Offline Bewegungsplanung: Preprocessing und Durchmesser

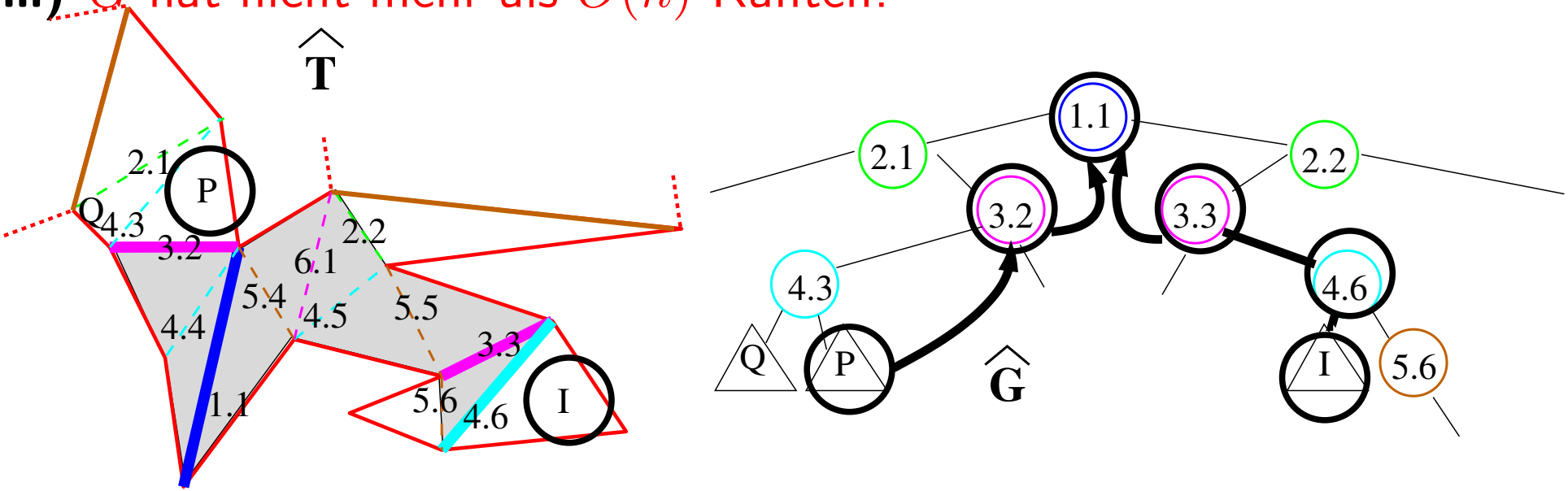
Elmar Langetepe
University of Bonn

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

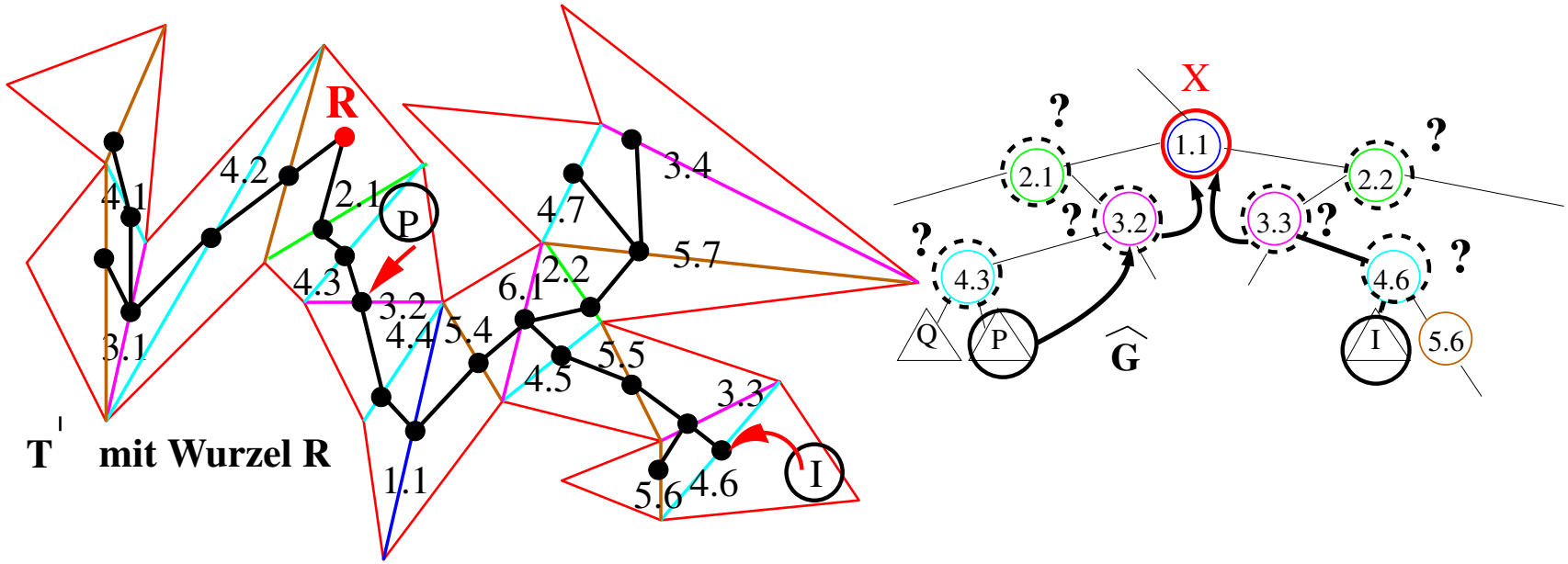
i) Pfad zwischen zwei Dreiecken entlang sukzessiver Diagonalen existiert!

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

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



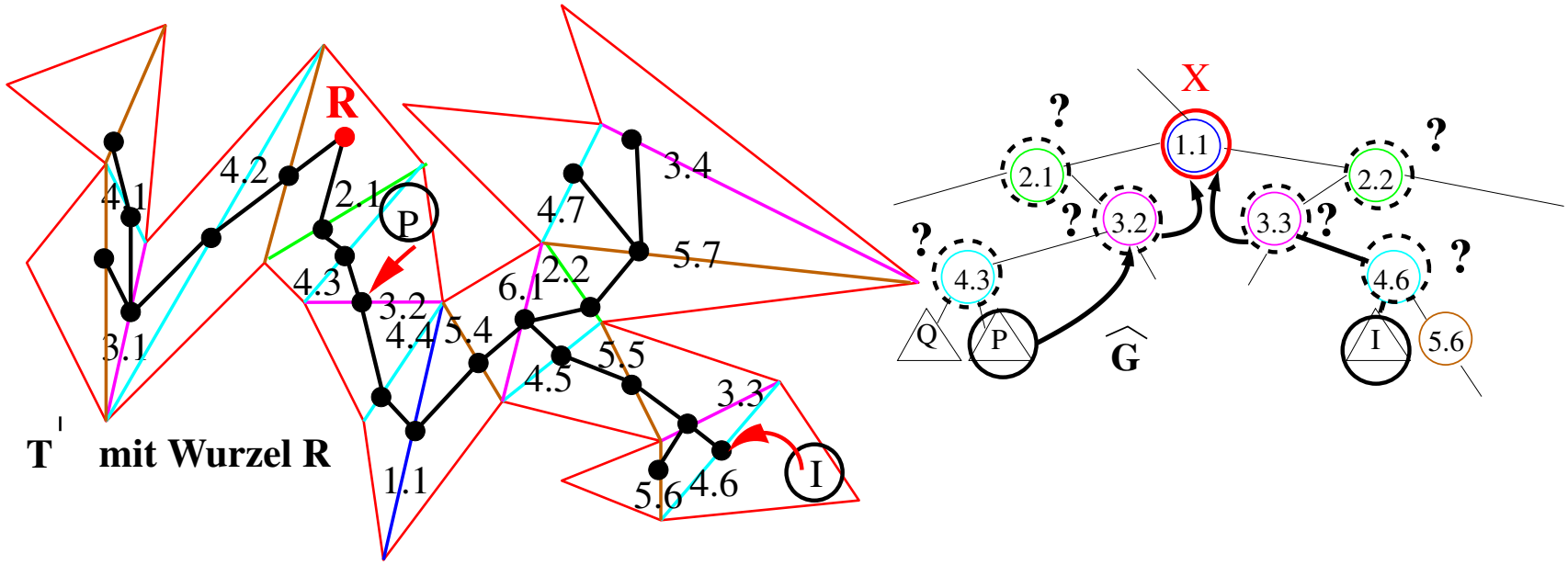
ii) Finden des Weges in $\widehat{G}$!



d Vorgänger entweder von delta(P) ODER delta(I) in Bezug auf R

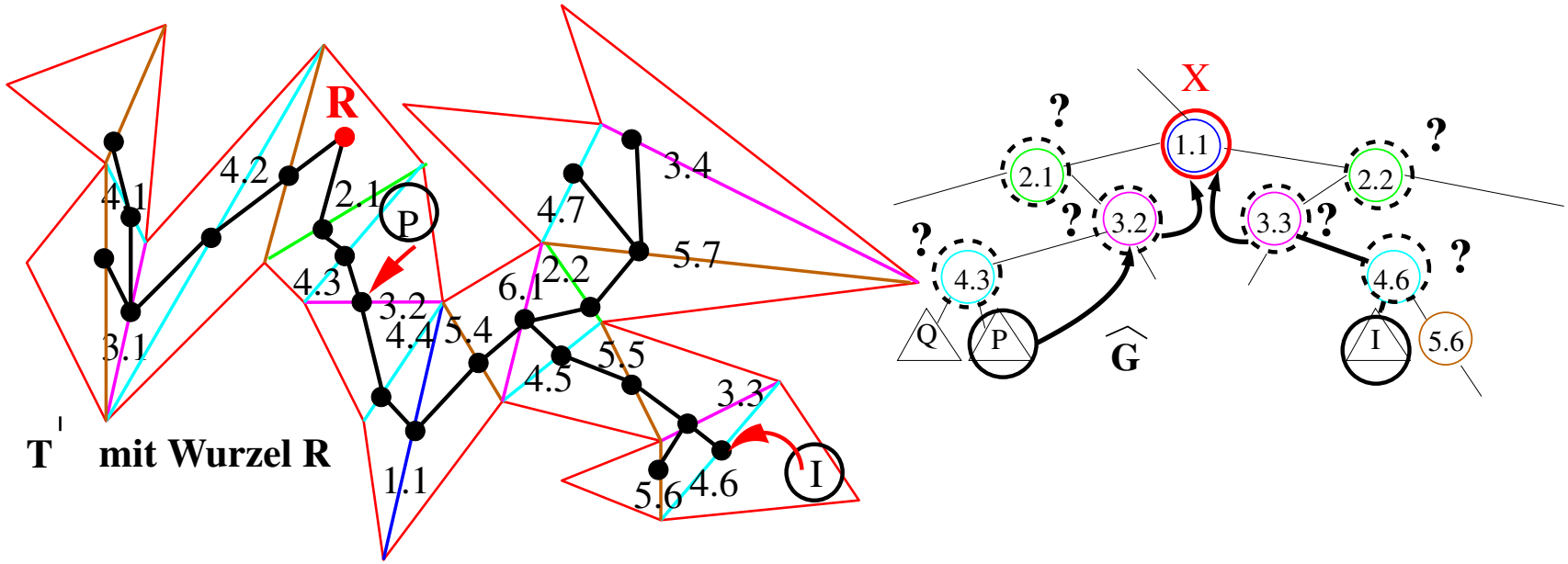
ii) Finden des Weges in $\widehat{G}$!

- Gemeinsamer Vorgänger X minimaler Höhe in $O(\log n)$ in $\widehat{T}$



ii) Finden des Weges in $\widehat{G}$!

- Gemeinsamer Vorgänger X minimaler Höhe in $O(\log n)$ in $\widehat{T}$
- Benutze leicht geänderten Dualen Baum von T
- Frage: Liegt Diagonale d auf dem Pfad von P nach I?

