Offline Bewegungsplanung: Preprocessing und Durchmesser

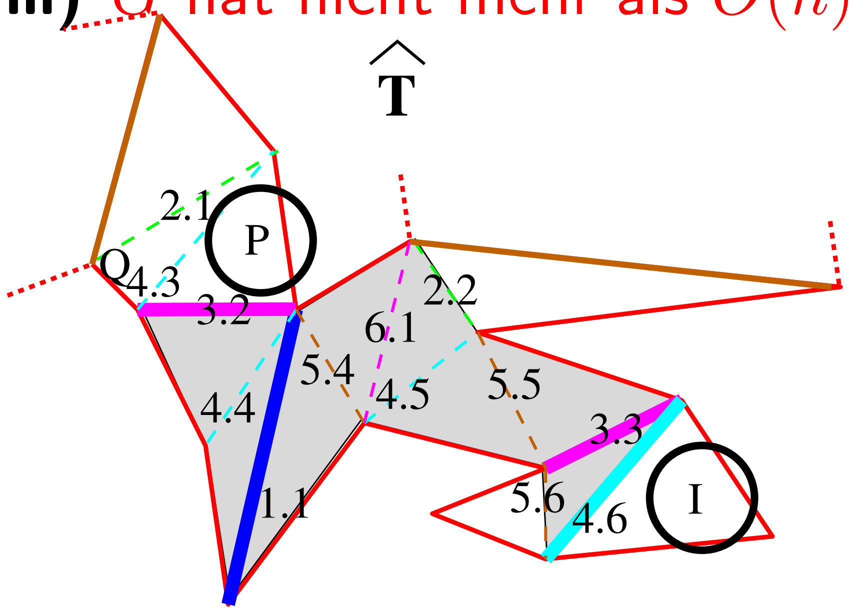
Elmar Langetepe
University of Bonn

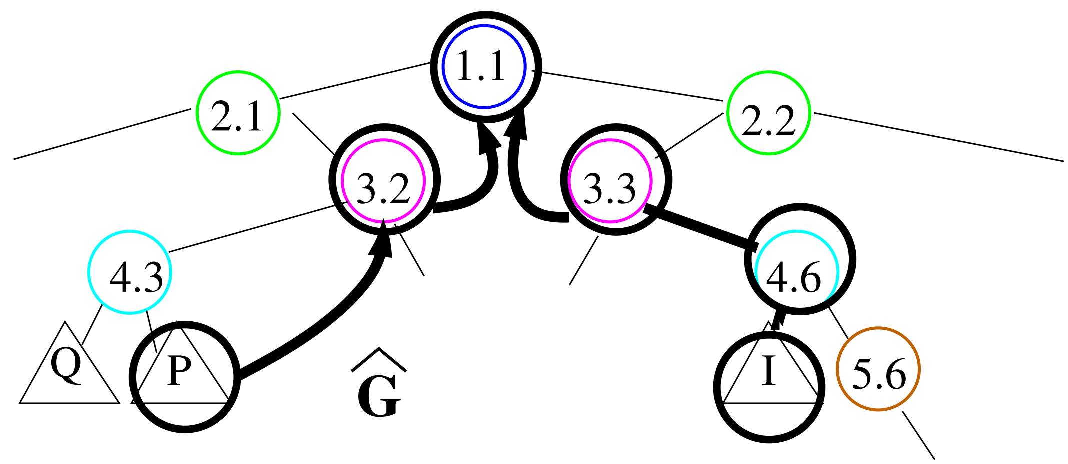
Eigenschaften von $\widehat{G}$: Lemma 1.13

i) Pfad zwischen zwei Dreiecken entlang sukzessiver Diagonalen existiert!

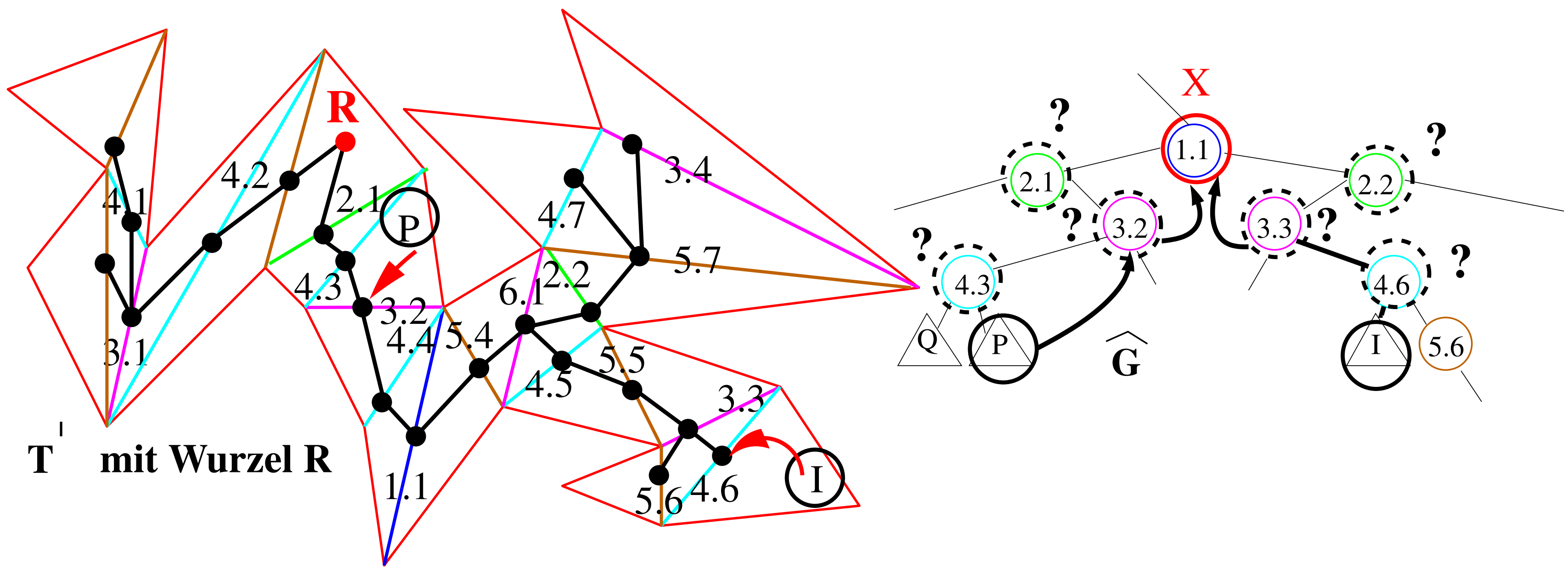
ii) Wir finden den Weg in $O(\log n)$ Zeit!

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





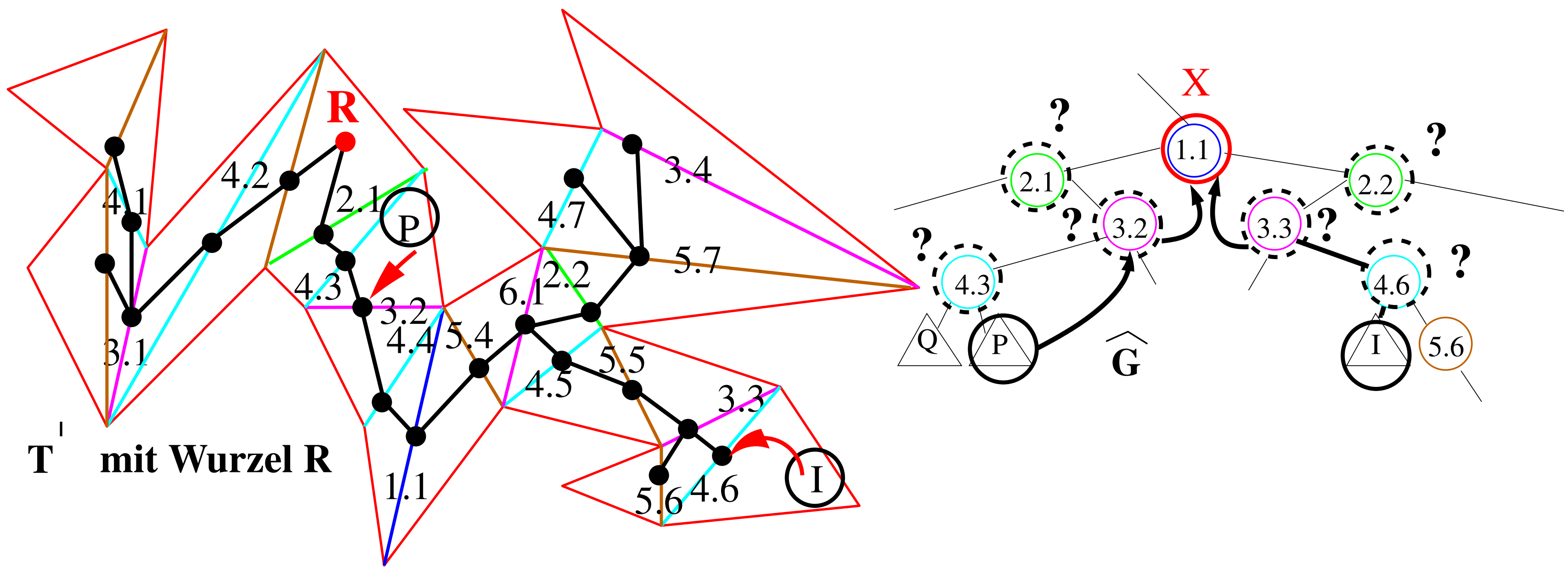
ii) Finden des Weges in $\widehat{G}$!



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

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

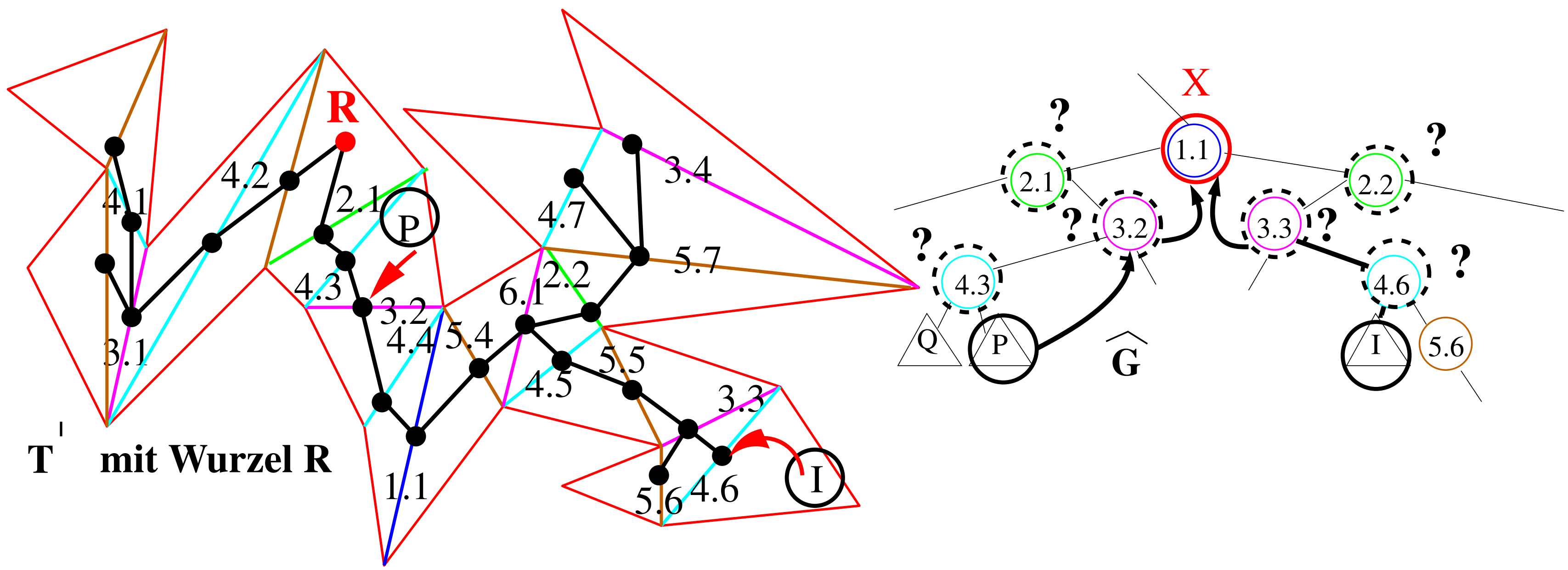
- Gemeinsamer Vorgänger X minimaler Höhe in $O(\log n)$ in $\widehat{T}$



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 $\widehat{T}$
- Benutze leicht geänderten Dualen Baum von T
- Frage: Liegt Diagonale d auf dem Pfad von P nach I?

