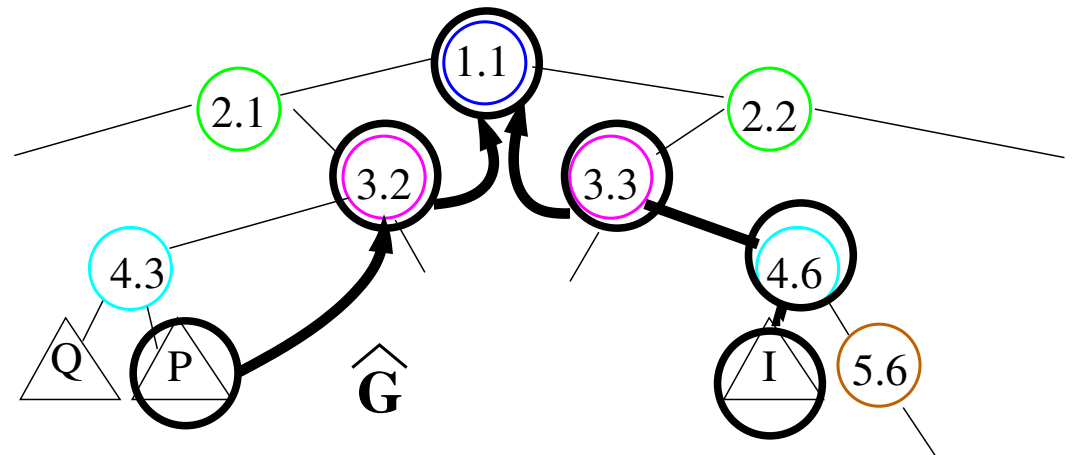
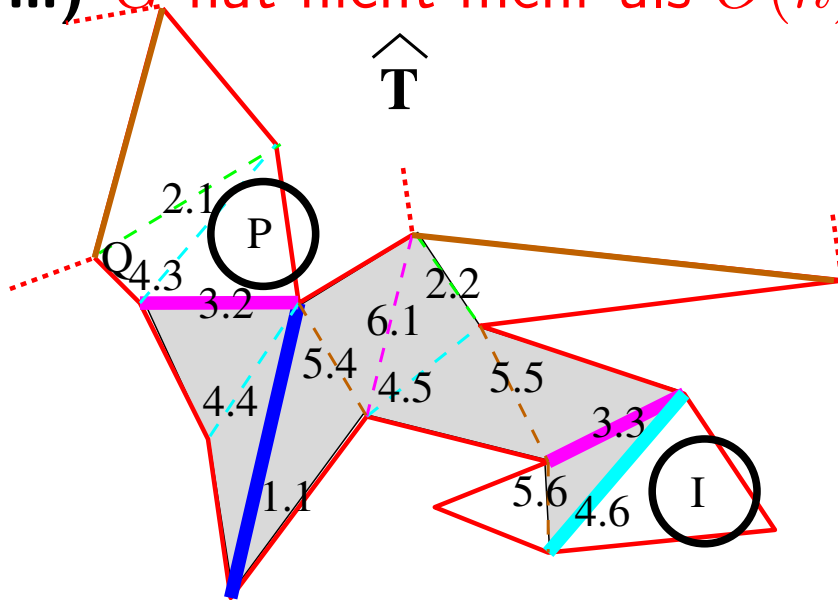


Offline Bewegungsplanung: Preprocessing und Durchmesser

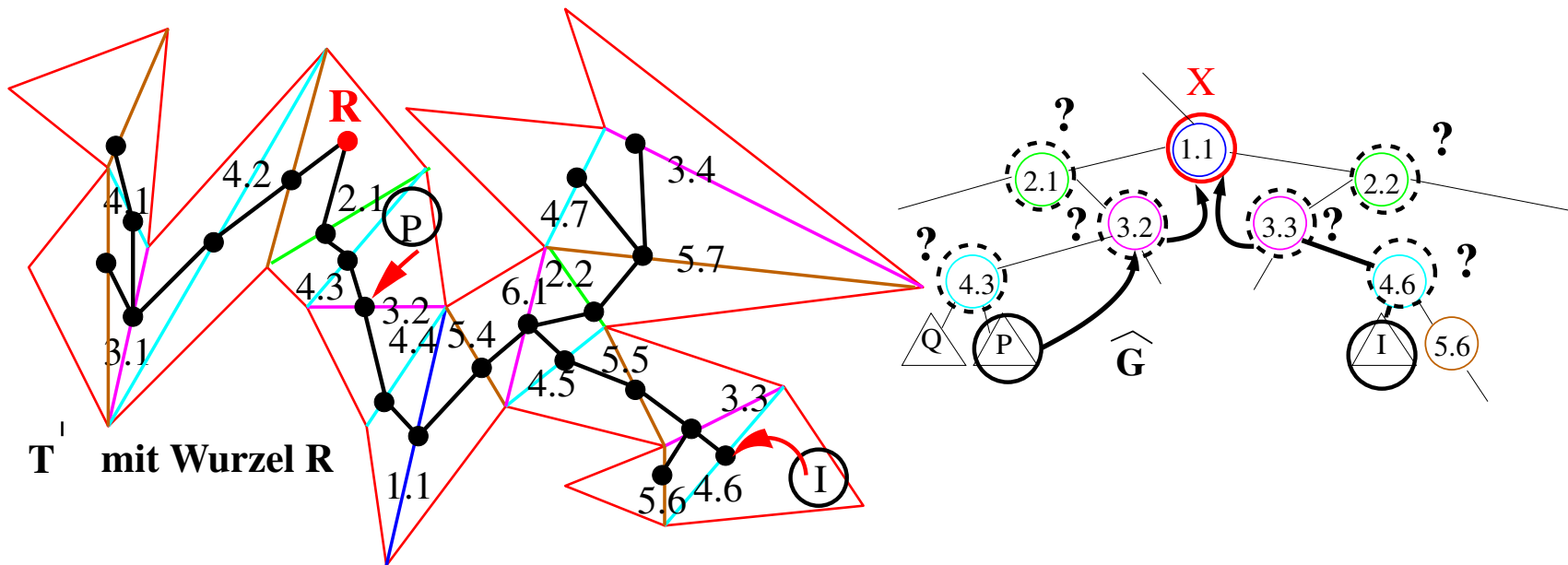
Elmar Langetepe
University of Bonn

Eigenschaften von \hat{G} : Lemma 1.13

- i) Pfad zwischen zwei Dreiecken entlang sukzessiver Diagonalen existiert!
- ii) Wir finden den Weg in $O(\log n)$ Zeit!
- iii) \hat{G} hat nicht mehr als $O(n)$ Kanten!



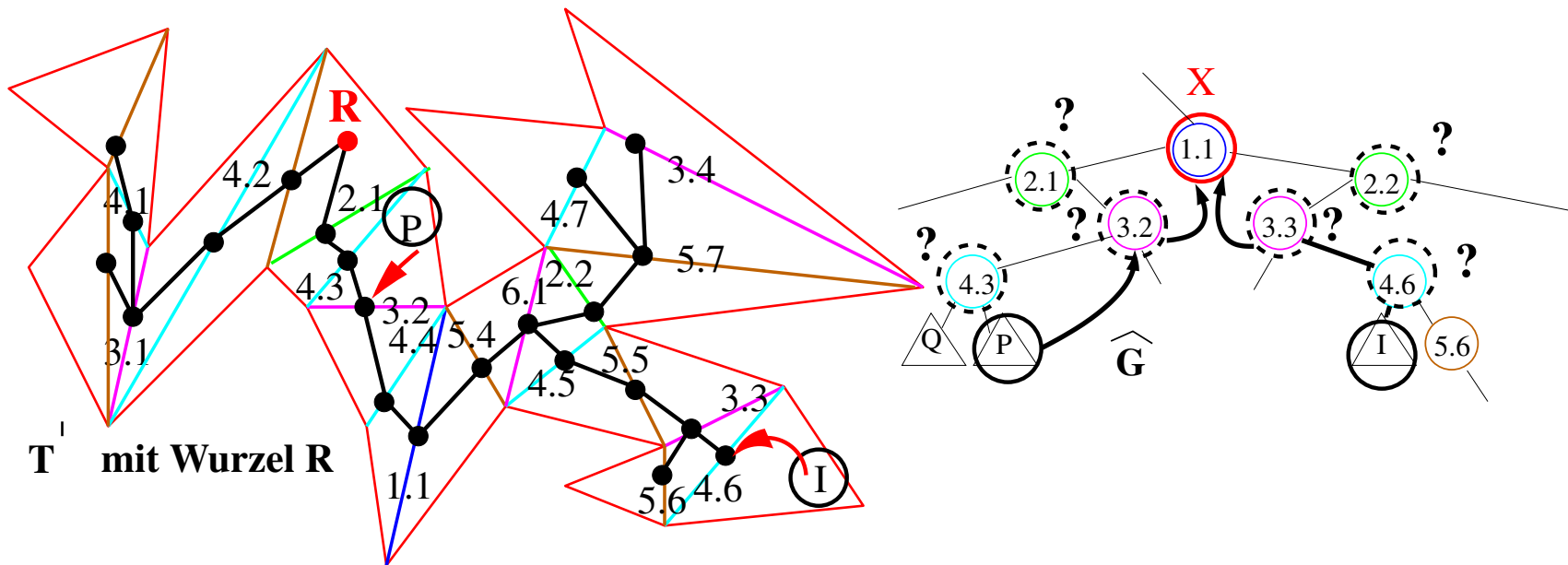
ii) Finden des Weges in \hat{G} !



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

ii) Finden des Weges in \hat{G} !

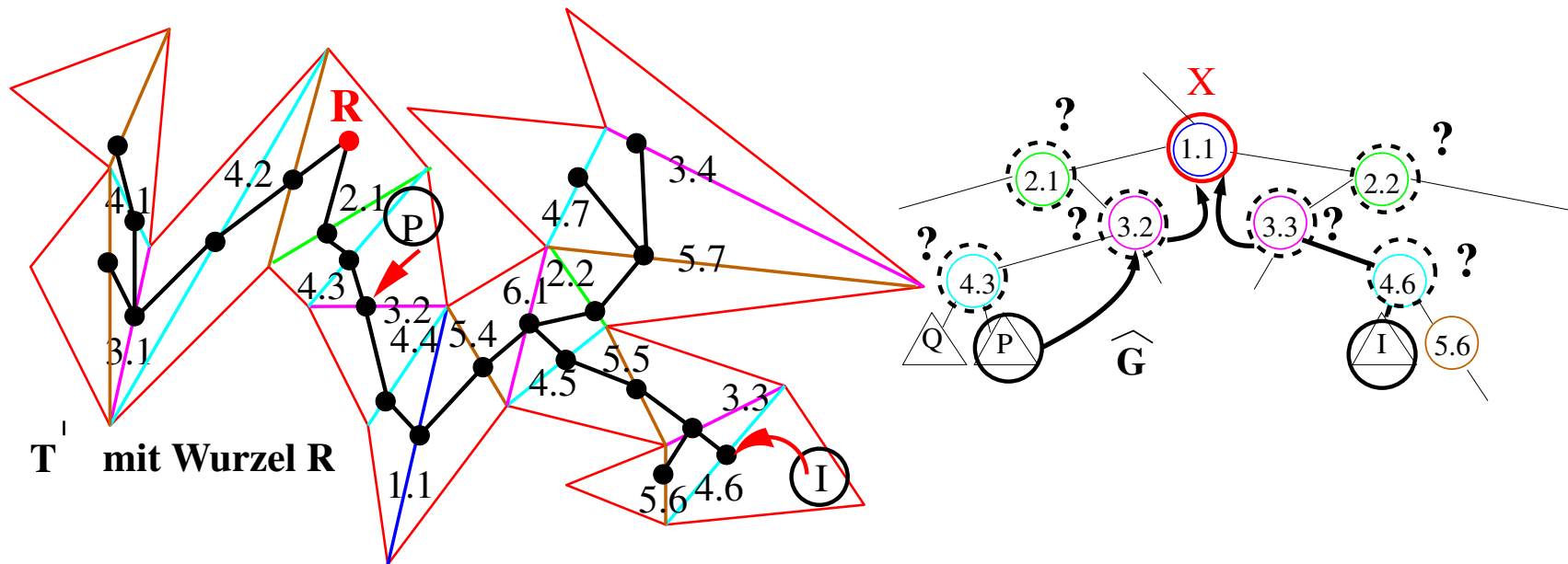
- Gemeinsamer Vorgänger X minimaler Höhe in $O(\log n)$ in \hat{T}



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

ii) Finden des Weges in \hat{G} !

