6 Tiefensuche in ungerichteten Graphen: Zweifache Zusammenhangskomponenten

Der Algorithmus ist ganz genau derselbe wie im gerichteten Fall. Abbildung 1 zeigt noch einmal den gerichteten Fall und danach betrachten wir in Abbildung 2 den ungerichteten Fall auf dem analogen Graphen (Kanten „→—“ sind Baumkanten, über die entdeckt wird.)

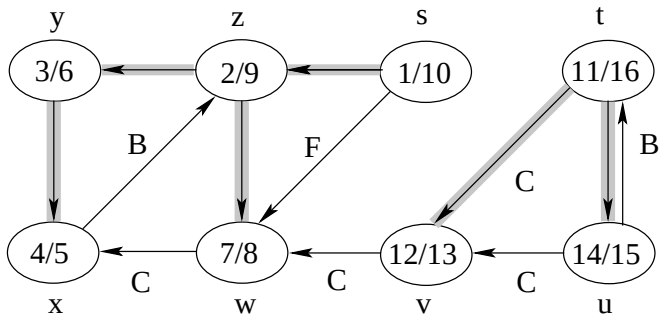


Abbildung 1: Das Ergebnis der Tiefensuche auf einem gerichteten Graphen

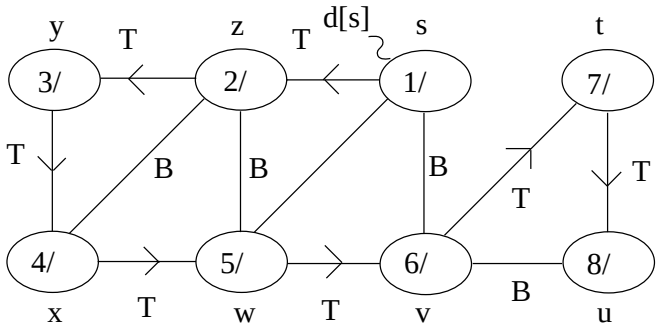
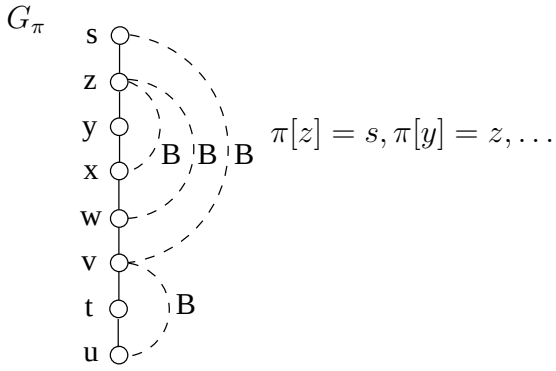


Abbildung 2: vorhergender Graph, ungerichtete Variante

Im ungerichteten Graphen gilt: beim ersten Gang durch eine Kante stößt man

- auf einen weißen Knoten (Kante: T)
- auf einen grauen Knoten (Kante: B)
- nicht auf einen schwarzen Knoten (Kante: F oder C)



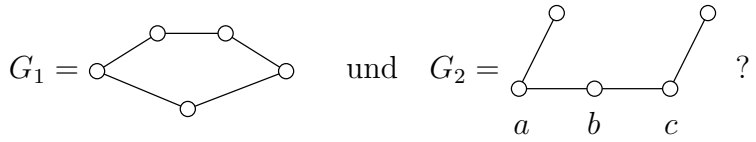
Im ungerichteten Fall der Tiefensuche ist festzuhalten:

- Jede Kante wird zweimal betrachtet: $\{u, v\}$ bei DFS-visit(u) und bei DFS-visit(v).
- Maßgeblich für den Kantentyp (Baumkante, Rückwärts-, Vorwärts-, Kreuzkante) ist die Betrachtung:
  - (i) $\{u, v\}$ Baumkante $\Longleftrightarrow$ Beim ersten Betrachten von $\{u, v\}$ findet sich ein weißer Knoten.
  - (ii) $\{u, v\}$ Rückwärtskante $\Longleftrightarrow$ Beim ersten Gehen von $\{u, v\}$ findet sich ein bereits grauer Knoten
  - (iii) Beim ersten Gehen kann sich kein schwarzer Knoten finden. Deshalb gibt es weder Kreuz- noch Vorwärtskanten.

Der Weiße-Weg-Satz gilt hier vollkommen analog. Kreise erkennen ist ganz ebenfalls analog mit Tiefensuche möglich.

Der Begriff der *starken Zusammenhangskomponente* ist nicht sinnvoll, da Weg (u, v) im ungerichteten Fall ebenso ein Weg (v, u) ist.