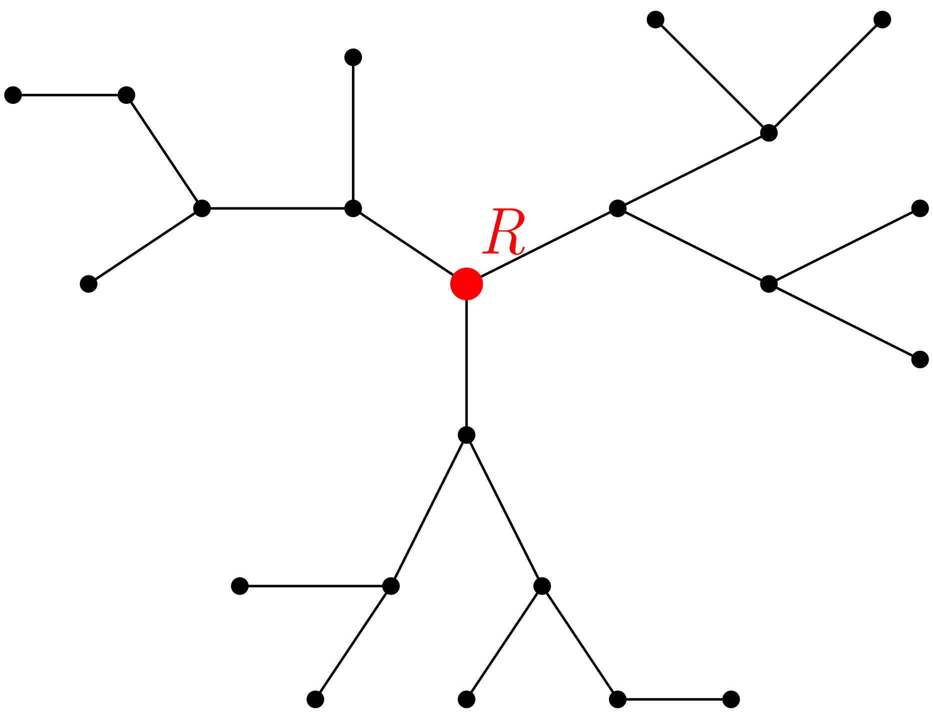


d Vorgänger entweder von delta(P) ODER delta(I) in Bezug auf R

Vorgängeranfrage in Baum

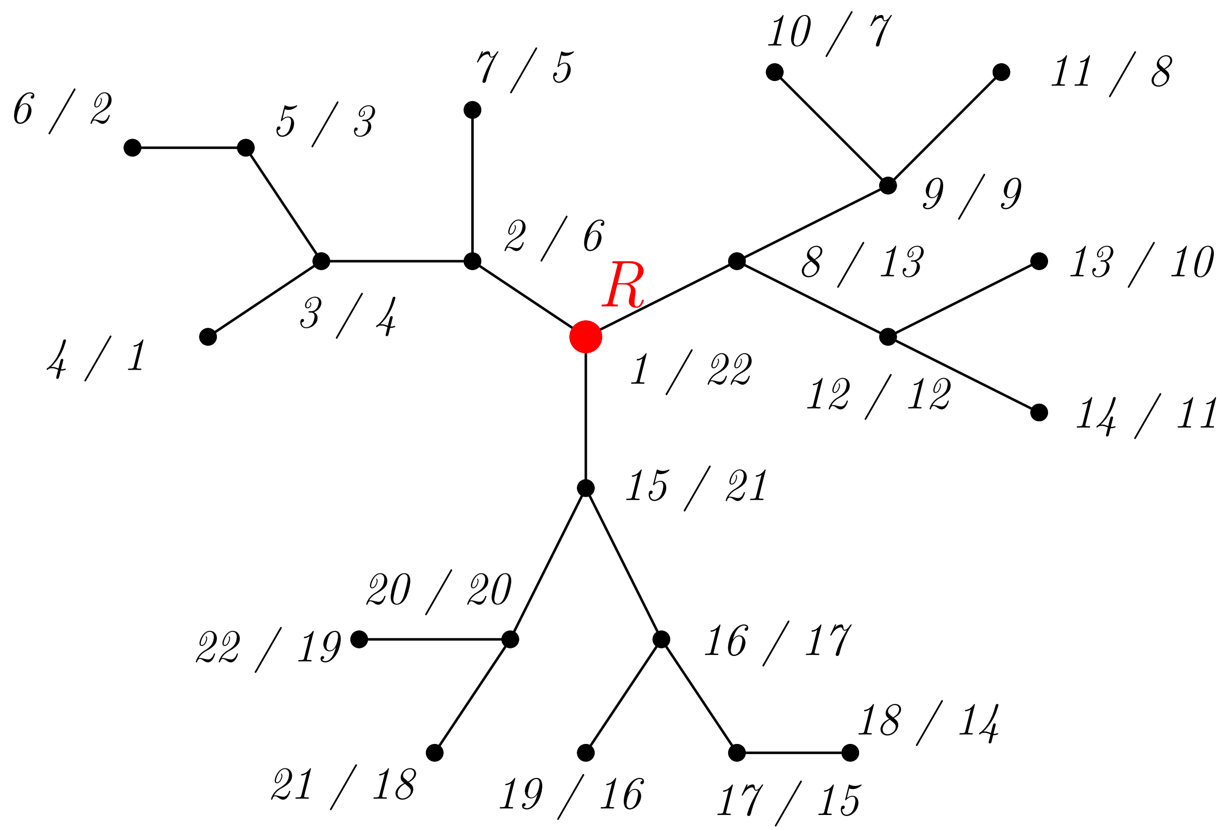
Vorgängeranfrage in Baum

- Preorder/Postorder,



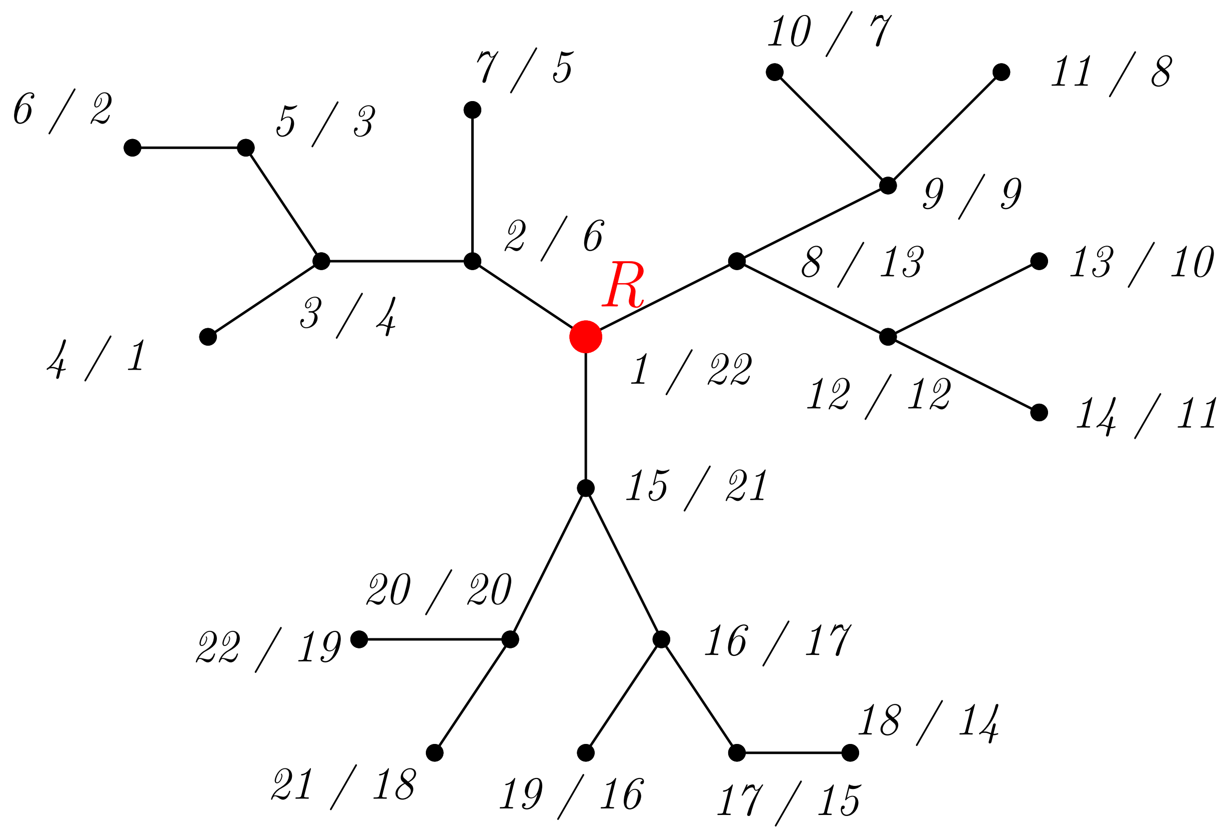
Vorgängeranfrage in Baum

- Preorder/Postorder, DFS und Labelling



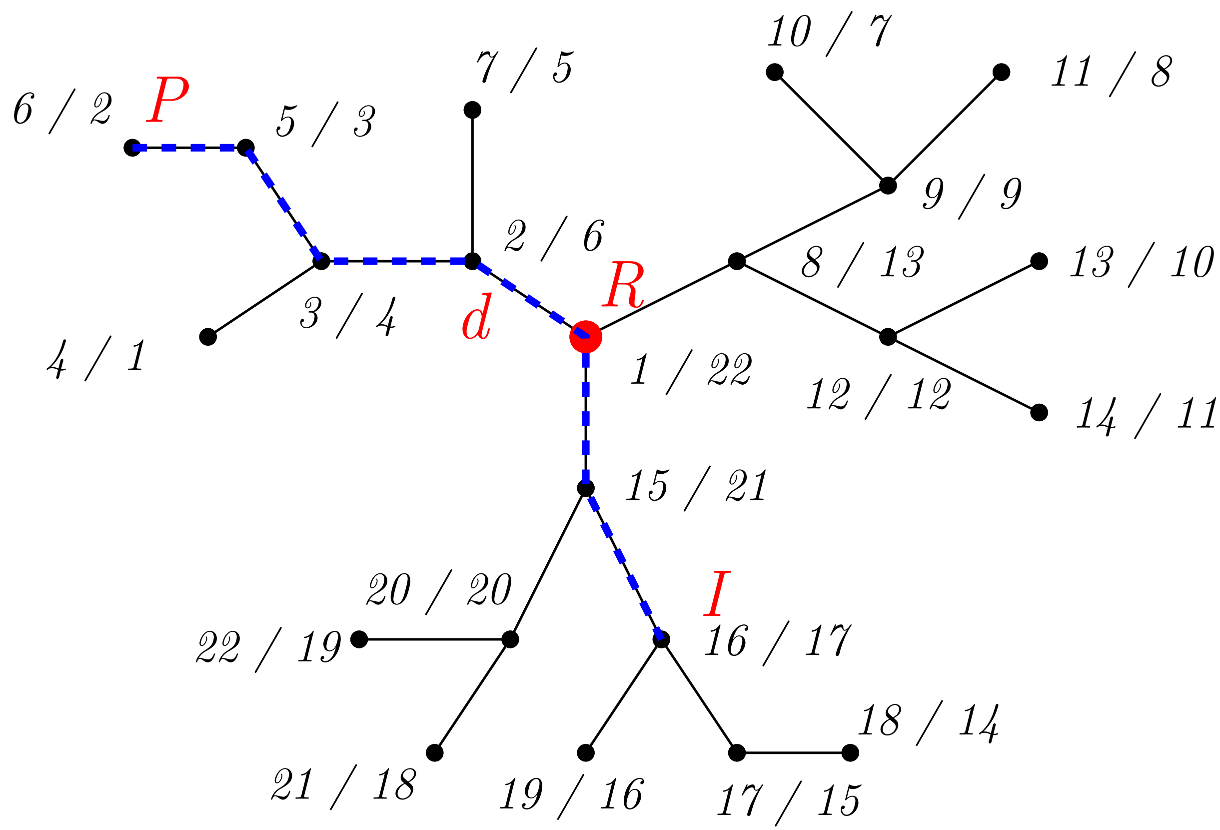
Vorgängeranfrage in Baum

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



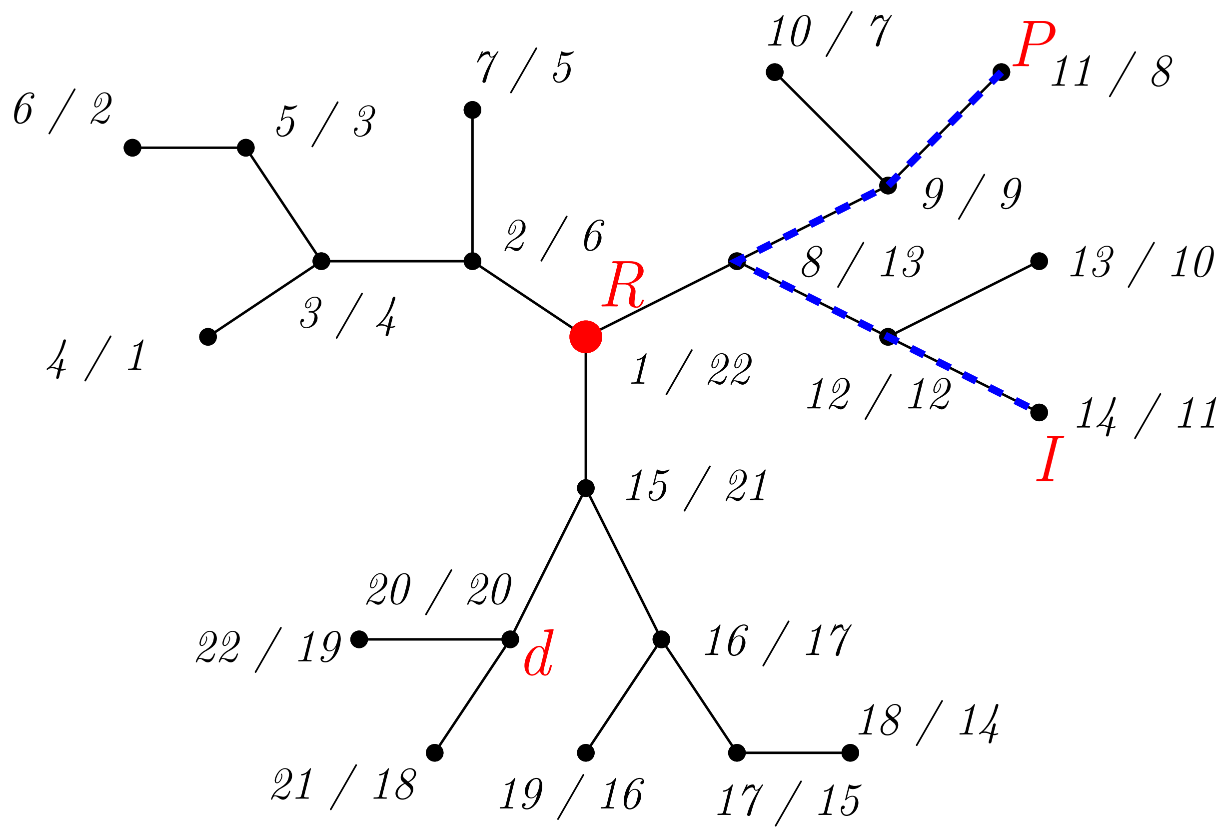
Vorgängeranfrage in Baum

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



Vorgängeranfrage in Baum

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

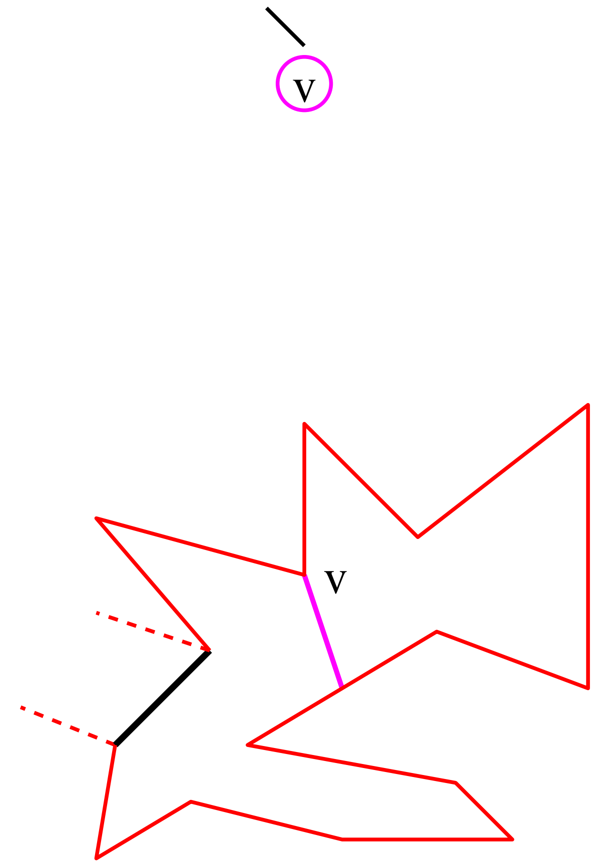
i) Pfad zwischen zwei Dreiecken existiert!

i) Länge ist in $O(\log n)$!

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

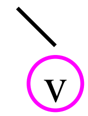
iii) $\widehat{G}$ hat $O(n)$ Kanten !

