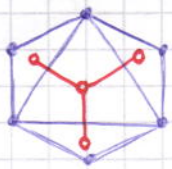
Algorithmische Geometrie 17.12.2010

Übungsblatt 2

1) Beweise oder widerlege: Der Dualgraph einer Triangulierung eines (y-) monotonen Polygons ist immer ein Pfad.

Lösung: Die Aussage ist falsch. Gegenbeispiel: gleichmäßiges 6-Eck ist konvex, also auch monoton.
Hat aber eine Triangulierung, deren Dualgraph kein Pfad ist:

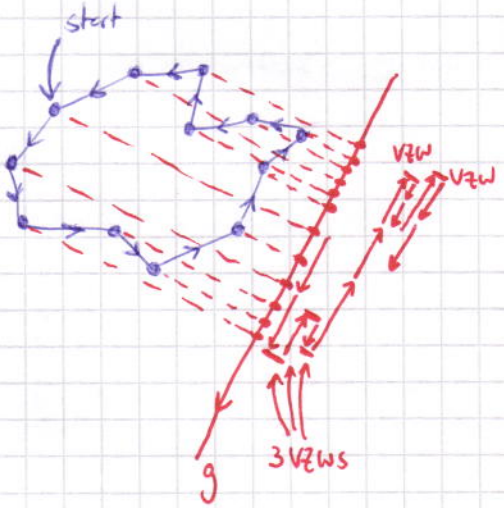


2)
a) Geg.: Ein einfaches Polygon P und eine Gerade g.

Gib einen Algorithmus an, der in O(n) Schritten überprüft, ob P entlang g monoton ist.

Ideen

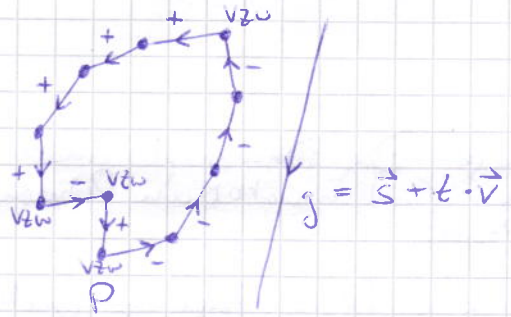
Projektion der Ecken auf g
⇒ Ordnung der Punkte
⇒ 2 Vorzeichenwechsel in der Richtung bzgl. g „erlaubt", Sonst nicht monoton.



2)

a) Musterlösung

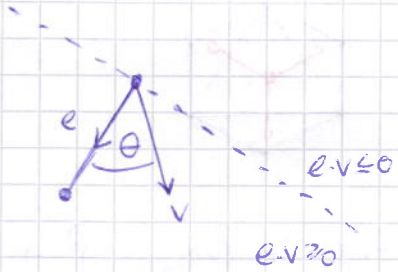
Idee:



Benutze das Skalarprodukt von Polygon-Kante und Richtungsvektor der Geraden um die Richtung der Kante bzgl. g zu erkennen. (im Grunde nichts anderes als Projektion)

Skalarprodukt:

$e \cdot v = |e| \cdot |v| \cdot \cos\theta$



2)

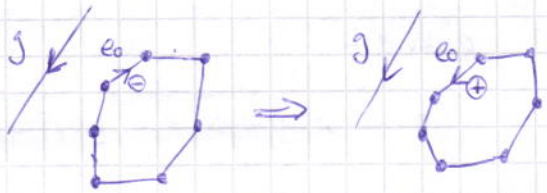
a) Algorithmus

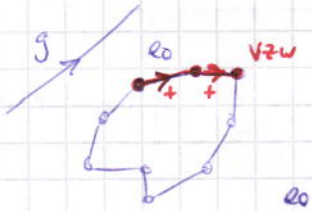
definiere: $m(e) := e.destination - e.origin$

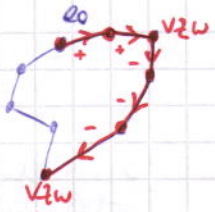
(Differenz der Positionen als Vektoren)

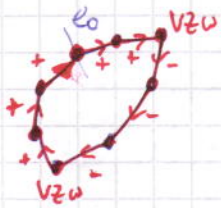
Starte bei irgendeiner HK e_0

```
if( m(e0)·v < 0 ) {
    e0 := e0.twin
}
e := e0
while ( m(e)·v ≥ 0 ) {
    e := e.next
}

while( m(e)·v ≤ 0 ) {
    if (e.destination = e0.origin ) {
        return MONOTON
    }
    e := e.next
}

while( m(e)·v ≥ 0 ) {
    if (e.destination = e0.origin) {
        return MONOTON
    }
    e := e.next
}

return NICHT_MONOTON
```

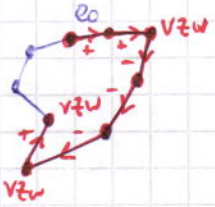








monoton



nicht monoton

2)

b) Geg.: Ein einfaches Polygon P.