Stattdessen: *zweifach zusammenhängend*. Unterschied zwischen



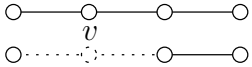
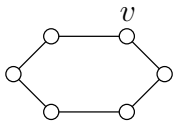
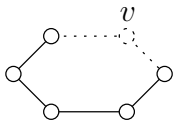
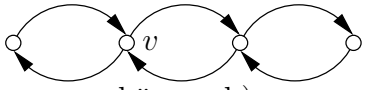
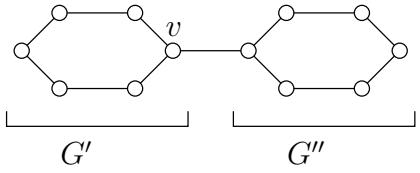
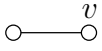
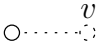
Löschen wir in G_1 einen beliebigen Knoten, hängt der Rest noch zusammen. Für a, b, c im Graphen G_2 gilt dies nicht.

Für den Rest dieses Kapitels gilt nun folgende *Konvention*:
Ab jetzt gehen wir nur immer von zusammenhängenden, gerichteten Graphen aus.

Definition 6.1(zweifach zusammenhängend): *$G = (V, E)$ ist zweifach zusammenhängend $\Longleftrightarrow$ $G \setminus \{v\}$ ist zusammenhängend für alle $v \in V$.*

$G \setminus \{v\}$ bedeutet $V \setminus \{v\}, E \setminus \{\{u, v\} \mid u \in V\}$.

Beispiel 6.1:

G:  $G \setminus \{v\}$: 	G zusammenhängend, nicht zweifach zusammenhängend.
G:  $G \setminus \{v\}$: 	(Beachte:  ist stark zusammenhängend.) Knoten v und adjazente Kanten fehlen. G ist zweifach zusammenhängend.
G: 	G ist nicht zweifach zusammenhängend, da $G \setminus \{v\}$ nicht zusammenhängend ist. Aber G' und G'' sind jeweils zweifach zusammenhängend.
G:  $G \setminus \{v\}$: 	G ist also zweifach zusammenhängend.

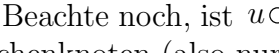
Was ist die Bedeutung zweifach zusammenhängend?

Satz 6.1(Menger 1927): *G zweifach zusammenh. $\iff$ Für alle $u, v \in V, u \neq v$ gibt es zwei disjunkte Wege zwischen u und v in G.*

Das heißt, Wege $(u, u_1, u2, \ldots, u_m, v)$, $(u, u'_1, u'_2, \ldots, u'_m, v)$ mit

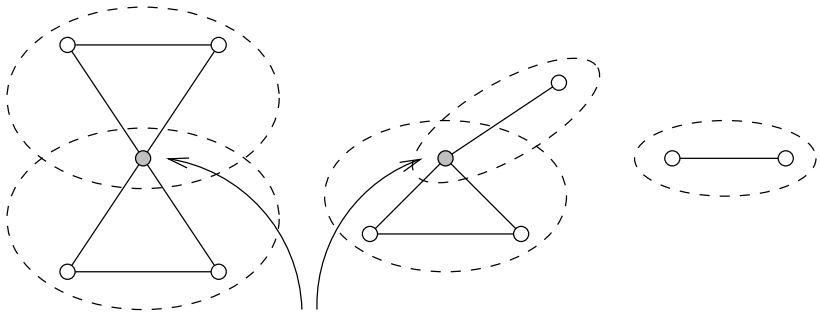
$$\{u_1, \ldots, u_m\} \cap \{u'_1, \ldots, u'_m\} = \emptyset.$$

Beweis erfolgt auf überraschende Weise später.

Ein induktiver Beweis im Buch Graphentheorie von Reinhard Diestel. Die Richtung „$\Leftarrow$“ ist einfach. Beachte noch, ist u  v, so tut es der Weg (u, v), mit leerer Menge von Zwischenknoten (also nur 1 Weg).

Nun gilt es wiederum nicht zweifach zusammenhängende Graphen in ihre zweifach zusammenhängenden Bestandteile zu zerlegen, in die zweifachen Zusammenhangskomponenten (auch einfach zweifache Komponenten genannt).

Beispiel 6.2:



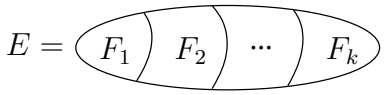
Knoten in zwei zweifachen Zusammenhangskomponenten

Definition 6.2(zweifache Komponenten): *Ein Teilgraph $H = (W, F)$ von G ist eine zweifache Komponente $\Longleftrightarrow$ H ist ein maximaler zweifach zusammenhängender Teilgraph von G (maximal bezüglich Knoten und Kanten).*

Bemerkung:

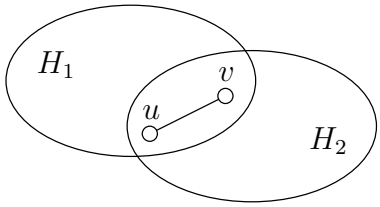
(a) Jede Kante ist in genau einer zweifachen Komponente.

