# Offline Bewegungsplanung: Preprocessing und Durchmesser

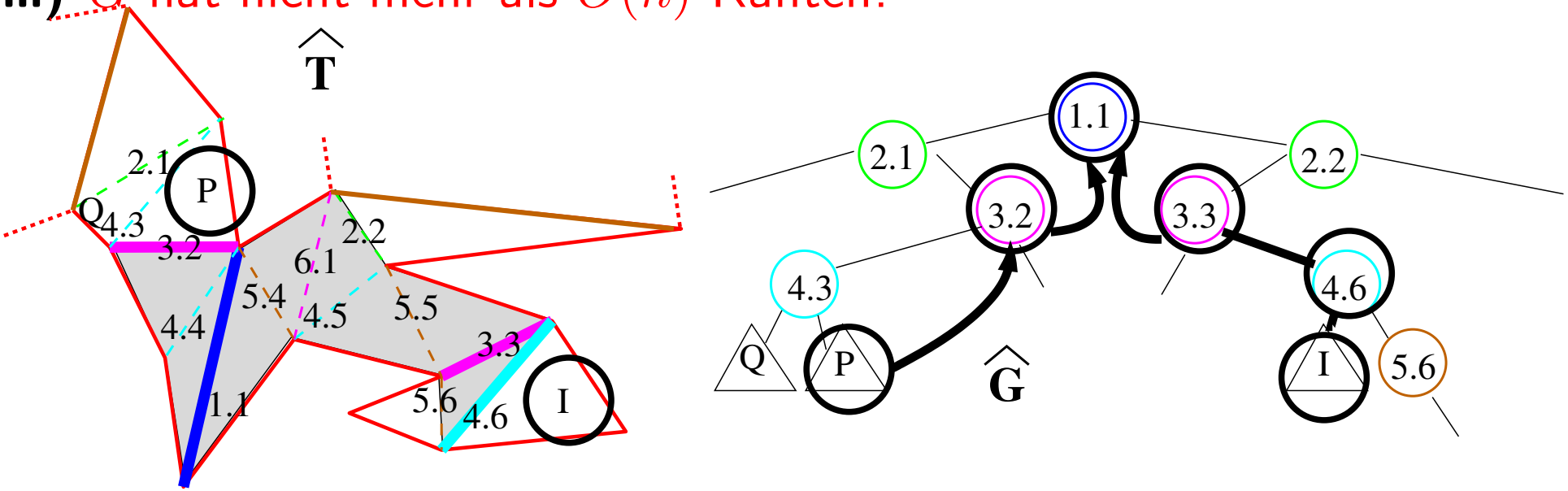
Elmar Langetepe
University of Bonn

# Eigenschaften von $\widehat{G}$: Lemma 1.13

**i)** Pfad zwischen zwei Dreiecken entlang sukzessiver Diagonalen existiert!

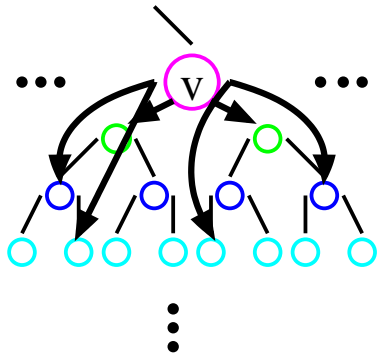
**ii)** Wir finden den Weg in $O(\log n)$ Zeit!

**iii)** $\widehat{G}$ hat nicht mehr als $O(n)$ Kanten!

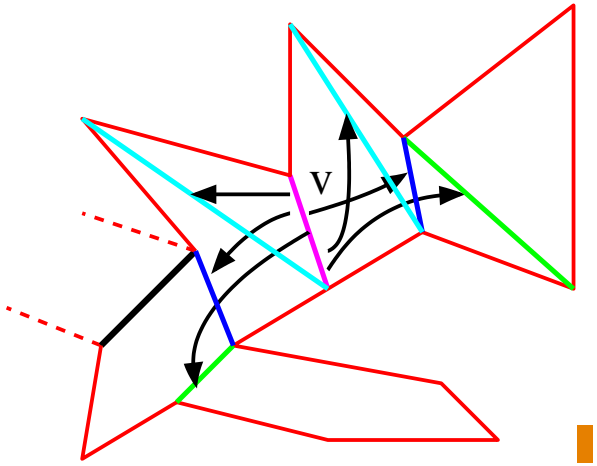


# iii) Komplexität von $\widehat{G}$

- Untere Kanten von $v$ aus: max $2 \times \texttt{height(v)}$
- Balance: Teilbaum bei $v$ hat $\geq C \cdot \left(\frac{3}{2}\right)^{\texttt{height(v)}}$ Blätter (Tafel)