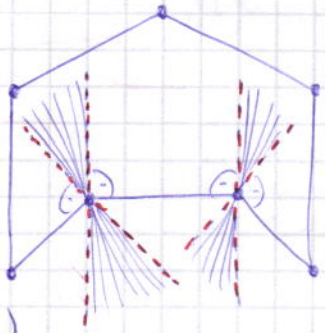
Gib einen Algorithmus an, der in $O(n)$ Schritten entscheidet, ob es eine Gerade g gibt, so dass P entlang g monoton ist.

Ideen

- Betrachte „erlaubte" Winkel an konkaven Ecken
- Schnittmenge der erlaubten Bereiche enthält alle Geraden-Winkel (Orientierungen) für die P monoton ist.
  (Hier stattdessen $[0, 2\pi)$ – Vereinigung der „verbotenen" Bereiche)

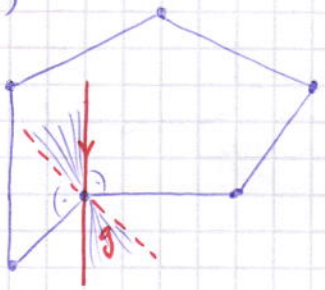
Musterlösung

Achtung! Spezialfall:



Sind die Ränder (gestrichelt) erlaubt oder verboten?

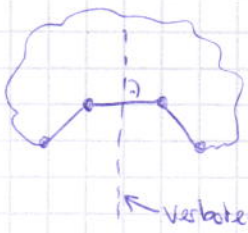
Bei P' muss g eine erlaubte Gerade sein (P' entlang g monoton), also Rand erlaubt.



P ist aber entlang g nicht monoton, obwohl g in beiden Rändern enthalten ist!

Beobachtung: P hat entlang g keinen Sattelpunkt gibt!

⇒ Verbiete Orientierungen, die senkrecht zu Kanten verlaufen, welche zu 2 konkaven Ecken gehören! („problematische" Kanten)



Das soll auch für entartete Ecken gelten:

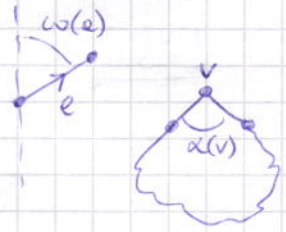
2)

b) Algorithmus

Definiere:

$\omega(e)$:= Orientierung der Halbkante e

$\alpha(v)$:= Innenwinkel am Knoten v



$S := [0, 2\pi)$ „erlaubte" Orientierungen von g

$E := \emptyset$ „problematische" Kanten

e_0 := irgendeine Halbkante von P

$e := e_0$

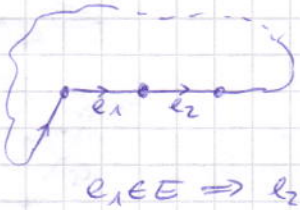
do {

  $v :=$ e.destination

  if ($e \in E \wedge \alpha(v) = \pi$) {

    $E := E \cup \{e.next\}$

  }



$e_1 \in E \Rightarrow e_2 \in E$

  if ($\alpha(v) > \pi$) { // v konkav

    $S := S \setminus \left(\omega(e.next) + \frac{\pi}{2},\ \omega(e) + \frac{\pi}{2} \right)$ (a)

    $S := S \setminus \left(\omega(e.next) + \frac{3\pi}{2},\ \omega(e) + \frac{3\pi}{2} \right)$ (b)

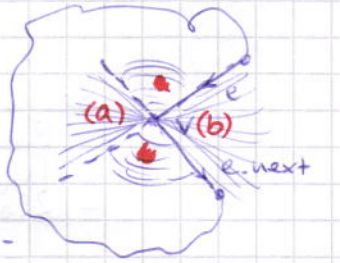
    if($e \in E$) { // verbiete Orientierungen senkrecht zu problematischen Kanten

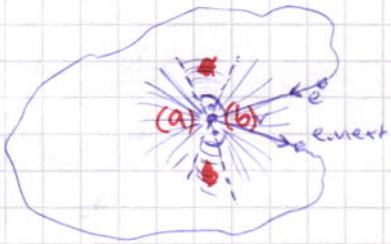
      $S := S \setminus \{\omega(e) + \frac{\pi}{2}\}$

    }

    $E := E \cup \{e, e.next\}$

  }





  $e := e.next$

}

while ($e \neq e_0$)

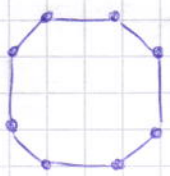
$S = \emptyset \Rightarrow$ Es ex. keine Gerade zu der P monoton ist

Sonst: P ist monoton für alle Geraden mit Orientierung aus S.

3) Zeige

a) Für jedes $n \in \mathbb{N}$ gibt es ein Polygon mit mindestens n Ecken, das entlang jeder Geraden monoton ist.