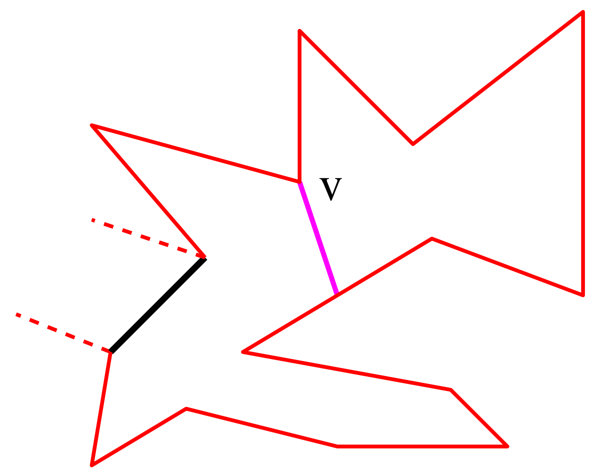
iii) Komplexität von $\widehat{G}$



iii) Komplexität von $\widehat{G}$

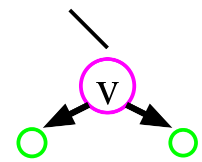
- Untere Kanten von v aus: max $2 \times \texttt{height}(\texttt{v})$

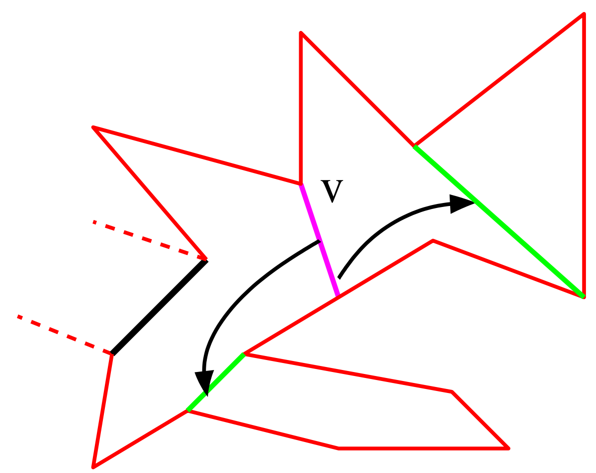




iii) Komplexität von $\widehat{G}$

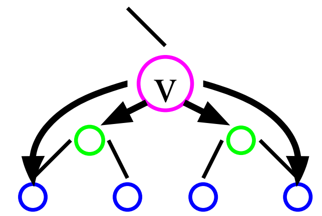
- Untere Kanten von v aus: max $2 \times \texttt{height}(\texttt{v})$

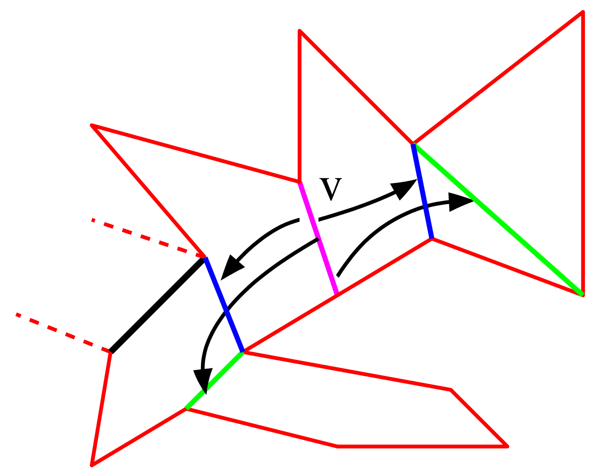




iii) Komplexität von $\widehat{G}$

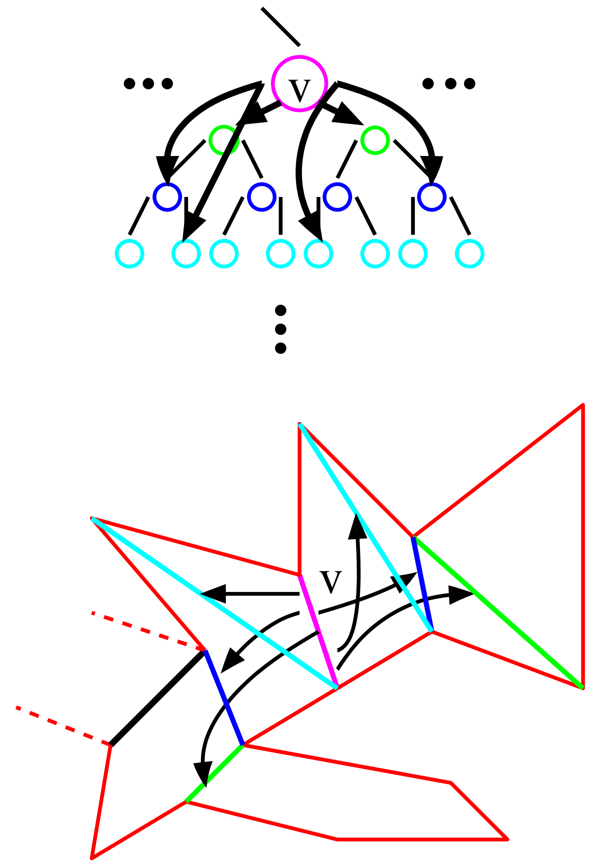
- Untere Kanten von v aus: max $2 \times \texttt{height}(\texttt{v})$





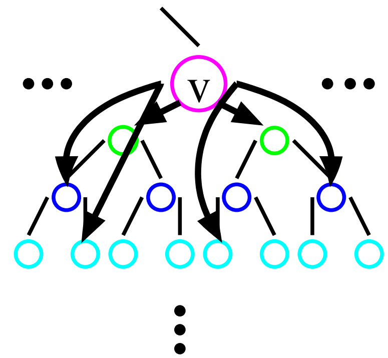
iii) Komplexität von $\widehat{G}$

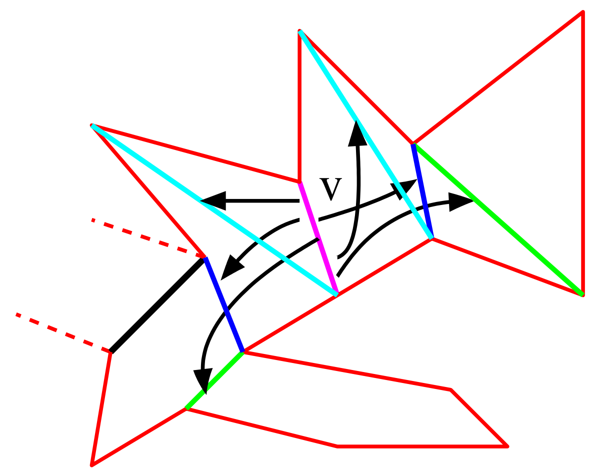
- Untere Kanten von v aus: max $2 \times \texttt{height}(\mathtt{v})$



iii) Komplexität von $\widehat{G}$

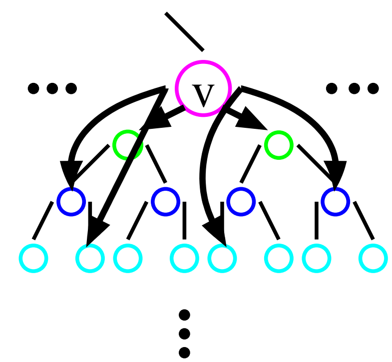
- Untere Kanten von v aus: max $2 \times \texttt{height}(\texttt{v})$
- Balance: Teilbaum bei v hat $\geq C \cdot \left(\frac{3}{2}\right)^{\texttt{height}(\texttt{v})}$ Blätter (Tafel)

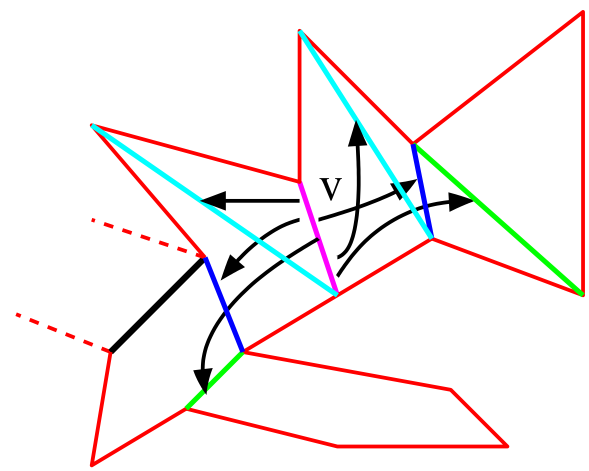




iii) Komplexität von $\widehat{G}$

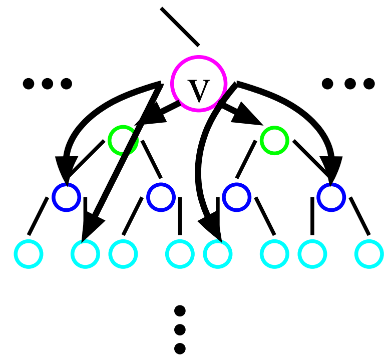
- Untere Kanten von v aus: max $2 \times \mathtt{height(v)}$
- Balance: Teilbaum bei v hat $\geq C \cdot \left(\frac{3}{2}\right)^{\mathtt{height(v)}}$ Blätter (Tafel)
- Anzahl Knoten der Höhe h: $\leq \frac{n}{C\left(\frac{3}{2}\right)^h} = \frac{n}{C}\left(\frac{2}{3}\right)^h$

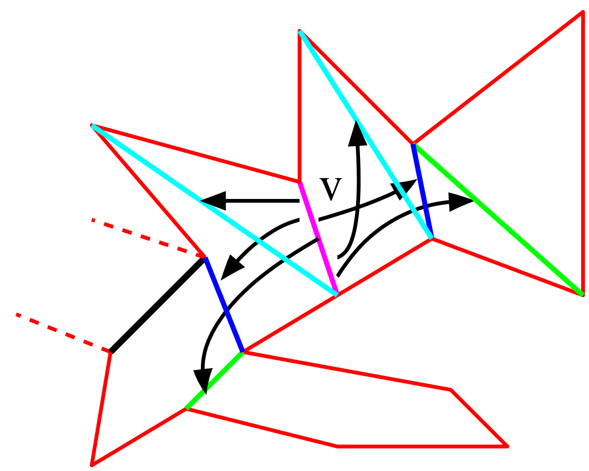




iii) Komplexität von $\widehat{G}$

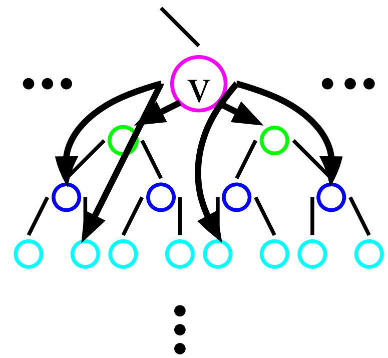
- Untere Kanten von v aus: max $2 \times \texttt{height}(\texttt{v})$
- Balance: Teilbaum bei v hat $\geq C \cdot \left(\frac{3}{2}\right)^{\texttt{height}(\texttt{v})}$ Blätter (Tafel)
- Anzahl Knoten der Höhe h: $\leq \frac{n}{C\left(\frac{3}{2}\right)^h} = \frac{n}{C}\left(\frac{2}{3}\right)^h$
-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



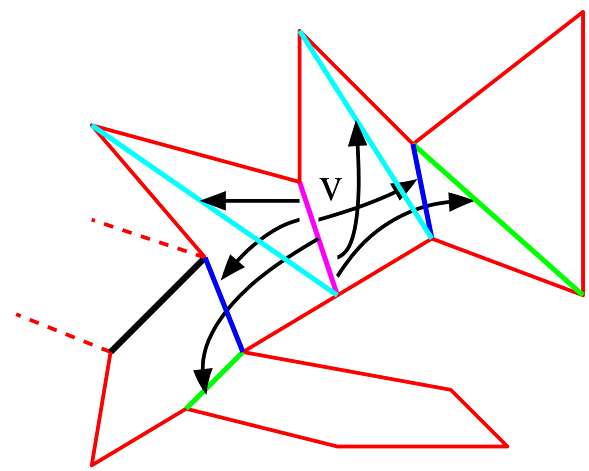


iii) Komplexität von $\widehat{G}$

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



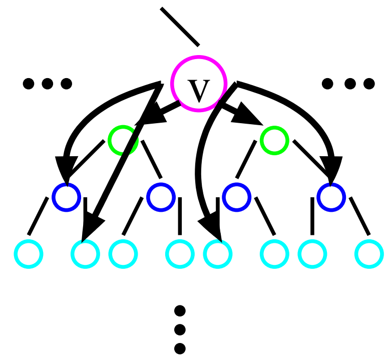
- Anzahl Knoten der Höhe h: $\leq \frac{n}{C\left(\frac{3}{2}\right)^h} = \frac{n}{C}\left(\frac{2}{3}\right)^h$



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

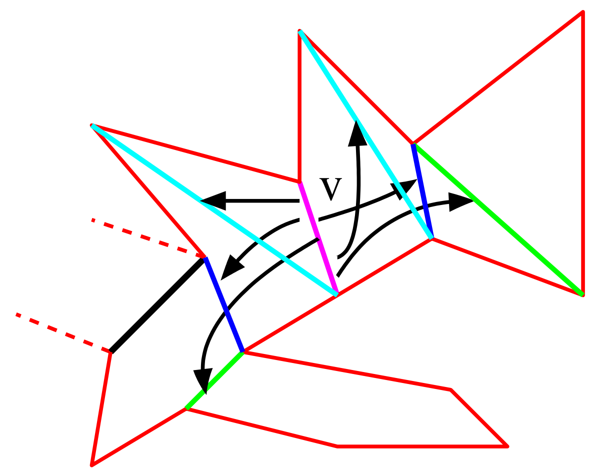


- Anzahl Knoten der Höhe h:

$$\leq \frac{n}{C\left(\frac{3}{2}\right)^h} = \frac{n}{C}\left(\frac{2}{3}\right)^h$$

- 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 $O(n)$

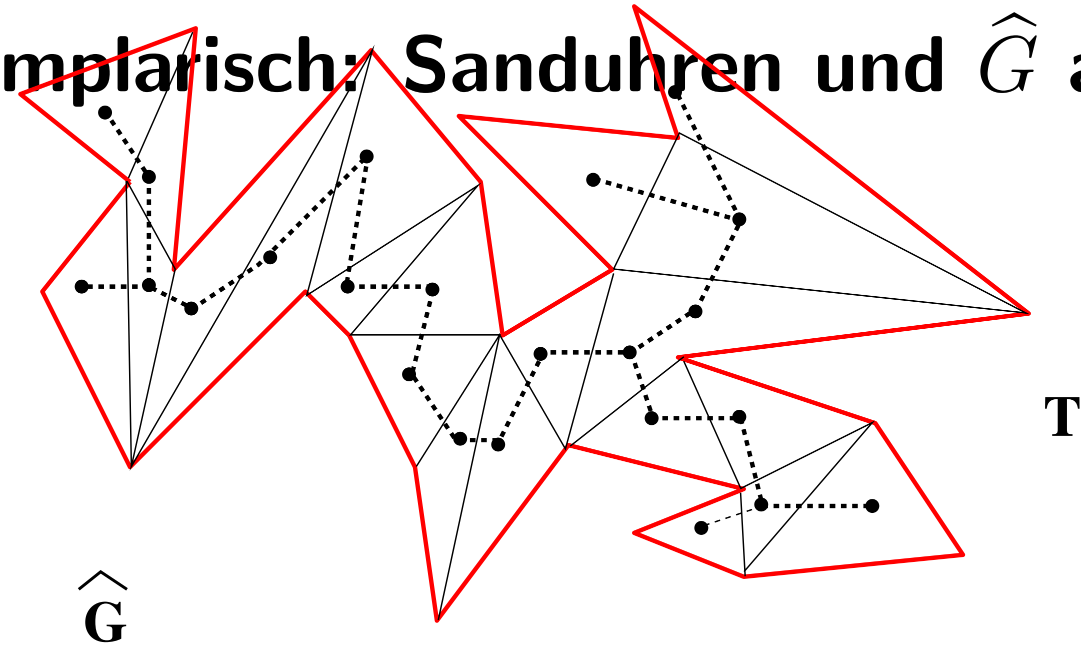
Exemplarisch: Sanduhren und $\widehat{G}$ aufbauen!

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!

