

Kapitel 1

Einführung

Zunächst einige Definitionen zu zentralen Begriffen dieser Vorlesung. Diese dienen vor allem der Erinnerung sowie der Auflösung von Mehrdeutigkeiten durch abweichende Verwendung in anderen Veranstaltungen.

1.1 Definition (Multimenge)

Eine Menge E zusammen mit einer Vielfachheit $\#_E : E \rightarrow \mathbb{N}_0$ ihrer Elemente heißt Multimenge. Wir schreiben kurz $\#e = \#_E(e)$ und $e \in_k E$, falls $\#e = k$, sowie $e \in \bar{E}$ bzw. $e \notin E$, falls $e \in_k E$ für ein $k \in \mathbb{N}$ bzw. $k = 0$.

Die Kardinalität (auch: Mächtigkeit) $|E| = \sum_{e \in E} \#e$.

Für Multimengen E, F gilt $E \subseteq F$, falls $e \in_k E \implies e \in_l F$ mit $k \leq l$. Ist F jedoch eine gewöhnliche Menge, so gilt $E \subseteq F$, falls $e \in E \implies e \in F$ (unabhängig von $\#e$).

1.2 Definition (Multigraph)

Ein (gerichteter) Multigraph ist ein Paar $G = (V, E)$ aus einer endlichen Menge V von Knoten und einer Multimenge $E \subseteq V \times V$ von Kanten. Kanten in $\{(v, v) : v \in V\}$ nennen wir Schleifen, und ein Multigraph ist schlicht, wenn er keine Schleifen hat. Kanten $e \in_k E$ mit $k > 1$ heißen Multikanten (auch: Mehrfachkanten).

1.3 Definition (symmetrischer Multigraph)

Ein Multigraph $G = (V, E)$ heißt symmetrisch, falls $(v, w) \in_k E \implies (w, v) \in_k E$, und anti-symmetrisch, falls $(v, w) \in E \implies (w, v) \notin E$. Zu einem beliebigen Multigraphen $G = (V, E)$ ist der symmetrisierte Multigraph

$G' = (V, E \cup E')$ definiert durch $(v, w) \in_{k'} E' \iff (v, w) \in_l E, (w, v) \in_k E, k' = k - l > 0$.

1.4 Bemerkung

Anti-symmetrische Multigraphen haben keine Schleifen.

1.5 Definition (Graph)

Ein Multigraph $G = (V, E)$ ohne Multikanten ($e \in_1 E$ für alle $e \in E$, d.h. E ist eine gewöhnliche Menge) ist ein (gewöhnlicher) Graph.

Zu einem Multigraphen $G = (V, E)$ vereinbaren wir außerdem:

- $n(G) = n = |V|$ und $m(G) = m = |E|$
- seine Größe $|G| = n + m$
- u Vorgänger von v , falls $(u, v) \in E$;
 $N_G^-(v) = N^-(v) = \{u : (u, v) \in E\}$ die Menge aller Vorgänger
- w Nachfolger von v , falls $(v, w) \in E$;
 $N_G^+(v) = N^+(v) = \{w : (v, w) \in E\}$ die Menge aller Nachfolger
- x adjazent (benachbart) zu v , falls x Vorgänger oder Nachfolger von v ;
 $N_G(v) = N(v) = N^-(v) \cup N^+(v)$ Menge aller Nachbarn von v
- e inzident zu v sowie v inzident zu e , falls $e = (u, v) \in E$ oder $e = (v, w) \in E$;
 e' inzident zu e , falls $e' = (u', w'), e = (u, w) \in E$ mit $|\{u, u', w, w'\}| < |\{u', w'\}| + |\{u, w\}|$.

Da für unsere Zwecke weitgehend gleichbedeutend, definieren wir ungerichtete Multigraphen als Vereinfachung symmetrischer Multigraphen. Alle anderen Begriffe (gewöhnliche Graphen, Nachbarn, usw.) übertragen sich entsprechend.

1.6 Definition (ungerichteter Multigraph)

Zu einem symmetrischen Multigraphen $\vec{G} = (V, \vec{E})$ ist der zugehörige ungerichtete Multigraph $\bar{G} = (V, \bar{E})$ definiert durch $\{u, w\} \in_k \bar{E} \iff (u, w) \in_k \vec{E}$ und umgekehrt.

Schließlich definieren wir noch eine Repräsentation von Multigraphen durch Matrizen. Diese wird später als Brücke zur Linearen Algebra dienen.

1.7 Definition (Adjazenzmatrix)

Zu einem Multigraphen $G = (V, E)$ ist die Adjazenzmatrix $A(G) = (a_{v,w})_{v,w \in V}$ definiert durch

$$a_{v,w} = k \iff (v, w) \in_k E .$$

1.8 Bemerkung

Die verschiedenen Graphenklassen sind wie oben definiert, damit die zugehörigen Adjazenzmatrizen gerade die folgenden Klassen bilden:

- (gerichtete) Multigraphen: $\mathbb{N}_0^{n \times n}$
- (gerichtete) Graphen: $\{0, 1\}^{n \times n}$

Symmetrische (Multi)graphen und ihre unterliegenden ungerichteten (Multi)graphen haben die gleiche, symmetrische Adjazenzmatrix und werden von uns daher auch oft gar nicht unterschieden. Die Adjazenzmatrix eines schleifenfreien (Multi)graphen hat auf der Diagonalen nur Nullen.

1.1 Gradfolgen

Wir beginnen mit *lokalen* Eigenschaften von Netzwerken.

1.9 Definition (Knotengrade)

Sind $\vec{G} = (V, \vec{E})$ ein Multigraph und $v \in V$, so heißen

- $d_{\vec{G}}^-(v) = d^-(v) = \sum_{(u,v) \in \vec{E}} \#(u, v)$ Eingangsgrad,
- $d_{\vec{G}}^+(v) = d^+(v) = \sum_{(v,w) \in \vec{E}} \#(v, w)$ Ausgangsgrad und
- $d_{\vec{G}}(v) = d(v) = d^-(v) + d^+(v)$ Knotengrad oder kurz Grad von v .

Ist $\bar{G} = (V, \bar{E})$ ein ungerichteter Multigraph, so definieren wir den (ungerichteten) Grad von v als $d_{\bar{G}}(v) = d(v) = \sum_{\substack{\{v,x\} \in \bar{E} \\ v \neq x}} \#\{v, x\} + 2 \cdot \#\{v, v\}$.

Für beliebige (Multi)graphen G definieren wir ferner

- $\delta(G) = \min_{v \in V} d(v)$ den minimalen Grad,
- $d(G) = \frac{1}{n} \cdot \sum_{v \in V} d(v)$ den durchschnittlichen Grad,
- $\Delta(G) = \max_{v \in V} d(v)$ den maximalen Grad,

sowie entsprechend für Ein- und Ausgangsgrade. Gilt $\delta(G) = d(G) = \Delta(G)$, so heißt G regulär.

1.10 Bemerkung

Für schlichte symmetrische Multigraphen \vec{G} ist $d_{\vec{G}}(v) = 2 \cdot d_G(v)$.

1.11 Bemerkung

In gewöhnlichen Graphen sind $d^-(v) = |N^-(v)|$ und $d^+(v) = |N^+(v)|$, aber $d(v) = |N(v)|$ nur in anti-symmetrischen oder schlichten ungerichteten Graphen.

1.12 Lemma

Jeder schlichte ungerichtete Graph mit mindestens zwei Knoten enthält zwei Knoten gleichen Grades.

■ **Beweis:** Gibt es keinen Knoten ohne Nachbarn, so liegen alle Knotengrade zwischen 1 und $n - 1$, sodass von den n Knoten mindestens zwei den gleichen Grad haben müssen (*Taubenschlagprinzip*).

Gibt es genau einen Knoten ohne Nachbarn, liegen die Knotengrade der $n - 1$ anderen Knoten zwischen 1 und $n - 2$. □

1.13 Lemma (Handschlaglemma)

In (gerichteten wie ungerichteten) Multigraphen gilt $\sum_{v \in V} d(v) = 2m$.

■ **Beweis:** Für gerichtete Multigraphen ist $\sum_{v \in V} d^-(v) = m = \sum_{v \in V} d^+(v)$, woraus wegen $d(v) = d^-(v) + d^+(v)$ für alle $v \in V$ die Behauptung folgt.

Ein ungerichteter Multigraph hat genau so viele Schleifen und halb so viele andere Kanten wie der zugehörige symmetrische Multigraph. Die Behauptung folgt wieder aus der Definition des (ungerichteten) Knotengrades. □

1.14 Definition (Gradfolgen)

Gegeben sei ein gerichteter oder ungerichteter Multigraph $G = (V, E)$ mit Knotenmenge $V = \{v_1, \dots, v_n\}$.

Die Folge $D(G) = ((d^-(v_1), d^+(v_1)), \dots, (d^-(v_n), d^+(v_n)))$ des gerichteten bzw. $D(G) = (d(v_1), \dots, d(v_n))$ des ungerichteten Multigraphen G heißt dessen Gradfolge.