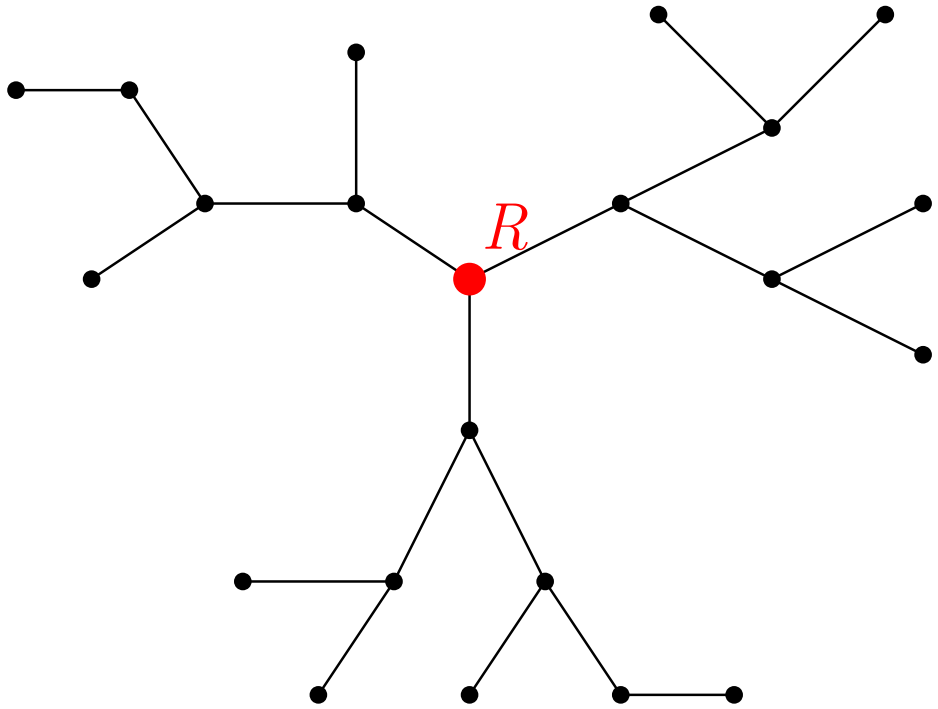


d Vorgänger entweder von delta(P) ODER delta(I) in Bezug auf R

Vorgängeranfrage in Baum

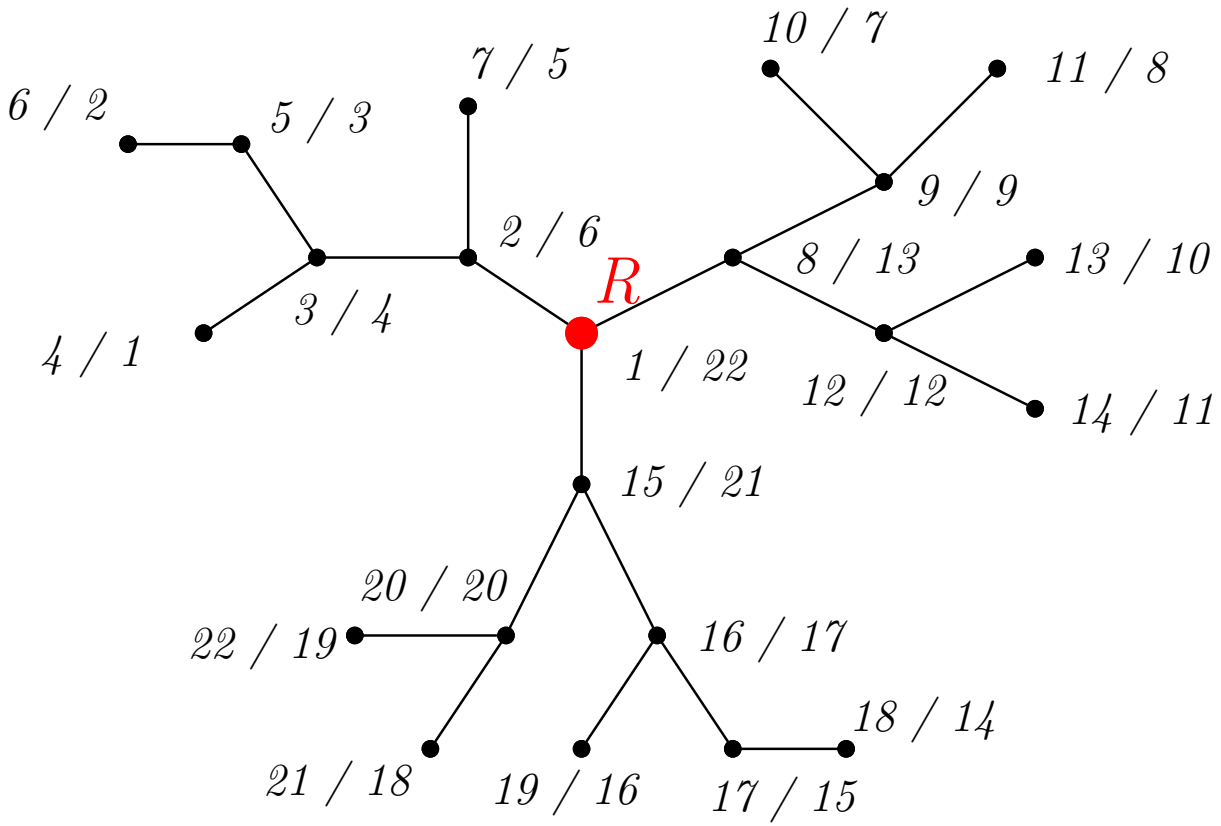
Vorgängeranfrage in Baum

- Preorder/Postorder,



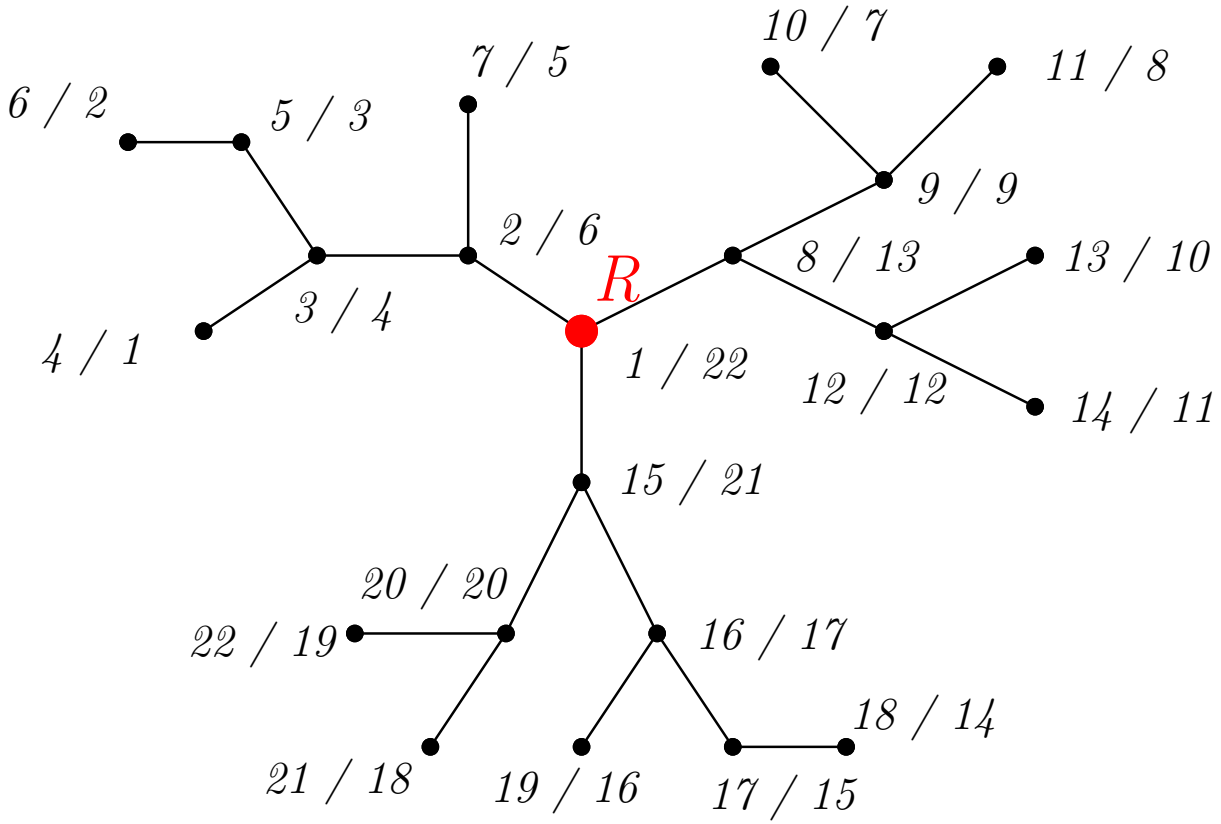
Vorgängeranfrage in Baum

- Preorder/Postorder, DFS und Labelling



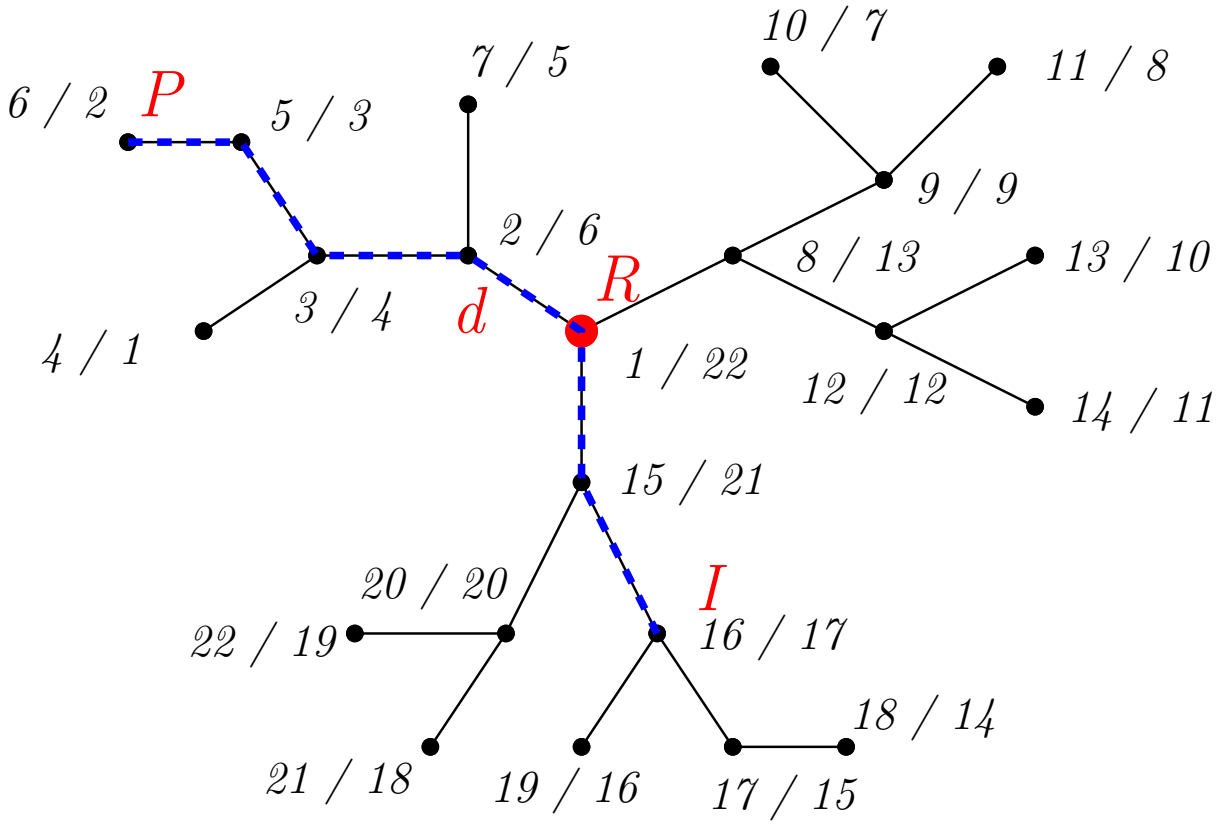
Vorgängeranfrage in Baum

- Preorder/Postorder, DFS und Labelling
- a Vorgänger von b $\Leftrightarrow$ `pre(a)` $<$ `pre(b)` and `post(a)` $>$ `post(b)` (Übung!)



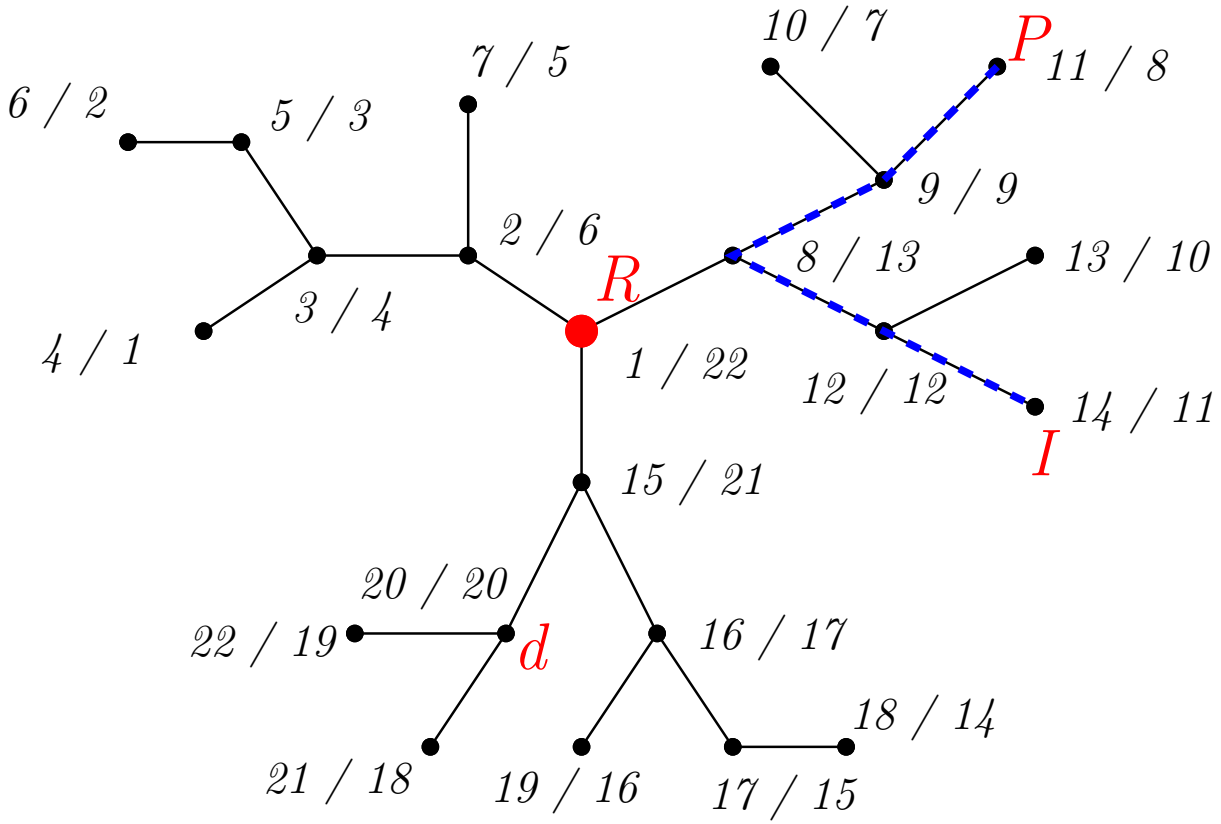
Vorgängeranfrage in Baum

- Preorder/Postorder, DFS und Labelling
- a Vorgänger von b $\Leftrightarrow$ `pre(a)` $<$ `pre(b)` and `post(a)` $>$ `post(b)` (Übung!)



Vorgängeranfrage in Baum

- Preorder/Postorder, DFS und Labelling
- a Vorgänger von b $\Leftrightarrow$ `pre(a)` $<$ `pre(b)` and `post(a)` $>$ `post(b)` (Übung!)



Eigenschaften von $\widehat{G}$

i) Pfad zwischen zwei Dreiecken existiert!

i) Länge ist in $O(\log n)$!

ii) Wir finden den Pfad in $O(\log n)$!

Eigenschaften von $\widehat{G}$

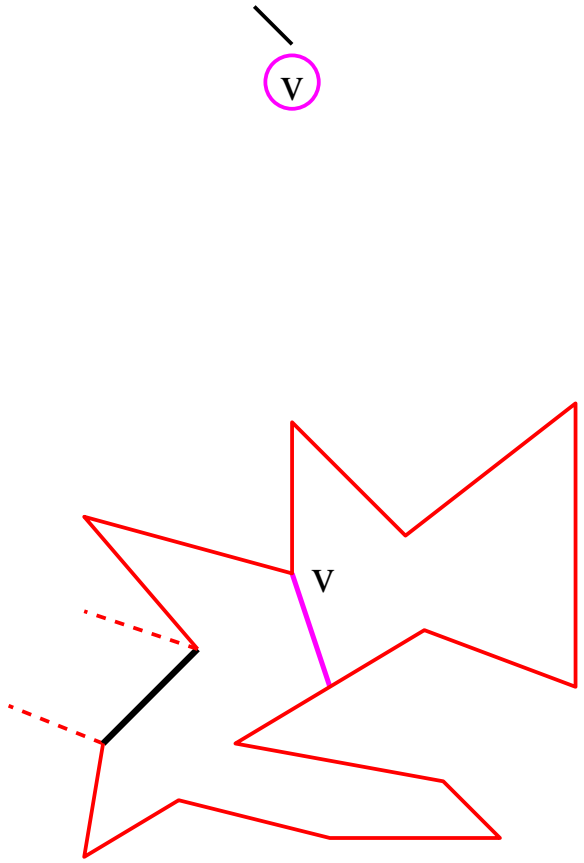
i) Pfad zwischen zwei Dreiecken existiert!

i) Länge ist in $O(\log n)$!

ii) Wir finden den Pfad in $O(\log n)$!

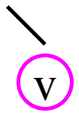
iii) $\widehat{G}$ hat $O(n)$ Kanten !

