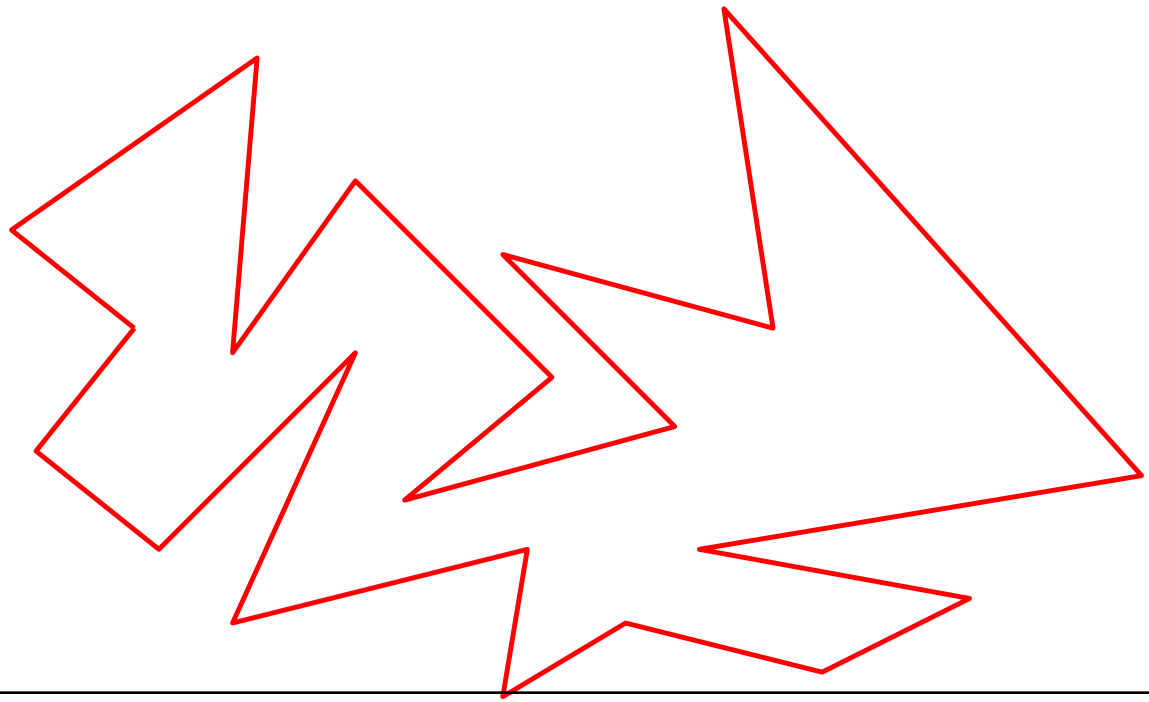
Offline Bewegungsplanung: Innerhalb von Polygonen

Elmar Langetepe
University of Bonn

Shortest Path in einfachen Polygonen

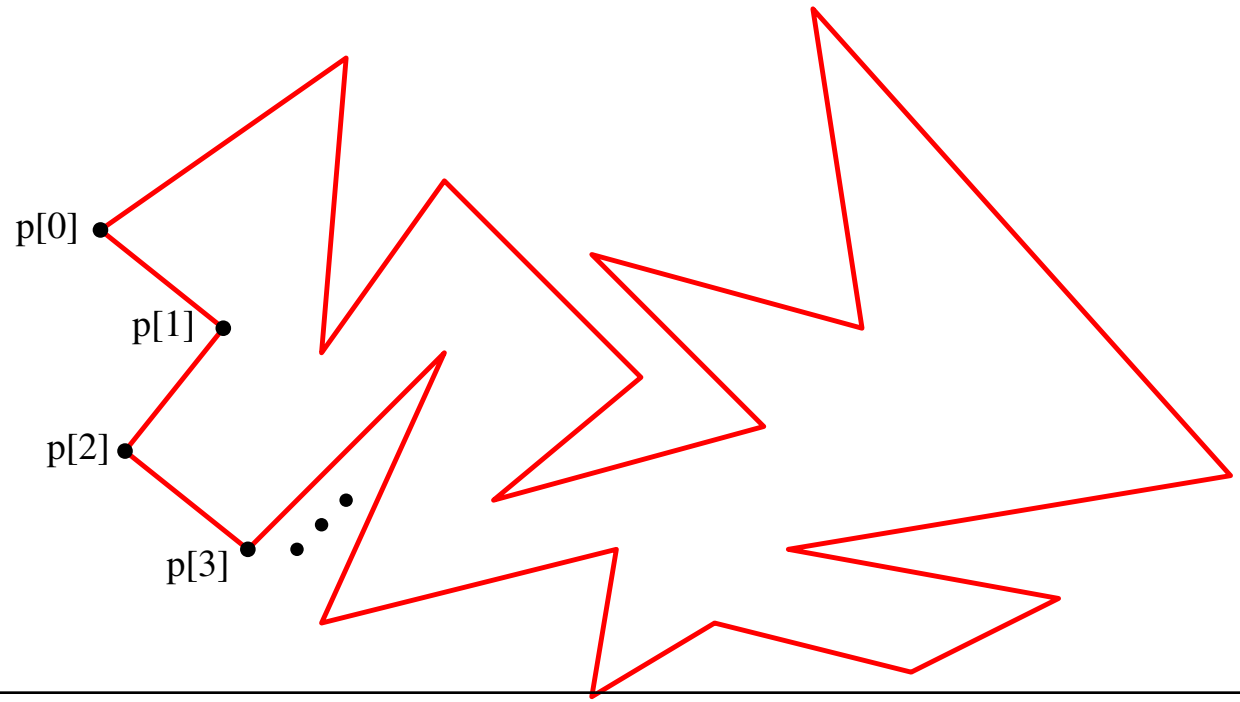
Shortest Path in einfachen Polygonen

- Einfaches Polygon P,



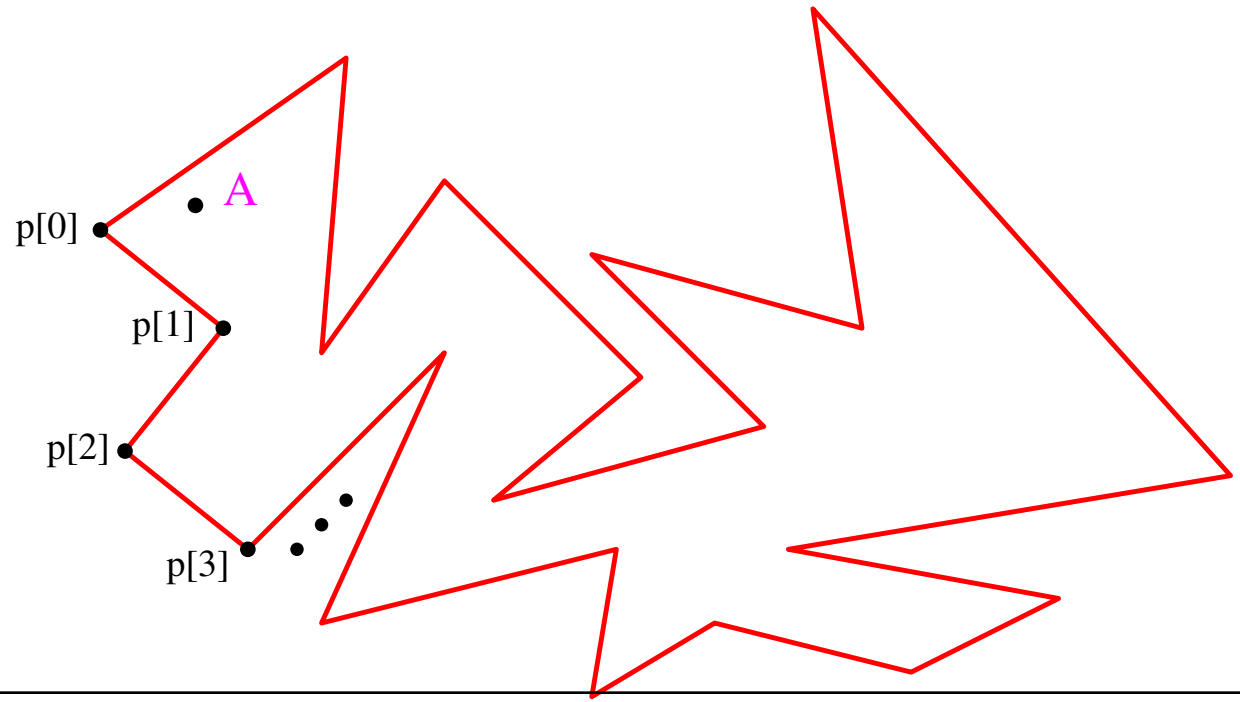
Shortest Path in einfachen Polygonen

- Einfaches Polygon P, DS: Seq. Knoten/Kanten, CCW-order



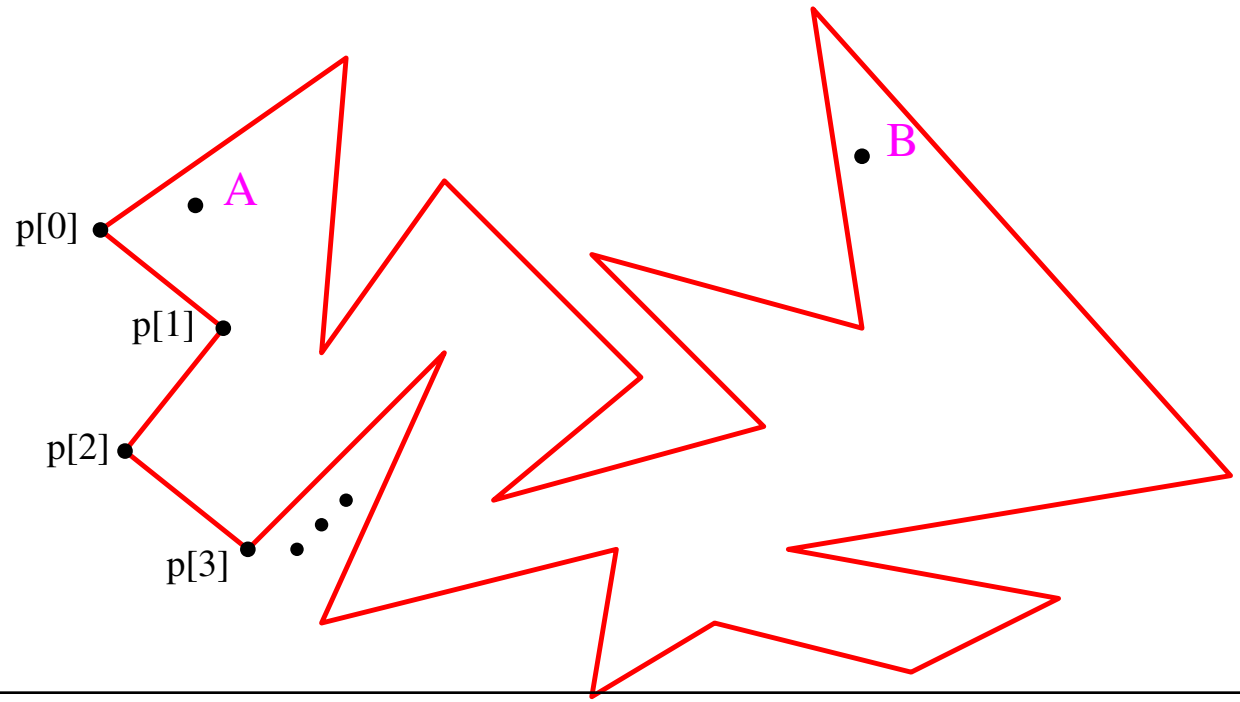
Shortest Path in einfachen Polygonen

- Einfaches Polygon P, DS: Seq. Knoten/Kanten, CCW-order
- Start A



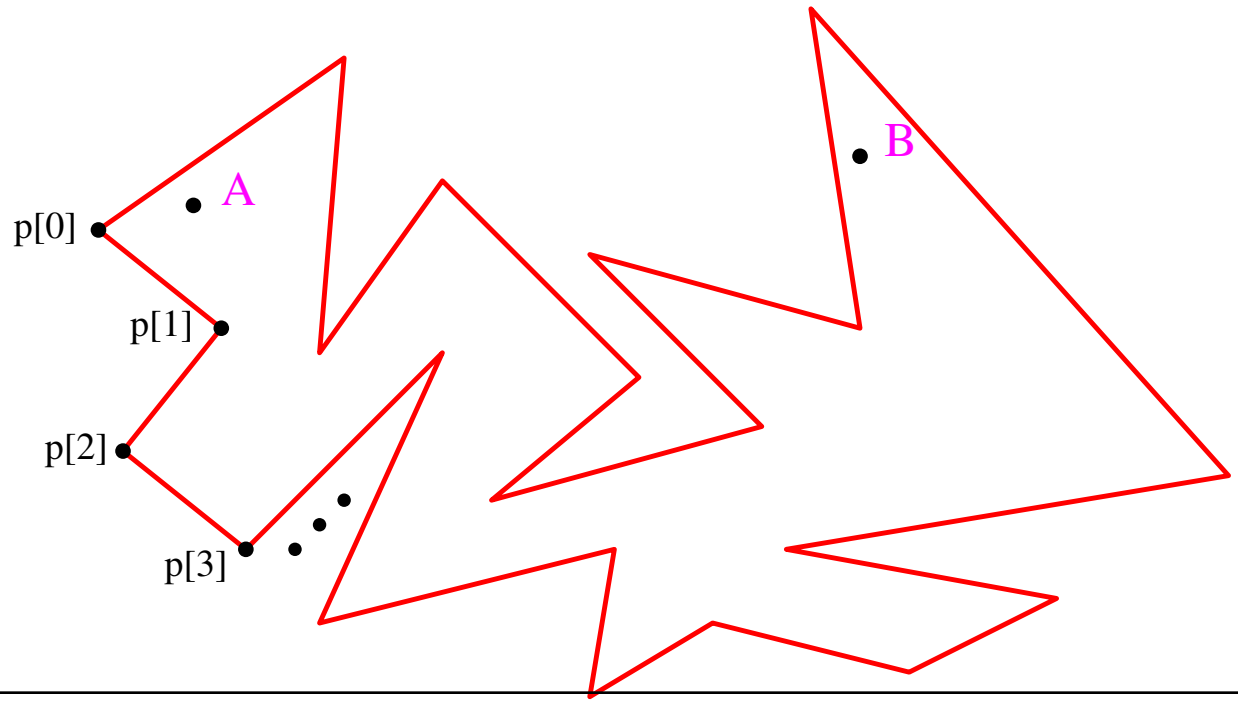
Shortest Path in einfachen Polygonen

- Einfaches Polygon P, DS: Seq. Knoten/Kanten, CCW-order
- Start A und Ziel B



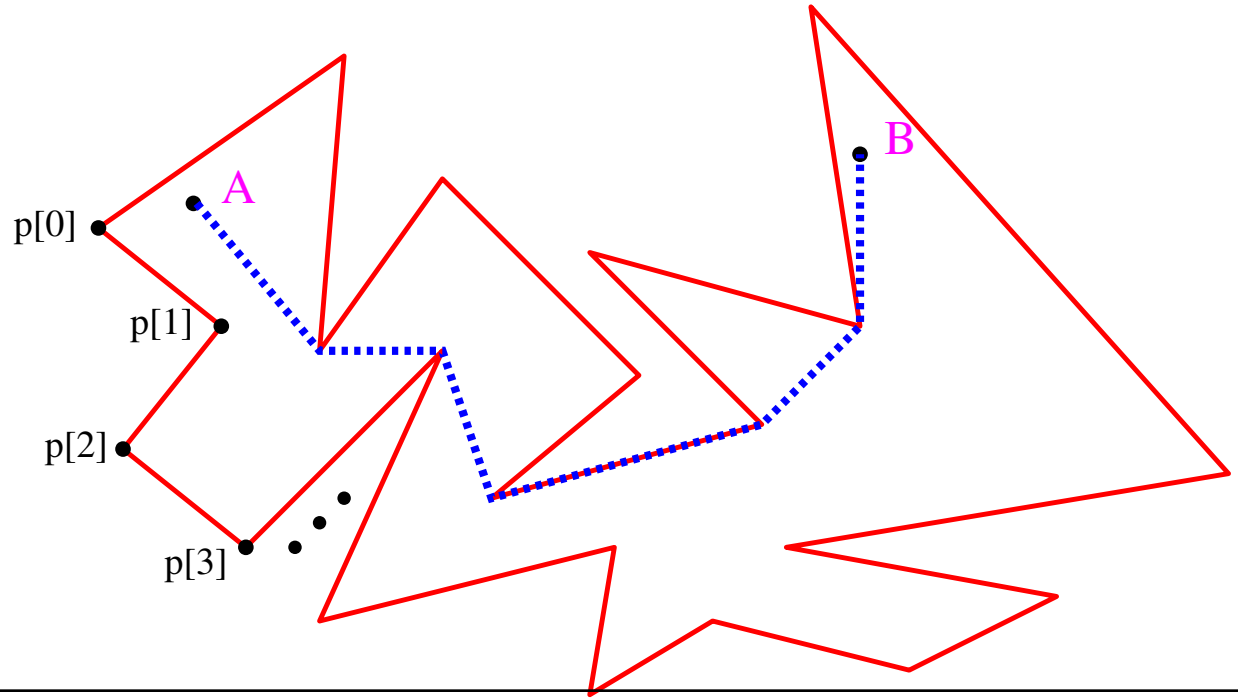
Shortest Path in einfachen Polygonen

- Einfaches Polygon P, DS: Seq. Knoten/Kanten, CCW-order
- Start A und Ziel B
- Berechne kürzesten Weg von A nach B innerhalb P



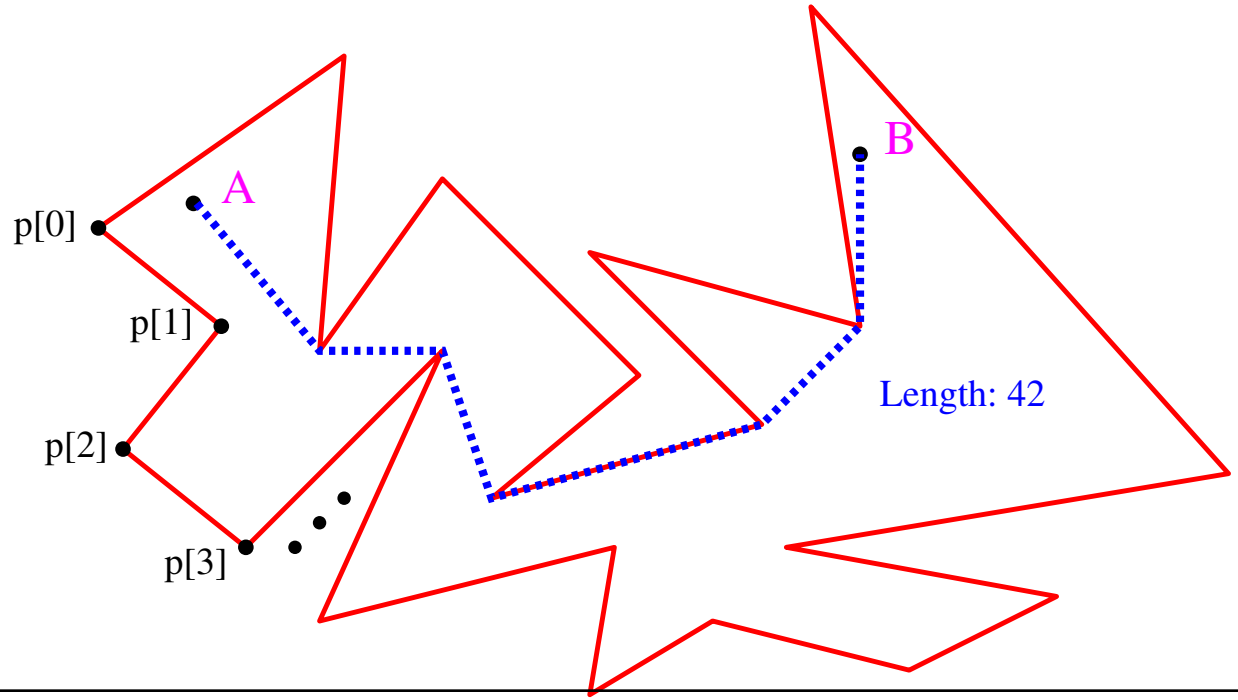
Shortest Path in einfachen Polygonen

- 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



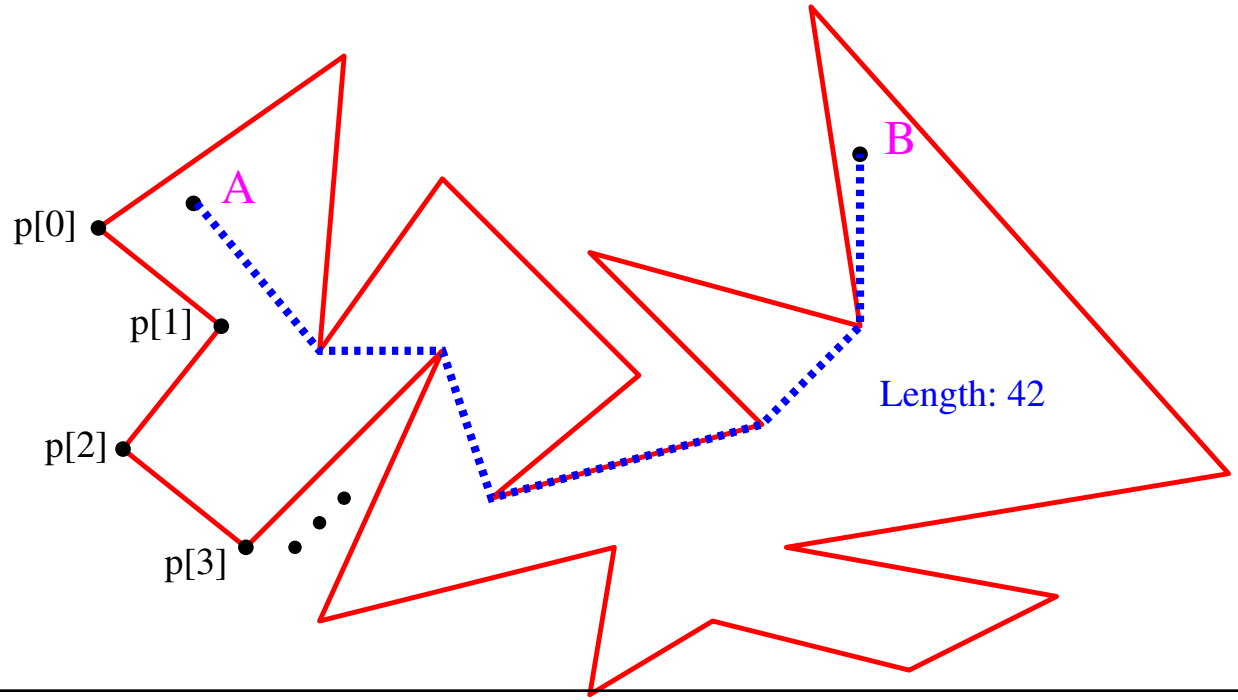
Shortest Path in einfachen Polygonen

- 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



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

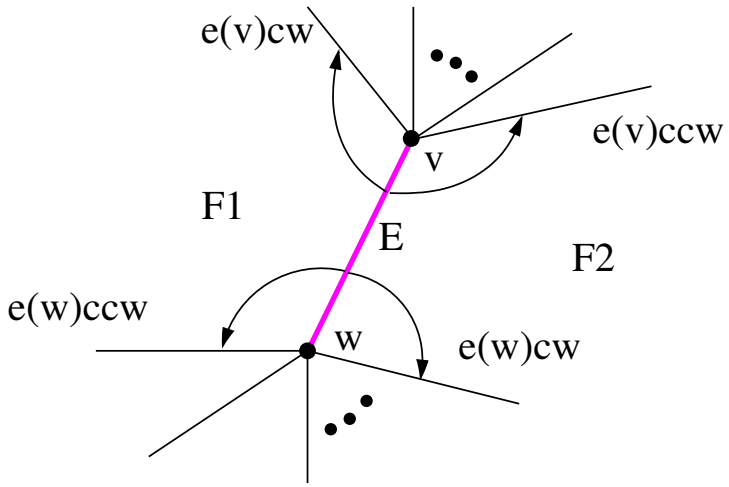
Triangulation von P

- 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



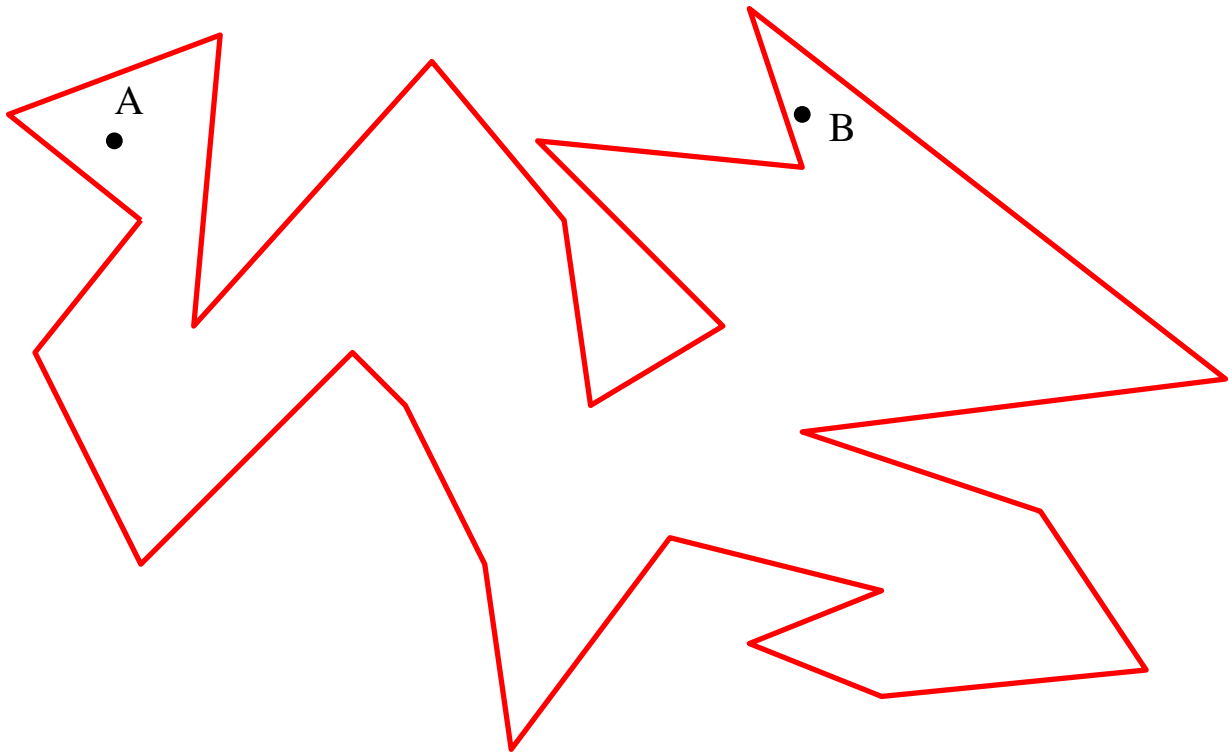
Dualer Graph eines Planaren Graphen

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

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

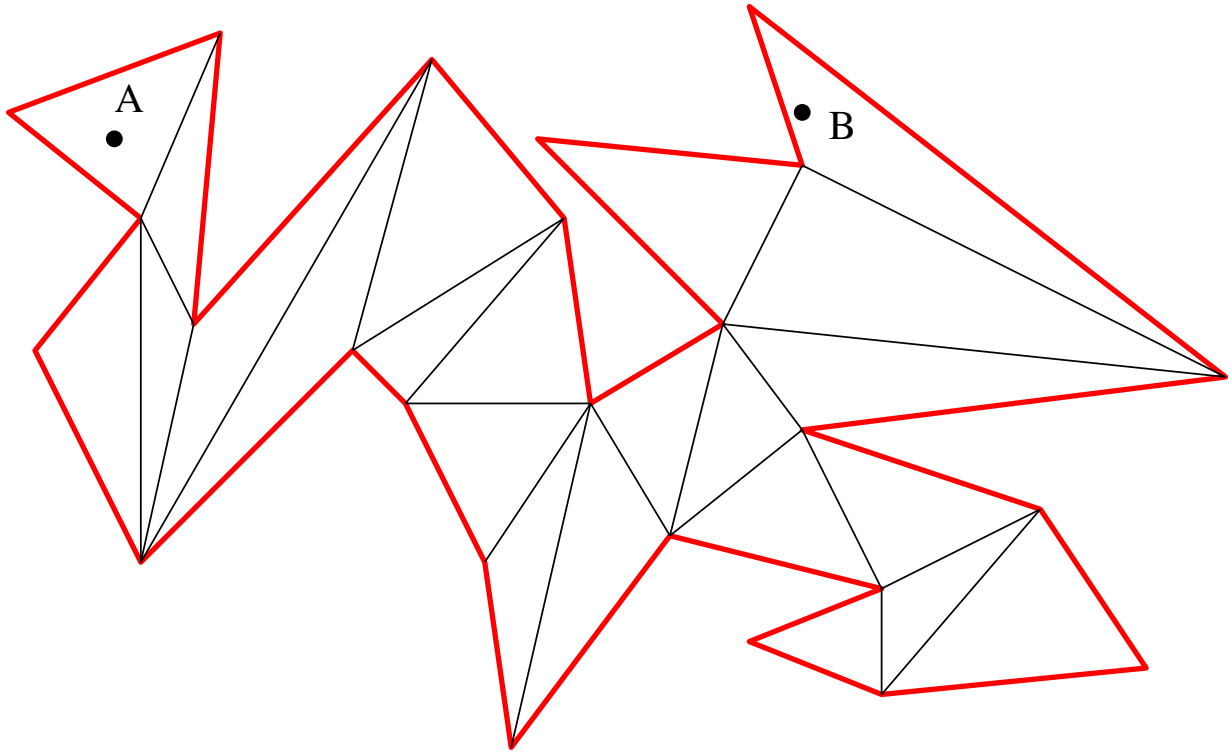
Einfache, effiziente Lösung (Lee/Preparata)

Einfache, effiziente Lösung (Lee/Preparata)



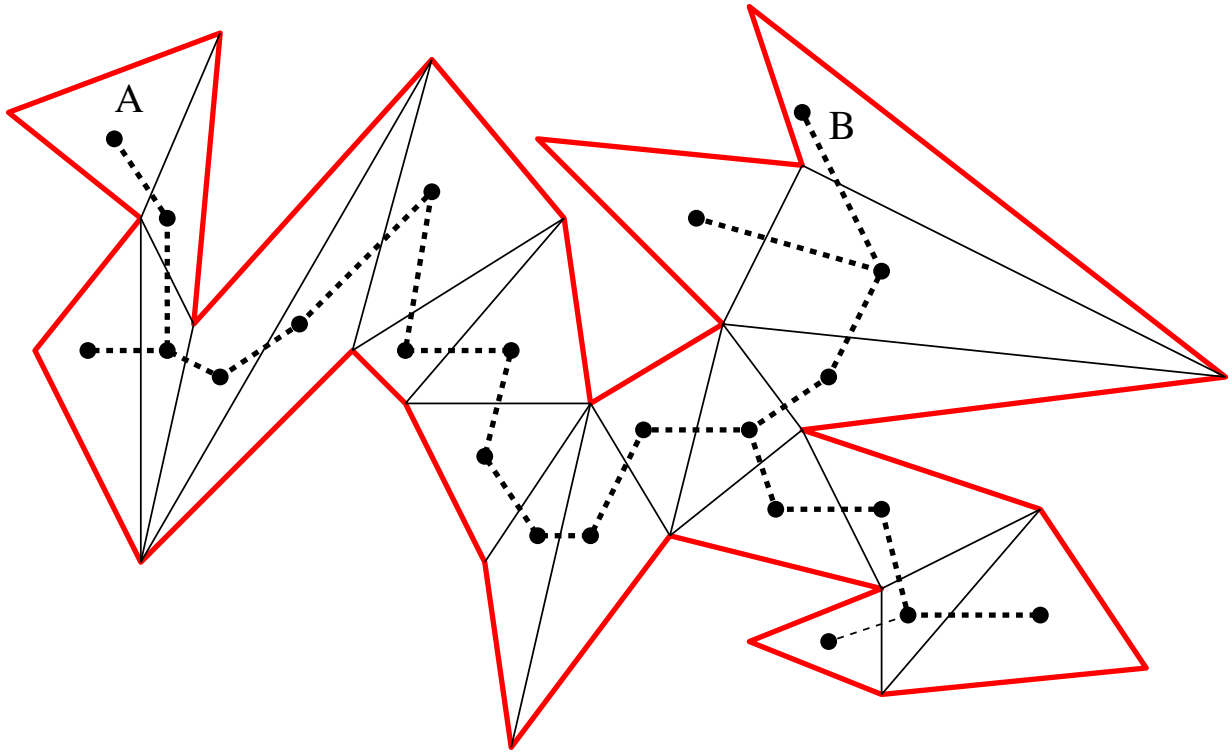
Einfache, effiziente Lösung (Lee/Preparata)

- Triangulation von P: Graph T, DS: DCEL, Adjacency list



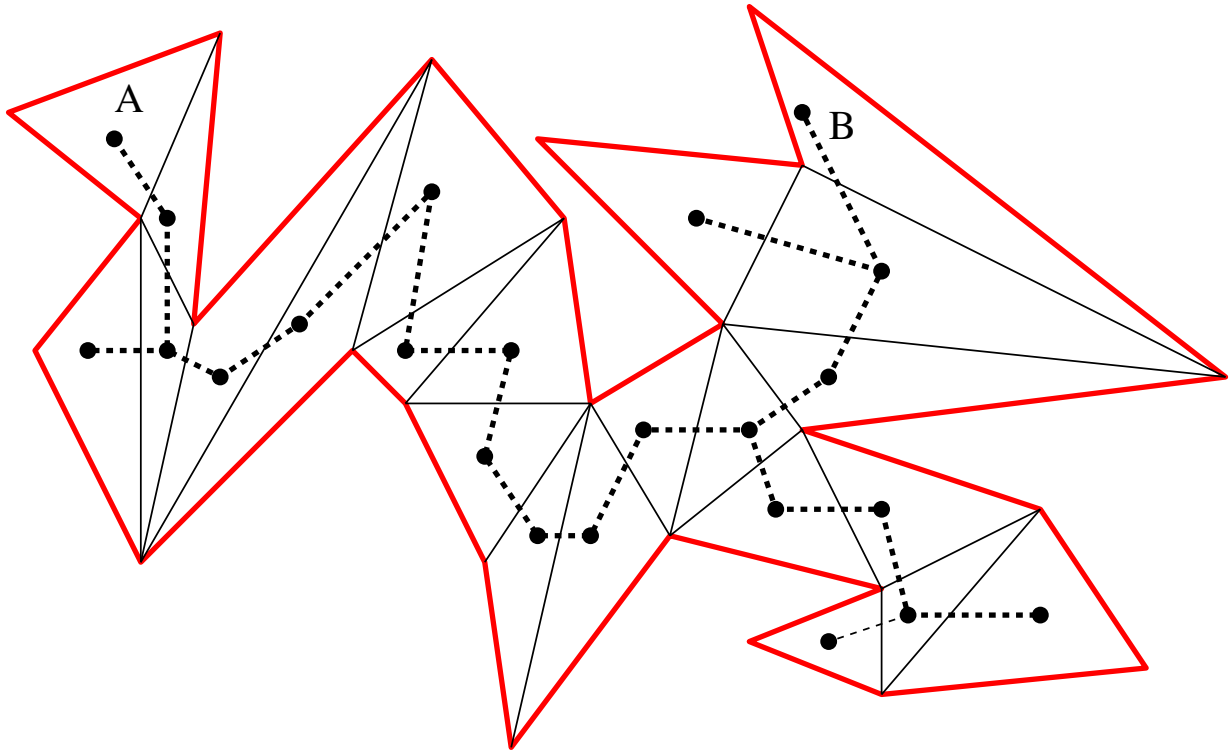
Einfache, effiziente Lösung (Lee/Preparata)

- Triangulation von P: Graph T, DS: DCEL, Adjacency list
- Dualer Graph: $D(T)$ (Tree),



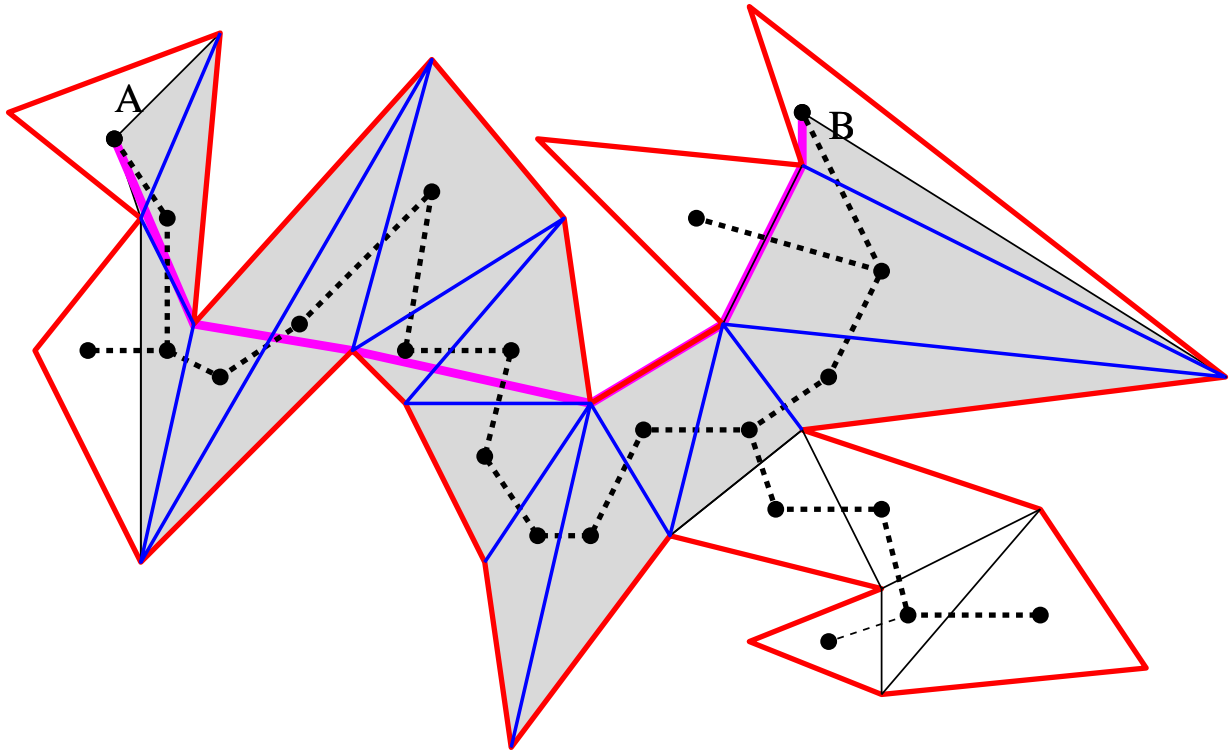
Einfache, effiziente Lösung (Lee/Preparata)