(b) Sind $H_1 = (W_1, F_1), \ldots, H_k = (W_k, F_k)$ die zweifachen Komponeten von $G = (V, E)$, so ist $F_1, \ldots, F_k$ eine Partition (Einteilung) von E.



$F_i \cap F_j = \emptyset$ für $i \neq j$, $\quad F_1 \cup \cdots \cup F_k = E$.

Beweis. (a) Wir betrachten die Teilgraphen H_1 und H_2 sowie eine Kante $\{u, v\}$, die sowohl in H_1 als auch in H_2 liegt.



Seien also $H_1 \neq H_2$ zweifach zusammenhängend, dann ist $H = H_1 \cup H_2$ zweifach zusammenhängend:

Für alle w aus H_1, $w \neq u, w \neq v$ ist $H \setminus \{w\}$ zusammenhängend (wegen Maximalität). Ebenso für alle w aus H_2.

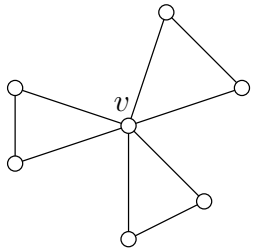
Für $w = u$ ist, da immer noch v da ist, $H \setminus \{w\}$ zweifach zusammenhängend. Also wegen Maximalität zweifache Komponenten $\supseteq H_1 \cup H_2$.

(b) $F_i \cap F_j = \emptyset$ wegen (a). Da Kante (u, v) zweifach zusammenhängend ist, ist nach Definition jede Kante von E in einem F_i.

□

Bezeichnung: Eine Kante, die eine zweifache Komponente ist, heißt *Brückenkante*.

Bei Knoten gilt (a) oben nicht:



v gehört zu 3 zweifachen Komponenten. v ist ein typischer Artikulationspunkt.

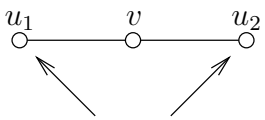
Definition 6.3(Artikulationspunkt): *v ist Artikulationspunkt von $G \Longleftrightarrow G \setminus \{v\}$* nicht *zusammenhängend.*

Bemerkung 6.3:

v ist Artikulationspunkt $\Longleftrightarrow$ v gehört zu ≥ 2 zweifachen Komponenten.

Beweis. „$\Rightarrow$“ Ist v Artikulationspunkt. Dann gibt es Knoten $u, w, u \neq v, w \neq v$, so dass jeder Weg von u nach w von der Art $(u, \ldots, v, \ldots, w)$ ist. (Sonst $G \setminus \{v\}$ zusammenhängend)

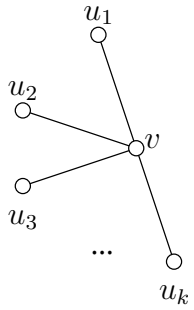
Also sind u, w nicht in einer zweifachen Komponente. Dann ist jeder Weg von der Art $(u, \ldots, u_1, v, u_2, \ldots, w)$, so dass u_1, u_2 nicht in einer zweifachen Komponente sind. Sonst ist in $G \setminus \{v\}$ ein Weg $(u, \ldots, u_1, u_2, \ldots w)$ (ohne v), Widerspruch. Also haben wir:



verschiedene Komponenten

Die Kante $\{u_1, v\}$ gehört zu einer Komponente (eventuell ist sie selber eine), ebenso $\{v, u_2\}$. Die Komponenten der Kanten sind verschieden, da u_1, u_2 in verschiedenen Komponenten liegen. Also v liegt in den beiden Komponenten.

