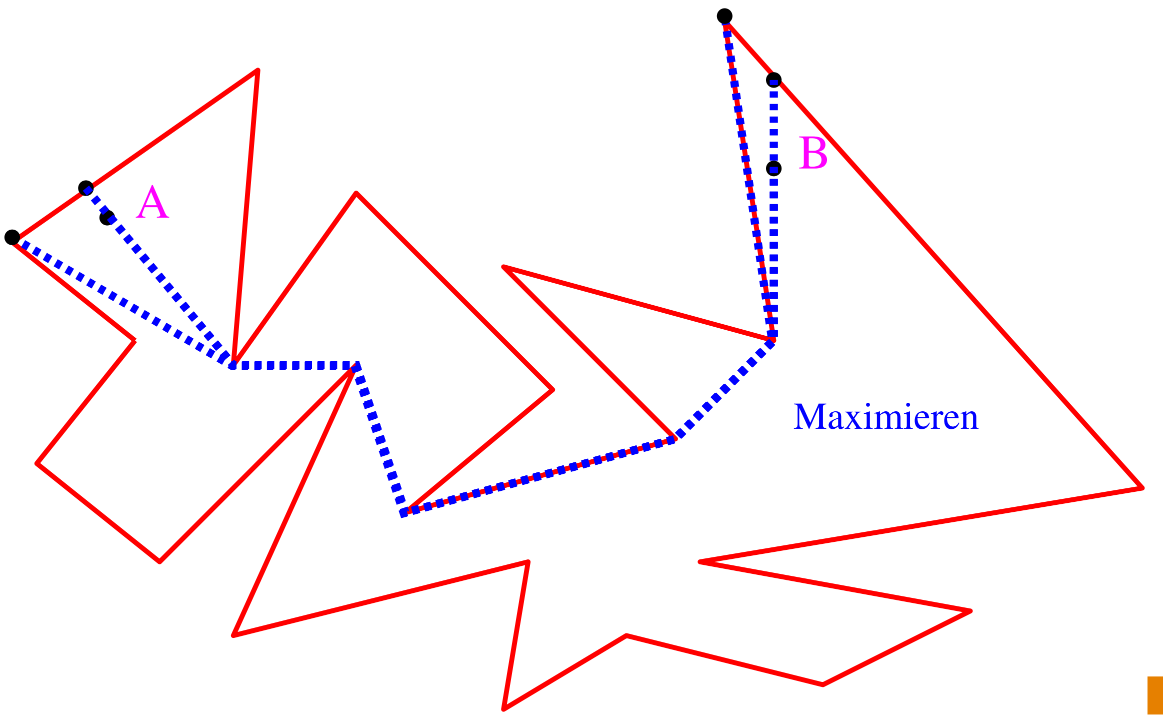
Offline Bewegungsplanung: Maximaler Durchmesser

Elmar Langetepe
University of Bonn

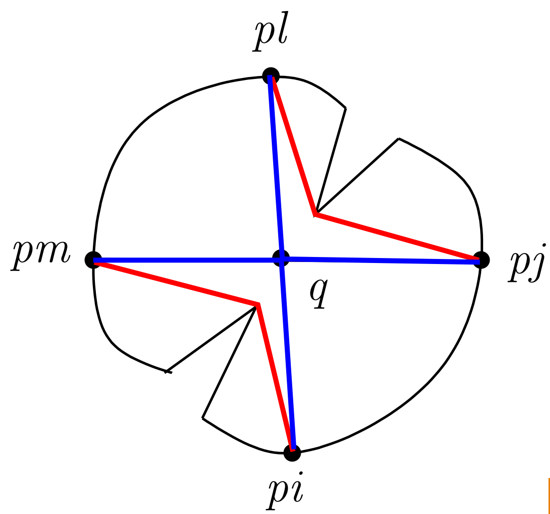
1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon P
- Längster Kürzester Weg zwischen zwei Punkten
- Endpunkte sind Ecken des Polygones n^2 Kandidatenpaare
- Formal: $\max_{p_i, p_j \text{ Ecken von } P} d(p_i, p_j)$



Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes
- Monge Eigenschaft: $d(p_i, p_m) + d(p_j, p_l) \leq d(p_i, p_l) + d(p_j, p_m)$
- Wege $\pi(p_i, p_l)$ und $\pi(p_j, p_m)$ schneiden sich
- Dreiecksungleichungen anwenden!!



