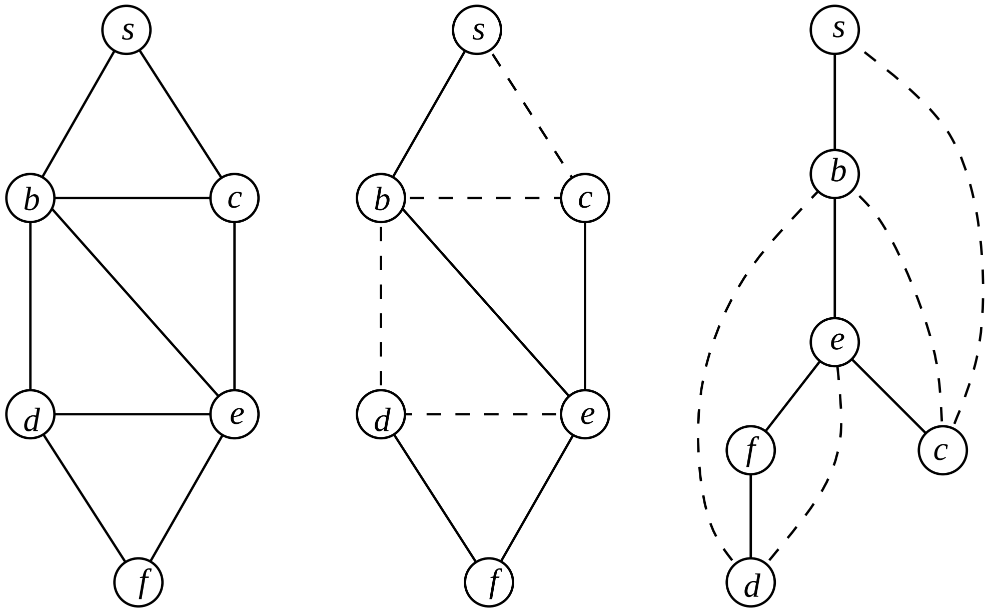
1 DFS-Bäume in ungerichteten Graphen

Sei ein ungerichteter, zusammenhängender Graph $G = (V, E)$ gegeben. Sei ferner ein Startknoten $s \in V$ ausgewählt. Startet man bei s eine Tiefensuche (DFS), so besucht diese DFS alle Knoten von G. Jede Kante $e \in E$ spielt bei der DFS eine von zwei möglichen Rollen:

1. Über die Kante e wird von einem Knoten v aus ein unbesuchter Nachbar w entdeckt; in diesem Fall nennen wir e eine *Baumkante*, die als *abwärts gerichtet* von v nach w aufgefasst wird.

2. Wenn die DFS die Kante e betrachtet, ist der Knoten am anderen Ende bereits besucht worden; in diesem Fall nennen wir e eine *Rückwärtskante*.

Ein Beispiel soll das verdeutlichen:



(a) (b) (c)

Das Bild zeigt (a) einen ungerichteten Graphen, (b) die Einteilung der Kanten in Baumkanten (durchgezogen) und Rückwärtskanten (gestrichelt) durch eine DFS mit Startknoten s, und (c) eine gewurzelte Darstellung des sich daraus ergebenden DFS-Baumes.

2 Zweifachzusammenhangskomponenten

Im folgenden sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph.

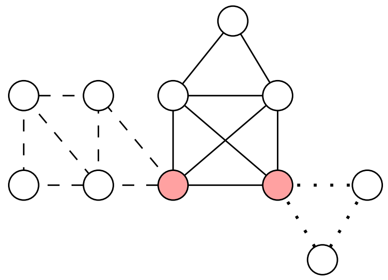
Definition 1 (Zusammenhang) *Der (Knoten-)Zusammenhang von G ist die minimale Anzahl von Knoten, deren Entfernung G in zwei oder mehr Zusammenhangskomponenten zerfallen lässt.*

Ein Graph mit Zusammenhang mindestens 2 heißt auch *zweifachzusammenhängend*.

Definition 2 (Artikulationsknoten) *Ein Knoten a von G heißt Artikulationsknoten, wenn G durch Entfernung von a in zwei oder mehr Zusammenhangskomponenten zerfällt.*

Offensichtlich ist G genau dann zweifachzusammenhängend, wenn er keinen Artikulationsknoten enthält.

Die *Zweifachzusammenhangskomponenten* (auch *Blöcke* oder engl. *biconnected components*) von G sind die maximalen Teilgraphen von G, die zweifachzusammenhängend sind. Der nebenstehende Graph besteht aus drei Blöcken, die durch durchgezogene, gestrichelte und gepunktete Kanten gekennzeichnet sind. Die Artikulationsknoten sind schattiert gezeichnet.