- Triangulation von P: Graph T, DS: DCEL, Adjacency list
- Dualer Graph: $D(T)$ (Tree), Depth First Search (DFS) on $D(T)$



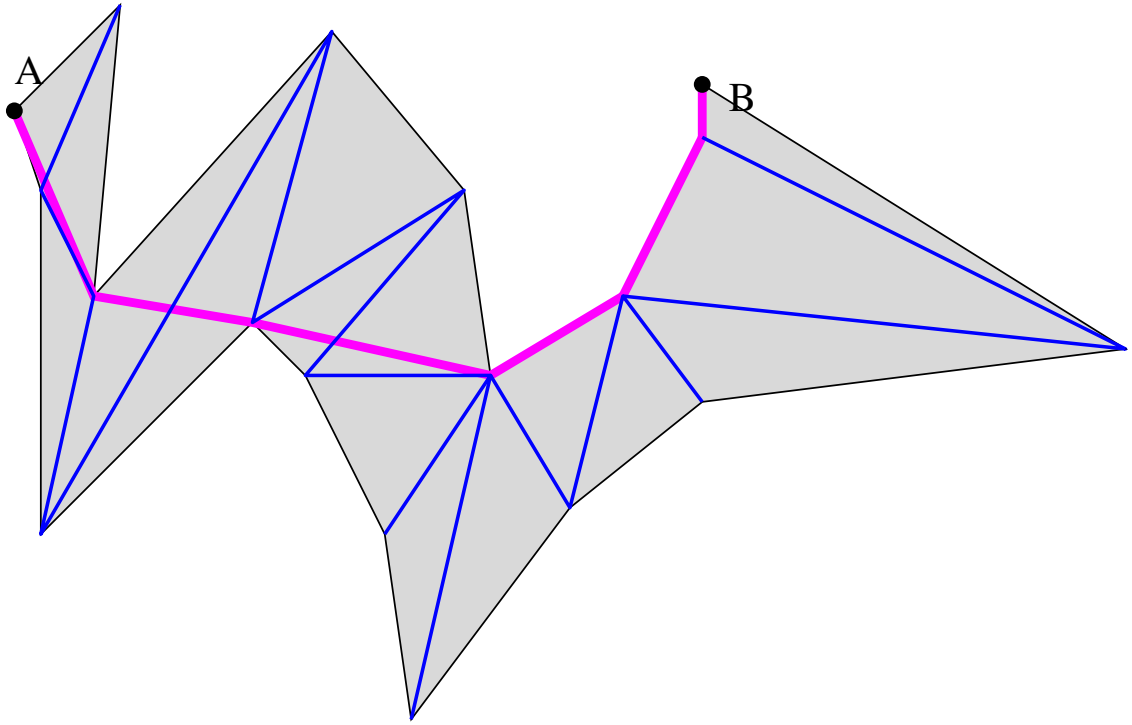
Einfache, effiziente Lösung (Lee/Preparata)

- 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



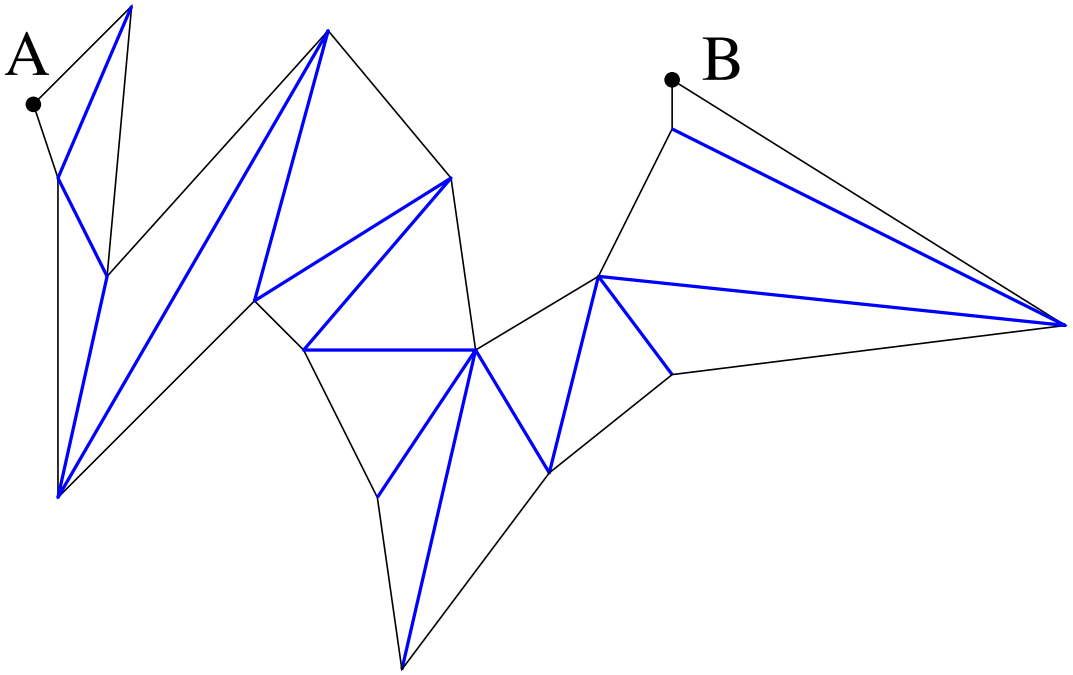
Einfache, effiziente Lösung (Lee/Preparata)

- 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



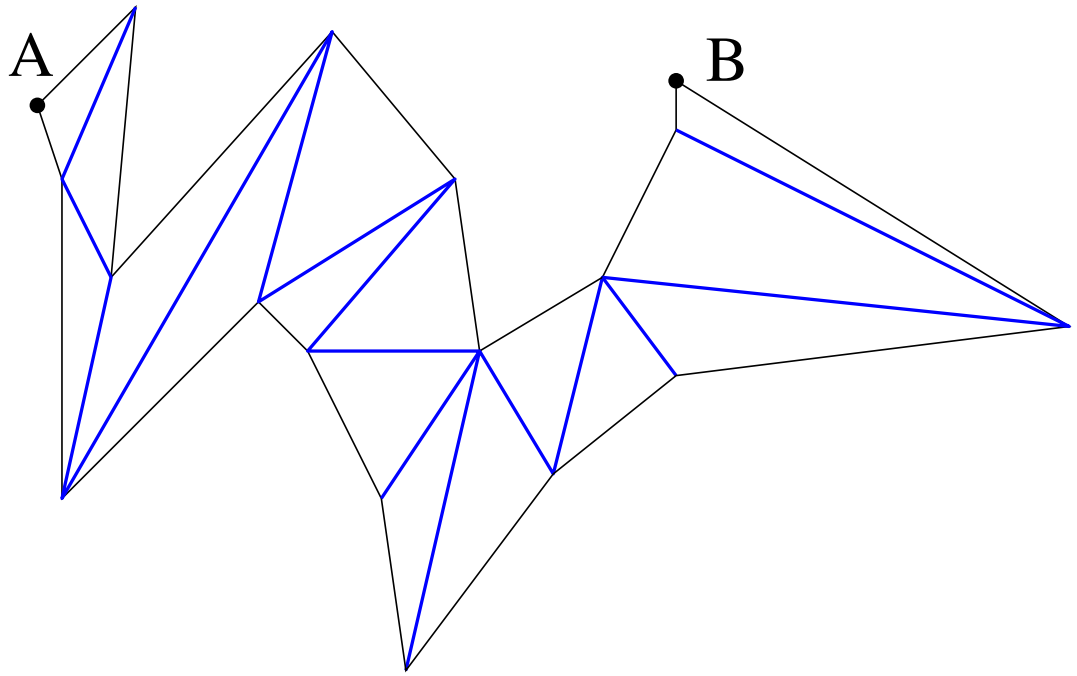
Shortest Path in P'

Shortest Path in P'



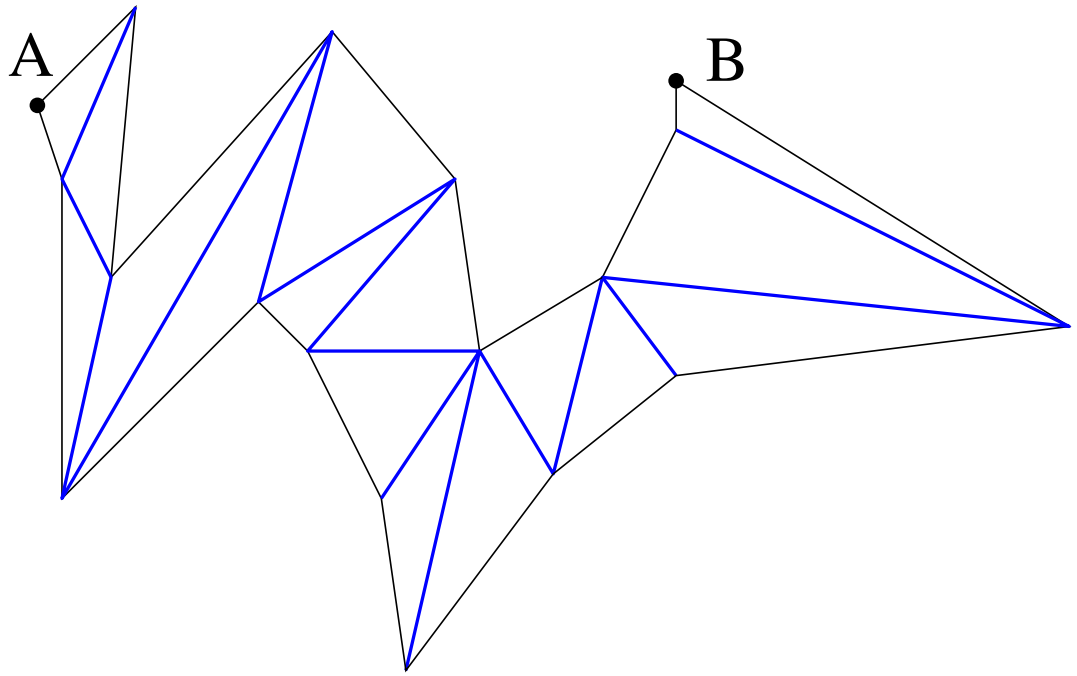
Shortest Path in P'

- Kette von of Dreiecken/Diagonalen (Data structure)



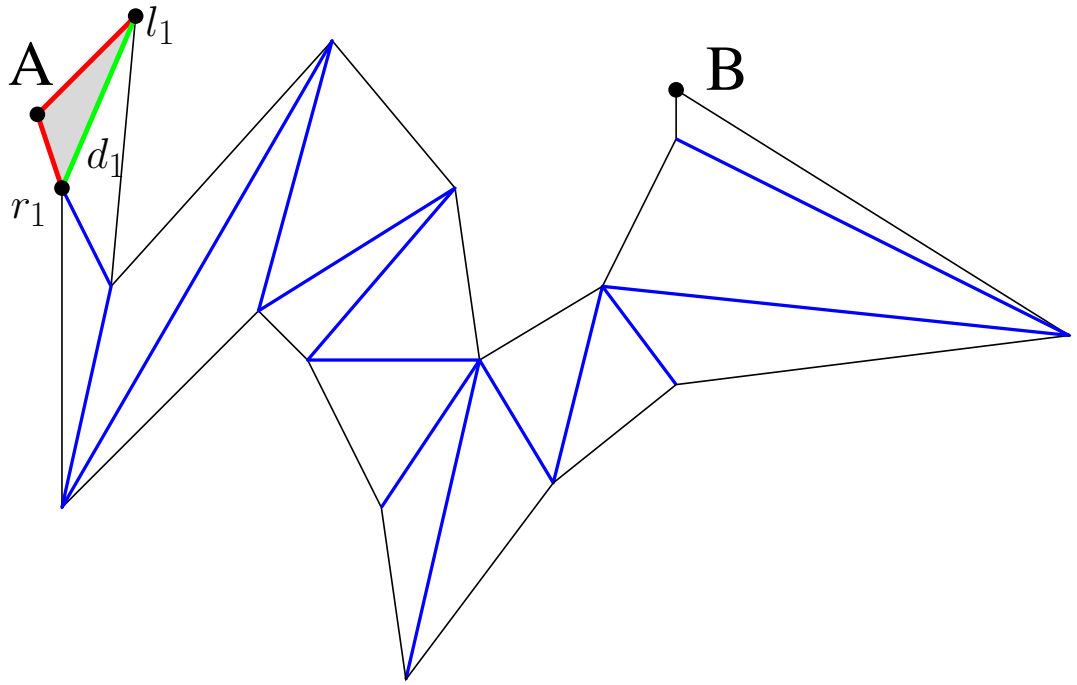
Shortest Path in P'

- Kette von of Dreiecken/Diagonalen (Data structure)
- Induktiv: Shortest paths zu den Endpunkten der Diagonalen



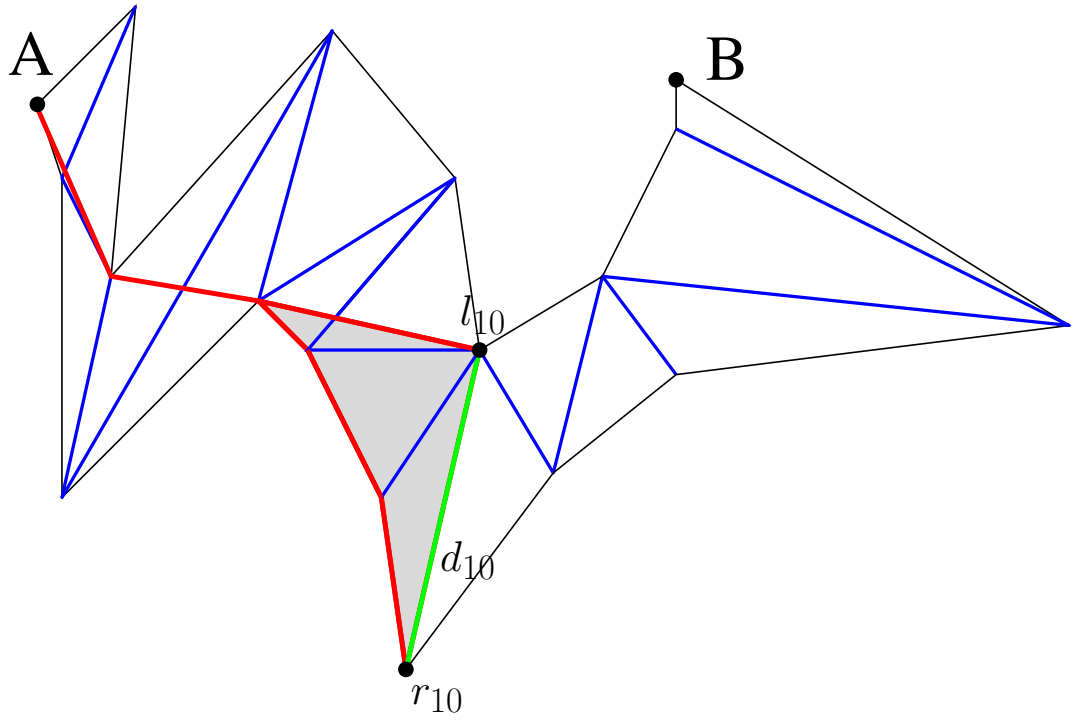
Shortest Path in P'

- Kette von of Dreiecken/Diagonalen (Data structure)
- Induktiv: Shortest paths zu den Endpunkten der Diagonalen
- Ind. Anf,



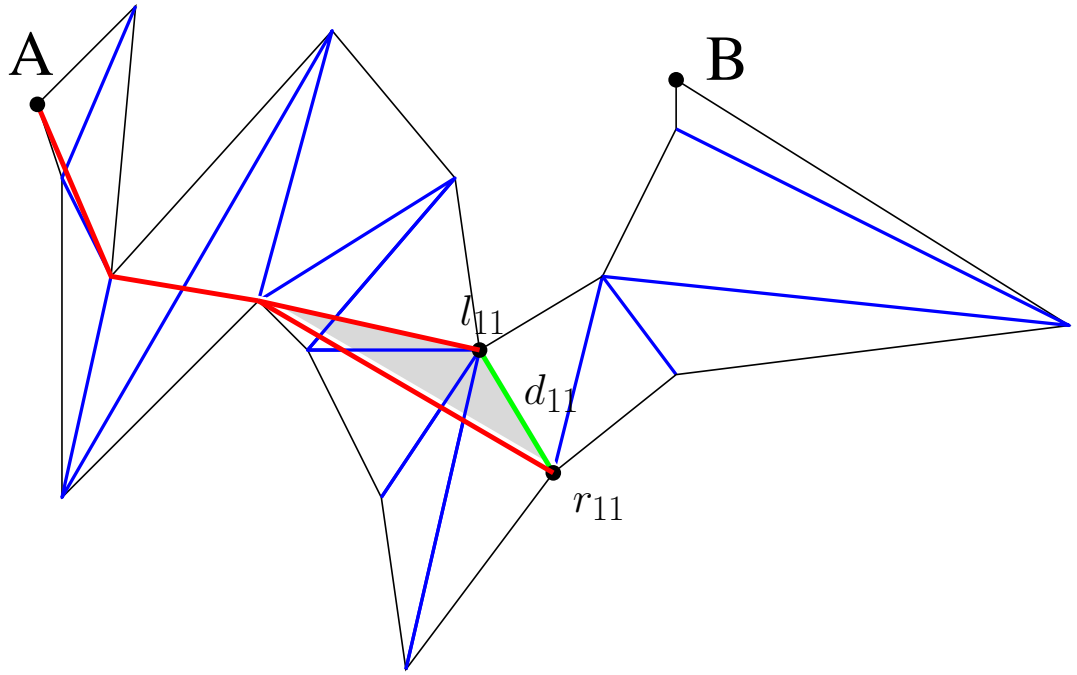
Shortest Path in P'

- Kette von of Dreiecken/Diagonalen (Data structure)
- Induktiv: Shortest paths zu den Endpunkten der Diagonalen
- Ind. Anf, Ind. Schritt,



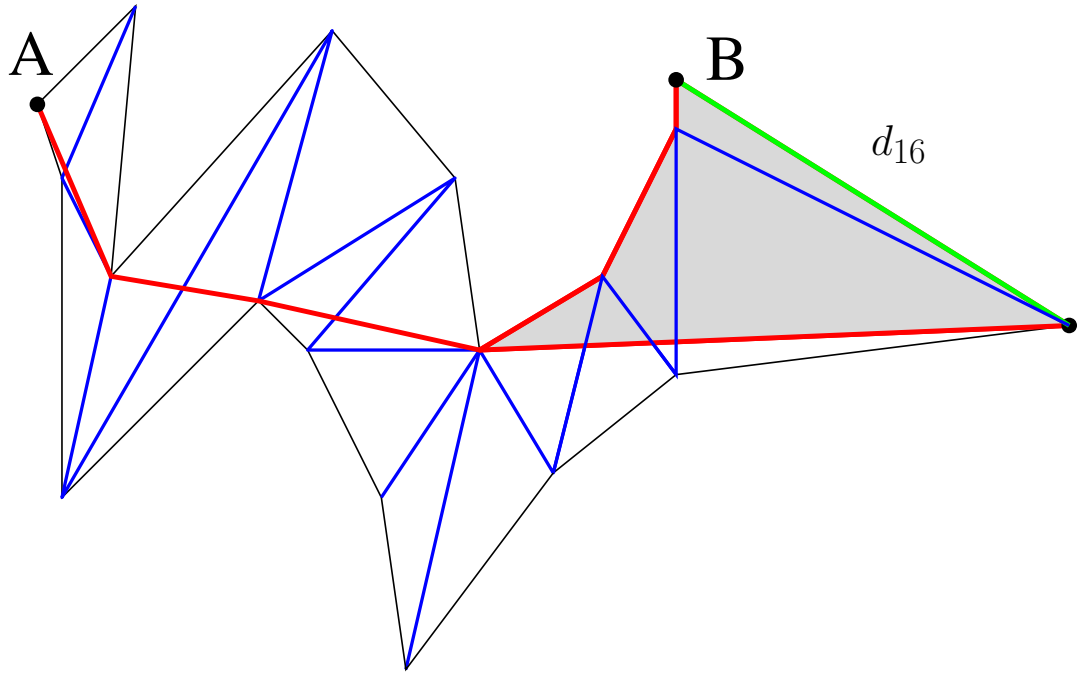
Shortest Path in P'

- Kette von of Dreiecken/Diagonalen (Data structure)
- Induktiv: Shortest paths zu den Endpunkten der Diagonalen
- Ind. Anf, Ind. Schritt,



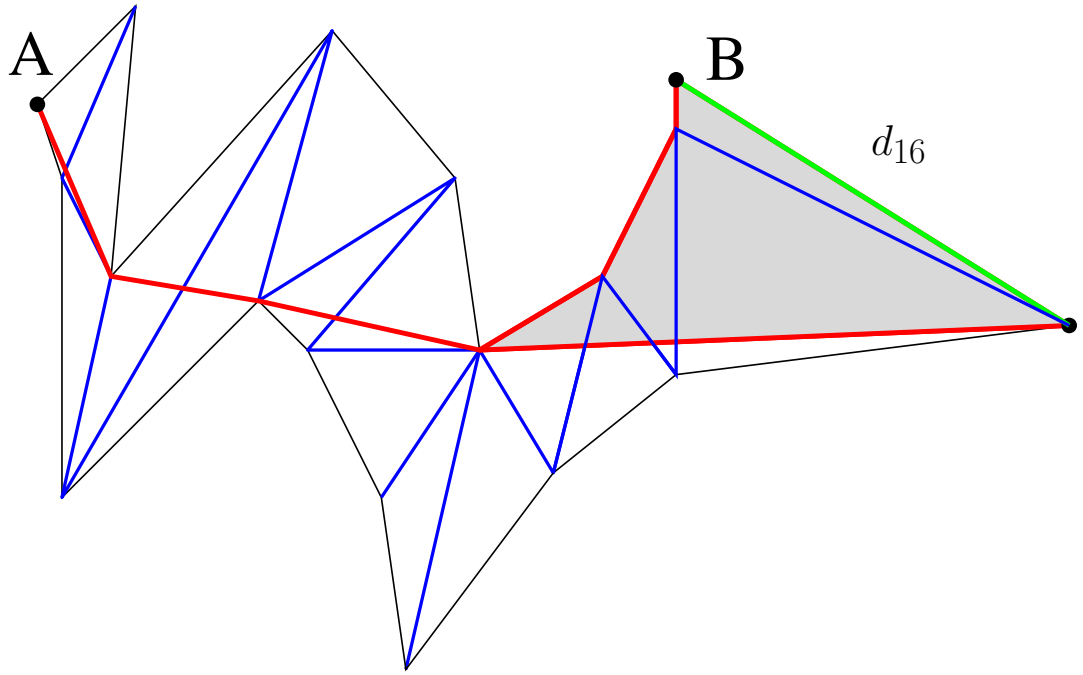
Shortest Path in P'

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



Shortest Path in P'

- Kette von of 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

Zusammenfassung: Alg. 1.4

- Triangulation von P:

Zusammenfassung: Alg. 1.4

- Triangulation von P: $O(n)$ Zeit und Platz

Zusammenfassung: Alg. 1.4

- Triangulation von P: $O(n)$ Zeit und Platz
- Dualer Graph der Triangulation:

Zusammenfassung: Alg. 1.4

- Triangulation von P: $O(n)$ Zeit und Platz
- Dualer Graph der Triangulation: $O(n)$ Zeit und Platz

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:

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

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':

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

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:

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: $O(n)$??

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!

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!!

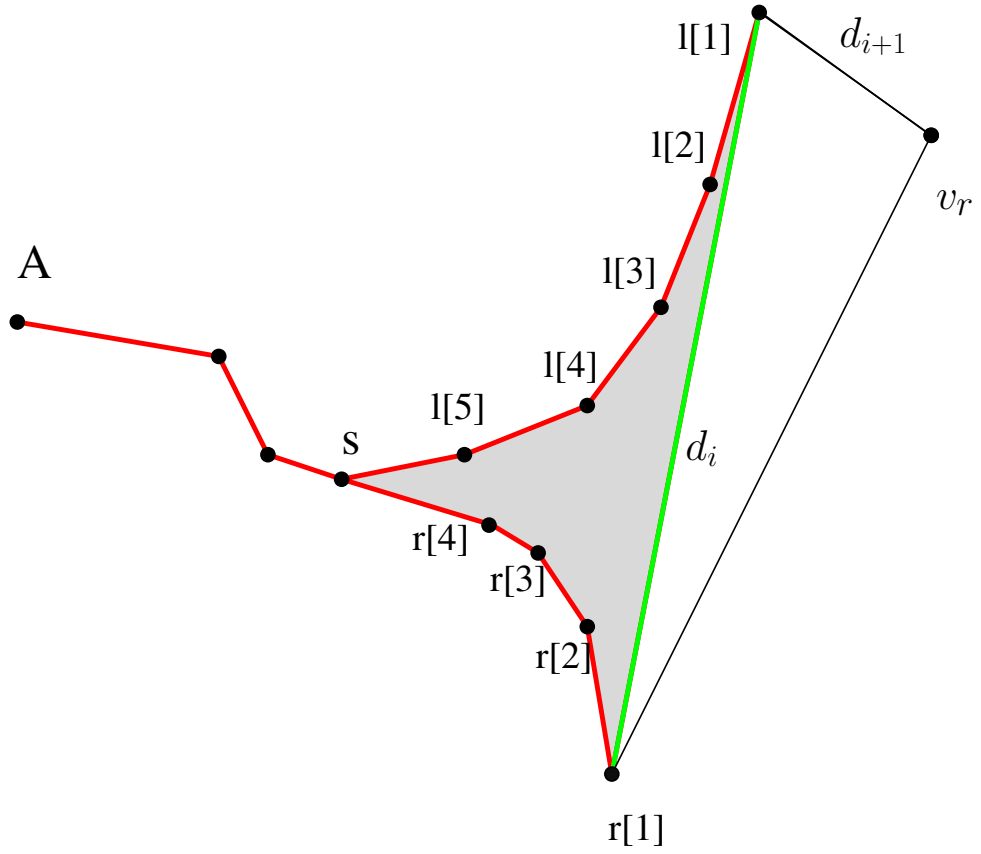
Zusammenfassung: Alg. 1.4

- 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: Indukiver Schritt!

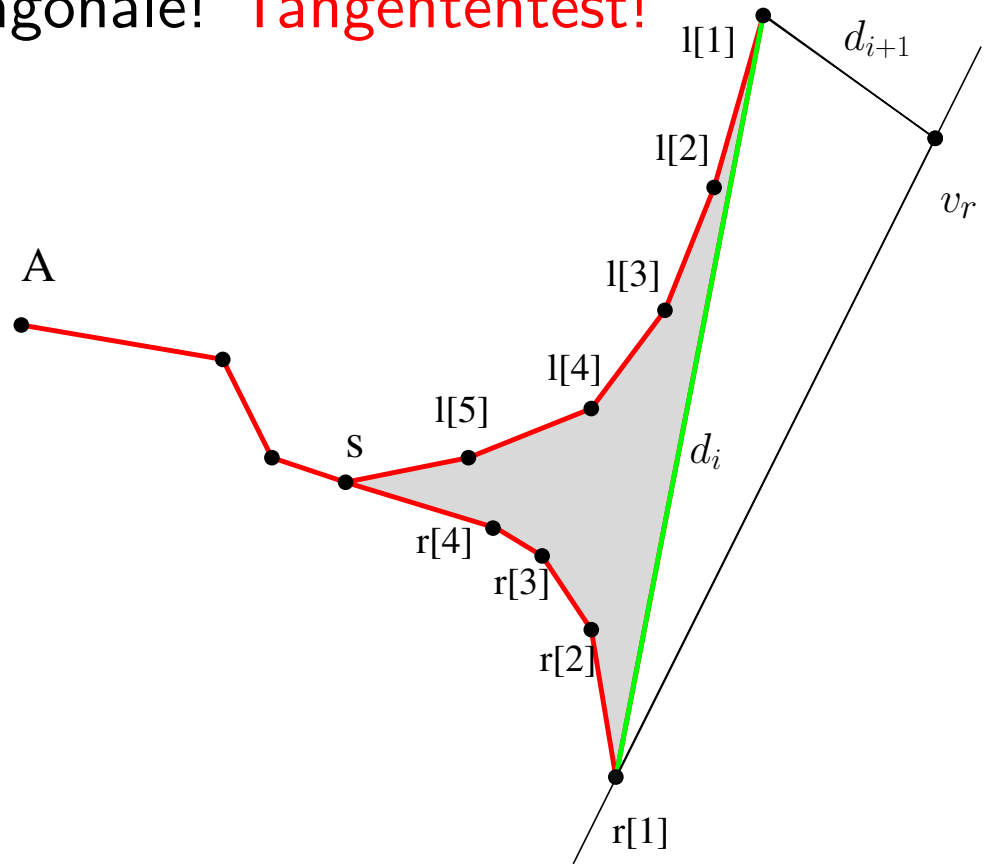
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!



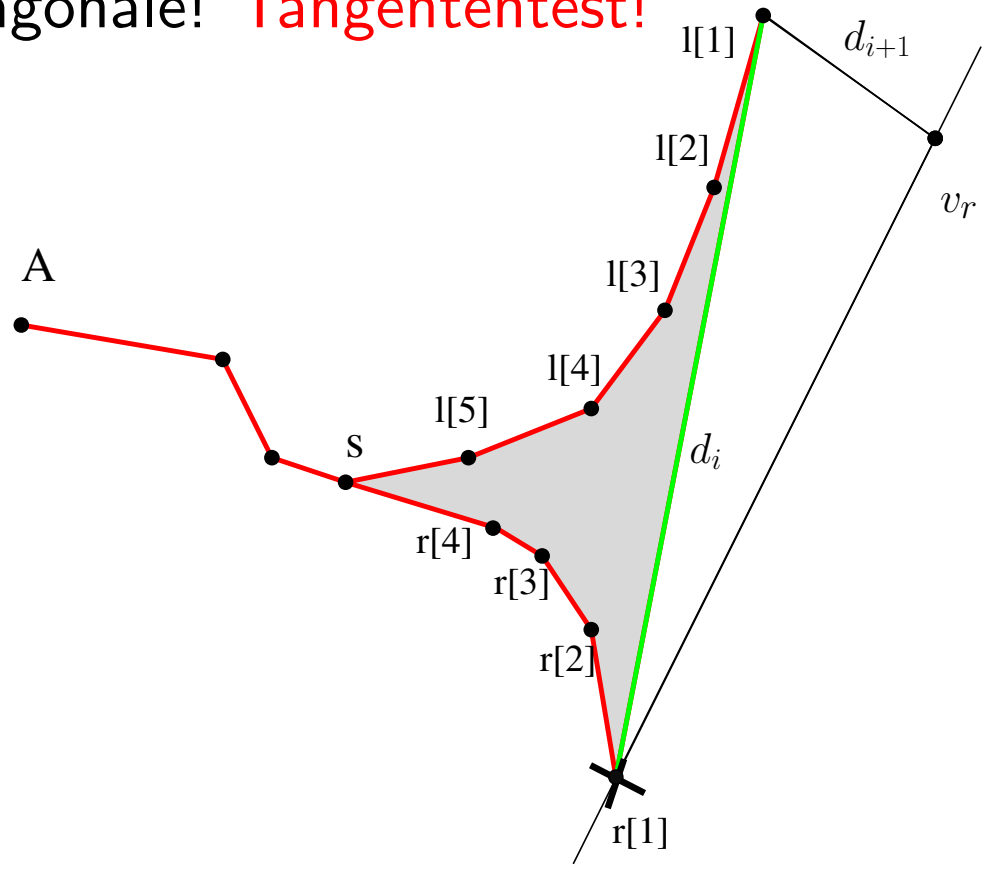
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest!



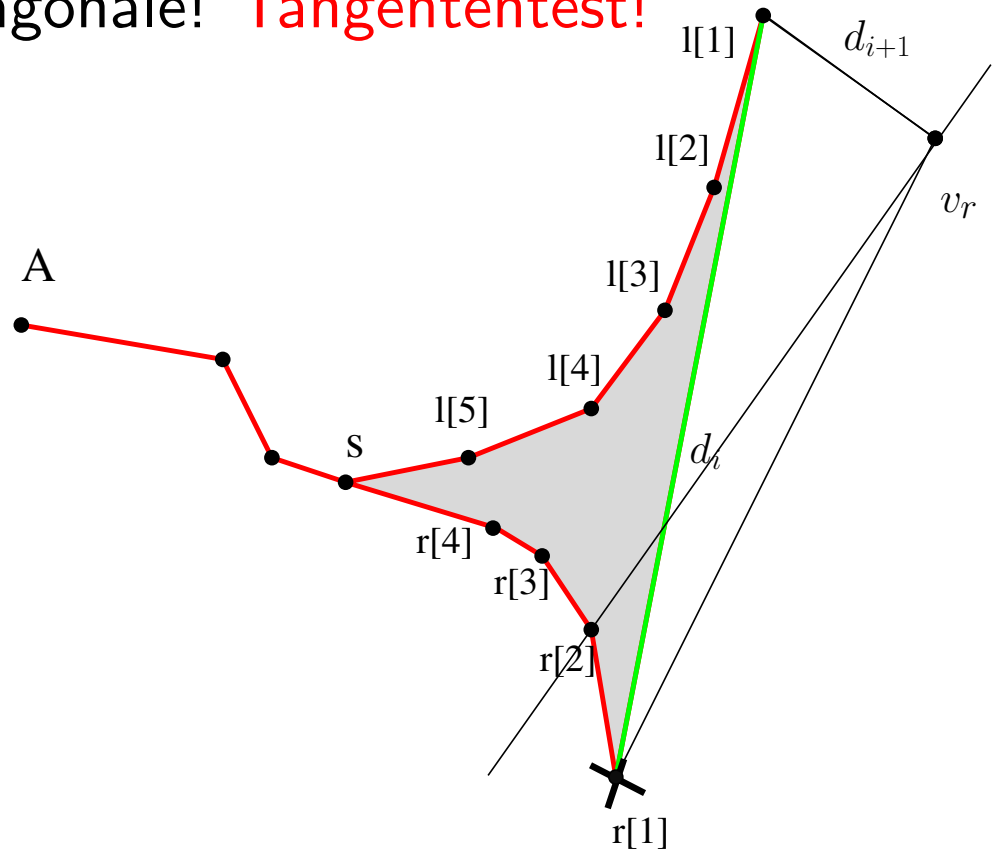
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest!



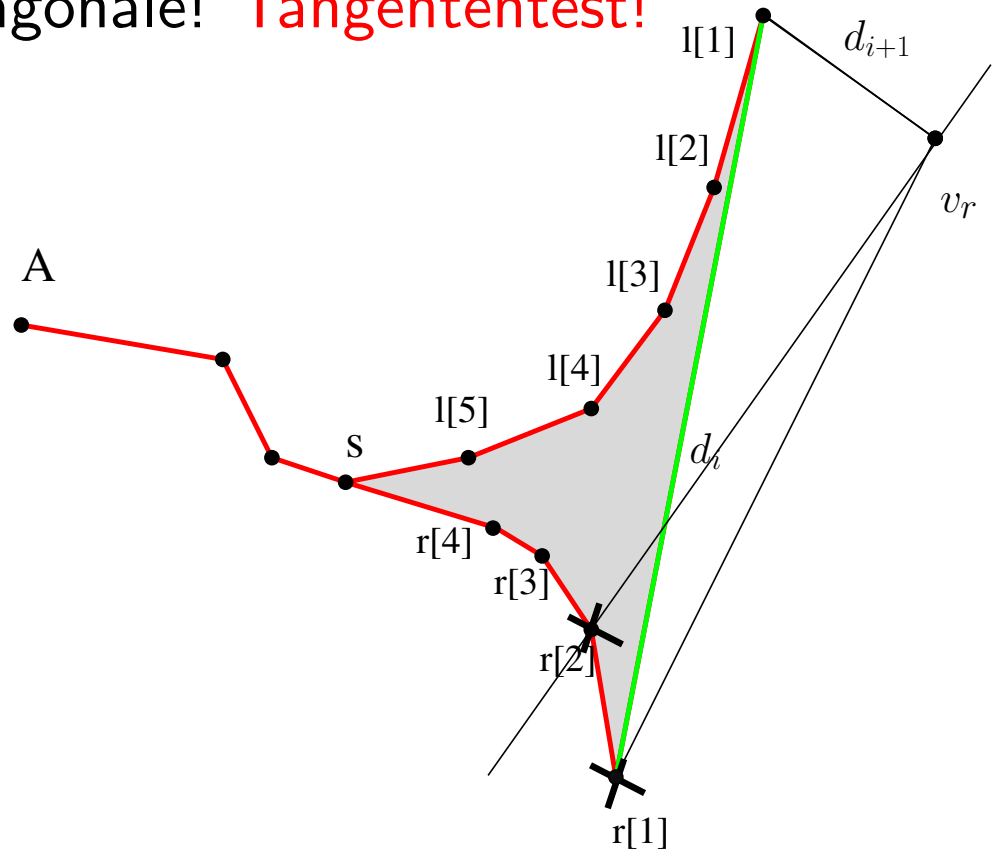
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest!



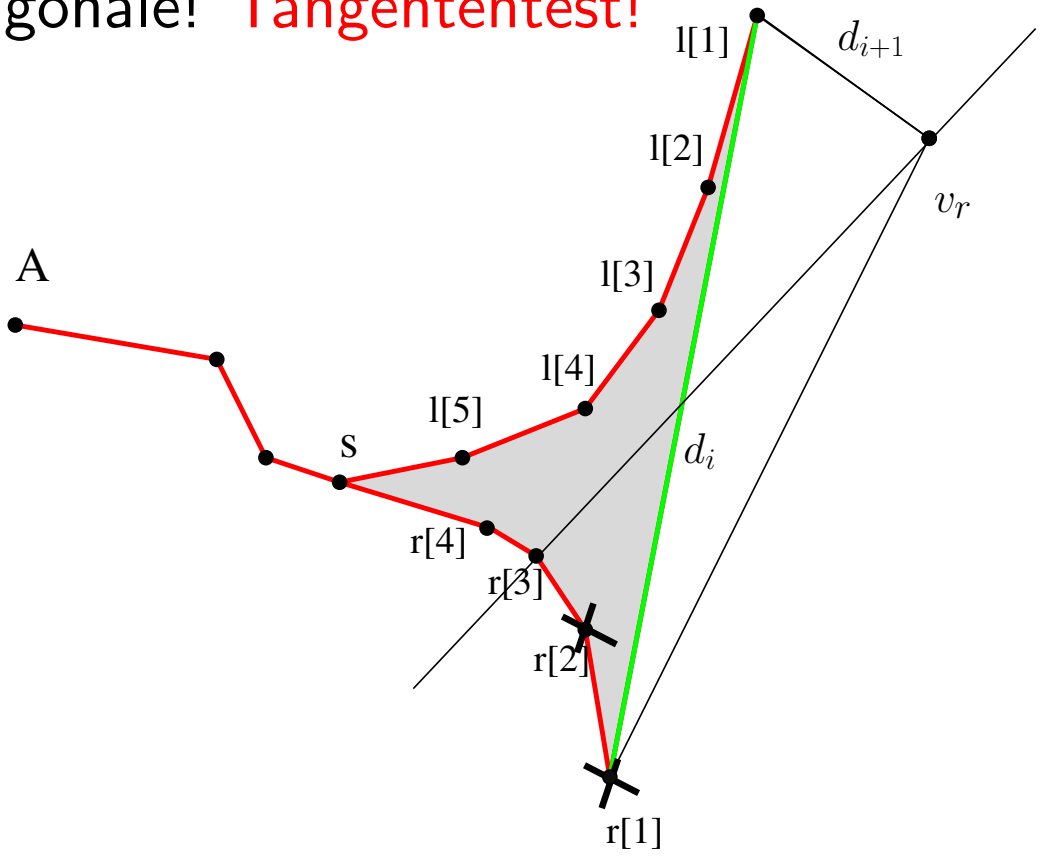
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest!