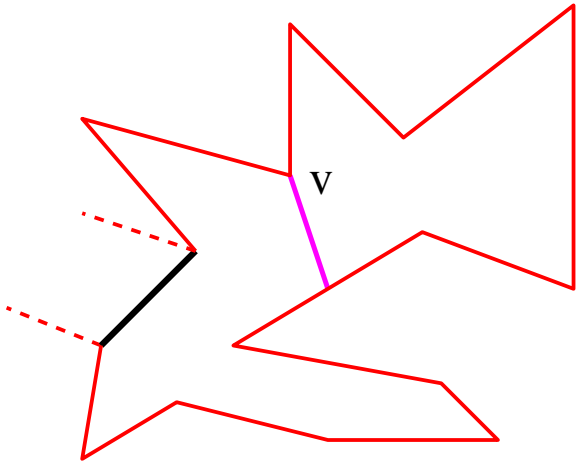
iii) Komplexität von $\widehat{G}$



iii) Komplexität von $\widehat{G}$

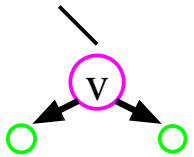
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$

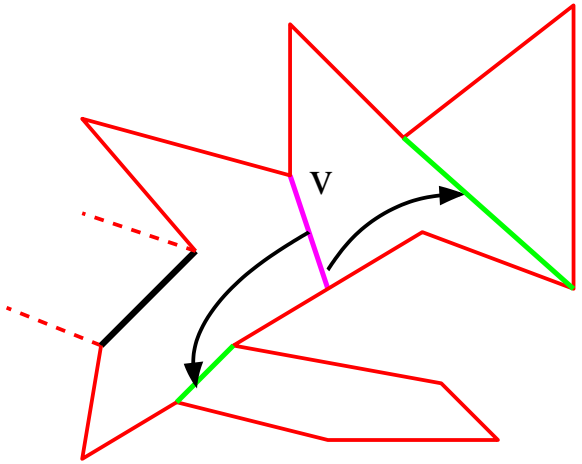




iii) Komplexität von $\widehat{G}$

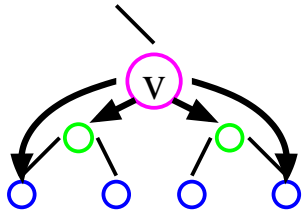
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$

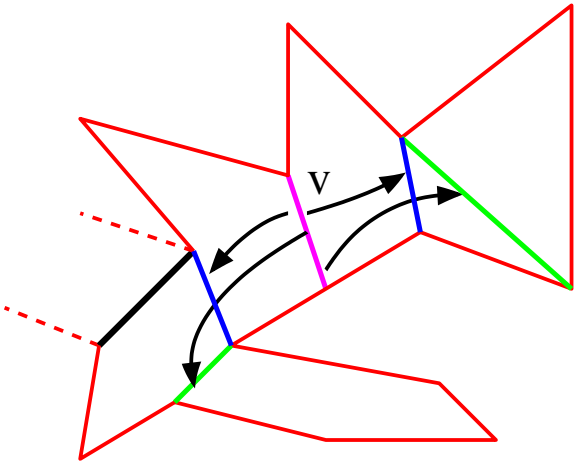




iii) Komplexität von $\widehat{G}$

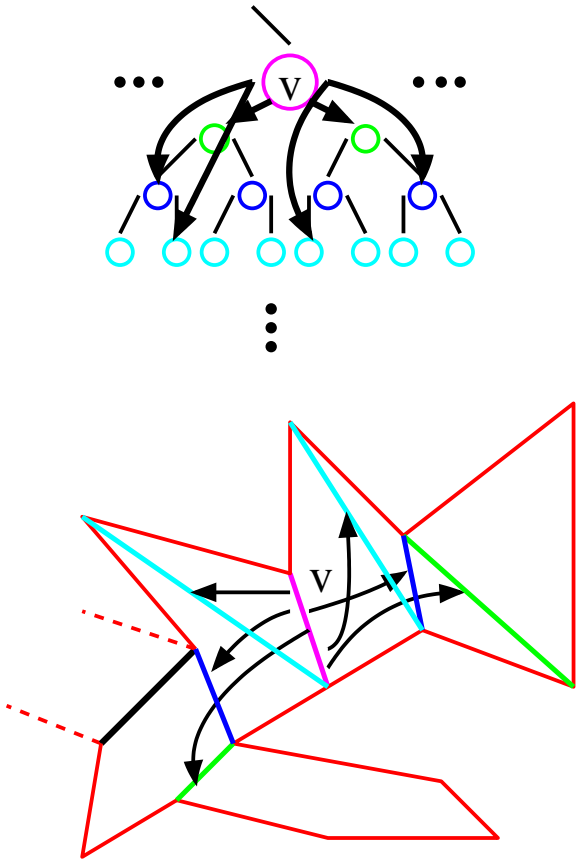
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$





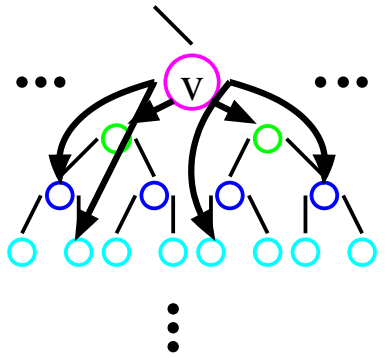
iii) Komplexität von $\widehat{G}$

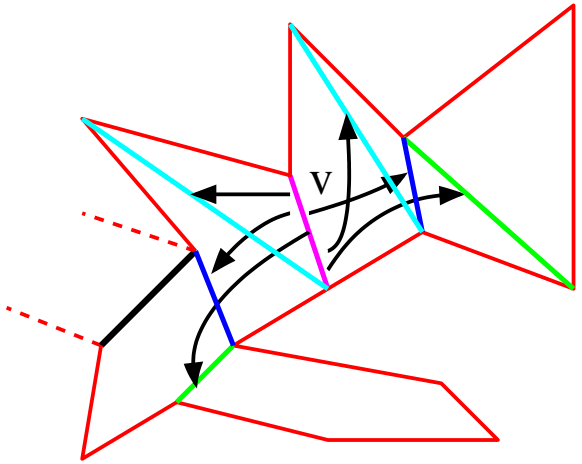
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$



iii) Komplexität von $\widehat{G}$

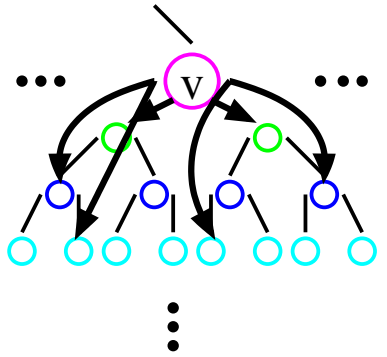
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$
- Balance: Teilbaum bei v hat $\geq C \cdot \left(\frac{3}{2}\right)^{\texttt{height(v)}}$ Blätter (Tafel)

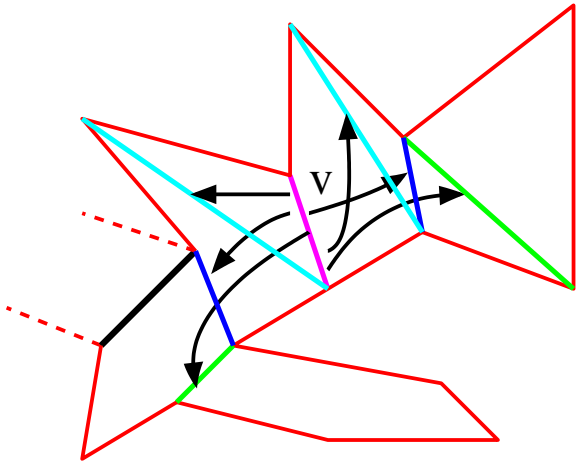




iii) Komplexität von $\widehat{G}$

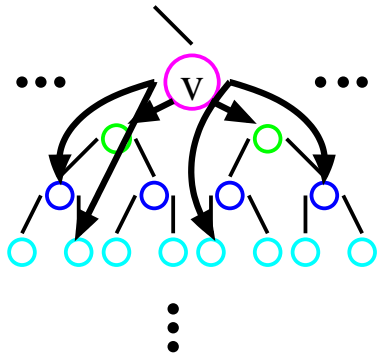
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$
- Balance: Teilbaum bei v hat $\geq C \cdot \left(\frac{3}{2}\right)^{\texttt{height(v)}}$ Blätter (Tafel)
- Anzahl Knoten der Höhe h: $\leq \frac{n}{C\left(\frac{3}{2}\right)^h} = \frac{n}{C}\left(\frac{2}{3}\right)^h$

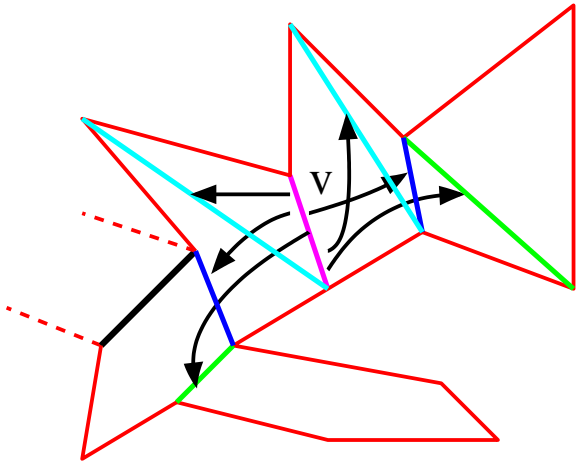




iii) Komplexität von $\widehat{G}$

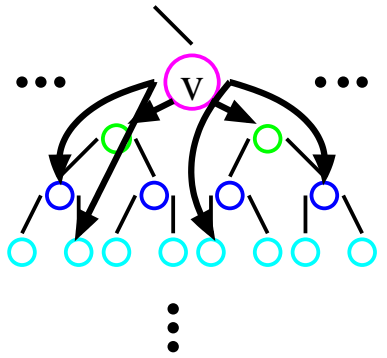
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$
- Balance: Teilbaum bei v hat $\geq C \cdot \left(\frac{3}{2}\right)^{\texttt{height(v)}}$ Blätter (Tafel)
- Anzahl Knoten der Höhe h: $\leq \frac{n}{C\left(\frac{3}{2}\right)^h} = \frac{n}{C}\left(\frac{2}{3}\right)^h$
- Sum. über alle Höhen: $\sum_{h=1}^{\log_{\frac{3}{2}} n} (2h) \times \left(\left(\frac{2}{3}\right)^h \times \frac{n}{C}\right)$

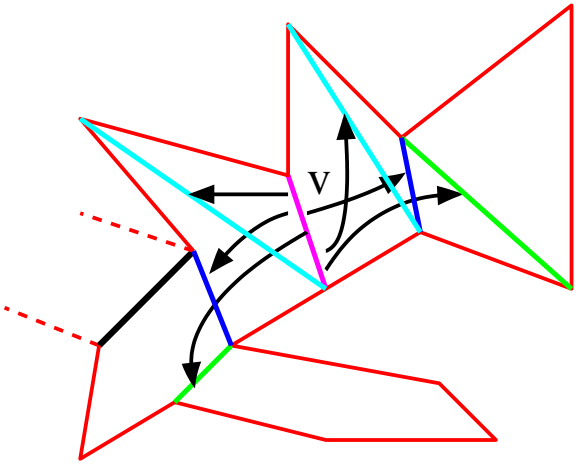




iii) Komplexität von $\widehat{G}$

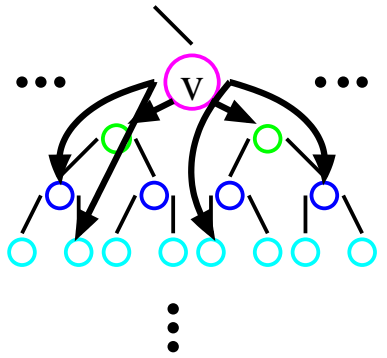
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$
- Balance: Teilbaum bei v hat $\geq C \cdot \left(\frac{3}{2}\right)^{\texttt{height(v)}}$ Blätter (Tafel)
- Anzahl Knoten der Höhe h: $\leq \frac{n}{C\left(\frac{3}{2}\right)^h} = \frac{n}{C}\left(\frac{2}{3}\right)^h$
- Sum. über alle Höhen: $\sum_{h=1}^{\log_{\frac{3}{2}} n} (2h) \times \left(\left(\frac{2}{3}\right)^h \times \frac{n}{C}\right) \in O(n)$

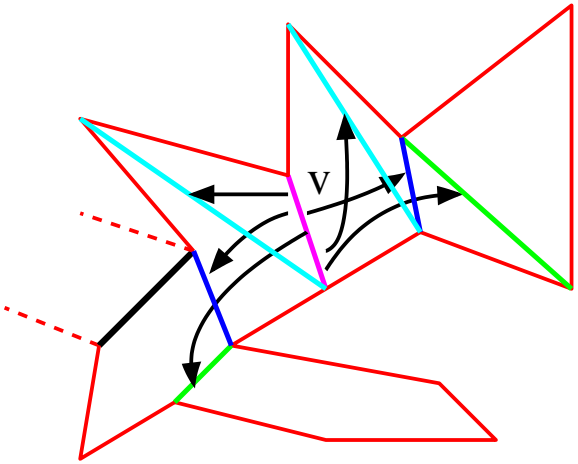




iii) Komplexität von $\widehat{G}$

- 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 $\widehat{G}$

Konstruktion $\widehat{G}$

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

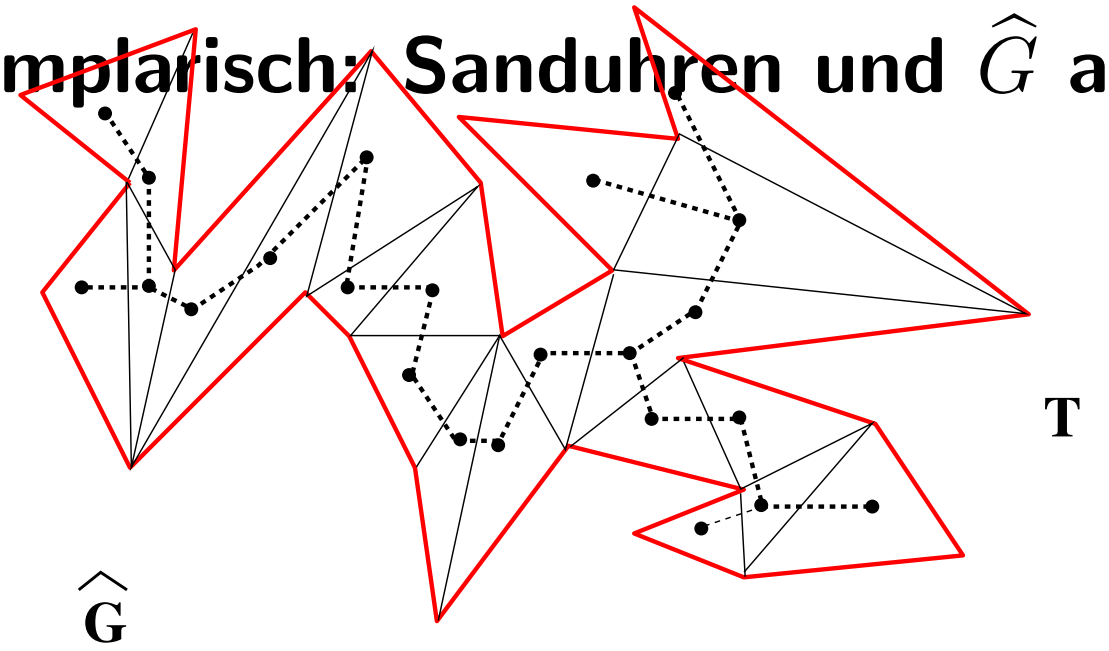
Exemplarisch: Sanduhren und $\widehat{G}$ aufbauen!

Durchlauf gemäß Cutting Theorem Durchlauf

Menge von Listen L_i mit $\sum_i |L_i| \in O(n)$ in Zeit $\sum_i |L_i|$

Aufbau Sanduhren in Zeit $\sum_i |L_i| \in O(n)$

Exemplarisch: Sanduhren und $\widehat{G}$ aufbauen!

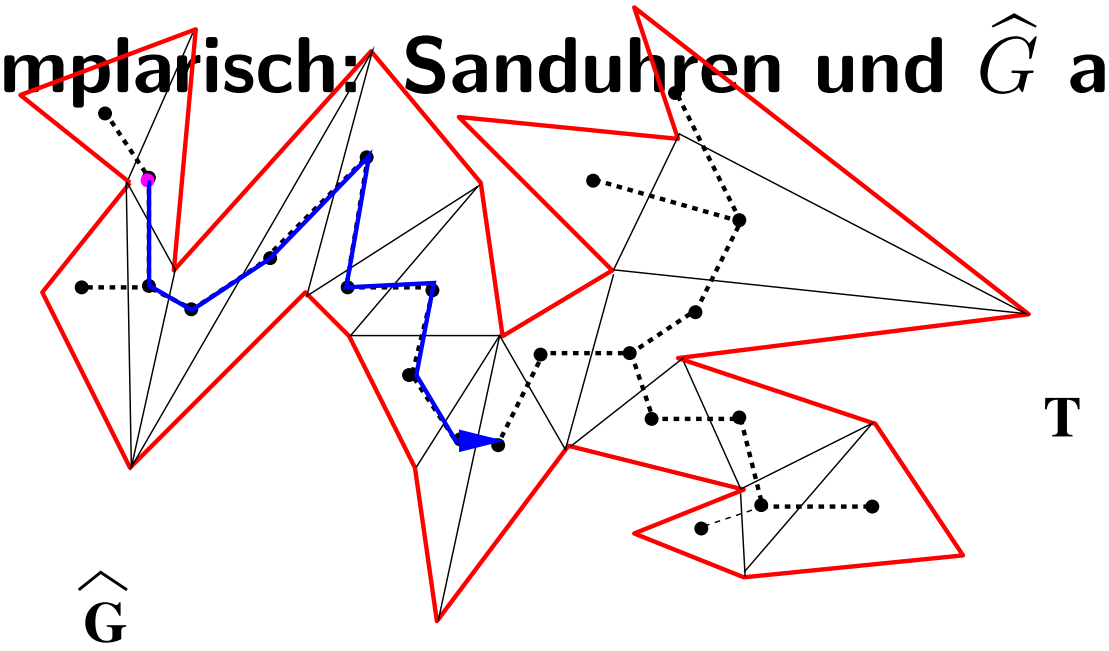


Durchlauf gemäß Cutting Theorem Durchlauf

Menge von Listen L_i mit $\sum_i |L_i| \in O(n)$ in Zeit $\sum_i |L_i|$

Aufbau Sanduhren in Zeit $\sum_i |L_i| \in O(n)$

Exemplarisch: Sanduhren und $\widehat{G}$ aufbauen!

