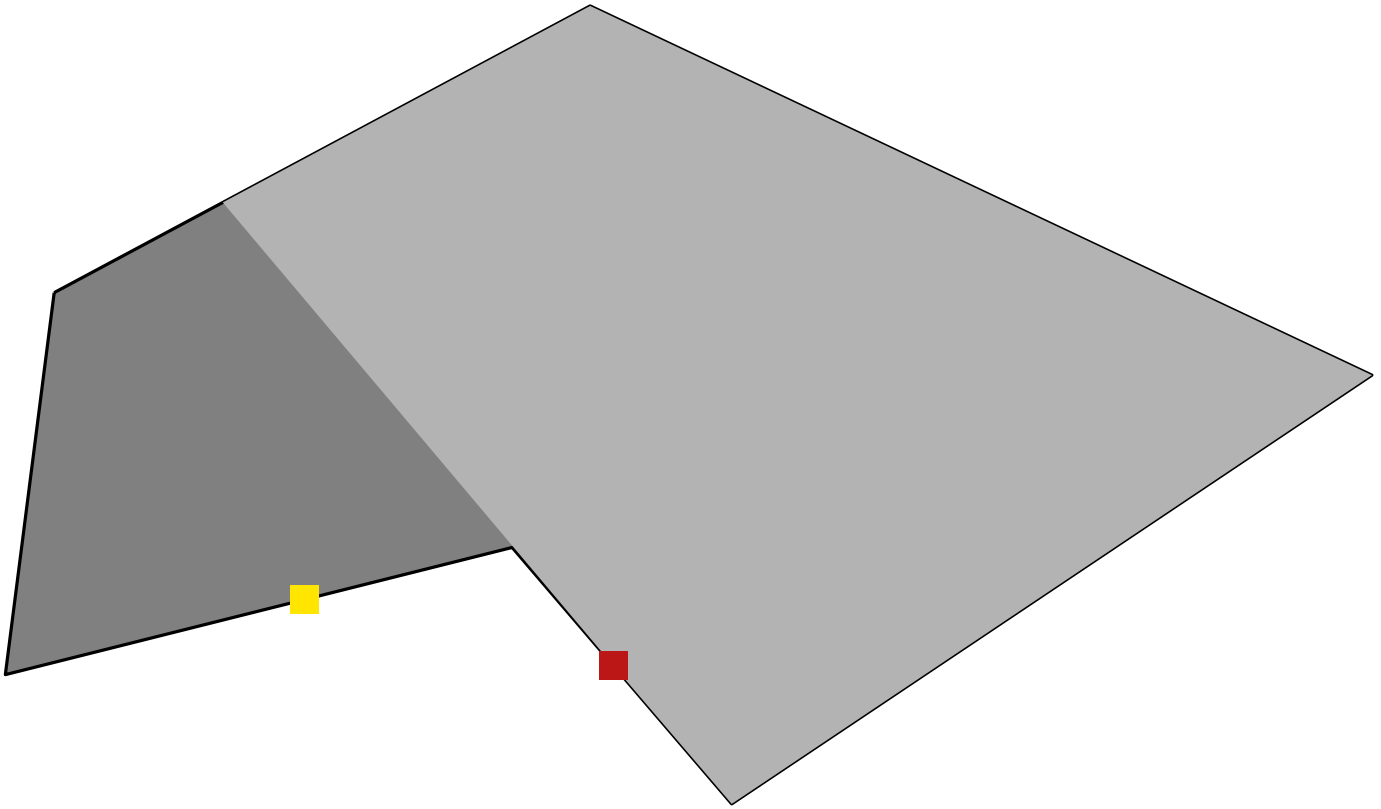
Aufgabe 3

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



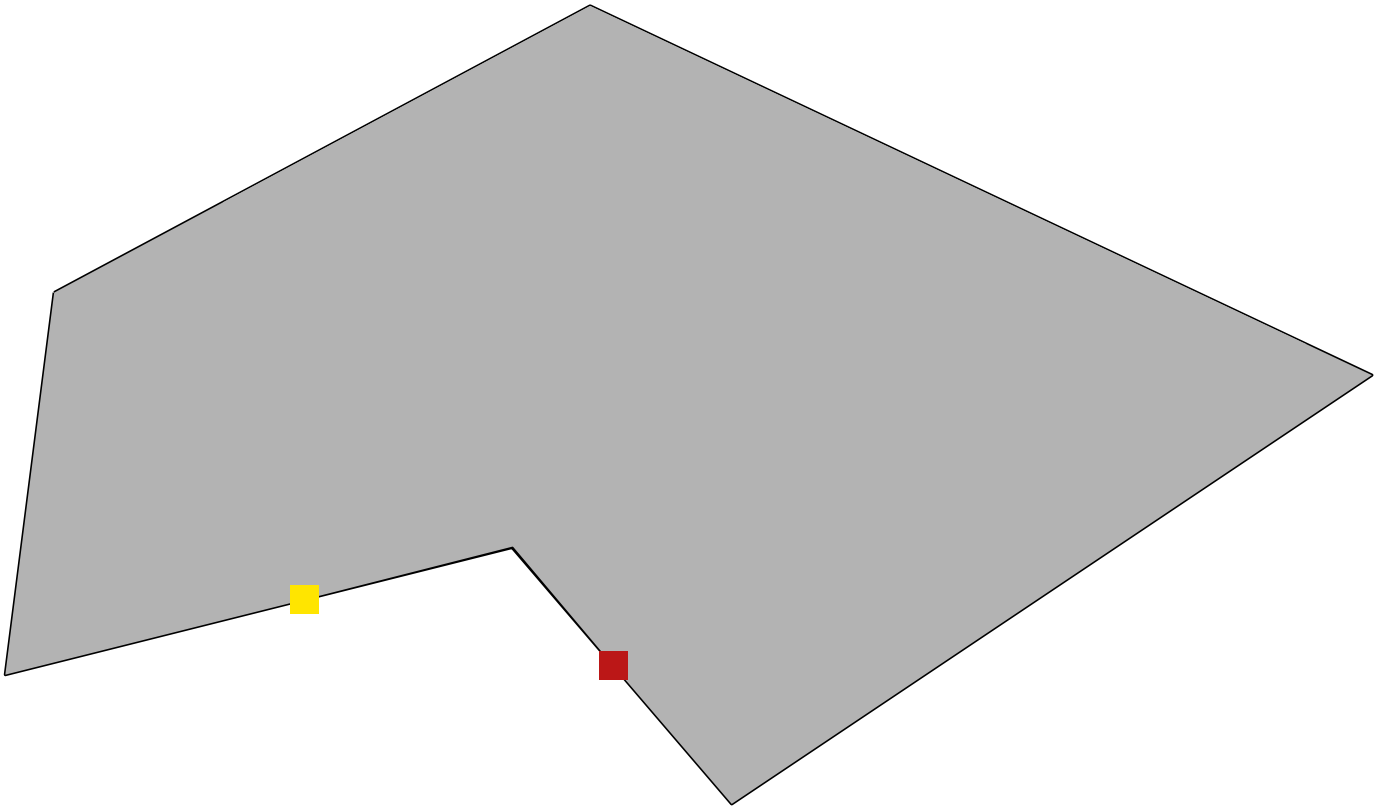
Aufgabe 3

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



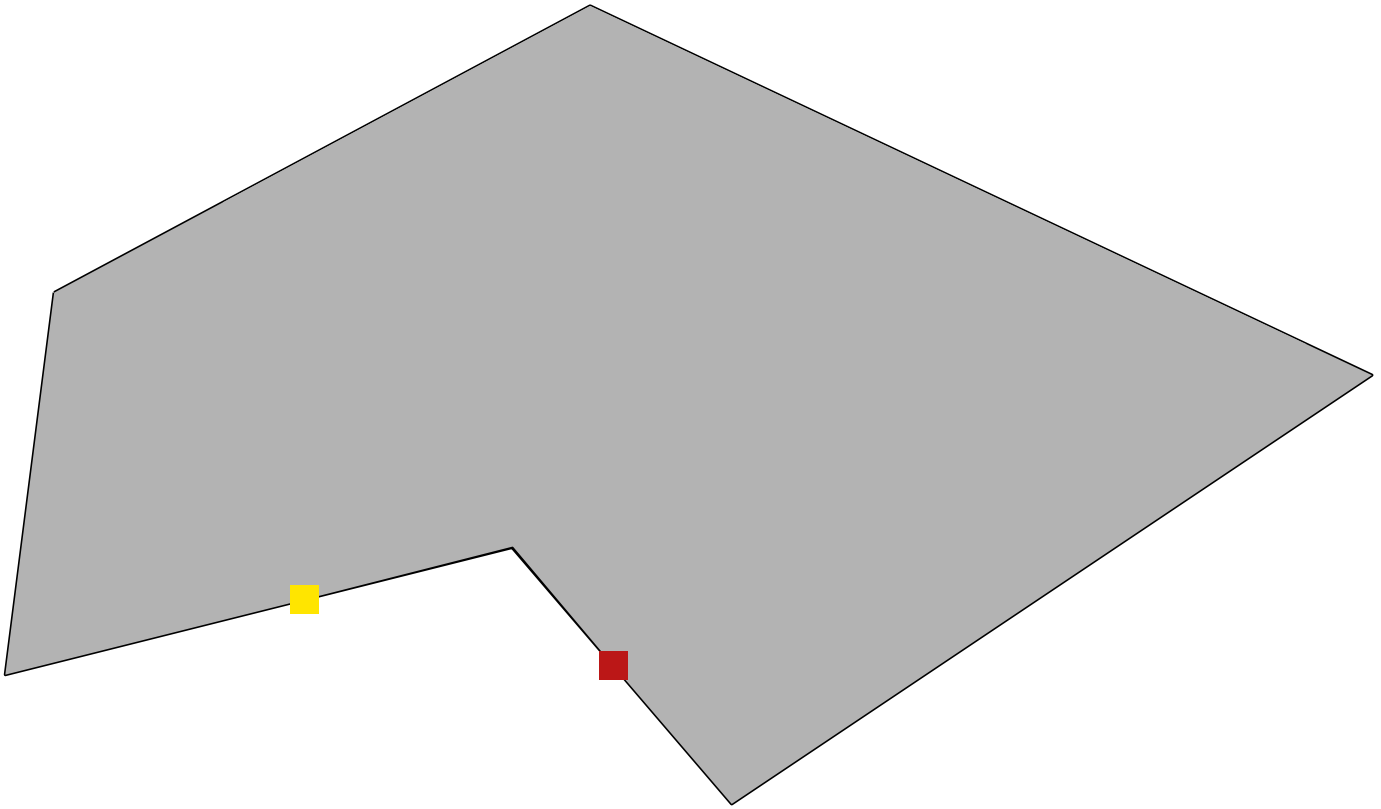
Aufgabe 3

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



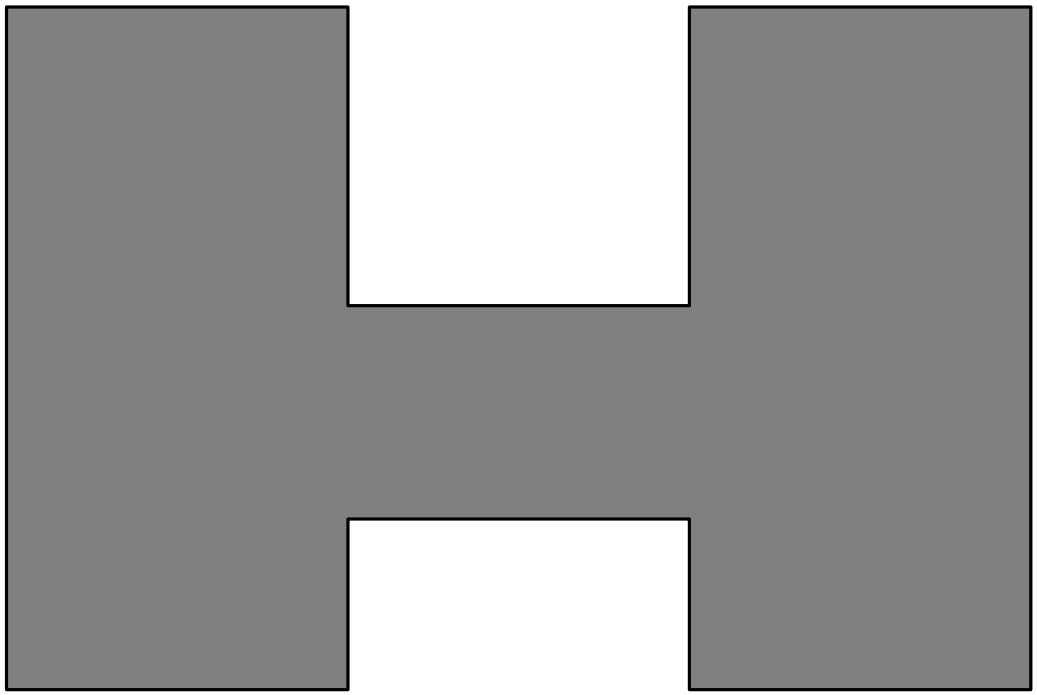
Aufgabe 3

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



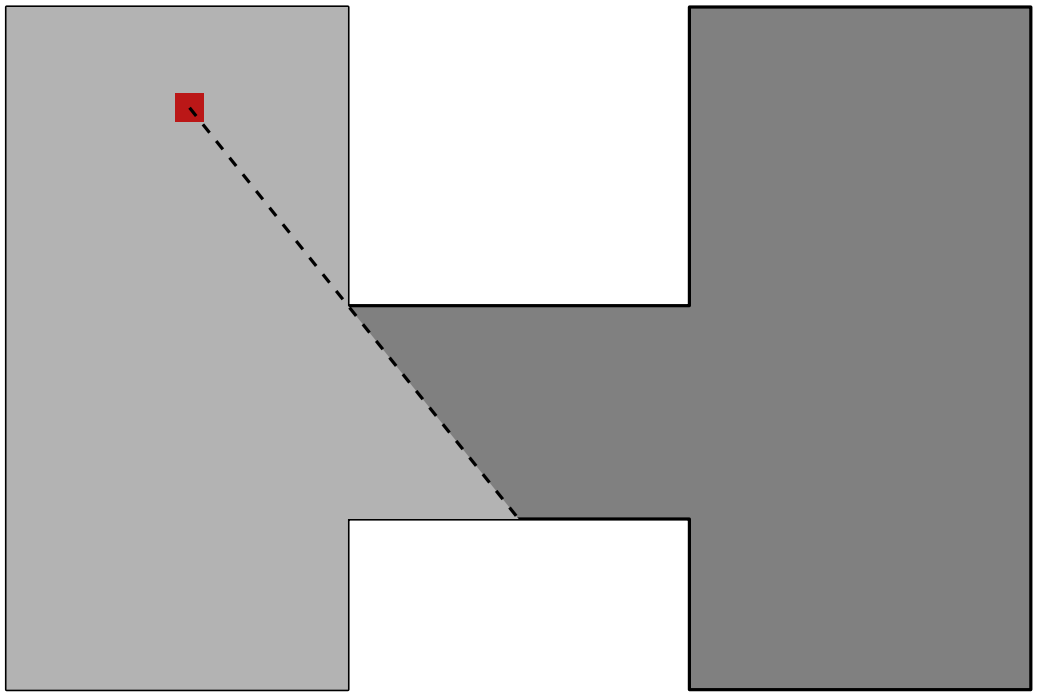
Aufgabe 3

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



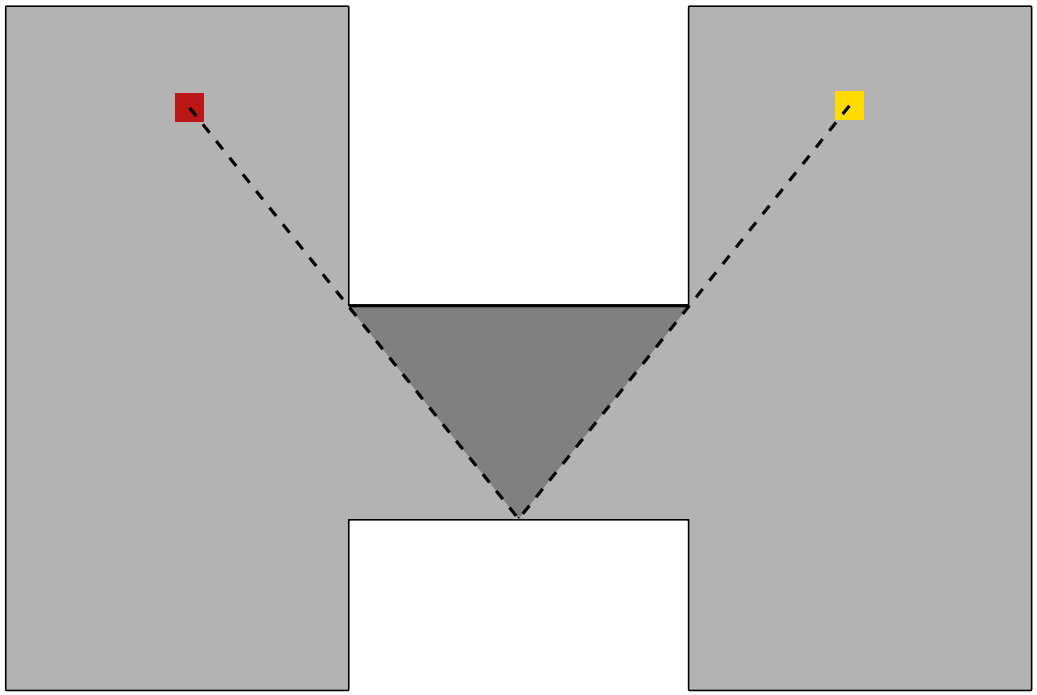
Aufgabe 3

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



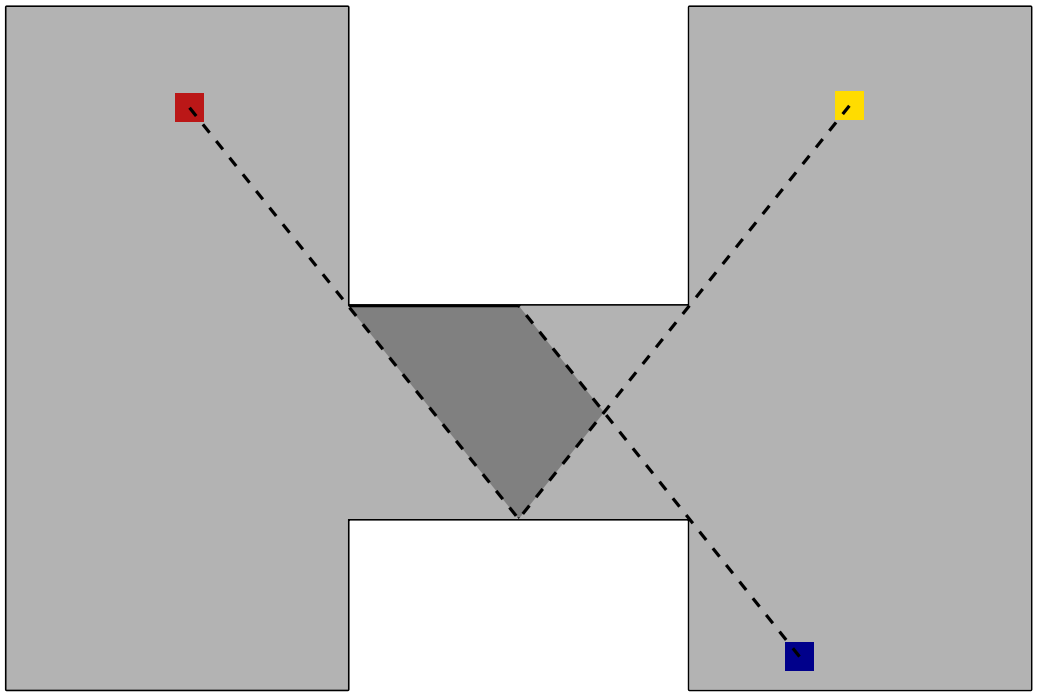
Aufgabe 3

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



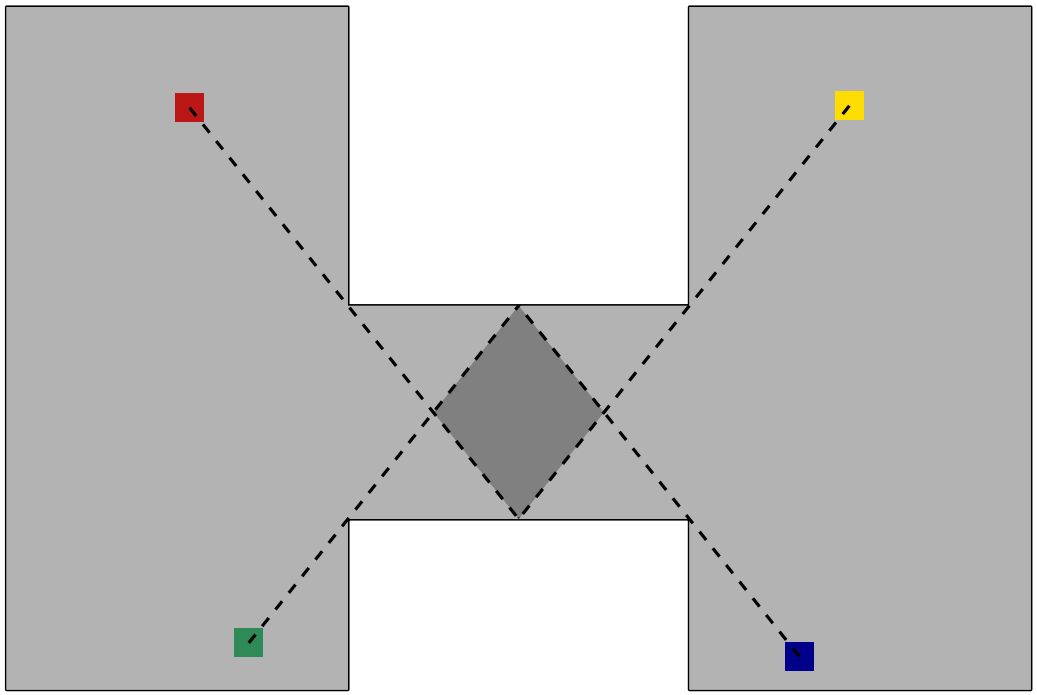
Aufgabe 3

$\mathcal{P}$ 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?

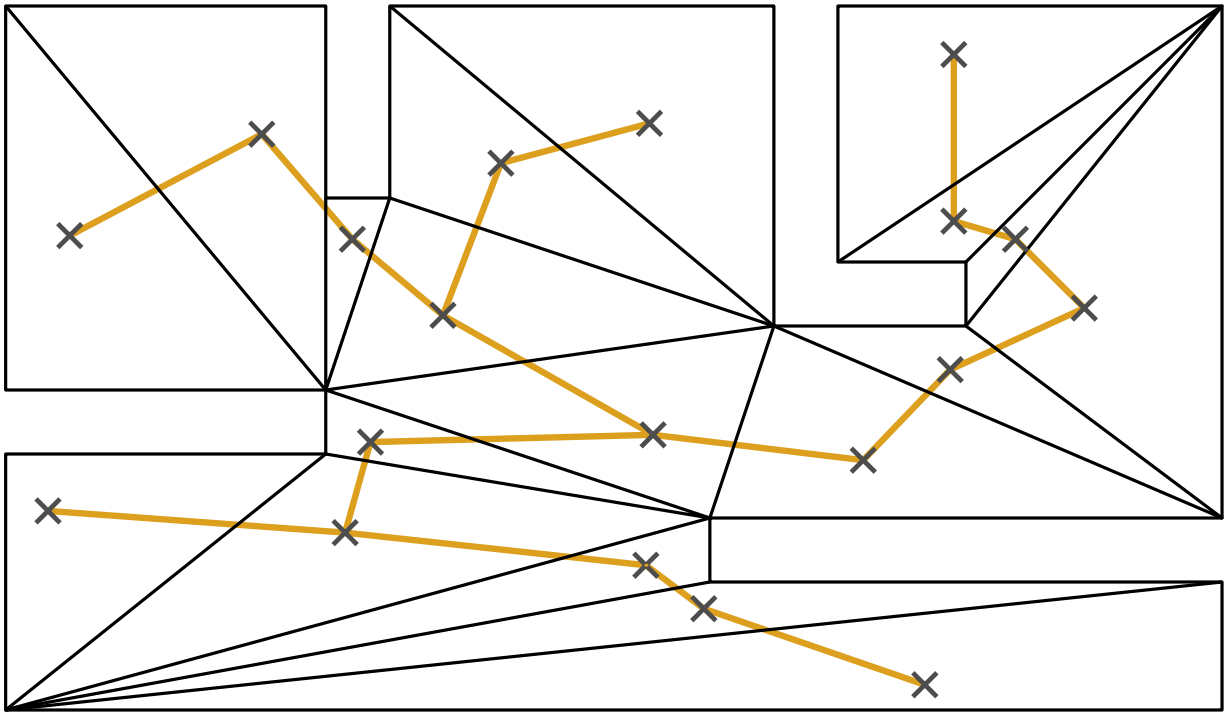


