Algorithmen zur Visualisierung von Graphen

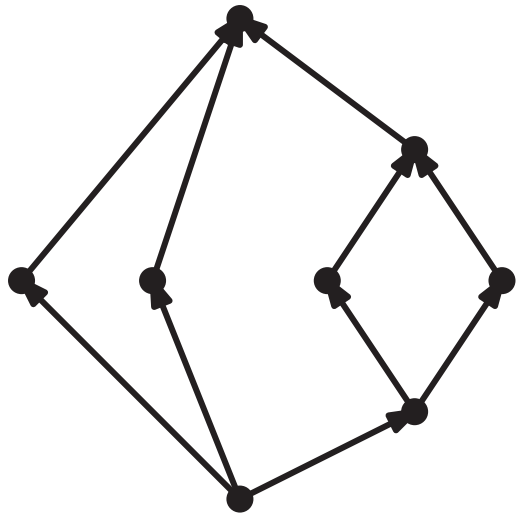
Serienparallele Graphen und Gitterzeichnungen

Vorlesung im Wintersemester 2010/2011

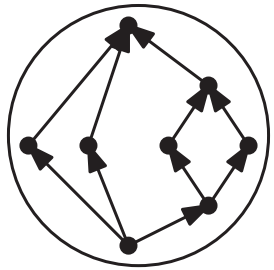
Robert Görke

02.02.2011

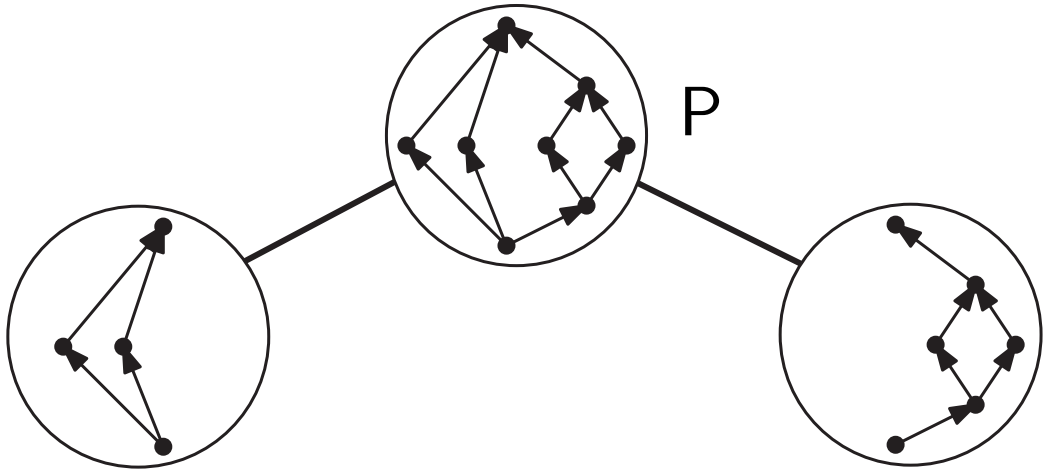
Serienparallele Graphen



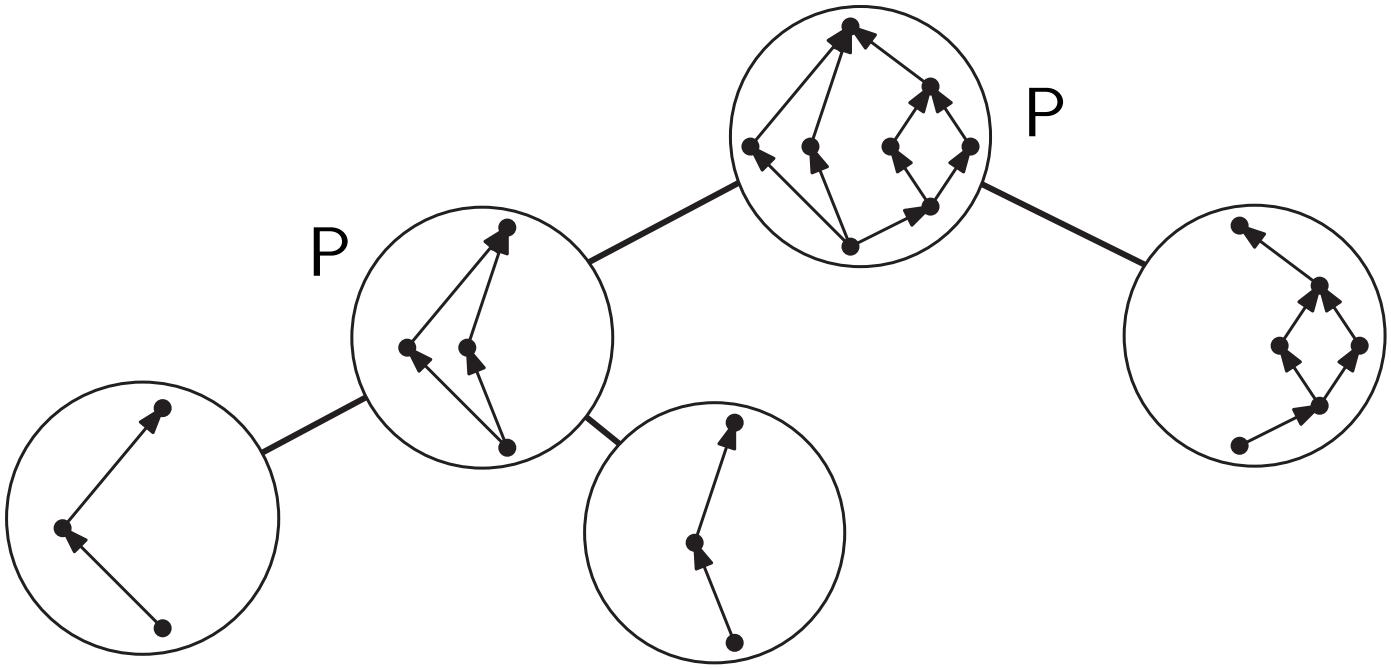
Serienparallele Graphen: Dekompositionsbaum



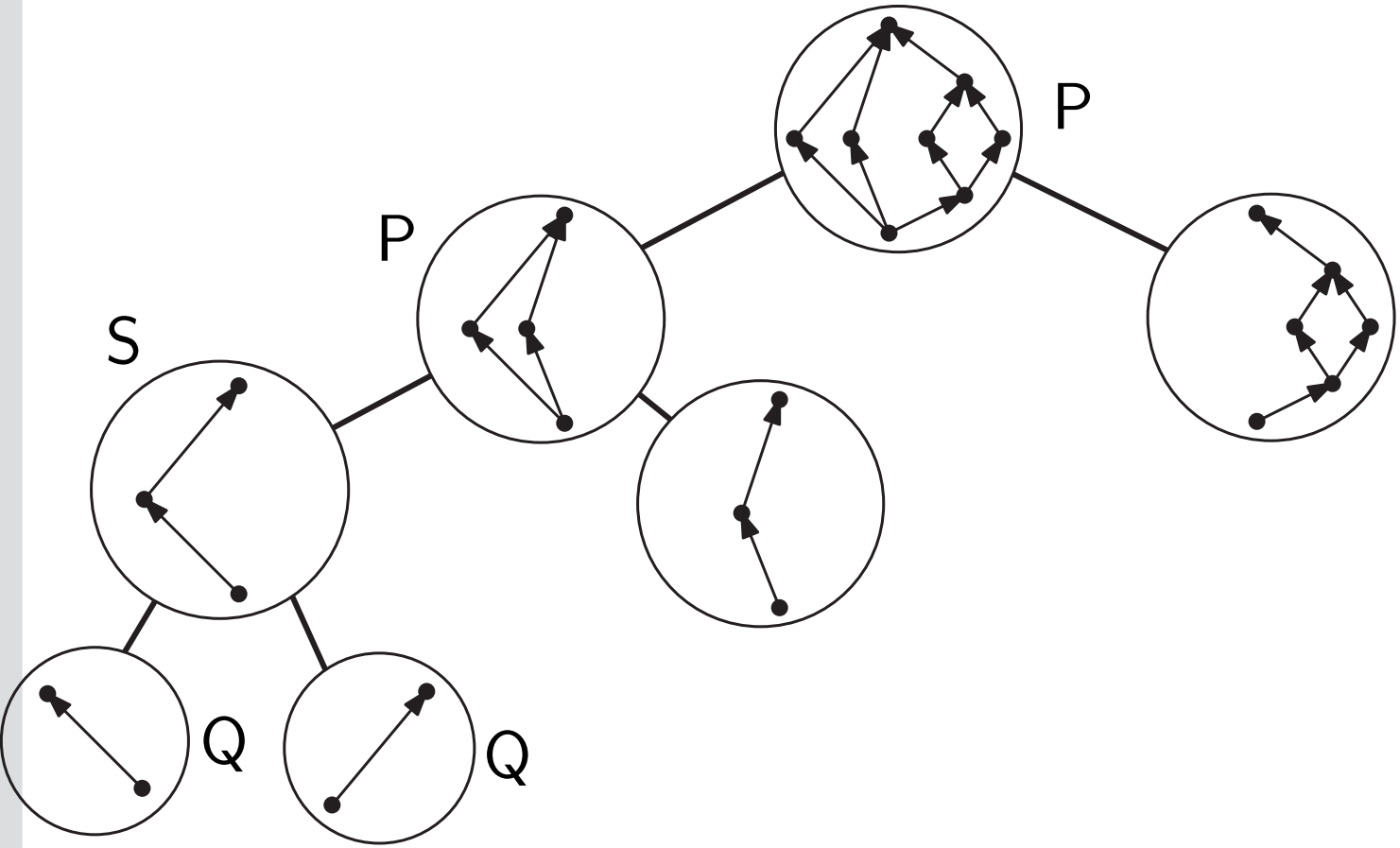
Serienparallele Graphen: Dekompositionsbaum



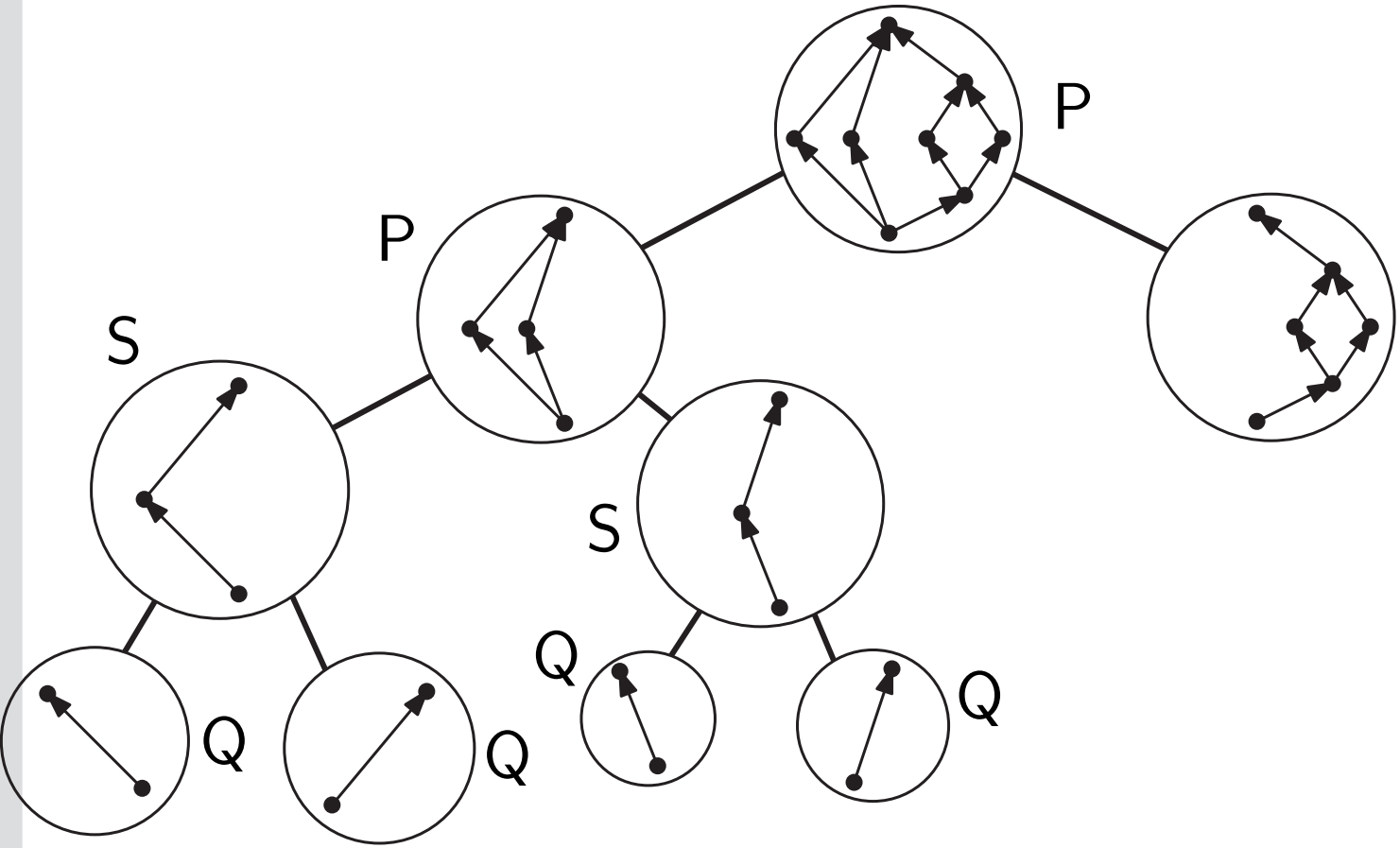
Serienparallele Graphen: Dekompositionsbaum



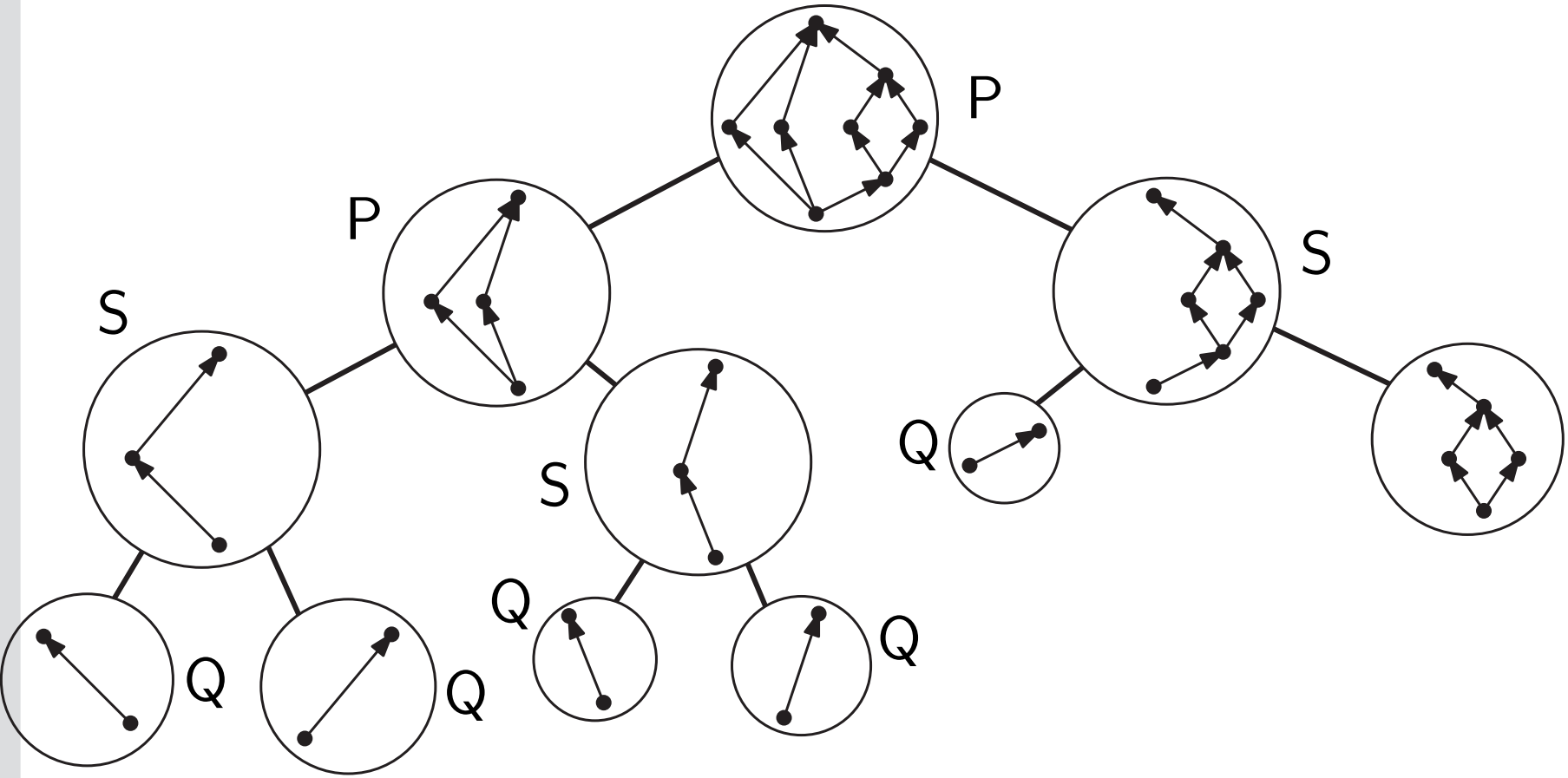
Serienparallele Graphen: Dekompositionsbaum



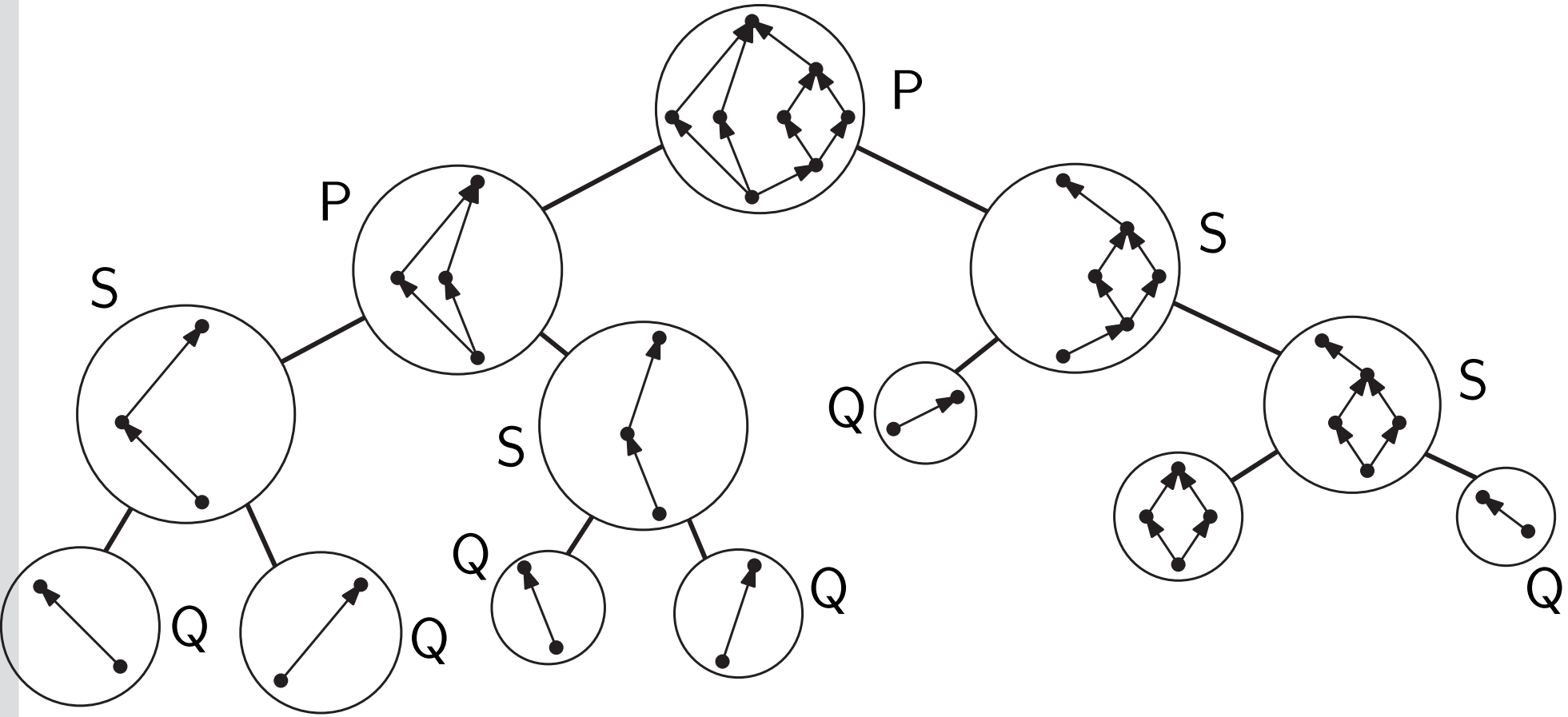
Serienparallele Graphen: Dekompositionsbaum



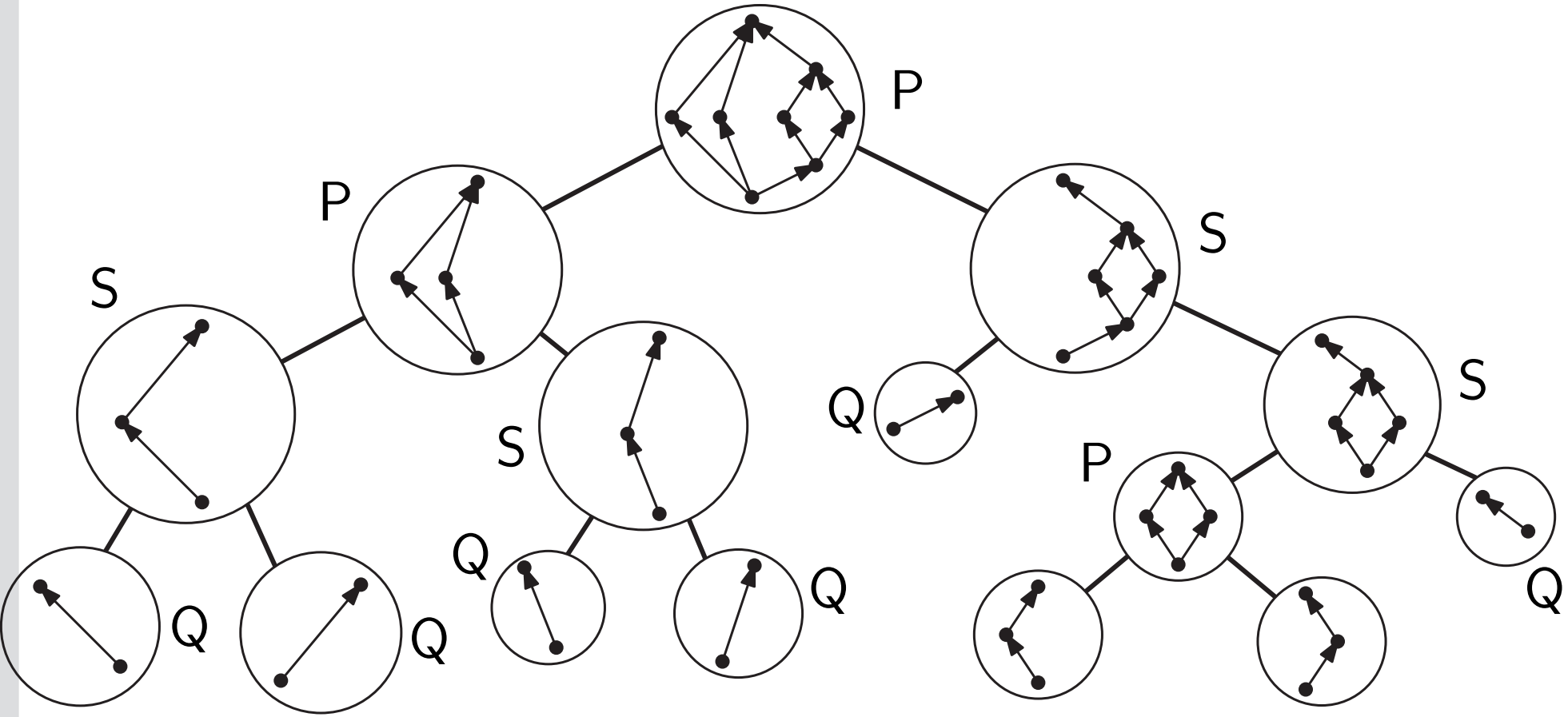
Serienparallele Graphen: Dekompositionsbaum



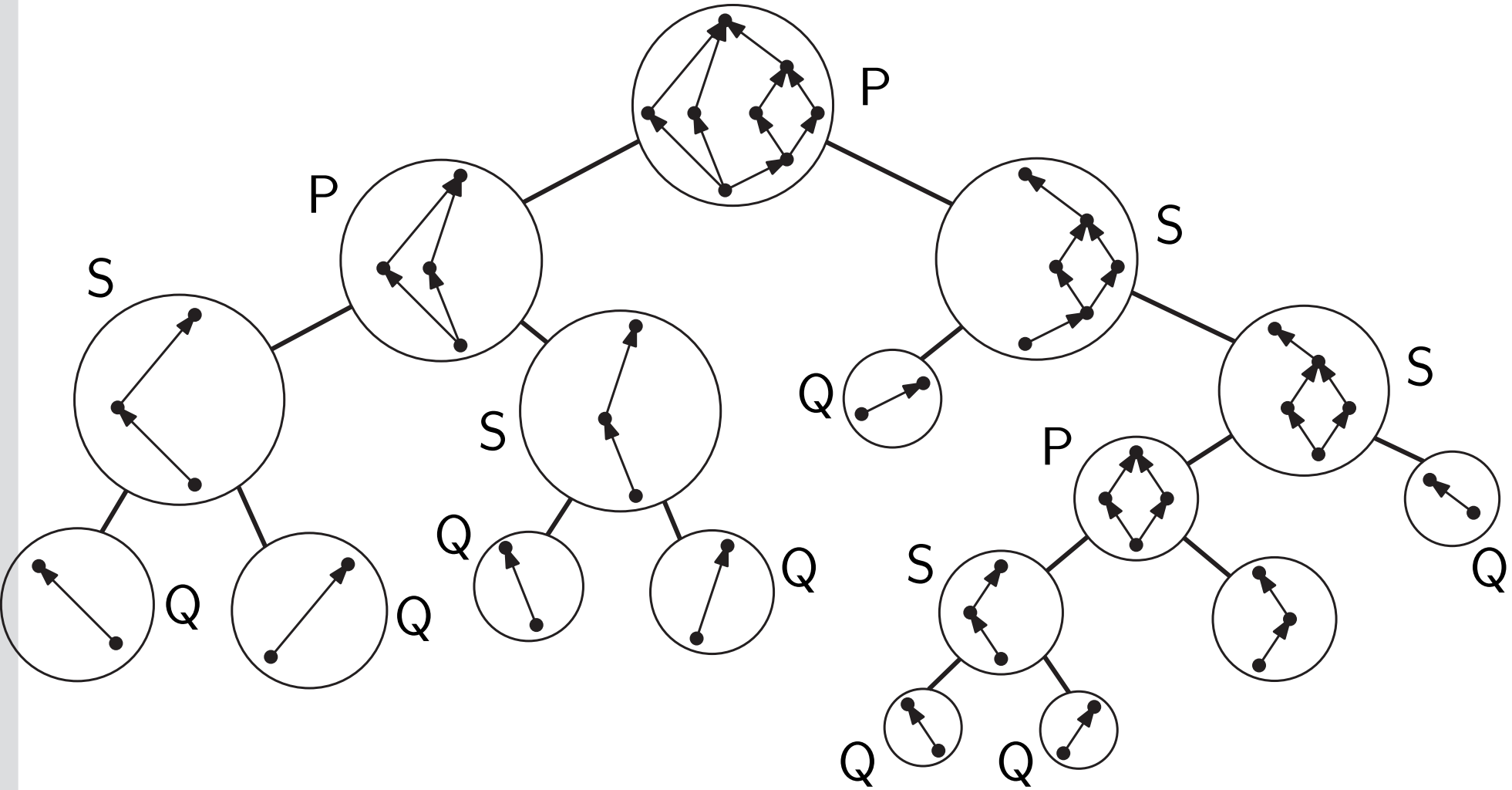
Serienparallele Graphen: Dekompositionsbaum