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

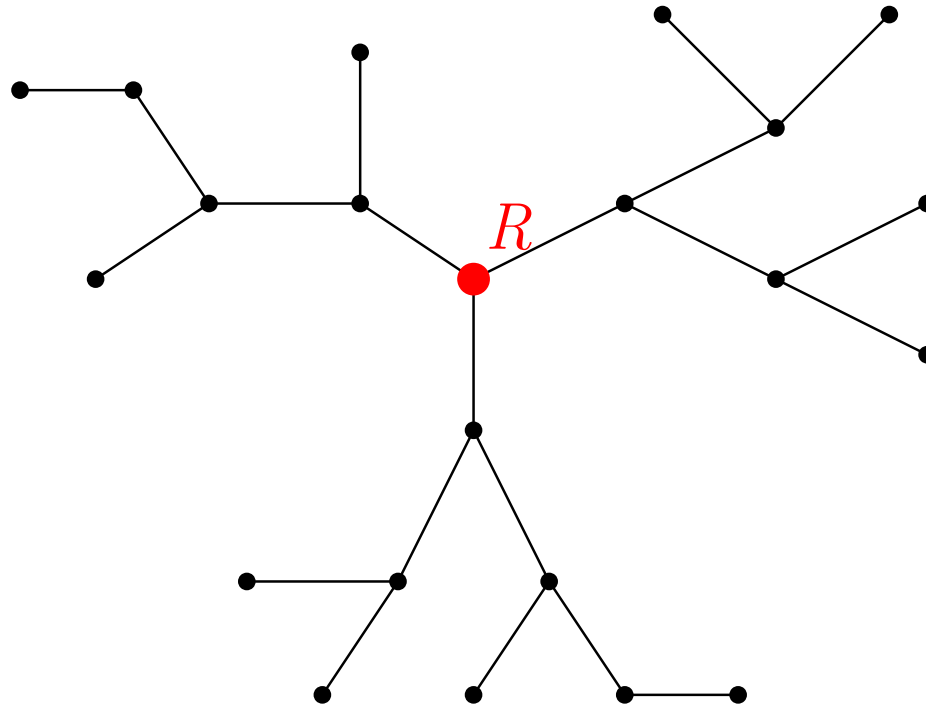


d Vorgänger entweder von $\delta(P)$ ODER $\delta(I)$ in Bezug auf R

Vorgängeranfrage in Baum

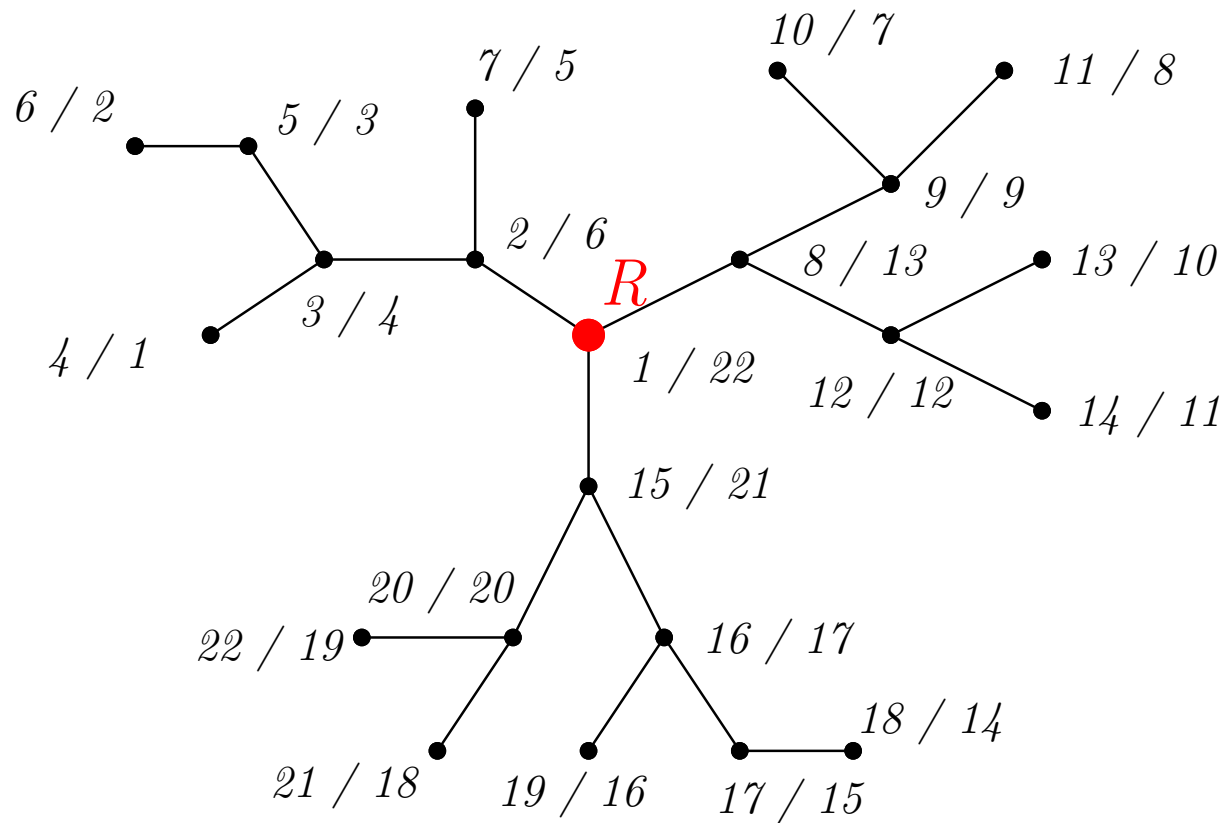
Vorgängeranfrage in Baum

- Preorder/Postorder,



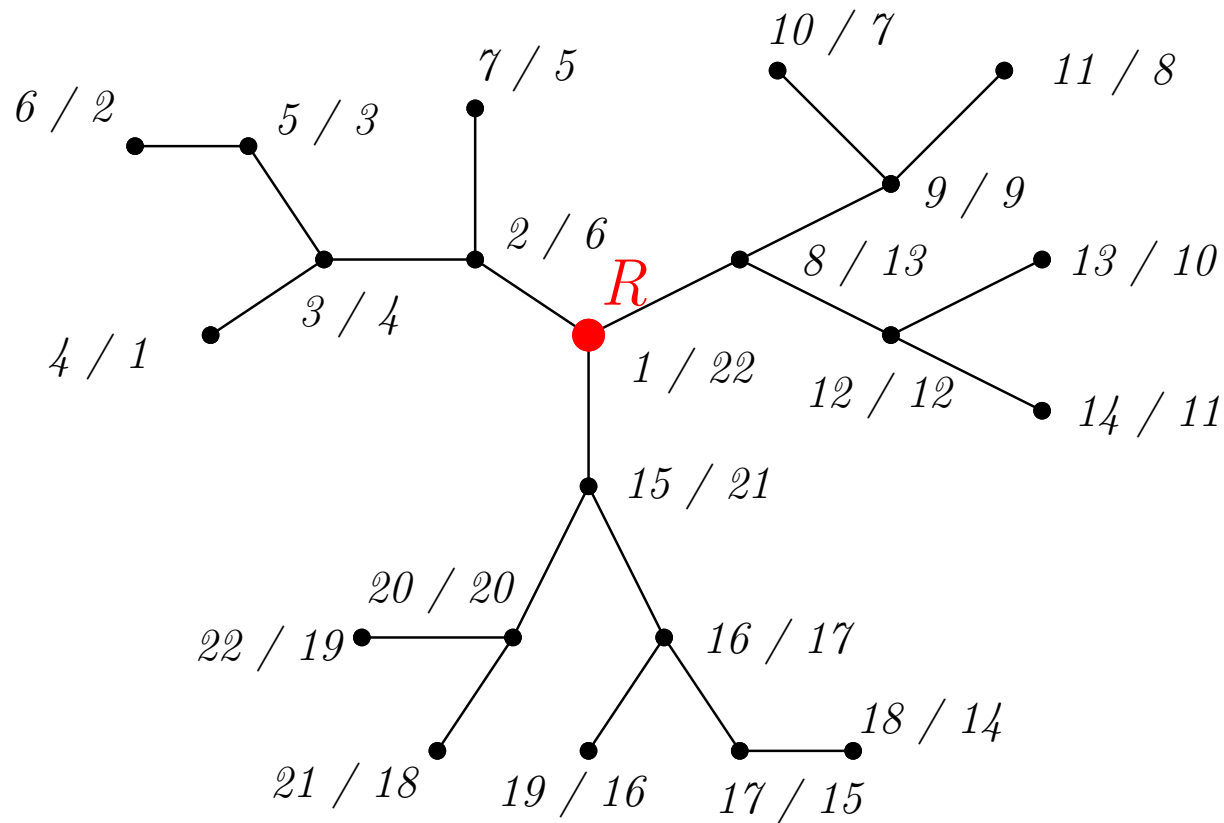
Vorgängeranfrage in Baum

- Preorder/Postorder, DFS und Labelling



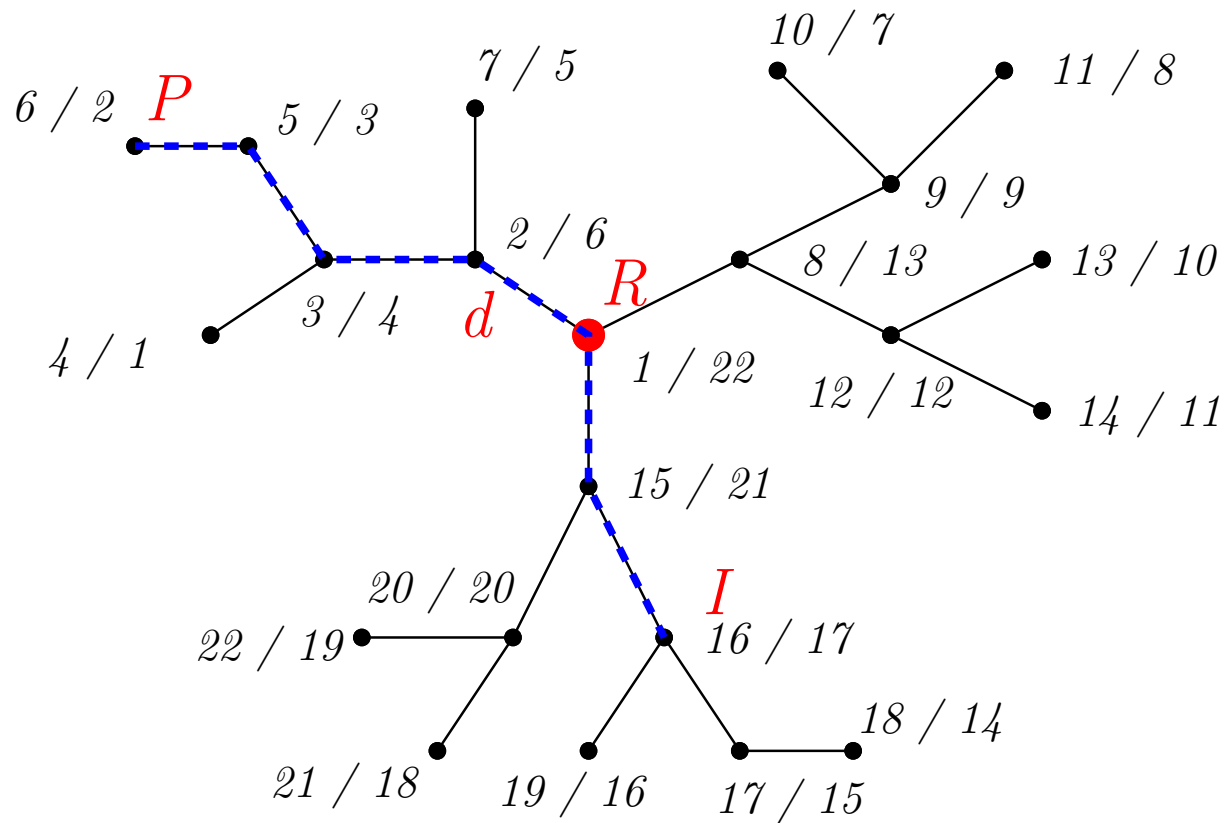
Vorgängeranfrage in Baum

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



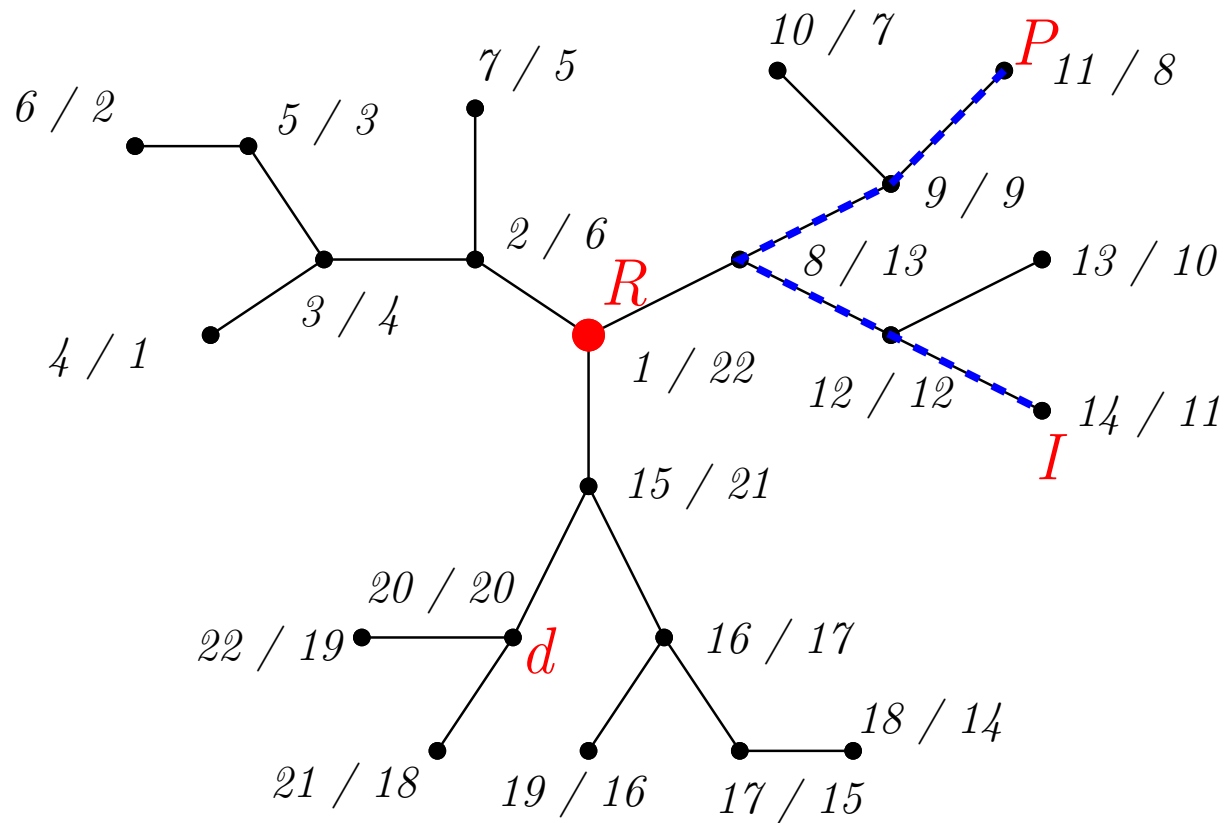
Vorgängeranfrage in Baum

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



Vorgängeranfrage in Baum

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



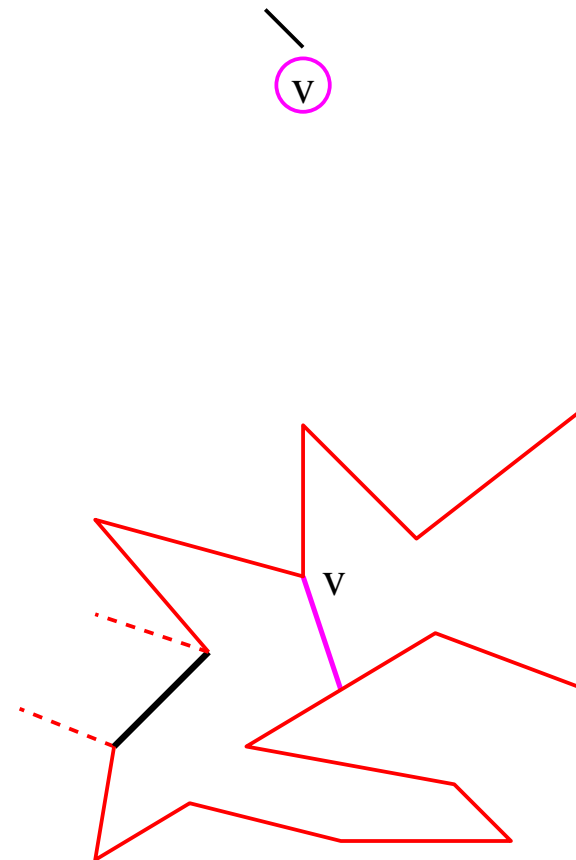
Eigenschaften von \hat{G}

- i) Pfad zwischen zwei Dreiecken existiert!
- i) Länge ist in $O(\log n)$!
- ii) Wir finden den Pfad in $O(\log n)$!