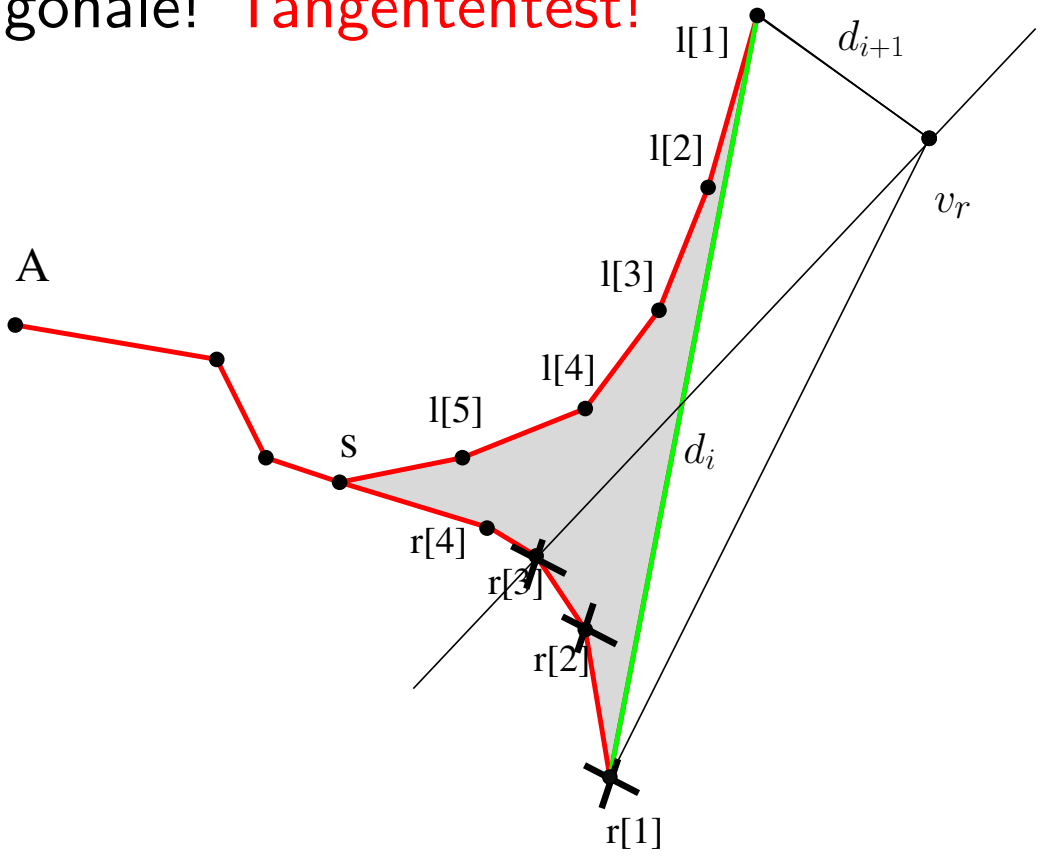
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest!



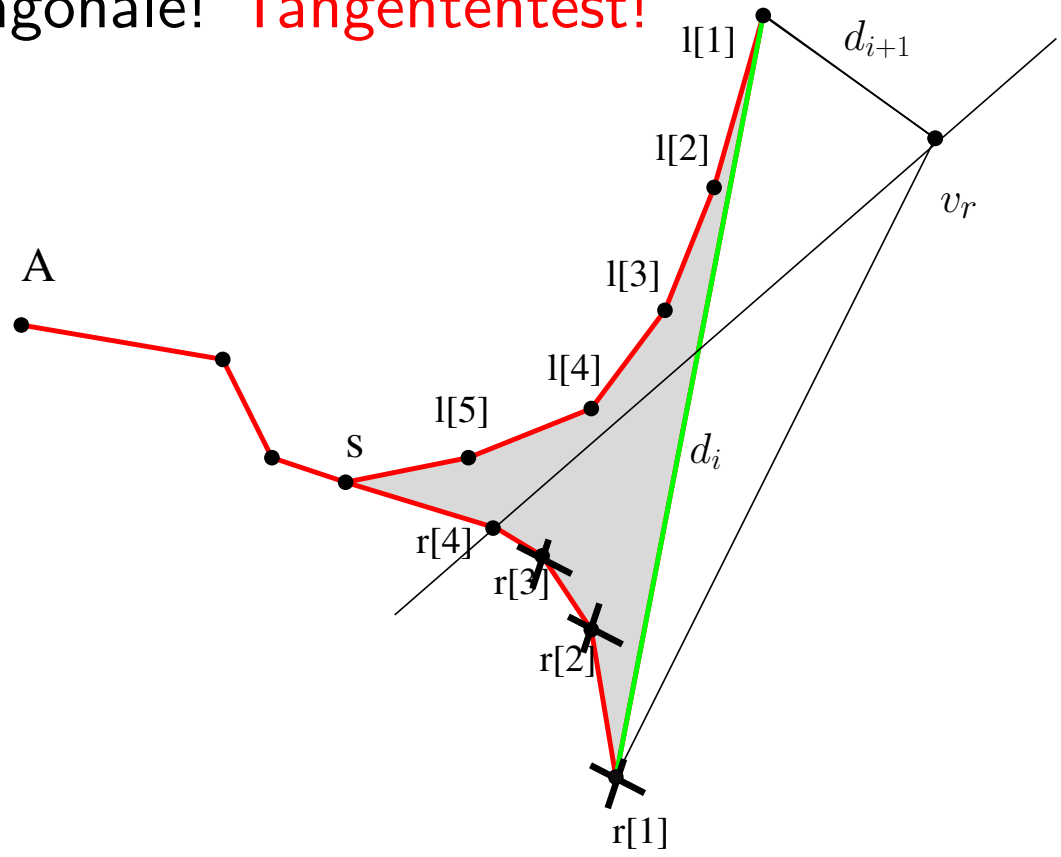
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest!



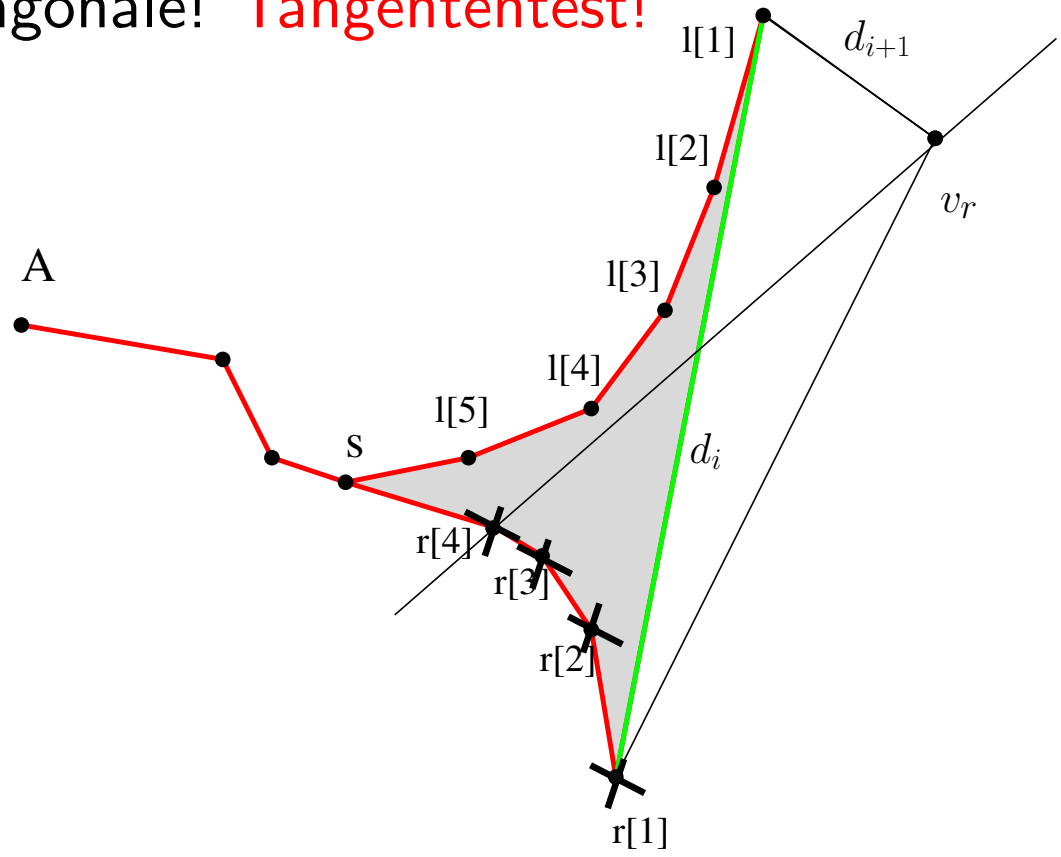
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest!



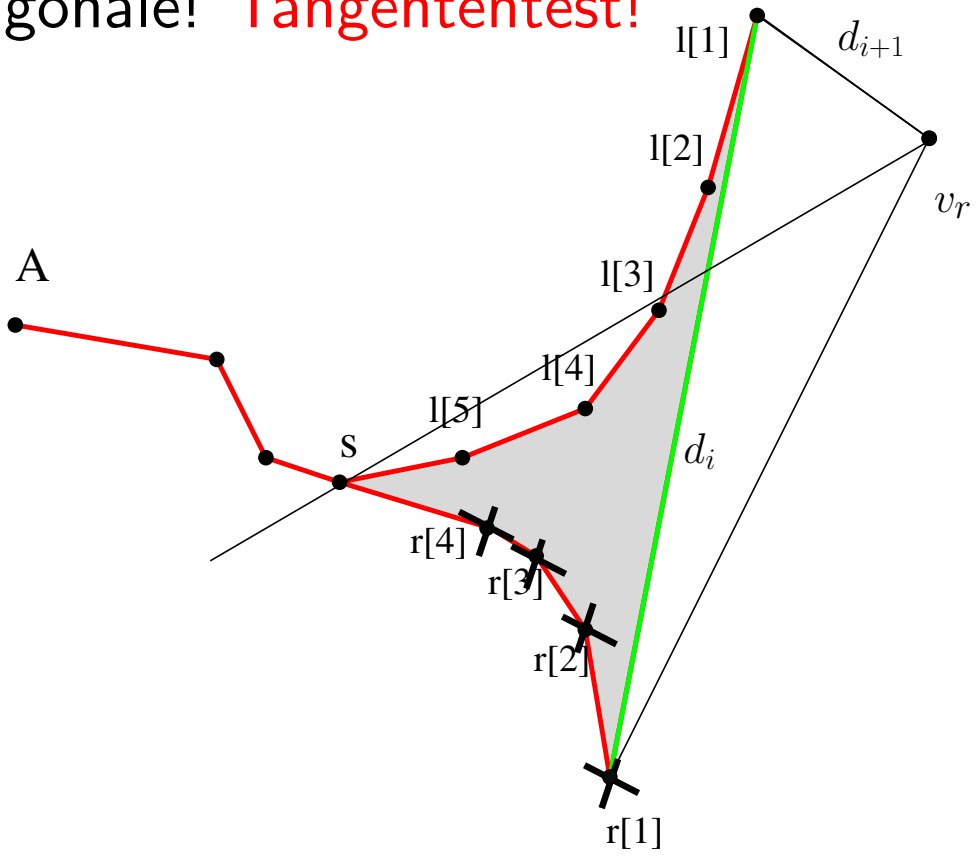
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest!



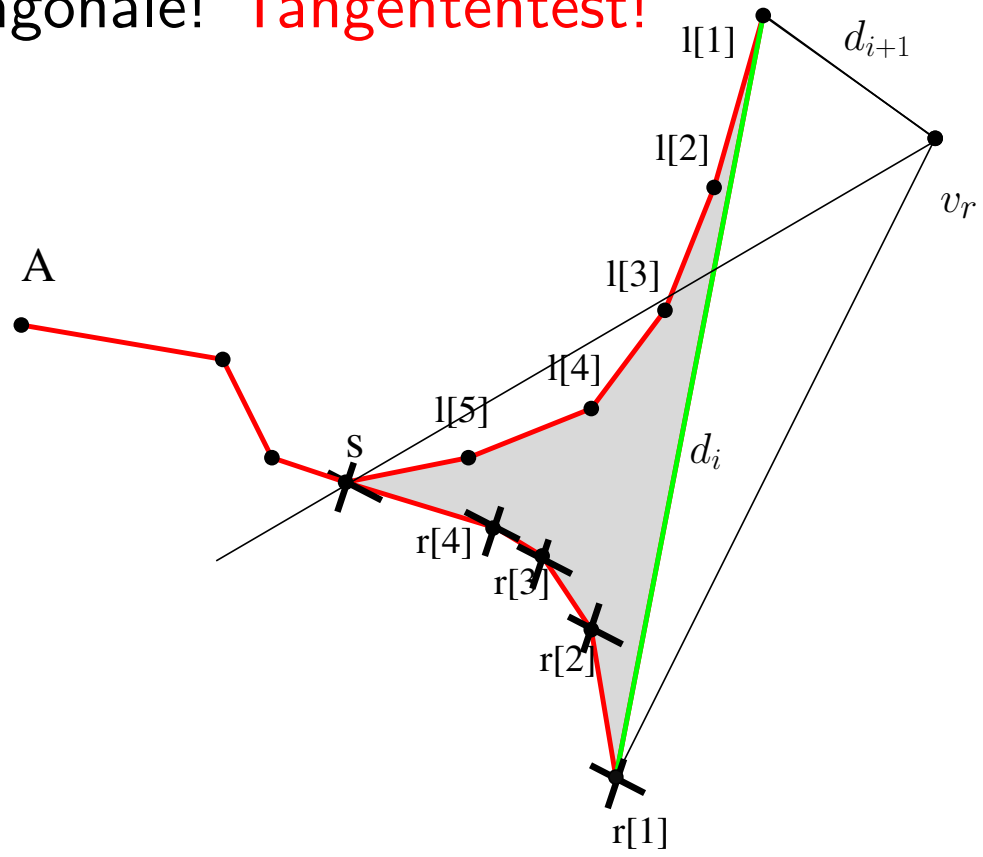
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest!



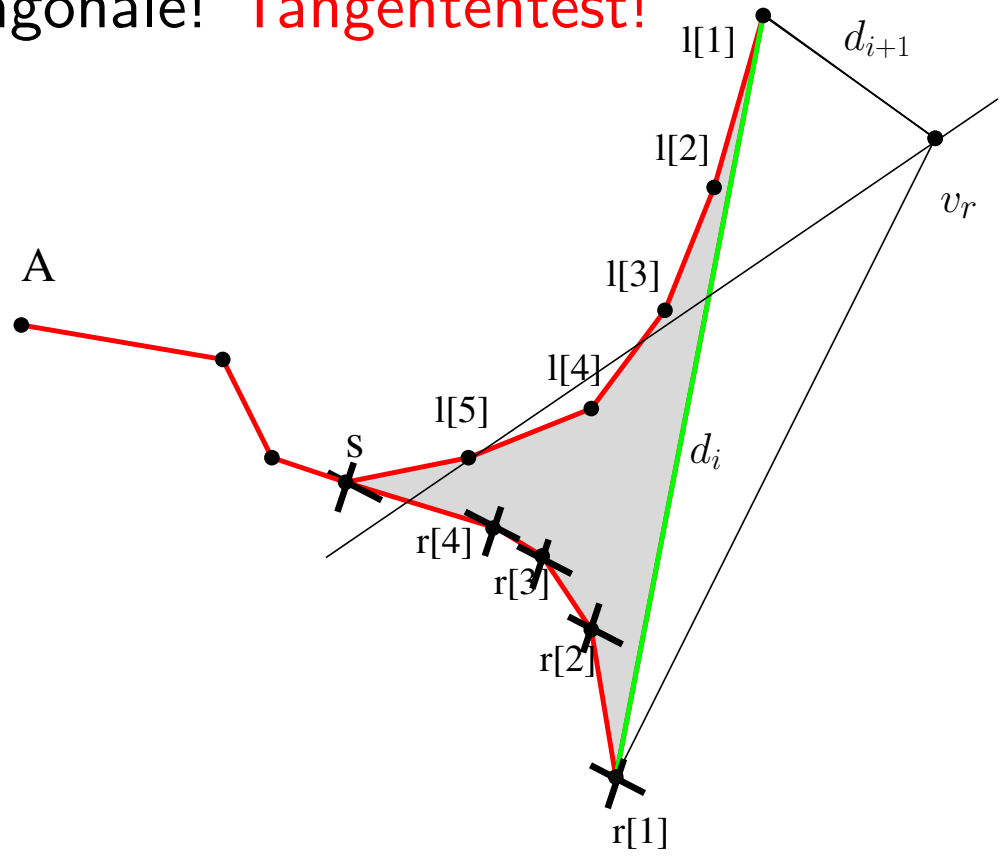
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest!



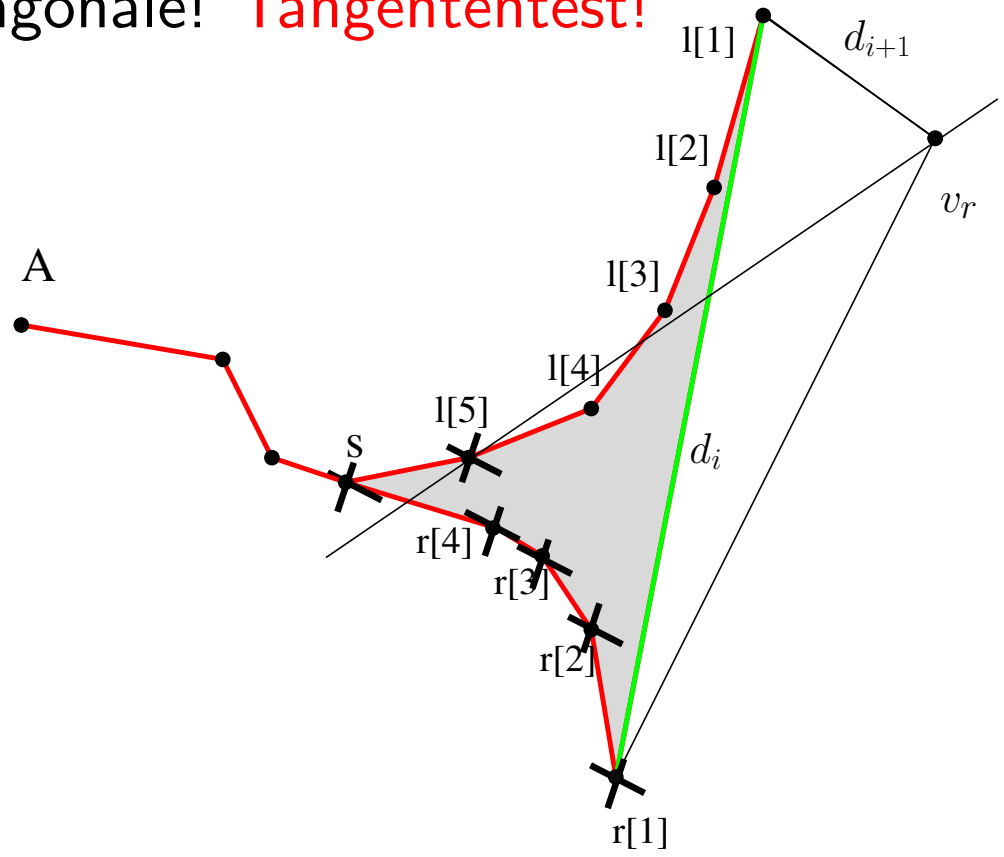
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest!



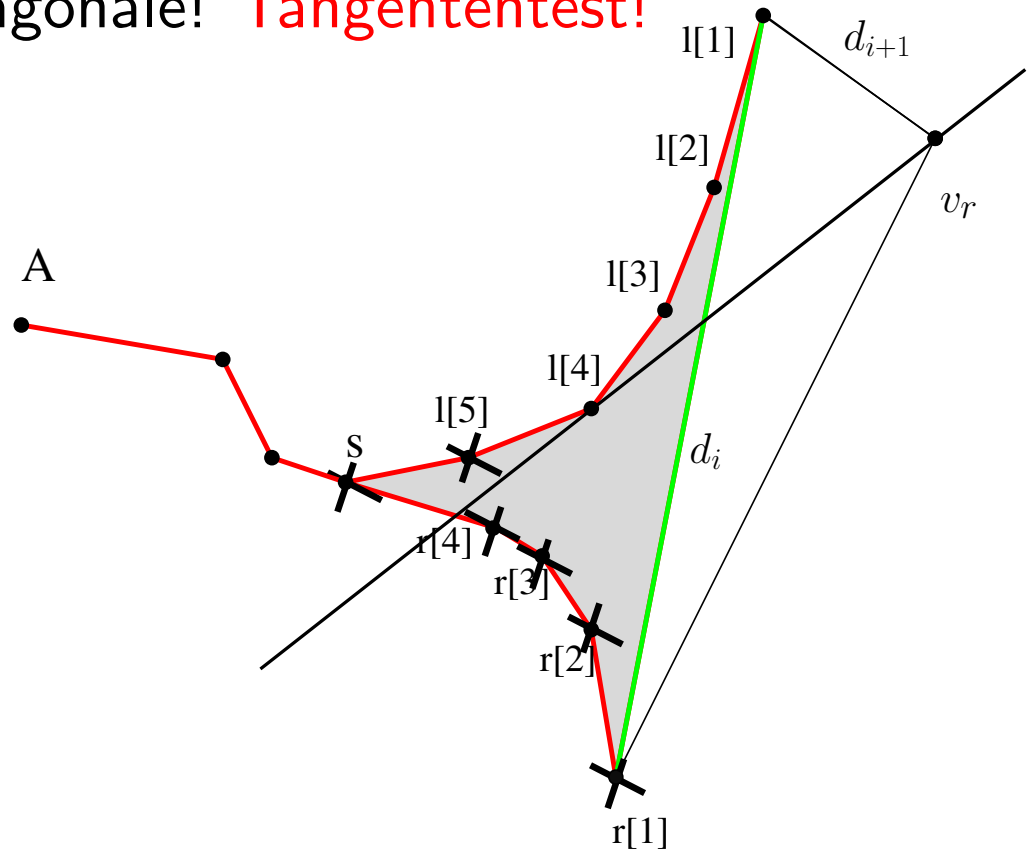
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest!



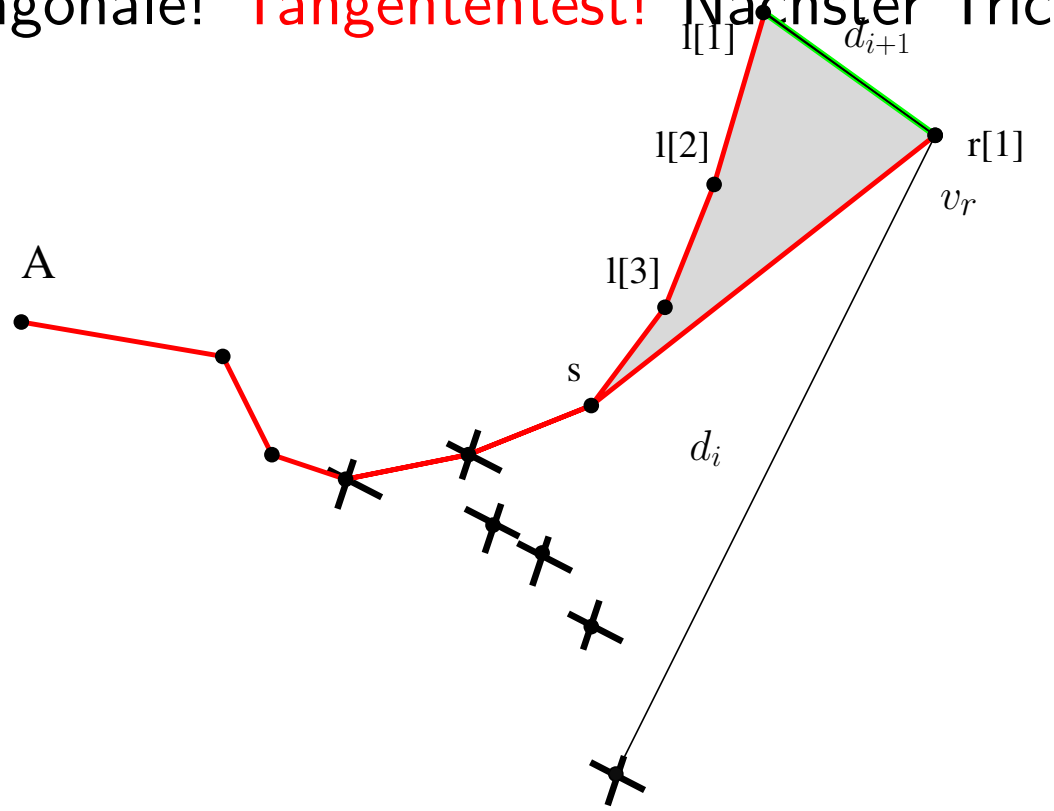
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest!



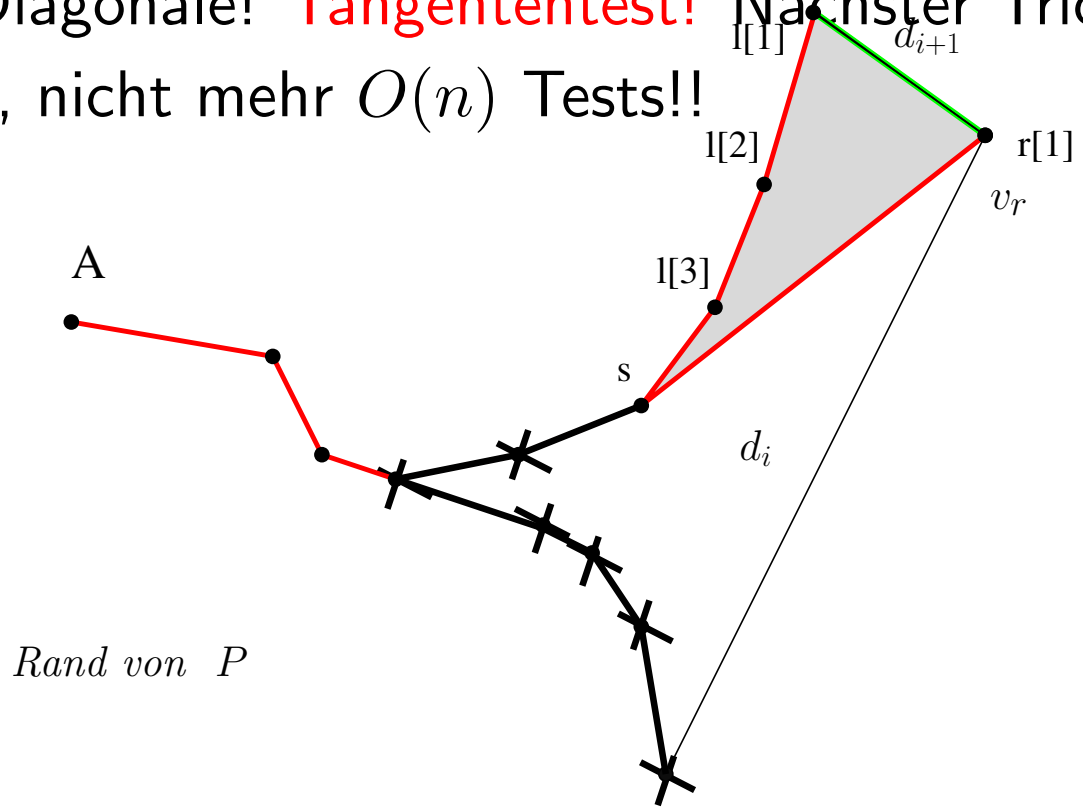
Verbleibt: Indukiver Schritt!

- Trichter Situation! DS: Zwei Ketten/Shortest Path!
- Nächste Diagonale! Tangententest! Nächster Trichter!



Verbleibt: Indukiver Schritt!

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



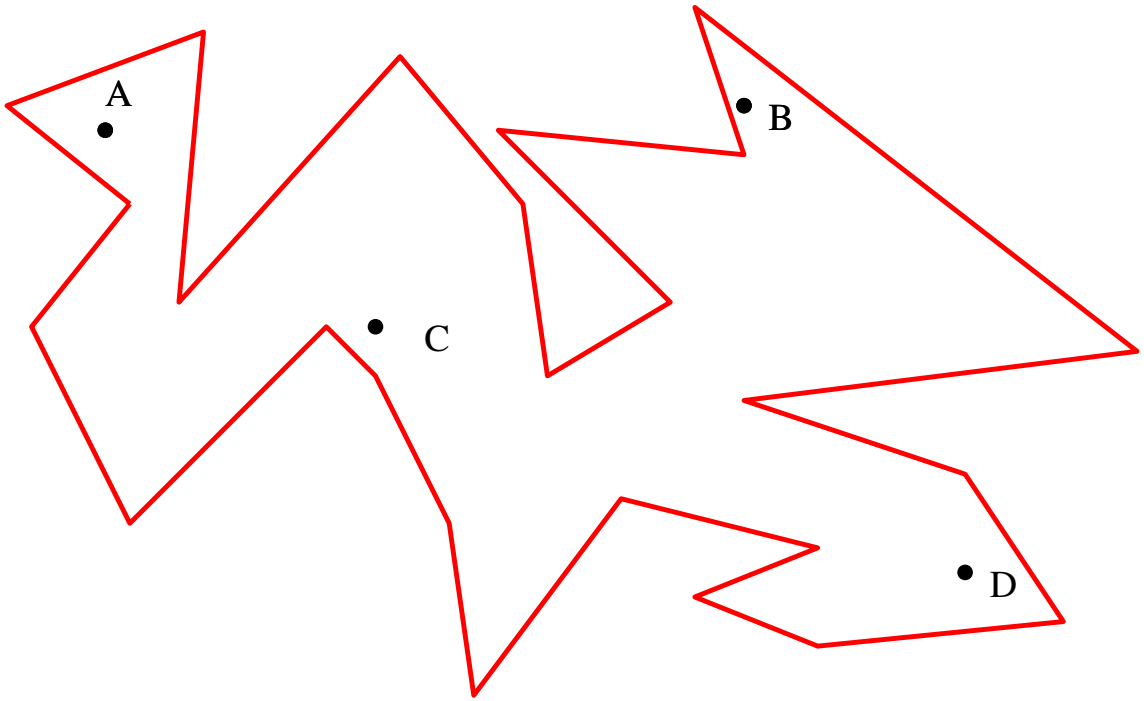
Zusammenfassung: Lee/Preparata: Shortest Path innerhalb P

- 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

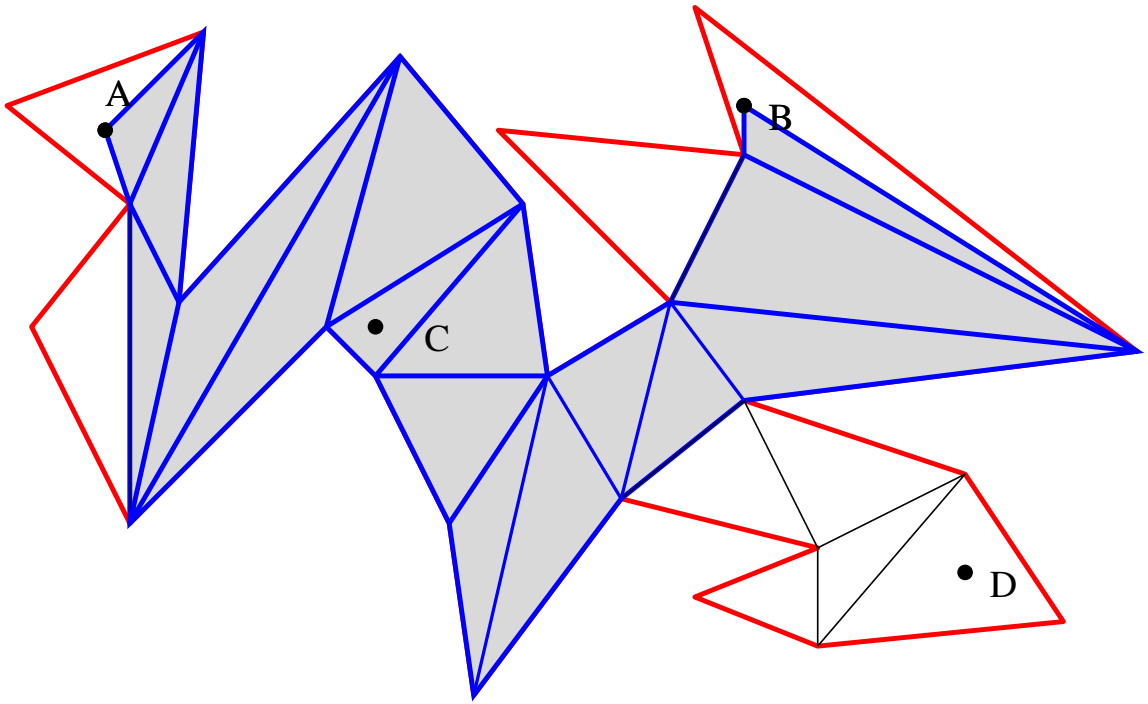
Mehrere Shortest Path Anfragen

- Viele Anfragen im Laufe der Zeit



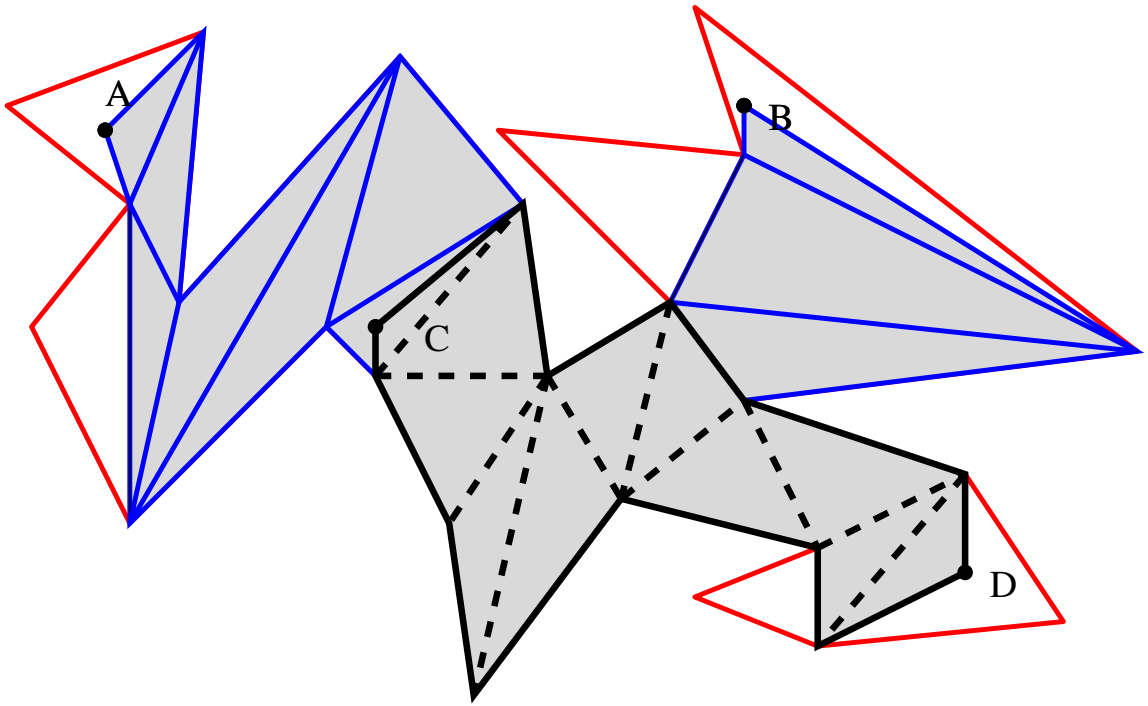
Mehrere Shortest Path Anfragen

- Viele Anfragen im Laufe der Zeit



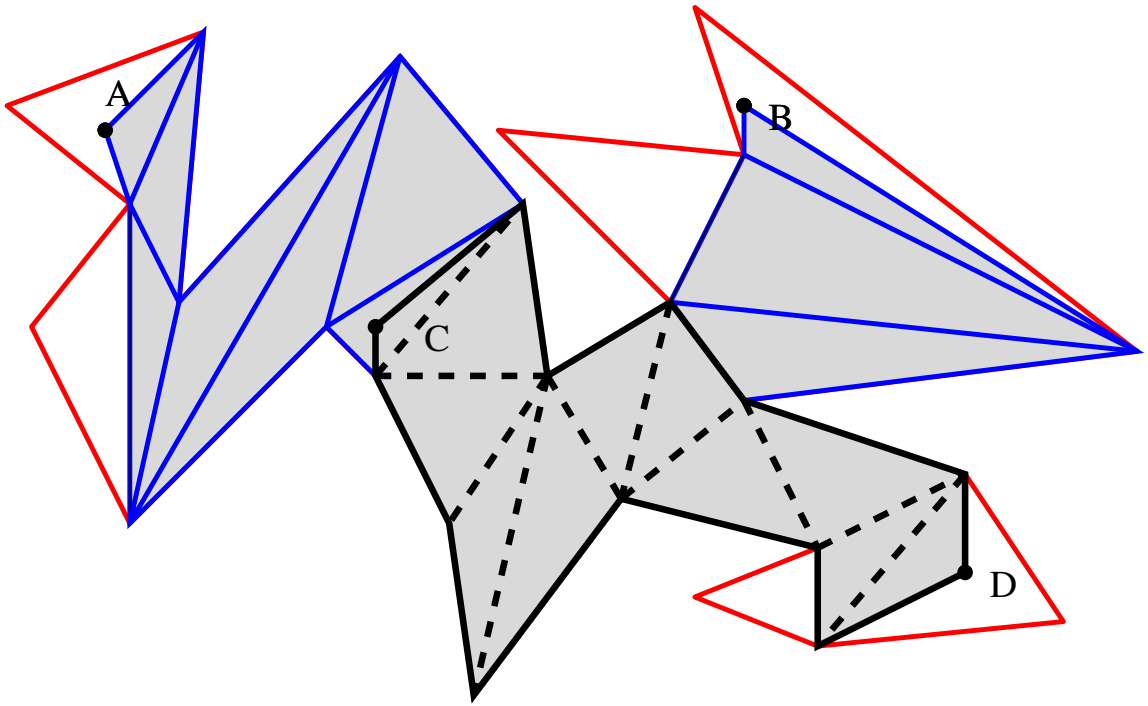
Mehrere Shortest Path Anfragen

- Viele Anfragen im Laufe der Zeit



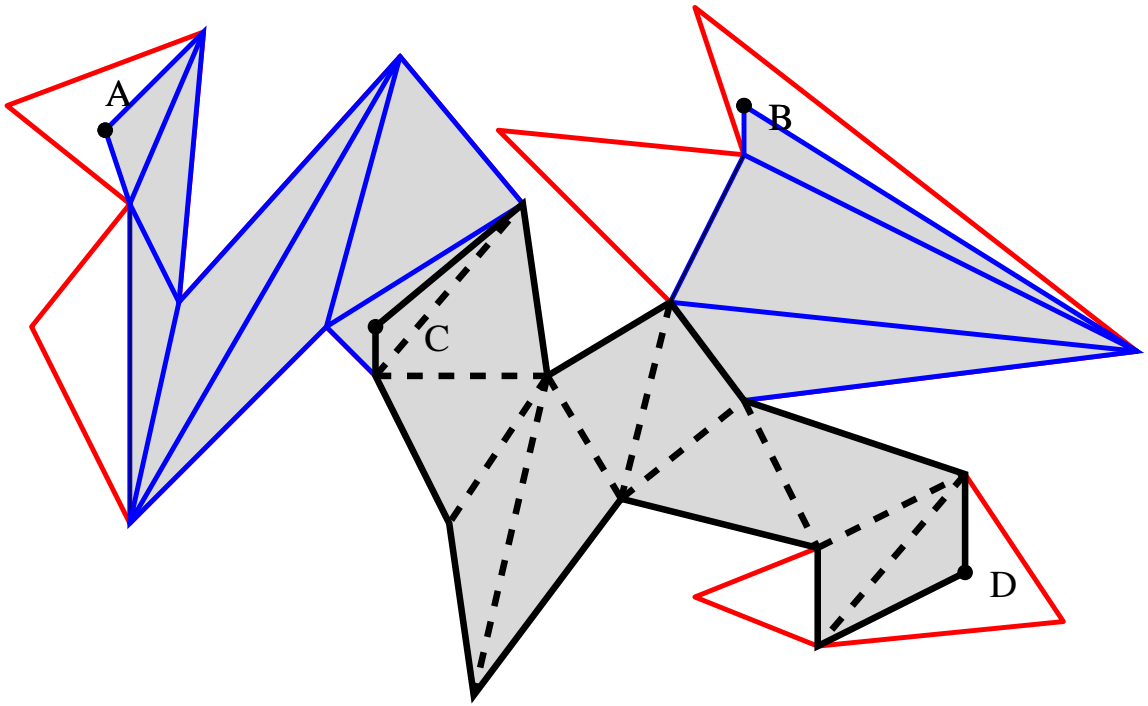
Mehrere Shortest Path Anfragen

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



Mehrere Shortest Path Anfragen

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



Mehrere Shortest Path Anfragen

Mehrere Shortest Path Anfragen

- Preprocessing:

Mehrere Shortest Path Anfragen

- Preprocessing: Für Kürzeste-Wege-Anfragen vorbereiten

Mehrere Shortest Path Anfragen

- Preprocessing: Für Kürzeste-Wege-Anfragen vorbereiten
- Struktur des Polygons in DS ablegen

Mehrere Shortest Path Anfragen

- Preprocessing: Für Kürzeste-Wege-Anfragen vorbereiten
- Struktur des Polygons in DS ablegen
- Komplexität in $O(n)$,

Mehrere Shortest Path Anfragen

- Preprocessing: Für Kürzeste-Wege-Anfragen vorbereiten
- Struktur des Polygons in DS ablegen
- Komplexität in $O(n)$, Aufbau in $O(n)$

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 $O(\log n)$

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. $O(\log n + k)$

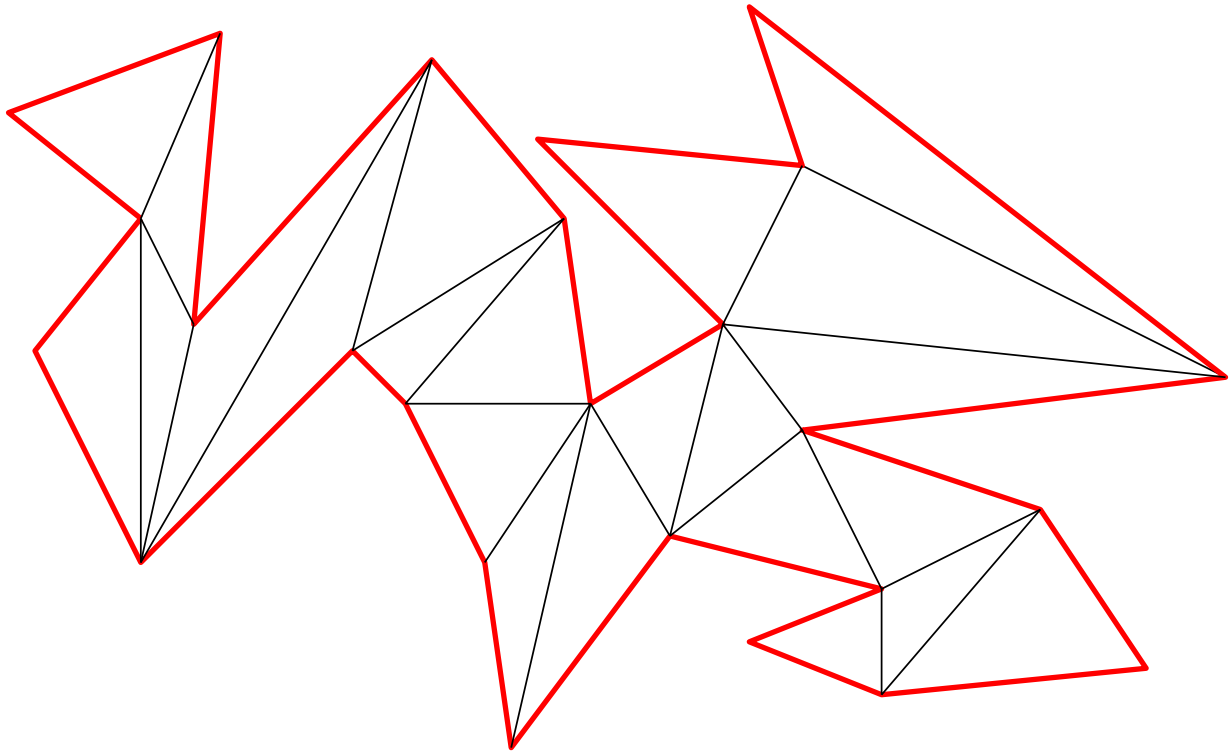
Mehrere Shortest Path Anfragen

- 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)

Preprocessing für P! (Guibas/Hershberger)

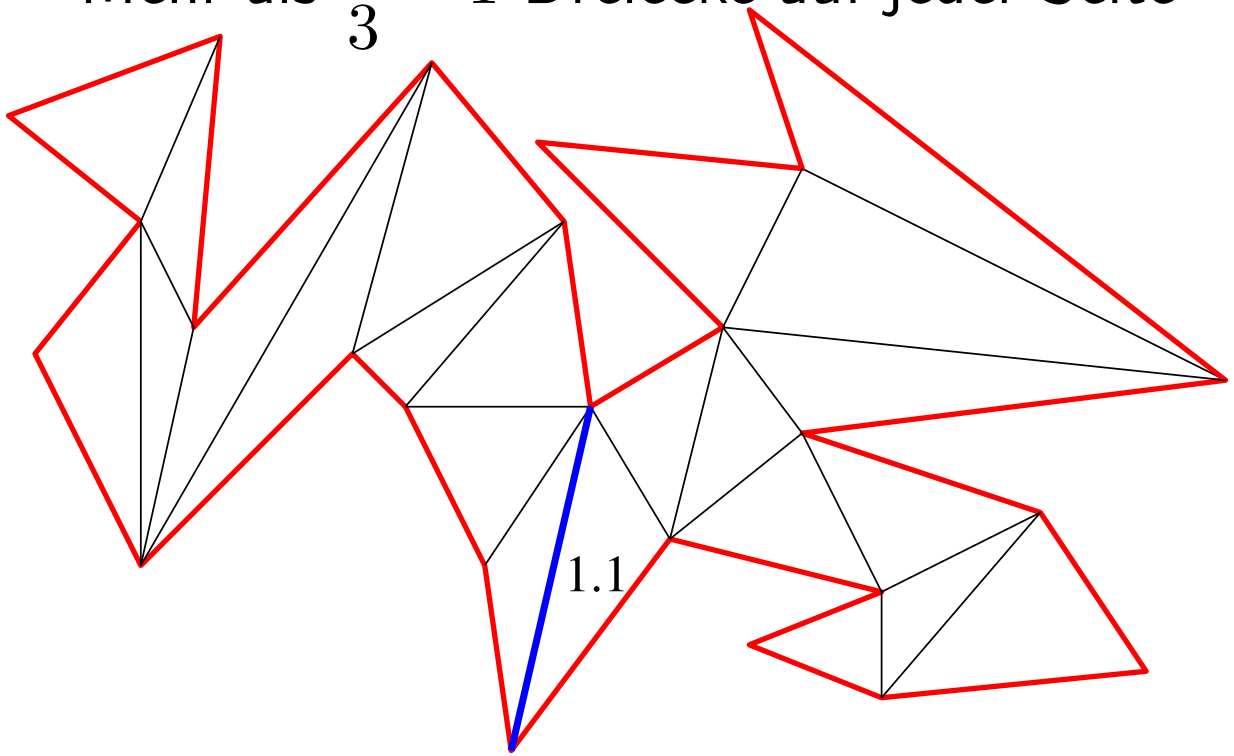
Cutting theorem (Theorem 1.12) von Chazelle '82: In jeder Triangulation T existiert immer eine Diagonale für eine balancierte Einteilung von T!



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!

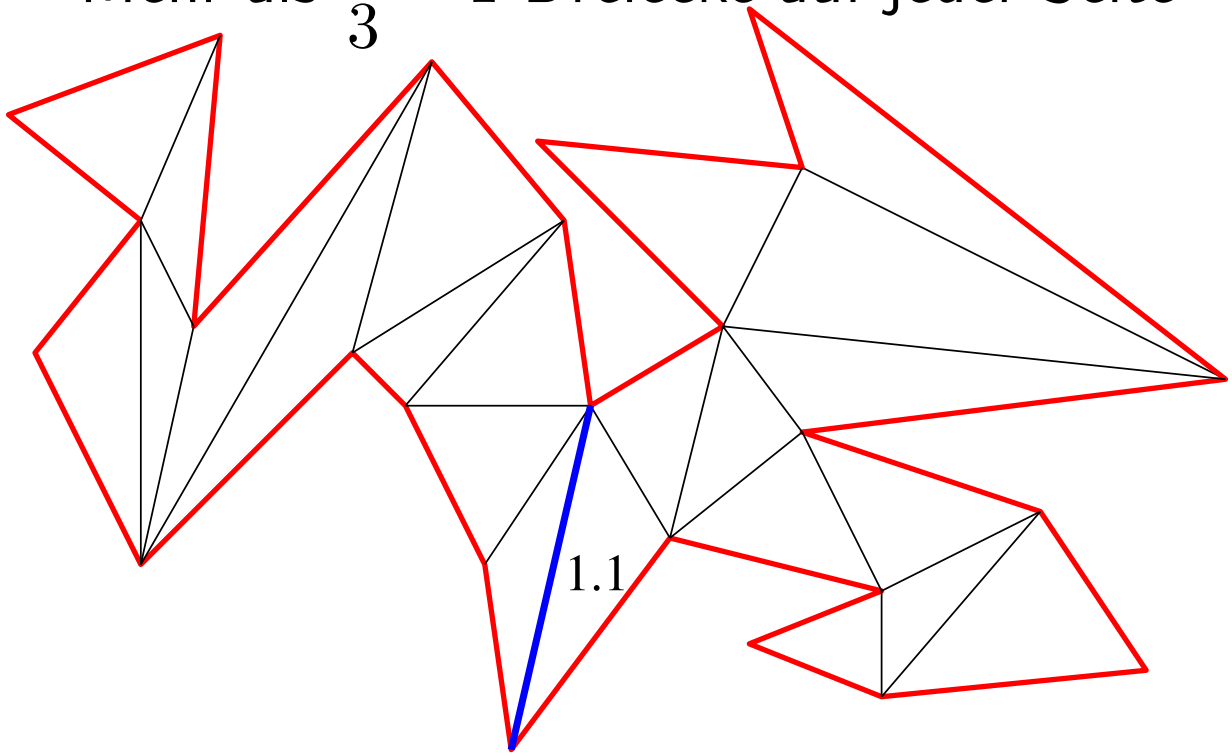
Mehr als $\frac{n}{3} - 1$ Dreiecke auf jeder Seite



Preprocessing für P! (Guibas/Hershberger)

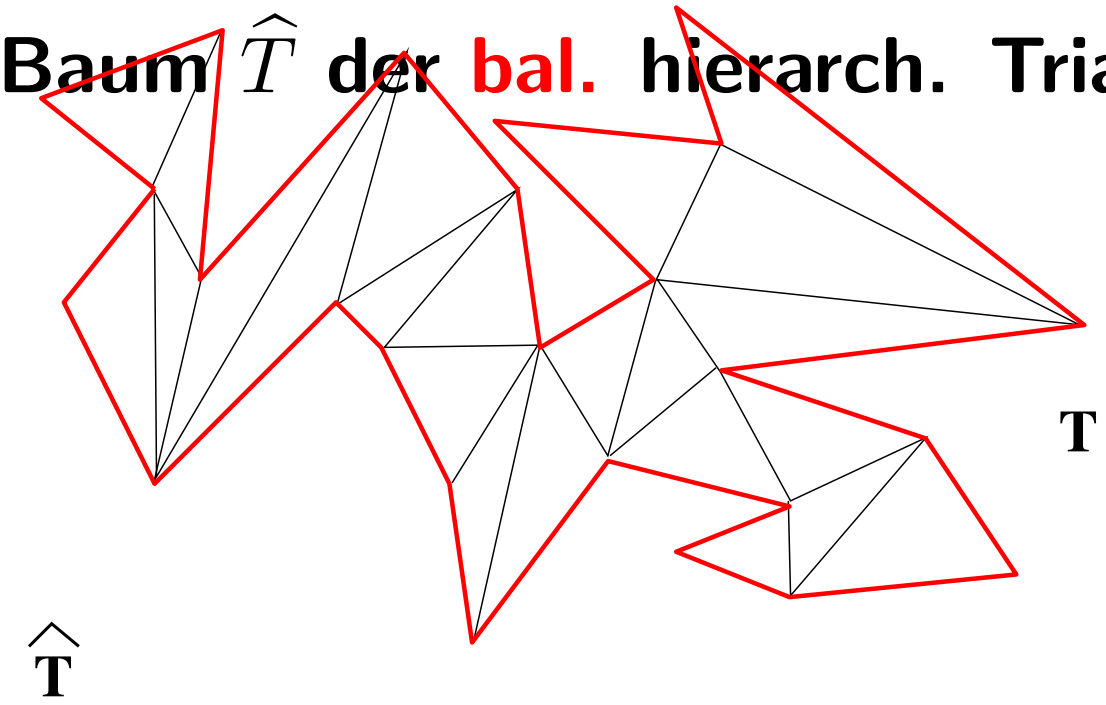
Cutting theorem (Theorem 1.12) von Chazelle '82: In jeder Triangulation T existiert immer eine Diagonale für eine balancierte Einteilung von T!

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

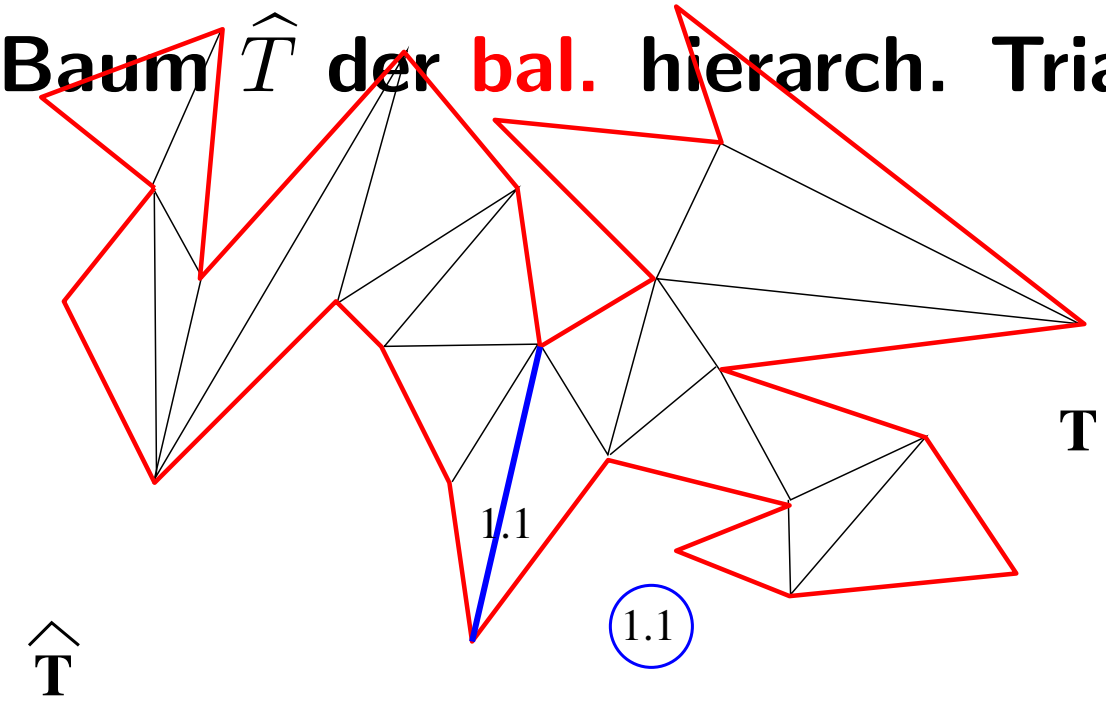


Baum $\widehat{T}$ der bal. hierarch. Triangulation

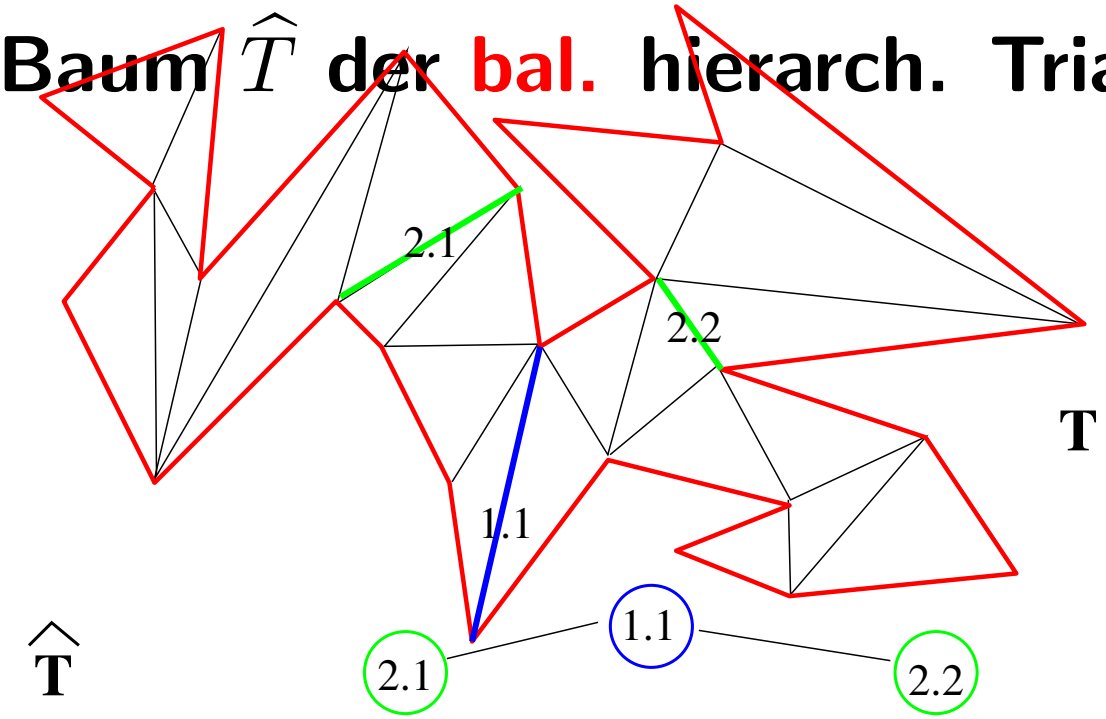
Baum $\widehat{T}$ der bal. hierarch. Triangulation



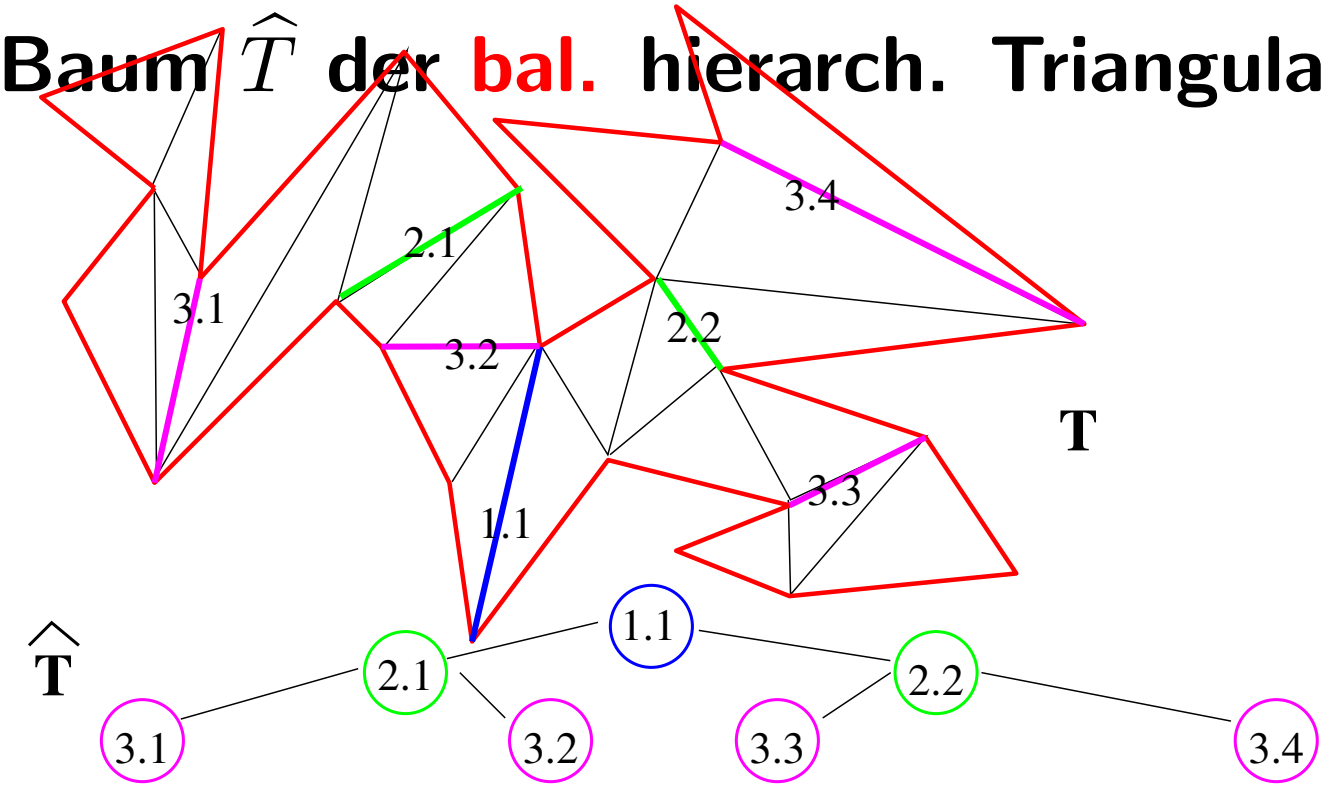
Baum $\widehat{T}$ der bal. hierarch. Triangulation



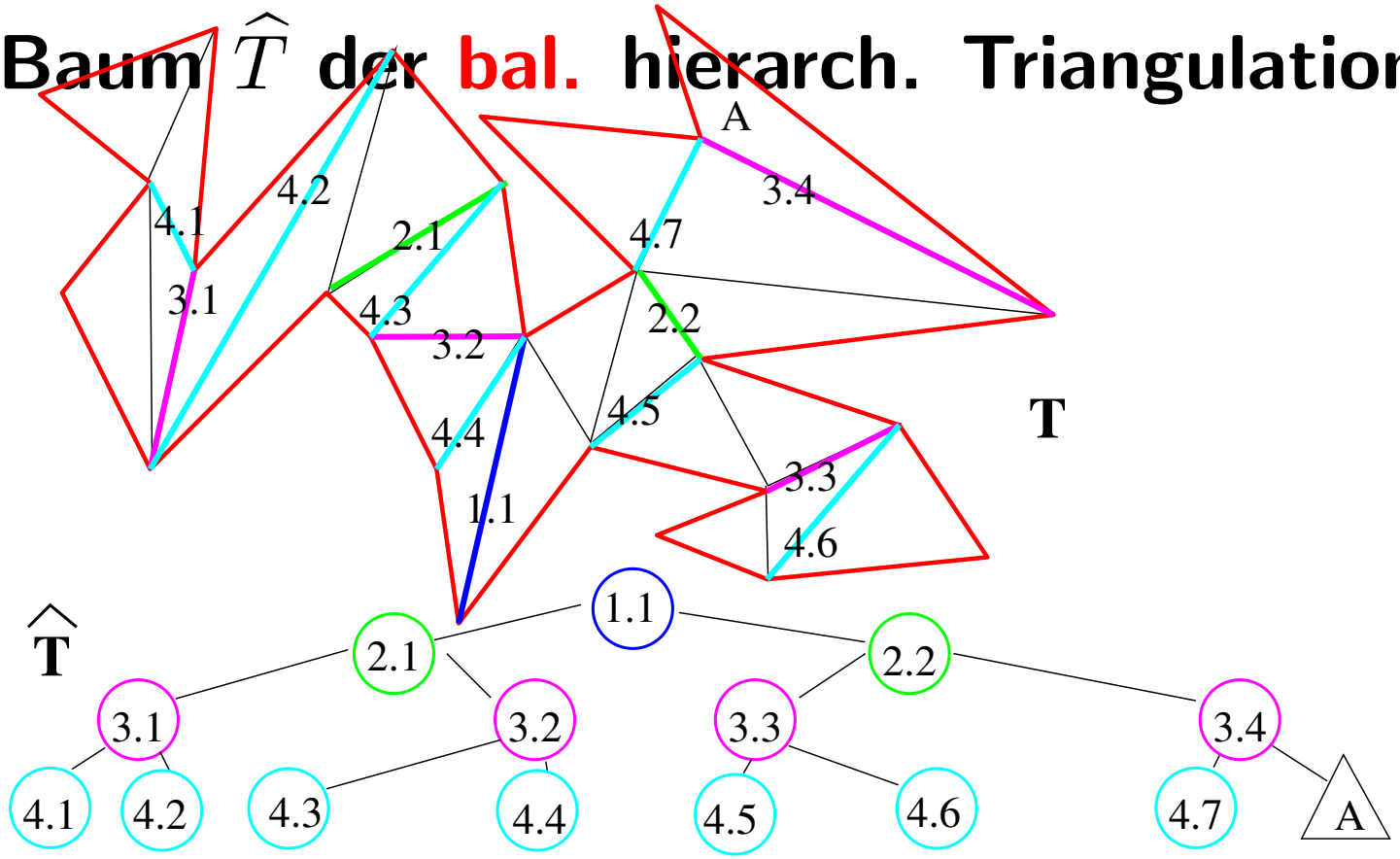
Baum $\widehat{T}$ der bal. hierarch. Triangulation



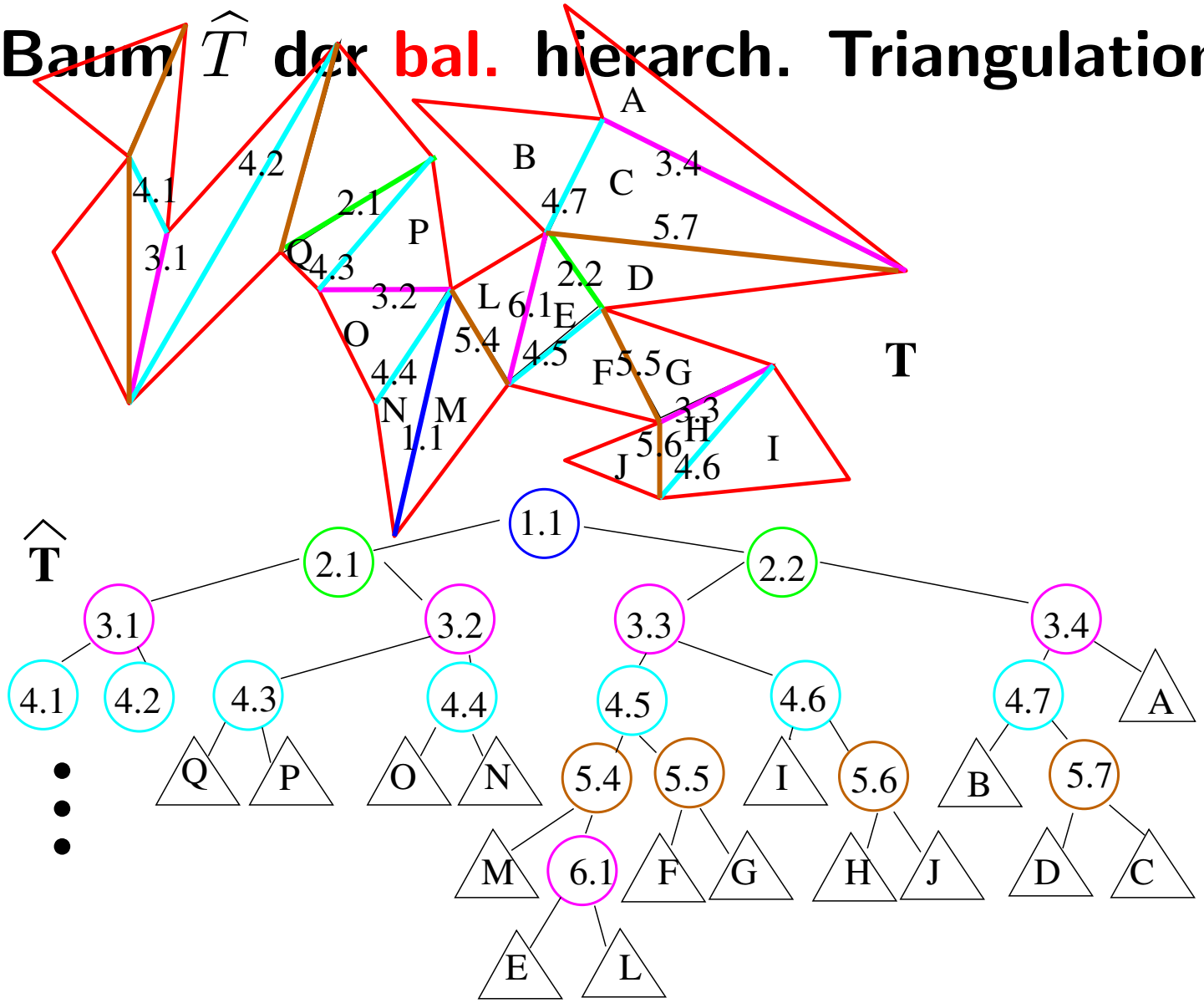
Baum $\widehat{T}$ der bal. hierarch. Triangulation



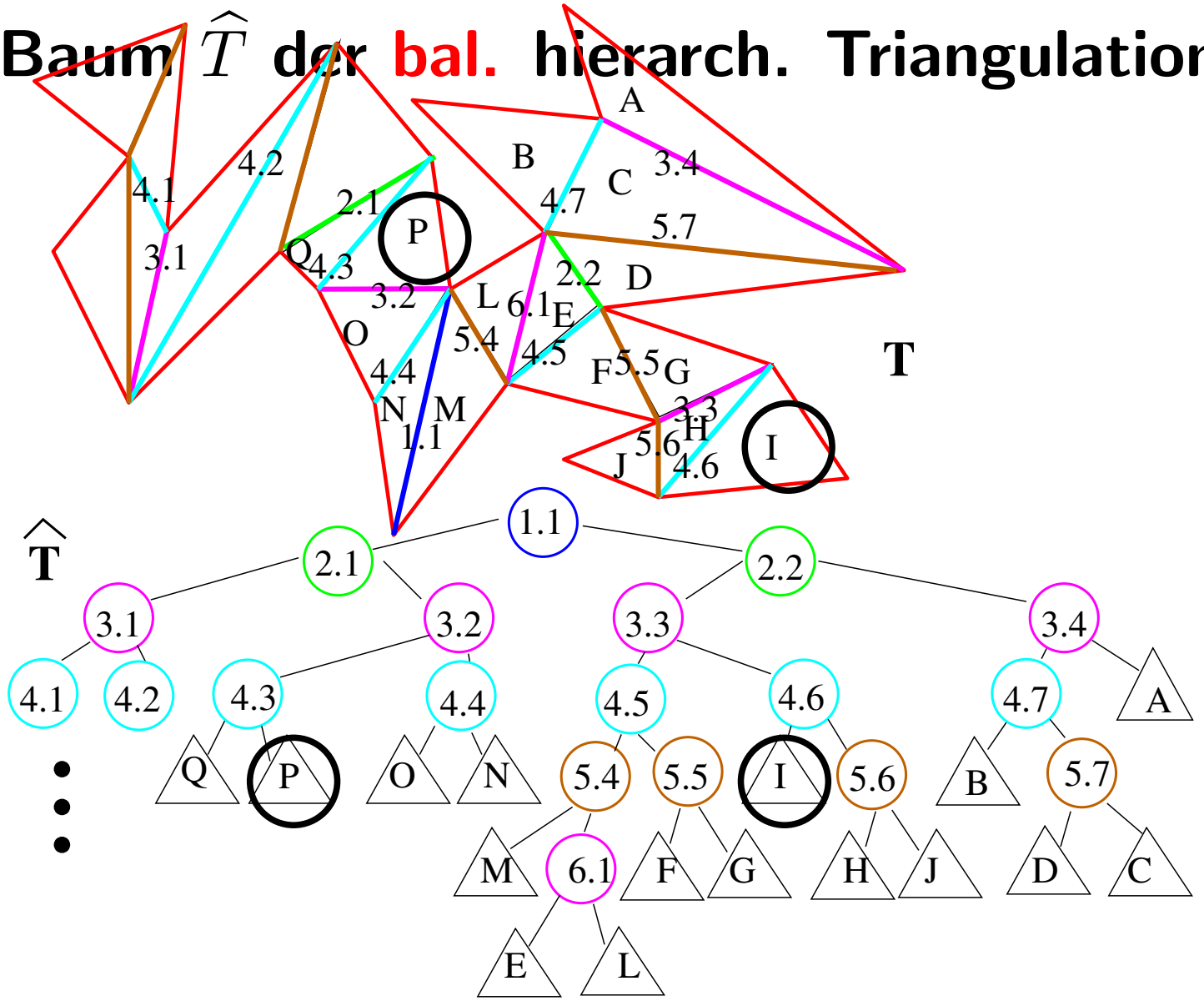
Baum $\widehat{T}$ der bal. hierarch. Triangulation



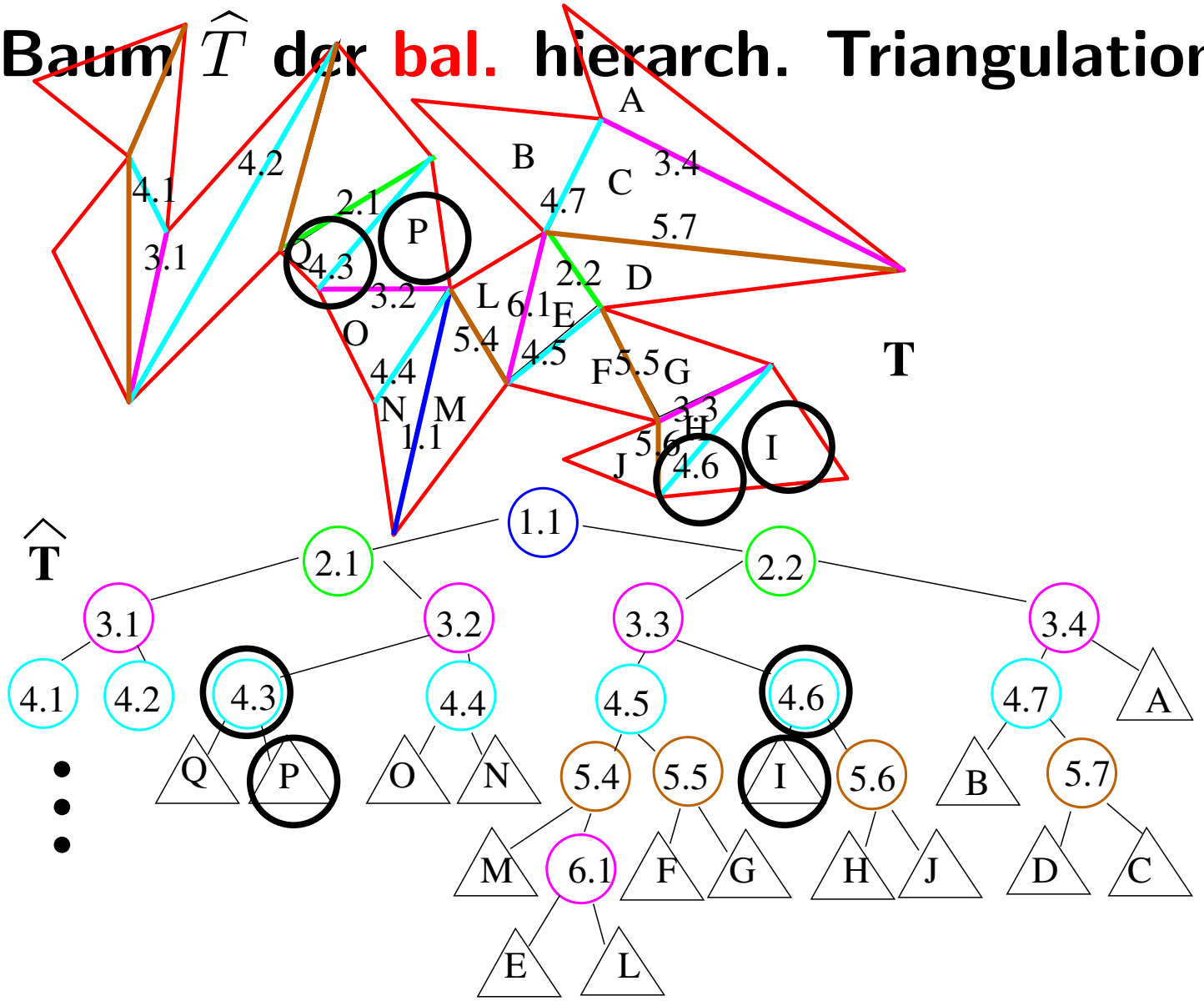
Baum $\widehat{T}$ der bal. hierarch. Triangulation