Aufgabe 4

Gegeben: Polygon P mit n Knoten.

Gesucht: $O(n \log n)$-Algorithmus, der P in zwei einfache Polygone auftrennt, sodass jedes höchstens $\lfloor 2n/3 \rfloor + 2$ Knoten besitzt.

Hinweis: Triangulieren Sie das Eingabe-Polygon und verwenden Sie dann den Dualgraphen dieser Triangulierung.



Aufgabe 4 – Lösung

Initialisierung: Alle Knoten $u \in V$ erhalten Gewicht $w(u) = 1$.

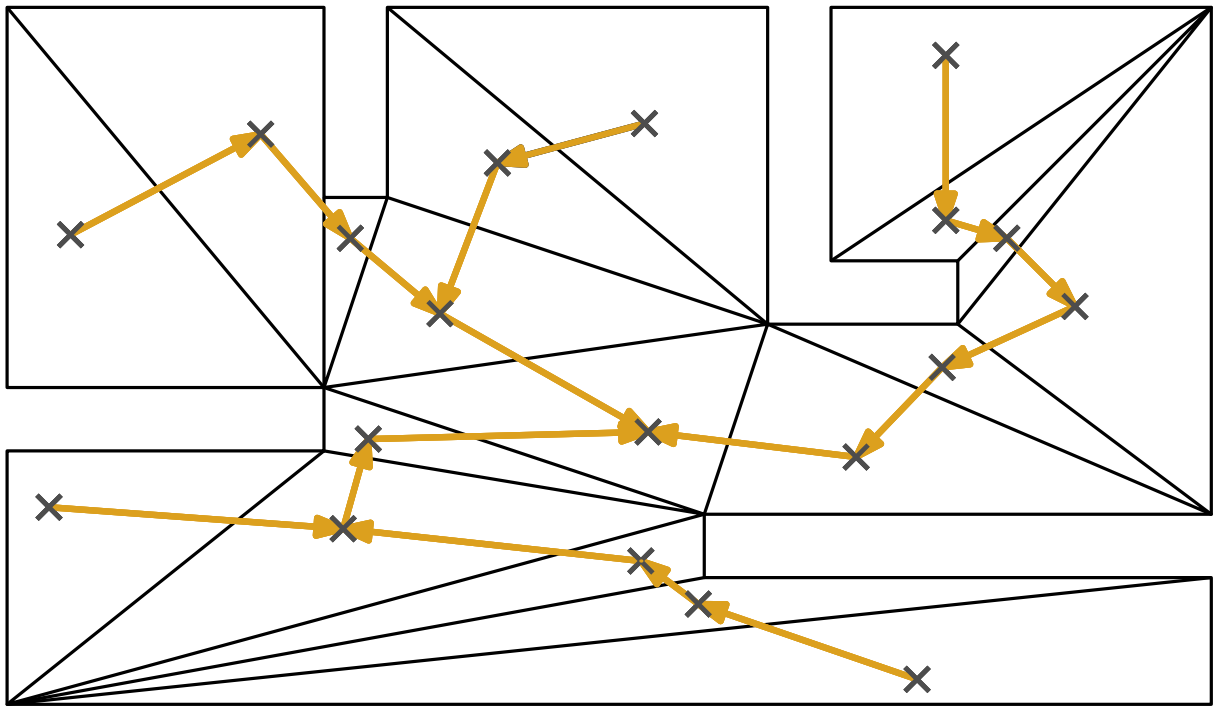
solange TRUE **tue**
- Sei u Blatt von T
- **solange** u hat Grad 1 **tue**
  - **wenn** $n - \lfloor 2n/3 \rfloor + 2 \leq w(u) \leq \lfloor 2n/3 \rfloor + 2$ **dann**
    - **return** Teilbaum von u indzuiert ges. Partitionierung.
  - $w(parent(u)) \leftarrow w(parent(u)) + w(u)$
  - Lösche u aus T
  - $u \leftarrow parent(u)$

$n = 19$

$n - \lfloor 2n/3 \rfloor + 2 = 5$

$\lfloor 2n/3 \rfloor + 2 = 14$

Annahme:
Baum hat Wurzel mit Grad $>= 2$.
Kanten sind zur Wurzel gerichtet.



