Starke Zusammenhangskomponente

Programmieren und Software-Engineering
Theorie

2. September 2025

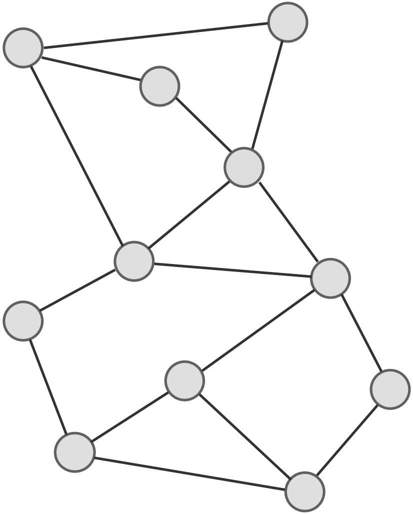
WH: Zusammenhangskomponenten

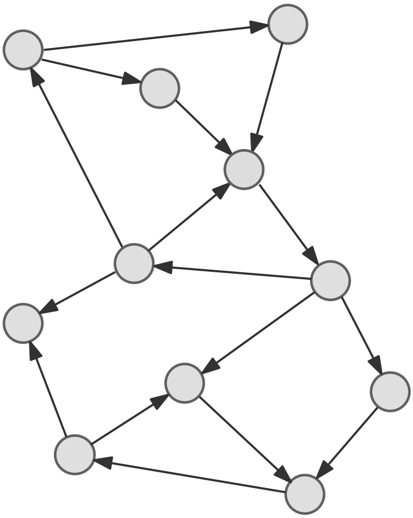
Definition 1 (Zusammenhangskomponenten)

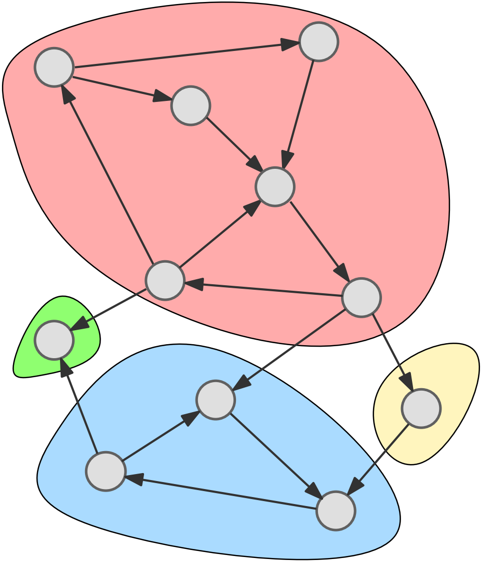
In einem ungerichteten Graphen G ist eine *Zusammenhangskomponente* ein maximaler, zusammenhängender Teilgraph. Zwei Knoten u und v sind in der selben Zusammenhangskomponente, genau dann wenn $u \rightsquigarrow v$. [a]

[a] $u \rightsquigarrow v$ bedeutet, dass v von u aus erreichbar ist. Im ungerichteten Graphen impliziert dies $v \rightsquigarrow u$.

Wie kann der Begriff der *Zusammenhangskomponente* auf gerichtete Graphen übertragen werden?







Zusammenhang in gerichteten Graphen

Definition 2 (Schwacher Zusammenhang)

Man spricht von *schwachem Zusammenhang* in einem gerichteten Graphen, wenn der zugrunde liegende ungerichtete Graph ("Schatten") zusammenhängend ist.

Definition 3 (Starke Zusammenhangskomponente)

Eine *Starke Zusammenhangskomponente* ist eine maximale Teilmenge an Knoten $U \subseteq V$ mit der Eigenschaft, dass für alle Paare an Knoten $u, v \in U$ gilt, dass sowohl $u \rightsquigarrow v$ als auch $v \rightsquigarrow u$.

Definition 4 (Transponierter Graph G^T)

Sei $G^T = (V, A^T)$ der Graph der aus dem gerichteten Graphen $G = (V, A)$ entsteht indem alle gerichteten Kanten "umgedreht" werden, also $A^T = \{(j, i) \mid (i, j) \in A\}$.