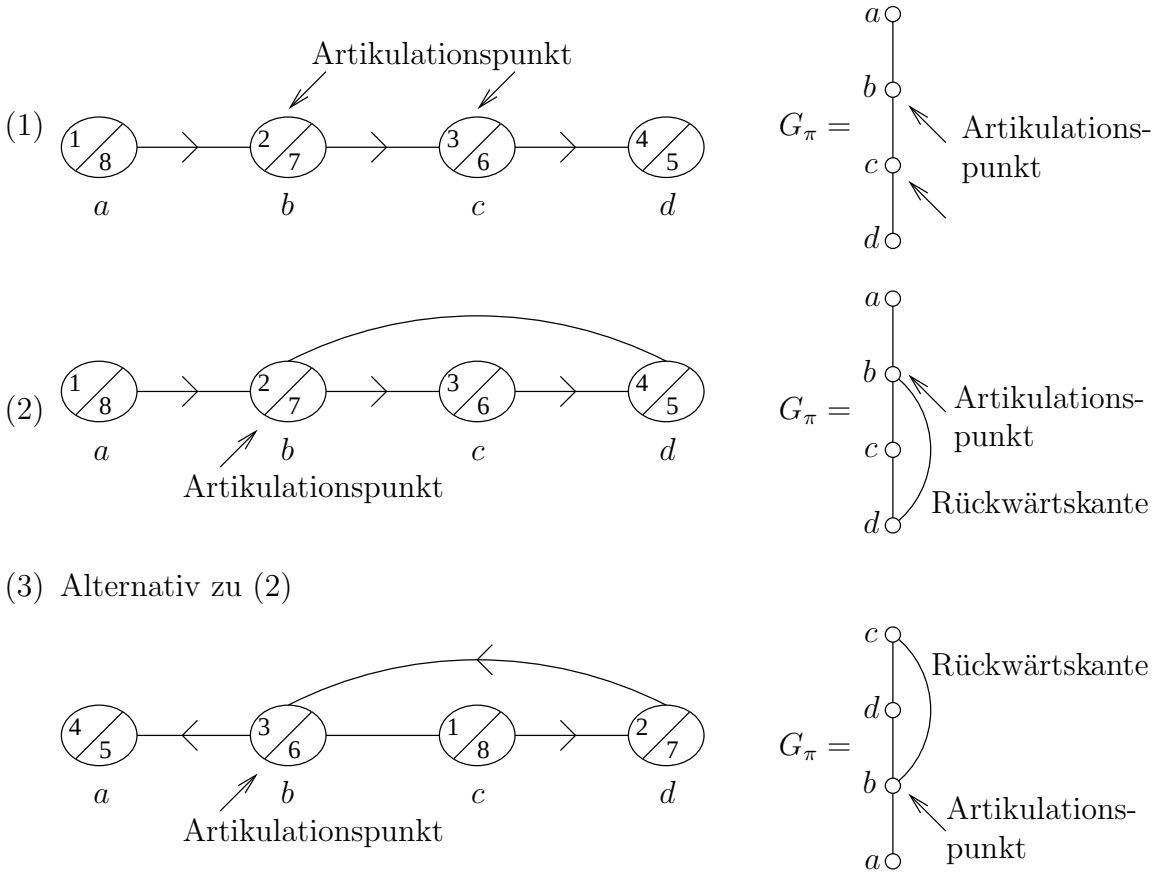
„$\Leftarrow$“ Gehört also v zu > 2 Komponenten. Dann



und ≥ 2 Kanten von $\{u_i, v\}$ liegen in verschiedenen Komponenten. Etwa u_1, u_2. Aber u_1 und u_2 sind in zwei verschiedenen Komponenten. Also gibt es w, so dass in $G \setminus \{w\}$ kein Weg $u_1 \circ\!\!-\!\!\circ u_2$ existiert. Denn wegen $u_1 \circ\!\!-\!\!\overset{v}{\circ}\!\!-\!\!\circ u_2$ muss $w = v$ sein und v ist Artikulationspunkt. □

Graphentheoretische Beweise sind nicht ganz so leicht!

Artikulationspunkte und Tiefensuche?



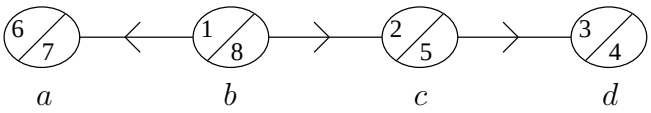
Was haben die Artikulationspunkte in allen genannten Fällen gemeinsam? Der Artikulationspunkt (sofern nicht Wurzel von G_π) hat einen Sohn in G_π, so dass es von dem Sohn oder Nachfolger keine Rückwärtskante zum echten Vorgänger gibt!

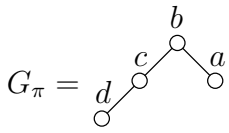
(1) Gilt bei b und c. d ist kein Artikulationspunkt und das Kriterium gilt auch nicht.

(2) b ist Artikulationspunkt, Sohn c erfüllt das Kriterium. c, d erfüllen es nicht, sind auch keine Artikulationspunkte.

(3) Hier zeigt der Sohn a von b an, dass b ein Artikulationspunkt ist. Also keineswegs immer derselbe Sohn!

Was ist, wenn der Artikulationspunkt Wurzel von G_π ist?



