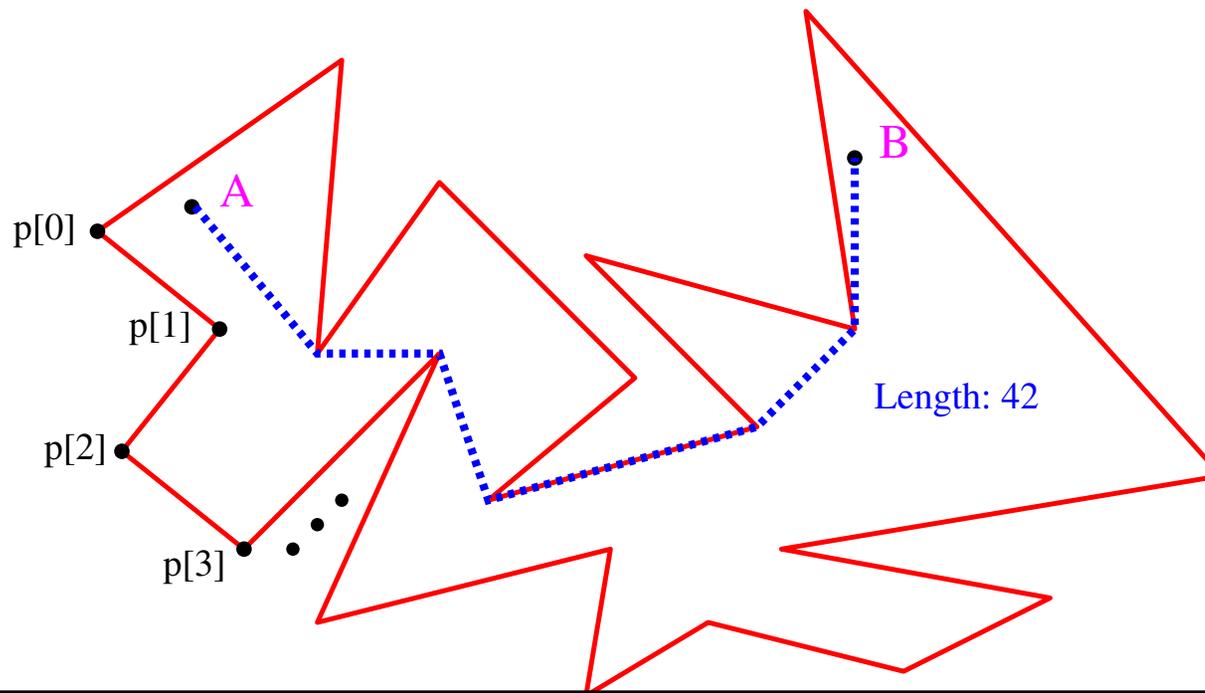


Offline Bewegungsplanung: Innerhalb von Polygonen

Elmar Langetepe
University of Bonn

Shortest Path in einfachen Polygonen

- Einfaches Polygon P , DS: Seq. Knoten/Kanten, CCW-order
- Start A und Ziel B
- Berechne kürzesten Weg von A nach B innerhalb P
- Polygonale Kette mit Knoten aus P /Länge des Pfades
- Algorithmus: $\Omega(n)$, $|P| = n$



Def. 1.10 Triangulation einfaches Polygon P

Diagonale

- Segment mit Knoten aus P
- Verläuft innerhalb P
- Schneidet den Rand von P nicht

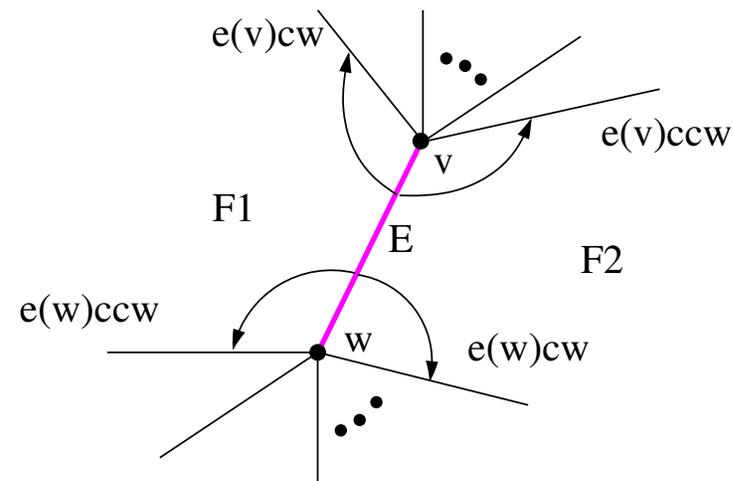
Triangulation von P

- Graph: Kanten und Knoten von P
- Plus: maximale Menge sich nicht-schneidener Diagonalen

- Komplexität und Berechnung
- Datenstruktur DCEL
- DFS walk-through/navigation

Doubly connected edge list (DCEL)

- Planarer Graph
- für jede Kante
- Name/Link für Flächen, Knoten
- nächste Kante in CW/CCW Order von Knoten v/w
- Navigation



Dualer Graph eines Planaren Graphen

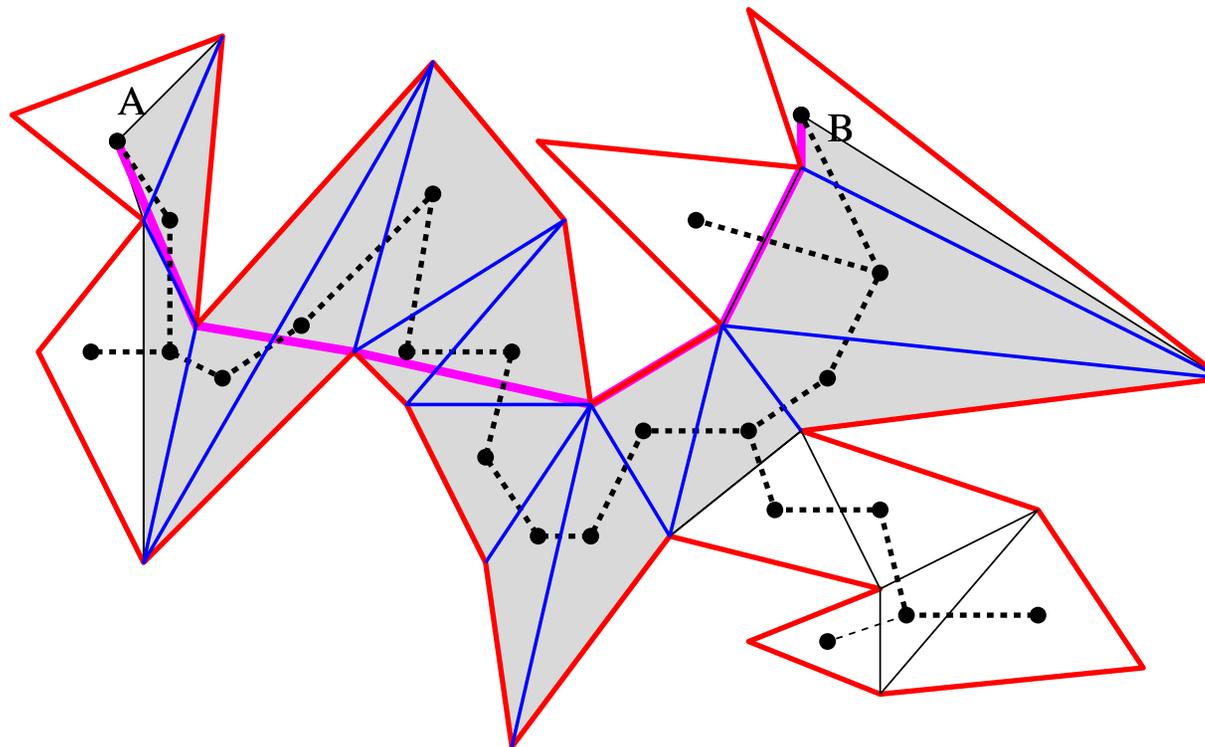
Dualer Graph $D(G)$ eines Planaren Graphen G