Welche Folgen natürlicher Zahlen sind Gradfolgen bestimmter Multigraphklassen? Das Handschlaglemma liefert eine notwendige Bedingung, die jedoch nur für allgemeine Multigraphen hinreichend ist. (Überlegen Sie sich ein Verfahren, um aus einer Folge $((a_1, b_1), \dots, (a_n, b_n))$ mit $\sum_{i=1}^n a_i = \sum_{i=1}^n b_i$ einen Multigraphen mit ebendieser Gradfolge zu konstruieren!)

Existenz von Isomeren

1.15 Definition (konjugierte Zahlenfolge)

Zu einer Folge $D = (d_1, \dots, d_n)$ von natürlichen Zahlen sei $\Delta = \max_{j=1, \dots, n} d_j$. Die zu D konjugierte Folge $D^* = (d_1^*, \dots, d_\Delta^*)$ ist definiert durch

$$d_i^* = |\{d_j \geq i : j = 1, \dots, n\}| .$$

Für $i > \Delta$ wird $d_i^* = 0$ vereinbart.

Die konjugierte Folge lässt sich leicht aus einem *Ferrers-Diagramm* ablesen:

	d_1^*	d_2^*	d_3^*	d_4^*	d_5^*	
d_1	•	•	•	•	•	5
d_2	•	•	•	•		4
d_3	•	•				2
d_4	•	•				2
d_5	•					1
d_6	•					1
	6	4	2	2	1	15

Da es sich lediglich um zwei verschiedene Weisen des Abzählens handelt, gilt

$$\sum_{i=1}^n d_i = \sum_{i \geq 1} d_i^* .$$

1.16 Satz (Ryser 1957; Gale 1957)

Eine Folge $(a_1, b_1), \dots, (a_n, b_n)$ von Paaren natürlicher Zahlen mit $a_1 \geq \dots \geq a_n$ und $a_i, b_j \leq n$ ist genau dann die Gradfolge eines Graphen, wenn

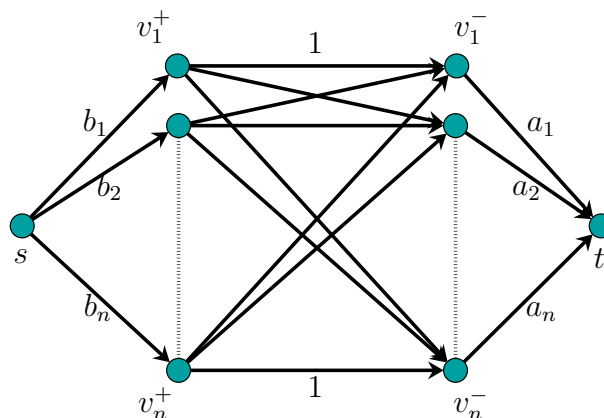
$$\sum_{i=1}^n a_i = \sum_{j=1}^n b_j^* \left(= \sum_{j=1}^n b_j \right)$$

und

$$\sum_{i=1}^k a_i \leq \sum_{j=1}^k b_j^* \left(= \sum_{j=1}^n \min\{b_j, k\} \right) \quad \text{für } k = 1, \dots, n-1$$

■ **Beweis:** Die erste Bedingung ist offensichtlich notwendig und man sieht leicht (wieder durch doppeltes Abzählen im Ferrers-Diagramm), dass die Gleichheiten auf der rechten Seite der zweiten Bedingung gelten.

Um zu zeigen, dass die zweite Bedingung ebenfalls notwendig und zusammen mit der ersten dann aber auch hinreichend ist, konstruieren wir ein bipartites Flussnetzwerk mit Knoten $s, v_1^+, \dots, v_n^+, v_1^-, \dots, v_n^-, t$ und Kanten (s, v_j^+) mit Kapazität b_j für alle $j = 1, \dots, n$, Kanten (v_j^+, v_i^-) mit Kapazität 1 für alle $i, j = 1, \dots, n$ und Kanten (v_i^-, t) mit Kapazität a_i für alle $i = 1, \dots, n$.



Ein maximaler st -Fluss, der alle zu s und t inzidenten Kanten saturiert, hat dann den Wert $\sum_{i=1}^n a_i = \sum_{j=1}^n b_j$. Zu jedem solchen Fluss definieren die flusstragenden Kanten (v_j^+, v_i^-) einen Graphen mit der gewünschten Gradfolge, und umgekehrt definiert jeder solche Graph einen entsprechenden Fluss.

Um zu zeigen, dass ein solcher Fluss genau dann existiert, wenn zusätzlich die zweite Bedingung gilt, betrachten wir Teilmengen $S \subseteq \{v_1^-, \dots, v_n^-\}$.

Der gewünschte Fluss existiert genau dann, wenn aus jedem beliebigen S höchstens so viel hinausfließen *muss* wie hineinfließen *kann*, d.h.

$$\sum_{i: v_i^- \in S} a_i \leq \sum_{j=1}^n \min\{b_j, |S|\} .$$

Da die a_i nicht-aufsteigend sortiert sind, gilt aber $\sum_{i: v_i^- \in S} a_i \leq \sum_{i=1}^{|S|} a_i$ für alle S . □

Die Charakterisierung erlaubt Schleifen, lässt sich aber mit Hilfe der folgenden Variante konjugierter Folgen für schlichte Graphen umformulieren.

1.17 Definition (korrigiert-konjugierte Zahlenfolge)

Zu einer nicht-aufsteigend sortierten Folge $D = (d_1, \dots, d_n)$ von natürlichen Zahlen ist die korrigiert-konjugierte Folge $\bar{D} = (\bar{d}_1, \dots, \bar{d}_{d_1+1})$ definiert durch

$$\bar{d}_i = |\{d_j \geq i - 1 : j = 1, \dots, i - 1\}| + |\{d_j \geq i : j = i + 1, \dots, n\}| .$$

Zu korrigiert-konjugierten Folgen lässt sich entsprechend ein *korrigiertes Ferrers-Diagramm* angeben:

	\bar{d}_1	\bar{d}_2	\bar{d}_3	\bar{d}_4	\bar{d}_5	\bar{d}_6	
d_1	×	•	•	•	•	•	5
d_2	•	×	•	•	•		4
d_3	•	•	·				2
d_4	•	•		·			2
d_5	•				·		1
d_6	•					·	1
	5	3	2	2	2	1	15

Auch für korrigiert-konjugierte Folgen gilt offensichtlich

$$\sum_{i=1}^n d_i = \sum_{i \geq 1} \bar{d}_i .$$

Statt der Charakterisierung für schlichte Graphen geben wir gleich eine für schlichte ungerichtete Graphen an.

1.18 Satz (Erdős und Gallai 1960)

Eine Folge $d_1 \geq \dots \geq d_n$ von natürlichen Zahlen ist genau dann die Gradfolge eines schlichten ungerichteten Graphen, wenn

$$\sum_{i=1}^n d_i \equiv 0 \pmod{2}$$

und

$$\sum_{i=1}^k d_i \leq \sum_{i=1}^k \bar{d}_i \quad \text{für } k = 1, \dots, n \quad (1.1)$$

■ **Beweis:** Gibt es einen schlichten ungerichteten Graphen G mit $D(G) = (d_1, \dots, d_n)$, so können wir o.E. $d_1 \geq \dots \geq d_n$ annehmen. Die erste Bedingung folgt dann aus dem Handschlaglemma. Wir folgern die zweite Bedingung aus dem Satz über die Gradfolgen von gerichteten Graphen, indem wir den zugehörigen symmetrischen Graphen so modifizieren, dass alle Knoten mit $d_i \geq i$ eine Schleife erhalten. Die Gradfolge dieses gerichteten Graphen sei $((a_1, b_1), \dots, (a_n, b_n))$. Dann ist für einen Index q ($q < n$ da $d_n \leq n - 1$)

$$a_i = b_i = \begin{cases} d_i + 1 & \text{für } i \leq q, \\ d_i & \text{sonst.} \end{cases}$$

Aus dem Ferrers-Diagramm erhalten wir außerdem

$$b_j^* = \begin{cases} \bar{d}_j + 1 & \text{für } j \leq q, \\ \bar{d}_j & \text{sonst.} \end{cases}$$

Mit dem Satz für gerichtete Graphen folgt dann

$$\begin{array}{ccc} \sum_{i=1}^k a_i & \leq & \sum_{i=1}^k b_i^* \\ \parallel & & \parallel \\ \min\{q, k\} + \sum_{i=1}^k d_i & & \sum_{i=1}^k \bar{d}_i + \min\{q, k\} \end{array}$$

für $k = 1, \dots, n - 1$. Die Ungleichung gilt auch für den Fall $k = n$, denn wegen $d_i \leq n - 1$ ist $\sum_{i=1}^n d_i = \sum_{i \geq 1} \bar{d}_i = \sum_{i=1}^n \bar{d}_i$.