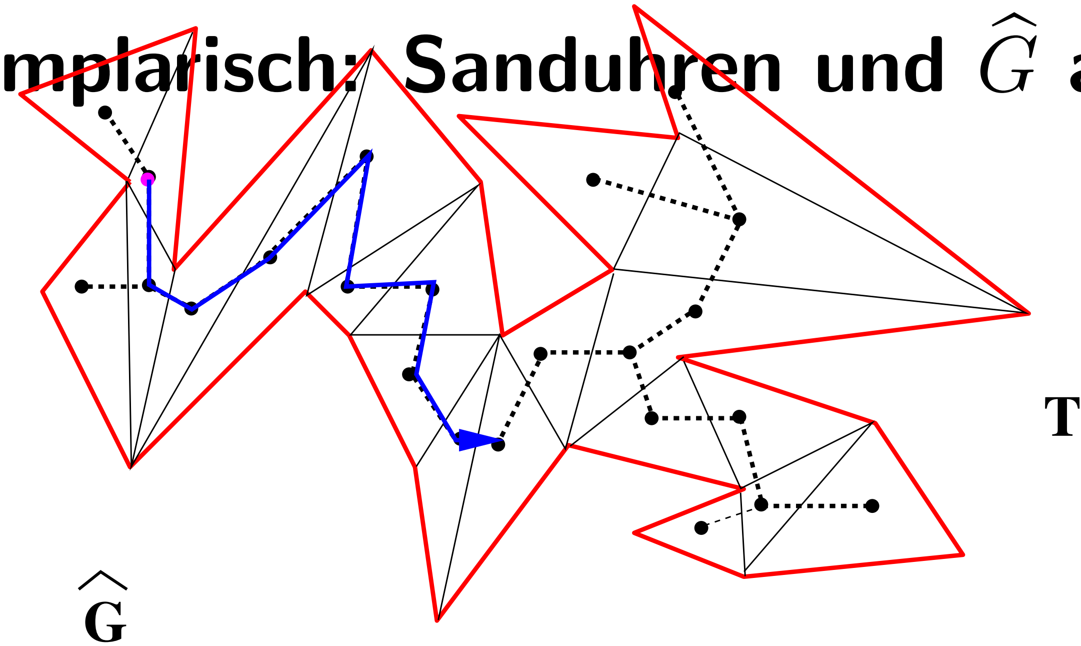


Durchlauf gemäß Cutting Theorem Durchlauf

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!

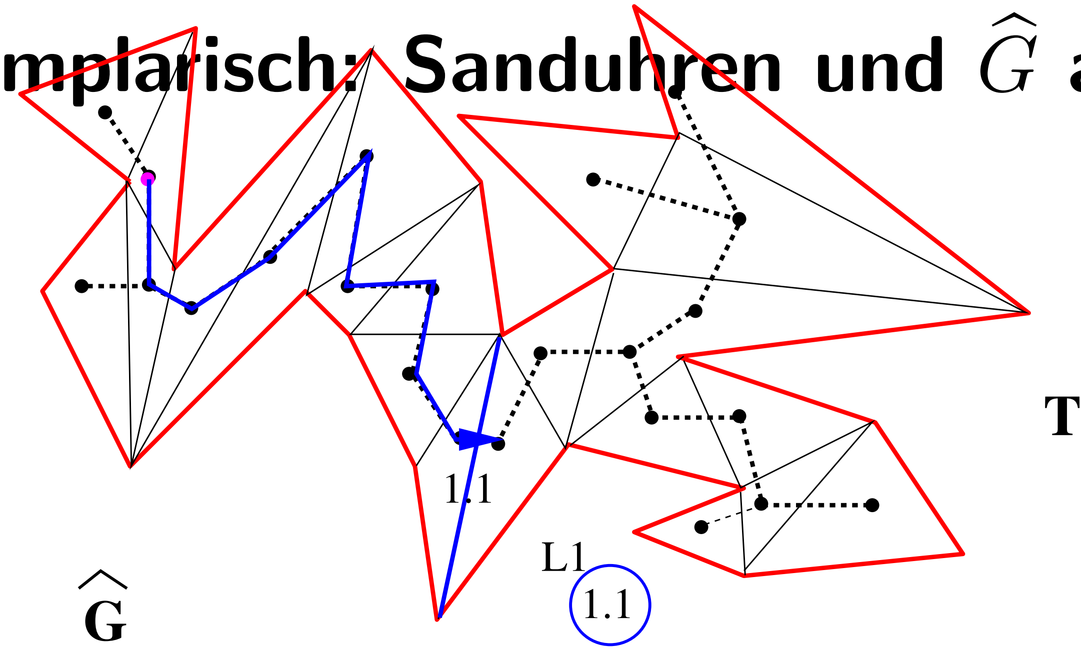


Durchlauf gemäß Cutting Theorem Durchlauf

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!

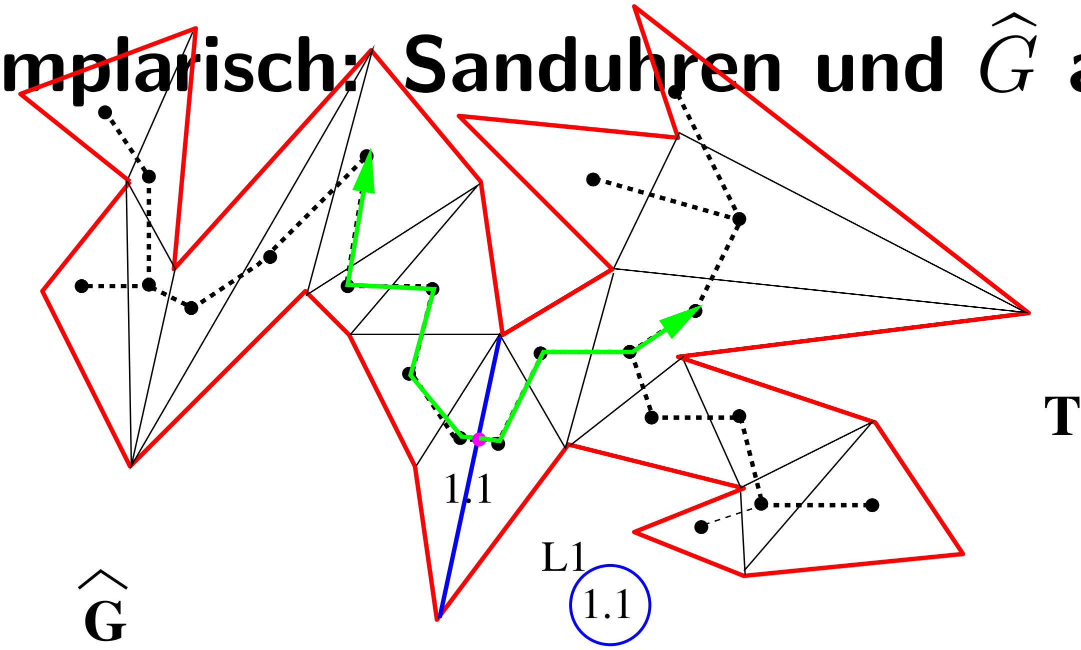


Durchlauf gemäß Cutting Theorem Durchlauf

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!

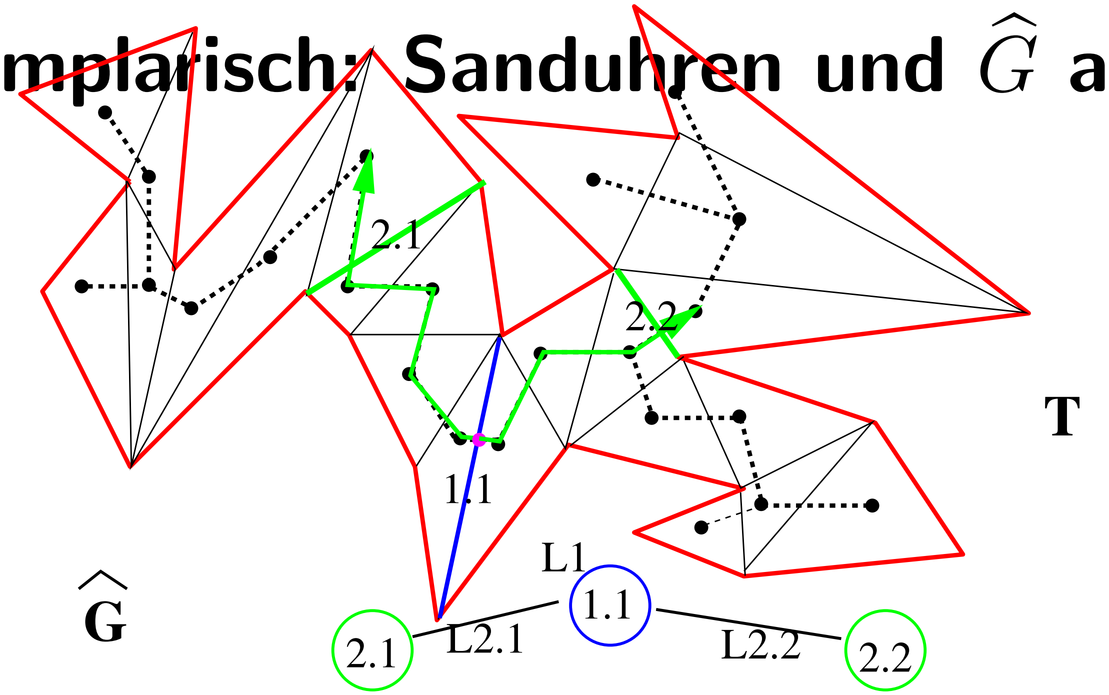


Durchlauf gemäß Cutting Theorem Durchlauf

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!

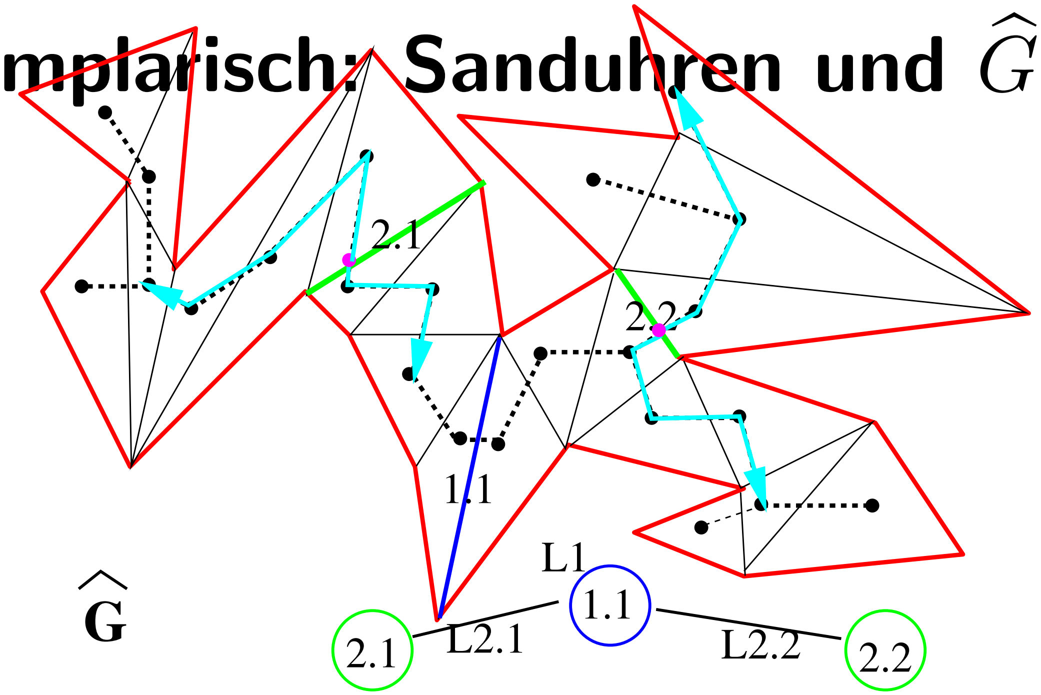


Durchlauf gemäß Cutting Theorem Durchlauf

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!

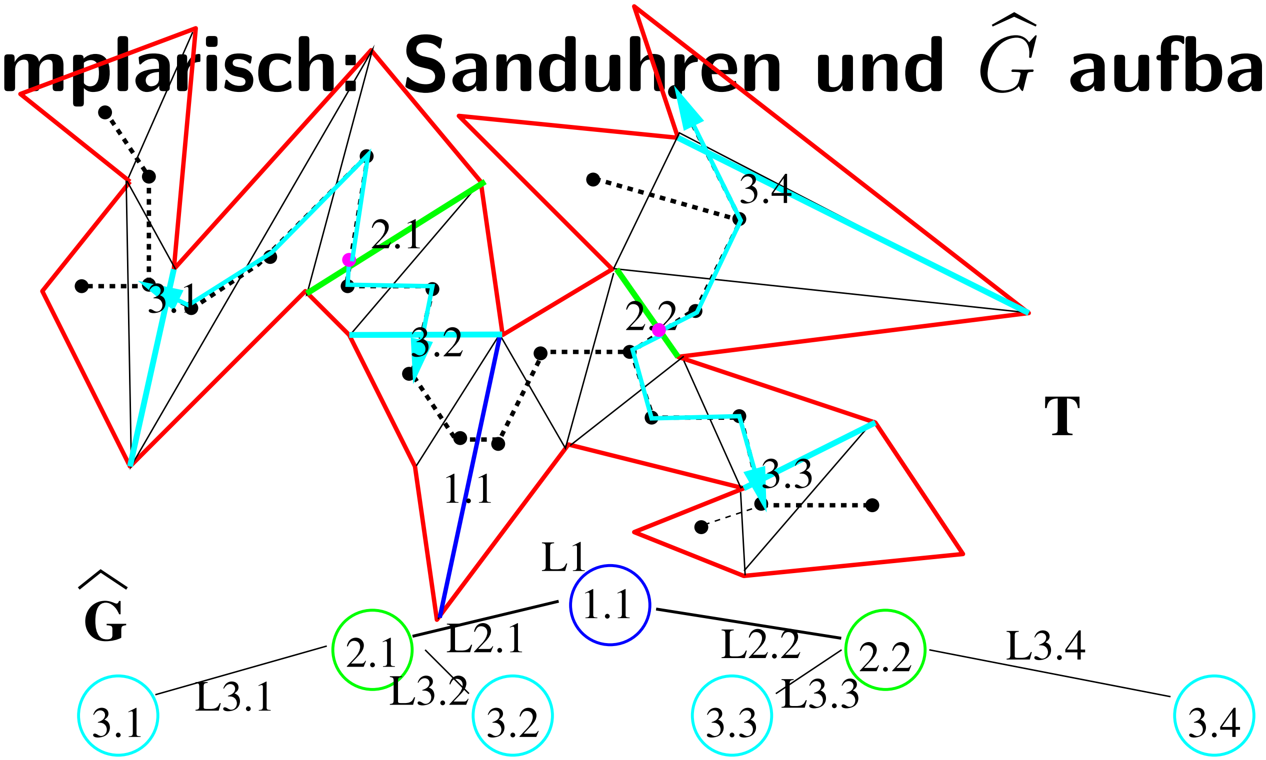


Durchlauf gemäß Cutting Theorem Durchlauf

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!