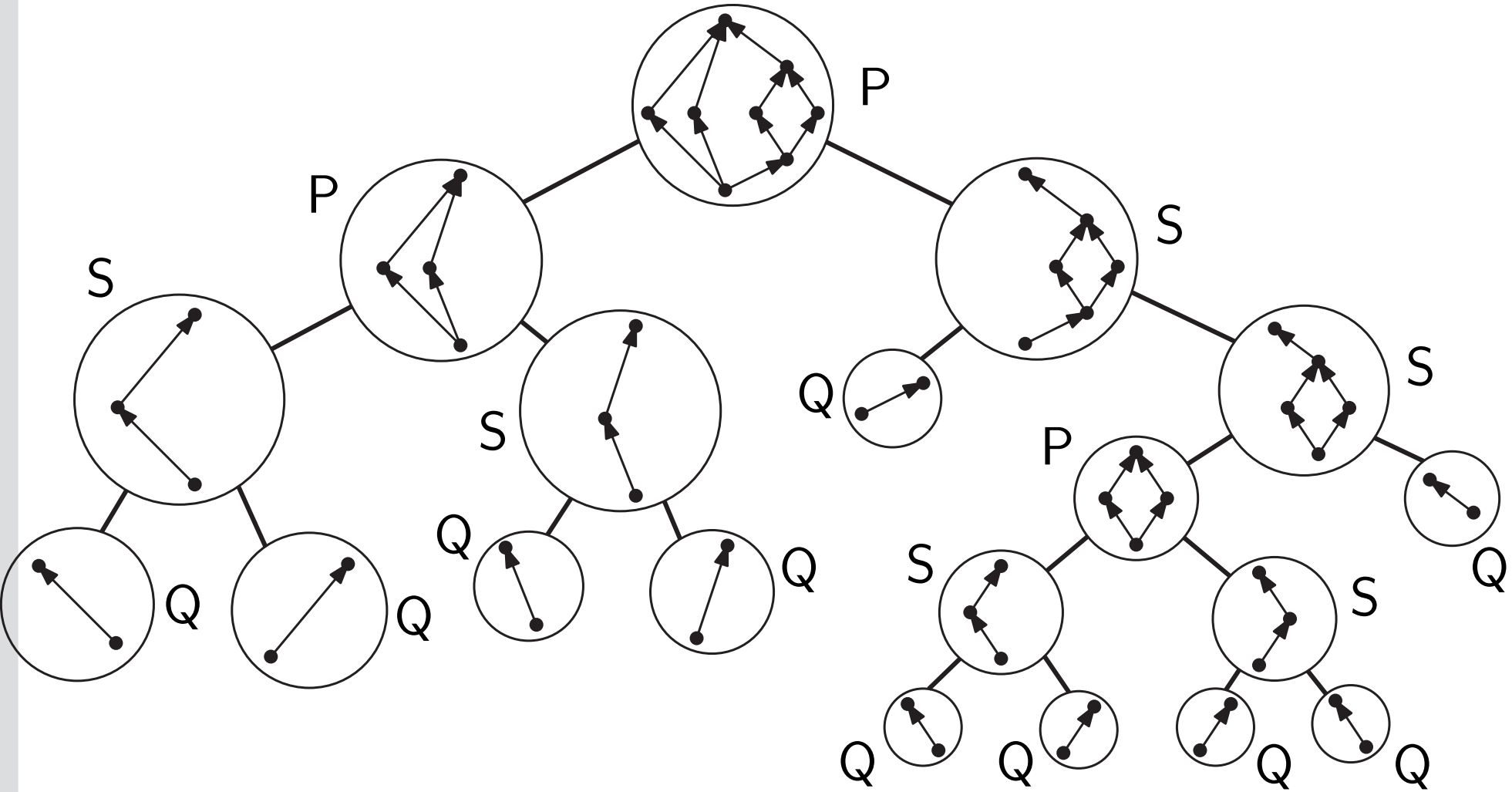
Serienparallele Graphen: Dekompositionsbaum



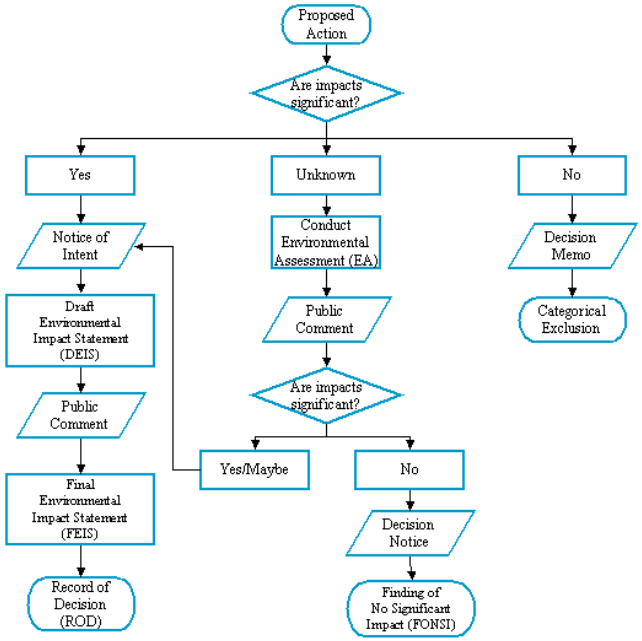
Serienparallele Graphen: Dekompositionsbaum



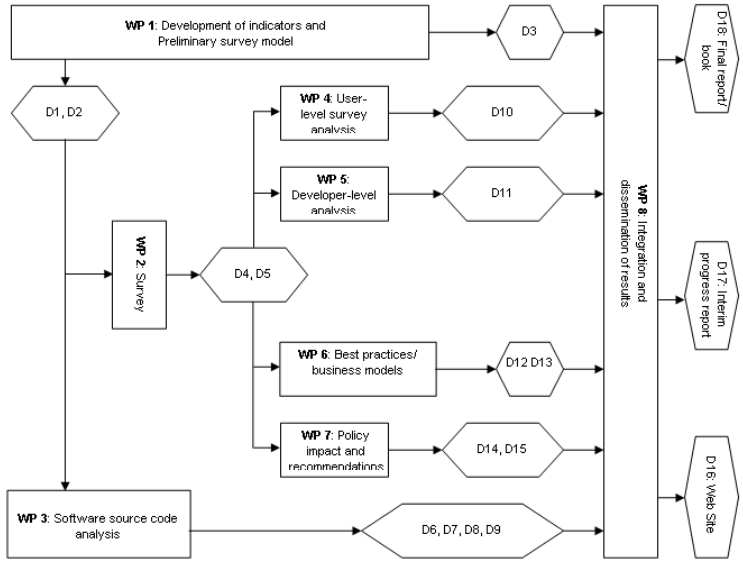
Serienparallele Graphen: Dekompositionsbaum



SP-Graphen in Anwendungen



Ablaufdiagramme



PERT-Diagramme
(Program Evaluation and Review Technique)

Außerdem: Linearzeitalgorithmen für sonst NP-vollständige Probleme (z.B. Maximum Independent Set)

Untere Schranke für die Fläche

Satz: (Bertolazzi et al. '94)

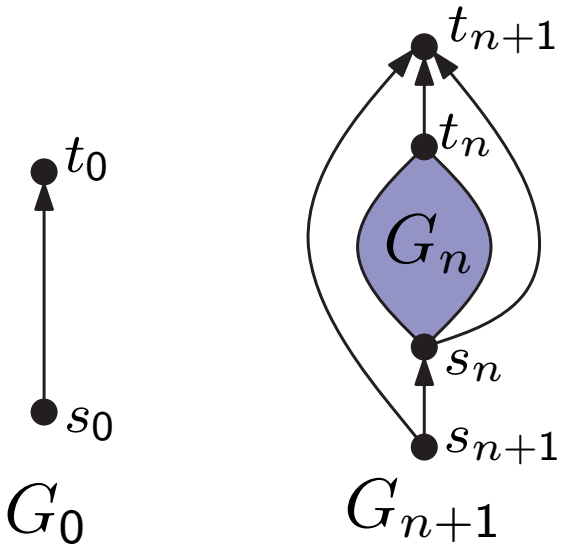
Es gibt eine Familie G_n von Graphen mit $2n$ Knoten, deren Zeichnung einen Platzbedarf von $\Omega(4^n)$ hat.

Untere Schranke für die Fläche

Satz: (Bertolazzi et al. '94)

Es gibt eine Familie G_n von Graphen mit $2n$ Knoten, deren Zeichnung einen Platzbedarf von $\Omega(4^n)$ hat.

Beweis:

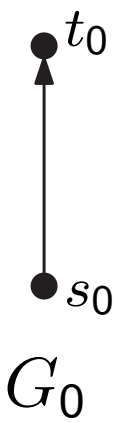


Untere Schranke für die Fläche

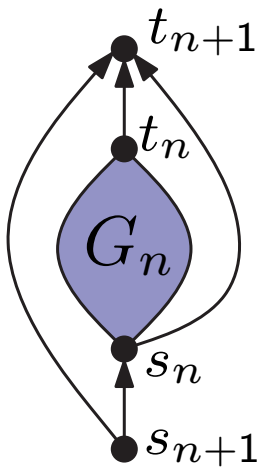
Satz: (Bertolazzi et al. '94)

Es gibt eine Familie G_n von Graphen mit $2n$ Knoten, deren Zeichnung einen Platzbedarf von $\Omega(4^n)$ hat.

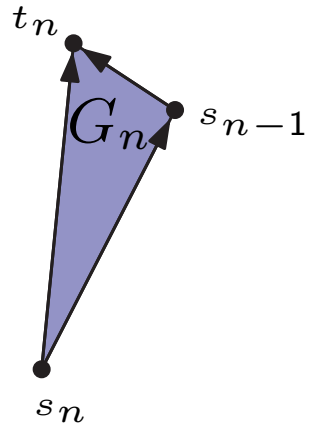
Beweis:



G_0



G_{n+1}



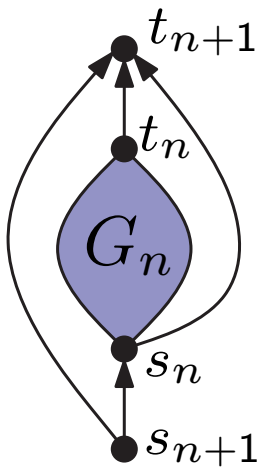
Untere Schranke für die Fläche

Satz: (Bertolazzi et al. '94)

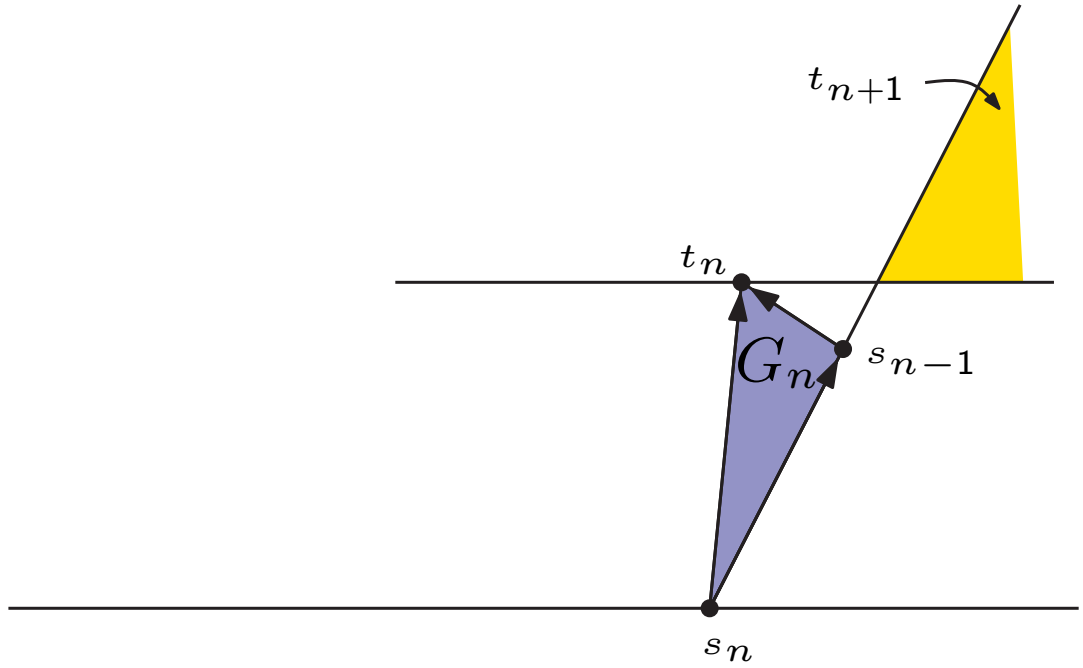
Es gibt eine Familie G_n von Graphen mit $2n$ Knoten, deren Zeichnung einen Platzbedarf von $\Omega(4^n)$ hat.

Beweis:

G_0



G_{n+1}



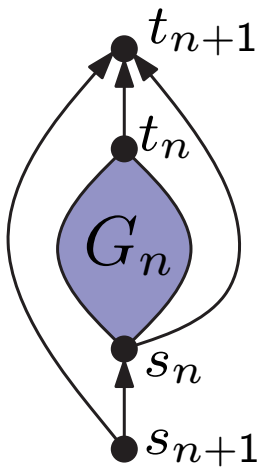
Untere Schranke für die Fläche

Satz: (Bertolazzi et al. '94)

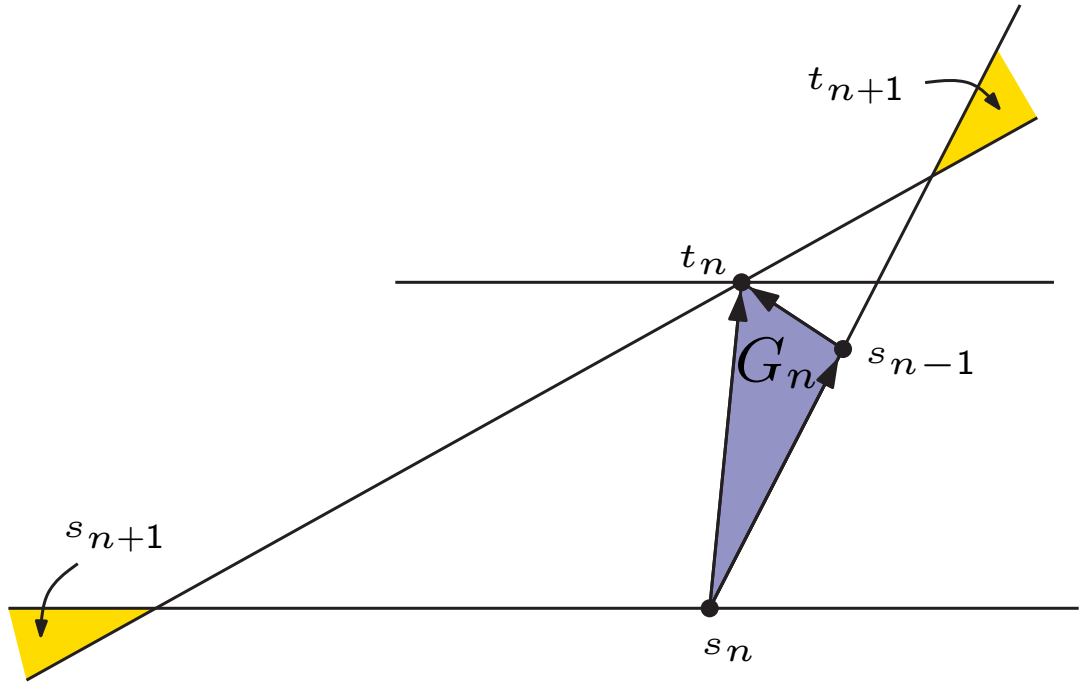
Es gibt eine Familie G_n von Graphen mit $2n$ Knoten, deren Zeichnung einen Platzbedarf von $\Omega(4^n)$ hat.

Beweis:

G_0



G_{n+1}



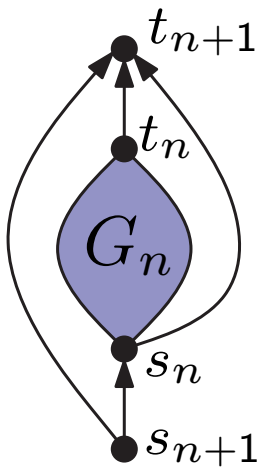
Untere Schranke für die Fläche

Satz: (Bertolazzi et al. '94)

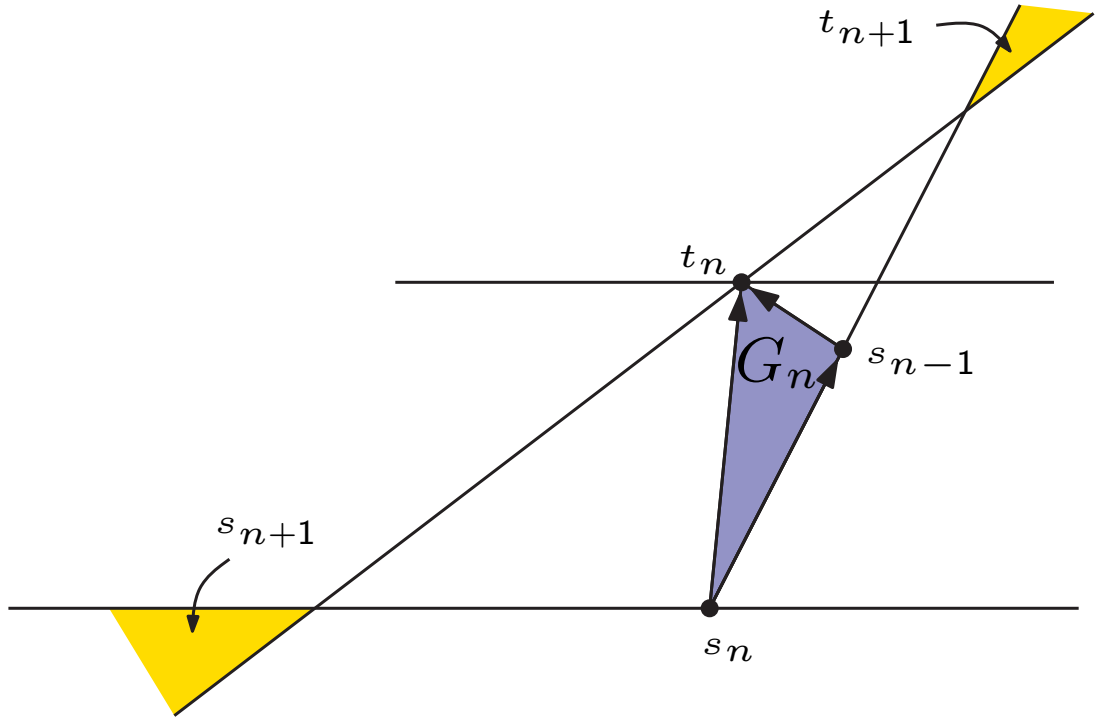
Es gibt eine Familie G_n von Graphen mit $2n$ Knoten, deren Zeichnung einen Platzbedarf von $\Omega(4^n)$ hat.

Beweis:

G_0



G_{n+1}



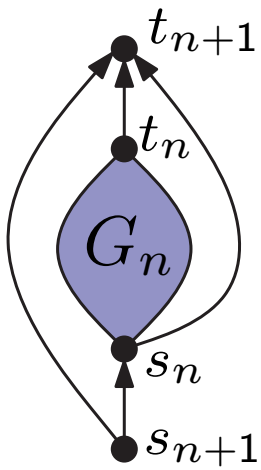
Untere Schranke für die Fläche

Satz: (Bertolazzi et al. '94)

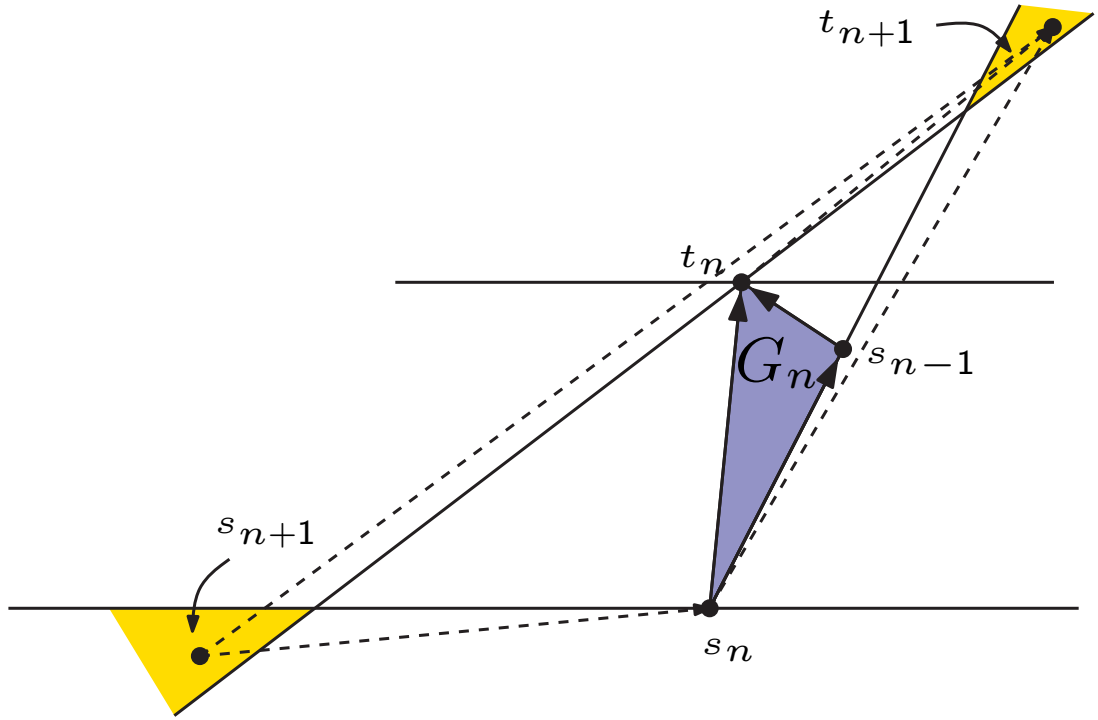
Es gibt eine Familie G_n von Graphen mit $2n$ Knoten, deren Zeichnung einen Platzbedarf von $\Omega(4^n)$ hat.

Beweis:

G_0



G_{n+1}



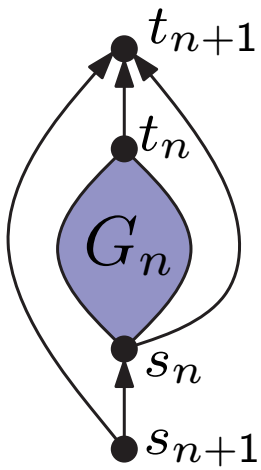
Untere Schranke für die Fläche

Satz: (Bertolazzi et al. '94)

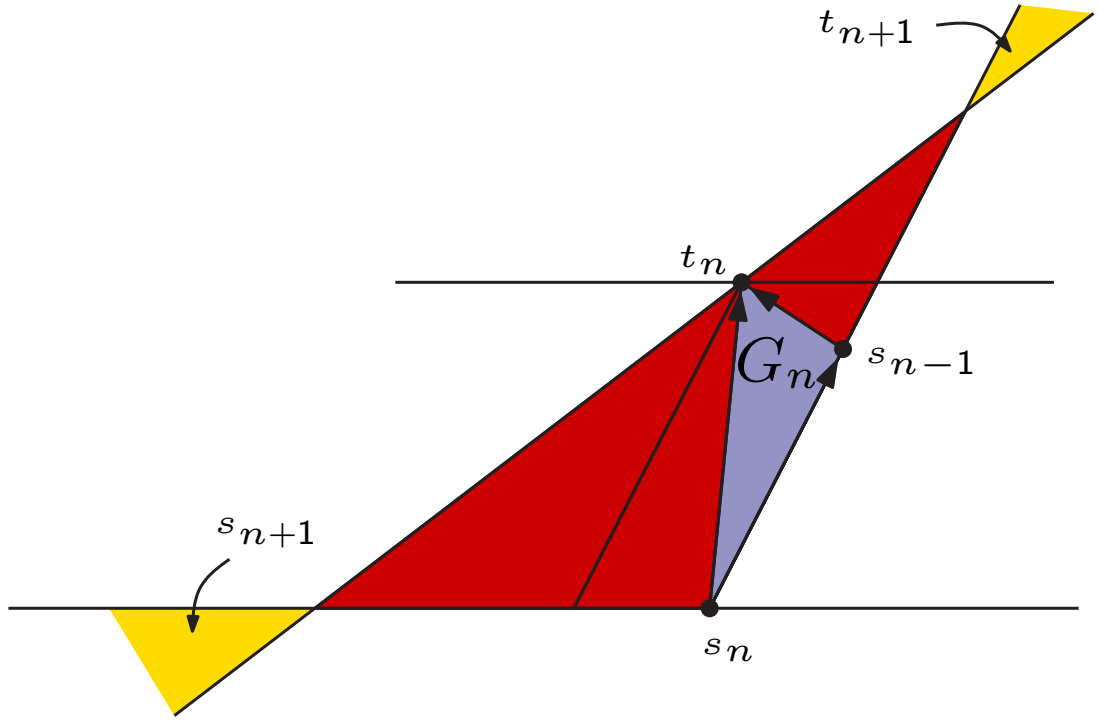
Es gibt eine Familie G_n von Graphen mit $2n$ Knoten, deren Zeichnung einen Platzbedarf von $\Omega(4^n)$ hat.