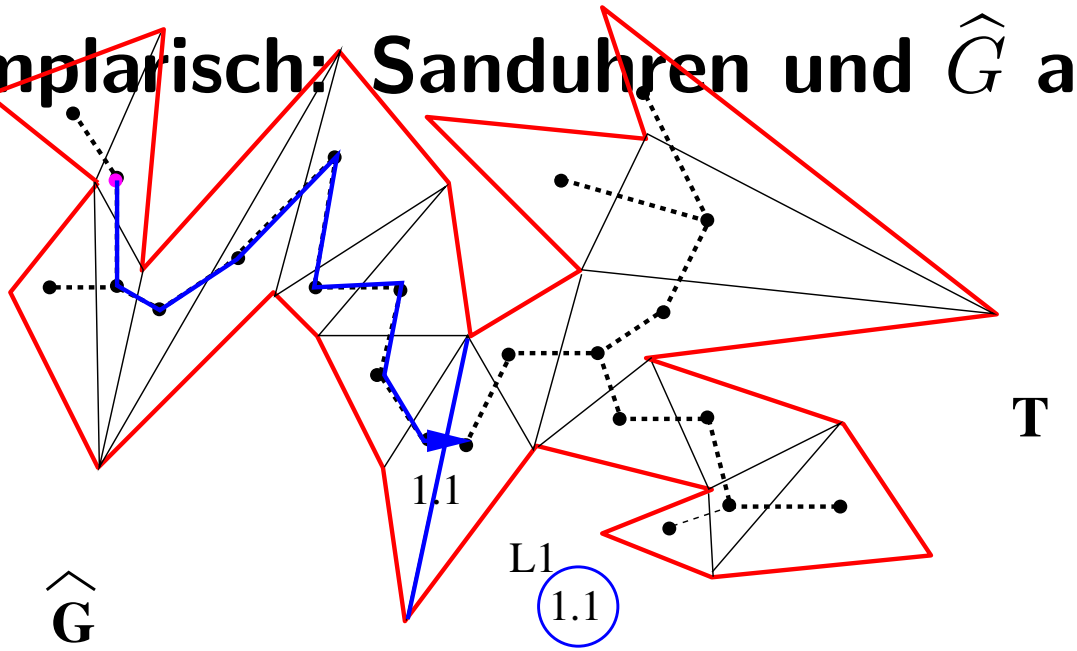


Durchlauf gemäß Cutting Theorem Durchlauf

Menge von Listen L_i mit $\sum_i |L_i| \in O(n)$ in Zeit $\sum_i |L_i|$

Aufbau Sanduhren in Zeit $\sum_i |L_i| \in O(n)$

Exemplarisch: Sanduhren und $\widehat{G}$ aufbauen!

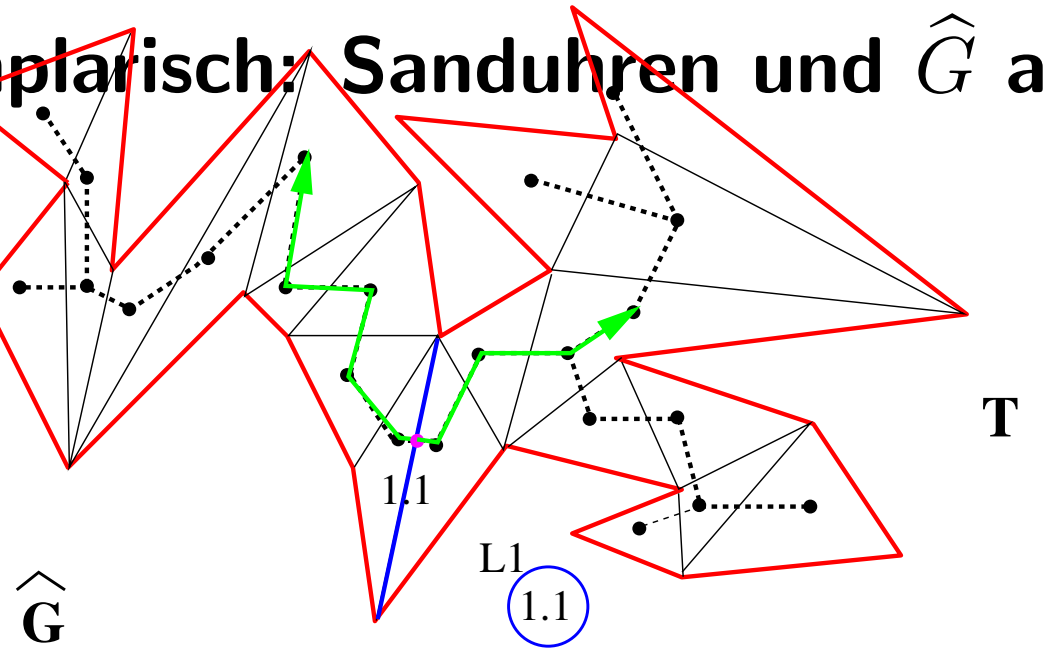


Durchlauf gemäß Cutting Theorem Durchlauf

Menge von Listen L_i mit $\sum_i |L_i| \in O(n)$ in Zeit $\sum_i |L_i|$

Aufbau Sanduhren in Zeit $\sum_i |L_i| \in O(n)$

Exemplarisch: Sanduhren und $\widehat{G}$ aufbauen!

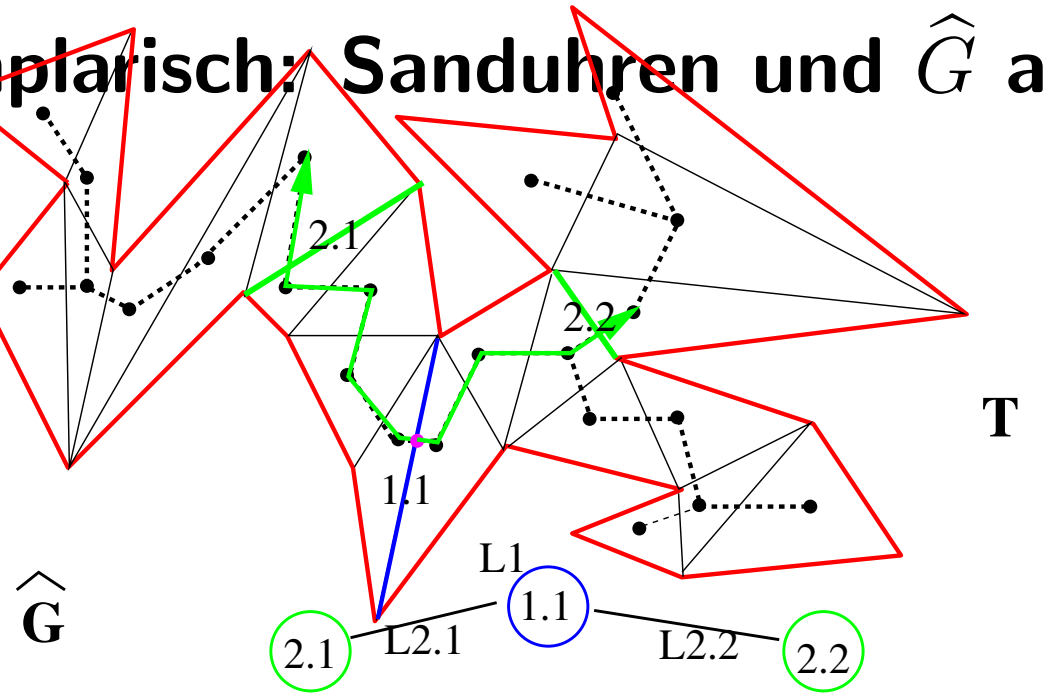


Durchlauf gemäß Cutting Theorem Durchlauf

Menge von Listen L_i mit $\sum_i |L_i| \in O(n)$ in Zeit $\sum_i |L_i|$

Aufbau Sanduhren in Zeit $\sum_i |L_i| \in O(n)$

Exemplarisch: Sanduhren und $\widehat{G}$ aufbauen!

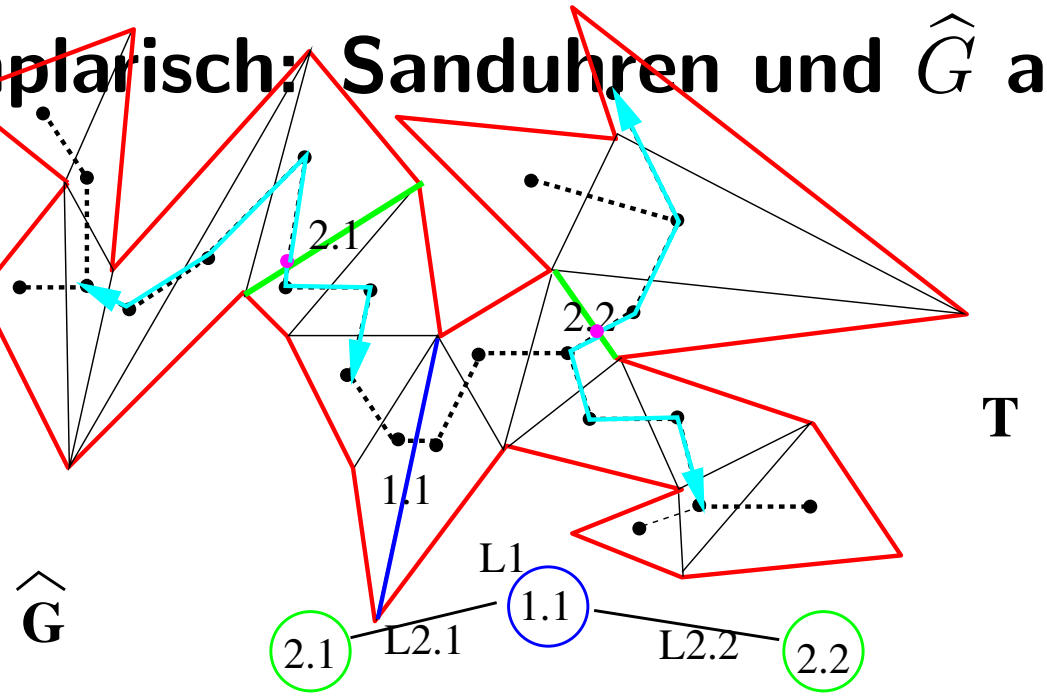


Durchlauf gemäß Cutting Theorem Durchlauf

Menge von Listen L_i mit $\sum_i |L_i| \in O(n)$ in Zeit $\sum_i |L_i|$

Aufbau Sanduhren in Zeit $\sum_i |L_i| \in O(n)$

Exemplarisch: Sanduhren und $\widehat{G}$ aufbauen!

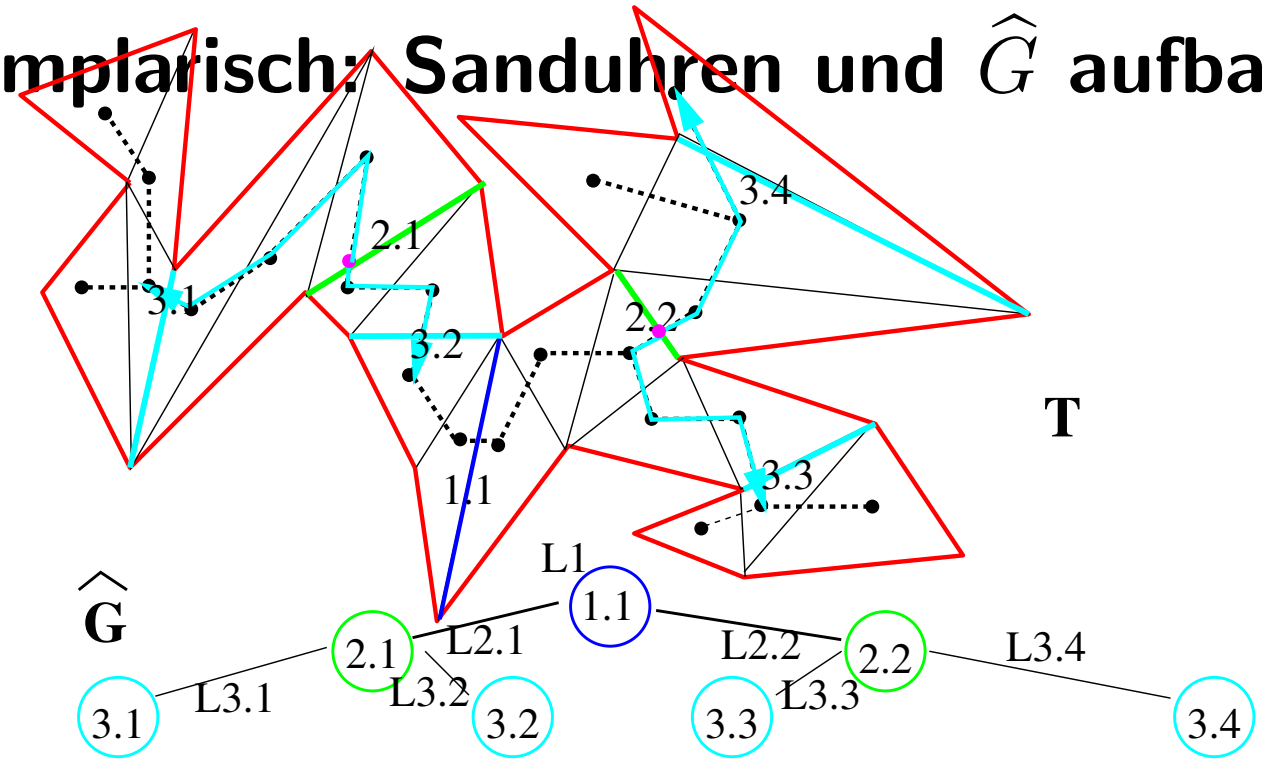


Durchlauf gemäß Cutting Theorem Durchlauf

Menge von Listen L_i mit $\sum_i |L_i| \in O(n)$ in Zeit $\sum_i |L_i|$

Aufbau Sanduhren in Zeit $\sum_i |L_i| \in O(n)$

Exemplarisch: Sanduhren und $\widehat{G}$ aufbauen!

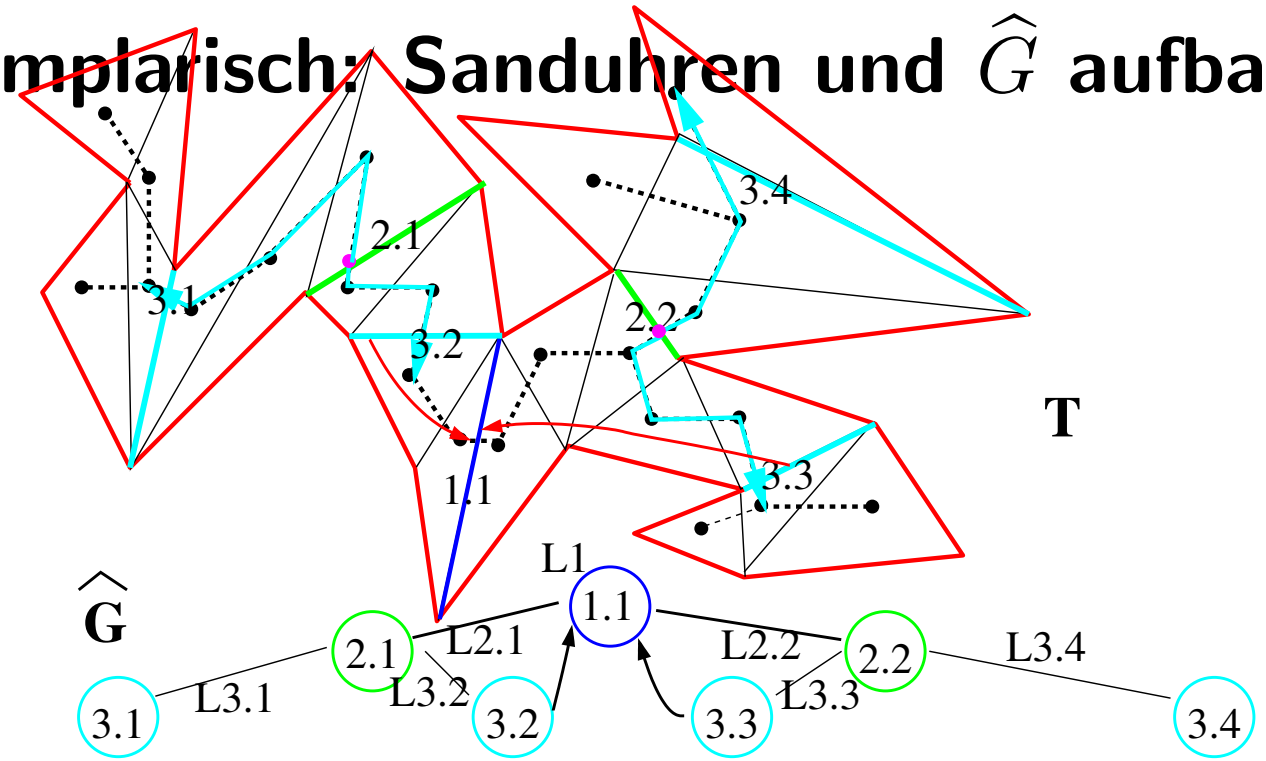


Durchlauf gemäß Cutting Theorem Durchlauf

Menge von Listen L_i mit $\sum_i |L_i| \in O(n)$ in Zeit $\sum_i |L_i|$

Aufbau Sanduhren in Zeit $\sum_i |L_i| \in O(n)$

Exemplarisch: Sanduhren und $\widehat{G}$ aufbauen!