Aufruf der Tiefensuche DFS(G):

- Im ersten Aufruf werden nicht unbedingt alle Knoten erreicht
- Die Tiefensuche wird dann so lange für die verbleibenden (unbesuchten) Knoten aufgerufen, bis alle Knoten besucht sind.
- Auch für erneuten Aufruf gilt: besuchte Knoten werden nicht erneut besucht
- Wenn zu jedem Knoten eine gerichtete Kante vom Vorgänger auf diesen Knoten gespeichert wird, entsteht **Tiefensuch-Wald** (mehrere gerichtete Bäume)

Algorithmus zur Berechnung der starken Zusammenhangskomponente

Algorithmus von Kosaraju-Shahir

1. Aufruf von DFS(G) $\Rightarrow \tau_f(v)$
2. Berechne G^T
3. Aufruf von DFS(G^T) für Knoten $v \in V$ in absteigender Reihenfolge bezüglich $\tau_f(v)$ aus Schritt (1).
4. Die Knotenmengen die in einem Aufruf der DFS in Schritt (3) gefunden werden entsprechen den starken Zusammenhangskomponenten.

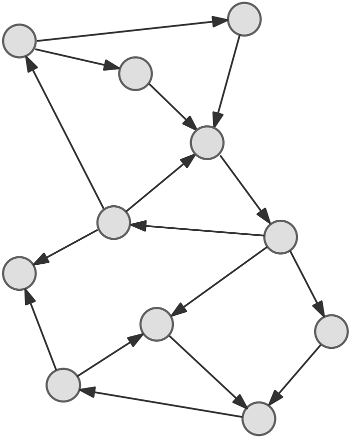


Abbildung: Beispiel: Bestimmung der starken Zusammenhangskomponenten

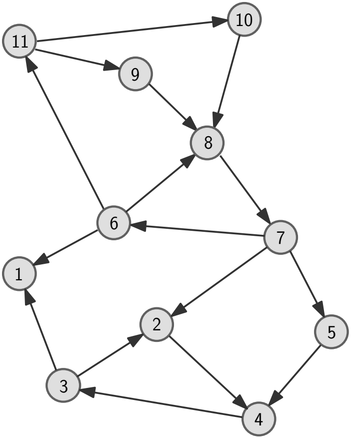


Abbildung: Die Werte in den Knoten entsprechen den Fertigstellungszeiten τ_f (Bem. τ_d's werden hier vernachlässigt)

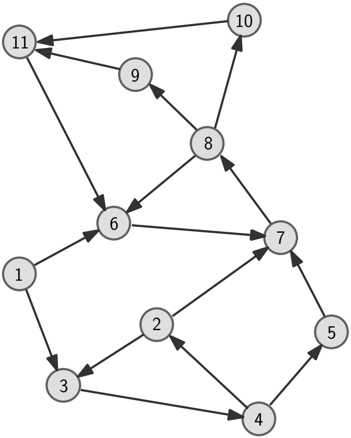


Abbildung: Nach dem ersten Aufruf der DFS wird der transponierte Graph G^T von G berechnet.

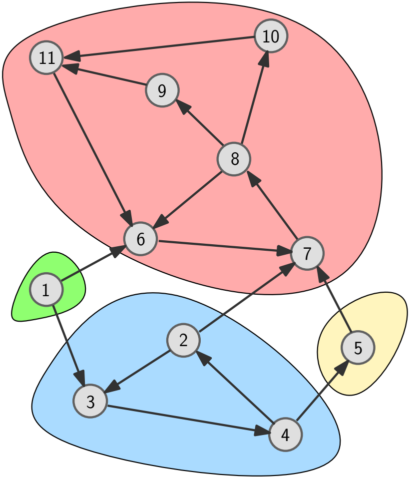


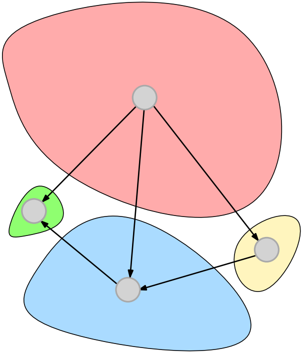
Abbildung: Die DFS Aufrufe in Schritt (3) liefern als Ergebnis die Starken Zusammenhangskomponenten von G^T (und somit von G).