Matrix A mit schöner Eigenschaft!

$$\begin{array}{c@{\quad}c}
 & \begin{array}{cccccc} 1 & 2 & 3 & & n-1 & n \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ \vdots \\ n-1 \\ n \end{array} &
\begin{pmatrix}
1-n & \mathtt{d(1,2)} & \mathtt{d(1,3)} & \ldots & \mathtt{d(1,n-1)} & \mathtt{d(1,n)} \\
1-n & 2-n & \mathtt{d(2,3)} & \ldots & \mathtt{d(2,n-1)} & \mathtt{d(2,n)} \\
1-n & 2-n & 3-n & \ldots & \mathtt{d(3,n-1)} & \mathtt{d(3,n)} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1-n & 2-n & 3-n & \ldots & \ldots & \mathtt{d(n-1,n)} \\
1-n & 2-n & 3-n & \ldots & \ldots & 0
\end{pmatrix}
\end{array}$$

$\mathtt{d(i,j)}$ entspricht $\mathtt{d(p_i,p_j)}$ und $p_1, p_2, \ldots, p_n$ Knoten entlang des Randes von P!

Def. 1.17: Monotone Matrix!

$n \times m$-Matrix $A = (a_{i\ell})$ heißt *monoton*, falls

$$\forall 1 \le i < j \le n, 1 \le k < \ell \le m : (a_{ik} < a_{i\ell} \Rightarrow a_{jk} < a_{j\ell}).$$

$$\begin{array}{cc} & \begin{array}{ccc} k & & \ell \end{array} \\ \begin{array}{c} i \\ \\ j \end{array} & \begin{pmatrix} a_{ik} & < & a_{i\ell} \\ & \Downarrow & \\ a_{jk} & < & a_{j\ell} \end{pmatrix} \end{array}$$

Linkeste Zeilenmaxima weiter nach rechts

$$\begin{pmatrix} \color{red}{5} & 5 & 5 & 5 & 5 \\ 1 & 7 & \color{red}{9} & 6 & 3 \\ 4 & 6 & 10 & 7 & \color{red}{12} \end{pmatrix}$$

Lemma 1.19: A ist monoton!

$$
\begin{array}{c}
\\
\begin{array}{c} 1 \\ 2 \\ 3 \\ \vdots \\ n-1 \\ n \end{array}
\end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & & n-1 & n \end{array} \\
\begin{pmatrix}
1-n & \mathtt{d}(1,2) & \mathtt{d}(1,3) & \ldots & \mathtt{d}(1,\mathtt{n}-1) & \mathtt{d}(1,\mathtt{n}) \\
1-n & 2-n & \mathtt{d}(2,3) & \ldots & \mathtt{d}(2,\mathtt{n}-1) & \mathtt{d}(2,\mathtt{n}) \\
1-n & 2-n & 3-n & \ldots & \mathtt{d}(3,\mathtt{n}-1) & \mathtt{d}(3,\mathtt{n}) \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1-n & 2-n & 3-n & \ldots & \ldots & \mathtt{d}(\mathtt{n}-1,\mathtt{n}) \\
1-n & 2-n & 3-n & \ldots & \ldots & 0
\end{pmatrix}
\end{array}
$$

Beweis!!!