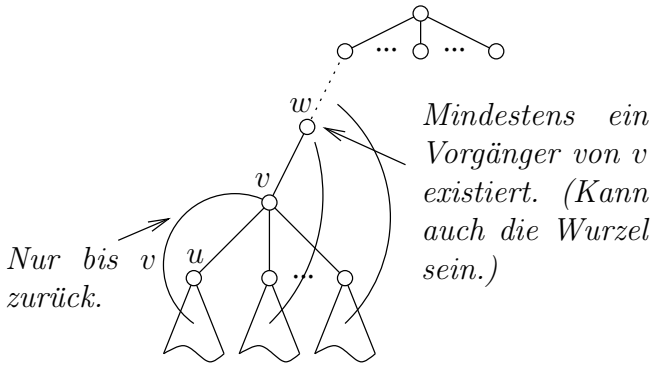


Zwei Söhne.

Satz 6.2(Artikulationspunkte erkennen): *Sei v ein Knoten von G und sei eine Tiefensuche gelaufen. (Erinnerung: G immer zusammenhängend).*

a) Ist v Wurzel von G_π. Dann ist v Arikulationspunkt $\Longleftrightarrow$ v in G_π  *hat ≥ 2 Söhne.*

b) Ist v nicht Wurzel von G_π. Dann ist v Artikulationspunkt $\Longleftrightarrow$ G_π folgendermaßen:



Das heißt, v hat einen Sohn (hier u), so dass von dem und allen Nachfolgern in G_π keine (Rückwärts-)Kanten zu echtem Vorgänger von v existieren.

Beweis.

a) „$\Leftarrow$“ $G \setminus \{v\}$ nicht zusammenhängend, damit v Artikulationspunkt.

„$\Rightarrow$“ Hat v nur einen Sohn, dann $G_\pi =$

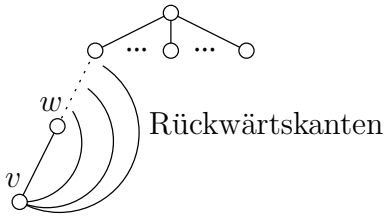
Rückwärts-kanten 

Zeige $A \Rightarrow B$ durch $\neg B \Rightarrow \neg A$.

Dann ist $G \setminus \{v\}$ zusammenhängend. (Können in $G \setminus \{v\}$ über w statt v gehen.) Also ist v kein Artikulationspunkt. Ist v ohne Sohn, dann ist er kein Artikulationspunkt.

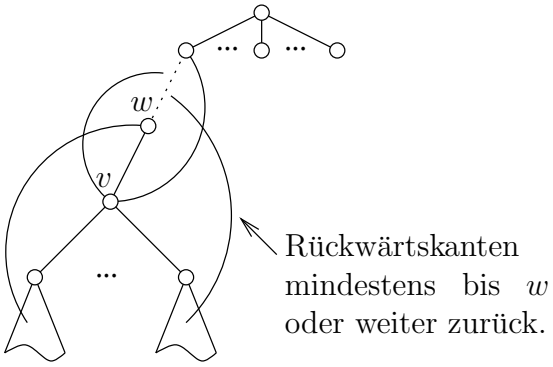
b) „$\Leftarrow$“ In $G \setminus \{v\}$ kein Weg $u \circ\!\sim\!\circ w$, also ist v Artikulationspunkt.
„$\Rightarrow$“ Gelte die Behauptung nicht, also v hat keinen Sohn wie u in G_π.

1. Fall v hat keinen Sohn. Dann in G_π



In $G \setminus \{v\}$ fehlen die Rückwärtskanten und $\{w, v\}$. Also bleibt der Rest zusammenhängend. v ist kein Artikulationspunkt.

2. Fall v hat Söhne, aber keinen wie u in der Behauptung. Dann $G_\pi =$



In $G \setminus \{v\}$ bleiben die eingetragenen Rückwärtskanten, die nicht mit v inzident sind, stehen. Also ist $G \setminus \{v\}$ zusammenhängend. Also ist v kein Artikulationspunkt.

$$\boxed{\text{Zeige } A \Rightarrow B \text{ durch } \neg B \Rightarrow \neg A.}$$

□

Wie kann man Artikulationspunkte berechnen?

Wir müssen für alle Söhne in G_π wissen, wie weit es von dort aus zurück geht.

Definition 6.4(Low-Wert): *Sei DFS(G) gelaufen, also G_π vorliegend.*

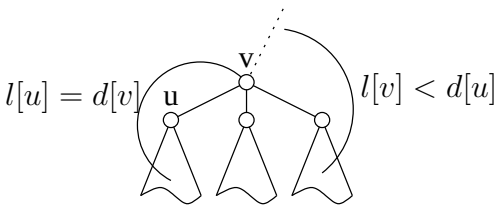
a) Für Knoten v ist die Menge von Knoten $L(v)$ gegeben durch $w \in L(v) \iff w = v$ oder w Vorgänger von v und es gibt eine Rückwärtskante von v oder dem Nachfolger zu w.