Eigenschaften von \hat{G}

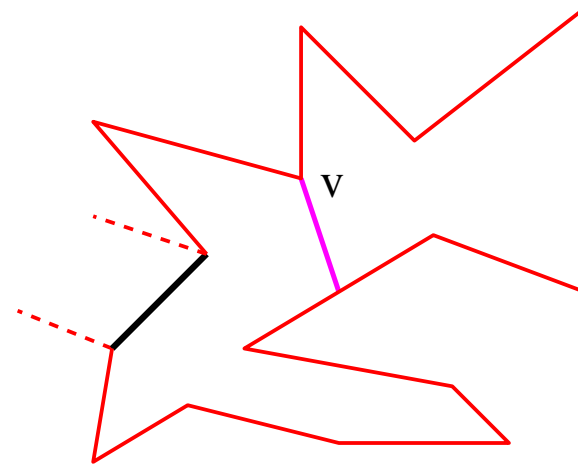
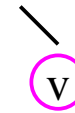
- i) Pfad zwischen zwei Dreiecken existiert!
- i) Länge ist in $O(\log n)$!
- ii) Wir finden den Pfad in $O(\log n)$!
- iii) \hat{G} hat $O(n)$ Kanten !

iii) Komplexität von \hat{G}



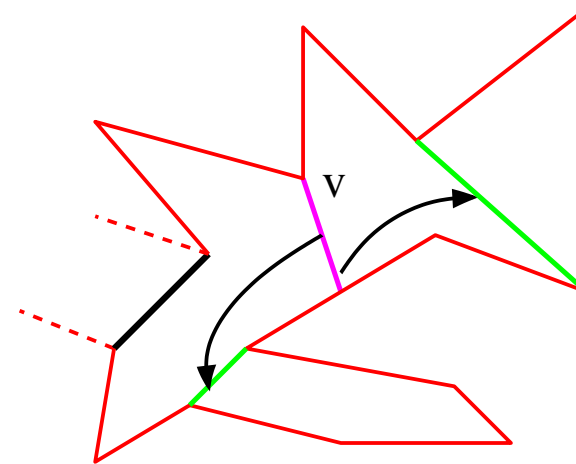
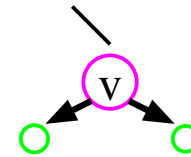
iii) Komplexität von \hat{G}

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



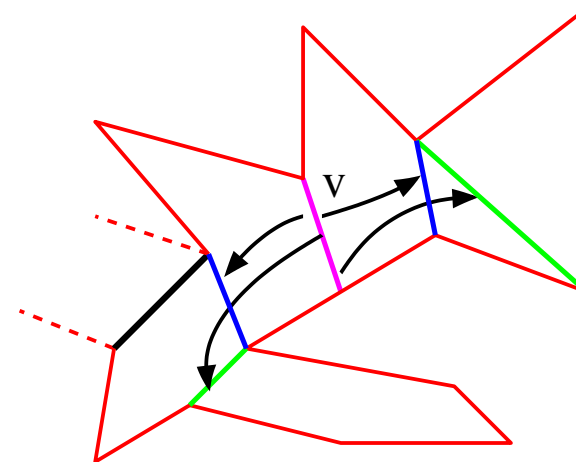
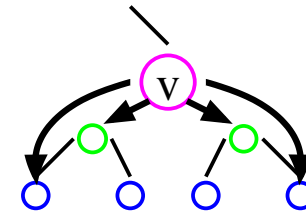
iii) Komplexität von \hat{G}

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



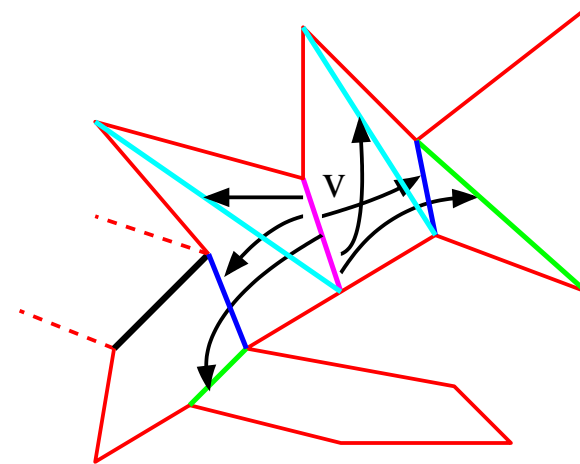
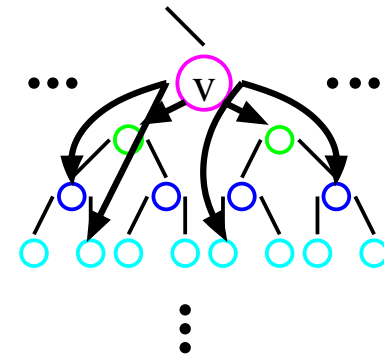
iii) Komplexität von \hat{G}

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



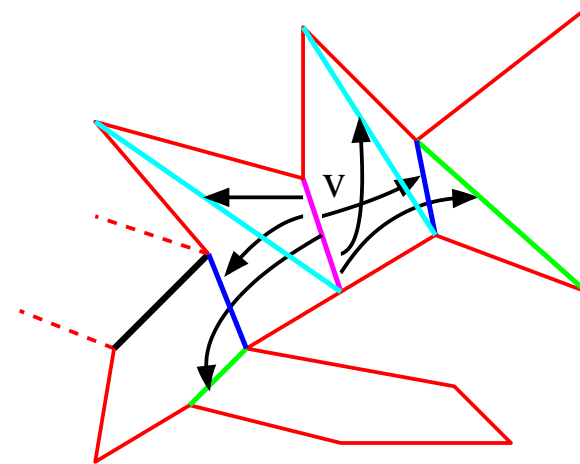
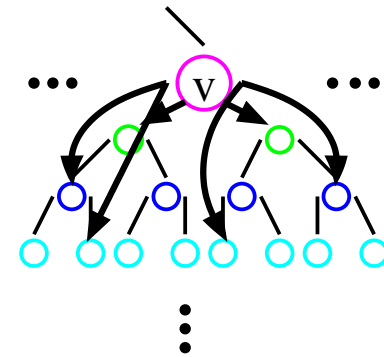
iii) Komplexität von \hat{G}

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



iii) Komplexität von \hat{G}

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

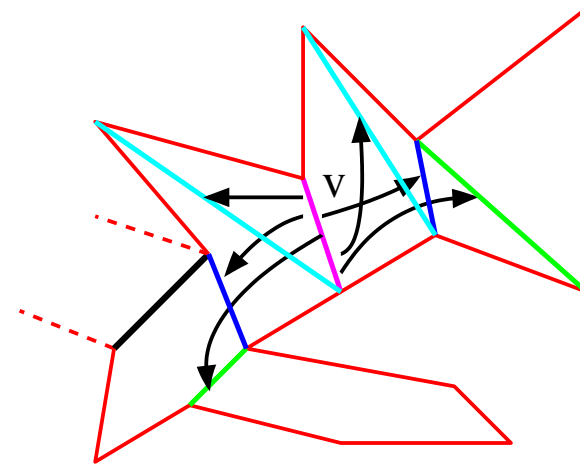
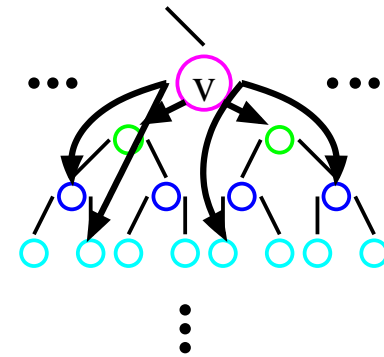


iii) Komplexität von \hat{G}

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

- Balance: Teilbaum bei v
hat $\geq C \cdot \left(\frac{3}{2}\right)^{\text{height}(v)}$ 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 \hat{G}

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

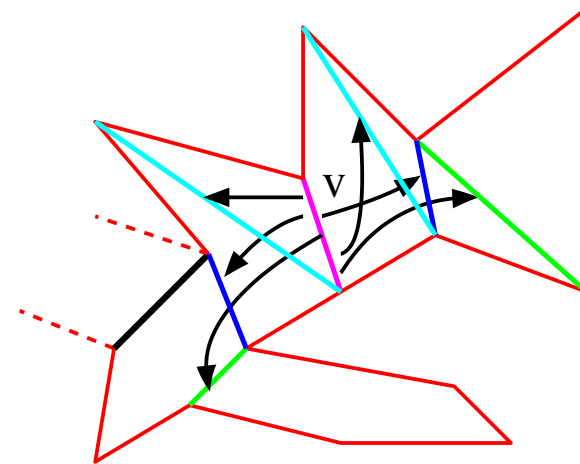
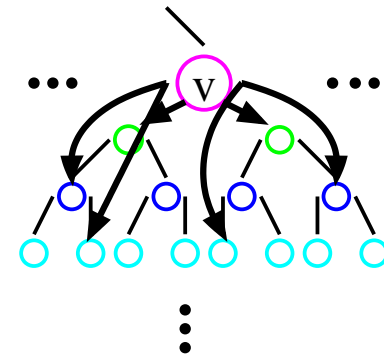
- Balance: Teilbaum bei v
hat $\geq C \cdot \left(\frac{3}{2}\right)^{\text{height}(v)}$ Blätter
(Tafel)

- 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:

$$\sum_{h=1}^{\log_3 n} (2h) \times \left(\left(\frac{2}{3}\right)^h \times \frac{n}{C} \right)$$



iii) Komplexität von \hat{G}

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

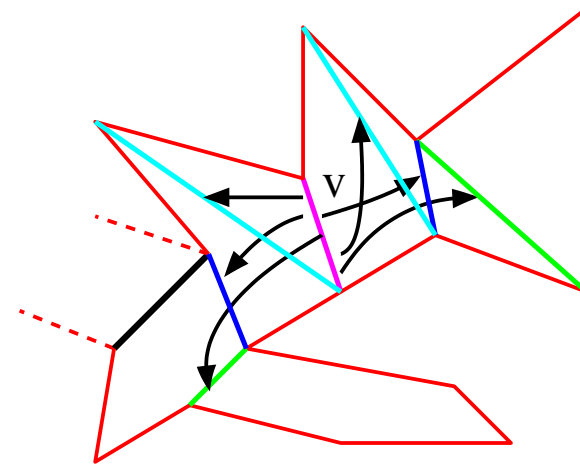
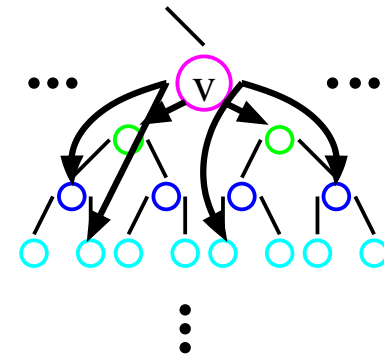
- Balance: Teilbaum bei v
hat $\geq C \cdot \left(\frac{3}{2}\right)^{\text{height}(v)}$ Blätter
(Tafel)

- 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:

$$\sum_{h=1}^{\log_3 n} (2h) \times \left(\left(\frac{2}{3}\right)^h \times \frac{n}{C} \right) \in O(n)$$



iii) Komplexität von \hat{G}

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

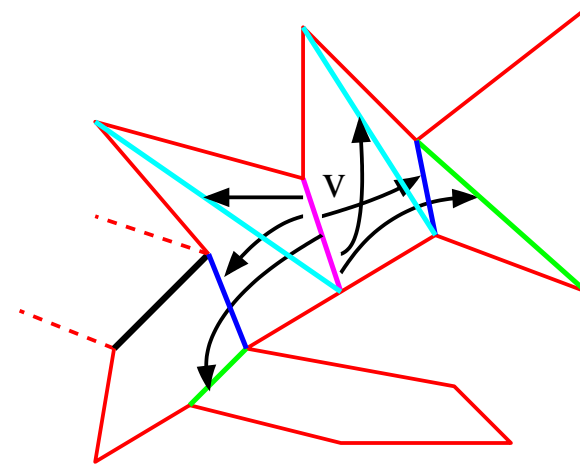
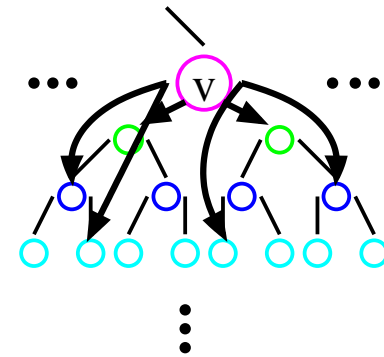
- Balance: Teilbaum bei v
hat $\geq C \cdot \left(\frac{3}{2}\right)^{\text{height}(v)}$ Blätter
(Tafel)

- 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:

$$\sum_{h=1}^{\log_3 n} (2h) \times \left(\left(\frac{2}{3}\right)^h \times \frac{n}{C} \right) \in O(n)$$



Konstruktion \hat{G}

Konstruktion \hat{G}

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

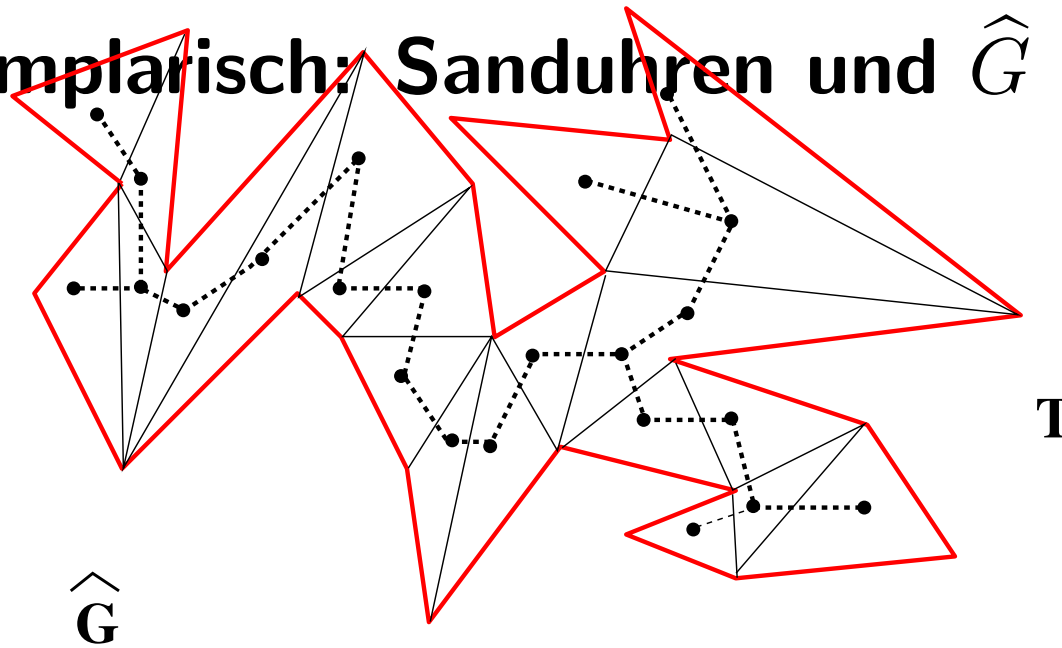
Exemplarisch: Sanduhren und \hat{G} aufbauen!

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 \hat{G} aufbauen!