- ein Knoten für jede Fläche
- eine Kante zwischen angrenzenden Flächen einer Kante
- nicht-schneidend
- $D(D(G)) = G$
- Realisation

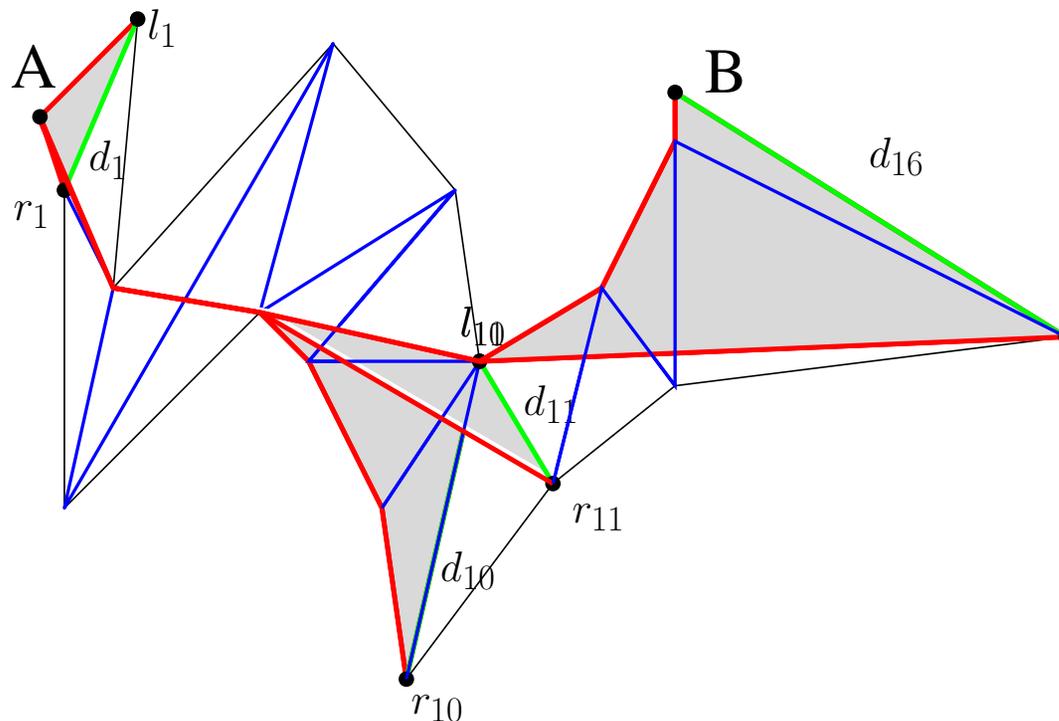
Einfache, effiziente Lösung (Lee/Preparata)

- **Triangulation** von P : Graph T , DS: **DCEL**, Adjacency list
- **Dualer Graph**: $D(T)$ (Tree), **Depth First Search** (DFS) on $D(T)$
- Subpolygon P' , Kette von Dreiecken/Diagonalen



Shortest Path in P'

- Kette von Dreiecken/Diagonalen (Data structure)■
- Induktiv: Shortest paths zu den Endpunkten der Diagonalen■
- Ind. Anf, ■ Ind. Schritt, ■ Letzte Diagonale \Rightarrow Ergebnis, ■ Laufzeit?■

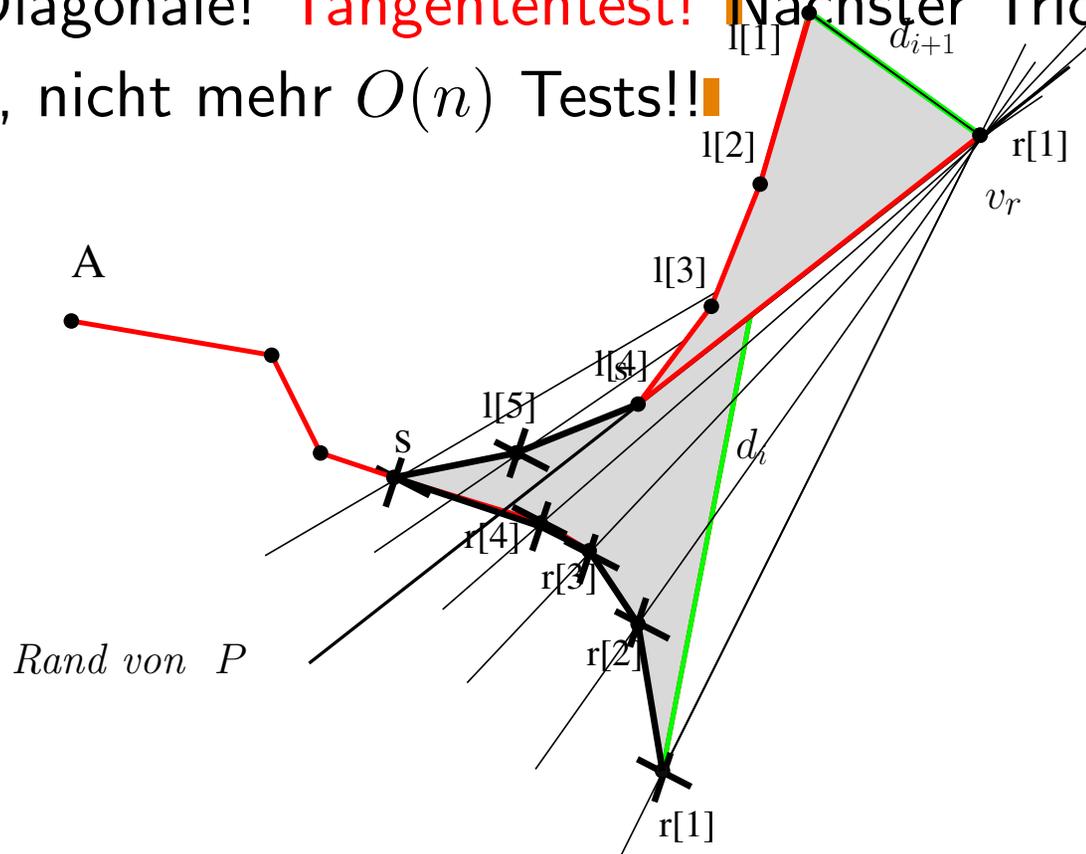


Zusammenfassung: Alg. 1.4

- Triangulation von P : $O(n)$ Zeit und Platz
- Dualer Graph der Triangulation: $O(n)$ Zeit und Platz
- DFS auf Dualen Graphen: $O(n)$ Zeit
- Berechne P' : $O(n)$ Zeit und Platz
- Berechne Shortest Path zu End-Diagonale, induktiv: $O(n)??$
- Shortest Path von A nach B innerhalb P in $O(n)$ Zeit und Platz!
optimal!!
- **Theorem 1.11**