Theorem 1.20: Maxima für monotone Matrizen

- Monotone $n \times m$-Matrix $A = (a_{i\ell})$, $m \geq n$
- Alg.1.7 Spaltenreduktion (Spalten streichen): monotone $n \times n$-Matrix A'
- Zeilenmaxima (A, i) aus Zeilenmaxima (A', i) gewinnen
- Alg.1.8 Zeilenreduktion (gerade Zeilen auswählen): Monotone $n \times m$-Matrix B
- Zeilenmaxima ungerade Zeilen aus Zeilenmaxima gerade Zeilen gewinnen
- Rekursiv mit Spaltenreduktion (Alg. 1.7)!!

Alg. 1.7: Spaltenreduktion

Invariante:

$$\begin{array}{ccccccccc}
a_{1,1} & \geq & a_{1,2} & \geq & a_{1,3} & \geq & a_{1,4} & & a_{1,5} \\
 & & a_{2,2} & \geq & a_{2,3} & \geq & a_{2,4} & & a_{2,5} \\
 & & & & a_{3,3} & \geq & a_{3,4} & & a_{3,5} \\
 & & & & & & a_{4,4} & \overset{?}{<} & a_{4,5}
\end{array}$$

Falls nein, dann:

$$\begin{array}{ccccccccc}
a_{1,1} & \geq & a_{1,2} & \geq & a_{1,3} & \geq & a_{1,4} & \geq & a_{1,5} \\
 & & & & & & & \Uparrow & \\
 & & a_{2,2} & \geq & a_{2,3} & \geq & a_{2,4} & \geq & a_{2,5} \\
 & & & & & & & \Uparrow & \\
 & & & & a_{3,3} & \geq & a_{3,4} & \geq & a_{3,5} \\
 & & & & & & & \Uparrow & \\
 & & & & & & a_{4,4} & \geq & a_{4,5}
\end{array}$$

Alg. 1.7: Spaltenreduktion

Invariante:

$$\begin{array}{ccccccccc}
a_{1,1} & \geq & a_{1,2} & \geq & a_{1,3} & \geq & a_{1,4} & & a_{1,5} \\
 & & a_{2,2} & \geq & a_{2,3} & \geq & a_{2,4} & & a_{2,5} \\
 & & & & a_{3,3} & \geq & a_{3,4} & & a_{3,5} \\
 & & & & & & a_{4,4} & \stackrel{?}{<} & a_{4,5}
\end{array}$$

Falls ja, dann:

$$\begin{array}{ccccccc}
a_{1,1} & \geq & a_{1,2} & \geq & a_{1,3} & & a_{1,5} \\
 & & a_{2,2} & \geq & a_{2,3} & & a_{2,5} \\
 & & & & a_{3,3} & \stackrel{?}{<} & a_{3,5} \\
 & & & & & & a_{4,5}
\end{array}$$

Streiche Spalte 4, weil:

$a_{i,4} \leq a_{i,3}$ für $i = 1, 2, 3$ und $a_{i,4} < a_{i,5}$ für $i = 4, \ldots, n$

Alg. 1.7: Spaltenreduktion

Letzte Zeile erreicht!

$$\begin{array}{ccccccccccc}
a_{1,1} & \geq & a_{1,2} & \geq & a_{1,3} & \dots & a_{1,n} & \geq & a_{1,n+1} & \dots & a_{1,m} \\
 & & a_{2,2} & \geq & a_{2,3} & \dots & a_{2,n} & \geq & a_{2,n+1} & \dots & a_{2,m} \\
 & & & \ddots & \dots & & \vdots & & & & \\
 & & & & \dots & \dots & a_{n,n} & \geq & a_{n,n+1} & \dots & a_{n,m}
\end{array}$$

Komplette Spalte $n+1$ Streichen!!

Weiter mit Vergleich: $a_{n,n} \stackrel{?}{<} a_{n,n+2}$

Beispiel!!!