Die umgekehrte Richtung ist deutlich schwieriger und wird hier nicht bewiesen. Idee: Zeige zunächst die Existenz eines schlichten gerichteten Graphen

mit entsprechenden Ein- und Ausgangsgraden und transformiere diesen sukzessive in einen schlichten symmetrischen Graphen mit gleicher Gradfolge. Dessen zugehöriger ungerichteter Graph ist der gesuchte. \square

1.19 Satz

Bedingung (1.1) ist äquivalent zu

$$\sum_{i=1}^k d_i \leq k(k-1) + \sum_{i=k+1}^n \min\{d_i, k\} \quad \text{für } k = 1, \dots, n. \quad (1.2)$$

■ **Beweis:** Zu gegebener Folge $d_1 \geq \dots \geq d_n$ sei c_k die Anzahl der leeren Felder (weder \bullet noch \times oder \cdot) im Quadrat $\{d_1, \dots, d_k\} \times \{\bar{d}_1, \dots, \bar{d}_k\}$ des korrigierten Ferrers-Diagramms. Dann lässt sich Bedingung (1.2) schreiben als

$$\sum_{i=1}^k d_i \leq \sum_{i=1}^k \bar{d}_i + c_k \quad (1.3)$$

Wegen $c_k \geq 0$ folgt (1.3), und damit auch (1.2), aus (1.1).

Die Umkehrung beweisen wir durch einen Widerspruch. Angenommen, es gilt (1.3), aber

$$\sum_{i=1}^k d_i > \sum_{i=1}^k \bar{d}_i$$

für ein $k \leq n$ (insbesondere also $c_k > 0$). Da das Feld $(1, 1)$ im korrigierten Ferrers-Diagramm nicht leer ist, gilt $c_1 = 0$ und damit $k > 1$. Sei nun q der größte Index mit $d_q \geq k - 1$. Dann ist $q < k$, weil $c_k > 0$, und wir erhalten zunächst

$$\sum_{i=1}^k d_i > \sum_{i=1}^k \bar{d}_i = q(k-1) + \sum_{i=q+1}^k d_i + \sum_{i=k+1}^n d_i$$

und durch einfache Umformung $\sum_{i=1}^q d_i > q(k-1) + \sum_{i=k+1}^n d_i$.

Im Widerspruch dazu folgt aus (1.2)

$$\begin{aligned}
 \sum_{i=1}^q d_i &\leq q(q-1) + \sum_{i=q+1}^n \min\{d_i, q\} \\
 &\leq q(q-1) + \sum_{i=q+1}^k \min\{d_i, q\} + \sum_{i=k+1}^n d_i \\
 &\leq q(q-1) + (k-q)q + \sum_{i=k+1}^n d_i \\
 &\leq q(k-1) + \sum_{i=k+1}^n d_i .
 \end{aligned}$$

□

Die obigen Sätze sind globale Kriterien für die Realisierbarkeit von Zahlenfolgen als Gradfolgen, die in Linearzeit getestet werden können. Zum Abschluss behandeln wir ein rekursives Kriterium, das unmittelbar auf einen Konstruktionsalgorithmus führt.

1.20 Satz (Havel 1955; Hakimi 1962)

Eine Folge $d_1 \geq \dots \geq d_n$ mit $d_1 \leq n-1$ ist genau dann Gradfolge eines schlichten ungerichteten Graphen, wenn auch $d_2-1, \dots, d_{d_1+1}-1, d_{d_1+2}, \dots, d_n$ Gradfolge eines solchen Graphen ist.

Der Beweis ergibt sich unmittelbar aus der folgenden Beobachtung.

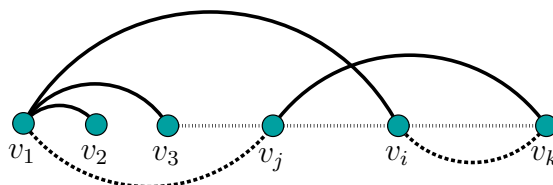
1.21 Lemma

Zu einer Folge $D = (d_1, \dots, d_n)$ mit $d_1 \geq \dots \geq d_n$ sei \mathcal{G}_D die Menge aller schlichten ungerichteten Graphen mit Knoten $\{v_1, \dots, v_n\}$, in denen v_i den Grad d_i hat. Ist \mathcal{G}_D nicht leer, dann enthält sie einen Graphen mit $N(v_1) = \{v_2, \dots, v_{d_1+1}\}$.

■ **Beweis:** Da für $d_1 = n-1$ jeder Graph in \mathcal{G}_D die Bedingung erfüllt, nehmen wir $d_1 < n-1$ an. Zu $G = (V, E) \in \mathcal{G}_D$ definieren wir dann $j(G)$ als den kleinsten Index mit $\{v_1, v_j\} \notin E$ und wählen einen solchen Graphen $G = (V, E) \in \mathcal{G}_D$, für den $j(G)$ maximal ist.

Wir müssen zeigen, dass $j(G) = d_1 + 2$ und nehmen daher an, es wäre $j = j(G) < d_1 + 2$. Von den d_1 Nachbarn von v_1 können dann höchstens

$d_1 - 1$ einen Index kleiner j haben, sodass es ein $v_i \in N(v_1)$ mit $i > j$ geben muss. Daraus folgt $d_j \geq d_i$, und wegen $v_1 \in N(v_i) \setminus N(v_j)$ existiert dann auch ein weiterer Knoten $v_k \in N(v_j) \setminus N(v_i)$.



Der Graph $G' = (V, E')$ mit $E' = (E \setminus \{\{v_1, v_i\}, \{v_j, v_k\}\}) \cup \{\{v_1, v_j\}, \{v_i, v_k\}\}$ ist dann aber ebenfalls in \mathcal{G}_D und $j(G') > j = j(G)$ im Widerspruch zur Maximalität. \square

1.2 Teilgraphen, Wege und Zusammenhang

1.22 Definition (Teilgraph)

Ein Multigraph $G = (V, E)$ enthält einen Multigraphen $G' = (V', E')$, falls $V' \subseteq V$ und $E' \subseteq E$. Wir nennen G' auch Teilgraph von G und schreiben $G' \subseteq G$.

1.23 Definition (Weg und Kreis)

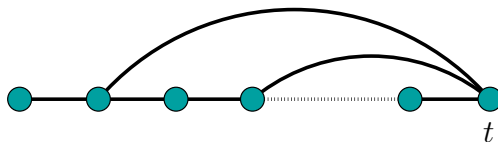
Ein (gerichteter) Weg (auch: Pfad) der Länge k ist ein Graph $P_k = (V, E)$ mit $V = \{v_1, \dots, v_{k+1}\}$ und $E = \{(v_1, v_2), \dots, (v_k, v_{k+1})\}$, wobei $|E| = k$ verlangt wird. Wir nennen P_k auch einen (v_1, v_{k+1}) -Weg. Der Graph $C_{k+1} = (V, E \cup \{(v_{k+1}, v_1)\})$, $(v_{k+1}, v_1) \notin E$, heißt (gerichteter) Kreis (auch: Zykel) der Länge $k + 1$. Die Graphen P_k und C_{k+1} heißen einfach, falls $|V| = k + 1$ (kein Knoten kommt doppelt vor).¹

1.24 Satz

Jeder schlichte ungerichtete Graph G enthält einen Weg der Länge $\delta(G)$ und, falls $\delta(G) \geq 2$, auch einen Kreis der Länge mindestens $\delta(G) + 1$.

¹Unter einem ungerichteten Weg bzw. Kreis verstehen wir sowohl den zugehörigen ungerichteten Graphen, als auch den Graphen, der aus einem Weg bzw. Kreis durch Umdrehen beliebig vieler seiner Kanten entsteht.

■ **Beweis:** Betrachte den letzten Knoten t eines längsten Weges in G . Da es keinen längeren Weg gibt, müssen alle $d(t) \geq \delta(G)$ Nachbarn von t Knoten des Weges sein, und weil G schlicht ist, handelt es sich um verschiedene Knoten.



Der erste zu t benachbarte Wegknoten schließt einen Kreis der Länge mindestens $\delta(G) + 1$. □

1.25 Definition (Zusammenhang)

Ein Multigraph $G = (V, E)$ heißt stark zusammenhängend, falls er für jedes Paar $v, w \in V$ sowohl einen (v, w) -Weg als auch einen (w, v) -Weg enthält. G heißt (schwach) zusammenhängend, wenn der symmetrisierte Multigraph stark zusammenhängend ist.

1.26 Definition (mehrfacher Zusammenhang)

Ein ungerichteter Multigraph heißt k -fach (knoten)zusammenhängend, falls jeder durch Entfernung von höchstens $k - 1$ beliebigen Knoten (und aller inzidenten Kanten) entstehende Multigraph zusammenhängend ist. Der Multigraph heißt k -fach kantenzusammenhängend, falls jeder durch Entfernung von höchstens $k - 1$ beliebigen Kanten entstehende Multigraph zusammenhängend ist.