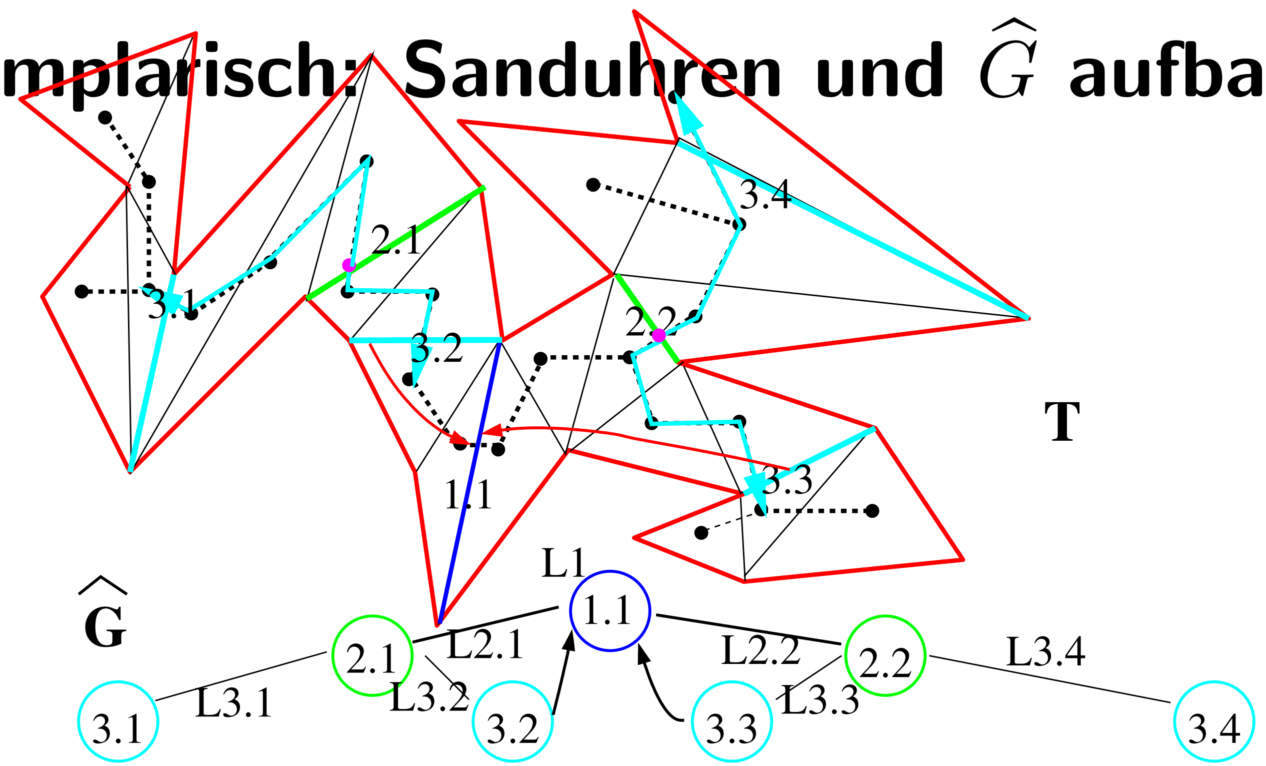


Durchlauf gemäß Cutting Theorem Durchlauf

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!

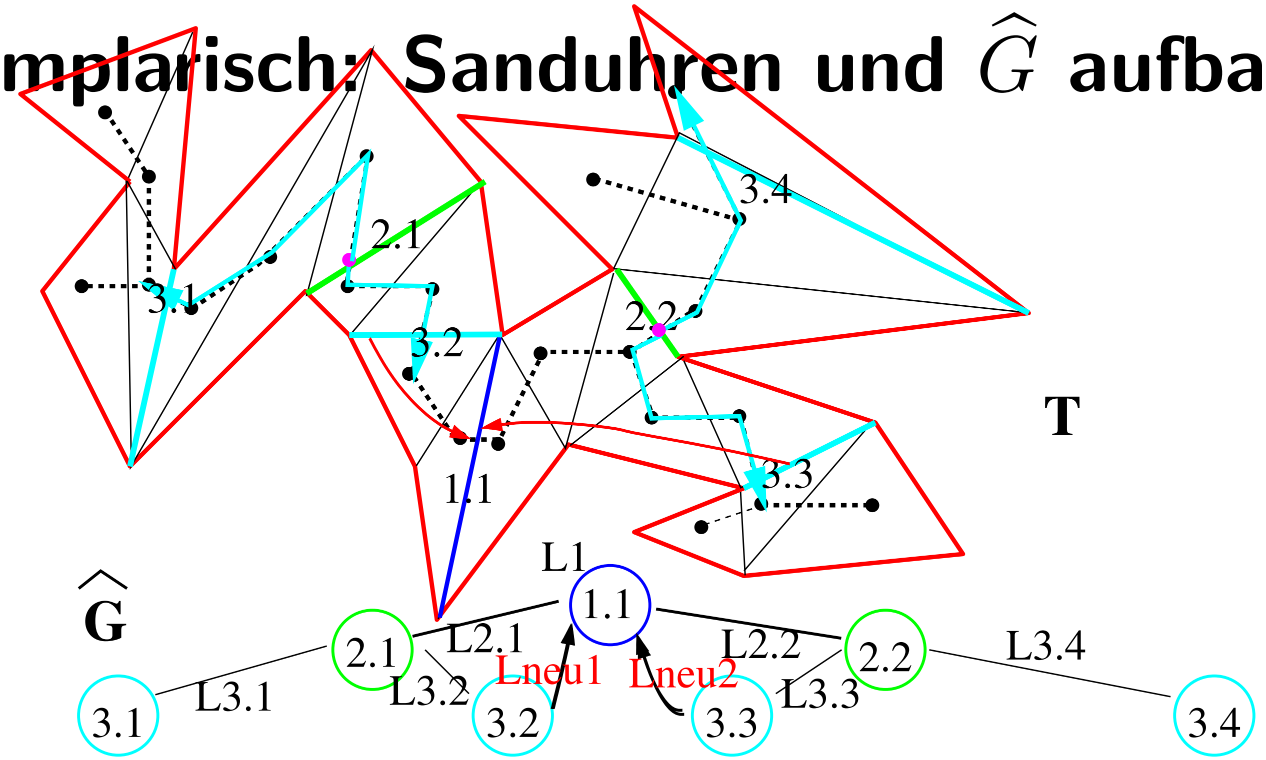


Durchlauf gemäß Cutting Theorem Durchlauf

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!

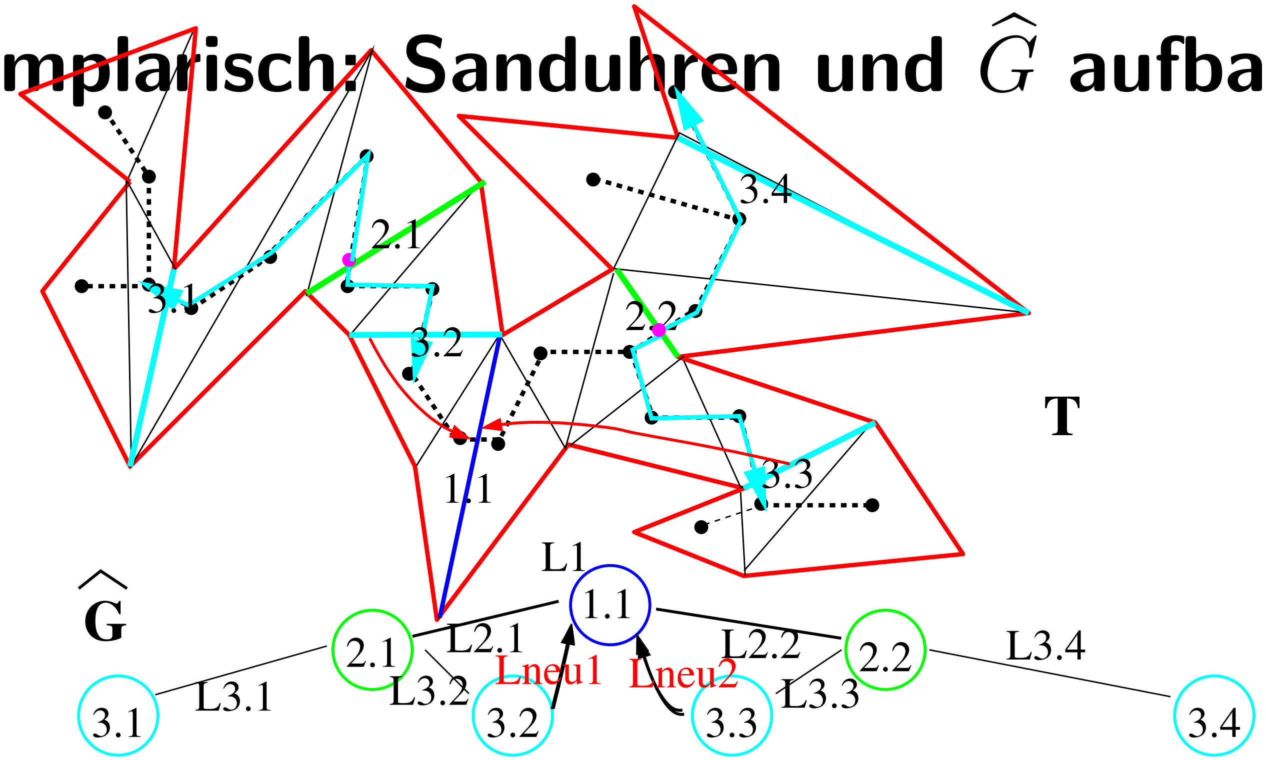


Durchlauf gemäß Cutting Theorem Durchlauf

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

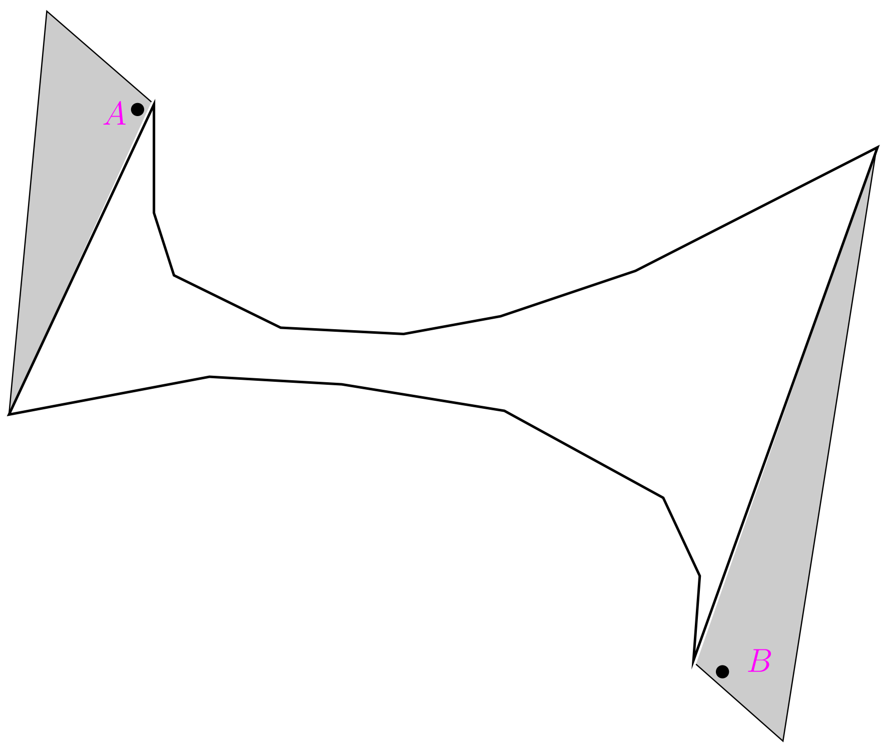
1. Berechne Triangulation T und Dual T^*: $O(n)$
2. Berechne hierarch. bal. Baum $\widehat{T}$, Sch.-Graph $\widehat{G}$: $O(n)$
3. Komplexität $\widehat{G}$: $O(n)$
4. Berechne *alle* Sanduhren von $\widehat{G}$: $O(n)$
5. Navigation zw. Dreiecken in $\widehat{G}$: Sequenz v. Diagonalen: $O(\log n)$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

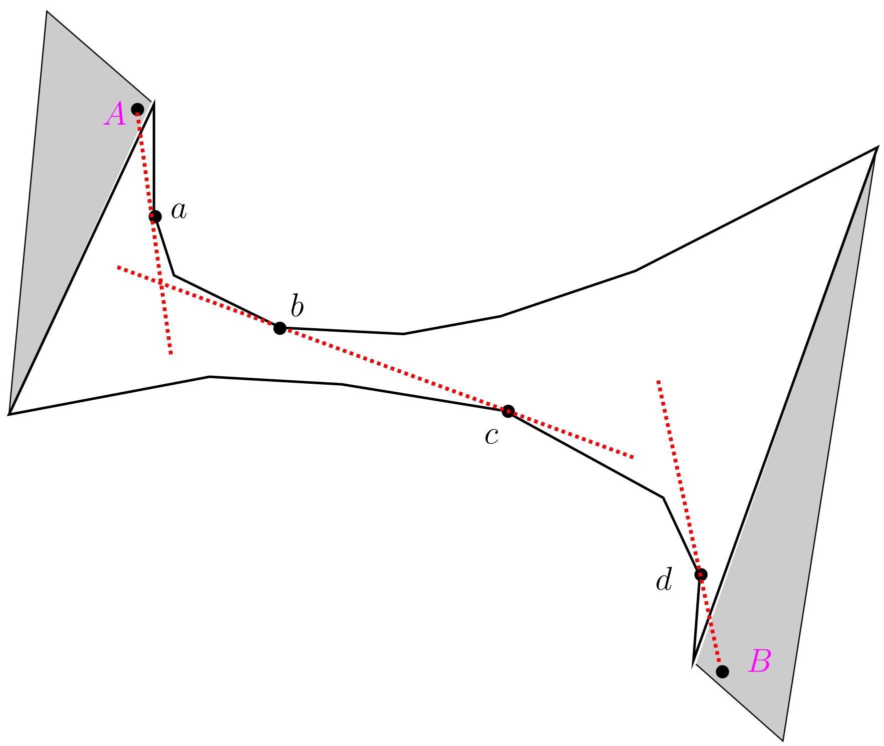
Berechne Shortest Path aus finaler Sanduhr

- Finale Sanduhr, Ziel und Start



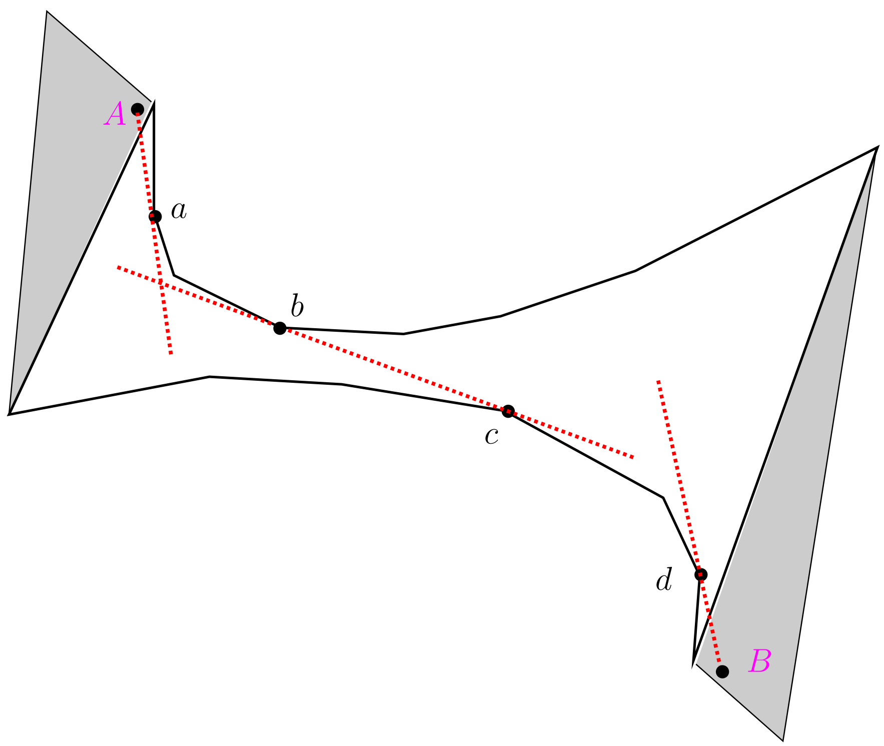
Berechne Shortest Path aus finaler Sanduhr