Alg. 1.7: Spaltenreduktion

- Input: monotone $n \times m$ Matrix A, $m \geq n$
- Output: monotone $n \times n$ Matrix A'
- Zeilenmaxima von A' und A identisch
- Analyse:
  - $O(m)$ Vergleiche
  - $O(m)$ Zeiger für Rekonstruktion

Alg. 1.8: Zeilenmaxima

- Input: monotone $n \times m$ Matrix B
- Output: alle linkesten Zeilenmaxima $\max(i)$, $1 \leq i \leq n$
- Algorithmus:
  - C Zeilen von B mit geradem Index $O(n)$
  - C' durch Spaltenreduktion(C) $O(m)$
  - Rekursiv: Zeilenmaxima(C'), $\frac{n}{2} \times \frac{n}{2}$ Matrix $T\left(\frac{n}{2}\right)$
  - Rekonstruktion (Zeiger) Zeilenmaxima von C $O(m)$
  - Berechnung Zeilenmaxima von B $O(m)$

Berechnung Zeilenmaxima von B

$$\begin{pmatrix}
x & x & x & x & x & x & x & x & x & x & x \\
x & x & x & x & x & x & x & x & x & x & x \\
x & x & x & x & x & x & x & x & x & x & x \\
x & x & x & x & x & x & x & x & x & x & x \\
x & x & x & x & x & x & x & x & x & x & x
\end{pmatrix}$$

$O(m)$ Schritte !!

Gesamtbeispiel

$$\begin{pmatrix} \Rightarrow 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 7 & 3 & \Rightarrow 9 & 4 & 3 \\ 4 & 6 & 5 & \Rightarrow 8 & 6 & 7 \\ 1 & 4 & 2 & 6 & 7 & \Rightarrow 8 \end{pmatrix}$$

Zeilenreduktion: $\begin{pmatrix} 1 & 7 & 3 & 9 & 4 & 3 \\ 1 & 4 & 2 & 6 & 7 & 8 \end{pmatrix}$

Spaltenreduktion: $\begin{pmatrix} \Rightarrow 9 & 3 \\ 6 & \Rightarrow 8 \end{pmatrix}$

Zeilenreduktion: $\begin{pmatrix} 6 & 8 \end{pmatrix}$

Spaltenreduktion: $\begin{pmatrix} \Rightarrow 8 \end{pmatrix}$

Theorem 1.20: Maxima für monotone Matrizen

Laufzeitanalyse:

$$
\begin{aligned}
T(m) &\leq T\left(\frac{n}{2}\right) \text{(Rek.)} + C \times m \text{(Sp.Red.+ Rekonstr. + Ber.)}\\
&\leq T\left(\frac{n}{4}\right) + C\left(m + \frac{n}{2}\right)\\
&\vdots \quad \vdots\\
&\leq T(1) + C\left(m + n\sum_{i=1}^{\log n}\frac{1}{2^i}\right) \in O(m)
\end{aligned}
$$

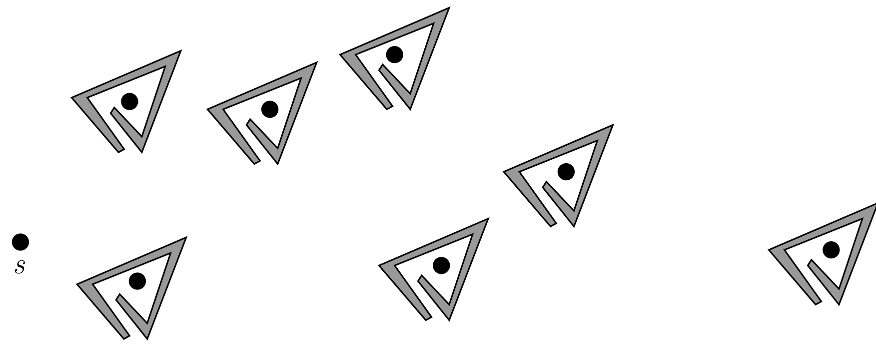
Nur $O(m)$ viele Vergleiche!!!