1.27 Definition (Komponenten)

Zu einem schlichten Multigraphen heißt ein inklusionsmaximaler zusammenhängender (stark zushgd., k -fach zushgd., k -fach kantenzushgd.) Teilgraph (starke, k -fache, k -fache Kanten-) Zusammenhangskomponente.²

²In Multigraphen mit Schleifen werden diese zu den eindeutigen (starken, k -fachen Kanten-) Zusammenhangskomponenten ihrer Knoten hinzugerechnet, für $k > 1$ aber als eigene k -fache Zusammenhangskomponente gezählt.

Tiefensuche

Linearzeitalgorithmen zur Bestimmung von Zusammenhangskomponenten können als Spezialisierungen der folgenden Form der Tiefensuche formuliert werden. Die Funktionen `root`, `traverse` und `backtrack` werden dazu abhängig vom Komponententyp implementiert.

Algorithmus 1: (Gerichtete) Tiefensuche (*depth-first search*, DFS)

Eingabe: Multigraph $G = (V, E)$
Daten: Stack S (für Knoten auf DFS-Pfad)
 Knotenarray `incoming` (für erste Eingangskante)
 Knoten- und Kantenmarkierungen

```

foreach  $s \in V$  do
  if  $s$  nicht markiert then
    markiere  $s$ 
    incoming[s]  $\leftarrow$  nil
    push  $s \rightarrow S$ 
     $\rightarrow$  root(s)
  while  $S$  nicht leer do
     $v \leftarrow \text{top}(S)$ 
    if ex. ein nicht markiertes  $e = (v, w) \in E$  then
      markiere  $e$ 
      if  $w$  nicht markiert then
        markiere  $w$ 
        incoming[w]  $\leftarrow$   $e$ 
        push  $w \rightarrow S$ 
       $\rightarrow$  traverse(v, e, w)
    else
       $w \leftarrow \text{pop}(S)$ 
       $\rightarrow$  backtrack(w, incoming[w], top(S))3

```

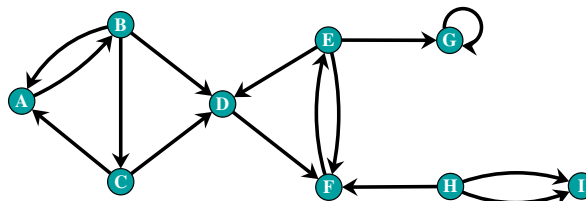
Ist v_1, \dots, v_n die Reihenfolge, in der die Knoten markiert werden, so heißt $DFS(v_i) = i$ die DFS-Nummer von v_i . Die DFS-Nummer $DFS((v, w)) = DFS(v)$ einer Kante sei die DFS-Nummer des Knotens, von dem aus sie durchlaufen wird. Wir definieren die Tiefensuch(halb)ordnung auf $V \cup E$:

$$p \prec q \iff DFS(p) < DFS(q) \quad \text{für alle } p, q \in V \cup E .$$

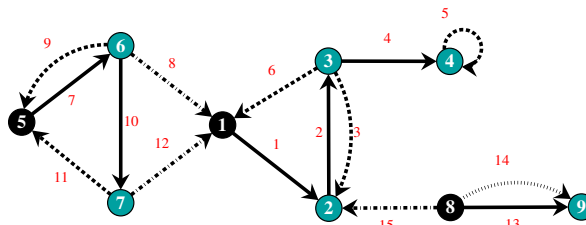
³Für einen leeren Stack S sei $\text{top}(S) = \text{nil}$.

Die Kanten werden während der Tiefensuche wie folgt klassifiziert. Zum Zeitpunkt, da die Kante (v, w) markiert wird, wird sie zu einer

- Baumkante (\longrightarrow), falls w noch nicht markiert ist,
- Rückwärtskante (\dashrightarrow), falls w markiert ist, $w \preceq v$ und $w \in S$,
- Querkante (\dashrightarrow), falls w markiert ist, $w \prec v$ und $w \notin S$, und
- Vorwärtskante (\dashrightarrow), falls w markiert ist und $v \prec w$.



Multigraph



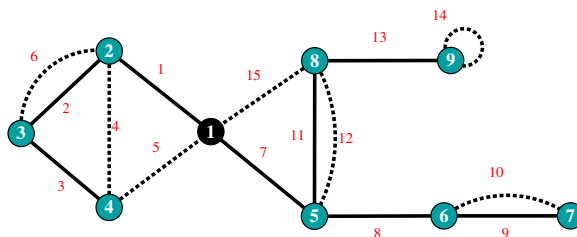
(gerichtete) Tiefensuche

Indizes entsprechen Durchlaufreihenfolge der Knoten bzw. Kanten

Ersetzt man die eingekastete Kantenauswahl durch

$$e = (v, w) \in E \text{ or } e = (w, v) \in E,$$

so werden die Kanten unabhängig von ihrer Richtung durchlaufen, z.B.:



ungerichtete Tiefensuche

Die folgenden Algorithmen benutzen alle das durch die Tiefensuche gegebene Gerüst. Wir geben daher nur die unterschiedlichen Versionen der drei ausgelassenen Methoden an. Alle diese Algorithmen können so implementiert werden, dass sie linear (in der Größe des Multigraphen) viel Zeit und Platz benötigen. In objekt-orientierten Programmen kann die Tiefensuche dann z.B. als abstrakte Klasse (von der speziellere Algorithmen ableiten) oder auch als Iterator über die Kanten (dessen Ausgabe die Spezialisierungen schrittweise verarbeiten) implementiert werden.

(Schwache) Zusammenhangskomponenten

Algorithmus 2: (Schwache) Zusammenhangskomponenten
(Spezialisierung der ungerichteten Tiefensuche)

Daten : Komponente c (repräsentiert durch ersten Knoten)

Ausgabe: Knoten- und Kantenarray **component**
(zeigt auf den Repräsentanten)

root(vertex s) begin
 | $c \leftarrow s$; $\text{component}[s] \leftarrow c$
end

traverse(vertex v , edge e , vertex w) begin
 | $\text{component}[e] \leftarrow c$
 | $\text{component}[w] \leftarrow c$
end

Zweifache Zusammenhangskomponenten

1.28 Lemma

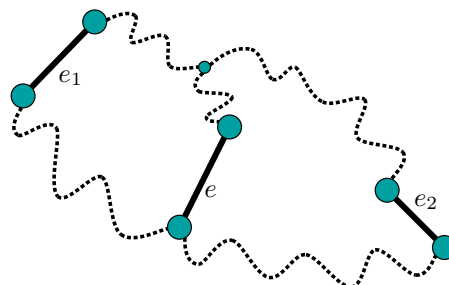
Zwei Kanten sind genau dann in einer zweifachen Zusammenhangskomponente, wenn es einen einfachen ungerichteten Kreis gibt, der beide enthält. Jede Kante liegt in genau einer zweifachen Zusammenhangskomponente.

■ **Beweis:** Zu zwei gegebenen Kanten eines zweifach zusammenhängenden Multigraphen betrachte den ebenfalls zweifach zusammenhängenden Multigraphen, der dadurch entsteht, dass die Kanten durch je einen neuen Knoten unterteilt werden. Wir erhalten dann einen einfachen ungerichteten Kreis aus dem *Satz von Menger* (knotendisjunkte Version): In einem zweifach zusammenhängenden ungerichteten Multigraphen mit mindestens zwei Kanten gibt es zwischen je zwei Knoten zwei knotendisjunkte Wege.



Umgekehrt ist jeder einfache ungerichtete Kreis zweifach zusammenhängend, sodass je zwei seiner Kanten in einer zweifachen Zusammenhangskomponente liegen.

Eine Kante e kann nicht in zwei verschiedenen zweifachen Zusammenhangskomponenten liegen, denn wegen der Inklusionsmaximalität müsste es sonst in jeder dieser Komponenten eine weitere Kante e_1 bzw. e_2 geben, die nicht in der jeweils anderen liegt. Zu diesen gibt es je einen einfachen ungerichteten Kreis, auf dem sie gemeinsam mit e liegen. Die Vereinigung der beiden Kreise enthält einen einfachen ungerichteten Kreis, auf dem e_1 und e_2 liegen.



□

Jede zweifache Zusammenhangskomponente ist also Vereinigung von einfachen ungerichteten Kreisen. Wir nutzen nun aus, dass bei der ungerichteten Tiefensuche jede Rückwärtskante einen Kreis schließt und jeder Kreis eine Rückwärtskante enthält.