Beweis:

G_0



G_{n+1}

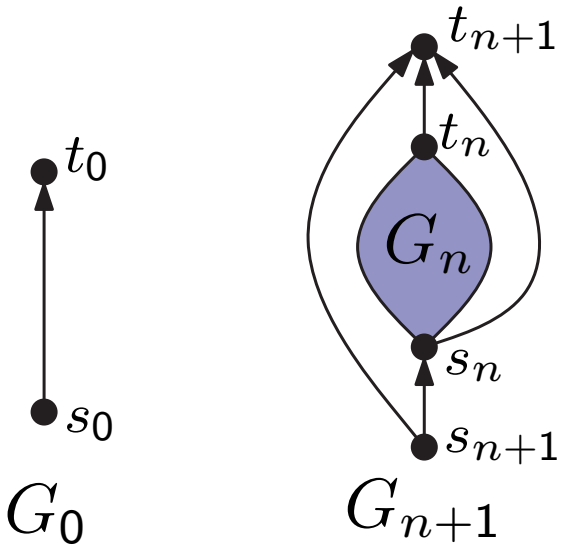


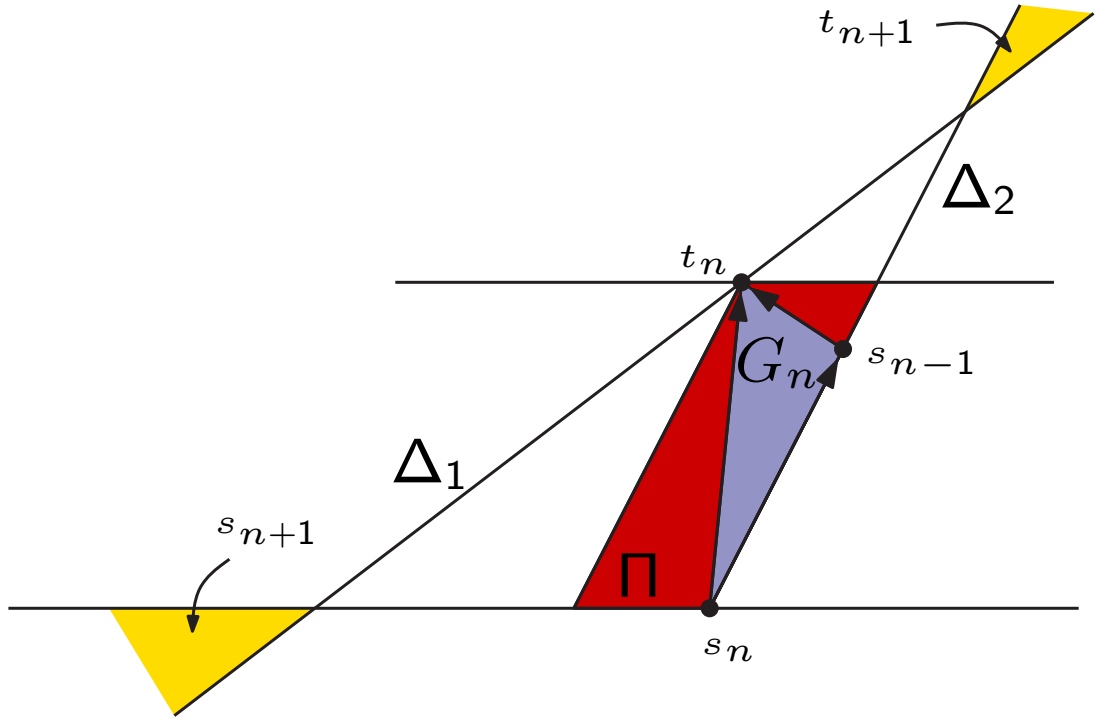
Untere Schranke für die Fläche

Satz: (Bertolazzi et al. '94)

Es gibt eine Familie G_n von Graphen mit $2n$ Knoten, deren Zeichnung einen Platzbedarf von $\Omega(4^n)$ hat.

Beweis:



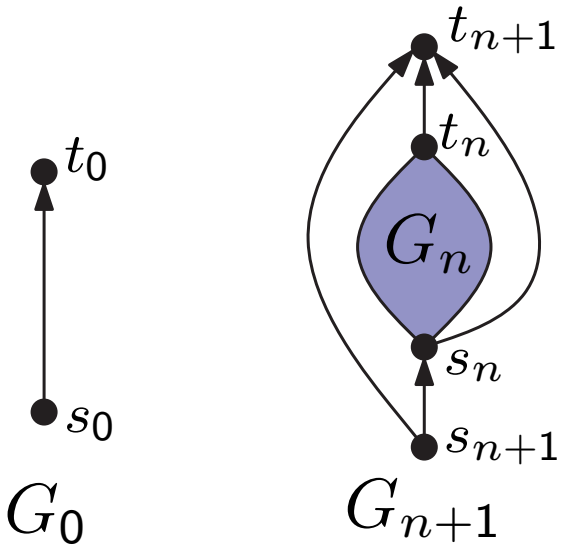


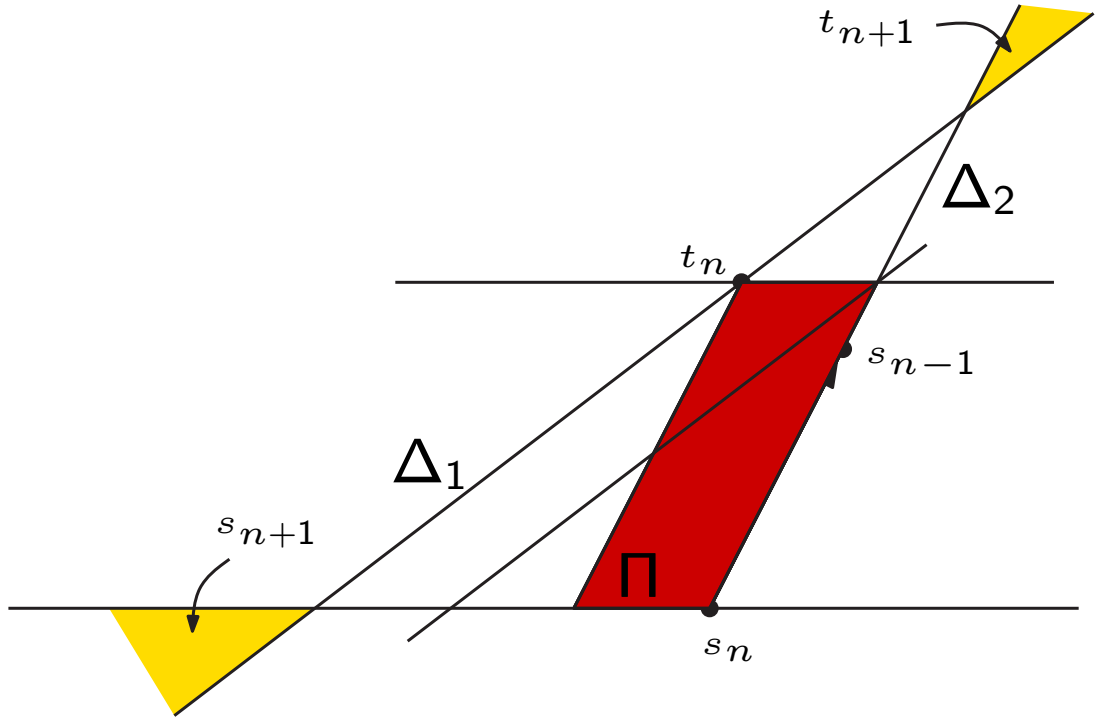
Untere Schranke für die Fläche

Satz: (Bertolazzi et al. '94)

Es gibt eine Familie G_n von Graphen mit $2n$ Knoten, deren Zeichnung einen Platzbedarf von $\Omega(4^n)$ hat.

Beweis:



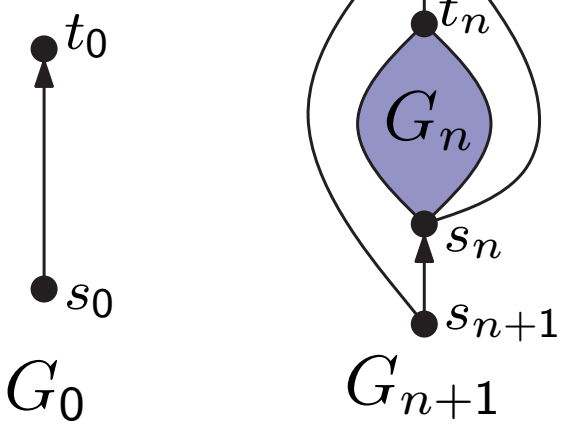


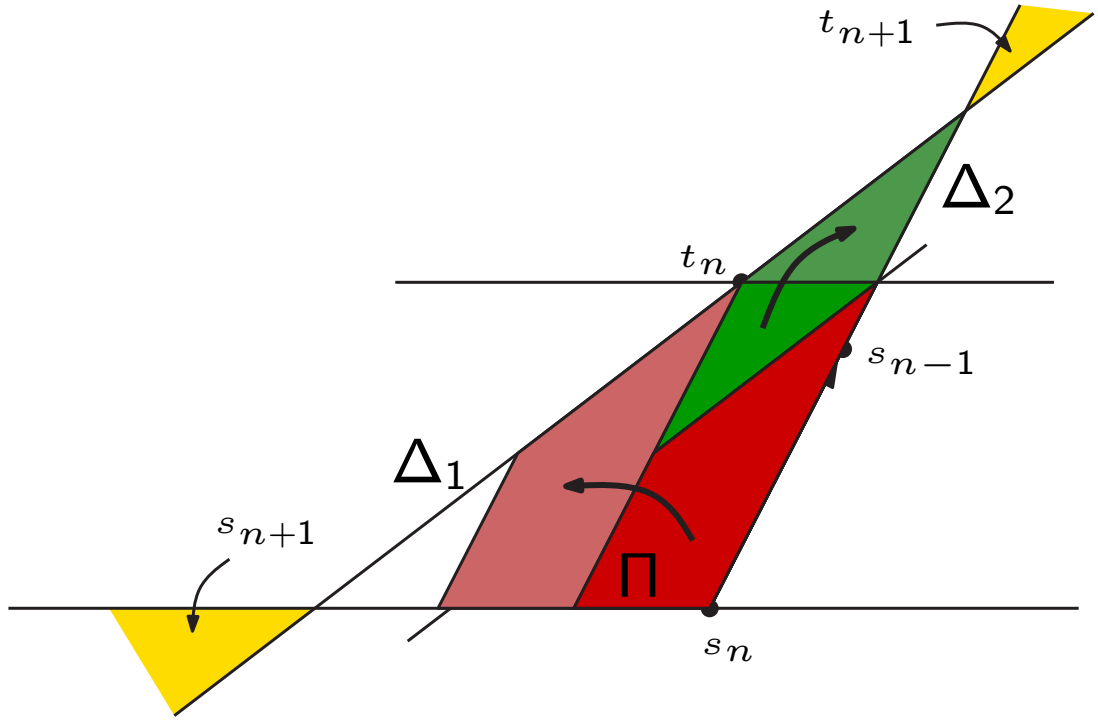
Untere Schranke für die Fläche

Satz: (Bertolazzi et al. '94)

Es gibt eine Familie G_n von Graphen mit $2n$ Knoten, deren Zeichnung einen Platzbedarf von $\Omega(4^n)$ hat.

Beweis:



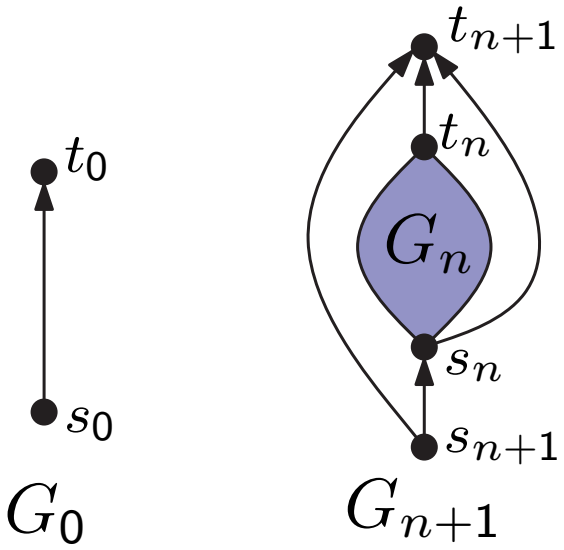


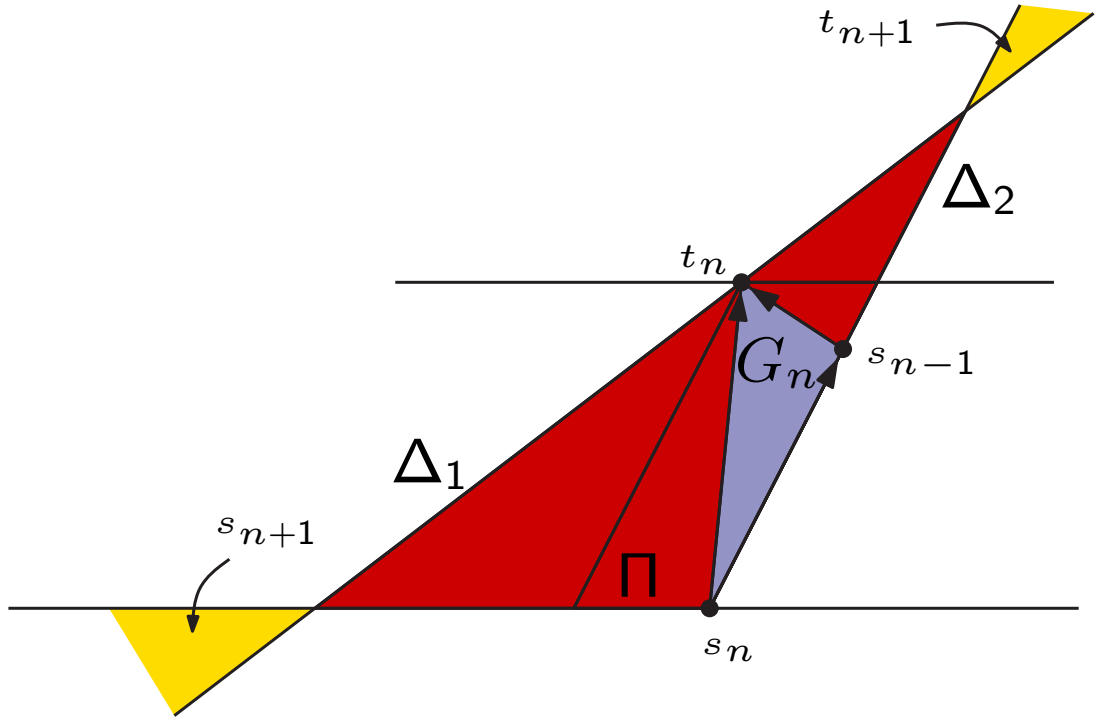
Untere Schranke für die Fläche

Satz: (Bertolazzi et al. '94)

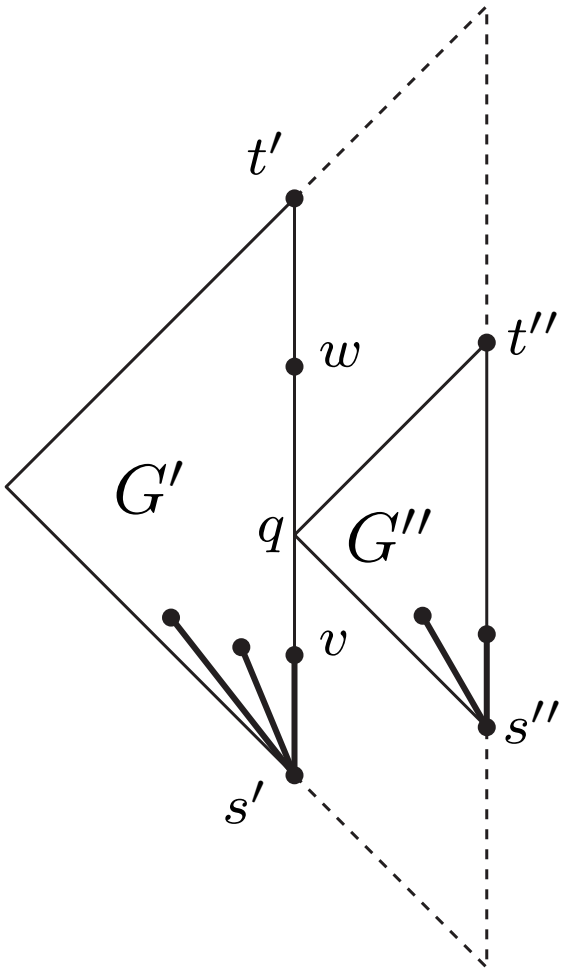
Es gibt eine Familie G_n von Graphen mit $2n$ Knoten, deren Zeichnung einen Platzbedarf von $\Omega(4^n)$ hat.

Beweis:

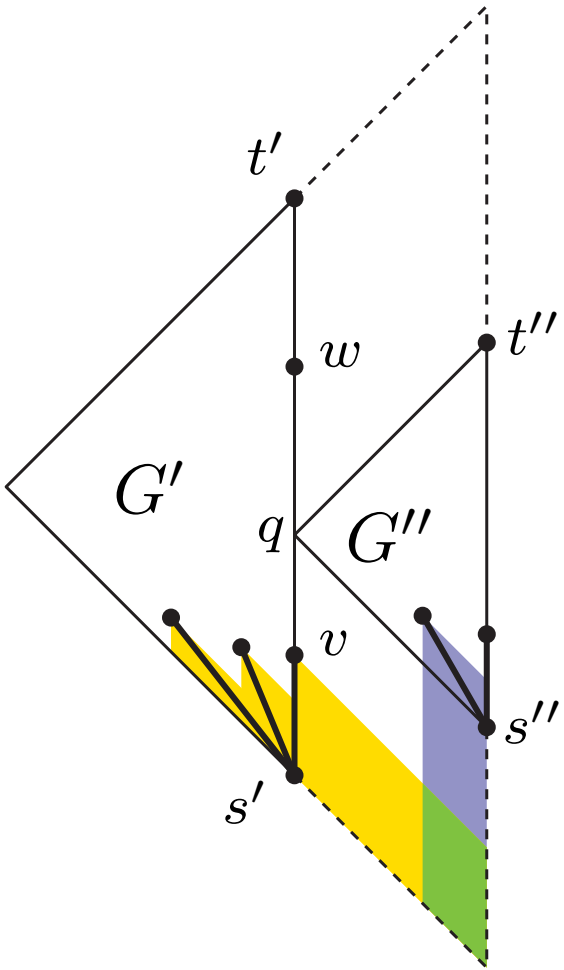




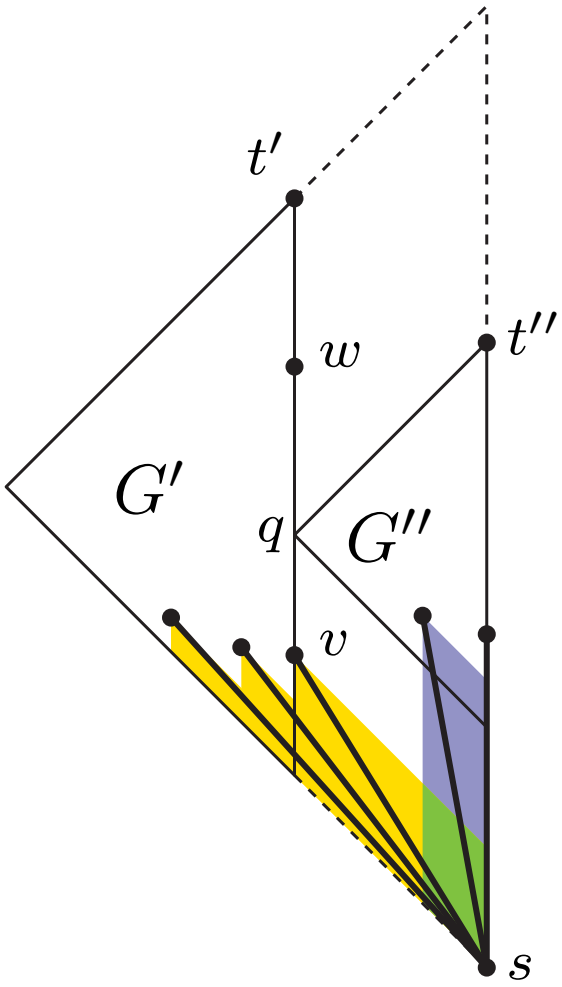
P-Komposition



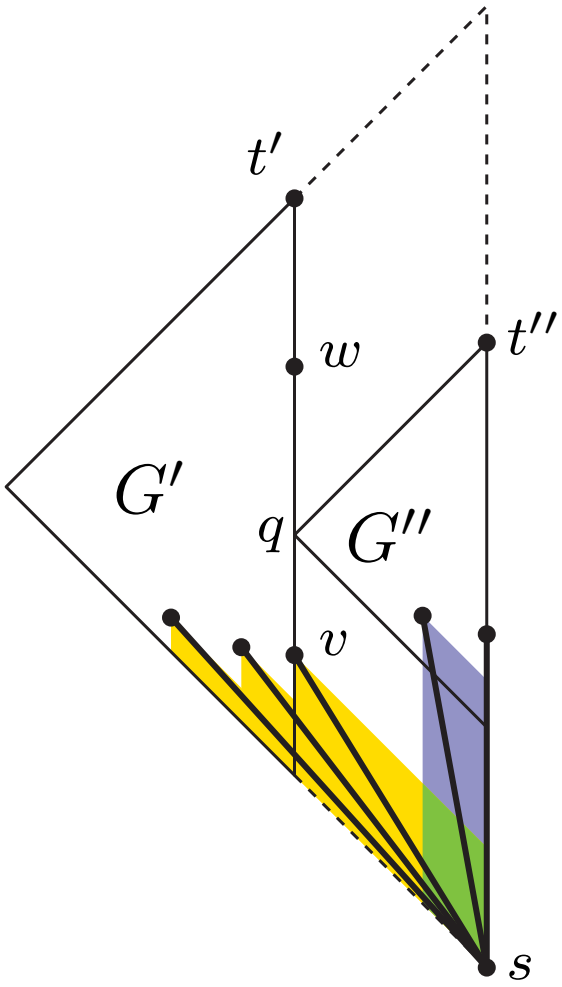
P-Komposition

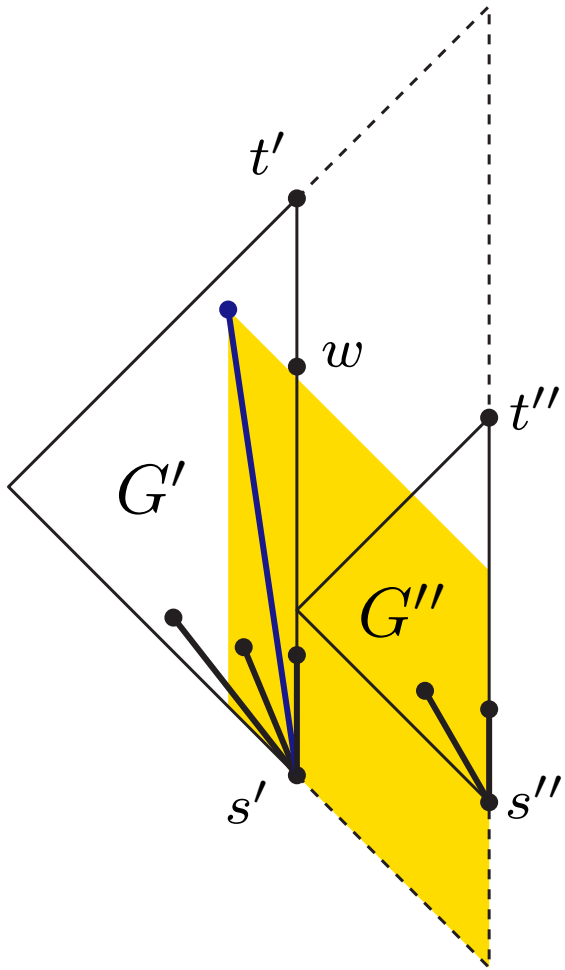


P-Komposition

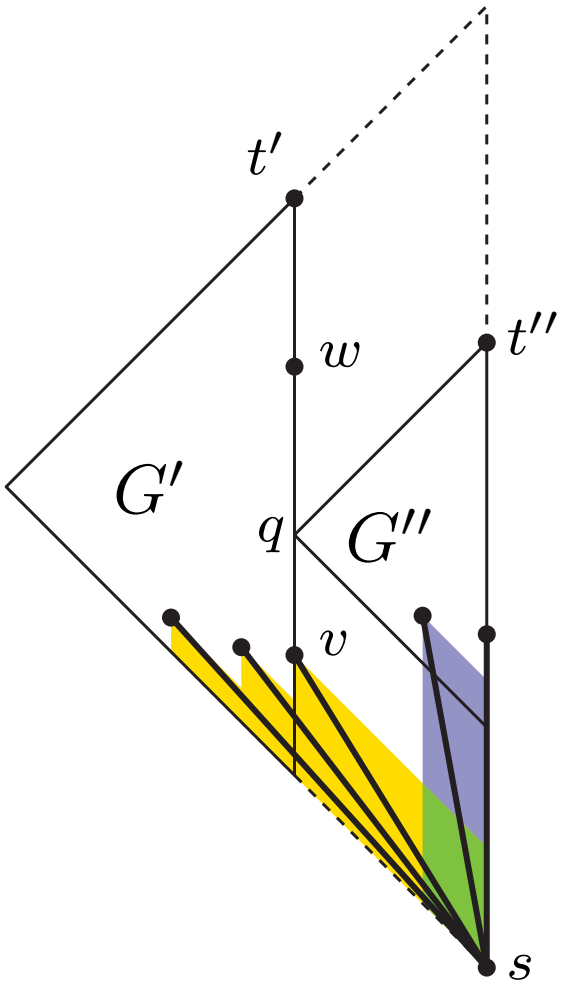


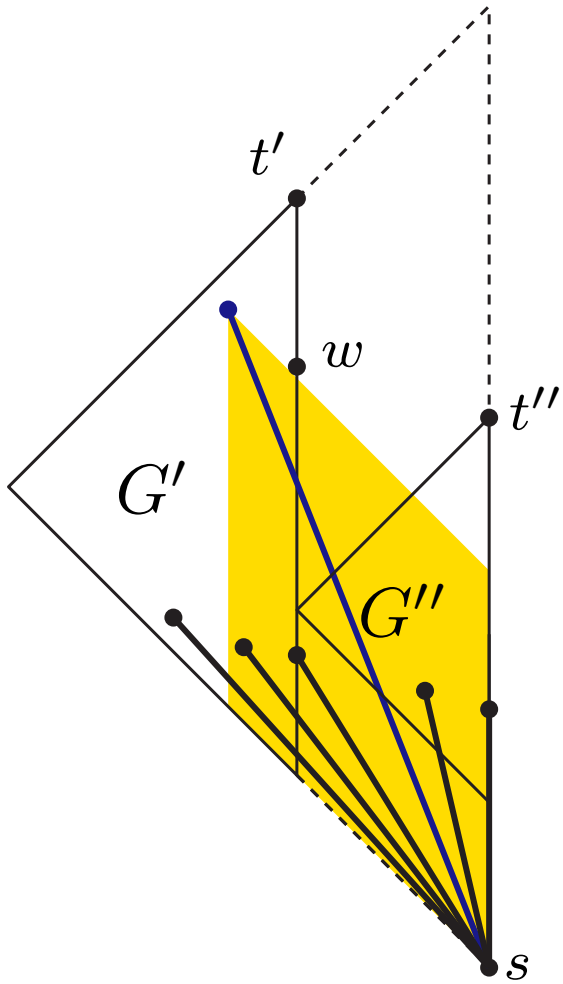
P-Komposition



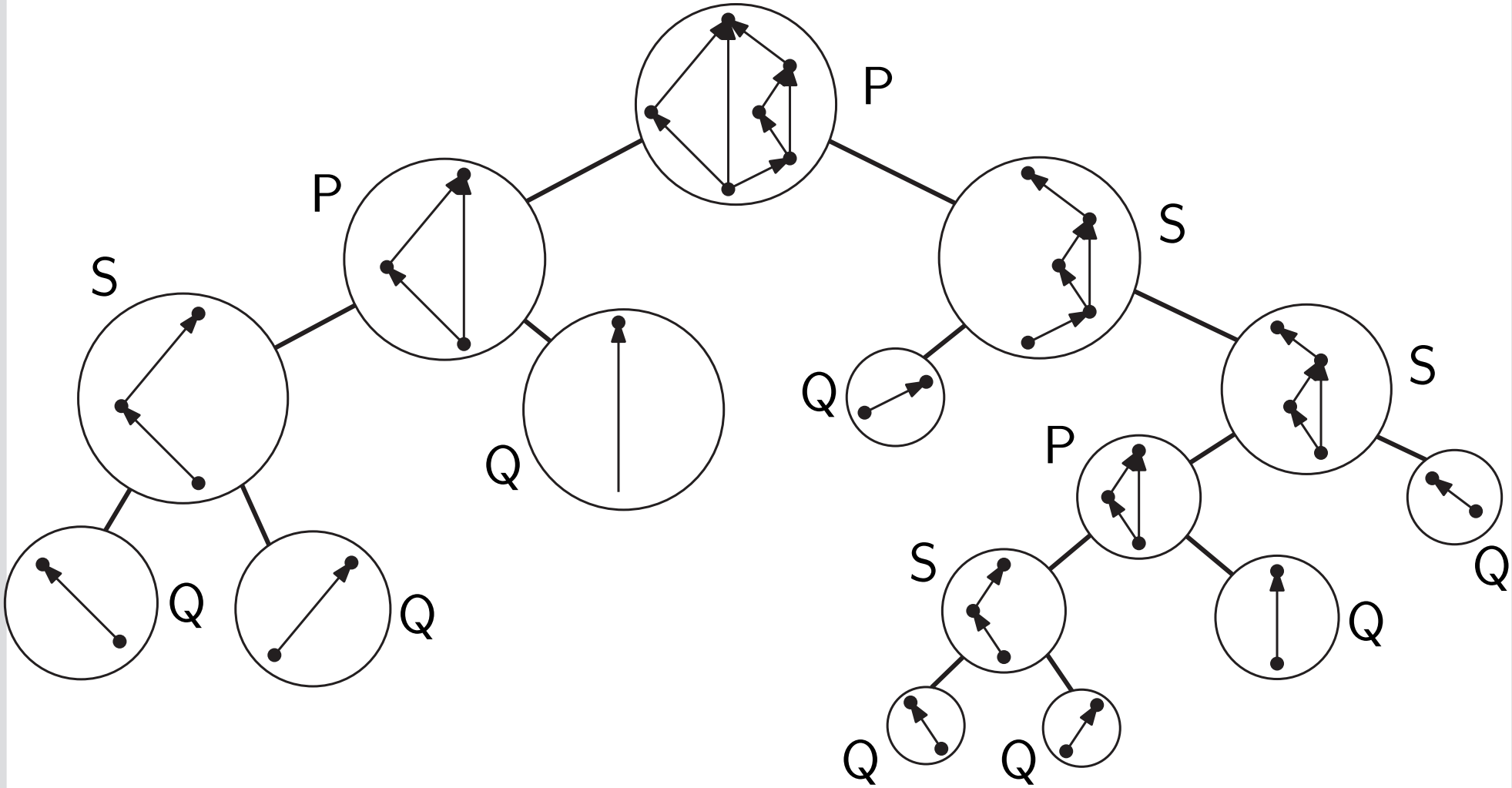


P-Komposition





Sichtbarkeitsrepräsentation



Symmetrien in SP-Zeichnungen

Definition: Automorphismen eines DAG

Ein Automorphismus eines DAG $G = (V, E)$ ist eine Knotenpermutation $\pi : V \to V$, die Adjazenzen respektiert und alle Kantenrichtungen erhält oder alle Kantenrichtungen umdreht:

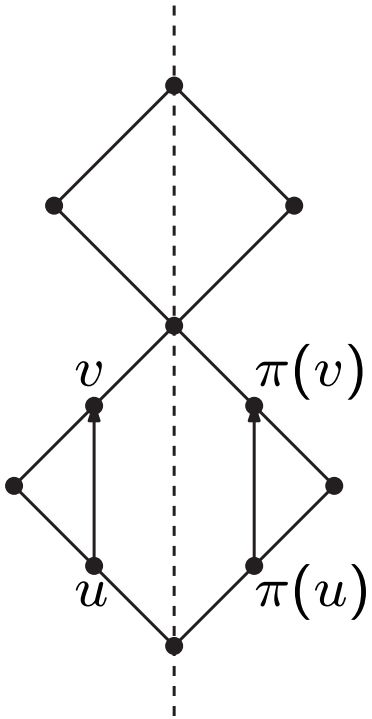
$$(u, v) \in E \Leftrightarrow (\pi(u), \pi(v)) \in E$$

oder

$$(u, v) \in E \Leftrightarrow (\pi(v), \pi(u)) \in E.$$

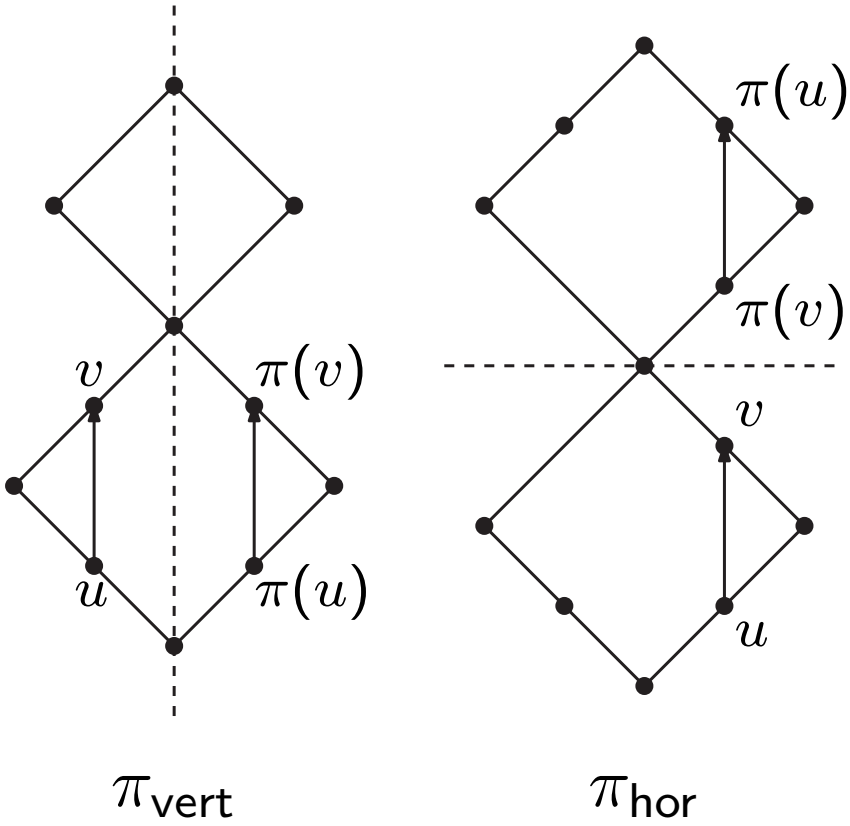
Die Automorphismen von G bilden mit der Hintereinanderausführung eine Gruppe.

Symmetrien in SP-Graphen

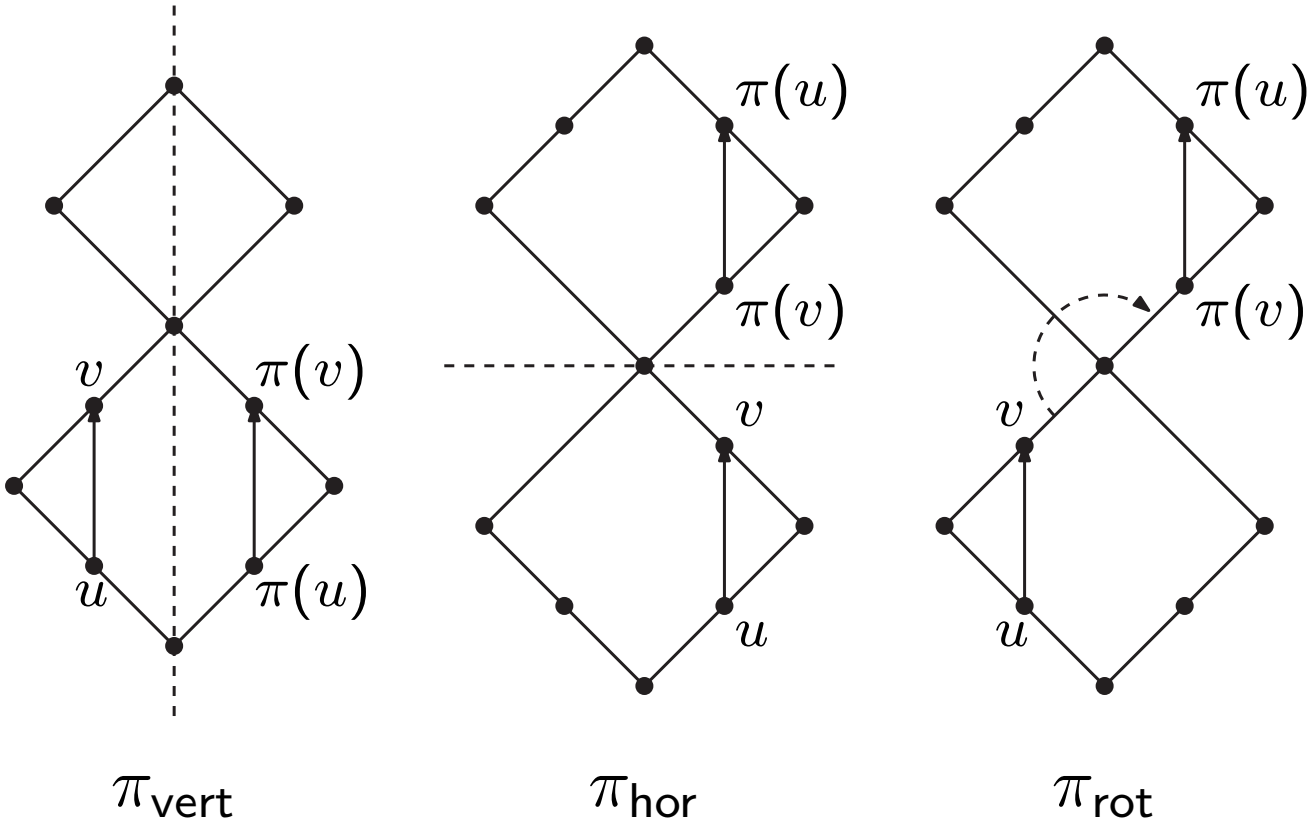


π_{vert}

Symmetrien in SP-Graphen

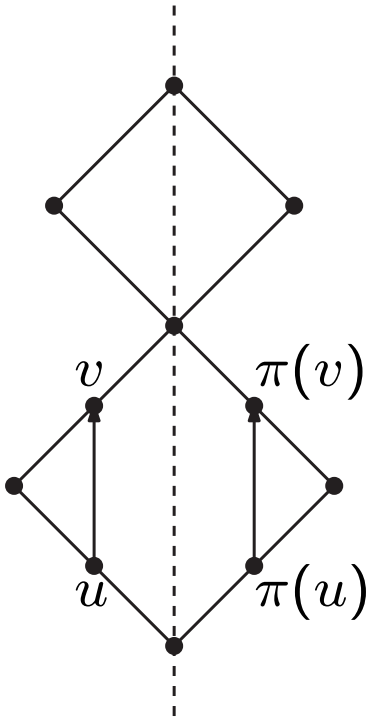


Symmetrien in SP-Graphen

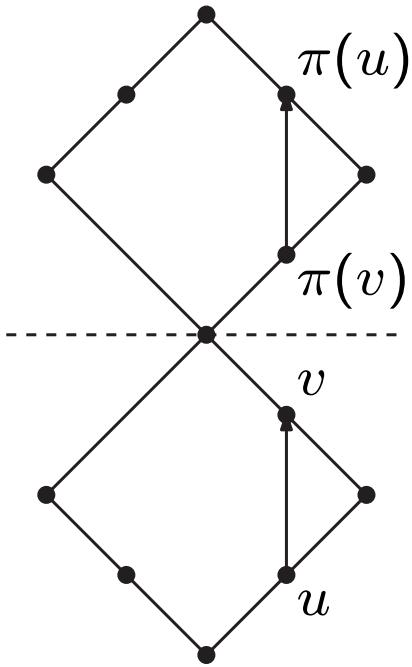


π_{vert} π_{hor} π_{rot}

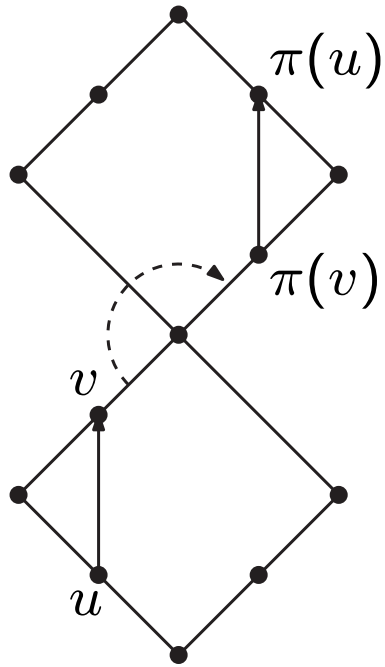
Symmetrien in SP-Graphen



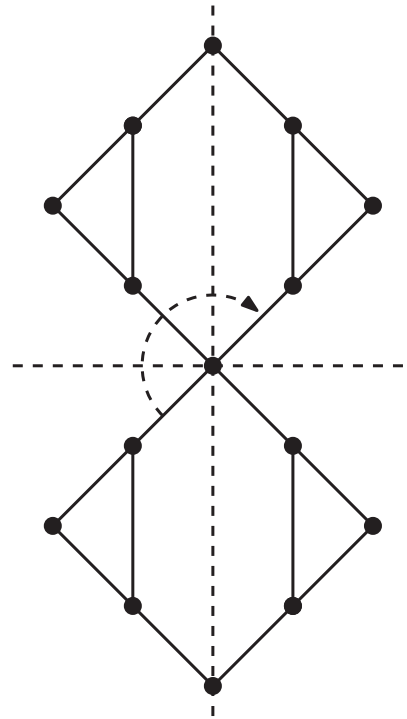
π_{vert}



π_{hor}



π_{rot}



$\{\pi_{\text{vert}}, \pi_{\text{hor}}, \pi_{\text{rot}}\}$

Symmetrien in SP-Zeichnungen

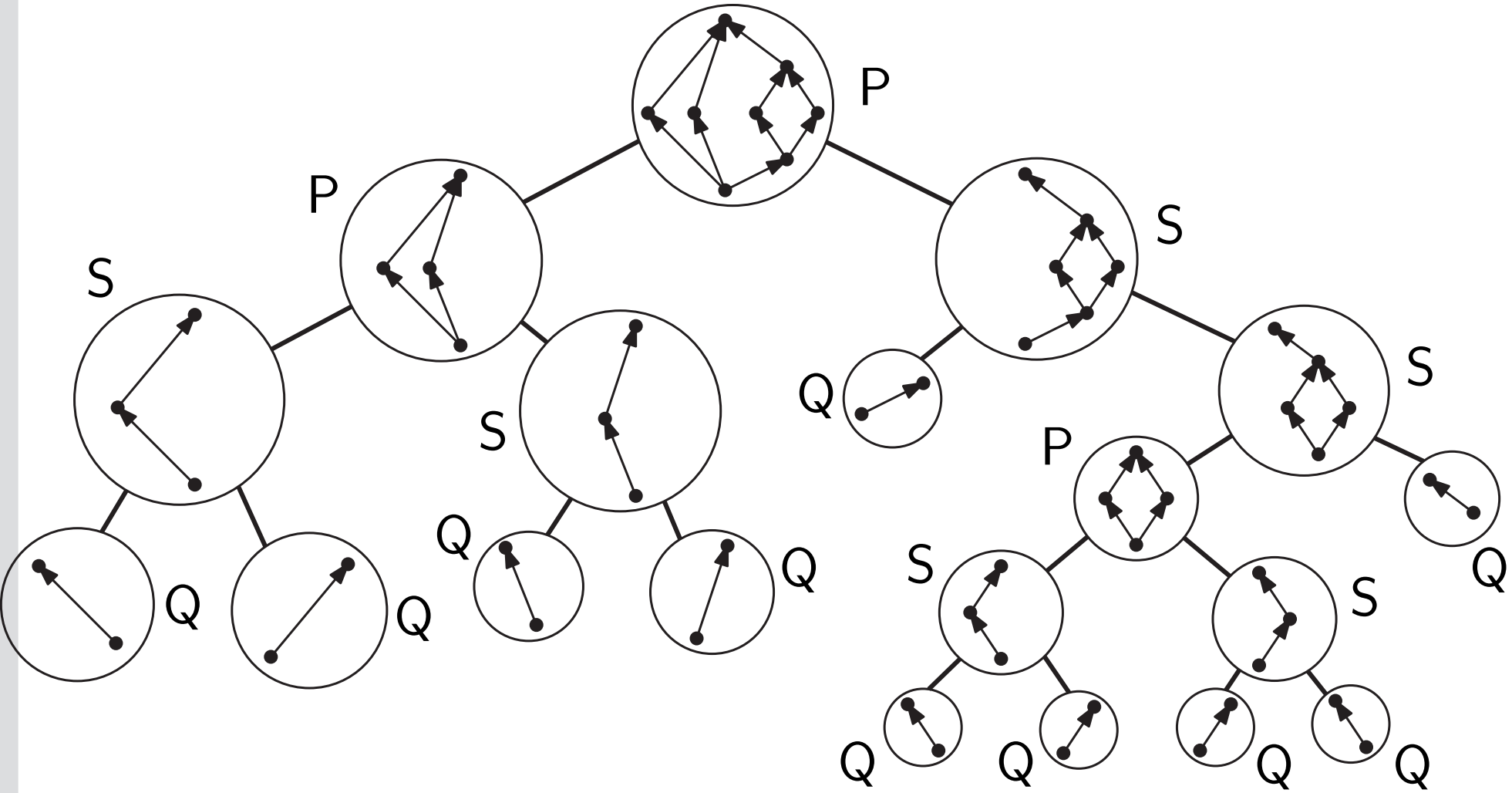
Karlsruher Institut für Technologie

Satz (Hong, Eades, Lee '00)

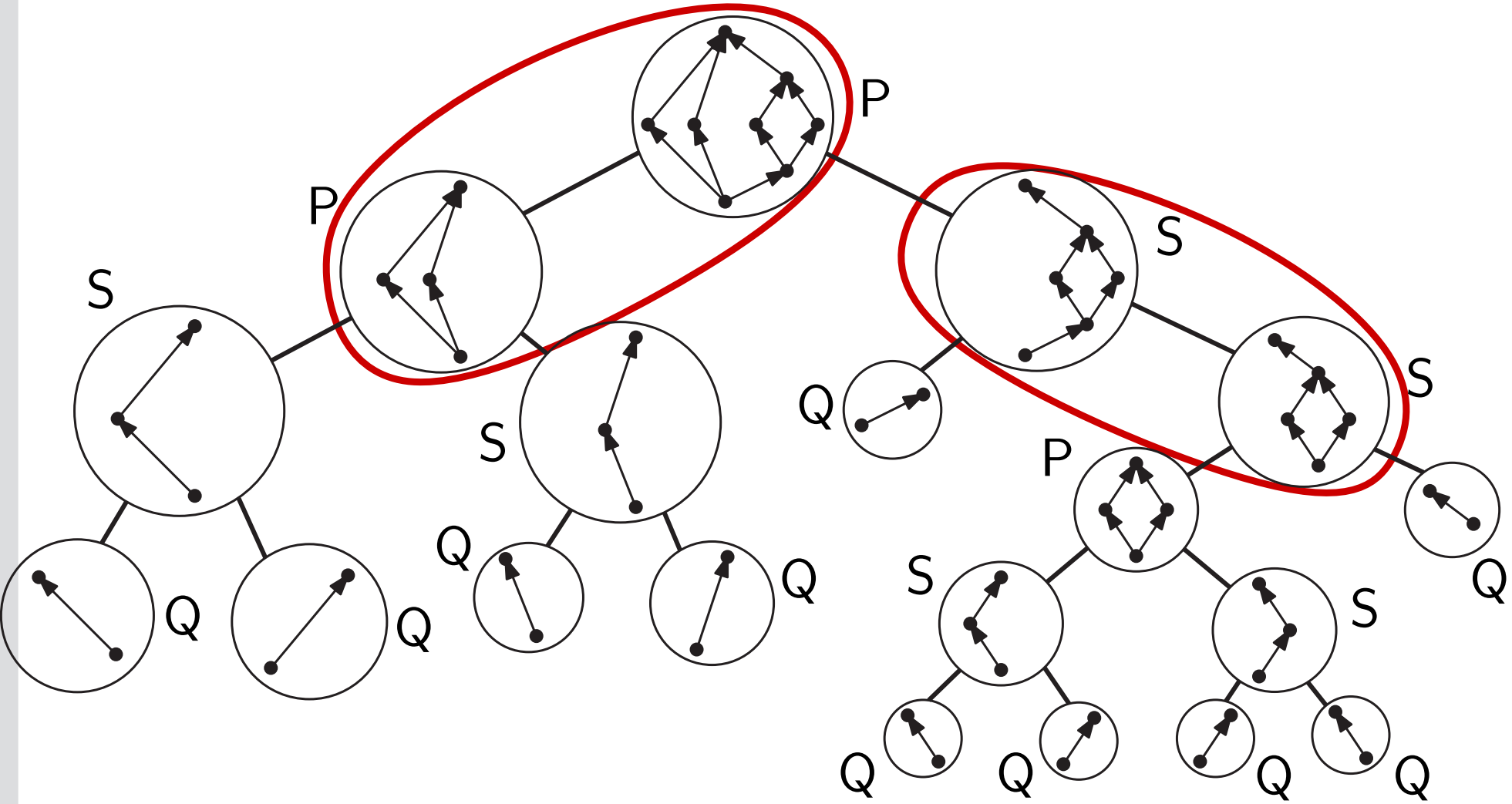
Die in einem kreuzungsfreien Aufwärtslayout eines SP-Graphen darstellbaren Symmetrien sind entweder

1. $\{\mathrm{id}\}$
2. $\{\mathrm{id}, \pi\}$ mit $\pi \in \{\pi_{\mathrm{vert}}, \pi_{\mathrm{hor}}, \pi_{\mathrm{rot}}\}$
3. $\{\mathrm{id}, \pi_{\mathrm{vert}}, \pi_{\mathrm{hor}}, \pi_{\mathrm{rot}}\}$.