- Anzahl Knoten der Höhe $h$: $\leq \frac{n}{C\left(\frac{3}{2}\right)^h} = \frac{n}{C}\left(\frac{2}{3}\right)^h$
- Sum. über alle Höhen: $\sum_{h=1}^{\log_{\frac{3}{2}} n} (2h) \times \left(\left(\frac{2}{3}\right)^h \times \frac{n}{C}\right) \in O(n)$



# Konstruktion $\widehat{G}$

- Cutting-Theorem (Übung): konstruktiv!!
- Durchlauf von $T^*$ von den Blättern aus!
- Während des Aufbaus: Insgesamt $O(n)$ viele Diagonalen überschreiten?
- Aufbau geht auch in $O(n)$

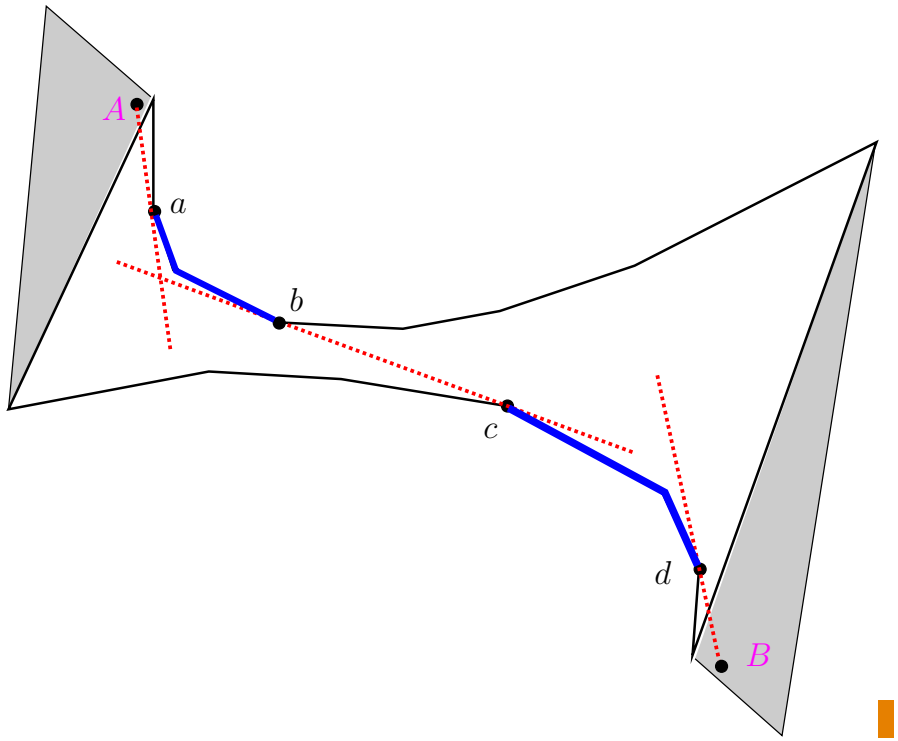
# Zusammenfassung des Problems/Analyse

1. Berechne Triangulation $T$ und Dual $T^*$: $O(n)$
2. Berechne hierarch. bal. Baum $\widehat{T}$, Sch.-Graph $\widehat{G}$: $O(n)$
3. Komplexität $\widehat{G}$: $O(n)$
4. Berechne *alle* Sanduhren von $\widehat{G}$: $O(n)$
5. Navigation zw. Dreiecken in $\widehat{G}$: Sequenz v. Diagonalen: $O(\log n)$
6. Konkat. Sanduhren für finale Sanduhr: $O(\log n)$
7. Berechne Shortest Path aus final. Sanduhr: $O(\log n + k)$

Query: Start $A \in P$, Ziel $B \in I$: Löse 5), 6) und 7)!!