Verbleibt: Induktiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path! ■
- Nächste Diagonale! **Tangententest!** ■ Nächster Trichter! ■
- Insgesamt, nicht mehr $O(n)$ Tests!! ■

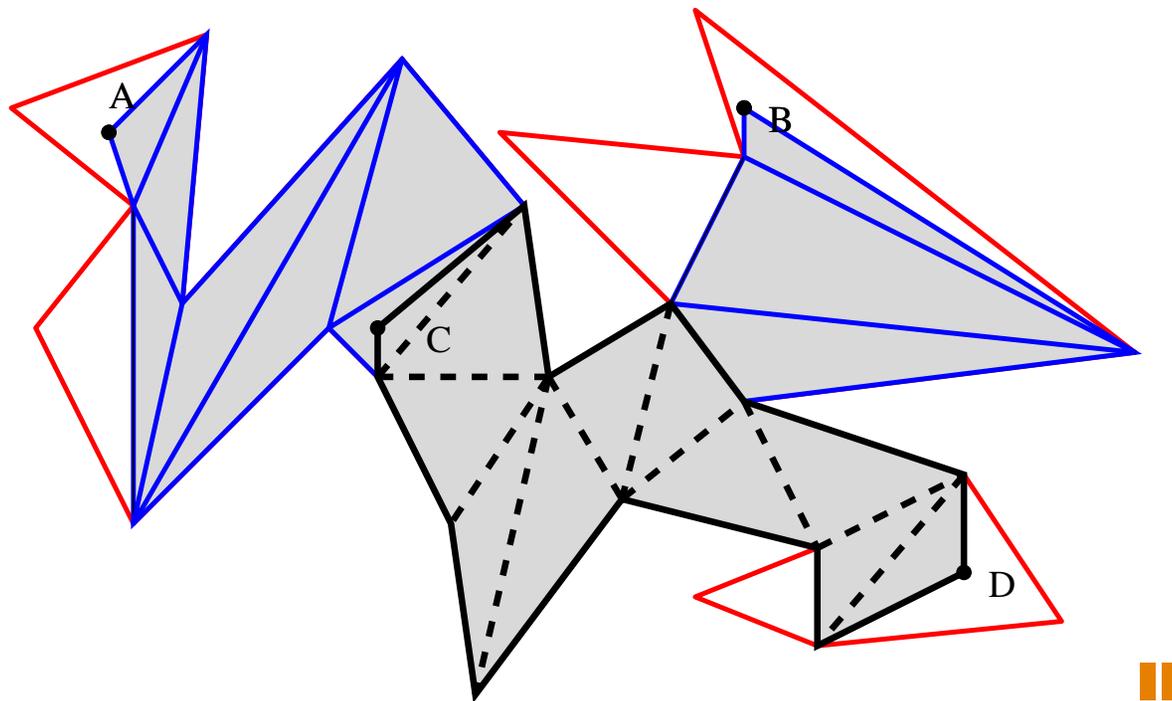


Zusammenfassung: Lee/Preparata: Shortest Path innerhalb P

- Triangulation von P : $O(n)$ Zeit und Platz
- Dualer Graph der Triangulation: $O(n)$ Zeit und Platz
- DFS im Dualen Graphen: $O(n)$ Zeit
- Berechnung P' : $O(n)$ Zeit und Platz
- Berechnung Shortest Path für End-Diagonale, induktiv: $O(n)$
- Shortest Path von A nach B in P in $O(n)$ Zeit und Platz
- Optimal für einzelne Anfrage!!
- **Theorem 1.11**

Mehrere Shortest Path Anfragen

- Viele Anfragen im Laufe der Zeit
- Häufig über gleiche Teil-Sequenz von Dreiecken
- Berechnungen ausnutzen



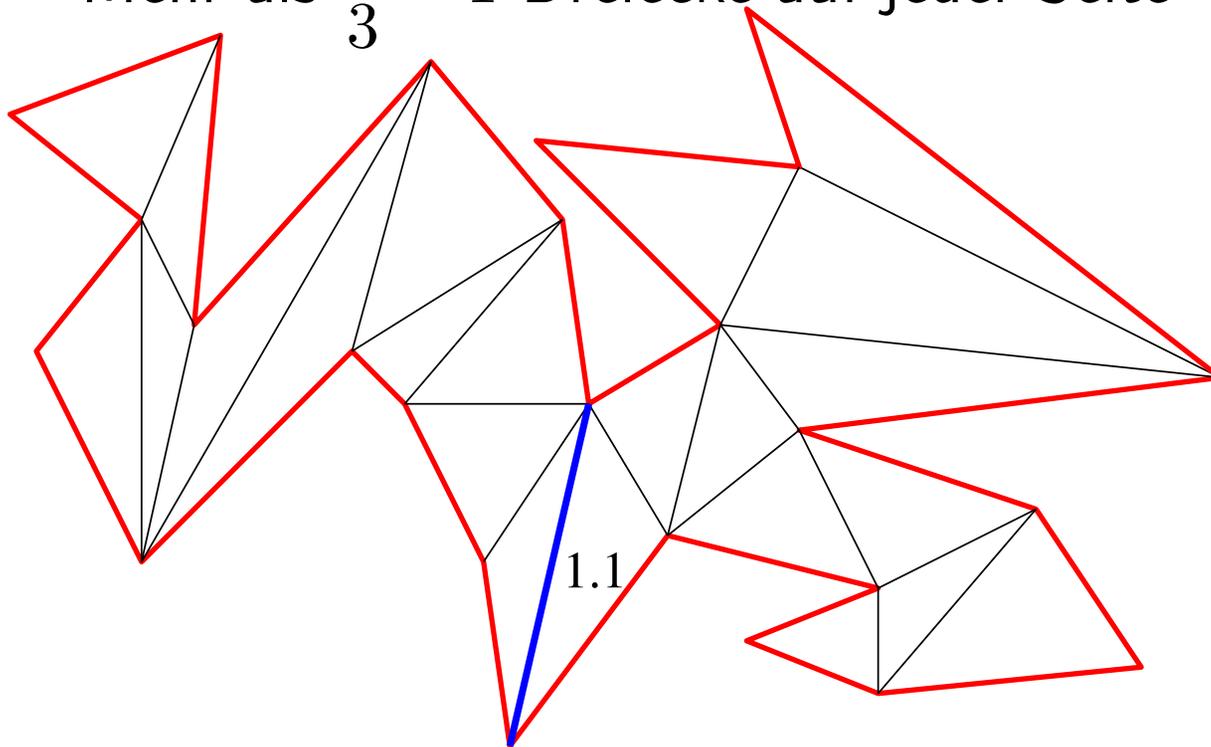
Mehrere Shortest Path Anfragen

- Preprocessing: Für Kürzeste-Wege-Anfragen vorbereiten
- Struktur des Polygons in DS ablegen
- Komplexität in $O(n)$, Aufbau in $O(n)$
- Anfrage in $O(\log n)$ bzw. $O(\log n + k)$
- Animation