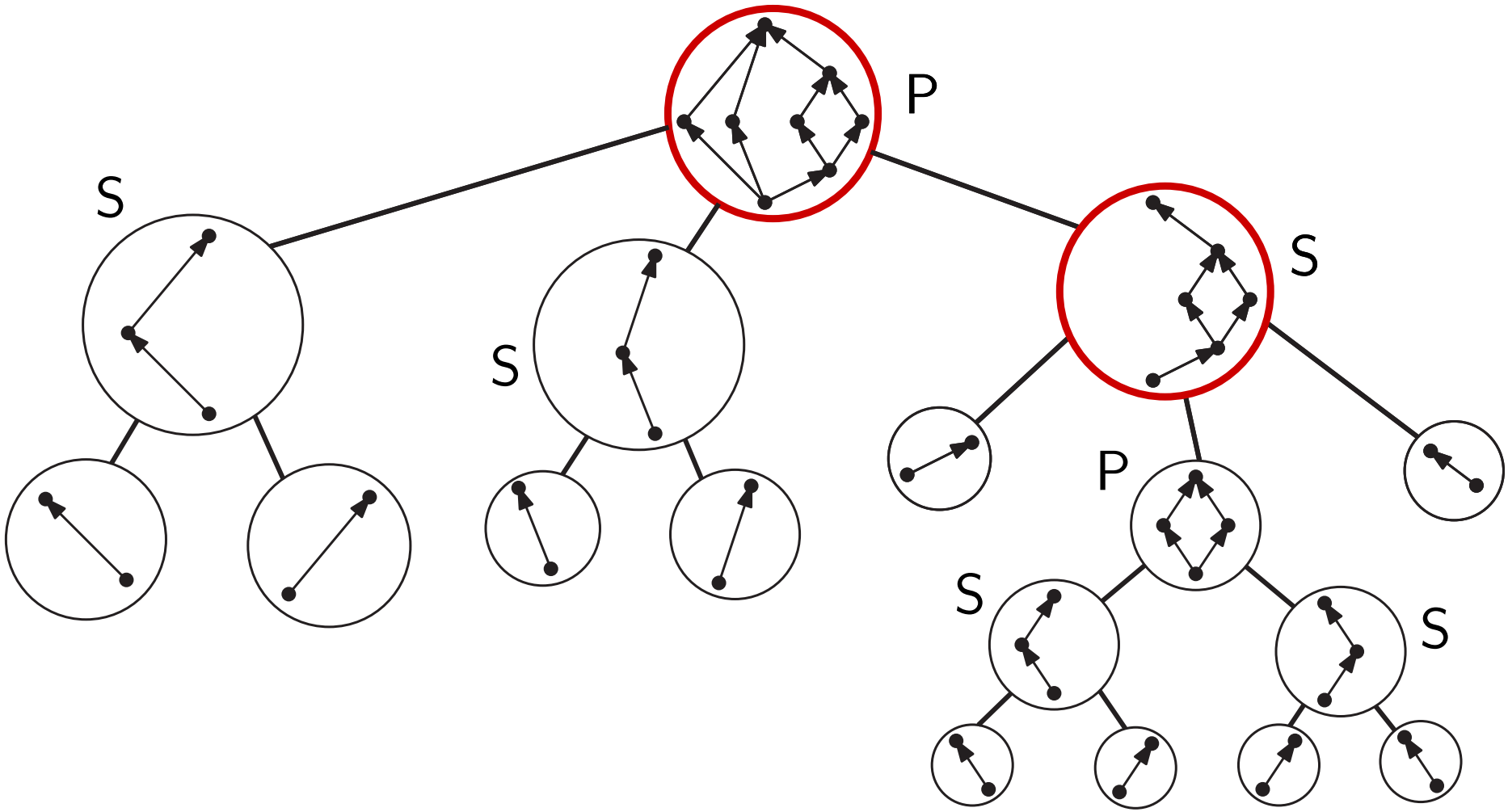
Knotenkodierung im Dekompositionsbaum



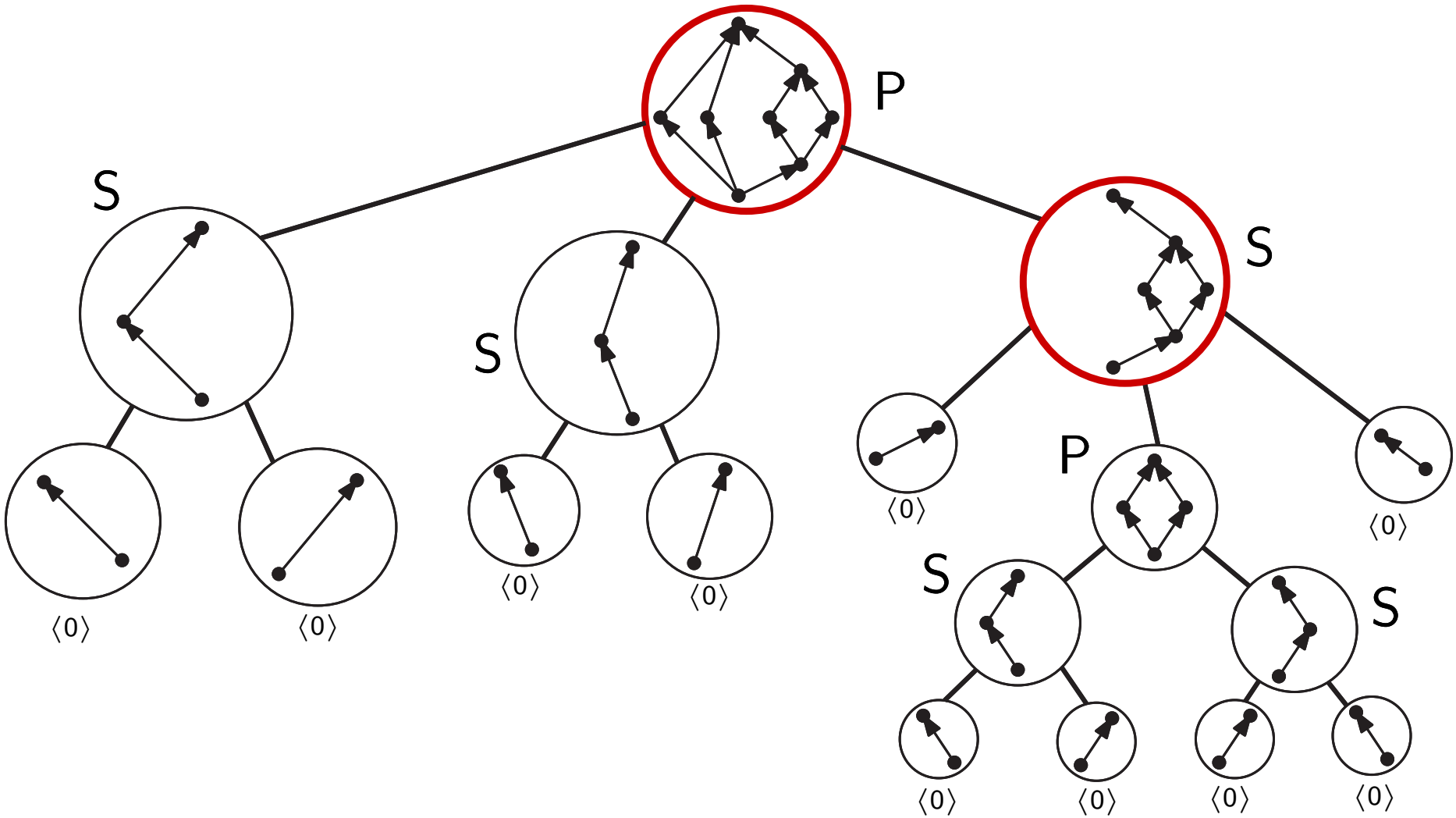
Knotenkodierung im Dekompositionsbaum



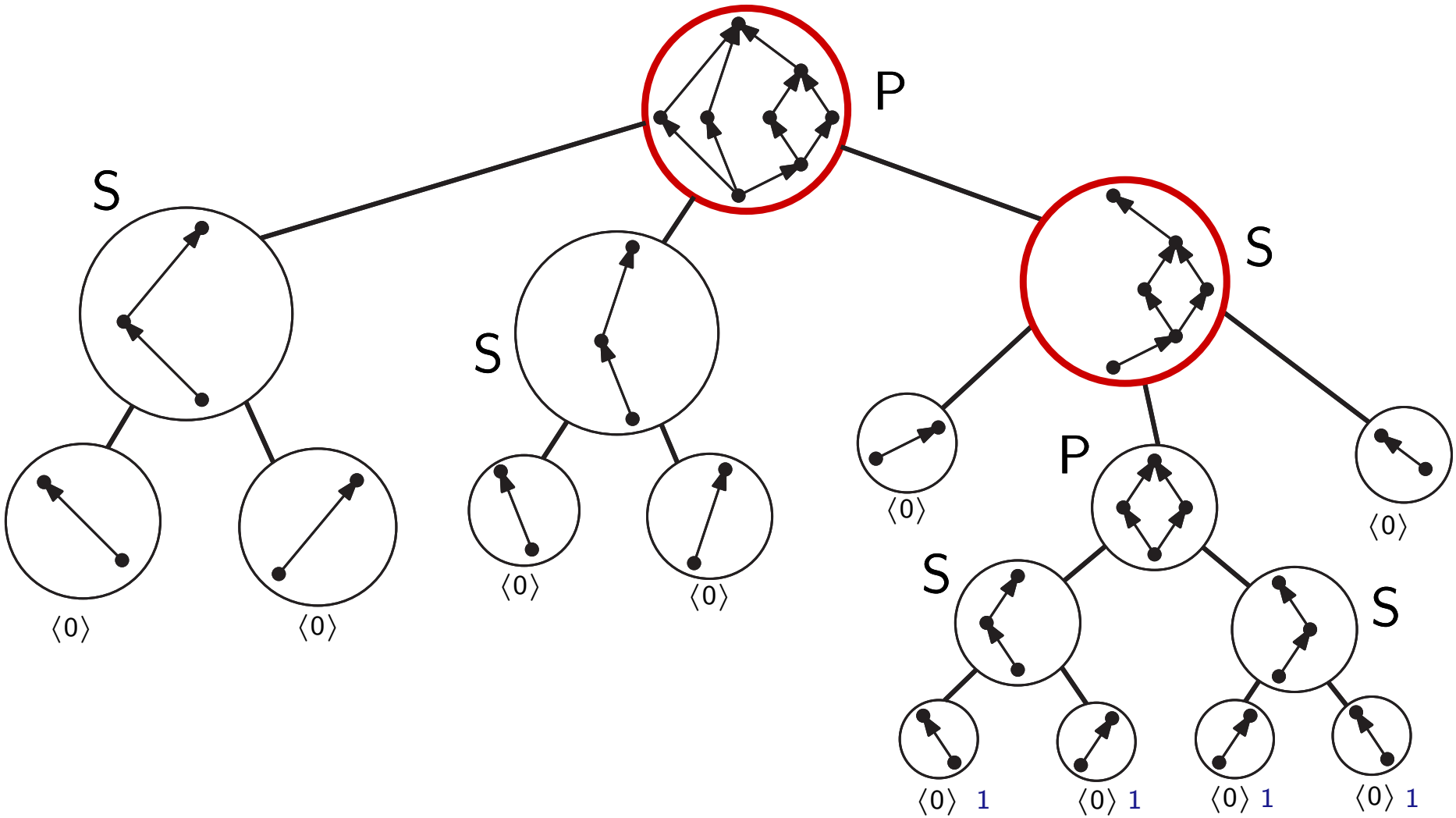
Knotenkodierung im Dekompositionsbaum



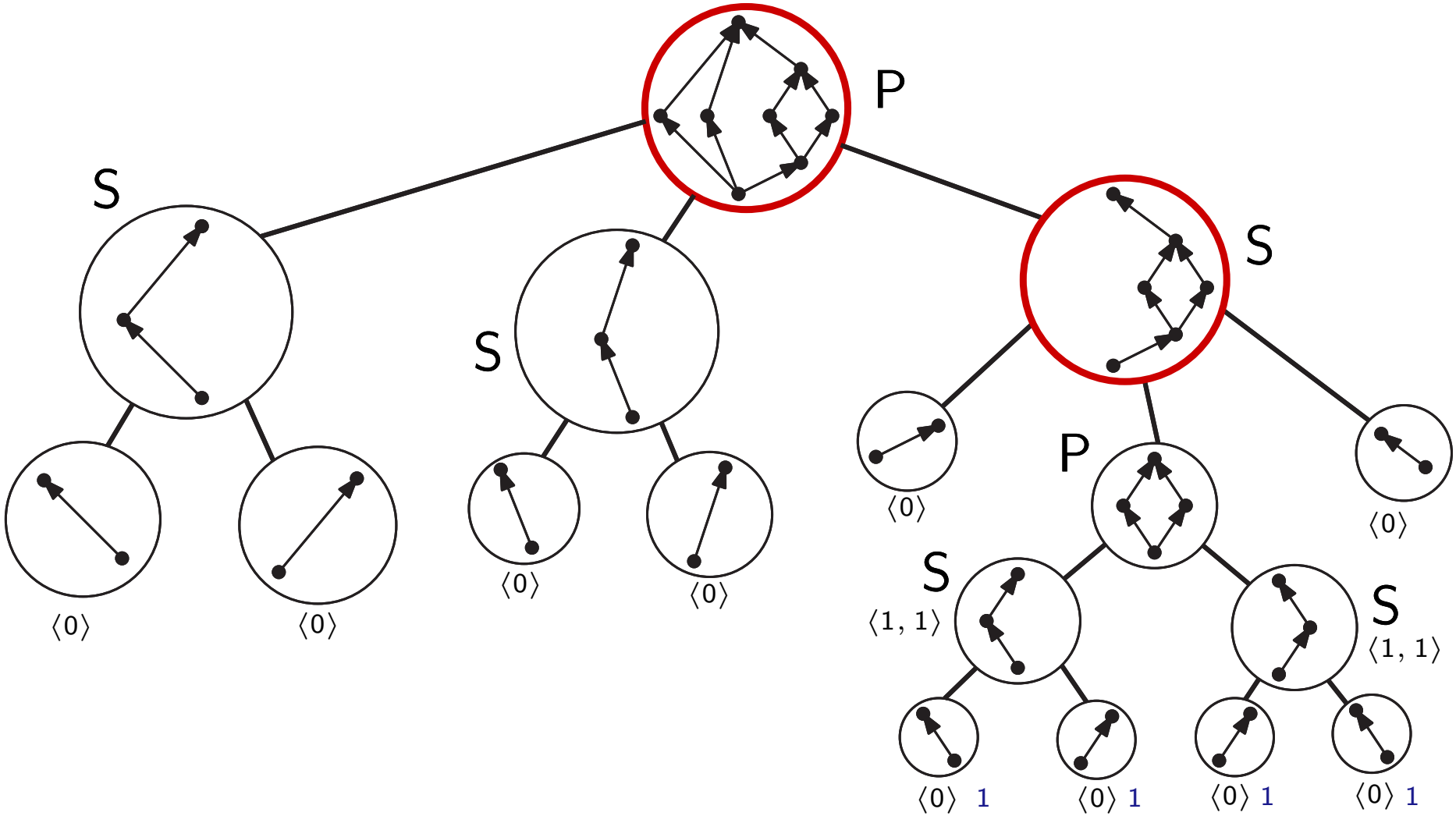
Knotenkodierung im Dekompositionsbaum



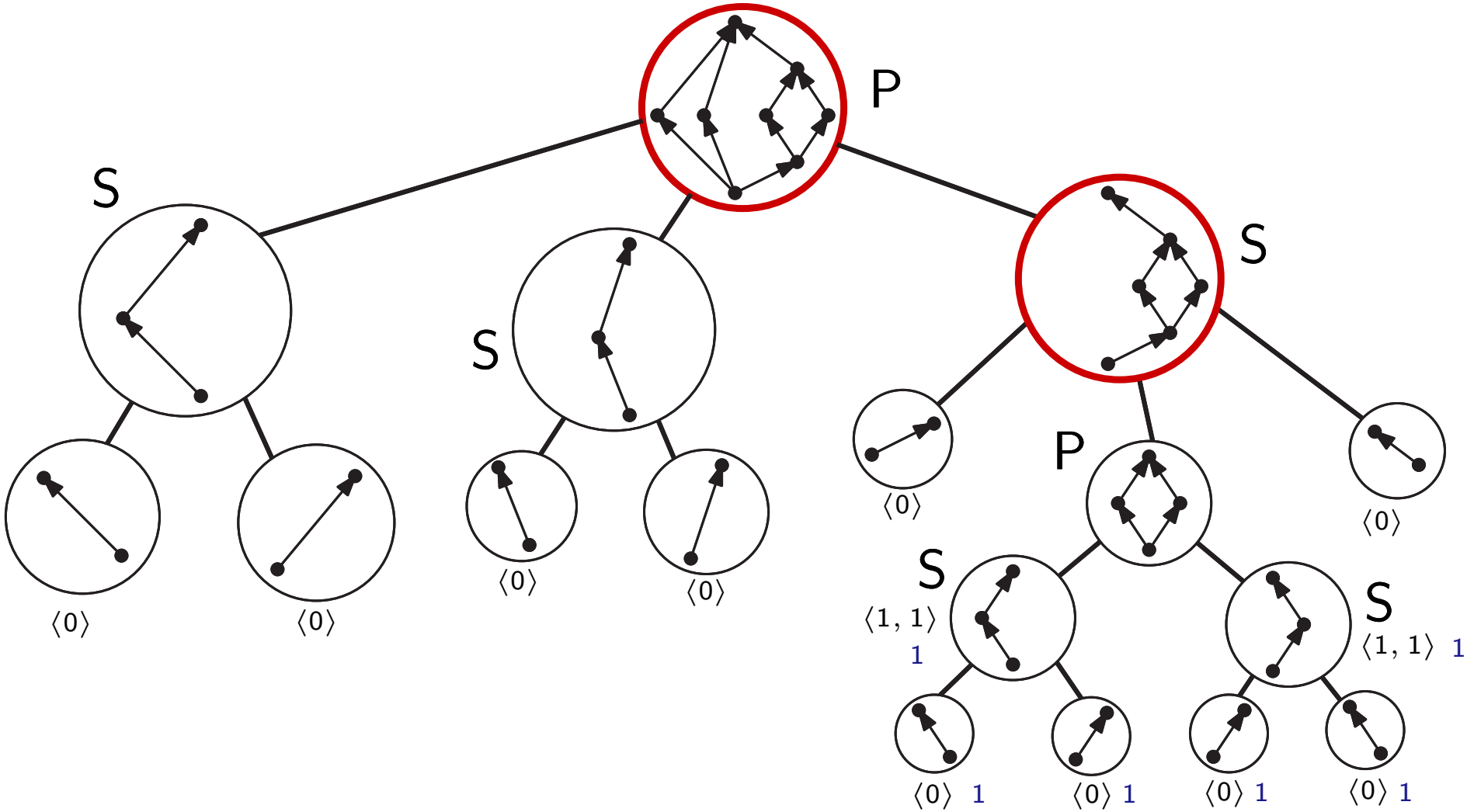
Knotenkodierung im Dekompositionsbaum



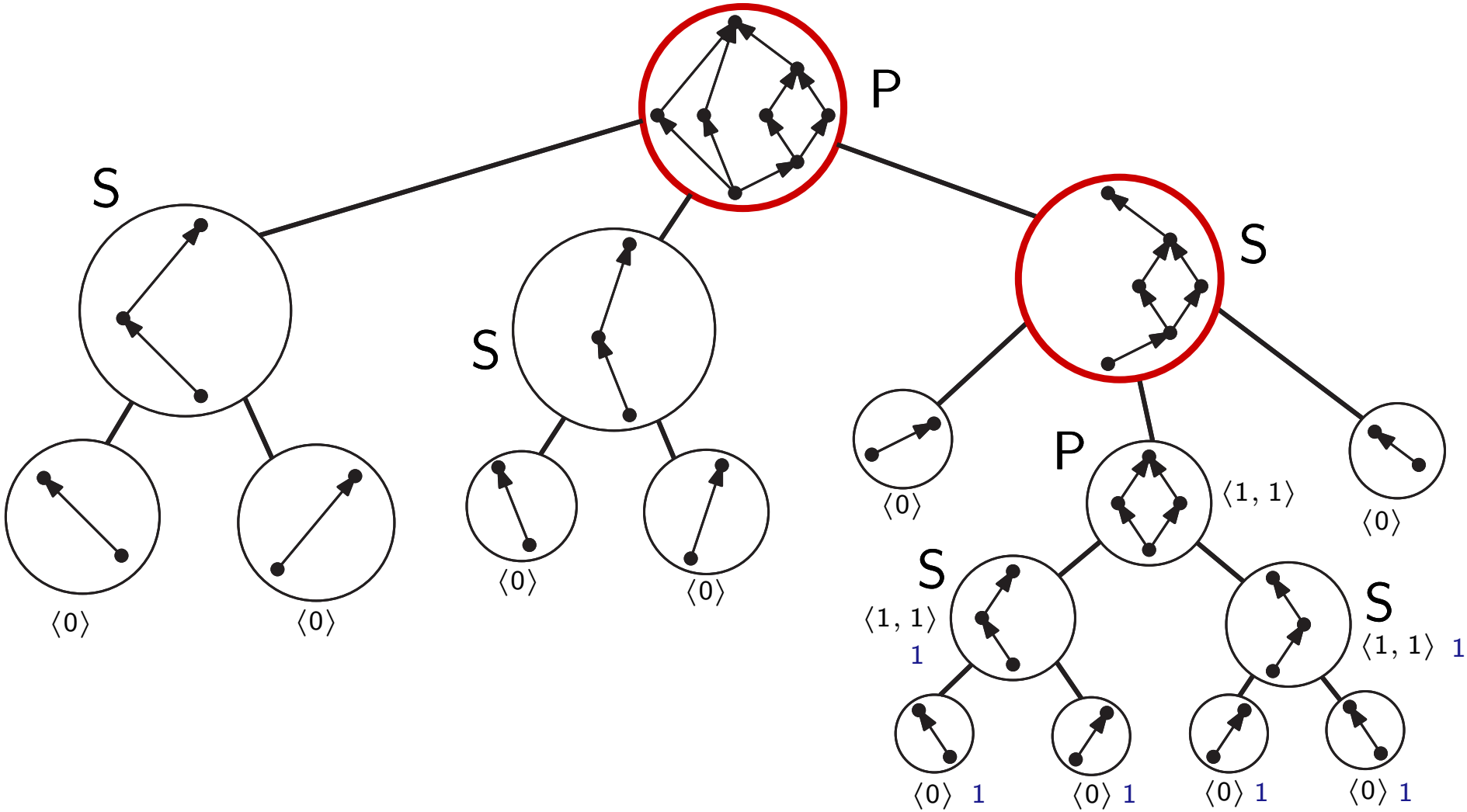
Knotenkodierung im Dekompositionsbaum



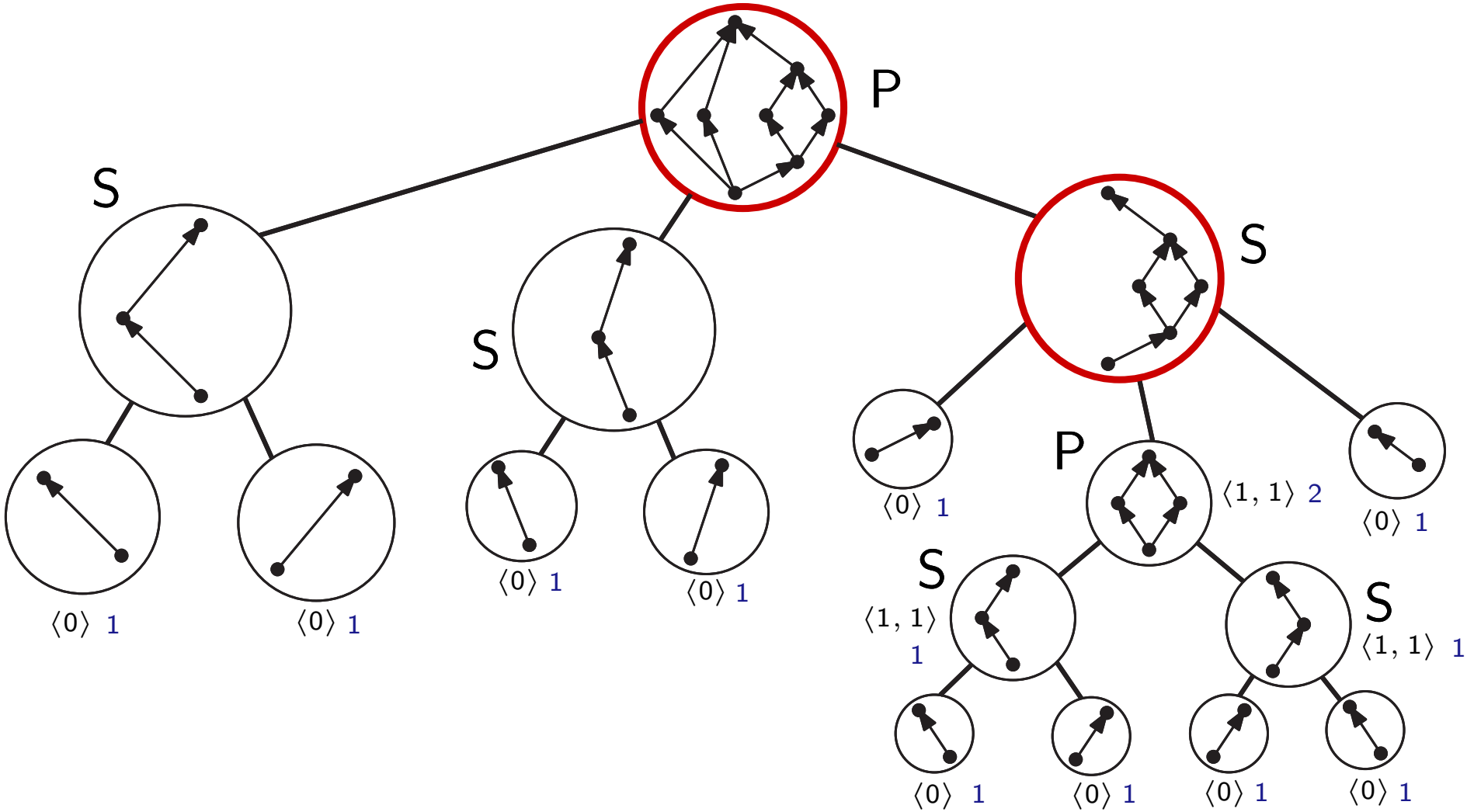
Knotenkodierung im Dekompositionsbaum



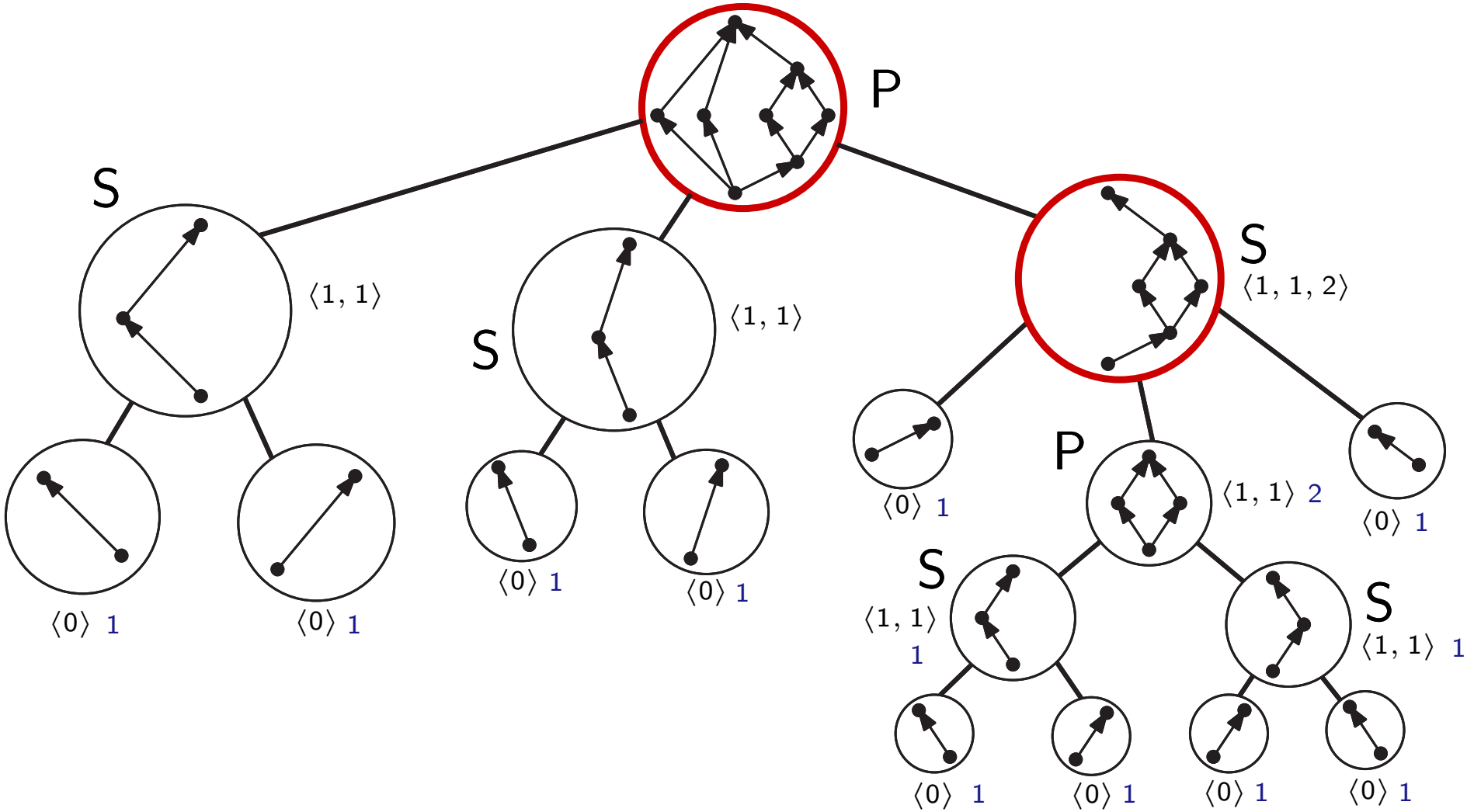
Knotenkodierung im Dekompositionsbaum



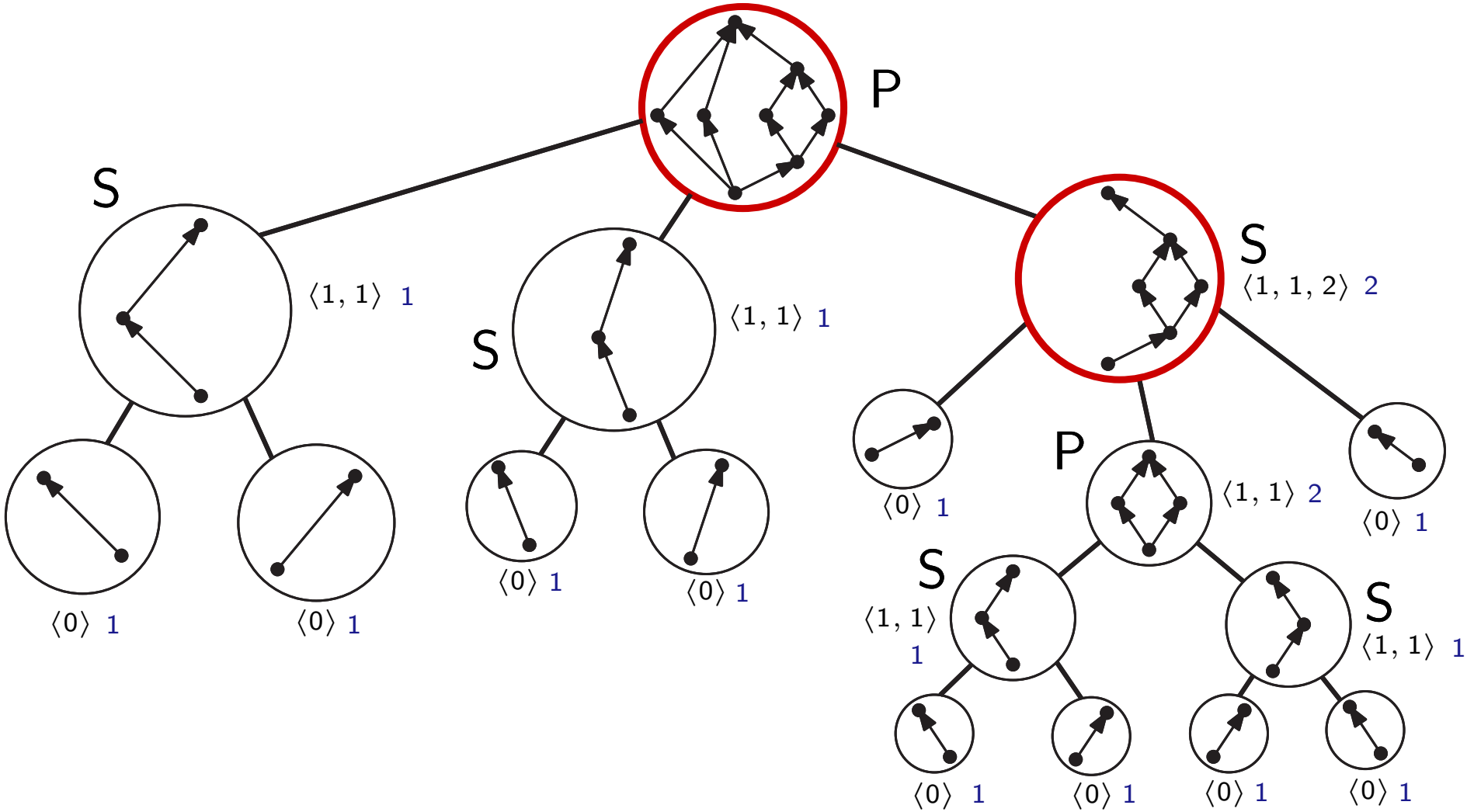
Knotenkodierung im Dekompositionsbaum



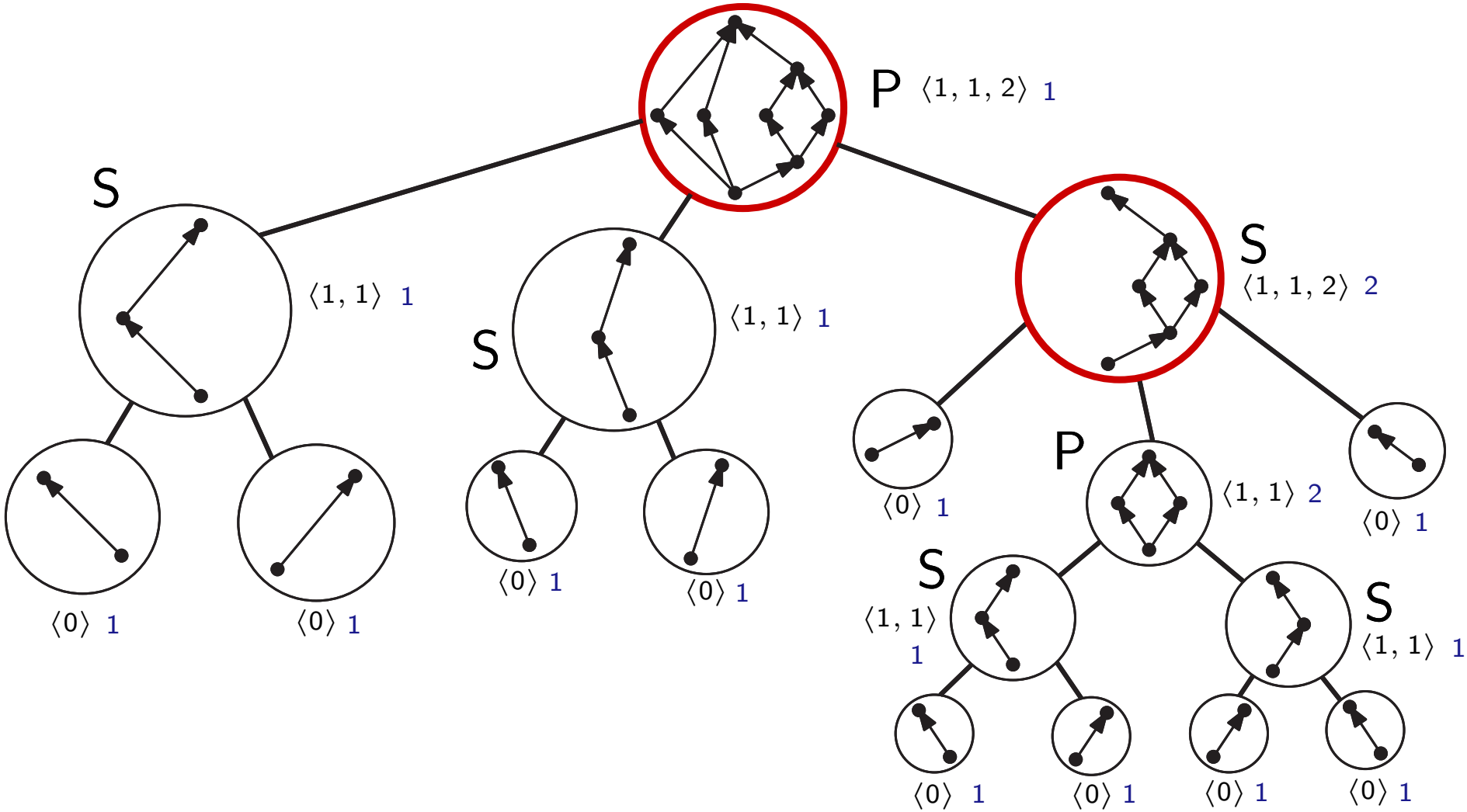
Knotenkodierung im Dekompositionsbaum



Knotenkodierung im Dekompositionsbaum



Knotenkodierung im Dekompositionsbaum



Algorithmus zur Kodierung

1. $\forall$ Q-nodes: $C(G) \leftarrow \langle 0 \rangle$
2. $\forall t = \max_G \text{depth}(G), \ldots, 0$
   (a) $\forall$ S,P-nodes G with $\text{depth}(G) = t$ and children $G_1, \ldots, G_k$: $C(G) \leftarrow \langle c(G_1), \ldots, c(G_k) \rangle$
   if P-node: sort $C(G)$ non-descending
   (b) Sort set of tuples at depth t lexicographically
   (c) $\forall G$ at depth t: $c(G) \leftarrow$ index of G in sorting

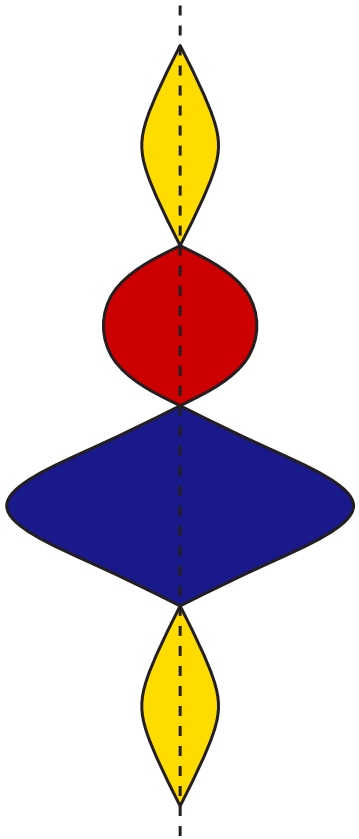
Vertikel Symmetrien Finden

Lemma (Hong et al. '98)

Gegeben kanonischer Dekompositionsbaum + Kodierung, und Knoten/Subgraph G mit Kindern $G_1, \ldots, G_k$:

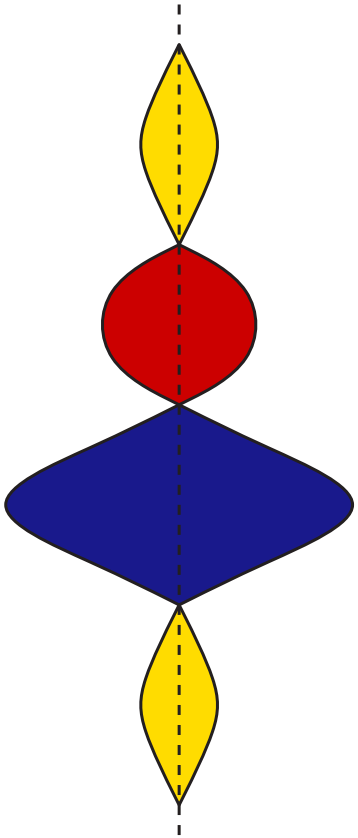
1. S-Knoten: G vertikal symmetrisch $\Leftrightarrow$ alle G_i vertikal symmetrisch
2. P-Knoten: Betrachte Klassen $\mathcal{G}_j = \{G_i : c(G_i) = j\}$
   (a) Alle $|\mathcal{G}_j|$ gerade $\Rightarrow$ G vert. symm.
   (b) Einzig $\mathcal{G}_{\hat{j}}$ ungerade $\Rightarrow$ G vert. symm. gdw. $G_{\hat{j}} \in \mathcal{G}_{\hat{j}}$ vert. symm.
   (c) Mehrere $|\mathcal{G}_j|$ ungerade $\Rightarrow$ G nicht vert. symm.

Beweisidee vertikale Symmetrie

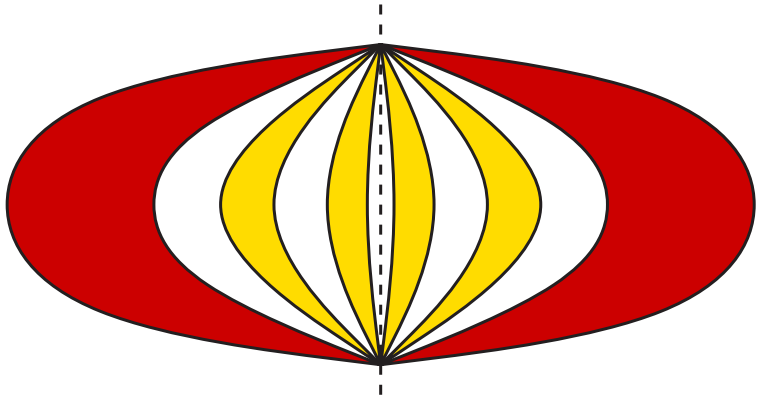


S-Knoten

Beweisidee vertikale Symmetrie

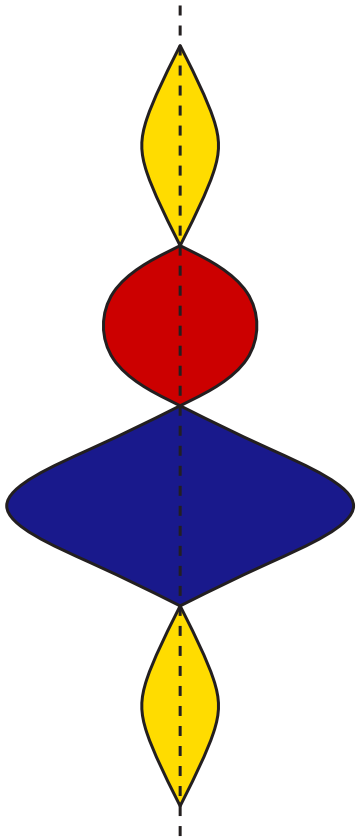