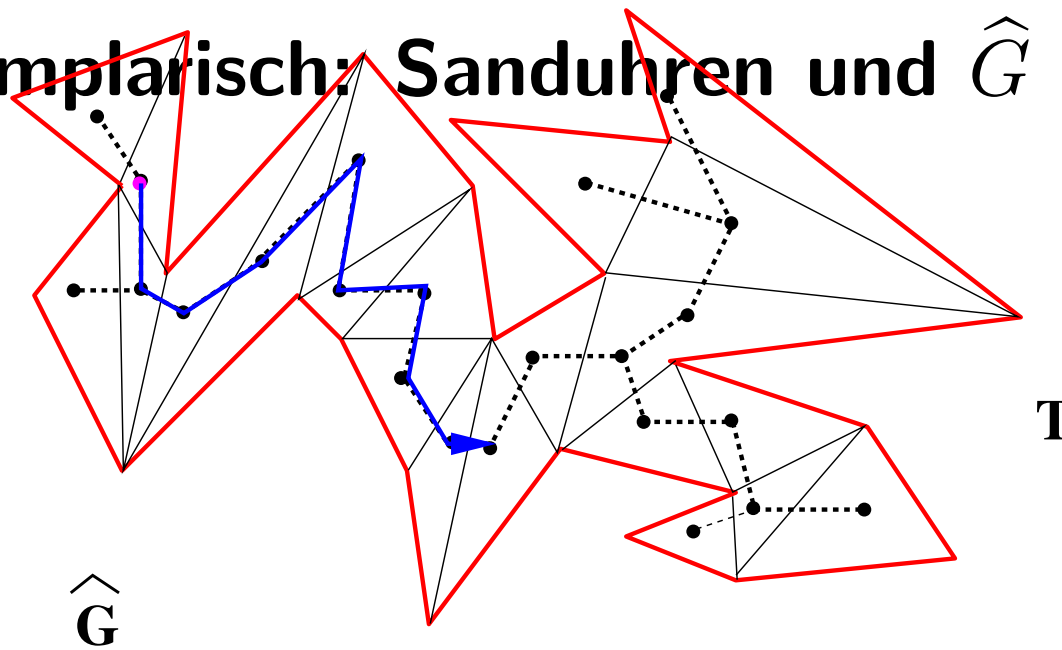


Durchlauf gemäß Cutting Theorem Durchlauf

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 \hat{G} aufbauen!

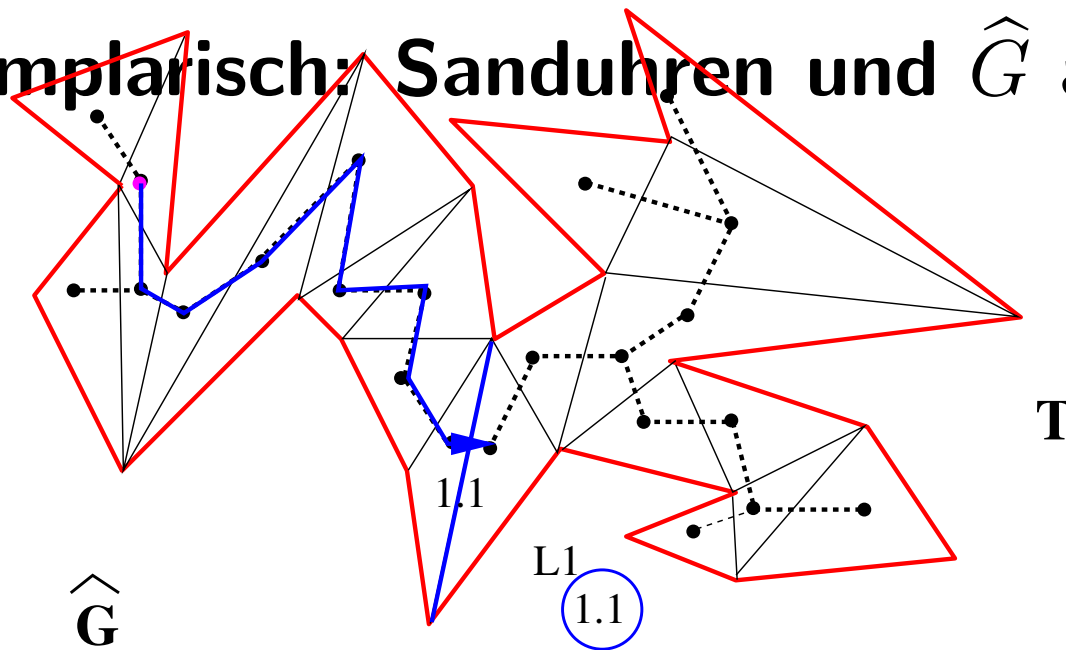


Durchlauf gemäß Cutting Theorem Durchlauf

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 \hat{G} aufbauen!

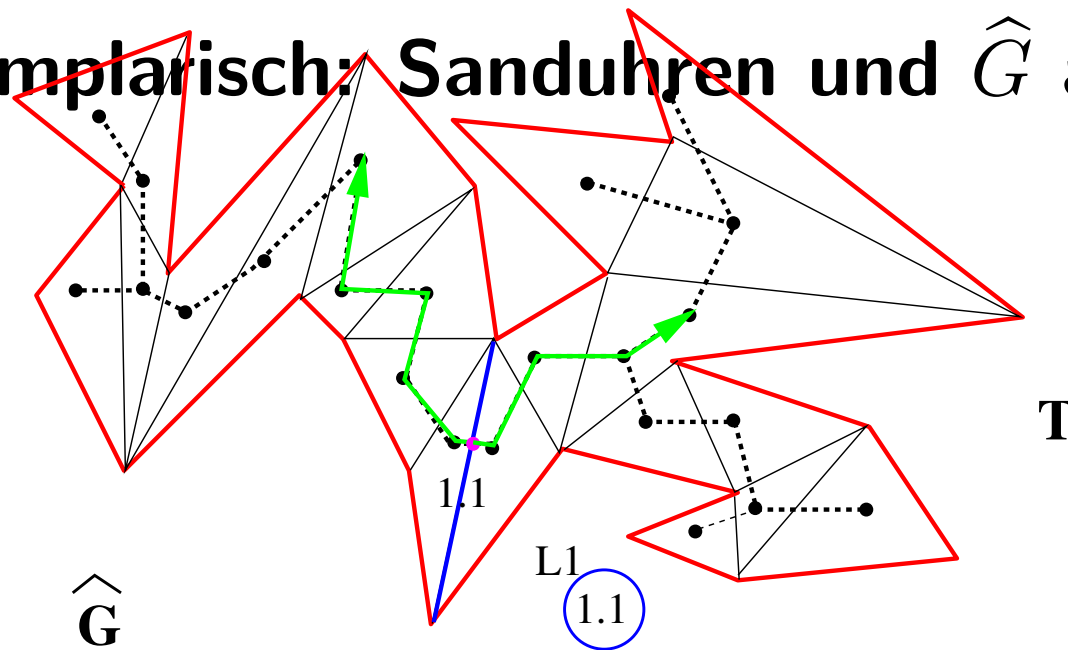


Durchlauf gemäß Cutting Theorem Durchlauf

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 \hat{G} aufbauen!

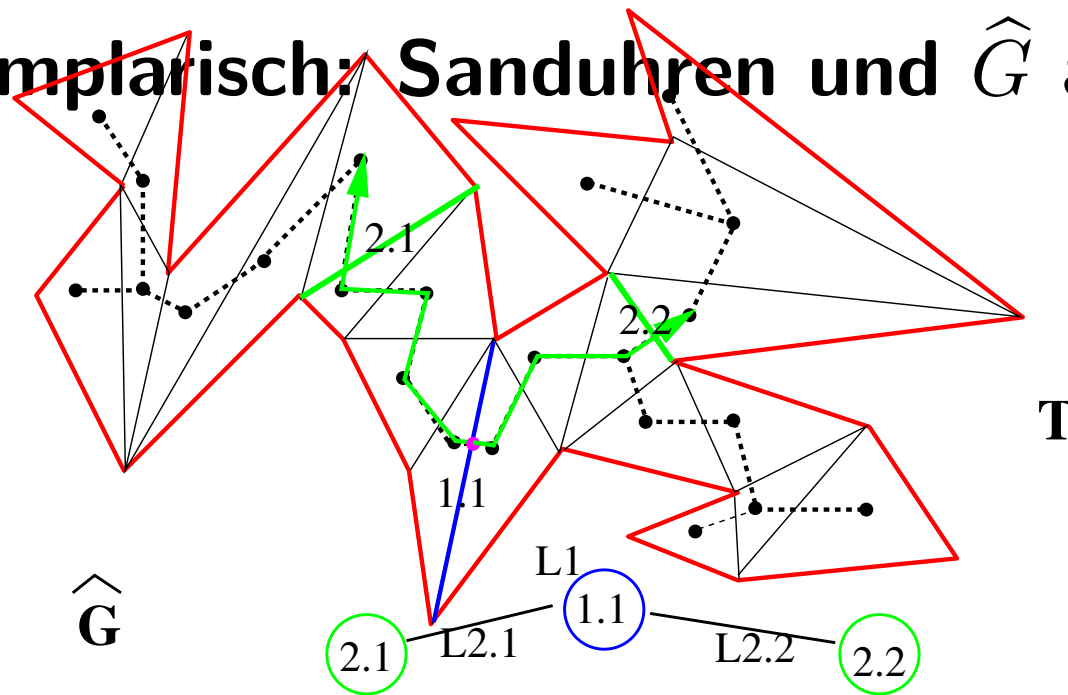


Durchlauf gemäß Cutting Theorem Durchlauf

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 \hat{G} aufbauen!

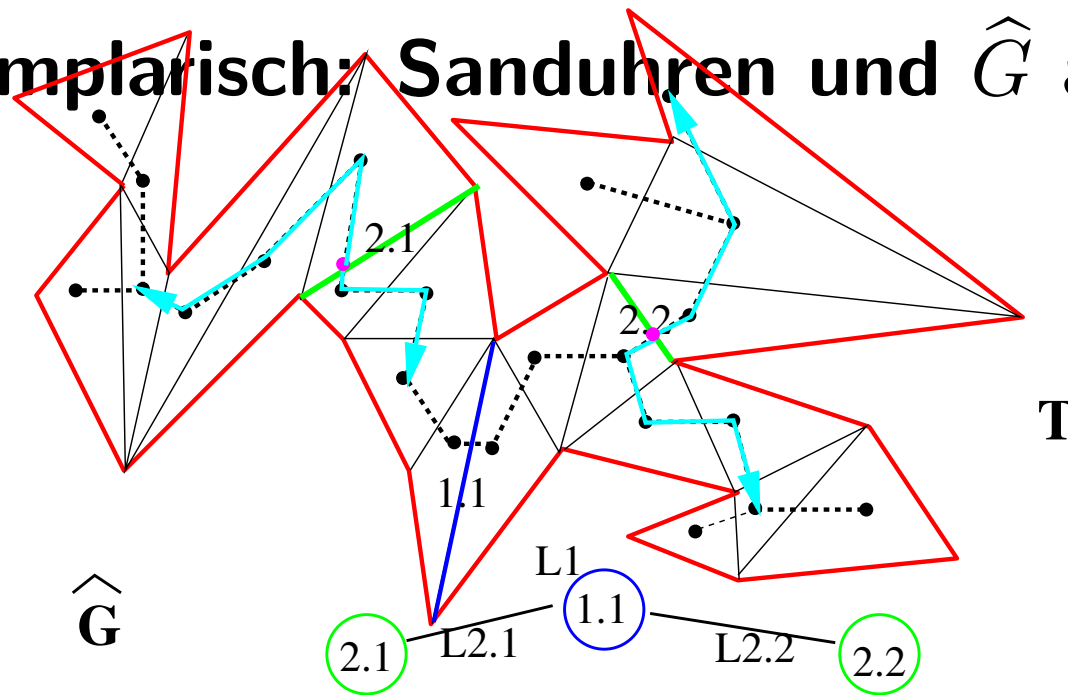


Durchlauf gemäß Cutting Theorem Durchlauf

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 \hat{G} aufbauen!

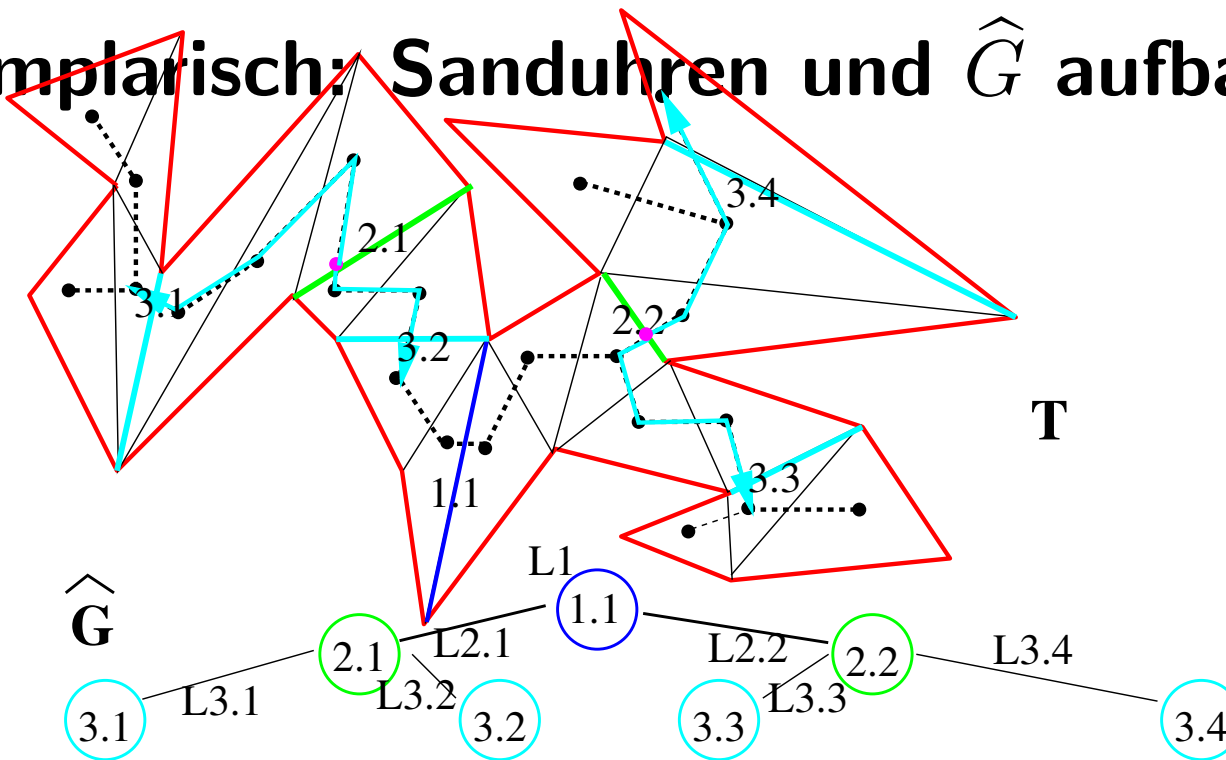


Durchlauf gemäß Cutting Theorem Durchlauf

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 \hat{G} aufbauen!