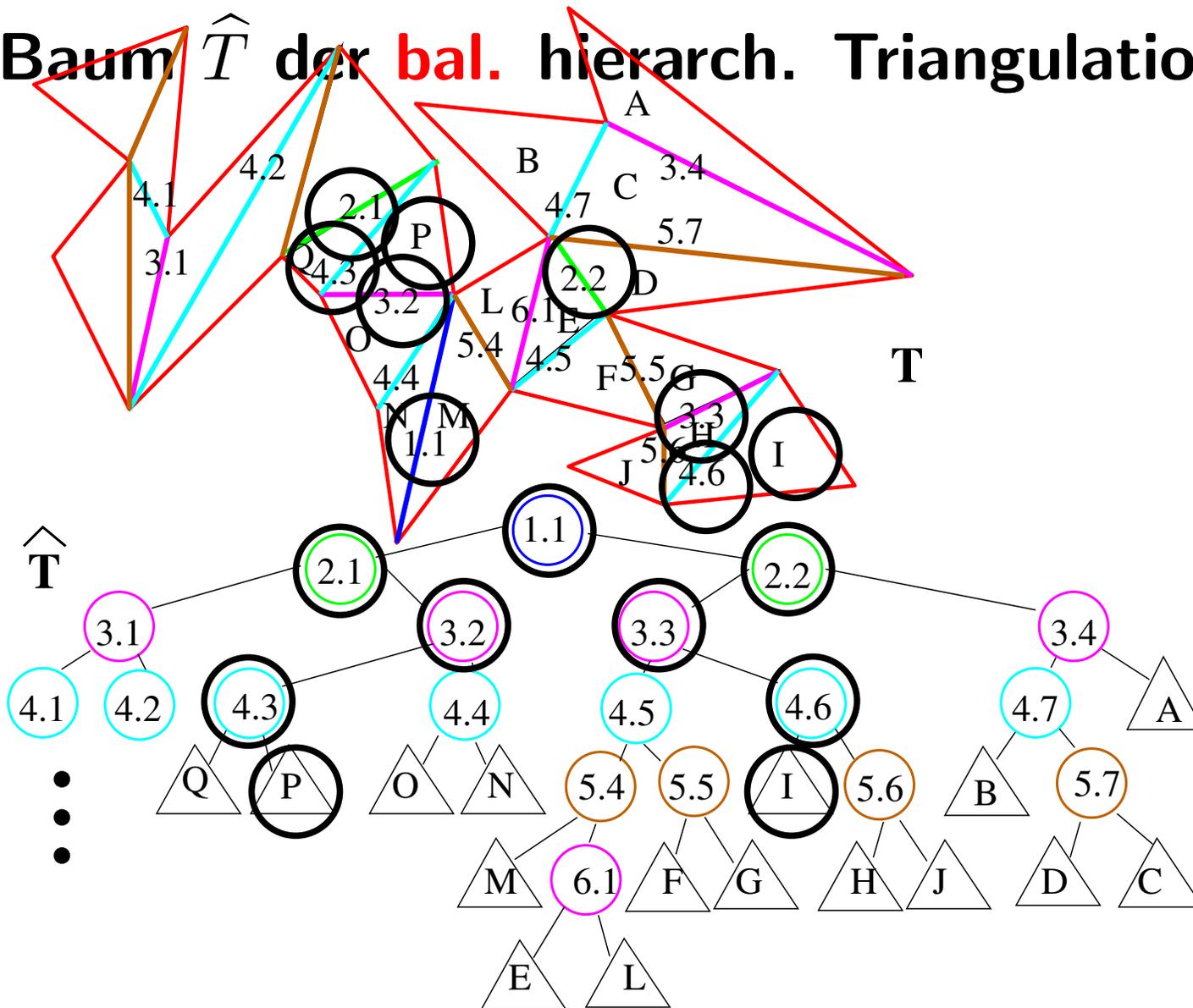
Preprocessing für $P!$ (Guibas/Hershberger)

Cutting theorem (Theorem 1.12) von Chazelle '82: In jeder Triangulation T existiert immer eine Diagonale für eine balancierte Einteilung von $T!$

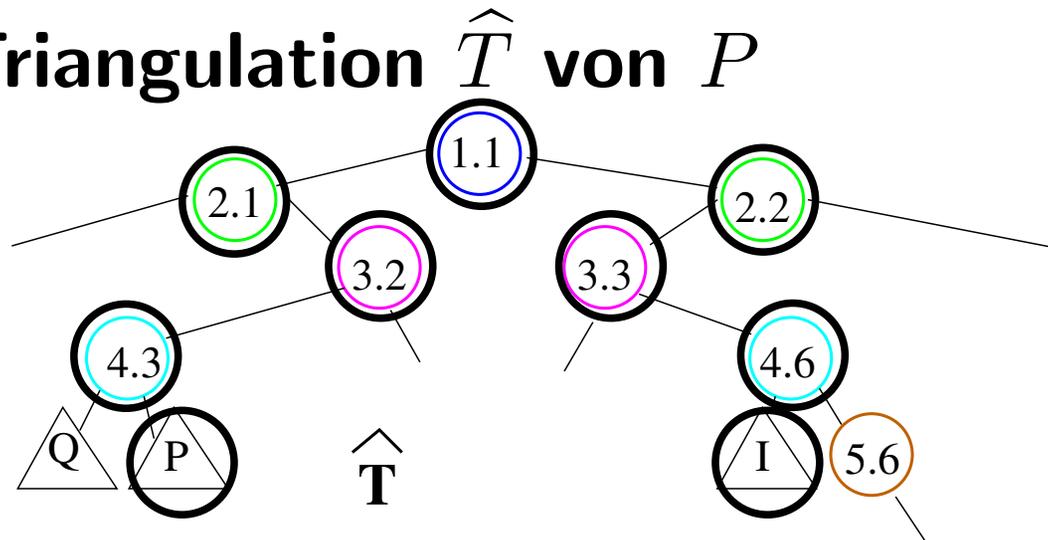
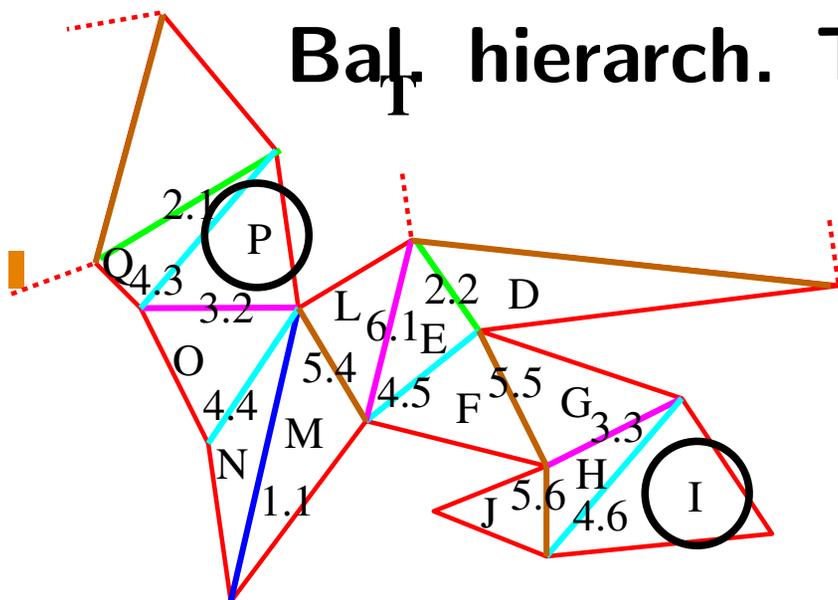
Mehr als $\frac{n}{3} - 1$ Dreiecke auf jeder Seite