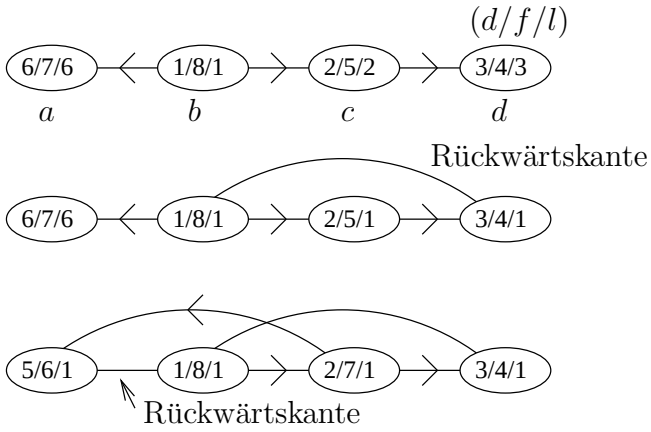
b) $l[v] = \min\{d[w] | w \in L(v)\}$ *ist der Low-Wert von* v*. (*$l[v]$ *hängt von Lauf von DFS(G) ab.)*

Folgerung 6.1: *Sei DFS(G) gelaufen und* v *nicht Wurzel von* G_π*.*

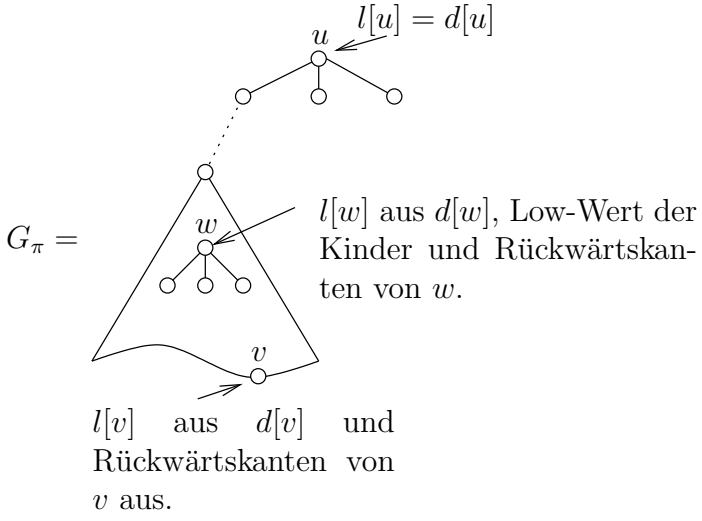
$$\boxed{v \text{ ist Artikulationspunkt} \iff v \text{ hat Sohn } u \text{ in } G_\pi \text{ mit } l[u] \geq d[v].}$$

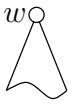
Beachte: $l[v] = d[v] \Longrightarrow v$ Artikulationspunkt. *Keine Äquivalenz!*





6.1 Berechnung von l[v]



Korrektheit: Induktion über die Tiefe des Teilbaumes 

.

6.2 Algorithmus (l-Werte)

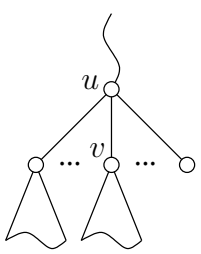
Wir benutzen eine modifizierte Version der Prozedur DFS-visit(u).

Prozedur MDFS-visit(u)

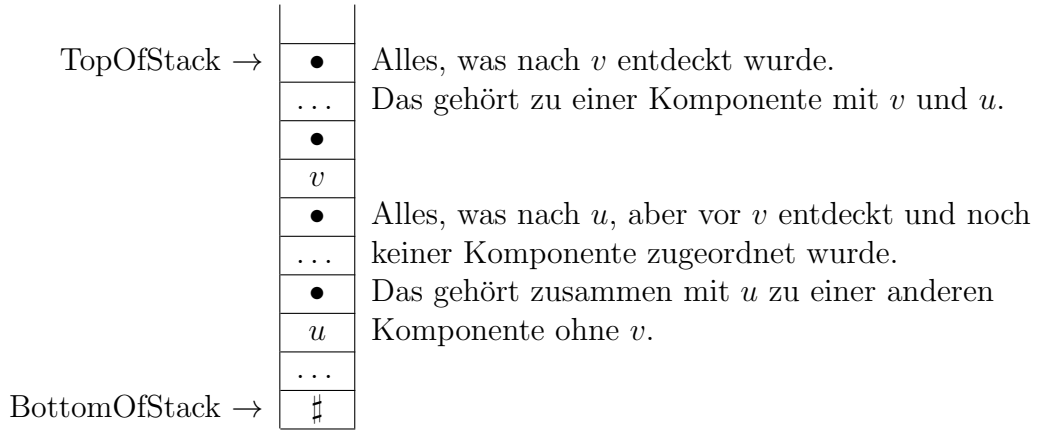
```
1  col[u] = grau;
2  d[u] = time;
3  l[u] = d[u];
4  time = time + 1;
5  foreach v ∈ Adj[u] do
6      if col[v] == weiß then
7          π[v] = u;
8          MDFS-visit(v);
           /* Kleineren Low-Wert des Kindes übernehmen. */
9          l[u] = min{l[u], l[v]};
10     end
11     if col[v] == grau und π[u] ≠ v then          /* Rückwärtskante */
           /* Merken, welche am weitesten zurück reicht. */
12         l[u] = min{l[u], d[v]};
13     end
14 end
15 col[u] = schwarz;
16 f[u] = time;
17 time = time + 1;
```