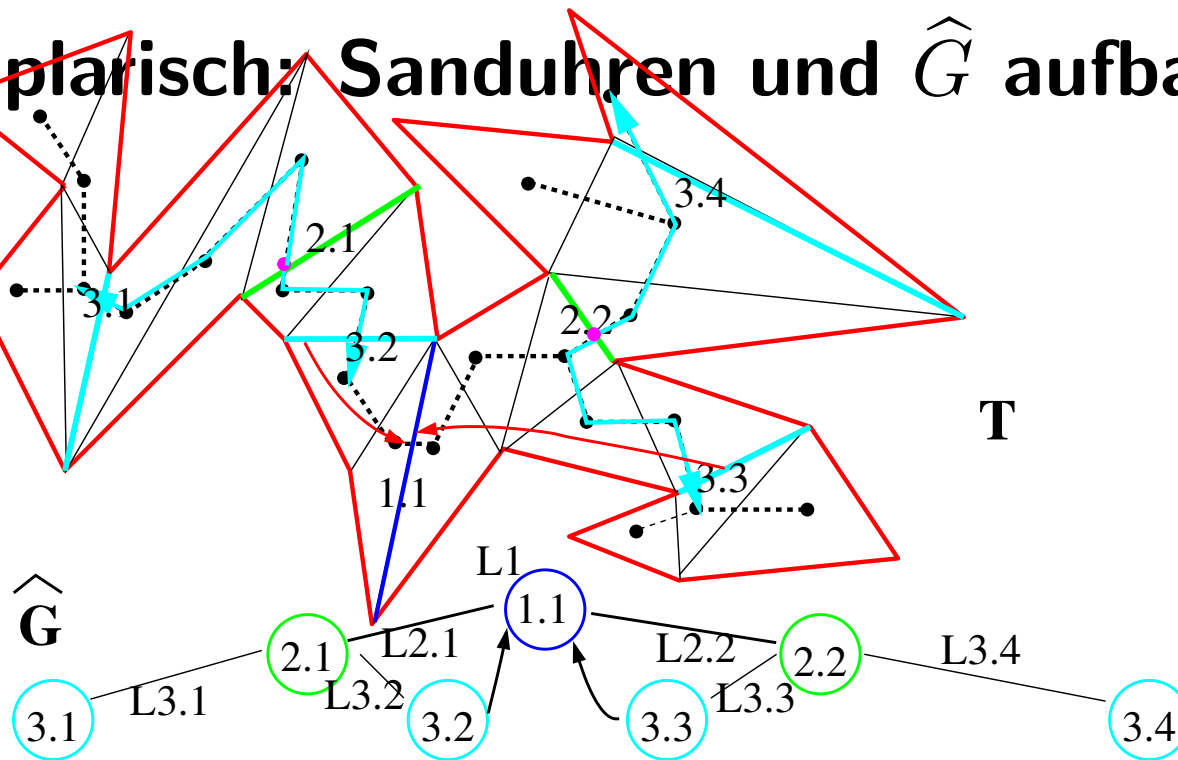


Durchlauf gemäß Cutting Theorem Durchlauf

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 \hat{G} aufbauen!

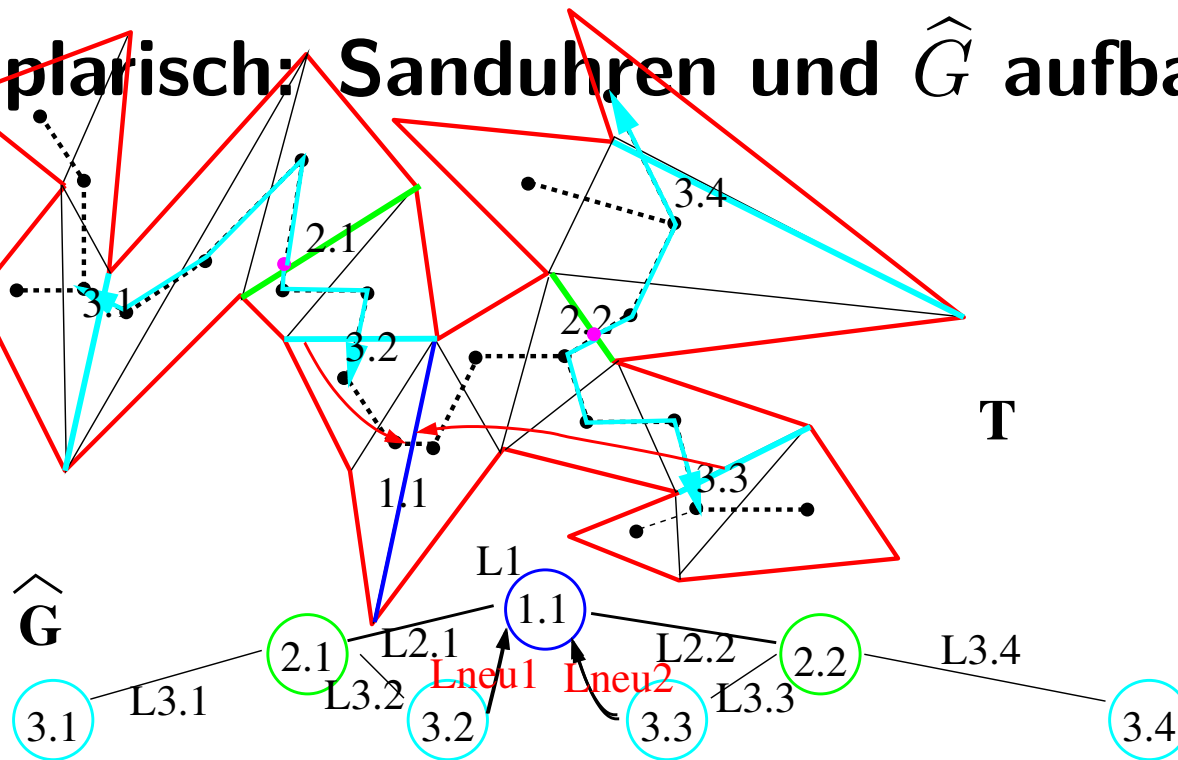


Durchlauf gemäß Cutting Theorem Durchlauf

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 \hat{G} aufbauen!

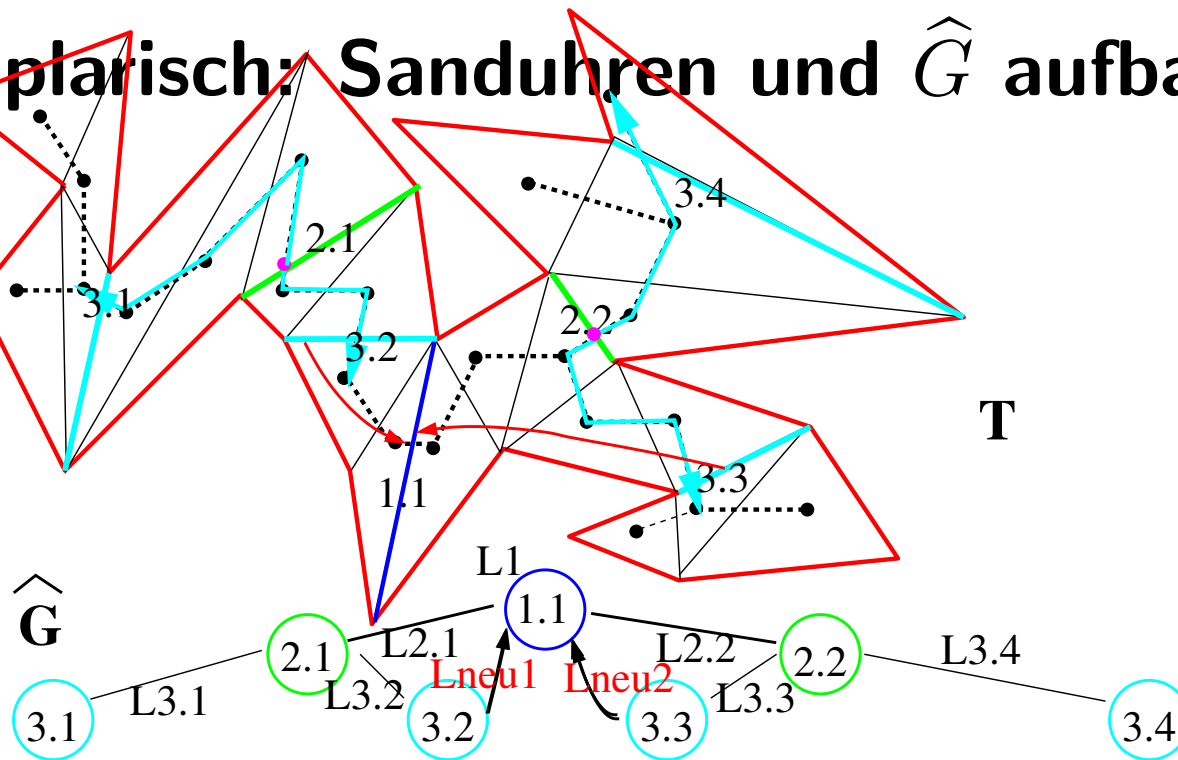


Durchlauf gemäß Cutting Theorem Durchlauf

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 \hat{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

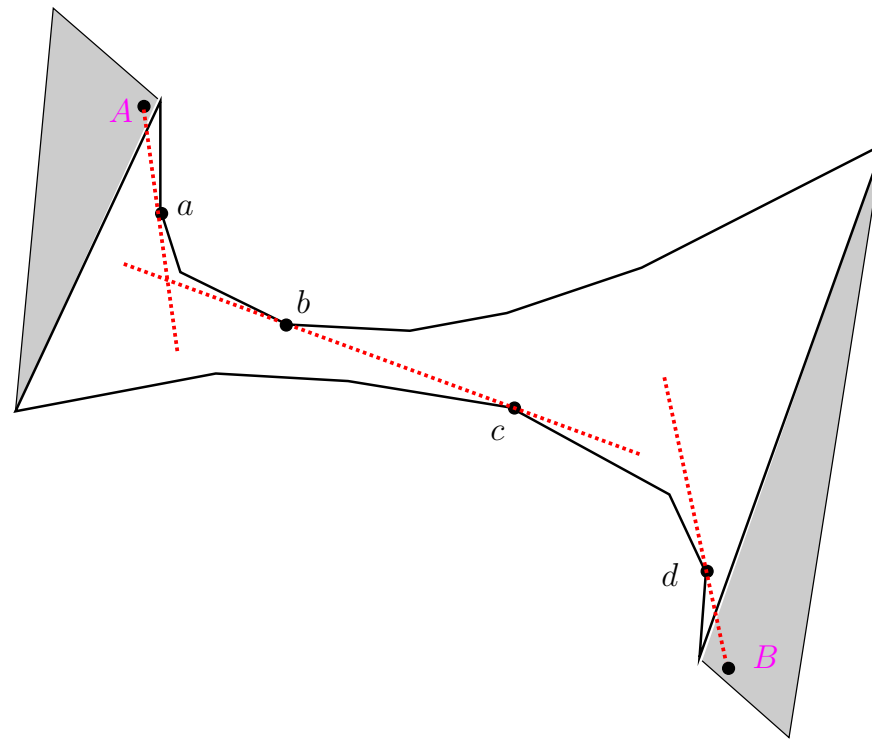
1. Berechne Triangulation T und Dual T^* : $O(n)$
2. Berechne hierarch. bal. Baum \hat{T} , Sch.-Graph \hat{G} : $O(n)$
3. Komplexität \hat{G} : $O(n)$
4. Berechne *alle* Sanduhren von \hat{G} : $O(n)$
5. Navigation zw. Dreiecken in \hat{G} : Sequenz v. Diagonalen: $O(\log n)$
6. Konkat. Sanduhren für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

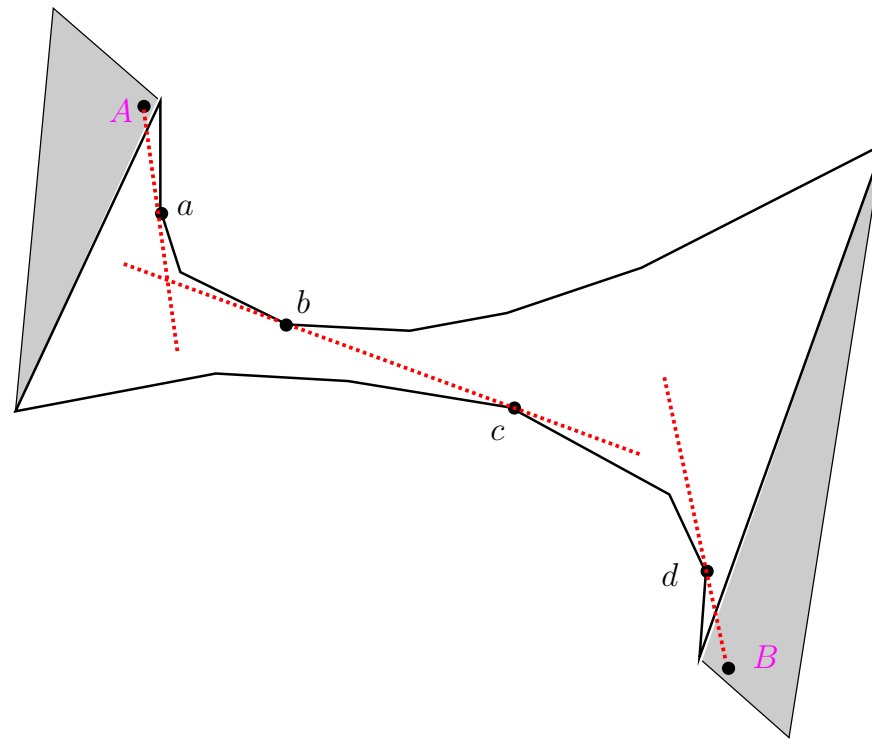
Berechne Shortest Path aus finaler Sanduhr

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



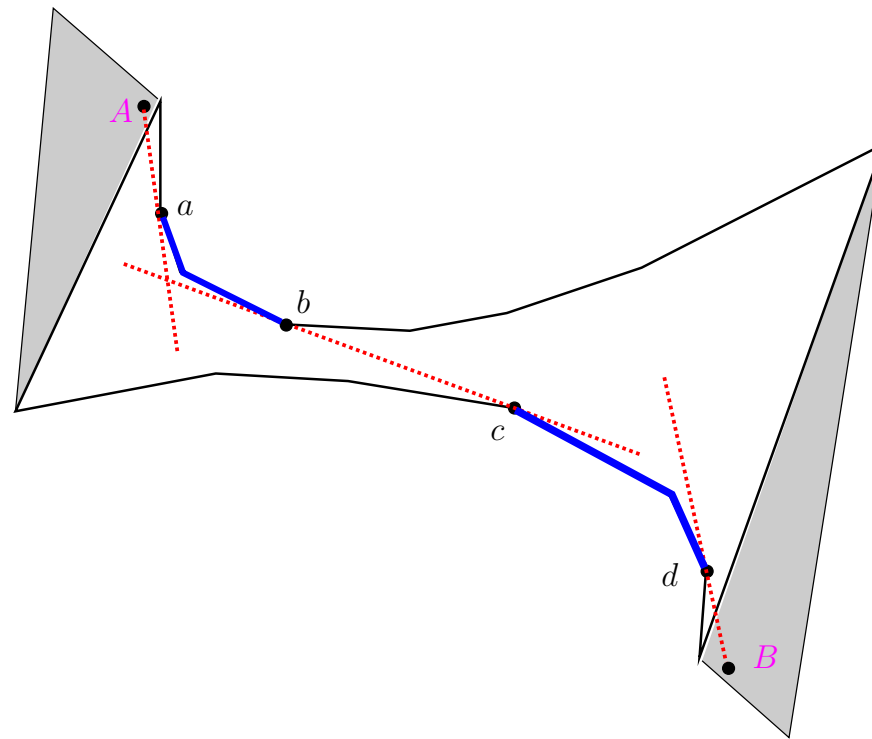
Berechne Shortest Path aus finaler Sanduhr