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



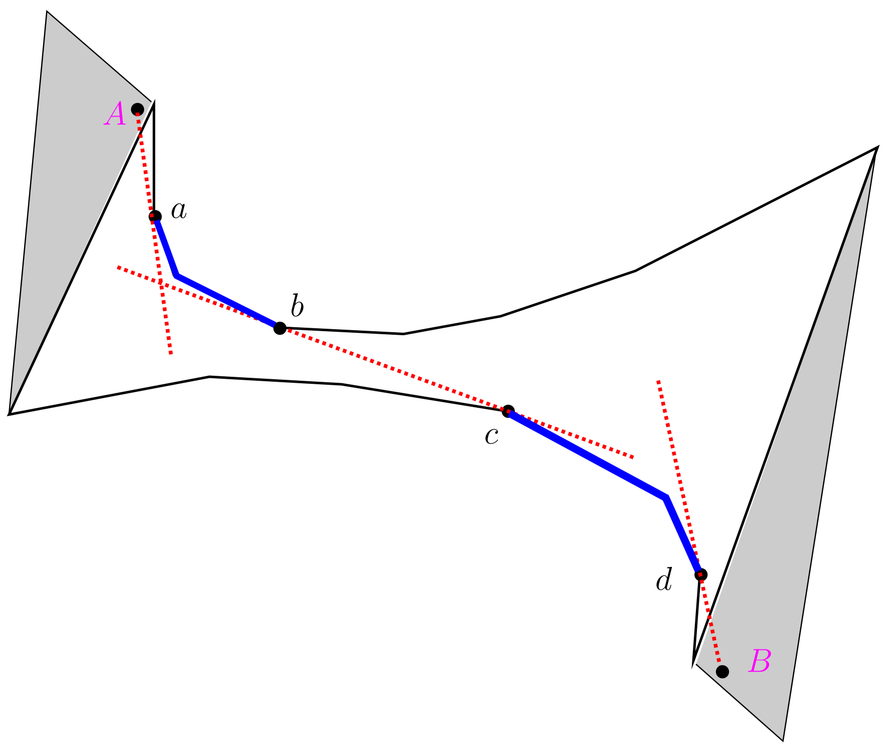
Berechne Shortest Path aus finaler Sanduhr

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

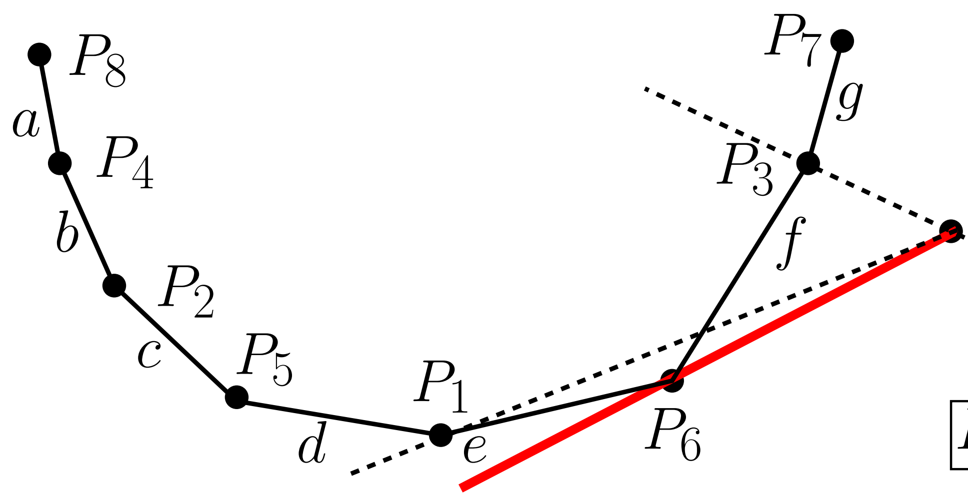


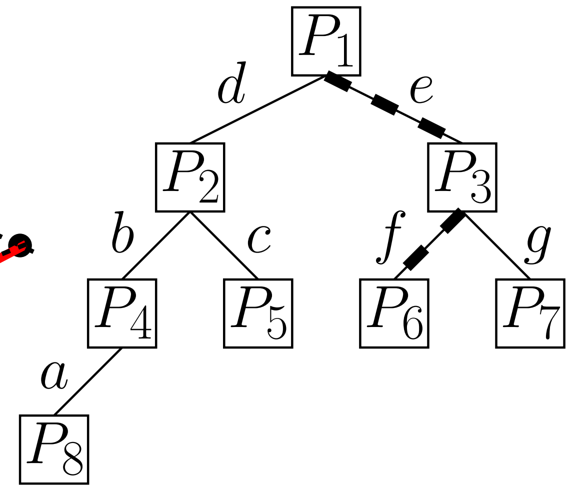
Berechne Shortest Path aus finaler Sanduhr

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



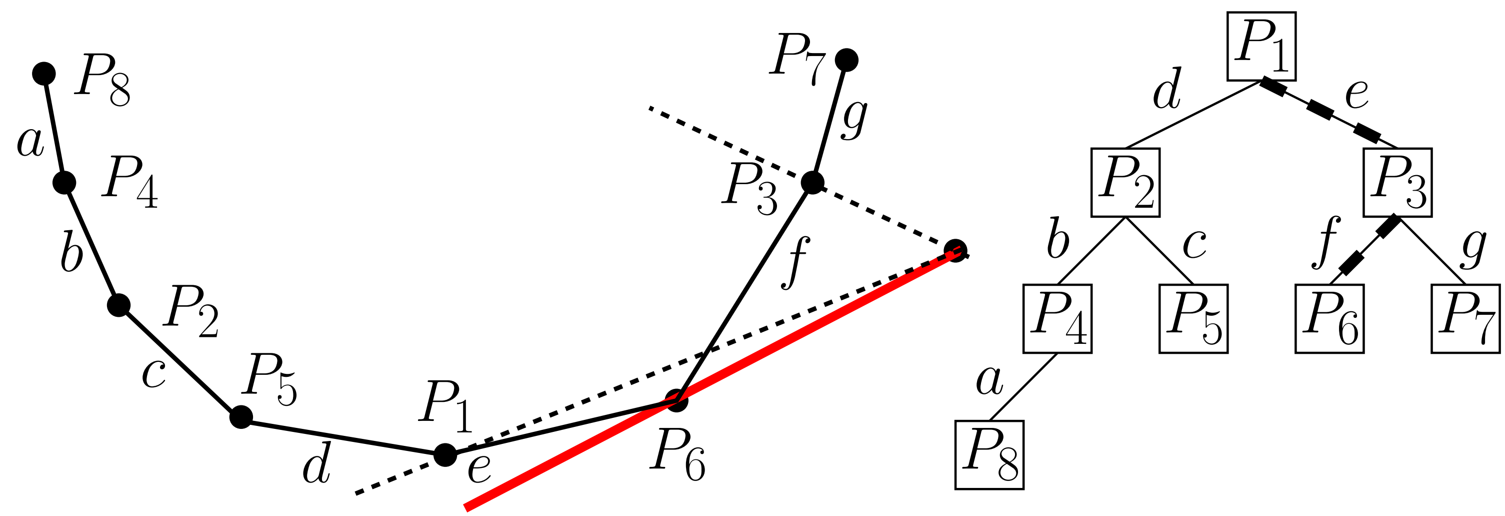
Datenstruktur Hourglass





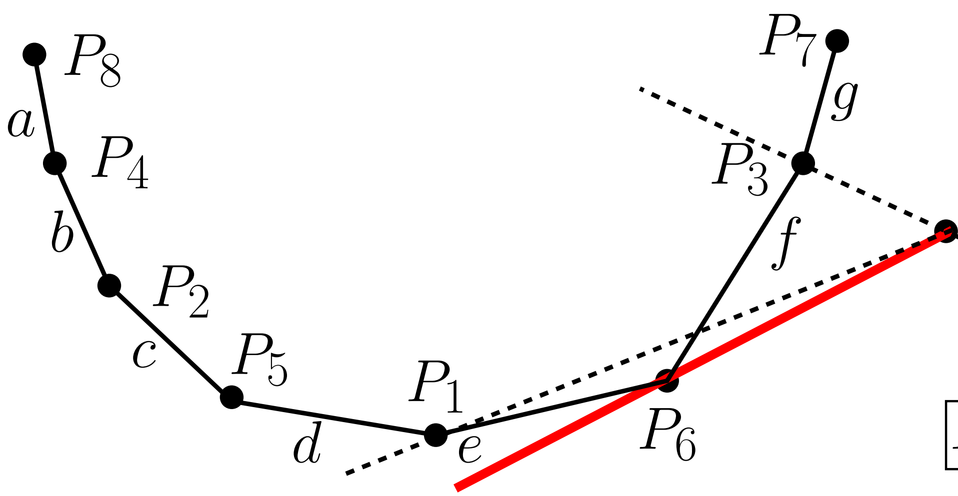
Datenstruktur Hourglass

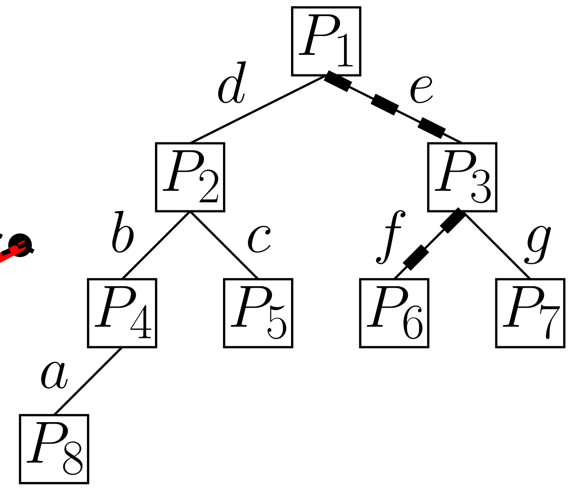
- Ketten der Sanduhren in bal. Baum speichern



Datenstruktur Hourglass

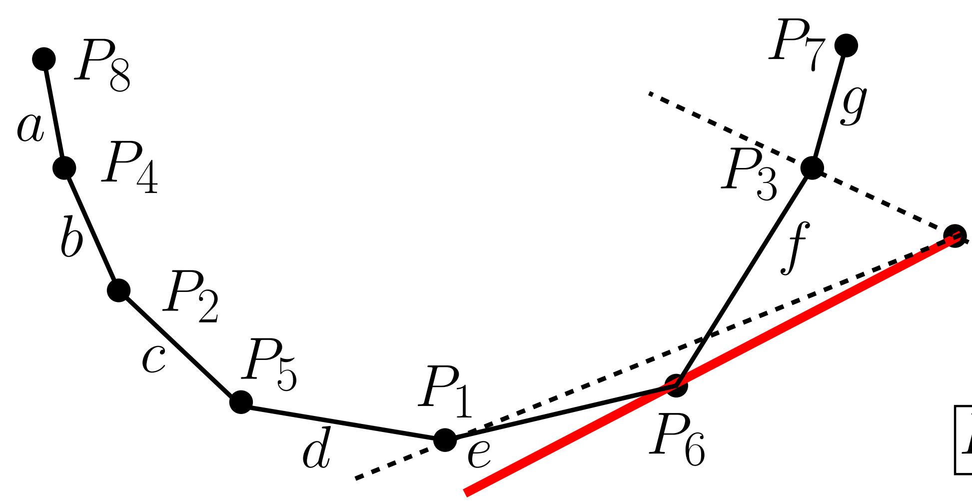
- Ketten der Sanduhren in bal. Baum speichern
- Tangente in logarithm. Zeit berechnen

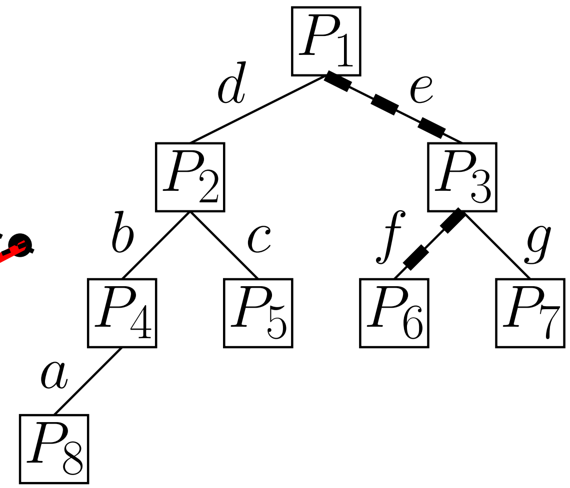




Datenstruktur Hourglass

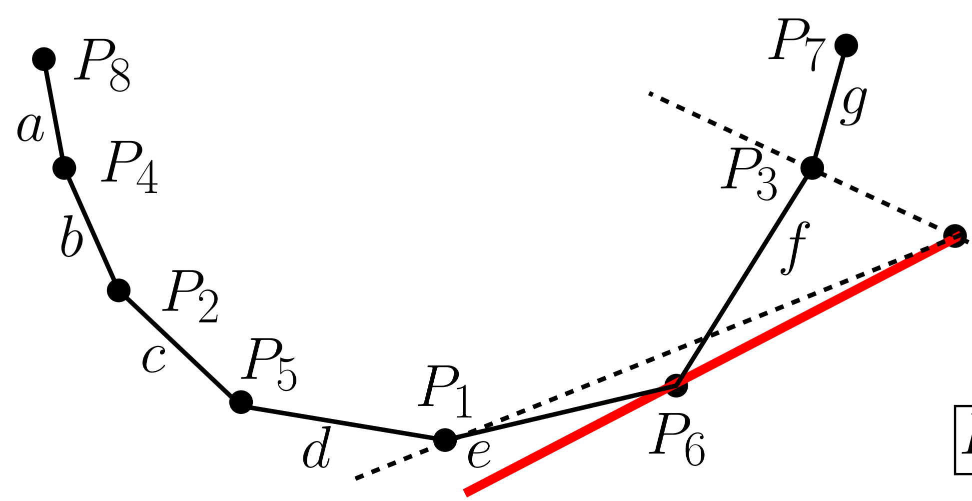
- Ketten der Sanduhren in bal. Baum speichern
- Tangente in logarithm. Zeit berechnen
- Länge in $O(1)$,

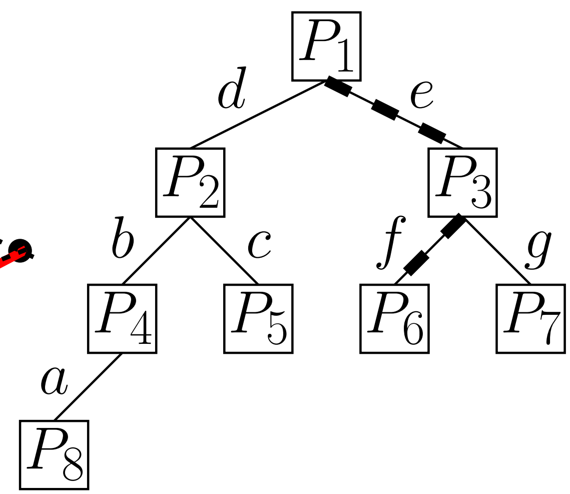




Datenstruktur Hourglass

- 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
- $\sum_{i=1}^{\mathtt{height(d_i)}} i$:

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
- $\sum_{i=1}^{\mathtt{height(d_i)}} i$: $O(\mathtt{height(d_i)}^2)$

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
- $\sum_{i=1}^{\mathtt{height(d_i)}} i$: $O(\mathtt{height(d_i)}^2)$
- 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