Algorithmus 3: Zweifache Zusammenhangskomponenten
(Spezialisierung der ungerichteten Tiefensuche)

Daten : Stack S_E (Kanten in offenen Komponenten)
Stack C (offene Komponenten, repräsentiert durch erste Kante)

Ausgabe: Kantenarray component (zeigt auf Repräsentanten)

traverse(vertex v , edge e , vertex w) begin

if e ist Schleife **then** component[e] $\leftarrow e$

else

 push $e \rightarrow S_E$

if e ist Baumkante **then** push $e \rightarrow C$

if e ist Rückwärtskante **then**

while $w \prec top(C)$ **do** pop(C)

end

backtrack(vertex w , edge e , vertex v) begin

if $e = top(C)$ **and** $e \neq nil$ **then**

 pop(C)

repeat

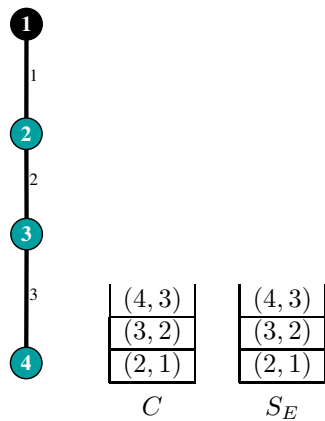
$e' \leftarrow pop(S_E)$

 component[e'] $\leftarrow e$

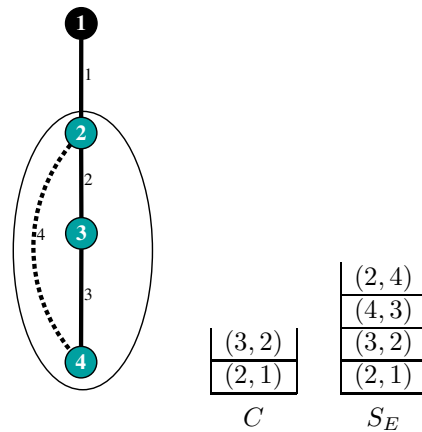
until $e' = e$

end

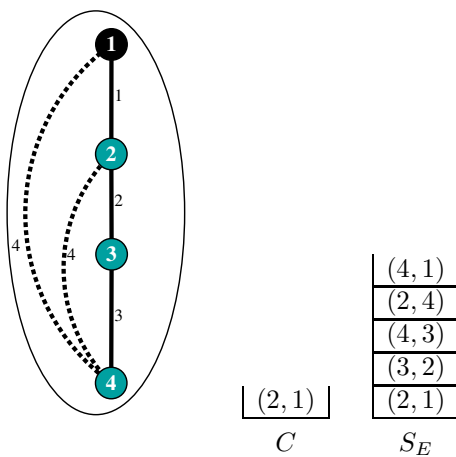
Mit der ungerichteten Tiefensuchreihenfolge aus dem obigen Beispiel durchläuft der Algorithmus zur Bestimmung der zweifachen Zusammenhangskomponenten die folgenden Zwischenstufen (Indizes entsprechen DFS-Nummern):



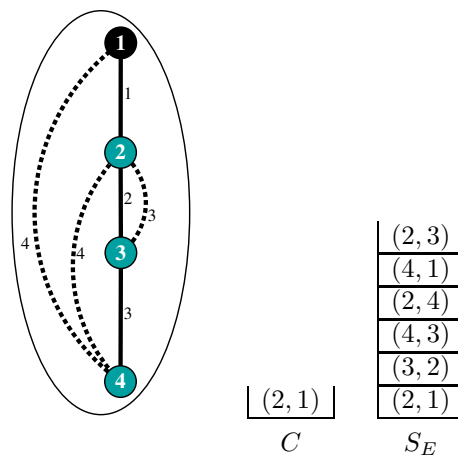
nach $\text{traverse}(3, (4, 3), 4)$



nach $\text{traverse}(4, (2, 4), 2)$

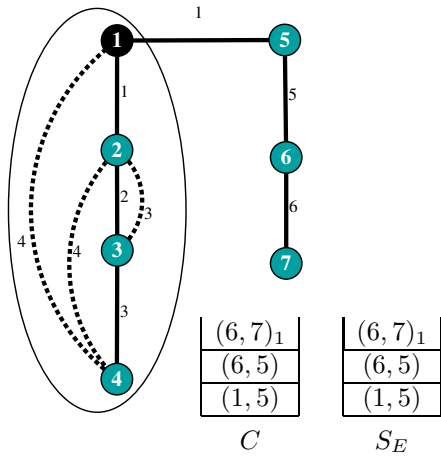


nach $\text{traverse}(4, (4, 1), 1)$

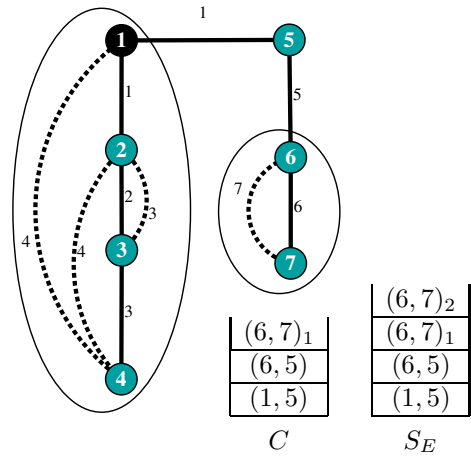


nach $\text{traverse}(3, (2, 3), 2)$

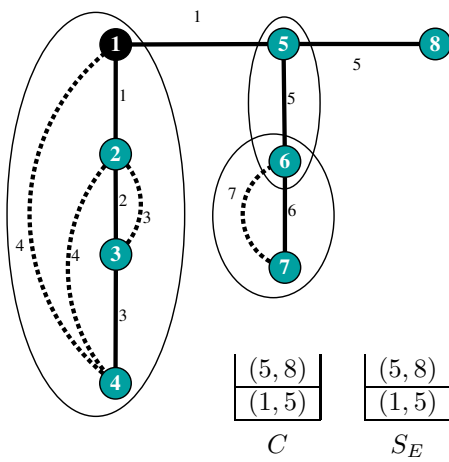
nach $\text{backtrack}(2, (2, 1), 1)$
sind C und S_E leer



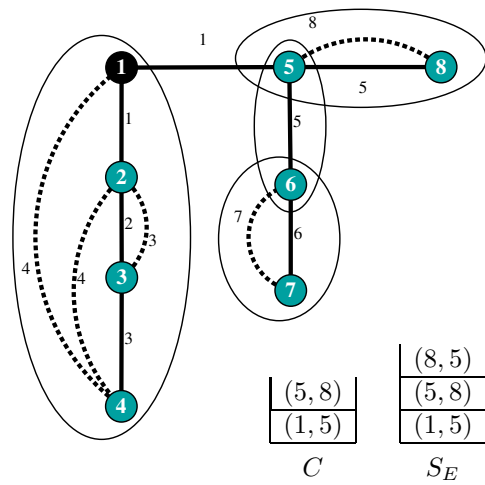
nach $\text{traverse}(6, (6, 7)_1, 7)$



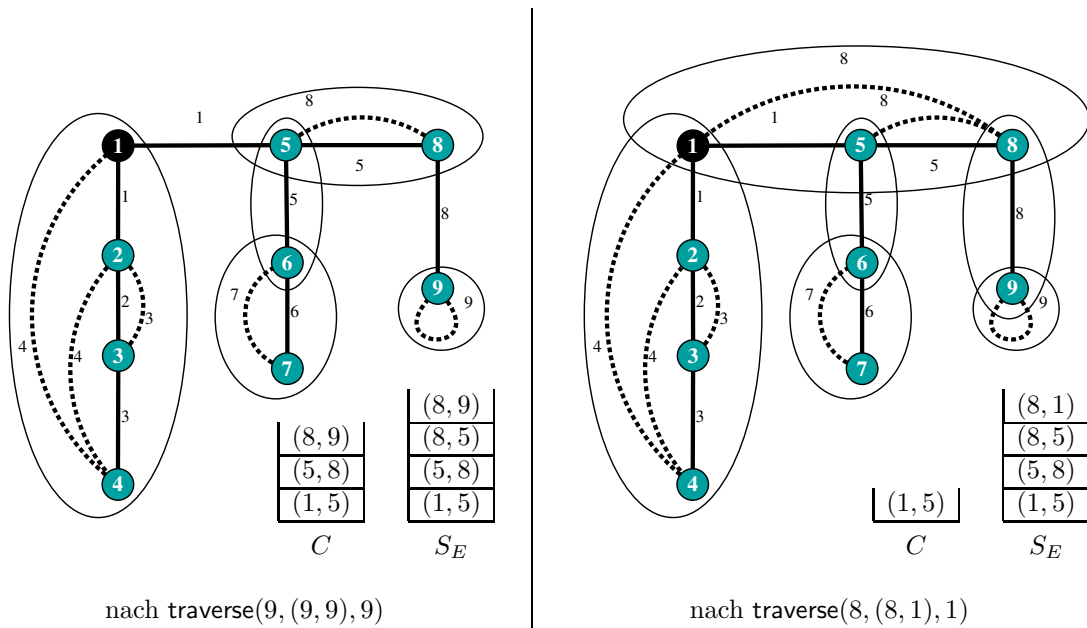
nach $\text{traverse}(7, (6, 7)_2, 6)$



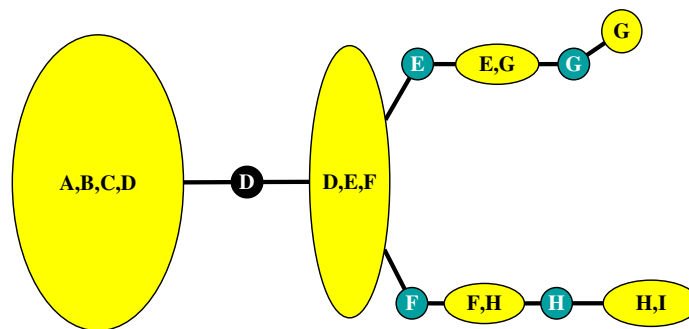
nach $\text{traverse}(5, (5, 8), 8)$



nach $\text{traverse}(8, (8, 5), 5)$



Eine kompakte Darstellung der zweifachen Zusammenhangskomponenten eines Multigraphen ist der (bipartite) Block-Schnittknoten Baum, in dem eine Sorte Knoten die Komponenten (Blöcke), und die andere Sorte die Schnittknoten (Knoten, die in mehr als einer Komponente liegen, weil ihre Herausnahme den Multigraph unzusammenhängend macht) repräsentiert.