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

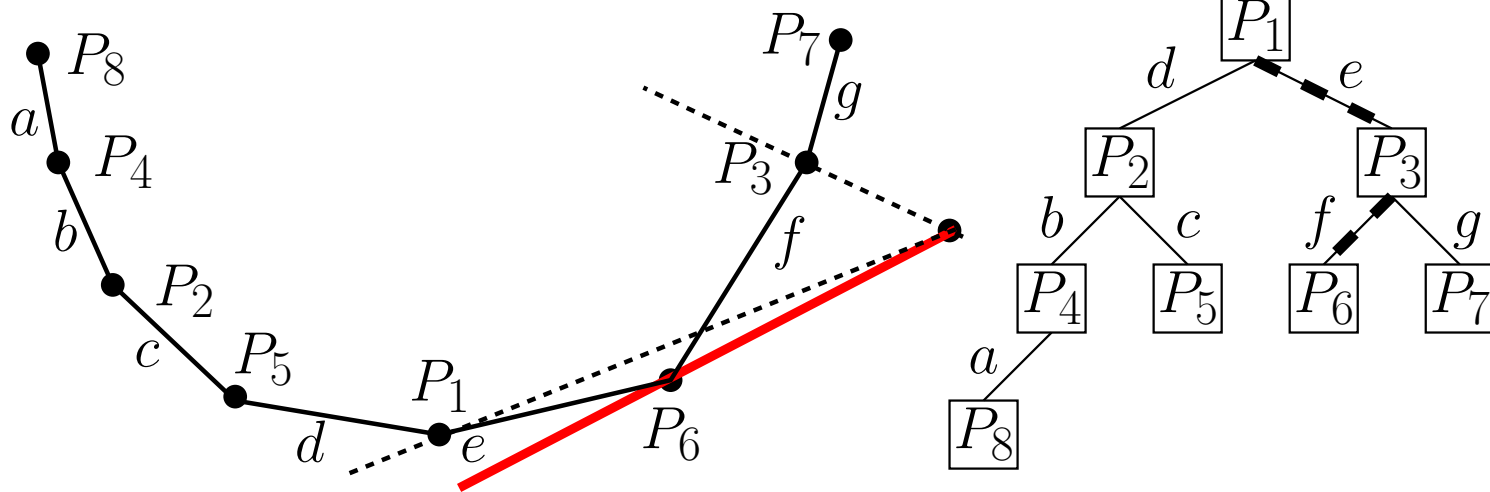


Berechne Shortest Path aus finaler Sanduhr

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

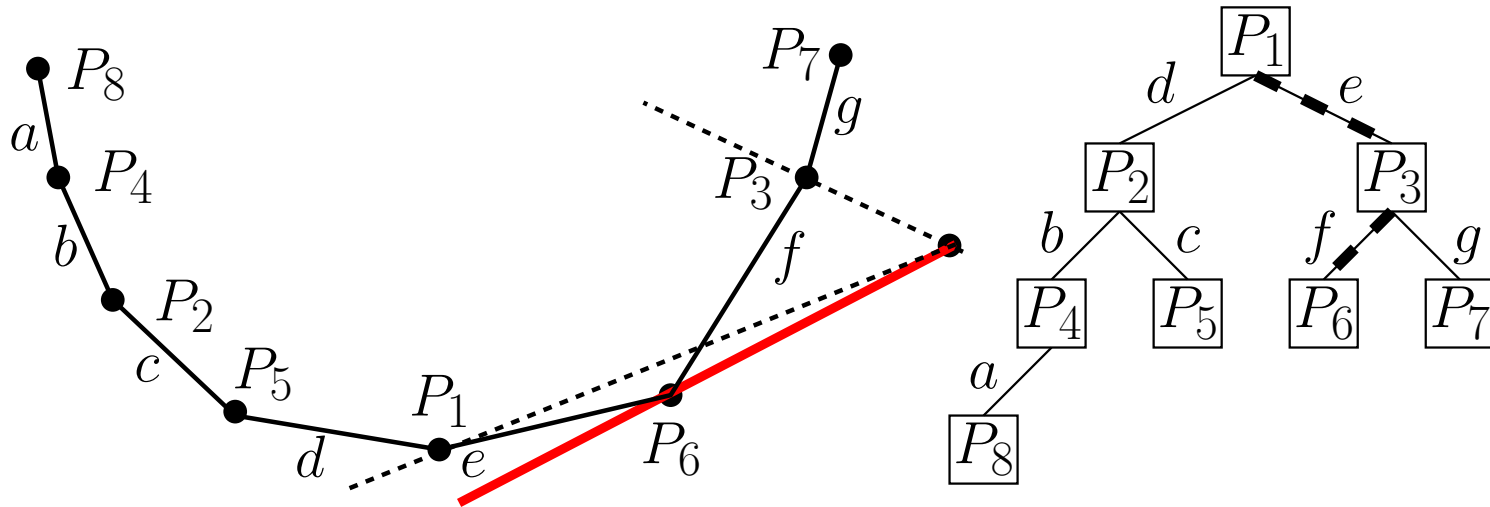


Datenstruktur Hourglass



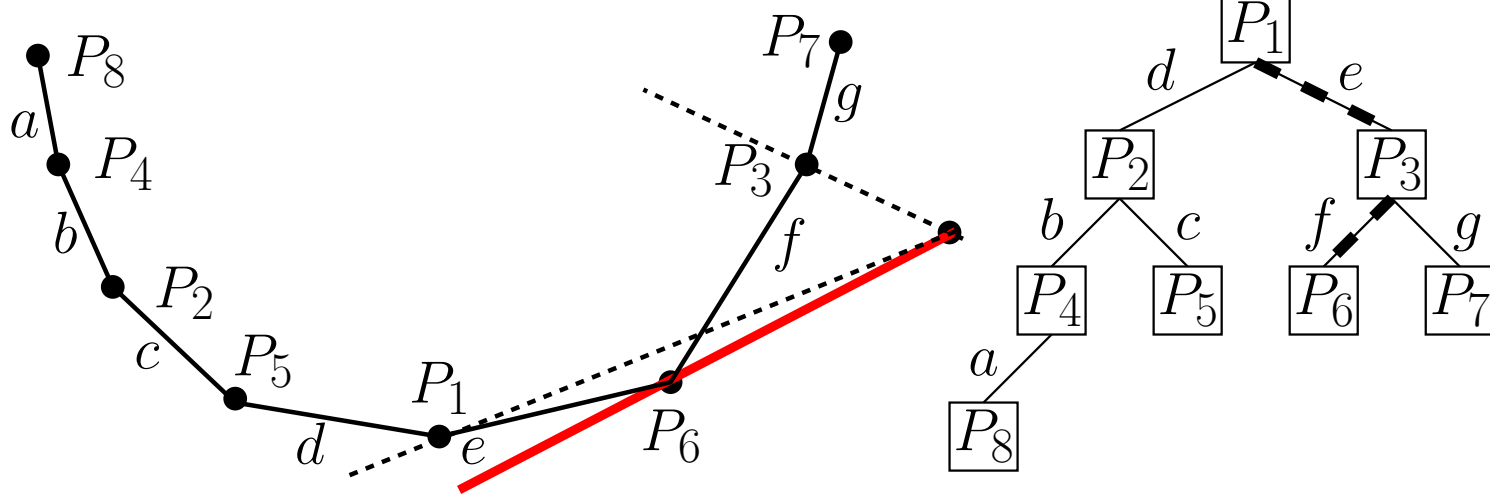
Datenstruktur Hourglass

- Ketten der Sanduhren in bal. Baum speichern



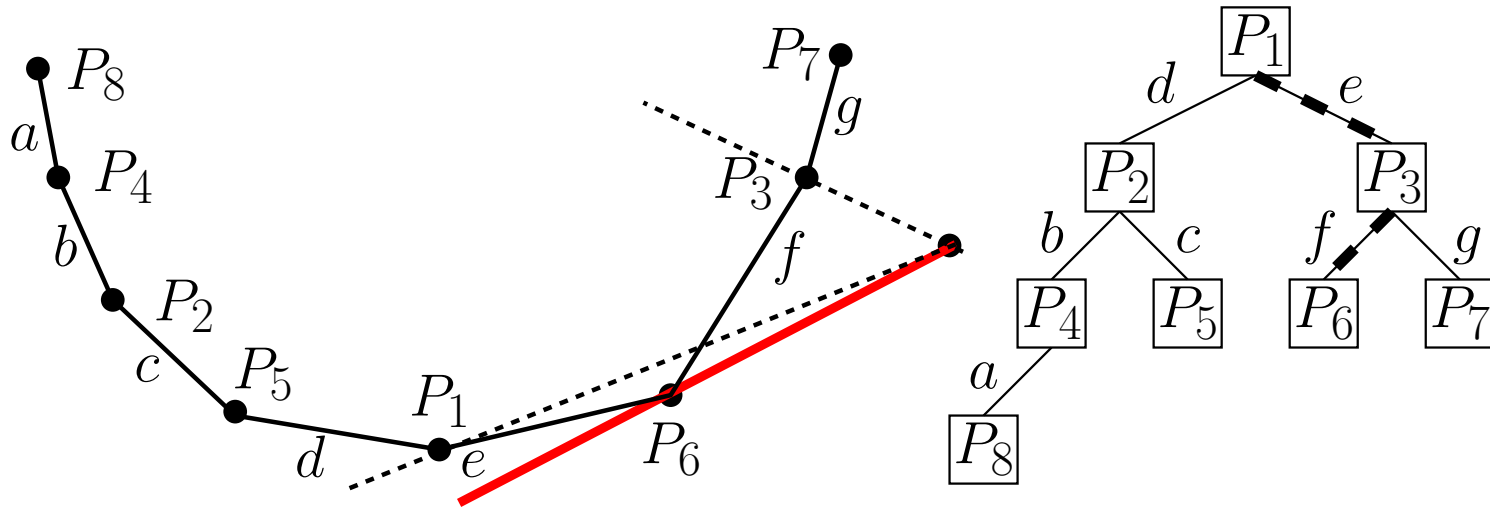
Datenstruktur Hourglass

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



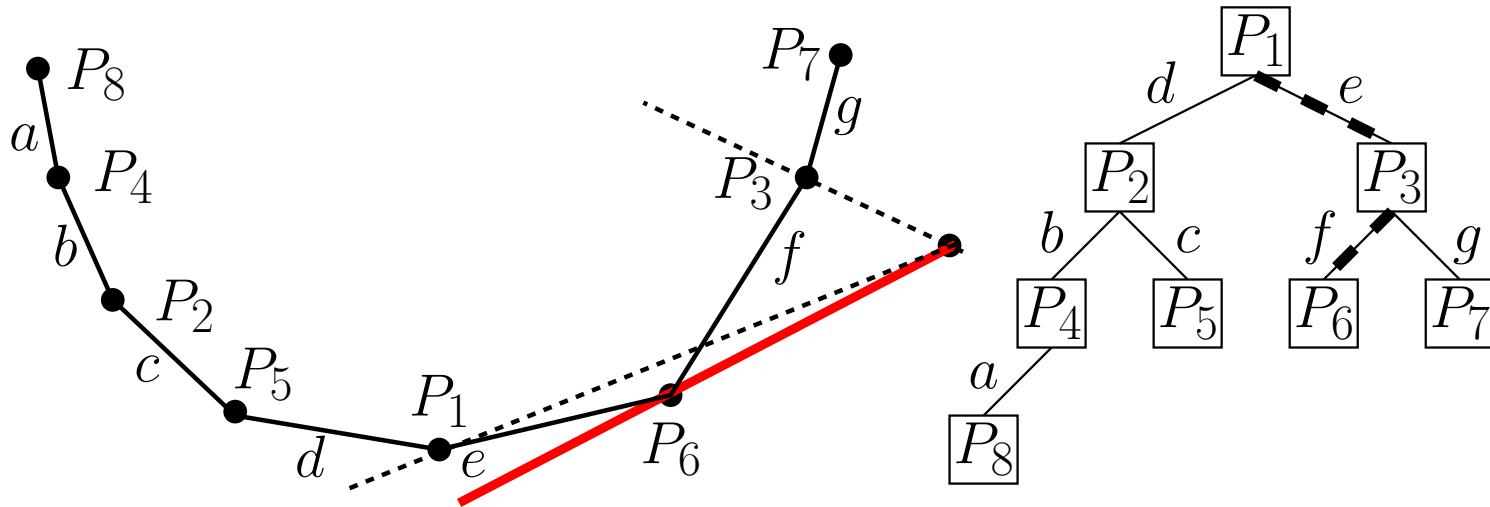
Datenstruktur Hourglass

- 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 $O(k)$



Effektiver Aufbau aller Sanduhren in $O(n)$

Effektiver Aufbau aller Sanduhren in $O(n)$

- Sanduhr $S(d_i, d_j)$ mit d_i und d_j

Effektiver Aufbau aller Sanduhren in $O(n)$

- Sanduhr $S(d_i, d_j)$ mit d_i und d_j
- Aus Sanduhren mit d' in $O(\log(\text{Komp}(S(d_i, d_j))))$

Effektiver Aufbau aller Sanduhren in $O(n)$

- Sanduhr $S(d_i, d_j)$ mit d_i und d_j
- Aus Sanduhren mit d' in $O(\log(\text{Komp}(S(d_i, d_j))))$
- Entspricht $O(\text{height}(d'))$

Effektiver Aufbau aller Sanduhren in $O(n)$

- Sanduhr $S(d_i, d_j)$ mit d_i und d_j
- Aus Sanduhren mit d' in $O(\log(\text{Komp}(S(d_i, d_j))))$
- Entspricht $O(\text{height}(d'))$
- Abschätzen durch $O(\text{height}(d_i))$ (größere Höhe d_i)

Effektiver Aufbau aller Sanduhren in $O(n)$

- Sanduhr $S(d_i, d_j)$ mit d_i und d_j
- Aus Sanduhren mit d' in $O(\log(\text{Komp}(S(d_i, d_j))))$
- Entspricht $O(\text{height}(d'))$
- Abschätzen durch $O(\text{height}(d_i))$ (größere Höhe d_i)
- $2 * \text{height}(d_i)$ viele solcher d' für d_i

Effektiver Aufbau aller Sanduhren in $O(n)$

- Sanduhr $S(d_i, d_j)$ mit d_i und d_j
- Aus Sanduhren mit d' in $O(\log(\text{Komp}(S(d_i, d_j))))$
- Entspricht $O(\text{height}(d'))$
- Abschätzen durch $O(\text{height}(d_i))$ (größere Höhe d_i)
- $2 * \text{height}(d_i)$ viele solcher d' für d_i
- Für alle Diagonalen!

Effektiver Aufbau aller Sanduhren in $O(n)$

- Sanduhr $S(d_i, d_j)$ mit d_i und d_j
- Aus Sanduhren mit d' in $O(\log(\text{Komp}(S(d_i, d_j))))$
- Entspricht $O(\text{height}(d'))$
- Abschätzen durch $O(\text{height}(d_i))$ (größere Höhe d_i)
- $2 * \text{height}(d_i)$ viele solcher d' für d_i
- Für alle Diagonalen!