Es gilt übrigens: Im Schnitt zweier Blöcke liegt höchstens ein einziger Knoten. Blöcke können also keine Kante gemeinsam haben. Die Knoten, die im Schnitt von Blöcken liegen, sind genau die Artikulationsknoten. Die Blöcke bilden eine Baumstruktur, d.h. die Struktur, die entsteht, wenn man Blöcke mit nicht-leerem Schnitt als benachbart betrachtet, enthält keine Kreise.

Der Zusammenhang von Graphen und die Berechnung der Blöcke ist zum Beispiel beim Aufbau ausfallsicherer Kommunikationsnetzwerke interessant. Ein zweifachzusammenhängendes Teilnetzwerk bleibt auch dann zusammenhängend, wenn ein beliebiger Knoten ausfällt.

2.1 Erweiterter DFS-Algorithmus

Mittels einer nur geringfügig modifizierten Tiefensuche in G können für jeden Knoten v die folgenden Werte ermittelt werden:

- $pre[v]$ = Präordernummer (DFS-Nummer) von v, d.h. Nummer, die angibt, als wie vielter Knoten der Knoten v von der DFS besucht wurde ($pre[s] = 0$).
- $low[v]$ = minimale Präordernummer $pre[w]$ eines Knotens w, der von v aus über ≥ 0 Baumkanten abwärts, evtl. gefolgt von einer einzigen Rückwärtskante, erreicht werden kann.

Die Berechnung der *low*-Werte geschieht einfach durch Ausnutzung der folgenden Tatsache:

$$\begin{aligned} low[v] \quad = \min \Big(\quad & \{pre[v]\} \cup \\ & \{low[w] \mid w \text{ ist Kind von } v \text{ im DFS-Baum } \} \cup \\ & \{pre[w] \mid \{v, w\} \text{ ist Rückwärtskante } \} \Big) \end{aligned}$$

Genauer wird am Anfang der rekursiven DFS-Funktion für einen Knoten v der Wert $pre[v]$ bestimmt und $low[v]$ mit $pre[v]$ initialisiert; dann werden alle Nachbarkanten von v betrachtet: bei einer abwärtsgerichteten Baumkante zu einem Kind w erfolgt ein rekursiver Aufruf und anschließend wird $low[v] = \min\{low[v], low[w]\}$ gesetzt; bei einer Rückwärtskante zu einem Knoten w setzen wir dagegen $low[v] = \min\{low[v], pre[w]\}$.

Nun können wir folgendes Lemma verwenden.

Lemma 3 *Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph und T ein DFS-Baum in G. Ein Knoten $a \in V$ ist Artikulationsknoten genau dann, wenn entweder (a) a die Wurzel von T ist und ≥ 2 Kinder hat, oder (b) a nicht die Wurzel von T ist und es ein Kind b von a mit $low[b] \geq pre[a]$ gibt.*

Somit lassen sich die Artikulationsknoten in G mittels leicht modifizierter DFS effizient in Zeit $O(|V| + |E|)$ ermitteln. Um die Blöcke selbst mit demselben Zeitaufwand zu bestimmen, kann man so vorgehen: Es wird ein Stack S von Kanten verwaltet, der anfangs leer ist. Wenn eine Baumkante abwärts gefunden wird, so wird sie vor dem rekursiven Aufruf auf S abgelegt; wenn eine Rückwärtskante gefunden wird (von dem unteren Endknoten aus), so wird sie sofort auf dem Stack abgelegt. Immer, wenn ein rekursiver Aufruf für Knoten w zu Knoten v zurückkehrt und $low[w] \geq pre[v]$ gilt, bilden die Kanten oben auf dem Stack bis einschließlich der Kante $\{v, w\}$ genau die Kanten des nächsten Blockes.

3 DFS-Wälder in gerichteten Graphen

Sei ein gerichteter Graph $G = (V, E)$ gegeben. Wenn man bei einem Knoten in G eine Tiefensuche startet, so werden nicht unbedingt alle Knoten des Graphen erreicht, und es muss eine weitere Tiefensuche bei einem unbesuchten Knoten gestartet werden. Deshalb ergibt eine DFS in einem gerichteten Graphen nicht einen einzigen DFS-Baum, sondern i.a. einen DFS-Wald, der aus mehreren Bäumen besteht. Jede gerichtete Kante $e = (v, w) \in E$ wird untersucht, wenn ihr Startknoten v bearbeitet wird, und spielt bei der DFS genau eine von vier möglichen Rollen:

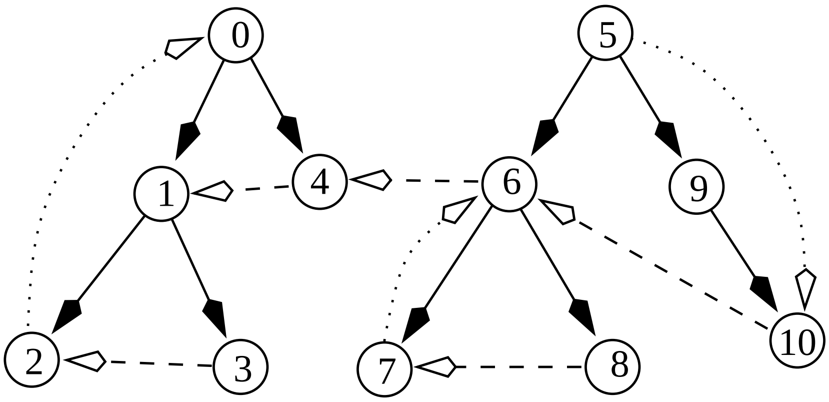
1. Über die Kante e wird ein unbesuchter Nachbar w entdeckt; in diesem Fall nennen wir e eine *Baumkante*, die als *abwärts gerichtet* von v nach w aufgefasst wird.

2. Der Knoten w ist bereits besucht worden und hat eine DFS-Nummer, die größer ist als die DFS-Nummer von v; in diesem Fall nennen wir e eine *Vorwärtskante*.

3. Der Knoten w ist bereits besucht worden und ist ein Vorfahre (Vater, Großvater, usw.) von v im selben DFS-Baum; in diesem Fall nennen wir e eine *Rückwärtskante*.

4. Der Knoten w ist bereits besucht worden und ist kein Vorfahre (Vater, Großvater, usw.) von v im selben DFS-Baum; in diesem Fall kann w im selben DFS-Baum oder in einem bereits früher erzeugten DFS-Baum enthalten sein, und wir nennen e eine *Querkante*.

Ein Beispiel soll das verdeutlichen:



Das Bild zeigt einen möglichen DFS-Wald, der sich bei der Tiefensuche in einem gerichteten Graphen ergeben kann. Baumkanten sind durchgezogen gezeichnet, Querkanten gestrichelt, Vorwärts- und Rückwärtskanten gepunktet. In den Knoten sind die DFS-Nummern (Präordernummern) angegeben.

4 Starke Zusammenhangskomponenten

Sei ein gerichteter Graph $G = (V, E)$ gegeben. Zwei Knoten x und y von G liegen in derselben *starken Zusammenhangskomponente*, falls es in G einen gerichteten Pfad von x nach y und von y nach x gibt. Dadurch lässt sich die Knotenmenge von G vollständig in starke Zusammenhangskomponenten einteilen (partitionieren). Der obige Beispielgraph besteht zum Beispiel aus den fünf starken Zusammenhangskomponenten $\{0, 1, 2, 3, 4\}$, $\{6, 7, 8\}$, $\{10\}$, $\{9\}$ und $\{5\}$. Wir wollen für einen gegebenen Graphen $G = (V, E)$ in Zeit $O(|V|+|E|)$ die starken Zusammenhangskomponenten berechnen. Dies kann wieder mittels modifizierter Tiefensuche erfolgen.

Wir betrachten den DFS-Wald und die Präordernummern, die sich durch Tiefensuche in G ergeben. Für jede starke Zusammenhangskomponente $Z \subseteq V$ nennen wir denjenigen Knoten in Z mit der kleinsten Präordernummer die *Wurzel* der starken Zusammenhangskomponente. Es lässt sich zeigen, dass ein Knoten r genau dann die Wurzel einer starken Zusammenhangskomponente ist, wenn es:

1. keine Rückwärtskante von einem Nachfahren (dem Knoten selbst oder einem Kind, Enkel, ...) von r im DFS-Wald zu einem echten Vorfahren (Vater, Großvater, ...) gibt

2. keine Querkante von r oder einem Nachfahren von r zu einem Knoten w gibt, so dass die Wurzel der starken Zusammenhangskomponente von w ein echter Vorfahre von r ist

(der Knoten 4 in obigem Beispiel hat so eine Querkante und ist daher keine Wurzel einer starken Zusammenhangskomponente). Die starken Zusammenhangskomponenten lassen sich dann aus dem DFS-Wald “abpflücken”, indem man einfach bottom-up vorgeht und an allen Wurzeln den noch daranhängenden Teilbaum aus dem Wald entfernt (im

Beispiel also nacheinander die Teilbäume, die an den Knoten 0, 6, 10, 9 und 5 hängen). Dies lässt sich alles im Rahmen einer einzigen Tiefensuche erledigen.