Baum \hat{T} der bal. hierarch. Triangulation



BaT hierarch. Triangulation \hat{T} von P



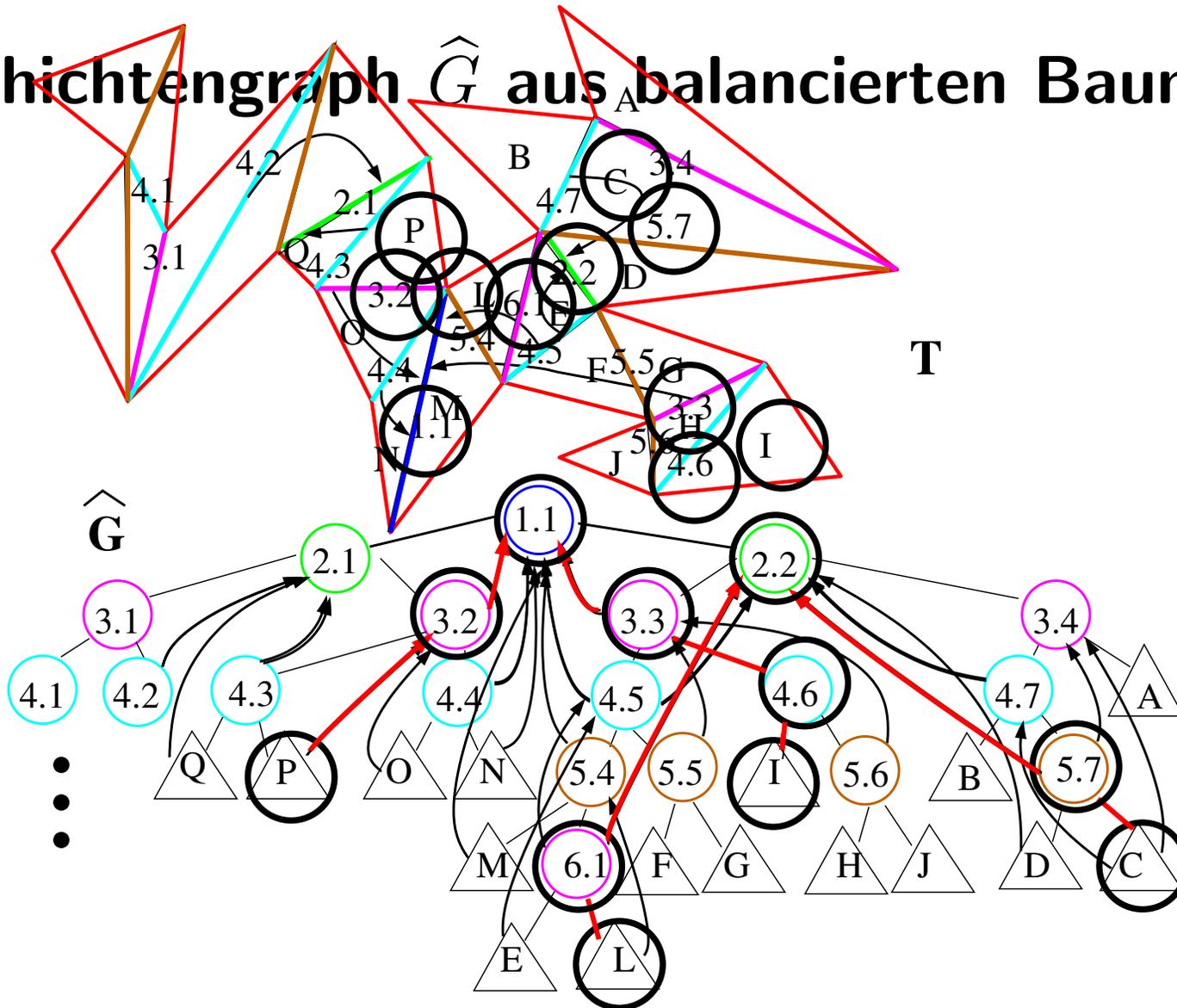
Kürzester Weg zwischen Dreiecken in T ($O(n)$) oder \hat{T} ($O(\log n)$)

T :	P	(3.2)	(4.4)	(1.1)	(5.4)	(6.1)	(4.5)	(5.5)	(3.3)	(4.6)	
\hat{T} :	P	(4.3)	(3.2)	(2.1)	(1.1)	(2.2)			(3.3)	(4.6)	