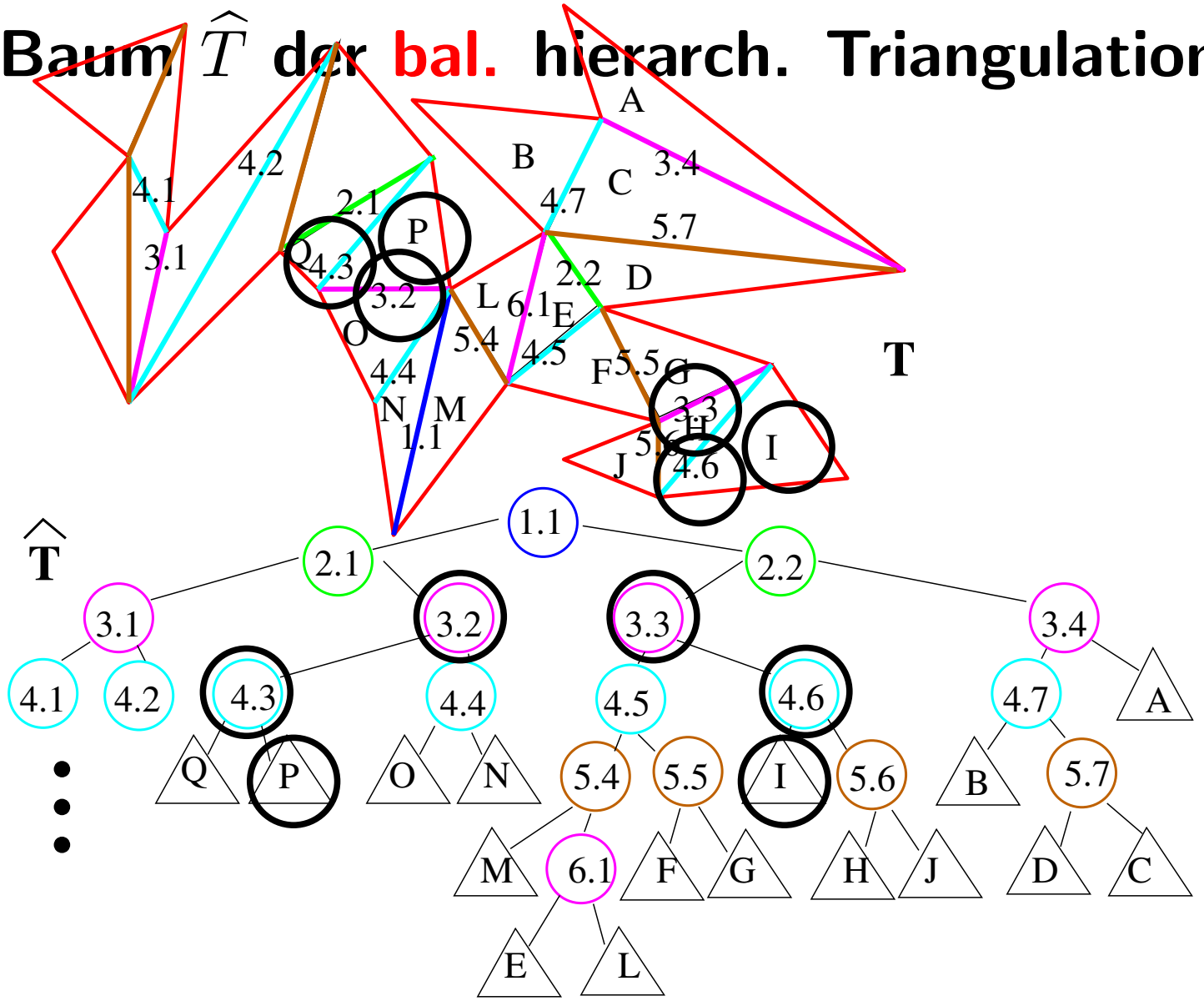
Baum $\widehat{T}$ der bal. hierarch. Triangulation



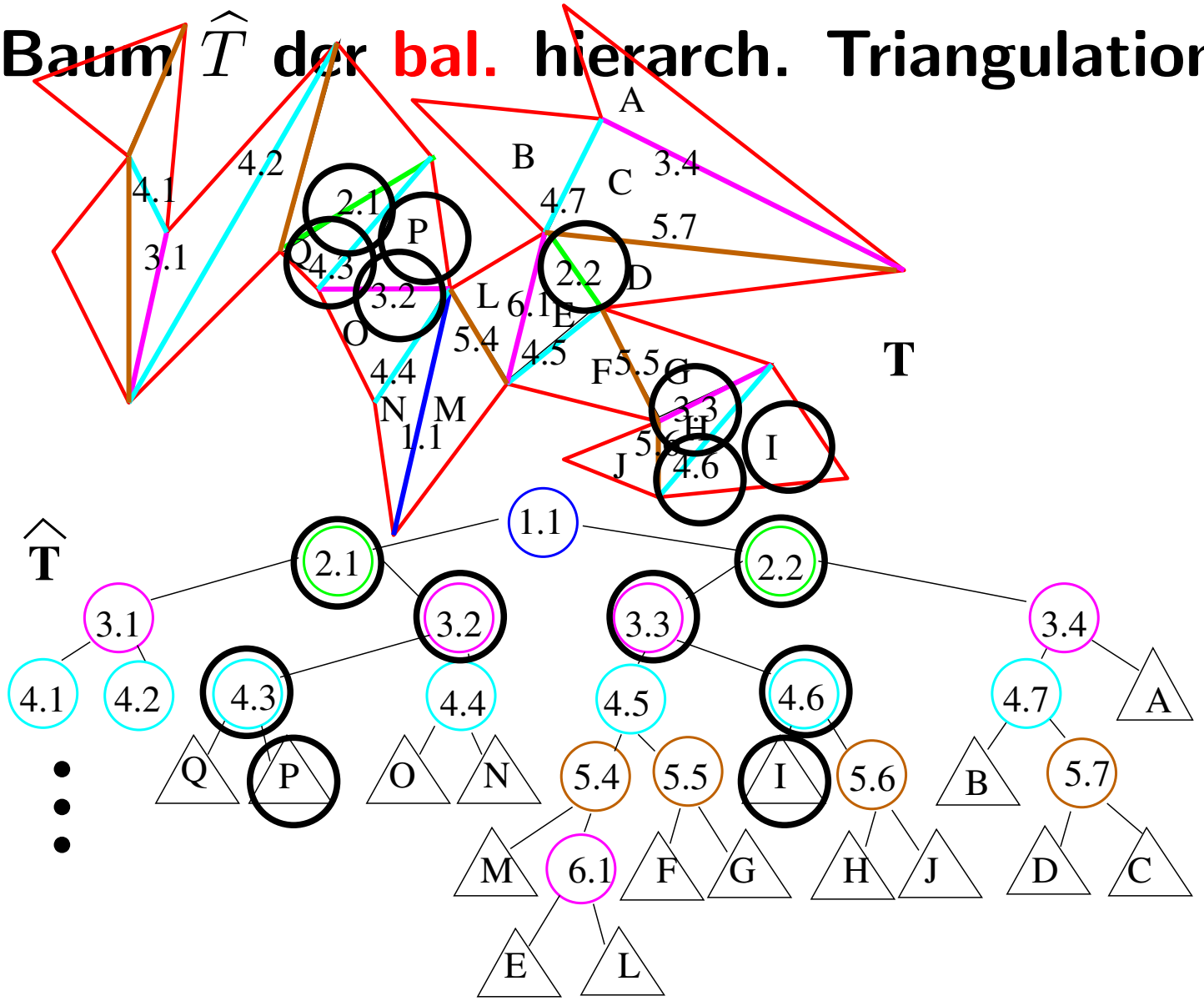
Baum $\widehat{T}$ der bal. hierarch. Triangulation



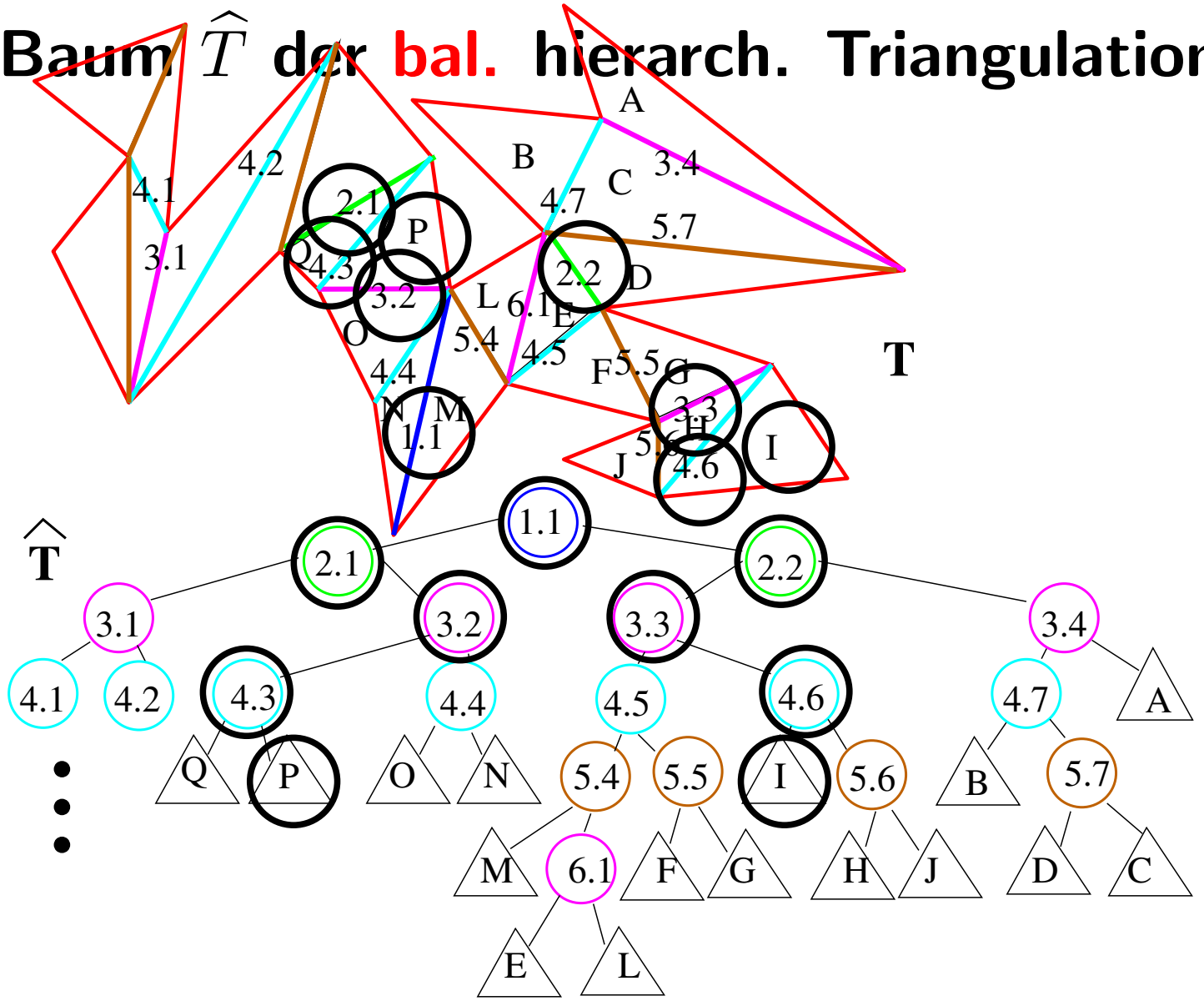
Baum $\widehat{T}$ der bal. hierarch. Triangulation



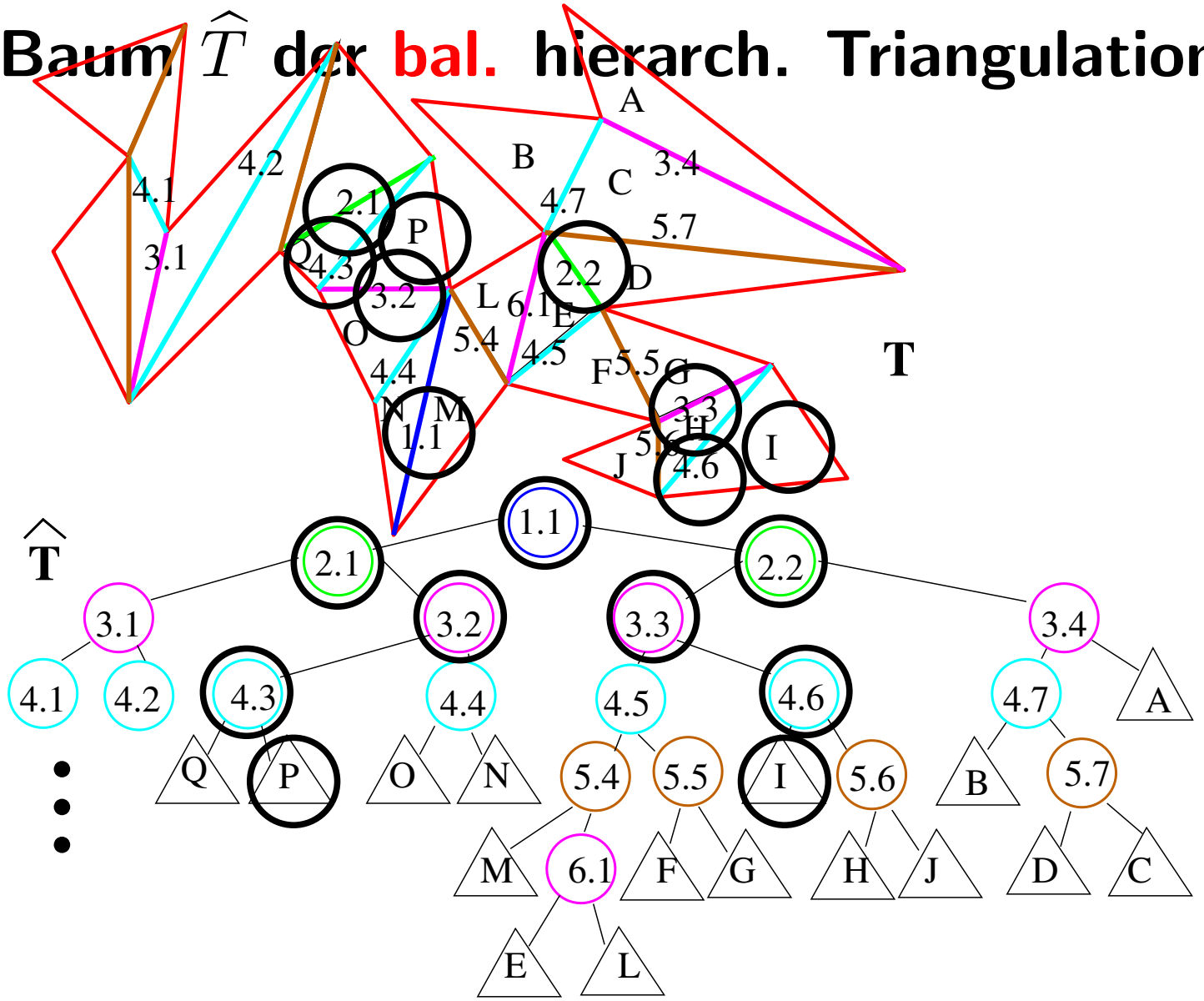
Baum $\widehat{T}$ der bal. hierarch. Triangulation



Baum $\widehat{T}$ der bal. hierarch. Triangulation

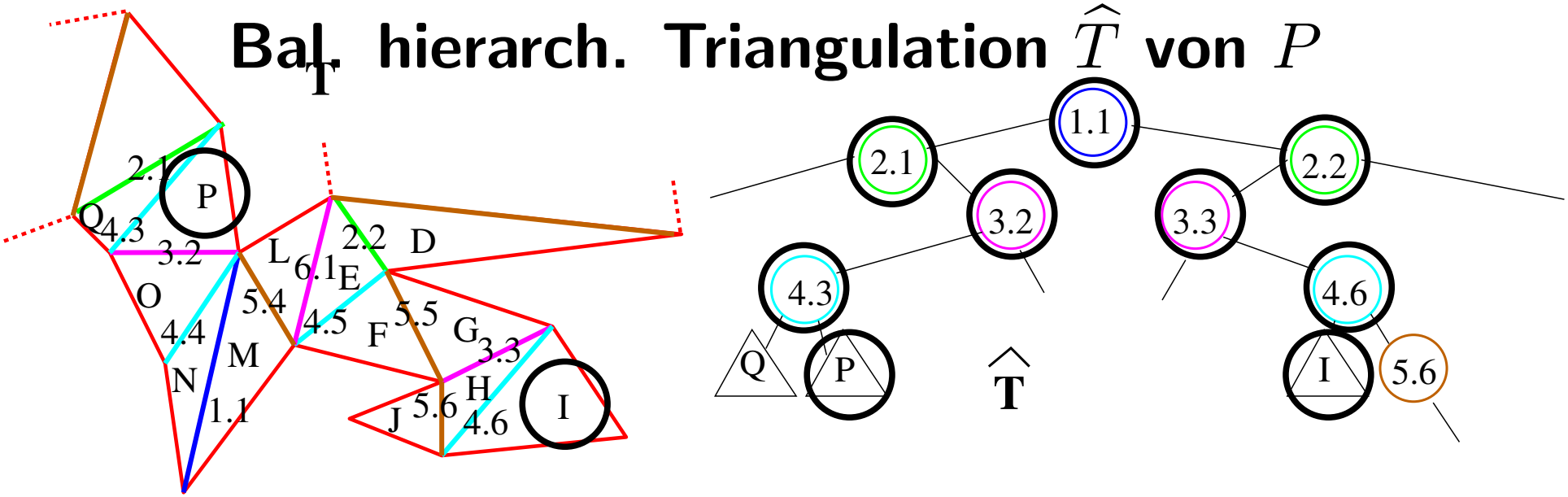


Baum $\widehat{T}$ der bal. hierarch. Triangulation



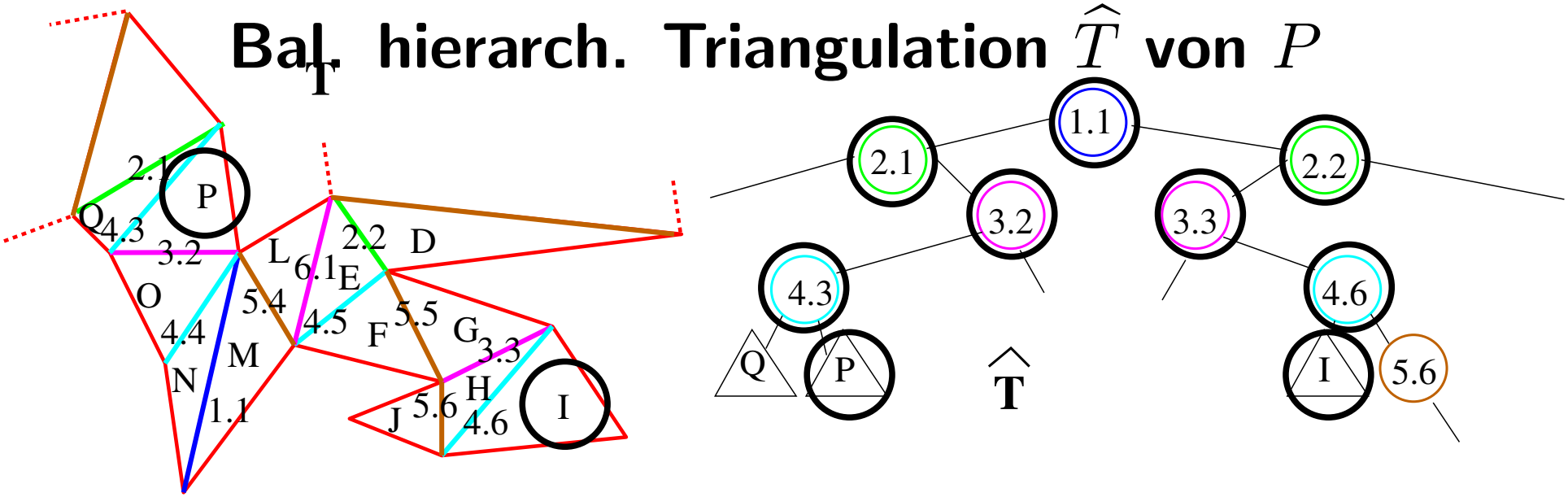
Bal. hierarch. Triangulation $\widehat{T}$ von P

Bal. hierarch. Triangulation $\widehat{T}$ von P



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

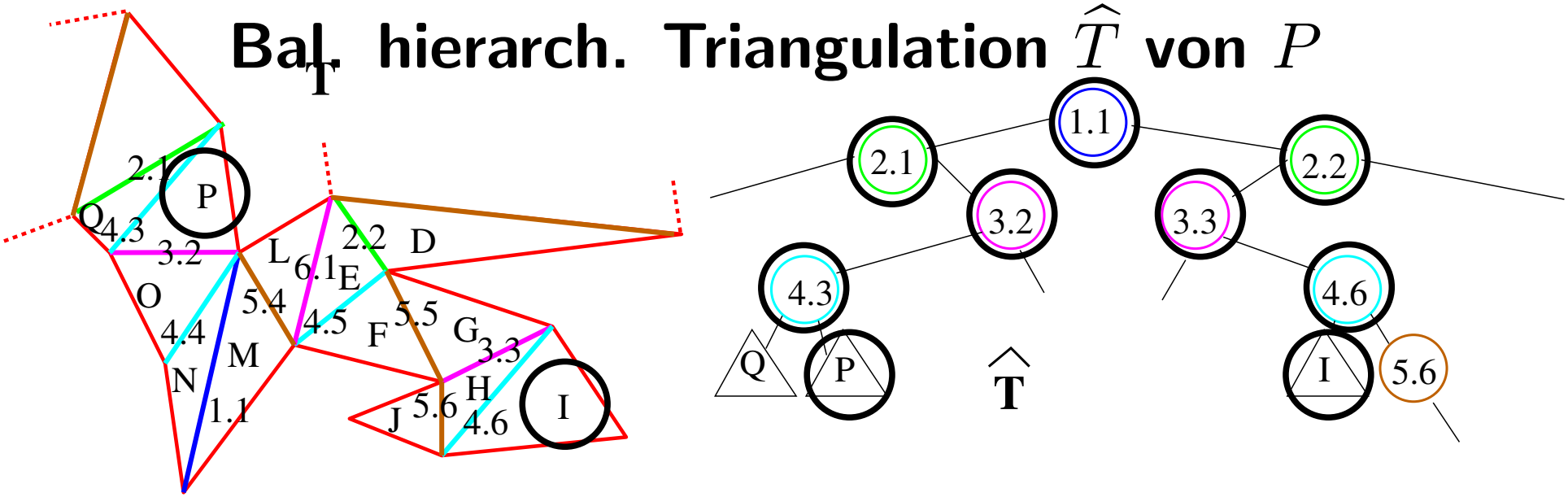
Bal. hierarch. Triangulation $\widehat{T}$ von P



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

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

Bal. hierarch. Triangulation $\widehat{T}$ von P

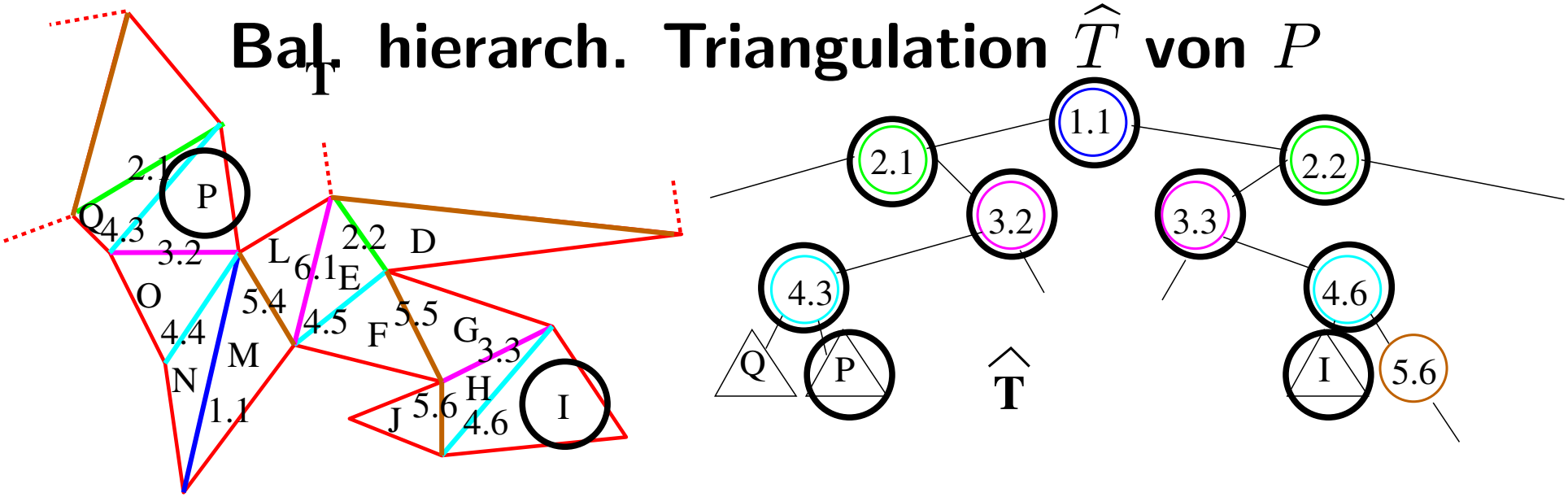


Kürzester Weg zwischen Dreiecken in T ($O(n)$) oder $\widehat{T}$ ($O(\log n)$)

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

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

Bal. hierarch. Triangulation $\widehat{T}$ von P



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

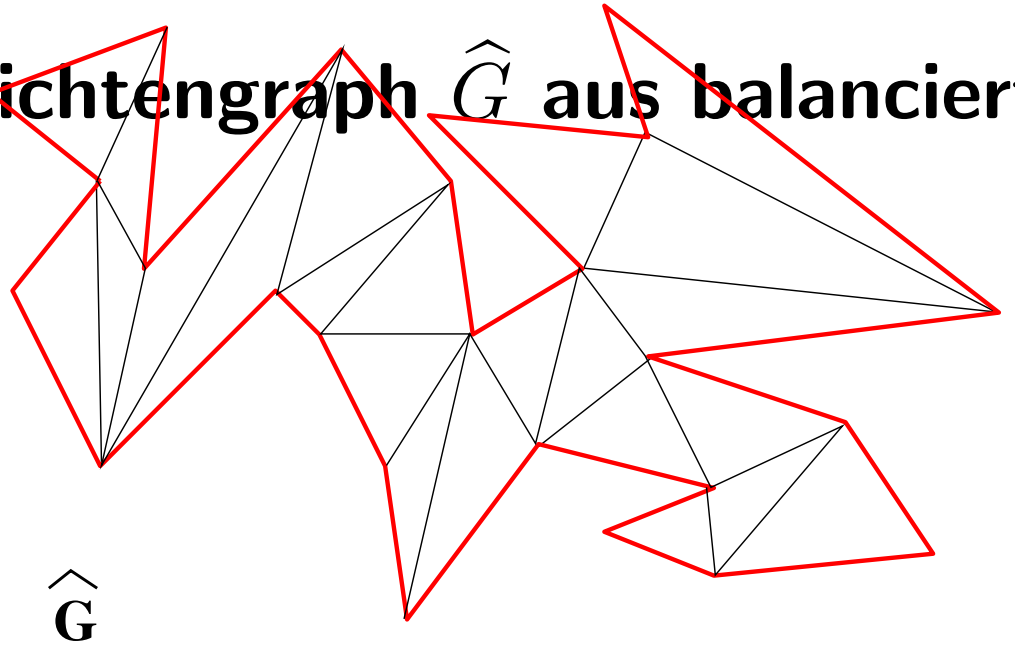
T:	P		(3.2)	(4.4)	(1.1)	(5.4)	(6.1)	(4.5)	(5.5)	(3.3)	(4.6)	I
$\widehat{T}$:	P	(4.3)	(3.2)	(2.1)	(1.1)		(2.2)			(3.3)	(4.6)	I

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

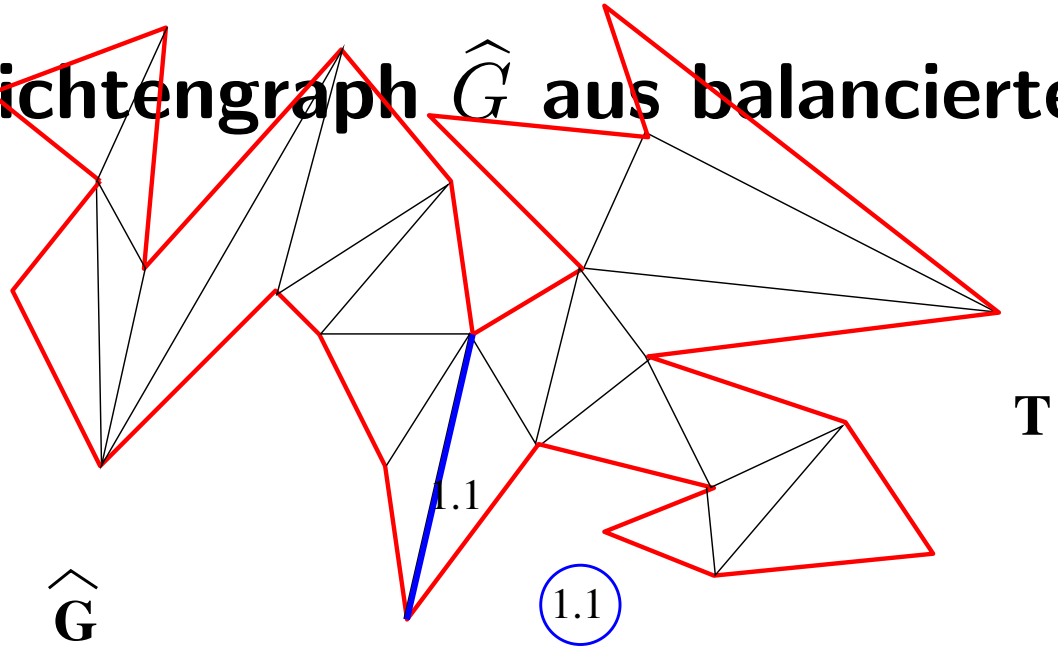
II) Fülle die *Löcher*: $(3.2) \Rightarrow (1.1)$ und $(1.1) \Rightarrow (3,3)$

Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$

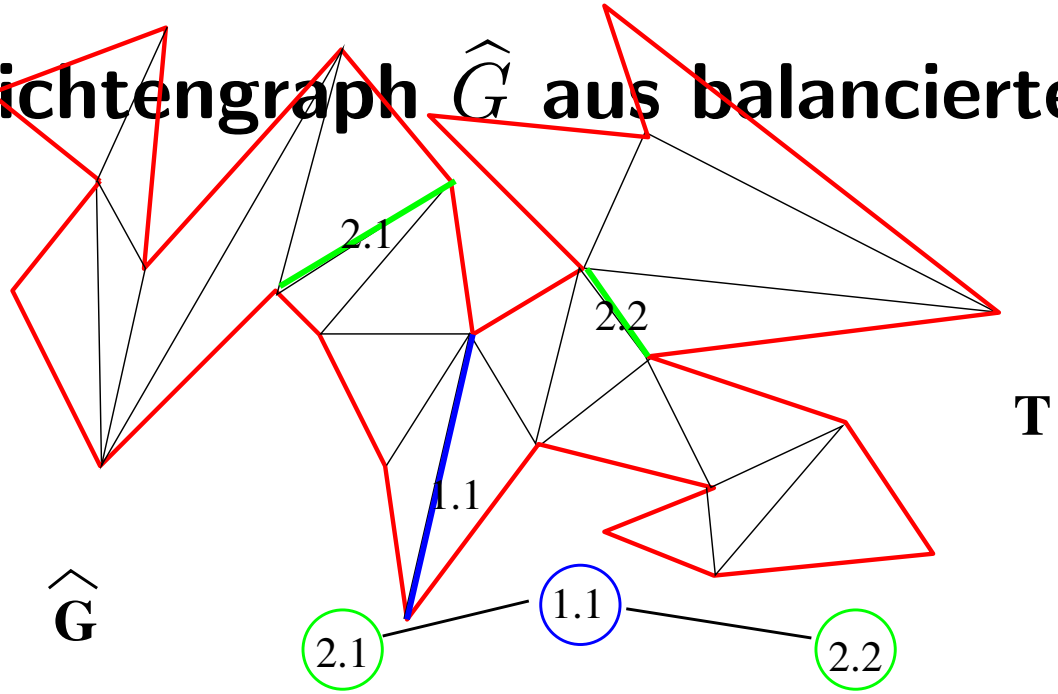
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



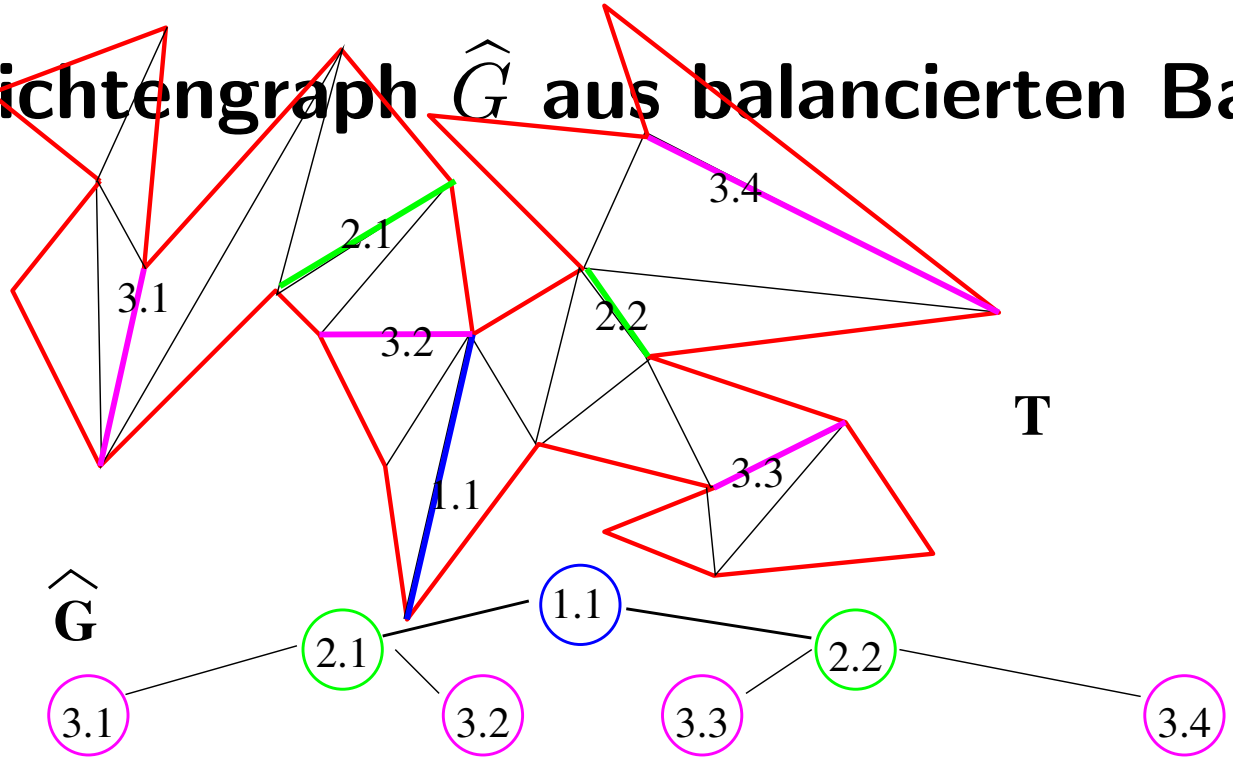
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



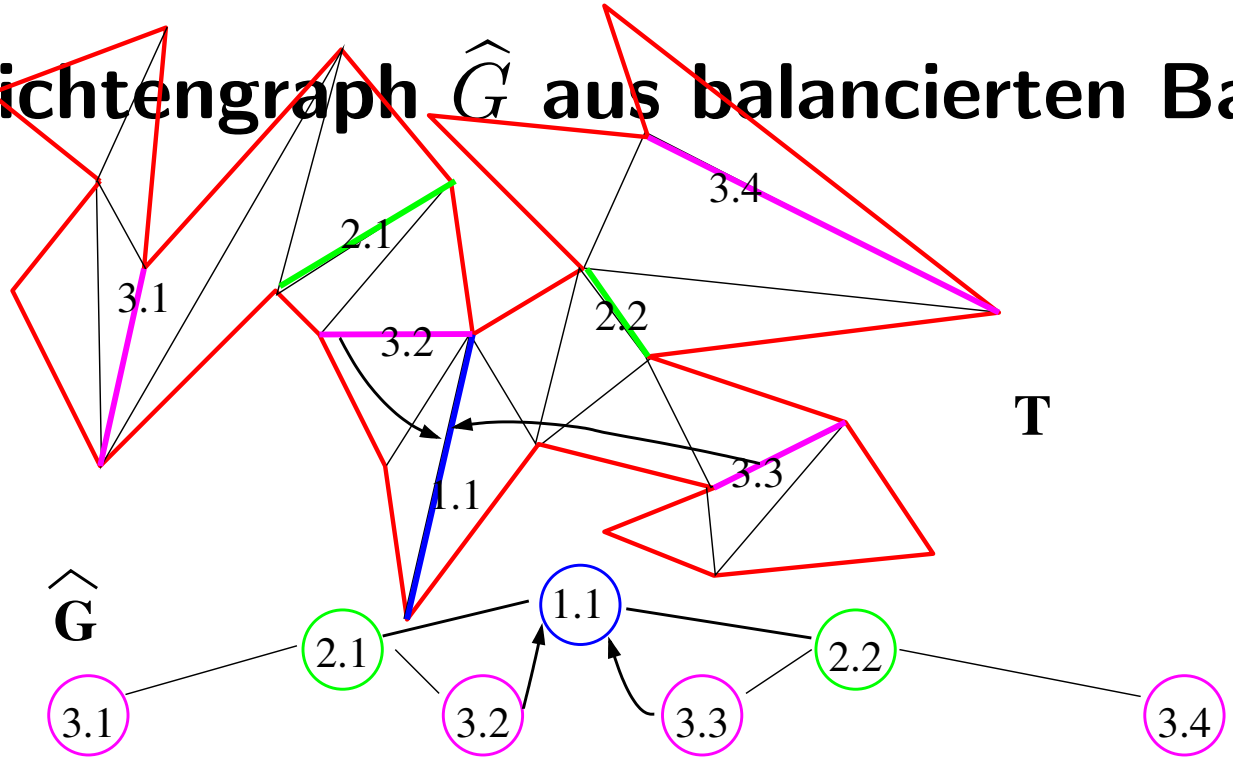
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