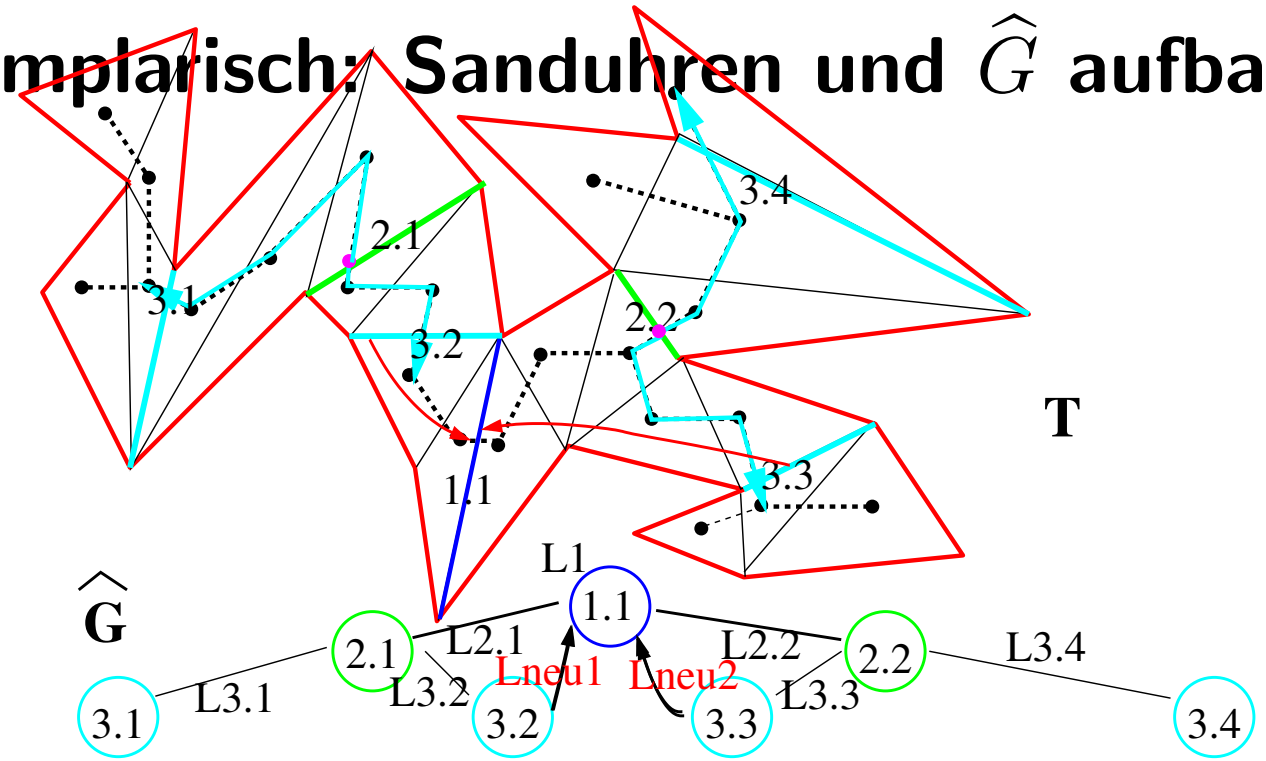


Durchlauf gemäß Cutting Theorem Durchlauf

Menge von Listen L_i mit $\sum_i |L_i| \in O(n)$ in Zeit $\sum_i |L_i|$

Aufbau Sanduhren in Zeit $\sum_i |L_i| \in O(n)$

Exemplarisch: Sanduhren und $\widehat{G}$ aufbauen!

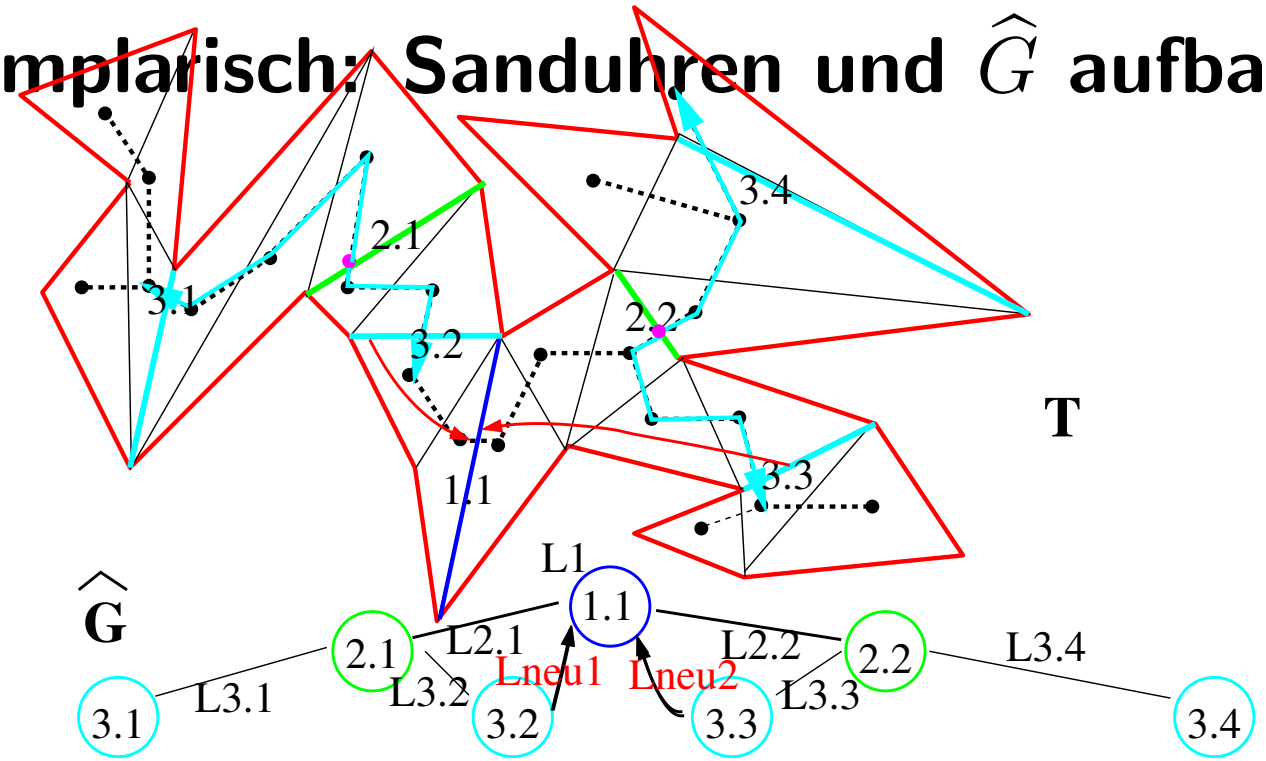


Durchlauf gemäß Cutting Theorem Durchlauf

Menge von Listen L_i mit $\sum_i |L_i| \in O(n)$ in Zeit $\sum_i |L_i|$

Aufbau Sanduhren in Zeit $\sum_i |L_i| \in O(n)$

Exemplarisch: Sanduhren und $\widehat{G}$ aufbauen!



Durchlauf gemäß Cutting Theorem Durchlauf

Menge von Listen L_i mit $\sum_i |L_i| \in O(n)$ in Zeit $\sum_i |L_i|$

Aufbau Sanduhren in Zeit $\sum_i |L_i| \in O(n)$

Zusammenfassung des Problems/Analyse

Zusammenfassung des Problems/Analyse

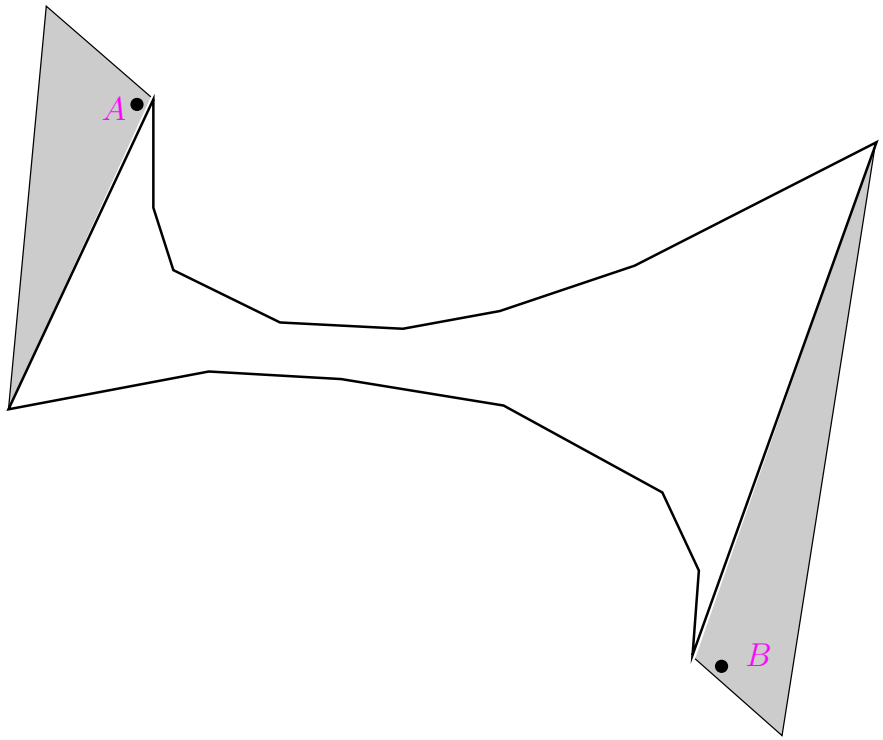
1. Berechne Triangulation T und Dual T^*: $O(n)$
2. Berechne hierarch. bal. Baum $\widehat{T}$, Sch.-Graph $\widehat{G}$: $O(n)$
3. Komplexität $\widehat{G}$: $O(n)$
4. Berechne *alle* Sanduhren von $\widehat{G}$: $O(n)$
5. Navigation zw. Dreiecken in $\widehat{G}$: Sequenz v. Diagonalen: $O(\log n)$
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

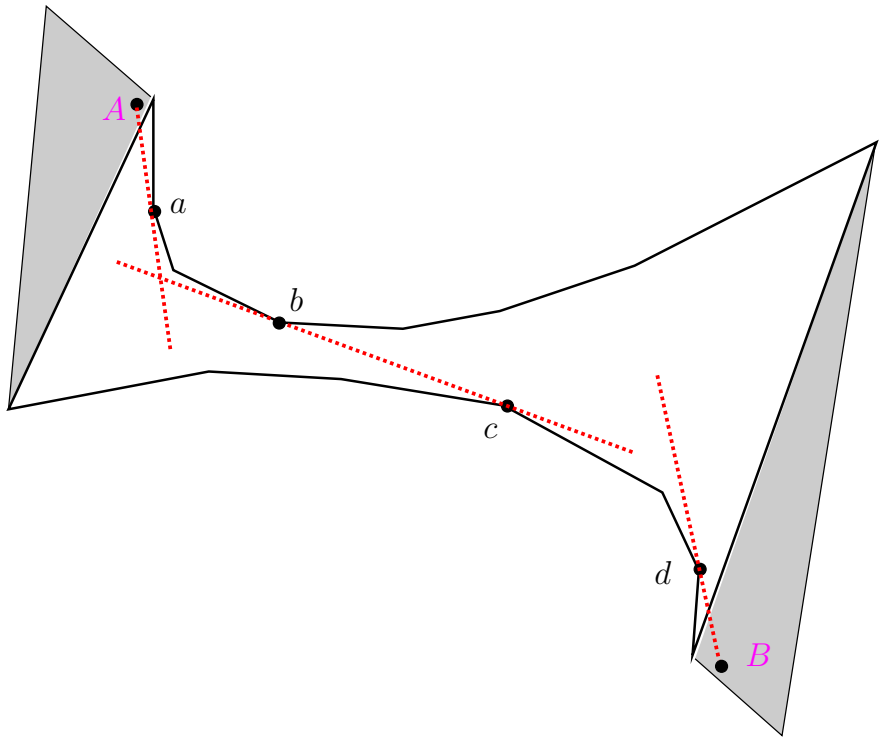
Berechne Shortest Path aus finaler Sanduhr

- Finale Sanduhr, Ziel und Start



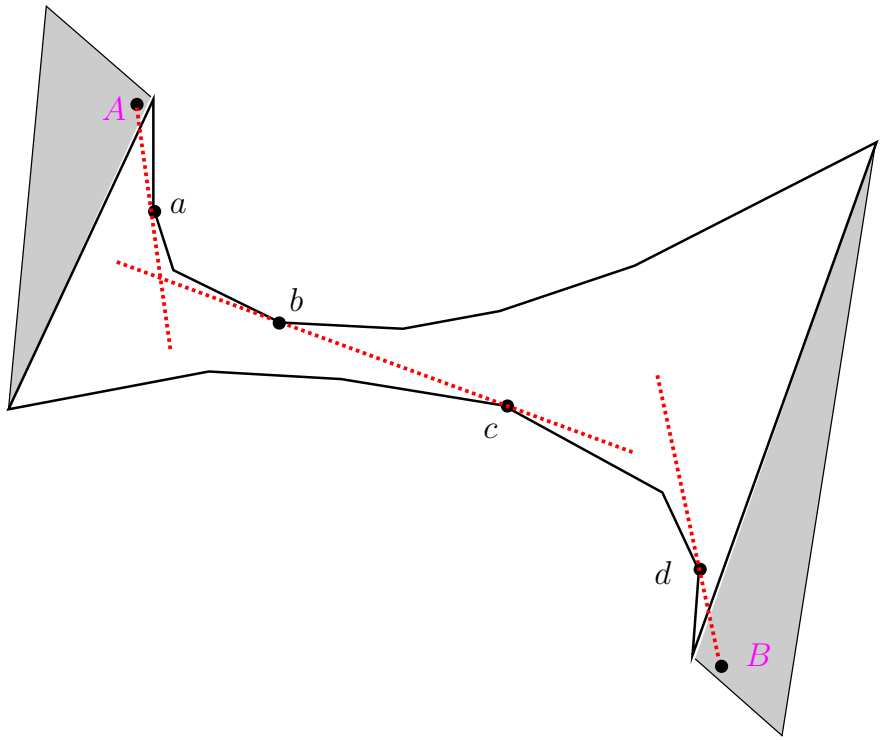
Berechne Shortest Path aus finaler Sanduhr

- Finale Sanduhr, Ziel und Start
- Data structure: Tangentenpunkte in logarithm. Zeit