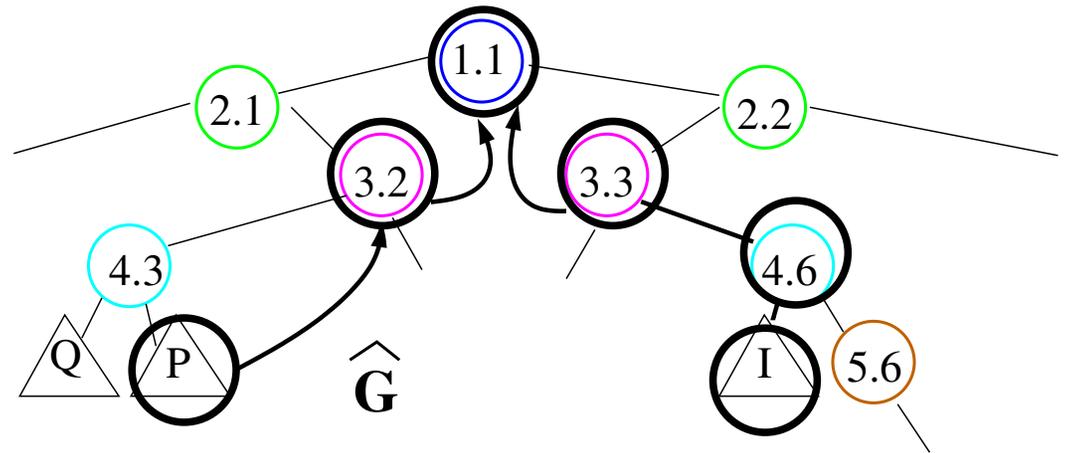
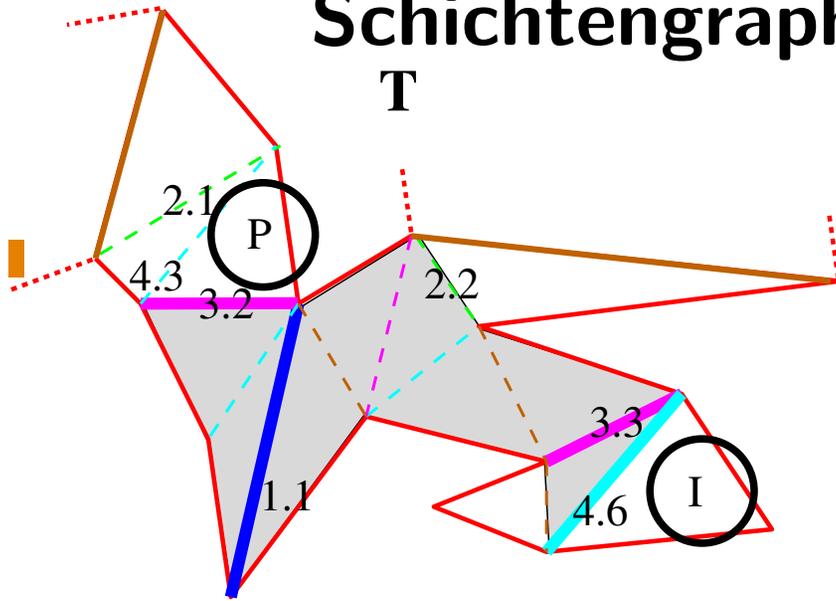
I) Finde richtige Sub-Sequenz in \hat{T} : P, (3.2), (1.1), (3, 3), (4, 6), I

II) Fülle die *Löcher*: (3.2) \Rightarrow (1.1) und (1.1) \Rightarrow (3, 3)

Schichtengraph \hat{G} aus A balancierten Baum \hat{T}



Schichtengraph \hat{G} aus bal. Baum \hat{T}



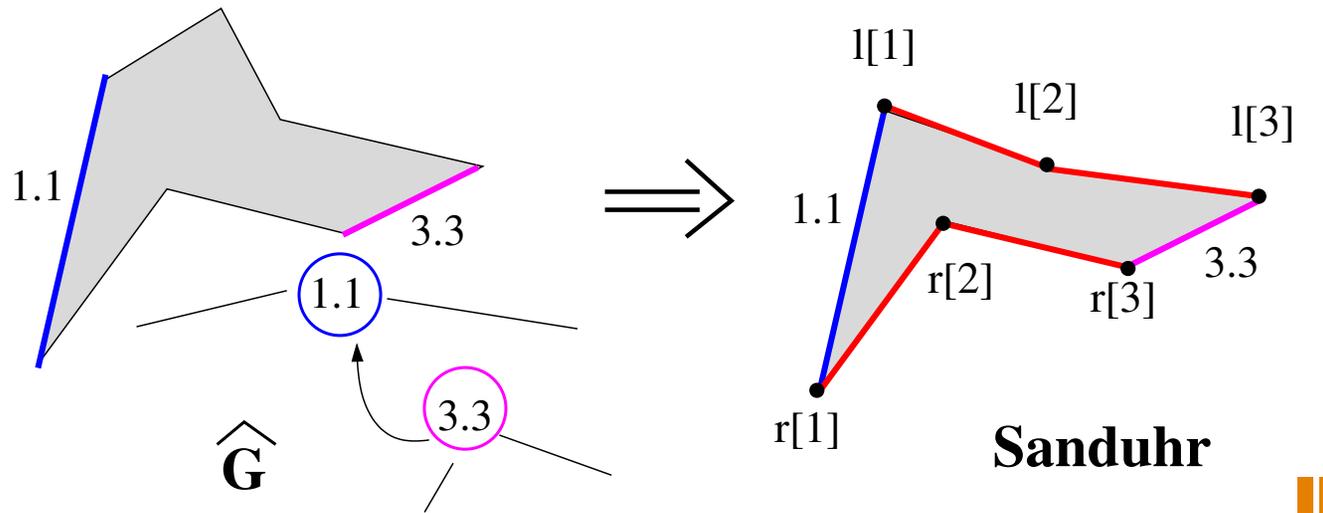
\hat{G} löst Problem I): Teil-Sequenz der Diagonalen: $O(\log n)$

Füllen der Löcher: Z.B.. zwischen (3.3) und (1.1) Kanten in \hat{G} !

Preprocessing: Kanten führen zu *Sanduhr* von Diagonalen

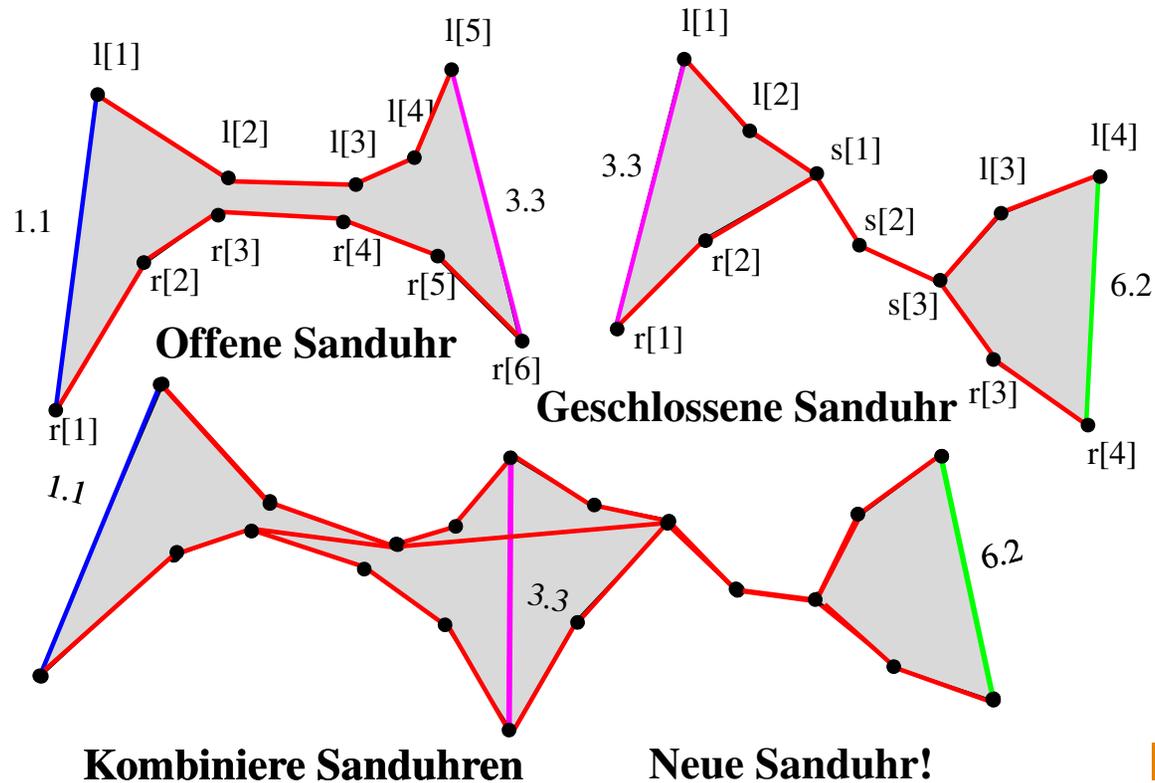
Sanduhren zwischen Diagonalen in P

- Kante in \hat{G}
- Shortest Path zwischen Endpunkte der Diagonalen: DS



DS Sanduhr im Allgemeinen

- Offene Sanduhr zwischen Diagonalen, DS
- Geschlossene Sanduhr zwischen Diagonalen, DS
- Konkatenation von mehreren Sanduhren, finale Sanduhr



