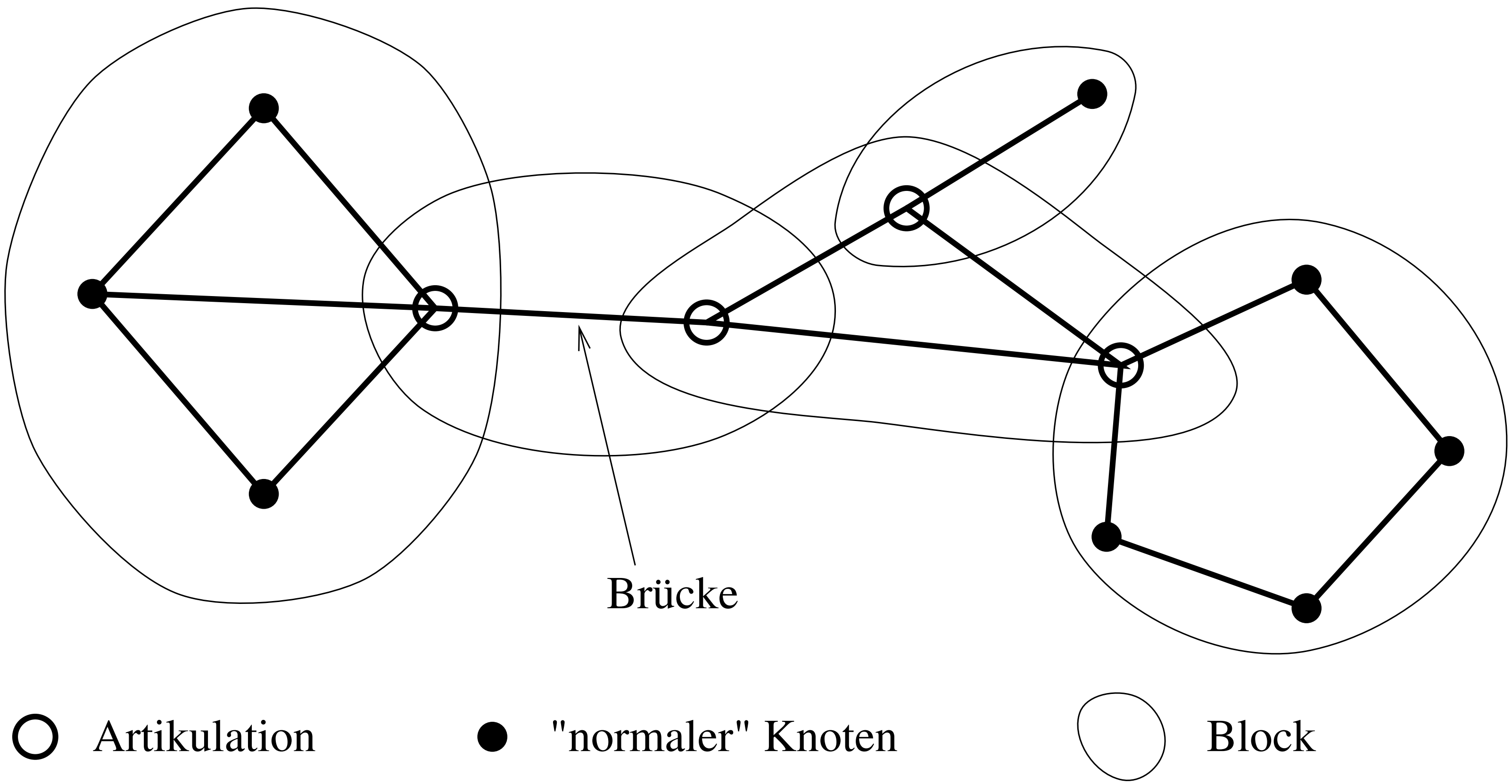


Brücke
Artikulation
"normaler" Knoten
Block

<u>Definition</u> Sei G ein zusammenhängender Graph. Ein Knoten $v \in V$ heißt ***Artikulationsknoten*** (oder ***trennender Knoten***), falls $G - v$ nicht zusammenhängend ist.

Ein ***Block*** in einem Graphen G ist ein maximaler zusammenhängender Subgraph von G auf mindestens zwei Knoten, der keinen trennenden Knoten enthält.

Ein Graph G auf mehr als $k + 1$ Knoten heißt k-*fach zusammenhängend*, falls es zwischen je zwei Knoten s und t in G mindestens k kreuzungsfreie s-t Pfade gibt.

<u>Beobachtungen:</u>

$a)$ Jede Kante ist in einem Block enthalten. Ggf. bildet sie einen eigenen Block.

$b)$ Ein Block besteht entweder aus genau einer Kante, oder er ist 2-fach zusammenhängend.

$c)$ Ein Kreis ist stets ganz in einem Block enthalten.