Aufgabe 4 – Lösung

Initialisierung: Alle Knoten $u \in V$ erhalten Gewicht $w(u) = 1$.

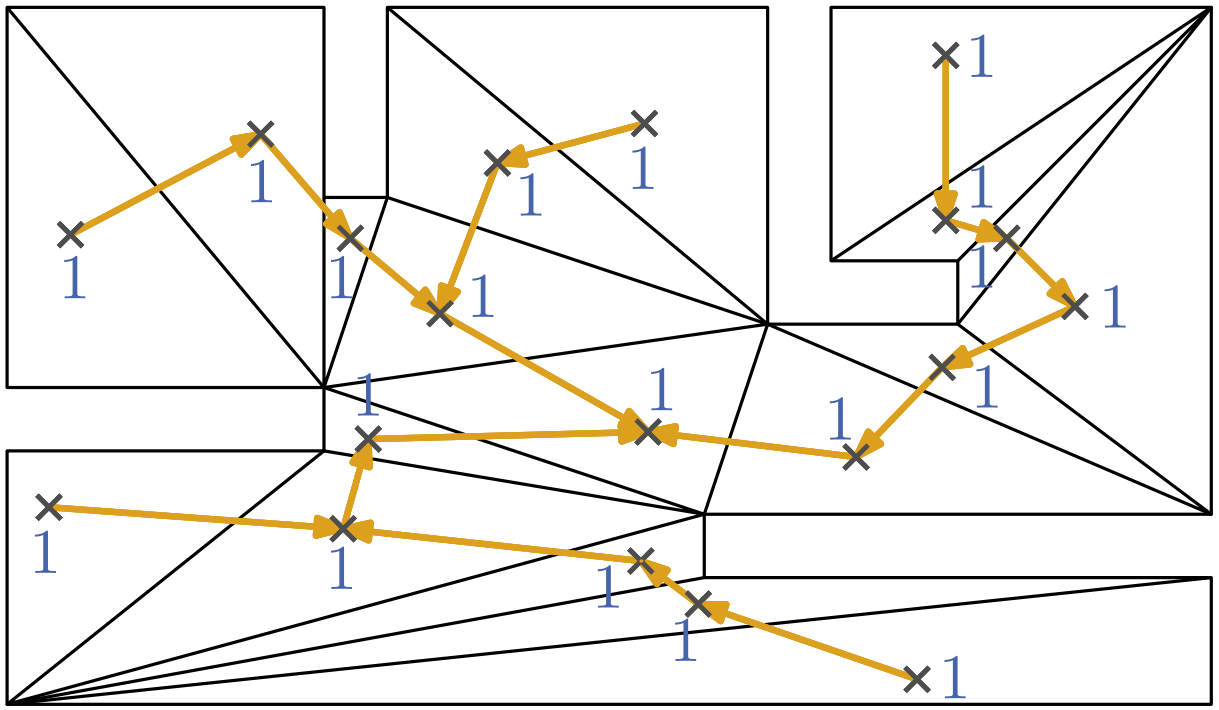
solange TRUE **tue**
- Sei u Blatt von T
- **solange** u hat Grad 1 **tue**
  - **wenn** $n - \lfloor 2n/3 \rfloor + 2 \leq w(u) \leq \lfloor 2n/3 \rfloor + 2$ **dann**
    - **return** Teilbaum von u indzuiert ges. Partitionierung.
  - $w(parent(u)) \leftarrow w(parent(u)) + w(u)$
  - Lösche u aus T
  - $u \leftarrow parent(u)$

$n = 19$

$n - \lfloor 2n/3 \rfloor + 2 = 5$

$\lfloor 2n/3 \rfloor + 2 = 14$

Annahme:
Baum hat Wurzel mit Grad $>= 2$.
Kanten sind zur Wurzel gerichtet.



Aufgabe 4 – Lösung

Initialisierung: Alle Knoten $u \in V$ erhalten Gewicht $w(u) = 1$.

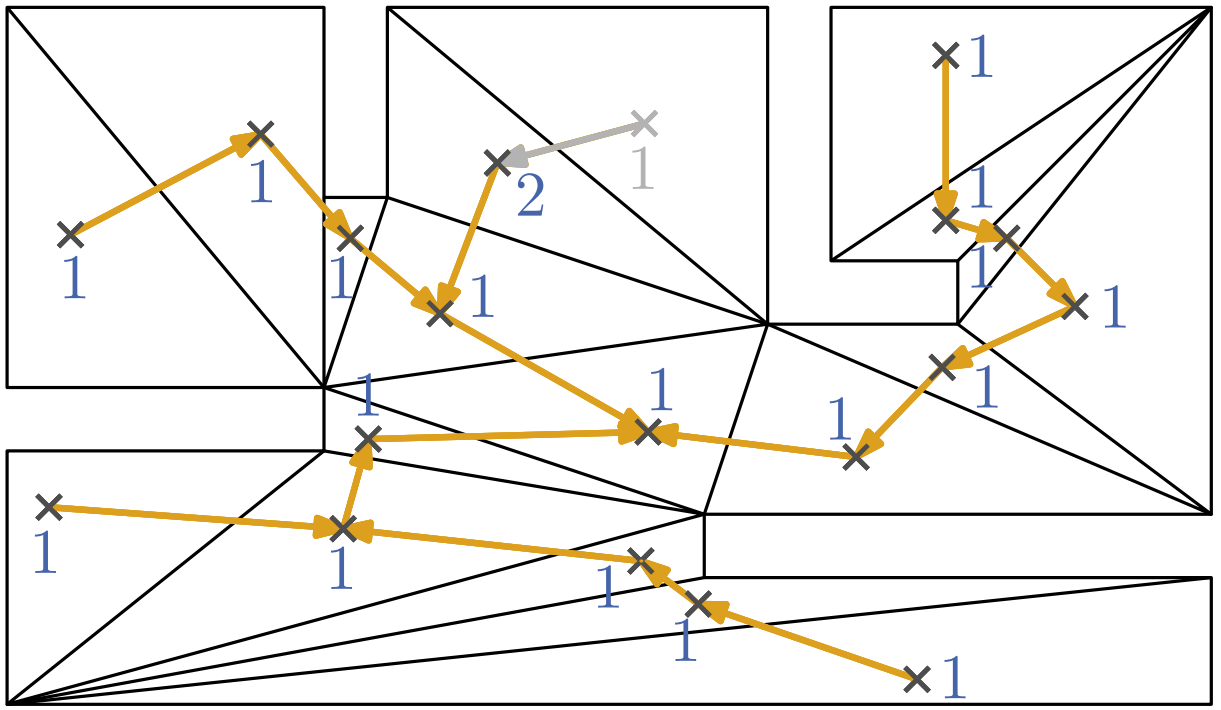
solange TRUE **tue**
- Sei u Blatt von T
- **solange** u hat Grad 1 **tue**
  - **wenn** $n - \lfloor 2n/3 \rfloor + 2 \leq w(u) \leq \lfloor 2n/3 \rfloor + 2$ **dann**
    - **return** Teilbaum von u indzuiert ges. Partitionierung.
  - $w(parent(u)) \leftarrow w(parent(u)) + w(u)$
  - Lösche u aus T
  - $u \leftarrow parent(u)$

$n = 19$

$n - \lfloor 2n/3 \rfloor + 2 = 5$

$\lfloor 2n/3 \rfloor + 2 = 14$

Annahme:
Baum hat Wurzel mit Grad $>= 2$.
Kanten sind zur Wurzel gerichtet.



Aufgabe 4 – Lösung

Initialisierung: Alle Knoten $u \in V$ erhalten Gewicht $w(u) = 1$.

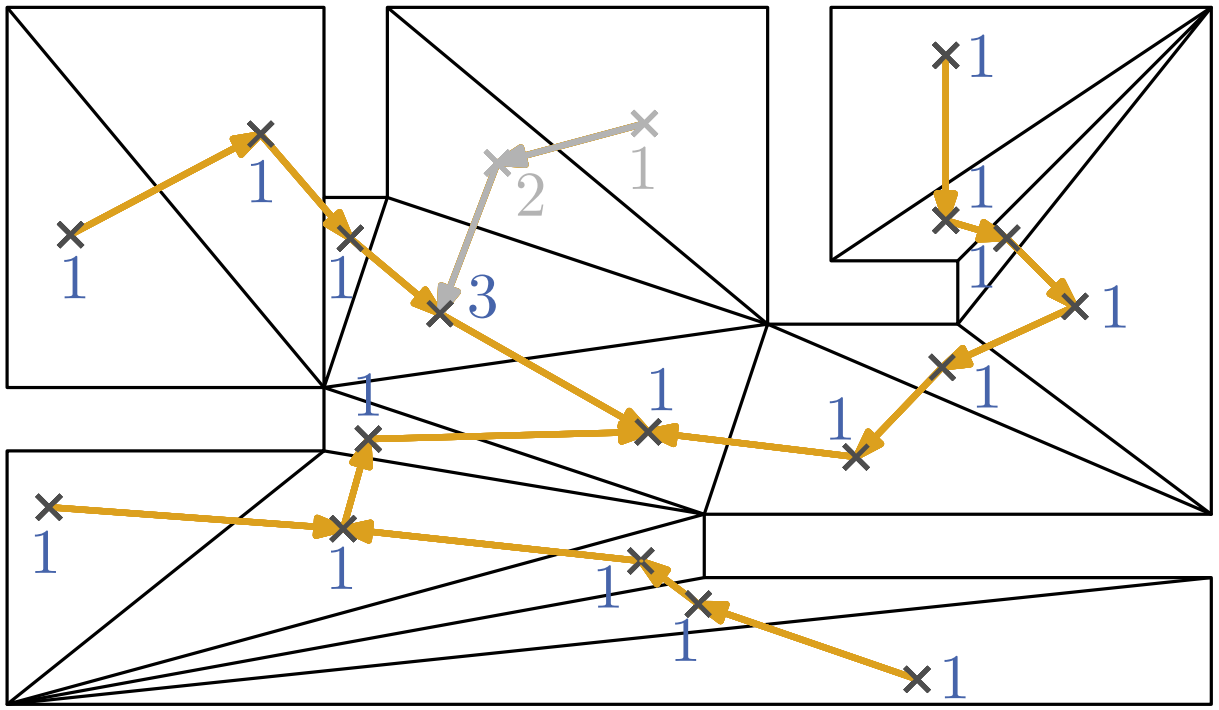
solange TRUE **tue**
- Sei u Blatt von T
- **solange** u hat Grad 1 **tue**
  - **wenn** $n - \lfloor 2n/3 \rfloor + 2 \leq w(u) \leq \lfloor 2n/3 \rfloor + 2$ **dann**
    - **return** Teilbaum von u indzuiert ges. Partitionierung.
  - $w(parent(u)) \leftarrow w(parent(u)) + w(u)$
  - Lösche u aus T
  - $u \leftarrow parent(u)$

$n = 19$

$n - \lfloor 2n/3 \rfloor + 2 = 5$

$\lfloor 2n/3 \rfloor + 2 = 14$

Annahme:
Baum hat Wurzel mit Grad $>= 2$.
Kanten sind zur Wurzel gerichtet.



Aufgabe 4 – Lösung

Initialisierung: Alle Knoten $u \in V$ erhalten Gewicht $w(u) = 1$.

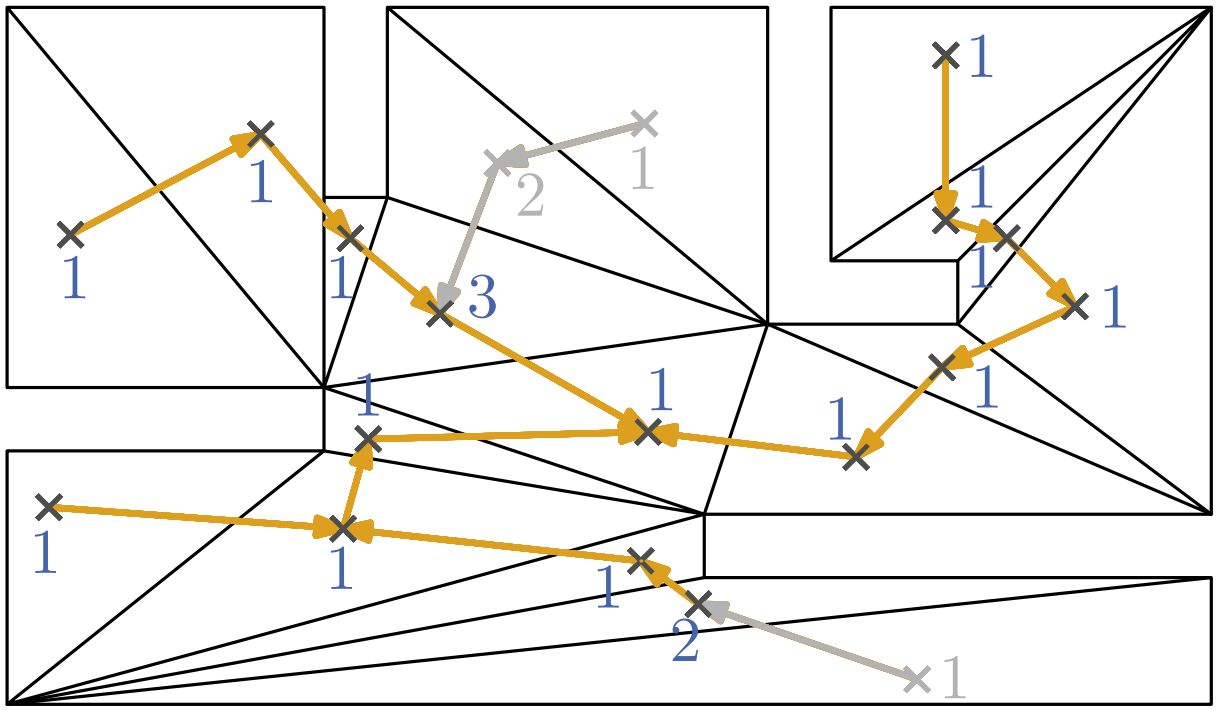
solange TRUE **tue**
- Sei u Blatt von T
- **solange** u hat Grad 1 **tue**
  - **wenn** $n - \lfloor 2n/3 \rfloor + 2 \leq w(u) \leq \lfloor 2n/3 \rfloor + 2$ **dann**
    - **return** Teilbaum von u indzuiert ges. Partitionierung.
  - $w(parent(u)) \leftarrow w(parent(u)) + w(u)$
  - Lösche u aus T
  - $u \leftarrow parent(u)$

$n = 19$

$n - \lfloor 2n/3 \rfloor + 2 = 5$

$\lfloor 2n/3 \rfloor + 2 = 14$

Annahme:
Baum hat Wurzel mit Grad $>= 2$.
Kanten sind zur Wurzel gerichtet.