S-Knoten

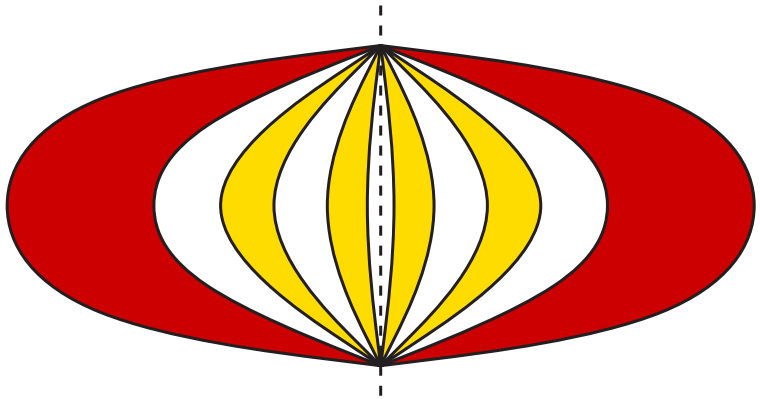


P-Knoten, alle Klassen gerade Anzahl

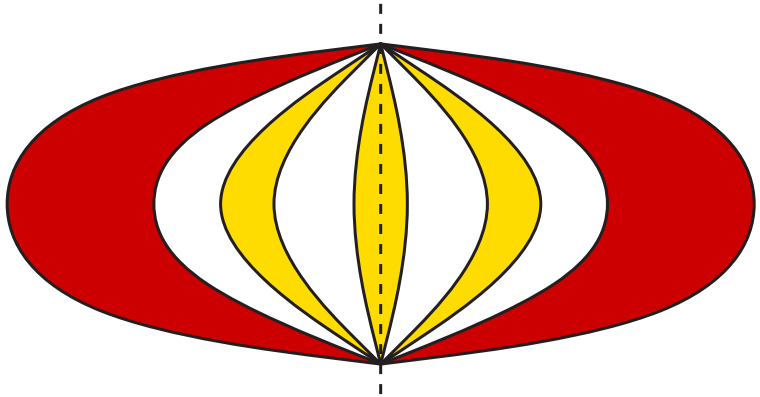
Beweisidee vertikale Symmetrie



S-Knoten
alle Knoten v-symm.



P-Knoten, alle Klassen gerade Anzahl



P-Knoten, eine Klasse ungerade Anzahl & v-symm.

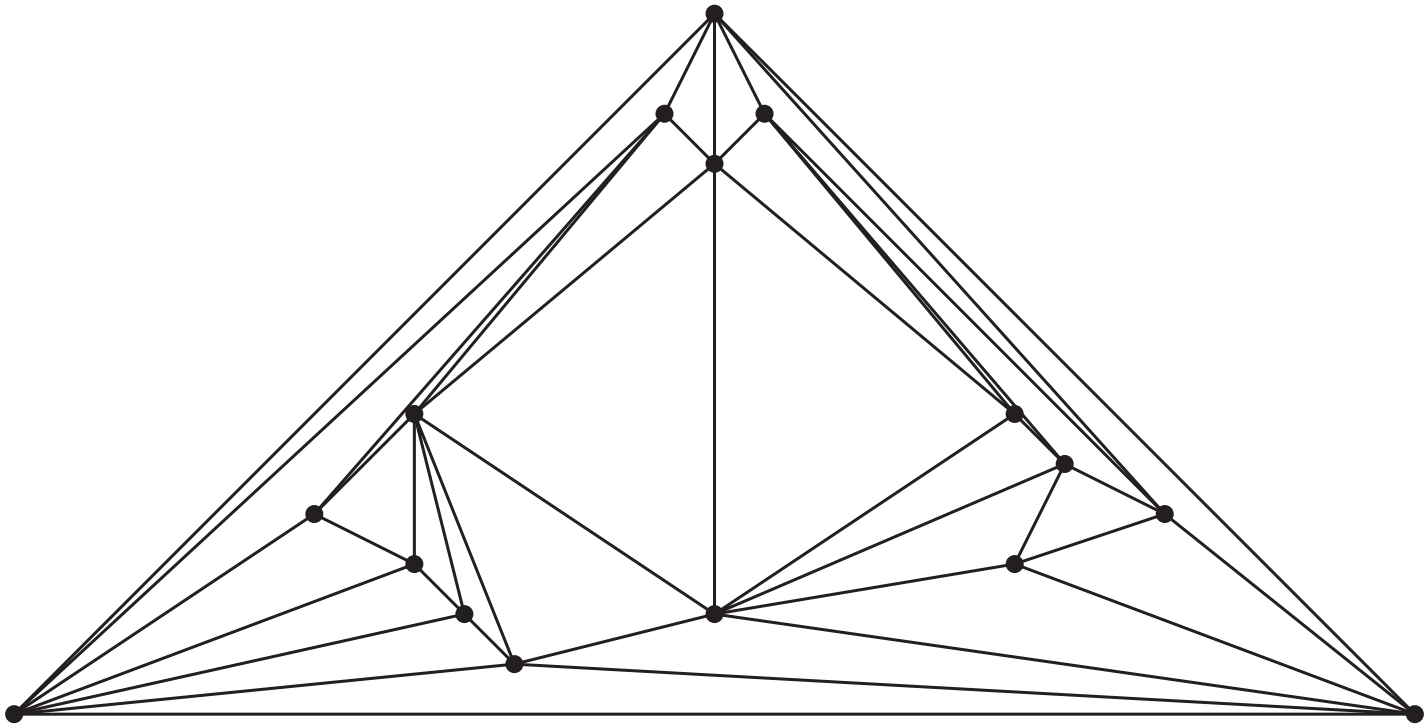
Elementare Beobachtung zur Kodierung

Gleiche Kodierung $\wedge$ Tiefe $\wedge$ Typ $\Leftrightarrow$ isomorph

Idee: Induktion über Tiefe, aufsteigend

Anfang: Ein paar Q-Knoten $\surd$
Annahme: Aussage gilt für alle Tiefen $\geq t+1$
Schritt: 2 Knoten G, H auf Tiefe t gleich kodiert
$\Leftrightarrow$ Nach Konstruktion: Folge der Klassen, i.e., der Kodierungen der Kinder G_i (auf $t+1$) gleich
$\Leftrightarrow_{\text{I.A.}}$ G, H isomorph

Inkrementelles Gitterlayout für planare Graphen



Kanonische Knotenordnung

Idee: Bestimme geeignete Knotenordnung und zeichne dann inkrementell [de Fraysseix, Pach, Pollack '90]

Kanonische Knotenordnung

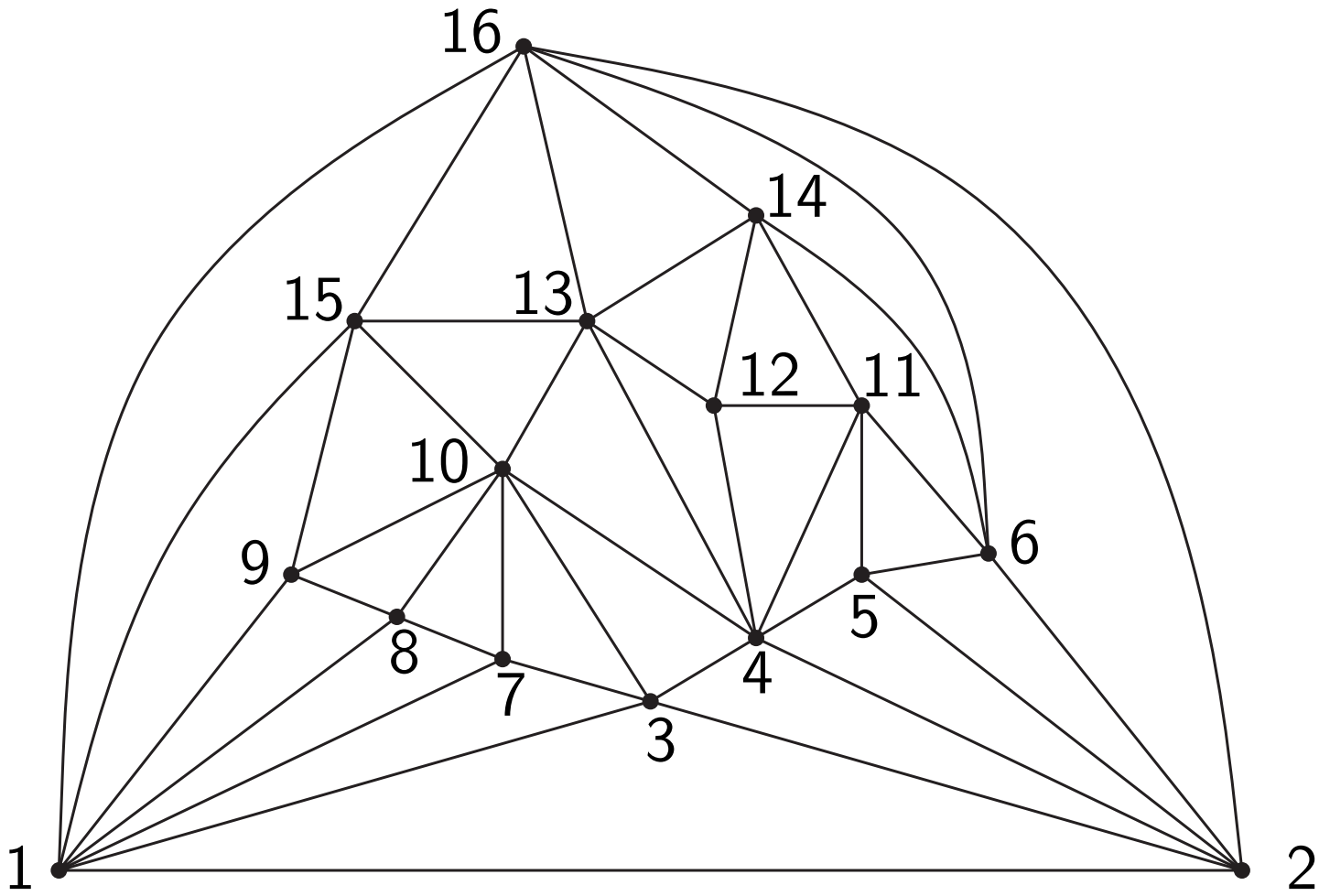
Idee: Bestimme geeignete Knotenordnung und zeichne dann inkrementell [de Fraysseix, Pach, Pollack '90]

Definition: Kanonische Knotenordnung

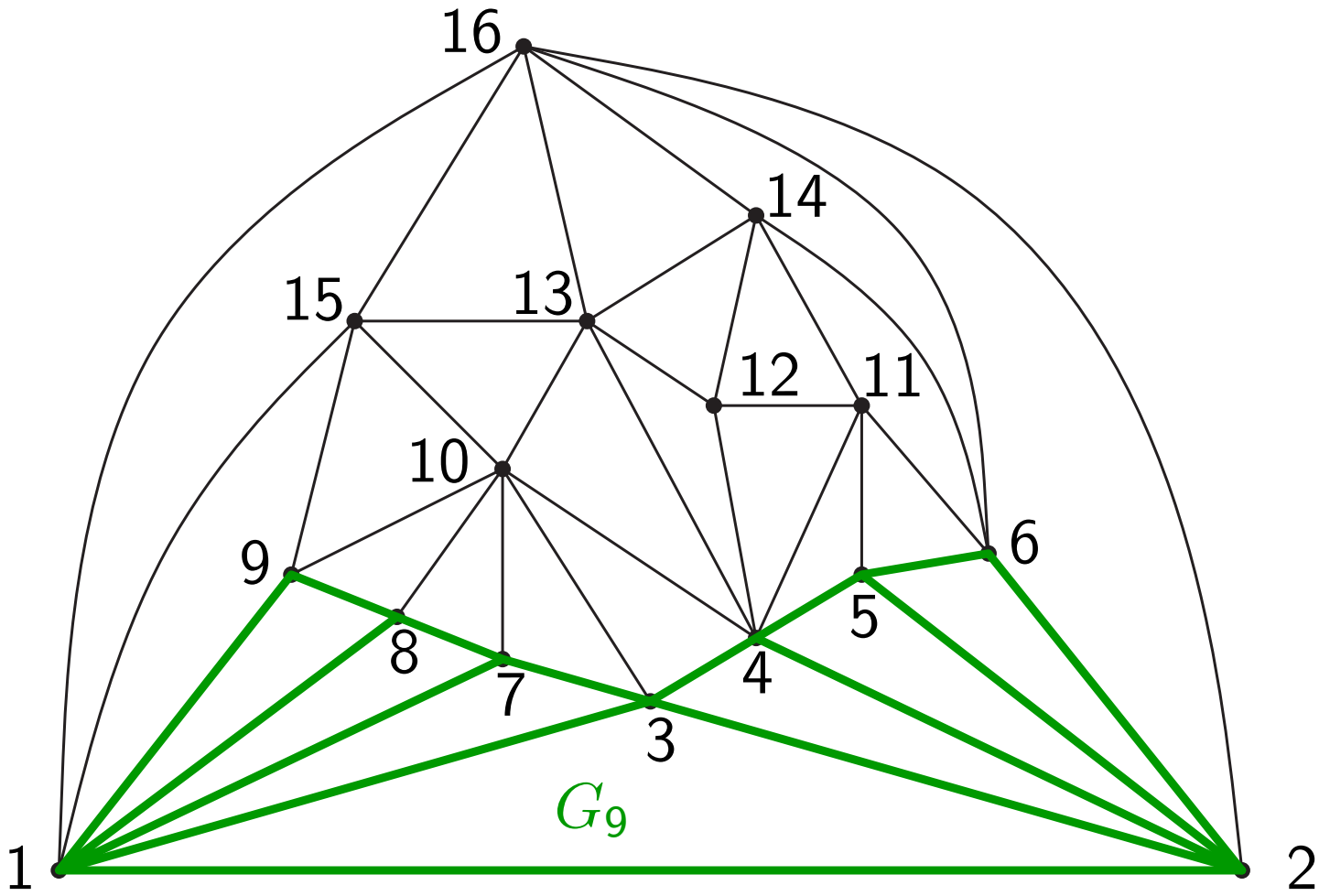
Sei $G = (V, E)$ ein triangulierter, planar eingebetteter Graph mit $n \geq 3$ Knoten. Eine Knotenordnung $\pi = (v_1, v_2, \ldots, v_n)$ heißt *kanonische Ordnung*, falls gilt

1. der von $\{v_1, \ldots v_k\}$ induzierte Teilgraph G_k ist 2-zusammenhängend und intern trianguliert
2. (v_1, v_2) ist Außenkante von G_k
3. für $k < n$ liegt v_{k+1} in der äußeren Facette von G_k und alle Nachbarn von v_{k+1} in G_k bilden ein Intervall auf dem Rand $C_o(G_k)$ der äußeren Facette von G_k

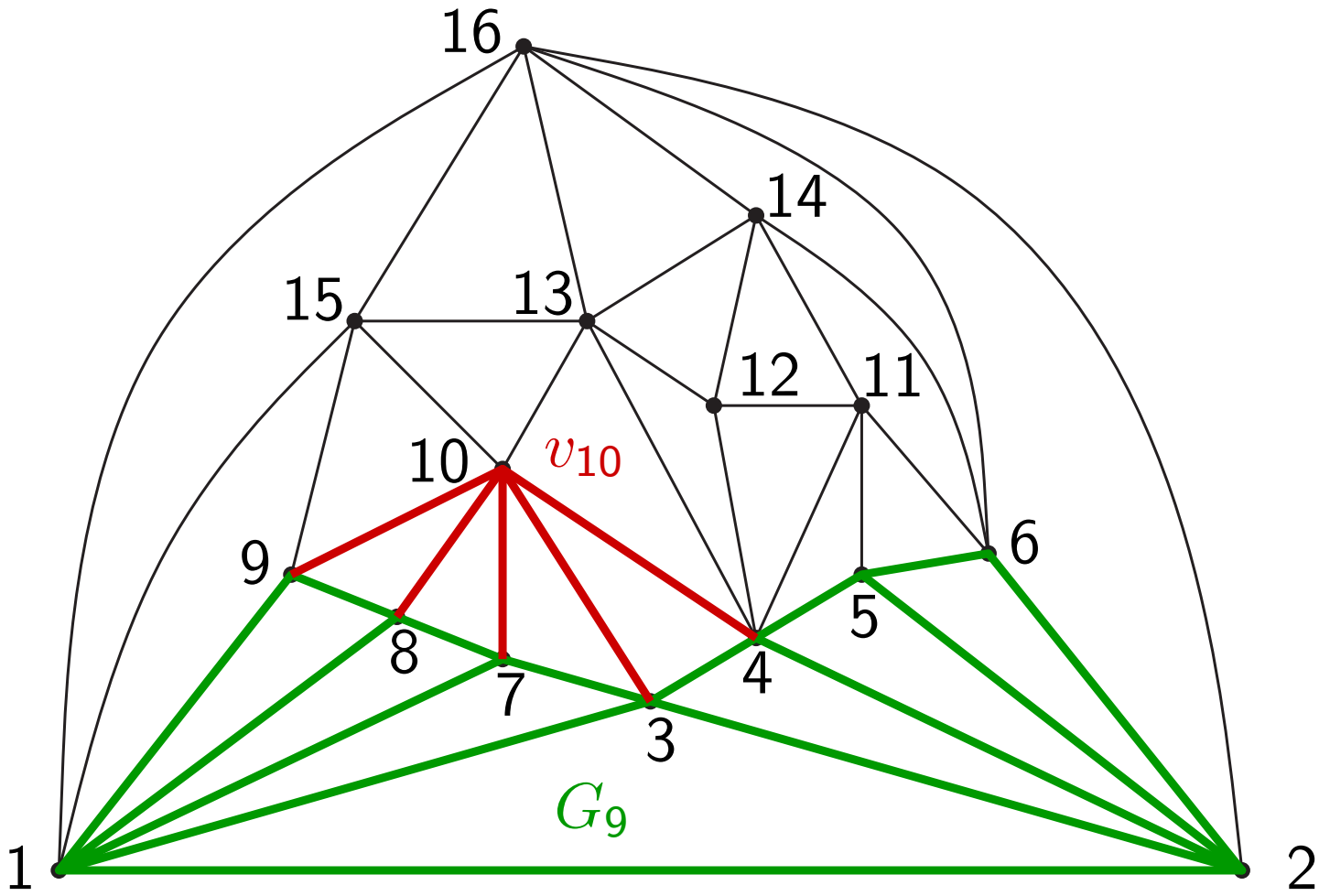
Beispiel kanonische Ordnung



Beispiel kanonische Ordnung



Beispiel kanonische Ordnung



Existenz einer kanonischen Knotenordnung

OBdA sei G trianguliert
(sonst tu's, benutze's und mach's wieder rückgängig)

Existenz einer kanonischen Knotenordnung

OBdA sei G trianguliert
(sonst tu's, benutze's und mach's wieder rückgängig)

Lemma

Jeder triangulierte, planar eingebettete Graph besitzt eine kanonische Ordnung.