Block-Schnittknoten Baum

1.29 Satz

Algorithmus 3 berechnet die zweifachen Zusammenhangskomponenten eines Multigraphen in Linearzeit.

■ **Beweis:** Wir beweisen die Korrektheit des Algorithmus durch Induktion über die Anzahl der Aufrufe von `traverse` und `backtrack`. Für jede Kante des Graphen wird einmal `traverse` aufgerufen, `backtrack` aber nur für Baumkanten. Wir nennen eine markierte Baumkante, über die noch kein Backtracking erfolgt ist, *offen*. Jede andere markierte Kante ist *fertig*. Zu jedem Zeitpunkt induzieren die offenen Kanten einen Weg. Zu jeder zweifachen Zusammenhangskomponente des markierten Teilgraphen gibt es eine eindeutige *erste* Kante, und eine Komponente heißt offen (fertig), wenn ihre erste Kante offen (fertig) ist.

Nach den ersten t Aufrufen von `traverse` und `backtrack` sei G_t der von den markierten Kanten induzierte Teilgraph. Die offenen zweifachen Zusammenhangskomponenten, ihre Kantenmengen und ihre ersten Kanten bezeichnen wir mit $G_t^{(i)}$, $E_t^{(i)}$ und $e_t^{(i)}$, $1 \leq i \leq k_t$, in der Reihenfolge, in der die $e_t^{(i)}$ markiert wurden. Wir zeigen die folgenden Invarianten:

1. Die Kanten jeder fertigen zweifachen Zusammenhangskomponenten zeigen auf deren erste Kante.
2. Auf C liegen $e_t^{(1)} \prec \dots \prec e_t^{(k_t)}$ in dieser Reihenfolge.
3. Auf S_E liegen die Kanten aus $E_t^{(1)}, \dots, E_t^{(k_t)}$ in dieser Reihenfolge.

Natürlich sind alle drei Aussagen vor dem ersten Aufruf richtig. Wir setzen daher voraus, dass sie für ein $t > 0$ nach $t - 1$ Aufrufen gelten und unterscheiden zwei Fälle.

1. *Fall* (Aufruf von `traverse`(v, e, w)):

Im Sonderfall, dass e eine Schleife ist, wird diese unmittelbar zu einer eigenen Komponente. Ist e eine Baumkante, so ist w ein Knoten vom Grad 1 in G_t und damit e die einzige Kante einer neuen offenen Komponente. Alle drei Invarianten bleiben richtig. Ist e eine Rückwärtskante aber keine Schleife, so bildet sie zusammen mit dem letzten, bei $w \preceq v$ beginnenden Teilstück des Weges der offenen Kanten einen einfachen Kreis. Alle auf diesem Kreis liegenden Kanten gehören zu derselben zweifachen Zusammenhangskomponente von G_t , und da alle Kanten e' mit $w \prec e'$ von C entfernt werden, gelten die Invarianten weiterhin.

2. *Fall* (Aufruf von `backtrack(w, e, v)`):

Im Fall $e = \text{nil}$ ist w ein Knoten, der in der äußeren Schleife der Tiefensuche auf den Stack S gelegt wurde. Alle Kanten der (schwachen) Zusammenhangskomponente und damit auch alle ihre zweifachen Zusammenhangskomponenten waren damit schon beim vorhergehenden Aufruf fertig, sodass C und S_E leer sind. Andernfalls wird die Kante e fertig, und aufgrund der zweiten Invariante ist sie genau dann erste Kante der offenen Komponente mit höchstem Index, wenn sie ganz oben auf C liegt. Da es in der ungerichteten Tiefensuche keine Quer- oder Vorwärtskanten gibt, kann die zugehörige Komponente nicht mehr wachsen und ist daher fertig. Wegen der dritten Invariante werden durch die `repeat`-Schleife die richtigen Kanten vom Stack entfernt.

Da die Tiefensuche sicher in Linearzeit durchgeführt werden kann, ist nur zu zeigen, dass die Aufrufe von `traverse` und `backtrack` insgesamt linear viel Zeit benötigen. Dies folgt aber unmittelbar aus der Beobachtung, dass jede Kante nur höchstens einmal auf S_E und C abgelegt wird. \square

Zweifache Kantenzusammenhangskomponenten

1.30 Lemma

Zwei Knoten sind genau dann in einer zweifachen Kantenzusammenhangskomponente, wenn es einen ungerichteten Kreis gibt, der beide enthält. Jeder Knoten liegt in genau einer zweifachen Kantenzusammenhangskomponente.

Algorithmus 4: Zweifache Kantenzusammenhangskomponenten (Spezialisierung der ungerichteten Tiefensuche)

Daten : Stack S_V (Knoten in offenen Komponenten)
Stack C (offene Komponenten, repräsentiert durch ersten Knoten)

Ausgabe: Knotenarray `component` (zeigt auf Repräsentanten)

```

root(vertex s) begin
  | push s  $\rightarrow$   $S_V$ 
  | push s  $\rightarrow$   $C$ 
end

traverse(vertex v, edge e, vertex w) begin
  | if e ist Baumkante then
  |   | push w  $\rightarrow$   $S_V$ 
  |   | push w  $\rightarrow$   $C$ 
  | if e ist Rückwärtskante then
  |   | while  $w \prec \text{top}(C)$  do  $\text{pop}(C)$ 
  | end

backtrack(vertex w, edge e, vertex v) begin
  | if  $w = \text{top}(C)$  then
  |   |  $\text{pop}(C)$ 
  |   | repeat
  |   |   |  $u \leftarrow \text{pop}(S_V)$ 
  |   |   |  $\text{component}[u] \leftarrow w$ 
  |   |   until  $u = w$ 
  | end

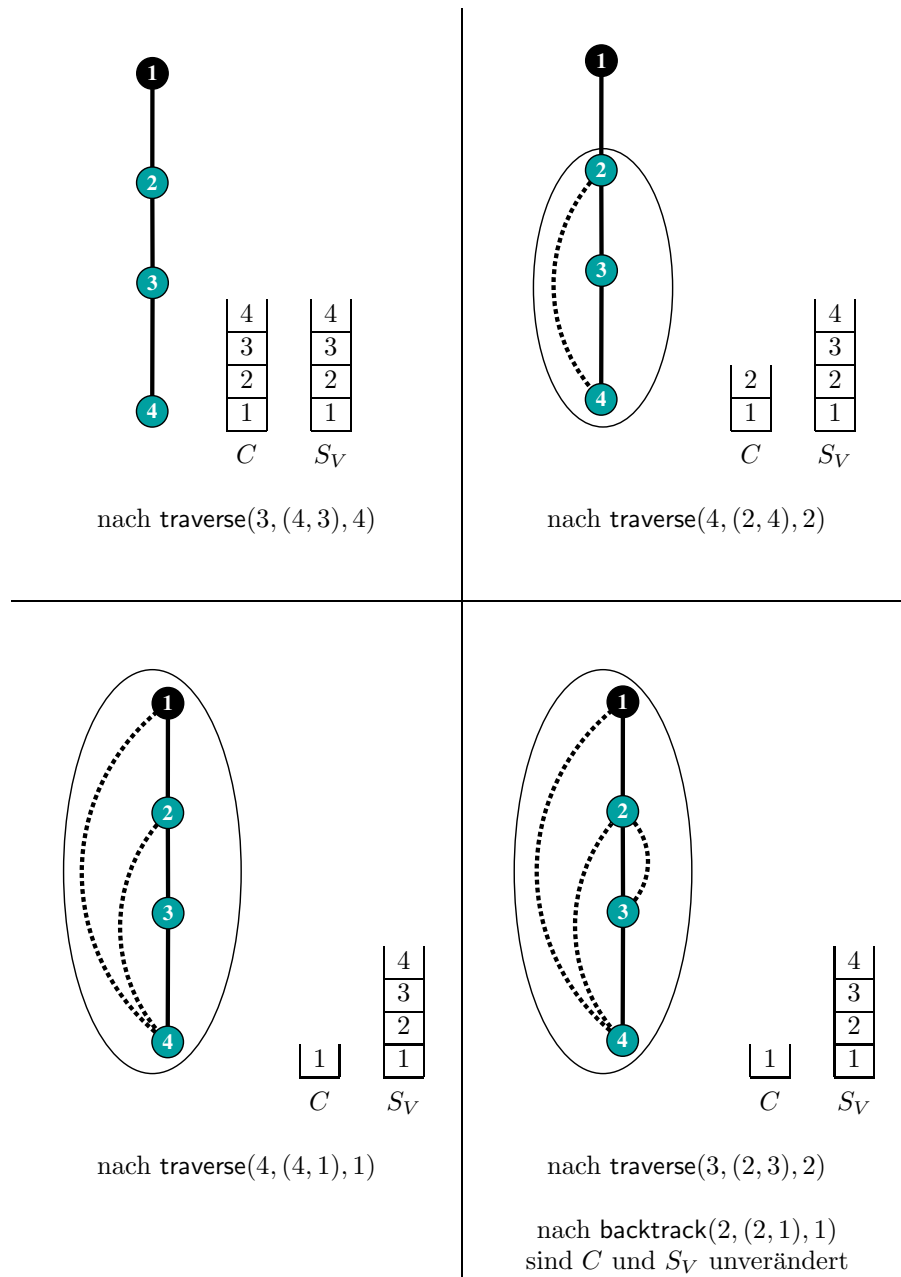
```