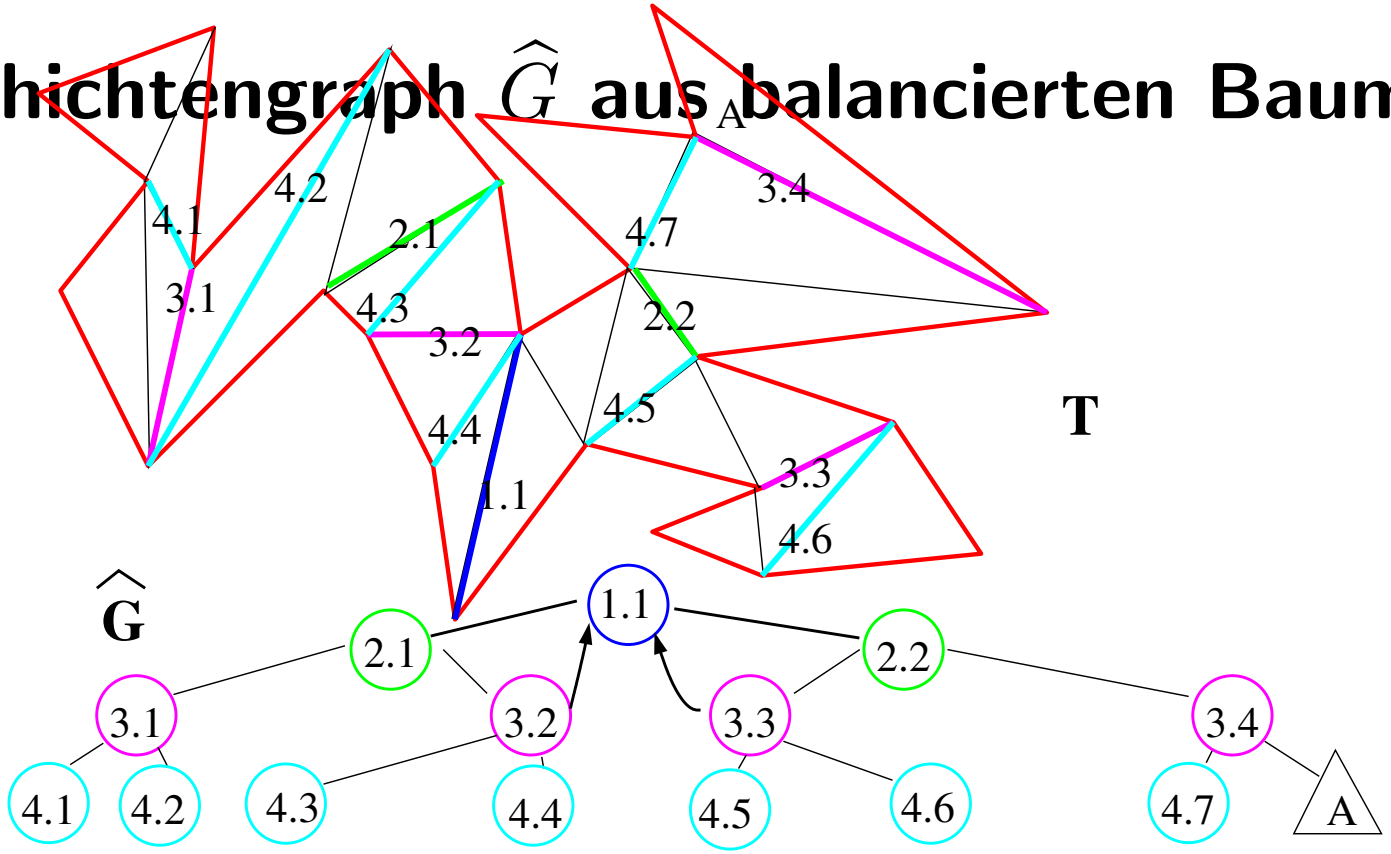
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



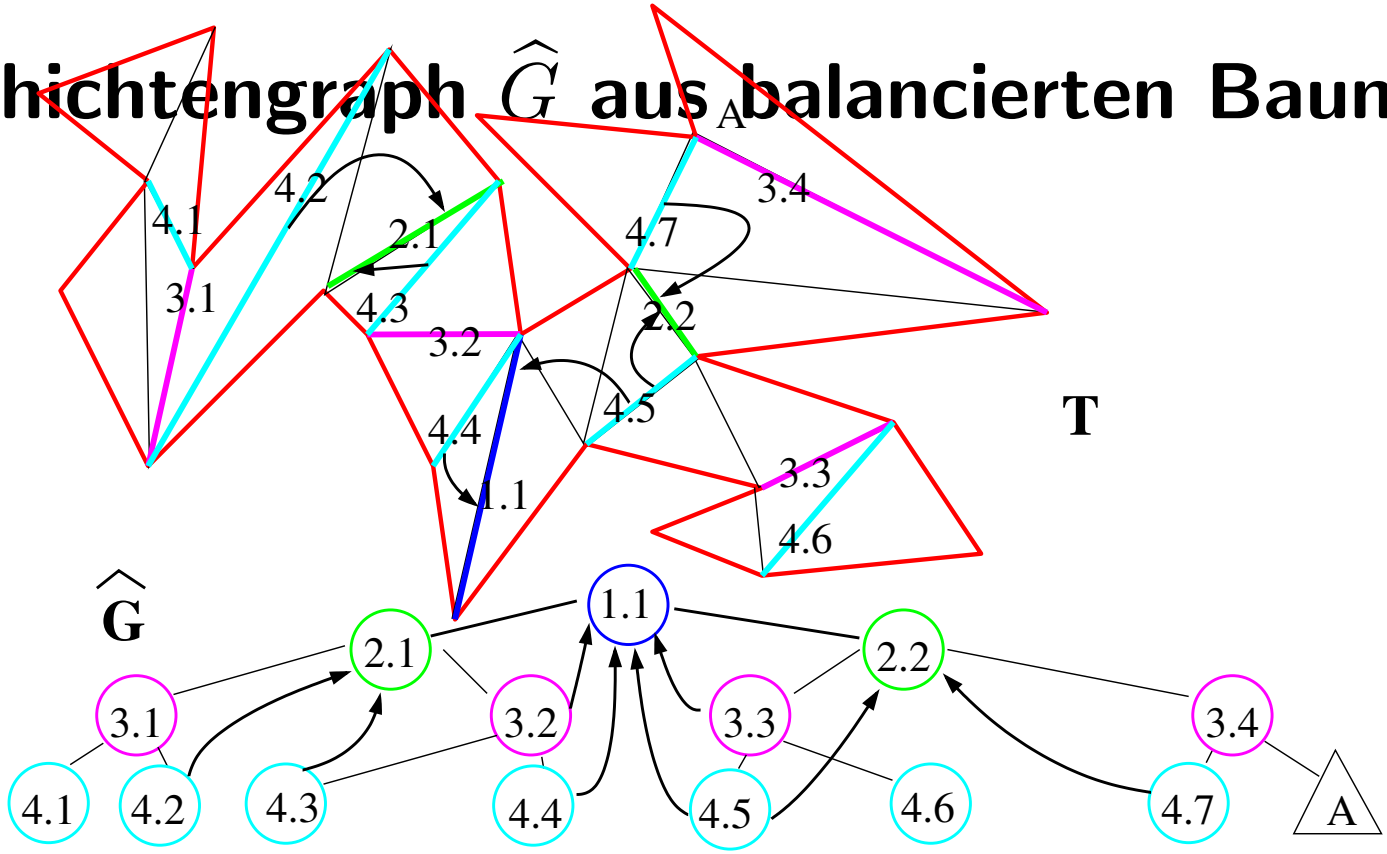
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



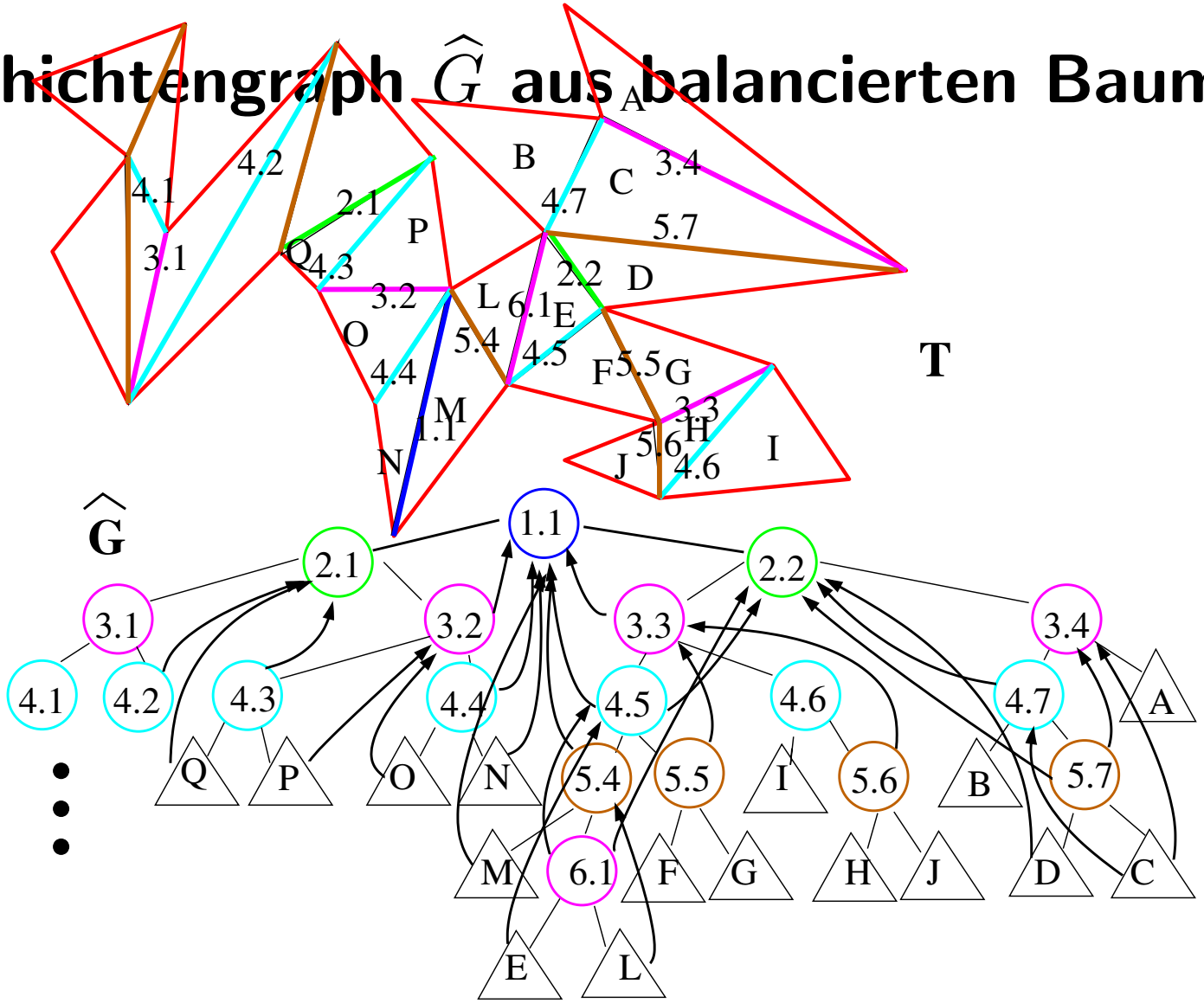
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



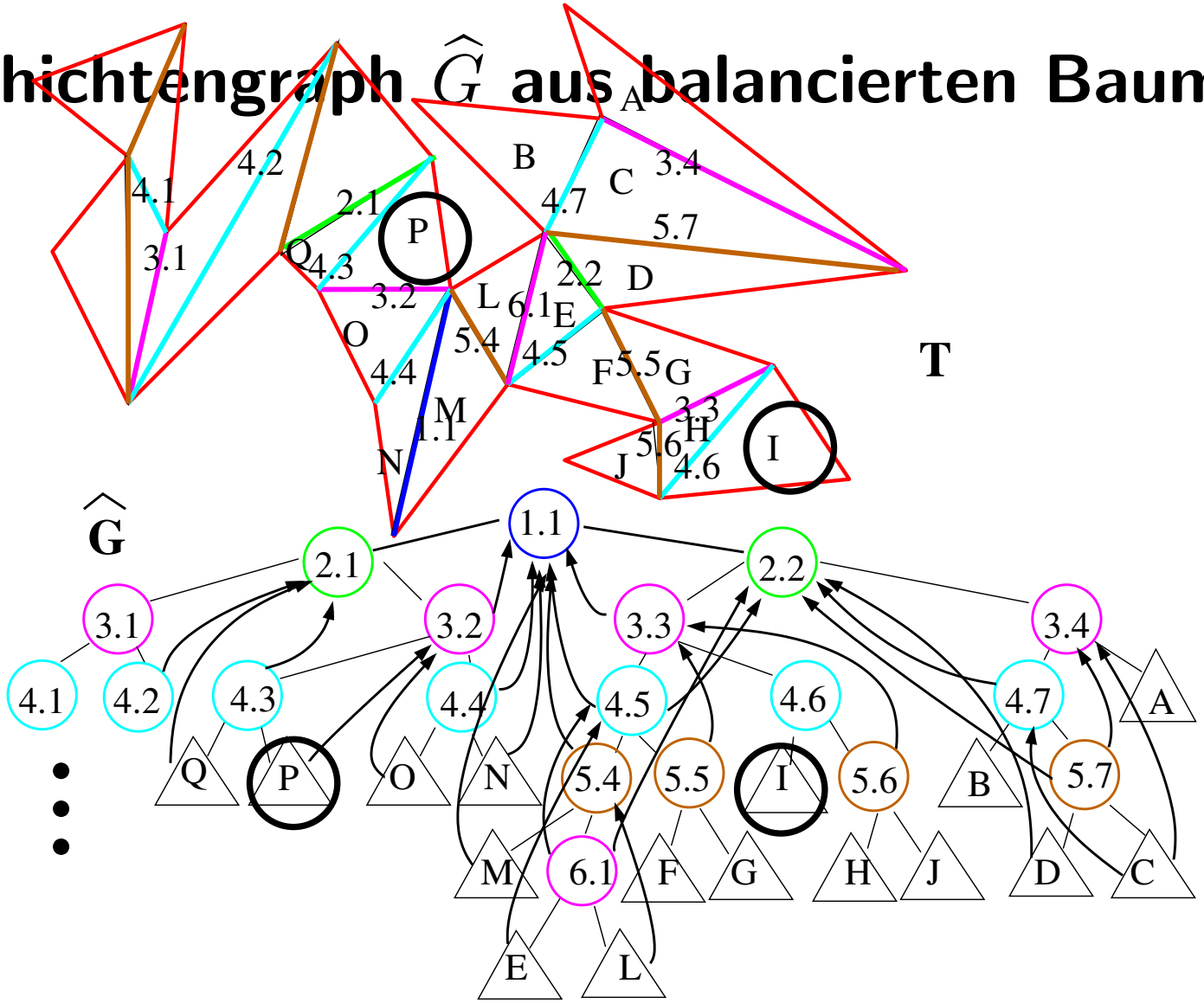
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



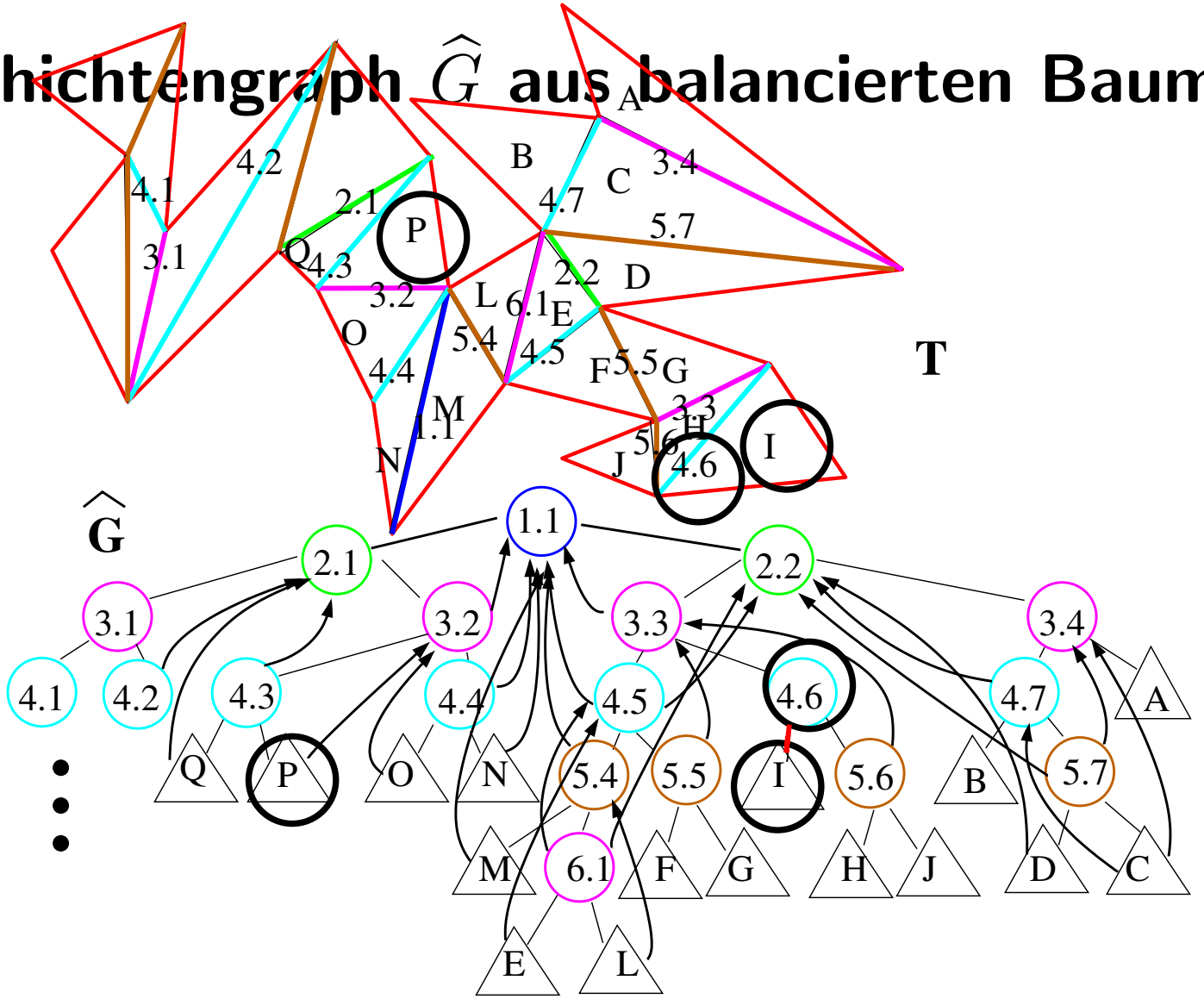
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



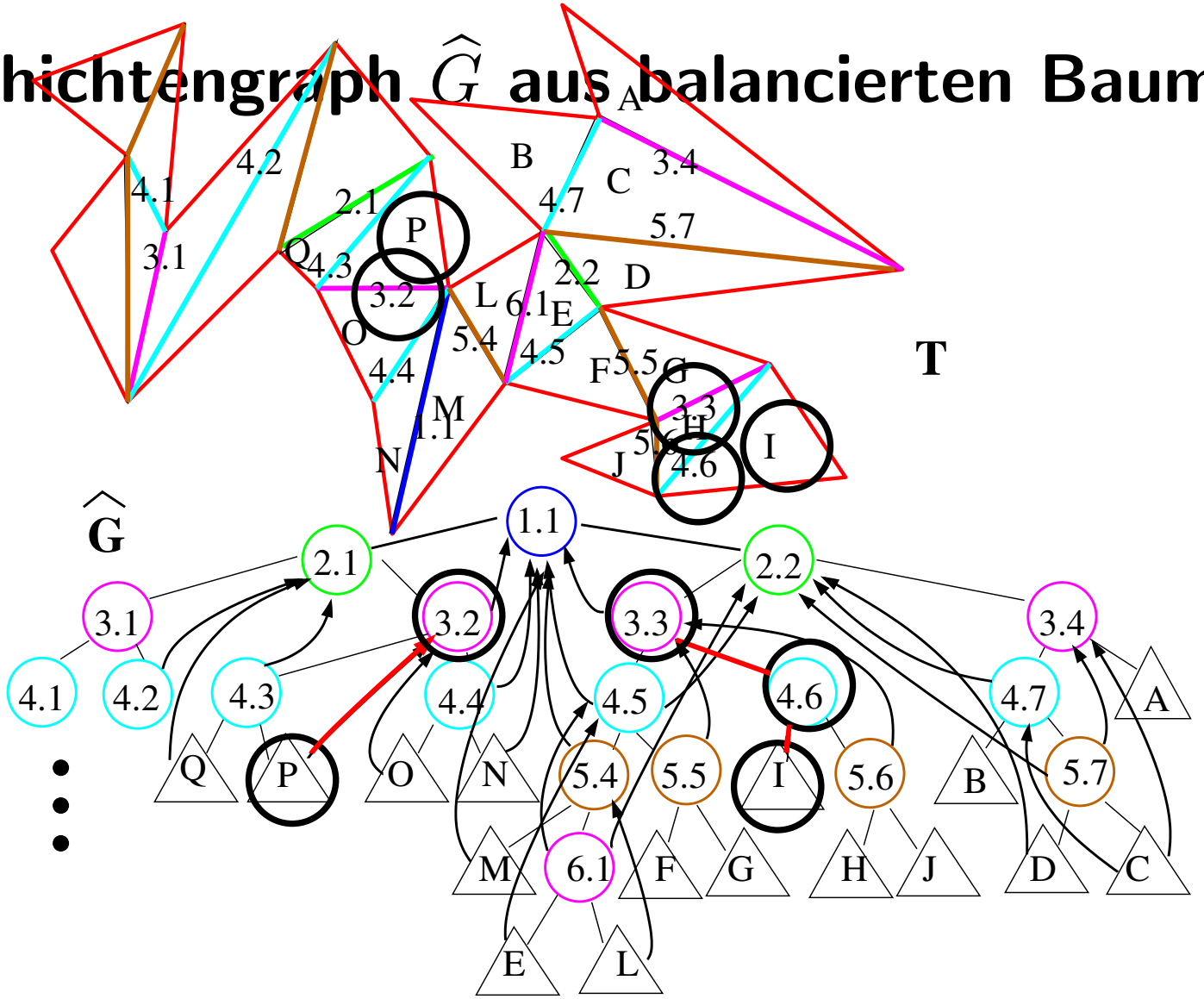
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



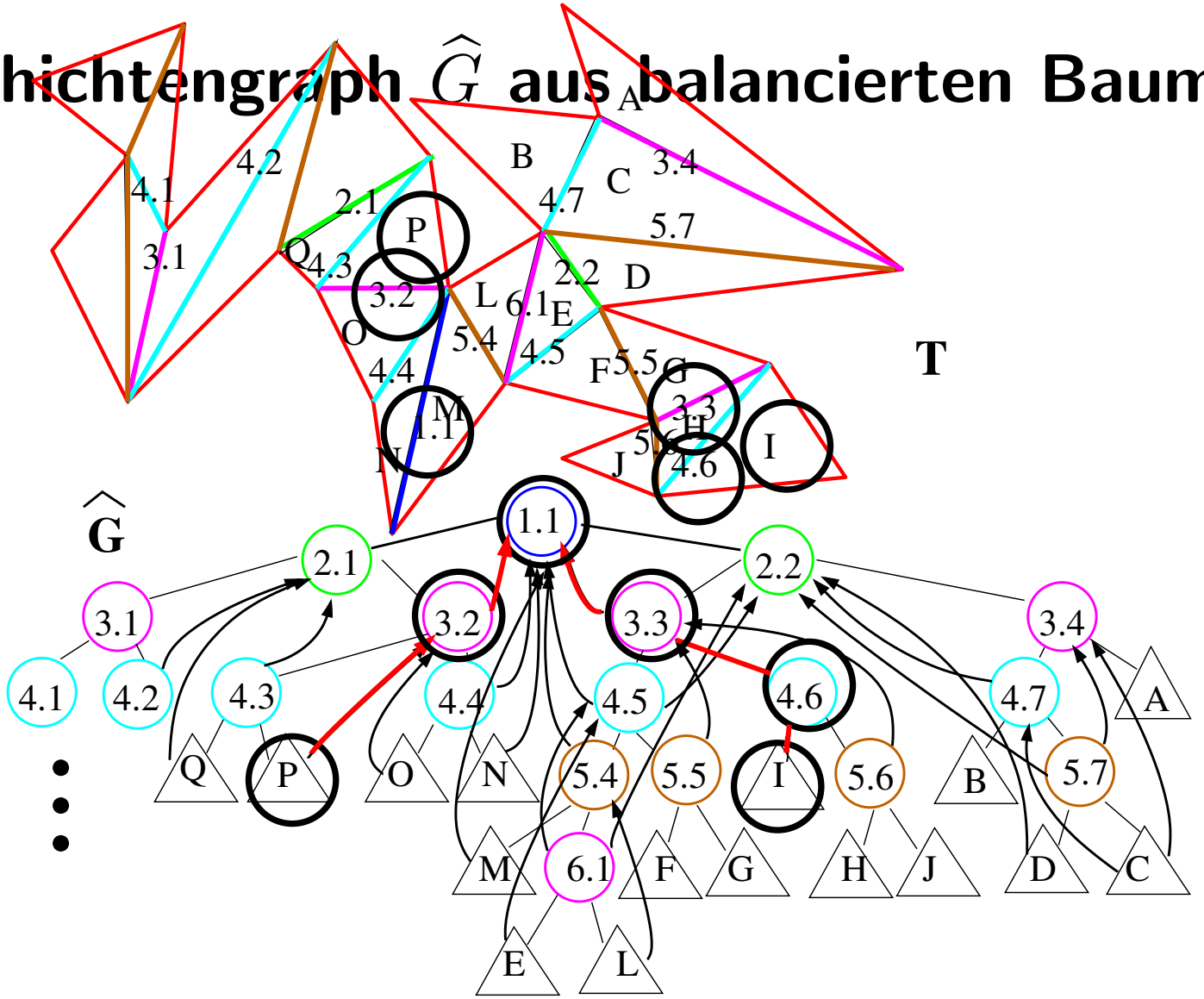
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



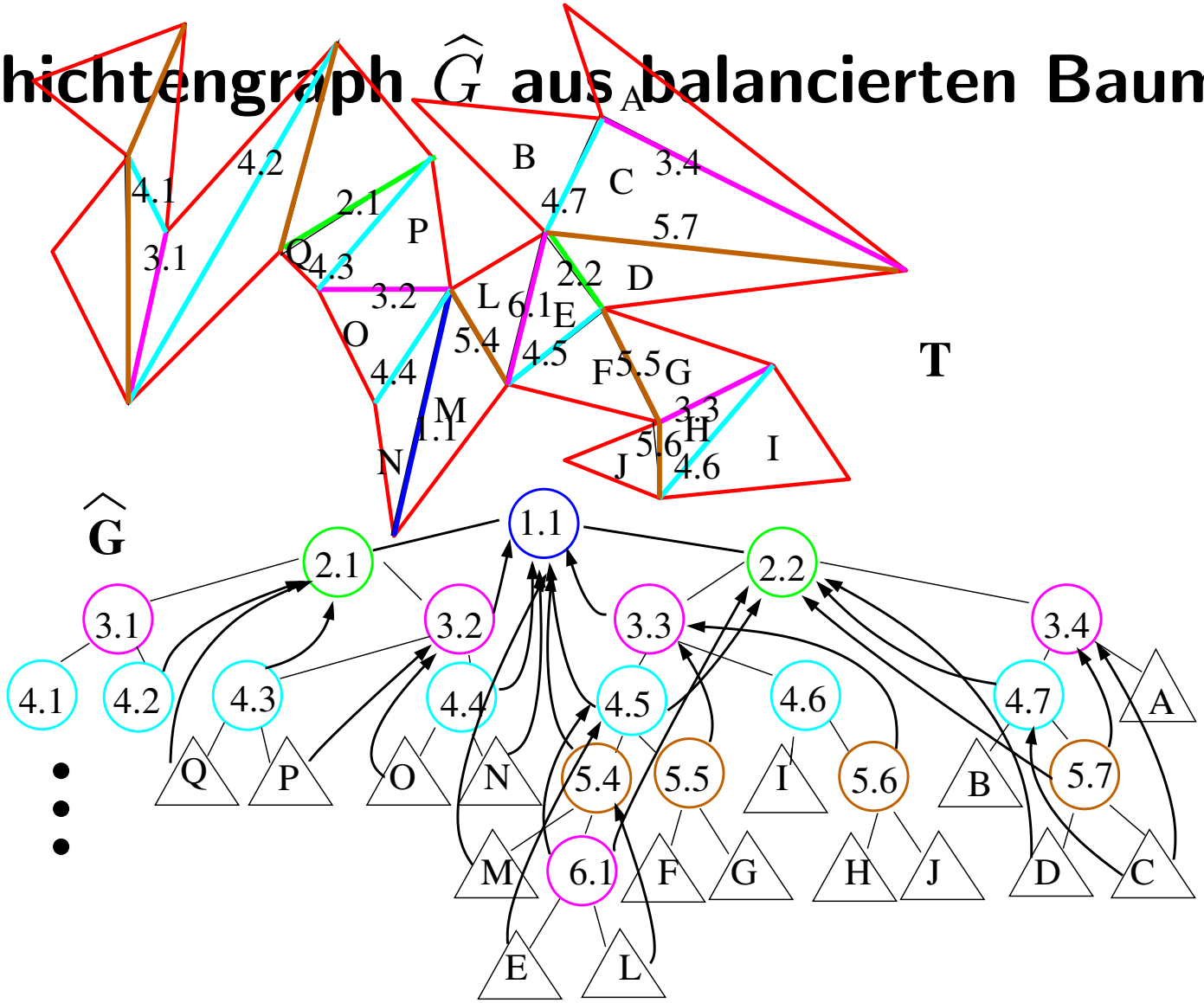
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



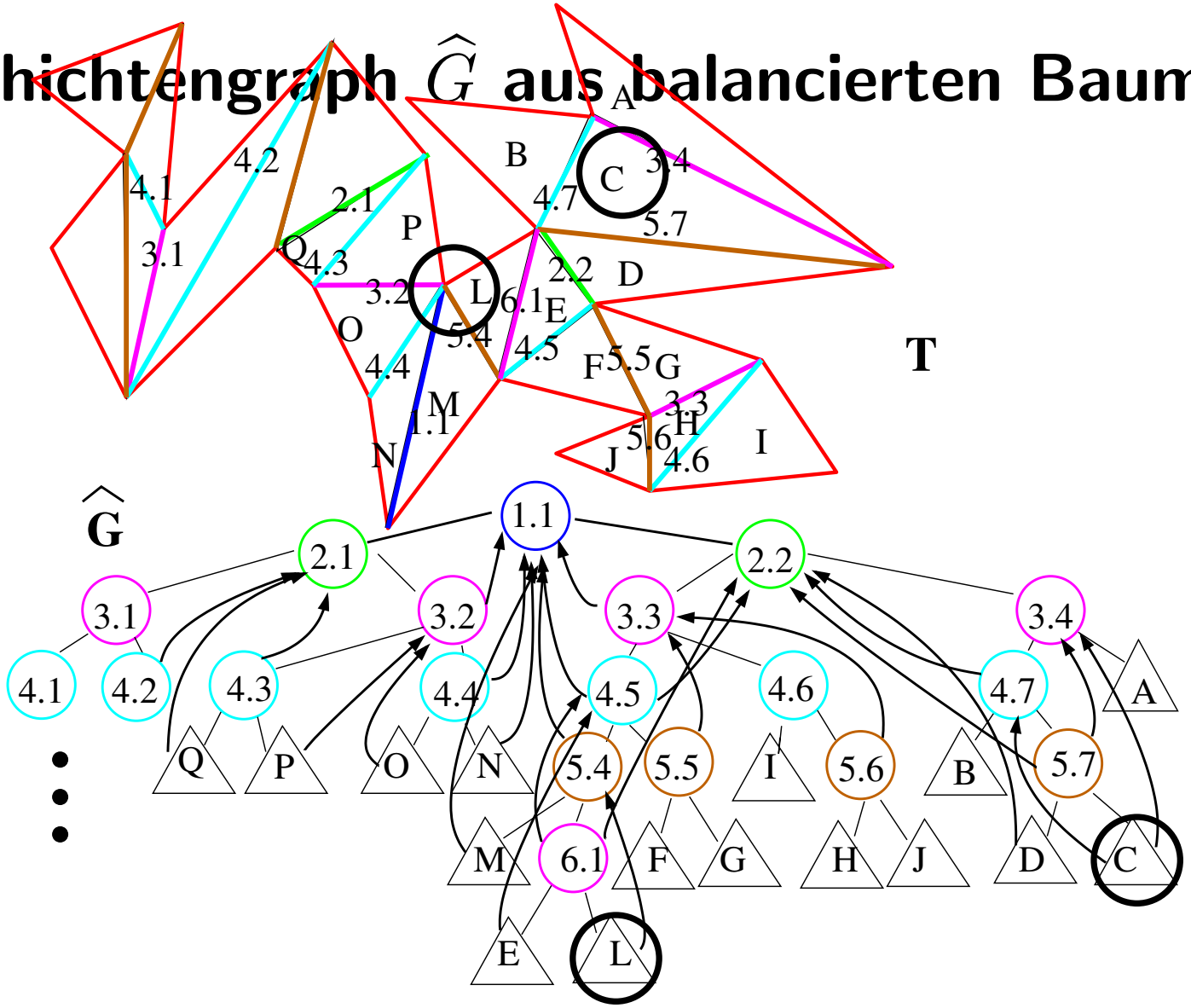
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



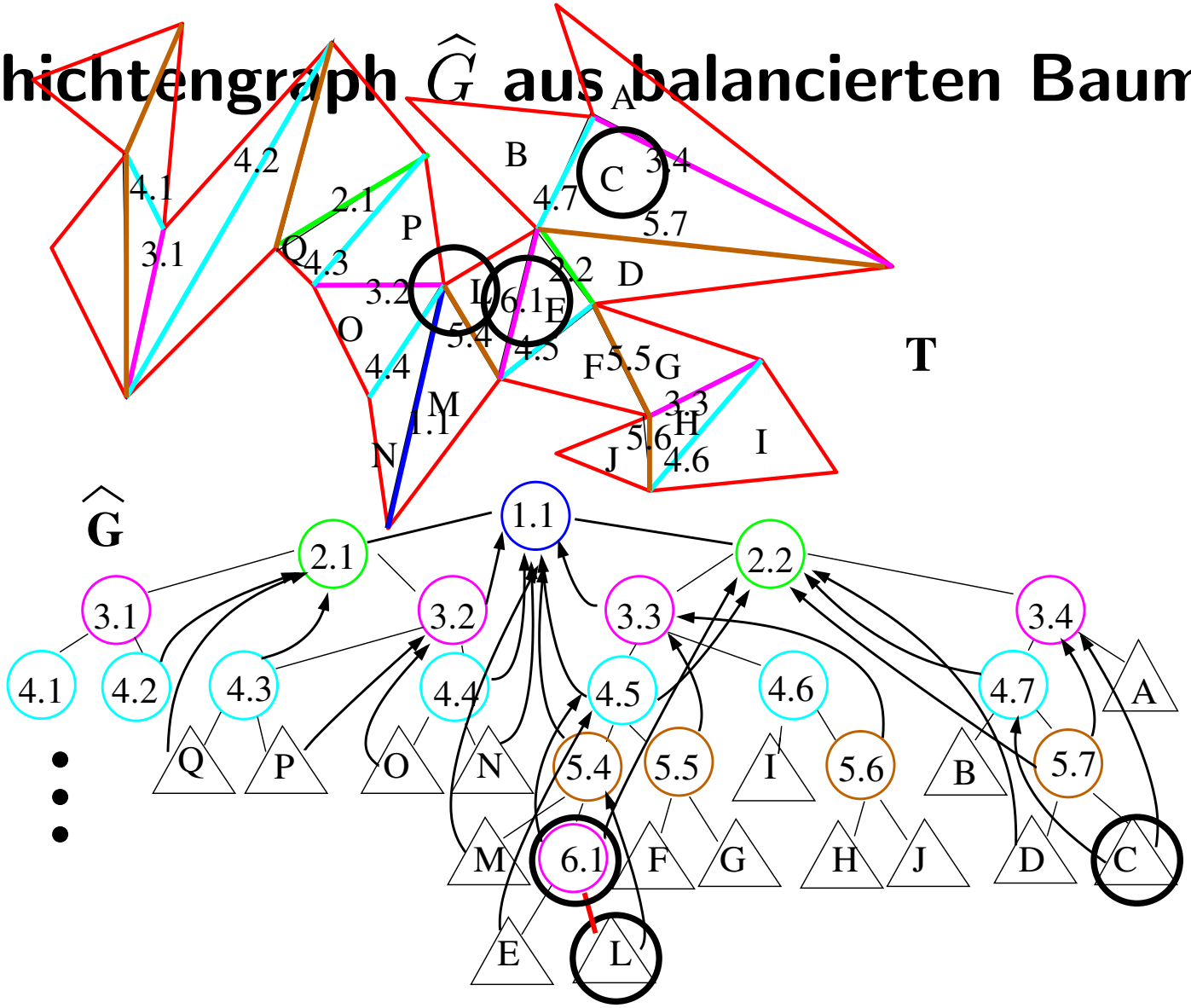
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



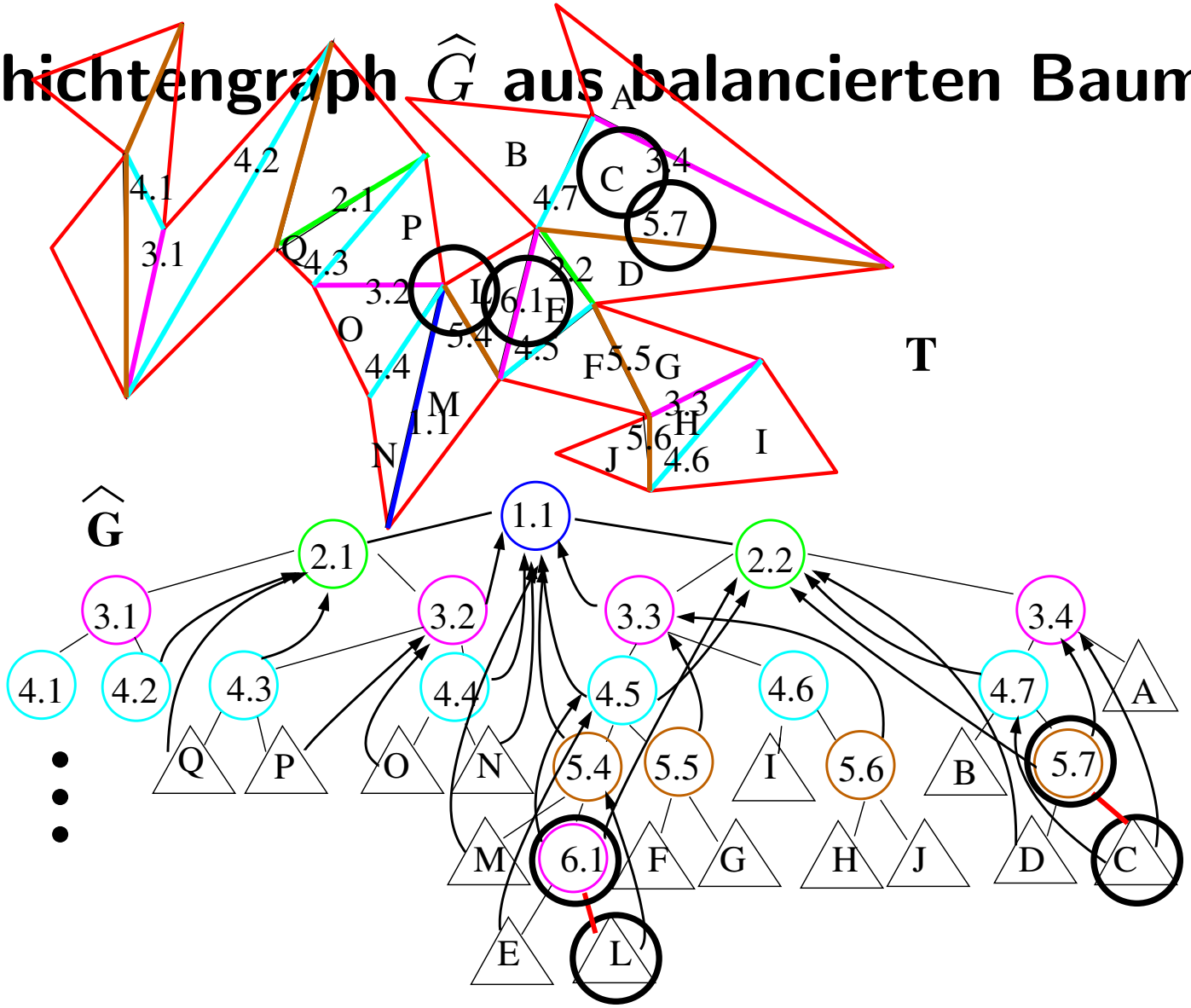
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