Aufgabe 4 – Lösung

Initialisierung: Alle Knoten $u \in V$ erhalten Gewicht $w(u) = 1$.

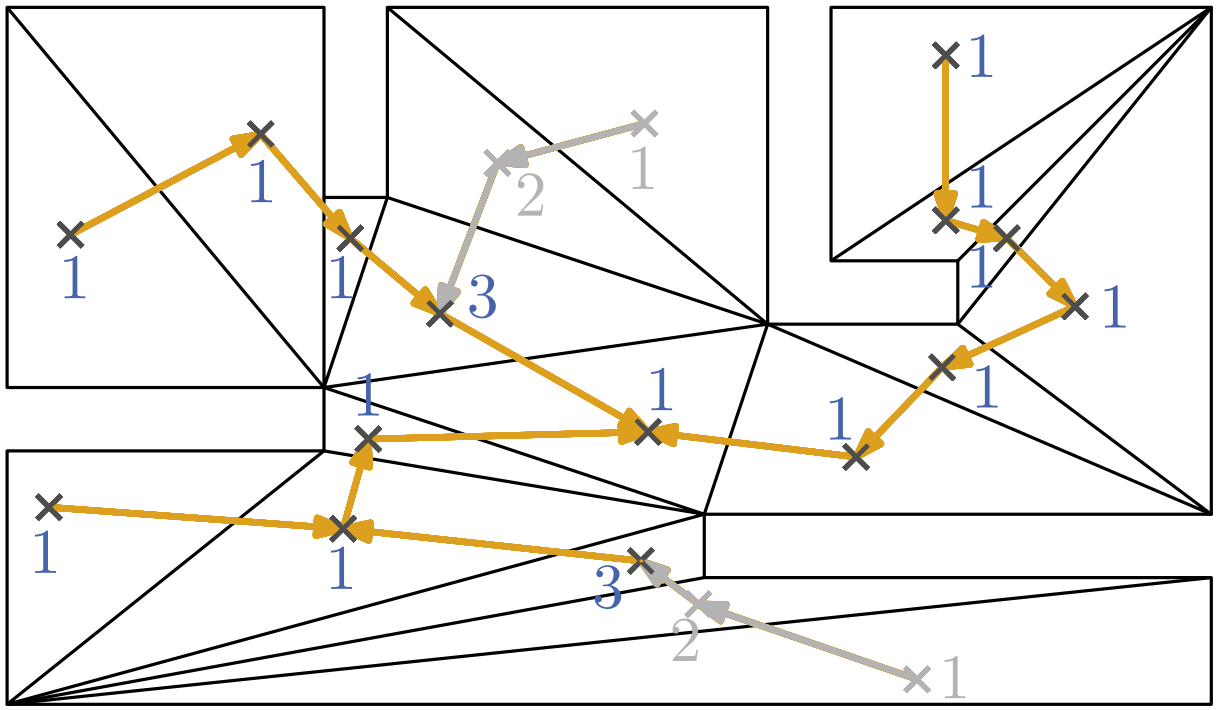
solange TRUE **tue**
- Sei u Blatt von T
- **solange** u hat Grad 1 **tue**
  - **wenn** $n - \lfloor 2n/3 \rfloor + 2 \leq w(u) \leq \lfloor 2n/3 \rfloor + 2$ **dann**
    - **return** Teilbaum von u indzuiert ges. Partitionierung.
  - $w(parent(u)) \leftarrow w(parent(u)) + w(u)$
  - Lösche u aus T
  - $u \leftarrow parent(u)$

$n = 19$

$n - \lfloor 2n/3 \rfloor + 2 = 5$

$\lfloor 2n/3 \rfloor + 2 = 14$

Annahme:
Baum hat Wurzel mit Grad $>= 2$.
Kanten sind zur Wurzel gerichtet.



Aufgabe 4 – Lösung

Initialisierung: Alle Knoten $u \in V$ erhalten Gewicht $w(u) = 1$.

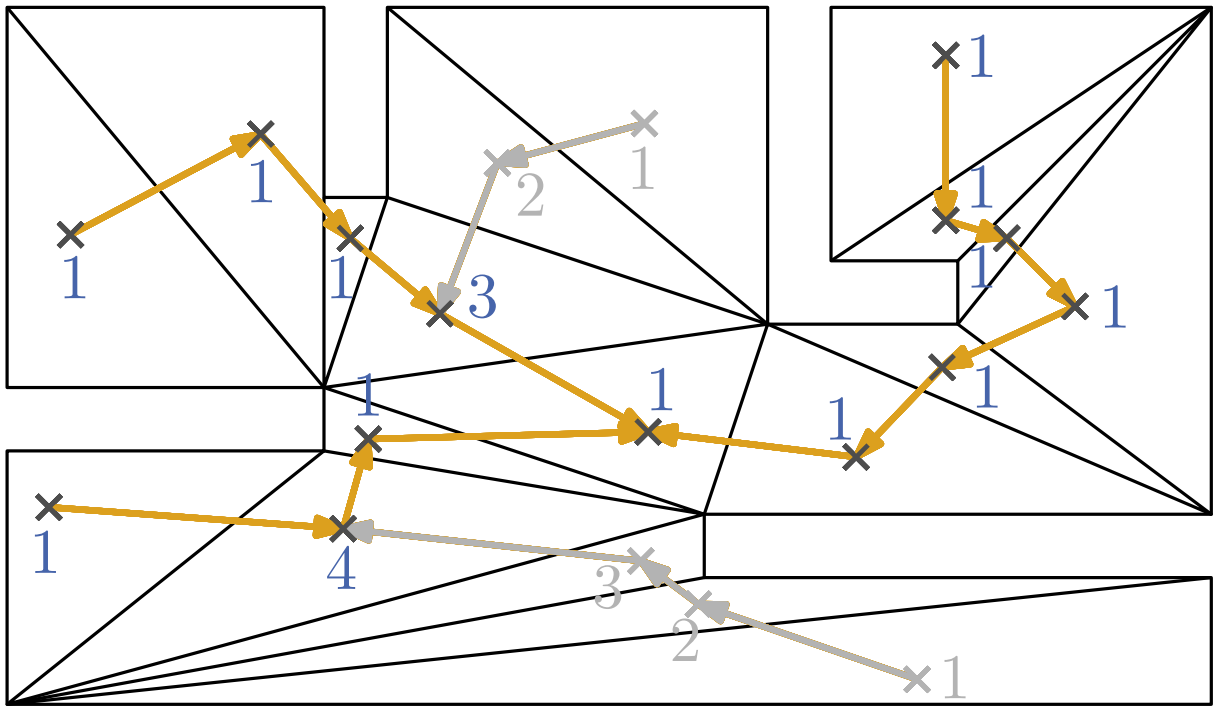
solange TRUE **tue**
- Sei u Blatt von T
- **solange** u hat Grad 1 **tue**
  - **wenn** $n - \lfloor 2n/3 \rfloor + 2 \leq w(u) \leq \lfloor 2n/3 \rfloor + 2$ **dann**
    - **return** Teilbaum von u indzuiert ges. Partitionierung.
  - $w(parent(u)) \leftarrow w(parent(u)) + w(u)$
  - Lösche u aus T
  - $u \leftarrow parent(u)$

$n = 19$

$n - \lfloor 2n/3 \rfloor + 2 = 5$

$\lfloor 2n/3 \rfloor + 2 = 14$

Annahme:
Baum hat Wurzel mit Grad $>= 2$.
Kanten sind zur Wurzel gerichtet.



Aufgabe 4 – Lösung

Initialisierung: Alle Knoten $u \in V$ erhalten Gewicht $w(u) = 1$.

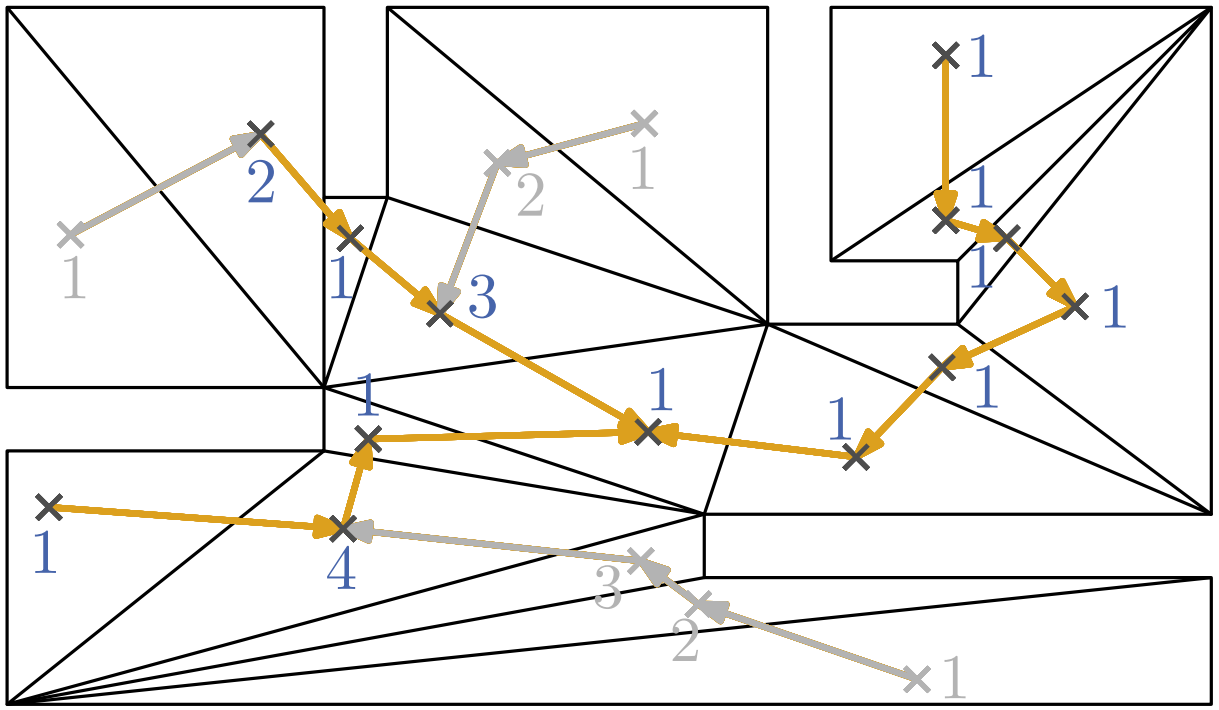
solange TRUE **tue**
- Sei u Blatt von T
- **solange** u hat Grad 1 **tue**
  - **wenn** $n - \lfloor 2n/3 \rfloor + 2 \leq w(u) \leq \lfloor 2n/3 \rfloor + 2$ **dann**
    - **return** Teilbaum von u indzuiert ges. Partitionierung.
  - $w(parent(u)) \leftarrow w(parent(u)) + w(u)$
  - Lösche u aus T
  - $u \leftarrow parent(u)$

$n = 19$

$n - \lfloor 2n/3 \rfloor + 2 = 5$

$\lfloor 2n/3 \rfloor + 2 = 14$

Annahme:
Baum hat Wurzel mit Grad $>= 2$.
Kanten sind zur Wurzel gerichtet.



Aufgabe 4 – Lösung

Initialisierung: Alle Knoten $u \in V$ erhalten Gewicht $w(u) = 1$.

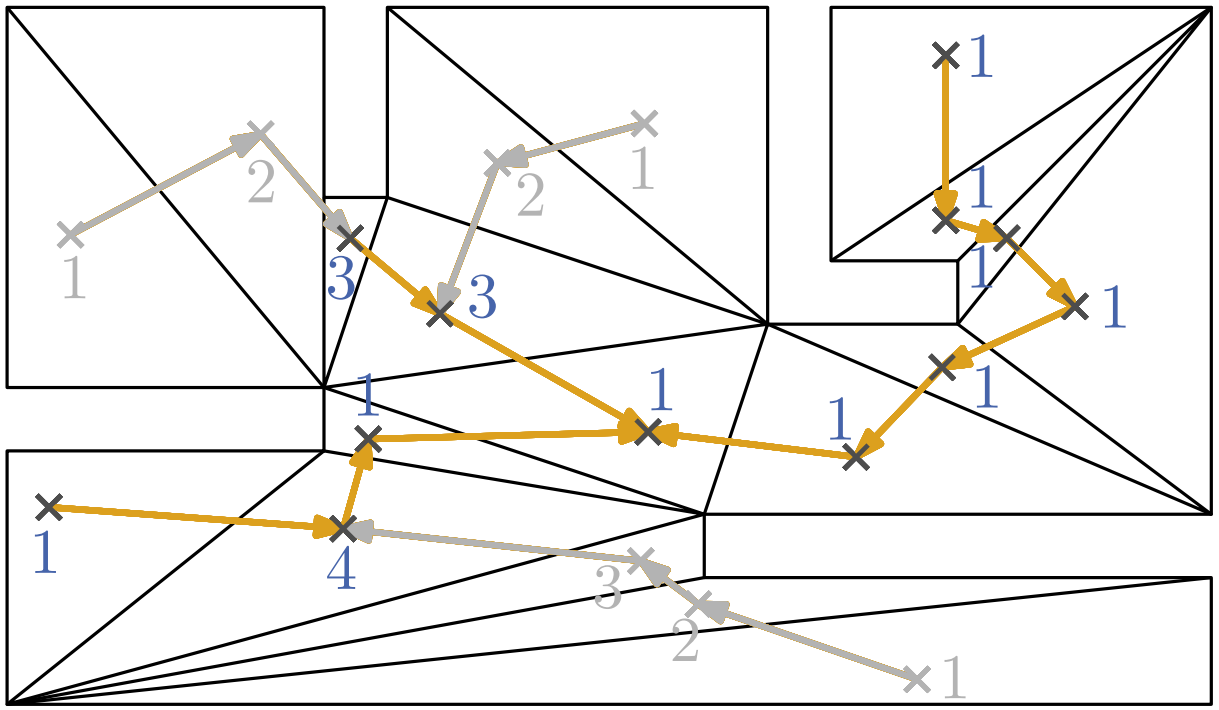
solange TRUE **tue**
- Sei u Blatt von T
- **solange** u hat Grad 1 **tue**
  - **wenn** $n - \lfloor 2n/3 \rfloor + 2 \leq w(u) \leq \lfloor 2n/3 \rfloor + 2$ **dann**
    - **return** Teilbaum von u indzuiert ges. Partitionierung.
  - $w(parent(u)) \leftarrow w(parent(u)) + w(u)$
  - Lösche u aus T
  - $u \leftarrow parent(u)$

$n = 19$

$n - \lfloor 2n/3 \rfloor + 2 = 5$

$\lfloor 2n/3 \rfloor + 2 = 14$

Annahme:
Baum hat Wurzel mit Grad $>= 2$.
Kanten sind zur Wurzel gerichtet.