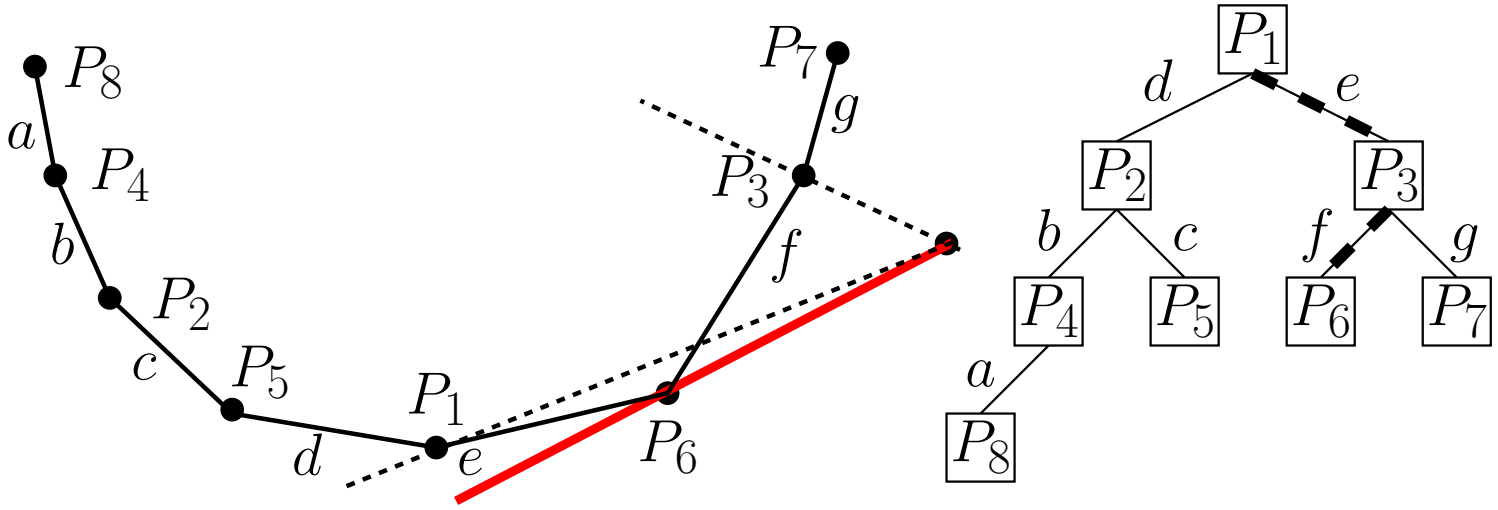
# Berechne Shortest Path aus finaler Sanduhr

- Finale Sanduhr, Ziel und Start
- Data structure: Tangentenpunkte in logarithm. Zeit
- Länge in $O(1)$, Pfad in $O(k)$



# Datenstruktur Hourglass

- Ketten der Sanduhren in bal. Baum speichern
- Tangente in logarithm. Zeit berechnen
- Länge in $O(1)$, Pfad in $O(k)$



# Komplexität und Aufbau aller Sanduhren in $O(n)$

- Sanduhr $S(d_i, d_j)$ mit $d_i$ und $d_j$
- Aus den Sanduhren mit $d'$ in $O(\log(\text{Komp}(S(d_i, d_j))))$
- Entspricht $O(\texttt{height}(\texttt{d}'))$ viele solcher Sanduhren
- Abschätzen durch $O(\texttt{height}(\texttt{d}_\texttt{i}))$ (größere Höhe $d_i$)
- $2 * \texttt{height}(\texttt{d}_\texttt{i})$ viele solcher $d'$ für $d_i$
- Für alle Diagonalen!
- Sum. über alle Höhen: $\displaystyle\sum_{h=1}^{\log_{\frac{3}{2}} n} 2 \times \left( \left( \frac{2}{3} \right)^h \times n \right) \times h^2 \in O(n)$
- **Lemma 1.14**

# Konstruktion $\widehat{G}$ + Sanduhren

- Cutting-Theorem (Übung): konstruktiv!!
- Durchlauf von $T^*$ möglich
- Dabei Sanduhren aufbauen möglich
- Effektiv auch in $O(n)$!

# Sanduhrenlemma: Lemma 1.15

Sanduhr $S(d_i, d_j)$ zwischen $d_i$ und $d_j$ mit $m(d_i, d_j)$ Diagonalen.
Datenstruktur mit folgender Eigenschaft existiert:

**i)** Entfernung zwischen Punkten in $D_i$ und $D_j$ in $O(\log(m(d_i, d_j))$

**ii)** Kürzeste Wege zwischen Punkten in $D_i$ und $D_j$ in $O(\log(m(d_i, d_j)) + k)$

**iii)** Konkatenation zweier Sanduhren $S(d_i, d_j)$ und $S(d_j, d_l)$ zu einer Sanduhr in Zeit $O(\log(m(d_i, d_j)) + \log(m(d_j, d_l)))$

Beweis: Skizze für iii)!!!