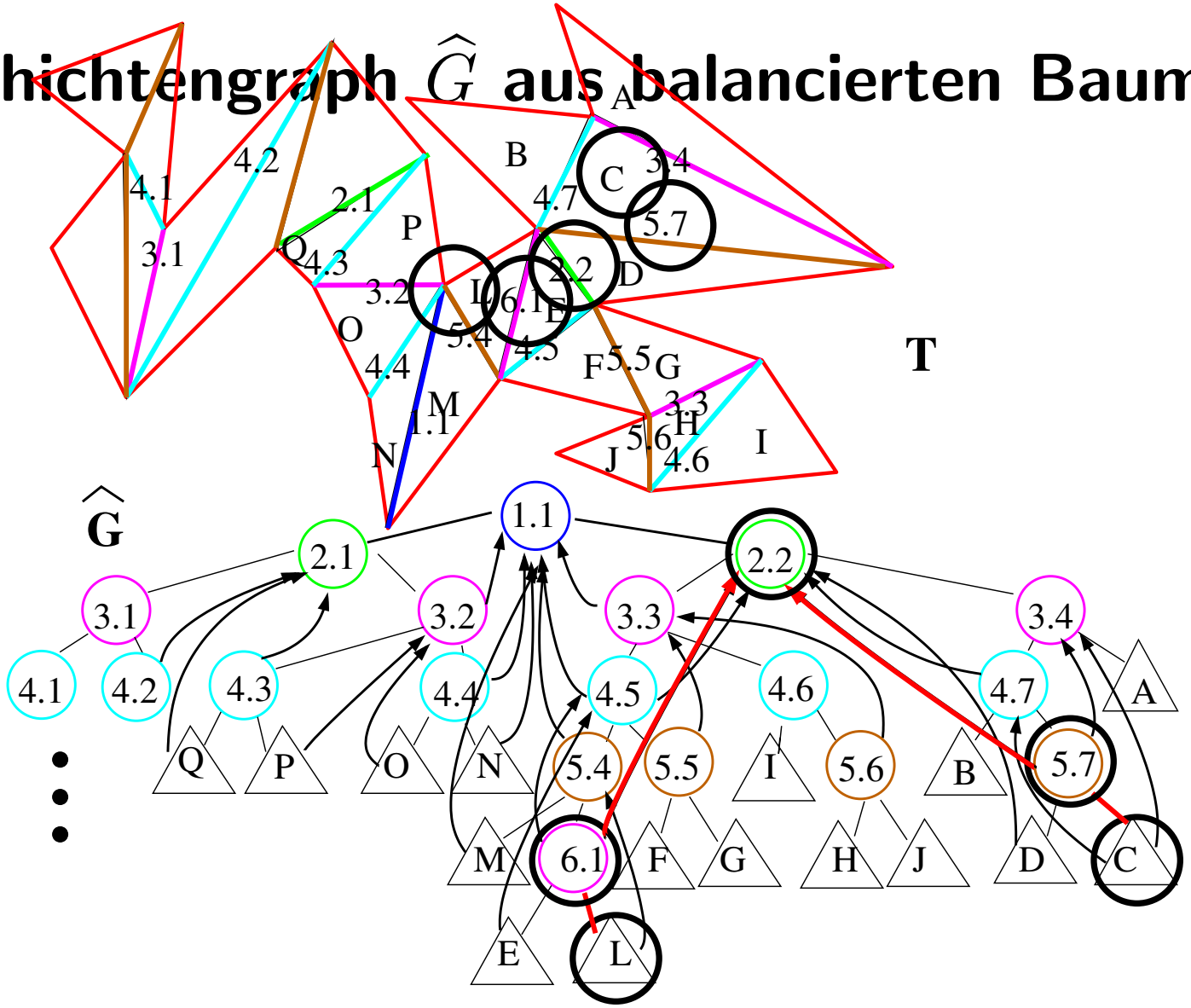
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



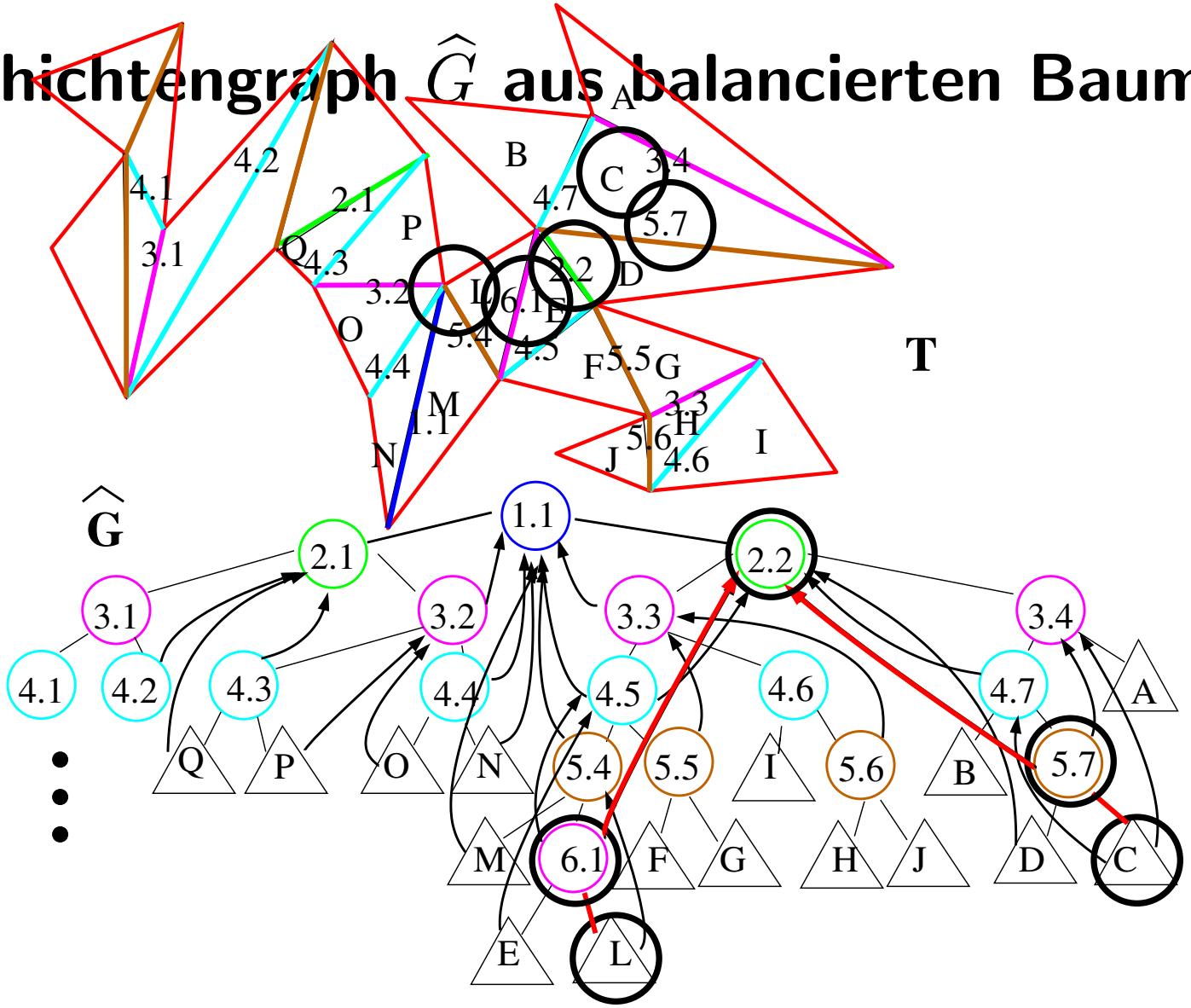
Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$

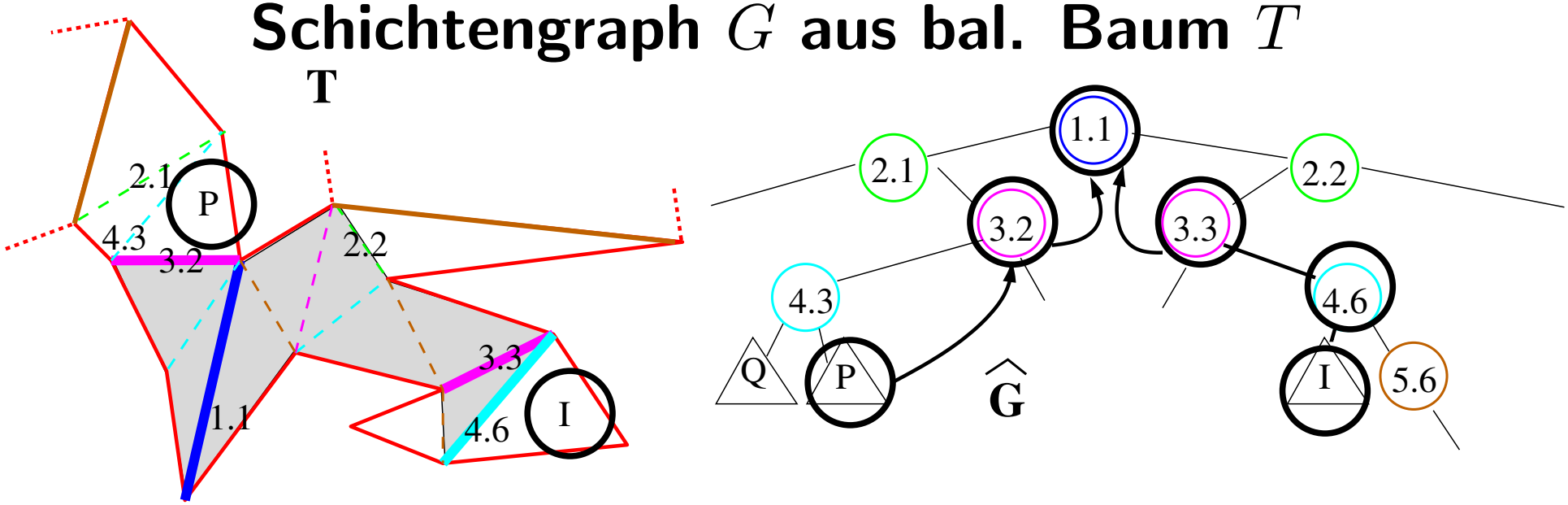


Schichtengraph $\widehat{G}$ aus balancierten Baum $\widehat{T}$



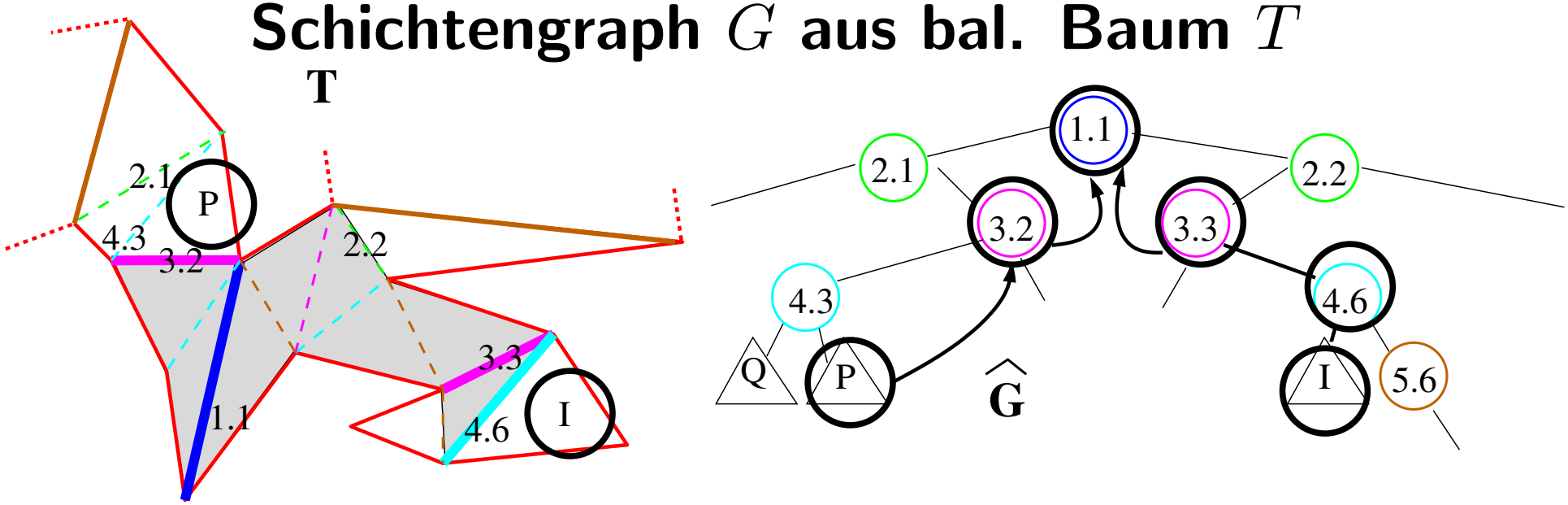
Schichtengraph $\widehat{G}$ aus bal. Baum $\widehat{T}$

Schichtengraph $\widehat{G}$ aus bal. Baum $\widehat{T}$



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

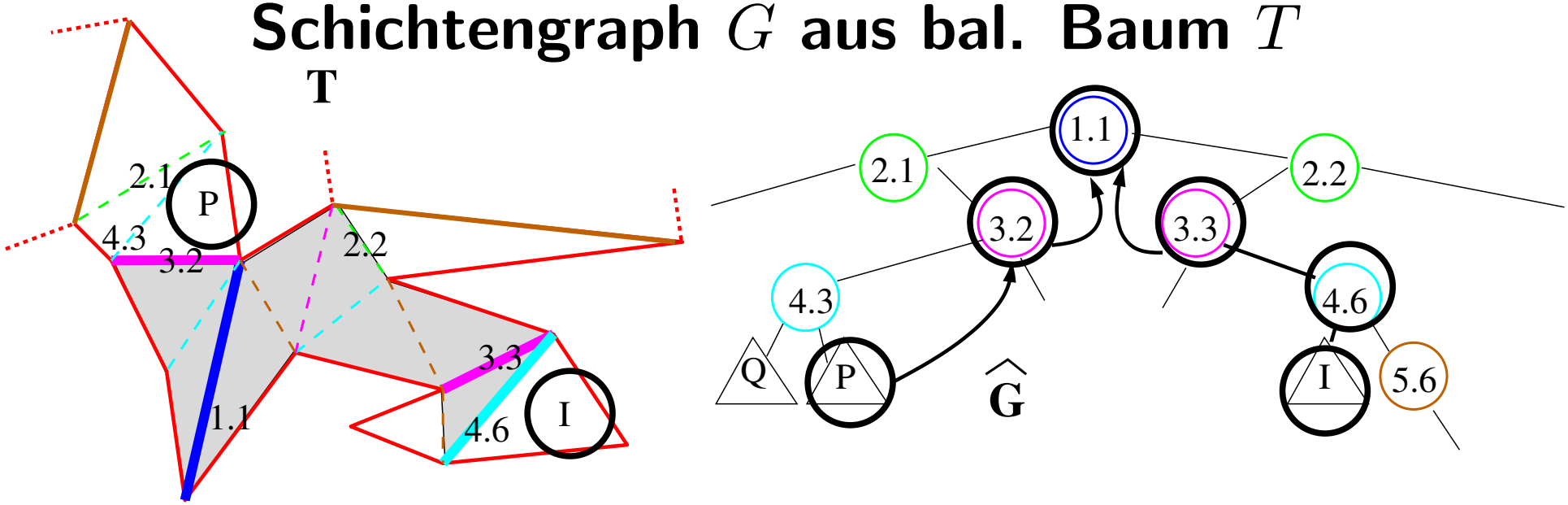
Schichtengraph $\widehat{G}$ aus bal. Baum $\widehat{T}$



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

Füllen der Löcher: Z.B.. zwischen (3.3) und (1.1)

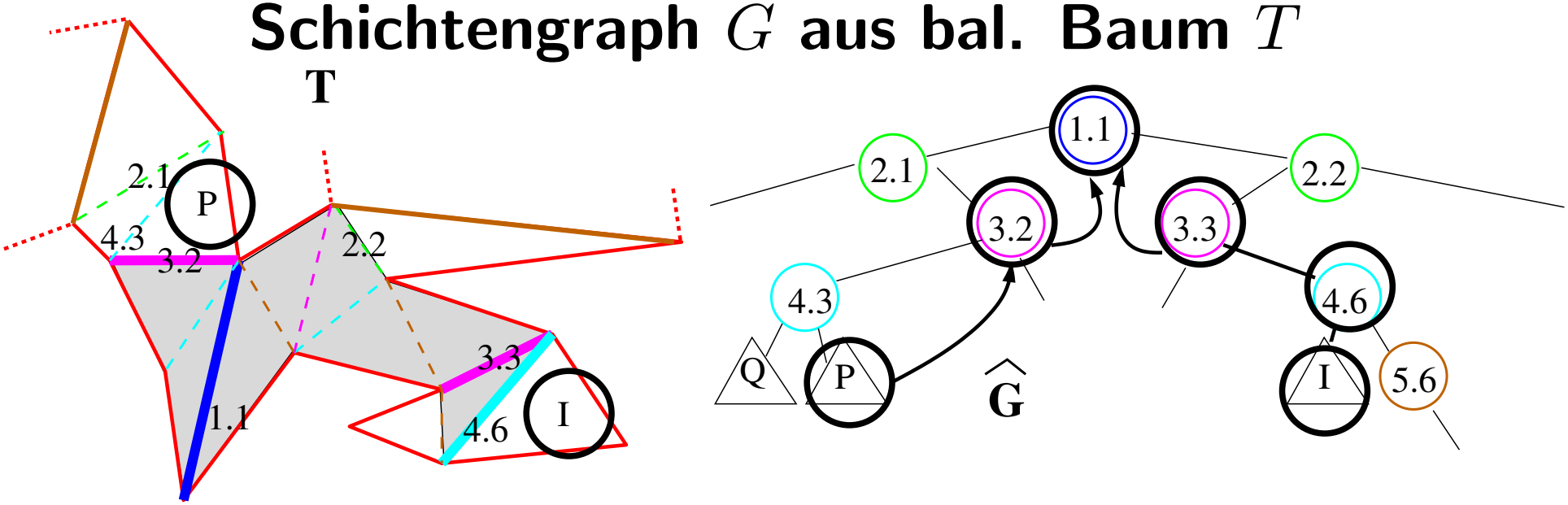
Schichtengraph $\widehat{G}$ aus bal. Baum $\widehat{T}$



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

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

Schichtengraph $\widehat{G}$ aus bal. Baum $\widehat{T}$



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

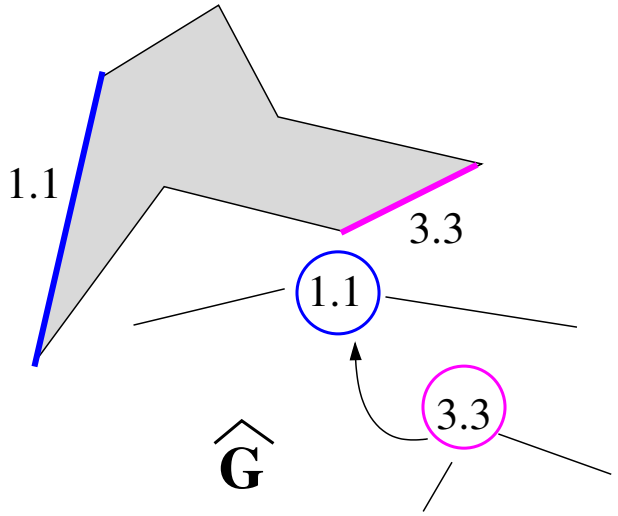
Füllen der Löcher: Z.B.. zwischen (3.3) und (1.1) Kanten in $\widehat{G}$!

Preprocessing: Kanten führen zu *Sanduhr* von Diagonalen

Sanduhren zwischen Diagonalen in P

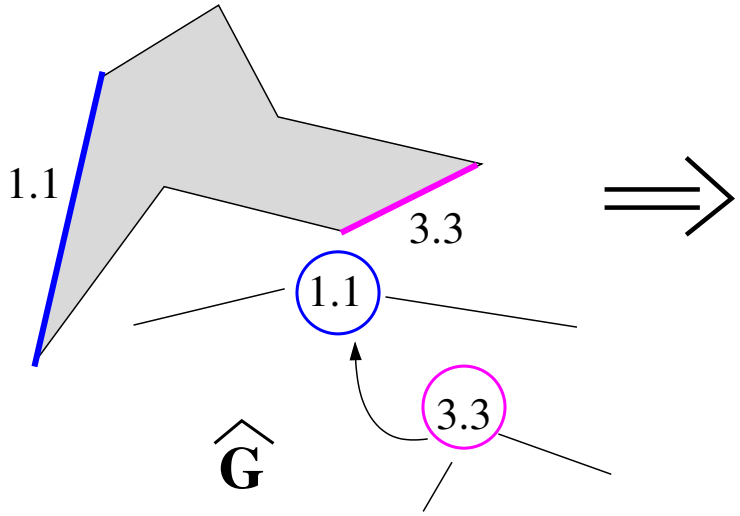
Sanduhren zwischen Diagonalen in P

- Kante in $\widehat{G}$



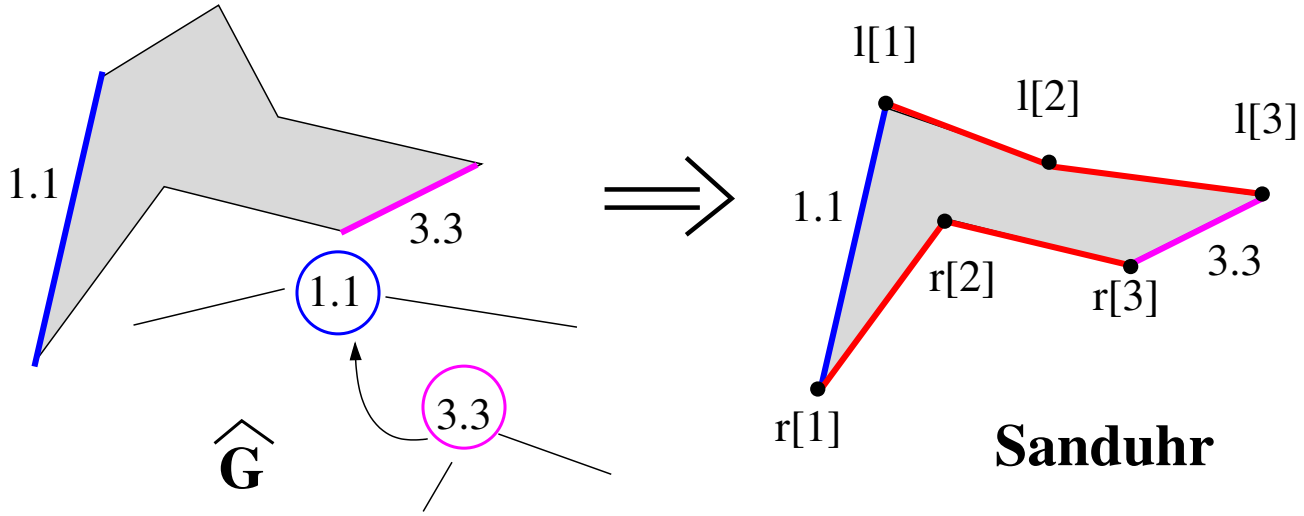
Sanduhren zwischen Diagonalen in P

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



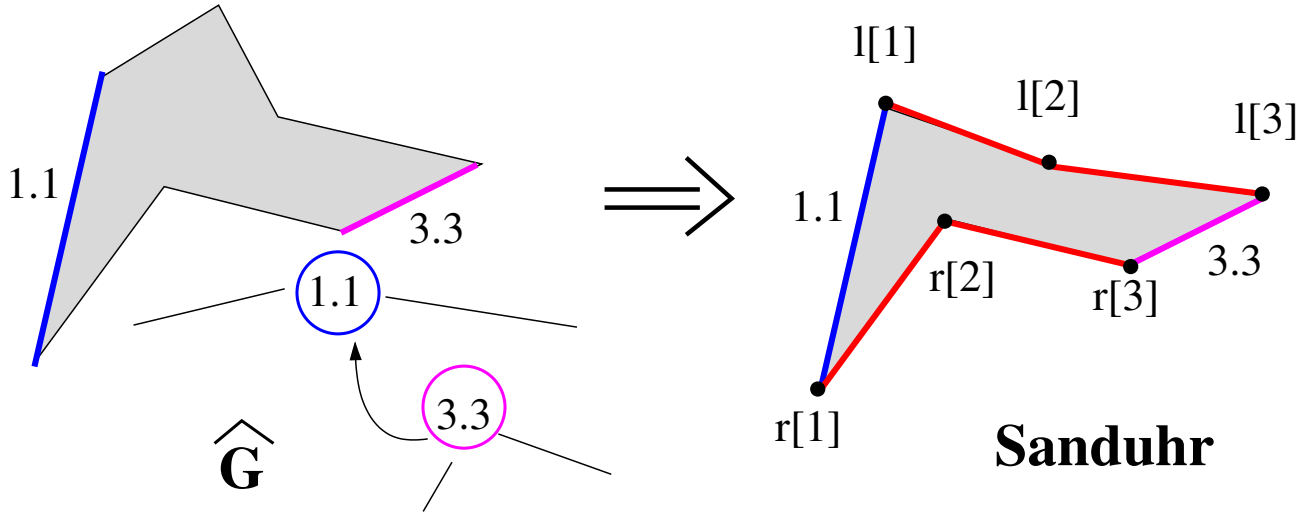
Sanduhren zwischen Diagonalen in P

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



Sanduhren zwischen Diagonalen in P

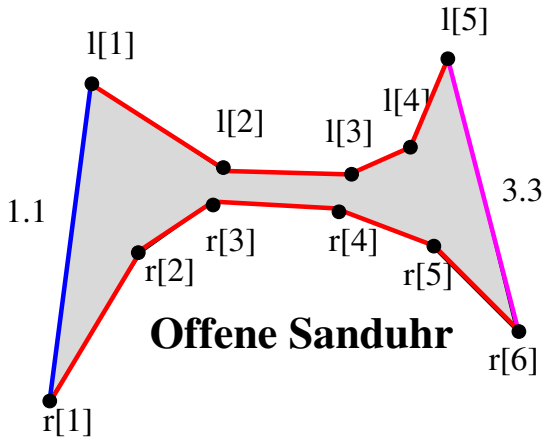
- Kante in $\widehat{G}$
- Shortest Path zwischen Endpunkte der Diagonalen: DS



DS *Sanduhr* im Allgemeinen

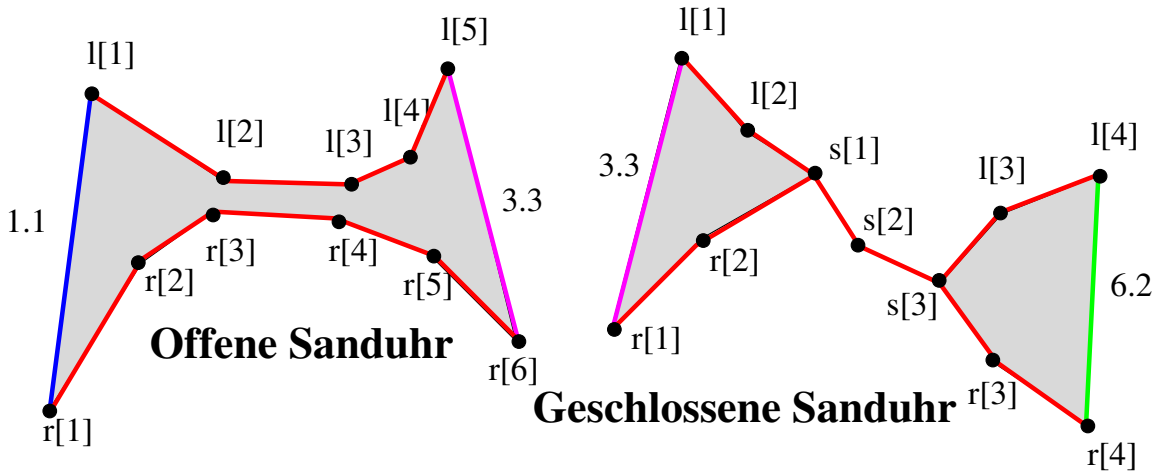
DS *Sanduhr* im Allgemeinen

- Offene Sanduhr zwischen Diagonalen, DS



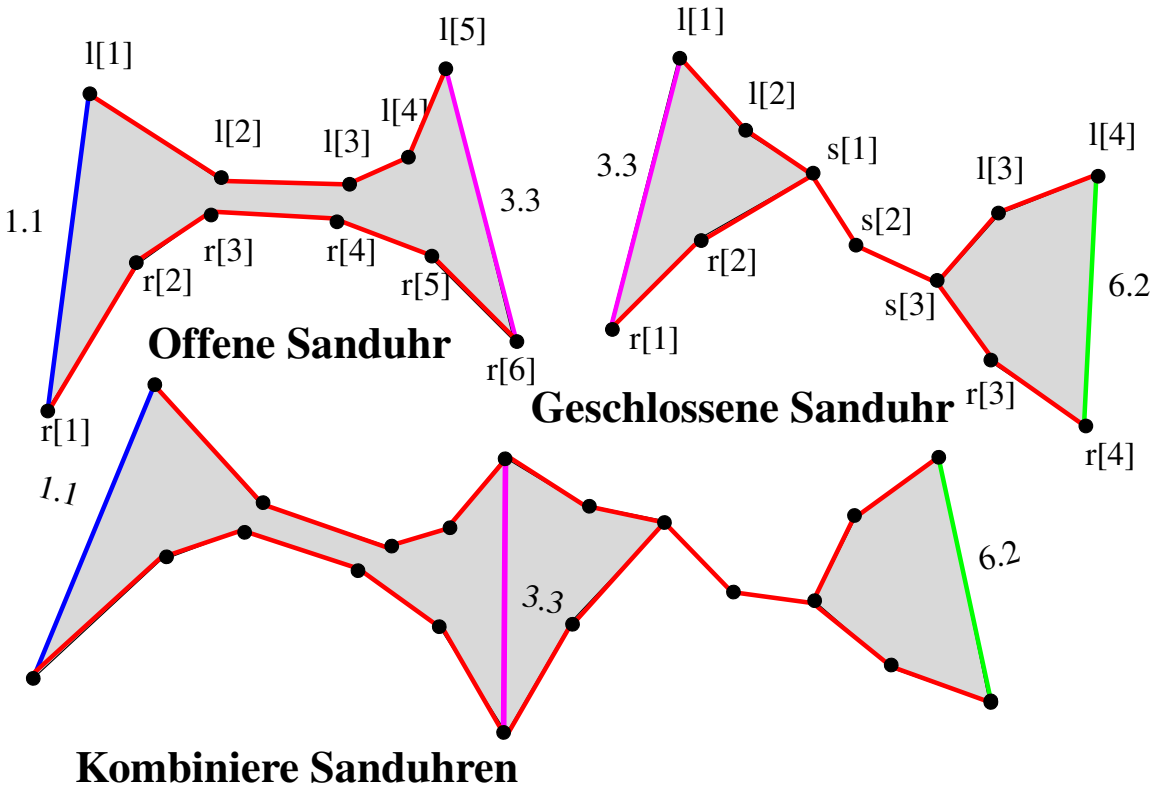
DS *Sanduhr* im Allgemeinen

- Offene Sanduhr zwischen Diagonalen, DS
- Geschlossene Sanduhr zwischen Diagonalen, DS



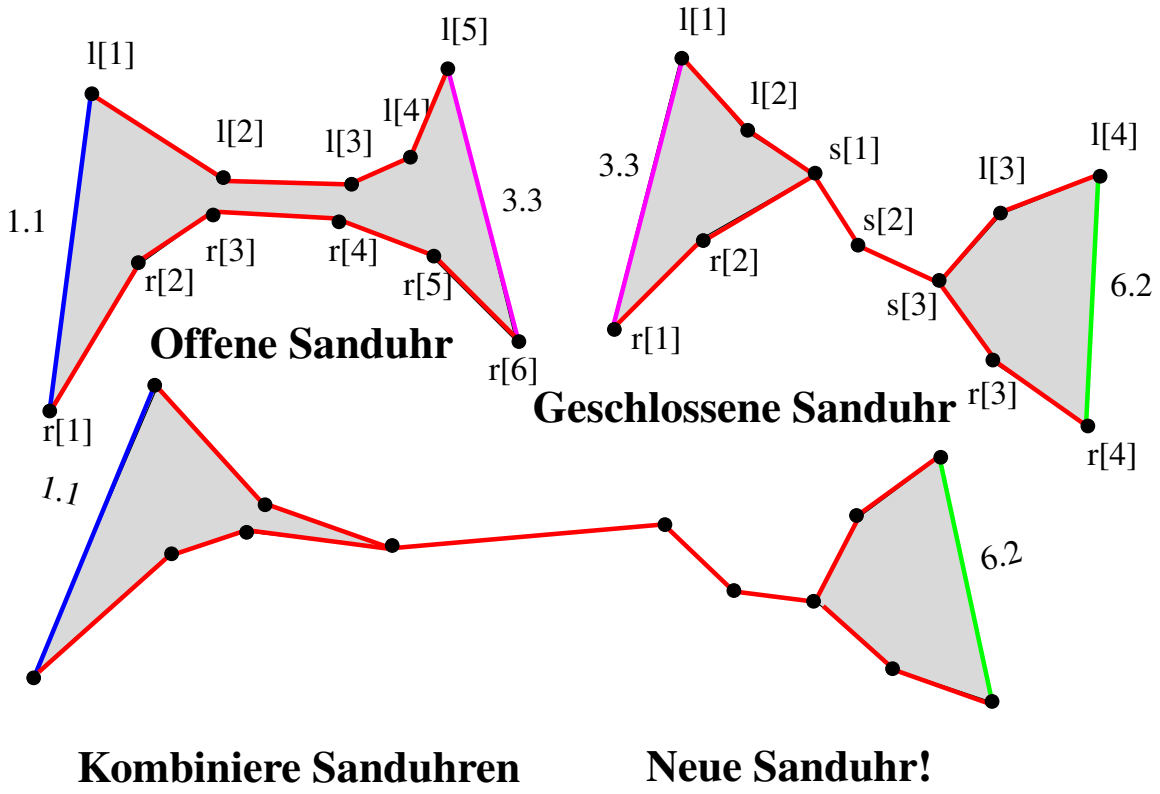
DS *Sanduhr* im Allgemeinen

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



DS *Sanduhr* im Allgemeinen

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



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: $O(\log n + k)$

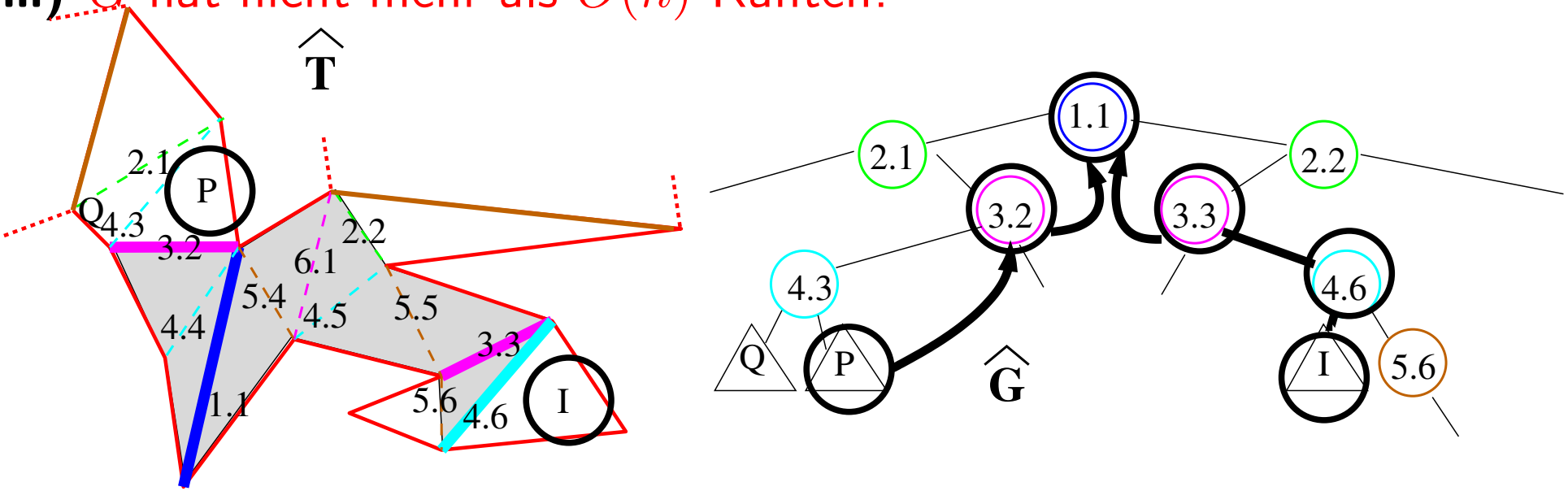
Query: Start $A \in P$, Ziel $B \in I$: Löse 5), 6) und 7)!!

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

i) Pfad zwischen zwei Dreiecken entlang sukzessiver Diagonalen existiert!