Lemma 5

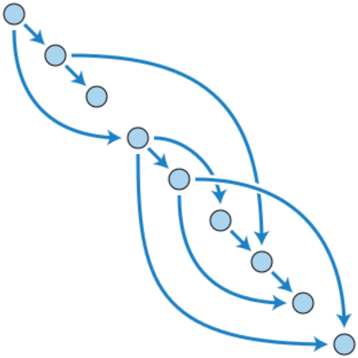
Sei $G' = (V', E')$ der Graph der daraus entsteht indem in G jede Starke-Zusammenhangs-Komponente (SZK) in einen einzelnen Knoten kontrahiert wird. G' ist dann ein gerichteter azyklischer Graph, (engl. directed acyclic graph, **DAG***).*

Beweis:

- Angenommen es gibt einen Kreis $C_1, C_2, \ldots, C_n$ mit $C_i \in V'$
- $\Rightarrow$ Dann existiert eine gerichtete Kante $a = (i, j)$ mit $i \in C_k$ und $j \in C_{k+1}$ (und jeweils $i \in C_n$ und $j \in C_1$) für $1 \leq k \leq n - 1$
- Es existiert ein Pfad P von jedem Knoten in C_i zu jedem Knoten in C_j für alle $i, j \in \{1 \ldots n\}$.
- Alle $v \in V(\bigcup_{i \in \text{Kreis}} C_i)$ gehören zur selben SZK. $\square$



- Ein DAG enthält keinen gerichteten Kreis.
- Die Knoten können so angeordnet werden, dass alle Kanten von links nach rechts verlaufen.
- Diese Anordnung wird auch *topologische Sortierung* genannt.
- $\Rightarrow$ **Jeder DAG enthält mindestens einen Knoten** v **mit** $d^+(v) = 0$**, und mindestens einen Knoten** v **mit** $d^-(v) = 0$.

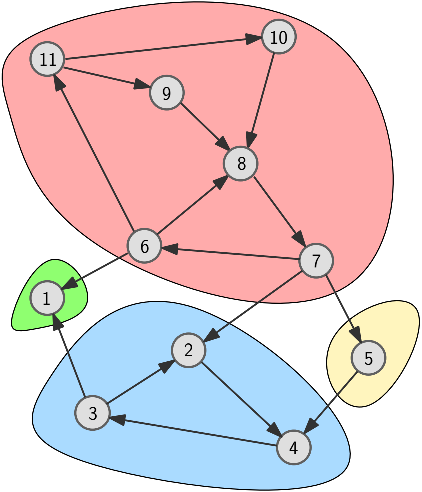


Lemma 6

Sei C eine SZK von G ohne auslaufenden Kanten. Ein DFS-Aufruf für einen Knoten $v \in C$ besucht **genau alle** *Knoten $u \in C$.*

Beweis:

- Sei v ein beliebiger Knoten in C.
- $\forall u \in C : v \rightsquigarrow u$.
- Da C keine auslaufenden Kanten hat, sind keine weiteren Knoten erreichbar. □



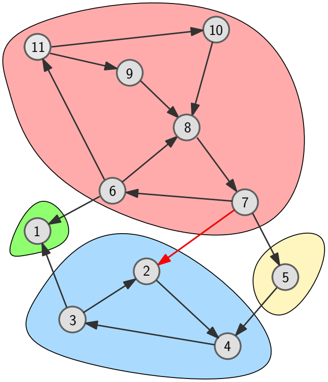
Lemma 7

Seien C_1 und C_2 zwei SZK von G. Weiters sei $a = (i, j)$ eine gerichtete Kante von einer Komponente in die andere, also $i \in C_1, j \in C_2$. Sei v^* der erste Knoten in C_1 der von DFS besucht wird. Dann gilt: $\tau_f(v^*) > \tau_f(v_k)\ \forall v_k \in C_2$.

Beweis:

case 1: DFS wird für einen Knoten $v \in C_1$ aufgerufen bevor sie für irgend einen Knoten in C_2 aufgerufen wird.

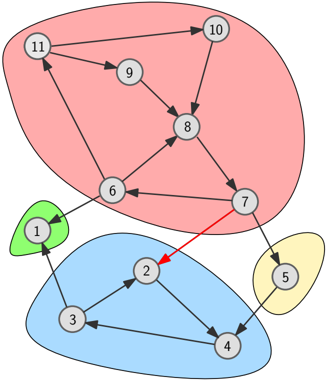
- Aufgrund der Annahme sind alle Knoten in C_1 und C_2 vom Knoten v aus erreichbar.
- DFS(v) ist fertig wenn alle Knoten in C_1 und C_2 abgeschlossen wurden.
- $\tau_f(v^*) > \tau_f(v_k) \quad \forall v_k \in C_2$.



Proof:

case 2: DFS wird für einen Knoten $v \in C_2$ bevor es für irgend einen anderen Knoten in C_1 aufgerufen wird.