- Sum. über alle Höhen:
$$\sum_{h=1}^{\log_3 n} 2 \times \left(\left(\frac{2}{3} \right)^h \times n \right) \times h^2$$

Effektiver Aufbau aller Sanduhren in $O(n)$

- Sanduhr $S(d_i, d_j)$ mit d_i und d_j
- Aus Sanduhren mit d' in $O(\log(\text{Komp}(S(d_i, d_j))))$
- Entspricht $O(\text{height}(d'))$
- Abschätzen durch $O(\text{height}(d_i))$ (größere Höhe d_i)
- $2 * \text{height}(d_i)$ viele solcher d' für d_i
- Für alle Diagonalen!

- Sum. über alle Höhen: $\sum_{h=1}^{\log_3 n} 2 \times \left(\left(\frac{2}{3} \right)^h \times n \right) \times h^2 \in O(n)$

Effektiver Aufbau aller Sanduhren in $O(n)$

- Sanduhr $S(d_i, d_j)$ mit d_i und d_j
- Aus Sanduhren mit d' in $O(\log(\text{Komp}(S(d_i, d_j))))$
- Entspricht $O(\text{height}(d'))$
- Abschätzen durch $O(\text{height}(d_i))$ (größere Höhe d_i)
- $2 * \text{height}(d_i)$ viele solcher d' für d_i
- Für alle Diagonalen!

- Sum. über alle Höhen: $\sum_{h=1}^{\log_3 n} 2 \times \left(\left(\frac{2}{3} \right)^h \times n \right) \times h^2 \in O(n)$

- **Lemma 1.14**

Konstruktion \hat{G} + Sanduhren

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

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:
 - \hat{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 \hat{G}
 - Sanduhr $S(d_p, d_q)$ aus Sanduhren entlang des Pfades
 - Länge oder kürzester Weg

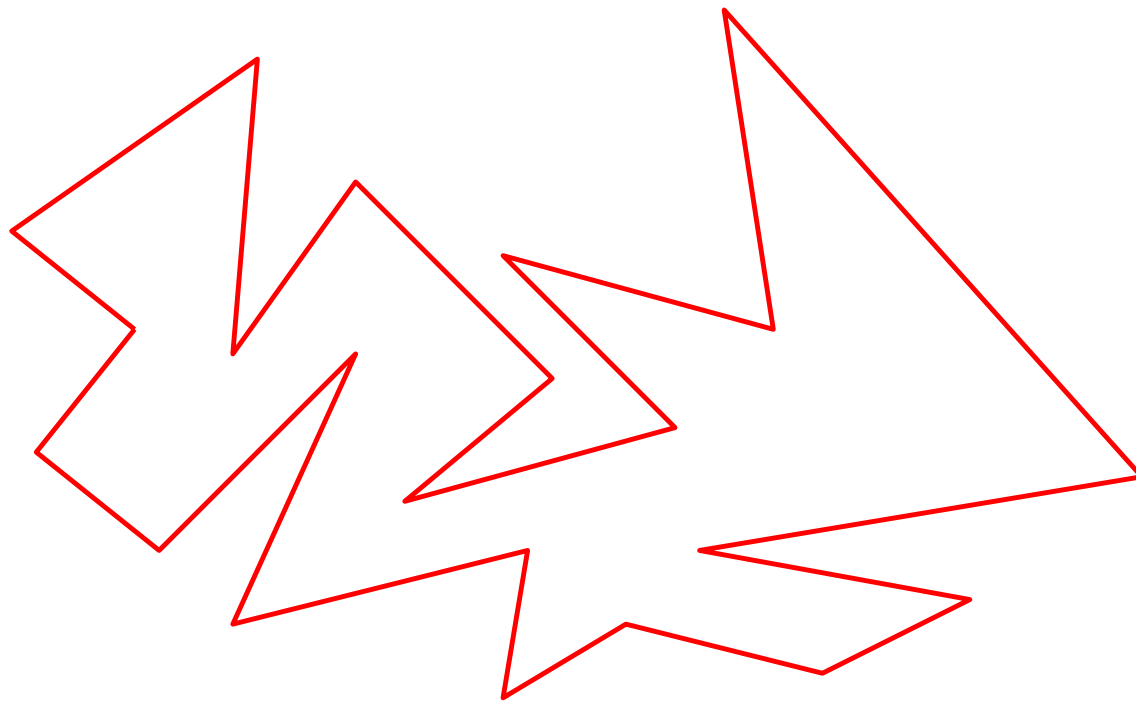
Guibas/Hershberger: Laufzeiten

- Datenstrukturen: \hat{G} , Rooted Tree, Sanduhren, Trapezzerlegung
- Preprocessing Zeit: $O(n)$
- Komplexität: $O(n)$
- Lokalisation Dreiecke: $O(\log n)$
- Pfad in \hat{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

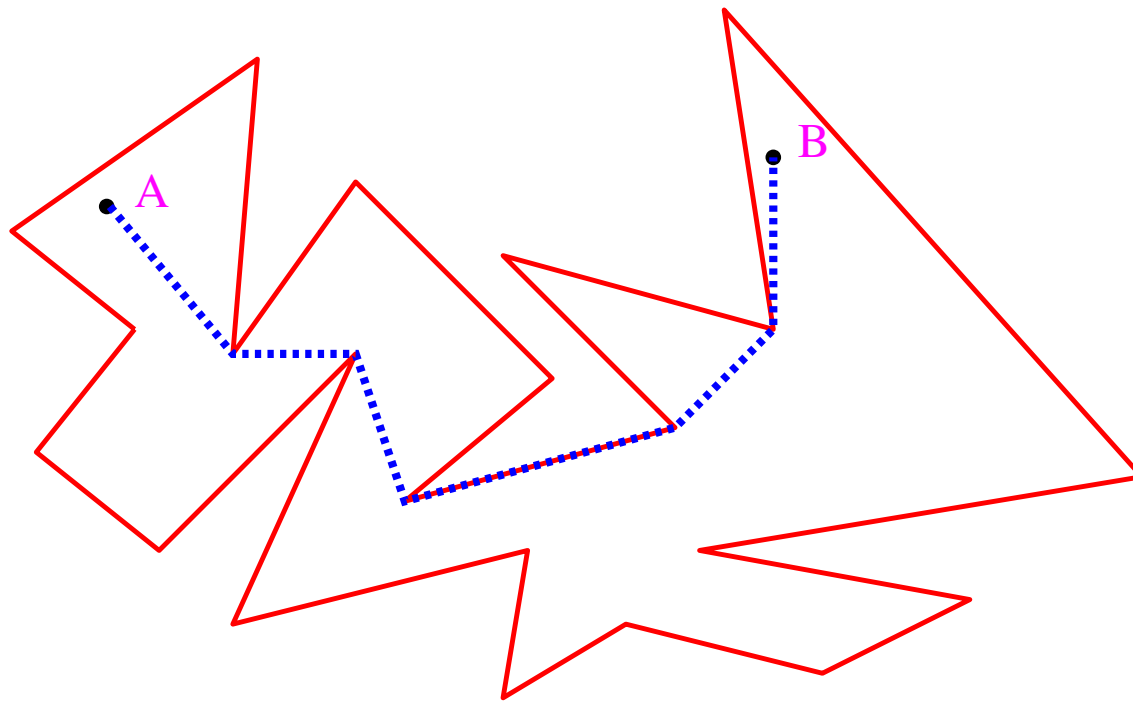
1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon P



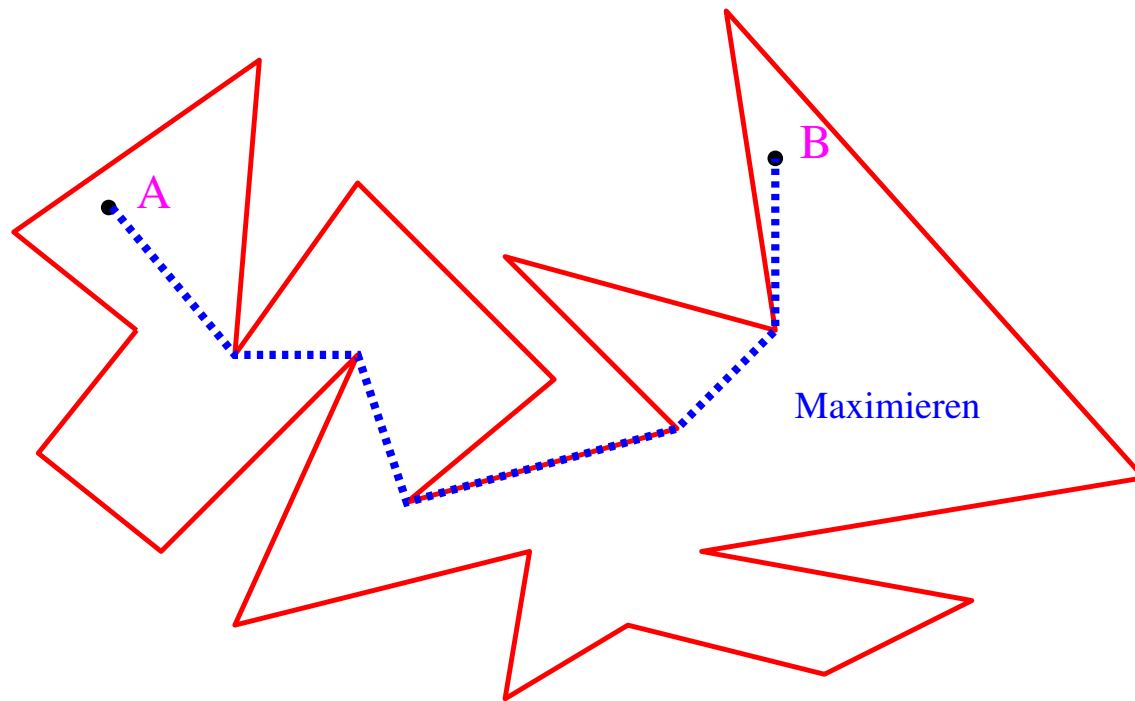
1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon P
- Längster Kürzester Weg zwischen zwei Punkten



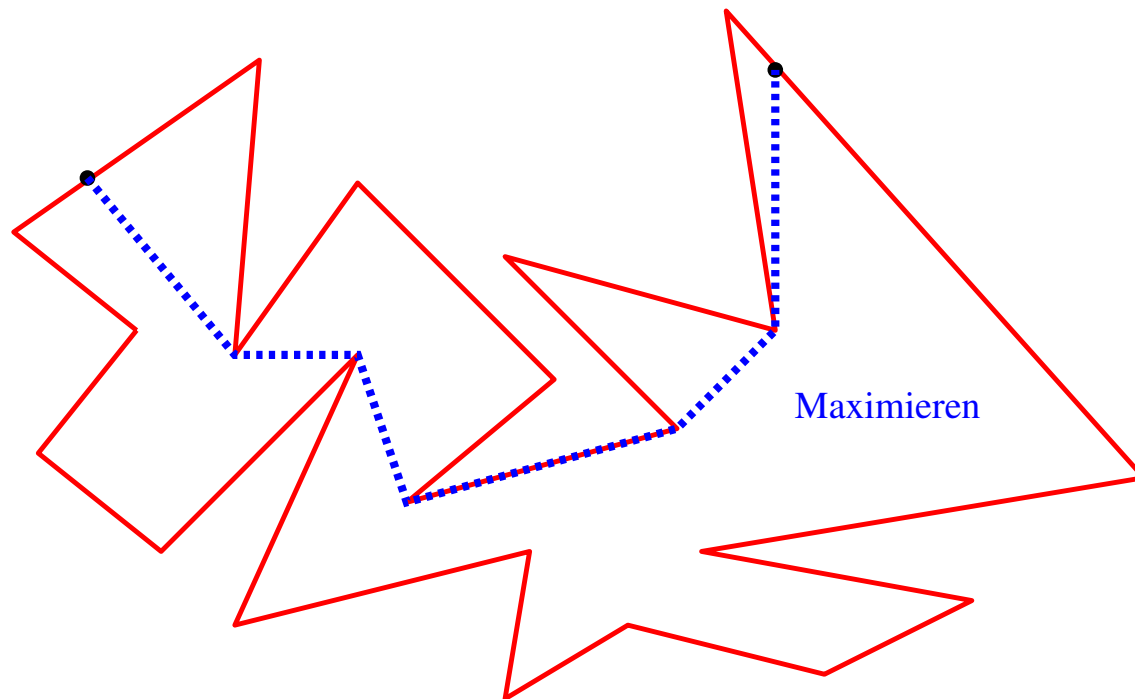
1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon P
- Längster kürzester Weg zwischen zwei Punkten
- Endpunkte sind Ecken des Polygons



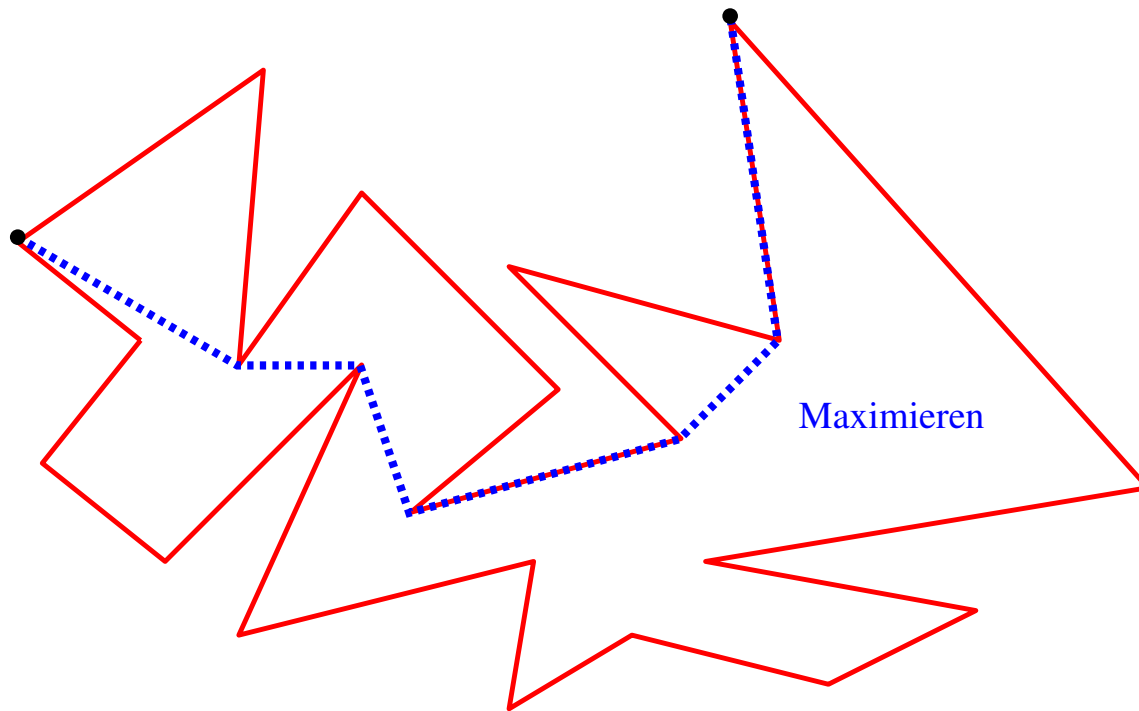
1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon P
- Längster kürzester Weg zwischen zwei Punkten
- Endpunkte sind Ecken des Polygons