Damit können wir Artikulationspunkte in Linearzeit erkennen. Es bleiben die zweifachen Komponenten zu finden. Seien die l-Werte gegeben. Nun zweite Tiefensuche folgendermaßen durchführen:

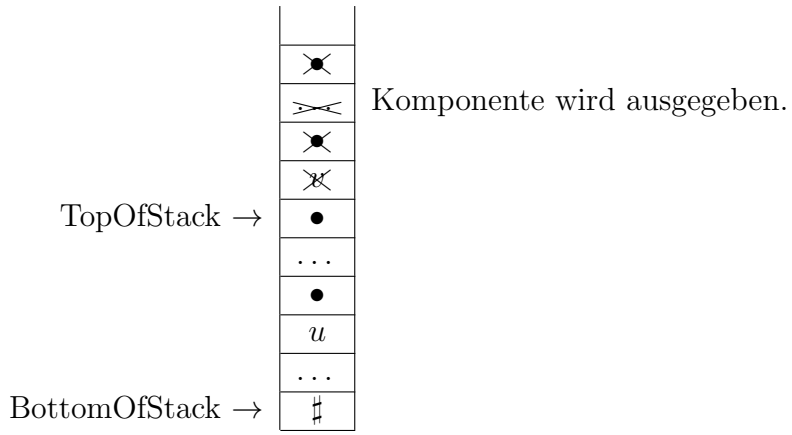
- Weiße Knoten werden vor dem Aufruf von DFS-visit auf einem Keller gespeichert. Wir befinden uns jetzt in DFS-visit(u). Der Tiefensuchbaum aus der ersten Tiefensuche sieht so aus:

$G_\pi =$ 

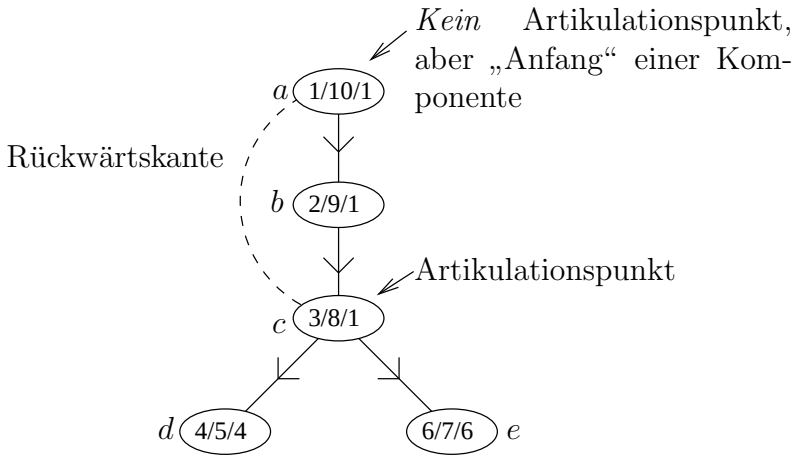
Sei jetzt $l[v] \geq d[u]$. Das heißt u ist ein Artikulationspunkt bezüglich v. Noch in DFS-visit(u), direkt nachdem DFS-visit(v) zurückgekehrt ist sieht unser Keller so aus: (Wenn unter v keine weiteren Artikulationspunkte sind.)



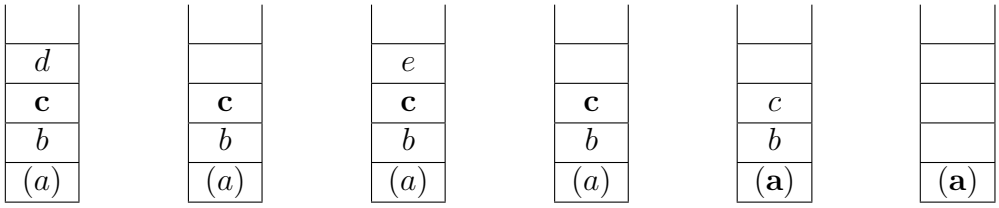
- Nachdem DFS-visit(v) zurückgekehrt ist, geben wir jetzt den Kellerinhalt bis einschließlich v aus. Dann macht die zweite Tiefensuche beim nächsten Kind von u weiter.