Wir definieren zusätzlich zur üblichen Präordernummer $pre[v]$ eines Knotens v noch einen Wert $lowlink[v]$ wie folgt:

- $lowlink[v]$ = minimale Präordernummer $pre[w]$ eines Knotens w, der über ≥ 0 Baumkanten abwärts, evtl. gefolgt von einer einzigen Rückwärtskante erreicht werden kann, oder der über ≥ 0 Baumkanten abwärts gefolgt von einer einzigen Querkante zu einem Knoten in einer Zusammenhangskomponente, deren Wurzel echter Vorfahre von v ist, erreicht werden kann.

Die *lowlink*-Werte können während einer geringfügig modifizierten DFS unter Ausnutzung folgender Gleichung berechnet werden:

$$\begin{aligned} lowlink[v] \;&= \min \Big(\; \{ \, pre[v] \, \} \cup \{ \, lowlink[w] \mid w \text{ ist Kind von } v \text{ im DFS-Wald } \} \\ &\qquad \cup \; \{ \, pre[w] \mid (v,w) \text{ ist Rückwärtskante } \} \\ &\qquad \cup \; \{ \, pre[w] \mid (v,w) \text{ ist Querkante und die Wurzel der starken} \\ &\qquad\qquad \text{Zusammenhangskomponente von } w \text{ ist Vorfahre von } v \, \} \Big) \end{aligned}$$

Genauer wird am Anfang der rekursiven DFS-Funktion für einen Knoten v der Wert $pre[v]$ bestimmt und $lowlink[v]$ mit $pre[v]$ initialisiert; dann werden alle Nachbarkanten von v betrachtet: bei einer abwärtsgerichteten Baumkante zu einem Kind w erfolgt ein rekursiver Aufruf und anschließend wird $lowlink[v] = \min\{lowlink[v], lowlink[w]\}$ gesetzt. Bei einer Rückwärtskante wird dagegen $lowlink[v] = \min\{lowlink[v], pre[w]\}$ gesetzt. Bei einer Querkante zu einem Knoten w muss entschieden werden, ob die Wurzel der starken Zusammenhangskomponente von w ein Vorfahre von v ist oder nicht; das ist genau dann der Fall, wenn w vom Algorithmus noch nicht einer starken Zusammenhangskomponente zugeordnet ("abgepflückt") wurde, und in diesem Fall setzen wir wieder $lowlink[v] = \min\{lowlink[v], pre[w]\}$ (ansonsten wird die Querkante ignoriert).

Wir verwenden die Tatsache, dass ein Knoten r genau dann Wurzel einer starken Zusammenhangskomponente ist, wenn nach Ausführung der DFS-Funktion für r die Bedingung $lowlink[r] = pre[r]$ gilt. Somit ist es für den Algorithmus leicht, die Wurzeln der starken Zusammenhangskomponenten zu erkennen. Die starken Zusammenhangskomponenten selbst können nebenbei während der Tiefensuche bestimmt werden, wenn man folgendermaßen vorgeht. Es wird ein Stack S von Knoten verwaltet, der anfangs leer ist. Wenn die rekursive DFS-Funktion für einen Knoten v aufgerufen wird, so wird am Anfang der Ausführung der Funktion der Knoten v auf den Stack S gelegt. Wenn am Ende der Ausführung der Funktion für Knoten v die Bedingung $lowlink[v] = pre[v]$ gilt, dann bilden alle Knoten oben auf dem Stack bis einschließlich v die nächste starke Zusammenhangskomponente Z. Alle Knoten in Z müssen als "bereits einer starken Zusammenhangskomponente zugeordnet" markiert werden, damit Querkanten zu diesen Knoten, die später gefunden werden, bei der Berechnung von *lowlink*-Werten nicht mehr berücksichtigt werden.

Bemerkung: Man sollte sich bewusst machen, dass es gelungen ist, die Berechnung von Zweifachzusammenhangskomponenten in ungerichteten Graphen und von starken Zusammenhangskomponenten in gerichteten Graphen durch eine nur geringfügig modifizierte Tiefensuche in linearer Zeit $O(|V|+|E|)$ zu erledigen. Auf den ersten Blick hätte man vielleicht vermutet, dass dazu Algorithmen mit erheblich größerem Laufzeitbedarf nötig sind!