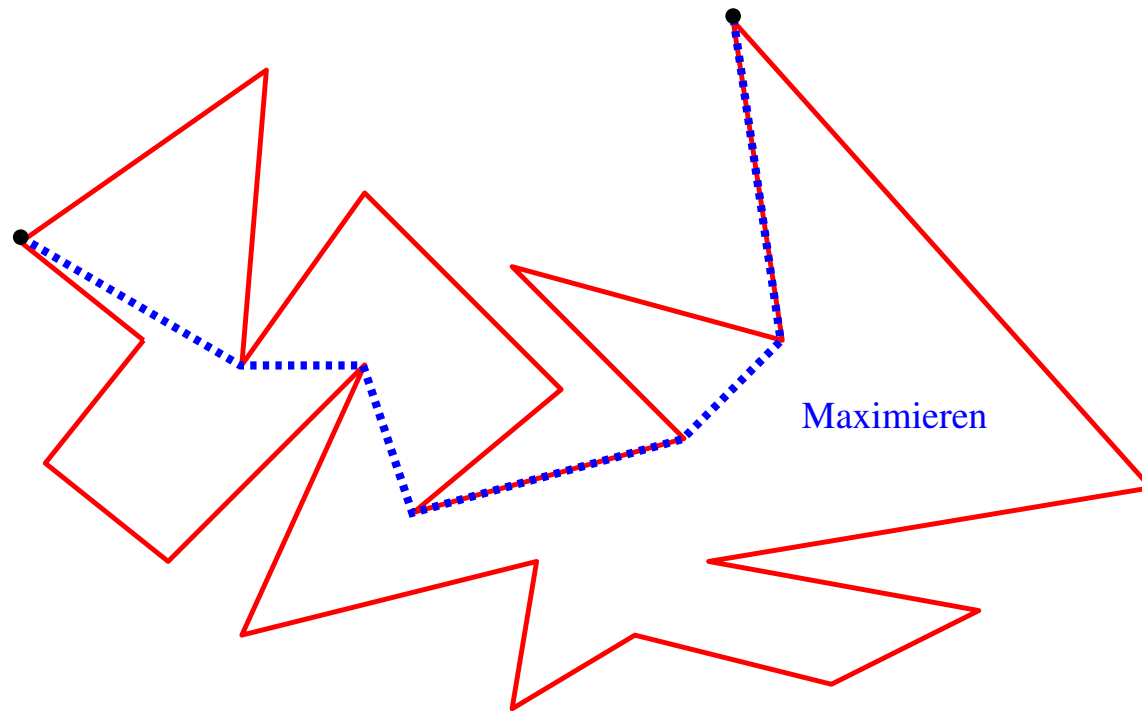
1.2.3 Durchmesser einfacher Polygone

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



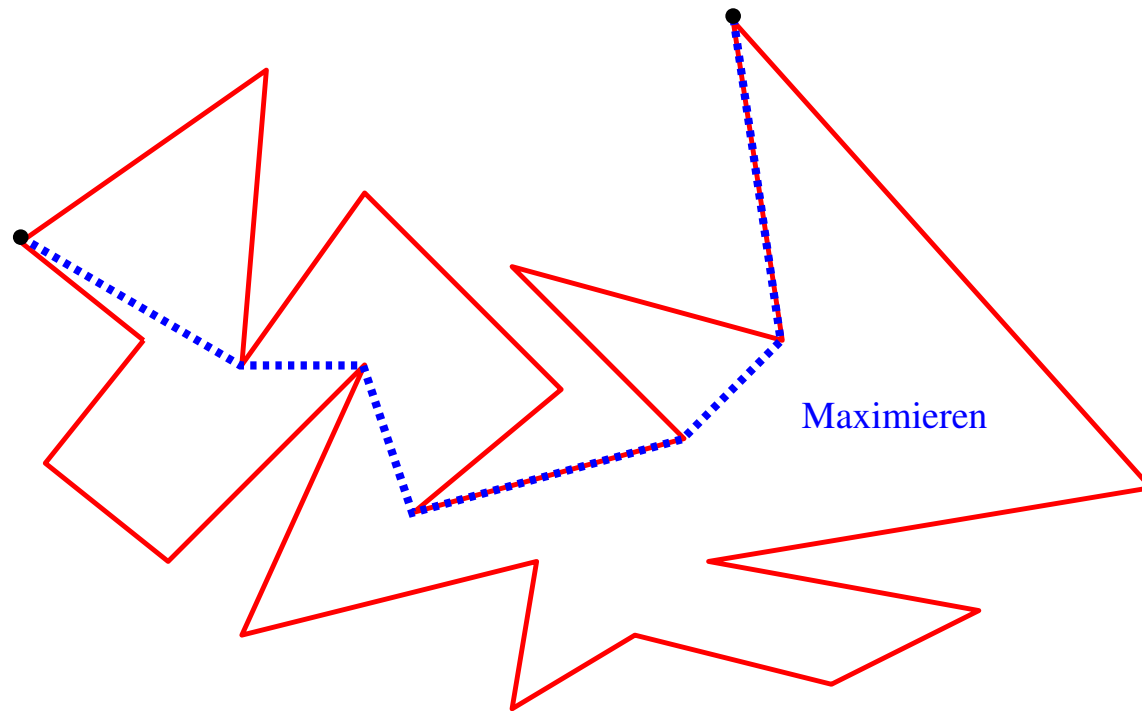
1.2.3 Durchmesser einfacher Polygone

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



1.2.3 Durchmesser einfacher Polygone

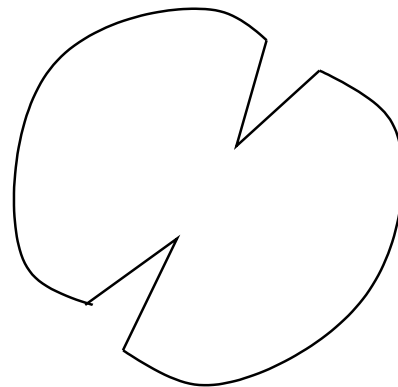
- Einfaches Polygon P
- Längster Kürzester Weg zwischen zwei Punkten
- Endpunkte sind Ecken des Polygons n^2 Kandidatenpaare
- Formal: $\max_{p_i, p_j} \text{Ecken von } P \ d(p_i, p_j)$



Idee der Berechnung:

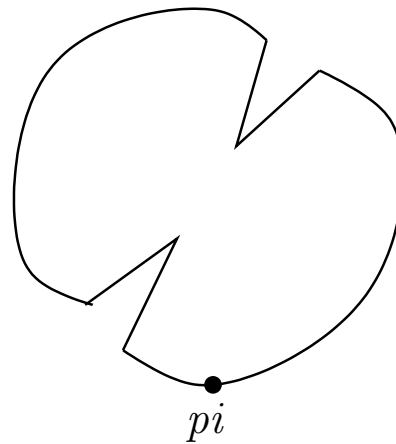
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



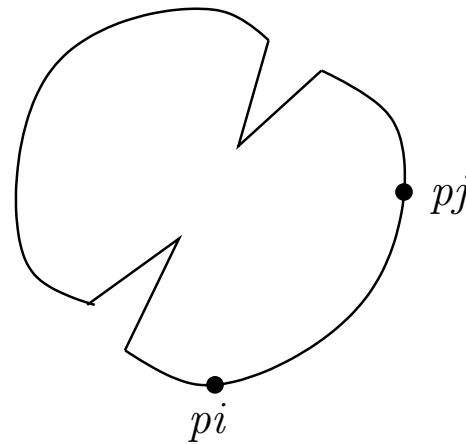
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



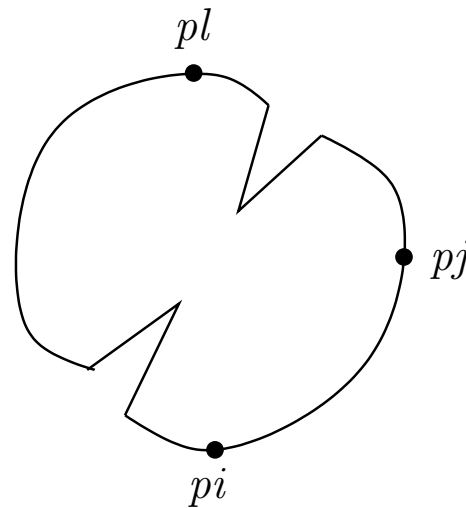
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



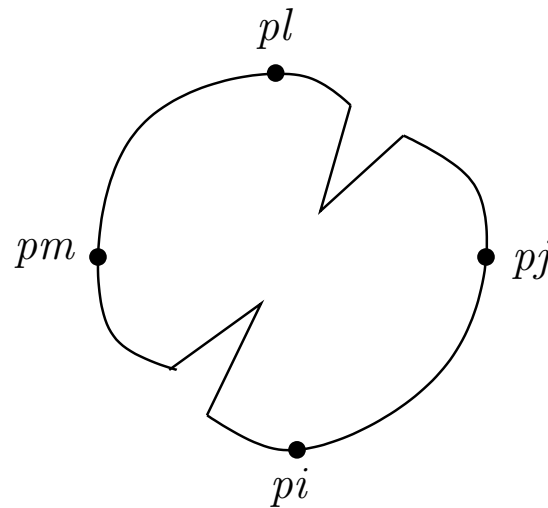
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



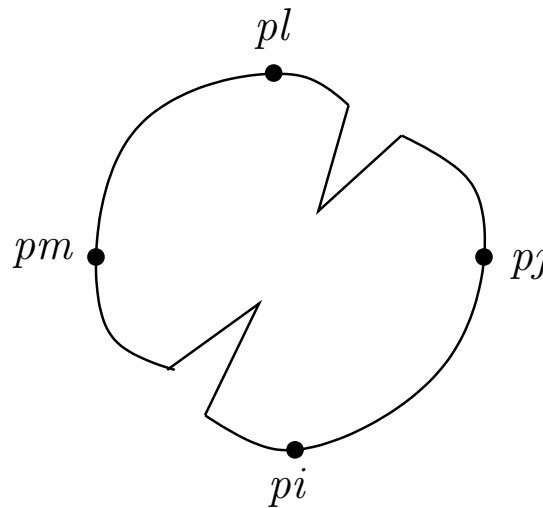
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



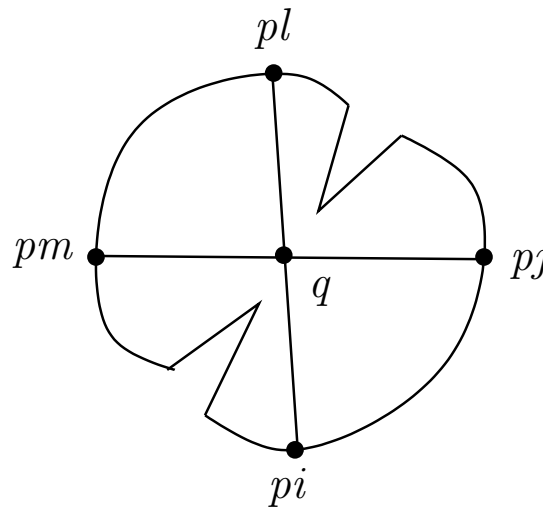
Idee der Berechnung:

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



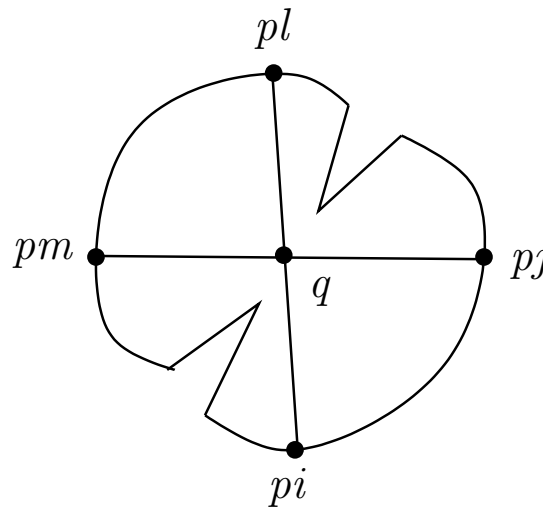
Idee der Berechnung:

- 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



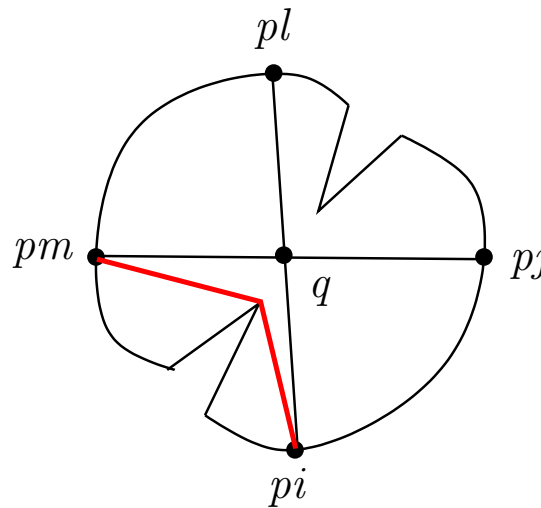
Idee der Berechnung:

- 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!!



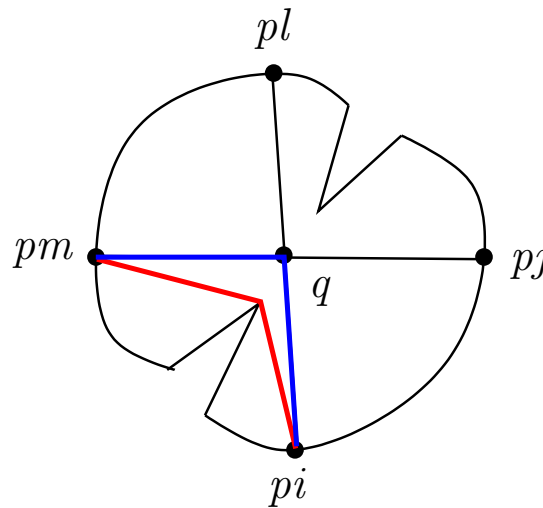
Idee der Berechnung:

- 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!!



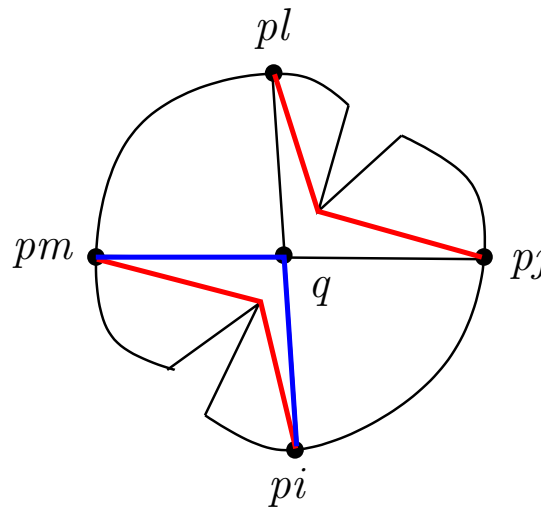
Idee der Berechnung:

- 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!!