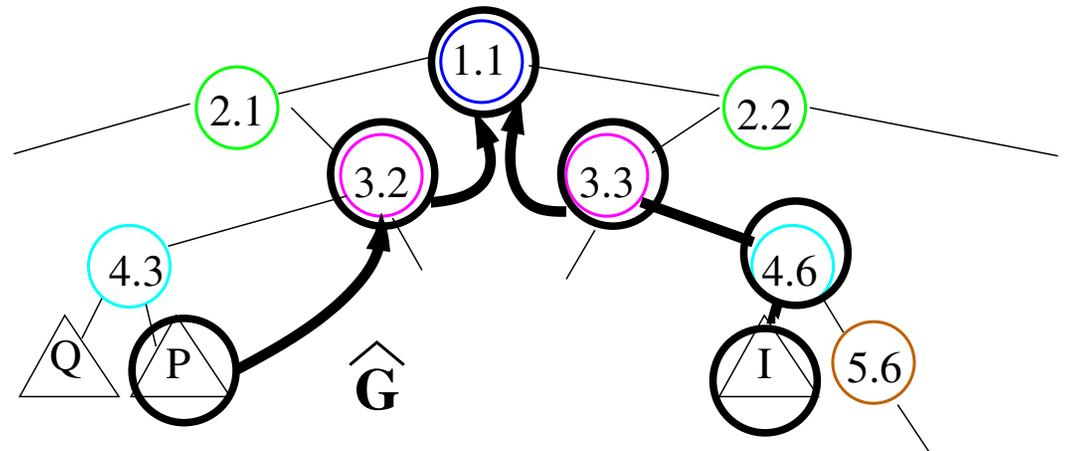
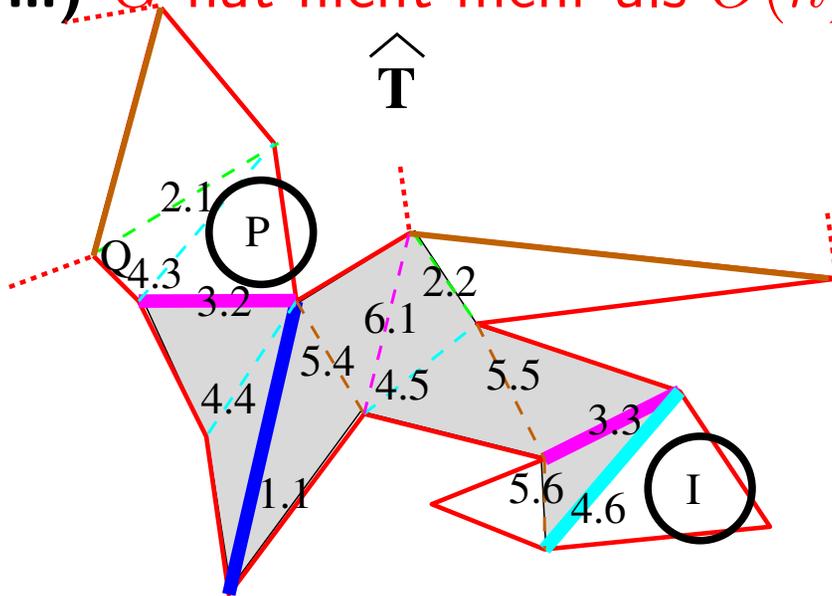
Zusammenfassung des Problems/Analyse

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)!!

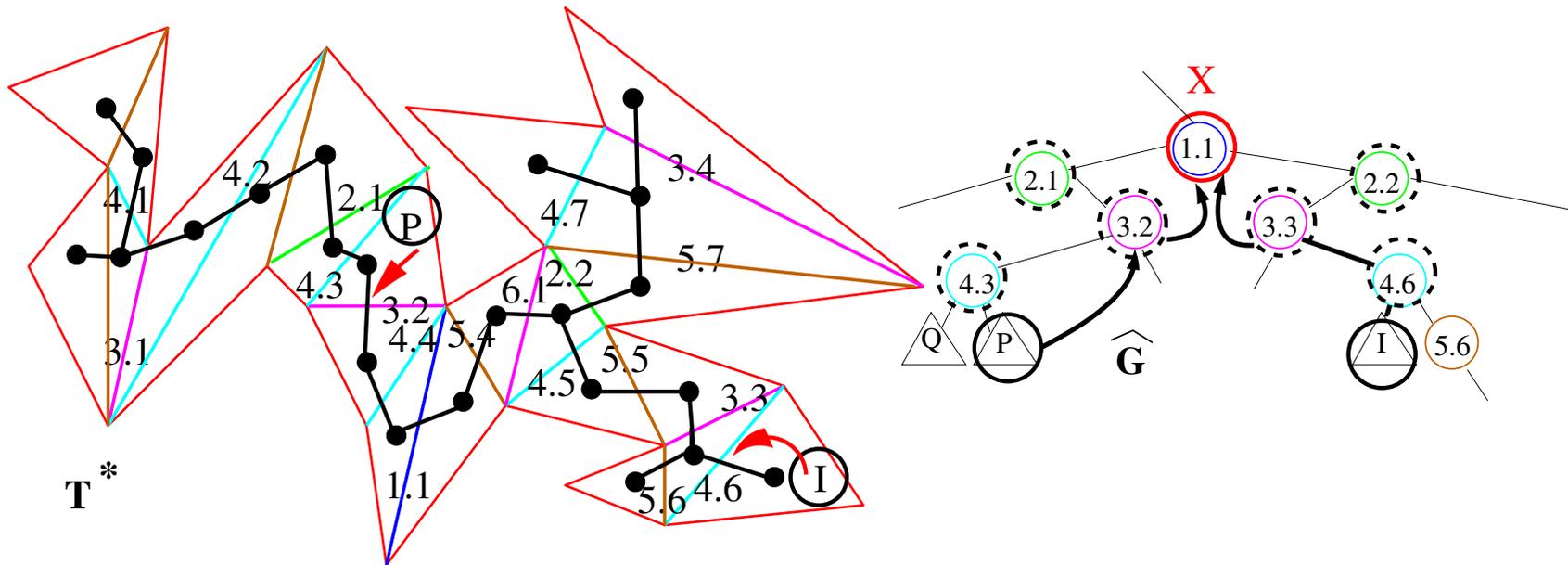
Eigenschaften von \hat{G} : Lemma 1.13

- 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!