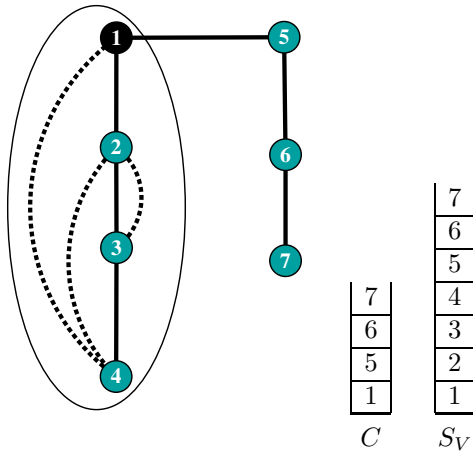
1.31 Satz

Algorithmus 4 berechnet die zweifachen Kantenzusammenhangskomponenten eines Multigraphen in Linearzeit.

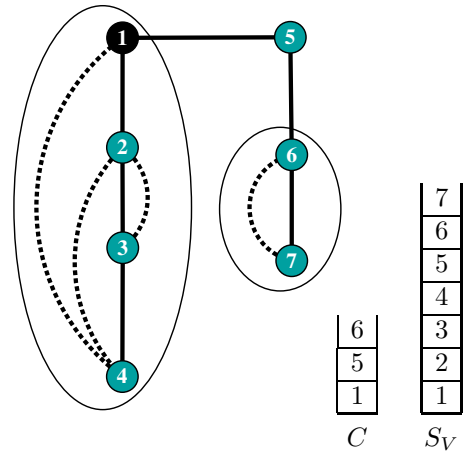
■ **Beweis:** Analog zu den zweifachen Zusammenhangskomponenten. \square

Mit der gleichen ungerichteten Tiefensuchreihenfolge wie oben ergeben sich die folgenden Zwischenschritte.

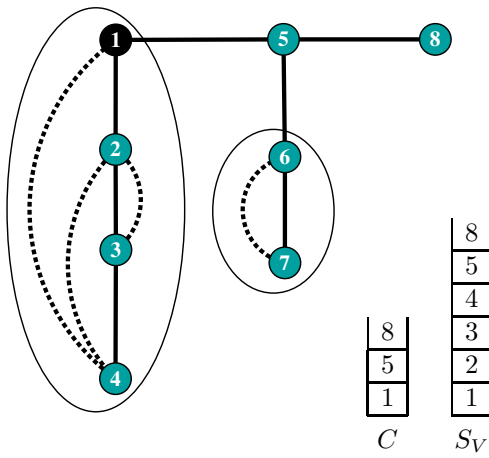




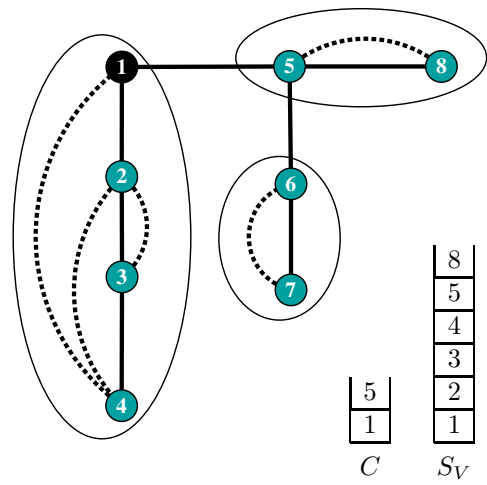
nach $\text{traverse}(6, (6, 7)_1, 7)$



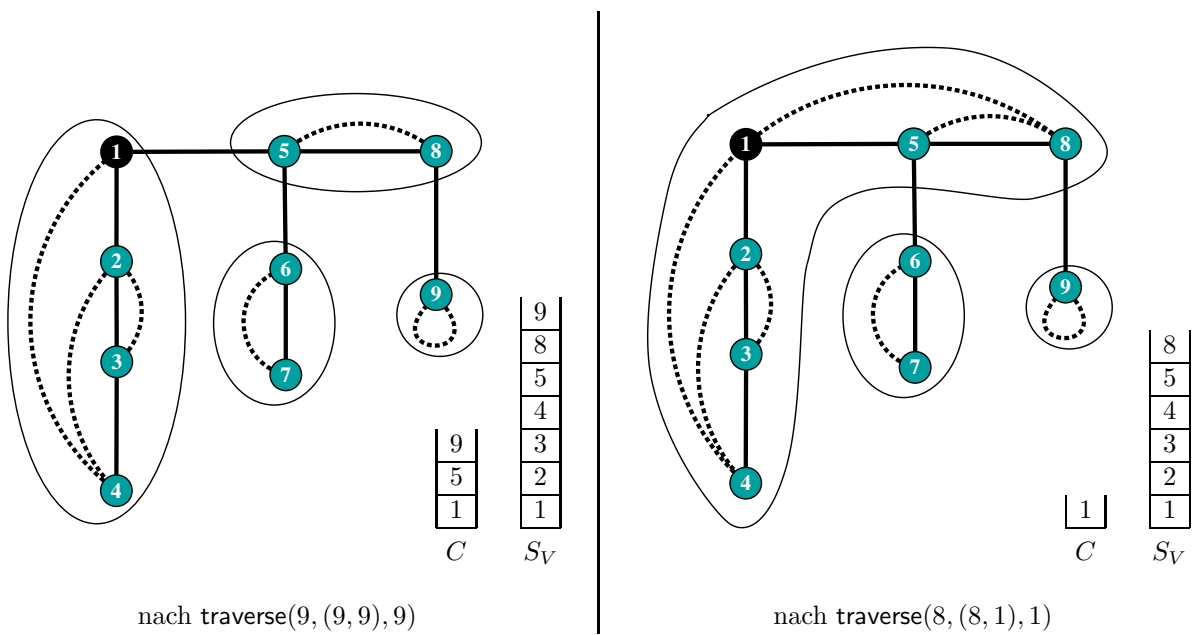
nach $\text{traverse}(7, (6, 7)_2, 6)$



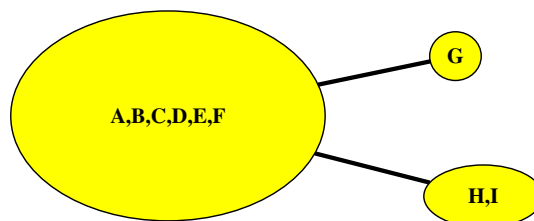
nach $\text{traverse}(5, (5, 8), 8)$



nach $\text{traverse}(8, (8, 5), 5)$



Eine kompakte Darstellung der zweifachen Kantenzusammenhangskomponenten eines Multigraphen ist der Baum (sic!) der brückenfreien Komponenten, dessen Knoten die kontrahierte Menge der Knoten einer Komponente und dessen Kanten die Brücken des Multigraphen (Kanten, deren Herausnahme ihn unzusammenhängend macht) sind.



Baum der brückenfreien Komponenten

Starke Zusammenhangskomponenten

1.32 Lemma

Zwei Knoten sind genau dann in einer starken Zusammenhangskomponente, wenn es eine geschlossene gerichtete Kantenfolge⁴ gibt, die beide enthält. Jeder Knoten liegt in genau einer starken Zusammenhangskomponente.