Beispiel 6.4:



DFS-visit(c), DFS-visit(d) fertig	Ausgabe: d, c	DFS-visit(c), DFS-visit(e) fertig	Ausgabe: e, c	DFS-visit(a), DFS-visit(b) fertig	Ausgabe: c, b, a



6.3 Algorithmus (Zweifache Komponenten)

Algorithmus 9: 2fache Komponenten(G)

Input : ungerichteter Graph $G = (V, E)$

```
   /* Modifizierte Tiefensuche für l-Werte */
1  MDFS(G);
   /* Alle Knoten wieder weiß. Die restlicher Werte bleiben
      erhalten. Insbesondere d und l werden noch gebraucht. */
2  foreach v ∈ V do
3  |  col[v] =weiß
4  end
5  S = ();                                  /* Keller initialisieren */
6  foreach v ∈ V do                 /* Dieselbe Reihenfolge wie bei 1. */
7  |  if col[v] == weiß then
8  |  |  NDFS-visit(v);
   |  |  /* Kann bei isolierten Knoten passieren. */
9  |  |  if Q ≠ () then
10 |  |  |  Knoten in Q ausgeben;
11 |  |  |  Q = ();
12 |  |  end
13 |  end
14 end
```

Prozedur NDFS-visit(u)

```
1  col[u] = grau;
2  foreach v ∈ Adj[u] do
3      if col[v] == weiß then
4          push(Q, v);                                  /* v auf Keller */
5          NDFS-visit(v);
6          if l[v] ≥ d[u] then                /* u ist Artikulationspunkt */
7              Solange Knoten von Q entfernen und Ausgeben, bis v
               ausgegeben;
8              Knoten u auch mit ausgeben;
9          end
10     end
11 end
12 col[u] = schwarz;
```

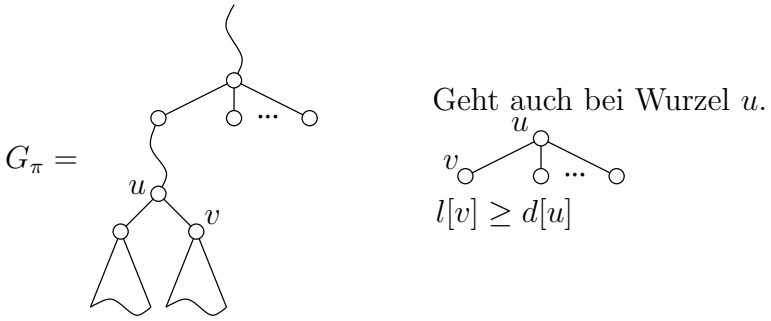
Beweis. Korrektheit induktiv über $s = \#$Zweifache Komponenten von G.

Induktionsanfang: $s = 1$ Ausgabe nur am Ende, da kein Artikulationspunkt. Also korrekt.

Induktionsschluss: Gelte Behauptung für alle(!) Graphen mit $s \geq 1$ *zweifachen Komponenten*. Zeige dies für G mit $s + 1$ Komponenten.

Wir betrachten die 2. Tiefensuche des Algorithmus. Die l-Werte stimmen also bereits, wegen der Korrektheit der 1. Tiefensuche.

Wir betrachten den Baum G_π, welcher der gleiche wie bei der 1. Tiefensuche und 2. Tiefensuche ist.



Sei NDFS-visit(v) der *erste* Aufruf, nach dem die Ausgabe erfolgt. Dann ist $l[v] \geq d[u]$.

Ist $l[v] = d[v]$, dann ist v Blatt (sonst vorher Ausgabe) und v, u wird ausgegeben.

Ist $l[v] > d[u]$, dann Ausgabe des Kellers bis v und zusätzlich u. Damit wird eine zweifache Komponente ausgegeben. Alles was nach v entdeckt wurde, fällt

in $G \setminus \{u\}$ ab. (Mehr kann nicht dazu gehören, weniger nicht, da bisher kein Artikulationspunkt).

Nach Ausgabe der Komponente liegt eine Situation vor, die in DFS-visit(u) auftritt, wenn wir den Graphen betrachten, in dem die Kante $\{u, v\}$ und die ausgegegenen Knoten außer u gelöscht sind.

Auf diesem ist die Induktionsvoraussetzung anwendbar und der Rest wird richtig ausgegeben.

□