Aufgabe 4 – Lösung

Initialisierung: Alle Knoten $u \in V$ erhalten Gewicht $w(u) = 1$.

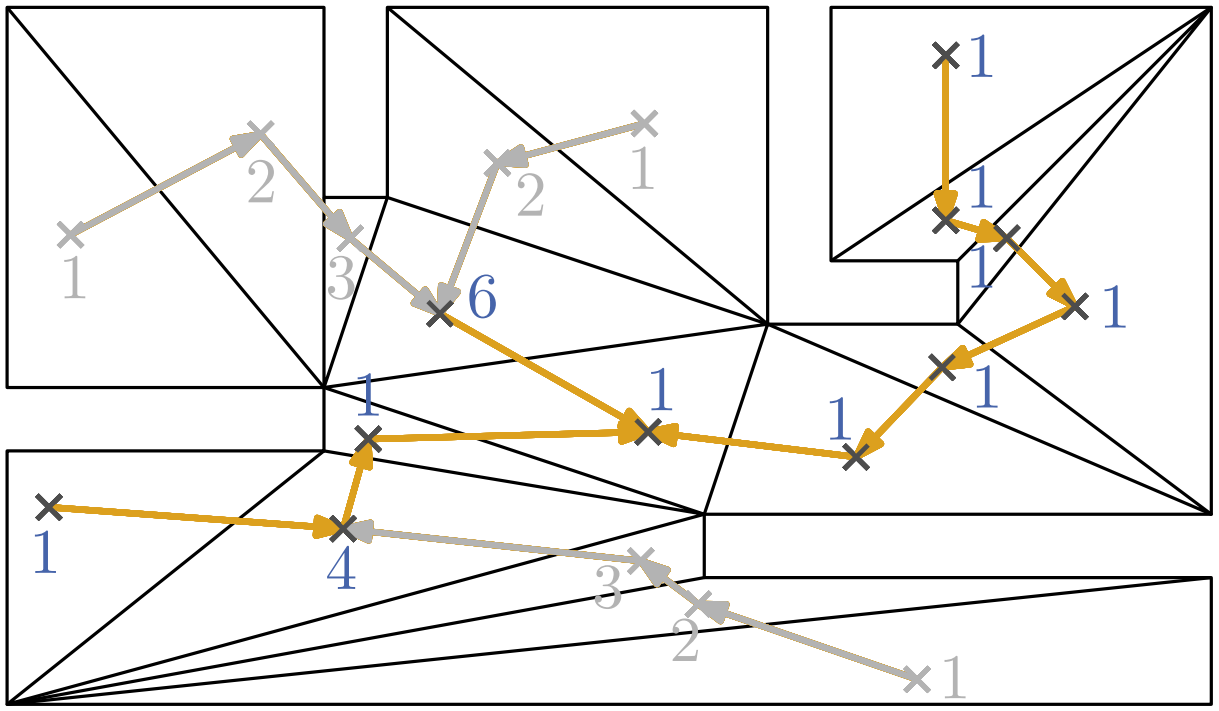
solange TRUE **tue**
- Sei u Blatt von T
- **solange** u hat Grad 1 **tue**
  - **wenn** $n - \lfloor 2n/3 \rfloor + 2 \leq w(u) \leq \lfloor 2n/3 \rfloor + 2$ **dann**
    - **return** Teilbaum von u indzuiert ges. Partitionierung.
  - $w(parent(u)) \leftarrow w(parent(u)) + w(u)$
  - Lösche u aus T
  - $u \leftarrow parent(u)$

$n = 19$

$n - \lfloor 2n/3 \rfloor + 2 = 5$

$\lfloor 2n/3 \rfloor + 2 = 14$

Annahme:
Baum hat Wurzel mit Grad $>= 2$.
Kanten sind zur Wurzel gerichtet.



Aufgabe 4 – Lösung

Initialisierung: Alle Knoten $u \in V$ erhalten Gewicht $w(u) = 1$.

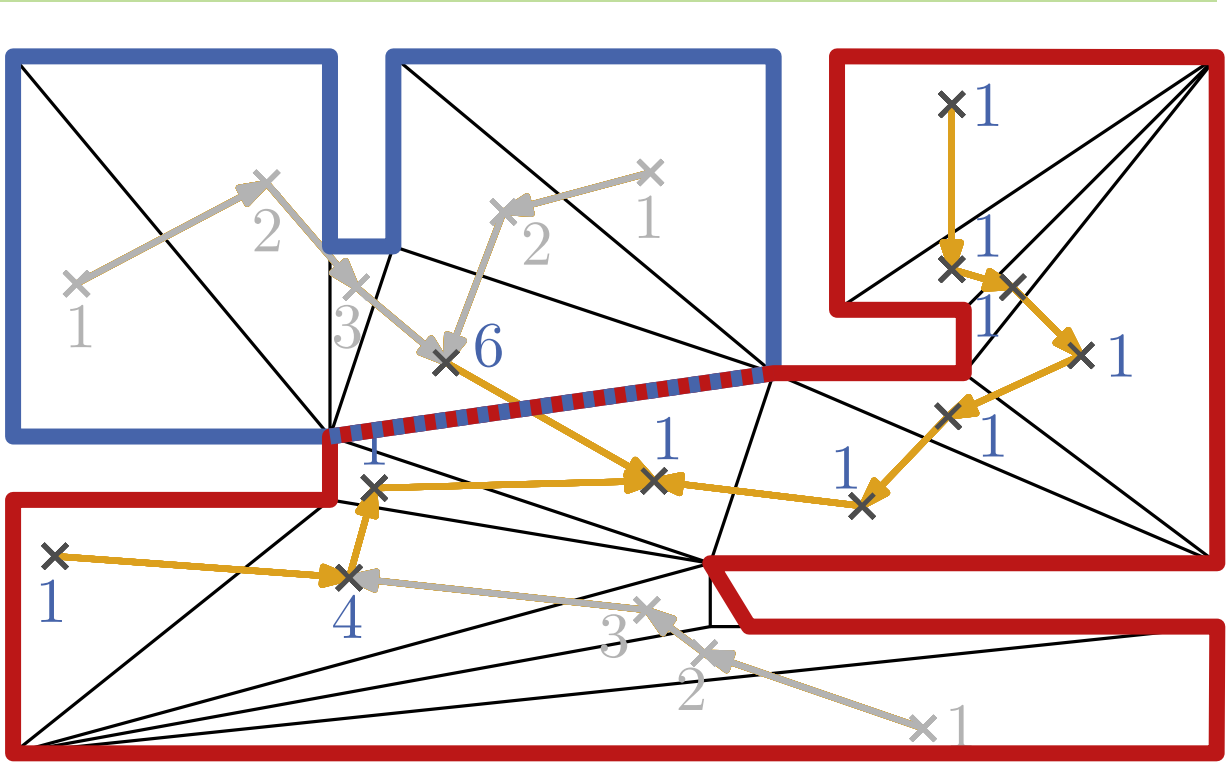
solange TRUE **tue**
- Sei u Blatt von T
- **solange** u hat Grad 1 **tue**
  - **wenn** $n - \lfloor 2n/3 \rfloor + 2 \leq w(u) \leq \lfloor 2n/3 \rfloor + 2$ **dann**
    - **return** Teilbaum von u indzuiert ges. Partitionierung.
  - $w(parent(u)) \leftarrow w(parent(u)) + w(u)$
  - Lösche u aus T
  - $u \leftarrow parent(u)$

$n = 19$

$n - \lfloor 2n/3 \rfloor + 2 = 5$

$\lfloor 2n/3 \rfloor + 2 = 14$

Annahme:
Baum hat Wurzel mit Grad $>= 2$.
Kanten sind zur Wurzel gerichtet.

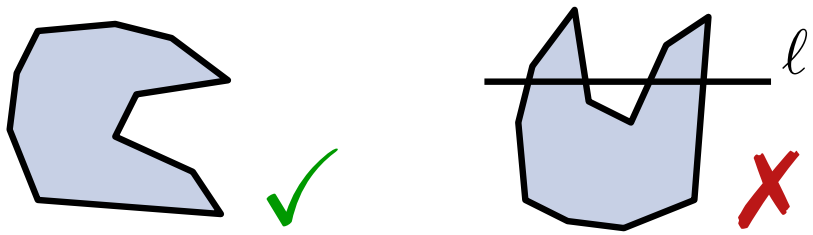


Triangulierung: Überblick

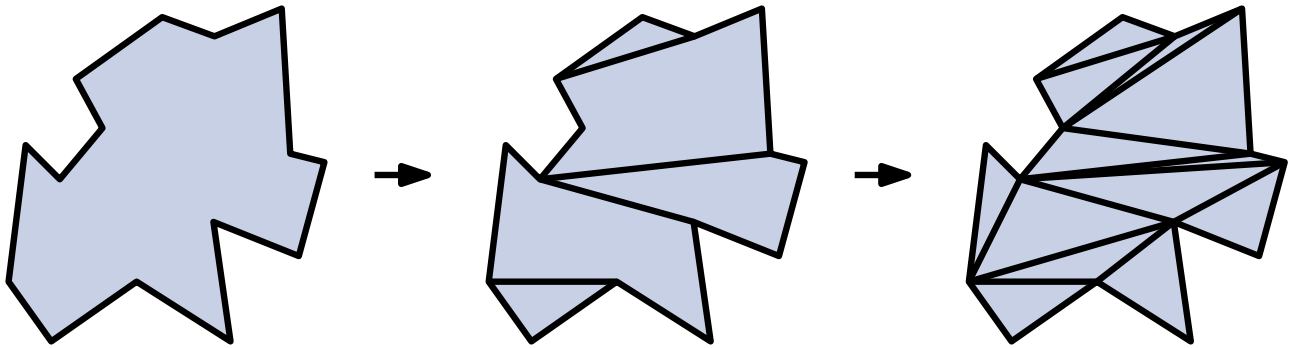
Zweistufiges Verfahren:

- Schritt 1: Zerlege P in y-monotone Teilpolygone

  Definition: Ein Polygon P ist y-monoton, falls der Schnitt $\ell \cap P$ für jede horizontale Gerade ℓ zusammenhängend ist.

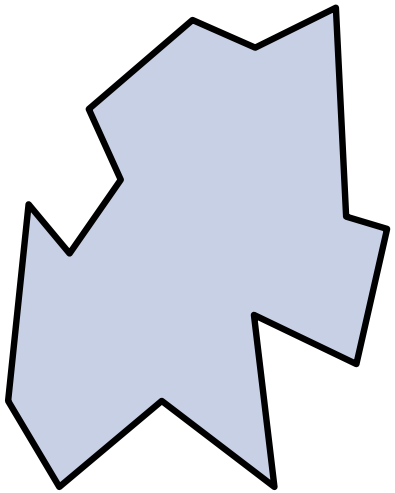


- Schritt 2: Trianguliere y-monotone Teilpolygone



Zerlegen in y-monotone Teile

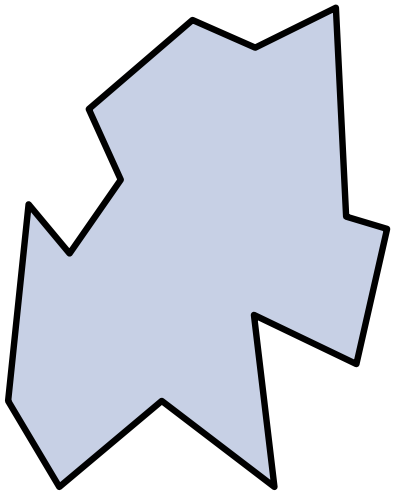
Idee: Unterscheide fünf verschiedene Knotenarten



Zerlegen in y-monotone Teile

Idee: Unterscheide fünf verschiedene Knotenarten

– *Wendeknoten*:

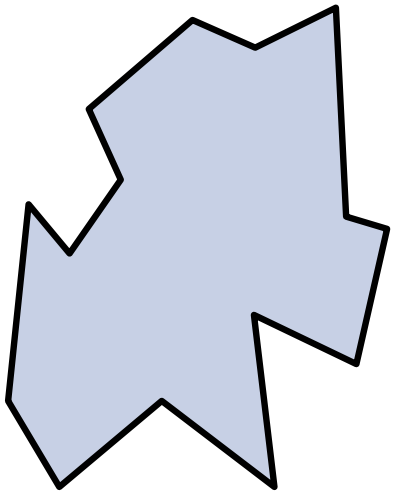


– *reguläre* Knoten

Zerlegen in y-monotone Teile

Idee: Unterscheide fünf verschiedene Knotenarten

– *Wendeknoten*:
vertikale Laufrichtung wechselt



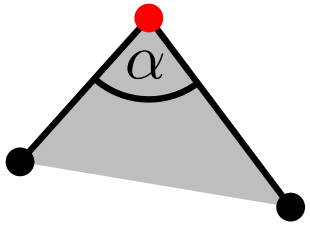
– *reguläre* Knoten

Zerlegen in y-monotone Teile

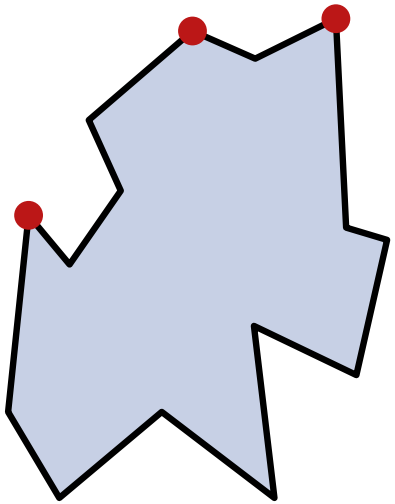
Idee: Unterscheide fünf verschiedene Knotenarten

– *Wendeknoten*:
vertikale Laufrichtung wechselt

- *Startknoten*



falls $\alpha < 180°$



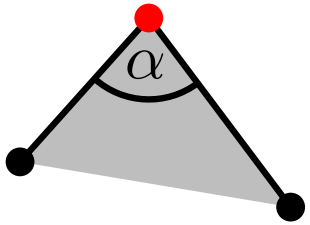
– *reguläre* Knoten

Zerlegen in y-monotone Teile

Idee: Unterscheide fünf verschiedene Knotenarten

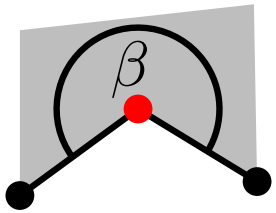
– *Wendeknoten*:
vertikale Laufrichtung wechselt