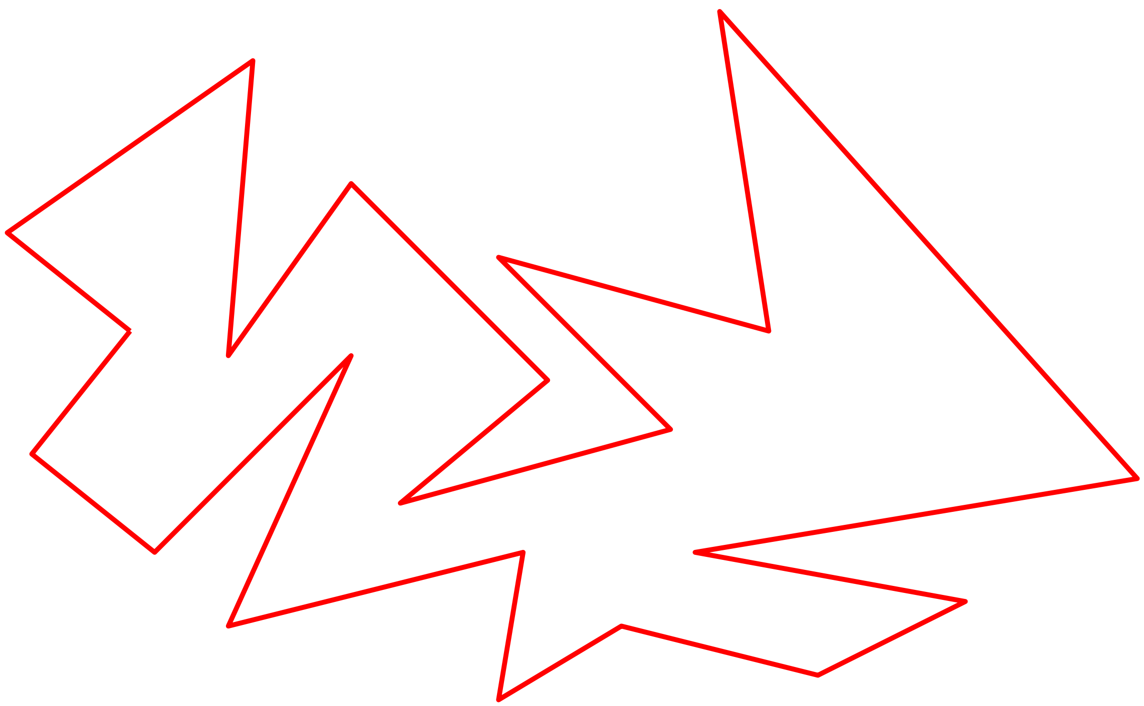
Guibas/Hershberger: Laufzeiten

- 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

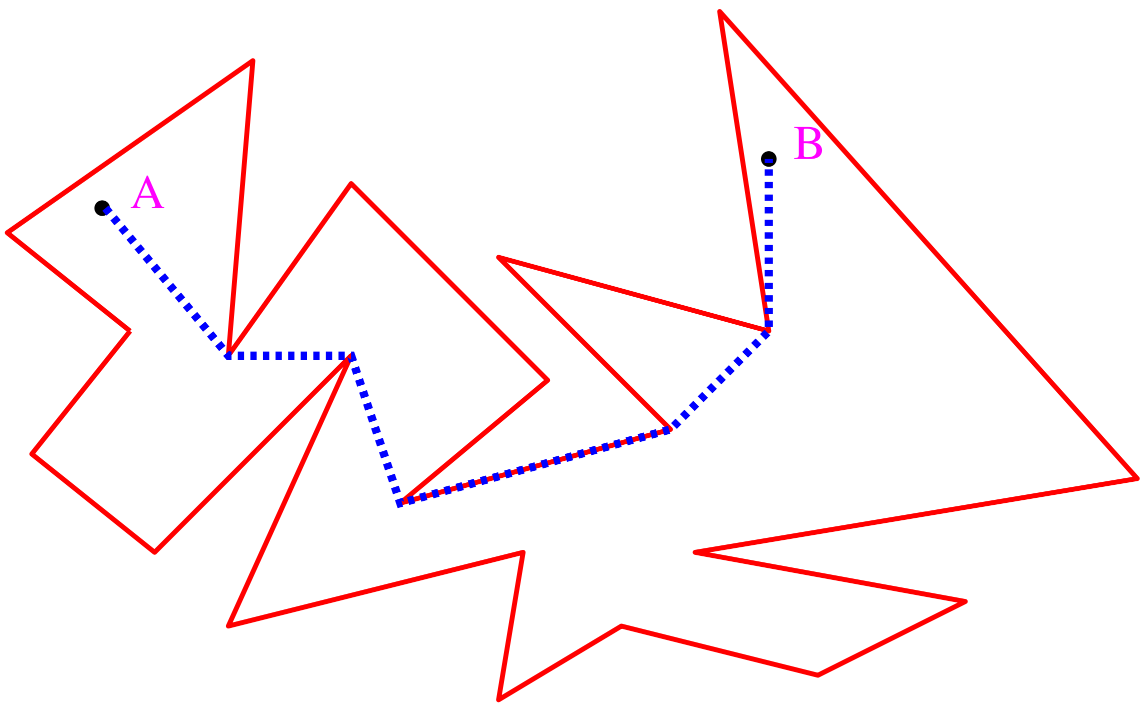
1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon P



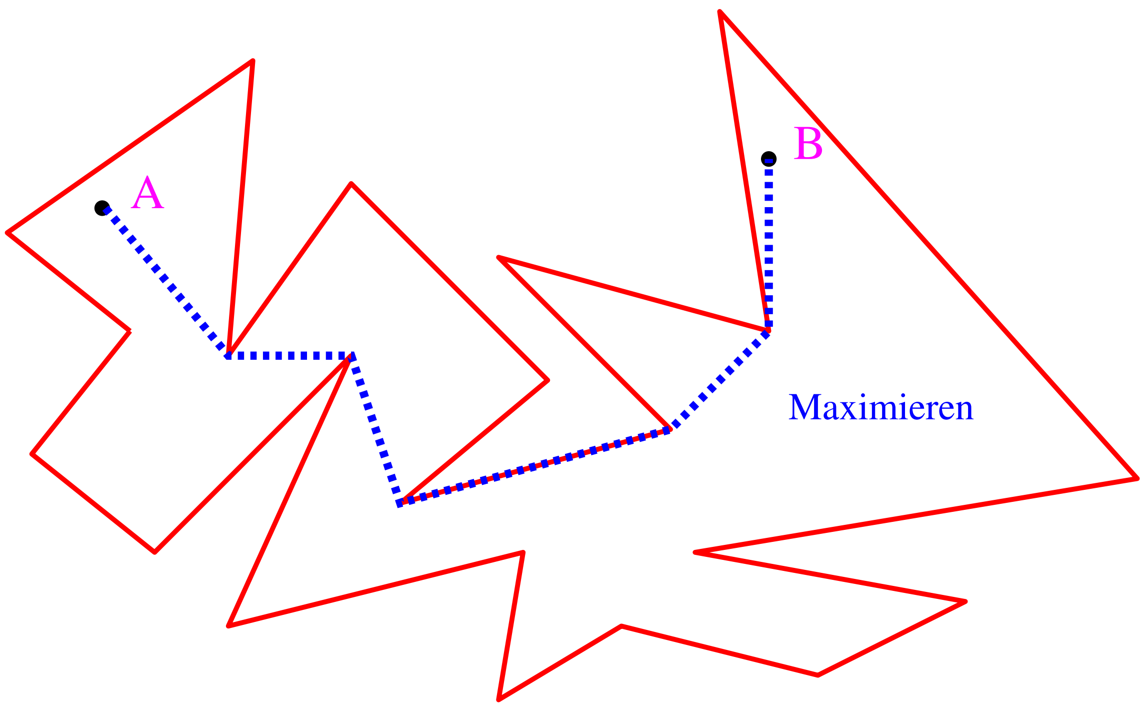
1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon P
- Längster Kürzester Weg zwischen zwei Punkten