1.33 Satz

Verwendet man Algorithmus 4 als Spezialisierung einer gerichteten Tiefensuche und erweitert die eingekastete Bedingung zu

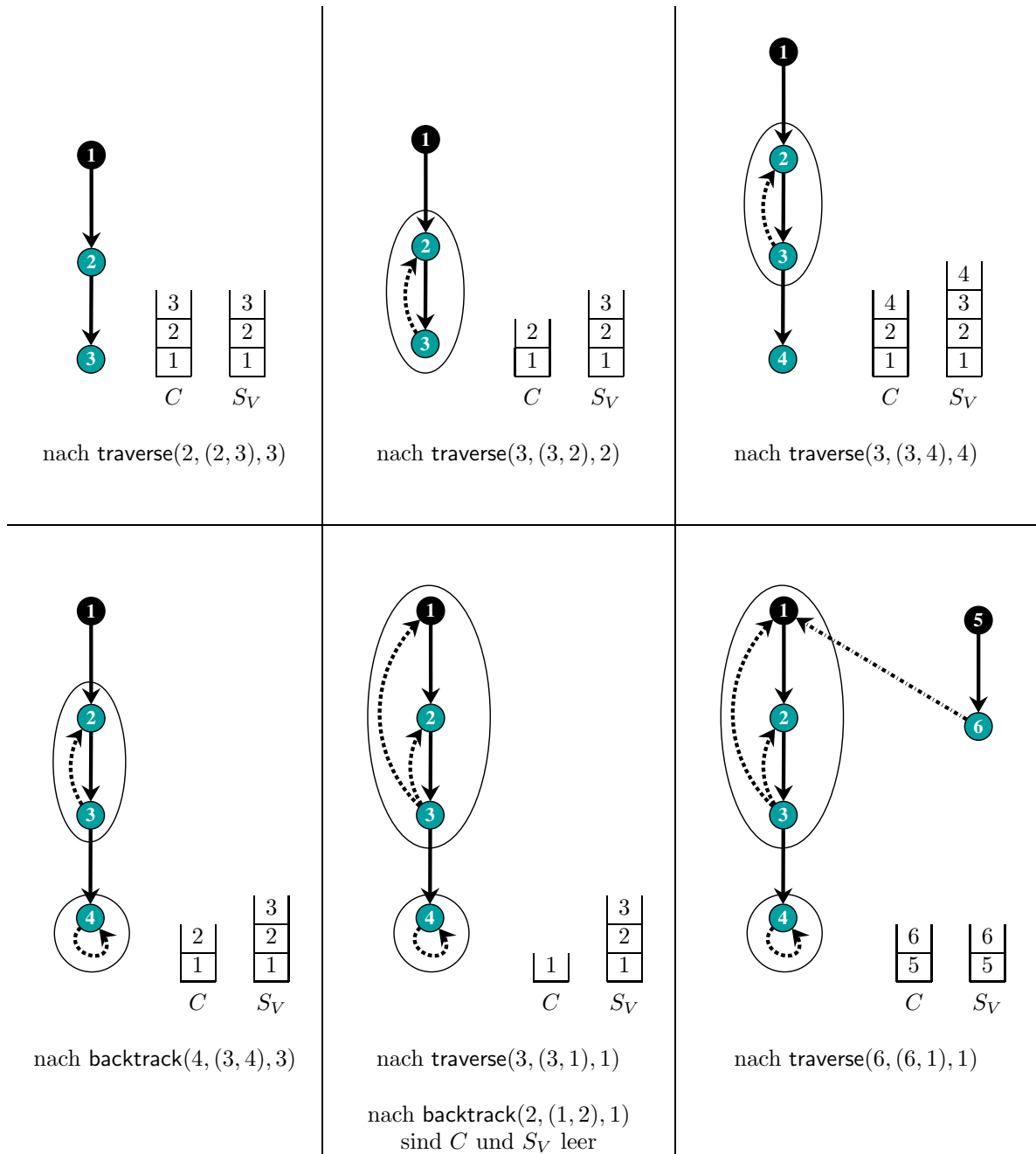
$$\boxed{e \text{ ist Rückwärtskante or } e \text{ ist Querkante mit } w \in S_V},$$

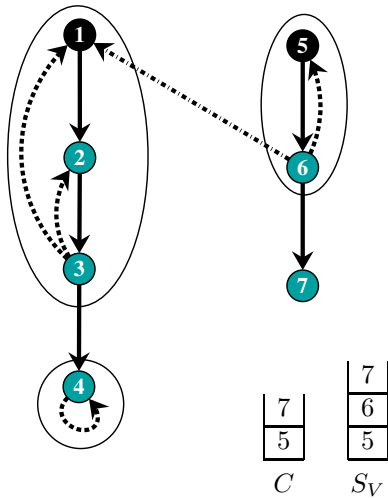
so berechnet der modifizierte Algorithmus die starken Zusammenhangskomponenten des eingegebenen Multigraphen in Linearzeit.

■ **Beweis:** Eine Querkante schließt genau dann einen Kreis, wenn die Komponente, in der sich der Zielknoten w befindet, noch offen ist. Vorwärtskanten sind lediglich Abkürzungen für bereits vorhandene gerichtete Wege. Alle anderen Argumente sind analog zum Fall der zweifachen Kantenzusammenhangskomponenten. \square

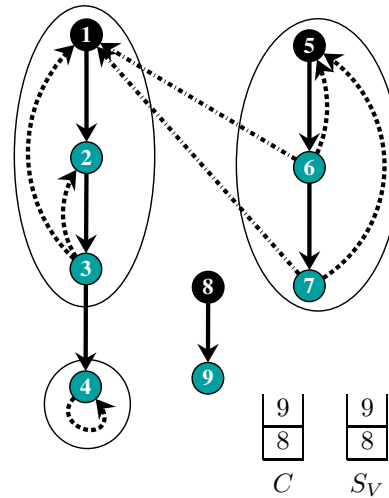
⁴D.h. ein Kreis, in dem Knoten wiederholt vorkommen dürfen.

Wie schon bei den beiden vorhergehenden Beispielen betrachten wir wieder die Zwischenschritte des Algorithmus, diesmal aber natürlich bezüglich der gerichteten Tiefensuche.

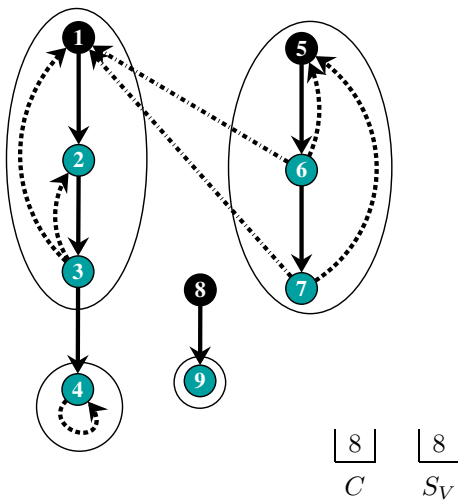




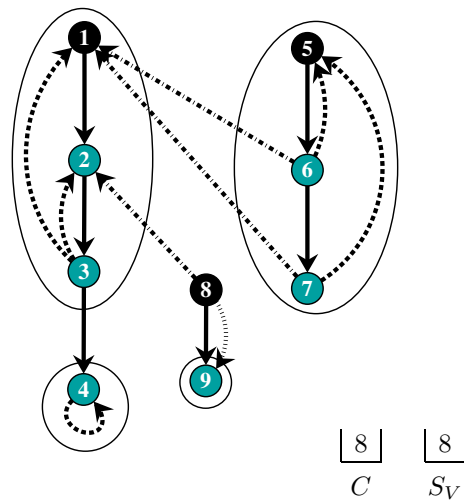
nach $\text{traverse}(6, (6, 7), 7)$



nach $\text{traverse}(8, (8, 9), 9)$



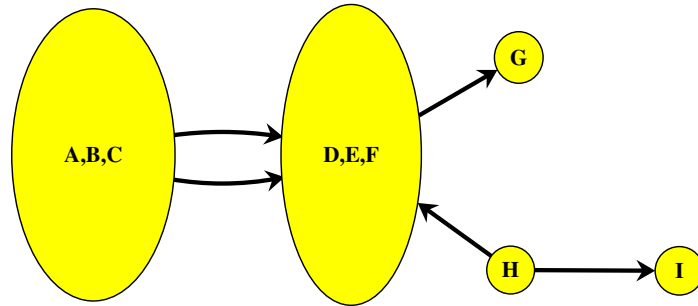
nach $\text{backtrack}(9, (8, 9), 8)$



nach $\text{traverse}(8, (8, 2), 2)$

Eine kompakte Darstellung der starken Zusammenhangskomponenten eines Multigraphen ist der gerichtete kreisfreie Multigraph (*directed acyclic graph, DAG*), dessen Knoten die kontrahierten Komponenten und dessen Kanten die

dabei nicht übrigbleibenden Kanten sind (genau die Kanten aller gerichteten Schnitte des Multigraphen).



DAG der starken Zusammenhangskomponenten