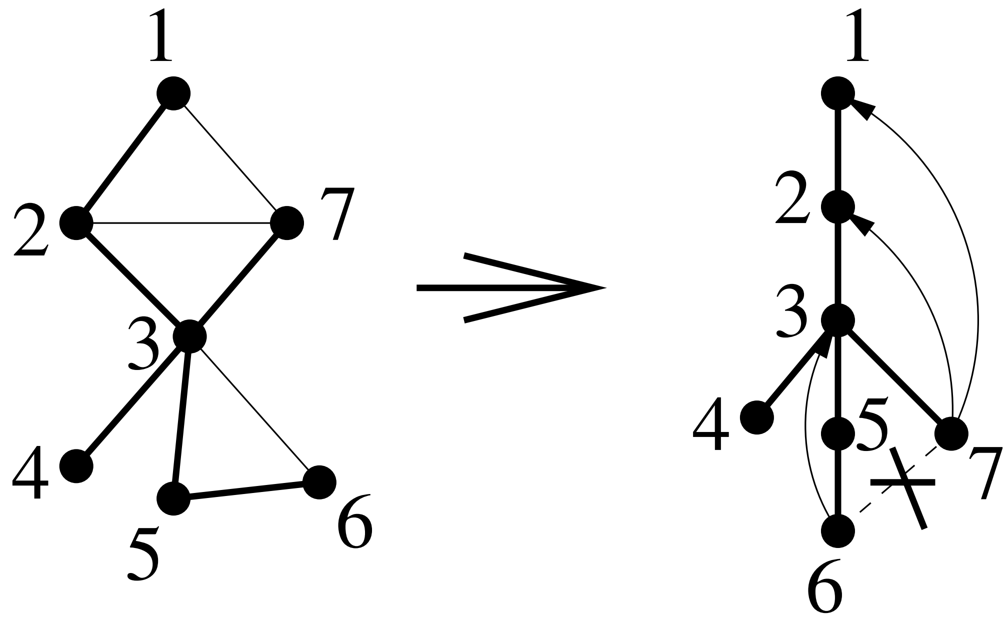
<u>Lemma</u> Die Blöcke bilden eine Partition der Kantenmenge. (D. h., jede Kante ist in genau einem Block enthalten.)

<u>Lemma</u> $a)$ Zwei verschiedene Blöcke eines Graphen haben *höchstens einen* Knoten gemeinsam.

$b)$ Ein Knoten gehört genau dann zu mehreren Blöcken, wenn er eine Artikulation ist.

Beobachtung: Kanten des Graphen zerfallen bei DFS in zwei Klassen.

- Baumkanten: $\{pred[v], v\}$
- Rückwärtskanten: $\{x, y\}$, wobei y auf dem $x - s$ Pfad von x zur Wurzel s liegt



,,Querkanten" kann es nicht geben.

Idee: Modifiziere den Algorithmus so, dass zusätzlich folgender Wert berechnet wird:

$$\texttt{low}[v] := min\{\,\texttt{dfs}[w] \mid w \text{ ist von } v \text{ aus über } \geq 0 \text{ Baumkanten abwärts und } \leq 1 \text{ Rückwärtskanten aufwärts erreichbar}\}$$

$\textsc{Blockzerlegung}(G)$

Initialisiere einen Stack;

$\forall v \in V$: $\texttt{pred}[v] := nil$;

$k := 0$; wähle einen beliebige Knoten $s \in V$;

$\textsc{Scan}(s)$;

procedure $\textsc{Scan}(v)$

$k{+}{+}$; $\texttt{dfs}[v] := k$; $\texttt{low}[v] := k$;

for all $w \in \Gamma(v)$, $w \neq \texttt{pred}[v]$ **do**

    if $\texttt{pred}[w] = nil$ **then**

        // $\{v, w\}$ ist Baumkante

        $\texttt{pred}[w] := v$; $\textsc{Push}(\{v, w\})$;

        $\textsc{Scan}(w)$;

        if $\texttt{low}[w] \geq \texttt{dfs}[v]$ **then**

            // v ist Artikulationsknoten, falls $v \neq s$ oder

            // v mind. 2 Söhne hat;

            Poppe Kanten vom Stack bis einschl. Kante $\{v, w\}$

            // ... diese bilden einen Block!

        $\texttt{low}[v] := \min\{\texttt{low}[v], \texttt{low}[w]\}$;

    else if $\texttt{dfs}[w] < \texttt{dfs}[v]$ **then**

        // $\{v, w\}$ ist Rückwärtskante

        $\textsc{Push}(\{v, w\})$;

        $\texttt{low}[v] := \min\{\texttt{low}[v], \texttt{dfs}[w]\}$;

Lemma:

Sei G ein zusammenhängender Graph und T ein DFS-Baum von G. Dann gilt folgendes.

Ein Knoten $v \in V$ ist genau dann eine Artikulation, wenn

- v Wurzel von T ist und mindestens zwei Söhne hat, *oder*
- es einen Sohn x von v in T gibt, so daß von *keinem* Knoten im Unterbaum T_x von T mit Wurzel x eine Rückwärtskante zu einem (echten) Vorgänger von v ausgeht.

Bestimmung stark zusammenhängender Komponenten

Idee: Verwende eine modifizierte Tiefensuche

Genauer: Modifiziere die Tiefensuche so, dass zusätzlich folgender Wert berechnet wird:

$$\texttt{lowlink}[v] := min\{\,\texttt{dfs}[w] \mid w \text{ ist von } v \text{ aus über } \geq 0 \text{ Baumkanten abwärts und } \leq 1 \text{ Rückwärtskanten aufwärts erreichbar}$$
$$\textit{oder}$$
$$w \text{ ist von } v \text{ aus über } \geq 0 \text{ Baumkanten abwärts und } \leq 1 \text{ Querkanten in eine Zusammenhangskomponente, deren Wurzel ein echter Vorfahre von } v \text{ ist, erreichbar}\}$$