- *Startknoten*

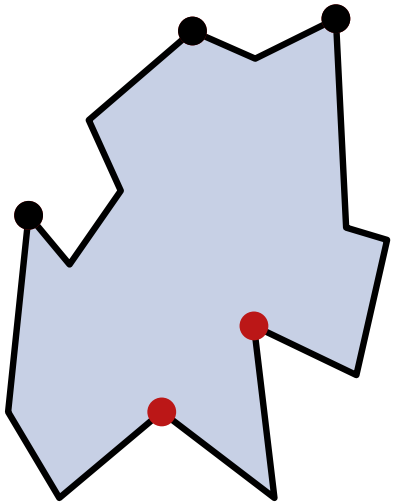


falls $\alpha < 180°$

- *Splitknoten*



falls $\beta > 180°$



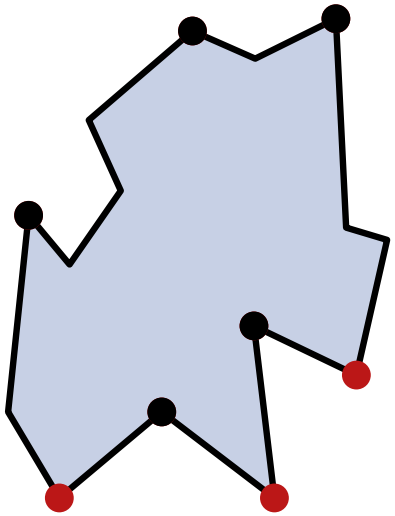
– *reguläre* Knoten

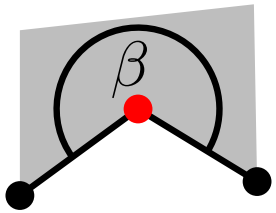
Zerlegen in y-monotone Teile

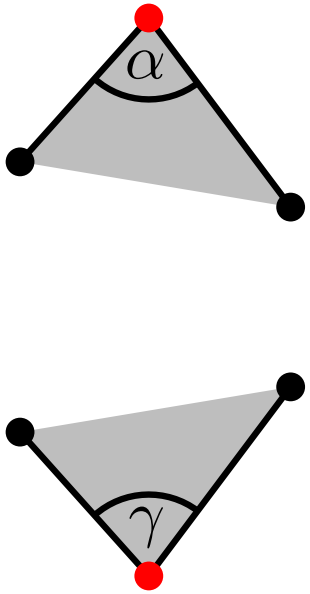
Idee: Unterscheide fünf verschiedene Knotenarten

– *Wendeknoten*:
vertikale Laufrichtung wechselt

- *Startknoten*
- *Splitknoten*
- *Endknoten*







falls $\alpha < 180°$

falls $\beta > 180°$

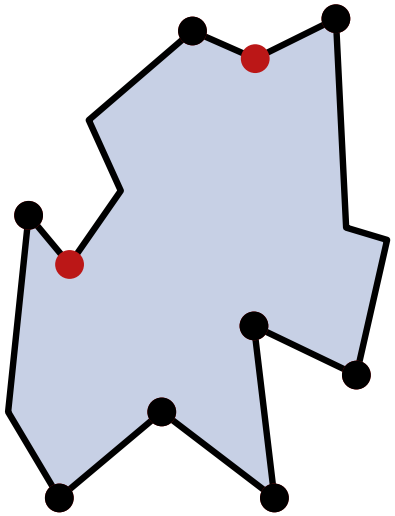
falls $\gamma < 180°$

– *reguläre* Knoten

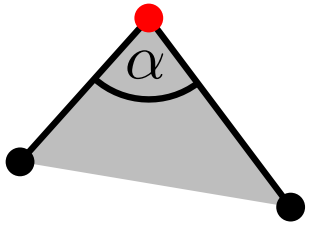
Zerlegen in y-monotone Teile

Idee: Unterscheide fünf verschiedene Knotenarten

– *Wendeknoten*:
vertikale Laufrichtung wechselt

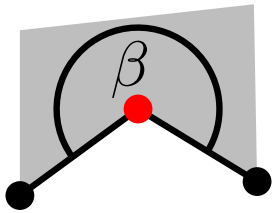


- *Startknoten*



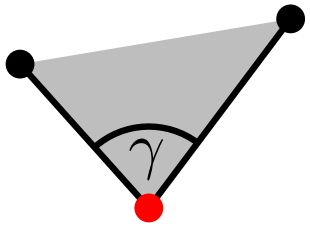
falls $\alpha < 180°$

- *Splitknoten*



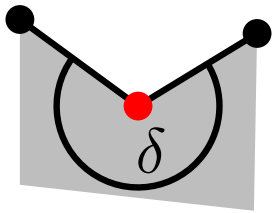
falls $\beta > 180°$

- *Endknoten*



falls $\gamma < 180°$

- *Mergeknoten*



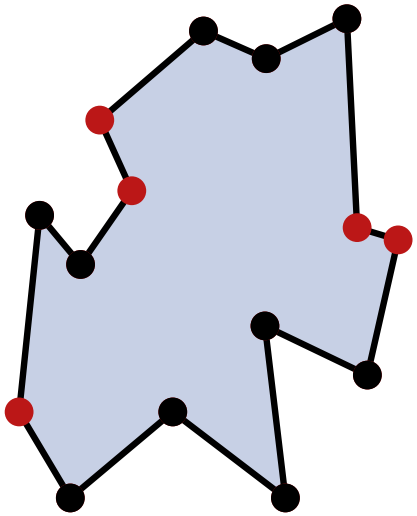
falls $\delta > 180°$

– *reguläre* Knoten

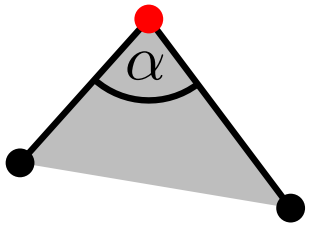
Zerlegen in y-monotone Teile

Idee: Unterscheide fünf verschiedene Knotenarten

– *Wendeknoten*:
vertikale Laufrichtung wechselt

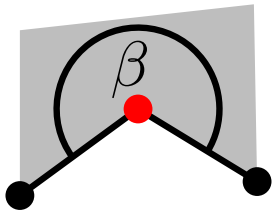


- *Startknoten*



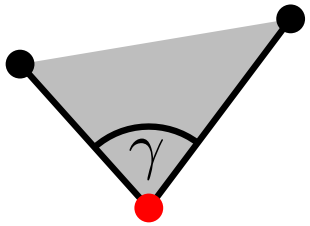
falls $\alpha < 180°$

- *Splitknoten*



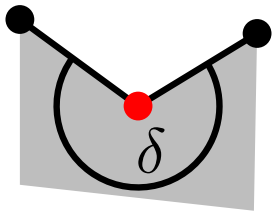
falls $\beta > 180°$

- *Endknoten*



falls $\gamma < 180°$

- *Mergeknoten*



falls $\delta > 180°$

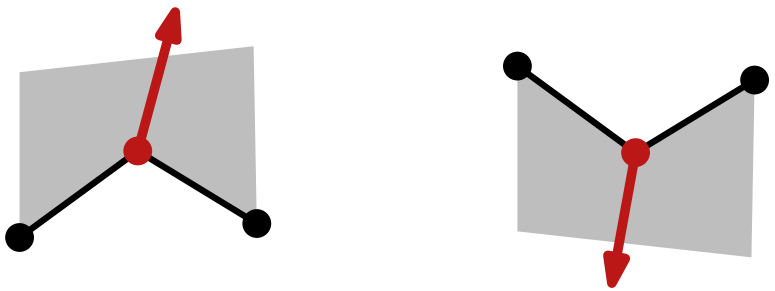
– *reguläre* Knoten

Charakterisierung

Lemma 1: Ein Polygon ist y-monoton, wenn es keine Split- oder Mergeknoten besitzt.

an der Tafel

$\Rightarrow$ Wir müssen alle Split- und Mergeknoten durch Einfügen von Diagonalen entfernen

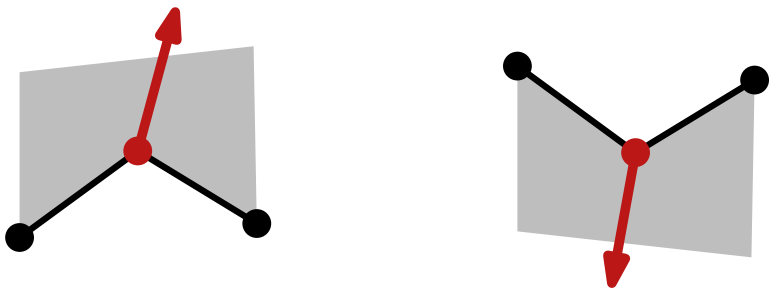


Charakterisierung

Lemma 1: Ein Polygon ist y-monoton, wenn es keine Split- oder Mergeknoten besitzt.

an der Tafel

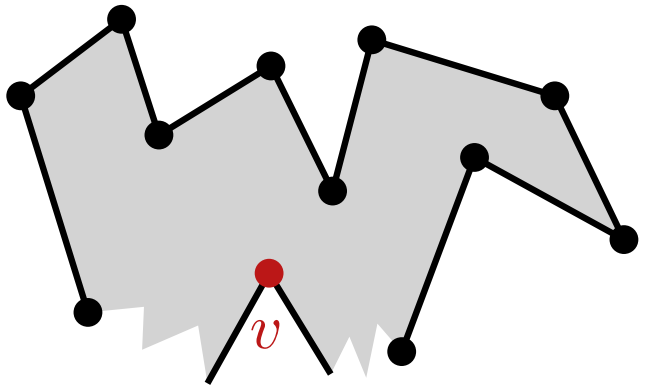
$\Rightarrow$ Wir müssen alle Split- und Mergeknoten durch Einfügen von Diagonalen entfernen



Vorsicht: Diagonalen dürfen weder Kanten von P noch andere Diagonalen schneiden

In Richtung Sweep-Line-Algorithmus

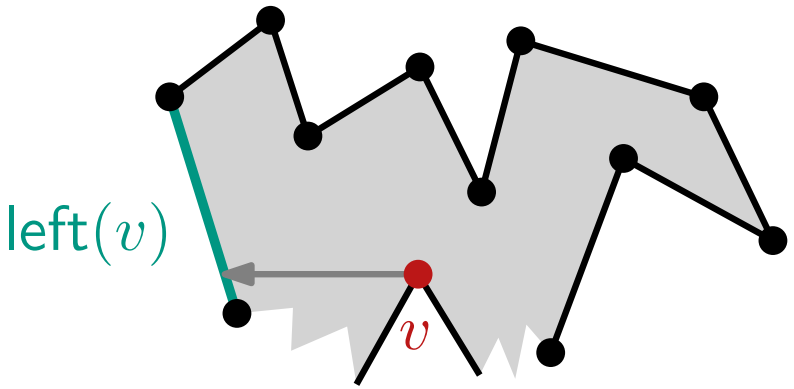
1) Diagonalen für Splitknoten



In Richtung Sweep-Line-Algorithmus

1) Diagonalen für Splitknoten

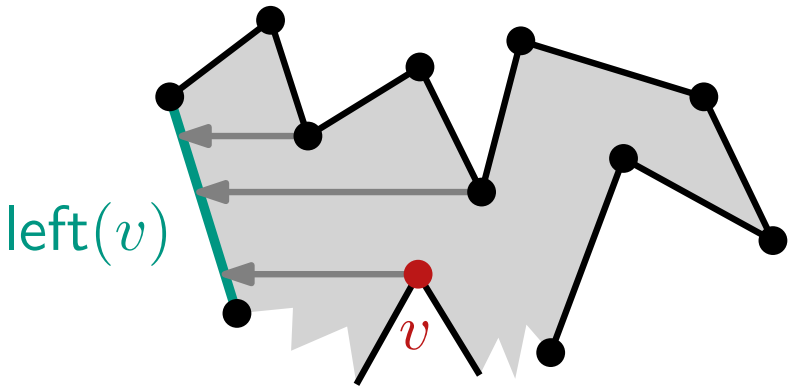
- betrachte für jeden Knoten v linke Nachbarkante $\text{left}(v)$ bzgl. horizontaler sweep line ℓ



In Richtung Sweep-Line-Algorithmus

1) Diagonalen für Splitknoten

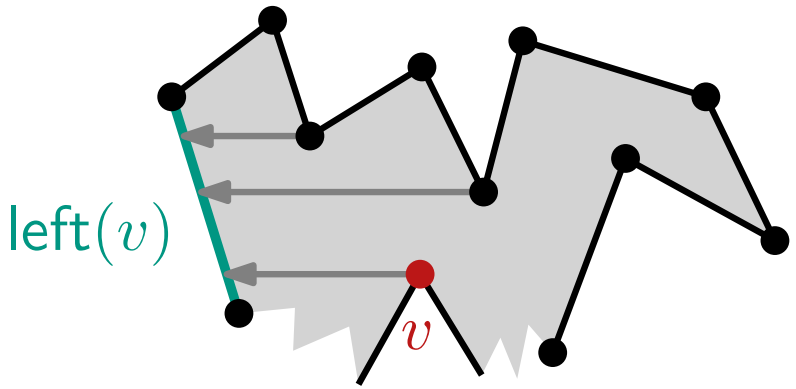
- betrachte für jeden Knoten v linke Nachbarkante $\text{left}(v)$ bzgl. horizontaler sweep line ℓ



In Richtung Sweep-Line-Algorithmus

1) Diagonalen für Splitknoten

- betrachte für jeden Knoten v linke Nachbarkante $\text{left}(v)$ bzgl. horizontaler sweep line ℓ

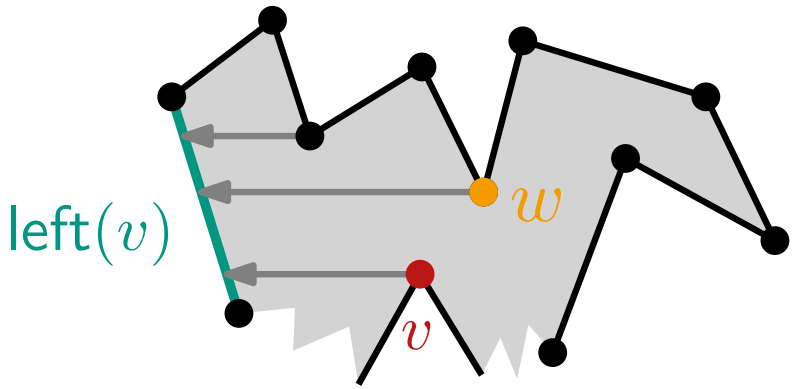


- verbinde Splitknoten v zu niedrigstem Knoten w oberhalb v mit $\text{left}(w) = \text{left}(v)$