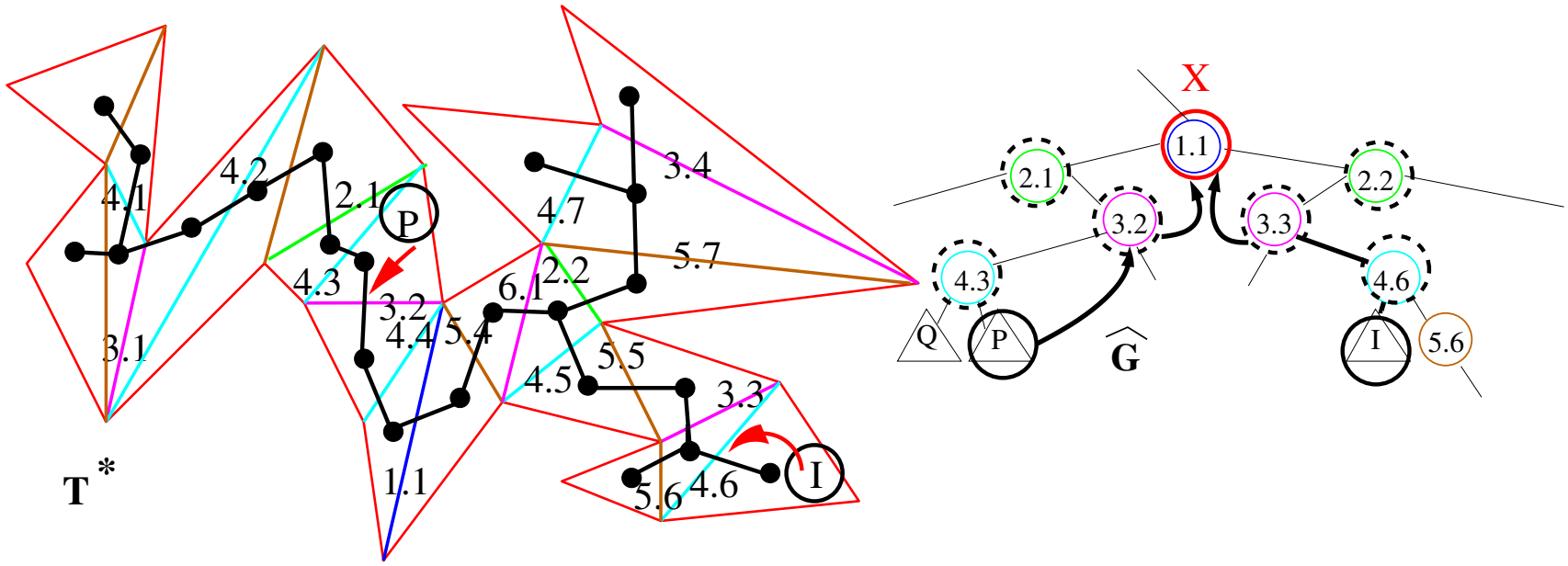
ii) Wir finden den Weg in $O(\log n)$ Zeit!

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

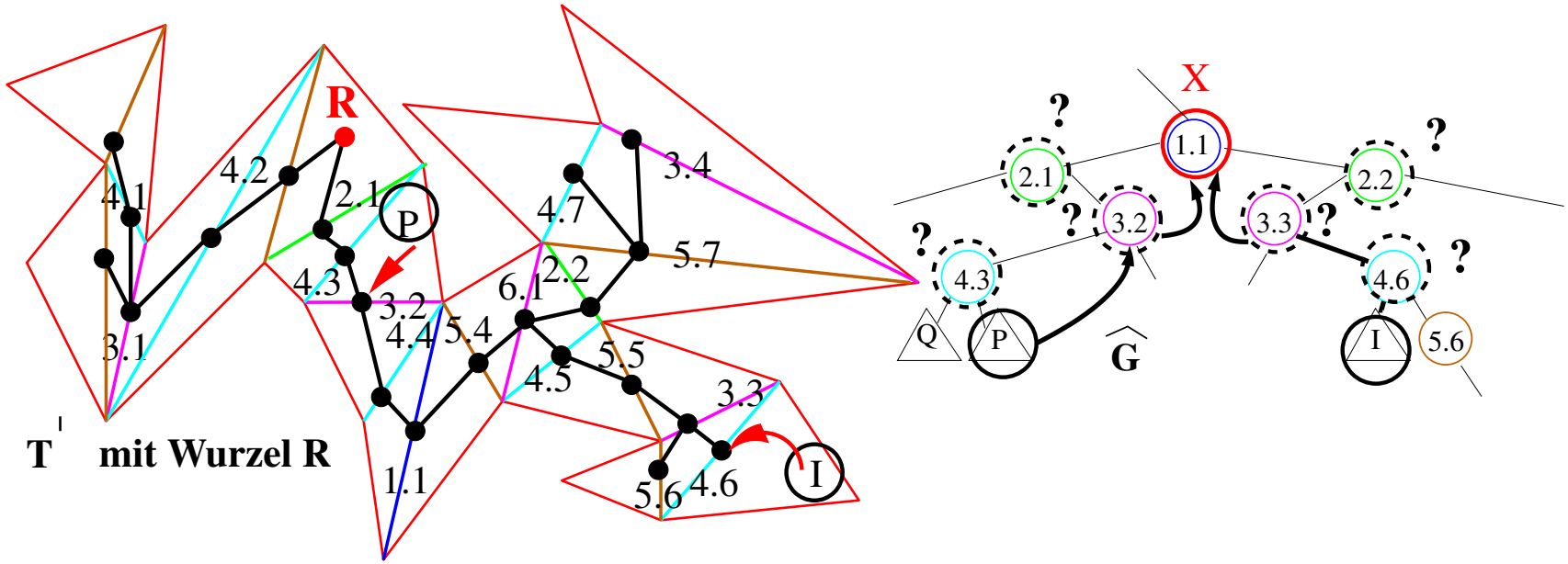


i) Weg in $\widehat{G}$ existiert

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



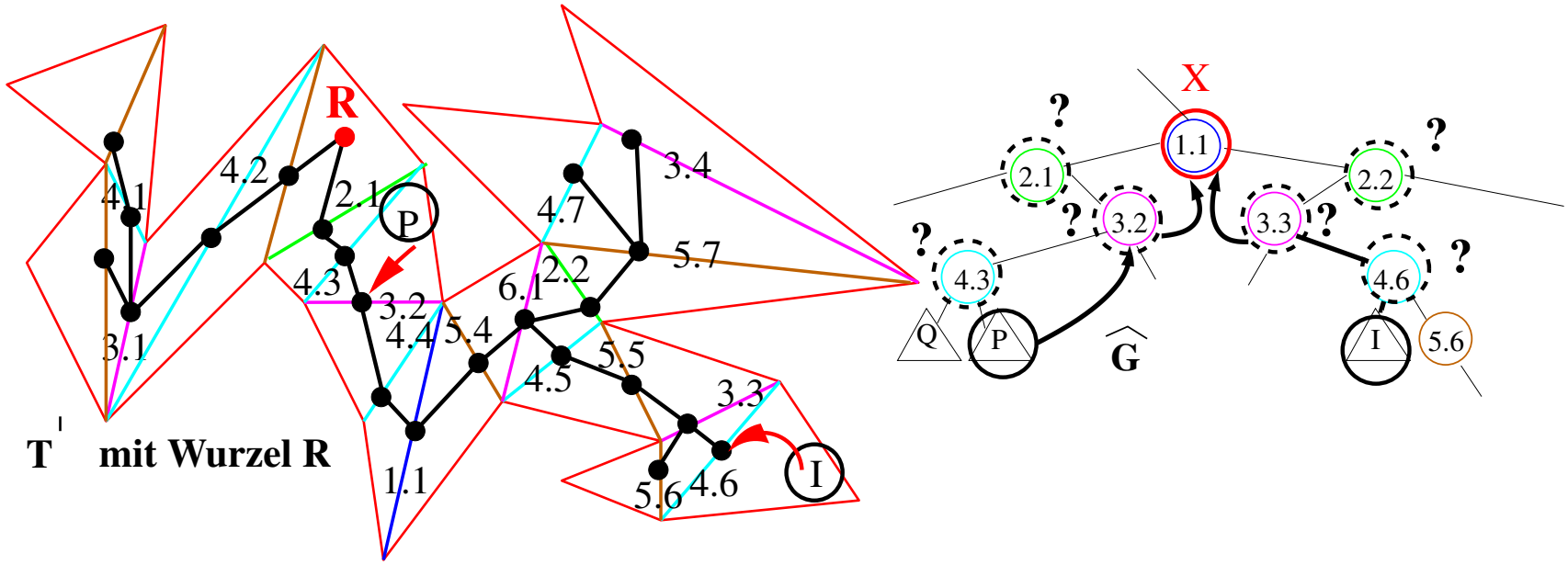
ii) Finden des Weges in $\widehat{G}$!



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

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

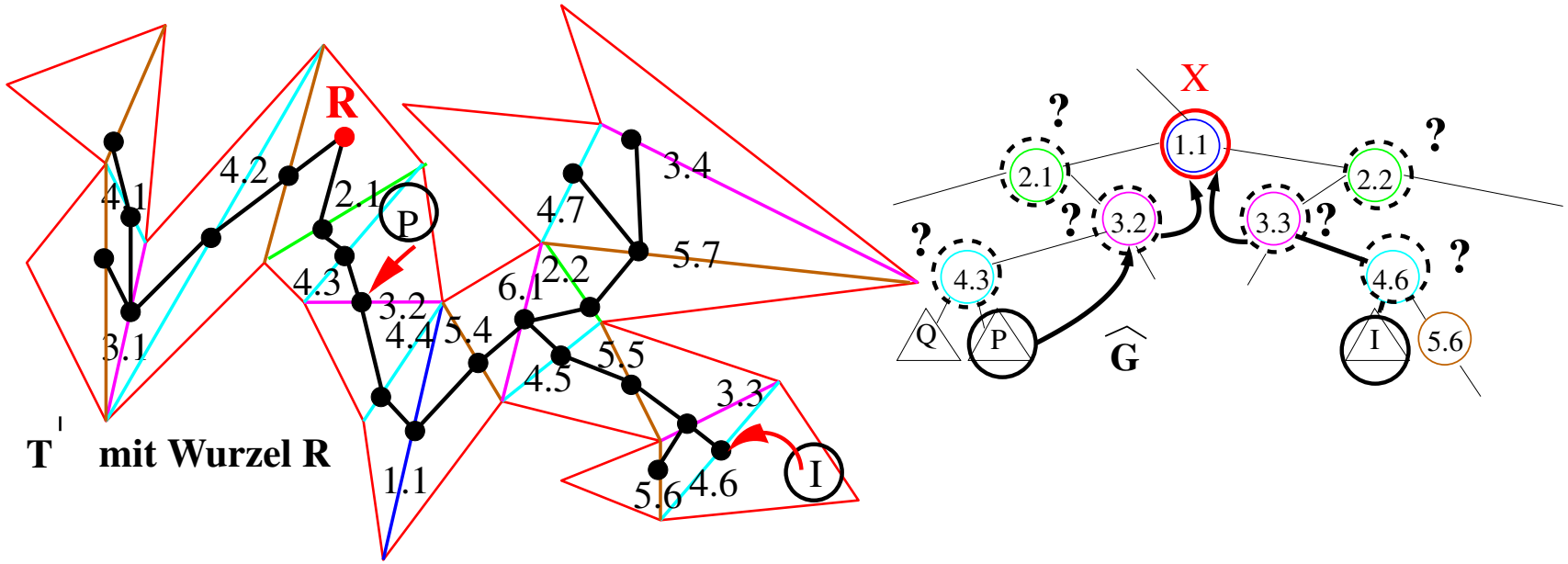
- Gemeinsamer Vorgänger X minimaler Höhe in $O(\log n)$ in $\widehat{T}$



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?

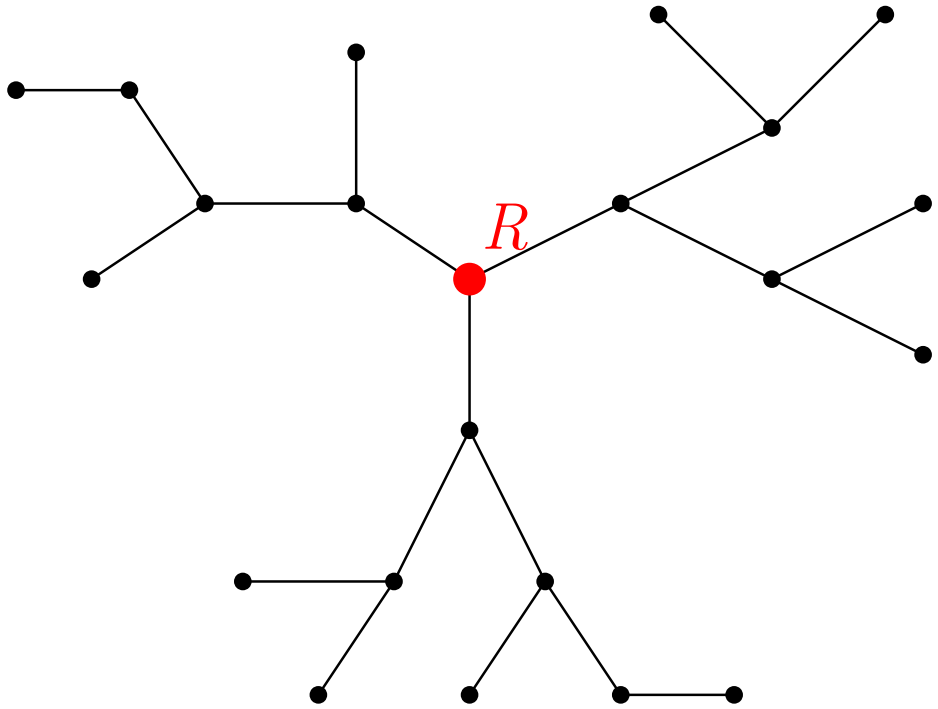


d Vorgänger entweder von delta(P) ODER delta(I) in Bezug auf R

Vorgängeranfrage in Baum

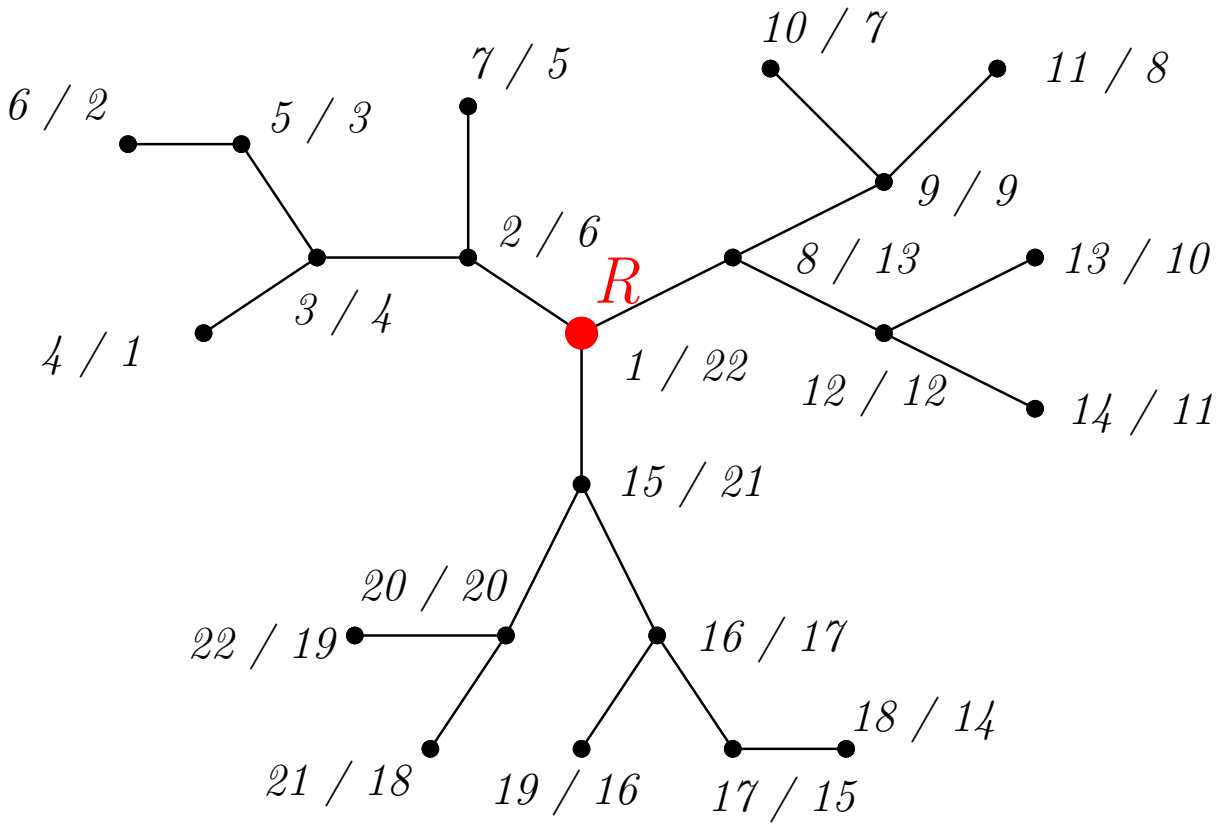
Vorgängeranfrage in Baum

- Preorder/Postorder,



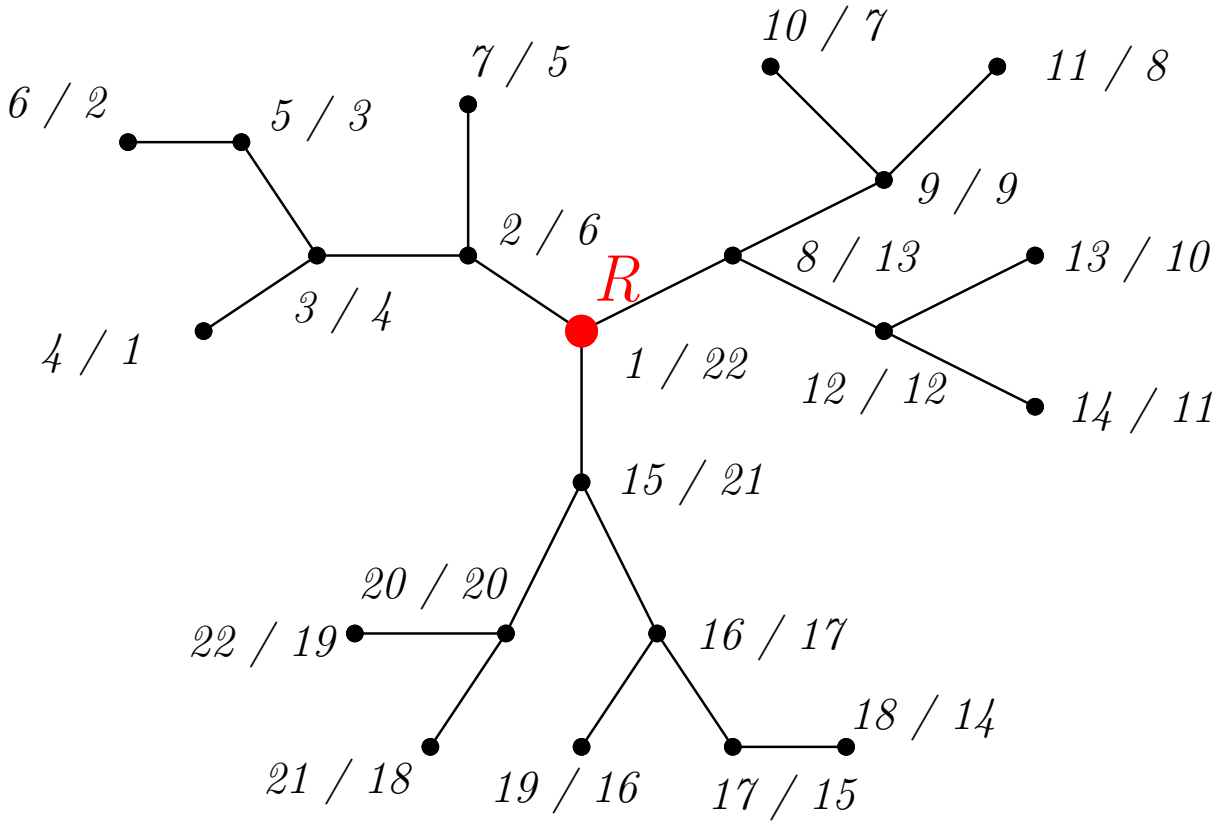
Vorgängeranfrage in Baum

- Preorder/Postorder, DFS und Labelling



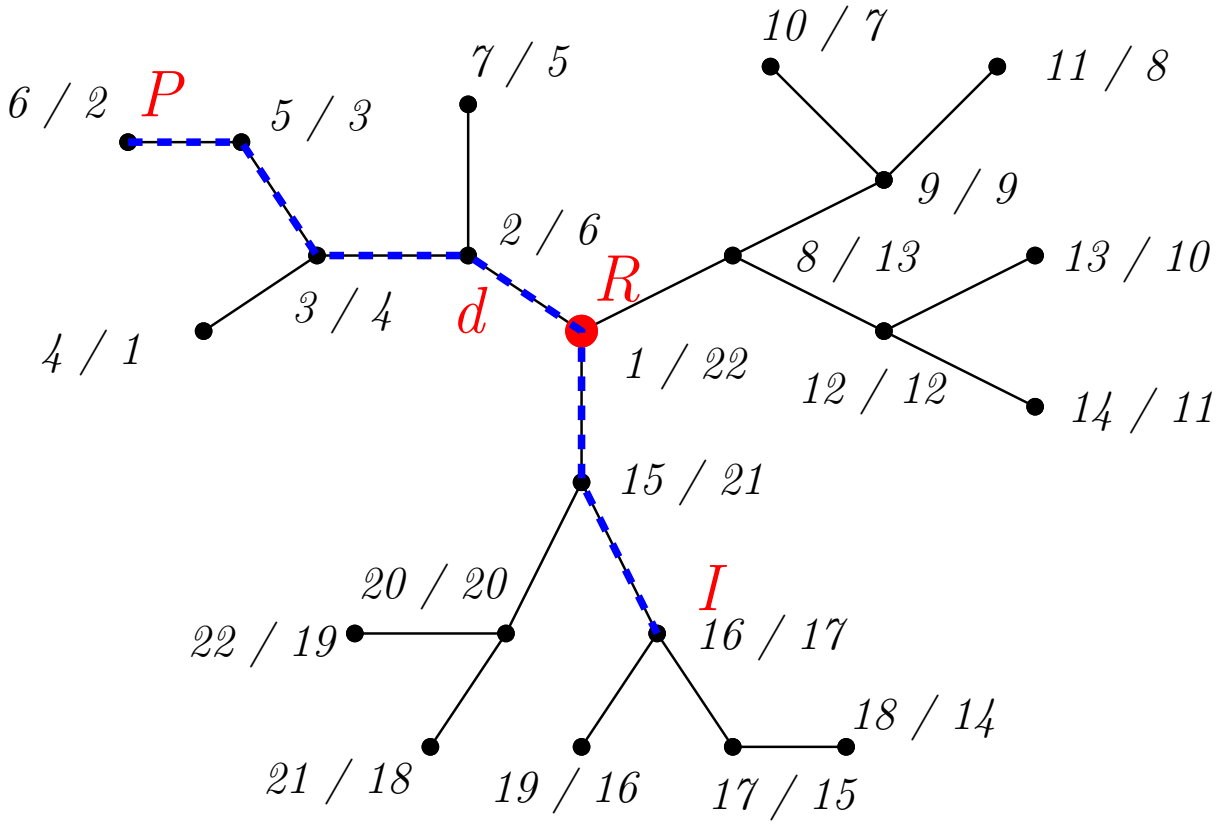
Vorgängeranfrage in Baum

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



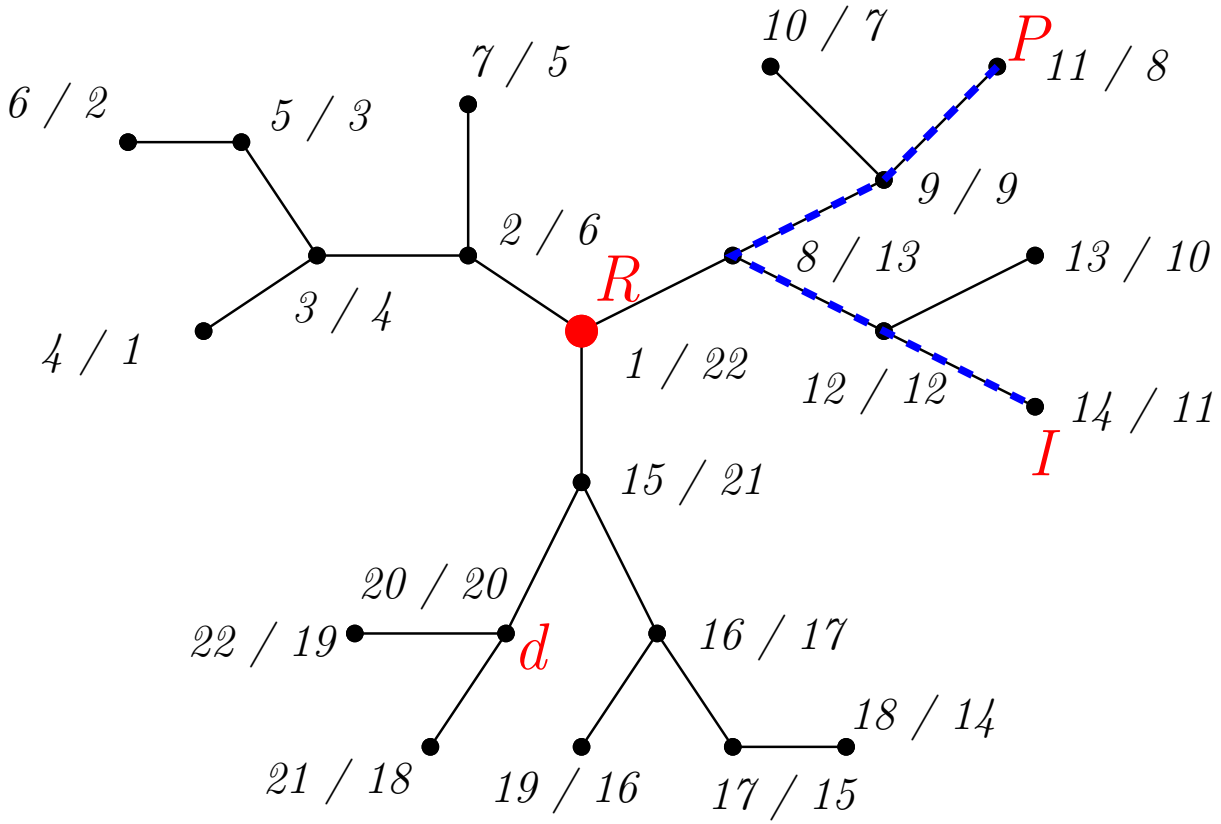
Vorgängeranfrage in Baum

- 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 $\widehat{G}$

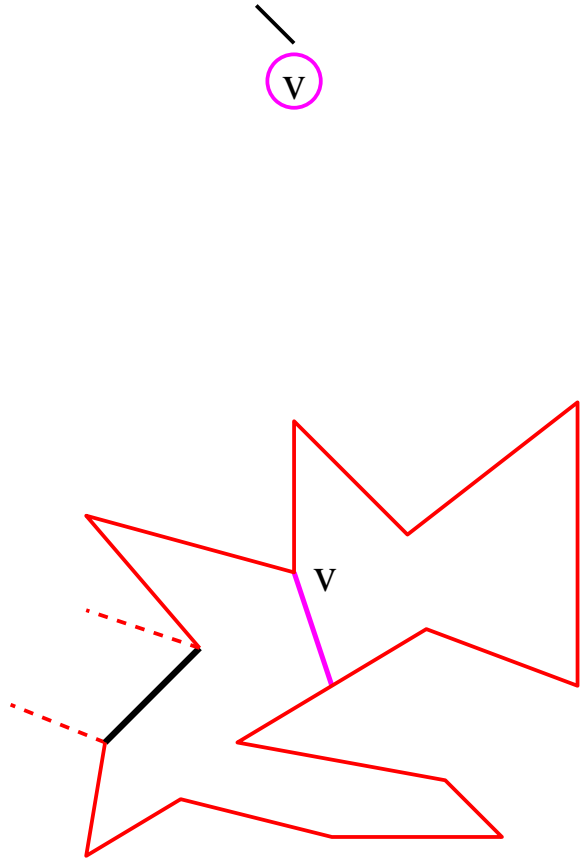
i) Pfad zwischen zwei Dreiecken existiert!

i) Länge ist in $O(\log n)$!

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

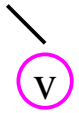
iii) $\widehat{G}$ hat $O(n)$ Kanten !

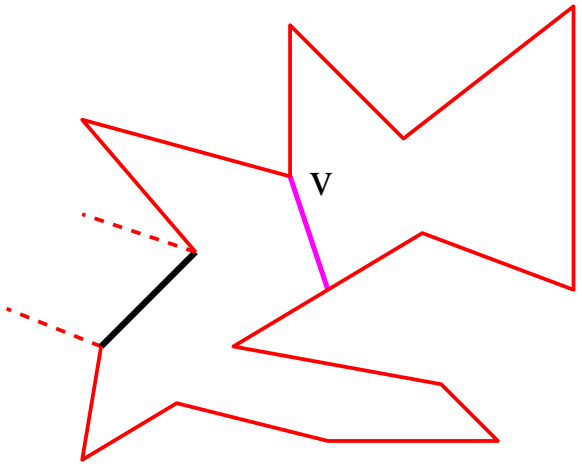
iii) Komplexität von $\widehat{G}$



iii) Komplexität von $\widehat{G}$

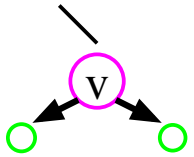
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$

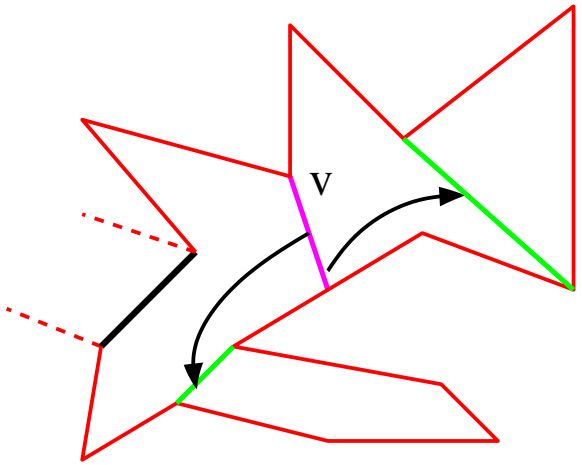




iii) Komplexität von $\widehat{G}$

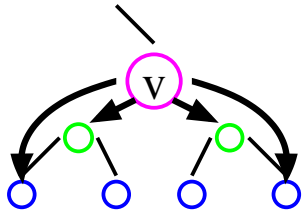
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$

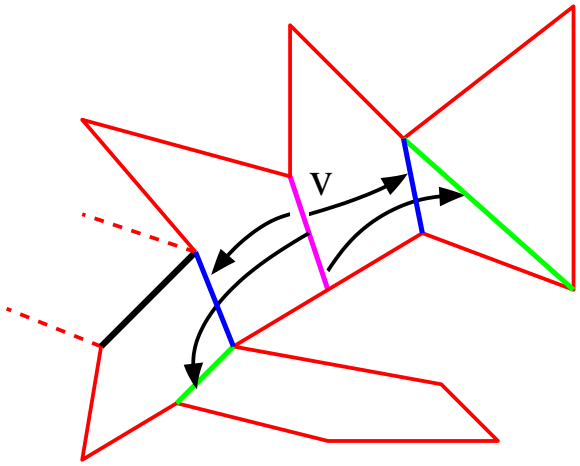




iii) Komplexität von $\widehat{G}$

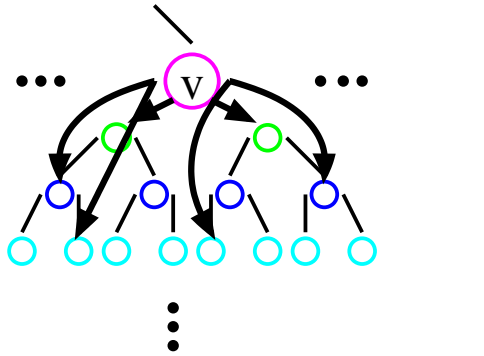
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$

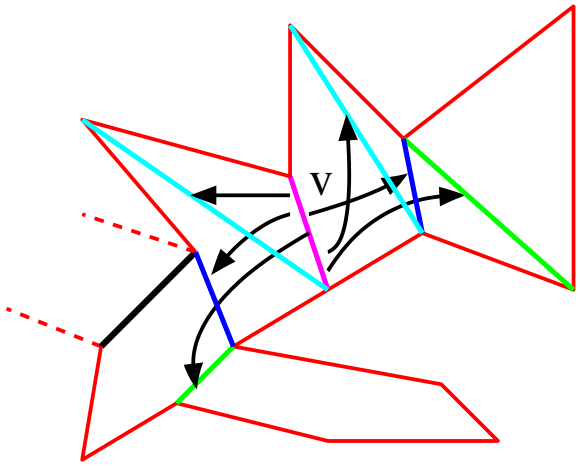




iii) Komplexität von $\widehat{G}$

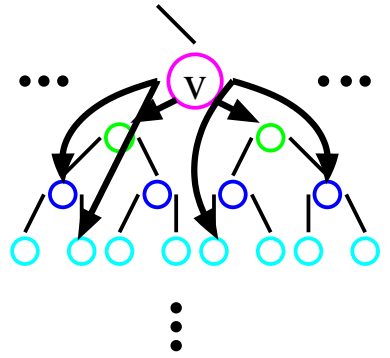
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$

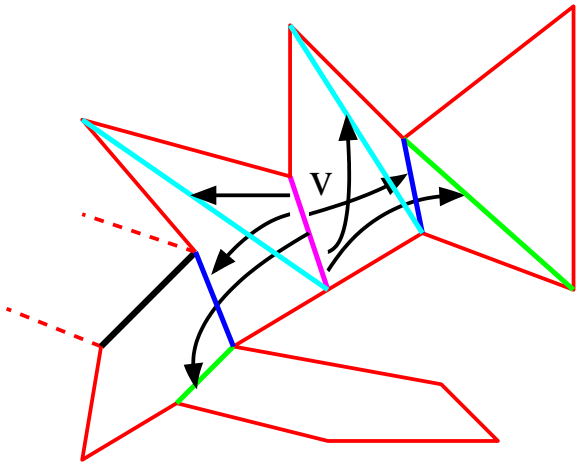




iii) Komplexität von $\widehat{G}$

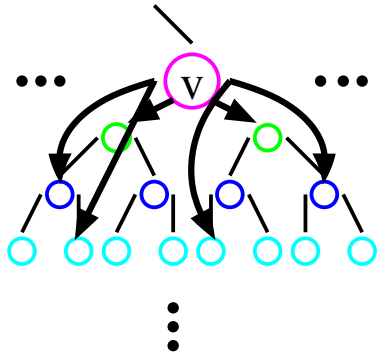
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$
- Balance: Teilbaum bei v hat $\geq \left(\frac{3}{2}\right)^{\texttt{height(v)}}$ Blätter (Tafel)

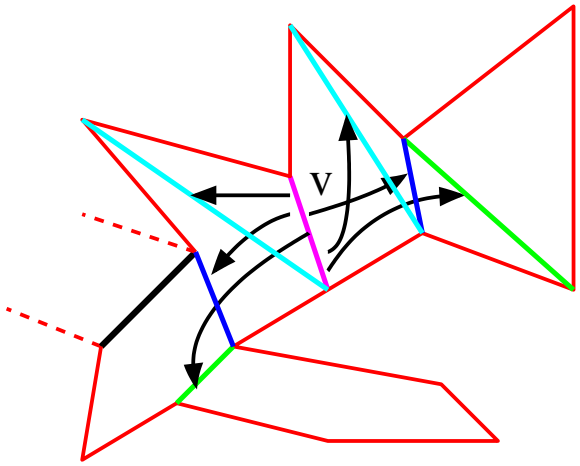




iii) Komplexität von $\widehat{G}$

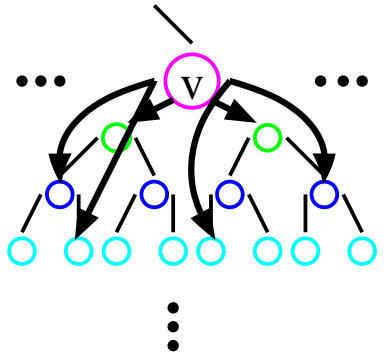
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$
- Balance: Teilbaum bei v hat $\geq \left(\frac{3}{2}\right)^{\texttt{height(v)}}$ Blätter (Tafel)
- Anzahl Knoten der Höhe h: $\leq \frac{n}{\left(\frac{3}{2}\right)^h} = n\frac{2^h}{3}$

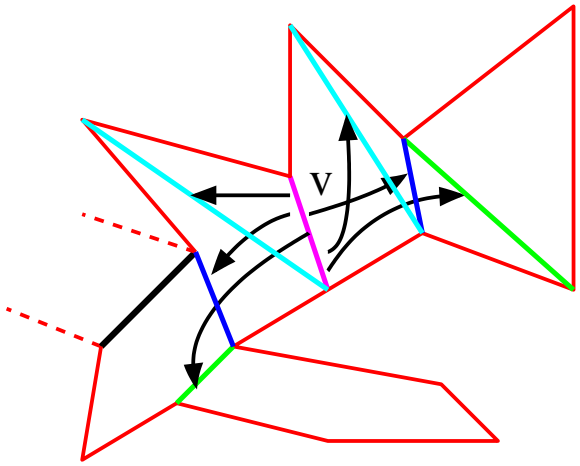




iii) Komplexität von $\widehat{G}$

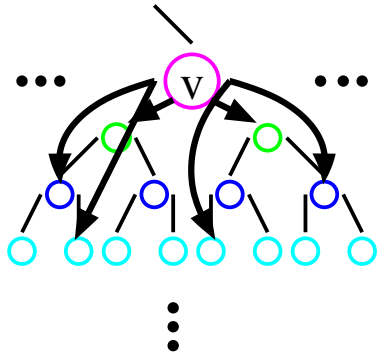
- Untere Kanten von v aus: max $2 \times \texttt{height(v)}$
- Balance: Teilbaum bei v hat $\geq \left(\frac{3}{2}\right)^{\texttt{height(v)}}$ Blätter (Tafel)
- Anzahl Knoten der Höhe h: $\leq \frac{n}{\left(\frac{3}{2}\right)^h} = n\frac{2^h}{3}$
- Sum. über alle Höhen: $\sum_{h=1}^{\log_{\frac{3}{2}}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

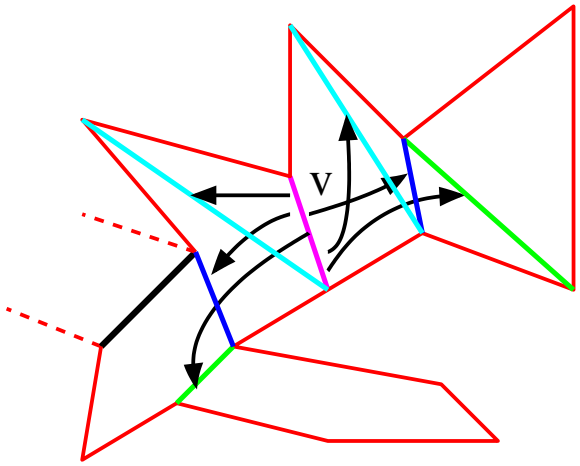




iii) Komplexität von $\widehat{G}$

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





iii) Komplexität von $\widehat{G}$

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