Theorem 1.21: Durchmesser eines Polygones

- $n \times n$ Matrix A (symbolisch)
- Preprocessing Guibas/Hershberger $O(n)$
- Zeilenmaxima von A mit $O(n)$ Vergleichen
- Je Vergleich $O(\log n)$ Aufwand für Wert
- Gesamtlaufzeit: $O(n \log n)$

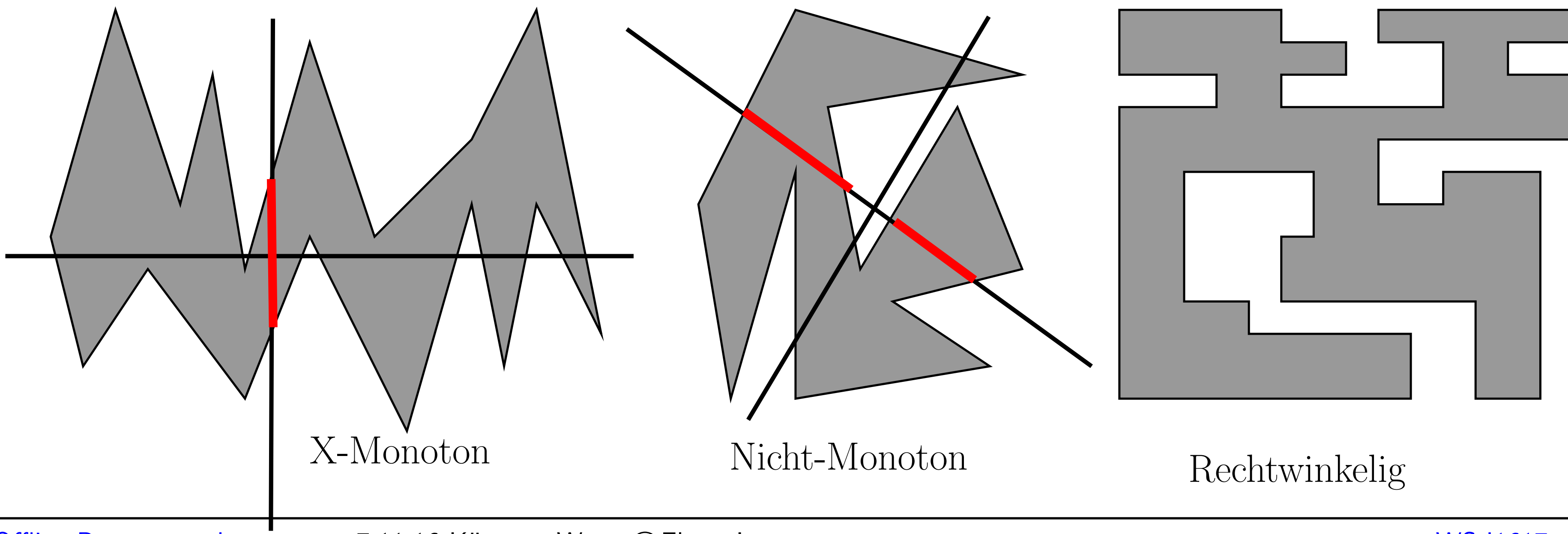
1.2.4 Shortest Watchman Routes

- Agent mit Sichtsystem, Startpunkt s
- Route: Einmal alles *sehen* und zurück
- Tour: Einmal alles *sehen* (offen)
- Polygonale Szene: NP-hart Theorem 1.22
- Beweis: TSP reduzieren auf SWR



Polynomiell innerhalb einfacher Polygonen

- Standardmittel (noch einfacher): Def. 1.23
- Monotone Polygone: Monoton bezüglich l
- Rechtwinkelige Polygone
- Rechtwinkelige und monotone Polygone: SWR in $O(n)$ Theorem 1.24