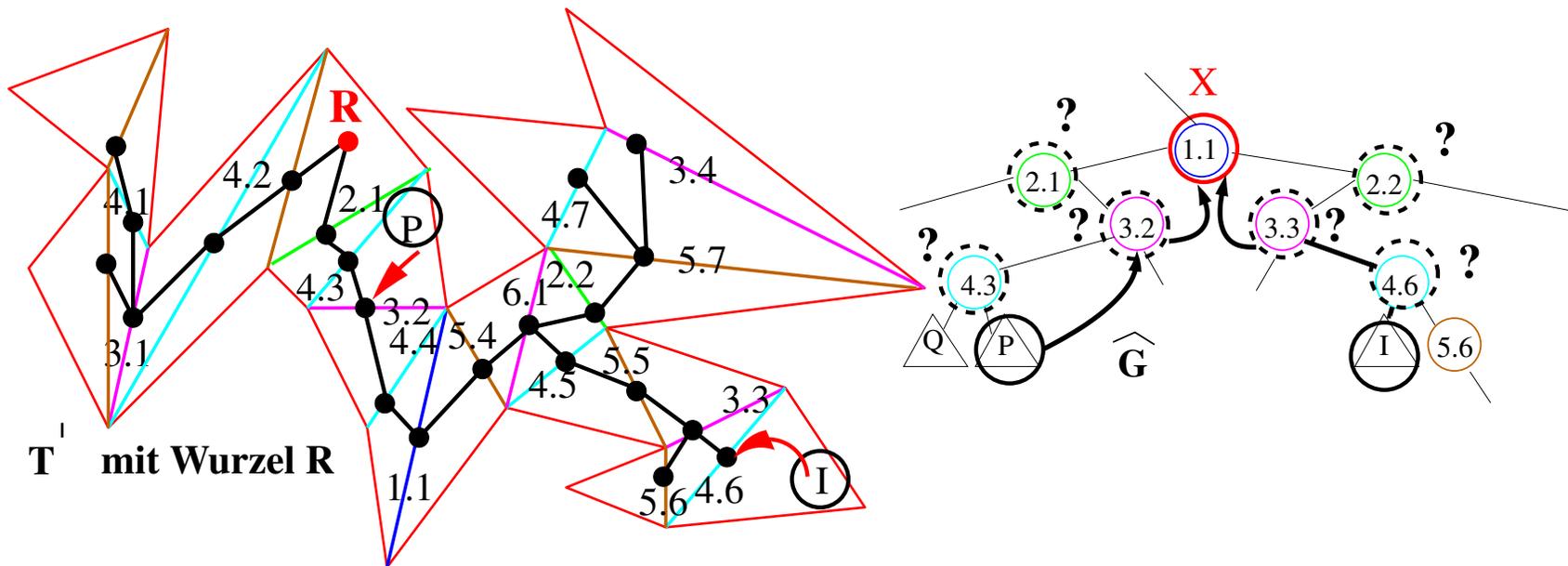
i) Weg in \hat{G} existiert

- Weg im Dualen Graphen T^* , Weg in \hat{T}
- Abgleichen mit Tiefenrelation
- Benachbart in der Konstruktion (Kante in \hat{G})!



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

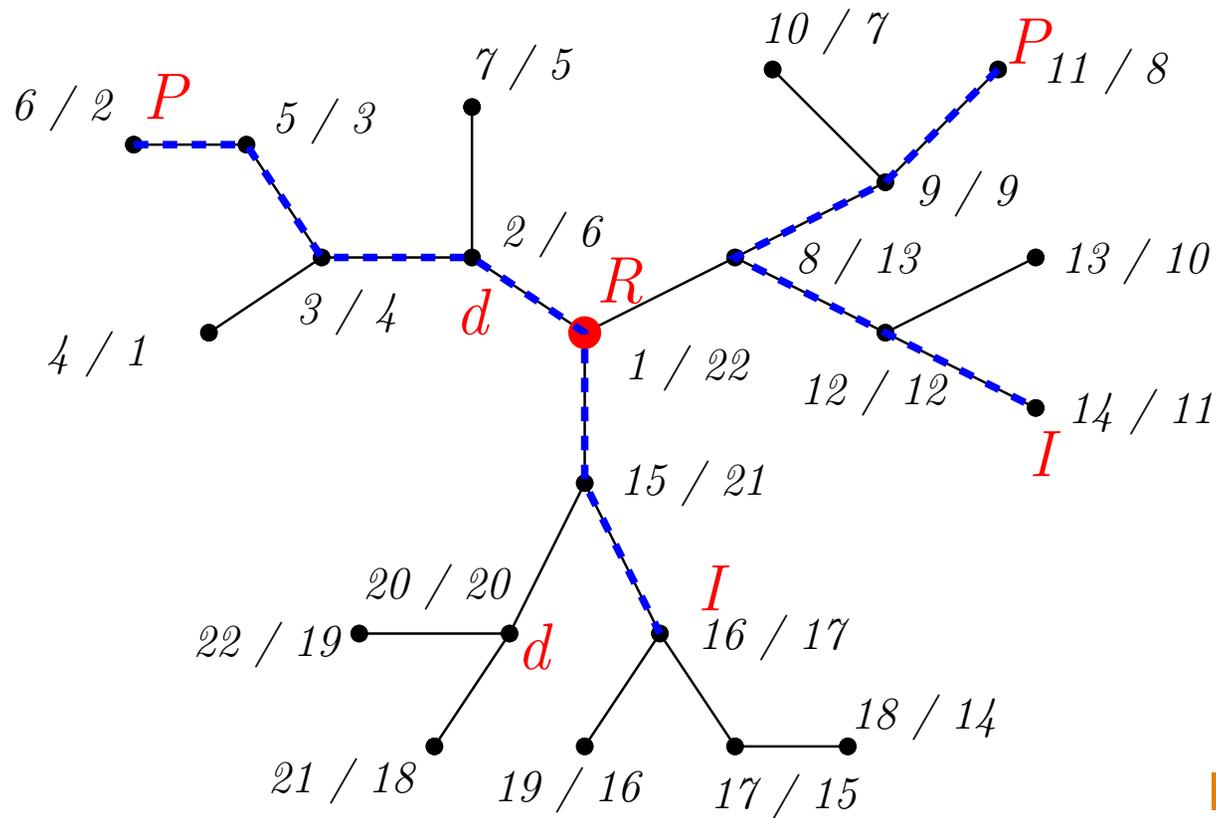
- Gemeinsamer Vorgänger X minimaler Höhe in $O(\log n)$ in \hat{T}
- Benutze leicht geänderten Dualen Baum von T
- Frage: Liegt Diagonale d auf dem Pfad von P nach I ?



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

Vorgängeranfrage in Baum

- 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 $O(\log n)$!■
- iii) \hat{G} hat $O(n)$ Kanten !■

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:

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