Lösung:

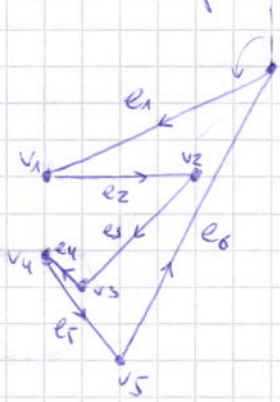


Konstruiere das regelmäßige n-Eck (P).

$\Rightarrow$ P ist konvex
$\Rightarrow$ keine konkaven Ecken
$\Rightarrow$ keine Sattelpunkte
$\Rightarrow$ monoton

b) Es gibt ein Polygon mit < 10 Ecken, das entlang keiner Geraden monoton ist.

Lösung: Viele Beispiele findbar. u.A.:



$\omega(e_1) = 116{,}57°$

$\omega(e_2) = 270°$

$\omega(e_3) = 135°$

$\omega(e_4) = 45°$

$\omega(e_5) = 206{,}57°$

$\omega(e_6) = 333{,}43°$

Benutze Idee aus 2b
- Bestimme „verbotene" Bereiche für jeden Konkaven Knoten
- Ziehe diese von $[0°, 360°)$ ab

Erlaubte Orientierungen

Verbotene Bereiche

v_2: $(225°, 0°) \cup (45°, 180°)$

v_3: $(135°, 225°) \cup (315°, 45°)$

e_3 problematisch $\Rightarrow \{45°, 225°\}$

$$[0°, 360°) \setminus \left((225°, 0°) \cup (45°, 180°) \cup (135°, 225°) \cup (315°, 45°) \cup \{45°, 225°\} \right) = \emptyset$$

4)

b) Idee: Laufe auf dem Rechteck entlang, besuche so alle Voronoi-Zellen, die ohne das Rechteck Halbgeraden enthalten würden.

Algorithmus

Starte bei einem Knoten v_0 auf dem Rechteck.
(wenn v_0 nur an äußerer Voronoi-Zelle aber nicht unbedingt auf dem Rechteck ist, iteriere zum Rand)

Iteriere über die ausgehenden Hks an v_0 bis e.face = Außenfläche.

L = () Liste von Ecken der konvexen Hülle.

do {

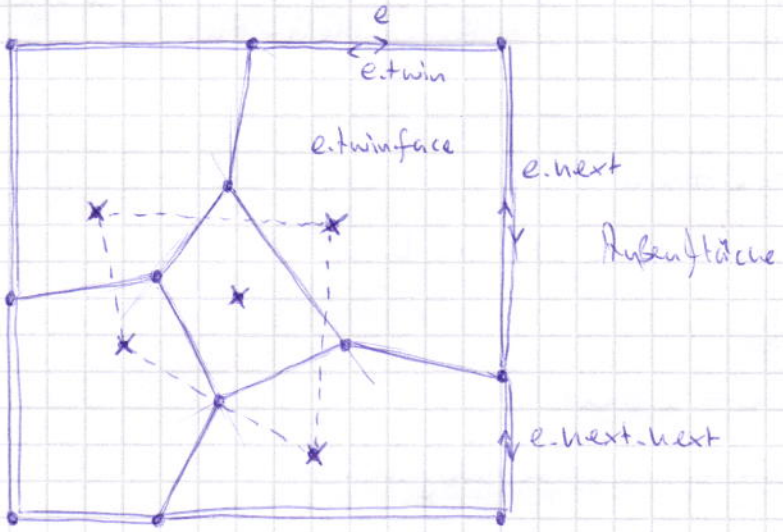
if (e.twin.face.site $\notin$ L) {

Hänge e.twin.face.site an L an

}

e := e.next

} while (e.origin $\neq v_0$)



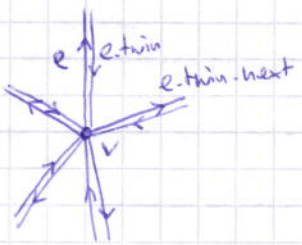
Es werden k+4 Kanten besucht, jede in O(1) abgearbeitet

4)

a) Grundidee (auch für b):

Die konvexe Hülle besteht aus den Sites, die Halbgeraden als Kanten haben!

Einschub: Über alle ausgehenden Hks eines Knotens iterieren



Geg: Knoten v

```
e := v.edge
e0 := e
do {
    // tue etwas mit e
    e := e.twin.next
} while (e ≠ e0)
```

Algorithmus L = () Liste von Knoten der Konv. Hülle

Starte am Knoten v_0, der an einer Halbgeraden liegt.

v := v_0

Iteriere über die Hks an v_1 bis auf die Halbgerade getroffen wurde. e = Halbgeraden-Hk.

```
do {
    e := e.twin
    Hänge e.face.site an L an
    do {
        e := e.next
    } while (e ist keine Halbgerade)
} while (e.origin ≠ v0)
```

Jede Hk wird höchstens einmal besucht und in O(1) abgearbeitet.

Da es nur O(n) Hks gibt (VL/Euler-Formel!), arbeitet der Algorithmus in O(n).