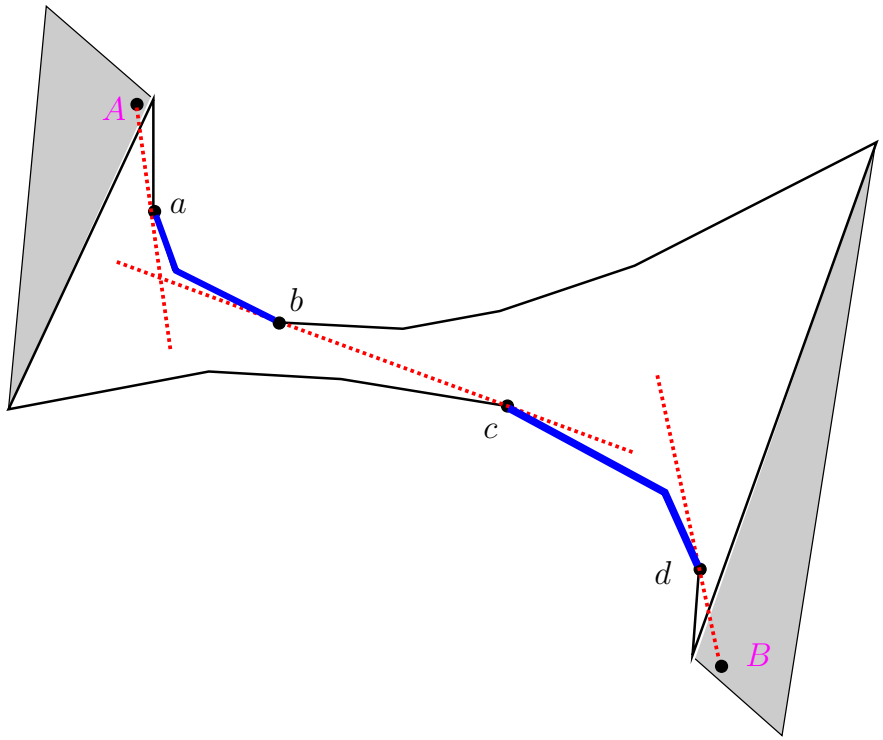
Berechne Shortest Path aus finaler Sanduhr

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

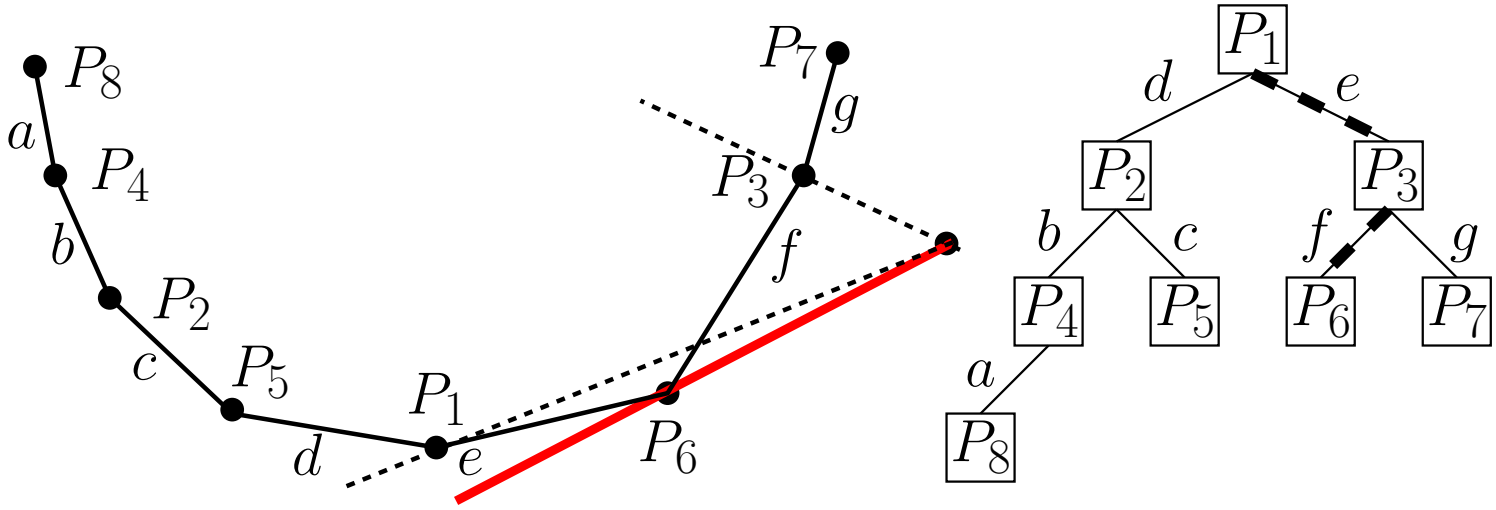


Berechne Shortest Path aus finaler Sanduhr

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

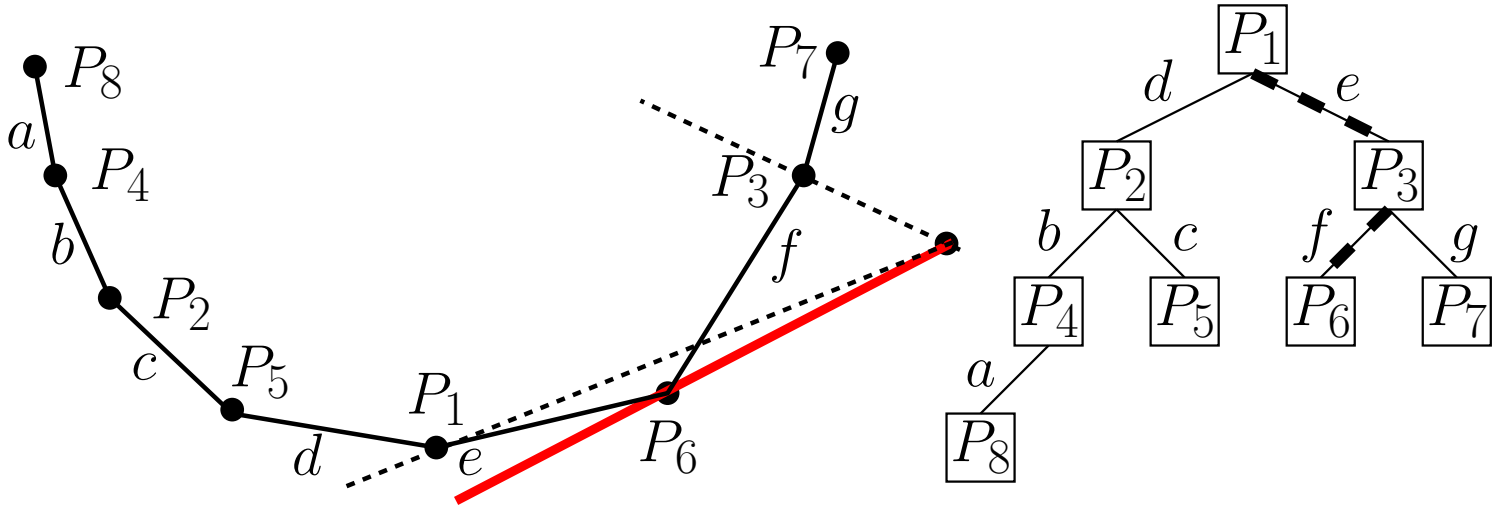


Datenstruktur Hourglass



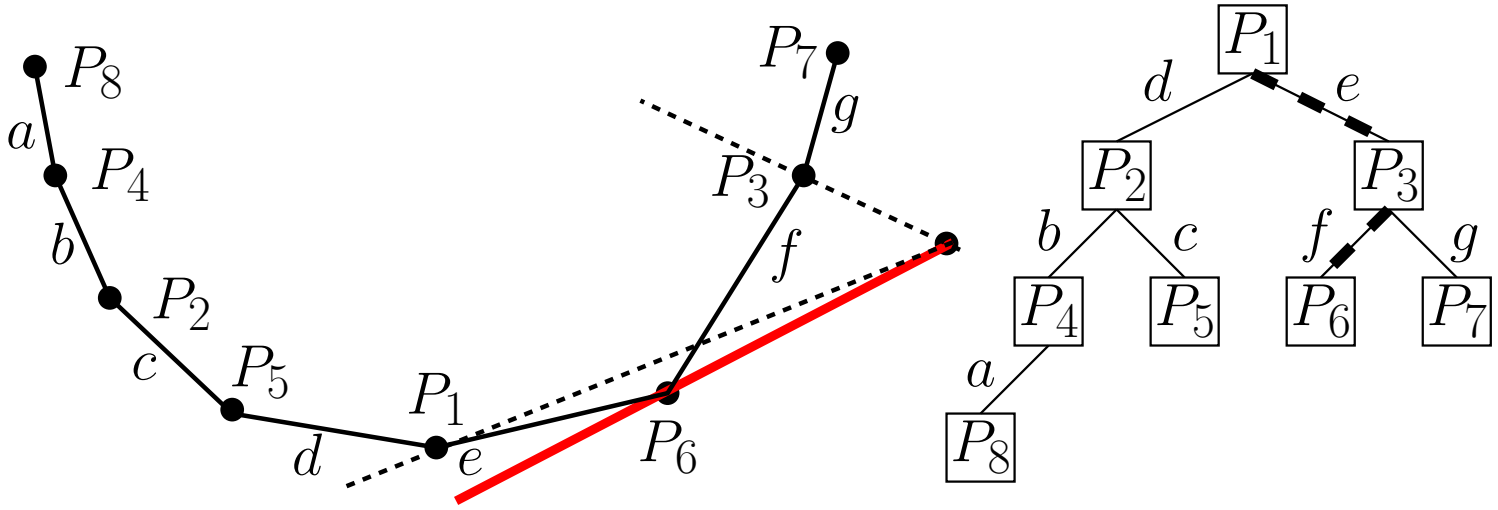
Datenstruktur Hourglass

- Ketten der Sanduhren in bal. Baum speichern



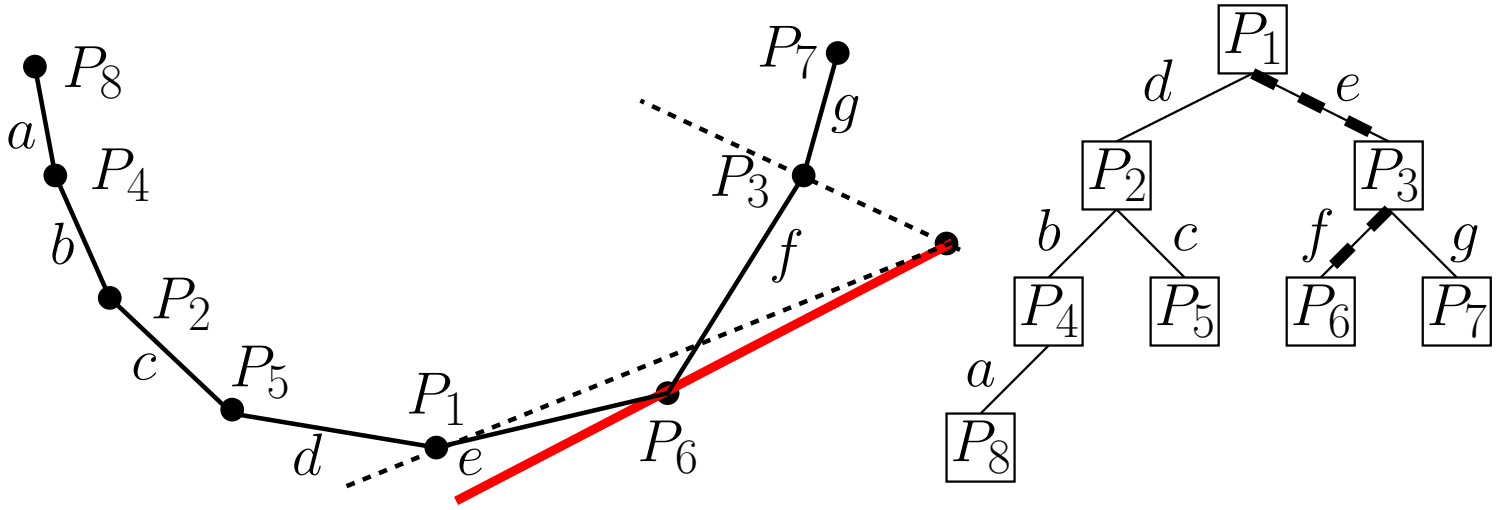
Datenstruktur Hourglass

- Ketten der Sanduhren in bal. Baum speichern
- Tangente in logarithm. Zeit berechnen



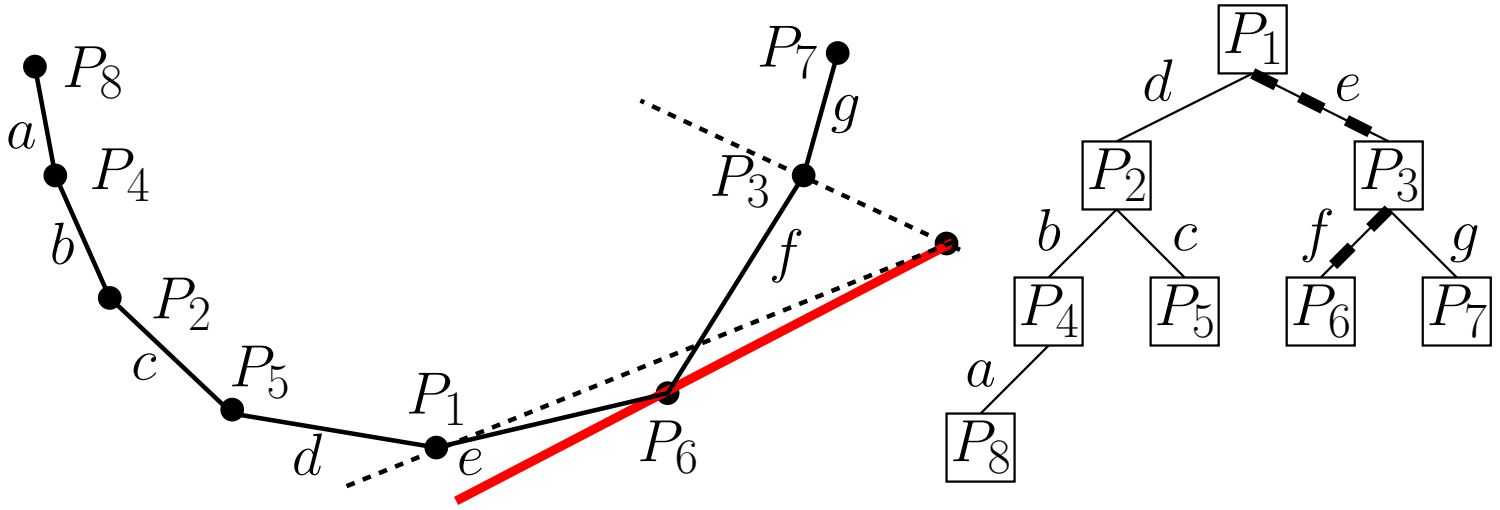
Datenstruktur Hourglass

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



Datenstruktur Hourglass

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



Komplexität aller Sanduhren in $O(n)$

Komplexität aller Sanduhren in $O(n)$

- Sanduhr mit Rand d_i benutzt d_j

Komplexität aller Sanduhren in $O(n)$

- Sanduhr mit Rand d_i benutzt d_j
- d_j liegt in $\widehat{G}$ auf Pfad von d_i zu d_j

Komplexität aller Sanduhren in $O(n)$

- Sanduhr mit Rand d_i benutzt d_j
- d_j liegt in $\widehat{G}$ auf Pfad von d_i zu d_j
- Verwendung in Sanduhren mit d_j: Länge dieses Pfades mal

Komplexität aller Sanduhren in $O(n)$

- Sanduhr mit Rand d_i benutzt d_j
- d_j liegt in $\widehat{G}$ auf Pfad von d_i zu d_j
- Verwendung in Sanduhren mit d_j: Länge dieses Pfades mal
- $\sum_{i=1}^{\texttt{height(d}_\texttt{i}\texttt{)}} i$:

Komplexität aller Sanduhren in $O(n)$

- Sanduhr mit Rand d_i benutzt d_j
- d_j liegt in $\widehat{G}$ auf Pfad von d_i zu d_j
- Verwendung in Sanduhren mit d_j: Länge dieses Pfades mal
- $\sum_{i=1}^{\texttt{height}(\texttt{d}_\texttt{i})} i$: $O(\texttt{height}(\texttt{d}_\texttt{i})^2)$