In Richtung Sweep-Line-Algorithmus

1) Diagonalen für Splitknoten

- betrachte für jeden Knoten v linke Nachbarkante $\text{left}(v)$ bzgl. horizontaler sweep line ℓ

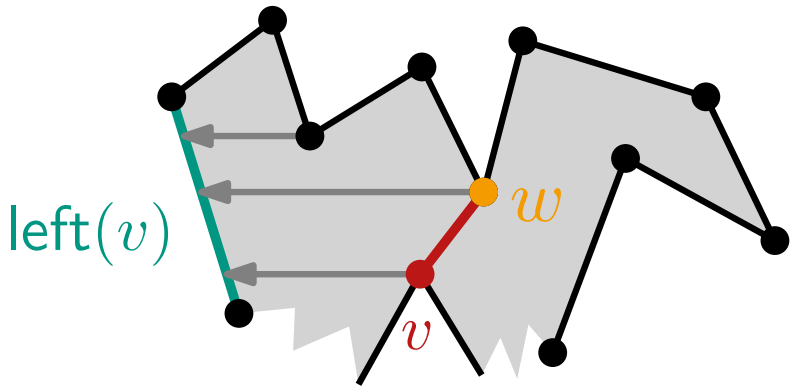


- verbinde Splitknoten v zu niedrigstem Knoten w oberhalb v mit $\text{left}(w) = \text{left}(v)$

In Richtung Sweep-Line-Algorithmus

1) Diagonalen für Splitknoten

- betrachte für jeden Knoten v linke Nachbarkante $\text{left}(v)$ bzgl. horizontaler sweep line ℓ

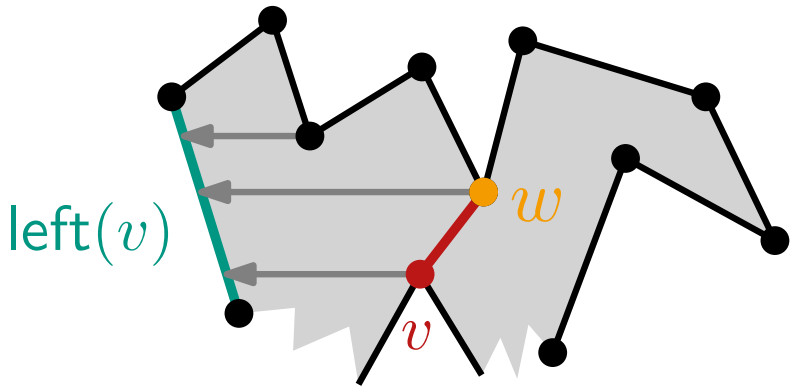


- verbinde Splitknoten v zu niedrigstem Knoten w oberhalb v mit $\text{left}(w) = \text{left}(v)$

In Richtung Sweep-Line-Algorithmus

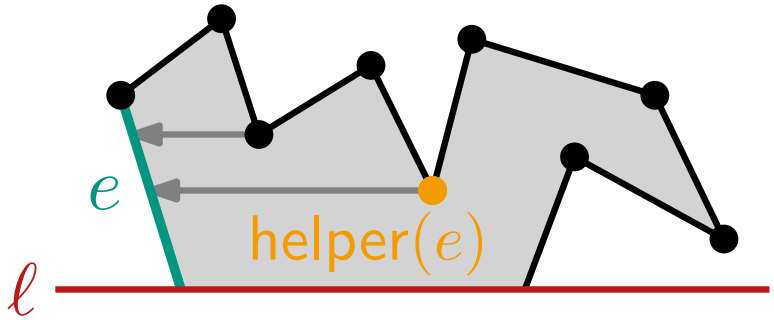
1) Diagonalen für Splitknoten

- betrachte für jeden Knoten v linke Nachbarkante $\text{left}(v)$ bzgl. horizontaler sweep line ℓ



- verbinde Splitknoten v zu niedrigstem Knoten w oberhalb v mit $\text{left}(w) = \text{left}(v)$

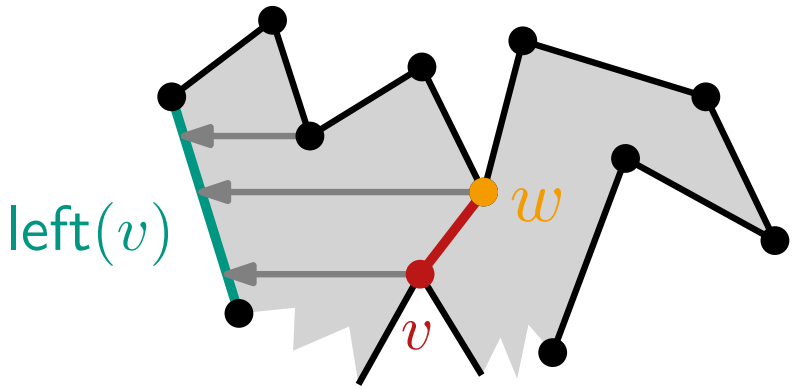
- speichere für jede Kante e den untersten Knoten w mit $\text{left}(w) = e$ als $\text{helper}(e)$



In Richtung Sweep-Line-Algorithmus

1) Diagonalen für Splitknoten

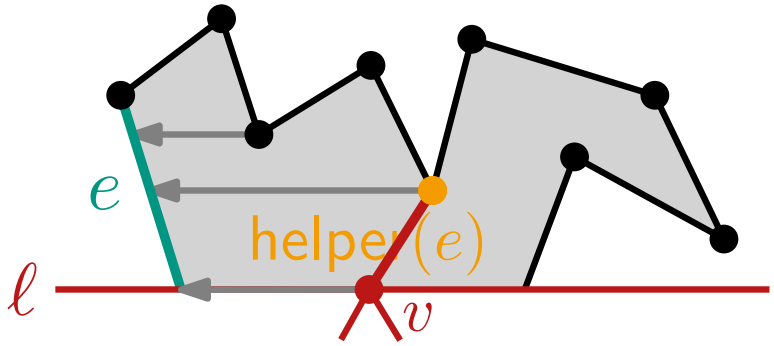
- betrachte für jeden Knoten v linke Nachbarkante left(v) bzgl. horizontaler sweep line ℓ



- verbinde Splitknoten v zu niedrigstem Knoten w oberhalb v mit left$(w) =$ left(v)

- speichere für jede Kante e den untersten Knoten w mit left$(w) = e$ als helper(e)

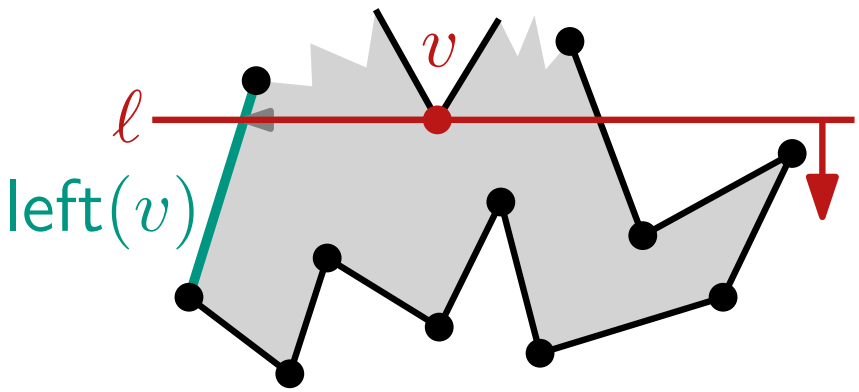
- trifft ℓ auf Splitknoten v: verbinde v mit helper(left(v))



In Richtung Sweep-Line-Algorithmus

2) Diagonalen für Mergeknoten

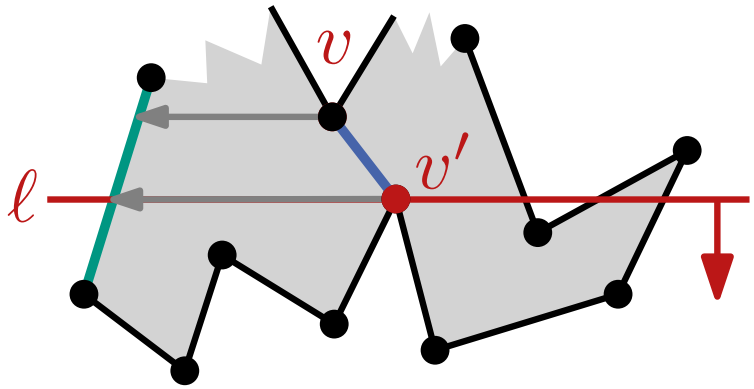
- erreicht man Mergeknoten v wird helper(left$(v)) = v$



In Richtung Sweep-Line-Algorithmus

2) Diagonalen für Mergeknoten

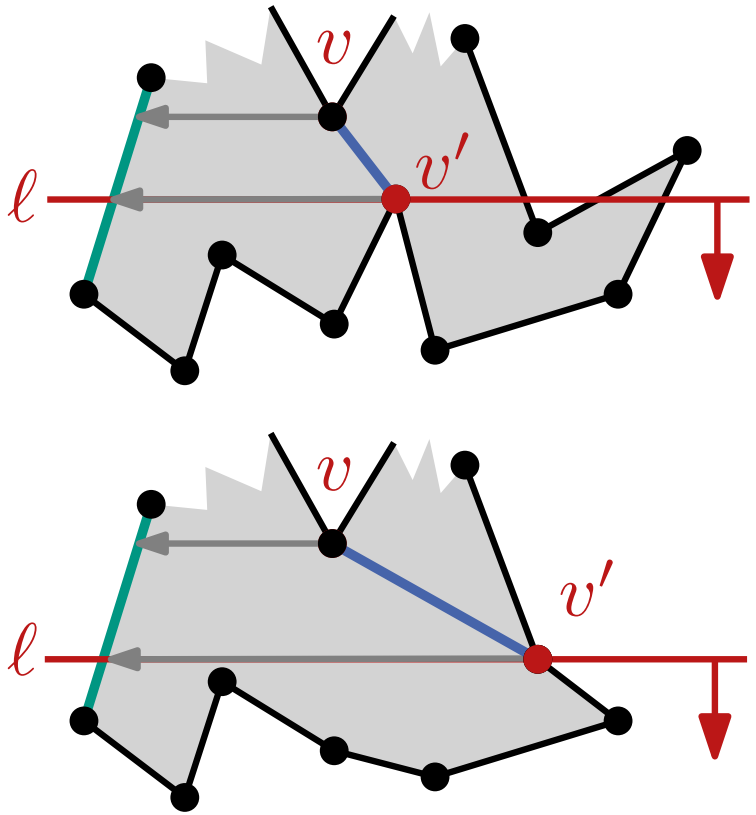
- erreicht man Mergeknoten v wird helper(left$(v)) = v$
- erreicht man Splitknoten v' mit left$(v') =$ left(v) wird Diagonale (v, v') eingefügt



In Richtung Sweep-Line-Algorithmus

2) Diagonalen für Mergeknoten

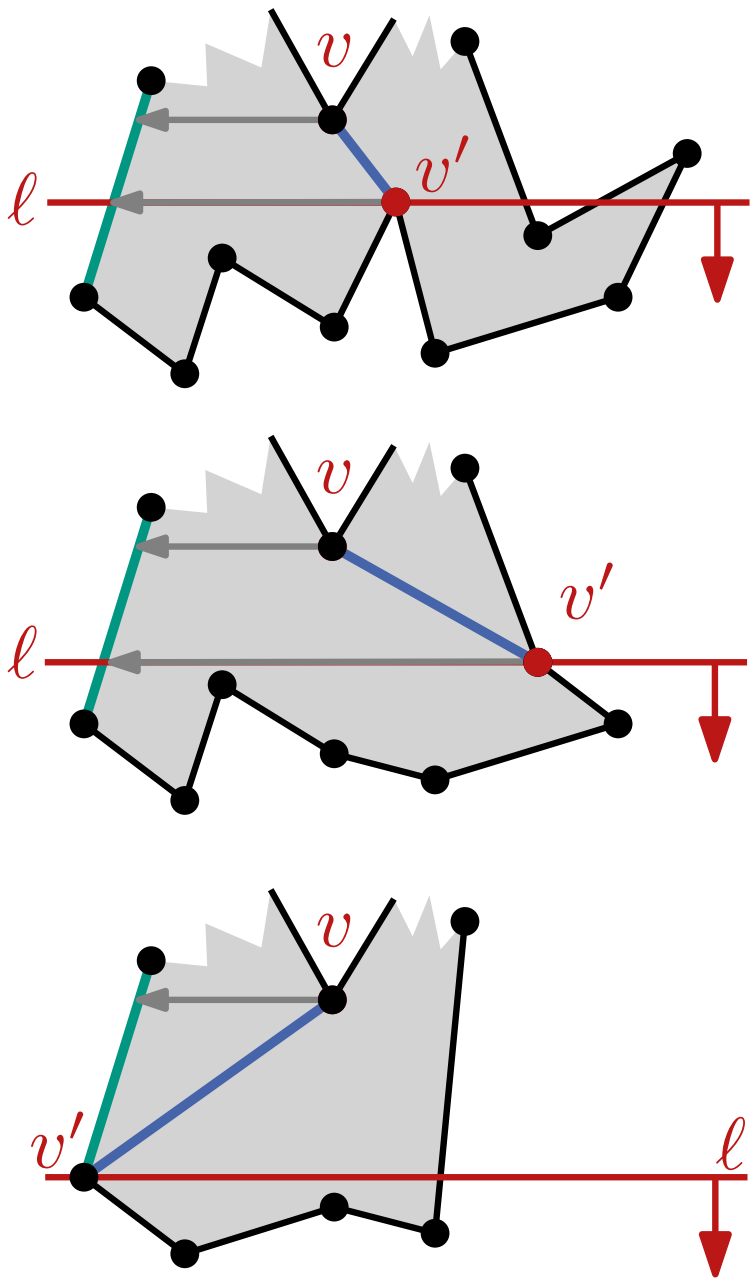
- erreicht man Mergeknoten v
  wird helper(left$(v)) = v$
- erreicht man Splitknoten v'
  mit left$(v') =$ left(v) wird
  Diagonale (v, v') eingefügt
- ersetzt man helper(left(v)) durch v'
  wird Diagonale (v, v') eingefügt



In Richtung Sweep-Line-Algorithmus

2) Diagonalen für Mergeknoten

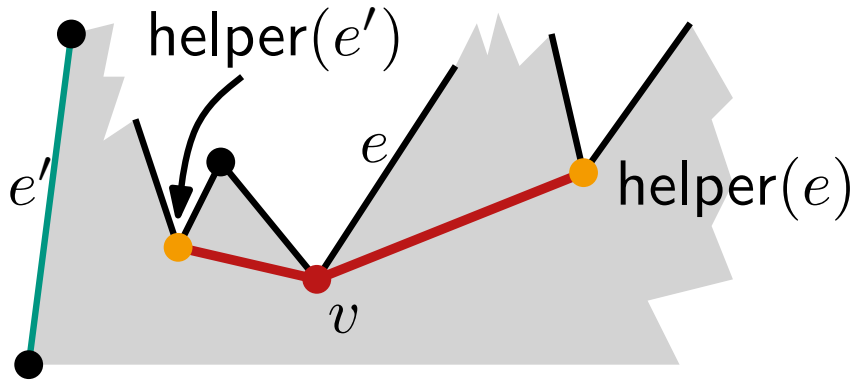
- erreicht man Mergeknoten v
  wird helper(left$(v)) = v$
- erreicht man Splitknoten v'
  mit left$(v') =$ left(v) wird
  Diagonale (v, v') eingefügt
- ersetzt man helper(left(v)) durch v'
  wird Diagonale (v, v') eingefügt
- erreicht man das Ende v' von left(v)
  wird Diagonale (v, v') eingefügt



Aufgabe 1

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') $\leftarrow v$

Aufgabe 2

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 $\mathcal{D}$

Annahme: P hat nur $O(1)$ Wendepunkte.