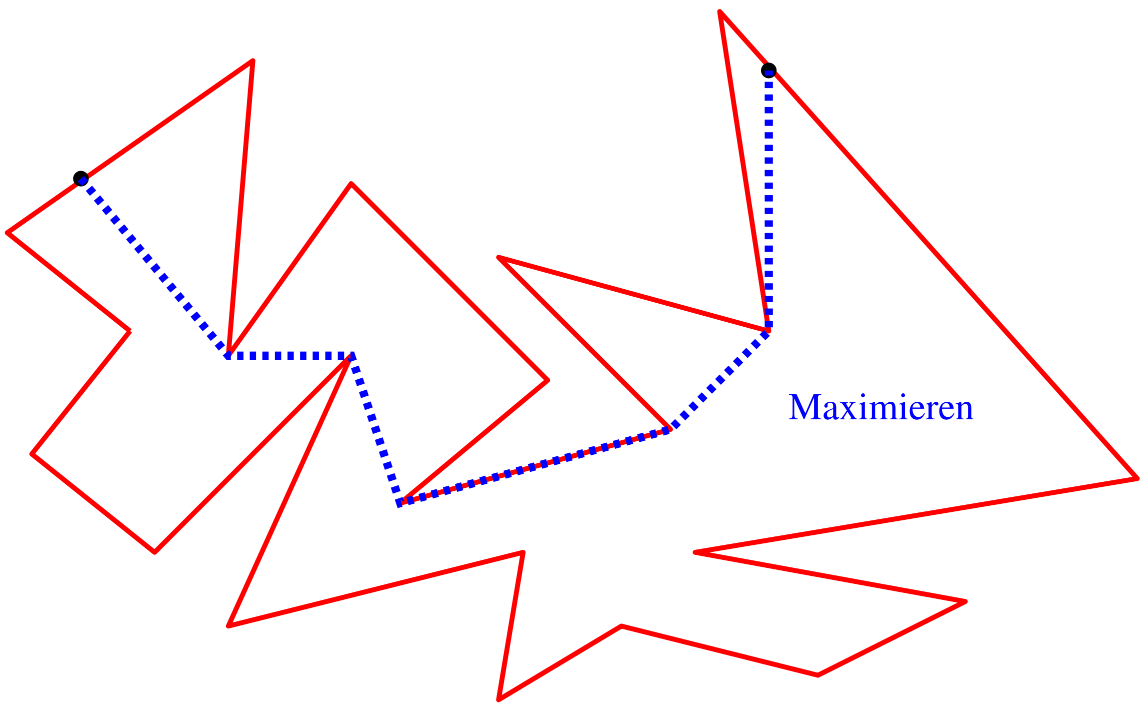
1.2.3 Durchmesser einfacher Polygone

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



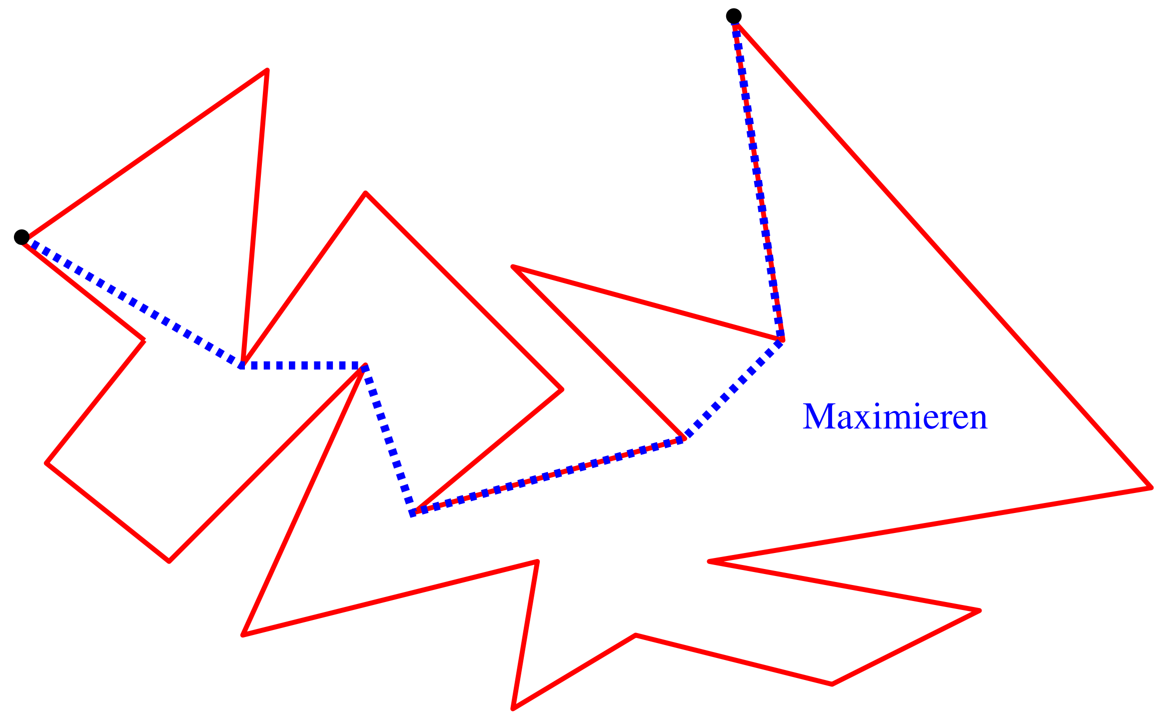
1.2.3 Durchmesser einfacher Polygone

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



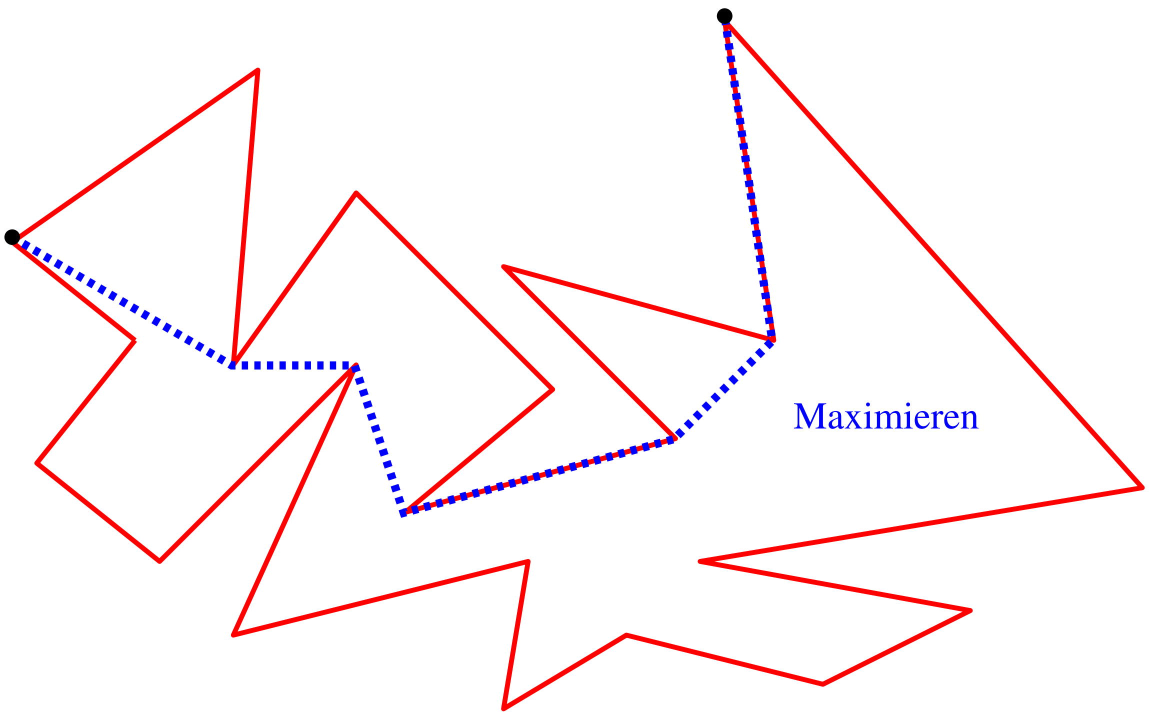
1.2.3 Durchmesser einfacher Polygone

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



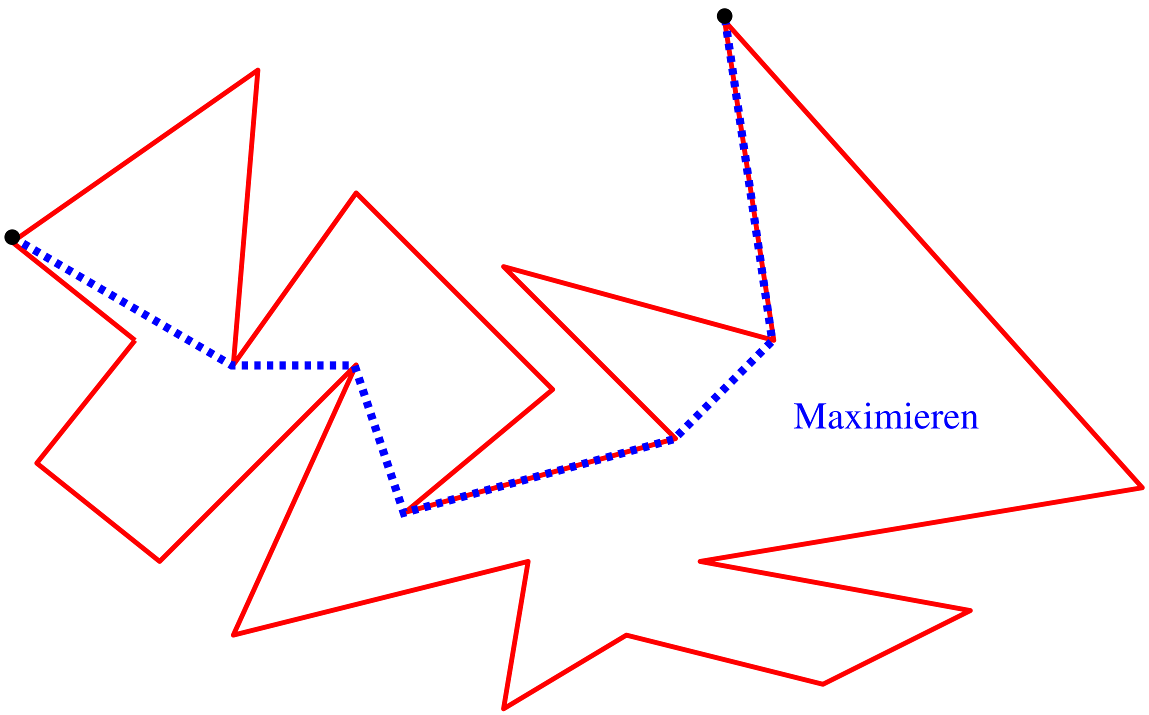
1.2.3 Durchmesser einfacher Polygone

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



1.2.3 Durchmesser einfacher Polygone

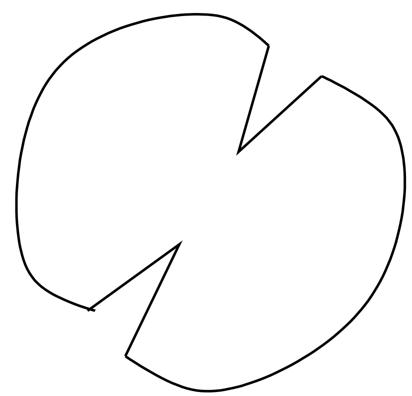
- Einfaches Polygon P
- Längster Kürzester Weg zwischen zwei Punkten
- Endpunkte sind Ecken des Polygones n^2 Kandidatenpaare
- Formal: $\max_{p_i,p_j \text{ Ecken von } P} d(p_i,p_j)$



Idee der Berechnung:

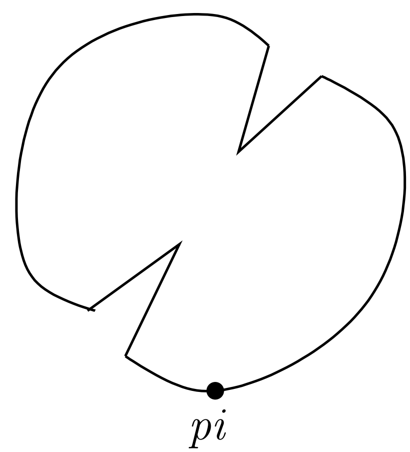
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



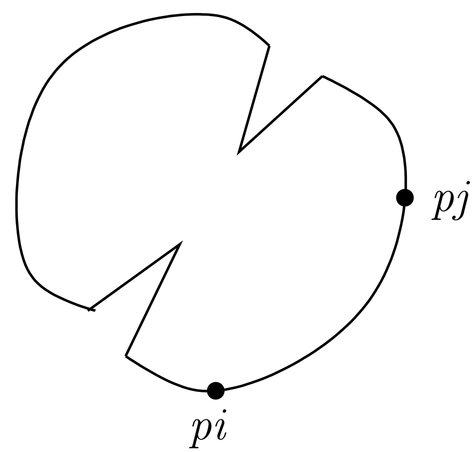
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



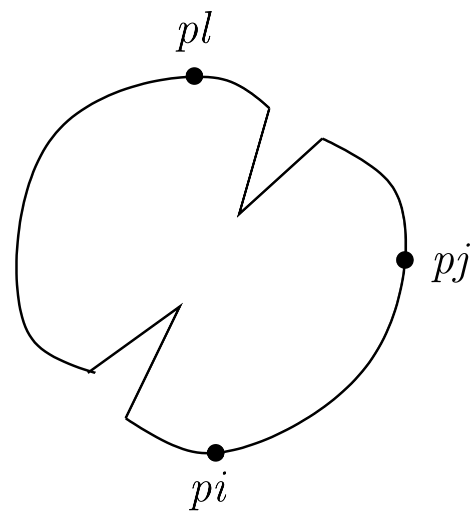
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



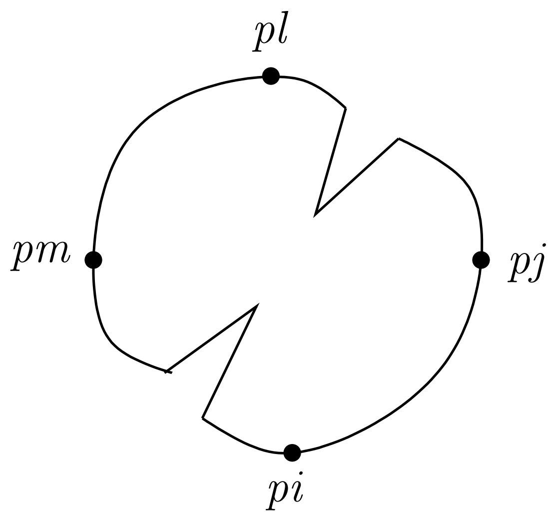
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



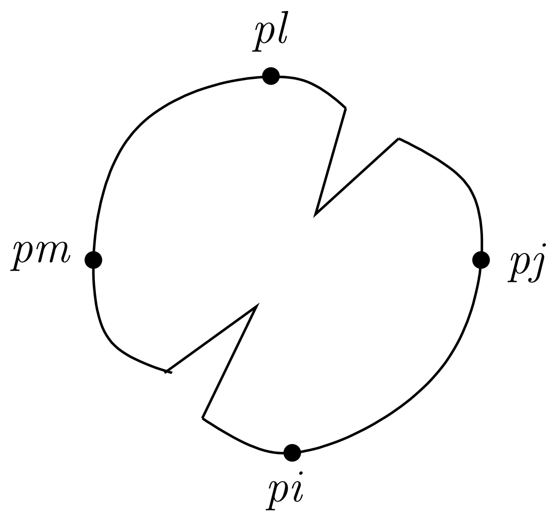
Idee der Berechnung:

- p_i, p_j, p_l, p_m entlang des Randes



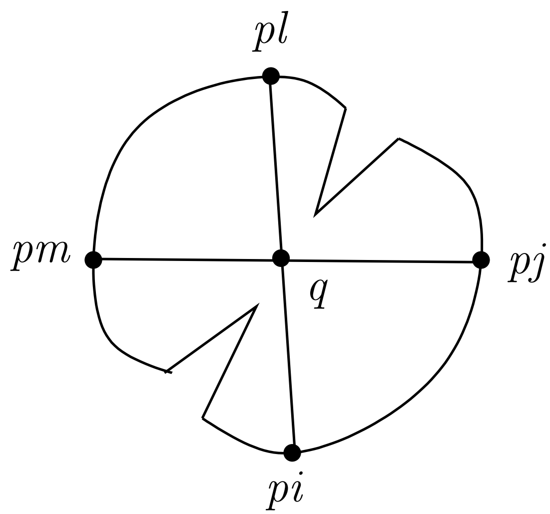
Idee der Berechnung:

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



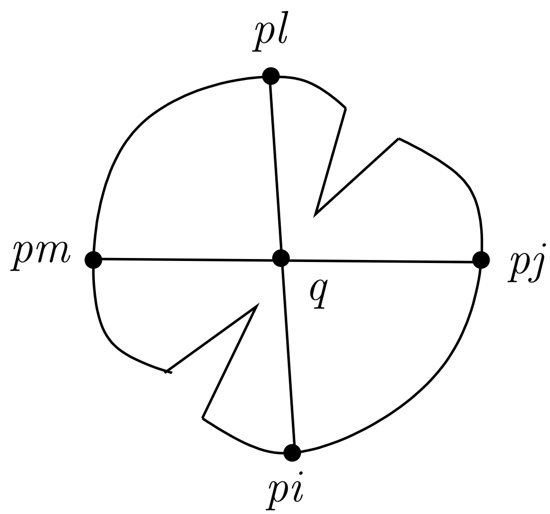
Idee der Berechnung:

- 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



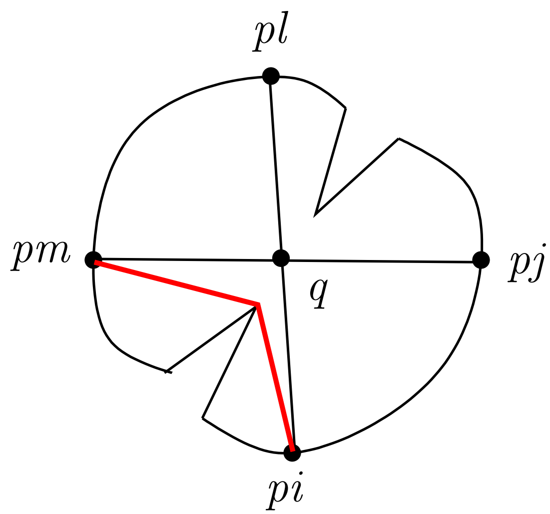
Idee der Berechnung:

- 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!!



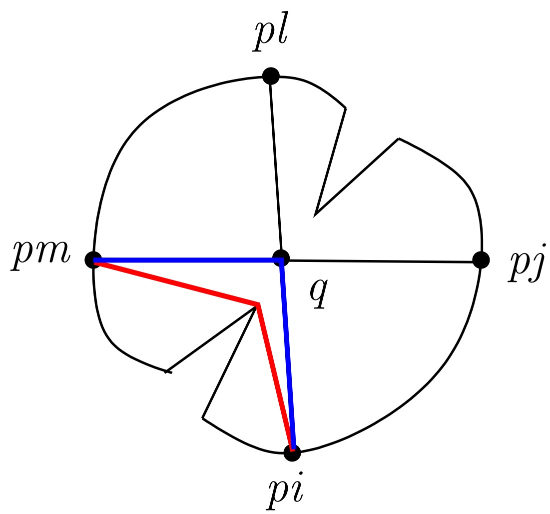
Idee der Berechnung:

- 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!!



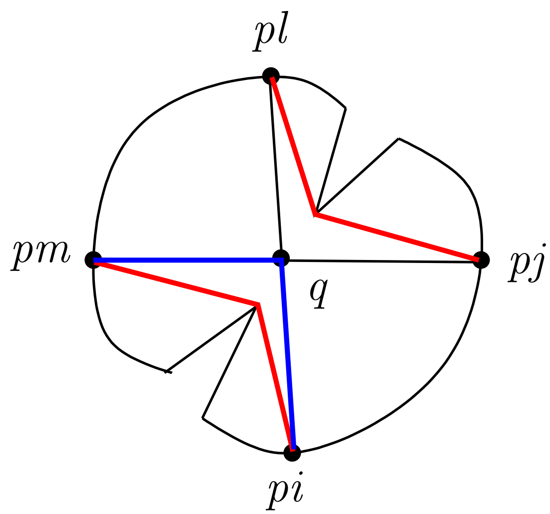
Idee der Berechnung:

- 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!!



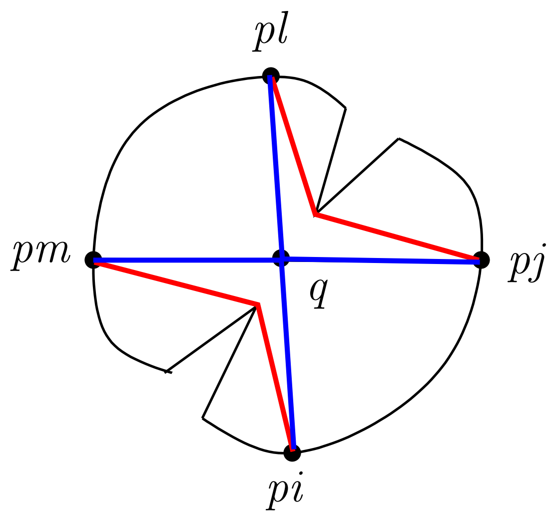
Idee der Berechnung:

- 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!

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

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

Def. 1.17: Monotone Matrix!

Def. 1.17: Monotone Matrix!

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

Def. 1.17: Monotone Matrix!

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

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

Def. 1.17: Monotone Matrix!

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

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

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

Linkeste Zeilenmaxima weiter nach rechts

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!

Lemma 1.19: A ist monoton!

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

Lemma 1.19: A ist monoton!

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

Beweis!!!

Theorem 1.20: Maxima für monotone Matrizen

Theorem 1.20: Maxima für monotone Matrizen

- Monotone $n \times m$-Matrix $A = (a_{i\ell})$, $m \geq n$

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'

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)

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

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

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

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

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

Alg. 1.7: Spaltenreduktion

Invariante:

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

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:

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 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

Alg. 1.7: Spaltenreduktion

Invariante:

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

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 ja, dann:

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 ja, dann:

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

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:

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 ja, dann:

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

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

Alg. 1.7: Spaltenreduktion

Letzte Zeile erreicht!

Alg. 1.7: Spaltenreduktion

Letzte Zeile erreicht!

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

Alg. 1.7: Spaltenreduktion

Letzte Zeile erreicht!

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

Komplette Spalte $n+1$ Streichen!!

Alg. 1.7: Spaltenreduktion

Letzte Zeile erreicht!

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

Komplette Spalte $n+1$ Streichen!!

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

Alg. 1.7: Spaltenreduktion

Letzte Zeile erreicht!

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

Komplette Spalte $n+1$ Streichen!!

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

Beispiel!!!

Alg. 1.7: Spaltenreduktion

Alg. 1.7: Spaltenreduktion

- Input: monotone $n \times m$ Matrix A, $m \geq n$

Alg. 1.7: Spaltenreduktion

- Input: monotone $n \times m$ Matrix A, $m \geq n$
- Output: monotone $n \times n$ Matrix A'

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

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:

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

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. 1.7: 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