Komplexität aller Sanduhren in $O(n)$

- Sanduhr mit Rand d_i benutzt d_j
- d_j liegt in $\widehat{G}$ auf Pfad von d_i zu d_j
- Verwendung in Sanduhren mit d_j: Länge dieses Pfades mal
- $\sum_{i=1}^{\texttt{height}(\texttt{d\_i})} i$: $O(\texttt{height}(\texttt{d\_i})^2)$
- Auch zur anderen Seite: 2 mal

Komplexität aller Sanduhren in $O(n)$

- Sanduhr mit Rand d_i benutzt d_j
- d_j liegt in $\widehat{G}$ auf Pfad von d_i zu d_j
- Verwendung in Sanduhren mit d_j: Länge dieses Pfades mal
- $\sum_{i=1}^{\mathtt{height(d_i)}} i$: $O(\mathtt{height(d_i)}^2)$
- Auch zur anderen Seite: 2 mal
- Sum. über alle Höhen: $\displaystyle\sum_{h=1}^{\log_{\frac{3}{2}} n} 2 \times \left(\left(\frac{2}{3} \right)^h \times n \right) \times h^2$

Komplexität aller Sanduhren in $O(n)$

- Sanduhr mit Rand d_i benutzt d_j
- d_j liegt in $\widehat{G}$ auf Pfad von d_i zu d_j
- Verwendung in Sanduhren mit d_j: Länge dieses Pfades mal
- $\sum_{i=1}^{\mathtt{height(d_i)}} i$: $O(\mathtt{height(d_i)}^2)$
- Auch zur anderen Seite: 2 mal
- Sum. über alle Höhen: $\displaystyle\sum_{h=1}^{\log_{\frac{3}{2}} n} 2 \times \left(\left(\frac{2}{3} \right)^h \times n \right) \times h^2 \in O(n)$

Komplexität aller Sanduhren in $O(n)$

- Sanduhr mit Rand d_i benutzt d_j
- d_j liegt in $\widehat{G}$ auf Pfad von d_i zu d_j
- Verwendung in Sanduhren mit d_j: Länge dieses Pfades mal
- $\sum_{i=1}^{\mathtt{height(d_i)}} i$: $O(\mathtt{height(d_i)}^2)$
- Auch zur anderen Seite: 2 mal
- 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^*
- Während des Aufbaus: $O(n)$ viele Diagonalen überschreiten
- Dabei Sanduhren aufbauen

Sanduhrenlemma: Lemma 1.15

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 $O(\log(m(d_i, d_j))$

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 $O(\log(m(d_i, d_j)) + k)$

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 $O(\log(m(d_i, d_j)) + \log(m(d_j, d_l)))$

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

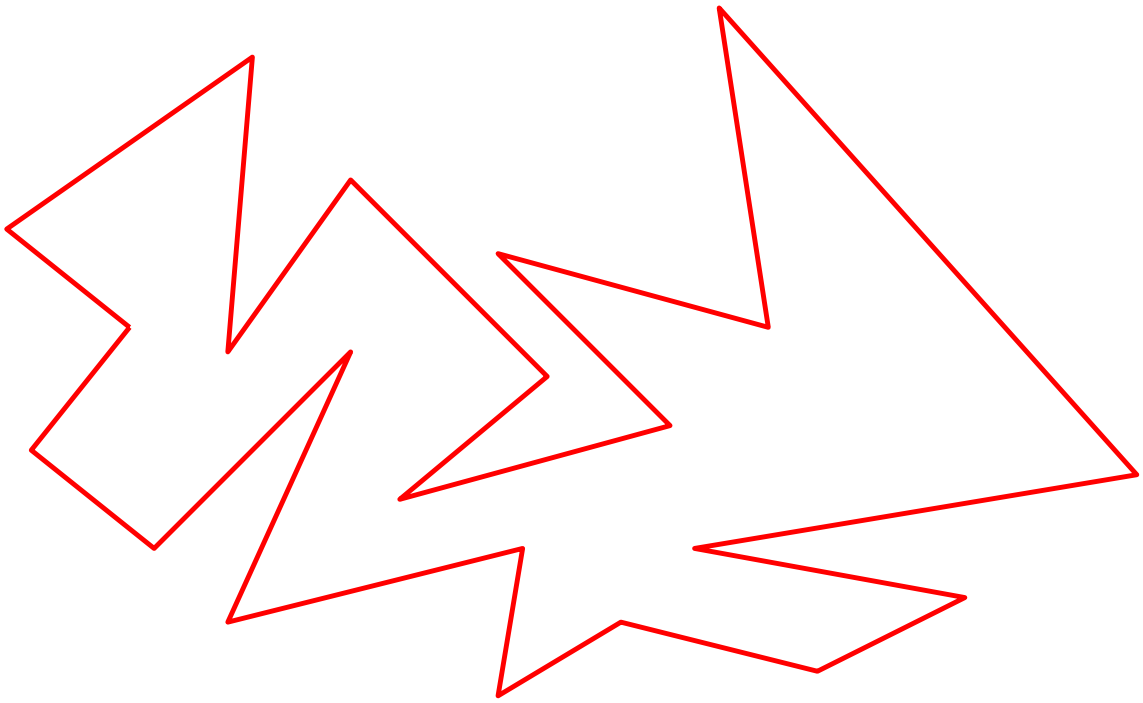
Guibas/Hershberger: Laufzeiten

- 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

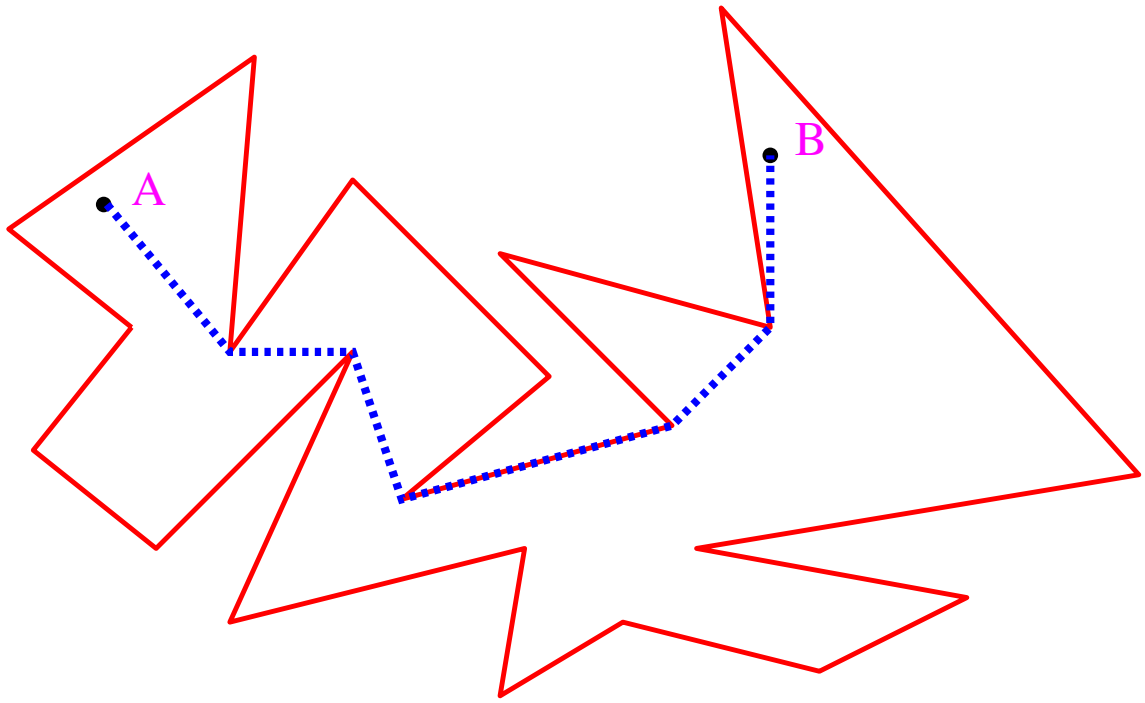
1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon P



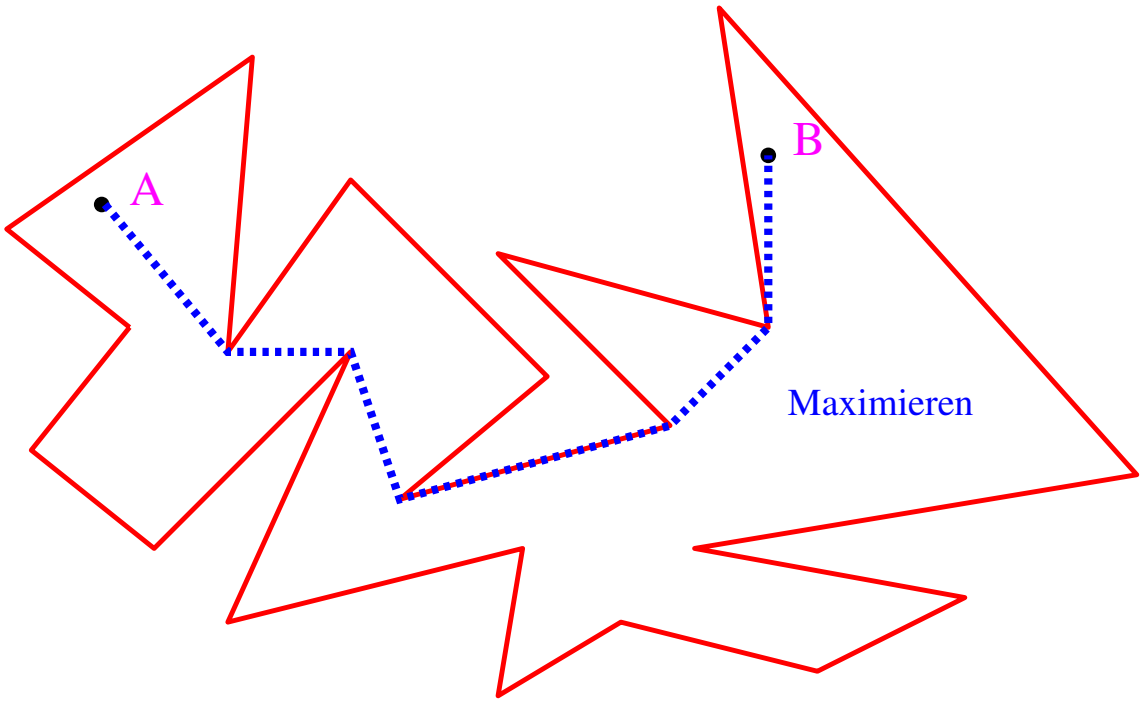
1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon P
- Längster Kürzester Weg zwischen zwei Punkten



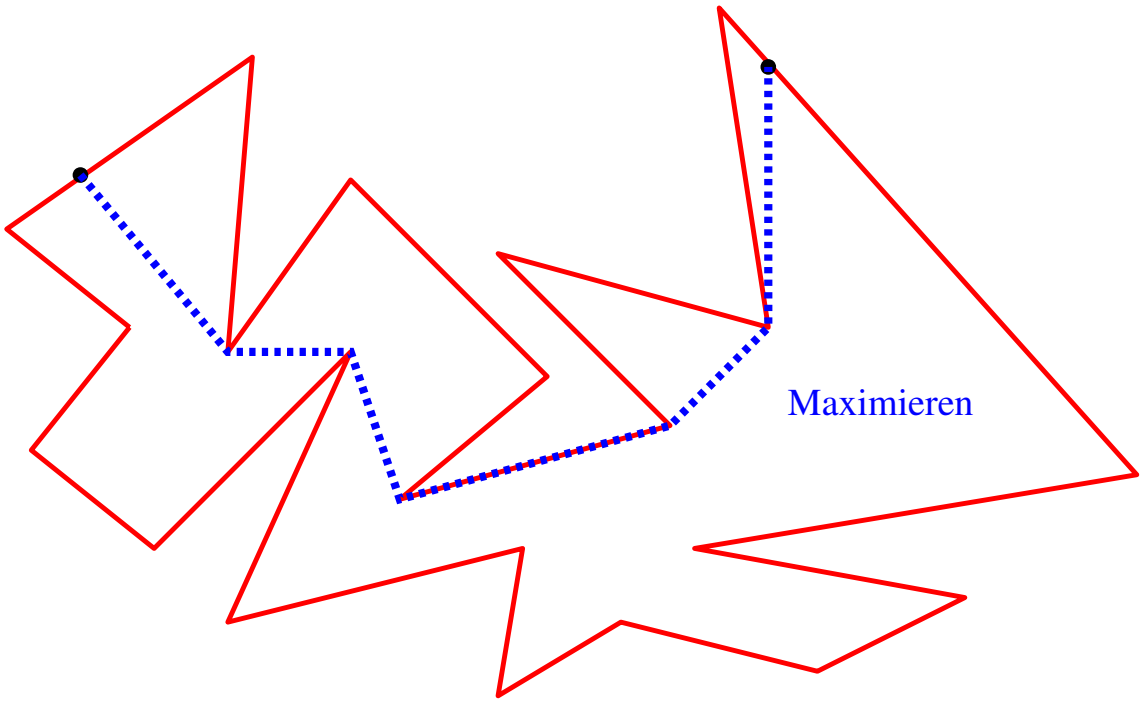
1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon P
- Längster Kürzester Weg zwischen zwei Punkten
- Endpunkte sind Ecken des Polygones



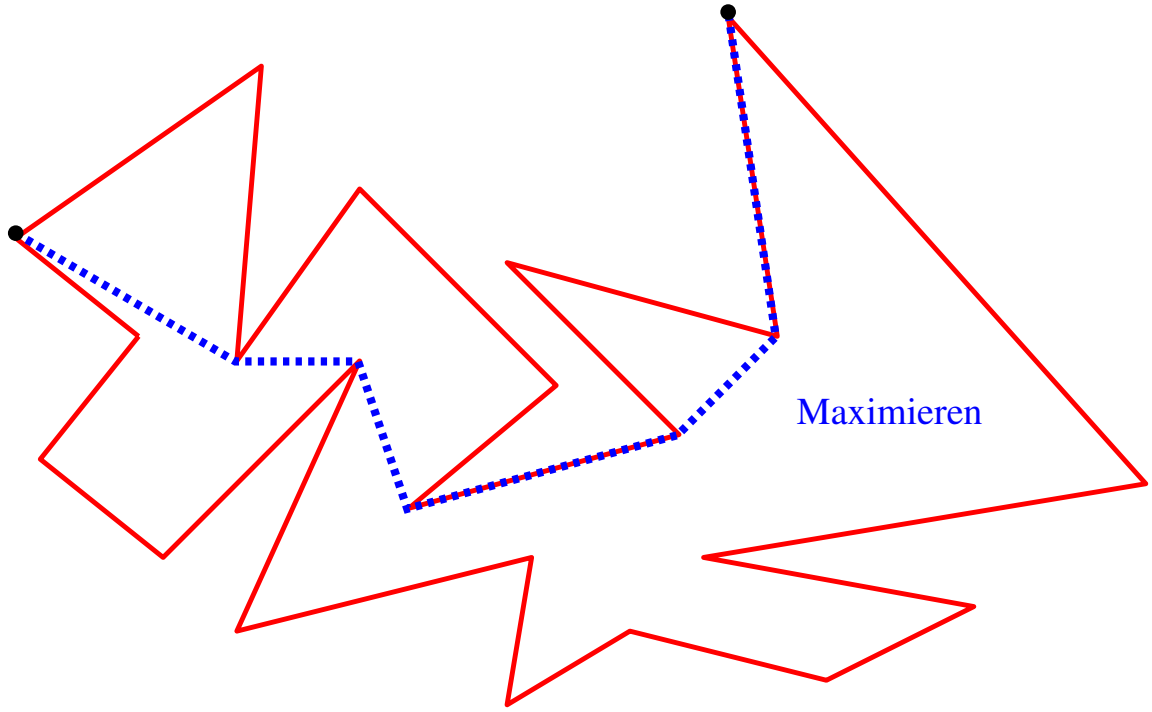
1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon P
- Längster Kürzester Weg zwischen zwei Punkten
- Endpunkte sind Ecken des Polygones