# Algorithmus 1.6

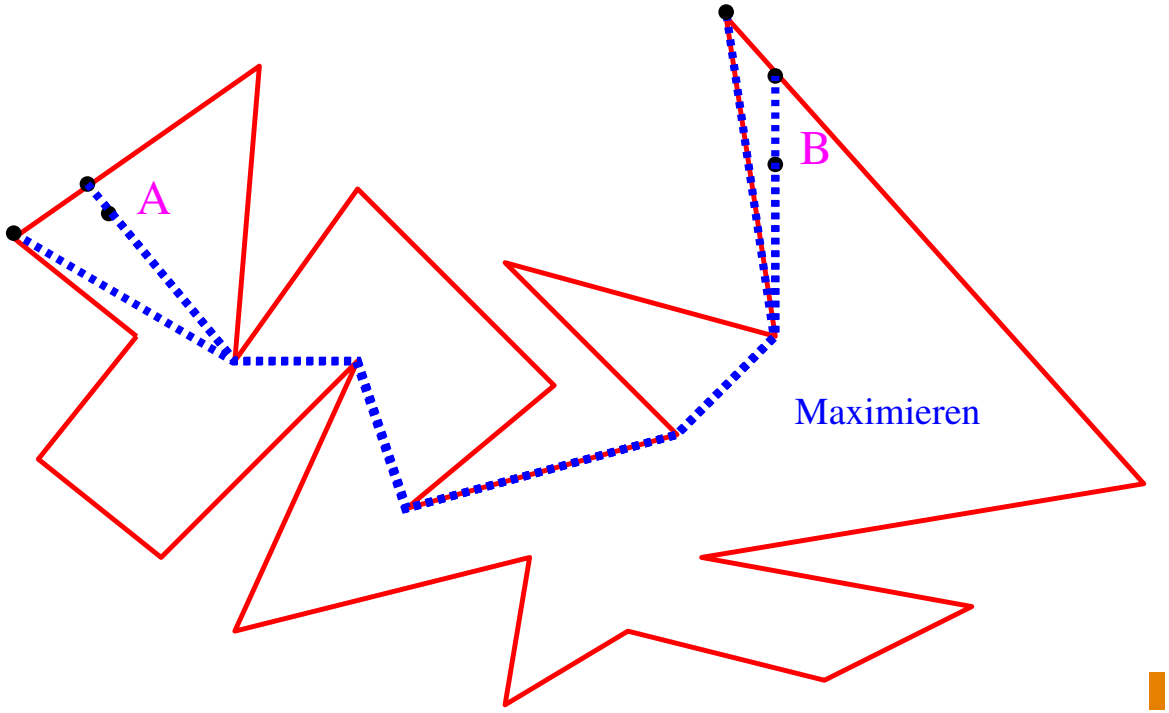
- Preprocessing:
  - $\widehat{G}$ + Rooted Tree
  - Sanduhren für Kanten
  - Lokalisation Dreiecke
- Query, $p$, $q$:
  - Lokalisation $D_p$ und $D_q$
  - Pfad zw. $D_p$ und $D_q$ in $\widehat{G}$
  - Sanduhr $S(d_p, d_q)$ aus Sanduhren entlang des Pfades
  - Länge oder kürzester Weg

# Guibas/Hershberger: Laufzeiten

- Datenstrukturen: $\widehat{G}$, Rooted Tree, Sanduhren, Trapezzerlegung
- Preprocessing Zeit: $O(n)$
- Komplexität: $O(n)$
- Lokalisation Dreiecke: $O(\log n)$
- Pfad in $\widehat{G}$ in $O(\log n)$
- Konkatenation Sanduhren in $O(\log^2 n)$ ($O(\log n)$)
- Query: $O(\log n + k)$ oder $O(\log n)$ für die Länge
- Theorem 1.16

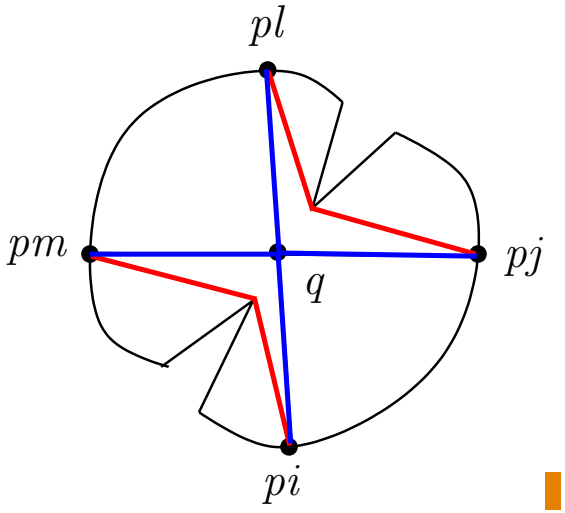
# 1.2.3 Durchmesser einfacher Polygone

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

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

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

# Def. 1.17: Monotone Matrix!

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

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

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

# Linkeste Zeilenmaxima weiter nach rechts

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

# Lemma 1.19: A ist monoton!

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

Beweis!!!