Was ist wichtig? Def. 1.25

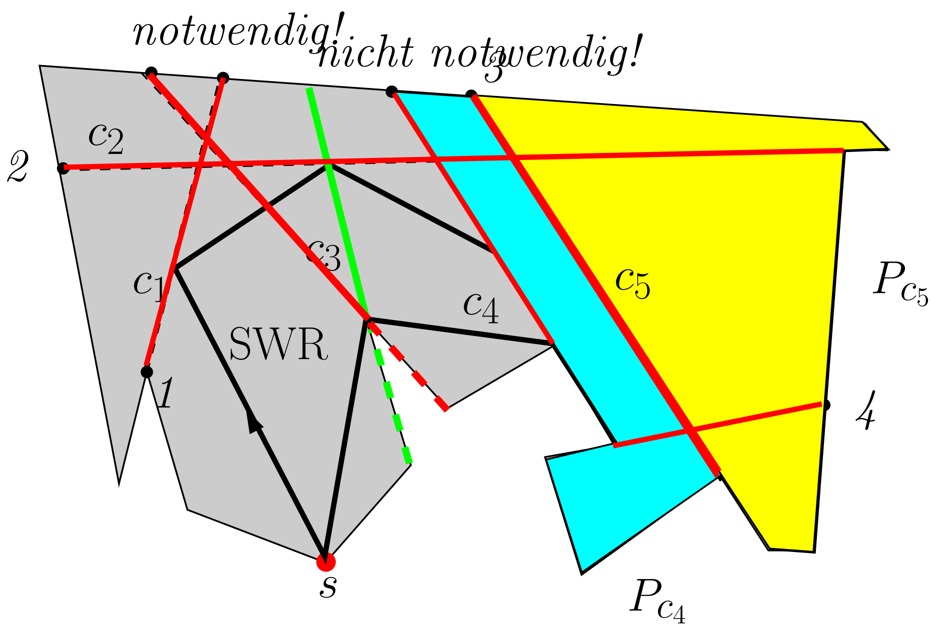
a) (Cuts) Extensionen der reflexen Ecken

b) Notwendige Cuts (bezüglich s)