- Es gibt keinen Weg von einem Knoten in C_2 zu einem in C_1.
- DFS ist für alle Knoten in C_2 abgeschlossen wenn es für den ersten Knoten in C_1 aufgerufen wird.
- $\tau_f(v^*) > \tau_f(v_k) \;\; \forall v_k \in C_2$.



Lemma 8

Der Knoten v mit $\max(\tau_f(v))$ in G gehört zur SZK ohne einlaufende Kanten.

Beweis: folgt direkt aus Lemma 7 □

Lemma 8

Der Knoten v mit $\max(\tau_f(v))$ in G gehört zur SZK ohne einlaufende Kanten.

Beweis: folgt direkt aus Lemma 7 □

Lemma 9

Die Knotenmengen der SZKn von G entsprechen jenen in G^T.

Beweis: Wir betrachten zwei Knoten $u, v \in G$ in der selben SZK in G.
$u \rightsquigarrow v \wedge v \rightsquigarrow u$
$\Leftrightarrow u \rightsquigarrow v \wedge v \rightsquigarrow u$ in G^T
u und v sind somit in der selben SZK in G^T. □

Die zusammenhängenden Komponenten des resultierenden Tiefensuch-Waldes in G^T (Schritt 3 des Algorithmus) entsprechen den SZK in G.

Die zusammenhängenden Komponenten des resultierenden Tiefensuch-Waldes in G^T (Schritt 3 des Algorithmus) entsprechen den SZK in G.

Beweis:

- *Teil 1:* Erster Aufruf der DFS in G^T:
  Es folgt aus Lemma 8, dass DFS für die SZK ohne auslaufende Kanten (in G^T) aufgerufen wird, $\Rightarrow$ DFS besucht genau alle Knoten dieser SZK.

- *Teil 2:* weitere DFS-Aufrufe in G^T:
  Sei v der Knoten für den DFS aufgerufen wird, und C_v die zugehörige SZK. Alle Knoten in C_v sind erreichbar und werden von diesem DFS-Aufruf besucht. Sei weiters $v \rightsquigarrow w$, $w \notin C_v$ (sondern in C_w). Es wird nun gezeigt, dass w *nicht* in diesem DFS-Aufruf besucht wird.

  In G gilt, dass $w \rightsquigarrow v$. Wenn w schon in einem vorangegangenen DFS-Aufruf besucht wurde, wird es nicht erneut besucht. Somit betrachten wir die (kritische) Situation, dass w noch unbesucht ist.

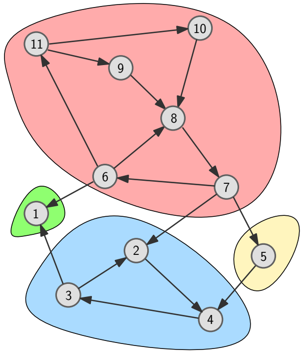
  Es muss somit gelten, dass $\max_{w' \in C_w}(\tau_f(w')) < \tau_f(v)$. (Anderenfalls wären ja alle Knoten von C_w schon zuvor besucht worden).

  Es folgt somit, dass der DFS-Aufruf in G schon für alle Knoten $u \in C_w$ abgeschlossen war, bevor v abgeschlossen wurde.

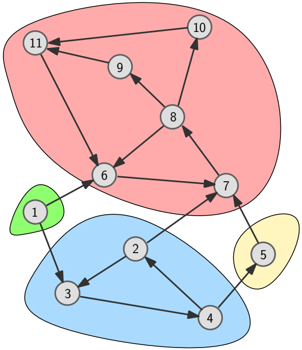
  Da jedoch $u \rightsquigarrow v \ \forall u \in C_w$in G folgt, dass ein Knoten $u \in C_w$ existieren muss, für den $\tau_f(u) > \tau_f(v)$ gilt, was ein Widerspruch zu der vorigen Aussage ist.

$\square$

G :



G^T:



Anmerkung: Wenn also eine Kante in G^T von C_v nach C_w führt, müssen alle Knoten in C_w *nach* jenen aus C_v fertiggestellt worden sein, da ja in G in diesem Fall eine Kante von C_w nach C_v verlaufen ist. Somit müssen diese Knoten im Schritt 3 schon zuvor fertiggestellt worden sein (da ja die Aufrufe in absteigender Reihenfolge der Fertigstellungszeiten aus Schritt 1 erfolgt.