Existenz einer kanonischen Knotenordnung

OBdA sei G trianguliert
(sonst tu's, benutze's und mach's wieder rückgängig)

Lemma

Jeder triangulierte, planar eingebettete Graph besitzt eine kanonische Ordnung.

Beweisidee: Für $n \leq 3$ klar, für $n \geq 4$ rückwärts.
Für $k = n$ gelten Bedingungen da $G_n = G$
Sei $k < n$ und Bed. gelten für G_{k+i}

Existenz einer kanonischen Knotenordnung

OBdA sei G trianguliert
(sonst tu's, benutze's und mach's wieder rückgängig)

Lemma

Jeder triangulierte, planar eingebettete Graph besitzt eine kanonische Ordnung.

Beweisidee: Für $n \leq 3$ klar, für $n \geq 4$ rückwärts.
Für $k = n$ gelten Bedingungen da $G_n = G$
Sei $k < n$ und Bed. gelten für G_{k+i}

Finde $w \neq v_1, v_2$ der nicht Endp. einer *Sehne* von $C_0(G_k)$ ist:
$C_0(G_k)$ hat keine Sehne $\Rightarrow \surd$
suche minimale Sehne $\Rightarrow$ wähle eingeschlossenes $w\surd$

Algorithmus kanonische Ordnung

Algorithm 1: Canonical-Ordering

1 **forall** $v \in V$ **do**
2 $\quad$ chords$(v) \leftarrow 0$; out$(v) \leftarrow$ false; mark$(v) \leftarrow$ false;
3 out(v_1), out(v_2), out$(v_n) \leftarrow$ true;
4 **for** $k = n$ **to** 3 **do**
5 $\quad$ wähle $v \neq v_1, v_2$ mit mark$(v) =$ false, out$(v) =$ true, chords$(v) = 0$;
6 $\quad$ $v_k \leftarrow v$; mark$(v) \leftarrow$ true;
7 $\quad$ $(w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2) \leftarrow C_o(G_{k-1})$;
8 $\quad$ $(w_p, \ldots, w_q) \leftarrow$ unmarkierte Nachbarn von v_k;
9 $\quad$ out$(w_i) \leftarrow$ true for all $p < i < q$;
10 $\quad$ aktualisiere chords$(\cdot)$ für diese w_i und ihre Nachbarn;

Details zum Algorithmus

Zeile 5: Knoten v existiert wegen Lemma immer

Details zum Algorithmus

Zeile 5: Knoten v existiert wegen Lemma immer

Laufzeit: Halte Liste aller unmarkierten Knoten vor

Details zum Algorithmus

Zeile 5: Knoten v existiert wegen Lemma immer

Laufzeit: Halte Liste aller unmarkierten Knoten vor

Zeile 10:

falls $q = p + 1$ (Sehne!) verringere chords(p), chords(q) um 1
sonst betrachte alle w_i $(p < i < q)$, und w_i's Nachbarn z:

Details zum Algorithmus

Zeile 5: Knoten v existiert wegen Lemma immer

Laufzeit: Halte Liste aller unmarkierten Knoten vor

Zeile 10:

falls $q = p + 1$ (Sehne!) verringere chords(p), chords(q) um 1
sonst betrachte alle w_i $(p < i < q)$, und w_i's Nachbarn z:

für z mit out(z) = true und $z \neq w_{i-1}, w_{i+1}$ ist $\{z, w_i\}$ Sehne
$\Rightarrow$ erhöhe chords(w_i)
ist zudem $z \notin \{w_{p+1}, \ldots, w_{q-1}\} \Rightarrow$ erhöhe chords(z)

Details zum Algorithmus

Zeile 5: Knoten v existiert wegen Lemma immer

Laufzeit: Halte Liste aller unmarkierten Knoten vor

Zeile 10:

falls $q = p + 1$ (Sehne!) verringere chords(p), chords(q) um 1
sonst betrachte alle w_i $(p < i < q)$, und w_i's Nachbarn z:

für z mit out(z) = true und $z \neq w_{i-1}, w_{i+1}$ ist $\{z, w_i\}$ Sehne
$\Rightarrow$ erhöhe chords(w_i)
ist zudem $z \notin \{w_{p+1}, \ldots, w_{q-1}\} \Rightarrow$ erhöhe chords(z)

Laufzeit: Benötigt $O(\deg(w_i))$ Zeit, nur einmal pro Knoten
$\Rightarrow$ insgesamt $O(n)$

Zeichenalgorithmus I

Ein: tria., planar eingeb. G, kanon. KO, $f_0 = \Delta(v_1, v_2, v_n)$
Aus: Geradlinige, planare Zeichnung auf Gitterpunkten

Zeichenalgorithmus I

Ein: tria., planar eingeb. G, kanon. KO, $f_0 = \Delta(v_1, v_2, v_n)$
Aus: Geradlinige, planare Zeichnung auf Gitterpunkten
Idee: Inkrementell, horizontales "Platzmachen"

Zeichenalgorithmus I

Ein: tria., planar eingeb. G, kanon. KO, $f_0 = \Delta(v_1, v_2, v_n)$

Aus: Geradlinige, planare Zeichnung auf Gitterpunkten

Idee: Inkrementell, horizontales "Platzmachen"

Invarianten: (i) $v_1 = (0, 0)$, $v_2 = (2k - 4, 0)$

(ii) Im UZS sei $C_0(G_k) = (v_1 = w_1, \ldots, w_t = v_2)$
es gilt $x(w_1) <, \ldots, < x(w_t)$

(iii) $e \in E \setminus \{v_1, v_2\}$: Steigung$(e) = \pm 1$

Zeichenalgorithmus I

Ein: tria., planar eingeb. G, kanon. KO, $f_0 = \Delta(v_1, v_2, v_n)$

Aus: Geradlinige, planare Zeichnung auf Gitterpunkten

Idee: Inkrementell, horizontales "Platzmachen"

Invarianten:
(i) $v_1 = (0, 0)$, $v_2 = (2k - 4, 0)$
(ii) Im UZS sei $C_0(G_k) = (v_1 = w_1, \ldots, w_t = v_2)$
es gilt $x(w_1) <, \ldots, < x(w_t)$
(iii) $e \in E \setminus \{v_1, v_2\}$: Steigung$(e) = \pm 1$

Bezeichnungen:
(i) $N(v_k)$ auf $C_0(G_k)$: $w_p, \ldots, w_q$
(ii) $L(v) =$ abhängige Knoten (beim Verschieben)
(iii) $\mu(a, b)$ Schnittpunkt der Geraden durch a/b mit Steigungen $1/-1$

Zeichenalgorithmus II

Setze bei Start: $v_1 = (0, 0)$, $v_2 = (2, 0)$, $v_3 = (1, 1)$,
$L(v_i) = \{v_i\}$

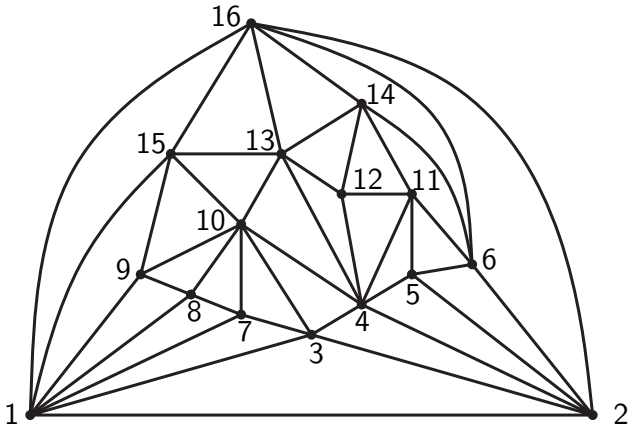
Zeichenalgorithmus II

Setze bei Start: $v_1 = (0,0)$, $v_2 = (2,0)$, $v_3 = (1,1)$,
$L(v_i) = \{v_i\}$

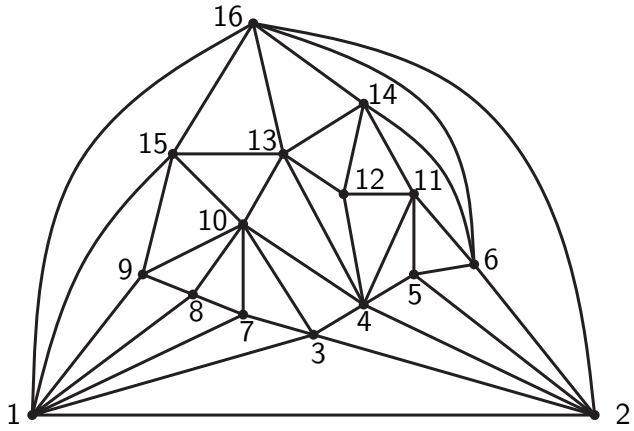
Algorithm 2: Incremental Draw

1 **for** $k = 4, \ldots, n$ **do**
2 schiebe $\bigcup_{i=p+1}^{q-1} L(w_i)$ um 1 nach rechts
3 schiebe $\bigcup_{i=q}^{t} L(w_i)$ um 2 nach rechts
4 $v_k \leftarrow \mu(w_p, w_q)$
5 $L(v_k) \leftarrow \{v_k\} \cup \bigcup_{i=p+1}^{q-1} L(w_i)$