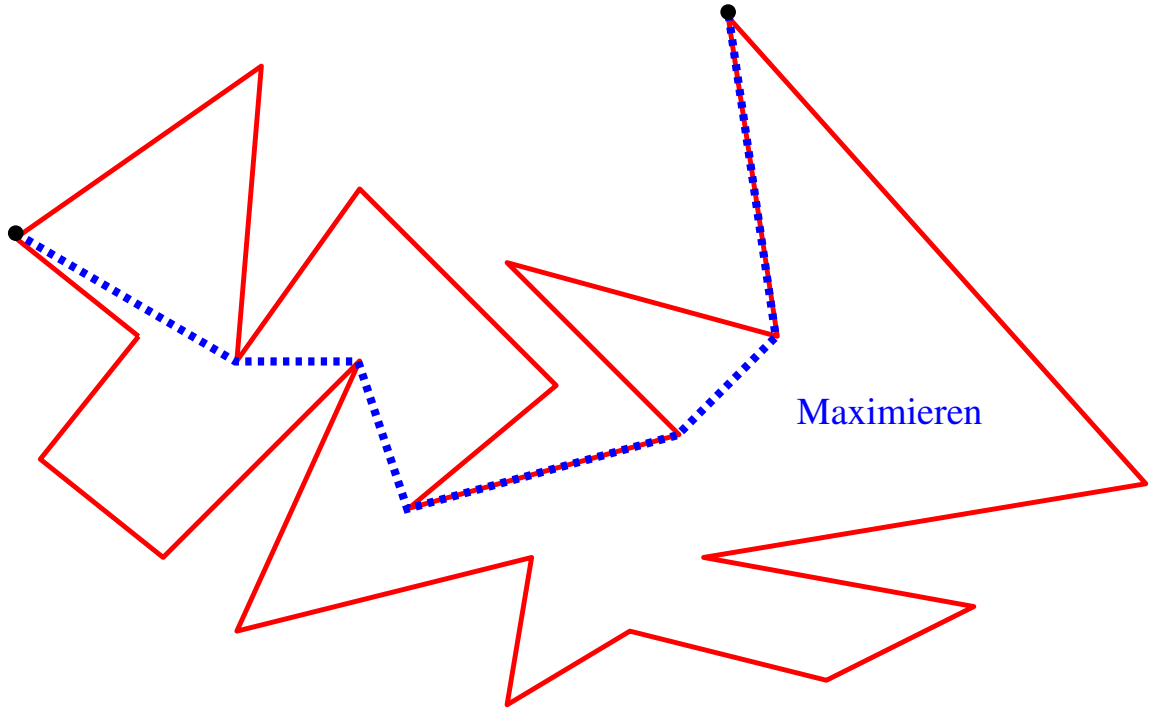
1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon P
- Längster Kürzester Weg zwischen zwei Punkten
- Endpunkte sind Ecken des Polygones



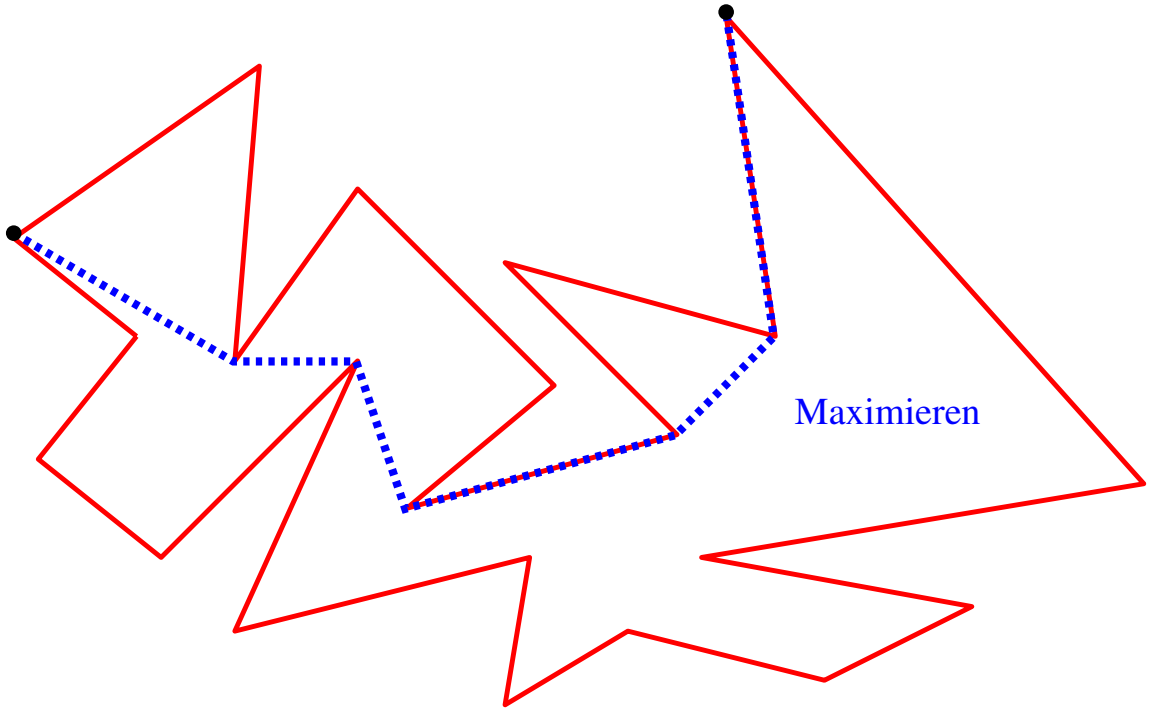
1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon P
- Längster Kürzester Weg zwischen zwei Punkten
- Endpunkte sind Ecken des Polygones n^2 Kandidatenpaare



1.2.3 Durchmesser einfacher Polygone

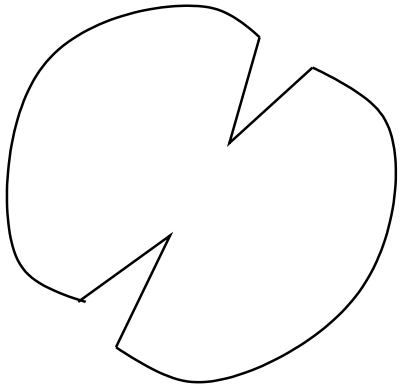
- Einfaches Polygon P
- Längster Kürzester Weg zwischen zwei Punkten
- Endpunkte sind Ecken des Polygones n^2 Kandidatenpaare
- Formal: $\max_{p_i,p_j \text{ Ecken von } P} d(p_i, p_j)$



Idee der Berechnung:

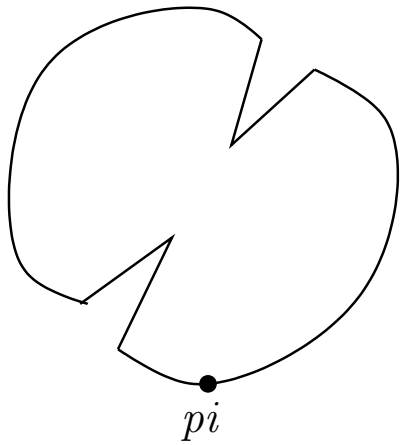
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



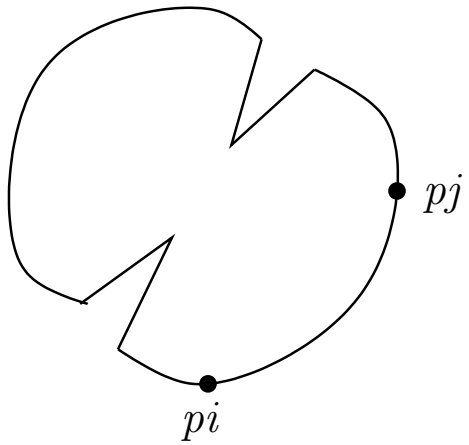
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



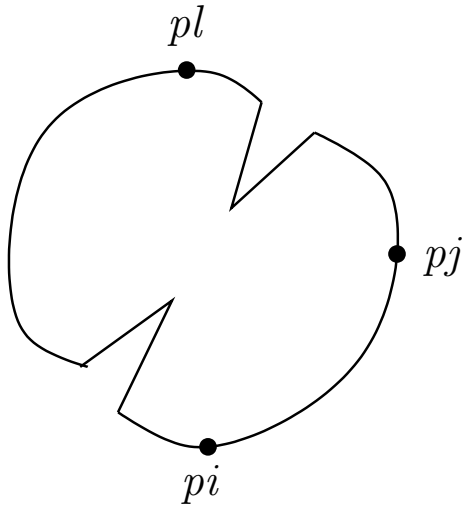
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



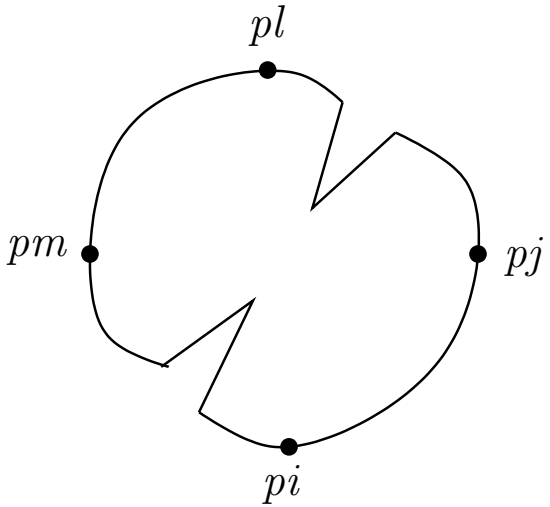
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



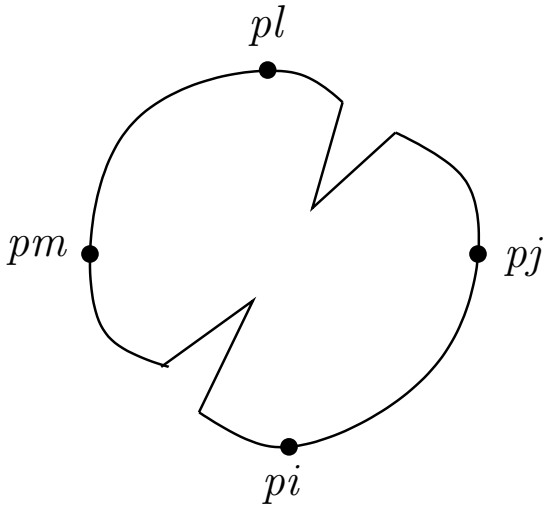
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



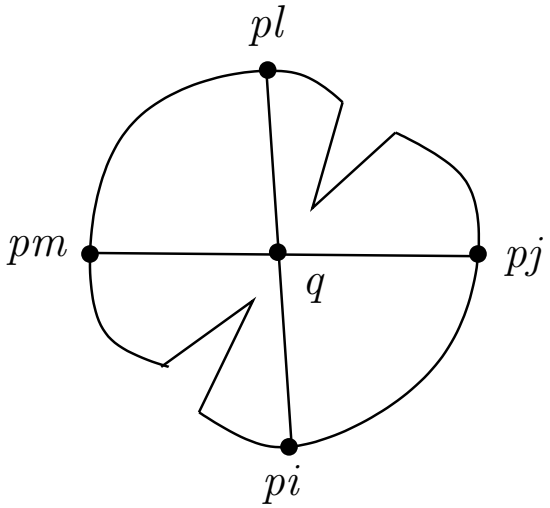
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes
- Monge Eigenschaft: $d(p_i, p_m) + d(p_j, p_l) \leq d(p_i, p_l) + d(p_j, p_m)$



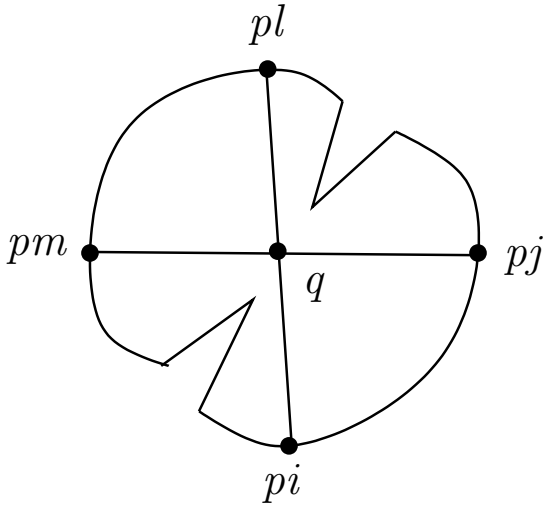
Idee der Berechnung:

- 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



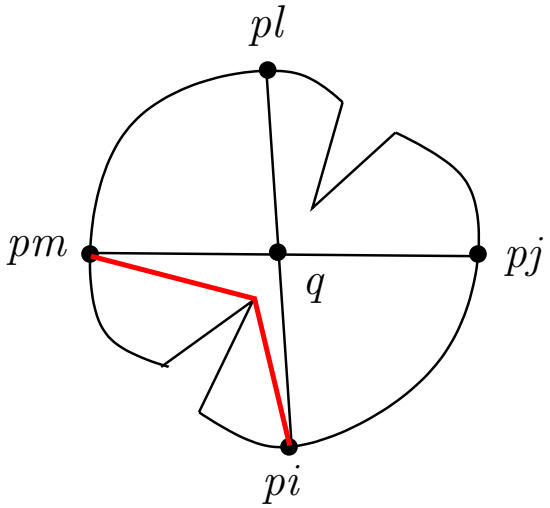
Idee der Berechnung:

- 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!!



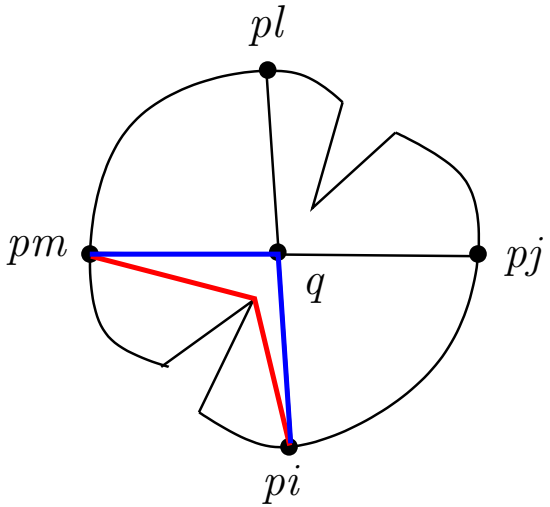
Idee der Berechnung:

- 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!!



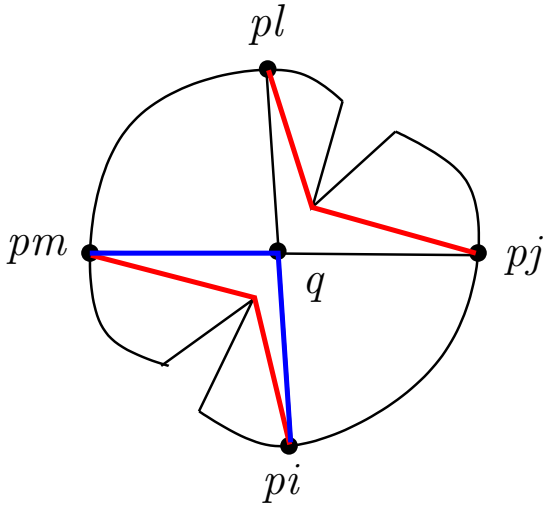
Idee der Berechnung:

- 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!!



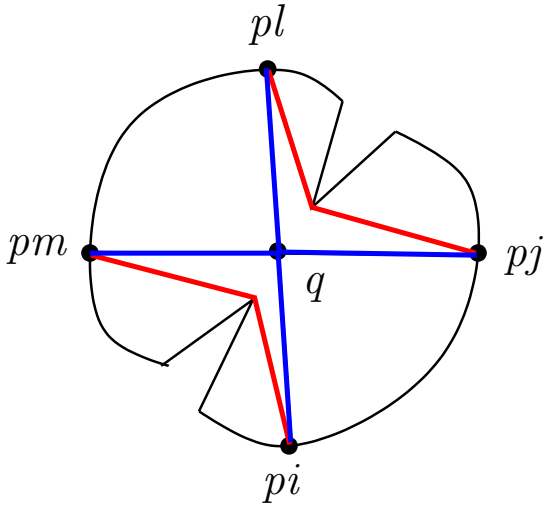
Idee der Berechnung:

- 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!!



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!

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

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