c) Dominanz-Beziehung $P_{c_i} \subseteq P_{c_j}$

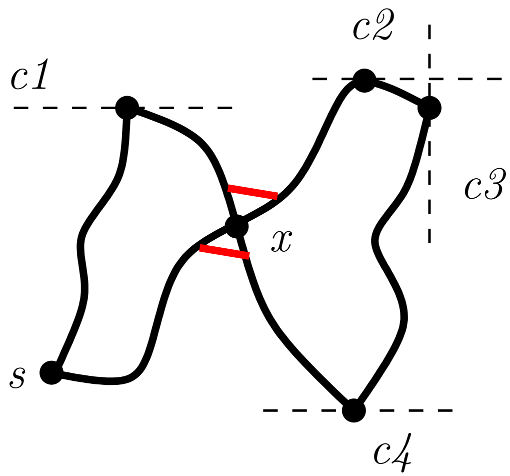
d) Wesentliche Cuts

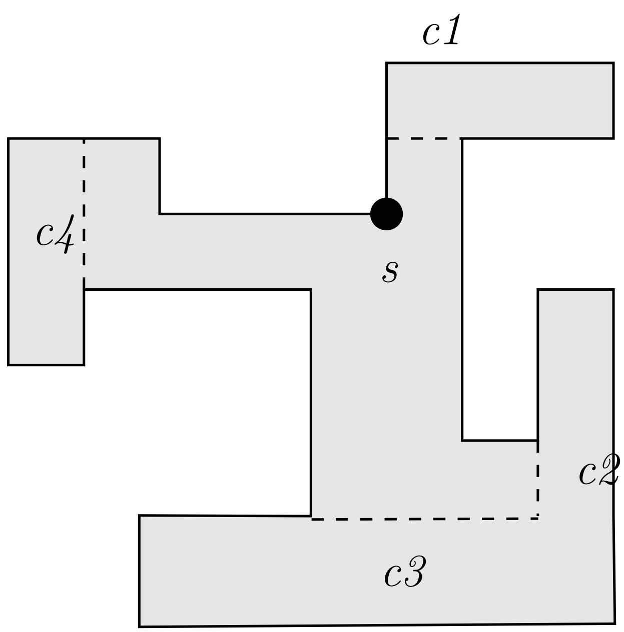
e) Ordnung der wesentlichen Cuts



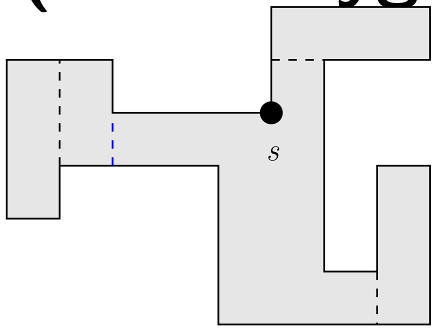
Lemma 1.26 Ordnung entlang des Randes

- Rechtwinkeliges Polygon
- Wesentliche Cuts werden nie überschritten!
- SWR besucht Cuts gemäß Ordnung
- Widerspruch! Abkürzung!
- $O(n)$ Algorithmus!!



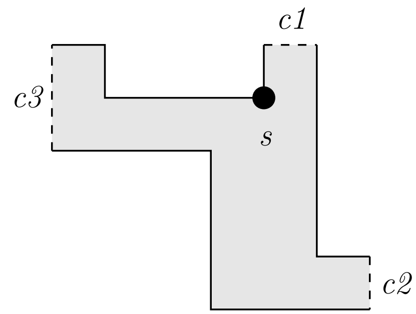


SWR (RW Polygon) Alg. 1.9/Th. 1.27: $O(n)$



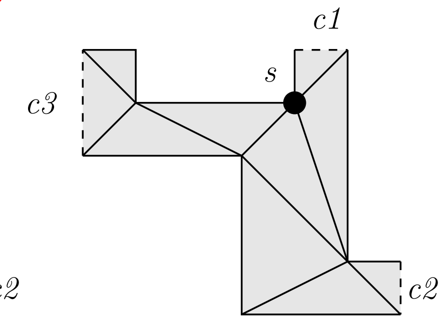
(i) Wesentliche Cuts

O(n)



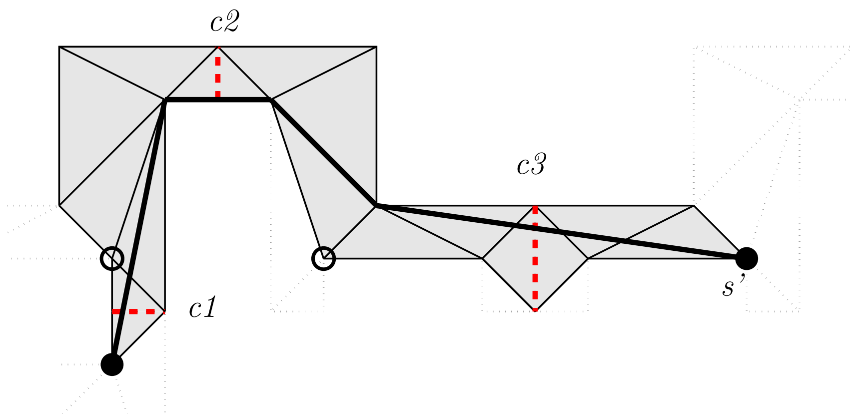
(ii) Abschneiden!

O(n)



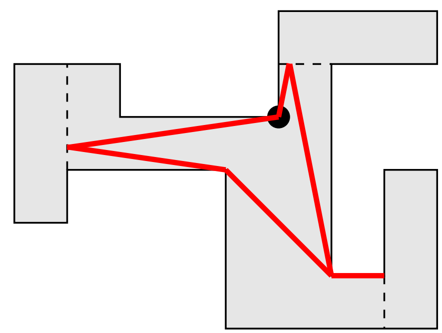
(iii) Triangulation

O(n)



(iv) Spiegeln und Ausrollen!!
(v) Weg berechnen

O(n)



(vi) Zusammenklappen!

O(n)