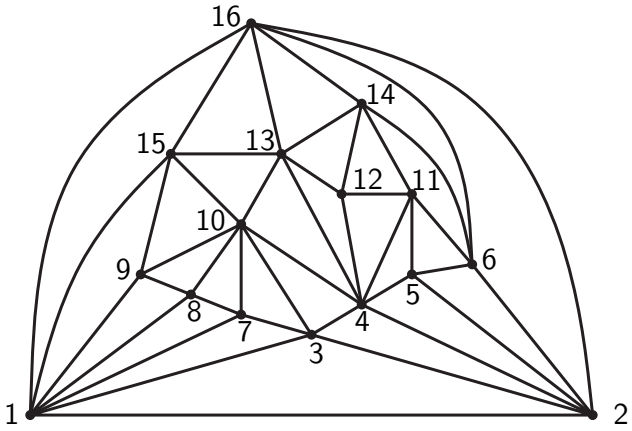
Beispiel

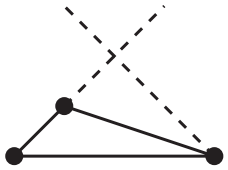


Beispiel

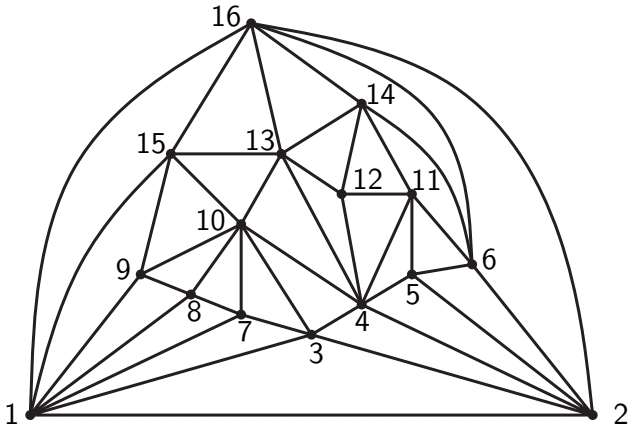


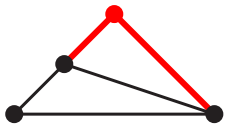
Beispiel



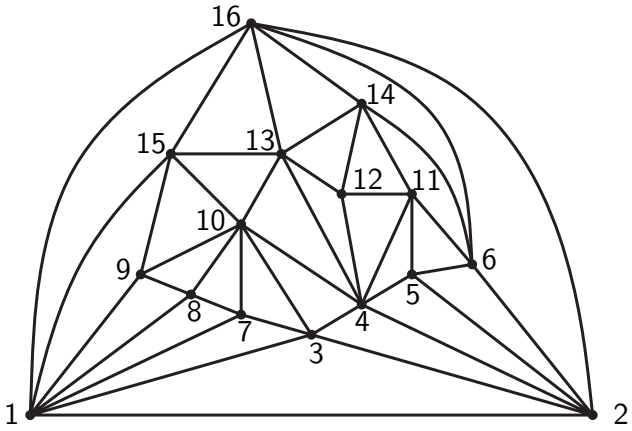


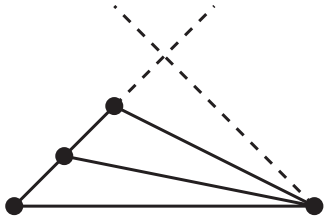
Beispiel



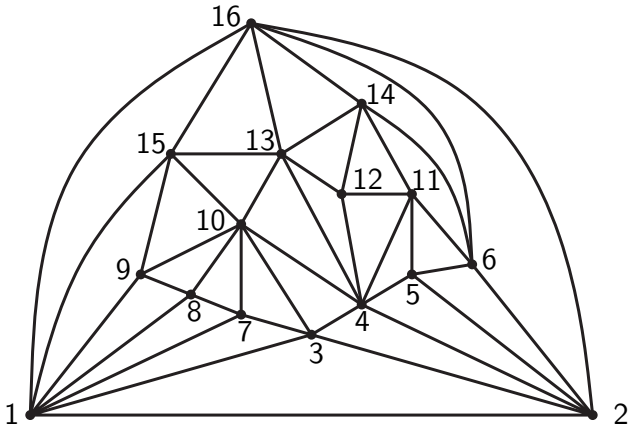


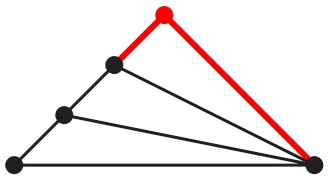
Beispiel



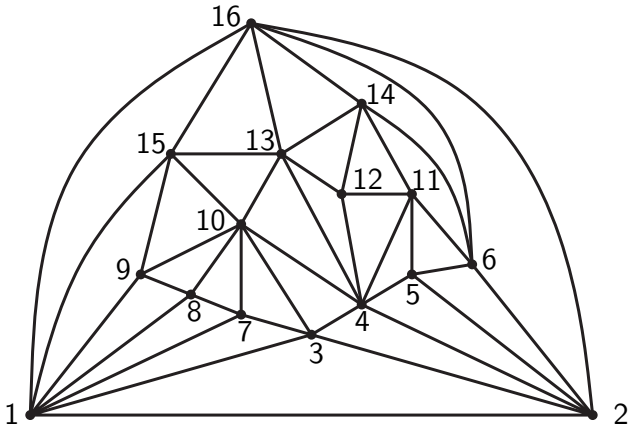


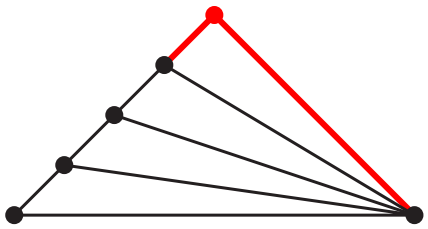
Beispiel



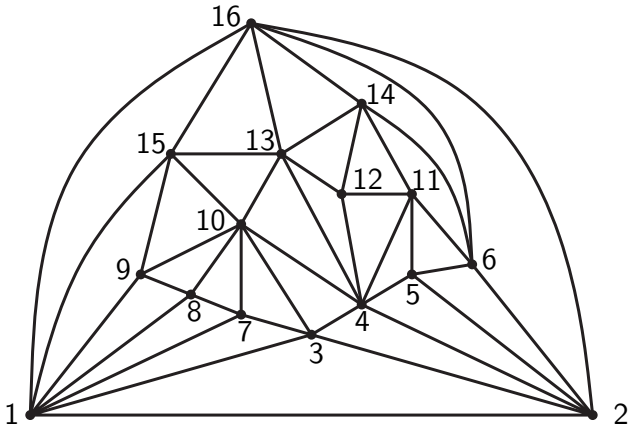


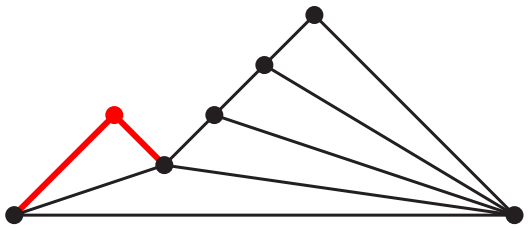
Beispiel



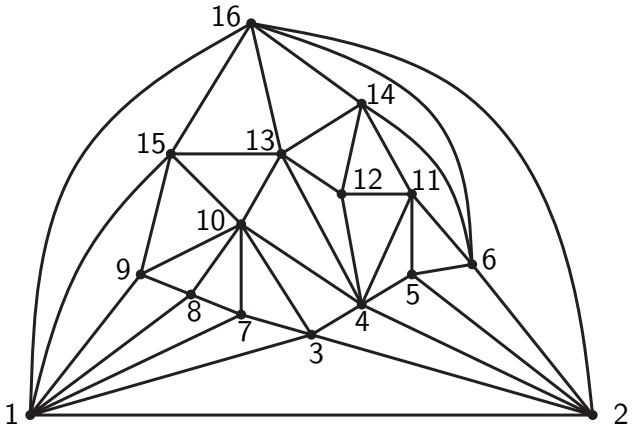


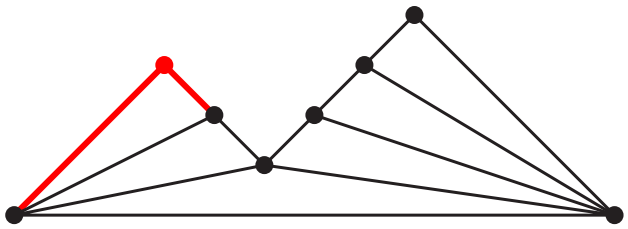
Beispiel



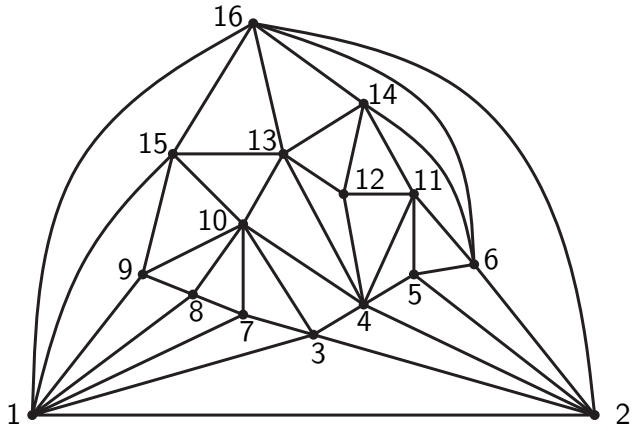


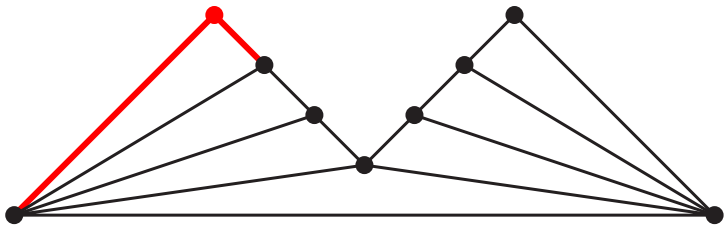
Beispiel



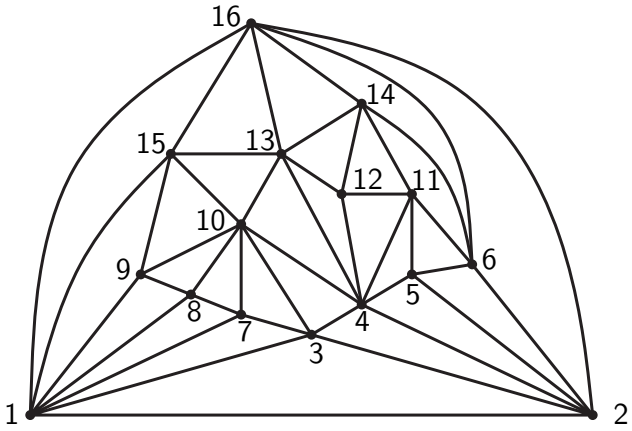


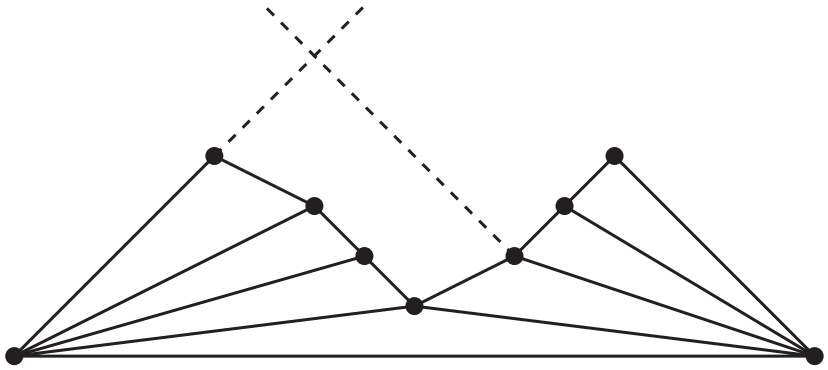
Beispiel



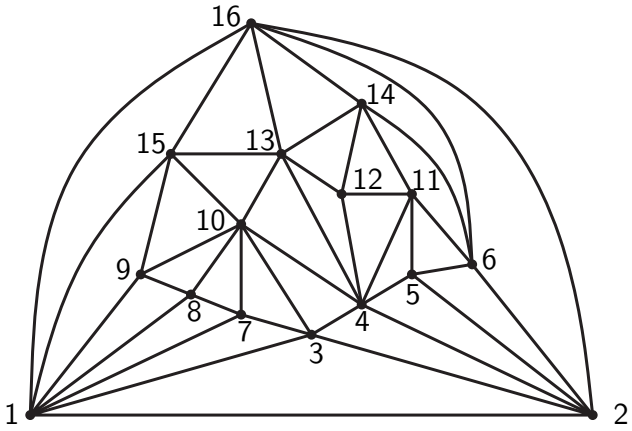


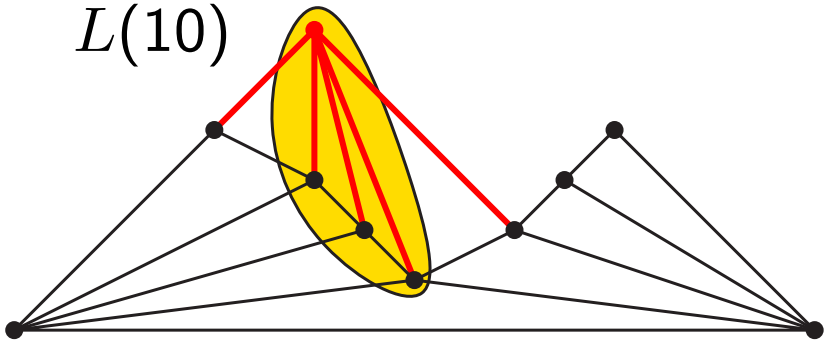
Beispiel



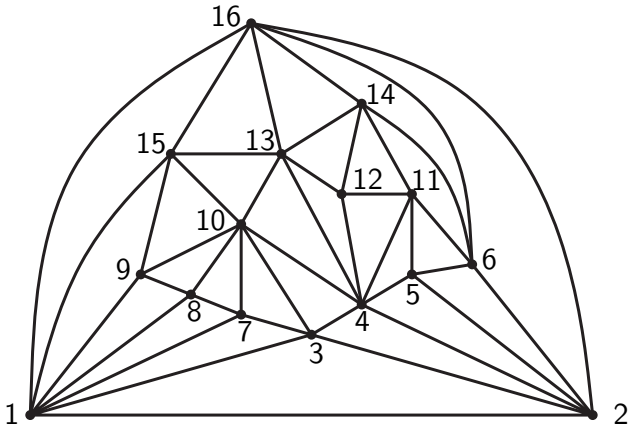


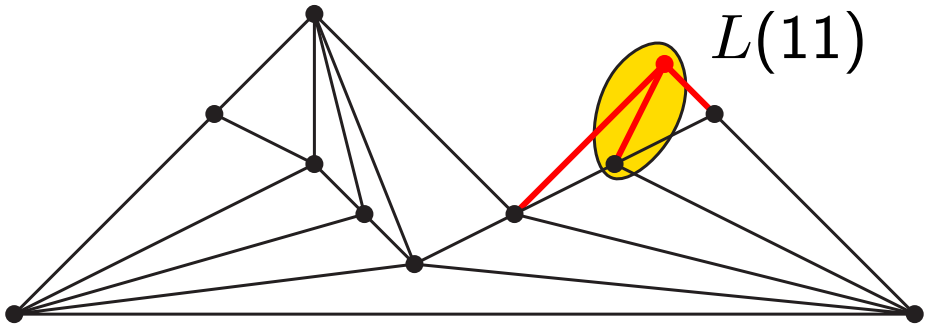
Beispiel



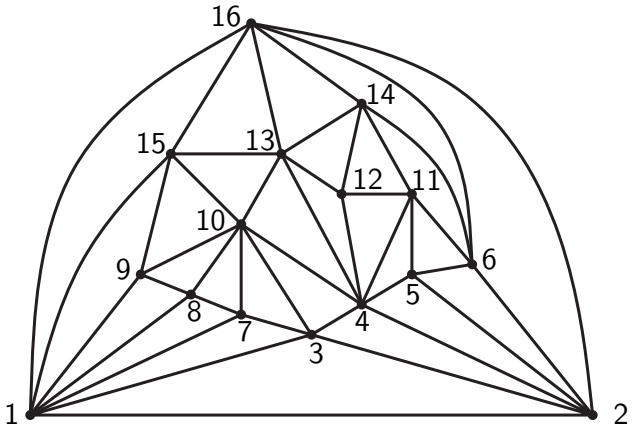


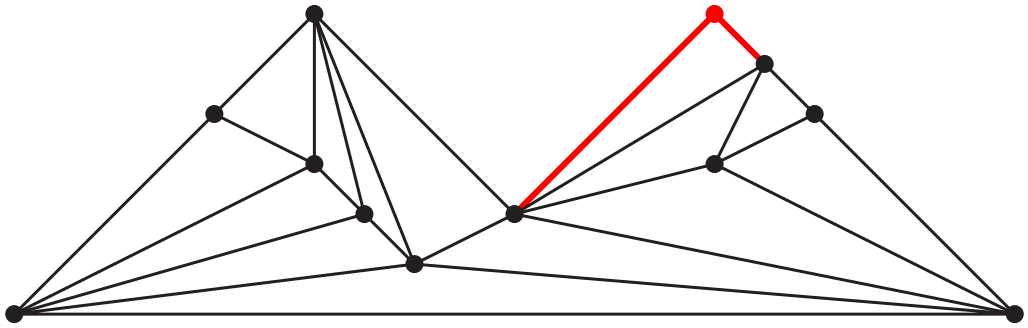
Beispiel



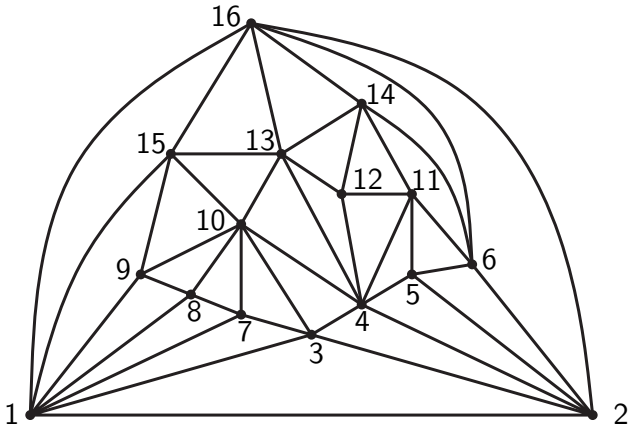


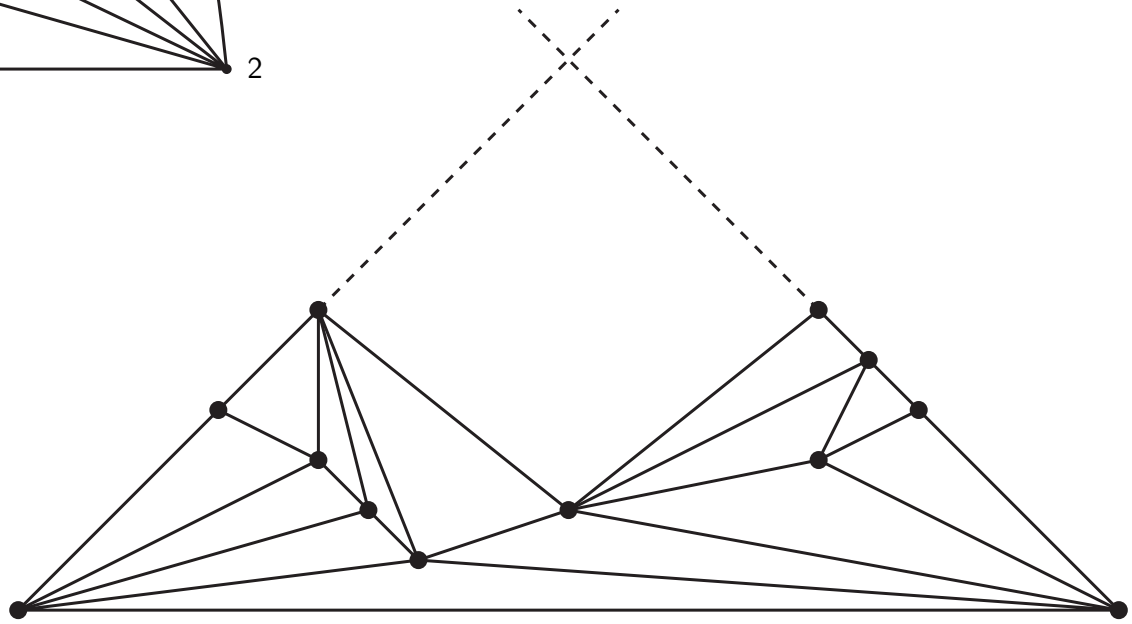
Beispiel



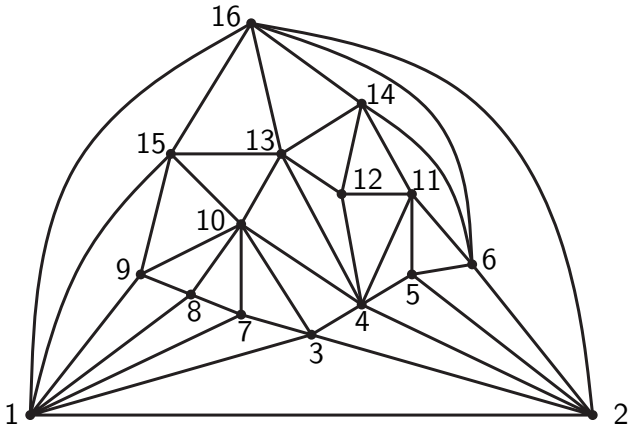


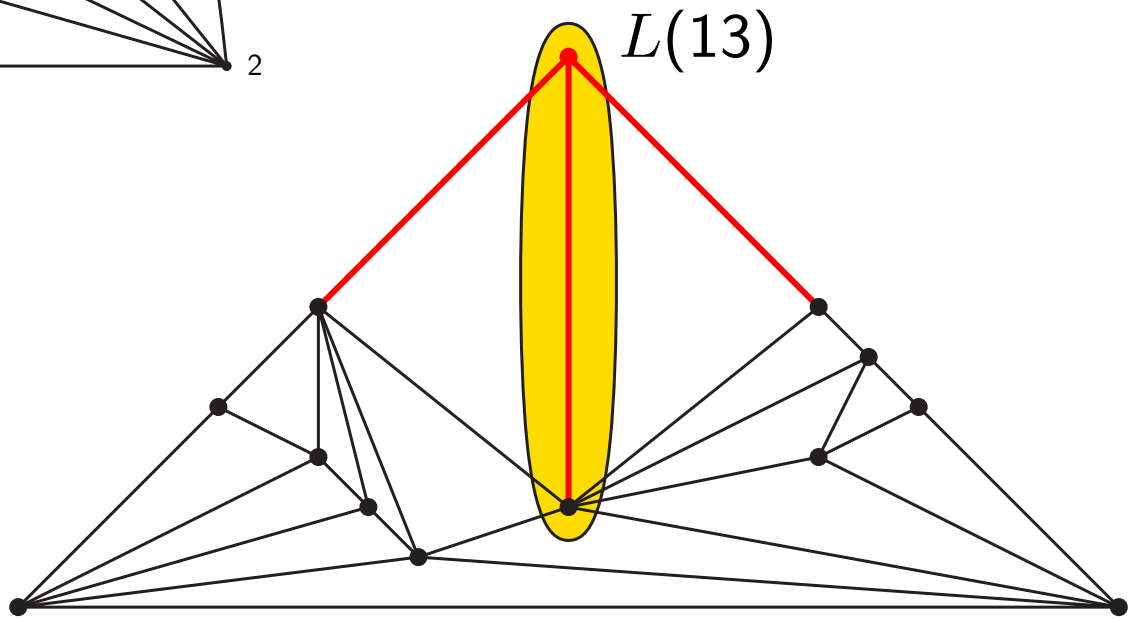
Beispiel



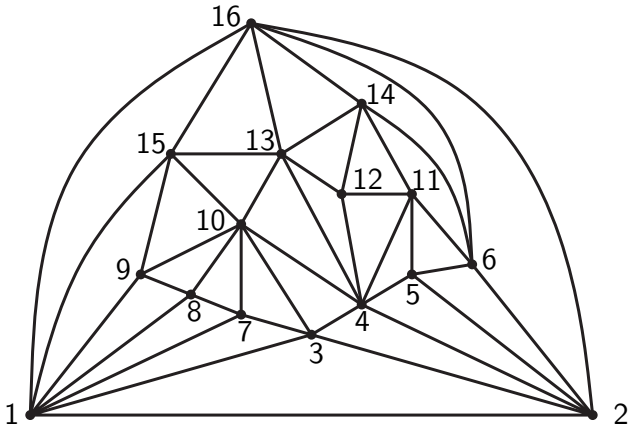


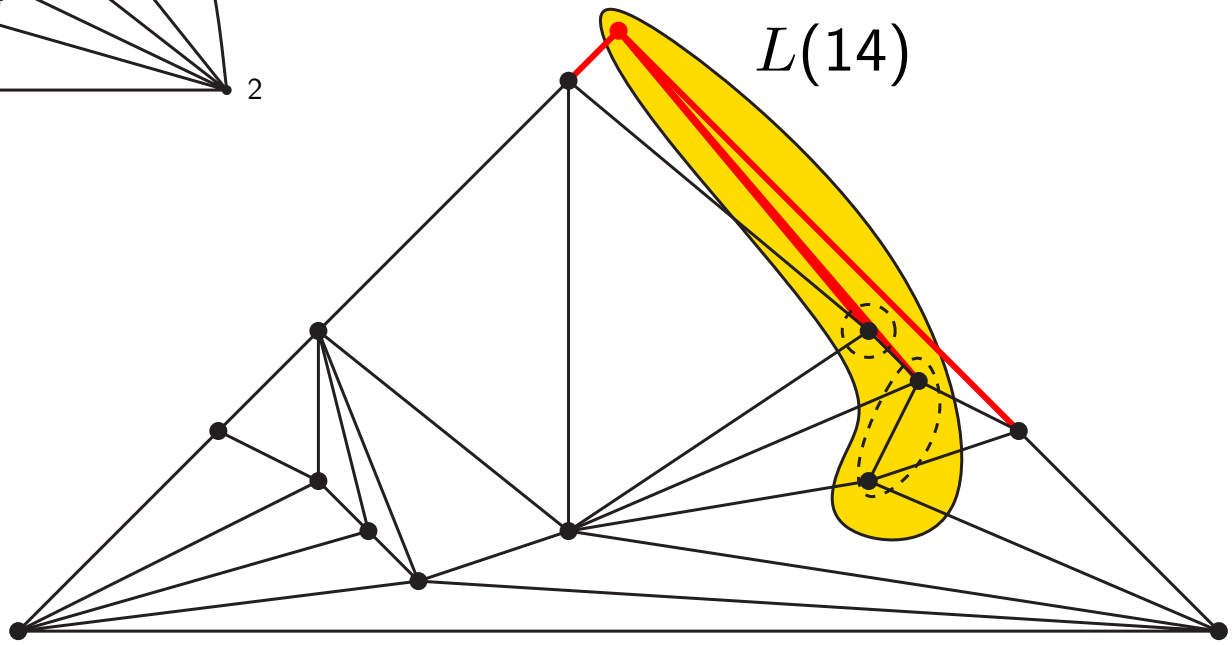
Beispiel



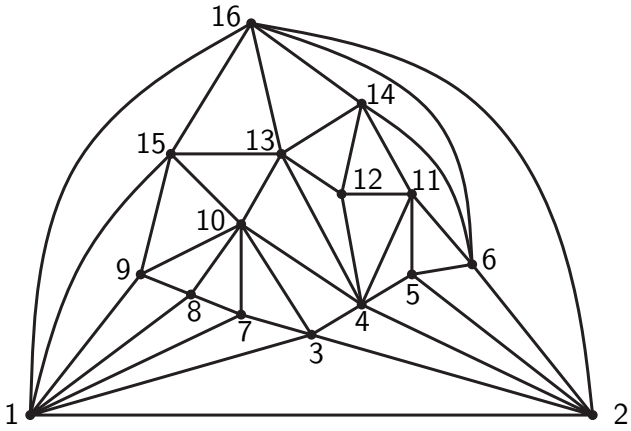


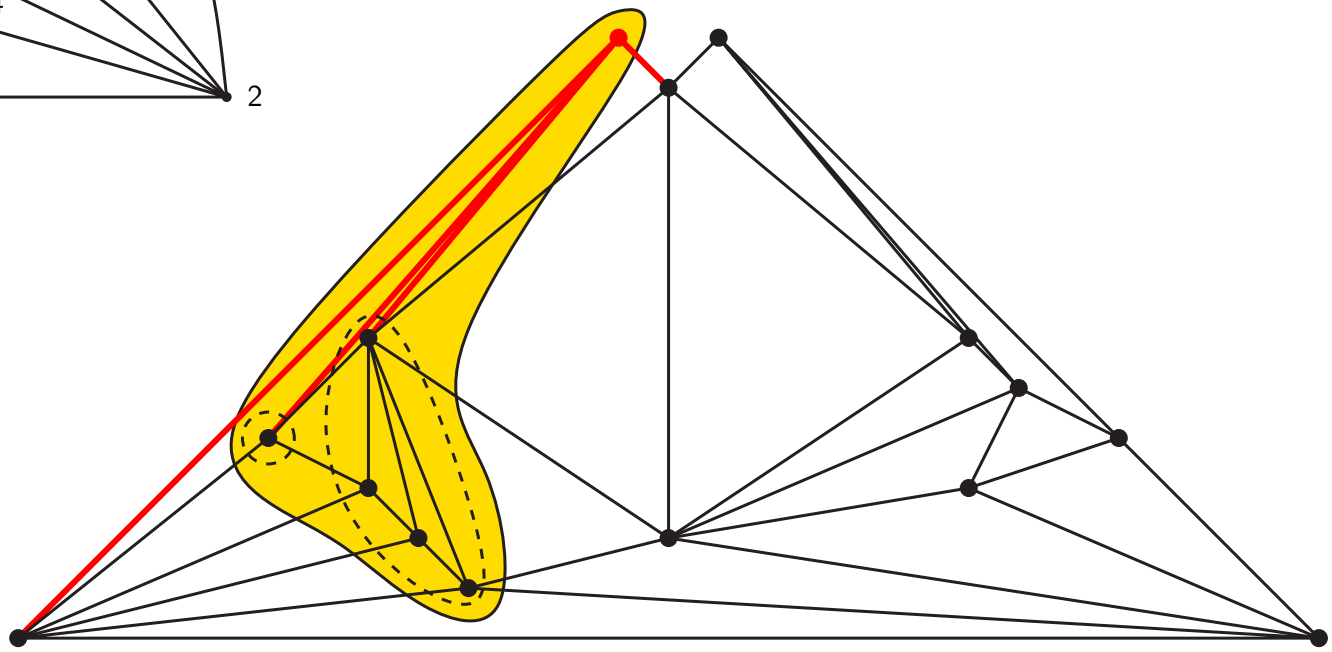
Beispiel



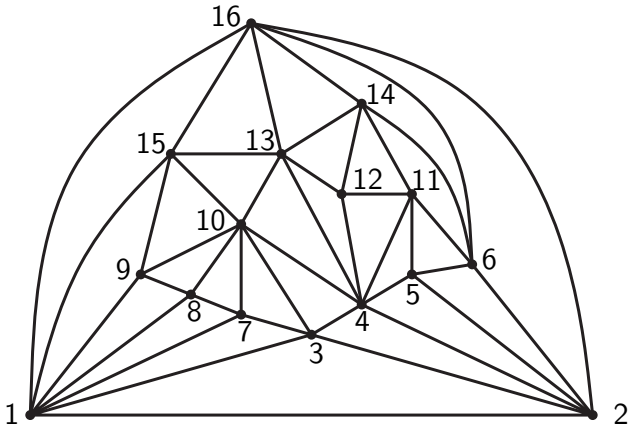


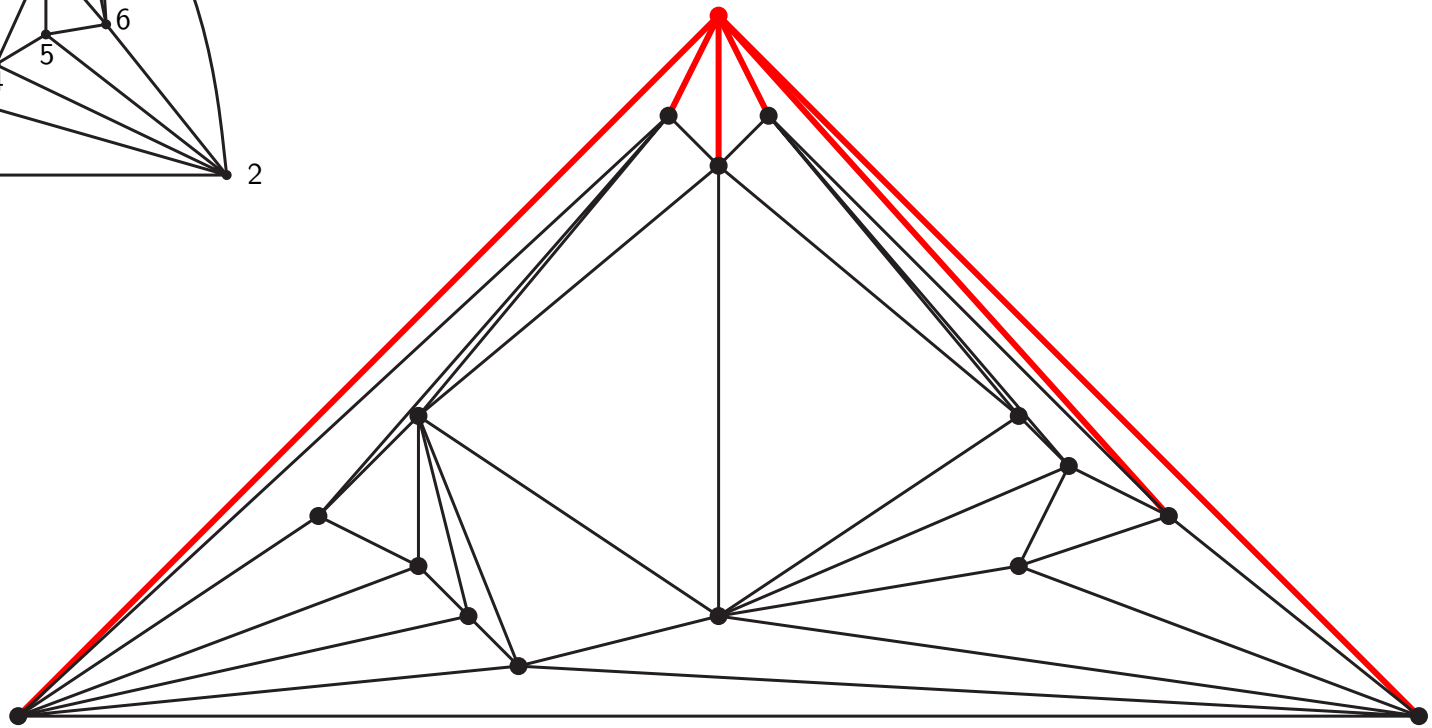
Beispiel



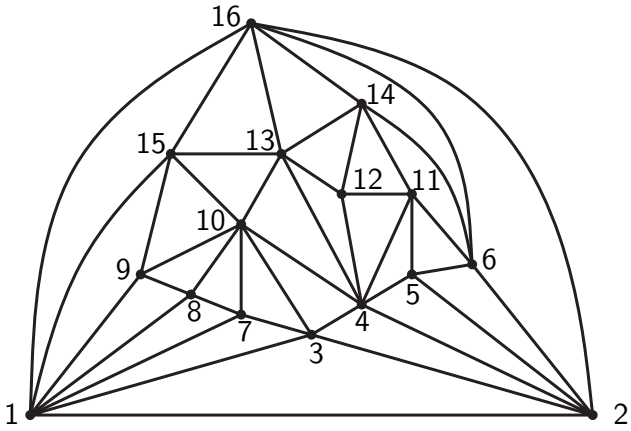


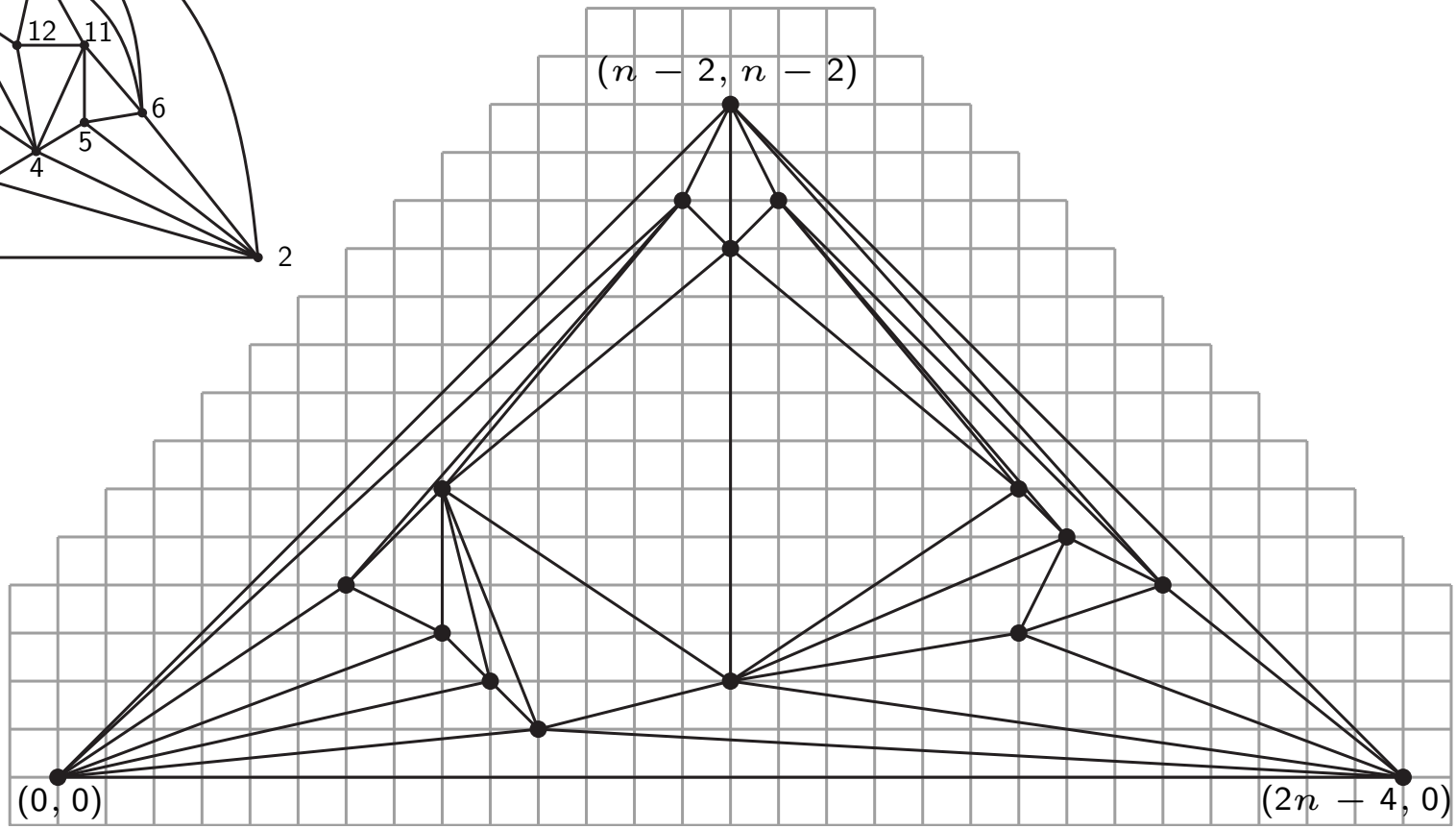
Beispiel





Beispiel





Aufgabe 4.1

Charakterisierung aufwärtsplanarer Graphen:

Für einen gerichteten azyklischen Graphen $D = (V, A)$ gilt:
aufwärtsplanar $\Leftrightarrow$ aufsp. Teilgraph eines planaren s-t-Graphen H

(s-t-Graph: azykl. Digraph mit eindeutiger Quelle s / Senke t und mit Kante (s, t))

$\Rightarrow$: Sei s Quelle mit tiefster y-Koord., t Senke mit höchster y-Koord.

1. Füge (s, t) ein
2. $\forall$ Senke $q \neq t$ suche vertikal nächste (u, v), baue (q, v) (oder (q, t))
3. Analog für Quellen

$\Leftarrow$:

1. Bestimme topologische Nummerierung von H für y-Koordinaten, respektiere zyklische Ordnung der Kanten, zeichne, z.B., knotenweise v.l.n.r., wird aufwärtsplanar.
2. D ist Subgraph von H, also D aufwärtsplanar.

Aufgabe 4.1

Charakterisierung aufwärtsplanarer Graphen:

Für einen gerichteten azyklischen Graphen $D = (V, A)$ gilt:
aufwärtsplanar $\Leftrightarrow$ aufsp. Teilgraph eines planaren s-t-Graphen H

(s-t-Graph: azykl. Digraph mit eindeutiger Quelle s / Senke t und mit Kante (s, t))

$\Rightarrow$: Sei s Quelle mit tiefster y-Koord., t Senke mit höchster y-Koord.

1. Füge (s, t) ein
2. $\forall$ Senke $q \neq t$ suche vertikal nächste (u, v), baue (q, v) (oder (q, t))
3. Analog für Quellen

$\Leftarrow$:

1. Bestimme topologische Nummerierung von H für y-Koordinaten, respektiere zyklische Ordnung der Kanten, zeichne, z.B., knotenweise v.l.n.r., wird aufwärtsplanar.
2. D ist Subgraph von H, also D aufwärtsplanar.

Aufgabe 5.3

(a) Sei $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ eine Permutation. Ein Paar (i, j) mit $1 \leq i < j \leq n$ heißt Inversion, wenn $\pi(i) > \pi(j)$. Entwerfen Sie einen Algorithmus, der die Anzahl der Inversionen einer Permutation von n Elementen in $O(n \log n)$ Zeit berechnet

Aufgabe 6.1

In der Vorlesung wurde ein einfaches Verfahren zum Zeichnen von Bäumen vorgestellt, das jedem Knoten v die Koordinaten $x(v) = \text{pre-/in-/postorder}(v)$ und $y(v) = -\text{tiefe}(v)$ zuordnet.

Garantiert dieses Verfahren auch dann noch eine kreuzungsfreie geradlinige Gitterzeichnung, wenn man für die x-Koordinaten die sogenannte *levelorder*-Nummerierung verwendet, die nacheinander in einer Breitensuche alle Knoten der gleichen Tiefe von links nach rechts besucht?

Aufgabe 6.4

Für den Algorithmus von Carlson und Eppstein zum konvexen Zeichnen von Bäumen mit optimaler Winkelauflösung aus der Vorlesung wurden drei einfache Basisfälle ausgeschlossen. Wie und mit welcher optimalen Winkelauflösung lässt sich

(a) ein Pfad

(b) eine nicht-eingebettete Ranke

(c) eine nicht-eingebettete Tripel-Ranke

konvex zeichnen?

Wie geht es weiter?

- Praktikum Graphengeneratoren im SS 11
- Vorlesung Algorithmische Geometrie im SS 11
- (Proseminar P $\neq$ NP Vermutung)
- Studien/Bachelor- und Master-/Diplomarbeiten:
  - Graphenclustern, Robustheit, Dynamik
  - Schematisches Graphenzeichnen
  - Sensornetzwerke
  - Routenplanung

Wie geht es weiter?

- Praktikum Graphengeneratoren im SS 11
- Vorlesung Algorithmische Geometrie im SS 11
- (Proseminar P $\neq$ NP Vermutung)
- Studien/Bachelor- und Master-/Diplomarbeiten:
  - Graphenclustern, Robustheit, Dynamik
  - Schematisches Graphenzeichnen
  - Sensornetzwerke
  - Routenplanung
- Sprechstunden in den Semesterferien:
  - ab 17. 02. – 25. 02.
  - ab 07 .03. (meistens)
  - nicht zw.16. 03. – 05. 04.

Wie geht es weiter?

- Praktikum Graphengeneratoren im SS 11
- Vorlesung Algorithmische Geometrie im SS 11
- (Proseminar P $\neq$ NP Vermutung)
- Studien/Bachelor- und Master-/Diplomarbeiten:
  - Graphenclustern, Robustheit, Dynamik
  - Schematisches Graphenzeichnen
  - Sensornetzwerke
  - Routenplanung

Viel Erfolg in der Prüfung!

- Sprechstunden in den Semesterferien:
  - ab 17. 02. – 25. 02.
  - ab 07 .03. (meistens)
  - nicht zw.16. 03. – 05. 04.