Aufgabe: Beschleunige Verfahren auf $O(n)$ Zeit.

Aufgabe 2

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 $\mathcal{D}$

Annahme: P hat nur $O(1)$ Wendepunkte.

Aufgabe: Beschleunige Verfahren auf $O(n)$ Zeit.

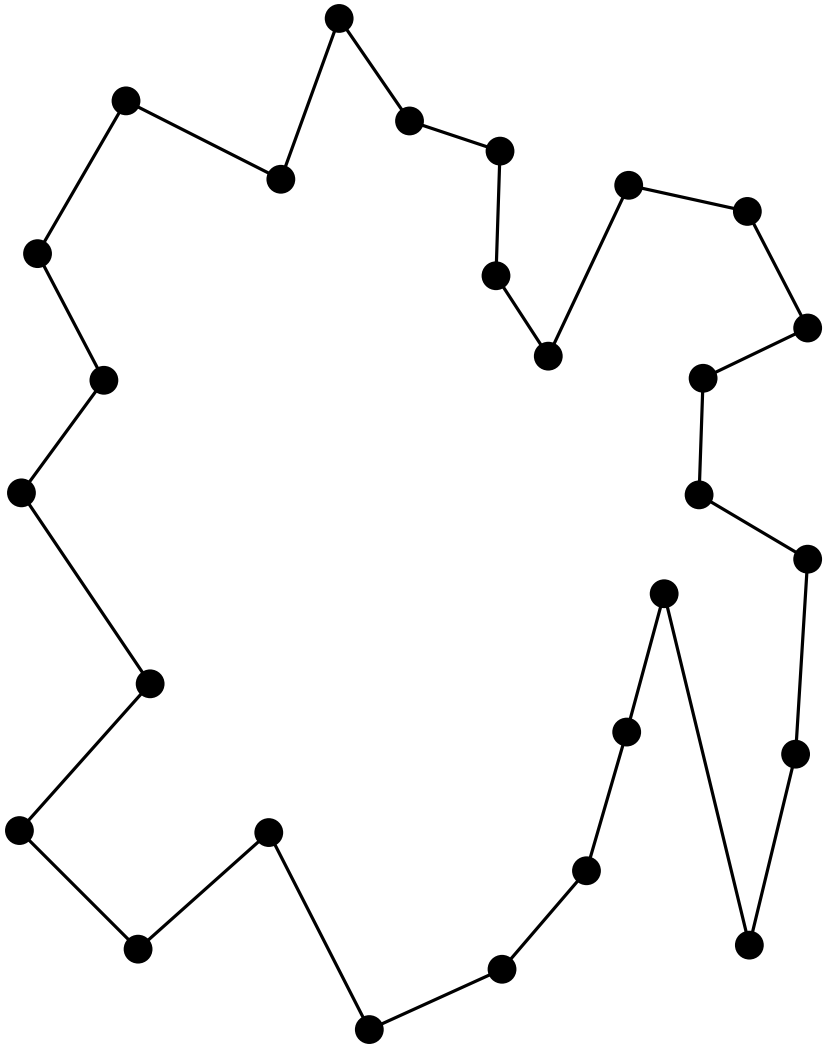
Beobachtung:

Erstellung von $\mathcal{Q}$ kostet $O(n \log n)$ Zeit.

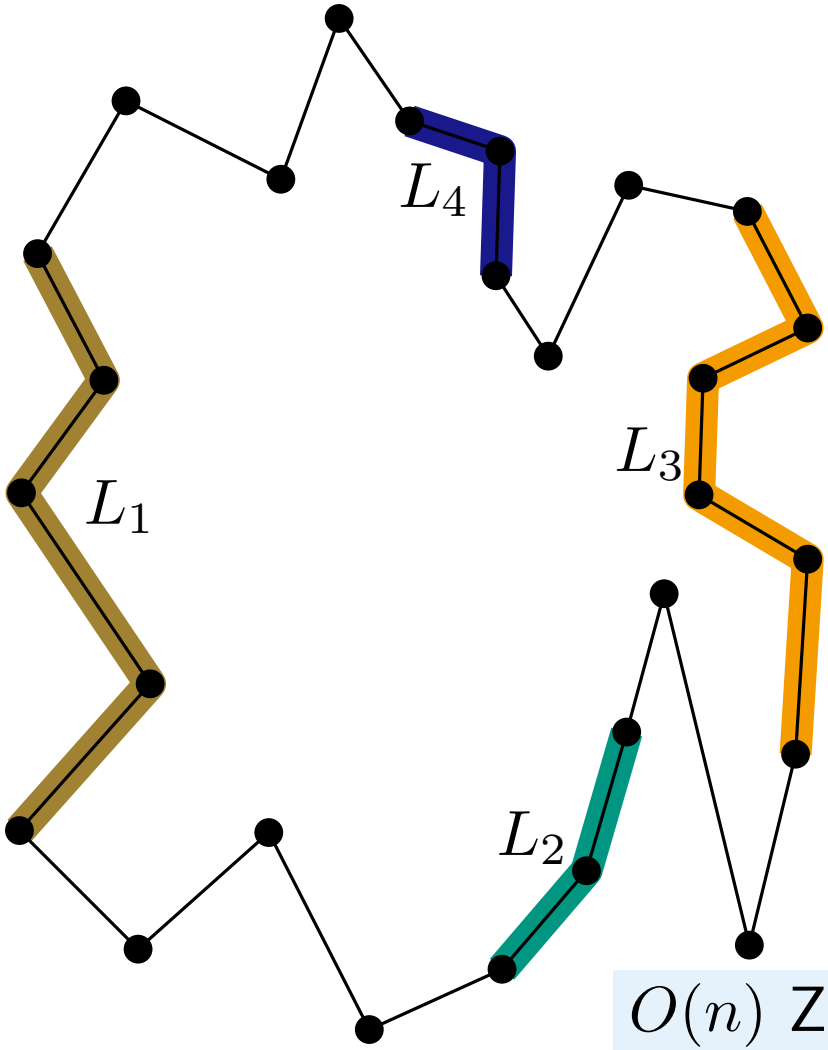
Anfragen in $\mathcal{T}$ kosten insgesamt $O(n \log n)$ Zeit.

Aufgabe 2

1. Schritt: Erstelle Queue $\mathcal{Q}$ in $O(n)$ Zeit.



Aufgabe 2



1. Schritt: Erstelle Queue $\mathcal{Q}$ in $O(n)$ Zeit.

Durchlauf P gegen den Uhrzeigersinn.

↳ Fasse aufeinanderfolgende reguläre Knoten in Liste zusammen.

Beobachtung:

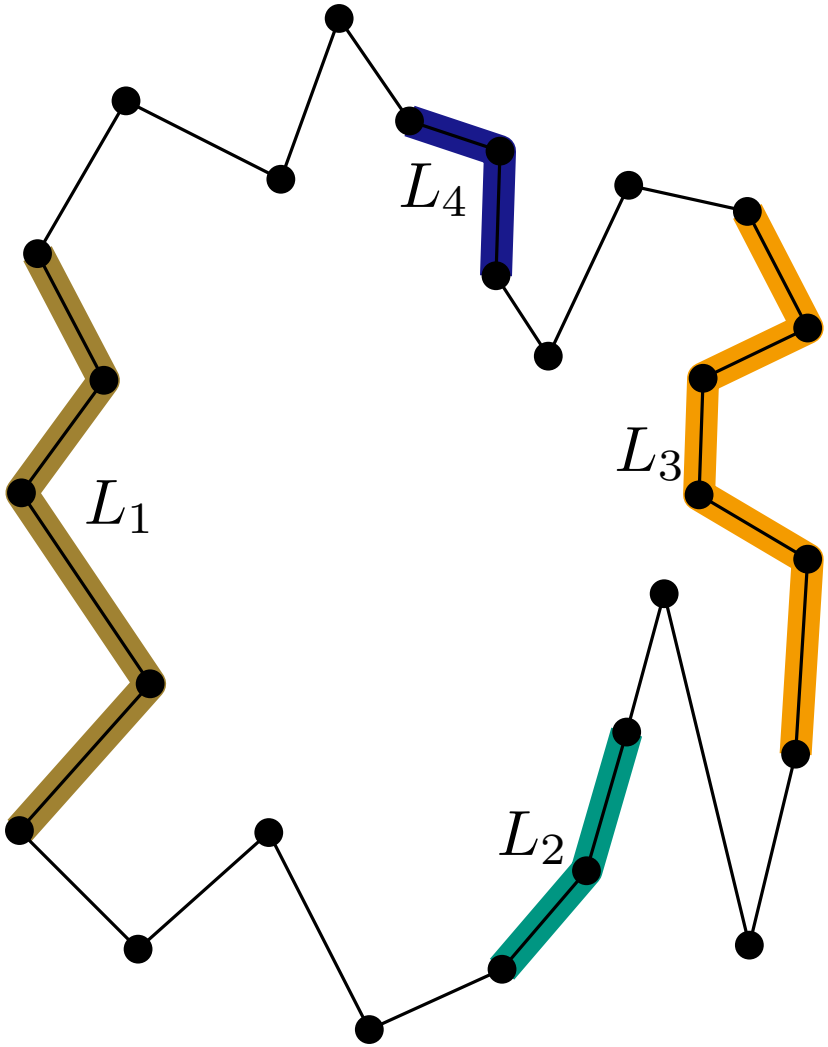
- Listen sind nach y-Koordinate sortiert.
- $O(1)$ viele Listen.

1. Verwende *Merge*-Schritt von Merge-Sort auf Listen an, um *eine* Liste zu erhalten.
2. Füge Wendeknoten sortiert in Liste ein.

$O(n)$ Zeit, da
$O(1)$ viele Listen und $O(1)$ viele Wendepunkte.

⟶ Queue $\mathcal{Q}$ kann in $O(n)$ Zeit aufgebaut werden.

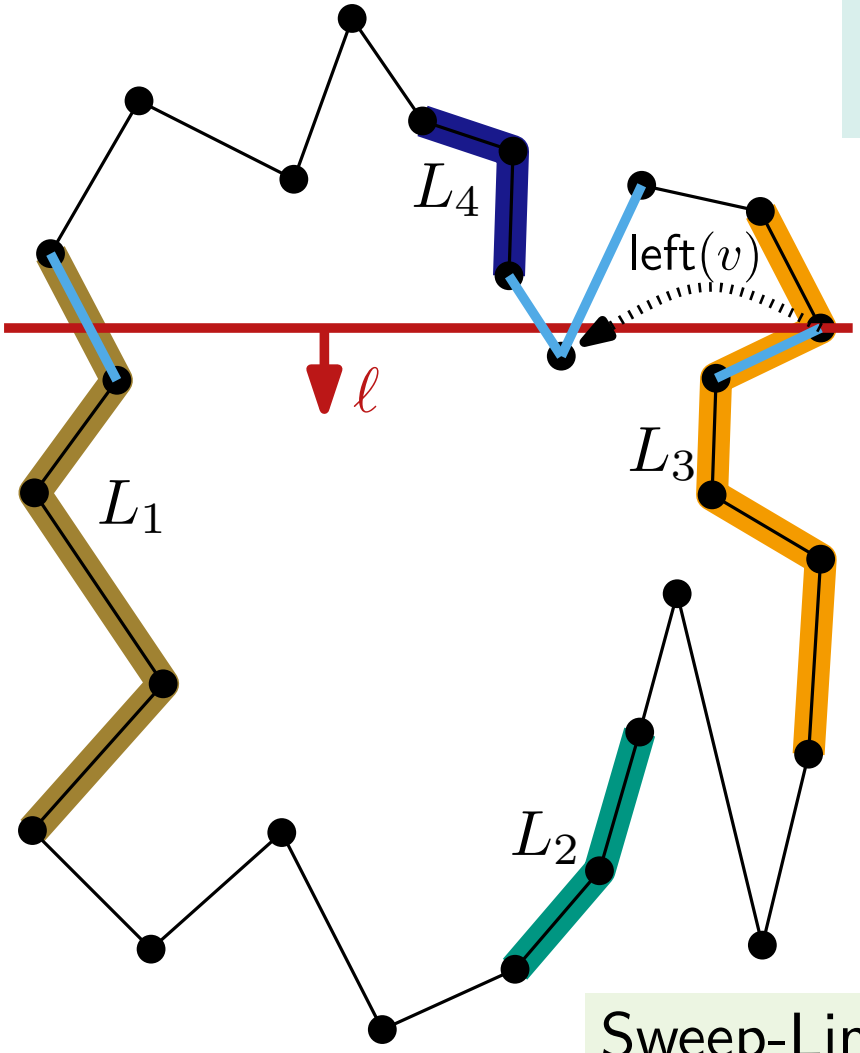
Aufgabe 2



2. Schritt: Ersetze $\mathcal{T}$.

Aufgabe von $\mathcal{T}$: Für Knoten v bestimme Kante left(v) direkt links von v.

Aufgabe 2



2. Schritt: Ersetze $\mathcal{T}$.

Aufgabe von $\mathcal{T}$: Für Knoten v bestimme Kante left(v) direkt links von v.

Idee: Berechne für jeden Knoten v die Kante left(v) vor.

Sweep-Line: von oben nach unten.

Sweep-Zustand:

Kanten die Sweep-Line schneiden

Event: Knoten des Polygons

Bestimme Kante, die Sweep-Line direkt links von aktuellen Knoten schneidet.

Sweep-Line schneidet immer nur $O(1)$ viele Kanten, da $O(1)$ viele Listen und $O(1)$ viele Wendepunkte.